МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____Д.В. Топольский
«___»_____2022 г.

РАЗРАБОТКА TELEGRAM-БОТА ДЛЯ ПОДДЕРЖКИ ПОИСКА ВАКАНСИЙ В СФЕРЕ ФРИЛАНСА

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.п.н., доцент каф. ЭВМ
_____Ю.Г. Плаксина
«___»_____2022 г.

Автор работы,
студент группы КЭ-406
_____Д.А. Лотмарин
«___»_____2022 г.

Нормоконтролёр,
к.п.н., доцент каф. ЭВМ
_____М.А. Алтухова
«___»_____2022 г.

Челябинск-2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____Д.В. Топольский
«___»_____2022 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-406
Лотмарину Даниилу Анатольевичу
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы: «**Разработка Telegram-бота для поддержки поиска вакансий в сфере фриланса» утверждена приказом по университету 12 декабря 2021 г. №308/141

2. **Срок сдачи студентом законченной работы:** 1 июня 2022 г.

3. **Исходные данные к работе:** техническое задание.

Бизнес-требования.

Требования к функциональным характеристикам (Функциональные требования).

– поиск вакансий в сфере фриланса;

– выбор категорий вакансий (вакансии для таргетологов, дизайнеров, веб-дизайнеров, сторисмейкеров, smm-специалистов, маркетологов, менеджеров,

разработчиков программного обеспечения (ПО), копирайтеров, моушн-дизайнеров, специалистов по контекстной рекламе);

– выбор источников вакансий (Telegram, VK);

– получение вакансий по выбранным параметрам;

– возможность составления отрицательного отзыва на источник вакансии;

– отправляемые пользователям вакансии в Системе должны иметь ссылку на первоисточник;

– поиск вакансий должен осуществляться с помощью ключевых слов;

– пользователь должен иметь возможность связаться с администратором Telegram-бота для получения обратной связи;

– возможность приостановки бота на определенный пользователем период (в днях).

Временные ограничения.

Время между появлением вакансий в первоисточниках и их отправкой пользователям не должно превышать 10 секунд.

Нефункциональные требования.

Программа должна быть написана в среде разработки PyCharm на языке программирования Python v3.x.

Telegram-бот должен быть бесплатным для конечного пользователя.

Система должна устанавливаться на ОС Windows 7 и выше.

4. **Перечень подлежащих разработке вопросов:**

– анализ существующих программных решений, осуществляющих сбор вакансий для мессенджера Telegram;

– определение оптимальных технологий и сред программирования;

– проектирование и разработка собственного программного продукта для поддержки поиска вакансий в сфере фриланса для Telegram;

– тестирование функционала разработанного программного продукта.

5. **Дата выдачи задания:** 1 декабря 2021 г.

Руководитель работы _____/*Ю.Г. Плаксина*/

Студент _____/*Д.А. Лотмарин*/

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы (анализ существующих программных решений, осуществляющих сбор вакансий для мессенджера Telegram; определение оптимальных технологий и сред программирования)	07.02.2022	
Разработка модели, проектирование (проектирование собственного программного продукта для поддержки поиска вакансий в сфере фриланса для Telegram)	07.03.2022	
Реализация системы (разработка собственного программного продукта для поддержки поиска вакансий в сфере фриланса для Telegram)	04.04.2022	
Тестирование, отладка (тестирование функционала разработанного программного продукта)	10.05.2022	
Компоновка текста работы и сдача на нормоконтроль	16.05.2022	
Подготовка презентации и доклада	24.05.2022	

Руководитель работы _____/*Ю.Г. Плаксина*/

Студент _____/*Д.А. Лотмарин*/

АННОТАЦИЯ

Д.А. Лотмарин. Разработка Telegram-бота для поддержки поиска вакансий в сфере фриланса. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2022, 81 с., 22 ил., библиогр. список – 33 наим.

В рамках выпускной квалификационной работы проведен анализ существующих современных аналогов программных решений, осуществляющих сбор вакансий для мессенджера Telegram. Рассмотрены достоинства и недостатки современных средств и технологии, применяемых при разработке ботов. Обоснованы выбор языка и среды программирования. Определен необходимый функционал Telegram-бота. Приведены этапы проектирования, разработка и процедура тестирования функционала Telegram-бота, публикующего актуальные вакансии для специалистов в сфере фриланса.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время мессенджеры и социальные сети являются востребованными приложениями, обеспечивающими коммуникацию между пользователями. Благодаря их существованию люди обмениваются сообщениями друг с другом, совершают аудио- и видеозвонки, находят новых друзей, читаю новости и др. Особое внимание стоит обратить на возможность интеграции собственных приложений в мессенджеры и социальные сети. Одним из самых мощных инструментов, доступных разработчикам, являются application programming interface (API) пользовательских приложений. Чаще всего API используются при разработке ботов для конкретных сервисов, например, такси или магазины.

Telegram – кроссплатформенная система мгновенного обмена сообщениями с функциями VoIP [1]. В данной платформе заметен один из самых высоких показателей прироста аудитории за последнее время среди других мессенджеров [2]. Такие показатели обеспечиваются в основном за счет грамотного визуального оформления мессенджера и использования различных протоколов шифрования при обмене сообщениями и файлами разных типов, обеспечивающих защищенность просмотра переписок от третьих лиц, что повышает доверие пользователей к системе. В Telegram также присутствует возможность создания ботов для ведения собственных бизнес-процессов разработчиков.

Актуальность выбранной темы обусловлена тем, что мессенджер Telegram активно используется фрилансерами для поиска вакансий с прямыми контактами заказчиков. В нем есть более ста каналов и чатов, в которых выкладываются объявления о поиске специалистов в сфере фриланса. Для автоматизации сбора вакансий из данных источников обычно используются Telegram-боты, которыми

активно пользуются специалисты различных направлений. Однако пользование существующими аналогами доступно только на платной основе, а поиск вакансий в них осуществляется только в мессенджере Telegram.

Целью представленной выпускной квалификационной работы является разработка Telegram-бота, обеспечивающего сбор и обработку вакансий из Telegram и VK, для последующего предоставления их пользователям, и имеющего преимущества по сравнению с существующими аналогами.

Для достижения поставленной цели, необходимо решить следующие поставленные задачи:

– найти существующие на данный момент аналоги;

– провести анализ существующих аналогов для выявления преимуществ и недостатков, сделать выводы;

– провести анализ существующих технологий и сред программирования, обосновать выбор используемых для выполнения выпускной квалификационной работы;

– разработать архитектуру разрабатываемого бота;

– выполнить программную реализацию бота;

– провести тестирование разрабатываемого проекта на корректность выполнения задач и удобства интерфейса.

Работа состоит из пяти глав. В первой главе описаны существующие на данный момент аналоги и основные технологические решения, сделаны выводы. Во второй главе определены функциональные и нефункциональные требования разрабатываемого программного продукта. В третьей главе проведено проектирование архитектуры предлагаемого решения, структуры базы данных и структуры графического интерфейса. В четвертой главе представлена программная реализация Telegram-бота. В пятой главе проведено юзабилити и функциональное тестирование бота.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Telegram-боты (роботы или боты) – это специальные аккаунты в Telegram, созданные для того, чтобы автоматически обрабатывать и отправлять сообщения. Пользователи могут взаимодействовать с ботами при помощи сообщений, отправляемых через обычные или групповые чаты [3].

Боты отличаются от пользовательских аккаунтов по следующим признакам [3]:

– у роботов нет статуса «онлайн»;

– боты не могут сами начать общение с пользователем. Пользователь должен либо добавить робота в группу, в которой состоит, либо первым начать с ним диалог;

– имя пользователя у робота должно заканчиваться на «bot» (например, @controllerbot);

– для ботов выделено ограниченное место на серверах – все сообщения удаляются по прошествии определенного срока после обработки.

Важность существования ботов обусловлена функционалом, доступным для их разработки. Так, с помощью поддерживаемых языков программирования Telegram и Bot API можно создать ботов под различные нужды разработчика: от простых чат-ботов, общающихся с пользователями по заданному сценарию, до сложных интернет-магазинов или игр.

Фрилансеры активно пользуются Telegram-ботами, поддерживающими сбор вакансий. В частности, многие интересуются ботами, которые публикуют вакансии с прямыми контактами заказчиков, а не с ссылками на биржи вакансий. В настоящее время существует множество подобных систем, некоторые из которых будут рассмотрены в следующем разделе.

Целью создания программного продукта является разработка Telegram-бота для обеспечения фрилансеров бесплатным инструментом для поиска вакансий в VK и Telegram с прямыми контактами заказчиков. В разрабатываемом боте будут предусмотрены возможности выбора категорий вакансий и их источников (VK и/или Telegram).

1.1 Обзор аналогов

Существует множество Telegram-ботов для поддержки поиска вакансий в сфере фриланса. Рассмотрим наиболее используемые на данный момент. Среди таких ботов следует выделить следующие аналоги: @leadsshop_bot [4], @digitalvacancies2_bot [5], @leadpoiskbot [6].

Сбор вакансий во всех роботах осуществляется из заранее, определенного их разработчиками, множества чатов и каналов Telegram. Во всех рассмотренных аналогах для пользователей доступен выбор категорий получаемых вакансий, причем в каждом возможен выбор одной или нескольких категорий.

Категории вакансий описывают сферу деятельности работы, например, вакансии для таргетологов, дизайнеров, веб-разработчиков и др. Наличие возможности выбора категорий очень важно для фрилансеров ввиду ограниченности их профессиональных компетенций.

На рисунке 1 представлен перечень категорий вакансий в @leadsshop_bot.

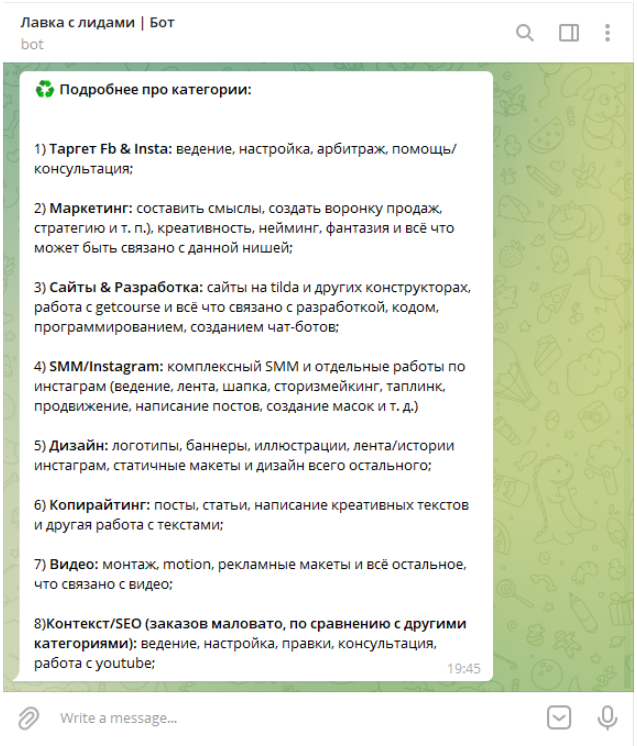

Рисунок 1 – Категории вакансий @leadsshop_bot

На рисунке 2 представлены категории вакансий в @digitalvacancies2_bot.

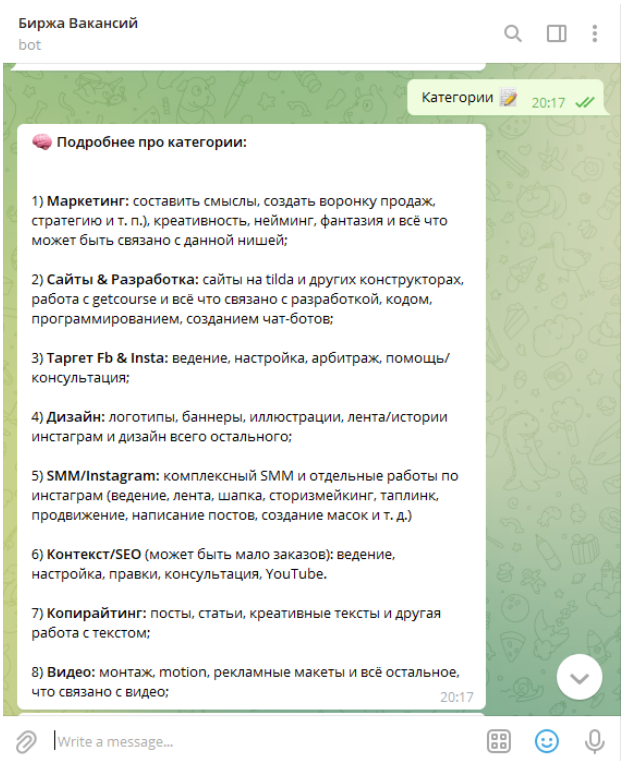

Рисунок 2 – Категории вакансий @digitalvacancies2_bot

Из рисунков 1 и 2 видно, что у ботов @leadsshop_bot и @digitalvacancies2_bot категории вакансий совпадают.

