

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА
на тему
Программные средства поддержки
процессов организации и
функционирования новостного портала

Студент _____ Персицков Данила Дмитриевич
(подпись, дата) (ФИО полностью)

Руководитель _____ Князев В.Н.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А. Ю. Козлов
«__» _____ 202__ г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент _____ Персицков Данила Дмитриевич _____ гр. 19ВП2 факультета _____ ВТ _____

<Фамилия, имя, отчество полностью>

2. Тема работы _____ Программные средства поддержки процессов организации и функционирования новостного портала _____

Тема утверждена приказом ПГУ № _____ от “___” _____ 2023

3. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Князев В.Н. _____

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) *Разработать приложение для мобильных устройств, обеспечивающее поддержку процессов организации и функционирования новостного портала.*

2) *Среда выполнения приложения: операционная система Android 6 и выше
Среда выполнения сервера: операционная система Windows 10 и выше
Языковые средства разработки: C#, Python, UML, SQL*

*Инструментальные средства разработки: Microsoft SQL Server 2018
Microsoft Visual Studio 2019, Microsoft Project, Draw.io, Visual Paradigm*

Технологии разработки: RUP, ООП, WPF, Entity Framework, Xamarin Forms, Keras

3) *Основные функции мобильного приложения:*

а) создание новостей и публикация их в системе;

б) чтение опубликованных новостей в системе;

в) редактирование и удаление своих опубликованных новостей;

г) управление подписками на другие профили пользователей;

д) добавление оценок и комментариев к новостям.

5. Перечень подлежащих разработке вопросов

1) *Анализ предметной области и постановка задачи*

2) *Проектирование*

3) Реализация

4) Контроль качества программного обеспечения

5) Оформление пояснительной записки

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	<i>Анализ предметной области постановка задачи</i>	15%	12.04.2023 – 16.04.2023	
2	<i>Проектирование</i>	25%	17.04.2023 – 26.04.2023	
3	<i>Реализация</i>	40%	27.04.2023 – 11.05.2023	
4	<i>Контроль качества программного обеспечения</i>	5%	12.05.2023 – 21.05.2023	
5	<i>Оформление пояснительной записки</i>	15%	22.05.2023 – 01.06.2023	

Дата выдачи задания «22» декабря 2022г.

Руководитель бакалаврской работы _____

(подпись, дата)

Задание к исполнению принял студент _____

(подпись, дата)

Работу к защите допустить

Декан ФВТ _____ Л.Р. Фионова

(подпись, дата)

Реферат

Пояснительная записка содержит 198 листов, 92 рисунка, 18 таблиц, 34 использованных источников, 5 приложений.

КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, НОВОСТНОЙ ПОРТАЛ, ANDROID, MS SQL SERVER, C#, RUP, UML, XAMARIN FORMS

Объектом разработки является мобильное приложение для поддержки процессов организации и функционирования новостного портала.

Цель работы – создание мобильного приложения для новостного портала, позволяющего пользователям публиковать, редактировать и удалять новости, читать новости других пользователей, добавлять комментарии и оценки, управлять своим профилем и подписками на профили других пользователей.

Разработка серверной части происходила на языке программирования C#. Для разработки нейронной сети использовался язык Python с библиотекой Keras. В качестве среды разработки использовалась Microsoft Visual Studio 2019. В качестве СУБД была использована СУБД MS SQL Server.

Клиентское приложение разрабатывалось на языке программирования C# в среде Microsoft Visual Studio 2019, с использованием фреймворка Xamarin Forms. Оно спроектировано для работы на устройствах под управлением операционной системы Android версии 6.0 и выше, а серверная его часть работает на операционной системе Windows 10 и выше.

					ПГУ 09.03.04 – 08БР192.17 ПЗ			
Изм.	Лист	№ докум.	Подп.	Дата	Программные средства поддержки процессов организации и функционирования новостного портала. Пояснительная записка	Лит.	Лист	Листов
Разраб.		Персицков Д.Д.					4	198
Пров.		Князев В.Н.						
Н. контр.		Такташкин Д.В.				Группа 19ВП2		
Утверд.		Козлов А.Ю.						

Содержание

Введение.....	7
1 Анализ предметной области и требований	11
1.1 Анализ предметной области и постановка задачи	11
1.2 Анализ программных продуктов аналогичного назначения	12
1.2.1 Новостной портал RB.RU.....	12
1.2.2 Интернет–ресурс РИА Новости.....	13
1.2.3 Новостной портал Пензаинформ.....	15
1.2.4 Сравнение аналогов	17
1.3 Анализ целевой аудитории	17
1.4 Анализ функциональных требований.....	22
1.5 Анализ нефункциональных требований.....	28
1.6 Планирование разработки и оценки бюджета	29
2 Проектирование мобильного приложения	48
2.1 Выбор технологий и инструментальных средств разработки.....	48
2.2 Проектирование архитектуры программного обеспечения.....	50
2.3 Проектирование интерфейса пользователя.....	51
2.4 Логическое проектирование базы данных	58
2.5 Проектирование серверной части приложения	63
2.6 Проектирование нейронной сети	64
3 Реализация мобильного приложения.....	66
3.1 Реализация базы данных	66
3.2 Разработка серверной части.....	67
3.3 Реализация клиентской части приложения	69
3.4 Реализация нейронной сети для классификации текста	72
4 Тестирование программного продукта	74
4.1 Тестирование серверной части приложения	74
4.2 Тестирование клиентской части приложения.....	77
4.2.1 Функциональное тестирование.....	77
4.2.2 Тестирование приложения методом чёрного ящика	81

	6
4.2.3 Модульное тестирование	82
4.3 Метрики кода.....	85
4.3.1 Метрики размера программы.....	86
4.3.2 Метрики Холстеда	87
Заключение	90
Список использованных источников	92
Приложение А. Исходный код скриптов базы данных	96
Приложение Б. Исходный код серверной части	100
Приложение В. Исходный код мобильного приложения	123
Приложение Г. Результаты тестирования серверной части	169
Приложение Д. Результаты тестирования мобильного приложения	182

Введение

В настоящее время компьютерные технологии стали неотъемлемой частью нашей жизни и вносят существенные изменения в наше общество. С помощью компьютеров и мобильных устройств мы можем получать новости и актуальную информацию из любой точки мира, что позволяет нам быть в курсе различных происшествий. Кроме того, их использование применяется в различных областях, таких как бизнес и наука, увеличивая скорость работы для получения требуемого результата. Таким образом, можно сказать, что компьютеризация оказывает положительное влияние на нашу жизнь и продолжает улучшать ее каждый день.

Россия, как и многие другие страны, развивает цифровые технологии для различных сфер жизни. Сегодня можно увидеть внушительный прогресс в области информационных технологий, который был совершен за последние года. Например, Россия является одним из лидеров в области кибербезопасности, обучению высоко квалифицированных специалистов в сфере компьютерных технологий, разработки игр и мультимедиа. Кроме того, научные кадры в России распространяют и развивают новые технологии в медицине и промышленности, что приводит к улучшению качества жизни граждан и повышает престиж нашей страны на мировой арене.

Наша страна осознаёт важность развития компьютерных технологий, так как они являются в настоящее время фундаментом для многих сфер в экономике. В последние годы Российская Федерация создаёт условия для разработок в сфере компьютерных технологий, включая искусственный интеллект и квантовые технологии. В 2019 году был подписан Указ Президента Российской Федерации о развитии искусственного интеллекта, который предусматривает создание соответствующих инфраструктурных и правовых условий для развития ИИ [1]. С 2021 года в России начинает развиваться инновационный научно-технологический центр «Квантовая долина», который занимается разработкой передовых цифровых технологий,

квантовых технологий и высокотехнологичных изделий на их основе [2]. Эти и другие действия направлены на качественное улучшение жизни граждан.

Современный мир, включая Россию, становится все более мобильным и цифровым. Мобильные устройства с доступом в Интернет играют важную роль в этом процессе, кроме того, они все больше вытесняют традиционный доступ к сети через настольные компьютеры. Это связано с тем, что мобильный интернет становится все более удобным и доступным для людей, которые желают быть онлайн в любое время и в любом месте. Согласно исследованию, опубликованному на ТАСС в 2020 году, 67% россиян используют смартфоны для доступа в Интернет [3]. В связи с этим, можно предположить, что традиционный доступ к сети через настольные компьютеры будет не так привлекателен для пользователей, как мобильный интернет в будущем. Аналогичные тенденции можно наблюдать и в остальных странах мира.

Цифровые технологии внесли изменения, которые отражаются на взаимодействии людей друг с другом, а также на получении информации, в том числе новостей. Сегодня подавляющее число людей узнаёт о последних новостях через интернет и социальные сети, а не через привычные источники, такие как газеты или телевидение. Это произошло благодаря качественному скачку в области компьютерных технологий, а также тот факт, что традиционные средства уведомления граждан не могут предоставить такой же широкий функционал и степень удобства, как использование электронных средств.

Одним из ключевых факторов, который привел к такому развитию, является доступность Интернета для широкой аудитории. Благодаря улучшению в сфере цифровых технологий и снижению стоимости устройств, которые участвуют в сети Интернет, все больше стало обладать возможностью к получению актуальных новостей онлайн. Кроме того, цифровизация позволяет быстро и эффективно распространять новости по всему миру, даже

в режиме реального времени, что делает их более актуальными и интересными для людей.

Также стоит отметить качественные изменения, которые появились в момент распространения новостей через мировую сеть. Раньше для того, чтобы опубликовать новость, нужно было иметь доступ к телевизионному или радиоэфиру, или иметь возможность распечатать ее в газете. Сегодня же любой человек с доступом к Интернету может публиковать новости и статьи, благодаря социальным сетям и другим онлайн-платформам, на которых можно высказывать свою субъективную точку зрения на события.

В России на законодательном уровне поддерживается создание отечественного программного обеспечения, которое работает с информацией. В указе президента Российской Федерации о стратегии развития информационного общества на период 2017-2030 годов указывается, что технологически независимое программное обеспечение и сервисы, которые могут быть использованы на всей территории Российской Федерации, обеспечены гарантийной и технической поддержкой российских организаций [4]. Также отмечается, что развитие российского программного обеспечения предоставляет большие возможности для создания качественных и современных сервисов для работы с информацией.

В России не только поддерживаются отечественные программные продукты, но и новостные порталы получают государственную поддержку. Так, в 2023 году Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации продолжило поддерживать развитие электронных СМИ [5]. Были выделены средства на финансирование проектов, направленных на развитие российских новостных порталов, а также на поддержку независимых журналистов. Эта поддержка позволяет развиваться новостным порталам и продолжать предоставлять качественную информацию своим пользователям.

На основании приведенных факторов создание собственного новостного портала является актуальным и перспективным решением. В

наше время мобильные устройства и доступ в интернет становятся неотъемлемой частью нашей жизни, а спрос на получения свежих новостей растет постоянно. Благодаря поддержке, а также заинтересованности со стороны правительства Российской Федерации, создание мобильного приложения, представляющего собой новостной портал, является многообещающим проектом. Такое приложение позволит пользователям быстро и удобно получать свежие новости на своих мобильных устройствах, а благодаря государственной поддержке, имеет все шансы на успех в будущем.

1 Анализ предметной области и требований

1.1 Анализ предметной области и постановка задачи

Разрабатываемое мобильное приложение должно предоставить пользователям удобный интерфейс к последним новостям через мобильные устройства. При использовании приложения, пользователи смогут получать актуальную информацию в кратчайшие сроки. Достаточно будет иметь мобильное устройство с установленным приложением и доступ к интернету. Это позволит пользователям быть в курсе последних новостей и событий на момент их появления, что является весьма удобным.

Приложение позволяет зарегистрированным пользователям быстро получать новости и делиться ими с другими читателями, при этом новости будут привязаны к конкретному адресу, что даст способность пользователям получать информацию по интересующему их адресу или местности. Пользователи могут создавать, редактировать, удалять новости, а также оценивать и комментировать их. Для удобства использования и обратной связи предусмотрены инструменты управления профилями и подписками.

Таким образом, в приложении должны быть представлены следующие основные функции:

- регистрация новых пользователей;
- вход зарегистрированных пользователей в систему;
- создание, чтение, редактирование и удаление новостей;
- управление профилем;
- управление подписками и отписками;
- оставление оценок и комментариев.

Для поддержания порядка в системе действуют специально назначенные модераторы, которые следят за соответствием новостей критериям публикации и имеют права для блокировки пользователей, которые нарушили правила сервиса. Модераторы принимают решение об

одобрении или отклонении новостей для публикации в приложении, основываясь на наличии в них нецензурной лексики, недопустимого содержания или изображений.

1.2 Анализ программных продуктов аналогичного назначения

В современном мире очень много программных продуктов, связанных с размещением новостей. Для сравнения рассмотрим несколько ярких представителей данного направления.

1.2.1 Новостной портал RB.RU

Платформа была запущена в октябре 2012 после привлечения seed-раунда от инвестиционной компании Black Ocean. В то время ресурс представлял собой отраслевое издание о венчурном бизнесе, сервис по поиску инвесторов и базу данных обо всех сделках на рынке [6].

Главная страница представлена на рисунке 1.

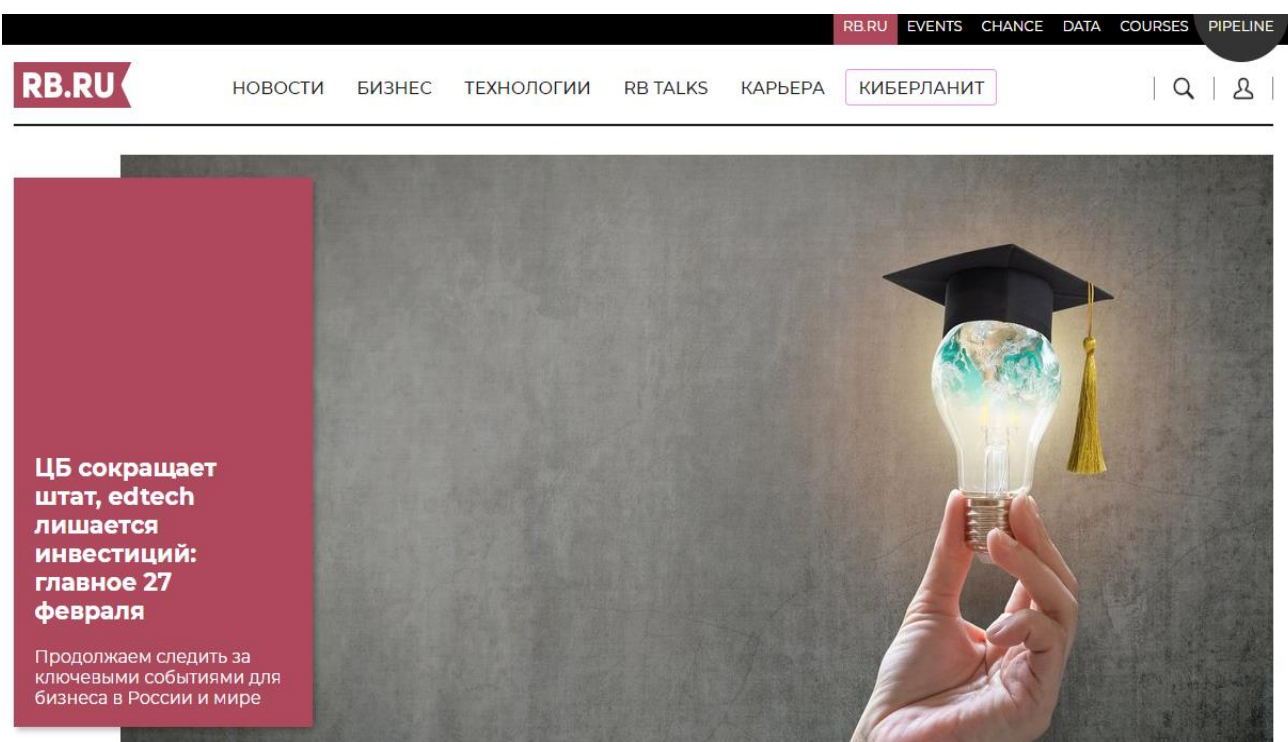


Рисунок 1 – Главная страница RB.RU

По последним данным, RB.RU посещает около одного миллиона уникальных пользователей в месяц, которые просматривают почти 2,5 миллион страниц. Редакция ресурса фокусируется не только на текстах, но и на видео, интерактивных проектах и других новых форматах. Данный проект позиционирует себя как независимое издание, где основной акцент падает на технологии и бизнес, а также организацию мероприятий для предпринимателей, инвесторов и корпораций.

Данный новостной портал имеет достаточно широкие функциональные возможности, которые позволяют просматривать публикации, фильтровать по тегам статьи и делиться новостями с помощью популярных отечественных мессенджеров. Ресурс имеет удобный интерфейс как для использования на персональном компьютере, так и на мобильном устройстве, на котором можно использовать либо мобильную версию сайта, либо специальное приложение.

Также портал имеет несколько недостатков, такие как отсутствие возможности оставлять новости и статьи обычным пользователям, отсутствие средств для оценивания и комментирование новостей, слабая поддержка мобильной версии ресурса и достаточно специализированная новостная повестка.

1.2.2 Интернет–ресурс РИА Новости

С 8 июня 2014 года РИА Новости – действующее информационное агентство. В качестве зарегистрированного СМИ, включая сетевое издание RIA.RU, входит в медиагруппу «Россия сегодня» и является частью этого бренда. С этого времени РИА Новости обеспечивает свою аудиторию точной и оперативной информацией обо всех важнейших событиях в России и за рубежом. Большую роль в достижении этого играет огромная команда корреспондентов, расположенных в 125 точках мира.

Главный интерфейс мобильного приложения данного новостного портала представлен на рисунке 2.

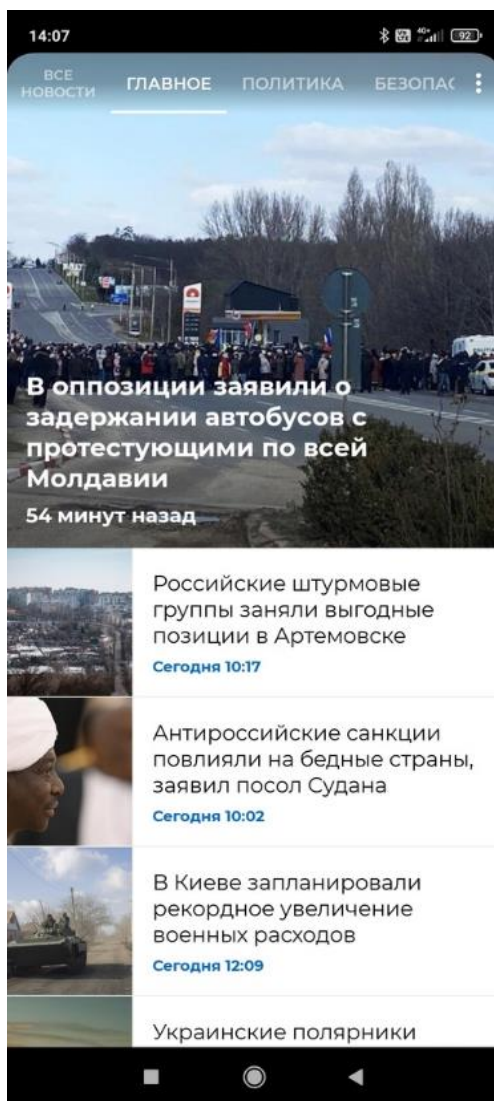


Рисунок 2 – Главная страница мобильного приложения РИА Новости

Следует отметить, что данный ресурс находится под санкциями некоторых стран в связи с политической ситуацией и вопросами свободы слова [7, 8]. Несмотря на это, РИА Новости остается одним из ведущих источников новостей на русском языке и предоставляет своим читателям широкий спектр информации.

Новости, которые публикуются на новостном портале, имеют самые разные тематики, начиная от социальных и заканчивая политическими. Новостной портал поддерживает Web-страницу и мобильное приложение. Новостной портал обеспечивает полный доступ к публикациям без каких-либо ограничений.

Данный новостной портал считается одним из лидеров по нескольким причинам, а именно простым интерфейсом, интуитивно понятной навигацией, большим спектром обозреваемых новостей, поддержкой мобильного приложения и бесплатным доступом ко всем публикациям.

Несмотря на плюсы, ресурс также содержит и минусы, такие как наличие навязчивой рекламы, частичное отсутствие оценивания новостей, за исключением использования готовых оценок, полное отсутствие пользовательских новостей.

1.2.3 Новостной портал Пензаинформ

Пензаинформ – российское информационное интернет-агентство, сетевое издание, основанное 26 декабря 2011 года, специализируется на новостях Пензенской области и Пензы, освещении общероссийских и мировых событий. Редакция находится в городе Пензе.

На рисунке 3 представлен главный интерфейс web-страницы новостного портала.



Рисунок 3 – Главный интерфейс новостного портала ПензаИнформ

Информационное агентство публиковало различные новости, прежде всего связанные с жизнью Пензы и Пензенской области. Портал стал правопреемником одного из первых информационных ресурсов региона PNZ.ru, где выкладывались видеорепортажи телеканала «Наш дом».

При создании ИА «ПензаИнформ» вся база информационных материалов портала PNZ.ru была перенесена на новый ресурс. Архив новостей охватывает период со 2 января 2004 года.

Целью создания информационного интернет-агентства «ПензаИнформ» послужило желание редакции оперативно информировать интернет-сообщество о событиях в Пензенской губернии, представить палитру мнений специалистов, помочь пензенским СМИ в получении достоверной информации [9].

По данным Liveinternet.ru сайт penzainform.ru по итогам 2020 года занимал первое место по посещаемости в рейтинге сайтов СМИ Пензенской области.

По данным компании «Медиалогия» за 2020 год сетевое издание «ПензаИнформ» является первым в рейтинге медиаресурсов Пензенской области, основой для построения которого стал Индекс Цитируемости.

Новостной портал предлагает уникальный контент и широкий набор функциональных возможностей. Кроме того, на ресурсе имеется мобильная версия сайта, что делает его удобным для использования на мобильных устройствах. Благодаря высокому качеству новостей, данный ресурс является перспективным и интересным источником информации.

Несмотря на плюсы, портал имеет серьёзные недостатки. Новости отображаются смешано с другой информацией, а интерфейс перегружен различными кнопками и мигающими баннерами, что затрудняет получение информации. Навигация также запутана и некорректна, часто при попытке поиска сервис выдаёт совершенно другие варианты. Ресурс содержит много рекламы, что делает работу с сервисом некомфортной. Кроме того, данный ресурс не предоставляет возможности пользователям создавать свои новости

и делиться ими, а также отсутствуют механизмы, которые отвечают за комментирование новостей.

1.2.4 Сравнение аналогов

После приведения нескольких аналогов, имеющих схожие полностью или частично функции, необходимо подвести итоги для текущей работы. Для этого будет составлена сводная таблица по критериям, таким как наличие карты геолокации, возможность создания пользовательских новостей и статей с возможностью оценивания и комментирования, список тем, которые затрагивают новости, и наличие мобильного приложения.

Сравнительный анализ рассмотренных выше новостных порталов представлен в таблице 1.

Таблица 1 – Сравнение аналогов

Новостные порталы	Наличие карты	Возможность создания пользовательских новостей	Способы оценивания	Список обозреваемых новостей	Мобильное приложение
RB.RU	Нет	Нет	Отсутствуют	Узкий, бизнес и технологии	Нет
РИА Новости	Нет	Нет	Частичные, использование готовых оценок	Широкий спектр	Есть
ПензаИнформ	Нет	Нет	Отсутствуют	Широкий спектр	Нет
Разрабатываемое приложение	Да	Да	Существуют	Без ограничений	Есть

Таким образом, в результате анализа систем аналогичного назначения было принято решение о разработке собственного приложения, в котором будут устранены недостатки аналогичных продуктов.

1.3 Анализ целевой аудитории

Целевой аудиторией проекта будут являться пользователи новостных порталов, которые хотят читать новости и создавать собственные новости, используя мобильное устройство. Такая потребность возникает в связи с растущей популярностью мобильных устройств и удобством, которое они предоставляют в доступе к информации. В современном мире количество

пользователей новостными порталами имеет огромный размер, об этом свидетельствует статистика трафика посетителей одного из крупнейших новостных порталов в рунете, а именно «РИА Новости» [10]. График, на котором представлено количество посетителей с 2017 – 2022 года изображен на рисунке 4.



Рисунок 4 – Рост числа посетителей новостного портала РИА Новости

Из данного графика можно сделать вывод, что новостные порталы пользуются высоким спросом и имеют потенциал для дальнейшего роста и привлечения новых пользователей. Из этого можно сделать вывод, что в настоящее время еще существуют возможности для разработки новых новостных порталов и привлечения аудитории. Это открывает перспективы для инноваций и конкурентных решений на рынке новостных сервисов.

Также растет медиасфера, увеличивая тем самым общее качество сервисов по созданию и публикации новостей. На рисунке 5 представлен график по зарегистрированным СМИ в интернете.

Число регистраций СМИ в 1990–2021 годах



действующие и уже недействующие

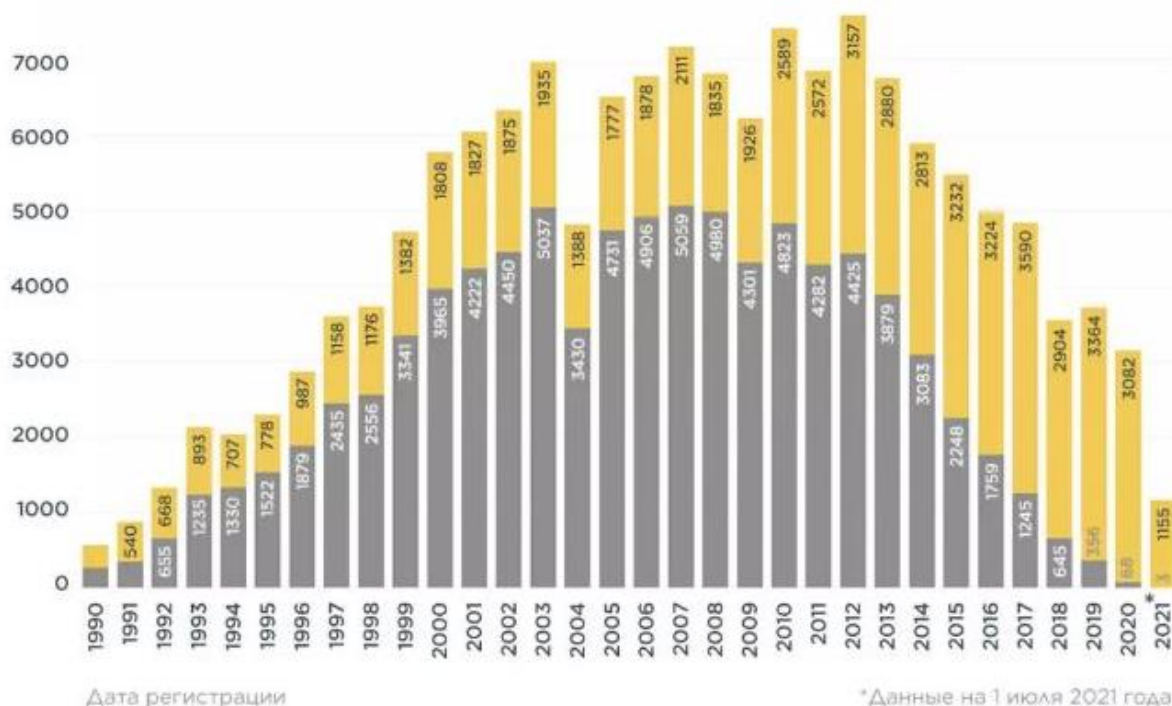


Рисунок 5 – Рост активности СМИ

На середину 2021 года в реестре Роскомнадзора 149 288 средств массовой информации, из них действуют только 61 576 – около 40%. С 1990 года по 2012 год количество новых регистраций СМИ в целом росло – от 572 в 1990-м до 7582 в 2012-м [11].

Из представленного графика становится ясно, что количество зарегистрированных СМИ продолжает расти, однако лишь около 40% из них активно функционируют. Это явление может быть связано с появлением новых и более передовых средств распространения информации, которые представляют серьезную конкуренцию для традиционных источников.

Представленный график на рисунке 6 отражает количество зарегистрированных средств массовой информации в разных типах изданий, включая печатные СМИ, телевидение, сетевые издания и радио.

Число регистраций СМИ в 1990–2020 годах по типам изданий

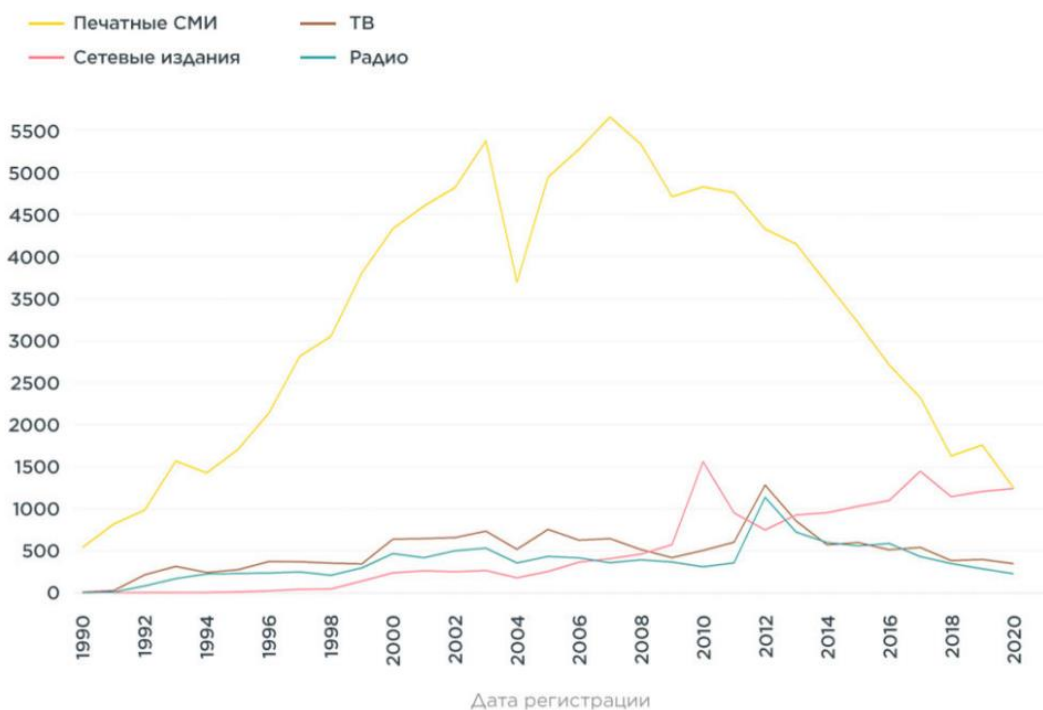


Рисунок 6 – Динамика изменения число регистраций СМИ по типам издания

Из графика можно заключить, что постепенно печатные СМИ теряют свою популярность и актуальность. На смену печатным СМИ приходят сетевые издания, которые уже на момент 2020 года сравнялись по числу регистраций.

Необходимо определить платформу, которая будет поддерживать разрабатываемое приложение. Выбор остановился на мобильном устройстве за счёт его привлекательности в глазах пользователей и текущей тенденции на мировом рынке. Кроме того, мобильные устройства обладают широким спектром возможностей и функций, которые могут быть использованы для улучшения пользовательского опыта и расширения функциональности приложения. Таким образом, разработка приложения для мобильной платформы обеспечит более полный и удовлетворительный опыт для пользователей, встречая их ожидания в сфере мобильных технологий. На рисунке 7 показано соотношение используемых устройств.

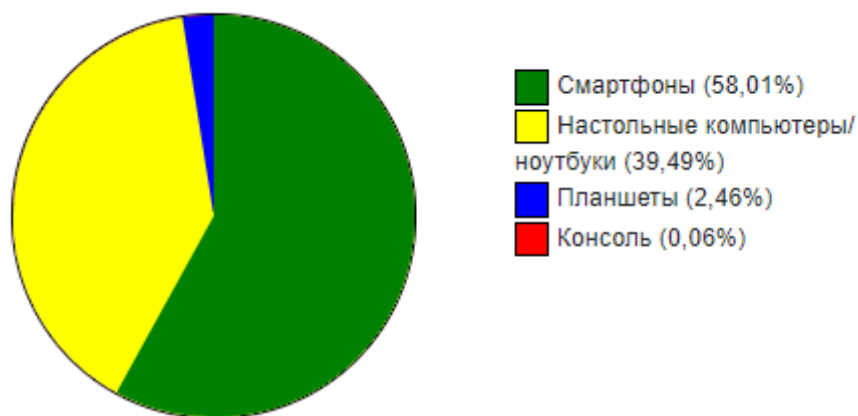


Рисунок 7 – Соотношение устройств на мировом рынке

С середины 2020 года в мире смартфонов используется больше, чем настольных компьютеров [12]. На данный момент в Российской Федерации соотношение устройств в пользу настольных компьютеров и ноутбуков. Доля рынка настольных компьютеров и мобильных устройств, планшетов и консолей в Российской Федерации на февраль 2023 составляет для настольных компьютеров и ноутбуков до 60.73%, мобильных устройств до 37.93%, а остальную часть делят между собой планшеты и консоли [13].

На рисунке 8 изображена диаграмма распределения в процентном соотношении операционных систем на мобильные устройства.



Рисунок 8 – Соотношения операционных систем для мобильных устройств на рынке

К 1 кварталу 2023 года мобильные операционные системы на смартфонах включают доминирующую Android от Google и iOS от Apple, которые в совокупности занимают почти 100% рынка [14].

Существует четкая корреляция между ВВП на душу населения страны и соответствующей долей рынка ОС для смартфонов в разных странах. Пользователи в самых богатых странах чаще выбирают iPhone от Apple, в то время как Android от Google преобладает в других странах [15, 16].

Из-за преимущественного перевеса в сторону Android приложение будет разрабатываться именно под данную операционную систему.

1.4 Анализ функциональных требований

В основе приложения лежит работа с новостями, следовательно, основным объектом будет новостная статья. В системы должны реализованы такие механизмы, которые дадут возможности по созданию, удалению, чтению и оцениваю новостей.

В системе необходимо реализовать средства, которые обеспечат пользователям возможность следить за публикациями друг друга, так называемые подписки, таким образом пользователи будут иметь свою личную аудиторию, которая будет являться мотивацией к созданию более лучших новостей, а также будет косвенно контролировать автора от публикации ложных или неподобающих статей.

В системе будет существовать два действующих лица, оказывающих на неё влияния: пользователь приложения и модератор. Пользователь будет иметь возможности работать с механизмом новостей, профиля и подписок.

Модератор имеет отдельные функции, а именно допуск или отклонение новых новостей для публикации и блокировка пользователей.

В результате анализа требований была сформирована диаграмма вариантов использования, которая представлена на рисунке 9.

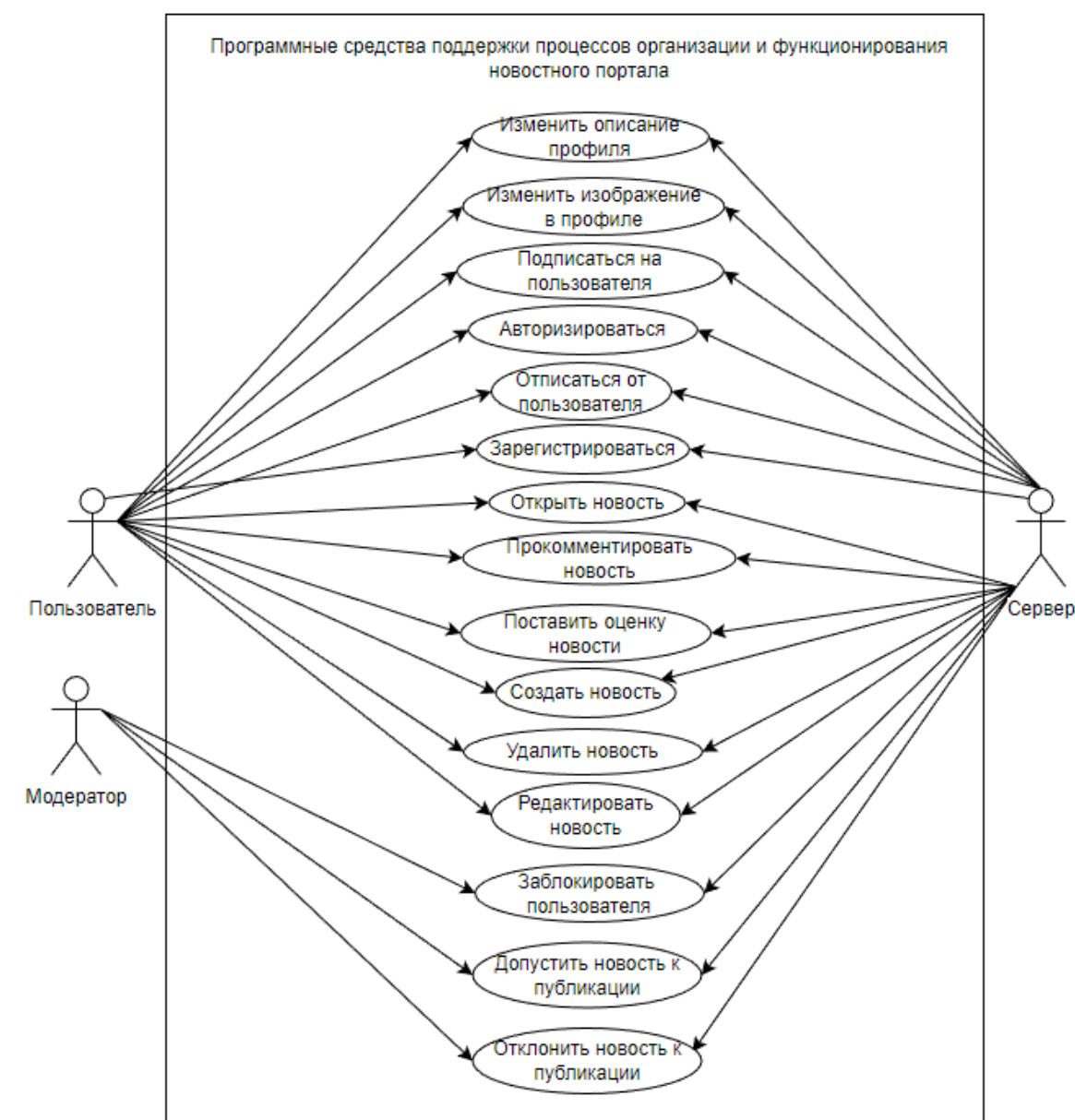


Рисунок 9 – Диаграмма вариантов использования по программным средствам поддержки процессов организации и функционирования новостного портала

Для данной диаграммы вариантов использования приведены спецификации для «Создать новость», «Зарегистрироваться» и «Авторизоваться», чтобы определить необходимые требования к системе. Спецификации помогают уточнить детали реализации, функциональности, интерфейсов и производительности системы. Эти данные могут быть полезны при проектировании, разработке и тестировании системы. На рисунке 10 отображена спецификация варианта использования «Создать новость».

Наименование: создать новость
ID: 1
Краткое описание: данный вариант использования описывает процесс создание новости для новостного портала
Действующие лица: пользователь, сервер
Предусловие: сервер запущен, пользователь авторизован в мобильном приложении
<p>Основной поток:</p> <ol style="list-style-type: none"> 1. Пользователь вводит заголовок новости 2. Мобильное приложение проверяет валидность заголовка 3. Пользователь вводит адрес или выбирает на карте 4. Мобильное приложение проверяет валидность адреса 5. Пользователь добавляет абзац и изображение (по желанию) 6. Пользователь нажимает кнопку «Опубликовать» 7. Мобильное приложение отправляет данные на сервер 8. Сервер проверяет контент новости с помощью нейронной сети 9. Нейронная сеть присваивает контенту степень допустимости 10. Сервер сохраняет данные в базе данных 11. Сервер добавляет новость в очередь для проверки модерацией 12. Сервер возвращает подтверждение о создании новости 13. Мобильное приложение отображает сообщение о создании новости
<p>Альтернативный поток:</p> <ol style="list-style-type: none"> 6а. Если заголовок невалидный, то пользователь возвращается к пункту 1 7а. Если адрес невалидный, то пользователь возвращается к пункту 3
Постусловие: новость успешно создана и добавлена в очередь для проверки модерацией

Рисунок 10 – Спецификация для варианта использования «Создать новость»
в системе новостного портала

Спецификация варианта использования «Зарегистрироваться» представлена на рисунке 11.

Наименование: зарегистрироваться
ID: 2
Краткое описание: данный вариант использования описывает процесс регистрации нового пользователя в системе
Действующие лица: пользователь, сервер
Предусловие: сервер запущен
<p>Основной поток:</p> <ol style="list-style-type: none"> 1. Пользователь открывает мобильное приложение. 2. Пользователь переходит на страницу регистрации. 3. Пользователь вводит псевдоним, логин и пароль. 4. Приложение производит валидацию данных 5. Приложение отправляет данные на сервер для регистрации. 6. Сервер проверяет наличие пользователя с аналогичным логином 7. Сервер хеширует пароль пользователя. 8. Сервер сохраняет данные в базу данных. 9. Сервер отправляет успешный ответ. 10. Приложение переходит на главную страницу.
<p>Альтернативный поток:</p> <ol style="list-style-type: none"> 4а. Если данные не прошли валидацию, приложение выводит сообщение об ошибке и возвращается к пункту 2. 6а. Если пользователь с аналогичным логином уже существует, сервер отправляет сообщение об ошибке и возвращается к пункту 2.
Постусловие: новый пользователь успешно зарегистрирован в системе и может использовать приложение.

Рисунок 11 – Спецификация для варианта использования «Зарегистрироваться» в системе новостного портала

На рисунке 12 изображена спецификация для варианта использования «Авторизоваться».

Наименование: авторизоваться
ID: 3
Краткое описание: пользователь пытается войти в свой зарегистрированный аккаунт в мобильном приложении
Действующие лица: пользователь, сервер
Предусловие: пользователь должен иметь зарегистрированный аккаунт в мобильном приложении
<p>Основной поток:</p> <ol style="list-style-type: none"> 1. Пользователь открывает экран авторизации в мобильном приложении. 2. Пользователь вводит свой логин и пароль. 3. Мобильное приложение проводит валидацию введенных данных. 4. Мобильное приложение отправляет запрос на сервер для проверки соответствия введенных данных зарегистрированному аккаунту. 5. Сервер проверяет соответствие введенных данных с зарегистрированным аккаунтом. 6. Если данные верны, сервер отправляет подтверждение авторизации мобильному приложению. 7. Мобильное приложение авторизует пользователя и перенаправляет его на главный экран приложения.
<p>Альтернативный поток:</p> <ol style="list-style-type: none"> 3а. Если введенные данные не проходят валидацию, то переход к шагу 2. 6а. Если введенные данные не соответствуют ни одному зарегистрированному аккаунту, то переход к шагу 2.
Постусловие: новый пользователь успешно зарегистрирован в системе и может использовать приложение.

Рисунок 12 – Спецификация для варианта использования «Авторизоваться» в системе новостного портала

Для раскрытия прецедента «Авторизироваться» была использована диаграмма деятельности, которая изображена на рисунке 13, чтобы структурировать процесс и понять логику его выполнения. Эта диаграмма помогла определить все необходимые шаги и выявить зависимости между ними.

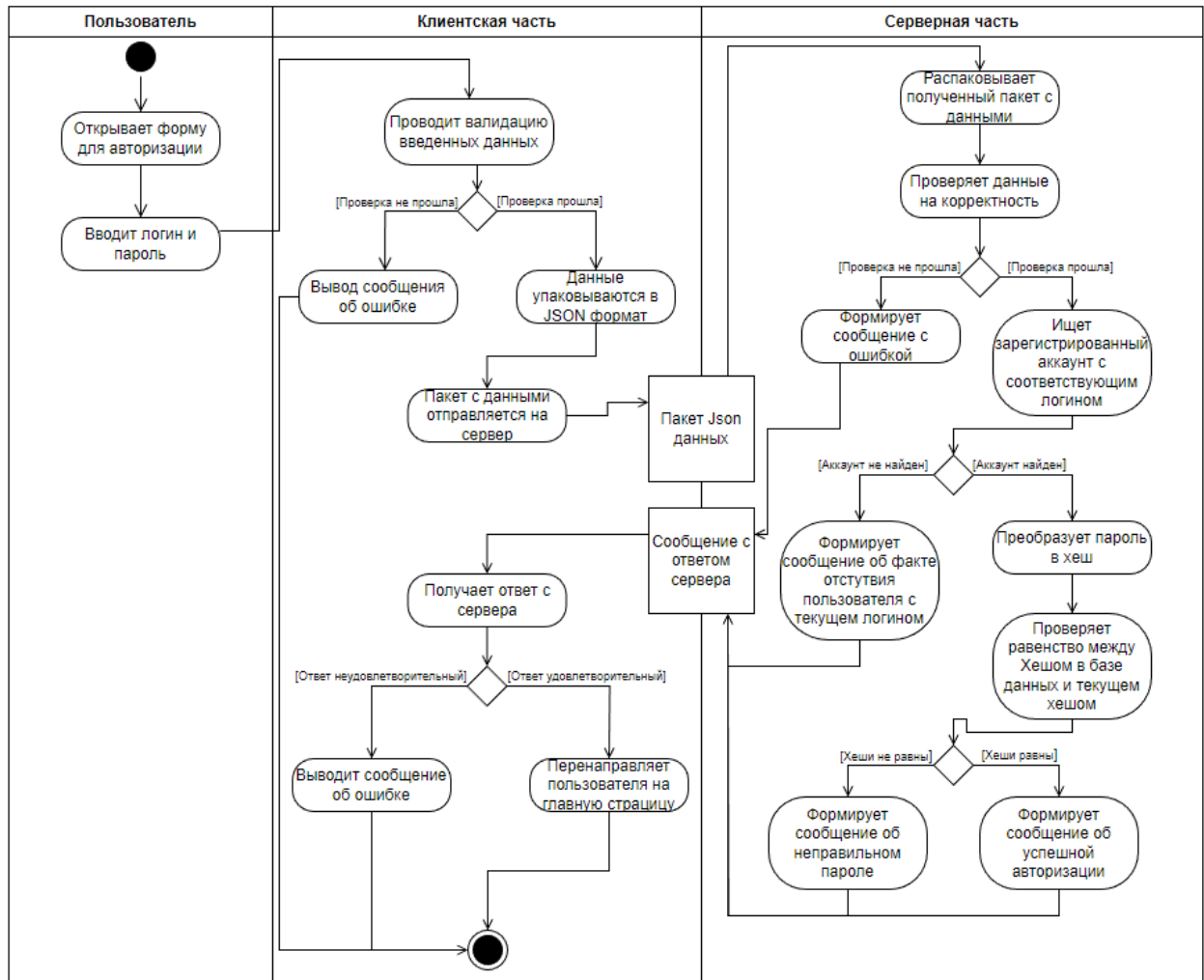


Рисунок 13 – Диаграмма деятельности для прецедента «Авторизироваться»

Диаграмма последовательности поможет упростить создание новости в данной работе, определить этапы работы, сократить время на поиск и устранение ошибок, улучшить взаимодействие между частями системы и продемонстрировать алгоритмы и процессы. Именно поэтому была реализована диаграмма последовательности для создания новости, которая отображена на рисунке 14, чтобы упростить весь процесс и сделать его более понятным и эффективным.

