**Реферат**

Утверд.

Н. контр.

Провер.

Разраб.

*Смелов В.В.*

*Копыток Д.В.*

*Голуб Е.В.*

*Кантарович В.С.*

Ф.И.О.

Подпись

Дата

1

*Реферат*

Лист

1

*БГТУ 00.00.ПЗ*

Лит.

Листов

*БГТУ 74218007, 2021*

Пояснительная записка дипломного проекта содержит 70 страниц пояснительной записки, 7 таблиц, 13 формул, 30 иллюстраций, 16 источников литературы, 9 приложений.

Веб-приложение, DJANGO, PYTHON, javascript, SQLITE, AJAX, jquery.

Целью дипломного проекта является разработка веб-приложения для создания записей и общения в сети в реальном времени.

В первой главе проводится аналитический обзор литературы по тематике дипломного проекта.

Вторая глава посвящена обзору средств разработки и содержит описание технологий, использованных во время выполнения проекта.

В третьей главе описывается процесс разработки, принципы функционирования и назначение созданных компонент проекта.

В четвертой главе описывается контрольный пример, с проведением тестирования, и показывается поведения системы при разных внештатных ситуациях.

В пятой главе описано руководство пользователя, позволяющее подробно понять интерфейс программного средства.

В шестой главе приводится расчет экономических параметров и себестоимости программного средства.

Объем графической части дипломного проекта составляет 1.75 листа А1.

**Abstract**

Утверд.

Н. контр.

Провер.

Разраб.

*Смелов В.В.*

*Копыток Д.В.*

*Голуб Е.В.*

*Кантарович В.С.*

Ф.И.О.

Подпись

Дата

1

*Abstract*

Лист

1

*БГТУ 00.00.ПЗ*

Лит.

Листов

*БГТУ 74218007, 2021*

Explanatory note of the diploma project 70 pages of explanatory note, 7 tables, 13 formulas, 30 illustrations, 16 sources of literature, 9 appendices.

WEB APPLICATION, DJANGO, PYTHON, javascript, SQLITE, AJAX, jquery.

The aim of the diploma project is to develop a web application for creating records and communicating on the network in real time.

The first chapter provides an analytical review of the literature on the subject of the diploma project.

The second chapter is devoted to a review of development tools and describes the technologies used during the project.

The third chapter describes the development process, the principles of functioning and the purpose of the created project components.

The fourth chapter describes a test case, with testing, as well as a demonstration of the behavior of the system in various emergency situations.

The fifth chapter describes a user manual that allows you to understand in detail the interface of the software tool.

The sixth chapter provides the calculation of economic parameters and the cost of software developed in the framework of the graduation project.

The volume of the graphic part of the diploma project is 1.75 sheets A1.

**Содержание**

Утверд.

Н. контр.

Провер.

Разраб.

*Смелов В.В.*

*Копыток Д.В.*

*Голуб Е.В.*

*Кантарович В.С.*

Ф.И.О.

Подпись

Дата

2

*Содержание*

Лист

1

*БГТУ 00.00.ПЗ*

Лит.

Листов

*БГТУ 74218007, 2021*

[Введение 7](#_Toc74306285)

[1 Обзор аналогов и постановка задачи 8](#_Toc74306286)

[1.1 Постановка задачи 8](#_Toc74306287)

[1.2 Актуальность задачи 8](#_Toc74306288)

[1.3 Обзор аналогов 9](#_Toc74306289)

[1.4 Патентный поиск 13](#_Toc74306290)

[1.5 Аналитический обзор литературы 14](#_Toc74306293)

[1.5.1 Web-приложение 14](#_Toc74306294)

[1.6 Выводы по разделу 15](#_Toc74306295)

[2 Проектирование программного средства 16](#_Toc74306296)

[2.1 Диаграмма вариантов использования 16](#_Toc74306297)

[2.2 Обоснование выбора средств реализации 17](#_Toc74306298)

[2.2.1 Основной язык программирования 17](#_Toc74306299)

[2.2.2 Фреймворк 18](#_Toc74306300)

[2.2.3 Система управления базами данных 19](#_Toc74306301)

[2.2.4 Вспомогательные инструменты 19](#_Toc74306302)

[2.3 Проектирование базы данных 20](#_Toc74306303)

[2.3.1 Таблица chatmessage 21](#_Toc74306304)

[2.3.2 Таблица chat 21](#_Toc74306305)

[2.3.3 Таблица userrecord 21](#_Toc74306306)

[2.3.4 Таблица commentrecord 22](#_Toc74306307)

[2.3.5 Таблица likerecord 22](#_Toc74306308)

[2.3.6 Таблица repostrecord 22](#_Toc74306309)

[2.3.7 Таблица myuser 23](#_Toc74306310)

[2.3.8 Таблица myfriends 23](#_Toc74306311)

[2.3.9 Таблица django\_admin\_log 23](#_Toc74306312)

[2.3.10 Таблица django\_content\_type 24](#_Toc74306313)

[2.3.11 Таблица auth\_permission 24](#_Toc74306314)

[2.4 Проектирование основных алгоритмов 24](#_Toc74306315)

[2.4.1 Алгоритм отображения страницы настроек пользователя 25](#_Toc74306316)

[2.4.2 Алгоритм добавления записи 26](#_Toc74306317)

[2.4.3 Алгоритм создания нового сообщения 27](#_Toc74306318)

[2.5 Выводы по разделу 28](#_Toc74306319)

[3 Реализация программного средства 29](#_Toc74306320)

[3.1 Установка Django 29](#_Toc74306321)

[3.2 Файловая структура проекта 30](#_Toc74306322)

[3.3 Разработка моделей 31](#_Toc74306323)

[3.4 Разработка представлений 33](#_Toc74306324)

[3.5 Разработка шаблонов 34](#_Toc74306325)

[3.6 Реализация алгоритмов 34](#_Toc74306326)

[3.6.1 Реализация добавления новой записи 34](#_Toc74306327)

[3.6.2 Реализация отображения страницы настроек пользователя 35](#_Toc74306328)

[3.6.3 Реализация создания нового сообщения 36](#_Toc74306329)

[3.7 Выводы по разделу 36](#_Toc74306330)

[4 Тестирование программного средства 37](#_Toc74306331)

[4.1 Тестирование регистрации пользователя 37](#_Toc74306332)

[4.2 Тестирование создания записи 38](#_Toc74306333)

[4.3 Тестирование обновления аватара 39](#_Toc74306334)

[4.4 Тестирование получения данных без авторизации 39](#_Toc74306335)

[4.5 Выводы по разделу 40](#_Toc74306336)

[5 Руководство пользователя 41](#_Toc74306337)

[5.1 Руководство для администратора 41](#_Toc74306338)

[5.2 Руководство для пользователя 43](#_Toc74306339)

[5.3 Выводы по разделу 47](#_Toc74306340)

[6 Технико-экономическое обоснование проекта 48](#_Toc74306341)

[6.1 Общая характеристика разрабатываемого программного средства 48](#_Toc74306342)

[6.2 Исходные данные и маркетинговый анализ 49](#_Toc74306343)

[6.3 Методика обоснования цены 50](#_Toc74306344)

[6.3.1 Объем программного средства 50](#_Toc74306345)

[6.3.2 Основная заработная плата 51](#_Toc74306346)

[6.3.3 Дополнительная заработная плата 52](#_Toc74306347)

[6.3.4 Отчисления в Фонд социальной защиты населения 52](#_Toc74306348)

[6.3.5 Расходы на материалы 53](#_Toc74306349)

[6.3.6 Расходы на оплату машинного времени 53](#_Toc74306350)

[6.3.7 Прочие прямые затраты 53](#_Toc74306351)

[6.3.8 Накладные расходы 54](#_Toc74306352)

[6.3.9 Сумма расходов на разработку программного средства 54](#_Toc74306353)

[6.3.10 Расходы на сопровождение и адаптацию 54](#_Toc74306354)

[6.3.11 Полная себестоимость 55](#_Toc74306355)

[6.4 Выводы по разделу 55](#_Toc74306356)

[Заключение 57](#_Toc74306357)

[Список использованной литературы 58](#_Toc74306358)

[ПРИЛОЖЕНИЕ А Диаграмма вариантов использования 60](#_Toc74306359)

[ПРИЛОЖЕНИЕ Б Логическая схема базы данных 61](#_Toc74306360)

[ПРИЛОЖЕНИЕ В Блок-схема алгоритма отображения настроек пользователя 62](#_Toc74306361)

[ПРИЛОЖЕНИЕ Г Блок-схема алгоритма добавления записи 63](#_Toc74306362)

[ПРИЛОЖЕНИЕ Д Блок-схема алгоритма создания нового сообщения 64](#_Toc74306363)

[ПРИЛОЖЕНИЕ Е Исходный код представления приложения my\_social\_net 65](#_Toc74306364)

[ПРИЛОЖЕНИЕ Ж Исходный код шаблона base.html 68](#_Toc74306365)

[ПРИЛОЖЕНИЕ З Исходный код функции my\_settings 71](#_Toc74306366)

[ПРИЛОЖЕНИЕ И Исходный код функции new\_message 72](#_Toc74306367)

# Введение

Утверд.

Н. контр.

Провер.

Разраб.

*Смелов В.В.*

*Копыток Д.В.*

*Голуб Е.В.*

*Кантарович В. С.*

Ф.И.О.

Подпись

Дата

1

*Введение*

Лист

1

*БГТУ 00.00.ПЗ*

Лит.

Листов

*БГТУ 74218007, 2021*

Современное общество – это общество высоких технологий. Интернет охватил весь мир и все сферы жизнедеятельности людей. Интернет-магазины дают возможность совершать покупки, не выходя из дома, онлайн трансляции позволяют просматривать телепередачи, новостные программы. Обмен информацией, поиск работы, общение между людьми, отдых переходят сегодня в виртуальный мир.

В информационном обществе огромную роль и значение на сегодняшний день играют системы распространения. Существуют межрегиональные и международные системы связи, позволяющие обмениваться информацией на больших расстояниях и территориях. Чисто теоретически любой человек или же любая фирма являются потребителями информации. И это качается как коллективов людей, так и отдельных личностей. Социальную сеть тоже можно назвать системой распространения. Невозможно жить без получения новой информации.

Также, социальные сети, безусловно, упрощают нам жизнь, экономя время, поддерживают наше моральное состояние, позволяя общаться с интересующими нас людьми, коротают досуг. Они, безусловно, будут продолжать свое развитие. Прогресс не стоит на месте. В данной дипломной работе, также, есть попытка создать свою социальную сеть.

Целью данного дипломного проекта является создание веб-приложения «Собеседник», предназначенного для создания пользователем интересных записей, а также для общения в сети в реальном времени.

Также были поставлены такие задачи, как:

* проектирование и разработка структуры базы данных;
* разработка интерфейса пользователя;
* реализация алгоритма отображения страницы настроек пользователя;
* реализация алгоритма добавления записей;
* реализация алгоритма создания нового сообщения;
* реализация обмена сообщениями;
* реализация просмотра, оценки и удаления записей;
* реализация просмотра и добавления комментариев к записи;
* реализация просмотра и удаления беседы;
* реализация просмотра, добавления и удаления друзей;
* реализация поиска друзей;
* реализация создания профиля пользователя.

Веб-приложение разрабатывается с помощью веб-фреймворка Django 3.0 с использованием языка программирования Python 3.7, а также языка программирования JavaScript и технологии AJAX.

# 1 Обзор аналогов и постановка задачи

Утверд.

Н. контр.

Провер.

Разраб.

*Смелов В.В.*

*Копыток Д.В.*

*Голуб Е.В.*

*Кантарович В.С.*

Ф.И.О.

Подпись

Дата

8

*1 Обзор аналогов и постановка задачи*

Лист

1

*БГТУ 01.00.ПЗ*

Лит.

Листов

*БГТУ 74218007, 2021*

В ходе выполнения дипломного проекта был изучен IT-рынок и выявлены аналоги разрабатываемого приложения. В результате чего в этой главе были описаны некоторые существующие социальные сети и были поставлены задачи дипломного проекта.

## 1.1 Постановка задачи

Целью дипломного проекта является разработка веб-приложения «Собеседник» для создания записей и общения в реальном времени. Это означает, что в итоге должен быть законченный программный продукт, представленный веб-приложением. Это обеспечит кроссплатформенность данного приложения. Приложение должно взаимодействовать с централизованной базой данных, в которой будут храниться данные, посредством архитектуры клиент-сервер.

Целью программного средства, реализуемого в дипломном проекте, является улучшить жизнь людей с помощью создания возможности виртуального общения в реальном времени, а также просмотра интересных записей.

Задачи программного средства, реализуемого в дипломном проекте:

* реализовать возможность пользователю редактировать свой профиль;
* реализовать возможности добавления, удаления записей для пользователя;
* реализовать возможности лайков, репостов и комментирования записей других пользователей для авторизованного пользователя;
* реализовать возможности поиска собеседников для пользователя;
* реализовать возможности добавления и удаления друзей для пользователя;
* реализовать возможность обмена сообщениями между пользователями.

## 1.2 Актуальность задачи

Основная функция социальных сетей – обеспечивать поддержание связи между людьми, даже когда они находятся далеко друг от друга. Каждый человек может посредством социальных сетей легко общаться с друзьями и коллегами, а также произвести поиск людей, связь с которыми была прервана, и обзавестись новыми приятными знакомствами. На сегодняшний день нередки случаи, когда в результате знакомства молодых парней и девушек образуются новые семьи. Просмотр фотографий, видеофильмов, прослушивание аудиомузыки.

Количество пользователей социальных сетей превысило отметку в 4,2 млрд человек. Это 53% населения мира. Рост за год составил 490 млн. чел. (+13%).

Пользователи поколения Z говорят, что они с большей вероятностью будут искать бренды в социальных сетях, чем в поисковых системах [1]. Социальные сети снимают многие ограничения и позволяют преодолеть в какой-то мере рамки, наложенные на личность пользователя культурой, самим собой, обстоятельствами. На основании всего этого можно сделать вывод, что социальные сети занимают огромную роль в жизни современных людей, и задача, вынесенная на дипломную работу, актуальна.

## 1.3 Обзор аналогов

Для того чтобы создать необходимое приложение, которое бы соответствовало современным стандартам, выглядело согласно новым тенденциям в области дизайна, а также имело исчерпывающий набор функционала, проведем небольшой обзор аналогов. Будем сравнивать аналоги по следующим критериям: функциональность, надежность, дизайн, удобство использования, мобильные возможности.

В качестве первого аналога была выбрана социальная сеть «Twitter» [2].

Twitter – социальная сеть (микроблог), позволяющая пользователям отправлять короткие текстовые заметки, используя веб-интерфейс, службы мгновенного обмена сообщениями или сторонние программы-клиенты для пользователей интернета любого возраста.

Пользование сервисом бесплатно. Любой пользователь может добавлять и читать сообщения в Twitter, зайдя на сайт. Данную социальную сеть используют также и в бизнес-целях. На рисунке 1.1 представлена страница сообщества.

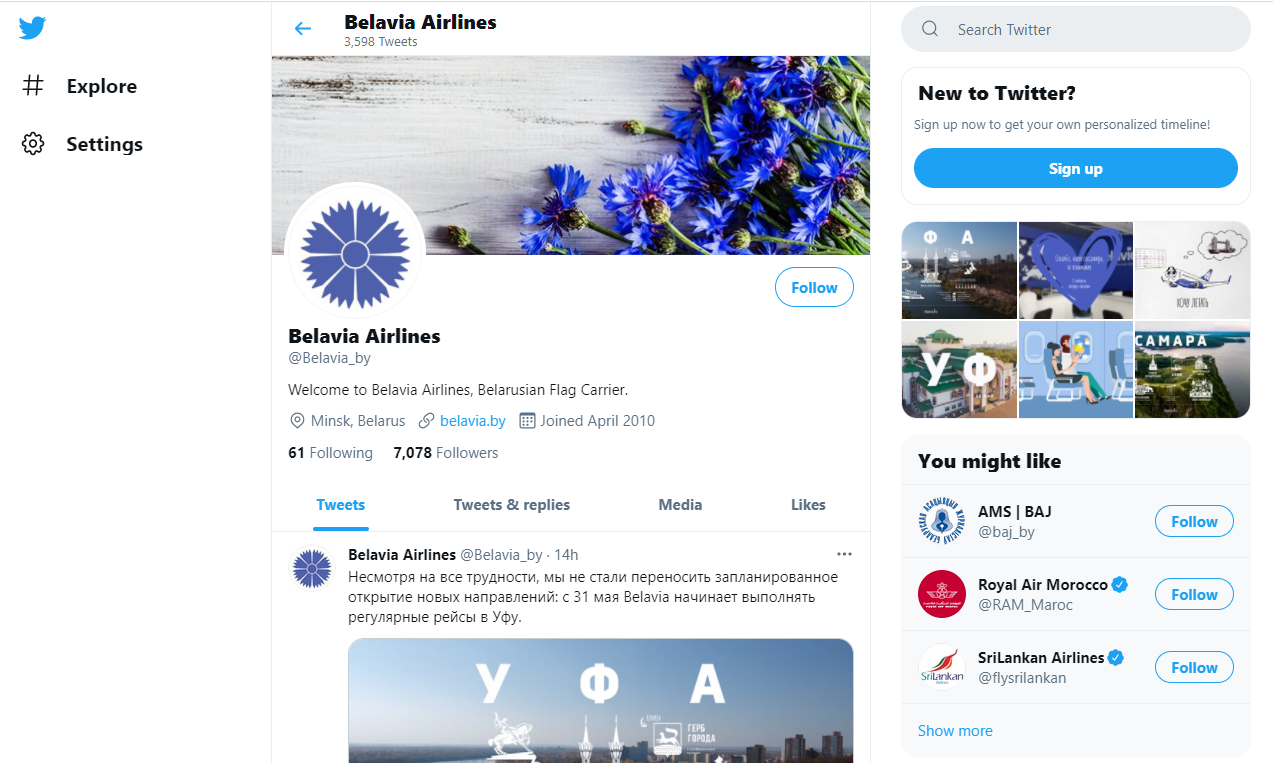


Рисунок 1.1 – Страница сообщества в Twitter

К основному функционалу сайта относится создание заметок, за счет которых люди могут общаться. Эти заметки оцениваются, на них можно ответить, их можно переслать. Также на сайте есть возможность поиска «твиттов» и пользователей. На пользователей можно подписаться и получать их заметки к себе в ленту.

