МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН
Satbayev University
Институт кибернетики и информационных технологий
Кафедра кибербезопасность, обработка и хранение информации

Әбдірайым Дастан Беркінбайұлы

Разработка WEB-приложения «Студент»

ДИПЛОМНАЯ РАБОТА

Специальность 5В070300 – Информационные системы

Алматы, 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН

Satbayev University

Институт кибернетики и информационных технологий

Кафедра кибербезопасность, обработка и хранение информации

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой КБОиХИ
канд. техн. наук, доцент
_____ Н. А. Сейлова
«____» _____ 20__ г.

ДИПЛОМНАЯ РАБОТА

На тему: “Разработка WEB-приложения «Студент»”

Специальность 5В070300 – Информационные системы

Выполнил**:** Әбдірайым Д. Б.

Научный руководитель
канд. техн. наук, ассоц. проф.
_____ Жумагалиев Б. И.
«____» _____ 20__ г.

Алматы, 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Satbayev University

Институт кибернетики и информационных технологий

Кафедра кибербезопасность, обработка и хранение информации

5В070300 – Информационные системы

УТВЕРЖДАЮ
Заведующий кафедрой КБОиХИ
канд. техн. наук, доцент
_____ Н. А. Сейлова
«____» _____ 20__ г.

ЗАДАНИЕ
на выполнение дипломной работы

Обучающемуся: Әбдірайым Дастан Беркінбайұлы
Тема: Разработка WEB-приложения «Студент»
Утверждена *приказом Ректора Университета № 762-б от «27» января 2020 г.*
Срок сдачи законченной работы «*27» мая 2020 г.*
Исходные данные к дипломной работе: результаты преддипломной практики, обзор предметной области, сбор теоретического материала.
Краткое содержание дипломной работы:
а) Обзор решений и технологий электронного обучения;
б) Выбор средств проектирования и разработки системы;
в) Проектирование и разработка системы.
Рекомендуемая основная литература: 10 наименований

ГРАФИК

подготовки дипломной работы (проекта)

Наименование разделов, перечень разрабатываемых вопросов	**Сроки представления научному руководителю**	**Примечание**
Обзор решений и технологий электронного обучения	28.02.2020 г.	
Выбор средств проектирования и разработки системы	30.03.2020 г.	
Проектирование и разработка системы	23.04.2020 г.	

Подписи

консультантов и нормоконтролера на законченную дипломную работу (проект) с указанием относящихся к ним разделов работы (проекта)

Наименование разделов	**Консультанты, Ф.И.О. (уч. степень, звание)**	**Дата подписания**	**Подпись**
“Разработка WEB-приложения «Студент»”	Жумагалиев Б. И., кандидат технических наук, ассоциированный профессор		
Нормоконтролер	Бауыржан М. Б., тьютор		
Программная часть	Кабдуллин М. А., ассистент		

Научный руководитель: _____ Жумагалиев Б. И.

Задание принял к исполнению обучающийся _____ Әбдірайым Д. Б.

Дата "27" января 2020 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН

Satbayev University

Отзыв научного руководителя

Дипломная работа

Әбдірайым Дастан Беркінбайұлы

5В070300 – Информационные системы

Тема: Разработка WEB-приложения «Студент»

Дипломная работа представляет собой выпускную квалификационную работу по специальности 5В070300 – «Информационные системы». Пояснительная записка состоит из введения, 3 глав, заключения, списка использованных источников и приложения.

Автор дипломной работы поставленные задачи полностью выполнил и показал владение современными технологиями в предметной области.

Дипломная работа выполнена на достаточном профессиональном уровне и содержит все необходимые сведения для такого рода работ. К замечаниям следует отнести незначительные стилистические ошибки.

Считаю, что дипломная соответствует требованиям предъявляемым к выпускным квалификационным работам по специальности 5В070300 – «Информационные системы». Автор работы Әбдірайым Дастан Беркінбайұлы заслуживает присвоения академической степени бакалавра.

Научный руководитель

Ассоц. проф., канд. техн. наук Жумагалиев Б. И.

«___» _____ 2020 г.

Протокол анализа Отчета подобия Научным руководителем

Заявляю, что я ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

Автор: Әбдірайым Д.Б.

Название: Разработка WEB-приложения «Студент»

Координатор:Биржан Жумагалиев

Коэффициент подобия 1:0,9

Коэффициент подобия 2:0,6

Замена букв:65

Интервалы:0

Микропробелы:0

Белые знаки: 0

После анализа Отчета подобия констатирую следующее:

☐ обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, признаю работу самостоятельной и допускаю ее к защите;

☐ обнаруженные в работе заимствования не обладают признаками плагиата, но их чрезмерное количество вызывает сомнения в отношении ценности работы по существу и отсутствием самостоятельности ее автора. В связи с чем, работа должна быть вновь отредактирована с целью ограничения заимствований;

☐ обнаруженные в работе заимствования являются недобросовестными и обладают признаками плагиата, или в ней содержатся преднамеренные искажения текста, указывающие на попытки сокрытия недобросовестных заимствований. В связи с чем, не допускаю работу к защите.

Обоснование:

……

…… ……

Дата *Подпись Научного руководителя*

Протокол анализа Отчета подобия

заведующего кафедрой / начальника структурного подразделения

Заведующий кафедрой / начальник структурного подразделения заявляет, что ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

Автор: Әбдірайым Д.Б.

Название: Разработка WEB-приложения «Студент»

Координатор: Биржан Жумагалиев

Коэффициент подобия 1:0,9

Коэффициент подобия 2:0,6

Замена букв:65

Интервалы:0

Микропробелы:0

Белые знаки:0

После анализа отчета подобия заведующий кафедрой / начальник структурного подразделения констатирует следующее:

☐ обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, работа признается самостоятельной и допускается к защите;
☐ обнаруженные в работе заимствования не обладают признаками плагиата, но их чрезмерное количество вызывает сомнения в отношении ценности работы по существу и отсутствием самостоятельности ее автора. В связи с чем, работа должна быть вновь отредактирована с целью ограничения заимствований;
☐ обнаруженные в работе заимствования являются недобросовестными и обладают признаками плагиата, или в ней содержатся преднамеренные искажения текста, указывающие на попытки сокрытия недобросовестных заимствований. В связи с чем, работа не допускается к защите.

Обоснование:

……
……
……
……
……

…… ……

Дата *Подпись заведующего кафедрой /*

начальника структурного подразделения

Окончательное решение в отношении допуска к защите, включая обоснование:

……
……
……
……
……

…… ……

Дата *Подпись заведующего кафедрой /*

начальника структурного подразделения

StrikePlagiarism.com

КАЗ УТУ

DATA RAPORTU: 2020-05-27 07:25:32

TYTUŁ:
Разработка WEB-приложения «Студент»

AUTOR/ZY:
Әбдірайым Д.Б.

PROMOTOR:
Биржан Жумагалиев

JEDNOSTKA ORGANIZACYJNA:
ИКиИТ

DATA WGRANIA DOKUMENTU:
2020-05-27 07:24:22

LICZBA SPRAWDZEŃ:
1

POMINIĘTE STRONY WWW:

Metryka podobieństw

Należy pamiętać, że wysokie wartości Współczynników nie oznaczają automatycznie plagiatu. Raport powinien zostać przeanalizowany przez kompetentną / upoważnioną osobę. Wyniki są uważane za wymagające szczegółowej analizy, jeśli WP 1 wynosi ponad 50%, a WP 2 ponad 5%.

0,95% WP1
% wyrazów odnalezionych w ramach 5-wyrazowych fraz bez BAP (Baza Aktów Prawnych)

0,56% WP2
% wyrazów odnalezionych w ramach 25-wyrazowych fraz bez BAP

0,96% CYT
% tekstu objęta cudzysłowem

25
Długość frazy dla WP 2

6236
Liczba słów

51622
Liczba znaków

Alerty

W tej sekcji znajdują się statystyki występowania w tekście zabiegów edytorskich, które mogą mieć na celu zaburzenie wyników analizy. Niewidoczne dla osoby zapoznającej się z treścią pracy na wydruku lub w pliku, wpływają na frazy porównywane podczas analizy tekstu (poprzez celowe błędy pisowni) w celu ukrycia zapożyczeń lub obniżenia wyników w Raporcie podobieństwa. Należy ocenić, czy zaznaczone wystąpienia wynikają z uzasadnionego formatowania tekstu (nadwrażliwość systemu), czy są celową manipulacją.

Znaki z innego alfabetu liczba znaków z innych alfabetów - mogą imitować litery z alfabetu właściwego dla języka pracy powodując błędy pisowni wyrazów w tekście, należy zweryfikować zasadność użycia	65	pokaż w tekście
Rozstrzelenia liczba zastosowań zwiększenia odległości pomiędzy literami - należy sprawdzić czy nie imitują spacji łącząc wyrazy	0	pokaż w tekście
Mikrospacje liczba spacji o zerowej długości - należy sprawdzić czy nie wywołują nieprawidłowego podziału wyrazów w tekście	0	pokaż w tekście
Białe znaki liczba znaków o białym kolorze czcionki - należy sprawdzić czy nie zastępują spacji powodując złączenie wyrazów (na Raporcie kolor liter jest zmieniany na czarny w celu ich uwidocznienia)	0	pokaż w tekście

Aktywne listy podobieństw

Uwagi wymagają szczególnie fragmenty, które zostały włączone do WP 2 (zaznaczone pogrubieniem). Użyj linku "Pokaż w tekście" i zobacz, czy są to krótkie frazy rozproszone w dokumencie (przypadkowe podobieństwa), skupione wokół siebie (parafraza) lub obszerne fragmenty bez wskazania źródła (tzw. "kryptocytaty").

10 najdłuższych fragmentów (0,95 %)

Dziesięć najdłuższych fragmentów znalezionych we wszystkich dostępnych zasobach.

LP	TYTUŁ LUB ADRES URL ŹRÓDŁA (NAZWA BAZY)	AUTOR/ZY	LICZBA IDENTYCZNYCH SŁÓW	
1	Хван А- Тусупова Б.docx *Satbayev University (ИКиИТ)*	Хван А	35	0,56 %
2	Хван А- Тусупова Б.docx *Satbayev University (ИКиИТ)*	Хван А	12	0,19 %
3	Разработка онлайн – курса в рамках модели открытого образования *NARXOZ (NEU) (Кафедра Технологии и экология)*	Болатбек Бакытжан	12	0,19 %

z bazy RefBooks (0,00 %)

Wszystkie fragmenty znalezione w bazie RefBooks, która zawiera ponad 3 miliony tekstów od redaktorów i autorów.

LP	TYTUŁ	AUTOR/ZY	LICZBA IDENTYCZNYCH SŁÓW (LICZBA FRAGMENTÓW)

NIE WYKRYTO ZAPOŻYCZEŃ

z bazy macierzystej (0,75 %)

Wszystkie fragmenty znalezione w bazie danych Twojej instytucji.

LP	TYTUŁ	AUTOR/ZY	DATA INDEKSOWANIA	IDENTYCZNYCH SŁÓW (FRAGMENTÓW)	
1	Хван А- Тусупова Б.docx *Satbayev University (ИКиИТ)*	Хван А	2017-05-15	47 (2)	0,75 %

z Programu Wymiany Baz (0,19 %)

Wszystkie fragmenty znalezione w bazie danych innych instytucji.

LP	TYTUŁ *NAZWA BAZY*	AUTOR/ZY	DATA INDEKSOWANIA	LICZBA IDENTYCZNYCH SŁÓW (LICZBA FRAGMENTÓW)	
1	Разработка онлайн – курса в рамках модели открытого образования *NARXOZ (NEU) (Кафедра Технологии и экология)*	Болатбек Бакытжан	2017-12-07	12 (1)	0,19 %

z Internetu (0,00 %)

Wszystkie fragmenty znalezione w globalnych zasobach Internetu o otwartym dostępie.

LP	ADRES URL ŹRÓDŁA	IDENTYCZNYCH SŁÓW (FRAGMENTÓW)

NIE WYKRYTO ZAPOŻYCZEŃ

АҢДАТПА

Бұл дипломдық жұмыстың мақсаты - жоғары оқу орындарында электрондық оқытуды одан әрі дамытуды қамтамасыз ететін «Студент» веб-қосымшасын құру. Осы жұмыс барысында зерттеу аймағына шолу және талдау жүргізілді. Қосымшаларды әзірлеу екіге бөлінді: сервер және клиент. Сервер жағы Spring Framework, Hibernate және Maven сияқты құралдарды қолдана отырып орындалды, ал клиент - Angular, NPM, HTML және CSS. Ақпараттық жүйені жобалау ER және UML диаграммаларын қолдану арқылы жүзеге асырылды.

АННОТАЦИЯ

Целью данной дипломной работы является разработка веб-приложения «Студент», которое обеспечит дальнейшее развитие электронного обучения в высших учебных заведениях. В ходе данной работы был проведен обзор и анализ предметной области. Разработка приложения разделилась на две части: серверную и клиентскую. Серверная часть была реализована с использованием таких инструментов, как Spring Framework, Hibernate и Maven, в то время как клиентская часть – с помощью Angular, NPM, HTML и CSS. Проектирование информационной системы было осуществлено с применением ER и UML-диаграмм.

SUMMARY

The purpose of this graduate work is to develop «Student» web application that will further develop e-learning in higher education institutions. During this work, a review and analysis of the subject area was carried out. Application development was divided into two parts: server and client. The server side was implemented using tools such as the Spring Framework, Hibernate and Maven, while the client side was using Angular, NPM, HTML and CSS. The design of the information system was carried out using ER and UML diagrams.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На сегодняшний день, информационные технологии значительно облегчают человеческое существование и во многом считаются продолжением его деятельности. Если раньше ИТ не выходили за рамки вычислений сложных математических алгоритмов, то сейчас трудно представить нашу жизнь без них.

