

**Санкт-Петербургский государственный электротехнический университет
“ЛЭТИ” им. В. И. Ульянова (Ленина)
(СПбГЭТУ “ЛЭТИ”)**

Направление подготовки: 09.03.01 “Информатика и вычислительная техника”
Профиль: “Вычислительные машины, комплексы, системы и сети”

**Факультет компьютерных технологий и информатики
Кафедра вычислительной техники**

К защите допустить:

Заведующий кафедрой

д. т. н., профессор

М. С. Куприянов

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
БАКАЛАВРА**

**Тема: “Чат-бот для корпоративных сервисов групповой
работы”**

Студентка

К. А. Мартынова

Руководитель

к. т. н., доцент

А. В. Тимофеев

Консультант по экономическому

обоснованию

ассистент

О. А. Скрынская

Консультант от кафедры

к. т. н., доцент, с. н. с.

И. С. Зуев

Санкт-Петербург
2021 г.

**Санкт-Петербургский государственный электротехнический университет
“ЛЭТИ” им. В. И. Ульянова (Ленина)
(СПбГЭТУ “ЛЭТИ”)**

Направление 09.03.01 “Информатика и вычислительная техника”

Профиль “Вычислительные машины, комплексы, системы и сети”

Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой ВТ
д. т. н., профессор
(М. С. Куприянов)
“___” _____ 202__ г.

**ЗАДАНИЕ
на выпускную квалификационную работу**

Студент Мартынова Ксения Александровна

Группа № 7306

1. Тема Чат-бот для корпоративных сервисов групповой работы

(утверждена приказом № _____ от _____)

Место выполнения ВКР: кафедра вычислительной техники

2. Объект исследования

Чат-бот для корпоративных сервисов групповой работы.

3. Предмет исследования

Корпоративный мессенджер Zulip.

3. Цель

Создание чат-бота для корпоративного мессенджера Zulip. Разработанный бот предназначен для использования в различных организациях, работники которых для коммуникации между собой используют мессенджер Zulip.

4. Исходные данные

Исходными данными являются документация Zulip API, содержащая в себе информацию по разработке встраиваемых ботов, документации библиотек, используемых при разработке программы, а также различные статьи из интернета.

5. Содержание

1. Обзор информации о корпоративных мессенджерах и чат-ботах.

2. Исследование возможностей мессенджера Zulip.

3. Разработка и реализация функционала чат-бота для Zulip.

6. Технические требования

Разработанный чат-бот должен позволять пользователям регистрироваться в системе, а также без ошибок выполнять предусмотренные его функционалом запросы.

7. Дополнительные разделы

Экономическое обоснование.

8. Результаты

Работающая программа, реализующая функционал чат-бота для мессенджера Zulip, пояснительная записка, реферат, аннотация и описание работы чат-бота.

Дата выдачи задания
« ____ » _____ 202__ г.

Дата представления ВКР к защите
« ____ » _____ 202__ г.

Руководитель
к. т. н., доцент
Студент

А. В. Тимофеев
К. А. Мартынова

**Санкт-Петербургский государственный электротехнический университет
“ЛЭТИ” им. В. И. Ульянова (Ленина)
(СПбГЭТУ “ЛЭТИ”)**

Направление 09.03.01 “Информатика и вычислительная техника”

Профиль “Вычислительные машины, комплексы, системы и сети”

Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

УТВЕРЖДАЮ

Заведующий кафедрой ВТ

д. т. н., профессор

(М. С. Куприянов)

“___” _____ 202__ г.

**КАЛЕНДАРНЫЙ ПЛАН
выполнения выпускной квалификационной работы**

Тема Чат-бот для корпоративных сервисов групповой работы

Студент Мартынова Ксения Александровна

Группа № 7306

№ этапа	Наименование работ	Срок выполнения
1	Изучение технического задания и постановка задачи	01.04 – 03.04
2	Обзор технической документации	04.04 – 07.04
3	Разработка функционала чат-бота	08.04 – 13.04
4	Написание программы	14.04 – 10.05
5	Тестирование и отладка программы	11.05 – 15.05
6	Написание и оформление пояснительной записки	16.05 – 08.06
7	Предварительное рассмотрение работы	10.06 – 20.06
8	Представление работы к защите	22.06

Руководитель

к. т. н., доцент

Студентка

А. В. Тимофеев

К. А. Мартынова

РЕФЕРАТ

Пояснительная записка содержит: 84 с., 47 рис., 6 табл., 7 приложений, 23 источника литературы.

Цель работы: разработка чат-бота для корпоративного мессенджера Zulip.

В выпускной квалификационной работе приводится обзор преимуществ корпоративных сервисов групповой работы, рассматриваются стратегии разработки функционала чат-бота для конкретных задач, а также исследуются возможности мессенджера Zulip, на основе которых разрабатывается функционал чат-бота.

Результатом работы является разработанный чат-бот с определенным функционалом, позволяющий автоматизировать рабочие процессы организации, а также осуществлять ряд запросов, которые позволяют пользователю сэкономить время на поиске различной корпоративной информации.

В перспективе программу можно дорабатывать и подстраивать под нужды конкретной организации.

ABSTRACT

The graduate qualification work gives an overview of the advantages of corporate group work services, strategies for developing chatbot functionality for specific tasks, as well as exploring the capabilities of the messenger Zulip, on the basis of which the chatbot functionality is developed.

The result of the work is a developed chat-bot with certain functional that allows to automate the work processes of the organization, as well as to carry out a number of requests, which allows the user to save time when searching for various corporate information.

In the future, the program can be refined and adjusted to the needs of a particular organization.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ	9
ВВЕДЕНИЕ	10
1 Корпоративные мессенджеры	12
1.1 Что представляют собой корпоративные мессенджеры	12
1.2 Функционал корпоративных мессенджеров	13
1.3 Zulip	15
1.4 Аналоги Zulip	21
2 Чат-боты	25
2.1 Какие бывают чат-боты	25
2.2 Примеры чат-ботов	26
2.2.1 Чат-бот InMind	26
2.2.2 Чат-бот Здоровье	28
2.2.3 Чат-бот Муга	30
2.3 Создание и настройка чат-бота	30
2.3.1 Проработка функционала	31
2.3.1 Выбор средств разработки	32
2.4 Чат-бот для Zulip	32
3 Разработка чат-бота	36
3.1 Список модулей программы	37
3.2 Структура БД чат-бота	37
3.3 Запуск программы на сервере	38
4 Описание функционала чат-бота	40
4.2 Получение информации о пользователях	42
4.3 Автоматическое заполнение документов	44
4.4 Составление расписания и списка задач на каждый день	47
4.5 Получение информации о нерабочих днях	50
4.6 Оповещение сотрудников	51
4.7 Обработка ошибок	52
5 Экономическое обоснование	55
5.1 Длительность и трудоемкость этапов разработки	56

5.2 Расчет размера заработной платы	57
5.3 Расчет обязательных социальных отчислений	59
5.4 Расчет затрат на материалы и покупные изделия	60
5.5 Расчет амортизационных отчислений	60
5.6 Расчет накладных расходов	63
5.7 Расчет совокупной величины затрат	63
5.8 Выводы	64
ЗАКЛЮЧЕНИЕ	65
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	66
ПРИЛОЖЕНИЕ А. Пример шаблона заявления	68
ПРИЛОЖЕНИЕ Б. Пример автоматически заполненного заявления	69
ПРИЛОЖЕНИЕ В. Список модулей программы и их назначение	70
ПРИЛОЖЕНИЕ Г. Фрагмент кода программы, в котором происходит обработка шаблона заявления на отпуск	76
ПРИЛОЖЕНИЕ Д. Распечатка XML-файла со списком нерабочих дней в 2021 году	78
ПРИЛОЖЕНИЕ Е. Фрагмент кода программы с парсингом XML-файла	79
ПРИЛОЖЕНИЕ Ж. Код модуля Schedule.py	81

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ПК – персональный компьютер.

ОС – операционная система.

БД – база данных.

API (Application programming interface) – описание способов, например, набор классов или функций, с помощью которых одна программа может взаимодействовать с другой программой.

VPS (virtual private server) – услуга предоставления виртуального выделенного сервера, запущенного на базе физического сервера, что предоставляет возможность устанавливать нужную операционную систему и программное обеспечение.

CPU (Central Processing Unit) – центральное процессорное устройство, которое выполняет вычислительные и логические операции с данными.

Оперативное запоминающее устройство (ОЗУ) – это энергозависимая память ПК, которая позволяет временно хранить данные и осуществлять к ним быстрый доступ.

SSH (Secure Shell) – это сетевой протокол, который используется для удаленного управления операционными системами, а также для передачи файлов, особенностью которого является шифрование трафика, что делает подключение безопасным.

Парсинг (англ. parsing – анализ) – это процесс автоматического сбора и структурирования данных сайта или файла.

ВВЕДЕНИЕ

В современных условиях существования, когда поток передаваемой и получаемой информации нескончаем, встает вопрос о ее структуризации и сортировке. А умение и возможность быстро находить нужную информацию, отделять ее от ненужной и бесполезной, обрабатывать ее и применять для решения различных задач очень важны, особенно в условиях рабочего процесса, когда важно не отвлекаться и не тратить время на обработку лишних данных.

В этом пользователям помогают корпоративные мессенджеры. Их существует огромное количество со своим уникальным функционалом. Они помогают не только структурировать рабочую информацию, но и улучшают внутреннюю коммуникацию компании, обеспечивают информационную поддержку сотрудников, а также оптимизируют различные рабочие процессы. Мессенджеры позволяют оперативно связываться с другими коллегами и задавать им возникшие вопросы или сообщать о проблемах. В мессенджерах можно хранить и искать информацию по работе, а также создавать различные чаты по проектам.

Для автоматизации внутренних процессов организации и упрощения информирования сотрудников в корпоративные мессенджеры внедряют чат-ботов. Чат-боты – это программы, имитирующие разговор с человеком. Они используются для выполнения рутинных задач, например, для поиска информации внутри компании или для оформления заявок и различных корпоративных документов. Чат-боты помогают сотрудникам рационально распоряжаться временем, значительно экономя его на поиске информации, например, в корпоративном чате, сокращая количество прямых коммуникаций между сотрудниками, что помогает донести информацию без искажений до исполнителя или руководителя.

Цель работы: создание чат-бота для корпоративного мессенджера Zulip.

Объект и предмет исследования: чат-бот для корпоративных сервисов групповой работы и корпоративный мессенджер Zulip.

Для достижения основной цели следует выявить ключевые особенности мессенджера Zulip, на основе полученных данных разработать функционал чат-бота, затем написать программу, реализующую разработанный ранее функционал, и запустить чат-бот на сервере.

В первой главе рассмотрены особенности и преимущества корпоративных мессенджеров, проведен анализ функционала мессенджера Zulip для дальнейшей разработки чат-бота.

Во второй главе исследованы методы реализации функционала и приведены примеры существующих чат-ботов, а также рассмотрены используемые средства разработки и первоначальная настройка бота для Zulip.

В третьей главе описана техническая составляющая реализации чат-бота: структура БД, список модулей и запуск программы на сервере.

В четвертой главе приведены описание функционала чат-бота и информация о сторонних библиотеках, используемых при разработке программы.

В пятой главе представлен дополнительный раздел по теме «Экономическое обоснование».

В приложении А приведен пример шаблона одного из заявлений для автоматического заполнения.

В приложении Б представлен пример заполненного заявления.

В приложении В представлена таблица со списком модулей программы и их назначением.

В приложении Г приведен фрагмент кода одного из модулей программы, в котором происходит обработка шаблона заявления на отпуск.

В приложении Д представлены содержание и скриншот XML-файла со списком нерабочих и сокращенных дней в 2021 году.

В приложении Е приведен фрагмент одного из модулей программы, где происходит парсинг XML-файла.

В приложении Ж приведен код модуля Schedule.py.

1 Корпоративные мессенджеры

В любой компании, большая она или маленькая, рано или поздно встает вопрос реализации обмена информацией между сотрудниками. Для выполнения такой задачи существуют корпоративные мессенджеры.

1.1 Что представляют собой корпоративные мессенджеры

Корпоративный мессенджер, так называемый корпоративный сервис групповой работы, – это приложение для ПК или мобильного устройства, с помощью которого сотрудники компании могут обмениваться сообщениями, файлами и различными документами [1]. Данный инструмент способствует оптимизации взаимодействия между сотрудниками как в предприятиях малого бизнеса, так и в крупных организациях. Мессенджеры позволяют быстро взаимодействовать и обмениваться данными в течение дня, а также помогают поддерживать вовлеченность сотрудников, и, таким образом, повышают продуктивность.

Корпоративный мессенджер – это хороший способ сэкономить время на работе, так как в чатах вполне допустимы быстрые, лаконичные ответы. В мессенджерах позволительно менее формальное общение, чем, например, в сообщениях, передаваемых по электронной почте. Также в рабочих чатах сотрудники могут поддерживать связь не только друг с другом, но и со своими руководителями или клиентами.

Данные средства общения предоставляют возможность решать рабочие вопросы быстро и эффективно. Они позволяют моментально передавать и получать информацию, а также удобны в использовании на мобильных устройствах и чаще всего поддерживают любые ОС. Также ими можно воспользоваться с помощью браузера и приложений для ПК. Мессенджеры просты в управлении и не требуют специального обучения для того, чтобы начать с ними работать.

Использование корпоративного мессенджера позволяет хранить всю рабочую информацию и документы, которые пересылают коллеги, в одном месте. Таким образом, сотрудник может с легкостью получить доступ к рабочим документам и файлам из дома. Также корпоративные чаты имеют преимущество перед обычными мессенджерами, так как ценная рабочая информация не теряется среди личных сообщений.

Если мессенджер будет использоваться для обмена конфиденциальными данными, то его лучше всего устанавливать на собственный корпоративный сервер. Тогда его применение будет обеспечивать почти сто процентное защищенное взаимодействие между сотрудниками.

1.2 Функционал корпоративных мессенджеров

В корпоративных мессенджерах пользователи могут создавать отдельные группы и темы для различных проектов или обсуждений. Сотрудники могут присоединяться только к тем беседам, которые имеют к ним отношение, что повышает производительность труда сотрудников и, следовательно, экономит деньги компании.

В чатах можно отмечать конкретных людей, чтобы вовлечь их в разговор. А в случае непредвиденных критических ситуаций с помощью мессенджеров можно уведомлять об этом всех сотрудников. Благодаря push-уведомлениям в реальном времени они не смогут пропустить какое-либо важное сообщение и узнают о возникшей проблеме независимо от своего местонахождения.

Конечно, если в рабочих чатах не прекращается поток сообщений, то бесконечные уведомления на телефоне или на компьютере будут отвлекать от работы. Но, чтобы справиться с отвлечением работников на уведомления о сообщениях, у корпоративных мессенджеров есть ряд функций, которые могут препятствовать этому. Например, можно отключить уведомления в менее важ-

ных чатах или оставить возможность оповещений только при упоминании человека в беседе. Данные функции не позволяют ненужным сообщениям отвлекать внимание пользователей и предоставляют работникам возможность сосредоточиться на выполнении важных рабочих задач.

Корпоративные мессенджеры позволяют интегрировать различные приложения. Например, можно интегрировать Google Drive, что позволит пользователям работать над документами и при этом одновременно получать доступ к своим сообщениям. Также компании могут создавать и интегрировать в свой мессенджер специализированные приложения, отвечающие их потребностям. Например, компании могут настроить приложение-планировщик встреч, которое помогает пользователям планировать встречи, приглашать участников, получать отзывы участников о местах проведения встреч, а также просматривать календари с расписанием команды.

Таким образом, разные мессенджеры могут иметь свой уникальный функционал. Например, наличие различных интеграций, добавление задач для сотрудников, маскировка IP-адреса и так далее. Но основа каждого корпоративного мессенджера – это возможность обмена текстовыми и аудио сообщениями, создание различных чатов и видеозвонки.

В итоге можно составить список основных качеств мессенджеров, подходящих для корпоративного применения:

- 1) простота использования;
- 2) наличие набора основных функций: командный чат, личные беседы, хранение и обмен медиафайлами, голосовые и видеозвонки;
- 3) безопасность;
- 4) адекватная цена;
- 5) наличие интеграций с другими приложениями.

Некоторые мессенджеры позволяют внедрение чат-ботов.

Чат-боты – это программы, которые автоматизируют общение с пользователем [2]. Подробнее о них будет рассказано позже.

Целью данной выпускной квалификационной работы является создание чат-бота для корпоративного мессенджера Zulip.

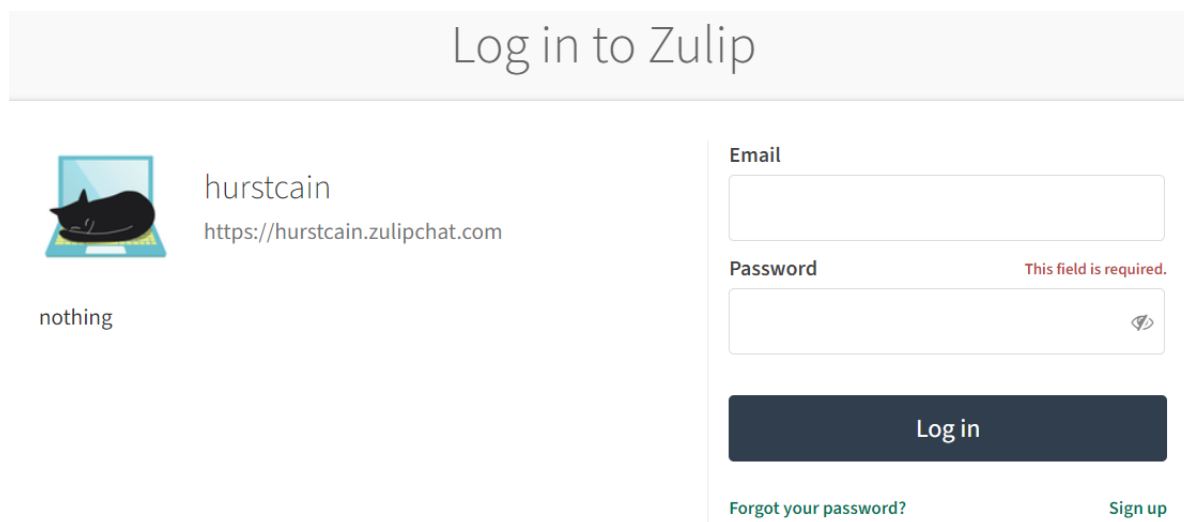
1.3 Zulip

Zulip – программное обеспечение для совместной работы, центром которого являются чаты внутри групп пользователей [3]. Zulip является корпоративным мессенджером с открытым исходным кодом.

Чтобы воспользоваться мессенджером, нужно создать в системе свою организацию. Для нее можно настроить имя, описание и картинку профиля. После создания можно приглашать сотрудников присоединиться к чату.

Пользователи могут авторизоваться в мессенджере с помощью аккаунтов электронной почты, GitHub, GitLab, Google, а также SAML. Для каждой организации можно выбрать несколько способов авторизации или даже один. Логичнее всего производить авторизацию с помощью аккаунтов электронной почты.

Страница входа вместе с профилем организации, созданного для выполнения данной выпускной квалификационной работы представлены на рисунке 1.1.



The screenshot displays the Zulip login interface. At the top, a light gray header contains the text "Log in to Zulip". Below this, the organization's profile is shown on the left, featuring a laptop icon with a black cat on the screen, the name "hurstcain", the URL "https://hurstcain.zulipchat.com", and the text "nothing" below the icon. To the right of the profile is the login form, which includes an "Email" input field, a "Password" input field with a red error message "This field is required." and an eye icon for toggling visibility, a dark blue "Log in" button, and two links at the bottom: "Forgot your password?" and "Sign up".

Рисунок 1.1 – Профиль организации и страница авторизации

Чаты в Zulip делятся на потоки. А каждый поток, в свою очередь, может иметь несколько топиков. Это уникальная модель потоков, в которой каждое

сообщение, помимо содержания, также имеет свой топик. Такая структура чатов позволяет осуществлять асинхронное общение. Например, если в команде находятся люди из разных часовых поясов, которые не могут взаимодействовать друг с другом в реальном времени, то им легче будет просматривать и отвечать на уже отсортированные сообщения в отдельном топике, а не пролистывать полный диалог в поисках нужной информации. Или можно рассмотреть ситуацию, когда руководящие сотрудники, часто проводящие большую часть рабочего дня на совещаниях, не в состоянии отследить полный диалог своих подчиненных в чате. Но с помощью топиков можно отсортировать ненужную информацию, что позволяет даже в конце дня спокойно принимать участие в обсуждениях.

