

Министерство образования Республики Беларусь

Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина Средства и технологии анализа и разработки информационных систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
к курсовому проекту  
на тему

СОЦИАЛЬНАЯ СЕТЬ

Студент

Стенник Н.А.  
гр. 672303

Руководитель

Гордиевская Е.А.,  
ассистент  
кафедры  
экономической  
информатики

Минск 2019

T3

T3

## СОДЕРЖАНИЕ

Введение .....	5
1 Описание деятельности социальной сети .....	7
2 Описание процесса продажи рекламы .....	11
3 Спецификация вариантов использования системы .....	19
4 Информационная модель системы и её описание .....	21
5 Обоснование выбора компонентов и технологий для реализации курсового проекта .....	24
5.1 Описание ключевых технологий и обоснование их выбора .....	24
5.2 Описание диаграммы развёртывания .....	25
5.3 Описание диаграммы компонентов .....	25
5.4 Описание диаграммы последовательности .....	26
6 Модели представления данных и их описание .....	27
6.1 Диаграмма состояний .....	27
6.2 Блок схемы, реализующие бизнес-логику системы.....	27
7 Описание применения паттернов проектирования .....	28
8 Руководство по развёртыванию системы .....	32
9 Результаты тестирования разработанной системы.....	40
Заключение .....	43
Список использованных источников .....	44
Приложение А (обязательное) UML диаграммы.....	45
Приложение Б (обязательное) Блок-схемы алгоритмов.....	49
Приложение В (обязательное) Листинг SQL-скрипта .....	51
Приложение Г (обязательное) Листинг исходного кода.....	55

## ВВЕДЕНИЕ

Социальные сети – наиболее популярны сегодня. Рассматривая результаты многих опросов и голосований, только 2% из пользователей интернет никогда не заходили в социальные сети и толком не знают, что это такое. Зато постоянных посетителей социальных сетей, которые заходят туда каждый день и проводят там не менее часа, по данным опросов, составляют 67%.

Отсюда можно сделать вывод – что социальные сети, как магнит, притягивают посетителей, а те в свою очередь отдают сетям массу свободного времени. Большая часть пользователей использует их, как поиск давних знакомых, друзей детства, бывших одноклассников и однокурсников и так далее.

Другая часть посетителей социальных сетей собирается в тематические сообщества внутри сети. Организовывая общение по интересам. Причём, в многомиллионном обществе, которое собралось внутри какой-либо социальной сети, нельзя не найти единомышленников.

Здесь очень хорошо заводятся и возобновляются не только дружеские связи, но и деловые. Поделиться опытом, бизнес-идеями, а также воплотить всё это в реальной жизни и создать довольно успешный проект, который принесёт хорошие деньги. В социальных сетях на сегодняшний момент это вполне нормальное явление. Обмен какими-либо услугами в Интернете, например, один пишет интересный текст, для публикации у кого-то блоге, а другой оказывает помощь с организацией такого же блога, как у него – это вполне реально. И профессиональные консультации от юридических до медицинских, не всегда конечно бесплатные, но здесь также очень часты. Многие вещи в социальных сетях делаются в основном с позиции взаимовыручки.

По доступу социальные сети бывают открытые и закрытые. Наиболее наглядные примеры открытых социальных сетей: VKontakte, Одноклассники, Facebook. Примеры закрытых социальных сетей: Teazel – сеть, в которой состоят только серфенгисты. Squ. are – сеть наполнена массой домашнего видео очень богатых людей с видами их модных аксессуаров, быстрых машин и очень отдаленных мест. И принимают туда конечно же только по приглашениям.

Таким образом можно сделать вывод, что при разработке социальной сети в данном курсовом проекте будут учтены все вышеперечисленные возможности. Основной идеей разрабатываемой социальной сети является

публикацией новостей и сведений из жизни пользователей. Это может быть, как обычная реакция на новости в мире, так и своё собственное мнение об обычных и банальных вещах. Каждый имеет право высказать, то что он думает. И если вам интересные мысли этого человека, вы можете следить и дальше за ним, при этом высказывая своё собственное мнение по поводу его мыслей у себя на стене. Такая возможность будет реализовано с помощью технологии following, а также repost.

Целью данного курсового проекта является оптимизация следующих процессов: общения между людьми, обмена необходимой информацией, изучения информации в различных сферах, получения знаний, обучения и развития, преодоления сложностей общения из-за различных факторов, поиск новых знакомств и поддержания общения с друзьями, удовлетворения потребностей досуга пользователя за счёт сосредоточения необходимых ресурсов в одном месте или же предоставление ссылок на них. В с помощью создания социальной сети.

Для достижения поставленной цели необходимо решить следующие задачи:

- изучить предметную область деятельности социальной сети;
- выполнить функциональное моделирование основных процессов социальной сети;
- разработать модели представления системы на основе UML;
- рассмотреть концептуальные основы развития социальной инженерии;
- создать логическую и физическую модель представления данных;
- разработать базу данных;
- разработать интерфейс приложения;
- обеспечить возможность добавления, просмотра, редактирования, и вывода данных;
- обеспечить разделение доступа к управлению программой с соответствующими программными возможностями;
- протестировать программное приложение.

Объектом исследования данного проекта является информационная система, содержащая сведения о пользователях и их сообщениях.

## 1 ОПИСАНИЕ ДЕЯТЕЛЬНОСТИ СОЦИАЛЬНОЙ СЕТИ

Предметной областью данной курсовой работы выступает обширное понятие «Социальная сеть». Рассмотрим понятие социальная сеть и дадим ей определение.

В качестве социальной сети может выступать некая платформа, онлайн-сервис или же веб-сайт, предназначенные для построения, организации социальных отношений в сети Интернет.

В недалёком прошлом, лет 5-7 назад большой популярностью пользовались социальные сети общего типа, предназначенные для личного и делового общения. Они построены на определённом типе контента. Яркими примерами таких сетей могут выступать Facebook, Вконтакте, Одноклассники. Данный формат социальных медиа одним из первых предложил пользователям создать бесплатный персональный мини-сайт, который позже стал известен как профиль. Цель этих сетей предложить максимум разносторонних возможностей в одном месте. [1]

Сейчас же очень активно развиваются тематические проекты, которые используют тот же механизм, что и социальные сети общего типа, но в каком-то ограниченном направлении. На текущий момент можно найти социальные сети абсолютно для любого типа контента, например, для IT-специалистов, для вашего домашнего питомца, для спортсменов, шопоголиков и т.д. Тематические социальные сети распространены глобально и не ориентируются на какую-то конкретную страну.

В нашей социальной сети будет преобладать новостной характер. В контексте данной социальной сети новости будут как глобальные в мире, так и обычные события из жизни пользователя, за которыми могут следить все те, кому интересна данная личность.

Стоит отметить, что пользователи социальной сети не хотят больше получать контент, который кто-то создаёт, редактирует и отбирает без учёта их мнения. Они хотят в какой-то степени контролировать этот процесс. Им хочется знать, что их мнение всегда учитывается. Эти сообщества основываются на тематическом контенте и коммуникации пользователей, особое место отводится вспомогательным сервисам в направлениях геолокации, электронной коммерции и т.д.

Наша социальная сеть подходит для этих целей как никак другая. Именно в нашей социальной сети люди смогут реагировать на посты других людей, на посты информационных порталов и новостных компаний. Высказывать своё мнение у себя на странице. Поддерживать мнения других

личностей, которые им импонируют. Основная тематика – это обсуждения того, на что вам не всё равно. Обсудить то, что для вас действительно важно.

Не менее важным аспектом социальной сети выступает технологическое развитие. Социальные сети становятся всё более популярными. Популярность влечёт за собой быстрый рост. Если же вначале социальные сети развивались в основном количественным путём, то сейчас в приоритете стадия качественного развития. Социальные сети придумывают новейшие инструменты взаимодействия с пользователем, и чем интересней инструмент, тем больше шанс привлечь новую целевую аудиторию.

Перейдём к качественным инструментам. Коммуникация – это фундамент любой социальной сети, поэтому коммуникационный инструмент стоит развивать в первую очередь. Что касается работы с контентом, то многие социальные сети работают по принципу «User-Generated Content», то есть социальная сеть предоставляет пользователю необходимые инструменты, а люди используя их генерируют свой собственный новый контент. Данная модель работает довольно хорошо, за исключением генерации профессионального качественного контента, ведь пользователи во всём мире хотят получать именно качественный разнообразный контент. В прошлом социальные сети пытались структурировать контент пользователей, чуть позже стали создавать инструменты фильтрации качественного контента, а в ближайшем будущем стоит ожидать создание функционала нового уровня по созданию-отбору контента. Модель сайтов, в которых присутствует и качественный тематический контент, и социальный функционал, является наиболее перспективной.

В разрабатываемой социальной контент будет генерироваться по принципу, описанному выше. Так же в сети будет присутствовать как качественный профессиональный контент, предоставляемый информационными порталами и блогами, так и любительский контент отражаемый в лице обычного юзера, реагирующего на некоторые события. Таким образом модель социальной модели является перспективной так как объединяет в себе и качественный контент и социальный функционал.

Не так давно социальные сети стараются делать интеграцию с интернет-магазинами и создавать те самые необходимые инструменты для торговли между пользователями. Некоторые же сети полностью ориентированы на торговлю, больше выступая уже как некая торговая площадка, объединяющая людей с общими интересами. В любом случае это довольно интересный сегмент для социальных сетей: коммерческий рынок имеет обороты в миллиарды долларов, а у социальных сетей есть сотни миллионов



потенциальных покупателей. Следовательно, осталось только применить наиболее эффективный инструмент продажи товаров и услуг. Как и с контентом в социальной сети, существует два пути развития: интеграция с существующими интернет-магазинами и создание собственных платформ.

По моему мнению интеграция с существующими-интернет магазинами не приведёт ни к чему хорошему, так как это будет совсем не та тематика, не тот контент, к которому привык пользователь и это только оттолкнёт пользователя не только от интернета магазина, но и от социальной сети в целом.

Нужно постепенно идти к созданию собственных платформ для электронной коммерции, что уже реализовал facebook выдвинув свою платформу Facebook Marketplace. Социальным сетям выгодно создавать именно свои платформы, специальный функционал, который позволит продавать и покупать, а сеть при этом будет выступать посредником, зарабатывая на рекламе, дополнительном функционале, комиссии и других услугах. Данный инструмент будет полезен как для обычных юзеров, так и для корпораций, а социальные сети смогут зарабатывать на этом, что делает эту отрасль развития одной из основных и даёт неоспоримую уверенность в её серьёзном развитии.

Инструмент для работы ещё одна значимая тенденция. Серьёзные шаги делаются в сетях для делового общения, например, LinkedIn. Люди все больше стараются перевести свою работу в интернет и в частности в социальные сети тематика которых удовлетворяет их профессиональную деятельность. Уже существуют сервисы для хранения контактов, поиска клиентов, управления персоналом, документооборота, совместной работы и многое другое. Сети уже создали приличный функционал для поиска работы (jobs.tut.by), торговли (kufar.by или deal.by), рекомендации людей и т.д.

Каждая социальная сеть предоставляет удобные возможности для персонализации своей работы под конкретного человека. Сеть может следить за юзером отслеживая его поведение, местоположение, интересы. На основе этой информации воспроизводить персонализированный контент любого характера. Этот принцип очень востребован из-за того, что в интернете огромное количество информации, большинство из которой абсолютно бесполезно, а благодаря этому инструменту, пользователь получает необходимую актуальную информацию в определённый момент времени и в определённом месте. Так же благодаря этому инструменты можно давать персонализированную рекламу, выделяя нужный контент пользователю. Как

пример можно привести сервис Discover Facebook Pages, позволяющий узнать важных для человека людей и искать интересные страницы для него.

Рекламные технологии до сих пор остаются серьёзным источником прибыли в социальных сетях. Сейчас используется поведенческая технология, которые показывают рекламу пользователю не только на основе информации из его профиля, но и на основе его поведения (посещение определённых сайтов и тд). Эта технология позволяет показывать рекламу максимально таргетировано и иметь высокий уровень заинтересованности в рекламированном продукте. Крупные бренды заинтересованы в возможности разместить свою рекламу и скупить рекламные места, так как видят в социальных сетях привлекательные рекламные площадки из-за огромной базы пользователей, которая постоянно увеличивается. Таргетирование рекламы в обычных случаях происходит по стране, полу, возрасту, семейному положению, должности, интересам и т.д. Можно так же указывать время показа, количество показов, показы после определённых действий и т.д

Геолокация в социальных сетях может использоваться во многих направлениях. Это относительно уже не новый тренд, который начал развиваться из-за современных мобильных технологий. На основе геолокации можно предлагать таргетированный контент для определенной местности, то есть всё что находится рядом возле пользователя (интересные заведения, развлечения, информация о месте, новые знакомства и т.д.). Определение местоположения человека происходит либо автоматически с помощью мобильных технологий, либо посредством информации, которую дает сам пользователь.

Подводя итог, можно сказать что на сегодняшний день социальные сети могут быть абсолютно разной тематики и иметь самый разнообразный контент. Социальная сеть обладает огромными перспективными возможностями благодаря специальным инструментам внедряемых в неё. Социальная сеть может так же и сама использоваться для каких-либо целей, например, для продаж. Инструменты, использующиеся в социальной сети, становятся всё более умными, хитрыми и инновационными, привлекая внимание пользователя. Можно предположить, что социальная сеть в будущем будет обладать следующими возможностями: искусственный интеллект, 3D сети и виртуальная реальность, голосовое управление.

## 2 ОПИСАНИЕ ПРОЦЕССА ПРОДАЖИ РЕКЛАМЫ

Процесс продажи рекламы довольно сложный и многогранный. В зависимости от того, в какой среде и в каких условиях будет проводиться реклама продукта, сложность, затраты на реализацию, а также эффективность будут отличаться.

В данной главе будет отображена ощутимая разница продажи рекламы в социальной сети и продажа внешней рекламы оффлайн. Каждый из этих двух видов процесса продажи рекламы обладает своими отличительными особенностями и характером проведения, на основе которых будут сделаны выводы.

Для того чтобы оценить эффективность использования рекламы в разных условиях были созданы функциональные модели «как есть» и «как должно быть» с использованием стандарта IDEF0.

В начале рассмотрим обычный способ продажи внешней рекламы оффлайн – это наша функциональная модель «как есть». Для простого понимания представим, что компания по продаже рекламы занимается внешними рекламными конструкциями.

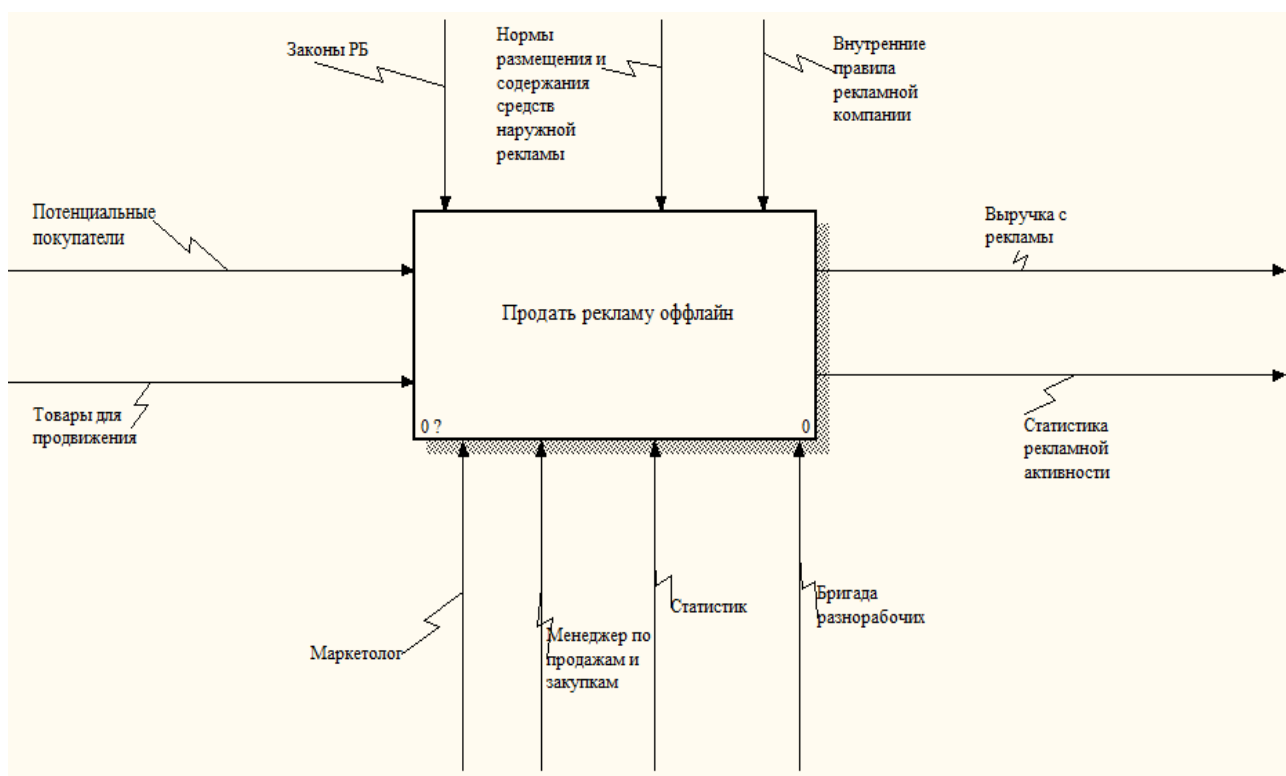


Рисунок 2.1 – Контекстная диаграмма верхнего уровня процесса продажи рекламы оффлайн

Для продажи рекламы оффлайн требуется персонал, который будет выполнять основные функции работы компании. Персонал представлен следующими сотрудниками: маркетолог, менеджер по продажам, статистик, бригада разнорабочих. Документы, которыми руководствуется компания: законы Республики Беларусь, нормы размещения и содержания внешней рекламы, внутренние правила компании. Основная цель продажи рекламы – это получение выручки за проделанную работу и сбор статистики активности рекламы. Исходными данными выступает товар и его потенциальные покупатели.

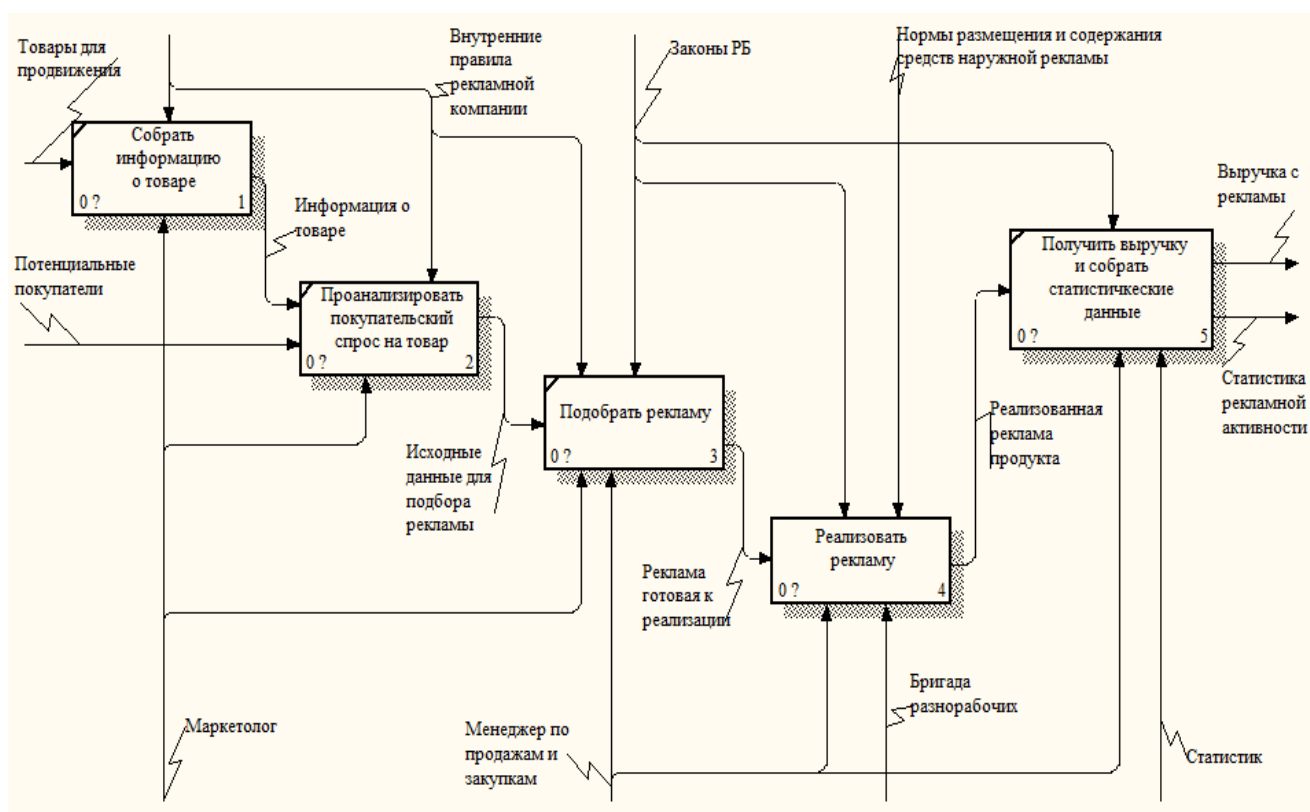


Рисунок 2.2 – Декомпозиция контекстной диаграммы процесса продажи рекламы оффлайн

Чтобы осуществить продажу рекламы оффлайн для начала нужно собрать информацию о товаре, который компания собирается рекламировать, узнать его сильные и слабые стороны. Далее нужно проанализировать популярность и потребность в этом товаре. После качественной обработки данных можно приступить к подбору рекламы, чтобы правильно прорекламировать товар, подчеркнув его сильные стороны. Конечный вариант подобранной рекламы нужно реализовать в кратчайшие сроки, чтобы сохранить его актуальность. После организованной рекламы продукта,

компания получает выручку, и следит за активностью своей рекламы. Сбор статистики является важным этапом, для оценки качества проделанной работы. Рассмотрим способ реализации продажи рекламы на рисунке 2.3.

