

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**  
**(ФГБОУ ВО «КубГУ»)**  
**Физико-технический факультет**

**Кафедра теоретической физики и компьютерных технологий**

Допустить к защите  
Заведующий кафедрой  
д-р физ.-мат. наук, доцент  
\_\_\_\_\_ В.А. Исаев  
\_\_\_\_\_ 2021 г.

Руководитель ООП  
д-р физ.-мат. наук, доцент  
\_\_\_\_\_ В.А. Исаев  
\_\_\_\_\_ 2021 г.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА  
(МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)**

**РАЗРАБОТКА ЧАТ-БОТА ДЛЯ ПРЕДОСТАВЛЕНИЯ  
ОТВЕТОВ НА ЗАПРОСЫ**

Работу выполнила \_\_\_\_\_ Снопкова Алла Михайловна

Направление подготовки 09.04.02 Информационные системы и технологии

Направленность (профиль) Администрирование информационных систем

Научный руководитель  
канд. биол. наук, доцент \_\_\_\_\_ Н.Н.Куликова

Нормоконтролер, ст. преподаватель \_\_\_\_\_ Г.Д.Цой

Краснодар  
2021

## РЕФЕРАТ

Выпускная квалификационная работа (магистерская диссертация) 81 с., 23 рис., 3 табл., 11 листингов, 39 источн.

ЧАТ-БОТ, НЕЧЁТКИЙ ПОИСК, РАССТОЯНИЕ ЛЕВЕНШТЕЙНА, ВЕБ-ПРИЛОЖЕНИЕ, РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, КЛИЕНТСКАЯ ЧАСТЬ, СЕРВЕРНАЯ ЧАСТЬ, PYTHON, FLASK, БАЗА ДАННЫХ, MONGODB, NOSQL, ПРОГРАММА, ПРОГРАММНЫЙ ПРОДУКТ, СОЦИАЛЬНАЯ СЕТЬ

Объектом разработки данной выпускной квалификационной работы (данной магистерской диссертации) является чат-бот с нечётким поиском для социальных сетей.

Целью работы является разработка чат-бота для предоставления ответов на запросы.

В результате выполнения выпускной квалификационной работы (магистерской диссертации) разработан чат-бот с нечётким поиском для социальной сети ВКонтакте и веб-приложение для управления данным ботом.

# СОДЕРЖАНИЕ

Введение .....	4
1 Понятие и виды ботов .....	6
1.1 Вредоносные боты и типы вредоносных активностей .....	7
1.2 Чат-боты для мобильных и веб-приложений .....	9
1.3 Социальные медиа-боты .....	12
1.4 Сканер-боты и их назначения, виды и области применения .....	16
1.5 Файл robots.txt и его назначение и структура .....	21
2 Архитектура чат-бота для социальных сетей .....	29
2.1 Программный интерфейс приложения .....	31
2.2 Синтаксический анализ .....	37
2.3 Инструменты и средства для разработки ботов .....	40
2.4 Расстояние Левенштейна .....	42
2.5 SQL и NoSQL базы данных .....	49
2.6 Тестирование программного обеспечения и его классификация .....	53
3 Разработка чат-бота для предоставления ответов на запросы .....	56
3.1 Разработка веб-приложения для управления чат-ботом .....	61
3.2 Тестирование разработанного чат-бота с нечётким поиском .....	71
Заключение .....	76
Список использованных источников .....	77

## **ВВЕДЕНИЕ**

Актуальность темы. В настоящее время чат-боты являются неотъемлемой частью жизни людей и бизнеса. Они чрезвычайно полезны для компаний, которые стремятся создать сильный имидж бренда и обеспечить наилучшее обслуживание клиентов.

Более 50 % пользователей ожидают от брендов немедленного ответа в любое время суток. Для этого бизнесы запускают круглосуточную службу поддержки на своих сайтах, в социальных сетях и формируют целые колл-центры, тратя на это большие деньги. Но большинство вопросов пользователей типичные, и на них вполне может отвечать правильно настроенная программа – чат-бот. И пользователей чаще всего это устраивает – в среднем чат-бот отвечает от 30 до 40 % от всего количества заданных им вопросов, а хорошо настроенные чат-боты выдают ответы в 90 % случаев.

Чат-бот создает ощущение, что бизнес всегда на связи с клиентами. И если робот решает все проблемы пользователя, присутствие человека не нужно. Поэтому компаниям дешевле всего внедрить чат-бота в свою службу поддержки, чем нанять человека на постоянную основу.

Исходя из этого, разработка чат-бота для предоставления ответов на запросы в социальных сетях представляет собой весьма актуальную задачу в области информационных технологий и систем.

Цель работы – разработка чат-бота для предоставления ответов на запросы.

Для достижения этой цели необходимо решить следующие задачи:

- изучить теоретические основы об интернет-ботах;
- ознакомиться с архитектурой чат-ботов для социальных сетей;
- изучить средства и инструменты разработки чат-ботов, а также способы их подключения к социальным сетям;
- освоить принцип разработки чат-ботов с нечётким поиском;

– разработать чат-бот для предоставления ответов на запросы для групп социальной сети ВКонтакте;

– разработать веб-приложение для управления чат-ботом для предоставления ответов на запросы для сообществ в социальной сети ВКонтакте.

Объект исследования в данной диссертации – интернет-боты.

Предмет исследования – чат-боты с нечётким поиском.

Новизна диссертации заключается в создании чат-бота с нечётким поиском с помощью эффективной, гибкой и мобильной технологии, которая не требует много ресурсов для её работы.

На защиту выносятся следующие положения:

1 Программная реализация чат-бота для предоставления ответов на запросы.

2 Программная реализация веб-приложения для управления чат-ботом для предоставления ответов на запросы.

Теоретическая и практическая значимость исследования состоит в том, что на данный момент в сообществах социальной сети ВКонтакте не предусмотрен автоответчик на часто задаваемые вопросы посетителей и участников групп.

Апробация материалов диссертации. По результатам работы опубликованы две статьи на следующие темы:

1 «Реализация оптимального алгоритма для расчета расстояния Левенштейна».

2 «Командный чат-бот для социальных сетей на языке программирования Python».

Структура работы. Диссертационная работа состоит из введения, трёх глав, заключения и списка использованных источников.

## **1 Понятие и виды ботов**

Бот – это программное приложение, в котором запрограммированы ряд функций для выполнения определённых задач. Боты автоматизированы – это значит, что они работают в соответствии со своими инструкциями, не требующие непосредственного вмешательства человека в их работу. Как правило, они выполняют систематические задачи, и работают гораздо быстрее, чем люди [1].

Боты чаще всего работают по сети. Более половины интернет-трафика – это боты, которые сканируют контент веб-страниц, общаются с людьми или ищут новые цели для атак на пользователей. Некоторые боты полезны. Например, боты поисковых систем, индексирующие контент для поиска полезной информации, или боты для обслуживания клиентов, помогающие пользователям в решении их ряда вопросов. Другие боты являются «плохими». Они запрограммированы, чтобы взламывать учётные записи пользователей, чтобы сканировать веб-страницы в поисках контактной информации для рассылки спама или выполнять другие вредоносные действия. Когда пользователь подключается к интернету, у бота появляется соответствующий IP-адрес.

Боты бывают четырех видов:

- чат-боты – боты, которые имитируют человеческую беседу, отвечая на определенные фразы уже запрограммированными ответами;
- сканер-боты – боты, которые сканируют контент на веб-страницах по всему интернету;
- социальные медиа-боты – боты, работающие на платформах социальных сетей;
- вредоносные боты – боты, которые сканируют контент, распространяют спам-информацию, а также могут выполнять атаки на базу

данных учетных записей пользователей [2].

### 1.1 Вредоносные боты и типы вредоносных активностей

Любые автоматизированные действия бота, нарушающие прямое назначение интернет-портала, условия предоставления услуг сайта или инструкции файла robots.txt можно считать вредоносными (рисунок 1).

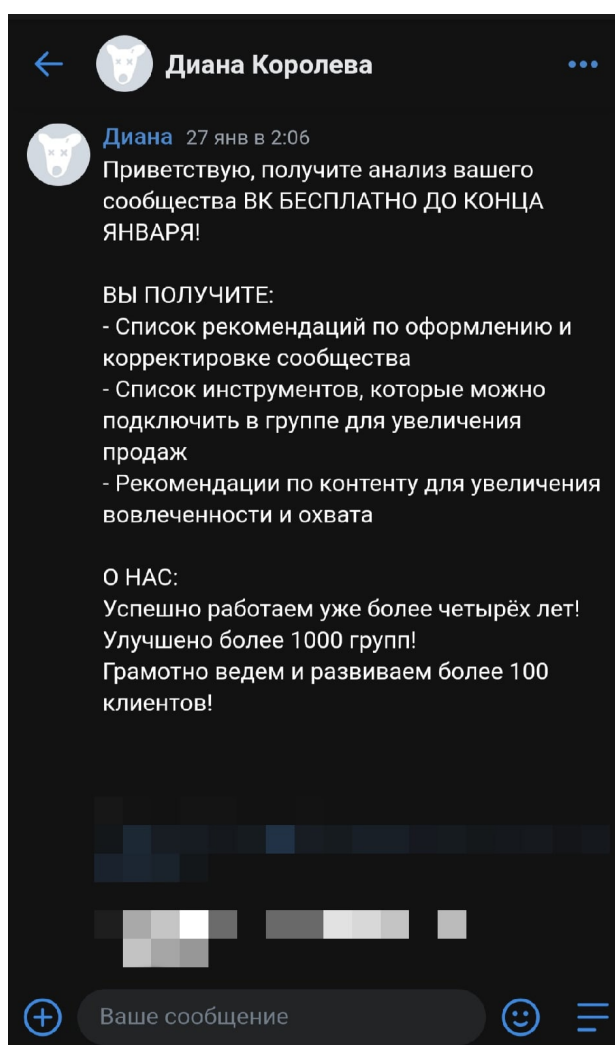


Рисунок 1 – Вредоносный бот, предназначенный для рассылки спама

Боты, которые пытаются совершать киберпреступления, такие как кража личных данных, захват учетных записей, также являются «плохими» ботами. Боты априори не должны нарушать закон.

Кроме того, чрезмерный трафик ботов может нагружать ресурсы веб-сервера, замедляя или останавливая обслуживание законопослушных пользователей, пытающиеся использовать веб-сайт или приложение в благих целях. Иногда это происходит намеренно и принимает форму DoS- или DDoS-атаки.

DoS- или DDoS-атака (Distributed Denial of Service attack) – комплекс действий, способный полностью или частично вывести из строя работу сервера [3].

Вредоносная активность бота включает в себя:

- кража учётных данных;
- чистка веб-страниц/контента;
- DoS или DDoS-атаки;
- несанкционированный взлом пароля;
- сбор информации;
- размещение спам-контента;
- взлом электронной почты;
- интернет-мошенничество.

Чтобы осуществить все эти атаки и замаскировать источник атакующего трафика, «плохие» боты могут тиражироваться, распространять свои копии и работать одновременно на нескольких устройствах, чаще всего без ведома владельцев данных устройств. Поскольку каждое устройство имеет свой собственный IP-адрес, трафик бота поступает с огромного количества различных IP-адресов, что затрудняет идентификацию и блокировку источника вредоносного бот-трафика.

Для того чтобы остановить действия вредоносной активности бота, существуют решения для управления ботами, которые способны отделять



вредную активность от деятельности пользователя и полезную активность бота с помощью машинного обучения. Решения для управления ботами должны быть способными идентифицировать и блокировать вредоносных ботов на основе поведенческого анализа, который обнаруживает аномалии, и при этом позволять полезным ботам получать доступ к веб-сервису.

## 1.2 Чат-боты для мобильных и веб-приложений

Чат-боты – это компьютерные программы, которые предназначены для взаимодействия с людьми с помощью голосовых и/или текстовых сообщений (рисунок 2) [4].

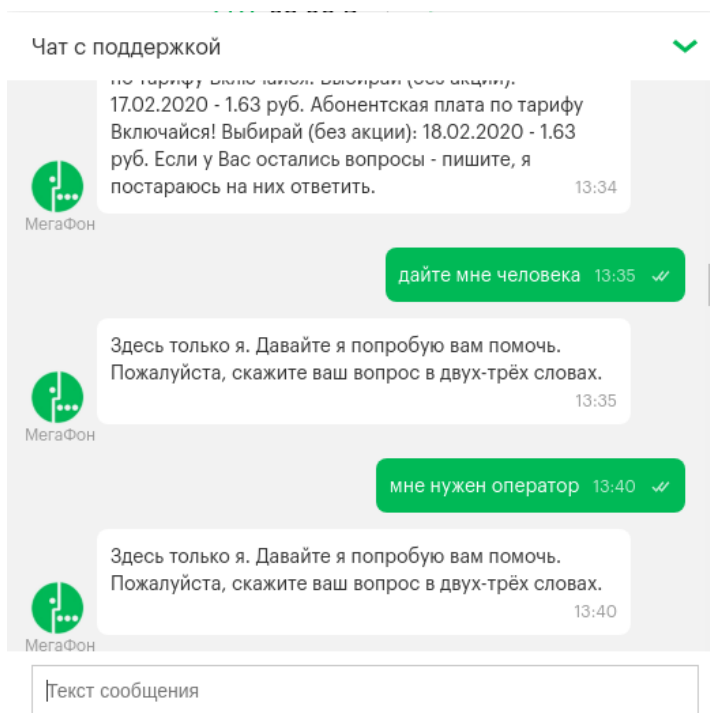


Рисунок 2 – Диалог с чат-ботом компании Мегафон

Чат-боты применяются во многих полезных приложениях: цифровые персональные помощники и чаты технической поддержки. Чат-боты также

могут использоваться в незаконных целях, например, рассылка спама.

С точки зрения функциональности существует два основных типа чат-ботов [5]:

- чат-боты с нечётким поиском;
- чат-боты с искусственным интеллектом.

Чат-боты с нечётким поиском – это боты, которые предоставляют ответы на основе набора фиксированных правил, созданных разработчиками. Они обычно сканируют сообщение пользователя на наличие ключевых слов и по ним предоставляют ответы.

Например, человек может сказать чат-боту с нечётким поиском: «Я очень рад предстоящему баскетбольному матчу». Бот определяет ключевые слова, такие как «баскетбол» и «игра», а затем отправляет ответ: «Скажите мне свою любимую спортивную команду».

Хоть чат-боты с нечётким поиском действуют по простому набору правил, эти боты достаточно эффективно могут пройти тест Тьюринга для многих базовых взаимодействий. Но чат-бот с искусственным интеллектом может быть более убедительным, когда речь заходит о глубоком разговоре.

Тест Тьюринга, созданный известным компьютерным ученым Аланом Тьюрингом, является тестом для выявления способности машины демонстрировать человеческое поведение. Если человек не может сказать, общается ли он с человеком или с машиной, эта машина, как говорят, проходит тест Тьюринга.

Чат-боты с искусственным интеллектом используют машинное обучение, которое обучается во время общения с людьми. Они предназначены для понимания языка, а не только для определения ключевых слов. Их привлекательность заключается в том, что они могут быть гораздо более эффективными в имитации реалистичных человеческих разговоров. Но чат-боты с искусственным интеллектом также значительно сложны в разработке и управлении. Кроме того, недостатки в разработке таких ботов

могут привести к некоторым странным и нежелательным поведениям. Одним из известных примеров был «Tay» от Microsoft. Tay был чат-ботом с искусственным интеллектом, выпущенным в Twitter в 2016 году, который начал делать оскорбительные и подстрекательские комментарии в течение 24 часов после запуска (частично это было связано с несколькими пользователями Twitter, которые быстро научились влиять на поведение бота).

В настоящее время чат-ботов чаще всего можно встретить в приложениях технической поддержки клиентов и программах с цифровым персональным помощником. Есть также несколько других применений, которые только набирают популярность.

Чат-боты технической поддержки клиентов – эти боты часто встречаются на веб-сайтах и веб-приложениях различных компаний. Их возможности варьируются от помощи в ориентировании пользователей на веб-странице каталога с продуктами или услугами до решения проблем клиентов через сложные технические ветвления. Эти боты могут с легкостью заменить работников колл-центра.

Цифровые персональные помощники – это голосовые чат-боты, предназначенные для выполнения административных задач, таких как создание календарных встреч или поиск в интернете, чтобы найти информацию, допустим, о погоде. Примерами таких помощников являются Яндекс.Алиса, Маруся от Мейл.ру, Джой и Афина от Сбербанка, Google Assistant, Amazon Alexa и Apple Siri.

Многие помощники предоставляют дополнительные полезные функции, такие как игры, музыку, управление умным домом (к примеру кофеваркой с поддержкой Wi-Fi) и даже обеспечивают дружеское общение с людьми.

Есть еще несколько новых способов использования чат-ботов, которые набирают все большую популярность. К ним относятся:

– боты психологической поддержки – это боты, предназначенные для психологической помощи клиентов. Они обеспечивают дружеское общение, а также дают пользователям возможность поделиться своими проблемами;

– развивающие игры – на рынке существует высокий спрос на игрушки со встроенным чат-ботом, предназначенных для развития у детей различных навыков. Например, кукла Барби со встроенным чат-ботом.

