

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

Факультет компьютерных наук

Кафедра информационных систем

Сервис для создания и управления персонализированными новостными лентами,
с возможностью фильтрации контента по интересам

«Мои Новости»

Курсовой проект

по дисциплине

Технологии программирования

09.03.02 Информационные системы и технологии

Встраиваемые вычислительные системы и интернет вещей

7 семестр 2024/2025 учебного года

Зав. кафедрой _____ к. т. н., доцент Д.Н. Борисов

Обучающийся _____ ст. 3 курса оч. отд. Д. Д. Рындин

Обучающийся _____ ст. 3 курса оч. отд. В.Д. Михайлов

Руководитель _____ В.С. Тарасов, ст. преподаватель ___. ___.20__

Воронеж 2024

Содержание

Введение.....	3
1 Постановка задачи.....	5
1.1 Цели создания системы	5
1.2 Требования к разрабатываемой системе.....	5
1.3 Задачи проекта.....	5
2 Анализ предметной области	6
2.1 Терминология	6
2.2 Обзор аналогов	7
2.2.1 Яндекс.Дзен	8
2.2.2 Telegram.....	9
2.3 Моделирование системы	10
2.3.1 Диаграммы прецедентов.....	10
2.3.2 Диаграмма последовательности	13
2.3.3 Диаграмма развертывания.....	14
2.3.4 Диаграмма состояний	14
2.3.5 Диаграмма активности.....	15
3 Реализация.....	18
3.1 Средства реализации.....	18
3.2 Характеристики системы.....	20
3.3 Реализация Backend-части.....	20
3.4 Реализация Frontend-части	22
Заключение	28
Список использованных источников	29

Введение

В современном мире приложения с новостными лентами занимают важное место, обеспечивая доступ к множеству источников информации и событий со всего мира. Эти приложения позволяют пользователям создавать персонализированные новостные ленты, отражающие их уникальные интересы и предпочтения.

Одной из ключевых особенностей таких приложений является возможность настройки контента под индивидуальные потребности каждого пользователя. Благодаря функции фильтрации по интересам, пользователи могут выбирать категории новостей, конкретные темы или ключевые слова, которые им наиболее интересны. Это позволяет получать информацию о том, что действительно важно для них, и игнорировать неактуальный или малозначимый контент.

Кроме того, приложения с новостными лентами обеспечивают удобный доступ к актуальным событиям в любое время и в любом месте. Благодаря мобильной оптимизации и возможности синхронизации между устройствами, пользователи могут быть в курсе последних новостей даже в движении, не теряя связи с миром вокруг себя.

Они также предлагают различные функции для улучшения пользовательского опыта, такие как создание персонализированных лент, сохранение интересных статей для последующего прочтения, уведомления о важных событиях и многое другое. Эти возможности делают использование приложений с новостными лентами не только информативным, но и удобным и приятным.

В данной курсовой работе рассматривается процесс разработки веб-приложения "Мои Новости", предназначенного для создания и управления персонализированными новостными лентами с возможностью фильтрации контента по интересам.

В рамках исследования будут описаны различные аспекты разработки такого приложения, начиная с анализа предметной области, определения его

концепции и основных требований. Затем будет изучено проектирование пользовательского интерфейса и пользовательского опыта с учетом современных тенденций и передовых практик в этой области. Также, внимание будет уделено выбору и интеграции необходимых технологий и API с целью обеспечения функциональности приложения, включая возможность настройки новостных лент, поиск информации и другие необходимые функции.

1 Постановка задачи

1.1 Цели создания системы

Целью данной работы является реализация сервиса, позволяющего просматривать новости с возможностью выбора тегов для пользователей данного сервиса для разных категорий пользователей, просмотр новостей по этим тегам, также с возможностями получения актуальной информации из журналов, написанных редакторами.

1.2 Требования к разрабатываемой системе

- использование протокола передачи данных HTTP;
- обеспечение защиты системы от SQL-инъекций, защиты конфиденциальных данных при помощи необходимых механизмов;
- создание возможности работы с аккаунтами (регистрация, авторизация, настройка);
- поддержка русского языка в приложении;
- приложение должно иметь архитектуру вида Клиент-Сервер.

1.3 Задачи проекта

- просмотр новостей или журналов;
- выбор тегов для настройки просмотра;
- редактирование информации аккаунта;
- просмотр новостей от издателя;
- создание, редактирование, удаление новостей или журналов в качестве редактора;
- создание, редактирование, удаление новостей или журналов в качестве администратора;
- возможность назначения редакторов администратором.

2 Анализ предметной области

2.1 Терминология

Таблица 1 - Глоссарий

Веб-приложение	Программное обеспечение, разновидность прикладного программного обеспечения, предназначенная для работы на смартфонах, планшетах и других мобильных (портативных, переносных, карманных) устройствах.
БД	Это организованная коллекция данных, хранящихся в централизованном месте и структурированные таким образом, чтобы обеспечивать эффективное добавление, доступ, управление и обновление информации.
Frontend	Презентационная часть информационной или программной системы, ее пользовательский интерфейс и связанные с ним компоненты
Backend	Логика работы сайта, внутренняя часть продукта, которая находится на сервере и скрыта от пользователя.
REST API (REST)	Стиль архитектуры программного обеспечения для построения масштабируемых веб-приложений.
Django	Высокоуровневый Python веб-фреймворк, который позволяет быстро создавать безопасные и поддерживаемые веб-сайты.

СУБД	Набор программ, которые управляют структурой БД и контролируют доступ к данным, хранящимся в БД.
JavaScript	Это язык программирования, предназначенный для front-end-разработки и использующийся для взаимодействия с пользователем.
CSS	Формальный язык описания внешнего вида веб-страницы, написанного с использованием языка разметки.
HTML	Стандартизированный язык разметки для просмотра веб-страниц в браузере.
Vue.js	JavaScript-фреймворк с открытым исходным кодом для создания пользовательских интерфейсов.
SQL-инъекция	Это один из видов атак, при которых в приложение внедряются SQL-запросы с целью получения несанкционированного доступа к базе данных или выполнения операций, неавторизованных для пользователя.

2.2 Обзор аналогов

При разработке сервиса для создания и управления персонализированными новостными лентами с функцией фильтрации контента по интересам "Мои Новости", важно уделить внимание актуальности проекта. Это подразумевает не только создание функционального продукта, но и обеспечение его конкурентоспособности на рынке в виду наличия уникальных качеств. Для этого необходимо тщательно изучить аналогичные сервисы, выявив их преимущества и недостатки, чтобы учесть успешные практики и избежать ошибок.

При анализе аналогов следует особенно обращать внимание на способы персонализации контента, эффективность алгоритмов фильтрации, а также удобство использования интерфейса. Отдельное внимание стоит уделить тому, какие дополнительные функции предлагают конкуренты для разработки уникальных возможностей.

2.2.1 Яндекс.Дзен

Яндекс.Дзен - это платформа для чтения и публикации контента, созданная компанией Яндекс. Она предоставляет пользователям персонализированные новостные ленты, собирая материалы со множества источников и адаптируя их под интересы каждого конкретного пользователя.

Яндекс.Дзен представляет собой медиа-платформу, где пользователи обладают возможностями читать статьи, просматривать фото, видео и другой контент, а также создавать и публиковать свои материалы. Сервис объединяет разнообразные источники информации, включая блоги, издательства, СМИ, и даже личные блоги пользователей для предоставления широкого спектра интересующих тем.

Яндекс.Дзен ориентирована на широкий круг пользователей, интересующихся получением информации на различные темы. Это могут быть люди всех возрастов и профессий.

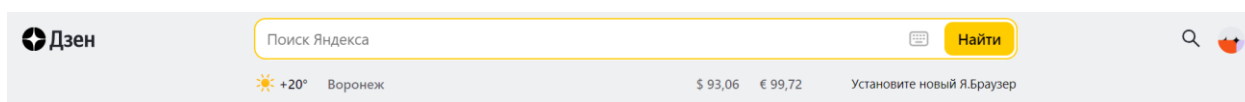


Рисунок 1 - Верхняя панель сервиса «Дзен»

Недостатки:

- из-за возможности публикации материалов пользователями, качество контента является неравномерным, ограничений по загрузке, кроме правил, нет;
- ограниченная настройка фильтрации контента;
- избыточное количество навязчивой рекламы, которую невозможно отключить в виду интеграции с другими сервисами;

2.2.2 Telegram

Telegram – это мессенджер с возможностью обмена сообщениями, файлами, медиафайлами, аудио и видео вызовами. Он известен своей высокой степенью шифрования и возможностью создания защищенных чатов.