Недостатком является то, что из-за публичных записей и возможности SMS сообщения Twitter является недостаточно надежным. Тем не менее, пользователь по желанию может настроить двойную аутентификацию.

Интерфейс сайта достаточно прост, что позволят быстро и удобно им пользоваться. На странице пользователя можно просмотреть все его заметки и увидеть кто и как их оценивает.

Сайт имеет удобную в использовании мобильную версию, что является преимуществом для пользователя, так как дает возможность создать «твитт» в любом месте, где есть подключение к сети интернет.

В качестве второго аналога была выбрана социальная сеть «ВКонтакте» [3].

ВКонтакте – наиболее известная социальная сеть Рунета. В данной социальной сети можно создать свою личную страницу, написать заметку, добавить аудио или видео, общаться с другими участниками ВКонтакте посредством личных или групповых чатов, подписываться на обновления групп или пользователей, организовывать встречи или группы. Страницу группы можно увидеть на рисунке 1.2.

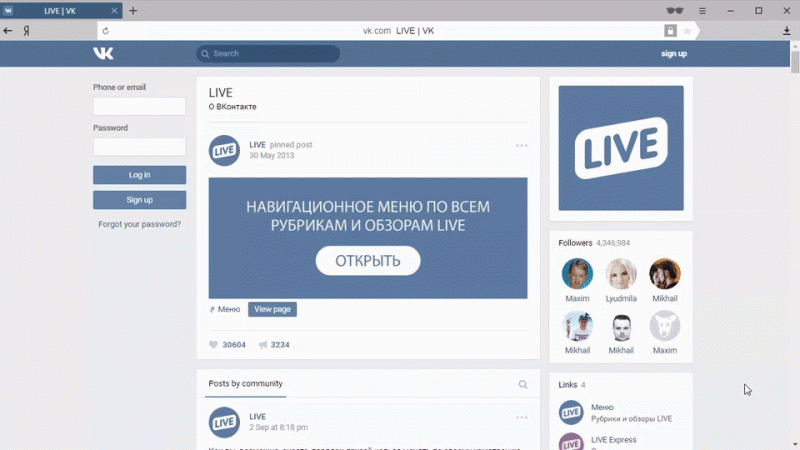


Рисунок 1.2 – Страница группы в ВКонтакте

Пользователь может зарегистрироваться под своим настоящим именем или вымышленным. Регистрация возможна только с 12 лет. Можно настроить двойную аутентификацию, уведомления на электронную почту и еще некоторые параметры защиты, как например, ограничение доступа.

ВКонтакте постоянно обновляет интерфейс программы, что, с одной стороны, облегчает навигацию пользователей по сети, с другой – пользователю часто приходится привыкать к чему-то новому.

Существует мобильная версия приложения, которая имеет огромную популярность. Она ничем не уступает основной версии сайта, в некоторых случаях даже удобнее.

На рисунке 1.3 представлена страница новостей мобильной версии сайта.

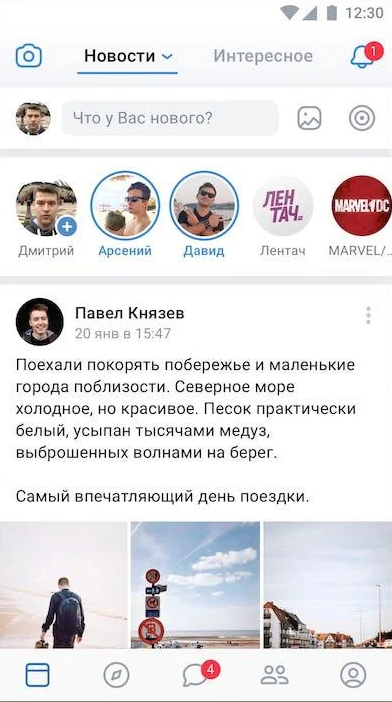


Рисунок 1.3 – Мобильная версия ВКонтакте

В качестве третьего аналога выбрана социальная сеть «Facebook» [4].

Facebook – крупнейшая в мире социальная сеть. Данный сайт имеет огромное количество возможностей и постоянно добавляет новые функции. Базовый функционал включает мгновенную отправку сообщений зарегистрированным пользователям в любую точку мира и создание фотоальбома, добавление и редактирование фотографий и заметок. Вид страницы представлен на рисунке 1.4.



Рисунок 1.4 – Страница ленты новостей Facebook

На главной странице пользователя помимо значков меню расположена новостная лента, основные элементы которой: строка поиска, хроника, последняя активность, рекомендации, чат. Интерфейс социальной сети интуитивно понятен и постоянно дорабатывается в соответствии с интересами пользователей.

Сайт поддерживает шифрование и защиту от шпионажа. И тем не менее, достаточно часто бывают новости о нарушении интеллектуальной собственности, неполном удалении аккаунта, нарушение тайной переписки, нарушение конфиденциальности и т.п.

Facebook имеет мобильную версию, вид которой представлен на рисунке 1.5.

Мобильная версия постоянно развивается и в основном в направлении бизнеса. Стало легче взаимодействовать с бизнес-страницами и появилась возможность оставлять отзывы и рекомендации.

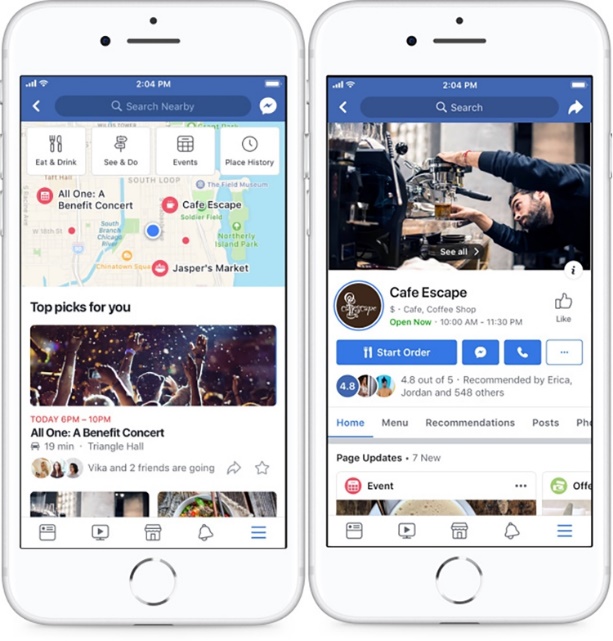


Рисунок 1.5 – Мобильная версия Facebook

После проделанного обзора аналогов, можно выявить все хорошие и плохие стороны рассмотренных сайтов, и отобразить в своем приложении. Хорошие стороны, которые нужно взять в разрабатываемое приложение – это наличие простого интерфейса, профиля, поиска пользователей.

В качестве еще одного аналога выбрана блог-платформа «LiveJournal» [5].

LiveJournal – блог-платформа для ведения онлайн-дневников (блогов), а также отдельный персональный блог, размещенный на этой платформе.

Данное веб-приложение предлагает пользователям стандартный для всех блогов набор функций и настроек, расширенный за счет дополнительных опций.

Среди настроек, функций и опций LiveJournal: разные типы записей и возможности их комментирования, указание расширенных сведений о пользователе, друзья и лента друзей, картинки пользователей, функции безопасности аккаунта.

На рисунке 1.6 представлена страница «Интересное» в LiveJournal.

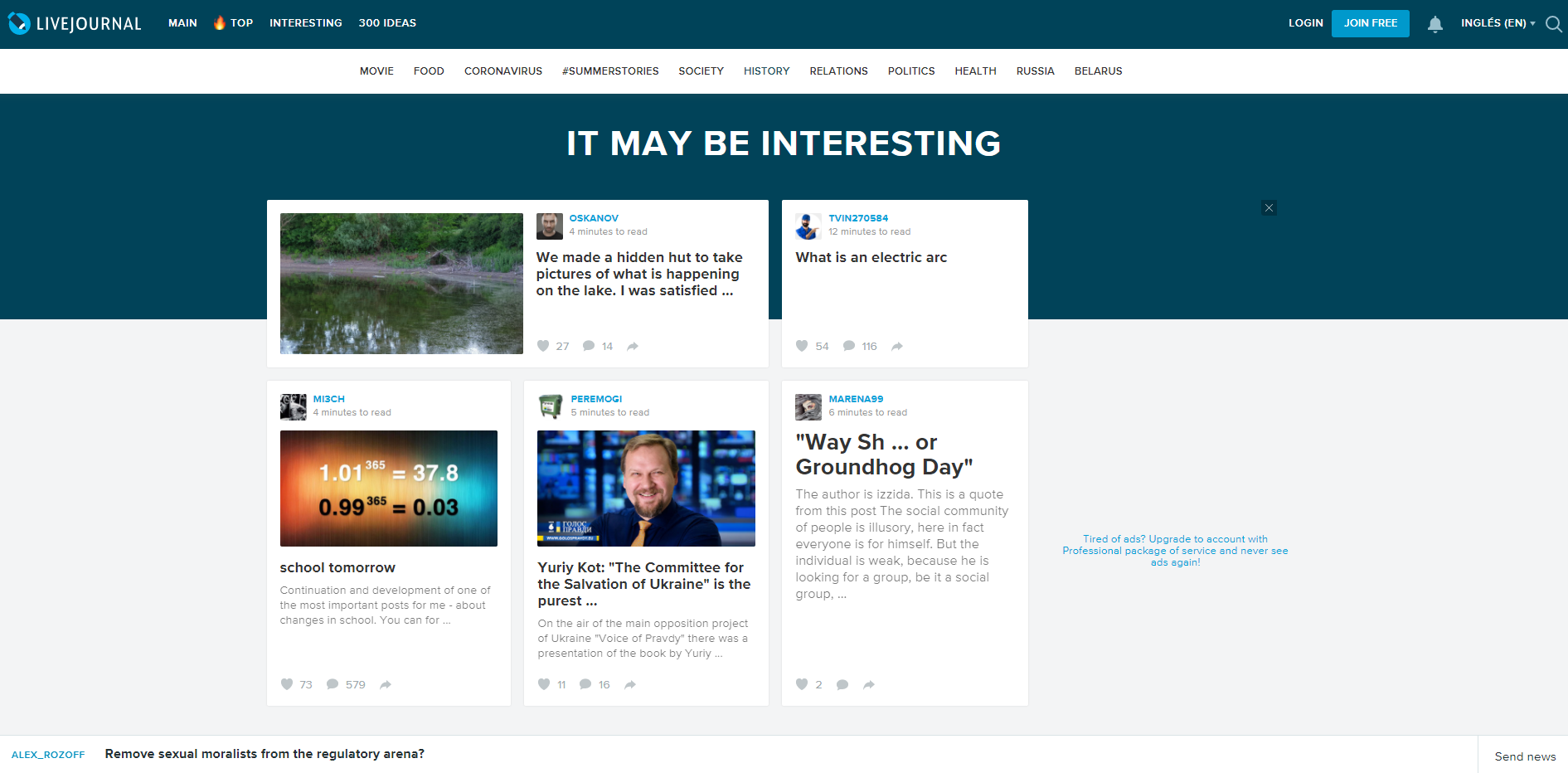


Рисунок 1.6 – Страница подборки «интересного» в LiveJournal

Как уже говорилось, все аккаунты сайта защищены. Несмотря на это была утечка базы данных и приложение довольно часто подвергается DDoS-атакам. Также, иногда бывает перегрузка сервера, из-за чего пользователи временно не могут писать в блог.

Дизайн LiveJournal достаточно прост и имеет множество полезных разделов, по которым легко понять, куда перейти. Из минусов, недостаточная контрастность заголовков. Также, в комментариях слишком большие аватары и огромные пространства для даты и ника. Лента постов выглядит однообразно. Но в целом, сайт интуитивно понятен в использовании, что хорошо.

Отдельного мобильного приложения у LiveJournal не имеется. Но данное веб-приложение оптимизировано для отображения на планшетах и смартфонах.

Таким образом, функционально приложение полностью реализуется, однако интерфейс по-прежнему нужно изменять к лучшему. Также можно добавить автоматическое приведение постов в более приемлемый вид, так как 90% LiveJournal – это работа с контентом.

## 1.4 Патентный поиск

В данном подразделе представлены результаты патентного исследования. В результате проведенного патентного поиска был выявлен ряд патентов, представленных в таблицах 1.1, 1.2.

Таблица 1.1 – Описание патента №1

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Номер патента | Опубликовано | Авторы |
| «Method and apparatus for augmented social networking messaging» | US20150326523A1 | 12.11.2015 | James Francis Reilly, Toni Peter Strandell, Jyri P. Salomaa and other |

В данном патенте предусмотрен подход к расширенным сообщениям в социальных сетях. Сообщение генерируется для размещения в социальной сети. Адресная информация определяется для группы из одного или нескольких устройств, связанных с одним или несколько пользователей. По крайней мере, один из одного или нескольких пользователей не является абонентом социальной сети. Адресная информация используется для отправки сообщения группе. Ответ на сообщение получен с одного из устройств группы. Инициирована публикация ответа в социальной сети [6].

Таблица 1.2 – Описание патента №2

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Номер патента | Опубликовано | Авторы |
| «Chat-enabled social-circles» | US8832190B1 | 09.09.2014 | Matthew Leske, Per J. Gustafsson, Lars Olof Joakim Lindholm, Christopher Vander Mey |

Методы, системы и аппараты, включая компьютерные программы, закодированные на компьютерной среде хранения данных, для чата, что позволяет социальному кругу в ответ на пользовательский ввод, предоставляемый первым пользователем, круг общения, определяемый первым пользователем и включающий один или несколько контактов, один или несколько контактов, включая второго пользователя, для каждого из одного или нескольких контактов, выдача запроса на подписку на чат, определение того, что запрос на подписку, связанный со вторым пользователем, выполнен, и в ответ на определение того, что запрос на подписку, связанный со вторым пользователем, выполняется, добавляя второго пользователя в качестве контакта в чате первого пользователя и добавляя первого пользователя в качестве контакта второго пользователя [7].

## 1.5 Аналитический обзор литературы

### 1.5.1 Web-приложение

Веб-приложение представляет собой веб-сайт, на котором размещены страницы с частично либо полностью несформированным содержимым. Окончательное содержимое формируется только после того, как посетитель сайта запросит страницу с веб-сервера. В связи с тем, что окончательное содержимое страницы зависит от запроса, созданного на основе действий посетителя, такая страница называется динамической.

Веб-приложение состоит из клиентской и серверной частей, тем самым реализуя [технологию «клиент-сервер»](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80). Клиентская часть реализует [пользовательский интерфейс](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81), формирует запросы к серверу и обрабатывает ответы от него. Серверная часть получает запрос от клиента, выполняет вычисления, после этого формирует [веб-страницу](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0) и отправляет ее клиенту с использованием [протокола](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB) [HTTP](https://ru.wikipedia.org/wiki/HTTP).

Само веб-приложение может выступать в качестве клиента других служб, например, [базы данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) или другого веб-приложения, расположенного на другом сервере. Ярким примером веб-приложения является [система управления содержимым](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D1%81%D0%BE%D0%B4%D0%B5%D1%80%D0%B6%D0%B8%D0%BC%D1%8B%D0%BC) статей [Википедии](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F): множество ее участников могут принимать участие в создании сетевой энциклопедии, используя для этого браузеры своих операционных систем (будь то [Microsoft Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows), [GNU/Linux](https://ru.wikipedia.org/wiki/GNU/Linux) или любая другая операционная система) и не загружая дополнительных исполняемых модулей для работы с базой данных статей.

В настоящее время набирает популярность новый подход к разработке веб-приложений, называемый [Ajax](https://ru.wikipedia.org/wiki/Ajax). При использовании [Ajax](https://ru.wikipedia.org/wiki/Ajax) страницы веб-приложения не перезагружаются целиком, а лишь догружают необходимые данные с сервера, что делает их более интерактивными и производительными. Используя данную технологию можно повысить общее ощущение быстродействия в десятки раз.

Также в последнее время набирает большую популярность технология [WebSocket](https://ru.wikipedia.org/wiki/WebSocket), которая не требует постоянных запросов от клиента к серверу, а создает двунаправленное соединение, при котором сервер может отправлять данные клиенту без запроса от последнего. Таким образом появляется возможность динамически управлять контентом в режиме реального времени.

Для создания веб-приложений на стороне сервера используются разнообразные технологии и любые [языки программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), способные осуществлять вывод в стандартную [консоль](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D0%BE%D0%BB%D1%8C) [8].

## 1.6 Выводы по разделу

В данном разделе был описан обзор аналогов по теме дипломного разрабатываемого программного средства.

В ходе обзора четырех аналогов разрабатываемого приложения были отмечены позитивные стороны для дальнейшей разработки. Таким образом была выявлена целесообразность разработки и основной спектр функциональных возможностей, на которые в будущем необходимо сделать упор, для выделения приложения на фоне его конкурентов. Так же, на основании использования приложений-аналогов были выработаны основные критерии для построения дизайна приложения, позволяющего осуществлять наиболее простое и интуитивное использование приложения.

В обзоре литературы были выявлены ключевые моменты разрабатываемого приложения. Это позволит в дальнейшем использовать найденные материалы при разработке программного средства.

# 2 Проектирование программного средства

Утверд.

Н. контр.

Провер.

Разраб.

*Смелов В.В.*

*Копыток Д.В.*

*Голуб Е.В.*

*Кантарович В.С.*

Ф.И.О.

Подпись

Дата

13

*2 Проектирование программного средства*

Лист

1

*БГТУ 02.00.ПЗ*

Лит.

Листов

*БГТУ 74218007, 2021*

Главной задачей данного дипломного проекта является создание веб-приложения для создания записей и общения в сети. Исходя из изученных аналогов, были поставлены задачи по созданию программных компонентов для работы пользователей, а также по созданию интуитивно понятного интерфейса.

В список функциональных задач для пользователей входят:

* вход в аккаунт;
* редактирование профиля;
* поиск пользователей;
* обмен сообщениями;
* просмотр и удаление бесед;
* просмотр, добавление и удаление друзей;
* просмотр, добавление и удаление записей;
* добавление лайков;
* добавление репостов;
* добавление комментариев к записи.

