**Министерство науки и высшего образования Российской Федерации**

Калужский филиал федерального государственного бюджетного

образовательного учреждения высшего образования

***«Московский государственный технический университет имени Н.Э. Баумана***

***(национальный исследовательский университет)» (КФ МГТУ им. Н.Э. Баумана)***

**ФАКУЛЬТЕТ *ИУК «Информатика и управление»***

## КАФЕДРА *ИУК5 «Системы обработки информации»*

**Р А С Ч Е Т Н О - П О Я С Н И Т Е Л Ь Н А Я**

**З А П И С К А**

**к курсовой работе на тему:**

«Приложение-чат с возможностью обмена сообщениями»

по дисциплине «***Сетевые технологии в автоматизированных системах обработки информации и управления****»*

### Студент гр. ИУК5-61Б ( \_\_\_Шилкина М.С. )

(подпись) (Ф.И.О.)

### Руководитель ( Вершинин Е.В. )

(подпись) (Ф.И.О.)

### Оценка руководителя баллов

30-50 (дата)

|  |  |  |
| --- | --- | --- |
| Оценка защиты | баллов  30-50 | (дата) |
| Оценка проекта | баллов | (оценка по пятибалльной шкале) |

Комиссия: ( )

(подпись) (Ф.И.О.)

( )

(подпись) (Ф.И.О.)

( )

(подпись) (Ф.И.О.)

Калуга, 2021

Калужский филиал

федерального государственного бюджетного образовательного учреждения высшего образования

#### «Московский государственный технический университет имени Н.Э. Баумана

***(национальный исследовательский университет)» (КФ МГТУ им. Н.Э. Баумана)***

УТВЕРЖДАЮ

Заведующий кафедрой **ИУК5**

(Вершинин Е.В.)

« 08» \_\_\_\_\_\_ 2021г.

**З А Д А Н И Е**

# на выполнение курсовой работы

# по дисциплине «Сетевые технологии в автоматизированных системах обработки информации и управления»

Студент *Шилкина М.С., ИУК5-61Б*

(фамилия, инициалы, индекс группы)

Руководитель

*Вершинин Е.В.*

(фамилия, инициалы)

График выполнения работы: 25% к 4 нед., 50% к 7 нед., 75% к 10 нед., 100% к 14 нед.

#### Тема курсовой работы

1. ***Техническое задание***

**«***Приложение-чат с возможностью обмена сообщениями****»***

*Выполнить исследование и описание предметной области, разработать методы и способы решения технических задач. Разработать и реализовать архитектуру и систему передачи информации приложения.*

#### Оформление курсовой работы

* 1. Расчетно-пояснительная записка на 40 листах формата А4.
  2. Перечень графического материала КР (плакаты, схемы, чертежи и т.п.)

1. UML-диаграмма
2. Схема структурная

Дата выдачи задания « 08» февраля 2021г.

Руководитель курсовой работы / Вершинин Е.В. /

(подпись) (Ф.И.О.)

Задание получил / Шилкина М.С. / « 08» \_\_\_\_\_\_\_\_\_ 2021 г.

(подпись) (Ф.И.О.)

Примечание:

Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**СОДЕРЖАНИЕ**

[1. Техническое задание 5](#_Toc72787370)

[1. Общие сведения 5](#_Toc72787371)

[1.1 Наименование 5](#_Toc72787372)

[1.1.1 Полное наименование: 5](#_Toc72787373)

[1.2 Исполнитель. 5](#_Toc72787374)

[2. Назначение и цели создания (развития) системы 5](#_Toc72787375)

[2.2 Цель создания системы. 5](#_Toc72787376)

[3. Характеристика объектов автоматизации 5](#_Toc72787377)

[4. Требования к системе 5](#_Toc72787378)

[4.1 Требование к архитектуре системы. 6](#_Toc72787379)

[4.2 Требования к функциям, выполняемым системой 6](#_Toc72787380)

[4.3 Состав и содержание работ по созданию системы. 6](#_Toc72787381)

[4.4 Требования к надежности 6](#_Toc72787382)

[5. Состав и содержание работ по созданию системы: 7](#_Toc72787383)

[6. Порядок контроля и приемки системы 7](#_Toc72787384)

[7. Требование к составу и содержанию работ по подготовке объекта автоматизации к вводу системы действия 7](#_Toc72787385)

[8. Требования к документированию 8](#_Toc72787386)

[9. Источники разработки 8](#_Toc72787387)

[2. Научно-исследовательская часть 9](#_Toc72787388)

[2.1 Постановка задачи проектирования 9](#_Toc72787389)

[2.2 Описание предметной области 9](#_Toc72787390)

[2.3 Анализ аналогов и прототипов 11](#_Toc72787391)

[2.4. Перечень задач, подлежащих решению в процессе разработки 13](#_Toc72787392)

[2.5 Обоснование выбора инструментов и платформы для разработки 14](#_Toc72787393)

[2.5.1 Сокеты 14](#_Toc72787394)

[2.5.2 Python 16](#_Toc72787395)

[2.5.3 Среда разработки PyCharm 20](#_Toc72787396)

[2.5.4 Tinker 21](#_Toc72787397)

[3. Проектно-конструкторская часть 23](#_Toc72787398)

[3.1. Разработка структуры приложения 23](#_Toc72787399)

[3.2 Разработка архитектуры приложения 24](#_Toc72787400)

[3.3 Разработка систем передачи информации 26](#_Toc72787401)

[3.4 Разработка интерфейса взаимодействия пользователя с системой 29](#_Toc72787402)

[4. Проектно-технологическая часть 31](#_Toc72787403)

[4.1. Тестирование и отладка макета рабочей программы 31](#_Toc72787404)

[4.2. Разработка руководства пользователя и руководства программиста (администратора) 35](#_Toc72787405)

[4.2.1 Руководство администратора сервера 35](#_Toc72787406)

[4.2.2 Руководство администратора чата 37](#_Toc72787407)

[4.2.3 Руководство обычного пользователя чата 39](#_Toc72787408)

[4.3 Экспериментальные данные тестирования процедур и функциональных задач 41](#_Toc72787409)

[4.4 Разработка спецификации на оснащение подразделений, эксплуатирующих разработанное приложение, компьютерным и сетевым оборудованием 43](#_Toc72787410)

[Заключение 45](#_Toc72787411)

[Список использованных источников 46](#_Toc72787412)

## Техническое задание

## Общие сведения

* 1. Наименование**.**
     1. Полное наименование: **«**Приложение-чат с возможностью обмена сообщениями»

## Исполнитель.

Студент 3-го курса КФ МГТУ им. Баумана гр. ИУК5-61Б по направлению «Информатика и вычислительная техника» профиль «Системы обработки информации и управления» Шилкина М.С.

## Назначение и цели создания (развития) системы

* 1. **Назначение системы.**

Разрабатываемая система предназначена для обмена сообщениями между несколькими пользователями в режиме реального времени.

## Цель создания системы.

Введение в эксплуатацию данной системы позволит осуществить процесс обмена текстовой информации между несколькими пользователями. Система позволит оперативно отправить сообщения с помощью удобного и понятного интерфейса.

## Характеристика объектов автоматизации

* 1. **Краткие сведения об объекте автоматизации.**

Объектом автоматизации является процесс обмена информации между несколькими пользователями.

## Требования к системе

## Требование к архитектуре системы.

* простота обслуживания и эксплуатации (минимальные требования к квалификации и дополнительному обучению эксплуатирующего персонала);
* экономичность (экономическая эффективность в процессе функционирования);
* модифицируемость (возможность перенастройки для работы с другими технологическими процессами);
* функциональная расширяемость (возможность ввода в систему дополнительных функциональных возможностей, не предусмотренных в техническом задании);

## Требования к функциям, выполняемым системой

Со стороны обычного пользователя реализовать возможность подключения к серверу и отправки информации (сообщения) другому пользователю с помощью удобного и понятного интерфейса.

Со стороны администратора сервера реализовать возможность запуска сервера и просмотра всех подключений.

Со стороны администратора чата реализовать возможность входа в чат под никнеймом администратора, прохождения аутенфикации с помощью пароля, быть участником сетевого чата (отсылать и получать сообщения от других пользователей), исключения пользователя из сетевого чата или добавления в черный список.

## Состав и содержание работ по созданию системы.

Работы по созданию системы выполняются в четыре этапа:

25% к 4 нед., 50% к 7 нед., 75% к 10 нед., 100% к 14 нед.

## Требования к надежности

Устойчивость сервиса при возникновении отказов сети.