Потоки могут быть приватными и публичными. Сообщения в приватном потоке могут просматривать только те, кто был приглашен в поток, в то время как к публичному потоку может присоединиться любой человек в организации.

На рисунке 1.2 представлен скриншот иерархии чата в Zulip. На нем можно видеть, как для каждого потока существует несколько входящих в него топиков.

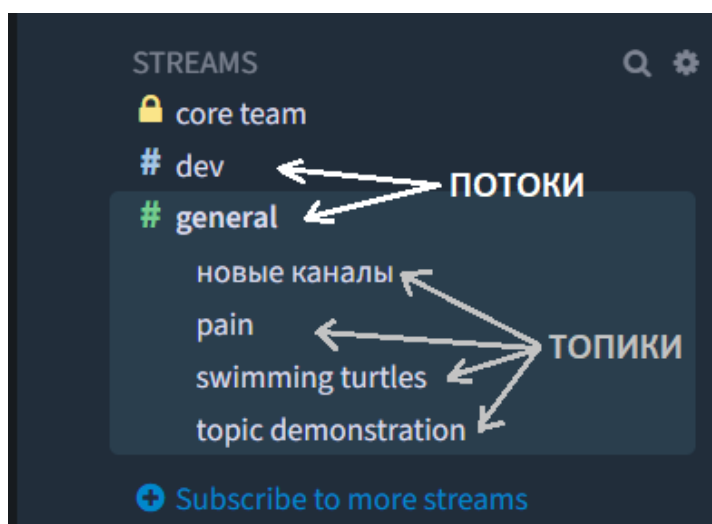


Рисунок 1.2 – Скриншот иерархии чата в Zulip

На рисунке 1.3 представлен скриншот сообщения, отправленного в топик pain потока general.

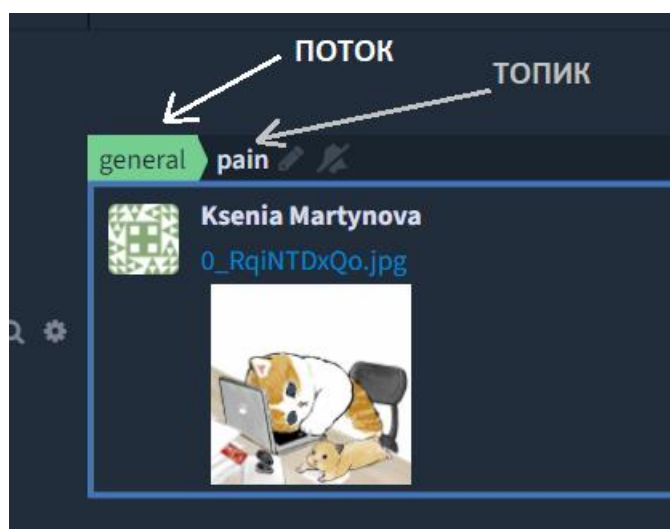


Рисунок 1.3 – Сообщение в топике pain потока general

Присланные в топик сообщения можно редактировать или пересылать в другой топик. Эта функция продемонстрирована на рисунке 1.4. Также можно ограничить время, в течение которого появляется возможность отредактировать отправленное сообщения. В тестовой организации, которая была создана для написания данной выпускной квалификационной работы, сообщения можно редактировать в течение 10 минут.

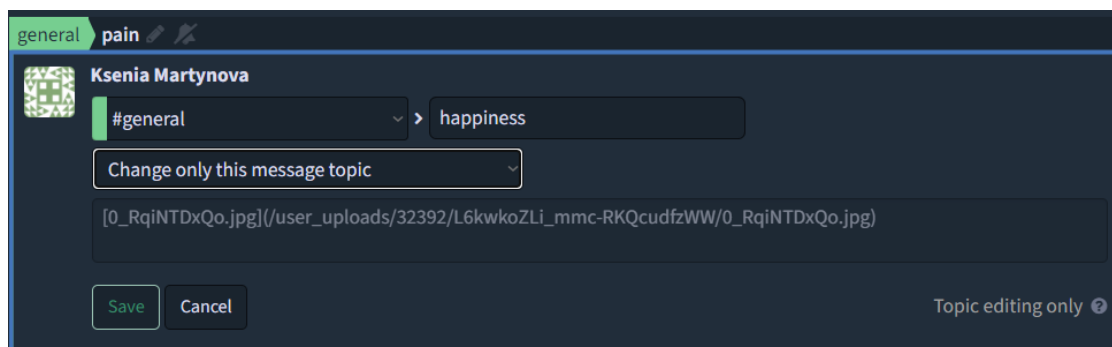


Рисунок 1.4 – Перенаправление сообщения в другой топик

Помимо потоков в мессенджере также есть приватные сообщения, где можно общаться с каждым из членов организации. На рисунке 1.5 представлен скриншот меню, где можно видеть наличие приватных сообщений.

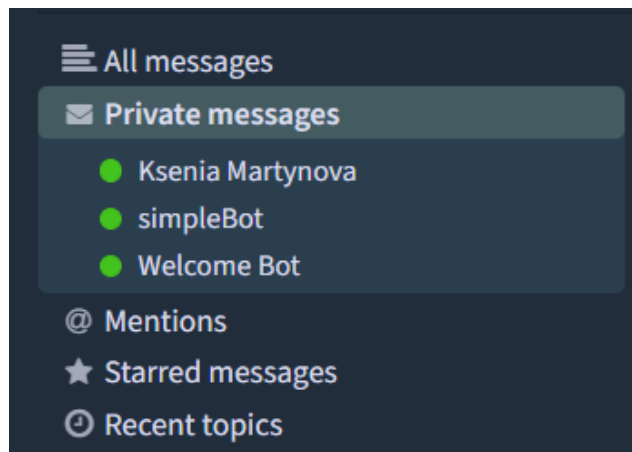


Рисунок 1.5 – Меню мессенджера Zulip

В мессенджере Zulip можно обмениваться картинками и различными файлами, осуществлять видеозвонки, реагировать на сообщения, а также упоминать человека, бота или сразу всех людей, состоящих в топике. На рисунке 1.6 приведен наглядный пример, как выглядят сообщения с упоминаниями конкретного пользователя или сразу всех в организации.

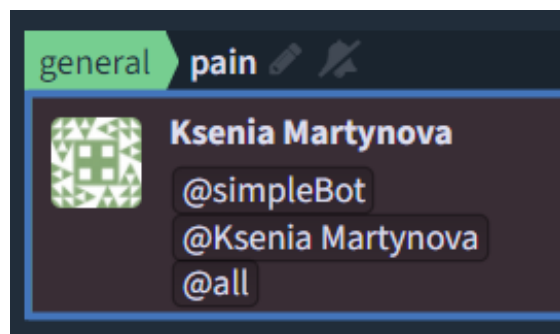


Рисунок 1.6 – Пример использования упоминаний в чате

Интересной функцией в Zulip является форматирование текста. Например, можно присылать в сообщения куски кода, которые будут отображаться как в продвинутом редакторе. На рисунке 1.7 приведен пример отформатированного сообщения с кодом на языке программирования Python.

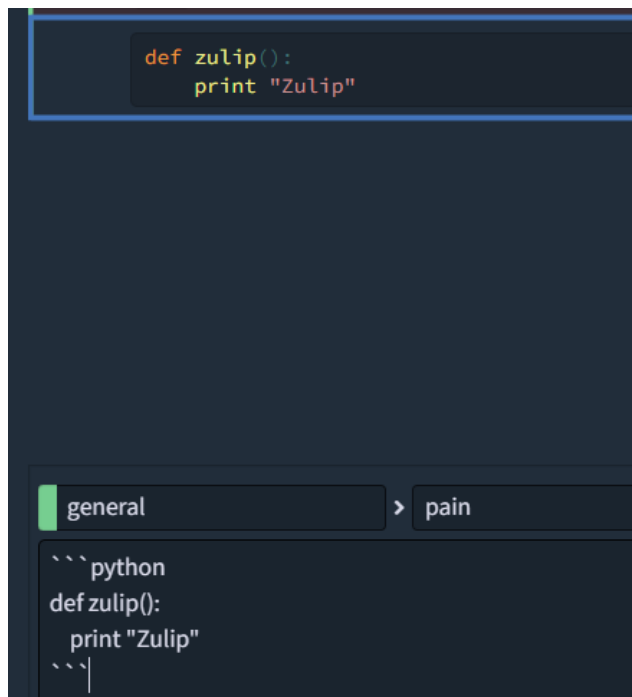


Рисунок 1.7 – Отформатированное сообщение с кодом на языке Python

Также присутствует обычное оформление текста, как в текстовом редакторе. Можно писать курсивом, полужирным и зачеркнутым шрифтом, что представлено на рисунке 1.8.

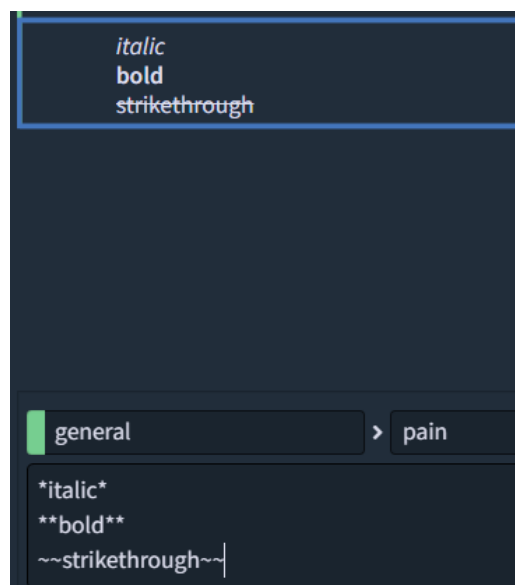


Рисунок 1.8 – Скриншот, демонстрирующий различные стили текста в чате

В чат можно отсылать отформатированные математические уравнения. Пример такого сообщения представлен на рисунке 1.9.

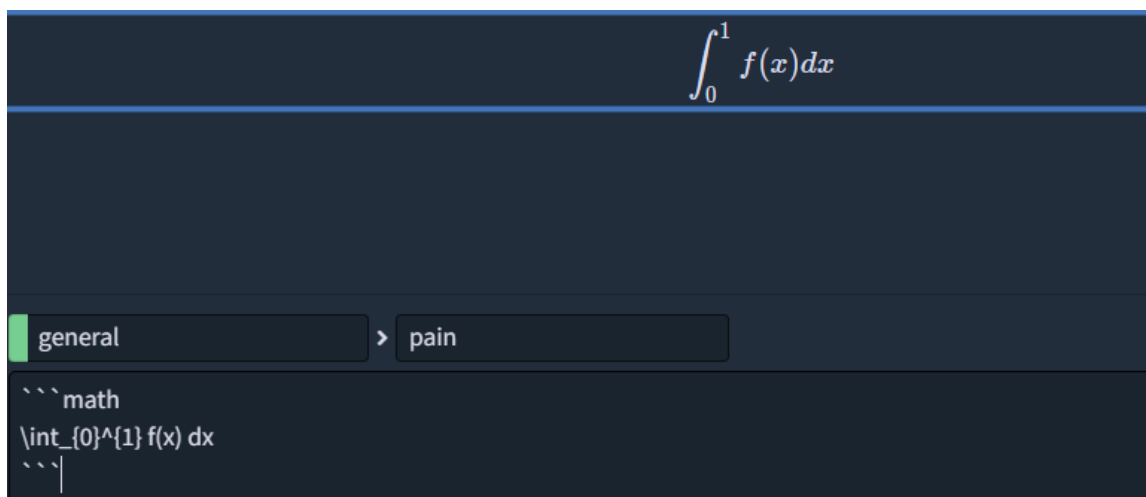


Рисунок 1.9 – Отформатированное математическое уравнение в чате

В Zulip есть возможность присылать сообщения со скрытым текстом. Они позволяют структурировать большие сообщения, таким образом пользователь может просто раскрыть нужный ему подзаголовок, а не просматривать текст целиком. Пример такого сообщения представлен на рисунке 1.10.

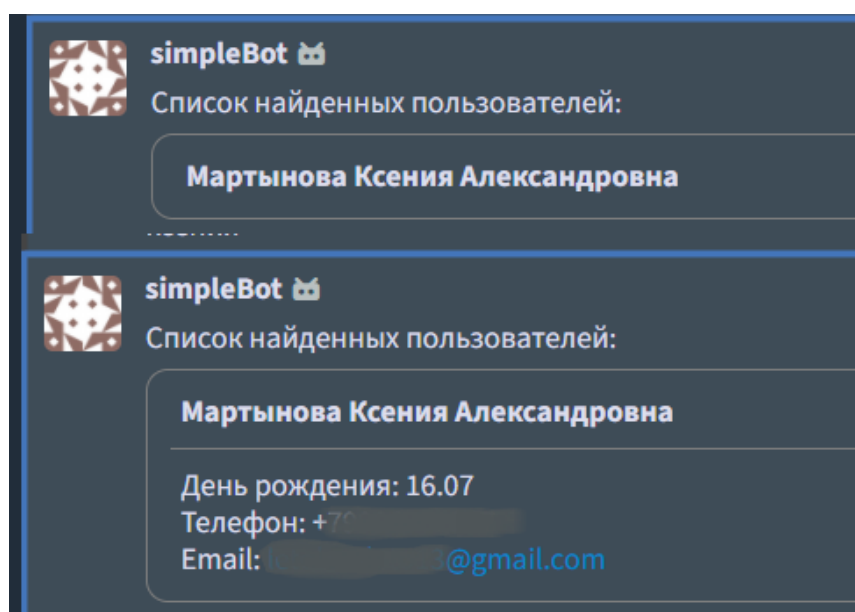


Рисунок 1.10 – Сообщение со скрытым текстом

В Zulip существует более 110 интеграций с другими сервисами, которые позволяют расширить функционал мессенджера. Среди них GitHub, Trello, Twitter, YouTube, Jira, Instagram, Google Translate, Google Calendar, Dropbox и много других [4].

Мессенджер Zulip можно устанавливать как на стороннее облако, аренда которого предоставляется мессенджером за платную подписку, так и на свой собственный сервер, тогда не придется платить за аренду. Существует и бесплатный тариф пользования мессенджером без установки его на свой сервер, но он имеет ряд ограничений: лимит истории поиска на 10 000 сообщений и на хранение файлов общим объемом до 5 ГБ на всех пользователей. Платная подписка не имеет ограничений на поиск по сообщениям, а также предоставляет файловое хранилище на 10 ГБ для каждого пользователя. За полный тариф придется платить \$6,67 в месяц за каждого пользователя, если вносить оплату сразу за год, а если платить ежемесячно, то цена \$8 [5].

1.4 Аналоги Zulip

Помимо Zulip существуют и другие мессенджеры, которые могут использоваться для взаимодействия между сотрудниками. Ниже представлен список с рассмотрением функционала каждого из наиболее известных мессенджеров.

1) Slack – корпоративный мессенджер, появившийся одним из первых на рынке. Помимо обычного обмена сообщениями, в Slack можно отправлять голосовые сообщения, демонстрировать экран собеседнику, осуществлять видеозвонки. Также данный мессенджер имеет синхронизацию с такими сервисами как: Trello, Google Docs, Google Drive и т. д. [6].

Приложение работает по протоколу HTTPS и имеет поддержку двухфакторной аутентификации. Помимо этого в мессенджере имеется функция kill switch, суть которой заключается в том, что если существует угроза утечки данных, то администратор чата может заставить всех пользователей одновременно сменить пароли.

В Slack присутствует возможность внедрения в корпоративный чат бота, который автоматизирует работу команды.

Одним из главных недостатков Slack является то, что мессенджер платный: \$6,67 в месяц за пользователя [7], а бесплатная версия имеет ряд ограничений.

2) Microsoft Teams – корпоративный мессенджер, разработанный компанией Microsoft в качестве аналога Slack. В приложении можно обмениваться сообщениями и файлами с сотрудниками. Имеется интеграция с различными приложениями от Microsoft: Word, Skype, SharePoint, OneNote, Planner и Power BI. Microsoft Teams является платным мессенджером и по сравнению с конкурентами имеет небогатый бесплатный функционал.

3) RocketChat – корпоративный мессенджер с открытым исходным кодом. Полностью бесплатный, если устанавливать Rocket Chat на собственный сервер и разворачивать на нем свои базы данных. Это гарантирует полную конфиденциальность. Также можно арендовать сервер, тогда оплата будет \$3 за пользователя [8]. Есть возможность пользоваться мессенджером бесплатно и без установки его на свой сервер, но тогда функционал будет ограничен.

4) Twist – корпоративный мессенджер, разработанный с целью помочь сотрудникам меньше отвлекаться на сообщения в чатах и больше заниматься работой. В Twist можно создавать проекты и для каждой задачи создавать отдельные обсуждения с различными членами команды. Также здесь можно сообщение превращать в задачу.

Мессенджер поддерживает интеграцию с GitHub и Zapier.

Данным сервисом можно пользоваться бесплатно. Но если команде необходим доступ к полной истории сообщений, то придется платить \$5 долларов в месяц за одного пользователя.

5) Myteam – корпоративный мессенджер для бизнеса от компании Mail.ru Group. С помощью данного мессенджера возможно контролировать доступ к чату: вручную создавать аккаунты сотрудников, блокировать доступ к информации, а также назначать администраторов. Таким образом, уволившийся сотрудник или посторонний человек не смогут просматривать чаты.

Myteam можно развернуть на локальном сервере своей компании. Также присутствует обычный вариант установки мессенджера с арендой серверов. В таком случае есть бесплатная и платная версии.

6) Dialog Messenger – корпоративный мессенджер, в котором можно создавать каналы для централизованного информирования сотрудников и чаты для проектов или команд. Сервис предоставляет программистам набор средств разработки, который позволяет создавать различные встраиваемые программы, например, чат-ботов.

Мессенджер можно установить на серверах своей компании или арендовать частное облако, предоставляемое Dialog Messenger.

7) Amo – относительно молодой мессенджер, запущенный в эксплуатацию в 2019 году, предназначенный в основном для малого бизнеса. В данном сервисе профили сотрудников настраиваются вручную руководителем, указываются телефон, почта, должность и устанавливается фотография профиля. Здесь, чтобы пообщаться с сотрудником, не нужно добавлять его контакты, как, например, в Telegram. Существуют бесплатный (до 200 пользователей) и платный тарифы.

8) Signal – мессенджер, являющийся одним из самых защищенных. Он предоставляет пользователям стандартный набор функций для общения: переписка, отправка голосовых и видеосообщений, пересылка фото, видео и файлов, а также многое другое.

Сервис можно установить на ПК, планшеты и телефоны. У мессенджера нет версии для браузера, так как ее сложно реализовать надежным образом. Связаться с человеком в Signal нельзя, если не знаешь его телефонного номера, в отличие от, например, Telegram. Сервис не хранит у себя список контактов пользователей в открытом виде, поэтому невозможно узнать, кто с кем общается, и построить социальный граф всех пользователей. Также Signal не может узнать о своем пользователе какую-либо информацию, так как при передаче

данных явно указывается только получатель, но не отправитель, а звонки, и переданные сообщения и файлы защищены сквозным шифрованием.

В мессенджере присутствует применяемая для всех чатов функция удаления сообщений через некоторое время или после единичного просмотра собеседником.

Если пользователь подключил к своему аккаунту новое устройство, то все переписки удаляются. Это реализовано для таких случаев, когда злоумышленник получил доступ к номеру телефона пользователя, например, к симке, и теперь имеет возможность просмотреть историю сообщений.

Signal является полностью бесплатным мессенджером. Единственное, у сервиса есть ряд ограничений. Так, например, максимальный размер пересылаемых файлов достигает 100 Мб, а видео – 500 Мб. В группе не может состоять более 1000 человек. А также нельзя редактировать уже отправленные сообщения.