Существуют следующие виды вредоносных чат-ботов:

– спам чат-боты – это чат-боты в социальных сетях, предназначенные для поиска людей или групп людей, которым можно отправить в сообщения нежелательный контент;

– чат-боты домогатели – чат-боты, которые используются для онлайн-травли. Эти боты могут прислать множество неприятных сообщений и комментариев пользователю, пытаясь его запугать;

– чат-бот дезинформатор – это боты, предназначенные для распространения ложной информации через сообщения в социальных сетях и электронную почту.

Для того чтобы идентифицировать вредоносность чат-ботов, существуют службы управления ботами на разных платформах, которые используются для отслеживания активности вредоносных ботов и предотвращают их попадание на веб-сайт или в приложение. Имеются достаточно много сервисов, которые могут помочь идентифицировать и ликвидировать незаконных ботов.

### **1.3 Социальные медиа-боты**

Социальные медиа-боты – это автоматизированные программы, которые встречаются в мессенджерах и социальных сетях (рисунок 3) [6]. Эти боты работают частично или полностью автономно и предназначены для имитации людей-пользователей. Многие социальные медиа-боты

используются в незаконных целях. По некоторым оценкам, такие вредоносные боты составляют значительный процент всех аккаунтов в социальных сетях.

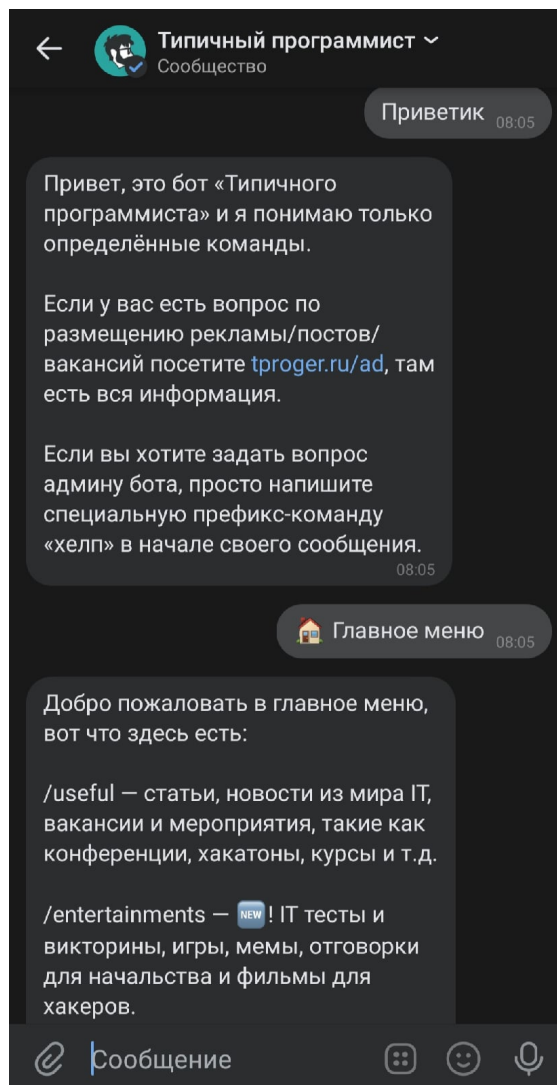


Рисунок 3 – Диалог с медиа-ботом в социальной сети ВКонтакте

Боты для социальных сетей чем-то похожи на чат-ботов. Чат-боты – это боты, которые могут самостоятельно вести разговор, в то время как медиа-боты не обладают такой способностью. На самом деле многие медиа-боты вообще не ведут диалог с людьми; они только выполняют простые команды пользователя, такие как предоставление большого

количества «подписчиков» и «лайков» и так далее.

В некоторых социальных сетях (ВКонтакте и Telegram) имеется целый набор инструментов для создания собственного бота. Чаще всего трафик таких ботов проверяется администрацией сети, поэтому вредоносные боты в социальных сетях скрываются под видом обычного профиля (их ещё называют юзерботами). Такие боты могут быть применяться для достижения следующих целей:

- искусственно увеличивать популярность блогера или целого движения – человек или организация с миллионами подписчиков в социальных сетях могут считаться важными и влиятельными. Основной пример использования ботов в социальных сетях – это повышение популярности других аккаунтов. Эти боты могут быть куплены и проданы в даркнете, причем более убедительные боты получают более высокую цену;

- манипулирование финансовыми рынками – социальные медиа-боты могут влиять на финансовые рынки. Например, аккаунты ботов могут обеспечивать ленту социальных сетей заранее сфабрикованными хорошими или плохими новостями о корпорации, пытаясь манипулировать ценами на акции;

- усиление фишинговых атак – фишинговая атака направлена на то, чтобы добиться доверия своей жертвы. Поддельные подписчики в социальных сетях и социальная вовлеченность могут помочь убедить жертву в том, что их мошеннику можно доверять;

- рассылка спама – социальные медиа-боты часто используются в незаконных рекламных кампаниях, распространяя спам в социальной сети со ссылками на коммерческие сайты.

Иногда встает вопрос о том, сколько аккаунтов в социальных сетях на самом деле являются ботами. Руководители Twitter представили отчёт перед Конгрессом, что до 5 % аккаунтов Twitter управляются ботами. Эксперты, применявшие алгоритмы, предназначенные для определения поведения

ботов, обнаружили, что это число может быть приравнено к 15 % [7].

Трудно определить, сколько точно аккаунтов в социальных сетях являются ботами, поскольку очень многие из ботов довольно хорошо имитируют действия человеческого аккаунта. Во многих случаях люди не могут просто так отличить аккаунты ботов от настоящих учетных записей людей. Тем более нет ни одного надежного способа идентифицировать профили многофункциональных ботов. Исследование, проведенное Школой системной инженерии Университета Рединга, показало, что 30 % участников исследования могут быть обмануты, полагая, что аккаунт бота для социальной сети управляется реальным человеком [8].

Также в некоторых случаях боты могут использовать реальные учётные записи пользователей, которые ранее были захвачены или куплены злоумышленником. Эти захваченные или купленные аккаунты ботов имеют очень убедительные фотографии, истории постов и социальные сети.

Несмотря на то, что некоторые из самых продвинутых ботов для социальных сетей могут быть не распознаны даже экспертами, все же существует несколько стратегий для идентификации ботов. Например:

- анализ фотографии профиля в поиске изображения, для того чтобы убедиться, что аккаунт не использует чужую фотографию, взятую из интернета;

- внимание на дату публикации постов в профиле – если они публикуют сообщения в то же время, когда большая часть постов была опубликована, или присылают сообщения через каждые несколько минут изо дня в день, это указывает на то, что учётная запись автоматизирована;

- обращение в службу обнаружения ботов.

К сожалению, нет простого способа избавиться от всех вредоносных ботов в социальных сетях. Некоторые люди призывают платформы социальных сетей применять более строгие требования к созданию учётных записей, но социальные платформы не решаются делать это, потому что:

– это может помешать некоторым законным пользователям зарегистрироваться, и к тому же штаб-квартиры социальных сетей используют количество учётных записей пользователей как меру своей популярности;

– так как не существует идеального способа для определения кто есть кто, более строгие требования к созданию учётных записей могут вызвать неудобства для регистрации реальных пользователей, и при этом не останавливая самих же ботов. Например, недавно встал вопрос о том, что капчи являются весьма хорошим сдерживающим фактором против ботов, но они, безусловно, приносят некоторые неудобства людям.

Несмотря на то, что социальные сети могут блокировать все найденные вредоносные боты, пользователь должен понимать, что они являются постоянной проблемой и поэтому необходимо быть бдительным в любых ситуациях.

#### **1.4 Сканер-боты и их назначения, виды и области применения**

Веб-искатель, сканер-бот или бот поисковой системы загружает и индексирует контент во всем интернете. Цель такого бота состоит в том, чтобы узнать, о чем та и или иная веб-страница. Их называют веб-сканерами, потому что они получают автоматический доступ к веб-сайту и извлекают оттуда данные с помощью сканирования веб-ресурса [9].

Некоторые боты управляются поисковыми системами. Применяя алгоритм поиска к данным, собранным веб-сканерами, поисковые системы могут предоставлять релевантные ссылки в ответ на поисковые запросы пользователей, генерируя список веб-страниц, которые появляются после того, как пользователь вводит запрос в Google или Яндекс (или другую поисковую систему).

Бот-искатель похож на человека, который просматривает все книги в



неорганизованной библиотеке и составляет каталог, чтобы любой, кто посещает данную библиотеку, мог быстро и легко найти нужную ему информацию. Чтобы помочь классифицировать и сортировать книги в библиотеке по темам, организатор прочитает название, краткое содержание и часть внутреннего текста каждой книги.

Однако, в отличие от библиотеки, интернет не состоит из физических стопок книг, и из-за этого трудно сказать, была ли вся необходимая информация проиндексирована должным образом, или же огромное количество данных с веб-сайта было упущено. Чтобы попытаться найти всю необходимую информацию, которую может предложить интернет, бот-искатель начнет с определенного набора известных ему веб-ресурсов, а затем перейдет по гиперссылкам с этих страниц на другие страницы, после по гиперссылкам с этих других страниц на дополнительные страницы и так далее.

Поисковая индексация похожа на создание каталога библиотечных карточек для интернета, чтобы поисковая система знала, где можно получить информацию, когда человек её ищет. Его также можно сравнить с указателем в конце книги, в котором перечислены все места в тексте, где упоминается определенная тема или фраза.

Индексация фокусируется в основном на тексте, который появляется на странице, и в метаданных о странице, которую пользователи не видят. Когда большинство поисковых систем производят индексацию страницы, они добавляют все слова на странице в индекс-массив – за исключением таких слов, как «а», «an» и «the» в случае с Google. Когда пользователи ищут эти слова, поисковая система просматривает свой индекс-массив всех страниц, где эти слова имеются, и выбирает наиболее релевантные веб-ресурсы [10].

В контексте поисковой индексации метаданные – это данные, которые сообщают поисковым системам, о чем та или иная веб-страница. Часто мета-заголовок и мета-описание – это то, что будет отображаться на

страницах результатов поиска, в отличие от содержимого веб-страницы, видимого пользователям.

Интернет постоянно меняется и расширяется, поэтому невозможно узнать, сколько всего веб-страниц существует в интернете, боты-искатели веб-сайтов начинают поиск со списка известных ему URL-адресов. Сначала они сканируют веб-страницы по этим URL-адресам. При обходе этих веб-страниц они находят гиперссылки на другие URL-адреса и добавляют их в список страниц для следующего сканирования.

Учитывая огромное количество веб-страниц в Интернете, которые могут быть проиндексированы для поиска, этот процесс может продолжаться бесконечно. Однако веб-искатель будет следовать по определенным правилам, которые делают его всё более избирательным в отношении того, какие страницы следует сканировать, в каком порядке и как часто они должны сканировать их заново, чтобы проверить наличие обновлений контента.

Нужно понимать, что большинство веб-искателей не сканируют весь общедоступный интернет и собственно для этого не предназначены; зачастую они сначала решают, какие страницы нужно сканировать в первую очередь, основываясь на количестве других страниц, которые ссылаются на данную страницу, количестве посетителей этой страницы и другие факторы, которые указывают на вероятность того, что страница содержит важную информацию.

Идея заключается в том, что веб-страница, на которую ссылаются многие другие веб-страницы и у него много посетителей, скорее всего, будет содержать высококачественную, авторитетную информацию, поэтому особенно важно, чтобы поисковая система проиндексировала именно его – точно так же, как библиотека может позаботиться о том, чтобы сохранить множество копий книги, которую читают большое количество людей.

Повторное посещение веб-страниц, контент в интернете постоянно обновляется, удаляется или перемещается в новые места. Веб-искателям

периодически необходимо пересматривать страницы, чтобы убедиться, что последняя версия контента была проиндексирована.

Правила поведения бота (robots.txt) – веб-искатели также решают, какие страницы следует сканировать, а какие нет, основываясь на протокол поведения ботов (также известный как протокол исключения robots.txt). Перед сканированием веб-страницы они проверяют файл протокола robots.txt, размещенный на веб-сервере данного ресурса. Протокол исключения ботов (или robots.txt) – это текстовый файл, который просматривает правила поведения для любых ботов, получающих доступ к размещенному веб-сайту или приложению. Эти правила показывают, какие страницы могут сканировать боты и по каким ссылкам они могут переходить [11].

Все вышеперечисленные факторы по-разному взвешиваются при разработке собственных алгоритмов, на основе которых каждая поисковая система создавала своих ботов. Веб-сканеры разных поисковых систем будут вести себя немного по-разному, хотя конечная цель одна и та же – загружать и индексировать контент с веб-страниц.

Чаще всего разработчики таких веб-искателей называют «пауками». Интернет, или, по крайней мере, та его часть, к которой обращается большинство пользователей, также имеет альтернативное название «Всемирная паутина» – фактически именно отсюда и происходит «www», часть структуры большинства URL-адресов веб-сайтов. Было вполне естественно называть поисковых роботов «пауками», потому что они ползают по всей сети также, как настоящие пауки ползают по своей паутине.

Боты-искатели не всегда имеют доступ ко всем веб-ресурсам. Это зависит как и от самого веб-ресурса, так и от ряда других факторов. Веб-искатели используют данные с серверов для индексации контента – они формируют запрос, чтобы получить ответ от сервера. В зависимости от количества информации на каждой странице или количества страниц на сайте, возможно, что это будет в интересах администратору веб-сайта не

допускать слишком частого индексирования для поиска, поскольку слишком большая индексация может перегружать сервер, увеличивать затраты на пропускную способность или и то, и другое.

Кроме того, разработчики или компании могут не захотеть, чтобы некоторые веб-страницы были доступны для поиска. Один из примеров такого случая для предприятий – когда они создают специальную целевую страницу для маркетинговой кампании, но они не хотят, чтобы тот, на кого не была изначально нацелена данная кампания, получил доступ к этой странице. Таким образом, предприятие может добавить тег «No index» на целевую страницу, и он не будет отображаться в результатах поиска. Они также могут добавить тег «Disallow» на странице или в файл robots.txt, и поисковые пауки вообще не смогут его сканировать.

Также владельцы веб-сайтов могут не захотеть, чтобы боты-искатели сканировали веб-страницы по целому ряду других причин. Например, веб-сайт, который предлагает пользователям возможность поиска внутри ресурса, может заблокировать страницы некоторых результатов поиска, поскольку они будут не нужны для большинства пользователей. Другие автоматически сгенерированные страницы, которые могут быть полезны только для одного пользователя или нескольких конкретных пользователей, также должны быть заблокированы.

Помимо веб-сканера, существуют также веб-собиратели.

Веб-собиратель, собиратель данных или собиратель контента – это бот, который загружает информацию с веб-сайта без разрешения, чтобы в дальнейшем использовать этот контент в злонамеренных целях [12].

Такие боты чаще всего не обращают внимания на нагрузку, которую они создают на веб-серверах, в то время как веб-сканеры, особенно из крупных поисковых систем, будут подчиняться инструкциям файла robots.txt на сайте и ограничить их запросы, чтобы не перегружать данный веб-сервер.

Веб-сканеры отлично влияют на SEO.

SEO (Search Engine Optimization) – это поисковая оптимизация, дисциплина подготовки контента для индексации поиска, чтобы веб-сайт показывался выше в результатах поиска [13].

Если боты-пауки не сканируют веб-сайт, то он не может быть проиндексирован и не будет отображаться в результатах поиска. По этой причине, если владелец сайта хочет получить популярность со стороны пользователей интернета из результатов поиска, очень важно, чтобы он не блокировал веб-искателей.

Боты из основных поисковых систем называются, которые активны по сей день:

- Google – Googlebot (на самом деле существует два сканера, Googlebot Desktop и Googlebot Mobile, для десктопного и мобильного поиска);
- Bing – Bingbot;
- Яндекс – YandexBot;
- Baidu (китайская поисковая система) – BaiduSpider.

Есть также множество менее распространенных ботов-пауков, некоторые из которых не связаны ни с одной поисковой системой.

Для управления ботом важно учитывать шаги поиска по сети. «Плохие» боты могут нанести большой ущерб – от банального каждодневного отсутствия посетителей сайта до сбоев сервера и кражи данных. Однако при блокировании «плохих» ботов важно по-прежнему разрешать «хорошим» ботам, например, веб-искателям, доступ к веб-ресурсам своего сайта.

### **1.5 Файл robots.txt и его назначение и структура**

Robots.txt – это набор инструкций для поведения ботов. Этот файл включен в исходные файлы большинства веб-сайтов. Файл robots.txt в основном предназначены для управления действиями «хороших» ботов (веб-сканерами) поскольку «плохие» боты вряд ли будут следовать этим

инструкциям.

Файл robots.txt можно сравнить с правилами поведения, вывешенными на стене в спортзале, баре или общественном центре – сами по себе они не имеют силы навязывать людям все перечисленные правила, но хорошие посетители будут им следовать, в то время как плохие, что скорее всего, нарушат их и получают штраф.

Веб-искатель сканирует веб-страницы и индексирует их контент, чтобы они в дальнейшем могли отображаться в результатах поиска. А файл robots.txt помогает управлять деятельностью этих веб-искателей, чтобы они не перегружали веб-сервер, на котором размещен сканируемый веб-сайт, и не индексировали страницы, которые не предназначены для публичного просмотра.