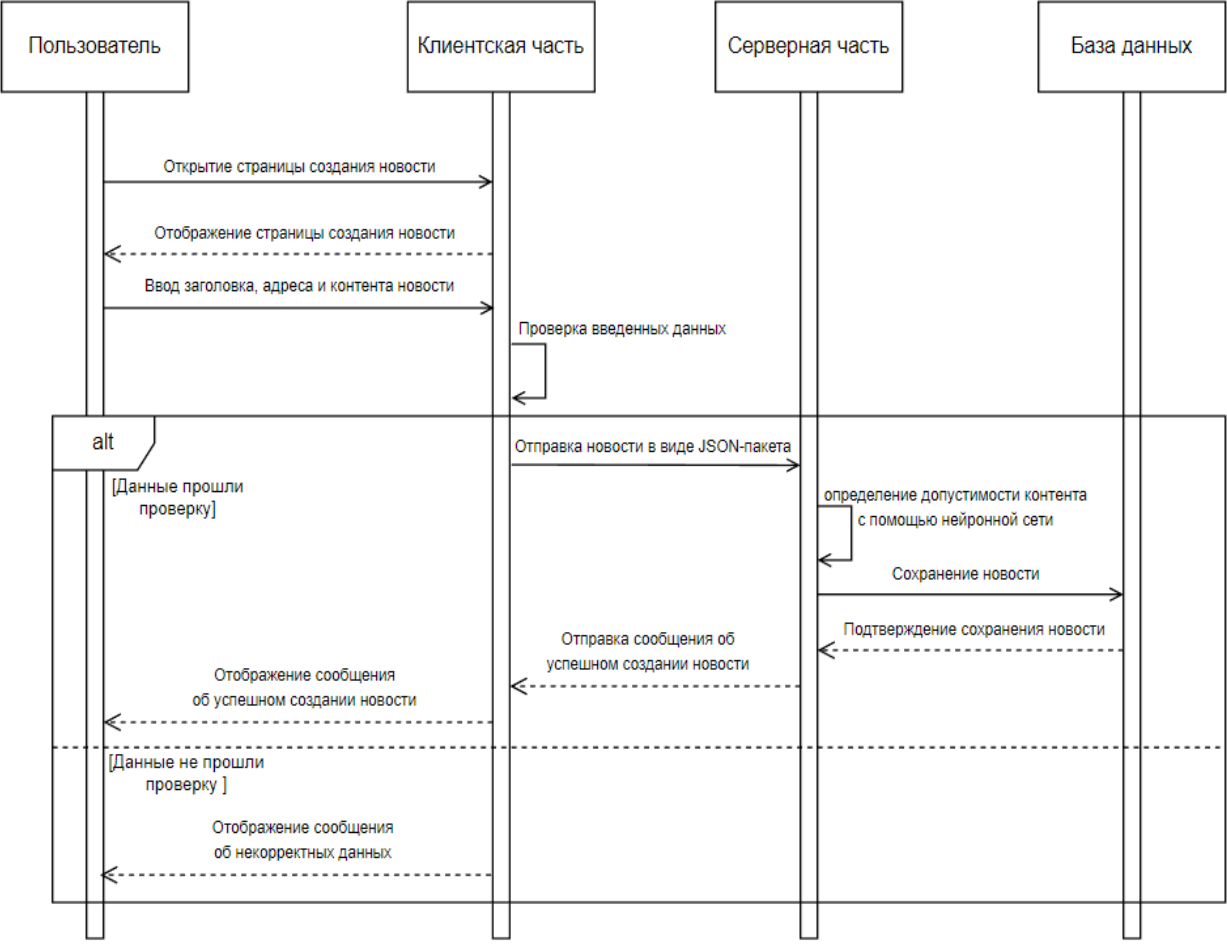


Рисунок 14 – Диаграмма последовательности для прецедента «Создание новости» в системе новостного портала

Путем анализа функциональных требований удалось успешно выявить основные функции, которые должны быть реализованы в разрабатываемом приложении. Для приложения были созданы диаграммы использования, деятельности и последовательности, которые отражают возможные сценарии использования и порядок выполнения операций. Кроме того, для некоторых определенных вариантов использования были составлены подробные технические описания процессов.

1.5 Анализ нефункциональных требований

Необходимо написать серверную часть приложения на языке C#, используя Microsoft Visual Studio 2019 в качестве среды разработки. Для классификации текста будет написана нейронная сеть на языке Python при

использовании библиотека Keras. Данные на сервере будут храниться в базе данных, которая будет создана с помощью СУБД Microsoft SQL Server. Для запросов к базе данных следует использовать язык SQL. Серверная часть должна быть совместима с операционной системой Windows 10 и более поздними версиями.

Клиентскую часть приложения следует написать на языке C# с использованием Xamarin Forms. Необходимо разработать пользовательский интерфейс для клиентской части приложения. Интерфейс должен быть удобным и интуитивно понятным для пользователей. Для этого следует использовать современные технологии и подходы, которые обеспечат удобство использования приложения. Кроме того, следует обратить внимание на привлекательность интерфейса, чтобы он был ненавязчив и соответствовал современным тенденциям в дизайне. Приложение должно быть отлажено и успешно работать на устройствах под управлением ОС Android 6.0 и более новых версий.

1.6 Планирование разработки и оценки бюджета

Составление плана разработки и расчет бюджета проекта выявляет проблемы и трудности, с которыми можно столкнуться в процессе реализации, а выявлении их на ранних этапах позволяет устранить их и реализовать проект успешно. Планирование помогает определить цели и задачи проекта, оценить стоимость и продолжительность проекта, а также определить необходимые ресурсы для достижения нужных целей [17 – 21]. В результате, было решено составить план разработки и рассчитать бюджет проекта, чтобы управлять проектом более эффективно, сократить затраты и риски, а также достичь поставленных целей вовремя и в рамках бюджета.

Этапы, которые необходимо выполнить над проектом приведены в таблице 2. Было выделено 43 взаимосвязанных между собой этапа, которые строго регламентированы по времени.

Таблица 2 – Задачи проекта

№	Наименование задачи	Длительность	Предшественники
1	Начало проекта	-	-
2	Анализ требований	-	-
3	Анализ предметной области	7	1
4	Анализ аналогичных приложений	7	3
5	Анализ целевой аудитории и платформы	5	3
6	Создание технического задания	10	4, 5
7	Планирование проекта	5	6
8	Завершение анализа требований	-	7
9	Выбор средств	-	-
10	Выбор программных средств	5	1
11	Выбор аппаратных средств	3	1
12	Завершение выбора средств	-	10,11
13	Проектирование приложения	-	-
14	Проектирование пользовательского интерфейса	7	8, 12
15	Проектирование базы данных	7	8, 12
16	Проектирование бизнес-логики приложения	14	8, 12
17	Проектирование клиентской части	7	14
18	Проектирование серверной части	7	15, 16
19	Завершение проектирование приложения	-	17, 18
20	Реализация приложения		-
21	Разработка пользовательского интерфейса	15	19
22	Разработка базы данных	10	19
23	Разработка бизнес-логики приложения	25	19
24	Разработка клиентской части	12	21
25	Разработка серверной части	20	22, 23
26	Завершение реализации приложения	-	24,25
27	Тестирование и отладка	-	-
28	Тестирование клиентской части	3	26
29	Тестирование серверной части	5	26
30	Тестирование взаимодействия клиентской и серверной частей	7	28, 29
31	Отладка и устранение ошибок	14	30
32	Завершение тестирования и отладки	-	31
33	Внедрение программного обеспечения	-	-
34	Установка и настройка приложения на сервере	3	32
35	Тестирование приложения после внедрения	7	34
36	Обеспечение безопасности приложения	5	34
37	Документирование и сопровождение приложения	2	34
38	Завершение внедрения ПО	-	35,36,37
39	Презентация результатов работы	-	-
40	Подготовка отчетности по проекту	3	38
41	Проведение презентации проекта	1	40
42	Завершение презентации результатов работы	-	41
43	Конец проекта	-	42

Реализация плана позволит планировать и контролировать выполнение задач в проекте, определить их последовательность и управлять зависимостями. Кроме того, такой подход улучшит взаимодействие между членами команды при работе над проектом, даст возможность им лучше понимать, что от них требуется, и уменьшит риски ошибок и просрочек. Каждая задача имеет свою длительность и предшественников, что позволяет

определить, когда она может быть выполнена и какие задачи должны быть завершены до ее начала. Также можно видеть, какой этап работы в данный момент выполняется и на каком этапе проекта мы находимся в целом.

Ресурсы, которые будут задействованы при работе над проектом, отображены в таблице 3. Это важно, так как планирование и определение необходимых ресурсов помогут эффективно распределить бюджет и управлять ими, чтобы достичь поставленных целей проекта.

Таблица 3 – Ресурсы проекта

Название ресурса	Тип ресурса	Ставка, руб/ч	Сверхурочная ставка, руб/ч	Единоразовые затраты, руб
Дизайнер	Трудовой	800	1200	-
Тестировщик	Трудовой	700	900	-
Разработчик мобильных приложений	Трудовой	1200	1800	-
Разработчик серверной части	Трудовой	1300	1950	-
Full Stack Разработчик	Трудовой	1600	2550	-
Компьютер	Материальный	-	-	1000
Мобильное устройство	Материальный	-	-	500
Хостинг	Затраты	-	-	10000
Домен	Затраты	-	-	2000
Интернет	Затраты	-	-	5000

Определение и распределение ресурсов проекта являются важным этапом планирования и управления проектом. После того, как были определены ресурсы проекта, их можно распределить на конкретные задачи. Таблица 4 содержит соответствующие назначения. Оптимальное распределение ресурсов помогает обеспечить эффективное использование ресурсов и соблюдение сроков выполнения задач. Это позволяет достичь оптимальной производительности и успешного завершения проекта.

Таблица 4 – Назначения ресурсов

№	Наименование задачи	Ресурс	Используемых ресурсов
1	Начало проекта	-	-
2	Анализ требований	-	-
3	Анализ предметной области	Full Stack Разработчик, Компьютер, Интернет	40%, 1 шт.
4	Анализ аналогичных приложений	Full Stack Разработчик, Дизайнер, Компьютер, Интернет	50%, 50%, 1 шт.

Продолжение таблицы 4

№	Наименование задачи	Ресурс	Используемых ресурсов
5	Анализ целевой аудитории и платформы	Full Stack Разработчик, Дизайнер, Компьютер, Интернет	50%, 50%, 1 шт.
6	Создание технического задания	Full Stack Разработчик, Дизайнер, Компьютер, Интернет	75%, 25%, 1 шт.
7	Планирование проекта	Разработчик мобильных приложений, Разработчик серверной части, Full Stack Разработчик, Дизайнер, Компьютер, Интернет	75%, 75%, 25%, 25%, 1 шт.
8	Завершение анализа требований	-	-
9	Выбор средств	-	-
14	Проектирование пользовательского интерфейса	Разработчик мобильных приложений, Дизайнер, Компьютер, Интернет	75%, 75%, 2 шт.
15	Проектирование базы данных	Разработчик серверной части, Full Stack Разработчик, Компьютер, Интернет	50%, 50%, 2 шт.
16	Проектирование бизнес-логики приложения	Разработчик серверной части, Разработчик мобильных приложений, Компьютер, Интернет	50%, 50%, 1 шт.
17	Проектирование клиентской части	Разработчик мобильных приложений, Full Stack Разработчик, Компьютер, Интернет	75%, 75%, 2 шт.
18	Проектирование серверной части	Разработчик серверной части, Full Stack Разработчик, Компьютер, Интернет	75%, 75%, 2 шт.
19	Завершение проектирование приложения	-	-
20	Реализация приложения	-	-
21	Разработка пользовательского интерфейса	Разработчик мобильных приложений, Full Stack Разработчик, Дизайнер, Компьютер, Мобильное устройство, Интернет	70%, 50%, 25%, 3 шт. , 2 шт.
22	Разработка базы данных	Разработчик серверной части, Full Stack Разработчик, Компьютер, Интернет, Домен, Хостинг	70%, 50%, 2 шт.
23	Разработка бизнес-логики приложения	Разработчик серверной части, Разработчик мобильных приложений, Компьютер, Интернет, Домен, Хостинг	50%, 50%, 2 шт.
24	Разработка клиентской части	Разработчик мобильных приложений, Full Stack Разработчик, Компьютер, Мобильное устройство, Интернет	50%, 50%, 2 шт, 2 шт.
25	Разработка серверной части	Разработчик серверной части, Full Stack Разработчик, Компьютер, Интернет, Домен, Хостинг	100%, 25%, 2 шт
26	Завершение реализации приложения	-	-
27	Тестирование и отладка	-	-
28	Тестирование клиентской части	Тестировщик, Full Stack Разработчик, Дизайнер, Компьютер, Мобильное устройство, Интернет,	50%, 50%, 75%, 1 шт. , 1 шт.
29	Тестирование серверной части	Тестировщик, Full Stack Разработчик, Компьютер, Интернет, Домен, Хостинг	100%, 50%, 1 шт.
30	Тестирование взаимодействия клиентской и серверной частей	Тестировщик, Full Stack Разработчик, Компьютер, Мобильное устройство, Интернет, Домен, Хостинг	50% , 50% , 1 шт., 1 шт.
31	Отладка и устранение ошибок	Тестировщик, Разработчик мобильных приложений, Разработчик серверной части, Full Stack Разработчик, Компьютер, Мобильное устройство, Интернет, Домен, Хостинг	25%, 25%, 25%, 25%, 4 шт. , 2 шт.

Окончание таблицы 4

№	Наименование задачи	Ресурс	Используемых ресурсов
32	Завершение тестирования и отладки	-	
33	Внедрение программного обеспечения	-	50%, 1 шт., 1 шт.
35	Тестирование приложения после внедрения	Тестировщик, Дизайнер, Компьютер, Мобильное устройство, Интернет, Домен, Хостинг	50%, 50%, 1 шт., 1 шт.
36	Обеспечение безопасности приложения	Full Stack Разработчик, Разработчик мобильных приложений, Разработчик серверной части, Компьютер, Мобильное устройство, Интернет, Домен, хостинг	25%, 25%, 25%, 3 шт., 2 шт.
37	Документирование и сопровождение приложения	Тестировщик, Full Stack Разработчик, Компьютер, Интернет	50%, 50%, 2 шт
38	Завершение внедрения ПО	-	-
39	Презентация результатов работы	-	-
40	Подготовка отчетности по проекту	Тестировщик, Full Stack Разработчик, Дизайнер, Компьютер, Интернет	20%, 20%, 40%, 3 шт.
41	Проведение презентации проекта	Дизайнер, Full Stack Разработчик, Компьютер, Интернет	50%, 50%, 1 шт.
42	Завершение презентации результатов работы	-	-
43	Конец проекта	-	-

Для реализации проекта нам необходимо нанять команду из Full Stack Разработчика, Разработчика серверной части, Разработчика мобильных приложений, Дизайнера и Тестировщика. Для успешной работы этой команде необходимы компьютеры, мобильные устройства, хостинг, домен и доступ в Интернет. Был построен сетевой график, который отображен на рисунке 15, чтобы оценить продолжительность выполнения проекта и его критический путь. Использование данного графика также позволяет выявить уязвимости и потенциальные проблемы при планировании, что позволяет предвидеть и минимизировать возможные негативные последствия и обеспечивать более гладкое протекание проекта.

Кроме того, важно учитывать, что в ходе выполнения проекта могут возникнуть изменения и новые требования, которые потребуют дополнительных ресурсов или времени. Постоянный мониторинг сетевого графика позволит адаптироваться к изменениям и принимать соответствующие меры для успешной реализации проекта.

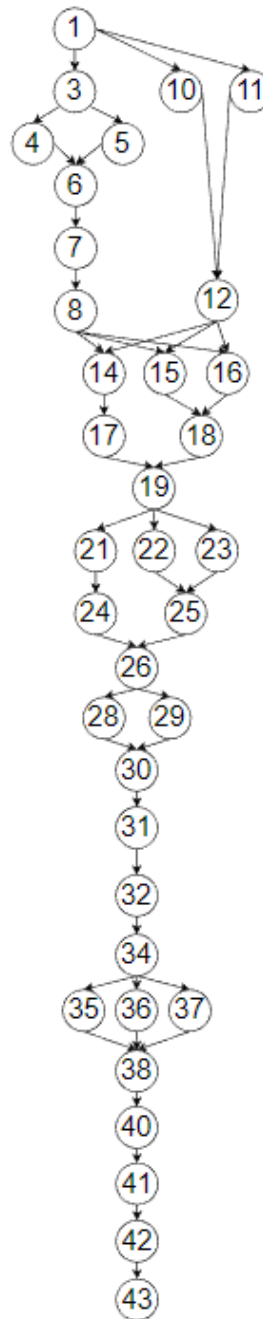


Рисунок 15 – Сетевой график проекта

Для того, чтобы определить наиболее критические задачи проекта, мы проведем ручной расчет критического пути. Формулы для расчета времени раннего (ES) и позднего (LS) начала задач в проекте:

- параметр ES = Максимальный (ES предшествующей задачи) + продолжительность предшествующей задачи;
- параметр LS = Минимальный (LS последующей задачи) – продолжительность текущей задачи.

В таблице 5 содержится информация о ранних сроках начала работ для каждой задачи проекта.

Таблица 5 – Расчеты раннего времени начала работ

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
1	-	-	-	-	0
3	1	0	0	0+0	0
4	3	0	7	0+7	7
5	3	0	7	0+7	7
6	4	7	7	Max(7+7;7+5)	14
	5	7	5		
7	6	14	10	14+10	24
8	7	24	5	24+5	29
10	1	0	0	0+0	0
11	1	0	0	0+0	0
12	10	0	5	Max(0+5;0+3)	5
	11	0	3		
14	8	29	0	Max(29+0;5+0)	29
	12	5	0		
15	8	29	0	Max(29+0;5+0)	29
	12	5	0		
16	8	29	0	Max(29+0;5+0)	29
	12	5	0		
17	14	29	7	29+7	36
18	15	29	7	Max(29+7;29+14)	43
	16	29	14		
19	17	36	7	Max(36+7;43+7)	50
	18	43	7		
21	19	50	0	50+0	50
22	19	50	0	50+0	50
23	19	50	0	50+0	50
24	21	50	15	50+15	65
25	22	50	10	Max(50+10; 50+25)	75
	23	50	25		
26	24	65	12	Max(65+12;75+20)	95
	25	75	20		
28	26	95	0	95+0	95
29	26	95	0	95+0	95

Продолжение таблицы 5

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
30	28	95	3	Max(95+3; 95+5)	100
	29	95	5		
31	30	100	7	100+7	107
32	31	107	14	107+14	121
34	32	121	0	121+0	121
35	34	121	3	121+3	124
36	34	121	3	121+3	124
37	34	121	3	121+3	124
38	35	124	7	Max(124+7;124+5;124+2)	131
	36	124	5		
	37	124	2		
40	38	131	0	131+0	131
41	40	131	3	131+3	134
42	41	134	1	134+1	135
43	42	135	0	135+0	135

В таблице 6 рассчитано позднее время начала работ задач.

Таблица 6 – Расчеты позднего времени начала работ

Работа	Последователи			Вычисление	Позднее время начала
	№	Позднее время начала	Длительность		
43	-	-	-	-	135
42	43	135	0	135-0	135
41	42	135	1	135-1	134
40	41	134	3	134-3	131
38	40	131	0	131-0	131
37	38	131	2	131-2	129
36	38	131	5	131-5	126
35	38	131	7	131-7	124
34	35	124	3	Min(124-3;126-3;129-3)	121
	36	126			
	37	129			
32	34	121	0	121-0	121
31	32	121	14	121-14	107

Продолжение таблицы 6

Работа	Последователи			Вычисление	Позднее время начала
	№	Позднее время начала	Длительность		
30	31	107	7	107-7	100
29	30	100	5	100-5	95
28	30	100	3	100-3	97
26	28	97	0	Min(97-0;95-0)	95
	29	95			
25	26	95	20	95-20	75
24	26	95	12	95-12	83
23	25	75	25	75-25	50
22	25	75	10	75-10	65
21	24	83	15	83-15	68
19	21	68	0	Min(68-0;65-0;50-0)	50
	22	65			
	23	50			
18	19	50	7	50-7	43
17	19	50	7	50-7	43
16	18	43	14	43-14	29
15	18	43	7	43-7	36
14	17	43	7	43-7	36
12	14	36	0	Min(36-0;36-0;29-0)	29
	15	36			
	16	29			
11	12	29	3	29-3	26
10	12	29	5	29-5	24
8	14	36	0	Min(36-0;36-0;29-0)	29
	15	36			
	16	29			
7	8	29	5	29-5	24
6	7	24	10	24-10	14
5	6	14	5	14-5	9
4	6	14	7	14-7	7
3	4	7	7	Min(7-7;9-7)	0
	5	9			
1	3	0	0	Min(0-0;24-0;26-0)	0
	10	24			
	11	26			

На основе найденных значений раннего и позднего времени начала работ можно найти критический путь. Соответствующие вычисления представлены в таблице 7. Анализ критического пути помогает определить наиболее важные этапы проекта, которым необходимо уделить особое внимание и контроль, чтобы обеспечить своевременное выполнение проекта и достижение его целей. Это позволяет минимизировать задержки и повышать эффективность всего проекта в целом.

Таблица 7 – Вычисление критического пути

Работа №	Раннее время начала	Позднее время начала	Резерв времени
1	0	0	0
3	0	0	0
4	7	7	0
5	7	9	2
6	14	14	0
7	24	24	0
8	29	29	0
10	0	24	24
11	0	26	26
12	5	29	24
14	29	36	7
15	29	36	7
16	29	29	0
17	36	43	7
18	43	43	0
19	50	50	0
21	50	68	18
22	50	65	15
23	50	50	0
24	65	83	18
25	75	75	0

Продолжение таблицы 7

Работа №	Раннее время начала	Позднее время начала	Резерв времени
26	95	95	0
28	95	97	2
29	95	95	0
32	121	121	0
34	121	121	0
35	124	124	0
36	124	126	2
37	124	129	5
38	131	131	0
40	131	131	0
41	134	134	0
42	135	135	0
43	135	135	0

На основе полученных данных можно сделать вывод, что критический путь проходит через работы с номерами 1, 3, 4, 6, 7, 8, 16, 18, 19, 23, 25, 26, 29, 30, 31, 32, 34, 35, 38, 40, 41, 42, 43, а общее время выполнения проекта составляет 135 дней. На рисунке 16 представлен граф проекта, на котором штриховыми линиями указан критический путь. Анализ критического пути позволяет идентифицировать ключевые этапы проекта, требующие особого внимания и управления, с целью минимизации задержек и обеспечения своевременного завершения проекта. Это помогает оптимизировать ресурсы, улучшить планирование и снизить риски. Важно отметить, что ручной анализ критического пути, хотя и мощное средство в планировании, становится менее надежным с увеличением объема задач, что увеличивает вероятность ошибок. Необходимо применять специализированные инструменты и программы, автоматизирующие анализ критического пути, для повышения точности и эффективности планирования проекта.

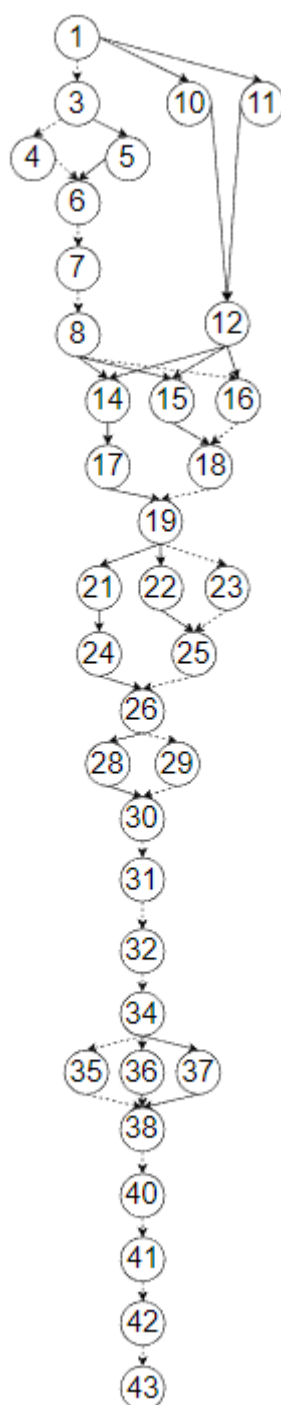


Рисунок 16 – Критический путь проекта

Для проверки точности расчетов длительности проекта и определения его бюджета будет использован инструмент управления проектами от Microsoft – Microsoft Project. Эта система может автоматически произвести все необходимые расчеты и получить точные данные о проекте. Использование специализированных инструментов, таких как Microsoft Project, позволяет значительно повысить эффективность управления

проектами и уменьшить риски ошибок в расчетах. Таким образом, для управления проектом рекомендуется использовать Microsoft Project или аналогичные инструменты, чтобы грамотно провести расчеты и минимизировать риски возможных ошибок [22 – 25].

На рисунке 17 показан список задач из таблицы 2, который был добавлен в программу Microsoft Project. Начальной датой проекта было выбрано 07.12.2022.

	Название задачи	Длительность	Начало	Окончание	Предш	Названия ресурсов
1	Начало проекта	0 дней	Ср 07.12.22	Ср 07.12.22		
2	Анализ требований	29 дней	Чт 08.12.22	Вт 17.01.23		
3	Анализ предметной области	7 дней	Чт 08.12.22	Пт 16.12.22	1	Full Stack Разработчик[40%];Компьютер[1];Инте
4	Анализ аналогичных приложений	7 дней	Пн 19.12.22	Вт 27.12.22	3	Full Stack Разработчик[50%];Дизайнер[50%];Комп
5	Анализ целевой аудитории и платформы	5 дней	Пн 19.12.22	Пт 23.12.22	3	Full Stack Разработчик[50%];Дизайнер[50%];Комп
6	Создание технического задания	10 дней	Ср 28.12.22	Вт 10.01.23	4;5	Full Stack Разработчик[75%];Дизайнер[25%];Комп
7	Планирование проекта	5 дней	Пн 11.01.23	Вт 17.01.23	6	Разработчик мобильных приложений[75%];Разр
8	Завершение анализа требований	0 дней	Вт 17.01.23	Вт 17.01.23	7	
9	Выбор средств	5 дней	Чт 08.12.22	Ср 14.12.22		
10	Выбор программных средств	5 дней	Чт 08.12.22	Ср 14.12.22	1	Разработчик мобильных приложений[25%];Разр
11	Выбор аппаратных средств	3 дней	Чт 08.12.22	Пн 12.12.22	1	Full Stack Разработчик[50%];Компьютер[1];Инте
12	Завершение выбора средств	0 дней	Ср 14.12.22	Ср 14.12.22	10;11	
13	Проектирование приложения	21 дней	Ср 18.01.23	Ср 15.02.23		
14	Проектирование пользовательского интерфе	7 дней	Ср 18.01.23	Чт 26.01.23	8;12	Разработчик мобильных приложений[75%];Диза
15	Проектирование базы данных	7 дней	Ср 18.01.23	Чт 26.01.23	8;12	Разработчик серверной части[50%];Full Stack Pa
16	Проектирование бизнес-логики приложения	14 дней	Ср 18.01.23	Пн 06.02.23	8;12	Разработчик серверной части[50%];Разработчи
17	Проектирование клиентской части	7 дней	Пт 27.01.23	Пн 06.02.23	14	Разработчик мобильных приложений[75%];Full S
18	Проектирование серверной части	7 дней	Вт 07.02.23	Ср 15.02.23	16;15	Разработчик серверной части[75%];Full Stack Pa
19	Завершение проектирования приложения	0 дней	Ср 15.02.23	Ср 15.02.23	17;18	
20	Реализация приложения	45 дней	Чт 16.02.23	Ср 19.04.23		
21	Разработка пользовательского интерфейса	15 дней	Чт 16.02.23	Ср 08.03.23	19	Разработчик мобильных приложений[70%];Full S
22	Разработка базы данных	10 дней	Чт 16.02.23	Ср 01.03.23	19	Разработчик серверной части[70%];Full Stack Pa
23	Разработка бизнес-логики приложения	25 дней	Чт 16.02.23	Ср 22.03.23	19	Разработчик серверной части[50%];Разработчи
24	Разработка клиентской части	12 дней	Чт 09.03.23	Пт 24.03.23	21	Full Stack Разработчик[50%];Компьютер[2];Моби
25	Разработка серверной части	20 дней	Чт 23.03.23	Ср 19.04.23	22;23	Разработчик серверной части;Full Stack Разрабо
26	Завершение реализации приложения	0 дней	Ср 19.04.23	Ср 19.04.23	24;25	
27	Тестирование и отладка	26 дней	Чт 20.04.23	Чт 25.05.23		
28	Тестирование клиентской части	3 дней	Чт 20.04.23	Пн 24.04.23	26	Тестирующий[50%];Full Stack Разработчик[50%];Д
29	Тестирование серверной части	5 дней	Чт 20.04.23	Ср 26.04.23	26	Тестирующий;Full Stack Разработчик[50%];Комп
30	Тестирование взаимодействия клиентской и	7 дней	Чт 27.04.23	Пт 05.05.23	28;29	Тестирующий[50%];Full Stack Разработчик[50%];Д
31	Отладка и устранение ошибок	14 дней	Пн 08.05.23	Чт 25.05.23	30	Тестирующий[25%];Разработчик мобильных при
32	Завершение тестирования и отладки	0 дней	Чт 25.05.23	Чт 25.05.23	31	
33	Внедрение программного обеспечения	10 дней	Пт 26.05.23	Чт 08.06.23		
34	Установка и настройка приложения на сервере	3 дня	Пт 26.05.23	Вт 30.05.23	32	Full Stack Разработчик[50%];Компьютер[1];Моби
35	Тестирование приложения после внедрения	7 дней	Ср 31.05.23	Чт 08.06.23	34	Тестирующий[50%];Дизайнер[50%];Компьютер[1
36	Обеспечение безопасности приложения	5 дней	Ср 31.05.23	Вт 06.06.23	34	Full Stack Разработчик[25%];Разработчик мобил
37	Документирование и сопровождение прилож	2 дня	Ср 31.05.23	Чт 01.06.23	34	Тестирующий[50%];Full Stack Разработчик[50%];Д
38	Завершение внедрения ПО	0 дней	Чт 08.06.23	Чт 08.06.23	35;36;3	
39	Презентация результатов работы	4 дня	Пт 09.06.23	Ср 14.06.23		
40	Подготовка отчетности по проекту	3 дня	Пт 09.06.23	Вт 13.06.23	38	Тестирующий[20%];Full Stack Разработчик[20%];Д
41	Проведение презентации проекта	1 день	Ср 14.06.23	Ср 14.06.23	40	Дизайнер[50%];Full Stack Разработчик[50%];Комп
42	Завершение презентации результатов работ	0 дней	Ср 14.06.23	Ср 14.06.23	41	
43	Конец проекта	0 дней	Ср 14.06.23	Ср 14.06.23	42	

Рисунок 17 – Задачи проекта в среде Microsoft Project

С использованием Microsoft Project была создана диаграмма Ганта на основе введенных данных. Из нее становится ясно, что критический путь проходит через задачи с номерами 1, 3, 4, 6, 7, 8, 16, 18, 19, 23, 25, 26, 29, 30, 31, 32, 34, 35, 38, 40, 41 и 42. Интересно отметить, что программа Microsoft Project вывела тот же критический путь, что и ручное вычисление. Данная

диаграмма Ганта представлена на рисунке 18. Диаграмма Ганта позволяет визуально представить распределение задач и зависимостей между ними, а также управлять временными ресурсами проекта. Она облегчает планирование, контроль и коммуникацию в команде, помогая лучше организовать работы, оптимизировать расписание и увидеть прогресс проекта на протяжении его выполнения.



Рисунок 18 – Диаграмма Ганта

Трудовые ресурсы «Full Stack Разработчик», «Разработчик мобильных приложений», «Разработчик серверной части» являются основными в процессе реализации проекта и требуют оптимизации и эффективного использования загрузки для успешного выполнения задач. Максимальная эффективность использования ресурсов способствует более эффективной реализации проекта и достижению поставленных целей. Ниже на рисунке 19 отображен график загрузки ресурса «Full Stack Разработчик».

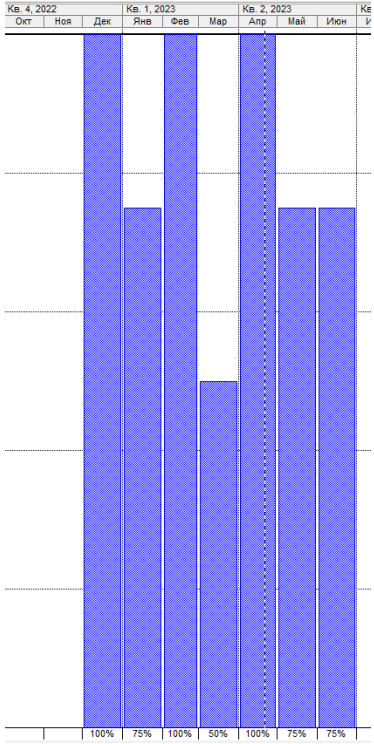


Рисунок 19 – График загрузки ресурса «Full Stack Разработчик»

На рисунке 20 представлен график загрузки ресурса «Разработчик мобильных приложений».

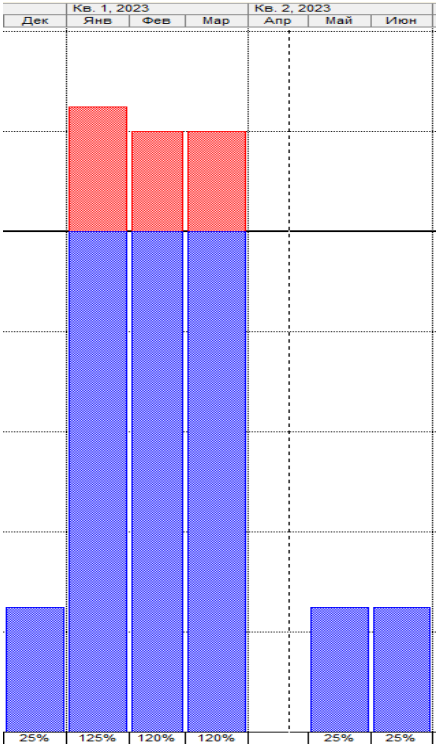


Рисунок 20 – График загрузки ресурса «Разработчик мобильных приложений»

График загрузки ресурса «Разработчик серверной части» представлен на рисунке 21.

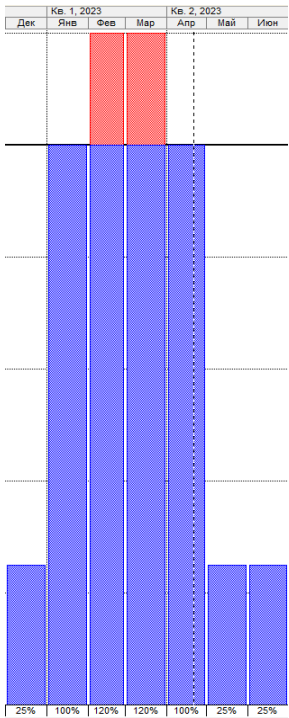


Рисунок 21 – График загрузки ресурса «Разработчик серверной части»

На рисунках 22 – 23 представлены графики трудовых ресурсов «Дизайнер» и «Тестировщик».

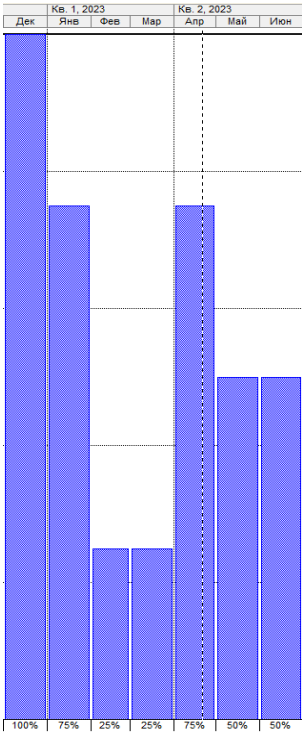


Рисунок 22 – График загрузки ресурса «Дизайнер»

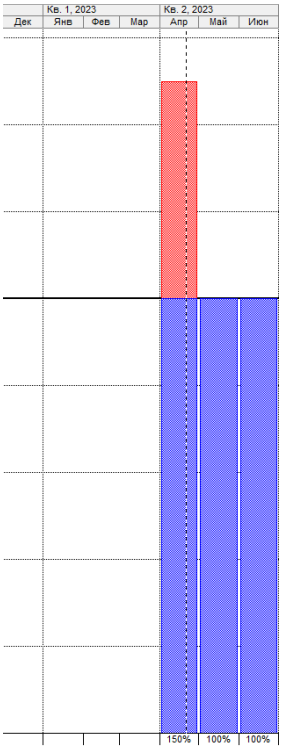


Рисунок 23 – График загрузки ресурса «Тестировщик»

На рисунке 24 представлен график материального ресурса «Компьютер».

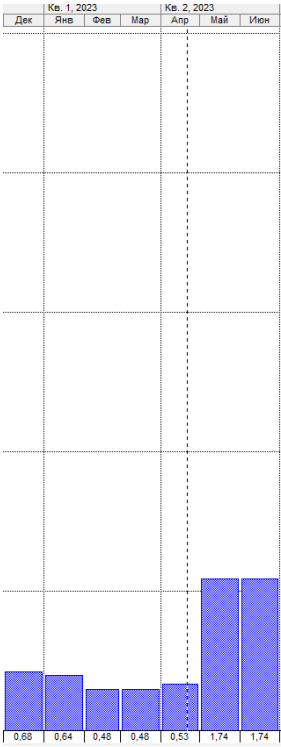


Рисунок 24 – График загрузки ресурса «Компьютер»

новостной портал
SPeCiAlIST RePack

на дату Пт 21.04.23

Даты			
Начало:	Ср 07.12.22	Окончание:	Ср 14.06.23
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней
Длительность			
Текущий план:	135 дней	Оставшаяся:	135 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	135 дней	Процент завершения:	0%
Трудозатраты			
Текущий план:	1 762,93 ч	Оставшиеся:	1 762,93 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	1 762,93 ч	Процент завершения:	0%
Затраты			
Текущий план:	2 402 360,00 Р	Оставшиеся:	2 402 360,00 Р
Базовый план:	0,00 Р	Фактические:	0,00 Р
Отклонение:	2 402 360,00 Р		
Состояние задачи		Состояние ресурса	
Еще не начатые задачи	43	Трудовые ресурсы:	6
Выполняющиеся задачи	0	Трудовые ресурсы с прев. дост	3
Завершенные задачи:	0	Материальные ресурсы:	2
Итого задач:	43	Итого ресурса:	11

Рисунок 26 – Обзорный отчет проекта

На основании проведенной оценки затрат и временных рамок проекта можно сделать вывод о его экономической целесообразности. Анализ показывает, что проект имеет высокий потенциал для получения положительного экономического результата. Прогнозируемые доходы превышают затраты, что указывает на возможность достижения прибыли. Кроме того, установленные временные рамки позволяют своевременно внедрить продукт на рынок и начать генерировать доходы в кратчайшие сроки. Таким образом, проект обладает значительным потенциалом для финансового успеха и может быть рекомендован для дальнейшей реализации.

2 Проектирование мобильного приложения

2.1 Выбор технологий и инструментальных средств разработки

В данной работе было решено использовать методологию RUP. Она представляет собой итеративную и инкрементальную методологию, основанную на объектно-ориентированном подходе и разработанную компанией Rational Software Corporation [26, 27]. Методология включает в себя четыре основных фазы разработки программного обеспечения:

- фаза начала работы – в ходе которой происходит сбор и анализ требований, определяются ресурсы и риски проекта, а также разрабатывается концепция проекта;
- фаза разработки – в ходе которой разрабатывается подробный план проекта, определяются архитектура, дизайн и прототипы программного продукта;
- фаза конструкции – в ходе которой происходит активная разработка и тестирование программного продукта, а также его интеграция и оптимизация;
- фаза трансформации – в ходе которой выполняется тестирование, документирование и передача программного продукта в эксплуатацию.

Каждая из этих фаз реализует определенные задачи, которые выполняются последовательно в процессе разработки программного обеспечения. RUP также даёт средства по управлению проектами и рисками, а также ресурсами и коммуникациями в рамках команды разработчиков.

Выбор методологии RUP был обусловлен тем, что данная методология разработки программного обеспечения является гибкой и адаптивной, что позволяет эффективно управлять сложными и масштабными проектами. Кроме того, еще одним фактором, который повлиял на выбор RUP, является то, что данная методология широко использует UML диаграммы, что облегчает процесс описания виденья проекта между разработчиками. В отличие от более других методологий, RUP предлагает поэтапный подход к

разработке, что позволяет получать обратную связь от заказчика и вносить изменения в проект в процессе его разработки. Методология RUP также предоставляет средства для управления проектом на каждом отдельном этапе.

Чтобы проанализировать требования и разработать структуру проекта, соблюдая принципы RUP, было решено применять диаграммы UML [28]. Для построения диаграмм использовалось средство UML-моделирования Draw.io.

При планировании проекта было решено использовать Microsoft Project для эффективного управления всеми стадиями его выполнения. Это приложение позволяет структурировать проект, составить его календарный план и рассчитать затраты труда, материальных и финансовых ресурсов в процессе работы над проектом.

Для создания мобильного приложения было решено использовать Xamarin Forms в среде программирования Visual Studio, которая предназначена для разработки программ для операционной системы Android и IOS, которая использует язык программирования C#. Серверная часть также будет реализована на языке C# с использованием среды разработки Visual Studio 2019. Был выбран язык Python при использовании библиотеки Keras для разработки нейронной сети. Этот выбор обусловлен мощными возможностями языка Python в области машинного обучения и широкой поддержкой библиотеки Keras, что обеспечивает эффективное создание и обучение нейронных сетей. Была выбрана СУБД Microsoft SQL Server для данного проекта, так как она обладает высокой производительностью и надежностью, обеспечивает удобный интерфейс для администрирования базы данных, а также имеет мощные инструменты для разработки и оптимизации запросов. Кроме того, Microsoft SQL Server имеет хорошую совместимость с другими продуктами Microsoft, что облегчает интеграцию с существующей системой. Проектирование базы данных выполнялось с помощью приложения Visual Paradigm, а реализация базы данных осуществлялась с использованием языка программирования SQL.

2.2 Проектирование архитектуры программного обеспечения

Для данного приложения было принято решение использовать двухуровневую архитектуру «клиент-сервер». Схема представлена на рисунке 27.



Рисунок 27 – Двухуровневая архитектура «клиент–сервер»

В этой архитектуре база данных и бизнес логика приложения располагаются на сервере и взаимодействуют с клиентом через интерфейс API. Клиентская часть, в свою очередь, предоставляет удобный интерфейс для пользователей, в котором можно просматривать и взаимодействовать с данными, а также удаленно обращаться к серверу.

Взаимодействие между клиентом и сервером происходит посредством HTTP-запросов, при этом сервер создает SQL-запросы для обработки информации в базе данных. Одним из преимуществ такой архитектуры является то, что все данные хранятся на сервере, что позволяет сократить размер клиентского приложения. Минусом данного подхода является выделения отдельного устройства под сервер, который должен быть весьма производителен и защищен от различных цифровых атак.

Использовании данной архитектуры даёт возможность реализовать клиент на любом устройстве. Это делает приложение более гибким и доступным для широкой аудитории, что может привести к увеличению числа пользователей. Кроме того, использование множества клиентов для одной и

той же серверной части повышает производительность приложения, так как различные клиенты оптимизированы для разных платформ и устройств. Это улучшает взаимодействие между приложением и пользователем и увеличивает удовлетворенность клиентов.

2.3 Проектирование интерфейса пользователя

Проектирование интерфейса играет ключевую роль в разработке мобильного приложения. Это важный этап, который позволяет создать удобную и привлекательную пользовательскую среду, соответствующую функциональным требованиям. Интерфейс мобильного приложения является связующим звеном между пользователем и функциональностью приложения, определяя его удобство использования, эстетическое восприятие и эффективность. При проектировании интерфейса уделяется внимание таким аспектам, как навигация, визуальное оформление и удобство взаимодействия.

На основе выявленных функциональных требований был разработан интерфейс мобильного приложения. Он включает в себя страницы регистрации, входа, карты с новостями, новостью, создания новой новости и профилем.

Страница регистрации представлена на рисунке 28. На ней имеются следующие элементы:

- 1 – поле для ввода псевдонима;
- 2 – поле для ввода логина;
- 3 – поле для ввода пароля;
- 4 – кнопка для регистрации;
- 5 – кнопка для открытия страницы с авторизацией.

При активации функции регистрации происходит двухэтапная проверка данных: сначала на стороне клиента, затем на стороне сервера. В случае ошибки на одной из сторон пользователю выводится соответствующее сообщение с ошибкой.

The diagram shows a registration form titled "Регистрация". It contains five elements labeled with numbers 1 through 5:

- 1: Points to the first input field.
- 2: Points to the second input field.
- 3: Points to the third input field.
- 4: Points to the button labeled "Зарегистрировать".
- 5: Points to the button labeled "Авторизироваться".

Рисунок 28 – Страница регистрации

Страница авторизации представлена на рисунке 29. На ней имеются следующие элементы:

- 1 – поле для ввода логина;
- 2 – поле для ввода пароля;
- 3 – кнопка для авторизации;
- 4 – кнопка для открытия страницы с регистрацией.

При использовании функции авторизации происходит двухэтапная проверка данных: вначале на стороне клиента, а затем на стороне сервера. Клиентская проверка выполняется для того, чтобы гарантировать правильность заполнения всех полей, в то время как серверная проверка определяет, существует ли пользователь с такими данными в системе и не заблокирован ли он в ней. Если происходит ошибка или пользователь заблокирован, ему выводится соответствующее сообщение.

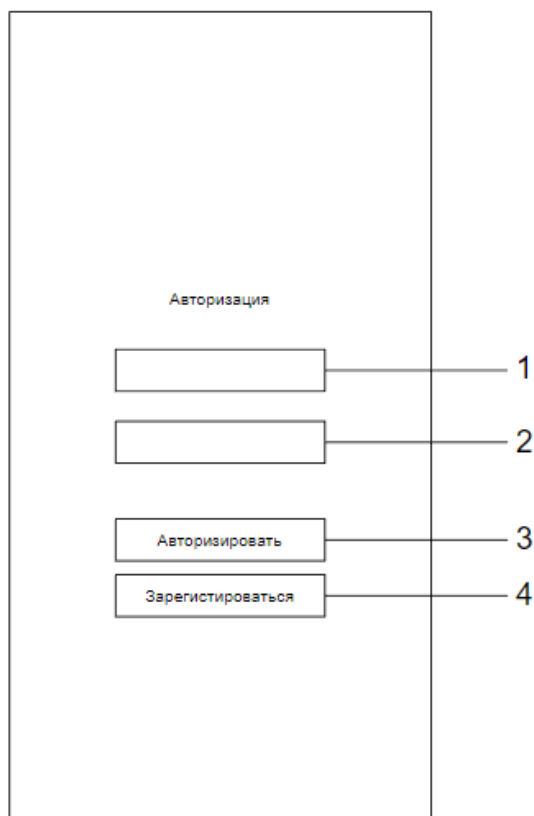


Рисунок 29 – Страница авторизации

Страница карты с новостями представлена на рисунке 30. На ней имеются следующие элементы:

- 1 – кнопка для закрытия меню;
- 2 – кнопка для перехода на страницу профиля;
- 3 – кнопка для открытия карты с новостями;
- 4 – кнопка для открытия формы по созданию новой новости;
- 5 – кнопка для выхода из аккаунта;
- 6 – панель меню с кнопками;
- 7 – карта мира;
- 8 – метки с новостями на карте.

При выборе опции из меню, происходит автоматическое закрытие меню и переход на соответствующую страницу. При клике на метку на карте, пользователь будет перенаправлен на страницу, содержащую соответствующую новость.

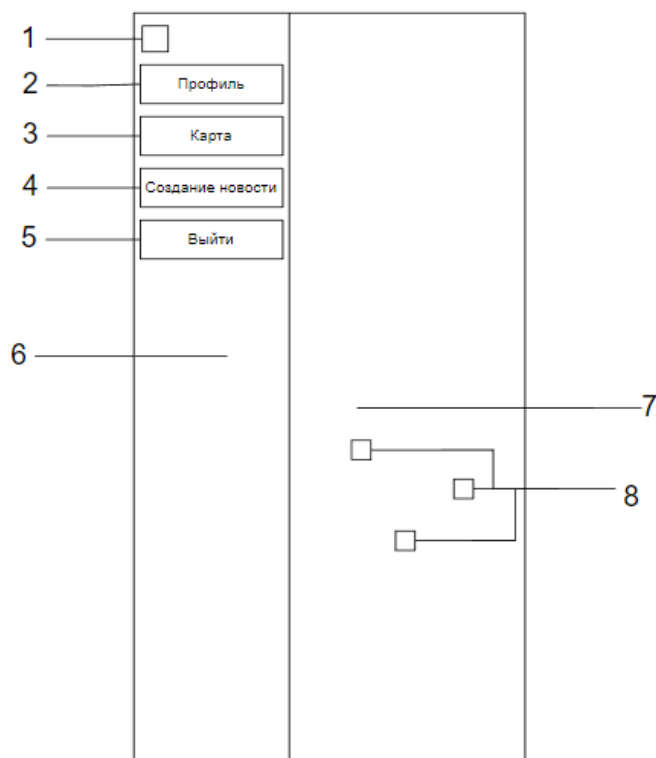


Рисунок 30 – Страница карты с новостями

Страница с новостью представлена на рисунке 31. На ней имеются следующие элементы:

- 1 – изображение профиля автора новости;
- 2 – кнопка для подписки или отписки от данного автора;
- 3 – область с содержимым новости;
- 4 – область для вывода даты новости;
- 5 – область для вывода заголовка новости;
- 6 – область для вывода адреса новости;
- 7 – область для вывода абзацев с текстом и изображениями новости;
- 8 – кнопка для добавления оценки «Нравится» к данной новости;
- 9 – кнопка для добавления оценки «Не нравится» к данной новости;
- 10 – поле для ввода комментария к данной новости;
- 11 – кнопка для отправки комментария;
- 12 – изображения пользователя, оставившего комментарий;
- 13 – текст комментария, оставленный пользователем.