Одной из сфер, на которую повлияли ИТ, является образование. В учебных заведениях вводят электронные расписания, по которым пользователи узнают информацию о месте проведений занятий, преподаватели размещают лекции в интернете, что позволяет практиковать самостоятельное изучение тем. Таким образом, различие между очной и заочной формой обучения все чаще сокращается.

Развивать электронное образование помогает направление в области компьютерных вычислений, которое считается актуальным и по сей день – разработка веб-приложений.

Веб-приложение – это прикладное программное обеспечение, логика которого распределена между клиентом и сервером, а обмен информацией происходит по сети. Данный вид приложений относится к кроссплатформенным службам, так как не зависит от операционной системы пользователя, что является большим преимуществом над десктопными приложениями.

Целью данной дипломной работы является обеспечение развития электронного обучения в стране посредством создания веб-приложения «Студент», позволяющее создавать индивидуальное расписание, проходить онлайн-тесты и иметь доступ к электронной библиотеке.

Для достижения поставленной цели требуется выполнить следующие действия:

– осуществить анализ решений и технологий в сфере e-learning;

– сделать постановку задачи, а также выбор инструментов разработки;

– произвести проектирование программного обеспечения;

– разработать и протестировать веб-приложение.

Актуальность данной работы обусловлена стремительным развитием e-learning в мире, поскольку проблема образования всегда оставалась одной из главных тем для совершенствования уровня жизни человечества. Благодаря появлению ИТ, появилась возможность сделать обучение более доступным, чем раньше.

Разработка приложения осуществлялась с использованием современных технологий, пользующихся популярностью в коммерческой разработке программного обеспечения. Приложение создано исключительно в рамках дипломной работы и предоставляет возможность получения опыта в создании собственного продукта, что будет полезно при дальнейшем трудоустройстве.

1 ОБЗОР РЕШЕНИЙ И ТЕХНОЛОГИЙ ЭЛЕКТРОННОГО ОБУЧЕНИЯ

1.1 Преимущества и недостатки электронного обучения

Разработка и применение образовательных технологий является одним из самых актуальных вопросов XXI века. Средства для упрощения жизни учащихся развиваются и улучшаются каждый день, что привело к появлению идеи электронного обучения.

Электронное обучение (E-learning) не только является фундаментом открытого и непрерывного образования, но и повышает качество традиционного обучения, внедряя современные технологии обучения в учебный процесс. В состав e-learning можно включить электронные учебники, общеобразовательные технологии и сервисы.

Использование информационно-коммуникационных технологий позволили определить процесс обучения, который сейчас называют электронным, благодаря чему данный вид образования имеет ряд достоинств и преимуществ над традиционным. [1]

Студент имеет возможность заниматься практически из любого места в мире. К тому же, не все процессы электронного образования реализуются посредством интернета.

Доступность обучения – студент имеет расходы только на носитель информации, интернет и, в некоторых случаях, на электронный курс. Это объясняется тем, что нет необходимости оплачивать зарплату учителей и содержание высших учебных заведений.

Длительность изучения курса учащийся выбирает сам, полностью подстраивая весь процесс обучения под свои требования. Нередко, это позволяет сделать обучение более эластичным.

Профессора и учащиеся совершенствуют свои теоретические знания и навыки в соответствии с современными технологиями и стандартами, что предоставляет возможность людям развиваться в ногу со временем. К тому же, электронное образование позволяет выставлять точные критерии, рассчитывающие знания, полученные студентами во время учебы.

По сравнению с традиционным, данный вид обучения имеет относительно быстрые циклы прохождения курсов. Как упоминалось до этого, студенты сами определяют собственную скорость обучения, что позволяет не следить за скоростью всей группы. Люди могут учиться, не выходя из собственного дома, из-за чего происходит экономия большого количества времени. Также студенты могут выбрать для изучения конкретные и соответствующие области учебного материала, не сосредотачиваясь на каждом разделе. То есть, они могут пропустить определенные курсы, которые не хотят проходить. [2, с. 24]

Благодаря этому, данный вид образования завоевал огромное признание в мире не только у студентов, но и у крупных корпораций, которые используют

его в качестве скорого и экономного повышения квалификации своих работников без отрыва от работы.

Необходимо подчеркнуть, что электронное обучение является безбумажным способом получения образования, за счет чего оно во многом спасает окружающую среду. Ученые провели исследование, после чего было установлено, что программы дистанционного обучения потребляют на 90% меньше энергии и выделяют на 85% меньше выбросов углекислого газа по сравнению с традиционным видом обучения. С e-learning нет необходимости рубить деревья для получения бумаги. Таким образом, оно является очень экологичным способом учебы. [3, с. 57]

Несмотря на преимущества, такой вид обучения испытывает трудности, включающие в себя:

– ограниченные навыки работы пользователей с компьютером;
– плохая обратная связь между преподавателями и студентами;
– слабое использование стандартов в электронном обучении;
– недостаточная развитость информационных технологий в Казахстане.

Учитывая вышеперечисленное, можно сказать, что использование электронных технологий обучения позволяет возможность из любого места загрузить дополнительные материалы, подкрепляющие теорию, повысить результативность студентов и школьников, увеличить доступность обучающих ресурсов посоветоваться с преподавателем удаленно, что положительно влияет на образовательный процесс и способствует развитию самостоятельного мышления среди учащихся.

1.2 Анализ существующих решений e-learning

Данная глава рассматривает крупнейшие проекты и решения, разработанные с помощью ИТ в области электронного обучения.

Coursera – проект, предоставляющий бесплатные онлайн-курсы на английском языке и основанный преподавателями Стэнфордского университета, который считается одним из самых престижных университетов мира. На текущий год в Coursera зарегистрировано около 25 миллионов пользователей и более 2000 курсов от 150 учебных организаций.

Приложение популярно своей базой, содержащей курсы не только по общеобразовательным предметам, таким как математика, физика, инженерное программирование, но и по экономике, маркетингу и бизнесу. Чтобы пройти курс, требуется приблизительно от шести до десяти недель стабильного обучения. Также курсы включают в себя домашние задания, еженедельные упражнения и даже завершающий экзамен.

Отличительной чертой Coursera является то, что большинство курсов написаны профессорами, которые работают в ведущих высших учебных заведениях мира. Вдобавок, многие из них имеют собственный бизнес.

Список имеющихся онлайн-курсов делится по темам и обучающим навыкам, перечисленным в таблице 1.1.

Таблица 1.1 – Список тем Coursera

Раздел	**Подразделы**
Гуманитарные науки и искусство	История, музыка, философия
Бизнес	Лидерство и менеджмент, финансы, маркетинг, предпринимательство
Компьютерные науки	Разработка ПО, разработка мобильных и веб-приложений, компьютерная безопасность и сети, дизайн и продукт
Наука о данных	Анализ данных, машинное обучение, статистика и теория вероятности
Информационные технологии	Облачные вычисления, безопасность, поддержка и операции
Здоровье	Животная медицина, информатика здоровья, питание, психология
Естественные и технические науки	Электротехническое проектирование, механическое проектирование, химия, экология, физика
Социальные науки	Экономика, образование, юриспруденция
Иностранные языки	Английский, другие языки

Udemy – это онлайн-платформа обучения, предназначенная для студентов, стремящихся улучшить свои профессиональные навыки. Главное отличие Udemy от Coursera в том, что курсы являются платными. На 2020 год в этой платформе отмечается более 50 миллионов студентов и 60 тысяч преподавателей, опубликовавших около 100 тысяч курсов.

Любой пользователь на Udemy может опубликовать собственный курс, что позволяет платформе охватить все виды деятельности, включая видеоигры и растениеводство. По этой же причине, Udemy является менее престижной, в отличии от Coursera, где преподают лучшие профессора мира.

К достоинствам Udemy можно отнести:

– огромное разнообразие курсов;

– легкость использования (понятный интерфейс);

– отсутствие технических проблем и ошибок;

– забота о пользователях (возврат денег).

Недостатки Udemy:

– несоответствие качества многих курсов к их цене;

– сомнительная система оценивания.

Стоит отметить, что купленные электронные курсы можно проходить неограниченное количество раз без лимита на их продолжительность прохождения.

Kahoot – это соревновательная онлайн-платформа, предназначенная в качестве образовательной технологии в учебных организациях. Kahoot предлагает пользователям публиковать онлайн-викторины, которые можно воспроизводить с другими игроками в веб-браузере или мобильном приложении.

Основной целью Kahoot является проверка знаний студентов и школьников, а также закрепление новых материалов. Данные задачи решаются

путем прохождения учащимися тестов, спроецированных на общем экране (интерактивная доска, монитор). В конце тестирования Kahoot выводит список лидеров, набравших высшие баллы.

Многие компании используют Kahoot в следующих назначениях:

– демонстрация простых презентаций;

– возможность самостоятельного обучения;

– переход к виртуальным мероприятиям.

Представленные решения активно используются не только учебными заведениями, но и крупными компаниями, что свидетельствует об очень быстром развитии электронного образования.

1.3 Постановка задачи дипломной работы

Цель дипломной работы – разработка веб-приложения, которое обеспечит дальнейшее развитие электронного обучения в высших учебных заведениях.

Задачи, решаемые данной работой:

– проанализировать современные архитектурные решения в веб-сфере;

– провести выбор системы управления базами данных (СУБД);

– исследовать актуальные средства проектирования и разработки программного обеспечения;

– спроектировать информационную систему (ИС), с помощью которой будет работать веб-приложение;

– разработать веб-приложение в соответствии с общепринятыми стандартами и технологиями.

Требования к разрабатываемому программному обеспечению:

– наличие серверной части, допускающей передачу данных из базы и их создание и изменение по запросу клиента;

– наличие клиентской части, отправляющей запросы на сервер;

– база данных должна быть создана с помощью объектно-реляционного отображения для более простого налаживания связей между таблицами;

– веб-приложение должно иметь кроссбраузерную и адаптивную верстку для правильного вывода страниц на устройство;

– возможность кэширования часто запрашиваемых данных, которая позволит повысить производительность информационной системы.

Веб-приложение должно предлагать такие сервисы, как:

– аутентификация и авторизация, которые являются важнейшими функциями сервера, обеспечивающими безопасность всего приложения путем защиты данных и операций от несанкционированного доступа со стороны злоумышленников;

– просмотр ленты последних новостей, связанных с образованием в Казахстане;

– ведение собственного органайзера, динамически меняющегося в зависимости от выбранного дня в календаре;

– электронная библиотека, позволяющая пользователю скачивать свободно-распространяемые ресурсы и материалы;

– прохождение онлайн-тестирования для закрепления имеющихся знаний и умений студента.

2 ВЫБОР СРЕДСТВ ПРОЕКТИРОВАНИЯ И РАЗРАБОТКИ СИСТЕМЫ

2.1 Архитектура веб-приложения

Комплекс ключевых решений, описывающих организацию программного обеспечения, является ее архитектурой. Архитектура характеризует выбор компонентов и интерфейсов, с помощью которых формируется структура информационной системы. На данный момент, существует два архитектурных стиля, которые активно используются веб-разработчиками: REST и GraphQL.

REST – это стиль взаимодействия компонентов распределенного приложения в сети. Достоинства REST-архитектуры:

– масштабируемость взаимодействия компонентов приложения;

– компоненты системы устанавливаются независимо друг от друга;

– надежность, обусловленная в устойчивости к отказам;

– простота и единство интерфейсов.

При использовании архитектурного стиля REST каждая единица данных (ресурс) определяется с помощью URL (Uniform Resource Locator), который является указателем к этой единице информации. В большинстве случаев, данные отправляются в формате JSON (JavaScript Object Notation), но можно использовать и другие форматы как HTML, PNG, PDF и другие.

Особенностью данного вида архитектуры является то, что она основана на протоколе HTTP, из-за чего управление ресурсами приложения происходит с помощью таких методов, как GET, POST, PUT и DELETE.

GET – метод, который применяется для запроса содержимого ресурса. Параметры запроса передаются после символа «?» (/api/resource?name=su). Если ресурс не был найден, возвращается ответ 404 (Not found).

POST – используется для загрузки новых ресурсов на сервер. Содержимое загружаемых данных помещается в тело запроса. При успешной загрузке данных сервер возвращает 200 (Ok) или 201 (Created).

PUT – обновляет ресурс по заданному идентификатору. Тело запроса должно содержать обновленные данные оригинала. Если ресурс не существует, создает его и передает ответ 201 (Created). В ином случае, изменяет его с ответом 200 (Ok).

DELETE – удаляет ресурс, указанный с помощью идентификатора. Как правило, ничего не возвращает.

С другой стороны, GraphQL – это архитектура интерфейса приложения, использующая более гибкий подход к программированию. GraphQL не работает с выделенными ресурсами приложениями, что является главным отличием данной архитектуры от REST.

GraphQL позволяет создавать более быстрые итерации программного обеспечения во внешнем интерфейсе, поскольку разработчики могут вносить изменения лишь в клиентской части, не подключаясь к серверу. К тому же, разработчики способны принять представление о данных, запрашиваемых на

сервере, и о том, как применяются эти данные. Это возможно благодаря тому, что в GraphQL клиент сам определяет, какая информация ему необходима. По этой причине, разработчики могут отказаться от полей, которые клиенты больше не используют, повышая производительность API.

GraphQL не зависит от протокола данных и может использовать любой из них. Для работы с данными пользователи обращаются к единой точке входа – серверу. При изменении структуры, поля и параметров запроса, клиент работает с разными данными. В отличии от REST-архитектуры, которая может возвращать данные разных форматов: JSON, XML, PDF и т.д., GraphQL способна передавать только JSON-формат. Стоит добавить, что GraphQL имеет возможность передать аргументы запроса на любой уровень вложенности, что делает данную архитектуру не такой строгой, как REST.

Изучив ключевые характеристики каждой из архитектур, было решено использовать архитектуру REST по следующим причинам:

– GraphQL использует одну конечную точку входа вместо следования спецификации HTTP для кэширования, что увеличивает объем траффика на сервер, перегружая его;

– в запросах GraphQL могут возникнуть проблемы с производительностью, если клиент запрашивает слишком много вложенных полей одновременно;

– так как разрабатываемое приложение является небольшим, то REST-архитектура может быть более ценным подходом для подключения управляемых ресурсами приложений, которым не нужны гибкие запросы.