## Состав и содержание работ по созданию системы:

**5.1. Этапы работ**

1 этап – изучение предметной области, определение состава и структуры данных. Результат: структура системы, ТЗ, описание объекта автоматизации и целей разработки.

2 этап – разработка архитектуры и систем передачи информации, интерфейса взаимодействия пользователя с системой. Результат: реализация функционирующего приложения.

3 этап – тестирование и отладка приложения. Результат: написание руководства пользователя по взаимодействию с системой.

## Порядок контроля и приемки системы

Установить контроль и приемку результатов работ на каждой стадии создания системы в соответствии с разделом 5.

Сдача-приёмка работ производится поэтапно, в соответствии с учебным планом. Основанием для сдачи-приёмки работ служит «Отчёт о завершении работ по стадии», представляемый Исполнителем. Для сдачи-приемки представляется также документация, перечисленная в разделе 8 настоящего ТЗ.

## Требование к составу и содержанию работ по подготовке объекта автоматизации к вводу системы действия

* настройка сервера для обеспечения безопасности и корректности функционирования системы;
* развёртывание системы и установка необходимого ПО для корректной работы системы;
* создание условий функционирования объекта автоматизации, при которых гарантируется соответствие создаваемой системы требованиям, содержащимся в ТЗ;

## Требования к документированию

Разработку системы требуется сопровождать расчетно-пояснительной запиской и графической частью. Расчетно-пояснительная записка состоит из следующих частей:

* Техническое задание
* Научно-исследовательская часть.
* Проектно-конструкторская часть.
* Проектно-технологическая часть.

## Источники разработки

В настоящем документе использованы следующая литература и нормативные документы:

* ГОСТ 34.602-89 «Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы»;
* ГОСТ 34.601-90 «Информационная технология. Комплекс стандартов на автоматизированные системы. Стадии создания»;

## 2. Научно-исследовательская часть

## 2.1 Постановка задачи проектирования

Задачей проектирования данной курсовой работы является реализация сетевого чата, который позволяет обмениваться сообщениями между несколькими пользователями в режиме реального времени.

## 2.2 Описание предметной области

Предметная область курсовой работы – передача информации по сети. Сегодня миллионы компьютеров и устройств связаны в глобальную сеть интернет, либо в отдельные локальные подсети. В связи с этим возникает необходимость создания приложений, которые бы использовали все преимущества передачи данных по сети.

В данном случае нас интересует общение между несколькими пользователями в сети. При рассмотрении общения в сети традиционно выделяют обмен текстовыми сообщениями, статическими изображениями, аудио и видеоданными. В некоторых программах используется комбинированный подход.

Принято считать, что наиболее универсальный способ общения - при помощи служб мгновенного обмена сообщениями, таких, как Skype, WhatsApp, Viber. Технология обмена [мгновенными](https://en.wikipedia.org/wiki/Instant) сообщениями  - это тип[онлайн-чата,](https://en.wikipedia.org/wiki/Online_chat) который предлагает передачу[текста в реальном времени](https://en.wikipedia.org/wiki/Real-time_text) через[Интернет](https://en.wikipedia.org/wiki/Internet).

Однако, со временем стало ясно, что использовать общеизвестные Интернет-мессенджеры в корпоративных сетях просто нельзя. Тому есть несколько очень серьёзных причин:

* Из соображений конфиденциальности. Переписка сотрудников компании должна находиться внутри компании. Историю сообщений нельзя доверить сторонней фирме, как бы вы ей ни верили. История с Эдвардом Сноуденом тому ярчайшее подтверждение.
* Невозможность управления и контроля. Скайп, ICQ, Mail.ru Agent — хорошие программы. Но вы никак не сможете ими управлять. Контролировать сотрудников, администрировать программу, настраивать её — всё это невозможно.
* Обязательное наличие выхода в Интернет. Без вариантов. Нет Интернета — нет связи. Нисколько не умаляя достоинств таких программ, стоит сказать, что некоторые, такие как Skype, в силу своей схемы работы, даже если вы ничего не принимаете и не отправляете, может в произвольные моменты работы генерировать весьма приличный трафик!
* Отсутствие ориентации на корпоративные нужды. Большая часть пользователей бесплатных Интернет-мессенджеров — обычные домашние пользователи, которые приносят доход компаниям-владельцам за счёт показываемой рекламы. Скайп и ICQ — это не офисные чаты, это публичные Интернет-службы.

[LAN мессенджер](https://en.wikipedia.org/wiki/LAN_messenger) работает подобным образом, только в локальной сети. По сравнению с обычным ПО для обмена мгновенными сообщениями подобные программы обладают следующими преимуществами:

* так как ПО работает внутри локальной сети, активное Интернет-соединение не требуется;
* лишь пользователи, находящиеся в данной локальной сети, смогут получить доступ к обмену сообщениями;
* передаваемые данные не покидают локальную сеть, поэтому система гарантированно будет свободна от спама из внешней сети.

## 2.3 Анализ аналогов и прототипов

Достаточно большое количество программ для передачи сообщений по локальной сети обладает возможностями отправки личных сообщений, передачи файлов и создания чатов.

**1. CommFort**

Одной из таких программ для общения в локальной сети является программа CommFort.

Она позволяет организовать работу с большим количеством пользователей, предоставляет возможность общения посредством голоса, обмена сообщениями (индивидуально и в группах), передачу файлов любого объема целыми папками, публикацию объявления и комментарии к ним, демонстрирования рабочего стола, передачу управления компьютером, а также позволяет проводить видеоконференции.



Рисунок 1 — Приложение CommFort

1. **MyChat**

Следующим аналогом является клиент-серверный мессенджер для локальной сети MyChat. Данный чат интегрируется с Active Directory, имеет многоуровневый список контактов и доску объявлений, предоставляет возможность системы массовых оповещений и конференций, SSL шифрования трафика, передачи оффлайн сообщений и файлов. MyChat имеет гибкую систему управления правами пользователей, передачи файлов и папок.

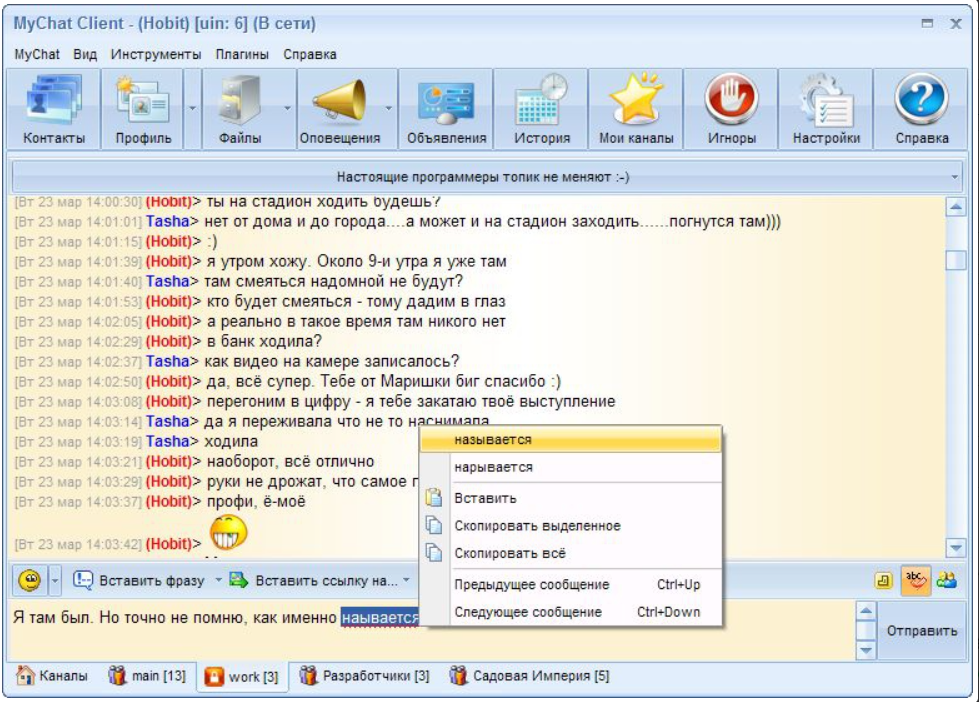


Рисунок 2 — Приложение MyChat

**3. Softros LAN Messenger**

Еще одним популярным приложением для общения в локальным сети является Softros LAN Messenger. Он является достаточно простым мессенджером и надежным для локальной сети. Не требует подключения к интернету. Кроме отправки мгновенных сообщений и передачи файлов, позволяет создавать виртуальные комнаты, делать массовые рассылки, сохранять историю сообщений, группировать контакты по отделам или должностям.