Telegram представляет собой популярный мессенджер, который позволяет пользователям обмениваться текстовыми сообщениями, мультимедийными файлами, аудио и видео вызовами. Он доступен на различных платформах, включая мобильные устройства и компьютеры, и предлагает синхронизацию между ними. Основной особенностью Telegram является его высокая степень шифрования, обеспечивающая безопасность переписки, а также возможность создания защищенных чатов с самоуничтожающимися сообщениями.

Сервис доступен на различных платформах, включая iOS, Android, Windows, macOS и Linux, что повышает удобство использования на различных устройствах. Также существует веб-версия, которая позволяет пользователям обмениваться сообщениями через браузер.

Telegram привлекает широкий круг пользователей. Он популярен среди тех, кто ценит конфиденциальность и безопасность в переписке, а также среди людей, которым важна возможность обмениваться файлами и медиафайлами в удобном интерфейсе. приложения.

Недостатки:

- ограничения в функциональности, связанной с тегами и таргетированными новостями (невозможно получить новости, если их нет в каналах или их не переслали в каналы, на которые оформлена подписка);
- отсутствие возможности писать большие статьи;
- наличие дополнительной функциональности, излишней для пользователей и невозможной для отключения.

2.3 Моделирование системы

2.3.1 Диаграммы прецедентов

Диаграмма прецедентов - это визуальное представление функциональных возможностей системы и взаимодействия между пользователями (актерами) и системой. В ней выделены основные действия, которые пользователи могут выполнить в системе, и показывает, как эти действия связаны между собой. Основной целью диаграммы прецедентов является предоставление общее представление о функциональности системы и ее использовании.

Каждый прецедент представляет собой конкретное действие или операцию, которую пользователь может выполнить. Он описывается названием и может быть дополнен подробным описанием, объясняющим его функциональность.

На следующем рисунке представлена диаграмма прецедентов для сценариев для пользователя, описывающая основную функциональность сервиса:

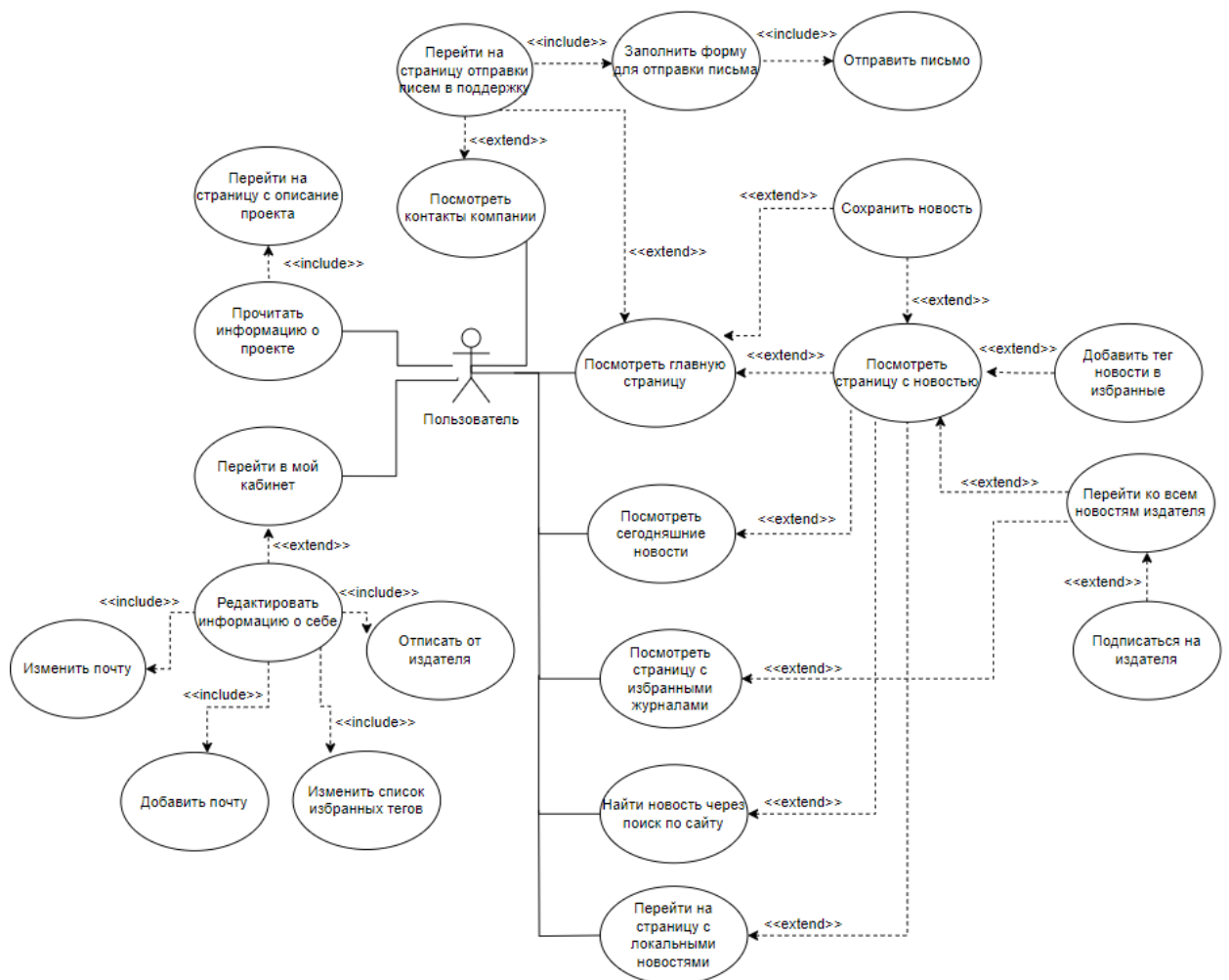


Рисунок 2 - Диаграмма прецедентов: Пользователь

Диаграмма прецедентов гостя описывает сценарии взаимодействия незарегистрированных пользователей, которые обладают ограниченными возможностями по использованию сервиса. Далее представлена диаграмма прецедентов для гостя:

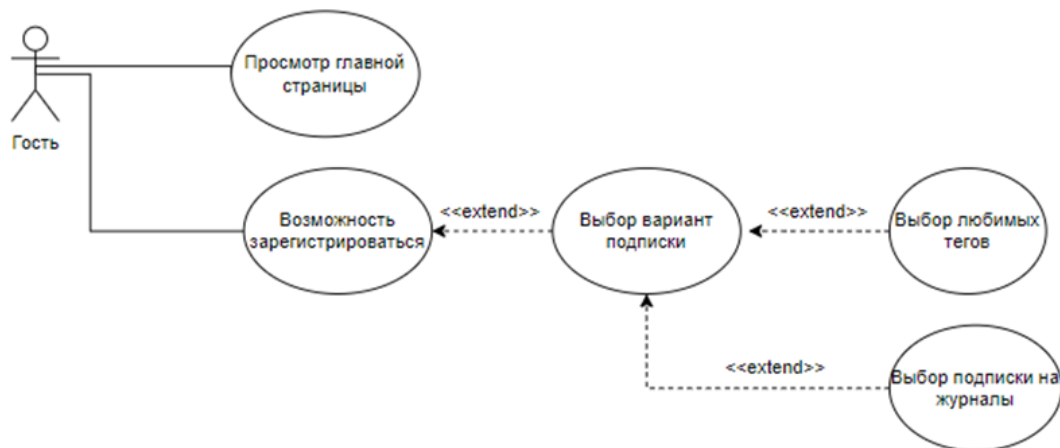


Рисунок 3 - Диаграмма прецедентов: Гость

Редактор является основным лицом, создающим контент для сервиса. В его возможности входит управление предоставляемым им контентом. Далее представлена диаграмма прецедентов для редактора:

