

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего
образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И
РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра комплексной информационной безопасности электронно-вычислительных систем
(КИБЭВС)

СОЦИАЛЬНАЯ СЕТЬ СО ВСТРОЕННЫМ МЕССЕНДЖЕРОМ

Курсовая работа по дисциплине «Технологии и методы программирования»

Пояснительная записка

Выполнил
Студент гр. 719-2
Горбатенко М.Д.
____.____.2021

Принял
Доцент каф. КИБЭВС, к.т.н.
оценка _____ Романов А.С.
____. ____ .2021

Томск 2021

Содержание

1 Введение	3
2 Техническое задание	4
2.1 Общие сведения	5
2.2 Назначение и цели создания системы.....	7
2.3 Характеристика объектов автоматизации	8
2.4 Требования к системе	9
2.5 Состав и содержание работ по созданию системы.....	15
2.6 Порядок контроля и приемки системы.....	16
2.7 Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие	17
2.8 Требования к документированию	18
2.9 Источники разработки.....	19
3 Проектирование программного приложения.....	21
4 Реализация программного приложения	24
4.1 Реализация	24
4.2 Запуск программы.....	25
4.3 Приложение «users».....	25
4.4 Приложение «news».....	31
4.5 Приложение «messenger»	33
5 Заключение.....	38
Приложение А.....	39
Приложение Б	47
Приложение В.....	50

1 Введение

Тема работы посвящена разработке клиент-серверного веб-приложения, которое позволяет создавать публикации, просматривать новостную ленту, общаться с другими пользователями посредством встроенного мессенджера. В роли клиента выступают страницы сайта, отображаемые в браузере, сервер же обрабатывает пользовательские данные.

Томский государственный университет систем управления и радиоэлектроники

наименование организации - разработчика ТЗ на АС

УТВЕРЖДЕНО

Д-р техн. наук,

заведующий кафедры КИБЭВС,

президент ТУСУР

Шелупанов Александр Сергеевич

Личная подпись Расшифровка подписи

Печать

Дата _____

Социальная сеть со встроенным мессенджером

наименование вида АС

Мессенджер и публикация новостей

наименование объекта автоматизации

Социальная сеть

сокращенное наименование АС

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

На 16 листах

Действует с _____

СОГЛАСОВАНО

Канд. техн. наук, доцент кафедры КИБЭВС

Романов Александр Сергеевич

Личная подпись Расшифровка подписи

Печать

Дата _____

СОГЛАСОВАНО

Студент кафедры КИБЭВС, группа 719-2

Горбатенко Матвей Дмитриевич

Личная подпись Расшифровка подписи

Печать

Дата _____

2.1 Общие сведения

2.1.1 Полное наименование системы и ее условное обозначение

Полное наименование системы – «Социальная сеть со встроенным мессенджером».

Условное обозначение – «Социальная сеть».

2.1.2 Шифр темы или шифр (номер) договора

Шифр темы и номера договора согласуется с заказчиком системы.

2.1.3 Наименование предприятий (объединений) разработчика и заказчика (пользователя) системы и их реквизиты

Заказчик: Романов Александр Сергеевич, кандидат технических наук, доцент кафедры КИБЭВС в Томском государственном университете систем управления и радиоэлектроники.

Адрес организации заказчика: г. Томск, пр. Ленина, 40.

Телефон организации заказчика: +7 (3822) 510530.

Факс организации заказчика: +7 (3822) 513262.

Разработчик: Горбатенко Матвей Дмитриевич, студент кафедры КИБЭВС (группа 719-2) в Томском государственном университете систем управления и радиоэлектроники.

Адрес организации разработчика: г. Томск, пр. Ленина, 40.

Телефон организации разработчика: +7 (913) 8042467.

E-mail разработчика: daniel.kekskeks@gmail.com.

2.1.4 Перечень документов, на основании которых создается система, кем и когда утверждены эти документ

Автоматизированная система «Социальная сеть» создается в рамках выполнения курсовой работы по дисциплине «Технологии и методы программирования» на основании учебного плана для студентов направления «Информационная безопасность» 2019 года набора.

2.1.5 Плановые сроки начала, и окончания работы по созданию системы

Начало работ: 20 сентября 2021 г.

Дата окончания работ: начало января 2021 г.

2.1.6 Сведения об источниках и порядке финансирования работ

Финансирование работ не предусмотрено.

2.1.7 Порядок оформления и предъявления заказчику результатов работ по созданию системы (ее частей), по изготовлению и наладке отдельных средств (технических, программных, информационных) и программно-технических (программно-методических) комплексов системы

Система передаётся в виде функционирующего клиент-серверного приложения и предъявляется заказчику согласно ГОСТ 19.301-79.

Результатом работы являются: исходный код всех программных модулей и разделов сайта, дампы базы данных, «Руководство пользователя», «Руководство администратора», «Руководство программиста».

Формат передачи результатов работы: файловый архив.

Результаты работы передаются заказчику поэтапно, согласно календарному плану проекта.

2.2 Назначение и цели создания системы

2.2.1 Назначение системы

Назначение системы «Социальная сеть» заключается в создании и поддержании социальных отношений между людьми (пользователями АС) и передачи между ними сообщений.

2.2.1.1 Вид автоматизируемой деятельности

В данной АС автоматизируется публикация постов пользователями и передача сообщений между ними в виде диалогов.

2.2.2 Цели создания системы

Цель создания системы «Социальная сеть» заключается в упрощении коммуникации между пользователями и сохранения конфиденциальности их переписки.

2.2.2.1 Наименования и требуемые значения технических, технологических, производственно-экономических или других показателей объекта автоматизации

Данная АС не имеет цели достижения каких-либо значений технических, технологических, производственно-экономических или других показателей.

2.3 Характеристика объектов автоматизации

2.3.1 Краткие сведения об объекте автоматизации или ссылки на документы, содержащие такую информацию

Объектом автоматизации является клиент-серверное веб-приложение «Социальная сеть». Данное приложение будет выполнять следующие обязанности:

- регистрировать новых пользователей;
- показывать публикации пользователей;
- передавать сообщения между пользователями во встроенном мессенджере;
- хранить все переписки пользователей в зашифрованном виде (шифрование на стороне сервера);
- выполнять поиск пользователей для добавления их в контакты и дальнейшей переписки с ними.

Документ, содержащий подробную информацию об объекте автоматизации, есть «Руководство пользователя», и прилагается к отчету по курсовой работе.

2.3.2 Сведения об условиях эксплуатации объекта автоматизации и характеристиках окружающей среды

Данная АС будет работать на стороннем сервере. Сторонние ресурсы для поддержания и обслуживания программных средств не требуются, за исключением выхода в Интернет. Пользователи будут иметь доступ к АС через свои персональные компьютеры с помощью веб-браузера. Документ, содержащий информацию о серверной части АС, есть «Руководство администратора».

2.4 Требования к системе

2.4.1 Требования к системе в целом

2.4.1.1 Требования к численности и квалификации персонала системы

Пользователи АС имеют возможность зарегистрироваться в веб-приложении через страницу регистрации. В таком случае число пользователей может быть не ограничено. Пользователи системы должны владеть базовыми знаниями и навыками использования компьютера и браузера. Взаимодействие с системой происходит через пользовательский интерфейс.

За обеспечение работоспособности системы отвечает администратор. Вместе с АС прилагается «Руководство администратора», содержащее всю информацию для работы администратора.

Конечный пользователь системы может использовать ее в своих целях, не противоречащих законодательству Российской Федерации.

2.4.1.2 Требования к эргономике технической эстетике системы

Система должна обеспечивать корректную обработку исключительных ситуаций, вызванных неверными действиями пользователей, неверным форматом или недопустимыми значениями входных данных. В указанных случаях система должна выдавать пользователю соответствующие сообщения, после чего возвращаться в рабочее состояние, предшествовавшее недопустимой команде или некорректному вводу данных.

Все экранные формы пользовательского интерфейса должны быть выполнены в едином графическом дизайне, с одинаковым расположением основных элементов управления и навигации.

Система должна иметь человеко-машинный интерфейс, удовлетворяющий следующим требованиям:

- взаимодействие системы и пользователя должно осуществляться на русском языке, за исключением системных сообщений, не подлежащих русификации;
- при работе с интерфейсом пользователь должен быть ориентирован на работу с клавиатурой и манипулятором графической информации «мышь»;
- должно быть реализовано отображение на экране только тех возможностей, которые доступны конкретному пользователю в соответствии с его функциональной ролью в системе.

2.4.1.3 Требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы

Условия эксплуатации, а также виды и периодичность обслуживания технических средств Системы должны соответствовать требованиям по эксплуатации, техническому обслуживанию, ремонту и хранению, изложенным в документации завода-изготовителя (производителя) на них.

Технические средства Системы и персонал должны размещаться в существующих помещениях Заказчика, которые по климатическим условиям должны соответствовать ГОСТ 15150-69 «Машины, приборы и другие технические изделия. Исполнения для различных климатических районов. Категории, условия эксплуатации, хранения и транспортирования в части воздействия климатических факторов внешней среды» (температура окружающего воздуха от 5 до 40 °С, относительная влажность от 40 до 80 % при T=25 °С, атмосферное давление от 630 до 800 мм ртутного столба). Размещение технических средств и организация автоматизированных рабочих мест должны быть выполнены в соответствии с требованиями ГОСТ 21958-76 «Система "Человек-машина". Зал и кабины операторов. Взаимное расположение рабочих мест. Общие эргономические требования».

Для электропитания технических средств должна быть предусмотрена трехфазная четырехпроводная сеть с глухо заземленной нейтралью 380/220 В (+10-15)% частотой 50 Гц (+1-1) Гц. Каждое техническое средство запитывается однофазным напряжением 220 В частотой 50 Гц через сетевые розетки с заземляющим контактом.

Для обеспечения выполнения требований по надежности должен быть создан комплект запасных изделий и приборов (ЗИП).

Для поддержания работы веб-приложения необходимы следующие характеристики:

- операционная система – Windows 10 и выше;
- процессор не старше 5 лет;
- оперативная память не менее 4 гигабайт.

Сервером веб-приложения должен являться фреймворк Django (свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC). Для взаимодействия с серверной частью используется СУБД - SQLite.

2.4.1.4 Требования к защите от несанкционированного доступа

АС «Социальная сеть» обеспечит безопасность сбора, сохранения и передачи личных сведений с использованием современных надежных методов шифрования (например, протокола HTTPS).

Пользователь веб-приложения будет иметь возможность полностью удалить свои персональные данные с сервера разработчика. В случае удаления учетной записи пользователя, вся пользовательская информация будет удалена с сервера разработчика веб-приложения. Для использования веб-приложения требуется обязательная регистрация в системе.

Контент веб-приложения соответствует обозначенному возрастному рейтингу и соответствующим соглашениям, нормам, правилам и положениям законов.

2.4.1.5 Требования безопасности

Для того, чтобы сохранить конфиденциальность информации применяется строгая аутентификация и авторизация, дополнительно будет применено хэширование пароля при регистрации.

Все сообщения сохраняются в базе данных в зашифрованном виде. Шифрование происходит на стороне сервера и сообщения заносятся в базу данных.

2.4.2 Требования к функциям (задачам), выполняемым системой

Данная АС предусматривает регистрацию пользователей с помощью логина и пароля. Так же необходимыми при регистрации будут поля «Имя» и «Фамилия». При регистрации будет создаваться личный кабинет пользователя, который будет отображаться на отдельной странице веб-приложения.

Данная АС должна:

- предоставлять доступ к авторизации в системе посредством использования логина и пароля.
- выводить основную информацию о пользователе в личном кабинете пользователя.
- предоставлять доступ к редактированию собственного профиля пользователя.
- предоставлять доступ к просмотру профиля пользователя другими пользователями.

- иметь систему сохранения контактов пользователем для дальнейшего взаимодействия с ними.
- предоставлять доступ к мессенджеру для отправки сообщений между пользователями. Предусматривается переписка в виде диалога (в котором будет обмен сообщениями между двумя пользователями).
- хранить переписку пользователей в зашифрованном виде в базе данных.

Данная АС предусматривает публикацию пользователями постов, содержащих информацию в виде текста или текста с одним изображением.

2.4.3 Требования к информационному обеспечению системы

2.4.3.1 Требования к составу, структуре и способам организации данных в системе

В состав АС «Социальная сеть» должны входить следующие подсистемы:

- подсистема хранения данных (SQLite);
- подсистема управления интерфейсом;
- подсистема передачи сообщений между пользователями;
- подсистема регистрации и авторизации;
- подсистема публикации постов;
- подсистема управление профилем.

Подсистема хранения данных предназначена для сохранения личной информации о пользователях, данных об их контактах, сохранения переписки.

Подсистема управления интерфейсом необходима для удобного использования в браузере на персональном компьютере пользователя.

Подсистема передачи сообщений между пользователями представляет из себя встроенный мессенджер с возможностью отправки сообщений и просмотра переписки.

Подсистема регистрации и авторизации предназначена для регистрации пользователей в системе и сохранения в базе данных информации для последующей авторизации.

Подсистема публикации постов предназначена для создания пользователями публикации и загрузки их в общий доступ.

Подсистема управление профилем предназначена для изменения информации о пользователе.

Структура АС «Социальная сеть» должна будет состоять из следующих страниц: «регистрация», «авторизация», «личный кабинет», «редактирование профиля», «контакты», «мессенджер», «новости».

Страница «авторизация» будет содержать поля ввода логина и пароля, ссылку перехода на страницу регистрации.

Страница «регистрация» будет содержать необходимую новому пользователю информацию об АС «Социальная сеть», поля ввода логина, пароля и повторного ввода пароля. Так же на странице будет кнопка перехода к завершению регистрации.