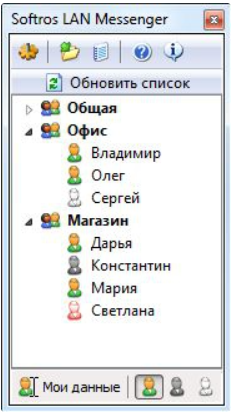


Рисунок 3 — Приложение Softros LAN Messenger

Выводы: существует огромное множество приложений для общения в локальной сети, даже есть возможность отправки видео и аудио, а также проведения видеоконференций. Поэтому тема сетевого локального чата является актуальной, и самостоятельная реализация такого чата является интересной задачей.

## 2.4. Перечень задач, подлежащих решению в процессе разработки

Для создания данной системы необходимо:

1)выбрать инструменты и платформы для разработки;

2)определить структуру приложения;

3)спроектировать пользовательский интерфейс;

4)определить архитектуру приложения;

5)разработать систему передачи данных и описать ее;

6)обнаружить и исправить возникающие ошибки.

## Обоснование выбора инструментов и платформы для разработки

## Сокеты

Сокет — это один конец двустороннего канала связи между двумя программами, работающими в сети. Соединяя вместе два сокета, можно передавать данные между разными процессами (локальными или удаленными). Реализация сокетов обеспечивает инкапсуляцию протоколов сетевого и транспортного уровней.

Сперва сокеты были разработаны в Калифорнийском университете в Беркли для UNIX. Сокеты Berkeley - это [интерфейс прикладного программирования](https://en.wikipedia.org/wiki/Application_programming_interface) (API) для [интернет-сокетов](https://en.wikipedia.org/wiki/Internet_socket) и [сокетов домена Unix](https://en.wikipedia.org/wiki/Unix_domain_socket) , используемый для [межпроцессного взаимодействия](https://en.wikipedia.org/wiki/Inter-process_communication" \o "Межпроцессного взаимодействия) (IPC). Обычно он реализуется как [библиотека](https://en.wikipedia.org/wiki/Library_(computing)) связываемых модулей.

В UNIX обеспечивающий связь метод ввода-вывода следует алгоритму open/read/write/close. Перед тем как ресурс использовать, его нужно открыть, задав соответствующие разрешения и другие параметры. После открытия ресурса, в него можно записывать или считывать из него данные. Для того чтобы пользователю подать сигнал операционной системе о завершении его работы с этим ресурсом нужно вызывать метод Close().

Когда в операционную систему UNIX были добавлены средства сетевого обмена и межпроцессного взаимодействия (Inter-Process Communication, IPC), был заимствован привычный шаблон ввода-вывода. Открытые для связи ресурсы в UNIX и Windows идентифицируются дескрипторами. Эти дескрипторы, или описатели (handles), могут указывать на файл, память или какой-либо другой канал связи, а фактически указывают на внутреннюю структуру данных, используемую операционной системой. Сокет, будучи таким же ресурсом, тоже представляется дескриптором. Следовательно, для сокетов жизнь дескриптора можно разделить на три фазы:

* открыть (создать) сокет,
* получить из сокета или отправить сокету
* и закрыть сокет.

Интерфейс IPC для взаимодействия между разными процессами построен поверх методов ввода-вывода. Они облегчают для сокетов отправку и получение данных. Каждый целевой объект задается адресом сокета, следовательно, этот адрес можно указать в клиенте, чтобы установить соединение с целью.

**Типы сокетов.** Сокеты делятся на два типа - потоковые и дейтаграммные. Для разработки программы были выбраны потоковые сокеты.

*Потоковые сокеты* (stream socket) — это сокет с установленным соединением, состоящий из потока байтов, который может быть двунаправленным, т.е. через эту конечную точку приложение может и передавать, и получать данные. Потоковый сокет обеспечивает исправление ошибок, обрабатывает доставку и сохраняет последовательность данных. Не подведет в доставке сдублированных, упорядоченных данных.

Потоковый сокет подходит для передачи больших объемов информации, поскольку накладные расходы, связанные с установлением отдельного соединения для каждого отправляемого сообщения, может оказаться неприемлемым для небольших объемов данных.

Потоковые сокеты достигают этого уровня качества за счет использования протокола Transmission Control Protocol (TCP), который свою очередь обеспечивает передачу данных на другую сторону без ошибок ив нужной последовательности.

Путь, для потокового типа сокетов формируется до начала передачи сообщений. Это гарантирует, что обе стороны, участвующие во взаимодействии, принимают и отвечают. Если приложение отправляет получателю два сообщения, то эти сообщения будут получены в той же последовательности. Однако отдельные сообщения могут дробиться на пакеты, и способа определить границы записей не существует.

Для реализации данного приложения был выбран механизм сокетов.

При использовании TCP этот протокол берет на себя разбиение передаваемых данных на пакеты соответствующего размера, отправку их в сеть и сборку их на другой стороне. Приложению известно только, что оно отправляет на уровень TCP определенное число байтов и другая сторона получает эти байты. В свою очередь TCP эффективно разбивает эти данные на пакеты подходящего размера, получает эти пакеты на другой стороне, выделяет из них данные и объединяет их вместе.

Потоки базируются на явных соединениях: сокет А запрашивает соединение с сокетом В, а сокет В либо соглашается с запросом на установление соединения, либо отвергает его.

Если данные должны гарантированно доставляться другой стороне или размер их велик, потоковые сокеты предпочтительнее дейтаграммных.

*Почему используется TCP?*

Протокол управления передачей (TCP):

Надежен: пакеты, отброшенные в сети, обнаруживаются и повторно передаются отправителем.

Имеет упорядоченную доставку данных: данные читаются вашим приложением в том порядке, в котором они были записаны отправителем.

Сокеты [протокола пользовательских дейтаграмм (UDP),](https://en.wikipedia.org/wiki/User_Datagram_Protocol) созданные с помощью socket.SOCK\_DGRAM, ненадежны, и данные, считываемые получателем, могут отличаться от порядка записи отправителя.

Наиболее распространенный тип приложений сокетов — это приложения клиент-сервер, где одна сторона действует как сервер и ожидает соединений от клиентов.

## Python

Python — это высокоуровневый язык программирования общего назначения, который используется в том числе для разработки веб-приложений и сетевого программирования. Язык ориентирован на повышение производительности разработчика и читаемости кода.

Python поддерживает несколько парадигм программирования: структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. В языке присутствет динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Программный код на Python организовывается в функции и классы, которые могут объединяться в модули, а они в свою очередь могут быть объединены в пакеты. Python обычно используется как интерпретируемый, но может быть скомпилирован в байт-код Java и в MSIL (в рамках платфоры .NET).

Область применения Python очень обширна. Его используют для создания самых различный типов задач. Это скрипты по автоматизации, системы управления жизнеобеспечением, игры, всевозможные приложения, веб-ресурсы.

Python, как и любой другой язык программирования, имеет свои **отличительные особенности**. Итак, можно выделить следующие:

* Кроссплатформенность. Python – это интерпретируемый язык, его интерпретаторы существуют для многих платформ. Поэтому с запуском его на любой ОС не должно возникнуть проблем.
* С Python доступно огромное количество сервисов, сред разработки, и фреймворков. Легко можно найти подходящий продукт для работы.
* Возможность подключить библиотеки, написанные на С. Это позволяет повысить эффективность, улучшить быстродействие.
* Наличие самых разных источников информации о Python. Не составит труда найти ответ на любой возникший вопрос, так существует много бесплатной литературы, обучающих видео-пособий, готовых исходников и шаблонов для работы в открытом доступе.

**Преимущества Python относительно других языков**

Python легко конкурирует с другими языками программирования, так как имеет множество достоинств. Во-первых, это понятный и простой язык программирования. Особенно хорошо он для новичков. Можно создать интересные приложения, и при этом не придется сидеть неделями, изучая сложный синтаксис.

Динамическая типизация – это одно из главных достоинств языка Python. Для новичков это возможность упростить написание кода и избежать множества фатальных ошибок и багов в работе. Также в Python нет операторных скобок, с расставлением которых зачастую возникают сложности.

По скорости выполнения программ, когда это касается крупных полномасштабных проектов, Python, конечно же, не лидер. Здесь минусом является и автоматическое управление памятью, и полная динамическая типизация. Python уступает значительно таким языкам как Java, С, С++, но и в то же время с легкостью дает фору JavaScript, Ruby, PHP. Подключение библиотек, написанных на С и возможность предварительной компиляции кода в байт-код – все это позволяет улучшить быстродействие.