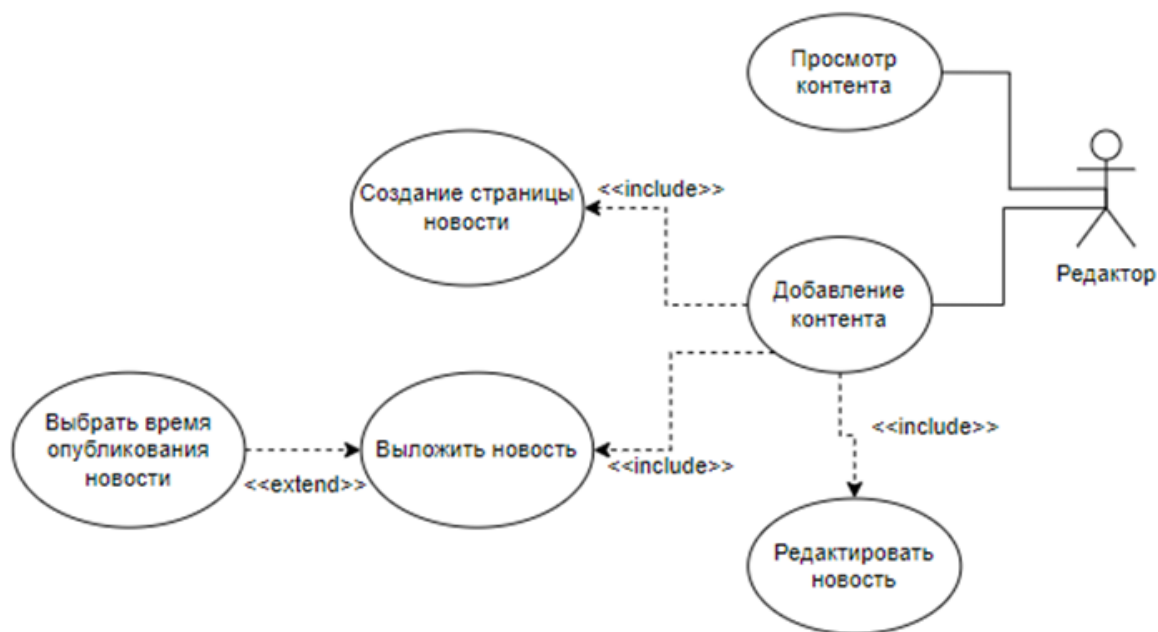


Рисунок 4 - Диаграмма прецедентов: Редактор

Администратор может управлять контентом. Также администратор управляет ролью редактора. Далее представлена диаграмма прецедентов для администратора:

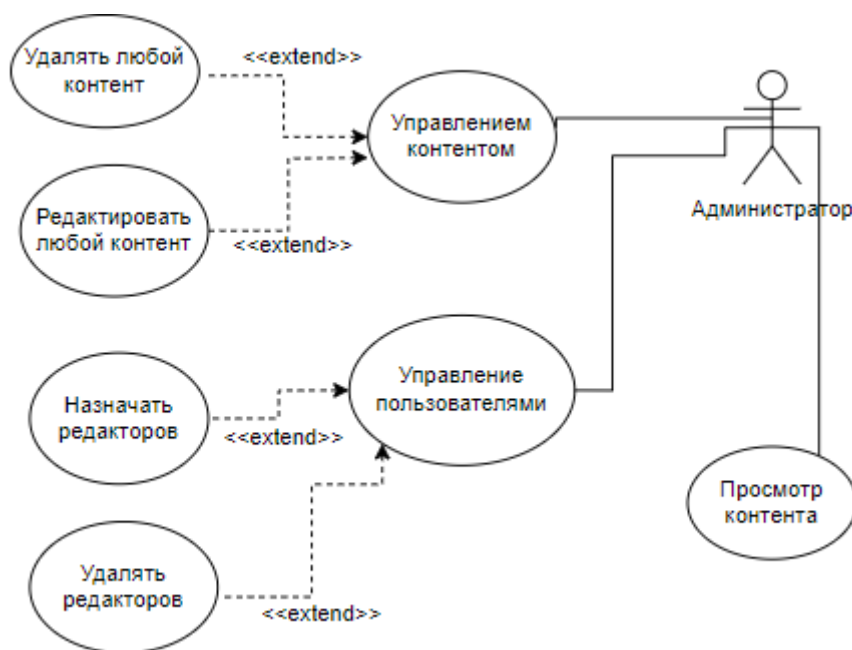


Рисунок 5 - Диаграмма прецедентов: Администратор

2.3.2 Диаграмма последовательности

Диаграмма последовательности взаимодействия пользователя с представляет собой отображение последовательности сообщений между клиентской и серверной частями при выполнении определенного сценария. Она демонстрирует, как клиентские объекты (например, пользовательский интерфейс) взаимодействуют с сервером для выполнения определенных операций или получения информации.

На диаграмме последовательности представлен каждый шаг взаимодействия для различных объекты (клиент, сервер. Каждое сообщение указывает на конкретное действие, которое выполняется, и может включать передачу данных или запрос на выполнение определенной операции.

Основной целью данной диаграммы последовательности является представление последовательность действий, необходимых для выполнения определенного сценария использования сервиса, иллюстрация взаимодействия между клиентом и сервером. Данная диаграмма нужна для отображения процесса взаимодействия.

Далее представлена соответствующая диаграмма последовательности:

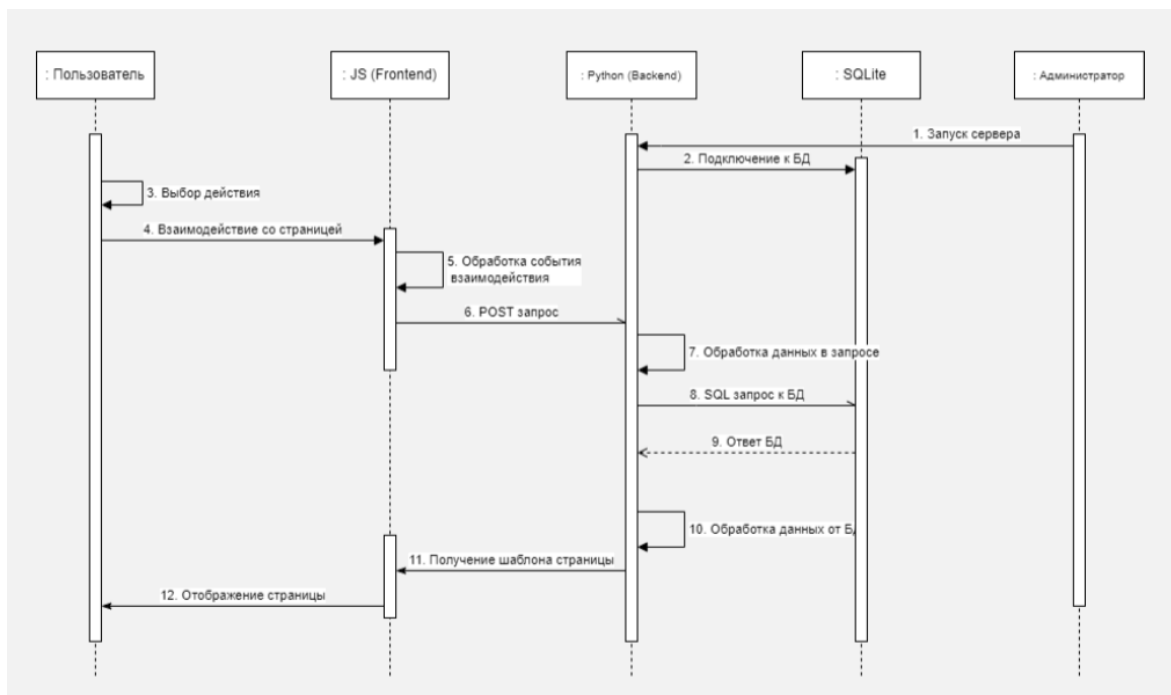


Рисунок 6 - Диаграмма последовательности взаимодействия пользователя и системы

2.3.3 Диаграмма развертывания

Диаграмма развертывания для сервиса представляет собой визуальное представление физической архитектуры системы, отображая размещение и взаимодействие различных компонентов приложения на различных узлах сети. Она позволяет понять, какие компоненты системы находятся на клиентской стороне, а какие на стороне сервера, а также как они связаны между собой.