На рисунке 3 представлены категории вакансий в @leadpoiskbot.

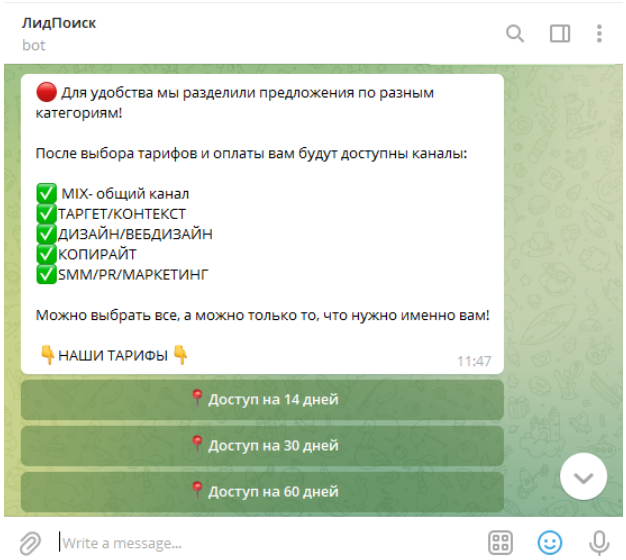

Рисунок 3 – Категории вакансий @leadpoiskbot

Из рисунка 3 видно, что в боте @leadpoiskbot только 4 категории вакансий.

В роботах @leadsshop_bot и @digitalvacancies2_bot вакансии публикуются непосредственно в ботах. Благодаря этому доступ к вакансиям независимо от выбранных категорий предоставляется в одном месте.

В отличие от вышеупомянутых аналогов, в боте @leadpoiskbot оформляется подписка, а вакансии публикуются в пяти закрытых каналах. Это может создавать дискомфорт для пользователя, если его интересуют вакансии из нескольких категорий, поскольку ему придется просматривать несколько каналов.

Под временем появления вакансий понимается промежуток времени между появлением вакансии в ее первоисточнике и боте. В ботах @leadsshop_bot и @digitalvacancies2_bot этот показатель составляет менее 10 секунд, а в роботе @leadpoiskbot – 2-3 минуты.

Каждый из рассматриваемых аналогов предоставляет доступ к вакансиям после оплаты подписки. В таблице 1 представлены их тарифы и цены.

Таблица 1 – Тарифы и цены аналогов

Тариф	**Цена (рублей)**		
	@leadsshop_bot	@digitalvacancies2_bot	@leadpoiskbot
5 дней	1300	–	–
1 неделя	1700	700	–
11 дней	2100	–	–
2 недели	2550	1000	499
17 дней	3000	–	–
3 недели	3400	–	–
25 дней	3800	–	–
1 месяц	4200	1500	650
2 месяца	–	–	1000

В ботах @leadsshop_bot и @digitalvacancies2_bot есть возможность приостанавливать подписку. Представим в таблице 2 для данных аналогов количество дней паузы в разных тарифах.

Таблица 2 – Время приостановления подписки аналогов

Тариф	**Пауза (дней)**	
	@leadsshop_bot	@digitalvacancies2_bot
5 дней	2	–
1 неделя	4	3
11 дней	5	–
2 недели	7	6
17 дней	8	–
3 недели	10	–
25 дней	12	–
1 месяц	14	14

Одной из самых важных функций в боте для поддержки поиска вакансий является наличие тестового (бесплатного) периода. Благодаря этой функции у пользователя будет время определить, подходит ему данный бот или нет. Из рассматриваемых аналогов тестовый период есть только в @digitalvacancies2_bot (1 день).

Разрабатываемый Telegram-бот предлагается сделать полностью бесплатным для пользователя. Поэтому наличие тестового периода и

возможность приостанавливать подписку для него необязательны. На основе просмотра и анализа текстового описания вакансий можно сделать вывод о том, что рассматриваемые аналоги осуществляют сбор вакансий только из Telegram. Разрабатываемый бот дополнительно будет искать вакансии в VK. Также важной составляющей робота является обеспечение быстрого появления вакансий (менее 10 секунд), наличие большого выбора категорий вакансий и удобство интерфейса. Собранные вакансии будут публиковаться непосредственно в Telegram-боте.

1.2 Анализ основных технологических решений

1.2.1 Выбор языка программирования

Разработка бота для Telegram возможна практически на любом языке программирования. Рассмотрим самые популярные языки программирования, используемые в разработке роботов для Telegram, и выберем оптимальный вариант, который будет удовлетворять заявленным требованиям.

На рисунке 4 представлен созданный Github рейтинг языков программирования PyPL [7].

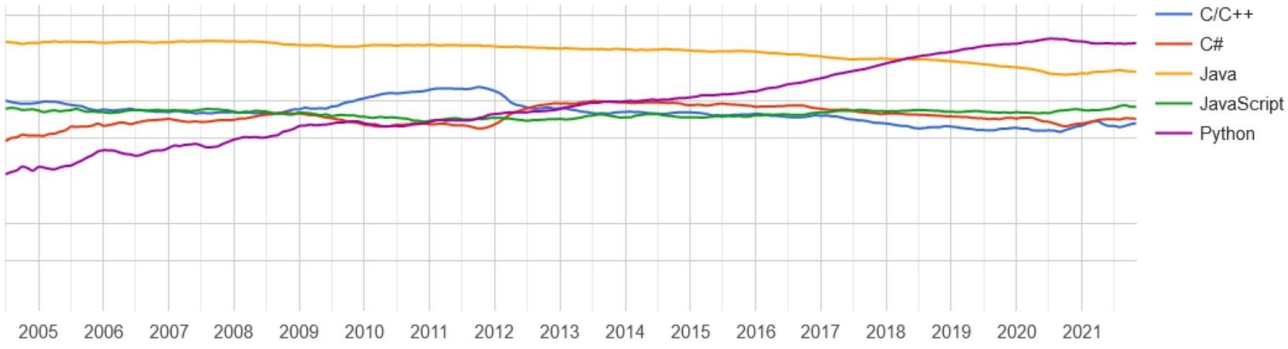

Рисунок 4 – Рейтинг PyPL [7]

Из данных на рисунке 4 можно сделать вывод, что, согласно рейтингу PyPL, самым популярным языком программирования является Python, затем идут Java, JavaScript, C# и C/C++.

Сравним популярность языков программирования по версии TIOBE (рисунок 5).

Dec 2021	Dec 2020	Change	Programming Language	Ratings	Change
1	3	↑	Python	12.90%	+0.69%
2	1	↓	C	11.80%	-4.69%
3	2	↓	Java	10.12%	-2.41%
4	4		C++	7.73%	+0.82%
5	5		C#	6.40%	+2.21%
6	6		Visual Basic	5.40%	+1.48%
7	7		JavaScript	2.30%	-0.06%
8	12	⇈	Assembly language	2.25%	+0.91%
9	10	↑	SQL	1.79%	+0.26%
10	13	↑	Swift	1.76%	+0.54%
11	9	↓	R	1.58%	-0.01%
12	8	⇊	PHP	1.50%	-0.62%
13	23	⇈	Classic Visual Basic	1.27%	+0.56%
14	11	↓	Groovy	1.23%	-0.30%
15	15		Ruby	1.16%	-0.01%

Рисунок 5 – Рейтинг TIOBE [8]

По итогу на декабрь 2021 года в пятерку самых популярных языков программирования вошли Python, C, Java, C++ и C#. JavaScript, в отличие от рейтинга PyPL, занял лишь 7-ю позицию.

Каждый язык программирования имеет свои особенности и применяется для решения конкретных целей разработчика. Проведем сравнительный анализ языков программирования Python, Java, JavaScript, C/C++ и C#.

Python – это один из самых популярных высокоуровневых языков программирования, предназначенный для создания приложений различных типов. На нем разрабатываются веб-приложения, игры, настольные программы, он позволяет работать с базами данных. Довольно большое распространение Python получил в области машинного обучения и исследований искусственного интеллекта [9].

Достоинства Python [10]:

– простота и удобство синтаксиса;

– кроссплатформенность;

– высокая скорость разработки;

– большое сообщество разработчиков;

– широкое применение;

– большое количество качественных библиотек;

– интерпретируемый язык программирования;

– используется компаниями-гигантами, такими как Spotify, Amazon, YouTube, Instagram, NASA и другими [11];

– бесплатный язык программирования.

Недостатки Python:

– низкая скорость выполнения программ;

– большое потребление памяти;

– плохая поддержка многопоточности.

Java – строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems [12]. Чаще всего данный язык программирования используют при разработке web-приложений, web-сайтов и игр.

Преимущества Java [10]:

– безопасность;

– независимость от платформы;

– поддержка многопоточности;

– простота синтаксиса;

– интерпретируемый язык программирования;

– стандарт для корпоративных вычислительных систем.

Недостатки Java:

– платное коммерческое использование;

– отсутствие нативного дизайна;

– низкая производительность.

JavaScript – это объектно-ориентированный язык создания сценариев. Чаще всего JavaScript находится внутри документов HTML и обеспечивает уровень взаимодействия с веб-страницами [13].

Достоинства JavaScript [14]:

– незаменимость для web-разработки;

– высокая скорость работы и производительность;

– большая инфраструктура;

– простота и рациональность применения.

Недостатки JavaScript:

– отсутствие возможности чтения и загрузки файлов;

– нестрогая типизация;

– нет поддержки удаленного доступа;

– доступность для злоумышленников.

C++ – компилируемый, статически типизированный язык программирования общего назначения [15]. Он является усовершенствованной версией языка C. C++ используется во всех сферах деятельности

программирования: от высоконагруженных систем до программирования микроконтроллеров.

Достоинства C++ [10]:

– высокая совместимость с языком программирования C;

– высокая скорость выполнения программ;

– кроссплатформенность;

– поддержка многопоточности;

– доступность документации;

– бесплатный язык программирования.

Недостатки C++:

– низкоуровневый язык программирования;

– сложный синтаксис;

– отсутствие автоматического сборщика мусора.

C# является одним из самых мощных, быстро развивающихся и востребованных языков в IT. На нем пишут решения под любые нужды: от небольших десктопных программ до крупных web-порталов и web-сервисов [16].

Преимущества C# [10]:

– поддержка компанией Microsoft;

– большое сообщество разработчиков;

– унифицированная система типизации;

– простота синтаксиса.

Недостатки C#:

– бесплатное использование языка только для не крупных фирм;

– не кроссплатформенный язык программирования.

Исходя из сравнительного анализа можно сделать вывод о том, что языки Java и JavaScript не подходят для разработки Telegram-бота для поддержки

поиска вакансий в сфере фриланса, поскольку они больше подходят для web-разработки.

Также, согласно запросам в Интернете [24], чаще всего для разработки ботов для мессенджера Telegram используется язык программирования Python. Этот язык является универсальным для различных разработок и имеет более простой синтаксис, в сравнение с рассмотренными языками программирования. Для Python существует множество библиотек с доступной документацией для взаимодействия с API, что ускоряет и упрощает разработку.

Для разработки Telegram-бота для поддержки поиска вакансий в сфере фриланса был выбран язык программирования Python, поскольку поддерживает множество библиотек, имеет простой синтаксис, что позволит ускорить процесс разработки.

1.2.2 Выбор среды разработки

IDE (Integrated Development Environment или интегрированная среда разработки) – это программа, предназначенная для разработки программного обеспечения (ПО). Интегрированные среды разработки созданы для ускорения процесса разработки ПО.

Обычно IDE ориентирована на определенный язык программирования. Однако некоторые IDE способны поддерживать несколько языков, например, Microsoft Visual Studio, WinDev и Xcode [17].

IDE обычно представляет собой единую программу, в которой проводится вся разработка. Она обычно содержит много функций для создания, изменения, компилирования, развертывания и отладки программного обеспечения. Цель среды разработки заключается в том, чтобы абстрагировать конфигурацию, необходимую, чтобы объединить утилиты командной строки в одном модуле.

IDE позволяет проанализировать код и тем самым обеспечить мгновенную обратную связь и уведомить о синтаксических ошибках [17].

