

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

Факультет компьютерных наук

Кафедра Программирования и информационных технологий

Курсовой проект

Веб-приложение для публикации и просмотра новостей
“Школьная новостная доска”

09.03.04 Программная инженерия

Обучающийся _____ Елфимова Е.В., 3 курс

Обучающийся _____ Стребкова О.В., 3 курс

Обучающийся _____ Мещеряков И.Г., 3 курс

Руководитель _____ А.В. Нужных, преподаватель

Воронеж 2020

Содержание

| | |
|---|-----------|
| ВВЕДЕНИЕ..... | 4 |
| 1. ПОСТАНОВКА ЗАДАЧИ..... | 5 |
| 2. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ..... | 6 |
| 2.1. Глоссарий..... | 6 |
| 2.2. Анализ существующих решений..... | 6 |
| 2.2.1. Сайт школы МБОУ СОШ имени Е.А. Болховитинова | 6 |
| 2.2.2. Дневник. ру - цифровая образовательная платформа | 7 |
| 2.3. Документы, на основании которых создается сайт..... | 8 |
| 2.4. Цели создания сайта | 8 |
| 2.5. Задачи, решаемые при помощи сайта..... | 8 |
| 2.6. UML диаграммы..... | 9 |
| 2.6.1. Диаграммы вариантов использования..... | 9 |
| 2.6.2. Диаграмма последовательности..... | 11 |
| 2.6.3. Диаграмма коммуникаций..... | 14 |
| 2.6.4. Диаграмма состояний..... | 17 |
| 2.6.5. Диаграмма развертывания | 18 |
| 2.6.6. Диаграмма деятельности..... | 19 |
| 2.6.7. Диаграмма объектов..... | 21 |
| 2.7. Схема базы данных | 22 |
| 2.8. Воронки | 23 |
| 3. РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ | 26 |
| 3.2. Обоснование архитектуры | 27 |
| 3.3. Реализация серверной части | 29 |
| 3.4. Интерфейс | 34 |
| 4. ТЕСТИРОВАНИЕ..... | 46 |
| ЗАКЛЮЧЕНИЕ..... | 48 |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ | 49 |
| ПРИЛОЖЕНИЕ 1..... | 50 |
| ПРИЛОЖЕНИЕ 2..... | 51 |

| | |
|--------------------------|-----------|
| ПРИЛОЖЕНИЕ 3..... | 52 |
| ПРИЛОЖЕНИЕ 4..... | 53 |
| ПРИЛОЖЕНИЕ 5..... | 55 |

ВВЕДЕНИЕ

На сегодняшний день мы привыкли получать любую информацию буквально в несколько кликов. Мы можем узнать все, от прогноза погоды до свежих новостей из дальних стран. Подобным образом обстоит дело и с информацией которую мы получаем от родных, друзей или коллег, мы используем для этого различные мессенджеры и социальные сети.

Но не везде для получения организационной информации удовлетворяющим вариантом является мессенджер или социальная сеть. Например учащимся школ удобно было бы просматривать организационную информацию от учителей онлайн. Но мессенджер не является удобным вариантом для этой цели, а социальная сеть слишком обладает слишком перегруженным функционалом.

В данной курсовой работе рассматривается проблема создания простого и не нагруженного излишней функциональностью, но обладающего всеми необходимыми функциями веб-приложения «Школьная новостная доска».

1. ПОСТАНОВКА ЗАДАЧИ

Цель курсовой работы - работая в команде, реализовать веб-приложение для просмотра, публикации и обсуждения школьных новостей, обладающее простым и интуитивно-понятным интерфейсом.

Приложение должно позволять учителю публиковать новости, редактировать их и удалять по истечению их актуальности. Также учитель может участвовать в обсуждении новости в комментариях. Для ученика приложение должно позволять просматривать новости и комментировать их.

Команда состоит из трех человек, среди которых распределены роли: разработчик, тестировщик, проектный менеджер и бизнес-аналитик. Управление проектом ведется по методологии Kanban. Этот метод разработки позволит каждому члену команды иметь представления о реальных проектах в сфере IT-бизнеса.

2. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

2.1. Глоссарий

В рамках данной системы для пользователя определены следующие роли:

- Администратор - авторизованный пользователь, осуществляющий информационную поддержку сайта от имени Заказчика;
- Учитель - авторизованный пользователь в системе, имеющий права на комментирование опубликованных новостей, публикацию и удалению своих новостей, а также удаление комментариев любых пользователей;
- Учащийся - авторизованный пользователь в системе, наделенный правами комментировать новости на доске;
- Новость - текстовая информация, опубликованная учителем;
- Роль - класс пользователей системы, обладающий определенным набором прав доступа;
- Неактуальная новость - новость считается неактуальной , когда учитель, опубликовавший ее на новостной школьной доске, больше не видит надобности в знании о ней для учеников;
- Соответствующая действительности информации - информация, опубликованная на доске учителем;
- Фреймворк - программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта;

2.2. Анализ существующих решений

2.2.1. Сайт школы МБОУ СОШ имени Е.А. Болховитинова

Действующий сайт школы №38. Предоставляет полезную информацию для учащихся школы №38, их родителей, а также поступающим в школу.

Достоинства:

- Разделение информации по категориям;
- Наличие версии “Для слабовидящих”;
- Возможность обратной связи для родителей.

Недостатки:

- Отсутствие возможности комментирования новостей учащимися.

2.2.2. Дневник. ру - цифровая образовательная платформа

“Дневник.ру” - единая электронная образовательная среда для учителей, учеников, родителей и органов государственного управления. Она предоставляет информацию об учебном плане, оценках и рейтинге учащихся.

Достоинства:

- Есть возможность просмотра оценок;
- Открыт доступ для родителей учащихся.

Недостатки:

- Перегруженная функциональность.

Рассмотрим процесс добавления новости.

Процесс добавления новости на сайте “Дневник.ру” проходит за 5 кликов. Для этого необходимо войти в систему, выбрать раздел “Моя школа”, написать текст для новости и выбрать “Отправить” для публикации новости на доске объявлений. Более подробно этот процесс отображен в диаграмме BPMN в Приложении 2.

Рассмотрим процесс создания нового пользователя.

“Моя школа” и перейти в раздел “Администрирование школы.” Далее учителю необходимо выбрать “Новый человек” для создания нового пользователя, затем выбрать, кого он хочет добавить: нового пользователя или изменить существующего, затем заполнить 35 текстовых полей с личной информацией пользователя (см. приложение 1), кликнуть на кнопку “Далее”, проверить правильность введенных данных и выбрать кнопку “Создать” для создания нового пользователя. Более подробно этот процесс отображен в диаграмме BPMN в Приложении 2.

2.3. Документы, на основании которых создается сайт

Устав муниципального бюджетного общеобразовательного учреждения средняя общеобразовательная школа с углубленным изучением отдельных предметов №38 имени Е. А. Болховитинова. (См. Приложение 3)

2.4. Цели создания сайта

- Упростить процесс распространения информации для учащихся (для добавления новости в нашем приложении требуется 3 клика вместо 5 на сайте Дневник.ру);
- Избежать распространения не соответствующей действительности информации между учащимися (Система предоставляет доступ к публикации новостей только учителям);
- Экономия времени учителей на распространение информации о предстоящих событиях. Так как максимальное число учащихся в классе равно 25 (Приказ от 30.09.2013 г. № 1015 «Об утверждении порядка организации и осуществления образовательной деятельности по основным общеобразовательным программам — образовательным программам начального общего, основного общего и среднего общего образования»), то чтобы обзвонить весь класс понадобится минимум 75 минут, если принять 3 минуты, как время одного телефонного разговора(без учета поиска номера учащегося и затрат времени на ожидание ответа).

2.5. Задачи, решаемые при помощи сайта

Система предназначена для публикации и просмотра новостей о проведении школьных событий. Основное назначение системы - предоставление учащимся школы информации о школьных событиях и возможность для учителей предоставлять информацию о школьных событиях для учащихся.

2.6. UML диаграммы

2.6.1. Диаграммы вариантов использования

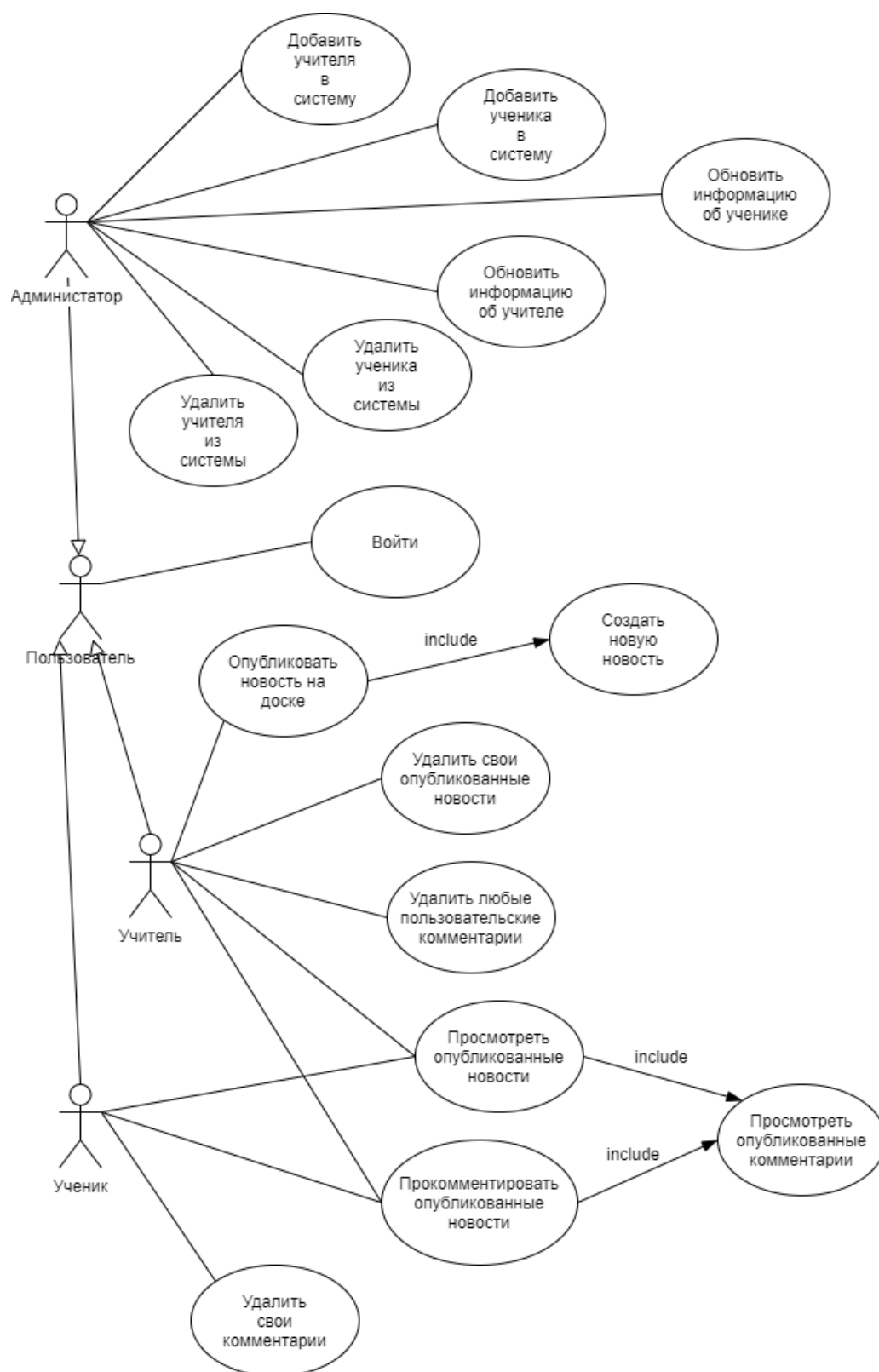


Рисунок 1 - Диаграмма вариантов использования

Рассмотрим диаграмму вариантов использования на рисунке 1.

Актер - пользователь.

Действия, которые может осуществлять пользователь (отношения ассоциации):

- Войти.

Пользователь может войти в систему как учитель, ученик или администратор. Рассмотрим действия каждого из них.

Актер - администратор.

Действия, который может осуществлять администратор (отношения ассоциации):

- Добавить учителя в систему;
- Добавить ученика в систему;
- Обновить информацию об ученике;
- Обновить информацию об учителе;
- Удалить ученика из системы;
- Удалить учителя из системы.

Актер - учитель.

Действия, который может осуществлять учитель (отношения ассоциации):

- Опубликовать новость на доске (отношение включения - создать новую новость);
- Удалить свои опубликованные новости;
- Удалить любые пользовательские комментарии;
- Просмотреть опубликованные новости (отношение включения - просмотреть опубликованные комментарии);
- Прокомментировать опубликованные новости (отношение включения - просмотреть опубликованные комментарии).

Актер - ученик.

Действия, которые может осуществлять учитель (отношения ассоциации):

- Просмотреть опубликованные новости (отношение включения - просмотреть опубликованные комментарии);
- Прокомментировать опубликованные новости (отношение включения - просмотреть опубликованные комментарии);
- Удалить свои комментарии.

2.6.2. Диаграмма последовательности

Диаграмма последовательности для удаления новости - рисунок 2.

Актер - пользователь.

После нажатия на кнопку удаления новости пользователем клиент отображает диалоговое окно для подтверждения удаления. Если пользователя подтверждает удаление, то клиент отправляет запрос об удалении новости на сервер, а сервер, в свою очередь, отправляет запрос в БД об удалении данных. Когда БД возвращает информацию об удалении, сервер возвращает эту информацию клиентской части приложения, которая отображает обновленную страницу новостей.