При клике на кнопку «Подписаться» или «Отписаться», её функциональность изменяется на противоположную. При нажатии на изображение пользователя система автоматически перенаправляет на страницу с профилем соответствующего пользователя. Кнопки для оценки контента настроены таким образом, что пользователь может поставить только одну оценку определенного типа.

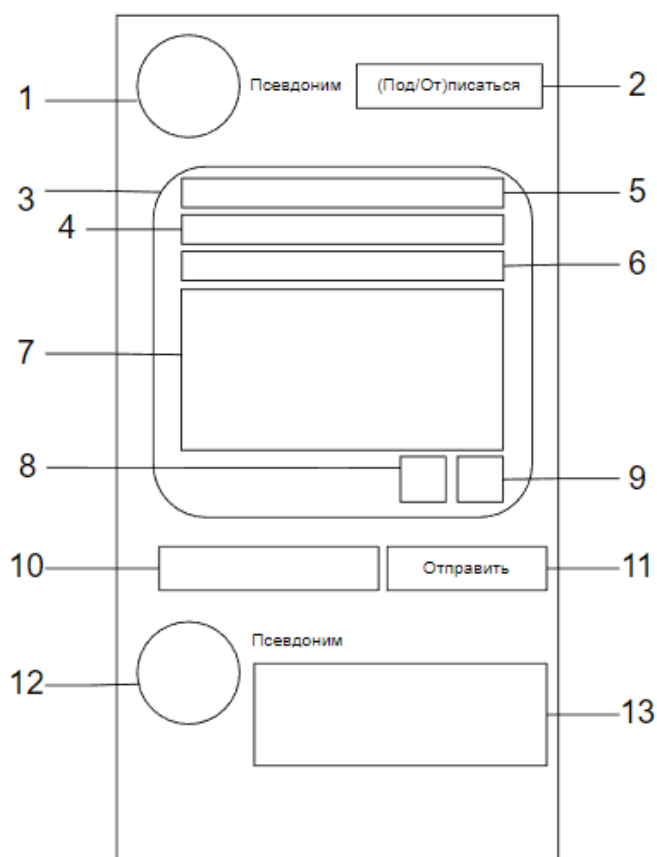


Рисунок 31 – Страница с новостью

Страница с созданием новости представлена на рисунке 32. На ней имеются следующие элементы:

- 1 – кнопка для сохранения новости в системе;
- 2 – поле для ввода заголовка новости;
- 3 – поле для ввода адреса новости;
- 4 – кнопка, которая открывает карту, на которой можно выбрать адрес новости;
- 5 – поле для ввода текста абзаца новости;

- 6 – область с отображением добавленной фотографии;
- 7 – кнопка для удаления изображения, которое находится над кнопкой;
- 8 – изменение изображения, которое находится над кнопкой;
- 9 – кнопка для добавления нового изображения в новость.

При клике на кнопку «Создать новость» данные упаковываются и отправляются на сервер, ответ от сервера приходит позже. При нажатии кнопки для добавления изображения открывается локальное хранилище телефона, где пользователь может выбрать нужное изображение. После добавления каждого изображения создается новое поле для ввода текста. Если изображение удалено, абзацы, которые окружают данное изображение, объединяются в один абзац с соответствующим содержанием.

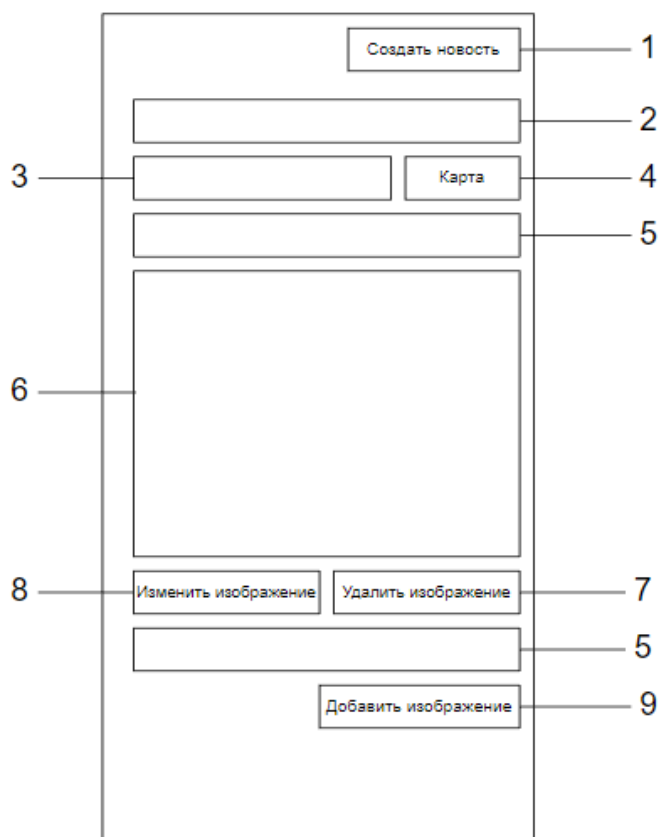


Рисунок 32 – Страница с созданием новости

Страница с профилем представлена на рисунке 33. На ней имеются следующие элементы:

- 1 – изображение профиля;
- 2 – отображение количества подписчиков данного пользователя;

- 3 – отображение количества новостей данного пользователя;
- 4 – кнопка для открытия описания профиля;
- 5 – кнопка для открытия созданных новостей данным пользователем;
- 6 – кнопка для открытия списка подписок данного пользователя;
- 7 – отображение данных, которые зависят от нажатой кнопки.

При клике на изображения в профиле пользователя открывается контекстное меню, позволяющее изменить или удалить изображение в профиле. При нажатии кнопки «Новости» отображается список новостей, каждая из которых может быть отдельно отредактирована или удалена. При нажатии кнопки «Подписки» открывается список, где пользователь может управлять своими подписками и перейти на профили соответствующих пользователей.

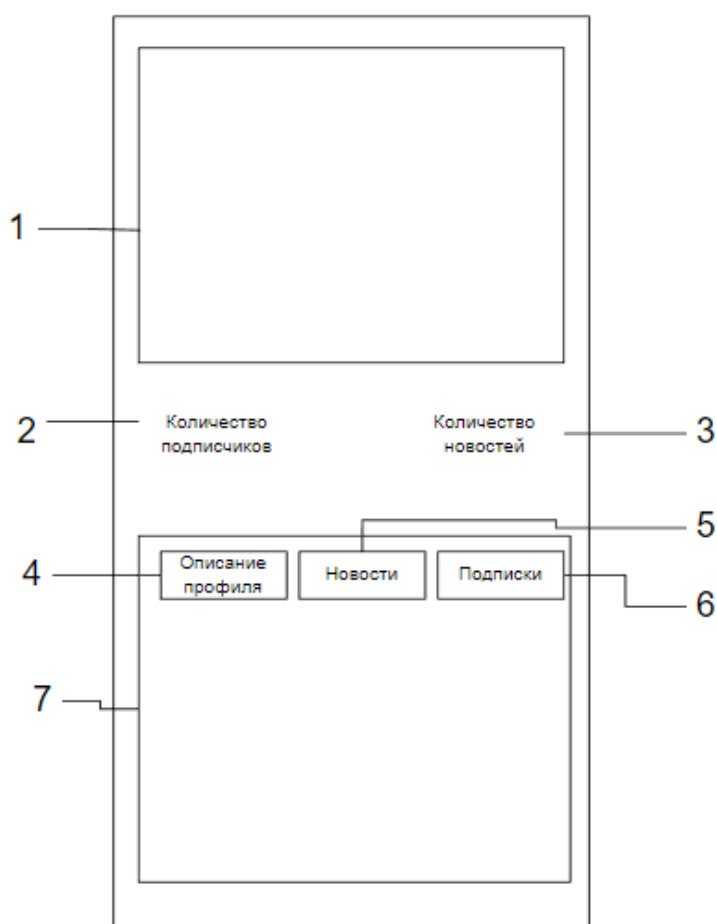


Рисунок 33 – Страница с профилем

Таким образом, был разработан пользовательский интерфейс, который является важным элементом взаимодействия между пользователем и системой. Он обеспечивает удобство использования приложения и повышает пользовательскую эффективность.

2.4 Логическое проектирование базы данных

После проведения анализа предметной области и описания функциональных требований была построена концептуальная модель базы данных [29]. Данная модель представлена в виде диаграммы «сущность–связь» на рисунке 34, что обеспечивает наглядное представление взаимосвязей между сущностями и помогает в дальнейшем проектировании и реализации базы данных.

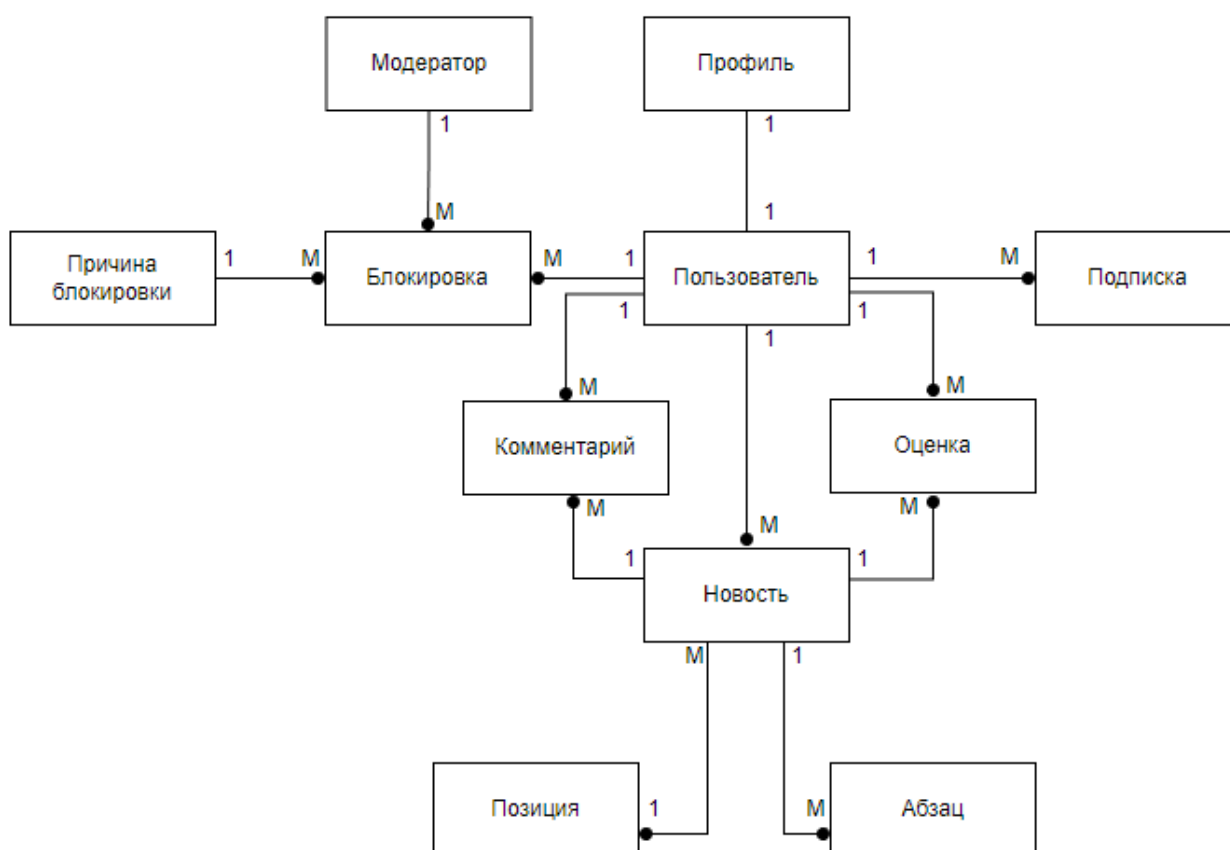


Рисунок 34 – Диаграмма «сущность-связь»

Сущности концептуальной модели данных описаны в таблице 8.

Таблица 8 – Сущности предметной области

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
1	Пользователь	Список пользователей	Стержневая	5000000	В систему добавляются новые пользователи после процедуры регистрации
2	Подписки	Список подписок на пользователей	Стержневая	10000000	Добавляются, когда один пользователь подписывается на другого, удаляются, когда пользователь отписывается от другого пользователя.
3	Причины блокировки	Список причин блокировки пользователей	Стержневая	100	Обновляются редко, добавление обычно происходит с появлением новой возможности, которую необходимо контролировать
4	Модератор	Список модераторов	Стержневая	1000	Обновляется при добавлении новых модераторов. Сами модераторы могут создавать новые записи.
5	Блокировка	Список временных блокировок аккаунтов	Стержневая	10000	Модераторы блокируют пользователей, которые нарушили правила приложения
6	Профиль	Список профилей пользователей	Стержневая	5000000	Добавляется в соответствии с пользователем, отдельно не может существовать.
7	Новость	Список созданных новостей пользователями	Стержневая	1000000	Каждый пользователь может создать свою новость.
8	Позиция	Список позиций в списке	Стержневая	100000000	Добавляется новая позиция при добавлении новости с адресом, которого нет в системе. Позиция проверяется сервисами и модераторами.

Продолжение таблицы 8

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
9	Абзац	Список фрагментов текста и изображений частей новостей	Стержневая	50000000	Добавляется при создании новости путем её разбиения на части. Отдельно абзац добавить невозможно без определенной новости.
10	Комментарий	Список комментариев к новостям	Стержневая	10000000	Каждый пользователь может оставить комментарий к любой новости.
11	Оценка	Список оценок новостей	Стержневая	50000000	Каждый пользователь может поставить свою оценку любой новости.

В таблице 9 представлено описание связей между сущностями предметной области. Описание связей между сущностями поможет лучше понять взаимосвязь между элементами предметной области и определить ключевые факторы, влияющие на работу системы или процесса.

Таблица 9 – Описание связей между сущностями предметной области

№	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность	Ограничения
1	Пользователь	Подписка	Каждый пользователь может участвовать в множестве подписок/ Каждая подписка производится строго между двумя пользователями	1:M	Необязательная (пользователь может не участвовать в подписках)	Нельзя иметь несколько подписок на одного и того же пользователя
2	Пользователь	Оценка	Каждый пользователь может поставить любое количество оценок / Каждая оценка ставится только одним пользователем	1:M	Необязательная (пользователь может не ставить оценки новостям)	На одну новость пользователь может поставить только одну оценку

Продолжение таблицы 9

№	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность	Ограничения
3	Пользователь	Новость	Пользователь может создать любое количество новостей/Каждая новость может быть создана только одним пользователем	1:M	Необязательная (пользователь может не создавать новости)	
4	Пользователь	Комментарий	Пользователь может оставить любое количество комментариев/Каждый комментарий может быть оставлен только одним пользователем	1:M	Необязательная (пользователь может не оставлять комментарии под новостями)	
5	Пользователь	Блокировка	Пользователь может иметь любое количество блокировок/Каждая блокировка относится только к одному пользователю	1:M	Необязательная (пользователь может не иметь блокировок)	
6	Пользователь	Профиль	Каждый пользователь имеет один профиль/Каждый профиль относится лишь к одному пользователю	1:1	Обязательная	
7	Блокировка	Причина блокировки	Каждая блокировка имеет только одну причину блокировки/Каждая причина блокировки может участвовать во множестве блокировок	1:M	Обязательная	

Окончание таблицы 9

№	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность	Ограничения
11	Новость	Позиция	Каждая новость имеет лишь одну позицию/Каждая позиция может присутствовать в любой новости	M:1	Обязательная	
12	Новость	Абзац	Каждая новость может состоять из любого количества абзацев/Каждый абзац относится к одной новости	1:M	Обязательная	

На основании концептуальной модели данных в среде Visual Paradigm была построена логическая модель базы данных. Ее структура представлена на рисунке 35.

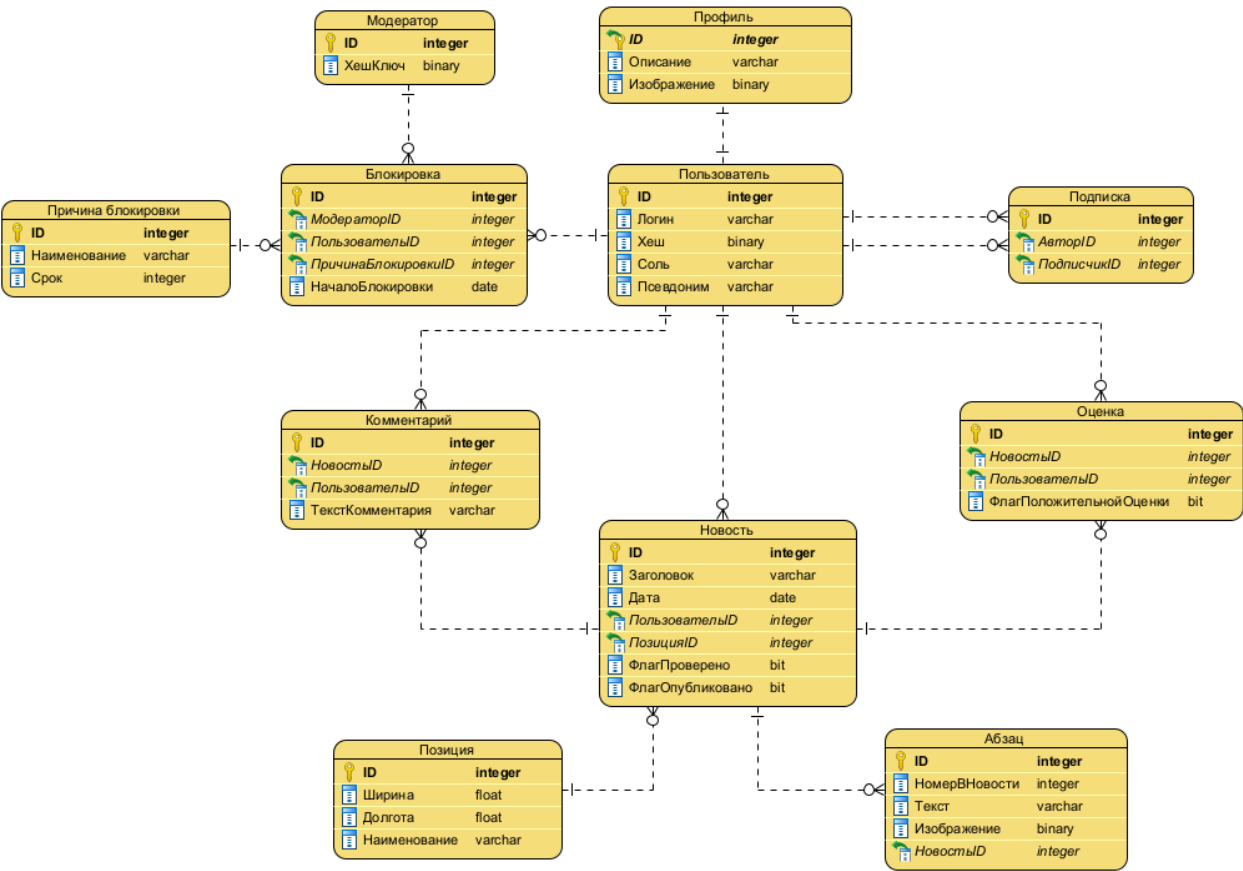


Рисунок 35 – Логическая модель базы данных

В данной логической модели представлена более подробная структура базы данных, включающая типы сущностей, связи между ними и конкретные атрибуты. Важно отметить, что модель разработана без учета конкретной реализации базы данных. В процессе ее разработки сущности были представлены в абстрактной форме, чтобы обеспечить гибкость и масштабируемость в будущем.

2.5 Проектирование серверной части приложения

Протокол HTTP с его стандартными методами (GET, POST, PUT, DELETE) используется для доступа к данным серверной части приложения, которая представляет собой REST API. Доступ к ресурсам осуществляется посредством URI-идентификаторов.

Архитектура приложения основана на шаблоне проектирования MVC (Model–View–Controller), который представлен на рисунке 36.

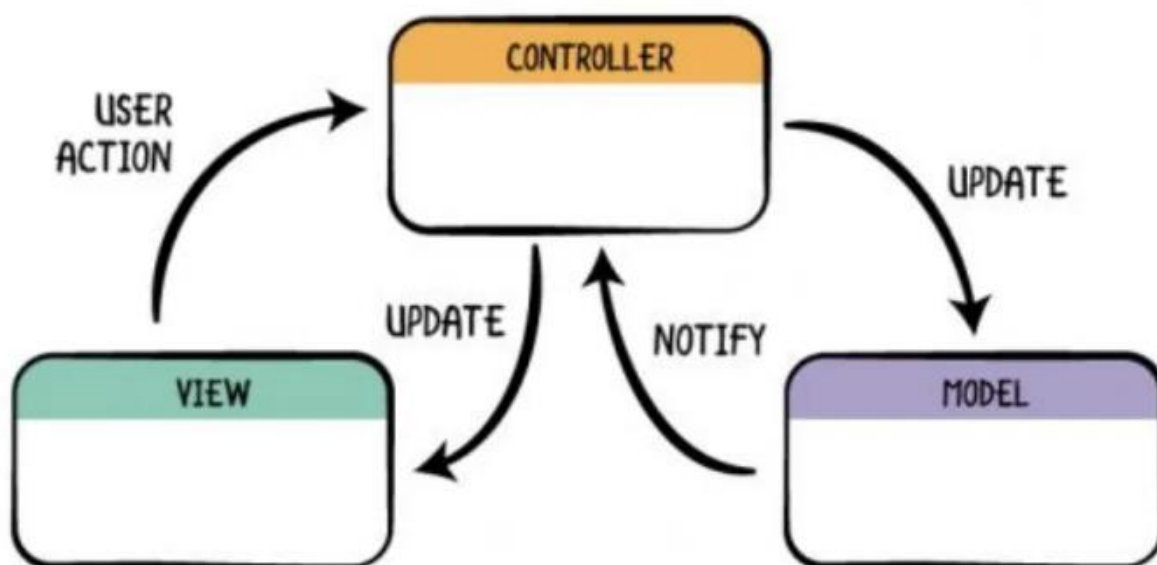


Рисунок 36 – Архитектура MVC

В приложении присутствует модель, которая содержит информацию из базы данных. Также имеется блок контроллеров, который реализует основную логику при получении запросов от клиента. Они получают и обрабатывают HTTP-запросы и возвращают ответы. Пользователь же

взаимодействует только с уровнем представления, который отвечает за интерфейс и обработку пользовательского ввода.

2.6 Проектирование нейронной сети

В помощь модератору для публикации новостей было решено применить нейронную сеть, которая будет отличать допустимый текст от недопустимого.

В проекте искусственный интеллект должен будет распознавать недопустимый текст, под такую классификацию будет попадать текст, который содержит пропаганду наркотиков, нецензурную лексику, экстремистские призывы и другое ненормативное содержание.

На сегодняшний день существует множество архитектур нейронных сетей для этой задачи, которые показывают высокую эффективность. Для новостного портала была выбрана простая полносвязная нейронная сеть, которая состоит преимущественно из входного слоя, скрытых слоев и выходного слоя [30]. Структура нейронной сети изображена на рисунке 37.

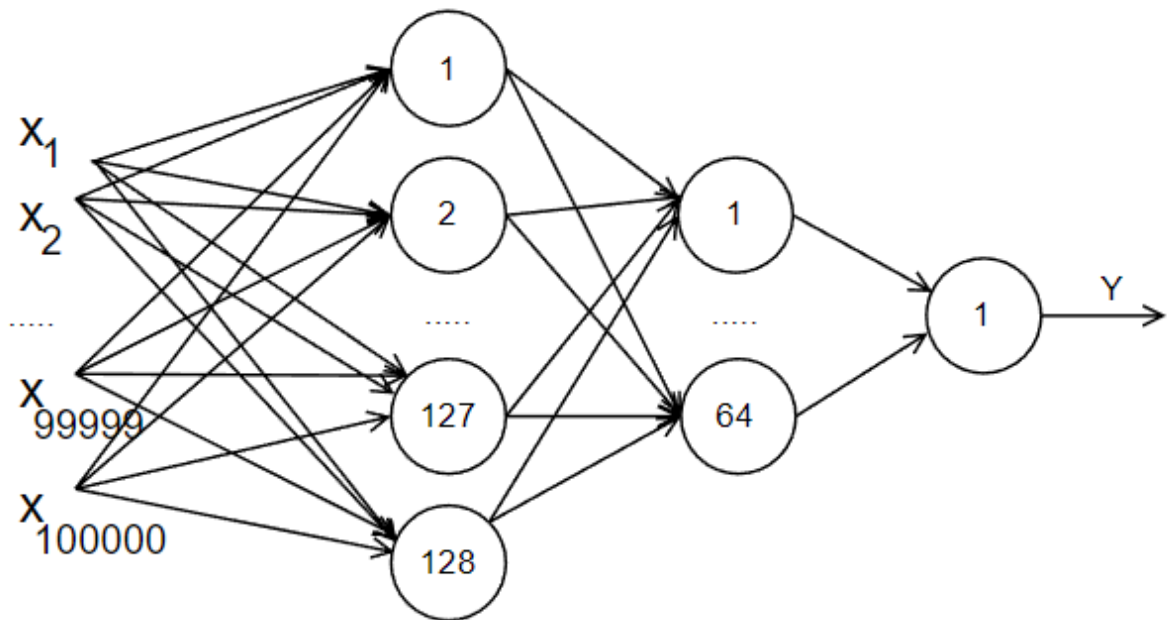


Рисунок 37 – Структура нейронной сети для классификации текста

Данная нейронная сеть имеет один входной слой из вектора самых встречающихся слов в русском языке, которые были ограничены в количестве

100000 штук, имеет два скрытых слоя, которые используют функцию активации ReLu и выходной слой, состоящий из одного нейрона с сигмоидальной функцией активации.

Такая структура позволит быстро обучить нейронную сеть, а также должна показать весьма высокие показатели эффективности.

Для обучения будет подготовлен набор данных, включающий 15 000 различных текстов, предварительно размеченных как допустимые и недопустимые. Такая выборка позволит эффективно обучить нейронную сеть, предотвращая переобучение.

Несмотря на то, что в настоящее время существуют более прогрессивные нейронные сети, такие как RNN или CNN, для данной задачи классификации была выбрана полносвязная нейронная сеть, так как она является наиболее подходящей для обработки текстов. Кроме того, ее разработка и обучение являются относительно простыми, что делает ее привлекательным вариантом для решения данной задачи.

3 Реализация мобильного приложения

3.1 Реализация базы данных

С использованием Visual Paradigm была создана физическая модель базы данных, основанная на предварительно составленной логической модели, учитывающей характеристики СУБД Microsoft SQL Server. Структура этой модели изображена на рисунке 38. Это обеспечивает наглядное представление организации данных и их взаимосвязей.

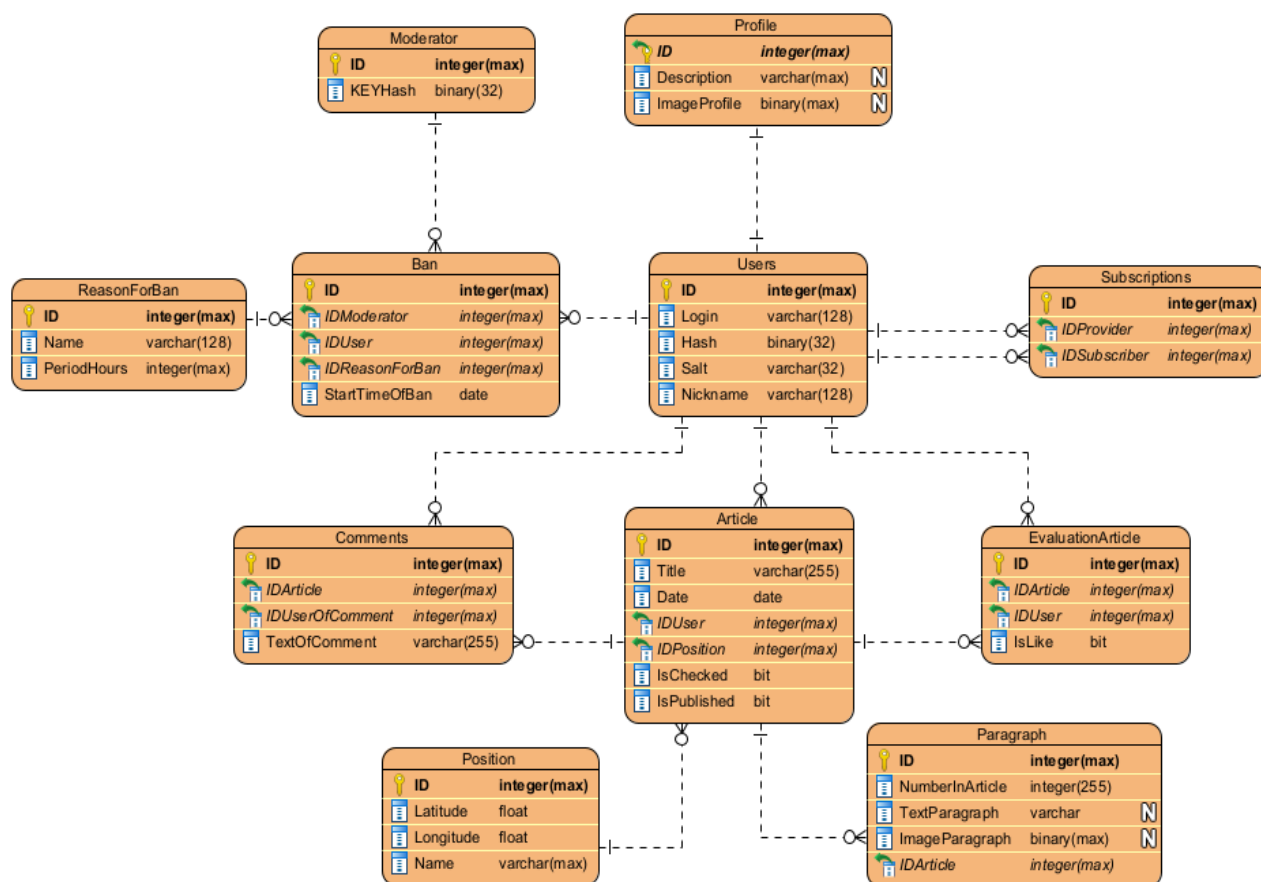


Рисунок 38 – Физическая модель базы данных

Используя сгенерированный на основе физической модели базы данных скрипт на языке SQL, было выполнено развертывание базы данных [31]. Код этого скрипта можно найти в приложении А. Для администрирования баз данных, работающих на СУБД Microsoft SQL Server, была использована утилита SQL Server Management Studio, путем запуска данного скрипта.

3.2 Разработка серверной части

В рамках разработки клиент-серверного мобильного приложения была реализована серверная часть с использованием Microsoft Visual Studio 2019 на языке C# и технологии Web API [32, 33]. Для взаимодействия с базой данных была использована технология Entity Framework [34]. Для обеспечения безопасности передачи данных между клиентом и сервером был использован протокол HTTPS. Диаграмма классов серверной части представлена на рисунке 37.

Так как ранее была выбрана модель MVC, серверная часть приложения разделена на два основных слоя: модель и контроллер. Слой модели взаимодействует с базой данных и содержит бизнес-логику приложения, а слой контроллеров обрабатывает запросы на сервер и управляет слоем модели. Это позволяет более эффективно управлять данными и запросами, а также отделить логику приложения от пользовательского интерфейса.

Приведена диаграмма классов для слоя модели, представленная на рисунке 39. Классы, которые были созданы, имеют полное соответствие с таблицами, находящимися в базе данных.

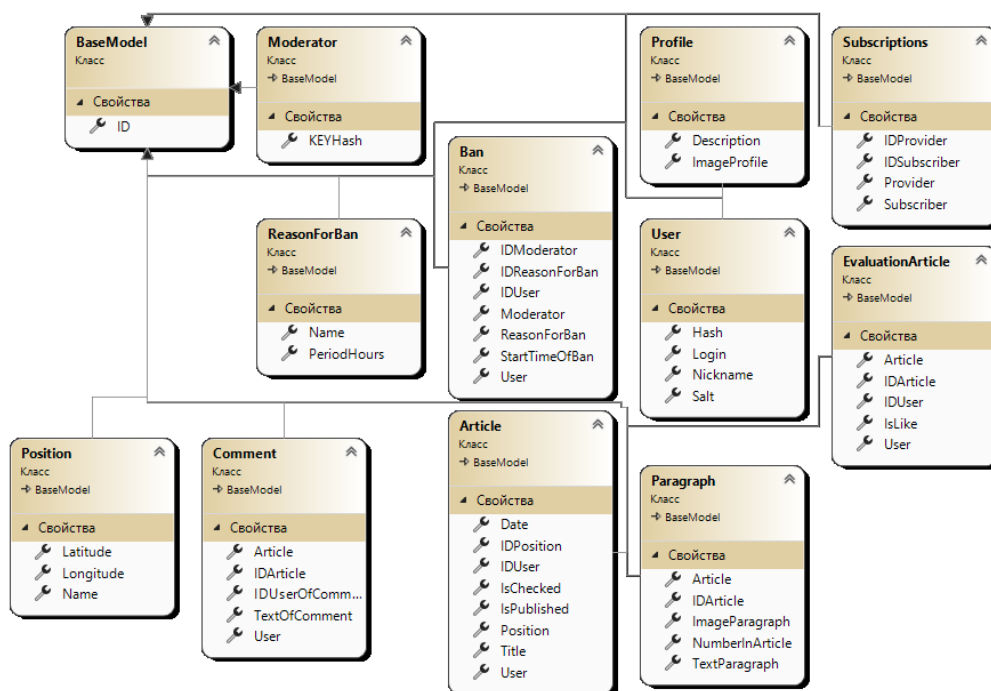


Рисунок 39 – Диаграмма классов серверной части (слой модели)

Для обеспечения полного использования преимуществ объектно-ориентированного программирования и уменьшения вероятности ошибок, был разработан вспомогательный класс BaseModel, который является базовым для всех остальных классов и содержит всего одно поле – уникальный идентификатор. Использование этого класса позволяет избежать дублирования кода и обеспечить единообразие в реализации классов, наследующих его. Кроме того, наличие единого поля в виде уникального идентификатора упрощает работу с базой данных и повышает читабельность кода, что в свою очередь улучшает поддержку кода в будущем.

Диаграмма классов слоя контроллеров представлена на рисунке 40.

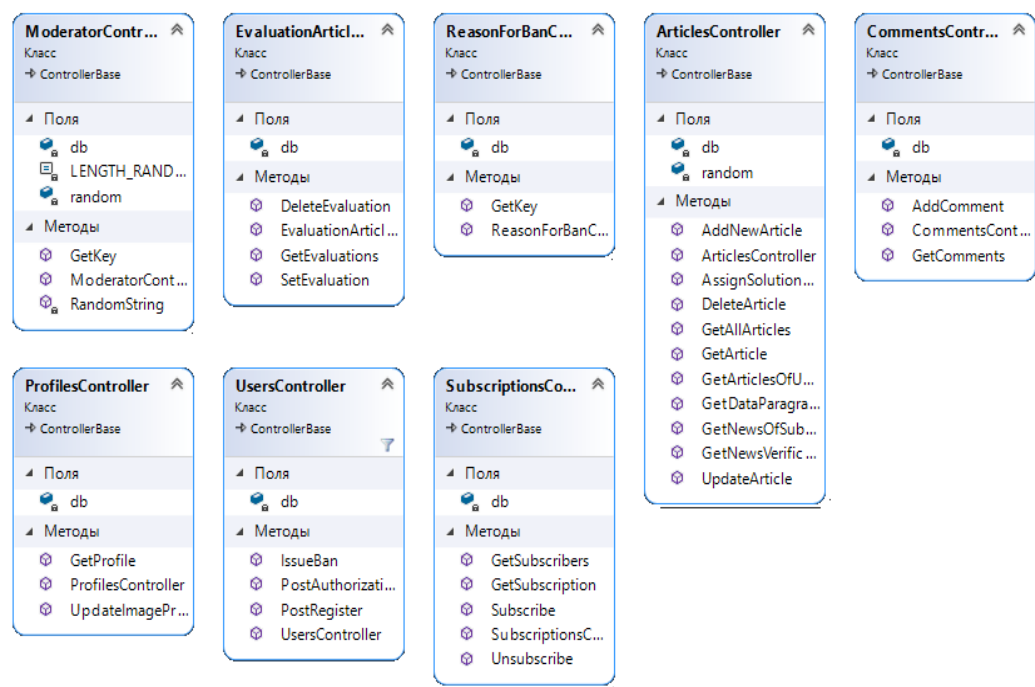


Рисунок 40 – Диаграмма классов серверной части (слой контроллеров)

Более подробное описание классов слоя контроллеров и их назначений представлено в таблице 10.

Таблица 10 – Описание классов слоя контроллеров

Класс	Назначение
UsersController	Отвечает за обработку запросов, связанных с пользователями, таких как регистрация, авторизация и блокировка пользователей.

Продолжение таблицы 10

Класс	Назначение
SubscriptionsController	Обрабатывает запросы по получению списка подписок пользователя, проверки подписки и отписки.
ReasonForBanController	Вспомогательный контроллер, который обрабатывает запросы модераторов на получение всех причин блокировок.
ProfilesController	Контроллер обрабатывает запросы по получению профиля и изменения изображения и описания своего профиля.
ModeratorController	Данный контроллер обрабатывает единственный запрос на создание нового модератора с уникальным ключом.
EvaluationArticleController	Обрабатывает запросы, связанные с оценками новостей, включая создание новых оценок, удаление существующих оценок и возврат всех оценок для определенной новости.
CommentsController	Контроллер для обработки запросов, связанных с комментариями, включая добавление нового комментария и получение всех комментариев к определенной новости.
ArticlesController	Контроллер для обработки запросов, связанных с новостями, включая создание новой новости, редактирование, получение, удаление новостей и другие функции.

Листинг серверной части представлен в приложении Б.

3.3 Реализация клиентской части приложения

Данное мобильное приложение было разработано на языке программирования C# с использованием среды разработки Visual Studio 2019 и библиотек Xamarin.Forms. Архитектура приложения была описана с помощью диаграммы компонентов, представленной на рисунке 41. Эта диаграмма позволяет визуально представить структуру приложения и взаимодействие его компонентов, что способствует лучшему пониманию и обеспечению его модульности и расширяемости. В данном случае приложение разбито на набор независимых сервисов и компонентов, каждый из которых выполняет четко определенные функции.

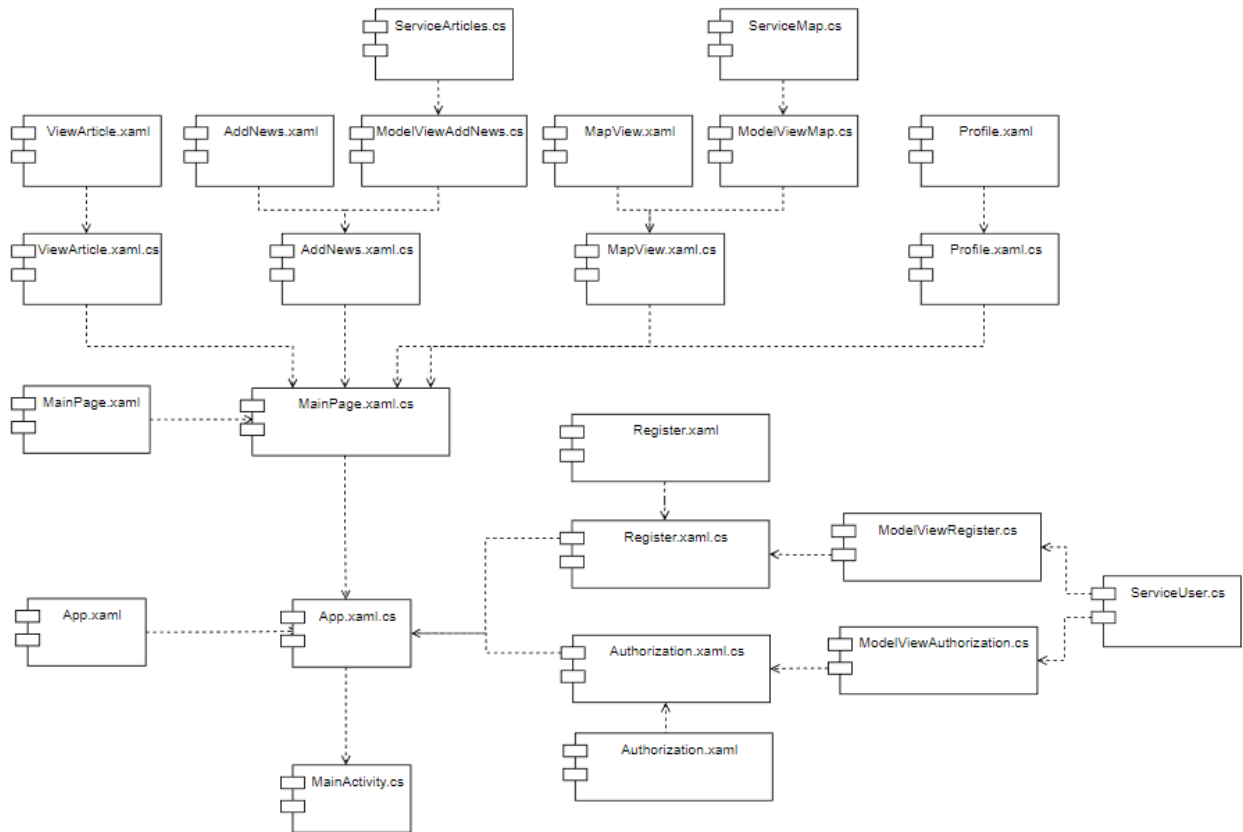


Рисунок 41 – Диаграмма компонентов мобильного приложения

Описание компонентов представлено в таблице 11.

Таблица 11 – Компоненты клиентского приложения

Компонент	Назначение
MainActivity.cs	Начальный класс, который отвечает за вход в систему
App.xaml.cs	Класс предназначен для отображения интерфейса в системе
App.xaml	Класс, который содержит ресурсы для интерфейса
Register.xaml.cs	Реализует страницу с регистрацией
Register.xaml	Реализует интерфейс страницы с регистрацией
ModelViewRegister.cs	Реализует прослойку отображения для страницы регистрации между представлением и данными, которые получены с сервера
Authorization.xaml.cs	Реализует страницу с авторизацией
Authorization.xaml	Реализует интерфейс для страницы авторизации

Продолжение таблицы 11

Компонент	Назначение
ModelViewAuthorization.cs	Реализует прослойку отображения для страницы авторизации между представлением и данными, которые получены с сервера
ServiceUser.cs	Класс предназначен для получения данных с сервера, которые касаются пользователя
MainPage.xaml.cs	Класс отображает выбранную страницу и меню
MainPage.xaml	Файл отображает интерфейс выбранной страницы и меню
ViewArticle.xaml.cs	Файл класса страницы выбранной новости
ViewArticle.xaml	Реализует интерфейс страницы выбранной новости
AddNews.xaml.cs	Файл класса страницы с созданием новой новости
AddNews.xaml	Реализует интерфейс страницы с созданием новости
ModelViewAddNews.cs	Реализует прослойку отображения страницы по созданию новости между представлением и данными, которые получены с сервера
ServiceArticle.cs	Класс предназначен для получения данных с сервера, которые касаются новостей
MapView.xaml.cs	Файл класса страницы с новостями на карте
MapView.xaml	Реализует интерфейс страницы с новостями на карте
ModelViewMap.cs	Реализует прослойку отображения страницы с новостями на карте между представлением и данными, которые получены с сервера
ServiceMap.cs	Класс предназначен для получения данных с сервера, которые касаются новостей на карте
Profile.xaml.cs	Файл класса страницы профиля пользователя
Profile.xaml	Реализует интерфейс страницы профиля пользователя

На рисунке 42 представлена диаграмма развертывания, которая показывает, как расположены компоненты системы в физическом отношении.

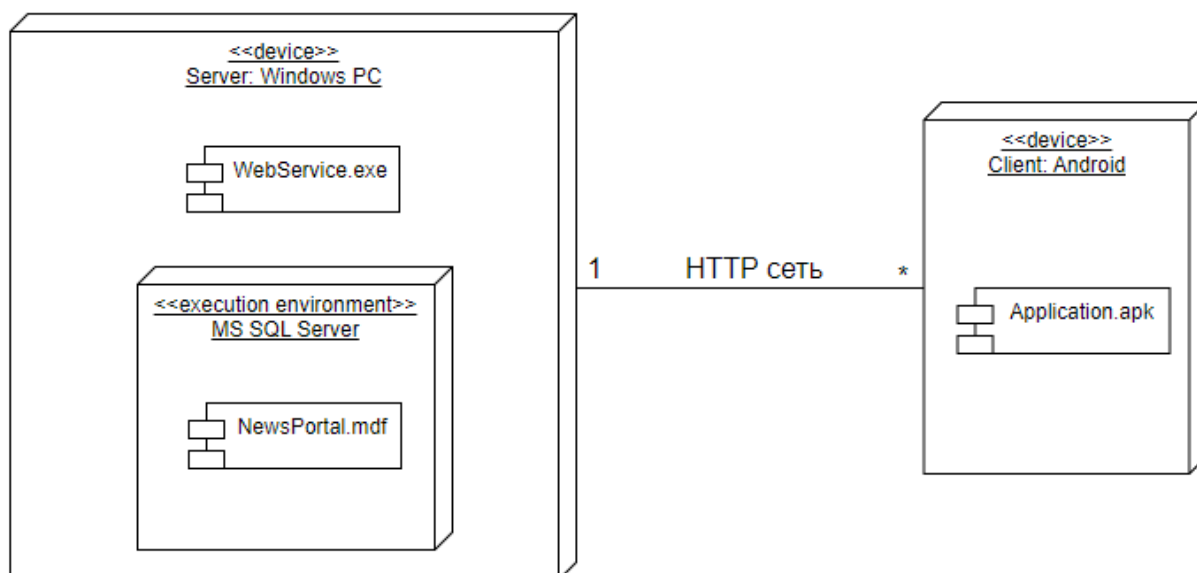


Рисунок 42 – Диаграмма развертывания мобильного приложения

Данная диаграмма развертывания демонстрирует, как в целом устроено приложение для мобильного устройства под операционной системой Android и сервером, который запущен на настольном компьютере под управлением ОС Windows. Серверная часть приложения содержит сервис обработки запросов и базу данных, которая реализована на платформе Microsoft SQL Server. Для создания связи между клиентом и сервером используется сеть HTTP. Такая архитектура предоставляет пользователю возможность удаленного взаимодействия с приложением с учетом того, что он находится в области сети, по которой осуществляется обмен данными.

Листинг клиентской части программы представлен в приложении В.

3.4 Реализация нейронной сети для классификации текста

Для решения задачи классификации текста на два класса – допустимый и недопустимый для публикации – в рамках данной выпускной квалификационной работы была разработана простая полносвязная нейронная сеть на языке Python с использованием библиотеки Keras.

Результат обучения нейронной сети приведен на рисунке 43.

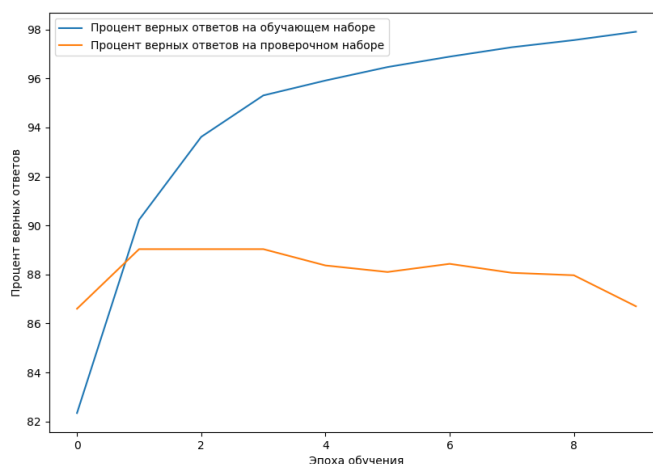


Рисунок 43 – Структура нейронной сети для классификации текста

Из графика видно, что доля правильных ответов на проверочном наборе составляет примерно 85%. Следует отметить, что для валидации было использовано 20% исходной выборки.

Такое поведение графика говорит об успешном обучении нейронной сети. Она способна давать достаточно точные прогнозы и не страдает от переобучения, то есть не запоминает ответы.

Для оптимизации процесса обучения был выбран алгоритм оптимизации Adam. Он является адаптивным методом градиентного спуска, который изменяет скорость обучения для каждого параметра в процессе обучения, что позволяет быстрее сходиться к оптимальному решению.

Для расчета ошибки была применена бинарная кросс-энтропия, которая идеально подходит для данной задачи, поскольку мы имеем дело с классификацией на два класса, что соответствует нашему приложению.