Большинство современных IDE являются имеют графический интерфейс, а благодаря цветовой подсветке синтаксиса и возможности интегрировать библиотеки они являются незаменимыми инструментами для разработки.

Поскольку разработка Системы будет осуществляться на языке программирования Python, в качестве среды разработки была выбрана PyCharm Community [18]. Данная IDE является бесплатной и обладает всеми необходимыми инструментами для разработки, такими как подсветка синтаксиса, возможность добавлять внешние библиотеки, добавлять в проект классы и файлы.

1.2.3 Выбор системы управления базой данных

Для хранения и обработки данных в боте понадобится система управления базой данных (СУБД).

Существует два основных типа баз данных (БД) – реляционные и нереляционные.

Реляционные базы данных основаны на реляционной модели – интуитивно понятном, наглядном табличном способе представления данных. Каждая строка, содержащая в таблице такой базы данных, представляет собой запись с уникальным идентификатором, который называют ключом. Столбцы таблицы имеют атрибуты данных, а каждая запись обычно содержит значение для каждого атрибута, что дает возможность легко устанавливать взаимосвязь между элементами данных [19].

Нереляционная база данных – это база данных, в которой, в отличие от большинства традиционных систем баз данных, не используется табличная схема строк и столбцов [20].

Большинство СУБД работают с базами данных, основанных на реляционной модели. Данные БД пользуются популярностью в основном благодаря поддержки языка программирования SQL, предназначенного для построения запросов к данным в таблицах баз данных.

Поскольку в качестве языка программирования для разработки бота был выбран Python, рассмотрим основные СУБД, которые с ним используются. Среди таких можно выделить SQLite, MySQL и PostgreSQL.

SQLite является одной из самых простых СУБД, к которой можно подключиться с помощью Python, поскольку для этого не требуется устанавливать какие-либо внешние модули. По умолчанию стандартная библиотека Python уже содержит модуль sqlite3 [21].

Особенности SQLite [22]:

– не требует отдельного процесса сервера или системы для работы;

– написан на ANSI-C и прост в использовании API;

– отсутствие внешних зависимостей;

– распространяется бесплатно;

– занимает мало места на жестком диске (от нескольких килобайт до нескольких десятков мегабайт).

Ограничения SQLite:

– имеется несколько неподдерживаемых функций.

Достоинства MySQL:

– большой функционал (поддержка групповых функций, полная поддержка для операторов SQL GROUP BY и ORDER BY с выражениями SQL, полная поддержка операторов и функций в SELECT- и WHERE- частях

запросов и др.);

– простой в использовании;

– высокая скорость обработки запросов в единицу времени;

– является универсальной, может применяться для работы как с малыми, так и с большими данными.

Недостатки MySQL:

– у Python по умолчанию нет модуля, который можно использовать для подключения к MySQL;

– низкая скорость настройки базы данных. В MySQL некоторые функции, в отличие от других баз данных, необходимо настраивать вручную, например, создавать инкрементные резервные копии;

– возможное прекращение поддержки продукта в России.

Преимущества PostgreSQL [23]:

– отказоустойчивость;

– кроссплатформенность;

– распространяется бесплатно.

Ограничения PostgreSQL:

– у Python по умолчанию нет модуля, который можно использовать для подключения к MySQL;

– сложная конфигурация;

– низкая скорость обработки запросов по сравнению с другими базами данных (например, SQLite, MySQL).

В качестве СУБД для разработки бота был выбран SQLite, поскольку является бесплатным в использовании, просто устанавливается, а сама база данных занимает мало места на жестком диске (от нескольких килобайт до нескольких десятков мегабайт).

1.3 Вывод

В данном разделе был проведен анализ основных аналогов разрабатываемой программы, были определены их основные преимущества и недостатки. Из недостатков существующих ботов стоит отметить высокую цену для их использования, отсутствие возможности оставлять жалобу на вакансии. Также сбор вакансий осуществляется только из Telegram. Главными преимуществами некоторых систем являются выбор категорий вакансий и быстрое появления вакансий, после появления их в первоисточниках. Для выполнения выпускной квалификационной работы были выбраны следующие технологические решения: IDE – PyCharm (является наиболее используемой у разработчиков проектов на языке программирования Python); СУБД – SQLite, поскольку является бесплатным в использовании, просто устанавливается и занимает мало места.

В качестве языка программирования был выбран Python, поскольку поддерживает множество библиотек, имеет простой синтаксис, что позволит ускорить процесс разработки. Также Telegram-боты не потребляют много ресурсов памяти, вследствие чего на любом языке программирования скорость работы бота будет приблизительно одинаковой. Python отлично подходит для реализации нейронных сетей, а Telegram-бот в дальнейшем можно будет развить, чтобы в нем использовался искусственный интеллект.

2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

2.1 Функциональные требования

Основные требования к функционалу системы:

– поиск вакансий в сфере фриланса в VK и Telegram;

– выбор категорий вакансий (вакансии для таргетологов, дизайнеров, веб-дизайнеров, сторисмейкеров, smm-специалистов, маркетологов, менеджеров, разработчиков ПО, копирайтеров, моушн-дизайнеров, специалистов по контекстной рекламе);

– выбор источников вакансий (Telegram, VK);

– получение вакансий по выбранным параметрам;

– возможность составления отрицательного отзыва на источник вакансии;

– отправляемые пользователям вакансии в Системе должны иметь ссылку на первоисточник;

– поиск вакансий в сфере фриланса должен осуществляться с помощью ключевых слов;

– пользователь должен иметь возможность связаться с администратором Telegram-бота для получения обратной связи;

– возможность приостановки бота на определенный пользователем период (в днях).

2.2 Нефункциональные требования

Основные нефункциональные требования:

– программа должна быть написана в среде разработки PyCharm на языке программирования Python v3.x;

– Telegram-бот должен быть бесплатным для конечного пользователя;

– система должна устанавливаться на ОС Windows 7 и выше.

3 ПРОЕКТИРОВАНИЕ

3.1 Основные сведения

Telegram-бот не может самостоятельно добавляться в группы или каналы, соответственно он самостоятельно не сможет осуществлять поиск вакансий в Telegram. Для решения данной задачи должно использоваться клиентское приложение (Клиент Telegram).

Клиент Telegram – это запрограммированный пользовательский аккаунт. Для того, чтобы Клиент Telegram мог выполнять определенные действия по заранее прописанному алгоритму, необходимо получить api_id и api_hash, которые представляют из себя логин и пароль клиентского приложения соответственно. Использование данных параметров позволяет общаться с сервером Telegram с помощью https-запросов и API.

Клиент Telegram будет искать вакансии в заранее определенных группах и каналах в Telegram с использованием ключевых слов, обрабатывая все новые сообщения и публикации. Ключевые слова подразделяются на обязательные и исключающие. Они настраиваются разработчиком Telegram-бота до его запуска для пользователей. Так, чтобы вакансия считалась подходящей для последующей ее отправки боту, ее текст должен содержать хотя бы одно обязательное ключевое слово и ни одного исключающего.

Для поиска вакансий в VK будет также использоваться клиентское приложение (Клиент VK). Клиент VK будет искать вакансии в заранее определенных каналах VK с помощью ключевых слов, обрабатывая все новые публикации. Клиентское приложение VK будет общаться с помощью https-запросов и API с сервером VK.

Текст собранных Клиентом Telegram вакансии будут отправляться боту в виде сообщений, который в свою очередь будет их отправлять пользователям, подключенным к нему, с учетом их предпочтений.

Поскольку Клиент VK не может напрямую отправлять сообщения роботу, при нахождении вакансии в VK будет вызываться метод в программе, в который в качестве параметра будет передаваться текст вакансий. В данном методе будет реализована отправка ботом вакансий подходящим пользователям.

3.1 Архитектура предлагаемого решения

На рисунке 6 представлена архитектура программы.

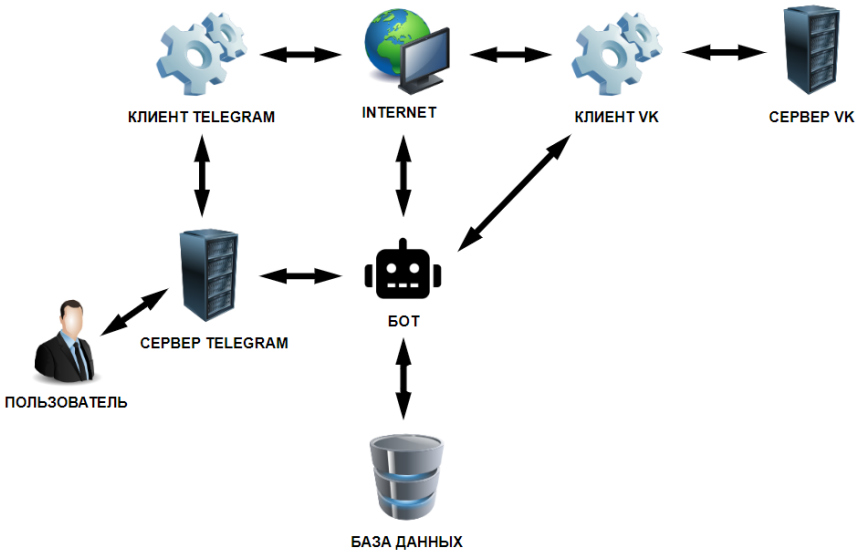

Рисунок 6 – Архитектура программы

Согласно архитектуре программы, Клиент VK, подключенный к Интернету, осуществляет сбор и обработку вакансий в социальной сети VK. Собранные заявки перенаправляются боту.

Клиент Telegram, подключенный к Интернету, осуществляет поиск вакансий в мессенджере Telegram. Бот, взаимодействуя с Клиентом Telegram с

помощью сервера Telegram, получает обработанные вакансии в виде сообщений от Клиента Telegram.

Бот, подключенный к сети Internet, взаимодействует с базой данных (БД). С помощью SQL запросов он записывает в нее информацию о пользователях, а именно id в Telegram, выбранные категории вакансий, источники вакансий (VK и/или Telegram), время приостановки бота, активность пользователя. Полученные от Клиентов вакансии бот отправляет пользователям, получая информацию о их предпочтениях из БД с помощью SQL запросов.

Списки каналов и чатов в VK и Telegram и списки ключевых слов не изменяются, а доступ к ним должен осуществляться непрерывно. В связи с этим, предлагается хранить данную информацию в файлах с форматом CSV (разделители запятые). После запуска программы, данные считываются в массивы.

3.2 Проектирование структуры базы данных

Эскиз базы данных представлен на рисунке 7.

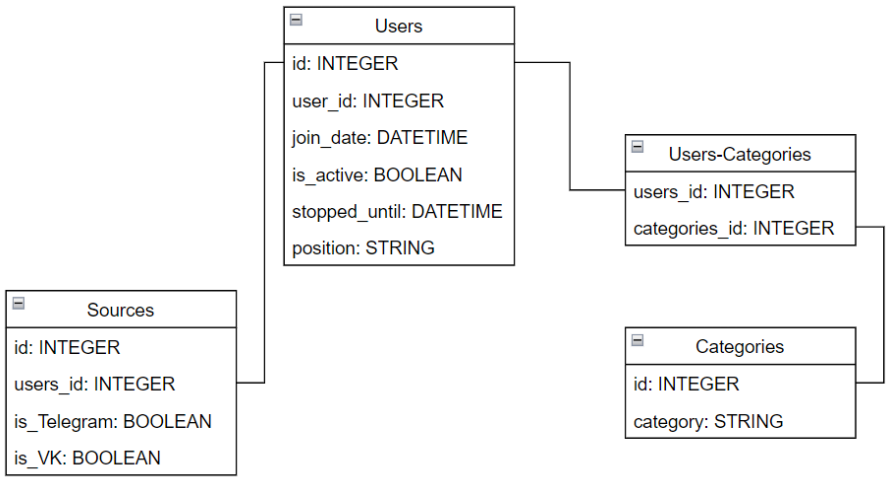

Рисунок 7 – Эскиз базы данных

Были спроектированы 4 таблицы: Users, Sources, Users-Categories, Categories. Данные таблицы хранят информацию о пользователях, которая проверяется Telegram-ботом для выполнения определенных действий, например, отправка вакансии всем пользователям, у которых выбрана определенная категория.

В таблице Users имеются следующие поля:

– id – уникальный идентификатор для каждого значения в таблице;

– user_id – id пользователя в Telegram;

– join_date – дата и время регистрации пользователя в боте;

– is_active – содержит информацию о том, отключил ли пользователь бота. Telegram-бот не может отправлять сообщения пользователям, которые его отключили;

– stopped_until – поле, необходимое для реализации функции приостановления бота на определенный срок. В данное поле записывается дата и время, когда бот сможет отправлять вакансии пользователю;

– position – для определения позиции пользователя в функциональном меню бота.