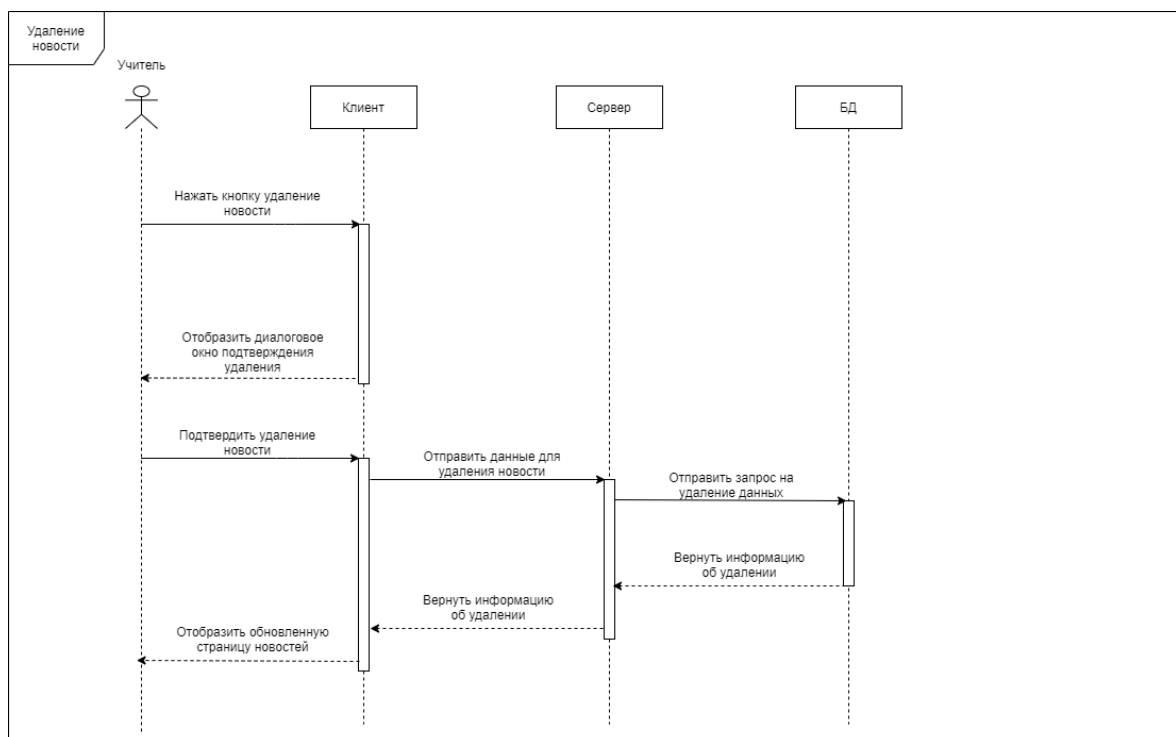


Рисунок 2 - Диаграмма последовательности для удаления новости

Диаграмма последовательности для добавления комментария - рисунок 3.

Актер - учитель или ученик.

После нажатия учителем или учеником кнопки добавления комментария клиент отображает форму добавления комментария. Далее учитель или ученик вводит комментарий в форму и подтверждает отправку. После этого клиент проверяет введенные данные и отправляет их на сервер, который проверяет права пользователя на добавление комментария, а затем отправляет запрос на добавление данных в БД. Когда БД возвращает информацию о добавлении, сервер возвращает эту информацию на клиентскую часть приложения, которая отображает информацию о добавлении комментария.

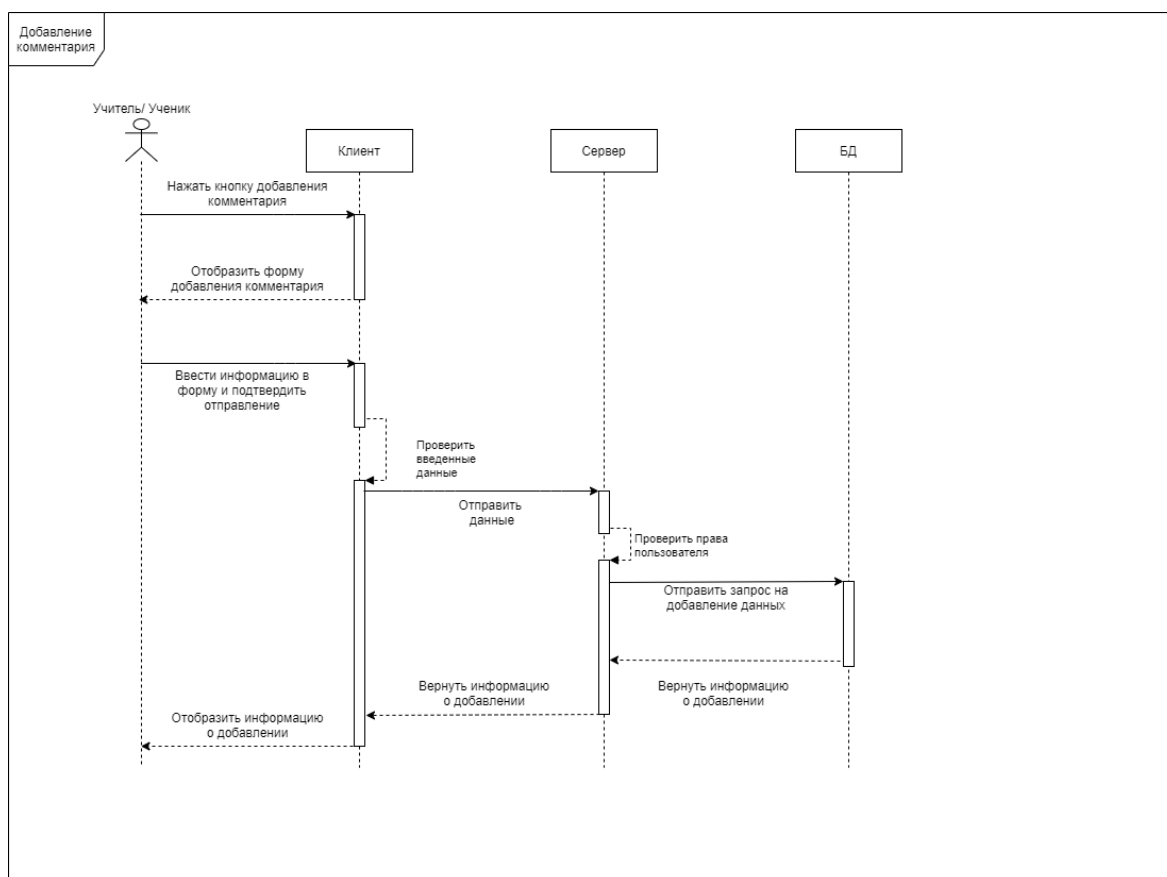


Рисунок 3 - Диаграмма последовательности для добавления комментария

Диаграмма последовательности для добавления новости - рисунок 4.

Актер - учитель.

После нажатия учителем кнопки добавления новости, клиентская часть приложения отображает форму добавления новости. Далее учитель вводит текстовую информацию в форму для добавления новости и подтверждает отправку. Затем клиент проверяет введенные данные и отправляет их на сервер, который проверяет права пользователя на добавление новости, после чего отправляет запрос на добавление новости в БД. После того, как БД возвращает информацию о добавлении, сервер возвращает эту информацию на клиентскую часть приложения, которая в свою очередь отображает информацию и добавлении.

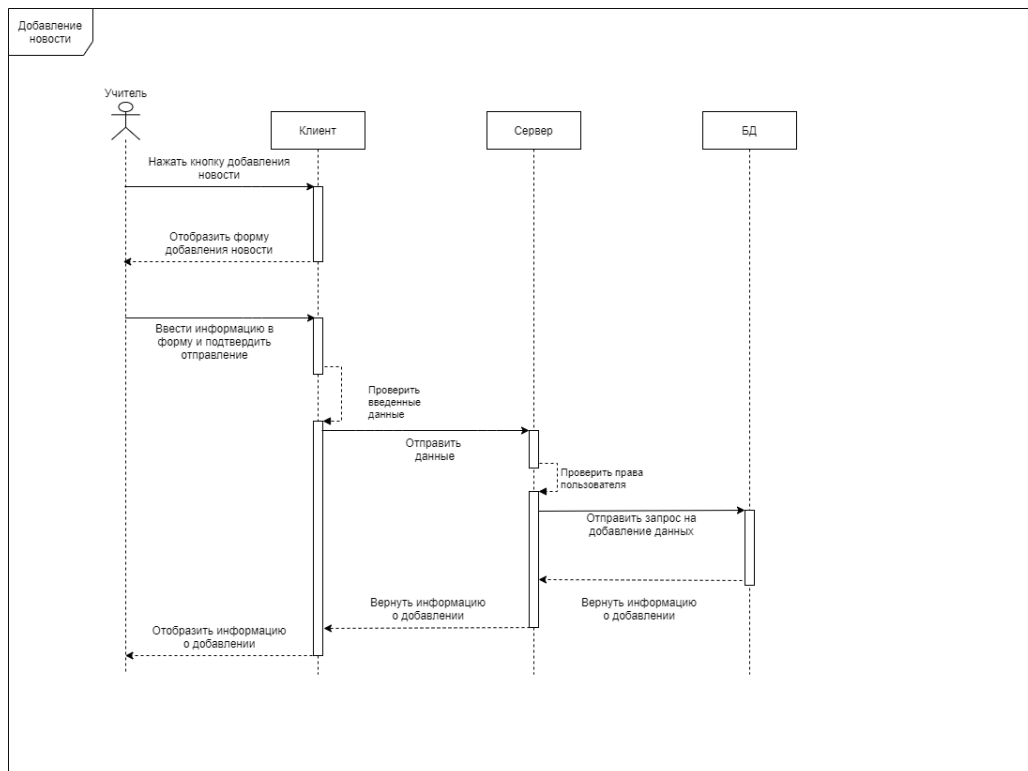


Рисунок 4 - Диаграмма последовательности для добавления новости

Диаграмма последовательности для добавления новости - рисунок 5.

Актёр - пользователь.

После ввода пользователем персонального логина и пароля, клиентская часть приложения отправляет данные на сервер, который запрашивает данные о пользователе в БД. Затем БД возвращает данные на сервер, который после создания JWT токена для данного пользователя, возвращает токен и информацию

о пользователе на клиентскую часть приложения. Далее клиент запрашивает у сервера данные для главной страницы с JWT токеном в заголовке, после чего сервер проверяет JWT токен и затем запрашивает данные у БД. Когда БД возвращает эти данные на сервер, он отправляет их на клиентскую часть приложения, которая, в свою очередь, отображает главную страницу приложения.

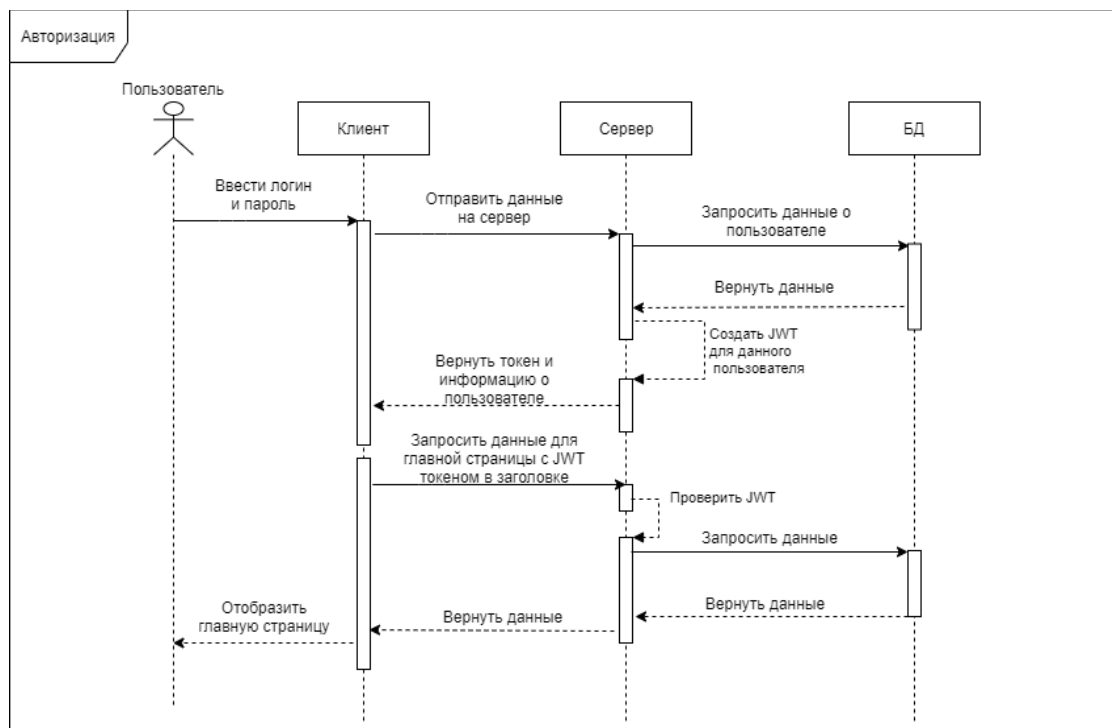


Рисунок 5 - Диаграмма последовательности для авторизации

2.6.3. Диаграмма коммуникаций

Диаграмма коммуникаций для авторизации - рисунок 6.

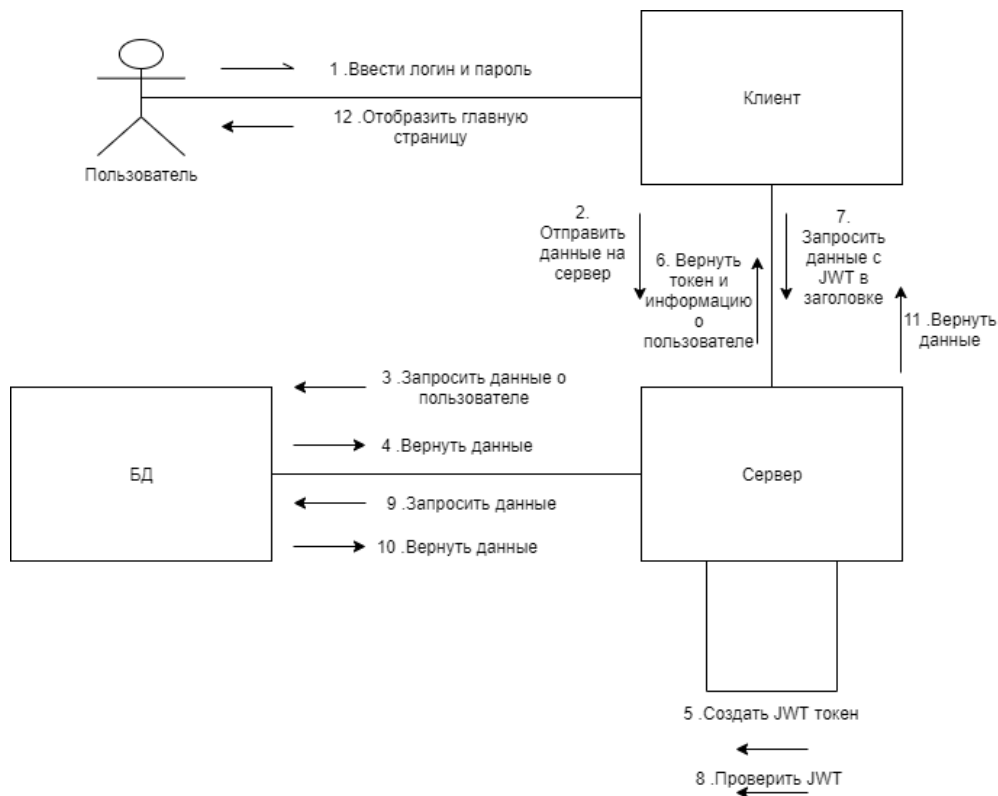


Рисунок 6 - Диаграмма коммуникаций для авторизации

Диаграмма коммуникаций для добавления поста - рисунок 7.