9) VIPole Secure Messenger – мессенджер, позиционирующий себя как защищенный. Он хранит зашифрованную историю сообщений не на сервере, а на самом устройстве, и синхронизирует ее, когда нужно. Данный мессенджер позволяет изменять голос при звонках, маскировать IP-адрес, запрещает делать скриншоты, а также поддерживает закодированные голосовые сообщения и видео звонки. Также присутствует возможность редактировать отправленные сообщения.

Существуют платная и бесплатная версии приложения. Бесплатная версия предоставляется с ограничениями.

10) Telegram – один из самых популярных мессенджеров, который также является медийной площадкой. У Telegram есть достаточно удобные библиотеки для разработки чат-ботов. С помощью них можно создавать многофункциональных ботов, которые удобны в использовании. В открытом доступе очень много руководств по созданию ботов для Telegram, а также много готовых примеров. Таким образом, для разработки бота это идеальная платформа.

Сервис совершенно бесплатный. Но так как Telegram является публичным мессенджером, человек может переслать информацию из рабочего чата кому угодно, и это никак не отследить. Также если сотрудников в компании много, и существует необходимость в большом количестве чатов, то ориентироваться в мессенджере, в котором помимо рабочих чатов присутствуют личные диалоги и публичные каналы, неудобно.

2 Чат-боты

Чат-бот – это компьютерная программа, которая помогает пользователям решать проблемы и выполняет различные запросы без участия человека. Чаще всего чат-боты запрограммированы на распознавание фраз или слов, которые используют для предоставления ответов или ссылок на информацию. Иногда их называют виртуальными помощниками.

Боты применяются в различных случаях: для поддержки клиентов, осуществления продаж, для автоматизации выполнения рутинных офисных задач, таких как поиск информации, заполнение документов и т. д., а также в качестве источника развлекательного контента (например, игровые боты). Эти программы оптимизируют повторяющиеся рутинные операции и освобождают время для более сложных задач.

Чат-боты являются дополнением уже существующего функционала и альтернативным методом поиска информации в рабочей среде, а не единственным способом для пользователей установить контакт или найти помощь.

2.1 Какие бывают чат-боты

Существуют различные виды чат-ботов:

- чат-боты на основе меню, которые следуют по заранее назначенному сценарию и предоставляют пользователю выбор из нескольких определенных вариантов запросов;

- чат-боты с распознаванием ключевых слов, которые читают и интерпретируют ключевые слова, а затем отвечают на набранные запросы;
- продвинутые чат-боты, которые используют возможности обработки естественного языка (NLP) и могут обучаться на основе непрерывного взаимодействия с пользователем.

Бывает, что чат-боты сочетают в себе эти две или три разновидности. Например, программа, которая использует и распознавание ключевых слов, и структуру меню.

2.2 Примеры чат-ботов

Существует большое количество чат-ботов для различных мессенджеров. Есть такие, которые уже запущены на платформе, например, как в Telegram. Чтобы воспользоваться чат-ботом в Telegram, нужно ввести в поиске его название. А есть боты, которые необходимо внедрять в свой корпоративный чат. Так работает, например, в мессенджере Zulip.

Ниже представлены примеры некоторых чат-ботов для Telegram и Zulip.

2.2.1 Чат-бот InMind

InMind [19] – это бот для Telegram, который помогает в изучении новых английских слов. При первом взаимодействии с ним дается возможность выбрать один из трех уровней сложности.

Суть данного чат-бота заключается в том, что тебе дают список из слов на английском, а твоя задача выкинуть из них те, перевод которых уже знаешь. Если ни одного знакомого слова не остается, то бот предлагает тренировку на запоминание этих слов. Сначала он задает вопросы с вариантами ответов, а потом просит напечатать на клавиатуре заданное слово в переводе на английский или русский.

Такие небольшие тренировки можно проводить раз в день, а, чтобы о них не забыть, бот предоставляет выбор времени для отправления напоминаний.

На рисунке 2.1 представлен скриншот меню чат-бота. InMind предоставляет возможность соревноваться в игре со своими друзьями или случайным противником, а также посмотреть статистику. В любой момент можно зайти в настройки и поменять расписание уведомлений о предстоящей тренировке.

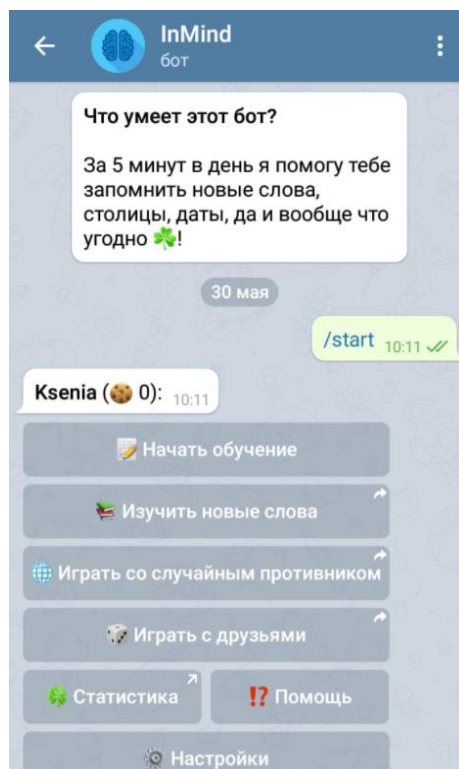


Рисунок 2.1 – Меню бота InMind

Выбор сложности и процесс игры по запоминанию английских слов представлен на рисунке 2.2.



Рисунок 2.2 – Процесс тренировки

2.2.2 Чат-бот Здоровье

Здоровье [20] – это бот для Telegram. Он позволяет получить инструкции к препаратам, провести первоначальную диагностику болезни в зависимости от указанных симптомов, а также записаться на прием в поликлинику или частный медицинский центр, если у пользователя есть полис обязательного медицинского страхования.

На рисунке 2.3 представлен скриншот меню бота. Оно представлено в виде кнопок, которые при нажатии автоматически отправляют запрос.

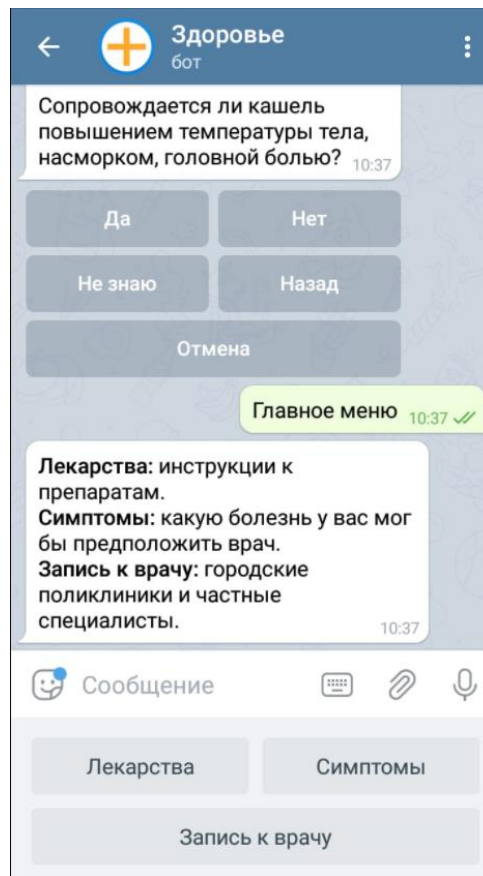


Рисунок 2.3 – Меню бота Здоровье

На рисунке 2.4 показан скриншот со списком симптомов и с примером ответа, в котором указан предположительный диагноз.

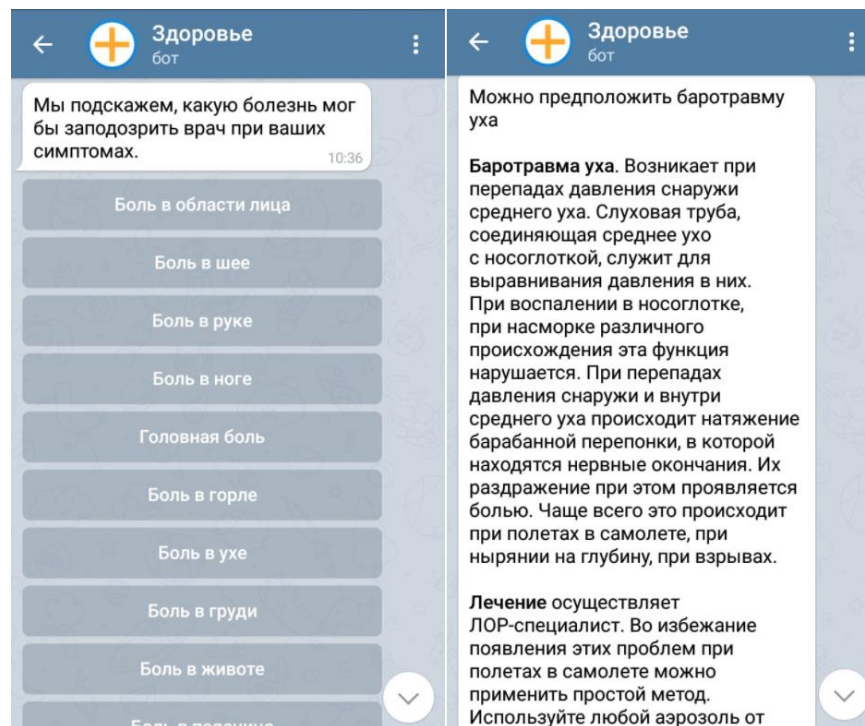


Рисунок 2.4 – Список симптомов и ответ с диагнозом бота Здоровье

2.2.3 Чат-бот Myra

Myra [21] – это бот для Zulip, с помощью которого можно осуществлять перевод с любого языка на английский, находить тексты песен, конвертировать валюту, узнавать подробности о фильмах, а также получать список праздников в текущем году. Данное приложение работает только на английском языке.

Чтобы воспользоваться ботом, нужно написать в чате сообщение, первое слово в котором – это название бота. Приложение корректно воспринимает только специально структурированные запросы. Для примера на рисунке 2.5 представлен скриншот с просьбой показать подробности о фильме Black Panther. Запрос в данном случае состоит из названия бота – myra, потом идет ключевое слово movie, которое определяет, какую именно функцию программа должна выполнить, а затем передаются аргументы, то есть название фильма, информация о котором присутствует в ответе.



Рисунок 2.5 – Пример взаимодействия с ботом Myra

2.3 Создание и настройка чат-бота

Основные этапы создания чат-бота: разработка функционала, выбор платформы для разработки, изучение платформы, выбор средств разработки, непосредственно написание программы, тестирование и внедрение.

2.3.1 Проработка функционала

Перед тем, как начать разработку программы, следует определиться с функционалом чат-бота.

Чтобы выяснить, какой инструментарий должен быть у корпоративного бота, следует провести опрос его потенциальных пользователей. При проектировании функционала должны быть учтены потребности работников, а также проблемы, которые могут быть решены автоматизацией. Если бот изначально разрабатывается для какой-либо конкретной платформы, то следует изучить существующие библиотеки по созданию ботов для выбранной платформы, а также рассмотреть ее возможности и ограничения.

Также при разработке бота стоит продумать перспективу проекта и дальнейшее развитие его функционала. Но для начала следует реализовать те возможности, которые необходимы в данный момент времени, и попытаться сделать так, чтобы программа была дружелюбной для внедрения новых функций.

Не следует забывать и про удобство пользования. Чтобы пользователи понимали, как чат-бот работает и какие запросы выполняет, его функционал прописывается доступным для каждого человека языком. Сообщение с подробным описанием работы бота присылается в начале сеанса или с помощью специальной команды.

Если пользователь некорректно задал вопрос, то бот должен сообщить об этом и предложить пример рабочего запроса. Также у пользователя всегда должен быть доступ к меню чат-бота.

Запросы могут формироваться различными способами. Это могут быть текстовые вопросы, структурированные особым образом, понятным для бота. Или же специальные кнопки или виджеты, которые помогают быстро выбирать варианты запросов без ввода текста.

Когда функционал проработан на бумаге, стоит переходить к его технической реализации.

2.3.1 Выбор средств разработки

Чат-боты могут разрабатываться с использованием различных языков программирования. Так, например, библиотека `python_zulip_api`, с помощью которой разрабатываются боты для мессенджера Zulip, поддерживает язык Python. У Telegram инструментов для написания ботов намного больше, например, `aiogram`, `python-telegram-bot`, `pyTelegramBotAPI` и другие. Они все также поддерживают язык программирования Python.

Такой метод разработки не создает ограничений для программиста. Можно реализовать для бота любые функции. Но главное, чтобы они вписывались в возможности выбранной платформы, на которой будет запущен чат-бот.

Помимо написания программы существует вариант создания чат-ботов с помощью специальных конструкторов. Данный метод способен сэкономить время и деньги, так как в этом случае не нужно нанимать программиста, потому что чаще всего интерфейс таких конструкторов предельно понятен. Но у этого способа есть ряд ограничений. Во-первых, большинство конструкторов платные. Во-вторых, невозможно реализовать функции, не предусмотренные конструктором. Таким образом, метод разработки ботов с помощью конструкторов подходит для небольших проектов с примитивным функционалом.

2.4 Чат-бот для Zulip

Для разработки ботов Zulip предоставляет специальную библиотеку Zulip API, которая позволяет писать программы на языке Python. С ее помощью можно написать бота, который способен обрабатывать сообщения любых чатов внутри организации в режиме реального времени. К данной библиотеке прилагается достаточно обширная документация, в которой объясняется, как следует писать программы, реализующие ботов, а также предоставляется список существующих функций и примеры их использования.

На рисунке 2.6 показан скриншот из документации Zulip API, на котором представлен пример использования функции `send_message()`, реализующей отправку сообщения.

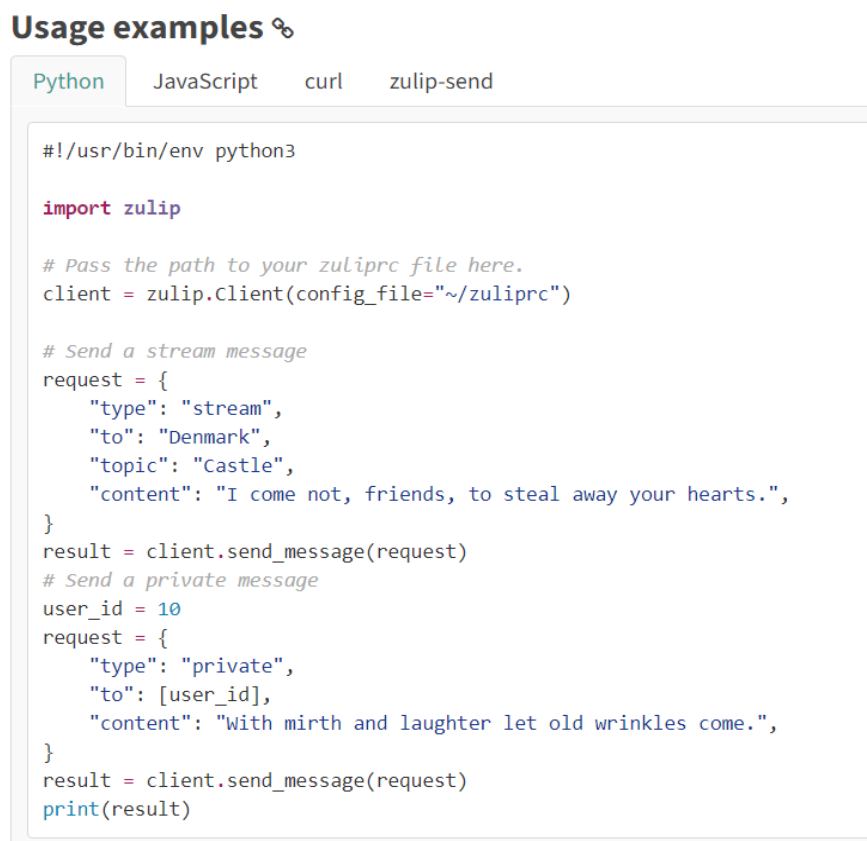


Рисунок 2.6 – Скриншот документации Zulip API

Мессенджер Zulip не предоставляет возможности использовать для взаимодействия с ботом кнопки и виджеты, которые позволяют отправлять запросы боту мгновенно, а не печатать их на клавиатуре. Таким образом, при разработке чат-бота нужно составить список ключевых слов и структуру сложных запросов, которые смогут правильно идентифицироваться программой, и для каждого запроса написать обрабатывающую их функцию. А чтобы пользователь смог ориентироваться в функционале, предоставляемым ботом, следует составить список запросов, для каждого из которых объяснить его структуру. Данный список должен выводиться в виде меню по запросу пользователя.

Чтобы начать разработку программы, нужно сначала добавить бота в организацию, созданную в Zulip. Таким образом будет сформирован специальный zuliprc файл и API-ключ чат-бота, которые впоследствии будут использованы при написании программы.

На рисунке 2.6 представлен скриншот страницы добавления бота в организацию. Здесь можно настроить тип, имя и email чат-бота, а также его изображение профиля.

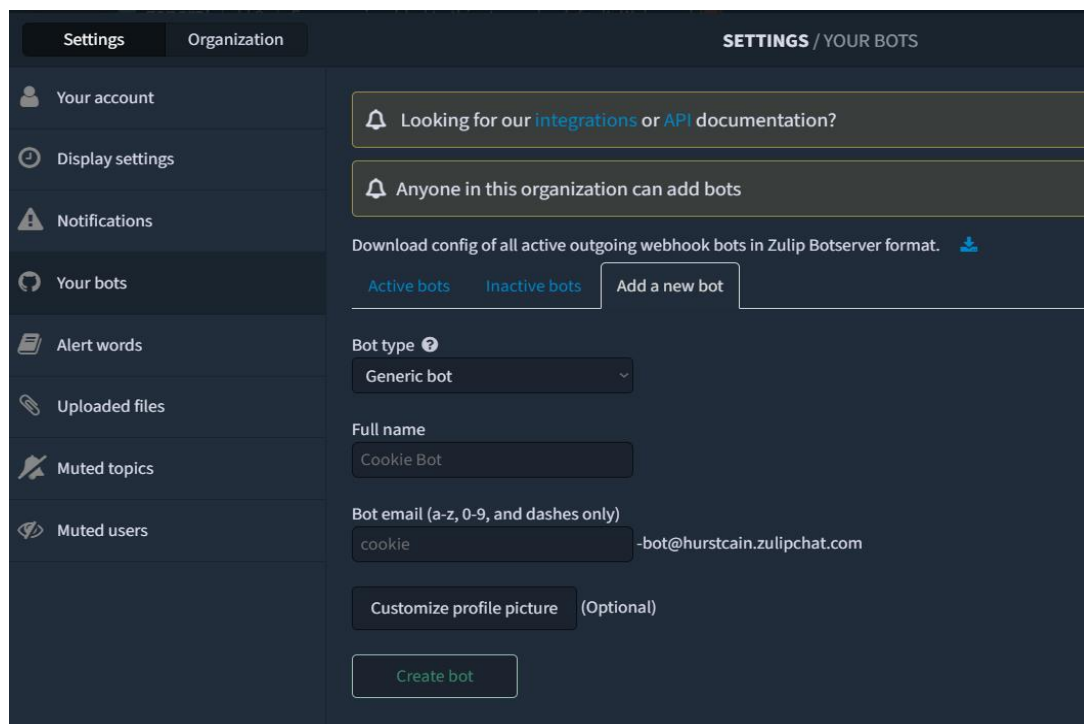


Рисунок 2.6 – Добавление бота в организацию Zulip

На рисунке 2.7 показан скриншот того, как выглядит страница чат-бота после его создания.