В таблице Sources имеются следующие поля:

– id – уникальный идентификатор для каждого значения в таблице;

– users_id – для связи с таблицей Users;

– is_Telegram – для выбора Telegram, как источника вакансий;

– is_Vk – для выбора VK, как источника вакансий.

Таблица Categories содержит два поля:

– id – уникальный идентификатор для каждого значения в таблице;

– category – наименование категории вакансий.

Таблица Users-Categories нужна для обеспечения связи многие-ко-многим между таблицами Users и Categories. В ней содержится два поля:

– users_id – для связи с таблицей Users;

– categories_id – для связи с таблицей Categories.

3.3 Проектирование структуры графического интерфейса

На рисунках 8-13 представлены прототипы пользовательского интерфейса Telegram-бота.

После нажатия команды /start, пользователь видит три кнопки: “Настроить бота”, “Помощь” и “Статус бота” (рисунок 8).

При нажатии на кнопку “Статус бота” пользователь может просмотреть выбранные категории вакансий и их источники, а также информацию о его активности.

При нажатии на кнопку “Настроить бота” пользователь попадает в окно выбора категорий вакансий (рисунок 9). При нажатии с соответствующей категорией появляется символ галочки, что означает, что данная категория выбрана пользователем, как желаемая. Пользователь может выбрать любое количество категорий вакансий. После нажатия кнопки “Подтвердить” пользователь может указать источники вакансий (рисунок 10). После подтверждения выбора, пользователь перенаправляется в главное меню и начинает получать вакансии.

Рисунок 8 – Главное меню

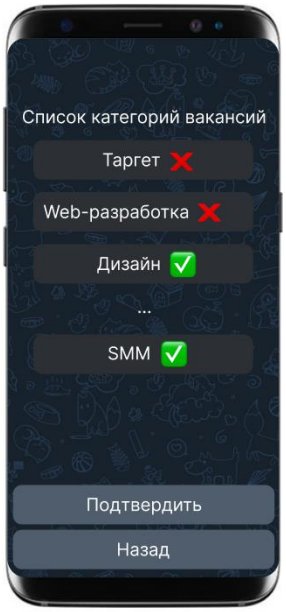

Рисунок 9 – Окно выбора категорий вакансий

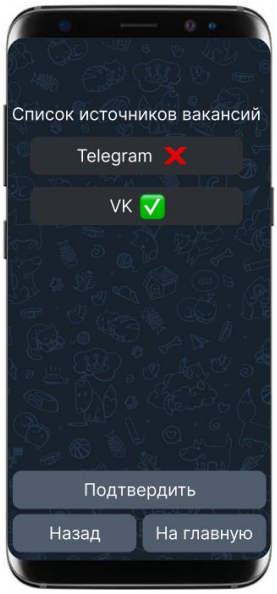

Рисунок 10 – Окно выбора источников вакансий

Если пользователь захочет изменить настройки уже запущенного бота, то для него будет доступно меню, представленное на рисунке 11.

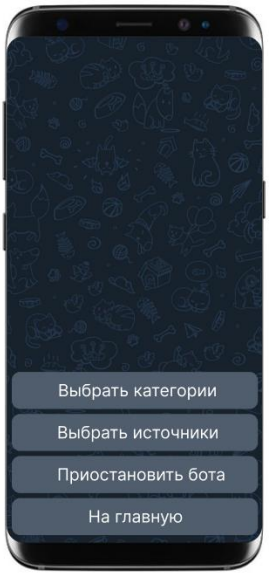

Рисунок 11 – Окно изменения настроек бота

На рисунке 11 кнопка “Приостановить бота” заменяется на “Возобновить бота”, если бот приостановлен для конкретного пользователя.

На рисунке 12 представлен прототип интерфейса окна приостановки бота. Пользователь может задать любую корректную дату и время, в соответствии с форматом записи, до какого момента будет приостановлен бот.

Рисунок 12 – Окно настройки приостановки бота

При нажатии на кнопку «Помощь» в главном меню бота (рисунок 8), пользователь перенаправляется в меню, где может написать в поддержку или прочитать информацию о боте (рисунок 13).

Рисунок 13 – Окно «Помощь»

4 РЕАЛИЗАЦИЯ

4.1 Реализация базы данных

На рисунке 14 представлена схема базы данных, реализованной в СУБД SQLite.

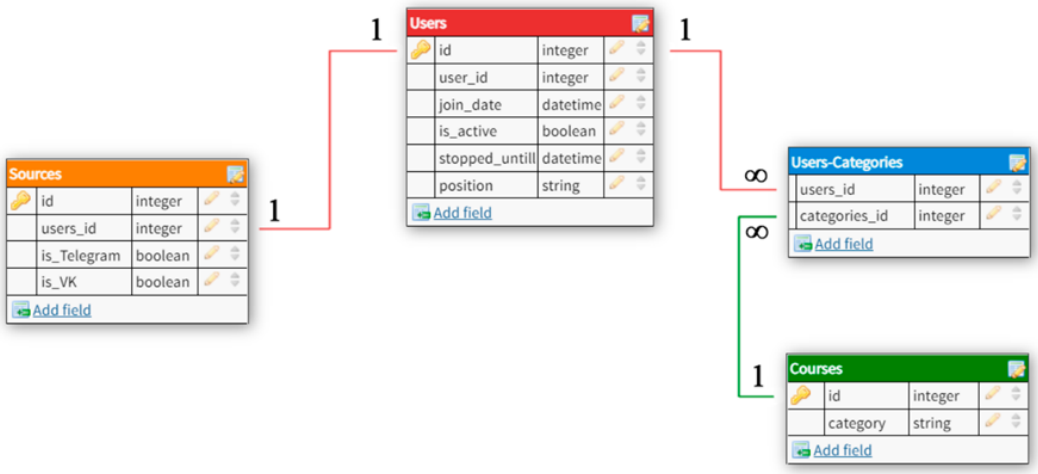

Рисунок 14 – Схема базы данных

4.2 Реализация графического интерфейса

Чтобы активировать бота, пользователь должен нажать команду /start (рисунок 15). Далее пользователь может приступить к первичной настройке категорий вакансий и их источников. Пользовательский выбор сохраняется в базе данных, а для пользователя отображается в виде зеленого чекбокса (рисунок 16).

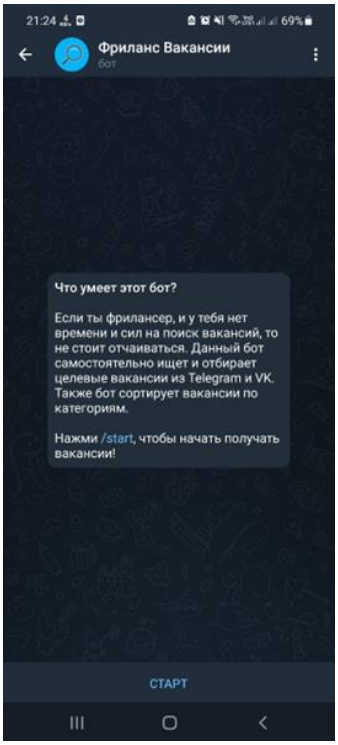

Рисунок 15 – Окно бота до его запуска

Рисунок 16 – Выбор категорий вакансий

Реализация приостановки бота представлена на рисунке 17. Бот проверяет указанные дату и время, введенные пользователем. Дата и время, до какого периода будет приостановлена функция получения вакансий, должны вводиться в формате YYYY-MM-DD hh-mm, где: YYYY – год, MM – месяц, DD – день, hh – часы, mm – минуты. Дата и время не могут быть меньше текущей, а год не должен быть больше 2030. Корректно введенные пользователем дата и время записываются в базу данных.

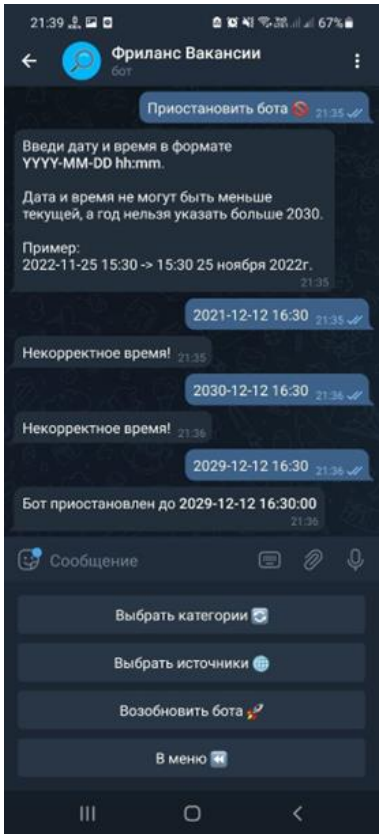

Рисунок 17 – Приостановка бота

Пользователь может посмотреть статус бота, список выбранных категорий и источников вакансий в виде одного сообщения. Для этого он должен нажать на кнопку «Статус бота» в главном меню (рисунок 18).

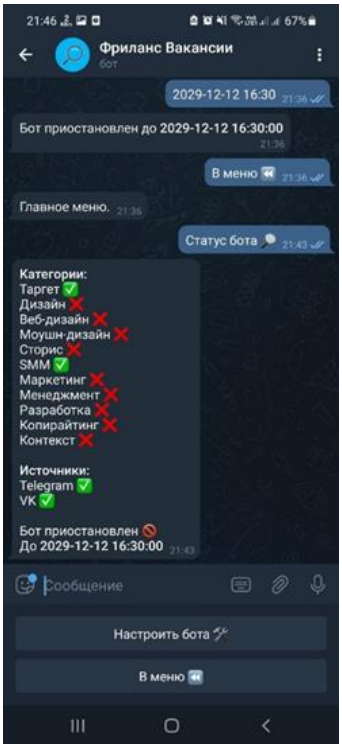

Рисунок 18 – Проверка статуса бота

Чтобы отправить сообщение в поддержку, пользователь должен нажать на кнопку «Написать в поддержку» в разделе «Помощь». Пример ответа реализации обратной связи представлен на рисунке 19.

Пользователь получает обработанные Клиентами VK и Telegram вакансии, в соответствие с выбранными категориями и источниками. Перед отправкой вакансий пользователям, делаются SQL-запросы, с помощью которых осуществляется проверка предпочтений фрилансеров. Например, если Клиент Telegram нашел вакансию для дизайнеров, запускаются SQL-запросы для сбора базы пользователей, у которых выбрана категория «дизайн» и источник «Telegram», затем вакансия отправляется всей базе пользователей. Перед отправкой вакансий также проверяются активность пользователя в боте и настроена ли приостановка бота. К каждой вакансии добавляется ссылка на ее источник. Пример реализации получения вакансий пользователями представлен на рисунке 20.

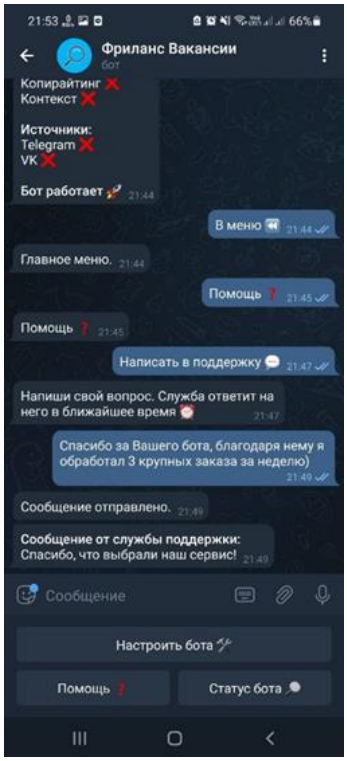

Рисунок 19 – Реализация обратной связи

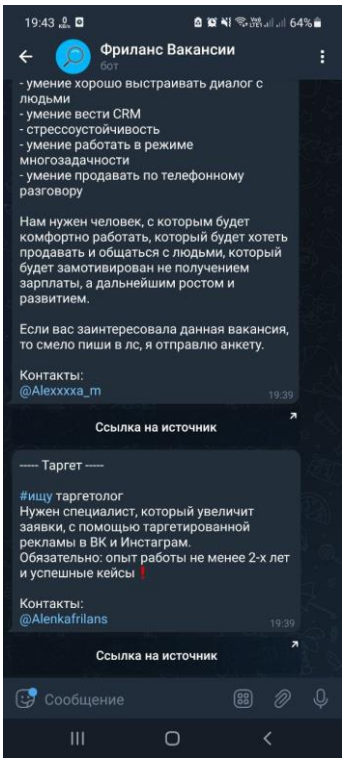

Рисунок 20 – Реализация получения вакансий

5 ТЕСТИРОВАНИЕ

Для тестирования Telegram-бота были выбраны следующие виды тестирования:

– юзабилити тестирование;

– функциональное тестирование.