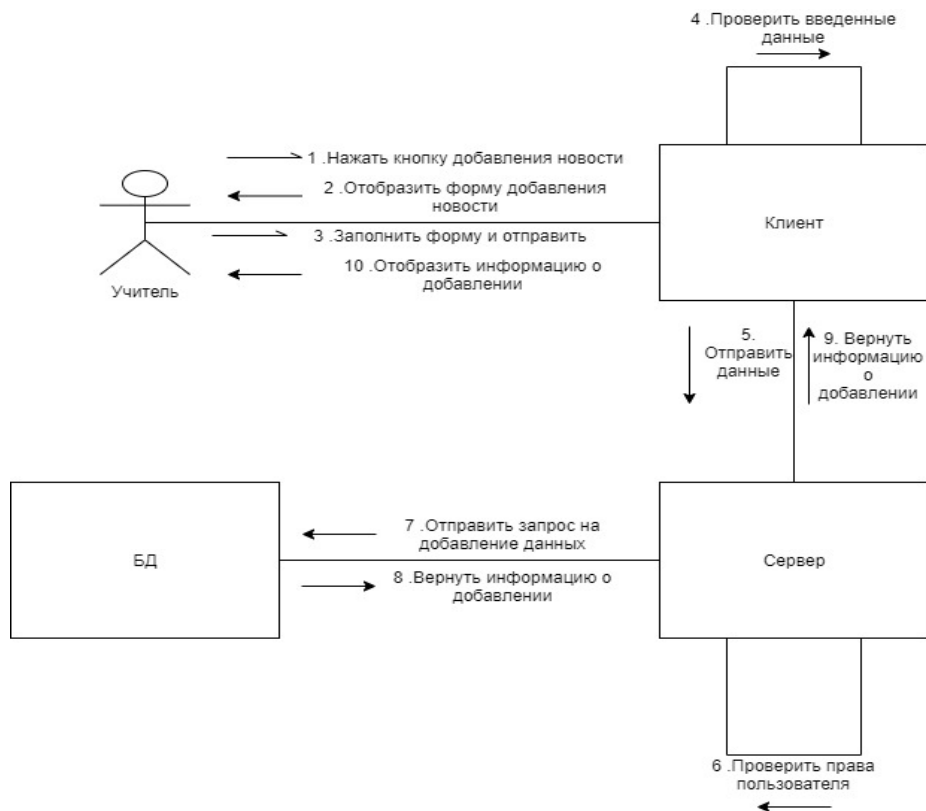


Рисунок 7 - Диаграмма коммуникаций для добавления поста

Диаграмма коммуникаций для удаления поста - рисунок 8.

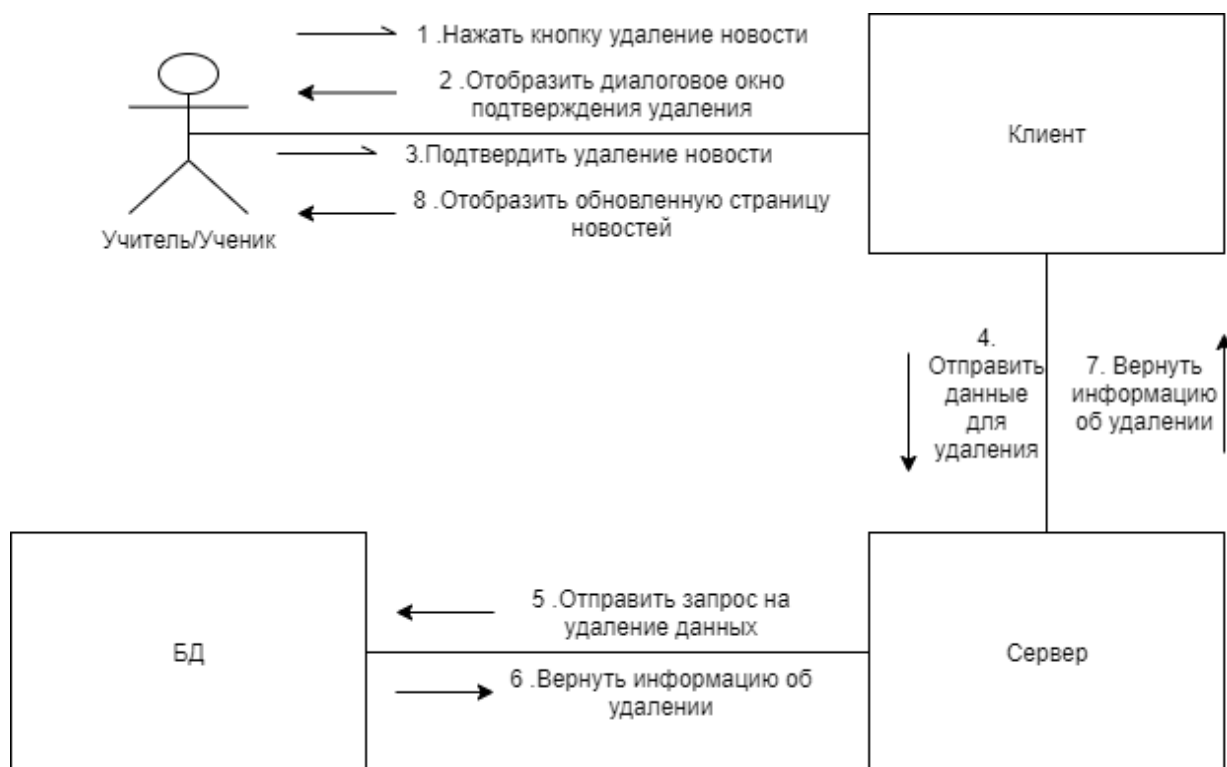


Рисунок 8 - Диаграмма коммуникаций для удаления поста

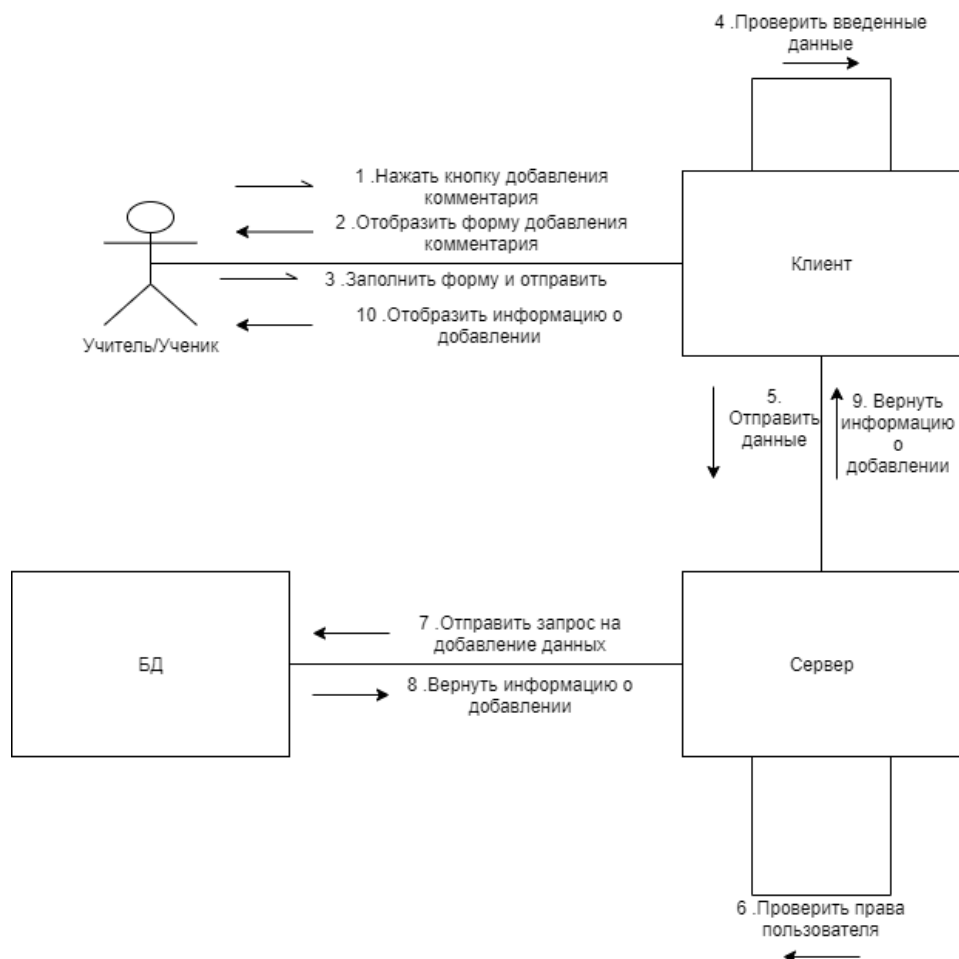


Рисунок 9 - Диаграмма коммуникаций для добавления комментария

Сообщения в диаграмме коммуникаций такие же, что и в диаграмме последовательности, поэтому их повторное описание не приводится.

2.6.4. Диаграмма состояний

Диаграмма состояний для авторизации - рисунок 10.

Диаграмма состояний отражает возможные состояния системы. Когда пользователь зашел в систему, она находится в состоянии ожидания ввода логина и пароля. После ввода пользователем этих данных, система переходит в состояние поиска логина и пароля. В зависимости от исхода этого состояние возможны две основные цепочки состояний.

Если данные, введенные пользователем, являются корректными, то система переходит в состояние отображения главной страницы приложения, в зависимости от роли пользователя.

Если пользователь ввел некорректные данные, система переходит в состояние ожидания ввода логина и пароля.

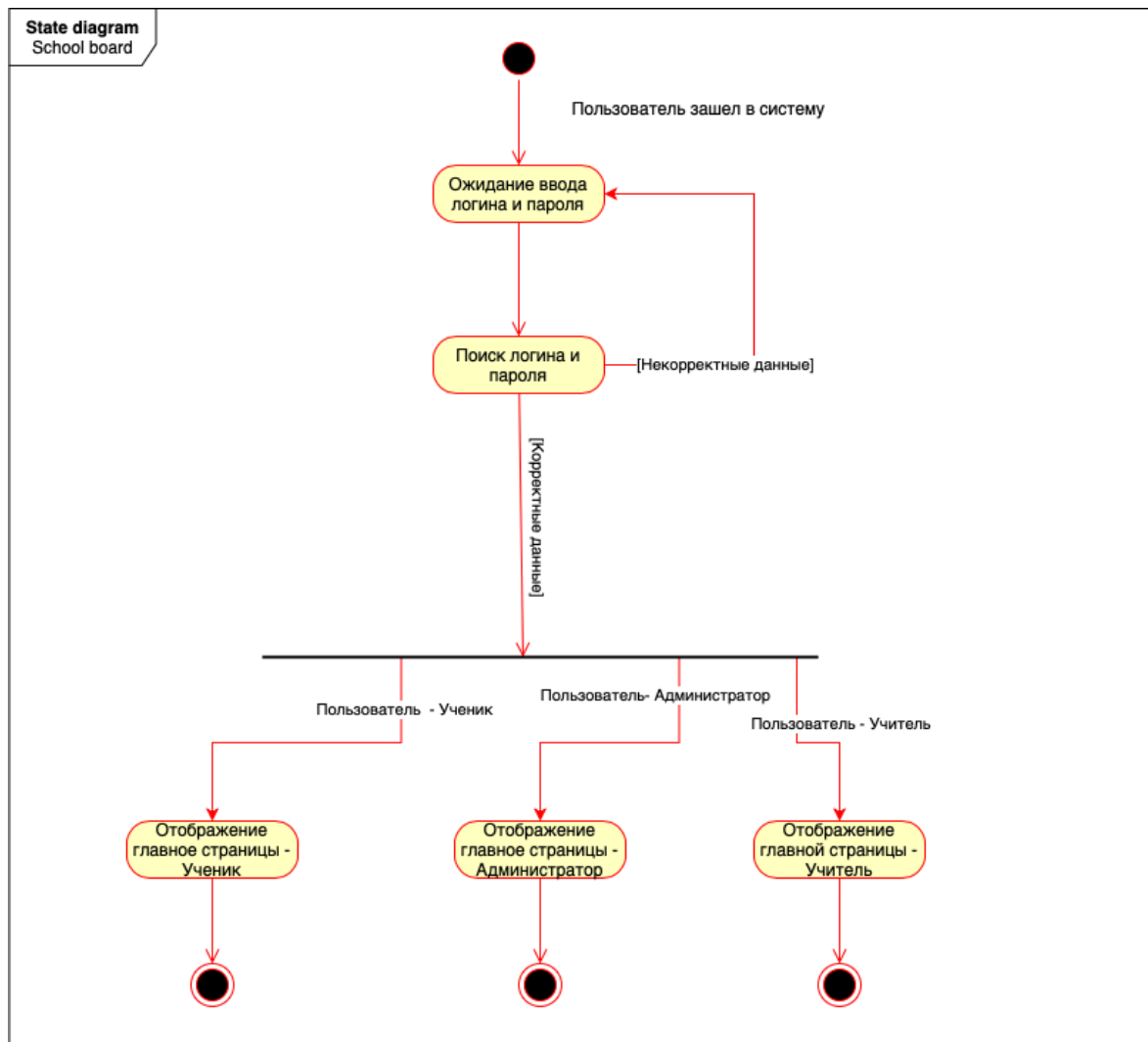


Рисунок 10 - Диаграмма состояний для авторизации пользователя.

2.6.5. Диаграмма развертывания

На рисунке 11 представлена диаграмма развертывания, которая отображает взаимосвязь между аппаратными компонентами приложения и их программными компонентами.

Для описываемого приложения, аппаратными компонентами являются компьютер пользователя и сервер. На узле сервера расположены бекэнд и фронтэнд приложения, а также база данных.