Страница «личный кабинет» должна отображать основную информацию о пользователе: имя, фамилию, логин, дополнительную информацию.

Страница «редактирование профиля» должна предоставлять пользователям возможность изменения своих данных (пароль, имя, фамилию, аватара и дополнительную информацию).

Страница «контакты» будет отображать список добавленных контактов.

Страница «мессенджер» будет содержать список диалогов пользователя и кнопки перехода к этим диалогам.

Страница «новости» будет отображать публикации пользователя и его контактов в виде новостной ленты. Также на главной странице будут кнопки перехода к личному кабинету пользователя, списку контактов, мессенджеру.

2.4.3.2 Требования к информационному обмену между компонентами системы

Информационный обмен между подсистемами должен осуществляться через единое информационное пространство и посредством использования стандартизированных протоколов и форматов обмена данными.

Все компоненты системы должны функционировать в пределах единого логического пространства, обеспеченного интегрированными средствами серверов данных и серверов приложений.

В состав информационного обеспечения приложения входит база данных SQLite.

2.4.3.3 Требования по применению систем управления базами данных

Система управлениями базами данных – SQLite.

2.4.3.4 Требования к структуре процесса сбора, обработки, передачи данных в системе и представлению данных

В структуре процесса сбора, обработки и передачи данных в системе и представлении данных должны использоваться следующие инструменты, средства и технологии:

- Docker Compose;
- язык программирования Python (с использованием фреймворка Django);
- язык программирования JavaScript (с использованием фреймворка Bootstrap);
- язык гипертекстовой разметки HTML;
- язык каскадных таблиц стилей CSS.

2.4.3.5 Требования к программному обеспечению

Для успешного функционирования системы у конечного пользователя должно присутствовать следующее ПО:

- программа Docker (последней версии);
- веб-браузер (Google Chrome последней версии).

Подробные требования к программному обеспечению конечного пользователя системы прописываются в «Руководстве пользователя», которая прилагается к отчету по курсовой работе.

2.4.4 Требования к методическому обеспечению

Методическое обеспечение представляет собой «Руководство пользователя», «Руководство администратора», «Руководство программиста» прилагаемые к отчету по курсовой работе, и содержит все требования к программному обеспечению конечного пользователя системы.

2.5 Состав и содержание работ по созданию системы

Перечень документов, предъявляемых по окончании соответствующих стадий по созданию системы, представлен в таблице 1.

Разработка системы предполагается по учебному календарному плану, приведенному в таблице 1.

Таблица 1 – Календарный план работ по созданию программного продукта (веб-приложения «Социальная сеть»)

Этап создания системы	Сроки выполнения	Результаты работ
Разработка ТЗ и согласование автоматизированной системы ГОСТ 34.602-289	20.09.21 – 09.11.21	Техническое задание на создание автоматизированной системы
Проектирование приложения (UML + ER модели + описание бизнес-процесса)	10.11.21 – 22.11.21	Описание программного продукта, описание БД
Разработка приложения	22.11.21 – 22.12.21	Впервые реализованное приложение
Тестирование и отладка приложения	20.12.21 – 30.12.21	Протокол испытаний, устранение неполадок, внесение изменений в документацию
Рабочая документация Разработка рабочей документации на систему и её части. Разработка или адаптация программ	31.12.21 - 03.01.22	Готовая версия ПП. Документация на ПП. Руководство администратора. Руководство программиста. Руководство пользователя.

2.6 Порядок контроля и приемки системы

Сдача-приёмка работ производится поэтапно, в соответствии с рабочей программой и календарным планом, представленном в разделе 5.

Приемка этапа заключается в рассмотрении и оценке проведенного объема работ и предъявленной технической документации в соответствии с требованиями настоящего технического задания.

Ответственность за организацию и проведение приемки системы должен нести заказчик. Приемка системы должна производиться по завершению приемки всех задач системы.

Сдача-приемка осуществляется комиссией, в состав которой входят представители Заказчика и Исполнителя. По результатам приемки подписывается акт приемочной комиссии.

Все создаваемые в рамках настоящей работы программные изделия (за исключением покупных) передаются Заказчику, как в виде готовых модулей, так и в виде исходных кодов, представляемых в электронной форме на стандартном машинном носителе.

2.7 Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие

Для обеспечения готовности веб-приложения к вводу системы в действие необходимо провести комплекс мероприятий:

- приобрести компоненты технического и программного обеспечения (если таковые требуются), заключить договора на их лицензионное использование;
- настройка всех модулей и подсистем АС;
- проверить системные требования к техническому обеспечению.

2.8 Требования к документированию

Проектная документация должна быть разработана в соответствии с ГОСТ 34.201-89 и ГОСТ ЕСПД.

Отчетные материалы должны включать в себя текстовые материалы (представленные в виде бумажной копии и на цифровом носителе в формате MS Word) и графические материалы.

Предоставить документы:

- описание автоматизируемых функций;
- схема функциональной структуры автоматизируемой деятельности;
- описание технологического процесса обработки данных;
- описание информационного обеспечения;
- описание программного обеспечения АС;
- схема логической структуры БД;
- руководство пользователя;
- руководство программиста;
- руководство администратора.

2.9 Источники разработки

Первоначальными источниками разработки являются:

- 1) ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.
- 2) ГОСТ 34.601-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания.
- 3) ГОСТ 34.201-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначение документов при создании автоматизированной системы.
- 4) РД 50-34.698-90. Методические указания. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Требования к содержанию документов.
- 5) ГОСТ 2.105-95. ЕСКД. Общие требования к текстовым документам.
- 6) Пример по оформлению ТЗ. [Электронный ресурс] – Режим доступа: https://studbooks.net/2110393/informatika/trebovaniya_programmnoy_dokumenta_tsii#94 (Дата обращения: 27.09.2021)
- 7) Документация Django на русском языке – <https://djbook.ru/rel3.0/>
- 8) Документация JavaScript на русском языке – <https://developer.mozilla.org/ru/docs/Web/JavaScript>
- 9) Документация Bootstrap на русском языке – <https://bootstrap-4.ru/>

(код ТЗ)

СОСТАВИЛИ

Наименование организации, предприятия	Должность исполнителя	Фамилия, имя, отчество	Подпись	Дата
Томский государственный университет систем управления и радиоэлектроники	Студент	Горбатенко Матвей Дмитриевич		

СОГЛАСОВАНО

Наименование организации, предприятия	Должность исполнителя	Фамилия, имя, отчество	Подпись	Дата
Томский государственный университет систем управления и радиоэлектроники	Канд. техн. наук, доцент кафедры КИБЭВС	Романов Александр Сергеевич		

3 Проектирование программного приложения

3.1 Диаграмма вариантов прецедентов

На рисунке 3.1 представлена диаграмма прецедентов.

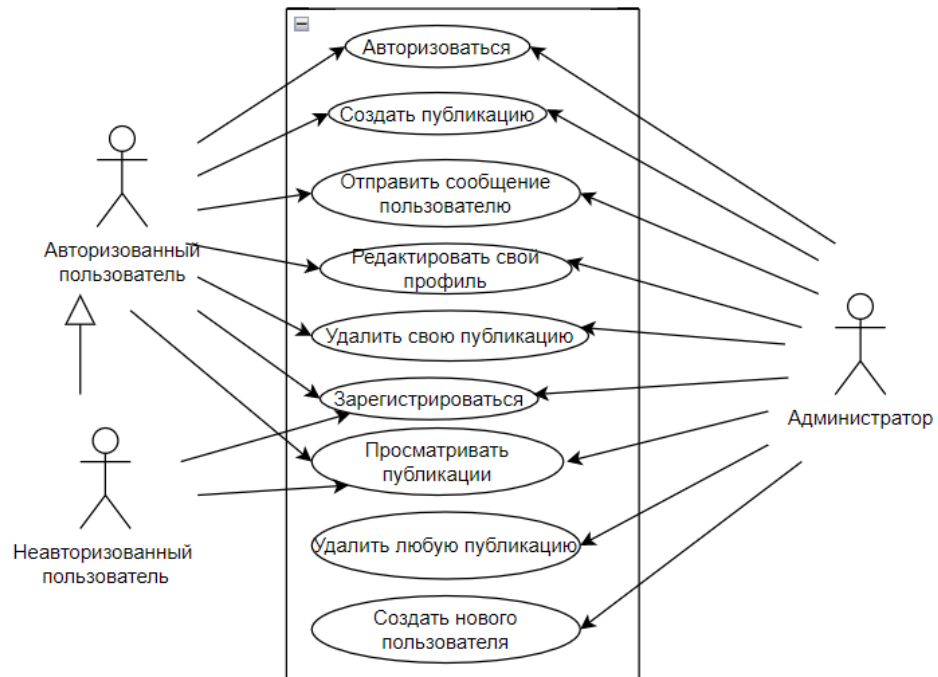


Рисунок 3.1 – Диаграмма прецедентов

3.2 Диаграмма классов

В реализованном приложении присутствуют следующие сущности, необходимые для наглядного представления их взаимодействия друг с другом: пользователь (User), профиль пользователя (Profile), контакты пользователей (Contact), публикации пользователей (Publication) и сообщения пользователей (Message).

На рисунке 3.2 представлена UML-диаграмма классов.

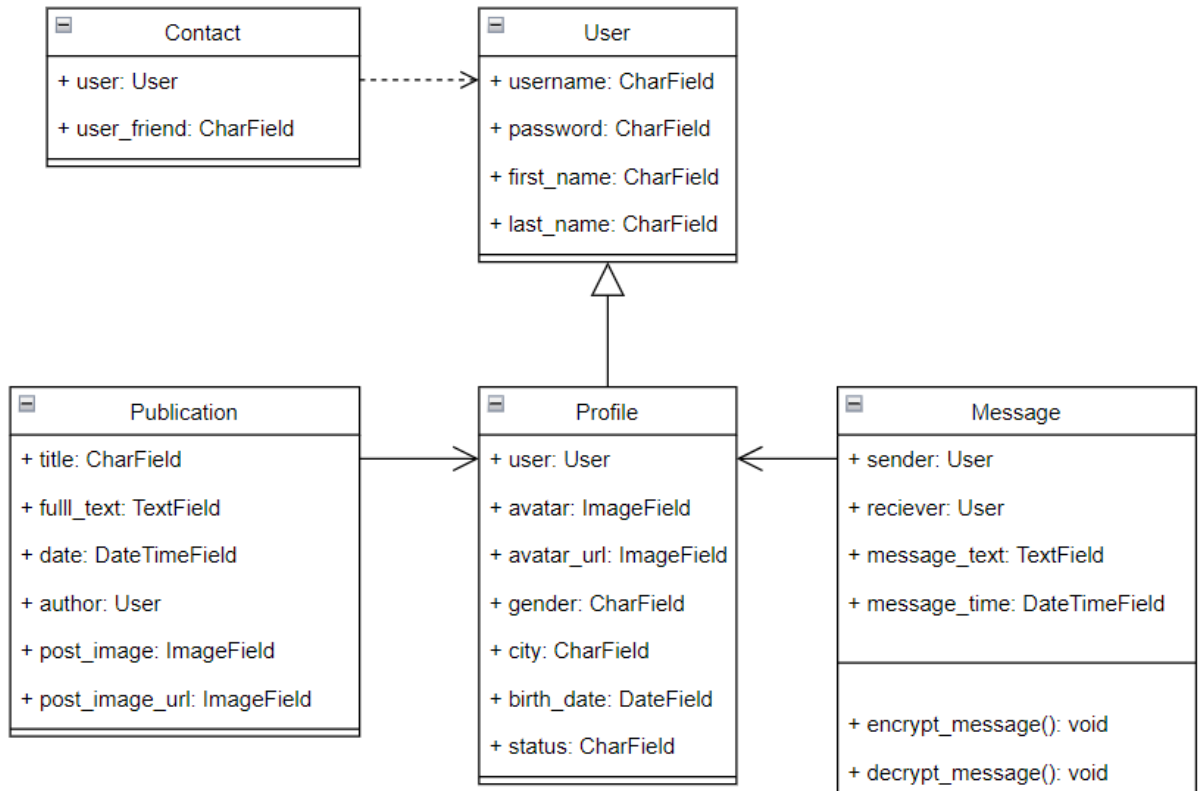


Рисунок 3.2 – Диаграмма классов

3.3 Реляционная модель базы данных

В данном веб-приложении таблицы базы данных создаются на основе классов, представленных пунктом выше. Таблица User связана с таблицами Publication и Message связями отношением один ко многим. Таблица User связана с таблицей Contact отношением многие ко многим. Связь между таблицами User и Profile один к одному.

На рисунке 3.3 представлена реляционная модель базы данных.