5.1 Методология тестирования

Юзабилити тестирование – это метод оценки интерфейса со стороны удобства и эффективности его использования. Для проведение данного метода тестирования необходимо привлечь представителей целевой аудитории программного продукта.

Функциональное тестирование – это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определенных условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает.

5.2 Проведение процедуры тестирования

Для проведения юзабилити тестирования были привлечены независимые пользователи. Было определено удобство интерфейса Telegram-бота со стороны пользователей. В ходе юзабилити тестирования у пользователей не возникло трудностей при использовании ботом. Интерфейс оказался простым, понятным и удобным.

Для проведения функционального тестирования были проверены основные функции Telegram-бота:

– приостановка и возобновление отправки вакансий пользователям;

– выбор источников вакансий;

– выбор категорий вакансий;

– поиск вакансий с помощью ключевых слов;

– отправка вакансий пользователям;

– добавление ссылки на источник вакансии;

– реализация обратной связи.

В ходе проверки правильности работы функций выбора источников и категорий вакансий ошибок не возникло. Данная информация корректно отображалась в БД.

Для проверки работоспособности функции обратной связи с администратором были отправлены запросы в поддержку от нескольких пользователей. Присылаемые запросы приходили администратору Telegram-бота в виде текста сообщения запроса с добавлением id пользователя в Telegram, обратившегося в поддержку (рисунок 21). Ответы администратора приходили каждому пользователю в соответствии с их id в Telegram.

Основной функцией Telegram-бота является отправка вакансий в сфере фриланса из Telegram и VK пользователям. Чтобы отправляемые пользователям вакансии соответствовали своим категориям, должны быть правильно настроены обязательные и исключающие слова. В ходе функционального тестирования были отлажены ключевые слова для каждой категории вакансий.

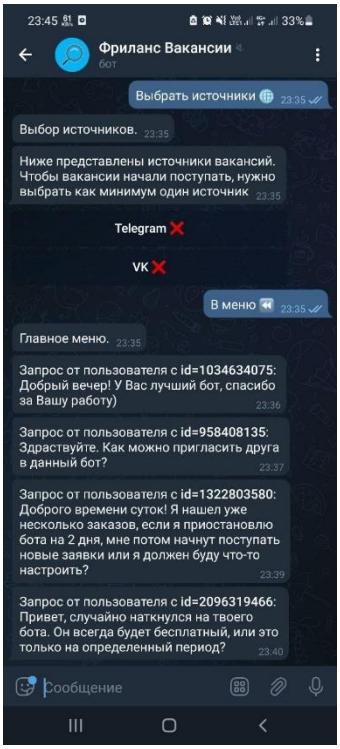

Рисунок 21 – Сообщения от пользователей, отправленные в поддержку

Основной функцией Telegram-бота является отправка вакансий в сфере фриланса из Telegram и VK пользователям. Чтобы отправляемые пользователям вакансии соответствовали своим категориям, должны быть правильно настроены обязательные и исключающие слова. В ходе функционального тестирования были отлажены ключевые слова для каждой категории вакансий.

Также было проведено тестирование на потребление ресурсов оперативной памяти программой. Программа запускалась в командной строке на ОС Windows 10. На рисунке 22 представлен снимок экрана диспетчера задач.

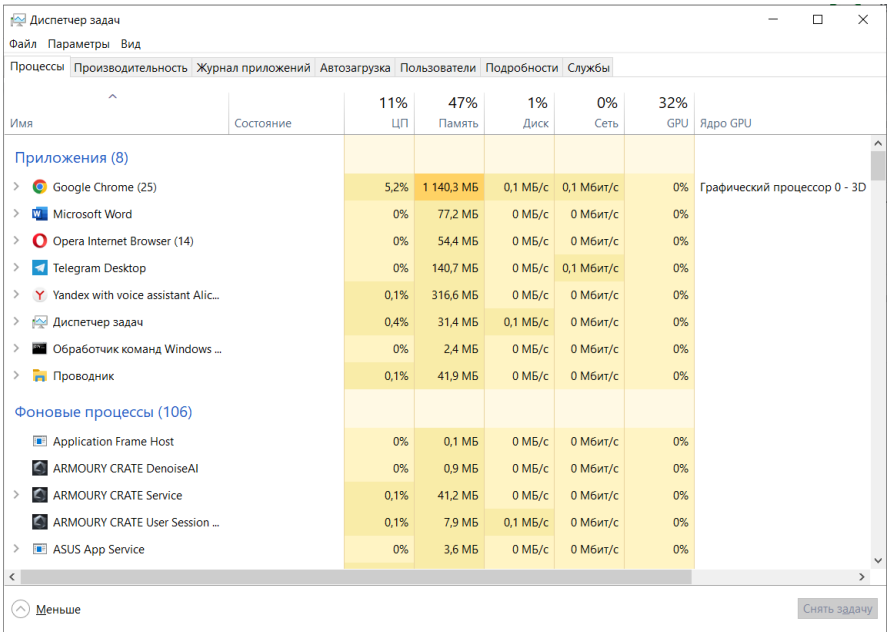

Рисунок 22 – Окно диспетчера задач

Из рисунка 22 видно, что программа потребляет около 2,5 мегабайт оперативной памяти, что является довольно низким показателем.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы был спроектирован и реализован Telegram-бот для поддержки поиска вакансий в сфере фриланса. Для достижения поставленной цели был проведен анализ существующих современных аналогов программных решений, осуществляющих сбор вакансий для мессенджера Telegram. Рассмотрены достоинства и недостатки современных средств и технологии, применяемых при разработке ботов. Для разработки программного продукта был выбран язык программирования Python. В качестве среды разработке был выбран PyCharm Community. Для хранения информации о пользователях и корректной работы Telegram-бота была спроектирована база данных в СУБД SQLite. Определен необходимый функционал Telegram-бота. Приведены этапы проектирования, разработка и процедура тестирования функционала Telegram-бота, публикующего актуальные вакансии для специалистов в сфере фриланса.

Реализованный Telegram-бот может быть полезным для фрилансеров различных направлений, поскольку он работает быстро и бесплатно. Специалистам в сфере удаленной работы не придется тратить время на поиск вакансий в мессенджере Telegram и социальной сети VK.

В дальнейшем планируется внедрить в Telegram-бота нейронные сети для более качественного отбора вакансий.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Telegram. – Текст : электронный // Википедия : свободная энциклопедия : [сайт]. – URL: https://ru.wikipedia.org/wiki/Telegram (дата обращения: 10.12.2021).

2. Мобильные операторы назвали самые популярные у россиян мессенджеры. – Текст : электронный // ТАСС : информационное агентство России : [официальный сайт]. – 26 января 2021. – URL: https://tass.ru/ekonomika/10544793 (дата обращения: 30.11.2021).

3. Боты: информация для разработчиков – Текст : электронный // Telegram : [сайт]. – URL: https://tlgrm.ru/docs/bots (дата обращения: 10.12.2021).

4. Лавка с лидами : Бот // Telegram : [бот в Telegram]. – URL: https://t.me/leadsshop_bot (дата обращения: 11.12.2021). – Режим доступа: доступ после авторизации.

5. Биржа Вакансий // Telegram : [бот в Telegram]. – URL: https://t.me/digitalvacancies2_bot (дата обращения: 11.12.2021). – Режим доступа: доступ после авторизации.

6. ЛидПоиск // Telegram : [бот в Telegram]. – URL: https://t.me/ leadpoiskbot (дата обращения: 11.12.2021). – Режим доступа: доступ после авторизации.

7. Новиков, И. Самые популярные языки программирования 2021 года. – Текст : электронный / И. Новиков // ХАБР : [сайт]. – 2 декабря 2021. – URL: https://habr.com/ru/company/skillfactory/blog/593025/ (дата обращения 18.01.2022).

8. Уткина, А. 6 языков программирования, которые будут популярны в 2022. – Текст : электронный / А. Уткина // Tproger : [сайт]. – 14 декабря 2021. – URL: https://tproger.ru/articles/jazyki-programmirovanija-2022/ (дата обращения 18.01.2022).

9. Введение в Python. – Текст : электронный // METANIT : [сайт]. – 5 января 2022. – URL: https://metanit.com/python/tutorial/1.1.php (дата обращения 20.01.2022).

10. Плюсы и минусы языков программирования. – Текст : электронный // teletype : [сайт]. – 17 марта 2020. – URL: https://teletype.in/@itvdn_it/zDfuiuHU (дата обращения 20.01.2022).

11. Что такое Python. – Текст : электронный // Медиа Нетологии : [сайт]. – 14 октября 2016. – URL: https://netology.ru/blog/python (дата обращения 20.01.2022).

12. Java. – Текст : электронный // Википедия : свободная энциклопедия : [сайт]. – URL: https://ru.wikipedia.org/wiki/Java (дата обращения: 21.01.2022).

13. В чем отличие JavaScript от Java? // Java : [официальный сайт]. – URL: https://www.java.com/ru/download/help/java_javascript_ru.html (дата обращения: 13.02.2022).

14. Язык программирования Javascript: особенности и преимущества. – Текст : электронный // vc.ru : [сайт]. – 10 августа 2020. – URL: https://vc.ru/hr/145461-yazyk-programmirovaniya-javascript-osobennosti-i-preimushchestva (дата обращения: 13.02.2022).

15. C++ (язык программирования). – Текст : электронный // Национальная библиотека им. Н. Э. Баумана : [сайт]. – 8 июня 2016. – URL: https://ru.bmstu.wiki/C%2B%2B_(язык_программирования) (дата обращения: 13.02.2022).

16. Язык C# и платформа .NET. – Текст : электронный // METANIT : [сайт]. – 8 ноября 2021. – URL: https://metanit.com/sharp/tutorial/1.1.php (дата обращения: 13.02.2022).

17. Интегрированная среда разработки. – Текст : электронный // Академик : [сайт]. – URL: https://dic.academic.ru/dic.nsf/ruwiki/940808 (дата обращения: 14.03.2022).

18. Скачать PyCharm // JetBrains: [официальный сайт]. – URL: https://www.jetbrains.com/ru-ru/pycharm/download/#section=windows (дата обращения: 15.03.2022).

19. Что такое реляционная база данных. – Текст : электронный // Oracle : [официальный сайт]. – URL: https://www.oracle.com/ru/database/what-is-a-relational-database/ (дата обращения: 30.03.2022).

20. Нереляционные данные и базы данных NoSQL. – Текст : электронный // Microsoft : [официальный сайт]. – URL: https://docs.microsoft.com/ru-ru/azure/architecture/data-guide/big-data/non-relational-data (дата обращения: 30.03.2022).

21. Как подружить Python и базы данных SQL. – Текст : электронный // Proglib : [сайт]. – URL: https://proglib.io/p/kak-podruzhit-python-i-bazy-dannyh-sql-podrobnoe-rukovodstvo-2020-02-27 (дата обращения: 30.03.2022).

22. SQLite – Введение. – Текст : электронный // Unetway : [сайт]. – URL: https://unetway.com/tutorial/sqlite (дата обращения: 30.03.2022).

23. Что такое PostgreSQL : [сайт]. – URL: https://ru.education-wiki.com/5154595-what-is-postgresql (дата обращения: 30.03.2022).

24. Подбор слов // Яндекс : [официальный сайт]. – URL: https://wordstat.yandex.ru/#!/?words=телеграм%20бот (дата обращения: 02.05.2022).

25. Telethon’s Documentation : [сайт]. – URL: https://docs.telethon.dev/en/latest/ (дата обращения: 30.11.2021).

26. Доусон, М. Программируем на Python / М. Доусон – Санкт-Петербург : Питер, 2014. – 416 с. : ил.

27. Asyncio : Asynchronous I/O. – Текст : электронный // Python 3.10.0 documentation : [сайт] / Python Software Foundation. – URL: https://docs.python.org/3/library/asyncio.html (дата обращения: 30.11.2021).

28. Чаллавала, Ш. MySQL 8 для больших данных / Ш. Чаллавала, Дж. Лакхатария, Ч. Мехта, К. Патель ; пер. с английского А. В. Логунова. – Москва : ДМК Пресс, 2018. – 226 с. : ил.