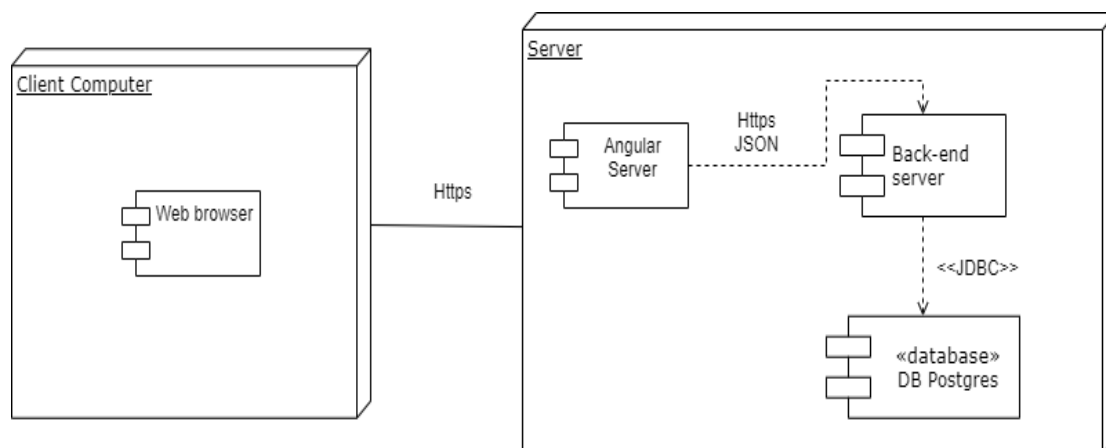


Рисунок 11 - Диаграмма развертывания

2.6.6. Диаграмма деятельности

На рисунке 12 представлена диаграмма деятельности для создания нового пользователя администратором.

После того, как страницы входа была открыта, администратор должен ввести свой логин и пароль, после чего система проверяет пароль, если он неверный, то у администратора запрашивается логин и пароль повторно. Если логин и пароль верны, то администратор перенаправляется в личный кабинет.

Далее администратор должен нажать на кнопку для создания нового пользователя. После чего система перенаправляет пользователя на страницу создания нового пользователя, где администратор должен выбрать роль пользователя, ввести имя пользователя, и добавить пароль, который администратор может ввести сам или выбрать создать пароль. Во втором случае система генерирует пароль и возвращает его на страницу создания пользователя. Далее администратор выбирает добавить пользователя, после чего система получает данные нового пользователя, а затем добавляет их. В завершение,

система отображает данные нового пользователя в личном кабинете администратора.

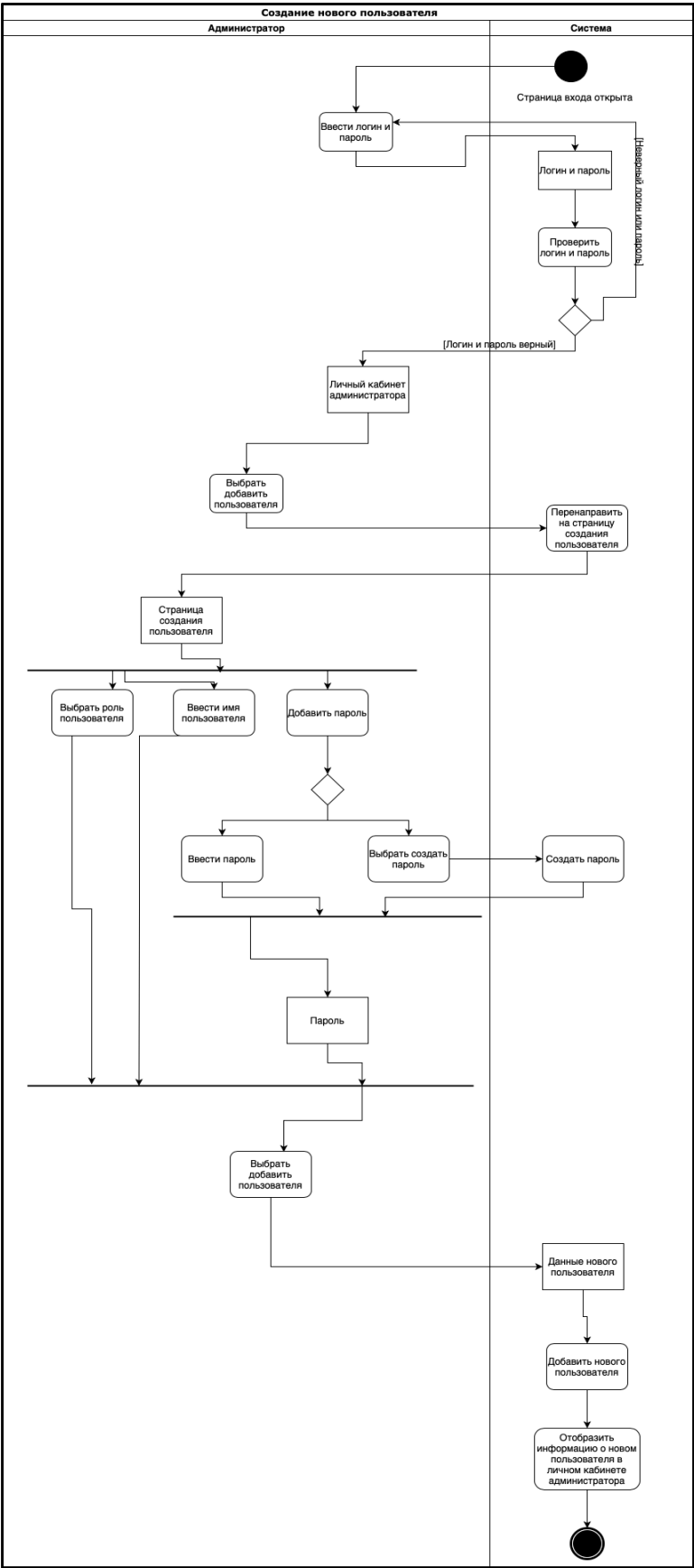


Рисунок 12 - Диаграмма деятельности для добавления нового пользователя.

2.6.7. Диаграмма объектов

На рисунке 13 представлена диаграмма объектов.

Пусть в некоторый момент времени в системе существуют следующие объекты:

1. Экземпляры класса “User” :
 - Пользователь “u”;
 - Пользователь “u1”;
 - Пользователь “u2”.
2. Экземпляры класса “Role”:
 - Роль “r1”, имеющая связь с пользователем “u”;
 - Роль “r2” имеющая связь с пользователем “u1” и “u2”.
3. Экземпляр класса “Post”:
 - Новость “p1”, имеющая связь с пользователем “u”.
4. Экземпляры класса “Comment”:
 - Комментарий “c1”, имеющий связь с постом “p1” и пользователем “u2”;
 - Комментарий “c2”, имеющий связь с постом “p1” и пользователем “u1”;
 - Комментарий “c3”, имеющий связь с постом “p1” и пользователем “u2”.

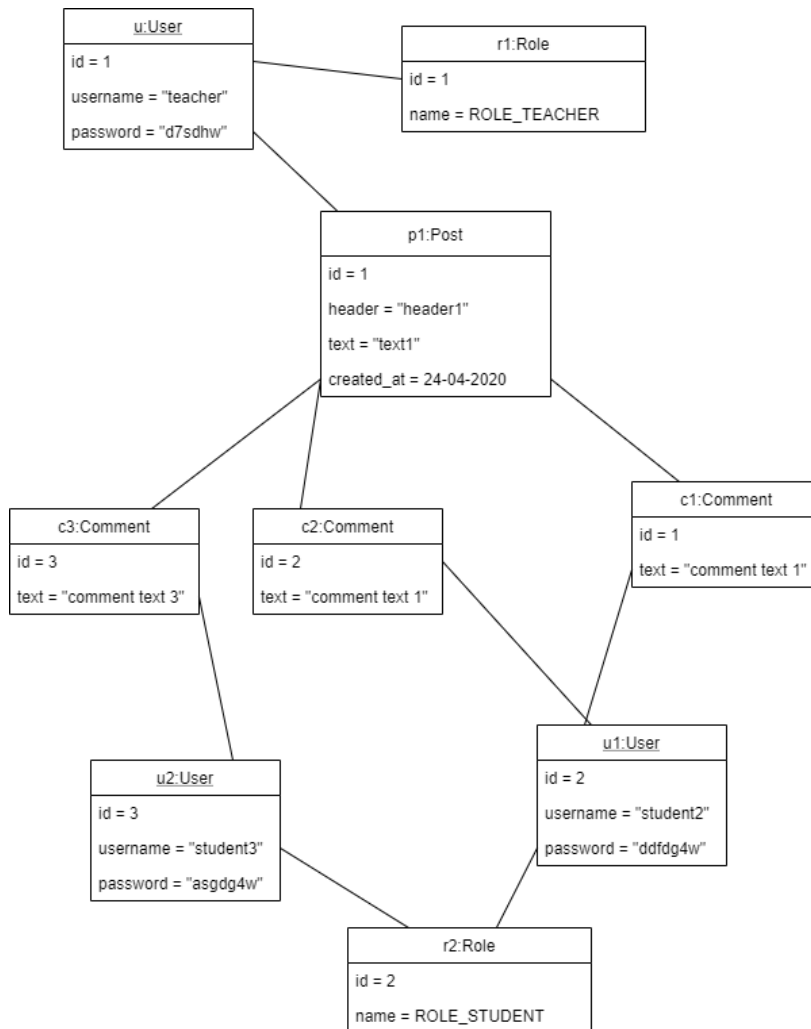


Рисунок 13 - Диаграмма объектов

2.7. Схема базы данных

На рисунке 14 представлена схема базы данных.

В базе данных содержится 5 таблиц.

Таблица “users” хранит информацию о пользователях системы, это id, имя пользователя и пароль. Id является первичным ключом. А поля “username” и “password” являются обязательными.

Таблица “roles” хранит информацию о ролях пользователей, это id и название роли. Id - первичный ключ, а поле “name” является обязательным.

Эти две таблицы (users, roles) соединяет таблица “users_roles”, которая хранит первичный ключ пользователя и первичный ключ роли.

Таблица “posts” хранит информацию о новостях. Это id новости, заголовок, текст новости, дата создания новости и id пользователя, создавшего новость. “Id” является первичным ключом, а поле “user_id” - внешним.

Таблица “comments” содержит информацию о комментариях пользователей. Это id комментария, текст комментария, id пользователя, оставившего комментарий и id поста, под которым оставили комментарий.

После “id” является первичным ключом, а поле “user_id” и “post_id” - внешним.

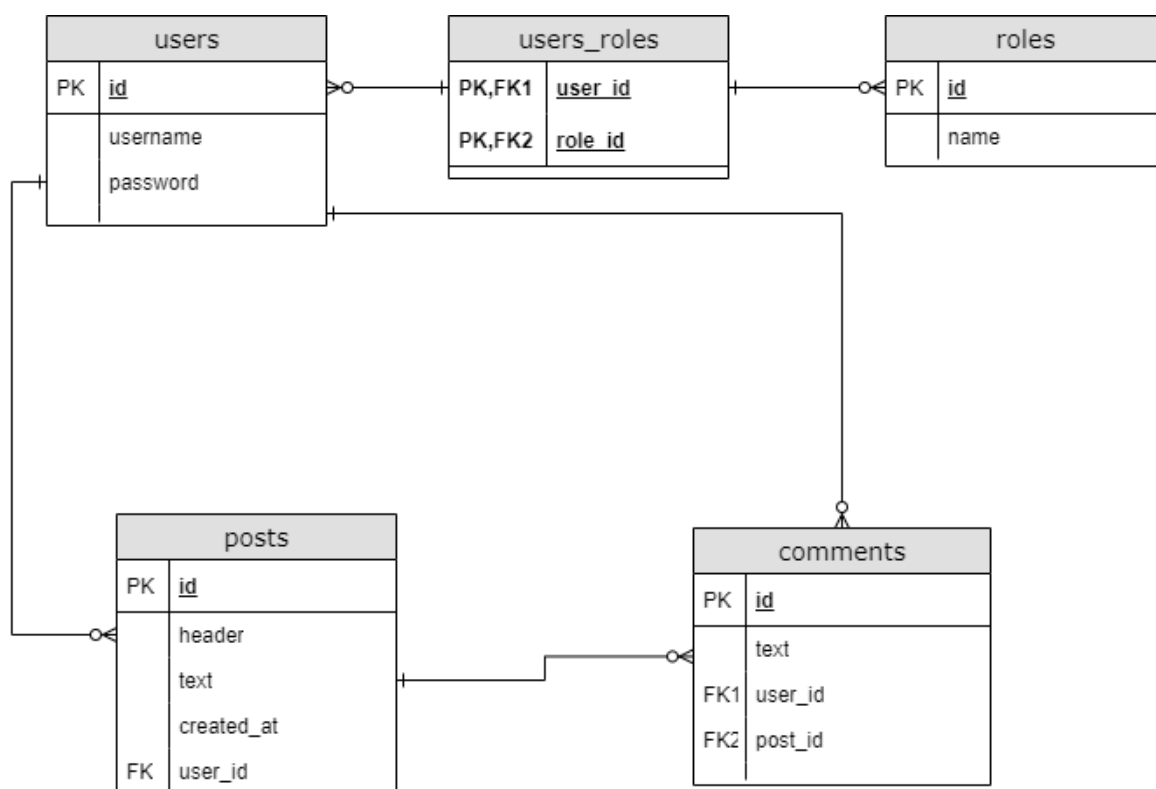


Рисунок 14 - Схема базы данных

2.8. Воронки

Были сформированы 3 различных воронки для анализа использования системы:

Добавление новости для учителя:

Цель создания воронки - определить какое число учителей пользуется системой по назначению.

Воронка состоит из двух шагов:

1. Нажать на кнопку добавления новости;
2. Нажать на кнопку публикации новости.

Редактировать цель

Название
Добавление новости для учителя

☐ Ретаргетинг

Тип условия:

Количество просмотров | Посещение страниц | JavaScript-событие | **Составная цель**

Задайте до пяти шагов, которые должны быть последовательно выполнены во время одного визита. Шагами могут быть просмотры страниц или JavaScript-события. [Подробнее](#). [Диагностика проблем](#).