Robots.txt – это просто текстовый файл без HTML-разметки (отсюда и расширение \*.txt). Файл robots.txt размещается на веб-сервере так же, как и любой другой файл веб-сайта. Чтобы просмотреть содержимое файла robots.txt абсолютно любого веб-ресурса, необходимо ввести полный URL-адрес главной страницы, а затем добавить «/robots.txt» (рисунок 4).

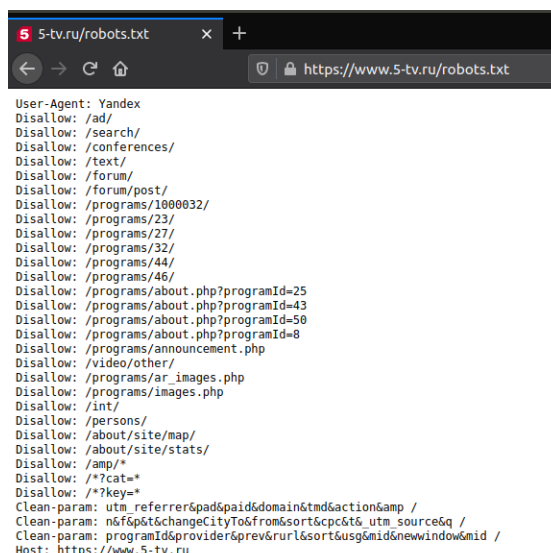


Рисунок 4 – Файл robots.txt на сайте Пятого канала

Несмотря на то, что файл robots.txt содержит все необходимые правила для ботов, он не может гарантировать выполнение всех этих инструкций. «Хороший» бот, такой как веб-искатель или бот для составления новостной ленты, сначала попытается посетить файл robots.txt, и только потом начнет сканировать страницы в домене, следуя всем инструкциям данного протокола. «Плохой» бот либо проигнорирует robots.txt, либо будет следовать ему, но только для того, чтобы найти веб-страницы, которые запрещены для индексации поисковых систем.

Веб-искатель всегда следует конкретному набору инструкций в файле robots.txt. Если в файле есть противоречивые команды, то бот будет следовать наиболее детализированной операции.

Важно также отметить, что и поддомены нуждаются в наличии собственных файлов robots.txt.

Файл robots.txt состоит из ряда протоколов. Протокол в сети интернет – это формат для представления инструкций или команд. Файлы robots.txt могут содержать несколько различных протоколов. Основным протоколом в таких файлах является Robots Exclusion Protocol. Это такой способ сказать ботам, каких веб-страниц и ресурсов нужно избегать.

Другой протокол, используемый в файле robots.txt – это протокол Sitemaps. Sitemaps показывает веб-искателю, какие страницы они могут сканировать. Это гарантирует то, что веб-искатель не пропустит ни одной важной страницы.

Рассмотрим файл robots.txt на примере сайта [www.5-tv.ru](http://www.5-tv.ru) (рисунок 5). И разберем каждую строку данного файла подробнее [14].

«User-agent: GoogleBot». Любой пользователь или бот, который активен в интернете, будет иметь инструкцию «User-agent». Для обычных пользователей инструкция включает в себя такую информацию, как тип браузера и версию операционной системы; это помогает веб-ресурсам

показывать контент, совместимый с системой пользователя.

```
User-Agent: GoogleBot
Disallow: /ad/
Disallow: /search/
Disallow: /conferences/
Disallow: /text/
Disallow: /forum/
Disallow: /forum/post/
Disallow: /programs/1000032/
Disallow: /programs/23/
Disallow: /programs/27/
Disallow: /programs/32/
Disallow: /programs/44/
Disallow: /programs/46/
Disallow: /programs/about.php?programId=25
Disallow: /programs/about.php?programId=43
Disallow: /programs/about.php?programId=50
Disallow: /programs/about.php?programId=8
Disallow: /programs/announcement.php
Disallow: /video/other/
Disallow: /programs/ar_images.php
Disallow: /programs/images.php
Disallow: /int/
Disallow: /persons/
Disallow: /about/site/map/
Disallow: /about/site/stats/
Disallow: /*?cat=*
Disallow: /*?key=*
Allow: /news/

Sitemap: https://www.5-tv.ru/sitemap-video.xml
Sitemap: https://www.5-tv.ru/sitemap-video-items.xml
Sitemap: https://www.5-tv.ru/sitemap-video-films.xml
Sitemap: https://www.5-tv.ru/sitemap-google-news.xml
Sitemap: https://www.5-tv.ru/sitemap-rubric.xml
Sitemap: https://www.5-tv.ru/sitemap-player.xml
Sitemap: https://www.5-tv.ru/sitemap-tags.xml
Sitemap: https://www.5-tv.ru/sitemap-amp.xml
```

Рисунок 5 – Фрагмент robots.txt на сайте Пятого канала

В файле robots.txt администраторы веб-сайтов могут внести конкретные команды для конкретных ботов, вписав инструкции под тэгом «User-agent» для различных веб-искателей. Например, если администратор хочет, чтобы определенная страница отображалась в результатах поиска Google (как показано на примере на рисунке 5), но не в результатах поиска Yandex, то он может включить в файл robots.txt два набора команд – один набор, которому предшествует конструкция «User-agent: Yandex», а другому набору



предшествует конструкция «User-agent: Googlebot».

Файл robots.txt может также включать в себя команду «User-agent: \*». В этом случае звездочка означает, что весь набор команд данной конструкции применим к каждому боту, а не к какому-либо конкретному веб-искателю.

Общие имена агентов пользователей ботов поисковых систем включают в себя:

а) Google:

- 1) Googlebot;
- 2) Googlebot-Image (для картинок);
- 3) Googlebot-News (для новостей);
- 4) Googlebot-Video (для видео);

б) Bing:

- 1) Bingbot;
- 2) MSNBot-Media (для картинок и видео);

в) Yandex:

- 1) YandexBot;
- 2) YandexNews (для новостей);
- 3) YandexMedia (для мультимедиа);
- 4) YandexMetrika (для метрики).

«Disallow: /ad/». Команда Disallow является самой распространенной инструкцией в протоколе исключения ботов. Она дает понять веб-искателю, что не стоит обращаться к той или иной веб-странице или набору веб-страниц при сканировании веб-ресурса. Запрещённые для поискового бота страницы вовсе не обязательно должны быть скрыты от посторонних людей – они просто бесполезны для обычного пользователя Google или Yandex, поэтому они им не показываются. В большинстве случаев пользователь веб-сайта все еще может перейти на эти страницы, если он знает, где их найти.

Команду Disallow можно объявлять несколькими способами, некоторые

из которых показаны в приведенном выше примере.

Для того чтобы заблокировать один файл или одну конкретную страницу веб-сайта, например, телепрограмму Пятого канала, необходимо прописать следующую инструкцию – «Disallow: /schedule/». После команды «Disallow» должна прописываться та часть URL-адреса, которая идет сразу после главной страницы – в данном случае «www.5-tv.ru». После объявления такой инструкции «хорошие» боты больше не получают доступа <https://www.5-tv.ru/schedule/>, и страница не будет отображаться в результатах поиска.

Иногда лучше всего блокировать сразу несколько страниц, вместо того чтобы перечислять их все по отдельности. Если все они находятся в одном разделе веб-сайта, то в файле robots.txt может просто заблокировать целый каталог, в котором содержатся все эти веб-страницы. Например, «Disallow: /video/other/». Это означает, что все страницы, содержащиеся в каталоге /video/other/, не должны сканироваться веб-искателем.

Также с помощью такой команды можно дать полный доступ бот-искателям. Такая инструкция будет выглядеть следующим образом – «Disallow: ». Это означает, что боты могут просматривать весь веб-ресурс, потому что никакие страницы не были заблокированы администрацией сайта.

Чтобы скрыть весь сайт от ботов, можно воспользоваться следующей командой – «Disallow: /» Символ «/» в данной инструкции представляет «корень» иерархии веб-сайта. Такой корень включает в себя домашнюю страницу и все страницы, которые с ней связаны. С помощью этой команды поисковые боты вообще не смогут сканировать ваш веб-сайт. Другими словами, одна косая черта может исключить целый веб-ресурс из поиска в интернете.

«Allow: /news/». Как и следовало ожидать, команда «Allow» сообщает ботам, что им разрешен доступ к определенной веб-странице или каталогу. Эта команда позволяет ботам добраться до одной конкретной веб-страницы, в

то время как остальные веб-страницы в файле запрещены. Не все поисковые системы распознают эту команду.

«Crawl-delay». На примере данная инструкция не была продемонстрирована, но ее стоит упомянуть. Это команда задержки сканирования веб-искателя. Она предназначена для того, чтобы веб-искатели не нарушили работу сервера. Такая команда позволяет администраторам указывать период задержки бота между каждым запросом, в миллисекундах. Вот пример команды «Crawl-delay» для задержки работы веб-сканера в 8 миллисекунд – «Crawl-delay: 8».

Поисковая система Google не распознает эту команду, в отличие от других поисковых систем. Для Google администраторы могут изменять частоту сканирования своего веб-сайта в консоли настроек системы на сайте Google.

«Sitemap: <https://www.5-tv.ru/sitemap-video-items.xml>». Протокол Sitemaps помогает веб-искателям понять, что можно включать в сканирование веб-сайта.

Sitemap – это XML-файл, структура которого представлена на рисунке 6. Это машиночитаемый список всех страниц сайта. С помощью протокола Sitemaps ссылки на эту «карту сайта» могут быть включены в файл robots.txt. Сначала прописывается конструкция «Sitemaps: », а следом указывается полный веб-адрес XML-файла.

Протокол Sitemaps не может гарантировать, что веб-искатели ничего не пропустят при сканировании веб-сайта. Sitemap не может изменить приоритет сканирования сайта ботом.

Файл robots.txt не только помогает администратору поднять популярность своего веб-ресурса, но и принимает немалое участие в управлении ботами поисковых систем.

```

<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9" xmlns:video="http://www.google.com/schemas/sitemap-video/1.0">
  <url>
    <loc>https://www.5-tv.ru/video/1025783/</loc>
    <video:video>
      <video:content_loc>https://vod5tv.cdnvideo.ru/shared/files/201711/1_633328.mp4</video:content_loc>
      <video:title>«Русский космос. Вперед планеты всей!»</video:title>
      <video:thumbnail_loc>https://img5tv.cdnvideo.ru/webp/shared/files/201806/1_757832.jpg</video:thumbnail_loc>
      <video:description>
        Космос и всё, что с ним связано, волнует и завораживает. Сегодня космос – в тренде. Он с нами и он вокруг нас – интернет,
        телевидение, связь, навигация...
      </video:description>
    </video:video>
    <lastmod>2018-12-03T18:40:36+03:00</lastmod>
  </url>
  <url>
    <loc>https://www.5-tv.ru/video/1025768/</loc>
    <video:video>
      <video:content_loc>https://vod5tv.cdnvideo.ru/shared/files/201708/1_555629.mp4</video:content_loc>
      <video:title>«Мэр»</video:title>
      <video:thumbnail_loc>https://img5tv.cdnvideo.ru/webp/shared/files/201708/2946_552880.jpg</video:thumbnail_loc>
      <video:description>
        10 августа 2017 года Анатолию Собчаку – первому мэру Петербурга – исполнилось бы 80 лет. Он стал целой эпохой и для
        страны, и для своего родного города.
      </video:description>
    </video:video>
    <lastmod>2018-12-03T18:49:33+03:00</lastmod>
  </url>

```

Рисунок 6 – Фрагмента файла «sitemap-video-items.xml»

Грамотное управление веб-искателями очень важно для поддержания веб-сайта или приложения в рабочем состоянии, потому что даже «хорошие» боты могут перегружать сервер, замедляя и даже уничтожая веб-ресурсы. Хорошо составленный файл robots.txt делает веб-сайт пригодным для SEO и держит «хорошую» активность бота под контролем.

Однако, robots.txt не может контролировать вредоносный трафик ботов. Для таких ситуация необходимы специальные решения для управления ботами, которые могут помочь сократить вредоносную активность ботов, не влияя на работу основных ботов, как веб-искатели.

Таким образом, были разобраны и изучены все определения, типы и назначения интернет-ботов. Конкретно то, насколько они важны для ведения современного бизнеса, в плане взаимодействия компаний со своими клиентами. Боты стали жизненно важной частью стратегии привлечения потребителей. В ближайшем будущем боты будут стремительно усложняться, чтобы расширить человеческие возможности и быть более инновационными в управлении стратегическими действиями предприятий.

## 2 Архитектура чат-бота для социальных сетей

Архитектура чат-бота – это самая главная и ответственная часть разработки чат-ботов [15]. В зависимости от удобства использования и контекста бизнес-операций архитектура, связанная с программированием чат-бота, может резко меняться. Несмотря на различные требования клиента в изменении или преобразовании того или иного элемента архитектуры, основной поток коммуникации остается прежним.

На рисунке 7 изображена обобщенная архитектура чата-бота, элементы которой не редко используются в разработке ботов для медиа-платформ.

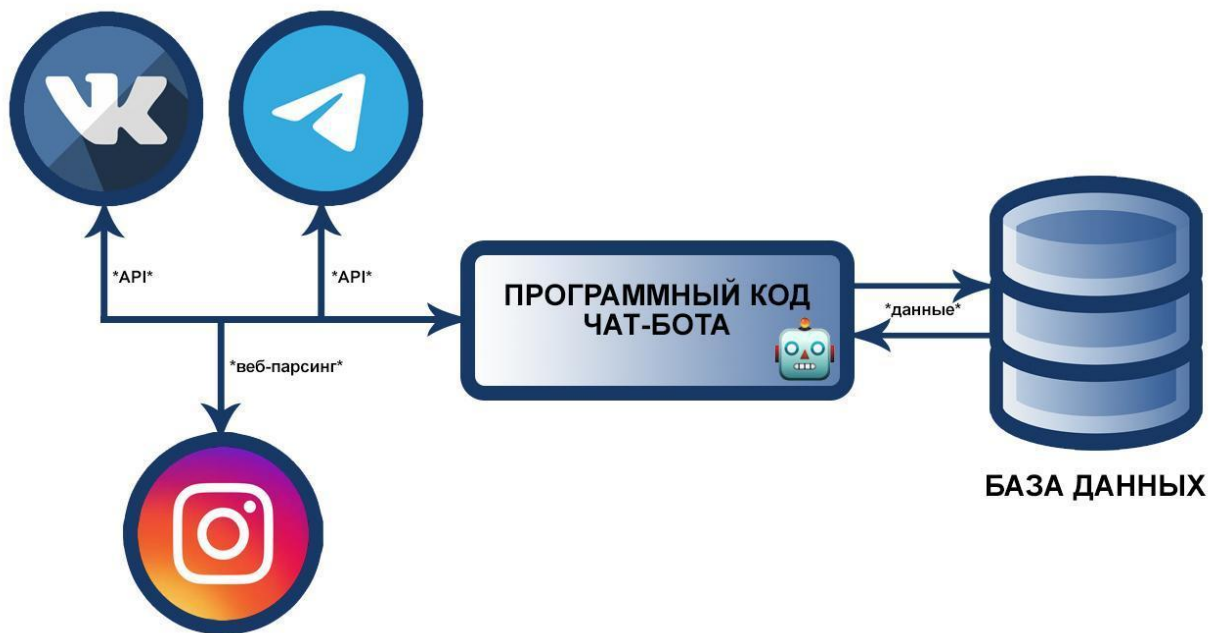


Рисунок 7 – Обобщенная архитектура чат-бота

Данная схема включает в себя:

а) программный код – здесь прописаны все операции и логика чат-бота. Во время своей активности чат-бот использует данные из хранилища, взаимодействует с веб-сервисами, с помощью которых и предоставляет

запрашиваемую информацию пользователям. Для разработки данного кода программисты чаще всего используют Java, Python, Ruby, PHP и так далее;

б) база данных – один из важнейших элементов рассматриваемой архитектуры, которая хранит все данные и необходимые настройки для полноценной работы чат-бота;

в) социальные сети – в рамках данной структуры они выполняют роль интерфейса, с помощью которого медиа-бот может взаимодействовать с интернет-пользователями.

Все вышеперечисленные элементы, которые представлены в этой коммуникации являются неотъемлемыми частями всей структуры чат-бота. А сама архитектура – эталонной.

Несмотря на то, что практически все социальные сети (например, ВКонтакте и Telegram) оснащены доступным для всех бесплатным в использовании набором разных функций, классов и процедур (Application Programming Interface) для разработки своего собственного приложения, с помощью которых можно создавать многофункциональных ботов с различным рядом возможностей, существуют также медиа-платформы, в которых API отсутствует, как например во всемирно известной социальной сети Instagram. Для того чтобы заставить бота работать в этой сети, используя логин и пароль человеческого профиля, некоторые разработчики применяют технологию парсинга (синтаксического анализа) веб-страниц.