29. Aiogram’s Documentation : [сайт]. – URL: https://docs.aiogram.dev/en/latest/ (дата обращения: 30.11.2021).

30. Flowchart Maker & Online Diagram Software : [сайт]. – URL: https://app.diagrams.net/ (дата обращения: 30.11.2021).

31. Харрисон, М. Как устроен Python : гид для разработчиков, программистов и интересующихся / М. Харрисон. – Санкт-Петербург : Питер, 2019. – 272 с. : ил.

32. Васильев, А. Н. Python на примерах : практический курс по программированию / А. Н. Васильев. – Санкт-Петербург : Наука и Техника, 2016. – 432 с. : ил.

33. Смыслова, Л. В. Чат-бот как современное средство интернет-коммуникаций / Л. В. Смыслова. – Текст : электронный // Молодой ученый. – 2018. – № 9 (195). – С. 36-39. – URL: https://moluch.ru/archive/195/48623/ (дата обращения: 30.11.2021).

ПРИЛОЖЕНИЕ А

Класс с SQL-запросами

Листинг 1 – Класс для обработки SQL-запросов

```
import sqlite3


class Database:
    def __init__(self, db_file):
        self.connection = sqlite3.connect(db_file)
        self.cursor = self.connection.cursor()

    ''' ---- Get ---- '''

    def user_exists(self, user_id):
        with self.connection:
            result = self.cursor.execute("SELECT * FROM 'users' WHERE
user_id = ?", (user_id,)).fetchmany(1)
            return bool(len(result))

    def get_position(self, user_id):
        with self.connection:
            return self.cursor.execute("SELECT position FROM 'users' WHERE
user_id = ?", (user_id,)).fetchmany(1)

    def get_stopped_until(self, user_id):
        with self.connection:
            return self.cursor.execute("SELECT stopped_until FROM 'users'
WHERE user_id = ?", (user_id,)).fetchmany(1)

    def get_users(self):
        with self.connection:
            return self.cursor.execute("SELECT user_id, is_active FROM
'users'").fetchall()

    def category_exists(self, user_id, category):
        with self.connection:
            users_id = self.cursor.execute("SELECT id FROM 'users' WHERE
user_id = ?",
                                           (user_id,)
                                           ).fetchone()[0]
            categories_id = self.cursor.execute("SELECT id FROM
'categories' WHERE category = ?",
                                                (category,)
                                                ).fetchone()[0]
            result = self.cursor.execute("SELECT * FROM 'users-categories'
WHERE users_id = ? AND categories_id = ?",
                                         (users_id, categories_id,)
                                         ).fetchmany(1)
            return bool(len(result))

    def source_exists(self, user_id, source):
```

```
        with self.connection:
            users_id = self.cursor.execute("SELECT id FROM 'users' WHERE
user_id = ?",
                                           (user_id,)
                                           ).fetchone()[0]
            result = self.cursor.execute(f"SELECT is_{source} FROM
'sources' WHERE users_id = ? ",
                                         (users_id, )
                                         ).fetchone()[0]
            return bool(result)

    def get_categories(self, user_id):
        with self.connection:
            users_id = self.cursor.execute("SELECT id FROM 'users' WHERE
user_id = ?",
                                           (user_id,)
                                           ).fetchone()[0]
            return self.cursor.execute("SELECT categories_id FROM 'users-
categories' WHERE users_id = ?",
                                       (users_id,)
                                       ).fetchall()

    def get_users_with_category(self, category, source):
        with self.connection:
            categories_id = self.cursor.execute("SELECT id FROM
'categories' WHERE category = ?",
                                                (category,)
                                                ).fetchone()[0]

            users_id = self.cursor.execute(
                "SELECT user_id FROM 'users' WHERE stopped_until <=
datetime('now')"
                f"AND id IN(SELECT users_id FROM 'sources' WHERE
is_{source} = 1)"
                f"AND id IN(SELECT users_id FROM 'users-categories' WHERE
categories_id = ?)", (categories_id,)
            ).fetchall()
```

```
            return users_id

    ''' ---- End Get ---- '''

    ''' ---- Set ---- '''

    def set_active(self, user_id, active):
        with self.connection:
            return self.cursor.execute("UPDATE 'users' SET is_active = ?
WHERE user_id = ?", (active, user_id,))

    def set_position(self, user_id, position):
        with self.connection:
            return self.cursor.execute("UPDATE 'users' SET position = ?
WHERE user_id = ?", (position, user_id,))

    def set_stopped_until(self, user_id, stopped_until):
        with self.connection:
            return self.cursor.execute("UPDATE 'users' SET stopped_until =
? WHERE user_id = ?",
                                       (stopped_until, user_id,))

    def set_source(self, user_id, source, value):
        with self.connection:
            users_id = self.cursor.execute("SELECT id FROM 'users' WHERE
user_id = ?",
                                           (user_id,)
                                           ).fetchone()[0]
            return self.cursor.execute(f"UPDATE 'sources' SET is_{source}
= ? WHERE users_id = ?",
                                       (value, users_id,)
                                       )

    ''' ---- End Set ---- '''

    ''' ---- Add ---- '''
```

```
    def add_user(self, user_id):
        with self.connection:
            self.cursor.execute("INSERT INTO 'users' ('user_id') VALUES
(?)", (user_id,))

    def add_sources(self, user_id):
        with self.connection:
            users_id = self.cursor.execute("SELECT id FROM 'users' WHERE
user_id = ?",
                                           (user_id,)
                                           ).fetchone()[0]
            return self.cursor.execute("INSERT INTO 'sources' ('users_id')
VALUES (?)", (users_id,))

    def add_category(self, user_id, category):
        with self.connection:
            users_id = self.cursor.execute("SELECT id FROM 'users' WHERE
user_id = ?",
                                           (user_id,)
                                           ).fetchone()[0]
            categories_id = self.cursor.execute("SELECT id FROM
'categories' WHERE category = ?",
                                                (category,)
                                                ).fetchone()[0]
            return self.cursor.execute("INSERT INTO 'users-categories'
('users_id', 'categories_id') VALUES (?, ?)",
                                       (users_id, categories_id,)
                                       )

    ''' ---- End Add ---- '''

    ''' ---- Remove ---- '''

    def del_category(self, user_id, category):
        with self.connection:
            users_id = self.cursor.execute("SELECT id FROM 'users' WHERE
user_id = ?",
```

```
                                        (user_id,)
                                        ).fetchone()[0]
            categories_id = self.cursor.execute("SELECT id FROM
'categories' WHERE category = ?",
                                                (category,)
                                                ).fetchone()[0]
            return self.cursor.execute("DELETE FROM 'users-categories'
WHERE users_id = ? AND categories_id = ?",
                                       (users_id, categories_id,)
                                       )

    ''' ---- End Remove ---- '''
```

ПРИЛОЖЕНИЕ Б

Описание кнопок в боте

Листинг 2 – Описание кнопок в Telegram-боте

```
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton

btnAccept = KeyboardButton('Подтвердить 🔥')
btnToMenu = KeyboardButton('В меню ⏪')
btnBack = KeyboardButton('Назад ↩')


''' ---- Main Menu ---- '''
btnSettings = KeyboardButton('Настроить бота 🛠')
btnHelp = KeyboardButton('Помощь ❓')
btnBotStatus = KeyboardButton('Статус бота 🔎')
mainMenu = ReplyKeyboardMarkup(resize_keyboard=True).row(
    btnSettings
).add(btnHelp, btnBotStatus)


''' ---- Help ---- '''
btnGetSupport = KeyboardButton('Написать в поддержку 💬')
btnAbout = KeyboardButton('О боте 🤖')
helpWindow = ReplyKeyboardMarkup(resize_keyboard=True).row(
    btnGetSupport
).add(btnAbout, btnToMenu)


''' ---- Back and To Menu ---- '''
backToMenu = ReplyKeyboardMarkup(resize_keyboard=True).add(
    btnBack, btnToMenu
)


''' ---- Accept and Back and To Menu ---- '''
```

```
acceptBackToMenu = ReplyKeyboardMarkup(resize_keyboard=True).row(
    btnAccept
).add(btnBack, btnToMenu)


''' ---- Bot Status ---- '''
botStatus = ReplyKeyboardMarkup(resize_keyboard=True).row(
    btnSettings
).add(btnToMenu)


''' ---- Settings Categories Before Activated ---- '''
settingsCategoriesBA = ReplyKeyboardMarkup(resize_keyboard=True).row(
    btnAccept
).add(btnToMenu)


''' ---- Settings ---- '''
btnChooseCategories = KeyboardButton('Выбрать категории 🔄')
btnChooseSource = KeyboardButton('Выбрать источники 🌐')
btnStopBot = KeyboardButton('Приостановить бота 🚫')
btnResumeBot = KeyboardButton('Возобновить бота 🚀')

settingsBotStarted = ReplyKeyboardMarkup(resize_keyboard=True).row(
    btnChooseCategories
).row(
    btnChooseSource
).row(
    btnStopBot
).add(btnToMenu)

settingsBotStopped = ReplyKeyboardMarkup(resize_keyboard=True).row(
    btnChooseCategories
).row(
    btnChooseSource
).row(
```

```
    btnResumeBot
).add(btnToMenu)


''' ---- Resume Bot ---- '''
btnYes = KeyboardButton('Да ✅')
resetBot = ReplyKeyboardMarkup(resize_keyboard=True).row(
    btnYes
).add(btnBack, btnToMenu)
```

ПРИЛОЖЕНИЕ В

Методы считывания данных из файлов

Листинг 3 – Определение функций для чтения файлов

```
import csv


def read_string_session():
    f = open('Telegram Client files\StringSession.txt')
    string_session = f.read()
    f.close()
    return string_session


def read_admin_id():
    f = open('Bot files\Admin_id.txt')
    admin_id = f.read()
    f.close()
    return admin_id


def read_bot_token():
    f = open('Bot files\Bot_token.txt')
    bot_token = f.read()
    f.close()
    return bot_token


def read_vk_client_token():
    f = open('VK Client files\Access_token.txt')
    vk_client_token = f.read()
    f.close()
    return vk_client_token


def vk_channels():
```

```
        with open('VK Client files\channels.txt') as f:
            channels = f.read().splitlines()
            channels = [int(channel_id) for channel_id in channels]
            return channels


    def telegram_channels():
        with open('Telegram Client files\channels.txt') as f:
            channels = f.read().splitlines()
            channels = [int(channel_id) for channel_id in channels]
            return channels


    def telegram_chats():
        with open('Telegram Client files\chats.txt') as f:
            chats = f.read().splitlines()
            chats = [int(chat_id) for chat_id in chats]
            return chats


    def read_words(filename, i, words):
        words.append([])
        with open(filename, newline='') as f:
            reader = csv.reader(f)
            for word in reader:
                words[i] += word


    def telegram_pass_words():
        words = []
        words.append([])

        read_words('Telegram  Client  files\Pass  words\_1_Target_ads.csv',  1,
words)
        read_words('Telegram Client files\Pass words\_2_Design.csv', 2, words)
        read_words('Telegram  Client  files\Pass  words\_3_Web_design.csv',  3,
words)
```

```
        read_words('Telegram Client files\Pass words\_4_Motion_design.csv', 4,
words)
        read_words('Telegram Client files\Pass words\_5_Stories.csv', 5, words)
        read_words('Telegram Client files\Pass words\_6_SMM.csv', 6, words)
        read_words('Telegram  Client  files\Pass  words\_7_Marketing.csv',  7,
words)
        read_words('Telegram  Client  files\Pass  words\_8_Management.csv',  8,
words)
        read_words('Telegram  Client  files\Pass  words\_9_Development.csv',  9,
words)
        read_words('Telegram Client files\Pass words\_10_Copywriting.csv', 10,
words)
        read_words('Telegram Client files\Pass words\_11_Context_ads.csv', 11,
words)

        return words


    def telegram_reject_words_chat():
        words = []
        with open('Telegram Client files\Reject words\_reject_words_chat.csv',
newline='') as f:
            reader = csv.reader(f)
            for word in reader:
                words += word
        return words


    def telegram_reject_words_channel():
        words = []
        with           open('Telegram           Client           files\Reject
words\_reject_words_channel.csv', newline='') as f:
            reader = csv.reader(f)
            for word in reader:
                words += word
        return words
```

ПРИЛОЖЕНИЕ Г

Код программы Telegram-бота и клиентов VK и Telegram

Листинг 4 – Импорт модулей и библиотек

```
import datetime
import sys
import threading

from telethon import TelegramClient, events
from telethon.sessions import StringSession

import logging
from aiogram import Bot, Dispatcher, executor, types

import buttons as nav
from db_users import Database
import reader

import vk_api

sys.path.insert(1, './Telegram Client files')
from settings import (API_ID, API_HASH)
```