Результаты экспериментов с полносвязной нейронной сетью показали ее высокую эффективность в задаче классификации текста на два класса. Оптимальное количество скрытых слоев, функция активации и оптимизатор позволили достичь высокой точности классификации. Этот результат говорит о том, что разработанная нейронная сеть может быть применена для решения практических задач в сфере обработки естественного языка.

4 Тестирование программного продукта

4.1 Тестирование серверной части приложения

В рамках тестирования приложения было проведено функциональное тестирование, включающее в себя проверку работоспособности основных функций приложения. Для тестирования серверной части использовалось удобное приложение Postman, а также были написаны тест-кейсы, в которых проверялась корректность обработки различных http запросов и взаимодействие с базой данных. Тесты для серверной части приложения приведены в таблице 12. Рисунки с отображением результатов тестирования приведены в приложении Г.

Таблица 12 – Результаты тестирования серверной части

№	Описание теста	Запрос		Результат
		Метод	Параметры	
1	Регистрация нового пользователя с логином NewUserLogin, паролем NewUserPassword и псевдонимом NewUserNickname	POST	Login = NewUserLogin Password = NewUserPassword Nickname = NewUserNickname	Тест пройден. Пользователь зарегистрирован в системе. Код ответа HTTP – 200. Содержимое ответа – 11 (уникальный идентификатор пользователя в системе) (Рисунок Г.1).
2	Авторизация существующего пользователя с логином admin и паролем admin	POST	Login = NewUserLogin Password = NewUserPassword	Тест пройден. Пользователь авторизован в системе. Код ответа HTTP – 200. Содержимое ответа – 1 (идентификатор пользователя в системе) (Рисунок Г.2)
3	Получения информации об актуальных новостях для карты	GET		Тест пройден. Новости получены. Код ответа HTTP – 200. Содержимое ответа – json список информации по новостям (Рисунок Г.3)
4	Создание новой новости	POST	ID=-1, Title = Новая новость, IDUser = 1, Address = (Latitude = 53,175368, Longitude = 45,034827, Name = г. Пенза), ParagraphTexts = (Новый абзац)	Тест пройден. Новость успешно добавлена. Код ответа HTTP – 200. Содержимое ответа – сообщение «Статья успешно добавлена» (Рисунок Г.4)

Продолжение таблицы 12

№	Описание теста	Запрос		Результат
		Метод	Параметры	
5	Редактирование у существующей новости с ID=16 (Идентификатор) заголовок на «Измененный заголовок» и адрес на «г Санкт-Петербург»	PUT	ID=16, Title = Измененный заголовок, IDUser = 1, Address = { Latitude = 53,175368, Longitude = 45,034827, Name = г. Пенза}, ParagraphTexts = (Новый абзац)	Тест пройден. Новость успешно отредактирована. Код ответа HTTP – 200. Содержимое ответа – сообщение «Статья успешно изменена» (Рисунок Г.5)
6	Получение базовой информации по новости с идентификатором ID=16 для формирования интерфейса	GET	ID=16	Тест пройден. Информация о новости получена. Код ответа HTTP – 200. Содержимое ответа – json объект с информацией о новости. (Рисунок Г.6)
7	Получение параграфа с номером NumberInArticle = 1 новости с идентификатором IDArticle=16	GET	NumberInArticle = 1, IDArticle=16	Тест пройден. Данные параграфа получены. Код ответа HTTP – 200. Содержимое ответа – Json объект (Рисунок Г.7)
8	Удаление новости из системы с идентификатор ID=16	DELETE	ID=16	Тест пройден. Новость удалена из системы. Код ответа HTTP – 200. Содержимое ответа – отсутствует (Рисунок Г.8)
9	Получение всех новостей, созданных пользователем с идентификатором ID=7	GET	ID=7	Тест пройден. Новости, определенного пользователя, получены. Код ответа HTTP – 200. Содержимое ответа – json список информации о новостях. (Рисунок Г.9)
10	Получение информации о профили пользователя с идентификатором ID=1	GET	ID=1	Тест пройден. Информация о профили получена. Код ответа HTTP – 200. Содержимое ответа – json объект с информацией о профили (Рисунок Г.10)
11	Изменение изображения в профили пользователя с идентификатором ID=1	PUT	ID=1, Image = null	Тест пройден. Изображение в профили изменено. Код ответа HTTP – 200. Содержимое ответа – отсутствует (Рисунок Г.11)
12	Добавления подписки пользователя с идентификатором ID=4 на пользователя с идентификатором ID=5	POST	IDSubscriber=4, IDProvider=5	Тест пройден. Подписка была осуществлена. Код ответа HTTP – 200. Содержимое ответа – отсутствует (Рисунок Г.12)
13	Получение всех подписок для пользователя с идентификатором ID=1	GET	ID = 1	Тест пройден. Подписки пользователя были получены. Код ответа HTTP – 200. Содержимое ответа – json список с пользователями, на которые подписан текущей пользователь (Рисунок Г.13)

Продолжение таблицы 12

№	Описание теста	Запрос		Результат
		Метод	Параметры	
14	Получение одной существующей подписки для проверки пользователя с идентификатором ID=1 на пользователя с идентификатором ID=7	GET	IDAuthor=1, IDUser = 7	Тест пройден. Подписка получена. Код ответа HTTP – 200. Содержимое ответа – json объект подписки. (Рисунок Г.14)
15	Отписка одного пользователя с идентификатором ID=4 от пользователя с идентификатором ID=5	DELETE	IDAuthor=5, IDUser = 4	Тест пройден. Пользователь отписался от другого пользователя. Код ответа HTTP – 200. Содержимое ответа – отсутствует. (Рисунок Г.15)
16	Добавление нового комментария от пользователя с идентификатором IDUserOfComment=7 на новость с идентификатором IDArticle=5 с текстом TextOfComment = «Спасибо, познавательно»	POST	IDUserOfComment= 7, IDArticle =5, TextOfComment = «Спасибо, познавательно»	Тест пройден. Комментарий добавлен к новости. Код ответа HTTP – 200. Содержимое ответа – отсутствует. (Рисунок Г.16)
17	Получение всех комментариев для новости с идентификатором IDArticle=5	GET	IDArticle=5	Тест пройден. Комментарии новости получены. Код ответа HTTP – 200. Содержимое ответа – json список комментариев. (Рисунок Г.17)
18	Установка оценки с положительной оценкой IsLike = true для новости с идентификатором IDArticle = 5 от пользователя с идентификатором IDUser = 5	POST	IDArticle =5, IDUser =5, IsLike = true	Тест пройден. Оценка установлена для новости. Код ответа HTTP – 200. Содержимое ответа – отсутствует. (Рисунок Г.18)
19	Получения всех оценок для новости с идентификатором IDArticle=5	GET	IDArticle = 5	Тест пройден. Оценки для новости получены. Код ответа HTTP – 200. Содержимое ответа – json список оценок. (Рисунок Г.19)
20	Удаление оценки для новости с идентификатором IDArticle = 5 от пользователя с идентификатором IDUser = 1	DELETE	IDArticle =5 IDUser =1	Тест пройден. Оценка для новости удалена. Код ответа HTTP – 200. Содержимое ответа – отсутствует. (Рисунок Г.20)

Окончание таблицы 12

№	Описание теста	Запрос		Результат
		Метод	Параметры	
21	Получение новости для модерации с ключом KEY = O EZST8JOX32ZDCTN83R6P7O79MAEF8ZM	GET	KEY = O EZST8JOX32ZDCTN83R6P7O79MAEF8ZM	Тест пройден. Новость для модерации получена. Код ответа HTTP – 200. Содержимое ответа – json список параграфов новости. (Рисунок Г.21)
22	Принять решение с запретом IsAcceptable=false по новости для публикации с идентификатором IDArticle =5, автор которой имеет идентификатор IDAuthor =1, от модератора с ключом KEY = O EZST8JOX32ZDCTN83R6P7O79MAEF8ZM	PUT	IDArticle =5, IDAuthor =1, IsAcceptable=false, KEY = O EZST8JOX32ZDCTN83R6P7O79MAEF8ZM	Тест пройден. Новость запрещена для публикации. Код ответа HTTP – 200. Содержимое ответа – отсутствует. (Рисунок Г.22)
23	Получить новый ключ для модерации по уже созданному ключу KEY = O EZST8JOX32ZDCTN83R6P7O79MAEF8ZM	GET	KEY = O EZST8JOX32ZDCTN83R6P7O79MAEF8ZM	Тест пройден. Новый ключ для модерации получен. Код ответа HTTP – 200. Содержимое ответа – ключ для модерации. (Рисунок Г.23)
24	Заблокировать пользователя с идентификатор IDUser =10 по причине с идентификатором IDReason = 1 по ключу модератора KEY = O EZST8JOX32ZDCTN83R6P7O79MAEF8ZM	POST	IDUser=10, IDReason = 1, KEY = O EZST8JOX32ZDCTN83R6P7O79MAEF8ZM	Тест пройден. Пользователь заблокирован в системе. Код ответа HTTP – 200. Содержимое ответа – отсутствует (Рисунок Г.24)

Приведенные тесты выше были пройдены успешно, следовательно, можно сделать вывод, что серверная часть работает корректно.

4.2 Тестирование клиентской части приложения

4.2.1 Функциональное тестирование

Для проверки клиентской части приложения будет проведено функциональное тестирование. Тестирование должно быть полным, затрагивая все функциональные возможности клиентского приложения. В таблице 13 приведены варианты тестов. Результат тестов приведен в приложении Д.

Таблица 13 – Функциональное тестирование

Вариант использования	Тест и параметры	Результат
Зарегистрироваться	Ввести данные для регистрации правильные. Логин:LoginNew Пароль: PasswordNew Псевдоним: Иван Васильевич	Тест пройден. Регистрация прошла успешно. (Рисунок Д.1)
	Использовать в регистрации уже созданный логин. Логин: admin Пароль: somePassword Псевдоним: Кирилл	Тест пройден. Система вывела сообщение о недопустимости данного логина. (Рисунок Д.2)
	Использовать в регистрации пароль меньше 4-х символов Логин: someLogin Пароль: pas Псевдоним: Максим	Тест пройден. Система вывела сообщение об ошибке введенных данных. (Рисунок Д.3)
	Использовать запрещенные символы при вводе данных Логин: MyLogin!? Пароль: \$Password% Псевдоним: Юлия	Тест пройден. Система вывела сообщение об ошибке введенных данных. (Рисунок Д.4)
Авторизироваться	Ввести корректные данные для авторизации. Логин:admin Пароль:admin	Тест пройден. Авторизация прошла успешно, пользователь получил доступ к приложению. (Рисунок Д.5)
	Авторизировать под несуществующими данными. Логин: DoesntExistLogin Пароль: DoesntExistPassword	Тест пройден. Выведено сообщение о невозможности авторизации. (Рисунок Д.6)
	Авторизироваться под заблокированным пользователем Логин:BanLogin Пароль:BanPassword	Тест пройден. Выведено сообщение об блокировке с датой. (Рисунок Д.7)

Продолжение таблицы 13

Вариант использования	Тест и параметры	Результат
Добавить новость	Создать новость с корректными данными. Заголовок: Отдых в Пензе Адрес: г. Пенза Первый абзац: Представлен дачный поселок, ниже вид утром Изображение: Добавленное из локального хранилища Второй абзац: Ждём всех на отдых	Тест пройден. Новость создана, система вывела сообщение. (Рисунок Д.8)
	Создать новость без заголовка Заголовок: «» Адрес: г. Москва Абзац: Новость без заголовка	Тест пройден. Система вывела сообщение о некорректных данных. (Рисунок Д.9)
	Создать новость без указания адреса. Заголовок: Новость без адреса Адрес: «» Абзац: Новость без адреса	Тест пройден. Система вывела сообщение о некорректности данных. (Рисунок Д.10)
Изменить изображение в профили	Установить новое изображение для профиля из локального хранилища	Тест пройден. Система установила новое изображение для профиля пользователя. (Рисунок Д.11)
Изменить описание в профили	Установить новое описание в профили в профили	Тест пройден. Система изменила текстовое описание профиля пользователя (Рисунок Д.12)
Подписаться на пользователя	Подписаться на пользователя	Тест пройден. Подписка оформлена, в профили отображается подписка. (Рисунок Д.13)
Отписаться от пользователя	Использовать возможность отписки в профили	Тест пройден. Подписка удалена, отображение в профили перестало существовать. (Рисунок Д.14)

Окончание таблицы 13

Вариант использования	Тест и параметры	Результат
Открыть новость	Открыть новость при выборе на карте	Тест пройден. Новость открылась. (Рисунок Д.15)
	Открыть новость в профиле	Тест пройден. Новость открылась (Рисунок Д.16)
Оставить комментарий	Написать комментарий для новости Тест комментария: Интересная новость	Тест пройден. Комментарий оставлен под новостью. (Рисунок Д.17)
Удалить новость	Удалить созданную новость в профиле.	Тест пройден. Новость удалена, её упоминание в профиле отсутствует, система вывела сообщение об удалении. (Рисунок Д.18)
Поставить оценку новости	Установить на новость положительную оценку	Тест пройден. У новости увеличился счётчик положительных оценок. (Рисунок Д.19)
	Установить на новость негативную оценку	Тест пройден. У новости увеличился счётчик негативных оценок. (Рисунок Д.20)
Заблокировать пользователя	Выбрать пользователя для блокировки Идентификатор пользователя: 9 Причина блокировки: Временная блокировка за нецензурный комментарий	Тест пройден. Пользователь заблокирован, система вывела сообщение о результате. (Рисунок Д.21)
Проверить новость для публикации	Новость поступила на проверку, которая разрешается для публикации. Нейронная сеть определила эту новость, как допустимую.	Тест пройден. Новость была допущена до публикации, система вывела сообщение об успешности выполнения. (Рисунок Д.22)
	Новость поступила на проверку, которая запрещается для публикации. Нейронная сеть также определила эту новость, как недопустимую.	Тест пройден. Новость была заблокирована для публикации, система вывела сообщение об успешности выполнения. (Рисунок Д.23)

Функциональное тестирование показало корректность работы клиентской части.

После определения структуры для логина необходимо привести примеры, которые будут содержать уникальную последовательность классов эквивалентности. Данные примеры в полной мере будут покрывать метод по созданию логина. Примеры приведены в таблице 15.

Таблица 15 – Тест -кейсы для создания логина

№	КЭ	Входные данные	Выходные данные
1	Все правильные 1,4,6,7,8,10	MyNewNickname-2	Успешно
2	2,4,6,7,8,10	L-1	Ошибка: меньше 4 символов
3	3,4,6,7,8,10	Super1-LongNicknameitsvery goodBecauseitmeansthatn verdoesntentertotheaccount	Ошибка: больше 32 символов
4	1,5,6,7,8,10	-Danila-25Pro	Ошибка: первый символ не является буквой
5	1,4,7,8,9,10	How-are-you2???	Ошибка: Символ не входящий в список разрешенных
6	1,4,7,8,11	text-2	Ошибка: Нет большой буквы

Как видно из таблицы примеры полностью покрывают условия для создания логина. Данные тест-кейсы в последующем могут быть использованы для реализации модульных тестов.

4.2.3 Модульное тестирование

Модульные тесты – это средство, помогающие автоматизировать процесс тестирования программного обеспечения. Модульные тесты обеспечивают постоянный контроль за корректностью работы программного обеспечения. Их основная задача заключается в получении и сравнение результата фактического с ожидаемого, используя заранее описанные функции и данные.

В данной работе были написаны модульные тесты для проверки данных, которые используются при регистрации, таких как логин и пароль. Тесты затрагивают функцию `SuitForm`, которая содержится в классе `Validation`, который отвечает за проверку введенных данных для регистрации.

Модульные тесты были реализованы в классе `ValidationRegisterTests`, код которого представлен в листинге 1.

Листинг 1 – Код модульных тестов

```

using Microsoft.VisualStudio.TestTools.UnitTesting;
using UnitTest;

namespace StorageOfTest
{
    [TestClass]
    public class ValidationRegisterTests
    {
        [TestMethod]
        public void RegisterCorrectDataLogin_admin_Password_admin()
        {
            Validation validation = new Validation();
            User user = new User() { Login = "admin", Password = "admin" };

            bool result = validation.SuitForm(user);

            Assert.IsTrue(result);
        }
        [TestMethod]
        public void RegisterLogin_log_Password_password()
        {
            Validation validation = new Validation();
            User user = new User() { Login = "log", Password = "password" };

            bool result = validation.SuitForm(user);

            Assert.IsFalse(result);
        }
        [TestMethod]
        public void RegisterLogin_123Login_Password_password()
        {
            Validation validation = new Validation();
            User user = new User() { Login = "123Login", Password =
"password" };

            bool result = validation.SuitForm(user);

            Assert.IsFalse(result);
        }
        [TestMethod]
        public void
RegisterLogin_TooMuchLoginSymbolsCountitdoesntSuitimsureaboutit_Password_pass
word()
        {
            Validation validation = new Validation();
            User user = new User() { Login =
"TooMuchLoginSymbolsCountitdoesntSuitimsureaboutitpluseveralsymbols",
Password = "password" };

            bool result = validation.SuitForm(user);

            Assert.IsFalse(result);
        }
        [TestMethod]
        public void RegisterLogin_EmptyLogin_Password_password()
        {
            Validation validation = new Validation();
            User user = new User() { Login = "", Password = "password" };

            bool result = validation.SuitForm(user);

```

```

        Assert.IsFalse(result);
    }
    [TestMethod]
    public void RegisterLogin_UseDash_Password_password()
    {
        Validation validation = new Validation();
        User user = new User() { Login = "Login-With-Dash", Password =
"password" };

        bool result = validation.SuitForm(user);

        Assert.IsTrue(result);
    }
    [TestMethod]
    public void RegisterLogin_IncorrectSymbol_Password_password()
    {
        Validation validation = new Validation();
        User user = new User() { Login = "MyLoginHave!?Incorrect
symbols", Password = "password" };

        bool result = validation.SuitForm(user);

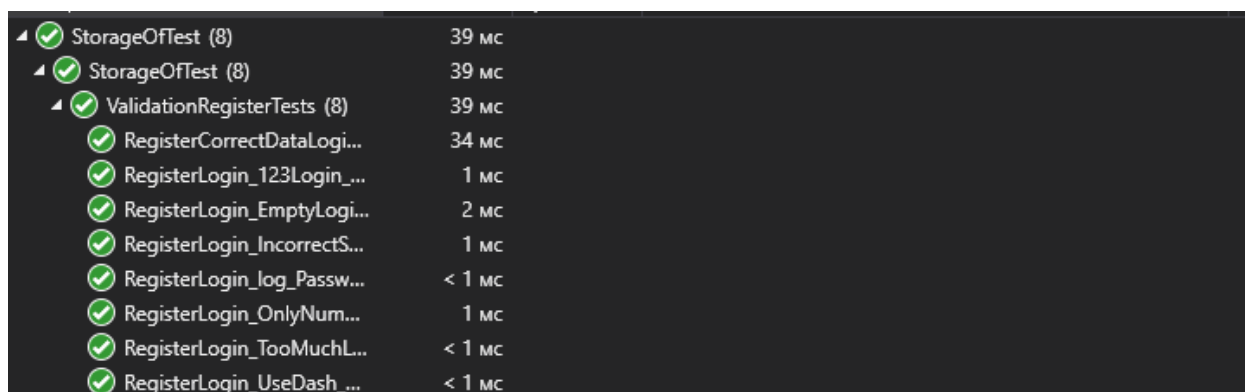
        Assert.IsFalse(result);
    }
    [TestMethod]
    public void RegisterLogin_OnlyNumbers_Password_password()
    {
        Validation validation = new Validation();
        User user = new User() { Login = "12345678909876", Password =
"password" };

        bool result = validation.SuitForm(user);

        Assert.IsFalse(result);}}

```

Результат, полученный при запуске модульных тестов представлен на рисунке 45.



StorageOfTest (8)	39 мс
StorageOfTest (8)	39 мс
ValidationRegisterTests (8)	39 мс
RegisterCorrectDataLogi...	34 мс
RegisterLogin_123Login_...	1 мс
RegisterLogin_EmptyLogi...	2 мс
RegisterLogin_IncorrectS...	1 мс
RegisterLogin_log_Passw...	< 1 мс
RegisterLogin_OnlyNum...	1 мс
RegisterLogin_TooMuchL...	< 1 мс
RegisterLogin_UseDash_...	< 1 мс

Рисунок 45 – Результаты модульного тестирования

Тесты, приведенные выше были успешно выполнены, что свидетельствует о корректной работоспособности проверяемого участка кода.

4.3 Метрики кода

Метрики кода отображают количественные величины программного обеспечения, которые предоставляют разработчику доступ к таким свойствам, как эффективность, оптимизацию и масштабируемость написанного кода приложения.

Способы, которые используют для получения метрик код, существует огромное количество. Данные способы различны по своим алгоритмам и по информации, полученной в виде результата. Для разработанного приложения будут проведены следующие группы метрик:

- метрики размера программы (SLOC-метрики);
- метрики сложности потока управления программ (метрики Холстеда и цикломатическая сложность).

Для оценки возьмём метод PostRegister класса UserController, который ответственен за регистрацию нового пользователя в системе. Код для метода регистрации приведен ниже в листинге 2.

Листинг 2 – Регистрация нового пользователя

```
public async Task<ActionResult> PostRegister([FromBody] UserData userData)
{
    try
    {
        //Проверяем корректность полученных данных
        if (userData == null || userData.Login==null ||
        userData.Login=="" || userData.Password==null || userData.Password=="")
            return BadRequest("Не все нужные данные были введены");
        //Находим пользователя в базе данных по логину
        var user =
        db.users.FirstOrDefault(x=>x.Login==userData.Login);
        //Если пользователь существует с таким логином, то заявляем,
        что это ошибка
        if (user!=null)
            return BadRequest("Пользователь с таким логином уже есть");
        //Создаём соль и соленый хеш для пароля нового пользователя
        var Salt = GenerateSalt();
        var Hash =
        GenerateSaltedHash(Encoding.Unicode.GetBytes(userData.Password),
        Encoding.Unicode.GetBytes(Salt));
```

```

        //Создаём и добавляем нового пользователя
        User userForAdded = new User
{Login=userData.Login,Nickname=userData.Nickname,Hash=Hash,Salt=Salt};
        db.users.Add(userForAdded);
        db.SaveChanges();
        //Создаём для нового пользователя профиль
        Profile profile = new Profile { ID = userForAdded.ID,
Description = null, ImageProfile = null };
        db.profiles.Add(profile);
        db.SaveChanges();
        return Ok(userForAdded.ID.ToString());}
    catch
    {
        return BadRequest();}
}

```

Метод для регистрации является серверным, который обрабатывает попытки пользователей создать новую учетную запись в системе. Метод принимает на вход логин, пароль и псевдоним. Задача метода выяснить существует ли уже учетная запись с аналогичным логином, также выполнить процесс по преобразованию пароля в хеш и создать профиль для данного пользователя.

Для данного метода будут вычислены две группы метрик: метрики размера программы и метрики Холстеда.

4.3.1 Метрики размера программы

Первой проведенной метрикой будет подсчёт размера программы, где будет вычислено количество строк программного кода, а также число комментариев. Метрики будут найдены на основе выше приведенного кода. Результат представлен в таблице 16.

Таблица 16 – Метрики размера программы

Метрика	Количество
Физических строк кода	31
Логических строк кода	17
Строки-комментарии	6

Приведенные количественные значения дают возможность вычислить процент комментариев для рассматриваемого участка кода. Процент

комментариев равен $(6/31) * 100 = 20$. Полученный результат свидетельствует о достаточности комментирования кода. Это полезно для облегчения понимания кода другими разработчиками и обеспечения его поддержки в будущем.

4.3.2 Метрики Холстеда

Для вычисления метрик Холстеда необходимо измерить 4 характеристики программы

- число уникальных операторов;
- число уникальных операндов;
- общее число операторов;
- общее число операндов.

Для вычисления характеристик были составлены словарь операторов, которой отображен в таблице 17 и словарь операндов, который описан в таблице 18.

Таблица 17 – Словарь операторов

Оператор	Число операторов
Try catch	1
if	2
==	5
return	4
=	12
;	12
!=	1
new	2
{}	6
.	25
=>	1
()	16

Таблица 18 – Словарь операндов

Операнд	Число операндов
userData	10
null	6
user	2
db	5
Salt	3
Hash	2
userForAdded	4
profile	2

Таким образом, число уникальных операторов программы $n1 = 12$, число уникальных операндов программы $n2 = 8$, общее число операторов в программе $N1 = 86$, общее число операндов в программе $N2 = 34$.

На основе метрик Холстеда можно вычислить целый ряд характеристик программы. Рассмотрим некоторые из них:

- словарь программы $n = n1 + n2$;
- длина программы $N = N1 + N2$;
- объем программы $V = N * \log_2 n$;
- сложность программы $D = (n1/2) * (N2/n2)$;
- оценка интеллектуальных усилий $E = D * V$.

Рассчитаем словарь программы: $n = 12 + 8 = 20$.

Рассчитаем длину программы: $N = 86 + 34 = 120$.

Рассчитаем объем программы: $V = 120 * \log_2 20 = 516$.

Рассчитаем сложность программы: $D = (12/2) * (34/8) = 25,5$.

Рассчитаем оценку интеллектуальных усилий: $E = 25,5 * 516 = 13158$.

Подсчитанные значения предоставляют информацию о том, что в проекте количество используемых переменных не велико, а значит данный проект будет проще поддерживать в будущем.

Также длина программы и её объем показывают, что код является обширным, но при этом он разбит на логические функциональные блоки, которые легко масштабировать.

Сложность программы и оценка интеллектуальных усилий демонстрирует, что имеется множества высокоуровневых связей.

Полученные характеристики показывают, что программный код имеет достаточно высокий уровень качества и удобства для дальнейшей поддержки.

Заключение

Выполнена выпускная квалификационная работа по разработке мобильного приложения для программных средств поддержки процессов организации и функционирования новостного портала.

При анализе требований и предметной области были проанализированы программные продукты, выполняющие аналогичные функции, с целью выявления их преимуществ и недостатков. Данный анализ позволил определить функциональные и нефункциональные требования к создаваемому приложению, а также произвести исследование целевой аудитории.

Произведено составление плана разработки приложения и проведен расчет бюджета. Из результатов планирования следует, что проект займет 135 дней на реализацию, а расчетные затраты составят 2 402 360 рублей, что говорит о том, что программа является экономически выгодным проектом.

В процессе проектирования приложения была разработана логическая модель базы данных и создана архитектура мобильного приложения. База данных была создана при помощи SQL, а для ее управления была выбрана СУБД Microsoft SQL Server 2018. Серверная часть приложения была создана на языке программирования C# с использованием среды разработки Microsoft Visual Studio 2019. Для разработки нейронной сети был выбран язык Python, который поддерживает библиотеку Keras. Для клиентской части был выбран язык программирования C#, а среда разработки – Visual Studio 2019. Для разработки клиентской части была использована библиотека Xamarin Forms.

После завершения разработки приложения было проведено тестирование функциональности как серверной, так и клиентской частей, которое показало, что они работают корректно. Для оценки качества программного кода были рассчитаны соответствующие метрики.

В планах на будущее – публикация разработанного приложения в магазине мобильных приложений для системы Android Google Play,

развертывание серверной части на облачном сервисе и создание версии мобильного приложения для операционной системы iOS, а также дальнейшее улучшение функционала приложения, включая средства рекомендации новостей и улучшения взаимодействия пользователей.

При выполнении выпускной квалификационной работы были успешно выполнены все поставленные задачи, что позволило достичь поставленных целей и получить желаемый результат. Результатом работы стало разработанное мобильное приложение для поддержки процессов организации и функционирования новостного портала с учетом всех требований и особенностей.

Список использованных источников

1. Указ Президента Российской Федерации от 10.10.2019 г. № 490 «О развитии искусственного интеллекта в Российской Федерации». [Электронный ресурс]. URL: <http://publication.pravo.gov.ru/Document/View/0001201910110003> (дата обращения: 14.05.2023).
2. Постановление от 30 ноября 2021 г. № 2133 «О создании инновационного научно-технологического центра «Квантовая долина»». [Электронный ресурс]. URL: <http://government.ru/docs/all/137847/> (дата обращения: 14.05.2023).
3. Исследование: почти 70% россиян используют смартфон для выхода в интернет // ТАСС [Электронный ресурс]. – 2020. – 21 сентября. – URL: <https://tass.ru/obschestvo/9508331> (дата обращения: 15.05.2023).
4. Указ Президента Российской Федерации о стратегии развития информационного общества в Российской Федерации на 2017-2030 годы [Электронный ресурс] – Москва, 2017. – URL: <http://static.kremlin.ru/media/acts/files/0001201705100002.pdf> (дата обращения: 15.05.2023).
5. Господдержка электронных СМИ в 2023 году // Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации [Электронный ресурс]. – URL: <https://digital.gov.ru/ru/activity/directions/1105/> (дата обращения: 15.05.2023).
6. О компании RB.RU // Новостной портал RB.RU [Электронный ресурс]. – URL: <https://rb.ru/about/> (дата обращения: 16.05.2023).
7. Canada Sanctions Additional Russian Propagandists // Global Affairs Canada [Электронный ресурс]. – URL: <https://www.canada.ca/en/global-affairs/news/2023/02/canada-sanctions-additional-russian-propagandists.html> (дата обращения: 16.05.2023).

8. Сайт РИА Новости заблокировали в Азербайджане // РИА Новости [Электронный ресурс]. URL: <https://ria.ru/20220603/sayt-1793059966.html> (дата обращения: 16.05.2023).
9. Информационное агентство «ПензаИнформ» // Сетевое издание СМИ «ПензаИнформ» [Электронный ресурс]. URL: <https://www.penzainform.ru/feedback/editors.xhtml> (дата обращения: 16.05.2023)
10. Статистика сайта «РИА Новости:главные новости часа» // LiveInternet – платформа аналитики. [Электронный ресурс]. URL: https://www.liveinternet.ru/stat/RS_Total/Riaru_Total/summary.html (дата обращения: 16.05.2023)
11. СМИ в России // TADVISER – Информационный портал в области технологий и бизнеса. [Электронный ресурс]. URL: https://www.tadviser.ru/index.php/Статья:СМИ_в_России (дата обращения: 16.05.2023)
12. Desktop vs Mobile vs Tablet Market Share Worldwide // StatCounter. [Электронный ресурс]. URL: <https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/2019> (дата обращения: 16.05.2023)
13. Desktop vs Mobile vs Tablet vs Console Market Share Russian Federation // Statcounter. [Электронный ресурс]. URL: <https://gs.statcounter.com/platform-market-share/all/russian-federation/> (дата обращения: 16.05.2023)
14. Mobile Operating System Market Share Worldwide // Statcounter. [Электронный ресурс]. URL: <https://gs.statcounter.com/os-market-share/mobile/worldwide> (дата обращения: 16.05.2023)
15. Higher GDP per capita corresponds with more Apple devices, stats suggest // InsightPortal. [Электронный ресурс]. URL: <https://www.insightportal.io/news/all-news/higher-gdp-per-capita-corresponds-with-more-apple-devices-stats-suggest> (дата обращения: 16.05.2023)

16. Mobile Operating System Market Share India // StatCounter. [Электронный ресурс]. URL: <https://gs.statcounter.com/os-market-share/mobile/india> (дата обращения: 16.05.2023)
17. Ехлаков, Ю.П. Управление программными проектами. Стандарты, модели: Учебное пособие / Ю.П. Ехлаков. – СПб.: Лань, 2019. – 244 с.
18. Иванов, П.В. Управление проектами: Учебное пособие / П.В. Иванов. – Рн/Д: Феникс, 2018. – 102 с.
19. Гонтарева, И.В. Управление проектами / И.В. Гонтарева, Р.М. Нижегородцев, Д.А. Новиков. – М.: КД Либроком, 2018. – 384 с.
20. Ньютон, Р. Управление проектами от А до Я / Р. Ньютон. – М.: Альпина Паблишер, 2018. – 180 с.
21. Вольфсон, Б.Л. Гибкое управление проектами и продуктами / Б.Л. Вольфсон. – СПб.: Питер, 2019. – 208 с.
22. Скороход, С.В. Управление проектами средствами Microsoft Project. Курс лекций. / С.В. Скороход. – М.: Интуит НОУ, 2016. – 319 стр.
23. Куперштейн, В.И. Microsoft Project 2013 в управлении проектами / В.И. Куперштейн. – СПб.:БХВ-Петербург, 2013. – 432 с.
24. Джонсон, Т. Microsoft Project 2016. Шаг за шагом / Т.Джонсон, К.Четфилд. – М., Изд-во «КТК Галактика», 2018. – 566 с.
25. Акатова, Н.А. Планирование и контроль выполнения проектов с использованием MS Project / Н.А. Акатова, М.Г. Коротких. – М.: МИСиС, 2019. – 74 с.
26. Кролл П. Rational Unified Process – это легко. Руководство по RUP для практиков: Пер. с англ. / М.: КУДИЦ-ОБРАЗ, 2004. – 432 с.
27. Джонс, К. Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies / К. Джонс. – New York: McGraw-Hill, 2010. - 560 с.
28. Рамбо, Дж. UML 2.0. Объектно-ориентированное моделирование и разработка / Дж. Рамбо, М. Блаха. – 2-е изд. – СПб.: Питер, 2007. – 544 с.

29. Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт ; пер. с англ. – 8-е изд. – Москва: Издательский дом «Вильямс», 2005. – 1328 с.
30. Хайкин С. Нейронные сети: полный курс, 2-е издание. / Хайкин С. ; пер. с англ. Москва: Издательский дом «Вильямс», 2006. 1104 с.
31. How to Design Relational Database with ERD // Visual Paradigm. [Электронный ресурс]. URL: <https://www.visual-paradigm.com/tutorials/databasedesign.jsp> (дата обращения: 21.05.2023)
32. Учебник. Создание веб-API с помощью ASP.NET Core // Microsoft Learn. [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/ASPNET/core/tutorials/first-web-api?view=aspnetcore-7.0&tabs=visual-studio> (дата обращения: 22.05.2023)
33. Руководство по ASP.NET Web API 2 // METANIT.COM. [Электронный ресурс]. URL: https://metanit.com/sharp/aspnet_webapi/ (дата обращения: 22.05.2023)
34. Смит Дж. П. Entity Framework Core в действии / пер. с англ. Д. А. Беликова. – Москва: ДМК Пресс, 2022. – 690 с.

Приложение А
ИСХОДНЫЙ КОД СКРИПТОВ БАЗЫ ДАННЫХ
(рекомендуемое)

Листинг А.1 – Исходный код скрипта создания таблиц базы данных

```

CREATE TABLE Users
(
    ID int IDENTITY NOT NULL,
    Login varchar(128) NOT NULL,
    Hash binary(32) NOT NULL,
    Salt varchar(32) NOT NULL,
    Nickname varchar(128) NOT NULL,
    PRIMARY KEY (ID)
);

CREATE TABLE Profile
(
    ID int NOT NULL,
    Description varchar(max) NULL,
    ImageProfile binary NULL,
    PRIMARY KEY (ID)
);

CREATE TABLE Subscriptions
(
    ID int IDENTITY NOT NULL,
    IDProvider int NOT NULL,
    IDSubscriber int NOT NULL,
    PRIMARY KEY (ID)
);

CREATE TABLE Moderator
(
    ID int IDENTITY NOT NULL,
    KEYHash binary(32) NOT NULL,
    PRIMARY KEY (ID)
);

CREATE TABLE ReasonForBan
(
    ID int IDENTITY NOT NULL,
    Name varchar(128) NOT NULL,
    PeriodHours int NOT NULL,
    PRIMARY KEY (ID)
);

CREATE TABLE Ban
(
    ID int IDENTITY NOT NULL,
    IDModerator int NOT NULL,
    IDUser int NOT NULL,
    IDReasonForBan int NOT NULL,
    StartTimeOfBan date NOT NULL,
    PRIMARY KEY (ID)
);

CREATE TABLE Article
(
    ID int IDENTITY NOT NULL,
    Title varchar(255) NOT NULL,
    [Date] date NOT NULL,
    IDUser int NOT NULL,
    IDPosition int NOT NULL,
    IsChecked bit NOT NULL,
    IsPublished bit NOT NULL,

```

```

        PRIMARY KEY (ID)
    );

CREATE TABLE Comments
(
    ID int IDENTITY NOT NULL,
    IDArticle int NOT NULL,
    IDUserOfComment int NOT NULL,
    TextOfComment varchar(255) NOT NULL,
    PRIMARY KEY (ID)
);

CREATE TABLE EvaluationArticle
(
    ID int IDENTITY NOT NULL,
    IDArticle int NOT NULL,
    IDUser int NOT NULL,
    IsLike bit NOT NULL,
    PRIMARY KEY (ID)
);

CREATE TABLE Position
(
    ID int IDENTITY NOT NULL,
    Latitude float(10) NOT NULL,
    Longitude float(10) NOT NULL,
    Name varchar(max) NOT NULL,
    PRIMARY KEY (ID)
);

CREATE TABLE Paragraph
(
    ID int IDENTITY NOT NULL,
    NumberInArticle int NOT NULL,
    TextParagraph varchar(255) NULL,
    ImageParagraph binary NULL,
    IDArticle int NOT NULL,
    PRIMARY KEY (ID)
);

ALTER TABLE Entity_Entity ADD CONSTRAINT FKEntity_Ent205330 FOREIGN KEY (
REFERENCES Entity ());
ALTER TABLE Entity_Entity ADD CONSTRAINT FKEntity_Ent205331 FOREIGN KEY (
REFERENCES Entity ());
ALTER TABLE Entity ADD CONSTRAINT FKEntity587403 FOREIGN KEY ( ) REFERENCES
Entity2 ( );
ALTER TABLE Profile ADD CONSTRAINT FKProfile259588 FOREIGN KEY (ID)
REFERENCES Users (ID);
ALTER TABLE Subscriptions ADD CONSTRAINT FKSubscripti71001 FOREIGN KEY
(IDProvider) REFERENCES Users (ID);
ALTER TABLE Subscriptions ADD CONSTRAINT FKSubscripti422109 FOREIGN KEY
(IDSubscriber) REFERENCES Users (ID);
ALTER TABLE Ban ADD CONSTRAINT FKBan501073 FOREIGN KEY (IDUser) REFERENCES
Users (ID);
ALTER TABLE Ban ADD CONSTRAINT FKBan118824 FOREIGN KEY (IDModerator)
REFERENCES Moderator (ID);
ALTER TABLE Ban ADD CONSTRAINT FKBan834641 FOREIGN KEY (IDReasonForBan)
REFERENCES ReasonForBan (ID);
ALTER TABLE Article ADD CONSTRAINT FKArticle291270 FOREIGN KEY (IDUser)
REFERENCES Users (ID);
ALTER TABLE Article ADD CONSTRAINT FKArticle228785 FOREIGN KEY (IDPosition)
REFERENCES Position (ID);
ALTER TABLE Paragraph ADD CONSTRAINT FKParagraph672488 FOREIGN KEY
(IDArticle) REFERENCES Article (ID);

```

```
ALTER TABLE Comments ADD CONSTRAINT FKComments203045 FOREIGN KEY
(IDUserOfComment) REFERENCES Users (ID);
ALTER TABLE Comments ADD CONSTRAINT FKComments611306 FOREIGN KEY (IDArticle)
REFERENCES Article (ID);
ALTER TABLE EvaluationArticle ADD CONSTRAINT FKEvaluation461864 FOREIGN KEY
(IDArticle) REFERENCES Article (ID);
ALTER TABLE EvaluationArticle ADD CONSTRAINT FKEvaluation482494 FOREIGN KEY
(IDUser) REFERENCES Users (ID);
```

Приложение Б
ИСХОДНЫЙ КОД СЕРВЕРНОЙ ЧАСТИ
(рекомендуемое)

Листинг Б.1 – Файл ApplicationContext.cs

```

using Microsoft.EntityFrameworkCore;

namespace WebServerForPortalNews.Models
{
    public class ApplicationContext : DbContext
    {
        public virtual DbSet<User> users { get; set; }
        public virtual DbSet<Position> adresses { get; set; }
        public virtual DbSet<Article> articles { get; set; }
        public virtual DbSet<Paragraph> paragraphs { get; set; }
        public virtual DbSet<Profile> profiles { get; set; }
        public virtual DbSet<Subscriptions> subscriptions { get; set; }
        public virtual DbSet<EvaluationArticle> evaluationArticles { get;
set; }
        public virtual DbSet<Comment> comments { get; set; }
        public virtual DbSet<Moderator> moderators { get; set; }
        public virtual DbSet<Ban> bans { get; set; }
        public virtual DbSet<ReasonForBan> reasonForBans { get; set; }
        public ApplicationContext(DbContextOptions<ApplicationContext>
optionsBuilder) : base(optionsBuilder)
        {
            Update();
        }
        public void Update()
        {
            users.Load();
            adresses.Load();
            articles.Load();
            paragraphs.Load();
            profiles.Load();
            subscriptions.Load();
            evaluationArticles.Load();
            comments.Load();
            moderators.Load();
            bans.Load();
            reasonForBans.Load();
        }
        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {
            optionsBuilder.UseSqlServer(@"Data Source=DESKTOP-RVN76E7;Initial
Catalog=NewsPortal;User Id=dpers; Integrated Security=True;
TrustServerCertificate=True;
MultipleActiveResultSets=True;Trusted_Connection=True;");
        }
    }
}

```

Листинг Б.2 – Файл Article.cs

```

using System;
using System.ComponentModel.DataAnnotations.Schema;

namespace WebServerForPortalNews.Models
{
    [Table("Article")]
    public class Article: BaseModel
    {
        [Column("Title")]

```

```

        public string Title { get; set; }
        [Column("Date")]
        public DateTime Date { get; set; }
        [Column("IDUser"), ForeignKey("User")]
        public int IDUser { get; set; }
        public virtual User User { get; set; }
        [Column("IDPosition"), ForeignKey("Position")]
        public int IDPosition { get; set; }
        public virtual Position Position { get; set; }
        [Column("IsChecked")]
        public bool IsChecked { get; set; }
        [Column("IsPublished")]
        public bool IsPublished { get; set; }
    }
}

```

Листинг Б.3 – Файл Ban.cs

```

using System;
using System.ComponentModel.DataAnnotations.Schema;

namespace WebServerForPortalNews.Models
{
    [Table("Ban")]
    public class Ban:BaseModel
    {
        [Column("IDModerator"), ForeignKey("Moderator")]
        public int IDModerator { get; set; }
        public virtual Moderator Moderator { get; set; }
        [Column("IDUser"), ForeignKey("User")]
        public int IDUser { get; set; }
        public virtual User User { get; set; }
        [Column("IDReasonForBan"), ForeignKey("ReasonForBan")]
        public int IDReasonForBan { get; set; }
        public virtual ReasonForBan ReasonForBan { get; set; }
        [Column("StartTimeOfBan")]
        public DateTime StartTimeOfBan { get; set; }
    }
}

```

Листинг Б.4 – Файл BaseModel.cs

```

using System.ComponentModel.DataAnnotations.Schema;

namespace WebServerForPortalNews.Models
{
    public class BaseModel
    {
        [Column("ID")]
        public int ID { get; set; }
    }
}

```

Листинг Б.5 – Файл Comment.cs

```

using System.ComponentModel.DataAnnotations.Schema;

namespace WebServerForPortalNews.Models
{
    [Table("Comments")]

```

```

public class Comment:BaseModel
{
    [Column("TextOfComment")]
    public string TextOfComment { get; set; }
    [Column("IDUserOfComment"), ForeignKey("User")]
    public int IDUserOfComment { get; set; }
    public virtual User User { get; set; }
    [Column("IDArticle"), ForeignKey("Article")]
    public int IDArticle { get; set; }
    public virtual Article Article { get; set; }
}
}

```

Листинг Б.6 – Файл EvaluationArticle.cs

```

using System.ComponentModel.DataAnnotations.Schema;

namespace WebServerForPortalNews.Models
{
    [Table("EvaluationArticle")]
    public class EvaluationArticle:BaseModel
    {
        [Column("IsLike")]
        public bool IsLike { get; set;}
        [Column("IDUser"), ForeignKey("User")]
        public int IDUser { get; set; }
        public virtual User User { get; set; }
        [Column("IDArticle"), ForeignKey("Article")]
        public int IDArticle { get; set; }
        public virtual Article Article { get; set; }
    }
}

```

Листинг Б.7 – Файл Moderator.cs

```

using System.ComponentModel.DataAnnotations.Schema;

namespace WebServerForPortalNews.Models
{
    [Table("Moderator")]
    public class Moderator:BaseModel
    {
        [Column("KEYHash")]
        public byte[] KEYHash { get; set; }
    }
}

```

Листинг Б.8 – Файл Paragraph.cs

```

using System.ComponentModel.DataAnnotations.Schema;

namespace WebServerForPortalNews.Models
{
    [Table("Paragraph")]
    public class Paragraph:BaseModel
    {
        [Column("NumberInArticle")]
        public byte NumberInArticle { get; set; }
        [Column("TextParagraph")]
        public string? TextParagraph { get; set; }
    }
}

```

```

        [Column("ImageParagraph")]
        public byte[]? ImageParagraph { get; set; }
        [Column("IDArticle"), ForeignKey("Article")]
        public int IDArticle { get; set; }
        public virtual Article Article { get; set; }
    }
}

```

Листинг Б.9 – Файл Position.cs

```

using System.ComponentModel.DataAnnotations.Schema;

namespace WebServerForPortalNews.Models
{
    [Table("Position")]
    public class Position:BaseModel
    {
        [Column("Name")]
        public string Name { get; set; }
        [Column("Latitude")]
        public double Latitude { get; set; }
        [Column("Longitude")]
        public double Longitude { get; set; }
    }
}

```

Листинг Б.10 – Файл Profile.cs

```

using System.ComponentModel.DataAnnotations.Schema;

namespace WebServerForPortalNews.Models
{
    [Table("Profile")]
    public class Profile:BaseModel
    {
        [Column("Description")]
        public string? Description { get; set; }
        [Column("ImageProfile")]
        public byte[]? ImageProfile { get; set; }
    }
}

```

Листинг Б.11 – Файл ReasonForBan.cs

```

using System.ComponentModel.DataAnnotations.Schema;

namespace WebServerForPortalNews.Models
{
    [Table("ReasonForBan")]
    public class ReasonForBan:BaseModel
    {
        [Column("Name")]
        public string Name { get; set; }
        [Column("PeriodHours")]
        public int PeriodHours { get; set; }}}
}

```

Листинг Б.12 – Файл Subscriptions.cs

```
using System.ComponentModel.DataAnnotations.Schema;

namespace WebServerForPortalNews.Models
{
    [Table("Subscriptions")]
    public class Subscriptions:BaseModel
    {
        [Column("IDProvider"), ForeignKey("Provider")]
        public int IDProvider { get; set; }
        public virtual User Provider { get; set; }
        [Column("IDSubscriber"), ForeignKey("Subscriber")]
        public int IDSubscriber { get; set; }
        public virtual User Subscriber { get; set; }
    }
}
```

Листинг Б.13 – Файл User.cs

```
using System.ComponentModel.DataAnnotations.Schema;

namespace WebServerForPortalNews.Models
{
    [Table("Users")]
    public class User:BaseModel
    {
        [Column("Login")]
        public string Login { get; set; }
        [Column("Hash")]
        public byte[] Hash { get; set; }
        [Column("Salt")]
        public string Salt { get; set; }
        [Column("Nickname")]
        public string Nickname { get; set; }
    }
}
```

Листинг Б.14 – Файл ArticlesController.cs

```
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Security.Cryptography;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using WebServerForPortalNews.Models;
using WebServerForPortalNews.Models.HelperModels;

namespace WebServerForPortalNews.Controllers
{
    [ApiController]
    public class ArticlesController : ControllerBase
```

```

{
    private readonly ApplicationDbContext db;
    private Random random;
    public ArticlesController(ApplicationDbContext context)
    {
        db = context;
        random = new Random();
    }

    [HttpGet]
    [Route("GetNewsVerification")]
    public async Task<ActionResult> GetNewsVerification([FromQuery(Name =
"KEY")] string KEY)
    {
        try
        {
            HashAlgorithm algorithm = new SHA256Managed();
            var hashKey =
algorithm.ComputeHash(Encoding.Unicode.GetBytes(KEY));

            if (!db.moderators.Any(x=>x.KEYHash.SequenceEqual(hashKey)))
                return BadRequest();
            var articlesForVerification = db.articles.Where(x =>
x.IsChecked == false).ToList();
            if (articlesForVerification.Count == 0)
                return BadRequest();
            var articleForVerification =
articlesForVerification[random.Next(0, articlesForVerification.Count)];
            var paragraphs = db.paragraphs.Where(x => x.IDArticle ==
articleForVerification.ID).ToList();
            var JsonString = JsonConvert.SerializeObject(paragraphs);
            return Ok(JsonString);
        }
        catch (Exception ex)
        {
            return BadRequest();
        }
    }

    [HttpPut]
    [Route("AssignSolutionNews")]
    public async Task<ActionResult> AssignSolutionNews([FromBody]
DecisionNews decisionNews)
    {
        try
        {
            HashAlgorithm algorithm = new SHA256Managed();
            var hashKey =
algorithm.ComputeHash(Encoding.Unicode.GetBytes(decisionNews.KeyOfModerator))
;

            var moderator = db.moderators.ToList();

            if (!db.moderators.Any(x =>
x.KEYHash.SequenceEqual(hashKey)))
                return BadRequest();

            var article = db.articles.FirstOrDefault(x => x.ID ==
decisionNews.IDArticle);
            article.IsChecked = true;
            article.IsPublished = decisionNews.IsAcceptable;
            db.SaveChanges();
            return Ok();
        }
    }
}

