МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра Систем информационной безопасности

«ДОПУЩЕН К ЗАЩИТЕ»
Зав. кафедрой к.п.н., доцент Р. Ш. Бердибаев
_____ «_____» _____2019 г.
(подпись)

# ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка системы автоматизированного процесса сбора персональных данных
Специальность Системы информационной безопасности
Выполнил Турсумбаев Данил Рустамович Группа СИБ-15-3
Научный руководитель Сатимова Елена Григорьевна

Консультант:
по экономической части:
к.э.н. профессор Ж. Т. Ирендаева
(ученая степень, звание, Ф.И.О)
(подпись) « 29 » апреля 2019 г.

по безопасности жизнедеятельности:
д.т.н. ст. пр. Ш. Ш. Бекбассаров
(ученая степень, звание, Ф.И.О)
(подпись) « 27 » апреля 2019 г.

по применению вычислительной техники:
к.т.н, доцент Е.Т. Сатимова
(ученая степень, звание, Ф.И.О)
(подпись) « 10 » мая 2019 г.

Нормоконтролер:
м.т.н., ст. преподаватель Н.Т. Искакова
(ученая степень, звание, Ф.И.О)
(подпись) « 29 » апреля 2019 г.

Рецензент:
к.т.н. ассоц. проф. Е.Ж. Ныгмановаева
(ученая степень, звание, Ф.И.О)
(подпись) « 05 » мая 2019 г.

Алматы 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Институт систем управления и информационных технологий

Кафедра Систем информационной безопасности

Специальность Системы информационной безопасности

**ЗАДАНИЕ**
на выполнение дипломного проекта

Студенту Турсумбаеву Данилу Рустамовичу

Тема проекта Разработка системы автоматизированного процесса сбора персональных данных

Утверждена приказом по университету №_____ от «____» _____2019 г.

Срок сдачи законченного проекта «_____» _____2019 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): проект подразумевает разработку Telegram бота, позволяющего в короткие сроки с разрешения искомого человека собрать персональные данные о нём, на основе минимальных идентификаторов. В качестве исходных данных бот будет принимать следующие идентификаторы искомого человека: город проживания, фамилию и имя. Помимо обширного функционала бота, необходимо реализовать надежную систему защиты. В качестве системы защиты будет использоваться: логирования запросов пользователя, разграничение прав доступа на основе уникального идентификатора каждого пользователя Telegram и фильтрация всех данных вводимых пользователем для предотвращения реализации атак типа: SQL injection, RCE, LFI и так далее.

Перечень вопросов, подлежащих разработке в дипломном проекте, или краткое содержание дипломного проекта: дипломный проект включает в себя 5 глав, разделенных на подглавы, каждая из которых освещает определенную тематику, используемую при разработке Telegram ботов.

В первой главе дипломного проекта представлена общая информация по ботам: типы ботов, функционал ботов, плюсы и минусы, надежность и безопасность ботов, а так же некоторые хитрости, которые используются при разработке ботов.

Во второй главе дипломного проекта представлена общая структура Telegram бота PSinfo: модули из которых состоит бот и принципы взаимодействия модулей между друг другом.

В третей главе подробно описывается функционал каждого модуля в отдельности.

В четвертой главе приводится технико-экономическое обоснование, показывающее актуальность разработки бота PSinfo с финансовой точки зрения.

В пятой главе рассматриваются необходимые условия для комфортной разработки программного обеспечения.

Перечень графического материала (с точным указанием обязательных чертежей):

1 блок-схема программного обеспечения;
2 скрины модулей бота;
3 скрин с результатами работы бота;
4 скрины исходного кода бота;
5 скрины с примером эксплуатации бота.

Основная рекомендуемая литература:

1 Аманжолова К. Б., Алибаева С. А. Экономика предприятии телекоммуникации: Учебное пособие. - Алматы: АИЭС, 2003

2 Голубицкая Е. А., Жигульская Г. М. Экономика связи. – М. Радио и связь, 2000

3 Самоучитель Python URL: https://pythonworld.ru/samouchitel-python

4 Разработка Telegram бота URL: https://habr.com/ru/post/262247/

5 Что такое боты и как они работают URL: https://ru.epicstars.com/botyi-telegram/

6 Нормы микроклимата URL: http://adilet.zan.kz/rus/docs /V050003789

Конструкции по проекту с указанием относящихся к ним разделов проекта

| Раздел | Консультант | Сроки | Подпись |
|---|---|---|---|
| Экономический | к.э.н., проф. Ж.Т. Аренбаева | 04.04.2019 | |
| Вычислительной техники | к.т.н., доцент Е.Т. Сатпаева | 10.05.2019 | |
| Безопасности жизнедеятельности | д.т.н. ст.пр. Ш.Ш. Бекбасаров | 10.04.2019 | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# График
## подготовки дипломного проекта

| Наименование разделов, перечень разрабатываемых вопросов | Сроки представления научному руководителю | Примечание |
|---|---|---|
| Сбор информации о существующих ботах | 10.01.2019 - 20.01.2019 | |
| Изучение преимуществ и недостатков существующих ботов | 20.01.2019 - 27.01.2019 | |
| Изучение технологий по разработке ботов | 27.01.2019 - 02.03.2019 | |
| Изучение методов обеспечения безопасности ботов | 02.03.2019 - 10.03.2019 | |
| Разработка бота | 10.03.2019 - 29.04.2019 | |
| Внедрение средств безопасности | 27.04.2019 - 02.05.2019 | |
| Расчет технико-экономического обоснования | 02.05.2019 - 10.05.2019 | |
| Ручное тестирование бота и устранение недостатков | 10.05.2019 - 15.05.2019 | |
| Заключение | 15.05.2019 - 20.05.2019 | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Дата выдачи задания «___» _____ 2019 г.

Заведующий кафедрой _____ (Р. Ш. Бердибаев)
(Подпись) (Ф.И.О)

Научный руководитель проекта _____ (Е. Т. Сатимова)
(Подпись) (Ф.И.О)

Задание принял к исполнению студента _____ (Д. Р. Турсунбаев)
(Подпись) (Ф.И.О)

## Аннотация

На основе тщательного ручного анализа и эксплуатации, различных Telegram ботов были выявлены критерии, характеризующие Telegram бот как качественный. В дипломном проекте был разработан Telegram бот PSinfo. При разработке была учтена большая часть положительных критериев, выявленных в процессе изучения других ботов. В боте реализованы необходимые меры защиты и интуитивно понятный интерфейс, позволяющий пользователю с легкостью управлять ботом. В качестве мер защиты реализована система управления доступ и фильтрация данных.

## Annotation

In this study based on the intimate hand testing and running of various Telegram bots were identified criteria that characterize the Telegram bot as high-quality and was developed Telegram bot PSinfo. The most positive criteria, identified in the process of studying other bots, were addressed in the course of the elaboration of the draft. The bot has the necessary protection and intuitive interface that allows the user to easily control the bot. The access control and data filtering system was implemented as a means of protection

## Аңдатпа

Telegram боттарын мұқият қолмен талдау және пайдалану негізінде, Telegram боттарын жоғары сапалы сипаттайтын әртүрлі өлшемдер анықталды. Дипломдық жобада Telegram bot PSinfo әзірленді. Құрастыру барысында басқа боттарды зерттеу барысында анықталған оң өлшемдердің басым бөлігі ескерілді. Қолданушыға ботты оңай басқаруға мүмкіндік беретін қажетті қорғау шараларын және қолдануға түсінікті сырт келбеті іске асырды. Қорғау шарасы ретінде, кіруді бақылау және деректерді сүзгілеу жүйесі енгізілген.

## Содержание

# Введение

Целью дипломной работы является разработка качественного бота, позволяющего автоматизировать процесс сбора персональных данных искомого человека на основе минимального количества данных.

Актуальность данной работы заключается в том, что на текущий момент нет ботов, которые качественно собирают данные из открытых источников и социальных сетей. Так же при разработке подобных ботов, часть из них является уязвимыми, так как разработчики не обладают навыками в сфере информационной безопасности. Главная цель – это разработать Telegram бота сочетающего в себе необходимые меры безопасности, простой и интуитивно понятный интерфейс и полезный и хорошо отфильтрованный выхлоп.

Для повышения информационной безопасности бота реализована система управления доступом, система логирования, система фильтрации данных, вводимых пользователем. Система фильтрации данных позволяет предотвратить атаки типа SQL-инъекции, удаленное выполнение кода, XSS-атаки, SSTI-инъекции и так далее. Так же используется система анонимности, которая позволяет совершать запросы к веб-ресурсам посредством различных proxy серверов. Помимо proxy серверов, все запросы бота пропускаются через VPN.

Для приятного взаимодействия пользователя с ботом вся информация выводится в форматированном виде. Данные подсвечиваются и выделяются специальными маркерами: жирный шрифт, курсивный шрифт и так далее.

На данный момент существует множество различных программ позволяющих собрать информацию о человеке. Разработанная программа работает только с открытыми источниками. Для того, чтобы воспользоваться разработанным программным обеспечением, владелец должен удостовериться, что искомый человек разрешает автоматизированный сбор информации. Только после того, как искомый человек даст согласие, владелец программного обеспечения сможет воспользоваться программой, не нарушая личное пространство искомого человека. В качестве открытых источников используются социальные сети: Yandex, Вконтакте и так далее.

## 1 Обзор и анализ классификации ботов

Telegram – это распространенный мессенджер, который используется для обмена информацией между людьми или группой людей. Имеет довольно обширный функционал, позволяет организовывать сообщества и групповые чаты. Широко распространен среди людей занимающихся IT сферой. Одно из преимуществ Telegram это то, что он поддерживается многими языками программирования, т.е. на текущий момент под многие языки уже разработаны библиотеки, позволяющие реализовывать ботов, например в языке Python разработана библиотека pyTelegramBotApi, которая с легкостью позволяет реализовывать различные Telegram боты.

Telegram Bot – это автоматизированное решение для выполнения различного рода действий с простым и интуитивно понятным интерфейсом. Преимущество ботов заключается в том, что их просто развернуть, так же они могут быть доступны в любой момент, в любой точке мира. Боты могут объединять в себе самый различный функционал, включая сбор информации, выполнение математических вычислений, предоставление различного рода услуг и т.д.

Боты могут использоваться в целях безопасности. На основе ботов можно развернуть систему оповещения, которая будет оповещать человека в случае инцидента. Приватная группа в Telegram может содержать бот, который будет реагировать на определенные команды, тем самым будет происходить различные действия в зависимости от переданной команды: открыть электронный замок на двери, обновить систему, запустить сканирование системы и так далее [1].

### 1.1 Общая классификация Telegram ботов

На сегодняшний день боты могут быть использованы в самых различных целях. Предназначение бота определяется фантазией и нуждами самого разработчика. В качестве бота может выступать огромная полноценная программа для управления целой информационной системой.

С точки зрения безопасности, как правило, все зависит от самого разработчика, насколько хорошо разработан бот и имеются ли ошибки в коде, которые могут привести к реализации различных типов атак, таких как SQL-инъекции, удаленное выполнение кода и так далее.

Для разделения ботов на некоторую классификацию необходимо определить следующие критерии:

1) наличие тех или иных функций и модулей;
2) понятность и доступность пользователю;
3) основной функционал бота;
4) информация предоставляемая ботом;
5) нацеленность на аудиторию.

На основе приведенных критериев ниже приведена классификация ботов и описание каждой классификации.

Бот-статист - цель данных ботов заключается в сборе информации и ее анализе. Данные боты используются для различного рода исследований. Боты такого типа должны обладать возможностью обработки большого количества информации, также некоторые боты такого типа могут предоставлять различные графики и статистику по проделанным исследованиям. Часто дополнительно к таким ботам разрабатываются сложные нейронные сети, позволяющие более эффективно производить обработку различного рода данных. Так же используется машинное обучение, что позволяет поддерживать актуальность бота на высоком уровне, постоянно обучая его и пополняя базу новыми данными из окружающей среды, на данном этапе важно не допустить отправление бота ложной информацией.

Рекламный бот - цель данного бота заключается в рекламировании некоторой продукции. Боты такого типа используются во многих областях и в различных социальных сетях. Данные боты часто маскируются под обычных людей, использую фейковые фотографии и фейковые персональные данные, чтобы максимально быть похожими на обычного человека и тем самым внушать доверие пользователям социальной сети. Данные боты массово кидают запросы в друзья и потом через друзей пользователя, принявшего дружбу, начинают свое распространение, оставляя за собой множество комментариев, постов на страницах и личных сообщений с рекламным контентом.

Агрегаторный бот - задача данного бота производить репост сообщений с различных сайтов. Данные боты работают с определенной информационной тематикой. Например, бот может репостить только информацию политического характера, тем самым на странице бота будет собрана куча репостов с различных сайтов с информационно-политическим характером.

Спам-бот - данные боты нацелены на совершение пропаганды определенной информации. Боты такого типа начинают массовую рассылку, по всем фронтам, используя личные сообщения, репосты в сообществах, спам-сообщения с определенной тематикой, спам-картинки с логотипом пропагандируемой информации и так далее. Боты такого типа являются назойливыми и чаще всего их банят, но пересоздать бот под другим именем с тем же самым функционалом не составляет сложности.

Бот-кукушка - данные боты используются для высказывания позитивного или негативного отношения при различных социальных движениях. Это делается для того, чтобы создать массовость. Например, можно взять два продукта разных компаний и предложить людям оценить их, многих людей не заинтересует данный опрос, но когда в дело вступают боты, вокруг данной оценки образуется массовость, что чаще всего заинтересует мимо проходящего пользователя, тем самым пользователь зайдет и оценит тот продукт, который ему больше нравится. Даже если пользователь никогда не пользовался данными продуктами, факт посещения данной площадки с оценкой продукта также фиксируется, и другие пользователи будут видеть,

что у площадки много посетителей и это так же служит средством привлечения пользователей.

Бот-консультант - данный тип ботов нацелен на оказание узконаправленной помощи и консультации людей, заинтересованных в получении некоторой узконаправленной информации. Боты такого типа используются в различных компаниях для оказания начальной помощи большому количеству человек, это удобно тем, что операторы не всегда успевают обслуживать всех пользователей, поэтому на начальном этапе клиента со стандартным вопросом могут обслуживать боты. Такие боты содержат ограниченное количество ответов для пользователя и могут реагировать только на определенные вопросы.

Как уже было упомянуто выше, бот может выполнять самые различные функции, это зависит от фантазии и нужд разработчика или заказчика, поэтому классифицировать все боты сложно [2].

### 1.2 Боты в информационной безопасности

Боты также применяются в информационной безопасности, например, в качестве системы оповещения, в случае наступления какого-либо события в информационной системе, связанного с информационной безопасностью. Бот может оповестить администратора и всех работников, имеющих отношение к подобным ситуациям. Помимо оповещения боты такого типа способны предоставлять информацию об информационной системе и ее состоянии администраторам, которые в данный момент не имею доступа к информационной системе. Это позволяет эффективно контролировать информационную систему в любое время суток и из любого места, при этом администратору достаточно иметь при себе только телефон.

Другое применение ботов в информационной безопасности – это предоставление доступа лицам, имеющим на это право. Например, для доступа к объекту необходимо предоставить электронный ключ-карту. Альтернативой ключ-карте может стать отправка боту специальной команды, по которой бот будет влиять на систему и предоставлять доступ к объекту. Важной деталью в таком применении ботов является правильная конфигурация самого бота, например, люди, не имеющие доступ к объекту, не должны иметь возможность отсылать сообщение боту. Такое ограничение можно достигнуть за счет создания приватных групп, доступ к которым можно получить только на основе специального приглашения.

Не менее важным элементом является безопасный код, т.е. разработчик бота должен обладать знаниями в информационной безопасности и правилами написания безопасного кода. Правила безопасного кода должны соблюдаться в обязательном порядке в случае, когда бот осуществляет взаимодействие с пользователем, т.е. когда бот принимает некоторые значения от пользователей. При разработке бота могут быть использованы структуры, являющиеся небезопасными, это может привести к возникновению различных типов атак, таких как SQL-инъекции, удаленное выполнение кода, XSS-атаки

и другие типы атак. Например, в случае, когда бот осуществляет взаимодействие с базой данных, т.е. бот запрашивает и добавляет данные в базу данных, при неправильной фильтрации вводимой информации от пользователя может произойти атака типа SQL-инъекция, что приведет к получению злоумышленником всей информации из базы данных.