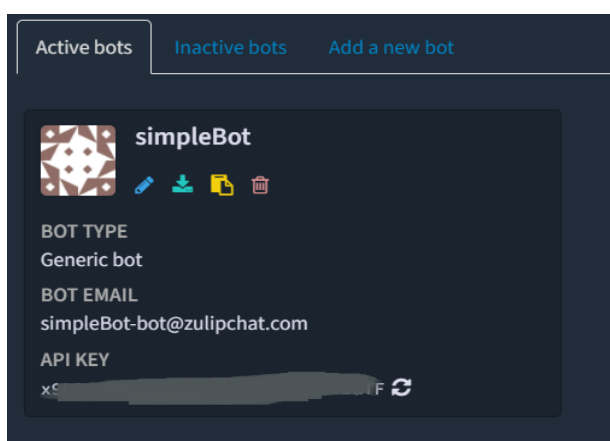


Рисунок 2.7 – Страница бота в Zulip

Файл `zulirc` содержит все подробности о конфигурациях бота. А ключ `API` – это своеобразный идентификатор чат-бота в Zulip. Текст ключа вставляется в программу, и при запуске она принимает сообщения, отправленные из чатов организации, в которой присутствует бот с данным `API`-ключом. И файл `zulirc`, и `API`-ключ чат-бота не должны находиться в открытом доступе.

Когда бот в организации создан, можно приступать к написанию реализующей его программы. Для этого нужно установить пакеты `python` (стандартная библиотека языка Python) и `zulip` (библиотека для разработки чат-ботов для Zulip). Разработка чат-бота производится в линуксовых операционных системах, например, Ubuntu.

Для хранения данных, которые программа может извлекать и преобразовывать, стоит использовать различные реляционные базы данных: MySQL, MariaDB, PostgreSQL, SQLite.

Для разработки программы на Python проще всего использовать SQLite, так как Python предоставляет специальный модуль `sqlite3` для работы с данной БД.

SQLite – это автономный, работающий без сервера транзакционный механизм базы данных SQL [9]. Созданная база данных хранится в одном файле. Файлы с БД можно редактировать не только с помощью запросов SQL, но и в специальных программах, что намного удобнее.

Одной из таких программ является SQLiteStudio. Она предназначена для создания и редактирования баз данных SQLite. На рисунке 2.8 представлен скриншот интерфейса данной программы.

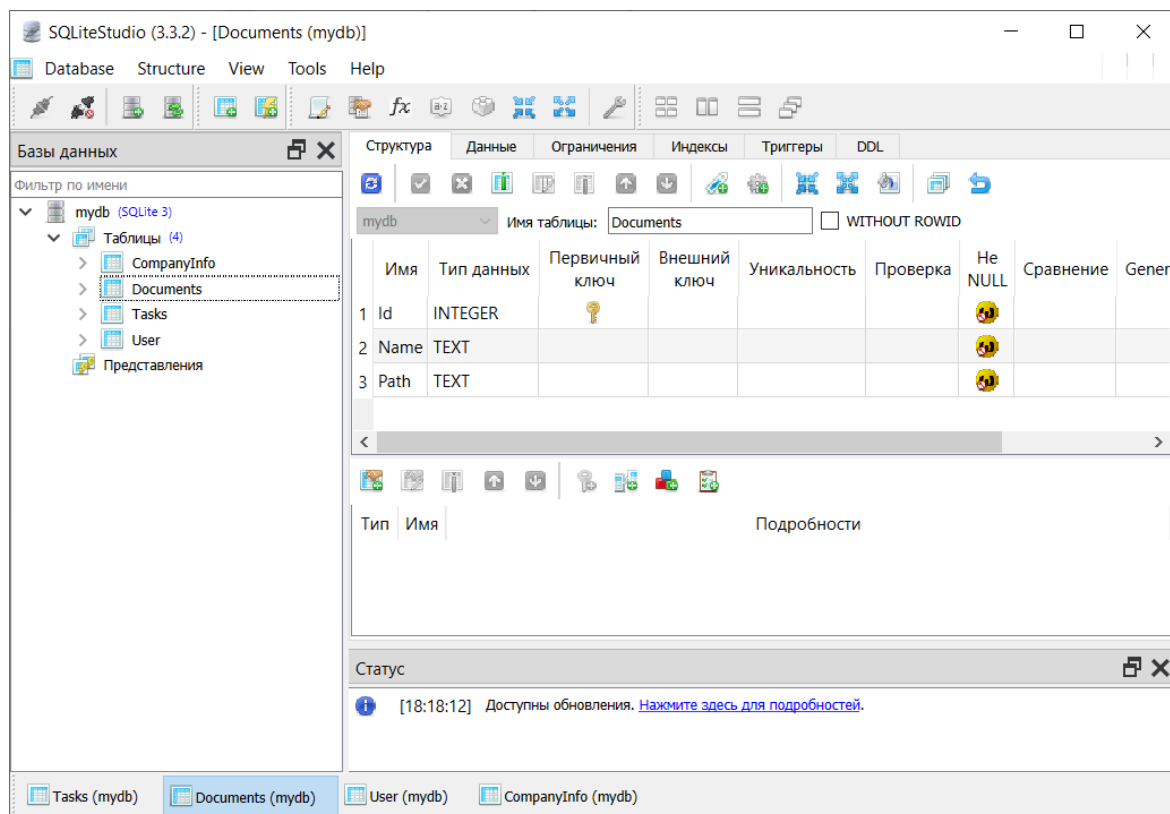


Рисунок 2.8 – Интерфейс программы SQLiteStudio

После написания программы, реализующей чат-бота, ее следует запустить на корпоративном или арендованном сервере. Пока программа работает, бот будет активен.

3 Разработка чат-бота

Так как проделанная работа не имела цели создать чат-бота для какой-либо конкретной организации, написанная программа имеет ряд функций, которые могут быть полезны для любой компании:

1. получение информации о пользователях;
2. автоматическое заполнение заявлений и корпоративных документов;
3. составление расписания и списка задач на каждый день;
4. получение информации о нерабочих днях в году.

Функционал бота всегда можно расширить, разработав для него запросы, нужные той или иной организации, и внедрив в код функции для обработки этих запросов.

3.1 Список модулей программы

Программа разрабатывалась на языке Python, который поддерживает API Zulip, библиотека для разработки чат-бота.

Перечень модулей написанной программы: bot.py, BotInfo.py, Users.py, UserInfo.py, Docs.py, Holydays.py, Schedule.py, BirthdayScript.py, ScheduleScript.py. Описание каждого из модулей, а также содержащихся в них функций, представлено в приложении В.

3.2 Структура БД чат-бота

Большая часть информации, которую использует бот для ответов на запросы, хранится в базе данных. В качестве хранилища используется SQLite. Схема базы данных представлена на рисунке 3.1.



Рисунок 3.1 – Схема БД

1. Таблица CompanyInfo хранит в себе название компании (CompanyName), фамилию (DirectorLastName), имя (DirectorFirstName), отчество (DirectorMiddleName) и пол (DirectorGender) директора организации. Данная информация предназначена для автоматического заполнения некото-

рых корпоративных документов, при оформлении которых требуется информация о названии организации и полном имени руководителя. Таблица предполагает только одну строку информации, которую при необходимости можно отредактировать.

2. В таблице Documents содержится информация о различных заявлениях, заполнение которых автоматизирует чат-бот: идентификатор документа (Id), наименование документа (Name), а также путь к нему (Path). Первичным ключом таблицы является столбец Id. Предполагается, что у каждого документа свой индивидуальный идентификатор. С помощью столбца Path программа получает информацию о местонахождении документа и извлекает его для последующего редактирования.

3. Таблица User хранит информацию о зарегистрированных пользователях: идентификатор (UserId), фамилия (Surname), имя (FirstName), отчество (MiddleName), пол (Gender), адрес электронной почты (Email), телефон (Phone), день (BirthdayDay), месяц (BirthdayMonth) и год рождения (BirthdayYear) сотрудника. В качестве первичного ключа здесь выступает столбец UserId, куда записывает адрес электронной почты сотрудника, с помощью которой тот был зарегистрирован в мессенджере.

4. В таблице Tasks хранятся предстоящие задания каждого из пользователей. Столбцы данной таблицы: идентификатор задания (TaskId), идентификатор пользователя (UserId), день (TaskDay), месяц (TaskMonth), время (TaskTime) и содержание задания (TaskText). Первичный ключ здесь – это TaskId, у каждого задания свой уникальный идентификатор. UserId, идентификатор пользователя, – это внешний ключ, который ссылается на таблицу User.

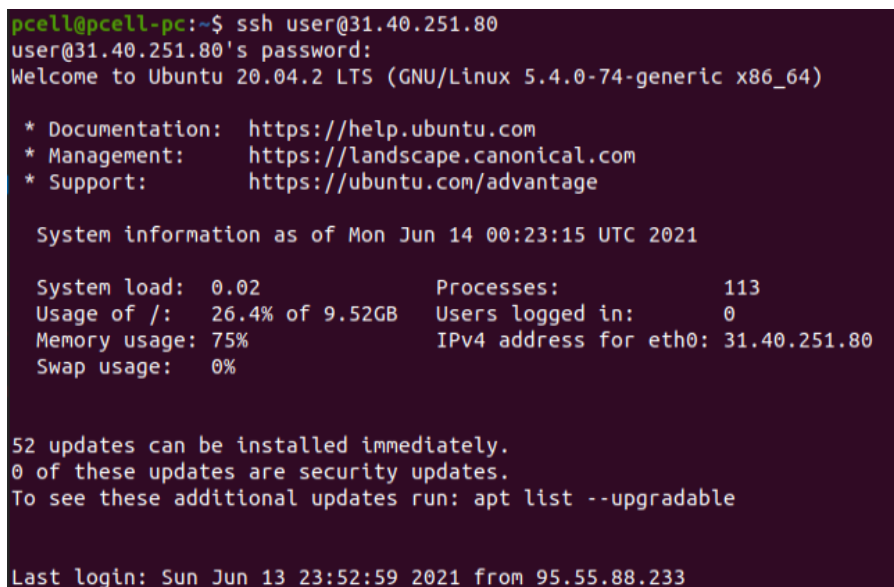
3.3 Запуск программы на сервере

Чтобы чат-бот работал перманентно, следует запустить написанную программу на арендованном сервере.

В данной работе в качестве хостинговой компании, предоставляющей аренду VPS, была выбрана Boonet Online [22]. Она предоставляет возможность самостоятельно настроить конфигурацию сервера: размер дискового пространства, ОЗУ, CPU, а также ОС.

В данной работе использовался VPS со следующими настройками: ОС Ubuntu 20.04, 1 CPU, 512 Мб ОЗУ, 10 Гб дискового пространства.

К выделенному серверу можно подключиться удаленно по SSH. Для этого необходимо знать IP-адрес сервера, логин и пароль для подключения, которые предоставляются хостинговой компанией при аренде сервера. Работа с данными на сервере происходит через Linux консоль. На рисунке 3.2 представлен скриншот процесса подключения к серверу по SSH.



```
pcell@pcell-pc:~$ ssh user@31.40.251.80
user@31.40.251.80's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-74-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

System information as of Mon Jun 14 00:23:15 UTC 2021

System load:  0.02               Processes:            113
Usage of /:   26.4% of 9.52GB    Users logged in:     0
Memory usage: 75%               IPv4 address for eth0: 31.40.251.80
Swap usage:   0%

52 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Last login: Sun Jun 13 23:52:59 2021 from 95.55.88.233
```

Рисунок 3.2 – Подключение к арендованному серверу по SSH

В Linux предоставляется возможность копировать файлы по SSH с локальной машины на виртуальный сервер и наоборот. Именно с помощью данной функции папка с проектом была передана на удаленный сервер.

Когда все файлы размещены на сервере, остается только запустить нужный исполняемый файл в фоновом режиме. Это делается с помощью специальной команды. На рисунке 3.3 представлен пример запуска файла bot.py в фоновом режиме. Для того, чтобы чат-бот работал корректно, нужно запустить

три исполняемых файла: bot.py, BirthdayScript.py и ScheduleScript.py. На рисунке 3.4 можно видеть, что эти три файла успешно выполняются в фоновом режиме.

```
user@16758:~/simpleBot$ python3 bot.py &
[1] 2130
```

Рисунок 3.3 – Запуск исполняемого файла bot.py в фоновом режиме

```
user      888  0.0  0.0  10528  5152  ?        S    Jun13   0:00 python3 bot.py
root     1253  0.0  0.0      0      0  ?        I    Jun13   0:00 [kworker/0:2-events]
user     1667  0.0  5.4  36416  21516  ?        S    00:26   0:00 python3 ScheduleScript.py
user     1793  0.0  5.5  36288  22024  ?        S    00:30   0:00 python3 BirthdayScript.py
```

Рисунок 3.4 – Перечень запущенных процессов на сервере

После запуска исполняемых файлов можно спокойно отключаться от сервера, так как процессы продолжают свою работу, и, следовательно, чат-бот будет непрерывно функционировать.

4 Описание функционала чат-бота

Разработанный чат-бот реализует следующие функции: регистрация пользователей, получение информации о сотрудниках, автоматическое заполнение документов, составление расписания и добавление списка задач на каждый день, получение информации о нерабочих днях, ежедневное отправление напоминаний о текущем расписании, поздравление сотрудников с днем рождения. Ниже подробно описаны каждую из функций.

4.1 Регистрация пользователей

Чтобы воспользоваться ботом, нужно быть зарегистрированным в его базе данных. Регистрацию можно провести самостоятельно прямо в мессенджере Zulip, если нет возможности заполнять базу данных сотрудников с помощью системного администратора.

Регистрация в базе данных необходима для того, чтобы автоматизировать заполнение ряда корпоративных документов, ведь тогда не придется самому вводить данные, а также чтобы другие сотрудники могли иметь доступ

к некоторой части информации о своих коллегах: к полному имени, адресу электронной почты, номеру телефона и дате рождения.

При проверке сотрудника на присутствие в базе данных, программа сверяет его адрес электронной почты, с помощью которого пользователь регистрировался в мессенджере, с адресами, присутствующими в базе. Если пользователь не обнаружен, то, соответственно, чат-бот просит сотрудника пройти регистрацию.

На рисунке 4.1 представлен скриншот с приветственным сообщением от чат-бота в том случае, если информации о пользователе в базе данных не обнаружено.

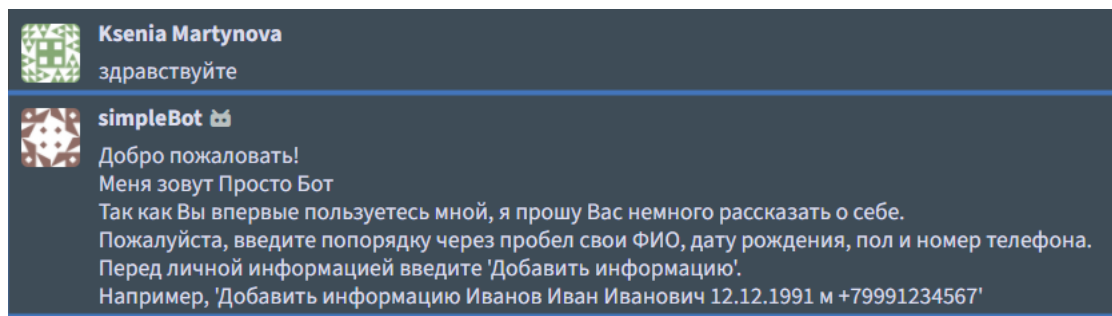


Рисунок 4.1 – Чат-бот просит пройти небольшую регистрацию, если пользователь не найден в БД

Если пользователь ввел данные верно, как указано в шаблоне, то программа добавляет введенную информацию в БД, и бот выводит сообщение об успешно выполненной операции, что показано на рисунке 4.2.

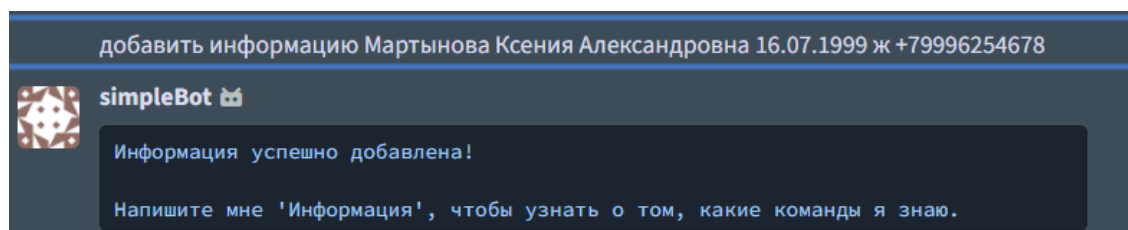


Рисунок 4.2 – Сообщение об успешной регистрации пользователя

Когда сотрудник зарегистрировался в системе, ему становятся доступны любые функции чат-бота.

Чтобы посмотреть, какие запросы умеет выполнять чат-бот, пользователь должен отправить сообщение с текстом «Информация». В ответ пользователь пришлет сообщение с содержанием, представленном на рисунке 4.3.

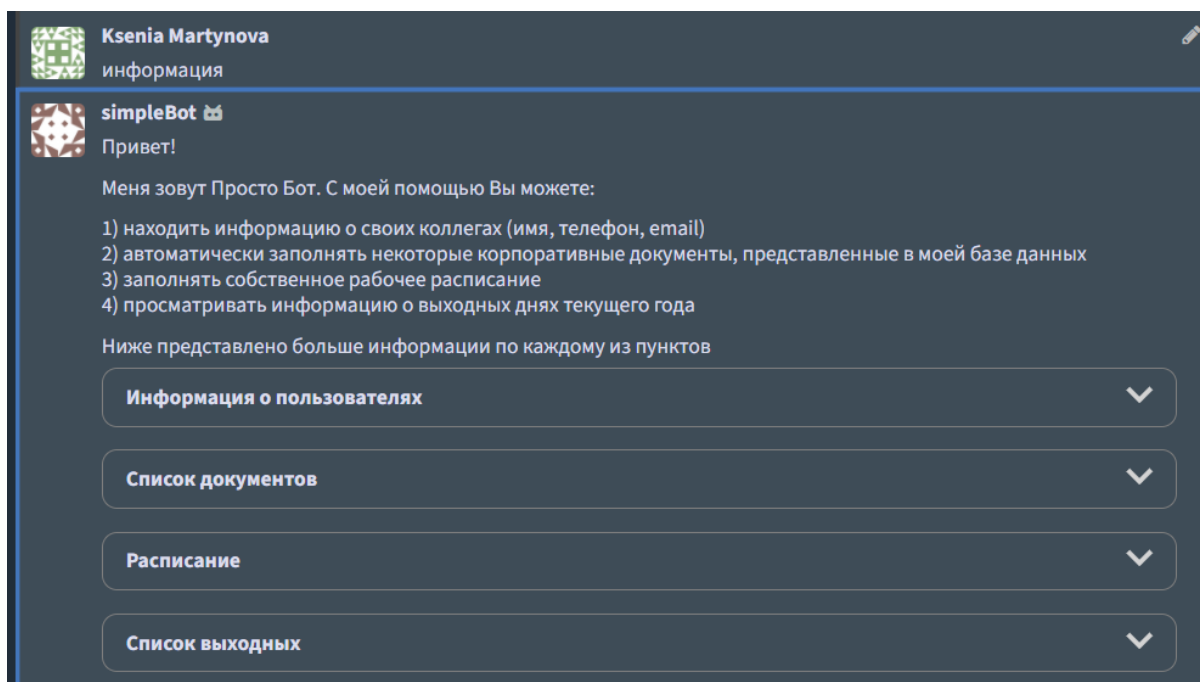


Рисунок 4.3 – Меню чат-бота

4.2 Получение информации о пользователях

Одной из функций чат-бота является получение информации о коллегах по работе, данные о которых присутствуют в БД. Инструкция по использованию данной функции представлена на рисунке 4.4.

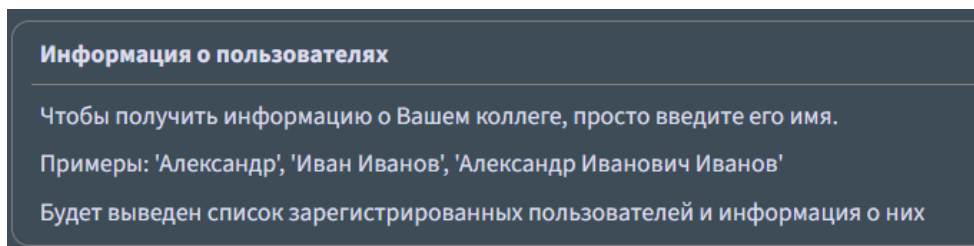


Рисунок 4.4 – Меню чат-бота, раздел “Информация о пользователях”

Данная функция работает следующим образом. Пользователь отправляет чат-боту сообщение только с именем своего коллеги без дополнительных слов, можно указать полные ФИО либо просто одну фамилию или же имя. А программа ищет в одной из таблиц базы данных людей с соответствующим

именем. Если находится один или несколько человек, то чат-бот отправляет сообщение со списком найденных пользователей. В ином случае чат-бот отправляет пользователю сообщение о том, что он не понял запрос.