Компоненты системы (например, клиентское приложение, серверное приложение, база данных) представлены в виде узлов.

Основная цель диаграммы развертывания - представить архитектуру системы и распределение ее компонентов. Это помогает разработчикам и системным администраторам понять, как организовано взаимодействие между клиентом и сервером, и обеспечить эффективную работу всей системы.

Далее представлена диаграмма развертывания для сервиса:

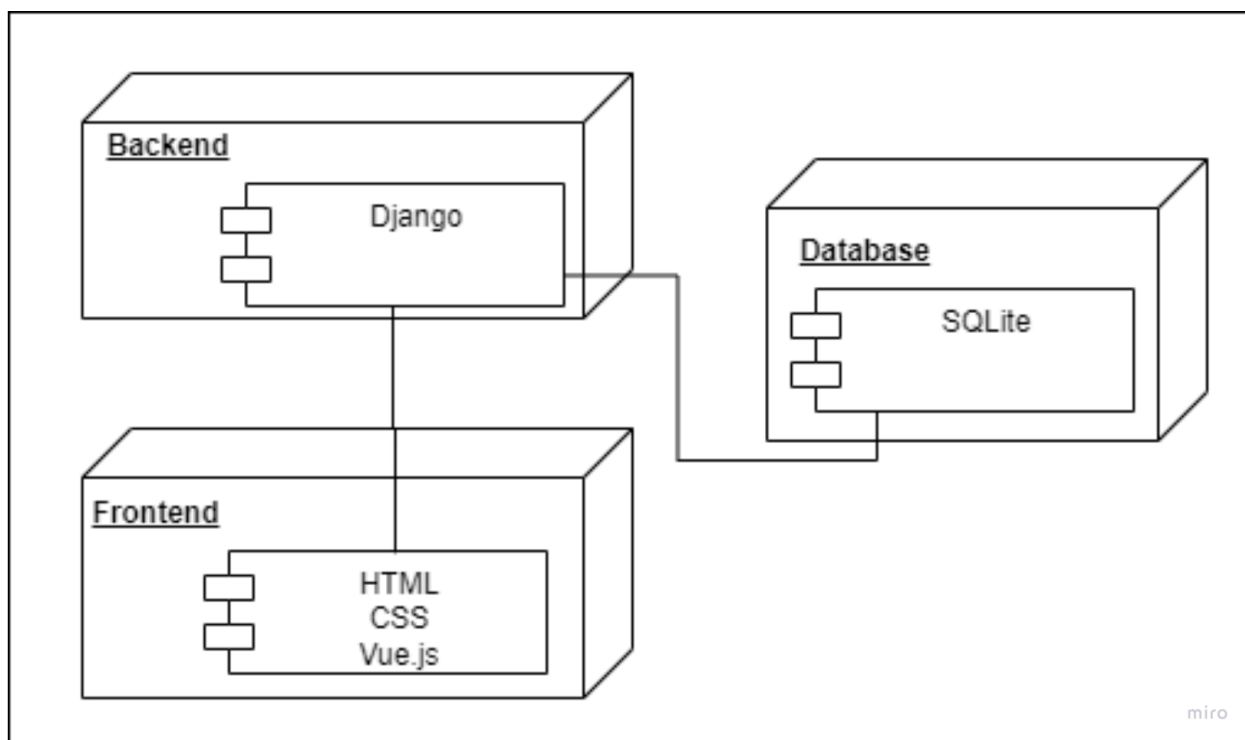


Рисунок 7 - Диаграмма развертывания

2.3.4 Диаграмма состояний

Диаграмма состояний для пользователя в сервисе является графическим представлением различных состояний, в которых может находиться

пользователь при взаимодействии с системой, а также переходов между этими состояниями. Это позволяет визуально представить поведение пользователя в процессе использования сервиса и его реакции на различные события.

Диаграмма состояний помогает понять, как пользователь взаимодействует с системой в различных сценариях использования, а также как система реагирует на действия пользователя. Это обеспечивает реализацию взаимодействия между пользователем и сервисом. Также диаграмма описывает получение роли редактора пользователем.

Далее представлена диаграмма состояний пользователя:

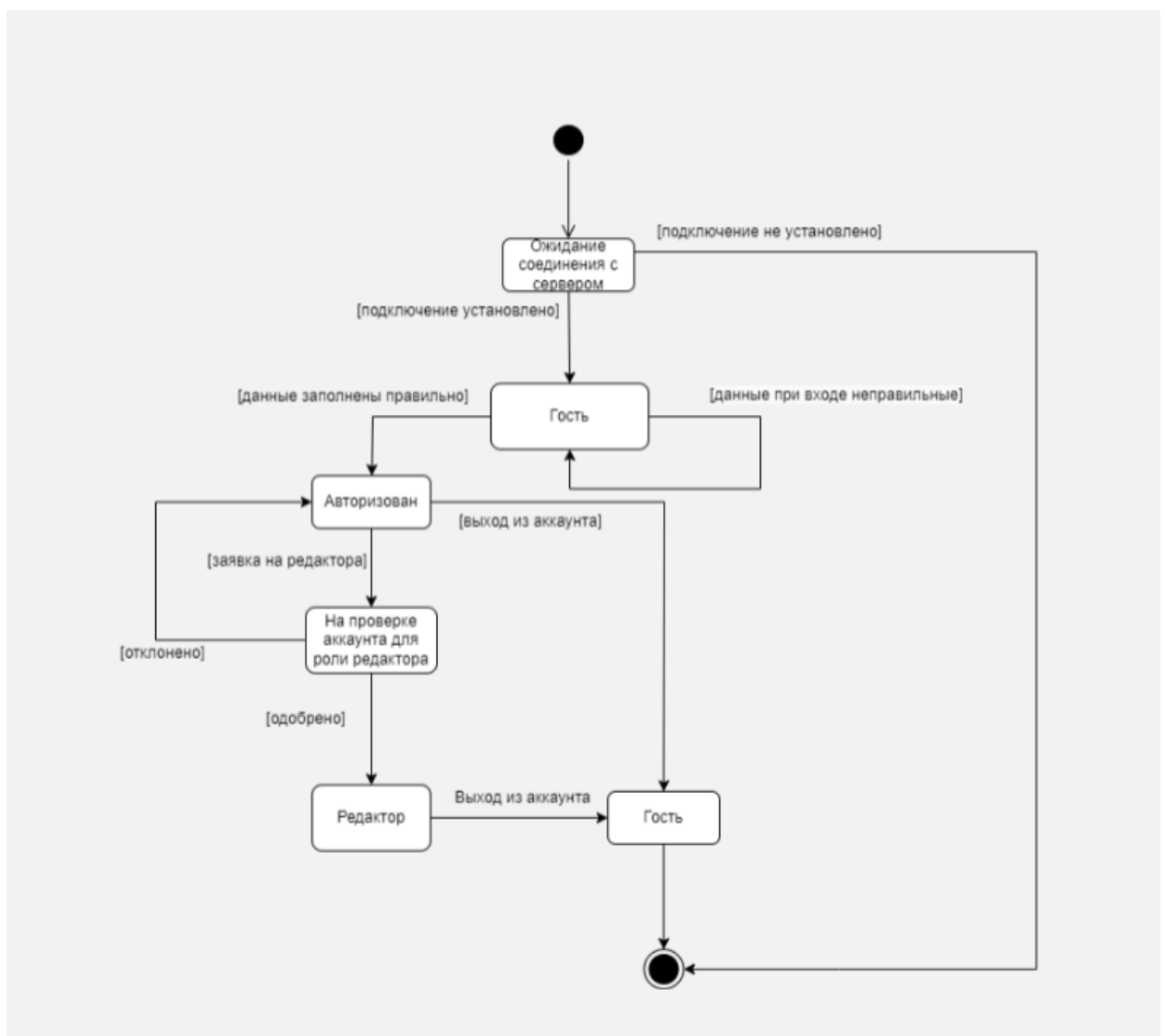


Рисунок 8 - Диаграмма состояний пользователя

2.3.5 Диаграмма активности

Диаграмма активностей для пользователя в данном сервисе представляет собой графическое отображение последовательности действий, которые пользователь выполняет при взаимодействии с системой. Она является визуальным представлением активностей пользователя в процессе использования сервиса и последовательности выполнения этих активностей.