Система считается Restful, если она использует клиент-серверную модель передачи данных. Сервер занимается хранением данных, в то время как клиент обращается к серверу за получением этих данных, благодаря чему мобильность клиента значительно растет. Состояние сессии должно храниться только на стороне клиента. Клиент должен изложить запросы так, чтобы в них содержалась лишь нужная для обработки запроса информация и, в некоторых случаях, идентификация пользователя. Кроме того, сервер должен иметь возможность кэширования данных, то есть отмечать некоторые ответы как кэшируемые с целью устранения возможности клиента получить устаревшие, либо неверные данные в ответ на последующие запрашивания.

Restful-система обладает единообразием интерфейса. Наличие уникального интерфейса позволяет сервисам приложения развиваться отдельно. Существует четыре принципа единого интерфейса:

– идентификация ресурсов (URI);

– манипуляция над ресурсами через представление;

– само-документируемые сообщения;

– гипермедиа как двигатель состояния приложения.

Масштабируемость Restful-системы повышается за счет использования промежуточных серверов, позволяя компенсировать нагрузки и распределенное кэширование между ними.

На рисунке 2.1 изображена схема работы разрабатываемого приложения, которая соответствует REST-архитектуре.

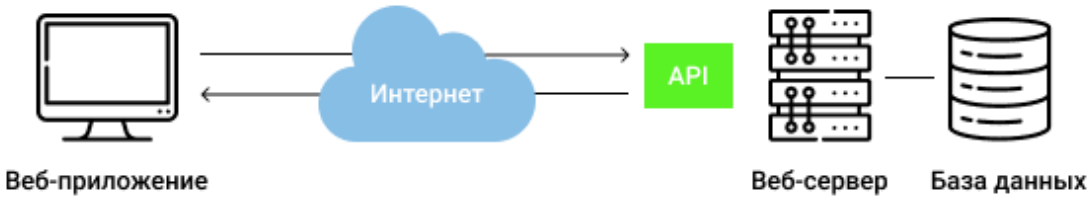

Рисунок 2.1 – REST-система

Архитектура REST не прикреплена к конкретным технологиям и протоколам, но построение Restful API обязательно требует применение HTTP и форматов обмена данными. В данной дипломной работе используются методы GET, POST и DELETE протокола HTTP, а передача данных осуществляется с помощью формата JSON.

2.2 Выбор системы управления базами данных

В области ИТ одну из важнейших ролей занимает база данных, которую можно охарактеризовать как организованную совокупность информации, предназначенную для ее хранения, обновления и обработки. Существуют следующие виды баз данных, классифицированных по модели данных:

– иерархическая;
– объектная и объектно-ориентированная;
– объектно-реляционная;
– реляционная;
– сетевая;
– функциональная.

Разрабатываемое веб-приложение не может существовать без базы данных по причине необходимости хранения значительного количества данных, которые следует активно передавать клиентам. Взаимодействие приложения и базы данных приводит к тому, что страницы приложения формируются в реальном режиме времени по запросу пользователя.

Для заполнения, изменения и представления информации используется комплекс программных средств специального назначения, именуемый как СУБД или система управления базами данных. Операции, которые предоставляет СУБД:

– администрирование данными во внешней памяти;
– администрирование данными в оперативной памяти;
– резервное копирование и восстановление БД;
– поддержка языков БД (DDL, DML). [4]

В настоящее время, самыми популярными СУБД в веб-области являются MySQL, PostgreSQL и MongoDB. Разберем каждую из них.

MySQL – это бесплатный менеджер баз данных с открытым исходным кодом. Чаще всего используется в сборке LAMP (Linux, Apache, MySQL, PHP) в качестве сервера, к которому обращаются клиенты. Одним из основных

достоинств СУБД считается то, что при увеличении объема информации, ее емкость хранения существенно снижается. Однако, после того как компания Oracle приобрела MySQL, СУБД стала обновляться реже, а функции, которые содержат обновления, в основном копируются у конкурентов (например, PostgreSQL).

PostgreSQL – это свободно-распространяемая объектно-реляционная СУБД, которая поддерживает огромный выбор встроенных и определяемых пользователем типов данных. К тому же, данная СУБД гарантирует расширенную емкость и целостность данных.

Главными особенностями PostgreSQL являются управление БД с большими объемами информации, превосходная безопасность и надежность на уровне обработки данных. PostgreSQL использует язык структурированных данных (SQL), который считается простым в управлении и получении информации. В интернете имеется внушительное количество документации и руководств для понимания того, как использовать данный инструмент. [5, с. 49]

MongoDB является документоориентированной СУБД, которая классифицируется как NoSQL-система. MongoDB использует документы формата JSON и не нуждается в описании схемы таблиц.

Модель данных документов системы позволяет разработчикам иметь высокую гибкость, а распределенная архитектура обеспечивает большую масштабируемость. Вследствие чего, СУБД выбирается для проектов, которые должны управлять большими объемами информации и не соответствуют реляционной модели данных.

Так как разрабатываемое приложение является небольшим и не требует большой гибкости, которую предлагает MongoDB, а MySQL не имеет должного развития, обусловленного сообществом, то будет разумно применить в качестве СУБД PostgreSQL.

В таблице 2.1 приведены ограничения в PostgreSQL.

Таблица 2.1 – Ограничения в PostgreSQL

Ограничение	**Значение**
Максимальный размер БД	Неограничен
Максимальный размер таблицы	32 TB
Максимальный размер строки	1.6 TB
Максимальный размер поля	1 GB
Максимальное количество строк в таблице	Неограниченно
Максимальное количество столбцов в таблице	Варьируется между 250-1600 в зависимости от типа столбца
Максимальное количество индексов в таблице	Неограниченно

В моем случае, PostgreSQL предназначен для приложения, которому требуется сильная стабильность в хранении информации, высокая безопасность и гарантированная целостность данных. С помощью данной СУБД присутствует возможность расширять базу приложения в неограниченном объеме.

2.3 Средства проектирования информационной системы

В дипломной работе большую роль играет проектирование информационной системы. По этой причине следует изучить актуальные средства проектирования ИС.

Под информационной системой понимают систему, которая включает в себя вычислительную установку и программное обеспечение, необходимое для сбора, обработки, хранения, выдачи информации, а также для принятия управленческих решений. Цель ИС – обеспечить пользователей требуемой информацией.

Проектирование ИС проекта состоит из следующих областей:

– проектирование объектов базы данных;

– проектирование схем работы веб-приложения;

– техническое проектирование. [6]

Так как в проекте будет использована объектно-реляционная база данных, то для проектирования объектов приложения будет использована ER-диаграмма, которая описывает сущности в виде таблиц и обозначает их связи. Модель строится методом последовательных уточнений исходных схем, что является главным преимуществом данного метода. ER-диаграмма является одним из методов, требуемых для определения требований программного обеспечения.

Проектирование схем работы веб-приложения упрощает понимание требований проекта и способов его реализации, а также позволяет посмотреть на приложение с отдельных точек зрения. Для решения данной проблемы применяется UML – унифицированный язык моделирования, способный визуализировать и описать программную систему. В дипломной работе используются следующие диаграммы, предлагаемые UML:

– диаграмма развертывания;

– диаграмма вариантов использования;

– диаграмма последовательности.

Техническое проектирование, в свою очередь, включает описание аппаратной среды и серверного программного обеспечения. Для стабильной работы ИС необходимо иметь высокий уровень адаптивности аппаратной среды к изменяемым условиям. ПО должно включать в себя операционную систему, утилиты и прикладные программы.

Проектирование информационной системы считается необходимой задачей при разработке приложений и позволяет описать условия, при которых система будет принимать, обрабатывать и хранить информацию. Проведенный обзор отмечает список современных методов моделирования ИС, что может помочь решить определенные проблемы верным способом.

2.4 Средства разработки веб-приложения

В настоящее время, индустрия веб-разработки растет стремительным образом, из-за чего постоянно появляются новые решения и технологии,

использование которых позволяет приложениям соответствовать современным стандартам ИТ. Ниже приведен список инструментов, с помощью которых осуществлялась разработка программного обеспечения.

Spring – фреймворк, основанный на языке программирования Java, с открытым исходным кодом. Сердцевиной Spring принято считать контейнер IoC (Inversion of Control), отвечающий за управление над жизненными циклами объектов. Объекты, которые создает контейнер, называют бинами (beans). Spring используется для создания сложных информационных систем, построенных с помощью Java-платформы. Концепция разработки, которую предлагает Spring, основана на передовых стандартах ИТ. Под данным фреймворком подразумевают семейство библиотек, включающих в себя:

– Spring Core или ядро платформы, позволяющее применять основные средства для создания приложений: управление бинами, внедрение зависимостей (DI), транзакции, доступ к БД.

– Spring MVC, которая является библиотекой, настраивающей направление запросов, создание контроллеров, интеграцию с шаблонными движками на основе HTTP.

– Spring Data, представляющая собой согласованную модель программирования для доступа к базам данным, которая сохраняет особые свойства основного хранилища.

– Spring Security или среда для настройки аутентификации и контроля доступа к данным, включает в себя таких провайдеров, как OAuth, LDAP.

Hibernate – программная платформа, реализующая спецификацию JPA (Java Persistence API), которая предоставляет возможность сохранять объекты в виде таблиц в БД. Позволяет связать объектно-ориентированное программирование и реляцию модель данных. Фреймворк предоставляет способы для автоматического создания и обновления таблиц, и существенно снижает время разработки, исключив использование SQL-запросов. Основным требованием Hibernate при создании классов является наличие конструктора без параметров.

Maven – средство, позволяющее автоматизировать сборку приложения. Использует язык POM (Project Object Model) для изменения структуры проекта путем добавления новых зависимостей. Maven считается простым в использовании, быстрым и гибким. Платформа предлагает возможность управления сборкой и развертыванием приложений в одном месте.

Angular – открытый фреймворк, предназначенный для разработки веб-приложений, на платформе TypeScript, расширяющей возможности JavaScript. Данный фреймворк имеет возможность создавать сложные реактивные формы, сокращая при этом время разработки. Также Angular обладает модулем, который позволяет отображать различные компоненты и данные для пользователя в зависимости от того, где пользователь находится в приложении в текущий момент. Маршрутизатор переходит от одного представления к другому, когда пользователь вводит URL в адресной строке, либо переходит по ссылкам на странице.

Одной из главных особенностей Angular является наличие Observables, поведенческого шаблона проектирования, который обеспечивает поддержку для передачи сообщений между частями приложения. Observables является рекомендуемой техникой для обработки событий, асинхронного программирования и обработки нескольких значений.

Вдобавок, Angular предлагает установку инструментов разработчика (CLI), которые облегчают установку компонентов и настройку платформы.

NPM – менеджер пакетов, предлагающий управление процессом установки, обновления, удаления и настройки различных компонентов проекта. Позволяет управлять клиентской частью приложения с помощью командной строки.

HTML & CSS – представляют собой фундамент любого веб-приложения и поддерживаются всеми современными браузерами. Первый является стандартизированным языком гипертекстовой разметки, а второй – языком, задающим внешний вид документам.

Знание перечисленных инструментов разработки, с помощью которых ведется создание приложения, является актуальней задачей для многих начинающих разработчиков и может помочь в трудоустройстве при окончании университета, что легло в основу решения выбора технологий.

3 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА СИСТЕМЫ

3.1 Моделирование базы данных

Опираясь на сервисы, которые должен обеспечивать проект, база данных требует наличия сущностей и их связей, которые определены в данной главе.

Пользователь – лицо, выполняющее определенные действия в системе.

Роль –описание совокупности возможностей, которые может выполнять пользователь.

Задача – действие, которое пользователь должен реализовать в определенный срок.

Новость – сообщение о недавно произошедшем событии в области образования.

Книга – свободный ресурс, который пользователь может загрузить.

Тест – инструмент, предназначенный для проверки знаний студентов и состоящий из списка вопросов.

Вопрос – обращение, которое требует ответа.

По правилам объектно-реляционной модели данных, каждый перечисленный объект представляет отдельную таблицу в БД. Данные таблицы не нужно описывать вручную благодаря ORM – методу, который преобразует части кода объектно-ориентированных языков программирования в SQL-запросы. Проект использует Hibernate в качестве ORM (подробнее в главе 2.4).

Некоторые объекты БД имеют также связи с другими объектами, которые можно определить как отношения между участниками. Связи строятся с использованием внешних ключей – атрибутов, которые указывают на строки других таблиц. [7, с. 364]

Многие ко многим – вид связи, при котором каждой сущности соответствует одна и более других сущностей. Для реализации такого вида связи применяется таблица-посредник, которая содержит внешние ключи связываемых таблиц.

Один ко многим – вид связи, при котором первая сущность принадлежит только одному экземпляру второй сущности, а вторая сущность содержит один и более экземпляров первой сущности.

Один к одному – вид связи, при котором каждая сущность указывает лишь на один экземпляр другой сущности. Чтобы избежать связи 1:M нужно сделать ограничение внешнего ключа одной из таблиц уникальным.

Свойства таблиц и их связи изображены с помощью диаграммы «сущность-связь, изображенной на рисунке 3.1.