## 2.1 Диаграмма вариантов использования

Диаграмма вариантов использования является исходным концептуальным представлением системы в процессе ее проектирования и разработки.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. В свою очередь вариант использования – это спецификация сервисов или функций, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемых системой при взаимодействии с актером. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий [9].

Диаграмма вариантов использования для дипломного проекта представлена в приложении А. На данной диаграмме есть 2 актера: пользователь и администратор. Пользователь может зарегистрироваться, авторизоваться, редактировать свой профиль. Также он может просматривать, добавлять и удалять записи. Оценивать записи других пользователей посредством лайков, репостов, комментариев. Актер может искать, добавлять и удалять друзей. Пользователь может обмениваться сообщениями с другими в реальном времени.

Администратор может делать всё, что могут делать пользователи с ролями выше. К этому добавляется возможность манипуляции базой данных пользователей. Также администратор выступает в качестве технического специалиста. Занимается расширением базы данных и поддержкой кода.

## 2.2 Обоснование выбора средств реализации

Этот этап является одним из важнейших моментов в разработке проекта, так как именно при выборе средств реализации нужно принимать во внимание десятки факторов. От выбора языка программирования, фреймворка, системы управления базами данных и других вспомогательных инструментов будут варьироваться сложность проекта, сроки его реализации, а также возможности, которые можно реализовать в проекте.

### 2.2.1 Основной язык программирования

Python является объектно-ориентированным языком, позволяющим разрабатывать различные приложения [10]. Данный язык программирования имеет понятный и простой синтаксис. Python обладает динамической типизацией и большим набором пользовательских библиотек, помогающих в разработке приложений.

Также Python является кроссплатформенным языком. Реализовано множество его интерпретаторов на различных операционных системах. Имеется возможность подключения библиотек, написанных на С.

Python с пакетами NumPy, SciPy и MatPlotLib активно используется как универсальная среда для научных расчетов в качестве замены распространенным специализированным коммерческим пакетам, таким как Matlab, предоставляя аналогичную функциональность и более низкий порог вхождения. Также Python подходит для выполнения нестандартных или сложных задач в системах сборки проектов, что обусловлено отсутствием необходимости предварительной компиляции исходных файлов.

Интерпретатор Python может использоваться в качестве мощной командной оболочки и скриптового языка для написания командных файлов ОС. Легкость обращения из Python-скриптов к внешним программам и наличие библиотек, дающих доступ к управлению системой, делают Python удобным инструментом для системного администрирования. Он широко используется для этой цели на платформе Linux: обычно Python поставляется с системой, во многих дистрибутивах инсталляторы и визуальный интерфейс системных утилит написаны именно на Python. Используется он и в администрировании других Unix-систем, в частности, в Solaris и macOS [11].

Python помогает создавать надежный код, который может использоваться в различных сферах. Простота использования и адаптивность облегчают разработку, позволяя эффективно достигать поставленных целей программирования.

### 2.2.2 Фреймворк

Фреймворки – это программные продукты, которые упрощают создание и поддержку технически сложных или нагруженных проектов. Фреймворк, как правило, содержит только базовые программные модули, а все специфичные для проекта компоненты реализуются разработчиком на их основе. Соответственно, тем самым достигается не только высокая скорость разработки, но и большая производительность и надежность решений.

Django считается лучшим веб-фреймворком, написанным на Python [12]. Его удобно использовать для разработки сайтов, работающих с базами данных.

В Django реализован принцип DRY (don’t repeat yourself). За счет этого сокращается время создания сайтов. То есть, при использовании Django не нужно несколько раз переписывать один и тот же код. Фреймворк позволяет создавать сайт из компонентов. Это можно сравнить со строительством крепости с помощью Lego.

Django подходит для разработки высоконагруженных веб-приложений. Это возможно благодаря архитектуре фреймворка, о которой пойдёт речь ниже.

Фреймворк Django написан на языке программирования Python, поэтому его структура соответствует особенностям языка. Создатели реализовали в Django [паттерн MVC](https://ru.hexlet.io/blog/posts/chto-takoe-mvc-rasskazyvaem-prostymi-slovami), и он применяется в текущей версии фреймворка.

Архитектура MVC позволяет разработчику работать с визуальным представлением и бизнес-логикой приложения отдельно. При работе с Django специалисты чаще используют термин MVT – Model-View-Template или модель-представление-шаблон [13]. Компоненты MVT можно использовать независимо друг от друга.

[Документация Django](https://docs.djangoproject.com/en/2.2/topics/db/models/) определяет модель (model) как «источник информации о данных, в которых содержатся ключевые поля и поведение данных». Обычно одна модель указывает на одну таблицу в базе данных. Django поддерживает базы данных PostgreSQL, MySQL, SQLite и Oracle.

Модели содержат информацию о данных, которые представлены атрибутами или полями. Т.к. модель представляет собой простой класс, она ничего не знает о других уровнях Django. Взаимодействие между уровнями происходит через API.

Модель отвечает за бизнес-логику, методы, свойства и другие элементы, связанные с манипуляцией данными. Также модели позволяют разработчикам создавать, читать, обновлять и удалять объекты в базе данных.

Представление (view) решает три задачи: принимает HTTP-запросы, реализует бизнес-логику, определённую методами и свойствами, отправляет HTTP-ответ в ответ на запросы. То есть, представление получает данные от модели и предоставляет шаблонам (templates) доступ к этим данным или предварительно обрабатывает данные и затем предоставляет к ним доступ шаблонам.

В Django реализован мощный движок шаблонов и собственный язык разметки. Шаблоны представляют собой файлы с HTML-кодом, с помощью которого отображаются данные. Содержимое файлов может быть статическим или динамическим. Шаблоны не содержат бизнес-логики, поэтому только отображают данные.

Такая архитектура позволяет Django успешно решать различные задачи.

### 2.2.3 Система управления базами данных

SQLite – компактная встраиваемая реляционная база данных. Исходный код библиотеки передан в общественное достояние. Является чисто реляционной базой данных [14]. Слово «встраиваемый» означает, что SQLite не использует парадигму клиент-сервер. Т.е. движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а предоставляет библиотеку, с которой программа компонуется и движок становится составной частью программы.

В качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа. Простота реализации достигается за счёт того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется; ACID-функции также достигаются за счёт создания файла журнала.

Несколько процессов или потоков могут одновременно без каких-либо проблем читать данные из одной базы. Запись в базу можно осуществить только в том случае, если никаких других запросов в данный момент не обслуживается; в противном случае попытка записи оканчивается неудачей, и в программу возвращается код ошибки. Другим вариантом развития событий является автоматическое повторение попыток записи в течение заданного интервала времени. Можно, также, ввести таймаут операций [15]. Тогда подключение, столкнувшись с занятостью БД, будет ждать N секунду прежде, чем отвалиться с ошибкой SQLITE\_BUSY.

Также с версии 3.7.0 присутствует режим WAL, который дает возможность использовать одну и ту же базу данных несколькими приложениями, как на чтение, так и на запись.

В комплекте поставки идёт также функциональная клиентская часть в виде исполняемого файла **sqlite3**, с помощью которого демонстрируется реализация функций основной библиотеки. Клиентская часть работает из командной строки, позволяет обращаться к файлу БД на основе типовых функций ОС.

Благодаря архитектуре движка можно использовать SQLite как на встраиваемых системах, так и на выделенных машинах с гигабайтными массивами данных.

Формат файла базы данных является кроссплатформенным, что позволяет без проблем использовать одну и ту же базу на нескольких операционных системах. Также присутствует возможность хранения базы в памяти, без её записи на диск. Этот вариант используется по умолчанию для консольной утилиты **sqlite3**, если не указано имя файла.

### 2.2.4 Вспомогательные инструменты

Создание веб-приложение не может обойтись только предложенными в главах выше инструментами. Так же в проекте будут задействованы такие общие инструменты, как HTML5, CSS3, JavaScript. Без них невозможно создать красивое и динамичное веб-приложения.

Для упрощения работы с языком JavaScript будет использоваться библиотека jQuery. jQuery является наиболее используемой JS-библиотекой в мире, упрощающая работу с DOM, обработку событий, анимацию и использование AJAX.

Дополнением будет использование дополнительной библиотеки Axios для отправки AJAX-запросов [16]. Данная библиотека использует у себя самое низкоуровневое API для отправки запросов – XmlHttpRequest. Однако, за счет расширения возможности дополнения содержимого запроса поддержкой асинхронности, данная библиотека является одной из самых популярных. Ее размер довольно мал, нет никаких уязвимостей в использовании, она имеет стабильную поддержку.

## 2.3 Проектирование базы данных

Для разработки веб-приложения в рамках дипломного проекта понадобилась база данных для хранения всей необходимой информации. Структуру базу данных, связи и поля можно увидеть в приложении Б и на рисунке 2.1.

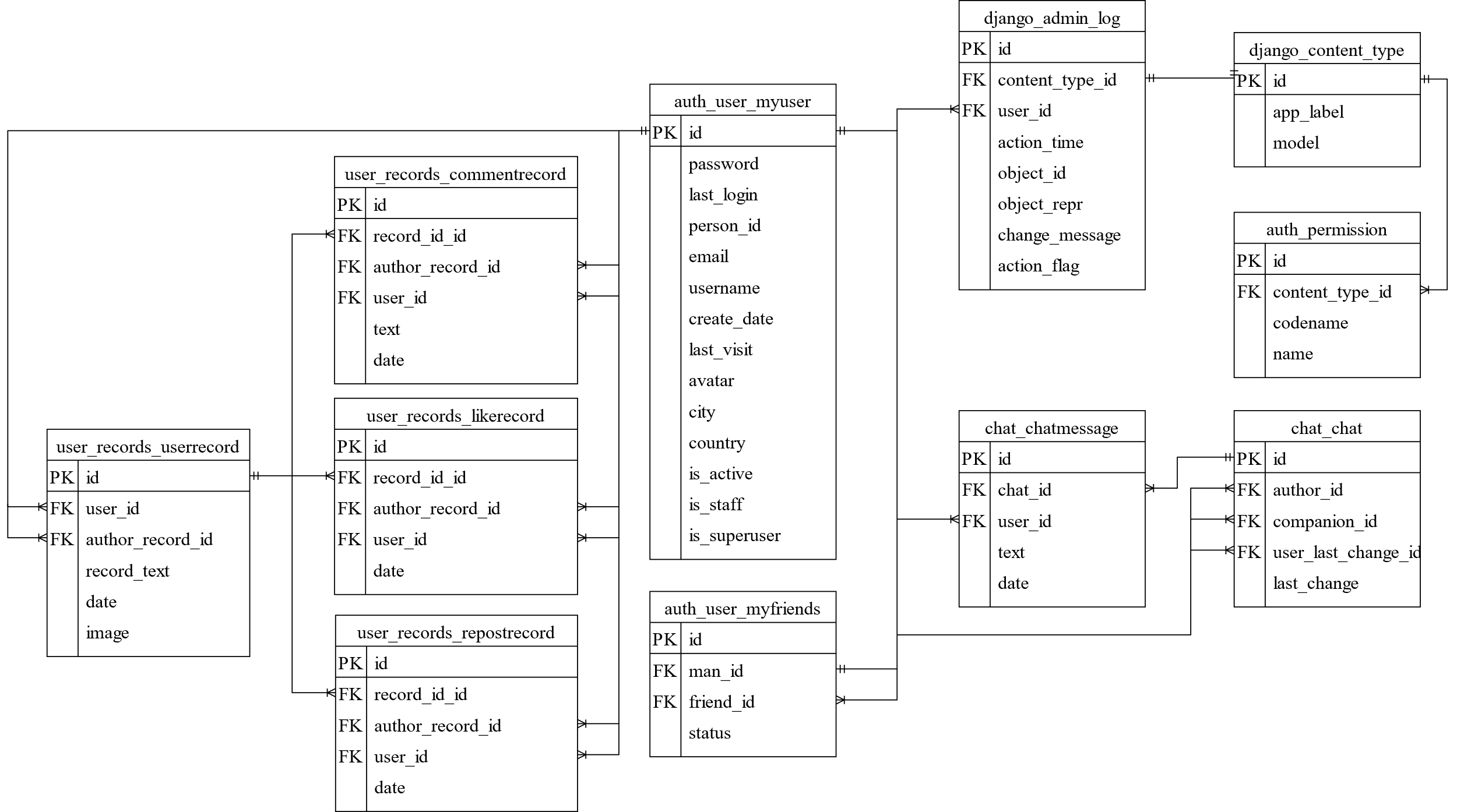


Рисунок 2.1 – Структура базы данных

В базе данных существует одиннадцать таблиц. Схема базы данных включает в себя описание содержания, структуры и ограничений целостности, используемые для создания и поддержки базы данных. Постоянные данные в среде базы данных включают в себя схему и базу данных. Система управления базами данных использует определения данных в схеме для обеспечения доступа и управления доступом к данным в базе данных.

Далее подробно описана структура каждой таблицы.

### 2.3.1 Таблица chatmessage

Таблица, содержащая все сообщения пользователей.

В нее входят следующие столбцы:

* столбец id. Хранит уникальный идентификатор сообщения;
* столбец chat\_id. Хранит идентификатор чата;
* столбец user\_id. Хранит идентификатор пользователя;
* столбец text. Хранит текст сообщения;
* столбец date. Хранит дату отправки сообщения.

Данная таблица содержит ограничение первичного ключа для столбца id. Также столбец id обладает автоматической идентификацией. Для столбца chat\_id существуют ограничения по внешнему ключу к таблице chat. А для столбца user\_id существуют ограничения по внешнему ключу к таблице myuser.

### 2.3.2 Таблица chat

Таблица, содержащая все существующие чаты пользователей.

В нее входят следующие столбцы:

* столбец id. Хранит уникальный идентификатор чата;
* столбец author\_id. Хранит идентификатор первого пользователя чата;
* столбец companion\_id. Хранит идентификатор собеседника;
* столбец user\_last\_change\_id. Хранит идентификатор пользователя, который последним отправил сообщение;
* столбец last\_change. Хранит дату последнего изменения чата.

Данная таблица содержит ограничение первичного ключа для столбца id. Также столбец id обладает автоматической идентификацией. Для author\_id, companion\_id, user\_last\_change\_id существуют ограничения целостности по внешним ключам к таблице myuser.

### 2.3.3 Таблица userrecord

Таблица, содержащая пользовательские записи.

В нее входят следующие столбцы:

* столбец id. Хранит уникальный идентификатор записи;
* столбец user\_id. Хранит идентификатор пользователя;
* столбец author\_record\_id. Хранит идентификатор автора записи;
* столбец record\_text. Хранит текст, относящийся к записи;
* столбец date. Хранит дату создания записи пользователем;
* столбец image. Хранит изображение, относящееся к записи.

Данная таблица содержит ограничение первичного ключа для столбца id. Также столбец id обладает автоматической идентификацией. Для столбцов user\_id и author\_record\_id существуют ограничения по внешним ключам к таблице myuser.

### 2.3.4 Таблица commentrecord

Таблица, содержащая комментарии к записям.

В нее входят следующие столбцы:

* столбец id. Хранит уникальный идентификатор комментария к записи;
* столбец record\_id\_id. Хранит id записи;
* столбец author\_record\_id. Хранит идентификатор автора комментария;
* столбец user\_id. Хранит идентификатор пользователя;
* столбец text. Хранит текст комментария;
* столбец date. Хранит дату добавления комментария к записи.

Данная таблица содержит ограничение первичного ключа для столбца id. Также столбец id обладает автоматической идентификацией. Для столбцов author\_record\_id и user\_id существуют ограничения по внешним ключам к таблице myuser. Также для record\_id\_id существуют ограничения целостности по внешнему ключу к таблице userrecord.

### 2.3.5 Таблица likerecord

Таблица, содержащая лайки к записям.

В нее входят следующие столбцы:

* столбец id. Хранит уникальный идентификатор лайка к записи;
* столбец record\_id\_id. Хранит id записи;
* столбец author\_record\_id. Хранит идентификатор автора лайка;
* столбец user\_id. Хранит идентификатор пользователя;
* столбец date. Хранит дату добавления лайка к записи.

Данная таблица содержит ограничение первичного ключа для столбца id. Также столбец id обладает автоматической идентификацией. Для столбцов author\_record\_id и user\_id существуют ограничения по внешним ключам к таблице myuser. Также для record\_id\_id существуют ограничения целостности по внешнему ключу к таблице userrecord.

### 2.3.6 Таблица repostrecord

Таблица, содержащая репосты записей.

В нее входят следующие столбцы:

* столбец id. Хранит уникальный идентификатор репоста записи;
* столбец record\_id\_id. Хранит id записи;
* столбец author\_record\_id. Хранит идентификатор автора репоста;
* столбец user\_id. Хранит идентификатор пользователя;
* столбец date. Хранит дату создания репоста записи.

Данная таблица содержит ограничение первичного ключа для столбца id. Также столбец id обладает автоматической идентификацией. Для столбцов author\_record\_id и user\_id существуют ограничения по внешним ключам к таблице myuser. Также для record\_id\_id существуют ограничения целостности по внешнему ключу к таблице userrecord.

### 2.3.7 Таблица myuser

Таблица, содержащая конкретную информацию о пользователе.

В нее входят следующие столбцы:

* столбец id. Хранит уникальный идентификатор пользователя в таблице;
* столбец password. Хранит зашифрованный пароль;
* столбец last\_login. Хранит дату открытия сессии;
* столбец email. Хранит электронную почту;
* столбец person\_id. Хранит выданный пользователю идентификатор;
* столбец username. Хранит имя пользователя;
* столбец create\_date. Хранит дату создания аккаунта;
* столбец last\_visit. Хранит дату последнего визита;
* столбец avatar. Хранит аватар пользователя;
* столбец city. Хранит город пользователя;
* столбец country. Хранит страну пользователя;
* столбец is\_active. Хранит статус подтверждения регистрации;
* столбец is\_staff. Хранит статус роли пользователя;
* столбец is\_superuser. Хранит статус о суперпользователе.

Данная таблица содержит ограничение первичного ключа для столбца id. Также столбец id обладает автоматической идентификацией.

### 2.3.8 Таблица myfriends

Таблица, содержащая друзей пользователя.

В нее входят следующие столбцы:

* столбец man\_id. Хранит идентификатор пользователя;
* столбец friend\_id. Хранит идентификатор друга;
* столбец status. Хранит информацию о состоянии заявки в друзья.