На диаграмме активностей каждая активность описывает конкретное действие или операцию, выполняемую пользователем. Связи между активностями показывают порядок выполнения действий.

Диаграмма активностей отображает, как пользователь взаимодействует с системой в различных сценариях использования, и какие шаги он должен выполнить для достижения конкретной цели. Это показывает взаимодействие между пользователем и сервисом. Также она показывает путь пользователя от запуска страницы, до основной функциональности.

Далее представлена диаграмма активностей пользователя:

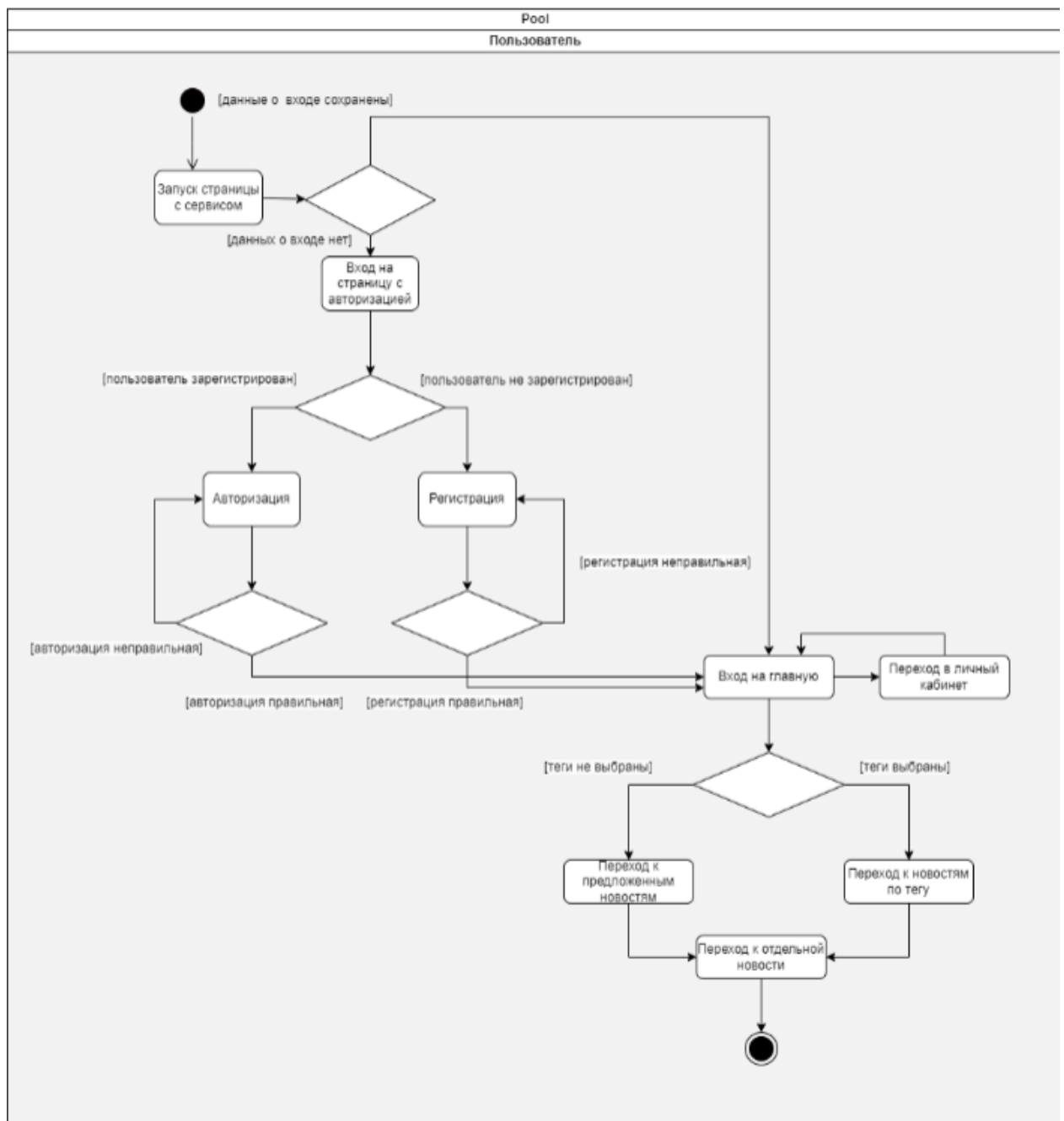


Рисунок 9 - Диаграмма активностей пользователя

3 Реализация

3.1 Средства реализации

Система спроектирована таким образом, чтобы обеспечивать высокую производительность и масштабируемость. Использование технологии клиент-серверной архитектуры с разделением логики между фронтендом и бэкендом позволяет системе эффективно обрабатывать большое количество одновременных запросов.

Таким образом, веб-приложение разработано на основе модели клиент-серверного взаимодействия. Приложение разделено на две основные части:

Серверная часть (Backend): Обработка запросов, работа с базой данных и логика приложения; обработка запросов от клиента, управление данными и выполнение логики приложения; обеспечение безопасности и защиты данных, обработка аутентификации и авторизации.

Клиентская часть (Frontend): Отображение информации на веб-странице и взаимодействие с пользователем; интерфейс пользователя, предоставляющий возможности настройки новостных лент, фильтрации контента и взаимодействия с новостями; интерактивные элементы, такие как выбор тегов, управление учетной записью и уведомления.

Для реализации части Frontend используются следующие технологии:

- HTML/CSS – используются для структурирования и стилизации веб-страниц.
- JavaScript – для динамического взаимодействия с пользователем.
- Vue.js – фреймворк для создания реактивных пользовательских интерфейсов, то есть, для построения пользовательского интерфейса, обеспечивающего высокую производительность и интерактивность.

Для реализации части Backend используются следующие технологии:

- Django – серверная платформа, обеспечивающая обработку запросов и взаимодействие с базой данных;

- SQLite – база данных для хранения информации о пользователях, новостях и тегах;
- Waitress – сервер, на котором запускается серверная платформа.

Для уже зарегистрированных пользователей реализована функция авторизации. Введенные данные проверяются, и при успешной авторизации осуществляется вход в систему. Пользователь также перенаправляется на главную страницу сайта.

Реализована возможность просмотра новостей и журналов. Неавторизованные пользователи могут просматривать до 10 новостей. При достижении лимита система уведомляет пользователя о необходимости регистрации или авторизации для продолжения просмотра.

Каждая новость имеет собственную страницу с названием, текстом и прикрепленным файлом. Просмотр журналов доступен только авторизованным пользователям. На страницах журналов отображаются название, текст и прикрепленный файл.

Реализована система тегов, позволяющая пользователям настраивать фильтрацию новостного контента. Пользователи могут выбрать интересующие их теги на отдельной странице сайта. При выборе конкретного тега система отображает только те новости и журналы, которые соответствуют выбранному тегу.

Реализованы функции редактирования информации об аккаунте для авторизованных пользователей. Пользователи могут изменить свои персональные данные, включая имя, адрес электронной почты и пароль.

Редакторы имеют возможность создавать, редактировать и удалять новости и журналы. Эти функции доступны через специальный интерфейс редактора, где они могут вводить текст публикации, добавлять прикрепленные файлы и назначать теги. Измененные данные отправляются на сервер для обновления базы данных.

Администраторы имеют расширенные права управления системой. Реализована возможность назначения редакторов, а также создание,

редактирование и удаление новостей и журналов. Администраторы могут управлять учетными записями пользователей, удаляя их при необходимости.

Реализована функция просмотра новостей от конкретного издателя. Пользователи могут выбирать издателей и просматривать публикации, исходящие только от них.

3.2 Характеристики системы

Система спроектирована таким образом, чтобы обеспечивать высокую производительность и масштабируемость. Использование технологии клиент-серверной архитектуры с разделением логики между фронтендом и бэкендом позволяет системе эффективно обрабатывать большое количество одновременных запросов.

Реализованы механизмы защиты от SQL-инъекций и других видов атак. Для хранения и передачи данных используется шифрование, что обеспечивает защиту конфиденциальной информации пользователей. Все пароли хранятся в зашифрованном виде с использованием алгоритмов хеширования (то есть, пароль проверяется по хешу, а не сравнивается в открытом виде). Реализована система контроля доступа, позволяющая разграничивать права пользователей, редакторов и администраторов.