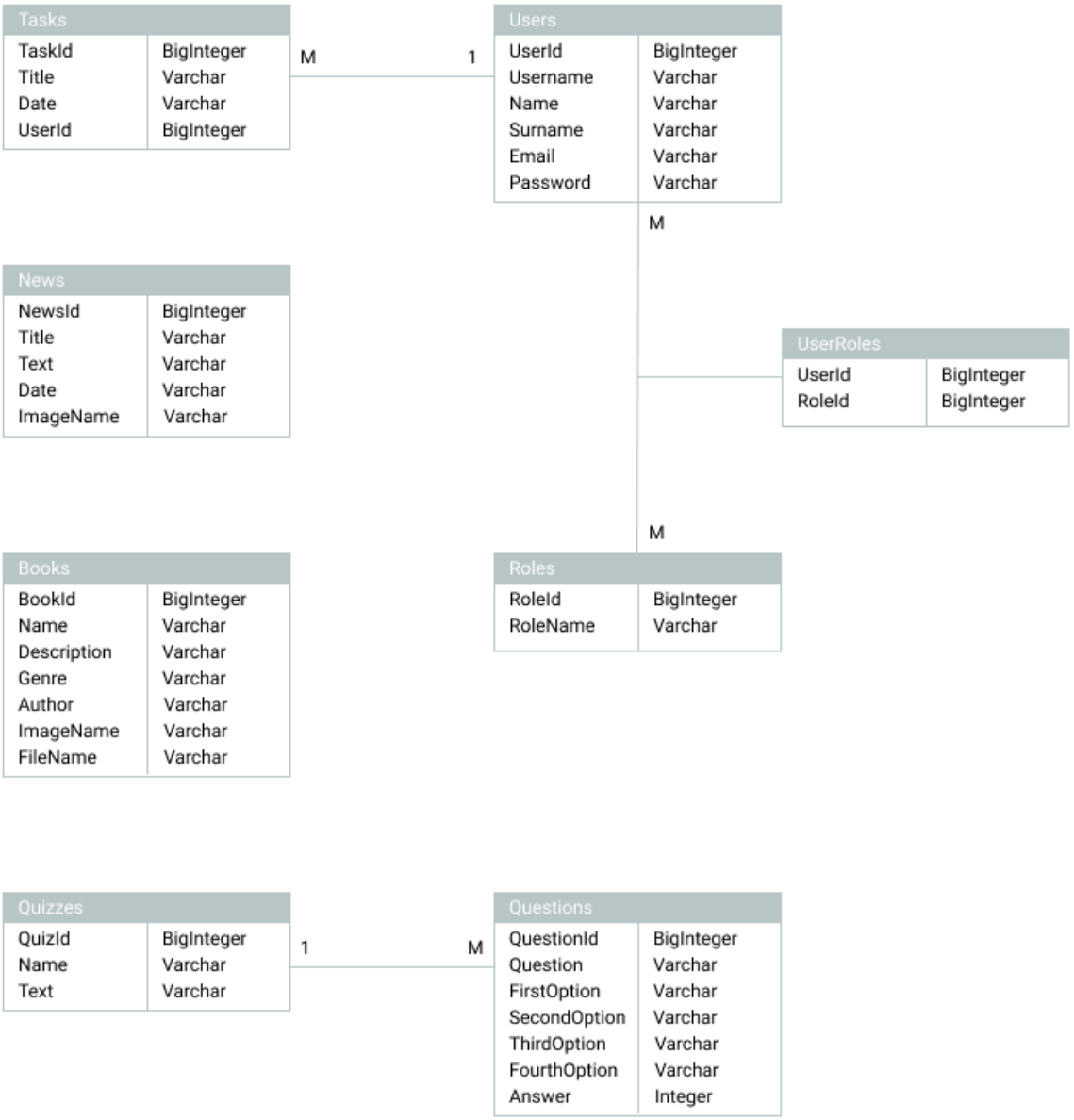

Рисунок 3.1 – ER-диаграмма

Как видно на рисунке 3.1, таблица пользователей связана с таблицей ролей отношением M:M и с таблицей задач отношением 1:M, в то время как таблица тестов связана с вопросами при помощи отношения 1:M. Стоит добавить, что проектирование базы данных является важной задачей, позволяющей идентифицировать данные и их связи с помощью описания и визуализации объектов.

3.2 Безопасность веб-приложения

Защита разрабатываемого программного обеспечения обеспечивается путем ограничения доступа к данным и действиям, связанных с ними, а также шифрованием некоторых сведений пользователей. Для ограничения доступа пользователей были применены такие процессы, как авторизация и аутентификация.

Авторизация – это предоставление пользователю возможностей на выполнение определенных операций в зависимости от его роли. Процесс авторизации определяет имеет ли текущее лицо доступ к следующим ресурсам: информации, файлам и базе данных.

Аутентификация – процедура подтверждения пользователя путем проверки введенных им учетных данных: идентификатора и пароля. В качестве идентификатора могут использоваться имя пользователя, адрес электронной почты или номер телефона.

Регистрация, в свою очередь, является процессом создания нового пользователя, вследствие чего он сохраняется в БД. Пользователь должен заполнить регистрационную форму, которая включает в себя учетные данные, с помощью которых он может пройти аутентификацию в будущем.

Для решения вопроса идентификации пользователя применяется JWT (JSON Web Token) – объект, определенный в открытом стандарте RFC 7519 и основанный на формате JSON. Данный способ передачи данных между сервером и клиентом является одним из самых безопасных в веб-сфере. Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения своей личности.

JWT представляет собой строку, состоящую из таких частей, как header (заголовок), payload (полезные данные) и signature (электронная подпись), разделенных точками (таблица 3.1).

Таблица 3.1 – Структура JWT-токена

Header	{
	“alg”: “HS256”,
	“typ”: “JWT”
	}
		Определяет, какой алгоритм используется для генерации подписи.

Payload	{
	“username”: “abdraymd”,
	“iat”: “1516239022”
	}
		Указывает на полезные данные, хранящиеся в токене.
		В данном примере используются две заявки, которые определяют имя пользователя и срок истечения токена в секундах.

Signature	HMAC-SHA256(base64urlEncoding(header) + ‘.’ + base64urlEncoding(payload), secret)	Подпись рассчитывается путем складывания кодированных header и payload, после чего полученная строка хэшируется на основе секретного ключа.

Объединив все части вместе, генерируется токен следующего вида: «eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFiZHJheW1kIiwiaWF0IjoxNTE2MjM5MDIyfQ.AVBTrNsSmmwbXKEuH_R3iWYTCR7UdBH-MOf_aMgbVdE».

После аутентификации пользователя, сервер передает клиенту JWT-токен, который должен быть сохранен в локальном хранилище или cookie-файле.

Каждый раз, когда пользователь обращается к API, требуется добавить токен в заголовок авторизации. Сервер проверяет данный токен и определяет, является ли пользователь тем, за кого себя выдает. Алгоритм применения JWT-токена изображен на рисунке 3.2.

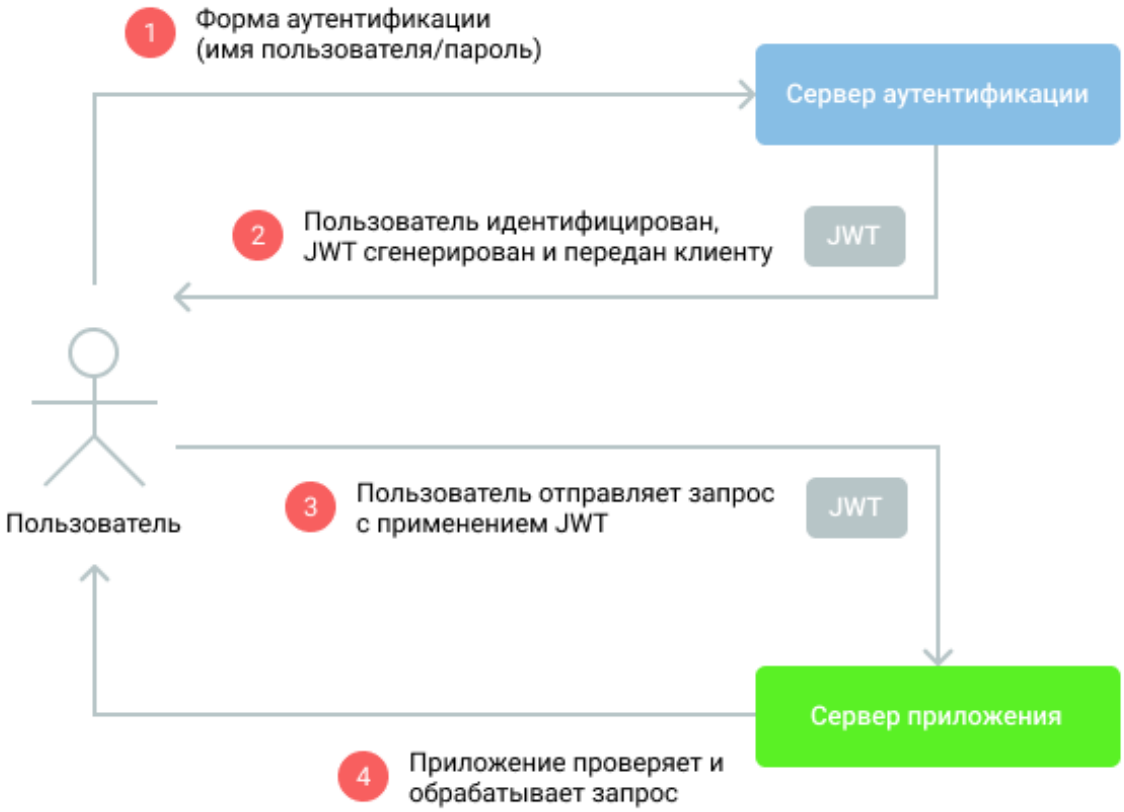

Рисунок 3.2 – Алгоритм применения JWT

Необходимо отметить, что доступом к системе располагают две роли:

– администратор;

– обычный пользователь.

Администратор имеет право управлять компонентами веб-приложения, а именно создавать и удалять элементы системы, в то время как пользователь способен лишь просматривать эти компоненты, а также загружать свободные-распространяемые ресурсы. Привилегии данных ролей представлены с помощью диаграммы, изображенной на рисунке 3.3.

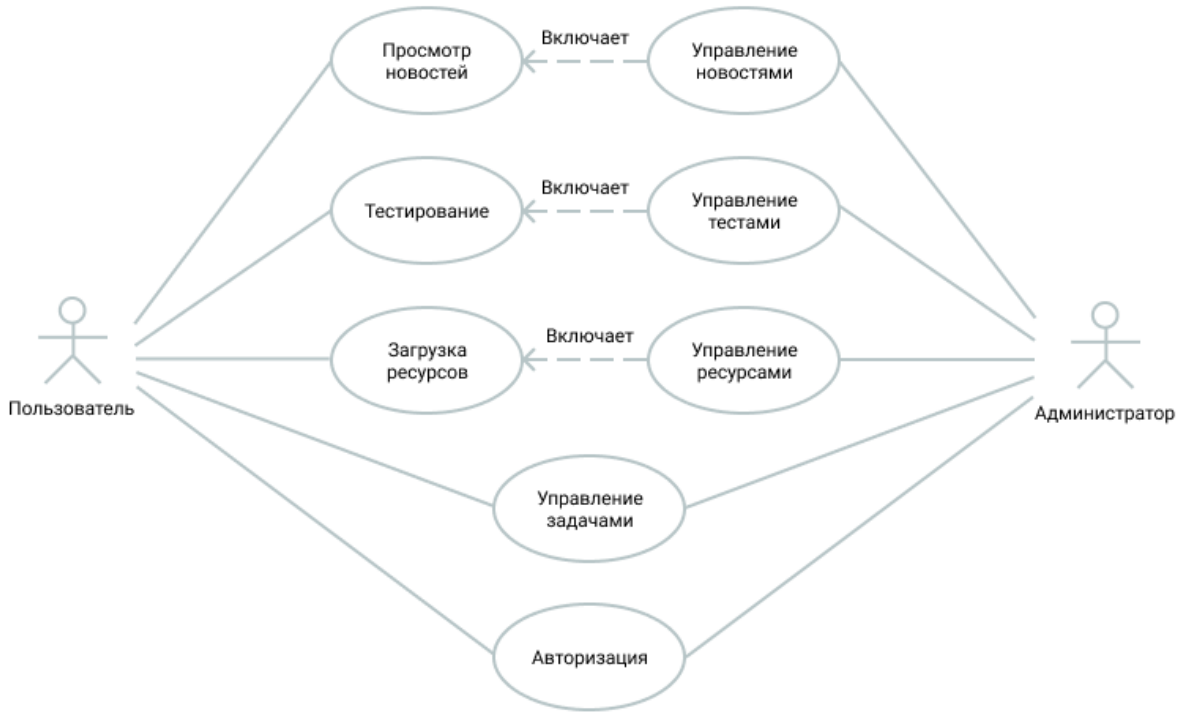

Рисунок 3.3 – Диаграмма вариантов использования

Таким образом, безопасность веб-приложения зависит от стандарта JWT, который предотвращает несанкционированный доступ к системе и предоставляет права, которыми обладает идентифицируемый пользователь. Данный метод аутентификации не сохраняет состояние пользователя на сервере, что соответствует одному из требований REST-архитектуры. Это позволяет уменьшить необходимость многократного запроса к БД. Методы авторизации и регистрации описаны в приложении А.

3.3 Разработка компонентов серверной части

Запросы к программно-аппаратной части сервиса осуществляются по адресу «http://localhost:8080». Например, для передачи списка новостей клиенту применяется GET-запрос, для осуществления которого требуется добавить к главному пути выражение «/news». Запросы создаются на стадии реализации контроллеров – центральных компонентов, обрабатывающих требования пользователя. Но перед началом создания контроллеров, требуется наличие других элементов системы, которые описаны в приложении Б.

Первым делом, при разработке какого-либо сервиса приложения создается модель – компонент, который предоставляет данные и меняет свое состояние в ответ на команды контроллера. Модель представляет собой обычный класс, который описывает свойства столбцов таблицы, в которую он преобразуется. Разберем модель News, которая описана в листинге Б.1. Аннотация @Entity указывает на то, что класс должен представлять таблицу в БД. В теле класса описываются признаки сущности, такие как идентификатор, заголовок, текст, дата и название изображения. Далее следует обязательный пустой конструктор

и конструктор с описанием создания нового объекта данной модели. В конце необходимо добавить геттеры и сеттеры, которые являются методами чтения и записи свойств объекта. [8, с. 375]

После создания модели, следует описание репозитория – механизма для обращения к БД, который отвечает за получение и сохранение объектов определенной модели. Пример создания репозитория на основе класса News представляет листинг Б.2. Аннотация @Repository означает, что описываемый интерфейс является репозиторием. Репозиторий должен наследоваться от другого интерфейса JpaRepository с параметрами, указывающими на название и идентификатор модели.