### 1.3 Преимущества и недостатки

Преимущества использования ботов:

- простота в разработке: нет необходимости обладать большими знаниями в программировании, чтобы разработать свой бот;
- дешевизна: для разработки бота нет необходимости выделять большое количество ресурсов;
- простой и интуитивно понятный интерфейс: основными элементами управления ботом являются команды, прописанные в исходном коде бота, при введении которых бот воспроизводит функционал, указанный для той или иной команды, введенной пользователем, так же в качестве управления именно в Telegram-боте могут использоваться специально запрограммированные графические элементы, например, кнопки;
- высокий уровень автоматизации: в боте имеется возможность автоматизировать огромное количество различных действий, это делает использование ботов таким популярным;
- поддерживаются во множествах мессенджерах и социальных сетях: на текущий момент практически в любом мессенджере есть возможность внедрения бота, это так же повышает популярность ботов;
- доступность 24/7: так как сервер, на котором размещен бот, должен работать в режиме 24/7, то и бот может без остановки выполнять функционал, заложенный в нем разработчиком;
- возможность разработки бота с искусственным интеллектом: данная возможность позволяет разработать более мощные боты, которые построены на основе нейронных сетей, это позволяет сделать бот развивающимся и постепенно обучать его, тем самым делая его более полезным и самостоятельным;
- мгновенные результаты на запросы: на текущий момент многие языки программирования поддерживают многопоточность, это позволяет работать боту сразу с большим количеством людей, тем самым у бота отсутствует время простоя, т.е. пользователь не должен ожидать пока бот обслужит другого пользователя, так как для каждого пользователя будет выделяться собственный поток.

Недостатки использования ботов:

- не полное понимание запросов: так как боты - это машины и им необходимо взаимодействовать с человеком, часто происходит недопонимание ботом запросов человека, данная проблема решается за счет развития бота и добавления в него нового функционала. Боты эффективны, когда пользователь обращается к боту с определенными статическими

командами, как только пользователь начинает внедрять сарказм или сокращение вопросов в запросы при взаимодействии с ботом, бот начинает негативно и не полноценно реагировать, а часто и вовсе не реагирует на подобные запросы;

- плохая импровизация: боты хорошо работают, пока разговор не выходит за пределы алгоритма;

- примитивный интерфейс: несмотря на то, что интерфейс ботов довольно прост и понятен, он имеет довольно скудные графические возможности [3, 4].

### 1.4 Обзор Telegram ботов

Бот @Get_contact - данный бот нацелен на сбор информации о телефоне, который запрашивает пользователь. Боту необходимо предоставить номер телефона интересующего человека, в результате бот предоставит информацию о том, как данный номер телефона подписан у пользователей, которые имеются в базе данных программы.

Бот содержит обширную базу, которая была собрана посредством добровольного предоставления пользователем доступа к своему списку контактов телефона. Это было обязательным условием, без которого у пользователя не было возможности использовать данное приложение. Пользователь был вынужден предоставить доступ к своему телефонному справочнику, чтобы пользоваться программой, в результате сейчас данный бот содержит обширную базу.

С технической стороны данный бот разработан на языке программирования Python с использование базы данных MySQL. Бот содержит специальные идентификаторы под названием Token. Данный Token позволяет ограничить количество запросов для одного пользователя, это сделано для того чтобы исключить автоматизацию запросов со стороны других ботов. Так как пользователь напрямую взаимодействует с ботом и отправляет сообщения (номер телефона), в боте организована фильтрация вводимых пользователем параметров, надежная фильтрация позволит исключить возможность совершения атак типа удаленное выполнение кода, SQL инъекции, XSS инъекции и подобные атаки. Бот довольно быстро отвечает на запросы пользователя, при запросе в течение 2-3 секунд предоставляется ответ.

Данный бот не обладает большим функционалом и выдает информацию с определенным и узконаправленным смыслом, тем не менее, бот работает с обширной базой данных, также бот обладает самым простым интерфейсом. Для использования данного бота пользователем нет необходимости обладать специальными знаниями, это позволяет расширить бот для всех опытных и не опытных пользователей Telegram.

Из недостатков бот не обладает обширным функционалом, это значит, что данным ботом может заинтересоваться только узкий круг лиц. Бот

обладает максимально простым интерфейсом, что так же не является привлекательным, но достаточным для функционала бота [5].

Бот @Aviata.kz - данный бот является информационным и предоставляет различную информацию по путевкам, скидкам и авиабилетам. Бот имеет более дружелюбный интерфейс по отношению к предыдущему боту. Бот предоставляет информацию о популярных путевках, цену на различные путевки и скидки. Помимо информации бот предоставляет различные видео и картинки, что делает его более дружелюбным по отношению к пользователю. Данный бот является полезным для компании Aviata тем, что отлично служит для распространения рекламной информации.

С технической стороны данный бот написан на языке программирования Python 3.6. У пользователя отсутствует возможность отправлять данные боту, это исключает возможность совершения различных атак со стороны пользователя. Информация пользователю предоставляется в привлекательном виде с использованием картинок, видео и форматированного текста, что так же может привлечь пользователя.

Из недостатков бот не обладает обширным функционалом, бот является информационным и все, что он может - это только предлагать некоторую информацию о путевках.

Бот @DrWebBot - данный бот обладает более обширным функционалом по сравнению с предыдущими ботами. Основное назначение бота заключается в обеспечении информационной безопасности пользователей. Бот обладает множеством зарезервированных команд, которые позволяют использовать различный функционал.

Интерфейс бота более продвинутый, используются кнопки и специально зарезервированные команды, так же бот поддерживает несколько языков. Команда /settings позволяет открыть меню, где пользователь может посредством кнопок выбрать необходимые для него действия, например, выбрать язык или настроить уведомления. Команда /feedback позволяет оставить комментарии о боте, оценить его работу и написать пожелания для его улучшения.

Из основного функционала бот является аналогом мини антивируса. Бот предоставляет возможность пользователю проверить безопасность файлов и ссылок различными способами. Так же бот поддерживает возможность его внедрения в различные Telegram группы, где он будет проверять ссылки и передаваемые файлы в группе. Данный функционал является довольно обширным, для разработки такого бота необходимо больше знаний, чем у программиста среднего уровня.

В отношении собственно безопасности бот так же является надежным, вся информация, передаваемая пользователем, тщательно фильтруется, т.е. исключается возможность реализовать атаки типа удаленное выполнение кода, SQL-инъекции, XSS-атаки. Так же реализована защита от различного рода нестандартных символов типа массив и других спец символов.

С технической точки зрения бот написал на языке программирования Python 3.6. Работает с базой данных MySQL. Бот содержит базу с сигнатурами, которые могут быть использованы для закладок и реализации атак различных типов. При переходе на ссылку или при открытии файлов, если бот обнаруживает имеющиеся у него сигнатуры, он сообщает пользователю о наличии некоторой вредоносной информации. С точки зрения скорости, бот работает быстро, за счет ограниченного размера на загружаемые файлы.

Из недостатков бота можно выделить то, что бот имеет ограниченный функционал по сравнению с многими другими полноценными антивирусами, также присутствует ограничение на размер загружаемых файлов в 10 мегабайт [6].

### 1.5 Выбор оптимального Telegram бота

При разработке собственного бота, для того чтобы он пользовался популярностью, автором было изучено множество различных ботов, проведена оценка их положительных и отрицательных сторон. Данный анализ позволил выявить сильные и слабые стороны, тем самым применить сильные стороны в своей реализации и не допустить слабые стороны.

На основе анализа различных ботов были выделены следующие критерии, которыми должен обладать качественный бот:

- функционал: функционал бота должен быть обширным и полезным, бот не должен быть узконаправленным и предоставлять информацию только одной тематики или же эта тематика должна быть интересна практически для любого пользователя, который увидит данный бот;

- интерфейс – интерфейс бота должен выглядеть приветливо и так же обладать широким функционалом, бот должен поддерживать несколько языков, хотя бы английский и русский, иметь многофункциональные кнопки, специальные зарезервированные команды, добавляющие дополнительный функционал боту и, в обязательном порядке, максимальное упрощенное пояснение для каждой функции бота;

- скорость работы: бот должен быстро реагировать на запросы пользователя; такой параметр сложно сочетать с высоко нагруженными ботами, но все же возможно. Для более высокого результата производительности обязательно должно использоваться многопоточное программирование, также, если бот работает сразу с несколькими ресурсами или ему необходимо выполнять множество различных операций для одного пользователя, можно для каждого пользователя помимо его собственного потока организовать многопоточное выполнение задач. Так же, для улучшения производительности бота, могут использоваться более производительные языки программирования, которые обладают более эффективными возможностями управления памяти, в качестве такого языка можно использовать язык программирования Golang;

- безопасность: для предотвращения нарушения работы бота или хищения информации с базы данных бота, необходимо обеспечить надежную защиту. Главной мерой защиты при использовании Telegram бота является качественная фильтрация всех параметров, вводимых пользователем, это позволит предотвратить атаки типа удаленное выполнение кода, SQL-инъекции, XSS-атаки и так далее. Помимо фильтрации параметров, если бот является приватным и использовать его могут только ограниченный круг лиц, имеющий на это доступ, необходимо ограничить доступ, чтобы пользоваться данным ботом мог только определенный круг лиц, имеющих на это право.

**Вывод**

В данной главе были рассмотрены различные типы ботов, выделены минусы и плюсы. В результате можно сказать, что боты действительно являются в большей части полезным программным обеспечением, позволяющим автоматизировать некоторый функционал. Боты полезны при взаимодействии с большим количеством людей, т.е. они, нацелены на массовость, что делает их такими популярными.

В результате автором было замечено, что не все боты являются полезными, некоторые даже служат для нарушения информационной безопасности, а некоторые наоборот нацелены на ее обеспечение. Также существуют и совсем бесполезные боты, которых люди разработали просто для получения опыта в разработке или это просто не удавшиеся проекты.

В результате изучения автор для себя выделил некоторые критерии, которые необходимо учесть при разработке собственного бота. Для разработки хорошего бота необходимо учитывать множество параметров, таких как безопасность, полезность, простота. При анализе различных ботов автор выделил следующие критерии, которые должны присутствовать в представленной разработке:

- интуитивно понятный интерфейс: интерфейс бота не должен вводить в заблуждение пользователя;
- многопоточность: для каждого пользователя каждый запрос должен обрабатываться в отдельном потоке;
- быстрота обработки запросов: ответы на запросы от пользователей не должны быть долгими;
- высокая полезность информации: информация должна приносить пользу пользователю;
- точность информации: программное обеспечение не должно выдавать информацию, не относящуюся к запросу пользователя;
- безопасность: должен быть организован надежный уровень безопасности для предотвращения инцидентов.

Множество данных критериев реализованы в рассмотренных ранее ботах, некоторые боты даже объединяют в себе все перечисленные критерии. Все перечисленные критерии должны быть учтены при разработке бота, это сделает его востребованным и популярным.

## 2 Разработка пользовательского интерфейса для PSinfo Bot

### 2.1 Разработка логической структуры Telegram бота PSinfo

В рамках данного дипломного проекта был разработан информационный Telegram-бот «PSinfo». При распространении бота, нет необходимости предоставлять пользователю исходный код программы, так как бот работает в режиме онлайн. Для управления ботом используется приватная группа, в которую пользователи могут попасть посредством их приглашения.

Бот включает множество различных модулей, каждый из которых выполняет какую-то функцию. В основном каждый модуль нацелен на отдельный информационный ресурс. Так же имеется основной модуль, в котором происходит поочередный вызов всех остальных модулей. В данном разделе описывается интерфейс бота и взаимодействия с другими модулями. Архитектура бота показана на рисунке 1, представлены основные модули Telegram-бота.


Рисунок 1 – Архитектура Telegram бота «PSinfo»

Самой важной деталью является главный модуль, через который идет передача всей информации, модуль занимается приемом информации от пользователя и передачей ее другим модулям, в дальнейшем приемом от модулей собранной информацией, ее обработкой и выводом пользователю. Решение разделить программное обеспечение на модули позволяет упростить его разработку. Так как в боте информация собирается из множества различных веб-ресурсов, проще под каждый ресурс писать отдельный модуль и организовать между ними взаимодействие.

### 2.2 Разбор логической структуры Telegram бота PSinfo

Так как Telegram бот состоит из множества модулей, необходимо разъяснить предназначение каждого модуля, чтобы увидеть полную картину. Решение разработать бот, разделив его на модули, показалось автору более надежным, это позволяет редактировать бот, не меняя полностью структуру. Если возникают проблемы с какой-то частью программы, можно изменить ее, не внося изменения в другие ее части, потому что основным взаимодействием между всеми модулями является только передача информации от пользователя и назад собранной информации. На рисунке 2 выделен главный модуль бота, который включает в себя основной функционал.

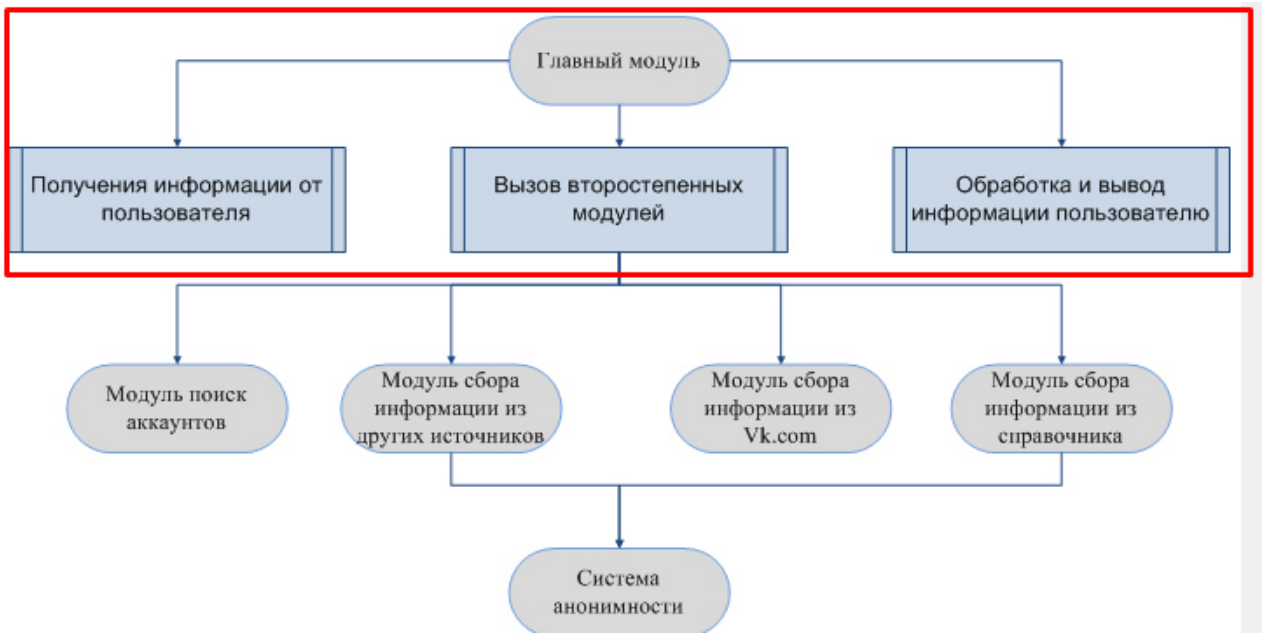


Рисунок 2 – Главный модуль Telegram бота PSinfo

В данном модуле прописан основной функционал и интерфейс бота «PSinfo». В функционал бота входит взаимодействие с пользователем, так как именно данный модуль занимается приемом сообщений от пользователей, проверкой сообщений пользователей и отправкой этих сообщений в другие модули.

Согласно конфигурации бота, пользователь должен ввести определенные идентификаторы в определенном порядке. Первым из идентификаторов должен быть предполагаемый город человека, данный параметр необходим для модуля, который работает со справочником. Вторым и последующими параметрами должны следовать фамилия, имя и отчество человека, на основе уже этой информации будут задействованы следующие модули.

Главный модуль содержит основной интерфейс, который будет предоставлен пользователю. В программе реализован простой интерфейс, так как часто бот выдает довольно много информации, она должна

предоставляться в читабельном виде. Бот предоставляет информацию по аккаунтам пользователя и сразу же передает кнопку, нажав на которую пользователь сможет перейти к выбранному аккаунту. Так же бот выводит более точную информацию в формате ключ и значение, т.е. найдя информацию о человеке, например, дату рождения, бот выдаст ее в следующем формате (Дата рождения: 18 июня 1997 года).

Также этот модуль содержит проверки, позволяющие проверить сообщение пользователя. Проверки позволяют определить, верно ли ввел пользователь данные, если пользователь ввел меньше идентификаторов, чем необходимо боту, это приведет к остановке поиска и бот будет вынужден сообщить пользователю, что данной информации не достаточно для поиска.

Помимо всего перечисленного выше, бот содержит определенные зарезервированные команды. Передав команды /help, /start и так далее, пользователь вызовет дополнительные функции. Например, при введении параметра /help, пользователю выведется информация о боте и инструкция как можно пользоваться данным ботом.

Следующим шагом является попытка найти аккаунты после того как пользователь передал корректную информацию, содержащую данные искомого человека. За данный поиск отвечает модуль, показанный на рисунке 3.