Шаг 1 Создать новость

Условие событие: и... create

Код цели для сайта
ym(62161168,'reachGoal','create') Скопировать код +

Шаг 2 Клик по кнопке "Опубликовать"

Условие событие: и... post

Код цели для сайта
ym(62161168,'reachGoal','post') Скопировать код +

Рисунок 15 - Этапы для воронки “добавление новости для учителя”

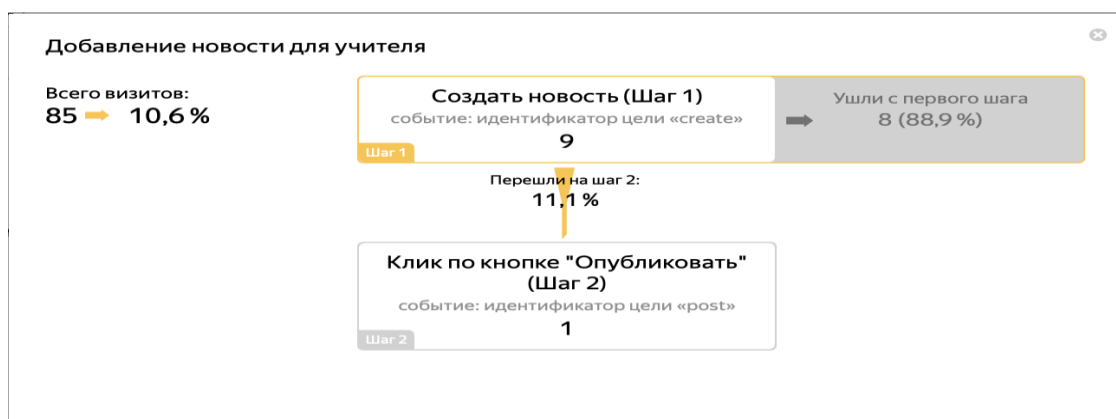
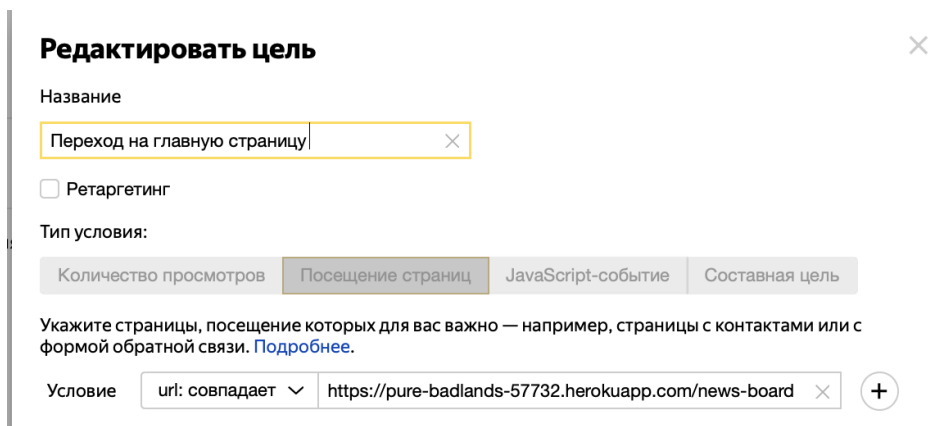


Рисунок 16 - воронка для добавления новости для учителя

Переход на главную страницу:

Цель создания воронки - определить сколько пользователей заходит в систему.



Редактировать цель [X]

Название
Переход на главную страницу [X]

☐ Ретаргетинг

Тип условия:

Количество просмотров | **Посещение страниц** | JavaScript-событие | Составная цель

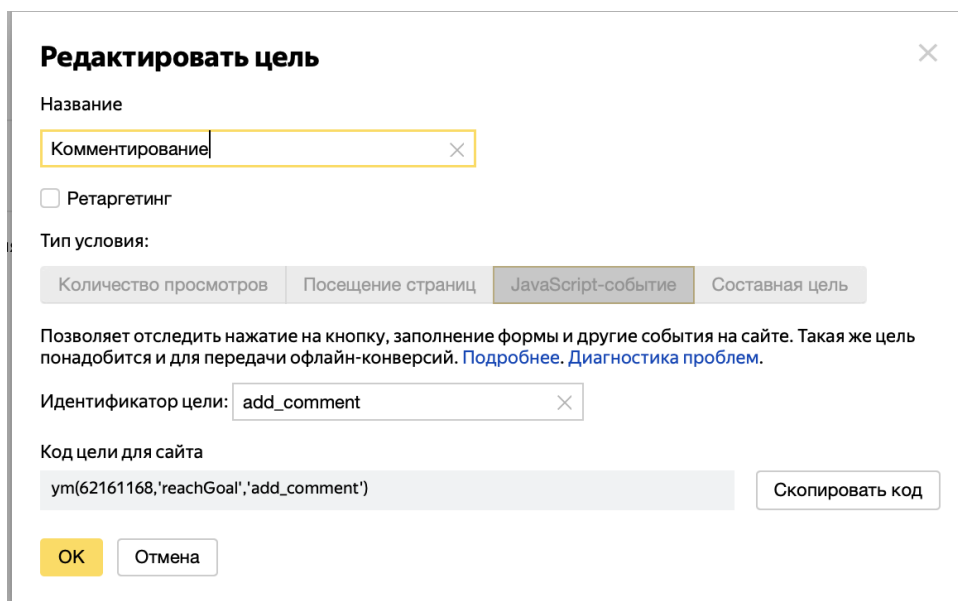
Укажите страницы, посещение которых для вас важно — например, страницы с контактами или с формой обратной связи. [Подробнее](#).

Условие url: совпадает [v] https://pure-badlands-57732.herokuapp.com/news-board [X] (+)

Рисунок 16 - Воронка для перехода на главную страницу

Комментирование новостей:

Цель создания - определить, является ли комментирование новостей актуальным для данной системы.



Редактировать цель [X]

Название
Комментирование [X]

☐ Ретаргетинг

Тип условия:

Количество просмотров | Посещение страниц | **JavaScript-событие** | Составная цель

Позволяет отследить нажатие на кнопку, заполнение формы и другие события на сайте. Такая же цель понадобится и для передачи офлайн-конверсий. [Подробнее](#). [Диагностика проблем](#).

Идентификатор цели: add_comment [X]

Код цели для сайта
ym(62161168,'reachGoal','add_comment')

[Скопировать код]

OK [Отмена]

Рисунок 17 - Воронка для добавления новости для учителя

3. РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

3.1. Обоснование выбора технологий

Рассмотрим выбранные для разработки технологии.

1. На сервере использовался язык Java, имеющий ряд преимуществ:
 - Язык Java является платформонезависимым, так как он компилируется в независимый от платформы байт-код, этот байт код интерпретируется в Java Virtual Machine (JVM);
 - Язык Java является архитектурно-нейтральным так как компилятор генерирует архитектурно-нейтральные объекты формата файла, что делает скомпилированный код исполняемым на многих процессорах, с наличием системы Java Runtime;
 - Сообщество разработчиков Java очень велико, около 45% респондентов опроса StackOverflow 2018 используют Java, что позволяет быстро искать решения каких-либо задач;
 - Позволяет не задумываться об управлении памятью - эту проблему решает автоматический сборщик мусора.
2. Spring Boot - фреймворк, который позволяет легко создавать приложения Spring, которые можно запускать, начиная работу с минимальным набором конфигураций. Spring Boot позволяет подключать стартеры, которые содержат множество зависимостей, необходимые для быстрого запуска и работы проекта с согласованным, поддерживаемым набором управляемых зависимостей. Кроме того, Spring Boot имеет приемлемую техническую документацию;
3. Spring Security - это Java фреймворк, предоставляющий механизмы построения систем аутентификации и авторизации;
4. Hibernate - ORM библиотека для языка программирования Java, предназначенная. Hibernate легко интегрируется со Spring Framework, устраняет множество повторяющегося кода, (по сравнению с использованием JDBC API), а

также предоставляет собственный мощный язык, объектно-ориентированный язык запросов HQL (Hibernate Query Language);

5. PostgreSQL - СУБД, которая использует реляционную модель для своих баз данных и поддерживает стандартный язык SQL. Рассмотрев аналоги, мы выяснили, что SQLite не подходит для многопользовательских приложений, а Oracle обладает неоправданно большой для нашего приложения функциональностью. Таким образом, решено было использовать PostgreSQL, как бесплатную и подходящую нашим потребностям СУБД.

6. Angular - мощный фреймворк для разработки одностраничных веб-приложений. Angular изначально использует TypeScript, который дает возможность применить к коду JavaScript систему типов, что облегчает разработку, а также позволяет применять принципы ООП. Кроме того, Angular позволяет использовать двустороннюю привязку данных, которая связывает хранилище данных и пользовательский интерфейс. Это приводит к обновлению данных хранилища при обновлении данных интерфейса и наоборот, что позволяет избавиться от большого количества кода.

3.2. Обоснование архитектуры

Серверная часть приложения написана с использованием REST архитектуры. REST - это архитектурный стиль для связи, основанный на строгом использовании типов запросов HTTP. Данная архитектура была выбрана по причинам:

- независимости клиентского и серверного приложения;
- присутствия в ответе сервера всех необходимых метаданных, например, информацию об ошибках.

Как альтернативу REST, можно рассмотреть SOAP, данный архитектурный подход не использовался нами так как:

- SOAP более тяжеловесный подход, нецелесообразный для нашего приложения;

- SOAP работает с сообщениями формата XML, мы используем JSON для передачи данных между клиентом и сервером.

Серверная часть приложения разработана согласно трехуровневой архитектуре (рисунок 18) и разбита на следующие слои:

- Data Access Layer - слой доступа к данным, содержит репозитории, взаимодействующие с БД;

- Service Layer - сервис-слой, содержит методы обработки данных для передачи на другие уровни;

- Presentation Layer - слой представления, содержит REST контроллеры.

Данный подход был выбран по ряду причин:

- Разделение логики работы приложения;
- Возможность параллельной разработки;
- Облегчается поддержка и тестирование кода;
- Соответствие выбранным фреймворкам.

Следствием использования такого архитектурного подхода является двунаправленная связь между “соседними” слоями. И также связь между слоями всегда проходит через слой сервисов.

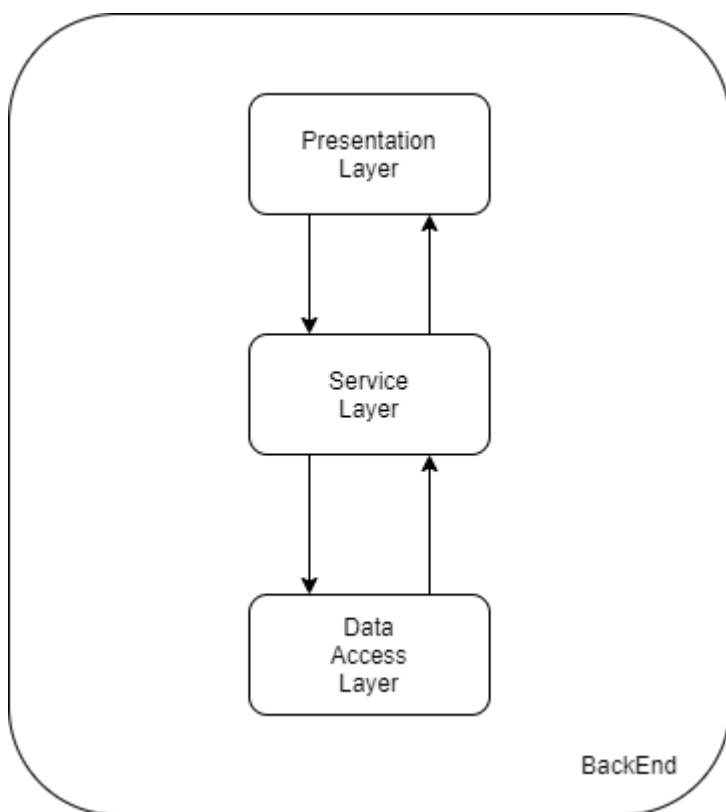


Рис. 18 - Слои серверной части приложения

В качестве аналога рассмотрим архитектурный шаблон MVC (Model- View Controller), в нем также происходит разделение функций модели данных, представления данных и логического программного потока. Но главным аргументом против, является топология данного паттерна. В MVC связи между слоями однонаправленные, представление отправляет обновления контроллеру, контроллер обновляет модель, а представление обновляется непосредственно из модели. Данный поток взаимодействия слоев не соответствует фреймворкам выбранным для разработки. Также среди аналогов можно привести еще паттерн MVVM (Model-View-View-Model), но он имеет еще более витиеватую, в сравнении с MVC архитектуру, что является неоправданным усложнением для разрабатываемого приложения.

3.3. Реализация серверной части

Рассмотрим реализацию серверной части приложения. Выделим основные классы различных слоев и рассмотрим их методы. Начнем со слоя доступа к данным, в нем расположены репозитории для взаимодействия с БД, рассмотрим подробно репозиторий для работы с данными о комментариях (рисунок 18).

```
@Repository
public interface CommentRepository extends JpaRepository<Comment, Long> {

    @Query("select c from Comment c where c.post.id =:id")
    Page<Comment> getCommentsByPostId(@Param("id") Long id, Pageable pageable);

    void deleteById(Long id);
}
```

Рис. 18 - Репозиторий для работы с комментариями

Репозиторий помечен соответствующей аннотацией `@Repository`, таким образом мы маркируем для Spring, что интерфейс выполняет роль хранилища. Данный интерфейс наследуется от `JpaRepository`, который предоставляет функциональность, связанную с сортировкой и пагинацией. Для получения комментариев определенного поста, мы передаем методу `id` поста, который будет подставлен в запрос (написанный на языке HQL), находящейся в аннотации `@Query`, а также объект типа `Pageable`, содержащий информацию о номере страницы и количестве комментариев на одной странице. Также определен метод для удаления комментария по его `id`, так как он является стандартным мы можем не писать запрос, Spring сформирует запрос по имени метода. Также важным моментом является то, что Spring автоматически создает имплементации описанных интерфейсов.

Далее перейдем к сервисному слою. Рассмотрим сервис для работы с пользователями системы (рисунок 19)

```

@Service
public class UserServiceImpl implements UserService {

    @Autowired
    private UserRepository userRepository;

    @Override
    public Boolean existsByUsername(String username) {
        return userRepository.existsByUsername(username);
    }

    @Override
    public void save(User user) { userRepository.save(user); }

    @Override
    public User findByUsername(String username) {
        return userRepository.findByUsername(username).orElse( other: null);
    }
}

```

Рис. 19 - Класс сервиса для работы с пользователями

Выше мы видим класс `ServiceImpl`, имплементирующий интерфейс `UserService`. Класс помечен аннотацией `@Service`, который указывает для Spring, что в нем содержится бизнес-логика приложения. Для получения данных из БД, мы внедряем репозиторий для получения данных пользователей. Далее описываем методы с бизнес логикой приложения.