В последнюю очередь идет создание контроллера, который, как отмечалось выше, отвечает за описание запросов пользователя. Контроллер управляет данными и их направлением от пользователя к системе и наоборот. Стоит отметить, что каждый контроллер должен иметь свой уникальный адрес обращения. Рассмотрим GET-запрос, представленный в листинге Б.3. Данный метод получает все объекты из таблицы новостей и возвращает код 200 (Ok), как представлено на рисунке 3.4.

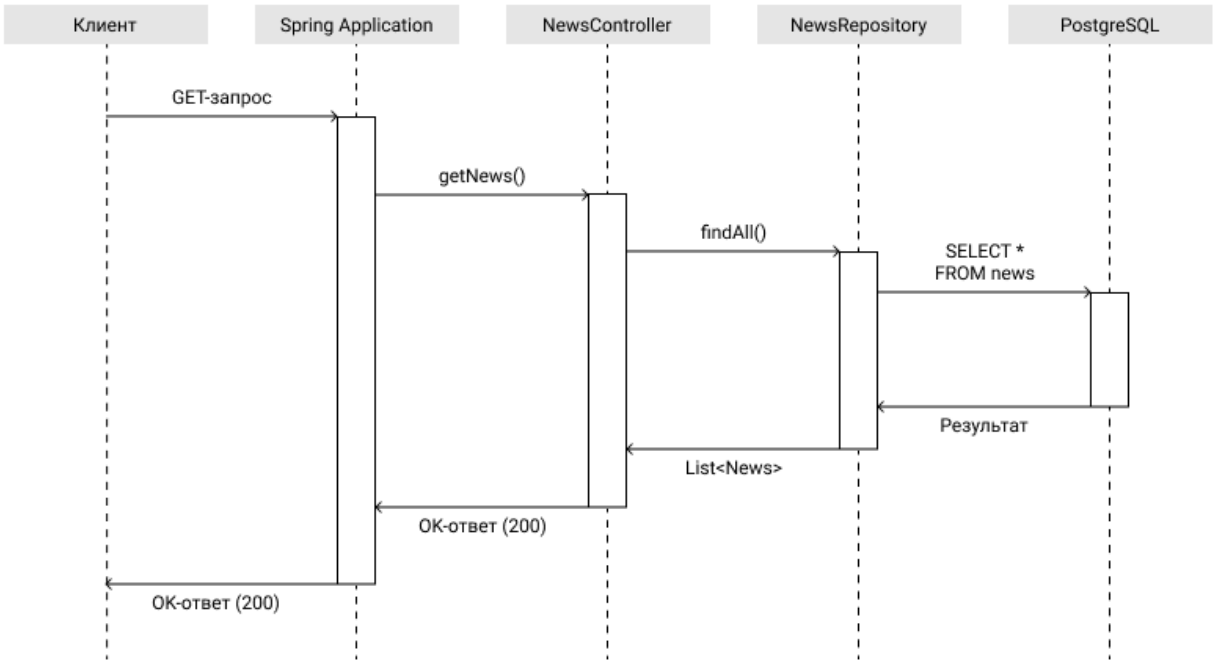

Рисунок 3.4 – Диаграмма последовательности

Аналогичным образом работают методы POST и DELETE данного контроллера (листинги Б.4 – Б.5), только доступ к ним имеет администратор, о чем свидетельствует аннотация @PreAuthorize. Метод createNews() создает новый объект класса News и загружает изображение на сервер, а метод deleteNews() удаляет объект из таблицы по его идентификатору.

Все остальные компоненты серверной части приложения были реализованы по алгоритму, который описывается в данной главе.

3.4 Разработка компонентов клиентской части

Реализация клиентской части приложения (frontend) – это практика преобразования данных в графический интерфейс, с которым может взаимодействовать пользователь. В данной дипломной работе frontend реализован с помощью одностраничного приложения (SPA) – приложения, которое во время использования работает без перезагрузки страниц. Другими словами, это одна веб-страница, которая загружает требуемый контент с помощью JavaScript. Разберем данную технологию на примере разработки ленты новостей, программный код которой представлен в приложении В. В качестве порта используется «http://localhost:4200».

Для упрощения разработки следует описать интерфейс, который соответствует сущности, с помощью которой новости должны быть представлены, как в листинге В.1.

Далее, нужно создать файл-сервис, который представляет собой класс, выполняющий следующие задачи:

– предоставление данных через обращение к серверу;

– взаимодействие с другими компонентами приложения;

– логирование и вычислительные решения.

Сервис NewsService, представленный в листинге В.2, обращается к API посредством переменной url. Метод getNews() отправляет GET-запрос для получения списка новостей, а метод create() – POST-запрос, который содержит новый объект новости в качестве тела запроса. Метод delete(), в свою очередь, отвечает за передачу DELETE-запроса и содержит идентификатор удаляемого объекта. Также данный сервис содержит методы listen() и filter(), необходимые для реагирования на действия пользователя. [9, с. 145]

Следующим шагом идет разработка TypeScript-компонента NewsComponent, который связан с HTML-шаблоном. Данный компонент внедряет ранее созданный сервис для применения его методов. Как видно на листинге В.3, метод ngOnInit() предоставляет список новостей шаблону с помощью метода getNews(), который был описан в сервисе NewsService, во время инициализации. Метод deleteNews() позволяет удалять определенную новость, после чего список новостей заново загружается. Для создания нового объекта News в БД, применяется метод onAdd(), который открывает модульное окно с формой добавления соответствующего элемента.

HTML-шаблон списка новостей представлен в листинге В.4. По данному шаблону видно, что каждая новость добавляется с применением цикла, который проходит по всем элементам массива новостей news, указанном в компоненте. К тому же, шаблон содержит кнопки, ссылающиеся на методы onAdd() и deleteNews() компонента NewsComponent.

Таким образом, SPA работает эффективно, загружая лишь те ресурсы, которые необходимо предоставить пользователю. К тому же, разработка SPA упрощена и оптимизирована, так как отсутствует надобность писать отдельный

код для разных данных. Все остальные компоненты клиентской части приложения были разработаны аналогично описанному алгоритму.

3.5 Дерево маршрутизации веб-приложения

Маршрутизация обеспечивает переходы пользователя между представлениями приложения в зависимости от URL. В клиентской части приложения конфигурация маршрутизации описана в приложении Г и представляет собой массив routes (листинг Г.1). Каждый маршрут в массиве routes является JavaScript-объектом, который содержит свойство path, определяющее URL-путь маршрута, а также свойство component, определяющее компонент соответствующего пути.

Незарегистрированный пользователь может переходить только по страницам авторизации и регистрации. При вводе пользователем своих учетных данных, происходит его идентификация, после успешного прохождения которой пользователь перенаправляется на страницу ленты новостей. Ограничения представлений обеспечивает класс AuthGuard, содержащий метод canActivate, который проверяет наличие JWT-токена в локальном хранилище браузера (листинг Г.2). Если токен не существует, метод возвращает false, а пользователь перенаправляется на страницу авторизации. В другом случае, метод возвращает true, что означает, что пользователь может переходить по всем страницам приложения.

Дерево маршрутизации приложения изображено на рисунке 3.5.

Рисунок 3.5 – Дерево маршрутизации

Так как веб-сайт использует технологию одностраничного приложения, переходы между страницами осуществляются с помощью изменения DOM-элементов главного файла представлений, определенного в структуре проекта как index.html. Почти полное отсутствие перезагрузки страниц позволяет ускорить время работы веб-приложения.

3.6 Требования к аппаратному обеспечению сервера

Сервер – это аппаратное обеспечение, предназначенное для выполнения сервисного ПО. В модели клиент-серверного программирования серверное приложение выполняет запросы, которые отправляет пользователь с удаленного компьютера. Для стабильной обработки большого количества запросов рекомендуется разделить систему на сервер приложений и сервер базы данных, взаимодействие которых описано на рисунке 3.6. [10]

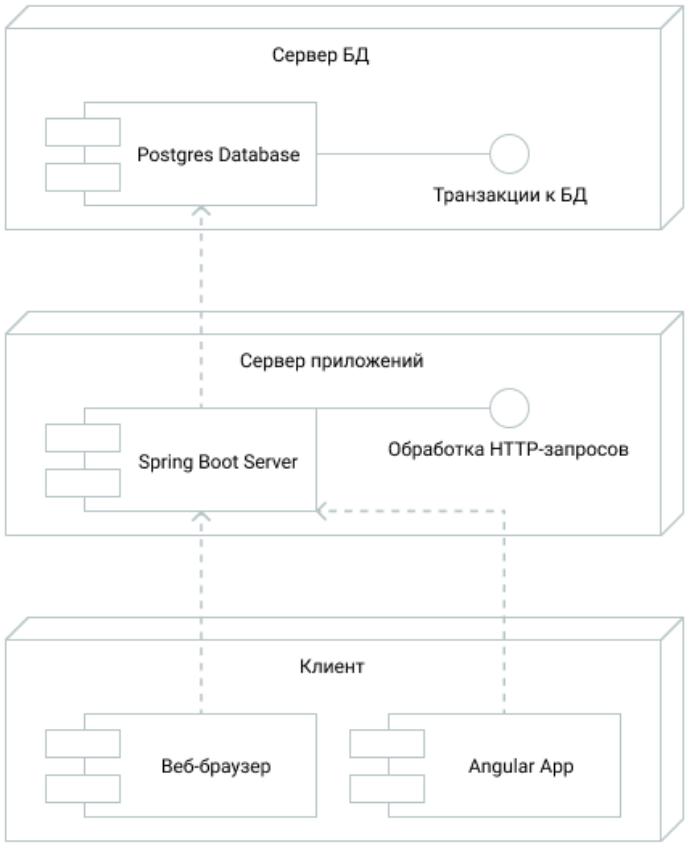

Рисунок 3.6 – Диаграмма развертывания

Сервер приложений – это сервер, специально разработанный для запуска прикладных программ. Данный сервер обеспечивает высокую вычислительную мощность для запуска разрабатываемого веб-приложения. Он также обрабатывает запросы пользователей, обращаясь к серверу базы данных за получением необходимых данных. Спецификация такого вида сервера зависит от количества пользователей (таблица 3.2).

Сервер базы данных – это система, которая предоставляет другим компьютерам услуги, связанные с доступом и получением данных из БД (таблица 3.3). Сервер базы данных выполняет следующие задачи:

– хранение данных;
– анализ данных;
– манипулирование данными;
– архивирование БД.

Таблица 3.2 – Требования к серверу приложений

Число пользователей	**Процессор (ГГц)**	**Число процессоров**	**ОЗУ (Гб)**
1000	3	1	4
5000	3	2	8
10000	3	2	16
25000	3	4	32

Таблица 3.3 – Требования к серверу БД

Вид требований	**Процессор (ГГц)**	**ОЗУ (Гб)**	**RAID-диск (Гб)**
Минимальные	1	2	100
Рекомендуемые	3	8	250

Развертывание программного обеспечения требует аппаратных характеристик, определенных в данной главе. Данные спецификации были составлены на основе численности управляемых устройств и соответствуют актуальным решениям размещения небольших веб-приложений.

3.7 Серверное программное обеспечение

Программное обеспечение сервера состоит из операционной системы, а также инструментов, которые позволяют запустить разработанное веб-приложение. Выбор ПО основан на средствах разработки приложения.

Операционная система (ОС) – комплекс программ, который управляет всеми остальными прикладными программами системы. Взаимодействие с ОС происходит через пользовательский интерфейс, такой как интерфейс командной строки (CLI) или графический интерфейс пользователя (GUI). В качестве ОС сервера подходит Linux Ubuntu, который считается идеальным вариантом для развертывания средних по размеру приложений. Ubuntu обеспечивает организованное управление пакетами, высокую совместимость и легкую настройку системы. Это стабильная платформа с сильным сообществом, которое постоянно поддерживает ее исходный код.

Для преобразования исходного кода требуется наличие трех технологических пакетов, используемых при программировании на языке Java:

– виртуальная машина Java (JVM);

– среда выполнения Java (JRE);

– комплект разработки Java (JDK).

Apache Tomcat – контейнер Java-сервлетов (интерфейсов, расширяющих функциональность сервера), который предоставляет платформу для развертывания веб-приложений на основе HTTP. Tomcat поддерживает все современные функции, необходимые для сервера приложений, а также позволяет размещать и настраивать его из командной строки.

В целях выполнения обновлений, обработки отладочных испытаний и развертывания клиентской части приложения используется Angular CLI, который обозначен как NPM-модуль (подробнее в главе 2.4).

pgAdmin – графический клиент для работы с сервером базы данных Postgres, который позволяет упростить его администрирование. Имеет панель мониторинга, с помощью которого можно отслеживать действия сервера, такие как блокировки БД и подготовленные транзакции. Предоставляет возможность удаленного доступа через браузер, так как является веб-приложением.

В качестве быстрого и стабильного браузера должен быть установлен Google Chrome, который поддерживает надежные функции безопасности и ведущие стандарты в области веб-технологий. Используется для тестирования веб-приложения.

Перечисленное программное обеспечение необходимо для развертывания и, в некоторых случаях, тестирования приложения, которые было разработано в рамках данной дипломной работы.

3.8 Тестирование веб-приложения

Разберем то, как выглядит приложение с позиции обычного пользователя, у которого нет административных прав на изменение содержания веб-приложения.

Главной страницей является страница «Новости», но, чтобы получить к ней доступ, необходимо войти в систему с помощью формы авторизации (рисунок 3.7).