Данная таблица содержит ограничение первичного ключа для столбца id. Также столбец id обладает автоматической идентификацией. Для столбцов man\_id и friend\_id существуют ограничения по внешним ключам к таблице myuser.

### 2.3.9 Таблица django\_admin\_log

Таблица, содержащая информацию о событиях веб-приложения.

В нее входят следующие столбцы:

* столбец id. Хранит уникальный идентификатор события;
* столбец content\_type\_id. Хранит идентификатор типа контента;
* столбец user\_id. Хранит идентификатор пользователя;
* столбец action\_time. Хранит дату и время события;
* столбец object\_id. Хранит идентификатор объекта (пользователя);
* столбец object\_repr. Хранит электронную почту объекта (пользователя);
* столбец change\_message. Хранит сообщение об изменении в приложении;
* столбец action\_flag. Хранит флаг события.

Данная таблица содержит ограничение первичного ключа для столбца id. Также столбец id обладает автоматической идентификацией. Для user\_id существуют ограничения по внешнему ключу к таблице myuser. Для content\_type\_id существуют ограничения по внешнему ключу к таблице django\_content\_type.

### 2.3.10 Таблица django\_content\_type

Таблица, содержащая информацию о типе контента. Т.е. она описывает соотношение приложений и относящихся к ним моделей.

В нее входят следующие столбцы:

* столбец id. Хранит версию миграции;
* столбец app\_label. Хранит название приложения проекта;
* столбец model. Хранит название модели, относящейся к приложению.

Данная таблица содержит ограничение первичного ключа для столбца id. Также столбец id обладает автоматической идентификацией.

### 2.3.11 Таблица auth\_permission

Таблица, содержащая информацию о разрешениях.

В нее входят следующие столбцы:

* столбец id. Хранит идентификатор разрешения;
* столбец content\_type\_id. Хранит идентификатор типа контента;
* столбец codename. Хранит кодовое название;
* столбец name. Хранит описание разрешения.

Данная таблица содержит ограничение первичного ключа для столбца id. Также столбец id обладает автоматической идентификацией. Для content\_type\_id существуют ограничения по внешнему ключу к таблице content\_type.

## 2.4 Проектирование основных алгоритмов

В данном подразделе будут описаны основные алгоритмы работы приложения, которые соответствуют целям дипломной работы.

Процесс последовательного построения алгоритма выглядел следующим образом: алгоритм сначала формулировался в «крупные» команды. Затем на каждом последующем этапе отдельные детали алгоритма уточнялись. Процесс продолжался до тех пор, по­ка алгоритм не начинает состоять из команд, которые были бы понятны базовому исполнителю. Данный метод называется методом последовательного уточнения алгоритма. Этот подход к проектированию алгоритмов позволяет повысить качество и надежность разрабатываемых программ.

Блок-схема – это наглядная демонстрация алгоритма, в котором отдельные шаги изображаются в виде блоков различной формы, соединенных между собой линиями, указывающими направление последовательности действий в веб-ресурсе.

Любая блок-схема начинается блоком «Начало» и заканчивается блоком «Конец». Остальные блоки схемы также имеют специальное назначение для определенных действий алгоритма. Квадратные блоки представляют процессы, программу или вложенный алгоритм. Блоки формы параллелограмм обозначают ввод или вывод данных. Блоки в форме ромбов проверяют определенные условия. В зависимости от того, выполняется условие или нет, алгоритм идет по линии «Да» или «Нет».

### 2.4.1 Алгоритм отображения страницы настроек пользователя

В данном пункте будет описан алгоритм работы с данными о пользователе. Алгоритм отвечает за отображение на странице личной информации о пользователе с учетом ее обновления. На рисунке 2.2 и в приложении В показано графическое оформление алгоритма отображения страницы настроек пользователя.



Рисунок 2.2 – Алгоритм отображения страницы настроек пользователя

Когда переходим на страницу настроек пользователя, происходит проверка на его аутентификацию. Если пользователь не аутентифицирован, то система выводит сообщение об ошибке. Иначе, после перехода на выбранную ранее страницу, мы видим личную информацию о нем, отображенную в таблице.

Для начала происходит проверка на наличие файла в первой форме. Если файла нет, то переходим к следующей форме. Если он есть, то определяем, является ли файл изображением. Если не является, то алгоритм заканчивается. Иначе, в результате положительного ответа системы, проверяем форму на валидность. Если полученное значение отрицательно, то переходим к концу алгоритма. Тогда, если форма валидна, обновляем фото и переходим к рендерингу настроек.

Вернемся к моменту проверки на наличие файла. Как уже говорилось, если файл не обнаружен или уже проверен, то переходим к следующей форме. В данной форме описаны текстовые данные о пользователе. Проверяем форму на валидность. Если ответ системы положительный, обновляем данные и отображаем страницу с настройками.

### 2.4.2 Алгоритм добавления записи

На рисунке 2.3 и в приложении Г показано графическое представление алгоритма добавления записи на страницу пользователя.



Рисунок 2.3 – Алгоритм добавления записи

В начале пользователь вводит некоторый текст записи. Далее проверяется, прикреплен ли файл к записи. Если значение условия отрицательно, то добавляем запись и выводим ее на страницу пользователя. Если же значение условия положительно, то определяем является ли файл изображением. Теперь, если файл не изображение, выводится сообщение об ошибке и ожидается замена файла. Иначе происходит проверка на размер изображения, который не должен превышать 2 Мб.

В таком случае, если картинка больше 2 Мб, мы также можем увидеть сообщение об ошибке и заменить ее. Когда все проверки файла завершены, добавляем запись и выводим ее на страницу пользователя.

### 2.4.3 Алгоритм создания нового сообщения

Данный алгоритм отвечает за создание нового сообщения. Графическое представление алгоритма можно увидеть на рисунке 2.4 и приложении Д.

Для начала пользователь набирает сообщение. После чего осуществляется проверка пользователя на аутентификацию. Если проверка отрицательна, то выводится сообщение об ошибке и алгоритм закончен. В ином случае пользователь, набравший сообщение, принимается за автора.

Теперь определяем собеседника с помощью условия. Если другой пользователь является собеседником, то получаем данные о нем по ранее id собеседника. Если нет, то получаем данные по id автора. Текст сообщения, id собеседника и id автора были получены ранее при отправке запроса на сервер.



Рисунок 2.4 – Алгоритм создания нового сообщения

Переходим к следующему действию. Проверяем существует ли уже беседа между этими пользователями. Если полученное значение отрицательно, создаем новую беседу. Если положительно, просто находим существующую. Теперь добавляем новое сообщение в беседу.

## 2.5 Выводы по разделу

В разделе проектирования в первом подразделе была построена и описана диаграмма вариантов использования. Диаграмма вариантов использования является самым общим представлением функциональных требований к системе. Варианты использования являются описаниями типичных взаимодействий между пользователями системы и самой системой.

Были обоснованы: выбор средств реализации проекта, в рамках которого был выбран язык программирования, фреймворк, система управления базами данных, а также вспомогательные инструменты. В целях оптимизации, расширения функциональных возможностей и улучшения качества кода и продукта в целом, были использованы следующие фреймворки, технологии и библиотеки:

* Python 3.7 как основной язык программирования;
* Django 3.0 как базовый фреймворк;
* Система управления базами данных, основанная на SQLite 3.34.0;
* HTML5, CSS3 и JavaScript 3.5.0 для создания динамических страниц;
* Библиотека jQuery 3.6.0 для ускорения разработки с помощью JavaScript.

Язык Python способствует качеству написания кода, отличается широким спектром операционных систем, которые могут быть использованы для разработки, позволяет разрабатывать приложение с высоким уровнем быстродействия, а также отвечает всем современным стандартам качества разработки.

Также в данном разделе было рассмотрено проектирование базы данных. Оно осуществлялось при помощи построения модели данных, были определены таблицы и связи между ними. Раздел содержит в себе информацию с описанием каждой из таблиц в спроектированной базе данных.

Были описаны и проиллюстрированы основные алгоритмы в приложении, необходимые для разработки дипломного проекта.

# 3 Реализация программного средства

Утверд. Утверд.

Н. контр. Н. контр.

Провер. Провер.

Разраб. Разраб.

*Смелов В.В.Смелов В.В.*

*Копыток Д.В.*

*Копыток Д.В.*

*Голуб Е.В.*

*Кантарович В. С.*

*Кантарович В. С*

*Кантарович В. С.*

Ф.И.О.

Попись

Дата

8

*3 Реализация программного средства*

Лист

1

*БГТУ 03.00.ПЗ*

Лит.

Листов

*БГТУ 74218007, 2021*

*74418025, 2020*

Главной задачей данного дипломного проекта является создание веб-приложения для создания записей и общения в сети. Исходя из изучения существующих аналогов, были поставлены задачи по созданию программных компонентов для работы пользователей, а также по созданию интуитивно понятного интерфейса.

## 3.1 Установка Django

Для установки Django в локальном окружении должны быть выполнены два условия: установлен Python и пакетный менеджер Pip.

Перед установкой Django создадим виртуальное окружение (оно также называется virtualenv), которое поможет содержать среду разработки в чистоте. Virtualenv изолирует настройки Python/Django для каждого отдельного проекта.

С помощью команды в консоли «pip install virtualenv» устанавливаем пакет виртуального окружения. Далее определяем каталог для виртуальных сред и запускаем из него консоль. Затем для создания виртуальной среды выполняем команду, представленную на рисунке 3.1.



Рисунок 3.1 – Команда для создания виртуальной среды

Команде «virtualenv» передается название среды, которая в данном случае будет называться «my\_virtenv». После этого в текущей папке будет создан подкаталог my\_virtenv.

Для использования виртуальную среду нужно активировать. И каждый раз, при работе с Django проектом, связанную с ним среду необходимо активировать.

Теперь активируем наше виртуальное окружение. Если при разработке используется ОС Windows, то в папке my\_virtenv/Scripts/ можно найти файл activate.bat, который активирует среду. Поэтому для Windows активация окружения будет выглядеть как показано на рисунке 3.2.

После окончания работы с виртуальной средой можно ее деактивировать. Для этого в той же папке находим файл deactivate.bat и так же запускаем его.

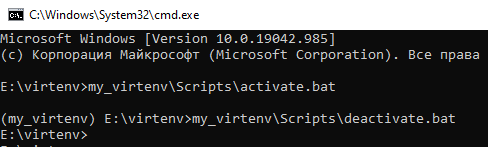


Рисунок 3.2 – Команды активации и деактивации виртуальной среды

После активации виртуальной среды для установки Django выполним в консоли следующую команду: «pip install django». Она устанавливает последнюю версию Django. С помощью данной команды могут устанавливаться любые пакеты языка Python. Для удаления пакета используем соответственно «uninstall».

## 3.2 Файловая структура проекта

Так как для создания веб-приложения используется фреймворк Django, у него есть определенная файловая структура, которая представлена на рисунке 3.3.

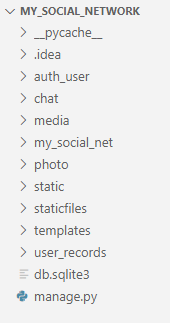


Рисунок 3.3 – Файловая структура проекта

Проект содержит в себе обязательное приложение my\_social\_net и три созданных дополнительно для удобной реализации отдельных компонентов. Также имеют папки с некоторыми статическими файлами и шаблонами. В таблице 3.1 кратко описаны все файлы и папки проекта.

Таблица 3.1 – Основные файлы и папки проекта

|  |  |
| --- | --- |
| Папка | Описание |
| auth\_user | Приложение, отвечающее за авторизацию/регистрацию пользователя. |
| chat | Приложение, отвечающее за беседы пользователей. |

Продолжение таблицы 3.1

|  |  |
| --- | --- |
| media | Папка, хранящая пользовательские медиафайлы |
| my\_social\_net | Главное приложение проекта, отвечающее за профиль и друзей юзера. |
| static | Папка, хранящая изображения, а также файлы .css и .js. |
| templates | Папка, хранящая все шаблоны проекта. |
| user\_records | Приложение, отвечающее за записи пользователя. |
| db.sqlite3 | Файл базы данных. |
| manage.py | Конфигурационный файл для запуска проекта. |

Это стандартная архитектура проекта Django, за исключением добавленных приложений и папки с медиафайлами. В дальнейшем будет рассмотрено создание моделей и представлений, хранящихся внутри каждого приложения.

## 3.3 Разработка моделей

Одной из начальных частей проекта является создание моделей. Для реализации цели дипломного проекта были созданы восемь моделей: MyUser, MyFriends, Chat, ChatMessage, UserRecord, LikeRecord, RepostRecord, CommentRecord. Каждая модель является представлением таблицы из базы данных, где объявляются поля таблицы и связи между ними.

Модель – прототип объекта, используемый вместо оригинала для решения задач. Модель строится на основании ограниченного множества известных свойств и поведения об оригинале.

Основной моделью приложения является MyUser, она имеет связи со всеми остальными. Создание данной модели представлено на листинге 3.1.

class MyUser(AbstractBaseUser):

    id = models.AutoField(primary\_key=True)

    person\_id = models.PositiveIntegerField()

    email = models.EmailField(unique=True)

    username = models.CharField(max\_length=100)

    create\_date = models.DateField(auto\_now\_add=True)

    last\_visit = models.DateTimeField(auto\_now=True)

    avatar = models.ImageField(upload\_to=user\_directory\_path, blank=True)

    city = models.CharField(max\_length=100, default='Неопределен')

    country = models.CharField(max\_length=100, default='Неопределена')

    is\_active = models.BooleanField(default=False)

    is\_staff = models.BooleanField(default=False)

    is\_superuser = models.BooleanField(default=False)

    USERNAME\_FIELD = 'email'

    REQUIRED\_FIELDS = ['person\_id', 'username']

Листинг 3.1 – Создание модели MyUser

Модель MyUser отвечает за хранение информации о пользователе. Также к данной модели были добавлены некоторые методы, отвечающие за получение имени пользователя и его аватара, а также проверяющие имеющиеся разрешения.

Еще одной важной моделью является UserRecord. У нее также имеются собственные методы, такие как сохранение и удаление, и три зависимые таблицы. Код создания данной модели представлен на листинге 3.2.

class UserRecord(models.Model):

    user= models.ForeignKey(MyUser,related\_name='user',on\_delete=models.CASCADE)

    author\_record = models.ForeignKey(MyUser, related\_name='author', on\_delete=

models.CASCADE)

    record\_text = models.TextField()

    date = models.DateTimeField(auto\_now\_add=True)

    image = models.ImageField(upload\_to=user\_directory\_path, blank=True)

Листинг 3.2 – Создание модели UserRecord

В обеих моделях, описанных выше, есть поле для типа данных изображений. В параметрах данного поля указан метод для создания пути при загрузке изображений. Код метода представлен на листинге 3.3.

def user\_directory\_path(instance, filename):

    return 'user\_{0}/gallery/{1}'.format(instance.user.person\_id, filename)

Листинг 3.3 – Метод формирования пути загрузки изображения

В данном методе возвращается путь к личной папке пользователя в католог gallery. Папка юзера имеет в названии id пользователя. Само название файла никак не изменяется, а сохраняется исходным.

Также важной для достижения цели проекта является модель Chat. Ее код представлен в листинге 3.4

class Chat(models.Model):

    author = models.ForeignKey(MyUser, related\_name='chat\_author', on\_delete=

models.CASCADE)

    companion = models.ForeignKey(MyUser, related\_name='chat\_companion', on\_

delete=models.CASCADE)

    last\_change = models.DateTimeField(auto\_now=True)

    user\_last\_change = models.ForeignKey(MyUser, related\_name='chat\_user\_last

\_change', on\_delete=models.CASCADE)

    def save(self, \*args, \*\*kwargs):

        old\_chat = Chat.objects.filter(author = self.companion, companion = self.author)

        if len(old\_chat):

            raise ValidationError('Name must be unique in pair')

        super(Chat, self).save(\*args, \*\*kwargs)

Листинг 3.4 – Создание модели Chat

Данная модель имеет четыре поля, три из которых внешние ключи. Модель реализует метод save, который осуществляет сохранение бесед и их участников. Также этот метод проверяет название чата на уникальность.

## 3.4 Разработка представлений

Центральным моментом любого веб-приложения является обработка запроса, который отправляет пользователь. В Django за обработку запроса отвечают представления или views. По сути, представления представляют функции обработки, которые принимают данные запроса в виде объекта request и генерируют некоторый результат, который затем отправляется пользователю.

По умолчанию представления размещаются в приложении в файле views.py. На листинге 3.5 представлен код метода для отображения страницы поиска.

def search\_page(request):

    ''' функция для отображения страницы с результатами поиска.'''

    if request.method == 'GET':

        search\_name = request.GET["search\_name"]

        all\_users = MyUser.objects.filter(username\_\_icontains = search\_name)

        if request.user.is\_authenticated:

            log\_user = request.user

            my\_friends, count\_my\_friends = all\_friends(log\_user)

            if count\_my\_friends > 10:

                    my\_friends = my\_friends[:11]

            count\_my\_photos = UserPhoto.objects.filter(user = log\_user).count()

            all\_chats = len(list(Chat.objects.filter(author = log\_user).order\_by('last\_change') | Chat.objects.filter(companion = log\_user).order\_by('last\_change')))

            count\_new\_friends = MyFriends.objects.filter(friend=log\_user, status=0).count()

            return render(request, 'search.html',

{'person': request.user, 'log\_user': log\_user,

'count\_new\_friends':count\_new\_friends,

'person\_friends':my\_friends, 'count\_friends':count\_my\_friends,

'count\_my\_friends':count\_my\_friends, 'count\_my\_photos': count\_my\_photos, 'count\_my\_chats': all\_chats, 'all\_users': all\_users})

        else:

             return render(request,

'search.html', {'all\_users': all\_users})

Листинг 3.5 – Метод отображения страницы поиска

В данном методе с помощью GET-запроса мы получаем данные по заданному поиску, в результате чего вызываем и рендерим нужный шаблон. Полный код файла представления находится в приложении Е.

## 3.5 Разработка шаблонов

Шаблоны отвечают за формирование внешнего вида приложения. Они представляют специальный синтаксис, который позволяет внедрять данные в код HTML. Для хранения шаблонов была создана папка templates.

В этой папке создаем файл base.html. По сути, это обычная веб-страница, которая содержит код html. Теперь используем эту страницу для получения некоторой личной информации пользователя. На листинге 3.6 представлен фрагмент кода, который будет отображать личные данные юзера. Полный код страницы находится в представлении Ж.