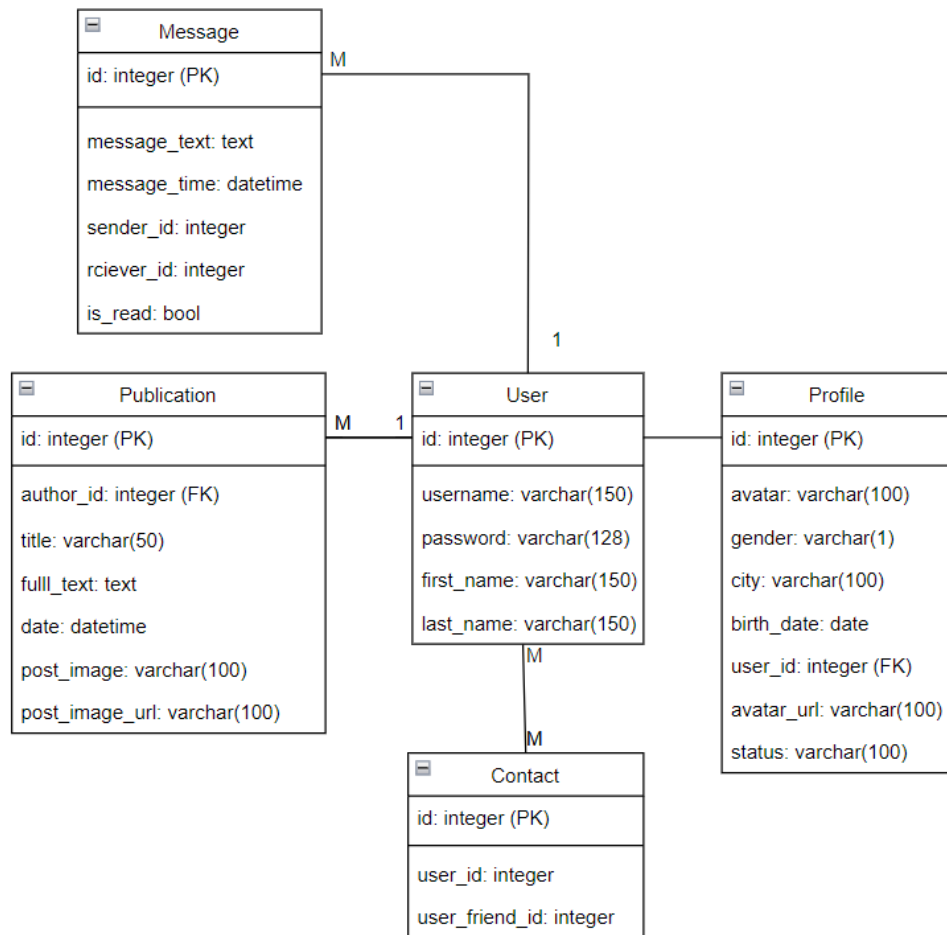


Рисунок 3.3 – Реляционная модель базы данных

4 Реализация программного приложения

4.1 Реализация

Все веб-приложение написано с использованием фреймворка «Django», языков программирования «Python» и «JavaScript», языка гипертекстовой разметки «HTML» и языка каскадных таблиц стилей «CSS».

Проекты «Django» состоят из отдельных приложений, которые выполняют определенный функционал всего веб приложения. В данной работе были определены следующие приложения: «main», «users», «news», «messenger».

Приложение «main» отвечает за базовые основы проекта, такие как: базовые шаблоны страниц, начальные страницы сайта, базовая маршрутизация сайта.

Приложение «users» отвечает за подсистему регистрации и авторизации на сайте, просмотра страниц пользователей, редактирования персональной страницы, просмотра, добавления и удаления контактов.

Приложение «news» отвечает за подсистему публикаций. Здесь реализована возможность просмотра новостей пользователями, их создания.

Приложение «messenger» отвечает за подсистему использования встроенного мессенджера (просмотр диалогов, ведения переписки).

Первым делом были определены какие страницы будет иметь приложение и написание их макетов. По этим макетам производилась верстка всех страниц, для них был определен дизайн и подобрана цветовая схема.

Далее была разработана подсистема регистрации и авторизации на сайте. Использовалась встроенная система авторизации фреймворка с добавлением форм на страницы и обработки полученных данных на сервере.

Все данные с сервера передаются по HTTP-протоколу.

Взаимодействие с базой данных реализовано внутренними средствами фреймворка.

4.2 Запуск программы

Структура всего веб-приложения представлена на рисунке 4.1.

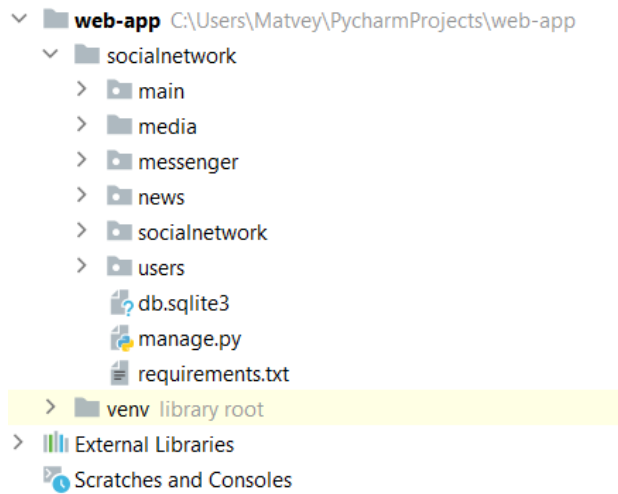


Рисунок 4.1 – Структура веб приложения

Весь проект находится в папке «web-app», внутри которой расположено приложение Django, в папке «socialnetwork». Внутри этой папки располагаются подсистемные приложения, описанные пунктом выше, папка «media», в которой хранится медиа контент проекта. С помощью файла «manage.py» происходит запуск сервера с помощью специальной команды. Запуск сервера представлен на рисунке 4.2.

```
PS C:\Users\Matvey\PycharmProjects\web-app> cd socialnetwork
PS C:\Users\Matvey\PycharmProjects\web-app\socialnetwork> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
January 26, 2022 - 21:13:14
Django version 3.2.9, using settings 'socialnetwork.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Рисунок 4.2 – Запуск сервера

4.3 Приложение «users»

Структура данного приложения представлена на рисунке 4.3. Для всех остальных приложений структура аналогична.

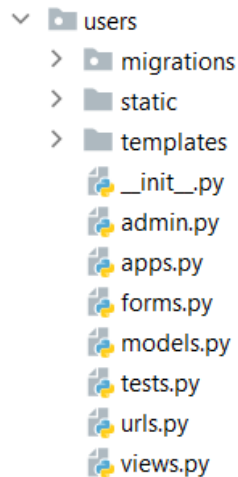


Рисунок 4.3 – Структура «users»

В данном приложении расписан функционал, связанный с пользователями, а именно: регистрация, авторизация, отображение личного кабинета и профилей других пользователей, списки контактов, страница редактирования профиля.

В каждом приложении есть две папки – «static» и «templates». В папке «static» находятся css файлы и скрипты на языке «JavaScript». В папке «templates» находятся html-шаблоны для страниц приложения. Так же в каждом приложении есть такие файлы как: admin, apps, forms, models, tests, urls и views.

В файлах forms находятся модели форм, которые представлены на страницах сайта, вроде форм регистрации или редактирования профиля. В данном приложении присутствует две формы: форма регистрации и форма редактирования профиля. Для авторизации используется стандартная форма, входящая во фреймворк, поэтому здесь нет ее реализации.

Ниже приведем пример кода для формы регистрации:

```
class UserRegisterForm(UserCreationForm):
    username = forms.CharField(label='Логин', required=True)
    first_name = forms.CharField(label='Имя', required=True)
    last_name = forms.CharField(label='Фамилия', required=True)

    error_messages = {
        'duplicate_username': "Пользователь с таким именем уже существует",
        'password_mismatch': "Введенные пароли не совпадают",
    }

    field_order = ['username', 'first_name', 'last_name', 'password1',
                  'password2']
```

В форме указываются те поля, которые будут приниматься со стороны клиента и заноситься в базу данных. В случае с формой регистрации это логин, имя и фамилия пользователя, пароль.

Пример форм приложения представлен на рисунке 4.4.

Регистрация

Логин:

Имя:

Фамилия:

Пароль:

Подтверждение пароля:

Зарегистрироваться

Рисунок 4.4 – Отображение формы регистрации на странице

В файлах `models` представлены модели объектов веб-приложения, также служат для создания таблиц на основе таких моделей.

Для данного приложения были реализованы две модели: «Profile» и «Contact». Модель «Profile» нужна для хранения дополнительной информации о пользователях, такой как родной город, пол или статус. Модель «Contact» нужна для хранения списка контактов для каждого пользователя.

Код модели профиля пользователя:

```
class Profile(models.Model):
    GENDER_CHOICE = (
        ("М", "М"),
        ("Ж", "Ж"),
        (None, "-")
    )

    user = models.OneToOneField(User, on_delete=models.CASCADE)
    avatar = models.ImageField('Аватар пользователя', blank=True,
                               upload_to='images/avatar/')
    avatar_url = models.ImageField('URL аватара пользователя', blank=True)

    gender = models.CharField('Пол', max_length=1, choices=GENDER_CHOICE,
                              blank=True)
    city = models.CharField('Город', max_length=100, blank=True)
    birth_date = models.DateField('Дата рождения', null=True, blank=True)

    status = models.CharField('Статус', max_length=100, blank=True)

    def __str__(self):
        return str(self.user)

class Meta:
    verbose_name = 'Профиль'
    verbose_name_plural = 'Профили'
```

На основе этих моделей создаются таблицы в базе данных. На рисунках 4.5-4.6 представлены структуры таблиц для данных моделей.

users_contact			CREATE TABLE "users_contact" ("id"
id	integer	"id" integer NOT NULL	
user_id	integer	"user_id" integer NOT NULL	
users_friend_id	integer	"users_friend_id" integer NOT NULL	

Рисунок 4.5 – Таблица модели «Contact»

users_profile			CREATE TABLE "users_profile" ("id"
id	integer	"id" integer NOT NULL	
avatar	varchar(100)	"avatar" varchar(100) NOT NULL	
gender	varchar(1)	"gender" varchar(1) NOT NULL	
city	varchar(100)	"city" varchar(100) NOT NULL	
birth_date	date	"birth_date" date	
user_id	integer	"user_id" integer NOT NULL UNIQUE	
avatar_url	varchar(100)	"avatar_url" varchar(100) NOT NULL	
status	varchar(100)	"status" varchar(100) NOT NULL	

Рисунок 4.6 – Таблица модели «Profile»

Эти данные отображаются на странице пользователей (клиенте).

На рисунке 4.7 представлен личный кабинет пользователя.



Рисунок 4.7 – Личный кабинет пользователя

На страницах пользователей отображается информация об их аккаунте, а также их публикации. На странице присутствует кнопка перехода к редактированию страницы. Редактор профиля представлен на рисунке 4.8.

Рисунок 4.8 – Редактор профиля

При переходе на страницу других пользователей появляются дополнительные кнопки: перехода к диалогу и добавление в контакты пользователя. Страница другого пользователя представлена на рисунке 4.9.

Рисунок 4.9 – Страница пользователя

При добавлении пользователя в контакты, он будет отображаться в списке контактов. На рисунке 4.10 представлен список контактов для авторизованного пользователя. Код функции добавления пользователя в контакты:

```
def add_contact(request, account_username):
    try:
        user = User.objects.get(username=account_username)
    except:
        raise Http404("Пользователь не найден!")

    is_friend = Contact.objects.filter(user=request.user, users_friend=user)