Архитектура должна быть устроена так, чтобы для пользователя она была абсолютно простой и легкой в понимании, но в рабочем режиме структура должна быть сложной и глубокой, так как чат-бот направлен в первую очередь на взаимодействие со всеми пользователями одновременно по нескольким каналам, сохраняя каждое необходимое в настройках изменение. Поэтому поначалу можно подумать, что чат-бот выглядит как обычное веб-приложение, хотя способы их реализации весьма различны.

Существует также множество моделей и функций, которые связаны с

определением личности пользователя чат-ботом и потока голосовых сообщений, а также функционал и информацию, к которым пользователи обычно получают доступ в качестве сторонних сервисов через интеграцию. Примером такой интеграции может послужить информационный сервис прогноза погоды, расписание автобусов или поездов, бронирование билетов на концерт или в кинотеатр и так далее.

Стоит отметить, что с каждым годом использование чат-ботов становится все более и более простым, но не стоит забывать, что за этим стоит множество сложных технологий и логических решений. Все эти технологии и решения были также тщательно протестированы.

## **2.1 Программный интерфейс приложения**

API – это аббревиатура слов Application Programming Interface, переводится с английского как интерфейс прикладного программирования или программный интерфейс приложения, который является программным посредником, позволяющим двум программам общаться друг с другом. Каждый раз, когда пользователи используют приложение, например, Вконтакте или Telegram, отправляют мгновенные сообщения или проверяют погоду на своем телефоне, они используют API [16].

Другими словами, API – это набор программного кода, который обеспечивает передачу данных между одним программным продуктом и другим. Он также содержит условия обмена данными.

API состоит из двух компонентов (рисунок 8):

- техническая спецификация, описывающая способы обмена данными, выполненными в форме запроса на обработку, и протокола предоставления данных;
- программный интерфейс, внесенный в спецификацию, которая его

предоставляет.

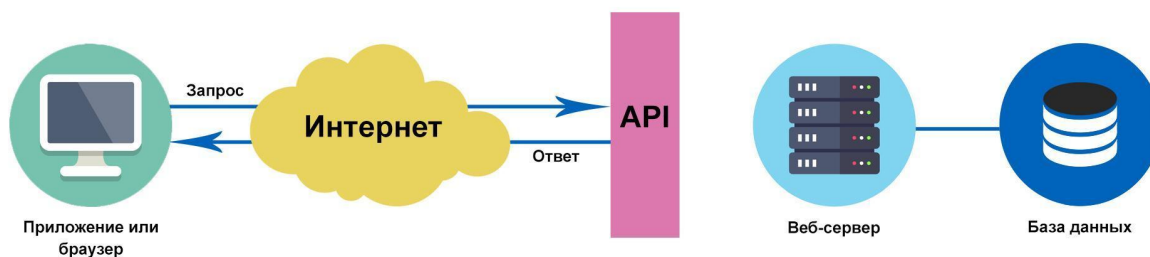


Рисунок 8 – Коммуникация клиента с веб-сервером через API

Программное обеспечение, которому необходим доступ к информации (например, цены на номера в отелях Сочи на определенные даты) или функционалу (например, маршрут из точки А в точку Б на карте, построенный исходя местоположения пользователя) из другого программного обеспечения, вызывает его API, указывая требования к тому, как должны предоставляться данные или функциональные возможности. Другое программное обеспечение возвращает данные или функции, запрошенные предыдущей программой. Все возможные требования и спецификации должны быть заранее указаны в документации по программному интерфейсу приложения.

Документация по API – это руководство для разработчиков, которое содержит всю необходимую информацию о том, как работать с API и использовать предоставляемые им сервисы.

Каждый API содержит и осуществляется с помощью вызова функций – операторов языка, которые запрашивают программное обеспечение для выполнения определенных действий. Все виды функций и способы их вызова описаны в документации по API.

У API множество преимуществ. Как правило, они могут упростить и ускорить разработку программного обеспечения. Программисты могут добавить функционал (например, прогноз погоды, бронирование жилья,



распознавание изображений, обработку платежей) от других приложений с уже существующими решениям или создать новые приложения с использованием услуг сторонних программ. Во всяком случае специалистам не нужно иметь дело с исходным кодом, пытаясь понять, как оно работает. Они просто подключают свое программное обеспечение к стороннему приложению, не тратя много времени в пустую.

API по способу доступа делят на три вида (рисунок 9):

- закрытый (частный);
- партнерский;
- общедоступный.

#### ВИДЫ API ПО СПОСОБУ ДОСТУПА

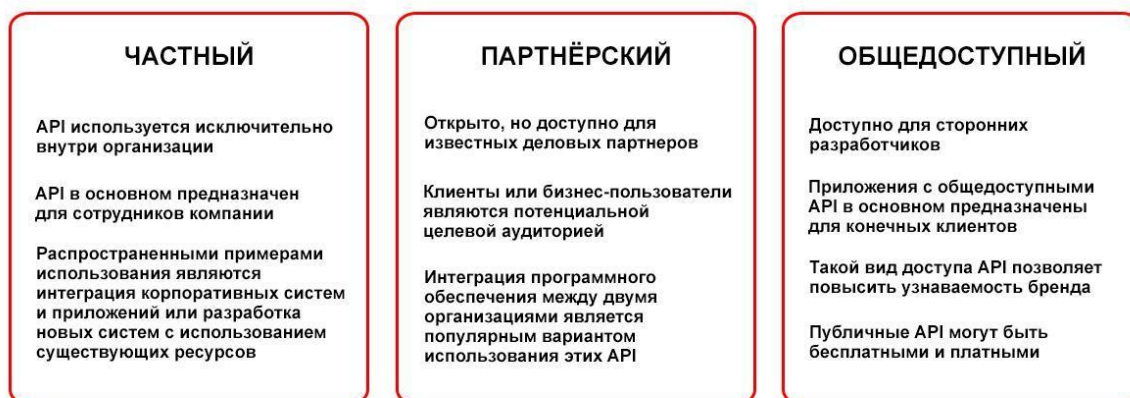


Рисунок 9 – Виды API по способу доступа

Частные API – это интерфейсы прикладного программного обеспечения предназначены для улучшения решений и программ внутри организации. Внутренние разработчики и инженеры могут использовать эти API для интеграции IT-систем или приложений компании, создания новых систем или приложений для клиентов, используя существующие механизмы. Даже если приложения общедоступны, сам интерфейс остается доступным

только для тех, кто работает непосредственно с издателем API. Такой вид доступа позволяет компании полностью контролировать использование API.

Партнерские API – это интерфейсы, которые открыто продвигаются, но делятся ими только с деловыми партнерами, подписавшими соглашение с организацией-издателем. Основным вариантом использования партнерских API является интеграция программного обеспечения между двумя сторонами. Компания, предоставляющая партнерам доступ к данным или функционалу, извлекает выгоду из дополнительных источников дохода. В то же время он может отслеживать, как используются открытые цифровые активы, следить за тем, соблюдают ли сторонние соглашения, использующие их API, и поддерживать корпоративный стиль в своих приложениях.

Общедоступные API, также известные как ориентированные на разработчиков или внешние – это интерфейсы, которые доступны для любых сторонних разработчиков. Публичные API позволяют повысить узнаваемость бренда и получить дополнительный источник дохода при их правильной работе.

Существует два типа общедоступных API:

- открытые (бесплатные);
- коммерческие.

Публичность бесплатных API предполагает, что все функции такого интерфейса являются общедоступными и могут использоваться без ограничительных условий. Например, можно создать приложение, использующее API, без явного одобрения компании-издателя или обязательных лицензионных соглашений и денежных сборов. В определении также говорится, что описание API и любая связанная с ним документация должны быть размещена в открытом доступе и то, что API можно свободно использовать для создания и тестирования своих приложений. Примерами таких API являются API социальных сетей ВКонтакте и Telegram [17].

За коммерческие API пользователи платят абонентскую плату или

используют API с оплатой по мере использования. Среди компаний-издателей распространено предлагать бесплатные пробные версии интерфейсов, чтобы пользователи могли оценить API перед покупкой или подпиской.

API можно классифицировать в соответствии с системами, для которых они предназначены.

API баз данных – это интерфейсы баз данных, которые обеспечивают связь между приложением и системой управления базами данных. Программисты работают с базами данных, создавая запросы для доступа к данным, изменения таблиц и так далее.

API операционных систем – эта группа интерфейсов определяет, как программы используют ресурсы и службы операционных систем. Каждая ОС имеет свой набор API, например, API Windows или API Linux (API пользовательского пространства ядра и внутренний API ядра).

Удалённые API – это интерфейсы, которые определяют стандарты взаимодействия для приложений, работающих на разных машинах или серверах. Другими словами, один программный продукт получает доступ к ресурсам, расположенными за пределами устройства, которое их запрашивает. Поскольку два удалённых приложения чаще всего подключаются через интернет, большинство удалённых API написаны на основе веб-стандартов.

Веб-API – этот класс API является наиболее распространённым. Веб-API обеспечивают передачу машиночитаемых данных и функциональных возможностей между веб-системами, представляющими архитектуру «клиент-сервер». Эти API в основном присылают запросы от веб-приложений и ответы от серверов с использованием протокола передачи гипертекста (HTTP).

Разработчики могут использовать веб-API для расширения функционала своих приложений или сайтов. Например, API ВКонтакте, который позволяет получать информацию из базы данных ВКонтакте и

пользоваться функциями данной сети, и Google Maps API, позволяющий добавлять карту с необходимым местоположением.

Для примера рассмотрим метод для получения данных о пользователе с идентификатором (ID) 140386454 из публичного некоммерческого API ВКонтакте [18]. Запрос такого метода выглядит следующим образом – [https://api.vk.com/method/users.get?user\\_id=140386454&v=5.52](https://api.vk.com/method/users.get?user_id=140386454&v=5.52). Рассмотрим каждый фрагмент URL-запроса подробнее:

- `https://` – протокол соединения;
- `api.vk.com/method` – адрес API-сервиса;
- `users.get` – название метода API ВКонтакте. Методы представляют собой условные команды, которые соответствуют той или иной операции с базой данных. Например, `users.get` – метод для получения информации о пользователе, `messages.send` – метод для отправки сообщения пользователю, `photos.get` – метод для запроса списка фотографий в альбоме;
- `?user_id=140386454&v=5.52` – параметры запроса. После названия метода нужно передать его входные данные (если они есть) – как обычные GET-параметры в HTTP-запросе. Здесь отправляем запрос серверу на получение данных о пользователе с `id=140386454` и формат этих данных должен соответствовать версии API 5.52.

Ответ на данный запрос будет выглядеть следующим образом (листинг 1).

```
{"response":[{"id":140386454,"first_name":"Алла","last_name":"Снопкова"}]}
```

### Листинг 1 – Парсинг даты на Python

В ответ сервер возвращает структуру JSON с запрошенными данными (или сообщение об ошибке, если что-то пошло не так). JSON – это формат

записи данных, основанный на JavaScript, в виде пар «имя свойства»: «значение».

## 2.2 Синтаксический анализ

Синтаксический анализ – это процесс преобразования форматированного текста в структуру данных. Типом структуры данных может быть любое подходящее представление информации, находящееся в исходном тексте [19].

Дерево является распространенным и стандартным выбором для синтаксического анализа XML, HTML, JSON и любого языка программирования. Выходное дерево называется деревом синтаксического анализа или абстрактным синтаксическим деревом. В контексте HTML он называется объектной моделью документа (DOM) [20].

Часть программы, которая выполняет синтаксический анализ, называется парсингом (рисунок 10).

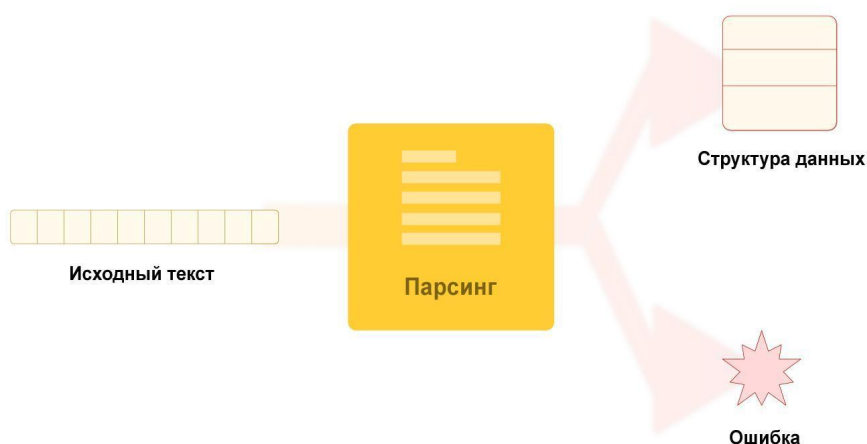


Рисунок 10 – Принцип работы парсера

Парсер анализирует исходный текст в соответствии с заданным форматом (формат кодируется внутри парсера). Если исходный текст не совпадает с форматом, то программа возвращает ошибку. А если есть совпадения, то она возвращает «структуру данных».

Рассмотрим пример синтаксического анализа даты из строки (источника) в формате DD-MM-YYYY (рисунок 11).

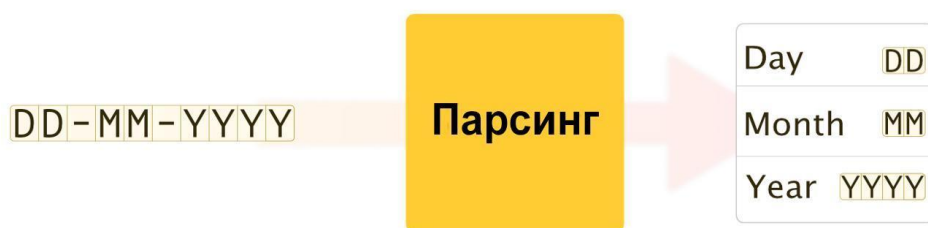


Рисунок 11 – Парсинг даты в формате DD-MM-YYYY

Для анализа даты используем регулярные выражения [21]. Данные выражения могут сопоставляться со строкой, что очень помогает в извлечении части исходного текста, если заданный формат анализа ему соответствует.

Программный код синтаксического анализа и извлечения даты с источника представлен на листинге 2.

```
import re
from dateutil import parser

date_string = u'''давайте распарсим даты из этой строки,
начнем с 2021-01-01, закончим 01.02.2021, и напоследок
посмотрим, что же было 10.06.21'''

date_line =
re.findall('\d{4}-\d{2}-\d{2}|\d{2}.\d{2}.\d{4}|\d{2}.\d{2}.\d{2}
```

```

}',date_string)
    if date_line:
        for dt in date_line:
            print parser.parse(dt)
    else:
        print("Error")

#Вывод в консоли:
##>>>
##2021-01-01 00:00:00
##2021-02-01 00:00:00
##2021-06-10 00:00:00

```

## Листинг 2 – Парсинг даты на Python

В данном коде все даты из строки `date_string` в различных форматах были сопоставлены с помощью регулярного выражения – `\d{4}-\d{2}-\d{2}|\d{2}.\d{2}.\d{4}|\d{2}.\d{2}.\d{2}`, где «`\d`» соответствует любой цифре от 0 до 9; «`{n}`» означает, число какой значности будет принимать заданный шаблон; каждая группа комбинаций объединена в общий формат шаблона поиска даты, которая отделена символом «`|`», означающий логический оператор «ИЛИ». Если совпадение прошло успешно, то все даты будут выведены в консоле интерпретатора, иначе программа выдаст сообщение об ошибке.

Данный пример иллюстрирует синтаксический анализ на основе регулярных выражений, который ограничен форматом, определенный ранее регулярным выражением. Это простой пример, который демонстрирует принцип работы технологии парсинга.

При помощи такой технологии написаны абсолютно все боты, которые бытуют в социальной сети Instagram. В ней уже больше не существует такого обширного и многофункционального публичного API, как это было ранее.

В 2020 году Facebook изменил свою политику конфиденциальности из-за растущей обеспокоенности пользователей и частых обвинений в нарушении обработки персональных данных в сторону многих крупных компаний, включая Facebook. В связи с чем руководством сети Instagram было принято решение пересмотреть и урезать их официальный открытый API. Теперь с помощью API Instagram можно получить доступ лишь только к собственным постам своего аккаунта и даже не к публичным комментариям и постам в сети, которые могут быть доступны незарегистрированному пользователю. Это весьма затруднило разработчикам программно сканировать и обрабатывать данные Instagram, поэтому им приходится прибегать к использованию в своих проектах технологию синтаксического анализа.

На языке программирования Python существует множество библиотек, которые помогают разработчикам в парсинге веб-ресурсов. На данный момент самой популярной из всех модулей является парсинг-библиотека BeautifulSoup.

BeautifulSoup – это библиотека Python для анализа HTML и XML-документов [22]. Она часто используется для сканирования ресурсов Всемирной паутины. BeautifulSoup преобразует сложный HTML-документ в сложное дерево объектов Python, таких как тег, навигационная строка или комментариев.

Именно BeautifulSoup многие программисты рекомендуют использовать в разработке своих ботов для таких социальных сетей, как Instagram и многих других, которые не имеют собственного открытого API.

### **2.3 Инструменты и средства для разработки ботов**

В основном чат-бот – это программный код, с которым люди могут общаться. Если использовать его правильно, то жизнь станет намного проще



– вместо того, чтобы искать что-то в интернете, можно просто запросить у чат-бота всю необходимую информацию.