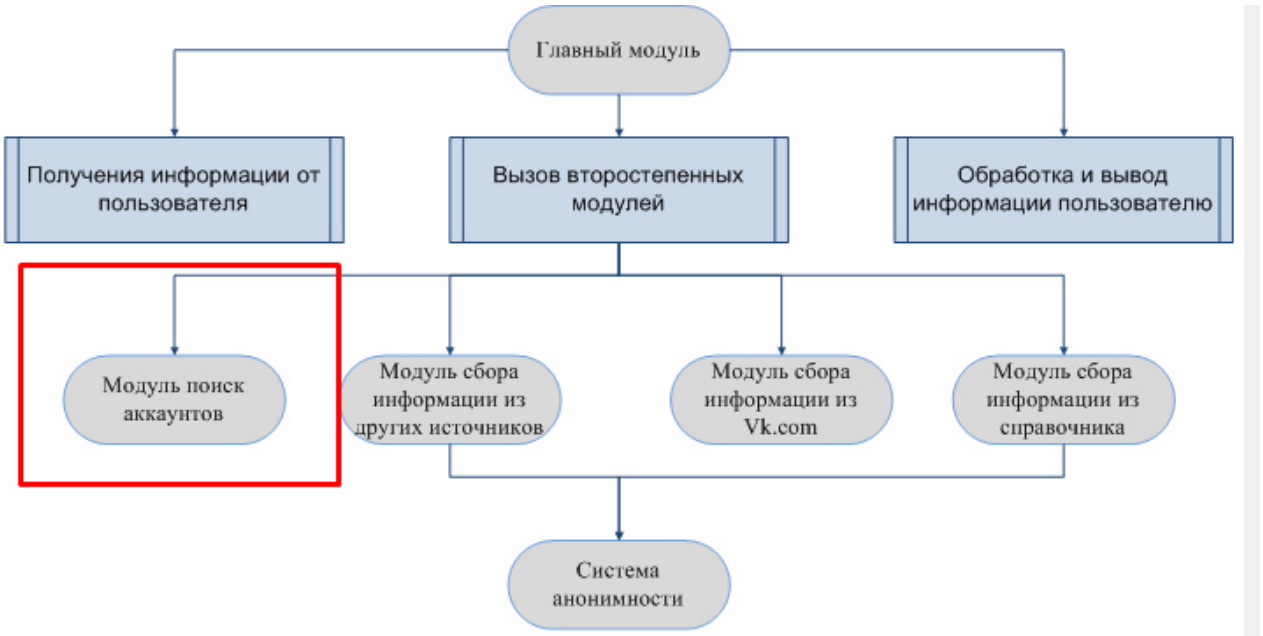


Рисунок 3 – Модуль для поиска аккаунтов

Данный модуль работает на основе Яндекс поиска. Яндекс поиск предоставляет функционал, позволяющий искать аккаунты людей в социальных сетях. Для поиска аккаунтов Яндекс поиску необходимо предоставить фамилию, имя и отчество человека. Чем больше данных

предоставить поиску, тем он будет более точным. Пример поиска человека через браузер приведен на рисунке 4.

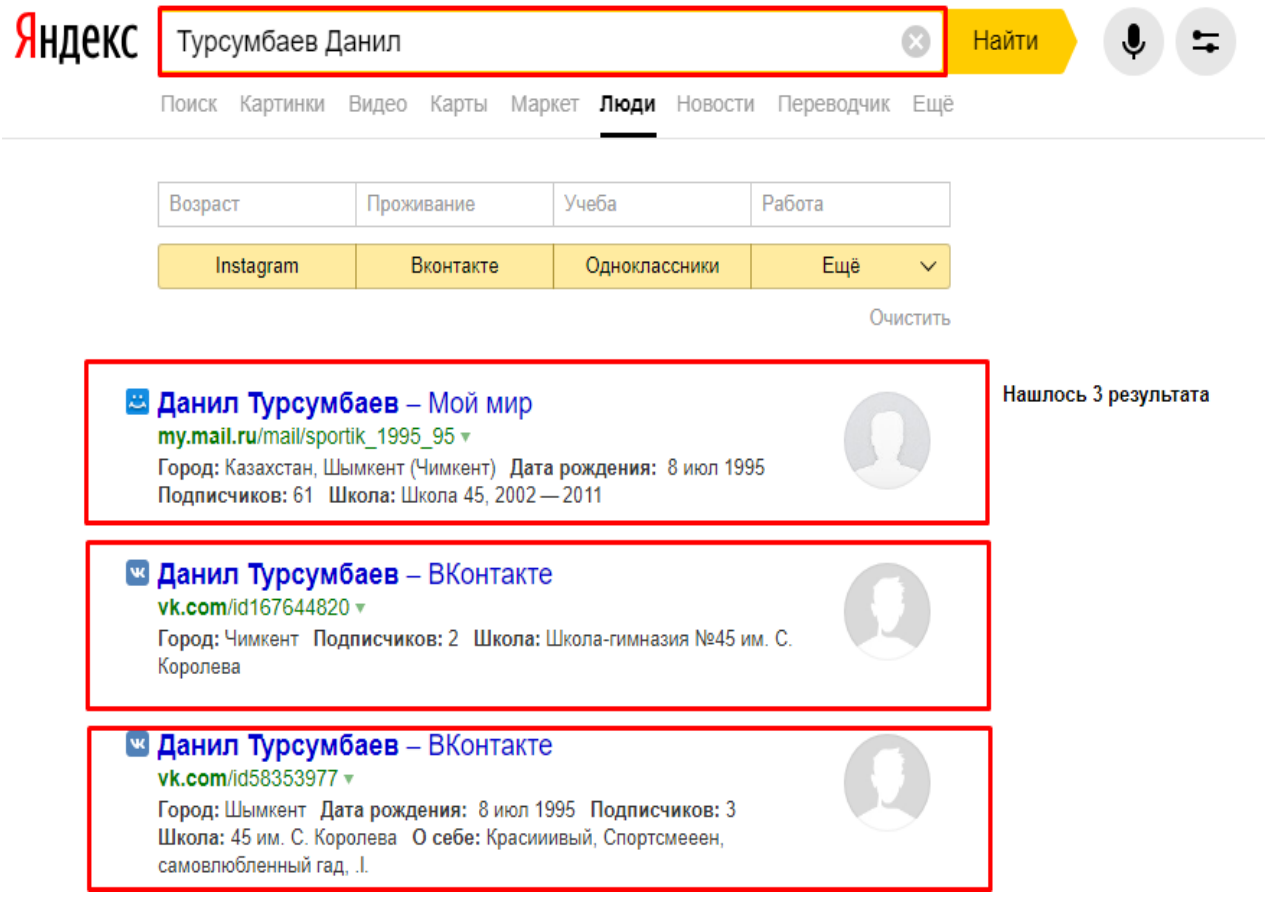


Рисунок 4 – Пример поиска человека посредством «Яндекс»

Как видно из рисунка «Яндекс» поиск поддерживать поиск людей в различных социальных сетях, таких как «Вконтакте», «Одноклассники», «Instagram», «mail.ru» и так далее. Так же не обязательно использовать комбинацию фамилия, имя и отчество, можно использовать другие комбинации, например, фамилия и имя или отчество и имя. Помимо различных комбинаций можно добавлять дополнительные параметры поиска, таких как возраст, город, место учебы и работа. Все эти идентификаторы позволяют повысить точность поиска запрашиваемого человека.

В результате после запроса на поиск человека, как показано на рисунке 4 «Яндекс» поиск выдаст всех найденных людей, бот соберет ссылки с полным именем и передаст обратно главному модулю на обработку. Главный модуль тем временем отсортирует всю полученную информацию и выделит некоторые аккаунты, которые будут переданы следующему модулю, показанному на рисунке 5. Остальные аккаунты будут выведены пользователю позднее, как только отработает модуль сбора информации из социальной сети «Вконтакте».

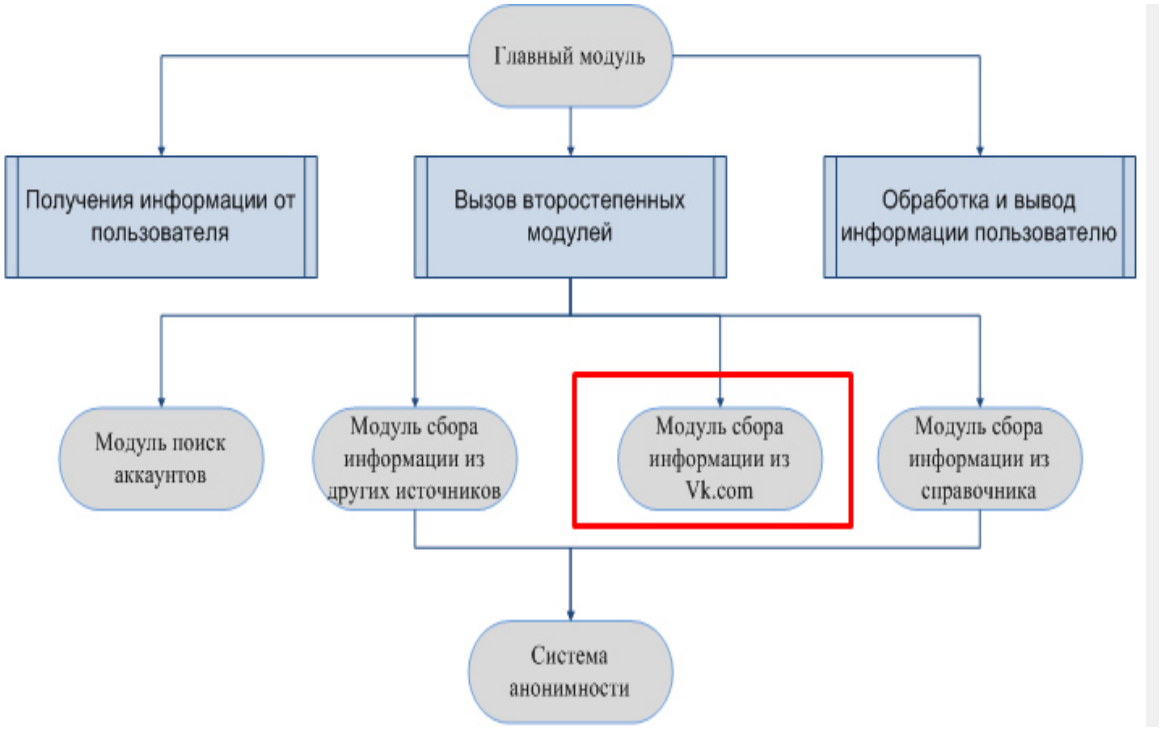


Рисунок 5 – Модуль сбора информации из «Вконтакте»

Найденные аккаунты передаются на парсинг - автоматический сбор информации с какого-либо источника с целью его дальнейшей обработки и преобразования. Многие люди оставляют о себе множество информации в социальных сетях, некоторые имеют сразу несколько аккаунтов, поэтому есть смысл произвести парсинг аккаунтов. Из найденных аккаунтом можно собрать большое количество информации, такой как город проживания, дату рождения, увлечения, место учебы, место работы и так далее.

Сложность создания данного модуля заключается в том, что необходимо написать качественные регулярные выражения, которые безошибочно будут собирать данные со страницы аккаунта. Необходимо учесть самый различный контент, который пользователь может оставить на странице, потому что, если регулярное выражение при поиске некоторой информации вернет пустое значение и даст ошибку, это может привести к ошибке при обращении к переменной, которая должна содержать собранную информацию, в дальнейшем эта ошибка вызовет нарушение работоспособности всей программы.

Некоторая информация, которую можно собрать со страницы аккаунта, показана на рисунке 6 из которого видно, что пользователи оставляют довольно много персональных данных. В результате текущий модуль должен собрать всю эту информацию и вывести ее пользователю в читабельном виде. В случае, когда пользователь не оставил персональные данные или его персональная страница закрыта для просмотра, необходимо задействовать другие веб-ресурсы.

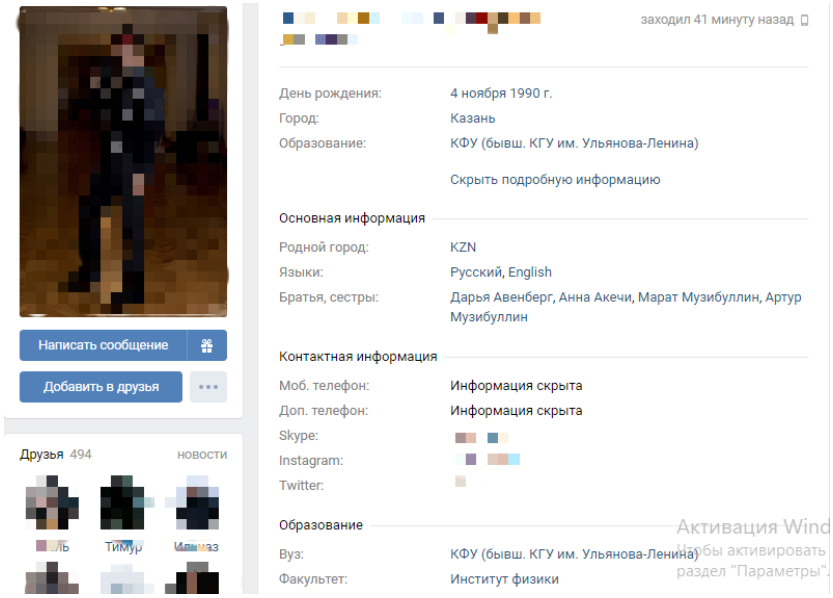


Рисунок 6 – Страница аккаунта с персональными данными

Ранее уже упоминалось, что пользователь должен первым параметром ввести примерный город, данное значение в связке с фамилией используется для сбора информации с более надежных источников, чем социальные сети. Данный веб-ресурс позволяет получить следующую информацию, адрес проживания человека и домашний телефон. Недостаток использования данного веб-ресурса заключается в том, что данные справочник содержит только людей, на которых зарегистрировано некоторое имущество. Но используя данный справочник можно выйти на однофамильцев искомого человека, это позволит определить родственников искомого человека. На рисунке 7 выделен модуль для сбора информации посредством телефонного справочника.

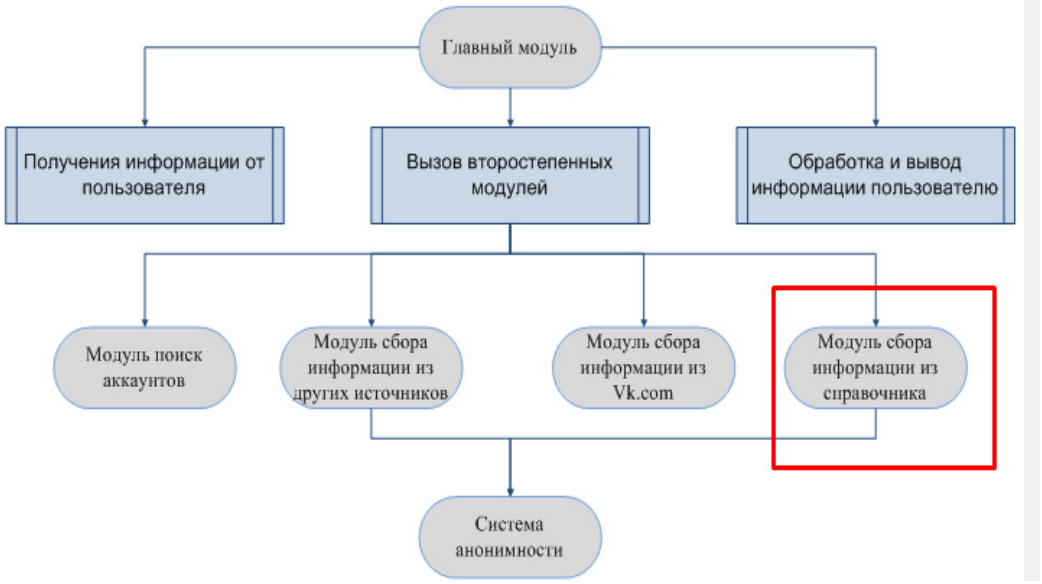


Рисунок 7 – Модуль для сбора информации посредством справочника

На рисунке 8 приведен пример поиска человека на основе города и первых трех букв фамилии. Как видно из рисунка веб-ресурс выдал все фамилии, начинающиеся на «ТУК» в городе Алматы. Так как автор уже владеет нужно ему фамилией и знает, что она начинается на «ТУК», остается ее только найти из списка выданного веб-ресурсом.