Интерфейс приложения разработан с учетом принципов удобства использования (usability). Реализована локализация интерфейса на русский язык, что делает его удобным для русскоязычных пользователей.

Приложение совместимо с основными современными веб-браузерами (Google Chrome, Mozilla Firefox, Safari и другие), что обеспечивает доступность для широкой аудитории пользователей. Использование стандартных технологий (HTML, CSS, JavaScript) гарантирует корректную работу приложения в различных операционных системах и на разных устройствах.

3.3 Реализация Backend-части

Веб-приложение оптимизировано для минимального использования ресурсов, что снижает нагрузку на устройства пользователей и сервера. Это

достигается за счет использования эффективных алгоритмов обработки данных и минимизации объемов передаваемой информации. Структура приложения является стандартной для приложений, написанных с использованием Django.

Файл `models.py` содержит определения моделей базы данных, используемых в приложении. Здесь представлены модели для пользователей, тегов, новостей, журналов и статусов пользователей. Модель пользователя включает в себя поля для хранения информации о пользователе, такие как имя, адрес электронной почты и пароль. Модель тегов используется для классификации контента, предоставляя возможность привязывать теги к новостям и журналам. Модель новостей включает поля для заголовка, текста, даты публикации и прикрепленных файлов. Модель журналов аналогична модели новостей. Модель статусов пользователей позволяет отслеживать состояния и роли пользователей в системе, что важно для разграничения прав доступа.

Файл `settings.py` содержит настройки и конфигурацию для Django-приложения. Здесь указаны параметры базы данных, взаимодействие с которой осуществляется через `models.py`. Кроме того, `settings.py` содержит информацию о подключенных приложениях (`installed apps`), настройках статических файлов и шаблонов, а также других параметрах, влияющих на поведение и производительность приложения.

Файл `urls.py` отвечает за маршрутизацию запросов в Django-приложении. В нем определены URL-маршруты, связывающие запросы пользователей с соответствующими представлениями (`views`). Например, здесь можно найти маршруты для обработки запросов на главную страницу, страницы новостей, страницы регистрации и авторизации пользователей. Этот файл играет роль управления навигацией и доступом к различным частям приложения, обеспечивая обработку запросов и передачу их соответствующим обработчикам.

Директория `utils` содержит утилиты, используемые в приложении для выполнения различных вспомогательных функций. Например, `tag_utils.py` включает функции для работы с тегами. Например, поиск тегов позволяет искать не в точности по всем символам, а искать приблизительный результат поиска. Также в директории содержится утилита для рандомизации вывода новостей в соответствии с выбранными пользователями тегами, что позволяет улучшить пользовательский опыт, предлагая более релевантный контент. Другие утилиты в этой директории включают функции для обработки данных, валидации ввода и выполнения других общих задач, необходимых для реализации предоставленных сценариев.

3.4 Реализация Frontend-части

Часть Frontend разработана с использованием фреймворка `Vue.js`. Это достигается за счет использования эффективных алгоритмов обработки данных и минимизации объемов передаваемой информации. Структура приложения также является стандартной для приложений, написанных с использованием `Vue.js`.

В директории `components` находятся компоненты `Vue.js`, используемые в приложении. Компоненты представляют собой переиспользуемые части пользовательского интерфейса.

Файл `router.js` в соответствующей директории отвечает за маршрутизацию в приложении `Vue.js`. В нем определены маршруты для различных страниц и компонентов приложения. Например, он связывает URL-адреса с соответствующими компонентами, такими как домашняя страница, страница входа, регистрации и страницы новостей. Маршрутизация позволяет пользователям переходить между страницами без перезагрузки всего приложения, обеспечивая плавный и интуитивно понятный пользовательский интерфейс. То есть, фактически, загружается только один файл (`index.html`), но при помощи такой маршрутизации меняется и ссылка, и само представление.

В директории `views` находятся представления (views), которые представляют собой компоненты `Vue.js`, связанные с конкретными страницами приложения. Например, `main.vue` — это компонент, отображающий домашнюю страницу с основным контентом. `login.vue` и `signup.vue` содержат формы для входа и регистрации пользователей. Файл `news.vue` отображает полную информацию о выбранной новости, включая заголовок, текст и любые прикрепленные файлы. Файл `magazines.vue` аналогичен файлу `news.vue`, но предоставляет доступ к журналу. Также есть отдельный файл `choose_tags.vue`, который предоставляет доступ к части страницы с выбором тегов. Эти представления обеспечивают структурированное отображение данных и взаимодействие с пользователем.

Собственно, реализована навигация по различным частям сайта. Далее представлены отображения для определенных страниц.

На основной странице расположен список новостей, а также навигация по некоторым страницам (например, ссылка на страницу с выбором тегов).

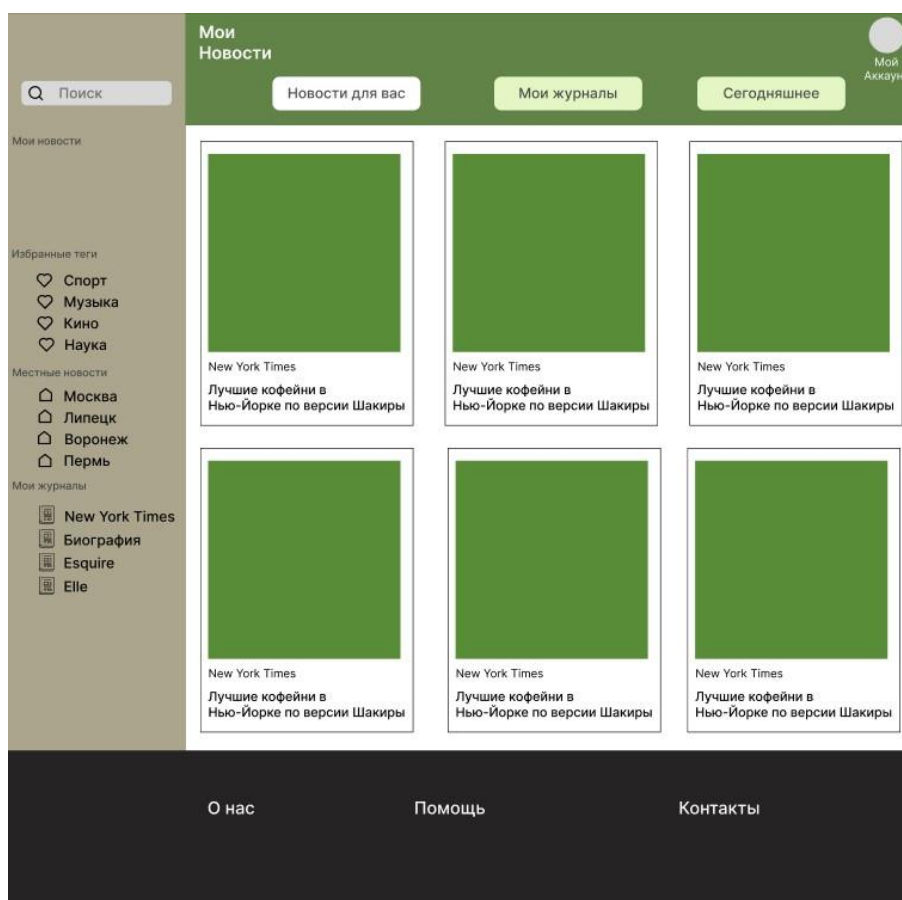


Рисунок 10 - Основная страница

Страница выбора тегов, соответственно, позволяет пользователю выбрать необходимые теги, которые будут влиять на отображение новостей и журналов.