    if not is_friend:
        add_contact = Contact(user=request.user, users_friend=user)
        add_contact.save()
    return HttpResponseRedirect(reverse('users:contacts'))
```

В результате выполнения данной функции пользователь добавляется в список контактов. При этом пользователь перенаправляется на страницу со списком контактов.

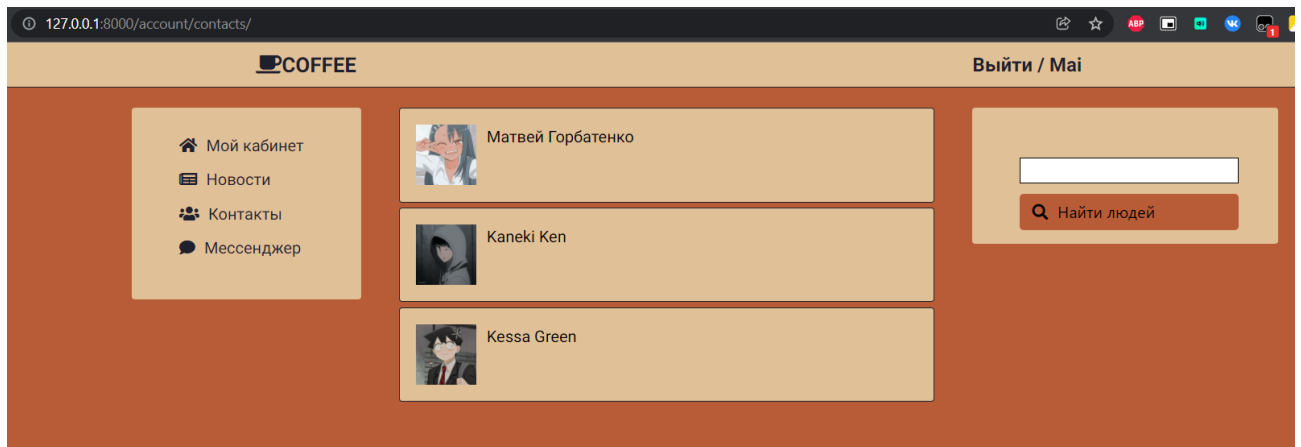


Рисунок 4.10 – Список контактов

В приложении реализован поиск пользователей. При нажатии на кнопку «Найти людей» не вводя имя интересующего пользователя в соответствующее поле, будет отображен список всех зарегистрированных пользователей. При этом в поле поиска можно вводить как имя, так и фамилию. Для этого список фильтруется по именам и фамилиям, введенным в поле для поиска.

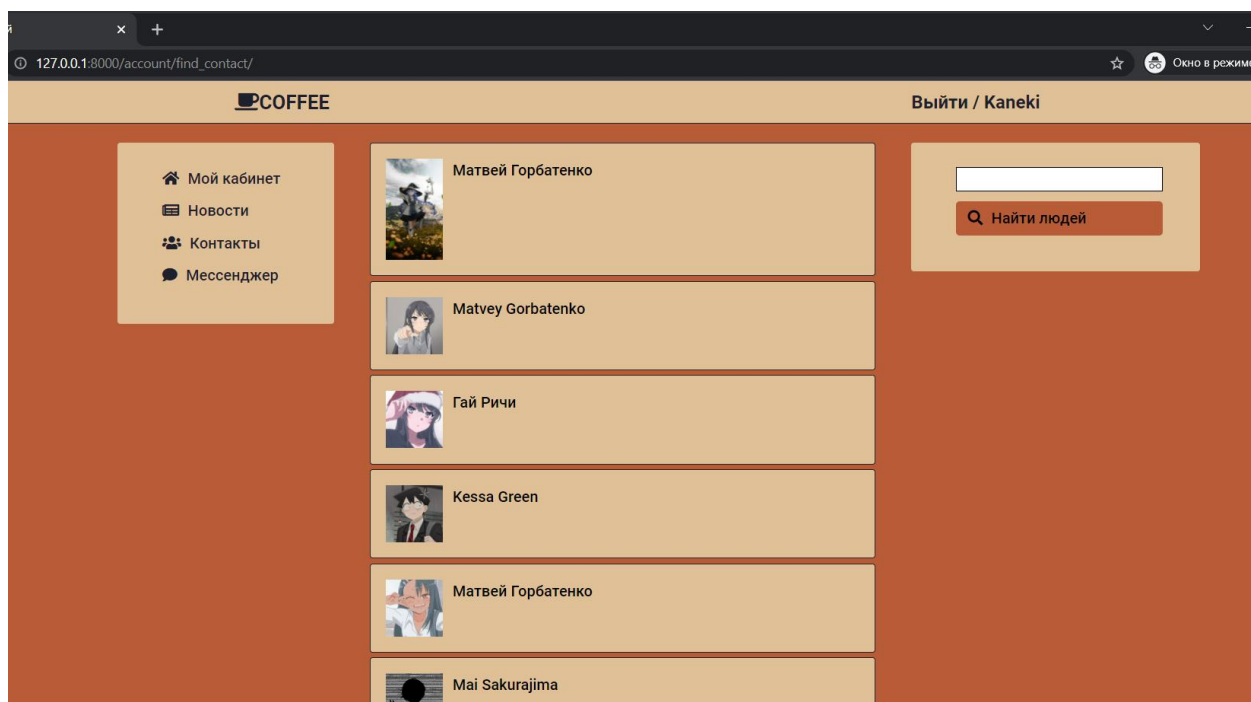


Рисунок 4.11 – Поиск пользователей

4.4 Приложение «news»

Публикации хранятся в базе данных, в таблице, созданной на основе модели «Publication». Запись в этой таблице содержит следующие поля: заголовок, текст публикации, дата публикации, автор, изображение. Структура таблицы на основе модели представлена на рисунке _.

news_publication			CREATE TABLE "news_publication" ("id" in
id	integer		"id" integer NOT NULL
title	varchar(50)		"title" varchar(50) NOT NULL
full_text	text		"full_text" text NOT NULL
date	datetime		"date" datetime NOT NULL
post_image	varchar(100)		"post_image" varchar(100) NOT NULL
post_image_url	varchar(100)		"post_image_url" varchar(100) NOT NULL
author_id	integer		"author_id" integer NOT NULL

Рисунок 4.12 – Таблица модели «Publication»

На рисунке 4.13 представлена новостная лента приложения.



Рисунок 4.13 – Новостная лента приложения

На странице с новостной лентой есть кнопка перехода к редактору публикаций. На рисунке 4.14 представлен редактор.

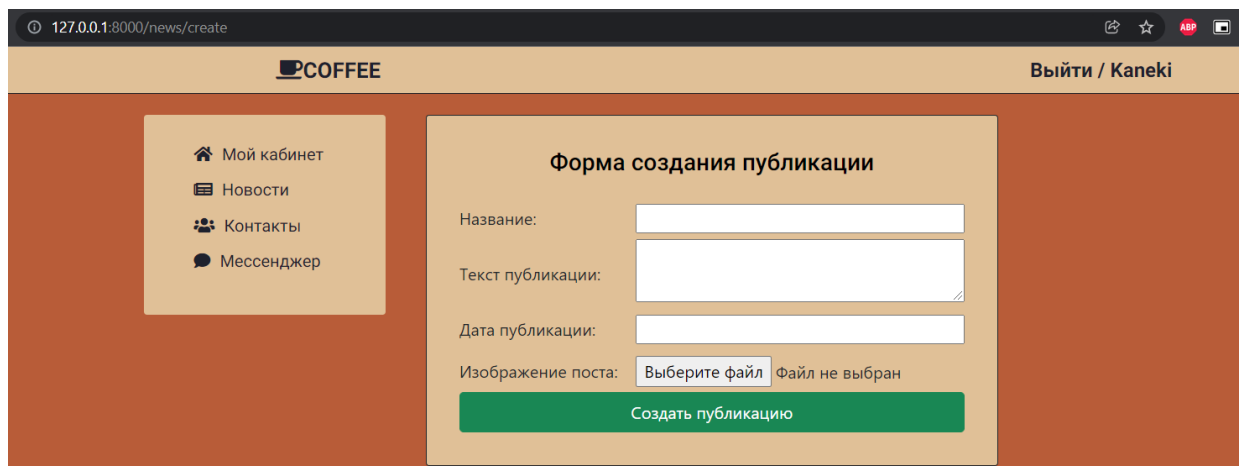


Рисунок 4.14 – Форма создания публикаций

При создании публикации вызывается функция, которая принимает данные из формы и заносит их в базу данных. После чего происходит перенаправление на страницу с новостями, где будет отображена новая публикация.

4.5 Приложение «messenger»

Для сообщений также определена модель. В коде она называется «Message», структура таблицы для данной модели представлена на рисунке 4.15.









messenger_message		CREATE TABLE "messenger_message"
 id	integer	"id" integer NOT NULL
 message_text	text	"message_text" text NOT NULL
 message_time	datetime	"message_time" datetime NOT NULL
 sender_visibility	bool	"sender_visibility" bool NOT NULL
 reciever_visibility	bool	"reciever_visibility" bool NOT NULL
 reciever_id	integer	"reciever_id" integer NOT NULL
 sender_id	integer	"sender_id" integer NOT NULL
 is_readed	bool	"is_readed" bool NOT NULL

Рисунок 4.15 – Таблица модели «Message»

Таблица содержит в себе следующие поля: id сообщения, пользователь-отправитель, пользователь-получатель, текст сообщения, время отправления, статус (прочитано или нет). В базе данных текст сообщения хранится в зашифрованном виде. Шифрование происходит на стороне сервера, с помощью алгоритма RSA. При отправке сообщения, текст принимает функция через POST запрос, которая перед тем, как занести сообщения в базу данных шифрует его данным алгоритмом, после чего сохраняет в базе данных. Для одновременного отображения сообщений у обоих пользователей диалога происходит постоянный обмен данными между сервером и клиентом, проверяя наличие новых сообщений. Данная функция реализована с помощью технологии «ајах» (асинхронного обмена данными), данные передаются с помощью формата обмена данными «json».

Код отправки сообщения:

```
def leave_message(request, reciever_name):
    reciever_user = User.objects.get(username = reciever_name)
    if request.method == 'POST':
        message_text = request.POST['message_text']
        print(message_text)

        a = Message(sender = request.user, reciever = reciever_user,
message_text = message_text, message_time = timezone.now())
        a.encrypt_message(message_text)
        a.save()
        messages = Message.objects.filter(id = a.id)
        for el in messages:
            el.decrypt_message()

    context = {'messages': messages, 'user': request.user}
    if messages:
        return HttpResponse(
            json.dumps({
```

```

        "result": True,
        "messages_list":
render_to_string('messenger/dialog_messages_block.html', context),
    }),
    content_type="application/json")
else:
    return HttpResponse(
        json.dumps({
            "result": False,
        }),
        content_type="application/json")

```

Пользователь, получающий сообщения принимает GET запрос, в котором предварительно на стороне сервера была произведена расшифровка сообщения. При этом происходит постоянная проверка новых сообщений, путем проверки параметра `is_read`. Если пришло новое сообщения, параметр `is_read` меняется на `true` и сообщение отправляется клиенту.

```

def post(request, username):
    companion = User.objects.get(username = username)
    messages = Message.objects.filter(reciever = request.user, sender =
companion, is_readed = False)
    for message in messages:
        message.is_readed = True
        message.save()