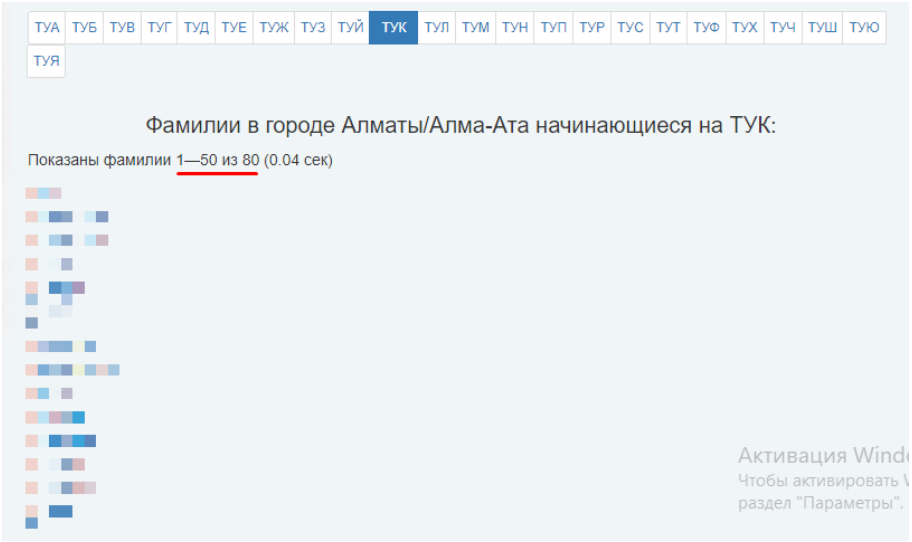


Рисунок 8 – Поиск необходимой фамилии

Как только необходимая фамилия будет найдена модулем, бот спарсит ссылку, которая прикреплена к данной фамилии, что позволит продвинуться вперед и найти всех однофамильцев, проживающих в искомом городе. Найдя всех однофамильцев можно предположить, что данные люди являются родственниками искомого человека или же сам искомый человек может оказаться среди найденных людей. Когда бот найдет нужную фамилию, остается опять же собрать ссылку, которая связана с данной фамилией и перейти по ней. После перехода по ссылке веб-ресурс выдаст информацию по данному человеку, включая домашний номер телефона и адрес проживания. Пример выдаваемой информации по человеку приведен на рисунке 9.

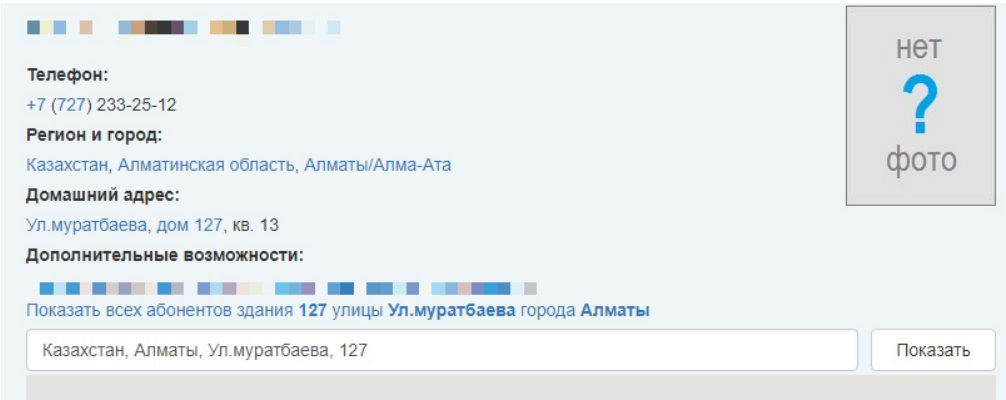


Рисунок 9 – Информация с телефонного справочника

На рисунке10 выделен модуль, отвечающий за сбор информации с иных источников. Под иными источниками подразумеваются любые веб-ресурсы, посредством которых можно получить персональные данные. Так как для сбора информации из этих источников нет необходимости писать отдельный модуль, все эти источники в программе объединены в одном месте.

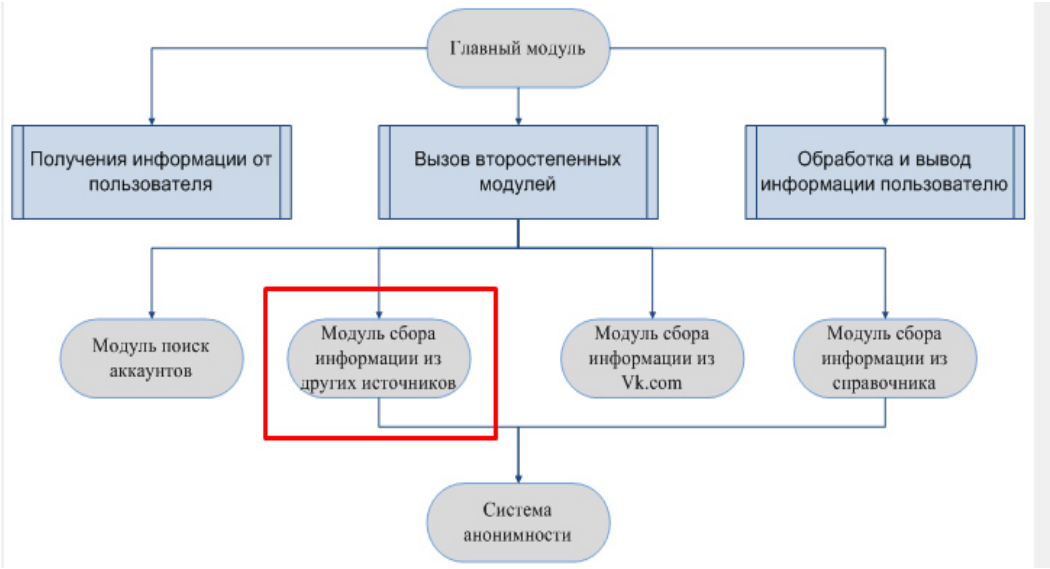


Рисунок 10 – Модуль для сбора информации с иных источников

Данный модуль будет возвращать целый массив информации главному модулю, в котором будет отсортирован и выведен пользователю [7].

### 2.3 Анонимность запросов к веб-ресурсам

Для работы со всеми источниками в работе использованы некоторые средства анонимности, модуль выделен на рисунке 11.

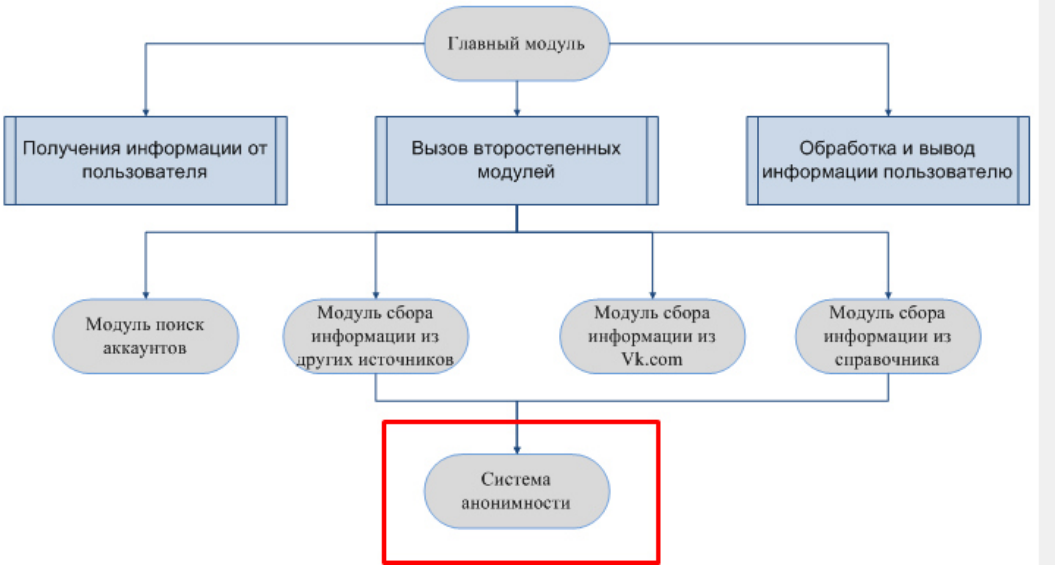


Рисунок 11 – Система анонимности

В качестве средств анонимности используются различные связки, например VPN + Tor или Tor + VPN, proxy сервера, или просто VPN. Первым делом используется VPN, приобретенный на https://fornex.com/, стоимость VPN на два месяца составляет 2 евро. На рисунке 12 показана информация по приобретенному VPN.

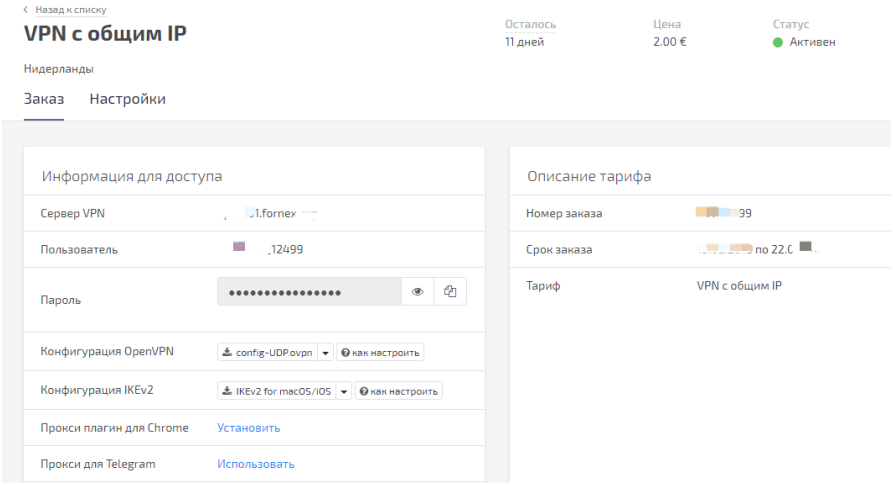


Рисунок 12 – Информация по VPN

Так как VPN включается на самой машине, где будет запущен бот, нет необходимости его конфигурировать средствами Python. Следующим этапом анонимности является использование сети Tor или Proxy серверов. Для работы с Proxy серверами понадобится конфигурация всех запросов исходящих из бота. Необходимо все запросы на любые веб-ресурсы пропускать через отдельный Proxy сервер, это можно реализовать использую конфигурацию самого запроса в Python. Запросы в Python поддерживают такие дополнения как headers, proxies, data, json и так далее. На рисунке 13 приведен пример запроса через Proxy сервер.

```
Import requests, re, queue

File = "proxy.txt"

Open_file = open(file, mode = 'w')
ochered = queue.Queue()

If ochered.empty() == True:
        for line in open_file:
                ochered.put(line)

prox = ochered.get()
zapros = requests.get('https://vk.com', proxies = {
                'http' : 'socks4://{}:443'.format(prox),
                'https' : 'socks4://{}:443'.format(prox)
                }
```

Рисунок 13 – Запрос через Proxy сервер

Все IP адреса Proxy серверов берутся из файла «proxy.txt» и загружаются в очередь. При попытке совершить запрос к vk.com, из ранее созданной очереди будет взят один IP адрес Proxy сервера и подставлен в запрос. В результате запрос пойдет не напрямую к vk.com, а через Proxy сервер. Но сам Proxy сервер не предоставляет шифрование, для этого используется связка VPN + Proxy, трафик будет идти через Proxy сервера в зашифрованном виде.

Еще одним вариантом осуществить анонимность является использование связку VPN+Tor. Tor будет пропускать весь трафик через Proxy сервера, а VPN обеспечивать шифрование. Данный метод тоже может использоваться эффективно, но на текущей момент использование Tor не рекомендуется, так как данная сеть не является безопасной. Некоторые веб ресурсы блокируют запросы, исходящие от Tor. Это делается посредством блокировки всего пула Tor адресов, который уже известен. Так же Tor-сеть не является безопасной, так как кто-то может пропускать своей трафик, используя вашу машину как Proxy сервер. Так же использование сети Tor сильно замедлит запросы пользователей. Пример реализации запросов через связь VPN + Tor приведен на рисунке 14.

```
proxies = {
        'http': 'socks5://127.0.0.1:9150',
        'https': 'socks5://127.0.0.1:9150'
}

def get_letter_city(surname, city):
        lets = ''.join(surname)

        city_number = cities['{}'.format(city)]
        list_number = 'list{}_{}_{}'.format(letter['{}'.format(lets[0])], letter['{}'.format(lets[1])],
letter['{}'.format(lets[2])])

        get_pages(city_number, list_number, surname)
        return data_from_spravka

def get_pages(city_number, list_number, surname):
        req = requests.get('http://spra.vkaru.net/names/7/{}/{}'.format(city_number,
list_number), proxies = proxies, headers = head, cookies = cookie)

        count_page = re.findall("{}/page".format(list_number), str(req.text))
        mass_with_pages = []

        count_page = len(count_page) + 2

        for i in range(1, count_page, 1):
                req = requests.get('http://spra.vkaru.net/names/7/{}/{}/page{}'.format(
city_number, list_number, i), proxies = proxies, headers = head, cookies = cookie)
```

Рисунок 14 – Анонимность через связку VPN + Tor

Эксплуатация сети Tor практически не отличается от использования просто Proxy-серверов. Однако, можно заметить, что Tor используется следующие порты 9150, 9050, 9051  в данном же случае Tor работает на порту 9150.

Так же можно использовать различные средства анонимности типа двойной VPN. В данном случае необходим еще один сервер, на котором будет вручную развернут VPN сервер. Так же сам хост, где запущен бот будет находиться под другим VPN. Трафик будет проходить через первый VPN сервер, к которому подключен основной хост, где запущен бот. Дале с этого сервера трафик пойдет на сервер, где вручную развернут собственный VPN сервер. После прохождения второго VPN сервера трафик пойдет к целевому ресурсу, которому и предназначался запрос. Данная конфигурация позволяет быть уверенным в собственном развернутом VPN и исключить ведение логов, что часто используется различными серверами, которые предоставляют VPN услуги [8, 9].

**Вывод**

В данной главе рассмотрена структура бота. Приведены все модули, которые включает в себя бот и описано предназначение и функционал каждого модуля. Так же рассмотрены средства анонимности, используемые ботом.

Стоит отметить, что каждый модуль является полноценной программой, которая способна работать самостоятельно, необходимо только передавать модулю определенные параметры в формате: фамилии, имена, города и отчества. Но лишь в совокупности все данные модули могут реализовать полноценный бот, способный эффективно добывать нужную информацию.

Выбранная модульная структура позволяет легко манипулировать как целиком всей программой, так и отдельными ее частями. В случае если какой-то из веб-ресурсов поменяет свою структуру, нет необходимости переписывать всю программу, нужно будет только переписать тот модуль, который отвечает за работу с данным веб-ресурсом. Все взаимодействие между модулями построено лишь на передаче информации от модуля к модулю, они не зависят друг от друга, в любой момент имеется возможность отключить любой модуль, закомментировав только одну строку в главном модуле. Как показала практика, такая реализация удобна в управлении программой.

## 3 Алгоритм работы и функционал бота

### 3.1 Блок-схемы работы бота

На рисунке 15 приведена блок-схема, которая показывает алгоритм работы главного модуля.

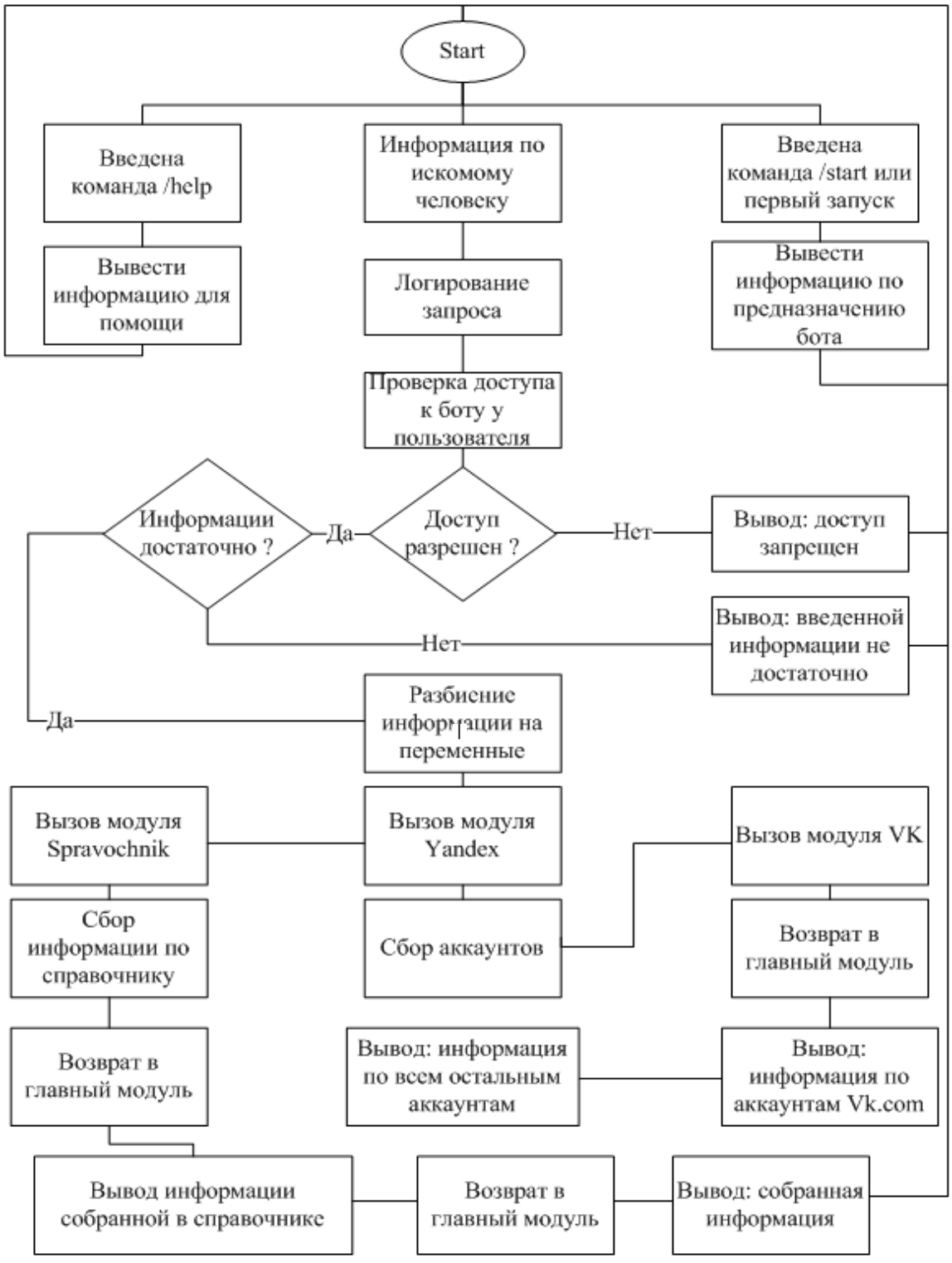


Рисунок 15 - Блок-схема работы главного модуля

На рисунке 16 показан принцип работы модуля Yandex.