В качестве информации предоставляется полное имя работника, день и месяц рождения, телефон и адрес электронной почты.

На рисунке 4.5 представлен скриншот с примером поиска работника только по имени. Программа обнаружила в базе данных двух человек с именем Ксения и вывела о них информацию. А на рисунке 4.6 в качестве примера предоставляется поиск по имени и отчеству. В этом случае программа нашла только одного сотрудника с соответствующим именем.

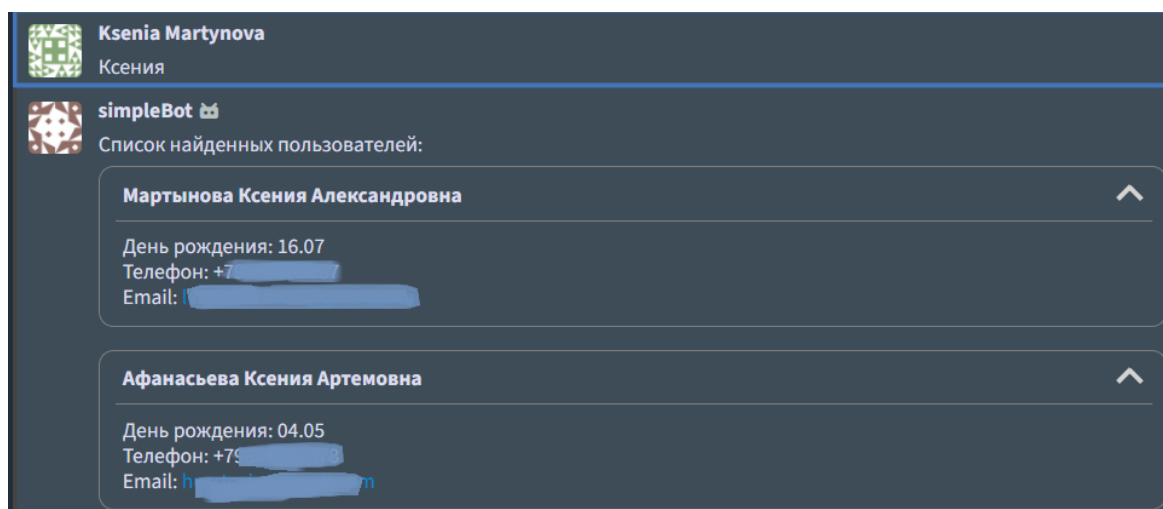


Рисунок 4.5 – Поиск пользователя по имени

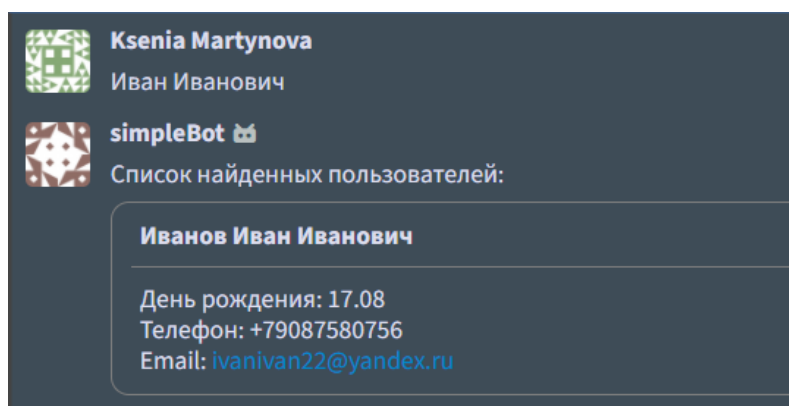


Рисунок 4.6 – Поиск пользователя по имени и отчеству

4.3 Автоматическое заполнение документов

Данная функция чат-бота автоматизирует заполнение некоторых документов, которые присутствуют в базе данных.

Для реализации этой функции использовались сторонние библиотеки для языка Python: `docxtpl` [10] и `Petrovich` [11].

Библиотека `docxtpl` позволяет автоматизировать заполнение документов с помощью создания шаблона для конкретного корпоративного документа. В местах, куда нужно вставить информацию, прописываются теги (по-другому, названия переменных, которые будут заменены на новые значения), обрамленные специальными символами. Пример такого шаблона представлен в приложении А. В самой же программе пишется код, в котором с помощью словарей (тип данных в Python) для каждого тега из шаблона указывается его значение. С помощью встроенных в библиотеку функций создается новый документ, где теги заменяются на их значения, прописанные в программе. Пример одной из функций обработки шаблона представлен в приложении Б.

С помощью библиотеки `Petrovich` можно склонять имена, отчества и фамилии по падежам. Это нужно в первую очередь для корректного заполнения документов, так как в заявлениях присутствуют места, где нужно вписывать полное имя в родительном, дательном или любом другом падеже. В базе данных имена хранятся в именительном падеже, и было бы ресурсозатратно вручную склонять имя, фамилию и отчество каждого сотрудника по падежам, а затем решать, где лучше всего хранить эти данные. Но, используя библиотеку `Petrovich`, есть возможность проделать эту работу автоматически. Единственное, чтобы корректно просклонять имя сотрудника, нужно обязательно знать его пол. Именно для этого данная информация запрашивается у пользователя при регистрации.

На рисунке 4.7 можно видеть пример использования функций из библиотеки `Petrovich`. Для склонения имен существуют три функции: `lastname()` –

склоняет фамилию; `firstname()` – склоняет имя; `middlename()` – склоняет отчество. В качестве аргументов передаются имя (фамилия или же отчество) в именительном падеже, падеж для склонения и пол.

```
# Создание экземпляра класса
p = Petrovich()

# Преобразование ФИО сотрудника в родительный падеж
# Если пол сотрудника мужской
if gender == 'male':
    # Склонение фамилии
    lastNameDative = p.lastname(lastName, Case.GENITIVE, Gender.MALE)
    # Склонение имени
    firstNameDative = p.firstname(firstName, Case.GENITIVE, Gender.MALE)
    # Склонение отчества
    middleNameDative = p.middlename(middleName, Case.GENITIVE, Gender.MALE)
# Если пол сотрудника женский
elif gender == 'female':
    lastNameDative = p.lastname(lastName, Case.GENITIVE, Gender.FEMALE)
    firstNameDative = p.firstname(firstName, Case.GENITIVE, Gender.FEMALE)
    middleNameDative = p.middlename(middleName, Case.GENITIVE, Gender.FEMALE)
```

Рисунок 4.7 – Использование функций из библиотеки `Petrovich` для склонения имен по падежам

Чтобы заполнить документы, необходимы некоторые статические данные: название организации и ФИО директора. Они берутся из таблицы `CompanyInfo` базы данных бота. Информация о работнике, от имени которого заполняется заявление, также берется из БД чат-бота. В запросе, который отправляет пользователь, свое имя вписывать не нужно, так как программа автоматически определяет отправителя, сохраняет в переменную `e-mail` сотрудника, отправившего запрос, и по адресу электронной почты в базе данных находит его полное имя. Дата заполнения документа также определяется автоматически с помощью встроенной в Python библиотеки `datetime`.

В запросе чат-боту работник указывает только изменяемые данные, которые необходимо прописать в заявлении. Например, для заявления на отпуск следует указать даты начала и конца отпуска, а также количество отпускных дней. А для заявления об увольнении нужна только дата увольнения.

После отправки запроса пользователем, программа формирует файл с заполненным заявлением. Созданный файл сохраняется в определенную папку

на сервере. Название файла соответствует электронной почте сотрудника, запросившего создание файла. Каждый раз файл с одним и тем же именем перезаписывается. Таким образом, не накапливается большое количество уже ненужных документов.

На рисунке 4.8 представлен список документов, которые присутствуют в базе данных чат-бота. Чтобы получить список заявлений, доступных для автоматического заполнения, нужно отправить боту сообщение с содержанием: “информация” или “документы”.

Каждый документ имеет свой номер. На данный момент автоматически можно заполнить только 9 документов. Идентификатор документу нужен для удобства пользователя. Для заполнения документа в запросе нужно прописать либо полное название заявления, либо его номер. Некоторые документы имеют длинные названия, и, чтобы пользователю не пришлось прописывать в запросе огромное количество слов, сотрудник может просто указать номер нужного ему заявления, что гораздо быстрее.

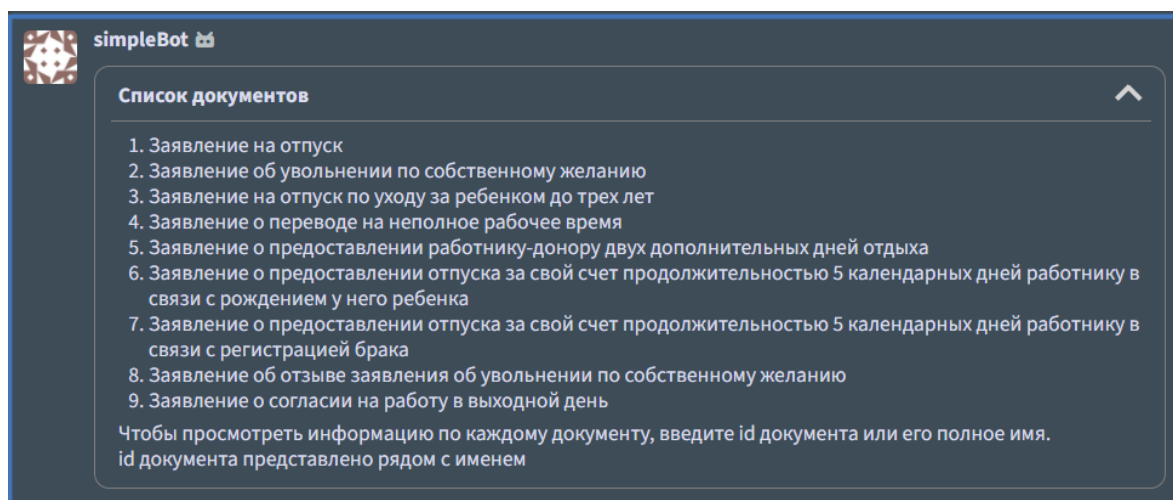


Рисунок 4.8 – Раздел меню чат-бота, в котором прописывается список документов, которые можно заполнить автоматически

Чтобы узнать, какие данные необходимо указать для заполнения заявления, нужно отправить чат-боту сообщение с номером или названием документа. На рисунке 4.9 можно видеть, что в ответ бот присылает инструкцию и пример оформления запроса.

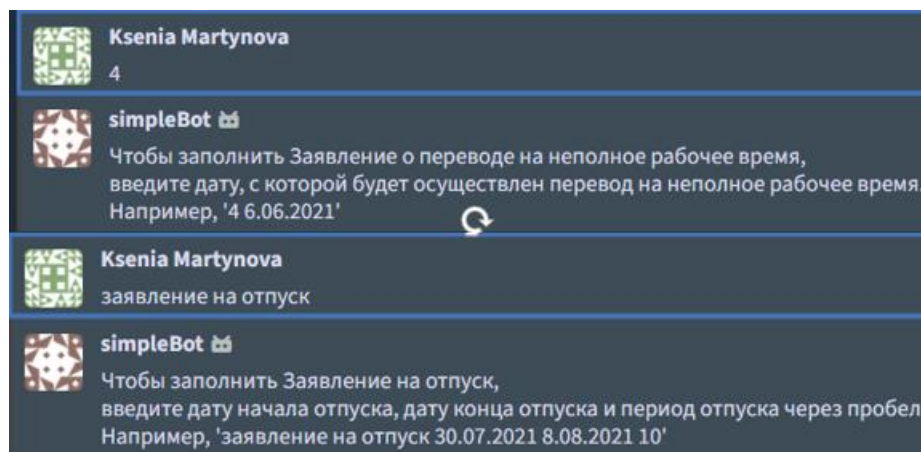


Рисунок 4.9 – Пример инструкций к оформлению запроса на заполнение заявления

На рисунке 4.10 представлен пример запроса на автоматическое заполнение заявления на отпуск. В ответ на запрос чат-бот присылает ссылку на скачивание файла с заполненным заявлением.

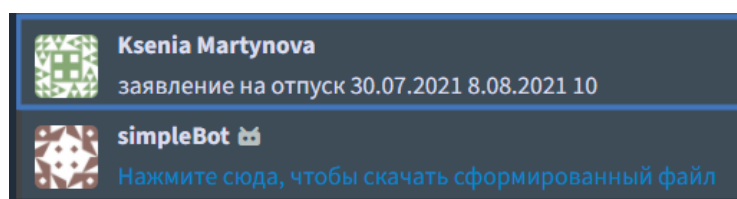


Рисунок 4.10 – Запрос на заполнение заявления на отпуск

Образец заполненного заявления представлен в приложении В.

4.4 Составление расписания и списка задач на каждый день

Чат-бот позволяет сотрудникам составлять список задач на каждый день. Пользователь может добавлять задачи и просматривать свое расписание на разные дни. Инструкция по использованию данной функции представлена на рисунке 4.11.

Добавленные сотрудником задачи помещаются в таблицу Tasks базы данных чат-бота. Просроченные задачи удаляются из базы данных автоматически.

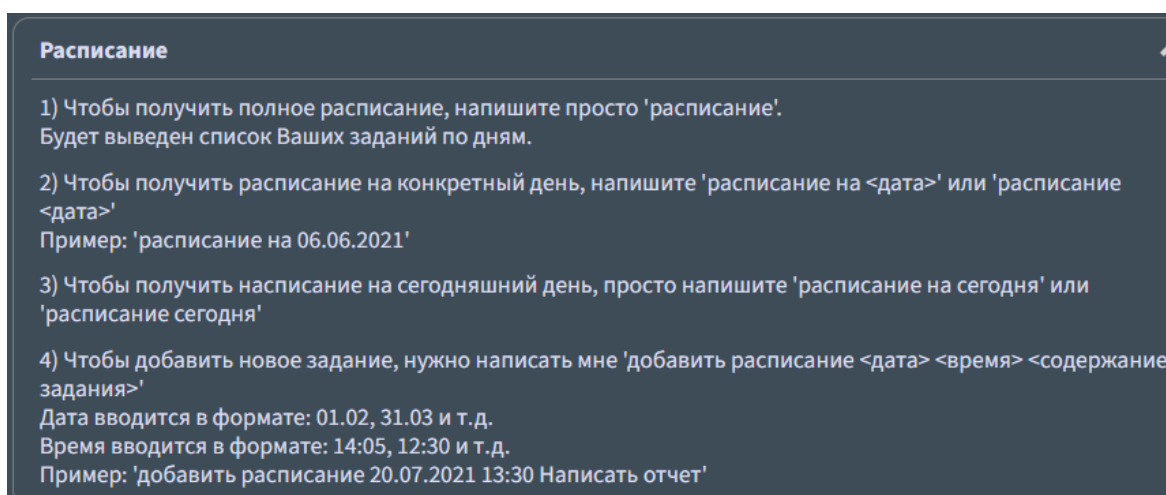


Рисунок 4.11 – Меню чат-бота, инструкция к разделу “Расписание”

Чтобы добавить задание, нужно написать чат-боту сообщение со следующим содержанием: “добавить расписание <дата> <время> <содержание задания>”. Дата и время должны вводиться в определенном формате, иначе чат-бот сообщит об ошибочной формулировке запроса.

На рисунке 4.12 представлен скриншот с запросом на добавление задания. Если запрос составлен верно, и задание добавлено в базу данных, то чат-бот отправляет соответствующее сообщение.

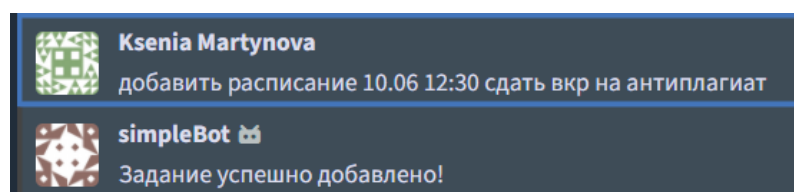


Рисунок 4.12 – Пример добавления задания

Чтобы просмотреть все предстоящие задачи, нужно написать боту “расписание”. Тогда в ответ будет отправлено сообщение с расписанием на все дни. Чтобы сообщение было удобно просматривать, расписание поделено на дни, пользователь может найти интересующую его дату и просмотреть скрытый текст. На рисунке 4.13 представлен пример того, как выглядит сообщение от чат-бота с расписанием на все дни.

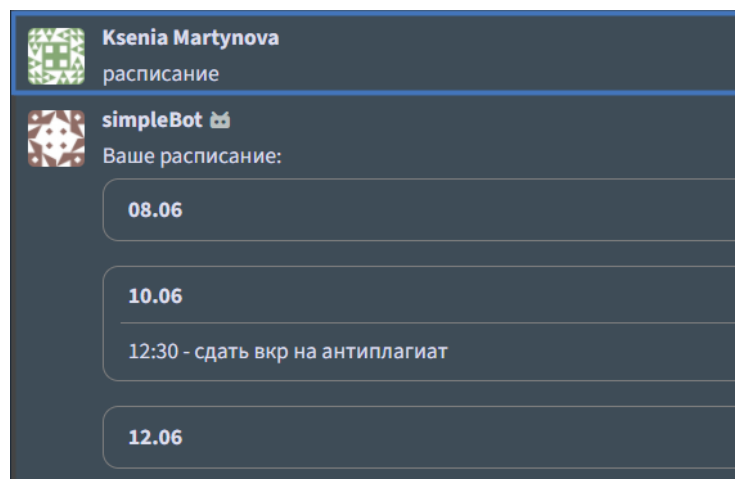


Рисунок 4.13 – Сообщение от чат-бота с расписанием пользователя
на все дни

Чтобы просмотреть расписание только на текущий день, нужно написать боту “расписание на сегодня” или “расписание сегодня”. На рисунке 4.14 представлен скриншот с запросом от пользователя вывести расписание на текущий день.

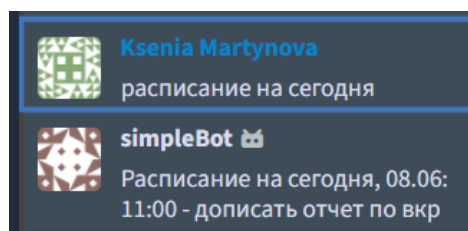


Рисунок 4.14 – Сообщение от чат-бота с расписанием на текущий день

Также чат-бот может выводить расписание на конкретный день. Для этого следует составить запрос таким образом: “расписание <дата>” или “расписание на <дата>”. Тогда будет выведено сообщение со списком задач на введенную пользователем дату. Пример такого запроса представлен на рисунке 4.15.

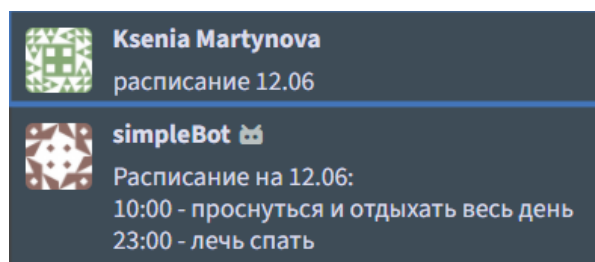


Рисунок 4.15 – Сообщение от чат-бота с расписанием на конкретный день

Если сотрудник не вводил задачи, то, при попытке запросить список с расписанием, чат-бот отправит пользователю сообщение о том, что расписание отсутствует. Также с расписанием на конкретную или текущую дату. Пример подобного сообщения представлен на рисунке 4.16.

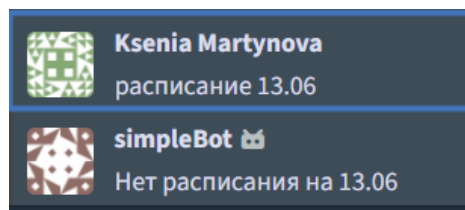


Рисунок 4.16 – Сообщение от чат-бота, если у пользователя нет задач

Код модуля, реализующего предоставление пользователю расписания, представлен в приложении Ж.