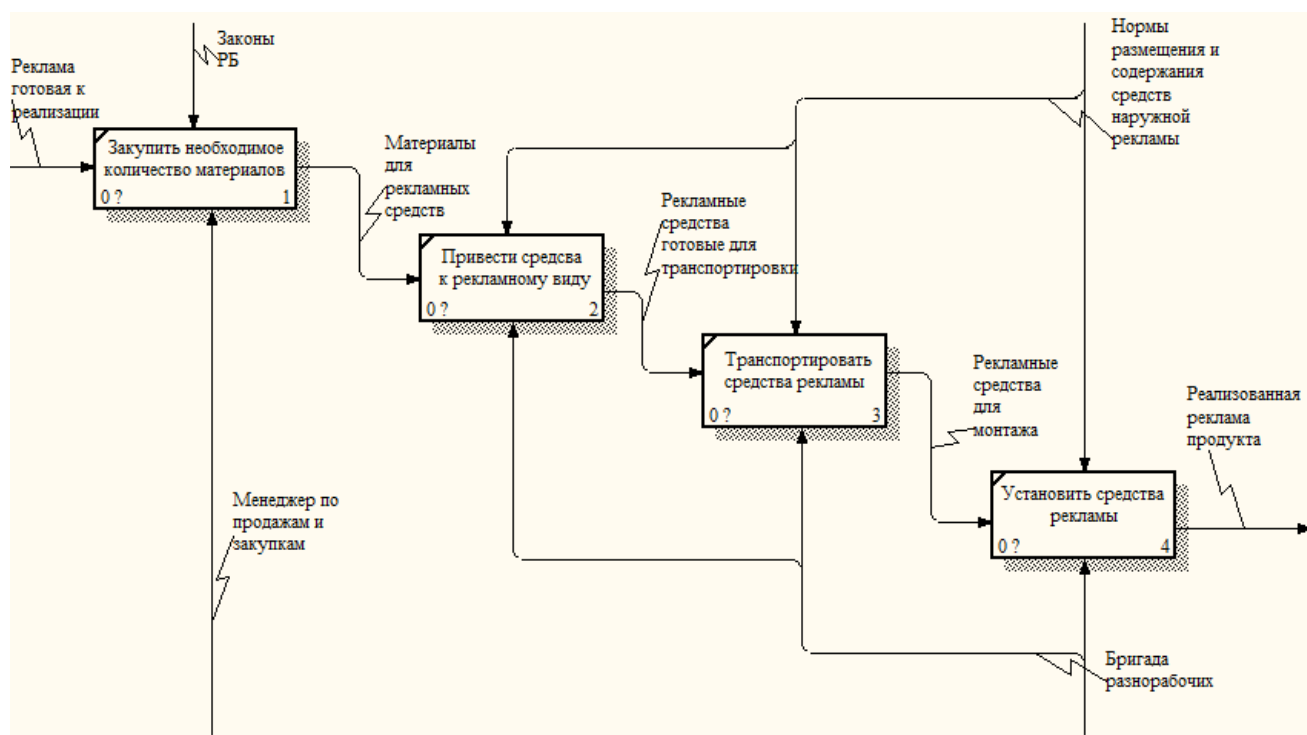


Рисунок 2.3 – Декомпозиция блока «Реализовать рекламу»

Имея информацию о нужной рекламе в первую очередь необходимо закупить нужное количество материалов для изготовления рекламных средств. Это могут быть различные каркасы, покрасочные материалы и т.д. После закупки материалов, нужно привести средства к рекламному виду. Под этим подразумевается изготовление определённых деталей, доработка макета, покраска или нанесения принта. Изготовленные рекламные средства нужно транспортировать в точку размещения. Прибыв на место размещения, рекламное средство нужно установить, проведя монтажные работы.

Декомпозиция данного процесса даёт чёткое понимание того, что процесс реализации рекламы довольно затратный и трудоёмкий. Закупка материалов, их подготовка, транспортировка и монтаж, индивидуально под каждое рекламное средство очень затратные процессы, именно поэтому такая реклама для заказчика обходится не дёшево. Рекламная компания тоже не находится в большом выигрыше, так как выделяет человеческие ресурсы для сбора информации о потенциальных покупателях, анализа статистики и многое другое, которые можно заменить на более выгодный ресурс. Этим

выгодным ресурсом выступает социальная сеть, которая является более комфортным, менее трудозатраты, и что самое главное более эффективным средством.

Рассмотрим процесс продажа рекламы используя социальную сеть. Процесс продажи рекламы в сети является одним из основополагающих, ведь именно с помощью этого социальные сети зарабатывают весомую часть прибыли. Чтобы понимать, каким образом будет происходить продажа рекламы в социальной сети была создана функциональная модель «как должно быть».

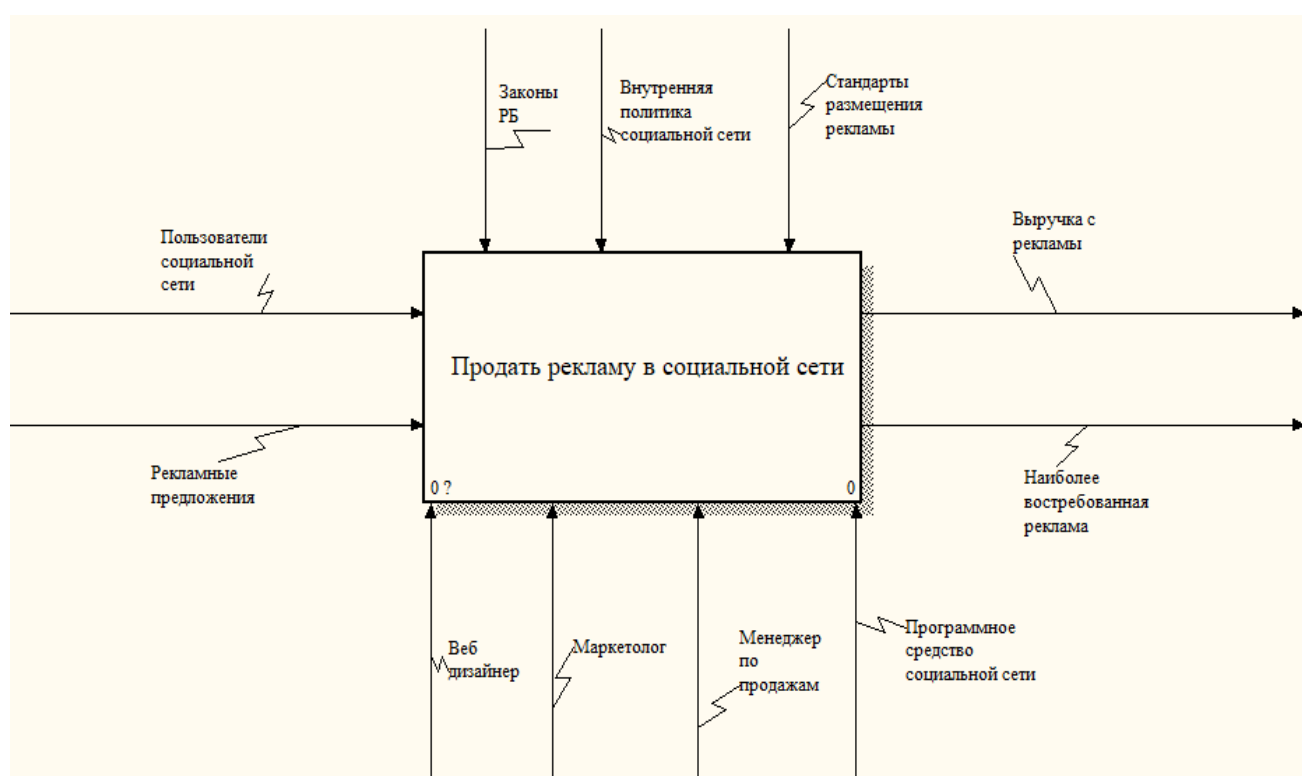


Рисунок 2.4 – Контекстная диаграмма верхнего уровня процесса продажи рекламы в социальной сети

Контекстная диаграмма верхнего уровня содержит некоторые изменения по сравнению с моделью «как есть». Потенциальными покупателями в социальной сети являются её пользователи. Огромную часть работы выполняет прикладная программа социальной сети заменяя собой людские ресурсы. Больше нет надобности проводить социальные опросы для выявления предпочтений пользователя, следить за покупками пользователя в реальной жизни, смотреть какой товар наиболее популярен и т.д. Пользователь сам подскажет системе необходимую информацию исходя из его интересов, посещаемых ресурсов, местоположения.

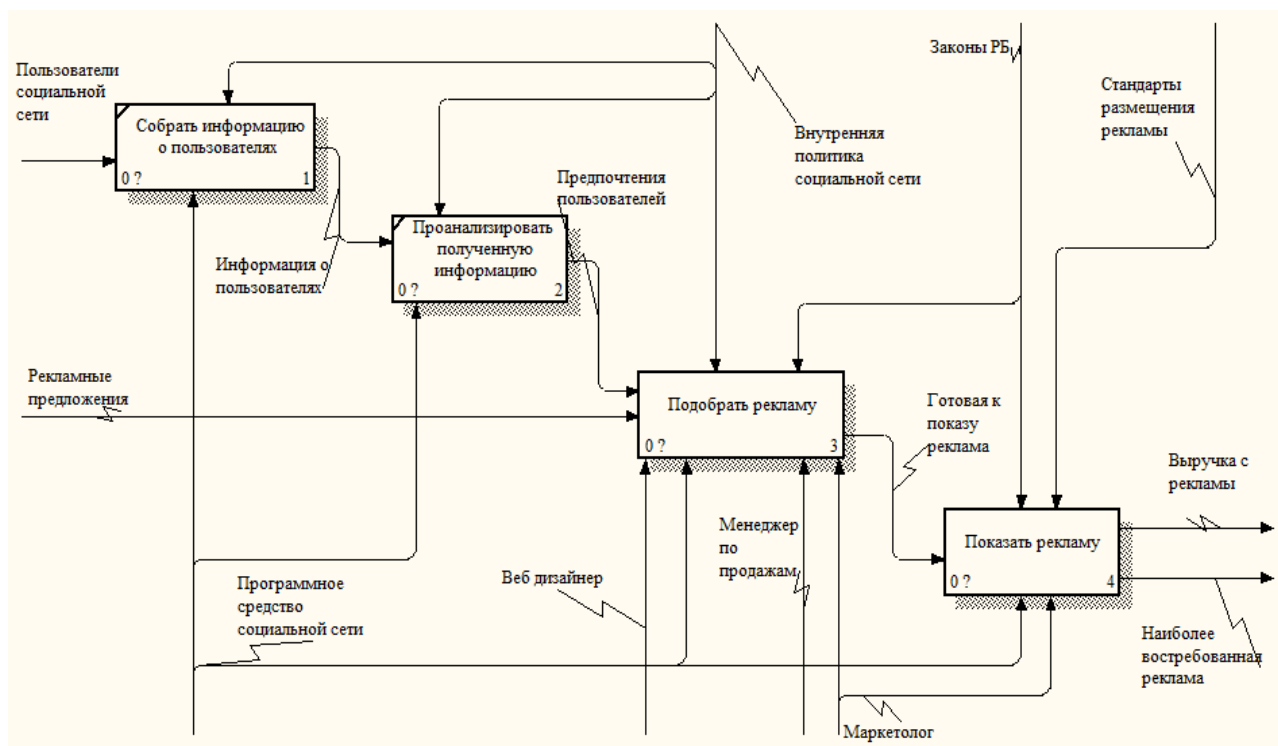


Рисунок 2.5 – Декомпозиция контекстной диаграммы процесса продажи рекламы в социальной сети

Социальная сеть проанализировав полученную информацию подберёт качественный контент, который будет интерес пользователю. Для подбора рекламы необходимы также человеческие ресурсы, но при этом основной рутинный и монотонный труд выполняет система.

Рассмотрим рисунок 2.6. Для того, чтобы правильно осуществлять продажу рекламу нужно проанализировать предложения от рекламодателей и подготовить итоговый список. После получения итогового списка рекламных предложений программное средство подбирает пользователей, которым будет интересна соответствующая реклама. Когда пользователи подобраны, нужно распределить рекламу между ними, из-за того, что несколько реклам одной тематики могут соответствовать определённым пользователям одновременно. Правильно распределив рекламу, мы имеем план показа рекламы. Следующим этапом будет моделирование рекламных элементов, которым занимается веб дизайнер.

Анализ предложений от рекламодателя – интересный этап. Именно здесь исходя из определённых критериев стороны заказчика и со стороны компании выбирается реклама к показу. На рисунке 2.7 – изображена декомпозиция данного блока.

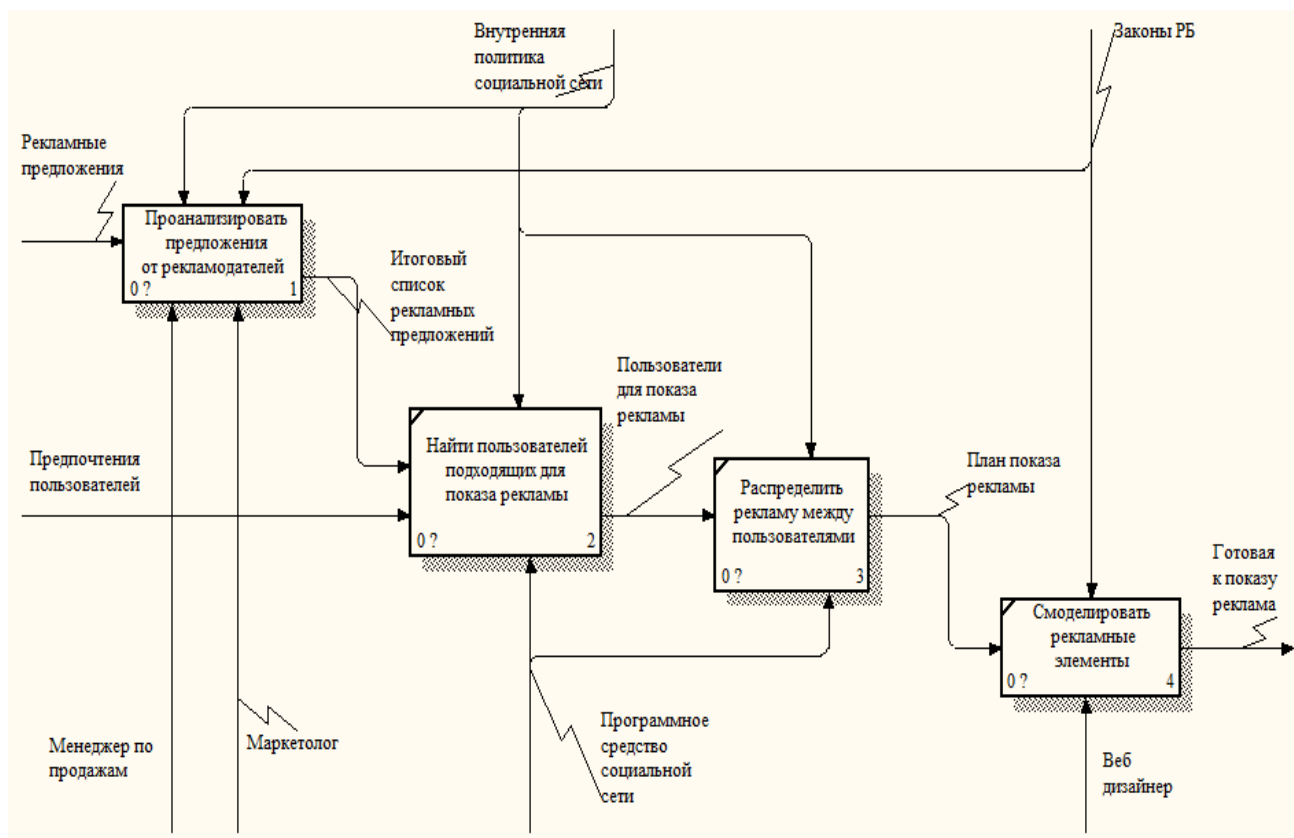


Рисунок 2.6 – Декомпозиция блока «Подобрать рекламу»

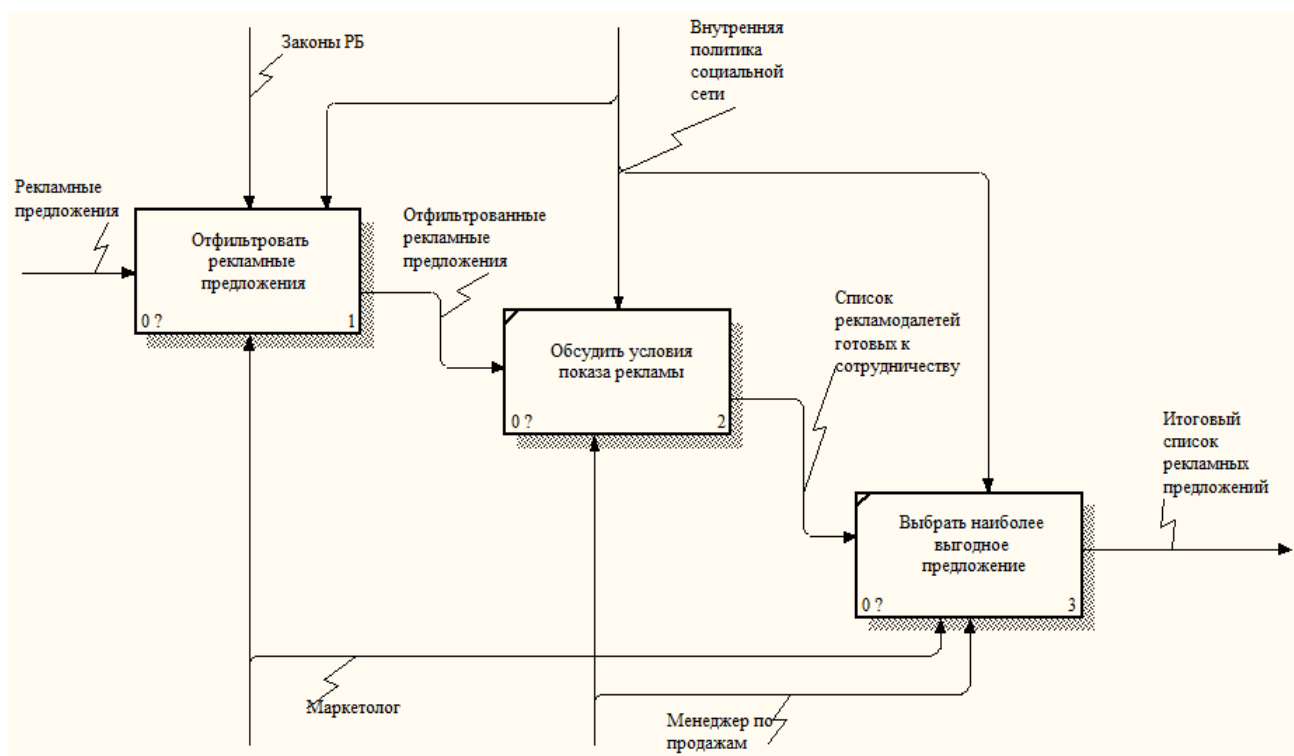


Рисунок 2.7 – Декомпозиция блока «Проанализировать предложения от рекламодателей»



Рекламные предложения проходят тщательную фильтрацию в соответствии с законодательством Республики Беларусь и внутренней политикой социальной сети (рисунок 2.8). В начале отсеивается реклама, нарушающая эти нормативные документы, далее нужно определить вид рекламы, потому что социальная сеть может не размещать рекламу определённого вида в связи со своими внутренними правилами. Хорошее качество контента рекламы необходимый фактор при фильтрации. Ведь именно это прямо влияет на репутацию социальной сети. Социальной сеть обладает разными параметрами целевой аудитории, начиная от возраста и заканчивая национальностью. Если социальная сеть не обладает нужными параметрами целевой аудитории рекламы, то и размещать её нет смысла.