Рисунок 16 – Блок-схема работы модуля Yandex

На рисунке 17 показан принцип работы модуля VK.


Рисунок 17 – Блок-схема работы модуля VK

На рисунке 18 показан принцип работы модуля Spravochnik.


Рисунок 18 – Блок-схема работы модуля Spravochnik

### 3.2 Обработка ошибок и непрерывность

Одним из основных свойств, которое характеризует программное обеспечение как качественное, является непрерывность. Непрерывность заключается в том, что в случае возникновения ошибок в ходе работы программного обеспечения, программа не должна переставать работать. Программа должна определять природу ошибок и реагировать на них, пытаясь повторить запрос пользователя и сообщить пользователю причину и происхождение ошибки. В Python всех версий используется обработчик ошибок Try – Except. Данный обработчик ошибок позволяет отлавливать ошибки различного типа и определять действия, которые необходимо произвести в случае возникновения ошибки. В случае возникновения ошибки, для того чтобы программа не переставала работать, в коде использована функция pass, которая подразумевает, что в случае возникновения ошибки не останавливается работа программы и выполняются действия, прописанные после pass. Использование данной структуры приведет к непрерывной работе программы, главное соблюсти логику, которая позволит обработать ошибку, сообщить пользователю о характере ошибки и на основе информации предоставленной пользователю, пользователь должен отреагировать на эту ошибку, тем самым не прерывая работу программы. На рисунке 19 показан пример эксплуатации структуры Try – Except.

```
def authorization(id_user):
    try:
        req_remixlhk = requests.get('https://m.vk.com/login', headers = {'Accept-
Language':'ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7'})
    except Exception as error:
        pass
        print(error)

    remixlhk = re.search('remixlhk=(.*?);', str(req_remixlhk.headers))
    lg_h = re.search('lg_h=(.*?)&', str(req_remixlhk.text))

    try:
        req_authorization =
requests.post('https://login.vk.com/?act=login&_origin=https://m.vk.com&lg_h={}&utf8=1'.for
mat(lg_h.group(1)), cookies = {'remixlhk':'{}'.format(remixlhk.group(1))}, data =
'email={}&pass={}'.format(login, password), allow_redirects = False)
    except Exception as error:
        pass
        print(error)

    __q_hash = re.search("__q_hash=(.*?)[',;&]", str(req_authorization.headers))

    try:
        req_remixsid =
requests.get('https://m.vk.com/login?role=fast&to=&s=1&__q_hash={}'.format(__q_hash.group
(1)), allow_redirects = False)
    except Exception as error:
        pass
        print(error)
```

Рисунок 19 – Пример эксплуатации структуры Try – Except – pass

В случае возникновения ошибки при запросе, бот активирует структуру (except Exception as error), после нее отрабатывает функция pass, выполнив её,

бот предотвратит завершение программы из-за ошибки и вызовет вывод контента ошибки для администратора, чтобы администратор мог увидеть, что в данном модуль произошла ошибка, причину и тип ошибки. Такой принцип использован во всех модулях бота.

Для контроля введенных значений пользователями в программе реализована структура, которая контролирует количество введенных значений пользователем. Всего пользователь должен ввести как минимум 3 параметра разделенных пробелами. Первым параметром должен быть город, последующими параметрами должны быть персональные данные пользователя, его фамилия, имя и отчество. Для того что бы программа отработала, пользователю необходимо ввести как минимум два идентификатора пользователя и первым идентификатором ввести город. После ввода, данные бот запишет в переменную personal_info. Далее массив personal_info разделен на разные отдельные переменные, если пользователь не ввел значение, то переменной бот присвоит пустое значение. Код обработки введенных пользователем значений показан на рисунке 20.

```
@bot.message_handler(content_types=['text'])
def get_text(message):
    personal_info = message.text.split(' ')

    try:
        city_from_user = personal_info[0]          # Город
    except Exception as error:
        pass
        city_from_user = ''

    try:
        last_name_from_user = personal_info[1]     # Фамилия
    except Exception as error:
        pass
        last_name_from_user = ''

    try:
        first_name_from_user = personal_info[2]    # Имя
    except Exception as error:
        pass
        first_name_from_user = ''

    try:
        middle_name_from_user = personal_info[3]   # Отчетстве
    except Exception as error:
        pass
        middle_name_from_user = ''
```

Рисунок 20 – Подготовка информации введенной пользователем

После того как все значения, введенные пользователем, поделены на разные переменные, бот проверит количество значений, введенных пользователем. Минимальное количество значений должно составлять 3, т.е. город и два параметра на выбор пользователя из фамилии, имени и отчества, рисунок 21.

```
if len(personal_info) < 3:
        bot.send_message(message.chat.id, "<b>[-] Недостаточно информации.</b>
<i>Должны быть заполнены два идентификатора пользователя из трех и город для
получения более ценной информации.</i>", parse_mode = 'HTML')
else:
        print(city_from_user, first_name_from_user, last_name_from_user,
middle_name_from_user)
        yandex(last_name_from_user, first_name_from_user)
        spravochnik()
```

Рисунок 21 – Проверка на количество введенных значений

Если пользователь ввел меньше трех значений, бот вызовет условие и пользователю будет выведено следующее сообщение: «Недостаточно информации. Должны быть заполнены два идентификатора пользователя из трех и город для получения более ценной информации».

Далее после передачи города и фамилии пользователя в следующий модуль, бот проверит, правильно ли пользователь ввел город. Если город был введен не правильно, бот перестанет выполнять запрос. Для проверки города в коде прописан словарь с параметрами типа (ключ:значение), рисунок 22. Бот проверит, есть ли город, введенный пользователем, в словаре, если он есть, то переменной city присвоится идентификатор города необходимый для поиска.

```
cities = {'Астана': '7172', 'Алматы': '727', 'Кокшетау': '7162', 'Актюбинск': '7132', 'Алга':
'71337', 'Бадамша': '71342', 'Байганин': '71345', 'Кандыагаш': '71333', 'Комсомольское':
'71339', 'Мартук': '71331', 'Хобда': '71341', 'Хромтау': '71336', 'Шалкар': '71335',
'Шубар-кудук': '71346', 'Эмба': '71334', 'Талдыкорган': '72822', 'Атырау': '7122',
'Семипалатинск': '7222', 'Тараз': '7262', 'Уральск': '7112', 'Караганда': '7212',
'Костанай': '7142', 'Кызылорда': '7242', 'Аральск': '72433', 'Жалагаш': '72431',
'Жанакорган': '72435', 'Кармакчи': '72437', 'Казалинск': '72438', 'Теренозек': '72436',
'Шиели': '72432', 'Павлодар': '7182', 'Аксу': '71837', 'Экибастуз': '7187', 'Петропавловск':
'7152', 'Актау': '7292', 'Шымкент': '7252'}

data_from_spravka = {}

def get_letter_city(surname, city):
        lets = ''.join(surname)

        city_number = cities['{}'.format(city)]
        list_number = 'list{}_{}_{}'.format(letter['{}'.format(lets[0])], letter['{}'.format(lets[1])],
letter['{}'.format(lets[2])])

        get_pages(city_number, list_number, surname)
        return data_from_spravka
```

Рисунок 22 – Проверка правильно введенного пользователем города

В помощь пользователю в коде программы внедрены зарезервированные команды, которые подсказывают пользователю о предназначении бота, принципах его работы и принципах его использования. Это реализовано, чтобы пользователю было легче ориентироваться при работе с ботом.

При первом включении бота пользователю выдается сообщение с информацией, рисунок 23. Так же, бот предоставляет команды, которыми пользователь может воспользоваться для получения дополнительной информации.

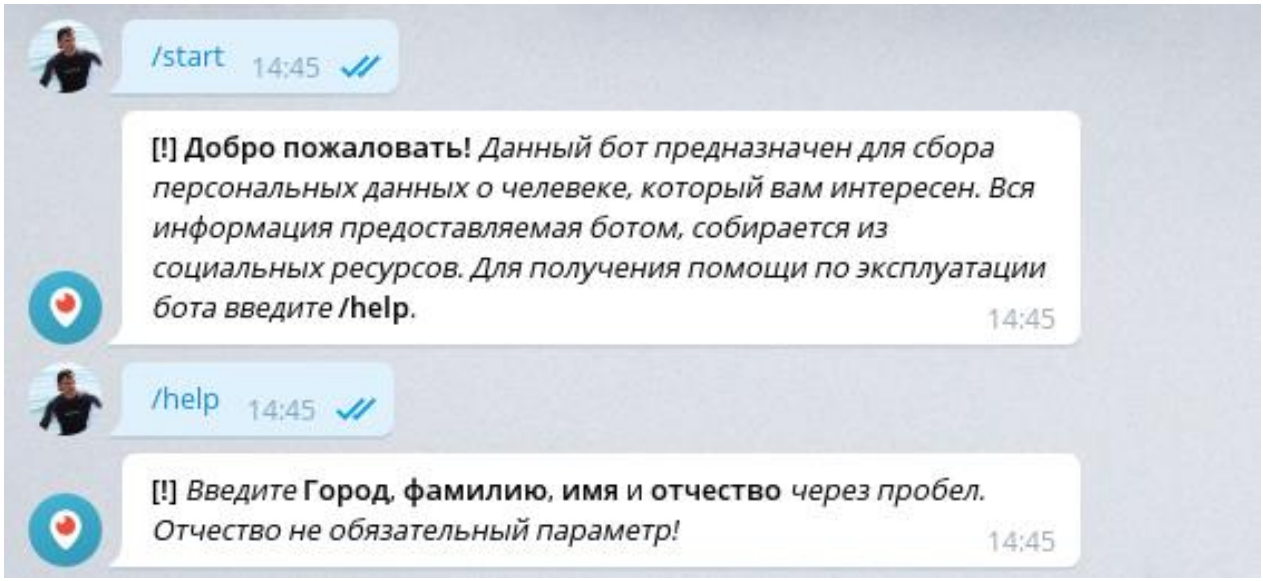


Рисунок 23 – Сообщение с информацией для пользователя

Если пользователь введет команду /help, бот выведет сообщение с информацией, что нужно ввести и в каком порядке, чтобы бот начал поиск нужного человека.

### 3.3 Система безопасного приватного доступа к боту PSinfo

Так как бот легко найти, и он не предназначен для публичного пользования, в коде реализована система приватного доступа. Это можно сделать, проверяя ID пользователя, который отправляет сообщения боту. В Telegram каждый пользователь имеет некоторые идентификаторы, такие как username, ID, first_name и так далее. Уникальным значением для каждого пользователя является ID, которое бот и будет проверять, рисунок 24.

Важным моментом в данном случае является то, что не все пользователи заполняют такие данные как username, first_name, last_name и так далее. Это может привести к возникновению ошибки в коде, которая в свою очередь приведет к остановке бота. Для исключения подобной ситуации в коде реализована обработка ошибок: в случае если при обращении к элементам массива, созданного посредством функции split, возникла ошибка, значит данный элемент массива отсутствует и его нужно заменить на некоторое

9значение. На рисунке 24 показана реализация обработки ошибок при их возникновении, в случае если ошибка появляется, бот присвоит переменной пустую строку и в таком случае такая переменная уже не будет равна null. В программе создан словарь chat_id_permission, в который будут записаны все ID, которым разрешено пользоваться ботом. Условие проверяет, есть ли ID пользователя, который отправил сообщение в словаре. Если ID пользователя отсутствует, то программа для данного пользователя не будет работать и бот выведет сообщение об отсутствии прав у пользователя, рисунок 24.

```
char_id_permission = {448688088}

if message.chat.id not in chat_id_permission:
        bot.send_message(message.chat.id,  str("Доступ  запрещен!  Обратитесь  к
администратору"))
else:
        personal_info = message.text.split(' ')

        try:
                city_from_user = personal_info[0]
        except Exception as error:
                pass
                city_from_user = ''

        try:
                last_name_from_user = personal_info[0]
        except Exception as error:
                pass
                last_name_from_user = ''


        try:
                first_name_from_user = personal_info[0]
        except Exception as error:
                pass
                first_name_from_user = ''
```

Рисунок 24 – Пример проверки ID пользователя

Можно использовать различные способы контролирования доступа к программному обеспечению, рисунок 25. Еще одним вариантом является использование пароля, хешированно хранящегося в базе данных или исходном коде, что не рекомендуется в целях безопасности. Пользователь будет передавать пароль, пароль переданный пользователем будет хешироваться посредством хеш функции и сравниваться с тем, который уже есть у бота. Если хеши совпадут, то пользователь получит доступ. При использовании данного метода необходимо реализовать защиту от брутфорс атаки. Для хеширования автором используется надежные алгоритмы шифрования, которые являются криптостойкими.

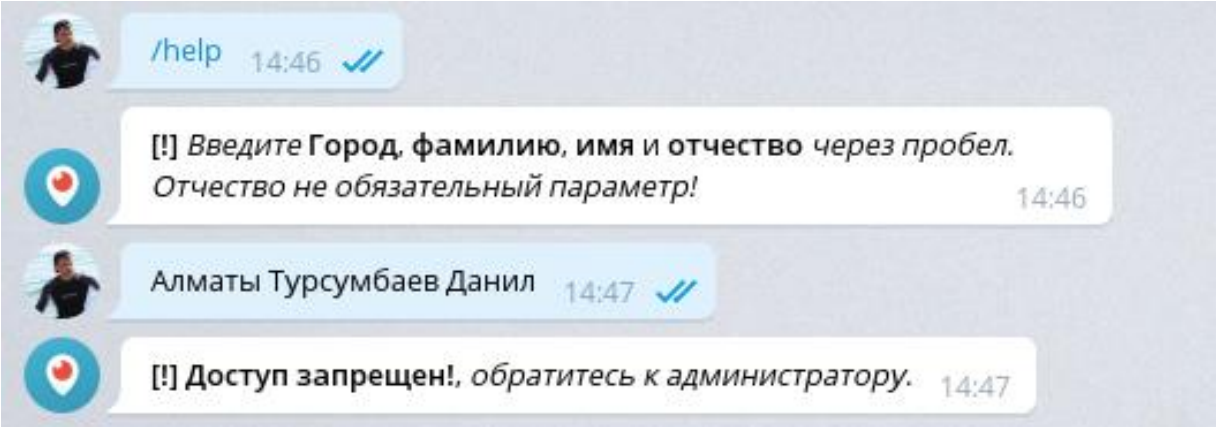


Рисунок 25 – Сообщение при отсутствии прав у пользователя

На рисунке выше показан пример выводимых сообщений пользователю, в случае отсутствия доступа у пользователя.

### 3.4 Логирование запросов к боту PSinfo

Данная реализация позволит администратору реагировать на странное поведение пользователей. Это так же позволит предотвращать попытки атак различного рода. В случае если пользователь или группа пользователей попытается провести атаки различных типов, это сразу отобразится в логах. Сигнатурами таких атак являются попытки вывести программное обеспечение из строя, путем подстановки различных символов и команд. Для SQL инъекций, это подстановка одинарных и двойных кавычек, использование команд UNION и SELECT. Для реализации XSS-атак злоумышленник должен попытаться внедрить JS код, стандартной проверкой наличия XSS-атак является выполнение команды alert. Так же для вывода программы из строя и получения ошибки, которая позволит узнать, куда передается переменная, злоумышленник может использовать внедрение массива. Для контролирования запросов к боту в программе реализована система логов. Все запросы записываются в отдельный журнал. Из запросов бот записывает следующую информацию: дата запроса, ID пользователя, username, first_name и сообщение, которое ввел пользователь. На рисунке 26 показан принцип логирования запросов пользователя. Логирование запросов пользователя – это еще один функционал позволяющий повысить информационную безопасность программного обеспечения. Используя данный функционал, администратор может в реальном времени отслеживать деятельность пользователей и блокировать пользователей совершивших попытку атаки на бота.