Python предоставляет два уровня доступа к сетевым сервисам. На низком уровне вы можете получить доступ к базовой поддержке сокетов в базовой операционной системе, которая позволяет реализовывать клиенты и серверы как для протоколов, ориентированных на соединение, так и для протоколов без установления соединения. В Python также есть библиотеки, которые предоставляют высокоуровневый доступ к определенным сетевым протоколам прикладного уровня, таким как FTP, HTTP и так далее.

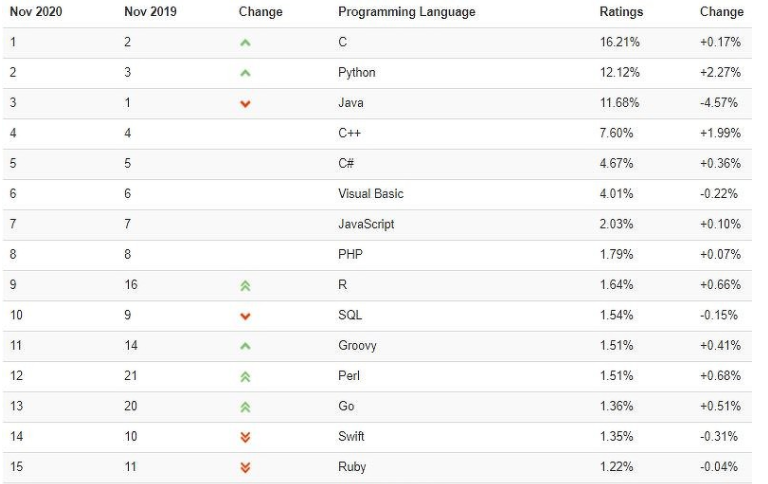
Python – это язык программирования, востребованный сегодня и с большим потенциалом в будущее. Он занимает второе место в рейтинге 2020 года по индексу TIOBE (рис. 4).

Рисунок 4 — Рейтинг языков программирования за 2020 год по индексу TIOBE

Интерфейс Python представляет собой прямую транслитерацию системного вызова Unix и интерфейса библиотеки для сокетов в объектно-ориентированный стиль Python: [socket()](https://docs.python.org/3/library/socket.html" \l "socket.socket" \o "socket.socket) функция возвращает объект сокета , методы которого реализуют различные системные вызовы сокетов.

[Модуль сокетов](https://docs.python.org/3/library/socket.html) Python предоставляет интерфейс к [API сокетов Беркли](https://en.wikipedia.org/wiki/Berkeley_sockets). Основные функции и методы API сокетов в этом модуле: socket(),

bind(), listen (), accept(), connect(), connect\_ex(), send(), recv(), close().

Python предоставляет удобный и согласованный API, который напрямую отображается на эти системные вызовы, их аналоги на C.

## Среда разработки PyCharm

**PyCharm** – это интеллектуальная Python IDE с полным набором средств для эффективной разработки на языке Python. Выпускается в двух вариантах – бесплатная версия PyCharm Community Edition и поддерживающая больший набор возможностей PyCharm Professional Edition. PyCharm выполняет инспекцию кода на лету, автодополнение, в том числе основываясь на информации, полученной во время исполнения кода, навигацию по коду, обеспечивает множество рефакторингов.

PyCharm делает разработку максимально продуктивной благодаря функциям автодополнения и анализа кода, мгновенной подсветке ошибок и быстрым исправлениям (рис.6). Автоматические рефакторинги помогают эффективно редактировать код, а удобная навигация позволяет мгновенно перемещаться по проекту.

Инструменты и функции, предоставляемые этой интегрированной средой, помогают Python-программистам быстро и эффективно писать код, синхронизироваться с системой контроля версий, использовать фреймворки и плагины, позволяя при этом настроить интерфейс так, как удобно самому пользователю и даже дают возможность дополнительного расширения IDE.

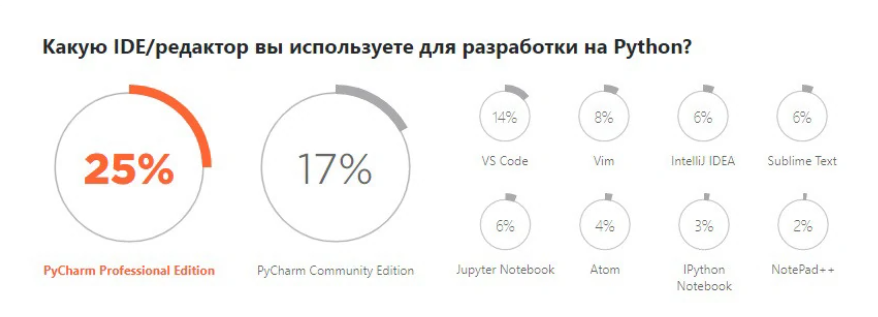
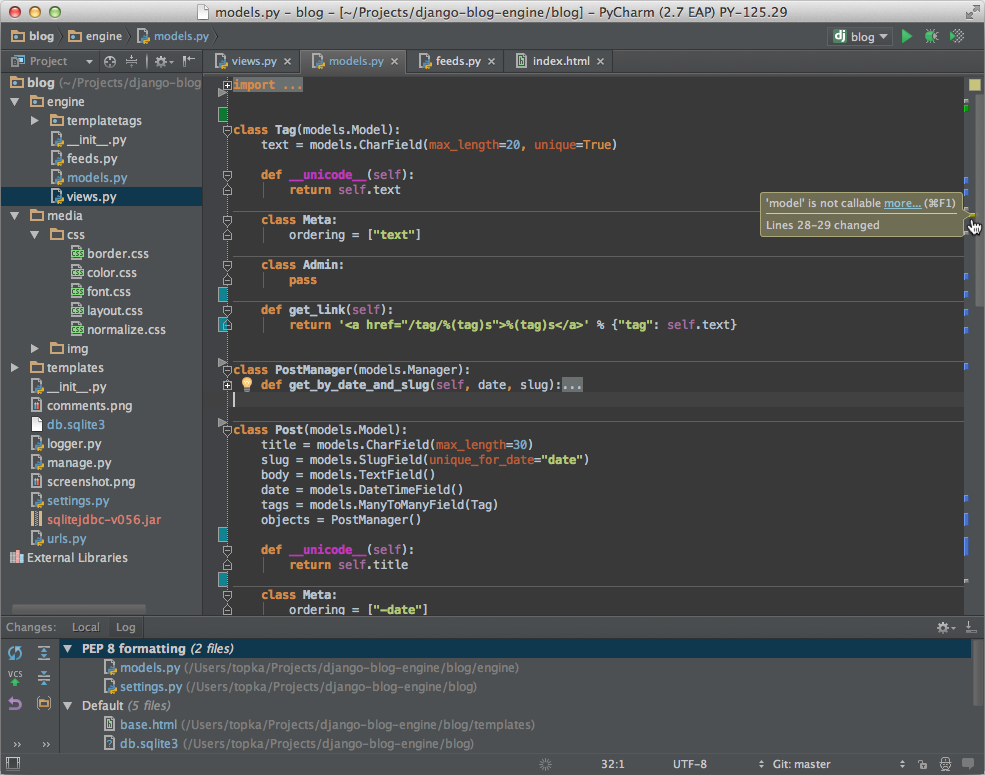
По результатам опроса JetBrains в 2020 году PyСharm заняла первое место (рис. 5).

Рисунок 5 — Опрос среди python-разработчиков в 2020 году

Рисунок 6 — PyCharm

## Tinker

В Python есть довольно много [GUI фреймворков](https://python-scripts.com/category/gui) (graphical user interface), однако только Tkinter встроен в стандартную библиотеку языка. У Tkinter есть несколько преимуществ. Он кроссплатформенный, поэтому один и тот же код можно использовать на Windows, macOS и Linux.

Визуальные элементы отображаются через собственные элементы текущей операционной системы, поэтому приложения, созданные с помощью Tkinter, выглядят так, как будто они принадлежат той платформе, на которой они работают.

Tkinter – это пакет для Python, предназначенный для работы с библиотекой Tk. Библиотека Tk содержит компоненты графического интерфейса пользователя (graphical user interface – GUI), написанные на языке программирования Tcl. Tkinter можно охарактеризовать как переводчик с языка Python на язык Tcl. Вы пишете программу на Python, а код модуля tkinter переводит ваши инструкции на язык Tcl, который понимает библиотека Tk.

Приложения с графическим интерфейсом пользователя событийно-ориентированные.

Tkinter является относительно легким по сравнению с другими библиотеками. Это отличный выбор для создания GUI приложений в Python, особенно если современный облик не в приоритете для программы, а большую роль играет функциональность и кроссплатформенная скорость.