```

```

        catch
        {
            return BadRequest();
        }
    }

    [HttpGet]
    [Route("NewsOfMap")]
    public async Task<ActionResult> GetAllArticles()
    {
        try
        {
            //Получаем актуальные статьи

            var ListArticles = db.articles.OrderByDescending(x =>
x.Date).Where(x=>x.IsChecked==true &&
x.IsPublished==true).Take(1000).ToList();
            var ActualArticlesGroup = ListArticles.GroupBy(x =>
x.IDPosition);
            var ActualArticlesSelect = ActualArticlesGroup.Select(x =>
x.OrderByDescending(x => x.Date).FirstOrDefault());
            var ActualArticles = ActualArticlesSelect.Take(100).ToList();

            List<PlaceHelper> placeHelpers = new List<PlaceHelper>();
            foreach (var article in ActualArticles)
            {
                if (article.Position == null)
                    continue;
                placeHelpers.Add(new PlaceHelper()
                {
                    IDArticle = article.ID,
                    Adress = article.Position.Name,
                    Name = article.Title,
                    Latitude = article.Position.Latitude,
                    Longitude = article.Position.Longitude,
                    PublishedDate = article.Date
                });
            }

            var JsonString = JsonConvert.SerializeObject(placeHelpers);
            return Ok(JsonString);
        }
        catch (Exception e)
        {
            return BadRequest();
        }
    }

    [HttpGet]
    [Route("GetParagraph/{ID}/{Number}")]
    public async Task<ActionResult> GetDataParagraph(int ID,int Number)
    {
        var paragraph = db.paragraphs.Where(x => x.IDArticle ==
ID).FirstOrDefault(x => x.NumberInArticle == Number);
        if (paragraph == null)
            return BadRequest();
        DataParagraph dataParagraph = new DataParagraph { TextParagraph =
paragraph.TextParagraph, ImageParagraph = paragraph.ImageParagraph };
        var json = JsonConvert.SerializeObject(dataParagraph);
        return Ok(json);
    }

    [HttpGet]
    [Route("NewsOfSubscribers/{IDUser}/{CurrentDate}")]

```

```

        public async Task<ActionResult> GetNewsOfSubscribers(int IDUser,
string CurrentDate)
        {
            var Date = DateTime.ParseExact(CurrentDate, "MM.dd.yyyy
HH:mm:ss",
System.Globalization.CultureInfo.InvariantCulture);

            var providers = db.subscriptions.Where(x => x.IDSubscriber ==
IDUser);

            var articles = db.articles.OrderBy(x => x.Date).Where(x=>x.Date>
Date && providers.FirstOrDefault(y=>y.IDProvider==x.IDUser) !=null).Take(15);

            if(articles.Count()==0)
            {
                return BadRequest();
            }
            var articlesPreview = new List<ArticlePreview>();
            foreach(var article in articles)
            {
                var ArticlePreview = new ArticlePreview
                {
                    ID = article.ID,
                    Adress = article.Position.Name,
                    NicknameAuthor = article.User.Nickname,
                    dateTime = article.Date,
                    Name = article.Title,
                    ImageAuthor = db.profiles.FirstOrDefault(x => x.ID ==
article.IDUser).ImageProfile,
                    IDUser = article.IDUser
                };
                articlesPreview.Add(ArticlePreview);
            }
            var json = JsonConvert.SerializeObject(articlesPreview);

            return Ok(json);
        }

[HttpGet]
[Route("GetArticlesUser/{ID}")]
public async Task<ActionResult> GetArticlesOfUser(int ID)
{
    try
    {
        List<PlaceHelper> articles = new List<PlaceHelper>();
        var articlesFromDB = db.articles.Where(x => x.IDUser == ID);
        foreach(var article in articlesFromDB)
        {
            articles.Add(new PlaceHelper {
                IDArticle = article.ID,
                Adress = article.Position.Name,
                Latitude = article.Position.Latitude,
                Longitude = article.Position.Longitude,
                Name = article.Title,
                PublishedDate = article.Date
            });
        }
        var json = JsonConvert.SerializeObject(articles);
        return Ok(json);
    }
    catch (Exception e)
    {

```

```

        return BadRequest("Произошла ошибка на сборе данных,
        обратитесь в поддержку");
    }
}

[HttpGet]
[Route("GetArticle/{ID}")]
public async Task<ActionResult> GetArticle(int ID)
{
    try
    {
        var Article = db.articles.FirstOrDefault(x => x.ID == ID);

        var CountParagraphs = db.paragraphs.Where(x => x.IDArticle ==
        Article.ID).Select(x => x.NumberInArticle).Max();

        var News = new PinArticle()
        {
            IDUser = Article.IDUser,
            Adress = Article.Position,
            Title = Article.Title,
            CountParagraphs = CountParagraphs
        };
        var jsonNews = JsonConvert.SerializeObject(News);
        return Ok(jsonNews);
    }
    catch (Exception e)
    {
        return BadRequest("Данная статья была не найдена");
    }
}

[HttpDelete]
[Route("DeletaArticle/{ID}")]
public async Task<ActionResult> DeleteArticle(int ID)
{
    try
    {
        var article = db.articles.FirstOrDefault(x => x.ID == ID);
        if (article == null)
            throw new Exception();
        var paragraphs = db.paragraphs.Where(x => x.IDArticle ==
        ID).ToList();
        foreach (var paragraph in paragraphs)
        {
            db.paragraphs.Remove(paragraph);
        }
        db.articles.Remove(article);
        db.SaveChanges();
        return Ok();
    }
    catch (Exception e)
    {
        return BadRequest("Данная статья была не найдена");
    }
}

[HttpPut]
[Route("UpdateArticle")]
public async Task<ActionResult> UpdateArticle()
{
    try
    {
        var Reader = new StreamReader(Request.Body);
        var content = await Reader.ReadToEndAsync();
    }
}

```

```

        var NewsFromJson =
JsonConvert.DeserializeObject<News>(content);

        if (NewsFromJson.ID == -1)
            return BadRequest("Ошибка сервера");
        var article = db.articles.FirstOrDefault(x => x.ID ==
NewsFromJson.ID);
        if(article==null)
            return BadRequest("Статья перестала существовать");
        var address = db.adresses.FirstOrDefault(x => x.Latitude ==
NewsFromJson.Adress.Latitude && x.Longitude ==
NewsFromJson.Adress.Longitude);
        if (address != null)
            NewsFromJson.Adress = address;
        article.IDPosition = NewsFromJson.Adress.ID;
        article.Position = NewsFromJson.Adress;
        article.Title = NewsFromJson.Title;
        article.IDUser = NewsFromJson.IDUser;

        var paragraphs = db.paragraphs.Where(x => x.IDArticle ==
article.ID);

        foreach(var paragraph in paragraphs)
        {
            db.paragraphs.Remove(paragraph);
        }
        db.SaveChanges();
        paragraphs.ToList().Clear();

        for (int i = 0; i < NewsFromJson.ParagraphTexts.Count; i++)
        {
            if (i > 255)
                break;
            var Paragraph = new Paragraph();
            Paragraph.Article = article;
            Paragraph.IDArticle = article.ID;
            Paragraph.NumberInArticle = (byte)(i + 1);
            Paragraph.TextParagraph = NewsFromJson.ParagraphTexts[i];
            if (NewsFromJson.ImagesBytes.Count > i)
                Paragraph.ImageParagraph =
NewsFromJson.ImagesBytes[i];
            else
                Paragraph.ImageParagraph = null;
            db.paragraphs.Add(Paragraph);
        }
        db.SaveChanges();
    }
    catch (Exception e)
    {
        return BadRequest("Ошибка сервера");
    }
    return Ok("Статья успешна изменена");
}
[HttpPost]
[Route("AddNewArticle")]
public async Task<ActionResult> AddNewArticle()
{
    try
    {
        var Reader = new StreamReader(Request.Body);
        var content = await Reader.ReadToEndAsync();
        var json = JsonConvert.DeserializeObject<News>(content);

        //Ищем добавлен ли уже был такой адрес

```

```

        var adressInBaseData = db.adresses.Where(x => x.Latitude ==
json.Address.Latitude && x.Longitude ==
json.Address.Longitude).FirstOrDefault();

        if (adressInBaseData != null)
            json.Address = adressInBaseData;
        else
        {
            db.adresses.Add(json.Address);
            db.SaveChanges();
        }

        Article article = new Article();
        article.IDPosition = json.Address.ID;
        article.Position = json.Address;
        article.Title = json.Title;
        article.IDUser = json.IDUser;
        article.Date = DateTime.Now;

        db.articles.Add(article);
        db.SaveChanges();

        for (int i = 0; i < json.ParagraphTexts.Count; i++)
        {
            if (i > 255)
                break;
            var Paragraph = new Paragraph();
            Paragraph.Article = article;
            Paragraph.IDArticle = article.ID;
            Paragraph.NumberInArticle = (byte)(i + 1);
            Paragraph.TextParagraph = json.ParagraphTexts[i];
            var resultText=await CheckText(json.ParagraphTexts[i]);
            if(resultText<0.2)
            {
                return BadRequest("В абзаце " + (i + 1).ToString() +
" замечен недопустимый контент, отредактируйте его. Не пытайтесь обойти
систему, иначе ваш аккаунт будет заблокирован");
            }
            if (json.ImagesBytes.Count > i)
                Paragraph.ImageParagraph = json.ImagesBytes[i];
            else
                Paragraph.ImageParagraph = null;
            db.paragraphs.Add(Paragraph);
            db.SaveChanges();
        }
        db.Update();
    }
    catch (Exception e)
    {
        return BadRequest("Ошибка сервера");
    }
    return Ok("Статья успешно добавлена");
}
private async Task<double> CheckText(string Text)
{
    string currentDirectory =
System.IO.Path.GetDirectoryName(Assembly.GetEntryAssembly().Location);
    string pythonScript = System.IO.Path.Combine(currentDirectory,
"Work.py");
    char[] splitter = { '\r' };

    Process proc = new Process();
    proc.StartInfo.FileName = "python.exe";

```

```

        proc.StartInfo.WorkingDirectory = currentDirectory; // Установка
рабочего каталога для процесса Python
        proc.StartInfo.RedirectStandardOutput = true;
        proc.StartInfo.UseShellExecute = false;
        proc.StartInfo.StandardOutputEncoding = Encoding.Default;

        // Выполнение скрипта Python с использованием полного пути
        proc.StartInfo.Arguments = string.Concat(pythonScript, " '",
Text, "'");
        proc.Start();

        StreamReader sReader = proc.StandardOutput;
        string[] output = sReader.ReadToEnd().Split(splitter);

        foreach (var stringResult in output)
        {
            try
            {
                var CurrentString = stringResult.Trim();
                CultureInfo temp_culture =
Thread.CurrentThread.CurrentCulture;
                Thread.CurrentThread.CurrentCulture =
CultureInfo.CreateSpecificCulture("en-US");
                double d = double.Parse(CurrentString);
                Thread.CurrentThread.CurrentCulture = temp_culture;
                return d;
            }
            catch
            {
                continue;
            }
        }

        proc.WaitForExit();
        return 0;
    }
}

```

Листинг Б.15 – Файл CommentsController.cs

```

using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using WebServerForPortalNews.Models;

namespace WebServerForPortalNews.Controllers
{
    [ApiController]
    public class CommentsController : ControllerBase
    {
        private readonly ApplicationContext db;
        public CommentsController(ApplicationContext context)
        {
            db = context;
        }
        [HttpGet]
        [Route("GetComments/{IDArticle}")]
        public async Task<ActionResult> GetComments(int IDArticle)

```

```

        {
            try
            {
                var comments = db.comments.Where(x => x.IDArticle ==
IDArticle).ToList();
                var json = JsonConvert.SerializeObject(comments);
                return Ok(json);
            }
            catch
            {
                return BadRequest();
            }
        }
        [HttpPost]
        [Route("AddComment")]
        public async Task<ActionResult> AddComment()
        {
            try
            {
                var Reader = new StreamReader(Request.Body);
                var content = await Reader.ReadToEndAsync();
                var Comment =
JsonConvert.DeserializeObject<Comment>(content);
                db.comments.Add(Comment);
                db.SaveChanges();
                return Ok();
            }
            catch
            {
                return BadRequest();
            }
        }
    }
}

```

Листинг Б.16 – Файл EvaluationArticleController.cs

```

using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using WebServerForPortalNews.Models;

namespace WebServerForPortalNews.Controllers
{
    [ApiController]
    public class EvaluationArticleController : ControllerBase
    {
        private readonly ApplicationContext db;
        public EvaluationArticleController(ApplicationContext context)
        {
            db = context;
        }
        [HttpGet]
        [Route("GetEvaluations/{IDArticle}")]
        public async Task<ActionResult> GetEvaluations(int IDArticle)
        {
            try
            {
                var Evaluations = db.evaluationArticles.Where(x =>
x.IDArticle == IDArticle).ToList();

```

```

        var json = JsonConvert.SerializeObject(Evaluations);
        return Ok(json);
    }
    catch
    {
        return BadRequest();
    }
}

[HttpPost]
[Route("SetEvaluation")]
public async Task<ActionResult> SetEvaluation()
{
    try
    {
        var Reader = new StreamReader(Request.Body);
        var content = await Reader.ReadToEndAsync();
        var Evaluation =
JsonConvert.DeserializeObject<EvaluationArticle>(content);
        var checkEvaluation = db.evaluationArticles.FirstOrDefault(x
=> x.IDUser == Evaluation.IDUser && x.IDArticle == Evaluation.IDArticle);
        if (checkEvaluation != null)
            checkEvaluation.IsLike = Evaluation.IsLike;
        else
            db.evaluationArticles.Add(Evaluation);
        db.SaveChanges();
        return Ok();
    }
    catch
    {
        return BadRequest();
    }
}

[HttpDelete]
[Route("DeleteEvaluation/{IDArticle}/{IDUser}")]
public async Task<ActionResult> DeleteEvaluation(int IDArticle, int
IDUser)
{
    try
    {
        var evaluation = db.evaluationArticles.FirstOrDefault(x =>
x.IDArticle == IDArticle && x.IDUser == IDUser);
        if (evaluation != null)
            db.evaluationArticles.Remove(evaluation);
        db.SaveChanges();
        return Ok();
    }
    catch
    {
        return BadRequest();
    }
}
}
}

```

Листинг Б.17 – Файл ModeratorController.cs

```

using Microsoft.AspNetCore.Mvc;
using System;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;

```

```

using WebServerForPortalNews.Models;

namespace WebServerForPortalNews.Controllers
{
    [ApiController]
    public class ModeratorController : ControllerBase
    {
        private readonly ApplicationDbContext db;
        private Random random;
        private const int LENGTH_RANDOM_STRING = 32;
        public ModeratorController(ApplicationDbContext context)
        {
            db = context;
            random = new Random();
        }

        [HttpGet]
        [Route("GetKey")]
        public async Task<ActionResult> GetKey([FromQuery(Name = "KEY")]
string KEY)
        {
            try
            {
                HashAlgorithm algorithm = new SHA256Managed();
                var hashKey =
algorithm.ComputeHash(Encoding.Unicode.GetBytes(KEY));

                if (!db.moderators.Any(x =>
x.KEYHash.SequenceEqual(hashKey)))
                    return BadRequest();

                string NewKey = RandomString(LENGTH_RANDOM_STRING);

                var hashNewKey =
algorithm.ComputeHash(Encoding.Unicode.GetBytes(NewKey));

                db.moderators.Add(new Moderator()
                {
                    KEYHash = hashNewKey
                });
                db.SaveChanges();

                return Ok(NewKey);
            }
            catch
            {
                return BadRequest();
            }
        }

        private string RandomString(int length)
        {
            const string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
            return new string(Enumerable.Repeat(chars, length)
                .Select(s => s[random.Next(s.Length)]).ToArray());
        }
    }
}

```

Листинг Б.18 – Файл ProfilesController.cs

```

using Microsoft.AspNetCore.Mvc;
using System.Linq;
using System.Threading.Tasks;
using WebServerForPortalNews.Models;
using System.IO;
using Newtonsoft.Json;
using WebServerForPortalNews.Models.HelperModels;

namespace WebServerForPortalNews.Controllers
{
    [ApiController]
    public class ProfilesController : ControllerBase
    {
        private readonly ApplicationDbContext db;
        public ProfilesController(ApplicationDbContext context)
        {
            db = context;
        }
        [HttpGet]
        [Route("GetProfile/{ID}")]
        public async Task<ActionResult> GetProfile(int ID)
        {
            try
            {
                var profileuser = db.profiles.FirstOrDefault(x => x.ID ==
ID);
                var subscribersCount = db.subscriptions.Where(x =>
x.IDProvider == ID).Count();
                var articlesCount = db.articles.Where(x => x.IDUser ==
ID).Count();
                var Nickname = db.users.FirstOrDefault(x => x.ID ==
ID).Nickname;
                var json = JsonConvert.SerializeObject(new ProfileData
                {
                    CountArticles = articlesCount,
                    Description = profileuser.Description,
                    ImageProfile = profileuser.ImageProfile,
                    Status = 50,
                    Subscribers = subscribersCount,
                    Nickname = Nickname
                });
                return Ok(json);
            }
            catch
            {
                return BadRequest();
            }
        }

        [HttpPut]
        [Route("UpdateImageProfile")]
        public async Task<ActionResult> UpdateImageProfile()
        {
            var StreamReader = new StreamReader(Request.Body);
            var jsonString = await StreamReader.ReadToEndAsync();
            try
            {
                var Data =
JsonConvert.DeserializeObject<ProfileImage>(jsonString);

```

```

        var profileuser = db.profiles.FirstOrDefault(x => x.ID ==
Data.ID);
        profileuser.ImageProfile = Data.Image;
        db.SaveChanges();
        return Ok();
    }
    catch
    {
        return BadRequest();
    }
}
}
}

```

Листинг Б.19 – Файл ReasonForBanController.cs

```

using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using System;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using WebServerForPortalNews.Models;

namespace WebServerForPortalNews.Controllers
{
    [ApiController]
    public class ReasonForBanController : ControllerBase
    {
        private readonly ApplicationDbContext db;
        public ReasonForBanController(ApplicationDbContext context)
        {
            db = context;
        }

        [HttpGet]
        [Route("GetReasonsForBan")]
        public async Task<ActionResult> GetKey([FromQuery(Name = "KEY")]
string KEY)
        {
            try
            {
                HashAlgorithm algorithm = new SHA256Managed();
                var hashKey =
algorithm.ComputeHash(Encoding.Unicode.GetBytes(KEY));

                if (!db.moderators.Any(x =>
x.KEYHash.SequenceEqual(hashKey)))
                    return BadRequest();

                var ReasonsForBan = db.reasonForBans.ToList();
                var JsonObject = JsonConvert.SerializeObject(ReasonsForBan);

                return Ok(JsonObject);
            }
            catch
            {
                return BadRequest();
            }
        }
    }
}

```

```

    }
}

```

Листинг Б.20 – Файл SubscriptionsController.cs

```

using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using System;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using WebServerForPortalNews.Models;

namespace WebServerForPortalNews.Controllers
{
    [ApiController]
    public class SubscriptionsController : ControllerBase
    {
        private readonly ApplicationDbContext db;
        public SubscriptionsController(ApplicationContext context)
        {
            db = context;
        }

        [HttpGet]
        [Route("GetSubscribers/{IDUser}")]
        public async Task<ActionResult> GetSubscribers(int IDUser)
        {
            try
            {
                var subscribers = db.subscriptions.Where(x => x.IDSubscriber
== IDUser).ToList();
                var json = JsonConvert.SerializeObject(subscribers);
                return Ok(json);
            }
            catch
            {
                return BadRequest();
            }
        }

        [HttpGet]
        [Route("GetSubscription/{IDAuthor}/{IDUser}")]
        public async Task<ActionResult> GetSubscription(int IDAuthor, int
IDUser)
        {
            try
            {
                var subscription = db.subscriptions.FirstOrDefault(x =>
x.IDProvider == IDAuthor && x.IDSubscriber == IDUser);
                var json = JsonConvert.SerializeObject(subscription);
                return Ok(json);
            }
            catch
            {
                return BadRequest();
            }
        }

        [HttpPost]
        [Route("Subscribe")]
        public async Task<ActionResult> Subscribe()
        {
            try

```

```

        {
            var Reader = new StreamReader(Request.Body);
            var content = await Reader.ReadToEndAsync();
            var Subscription =
JsonConvert.DeserializeObject<Subscriptions>(content);
            if (db.subscriptions.FirstOrDefault(x => x.IDProvider ==
Subscription.IDProvider && x.IDSubscriber == Subscription.IDSubscriber) !=
null)
                throw new Exception();
            db.subscriptions.Add(Subscription);
            db.SaveChanges();
            return Ok();
        }
        catch
        {
            return BadRequest();
        }
    }
    [HttpDelete]
    [Route("Unsubscribe/{IDAuthor}/{IDUser}")]
    public async Task<ActionResult> Unsubscribe(int IDAuthor, int IDUser)
    {
        try
        {
            var subscription = db.subscriptions.FirstOrDefault(x =>
x.IDProvider == IDAuthor && x.IDSubscriber == IDUser);
            db.subscriptions.Remove(subscription);
            db.SaveChanges();
            return Ok();
        }
        catch
        {
            return BadRequest();
        }
    }
}

```

Листинг Б.21 – Файл UsersController.cs

```

using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using WebServerForPortalNews.Models;
using WebServerForPortalNews.Models.HelperModels;

namespace WebServerForPortalNews.Controllers
{
    [ApiController]
    public class UsersController : ControllerBase
    {
        //Контекст базы данных
        private readonly ApplicationContext db;
        public UsersController(ApplicationContext context)
        {
            db = context;
        }
    }
}

```

```

    }
    //Метод обрабатывающий заявки для регистрации
    [HttpPost]
    [Route("Register")]
    public async Task<ActionResult> PostRegister()
    {
        try
        {
            //Блок, который считывает данные, полученные от клиента
            var Reader = new StreamReader(Request.Body);
            var content = await Reader.ReadToEndAsync();
            var jsonDeserialize =
                JsonConvert.DeserializeObject<Dictionary<string, string>>(content);
            //Если пользователь ввёл не все данные или они не все дошли,
            то данный запрос считается ошибочным
            if (jsonDeserialize==null || jsonDeserialize.Any(x => x.Value
                == null))
            {
                return BadRequest("Не все нужные данные были введены");
            }
            //Ищем пользователя с логином, который пользователь хочет
            зарегистрировать
            string login = jsonDeserialize["login"];
            var user = db.users.FirstOrDefault(x=>x.Login==login);
            //Если пользователь с таким логином уже существует, то
            аналогичного пользователя регистрировать нельзя
            if (user!=null)
            {
                return BadRequest("Пользователь с таким логином уже
                есть");
            }
            //Создаём соль и солёный хеш для нового пользователя
            var Salt = GenerateSalt();
            var Hash =
                GenerateSaltedHash(Encoding.Unicode.GetBytes(jsonDeserialize["password"]),
                Encoding.Unicode.GetBytes(Salt));
            //Создаём пользователя по данным, которые пришли с клиента
            User userForAdded = new User
            {Login=login,Nickname=jsonDeserialize["nickname"],Hash=Hash,Salt=Salt};
            db.users.Add(userForAdded);
            db.SaveChanges();
            //Создаём профиль для нового пользователя
            Profile profile = new Profile { ID = userForAdded.ID,
            Description = null, ImageProfile = null };
            db.profiles.Add(profile);
            db.SaveChanges();
            return Ok(userForAdded.ID.ToString());
        }
        catch
        {
            return BadRequest();
        }
    }

    [HttpPost]
    [Route("Ban")]
    public async Task<ActionResult> IssueBan([FromBody] BanData banData)
    {
        try
        {
            HashAlgorithm algorithm = new SHA256Managed();
            var hashKey =
                algorithm.ComputeHash(Encoding.Unicode.GetBytes(banData.KEY));

```

```

        var moderator = db.moderators.FirstOrDefault(x =>
x.KEYHash.SequenceEqual(hashKey));

        Ban ban = new Ban()
        {
            IDModerator = moderator.ID,
            IDReasonForBan = banData.IDReason,
            IDUser = banData.IDUser,
            StartTimeOfBan = DateTime.Now
        };
        db.bans.Add(ban);
        db.SaveChanges();
        return Ok();
    }
    catch
    {
        return BadRequest();
    }
}

//Метод обрабатывающий заявки для авторизации пользователя
[HttpPost]
[Route("Authorization")]
public async Task<ActionResult> PostAuthorization()
{
    try
    {
        //Считываем данные полученные от пользователя
        var Reader = new StreamReader(Request.Body);
        var content = await Reader.ReadToEndAsync();
        var jsonDeserialize =
JsonConvert.DeserializeObject<Dictionary<string, string>>(content);
        //Если данные введенные пользователем не существуют, то
считаем данный запрос ошибочным
        if (jsonDeserialize==null || jsonDeserialize["login"] == null
|| jsonDeserialize["password"] == null)
        {
            return BadRequest("Не все нужные данные были введены");
        }
        //Ищем пользователя по логину
        string login = jsonDeserialize["login"];

        var user = db.users.FirstOrDefault(x=>x.Login==login);
        if(user==null)
        {
            return BadRequest("Пользователь не найден");
        }
        //Сравниваем хеши в бд и хеш, который генерируется в процессе
выполнения процесса
        var Salt = user.Salt;
        var Hash =
GenerateSaltedHash(Encoding.Unicode.GetBytes(jsonDeserialize["password"]),
Encoding.Unicode.GetBytes(Salt));
        var HashFromBD = user.Hash;

        //Если хеши совпали, то считаем данную авторизацию успешной
        if (Hash.SequenceEqual(HashFromBD))
        {
            var IsActiveBan = db.bans.Where(x => x.IDUser ==
user.ID).Any(x => x.StartTimeOfBan.AddHours(x.ReasonForBan.PeriodHours) >
DateTime.Now);

            if (IsActiveBan)
            {

```

```

        var BanTime = db.bans.Where(x => x.IDUser ==
user.ID).Max(x => x.StartTimeOfBan.AddHours(x.ReasonForBan.PeriodHours));
        return BadRequest("Вы заблокированы до:" +
BanTime.ToString("dd.MM.yyyy HH:mm:ss"));
    }
    return Ok(user.ID.ToString());
}
else
{
    return BadRequest("Мы не смогли вас определить, проверить
данные для входа!");
}
}
catch
{
    return BadRequest();
}
}
//Метод для генерации из пароля и соли соленого хеша
private byte[] GenerateSaltedHash(byte[] plainText, byte[] salt)
{
    HashAlgorithm algorithm = new SHA256Managed();
    byte[] plainTextWithSaltBytes = new byte[plainText.Length +
salt.Length];
    for (int i = 0; i < plainText.Length; i++)
    {
        plainTextWithSaltBytes[i] = plainText[i];
    }
    for (int i = 0; i < salt.Length; i++)
    {
        plainTextWithSaltBytes[plainText.Length + i] = salt[i];
    }
    return algorithm.ComputeHash(plainTextWithSaltBytes);
}
//Метод для генерации случайно соли
private string GenerateSalt()
{
    {
        var bytes = new byte[128 / 8];
        var rng = new RNGCryptoServiceProvider();
        rng.GetBytes(bytes);
        return Convert.ToBase64String(bytes);
    }
}
}

```

Приложение В
ИСХОДНЫЙ КОД МОБИЛЬНОГО ПРИЛОЖЕНИЯ
(рекомендуемое)


```

        matrix.PostRotate(270);
        bitmap = Bitmap.CreateBitmap(bitmap, 0, 0,
bitmap.Width, bitmap.Height, matrix, true);
        break;
    case 3:
        matrix.PostRotate(180);
        bitmap = Bitmap.CreateBitmap(bitmap, 0, 0,
bitmap.Width, bitmap.Height, matrix, true);
        break;
    default:
        break;
}

var imgsrc = ImageSource.FromStream(() =>
{
    MemoryStream ms = new MemoryStream();
    bitmap.Compress(Bitmap.CompressFormat.Jpeg, 100, ms);
    bitmap.Recycle();
    ms.Seek(0L, SeekOrigin.Begin);
    return ms;
});

if(requestCode==0)
    AddNews.currentObject.AddImageCached(imgsrc);
if (requestCode == 1)
    Profile.currentObject.UpdatePhoto(imgsrc);
}

}

public override void OnRequestPermissionsResult(int requestCode,
string[] permissions, [GeneratedEnum] Android.Content.PM.Permission[]
grantResults)
{

Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode,
permissions, grantResults);

        base.OnRequestPermissionsResult(requestCode, permissions,
grantResults);
    }
}
}

```

Листинг В.2 – Файл App.xaml.cs

```

using Android.App;
using Android.Content;
using System;
using Xamarin.Essentials;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Authentication
{
    public partial class App : Xamarin.Forms.Application
    {
        internal static Context ActivityContext { get; private set; }
        internal static Activity ActivityCurrent { get; private set; }
        internal static int IDCurrentUser { get; set; }
        public App(Context context, Activity activity)
        {
            InitializeComponent();

```

```

        ActivityContext = context;
        ActivityCurrent = activity;
        MainPage = new NavigationPage(new Authorization());
        string login = Preferences.Get("Login", string.Empty);
        string password = Preferences.Get("Password", string.Empty);
        if(login!=string.Empty && password!=string.Empty)
        {
            ModelView.ModelViewAuthorization modelView = new
ModelView.ModelViewAuthorization(MainPage);
            modelView.TextLogin = login;
            modelView.TextPassword = password;
            modelView.ExecuteProcedure();
        }
    }

    protected override void OnStart()
    {
    }

    protected override void OnSleep()
    {
    }

    protected override void OnResume()
    {
    }
}

```

Листинг В.3 – Файл App.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<Application xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Authentication.App">
    <Application.Resources>

        </Application.Resources>
    </Application>

```

Листинг В.4 – Файл Register.xaml.cs

```

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Authentication
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class Register : ContentPage
    {
        public Register()
        {
            InitializeComponent();
            BindingContext = new ModelView.ModelViewRegister(this);
        }
    }
}

```

Листинг В.5 – Файл Register.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
              xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
              x:Class="Authentication.Register">
    <ContentPage.Content>
        <StackLayout Spacing="2" VerticalOptions="CenterAndExpand"
HorizontalOptions="CenterAndExpand">
            <Label Text="Регистрация"
                VerticalOptions="Center" HorizontalOptions="Center"
FontSize="18"/>
            <Editor x:Name="EditorNickname" Text="{Binding TextNickname}"
Placeholder="Имя" VerticalOptions="Center" HorizontalOptions="Center"
FontSize="16" WidthRequest="300"></Editor>
            <Editor x:Name="EditorLogin" Text="{Binding TextLogin}"
Placeholder="Логин" VerticalOptions="Center" HorizontalOptions="Center"
FontSize="16" WidthRequest="300"></Editor>
            <Editor x:Name="EditorPassword" Text="{Binding TextPassword}"
Placeholder="Пароль" VerticalOptions="Center" HorizontalOptions="Center"
FontSize="16" WidthRequest="300"></Editor>
            <Button Text="Зарегистрировать" Command="{Binding
ClickCommand}"></Button>
            <Button Text="Авторизироваться" Command="{Binding
CommandChangePage}"></Button>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

Листинг В.6 – Файл ModelViewRegister.cs

```
using Authentication.Helpers;
using Authentication.Interfaces;
using Authentication.Model;
using Authentication.Services;
using System.ComponentModel;
using System.Windows.Input;
using Xamarin.Forms;

namespace Authentication.ModelView
{
    class ModelViewRegister : INotifyPropertyChanged
    {
        private IUserUnitWork UserUnitWork;
        private ValidationRegister validationRegister;
        private Page View;
        private bool IsWaiting;
        public ModelViewRegister(Page view)
        {
            View = view;
            UserUnitWork = new ServiceUser();
            IsWaiting = false;
            validationRegister = new ValidationRegister();
        }
        private string _TextLogin { get; set; }
        public string TextLogin
        {
            get
            {
                return _TextLogin;
            }
        }
    }
}
```

```

        set
        {
            _TextLogin = value;
            OnPropertyChanged("TextLogin");
        }
    }

    private string _TextPassword { get; set; }
    public string TextPassword
    {
        get
        {
            return _TextPassword;
        }
        set
        {
            _TextPassword = value;
            OnPropertyChanged("TextPassword");
        }
    }

    private string _TextNickname { get; set; }
    public string TextNickname
    {
        get
        {
            return _TextNickname;
        }
        set
        {
            _TextNickname = value;
            OnPropertyChanged("TextNickname");
        }
    }

    private ICommand _clickCommand;
    public ICommand ClickCommand
    {
        get
        {
            return _clickCommand ?? (_clickCommand = new DelegateCommand(
                (parameters) => ExecuteProcedure(parameters)));
        }
    }

    private ICommand _CommandChangePage;
    public ICommand CommandChangePage
    {
        get
        {
            return _CommandChangePage ?? (_CommandChangePage = new
DelegateCommand(
                (parameters) => ChangePageProcedure(parameters)));
        }
    }

    private void ChangePageProcedure(object parameters)
    {
        App.Current.MainPage = new NavigationPage(new Authorization());
    }

    private async void ExecuteProcedure(object parameters)
    {
        if (IsWaiting)
            return;
        IsWaiting = true;
    }

```

```

        User user = new User { Login = _TextLogin, Password =
        _TextPassword, Nickname = _TextNickname };
        if (!validationRegister.SuitForm(user))
        {
            await View.DisplayAlert("Результат", "Некорректные данные,
введите заново", "Заккрыть");
            IsWaiting = false;
            return;
        }
        var response = await UserUnitWork.Register(user);
        var stream = response.Content;
        string str = await stream.ReadAsStringAsync();
        if (response.IsSuccessStatusCode)
        {
            if (int.TryParse(str, out int IDUser))
            {
                App.IDCurrentUser = IDUser;
                App.Current.MainPage = new MainPage();
                return;
            }
        }
        await View.DisplayAlert("Результат", str, "Заккрыть");
        IsWaiting = false;
    }

    private void OnPropertyChanged(string prop = "")
    {
        PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(prop));
    }
    public event PropertyChangedEventHandler PropertyChanged;
}
}

```

Листинг В.7 – Файл ServiceUser.cs

```

using Authentication.Helpers;
using Authentication.Interfaces;
using Authentication.Model;
using System.Net.Http;
using System.Net.Http.Json;
using System.Threading.Tasks;

namespace Authentication.Services
{
    class ServiceUser : IUserUnitWork
    {
        public async Task<HttpResponseMessage> Authorization(User user)
        {
            JsonContent content = JsonContent.Create(user);
            var result = await
HttpHelper.POST("http://192.168.0.29:5000/Authorization/", content);
            return result;
        }

        public async Task<HttpResponseMessage> Register(User user)
        {
            JsonContent content = JsonContent.Create(user);
            var result = await
HttpHelper.POST("http://192.168.0.29:5000/Register/", content);
            return result;
        }
    }
}

```

```

    }
}

```

Листинг В.8 – Файл ModelViewAuthorization.cs

```

using Authentication.Interfaces;
using Authentication.Model;
using Authentication.Services;
using System.ComponentModel;
using System.Windows.Input;
using Xamarin.Essentials;
using Xamarin.Forms;

namespace Authentication.ModelView
{
    class ModelViewAuthorization : INotifyPropertyChanged
    {
        private IUserUnitWork UserUnitWork;
        private Page View;
        private bool IsWaiting;
        public ModelViewAuthorization(Page view)
        {
            View = view;
            UserUnitWork = new ServiceUser();
            IsWaiting=false;
        }

        private string _TextLogin { get; set; }
        public string TextLogin
        {
            get
            {
                return _TextLogin;
            }
            set
            {
                _TextLogin = value;
                OnPropertyChanged("TextLogin");
            }
        }

        private string _TextPassword { get; set; }
        public string TextPassword
        {
            get
            {
                return _TextPassword;
            }
            set
            {
                _TextPassword = value;
                OnPropertyChanged("TextPassword");
            }
        }

        private ICommand _clickCommand;
        public ICommand ClickCommand
        {
            get
            {
                return _clickCommand ?? (_clickCommand = new DelegateCommand(
                    (parameters) => ExecuteProcedure()));
            }
        }
    }
}

```

```

    }
}
private ICommand _CommandChangePage;
public ICommand CommandChangePage
{
    get
    {
        return _CommandChangePage ?? (_CommandChangePage = new
DelegateCommand(
        (parameters) => ChangePageProcedure(parameters)));
    }
}
private async void ChangePageProcedure(object parameters)
{
    App.Current.MainPage = new NavigationPage(new Register());
}
public async void ExecuteProcedure()
{
    if (IsWaiting)
        return;
    IsWaiting = true;
    User user = new User { Login = _TextLogin, Password =
_TextPassword };
    var response = await UserUnitWork.Authorization(user);
    var stream = response.Content;
    string str = await stream.ReadAsStringAsync();
    if (response.IsSuccessStatusCode)
    {
        if(int.TryParse(str,out int IDUser))
        {
            App.IDCurrentUser = IDUser;
            App.Current.MainPage = new MainPage();
            Preferences.Set("Login",user.Login);
            Preferences.Set("Password", user.Password);
            return;
        }
    }
    await View.DisplayAlert("Результат", str, "Заккрыть");
    IsWaiting = false;
}

private void OnPropertyChanged(string prop = "")
{
    PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(prop));
}
public event PropertyChangedEventHandler PropertyChanged;
}
}

```

Листинг В.9 – Файл Authorization.xaml.cs

```

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Authentication
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class Authorization : ContentPage
    {
        public Authorization()
        {

```

```

        InitializeComponent();
        BindingContext = new ModelViewAuthorization(this);
    }
}

```

Листинг В.10 – Файл Authorization.xaml

```

using System; <?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:converter="clr-namespace:Authentication.ModelView"
    x:Class="Authentication.Authorization"
    Title="Авторизация">
    <ContentPage.Content>
        <StackLayout Spacing="2" VerticalOptions="CenterAndExpand"
            HorizontalOptions="CenterAndExpand">
            <Label Text="Авторизация"
                VerticalOptions="Center" HorizontalOptions="Center"
                FontSize="18"/>
            <Editor x:Name="EditorLogin" Text="{Binding TextLogin}"
                Placeholder="Логин" VerticalOptions="Center" HorizontalOptions="Center"
                FontSize="16" WidthRequest="300"></Editor>
            <Editor x:Name="EditorPassword" Text="{Binding TextPassword}"
                Placeholder="Пароль" VerticalOptions="Center" HorizontalOptions="Center"
                FontSize="16" WidthRequest="300"></Editor>
            <Button Text="Войти" Command="{Binding ClickCommand}"></Button>
            <Button Text="Зарегистрироваться" Command="{Binding
                CommandChangePage}"></Button>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>

```

Листинг В.11 – Файл MainPage.xaml.cs

```

using System;
using Xamarin.Essentials;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Authentication
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class MainPage : FlyoutPage
    {
        public static MainPage Current { get; set; }
        public MainPage()
        {
            InitializeComponent();
            Detail = new NavigationPage(new MapView());
            Current = this;
        }

        private void Button_Clicked(object sender, EventArgs e)
        {
            Detail = new NavigationPage(new MapView());
            IsPresented = false;
        }

        private void Button_Clicked_1(object sender, EventArgs e)
        {

```

```

        Detail = new NavigationPage(new AddNews());
        IsPresented = false;
    }

    private void Button_Clicked_2(object sender, EventArgs e)
    {
        Detail = new NavigationPage(new Profile(App.IDCurrentUser));
        IsPresented = false;
    }

    private void Button_Clicked_3(object sender, EventArgs e)
    {
        Detail = new NavigationPage(new Authorization());
        Preferences.Set("Login", string.Empty);
        Preferences.Set("Password", string.Empty);
        App.IDCurrentUser = -1;
        IsPresented = false;
    }

    private void Button_Clicked_4(object sender, EventArgs e)
    {
        Detail = new NavigationPage(new ListOfNews());
        IsPresented = false;
    }
}

```

Листинг В.12 – Файл MainPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<FlyoutPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Authentication.MainPage">
    <FlyoutPage.Flyout>
        <ContentPage Title="Master">
            <ContentPage.Content>
                <StackLayout Padding="5,30,5,5">
                    <Button Text="Профиль"
Clicked="Button_Clicked_2"></Button>
                    <Button Text="Капта" Clicked="Button_Clicked"></Button>
                    <Button Text="Новости списком"
Clicked="Button_Clicked_4"></Button>
                    <Button Text="Оставить новость"
Clicked="Button_Clicked_1"></Button>
                    <Button Text="Выйти" Clicked="Button_Clicked_3"></Button>
                </StackLayout>
            </ContentPage.Content>
        </ContentPage>
    </FlyoutPage.Flyout>
    <FlyoutPage.Detail>
        <ContentPage>
            <ContentPage.Content>
                <Label Text="DetailPage" VerticalOptions="Center"
HorizontalOptions="Center"></Label>
            </ContentPage.Content>
        </ContentPage>
    </FlyoutPage.Detail>
</FlyoutPage>

```

Листинг В.13 – Файл ViewArticle.xaml.cs

```

using Authentication.Helpers;
using Authentication.Model;
using FFImageLoading.Forms;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Net.Http.Json;
using System.Threading.Tasks;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Authentication
{
    enum SubscribeState { IsSubscribe, UnSubscribe };

    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ViewArticle : ContentPage
    {
        private ContentPage BeforePage;
        private PinArticle InfoArticle;
        private SubscribeState SubscribeState;
        private EvaluationArticle StateEvaluation;
        private int IDArticle;
        private bool IsWaitingEvaluation;
        public ViewArticle(ContentPage page, int IDArticle)
        {
            InitializeComponent();
            IsWaitingEvaluation = false;
            this.IDArticle = IDArticle;
            ShowArticle();
            GetAllComments();
            BeforePage = page;
        }
        private async void ShowStatistics()
        {
            var response = await
HttpHelper.GET($"http://192.168.0.29:5000/GetEvaluations/{IDArticle}");
            if (response.StatusCode == HttpStatusCode.OK)
            {
                var JsonInfo = await response.Content.ReadAsStringAsync();
                var EvaluationArticles =
JsonConvert.DeserializeObject<List<EvaluationArticle>>(JsonInfo);
                LabelLikes.Text = EvaluationArticles.Where(x => x.IsLike ==
true).Count().ToString();
                LabelDislikes.Text = EvaluationArticles.Where(x => x.IsLike
== false).Count().ToString();
                var UserEvaluation = EvaluationArticles.FirstOrDefault(x =>
x.IDUser == App.IDCurrentUser);
                StateEvaluation = UserEvaluation;
                if (UserEvaluation != null)
                {
                    if (UserEvaluation.IsLike)
                        FrameLikes.BackgroundColor = Color.LightGreen;
                    else
                        FrameDislikes.BackgroundColor = Color.IndianRed;
                }
            }
        }
    }
}

```

```

    }
}

private async void ShowAuthorData(int IDUser)
{
    var response = await
HttpHelper.GET($"http://192.168.0.29:5000/GetProfile/{IDUser}");
    if (response.StatusCode == HttpStatusCode.OK)
    {
        var CurrentProfileDataJson = await
response.Content.ReadAsStringAsync();
        var CurrentProfileData =
JsonConvert.DeserializeObject<ProfileData>(CurrentProfileDataJson);
        if (CurrentProfileData.ImageProfile != null)
        {
            AuthorArticleImage.Source = ImageSource.FromStream(() =>
new MemoryStream(CurrentProfileData.ImageProfile));
        }
        NicknameLabel.Text = CurrentProfileData.Nickname;
        if (App.IDCurrentUser != IDUser)
        {
            var responseSubscribe = await
HttpHelper.GET($"http://192.168.0.29:5000/GetSubscription/{IDUser}/{App.IDCur
rentUser}");
            if(response.StatusCode==HttpStatusCode.OK)
            {
                var contentSubscription = await
responseSubscribe.Content.ReadAsStringAsync();
                var subscription =
JsonConvert.DeserializeObject<Subscriptions>(contentSubscription);
                if (subscription == null)
                {
                    ButtonSubscribe.Text = "Подписаться";
                    SubscribeState = SubscribeState.UnSubscribe;
                }
                else
                {
                    ButtonSubscribe.Text = "Отписаться";
                    SubscribeState = SubscribeState.IsSubsribe;
                }
            }
            ButtonSubscribe.IsVisible = true;
        }
    }
}

private async void ButtonSubscribeAction(object sender,EventArgs e)
{
    switch (SubscribeState)
    {
        case SubscribeState.IsSubsribe:
            var responseDeleteSub = await
HttpHelper.DELETE($"http://192.168.0.29:5000/Unsubscribe/{InfoArticle.IDUser}
/{App.IDCurrentUser}");
            if(responseDeleteSub.StatusCode == HttpStatusCode.OK)
            {
                SubscribeState = SubscribeState.UnSubscribe;
                ButtonSubscribe.Text = "Подписаться";
            }
            break;
        case SubscribeState.UnSubscribe:
            var NewSubscription = new Subscriptions { IDProvider =
InfoArticle.IDUser, IDSubscriber = App.IDCurrentUser };
            var contentJson = JsonConvert.Create(NewSubscription);

```

```

        var result = await
HttpHelper.POST("http://192.168.0.29:5000/Subscribe/", contentJson);
        if (result.StatusCode == HttpStatusCode.OK)
        {
            SubscribeState = SubscribeState.IsSubscribe;
            ButtonSubscribe.Text = "Отписаться";
        }
        break;
    }
}
private async void ShowArticle()
{
    var response = await
HttpHelper.GET($"http://192.168.0.29:5000/GetArticle/{IDArticle}");
    if (response.StatusCode == HttpStatusCode.OK)
    {
        var NewsJson = await response.Content.ReadAsStringAsync();
        var News =
JsonConvert.DeserializeObject<PinArticle>(NewsJson);
        InfoArticle = News;
        ShowAuthorData(News.IDUser);
        ShowStatistics();
        var task = CreateArticleInterface(News);
        await task;
    }
}
private async Task CreateArticleInterface(PinArticle pinArticle)
{
    //LayoutForNews.Children.Clear();
    await ScrollViewNews.ScrollToAsync(LayoutForNews,
ScrollToPosition.Start, false);
    Label Title = new Label();
    Title.Text = pinArticle.Title;
    Title.FontSize = 32;
    Title.FontAttributes = FontAttributes.Bold;
    Title.TextColor = Color.Black;

    Label Address = new Label();
    Address.Text = pinArticle.Adress.Name;
    Address.FontSize = 26;
    Address.FontAttributes = FontAttributes.Bold;
    List<StackLayout> paragraphs = new List<StackLayout>();
    for (int i = 0; i < pinArticle.CountParagraphs; i++)
    {
        StackLayout paragraph = new StackLayout();
        Label TextLabel = new Label();
        TextLabel.FontSize = 20;
        TextLabel.TextColor = Color.Black;
        TextLabel.BackgroundColor = Color.Gray;

        CachedImage cachedImage = new CachedImage();
        cachedImage.HeightRequest = 400;
        cachedImage.Aspect = Aspect.AspectFill;
        cachedImage.DownsamplingToViewSize = true;
        cachedImage.BackgroundColor = Color.Gray;
        if (i == pinArticle.CountParagraphs - 1)
        {
            paragraph.Children.Add(TextLabel);
        }
        else
        {
            paragraph.Children.Add(TextLabel);
            paragraph.Children.Add(cachedImage);
        }
    }
}