4.5 Получение информации о нерабочих днях

Чат-бот предоставляет пользователю возможность получить список нерабочих дней в году. Инструкция к составлению запроса представлена на рисунке 4.17.

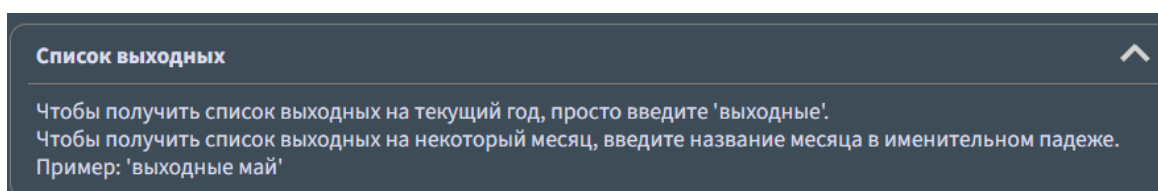


Рисунок 4.17 – Меню чат-бота, инструкция к составлению запроса на предоставление списка нерабочих дней

Для получения информации о выходных днях используется парсинг XML-файла, распечатка которого представлена в приложении Д. Парсинг осуществляется с помощью функций программного модуля `xml.etree.ElementTree` [23], который реализует API для анализа и создания XML-данных. В приложении Е представлен пример с кодом, в котором происходит парсинг XML-документа.

На рисунке 4.18 представлен выведенный чат-ботом список выходных на текущий год.

Пользователь может запросить список выходных дней как на год, так и на конкретный месяц. Например, на рисунке 4.19 представлен ответ бота на запрос пользователя вывести список нерабочих дней на июнь.

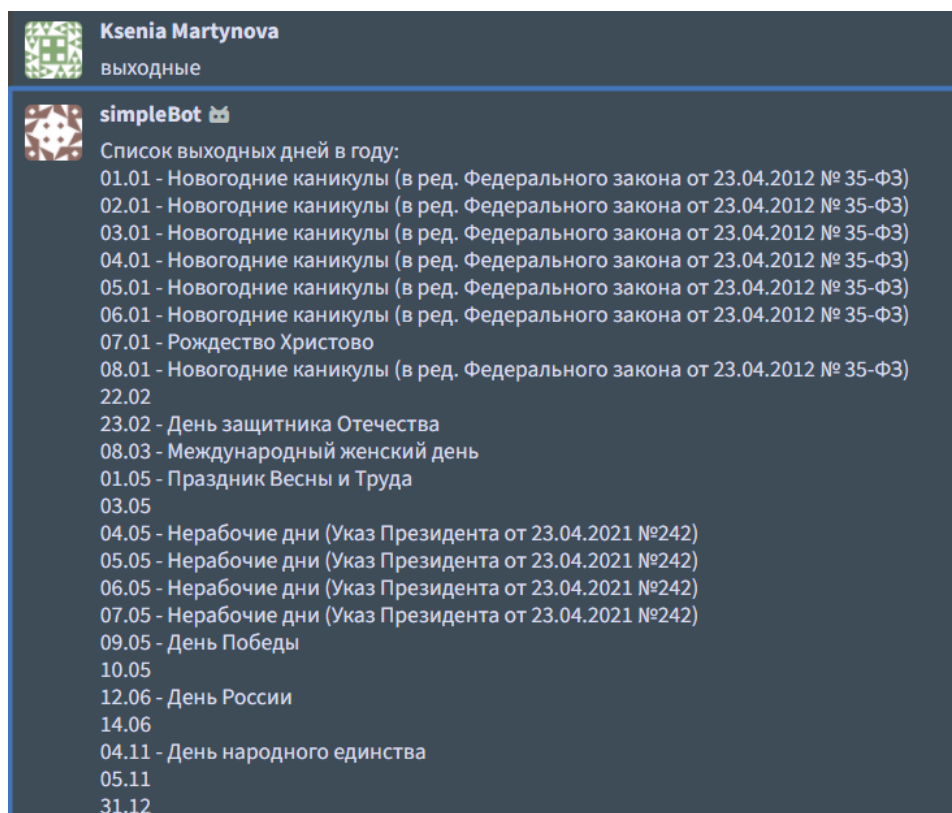


Рисунок 4.18 – Ответ чат-бота на запрос вывести все нерабочие дни в текущем году

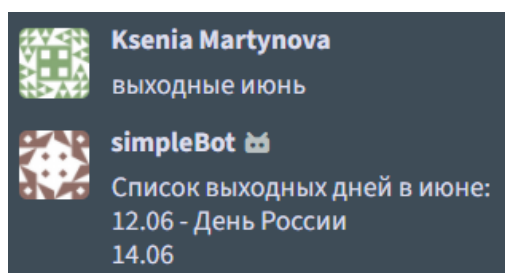


Рисунок 4.19 – Ответ чат-бота на запрос о предоставлении списка нерабочих дней в июне

4.6 Оповещение сотрудников

Чат-бот настроен на то, чтобы отправлять пользователям уведомления: о задачах на текущий день, если они есть, и о поздравлении с днем рождения.

Также, если о одного из сотрудников день рождения, бот присылает остальным пользователем напоминание о том, что их коллегу стоит поздравить.

Оповещение реализовано двумя модулями: BirthdayScript.py и ScheduleScript.py, которые запускаются каждый день в одно и то же время.

Сообщение с уведомлением о наличии заданий на текущий день аналогично ответу чат-бота на запрос о выводе расписания на текущий день, пример которого представлен на рисунке 4.14.

Пример сообщения с поздравлением представлен на рисунке 4.20.

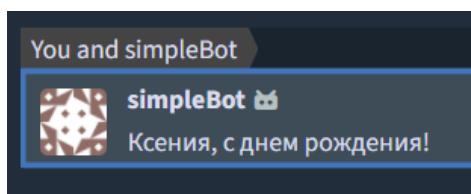


Рисунок 4.20 – Поздравление пользователя с днем рождения

4.7 Обработка ошибок

Так как чат-бот в Zulip может выполнять только текстовые запросы, очень важно, чтобы пользователь вводил их верно и без ошибок. Тогда программа будет работать корректно. А если сотрудник все же ошибся, то необходимо вывести сообщение о неправильно сформулированном запросе или некорректно введенных данных, а также предотвратить экстренное завершение программы. Для этого следует прописать в коде программы обработку исключений.

Чат-бот, разработанный в рамках данной выпускной квалификационной работы, проверяет каждый запрос, введенный пользователем, на правильность формулировки и представленных в запросе данных.

На рисунке 4.21 можно видеть несколько примеров, когда пользователь при регистрации вводит запрос или данные в запросе некорректно. В таких случаях чат-бот выводит сообщение об ошибке и просит сотрудника исправить запрос.

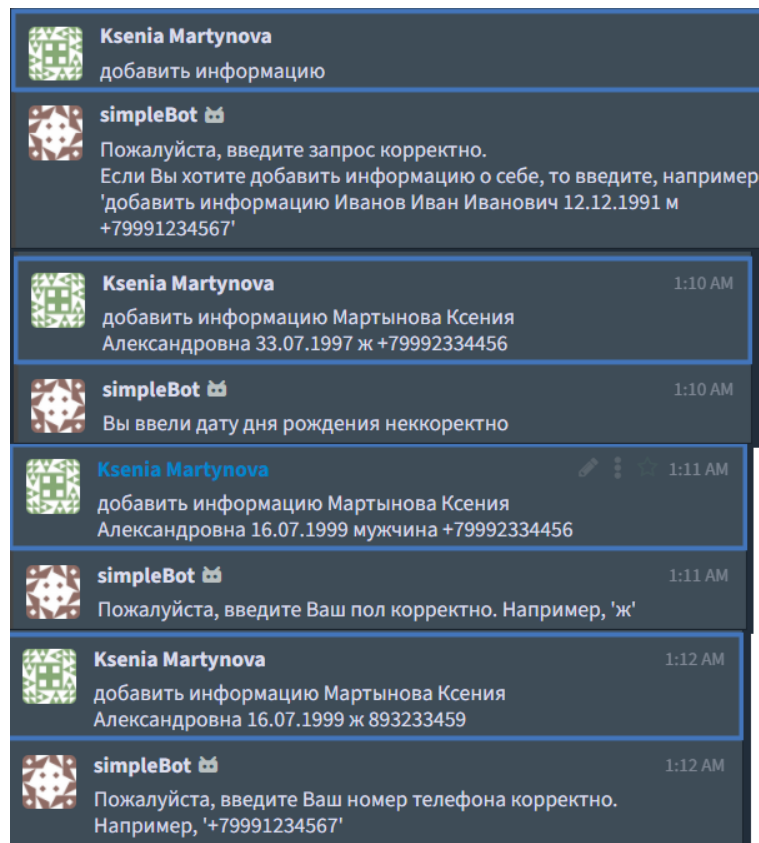


Рисунок 4.21– Обработка ботом некорректных запросов пользователя

Обработка вводимых данных присутствует в запросах о добавлении и предоставлении расписания на конкретный день.

При добавлении расписания важно составить вопрос так, как указано в шаблоне, а также придерживаться определенного формата даты и времени. Если запрос о добавлении задания не содержит в себе полных данных, то бот отправляет сообщение о том, что пользователю следует добавить в запрос недостающую информацию, что показано на рисунке 4.22. Если пользователь ввел в запросе некорректно дату задания, то бот также отправляет пользователю сообщение с предупреждением. Пример с таким сообщением показан на рисунке 4.23.

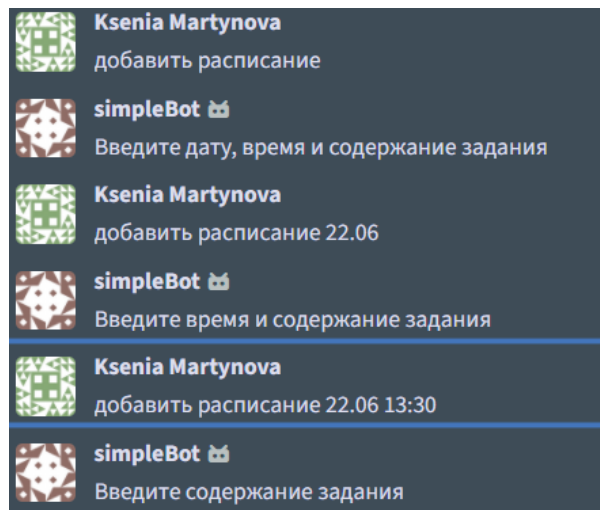


Рисунок 4.23 – Сообщения от чат-бота, предупреждающие пользователя о неверном формировании запроса на добавление расписания

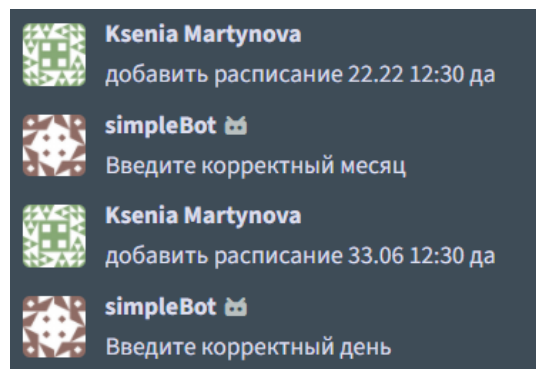


Рисунок 4.23 – Сообщения от чат-бота, предупреждающие пользователя о некорректном вводе даты выполнения задания

Если пользователь в запросе о выводе расписания на конкретный день ввел некорректную дату, то бот выводит сообщение об ошибке. Пример такого сообщения представлен на рисунке 4.24.

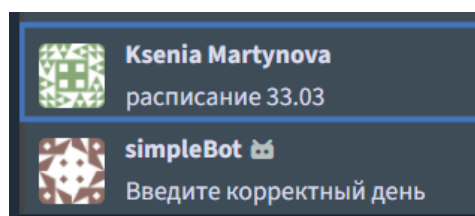


Рисунок 4.24 – Сообщения от чат-бота об ошибке ввода данных при запросе пользователя вывести расписание на конкретный день

При обработке запроса на вывод списка нерабочих дней в конкретном месяце, программа проверяет, правильно ли пользователь ввел название месяца. Если неправильно, то бот отправляет сообщение об ошибке. Пример такого сообщения представлен на рисунке 4.24.

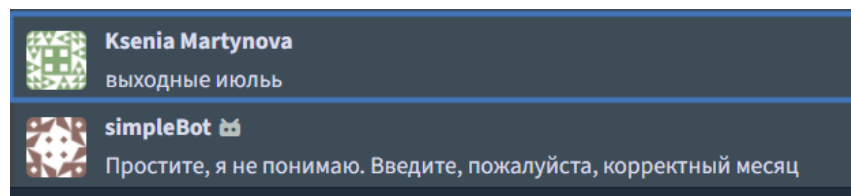


Рисунок 4.24 – Сообщения о неверно введенном названии месяца

Если пользователь отправит сообщение, которое чат-бот не может обрабатывать, то в ответ бот посоветует пользователю ознакомиться со своим функционалом, что видно на рисунке 4.25.

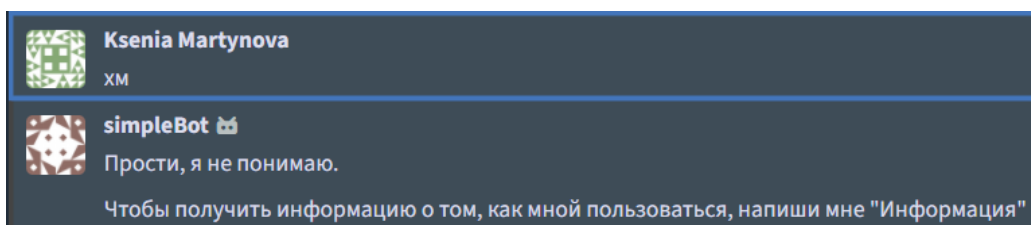


Рисунок 4.25 – Ответ чат-бота на сообщение, которое он не может обработать

5 Экономическое обоснование

Целью данной выпускной квалификационной работы является написание чат-бота для корпоративного мессенджера ZulipChat.

В текущем разделе, следуя алгоритму, представленному в методических рекомендациях [12], осуществляется анализ и расчет затрат на реализацию проекта, а также оценивается его экономическая выгода.

В рамках проведения экономического обоснования необходимо выполнить следующие действия:

- составить план-график выполнения работ, а также определить длительность выполнения работ;
- рассчитать величину заработной платы и социальных отчислений участников разработки;

- определить затраты, связанные с приобретением сырья и материалов;
- оценить затраты, необходимые для содержания и эксплуатации оборудования, используемого при написании программы;
- рассчитать величину амортизационных отчислений;
- оценить накладные расходы;
- рассчитать совокупную величину затрат, связанную с написанием программы.

5.1 Длительность и трудоемкость этапов разработки

Расчет полных затрат на разработку программы начинается с подготовки детализированного плана работ, выполняемых на каждом этапе проектирования.

Под проектированием в данном случае подразумевается совокупность работ, которые необходимо выполнить для решения поставленных в выпускной квалификационной работе задач.

Для расчета затрат на этапе проектирования следует определить продолжительность каждой из работ. В данном случае продолжительность работ определяется по факту, то есть рассматривается время выполнения, которое непосредственно было затрачено на выполнение каждого из этапов написания программы. Продолжительность работ определяется либо в человеко-днях, либо в человеко-часах.

Продолжительность и трудоемкость этапов работы представлена в таблице 5.1.

Таблица 5.1 – Продолжительность и трудоемкость этапов разработки

№ этапа	Наименование этапа	Исполнитель	Трудоемкость исполнителя (чел./час)
1	Разработка и выдача технического задания	Руководитель	10
2	Получение и изучение задания	Студент	5
3	Обзор технической документации	Студент	10
4	Установка и настройка средств разработки чат-бота	Студент	15
5	Обзор вариантов и идей для реализации функционала чат-бота	Студент	10
6	Разработка функционала чат-бота	Студент	20
7	Написание программы	Студент	70
8	Тестирование и отладка написанной программы	Студент	30
9	Написание пояснительной записки	Студент	40
10	Оформление пояснительной записки	Студент	10

5.2 Расчет размера заработной платы

Для каждого исполнителя нужно определить ставку заработной платы за единицу времени, то есть за час, которая рассчитывается исходя из месячной заработной платы конкретного исполнителя. Для подсчета дневной ставки заработной платы следует разделить месячный оклад на количество рабочих часов в месяце. Обычно рабочих дней в месяце – 21, а рабочих часов в одном дне – 8. Тогда рабочих часов в месяце: $21 \cdot 8 = 168$ часов.

В качестве месячной заработной платы студента принимается заработная плата инженера. В Санкт-Петербурге средняя зарплата инженера в месяц на 2021 год составляет 54000 рублей [13], а заработная плата руководителя разработки – 80000 рублей [14]. На основе этих данных определяются дневные ставки исполнителя и руководителя:

$$C_{\text{руководителя}} = \frac{80000}{168} = 476 \text{ руб./час}$$

$$C_{\text{исполнителя}} = \frac{54000}{168} = 321 \text{ руб./час}$$

Расчеты сведены в таблицу 5.2.

Таблица 5.2 – Затраты на основную заработную плату

№ этапа	Наименование этапа	Исполнитель	Трудоемкость (чел./час)	Ставка (руб./час)
1	Разработка и выдача технического задания	Руководитель	10	476
2	Получение и изучение задания	Студент	5	321
3	Обзор технической документации	Студент	20	321
4	Установка и настройка средств разработки чат-бота	Студент	25	321
5	Обзор вариантов и идей для реализации функционала чат-бота	Студент	15	321
6	Разработка функционала чат-бота	Студент	20	321
7	Написание программы	Студент	80	321
8	Тестирование и отладка написанной программы	Студент	40	321
9	Написание пояснительной записки	Студент	55	321
10	Оформление пояснительной записки	Студент	10	321

Расходы на основную заработную плату исполнителей рассчитываются по формуле:

$$З_{\text{осн.з./пл}} = \sum_{i=1}^k T_i \cdot C_i, \quad (1)$$

где $З_{\text{осн.з./пл}}$ – расходы на основную заработную плату исполнителей, измеряемые в рублях;

k – количество исполнителей;

T_i – время, затраченное i -м исполнителем на проведение исследования, измеряется в часах;

C_i – ставка i -го исполнителя, руб./час.

Расходы на основную заработную плату исполнителей составляют:

$$З_{\text{осн.з./пл}} = 10 \cdot 476 + 270 \cdot 321 = 91430 \text{ руб.}$$

Затраты на дополнительную заработную плату определяются по формуле:

$$З_{\text{доп.з/пл}} = З_{\text{осн.з/пл}} \cdot \frac{Н_{\text{доп}}}{100}, \quad (2)$$

где $З_{\text{доп.з/пл}}$ – расходы на дополнительную заработную плату исполнителей, руб.;

$З_{\text{осн.з/пл}}$ – расходы на основную заработную плату исполнителей, руб.;

$Н_{\text{доп}}$ – норматив дополнительной заработной платы, которые равняется 14 %.

Расходы на дополнительную заработную плату исполнителей составляют:

$$З_{\text{доп.з/пл}} = 91430 \cdot \frac{14}{100} = 12800 \text{ руб.}$$

5.3 Расчет обязательных социальных отчислений

Отчисления на страховые взносы, в которые входят расходы на обязательное пенсионное, социальное и медицинское страхование, с основной и дополнительной заработной платы исполнителей рассчитываются по формуле:

$$З_{\text{соц}} = (З_{\text{осн.з/пл}} + З_{\text{доп.з/пл}}) \cdot \frac{Н_{\text{соц}}}{100}, \quad (3)$$

где $З_{\text{соц}}$ – отчисления на социальные нужды с заработной платы исполнителей, руб.;

$З_{\text{осн.з/пл}}$ – расходы на основную заработную плату исполнителей, руб.;

$З_{\text{доп.з/пл}}$ – расходы на дополнительную заработную плату исполнителей, руб.;

$Н_{\text{соц}}$ – норматив отчислений на страховые взносы: на обязательное социальное, медицинское, и пенсионное страхование. На 2021 год составляет 30% [15].