Листинг 5 – Меню Telegram-бота

```
@dp.message_handler()
async def menu(message: types.Message):
    if message.from_user.id == telegramClientID:
        text = message.html_text.rsplit('\n', 3)    # message; category;
source; link

        linkSource = types.InlineKeyboardMarkup(row_width=1)
        btnLink = types.InlineKeyboardButton(text='Ссылка на источник',
url=text[3])
        linkSource.insert(btnLink)

        for user in db_users.get_users_with_category(str(text[1]),
str(text[2])):
```

```
                try:
                    await bot.send_message(user[0], f'----- {text[1]} -----
\n\n' + text[0],
                                           parse_mode=types.ParseMode.HTML,
                                           reply_markup=linkSource,
                                           disable_web_page_preview=True
                                           )
                except:
                    db_users.set_active(user[0], 0)

        elif message.chat.type == 'private':

            db_users.set_active(message.from_user.id, 1)
            position = db_users.get_position(message.from_user.id)[0][0]

            ##################################################
            if message.text == 'Настроить бота 🛠' and (position == 'Main menu'
or position == 'Bot status'):

                if db_users.get_stopped_until(message.from_user.id)[0][0] ==
'2100-01-01 00:00:00':
                    await bot.send_message(
                        message.from_user.id,
                        'Настроить бота 🛠',
                        reply_markup=nav.settingsCategoriesBA
                    )
                    db_users.set_position(message.from_user.id, 'First Choose
Categories')
                    await bot.send_message(
                        message.from_user.id,
                        'Ниже представлены категории вакансий. Чтобы вакансии
начали поступать, нужно выбрать как минимум одну категорию',
                        reply_markup=categories_keyboard(message.from_user.id)
                    )

                else:
```

```
                    result = compare_time(str(datetime.datetime.now())[:19],

db_users.get_stopped_until(message.from_user.id)[0][0]
                                          )
                    if result:
                        await bot.send_message(
                            message.from_user.id,
                            'Настроить бота 🛠',
                            reply_markup=nav.settingsBotStarted
                        )
                        db_users.set_position(message.from_user.id, 'Settings')

                    else:
                        await bot.send_message(
                            message.from_user.id,
                            'Настроить бота 🛠',
                            reply_markup=nav.settingsBotStopped
                        )
                        db_users.set_position(message.from_user.id, 'Settings')

            ##################################################
            elif message.text == 'Подтвердить 🔥':
                if position == 'First Choose Categories':
                    await bot.send_message(
                        message.from_user.id,
                        'Выбор источников.',
                        reply_markup=nav.acceptBackToMenu
                    )
                    db_users.set_position(message.from_user.id,  'First  Choose
Souces')
                    await bot.send_message(
                        message.from_user.id,
                        'Ниже представлены источники вакансий. Чтобы вакансии
начали поступать, нужно выбрать как минимум один источник',
                        reply_markup=sources_keyboard(message.from_user.id)
                    )
```

```
                elif position == 'First Choose Souces':
                    await bot.send_message(
                        message.from_user.id,
                        'Бот настроен, хороших лидов!',
                        reply_markup=nav.mainMenu
                    )
                    db_users.set_stopped_until(message.from_user.id,
datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S'))
                    db_users.set_position(message.from_user.id, 'Main menu')

                else:
                    await bot.send_message(
                        message.from_user.id,
                        'Я не знаю, что ответить.'
                    )

            ##################################################
            elif message.text == 'Статус бота 🔎' and position == 'Main menu':
                if True:
                    await bot.send_message(
                        message.from_user.id,
                        status_text(message.from_user.id),
                        parse_mode=types.ParseMode.HTML,
                        reply_markup=nav.botStatus
                    )
                db_users.set_position(message.from_user.id, 'Bot status')

            ##################################################
            elif message.text == 'Помощь ❓' and position == 'Main menu':
                await bot.send_message(
                    message.from_user.id,
                    'Помощь ❓',
                    reply_markup=nav.helpWindow
                )
                db_users.set_position(message.from_user.id, 'Help')
```

```
            ##################################################
            elif message.text == 'Написать в поддержку 💬' and position ==
'Help':
                await bot.send_message(
                    message.from_user.id,
                    'Напиши свой вопрос. Служба ответит на него в ближайшее
время ⏲',
                    reply_markup=nav.backToMenu
                )
                db_users.set_position(message.from_user.id, 'Get support')

            ##################################################
            elif message.text == 'О боте 🤘' and position == 'Help':
                await bot.send_message(
                    message.from_user.id,
                    'Тут будет информация о боте!',
                    reply_markup=nav.backToMenu
                )
                db_users.set_position(message.from_user.id, 'Help2')

            ##################################################
            elif message.text == 'Выбрать категории 🔄' and position ==
'Settings':
                await bot.send_message(
                    message.from_user.id,
                    'Выбор категорий.',
                    reply_markup=nav.backToMenu
                )
                db_users.set_position(message.from_user.id, 'Settings2')
                await bot.send_message(
                    message.from_user.id,
                    'Ниже представлены категории вакансий. Чтобы вакансии начали
поступать, нужно выбрать как минимум одну категорию',
                    reply_markup=categories_keyboard(message.from_user.id)
                )
```

```
            ##################################################
            elif message.text == 'Выбрать источники 🌐' and position ==
'Settings':
                await bot.send_message(
                    message.from_user.id,
                    'Выбор источников.',
                    reply_markup=nav.backToMenu
                )
                db_users.set_position(message.from_user.id, 'Settings2')
                await bot.send_message(
                    message.from_user.id,
                    'Ниже представлены источники вакансий. Чтобы вакансии начали
поступать, нужно выбрать как минимум один источник',
                    reply_markup=sources_keyboard(message.from_user.id)
                )

            ##################################################
            elif message.text == 'Приостановить бота 🚫' and position ==
'Settings':
                result = compare_time(str(datetime.datetime.now())[:19],

db_users.get_stopped_until(message.from_user.id)[0][0]
                                      )
                if result:
                    await bot.send_message(
                        message.from_user.id,
                        'Введи дату и время в формате <b>YYYY-MM-DD
hh:mm</b>.\n\n'
                        'Дата и время не могут быть меньше текущей, а год нельзя
указать больше 2030.\n\n'
                        'Пример:\n2022-11-25 15:30 -> 15:30 25 ноября 2022г.',
                        parse_mode=types.ParseMode.HTML,
                        reply_markup=nav.backToMenu
                    )
                    db_users.set_position(message.from_user.id, 'Stop bot')
                else:
```

```
                    await bot.send_message(
                        message.from_user.id,
                        'Я не знаю, что ответить.'
                    )

            ##################################################
            elif message.text == 'Возобновить бота 🚀' and position ==
'Settings':
                result = compare_time(str(datetime.datetime.now())[:19],

db_users.get_stopped_until(message.from_user.id)[0][0]
                                      )
                if not result:
                    await bot.send_message(
                        message.from_user.id,
                        'Возобновить бота?',
                        reply_markup=nav.resetBot
                    )
                    db_users.set_position(message.from_user.id, 'Settings3')

                else:
                    await bot.send_message(
                        message.from_user.id,
                        'Я не знаю, что ответить.'
                    )

            ##################################################
            elif message.text == 'Да ✅' and position == 'Settings3':
                db_users.set_stopped_until(message.from_user.id,
datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S'))
                await bot.send_message(
                    message.from_user.id,
                    'Работа бота вообновлена!',
                    reply_markup=nav.settingsBotStarted
                )
                db_users.set_position(message.from_user.id, 'Settings')
```

```
            ##################################################
            elif message.text == 'В меню ⏪':
                await bot.send_message(
                    message.from_user.id,
                    'Главное меню.',
                    reply_markup=nav.mainMenu
                )
                db_users.set_position(message.from_user.id, 'Main menu')

            ##################################################
            elif message.text == 'Назад ↩':
                if position == 'Help2' or position == 'Get support':
                    await bot.send_message(
                        message.from_user.id,
                        'Помощь ❓',
                        reply_markup=nav.helpWindow
                    )
                    db_users.set_position(message.from_user.id, 'Help')

                elif position == 'First Choose Souces':
                    await bot.send_message(
                        message.from_user.id,
                        'Ты в меню настройки бота! Бот еще не активировался.
Сейчас будет выбор категорий.',
                        reply_markup=nav.settingsCategoriesBA
                    )
                    db_users.set_position(message.from_user.id, 'First Choose
Categories')
                    await bot.send_message(
                        message.from_user.id,
                        'Здесь ты видишь категории вакансий. Подробнее про
каждую категорию будет описано немного позже',
                        reply_markup=categories_keyboard(message.from_user.id)
                    )
```

```
                elif position == 'Settings2' or position == 'Settings3' or
position == 'Stop bot':
                    result = compare_time(str(datetime.datetime.now())[:19],

db_users.get_stopped_until(message.from_user.id)[0][0]
                                          )
                    if result:
                        await bot.send_message(
                            message.from_user.id,
                            'Настроить бота 🛠',
                            reply_markup=nav.settingsBotStarted
                        )
                        db_users.set_position(message.from_user.id, 'Settings')

                    else:
                        await bot.send_message(
                            message.from_user.id,
                            'Настроить бота 🛠',
                            reply_markup=nav.settingsBotStopped
                        )
                        db_users.set_position(message.from_user.id, 'Settings')

                else:
                    await bot.send_message(
                        message.from_user.id,
                        'Я не знаю, что ответить.'
                    )

            ################################################
            elif position == 'Stop bot':
                if check_date(message.text):
                    db_users.set_stopped_until(message.from_user.id,
(str(message.text) + ':00'))
                    await bot.send_message(
                        message.from_user.id,
                        f'Бот приостановлен до <b>{message.text}:00</b>',
```

```
                    parse_mode=types.ParseMode.HTML,
                    reply_markup=nav.settingsBotStopped
                )
                db_users.set_position(message.from_user.id, 'Settings')
            else:
                await bot.send_message(
                    message.from_user.id,
                    f'Некорректное время!'
                )

        elif position == 'Get support':
            await bot.send_message(
                message.from_user.id,
                f'Сообщение отправлено.',
                reply_markup=nav.mainMenu
            )
            db_users.set_position(message.from_user.id, 'Main menu')

            text = f"Запрос от пользователя с
<b>id={message.from_user.id}</b>:\n"
            await bot.send_message(admin_id, text + message.text,
                                   parse_mode=types.ParseMode.HTML,
                                   )

        ##################################################
        else:
            await bot.send_message(
                message.from_user.id,
                'Я не знаю, что ответить.'
            )
```

Листинг 6 – Команда для ответа на сообщение в службе поддержки

```
@dp.message_handler(commands=['answer'])
async def sendall(message: types.Message):
    if message.chat.type == 'private':
        if message.from_user.id == admin_id:
            text = message.text[8:]
```

```
                user_id = text.split(":")[0]
                text = text.split("\n")[1]
                text = "<b>Сообщение от службы поддержки:</b>\n" + text

                try:
                    await bot.send_message(user_id, text,
                                           parse_mode=types.ParseMode.HTML,
                                           )
                    await   bot.send_message(message.from_user.id,   "Сообщение
отправлено.")

                except:
                    await  bot.send_message(message.from_user.id,  "Пользователь
неактивен.")
```