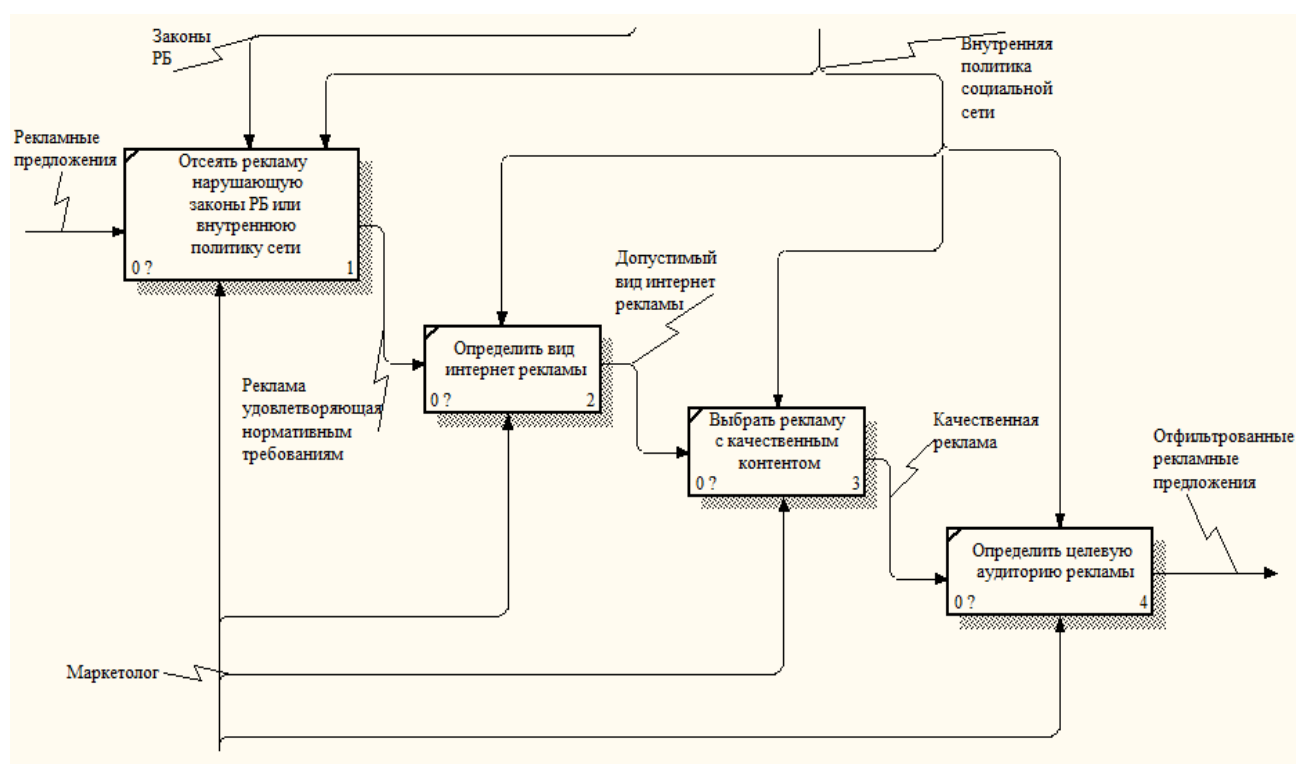


Рисунок 2.8 – Декомпозиция блока «Отфильтровать рекламные предложения»

Если рекламное предложение прошло все этапы проверки далее ведутся переговоры и обсуждаются условия показа рекламы с рекламодателем. Проведя множество таких переговоров, выбираются наиболее выгодные для компании социальной сети.

Рассмотрев процесс подбора рекламы с использованием социальной сети. Можем сделать вывод, что использование социальной сети для продажи рекламы уже является большим преимуществом в связи с облегчением работы

для человека при поиске подходящих пользователей для рекламы и её распределения. Однако всё преимущество социальной сети содержится в размещении рекламы.

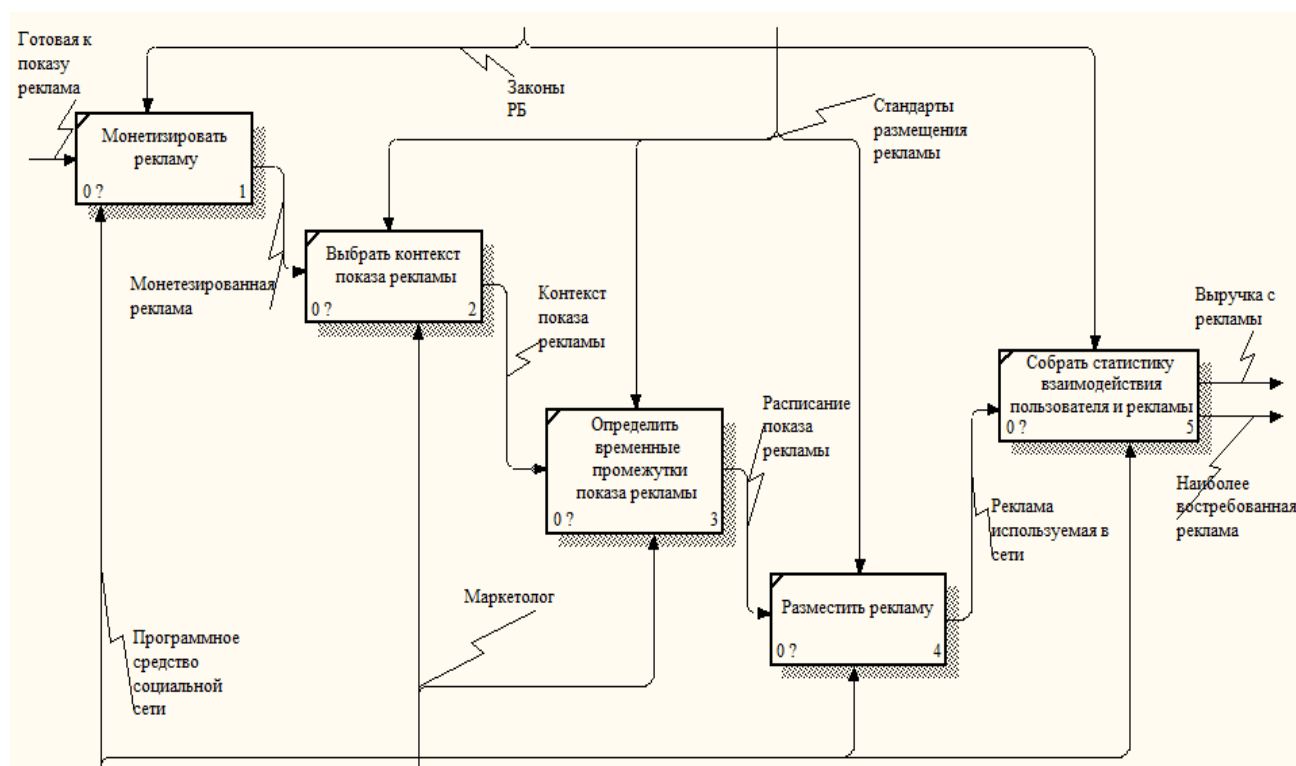


Рисунок 2.9 - Декомпозиция блока «Показать Рекламу»

При показе рекламы в социальной сети отпадают нужды в закупке материалов для изготовления рекламных средств, так как рекламные элементы уже изготовлены дизайнером в специальных приложениях без использования каких-либо физических ресурсов и приведены к рекламному виду. Транспортировать и монтировать рекламу так же больше не нужно. Всё что нужно для размещения рекламы социальной сети – это монетизировать её выбрав способ монетизации в виде переходов по ссылке, просмотров и т.д. На основе этого можно проанализировать эффективность рекламы, а также получать реальную выручку, зависящую от активности пользователей. Выбирать контекст рекламы нужно аккуратно, чтобы не доставлять дискомфорт пользователю. Показывать рекламу нужно в определённые промежутки времени в зависимости от целей рекламы.

Благодаря размещению рекламы в социальной сети компания сохраняет большое количество денежных средств в связи с отсутствием больших затрат на реализацию рекламы. Это так же дешевле и для заказчика рекламы, что повышает спрос на рекламу в социальной сети.

### 3 СПЕЦИФИКАЦИЯ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ СИСТЕМЫ

Чтобы описать основную функциональность системы, продемонстрировать взаимоотношения и зависимости между группами вариантов использования была создана диаграмма вариантов использования.

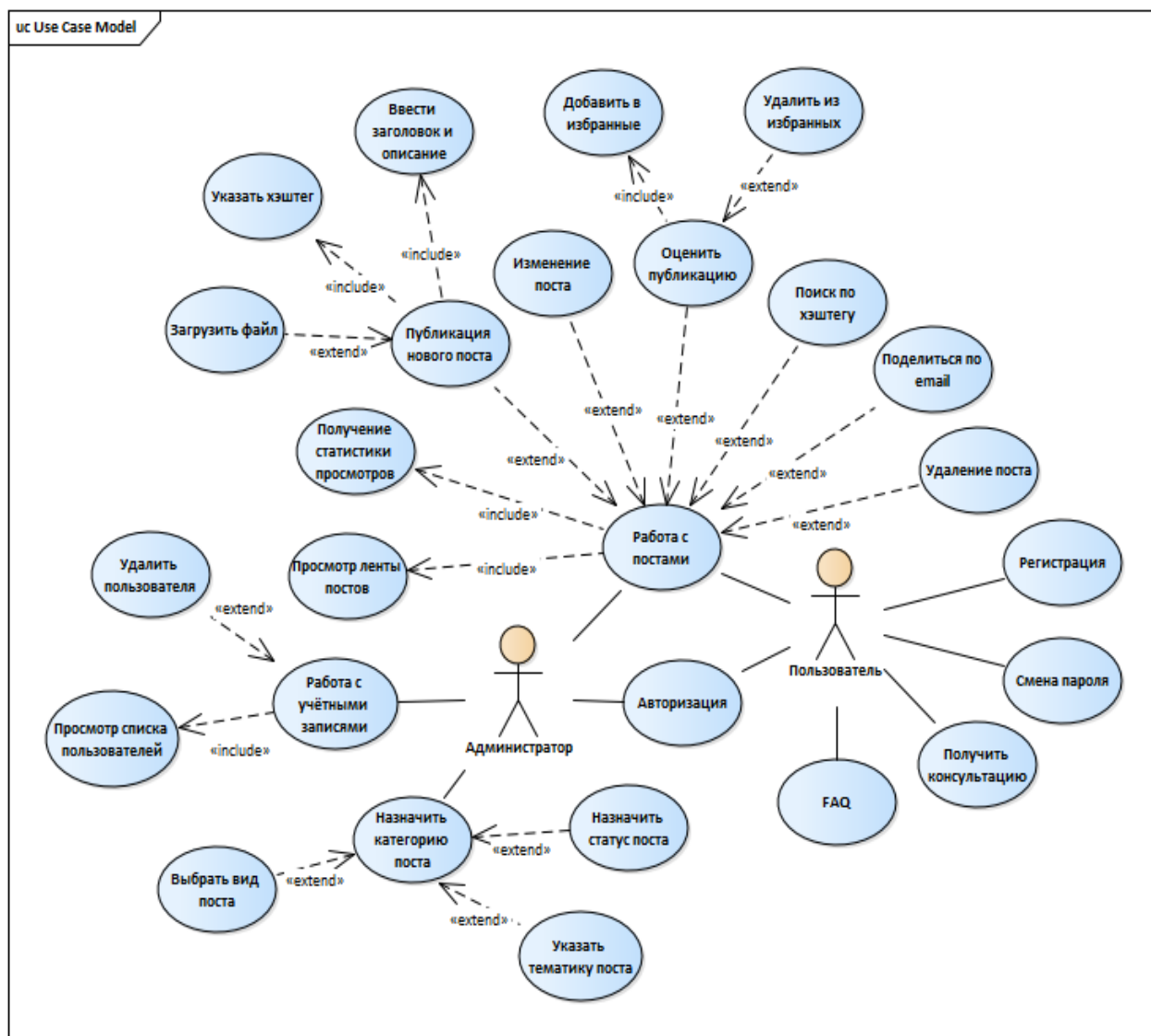


Рисунок 3.1 – Диаграмма вариантов использования системы

В разрабатываемой социальной сети выделяются две основные роли: администратор и пользователь. Как принято в большинстве систем – администратор обладает всеми возможностями обычного юзера, а также располагает своим индивидуальным пакетом функций.

Рассмотрим роль пользователя и его возможности. В самом начале перед использованием основного функционала сети пользователя требуется обучить

и ознакомить с тематикой контента. Именно поэтому пользователь ознакамливается с информацией о социальной сети в разделе FAQ. В случае если у пользователя возникнут проблемы с использованием социальной сети, либо же какие-то предложения по улучшению её работы, он может связаться с разработчиками, где получит необходимую консультацию. Пользователь может сомневаться в приватности своего пароля и захочет его сменить, сервис социальной сети позволит сменить пароль автономно без участия других человеческих ресурсов.

Администратор имеет список всей пользователей и может удалять пользовательские аккаунты в зависимости от причин. Так же в отличие от пользователя, администратор может редактировать и удалять любые посты в ленте. Пользователь же может редактировать те посты, где он является автором. Ещё одной отличительной особенностью является назначение категории поста администратором. Выбирается вид поста (рекламный, обычный), тематика (новости, объявления, оффтоп), статус (обычный, vip).

Основная логика работы социальной сети заключена в работе с постами. Любой пользователь может просмотреть новостную ленту и конкретный пост. При просмотре конкретного поста будет отображаться статистика просмотров, его автор. Чтобы добавить пост нужно ввести основную информацию в виде заголовка и описания, хэштег, и при необходимости загрузить изображение.

Если пользователю понравилась публикация другого пользователя он может оценить его добавив пост к себе в избранные. Каждый пользователь имеет свой список избранных публикаций. Со временем предпочтения пользователя меняются и в связи с этим имеется функция удаления поста из списка избранных. Есть возможность поделиться постом с другими людьми, не только пользователями социальной сети, отправив пост с интерактивным текстом на указанный email от лица социальной сети.

Так как в ленте имеется огромное количество постов пользователю нужна возможность фильтрации контента. Данная возможность реализована с помощью функции поиска по хэштегу. В зависимости от конкретного хэштега посты в ленте будут отфильтрованы, отсортированы по дате публикации и выведены на экран.

В данной главе был рассмотрен основной функционал социальной сети как со стороны пользователя, так и со стороны администратора без подробного описания всех функций.

## 4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЁ ОПИСАНИЕ

Информационная модель является информационной системой, которая описывает свойства и состояния различных объектов, процессов, а также явлений, показывает взаимосвязи между ними, параметры объекта на входе и на выходе. Из множества доступных баз данных, была выбрана и использована MySQL. Выбор аргументируется простотой и дружелюбностью интерфейса среды разработки для работы с базой. Имеются необходимый функционал для работы и его более чем достаточно. База данных курсового проекта предназначена для хранения данных о пользователях социальной сети их публикациях. [2]

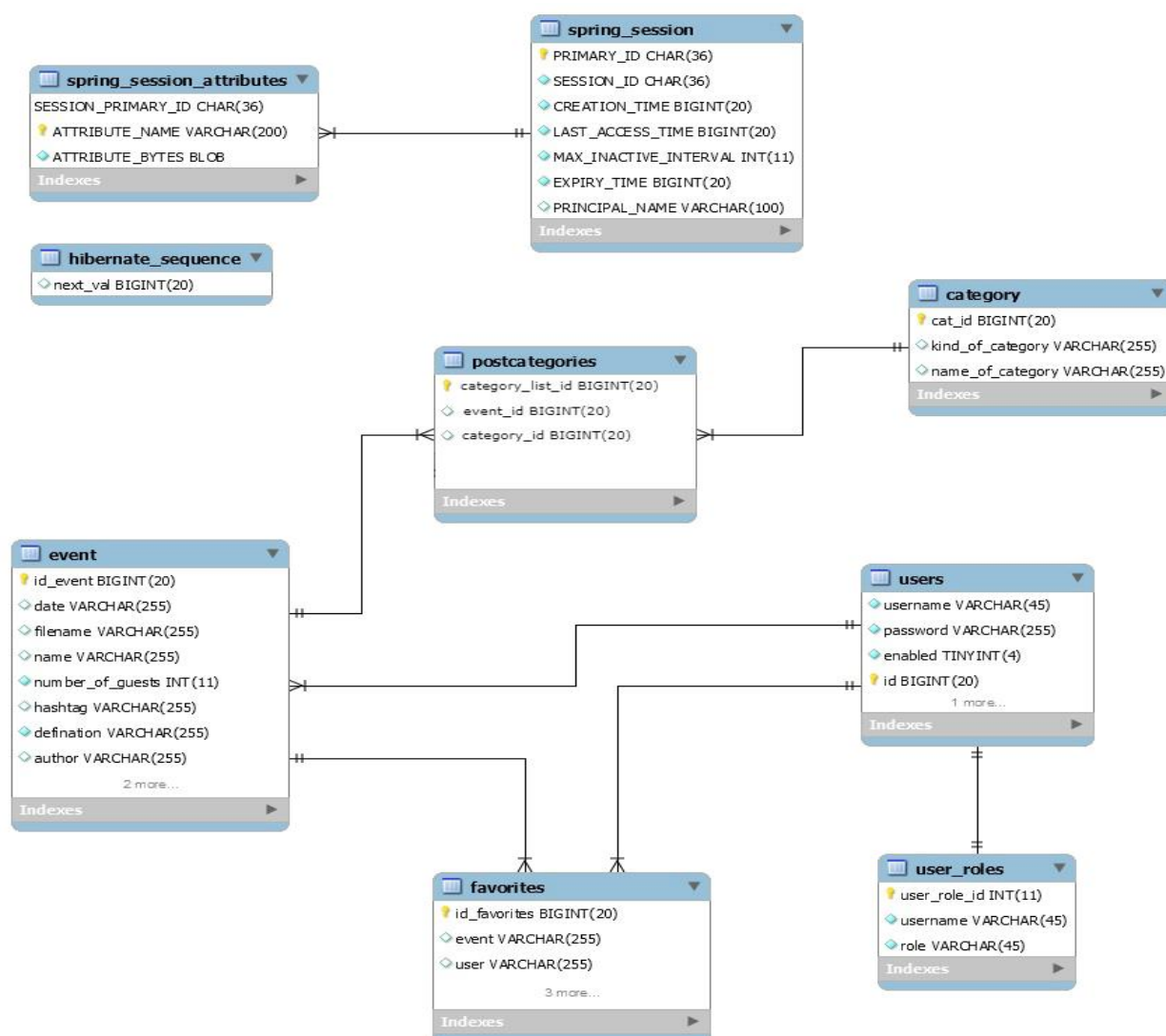


Рисунок 4.1 – Информационная модель базы данных

Выше приведена информационная модель - структурированная база данных с различными таблицами, связанными между собой (рисунок 4.1).

Перейдём к рассмотрению сущностей, используемых в курсовом проекте.

Сущность «user» хранит информацию об учётных записях пользователей социальной сети.

Атрибуты сущности «user»:

- id – первичный ключ и идентификатор конкретного пользователя;
- username – логин пользователя;
- password – пароль пользователя;
- enabled – показывает статус активности аккаунта.

Атрибуты «username» и «password» служат для аутентификации пользователя. Аккаунт юзера может быть активен и не активен, что отображает атрибут enabled. Используется для блокировки пользователей за различные нарушения.

Сущность «user\_roles» содержит информацию о роли пользователя в системе.

Атрибуты сущности «user\_roles»:

- user\_role\_id – уникальный идентификатор роли;
- username – логин пользователя принадлежащей роли;
- role – название роли (полномочия аккаунта в системе).

Атрибут «Role» определяет роль пользователя в программе. Существует только две роли: «Admin» и «User». Пользователь с ролью «Admin» обладает большими полномочиями и возможностями.

Сущность «favorites» представляет информацию об избранных публикациях, которые понравились юзеру.

Атрибуты сущности «favorites»:

- id\_favorites – уникальный идентификатор избранной публикации;
- event – название события добавляемого в избранные;
- user – логин пользователя, который добавляет к себе в избранные.

Сущность «category» включает в себе информацию о доступных категориях постов.

Атрибуты сущности «category»:

- cat\_id – уникальный идентификатор категории;
- kind\_of\_category – название типа категории;
- name\_of\_category – имя категории.