Обязательные социальные отчисления составляют:

$$З_{\text{соц}} = (91430 + 12800) \cdot \frac{30}{100} = 31269 \text{ руб.}$$

5.4 Расчет затрат на материалы и покупные изделия

Стоимость покупных комплектующих изделий рассчитывается по формуле:

$$З_{\text{п}} = \sum_{l=1}^L N_l \text{Ц}_l \left(1 + \frac{Н_{\text{т.з.}}}{100}\right), \quad (4)$$

Где $З_{\text{п}}$ – затраты на покупные комплектующие изделия, руб.;

N_l – количество l -ых комплектующих изделий, которые входят в единицу продукции, шт.;

Ц_l – цена приобретения единицы l -го комплектующего, руб./шт.;

$Н_{\text{т.з.}}$ – норма транспортно-заготовительных расходов (ТЗР), которая равна 10%.

Цена офисной бумаги составляет 246 рублей, а цена картриджа для принтера – 420 рублей. Определено по фактической стоимости приобретения 10.06.2021.

Расчет затрат на материалы и покупные изделия:

$$З_{\text{п}} = 1 \cdot 246 \cdot (1 + 0,1) + 1 \cdot 420 \cdot (1 + 0,1) = 733 \text{ руб.}$$

Результаты вычислений затрат на материалы и покупные изделия представлены в таблице 5.3.

Таблица 5.3 – Затраты на покупные изделия и материалы

Наименование покупных изделий и материалов	Единица измерения	Количество единиц	Цена за единицу изделия, руб.	Сумма, руб.	ТЗР, %	Итого затрат, руб.
Бумага офисная А4	упаковка	1	246	246	10	271
Картридж для принтера	штука	1	420	420		462
ИТОГО						733

5.5 Расчет амортизационных отчислений

В данной выпускной квалификационной работе разрабатывается программа. Следовательно, используются основные средства. Значит, следует

учесть и включить в затраты амортизационные отчисления по этим основным средствам.

Амортизационные отчисления по i -му основному средству за год определяются по формуле:

$$A_i = Ц_{п.н.i} \cdot \frac{H_{ai}}{100}, \quad (5)$$

где A_i – амортизационные отчисления по i -ому основному средству за год, руб.;

$Ц_{п.н.i}$ – первоначальная стоимость i -го основного средства, руб.;

H_{ai} – годовая норма амортизации i -го основного средства, %.

Годовая норма амортизации рассчитывается по формуле:

$$H_{ai} = \frac{1}{T_{п.н.}} \cdot 100\%, \quad (6)$$

где $T_{п.н.}$ – срок полезного использования основного средства.

Для ноутбуков срок полезного использования составляет максимум 3 года [16]. Для принтеров аналогично [17]. Тогда годовая норма амортизации для принтера и ноутбука равняется:

$$H_a = \frac{1}{3} \cdot 100\% = 33,3\%$$

Стоимость ноутбука составляет 68000 рублей, определено по фактической стоимости приобретения летом 2017 года. Стоимость принтера – 6859 рублей [18].

Амортизационные отчисления за год по ноутбуку составляют:

$$A_i = 68000 \cdot \frac{33,3}{100} = 22644 \text{ руб.}$$

Амортизационные отчисления за год по принтеру равняются:

$$A_i = 6859 \cdot \frac{33,3}{100} = 2284 \text{ руб.}$$

Сведения об амортизационных отчислениях за год представлены в таблице 5.4.

Таблица 5.4 – Амортизационные отчисления за год

Основное средство	Первоначальная стоимость, руб.	Срок полезного использования, лет	Годовая норма амортизации, %	Амортизационные отчисления за год, руб.
Ноутбук Dell Inspiron 15 7000 gaming	68000	3	33,3%	22644
Принтер HP Laser 107a	6859	3	33,3%	2284

Для определения величины амортизационных отчислений по используемым в процессе выполнения выпускной квалификационной работы основным средствам следует определить время, в течение которого используется это основное средство.

Срок использования ноутбука и принтера составляет 3 месяца.

Величина амортизационных отчислений по основному средству, используемому в выпускной квалификационной работе, рассчитывается по формуле:

$$A_{iВКР} = A_i \cdot \frac{T_{iВКР}}{12}, \quad (7)$$

где $A_{iВКР}$ – амортизационные отчисления по i -му основному средству, которое используется в выпускной квалификационной работе, руб.;

A_i – амортизационные отчисления по i -ому основному средству за год, руб.;

$T_{iВКР}$ – время, в течение которого используется i -ое основное средство, мес.

Амортизационные отчисления за время работы над ВКР по ноутбуку составляют:

$$A_{iВКР} = 22644 \cdot \frac{3}{12} = 5661 \text{ руб.}$$

Амортизационные отчисления по основным средствам за время работы над ВКР по принтеру равняются:

$$A_{iВКР} = 2284 \cdot \frac{3}{12} = 571 \text{ руб.}$$

Сведения об амортизационных отчислениях по основным средствам за время работы над ВКР приведены в таблице 5.5.

Таблица 5.5 – Амортизационные отчисления за время использования

Основное средство	Амортизационные отчисления за год, руб.	Срок полезного использования, мес.	Амортизационные отчисления за время использования, руб.
Ноутбук Dell Inspiron 15 7000 gaming	22644	3	5661
Принтер HP Laser 107a	2284	3	571
Итого			6232

5.6 Расчет накладных расходов

Накладные расходы – это расходы на управление и хозяйственное обслуживание. Они рассчитываются по формуле:

$$C_H = (З_{\text{осн.з./пл}} + З_{\text{доп.з./пл}}) \cdot \frac{H_{\text{н.р.}}}{100}, \quad (8)$$

где C_H – накладные расходы, руб.;

$З_{\text{осн.з./пл}}$ – расходы на основную заработную плату исполнителей, руб.;

$З_{\text{доп.з./пл}}$ – расходы на дополнительную заработную плату исполнителей (руб.);

$H_{\text{н.р.}}$ – норма накладных расходов, в рамках данной работы равная 20%.

Таким образом, величина накладных расходов составляет:

$$C_H = (91430 + 12800) \cdot \frac{20}{100} = 20846 \text{ руб.}$$

5.7 Расчет совокупной величины затрат

Чтобы определить полную стоимость разработки, нужно сложить все получившиеся затраты. Таким образом, полная стоимость разработки равна 182006 руб.

Совокупная величина затрат, связанная с разработкой, представлена в таблице 5.6.

Таблица 5.6 – Совокупная величина затрат

Наименование статьи	Сумма, руб.	Доля затрат, %
Расходы на оплату труда	104230	63,8
Отчисления на социальные нужды	31269	19,1
Материалы	733	0,4
Амортизационные отчисления	6232	3,8
Накладные расходы	20846	12,8
Итого	163310	100

5.8 Выводы

В данной главе были проанализированы и определены полные затраты на разработку чат-бота для ZulipChat. Таким образом, себестоимость написания программы составляет 163310 рублей.

Проанализировав все затраты, можно сказать, что наибольшую долю затрат составляют расходы на оплату труда (63,8 %), на втором месте отчисления на социальные нужды (19,1 %), а самыми незначительными издержками являются расходы на материалы (0,4 %).

ЗАКЛЮЧЕНИЕ

Целью дипломной работы было создание чат-бота для корпоративного мессенджера Zulip. В процессе выполнения работы были рассмотрены ключевые особенности мессенджера Zulip, а также предоставляемые им возможности для разработки; проанализирован подход к проработке функционала программы, на основе чего был прописан инструментарий чат-бота для Zulip; затем произведена настройка чат-бота в мессенджере и создана структура БД.

В итоге была разработана и запущена на арендованном сервере программа, реализующая чат-бота со следующим функционалом: регистрация пользователей, получение информации о сотрудниках, автоматическое заполнение корпоративных документов, составление расписания и добавление списка задач на каждый день, получение информации о нерабочих днях текущего года, ежедневное отправление напоминаний о текущем расписании, поздравление сотрудников с днем рождения. Набор данных функций является универсальным, поэтому такой чат-бот может быть полезен для любой организации.

В дальнейшем инструментарий бота можно расширить, доработать уже существующий функционал и добавить новые данные в БД, в зависимости от нужд конкретной организации. Также есть возможность реализации логирования программы, то есть создание файла, в котором будет храниться информация о работе программы. Это может быть полезно для отслеживания возникающих ошибок в ходе работы программы, а также для сбора данных, предоставляемых пользователями, например, ошибки в запросах, проанализировав которые, можно улучшить работу бота и упростить структуру запросов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Определение корпоративного мессенджера. – https://ru.wikipedia.org/wiki/Корпоративный_мессенджер
2. Определение чат-бота. – <https://www.unisender.com/ru/blog/kuhnya/chat-boty-vnedrenie/>
3. Что такое Zulip. – <https://bizzapps.ru/p/zulip/>
4. Интеграции Zulip. – <https://zulip.com/integrations/>
5. Тарифы пользования Zulip. – <https://zulip.com/plans/>
6. Интеграции Slack. – <https://slack.com/intl/en-ru/integrations>
7. Тарифы пользования Slack. – <https://slack.com/intl/en-ru/pricing>
8. Тарифы пользования RocketChat. – <https://rocketchat.com/pricing/>
9. SQLite. – <https://python-scripts.com/sqlite>
10. Библиотека docxtpl. – <https://pypi.org/project/docxtpl/>
11. Библиотека Petrovich. – <https://pypi.org/project/Petrovich/>
12. Алексеева О. Г. Методические указания по экономическому обоснованию выпускных квалификационных работ бакалавров: Метод. Указания, СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2013. 17 с.
13. Статистика зарплат Инженер в Санкт-Петербурге. Средний доход Инженер в Санкт-Петербурге. Индексы зарплат на Trud.com. – <https://sankt-peterburg.trud.com/salary/865/3670.html>
14. Статистика зарплат Руководитель проекта в Санкт-Петербурге. Средний доход Руководитель проекта в Санкт-Петербурге. Индексы зарплат на Trud.com. – <https://sankt-peterburg.trud.com/salary/865/67664.html>
15. Тарифы страховых взносов. – <https://www.nalog.gov.ru/rn77/taxation/insprem/>
16. Срок полезного использования ноутбука. – <https://taxslov.ru/ag/ag60.htm>
17. Срок полезного использования принтера. – <https://taxslov.ru/ag/ag3.htm>

18. Стоимость принтера в 2021 году. – <https://www.dns-shop.ru/product/d61d20807c3b3332/printer-lazernyj-hp-laser-107a/>
19. Чат-бот InMind для Telegram. – <https://ru.botostore.com/c/inmind-bot/>
20. Чат-бот Здоровье для Telegram. – <https://ru.botostore.com/c/zdorobot/>
21. Чат-бот Мура для Zulip. – <https://github.com/zerefwayne/myra-chat-bot>
22. Хостер Boodet Online. – <https://boodet.online>
23. Программный модуль xml.etree.ElementTree. – <https://docs.python.org/3/library/xml.etree.elementtree.html>

ПРИЛОЖЕНИЕ А

Пример шаблона заявления

Данные, необходимые для заполнения заявления на отпуск: название организации (organization), полное имя директора (directorFullName), полное имя работника (employeeFullName), дата начала отпуска (beginDate), дата окончания отпуска (endDate), количество отпускных дней (period), дата заполнения документа (todayDate) и имя сотрудника, записанное по шаблону “Фамилия И.О.” (employeeFIO).

Директору {{ organization }}
{{ directorFullName }}
от {{ employeeFullName }}

ЗАЯВЛЕНИЕ

Прошу предоставить мне ежегодный оплачиваемый отпуск с
{{ beginDate }} года по {{ endDate }} года сроком на {{ period }}
календарных дней.

Дата {{ todayDate }}

_____ {{ employeeFIO }}

Рисунок А.1 – Шаблон заявления на отпуск

ПРИЛОЖЕНИЕ Б

Пример автоматически заполненного заявления

Директору ООО "Название Организации"
Иванову Ивану Ивановичу
от Мартыновой Ксении Александровны

ЗАЯВЛЕНИЕ

Прошу предоставить мне ежегодный оплачиваемый отпуск с 30.07.2021 года по 8.08.2021 года сроком на 10 календарных дней.

Дата 7.6.2021

_____ Мартынова К.А.

Рисунок В.1 – Заполненное заявления на отпуск

ПРИЛОЖЕНИЕ В

Список модулей программы и их назначение

Таблица В.1 – Модули программы

Название модуля	Назначение модуля
bot.py	<p>Данный модуль необходим для обработки входящих сообщений. Именно здесь программа принимает сообщения для чат-бота от пользователей, обрабатывает их, на основе содержания сообщения определяет вид запроса, а затем формирует и отправляет ответ.</p> <p>Список функций:</p> <p>1) get_message() – принимает на вход переменную словарь, в которой хранится информация о входящем сообщении: текстовое содержание, список данных об отправителе и т. д.; обрабатывает содержание сообщения и, исходя из результатов обработки, передает данные одной из функций-обработчиков запросов, получает ответ и передает его в функцию send_msg() или функцию send_file();</p> <p>2) send_msg() – принимает на вход переменную с информацией о входящем сообщении и содержание ответного сообщения; отправляет текстовое сообщение в ответ на запрос пользователя;</p> <p>3) send_file() – принимает на вход переменную с информацией о входящем сообщении и путь к файлу, который будет отправлен пользователю; отправляет файл в ответ на запрос.</p>
BotInfo.py	<p>Модуль, в котором хранятся текстовые переменные с информацией о функциях чат-бота и о структуре запросов. Они необходимы для вывода меню.</p> <p>В данном модуле только одна функция – fullInfo(), которая возвращает текстовую переменную с полным меню чат-бота.</p>
Users.py	<p>Данный модуль необходим для реализации регистрации пользователя в базе данных чат-бота. Здесь пользователь проверяется на наличие регистрации в БД. Если сотрудник не обнаружен в базе данных, то он обязательно должен пройти регистрацию.</p> <p>Список функций:</p> <p>1) checkUser() – принимает на вход переменную с идентификатором пользователя; функция проверяет, существует ли пользователь с данным идентификатором в БД. Если существует, то происходит дальнейшая обработка запроса. А если нет, то пользователю предлагается пройти регистрацию;</p> <p>2) helloMessage() – возвращает текстовую переменную с информацией о регистрации в БД чат-бота, то есть сведения о формировании и содержании запроса;</p>

Продолжение таблицы В.1

Users.py	<p>3) getUserInfo() – принимает на вход переменную с идентификатором пользователя, а также содержание запроса о регистрации; данная функция обрабатывает запрос пользователя о регистрации, если все данные присутствует в запросе и введены корректно, то информация о пользователе заносится в БД, иначе выводится сообщение о неправильности введенных данных;</p> <p>4) capitalizeTheFirstLetter() – принимает на вход текстовую переменную с одним словом; функция заменяет первую букву в слове на заглавную, это необходимо для того, чтобы все имена пользователей, хранящиеся в БД, начинались с заглавной буквы;</p> <p>5) checkGender() – принимает на вход информацию о поле пользователя; проверяет правильность введенных данных;</p> <p>6) checkPhone() – принимает на вход информацию о мобильном телефоне сотрудника; функция проверяет, в правильном ли формате был введен номер телефона;</p> <p>7) checkDay() – принимает на вход информацию о дне рождения пользователя; проверяет, корректно ли введен день;</p> <p>8) checkMonth() – принимает на вход информацию о месяце рождения пользователя; проверяет, корректно ли введено число месяца;</p> <p>9) checkYear() – принимает на вход информацию о годе рождения пользователя; проверяет, корректно ли введен год, сравнивая его с текущим годом;</p> <p>10) getCurrentYear() – возвращает значение текущего года.</p>
UserInfo.py	<p>Модуль позволяет обработать запрос о получении информации о конкретном сотруднике. Если отправленное пользователем сообщение содержит в себе имя, фамилию, отчество сотрудника, существующего в БД, по отдельности или вместе, то чат-бот отправит сообщение с информацией об одном или нескольких найденных пользователях.</p> <p>Список функций:</p> <p>1) checkUserExistence() – принимает на вход содержание запроса пользователя; функция по отдельности проверяет каждое слово в запросе и сравнивает его с фамилией, именем или отчеством сотрудников в базе данных; если все слова в сообщении находятся в БД, то формируется массив из трех текстовых переменных (фамилии, имени и отчества, хотя бы одна переменная не должна быть пустой) и возвращается для дальнейшей обработки программой;</p>

Продолжение таблицы В.1

<p>UserInfo.py</p>	<p>2) capitalizeTheFirstLetter() – принимает на вход текстовую переменную с одним словом; функция заменяет первую букву в слове на заглавную, это необходимо для того, чтобы слова корректно сравнивались с именами сотрудников, которые хранятся в БД с заглавной буквы;</p> <p>3) checkSurname() – принимает на вход текстовую переменную с одним словом; проверяет, является ли это слово фамилией конкретного сотрудника из БД;</p> <p>4) checkFirstName() – принимает на вход текстовую переменную с одним словом; проверяет, является ли это слово именем конкретного сотрудника из БД;</p> <p>5) checkMiddleName() – принимает на вход текстовую переменную с одним словом; проверяет, является ли это слово отчеством конкретного сотрудника из БД</p> <p>6) getUserInfo() – принимает на вход массив из трех текстовых переменных: фамилии, имени и отчества сотрудника, информация о которых ищется в БД; в зависимости от того, сколько непустых переменных в массиве, функция формирует сообщение со списком найденных пользователей и информацией о них, а затем возвращает текст сообщения;</p> <p>7) dateForm(day, month) – принимает на вход информацию о дне и месяце рождения пользователя; формирует текстовую переменную, которая содержит в себе дату дня рождения найденного пользователя без года.</p>
<p>Docs.py</p>	<p>Данный модуль обрабатывает запросы на автоматическое заполнение документов. Если запрос для конкретного документа сформирован верно, то программа создает на сервере файл, а затем пересылает его в ответном сообщении пользователю. Иначе возвращается сообщение с инструкцией заполнения запроса к необходимому заявлению.</p> <p>Модуль содержит в себе ряд функций, которые написаны с целью обработки шаблонов корпоративных документов, содержащихся в БД чат-бота, а также формирования новых заполненных заявлений. Они возвращают путь к сформированному файлу. Количество этих функций равносильно количеству заявлений, присутствующих в БД. Список остальных функций:</p> <p>1) messageProcessing() – принимает на вход идентификатор пользователя и содержание сообщения; на основе содержания запроса функция решает, какой документ нужно заполнить пользователю и, в зависимости от решения, передает нужные данные в функцию обработчик шаблонов; возвращает путь к сформированному файлу;</p>

Продолжение таблицы В.1

Docs.py	<p>2) getOrganization() – функция возвращает название организации;</p> <p>3) getDirectorName() – возвращает полные имя, фамилию и отчество директора организации в дательном падеже;</p> <p>4) getEmployeeName() – принимает на вход идентификатор пользователя; возвращает полные имя, фамилию и отчество сотрудника, отправившего сообщение, в родительном падеже;</p> <p>5) getEmployeeFIO() – принимает на вход идентификатор пользователя; возвращает имя сотрудника, записанное по шаблону “Фамилия И.О.” ;</p> <p>6) getEmployeeFullFIO() – принимает на вход идентификатор пользователя; возвращает имя сотрудника в именительном падеже;</p> <p>7) getTodayDate() – возвращает сегодняшнюю дату в формате “ДД.ММ.ГГГГ”;</p> <p>8) getDocPath(id) – принимает на вход id документа; возвращает путь к шаблону документа;</p> <p>9) getDocName(id) – принимает на вход id документа; возвращает название корпоративного документа;</p> <p>10) getDocsList() – возвращает список документов, присутствующих в БД чат-бота.</p>
Holydays.py	<p>Модуль выполняет запросы на вывод информации о нерабочих днях в году или в конкретном месяце. Данные о выходных днях берутся из xml файла, расположенного в интернете. Здесь выполняется парсинг xml файла и, таким образом, формируется сообщение с информацией о нерабочих днях.</p> <p>Список функций:</p> <p>1) getUrl() – функция возвращает ссылку для доступа к xml файлу;</p> <p>2) getHolydaysName() – возвращает список с названиями выходных дней;</p> <p>3) getHolydays() – функция возвращает двумерный массив, в котором хранятся даты нерабочих дней, а также названия выходных, если они есть;</p> <p>4) getCurrentYear() – возвращает текстовую переменную, содержащую значение текущего года;</p>