## Проектно-конструкторская часть

## 3.1. Разработка структуры приложения

После исследования предметной области было принято решение разработать программу «Chat», с помощью которой пользователям было бы удобно и оперативно обмениваться сообщениями в сетевом чате.

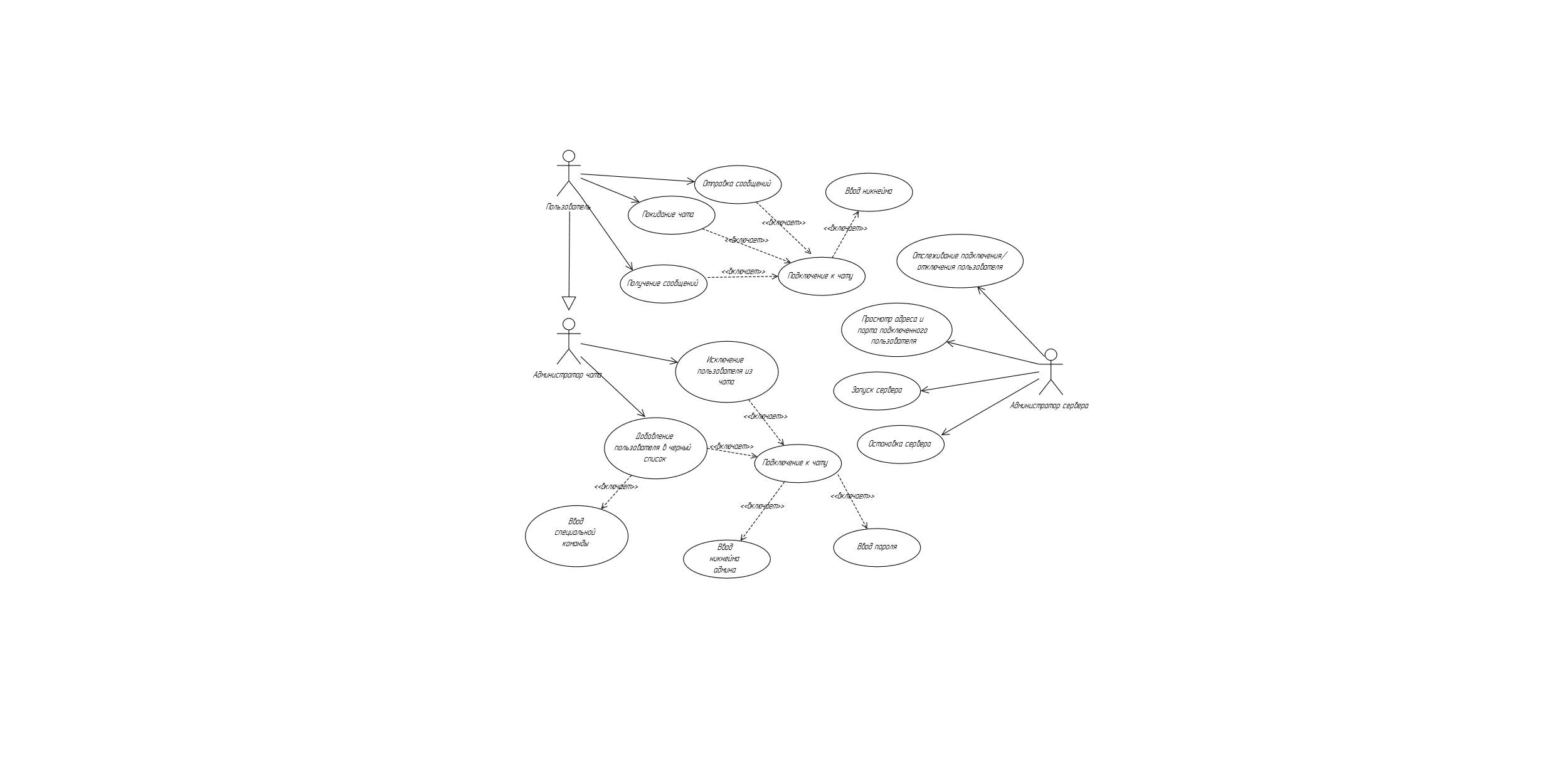
****Далее после определения действующих лиц (акторов), их взаимодействия с системой, ожидаемой функциональности системы и понятий предметной области, относящихся к детальному описанию функциональности системы (прецедентов), была составлена UML диаграмма вариантов использования (use case diagram). Данная диаграмма отражает функциональность и поведение системы и является составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Рисунок 7 — UML диаграмма вариантов использования

В приложении будет три типа действующих лиц (акторов): пользователь, администратор чата и администратор сервера.

Со стороны обычного пользователя будет реализована возможность зайти в чат, ввести свой никнейм, отсылать сообщения и получать сообщения в сетевом чате.

Со стороны администратора чата будет реализована возможность зайти в чат под никнеймом администратора, пройти аутенфикацию с помощью пароля, быть участником сетевого чата (отсылать и получать сообщения от других пользователей), исключать пользователя из сетевого чата или добавлять в черный список. После исключения из чата администратором пользователь сможет зайти в сетевой чат в следующий раз, но после добавления пользователя в черный список, он не сможет зайти в сетевой чат позднее.

Со стороны администратора сервера будет реализована возможность отслеживания подключения/отключения пользователей, просмотр информации об адресе и порте подключения, запуск/остановка самого сервера.

## 3.2 Разработка архитектуры приложения

Для нашего приложения мы будем использовать архитектуру клиент-сервер. Это означает, что у нас будет несколько клиентов (пользователей) и один центральный сервер, на котором размещено все и будет предоставляться данные для этих клиентов.

Характеристика клиент-сервер описывает отношения взаимодействующих программ в приложении. Серверный компонент предоставляет функцию или услугу одному или нескольким клиентам, которые инициируют запросы на такие услуги.

Вообще говоря, служба — это абстракция компьютерных ресурсов, и клиенту не нужно беспокоиться о том, как сервер работает при выполнении запроса и доставке ответа. Клиенту нужно только понять ответ, основанный на известном протоколе приложения, то есть содержание и форматирование данных для запрашиваемой услуги.

Клиенты и серверы обмениваются сообщениями в шаблоне запрос-ответ. Клиент отправляет запрос, а сервер возвращает ответ. Этот обмен сообщениями является примером межпроцессного взаимодействия. Для взаимодействия компьютеры должны иметь общий язык, и они должны следовать правилам, чтобы и клиент, и сервер знали, чего ожидать. Язык и правила общения определены в протоколе связи. Все протоколы клиент-серверной модели работают на уровне приложений. Протокол прикладного уровня определяет основные шаблоны диалога.

Система разбивается на две части, которые могут выполняться в разных узлах сети, - клиентскую и серверную части. Прикладная программа или конечный пользователь взаимодействуют с клиентской частью системы, которая обеспечивает надсетевой интерфейс. Клиентская часть системы при потребности обращается по сети к серверной части. Интерфейс серверной части определен и фиксирован. Поэтому возможно создание новых клиентских частей существующей системы.

Разделен код программы клиентского и серверного приложения. Это главное преимущество архитектуры. Выбрана локальная сеть. Поэтому плюсы следующие:

* к клиентским рабочим станциям выдвигают низкие запросы;
* преимущественно все вычислительные операции выполняются на серверах;
* гибкая система;
* реально повысить защиту локальной сети.

Наше приложение будет состоять из двух частей:

1. Программа Server. Сервером предоставляется IP-адрес для подключения программы клиент. Администратор сервера следит за подключением/отключением пользователей, может просматривать информацию о подключенном пользователе (никнейм, адрес и т.д. )

2. Программа Client. Во втором случае пользователь может выбрать себе имя для входа в систему, отправлять сообщения и всем пользователям, читать сообщения, адресованные всем. Также ели пользователем является администратор чата, он может исключать пользователя из сетевого чата или добавлять в черный список.

В нашей системе будет один сервер и множество клиентов (от 1). В проекте нашего приложения программный код также будет разделен на две части: server.py и client.py. Интерфейс взаимодействия с пользователем будет иметь программа-клиент.

## Разработка систем передачи информации

Для взаимодействия между сервером и клиентами будет использовать потоковые сокеты (на основе TCP, в коде обозначаются SOCK\_STREAM) - сокеты с установленным соединением на основе протокола TCP, передают поток байтов, который может быть двунаправленным - т. е. приложение может и получать, и отправлять данные.