    for message in messages:
        message.decrypt_message()
    context = {'messages': messages, 'user': request.user}
    if messages:
        return HttpResponse(
            json.dumps({
                "result": True,
                "messages_list":
render_to_string('messenger/dialog_messages_block.html', context),
            }),
            content_type="application/json"
        )
    else:
        return HttpResponse(
            json.dumps({
                "result": False,
            }),
            content_type="application/json"
        )

```

Пример хранения сообщений представлен на рисунке 4.16.

Таблица: messenger_message

	id	message_text	message_time	sender_visi
	Фи...	Фильтр	Фильтр	Фильтр
1	140	246177 501017 210943 683125 828645 206172 361869	2022-01-23 11:51:41.070868	1
2	141	246177 501017 210943 683125 828645 206172 85526	2022-01-23 11:51:57.204730	1
3	142	361071 254069 56028 927172 206172 683125 349864 210943 927172 749120 828645 680974 254069 43407...	2022-01-23 11:52:11.346813	1
4	143	246177 501017 210943 683125 828645 206172 361869	2022-01-26 07:44:55.490828	1
5	144	246177 501017 210943 683125 828645 206172 210943 56028 210943 361869	2022-01-26 07:45:31.006255	1
6	145	529920 440530 349864 749120 210943 760788 927172 383235 828645 522430 349864 749120 1010828 ...	2022-01-26 07:45:46.251554	1
7	146	146515 254069 415451 927172 56028 349864 1010828 828645 146105 1010828 349864 85526	2022-01-26 07:45:55.490131	1

Рисунок 4.16 – Пример хранения сообщений

При переходе на страницу мессенджера пользователь видит список контактов, с которыми может переписываться. При переходе к переписке, открывается окно мессенджера, в котором отображаются сообщения диалога, место ввода сообщения и кнопка отправки.

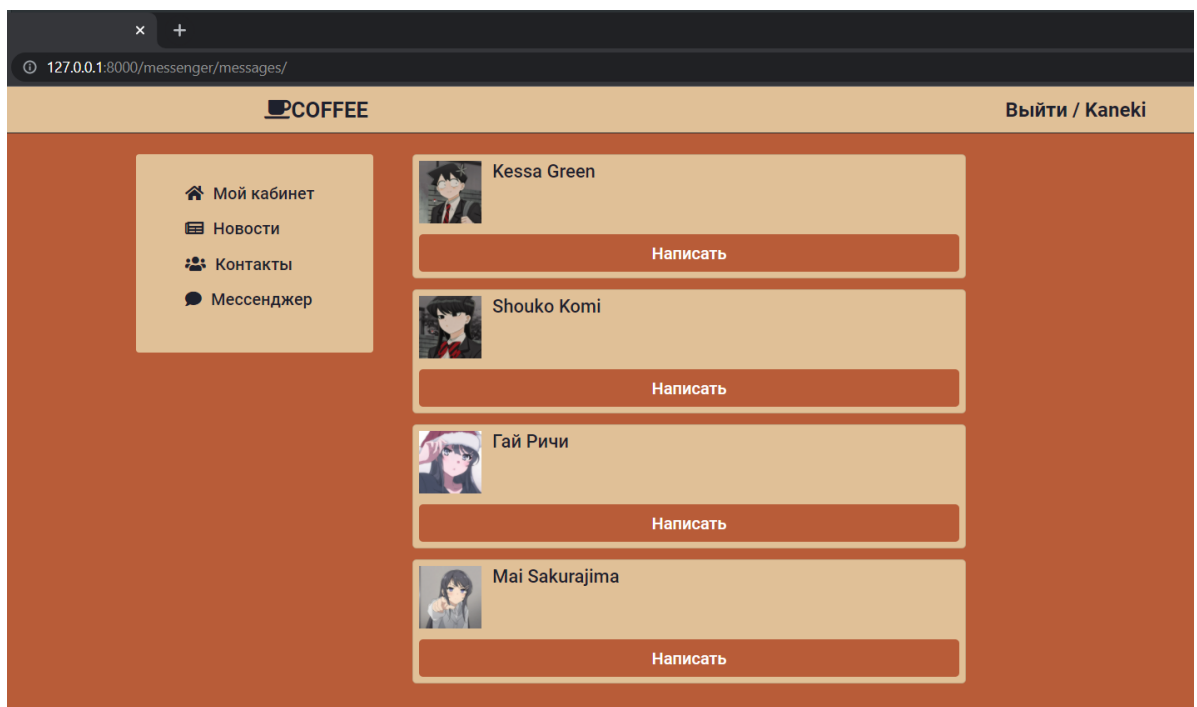


Рисунок 4.17 – Страница мессенджера веб-приложения

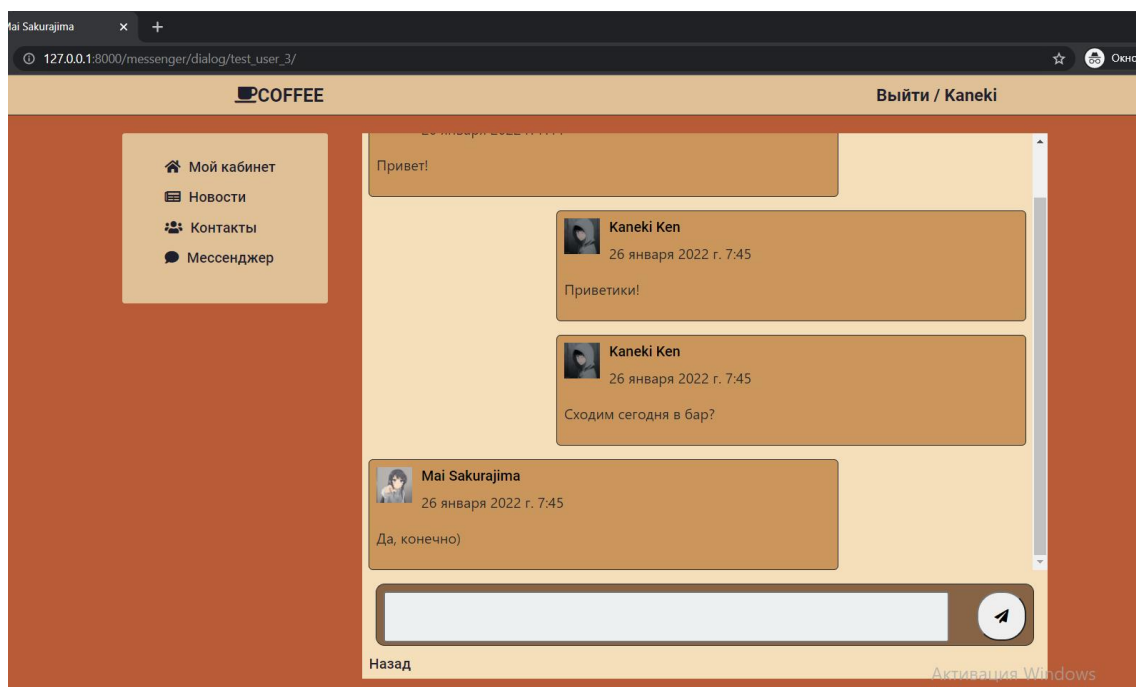


Рисунок 4.18 – Страница диалога

4.7 Тестирование

Тестирование проводилось функциональное. Проводилось тестирование системы авторизации и регистрации, реакция на попытку создания аккаунта с существующим логином, недопустимым паролем, реакция на попытку входа с неправильно введенным паролем. Были проверены основные функции веб приложения.

Сообщения в мессенджере отображаются одновременно у обоих пользователей диалога. Шифрование и расшифровка сообщения работает корректно. В базе данных сообщения хранятся в зашифрованном виде. При утечке данных из базы данных будет сохранена конфиденциальность переписки пользователей. Сообщения отображаются корректно в окне диалога. Добавленные в контакты пользователи отображаются корректно.

Не авторизованный пользователь не имеет доступа к функционалу доступному авторизованным пользователем. При попытке перехода на страницу пользователя через публикацию в новостной ленте, неавторизованного пользователя будет перебрасывать на начальную страницу с предложением войти или зарегистрироваться.

При удалении пользователя из системы администратором вместе с этим удаляются все данные пользователя, находящиеся в БД – персональные данные, все публикации и сообщения.

5 Заключение

В ходе выполнения курсовой работы были выполнены все задачи. Проект полностью готов к использованию при условии выгрузки его на хостинг сервис для использования через сеть Интернет. Был выбран стек технологий для разработки ПО, написано техническое задание, спроектировано будущее приложение, произведено веб-приложение, написана документация по реализации программного приложения в виде руководства программиста, написано руководство пользователя и администратора, изучены принципы работы клиент-серверного приложения. Были реализованы следующие функции: клиент и сервер взаимодействуют по протоколу HTTP. Сервер предоставляет возможности для регистрации и авторизации в приложении, публикации новостей, отправки сообщений между пользователями, хранением их в зашифрованном виде для сохранения конфиденциальности переписки. Приложение было функционально протестировано, с непосредственным исправлением всех найденных ошибок.

Отчет по курсовой работе составлен согласно ОС ТУСУР 01-2013.

Приложение А
(обязательное)

Руководство пользователя

При переходе на главную страницу сайта пользователя встречает окно приветствия. В шапке сайта есть три ссылки: открытие новостной ленты (логотип слева), вход и регистрация.

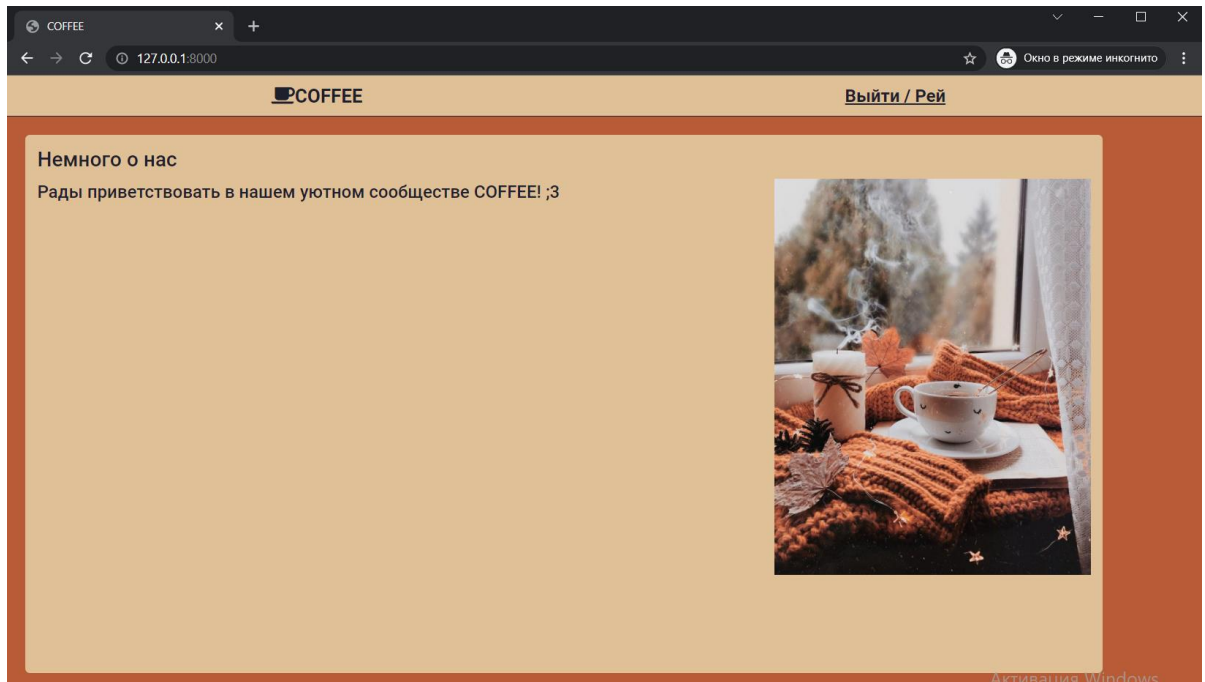


Рисунок 1 – Базовая страница сайта

Для регистрации нужно перейти по ссылке «Зарегистрироваться» для прохождения процедуры регистрации. Откроется страница регистрации с формой для заполнения регистрационных данных. Для регистрации нужно придумать уникальный логин, пароль, заполнить поля «имя», «фамилия».