Далее рассмотрим слой представления, содержащий REST контроллеры.

```

@CrossOrigin(origins = "*", maxAge = 3600)
@RestController
@RequestMapping("/api")
@Api(value = "schoolboard", description = "Operations with comments")
public class CommentController {

    @Autowired
    private CommentService commentService;

    @Autowired
    private UserService userService;

    |
    @ApiOperation(value = "Add new comment to post with post_id")
    @PostMapping("/posts/{post_id}/comments")
    @PreAuthorize("hasAnyRole('STUDENT', 'TEACHER')")
    public void addComment(@PathVariable("post_id") String id, @RequestBody CommentRequest comment){
        String username = SecurityContextHolder.getContext().getAuthentication().getName();
        commentService.saveOrUpdate(comment, Long.parseLong(id), username);
    }
}

```

Рис. 20 - Класс с контроллерами для комментариев

Рассмотрим контроллер, представленный на рисунке 20. Аннотация `@CrossOrigin` служит для включения возможности обработки перекрестных запросов. `@RestController` указывает Spring, что диспетчер запросов должен искать необходимые методы в этом классе. Аннотация `@RequestMapping` предназначена для того, чтобы задать методам вашего контроллера адреса, по которым они будут доступны на клиенте. Мы используем данную аннотацию, как на весь класс, так и на каждый метод. Применение аннотации ко всему классу задает "каталог", в котором будут размещаться методы контроллера. Применение аннотации к каждому методу добавляет к исходному пути адреса контроллеров. Аннотации `@Api` и `@ApiOperation`, содержат описания, необходимые для swagger. Также для определения типа запроса, обрабатываемого тем или иным контроллером, вместо `RequestMapping`, мы можем написать `GetMapping` или `PostMapping`, определив тип запроса, POST или GET. Аннотация `PreAuthorize` определяет набор ролей пользователя, которому разрешено получать данные из контроллера, помеченного такой аннотацией. Сам метод на рисунке 20 реализует функционал добавления комментария. Рассмотрим еще несколько контроллеров.

Контроллер для получения комментариев поста с определенным id:

```
@ApiOperation(value = "Find all comments below post with post_id")
@GetMapping("/posts/{post_id}/comments")
@PreAuthorize("hasAnyRole('STUDENT','TEACHER')")
public Page<CommentResponse> getAllComments(@PathVariable("post_id") String post_id, @RequestParam(defaultValue = "0") int page, @RequestParam("size") int size){
```

Контроллер для удаления комментария определенного поста

```
@ApiOperation(value = "Delete comment with comment_id")
@DeleteMapping("/posts/{post_id}/comments/{comment_id}")
@PreAuthorize("hasAnyRole('STUDENT','TEACHER')")
public ResponseEntity<> deleteComment(@PathVariable("post_id") String id, @PathVariable("comment_id") String comment_id){
```

Контроллер для получения всех постов

```
@ApiOperation(value = "Find all posts")
@GetMapping("")
@PreAuthorize("hasAnyRole('STUDENT','TEACHER')")
public Page<PostResponse> getAllPosts(@RequestParam(defaultValue = "0") int page, @RequestParam("size") int size){
```

Контроллер для получения конкретного поста по его id


```

@ApiOperation(value = "Find post with id")
@GetMapping("/{id}")
@PreAuthorize("hasRole('STUDENT') or hasRole('TEACHER')")
public ResponseEntity<?> getPostById(@PathVariable("id") String id){

```

Контроллер для добавления поста

```

@ApiOperation(value="Add new post")
@PostMapping("")
@PreAuthorize("hasRole('TEACHER')")
public void addPost(@RequestBody PostRequest post){

```

Контроллер для редактирования поста

```

@ApiOperation(value="Edit post with post_id")
@PutMapping("/{post_id}")
@PreAuthorize("hasRole('TEACHER')")
public void editPost(@PathVariable("post_id") Long post_id, @RequestBody PostRequest post){

```

Контроллер для удаления поста

```

@ApiOperation(value="Delete post with post_id")
@DeleteMapping("/{post_id}")
@PreAuthorize("hasRole('TEACHER')")
public ResponseEntity<?> deletePost(@PathVariable("post_id") String id){

```

Контроллер для получения списка всех пользователей

```

@ApiOperation(value = "View a list of all users")
@GetMapping("")
public List<UserResponse> getUsers() { return userService.findAll(); }

```

Контроллер для получения пользователя по id

```

@ApiOperation(value="Search user by his ID")
@GetMapping("/{id}")
public ResponseEntity<?> getUser(@PathVariable("id") Long id){

```

Контроллер для регистрации нового пользователя в системе

```

@ApiOperation(value = "Register new user in system")
@PostMapping("")
public ResponseEntity<?> registerUser(@Valid @RequestBody SignupRequest signUpRequest) {

```

Контроллер для обновления пользователя

```

@ApiOperation(value = "Update information about user with id")
@PutMapping("/{id}")
public ResponseEntity<?> updateUser(@Valid @RequestBody SignupRequest signUpRequest, @PathVariable("id") String id) {

```

Контроллер для удаления пользователя

```

@ApiOperation(value = "Delete user with id")
@DeleteMapping("/{id}")
public ResponseEntity<?> deleteUser(@PathVariable("id") String id) {

```

Контроллер для осуществления процесса входа пользователя в систему

```
@ApiOperation(value = "Authenticate user, that already registered")
@PostMapping("/signin")
public ResponseEntity<?> authenticateUser(@Valid @RequestBody LoginRequest loginRequest) {
```

3.4. Интерфейс

Рассмотрим интерфейс разработанного приложения. На рисунке 21 изображена страница входа. На ней пользователю предлагается заполнить поля именем пользователя и паролем для осуществления входа.

Школьная новостная доска

Вход

Войти

Сайт разработан Екатериной, Ольгой и Ильей.

Рисунок 21 - Страница входа

Если вход был произведен успешно, то слева в шапке сайта авторизованный пользователь увидит свое имя на сайте и кнопку “Выйти”, нажав на которую пользователь произведет выход и будет перенаправлен на страницу входа.

После успешного входа на сайт авторизованный пользователь имеющий в системе роль ученика будет направлен на страницу новостей школьной доски, ее вид показан на рисунке 22. Здесь ученик сможет увидеть все новости, опубликованные на сайте. Для каждой новости он увидит ее заголовок, который

выделен жирным шрифтом для привлечения внимания, автора новости, текст ее содержания и дату публикации. Также рядом с каждой новостью ученик сможет увидеть кнопку “Открыть”, нажав на которую он перейдет на страницу подробного просмотра новости.

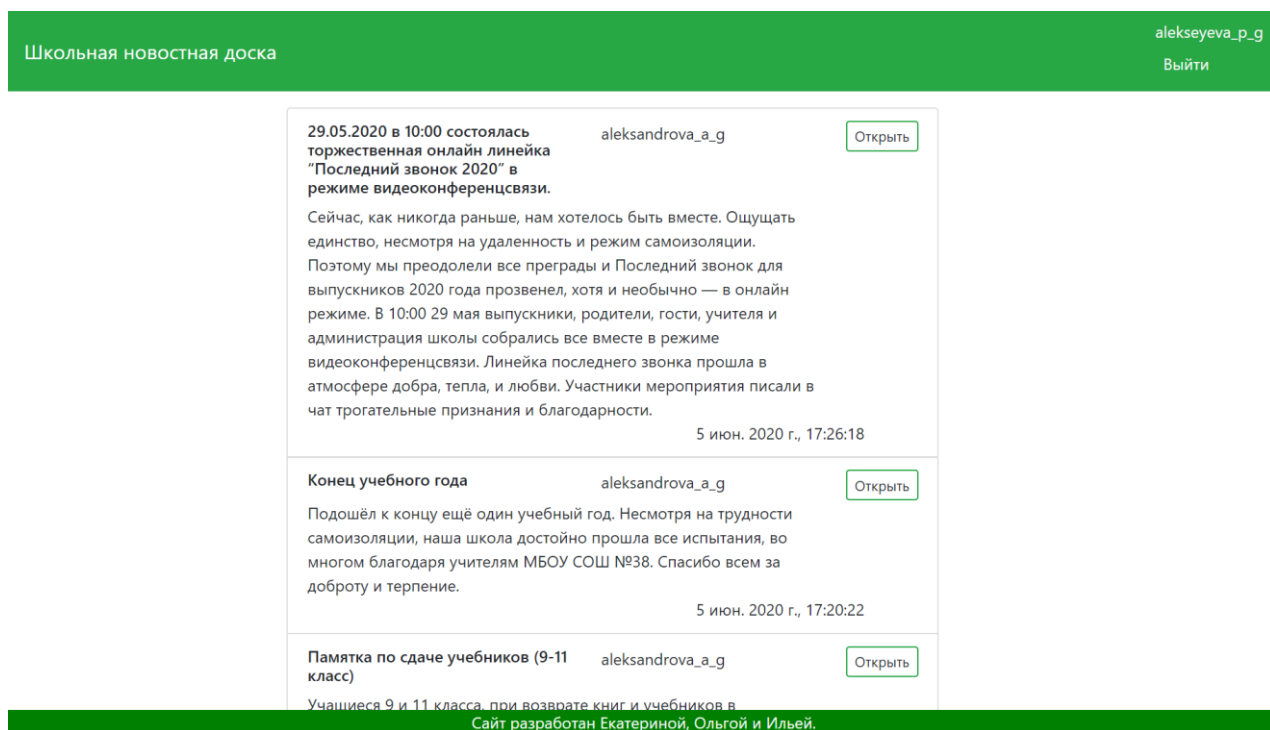


Рисунок 22 - Страница новостей школьной доски (ученик)

После нажатия на описанную ранее кнопку “Открыть” пользователь будет перенаправлен на страницу подробного просмотра новости, которая изображена на рисунке 23. На этой странице будет размещена вся информация об этой новости: ее заголовок, автора новости, текст ее содержания и дату публикации, а также комментарии написанные пользователями сайта к данной новости. Каждый комментарий содержит имя его автора в системе и само содержание комментария. Для комментариев к новости здесь реализована пагинация. Ученик может удалить свои комментарии к этой новости, нажав на кнопку “Удалить”, которая будет находиться рядом с комментариями, автором которых является текущий пользователь (ученик). Кроме того, внизу новости находится кнопка

“+”, нажав на которую текущий пользователь перейдет к созданию нового комментария к данной новости.

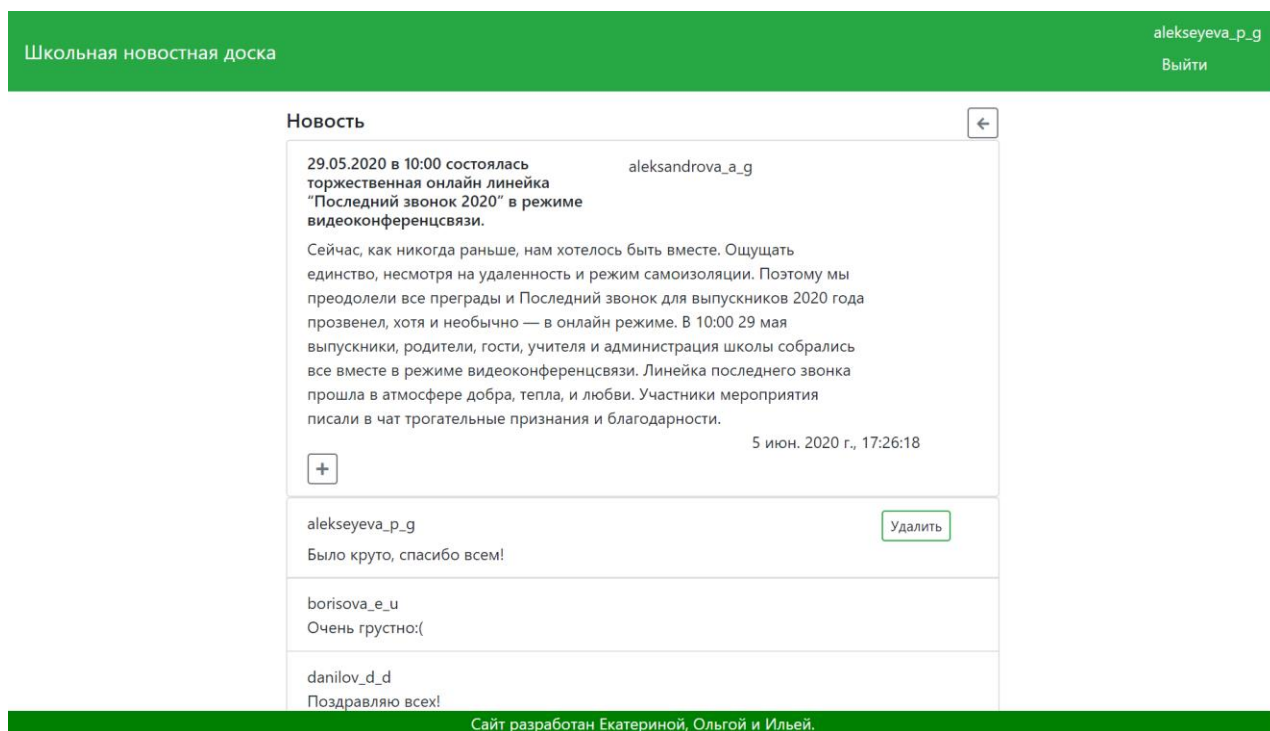


Рисунок 23 - Страница подробного просмотра новости (ученик)

На рисунке 24 показана страница создания комментария, где пользователь (ученик) может ввести текст своего комментария в соответствующее поле и нажать кнопку “Опубликовать”, после чего данный комментарий будет добавлен к новости. Если же пользователь (ученик) передумал создавать комментарий к новости, то он может нажать на кнопку “←”, которая возвращает его обратно к странице подробного просмотра новости.

Комментарий



Создать комментарий

Опубликовать

Сайт разработан Екатериной, Ольгой и Ильей.

Рисунок 24 - Страница создания комментария к новости (ученик)

Далее рассмотрим интерфейс страниц доступных авторизованному пользователю с ролью учитель в системе. На рисунке 25 изображена страница новостей школьной доски. Она выглядит также как и для ученика, но у учителя справа от новостей есть кнопка “+”, нажав на которую учитель сможет перейти к созданию новой новости.