Сущность «post\_categories» охватывает данные о всех категориях публикации.

Атрибуты сущности «post\_categories»:

- category\_list\_id – уникальный идентификатор включённых категорий;
- event\_id – идентификатор публикации, к которой принадлежит категория;
- category\_id – уникальный идентификатор конкретной категории из доступных в системе.

Сущность «event» включает всю доступную информацию о посте.

Атрибуты сущности «event»:

- id\_event – уникальный идентификатор категории;
- date – дата публикации поста;
- filename – имя загружаемого файла, который будет храниться в системе;
- name – информативный заголовок публикации;
- number\_of\_guests – количество просмотров публикации;
- hashtag – хэштэг поста;
- definition – основное содержание публикации;
- author – автор публикации (юзер);

Таблицы, связанные связью один ко многим:

- user – event;
- user – favorites;
- event – postcategories;
- category – postcategories.

Таблицы user – user-roles связаны связью один к одному

База данных находится в 3-ей нормальной форме из-за следующих признаков:

База данных находится в 1-й нормально форме потому, что в сущностях не содержится копий одних строк, все атрибуты простые и понятного типа, множество допустимых значений, которые может принимать атрибут, содержат только скалярные значения.

Условия приведения базы данных ко 2-й нормальной форме является:

Приведение базы данных к 1-й форме (выше доказано). Необходимость уникального первичного ключа. Атрибуты каждой сущности соответственно характеризуют свой первичный ключ полностью, а не какую-то его часть. [3]

Критерии подтверждения приведения базы данных к 3-ей нормальной форме:

Доказательство приведения ко 2 форме, а также правильность суждения, что не должно быть зависимостей одних не ключевых атрибутов от других. Все атрибуты зависят только от первичного ключа.

## **5 ОБОСНОВАНИЕ ВЫБОРА КОМПОНЕНТОВ И ТЕХНОЛОГИЙ ДЛЯ РЕАЛИЗАЦИИ КУРСОВОГО ПРОЕКТА**

### **5.1 Описание ключевых технологий и обоснование их выбора**

Разрабатывая социальную сеть использовались передовые web-технологии, что во многом облегчило разработку программного кода, и упростило работу с системой в будущем, сделало её более удобной и гибкой для модернизации. Рассмотрим использованные технологии по подробнее.

Начнём с того, что приложения построено на основе Spring MVC application. Основной используемой технологией выступает spring framework для java. Именно он обеспечивает комплексную модель разработки и конфигурации, обеспечивает базовую поддержку управления зависимостями, управления транзакциями, доступ к данным, обмен сообщениями и многое другое.

Для передачи данных использовался известный протокол HTTP, основанный на TCP/IP. Протокол следует классической клиент-серверной модели, когда клиент открывает соединение для создания запроса, а затем ждет ответа от сервера.

В качестве стиля архитектуры приложения использовался REST, использованный для построения веб-служб. REST является очень простым интерфейсом управления информацией без использования каких-то дополнительных внутренних прослоек. Каждая единица информации однозначно определяется глобальным идентификатором, таким как URL. Каждая URL в свою очередь имеет строго заданный формат.

Для работы со spring framework были использованы следующие библиотеки: Spring Boot, Spring Data JPA и Spring Session.[4]

Spring Boot позволяет создать полноценное Spring-приложения, которое нужно просто запустить, прилагая минимальные усилия. В spring boot включена spring-платформа, tomcat или jetty (не требуется установки WAR файлов) и множество сторонних библиотек. Он автоматически конфигурирует Spring когда это возможно, используется без генерации кода и без написания XML конфигурации. Обеспечивает возможностями: метрики, мониторинг состояниями и расширенная конфигурация.

Spring Data делает проще использование технологий доступа к данным. Она делиться на подпроекты в зависимости от конкретной СУБД. Для работы с базой данных была использована технология Spring Data JPA. Этот модуль расширяет поддержку JPA-слоя доступа к данным. Как



разработчик, вы пишете интерфейс репозитория, включая собственные методы поиска, а Spring обеспечивает их автоматическую реализацию. Главным преимуществом Spring Data является сокращение огромного количества шаблонного кода. [5]

JPA (Java Persistence API) это спецификация Java EE и Java SE, описывающая систему управления сохранением java объектов в таблицы реляционных баз данных в удобном виде. [6]

## **5.2 Описание диаграммы развёртывания**

Перейдём к рассмотрению диаграммы развёртывания изображённой в приложении А на рисунке А.1.

Устройством, поддерживающим браузер может выступать персональный компьютер, телефон и прочие гаджеты, которых в современном мире полно. Они могут находиться абсолютно в разных уголках света и взаимодействовать с веб страницами браузера по средствам протокола. Взаимодействуя с ними, пользователи отправляют различные запросы на сервер приложения, являющегося контроллером. Контроллер в свою очередь обрабатывает эти запросы и взаимодействует по протоколу TCP/IP с различными сервисами включая ORM, которая обращается к серверу базы данных. Тем самым контроллер получает доступ к базе данных для CRUD операций и другие возможности.

## **5.3 Описание диаграммы компонентов**

Разберём построение диаграммы компонентов, представленной в приложении А на рисунке А.2.

Первым компонентом диаграммы выступает браузер, а именно веб-страницы. Веб страницы представляют собой отдельный документ или ресурс в интернете. С помощью браузера выполняется взаимодействие с сервером. Под сервером подразумевается работа с системой контроллеров, которые обмениваются данными с веб сервисами, и отвечают за отображение информации на веб страницах, которые просматривает пользователь. Веб сервисы (основная логика программы) взаимодействуют со слоем ORM. Слой ORM взаимодействует с сервером по протоколу TCP/IP, где размещена нужная проекту база данных и предоставляет данные веб сервисам. [7]

## 5.4 Описание диаграммы последовательности

Рассмотрим диаграмму последовательности процесса добавления категорий в приложении А на рисунке А.3.

Пользователь (администратор) взаимодействует с клиентом (браузером) выбирая конкретную публикацию, в которой он хочет добавить определённые категории. По клику на публикацию выполняется запрос на её получение. С серверной стороны приходят данные, которые заполняют форму на клиенте и отображаются в виде поста пользователю. После пользователь по клику на кнопку «добавить категории» выбирает необходимые ему значения. На клиенте проверяются и обрабатываются выбранные данные. После обработки данные отправляются к веб сервису, который передаёт их ORM слою. ORM слой обращается к базе данных на сохранение изменений и получает ответ. Далее данные передаются в обратном порядке и в конечном итоге отображаются на экране пользователя.

## **6 МОДЕЛИ ПРЕДСТАВЛЕНИЯ ДАННЫХ И ИХ ОПИСАНИЕ**

### **6.1 Диаграмма состояний**

Проанализируем имеющуюся диаграмму состояний процесса добавления публикации в приложении А на рисунке А.4. Диаграмма описывает состояние добавления публикации. Сначала при публикации поста нужно выбрать файл загрузки. В качестве файла, может выступать только изображение, которое будет отображаться при просмотре публикации. Далее следует ввести обязательные поля (заголовок и хэштэг), при желании добавить описание. Если данные прошли успешно валидацию – они отправляются на сервер. Сервер принимает данные публикации, генерирует уникальное имя файла, устанавливает сегодняшнюю дату и записывает публикацию в базу данных. После записи сервер отправляет данные на клиентское приложение об успешной записи. Клиент обрабатывает полученные данные и в случае успеха открывает страницу публикации.

### **6.2 Блок схемы, реализующие бизнес-логику системы**

На рисунке Б.1 в приложении Б продемонстрирована работа алгоритма добавления публикации. В начале алгоритма вводятся необходимые поля, такие как заголовок, хэштэг и описание, в котором раскрывается главная мысль публикации. Определяется имеется ли изображение, прикрепляемое к публикации, так как изображения не является обязательным. Если изображение имеется, то выполняется загрузка изображения с устройства пользователя и подготовка его к отправке на сервер. Выполняется проверка, чтобы пользователь не мог оставить пустых полей и неподходящих значений. Если данные некорректны – пользователю придётся ввести заново правильные значения. В случае валидных данных сервер генерирует уникальное имя файлу для хранения в базе данных, устанавливает время, когда была произведена публикация и записывает в дату поста. Также указывается автор, который опубликовал публикацию. Вся проанализированная и обработанная информация записываются в базу данных. После успешной записи на экран пользователя выводится информация о созданной публикации.

На рисунке Б.2 в приложении Б отображена схема работы алгоритма при выборе категорий публикации. Изначально, чтобы добавить категорию, нам нужно выбрать конкретную публикацию кликнув на неё. Далее следует указать все необходимые категории.

## 7 ОПИСАНИЕ ПРИМЕНЕНИЯ ПАТТЕРНОВ ПРОЕКТИРОВАНИЯ

В разработке и реализации курсового проекта использовались два паттерна проектирования: MVC, Repository, Singleton. Их наглядное применение можно увидеть на диаграмме классов в приложении А рисунок А.5.

Архитектурный шаблон «MVC» или «Model-View-Controller» использовался для разделения приложения на три функциональные части: модель данных (model), пользовательский интерфейс (view) и управляющую логику (controller).

Модель - это данные с которыми непосредственно взаимодействует контроллер. Эти данные содержат различную информацию, операции, преобразования и правила их использования. За счет независимости от представления, есть возможность реализации нескольких различных представлений для одной модели.

Контроллер – обрабатывает запросы пользователя, т.е. выступает как интерпретатор действий пользователя. В случае изменения некоторых данных, взаимодействует с моделью оповещая её. Занимается выборкой представления для визуализации.

Представление – уровень схемы, который отвечает за отображение данных из модели к пользователю. Не отвечает за обработку данных, введенных пользователем, так как этим занимается контроллер, но изменяет своё отображение (представление), в зависимости от данных модели.

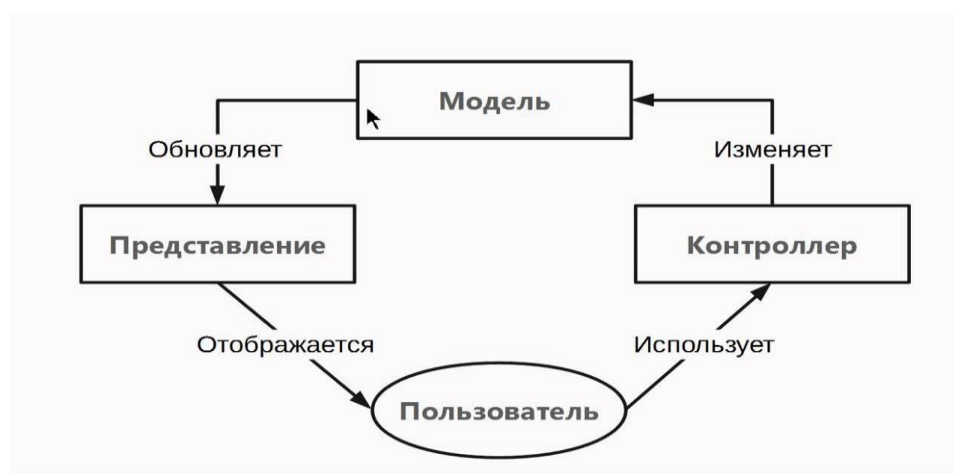


Рисунок 7.1 – Архитектурный паттерн MVC

В курсовом проекте использовался фреймврк Spring MVC, который обеспечил реализацию архитектуры паттерна MVC. Логика использования Spring MVC сосредоточена на работе DispatcherServlet, который принимает и обрабатывает все HTTP-запросы (из UI) и отправляет ответы на них. Процесс обработки запроса и отправки ответа DispatcherServlet'ом продемонстрирован на следующей диаграмме:

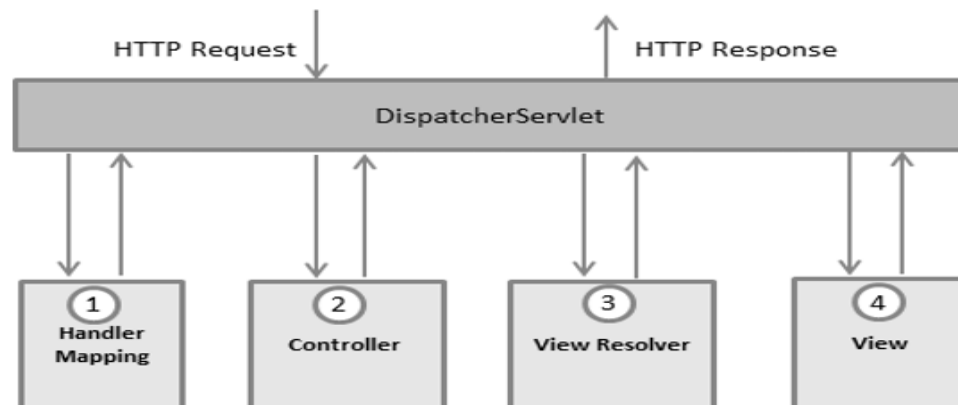


Рисунок 7.2 – Схема работы DispatcherServlet

Когда http запрос получен, обращаясь к интерфейсу Handler Mapping определяется контроллер, который необходимо вызвать, после этого запрос направляется в этот контроллер.

Контроллер принимает запрос и вызывает соответствующий метод (get, post, put, delete). Вызванный метод определяет данные Модели, основанные на определённой бизнес-логике и возвращает в DispatcherServlet имя Вида (View). При помощи интерфейса ViewResolver DispatcherServlet определяет, какой Вид нужно использовать на основании полученного имени. После того, как Вид (View) создан, DispatcherServlet отправляет данные Модели в виде атрибутов в Вид, который в конечном итоге отображается в браузере.

```
@Controller
@RequestMapping("/hello")
public class HelloController {
    @RequestMapping(method = RequestMethod.GET)
    public String printHello(ModelMap model) {
        model.addAttribute("message", "Hello Spring MVC Framework!");
        return "hello";
    }
}
```

Рисунок 7.3 – Пример определения контроллера в Spring MVC

В качестве view использовался шабланизатор Apache FreeMarker - это механизм шаблонов: библиотека Java для генерации текстового вывода (HTML-страницы, xml, файлы конфигурации, исходный код и.т.д. На вход подается шаблон, например, html в котором есть специальные выражения, подготавливаются данные соответствующие этим выражением, а Freemarker динамически вставляет эти данные и получается динамически заполненный документ.



Рисунок 7.4 – Схема работы шабланизатора FreeMarker

Следующий паттерн, который применялся при разработке это Repository. Если речь идёт о работе с данными, то паттерн Repository выступает лучшим решением. Он позволяет абстрагироваться от конкретных подключений к источникам данных и избежать технических зависимостей. Выступает буфером между классами, которые взаимодействуют с данными и остальной часть программы.

Представим ситуации где имеется существующее подключение к базе данных MySQL, но в друг в определённом момент возникла необходимость в использовании другой базы данных. При стандартном подходе пришлось бы внести много изменений. В этой ситуации Repository добавляет гибкость программе при работе с разными типами подключений.

Другим преимуществом выступает сокращение написания большего количества шаблонного кода. Можно рассматривать Repository как коллекцию объектов. Можно добавить и удалить объекты из репозитория, как из простой коллекции объектов, в то время как осуществляется работа Repository с базой данных. Объект репозитория, обеспечит проведение соответствующих операций.

Репозиторий инкапсулирует объекты, находящиеся в базе данных, но не только объекты, но и методы работы с ними, таким образом явно отображая концепцию объектно-ориентированного подхода.

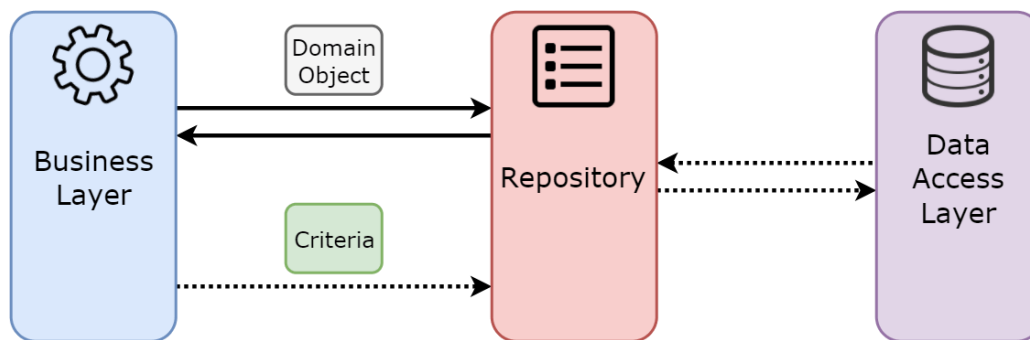


Рисунок 7.5 – Паттерн «Repository»

```

@Repository
public interface UsersRepository extends JpaRepository<Users, Integer> {
    Users getUserByLoginAndPassword(String login, String password);
    Optional<Users> getUserByLogin(String login);
    List<Users> getAllByPost(String post);
}
  
```

Рисунок 7.6 – Пример использования паттерна «Repository»

Паттерн Singleton нужен для того чтобы определить класс, у которого будет только один экземпляр, и к которому будет предоставлен глобальный уровень доступа. Другими словами, класс должен обеспечивать создание только одного объекта и этот объект может быть использован всеми остальными классами в приложении. [8]

Показать использования singleton можно при определении bean в Spring Framework. Если bean объявлен как singleton, это означает, что контейнер Spring IoC создает единственный экземпляр этого компонента, и все запросы на имя этого компонента возвращают один и тот же объект, который кэшируется.

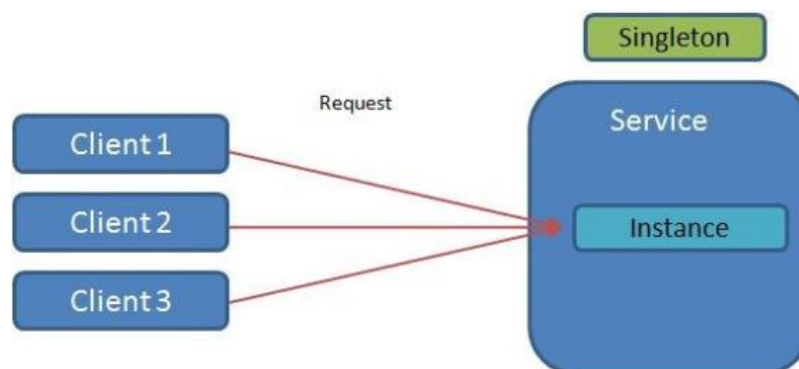


Рисунок 7.7 – Использование паттерна «Singleton»

## 8 РУКОВОДСТВО ПО РАЗВЁРТЫВАНИЮ СИСТЕМЫ

При старте программы автоматически запускается сервер и клиент. После запуска программы перед пользователем появляется главная страница сайта (рисунок 8.1).

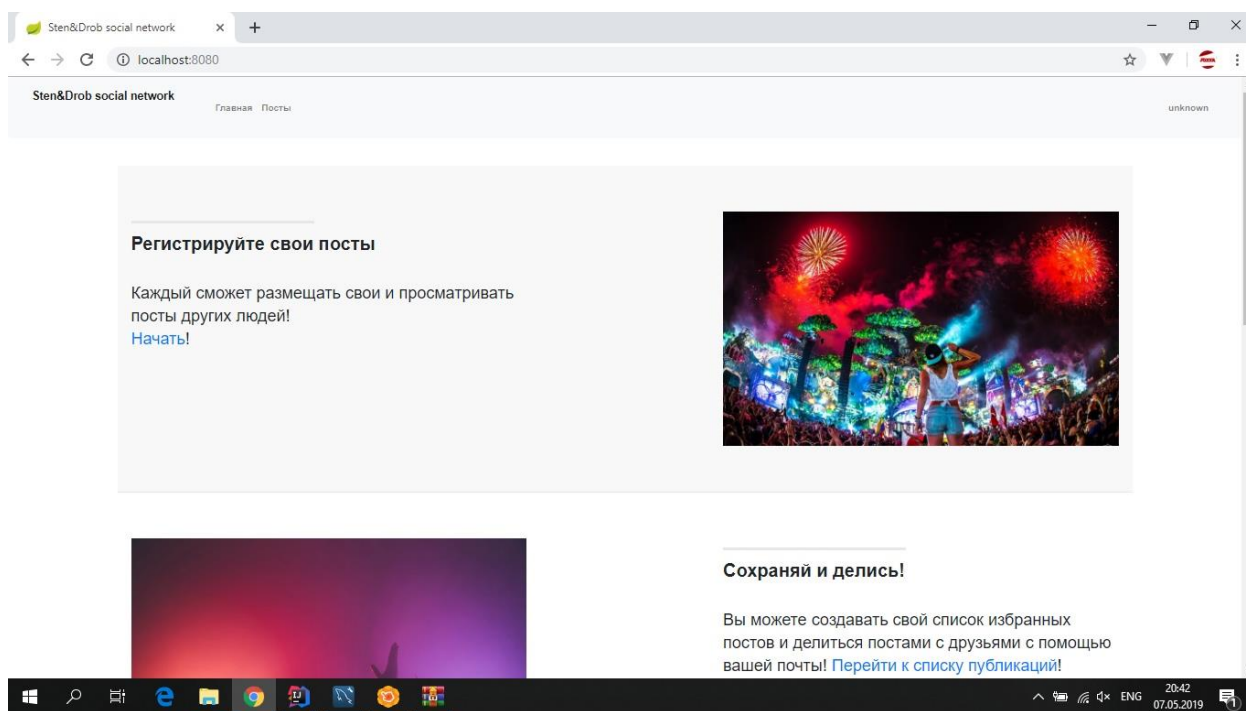


Рисунок 8.1 – Главная страница сайта

На главной странице сайта пользователю предоставляется основная тематика социальной сети, её идея, способы использования и всё что необходимо для того чтобы приступить к использованию. Ссылка «Начать» переадресует пользователя на страницу регистрации (рисунок 8.3), так как основным условием использования полного функционала социальной сети является регистрация. Именно регистрация обеспечит пользователю основной функционал работы с публикациями.