The screenshot shows a web browser window with the title 'Регистрация' and the address bar displaying '127.0.0.1:8000/account/register/'. The page features a header with the 'COFFEE' logo and navigation links 'Войти / Зарегистрироваться'. The main content area has a dark orange background and a central light orange box titled 'Регистрация'. This box contains five input fields labeled 'Логин:', 'Имя:', 'Фамилия:', 'Пароль:', and 'Подтверждение пароля:', followed by a green button labeled 'Зарегистрироваться'.

Рисунок 2 – Страница регистрации

Если же пользователь уже зарегистрирован в системе, то нужно перейти по ссылке «Вход» для прохождения авторизации. Для авторизации нужно ввести свой логин и пароль.

The screenshot shows a web browser window with the title 'Вход' and the address bar displaying '127.0.0.1:8000/account/login/'. The page features a header with the 'COFFEE' logo and navigation links 'Войти / Зарегистрироваться'. The main content area has a dark orange background and a central light orange box titled 'Вход'. This box contains two input fields labeled 'Имя пользователя' and 'Пароль', followed by a green button labeled 'Войти'. Below the button is a link labeled 'Регистрация'.

Рисунок 3 – Страница входа

Системой предусмотрено использование приложения без авторизации. В таком случае будет возможен только просмотр публикации, без возможности их создания. Для этого нужно перейти по ссылке, нажав на логотип в шапке сайта.

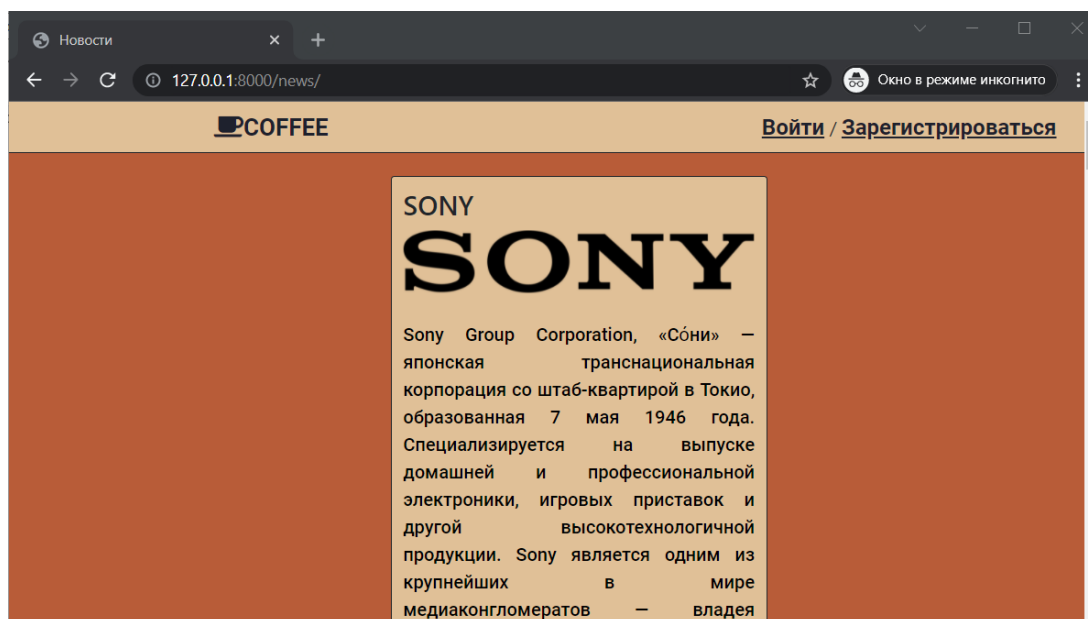


Рисунок4 – Страница «Новости» (неавторизованный пользователь)

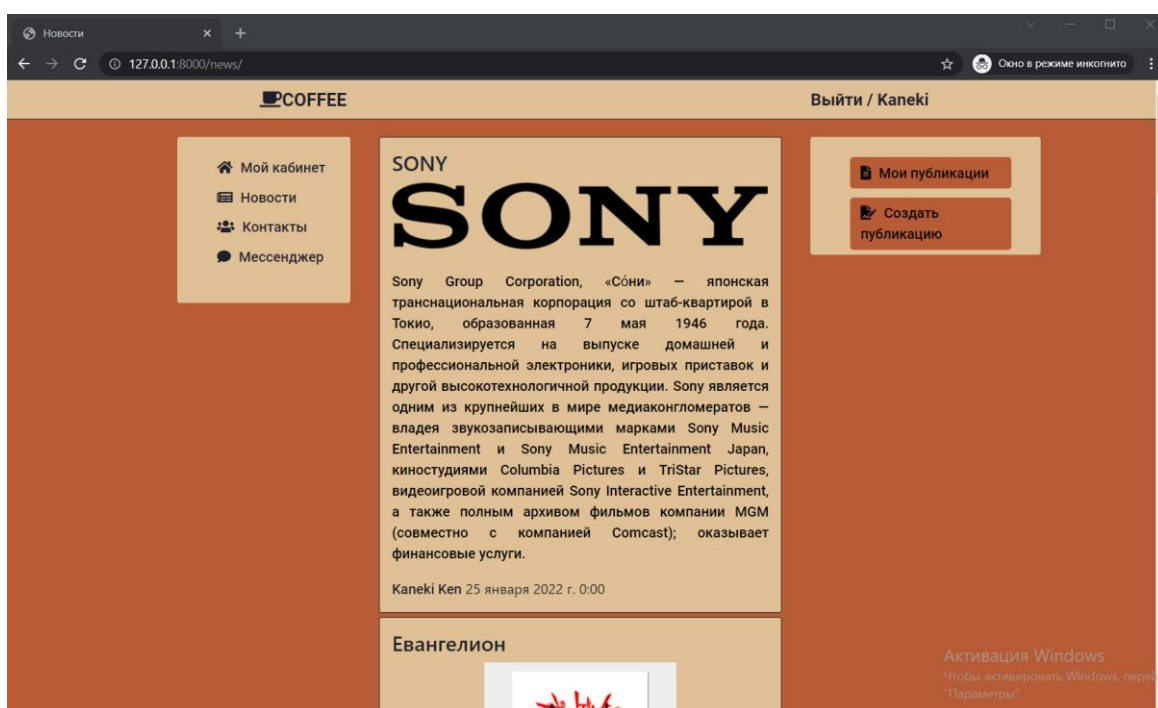


Рисунок 5 – Страница «Новости» (авторизованный пользователь)

Навигация по сайту происходит через меню в левой части. С помощью этого меню можно перейти в личный кабинет, страницу с новостной лентой, к списку контактов и мессенджеру. С помощью навигационного блока справа от новостной ленты можно создать публикацию. При нажатии на кнопку «Создать публикацию» откроется страница редактирования публикаций. Для создания публикации нужно заполнить форму, в которой указывается заголовок, текст и изображение публикации.

Рисунок 6 – Страница создания публикации

При переходе в личный кабинет будет отображена основная информация о пользователе, все новости, опубликованные данным пользователем, и кнопка редактирования профиля.

Рисунок 7 – Личный кабинет пользователя

При переходе на страницу редактирования профиля, откроется страница с формой для редактирования. Здесь можно изменить все данные о профиле пользователя. Для применения настроек нужно нажать на кнопку «Сохранить изменения».

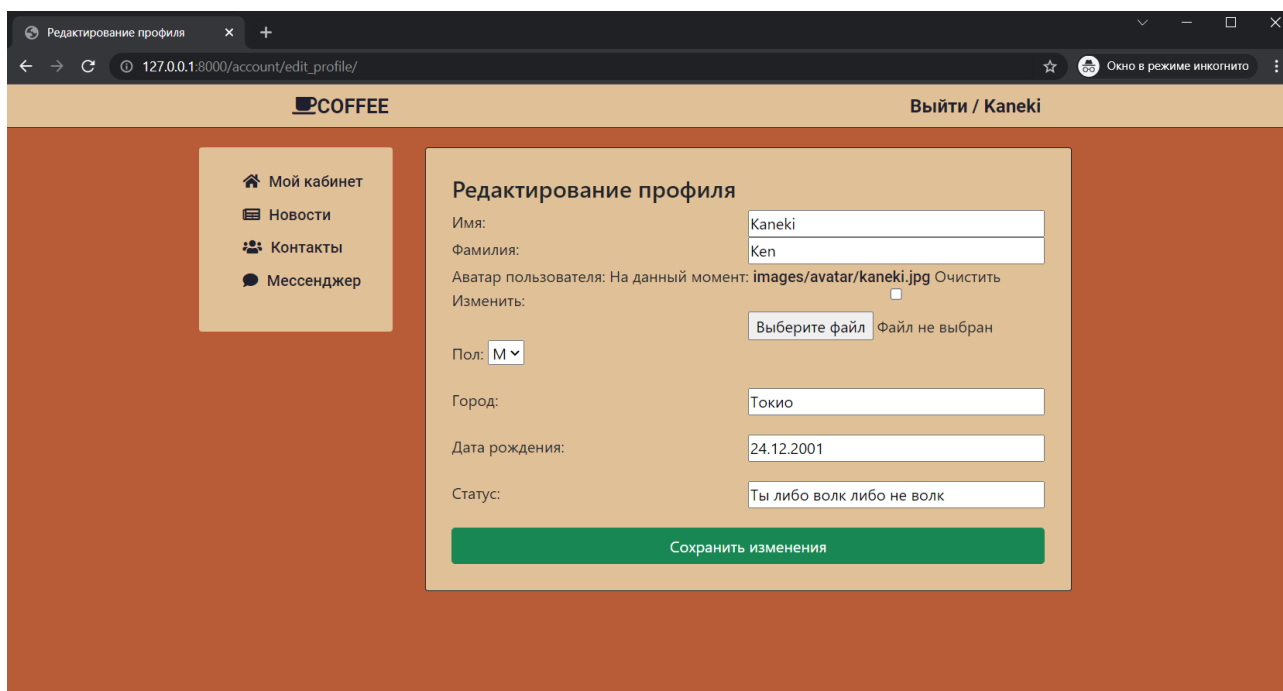


Рисунок 8 – Страница редактирования профиля

При переходе на страницу «Контакты» пользователю отображается список его контактов. На этой же странице есть кнопка поиска пользователей и поле для ввода имени для поиска.

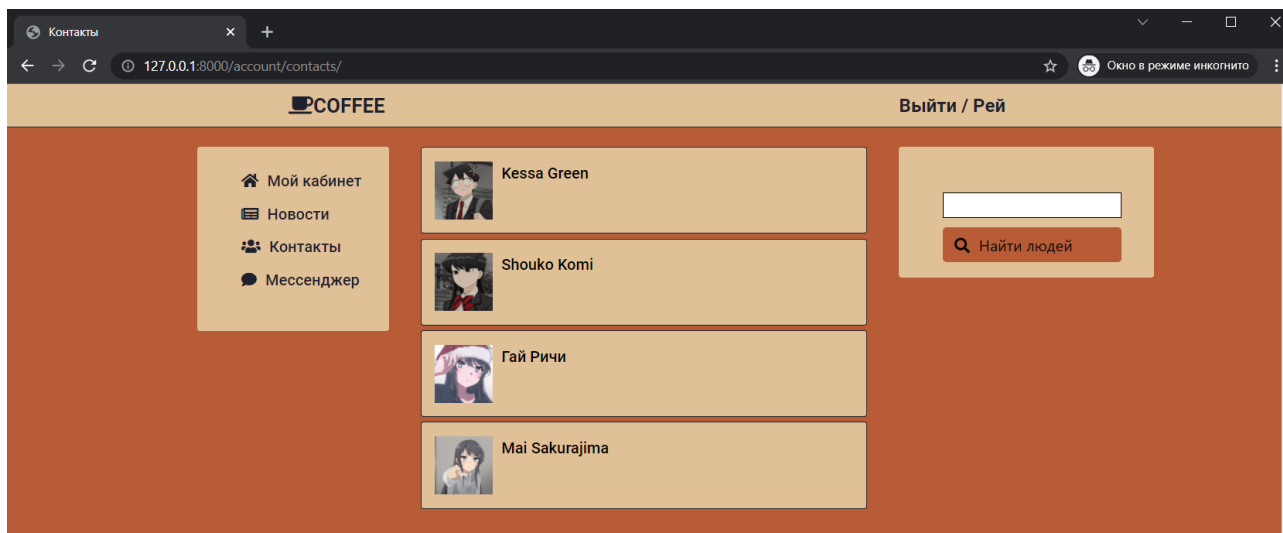


Рисунок 9 – Страница «Контакты»

При нажатии на кнопку «Найти людей» не вводя имя для поиска будет отображен список всех пользователей, зарегистрированных в системе. Если ввести в поле имя существующего пользователя, то будет выдан список пользователей с заданным именем или фамилией.

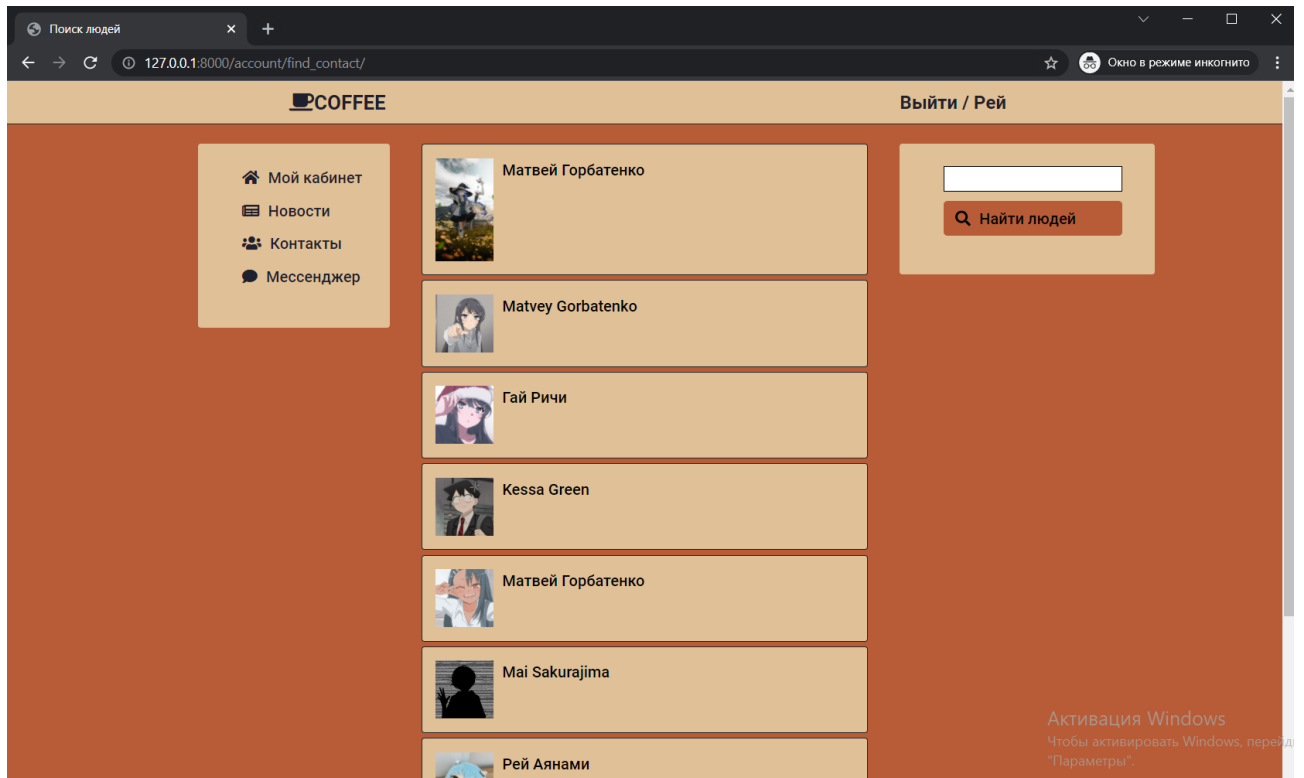


Рисунок 10 – Результат поиска (поле пустое)

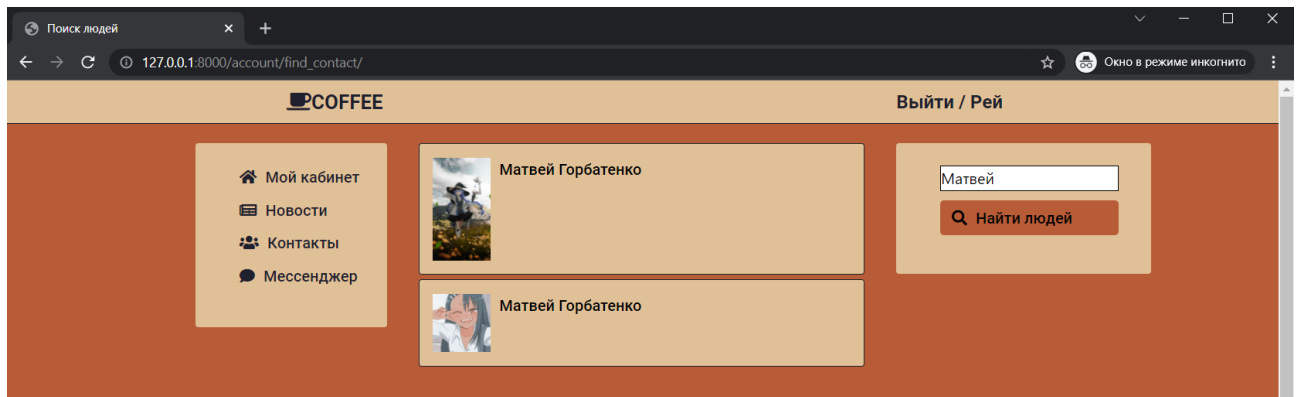


Рисунок 11 – Результат поиска (поле заполнено)

Отсюда можно перейти на страницу пользователя. На странице пользователя есть две кнопки: переход к диалогу с пользователем и добавление его в контакты. При добавлении пользователя в контакты, кнопка «Добавить в контакты» изменяется на «Удалить из контактов».

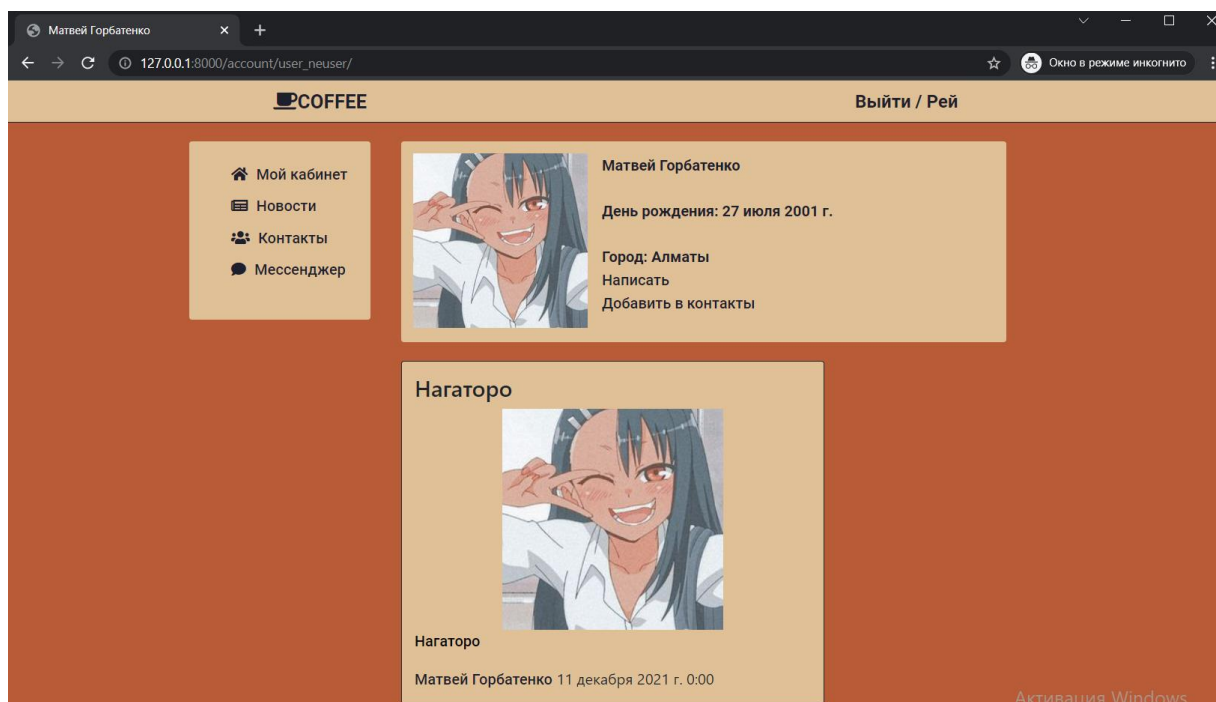


Рисунок 12 – Страница пользователя

При переходе на страницу «Мессенджер» пользователю отображается список его диалогов.

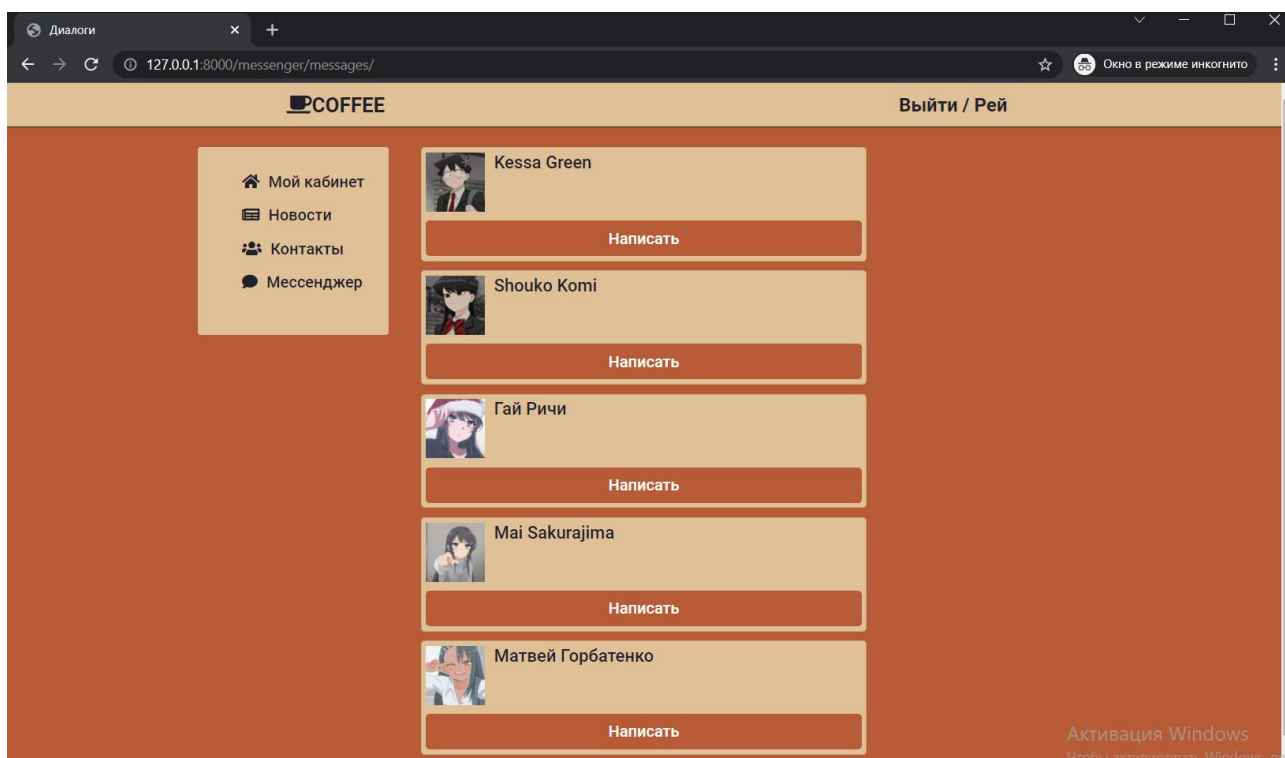


Рисунок 13 – Страница «Мессенджер»

Чтобы написать пользователю, нужно нажать на кнопку «Написать». Тогда откроется диалог с выбранным пользователем.

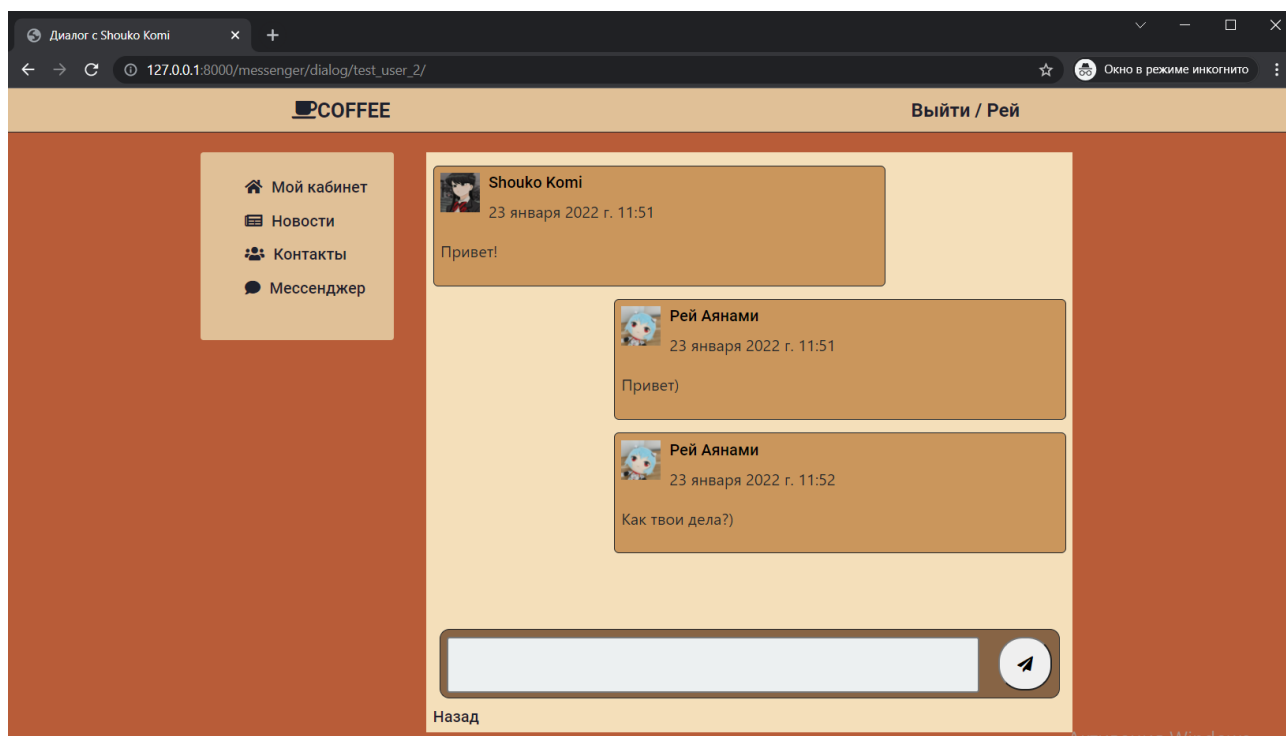


Рисунок 14 – Страница диалога с пользователем

В окне диалога отображается история переписки с данным пользователем. Сообщения собеседника прижаты к левому краю, ваши к правому. Внизу диалогового окна расположено поле ввода текста сообщения, кнопка отправки сообщения и кнопка перехода к списку диалогов.

Чтобы завершить сеанс, нужно перейти по ссылке «Выйти». После этого вы выйдете из системы и будет отображена страница с новостной лентой.

Приложение Б (обязательное)

Руководство администратора

Запуск сервера происходит при вводе в терминал специальной команды (рисунок _). Предварительно нужно перейти в каталог *socialnetwork*. Для создания пользователя-администратора нужно ввести команду *python manage.py createsuperuser*.