Листинг 7 – Методы для выбора категорий и источников вакансий

```
    async def update_sources_keyboard(message: types.Message, user_id):
        await message.edit_text(
            "Ниже представлены источники вакансий. Чтобы вакансии начали
поступать, нужно выбрать как минимум один источник",
            reply_markup=sources_keyboard(user_id))


    def sources_keyboard(user_id):
        emg = ['❌', '❌']
        if db_users.source_exists(user_id, "Telegram"):
            emg[0] = '✅'

        if db_users.source_exists(user_id, "VK"):
            emg[1] = '✅'

        buttons = [
            types.InlineKeyboardButton(text="Telegram      "      +      emg[0],
callback_data="src_Telegram"),
            types.InlineKeyboardButton(text="VK        "        +        emg[1],
callback_data="src_VK"),
        ]
```

```
        keyboard = types.InlineKeyboardMarkup(row_width=1)
        keyboard.add(*buttons)
        return keyboard


    async def update_categories_keyboard(message: types.Message, user_id):
        await message.edit_text(
            "Ниже представлены категории вакансий. Чтобы вакансии начали
поступать, нужно выбрать как минимум одну категорию",
            reply_markup=categories_keyboard(user_id))


    def categories_keyboard(user_id):
        emg = ['', '❌', '❌', '❌', '❌', '❌', '❌', '❌', '❌', '❌', '❌',
'❌']
        categories = db_users.get_categories(user_id)
        for category in categories:
            emg[category[0]] = '✅'

        buttons = [
            types.InlineKeyboardButton(text="Таргет      "      +      emg[1],
callback_data="cat_Таргет"),
            types.InlineKeyboardButton(text="Дизайн      "      +      emg[2],
callback_data="cat_Дизайн"),
            types.InlineKeyboardButton(text="Веб-дизайн     "     +     emg[3],
callback_data="cat_Веб-дизайн"),
            types.InlineKeyboardButton(text="Моушн-дизайн     "     +     emg[4],
callback_data="cat_Моушн-дизайн"),
            types.InlineKeyboardButton(text="Сторис      "      +      emg[5],
callback_data="cat_Сторис"),
            types.InlineKeyboardButton(text="SMM       "       +       emg[6],
callback_data="cat_SMM"),
            types.InlineKeyboardButton(text="Маркетинг     "     +     emg[7],
callback_data="cat_Маркетинг"),
```

```
            types.InlineKeyboardButton(text="Менеджмент     "     +     emg[8],
callback_data="cat_Менеджмент"),
            types.InlineKeyboardButton(text="Разработка     "     +     emg[9],
callback_data="cat_Разработка"),
            types.InlineKeyboardButton(text="Копирайтинг    "    +    emg[10],
callback_data="cat_Копирайтинг"),
            types.InlineKeyboardButton(text="Контекст     "     +     emg[11],
callback_data="cat_Контекст")
        ]

        keyboard = types.InlineKeyboardMarkup(row_width=1)
        keyboard.add(*buttons)
        return keyboard

    @dp.callback_query_handler(text_contains="cat_")
    async def callbacks_category(call: types.CallbackQuery):
        category = call.data.split("_")[1]

        if db_users.category_exists(call.from_user.id, str(category)):
            db_users.del_category(call.from_user.id, str(category))
        else:
            db_users.add_category(call.from_user.id, str(category))

        await update_categories_keyboard(call.message, call.from_user.id)


    @dp.callback_query_handler(text_contains="src_")
    async def callbacks_source(call: types.CallbackQuery):
        source = call.data.split("_")[1]

        if db_users.source_exists(call.from_user.id, str(source)):
            db_users.set_source(call.from_user.id, str(source), 0)
        else:
            db_users.set_source(call.from_user.id, str(source), 1)

        await update_sources_keyboard(call.message, call.from_user.id)
```

Листинг 8 – Проверка статуса бота

```
def compare_time(current_time, user_time):
    if int(current_time[:4]) > int(user_time[:4]):
        return 1
    elif int(current_time[:4]) < int(user_time[:4]):
        return 0
    if int(current_time[:4]) == int(user_time[:4]) and
int(current_time[5:7]) > int(user_time[5:7]):
        return 1
    elif int(current_time[:4]) == int(user_time[:4]) and
int(current_time[5:7]) < int(user_time[5:7]):
        return 0
    if int(current_time[5:7]) == int(user_time[5:7]) and
int(current_time[8:10]) > int(user_time[8:10]):
        return 1
    elif int(current_time[5:7]) == int(user_time[5:7]) and
int(current_time[8:10]) < int(user_time[8:10]):
        return 0
    if int(current_time[8:10]) == int(user_time[8:10]) and
int(current_time[11:13]) > int(user_time[11:13]):
        return 1
    elif int(current_time[8:10]) == int(user_time[8:10]) and
int(current_time[11:13]) < int(user_time[11:13]):
        return 0
    if int(current_time[11:13]) == int(user_time[11:13]) and
int(current_time[14:16]) >= int(user_time[14:16]):
        return 1
    return 0


def status_text(user_id):
    cat = ['', '❌', '❌', '❌', '❌', '❌', '❌', '❌', '❌', '❌', '❌',
'❌']
    categories = db_users.get_categories(user_id)
    for category in categories:
        cat[category[0]] = '✅'
```

```
text = "<b>Категории:</b>" + \
       "\nТаргет " + cat[1] + \
       "\nДизайн " + cat[2] + \
       "\nВеб-дизайн " + cat[3] + \
       "\nМоушн-дизайн " + cat[4] + \
       "\nСторис " + cat[5] + \
       "\nSMM " + cat[6] + \
       "\nМаркетинг " + cat[7] + \
       "\nМенеджмент " + cat[8] + \
       "\nРазработка " + cat[9] + \
       "\nКопирайтинг " + cat[10] + \
       "\nКонтекст " + cat[11]

src = ['❌', '❌']
if db_users.source_exists(user_id, "Telegram"):
    src[0] = '✅'

if db_users.source_exists(user_id, "VK"):
    src[1] = '✅'

text = text + "\n\n<b>Источники:</b>" + \
       "\nTelegram " + src[0] + \
       "\nVK " + src[1]

result = compare_time(str(datetime.datetime.now())[:19],
                      db_users.get_stopped_until(user_id)[0][0]
                      )
if result:
    text = text + "\n\n<b>Бот работает</b> 🚀"
else:
    tmp = db_users.get_stopped_until(user_id)[0][0]
    text = text + f"\n\nБот приостановлен 🚫\nДо <b>{tmp}</b>"

return text
```

Листинг 9 – Обработка вакансий Telegram Клиентом

```
SESSION_STRING = reader.read_string_session()

telegramClient = TelegramClient(StringSession(SESSION_STRING), API_ID,
API_HASH)

TELEGRAM_CHANNELS = reader.telegram_channels()
TELEGRAM_CHATS = reader.telegram_chats()

pass_words = reader.telegram_pass_words()
reject_words_channel = reader.telegram_reject_words_channel()
reject_words_chat = reader.telegram_reject_words_chat()


def apply_message(msg, words):
    for key_word in words:
        if key_word in msg.lower():
            return 1
    return 0


@telegramClient.on(events.NewMessage(TELEGRAM_CHANNELS,
blacklist_chats=False))
async def message_from_telegram(msg):
    while True:
        if apply_message(msg.message.message, reject_words_channel):
            break

        if apply_message(msg.message.message, pass_words[6]):  # SMM
            category = 'SMM'
        elif apply_message(msg.message.message, pass_words[8]):      #
Менеджмент
            category = 'Менеджмент'
        elif apply_message(msg.message.message, pass_words[7]):  # Маркетинг
            category = 'Маркетинг'
        elif apply_message(msg.message.message, pass_words[9]):      #
Разработка
            category = 'Разработка'
```

```
            elif apply_message(msg.message.message, pass_words[1]):  # Таргет
                category = 'Таргет'
            elif apply_message(msg.message.message, pass_words[11]):  # Контекст
                category = 'Контекст'
            elif   apply_message(msg.message.message,   pass_words[10]):      #
Копирайтинг
                category = 'Копирайтинг'
            elif apply_message(msg.message.message, pass_words[5]):  # Сторис
                category = 'Сторис'
            elif  apply_message(msg.message.message,  pass_words[4]):   #  Моушн-
дизайн
                category = 'Моушн-дизайн'
            elif  apply_message(msg.message.message,  pass_words[3]):    #  Веб-
дизайн
                category = 'Веб-дизайн'
            elif apply_message(msg.message.message, pass_words[2]):  # Дизайн
                category = 'Дизайн'
            else:
                break

            entity  =  await  telegramClient.get_entity(msg.message.peer_id)   #
Ссылка на источник

            msg.message.message  =  msg.message.message  +  '\n'  +  category  +
'\nTelegram\n' + f't.me/{entity.username}'

            await telegramClient.send_message(bot_username, msg.message)
            break


    @telegramClient.on(events.NewMessage(TELEGRAM_CHATS,
blacklist_chats=False))
    async def message_from_telegram(msg):
        while True:
            sender = await msg.get_sender()
            if str(sender) == "None":
                break
```

```
            elif str(sender.username) != "None":
                print(sender.username)
                if apply_message(msg.message.message, reject_words_chat):
                    break

                if apply_message(msg.message.message, pass_words[6]):  # SMM
                    category = 'SMM'
                elif  apply_message(msg.message.message,  pass_words[8]):    #
Менеджмент
                    category = 'Менеджмент'
                elif  apply_message(msg.message.message,  pass_words[7]):    #
Маркетинг
                    category = 'Маркетинг'
                elif  apply_message(msg.message.message,  pass_words[9]):    #
Разработка
                    category = 'Разработка'
                elif  apply_message(msg.message.message,  pass_words[1]):    #
Таргет
                    category = 'Таргет'
                elif  apply_message(msg.message.message,  pass_words[11]):    #
Контекст
                    category = 'Контекст'
                elif  apply_message(msg.message.message,  pass_words[10]):    #
Копирайтинг
                    category = 'Копирайтинг'
                elif  apply_message(msg.message.message,  pass_words[5]):    #
Сторис
                    category = 'Сторис'
                elif  apply_message(msg.message.message,  pass_words[4]):    #
Моушн-дизайн
                    category = 'Моушн-дизайн'
                elif apply_message(msg.message.message, pass_words[3]):  # Веб-
дизайн
                    category = 'Веб-дизайн'
                elif  apply_message(msg.message.message,  pass_words[2]):    #
Дизайн
                    category = 'Дизайн'
```

```
                else:
                    break

                entity = await telegramClient.get_entity(msg.message.peer_id)  #
Ссылка на источник

                msg.message.message        +=        '\n\nКонтакты:\n@'        +
str(sender.username)
                msg.message.message = msg.message.message + '\n' + category +
'\nTelegram\n' + f't.me/{entity.username}'

                await telegramClient.send_message(bot_username, msg.message)
            break
```

Листинг 10 – Обработка вакансий VK Клиентом

```
import vk_api

token = reader.read_vk_client_token()

session = vk_api.VkApi(token=token)
vk = session.get_api()
vk_channels = reader.vk_channels()

id_array = []


def get_wall():
    i = 0
    id_array[0] = 0
    for channel in vk_channels:
        post = vk.wall.get(owner_id=channel, count=2)
        post = post["items"]

        if int(post[0]['id']) > int(post[1]['id']):
            post = post[0]
        else:
            post = post[1]
```

```
                if id_array[i] >= int(post['id']):
                    continue
                else:
                    id_array[i] = int(post['id'])

                message = post["text"]  # текст вакансии
                while True:
                    if apply_message(message, reject_words_channel):
                        break

                    if apply_message(message, pass_words[6]):  # SMM
                        category = 'SMM'
                    elif apply_message(message, pass_words[8]):  # Менеджмент
                        category = 'Менеджмент'
                    elif apply_message(message, pass_words[7]):  # Маркетинг
                        category = 'Маркетинг'
                    elif apply_message(message, pass_words[9]):  # Разработка
                        category = 'Разработка'
                    elif apply_message(message, pass_words[1]):  # Таргет
                        category = 'Таргет'
                    elif apply_message(message, pass_words[11]):  # Контекст
                        category = 'Контекст'
                    elif apply_message(message, pass_words[10]):  # Копирайтинг
                        category = 'Копирайтинг'
                    elif apply_message(message, pass_words[5]):  # Сторис
                        category = 'Сторис'
                    elif apply_message(message, pass_words[4]):  # Моушн-дизайн
                        category = 'Моушн-дизайн'
                    elif apply_message(message, pass_words[3]):  # Веб-дизайн
                        category = 'Веб-дизайн'
                    elif apply_message(message, pass_words[2]):  # Дизайн
                        category = 'Дизайн'
                    else:
                        break

                    entity = f'vk.com/public{str(post["from_id"])[1:]}'  # ссылка
на источник
```

```
                try:
                    from_user = f'vk.me/id{post["signer_id"]}'       # контакты
пользователя
                    message = message + '\n\nКонтакты:\n' + from_user
                except KeyError:
                    pass

                await send_vacancy(category, message, "VK", entity)
                break


    async def send_vacancy(category, message, source, entity):
        linkSource = types.InlineKeyboardMarkup(row_width=1)
        btnLink   =   types.InlineKeyboardButton(text='Ссылка   на   источник',
url=entity)
        linkSource.insert(btnLink)

        for     user     in     db_users.get_users_with_category(str(category),
str(source)):
            try:
                await bot.send_message(user[0], f'----- {category} -----\n\n' +
message,
                                       parse_mode=types.ParseMode.HTML,
                                       reply_markup=linkSource,
                                       disable_web_page_preview=True
                                       )
            except:
                db_users.set_active(user[0], 0)
```