Часто пользователи Telegram не заполняют такие данные, как username, first_name, last_name и так далее. В результате если бот обратится к переменной, в которой должно быть записано значение username и она окажется пустой, произойдет ошибка. В результате ошибки бот прекратит свою работу. Для предотвращения подобной ошибки в программе реализована структура обработок ошибок Try–Except. Теперь, если пользователь не

заполнил username, но программа обратится к этой переменной, произойдет ошибка, программа отловит эту ошибку и предотвратит остановку бота, вместо остановки бота переменная будет заполнена значением None.

```
def log_information(user_infromation):
########################LOG#########################
        file = 'log.txt'
        open_file = open(file, mode = 'a')

        data_time = datetime.datetime.now()
        id = user_infromation.from_user.id

        try:
                first_name = user_infromation.from_user.first_name
        except Exception as error:
                pass
                first_name = "None"

        try:
                username = user_infromation.username
        except Exception as error:
                pass
                username = "None"

        try:
                text = user_infromation.text
        except Exception as error:
                pass
                text = "None"

        open_file.write("{} {} {} {} {} \n".format(data_time, id, first_name, username, text))
        open_file.close()
log_information(message)
```

Рисунок 26 – Код для логирования сообщений

При тестировании работоспособности программы было приведено несколько запросов для наглядности, производились попытки вызвать ошибку у бота, отправив кавычку и пустой массив, в результате бот сообщил, что данных недостаточно для поиска или данные введены некорректно и просто добавила запрос в логи. На рисунке 27 показаны примеры ошибочных логированных запросов. При проверке логов администратор сможет с легкостью обнаружить подобные запросы и произвести блокировку пользователя, путем исключения его ID из массива, который содержит ID имеющие доступ к боту.

```
2019-03-11 15:47:06.539974 448688088 danilkib95 None asdasd
2019-03-11 15:47:12.435267 448688088 danilkib95 None qweqweqwe
2019-03-11 15:48:03.927735 448688088 danilkib95 None lsdkflskfdlk
2019-03-11 15:48:07.931031 448688088 danilkib95 None Алматы
2019-03-11 15:48:12.317553 448688088 danilkib95 None Алматы Костерин
2019-03-11 15:48:15.989717 448688088 danilkib95 None '
2019-03-11 15:48:17.869003 448688088 danilkib95 None []
2019-03-12 14:46:13.032847 164746108 D3N None ololo
2019-03-12 14:46:53.255849 448688088 danilkib95 None Алматыва
2019-03-12 14:47:04.749123 448688088 danilkib95 None Алматы Турсумбаев Данил
```

Рисунок 27 – Логированные запросы

### 3.5 Меры безопасности для предотвращения распространенных атак

Для предотвращения реализации различного типа атак, таких как SQL-инъекции, удаленное выполнение кода, XSS-атак и так далее, были рассмотрены уязвимые функции Python, которые не желательно использовать при написании программного обеспечения.

Была установлена последняя версию Python, на текущий момент — это версия 3.6. Так как бот будет работать на Linux системе, необходимо сразу же провести все доступные обновления после установки. Часто такие системы поставляются с Python версии 2.7. Так как Python (точнее CPython) написан на C, бывают случае, когда сам интерпретатор имеет уязвимости. Основные проблемы безопасности в C связаны с распределением памяти и ошибками переполнения буфера. На рисунке 28 показана текущая версия Python.

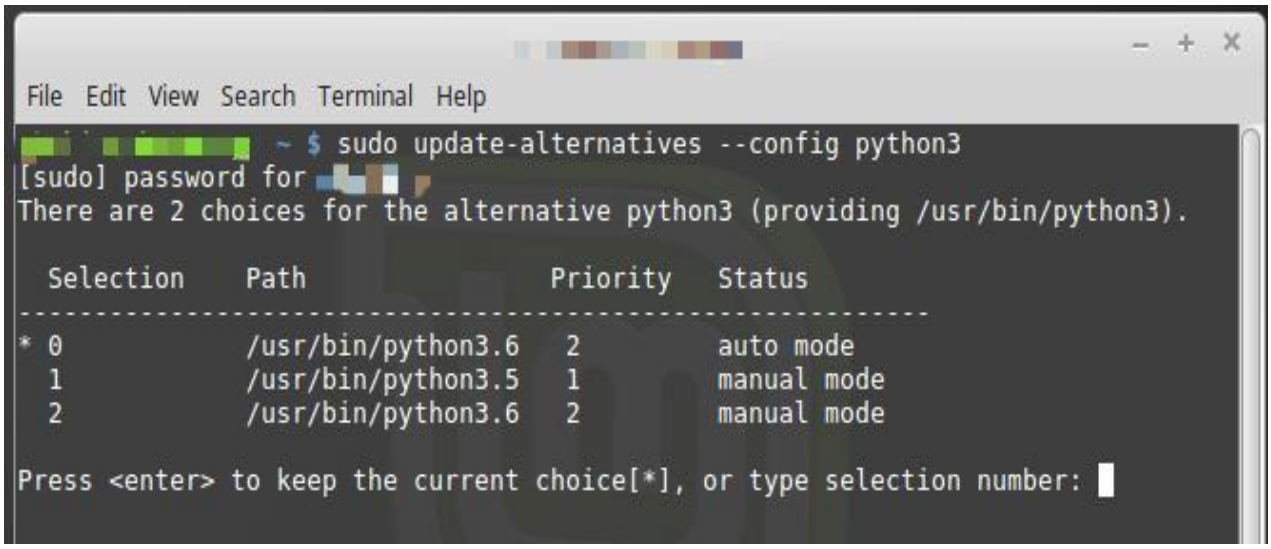


Рисунок 28 – Установленные версии Python

Как и интерпретатор, также были обновлены все зависимости. Существуют множество зависимостей, которые работают стабильно, но при этом содержат дыры в безопасности.

Так как бот не взаимодействует с базой данных, атаки типа SQL-инъекции исключаются, и нет необходимости реализовывать защиту от данного типа атак. Однако, в боте присутствует пользовательский ввод и данные передаются прямиком в код на обработку, это говорит о том, что необходимо позаботиться о предотвращении атаки типа удаленное выполнение кода. Для предотвращения атаки такого типа были изучены рекомендации, которые необходимо использовать для предотвращения RCE. Согласно рекомендациям часто такие атаки реализуются при форматировании строк, когда в строку необходимо внедрить какое-то значение. Изучены и применены несколько типов форматирования строк. На рисунке 29 показаны примеры форматирования строк в Python.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import telebot, requests, re, time, datetime

user_message = “ls –l”

#Способ первый
print(“Это ввод пользователя {}”.format(user_message))

#Способ второй
print(“Это ввод пользователя %s” % (user_message))

#Способ третий
print(f”Это ввод пользователя {user_message}”)
```

Рисунок 29 – Примеры использования форматированных строк

Из выше приведенных примеров, безопасным методом является первый метод и второй метод, так как они не позволяют производить удаленное выполнение кода, даже при внедрении пользовательского ввода. Метод 3 является не безопасным и при правильном его использовании злоумышленником это может привести к уязвимости типа RCE. Для предотвращения RCE во всем боте использован только первый метод формирования строк, рисунок 30.

```
def authorization(id_user):
	try:
		req_remixlhk = requests.get('https://m.vk.com/login', headers = {'Accept-
Language':'ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7'})
	except Exception as error:
		pass
		print(error)

	remixlhk = re.search('remixlhk=(.*?);', str(req_remixlhk.headers))
	lg_h = re.search('lg_h=(.*?)&', str(req_remixlhk.text))

	try:
		req_authorization =
requests.post('https://login.vk.com/?act=login&_origin=https://m.vk.com&lg_h={}&utf8=1'.for
mat(lg_h.group(1)), cookies = {'remixlhk':'{}'.format(remixlhk.group(1))}, data =
'email={}&pass={}'.format(login, password), allow_redirects = False)
	except Exception as error:
		pass
		print(error)

	__q_hash = re.search("__q_hash=(.*?)[';&]", str(req_authorization.headers))

	try:
		req_remixsid =
requests.get('https://m.vk.com/login?role=fast&to=&s=1&__q_hash={}'.format(__q_hash.group
(1)), allow_redirects = False)
	except Exception as error:
		pass
		print(error)
```

Рисунок 30 – Функция для авторизации из модуля для VK.com

### 3.6 Разбор главного модуля Telegram бота PSinfo

Главный модуль отвечает за взаимодействие всех модулей между друг другом, прием информации от пользователя и ее обработку и за вывод информации пользователю. На рисунке 31 приведена шапка главного модуля, данная шапка используется во всех других модулях, отличается шапка в каждом модуле только библиотеками, которые были импортированы. Первая строка указывает на используемый интерпретатор, который применяется, когда в системе установлено несколько версий Python. Вторая строка указывает на кодировку, которую автор использовал в коде, если не указать эту строку, то будет происходить ошибка при внедрении русских символов. Последующий строки кода указывают на библиотеки, которые импортированы в программу, без импорта данных библиотека бота не будет работать вовсе.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import telebot, requests, re, time, datetime
from telebot import types
from VK import authorization
from Yandex_search import get_account
from url_decode import urldecode
from spra_vkaru import get_letter_city
```

Рисунок 31 – Шапка главного модуля

Для работы бота необходим специальный идентификатор token, который идентифицирует бот. Данный идентификатор приведен на рисунке 32 первые две строки - это идентификатор и запуск бота. Далее идут декораторы, которые можно сравнить с триггерами, когда происходит некоторое событие, срабатывает декоратор и выполняется тело декоратора. Token - это уникальное значение, которое присваивается каждому боту. За генерацию токена отвечает Bot Father - специальный бот, который использован для регистрации своего бота. Токен состоит из 45 символов разного формата. Для составления токена используются буквы верхнего регистра, буквы нижнего регистра, цифры и специальные символы. Данные символы используются для исключения реализации атаки типа брутфорс. Брутфорс атака заключается в попытке перебора токена путем его генерации из символов перечисленных выше. Использование длины токена равной 45 символов и использование всех перечисленных символов обеспечивает надежную защиту от брутфорс атаки, так как перебрать такой токен практически невозможно на сегодняшний день. Часто токены используются не для подтверждения личности пользователя, а в качестве дополнительного параметра необходимому серверу при запросе.

```
token = '********1431:AAFg**********GqhfJnIvRDY****'
bot = telebot.TeleBot(token)

@bot.message_handler(commands=['help'])
def get_help(message):
        bot.send_message(message.chat.id, '<b>[!]</b> <i>Введите</i> <b>Город</b>, <b>фамилию</b>, <b>имя</b> и <b>отчество</b> <i>через пробел. Отчество не обязательный параметр!</i>', parse_mode = 'HTML')

@bot.message_handler(commands=['start'])
def get_help(message):
        bot.send_message(message.chat.id, '<b>[!] Добро пожаловать!</b> <i>Данный бот предназначен для сбора персональных данных о челевеке, который вам интересен. Вся информация предоставляемая ботом, собирается из социальных ресурсов. Для получения помощи по эксплуатации бота введите</i> <b>/help</b>.', parse_mode = 'HTML')

@bot.message_handler(content_types=['text'])
def get_text(message):
```

Рисунок 32 – Декораторы

Декоратор @bot.message_handler(commands=['help']) используется для оказания помощи пользователю, если пользователь вводить сообщение /help, отрабатывает тело декоратора и пользователю бот выведет сообщение с информацией о том как пользоваться ботом, рисунок 33.

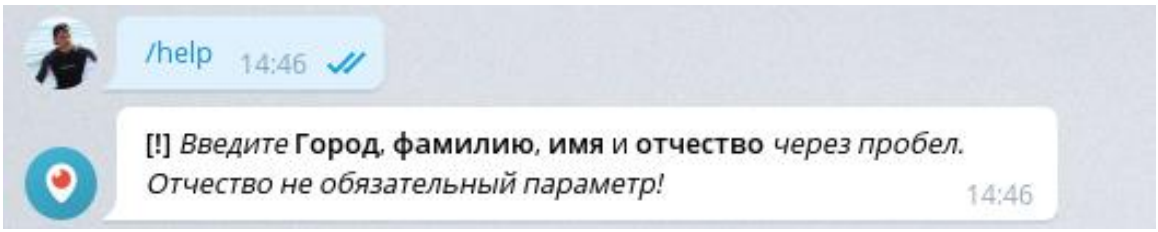


Рисунок 33 – Результат отработки команды /help

Декоратор @bot.message_handler(commands=['start']) также используется для вывода сообщения при первом запуске бота, также пользователь может вызвать данное сообщение, отправив команду /start, рисунок 34.


Рисунок 34 – Результат отработки команды /start

Декоратор @bot.message_handler(content_types=['text']) используется для приема сообщений от пользователя, если пользователь не ввел /help или /start, а ввел другое сообщение, бот задействует данный декоратор. Данный декоратор включает в себя основной функционал главного модуля. При вводе сообщения от пользователя первым делом бот выполнит функцию логирования запросов от пользователя. Далее бот проверит, имеет ли доступ пользователь к данному боту. Если у пользователя нет доступа, бот выдаст соответствующее сообщение. Если у пользователя есть доступ к боту, бот начинает проверку сообщения, которое ввел пользователь. Когда все проверки пройдены, бот произведет последовательный вызов функций, каждой функции соответствует отдельный модуль, рисунок 35.

```
yandex(last_name_from_user,  first_name_from_user)

spravochnik(city_from_user,  last_name_from_user)
```

Рисунок 35 – Вызов функций

Первым делом бот вызовет функцию с названием yandex, данная функция принимает 2 параметра фамилию и имя пользователя, рисунок 36.

```
def yandex(last_name,  first_name):
########################Yandex#######################
        accounts = get_account(last_name,  first_name)
        not_vk = []

        for account in accounts:
                account = str(urldecode(account))

                if 'vk.com' in account:
                        vk(account)
                else:
                        not_vk.append(account)

        for no_vk in not_vk:
                keyboard = types.InlineKeyboardMarkup()
                url_button = types.InlineKeyboardButton(text="Перейти к аккаунту",
url="{}".format(no_vk))
                keyboard.add(url_button)
                bot.send_message(message.chat.id,  "<b>[+] Account: {}</b>".format(no_vk),
reply_markup = keyboard, parse_mode = 'HTML')
```

Рисунок 36 – Функция yandex

Функция yandex находит все аккаунты, которые есть у искомого человека и проверяет, относится ли этот аккаунт к социальной сети «Вконтакте». Функции vk передается ID аккаунта пользователя, вызов функции vk показан на рисунке 37.

```
for account in accounts:
        account = str(urldecode(account))

        if 'vk.com' in account:
                vk(account)
        else:
                not_vk.append(account)
```

Рисунок 37 – Код для вызова функции vk

Если аккаунт относится к социальной сети «Вконтакте», то бот вызывает функцию vk, рисунок 38.

```
def vk(acc):
########################VK#########################
        keyboard = types.InlineKeyboardMarkup()
        url_button = types.InlineKeyboardButton(text="Перейти к аккаунту",
url="{}".format(acc))
        keyboard.add(url_button)

        id = re.search('vk.com/(.*?)`', str(acc) + '`')

        data_from_vk = authorization(id.group(1))

        bot.send_message(message.chat.id, str("<b>[+] Account: {}</b>".format(acc)) + "\n" +
str('[+] <i>Полное имя</i>: {}'.format(data_from_vk['full_name'])) + "\n" + str('[+] <i>Дата
рождения</i>: {} {}'.format(data_from_vk['month'], data_from_vk['year'])) + "\n" + str('[+]
<i>Город</i>: {}'.format(data_from_vk['city'])) + "\n" + str('[+] <i>Место работы</i>:
{}'.format(urldecode(data_from_vk['work']))) + "\n" + str('[+] <i>Образование</i>:
{}'.format(urldecode(data_from_vk['education']))) + "\n" + str('[+] <i>Телефон</i>:
{}'.format(urldecode(data_from_vk['tel']))), reply_markup = keyboard, parse_mode = 'HTML')
```

Рисунок 38 – Функция vk

Как только бот выполнит функцию vk, курсор возвратится в функцию yandex. Уже в функции yandex бот будет выводить информацию по аккаунтам, которую вернула функция vk, рисунок 39.

```
    for no_vk in not_vk:
            keyboard = types.InlineKeyboardMarkup()
            url_button = types.InlineKeyboardButton(text="Перейти к аккаунту",
url="{}".format(no_vk))
            keyboard.add(url_button)
            bot.send_message(message.chat.id, "<b>[+] Account: {}</b>".format(no_vk),
reply_markup = keyboard, parse_mode = 'HTML')
```

Рисунок 39 – Вывод данных пользователю

Бот предоставляет пользователю все найденные данные в форматированном виде. Также после каждого аккаунта бот выводит кнопку, нажав на которую пользователь может перейти к данном аккаунту, рисунок

36. Для более читабельного вида в коде использованы HTML теги, которые позволили выделить некоторую информацию для привлечения внимания пользователя. Первая строка вывода сообщает пользователю о найденном аккаунте и выводит url адрес этого аккаунта, данная строка выводится жирным шрифтом. Далее бот выводит информацию, которая содержится на странице данного аккаунта, вся информация такого рода будет выведена курсивом, рисунок 40.