```
PS C:\Users\Matvey\PycharmProjects\web-app> cd socialnetwork
PS C:\Users\Matvey\PycharmProjects\web-app\socialnetwork> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
```

```
System check identified no issues (0 silenced).
January 26, 2022 - 21:13:14
Django version 3.2.9, using settings 'socialnetwork.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Рисунок 1 – Запуск сервера

Чтобы зайти в панель администратора нужно перейти по url адресу admin.

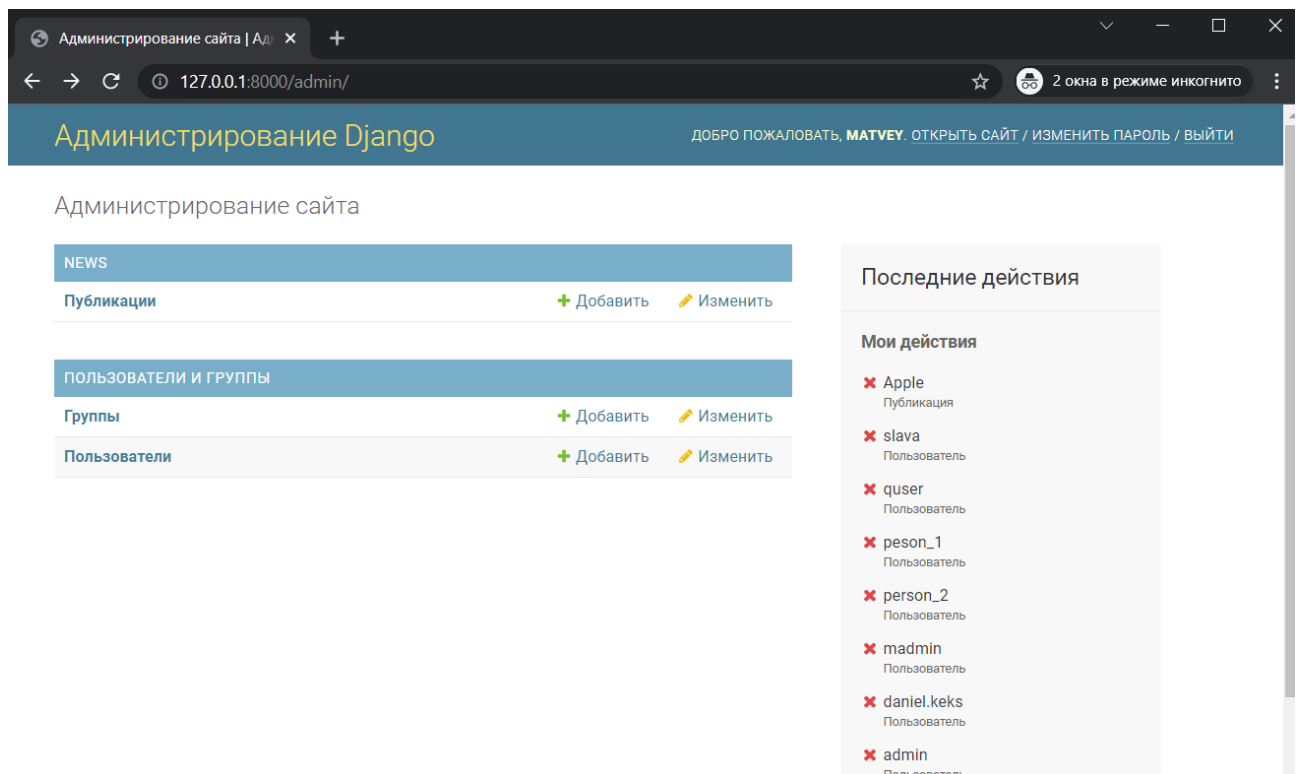


Рисунок 2 – Панель администрирования Django

В панели администратора можно просматривать данные пользователей, зарегистрированных в системе, изменять их данные и выдавать различные права. Также администраторы могут просматривать и удалять публикации пользователей, создавать свои. В панель администратора можно добавлять различные таблицы для последующей работы с ними. Для данного приложения администратору предоставлены только пользователи и их публикации. Администратор является таким же пользователем, как и все, поэтому у него тоже есть своя страница на сайте. Администратор имеет право поменять пароль пользователя, при утере пароля таковым. Для этого пользователь может обратиться к администратору.