Если пользователь не совсем понял задумку или направление социальной сети, он может перейти к последним публикациям других пользователей после регистрации и просмотреть их с помощью ссылки «Перейти к списку публикаций». Это сделано, чтобы ознакомить пользователей с контентом сети и сформировать окончательное мнение.

В нижней части главной страницы есть возможность пользователю связаться с командой разработчиков, если у него возникли вопросы или же он хочет внести предложения по улучшению работы социальной сети. Если



кликнуть на кнопки «twitter», «vk», «linkedin» то пользователь перейдёт на соответствующие странички в других социальных сетях (рисунок 8.2).

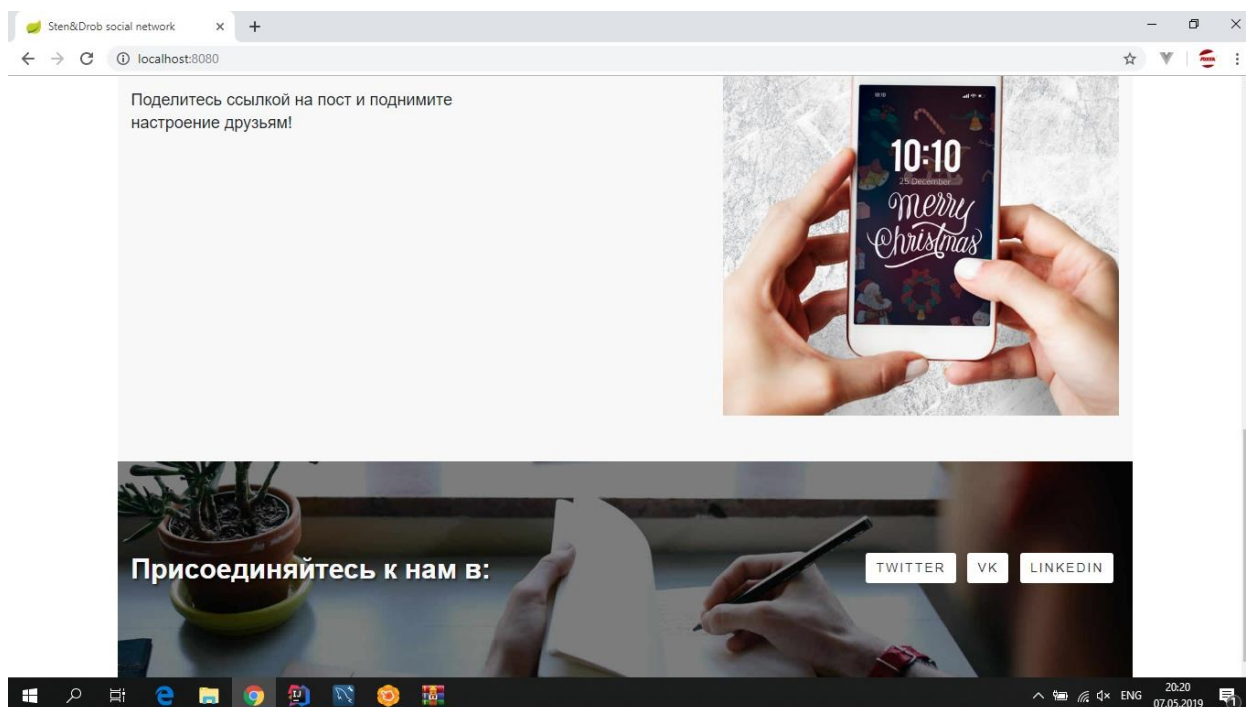


Рисунок 8.2 – Раздел главной страницы «Присоединяйтесь к нам в»

A screenshot of a web browser window displaying a registration form. The browser's address bar shows 'localhost:8080'. The page content includes a header with the text 'Sten&Drob social network' and 'Главная Посты'. The registration form has three input fields: 'Логин' (User name), 'Пароль' (Password), and 'Повторите пароль' (Retype password). Below these fields is a CAPTCHA image with the text 'Я не робот' and 'geCAPTCHA'. A blue button labeled 'Зарегистрироваться' is at the bottom.

Рисунок 8.3 – Регистрация пользователя в сети

При регистрации пользователя необходимо заполнить следующие поля:

- логин;
- пароль;
- повтор пароля.

Логин должен быть уникальным, корректным, так как при публикации поста именно это имя будет представлять автора и отображаться в посте.

Пароли должны обязательно совпадать чтобы исключить возможность ошибки при напечатании. Чтобы избежать различного рода вредоносных программ, которые могут негативно воздействовать на приложение используется капчка разработанная google. Нужно поставить галочку, для подтверждения того что вы не робот. Окончательным этапом будет нажатие кнопки «Зарегистрироваться».

Если пользователь уже имеет аккаунт он может авторизоваться без особо труда, введя свои персональные данные.

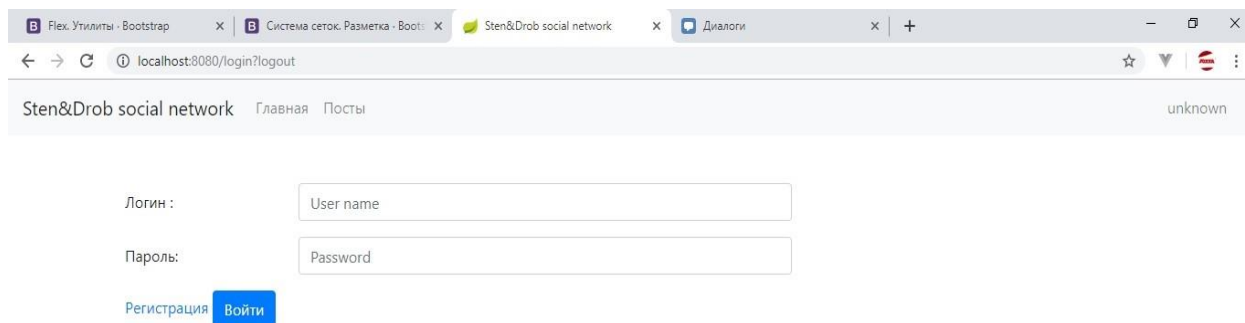


Рисунок 8.4 – Авторизация пользователя в сети

После регистрации пользователя автоматически перенаправит на страницу всех публикаций, выглядит она следующим образом (рисунок 8.5).

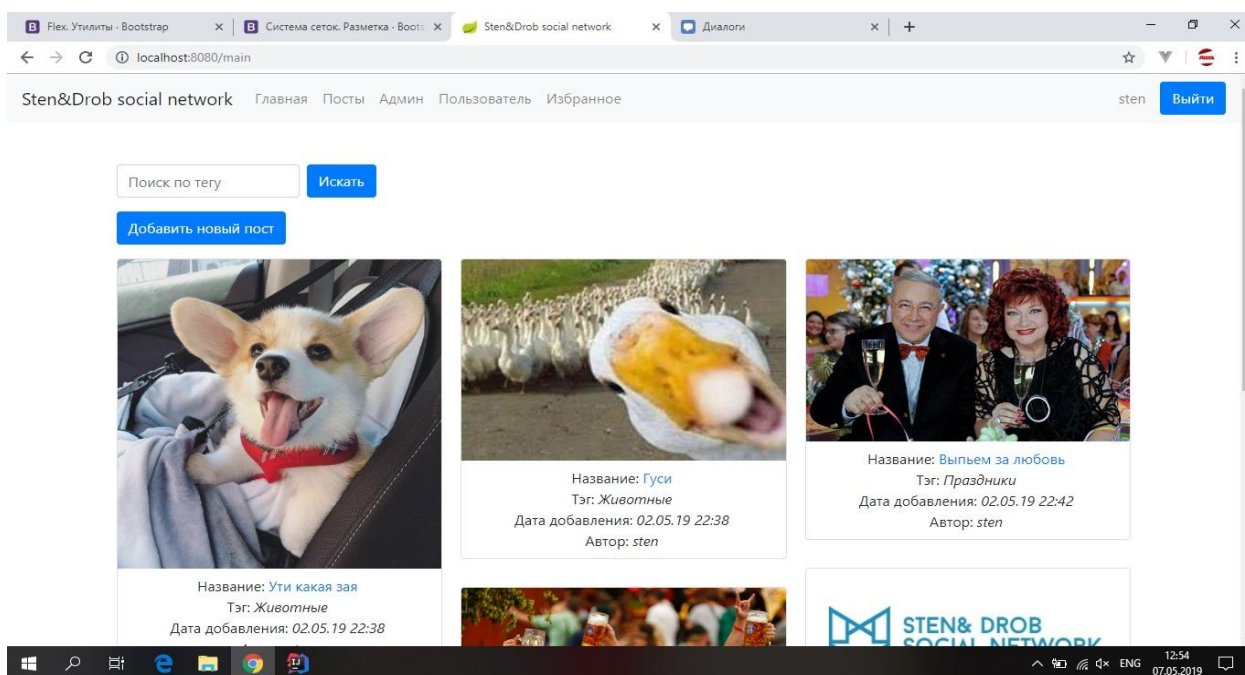


Рисунок 8.5 – Страница публикаций пользователей

Справа вверху находится никнейм пользователя рядом с кнопкой «выход». В верхней части сайта имеется навигация для перехода между страницами. Чтобы облегчить пользователю возможность поиска и сортировки, можно воспользоваться функцией «поиск по тегу». Чтобы ей воспользоваться нужно ввести необходимый тег, например, «Животные» и нажать кнопку «Поиск». Записи будут отфильтрованы по тегу и отсортированы по дате публикации.

Перейдём к рассмотрению поста пользователя автором которого является сам пользователь (рисунок 8.6).

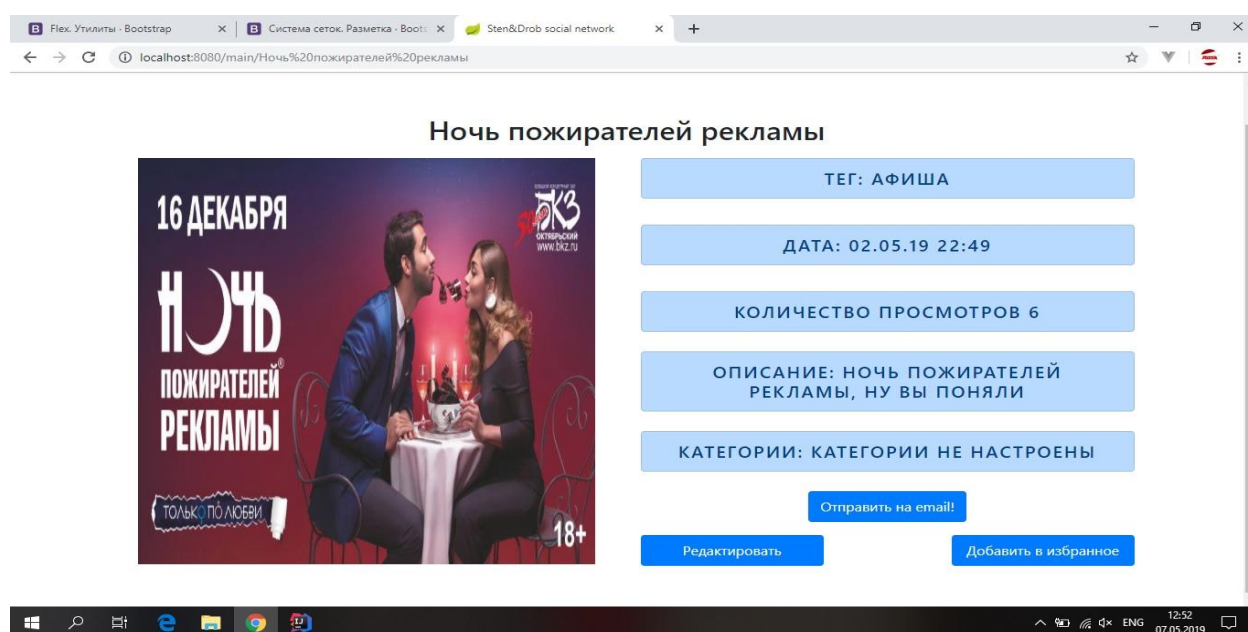


Рисунок 8.6 – Публикация со страницы пользователя

В самом верху публикации расположен заголовок, он выделен жирным шрифтом. Слева расположено фото публикации, которое загрузил пользователь. Справа основная информации с тегом, датой публикации, количеством просмотров, кратким описанием.

Если рассматривать функционал со страницы пользователя, то он довольно скромнен по сравнению с администратором. Пользователь может редактировать только свою публикацию, в то время как администратор любую. Кнопка «Добавить в избранное» добавит выбранную публикацию к списку избранных публикаций пользователя. Он может посмотреть все свои избранные публикации нажав на кнопку в навигации «Избранное». Количество добавления постов в избранное неограниченно. Так же доступна возможность поделиться записью с другими людьми, в не зависимо от того, являются ли они пользователями социальной сети. При нажатии кнопки

«отправить на email», будет отправлено письмо на email, который укажет пользователь.

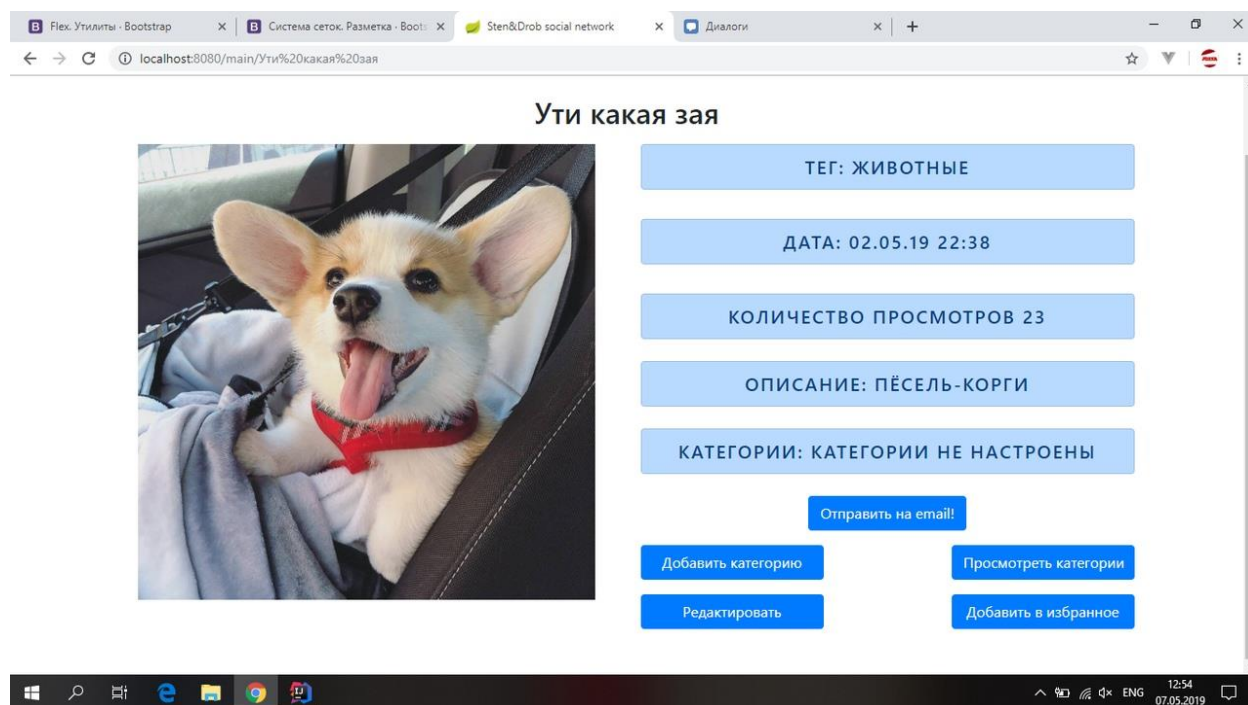


Рисунок 8.7 – Публикация со страницы администратора

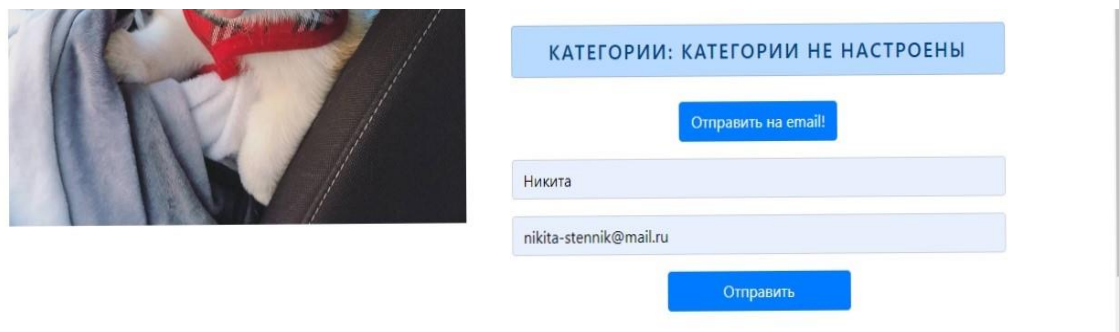


Рисунок 8.8 – Отправка на email

Администратор может назначать категорию поста, пользователь только просматривать. Добавление категории отображено на рисунке ниже.

Вид поста может быть обычным или рекламным в зависимости от целей публикации. Выбор тематики поста позволяет определить контент и целевую аудиторию поста. Что касается статуса поста то он позволяет постам со статусом «vip» быть выше в ленте публикаций чем «simple».

Добавление публикации не должны вызывать у пользователя каких-либо сложностей, так как интерфейс страницы добавления интуитивно понятен.

Sten&Drob social network Главная Посты Админ Пользователь Избранное sten Выйти

**Вид поста:**

- ☐ Рекламный
- ☐ Обычный

**Тематика поста:**

- ☐ Новости
- ☐ Объявления
- ☐ Оффтоп

**Категория значимости поста:**

- ☐ VIP
- ☐ simple

Добавить категории

Рисунок 8.9 – Добавление категории публикации администратором

Sten&Drob social network x 1 новое сообщение x B Группа списков. Компоненты x +

localhost:8080/main sten Выйти

Поиск по тегу Искать

Добавить новый пост

Введите тег

Введите заголовок

Введите описание

Choose file Browse

Добавить




Рисунок 8.10 – Создание публикации

Нужно выбрать картинку для загрузки в качестве файла. Указать заголовок, тег и описание публикации.

Возможно пользователь захочет изменить свой пароль. Данная возможность предусмотрена и реализована на странице «Пользователь».



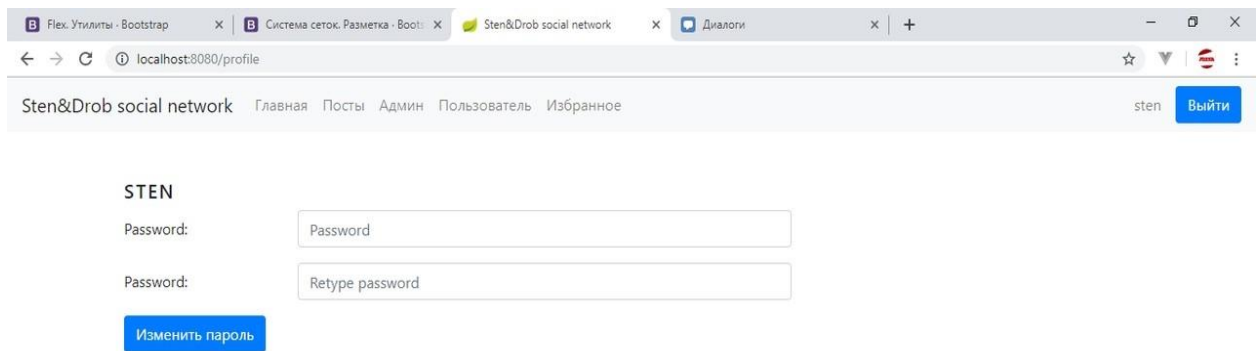


Рисунок 8.11 – Изменение пароля пользователя

Администратор может блокировать пользователей за определённые нарушения. В своём меню он имеет список всех пользователей, а также список всех постов.