<div id="person-information" class="logo">

    {%block person\_description%}

    <div class="person-description">

         <table class="person-description-table-base" style="margin-bottom: 68px;">

              <tr>

                  <td colspan="2"><b>

<a href="/person/id\_{{person.person\_id}}"

                    style="text-align: left; color: #00BFFF;

 font-size: 26px;">{{person.username}}</a></b>

                   </td>

               </tr>

                <tr>

                    <td colspan="2" style="text-align: left;

color: #6E6E70; font-weight: bolder;">

                    {%if person.country == 'Неопределена' and

person.city == 'Неопределен'%}

                                    {{person.city}}

                    {%else%}

                            {{person.city}}, {{person.country}}

                    {%endif%}

                    </td>

                </tr>

Листинг 3.6 – Фрагмент кода отображения личных данных

По подобному принципу построены все шаблоны приложения. Нужный шаблон вызывается в методах реализованных в файлах представлений. Также в шаблоны можно подключать дополнительно созданные формы. Реализация форм происходит по тому же принципу, что и у шаблонов.

## 3.6 Реализация алгоритмов

### 3.6.1 Реализация добавления новой записи

Алгоритм добавления новой записи, описанный в пункте 2.4.2, был реализован в функции add\_record, входящей в представление приложения user\_records. Код данного алгоритма представлен на листинге 3.7.

def add\_record(request):

    ''' функция добавления пользователем новой записи'''

    if request.user.is\_authenticated and request.method == 'POST':

        record\_text = request.POST["record\_text"]

        if not request.FILES:

            record = UserRecord(user = request.user, author\_record =

request.user, record\_text = record\_text)

            record.save()

        else:

            user\_image = request.FILES['image']

            record = UserRecord(user = request.user, author\_record =

request.user, record\_text = record\_text,

image = user\_image)

            record.save()

        return redirect(request.META['HTTP\_REFERER'])

Листинг 3.7 – Код алгоритма создания записи

### 3.6.2 Реализация отображения страницы настроек пользователя

Алгоритм отображения настроек пользователя, описанный в пункте 2.4.1, был реализован в функции my\_settings, созданной в представлении приложения my\_social\_net. Фрагмент кода данного алгоритма представлен на листинге 3.8, полная версия метода находится в приложении З.

def my\_settings(request):

    ''' функция для отображения страницы настроек пользователя'''

    if request.user.is\_authenticated:

        user = request.user

        if request.method == 'POST':

            #проверяем передаёт ли пользователь файл

            if request.FILES:

                if request.FILES['avatar']:

                    #проверяем является ли файл картинкой

                    form = ImageForm(request.POST, request.FILES)

                    if form.is\_valid():

                        person\_image = request.FILES['avatar']

                        person\_id = form.cleaned\_data['person\_id']

                        #удаляем старое фото

                        old\_path\_to\_avatar = os.path.join(settings.BASE\_DIR, '../media/person/user\_%s/avatar/image') % (person\_id)

                        delete\_file(old\_path\_to\_avatar)

                        #сохраняем новое фото в качестве аватара

                        user.avatar.save('image', person\_image)

                        user.save()

                        return redirect('/my\_settings')

Листинг 3.8 – Фрагмент кода алгоритма отображения настроек пользователя

### 3.6.3 Реализация создания нового сообщения

Алгоритм создания нового сообщения, описанный в пункте 2.4.3, был реализован в функции new\_massage, входящей в представление приложения chat. Фрагмент кода данной функции представлен на листинге 3.9, а полная версия находится в приложении И.

def new\_message(request):

    ''' функция для создания нового сообщения'''

    if request.user.is\_authenticated and request.method == 'POST':

        companion\_id = request.POST["companion\_id"]

        author\_id = int(request.POST["author\_id"])

        message\_text = request.POST["message\_text"]

        author = request.user

        if author\_id == request.user.person\_id:

            companion = MyUser.objects.get(person\_id = companion\_id)

        else:

            companion = MyUser.objects.get(person\_id = author\_id)

        try:

            chat = Chat(author = author, companion = companion, user\_last\_change = request.user)

            chat.save()

            print('new\_chat')

Листинг 3.9 – Фрагмент кода создания нового сообщения

В данном алгоритме id обоих собеседников должны быть уникальными. Поэтому мы дважды проверяем это условие. Таким образом, определяем, кто является автором, а кто собеседником. А уже после переходим к чату.

## 3.7 Выводы по разделу

В данном разделе была описана программная реализация веб-приложения. Итогом выполнения разработки программного средства стало веб-приложение для создания записей и общения в сети в реальном времени, отвечающее всем требованиям, предъявленным к нему в начале разработки.

# 4 Тестирование программного средства

Утверд.

Н. контр.

Провер.

Разраб.

*Смелов В.В.*

*Копыток Д.В.*

*Голуб Е.В.*

*Кантарович В. С.*

Ф.И.О.

Подпись

Дата

4

*4 Тестирование программного средства*

Лист

1

*БГТУ 04.00.ПЗ*

Лит.

Листов

*БГТУ 74218007, 2021*

Тестирование является неотъемлемой частью процесса разработки программного продукта, так как данный этап позволяет найти неисправности в работе приложения и исправить их.

В данной главе будут описаны некоторые тесты, проведенные для выявления и устранения неисправностей разработанного программного продукта.

Для начала представим названия тестов, которые и будем рассматривать:

– регистрация пользователя;

– создание новой записи;

– обновление аватара;

– получение данных без авторизации.

## 4.1 Тестирование регистрации пользователя

В данном тесте будет проверена правильность ввода данных. Для этого введем, например, в поле с почтой неправильное значение, без знака “@”. На рисунке 4.1 можно увидеть результат данных действий.

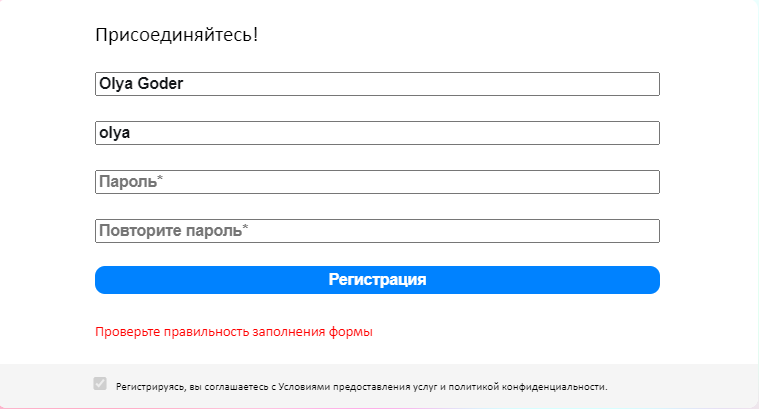


Рисунок 4.1 – Страница регистрации пользователя с валидацией

После ввода некорректных данных внизу отобразилось сообщение об ошибке, говорящее о том, что был введен неверный формат данных.

## 4.2 Тестирование создания записи

В данном тесте будет проверена возможность создания пустой записи. Для этого в текстовое поле ничего не вводим, изображение также не прикрепляем. На рисунке 4.2 можно увидеть результат данных действий.

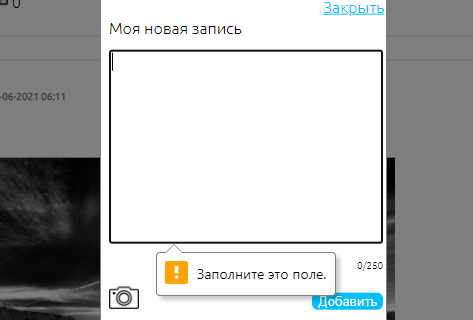


Рисунок 4.2 – Создание записи с валидацией

В результате того, что никакой текст не был введен, внизу поля отобразилось сообщение об ошибке, говорящее о том, что поле должно быть заполнено.

Также, будет проведен тест на добавление файла. Для начала прикрепим текстовый файл к записи. На рисунке 4.3 можно увидеть результат данных действий.



Рисунок 4.3 – Прикрепление файла к новой записи с валидацией

В результате прикрепления к записи файла неверного формата, внизу поля было выведено сообщение об ошибке.

## 4.3 Тестирование обновления аватара

Для проведения тестирования обновления аватара пользователя переходим на страницу настроек и кликаем «Сменить фото». После появления диалогового окна не выбираем никакой файл. Демонстрация представлена на рисунке 4.4.

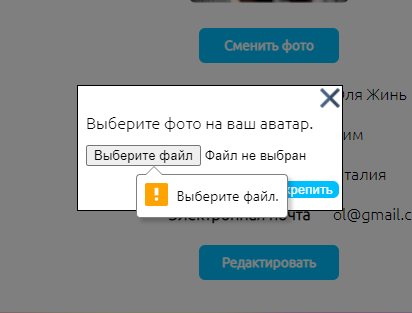


Рисунок 4.4 – Обновление аватара пользователя с валидацией

В результате того, что не был прикреплен ни один файл, внизу поля выведено сообщение, говорящее о том, что нужно выбрать файл.

## 4.4 Тестирование получения данных без авторизации

Для проведения тестирования получения данных из базы данных нужно в поисковую строку ввести адрес сервера и строку запроса на получение данных, например, о друзьях. Демонстрация этого представлена на рисунке 4.5.

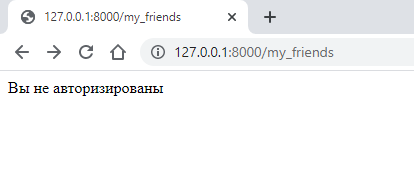


Рисунок 4.5 – Запрос без авторизации

После чего, авторизуемся и запрос отправиться уже с заголовком, который включает в себя токен авторизации. Результат можно увидеть на рисунке 4.6.

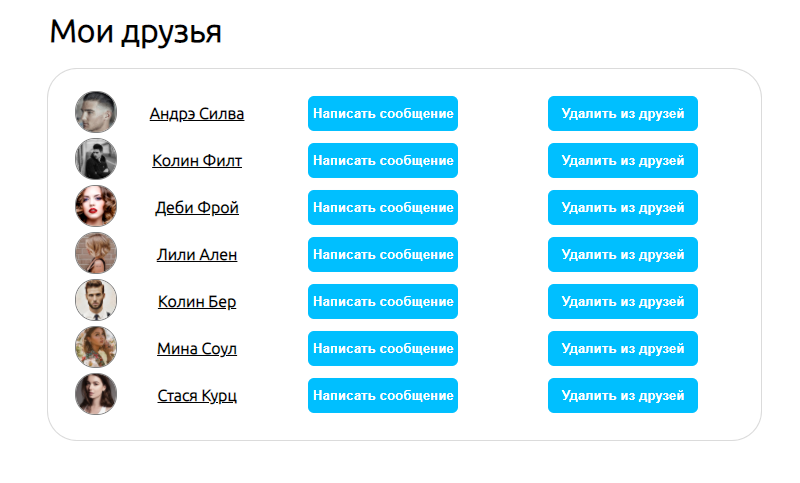


Рисунок 4.6 – Страница «Друзья»

На данной странице видны данные, которые были получены с сервера после авторизации. Это позволяет подтвердить, что тест прошел успешно.

## 4.5 Выводы по разделу

В данной главе были выполнен ряд тестов для проверки стабильности и правильности работы веб-приложения. В результате проделанной работы установлено, что система стабильно работает и в критических ситуациях.

# 5 Руководство пользователя

Утверд.

Н. контр.

Провер.

Разраб.

*Смелов В.В.*

*Копыток Д.В.*

*Голуб Е.В.*

*Кантарович В. С.*

Ф.И.О.

Подпись

Дата

7

*5 Руководство пользователя*

Лист

1

*БГТУ 05.00.ПЗ*

Лит.

Листов

*БГТУ 74218007, 2021*

## 5.1 Руководство для администратора

Администратор данного проекта задумывался как технический специалист, который будет способен заниматься поддержкой кода и управлять базой данных пользователей. Администратору доступен весь функционал, что и пользователю.

Для того, чтобы попасть на администраторскую часть приложения, нужно в адресной строке браузера дописать «/admin». При запуске администратор попадает на страницу авторизации, представленную на рисунке 5.1.

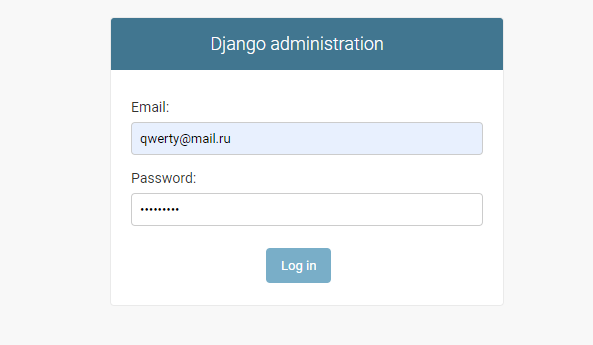


Рисунок 5.1 – Страница авторизации администратора

Первым делом администратору необходимо войти в систему под своим логином и паролем.

Один из предустановленных администраторов системы имеет логин «[qwerty@mail.ru*»*](mailto:qwerty@mail.ru) и пароль «qwerty123».

Для входа нужно ввести данные авторизации и нажать кнопку «Log in». Далее пользователь попадает на страницу административной панели. Страница представлена на рисунке 5.2.

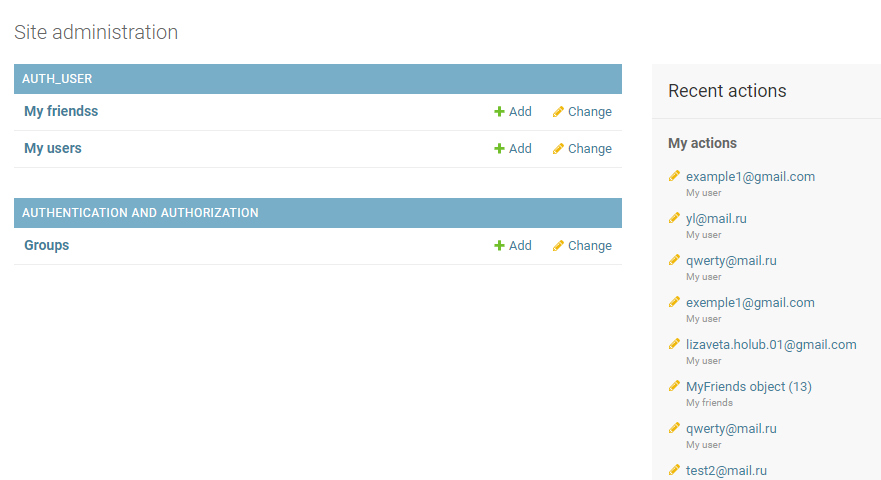


Рисунок 5.2 – Страница административной панели

После успешной авторизации мы можем наблюдать сообщение с логином пользователя вверху страницы. Данное сообщение продемонстрировано на рисунке 5.3.

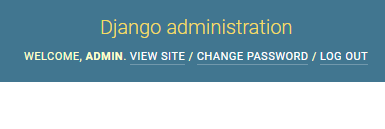


Рисунок 5.3 – Сообщение с именем пользователя

Далее администратор может выполнять все доступные ему действия.

Справа на странице размещен список недавних пользовательских событий. Слева расположено меню. Переходя в пункты меню, администратор может добавлять, удалять и изменять пользователей, а также их отношения с другими.

На рисунке 5.4 в панели администратора представлена страница пользователя с зашифрованным паролем и данными о последнем входе в аккаунт.

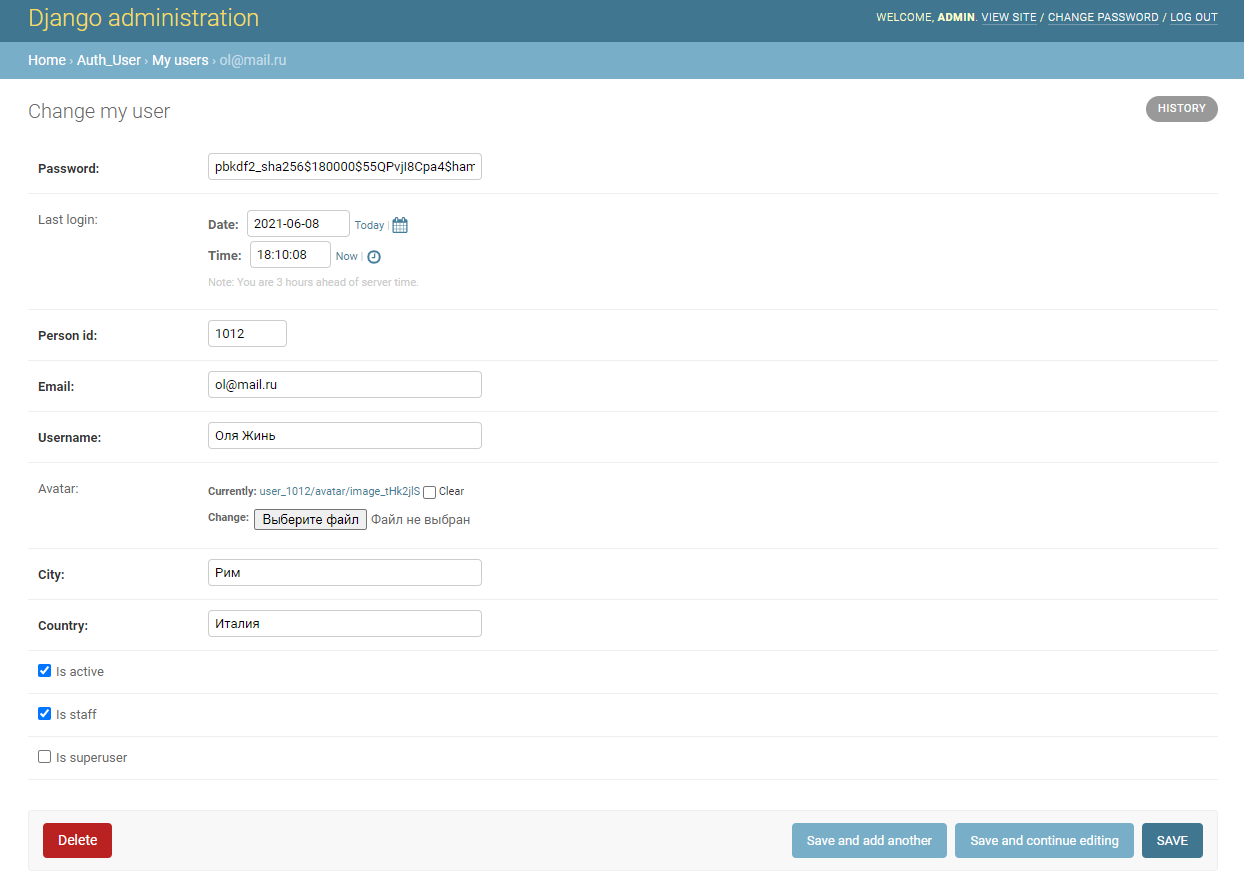


Рисунок 5.4 – Страница пользователя в панели администратора

Через данную панель также можно легко перейти к просмотру пользовательской части приложения. Вверху, рядом с сообщением об имени текущего пользователя имеется кнопка «View site». Если на нее кликнуть, произойдет переход к пользовательскому интерфейсу приложения. Рядом с именем текущего пользователя также расположены кнопки для изменения пароля и выхода из администраторской панели.