Сокет состоит из IP-адреса и порта. В python для работы с сокетами будем использвать встроенную библиотеку socket. Одной из основных функций модуля является функция socket(), которая возвращает объект типа сокет, обладающий соответствующими функциями для работы с соединением.

Функции, которые мы будем использовать в нашем приложении для работы с сокетами.

Таблица 1 — Используемые функции при работе с сокетами

|  |  |
| --- | --- |
| Используемая функция | Назначение функции |
| socket.bind(address) | Привязывает сокет к адресу address (инициализирует IP-адрес и порт). Сокет не должен быть привязан до этого. |
| socket.listen([backlog]) | Переводит сервер в режим приема соединений. Параметр «backlog (int)» – количество соединений, которые будет принимать сервер. |
| socket.accept() | Принимает соединение и блокирует приложение в ожидании сообщения от клиента. В результате возвращает кортеж:   * conn: объект соединения (сокет), который можно использовать для отправки/получения данных; * address: адрес клиента. |
| socket.recv(bufsize[, flags]) | Читает и возвращает данные в двоичном формате (набор байтов) из сокета. Параметр bufsize (int) – максимальное количество байтов в одном сообщении. |
| socket.send(bytes[, flags]) | Отправляет данные клиенту и возвращает количество отправленных байт. Параметр bytes (bytes) – двоичные данные. |
| socket.close() | Закрывает сокет. |

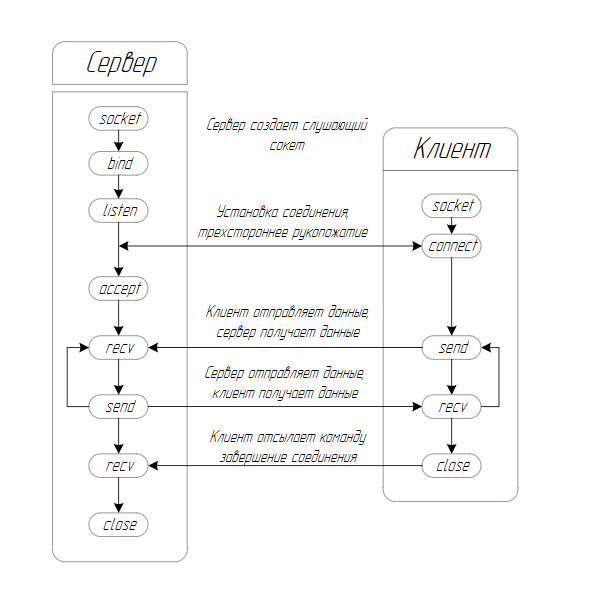
Работа с сокетом во многом схожа с работой с файловым объектом. Принцип - открыли соединение - считали данные - закрыли соединение. На рисунке 8 представлена схема работы сокетов в приложении.

Рисунок 8 — Схема работы сокетов в приложении

Начиная с верхнего левого столбца, обратите внимание на вызовы API, которые сервер делает для настройки «прослушивающего» сокета:

* socket()
* bind()
* listen()
* accept()

Слушающий сокет прослушивает соединения от клиентов. Когда клиент подключается, сервер вызывает accept(), чтобы принять или завершить подключение.

Клиент вызывает connect(), чтобы установить соединение с сервером и инициировать трехстороннее рукопожатие. Шаг подтверждения важен, поскольку он обеспечивает доступность каждой стороны соединения в сети, другими словами, что клиент может связаться с сервером и наоборот. Может случиться так, что только один хост, клиент или сервер может связаться с другим.

В середине находится секция двустороннего обмена, где данные обмениваются между клиентом и сервером с помощью вызовов send()и recv().

Внизу - close()— закрывает соответствующие сокеты клиента и сервера.

## 3.4 Разработка интерфейса взаимодействия пользователя с системой

Разработка графического интерфейса является необходимой задачей для того, чтобы пользователям было удобно взаимодействовать с системой.

Программа-клиент будет представлять из себя GUI-программу, которую мы реализуем с помощью [GUI-фреймворка](https://python-scripts.com/category/gui) Tkinter.

Программа-сервер будет представлять из себя консольную программу, наподобие терминала, в котором будет последовательно «печататься» информация.

Для правильной разработки прототипа интерфейса и последующей реализации распишем User Story.

Общая US: Я, как пользователь, хочу иметь понятный и удобный интерфейс приложения, чтобы оперативно обмениваться сообщениями с пользователями чата.

Далее разделим ее на более детализированные US.

1.Я, как пользователь, хочу иметь достаточное поле для ввода сообщения для того, чтобы написать полностью свое сообщение.

2. Я, как пользователь, хочу иметь кнопку возле поля для ввода сообщения для того, чтобы быстро отправить сообщение.

3. Я, как пользователь, хочу иметь поле со всеми сообщениями от пользователей, чтобы видеть кто и какое сообщение прислал в чате.

4. Я, как пользователь, хочу иметь кнопку для выхода, чтобы правильно и оперативно закрыть чат.

На основе US был разработан следующий прототип, представленный на рисунке 9.

Рисунок 9 — Прототип разрабатываемого приложения

## 4. Проектно-технологическая часть

## 4.1. Тестирование и отладка макета рабочей программы

Для успешной эксплуатации разработанной системы и завершения проектирования приложения, необходимо оформить требования к обработке ошибок и к обработке некорректных данных данной системы. Эти требования оформим в виде тест-кейсов для визуального и наглядного понимания.

Тест-кейс №1. Проверка авторизации администратора.

Шаги:

* 1. В модальном окне «Никнейм» заполнить поле значением «admin».
  2. Нажать кнопку «ОК».
  3. В новом модальном окне «Пароль» заполнить поле значением правильного пароля, в данном случае оно установлено, как «admin».
  4. Нажать кнопку «ОК».

Таблица 2 — Ожидаемый результат для тест-кейса №1

|  |  |
| --- | --- |
| Вводимое значение | Ожидаемый результат |
| Ввод отличного никнейма от «admin» в модальном окне «Никнейм» | Авторизация в чате, как обычный пользователь |
| Оставить поле для никнейма пустым | Модальное окно «Пароль» не откроется |
| Оставить поле для пароля пустым | Модальное окно «Чат» не откроется, выведется сообщение «Введите правильный пароль» |
| Ввод неправильного пароля | Модальное окно «Чат» не откроется, выведется сообщение «Введен неправильный пароль» и окно закроется |
| Ввод правильного пароля | Авторизация успешна, откроется модальное окно «Чат» |
| Ввод правильного никнейма | Откроется модальное окно «Пароль» |

Тест-кейс №2. Проверка авторизации обычного пользователя чата.

Шаги:

1. В модальном окне «Никнейм» заполнить поле значением никнейма.
2. Нажать кнопку «ОК».

Таблица 3 — Ожидаемый результат для тест-кейса №2

|  |  |
| --- | --- |
| Вводимое значение | Ожидаемый результат |
| Ввод никнейма, не состоящего в списке «BANS» | Успешная авторизация в чате, откроется модальное окно «Чат» |
| Оставить поле для никнейма пустым | Модальное окно «Чат» не откроется |
| Ввод никнейма, состоящего в списке «BANS» | Откроется модальное окно «Чат», но пользователь не сможет отправлять и видеть сообщения в чате |

Тест-кейс №3. Проверка функции администратора по исключению пользователей из чата.

Шаги:

1. В модальном окне «Чат» администратор вводит команду «/kick nickname».
2. Нажимает кнопку «Отправить».

Таблица 4 — Ожидаемый результат для тест-кейса №3

|  |  |
| --- | --- |
| Вводимое значение | Ожидаемый результат |
| Администратор ввел команду «/kick nickname» | 1) В поле чата у всех, кроме исключенного пользователя выводится сообщение «Nickname исключен из чата админом!».  2) У исключенного пользователя выводится сообщение «Вы были исключены из чаты админом!».  3) Исключенный пользователь не может отправлять сообщения и видеть все сообщения чата. |
| Администратор ввел неправильную команду, начинающуюся с «/», но с ошибкой в тексте команды или только «/», например, «/kikcc nickname» | В чате только администратору выведется сообщение «Неправильная команда», но администратор остается пользователем чата |

Тест-кейс №4. Проверка функции администратора по удалении пользователя из чата навсегда.

Шаги:

1. В модальном окне «Чат» администратор вводит команду «/ban nickname».
2. Нажимает кнопку «Отправить».