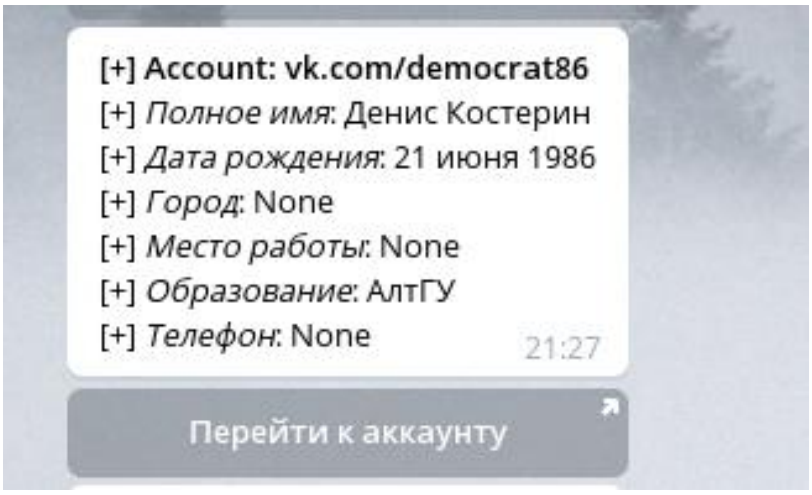


Рисунок 40 – Вывод информации из социальной сети «Вконтакте»

После того как бот выведет все аккаунты связанные с социальной сетью «Вконтакте» и всю найденную информацию по аккаунтам, бот выведет все остальные найденные аккаунты, рисунок 41.

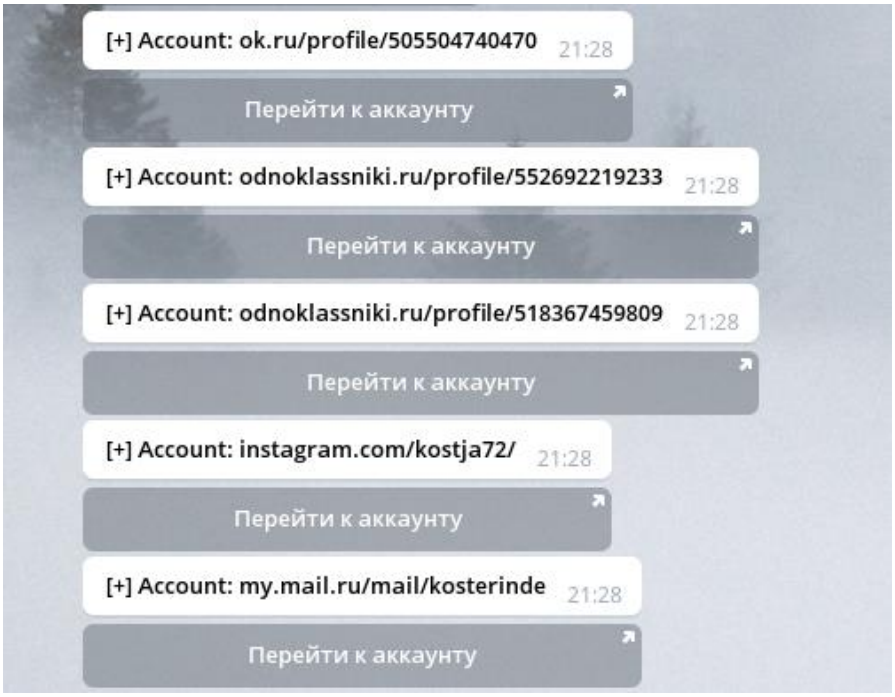


Рисунок 41 – Найденные аккаунты

После того, как отработает функция с именем yandex, бот вызовет функцию с именем spravochnik. Данная функция используется для поиска более конкретной информации о пользователе, рисунок 42. Функции необходимо передать два параметра - это город проживания человека и его фамилию. Если пользователь не знает город проживания искомого человека, то ему все равно необходимо передать хотя бы примерный город проживания человека, так как бот первым параметром ожидает ввода именно города. Такой подход реализован потому что, бот не может определить, что из параметров ввел пользователь - отчество или фамилию, фамилию или имя. Если введен город, то бот может определить на основе словаря, содержащего все города Казахстана.

```
def spravochnik(city, surname):
####################SPRAVOCHNIK#################
        data_from_spravka = get_letter_city(surname, city)

        bot.send_message(message.chat.id, "<b>[+] Справочник:</b>", parse_mode = 'HTML')

        for line in data_from_spravka:
        data = data_from_spravka[line].split(';')

        bot.send_message(message.chat.id, str("[+] <i>Полное имя</i>: {}".format(data[1])) +
"\n" + str("[+] <i>Дом. номер тел.</i>: {}".format(data[0])) + "\n" +str("[+] <i>Улица</i>:
{}".format(data[2])) + "\n" + str("[+] <i>Дом</i>: {}".format(data[3])) + "\n" +str("[+]
<i>Квартира</i>: {}".format(data[4])), parse_mode = 'HTML')
```

Рисунок 42 – Функция для работы со справочником

Функция принимает два параметра: city – город искомого человека и surname – фамилию искомого человека. Далее бот вызывает модуль справочника и передает ему эти два параметра. Как только бот отработает модуль справочника, функции вернется найденная информация по пользователю, которая будет обработана и бот выведет информацию пользователю в форматированном виде.

Последние строчки кода главного модуля показаны на рисунке 43. Строки кода предотвращают завершение программы в моменты простоя, т.е. тогда, когда пользователи не совершают запросы боту. Если не прописать данный код, то при запуске бота он сразу же будет завершаться, так как бот работает посредством декораторов и должен ожидать определенной команды от пользователя.

```
if __name__ == '__main__':
        bot.polling(none_stop = True)
```

Рисунок 43 – Предотвращение завершения работы бота

### 3.7 Разбор модуля Yandex_search Telegram бота PSinfo

Данный модуль является самым маленьким из всех модулей, на рисунке 44 показан весь код модуля. Вызывая данные, бот создает специально сформированный запрос в поисковую систему «Яндекс», с подстановкой параметров, переданных пользователем.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import requests, re

def get_account(last_name, first_name):
    req = requests.get('https://yandex.ru/people?ajax=%7B%22advanced-
search%22%3A%7B%7D%2C%22amt%22%3A%7B%22las%22%3A%22b-page__counter-
open%3Bb-page__counter-open%3Bcolorize-1s59zyf%3Bcolorize-e-1c260h%3Bcolorize-p-
qmspga%3Bfavicons%3Bfavicons%3Boverlay_base%3Bpromo-
popup_theme_ffffff%22%7D%2C%22b-
cntrs%22%3A%7B%7D%2C%22bmt%22%3A%7B%22lb%22%3A%220af0bb0bx0br0bk0c40
3q0bn0bq04w0dm0ge03p0c30is0by0hp0hx0hs0bp00q0100150in08n08t08r0fu02901f01100r00u
05k0cg0ek0010040gp0gu0ew08b0cu0cv0ep02b0ag00900b0gl00t01405y06b0kr%22%7D%2C%
22extra-content%22%3A%7B%22advanced-search__input_tooltip-
popup%22%3A1%7D%2C%22footer__settings%22%3A%7B%7D%2C%22log%22%3A%7B
%22event%22%3A%22replace%22%2C%22parentId%22%3A%22g0nj%22%2C%22parentReq
id%22%3A%221550385337764376-819000089617354421800035-sas1-
1427%22%7D%2C%22main%22%3A%7B%7D%2C%22main__carousel%22%3A%7B%22pat
hLinksCount%22%3A0%7D%2C%22main__direct-abuse-
modal%22%3A%7B%7D%2C%22main__distr-
popup%22%3A%7B%22exists%22%3Afalse%7D%2C%22main__result%22%3A%7B%7D%2
C%22main__serp-
auth%22%3A%7B%7D%2C%22metrika%22%3A%5B%22731962%22%5D%2C%22navigatio
n%22%3A%7B%7D%2C%22search2__ajax%22%3A%7B%7D%2C%22search2__input%22%
3A%7B%7D%2C%22serp%22%3A%7B%7D%2C%22suggest2-
history%22%3A%7B%7D%7D&text={}+{}'.format(last_name, first_name))

    data = re.findall('fmode=inject&amp;url=%2F%2F(.*?)&amp', str(req.text))
    return data
```

Рисунок 44 – Модуль Yandex-search

После того, как бот совершит запрос на поиск человека, бот задействует регулярные выражения, которые соберет всех найденные аккаунты и разместит их в массиве data, рисунок 45. Как только бот выполнит регулярное выражение и сохранит все данные в массиве, бот задействует функцию return, которая возвратит данный массив в главный модуль.

```
data = re.findall('fmode=inject&amp;url=%2F%2F(.*?)&amp', str(req.text))
return data
```

Рисунок 45 – Регулярное выражение

Можно заметить, что шапка у данного модуля такая же, как и у главного модуля, отличается шапка только импортами библиотек.

### 3.8 Разбор модуля VK Telegram бота PSinfo

Данный модуль работает с социальной сетью «Вконтакте», модуль включается в себя 2 большие функции.

На рисунке 46 показаны заголовки модуля vk. Первые 2 строки являются стандартными для всех модулей, далее идут импорты необходимых библиотек для данного модуля. После стандартных строк в данном модуле объявлены переменные, необходимые для работы модуля: password – пароль к аккаунту, login – логин к аккаунту, data – словарь, который будет содержать информацию, которую возвратит модуль. Авторизация в данном случае необходима так как, если бот не авторизуется, он получит меньше информации об искомом человеке.

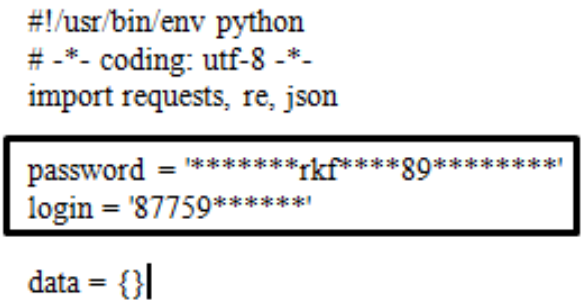


Рисунок 46 – Заголовок модуля vk

На рисунке 47 показана функция авторизация. Использую данную функцию, бот проходит авторизацию в социальной сети «Вконтакте», на основе пароля и логина, показанных на предыдущем рисунке. Чтобы пройти авторизацию, бот собирает множество параметров, которые требуются при авторизации помимо логина и пароля. Авторизация в социальной сети «Вконтакте» является сложной и для ее прохождения необходимо предоставить различную информацию. Для авторизации бот соберет множество идентификаторов, чтобы предстать перед серверами в качестве валидного пользователя. Помимо всех идентификаторов также понадобятся уникальные данные пользователя – это пароль и логин. После авторизации на стороне сервера будет создан специальный токен, который будет является уникальным и будет соответствовать только одному пользователю. Данный токен будет отправлен пользователю, прошедшему авторизацию. Токен на стороне сервера называется session, токен на стороне клиента называется cookies. При запросах пользователь будет предоставлять cookies серверу, а сервер тем временем будет просматривать имеется ли соответствующая session для данного пользователя. Если session есть на стороне сервера, запрос отработает.

```
def authorization(id_user):
    try:
        req_remixlhk = requests.get('https://m.vk.com/login', headers = {'Accept-
Language':'ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7'})
    except Exception as error:
        pass
        print(error)

    remixlhk = re.search('remixlhk=(.*?);', str(req_remixlhk.headers))
    lg_h = re.search('lg_h=(.*?)&', str(req_remixlhk.text))

    try:
        req_authorization = requests.post('https://login.vk.com/?act=login&_origin
=https://m.vk.com&lg_h={}&utf8=1'.format(lg_h.group(1)), cookies =
{'remixlhk':'{}'.format(remixlhk.group(1))}, data = 'email={}&pass={}'.format(login,
password), allow_redirects = False)
    except Exception as error:
        pass
        print(error)

    __q_hash = re.search("__q_hash=(.*?)[';&]", str(req_authorization.headers))

    try:
        req_remixsid = requests.get('https://m.vk.com/login?role=fast&to=&s=1&
__q_hash={}'.format(__q_hash.group(1)), allow_redirects = False)
    except Exception as error:
        pass
        print(error)

    remixsid = re.search(' remixsid=(.*?);', str(req_remixsid.headers))

    get_information(remixsid.group(1), id_user)
    return data
```

Рисунок 47 – Функция авторизации

Первый параметр, который необходим боту для авторизации это remixlhk, данный параметр бот получает в результате запроса, показанного на рисунке 48. Бот совершает запрос на страницу авторизации социальной сети «Вконтакте», с обязательным заголовком Accept-Language, если не указать данный заголовок, запрос не пройдет. После того, как запрос прошел, бот получает ответ от сервера с контентом и уже, используя полученный контент, может собрать параметр remixlhk из контента посредством регулярного выражения.

```
    try:
        req_remixlhk = requests.get('https://m.vk.com/login', headers = {'Accept-
Language':'ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7'})
    except Exception as error:
        pass
        print(error)

    remixlhk = re.search('remixlhk=(.*?);', str(req_remixlhk.headers))
```

Рисунок 48 – Получение параметра remixlhk

Вторым параметром, который необходим для авторизации является параметр lg_h. Данный параметр бот собирает, используя тот же самый контент, полученный в результате первого запроса. Параметр lg_h бот собирает посредством регулярного выражения, рисунок 49.

```
lg_h = re.search('lg_h=(.*?)&', str(req_remixlhk.text))
```

Рисунок 49 – Получение параметра lg_h

Как только бот соберет все необходимые параметры для авторизации (lg_h, password, login, remixlhk), бот совершит запрос на авторизацию, подставив все перечисленные параметры в запрос авторизации, рисунок 50.

```
try:
        req_authorization = requests.post('https://login.vk.com/?act=login&_origin
=https://m.vk.com&lg_h={}&utf8=1'.format(lg_h.group(1)), cookies =
{'remixlhk':'{}'.format(remixlhk.group(1))}, data = 'email={}&pass={}'.format(login,
password), allow_redirects = False)
    except Exception as error:
        pass
        print(error)
```

Рисунок 50 – Запрос на авторизацию

После того как бот совершит запрос на авторизацию, сервер должен вернуть параметр __q_hash, данный параметр необходим для получения конечного токена. Параметр __q_hash бот собирает посредством регулярного выражения из контента запроса авторизации, рисунок 51.

```
__q_hash = re.search("__q_hash=(.*?)[';;&]", str(req_authorization.headers))
```

Рисунок 51 – Получение параметра __q_hash

Используя параметр __q_hash, бот совершает последний запрос, в результате которого сервер возвратит конечный токен (remixsid), который будет подтверждать авторизацию пользователя, рисунок 52.

```
try:
        req_remixsid = requests.get('https://m.vk.com/login?role=fast&to=&s=1&
__q_hash={}'.format(__q_hash.group(1)), allow_redirects = False)
    except Exception as error:
        pass
        print(error)
```

Рисунок 52 – Запрос на получения токена remixsid

Токен remixsid бот собирает посредством регулярного выражения, рисунок 53. После получения токена бот вызывает вторую функцию данного модуля и передает ей основной токен remixsid.

```
remixsid = re.search(' remixsid=(.*?);', str(req_remixsid.headers))

get_information(remixsid.group(1), id_user)
return data
```

Рисунок 53 – Получение токена remixsid

Токен remixsid – это уникальный параметр, который бот получит после авторизации на сайте. Данный токен передается серверу с каждым запросом бота, передавая данный токен, бот подтверждает свою личность перед сервером, тем самым получает доступ к аккаунту, под которым проходил авторизацию.

После того как бот получил токен remixsid, бот перейдет к выполнению второй функции данного модуля. Основной функционал второй функции приведен на рисунке 54.

```
def get_information(remixsid, id_user):

    page_info = requests.get('https://vk.com/{}'.format(id_user), cookies = {'remixsid':remixsid})

    full_name = re.search('<title>(.*?)</title>', str(page_info.text))
    year = re.search('byear]=([0-9]+)["'\s>&?]', str(page_info.text))
    month_day = re.search('bmonth]=[a-zA-Z0-9]+">(.*?)</a>', str(page_info.text))
    city = re.search('city]=[0-9a-zA-Z]+">(.*?)</a>', str(page_info.text))
    education = re.search('university]=[0-9]+">(.*?)</a>', str(page_info.text))
    work = re.search('company]=(.*?)">', str(page_info.text))
    tel = re.search('labeled">([0-9\+]+)</div>', str(page_info.text))
```

Рисунок 54 – Основной функционал второй функции

Функция на рисунке 55 принимает два параметра: remixsid и id_user – id аккаунта искомого человека. ID аккаунта бот находит посредством модуля yandex. Получив два необходимых параметра, бот совершает запрос на главную страницу переданного аккаунта.

```
page_info = requests.get('https://vk.com/{}'.format(id_user), cookies = {'remixsid':remixsid})
```

Рисунок 55 – Запрос на главную страницу аккаунта

После запроса, получив ответ от сервера, бот, используя контент ответа, начинает собирать со страницы информацию, рисунок 56.