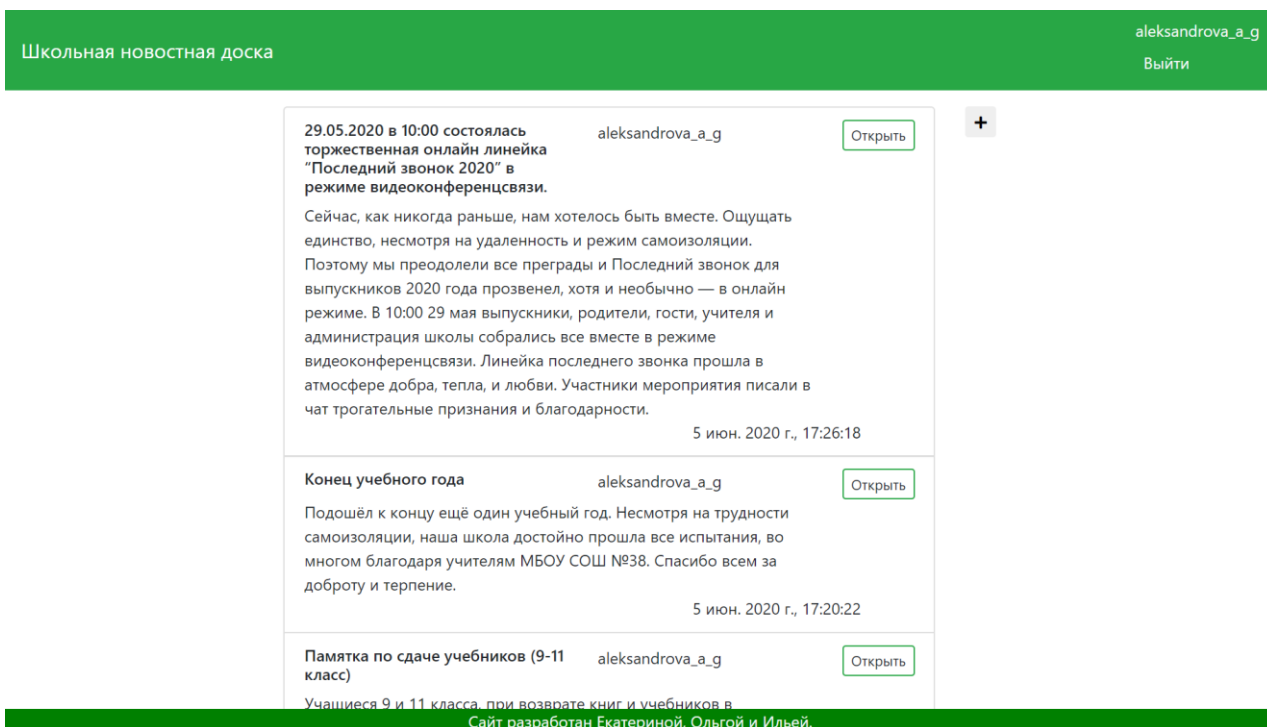


Рисунок 25 - Страница новостей школьной доски (учитель)

На рисунке 26 изображена страница создания новости на школьной новостной доске. Здесь пользователю (учителю) для публикации новости необходимо заполнить поля заголовков новости и текст новости, после чего нажать кнопку “Опубликовать”. В случае если пользователь (учитель) передумал публиковать новость вверху страницы предусмотрена кнопка “←”, которая переместит на страницу новостей школьной доски.

Новость


Заголовок новости

Текст новости

Опубликовать

Рисунок 26 - Страница создания новости

Также учитель, как и ученик может перейти на страницу подробного просмотра новости, где он будет иметь некоторые дополнительные функции. Учитель может перейти к редактированию новости нажав на кнопку “Редактировать”. Также учитель будет видеть кнопку “Удалить”, если он является автором этой новости. Кроме того, учитель может удалять любые комментарии пользователей. Для этого рядом с текстом каждого комментария находится кнопка “Удалить”. Вид данной страницы приведен на рисунке 27.

Новость 


Конец учебного года

aleksandrova_a_g

Редактировать

Удалить

Подошёл к концу ещё один учебный год. Несмотря на трудности самоизоляции, наша школа достойно прошла все испытания, во многом благодаря учителям МБОУ СОШ №38. Спасибо всем за доброту и терпение.



5 июн. 2020 г., 17:20:22

borisova_e_u

Удалить

Спасибо, дорогие учителя!

1

Рисунок 27 - Страница подробного просмотра новости (учитель)

Нажав на кнопку редактировать на странице подробного просмотра новости пользователь (учитель) будет перенаправлен на страницу редактирования новости, она представлена на рисунке 28. Здесь пользователю будут предложены поля, заполненные текущим заголовком новости и ее текстом, которые он может изменить и нажать на кнопку “Редактировать” после чего новость будет отредактирована, а пользователь перенаправлен на страницу просмотра новостей школьной доски.

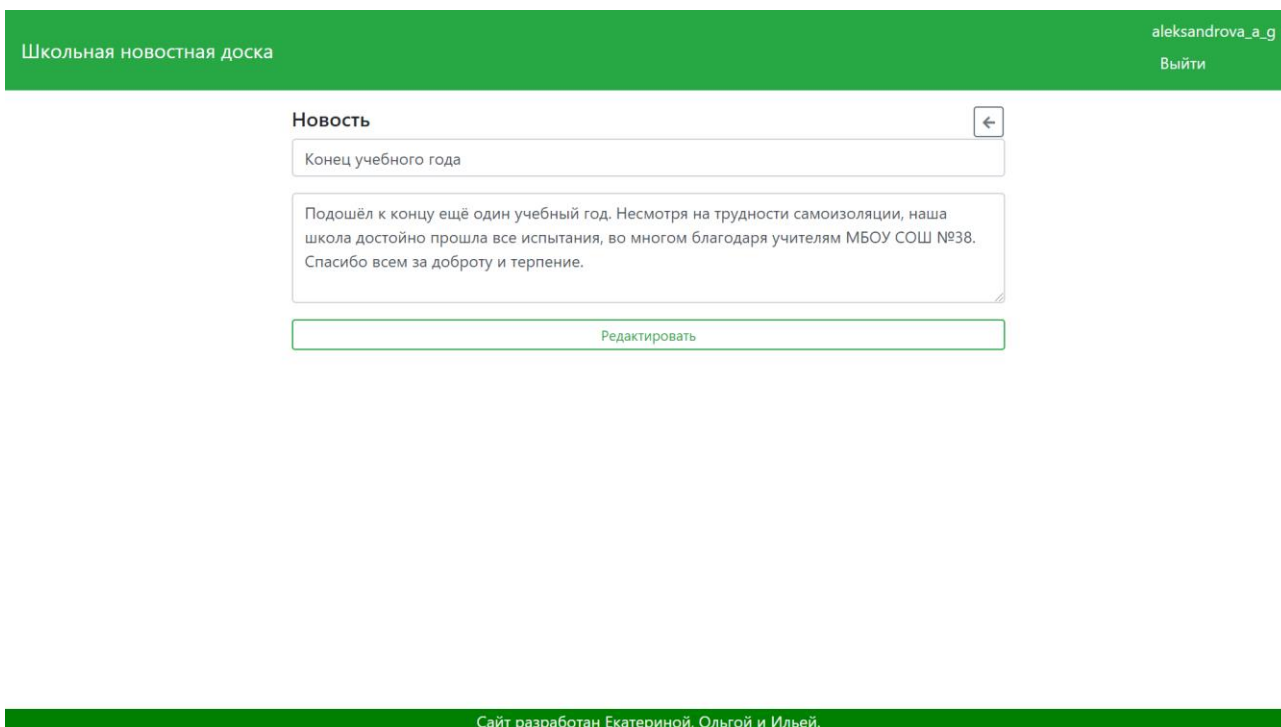


Рисунок 28 - Страница редактирования новости

При попытке удаления новости пользователь (учитель) будет видеть диалоговое окно, в котором требуется подтвердить удаление (рисунок 29).

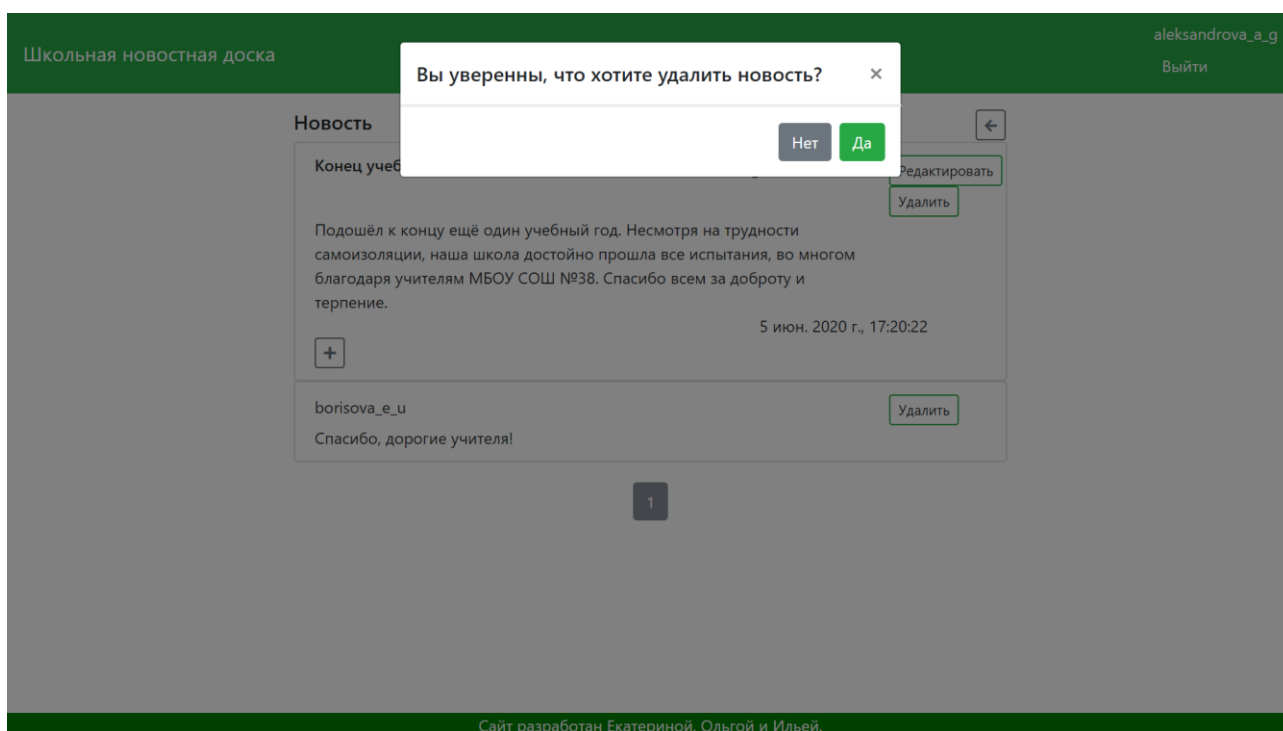


Рисунок 29 - Диалоговое окно, возникающее при попытке удаления новости

Также учитель, как и ученик может оставлять комментарии к новостям, заполнив форму для комментария и нажав кнопку “Опубликовать” (рисунок 30).

The screenshot displays a web interface for a school news board. At the top, a green header bar contains the text "Школьная новостная доска" on the left and the username "aleksandrova_a_g" with a "Выйти" (Logout) link on the right. Below the header, there is a section titled "Комментарий" (Comment) with a back arrow icon in the top right corner. Inside this section is a large text input field with the placeholder text "Создать комментарий" (Create comment). Below the input field is a green button labeled "Опубликовать" (Publish). At the bottom of the page, a green footer bar contains the text "Сайт разработан Екатериной, Ольгой и Ильей." (The site was developed by Ekaterina, Olga, and Ilya).

Рисунок 30 - Страница создания комментария к новости (учитель)

Рассмотрим страницы, доступные администратору. На рисунке 31 изображена страница управления пользователями сайта, доступная администратору. Здесь администратор может увидеть список всех пользователей (учеников и учителей), зарегистрированных на сайте. Для каждого пользователя указана его роль (ученик/учитель) и имя пользователя на сайте. Администратор может удалить какого-либо пользователя, нажав на кнопку “Удалить”, расположенную рядом с информацией о пользователе или перейти к редактированию данных о нем, нажав на кнопку “Редактировать”. Также администратор может перейти к созданию нового пользователя, нажав на кнопку “Новый пользователь”, расположенную справа от списка пользователей.

| Новый пользователь | | | |
|--------------------|------------------|--------|---|
| Учитель/ Ученик | Имя Пользователя | Пароль | |
| Ученик | alekseyeva_p_g | ***** | <div>Удалить</div> <div>Редактировать</div> |
| Ученик | alekseyev_f_a | ***** | <div>Удалить</div> <div>Редактировать</div> |
| Ученик | borisova_e_u | ***** | <div>Удалить</div> <div>Редактировать</div> |
| Ученик | vasil'yev_i_u | ***** | <div>Удалить</div> <div>Редактировать</div> |
| Ученик | danilov_d_d | ***** | <div>Удалить</div> <div>Редактировать</div> |
| Ученик | zhidova_e_g | ***** | <div>Удалить</div> <div>Редактировать</div> |

Рисунок 31 - Страница управления пользователями сайта

Нажав на кнопку “Новый пользователь”, администратор перейдет на страницу создания пользователя, изображенную на рисунке 32. Здесь администратору необходимо указать роль ученика в системе (ученик/учитель) имя пользователя в системе, а также его пароль. Пароль можно ввести вручную или сгенерировать случайный, нажав на кнопку создать пароль.

Новый пользователь

☒ Ученик ☐ Учитель

Имя пользователя

Пароль

Создать пароль

Рисунок 32 - Страница создания пользователя

Если администратор нажмет на кнопку “Редактировать” (рисунок 31), то он перейдет на страницу редактирования пользователя, которая изображена на рисунке 33. Здесь администратор может изменять имя пользователя и задать новый пароль в системе. Для сохранения изменений необходимо нажать на кнопку “Редактировать”.

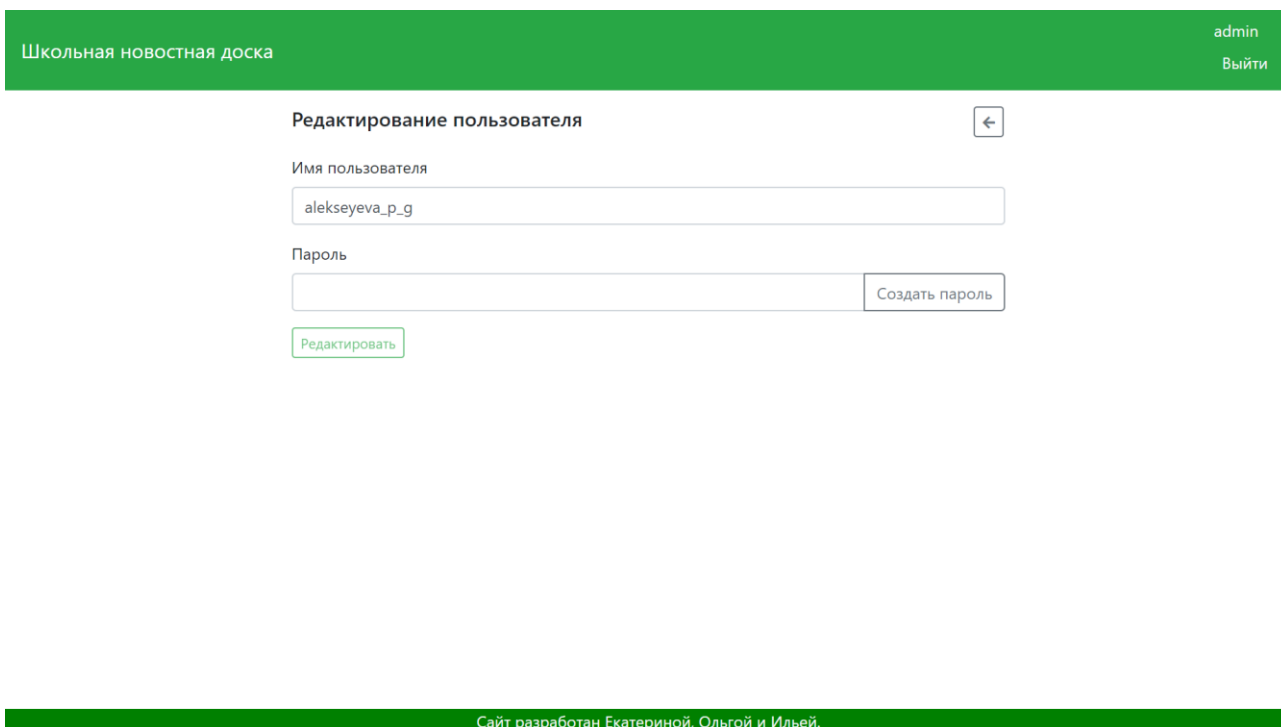


Рисунок 33 - Страницу редактирования пользователя

Если администратор предпримет попытку удалить какого-либо пользователя, нажав на кнопку “Удалить” (рисунок 31), то он увидит диалоговое окно, в котором требуется подтвердить удаление (рисунок 34).