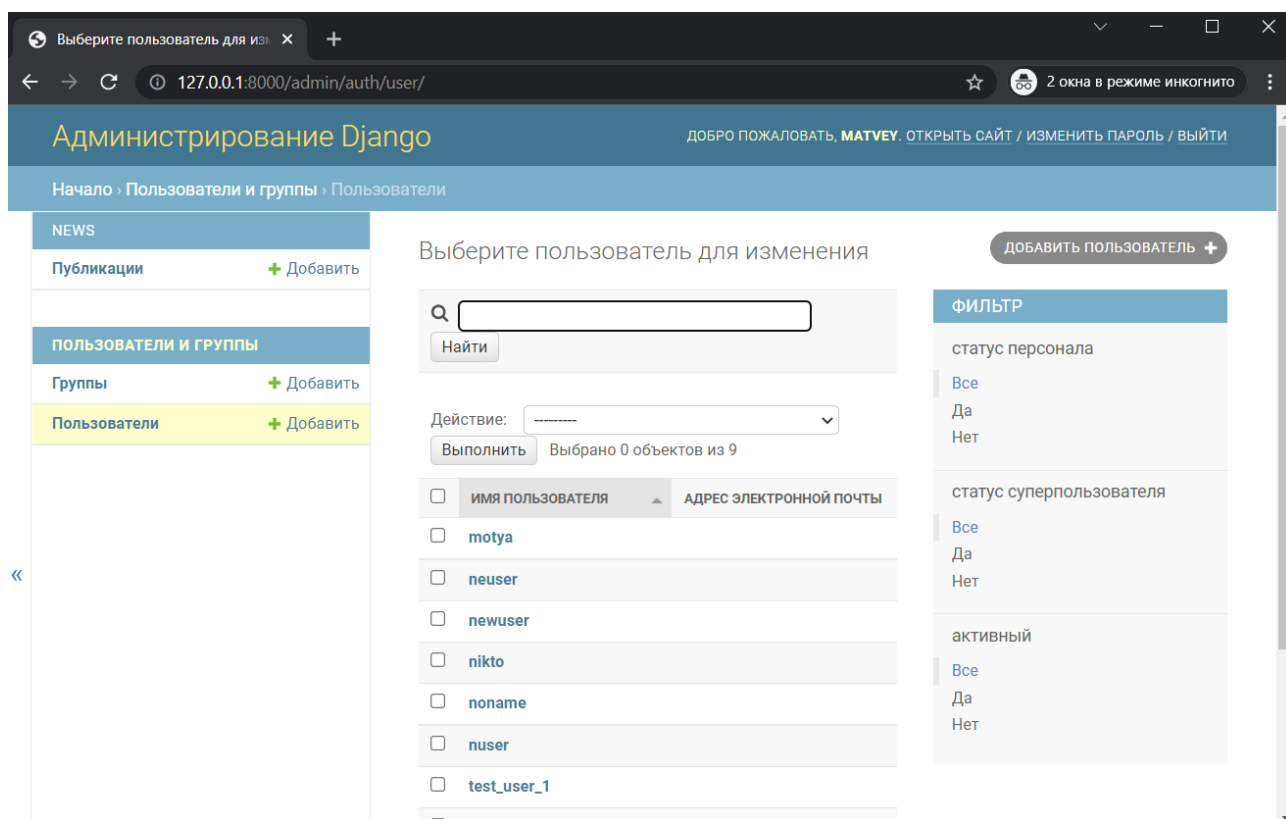


Рисунок 3 – Работа с пользователями

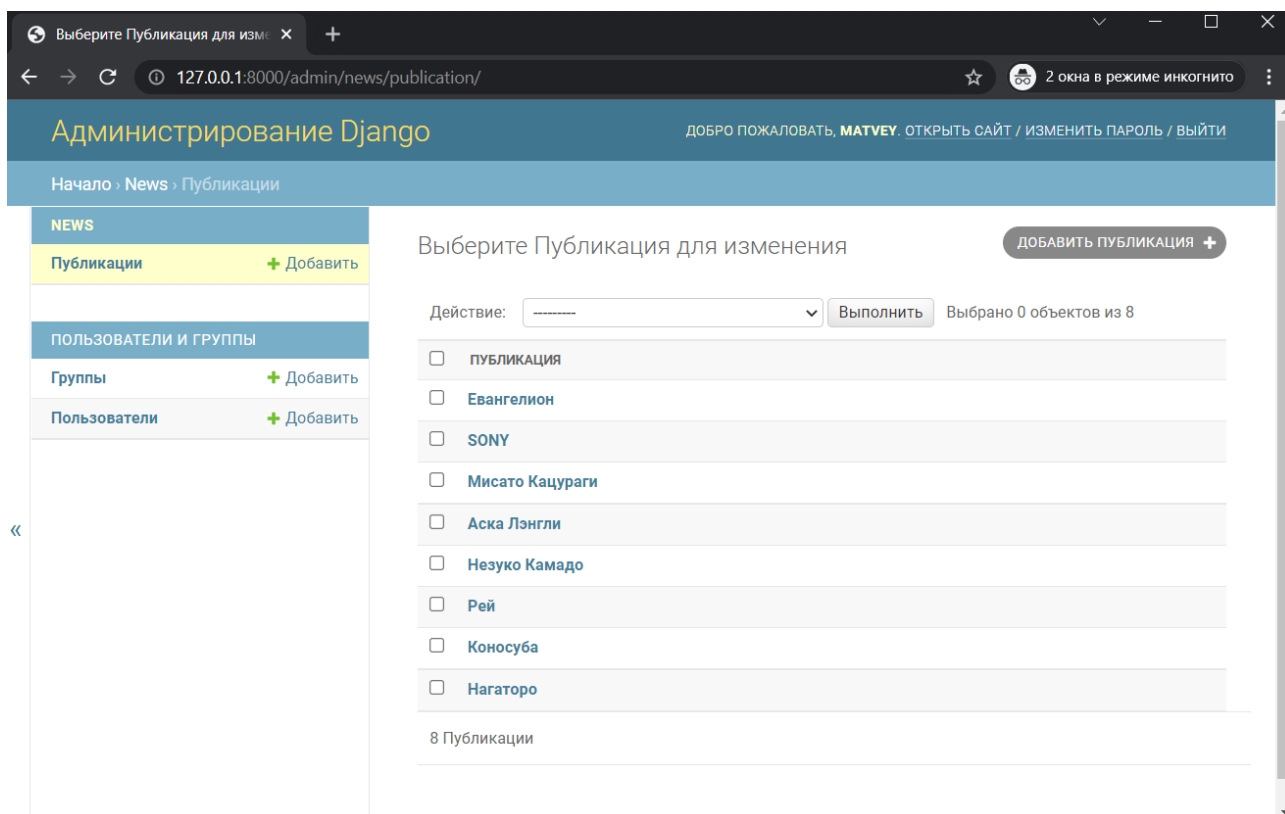


Рисунок 4 – Работа с публикациями

Приложение В

(обязательное)

Руководство программиста

Для разработки и устранения неполадок в веб-приложении необходимы знания языка программирования Python, фреймворка Django, шаблонизатора Jinja2, знания HTML, CSS, JavaScript, Ajax, ООП.

Приложение main содержит только одну страницу (базовая страница приветствия). В папке templates хранится базовый шаблон, шаблоны шапки сайта и меню. Все страницы веб-приложения наследуют данный базовый шаблон.

В приложении news содержится вся подсистема публикаций новостей. Здесь маршрутизация ограничивается страницей новостной ленты и страницей редактирования публикаций. Соответственно в файле views только две функции для открытия данных страниц. В шаблон news.html передается массив всех публикаций, отсортированных по дате создания, начиная с самой новой, и авторизованный пользователь.

```
def news_home(request):
    news = Publication.objects.order_by('-date')

    return render(request, 'news/news.html', {'news': news, 'username':
auth.get_user(request).username})
```

В функции create в шаблон передается форма редактирования публикации. Сама модель формы находится в файле forms.py.

```
def create(request):
    user_name = auth.get_user(request).username
    error = ''
    if request.method == 'POST':
        form = PublicationForm(request.POST, request.FILES)

        if form.is_valid():
            post = form.save(commit=False)
            post.author = User.objects.get(username=user_name)

            form.save()
            return redirect('news')
        else:
            error = 'Форма была неверной'
    else:
        form = PublicationForm()
        data = {
            'form': form,
            'error': error,
            'username': auth.get_user(request).username,
        }
    return render(request, 'news/create.html', data)
```

В приложении users содержится подсистема, связанная с пользователями. На страницы login.html и register.html передаются соответствующие формы для отправки клиентских данных на сервер.

На странице profile.html отображается информация о профиле авторизованного пользователя с кнопкой перехода на редактирование профиля. Вместе с шаблоном отправляется экземпляр объекта User (авторизованного пользователя) и новости этого пользователя.

```
def profile(request):
    username = auth.get_user(request).username
    user = User.objects.get(username=username)
    news = apps.get_model('news',
'Publication').objects.filter(author=user).order_by('-date')
    return render(request, 'users/profile.html', {'user': user, 'news': news})
```

Для страницы user.html данные передаются аналогичным образом, за исключением информации о том, кто владелец данной страницы и является ли владелец страницы контактом авторизованного пользователя. При переходе по ссылке add_contact срабатывает одноименная функция в файле views.py, и пользователь добавляется в контакты авторизованного пользователя.

```
def add_contact(request, account_username):
    try:
        user = User.objects.get(username=account_username)
    except:
        raise Http404("Пользователь не найден!")

    is_friend = Contact.objects.filter(user=request.user, users_friend=user)
    if not is_friend:
        add_contact = Contact(user=request.user, users_friend=user)
        add_contact.save()
    return HttpResponseRedirect(reverse('users:contacts'))
```

При перезагрузке страницы ссылка меняется на delete_contact, при переходе на которую пользователь удаляется из списка контактов.

```
def delete_contact(request, account_username):
    try:
        user = User.objects.get(username=account_username)
    except:
        raise Http404("Пользователь не найден!")

    new_friend = Contact.objects.filter(user=request.user, users_friend=user)
    new_friend.delete()
    return HttpResponseRedirect(request.META.get('HTTP_REFERER'))
```

Для просмотра списка контактов осуществляется переход по ссылке contacts. В результате вызывается функция contacts которая открывает страницу со списком контактов,

которые принадлежат данному пользователю. В шаблон отправляется контекст в виде словаря.

В словаре находится сам пользователь, список его контактов.

```
def contacts(request):
    username = auth.get_user(request).username
    user = User.objects.get(username=username)
    contacts_list = Contact.objects.filter(user=request.user)
    data = {
        'contacts': contacts_list,
        'user': user,
        'username': username,
    }
    return render(request, 'users/contacts.html', data)
```

В приложении messenger содержится подсистема обмена сообщениями. На странице messages.html отображается список диалогов. Для перехода к диалогу с пользователем нужно перейти по ссылке dialog.html. На этой странице в окне диалога отображается переписка, поле ввода текста и кнопка отправки сообщения. Отправка сообщения происходит с помощью функции leave_message:

```
def leave_message(request, reciever_name):
    reciever_user = User.objects.get(username = reciever_name)
    if request.method == 'POST':
        message_text = request.POST['message_text']
        print(message_text)

        a = Message(sender = request.user, reciever = reciever_user,
message_text = message_text, message_time = timezone.now())
        a.encrypt_message(message_text)
        a.save()
        messages = Message.objects.filter(id = a.id)
        for el in messages:
            el.decrypt_message()

        context = {'messages': messages, 'user': request.user}
        if messages:
            return HttpResponse(
                json.dumps({
                    "result": True,
                    "messages_list":
render_to_string('messenger/dialog_messages_block.html', context),
                })),
                content_type="application/json")
        else:
            return HttpResponse(
                json.dumps({
                    "result": False,
                })),
                content_type="application/json")
```

Текст сообщения отправляется с формы POST запросом и попадает на сервер, где на основе модели Message создается объект, в который записывается сообщение, предварительно зашифрованное с помощью шифра RSA. В ответ на клиент посылаются данные в виде json,

для мгновенного отображения на странице диалога отправленного сообщения. Для проверки новых сообщения реализована функция `post`, код которой представлен ниже.

```
def post(request, username):
    companion = User.objects.get(username = username)
    messages = Message.objects.filter(reciever = request.user, sender =
companion, is_readed = False)
    for message in messages:
        message.is_readed = True
        message.save()

    for message in messages:
        message.decrypt_message()
    context = {'messages': messages, 'user': request.user}
    if messages:
        return HttpResponse(
            json.dumps({
                "result": True,
                "messages_list":
render_to_string('messenger/dialog_messages_block.html', context),
            })),
            content_type="application/json"
        )
    else:
        return HttpResponse(
            json.dumps({
                "result": False,
            })),
            content_type="application/json"
        )
```

Данная функция работает на сервере. С клиент каждую секунду отправляется GET запрос на проверку новых сообщений:

```
setInterval(function () {
    $.ajax({
        url: "{% url 'dialogs:post' contact.username %}",
        type: 'GET',
        data: {'check': true},

        success: function (json) {
            if (json.result) {
                var doc = $.parseHTML(json.messages_list);
                $('#message_block').append(doc);
                $('#message_block').scrollTop(9999);
            }
        }
    });
}, 1000);
```

Если сервер находит в базе новое сообщение от собеседника, то оно появляется в диалоговом окне. Для этого используется асинхронный запрос `ajax`. Передача данных осуществляется через `json`. Поэтому сообщения при переписке отображаются без перезагрузки страницы.

Шифрование сообщение происходит на основе подготовленных заранее ключей. Так же прописан функционал для генерации новых ключей. В базе данных сообщения хранятся в

текстовом формате в виде набора чисел, количество чисел в сообщении сопоставимо количеству символов в сообщении. Символы сообщения декодируются в код ASCII. После чего все коды подвергаются математическим преобразованиям в соответствии с алгоритмом RSA.

```
def encrypt(input_text, e, n):          # шифрование сообщения
    text_nums = to_nums(input_text)
    encrypted_text_nums = [0] * len(text_nums)
    for i in range(len(encrypted_text_nums)):
        encrypted_text_nums[i] = pow(text_nums[i], to_bin(e), n)

    result = ''
    for i in range(len(encrypted_text_nums)):
        result += str(encrypted_text_nums[i])
        result += ' '
    print(result)

    return result

def decrypt(text_nums, d, n):          # расшифровка сообщения
    new_enc_txt = []
    encrypted_text = text_nums.split()
    print(encrypted_text)
    for el in range(len(encrypted_text)):
        new_enc_txt.append(int(encrypted_text[el]))

    print(new_enc_txt)

    decrypted_text_nums = [0] * len(new_enc_txt)
    for i in range(len(decrypted_text_nums)):
        decrypted_text_nums[i] = pow(new_enc_txt[i], to_bin(d), n)

    decrypted_text = to_letters(decrypted_text_nums)
    print(decrypted_text_nums)
    return decrypted_text
```

Функции шифрования вызываются с помощью методов класса Message: `encrypt_message` и `decrypt_message`.

Для отображения списка доступных диалогов используется функция `messages`. Из базы данных вытягиваются все пользователи, с которыми переписывался текущий пользователь, и посылает клиенту список этих пользователей вместе с контекстом страницы.