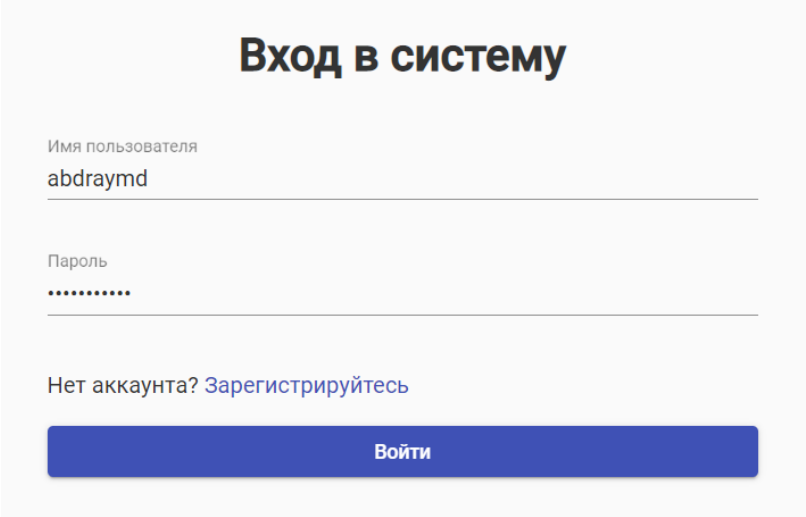

Рисунок 3.7 – Вход в систему

Если пользователь впервые использует данное приложение, он должен пройти форму регистрации, которая изображена на рисунке 3.8.

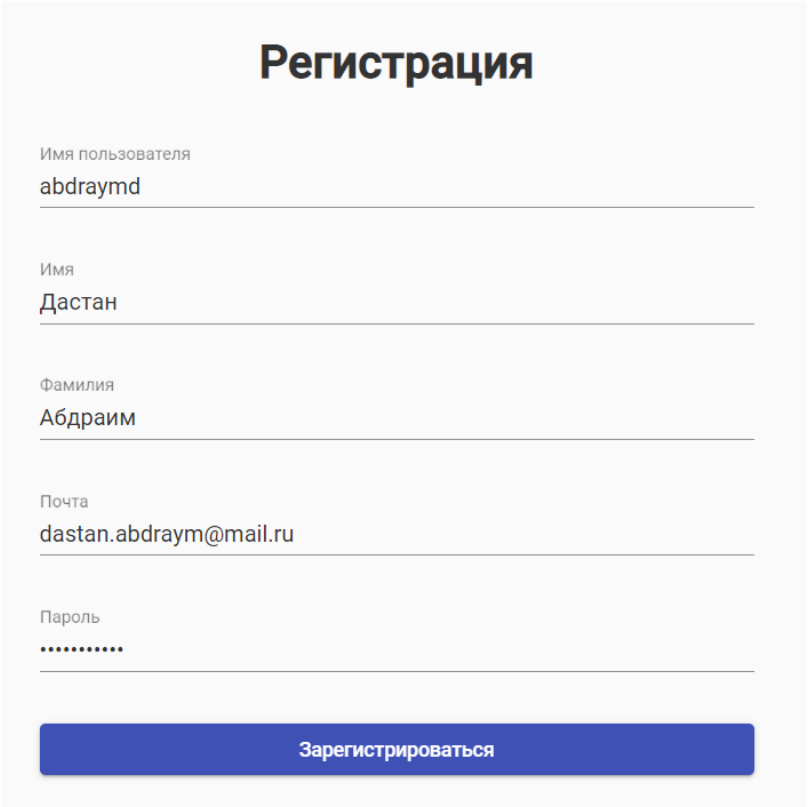

Рисунок 3.8 – Регистрация

Лента новостей содержит последнюю информацию об образовании в рамках страны (рисунок 3.9).

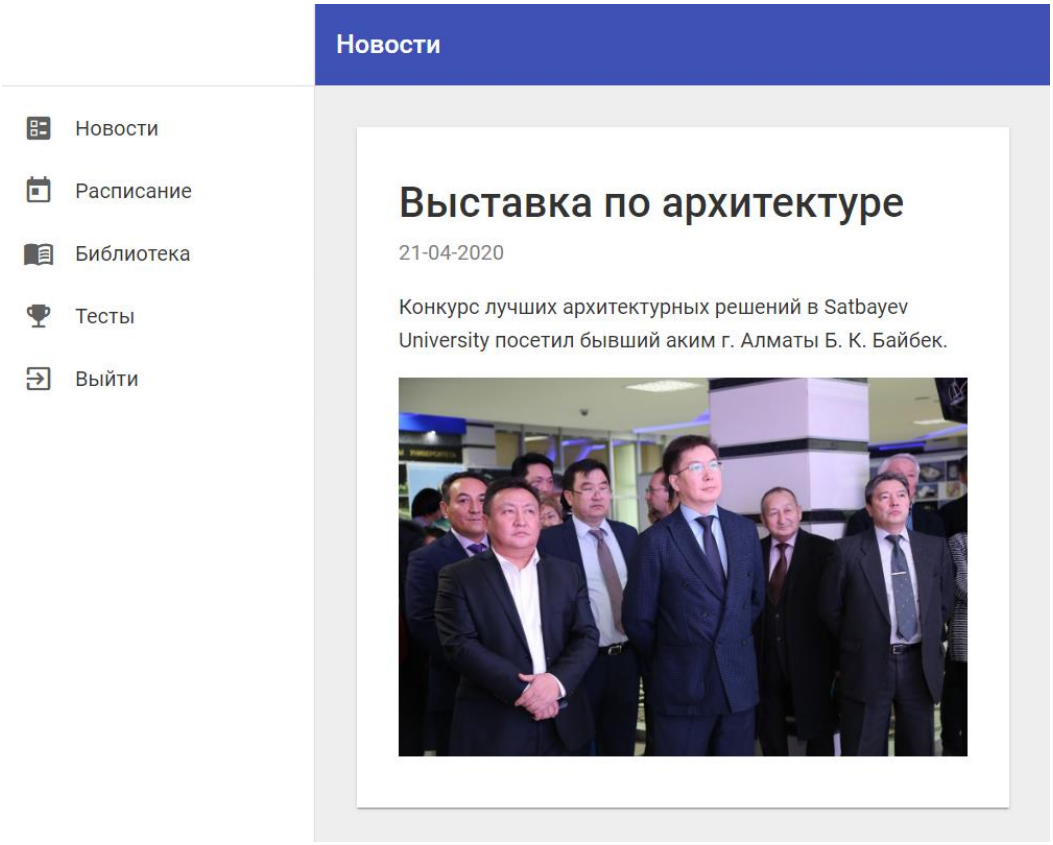

Рисунок 3.9 – Лента новостей

Как видно на рисунке 3.9, навигационная панель веб-приложения находится слева и позволяет пользователю переходить по таким страницам, как «Новости», «Расписание», «Библиотека» и «Тесты». К тому же, имеется функция выхода из системы – «Выйти».

Веб-приложение представляет возможность создавать собственный органайзер, позволяющий записывать действия, которые должен выполнить пользователь в определенный день (рисунок 3.10).

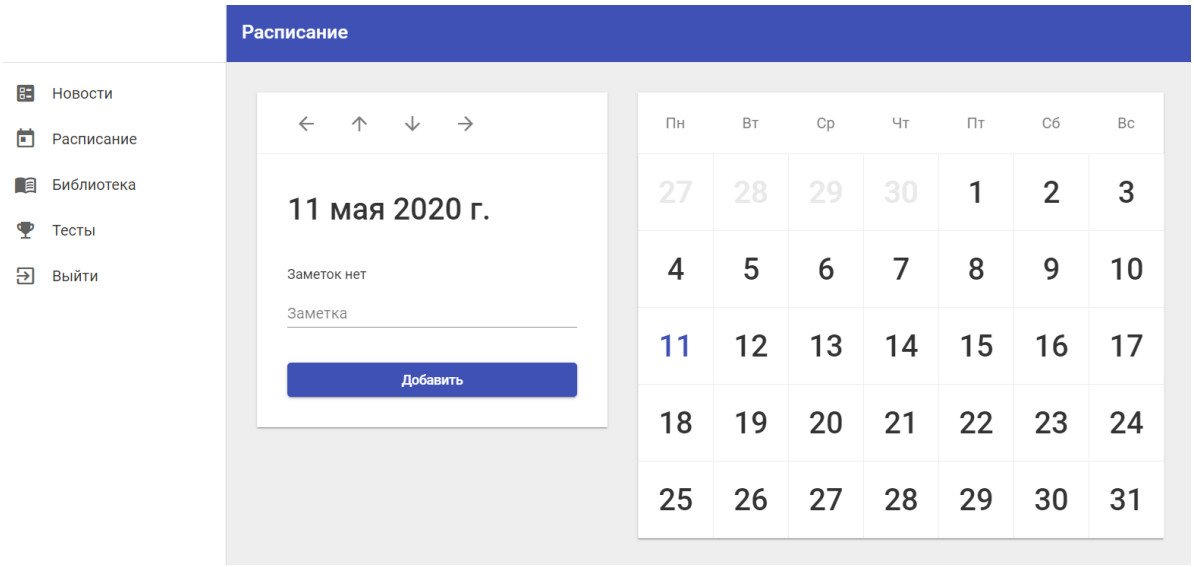

Рисунок 3.10 – Индивидуальное расписание

Электронная библиотека представляет список бесплатных книг, которые пользователь может загрузить на свой компьютер (рисунок 3.11).

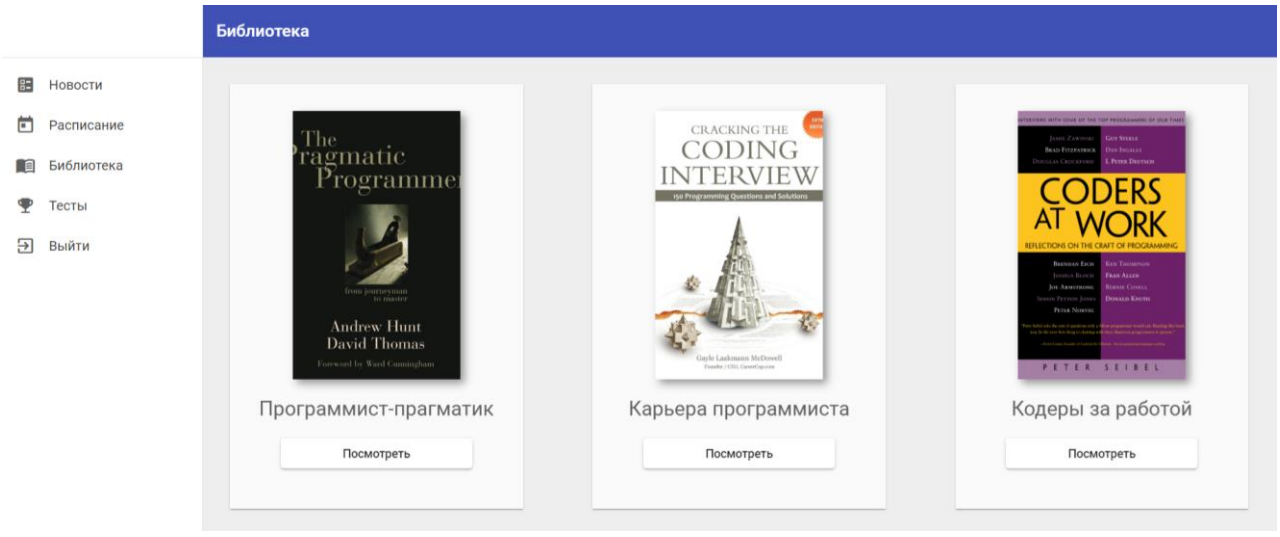

Рисунок 3.11 – Электронная библиотека

Если пользователь нажмет на кнопку «Посмотреть», откроется представление, которое содержит информацию об определенной книге, а именно: жанр, автора и описание книги (рисунок 3.12).

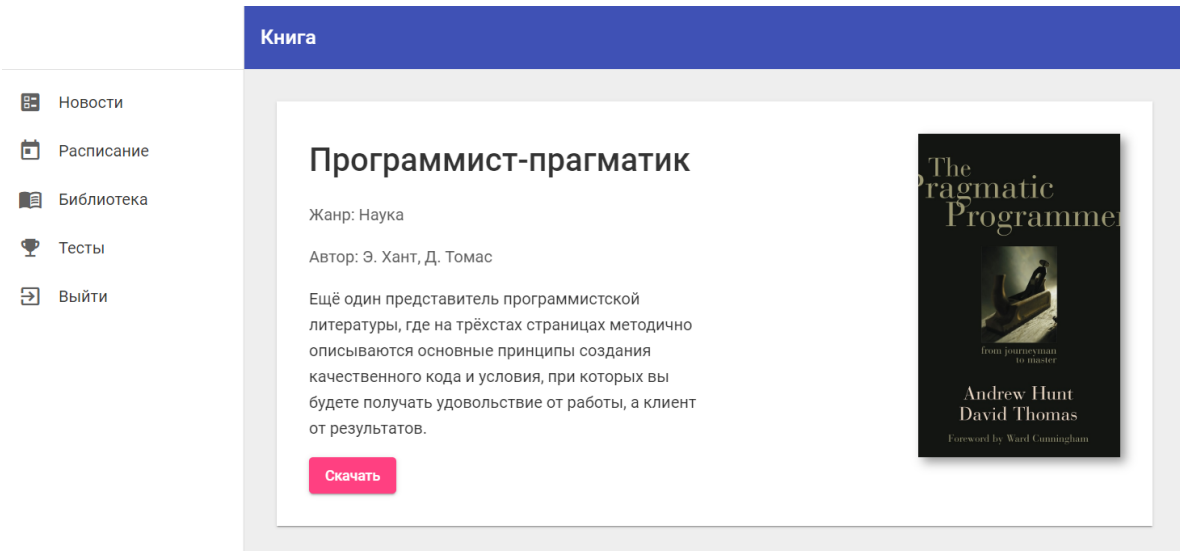

Рисунок 3.12 – Информация о книге

Страница «Тесты» содержит тесты, которые пользователь может пройти (рисунок 3.13).