## 5.2 Руководство для пользователя

На главной странице приложения расположено название веб-приложения и логотип с небольшим описанием, как показано на рисунке 5.5.

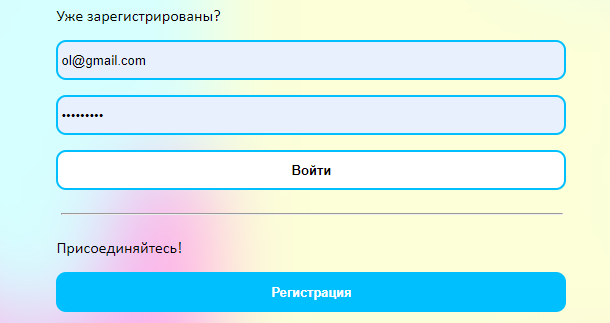


Рисунок 5.5 – Страница авторизации пользователя

Первым делом пользователю необходимо войти в систему или зарегистрироваться. Для регистрации нажимаем кнопку «Регистрация» и заполняем форму, как представлено на рисунке 5.6. Для входа же на главной странице нужно ввести логин и пароль и нажать кнопку «Вход».

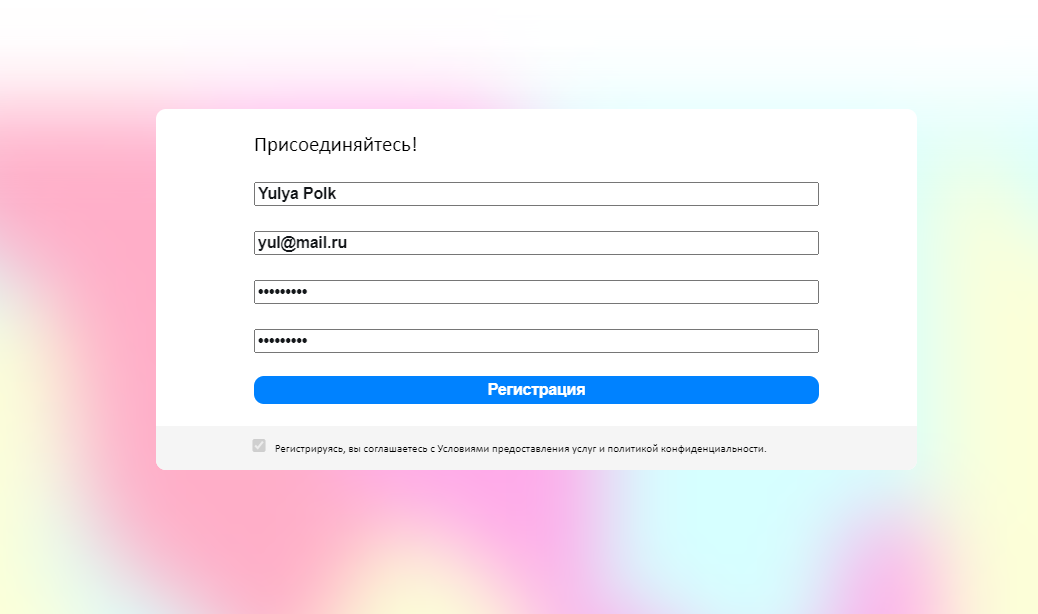


Рисунок 5.6 – Страница регистрации

После того, как авторизовались, попадаем на страницу пользователя, которая продемонстрирована на рисунке 5.7. На данной странице представлена краткая информация о пользователе.

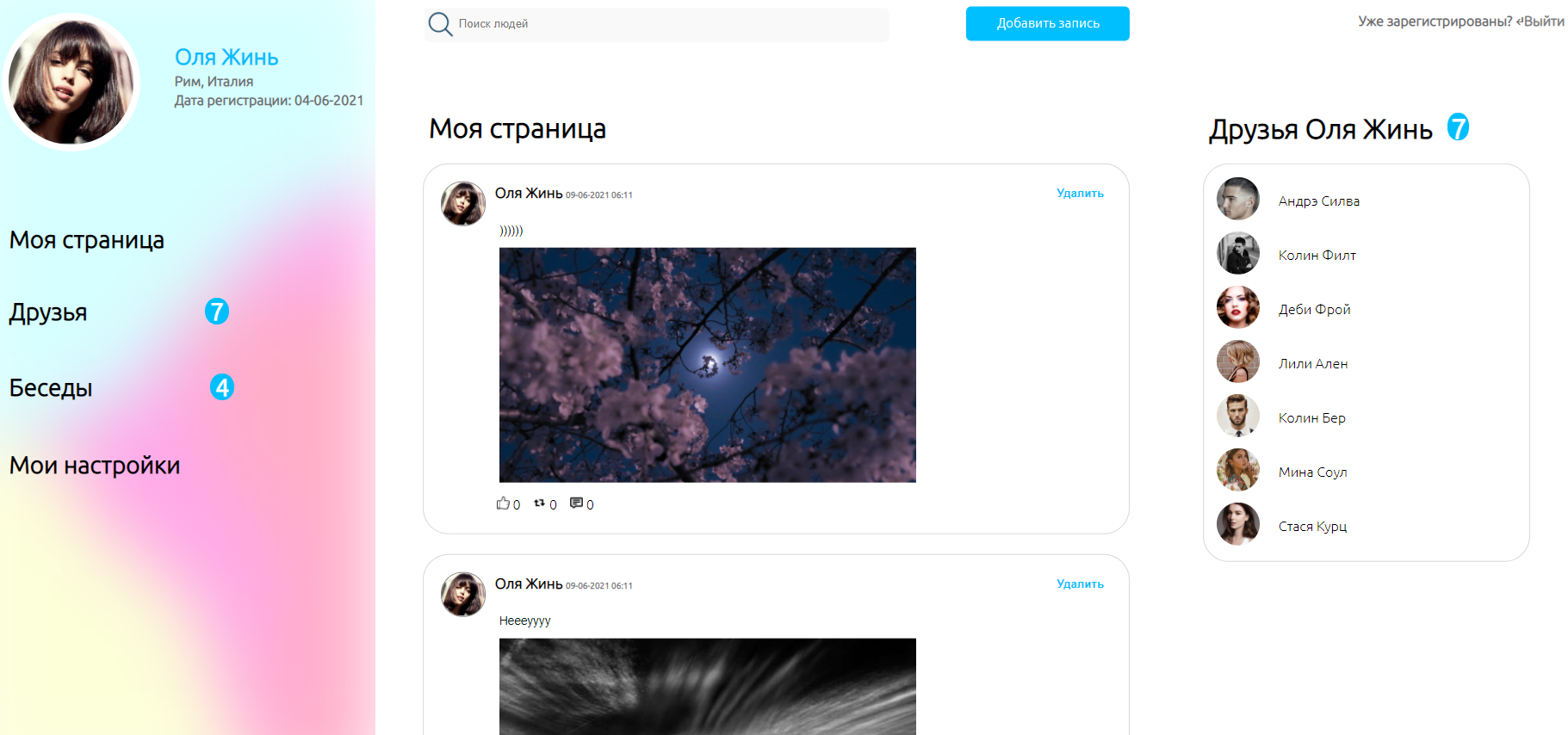


Рисунок 5.7 – Страница пользователя

Все страницы разработаны идентичным образом: в левой части меню пользователя, в правой части панель с друзьями или возможная область для рекламы, а вся динамическая информация в свободной области по середине.

В меню пользователя находится четыре гиперссылки: «Моя страница», «Друзья», «Беседы», «Мои настройки». С помощью ссылки «Моя страница» всегда можно вернуться к своему профилю.

В средней области пользовательской страницы размещены строка поиска, кнопка для добавления записи и стена с записями.

Для того, чтобы добавить запись на стену, нужно нажать кнопку «Добавить запись», заполнить текстовое поле и, возможно, прикрепить изображение. Новый пост сразу же отобразится на стене. Вверху поста расположена кнопка «Удалить», кликнув на которую запись легко удаляется.

Также пользователь не может лайкать, репостить или комментировать собственные записи. Это возможно только на страницах других пользователей. Перейдя на страницу интересующего вас юзера, нажав на какую-либо кнопку внизу поста, можно произвести оценку записи, такую как лайк, репост, комментарий.

Перейти на страницу другого пользователя достаточно легко. Для этого просто требуется кликнуть по его имени, которое вы можете увидеть либо в ваших друзьях, либо при поиске.

Для поиска можно попробовать ввести имя пользователя или нажать на значок лупы рядом со строкой. При первом варианте, будут найдены пользователи с определенным именем. При втором, будет выведен список всех пользователей приложения, как представлено на рисунке 5.8.

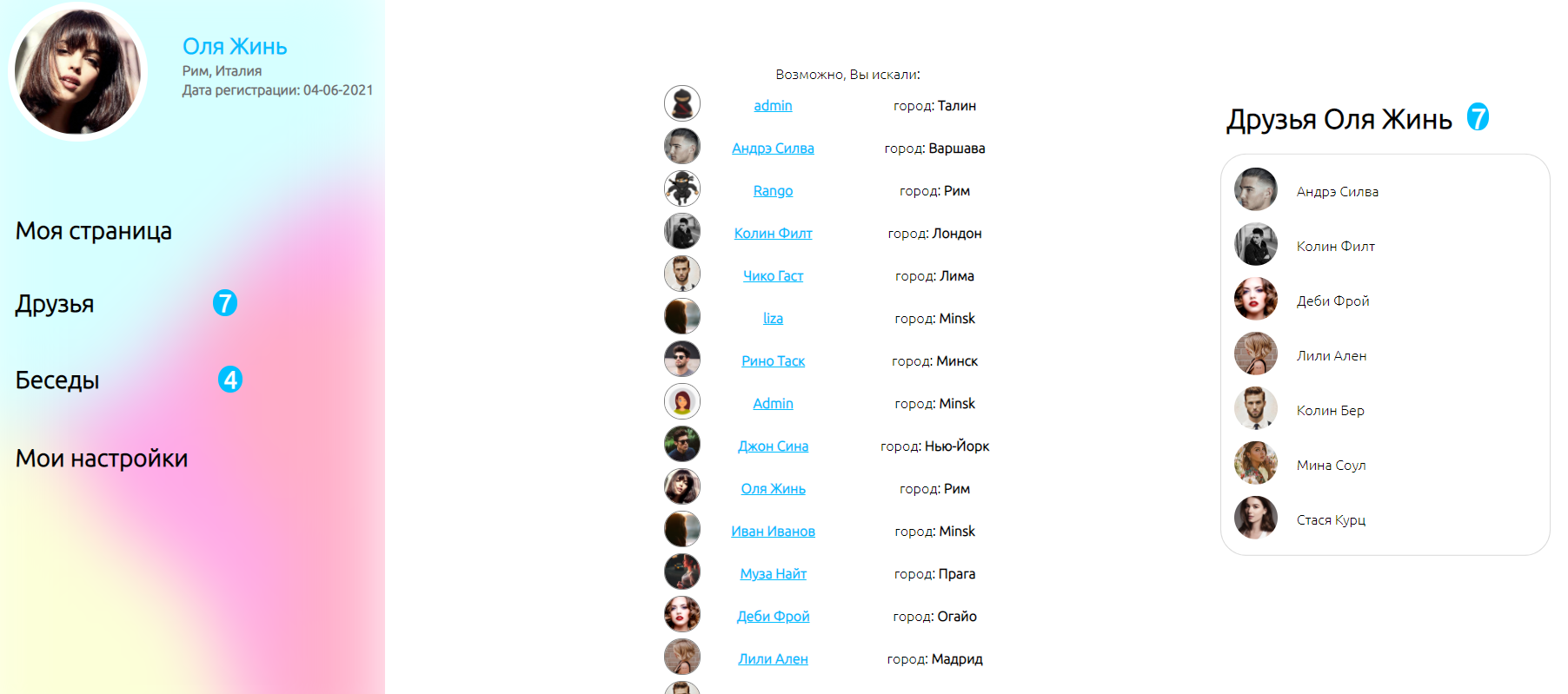


Рисунок 5.8 – Страница поиска

С помощью пункта меню «Друзья» пользователь попадает на страницу со своими друзьями в приложении. На этой странице, помимо списка друзей, есть кнопки «Написать сообщение» и «Удалить из друзей».

Нажав кнопку «Удалить из друзей», пользователь удаляет отношение с данным юзером. Кликнув же кнопку «Написать сообщение», можно увидеть диалоговое окно, с помощью которого вы можете отправить другому пользователю сообщение, не заходя в беседу.

На рисунке 5.9 продемонстрирована страница друзей пользователя.

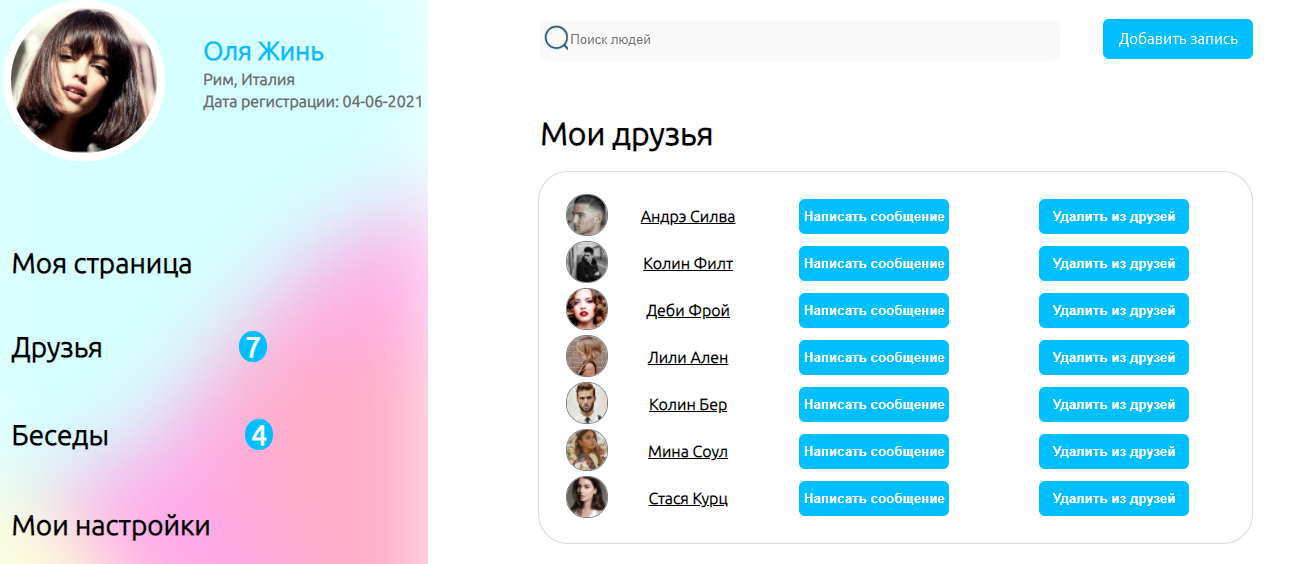


Рисунок 5.9 – Страница друзей

Для того, чтобы у пользователя появились друзья, на странице интересующего юзера ему нужно нажать кнопку «Добавить в друзья», которая находится вверху левой панели. Таким образом, будет отправлен запрос. После того, как пользователь, получивший такой запрос, в зависимости от того, принял или отклонил он заявку, отобразится или нет в друзьях юзера, отправившего запрос.

Чтобы перейти к общению в чате, пользователю нужно либо воспользоваться кнопкой «Написать сообщение», которую можно найти в друзьях пользователя или на странице другого юзера, либо с помощью пункта меню «Беседы» перейти на страницу существующих бесед пользователя. Про первый вариант уже было рассказано выше. При втором варианте, пользователь попадает на страницу его бесед, представленную на рисунке 5.10.