```

```

        paragraphs.Add(paragraph);
    }

    LayoutForNews.Children.Add(Title);
    LayoutForNews.Children.Add(Address);
    for (int i = 0; i < paragraphs.Count; i++)
    {
        GetDataArticle(i + 1, paragraphs[i]);
        LayoutForNews.Children.Add(paragraphs[i]);
    }
}

private async void GetDataArticle(int NumberParagraph, StackLayout
InterfaceParagraph)
{
    var response = await
HttpHelper.GET($"http://192.168.0.29:5000/GetParagraph/{IDArticle}/{NumberPar
agraph}");
    if (response.StatusCode == HttpStatusCode.OK)
    {
        var Json = await response.Content.ReadAsStringAsync();
        var paragraph =
JsonConvert.DeserializeObject<DataParagraph>(Json);
        foreach (var element in InterfaceParagraph.Children)
        {
            element.BackgroundColor = Color.Default;
        }
        var label = InterfaceParagraph.Children[0] as Label;
        label.Text = paragraph.TextParagraph;
        if (paragraph.ImageParagraph != null)
        {
            var cachedImage = InterfaceParagraph.Children[1] as
CachedImage;
            cachedImage.Source = ImageSource.FromStream(() => new
MemoryStream(paragraph.ImageParagraph));
        }
    }
}

protected override bool OnBackButtonPressed()
{
    MainPage.Current.Detail = new NavigationPage(BeforePage);
    App.Current.MainPage = MainPage.Current;
    return true;
}

private async void SetLike(object sender, EventArgs e)
{
    if (IsWaitingEvaluation)
        return;
    IsWaitingEvaluation = true;
    HttpResponseMessage response;
    EvaluationArticle evaluationArticle = new EvaluationArticle {
IDArticle = IDArticle, IDUser = App.IDCurrentUser, IsLike = true };
    var content = JsonConvert.Create(evaluationArticle);
    if (StateEvaluation==null)
    {
        response = await
HttpHelper.POST("http://192.168.0.29:5000/SetEvaluation/", content);
        if (response.StatusCode == HttpStatusCode.OK)
        {
            FrameLikes.BackgroundColor = Color.LightGreen;
            LabelLikes.Text = (int.Parse(LabelLikes.Text) +
1).ToString();
            StateEvaluation = evaluationArticle;
        }
    }
}

```

```

        IsWaitingEvaluation = false;
        return;
    }

    if(StateEvaluation.IsLike)
    {
        response = await
HttpHelper.DELETE($"http://192.168.0.29:5000/DeleteEvaluation/{IDArticle}/{Ap
p.IDCurrentUser}");
        if(response.StatusCode==HttpStatusCode.OK)
        {
            FrameLikes.BackgroundColor = Color.LightGray;
            LabelLikes.Text = (int.Parse(LabelLikes.Text) -
1).ToString();

            StateEvaluation = null;
        }
    }
    else
    {
        response = await
HttpHelper.POST("http://192.168.0.29:5000/SetEvaluation/", content);
        if (response.StatusCode == HttpStatusCode.OK)
        {
            FrameDislikes.BackgroundColor = Color.LightGray;
            LabelDislikes.Text= (int.Parse(LabelDislikes.Text) -
1).ToString();

            FrameLikes.BackgroundColor = Color.LightGreen;
            LabelLikes.Text = (int.Parse(LabelLikes.Text) +
1).ToString();

            StateEvaluation.IsLike = true;
        }
    }
    IsWaitingEvaluation = false;
}
private async void SetDislike(object sender, EventArgs e)
{
    if (IsWaitingEvaluation)
        return;
    IsWaitingEvaluation = true;
    HttpResponseMessage response;
    EvaluationArticle evaluationArticle = new EvaluationArticle {
IDArticle = IDArticle, IDUser = App.IDCurrentUser, IsLike = false };
    var content = JsonConvert.Create(evaluationArticle);
    if (StateEvaluation == null)
    {
        response = await
HttpHelper.POST("http://192.168.0.29:5000/SetEvaluation/", content);
        if (response.StatusCode == HttpStatusCode.OK)
        {
            FrameDislikes.BackgroundColor = Color.IndianRed;
            LabelDislikes.Text = (int.Parse(LabelDislikes.Text) +
1).ToString();

            StateEvaluation = evaluationArticle;
        }
        IsWaitingEvaluation = false;
        return;
    }

    if (!StateEvaluation.IsLike)
    {
        response = await
HttpHelper.DELETE($"http://192.168.0.29:5000/DeleteEvaluation/{IDArticle}/{Ap
p.IDCurrentUser}");
        if (response.StatusCode == HttpStatusCode.OK)

```

```

        {
            FrameDislikes.BackgroundColor = Color.LightGray;
            LabelDislikes.Text = (int.Parse(LabelDislikes.Text) -
1).ToString();
            StateEvaluation = null;
        }
    }
    else
    {
        response = await
HttpHelper.POST("http://192.168.0.29:5000/SetEvaluation/", content);
        if (response.StatusCode == HttpStatusCode.OK)
        {
            FrameLikes.BackgroundColor = Color.LightGray;
            LabelLikes.Text = (int.Parse(LabelLikes.Text) -
1).ToString();
            FrameDislikes.BackgroundColor = Color.IndianRed;
            LabelDislikes.Text = (int.Parse(LabelDislikes.Text) +
1).ToString();
            StateEvaluation.IsLike = false;
        }
        IsWaitingEvaluation = false;
    }
    private async void SetComment(object sender, EventArgs e)
    {
        if(EditorComment.Text=="")
        {
            await DisplayAlert("Ошибка", "Комментарий пуст", "Заккрыть");
            return;
        }
        Comment comment = new Comment { IDArticle = IDArticle,
IDUserOfComment = App.IDCurrentUser, TextOfComment = EditorComment.Text };
        var content = JsonConvert.Create(comment);
        var response = await
HttpHelper.POST("http://192.168.0.29:5000/AddComment/", content);
        if(response.StatusCode==HttpStatusCode.OK)
        {
            EditorComment.Text = "";
            ShowComment(comment);
        }
    }
    private async void GetAllComments()
    {
        var response = await
HttpHelper.GET($"http://192.168.0.29:5000/GetComments/{IDArticle}");
        if(response.StatusCode==HttpStatusCode.OK)
        {
            var Json = await response.Content.ReadAsStringAsync();
            var comments =
JsonConvert.DeserializeObject<List<Comment>>(Json);
            foreach(var comment in comments)
            {
                ShowComment(comment);
                await Task.Delay(100);
            }
        }
    }
    private async void ShowComment(Comment comment)
    {
        Grid grid = new Grid();
        ColumnDefinition columnFirst = new ColumnDefinition();
        columnFirst.Width = new GridLength(100);
    }

```

```

ColumnDefinition columnSecond = new ColumnDefinition();
columnSecond.Width = new GridLength(70,GridUnitType.Star);

RowDefinition rowFirst = new RowDefinition();
rowFirst.Height = new GridLength(100);

RowDefinition rowSecond = new RowDefinition();
rowSecond.Height = new GridLength(30,GridUnitType.Star);

grid.RowDefinitions.Add(rowFirst);
grid.RowDefinitions.Add(rowSecond);

grid.ColumnDefinitions.Add(columnFirst);
grid.ColumnDefinitions.Add(columnSecond);

StackLayout stackLayoutImage = new StackLayout();
grid.Children.Add(stackLayoutImage);
stackLayoutImage.Orientation = StackOrientation.Horizontal;
Grid.SetRow(stackLayoutImage, 0);
Grid.SetColumn(stackLayoutImage, 0);

Frame FrameImage = new Frame();
FrameImage.CornerRadius = 100;
FrameImage.WidthRequest = 100;
FrameImage.HeightRequest = 100;
FrameImage.IsClippedToBounds = true;
FrameImage.Padding = 0;

stackLayoutImage.Children.Add(FrameImage);

CachedImage image = new CachedImage() { DownsampleToViewSize =
true , Aspect = Aspect.AspectFill };

var tapGestureRecognizer = new TapGestureRecognizer();
tapGestureRecognizer.NumberOfTapsRequired = 1;
tapGestureRecognizer.Tapped += (s, e) =>
{
    MainPage.Current.Detail = new NavigationPage(new
Profile(comment.IDUserOfComment,this));
    App.Current.MainPage = MainPage.Current;
};
image.GestureRecognizers.Add(tapGestureRecognizer);

//Код для получения изображения

FrameImage.Content = image;

StackLayout stackLayoutComment = new StackLayout();
grid.Children.Add(stackLayoutComment);
Grid.SetRow(stackLayoutComment,0);
Grid.SetRowSpan(stackLayoutComment,2);
Grid.SetColumn(stackLayoutComment,1);

Label LabelNick = new Label();
LabelNick.TextColor = Color.Black;
LabelNick.VerticalOptions = LayoutOptions.Center;
LabelNick.FontSize = 18;

stackLayoutComment.Children.Add(LabelNick);

Label LabelComment = new Label();
stackLayoutComment.Children.Add(LabelComment);

```

```

        var response = await
HttpHelper.GET($"http://192.168.0.29:5000/GetProfile/{comment.IDUserOfComment
}");
        if (response.StatusCode == HttpStatusCode.OK)
        {
            var Json = await response.Content.ReadAsStringAsync();
            var ProfileOfUser =
JsonConvert.DeserializeObject<ProfileData>(Json);

            LabelNick.Text = ProfileOfUser.Nickname;
            LabelComment.Text = comment.TextOfComment;
            if (ProfileOfUser.ImageProfile != null)
            {
                image.Source = ImageSource.FromStream(() => new
MemoryStream(ProfileOfUser.ImageProfile));
            }
            CommentsLayout.Children.Add(grid);
        }
    }

    private void TapOpenProfileAuthor(object sender, EventArgs e)
    {
        MainPage.Current.Detail = new NavigationPage(new
Profile(InfoArticle.IDUser, this));
        App.Current.MainPage = MainPage.Current;
    }
}
}c

```

Листинг В.14 – Файл ViewArticle.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:ffimageloading="clr-
namespace:FFImageLoading.Forms;assembly=FFImageLoading.Forms"
    x:Class="Authentication.ViewArticle">
    <ContentPage.Content>
        <ScrollView x:Name="ScrollViewNews" Padding="2">
            <StackLayout Margin="5">
                <StackLayout Orientation="Horizontal">
                    <Frame CornerRadius="100"
                        HeightRequest="80"
                        WidthRequest="80"
                        Padding="0"
                        IsClippedToBounds="True">
                        <Frame.GestureRecognizers>
                            <TapGestureRecognizer NumberOfTapsRequired="1"
Tapped="TapOpenProfileAuthor"></TapGestureRecognizer>
                        </Frame.GestureRecognizers>
                        <ffimageloading:CachedImage
x:Name="AuthorArticleImage" Aspect="AspectFill" DownsampleToViewSize="True"/>
                        </Frame>
                        <Label x:Name="NicknameLabel" Text="Мой ник"
VerticalOptions="Center" TextColor="Black" FontSize="26"></Label>
                    <Button
                        x:Name="ButtonSubscribe"
                        HorizontalOptions="FillAndExpand"
                        CornerRadius="100"
                        Text="Подписаться"
                        TextColor="Black"
                        FontAttributes="Bold"

```

```

        FontSize="16"
        WidthRequest="25"
        HeightRequest="25"
        IsVisible="false"
        Clicked="ButtonSubscribeAction">
    </Button>
</StackLayout>
<Frame BackgroundColor="White" CornerRadius="20" Padding="0">
    <StackLayout x:Name="LayoutForNews"
Margin="10"></StackLayout>
</Frame>
<StackLayout Orientation="Horizontal"
HorizontalOptions="EndAndExpand" Margin="10,20,0,0">
    <Frame x:Name="FrameLikes" BackgroundColor="LightGray"
CornerRadius="50" Padding="0" IsClippedToBounds="True" HeightRequest="50"
WidthRequest="95">
        <Frame.GestureRecognizers>
            <TapGestureRecognizer NumberOfTapsRequired="1"
Tapped="SetLike"></TapGestureRecognizer>
        </Frame.GestureRecognizers>
        <StackLayout Orientation="Horizontal">
            <ImageButton BackgroundColor="Transparent"
Clicked="SetLike" Source="like.png" Margin="6"></ImageButton>
            <Label x:Name="LabelLikes" Text=""
TextColor="Black" HorizontalOptions="Start" VerticalOptions="CenterAndExpand"
FontSize="24"></Label>
        </StackLayout>
    </Frame>
    <Frame x:Name="FrameDislikes" BackgroundColor="LightGray"
CornerRadius="50" Padding="0" IsClippedToBounds="True" HeightRequest="50"
WidthRequest="95">
        <Frame.GestureRecognizers>
            <TapGestureRecognizer NumberOfTapsRequired="1"
Tapped="SetDislike"></TapGestureRecognizer>
        </Frame.GestureRecognizers>
        <StackLayout Orientation="Horizontal">
            <ImageButton BackgroundColor="Transparent"
Clicked="SetDislike" Source="dislike.png" Margin="6"></ImageButton>
            <Label x:Name="LabelDislikes" Text=""
TextColor="Black" HorizontalOptions="Start" VerticalOptions="CenterAndExpand"
FontSize="24"></Label>
        </StackLayout>
    </Frame>
</StackLayout>
<Editor x:Name="EditorComment" Placeholder="Комментарий"
TextColor="Black" FontSize="20" AutoSize="TextChanges"></Editor>
<StackLayout Orientation="Horizontal">
    <Button Clicked="SetComment" BackgroundColor="LightGray"
Text="Оставить комментарий" CornerRadius="100"
HorizontalOptions="EndAndExpand"></Button>
</StackLayout>
<StackLayout x:Name="CommentsLayout">
</StackLayout>
</StackLayout>
</ScrollView>
</ContentPage.Content>
</ContentPage>

```

Листинг В.15 – Файл AddNew.xaml.cs

```

using Android.Content;
using Authentication.Helpers;

```

```

using Authentication.ModelView;
using Dadata;
using FFImageLoading.Forms;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Threading.Tasks;

using Xamarin.Forms;
using Xamarin.Forms.Maps;
using Xamarin.Forms.Xaml;

namespace Authentication
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class AddNews : ContentPage
    {
        private ModelViewAddNews MV;
        public static AddNews currentObject { get; private set; }
        private CachedImage imageChange { get; set; }
        private List<Editor> editorsTexts { get; set; }
        private List<CachedImage> cachedImages { get; set; }
        private Page previewPage { get; set; }

        private string oldText;
        private bool close;
        private string token = "09eba80bbf85f31b257fb416b1ea7dff39e6aa80412";
        public AddNews(Page page = null, int IDArticle = -1)
        {
            InitializeComponent();
            previewPage = page;
            MV = new ModelViewAddNews(IDArticle, this);
            BindingContext = MV;
            editorsTexts = new List<Editor>();
            cachedImages = new List<CachedImage>();
            MV.ImagesViews = cachedImages;
            editorsTexts.Add(CurrentInputText);
            oldText = "";
            close = false;
            currentObject = this;
            if (IDArticle != -1)
                SetInterfaceForEditArticle(IDArticle);
        }

        private async void SetInterfaceForEditArticle(int IDArticle)
        {
            ButtonToFinish.IsEnabled = false;
            var response = await
HttpHelper.GET($"http://192.168.0.29:5000/GetArticle/{IDArticle}");
            if (response.StatusCode == HttpStatusCode.OK)
            {
                var NewsJson = await response.Content.ReadAsStringAsync();
                var News =
JsonConvert.DeserializeObject<PinArticle>(NewsJson);
                MV.Title = News.Title;
                MV.ChoiceAdress = News.Adress;
                var task = CreateArticleInterface(News, IDArticle);
                await task;
            }
        }
    }
}

```

```

        private async Task CreateArticleInterface(PinArticle news,int
IDArticle)
        {
            for(int i=0;i<news.CountParagraphs-1;i++)
            {
                AddImageCached(null);
            }
            for (int i = 0; i < news.CountParagraphs; i++)
            {
                GetDataArticle(i + 1, IDArticle);
            }
        }
        private async void GetDataArticle(int NumberParagraph,int IDArticle)
        {
            var response = await
HttpHelper.GET($"http://192.168.0.29:5000/GetParagraph/{IDArticle}/{NumberPar
agraph}");
            if (response.StatusCode == HttpStatusCode.OK)
            {
                var Json = await response.Content.ReadAsStringAsync();
                var paragraph =
JsonConvert.DeserializeObject<DataParagraph>(Json);
                editorsTexts[NumberParagraph - 1].Text =
paragraph.TextParagraph;
                if (paragraph.ImageParagraph != null)
                {
                    cachedImages[NumberParagraph-1].Source =
ImageSource.FromStream(() => new MemoryStream(paragraph.ImageParagraph));
                }
                else
                {
                    ButtonToFinish.IsEnabled = true;
                }
            }
        }

        public async void AddImageCached(ImageSource imagesrc)
        {
            var Image = new CachedImage {HeightRequest = 400, Aspect =
Aspect.AspectFill, DownsampleToViewSize = true };
            if (MV.ChangePhoto == true)
            {
                imageChange.Source = imagesrc;
                return;
            }
            MV.Images.Add(imagesrc);
            Image.SetBinding(CachedImage.SourceProperty,new
Binding($"Images[{MV.Images.Count-1}]"));

            var grid = new Grid();
            var row = new RowDefinition();
            var column1 = new ColumnDefinition();
            var column2 = new ColumnDefinition();
            column1.Width = new GridLength(150, GridUnitType.Star);
            column2.Width = new GridLength(150, GridUnitType.Star);
            row.Height = 70;
            grid.RowDefinitions.Add(row);
            grid.ColumnDefinitions.Add(column1);
            grid.ColumnDefinitions.Add(column2);
            var buttonDelete = new Button();
            var buttonChange = new Button();
            grid.Children.Add(buttonDelete);
            grid.Children.Add(buttonChange);

```

```

        buttonDelete.Clicked += new EventHandler(delegate (Object o,
EventArgs a)
        {
            LayoutForNews.Children.Remove(Image);
            int indexNumber = cachedImages.IndexOf(Image);
            var EditorUnion = editorsTexts[indexNumber];
            var EditorForRemove = editorsTexts[indexNumber + 1];
            MV.Texts.RemoveAt(indexNumber + 1);
            MV.Images.RemoveAt(indexNumber);
            EditorUnion.Text += "\n" + EditorForRemove.Text;
            editorsTexts.Remove(EditorForRemove);
            LayoutForNews.Children.Remove(EditorForRemove);
            cachedImages.Remove(Image);
            //LayoutForNews.Children.Remove(button);
            LayoutForNews.Children.Remove(grid);
            for(int i = 0; i < MV.Texts.Count; i++)
            {
                editorsTexts[i].SetBinding(Editor.TextProperty, new
Binding($"Texts[{i}]"));
                if (i != MV.Texts.Count - 1)

cachedImages[i].SetBinding(CachedImage.SourceProperty, new
Binding($"Images[{i}]"));
            }
        });

        buttonChange.Clicked += new EventHandler(delegate (Object o,
EventArgs a)
        {
            MV.ChangePhoto = true;
            imageChange = Image;
            Intent intent = new Intent(Intent.ActionSend);
            intent.SetType("image/*");
            intent.SetAction(Intent.ActionGetContent);

App.ActivityCurrent.StartActivityForResult(Intent.CreateChooser(intent,
"Select Picture"), 0);
        });
        buttonDelete.Text = "Удалить изображение";
        buttonDelete.HorizontalOptions = LayoutOptions.End;
        buttonDelete.VerticalOptions = LayoutOptions.Fill;

        buttonChange.Text = "Изменить изображение";
        buttonChange.HorizontalOptions = LayoutOptions.End;
        buttonChange.VerticalOptions = LayoutOptions.Fill;

        Grid.SetRow(buttonDelete, 0);
        Grid.SetRow(buttonChange, 0);

        Grid.SetColumn(buttonDelete, 1);
        Grid.SetColumn(buttonChange, 0);
        var NewEditor = new Editor();
        NewEditor.AutoSize = EditorAutoSizeOption.TextChanges;
        NewEditor.Placeholder = "Введите абзац";

        MV.Texts.Add("");
        NewEditor.SetBinding(Editor.TextProperty, new
Binding($"Texts[{MV.Texts.Count-1}]"));

        editorsTexts.Add(NewEditor);
        cachedImages.Add(Image);
        LayoutForNews.Children.Remove(AddButton);
        LayoutForNews.Children.Add(Image);
        LayoutForNews.Children.Add(grid);

```

```

        LayoutForNews.Children.Add(NewEditor);
        LayoutForNews.Children.Add(AddButton);
    }

    private void InputAdress_Focused(object sender, FocusEventArgs e)
    {
        ScrollViewNews.BackgroundColor=Color.Gray;
        ScrollViewNews.IsEnabled = false;
        LayoutForNews.IsEnabled = false;
        ShowChoice.IsVisible = true;
        InputAdress.Unfocus();
    }

    private void InputAdress_TextChanged(object sender,
    TextChangedEventArgs e)
    {
        oldText = e.NewTextValue;
        if (close)
            return;
        AutoCompleteFill(e.NewTextValue);
    }
    private async void AutoCompleteFill(string adressString)
    {
        close = true;
        await Task.Delay(500);
        if (oldText != adressString)
        {
            AutoCompleteFill(oldText);
            return;
        }
        var api = new SuggestClientAsync(token);
        var response = await api.SuggestAddress(adressString);
        List<HelperAdress> addresses = new List<HelperAdress>();
        foreach (var Currentaddress in response.suggestions)
        {
            if (Currentaddress.data.qc_geo == "5")
                continue;
            addresses.Add(new HelperAdress(Currentaddress.value,
            Currentaddress));
        }
        if (addresses.Count == 0)
        {
            addresses.Add(new HelperAdress("Адрес не нашёлся, попробуйте
            указать его на карте", null));
        }
        MyListView.ItemsSource = addresses;
        close = false;
        //var address = response.suggestions[0].data;
    }
    protected override bool OnBackButtonPressed()
    {
        if (ShowChoice.IsVisible || MyMap.IsVisible)
        {
            MyMap.IsVisible = false;
            ShowChoice.IsVisible = false;
            LayoutForNews.IsEnabled = true;
            ScrollViewNews.IsVisible = true;
            ScrollViewNews.BackgroundColor = Color.Default;
            ScrollViewNews.IsEnabled = true;
            return true;
        }
        if (previewPage!=null)
        {
            MainPage.Current.Detail = new NavigationPage(previewPage);
        }
    }

```

```

        App.Current.MainPage = MainPage.Current;
        return true;
    }
    return base.OnBackButtonPressed();
}
private void MyListView_ItemSelected(object sender,
SelectedItemChangedEventArgs e)
{
    var myTemp = e.SelectedItem as HelperAddress;
    if(myTemp.Address==null)
    {
        ButtonOnMap_Clicked(sender, e);
        return;
    }
    //var address = myTemp.Address;
    //InputAddress.Text = myTemp.Title;

    MV.ChoiceAddress = new Adress() {
        Name = myTemp.Title,
        Longitude = double.Parse(myTemp.Address.data.geo_lon),
        Latitude = double.Parse(myTemp.Address.data.geo_lat)};
    ShowChoice.IsVisible = false;

    ScrollViewNews.BackgroundColor=Color.Default;
    ScrollViewNews.IsEnabled = true;
    LayoutForNews.IsEnabled = true;
}
private void ButtonOnMap_Clicked(object sender, EventArgs e)
{
    ShowChoice.IsVisible = false;
    ScrollViewNews.IsVisible = false;
    MyMap.IsVisible = true;
}

private void MyMap_MapClicked(object sender, MapClickedEventArgs e)
{
    GeoGet(sender, e);
}
private async void GeoGet(object sender, MapClickedEventArgs e)
{
    Xamarin.Forms.Maps.Geocoder geoCoder = new
Xamarin.Forms.Maps.Geocoder();
    //var result = await
Geocoding.GetPlacemarksAsync(e.Point.Latitude,e.Point.Longitude);
    var result = await geoCoder.GetAddressesForPositionAsync(new
Xamarin.Forms.Maps.Position(e.Position.Latitude, e.Position.Longitude));
    if(result==null || result.Count()==0)
    {
        string position = "Широта:" + e.Position.Latitude + "
Долгота:" + e.Position.Longitude;
        //InputAddress.Text = position;
        MV.ChoiceAddress = new Adress()
        {
            Name = position,
            Latitude = e.Position.Latitude,
            Longitude = e.Position.Longitude
        };
        OnBackButtonPressed();
        return;
    }
    //InputAddress.Text = result.First().ToString();
    MV.ChoiceAddress = new Adress()
    {
        Name = result.First().ToString(),

```

```

        Latitude = e.Position.Latitude,
        Longitude = e.Position.Longitude
    };
    OnBackPressed();
    //await DisplayAlert("Info", result.First().ToString(), "ok");
}
}
public class News
{
    public int ID { get; set; }
    public int IDUser { get; set; }
    public string Title { get; set; }
    public Address Address { get; set; }
    public List<string> ParagraphTexts { get; set; }
    public List<byte[]> ImagesBytes { get; set; }
}
public class Address
{
    public string Name { get; set; }
    public double Latitude { get; set; }
    public double Longitude { get; set; }
}
public class HelperAddress
{
    public string Title { get; set; }
    public Dadata.Model.Suggestion<Dadata.Model.Address> Address { get;
set; }
    public HelperAddress(string title,
Dadata.Model.Suggestion<Dadata.Model.Address> address)
    {
        Address = address;
        Title = title;
    }
}
}

```

Листинг В.16 – Файл AddNews.xaml

```

?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:map="clr-
namespace:Xamarin.Forms.Maps;assembly=Xamarin.Forms.Maps"
    x:Class="Authentication.AddNews"
    Title="{Binding TitlePage}">
    <ContentPage.ToolbarItems>
        <ToolbarItem x:Name="ButtonToFinish" Icon="confirmIcon.png"
Command="{Binding CommandCreateArticle}"/>
    </ContentPage.ToolbarItems>
    <ContentPage.Content>
        <Grid>
            <Grid.RowDefinitions>
                <RowDefinition Height="25"></RowDefinition>
                <RowDefinition Height="400*"></RowDefinition>
            </Grid.RowDefinitions>
            <ScrollView Grid.Row="0" Grid.RowSpan="2"
x:Name="ScrollViewNews">
                <StackLayout x:Name="LayoutForNews">
                    <Editor x:Name="TitleNews" Text="{Binding Title}"
Placeholder="Заголовок" FontSize="20" FontAttributes="Bold"></Editor>
                <Grid>
                    <Grid.ColumnDefinitions>

```

```

        <ColumnDefinition
Width="300*"></ColumnDefinition>
        <ColumnDefinition
Width="100*"></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <Editor x:Name="InputAdress" Text="{Binding
ChoiceAdress.Name}" Focused="InputAdress_Focused" FontSize="16"
HeightRequest="55" Placeholder="Введите адрес" Grid.Column="0"></Editor>
    <Button x:Name="ButtonOnMap" Text="На карте"
Clicked="ButtonOnMap_Clicked" Grid.Column="1"></Button>
</Grid>
    <Editor x:Name="CurrentInputText" Text="{Binding
Texts[0]}" AutoSize="TextChanges" Placeholder="Введите абзац" ></Editor>
    <Button x:Name="AddButton" Text="Добавить изображение"
Command="{Binding CommandAddImage}" HorizontalOptions="End"
VerticalOptions="Fill"></Button>
</StackLayout>
</ScrollView>
<StackLayout BackgroundColor="White" x:Name="ShowChoice"
IsVisible="false" Grid.Row="1">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="5"></RowDefinition>
            <RowDefinition Height="30*"></RowDefinition>
            <RowDefinition Height="400*"></RowDefinition>
        </Grid.RowDefinitions>
        <Grid Grid.Row="1">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="300*"></ColumnDefinition>
                <ColumnDefinition Width="100*"></ColumnDefinition>
            </Grid.ColumnDefinitions>
            <Editor x:Name="InputAdressPlacehold" Text="{Binding
ChoiceAdress.Name}" TextChanged="InputAdress_TextChanged" FontSize="16"
HeightRequest="65" Placeholder="Введите местоположение"
Grid.Column="0"></Editor>
            <Button x:Name="ButtonHolderMap" Text="На карте"
Grid.Column="1" Clicked="ButtonOnMap_Clicked"></Button>
        </Grid>
        <ListView x:Name="MyListView" Margin="10,5,15,5"
RowHeight="45" HorizontalOptions="CenterAndExpand"
ItemSelected="MyListView_ItemSelected" VerticalOptions="CenterAndExpand"
Grid.Row="2">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <StackLayout IsClippedToBounds="True">
                            <Label Text="{Binding Title}"
TextColor="Black" FontSize="18"></Label>
                        </StackLayout>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </Grid>
</StackLayout>
<map:Map x:Name="MyMap" MapType="Street" HasZoomEnabled="True"
Grid.Row="0" Grid.RowSpan="2" IsVisible="false"
MapClicked="MyMap_MapClicked"></map:Map>
</Grid>
</ContentPage.Content>
</ContentPage>

```

Листинг В.17 – Файл ServiceArticles.cs

```

using Authentication.Helpers;
using Authentication.Interfaces;
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Net.Http.Json;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;

namespace Authentication.Services
{
    class ServiceArticles : IUnitOfArticles
    {
        public async Task<HttpResponseMessage> CreateArticle(string Title,
        Address address, List<string> Texts, List<byte[]> Images)
        {
            var content = await CreateContent(Title, address, Texts, Images);
            var result = await
            HttpHelper.POST("http://192.168.0.29:5000/AddNewArticle/", content);
            return result;
        }

        public async Task<HttpResponseMessage> UpdateArticle(string Title,
        Address address, List<string> Texts, List<byte[]> Images, int IDArticle)
        {
            var content = await CreateContent(Title, address, Texts, Images,
            IDArticle);
            var result = await
            HttpHelper.PUT("http://192.168.0.29:5000/UpdateArticle/", content);
            return result;
        }

        private async Task<JsonContent> CreateContent(string Title, Address
        address, List<string> Texts, List<byte[]> Images, int IDArticle=-1)
        {
            var news = new News();
            news.IDUser = App.IDCurrentUser;
            news.ImagesBytes = Images;
            news.ParagraphTexts = Texts;
            news.Title = Title;
            news.Address = address;
            news.ID = IDArticle;
            JsonContent content = JsonContent.Create(news);
            return content;
        }
    }
}

```

Листинг В.18 – Файл ModelViewAddNews.cs

```

using Android.Content;
using Authentication.Helpers;
using Authentication.Interfaces;
using Authentication.Services;
using FFImageLoading.Forms;
using System;
using System.Collections.Generic;
using System.ComponentModel;

```

```

using System.Net.Http;
using System.Text;
using System.Windows.Input;
using Xamarin.Forms;

namespace Authentication.ModelView
{
    class ModelViewAddNews : INotifyPropertyChanged
    {
        private IUnitOfArticles WorkingWithArticle { get; set; }
        public bool ChangePhoto { get; set; }
        private Page CurrentView { get; set; }
        public ModelViewAddNews(int ID, Page view)
        {
            CurrentView = view;
            WorkingWithArticle = new ServiceArticles();
            ChangePhoto = false;
            IDArticle = ID;
            _Texts = new List<string>();
            _Texts.Add("");
            _Images = new List<ImageSource>();
            if (ID == -1)
                TitlePage = "Создание новости";
            else
                TitlePage = "Редактирование новости";
        }
        public int IDArticle { get; set; }
        public List<CachedImage> ImagesViews { get; set; }
        private string _Title { get; set; }
        public string Title
        {
            get
            {
                return _Title;
            }
            set
            {
                _Title = value;
                OnPropertyChanged("Title");
            }
        }
        private string _TitlePage { get; set; }
        public string TitlePage
        {
            get
            {
                return _TitlePage;
            }
            set
            {
                _TitlePage = value;
                OnPropertyChanged("TitlePage");
            }
        }
        private Address _ChoiceAddress { get; set; }
        public Address ChoiceAddress
        {
            get
            {
                return _ChoiceAddress;
            }
            set
            {
                _ChoiceAddress = value;
            }
        }
    }
}

```

```

        OnPropertyChanged("ChoiceAdress");
    }
}
private ICommand _CommandAddImage;
public ICommand CommandAddImage
{
    get
    {
        return _CommandAddImage ?? (_CommandAddImage = new
DelegateCommand(
        (parameters) => AddImage(parameters)));
    }
}

private ICommand _CommandCreateArticle;
public ICommand CommandCreateArticle
{
    get
    {
        return _CommandCreateArticle ?? (_CommandCreateArticle = new
DelegateCommand(
        (parameters) => CreateArticle(parameters)));
    }
}

private async void CreateArticle(object parameters)
{
    if(Title=="" || ChoiceAdress==null || Title==null)
    {
        await CurrentView.DisplayAlert("Результат", "Некорректные
данные для создания новости", "Закрыть");
        return;
    }
    List<byte[]> DataImages = new List<byte[]>();
    foreach(var image in ImagesViews)
    {
        image.SetIsLoading(true);
        DataImages.Add(await image.GetImageAsPngAsync());
    }
    HttpResponseMessage result = null;
    if (IDArticle == -1)
    {
        result = await WorkingWithArticle.CreateArticle(Title,
ChoiceAdress, Texts, DataImages);
    }
    else
    {
        result = await WorkingWithArticle.UpdateArticle(Title,
ChoiceAdress, Texts, DataImages, IDArticle);
    }
    var stream = result.Content;
    string message = await stream.ReadAsStringAsync();
    await CurrentView.DisplayAlert("Результат", message, "Закрыть");
}

private void AddImage(object parameters)
{
    ChangePhoto = false;
    Intent intent = new Intent(Intent.ActionSend);
    intent.SetType("image/*");
    intent.SetAction(Intent.ActionGetContent);
}

```

```

App.ActivityCurrent.StartActivityForResult(Intent.CreateChooser(intent,
"Select Picture"), 0);
    }

    private List<string> _Texts { get; set; }
    public List<string> Texts
    {
        get
        {
            return _Texts;
        }
        set
        {
            _Texts = value;
            OnPropertyChanged("Texts");
        }
    }

    private List<ImageSource> _Images { get; set; }
    public List<ImageSource> Images
    {
        get
        {
            return _Images;
        }
        set
        {
            _Images = value;
            OnPropertyChanged("Images");
        }
    }
    private void OnPropertyChanged(string prop = "")
    {
        PropertyChanged?.Invoke(this, new
PropertyChangeEventArgs(prop));
    }
    public event PropertyChangedEventHandler PropertyChanged;
}
}

```

Листинг В.19 – Файл MapView.xaml.cs

```

using System;

using Xamarin.Forms;
using Xamarin.Forms.Maps;
using Xamarin.Forms.Xaml;
using Authentication.ModelView;

namespace Authentication
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class MapView : ContentPage
    {
        private ModelViewMap MV;
        public MapView()
        {
            InitializeComponent();
            MV = new ModelViewMap(this);
            BindingContext = MV;
        }
    }
}

```

```

private void OpenArticle(object sender, EventArgs e)
{
    var pin = sender as Pin;
    if (pin == null)
        return;

    MV.ExecuteProcedure(pin);
}
}
public class PinArticle
{
    public int IDUser { get; set; }
    public string Title { get; set; }
    public Address Address { get; set; }
    public byte CountParagraphs { get; set; }
}
public class DataParagraph
{
    public string TextParagraph { get; set; }
    public byte[] ImageParagraph { get; set; }
}
public class Place
{
    public int IDArticle { get; set; }
    public Position Position { get; set; }
    public string Address { get; set; }
    public string Name { get; set; }
    public Place(PlaceHelper placeHelper)
    {
        IDArticle = placeHelper.IDArticle;
        Position = new Position(placeHelper.Latitude,
placeHelper.Longitude);
        Address = placeHelper.Address;
        Name = placeHelper.Name;
    }
}
public class PlaceHelper
{
    public int IDArticle { get; set; }
    public double Latitude { get; set; }
    public double Longitude { get; set; }
    public string Address { get; set; }
    public string Name { get; set; }
}
}

```

Листинг В.20 – Файл MapView.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:map="clr-
namespace:Xamarin.Forms.Maps;assembly=Xamarin.Forms.Maps"
    xmlns:behaviors="clr-namespace:Authentication.CustomXaml"
    x:Class="Authentication.MapView">
    <ContentPage.Content>
        <Grid>
            <Grid.RowDefinitions>
                <RowDefinition Height="400*"></RowDefinition>
                <RowDefinition Height="400*"></RowDefinition>

```

```

        </Grid.RowDefinitions>
        <StackLayout x:Name="LayoutForMap" Grid.Row="0" Grid.RowSpan="2">
            <map:Map x:Name="MyMap" MapType="Street"
                HasZoomEnabled="True" ItemsSource="{Binding Places}">
                <map:Map.ItemTemplate>
                    <DataTemplate>
                        <map:Pin Clicked="OpenArticle"
                            MarkerClicked="OpenArticle"
                                Position="{Binding Position}"
                                Address="{Binding Address}"
                                Label="{Binding Name}">
                        </map:Pin>
                    </DataTemplate>
                </map:Map.ItemTemplate>
            </map:Map>
        </StackLayout>
    </Grid>
</ContentPage.Content>
</ContentPage>

```

Листинг В.21 – Файл ModelViewMap.cs

```

using Authentication.Interfaces;
using Authentication.Services;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;
using Xamarin.Forms;
using Xamarin.Forms.Maps;

namespace Authentication.ModelView
{
    class ModelViewMap : INotifyPropertyChanged
    {
        private IArticleUnitWork ArticleUnitWork;
        private ContentPage View;

        public ModelViewMap(ContentPage view)
        {
            View = view;
            ArticleUnitWork = new ServiceMap();
            GetData();
        }

        private List<Place> _Places { get; set; }
        public List<Place> Places
        {
            get
            {
                return _Places;
            }
            set
            {
                _Places = value;
                OnPropertyChanged("Places");
            }
        }

        public void ExecuteProcedure(Pin pin)

```

```

        {
            var place = pin.BindingContext as Place;
            App.Current.MainPage = new NavigationPage(new ViewArticle(View,
place.IDArticle));
        }

        private async void GetData()
        {
            Places = await ArticleUnitWork.GetArticlesForMap();
        }
        private void OnPropertyChanged(string prop = "")
        {
            PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(prop));
        }
        public event PropertyChangedEventHandler PropertyChanged;
    }
}

```

Листинг В.22 – Файл ServiceMap.cs

```

using Authentication.Interfaces;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Net;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms.Maps;

namespace Authentication.Services
{
    public class ServiceMap : IArticleUnitWork
    {
        public async Task<List<Place>> GetArticlesForMap()
        {
            var response = await
HttpHelper.GET("http://192.168.0.29:5000/NewsOfMap/");
            List<Place> places = new List<Place>();
            if (response.StatusCode == HttpStatusCode.OK)
            {
                var NewsJsonList = await
response.Content.ReadAsStringAsync();
                var ListPlaceHelper =
JsonConvert.DeserializeObject<List<PlaceHelper>>(NewsJsonList);

                foreach (var placehelper in ListPlaceHelper)
                    places.Add(new Place(placehelper));
            }
            return places;
        }
    }
}

```

Листинг В.23 – Файл Profile.xaml.cs

```

using Android.Content;
using Authentication.Helpers;

```

```

using Authentication.Model;
using Authentication.ModelView;
using FFImageLoading.Forms;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Net.Http.Json;
using System.Text;
using System.Threading.Tasks;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Authentication
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class Profile : ContentPage
    {
        //ffimageloading:CachedImage
        public static Profile currentObject { get; private set; }
        private Page BeforePage;
        private int IDUser { get; set; }
        public Profile(int IDUser, Page page = null)
        {
            BeforePage = page;
            InitializeComponent();
            this.IDUser = IDUser;
            currentObject = this;
            GetDataProfile();
            GetArticlesProfile();
            GetSubscribersData();
        }

        private async void GetSubscribersData()
        {
            var response = await
HttpHelper.GET($"http://192.168.0.29:5000/GetSubscribers/{IDUser}");
            if (response.StatusCode == HttpStatusCode.OK)
            {
                var Json = await response.Content.ReadAsStringAsync();
                var subscriptions =
JsonConvert.DeserializeObject<List<Subscriptions>>(Json);
                foreach (var subscription in subscriptions)
                {
                    ShowSubscription(subscription);
                }
            }
        }

        private async void ShowSubscription(Subscriptions subscription)
        {
            var response = await
HttpHelper.GET($"http://192.168.0.29:5000/GetProfile/{subscription.IDProvider}");
            if (response.StatusCode == HttpStatusCode.OK)
            {
                var Json = await response.Content.ReadAsStringAsync();
                var profileData =
JsonConvert.DeserializeObject<ProfileData>(Json);

```

```

Frame frame = new Frame();
frame.CornerRadius = 100;

Grid grid = new Grid();
frame.Content = grid;
ColumnDefinition columnFirst = new ColumnDefinition();
ColumnDefinition columnSecond = new ColumnDefinition();
ColumnDefinition columnThird = new ColumnDefinition();

columnFirst.Width = new GridLength(50);
columnSecond.Width = new GridLength(30, GridUnitType.Star);
columnThird.Width = new GridLength(50, GridUnitType.Star);

grid.ColumnDefinitions.Add(columnFirst);
grid.ColumnDefinitions.Add(columnSecond);
grid.ColumnDefinitions.Add(columnThird);

Frame FrameImage = new Frame();
FrameImage.CornerRadius = 100;
FrameImage.HeightRequest = 50;
FrameImage.WidthRequest = 50;
FrameImage.Padding = 0;
FrameImage.IsClippedToBounds = true;

grid.Children.Add(FrameImage);
Grid.SetColumn(FrameImage, 0);

CachedImage image = new CachedImage() { Aspect =
Aspect.AspectFill, DownsampleToViewSize = true };
if(profileData.ImageProfile!=null)
    image.Source = ImageSource.FromStream(() => new
MemoryStream(profileData.ImageProfile));
FrameImage.Content = image;

var tapGestureRecognizer = new TapGestureRecognizer();
tapGestureRecognizer.NumberOfTapsRequired = 1;
tapGestureRecognizer.Tapped += (s, e) =>
{
    MainPage.Current.Detail = new NavigationPage(new
Profile(subscription.IDProvider, this));
    App.Current.MainPage = MainPage.Current;
};
image.GestureRecognizers.Add(tapGestureRecognizer);

Label LabelNick = new Label();
LabelNick.Margin = new Thickness(15,0,0,0);
LabelNick.TextColor = Color.Black;
LabelNick.VerticalOptions = LayoutOptions.Center;
LabelNick.FontSize = 20;
LabelNick.Text = profileData.Nickname;
grid.Children.Add(LabelNick);
Grid.SetColumn(LabelNick, 1);

if (IDUser == App.IDCurrentUser)
{
    Button button = new Button();
    //button.Text = "Отписаться";
    button.CornerRadius = 100;
    var MV = new ModelViewSubscribers("Отписаться", true,
subscription.IDProvider);
    button.BindingContext = MV;
    button.SetBinding(Button.TextProperty, new
Binding("Text"));

```

```

        button.SetBinding(Button.CommandProperty, new
Binding("ClickCommand"));
        grid.Children.Add(button);
        Grid.SetColumn(button, 2);
    }

    SubscribersProfile.Children.Add(frame);
}

protected override bool OnBackButtonPressed()
{
    if (BeforePage != null)
    {
        MainPage.Current.Detail = new NavigationPage(BeforePage);
        App.Current.MainPage = MainPage.Current;
        return true;
    }
    return base.OnBackButtonPressed();
}
private async void GetArticlesProfile()
{
    var response = await GetArticlesProfileResponse();
    if(response.StatusCode==HttpStatusCode.OK)
    {
        var ArticlesJson = await
response.Content.ReadAsStringAsync();
        var Articles =
JsonConvert.DeserializeObject<List<PlaceHelper>>(ArticlesJson);
        CreateArticlesProfile(Articles);
    }
}

private void CreateArticlesProfile(List<PlaceHelper> articles)
{
    foreach(var article in articles)
    {
        Frame frame = new Frame();
        frame.CornerRadius = 20;
        frame.BackgroundColor = Color.LightGray;

        Label TitleArticle = new Label();
        TitleArticle.Text = article.Name;
        TitleArticle.TextColor = Color.Black;
        TitleArticle.FontSize = 18;

        Label AdressArticle = new Label();
        AdressArticle.Text = article.Adress;
        AdressArticle.TextColor = Color.DarkGray;
        AdressArticle.FontSize = 16;

        StackLayout stackLayout = new StackLayout();

        if (IDUser == App.IDCurrentUser)
        {
            Grid gridButtonsForArticle = new Grid();

            var columnFirst = new ColumnDefinition();
            columnFirst.Width = new GridLength(5, GridUnitType.Star);

            var columnSecond = new ColumnDefinition();

```

```

        columnSecond.Width = new GridLength(5,
GridUnitType.Star);

        gridButtonsForArticle.ColumnDefinitions.Add(columnFirst);

gridButtonsForArticle.ColumnDefinitions.Add(columnSecond);

        Button button = new Button();
        button.Text = "Удалить статью";
        button.BackgroundColor = Color.DarkGray;
        button.FontSize = 8;
        button.CornerRadius = 25;
        button.Clicked += async (s, e) =>
        {
            var result = await ConfirmDialog();
            if (result)
            {
                ArticlesProfile.Children.Remove(frame);
                DeleteArticle(article.IDArticle);
            }
        };
        Button buttonEditArticle = new Button();
        buttonEditArticle.Text = "Редактировать статью";
        buttonEditArticle.BackgroundColor = Color.DarkGray;
        buttonEditArticle.FontSize = 8;
        buttonEditArticle.CornerRadius = 25;
        buttonEditArticle.Clicked += async (s, e) =>
        {
            App.Current.MainPage = new NavigationPage(new
AddNews(this, article.IDArticle));
        };
        gridButtonsForArticle.Children.Add(button);
        gridButtonsForArticle.Children.Add(buttonEditArticle);

        Grid.SetColumn(button, 0);
        Grid.SetColumn(buttonEditArticle, 1);

        stackLayout.Children.Add(gridButtonsForArticle);
    }

    stackLayout.Children.Add(TitleArticle);
    stackLayout.Children.Add(AdressArticle);

    frame.Content = stackLayout;

    var tapGestureRecognizer = new TapGestureRecognizer();
    tapGestureRecognizer.NumberOfTapsRequired = 1;
    tapGestureRecognizer.Tapped += (s, e) =>
    {
        ShowArticle(article.IDArticle);
    };
    frame.GestureRecognizers.Add(tapGestureRecognizer);

    ArticlesProfile.Children.Add(frame);
}
}
private async void ShowArticle(int IDArticle)
{
    App.Current.MainPage = new NavigationPage(new
ViewArticle(this, IDArticle));
}
private async Task<HttpResponsMessage> GetArticlesProfileResponse()
{
    var client = new HttpClient();

```

```

        HttpResponseMessage responseMessage = null;
        try
        {
            responseMessage = await
client.GetAsync($"http://192.168.0.29:5000/GetArticlesUser/{IDUser}");
        }
        catch (Exception ex)
        {
            if (responseMessage == null)
            {
                responseMessage = new HttpResponseMessage();
            }
            responseMessage.StatusCode =
HttpStatusCode.InternalServerError;
            await DisplayAlert("Ошибка", "Подключение отсутствует",
"Заккрыть");
        }
        return responseMessage;
    }

    private async void GetDataProfile()
    {
        var response = await GetProfileData();
        if (response.StatusCode == HttpStatusCode.OK)
        {
            var CurrentProfileDataJson = await
response.Content.ReadAsStringAsync();
            var CurrentProfileData =
JsonConvert.DeserializeObject<ProfileData>(CurrentProfileDataJson);
            SetProfileView(CurrentProfileData);
            //await DisplayNews(News);
        }
    }

    private void SetProfileView(ProfileData currentProfileData)
    {
        if (currentProfileData.ImageProfile != null)
        {
            //Stream stream = new
MemoryStream(currentProfileData.ImageProfile);
            ImageProfile.Source = ImageSource.FromStream(() => new
MemoryStream(currentProfileData.ImageProfile));
        }
        LabelArticlesCount.Text =
currentProfileData.CountArticles.ToString();
        LabelNickname.Text = currentProfileData.Nickname;
        LabelSubscribers.Text =
currentProfileData.Subscribers.ToString();
        //LabelStatusCount.Text = currentProfileData.Status.ToString();
        if (currentProfileData.Description != null)
            LabelDescription.Text =
currentProfileData.Description.ToString();
    }

    private async Task<HttpResponseMessage> GetProfileData()
    {
        var client = new HttpClient();
        HttpResponseMessage responseMessage = null;
        try
        {
            responseMessage = await
client.GetAsync($"http://192.168.0.29:5000/GetProfile/{IDUser}");
        }
        catch (Exception ex)
    }

```

```

        {
            if (responseMessage == null)
            {
                responseMessage = new HttpResponseMessage();
            }
            responseMessage.StatusCode =
HttpStatusCode.InternalServerError;
            await DisplayAlert("Ошибка", "Подключение отсутствует",
"Заккрыть");
        }
        return responseMessage;
    }
    private void SetProfileView()
    {
        LineChoice.TranslateTo(0, 0);
        ScrollViewArticles.IsVisible = false;
        ScrollViewSubscribers.IsVisible = false;
        ScrollViewProfile.IsVisible = true;
    }
    private void SetArticlesView()
    {
        LineChoice.TranslateTo(LineChoice.Width + GridMenu.ColumnSpacing,
0);
        ScrollViewProfile.IsVisible = false;
        ScrollViewSubscribers.IsVisible = false;
        ScrollViewArticles.IsVisible = true;
    }
    private void SetSubscribersView()
    {
        LineChoice.TranslateTo((LineChoice.Width +
GridMenu.ColumnSpacing) * 2, 0);
        ScrollViewProfile.IsVisible = false;
        ScrollViewArticles.IsVisible = false;
        ScrollViewSubscribers.IsVisible = true;
    }
    private void Button_Clicked(object sender, EventArgs e)
    {
        SetProfileView();
    }
    private void Button_Clicked_1(object sender, EventArgs e)
    {
        SetArticlesView();
    }
    private void Button_Clicked_2(object sender, EventArgs e)
    {
        SetSubscribersView();
    }
    private void TapGestureRecognizer_Tapped(object sender, EventArgs e)
    {
        if (IDUser != App.IDCurrentUser)
            return;
        if (SettingImageProfile.IsVisible)
        {
            SettingImageProfile.IsVisible = false;
            return;
        }
        SettingImageProfile.IsVisible = true;
    }
    private void ImageButton_Clicked(object sender, EventArgs e)
    {