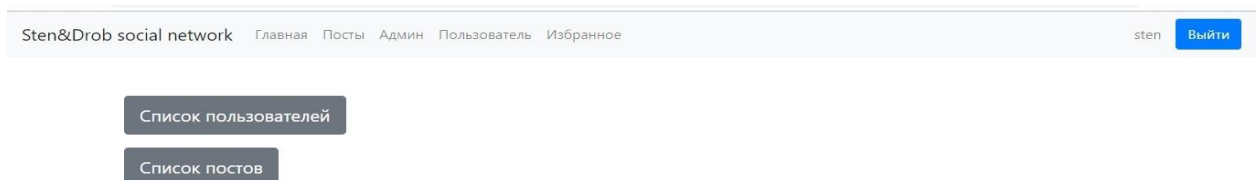


Рисунок 8.12 – Меню администратора

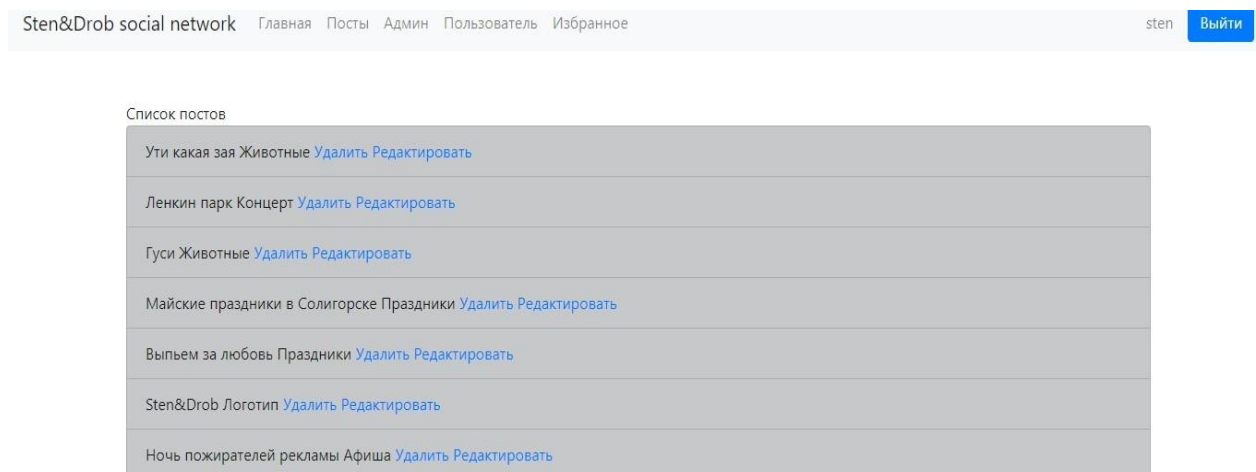


Рисунок 8.13 – Управление публикациями

Список пользователей

sten	<a href="#">Блокировать</a>
user	<a href="#">Блокировать</a>
shakree	<a href="#">Блокировать</a>
Vladkoo	<a href="#">Блокировать</a>

Рисунок 8.14 – Управление пользователями

В данном разделе был продемонстрирован почти весь функционал курсового проекта. Были затронуты ключевые моменты воздействия пользователя на систему и наоборот. Можно сделать вывод что система не обладает сложным интерфейсом, проста для понимания неопытным пользователем, а самое главное обладает рабочим интерфейсом. Развёртывание системы не является трудоёмким процессом, так что любой пользователь интернета сможет использовать разработанную социальную сеть.

## 9 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ

При разработке социальной сети были продуманы ключевые аспекты обработки ошибок и защиты системы от критических повреждений. Потенциальным источником опасности для системы выступает неопытный пользователь. В связи с этим при работе с социальной сетью учтены распространённые ошибки пользователей. Обработка ошибок очень важна, как и для системы так и для пользователя. Система должно подсказать пользователю, что он делает не так и помочь реализовать его задумку. Если в разработанной системе не будет подсказок для пользователя, то ей просто не станут пользоваться в связи со сложностью использования и не дружелюбным интерфейсом.

Рассмотрим основные аспекты обработки ошибок. Если пользователь не имеет аккаунта и попробуем перейти на страницу новостей, то сделать он этого не сможет. Для работы с социальной сетью, регистрация является обязательным условием.

При регистрации пользователю требуется ввести свой уникальный логин. Но не редко несколько пользователей могут ввести один логин. Чтобы избежать проблем, которые могут возникнуть у системы, предусмотрена проверка на дубликацию логина при регистрации. Система уведомит пользователя что такой логин уже используется и сообщит выбрать другой.

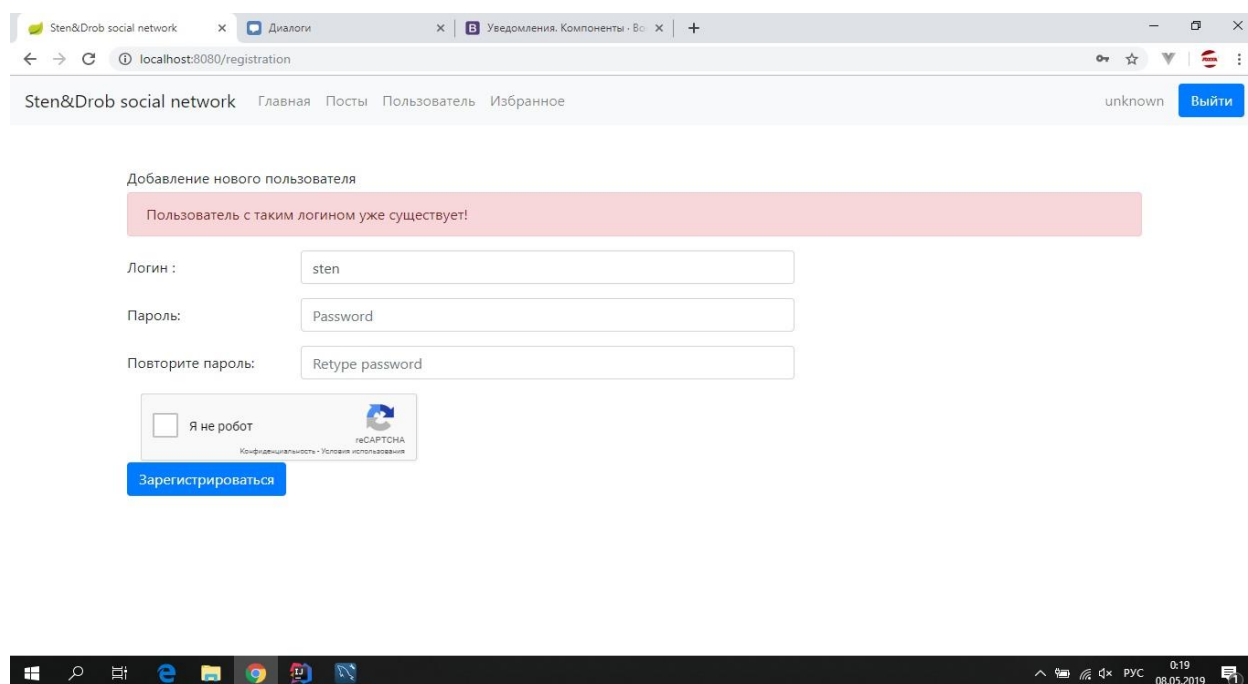


Рисунок 9.1 – Попытка регистрации существующего логина



Ещё одна проблема которая может возникнуть – это наличие пустых полей. Пользователь из-за невнимательности или преднамеренно может оставить некоторые поля пустыми. Также при регистрации требуется чтобы два ведённых пароля совпадали между собой, это исключает возможность опечатки пользователя. Чтобы социальная сеть не содержала фейковых аккаунтов, зарегистрированных специальными ботами, используется капча от google. Капча – это компьютерный тест для определения кем является пользователь системы: человеком или роботом.

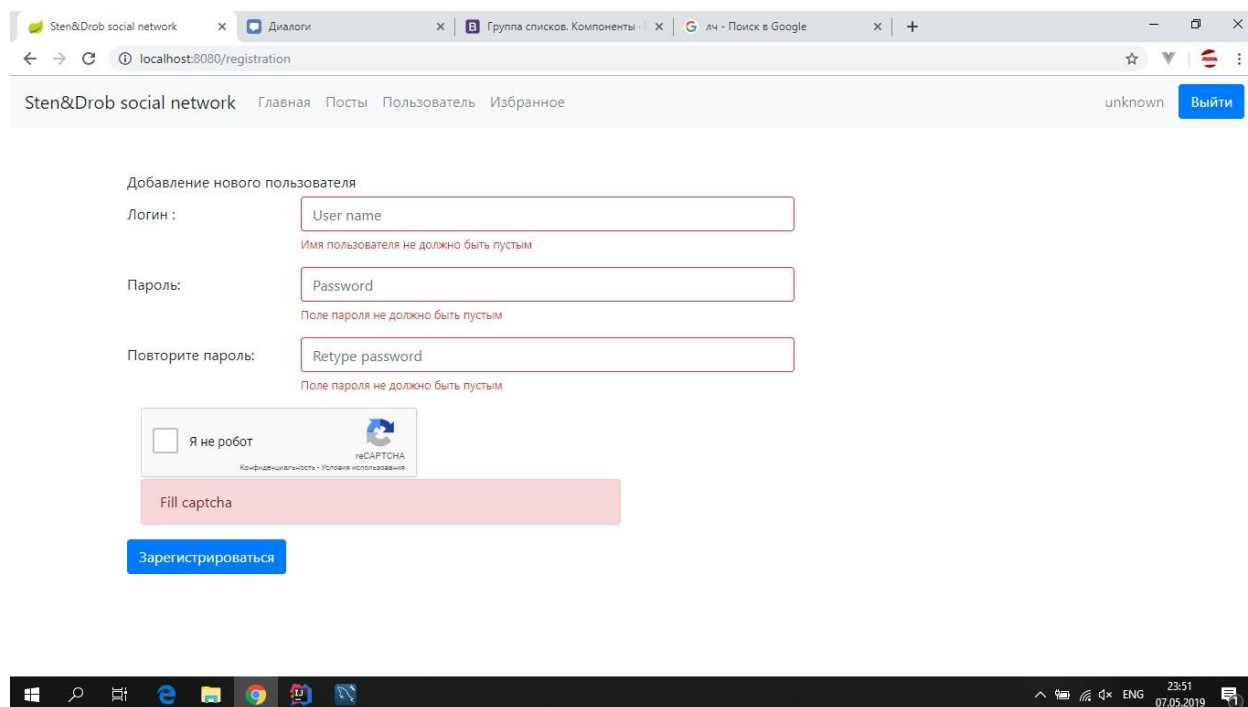


Рисунок 9.2 – Валидация пустых полей при регистрации

При условии пользователь, что пользователь зарегистрирован, он может авторизовать в системе и войти под своим аккаунтом. Естественно предусмотрена проверка на корректность и правдивость логина и пароля от учётной записи (рисунок 9.3).

Проблемы так же могут возникнуть при добавлении поста. Не смотря на то что интерфейс добавления публикации интуитивно понятен, имеются необходимые поля, которые нужно заполнить: тег и заголовок. Остальные поля включая загружаемый файл могут быть пустыми (рисунок 9.4).

Имеется валидация при отправке письма с публикацией на email другого человека. Требуется указать существующий и корректный email. Это сделано для того чтобы не нагружать систему не нужными запросами и избежать ошибки ввода пользователем (рисунок 9.5).

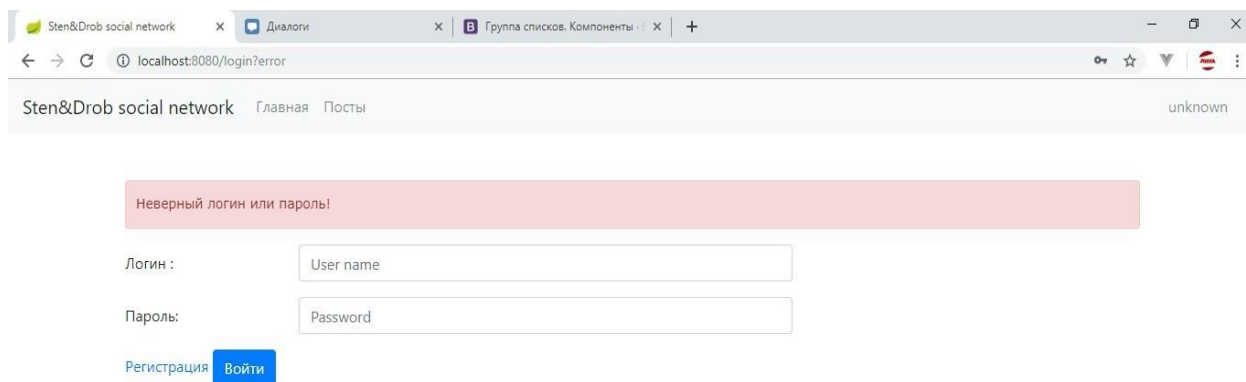


Рисунок 9.3 – Неверный логин или пароль

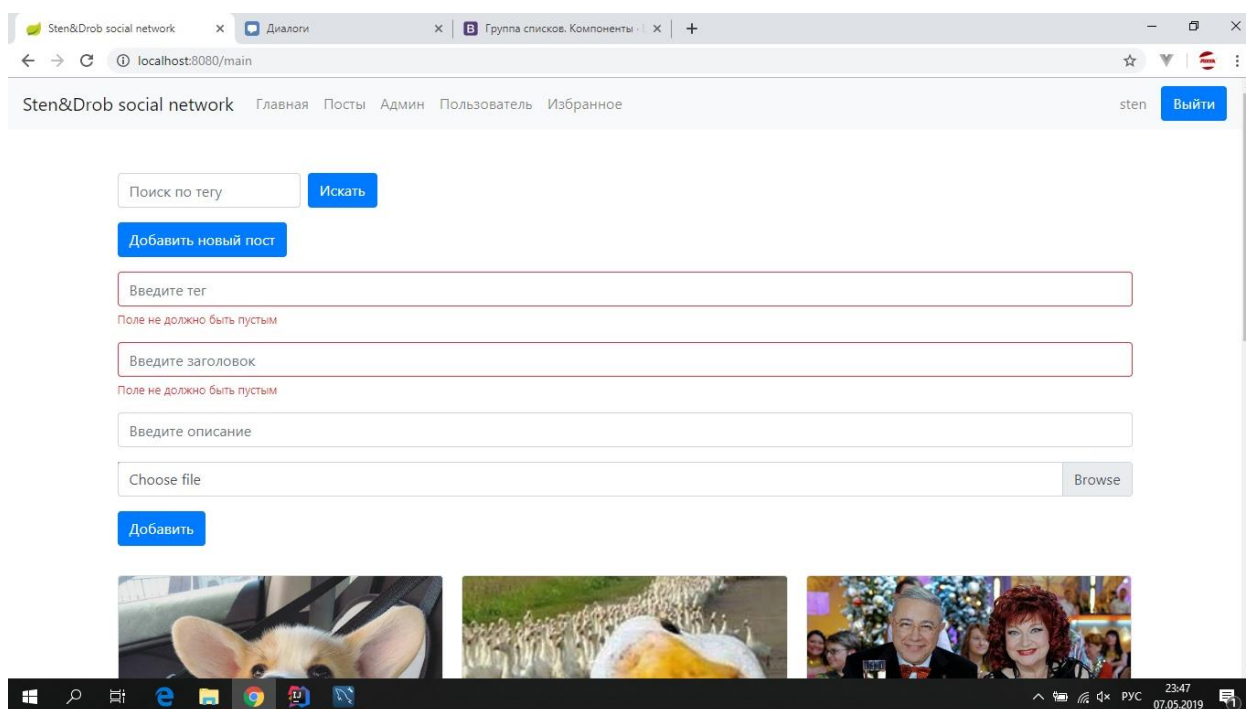


Рисунок 9.4 – Пустые поля при добавлении публикации

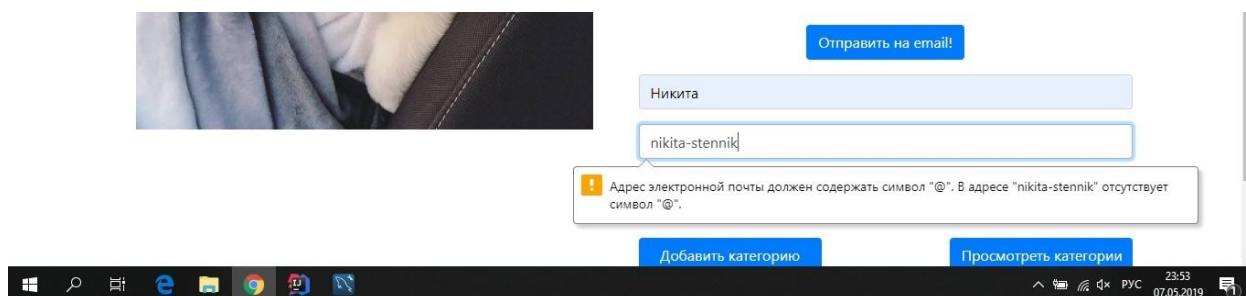


Рисунок 9.5 – Валидация email при отправке

## ЗАКЛЮЧЕНИЕ

В ходе написания курсовой работы был исследован процесс взаимоотношения пользователей в сети. В связи с этим было спроектировано программное средство, позволяющее уменьшить трудозатратность, повысить комфорт и удобство обмена информацией. Разработана иерархия классов, в связи с этим написание, когда ощутимо облегчилось, что позволило сделать код модулярным.

Разработанная социальная сеть обеспечивает удобную работу, помогает быстро взаимодействовать с другими юзерами в сети, следить за обновлениями других пользователей, заполнять обширную и детальную информацию о себе, а также публиковать сообщения у себя на стене.

Спроектирована база данных, содержит данные о пользователях сети, их постах, информация о контенте постов, их типе, об избранных постах пользователя и прочие данные. Реализована возможность добавления, редактирования, удаления и получения нужных данных с помощью выборки.

Делая вывод, хочется сказать, что разработанная система обладает простым и комфортным интерфейсом, интуитивно понятную навигацию и позволяет эффективно обмениваться информацией. Вся информация о конкретном пользователе хранится в удобной форме в базе данных. Социальная сеть устойчива к ошибкам совершаемым пользователем, то есть исключается возможность появления технических ошибок при работе внутри сети. Данная социальная сеть может быть использована всеми пользователями в Республике Беларусь в любом регионе или населённом пункте с доступом в интернет.

Социальная сеть имеет хорошие перспективы по расширению своей инфраструктуры, функциональности. Может быть масштабируема и доработана под определённые требования страны, а также внедрена в короткие сроки.