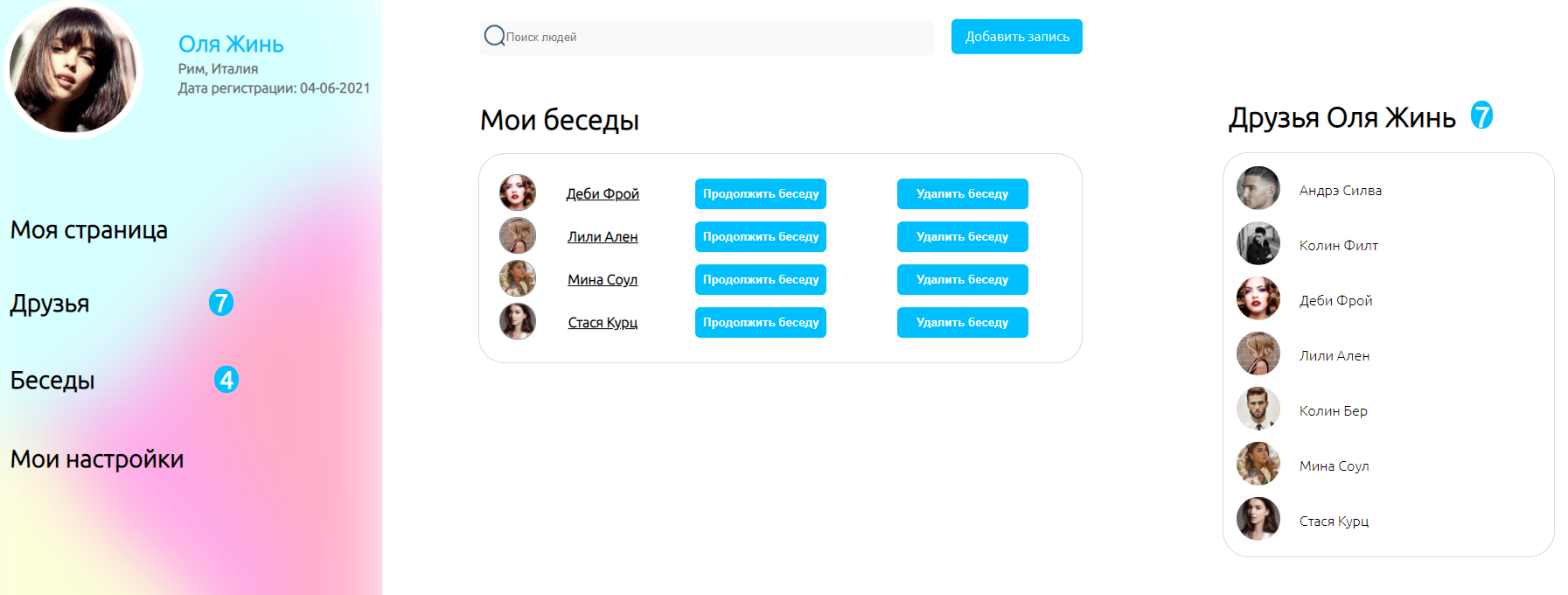


Рисунок 5.10 – Страница бесед

На данной странице, как и на странице друзей, имеется список юзеров, с которыми у пользователя есть беседы, и две кнопки: «Продолжить беседу» и «Удалить беседу». Если нажать «Удалить беседу», соответственно она исчезнет со страницы пользователя. Если же нажать кнопку «Продолжить беседу», появится диалоговое окно с отображением истории сообщений двух пользователей, полем для ввода текста и кнопкой отправить. Таким образом, юзер легко может общаться с другими в сети.

Последний пункт меню пользователя – «Мои настройки». Перейдя по данной гиперссылке? можно увидеть страницу, продемонстрированную на рисунке 5.11.

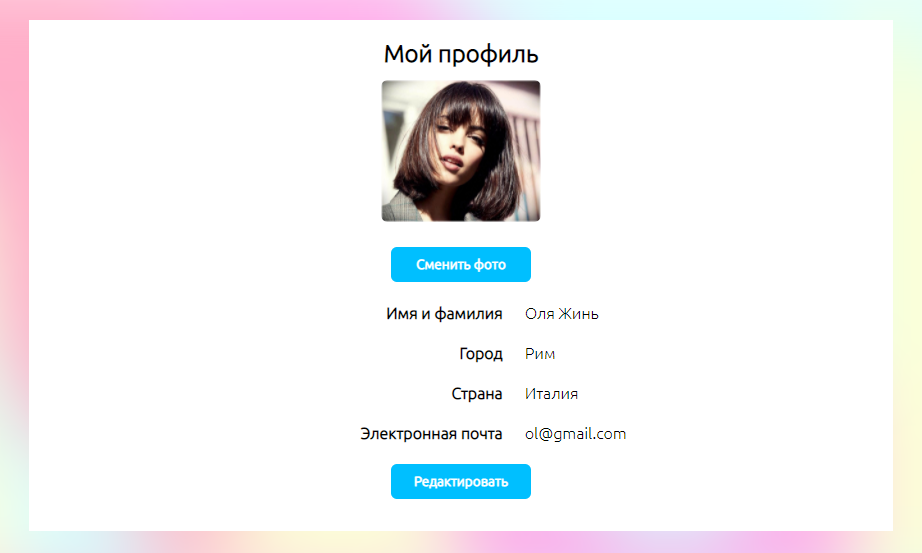


Рисунок 5.11 – Страница настроек пользователя

На данной странице пользователь может изменить информацию о себе. Для этого нужно нажать кнопку «Редактировать», в открывшемся диалоговом окне ввести необходимые данные и кликнуть «Изменить». После этого данные будут обновлены. Таким же образом можно изменить аватар, нажав кнопку «Сменить фото». Вернувшись на свою страницу, юзер увидит уже новые данные о себе.

## 5.3 Выводы по разделу

Данный раздел предоставляет пояснения по работе пользователя с веб-приложением. Так как интерфейс приложения прост и интуитивно понятен, то у пользователя не может возникнуть трудностей с его эксплуатацией.

Так как приложение имеет разделение ролей на администратора и пользователей, то руководство в данной главе было описано для двух ситуаций: когда приложением будет пользоваться администратор и когда пользователь.

Архитектуру приложения и большие функциональные возможности, предоставляемые пользователю, получилось ограничить небольшим количеством экранов, что является прямым доказательством хорошо спроектированного и простого пользовательского опыта в данном приложении.

# 6 Технико**-экономическое обоснование проекта**

Утверд.

Н. контр.

Провер.

Разраб.

*Смелов В.В.*

*Копыток Д.В.*

*Голуб Е.В.*

*Кантарович В.С.*

Ф.И.О.

Подпись

Дата

9

*6 Технико-экономическое обоснование проекта*

Лист

1

*БГТУ 06.00.ПЗ*

Лит.

Листов

*БГТУ 74218007, 2021*

Консульт.

*Евлаш А.И.*

Главной целью экономического раздела является экономическое обоснование целесообразности разработки программного средства (ПС), представленного в дипломном проекте. В данном разделе пояснительной записки проводится расчет затрат на всех стадиях разработки и расчет экономии основных видов ресурсов в связи с использованием данного ПС.

Разработка проектов ПС требует разнообразных затрат и нередко значительных объемов ресурсов (трудовых, материальных, финансовых). В связи с этим, разработка и реализация каждого проекта должна быть обоснована, как технически, так и экономически.

## 6.1 Общая характеристика разрабатываемого программного средства

Разработанное в данном дипломном проекте программное средство предназначено для создания записей и общения в сети. Приложение является веб-ресурсом, благодаря чему может быть запущено и с персонального компьютера, и с мобильного устройства.

Программное средство создано при помощи среды разработки PyCharm и языка программирования Python. Для реализации функционала был использован фреймворк Django. Также дополнительно были использованы языки JavaScript, HTML. Для хранения данных использовалась база данных SQLite.

Разработанный продукт позволит выполнять следующие действия:

* редактирование профиля;
* добавление записей;
* удаление записей;
* добавление комментария к записи;
* добавление лайка записи;
* репост записи;
* поиск друзей;
* добавление друзей;
* удаление друзей;
* общение в реальном времени.

Программное средство будет предоставляться бесплатно.

Далее в этом разделе будет произведен расчет затрат на всех стадиях разработки ПС и определена себестоимость. Также необходимо провести расчет экономии основных видов ресурсов в связи с использованием данного программного средства.

## 6.2 Исходные данные и маркетинговый анализ

Исходные данные для расчета представлены в таблице 6.1. Данные взяты из нормативов, действующих на момент написания дипломного проекта.

Таблица 6.1 – Исходные данные для расчета

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование показателя | Единица измерения | Условные  обозначения | Норматив |
| Численность разработчиков | чел. | Чр | 1 |
| Коэффициент изменения скорости обработки информации |  | Кск | 0,4 |
| Норматив дополнительной заработной платы | % | Ндз | 12 |
| Ставка отчислений в Фонд социальной защиты населения | % | Нфсзн | 34 |
| Ставка отчислений в БРУСП «Бел-госстрах» | % | Ннбгс | 0,4 |
| Норма расхода материалов | руб. / 100 строк кода | Нм | 0,46 |
| Норматив машинного времени на отладку исходного кода | машино-час / 100 строк кода | Нмв | 15 |
| Цена одного машино-часа | руб. | Смч | 0,06 |
| Норматив прочих затрат | % | Нпз | 8,5 |
| Норматив накладных расходов | % | Нобп*,* обх | 114 |
| Норматив расходов на сопровожде-ние и адаптацию | % | Нрса | 10 |

Источниками исходных данных для расчёта средней цены разработки программного средства данного дипломного проекта, выступает таблица 6.2 с ценами аналогичных программных продуктов с ограниченным функционалом.

Таблица 6.2 – Стоимость разработки аналогичных веб-приложений.

|  |  |
| --- | --- |
| Наименование программного продукта | Стоимость |
| Веб-приложение «Twitter» | 15 500 руб. |
| Веб-приложение «ВКонтакте» | 19 000 руб. |
| Веб-приложение «Facebook» | 23 000 руб. |

В вышеуказанную стоимость разработки входят некоторые программные модули, такие как проектирование и реализация серверной части, базы данных и пользовательского интерфейса, реализованного в виде сайта. Также в ходе маркетингового анализа было установлено, что средняя цена разработки программных продуктов, которые представляют собой веб-приложения, разработанные с использованием базы данных SQLite и фреймворка Django равна сумме от 11000 рублей. Отдельно стоит учитывать разработку дизайна веб-ресурса 1500 рублей. Таким образом, примерная стоимость разработки программного средства, выбранного в качестве базы сравнения составит от 12500 руб.

## 6.3 Методика обоснования цены

Разработка проектов программных средств, требует затрат разнообразных ресурсов (трудовых, материальных и финансовых). В связи с этим, необходимость разработки и реализации каждого проекта обосновывается, как с технической точки зрения, так и с экономической. Для обоснования экономической целесообразности проекта вычисляются различные показатели.

В современных рыночных экономических условиях программное средство выступает преимущественно в виде продукции организаций, представляющей собой функционально завершенные и имеющие товарный вид программы, реализуемые покупателям по рыночным отпускным ценам.

Широкое применение вычислительных технологий требует постоянного обновления и совершенствования программного средства. Выбор эффективных проектов связан с их экономической оценкой и расчетом экономического эффекта, который может определяться как у разработчика, так и у пользователя.

У разработчика экономический эффект выступает в виде чистой прибыли от реализации программного средства, остающейся в распоряжении организации, а у пользователя – в виде экономии трудовых, материальных и финансовых ресурсов, получаемой за счет следующего:

* снижения трудоемкости расчетов и алгоритмизации программирования и отладки программ;
* сокращения расходов на оплату машинного времени и других ресурсов на отладку программ;
* снижения расходов на материалы;
* ускорение ввода в эксплуатацию новых систем;
* улучшения показателей деятельности в результате использования программного средства, разработанного в данной дипломной работе.

Стоимостная оценка ПС у разработчиков предполагает определение затрат, что включает следующие статьи:

* дополнительная заработная плата исполнителей;
* основная заработная плата исполнителей;
* отчисления в фонд социальной защиты населения;
* отчисления по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний;
* расходы на материалы и комплектующие;
* расходы на спецоборудование;
* расходы на оплату машинного времени;
* прочие прямые затраты;
* накладные расходы.

На основании затрат рассчитывается себестоимость и отпускная цена ПС.

### 6.3.1 Объем программного средства

Для оценки объема программного средства, все его функции классифицируются с использованием специального каталога функций, который определяет их объем. Общий объем программного средства *Vo*, вычисляется как сумма объемов *Vi* каждой из *n* его функций (формула 6.1).

|  |  |  |
| --- | --- | --- |
|  | , | (6.1) |

В таблице 6.3 представлены функции, присутствующие в рассматриваемом программном средстве и их объем в условных машино-командах.

Таблица 6.3 – Содержание и объем функций в программном средстве

|  |  |  |
| --- | --- | --- |
| № функции | Содержание функции | Объем, условных  машино-команд |
| 101 | Организация ввода информации | 480 |
| 102 | Контроль, предварительная обработка | 620 |
| 109 | Организация ввода/вывода информации в интерактивном режиме | 500 |
| 111 | Управление вводом/выводом | 1500 |
| 203 | Формирование БД | 410 |
| 207 | Манипулирование данными | 1300 |
| 506 | Обработка ошибочных и сбойных ситуаций | 390 |
| 507 | Обеспечение интерфейса между компонентами | 860 |
| 604 | Справка и обучение | 490 |
| 707 | Графический вывод результатов | 970 |

Опираясь на данные таблицы 6.3, можно определить объем программного средства, разработанного в ходе дипломного проектирования.

*Vo* = 480 + 620 + 500 + 1500 + 410 + 1300 + 390 + 860 + 490 + 970 = 7520 (условных машино-команд).

Уточненный объем программного средства *Vo/* равен произведению объема программного средства Vo на коэффициент изменения скорости обработки информации Кск (формула 6.2).

|  |  |  |
| --- | --- | --- |
|  | *Vo/* = *Vo*· Кск , | (6.2) |

Исходя из вычисленного объема программного средства, можно определить уточненный объем программного средства:

*Vo/* = 7520 ⋅ 0,6 = 4512 (условных машино-команд).

### 6.3.2 Основная заработная плата

Для определения величины основной заработной платы, было проведено исследование величин заработных плат для начинающих специалистов в сфере разработки приложений на Django. В итоге было установлено, что средняя месячная заработная плата на позиции junior составляет 800 рублей.

Проект разрабатывался одним человеком на протяжении трех месяцев. Таким образом, основная заработная плата будет рассчитываться по формуле (6.3):

|  |  |  |
| --- | --- | --- |
|  | , | (6.3) |

где Соз – основная заработная плата, руб.;

Траз – время раработки, месяцев;

Краз – количество разработчиков, человек;

Сзп – средняя месячная заработная плата.

|  |
| --- |
| (руб.). |

Таким образом, основная заработная плата специалиста по разработке приложений на Django на позиции Junior составила 2400 рублей.

### 6.3.3 Дополнительная заработная плата

Дополнительная заработная плата на конкретное программное средство включает выплаты, предусмотренные законодательством о труде, и определяется по нормативу в процентах к основной заработной плате по формуле (6.4):

|  |  |
| --- | --- |
|  | (6.4) |

где Соз – основная заработная плата, руб.;

Ндз – норматив дополнительной заработной платы, %.

(руб.).

Таким образом, дополнительная заработная плата специалиста по разработке приложений на Django на позиции Junior составила 288 рублей.

### 6.3.4 Отчисления в Фонд социальной защиты населения

Отчисления в Фонд социальной защиты населения (ФСЗН) определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей.

Таким образом, отчисления в Фонд социальной защиты населения, вычисляются по формуле (6.5):

|  |  |
| --- | --- |
|  | (6.5) |

где Соз – основная заработная плата, руб.;

Сдз – дополнительная заработная плата на конкретное ПС, руб.;

Нфсзн – норматив отчислений в Фонд социальной защиты населения и БРУСП «Белгосстрах», %.

Отчисления в БРУСП «Белгосстрах» вычисляются по формуле 6.6:

|  |  |
| --- | --- |
|  | (6.6) |

где Соз – основная заработная плата, руб.;

Сдз – дополнительная заработная плата на конкретное ПС, руб.;

Нбгс – норматив отчислений в БРУСП «Белгосстрах», %.

(руб.).

(руб.).

Таким образом, общие отчисления в фонд социальной защиты населения составили 913,92 руб., а в БРУСП «Белгосстрах» – 10,75 руб.

### 6.3.5 Расходы на материалы

Сумма расходов на материалы См определяется как произведение нормы расхода материалов в расчете на сто строк исходного кода Нм на уточненный объем программного средства *V*o*/* (формула 6.7).

|  |  |
| --- | --- |
|  | (6.7) |

Учитывая, что норма расхода материалов в расчете на сто строк исходного кода равна 0,460 рублей, можно определить сумму расходов на материалы:

См = 0,46 ⋅ 4512 / 100 = 20,75 (руб.).

### 6.3.6 Расходы на оплату машинного времени

Сумма расходов на оплату машинного времени Смв определяется как произведение стоимости одного машино-часа Смч на уточненный объем программного средства *V*o*/* и на норматив расхода машинного времени на отладку ста строк исходного кода НМВ (формула 6.8).

|  |  |
| --- | --- |
|  | (6.8) |

Учитывая, что норматив машинного времени на отладку ста строк исходного кода равен 15, можно определить сумму расходов на оплату машинного времени:

Смв =0,06 ⋅ 4512 ⋅ 15 / 100 = 40,61 (руб.).

### 6.3.7 Прочие прямые затраты

Прочие затраты являются частью затрат, которые стоит так же учитывать при подсчете затрат. Невозможно заранее спланировать затраты на операционные, внереализационные или чрезвычайные расходы.

Сумма прочих затрат Спз определяется как произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив прочих затрат в целом по организации Нпз (формула 6.9).

|  |  |  |
| --- | --- | --- |
|  |  | (6.9) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму прочих затрат:

Спз = 2400 ⋅ 8,5 / 100 = 204 (руб.).

### 6.3.8 Накладные расходы

Сумма накладных расходов Собп,обх – произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив накладных расходов в целом по организации Нобп,обх (формула 6.10).

|  |  |  |
| --- | --- | --- |
|  |  | (6.10) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму накладных расходов:

Собп,обх = ⋅ 114 / 100 = 2736 руб.

### 6.3.9 Сумма расходов на разработку программного средства

Сумма расходов на разработку программного средства Ср определяется как сумма основной и дополнительной заработных плат исполнителей на конкретное программное средство, отчислений на социальные нужды, расходов на материалы, расходов на оплату машинного времени, суммы прочих затрат и суммы накладных расходов (формула 6.11).

|  |  |  |
| --- | --- | --- |
|  | Ср = Соз + Сдз + Сфсзн+ Сбгс + См + Смв + Спз + Собп,обх | (6.11) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму расходов на разработку программного средства:

Ср = 2400 + 288 +913,92 + 10,75 + 20,75 + 40,61 + 204 + 2736 = 6614,03 (руб.).

### 6.3.10 Расходы на сопровождение и адаптацию

Сумма расходов на сопровождение и адаптацию программного средства Срса определяется как произведение суммы расходов на разработку Ср на расходы на сопровождение и адаптацию Нрса.

Таким образом, сумма расходов на сопровождение и адаптацию программного средства на сопровождение и адаптацию программного средства вычисляется по формуле (6.12):

|  |  |  |
| --- | --- | --- |
|  |  | (6.12) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму расходов на сопровождение и адаптацию программного средства:

Срса = 6614,03 ⋅ 10 / 100 = 661,4 (руб.).