Чаще всего в разработке чат-ботов используются следующие языки программирования – Python, PHP, Java и Ruby.

Python используется в основном из-за его простоты в написании кода. Язык объектно-ориентированный и имеет простой синтаксис [23]. Он также один из наиболее широко используемых языков программирования в области искусственного интеллекта, благодаря своей простоте. Практически все чат-боты легко писать именно на этом языке программирования.

На PHP можно легко создать чат-бота, потому что это язык с открытым исходным кодом, и он очень прост в использовании [24]. Он намного быстрее, чем другие сценарные языки, и для работы с API доступно множество библиотек. Если разработка чат-бота ведется на PHP, то его будет легко запускать на любом сервере.

Java может предоставить программисту все функции высокого уровня, необходимые для разработки проектов с искусственным интеллектом. В создании чат-бота с искусственным интеллектом, Java, вероятно, лучший язык, который можно использовать разработчику, так как он предлагает простой способ кодирования алгоритмов. Существует также технология «виртуальная машина Java», с помощью которой разработанное приложение будет работать на любой платформе [25].

Ruby имеет очень простой синтаксис, который позволяет новичкам легко и быстро создавать чат-ботов. Он во многом похож на Perl и Python. Ruby – это динамический объектно-ориентированный язык программирования [26]. В этом языке все конструкции являются выражениями, и они выполняются императивно. Однако создание чат-бота с помощью Ruby может быть дорогостоящим. Изначально разработка на нём будет бесплатной, но в какой-то момент необходимо будет купить лицензию. Он не очень гибок для разработчиков, поскольку является зависимым.

## 2.4 Расстояние Левенштейна

Расстояние Левенштейна (его ещё называют редакционным расстоянием или дистанцией редактирования) между двумя последовательностями символов в теории информации и компьютерной лингвистике – это самое малое число операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для преобразования одной строки в другую.

Например, чтобы преобразовать слово «authentication» в слово «verification» необходимо произвести одну вставку, три удаления и три замены, отсюда следует, что расстояние Левенштейна составляет 7:

- 0 – authentication;
- 1 – vuthentication (замена «a» на «v»);
- 2 – vtentication (удаление «u»);
- 3 – vhentication (удаление «t»);
- 4 – ventication (удаление «h»);
- 5 – vertication (замена «n» на «r»);
- 6 – veritication (вставка «i»);
- 7 – verification (замена «t» на «f»).

Именно 7 операций необходимо совершить, чтобы получить из слова «authentication» слово «verification» кратчайшим образом.

Чаще всего расстояние Левенштейна применяется в определении схожести строк при проверке наличие орфографических ошибок при вводе текста в системах управления базами данных, поисковых системах и текстовых процессорах, при автоматическом распознавании отсканированного текста или голосовых сообщений в социальных сетях, их исправлении, а также при поиске дубликатов слов. Широкое применение эта метрика получила в биоинформатике для сравнения аминокислот.

Впервые эту задачу сформулировал советский и российский математик,

доктор физико-математических наук Владимир Иосифович Левенштейн в 1965 году при изучении последовательностей 0-1 [27]. Уже в дальнейшем Дэн Гасфилд более подробно изучал данную проблему и более общая задача для произвольного алфавита было соотнесено с именем Владимира Левенштейна [28].

Однако, у данного метода определения расстояния между двумя словами есть несколько проблем:

- при перестановке последовательностей символов или частей последовательностей символов могут получиться достаточно большие расстояния по сравнению с исходным результатом;
- расстояния между очень похожими длинными словами значительно больше по сравнению с расстояниями между совершенно разными короткими словами.

Рассмотрим формулу расчета редакционного расстояния. Пусть  $S_1$  – одна строка с длиной  $M$  символов,  $S_2$  – вторая строка с длиной  $N$  символов над некоторым произвольным алфавитом, следовательно формула расстояния Левенштейна  $d(S_1, S_2)$  будет иметь вид (1).

$$d(S_1, S_2) = D(M, N), \text{ где}$$

$$D(i, j) = \begin{cases} 0, & i = 0, j = 0 \\ i, & j = 0, i > 0 \\ j, & i = 0, j > 0 \\ \min\{ & \\ D(i, j-1) + 1, & \\ D(i-1, j) + 1, & j > 0, i > 0 \\ D(i-1, j-1) + m(S_1[i], S_2[j]) & \end{cases} \quad (1)$$

В формуле 1,  $m(a, b) = 0$ , если  $a = b$  и единице в противном случае;  $\min\{a, b, c\}$  возвращает минимальный из аргументов. Операция по  $i$  символизирует удаление из первой строки, по  $j$  – вставку в первую строку, а

операция по обоим индексам означает замену символа или отсутствие всяческих изменений.

Отсюда вытекают следующие утверждения:

- $d(S_1, S_2) \geq ||S_1| - |S_2||$ ;
- $d(S_1, S_2) \leq \max(|S_1|, |S_2|)$ ;
- $d(S_1, S_2) = 0 \Leftrightarrow S_1 = S_2$ .

Рассмотрим работу данного алгоритма на примере перевода строки «authentication» в строку «verification» (таблица 1). Расстояние Левенштейна в этом случае равно 7.

Таблица 1 – Матрица редактирования «authentication» в «verification»

		a	u	t	h	e	n	t	i	c	a	t	i	o	n
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
v	1	1	2	3	4	5	6	7	8	9	10	11	12	13	14
e	2	2	2	3	4	4	5	6	7	8	9	10	11	12	13
r	3	3	3	3	4	5	5	6	7	8	9	10	11	12	13
i	4	4	4	4	4	5	6	6	6	7	8	9	10	11	12
f	5	5	5	5	5	5	6	7	7	7	8	9	10	11	12
i	6	6	6	6	6	6	6	7	7	8	8	9	9	10	11
c	7	7	7	7	7	7	7	7	8	7	8	9	10	10	11
a	8	7	8	8	8	8	8	8	8	8	7	8	9	10	11
t	9	8	8	8	9	9	9	8	9	9	8	7	8	9	10
i	10	9	9	9	9	10	10	9	8	9	9	8	7	8	9
o	11	10	10	10	10	10	11	10	10	9	10	9	8	7	8
n	12	11	11	11	11	11	10	11	11	10	11	10	9	8	7

В листинге 3 приведена программная реализация данного алгоритма на языке Python с использованием рекурсии, а также результат его работы.

```
def levenshteinDistance(S1, S2):
    # в функцию передаются две последовательности символов
    def D(i, j):
        if i == 0 or j == 0:
            # если одна из двух последовательностей символов
            # является пустой, то расстояние до второй последовательности
            # будет равна её длине, то есть N вставок
            return max(i, j)
        elif S1[i - 1] == S2[j - 1]:
            return D(i - 1, j - 1)
        else:
            return 1 + min(
                D(i, j - 1), # удаление символа
                D(i - 1, j), # вставка символа
                D(i - 1, j - 1) # замена символа
            )
    return D(len(S1), len(S2))

if __name__ == "__main__":
    print("LevenshteinDistance(\"authentication\",
    \"verification\") = " +
    str(levenshteinDistance("authentication", "verification")))

#OUTPUT: LevenshteinDistance("authentication", "verification") =
7
```

### Листинг 3 – Реализация алгоритма нахождения расстояния Левенштейна

Далее с помощью вспомогательной функции протестируем производительность данного алгоритма с выводом результата (листинг 4).

```

from timeit import timeit

def levenshteinDistance(S1, S2):
    {...}
    def testLevenshteinDistance(f: callable, S1, S2, number=1):
        # данная функция замеряет время выполнения функции
        levenshteinDistance(S1, S2) при одном вызове (number = 1) по
        умолчанию

        tmt = timeit("f(a, b)", globals={
            'f': f, 'a': S1, 'b': S2
        }, number=number)

        D = f(S1, S2)
        print(f'S1 = {S1!r}\nS2 = {S2!r}\n{f.__name__} =
        {D}\nWorktime = {tmt:.4f}')

if __name__ == "__main__":
    testLevenshteinDistance(levenshteinDistance,
    "authentication", "verification")

# OUTPUT: S1 = 'authentication'
S2 = 'verification'
levenshteinDistance = 7
Worktime = 0.0028

```

#### Листинг 4 – Тестирование производительности алгоритма нахождения расстояния Левенштейна

Результат работы программы показывает, что исходный вариант алгоритма для расчета редакционного расстояния выполняется достаточно медленно. В данном примере может быть это не особо ощутимо, но при

сравнении довольно длинных последовательностей символов, состоящих из целых предложений, которые будет вводить пользователь, времени уйдёт достаточно много. Это объясняется тем, что программа много раз вычисляет функцию для одних и тех же входных параметров. Для того, чтобы радикально повысить производительность работы программного кода, можно закешировать повторяющиеся вызовы с одинаковыми параметрами с помощью декоратора `lru_cache` из библиотеки `functools`, который позволяет сэкономить время при дорогих вычислениях, если метод периодически вызывается с одними и теми же параметрами.

Внедрим декоратор `lru_cache(maxsize=len(S1) * len(S2))` и проведём замеры времени выполнения работы нашей функции (листинг 5).

```
from functools import lru_cache
from timeit import timeit

def levenshteinDistance(S1, S2):
    # декоратор, сохраняющий результаты последних вызовов
    @lru_cache(maxsize=len(S1) * len(S2))
    def D(i, j):
        if i == 0 or j == 0:
            # если одна из двух последовательностей символов
            # является пустой, то расстояние до второй последовательности
            # будет равна её длине, то есть N вставок
            return max(i, j)
        elif S1[i-1] == S2[j-1]:
            # если два последних символа равны, то убираем их, не
            # меняя расстояния
            return D(i - 1, j - 1)
        else:
            # иначе берем самое маленькое число из трех операций,
            # прибавляя единицу
            return 1 + min(
```

```

        D(i, j - 1), # удаление символа
        D(i - 1, j), # вставка символа
        D(i - 1, j - 1) # замена символа
    )
    return D(len(S1), len(S2))

def testLevenshteinDistance(f: callable, S1, S2, number=1):
    {...}

if __name__ == "__main__":
    testLevenshteinDistance(levenshteinDistance,
        "authentication", "verification")

# OUTPUT: S1 = 'authentication'
S2 = 'verification'
levenshteinDistance = 7
Worktime = 0.0001

```

### Листинг 5 – Тестирование производительности оптимального алгоритма нахождения расстояния Левенштейна

В результате проведённой оптимизации производительность алгоритма улучшилась в 28 раз. Сложность данного решения растёт как произведение последовательностей символов –  $O(n*m)$ . Данная реализация алгоритм также может быть доработана. Например, при использовании красно-черных или АВЛ деревьев можно добиться ещё большей скорости вычислений. Ещё необходимо учесть то, что на известном словаре можно заранее вычислить расстояния между словами.

Такой алгоритм для расчёта расстояния Левенштейна уже был реализован на языке программирования C и используется в пакете python-Levenshtein для Python-библиотеки FuzzyWuzzy [29]. FuzzyWuzzy –



модуль для языка программирования Python, который содержит функции для нечёткого поиска строк, удаления дубликатов и корректировки возможных ошибок. Модуль FuzzyWuzzy может работать без набора python-Levenshtein, но гораздо медленнее (в 3-10 раз), поэтому для ускорения работы этого модуля рекомендуется установить данный пакет. Он «лёгкий» – весит порядка 50 Кб. Работает в версии Python 2.7 и выше.

На сегодняшний день библиотека FuzzyWuzzy с пакетом python-Levenshtein является самым оптимальным, легковесным и эффективным инструментом для нечеткого сравнения строк на языке программирования Python. На базе данного модуля написано множество программ, которые поддерживают редактирование внесённых данных пользователем (например, FuzzySearch).

Существует несколько аналогов FuzzyWuzzy, которые не уступают по своей эффективности вычислений и быстродействию, например, модули FuzzyLogic на Matlab и в языке программирования Ruby [30]. Но реализация кода при помощи одной из этих аналогов будет весьма затратной, как по финансам, так и по использованию системных ресурсов сервера.

## **2.5 SQL и NoSQL базы данных**

SQL и NoSQL представляют собой два наиболее распространенных типа баз данных. SQL – это язык структурированных запросов, который используется в большинстве современных систем управления реляционными базами данных (СУБД). NoSQL означает либо «no SQL» (он не использует SQL для запросов), либо «not only SQL» (он использует как методы запросов SQL, так и методы, отличные от SQL). NoSQL обычно используется в нереляционной базе данных (то есть в той, где не поддерживаются первичные и вторичные ключи и соединения между таблицами). Различия между базами данных SQL и NoSQL включают в себя то, как они построены, структуру и

типы данных, которые они содержат, а также то, как данные хранятся и запрашиваются.

Реляционные базы данных (SQL) используют жесткую структуру таблиц со столбцами и строками. В каждой строке есть одна запись, и каждый столбец содержит определённую информацию. Структурированные данные требуют нормализации, что снижает избыточность данных и повышает их надежность. SQL – это строго контролируемый стандарт, поддерживаемый Американским национальным институтом стандартов (ANSI) и Международной организацией по стандартизации (ISO) [31].

Нереляционные базы данных часто реализуются в виде систем NoSQL. Существуют множество типов баз данных NoSQL – от хранилищ значений ключей до хранилищ документов, баз данных графиков, баз данных временных рядов и хранилищ с широкими столбцами. Некоторые системы NoSQL также поддерживают «многомодельные» схемы – это значит, что они могут поддерживать внутри себя более одной схемы данных [32].

В отличие от стандартов ANSI/ISO для языка SQL, не существует отраслевого стандарта для реализации систем NoSQL. Точный способ поддержки разных схем NoSQL зависит от различных независимых разработчиков программного обеспечения. Реализации баз данных NoSQL могут быть очень разными и несовместимыми. Например, даже если две системы являются базами данных с ключевыми значениями, их API, модели данных и методы хранения могут сильно расходиться и быть взаимно несовместимы.

Системы NoSQL не полагаются на объединенные таблицы и не могут их поддерживать.

Сравнение NoSQL и SQL основано на таких критериях, как непрерывность, доступность и скорость. Как правило, компании сами определяют, какой тип системы баз данных следует использовать, исходя от своих нужд и потребностей (рисунок 12).

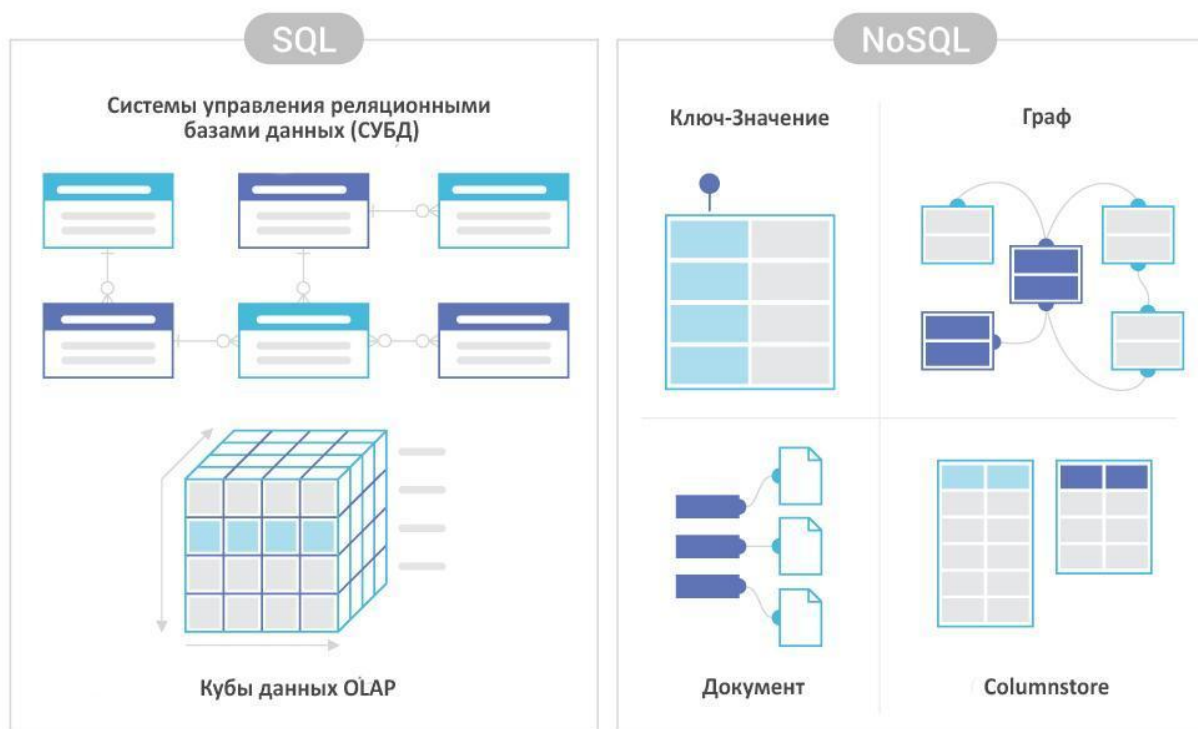


Рисунок 12 – Ключевые различия между SQL и NoSQL

Реляционная база данных (SQL) идеально подходит для обработки моделей данных, которые хорошо проработаны и неизменны на протяжении большого количества времени, и требуют соблюдения строгих международных стандартов, а также для компаний, которые ценят хранение данных с высокой скоростью обработки и передачи.