```

```

        SettingImageProfile.IsVisible = false;
    }
    public async void UpdatePhoto(ImageSource imagesrc)
    {
        //var Image = new CachedImage { Source = imagesrc, Aspect =
Aspect.AspectFill, DownsampleToViewSize = true };
        ImageProfile.Source = imagesrc;
        var response = await UpdatePhotoResponse(imagesrc);
        if(response.StatusCode!=HttpStatusCode.OK)
        {
            return;
        }
    }
    private void Button_Clicked_3(object sender, EventArgs e)
    {
        SettingImageProfile.IsVisible = false;
        Intent intent = new Intent(Intent.ActionSend);
        intent.SetType("image/*");
        intent.SetAction(Intent.ActionGetContent);

App.ActivityCurrent.StartActivityForResult(Intent.CreateChooser(intent,
"Select Picture"), 1);
    }
    private byte[] ImageSourceToBytes(ImageSource imageSource)
    {
        if (imageSource == null)
            return null;
        StreamImageSource streamImageSource =
(StreamImageSource)imageSource;
        System.Threading.CancellationToken cancellationToken =
System.Threading.CancellationToken.None;
        Task<Stream> task = streamImageSource.Stream(cancellationToken);
        Stream stream = task.Result;
        byte[] bytesAvailable = new byte[stream.Length];
        stream.Read(bytesAvailable, 0, bytesAvailable.Length);
        return bytesAvailable;
    }
    private async Task<HttpResponseMessage>
UpdatePhotoResponse(ImageSource image)
    {
        var client = new HttpClient();
        HttpResponseMessage responseMessage = null;
        var imageBytes = ImageSourceToBytes(image);
        JsonContent content = JsonContent.Create(new ProfileImage
{ID=App.IDCurrentUser, Image= imageBytes });
        try
        {
            responseMessage = await
client.PutAsync($"http://192.168.0.29:5000/UpdateImageProfile/", content);
        }
        catch (Exception ex)
        {
            if (responseMessage == null)
            {
                responseMessage = new HttpResponseMessage();
            }
            responseMessage.StatusCode =
HttpStatusCode.InternalServerError;
            await DisplayAlert("Ошибка", "Подключение отсутствует",
"Заккрыть");
        }
        return responseMessage;
    }

```

```

private void TapGestureRecognizer_Tapped_1(object sender, EventArgs
e)
{
    if (App.IDCurrentUser != IDUser)
        return;
    LabelDescription.IsVisible = false;
    EditorDescription.Text = LabelDescription.Text;
    EditorDescription.IsVisible = true;
    EditorDescription.Focus();
}

private void EditorDescription_Unfocused(object sender,
FocusEventArgs e)
{
    EditorDescription.IsVisible = false;
    LabelDescription.Text = EditorDescription.Text;
    LabelDescription.IsVisible = true;
}

private async void Button_Clicked_4(object sender, EventArgs e)
{
    ImageProfile.Source = null;
    SettingImageProfile.IsVisible = false;
    await UpdatePhotoResponse(null);
}

private async void DeleteArticle(int ID)
{
    var response = await DeleteArticleResponse(ID);
    if(response.IsSuccessStatusCode)
        await DisplayAlert("Результат", "Новость успешно удалена",
"Заккрыть");
    else
        await DisplayAlert("Результат", "При удалении новости
произошла ошибка", "Заккрыть");
}

private async Task<HttpResponseMessage> DeleteArticleResponse(int ID)
{
    var client = new HttpClient();
    HttpResponseMessage responseMessage = null;
    try
    {
        responseMessage = await
client.DeleteAsync($"http://192.168.0.29:5000/DeletaArticle/{ID}");
    }
    catch (Exception ex)
    {
        if (responseMessage == null)
        {
            responseMessage = new HttpResponseMessage();
        }
        responseMessage.StatusCode =
HttpStatusCode.InternalServerError;
        await DisplayAlert("Ошибка", "Подключение отсутствует",
"Заккрыть");
    }
    return responseMessage;
}

private async Task<bool> ConfirmDialog()
{
    var result =await DisplayAlert("Удаление статьи", "Вы
действительно хотите удалить статью?", "Да", "Нет");
    return result;
}
}

```

```

public class ProfileData
{
    public string Nickname { get; set; }
    public int Subscribers { get; set; }
    public int CountArticles { get; set; }
    public int Status { get; set; }
    public string Description { get; set; }
    public byte[] ImageProfile { get; set; }
}
public class ProfileImage
{
    public int ID { get; set; }
    public byte[] Image { get; set; }
}
}

```

Листинг В.24 – Файл Profile.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:yummy="clr-
namespace:Xamarin.Forms.PancakeView;assembly=Xamarin.Forms.PancakeView"
    xmlns:ffimageloading="clr-
namespace:FFImageLoading.Forms;assembly=FFImageLoading.Forms"
    x:Class="Authentication.Profile">
    <ContentPage.Content>
        <Grid x:Name="MainGrid" RowSpacing="0" ColumnSpacing="0"
        BackgroundColor="Black">
            <Grid.RowDefinitions>
                <RowDefinition Height="200*"></RowDefinition>
                <RowDefinition Height="90*"></RowDefinition>
                <RowDefinition Height="300*"></RowDefinition>
            </Grid.RowDefinitions>

            <yummy:PancakeView x:Name="FrameImage" BackgroundColor="Coral"
            CornerRadius="0,0,40,40" Grid.Row="0" Padding="0" IsClippedToBounds="True">
                <Grid>
                    <Grid.RowDefinitions>
                        <RowDefinition Height="*"></RowDefinition>
                        <RowDefinition Height="*"></RowDefinition>
                        <RowDefinition Height="*"></RowDefinition>
                        <RowDefinition Height="*"></RowDefinition>
                        <RowDefinition Height="*"></RowDefinition>
                    </Grid.RowDefinitions>
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="*"></ColumnDefinition>
                        <ColumnDefinition Width="*"></ColumnDefinition>
                        <ColumnDefinition Width="*"></ColumnDefinition>
                        <ColumnDefinition Width="*"></ColumnDefinition>
                        <ColumnDefinition Width="*"></ColumnDefinition>
                    </Grid.ColumnDefinitions>
                    <ffimageloading:CachedImage x:Name="ImageProfile"
                    BackgroundColor="LightGray" Aspect="AspectFill" Grid.ColumnSpan="5"
                    Grid.RowSpan="5">
                        <ffimageloading:CachedImage.GestureRecognizers>
                            <TapGestureRecognizer NumberOfTapsRequired="1"
                            Tapped="TapGestureRecognizer_Tapped"></TapGestureRecognizer>
                        </ffimageloading:CachedImage.GestureRecognizers>
                    </ffimageloading:CachedImage>
                </Grid>
            </yummy:PancakeView>
        </Grid>
    </ContentPage.Content>
</ContentPage>

```

```

        <Frame x:Name="SettingImageProfile" IsVisible="false"
        BackgroundColor="White" CornerRadius="15" Grid.Column="1" Grid.Row="1"
        Grid.RowSpan="3" Grid.ColumnSpan="3" IsClippedToBounds="True">
            <ScrollView>
                <StackLayout IsClippedToBounds="True"
                Spacing="1">
                    <ImageButton Clicked="ImageButton_Clicked"
                    Source="confirmIcon.png" WidthRequest="20" HeightRequest="20"
                    BackgroundColor="White" HorizontalOptions="End"></ImageButton>
                    <Button BackgroundColor="White"
                    HeightRequest="38" Text="Редактировать фото" Clicked="Button_Clicked_3"
                    FontSize="14" TextColor="Black"></Button>
                    <Button BackgroundColor="White"
                    HeightRequest="38" ContentLayout="Top,0" Text="Удалить фото"
                    Clicked="Button_Clicked_4" FontSize="14" TextColor="Black"></Button>
                </StackLayout>
            </ScrollView>
        </Frame>
    </Grid>
</yummy:PancakeView>
<Frame x:Name="FrameInformation" Background="Black" Grid.Row="1"
Padding="20" IsClippedToBounds="True">
    <Grid >
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="100*"></ColumnDefinition>
            <ColumnDefinition Width="100*"></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <Grid Grid.Column="0" RowSpacing="0">
            <Grid.RowDefinitions>
                <RowDefinition Height="50*"></RowDefinition>
                <RowDefinition Height="50*"></RowDefinition>
            </Grid.RowDefinitions>
            <Label x:Name="LabelSubscribers" Text="0"
            FontSize="32" FontAttributes="Bold" TextColor="White"
            HorizontalOptions="Center" Grid.Row="0"></Label>
            <Label Text="Подписчиков" FontSize="18"
            FontAttributes="Bold" TextColor="Gray" HorizontalOptions="Center"
            Grid.Row="1"></Label>
        </Grid>
        <Grid Grid.Column="1" RowSpacing="0">
            <Grid.RowDefinitions>
                <RowDefinition Height="50*"></RowDefinition>
                <RowDefinition Height="50*"></RowDefinition>
            </Grid.RowDefinitions>
            <Label x:Name="LabelArticlesCount" Text="0"
            FontSize="32" FontAttributes="Bold" TextColor="White"
            HorizontalOptions="Center" Grid.Row="0"></Label>
            <Label Text="Новостей" FontSize="18"
            FontAttributes="Bold" TextColor="Gray" HorizontalOptions="Center"
            Grid.Row="1"></Label>
        </Grid>
    </Grid>
</Frame>
<yummy:PancakeView x:Name="FrameData" BackgroundColor="White"
CornerRadius="40,40,0,0" IsClippedToBounds="True" Grid.Row="2">
    <Grid IsClippedToBounds="True">
        <Grid.RowDefinitions>
            <RowDefinition Height="100*"></RowDefinition>
            <RowDefinition Height="500*"></RowDefinition>
        </Grid.RowDefinitions>
        <Grid Grid.Row="0" BackgroundColor="LightGray"
        Padding="15,5,15,0" RowSpacing="0" x:Name="GridMenu">
            <Grid.ColumnDefinitions>

```

```

        <ColumnDefinition
Width="100*"></ColumnDefinition>
        <ColumnDefinition
Width="100*"></ColumnDefinition>
        <ColumnDefinition
Width="100*"></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="150*"></RowDefinition>
        <RowDefinition Height="15"></RowDefinition>
    </Grid.RowDefinitions>
    <Button x:Name="ButtonProfile" Text="Профиль"
Grid.Column="0" Grid.Row="0" BackgroundColor="LightGray" CornerRadius="25"
Clicked="Button_Clicked"></Button>
    <Button x:Name="ButtonArticles" Text="Новости"
Grid.Column="1" Grid.Row="0" BackgroundColor="LightGray" CornerRadius="25"
Clicked="Button_Clicked_1"></Button>
    <Button x:Name="ButtonStatistics" Text="Подписки"
Grid.Column="2" Grid.Row="0" BackgroundColor="LightGray" CornerRadius="25"
Clicked="Button_Clicked_2"></Button>
    <Line x:Name="LineChoice" Stroke="Gray" Grid.Row="1"
X1="0" X2="150" StrokeThickness="5"></Line>
</Grid>
<Grid x:Name="GridBar" Grid.Row="1">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*"></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <ScrollView x:Name="ScrollViewProfile">
        <Grid RowSpacing="0">
            <Grid.RowDefinitions>
                <RowDefinition
Height="55"></RowDefinition>
                <RowDefinition
Height="55"></RowDefinition>
                <RowDefinition
Height="33*"></RowDefinition>
            </Grid.RowDefinitions>
            <Label x:Name="LabelNickname" Text="Nickname"
Margin="20,20,0,0" HorizontalOptions="Start" FontSize="20" TextColor="Black"
FontAttributes="Bold" Grid.Row="0"></Label>
            <Label Text="About" Margin="20,20,0,0"
HorizontalOptions="Start" FontSize="20" TextColor="Black"
FontAttributes="Bold" Grid.Row="1"></Label>
            <Label x:Name="LabelDescription"
Text="Описание отсутствует" Margin="20,0,0,0" HorizontalOptions="Start"
FontSize="20" TextColor="Black" Grid.Row="2">
                <Label.GestureRecognizers>
                    <TapGestureRecognizer
NumberOfTapsRequired="1"
Tapped="TapGestureRecognizer_Tapped_1"></TapGestureRecognizer>
                </Label.GestureRecognizers>
            </Label>
            <Editor x:Name="EditorDescription"
Margin="20,0,0,0" Unfocused="EditorDescription_Unfocused" Grid.Row="2"
FontSize="20" TextColor="Black" IsVisible="false"></Editor>
        </Grid>
    </ScrollView>
    <ScrollView x:Name="ScrollViewArticles"
IsVisible="false" IsClippedToBounds="True" Margin="0">
        <StackLayout x:Name="ArticlesProfile"
Margin="10,0,10,10">

            </StackLayout>
        </ScrollView>

```

```

        <ScrollView x:Name="ScrollViewSubscribers"
IsVisible="false" IsClippedToBounds="True" Margin="0">
            <StackLayout x:Name="SubscribersProfile"
Margin="10,0,10,10">

                </StackLayout>
            </ScrollView>
        </Grid>
    </Grid>
</yummy:PancakeView>
</Grid>
</ContentPage.Content>
</ContentPage>

```

Приложение Г
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ СЕРВЕРНОЙ ЧАСТИ
(рекомендуемое)

The screenshot shows a REST client interface with the URL `http://192.168.0.29:5000/Register`. The request is a POST method. The body is set to 'Text' and contains the following JSON:

```
1 {
2   "login": "NewUserLogin",
3   "password": "NewUserPassword",
4   "nickname": "NewUserNickname"
5 }
```

The response status is 200 OK, with a response time of 4.64 s and a body size of 144 B. The response is displayed in 'Text' format.

Рисунок Г.1 – Регистрация нового пользователя

The screenshot shows a REST client interface with the URL `http://192.168.0.29:5000/Authorization`. The request is a POST method. The body is set to 'Text' and contains the following JSON:

```
1 {
2   "login": "admin",
3   "password": "admin",
4 }
```

The response status is 200 OK, with a response time of 685 ms and a body size of 143 B. The response is displayed in 'Text' format.

Рисунок Г.2 – Авторизация пользователя

http://192.168.0.29:5000/NewsOfMap

GET http://192.168.0.29:5000/NewsOfMap

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

200 OK 3.50 s 2.8 KB Save as Example

Pretty Raw Preview Visualize

```
[{"IDArticle":15,"Latitude":56.31275942281241,"Longitude":43.9868176728487,"Adress":"Maxim Gorky St, 63 262, Nizhniy Novgorod, Nizhegorodskaya oblast, Russia, 603155","Name":"новость из одного абзаца","PublishedDate":"2023-03-14T16:33:56.68"}, {"IDArticle":14,"Latitude":56.837539,"Longitude":60.592848,"Adress":"г Екатеринбург, пр-кт Ленина, д 17","Name":"экологическая катастрофа с отходами около жилых домов","PublishedDate":"2023-03-07T17:29:00.06"}, {"IDArticle":12,"Latitude":46.86188060152884,"Longitude":64.85762480646372,"Adress":"Ulytau District, Kazakhstan","Name":"рестовая статья","PublishedDate":"2023-02-05T00:00:00"}, {"IDArticle":13,"Latitude":41.82407996979188,"Longitude":12.487377114593983,"Adress":"Via Giuseppe Armellini, 37 sc. A, 00143 Roma RM, Italy","Name":"моя новость","PublishedDate":"2023-02-05T00:00:00"}, {"IDArticle":10,"Latitude":55.7674794696951,"Longitude":37.69458498805761,"Adress":"Красноказарменная Ул., С17, 17СТР, Moskva, Moscow Oblast, Russia, 111250","Name":"мой заголовок","PublishedDate":"2023-01-23T00:00:00"}, {"IDArticle":4,"Latitude":53.2151828898339,"Longitude":44.98578730970621,"Adress":"Prospekt Pobedy, 4-a, Penza, Penzenskaya oblast, Russia, 440011","Name":"Происшествие с двумя машинами","PublishedDate":"2023-01-22T00:00:00"}, {"IDArticle":5,"Latitude":41.90424264511552,"Longitude":12.497552745044231,"Adress":"Piazza delle Finanze, 4616, 00185 Roma RM, Italy","Name":"1111","PublishedDate":"2023-01-22T00:00:00"}, {"IDArticle":6,"Latitude":41.92896735563983,"Longitude":12.466932982206345,"Adress":"Via Guido Reni, 4 A, 00196 Roma RM, Italy","Name":"2222","PublishedDate":"2023-01-22T00:00:00"}, {"IDArticle":7,"Latitude":41.85338598461619,"Longitude":12.527432553470135,"Adress":"Appia Pignatelli/Almone, 00178 Roma RM, Italy","Name":"2222134","PublishedDate":"2023-01-22T00:00:00"}, {"IDArticle":8,"Latitude":41.87723443662578,"Longitude":12.458792477846146,"Adress":"Via Antonio Cesari, 54, 00152 Roma RM, Italy","Name":"мой новый","PublishedDate":"2023-01-22T00:00:00"}, {"IDArticle":9,"Latitude":43.585525,"Longitude":39.723062,"Adress":"Краснодарский край, г Сочи","Name":"икрео","PublishedDate":"2023-01-22T00:00:00"}, {"IDArticle":1,"Latitude":53.175368,"Longitude":45.034827,"Adress":"г Пенза","Name":"мой заголовок 2","PublishedDate":"2023-01-12T00:00:00"}, {"IDArticle":2,"Latitude":59.939084,"Longitude":30.315879,"Adress":"г Санкт-Петербург","Name":"вторая новость","PublishedDate":"2023-01-12T00:00:00"}]
```

Рисунок Г.3 – Получения информации об актуальных новостях

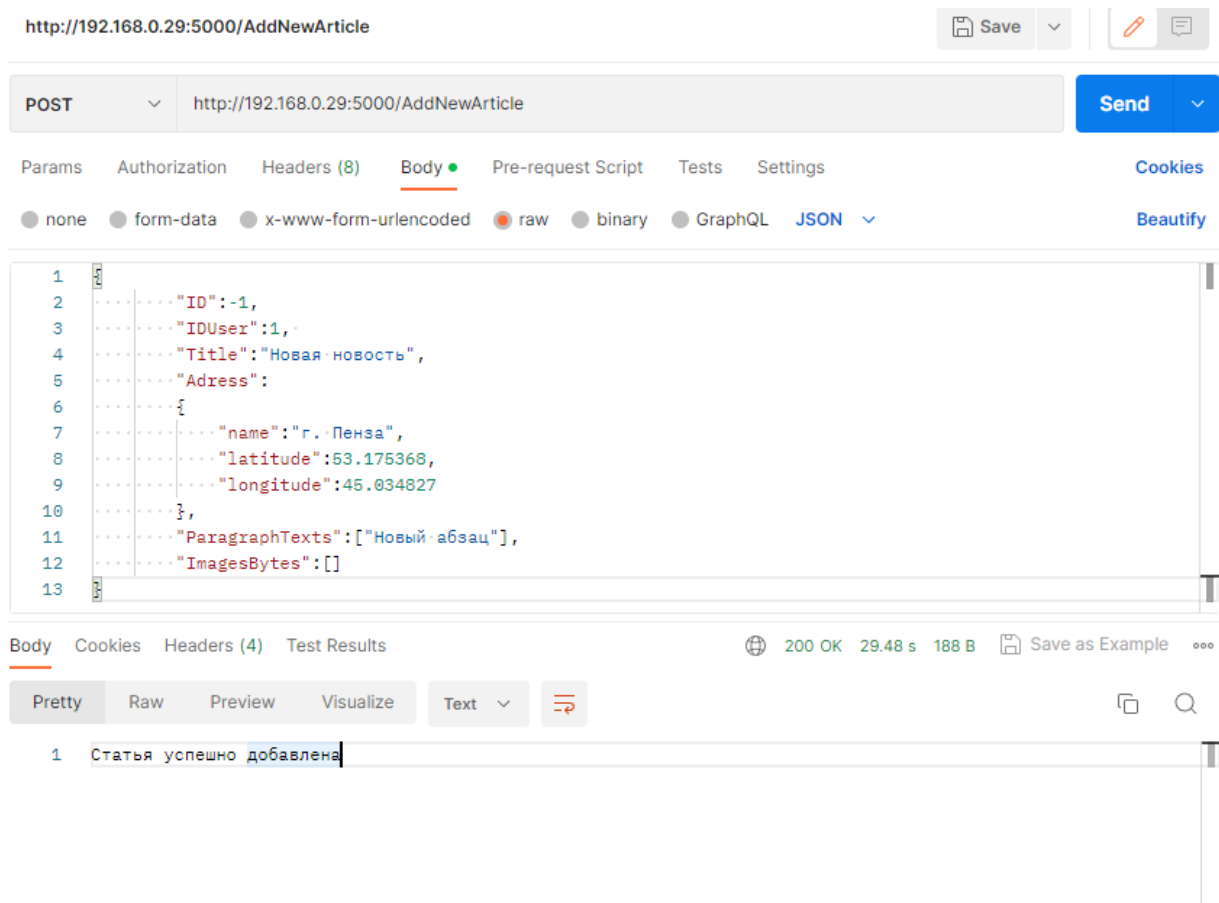


Рисунок Г.4 – Создание новой новости

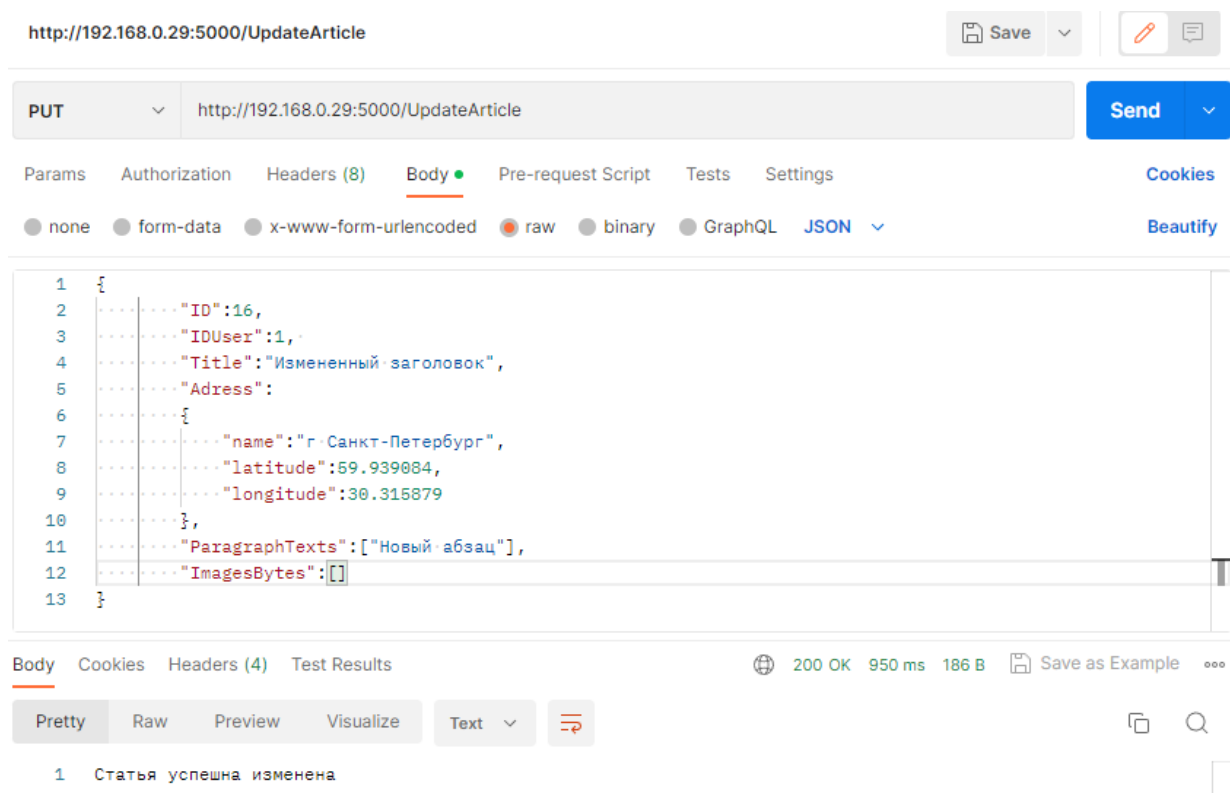


Рисунок Г.5 – Редактирование новости

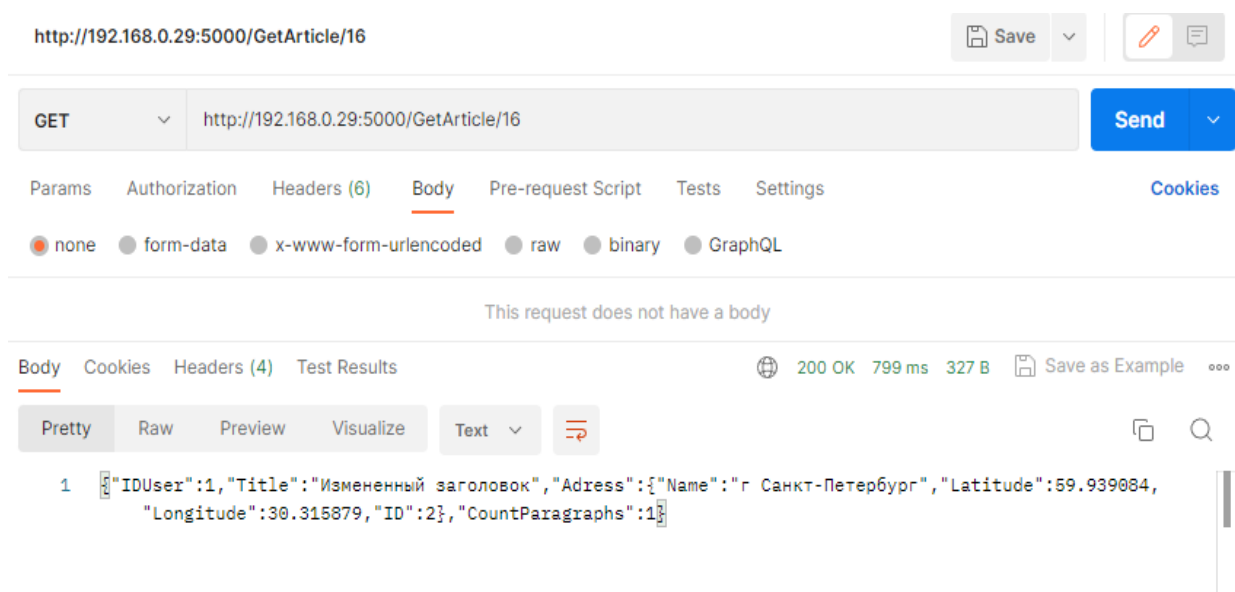


Рисунок Г.6 – Получение информации о новости

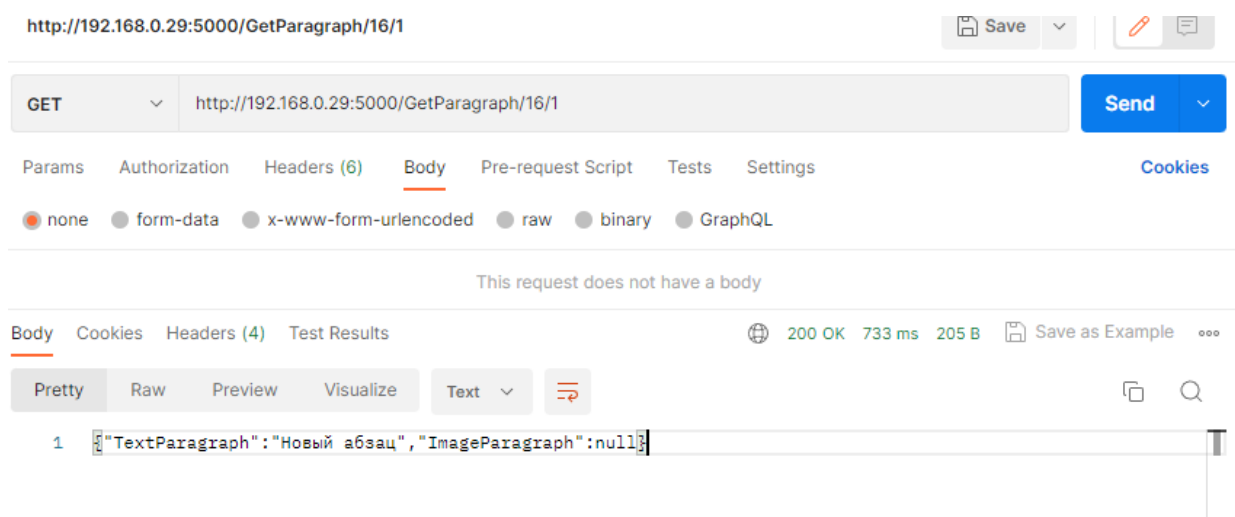


Рисунок Г.7 – Получение данных отдельного взятого параграфа
определенной новости

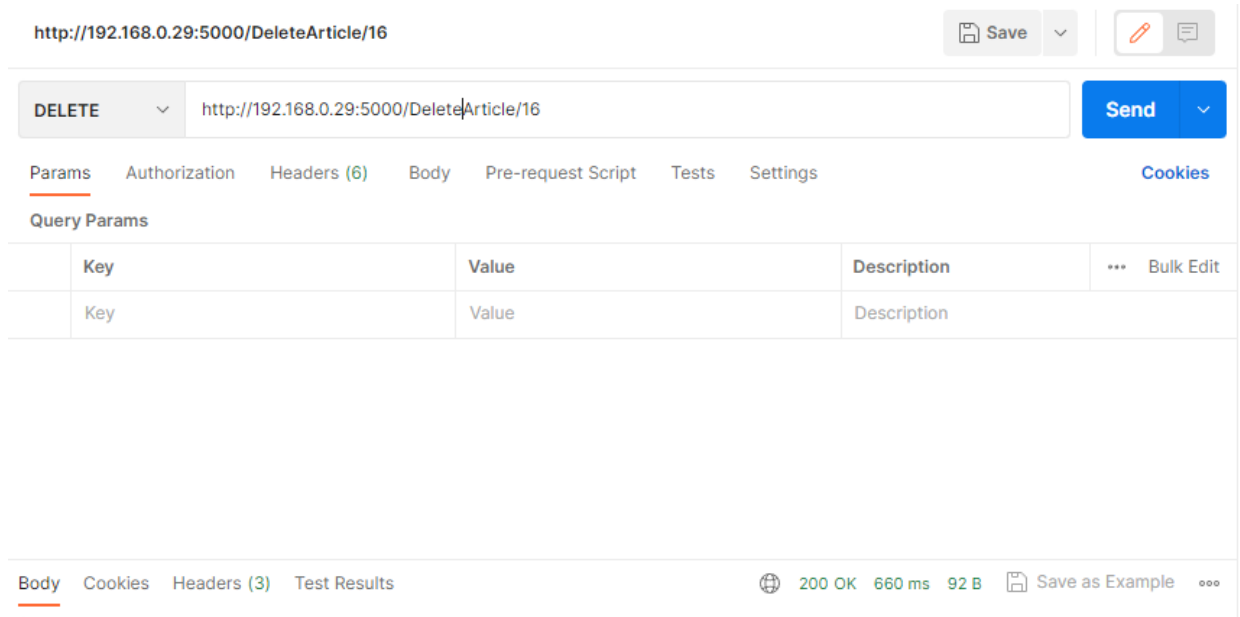


Рисунок Г.8 – Удаление новости из системы

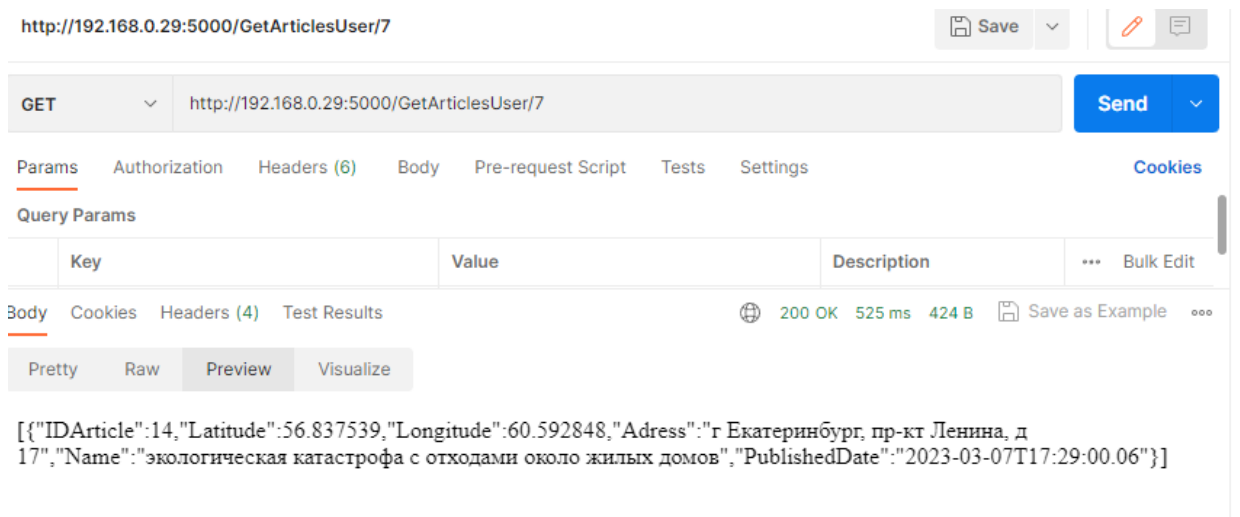


Рисунок Г.9 – Получение информации о новостях определенного пользователя

http://192.168.0.29:5000/GetProfile/1 Save

GET ▼ http://192.168.0.29:5000/GetProfile/1 Send ▼

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit

Body Cookies Headers (4) Test Results 200 OK 727 ms 8.03 MB Save as Example ...

Pretty Raw Preview Visualize Text ▼

```

1 [{"Nickname": "Данила", "Subscribers": 1, "CountArticles": 12, "Status": 50, "Description": null, "ImageProfile": "/
  9j/4AAQSkZJRgABAQAAQABAAQ/
  4gIoSUNDX1BST0ZJTEUAAQEAIAAAIQAABtnRyUkdCIFhZWiAAAAAAAAAAAAAAAAABhY3NwAAAAAAAAAAAAAAAAAAAAAAAA
  AAAAAAAAAAQA9tYAAQAAADTLQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAlkZXNj
  AAAA8AAAAHRyWFlaAAABZAAAABRnWFlaAAABeAAAAABRiWFlaAAABjAAAABRyVFJDAAABoAAAAAChnVFJDAAABoAAAAChiVFJDAAABoA
  AACh3dHB0AAABYAAAABRjcHJ0AAAB3AAAAADxtbHVjAAAAAAAAAAEAAAMZW5UwAAAFgAAAAcAHMAUgBHAETIAAAAAAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFhZWiAAAAAAAA
  
```

Рисунок Г.10 – Получение информации о профили пользователя

http://192.168.0.29:5000/UpdateImageProfile Save

PUT ▼ http://192.168.0.29:5000/UpdateImageProfile Send ▼

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings Cookies

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL **JSON** ▼ Beautify

```

1 [
2   ... "ID": 1,
3   ... "Image": null
4 ]

```

Body Cookies Headers (3) Test Results 200 OK 777 ms 92 B Save as Example ...

Рисунок Г.11 – Изменение изображение в профили пользователя

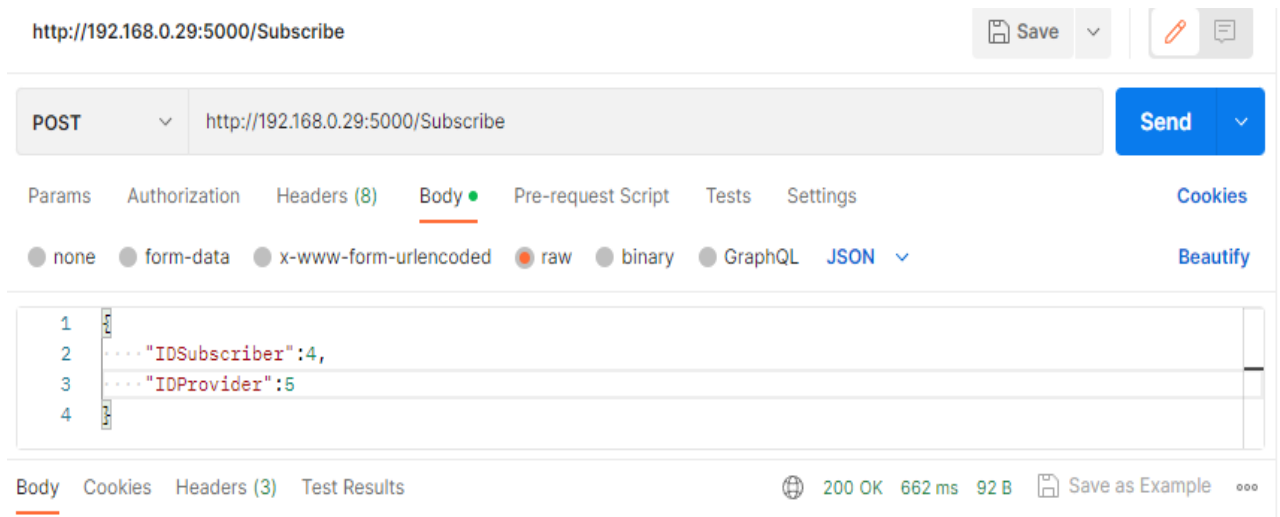


Рисунок Г.12 – Осуществление подписки пользователей

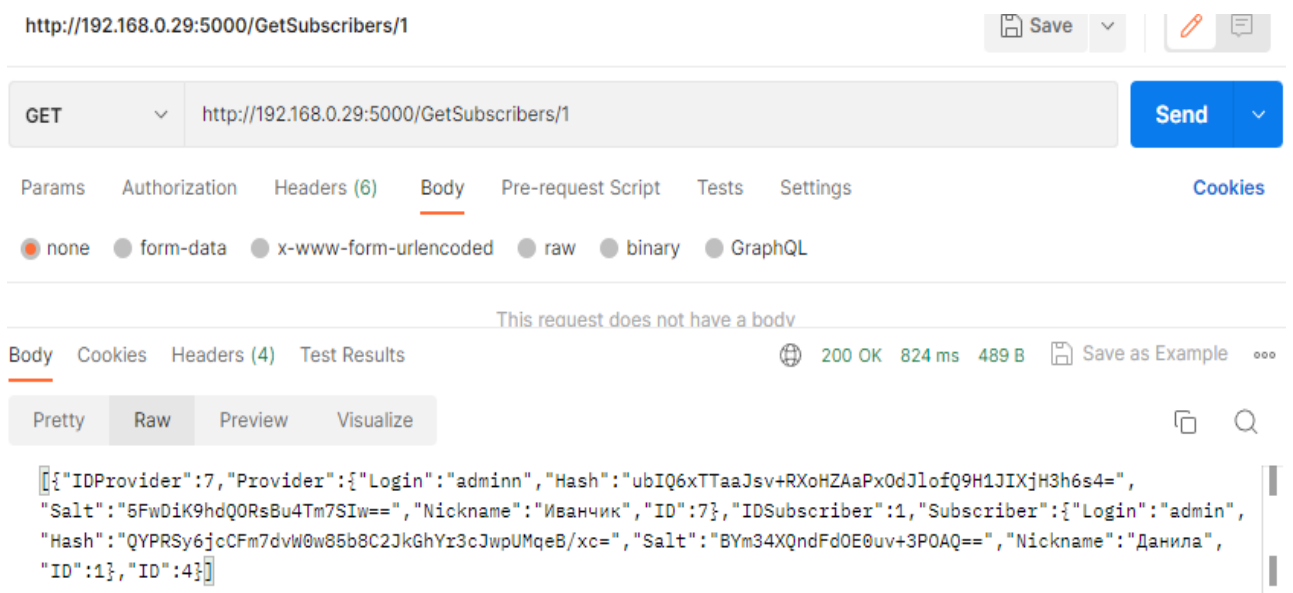


Рисунок Г.13 – Получение подписок определенного пользователя

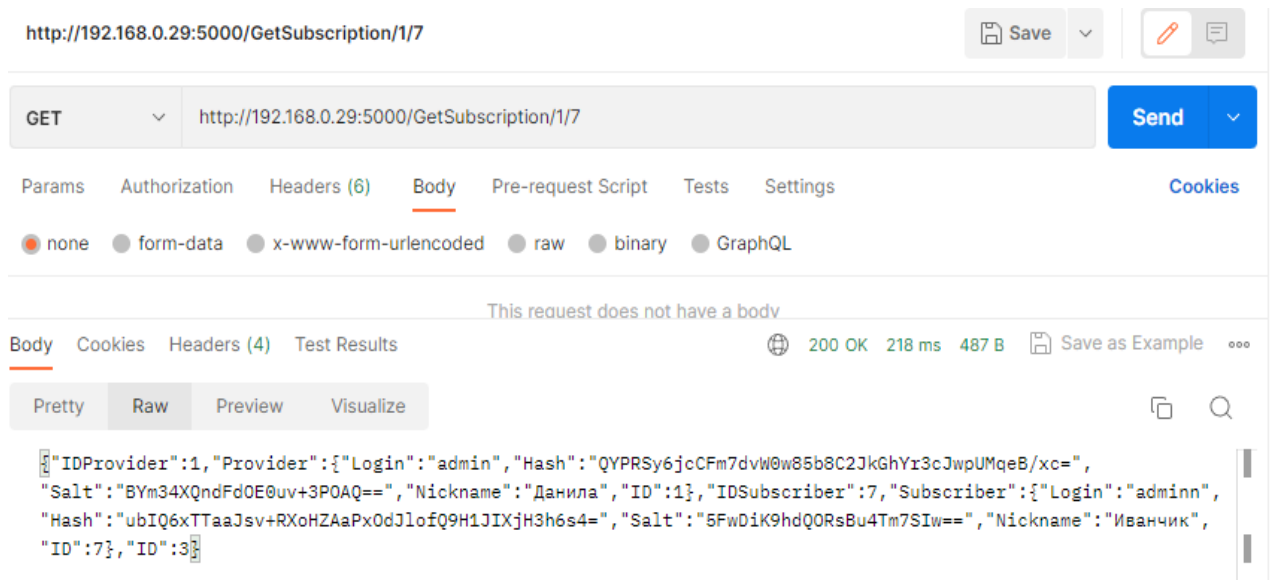


Рисунок Г.14 – Получение подписки одного пользователя на другого пользователя

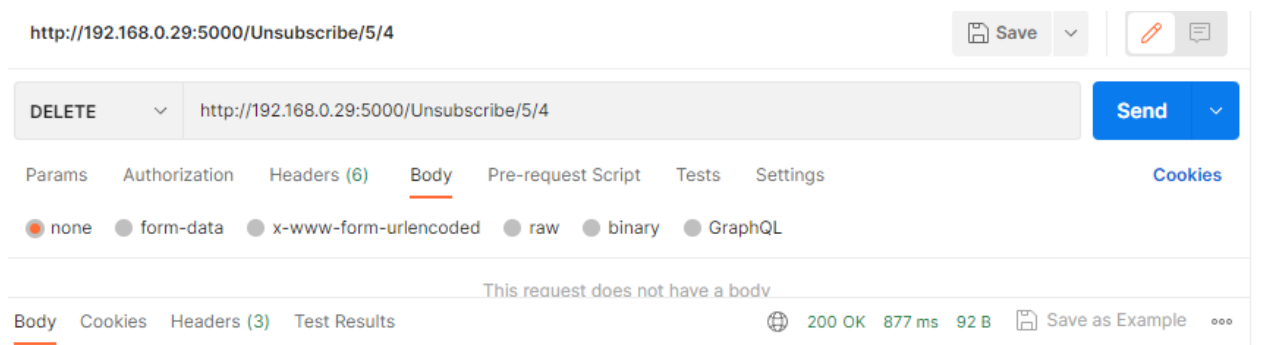


Рисунок Г.15 – Отписка одного пользователя от другого пользователя

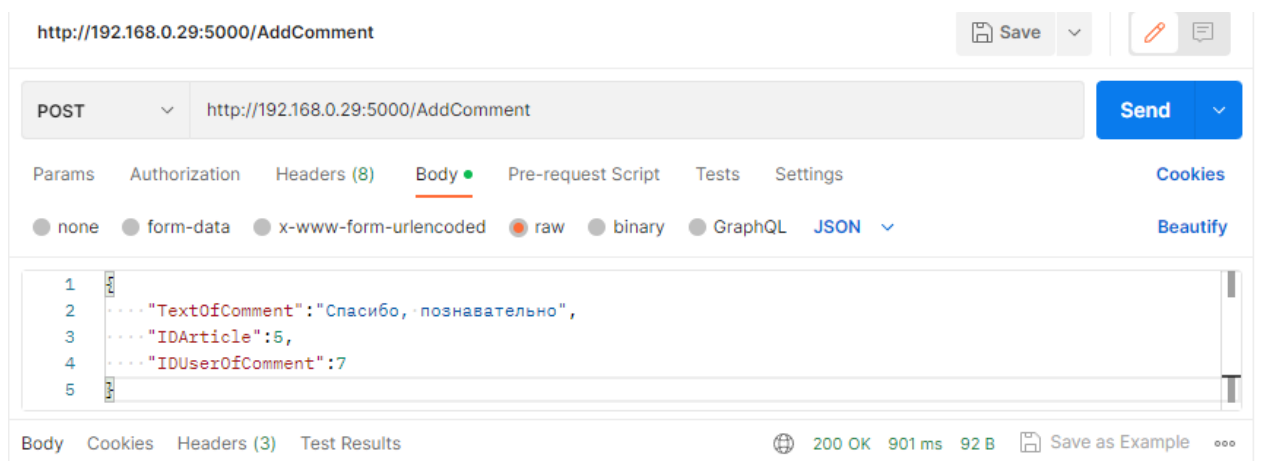


Рисунок Г.16 – Добавление нового комментария к новости

http://192.168.0.29:5000/GetComments/5

GET http://192.168.0.29:5000/GetComments/5 Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

Body Cookies Headers (4) Test Results 200 OK 700 ms 2.93 KB Save as Example

Pretty Raw Preview Visualize