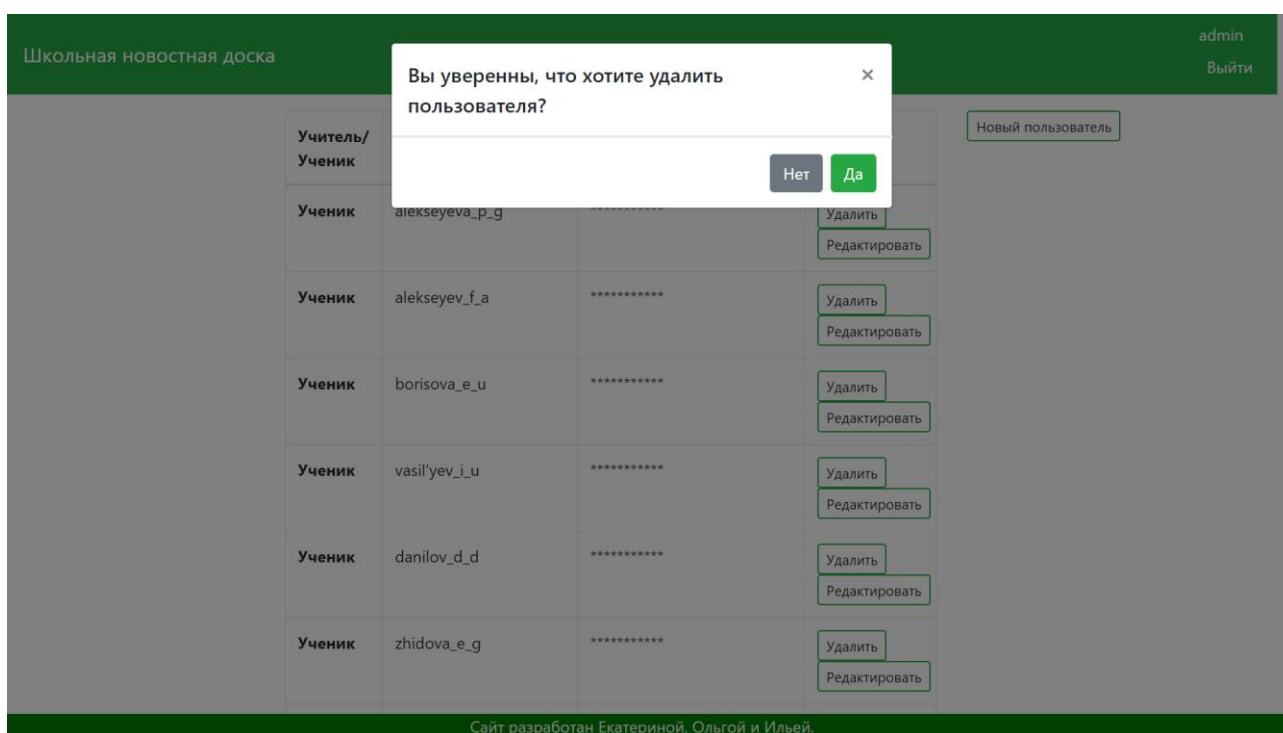


Рисунок 34 - Диалоговое окно, возникающее при попытке удаления пользователя

4. ТЕСТИРОВАНИЕ

В рамках разработки приложения были проведены виды тестирования, заявленные в техническом задании, в соответствии с поставленными целями тестирования, которые были описаны в плане тестирования:

1. Smoke testing – поверхностное тестирование системы на предмет работоспособности. В рамках этого тестирования были проверены следующие возможности:

- запуск приложения;
- вход ученика в систему;
- вход учителя в систему;
- вход администратора в систему;
- отображение всех новостей (ученик);
- отображение всех новостей (учитель);
- подробный просмотр новости (ученик);
- подробный просмотр новости (учитель);
- удаление комментария (учитель или ученик);
- публикация новости;
- редактирование новости;
- удаление новости (учитель);
- отображение страницы администрирования сайта;
- удаление пользователя из системы;
- выход из системы.

2. Sanity testing – тестирование работы конкретных функций системы, подтверждающих работу системы согласно требованиям в Техническом Задании. Были протестированы функциональные возможности системы:

- комментирование новости;
- добавление нового пользователя в систему;
- редактирование данных пользователя.

3. Negative testing – тестирование поведения системы, при попытке ввода некорректных данных. Было изучено поведение системы при выполнении следующих функций:

- вход в систему с некорректными данными,
- создание пользователя с существующим именем,
- добавление новости с незаполненным заголовком и/или текстом.

4. Usability testing – тестирование того, насколько легко конечный пользователь сможет освоить систему. Для проведения данного тестирования были выбраны 3 человека не знакомые с системой, которым предлагалось выполнить основные функции системы.

С результатами проведения тестирований можно ознакомиться в приложении 4.

В рамках проведения составленных тестовых кейсов для системы, можно утверждать, что поведение системы является корректным для всех описанных тестовых сценариев smoke, sanity и negative тестирований. При проведении usability тестирования у одного пользователя возникли затруднения при прохождении тестового кейса, которые не являются критическими.

ЗАКЛЮЧЕНИЕ

По итогу выполненной работы нами было разработано веб-приложение, соответствующее заявленным требованиям. Благодаря нашему продукту, был упрощен процесс распространения информации для учителей и учащихся.

В ходе выполнения работы были проанализированы существующие решения в области школьных информационных ресурсов. В результате анализа был сделан вывод о том, что существующие приложения обладают перегруженной функциональностью и отсутствует возможность добавления комментариев.

В ходе разработки курсового проекта в команде было реализовано клиент-серверное приложение для публикации и обсуждения школьных новостей, которое удовлетворяет следующим требованиям:

- имеет простой и понятный интерфейс;
- позволяет учителю создавать, редактировать и удалять новости, а также добавлять и удалять комментарии;
- позволяет ученику просматривать и комментировать новости.

Кроме того, мы овладели навыками работы в команде при разработке веб-приложения и распределения ролей и задач для достижения поставленных целей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Craig Walls. Spring in Action (5th Edition) / Craig Walls, 2018.
2. Spring Framework Documentation [сайт] – URL: <https://docs.spring.io/spring/docs/current/spring-framework-reference/>
3. Информационный ресурс Baeldung [сайт] – URL: <https://www.baeldung.com/swagger-2-documentation-for-spring-rest-api>
4. Angular Documentation [сайт] – URL: <https://angular.io/docs>
5. Фаулер М. UML. Основы, 3 е издание. / Фаулер М. – Пер. с англ. – СПб: Символ Плюс, 2005. – 192 с.

ПРИЛОЖЕНИЕ 1

Создание нового пользователя на сайте “Дневник.ру”

Администрирование - Люди

Создание новой персоны

1. Способ добавления персоны 2. **Добавление персоны** 3. Проверка данных

Роль и статус

Роль: ☐ Сотрудник ☒ Ученик

Статус:

Подпись:

Дата прибытия в школу:

Дата прибытия в класс:

По призыву №:

Дата призыва:

Тип призыва:

Личные данные

Фамилия:

Имя:

Отчество:

Дата рождения:

Пол: ☐ Мужской ☐ Женский

Место рождения:

Гражданство:

СНИЛС:

Виза:

Свидетельство о рождении

Серия:

Номер:

Кем выдан:

Дата выдачи:

Место выдачи:

Номер выданной визы:

Документ, удостоверяющий личность

Тип документа:

Серия:

Номер:

Кем выдан:

Место выдачи:

Дата выдачи:

Контактные данные

Адрес постоянной регистрации:

Адрес временной регистрации:

Дата окончания временной регистрации:

Односторонний адрес проживания: ☐ Совпадает с адресом постоянной регистрации

Подтверждение проживания на территории, включенной в ОО: ☐ Да ☐ Нет

Адрес:

Рабочий телефон:

Мобильный телефон:

Домашний телефон:

ПРИЛОЖЕНИЕ 2

1. Процесс добавления новости на сайте “Дневник.ру”.

Ссылка на BPMN диаграмму:

<https://github.com/Zeleboba99/school-board/blob/master/Документация/bpmn/Добавление%20новости.pdf>

2. Процесс добавления нового пользователя на сайте “Дневник.ру”.

Ссылка на BPMN диаграмму:

<https://github.com/Zeleboba99/school-board/blob/master/Документация/bpmn/Добавление%20пользователя.png>

ПРИЛОЖЕНИЕ 3

Устав муниципального бюджетного общеобразовательного учреждения средняя общеобразовательная школа с углубленным изучением отдельных предметов №38 имени Е. А. Болховитинова.

Ссылка на документ:

<https://disk.yandex.ru/i/5YETJQWhkLv69w>

ПРИЛОЖЕНИЕ 4.

Результаты проведенного тестирования.

Таблица 1 - Результаты проведения Smoke testing

| Идентификатор теста | Тип тестирования | Краткое описание | Статус теста | Приоритет |
|---------------------|------------------|--|--------------|-----------|
| T-1 | Smoke | Запуск приложения | Пройден | Высокий |
| T-2 | Smoke | Вход ученика в систему | Пройден | Высокий |
| T-3 | Smoke | Вход учителя в систему | Пройден | Высокий |
| T-4 | Smoke | Вход администратора в систему | Пройден | Высокий |
| T-5 | Smoke | Отображение всех новостей(ученик) | Пройден | Высокий |
| T-6 | Smoke | Отображение всех новостей(учитель) | Пройден | Высокий |
| T-7 | Smoke | Побробный просмотр новости (ученик) | Пройден | Высокий |
| T-8 | Smoke | Побробный просмотр новости (учитель) | Пройден | Высокий |
| T-9 | Sanity | Комментирование новости | Пройден | Высокий |
| T-10 | Smoke | Удаление комментария (учитель или ученик) | Пройден | Высокий |
| T-11 | Smoke | Публикация новости | Пройден | Высокий |
| T-12 | Smoke | Редактирование новости | Пройден | Средний |
| T-13 | Smoke | Удаление новости (учитель) | Пройден | Высокий |
| T-14 | Smoke | Отображение страницы администрирования сайта | Пройден | Высокий |
| T-17 | Smoke | Удаление пользователя из системы | Пройден | Высокий |
| T-18 | Smoke | Выход из системы | Пройден | Высокий |

Таблица 2 - Результаты проведения Sanity testing

| Идентификатор теста | Тип тестирования | Краткое описание | Статус теста | Приоритет |
|---------------------|------------------|--|--------------|-----------|
| T-15 | Sanity | Добавление нового пользователя в систему | Пройден | Высокий |
| T-16 | Sanity | Редактирование данных пользователя | Пройден | Высокий |

Таблица 3 - Результаты проведения Negative testing

| Идентификатор теста | Тип тестирования | Краткое описание | Статус теста | Приоритет |
|---------------------|------------------|---|--------------|-----------|
| T-19 | Negative | Вход в систему с некорректными данными | Пройден | Высокий |
| T-20 | Negative | Создание пользователя с существующим именем | Пройден | Средний |
| T-21 | Negative | Добавление новости с незаполненным заголовком и/или текстом | Пройден | Высокий |

Таблица 4 - Результаты проведения Usability testing

| Функция | Пользователь 1 | Пользователь 2 | Пользователь 3 |
|--|----------------------|----------------|----------------|
| Авторизоваться | Пройден | Пройден | Пройден |
| Опубликовать новость на доске | Возникли затруднения | Пройден | Пройден |
| Отредактировать опубликованную новость | Пройден | Пройден | Пройден |
| Удалить опубликованную новость | Пройден | Пройден | Пройден |

| | | | |
|---|---------|---------|---------|
| Прокомментировать Опубликованную новость | Пройден | Пройден | Пройден |
| Удалить комментарий | Пройден | Пройден | Пройден |

ПРИЛОЖЕНИЕ 5

Распределение задач по участникам команды

Распределение задач для участника Мещерякова Ильи:

1. Создание диаграмм:
 - Диаграмма прецедентов;
 - Диаграмма классов;
 - Диаграммы последовательностей;
 - Диаграммы коммуникаций;
 - Диаграмма развертывания.
2. Написание серверной части приложения.
3. Написание плана тестирования.
4. Проведение тестирования;
5. Подключение “Swagger” для документирования работы сервера.
6. Оформление курсовой работы.
7. Написание ТЗ по ГОСТ 34.

Распределение задач для участника Елфимовой Екатерины:

1. Создание диаграмм:
 - Диаграмма объектов;
 - Диаграмма базы данных;
 - Диаграмма IDEF0.
2. Анализ предметной области в курсовой:
 - Анализ существующих решений.
3. Реализация клиентской части приложения.
4. Написание скрипта для создания и заполнения базы данных.
5. Написание тест-кейсов.
6. Развертывание приложения на сервере.
7. Подключение “Swagger” для документирования работы сервера.
8. Написание ТЗ по ГОСТ 34.

Распределение задач для участника Стребковой Ольги:

1. Создание диаграмм:
 - Диаграмма реализации;
 - Диаграммы BPMN;
 - Диаграмма состояний;
 - Диаграмма активностей
2. Анализ предметной области в курсовой работе:
 - Анализ существующих решений;
 - Описание диаграмм;

- Анализ процесса добавления новости и создания нового пользователя в приложении-аналоге;
 - Цели создания сайта.
3. Написание отчета о выполненной работе.
 4. Подключение Яндекс.Метрики.
 5. Оформление курсовой работы.
 6. Презентация.
 7. Написание ТЗ по ГОСТ 34.