Нереляционная база данных (NoSQL) идеально подходит для компаний, которые постоянно сталкиваются с меняющимися требованиями к хранению данных, с теми, кто может адаптироваться к быстро развивающимся стандартам и API, а также с теми, кому необходимо иметь дело с несколькими типами данных и большими объемами потока информации.

Список самых популярных реляционных баз данных (SQL) включает в себя:

- Oracle Database;

- MySQL;
- SQLite;
- PostgreSQL;
- Microsoft SQL Server.

В список популярных нереляционных баз данных (NoSQL) включают:

- Apache Cassandra;
- Apache HBase;
- Scylla;
- MongoDB;
- Redis.

Существуют следующие типы баз данных NoSQL:

- «ключ-значение» – хранит данные с простыми индексированными ключами и значениями. Например, такой тип есть у баз данных Oracle NoSQL, Redis, Aerospike, Oracle Berkeley DB, Voldemort, Amazon DynamoDB и Infinity DB;

- хранилище широких столбцов (columnstore) – использует таблицы, строки и столбцы, но формат и имена столбцов могут отличаться в разных строках одной и той же таблицы. Например, Apache Cassandra, Scylla, Datastax Enterprise, Apache HBase, Apache Kudu, Apache Parquet и MonetDB;

- «документ» – более сложная и структурированная версия модели ключ-значение, которая дает каждому документу свой собственный ключ поиска. Примеры включают Orient DB, MarkLogic, MongoDB, IBM Cloudant, Couchbase и Apache CouchDB;

- «граф» – представляет взаимосвязанные данные в виде логического графа. Примеры – Neo4j, JanusGraph, FlockDB и GraphDB.

## 2.6 Тестирование программного обеспечения и его классификация

Тестирование программного обеспечения – это метод проверки соответствия фактического программного продукта ожидаемым требованиям и обеспечения отсутствия дефектов программного продукта [33]. Целью тестирования программного обеспечения является выявление ошибок, пробелов и отсутствующих требований, которые были прописаны в техническом задании, но впоследствии не были реализованы.

Существуют следующие уровни тестирования:

а) модульное – проверяет, выполняют ли программные модули функциональные возможности или нет. Данное тестирование осуществляется разработчиками до того, как программа будет передана команде QA;

б) интеграционное – проверяет, успешно ли отдельные модули функционируют в сочетании друг с другом. Есть три способа интеграционного тестирования:

- 1) «снизу вверх»;
- 2) «сверху вниз»;
- 3) «большой взрыв»;

в) системное – тестируется приложение в целом для того, чтобы оценить, соответствует ли система всем требованиям, указанным в начале разработки. Имеются два вида системного тестирования:

- 1) на базе требований;
- 2) на базе случаев использования;

г) приёмочное – проверяет, соответствует ли приложение предполагаемым спецификациям и удовлетворяет требованиям клиента. Данное тестирование проводится командой QA, и включает в себя:

- 1) контактное и правовое приёмочное;
- 2) эксплуатационное;

### 3) альфа- и бета-тестирование.

Список методов тестирования включает в себя:

а) статическое (рецензирование) – метод тестирования, которое выполняется без запуска программного кода и проводится с целью выявить недостатки уже в фазах проектирования программы и спецификации;

б) динамическое – метод тестирования, который проверяет функциональность приложения, когда код выполняется [34]. Есть три способа динамического тестирования:

- 1) «белый ящик»;
- 2) «серый ящик»;
- 3) «чёрный ящик».

Тестирование также подразделяют на следующие виды:

а) по признаку позитивности:

- 1) позитивное;
- 2) негативное;

б) по степени автоматизации:

- 1) ручное;
- 2) полуавтоматизированное;
- 3) автоматизированное.

Помимо уровней, методов и видов, есть несколько типов тестирования:

а) функциональное – проверяет реализуемость функциональных требований, то есть способность приложения в определенных условиях решать задачи, необходимые пользователям;

б) нефункциональное – включает в себя тестирование качественных характеристик компонента или системы, которые могут быть измерены различными величинами:

- 1) юзабилити;
- 2) конфигурация (клиент-сервер);
- 3) инсталляция;

- 4) локализация;
- 5) производительность;

в) структурное – основывается на детальном изучении логики программы;

г) тестирование изменений – производится после того, как был исправлен дефект продукта или было внесено изменение в структуру приложения. Данное тестирование разделяется на:

- 1) «дымовое»;
- 2) тестирование сборки;
- 3) повторное;
- 4) регрессионное;
- 5) санитарное.

К каждому вышеперечисленному типу тестирования существуют следующие подходы:

- сценарное;
- исследовательское;
- интуитивное.

Отсюда следует, что существуют множество различных подходов, методов и средств, которые можно использовать для создания чат-бота. Выбор того или иного инструмента разработки уже зависит от требований и видения самой компании.

### 3 Разработка чат-бота для предоставления ответов на запросы

В целях урегулирования потока однотипных сообщений в мессенджере сообщества социальной сети ВКонтакте был разработан чат-бот для предоставления ответов на запросы посетителей и участников группы.

Архитектура разрабатываемого чат-бота для предоставления ответов на запросы представлена на рисунке 13.



Рисунок 13 – Архитектура разрабатываемого чат-бота для предоставления ответов на запросы

Она состоит из трёх компонентов:

- интерфейс чат-бота, в роли которой выступает социальная сеть ВКонтакте. Для обработки сообщений и предоставления ответа на запрос пользователю ВКонтакте чат-бот будет соединяться к данному интерфейсу с помощью уникального ключа – api-token;
- программный код чат-бота на языке программирования Python – логика чат-бота с нечётким поиском;
- база данных, которая хранит все настройки и данные, необходимые для работы чат-бота.

Разберем каждый элемент архитектуры подробнее.



Интерфейс чат-бота, в качестве которого выступает вкладка «Сообщения сообщества» в группах социальной сети ВКонтакте. Чтобы авторизоваться, как группа, в сети Вконтакте необходимо использовать модуль языка программирования Python `vk_api` [35]. Затем в «Управлении» группы важно включить «Сообщения сообщества» и создать ключ доступа к API во вкладке «Работа с API» (рисунок 14).

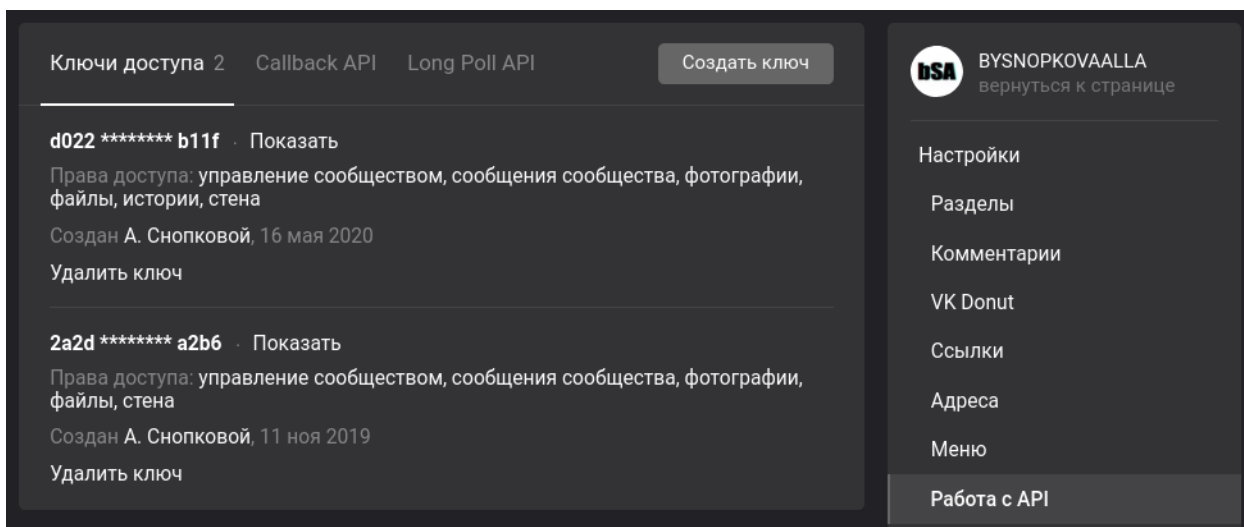


Рисунок 14 – Работа с API в сообществе Вконтакте

В программном коде процедура авторизации занимает всего несколько строк (листинг 6).

```
import vk_api
vk = vk_api.VkApi(token="1q2w...9o0p")
vk._auth_token()
```

Листинг 6 – Авторизация бота в сообществе ВКонтакте

Для обработки сообщений необходимо использовать метод `vk.method`.

В `vk.method` можно вызывать любой метод из VK-API и передавать параметры в виде словаря. На листинге 7 представлен пример метода для отправки сообщения пользователю.

```
vk.method("messages.send", {"peer_id": id, "random_id": 0,  
"message": "Привет!"})
```

### Листинг 7 – Авторизация бота в сообществе ВКонтакте

Здесь `vk.method` вызывает метод `messages.send` и в качестве параметров передаем `id` пользователя, рандомный `id` равный 0 (нужен для того, чтобы нечаянно не отправить ответ другому пользователю в случае сбоя работы программы) и текст сообщения.

Помимо `messages.send` в разрабатываемом чат-боте используются следующие методы:

а) `messages.getConversations` – возвращает список бесед сообщества согласно нижеуказанным параметрам:

1) `offset = 0` – принимаем список без смещения итератора сбора;  
2) `count = 20` – запрашиваемый список не должен превышать 20 записей;

3) `filter = unanswered` – запрашиваемый список должен состоять из неотвеченных бесед в мессенджере группы;

б) `users.get` – возвращает расширенную информацию о пользователе с помощью параметра `user_ids = id`;

в) `messages.markAsAnsweredConversation` – помечает беседу как отвеченную или снимает отметку по нижеуказанным параметрам:

- 1) `peer_id = id` – идентификатор беседы;
- 2) `answered = 1` – беседа отмечается как отвеченная;
- 3) `group_id = id_group` – идентификатор сообщества, который

можно найти в настройках сообществе в разделе «Основная информация».

Необходимо также отметить, что в случае если чат-бот не сможет пройти успешно авторизацию, то об этом важно сообщить пользователю. Это может произойти в том случае, если пользователь в настройках бота ввёл неверный ключ доступа API. Для таких вариантов развития событий в разработке использован обработчик исключения `vk_api.exceptions.ApiError`.

Далее разберем саму логику программы. Анализ текстового сообщения пользователя ВКонтакте производится чат-ботом с помощью технологии нечёткого поиска. В качестве метрики нечёткого поиска была использована функция нахождения расстояния Левенштейна. Для этого в коде была задействована библиотека `fuzzywuzzy` с пакетом `python-Levenshtein` [36].

На листинге 8 показан фрагмент кода, который отвечает за обработку сообщения пользователя и выдачу ответа по найденному в результате обработки ключевому слову с помощью метода `process.extractOne`(«текст сообщения», [«список», «ключевых», «слов»]), который высчитывает совместимость двух строк и возвращает кортеж из двух элементов – наиболее соответствующее слово из полученного на вход массива строк с текстом сообщения и число совместимости в процентом соотношении.

```
import vk_api
from fuzzywuzzy import process

{...}

res_find_meta_kw = process.extractOne(msg, meta_kw_bd)
meta_kw = " - "
if res_find_meta_kw[1] > 45:
    meta_kw = res_find_meta_kw[0]
find_ans = []
for s in answers_collection.find({"meta_keyword":
```

```

meta_kw}):
    find_ans += s["keyword"]
    res_find_answers = process.extractOne(msg, find_ans)

    if res_find_answers[1] > 45:

        for answer in answers_collection.find({"meta_keyword":
ctg}):
            if res_find_answers[0] in answer["keyword"]:

                vk.method("messages.send", {"peer_id": id,
"random_id": 0, "message": answer["text"].replace("{user name}",
profiles[0]['first_name'])})

            else:
                vk.method("messages.markAsAnsweredConver-
sation", {"peer_id": id, "answered": 1, "group_id": id_group})
{...}

```

### Листинг 8 – Фрагмент кода чат-бота с нечетким поиском

Логика данного кода простая – сначала происходит поиск ключевого мета-слова в тексте запроса, следом, уже по найденному мета-слову, программа собирает все существующие ключевые слова из базы данных и сопоставляет их с сообщением пользователя. Если расстояние Левенштейна одного из ключевых слов с текстом сообщения превосходит 45 %, то чат-бот выдает ответ, иначе данная беседа помечается для бота, как «отвеченная», а участник или посетитель сообщества будет ждать своего ответа уже от администратора группы.

Число 45 указан для проверки неслучайно. В ходе изучения расстояния Левенштейна и применения данного алгоритма на практике, было выявлено,

что 45 – это минимальный процент верного соответствия ключевого слова с исходным текстом. Если метод `process.extractOne()` выдает число, которое меньше 45, то данное ключевое слово точно не присутствует в тексте, независимо от того, насколько полученное сообщение является длинным.

Также в ходе анализа данной метрики нечёткого поиска было замечено, что если ключевые фразы состоят из одного и того же слова, например, «обложка для ВКонтакте» и «обложка для Ютуб», то она показывает одинаковый процент совместимости этих фраз с исходным сообщением, в связи с этим было принято решение о внедрении ключевого мета-слова. В данном примере, в качестве мета-слова выступает «обложка». Такое решение также помогает чат-боту быстрее найти правильный ответ из огромного списка ответов в базе данных и выдать его пользователю сети, так как ключевое мета-слово группирует некоторые ответы в некоторую категорию.

И наконец, коснемся последнего компонента архитектуры – база данных системы. Данная база хранит все ответы, которыми будет отвечать чат-бот, и ключевые слова, с помощью которых чат-бот будет определять какой ответ необходимо отправить пользователю, а также все настройки для успешной авторизации чат-бота в сообществе Вконтакте.

Подробнее о базе данных описываемой системы рассмотрим в следующем параграфе.

### **3.1 Разработка веб-приложения для управления чат-ботом**

Для того чтобы пользователь, который не является специалистом в области информационных технологий, смог сам настроить чат-бот для предоставления ответов на запросы по своему усмотрению, было принято решение о создании многопользовательского веб-приложения для управления этим чат-ботом (рисунок 15).



## База данных ответов

Привет, bysnorkovaalla

В этой таблице представлены все ответы, которыми будет отвечать ваш чат-бот:

Название вопроса	Текст ответа	Ключевое мета-слово	Ключевые слова		
Цена обложки для ВКонтакте?	{user name}, динамическая обложка стоит 500 рублей (за 3 обложки для карусели) +50 рублей (за *1 обложку). Отметим, что Вам надо будет оплатить услуги vk.com/dycover (цена 100 руб./мес.), чтобы эта обложка работала. Цена простой обложки варьируется в диапазоне 100-300 рублей. Зависит от сложности. Предоплата 50%.	обложка	ВКонтакте динамическая		
Цена удаления объектов?	{user name}, цена удаления водяных знаков составляет 100-300 рублей (цена зависит от сложности). Предоплата 50%.	-	водяные знаки удалить очистка убрать		
Цена арта?	{user name}, арт стоит 50-200 рублей (цена зависит от сложности). Предоплата 50%. P.S. Цена на удаление объектов ДРУГАЯ!	-	арт фотошоп обра		
Цена обложки для Ютуб?	{user name}, здравствуйте! Обложка для Ютуб стоит 300 рублей независимо от сложности. Предоплата 50%.	обложка	Ютуб		
Ваше портфолио!	{user name}, доброе время суток! Моё портфолио представлено на стене группы vk.com/bysnorkovaalla	-	портфолио творчество примеры работ		
Цена видео?	{user name}, здравствуйте! Цена на видеомонтаж начинается с 500 рублей! Подробнее о цене можно договориться с нашим мастером! Предоплата 70%!	-	монтаж видео смонтировать		
Приветствие!	{user name}, здравствуйте! Здесь Вы можете узнать сколько стоит наши услуги и/или заказать работу! Что конкретно вы хотели узнать?	-	привет здравствуйте добрый		
Хочу заказать!	{user name}, чтобы оформить заказ на фотообработку напишите в ЛС нашему админу - <a href="https://vk.com/beautifuldirt">https://vk.com/beautifuldirt</a> , а чтобы заказать видеомонтаж - <a href="https://vk.com/okumurrrra">https://vk.com/okumurrrra</a>	-	заказ купить		

Запустить чат-бот

Остановить чат-бот

Добавить ответ

Рисунок 15 – Главная страница веб-приложения

Формат «веб-приложение» программному продукту был выбран по следующим причинам:

- программисту достаточно установить разработанное веб-приложение на сервер, работающей под любой современной операционной системой, чтобы пользователи могли пользоваться им через интернет на любом компьютере или мобильном устройстве, работающем под управлением Mac OS, Windows, Linux или какой-либо другой системой;
- обновлять веб-приложение будет дешевле и намного проще;

– если есть необходимость обращаться к одним и тем же данным из разных мест, то лучше всего организовать их хранение в одном конкретном месте, вместо того чтобы разбрасывать их по разным базам данных.