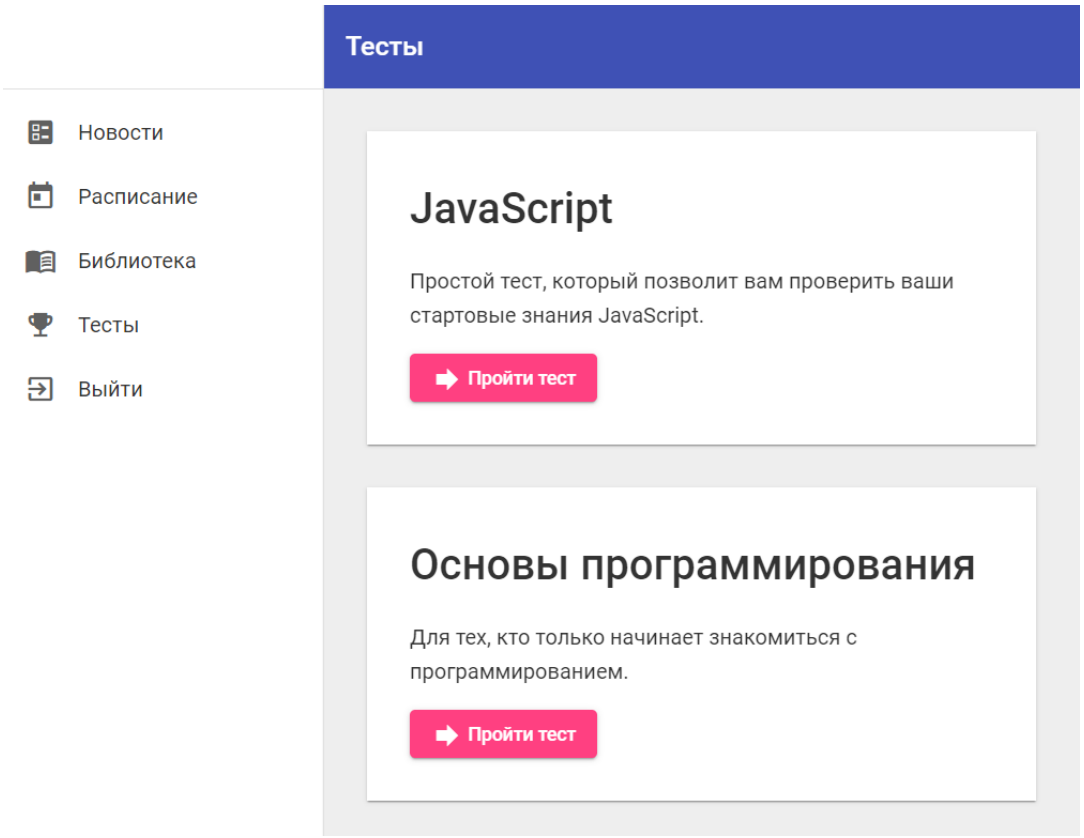

Рисунок 3.13 – Список тестов

При нажатии на кнопку «Пройти тест» пользователь перейдет на страницу, которая содержит список вопросов. Список динамически меняется в зависимости от того, ответил пользователь на предыдущий вопрос или нет (рисунок 3.14).

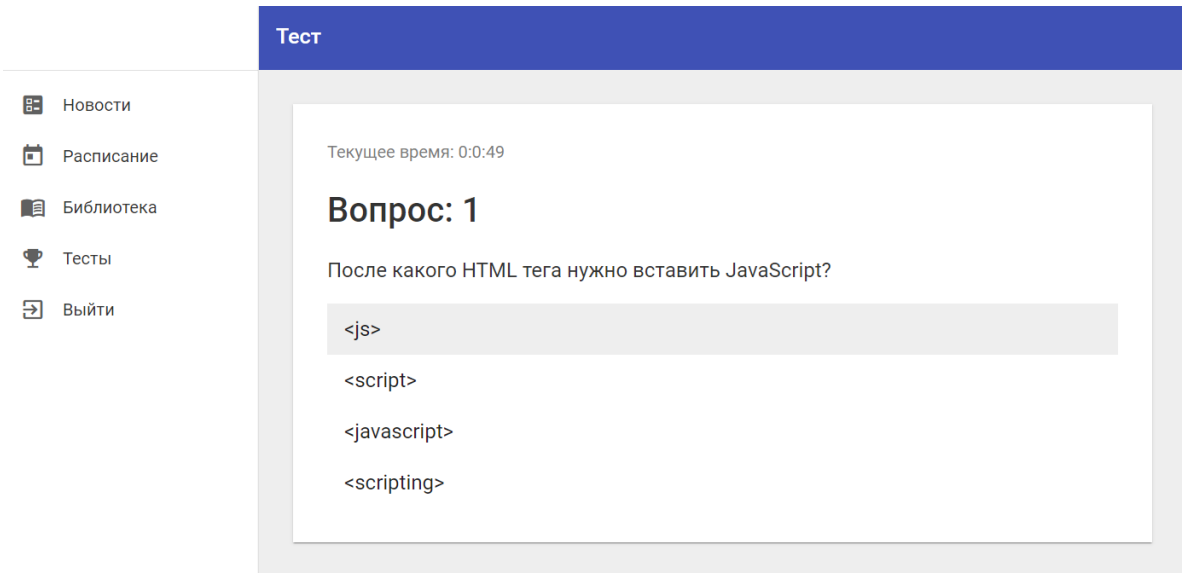

Рисунок 3.14 – Прохождение теста

При успешной сдаче теста, веб-приложение открывает представление «Результат», которое содержит результат теста и ответы пользователя, а также возможность повторного прохождения теста (рисунок 3.15).

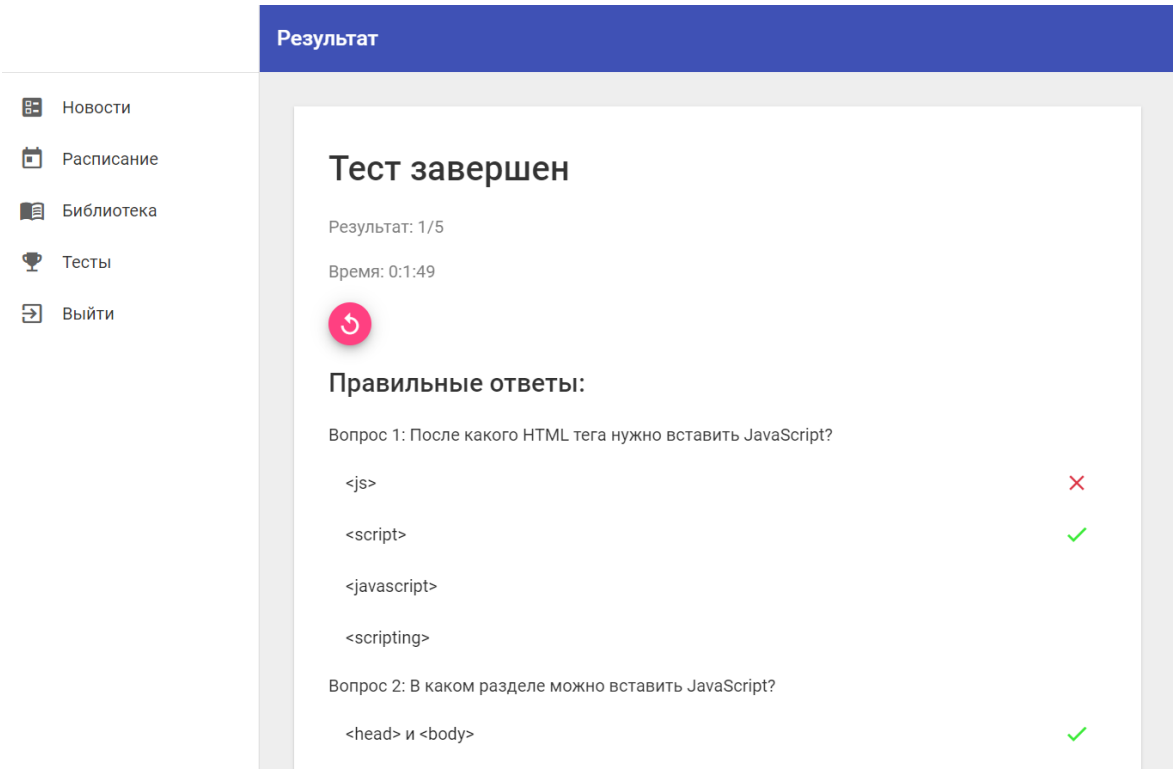

Рисунок 3.15 – Результат теста

В случае, если доступ к приложению имеет администратор, то на каждой странице приложения имеется возможность добавления и удаления контента. К примеру, на рисунке 3.16 представлена страница электронной библиотеки от лица администратора.

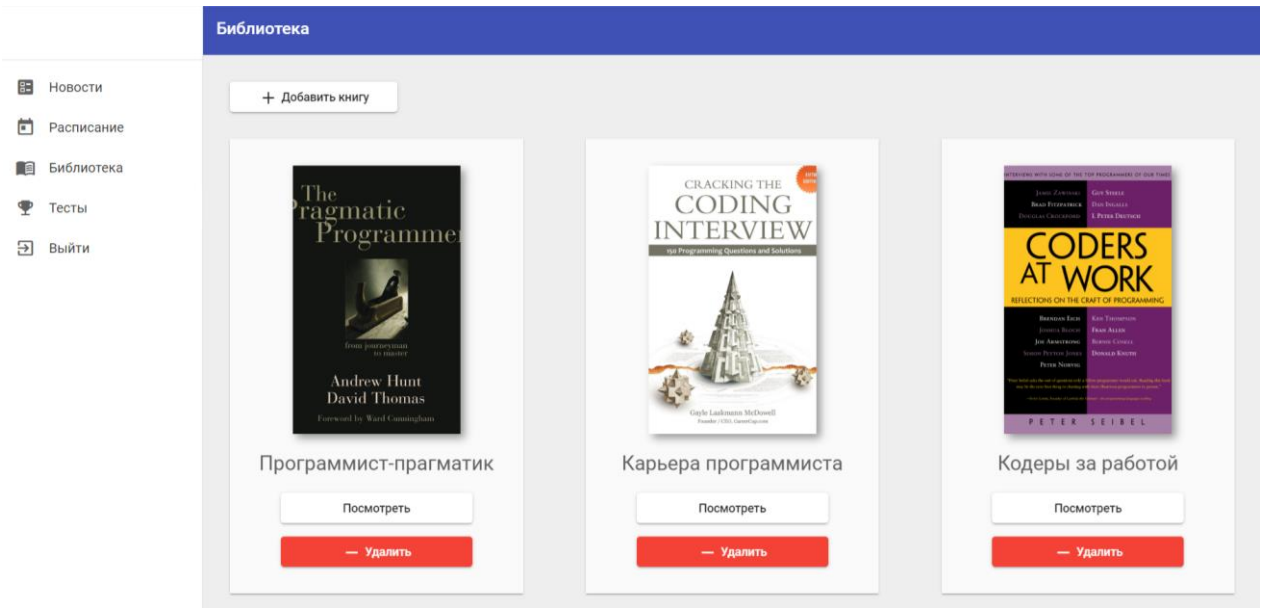

Рисунок 3.16 – Административные функции библиотеки

Более того, администратору доступно посещение страницы «Тест-администрирование», которую не может видеть обычный пользователь. Данная страница содержит список вопросов определенного теста с правильными вариантами ответов (рисунок 3.17).

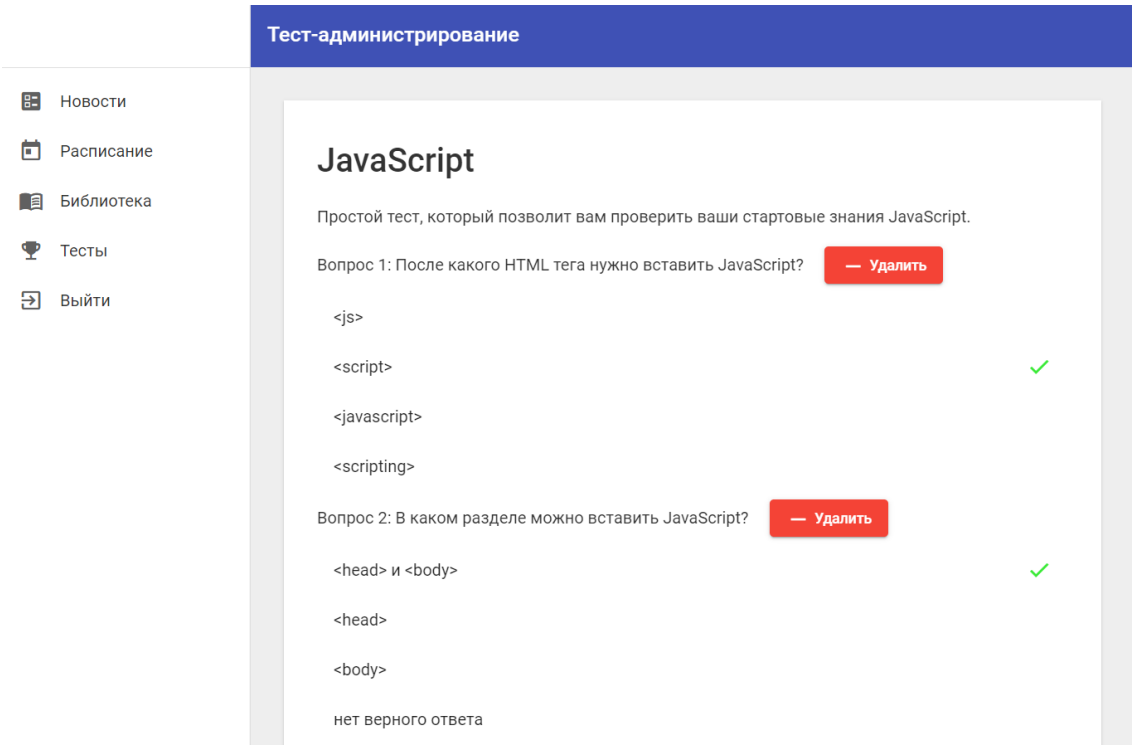

Рисунок 3.17 – Тест-администрирование

Учитывая разработанные сервисы, веб-приложение позволяет находиться в курсе последний событий в области образования, вести собственный дневник действий, скачивать электронные книги, а также проходить тесты для проверки собственных знаний.