```
[{"TextOfComment": "очень интересная статья, побольше бы таких))", "IDUserOfComment": 1, "User": {"Login": "admin", "Hash": "QYPRSy6jcCFm7dvW0w85b8C2JkGhYr3cJwpUMqeB/xc=", "Salt": "BYm34XQndFd0E0uv+3POAQ==", "Nickname": "Данила", "ID": 1}, "IDArticle": 5, "Article": {"Title": "1111", "Date": "2023-01-22T00:00:00", "IDUser": 1, "User": {"Login": "admin", "Hash": "QYPRSy6jcCFm7dvW0w85b8C2JkGhYr3cJwpUMqeB/xc=", "Salt": "BYm34XQndFd0E0uv+3POAQ==", "Nickname": "Данила", "ID": 1}, "IDPosition": 4, "Position": {"Name": "Piazza delle Finanze, 4616, 00185 Roma RM, Italy", "Latitude": 41.90424264511552, "Longitude": 12.497552745044231, "ID": 4}, "ID": 5}, "ID": 1}, {"TextOfComment": "очень интересная статья, побольше бы таких)) оаоаоаоа провёл словно аншлаг Алан аниме мотив садик вод уж взвод был перегруз лыжных оценила взад завоз заведения", "IDUserOfComment": 1, "User": {"Login": "admin", "Hash": "QYPRSy6jcCFm7dvW0w85b8C2JkGhYr3cJwpUMqeB/xc=", "Salt": "BYm34XQndFd0E0uv+3POAQ==", "Nickname": "Данила", "ID": 1}, "IDArticle": 5, "Article": {"Title": "1111", "Date": "2023-01-22T00:00:00", "IDUser": 1, "User": {"Login": "admin", "Hash": "QYPRSy6jcCFm7dvW0w85b8C2JkGhYr3cJwpUMqeB/xc=", "Salt": "BYm34XQndFd0E0uv+3POAQ==", "Nickname": "Данила", "ID": 1}, "IDPosition": 4, "Position": {"Name": "Piazza delle Finanze, 4616, 00185 Roma RM, Italy", "Latitude": 41.90424264511552, "Longitude": 12.497552745044231, "ID": 4}, "ID": 5}, "ID": 2}, {"TextOfComment": "мда, сразу видно комменты накручены, статья гавно, автор тоже", "IDUserOfComment": 7, "User": {"Login": "adminn", "Hash": "ubIQ6xTTaaJsv+RXoHZAaPxOdJlofQ9H1JIXjH3h6s4=", "Salt": "5FwDiK9hdQ0RsBu4Tm7SIw==", "Nickname": "Иванчик", "ID": 7}, "IDArticle": 5, "Article": {"Title": "1111", "Date": "2023-01-22T00:00:00", "IDUser": 1, "User": {"Login": "admin", "Hash": "QYPRSy6jcCFm7dvW0w85b8C2JkGhYr3cJwpUMqeB/xc=", "Salt": "BYm34XQndFd0E0uv+3POAQ==", "Nickname": "Данила", "ID": 1}, "IDPosition": 4, "Position": {"Name": "Piazza delle Finanze, 4616, 00185 Roma RM, Italy", "Latitude": 41.90424264511552, "Longitude": 12.497552745044231, "ID": 4}, "ID": 5}, "ID": 3}, {"TextOfComment": "Спасибо, познавательно", "IDUserOfComment": 7, "User": {"Login": "adminn", "Hash": "ubIQ6xTTaaJsv+RXoHZAaPxOdJlofQ9H1JIXjH3h6s4=", "Salt": "5FwDiK9hdQ0RsBu4Tm7SIw==", "Nickname": "Иванчик", "ID": 7}, "IDArticle": 5, "Article": {"Title": "1111", "Date": "2023-01-22T00:00:00", "IDUser": 1, "User": {"Login": "admin", "Hash": "QYPRSy6jcCFm7dvW0w85b8C2JkGhYr3cJwpUMqeB/xc=", "Salt": "BYm34XQndFd0E0uv+3POAQ==", "Nickname": "Данила", "ID": 1}, "IDPosition": 4, "Position": {"Name": "Piazza delle Finanze, 4616, 00185 Roma RM, Italy", "Latitude": 41.90424264511552, "Longitude": 12.497552745044231, "ID": 4}, "ID": 5}, "ID": 5}]
```

Рисунок Г.17 – Получение всех комментариев для определенной новости

http://192.168.0.29:5000/SetEvaluation

POST http://192.168.0.29:5000/SetEvaluation Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

Body Cookies Headers (3) Test Results 200 OK 720 ms 92 B Save as Example

```
1 {
2   ... "IsLike": true,
3   ... "IDUser": 5,
4   ... "IDArticle": 5
5 }
```

Рисунок Г.18 – Добавления положительной оценки к новости

http://192.168.0.29:5000/GetEvaluations/5 Save

GET ▼ http://192.168.0.29:5000/GetEvaluations/5 Send ▼

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL **JSON** ▼ Beautify

1

Body Cookies Headers (4) Test Results 200 OK 739 ms 2.36 KB Save as Example ...

Pretty Raw Preview Visualize

```
[{"IsLike":true,"IDUser":7,"User":{"Login":"adminn","Hash":"ubIQ6xTTaaJsv+RXoHZAApXOdJlofQ9H1JIXjH3h6s4=",
"Salt":"5FwDiK9hdQORsBu4Tm7SIw==","Nickname":"Иванчик","ID":7},"IDArticle":5,"Article":{"Title":"1111",
"Date":"2023-01-22T00:00:00","IDUser":1,"User":{"Login":"admin",
"Hash":"QYPRSy6jcCFm7dvW0w85b8C2JkGhYr3cJwpUMqeB/xc=","Salt":"BYm34XQndFd0E0uv+3POAQ==","Nickname":"Данила",
"ID":1},"IDPosition":4,"Position":{"Name":"Piazza delle Finanze, 4616, 00185 Roma RM, Italy","Latitude":41.
90424264511552,"Longitude":12.497552745044231,"ID":4},"ID":5},"ID":5},"IsLike":false,"IDUser":8,"User":
{"Login":"adminnn","Hash":"zboq0a+v1FLDweZSKMV9LHaGa5+F+XQYXEY1xkojGro=","Salt":"q2PtDNH4fYfF6tKM4vCe3g==",
"Nickname":"третий","ID":8},"IDArticle":5,"Article":{"Title":"1111","Date":"2023-01-22T00:00:00","IDUser":1,
"User":{"Login":"admin","Hash":"QYPRSy6jcCFm7dvW0w85b8C2JkGhYr3cJwpUMqeB/xc=","Salt":"BYm34XQndFd0E0uv
+3POAQ==","Nickname":"Данила","ID":1},"IDPosition":4,"Position":{"Name":"Piazza delle Finanze, 4616, 00185
Roma RM, Italy","Latitude":41.90424264511552,"Longitude":12.497552745044231,"ID":4},"ID":5},"ID":9},
{"IsLike":true,"IDUser":1,"User":{"Login":"admin","Hash":"QYPRSy6jcCFm7dvW0w85b8C2JkGhYr3cJwpUMqeB/xc=",
"Salt":"BYm34XQndFd0E0uv+3POAQ==","Nickname":"Данила","ID":1},"IDArticle":5,"Article":{"Title":"1111",
"Date":"2023-01-22T00:00:00","IDUser":1,"User":{"Login":"admin",
"Hash":"QYPRSy6jcCFm7dvW0w85b8C2JkGhYr3cJwpUMqeB/xc=","Salt":"BYm34XQndFd0E0uv+3POAQ==","Nickname":"Данила",
"ID":1},"IDPosition":4,"Position":{"Name":"Piazza delle Finanze, 4616, 00185 Roma RM, Italy","Latitude":41.
90424264511552,"Longitude":12.497552745044231,"ID":4},"ID":5},"ID":47},"IsLike":true,"IDUser":5,"User":
{"Login":"gena2","Hash":"BY6GWPBANUEleyqXKmcCl34YvO0mLDFwLdrZH9MHo6o=","Salt":"HbPMx/uLLBjiCwttq0+5Zw==",
"Nickname":"Геннадий","ID":5},"IDArticle":5,"Article":{"Title":"1111","Date":"2023-01-22T00:00:00","IDUser":1,
"User":{"Login":"admin","Hash":"QYPRSy6jcCFm7dvW0w85b8C2JkGhYr3cJwpUMqeB/xc=","Salt":"BYm34XQndFd0E0uv
+3POAQ==","Nickname":"Данила","ID":1},"IDPosition":4,"Position":{"Name":"Piazza delle Finanze, 4616, 00185
Roma RM, Italy","Latitude":41.90424264511552,"Longitude":12.497552745044231,"ID":4},"ID":5},"ID":51}]
```

Рисунок Г.19 – Получения всех оценок для определенной новости

http://192.168.0.29:5000/DeleteEvaluation/5/1 Save

DELETE ▼ http://192.168.0.29:5000/DeleteEvaluation/5/1 Send ▼

Params Authorization Headers (6) **Body** Pre-request Script Tests Settings Cookies

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL

This request does not have a body

Body Cookies Headers (3) Test Results 200 OK 785 ms 92 B Save as Example ...

Рисунок Г.20 – Удаление оценки у новости от определенного пользователя

http://192.168.0.29:5000/GetNewsVerification?KEY=OEZST8J0X32ZDCTN83R6P7O79MAEF8ZM

GET http://192.168.0.29:5000/GetNewsVerification?KEY=OEZST8J0X32ZDCTN83R6P7O79MAEF8ZM

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
☒	KEY	OEZST8J0X32ZDCTN83R6P7O79MAEF8ZM			
	Key	Value	Description		

Body Cookies Headers (4) Test Results 200 OK 743 ms 1.49 MB Save as Example

Pretty Raw Preview Visualize Text

```
1 [{"NumberInArticle":1,"TextParagraph":"query",
  "ImageParagraph":"iVBORw0KGgoAAAANSUHEUgAAAU4AAAPoCAIAAAAUew0AAAAAXNSR0IArs4c6QAAAAANzQk1UCAGI2
  +FP4AAAIABJREFUeJzsvWuvJcJyHbZWRFbtvc/pnp65D/LyYKUbD0MQf5gG4Zh+w/4p/h/
  GrABfRIESwYNSiJ1Sd47Mz3dfR57V2VghD9EZu7quZfkFWCljCFRu0cfWzXIysrYswKFZH83/7X/15VAZQii+jbh0vYbmagu3vY/
  vDwo0Q/+if/zW/94Y9/9Ae/+9Pf/dkXP/o9ffw9BBH2p//7//HH/+c/4/5cCtyqmbm7iEQEzPe9Ww2ny7osiygtota6lkJy7jb/
  d3dVRdDdSsrZsZsQKSn7TWVHVdV5IR/Vy5T0QggqSg5gFFJAARITUich9V1fW0b7FH+ed/
  8fSvfv7ytG21tfP53Gx3dwB56ggTEVVdt0QPAC+xi9h3X9c19ycpCADubmbndT2dFrN6Wh6+eL0utDdLuai/Wf3xZAvxWE5SiIhS
  +HBSxh7mZRGh173lmAcMHnOgOC6mtSYigKuqiJi1UoqKuLtQIQAFebAaywIKzRCBteRlkxHesBaSgIMaRRiAKBhBUMaegYgAACEDQZ
  AEC4BQITWEhIYS1tg8ilAF1KBSS5A5MhCBFEg
```

Рисунок Г.21 – Получение новости для модерации

http://192.168.0.29:5000/AssignSolutionNews

PUT http://192.168.0.29:5000/AssignSolutionNews

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
2 ..... "KeyOfModerator": "OEZST8J0X32ZDCTN83R6P7O79MAEF8ZM",
3 ..... "IsAcceptable": false,
4 ..... "IDArticle": 5,
5 ..... "IDAuthor": 1
6
```

Body Cookies Headers (3) Test Results 200 OK 389 ms 92 B Save as Example

Рисунок Г.22 – Установка решения по новости для публикации

http://192.168.0.29:5000/GetKey?KEY=OEZST8J0X32ZDCTN83R6P7O79MAEF8ZM

GET http://192.168.0.29:5000/GetKey?KEY=OEZST8J0X32ZDCTN83R6P7O79MAEF8ZM

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
☒	KEY	OEZST8J0X32ZDCTN83R6P7O79MAEF8ZM			
	Key	Value	Description		

Body Cookies Headers (4) Test Results 200 OK 1043 ms 174 B Save as Example

Pretty Raw Preview Visualize Text

```
1 91J1BX3GGV06KX8K0VQXE8HEWW4AINIJ
```

Рисунок Г.23 – Получение нового ключа для модерации

http://192.168.0.29:5000/Ban

POST http://192.168.0.29:5000/Ban

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Beautify

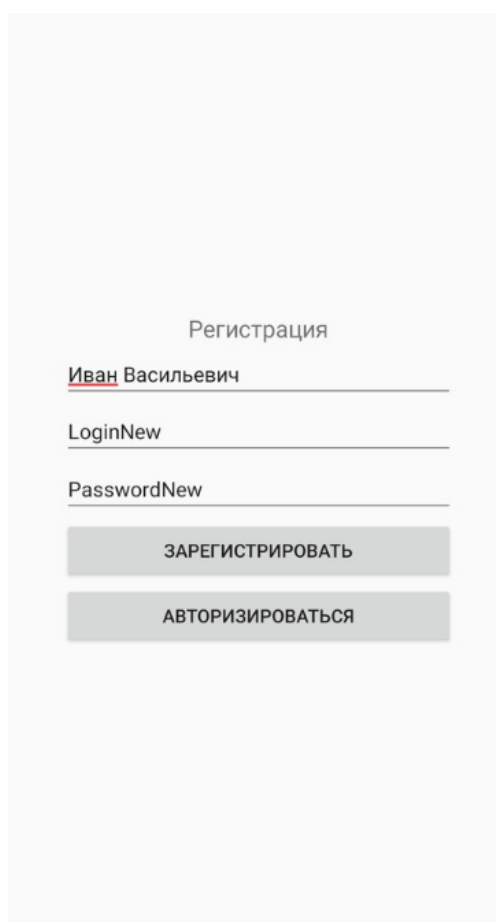
none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "IDReason": 1,
3   "IDUser": 10,
4   "KEY": "OEZST8J0X32ZDCTN83R6P7O79MAEF8ZM"
5 }
```

Body Cookies Headers (3) Test Results 200 OK 812 ms 92 B Save as Example

Рисунок Г.24 – Блокировка пользователя

Приложение Д
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ
(рекомендуемое)



Регистрация

Иван Васильевич

LoginNew

PasswordNew

ЗАРЕГИСТРИРОВАТЬ

АВТОРИЗИРОВАТЬСЯ

The image shows a registration form on a light gray background. At the top, the word 'Регистрация' (Registration) is centered. Below it, there are three input fields. The first field contains the text 'Иван Васильевич', with 'Иван' underlined in red. The second field contains 'LoginNew'. The third field contains 'PasswordNew'. Below these fields are two gray buttons. The top button is labeled 'ЗАРЕГИСТРИРОВАТЬ' (Register) and the bottom button is labeled 'АВТОРИЗИРОВАТЬСЯ' (Log In).

Рисунок Д.1 – Введенные данные для успешной регистрации пользователя

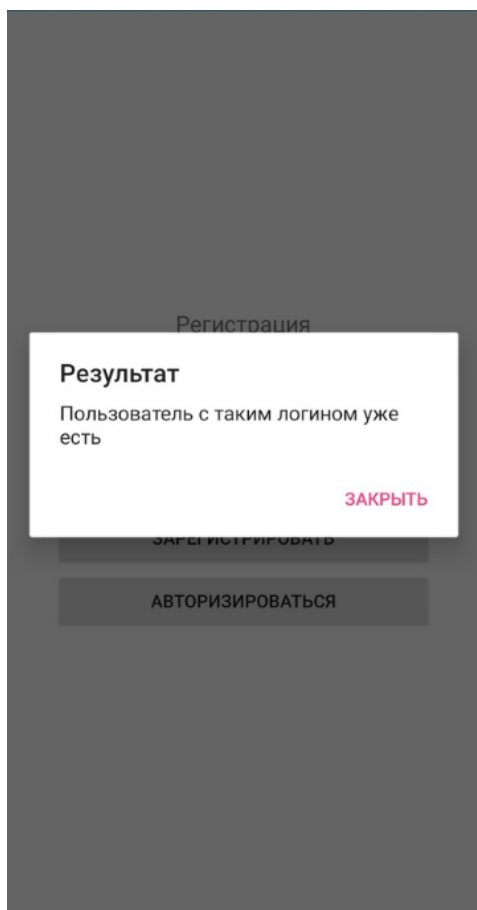


Рисунок Д.2 – Попытка регистрации под существующим логином

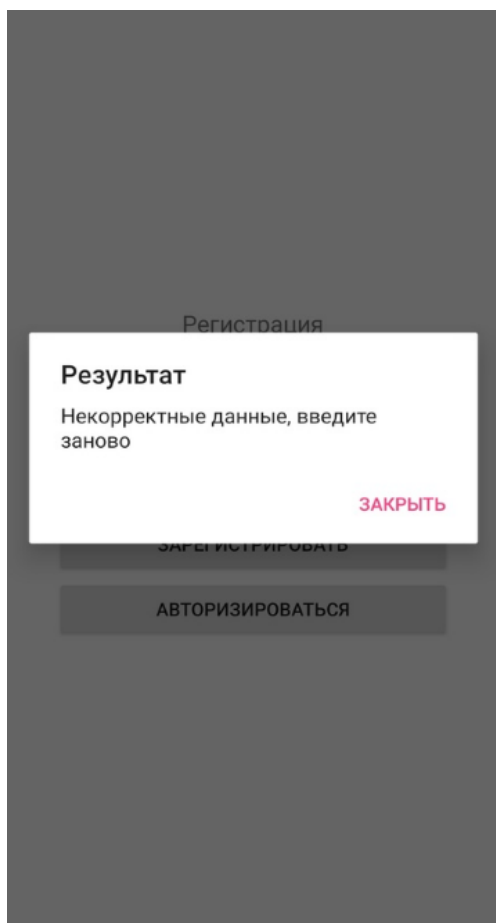


Рисунок Д.3 – Попытка регистрации с использованием пароля или логина
короче 4-х символов

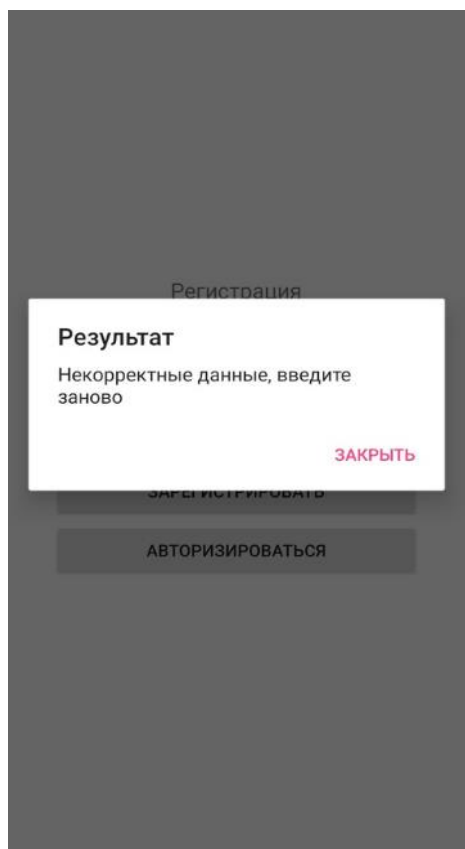
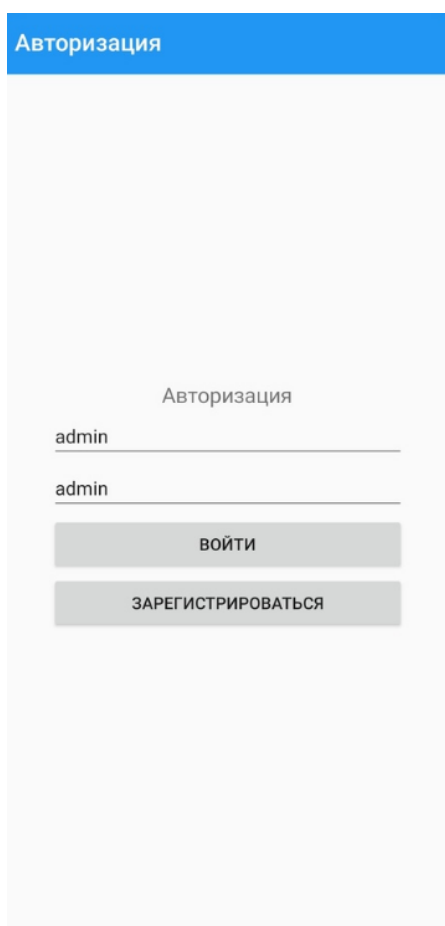


Рисунок Д.4 – Попытка регистрации при использовании некорректных СИМВОЛОВ



Авторизация

Авторизация

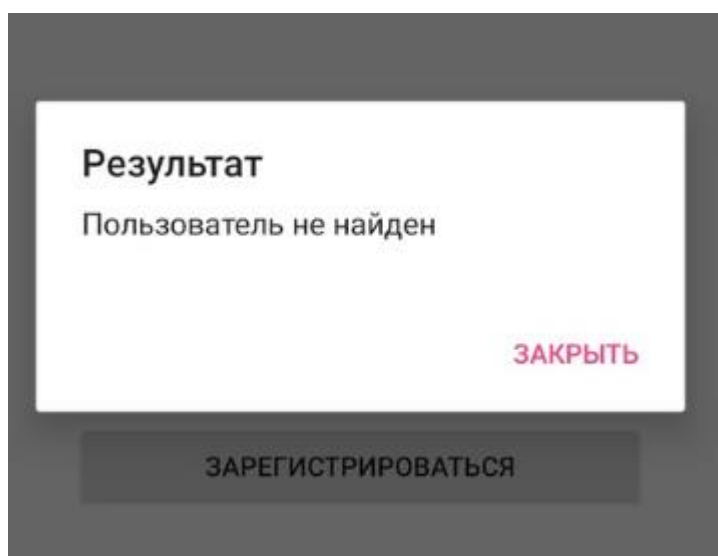
admin

admin

ВОЙТИ

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок Д.5 – Авторизации под существующим логином и паролем



Результат

Пользователь не найден

ЗАКРЫТЬ

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок Д.6 – Авторизация под несуществующим логином

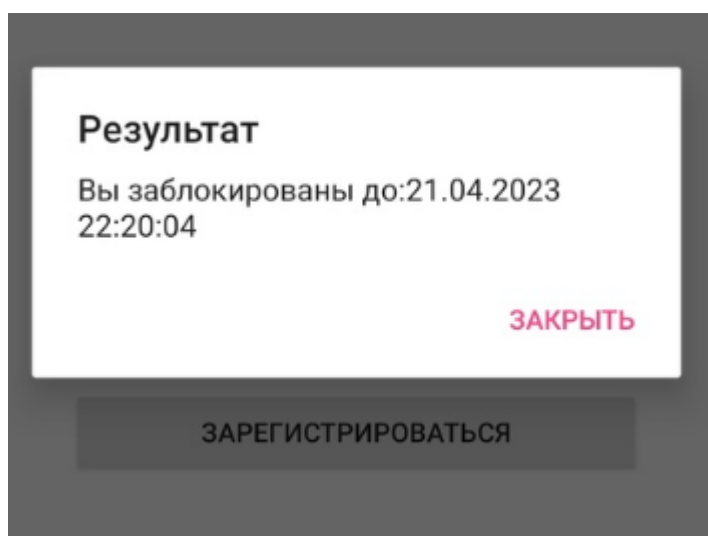


Рисунок Д.7 – Авторизация под заблокированным аккаунтом

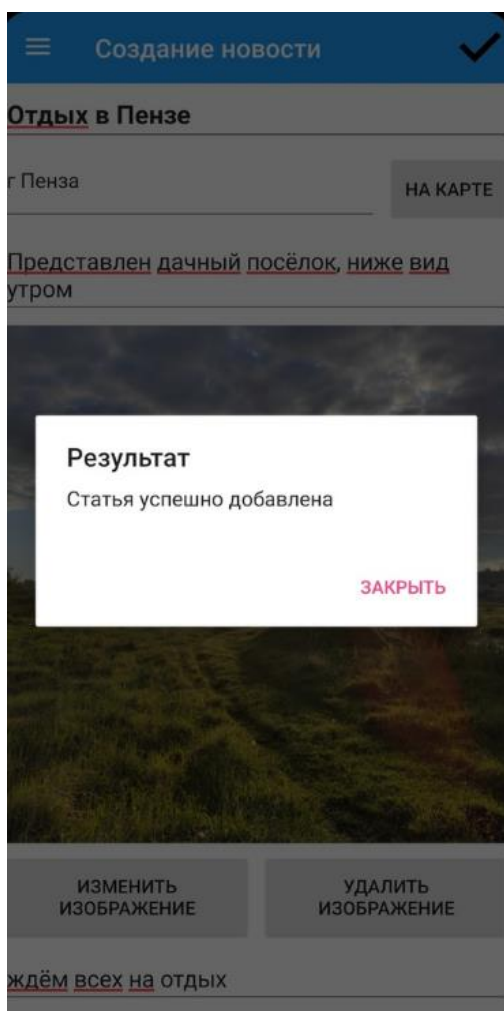


Рисунок Д.8 – Результат создание новости

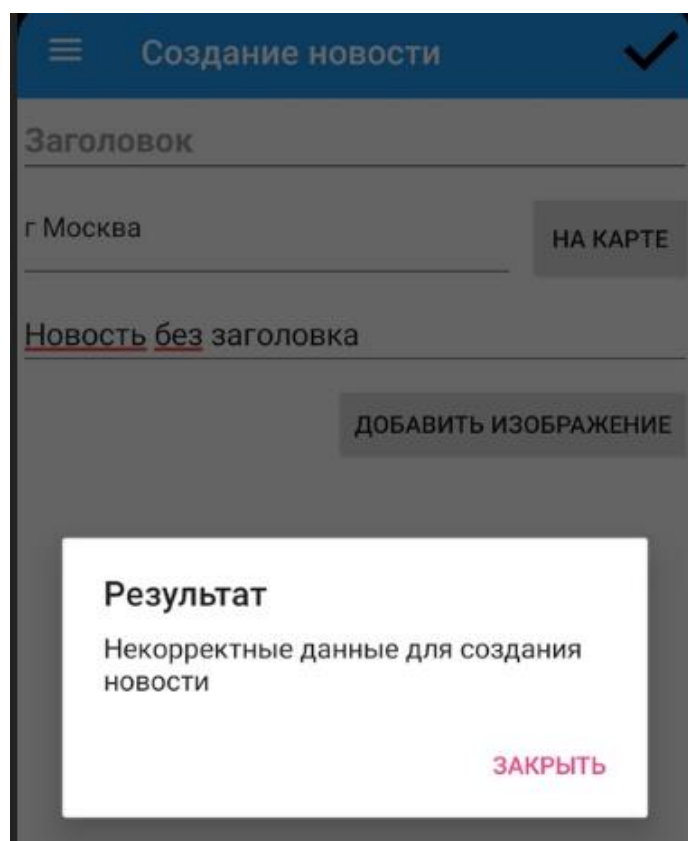


Рисунок Д.9 – Результат создание новости без ввода заголовка

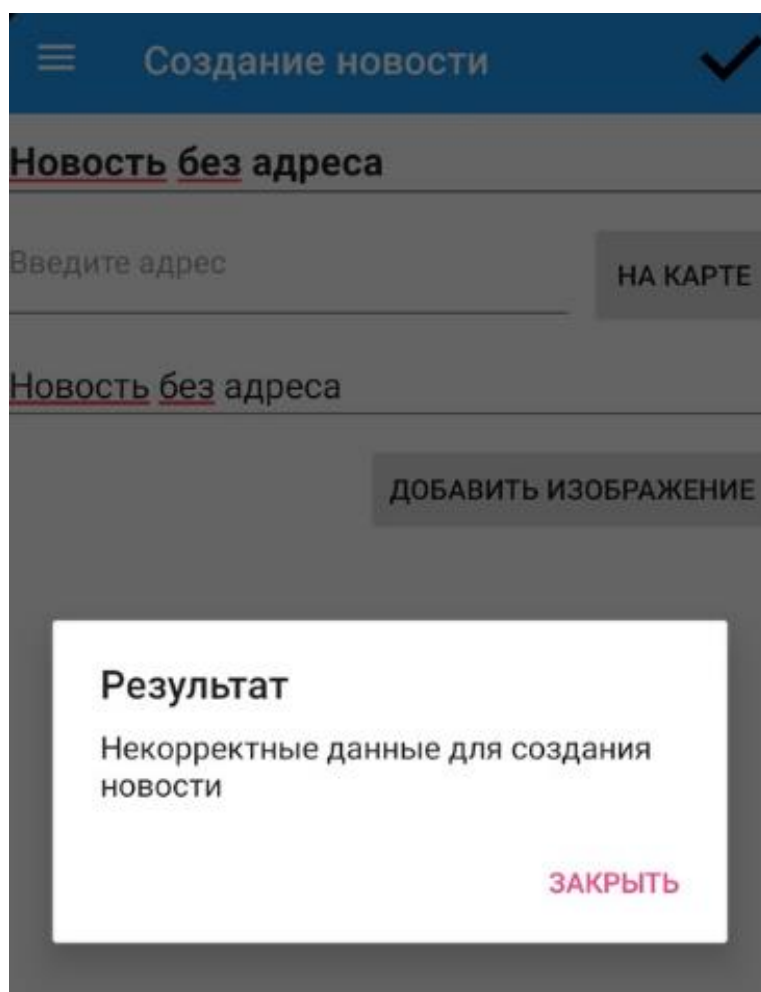
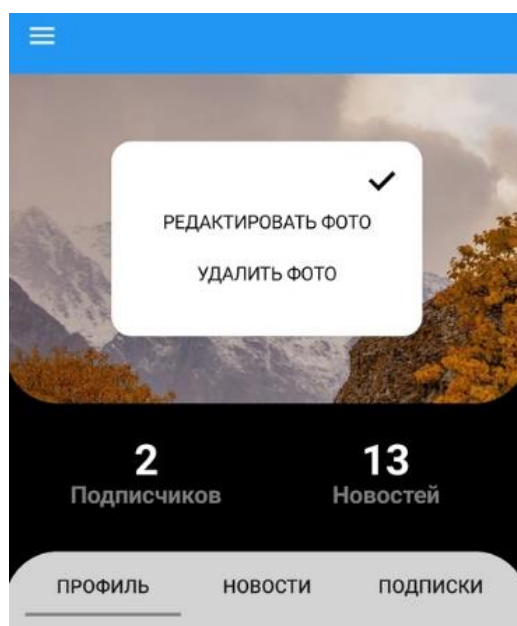


Рисунок Д.10 – Результат создания новости без ввода адреса



Данила

About

Описание отсутствует

Рисунок Д.11 – Изменение изображение в профили

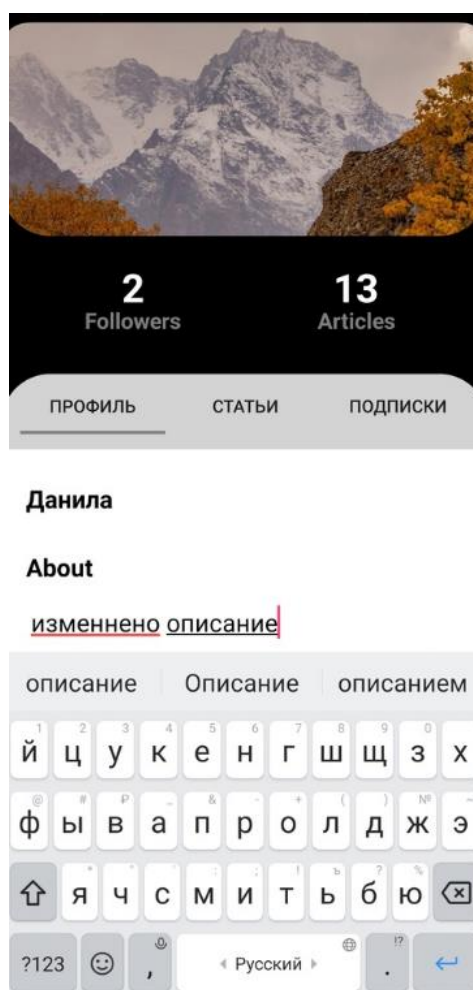


Рисунок Д.12 – Изменение описания профиля

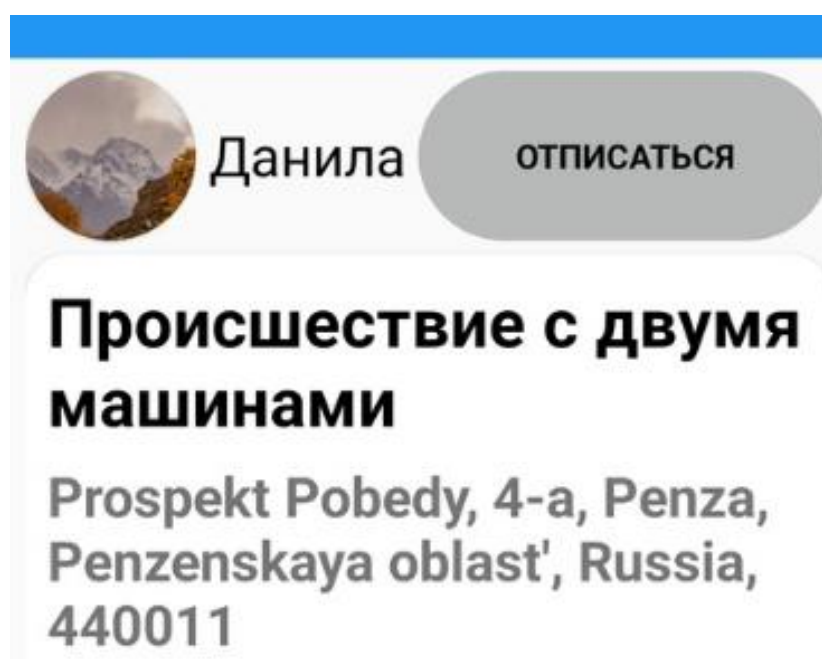


Рисунок Д.13 – Результат после попытки подписаться

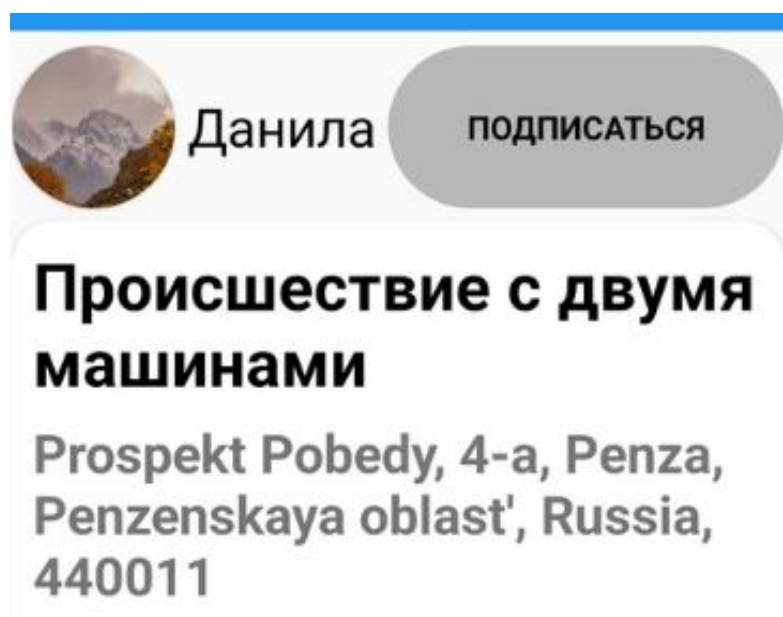


Рисунок Д.14 – Результат после попытки отписаться

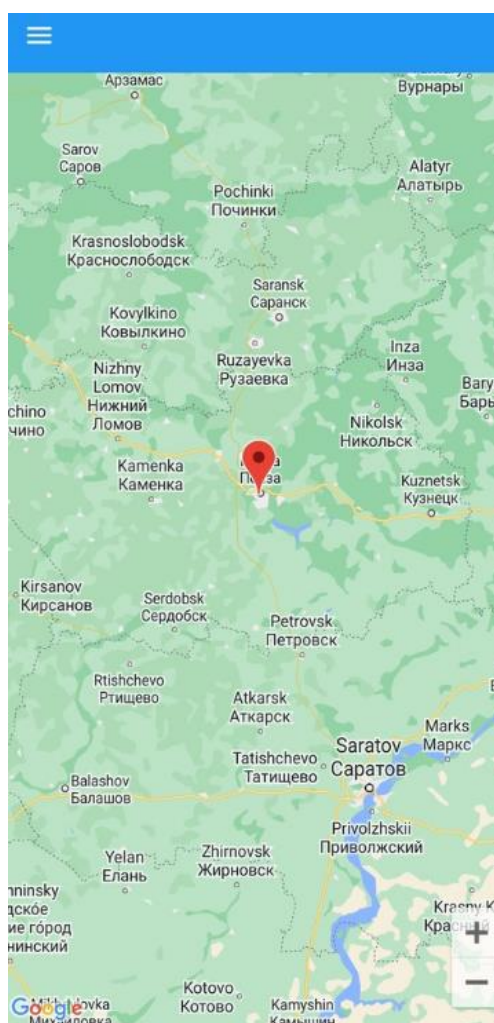


Рисунок Д.15 – Маркер для открытия новости на карте



Рисунок Д.16 – Новости для открытия в профили

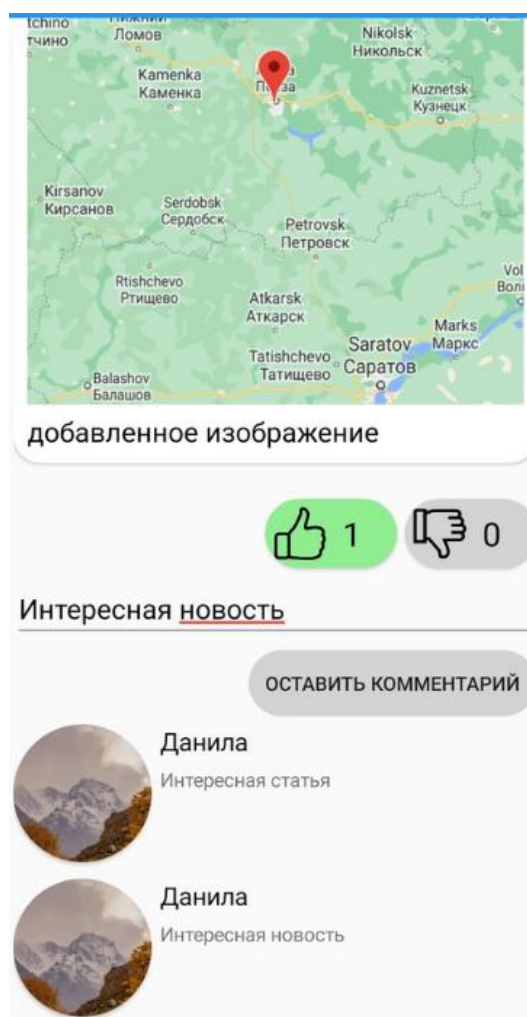


Рисунок Д.17 – Результат создания комментария к новости

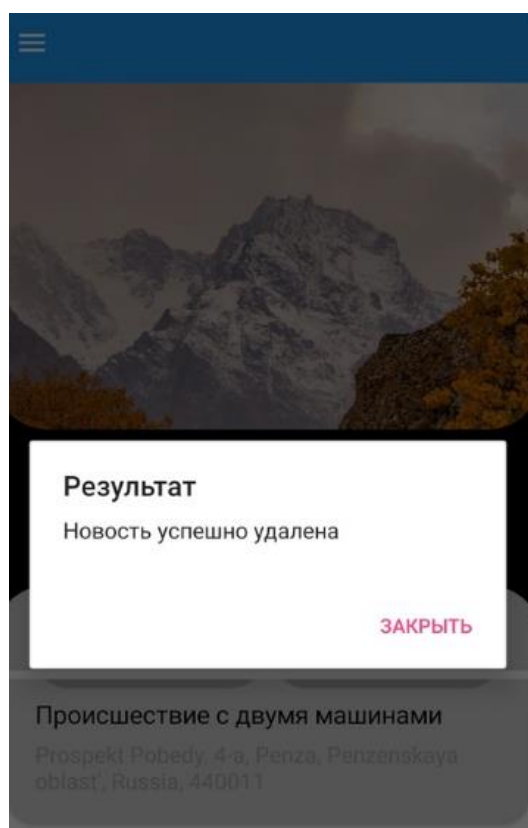


Рисунок Д.18 – Результат удаления новости

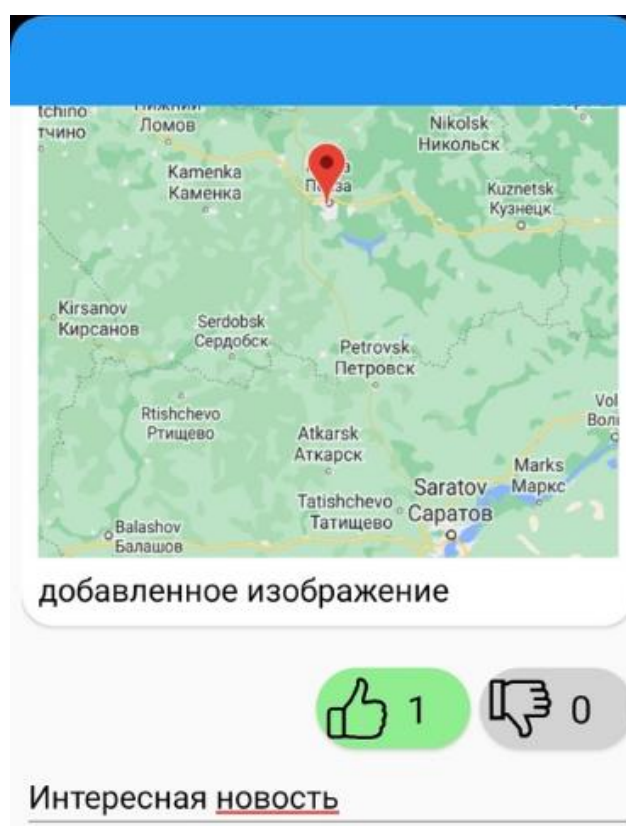


Рисунок Д.19 – Результат установки положительной оценки новости

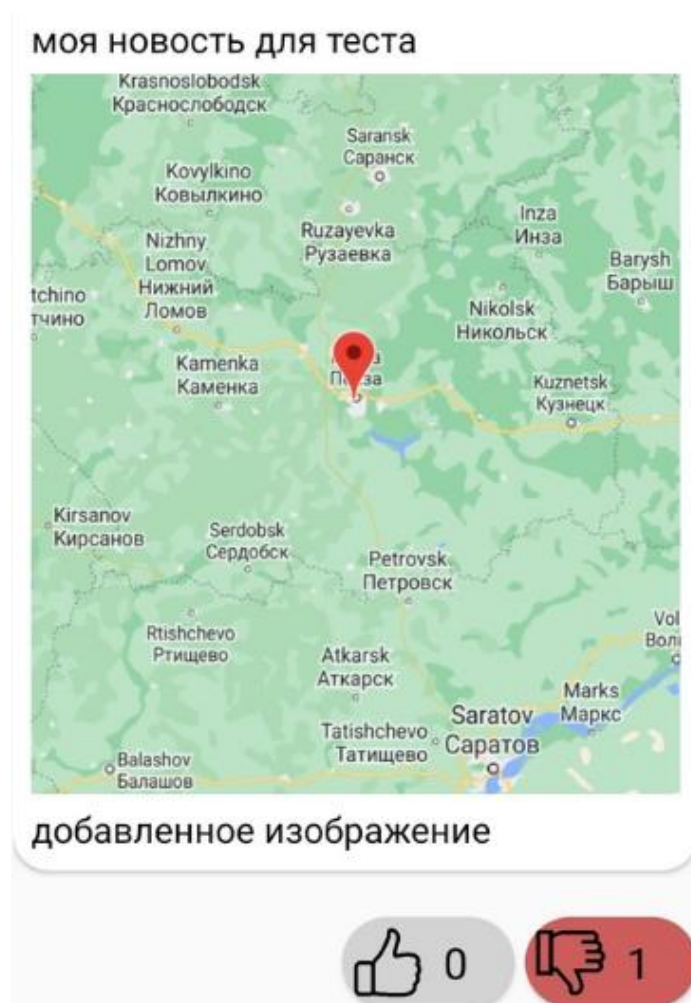


Рисунок Д.20 – Результат установки негативной оценки для новости

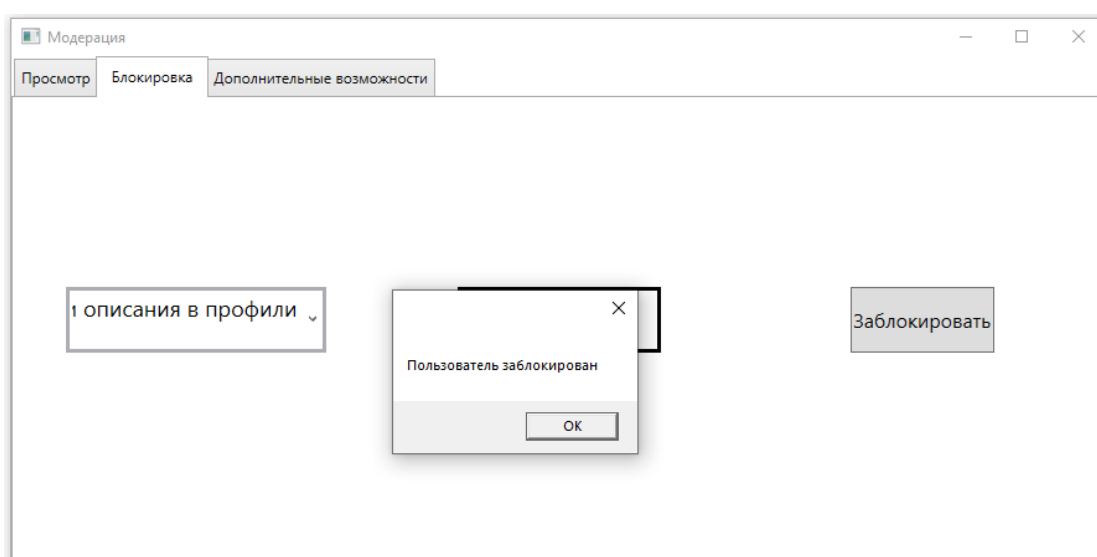


Рисунок Д.21 – Блокировка пользователя в приложении для модерации

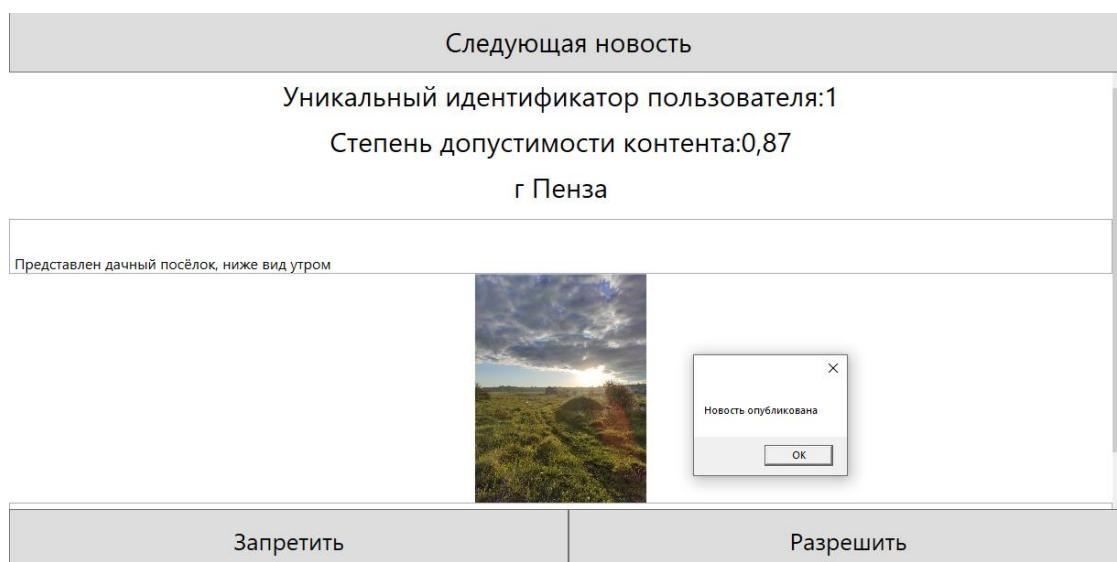


Рисунок Д.22 – Результат публикации новости

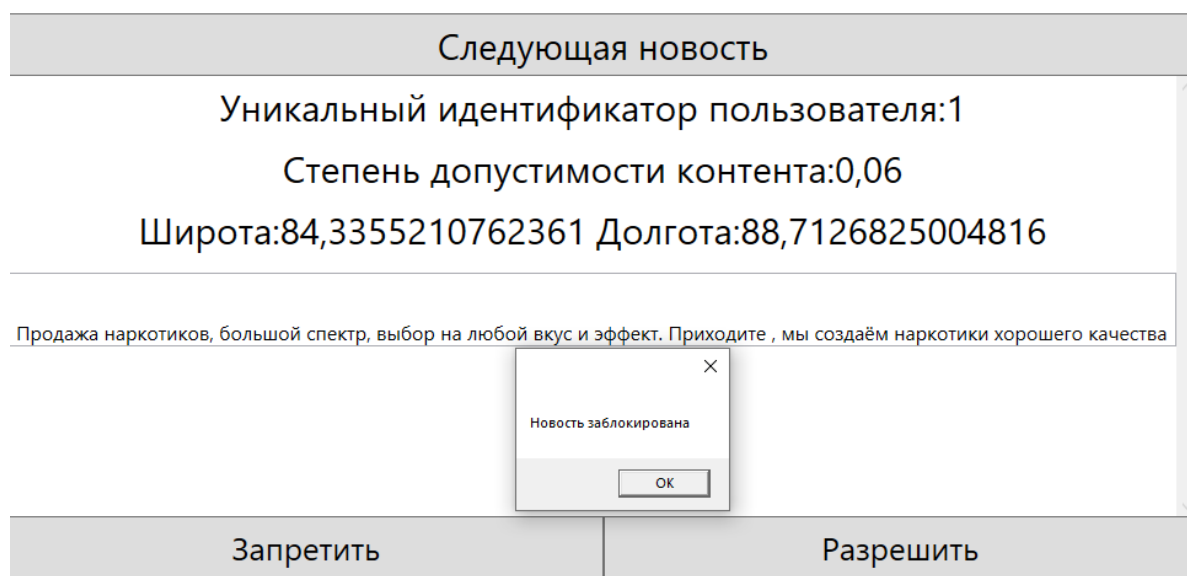


Рисунок Д.23 – Результат блокировки новости для публикации