Для реализации веб-приложения для управления чат-ботом были использованы следующие языки программирования и технологии (рисунок 16):

- клиентская часть – HTML, CSS и JavaScript (Bootstrap);
- серверная часть – Python 3 (Flask);
- база данных – MongoDB.

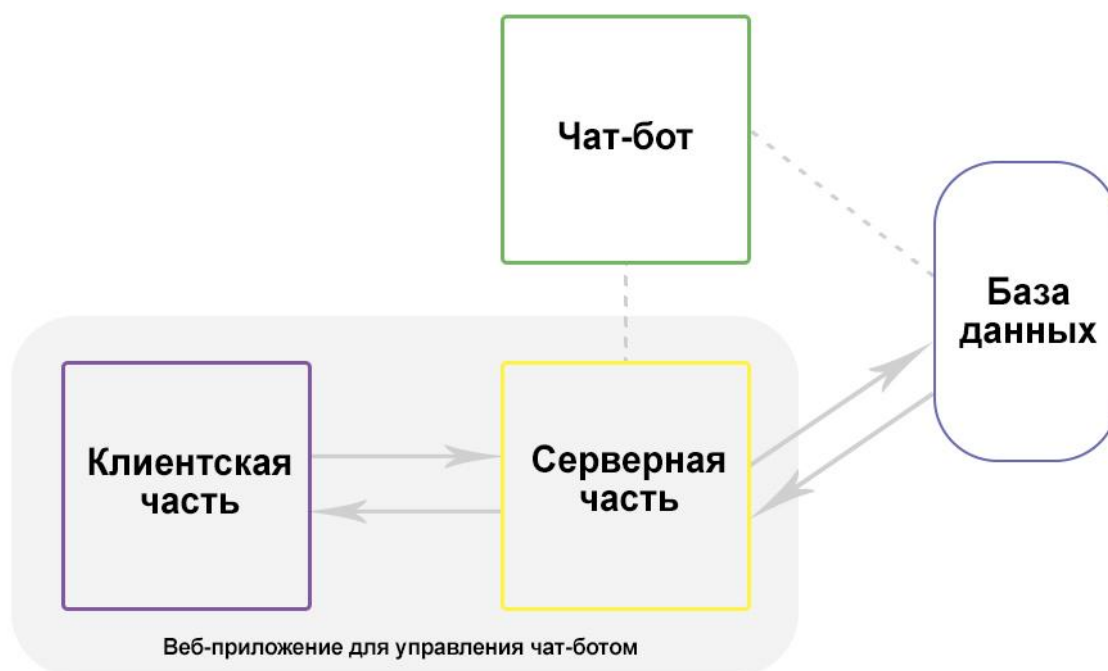


Рисунок 16 – Архитектура разрабатываемого веб-приложения

Рассмотрим выбранные инструменты для разработки веб-приложения подробнее.

Bootstrap – это открытый и бесплатный HTML, CSS и JS фреймворк, который используется веб-разработчиками для быстрой верстки адаптивных дизайнов сайтов и веб-приложений [37].

Для того чтобы начать работу (листинг 9) над интерфейсом приложения, для начала необходимо подключить CSS- и JS-библиотеки с помощью тегов `<link>` и `<script>` во всех \*.html файлах проекта.

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootst
rap.min.css"
integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJl
SAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstra
p.min.js"
integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpilMqu
VdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>
```

#### Листинг 9 – Подключение CSS- и JS-библиотек

##### Bootstrap в файл \*.html

Flask – это всемирно известный микрофреймворк, который предназначен для создания веб-приложений на языке программирования Python. Этот фреймворк предоставляет самые базовые возможности для создания веб-приложений, поэтому в рамках данной работы Flask будет использован в разработке серверной части приложения для управления чат-ботом [38].

Для того чтобы запустить сервер веб-приложения, необходимо выполнить код, который представлен на листинге 10.



```

from flask import Flask, render_template, json, request
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('myapp.html')

if __name__ == '__main__':
    app.run()

```

### Листинг 10 – Фрагмент кода запуска сервера приложения

Многопользовательское веб-приложение для управления чат-ботом для предоставления ответов на запросы имеет следующие возможности:

а) редактирование личного аккаунта пользователя:

1) выбор статуса профиля;

2) ввод и редактирование необходимых данных для функционирования чат-бота – api-token и id сообщества;

б) просмотр всех имеющихся ответов для чат-бота в профиле пользователя на главной странице приложения;

г) удаление и редактирование ответа и добавление нового;

д) запуск и отключение чат-бота.

Как было упомянуто ранее, разрабатываемое веб-приложение является многопользовательским, поэтому перед началом работы веб-приложение потребует пользователя авторизоваться в системе (рисунок 17).

Указанные пароли пользователями при регистрации шифруются с помощью алгоритма криптографического хеширования SHA-1 методом `sha1(password)` из Python-библиотеки `hashlib`.

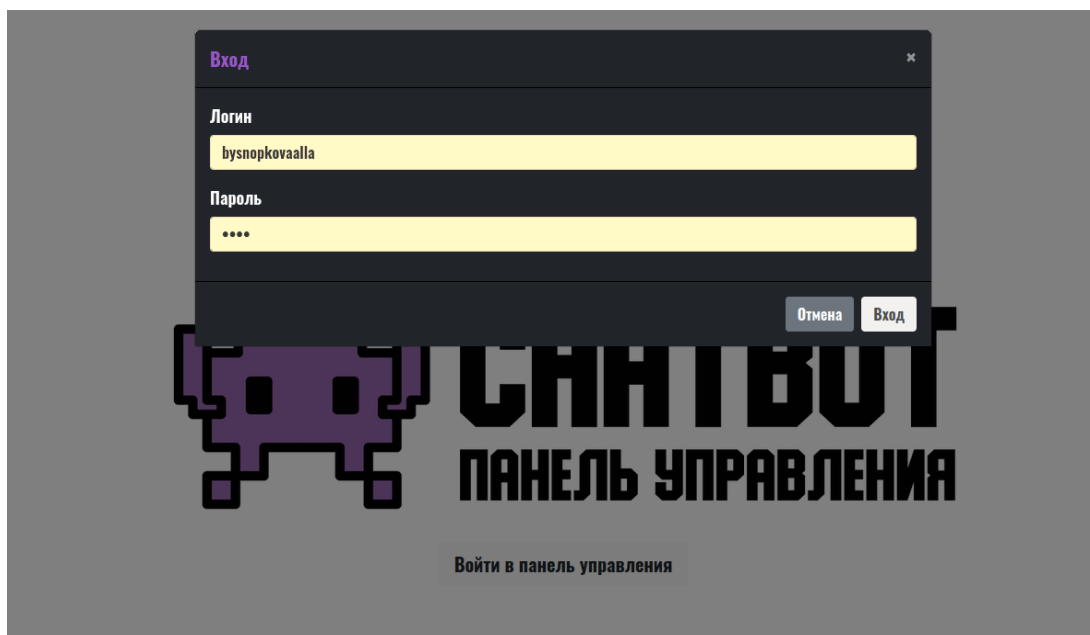


Рисунок 17 – Окно авторизации веб-приложения

После успешной авторизации пользователь попадает на главную страницу панели управления, содержащая таблицу ответов для чат-бота и кнопки, которые выполняют все вышеперечисленные функции (рисунок 15). Информация с сервера из базы данных передаётся в тэг `<table>` в виде JSON-структуры. На листинге 11 показан фрагмент кода, который получает данные из MongoDB и отправляет в виде словаря на клиент веб-приложения.

```
from flask import Flask, render_template, url_for, session,
redirect
import pymongo

app = Flask(__name__)

client = pymongo.MongoClient('host', 'port')
db = client['answers']

@app.route('/my', methods=['GET', 'POST'])
```

```

def my():

    if 'username' in session:
        answers_collection = db[session['username']]
        userprofile = db['users'].find({'username':
session['username']})[0]

        return render_template('maypp_start.html',
username=session['username'],userprofile = userprofile, answers
= answers_collection.find())