Результатом работы является готовая социальная сеть, которая своими основными возможностями характеризует установленные к ней современные требования. Учтены наиболее востребованные потребности пользователей, такие как защищённость их личных данных, свободное высказывание мнение и подбор определённого качественного контента.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Перспективы развития социальных сетей [Электронный ресурс]. – Режим доступа <https://secl.com.ua/article-vse-o-socialnyh-setjah-perspectivy-razvitiya.html#part1>
- [2] Справочное руководство MySQL [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://www.mysql.ru/docs/man/>
- [3] Нормализация отношений. Шесть нормальных форм [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://habr.com/post/254773/>
- [4] Крейг Уоллс Spring в действии 4-е издание: Практическое пособие. – ДМК Пресс: Москва, 2013. – 754 с.
- [5] Общий раздел по Spring MVC [Электронный ресурс]. – Режим доступа <http://javastudy.ru/frameworks/spring/spring-mvc/>.
- [6] Блинов И.Н., В.С. Романчик. Java: методы программирования. Минск: «Четыре четверти», 2013. — 896 с.
- [7] UML. Классика CS. 2-у изд./Пер. с англ.; Под общей редакцией проф. С.Орлова - СПб.: Питер, 2006. - 736 с.
- [8] Паттерны проектирования [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://refactoring.guru/ru/design-patterns>