Таблица 5 — Ожидаемый результат для тест-кейса №4

|  |  |
| --- | --- |
| Вводимое значение | Ожидаемый результат |
| Администратор ввел команду «/ban nickname» | 1) В поле чата у всех, кроме удаленного пользователя выводится сообщение «Nickname исключен из чата админом!».  2) У удаленного пользователя выводится сообщение «Вы были исключены из чаты админом!».  3) Удаленный пользователь не может отправлять сообщения и видеть все сообщения чата. |
| Администратор ввел неправильную команду, начинающуюся с «/», но с ошибкой в тексте команды или только «/», например, «/bann nickname». | В чате только администратору выведется сообщение «Неправильная команда», но администратор остается пользователем чата |
| Удаленный пользователь заново заходит в чат и вводит тот же свой nickname | Открывается модальное окно «Чат», но пользователь не может отправлять сообщения и смотреть сообщения чата |

Тест-кейс №5. Проверка отображения сообщений у администратора сервера и в общем чате в зависимости от действий пользователя.

Таблица 5 — Ожидаемый результат для тест-кейса №4

|  |  |
| --- | --- |
| Действие пользователя | Ожидаемый результат |
| Пользователь ввел никнейм в модальном окне «Никнейм» и подключился к чату | 1) У администратора сервера выводится сообщение: «Пользователь подключен к ip-address, port»  Далее выводится следующее сообщение: «Никнейм: nickname» |
| Пользователь отправил сообщение в чат | 2) У администратора сервера выводится сообщение: «В чате: message» |
| Пользователь покинул чат | У администратора сервера выводится сообщение: «Пользователь nickname покинул чат» |
| Пользователь был удален администратором | У администратора сервера выводится сообщение: «Пользователь nickname был исключен админом» |
| Пользователя с таким никнеймом был исключен из чата администратором навсегда | У администратора сервера выводится сообщение: «Пользователь nickname был исключен админом» |

После написания тест-кейсов была отлажена и протестирована работа программы при возникновении ошибок.

## 4.2. Разработка руководства пользователя и руководства программиста (администратора)

После окончания проектирования и тестирования приложения были разработаны руководства обычного пользователя, администратора сервера и администратора чата.

## 4.2.1 Руководство администратора сервера

Для начала работы с программой и запуска сервера, администратору сервера необходимо запустить файл server.py, с помощью двойного клика по файлу или с помощью консоли (перейти по корректному пути нахождения файла server.py, ввести команду python server.py (подразумевается наличие на компьютере [интерпретатор Python](https://pythonru.com/baza-znanij/top-5-onlajn-kompiljatora-python)).

Далее в окне консоли выведется сообщение о запуске и начале работе сервера (рис.10).

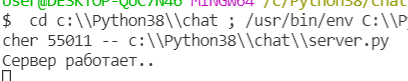


Рисунок 10 — Запуск сервера

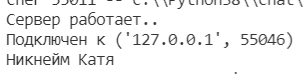
После того как обычный пользователь подключаться к чату у администратора выведутся сообщения, показанные на рис. 11.

Рисунок 11 — Сообщение о подключении пользователя

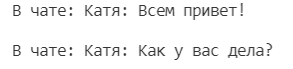
Если обычный пользователь напишет сообщение в чат, у администратора выведется сообщение, показанное на рис. 12.

Рисунок 12 — Сообщение об активности пользователя

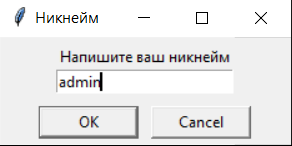
Если обычный пользователь покинет чат, у администратора выведется сообщение, показанное на рис. 13.



Рисунок 13 — Сообщение об уходе пользователя из чата

## 4.2.2 Руководство администратора чата

Для того, чтобы администратор начал пользоваться чатом, ему необходимо запустить файл client.py, с помощью двойного клика по файлу или с помощью консоли (перейти по корректному пути нахождения файла client.py, ввести команду python client.py (подразумевается наличие на компьютере [интерпретатор Python](https://pythonru.com/baza-znanij/top-5-onlajn-kompiljatora-python)).

После запуска администратору необходимо авторизоваться: 1) ввести никнейм admin в открывшемся диалоговом окне никнейм (рис.14); 2) ввести пароль от учетной записи администратора, в данном случае adminpass (рис. 15).

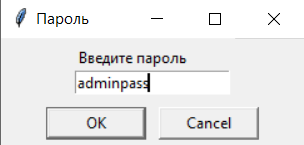
Рисунок 14 — Диалоговое окно «Никнейм»

Рисунок 15 — Диалоговое окно «Пароль»

После успешной авторизации администратора чата ему открывается главное модальное окно чата (Рис. 16). Администратор чата может отправлять сообщения в чат, как обычный пользователь.

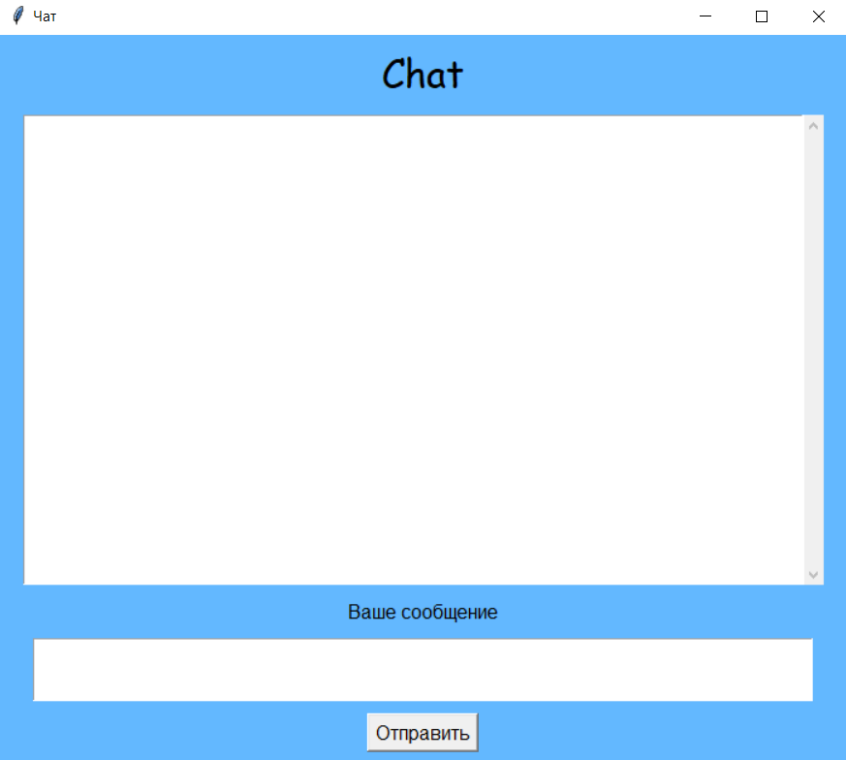


Рисунок 16 — Модальное окно «Чат»

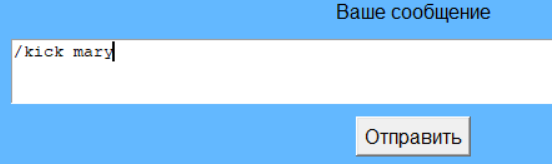
Для того, чтобы исключить пользователя из чата, администратору необходимо ввести в поле ввода команду /kick nickname(никнейм пользователя) (Рис.17). И пользователь с таким никнеймом будет исключен из чата, то есть он не сможет отправлять сообщения и видеть сообщения в чате, ему придет соответствующее сообщение (Рис. 18). Всем пользователям чата придет сообщение о том, что этот пользователь был исключен из чата (Рис.19).

Рисунок 17 — Команда kick



Рисунок 18 — Сообщение удаленному пользователю

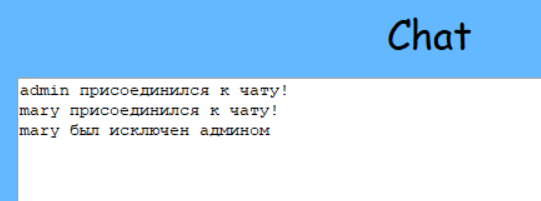


Рисунок 19 — Сообщение всем пользователям

Для того, чтобы добавить пользователя в черный список и удалить пользователя из чата навсегда, администратору необходимо ввести в поле ввода команду /ban nickname(никнейм пользователя) (Рис.20). И пользователь с таким никнеймом будет исключен из чата и добавлен в черный список, то есть в следующий раз он не сможет зайти под таким никнеймом в чат. Всем пользователям чата придет сообщение о том, что этот пользователь был исключен из чата (Рис.18) и он добавится в черный список. В текстовом файле bans.txt содержится список исключенных пользователей (Рис.21).