    return redirect(url_for('login'))

```

### Листинг 11 – Получение и отправка на клиент данных из базы MongoDB

Как только пользователь попадает в маршрут «/my», сервер сразу проверяет авторизован он в системе или нет. Если авторизован, то по логину, db[session[«username»]], программа возвращает таблицу ответов с помощью метода find() из Python-библиотеки pymongo, иначе перенаправляет его в маршрут «/login».

MongoDB – это система управления базами данных NoSQL с открытым исходным кодом. Она основана на модели хранения документов NoSQL. Объекты данных хранятся в виде отдельных документов внутри коллекции – это является альтернативой хранения данных в столбцах и строках традиционной реляционной базы данных [39].

Документ, который хранит ответы для чат-бота пользователя имеет следующую структуру (таблица 2).

Таблица 2 – Структура документа коллекции «answers» с примером

Поле	Пример
_id	ObjectId(«609b51d877950f2450571312»)
name	Цена обложки для ВКонтакте?
text	{user name}, динамическая обложка стоит 500 рублей (за 3 обложки для карусели) + 50 рублей (за +1 обложку). Отметим, что Вам надо будет оплатить услуги vk.com/dycover (цена 100 руб./мес.), чтобы эта обложка работала. Цена простой обложки варьируется в диапазоне 100-300 рублей. Зависит от сложности. Предоплата 50 %.
meta_keyword	обложка
keyword	[ «ВКонтакте», «динамическая»]

Добавить новый ответ в базу данных в приложении можно с помощью формы, которая показана на рисунке 18.

Рисунок 18 – Веб-форма для добавления нового ответа

Добавление нового ответа в базу данных производится с помощью метода `insert(newAns)`, где `newAns` – это структура JSON, принятая с формы клиента сервером. А редактирование и удаление – `update()` и `remove()` по `_id` выбранной пользователем записи соответственно. Веб-форма для редактирования ответа имеет точно такой же интерфейс, что и при добавлении нового ответа.


Перед запуском чат-бота для предоставления ответов на запросы необходимо пользователю указать `id` и `api-token` сообщества ВКонтакте (рисунок 19).

Рисунок 19 – Веб-форма настроек профиля

Помимо вышеуказанных параметров в настройках можно установить статус профиля в виде смайлика. Статус выполняет исключительно декоративную функцию профиля и был создан в развлекательных целях по аналогии с GitHub.

Документ, который хранит все настройки профиля пользователя имеет структуру, которая представлена в таблице 3.

Таблица 3 – Структура документа коллекции «users» с примером

Поле	Пример
_id	ObjectId(«60917bce2152bb6df7d3a0c4»)
username	bysnopkovaalla
status	
vk_token	d022 ***** b11f
id_group	164529320

После того, как все необходимые ответы в базу данных внесены и указаны id сообщества и уникальный ключ доступа, пользователь уже может запустить чат-бот.

Бот запускается при нажатии на кнопку «Запустить чат-бот», после чего активируется кнопка «Остановить чат-бот», при нажатии которой можно прекратить его активность. Если пользователь неверно ввел api-token в настройках профиля, то программа выдаст сообщение об ошибке авторизации бота в сообществе социальной сети ВКонтакте и попросит скорректировать ранее введенные им данные в личном кабинете.

На рисунке 20 изображено окно уведомления об ошибке доступа в случае неверно введенных пользователем данных.

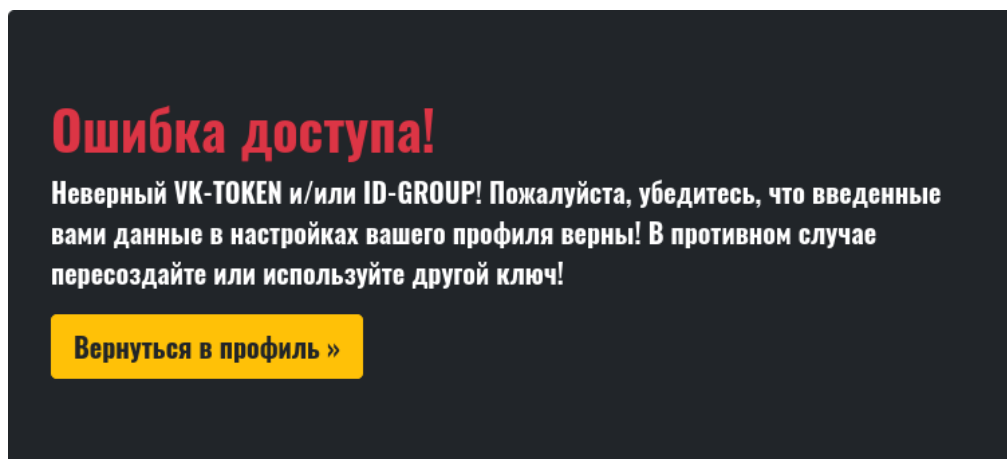


Рисунок 20 – Сообщение об ошибке доступа

### 3.2 Тестирование разработанного чат-бота с нечётким поиском

В данном параграфе протестируем разработанный чат-бот для предоставления ответов на запросы. Тестирование будет позитивным и интуитивным. Оно будет выполняться в двух разноплановых группах социальной сети Вконтакте:

- BYSNOPKOVAALLA ([vk.com/bysnopkovaalla](https://vk.com/bysnopkovaalla)) – это маленькое ВК-сообщество по предоставлению услуг мастера по высококачественной обработке фотографий в фотошопе и опытного видеомонтажера, в которой будет рассмотрено функциональное тестирование;

- Сериал «След» ([vk.com/serial\\_sled](https://vk.com/serial_sled)) – официальное сообщество телевизионного сериала «След» во Вконтакте, которое насчитывает более 175 тысяч участников, где будет выполнено ручное нагрузочное тестирование системы.

Сначала перейдем к функциональному тестированию чат-бота для ВК-группы BYSNOPKOVAALLA (рисунок 21).

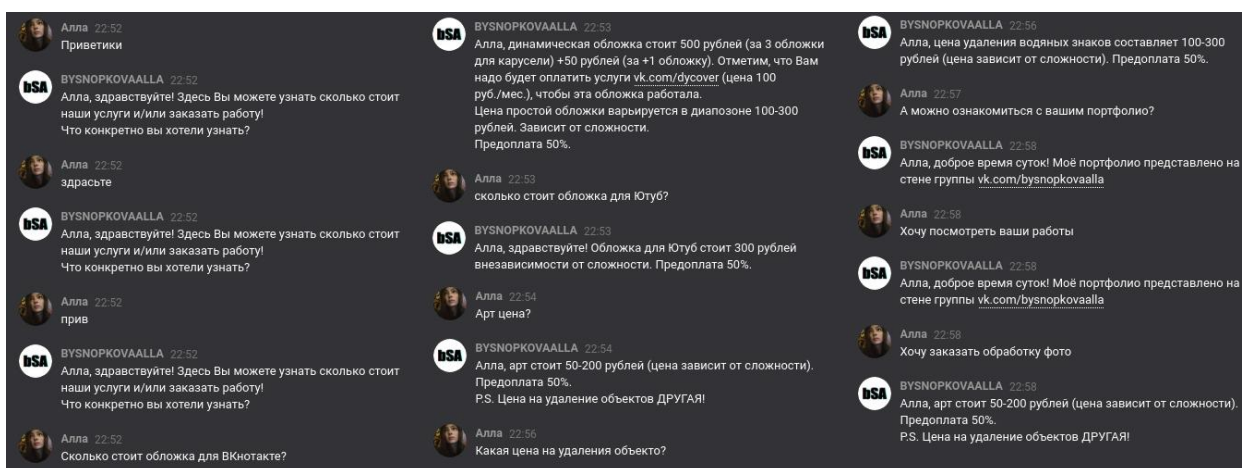


Рисунок 21 – Диалог с ВК-сообществом BYSNOPKOVAALLA

Были проведены пять проверок для функционального тестирования:

- проверка на успешную авторизацию чат-бота в сообществе BYSNOPKOVAALLA социальной сети ВКонтакте;
- проверка на успешное предоставление чат-ботом ответа на сообщение участника группы. Другими словами, отвечает ли бот на присланное ему сообщение;
- проверка на правильное обращение чат-бота к пользователю, то есть соответствует ли имя, указанное в сообщении бота, с именем подписчика приславшего сообщение в мессенджер группы;
- проверка на правильность предоставления чат-ботом ответа на запрос пользователя сети. Иными словами, соответствует ли контекст ответа бота с контекстом присланного вопроса от участника сообщества;
- проверка работоспособности технологии нечеткого поиска.

Все вышеперечисленные проверки для функционального тестирования чат-бота для предоставления ответов на запросы были выполнены успешно без замечаний.

Теперь перейдем к ручному нагрузочному тестированию.



В качестве инструмента ручного нагрузочного тестирования выступал мессенджер официального сообщества сериала «След» во ВКонтакте (рисунок 22). Такое тестирование стойкости к нагрузке чат-бота необходимо было провести, для того чтобы понять может ли справиться сервер с реальным количеством присылаемых в день сообщений участниками и посетителями группы.

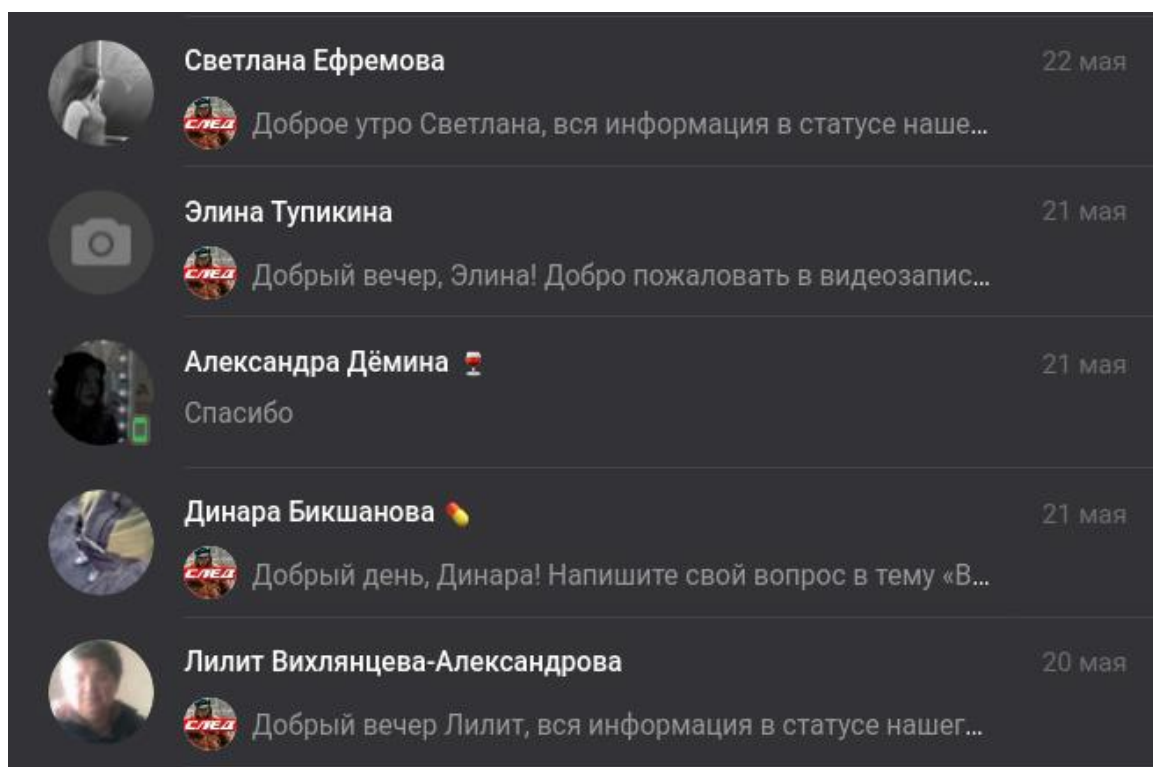


Рисунок 22 – Мессенджер официального ВК-сообщества  
сериала «След»

Тестирование чат-бота в группе проходило 5 дней – с 18 по 22 мая 2021 года. На рисунке 23 показан график соотношения отправленных и полученных сообщений сообществом сериала «След» в период тестирования чат-бота для предоставления ответов на запросы.

## Сообщения

На этом графике показано количество отправленных/полученных сообщений

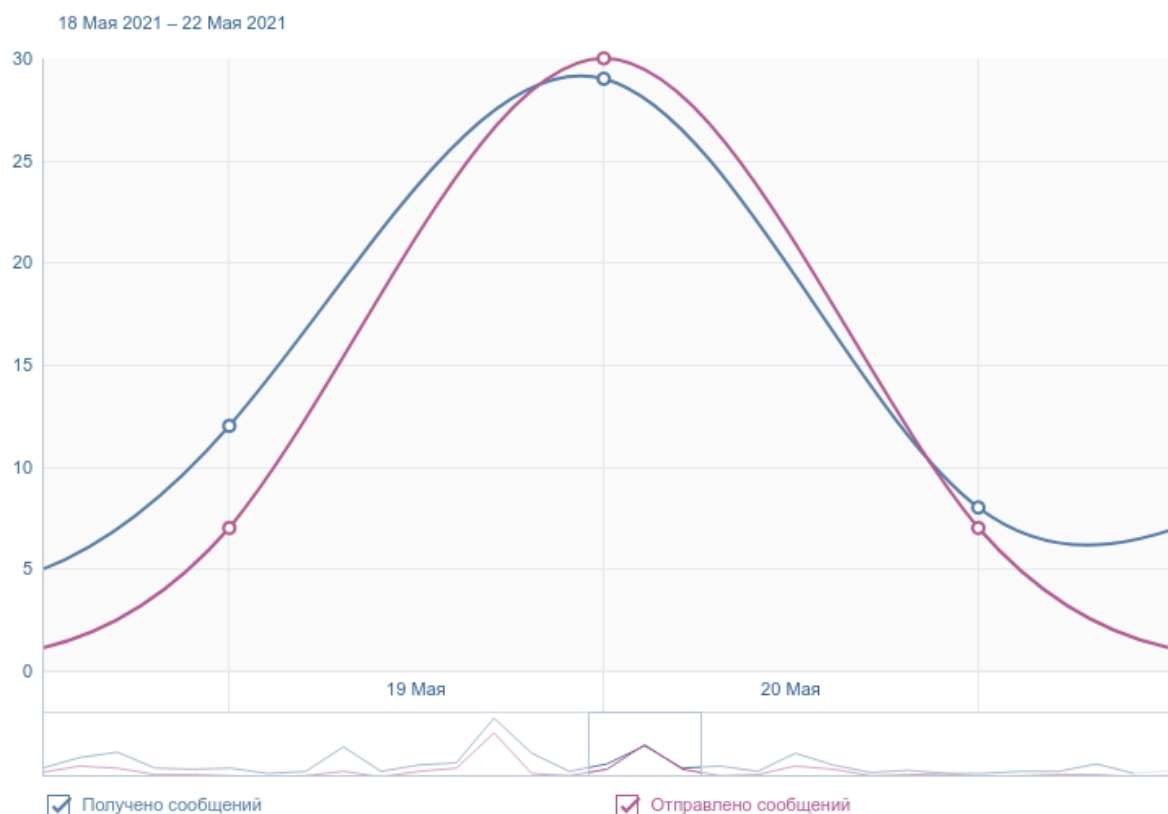


Рисунок 23 – Статистика сообщений сообщества за период работы чат-бота

Анализ графика соотношения полученных и отправленных сообщений показывает, что степень корреляции показателей относительно невелика. Оба показателя демонстрируют тенденцию точной синхронности, однако пики и спады совпадают нечасто. Это связано с тем, что чат-бот не смог дать нужный ответ некоторым пользователям и с ними уже работал реальный человек в лице администратора группы.

Отсюда следует, что чат-бот успешно прошел это тестирование без каких либо замечаний, но важно указать, что сообщества данной социальной сети на сегодняшний день не может похвастаться столь высокой

активностью, поэтому есть вероятность, что в других социальных сетях разработанный алгоритм бота может не выдержать огромную нагрузку.

Таким образом, были разработаны и успешно протестированы чат-бот для предоставления ответов на запросы и веб-приложение для управления этим чат-ботом. Важно отметить, что данный бот будет работать успешно только в том случае, если ключевые слова и фразы не будут между собой никак контекстно пересекаться. Если присутствует слишком много повторяющихся ключевых слов (в том числе и части слов, например, «сериал» и «серия») у абсолютно разных вопросов, то для их анализа будет нужен искусственный интеллект.

## ЗАКЛЮЧЕНИЕ

Основные результаты выпускной квалификационной работы (магистерской диссертации) состоят в следующем:

1 Изучена теоретическая основа об интернет-ботах. Есть четыре вида ботов – вредоносные, чат-боты, медиа-боты и веб-искатели, которые подчиняются правилам robots.txt.

2 Рассмотрена архитектура чат-бота для социальных сетей. Существуют несколько видов подключения бота к социальным сетям –API и синтаксический анализ.

3 Изучены все средства и инструменты разработки чат-ботов, а также способы их подключения к социальным сетям.

4 Изучены все средства и инструменты разработки веб-приложений.

5 Освоен принцип разработки чат-ботов с нечётким поиском. В разработке чат-бота с нечётким поиском использовался алгоритм нахождения расстояния Левенштейна.

6 Изучены методы и приёмы тестирования программных продуктов.

7 Разработан чат-бот для предоставления ответов на запросы для сообществ социальной сети ВКонтакте. Для подключения чат-бота к социальной сети ВКонтакте был использован инструмент API. Сама разработка велась на языке программирования Python.

8 Разработано веб-приложение для управления чат-ботом для предоставления ответов на запросы для сообществ в социальной сети ВКонтакте. Веб-приложение имеет клиент-серверную архитектуру с подключением к базе данных MongoDB.

9 Разработанный программный продукт был успешно протестирован. Критических ошибок найдено не было.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Джанарсанам С. Разработка чат-ботов и разговорных интерфейсов / С. Джанарсанам. – Москва: ДМК-Пресс, 2019. – 340 с. – ISBN: 978-5-97060-542-4.
2. О вредоносных скриптах: как они работают, чем опасны и как не столкнуться с ними // Москва: Dr. WEB. – 2021. – Антивирусная правДА! – URL: <https://www.drweb.ru/pravda/issue/?number=1159&lng=ru> (дата обращения: 18.04.2021).
3. Нестеров С. А. Основы информационной безопасности. Учебное пособие / С. А. Нестеров. – Санкт-Петербург: Лань, 2016. – 324 с. – ISBN: 978-5-8114-2290-6.
4. Акулич М. Чат-боты и маркетинг / М. Акулич. – Екатеринбург: ЛитРес, 2019. – 154 с. – ISBN: 978-5-4490-7160-6.
5. Хобсон Л. Обработка естественного языка в действии / Л. Хобсон, Х. Ханнес, Х. Коул. – Санкт-Петербург: Питер, 2020. – 576 с. – ISBN: 978-5-4461-1371-2.
6. Что такое социальные медиа-боты? // Лондон: Корпорация Cloudflare. – 2018. – Всё об интернет-ботах. – URL: <https://www.cloudflare.com/ru-ru/learning/bots/what-is-a-social-media-bot/> (дата обращения: 19.04.2021).
7. Center for Complex Networks and Systems Research. Online Human-Bot Interactions: Detection, Estimation, and Characterization. March 27, 2017. – URL: <https://arxiv.org/pdf/1703.03107.pdf> (дата обращения: 19.04.2021).
8. University of Reading. Turing Test success marks milestone in computing history. June 08, 2014. – URL: <http://www.reading.ac.uk/news-archive/press-releases/pr583836.html> (дата обращения: 19.04.2021).

9. Как работает поиск Яндекса // Москва: Компания Яндекс. – 2015. – Яндекс.Справка:           Помощь           вебмастеру.           –           URL: <https://yandex.ru/support/webmaster/yandex-indexing/site-indexing.html> (дата обращения: 20.04.2021).

10. Поисковые роботы Google // Маунтин-Вью: Корпорация Google. – 2020. – Документация: Расширенная поисковая оптимизация. – URL: <https://developers.google.com/search/docs/advanced/crawling/overview-google-crawlers?hl=ru> (дата обращения: 20.04.2021).

11. Использование файла robots.txt // Москва: Компания Яндекс. – 2015. – Яндекс.Справка:           Помощь           вебмастеру.           –           URL: <https://yandex.ru/support/webmaster/controlling-robot/robots-txt.html> (дата обращения: 20.04.2021).

12. Митчелл Р. Современный скрапинг веб-сайтов с помощью Python / Р. Митчелл. – Санкт-Петербург: Питер, 2021. – 816 с. – ISBN: 978-5-9775-3686-8.

13. Энж Э. SEO – искусство раскрутки сайтов / Э. Энж, Д. Спенсер, С. Стрикчиола. – Санкт-Петербург: БХВ-Петербург, 2017. – 336 с. – ISBN: 978-5-4461-1693-5.

14. Файл robots.txt // Санкт-Петербург: АО «Телерадиокомпания (ТРК) Петербург». – 2015. – Официальный сайт Пятого канала. – URL: <https://www.5-tv.ru/robots.txt> (дата обращения: 20.04.2021).

15. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения / Р. Мартин. – Санкт-Петербург: Питер, 2018. – 352 с. – ISBN: 978-5-4461-0772-8.

16. Лоре А. Проектирование веб-API / А. Лоре. – Москва: ДМК-Пресс, 2020. – 440 с. – ISBN: 978-5-97060-861-6.

17. Разработчикам // Санкт-Петербург: Штаб квартира ВКонтакте. – 2016. – Документация для разработчиков об использовании API ВКонтакте. –

URL: <https://vk.com/dev> (дата обращения: 21.04.2021).

18. API Телеграм // Берлин: Штаб квартира Телеграм. – 2015. – Документация для разработчиков об использовании API Телеграм. – URL: <https://core.telegram.org/api> (дата обращения: 21.04.2021).

19. Dale K. Data Visualization with Python and Javascript / K. Dale. – Sebastopol: O'Reilly Media, 2016. – 510 с. – ISBN: 978-1-49192-051-0.

20. Диков А. В. Клиентские технологии веб-программирования: JavaScript и DOM / А. В. Диков. – Санкт-Петербург: Лань, 2020. – 440 с. – ISBN: 978-5-8114-4074-0.

21. Форта Б. Изучаем регулярные выражения / Б. Форта. – Москва: Вильямс, 2019. – 192 с. – ISBN: 978-5-6041394-2-4.

22. Пакет BeautifulSoup4 [Электронный ресурс] // The Python Package Index (PyPI) is a repository of software for the Python programming language. – Режим доступа: <https://pypi.org/project/beautifulsoup4/> (дата обращения: 22.04.2021).

23. Яворски М. Python. Лучшие практики и инструменты / М. Яворски, Т. Зиаде. – Санкт-Петербург: Питер, 2021. – 560 с. – ISBN: 978-5-4461-1589-1.

24. Котеров Д. В. PHP 7: Наиболее полное руководство / Д. В. Котеров, И. В. Симдянов. – Санкт-Петербург: БХВ-Петербург, 2019. – 1088 с. – ISBN: 978-5-9775-3725-4.

25. Эванс Б. Java. Справочник разработчика / Б. Эванс, Д. Флэнаган. – Москва: Вильямс, 2019. – 592 с. – ISBN: 978-5-907144-61-3.

26. Метц С. Ruby. Объектно-ориентированное проектирование / С. Метц. – Санкт-Петербург: Питер, 2018. – 304 с. – ISBN: 978-5-4461-0875-6.

27. Левенштейн В. И. Двоичные коды с исправлением выпадений, вставок и замещений символов / В. И. Левенштейн. – Москва: Доклады Академий Наук СССР, 1965. – 163.4: с. 845-848.

28. Гасфилд Д. Строки, деревья и последовательности в алгоритмах.

Информатика и вычислительная биология / Д. Гасфилд. – Санкт-Петербург: Невский Диалект БВХ-Петербург, 2003. – 654 с. – ISBN: 0-521-58519-8.

29. Репозиторий FuzzyWuzzy [Электронный ресурс] // Github. Веб-сервис для хостинга IT-проектов и их совместной разработки. – Режим доступа: <https://github.com/seatgeek/fuzzywuzzy> (дата обращения: 23.04.2021).

30. Леоненков А. В. Нечёткое моделирование в среде MATLAB и fuzzy TECH / А. В. Леоненков. – Санкт-Петербург: БВХ-Петербург, 2005. – 736 с. – ISBN: 5-94157-087-2.

31. Грофф Р. SQL. Полное руководство / Р. Грофф, Э. Оппель, П. Вайнберг. – Киев: Диалектика, 2019. – 960 с. – ISBN: 978-5-8459-1654-9.

32. Садаладж П. NoSQL. Методология разработки нереляционных баз данных / П. Садаладж, М. Фаулер. – Киев: Диалектика, 2020. – 192 с. – ISBN: 978-5-907144-91-0.

33. Уиттакер Д. Как тестируют в Google / Д. Уиттакер, Д. Арбон, Д. Каролло. – Санкт-Петербург: Питер, 2014. – 320 с. – ISBN: 978-5-496-00893-8.

34. Бейзер Б. Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем / Б. Бейзер. – Санкт-Петербург: Питер, 2004. – 318 с. – ISBN: 5-94723-698-2.

35. API для чат-ботов // Санкт-Петербург: Штаб квартира ВКонтакте. – 2016. – Документация для разработчиков об использовании API для сообщений сообществ. – URL: [https://vk.com/dev/bots\\_docs](https://vk.com/dev/bots_docs) (дата обращения: 29.04.2021).

36. Пакет FuzzyWuzzy [Электронный ресурс] // The Python Package Index (PyPI) is a repository of software for the Python programming language. – Режим доступа: <https://pypi.org/project/fuzzywuzzy/> (дата обращения: 30.04.2021).

37. Морето С. Bootstrap в примерах / С. Морето. – Москва: ДМК-Пресс, 2017. – 314 с. – ISBN: 978-5-97060-423-6.



38. Гринберг М. Разработка веб-приложений с использованием Flask на языке Python / М. Гринберг. – Москва: ДМК-Пресс, 2016. – 272 с. – ISBN: 978-5-97060-138-9.

39. Бэнкер К. MongoDB в действии / К. Бэнкер. – Москва: ДМК-Пресс, 2017. – 394 с. – ISBN: 978-5-94074-831-1.