ЗАКЛЮЧЕНИЕ

В данной дипломной работе было разработано веб-приложение под названием «Студент», позволяющее сделать электронное образование в университетах более доступным. При разработке программного обеспечения были учтены все современные стандарты в веб-сфере и требования, указанные в постановке задач.

Были выполнены следующие действия:

– анализ актуальных архитектур веб-приложений;

– выбор инструментов проектирования и разработки ПО;

– проектирование ИС с помощью ER и UML-диаграмм;

– создание веб-приложения.

Для реализации серверной части приложения были применены такие средства, как Spring Framework, Hibernate и Maven, а для клиентской части – Angular, NPM и HTML вместе с CSS. Хранение данных осуществлялось с помощью базы данных PostgreSQL, которая использует объектно-реляционную модель данных.

Для обеспечения безопасности веб-приложения был использован стандарт JWT, который предотвращает несанкционированный доступ к веб-сайту. JWT-токен создается на сервере при входе пользователя в систему и передается клиенту для хранения в браузере. При каждом обращении к API сгенерированный токен проверяет учетные данные и роли пользователя.

В работе также было описано аппаратное и программное обеспечение информационной системы, на которой следует развернуть разработанное веб-приложение. Было решено разделить систему на сервер приложений и сервер базы данных. В качестве операционной системы была выбрана Linux Ubuntu – надежная и стабильная ОС, предназначенная для размещения небольших приложений.

Таким образом, дипломная работа предлагает развитие электронного обучения в высших учебных заведениях посредством разработанного программного обеспечения, которое предлагает такие функции, как просмотр образовательных новостей, ведение собственного органайзера, чтение электронных ресурсов, а также прохождение онлайн-тестов.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1 Болкунов И. Электронное обучение: проблемы, перспективы, задачи [Электронный ресурс]. / И. Болкунов – Евпаторийский институт социальных наук, 2016. – Режим доступа: https://cyberleninka.ru/article/n/elektronnoe-obuchenie-problemy-perspektivy-zadachi.

2 Аллен М. E-learning: как сделать электронное обучение понятным [Текст]. / М. Аллен – Альпина Паблишер, 2016. – 196 с.

3 Тихомирова Е. Живое обучение. Что такое e-learning и как заставить его работать [Текст]. / Е. Тихомирова – Альпина Паблишер, 2020. – 238 с.

4 Прохорова А. Классификация и различия современных СУБД [Электронный ресурс]. / А. Прохорова – Ростовский государственный экономический университет, 2016. – Режим доступа: https://www.fundamental-research.ru/ru/article/view?id=40931.

5 Волков А. Изучаем PostgreSQL 10 [Текст]. / А. Волков, Д. Салахалдин – ДМК Пресс, 2019. – 400 с.

6 Егорова А. Информационные системы: методы и средства проектирования [Электронный ресурс]. / А. Егорова – Московский государственный технический институт, 2006. – Режим доступа: https://cyberleninka.ru/article/n/informatsionnye-sistemy-metody-i-sredstva-proektirovaniya.

7 Малыхина М. Базы данных. Основы, проектирование, использование [Текст]. / М. Малыхина – БХВ-Петербург, 2006. – 528 с.

8 Уоллс К. Spring в действии [Текст]. / К. Уоллс – ДМК Пресс, 2015. – 754 с.

9 Яков Ф. Angular и TypeScript. Сайтостроение для профессионалов [Текст]. / Ф. Яков, А. Моисеев – Питер, 2018. – 464 с.

10 Бобров А. Повышение надежности сервера [Электронный ресурс]. / А. Бобров – Институт комплексной безопасности и специального приборостроения, 2018. – Режим доступа: https://cyberleninka.ru/article/n/povyshenie-nadyozhnosti-servera.

ПРИЛОЖЕНИЕ А

Листинг А.1 – Метод авторизации

```
@PostMapping("/signin")
public ResponseEntity<?> authenticateUser(@Valid @RequestBody SignInForm
signInRequest) {
	Authentication authentication = authenticationManager.authenticate(new
UsernamePasswordAuthenticationToken(signInRequest.getUsername(),
signInRequest.getPassword()));

	SecurityContextHolder.getContext().setAuthentication(authentication);
	String jwt = jwtProvider.generateJwtToken(authentication);
	UserDetails userDetails = (UserDetails) authentication.getPrincipal();

	return ResponseEntity.ok(new JwtResponse(jwt, userDetails.getUsername(),
userDetails.getAuthorities()));
}
```

Листинг А.2 – Метод регистрации

```
@PostMapping("/signup")
public ResponseEntity<?> registerUser(@Valid @RequestBody SignUpForm
signUpRequest) {
	if (userRepository.existsByUsername(signUpRequest.getUsername())) {
		return new ResponseEntity<>(new ResponseReport("Fail -> Username
	is already taken!"), HttpStatus.BAD_REQUEST);
	}

	if (userRepository.existsByEmail(signUpRequest.getEmail())) {
		return new ResponseEntity<>(new ResponseReport("Fail -> Email is
	already in use!"), HttpStatus.BAD_REQUEST);
	}

	User user = new User(signUpRequest.getUsername(),
signUpRequest.getName(), signUpRequest.getSurname(), signUpRequest.getEmail(),
passwordEncoder.encode(signUpRequest.getPassword()));

	Set<String> strRoles = signUpRequest.getRoles();
	Set<Role> roles = new HashSet<>();

	strRoles.forEach(role -> {
		if ("admin".equals(role)) {
			Role adminRole =
		roleRepository.findByName(RoleName.ROLE_ADMIN).
```

```
                orElseThrow(() -> new RuntimeException("Fail! -> Cause: User
            Role not found"));
                roles.add(adminRole);

            } else {
                Role userRole =
            roleRepository.findByName(RoleName.ROLE_USER)
                .orElseThrow(() -> new RuntimeException("Fail! -> Cause: User
            Role not found"));
                roles.add(userRole);
            }
        });

        user.setRoles(roles);
        userRepository.save(user);

        return new ResponseEntity<>(new ResponseReport("User registered
successfully!"), HttpStatus.OK);
}
```

ПРИЛОЖЕНИЕ Б

Листинг Б.1 – Сущность News

```
@Entity
public class News {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private Long id;

	@NotBlank
	@Size(max = 50)
	private String title;

	@NotBlank
	@Size(max = 50)
	private String text;

	@NotBlank
	@Size(max = 50)
	private String date;

	private String imageName;

	public News() { }

	public News(String title, String text, String date) {
		this.title = title;
		this.text = text;
		this.date = date;
	}

	//геттеры и сеттеры
}
```

Листинг Б.2 – Репозиторий NewsRepository

```
@Repository
public interface NewsRepository extends JpaRepository<News, Long> { }
```

Листинг Б.3 – GET-метод контроллера NewsController

```
@GetMapping
public ResponseEntity<?> getAllNews() {
	List<News> allNews = newsRepository.findAll();
	return new ResponseEntity<>(allNews, HttpStatus.OK);
}
```

Листинг Б.4 – POST-метод контроллера NewsController

```
@PreAuthorize("hasRole('ROLE_ADMIN')")
@PostMapping
public ResponseEntity<?> createNews(@RequestParam String title, @RequestParam
String text, @RequestParam String date, @RequestParam("image") MultipartFile
image) throws IOException {
	News news = new News(title, text, date);

	if (image != null && !image.getOriginalFilename().isEmpty()) {
		File uploadDir = new File(uploadPath + "/images");

		if (!uploadDir.exists()) {
			uploadDir.mkdir();
		}

		String uuid = UUID.randomUUID().toString();
		String imageName = uuid + "." + image.getOriginalFilename();
		image.transferTo(new File(uploadDir + "/" + imageName));
		news.setImageName(imageName);
	}

	return ResponseEntity.ok(newsRepository.save(news));
}
```

Листинг Б.5 – DELETE-метод контроллера NewsController

```
@PreAuthorize("hasRole('ROLE_ADMIN')")
@DeleteMapping("/{id}")
public ResponseEntity<?> deleteNews(@PathVariable Long id) {
	News news = newsRepository.findById(id)
		.orElseThrow(() -> new EntityNotFoundException("Fail! -> News not
	found!"));

	File image = new File(uploadPath + "/images/" + news.getImageName());

	if (image.exists()) {
		image.delete();
	}

	newsRepository.delete(news);
	return new ResponseEntity<News>(HttpStatus.NO_CONTENT);
}
```

ПРИЛОЖЕНИЕ В

Листинг В.1 – Интерфейс News

```
export interface News {
	id?: number,
	title: string,
	text: string,
	date: string,
	imageName: string
}
```

Листинг В.2 – Сервис NewsService

```
@Injectable({ providedIn: 'root' })
export class NewsService {
	private listeners = new Subject<any>();
	static url = 'http://localhost:8080/news';
	constructor(private http: HttpClient) { }

	getNews(): Observable<News[]> {
		return this.http.get<News[]>(`${NewsService.url}`, httpOptions);
	}

	create(formData: FormData): Observable<FormData> {
		return this.http.post<FormData>(`${NewsService.url}`, formData);
	}

	delete(news: News): Observable<void> {
		return this.http.delete<void>(`${NewsService.url}/${news.id}`,
			httpOptions);
	}

	listen() {
		return this.listeners.asObservable();
	}

	filter(filterBy: string) {
		this.listeners.next(filterBy);
	}
}
```

Листинг В.3 – Компонент NewsComponent

```
@Component({ selector: 'app-news', templateUrl: './news.component.html', styleUrls:
['./news.component.scss'] })
export class NewsComponent implements OnInit {
```

Продолжение приложения В

```
news: News[] = [];
authority: string;
constructor(private newsService: NewsService, private dialog: MatDialog, private token: TokenService, private title: Title) {
    this.title.setTitle('Новости');
    this.newsService.listen().subscribe(
        response => {
            console.log(response);
            this.getNews();
        }
    );
}

ngOnInit() {
    if (this.token.getToken()) {
        const roles = this.token.getAuthorities();
        roles.every(role => {
            if (role === 'ROLE_ADMIN') {
                this.authority = 'admin';
                return false;
            }
            this.authority = 'user';
            return true;
        });
    }
    this.getNews();
}

getNews() {
    this.newsService.getNews().subscribe(
        response => {
            this.news = response.sort((a, b) => {
                return b.id - a.id;
            });
        }
    );
}

deleteNews(news: News) {
    this.newsService.delete(news).subscribe(
        response => {
```

```
                        this.getNews();
                    }
                );
        }

        onAdd() {
                const dialogConfig = new MatDialogConfig();
                dialogConfig.disableClose = true;
                dialogConfig.autoFocus = true;
                dialogConfig.width = '50%';
                this.dialog.open(NewsCreateComponent, dialogConfig);
        }
}
```

Листинг В.4 – HTML-шаблон списка новостей

```
<app-main-nav>
        <button mat-raised-button style="width: 200px; margin: 2rem 2rem 0 2rem;"
*ngIf="authority === 'admin'" (click)="onAdd()">
                <mat-icon>add</mat-icon> Добавить новость
        </button>
        <div class="news-wrapper">
                <div class="news" *ngFor="let element of news">
                        <div class="wrapper" style="margin-bottom: 1rem;">
                                <div>
                                        <h3>{{ element.title }}</h3>
                                        <p style="color: #7d7d7d;">{{ element.date }}</p>
                                </div>
                                <button mat-mini-fab color="warn" *ngIf="authority ===
                        'admin'" (click)="deleteNews(element)">
                                        <mat-icon>delete</mat-icon>
                                </button>
                        </div>
                        <p style="margin-bottom: 1rem;">{{ element.text }}</p>
                        <img [src]="'http://localhost:8080/img/' + element.imageName">
                </div>
        </div>
</app-main-nav>
```

ПРИЛОЖЕНИЕ Г

Листинг Г.1 – Массив маршрутизации

```
const routes: Routes = [
	{ path: 'signin', component: LoginComponent },
	{ path: 'signup', component: RegisterComponent },
	{ path: 'news', component: NewsComponent, canActivate: [AuthGuard] },
	{ path: 'schedule', component: ScheduleComponent, canActivate: [AuthGuard]
},
	{ path: 'quizzes', component: QuizListComponent, canActivate: [AuthGuard] },

	{ path: 'quizzes/:id', component: QuizComponent, canActivate: [AuthGuard] },
	{ path: 'quizzes/management/:id', component: QuizManagementComponent,
canActivate: [AuthGuard] },
	{ path: 'result', component: ResultComponent, canActivate: [AuthGuard] },
	{ path: 'books', component: BooksComponent, canActivate: [AuthGuard] },
	{ path: 'books/:id', component: BookComponent, canActivate: [AuthGuard] },
	{ path: '', redirectTo: 'news', pathMatch: 'full' }
];
```

Листинг Г.2 – Класс, ограничивающий доступ незарегистрированных пользователей

```
@Injectable({ providedIn: ‘root’ })
export class AuthGuard implements CanActivate {
	// внедрение зависимостей

	canActivate(
		route: ActivatedRouteSnapshot,
		state: RouterStateSnapshot
	): boolean {
		if (!this.token.getToken()) {
			this.router.navigate([‘signin’]);
			return false;
		}
		return true;
	}
}
```