Продолжение таблицы В.1

Holydays.py	<p>5) <code>getResponse()</code> – принимает на вход название месяца, если пользователь желает получить список выходных дней для конкретного месяца, по умолчанию значение переменной равно <code>None</code>; если месяц не указан пользователем, то функция формирует ответное сообщение для пользователя со списком всех нерабочих дней в году; если месяц корректно указан сотрудником, то формируется сообщение со списком выходных в этом месяце; если месяц указан неверно, то выводится сообщение об ошибке;</p> <p>6) <code>getHolydaysByMonth()</code> – принимает на вход двумерный массив со списком нерабочих дней, а также указанный пользователем месяц; возвращает массив со списком выходных в указанном месяце.</p>
Schedule.py	<p>Модуль обрабатывает запросы на добавление задач и предоставление существующего списка заданий пользователю.</p> <p>Список функций:</p> <p>1) <code>addTask()</code> – получает в качестве аргументов идентификатор пользователя и содержание запроса: дату, время и текст задачи; если запрос построен корректно, то функция добавляет новую задачу в БД, иначе выводит сообщение об ошибке;</p> <p>2) <code>getAllTasks()</code> – получает в качестве аргументов идентификатор пользователя; возвращает список с названиями выходных дней;</p> <p>3) <code>getTasksByDate()</code> – получает в качестве аргументов идентификатор пользователя и дату, на которую следует вывести список задач; если пользователем была введена корректная дата, то функция формирует сообщение со списком задач на конкретный день;</p> <p>4) <code>getTodayTasks()</code> – получает в качестве аргументов идентификатор пользователя; формирует сообщение со списком задач на текущий день;</p> <p>5) <code>deleteOldTasks()</code> – функция удаляет задачи, дата выполнения которых ранее текущего дня;</p> <p>6) <code>messageCheck()</code> – принимает в качестве аргументов тип (флаг) проверки сообщение и содержание запроса; функция проверяет запрос на корректность формулировки и значений входных данных (дня и месяца);</p> <p>7) <code>dateForm(day, month)</code> – принимает на вход день и дату задачи; формирует текстовую переменную, содержащую дату запроса в формате “ДД.ММ”.</p>

Продолжение таблицы В.1

ScheduleScript.py	<p>Это отдельный модуль, который каждый день в одно и то же время запускается на сервере. Необходим для того, чтобы информировать сотрудников о существующем расписании на текущий день.</p> <p>Список функций:</p> <p>1) checkToday() – для каждого пользователя функция проверяет перечень задач и в итоге формирует список пользователей, у которых присутствует расписание на текущий день, а затем каждому из них отправляет сообщение с расписанием;</p> <p>2) getTodayDate() – возвращает массив из двух числовых переменных: текущих дня и месяца.</p>
BirthdayScript.py	<p>Это отдельный модуль, который каждый день в одно и то же время запускается на сервере. Он нужен для того, чтобы поздравлять сотрудников организации с днем рождения, а также высылать напоминания другим пользователям о том, что у их коллеги сегодня день рождения.</p> <p>Список функций:</p> <p>1) checkToday() – функция проверяет, есть ли на текущий день сотрудники, празднующие свой день рождения; если есть, то информация об именинниках передается функциям sendHappyBirthday() и sendReminder();</p> <p>2) getTodayDate() – возвращает массив из двух числовых переменных: текущих дня и месяца;</p> <p>3) sendHappyBirthday() – принимает на вход идентификатор пользователя, у которого день рождения, и его имя; отправляет поздравление с днем рождения именинникам;</p> <p>4) sendReminder() – принимает на вход идентификатор пользователя, у которого день рождения, а также его полное имя; отправляет всем сотрудникам напоминание о том, что сегодня день рождения у одного или нескольких их коллег.</p>

ПРИЛОЖЕНИЕ Г

Фрагмент кода программы, в котором происходит обработка шаблона заявления на отпуск

```
# Функция, атоматически заполняющая заявление на отпуск на примере готового
шаблона
# Переменная path - путь к нужному шаблону, берется из БД
# userId - это электронная почта пользователя, является идентификатором
# каждого из пользователей
# beginDate - дата начала отпуска, вводится пользователем
# endDate - дата окончания отпуска, вводится пользователем
# period - количество отпускных дней

def vacationRequest(path, userId, beginDate, endDate, period):

    # Переменная, в которую записывается название организации
    # Функция getOrganization() извлекает из таблицы CompanyInfo базы данных
    # содержание столбца с названием организации и возвращает его
    organization = getOrganization()

    # Переменная, в которую записывается полное имя директора организации
    # в дательном падеже
    # Функция getDirectorName() извлекает из таблицы CompanyInfo содержание
    # четырех столбцов с именем, фамилией, отчеством и полом; преобразовывает
    # имена из именительного в дательный падеж; а затем соединяет три имени в
    # одну текстовую переменную и возвращает ее значение
    directorFullName = getDirectorName()

    # Переменная, содержащая полное имя сотрудника в родительном падеже
    # Функция getEmployeeName() извлекает из таблицы User фамилию, имя,
    # отчество и пол сотрудника с идентификатором userId; преобразовывает
    # имена из именительного в родительный падеж; а затем соединяет три имени
    # в одну текстовую переменную и возвращает ее значение
    employeeFullName = getEmployeeName(userId)

    # Переменная с текущей датой
    # Функуия getTodayDate() преобразовывает текущую дату в формат
    # "ЧЧ.ММ.ГГГГ", записывает значение в текстовую переменную
    # и возвращает ее
    todayDate = getTodayDate()

    # Переменная, в которой хранится имя пользователя в именительном падеже,
    # записанное по шаблону "Фамилия И.О."
    # Функция getEmployeeFIO() работает по такому же принципу, что и функция
    # getEmployeeName()
    # Достает данные из таблицы БД и преобразовывает имя в надлежащий вид,
    # затем записывает в текстовую переменную и возвращает ее
    employeeFIO = getEmployeeFIO(userId)

    # Переменная с названием документа. Документ располагается в папке
```

```

# user_docs, а в качестве имени используется идентификатор пользователя
finalDocName = "user_docs/" + userId + ".docx"

# Открытие шаблона
doc = DocxTemplate(path)

# Словарь с тегами шаблона, которым присваиваются соответствующие
# значения
context = { 'organization' : organization,
            'directorFullName' : directorFullName,
            'employeeFullName' : employeeFullName,
            'beginDate' : beginDate,
            'endDate' : endDate,
            'period' : period,
            'todayDate' : todayDate,
            'employeeFIO' : employeeFIO}

# Преобразование шаблона в заполненное заявление
doc.render(context)

# Сохранение заявления
doc.save(finalDocName)

# Функция возвращает полное название документа,
# чтобы впоследствии отправить его пользователю
return finalDocName

```

ПРИЛОЖЕНИЕ Д

Распечатка XML-файла со списком нерабочих дней в 2021 году

Элемент `holidays` содержит следующие атрибуты: `id` – идентификатор названия; `title` – название выходного дня. Элемент `days` содержит следующие атрибуты: `d` – дата нерабочего или сокращенного дня; `t` – атрибут, определяющий выходной (1) день или сокращенный (2); `h` – идентификатор названия выходного дня, если у выходного отсутствует название, то атрибут не указывается.

```
▼<calendar year="2021" lang="ru" date="2021.04.25">
  ▼<holidays>
    <holiday id="1" title="Новогодние каникулы (в ред. Федерального закона от 23.04.2012 № 35-ФЗ)"/>
    <holiday id="2" title="Рождество Христово"/>
    <holiday id="3" title="День защитника Отечества"/>
    <holiday id="4" title="Международный женский день"/>
    <holiday id="5" title="Праздник Весны и Труда"/>
    <holiday id="6" title="День Победы"/>
    <holiday id="7" title="День России"/>
    <holiday id="8" title="День народного единства"/>
    <holiday id="9" title="Нерабочие дни (Указ Президента от 23.04.2021 №242)"/>
  </holidays>
  ▼<days>
    <day d="01.01" t="1" h="1"/>
    <day d="01.02" t="1" h="1"/>
    <day d="01.03" t="1" h="1"/>
    <day d="01.04" t="1" h="1"/>
    <day d="01.05" t="1" h="1"/>
    <day d="01.06" t="1" h="1"/>
    <day d="01.07" t="1" h="2"/>
    <day d="01.08" t="1" h="1"/>
    <day d="02.20" t="2"/>
    <day d="02.22" t="1" f="02.20"/>
    <day d="02.23" t="1" h="3"/>
    <day d="03.08" t="1" h="4"/>
    <day d="04.30" t="2"/>
    <day d="05.01" t="1" h="5"/>
    <day d="05.03" t="1"/>
    <day d="05.04" t="1" h="9"/>
    <day d="05.05" t="1" h="9"/>
    <day d="05.06" t="1" h="9"/>
    <day d="05.07" t="1" h="9"/>
    <day d="05.09" t="1" h="6"/>
    <day d="05.10" t="1"/>
    <day d="06.11" t="2"/>
    <day d="06.12" t="1" h="7"/>
    <day d="06.14" t="1"/>
    <day d="11.03" t="2"/>
    <day d="11.04" t="1" h="8"/>
    <day d="11.05" t="1" f="01.02"/>
    <day d="12.31" t="1" f="01.03"/>
  </days>
</calendar>
```

Рисунок Д.1 – Распечатка XML-файла

ПРИЛОЖЕНИЕ Е

Фрагмент кода программы с парсингом XML-файла

```
# Функция, возвращающая ссылку на xml-файл текущего года
def getUrl():
    url = 'http://xmlcalendar.ru/data/ru/' +
        getCurrentYear() + '/calendar.xml'

    return url

# Функция, возвращающая массив, содержащий названия каждого из
# нерабочих дней
def getHolydaysName():
    # Ссылка на xml-файл
    url = getUrl()

    # Берем данные из xml-файла, расположенного по адресу url,
    # преобразовываем их в строку, а затем из строки представляем
    # весь xml-документ в виде дерева. root - переменная, являющаяся
    # корневым узлом дерева
    root = ET.fromstring(requests.get(url).content)

    # Массив, содержащий названия выходных дней
    holydaysName = []

    # Проходимся по всем дочерним узлам дерева
    for elem in root:
        # Если текущий тег равняется 'holidays', то
        if elem.tag == 'holidays':
            # проходимся по дочерним элементам в этом теге
            for item in elem:
                # и добавим в массив значения элемента с атрибутом
                # title, содержащим название нерабочего дня
                holydaysName.append(item.attrib.get('title'))

    # Возвращаем массив с наименованиями выходных дней
    return holydaysName

# Функция возвращает двумерный массив с датами нерабочих дней
# и их названиями
def getHolydays():
    url = getUrl()

    root = ET.fromstring(requests.get(url).content)

    # Массив, содержащий названия выходных дней
    holydaysName = getHolydaysName()

    # Инициализация массива, содержащего даты выходных и их названия
    holydays = []
    holydays.append([])
    holydays.append([])

    # Проходимся по всем дочерним узлам дерева
    for elem in root:
        # Если текущий тег равняется 'days', то
        if elem.tag == 'days':
            # проходимся по дочерним элементам в этом теге
            for item in elem:
                # Если атрибут t, определяющий выходной (1) это день
```

```

# или сокращенный (2), текущего элемента равен 1, то
if item.attrib.get('t') == '1':
    # добавляем в массив сначала число в формате "ДД.ММ"
    holydays[0].append(item.attrib.get('d').split('.')[1]+\
        '.'+item.attrib.get('d').split('.')[0])
    # Затем проверяем, существует ли название у нерабочего
    # дня
    id = item.attrib.get('h')
    # Если существует, то добавляем его в массив
    if id != None:
        holydays[1].append(holydaysName[int(id)-1])
    else:
        # иначе добавляем в массив пустое значение
        holydays[1].append(None)

# Возвращаем массив с датами и наименованиями нерабочих дней
return holydays

# Функция возвращает текстовую переменную со значением текущего года
def getCurrentYear():
    year = str(datetime.now().year)

    return year

```


ПРИЛОЖЕНИЕ Ж

Код модуля Schedule.py

```
import sqlite3
from datetime import datetime

# Подключение к базе данных
db = sqlite3.connect("mydb.db")

# Функция добавления задачи
def addTask(userId, content):
    # Проверка запроса пользователя на корректность
    errorMessage = messageCheck(1, content)
    # Если запрос составлен некорректно, то функция возвращает
    # сообщение об ошибке
    if errorMessage is not None:
        return errorMessage

    # Дата задачи
    taskDate = content[0].split('.')

    # Если дата введена некорректно, то функция возвращает
    # сообщение об ошибке
    errorMessage = messageCheck(3, taskDate)
    if errorMessage is not None:
        return errorMessage

    # День выполнения задания
    taskDay = int(taskDate[0])
    # Месяц выполнения задания
    taskMonth = int(taskDate[1])
    # Время выполнения задания
    taskTime = content[1]
    # Содержание задания
    taskText = ""

    for i in range(2, len(content)):
        taskText += content[i] + " "

    # Массив с информацией о задаче, которая будет отправлена в БД
    data = (userId, taskDay, taskMonth, taskTime, taskText)

    # Создаем курсор, специальный объект, который делает запросы
    # и получает их результаты
    dbCursor = db.cursor()
    # Добавляем данные в БД
    dbCursor.execute("INSERT INTO Tasks(UserId, TaskDay, TaskMonth,\
        TaskTime, TaskText) VALUES(?, ?, ?, ?, ?)", data)
    # Сохраняем транзакцию
    db.commit()
    # Закрываем соединение с БД
    dbCursor.close()

    # Возвращаем сообщение, подтверждающее успешное выполнение
    # запроса
    return 'Задание успешно добавлено!'

# Функция, возвращающая список всего расписания
# Принимает на вход идентификатор пользователя
def getAllTasks(userId):
```

```

dbCursor = db.cursor()
# Выбираем из БД все задания, которые существуют у пользователя
# с данным идентификатором, и сортируем их по дате
dbCursor.execute("SELECT * from Tasks WHERE UserId = '%s' \
ORDER BY TaskMonth, TaskDay, TaskTime" % userId)
# Список с задачами
tasksList = dbCursor.fetchall()
db.commit()
dbCursor.close()

# Содержание ответного сообщения
response = ''

# Если расписание отсутствует, то возвращаем соответствующее
# сообщение
if not tasksList:
    response = 'У Вас нет расписания'

    return response

# Иначе формируем сообщение с расписанием пользователя
response = 'Ваше расписание:\n'

# Распределяем расписание по дням и формируем структурированное сообщение
for i in range(0, len(tasksList)):
    if i == 0:
        response += '``spoiler ' + dateForm(tasksList[i][2], \
tasksList[i][3]) + '\n' + tasksList[i][4] + ' - ' + \
tasksList[i][5] + '\n'
    elif tasksList[i-1][2] == tasksList[i][2] and \
tasksList[i-1][3] == tasksList[i][3]:
        response += tasksList[i][4] + ' - ' + tasksList[i][5] + '\n'
    elif tasksList[i-1][2] != tasksList[i][2] \
or tasksList[i-1][3] != tasksList[i][3]:
        response += '``\n' + '``spoiler ' + \
dateForm(tasksList[i][2], tasksList[i][3]) + \
'\n' + tasksList[i][4] + ' - ' + tasksList[i][5] + '\n'
    elif i == len(tasksList) - 1:
        if tasksList[i-1][2] == tasksList[i][2] and \
tasksList[i-1][3] == tasksList[i][3]:
            response += tasksList[i][4] + ' - ' + \
tasksList[i][5] + '\n' + '``\n'
        elif tasksList[i-1][2] != tasksList[i][2] or \
tasksList[i-1][3] != tasksList[i][3]:
            response += '``\n' + '``spoiler ' + \
dateForm(tasksList[i][2], tasksList[i][3]) + '\n' \
+ tasksList[i][4] + ' - ' + tasksList[i][5] + '\n' + '``\n'

# Возвращаем переменную с содержанием ответного сообщения
return response

# Функция, возвращающая сообщение, содержащие расписание
# на конкретный день
# Принимает в качестве аргументов идентификатор пользователя
# и дату, на которую следует предоставить расписание
def getTasksByDate(userId, content):
    # Дата расписания
    taskDate = content.split('.')

    # Проверка даты на корректность ввода
    errorMessge = messageCheck(3, taskDate)
    if errorMessge is not None:
        return errorMessge

```

```

# День расписания
taskDay = int(taskDate[0])
# Месяц расписания
taskMonth = int(taskDate[1])

dbCursor = db.cursor()
# выбираем из БД задачи конкретного пользователя
# на указанную дату, отсортированные по времени
dbCursor.execute("SELECT * from Tasks WHERE TaskDay=:taskDay \
and TaskMonth=:taskMonth and UserId=:userId ORDER BY TaskTime",
                 {"taskDay": taskDay, "taskMonth": taskMonth, \
                 "userId": userId})
# Переменная, содержащая в себе массив с задачами пользователя
tasksList = dbCursor.fetchall()
db.commit()
dbCursor.close()

# ответное сообщение
response = ''

# Если расписания нет, то отправляем соответствующее сообщение
if not tasksList:
    response += 'Нет расписания на ' + date
# Иначе формируем ответ с расписанием на указанную дату
else:
    response += 'Расписание на ' + date + ':\n'
    for task in tasksList:
        response += task[4] + ' - ' + task[5] + '\n'

return response

# Функция возвращает расписание на текущий день
# Принимает на вход идентификатор пользователя
def getTodayTasks(userId):
    # Текущий день
    day = datetime.now().day
    # Текущий месяц
    month = datetime.now().month

    dbCursor = db.cursor()
    # Выбираем из БД задачи на текущий день для указанного пользователя
    dbCursor.execute("SELECT * from Tasks WHERE TaskDay=:taskDay \
and TaskMonth=:taskMonth and UserId=:userId ORDER BY TaskTime",
                    {"taskDay": day, "taskMonth": month, "userId": userId})
    # Массив со списком задач
    tasksList = dbCursor.fetchall()
    db.commit()
    dbCursor.close()

    # Формируем содержание ответного сообщения
    response = ''

    if not tasksList:
        response += 'Нет расписания на сегодня, ' + dateForm(day, month)
    else:
        response += 'Расписание на сегодня, ' + dateForm(day, month) + ':\n'
        for task in tasksList:
            response += task[4] + ' - ' + task[5] + '\n'

    return response

# Функция удаляет просроченные задачи

```

```

def deleteOldTasks():
    day = datetime.now().day
    month = datetime.now().month

    dbCursor = db.cursor()
    # Удаляет задачи прошедших месяцев
    dbCursor.execute("DELETE from Tasks WHERE TaskMonth<:month",
                     {"month": month})
    # Удаляет задачи текущего месяца прошедших дней
    dbCursor.execute("DELETE from Tasks WHERE \
                      TaskMonth=:month and TaskDay<:day",
                     {"month": month, "day": day})

    db.commit()
    dbCursor.close()

# Проверка запроса пользователя на корректность
# введенных данных
def messageCheck(flag, content):
    if flag == 1:
        if len(content) == 0:
            message = "Введите дату, время и содержание задания"
        elif len(content) == 1:
            message = "Введите время и содержание задания"
        elif len(content) == 2:
            message = "Введите содержание задания"
        else:
            message = None
    elif flag == 3:
        try:
            if len(content) != 2:
                message = "Введите корректную дату. Например, '01.01'"
            elif int(content[0]) > 31:
                message = "Введите корректный день"
            elif int(content[1]) > 12:
                message = "Введите корректный месяц"
            else:
                message = None
        except Exception:
            message = 'Данные введены некорректно'

    return message

# Функция преобразовывает дату в формат "ДД.ММ"
def dateForm(day, month):
    date = ''

    if day < 9:
        date += '0' + str(day)
    else:
        date += str(day)

    date += '.'

    if month < 9:
        date += '0' + str(month)
    else:
        date += str(month)

    return date

```