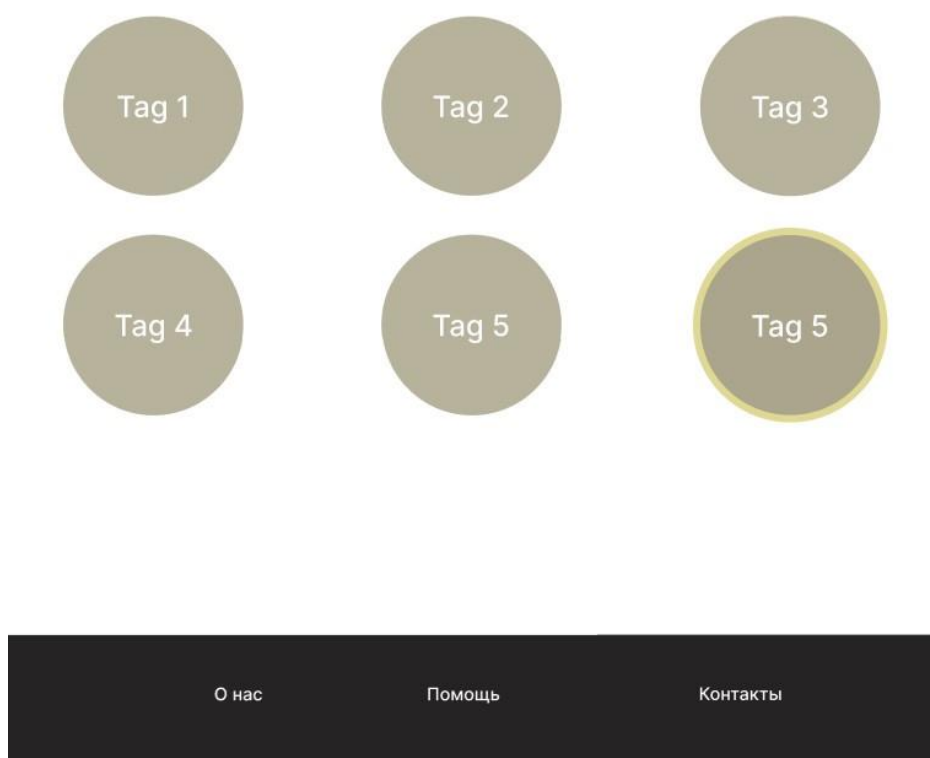
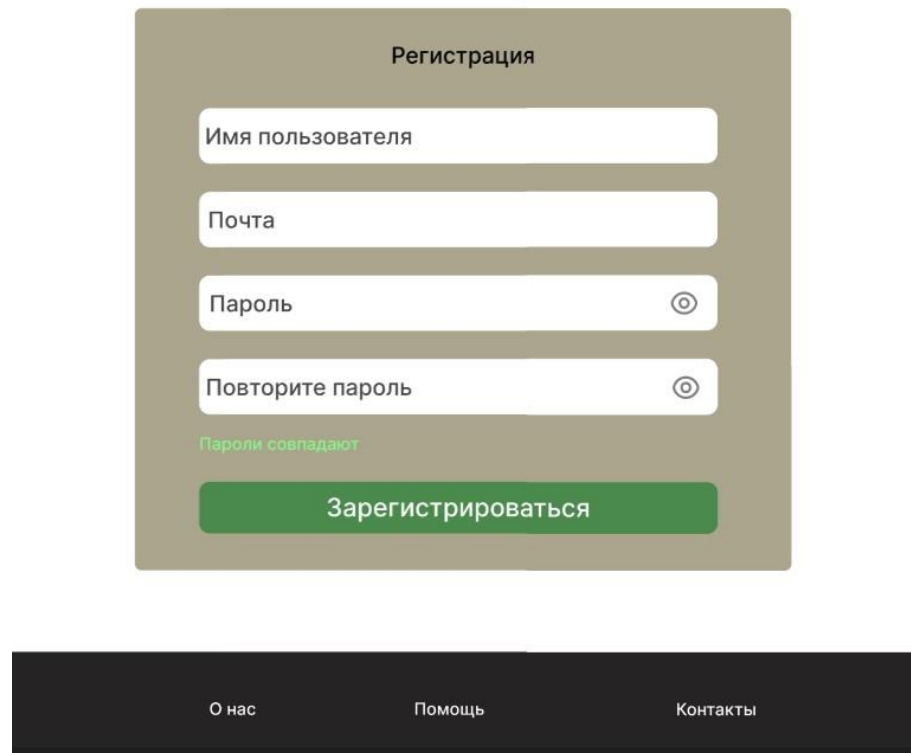


Рисунок 11 - Страница выбора тегов

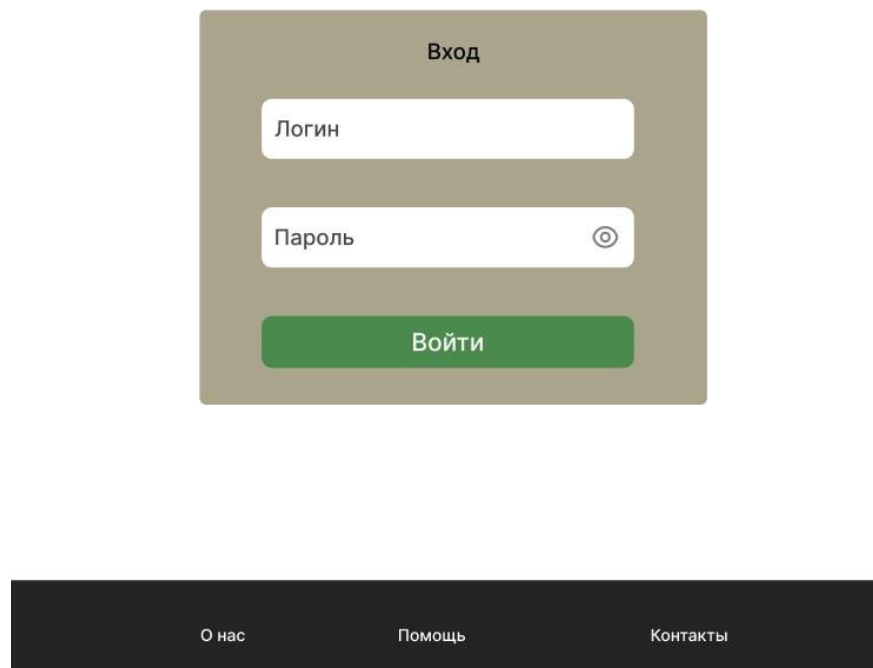
Страница регистрации доступна неавторизованным пользователям, позволяет создать аккаунт.



The registration form is titled "Регистрация" and is set against a light beige background. It contains four input fields: "Имя пользователя", "Почта", "Пароль", and "Повторите пароль". The password fields include an eye icon for toggling visibility. A green confirmation message "Пароли совпадают" is displayed below the second password field. A large green button labeled "Зарегистрироваться" is positioned at the bottom of the form. Below the form is a dark grey footer bar with three links: "О нас", "Помощь", and "Контакты".

Рисунок 12 - Страница регистрации

Страница авторизации позволяет неавторизованным пользователям войти в аккаунт.



The authorization form is titled "Вход" and is set against a light beige background. It contains two input fields: "Логин" and "Пароль". The password field includes an eye icon for toggling visibility. A large green button labeled "Войти" is positioned at the bottom of the form. Below the form is a dark grey footer bar with three links: "О нас", "Помощь", and "Контакты".

Рисунок 13 - Страница авторизации

Страница новости содержит текст и прикрепленный файл.