### 6.3.11 Полная себестоимость

Общая сумма расходов (полная себестоимость) Сп определяется как сумма двух элементов: суммы расходов на разработку Ср и суммы расходов на сопровождение и адаптацию программного средства Срса (формула 6.13).

|  |  |  |
| --- | --- | --- |
|  | Сп = Cp + Cpca. | (6.13) |

Все данные необходимые для вычисления есть, поэтому можно определить общую сумму расходов:

Сп = 6614,03 + 661,4 = 7275,43 (руб.).

## 6.4 Выводы по разделу

В разделе были произведены экономические рacчёты, позволяющие определить затраты на разработку программного средства, a также определить прогнозируемую отпускную цену всего компьютерного программного средства.

Таблица 6.4 содержит результаты расчетов экономических показателей, проведенных в данном разделе.

Таблица 6.4 – Результаты расчетов

|  |  |
| --- | --- |
| Наименование показателя | Значение показателя |
| Время разработки, мес. | 3 |
| Количество программистов, чел. | 1 |
| Зарплата с отчислениями, руб. | 3 612,67 |
| Расходы на материалы, оплату машинного времени, прочие, руб | 265,36 |
| Накладные расходы, руб | 2 736 |
| Себестоимость разработки программного средства, руб. | 6 614,03 |
| Расходы на сопровождение и адаптацию, руб. | 661,4 |
| Полная себестоимость, руб. | 7 275,43 |
| Средняя рыночная цена разработки, руб. | 12 500 |

Так как веб-приложение предоставляется бесплатно, нет необходимости рассчитывать прибыль от реализации программного средства и рентабельность. Здесь лишь можно рассчитать сколько будет сэкономлено по сравнению с аналогичными веб-приложениями.

Средняя рыночная стоимость аналогичного продукта составляет 12500 рублей. Разработка программного средства, осуществляемая одним программистом в течении трех месяцев, при заданных условиях обойдется в 7275,43 рублей. Исходя из этого, мы получаем экономию финансовых средств в размере 5224,57 рублей.

Таким образом, веб-приложение хоть и уступает аналогам в функционале, но и по стоимости значительно дешевле.

Данное программное средство позволит пользователям легко общаться в сети в реальном времени. Разработанное веб-приложение даст возможность создавать и просматривать интересные записи, оценивать их, а также заводить новые и сохранять старые знакомства.

Программный продукт позволит выполнять следующие действия:

* редактирование профиля;
* добавление записей;
* удаление записей;
* добавление комментария к записи;
* добавление лайка записи;
* репост записи;
* поиск собеседников;
* добавление друзей;
* удаление друзей;
* общение в реальном времени.

# Заключение

Утверд.

Н. контр.

Провер.

Разраб.

*Смелов В.В.*

*Копыток Д.В.*

*Голуб Е.В.*

*Кантарович В. С.*

Ф.И.О.

Подпись

Дата

1

*Заключение*

Лист

1

*БГТУ 00.00.ПЗ*

Лит.

Листов

*БГТУ 74218007, 2021*

В ходе выполнения дипломного проекта были рассмотрены основные аналоги существующих социальных сетей.

Также были рассмотрены их преимущества и недостатки, изучены возможности рассматриваемых программных средств. Были проанализированы и выбраны основные технологии и средства для разработки дипломного проекта.

При разработке программного средства была спроектирована структура базы данных. При составлении руководства пользователя была подробно описана работа с программным средством. Также было проведено тестирование, которое показало, что разработанное в рамках дипломного проекта программное средство соответствует заданным ранее требованиям.

При рассмотрении технико-экономического обоснования разработанного проекта было рассчитано количество денежных затрат и трудозатрат на разработку программного средства.

Пользователями программного средства могут являться люди любой возрастной категории. В рамках разработанного приложения в рамках дипломного проекта пользователям доступны следующие возможности:

* вход в систему для пользователя;
* редактирование профиля;
* поиск других пользователей;
* просмотр, добавление и удаление записей;
* лайки чужих записей;
* репосты чужих записей;
* просмотр и добавление комментариев к чужим записям;
* добавление и удаление друзей;
* обмен сообщениями в реальном времени;
* просмотр и удаление бесед.

В рамках дипломного проекта было разработано программное средство, которое соответствует предъявленным требованиям и поставленным задачам.

# Список использованной литературы

Утверд.

Н. контр.

Провер.

Разраб.

*Смелов В.В.*

*Копыток Д.В.*

*Голуб Е.В.*

*Кантарович В.С.*

Ф.И.О.

Подпись

Дата

2

*Список использованной литературы*

Лист

1

*БГТУ 00.00.ПЗ*

Лит.

Листов

*БГТУ 74218007, 2021*

1. Digital 2021: Актуальная статистика и аудитория социальных сетей в мире и Беларуси [Электронный ресурс]. – Режим доступа: <https://belretail.by/article/digital-aktualnaya-statistika-i-auditoriya-sotsialnyih-setey-v-mire-i-belarusi>. – Дата доступа: 15.04.2021.
2. Социальная сеть «Twitter» [Электронный ресурс]. – Режим доступа: <https://twitter.com/>. – Дата доступа: 21.04.2021.
3. Социальная сеть «Вконтакте» [Электронный ресурс]. – Режим доступа: http://vk.com/. – Дата доступа: 22.04.2021.
4. Социальная сеть «Facebook» [Электронный ресурс]. – Режим доступа: http://www.facebook.com/. – Дата доступа: 25.04.2021.
5. Блог-платформа «LiveJournal» [Электронный ресурс]. – Режим доступа: <https://livejournal.com/> . – Дата доступа: 25.04.2021.
6. Патент US20150326523A1 [Электронный ресурс]. – Режим доступа: https://patents.google.com/patent/US20150326523A1/en?q=message+social+network&oq=message+social+network. – Дата доступа: 26.04.2021.
7. Патент US8832190B1 [Электронный ресурс]. – Режим доступа: https://patents.google.com/patent/US8832190B1/en?q=chat+social+network&oq=chat+social+network. – Дата доступа: 26.04.2021.
8. Общие сведения о веб-приложениях [Электронный ресурс]. – Режим доступа: https://helpx.adobe.com/ru/dreamweaver/using/web-applications.html. – Дата доступа: 27.04.2021.
9. Диаграммы вариантов использования [Электронный ресурс]. – Режим доступа: [https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema12/](https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema12/tema12_2). – Дата доступа: 08.05.2021.
10. Что такое Python: чем он хорошо [Электронный ресурс]. – Режим доступа: <https://netology.ru/blog/python>. – Дата доступа: 08.05.2021.
11. Python – что это такое [Электронный ресурс]. – Режим доступа: https:// mchost.ru/articles/chto-takoe-python/. – Дата доступа: 08.05.2021.
12. The Web framework for perfectionists with deadlines | Django [Электронный ресурс]. – Режим доступа: https://www.djangoproject.com/. – Дата доступа: 12.05.2021.
13. Django documentation | Django documentation | Django [Электронный ресурс]. – Режим доступа: https://docs.djangoproject.com/en/3.2/. – Дата доступа: 12.05.2021.
14. SQLite [Электронный ресурс]. – Режим доступа: https://lecturesdb.readthedocs.io/databases/sqlite.html. – Дата доступа: 14.05.2021.
15. Базы данных [Электронный ресурс]. – Режим доступа: <https://ru.hostings.info/schools/bazy-dannyh.html>. – Дата доступа: 14.05.2021.
16. Основы JavaScript [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/JavaScript_basics>. – Дата доступа: 16.05.2021.

ПРИЛОЖЕНИЕ А

Диаграмма вариантов использования

ПРИЛОЖЕНИЕ Б

Логическая схема базы данных

ПРИЛОЖЕНИЕ В

Блок-схема алгоритма отображения страницы настроек пользователя

ПРИЛОЖЕНИЕ Г

Блок-схема алгоритма добавления записи

ПРИЛОЖЕНИЕ Д

Блок-схема алгоритма создания нового сообщения

ПРИЛОЖЕНИЕ Е

Исходный код представления приложения my\_social\_net

from django.shortcuts import render, redirect

from django.http import HttpResponse

from auth\_user.models import MyUser, MyFriends

from user\_records.models import UserRecord, LikeRecord, RepostRecord, CommentRecord

from auth\_user.forms import ImageForm, SettingForm

from photo.models import UserPhoto

from chat.models import Chat

import os

from django.conf import settings

def show\_person\_page(request, person\_id):

    ''' функция для оотображения страницы пользователя.'''

    try:

        user = MyUser.objects.get(person\_id=person\_id)

        #определяем все записи пользователя

        all\_user\_records = UserRecord.objects.filter(user=user).order\_by('-date')

        user\_records = []

        #для каждой записи находим мета-информацию: количество лайков, репостов, комментариев, сами комментарии

        for record in all\_user\_records:

            count\_likes, count\_reposts, count\_comments, comments = record\_info(record)

            user\_records.append((record, count\_likes, count\_reposts, count\_comments, comments))

        if request.user.is\_authenticated:

            #если пользователь авторизован, находим его личную информацию: количество друзей, бесед, фото.

            log\_user = request.user

            my\_friends, count\_my\_friends = all\_friends(log\_user)

            if count\_my\_friends > 10:

                    my\_friends = my\_friends[:11]

            count\_my\_photos = UserPhoto.objects.filter(user = log\_user).count()

            all\_chats = len(list(Chat.objects.filter(author = log\_user).order\_by('last\_change') | Chat.objects.filter(companion = log\_user).order\_by('last\_change')))

            if user==log\_user:

                count\_new\_friends = MyFriends.objects.filter(friend=log\_user, status=0).count()

                return render(request, 'my\_page.html', {'person': user, 'log\_user': log\_user, 'count\_new\_friends':count\_new\_friends, 'person\_friends':my\_friends, 'count\_friends':count\_my\_friends, 'count\_my\_friends':count\_my\_friends, 'count\_my\_photos': count\_my\_photos, 'user\_records':user\_records, 'count\_my\_chats': all\_chats})

            else:

                #определяем является ли текущий пользователем другом авторизованного пользователя

                is\_my\_friend = False

                person\_friends, count\_friends = all\_friends(user)

                if user in my\_friends:

                    is\_my\_friend = True

                if count\_friends > 10:

                    person\_friends = person\_friends[:11]

            return render(request, 'person\_page.html', {'person': user, 'log\_user': log\_user, 'is\_my\_friend':is\_my\_friend, 'person\_friends':person\_friends, 'count\_friends':count\_friends, 'count\_my\_friends':count\_my\_friends, 'count\_my\_photos': count\_my\_photos, 'user\_records':user\_records, 'count\_my\_chats': all\_chats})

        else:

            person\_friends, count\_friends = all\_friends(user)

            if count\_friends > 10:

                    person\_friends = person\_friends[:11]

            return render(request, 'person\_page.html', {'person': user, 'person\_friends':person\_friends, 'count\_friends':count\_friends, 'user\_records':user\_records})

    except MyUser.DoesNotExist:

        return HttpResponse("Данный пользователь не зарегистрирован")

def search\_page(request):

    ''' функция для отображения страницы с результатами поиска.'''

    if request.method == 'GET':

        search\_name = request.GET["search\_name"]

        all\_users = MyUser.objects.filter(username\_\_icontains = search\_name)

        if request.user.is\_authenticated:

            log\_user = request.user

            my\_friends, count\_my\_friends = all\_friends(log\_user)

            if count\_my\_friends > 10:

                    my\_friends = my\_friends[:11]

            count\_my\_photos = UserPhoto.objects.filter(user = log\_user).count()

            all\_chats = len(list(Chat.objects.filter(author = log\_user).order\_by('last\_change') | Chat.objects.filter(companion = log\_user).order\_by('last\_change')))

            count\_new\_friends = MyFriends.objects.filter(friend=log\_user, status=0).count()

            return render(request, 'search.html', {'person': request.user, 'log\_user': log\_user, 'count\_new\_friends':count\_new\_friends, 'person\_friends':my\_friends, 'count\_friends':count\_my\_friends, 'count\_my\_friends':count\_my\_friends, 'count\_my\_photos': count\_my\_photos, 'count\_my\_chats': all\_chats, 'all\_users': all\_users})

        else:

            return render(request, 'search.html', {'all\_users': all\_users})

ПРИЛОЖЕНИЕ Ж

Исходный код шаблона base.html

 <!doctype html>

{%load static%}

<html>

<head>

    {%block head%}

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

    <!-- <link rel="shortcut icon" href="img/repost.png" type="image/png"> -->

    <link type="text/css" href="{% static 'base.css' %}" rel="stylesheet">

    <link rel="stylesheet" type="text/css" href="{% static 'my\_page.css' %}">

    <link rel="preconnect" href="https://fonts.gstatic.com/%22%3E">

    <link href="https://fonts.googleapis.com/css2?family=Ubuntu:wght@300&display=swap" rel="stylesheet">

    <script type="text/javascript" src="{% static 'js/my\_page.js' %}"></script>

    <script type="text/javascript" src="{% static 'js/base.js' %}"></script>

    <script type="text/javascript" src="{% static 'js/person\_page.js' %}"></script>

    {%endblock%}

</head>

<body>

    <div id="main-container">

        <div class="bg">

        </div>

        <div class="left-container">

            <div id="avatar">

                {%block logo\_img%}

                <div class="logo">

                    {%if person.show\_avatar%}

                    <img src="{{person.show\_avatar}}" id="avatar-img">

                    {%else%}

                    <img src="{% static 'img/avatar.png' %}" id="avatar-img">

                    {%endif%}

                </div>

                {%endblock%}

                <div id="person-information" class="logo">

                    {%block person\_description%}

                    <div class="person-description">

                        <table class="person-description-table-base" style="margin-bottom: 68px;">

                            <tr>

                                <td colspan="2">

                                    <b><a href="/person/id\_{{person.person\_id}}"

                                            style="text-align: left; color: #00BFFF; font-size: 26px;">{{person.username}}</a></b>

                                </td>

                            </tr>

                            <tr>

                                <td colspan="2" style="text-align: left; color: #6E6E70; font-weight: bolder;">

                                    {%if person.country == 'Неопределена' and person.city == 'Неопределен'%}

                                    {{person.city}}

                                    {%else%}

                                    {{person.city}}, {{person.country}}

                                    {%endif%}

                                </td>

                            </tr>

                            <tr>

                                <td colspan="2" style="text-align: left; color: #6E6E70; font-weight: bolder;">Дата

                                    регистрации: {{person.create\_date|date:"d-m-Y"}}</td>

                            </tr>

                        </table>

                    </div>

                    {%endblock%}

                </div>

            </div>

            <div class="info-line">

                {%if not user.is\_authenticated%}

                {%else%}

                <table>

                    <tr class="table-names">

                        <td id="td-person">

                            <a href="/person/id\_{{user.person\_id}}"><span>Моя страница</span></a>

                        </td>

                        <td class="my\_friends" id="td-person">

                            <a href="/my\_friends"><span>

                                    <span> Друзья <h4 class="number" id="count-fr">{{count\_my\_friends}}</h4></span>

                                    {%if count\_new\_friends%}

                                    <div class="new\_friends">

                                        +{{count\_new\_friends}}

                                    </div>

                                    {%endif%}

                                </span></a>

                        </td>

                        <td id="td-person">

                            <a href="/my\_chats"><span>

                                    <span> Беседы <h4 class="number">{{count\_my\_chats}}</h4></span>

                                </span></a>

                        </td>

                        <td id="td-person" style="margin-top: 20px;">

                            <a href="/my\_settings"><span>Мои настройки</span></a>

                        </td>

                    </tr>

                </table>

                {%endif%}

            </div>

        </div>

</body>

</html>

ПРИЛОЖЕНИЕ З

Исходный код функции my\_settings

def my\_settings(request):

    ''' функция для отображения страницы настроек пользователя'''

    if request.user.is\_authenticated:

        user = request.user

        if request.method == 'POST':

            #проверяем передаёт ли пользователь файл

            if request.FILES:

                if request.FILES['avatar']:

                    #проверяем является ли файл картинкой

                    form = ImageForm(request.POST, request.FILES)

                    if form.is\_valid():

                        person\_image = request.FILES['avatar']

                        person\_id = form.cleaned\_data['person\_id']

                        #удаляем старое фото, которое являлось аватаром пользователя

                        old\_path\_to\_avatar = os.path.join(settings.BASE\_DIR, '../media/person/user\_%s/avatar/image') % (person\_id)

                        delete\_file(old\_path\_to\_avatar)

                        #сохраняем новое фото в качестве аватара

                        user.avatar.save('image', person\_image)

                        user.save()

                        return redirect('/my\_settings')

            else:

                #меняем личные данные пользователя согласно переданным данным

                form = SettingForm(request.POST)

                if form.is\_valid():

                    person\_id = form.cleaned\_data['person\_id']

                    username = form.cleaned\_data['username']

                    city = form.cleaned\_data['city']

                    country = form.cleaned\_data['country']

                    email = form.cleaned\_data['email']

                    user.city = city

                    user.country = country

                    user.username = username

                    user.email = email

                    user.save()

                    return redirect('/my\_settings')

        return render(request, 'settings.html', {'person':user})

    else:

        return HttpResponse("Вы не авторизированы")

ПРИЛОЖЕНИЕ И

Исходный код функции new\_message

def new\_message(request):

    ''' функция для создания нового сообщения'''

    if request.user.is\_authenticated and request.method == 'POST':

        companion\_id = request.POST["companion\_id"]

        author\_id = int(request.POST["author\_id"])

        message\_text = request.POST["message\_text"]

        author = request.user

        if author\_id == request.user.person\_id:

            companion = MyUser.objects.get(person\_id = companion\_id)

        else:

            companion = MyUser.objects.get(person\_id = author\_id)

        try:

            #проверяем существует ли беседа между текущими пользователями. Так как в модели chat установлено, что author и companion должны быть уникальными, то если в текущем блоке try не возникает исключения, создаем новую беседу.

            chat = Chat(author = author, companion = companion, user\_last\_change = request.user)

            chat.save()

            print('new\_chat')

        except:

            #если беседа существует, находим её, правильно определив кто является автором, а кто собеседником

            chat = Chat.objects.filter(author = author, companion = companion)

            if not len(chat):

                chat = Chat.objects.filter(companion = author, author = companion)[0]

            else:

                chat = chat[0]

        message = ChatMessage(chat = chat, user = author, text = message\_text)

        message.save()

        return redirect(request.META["HTTP\_REFERER"])

    else:

        return HttpResponse('ВЫ не авторизировались.')