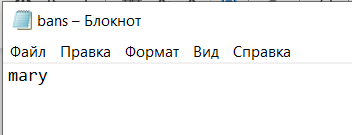
Рисунок 20 — Команда ban

Рисунок 21 — Файл bans.txt

## 4.2.3 Руководство обычного пользователя чата

Для того, чтобы обычный пользователь начал пользоваться чатом, ему необходимо запустить файл client.py, с помощью двойного клика по файлу или с помощью консоли (перейти по корректному пути нахождения файла client.py, ввести команду python client.py (подразумевается наличие на компьютере [интерпретатор Python](https://pythonru.com/baza-znanij/top-5-onlajn-kompiljatora-python)).

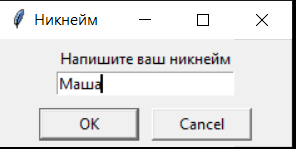
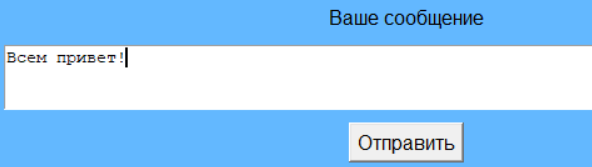
Далее в открывшемся диалоговом окне необходимо ввести никнейм и нажать кнопку «OK» (Рис. 22). После этого откроется главное модальное окно чата (Рис. 16).

Рисунок 22 — Ввод никнейма

Для того, чтобы отправить сообщение в чат необходимо написать его в поле ввода и надать на кнопку «Отправить» (Рис.23). Соответсвубщее собщение выведется у всех пользователей чата (Рис.24).



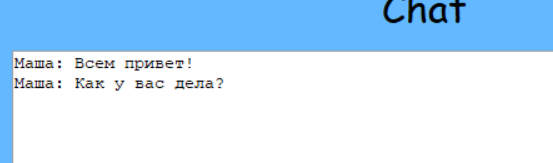
Рисунок 23 — Ввод сообщения

Рисунок 24 — Вывод сообщения в общем чате

Для того, чтобы покинуть чат, необходимо нажать на крестик, в правом верхнем углу. У всех пользователей чата выведется соответсвуещее сообщение (Рис.25).

Рисунок 25 — Вывод сообщения в общем чате

## 4.3. Экспериментальные данные тестирования процедур и функциональных задач

Для тестирования функций приложения и анализа, используемых приложением ресурсов, с клиента были отправлены сообщения. А также было проверено объем оперативной памяти, который могут занимать сразу несколько запущенных приложений.

Сначала была протестирована нагрузка на сеть. В первом тесте было отправлено одно сообщение (Рис.26).

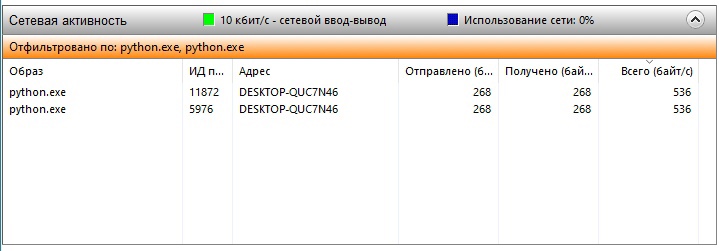
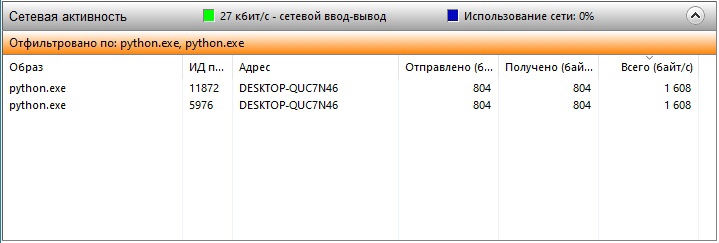


Рис. 26 — Сетевые ресурсы, затраченные при отправке одного сообщения

В этом тесте клиент отправил сообщение со скоростью 268 байт/с, сервер получил от клиента данные со скоростью 268 байт/с и отправил данные на все клиенты со скоростью 268 байт/с все клиенты получили данные со скоростью 268 байт/с.

Во втором тесте было отправлено 5 сообщений (Рис. 27).

Рис. 27 – Сетевые ресурсы, затраченные при отправке пяти сообщений

В этом тесте клиент отправил сообщение со скоростью 804 байт/с, сервер получил от клиента данные со скоростью 804 байт/с и отправил данные на все клиенты со скоростью 804 байт/с все клиенты получили данные со скоростью 804 байт/с.

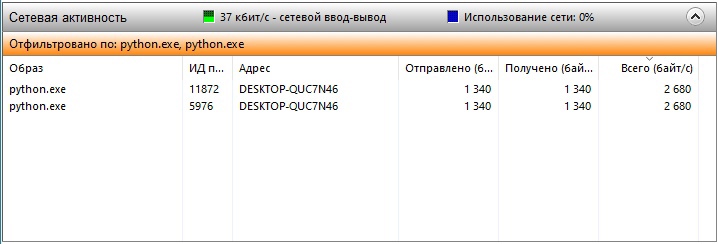
В третьем тесте было отправлено 10 сообщений (Рис.28).

Рис. 28 — Сетевые ресурсы, затраченные при отправке десяти сообщений

В этом тесте клиент отправил сообщение со скоростью 1340 байт/с, сервер получил от клиента данные со скоростью 1340 байт/с и отправил данные на все клиенты со скоростью 1340 байт/с все клиенты получили данные со скоростью 1340 байт/с.

Далее была протестирована нагрузка на оперативную память (Рис. 29).

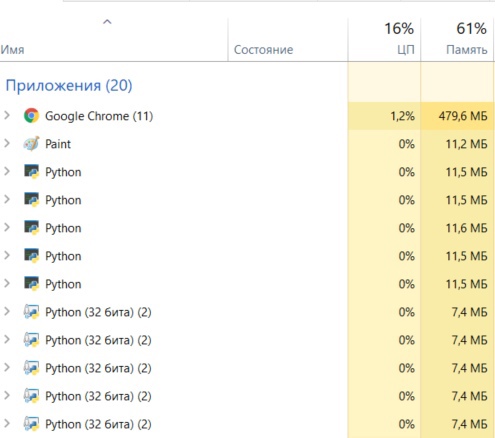


Рис. 29 — Нагрузка на оперативную память

В данном тесте на одном компьютере было запущено 5 клиентов. Было выяснено, что каждый клиент занимает 19 Мбайт оперативной памяти, а сервер занимает 11,4 Мбайт.

## 4.4. Разработка спецификации на оснащение подразделений, эксплуатирующих разработанное приложение, компьютерным и сетевым оборудованием

Рекомендуемые операционные системы:

Windows 7

Windows 8

Windows 8.1

Windows 10

Windows Server 2008 R2

Linux

MacOS

Рекомендуемая аппаратная конфигурация для сервера:

• Процессор Pentium 4 или более новый

• 200 МБ RAM для 64-битной сборки

• 40 KБ дискового пространства

Рекомендуемая аппаратная конфигурация для клиента:

• Процессор Pentium 4 или более новый

• 200 МБ RAM

• 200 МБ RAM для 64-битной сборки

• 40 KБ дискового пространства

Требования подключения к сети: Соединение клиента с сервером через витую пару третьей категории, подключение к сети интернет не обязательно. Минимальная скорость соединения: 50 кбит/с.

## Заключение

При выполнении курсовой работы на тему «Приложение-чат с возможностью обмена сообщениями» была исследована и описана предметная область, проведен анализ аналогов данной системы, а также выбор инструментов и платформы для разработки.

Проведен анализ объектов автоматизации и разработаны методы решения технических задач. Также была разработана и реализована структура и архитектура системы, структура системы передачи информации и интерфейса взаимодействия пользователя с системой. Также проведено тестирование и отладка приложения.

Результатом данной курсовой работы является рабочее приложение-чат с возможностью обмена сообщениями.

## Список использованных источников

1. Доусон М. Программируем на Python — СПб.:Питер, 2014 —418 с.: ил.;
2. [Д. Куроуз, К. Росс — Компьютерные сети. Нисходящий подход, 2016,](https://vk.com/doc306640305_440290462) 350 с.;
3. Олифер В. Г., Олифер Н. А. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 4-е изд. — СПб.: Питер, 2010. — 944 е.: ил. ;
4. Хахаев И. А. Практикум по алгоритмизации и программированию на Python: / И. А. Хахаев — М. : Альт Линукс, 2010. — 126 с. : ил.;
5. Чаплыгин А.Н. — Учимся программировать вместе с Питоном, 226 с.