# **ПРИЛОЖЕНИЕ А** **(обязательное)** **UML Диаграммы**

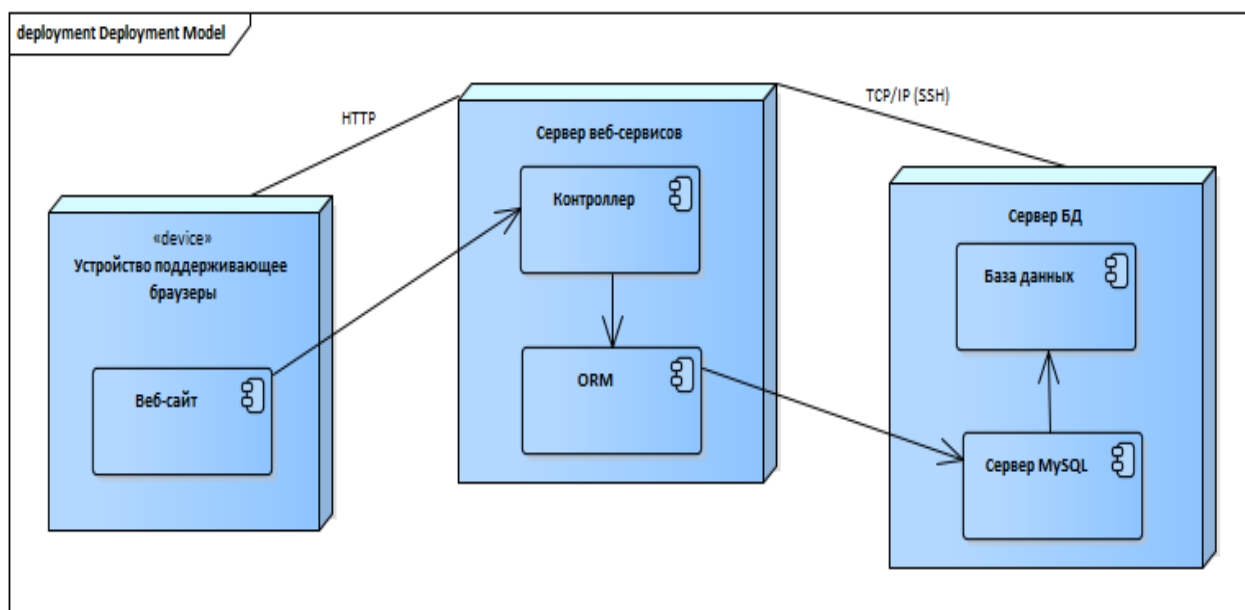


Рисунок А.1 – Диаграмма развёртывания

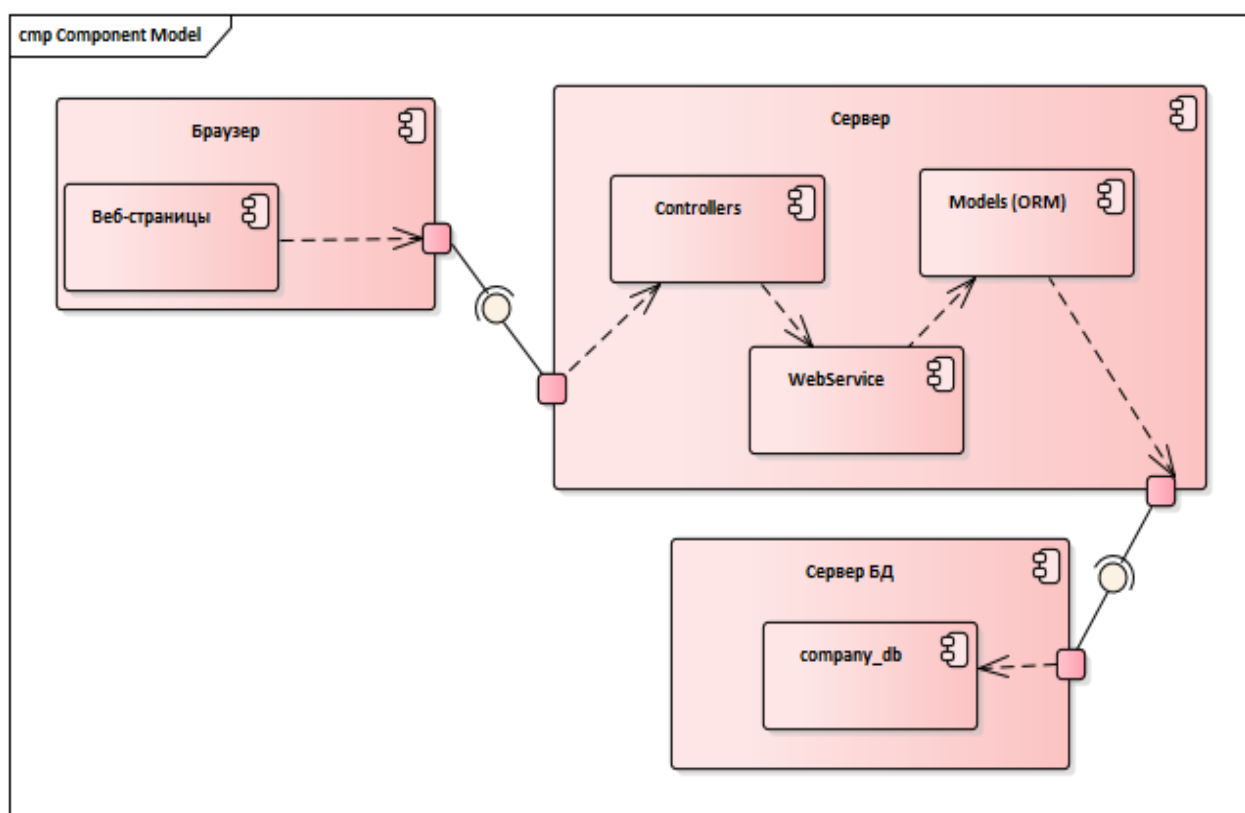


Рисунок А.2 – Диаграмма компонентов

## Продолжение приложения А

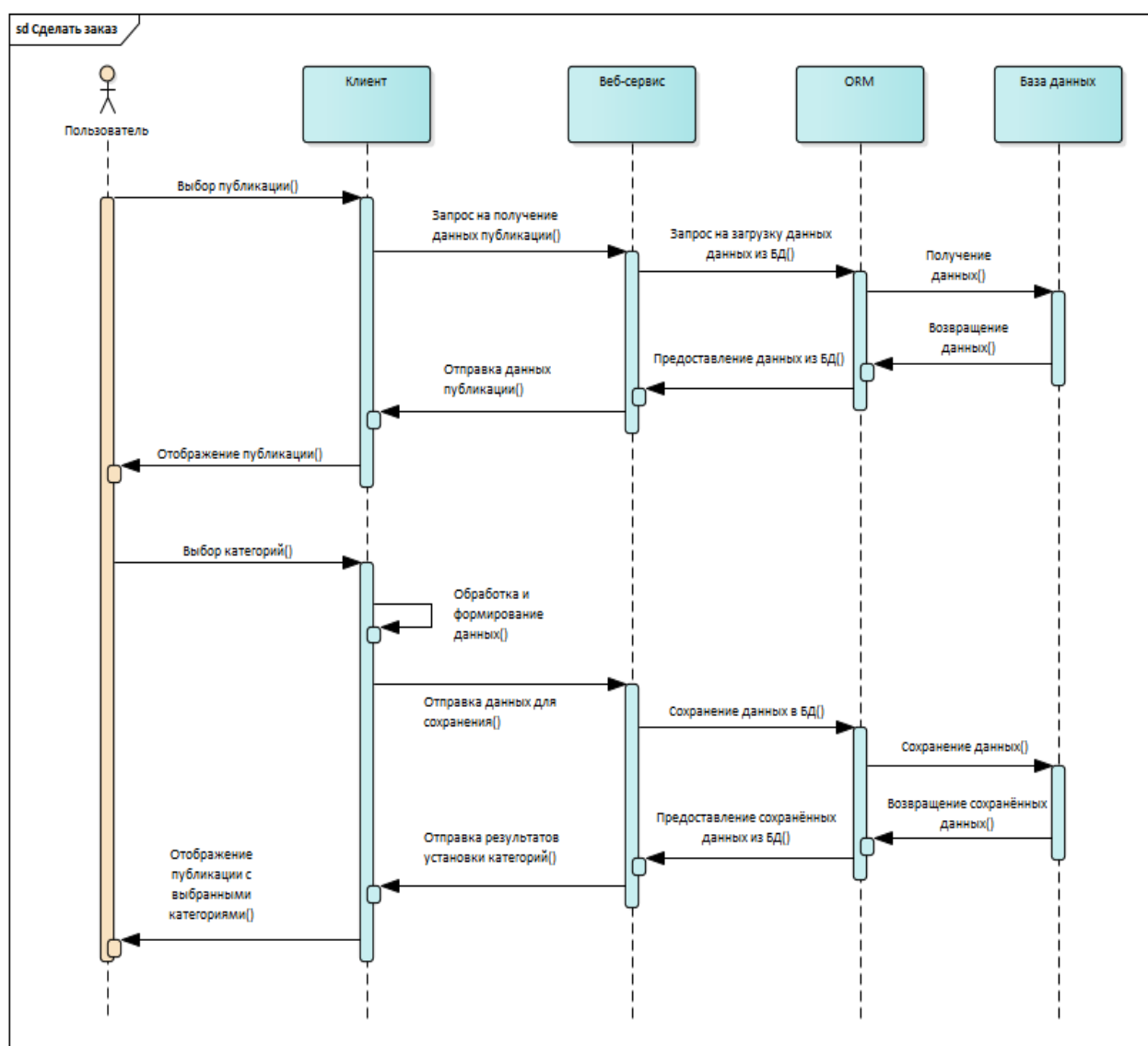


Рисунок А.3 – Диаграмма последовательности процесса добавления категорий

## Продолжение приложения А

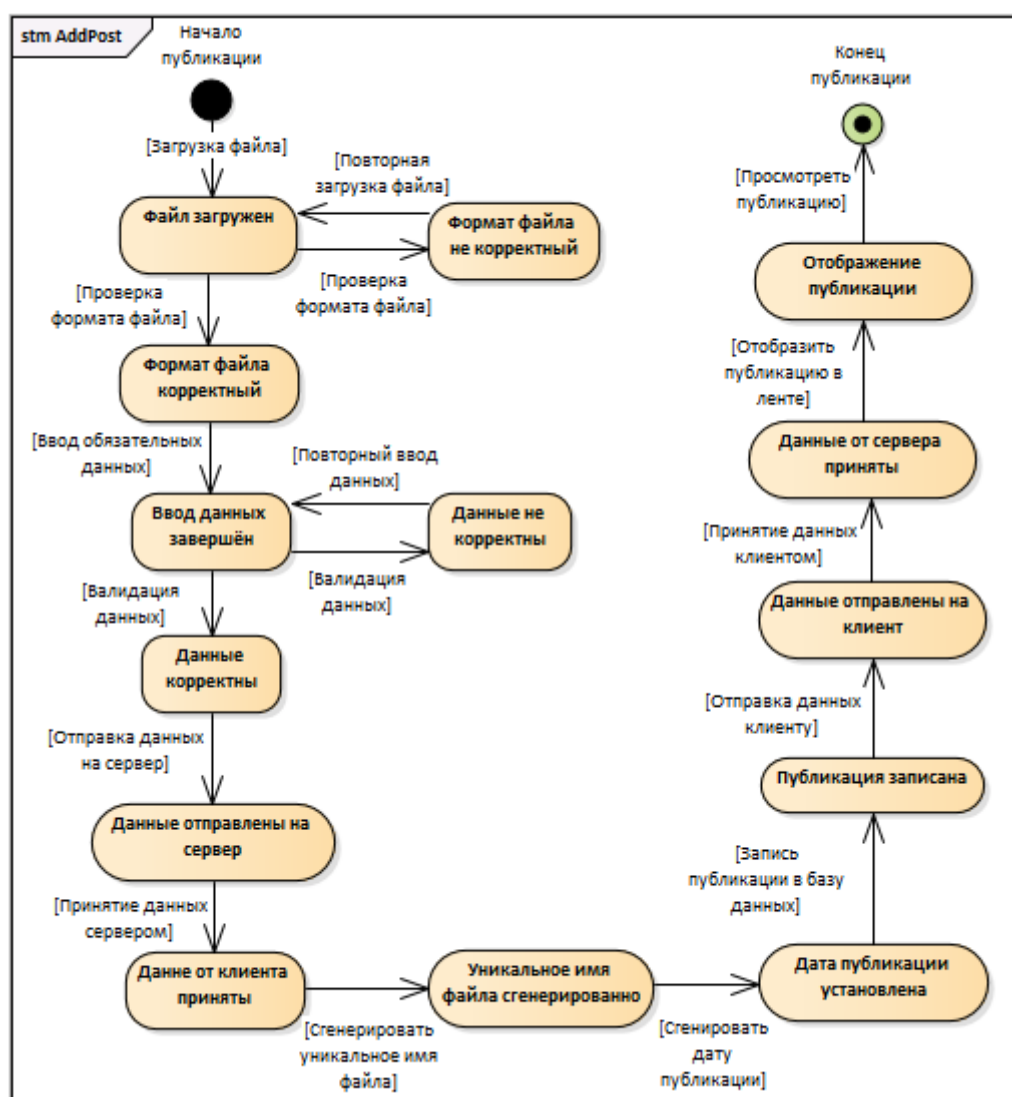


Рисунок А.4 – Диаграмма состояний процесса добавления публикации

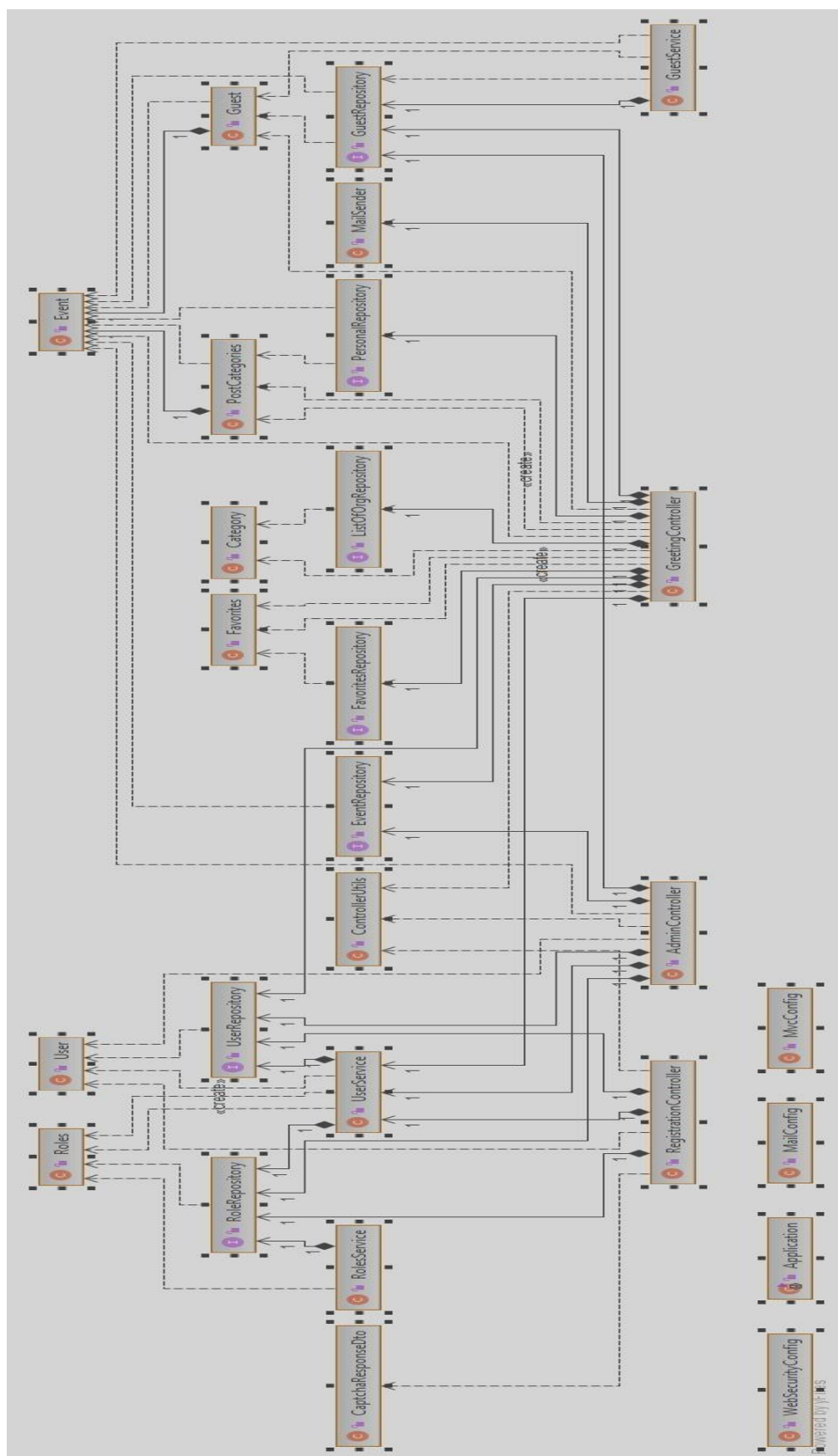


Рисунок А.5 – Диаграмма классов



**ПРИЛОЖЕНИЕ Б**  
**(обязательное)**  
**Блок-схемы алгоритмов**

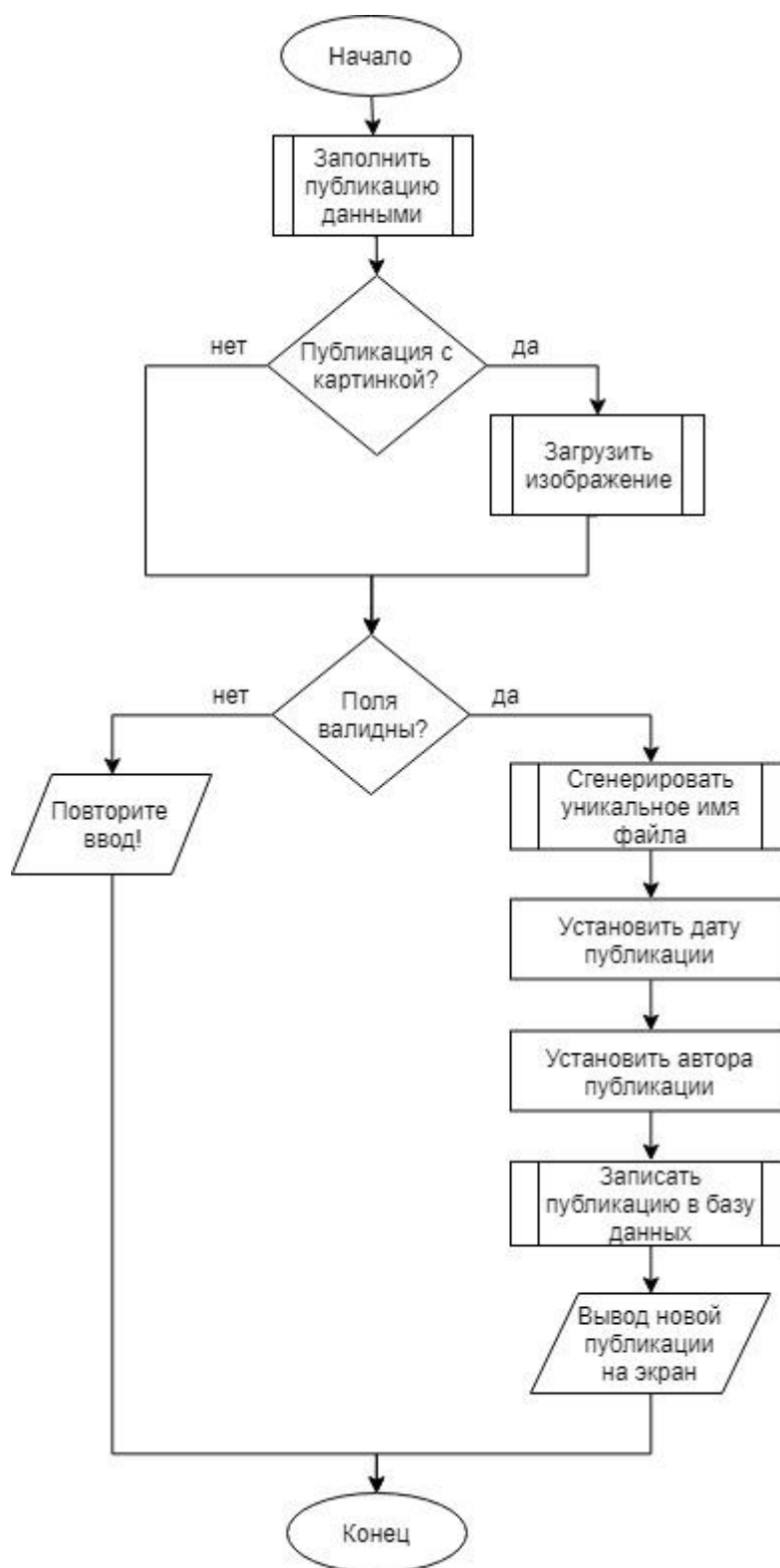


Рисунок Б.1 – Блок-схема алгоритма добавления публикации

## Продолжение приложения Б

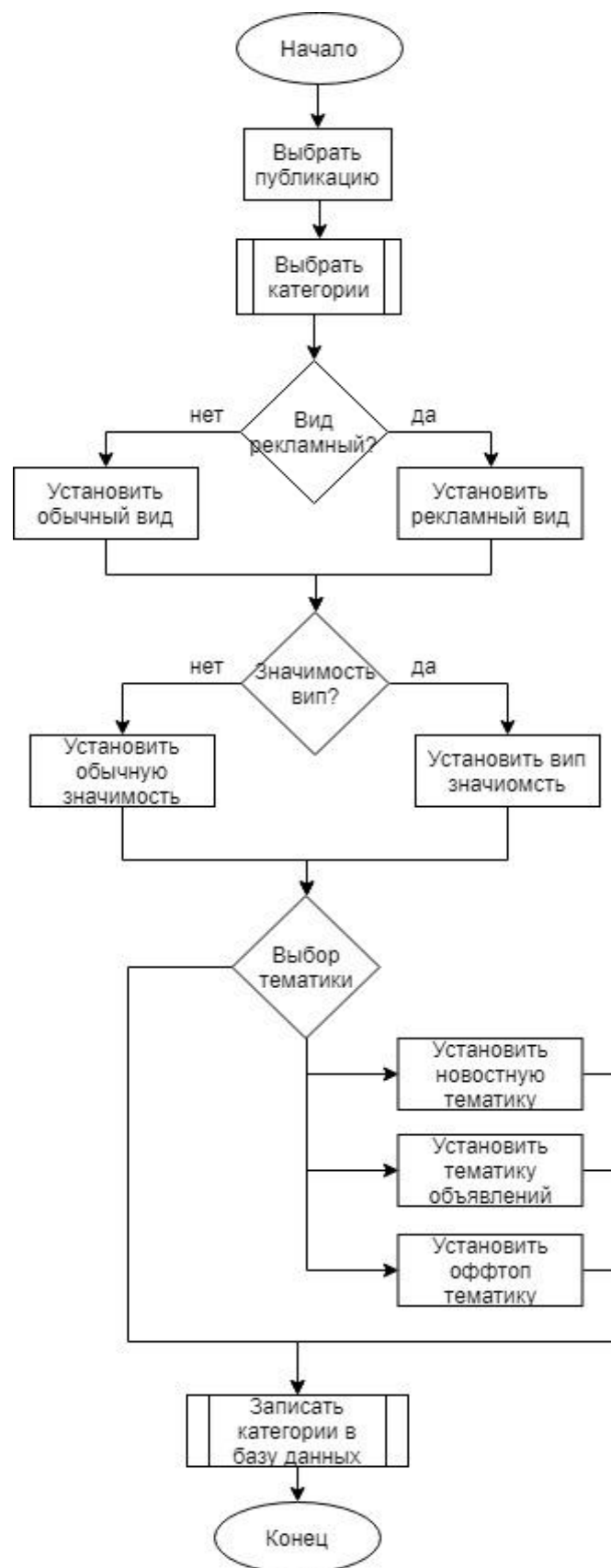


Рисунок Б.2 – Блок-схема алгоритма выбора категорий

**ПРИЛОЖЕНИЕ В**  
**(обязательное)**  
**Листинг SQL-скрипта**

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE
,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----
-- Schema mydb
-- -----
-- -----
-- Schema event
-- -----

-- -----
-- Schema event
-- -----
CREATE SCHEMA IF NOT EXISTS `event` DEFAULT CHARACTER SET cp1251 ;
USE `event` ;

-- -----
-- Table `event`.`category`
-- -----
CREATE TABLE IF NOT EXISTS `event`.`category` (
  `cat_id` BIGINT(20) NOT NULL,
  `kind_of_category` VARCHAR(255) NULL DEFAULT NULL,
  `name_of_category` VARCHAR(255) NULL DEFAULT NULL,
  PRIMARY KEY (`cat_id`))
ENGINE = MyISAM
DEFAULT CHARACTER SET = cp1251;

-- -----
-- Table `event`.`user_roles`
-- -----
CREATE TABLE IF NOT EXISTS `event`.`user_roles` (
  `user_role_id` INT(11) NOT NULL AUTO_INCREMENT,
  `username` VARCHAR(45) NOT NULL,
  `role` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`user_role_id`),
  UNIQUE INDEX `uni_username_role` (`role` ASC, `username` ASC) VISIBLE,
  INDEX `fk_username_idx` (`username` ASC) VISIBLE)
ENGINE = MyISAM
AUTO_INCREMENT = 148
DEFAULT CHARACTER SET = cp1251;

-- -----
```

## Продолжение приложения В

-- Table `event`.`users`

```
-----  
CREATE TABLE IF NOT EXISTS `event`.`users` (  
  `username` VARCHAR(45) NOT NULL,  
  `password` VARCHAR(255) NOT NULL,  
  `enabled` TINYINT(4) NOT NULL DEFAULT '1',  
  `id` BIGINT(20) NOT NULL,  
  `user_roles_user_role_id` INT(11) NOT NULL,  
  PRIMARY KEY (`id`, `user_roles_user_role_id`),  
  INDEX `fk_users_user_roles1_idx` (`user_roles_user_role_id` ASC) VISIBLE)  
ENGINE = MyISAM  
DEFAULT CHARACTER SET = cp1251;
```

-- Table `event`.`event`

```
-----  
CREATE TABLE IF NOT EXISTS `event`.`event` (  
  `id_event` BIGINT(20) NOT NULL,  
  `date` VARCHAR(255) NULL DEFAULT NULL,  
  `filename` VARCHAR(255) NULL DEFAULT NULL,  
  `name` VARCHAR(255) NULL DEFAULT NULL,  
  `number_of_guests` INT(11) NOT NULL,  
  `hashtag` VARCHAR(255) NULL DEFAULT NULL,  
  `defination` VARCHAR(255) NOT NULL,  
  `author` VARCHAR(255) NULL DEFAULT NULL,  
  `users_id` BIGINT(20) NOT NULL,  
  `users_user_roles_user_role_id` INT(11) NOT NULL,  
  PRIMARY KEY (`id_event`, `users_id`, `users_user_roles_user_role_id`),  
  INDEX `fk_event_users1_idx` (`users_id` ASC, `users_user_roles_user_role_id`  
ASC) VISIBLE)  
ENGINE = MyISAM  
DEFAULT CHARACTER SET = cp1251;
```

-- Table `event`.`favorites`

```
-----  
CREATE TABLE IF NOT EXISTS `event`.`favorites` (  
  `id_favorites` BIGINT(20) NOT NULL,  
  `event` VARCHAR(255) NULL DEFAULT NULL,  
  `user` VARCHAR(255) NULL DEFAULT NULL,  
  `users_id` BIGINT(20) NOT NULL,  
  `users_user_roles_user_role_id` INT(11) NOT NULL,  
  `event_idevent` BIGINT(20) NOT NULL,  
  PRIMARY KEY (`id_favorites`, `users_id`, `users_user_roles_user_role_id`,  
  `event_idevent`),  
  INDEX `fk_favorites_users1_idx` (`users_id` ASC,  
  `users_user_roles_user_role_id` ASC) VISIBLE,  
  INDEX `fk_favorites_event1_idx` (`event_idevent` ASC) VISIBLE)  
ENGINE = MyISAM  
DEFAULT CHARACTER SET = cp1251;
```

## Продолжение приложения В

```
-- -----
-- Table `event`.`hibernate_sequence`
-- -----
CREATE TABLE IF NOT EXISTS `event`.`hibernate_sequence` (
  `next_val` BIGINT(20) NULL DEFAULT NULL)
ENGINE = MyISAM
DEFAULT CHARACTER SET = cp1251;

-- -----
-- Table `event`.`postcategories`
-- -----
CREATE TABLE IF NOT EXISTS `event`.`postcategories` (
  `category_id` BIGINT(20) NOT NULL,
  `event` BIGINT(20) NULL DEFAULT NULL,
  `category_cat_id` BIGINT(20) NOT NULL,
  `event_idevent` BIGINT(20) NOT NULL,
  PRIMARY KEY (`category_id`, `category_cat_id`, `event_idevent`),
  INDEX `FKqtmjkd53m7kku0up2b9484gal` (`event` ASC) VISIBLE,
  INDEX `fk_postcategories_category1_idx` (`category_cat_id` ASC) VISIBLE,
  INDEX `fk_postcategories_event1_idx` (`event_idevent` ASC) VISIBLE)
ENGINE = MyISAM
DEFAULT CHARACTER SET = cp1251;

-- -----
-- Table `event`.`spring_session`
-- -----
CREATE TABLE IF NOT EXISTS `event`.`spring_session` (
  `PRIMARY_ID` CHAR(36) NOT NULL,
  `SESSION_ID` CHAR(36) NOT NULL,
  `CREATION_TIME` BIGINT(20) NOT NULL,
  `LAST_ACCESS_TIME` BIGINT(20) NOT NULL,
  `MAX_INACTIVE_INTERVAL` INT(11) NOT NULL,
  `EXPIRY_TIME` BIGINT(20) NOT NULL,
  `PRINCIPAL_NAME` VARCHAR(100) NULL DEFAULT NULL,
  PRIMARY KEY (`PRIMARY_ID`),
  UNIQUE INDEX `SPRING_SESSION_IX1` (`SESSION_ID` ASC) VISIBLE,
  INDEX `SPRING_SESSION_IX2` (`EXPIRY_TIME` ASC) VISIBLE,
  INDEX `SPRING_SESSION_IX3` (`PRINCIPAL_NAME` ASC) VISIBLE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = cp1251;

-- -----
-- Table `event`.`spring_session_attributes`
-- -----
CREATE TABLE IF NOT EXISTS `event`.`spring_session_attributes` (
  `SESSION_PRIMARY_ID` CHAR(36) NOT NULL,
  `ATTRIBUTE_NAME` VARCHAR(200) NOT NULL,
  `ATTRIBUTE_BYTES` BLOB NOT NULL,
  PRIMARY KEY (`SESSION_PRIMARY_ID`, `ATTRIBUTE_NAME`),
```

## Продолжение приложения В

```
INDEX `SPRING_SESSION_ATTRIBUTES_IX1` (`SESSION_PRIMARY_ID` ASC) VISIBLE,  
CONSTRAINT `SPRING_SESSION_ATTRIBUTES_FK`  
  FOREIGN KEY (`SESSION_PRIMARY_ID`)  
    REFERENCES `event`.`spring_session` (`PRIMARY_ID`)  
    ON DELETE CASCADE)  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = cp1251;  
  
SET SQL_MODE=@OLD_SQL_MODE;  
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;  
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

**ПРИЛОЖЕНИЕ Г**  
**(обязательное)**  
**Листинг исходного кода**

```
package com.example.sweater.Controllers;

import com.example.sweater.Repositories.*;
import com.example.sweater.Services.GuestService;
import com.example.sweater.Services.UserService;
import com.example.sweater.config.MailSender;
import com.example.sweater.domain.*;
import com.sun.org.apache.xpath.internal.operations.Mod;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import
org.springframework.security.web.bind.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import javax.validation.Valid;
import java.io.File;
import java.io.IOException;
import java.security.Principal;
import java.text.SimpleDateFormat;
import java.util.*;

@Controller
public class GreetingController {
    @Value("${upload.path}")
    String uploadPath;

    @GetMapping("/")
    public String greeting(Map<String, Object> model) {
        return "greeting";
    }

    @GetMapping("/main")
    public String main(@RequestParam(required = false, defaultValue = "")
String filter, Model model) {
        boolean isMain = true;
        Iterable<Event> events = eventRepository.findAll();
        if (filter != null && !filter.isEmpty()) {
            events = eventRepository.findByplace(filter);
        } else {
            events = eventRepository.findAll();
        }
        model.addAttribute("isMain", isMain);
        model.addAttribute("events", events);
        model.addAttribute("filter", filter);
        return "main";
    }
}
```

## Продолжение приложения Г

```

    }

    @Autowired
    private UserRepository userRepository;

    @Autowired
    private EventRepository eventRepository;
    @Autowired
    private GuestRepository guestRepository;
    @Autowired
    private MailSender mailSender;
    @Autowired
    private UserService userService;
    @Autowired
    private ListOfOrgRepository listOfOrgRepository;
    @Autowired
    PersonalRepository personalRepository;
    @Autowired
    FavoritesRepository favoritesRepository;

    @GetMapping("/main/{name}")
    public String aboutEvent(@PathVariable String name, Model model, Principal
user) {
        model.addAttribute("categories",
personalRepository.findByevent(eventRepository.findByName(name)));
        Event event = eventRepository.findByName(name);
        int currentNumber = event.getNumberOfguests();
        int newnumber = currentNumber + 1;
        event.setNumberOfguests(newnumber);
        eventRepository.save(event);
        boolean isUser = false;
        boolean isFavorite = false;
        Favorites favorites = favoritesRepository.findByUserAndEvent(user.getName(), name);
        if (favorites == null) {
            isFavorite = true;
        }
        if (event.getUser().equals(user.getName())) isUser = true;
        model.addAttribute("isFavorite", isFavorite);
        model.addAttribute("event", event);
        model.addAttribute("isUser", isUser);
        return "event";
    }

    @PostMapping("/main/{name}")
    public String addGuest(@PathVariable String name, @Valid Guest guest,
BindingResult bindingResult, Model model) {
        if (bindingResult.hasErrors()) {
            Map<String, String> errorsMap =
ControllerUtils.getErrors(bindingResult);
            model.mergeAttributes(errorsMap);
            model.addAttribute("guest", guest);
        }
    }

```



## Продолжение приложения Г

```
        Event event = eventRepository.findByName(name);
        model.addAttribute("event", event);
        return "event";
    } else {
        Event thisevent = eventRepository.findByName(name);
        mailSender.send(guest.getEmail(), "Новый пост: " + name,
"Посмотрите новый пост " + name + " ! От: " + thisevent.getDate());
        model.addAttribute("message", "Сообщение успешно отправлено");
        return "acceptguest";
    }
}

@PostMapping("/main")
public String addNewUser(@Valid Event event, BindingResult bindingResult,
Model model, @RequestParam("file") MultipartFile file, Principal user) throws
IOException {

    if (bindingResult.hasErrors()) {
        Map<String, String> errorsMap =
ControllerUtils.getErrors(bindingResult);
        model.mergeAttributes(errorsMap);
        model.addAttribute("event", event);
    } else {
        if (file != null && !file.getOriginalFilename().isEmpty()) {
            File uploadDir = new File(uploadPath);
            if (!uploadDir.exists()) {
                uploadDir.mkdir();
            }
            String uuidFile = UUID.randomUUID().toString();
            String resultFileName = uuidFile + "." +
file.getOriginalFilename();
            file.transferTo(new File(uploadPath + "/" + resultFileName));
            event.setFilename(resultFileName);
        }
        event.setUser(user.getName());
        Date currentDate = new Date();
        SimpleDateFormat dateFormat = new SimpleDateFormat();
        event.setDate(dateFormat.format(currentDate));
        event.setNumberOfguests(0);
        eventRepository.save(event);
    }
    model.addAttribute("events", eventRepository.findAll());
    boolean isMain = true;
    model.addAttribute("isMain", isMain);
    return "main";
}

@GetMapping("/profile")
public String getProfile(Model model, Principal user) {
    model.addAttribute("username", user.getName());
    return "profile";
}
```

## Продолжение приложения Г

```
}

@PostMapping("/profile")
public String updateProfile(Principal user, @RequestParam String password,
@RequestParam String password2, Model model) {
    if (password.equals(password2)) userService.updateUser(user.getName(),
password);
    else model.addAttribute("message", "Пароли не совпадают");
    return "redirect:/profile";
}

@GetMapping("/addpersonal/{name}")
public String getPersonal(Model model) {
    Iterable<Category> ohrOrg =
listOfOrgRepository.findByKindOfCategory("kind");
    Iterable<Category> reklOrg =
listOfOrgRepository.findByKindOfCategory("theme");
    Iterable<Category> cleanOrg =
listOfOrgRepository.findByKindOfCategory("category");
    model.addAttribute("kind", ohrOrg);
    model.addAttribute("theme", reklOrg);
    model.addAttribute("category", cleanOrg);
    return "addpersonal";
}

@PostMapping("/addpersonal/{name}")
public String setPersonal(Model model, @PathVariable String name,
@RequestParam(defaultValue = "null") String[] kinds, @RequestParam(defaultValue
= "null") String[] themes, @RequestParam(defaultValue = "null") String[]
categories) {
    Event event = eventRepository.findByName(name);

    if (!kinds[0].equals("null")) {
        for (int i = 0; i < kinds.length; i++) {
            PostCategories postCategories = new PostCategories();
            postCategories.setEvent(event);
            postCategories.setKindOfCategory("Виды");
            postCategories.setNameOfCategory(kinds[i]);
            personalRepository.save(postCategories);
        }
    }

    if (!categories[0].equals("null")) {
        for (int i = 0; i < categories.length; i++) {
            PostCategories postCategories = new PostCategories();
            postCategories.setEvent(event);
            postCategories.setKindOfCategory("Темы");
            postCategories.setNameOfCategory(categories[i]);
            personalRepository.save(postCategories);
        }
    }

    if (!themes[0].equals("null")) {
        for (int i = 0; i < themes.length; i++) {
```

## Продолжение приложения Г

```

        PostCategories postCategories = new PostCategories();
        postCategories.setEvent(event);
        postCategories.setKindOfCategory("Категории");
        postCategories.setNameOfCategory(themes[i]);
        personalRepository.save(postCategories);
    }
}
model.addAttribute("message", "Категории успешно добавлены");
return "acceptguest";
}
@GetMapping("/listOfpersonal/{name}")
public String getListOfpersonal(Model model, @PathVariable String name) {
    model.addAttribute("categories",
personalRepository.findByevent(eventRepository.findByName(name)));
    return "listofpersonal";
}
@PostMapping("/listOfpersonal/{name}")
public String getListOfpersonal(Model model, @PathVariable String name,
@RequestParam String[] delOrgs) {
    Event event = eventRepository.findByName(name);
    for (int i = 0; i < delOrgs.length; i++) {

        List<PostCategories>                postCategories                =
personalRepository.findByNameOfCategoryAndEvent(delOrgs[i], event);
        for (PostCategories categories : postCategories
        ) {
            personalRepository.delete(categories);
        }
    }
    model.addAttribute("categories",
personalRepository.findByevent(eventRepository.findByName(name)));
    return "listofpersonal";
}
@GetMapping("/events/update/{name}")
public String updateEvent(@PathVariable String name, Model model) {
    Event event = eventRepository.findByName(name);
    model.addAttribute("event", event);
    return "updateevent";
}
@PostMapping("/events/update/{name}")
public String addNewUser(@Valid Event event, @PathVariable String name,
BindingResult bindingResult, Model model, @RequestParam("file") MultipartFile
file) throws IOException {
    Event thisevent = eventRepository.findByName(name);
    if (bindingResult.hasErrors()) {
        Map<String,                String>                errorsMap                =
ControllerUtils.getErrors(bindingResult);
        model.mergeAttributes(errorsMap);
    } else {
        if (file != null && !file.getOriginalFilename().isEmpty()) {
            File uploadDir = new File(uploadPath);
            if (!uploadDir.exists()) {

```

## Продолжение приложения Г

```
        uploadDir.mkdir();
    }
    String uuidFile = UUID.randomUUID().toString();
    String resultFileName = uuidFile + "." +
file.getOriginalFilename();
    file.transferTo(new File(uploadPath + "/" + resultFileName));
    thisevent.setFilename(resultFileName);
}

thisevent.setPlace(event.getPlace());
thisevent.setRequiredage(event.getRequiredage());
thisevent.setName(event.getName());
eventRepository.save(thisevent);
}
model.addAttribute("events", eventRepository.findAll());
return "redirect:/main";
}

@GetMapping("/favorites")
public String favorites(@RequestParam(required = false, defaultValue = "")
String filter, Model model, Principal user) {
    boolean isMain = false;
    Iterable<Favorites> favorites =
favoritesRepository.findByUser(user.getName());
    List<Event> events = new ArrayList<>();
    for (Favorites favorite : favorites) {
        Event event = eventRepository.findByName(favorite.getEvent());
        events.add(event);
    }
    model.addAttribute("events", events);
    model.addAttribute("filter", filter);
    model.addAttribute("isMain", isMain);
    return "main";
}

@PostMapping("/favorites/add/{name}")
public String addFavorite(@PathVariable String name, Principal user) {
    Favorites favorites = new Favorites();
    favorites.setEvent(name);
    favorites.setUser(user.getName());
    favoritesRepository.save(favorites);
    return "redirect:/main";
}

@PostMapping("/favorites/delete/{name}")
public String deleteFavorites(@PathVariable String name, Principal user) {
    Favorites favorites =
favoritesRepository.findByUserAndAndEvent(user.getName(), name);
    favoritesRepository.delete(favorites);
    return "redirect:/main";
}
}
```