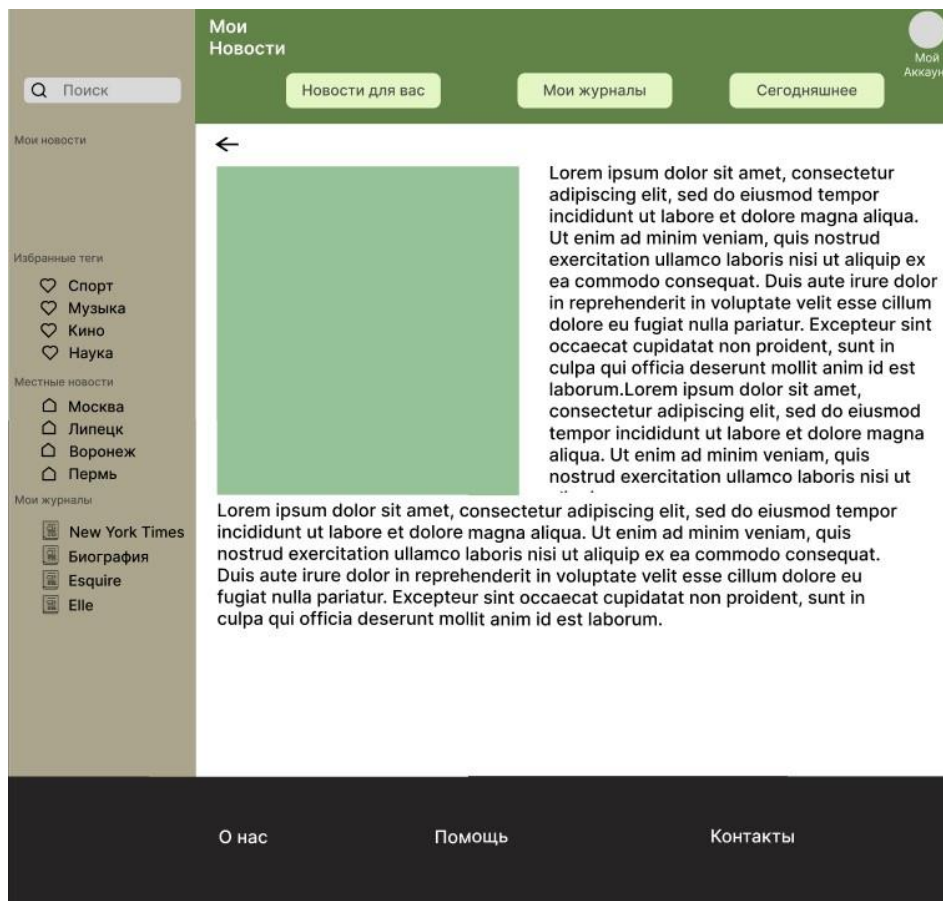


Рисунок 14 - Страница отдельной новости

Страница выбора журналов аналогична основной странице, но позволяет выбирать не новости, а журналы.

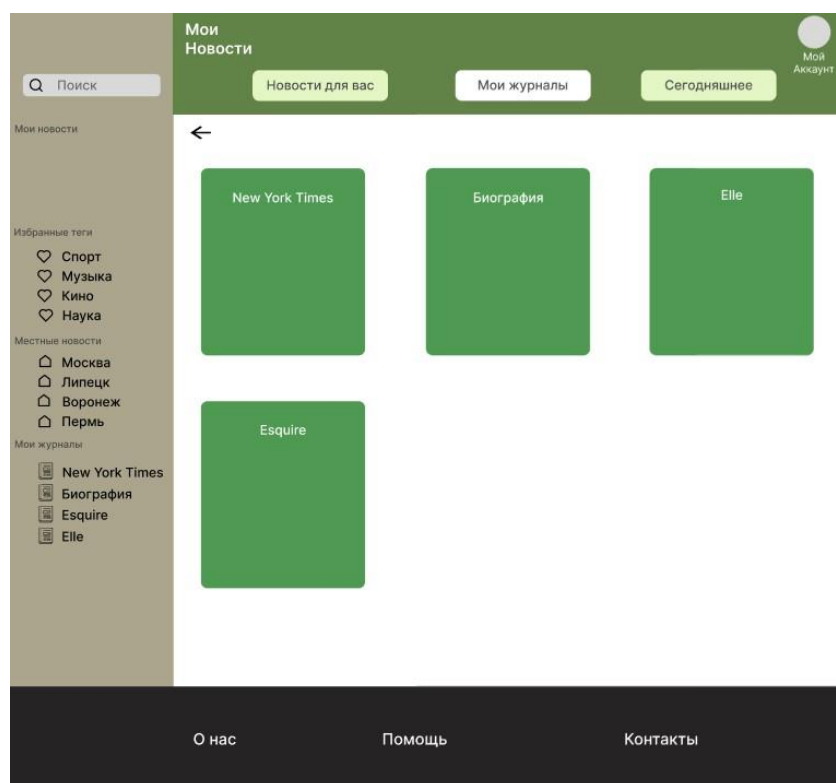


Рисунок 15 - Страница выбора журналов

Страница аккаунта пользователя доступна только авторизованным пользователям. На ней отображается информация о пользователе, а также она позволяет пользователю сменить эту информацию.

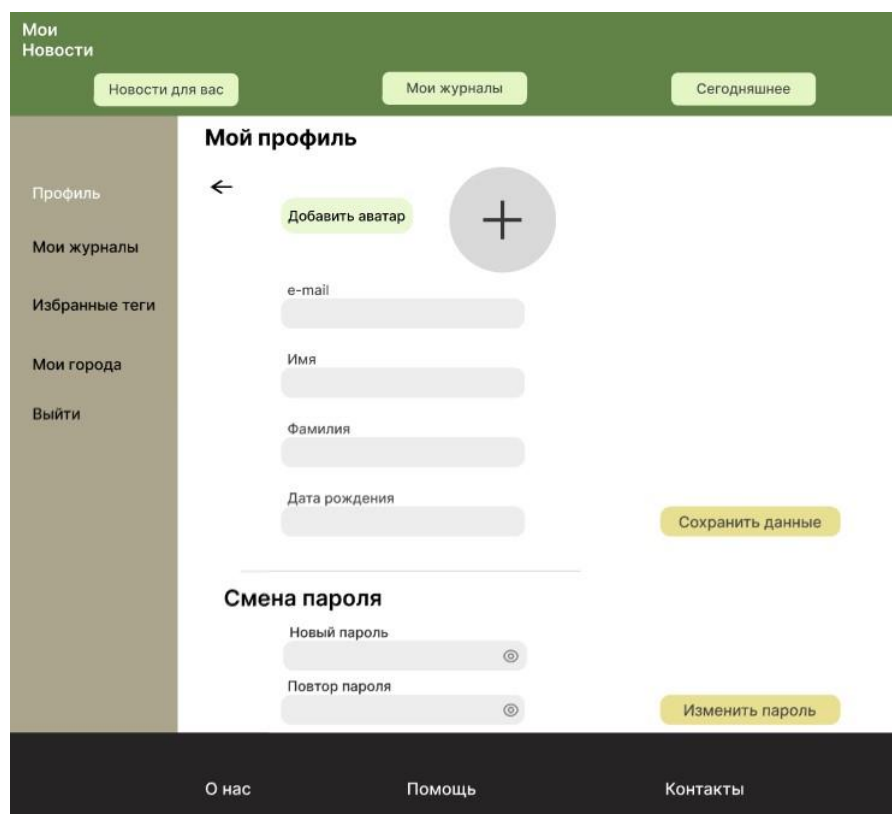


Рисунок 16 - Страница аккаунта

Заключение

Проект "Мои Новости" представляет собой современный сервис для создания и управления персонализированными новостными лентами, который был успешно реализован в соответствии с функциональными и нефункциональными требованиями, изложенными в техническом задании. Система включает в себя клиент-серверную архитектуру на базе Django для серверной части и Vue.js для клиентской части, обеспечивая высокую производительность, масштабируемость и удобство использования.

В процессе реализации были реализованы такие функции приложения, как регистрация и авторизация пользователей, просмотр и управление новостями и журналами, система тегов для фильтрации контента, а также управление аккаунтами и ролями пользователей. Особое внимание было уделено безопасности данных, включая защиту от SQL-инъекций, шифрование данных и контроль доступа.

Проект также соответствует высоким стандартам надежности и отказоустойчивости. Русский язык делает систему доступной для русскоязычных пользователей на различных устройствах.

Модульная архитектура проекта и файловая структура, описанная ранее, обеспечивают простоту поддержки и расширения функциональности в будущем.

Итогом работы стало создание сервиса с возможностями персонализации новостного контента, который готов к использованию и дальнейшему развитию.

Список использованных источников

1. Django – URL: <https://www.djangoproject.com/> (Дата обращения 23.04.2024). – Текст: электронный.
2. Vue.js – URL: <https://vuejs.org/> (Дата обращения 23.04.2024). – Текст: электронный
3. HTML – URL: <https://developer.mozilla.org/en-US/docs/Web/HTML> (Дата обращения 23.04.2024). – Текст: электронный
4. CSS – URL: <https://developer.mozilla.org/en-US/docs/Web/CSS> (Дата обращения 23.04.2024). – Текст: электронный
5. Node.js – URL: <https://nodejs.org/> (Дата обращения 23.04.2024). – Текст: электронный
6. SQLite – URL: <https://www.sqlite.org/> (Дата обращения 23.04.2024). – Текст: электронный