```
full_name = re.search('<title>(.*?)</title>', str(page_info.text))
year = re.search('byear]=([0-9]+)["\s>&?]', str(page_info.text))
month_day = re.search('bmonth]=[a-zA-Z0-9]+">(.*?)</a>', str(page_info.text))
city = re.search('city]=[0-9a-zA-Z]+">(.*?)</a>', str(page_info.text))
education = re.search('university]=[0-9]+">(.*?)</a>', str(page_info.text))
work = re.search('company]=(.*?)">', str(page_info.text))
tel = re.search('labeled">([0-9\+]+)</div>', str(page_info.text))
```

Рисунок 56 – Сбор информации из контента ответа на запрос

После того как бот выполнит все регулярные выражения, бот начинает загружать значения в массив data, чтобы передать всю информацию целиком, одним объектом, рисунок 57.

```
try: data['full_name'] = full_name.group(1)
except Exception as eror:
        pass
        data['full_name'] = 'None'
try: data['year'] = year.group(1)
except Exception as error:
        pass
        data['year'] = 'None'
try: data['month'] = month_day.group(1)
except Exception as error:
        pass
        data['month'] = 'None'
try: data['city'] = city.group(1)
except Exception as error:
        pass
        data['city'] = 'None'
try: data['work'] = work.group(1)
except Exception as error:
        pass
        data['work'] = 'None'
try: data['education'] = education.group(1)
except Exception as error:
        pass
        data['education'] = 'None'
try: data['tel'] = tel[0]
except Exception as error:
        pass
        data['tel'] = 'None'
try: data['tel_dop'] = tel.group(1)
except Exception as error:
        pass
        data['tel_dop'] = 'None'
```

Рисунок 57 – Сбор всей информации в массив data

В приведенном коде видно, что если при обращении бота к переменной, организованной посредством регулярного выражения, возникает ошибка, то бот обрабатывает ошибку и не дает завершиться программе, вместо завершения бот записывает в поле массива строку None.

### 3.9 Разбор модуля spra_vkaru Telegram бота PSinfo

Данный модуль используется ботом для поиска более конкретной информации о пользователе. Для работы модуля пользователь должен ввести два обязательных значения: фамилию искомого человека и город проживания искомого человека. Данный модуль является более большим и содержит множество различных словарей. На рисунке 58 показана шапка модуля, которая практически не отличается от других модулей.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import requests, re
import time
```

Рисунок 58 – Шапка модуля spra_vkaru

Для работы с данным модулем бот использует средства анонимности, каждый запрос к веб-ресурсу проходит через сеть Tor. Для работы с сетью Tor в коде прописан словарь с прокси конфигурацией, рисунок 59.

```
proxies = {
  'http': 'socks5://127.0.0.1:9150',
  'https': 'socks5://127.0.0.1:9150'
}
```

Рисунок 59 – Конфигурация прокси

Структуру, приведенную выше, бот подставляет в каждый запрос, в параметр proxies, тем самым повышая анонимность запросов, рисунок 60.

```
req = requests.get('http://****** /****/**/{}/{}'.format(city_number, list_number), proxies =
proxies, headers = head, cookies = cookie)
```

Рисунок 60 – Внедрение прокси в запрос

Для выполнения запросов бот снабжает каждый запрос определенными заголовками, рисунок 61. Некоторые из этих заголовков являются обязательными параметрами, так как без них бот не сможет делать запросы.

Остальные заголовки бот использует для повышения анонимности, заменяя системные значения, на системные значения другого устройства.

```
head = {
        'User-Agent': 'Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_10_1; en-US)
AppleWebKit/538.10.34 (KHTML, like Gecko) Version/12.1.5 Safari/538.10.34',

        'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',

        'Accept-Language': 'ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3',

        'Accept-Encoding': 'gzip, deflate'
}
```

Рисунок 61 – Заголовки для запросов

Заголовки, приведенные в коде, бот подставляет в каждый запрос, в параметр headers, рисунок 62. В данном случае, помимо прокси в системе, на которой запущен бот, включен VPN. Получается связка VPN + Tor, каждый элемент данной связки имеет свое значение. VPN используется для шифрования трафика, создающегося в процессе работы бота. Используя Tor, бот пропускает запрос через множество хостов, в результате серверу видны характеристики последней системы, через которую прошел бот.

```
req = requests.get('http://******  /****/**/{}/{}'.format(city_number, list_number), proxies =
proxies, headers = head, cookies = cookie)
```

Рисунок 62 – Внедрение заголовков в запрос

Еще одним параметром бот передает cookies, это так же обязательные параметры без которых запросы бота не будут работать, рисунок 63.

```
cookie = {
            '__cfduid':'d6c3703688c0a028cb9a46e5e410492e41550507567;',

            'hibext_instdsigdipv2':'1;',

            '_ga':'GA1.2.1816151957.1550507635;',

            '_gid':'GA1.2.1195866266.1550507638;'
        }
```

Рисунок 63 – Cookies для запросов

Заголовки, приведенные в коде, подставляются в каждый запрос в параметр cookies, рисунок 64. Без предоставления данного заголовка сервер не

будет принимать запрос. В результате можно не получить контент ответа сервера на запрос.

```
req = requests.get('http://******  /****/**/{}/{}'.format(city_number, list_number), proxies =
proxies, headers = head, cookies = cookie)
```

Рисунок 64 – Внедрение cookies в запрос

Для управления запросами в модуле внедрено еще два дополнительных словаря, один из которых содержит города и их идентификаторы, второй содержит буквы и их идентификаторы. Дело в том, что в ссылках на данном ресурсе используются не название городов и не буквы, а целочисленные идентификаторы. Словарь с городами и идентификаторами показан на рисунке 65.

```
cities = {
'Астана': '7172',                'Алматы': '727',
'Кокшетау': '7162',              'Актюбинск': '7132',
'Алга': '71337',                 'Бадамша': '71342',
'Байганин': '71345',             'Кандыагаш': '71333',
'Комсомольское': '71339',        'Мартук': '71331',
'Хобда': '71341',                'Хромтау': '71336',
'Шалкар': '71335',               'Шубар-кудук': '71346',
'Эмба': '71334',                 'Талдыкорган': '72822',
'Атырау': '7122',                'Семипалатинск': '7222',
'Тараз': '7262',                 'Уральск': '7112',
'Караганда': '7212',             'Костанай': '7142',
'Кызылорда': '7242',             'Аральск': '72433',
'Жалагаш': '72431',              'Жанакорган': '72435',
'Кармакчи': '72437',             'Казалинск': '72438',
'Теренозек': '72436',            'Шиели': '72432',
'Павлодар': '7182',              'Аксу': '71837',
'Экибастуз': '7187',             'Петропавловск': '7152',
'Актау': '7292',                 'Шымкент': '7252'
}
```

Рисунок 65 – Словарь с городами

В качестве букв бот так же использует идентификаторы. Каждой букве соответствует определенное целочисленное значение, рисунок 66. Первым в ссылку бот подставляет идентификатор города. После того как бот подставил идентификатор города, он циклически перебирает 3 первые буквы фамилии искомого человека и собирает идентификатор для начала ссылки. Иногда бот может получить информацию сразу по нескольким людям, имеющим одинаковую фамилию, бот выведет нескольких людей.

```
letter = {
        'А':'1',    'Б':'2',    'В':'3',    'Г':'4',
        'Д':'5',    'Е':'6',    'Ж':'8',    'З':'9',
        'И':'10',   'Й':'11',   'К':'12',   'Л':'13',
        'М':'14',   'Н':'15',   'О':'16',   'П':'17',
        'Р':'18',   'С':'19',   'Т':'20',   'У':'21',
        'Ф':'22',   'Х':'23',   'Ц':'24',   'Ч':'25',
        'Ш':'26',   'Щ':'27',   'Ы':'29',   'Э':'31',
        'Ю':'32',   'Я':'33',   'а':'1',    'б':'2',
        'в':'3',    'г':'4',    'д':'5',    'е':'6',
        'ж':'8',    'з':'9',    'и':'10',   'й':'11',
        'к':'12',   'л':'13',   'м':'14',   'н':'15',
        'о':'16',   'п':'17',   'р':'18',   'с':'19',
        'т':'20',   'у':'21',   'ф':'22',   'х':'23',
        'ц':'24',   'ч':'25',   'ш':'26',   'щ':'27',
        'ы':'29',   'э':'31',   'ю':'32',   'я':'33'
        }
```

Рисунок 66 – Словарь с буквами

Модуль включает в себя 3 функции, функция, приведенная на рисунке 67 используется ботом для создания ссылки, на искомого человека. На основе словарей, приведенных выше бот собирает идентификаторы и организует из них ссылку.

```
def get_letter_city(surname, city):
        lets = ''.join(surname)

        city_number = cities['{}'.format(city)]

        list_number = 'list{}_{}_{}'.format(letter['{}'.format(lets[0])], letter['{}'.format(lets[1])],
letter['{}'.format(lets[2])])

        get_pages(city_number, list_number, surname)
        return data_from_spravka
```

Рисунок 67 – Функция для организации ссылки

Далее бот вызывает функцию get_pages, передавая ей две переменные city_number и list_number, рисунок 68.

```
get_pages(city_number, list_number, surname)

return data_from_spravka
```

Рисунок 68 – Вызов функции get_pages

На рисунке 69 показана функция get_pages, которую бот вызывает для получения идентификатора страницы пользователя.

```
def get_pages(city_number, list_number, surname):
        req = requests.get('http://*******/****/7/{}/{}'.format(city_number, list_number),
proxies = proxies, headers = head, cookies = cookie)

        count_page = re.findall("{}/page".format(list_number), str(req.text))
        mass_with_pages = []

        count_page = len(count_page) + 2

        for i in range(1, count_page, 1):
                req = requests.get('http://********/*****/7/{}/{}/page{}'.format( city_number,
list_number, i), proxies = proxies, headers = head, cookies = cookie)

                number_page = re.findall("href='/******/7/{}/([0-9]+)/'>{}</a>".format(
city_number, surname), str(req.text))

        if len(number_page) != 0:
                        break

        try:
                get_info(number_page[0], city_number)
        except Exception as error:
                pass
                return
        return
```



Рисунок 69 – Функция get_pages

Попав в функцию get_pages, бот совершает запрос, подставляя собранные идентификаторы пользователей. Используя контент ответа, бот, посредством регулярного выражения, собирает количество страниц, выданное в результате ответа и записывает значение в переменную count_page, рисунок 70.

```
        req = requests.get('http://*******/****/7/{}/{}'.format(city_number, list_number),
proxies = proxies, headers = head, cookies = cookie)

        count_page = re.findall("{}/page".format(list_number), str(req.text))
        mass_with_pages = []

        count_page = len(count_page) + 2
```

Рисунок 70 – Запрос для определения количества страниц

Определив количество страниц, бот запускает цикл и начинает перебирать все найденные страницы. Перебирая страницы, бот ищет фамилию, которую ввел пользователь. Найдя номер страницы, на которой находится нужная фамилия, бот спарсит ссылку на данную страницу и запишет её в переменную number_page, рисунок 71.

```
for i in range(1, count_page, 1):
        req = requests.get('http://********/*****/7/{}/{}/page{}'.format( city_number,
list_number, i), proxies = proxies, headers = head, cookies = cookie)

        number_page = re.findall("href='/******/7/{}/([0-9]+)/'>{}</a>".format(
city_number, surname), str(req.text))
```

Рисунок 71 – Перебор страниц и поиск необходимой ссылки на пользователя

В качестве точки остановки цикла для бота является не пустая переменная number_page. Как только в number_page будет записано некоторое значение, бот прервет цикл и вызовет функцию get_info, рисунок 72. В случае если бот не найдет фамилию в результате перебора страниц, при вызове функции get_info произойдет ошибка, которая будет обработана структурой Try–Except и бот вызовет функцию return.

```
if len(number_page) != 0:
        break

try:
        get_info(number_page[0], city_number)
except Exception as error:
        pass
        return
return
```

Рисунок 72 – Принцип остановки цикла и вызов функции get_info

Когда бот нашел нужную страницу, он переходит к выполнению функции get_info. Функция get_info используется ботом для сбора информации с найденной страницы. Функция get_info показана на рисунке 73. Функция get_info выполняет основную задачу по сбору информации и является самой большой в данном модуле. Совершив запрос по собранному url, бот получает контент страницы, на которой находится основная информация по искомому человеку. Используя полученный контент и регулярные выражения, бот будет собрать данные со страницы в цикле. Цикл необходим, так как пользователей с одной и той же фамилией может быть

несколько, бот переберет все фамилии, подходящие под искомую, и соберет информацию по каждой. Вывод бота по каждой фамилии будет отдельным.

```
def get_info(number_page, city_number):
        req = requests.get('http://********/******/7/{}/{}/'.format(city_number, number_page),
proxies = proxies, headers = head, cookies = cookie)

        bloks = re.findall('<tr>\s+<td><a(.*?)</td>\s+</tr>', str(req.text), re.DOTALL|re.M)

        i = 1

        for blok in bloks:
                number    =    re.search("peoples/7/{}/[0-9]+/'>([0-9]+)</a></td>".format(city_
number), blok)
                full_name   =   re.search("peoples/7/{}/[0-9]+/'>([\sA-Яа-яёЁ_-]+)</a>".format(
city_number), blok)
                street = re.search("/streets/7/{}/[0-9]+/'>(.*?)</a>".format(city_number), blok)
                home = re.search("/streets/7/{}/[0-9]+/[0-9]+/'>(.*?)</a>".format(city_number),
blok)
                kv        =        re.search("/streets/7/{}/[0-9]+/[0-9]+/'>.+</a>,\s[а-яА-Я]+.\s([0-
9]+)".format(city_number), blok)
```



Рисунок 73 – Функция get_info

Функция get_info принимает два значения: ID города и ID страницы, которую необходимо спарсить. Попав в функцию get_info, первым делом бот совершает запрос, который формирует из параметров переданных функции get_info, рисунок 74.

```
def get_info(number_page, city_number):
        req = requests.get('http://********/******/7/{}/{}/'.format(city_number, number_page),
proxies = proxies, headers = head, cookies = cookie)
```

Рисунок 74 – Запрос для получения контента с персональными данными

На рисунке 75 показан пример ответа в браузере на запрос, происходящий на рисунке выше.

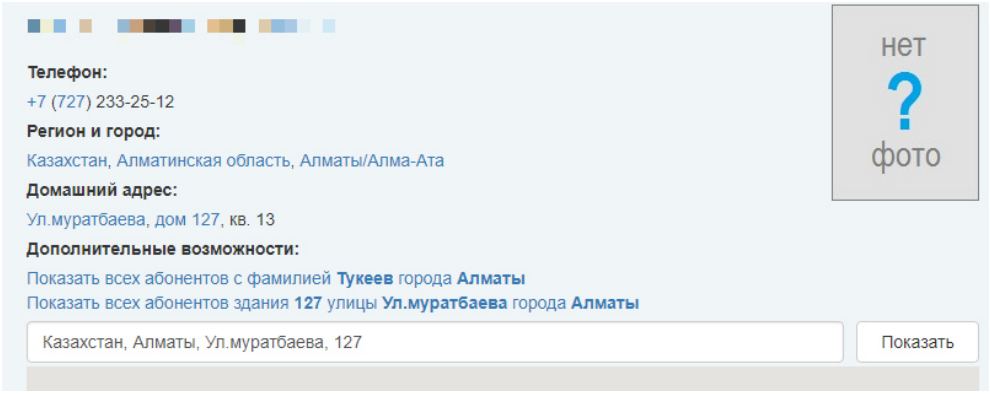


Рисунок 75 – Пример контента, получаемого в результате запроса

Для удобства сбора информации из контента, весь контент разделен на блоки посредством регулярного выражения. Каждый блок содержит информацию определенного типа, которую легко спарсить, рисунок 76. В результате выполнения регулярного выражения будет создан массив bloks, который будет содержать собранную информацию. В конце регулярного выражения автор добавил параметр (re.DOTALL|re.M), который используется при парсинге многострочного контента, например, если необходимо собрать не просто какой-то параметр, а много информации содержащейся в многострочном контенте.

```
bloks = re.findall('<tr>\s+<td><a(.*?)</td>\s+</tr>', str(req.text), re.DOTALL|re.M)

i = 1
```

Рисунок 76 – Деление контента на блоки

Далее бот использует цикл для перебора всех собранных блоков. Используя каждый бок по отдельности посредством регулярных выражений, бот парсит всю информацию, рисунок 77.

```
for blok in bloks:
        number = re.search("peoples/7/{}/[0-9]+/'>([0-9]+)</a></td>".format(city_ number),
blok)
        full_name = re.search("peoples/7/{}/[0-9]+/'>([\sA-Яа-яёЁ_-]+)</a>".format(
city_number), blok)
        street = re.search("/streets/7/{}/[0-9]+/'>(.*?)</a>".format(city_number), blok)
        home = re.search("/streets/7/{}/[0-9]+/[0-9]+/'>(.*?)</a>".format(city_number), blok)
        kv = re.search("/streets/7/{}/[0-9]+/[0-9]+/'>.+</a>,\s[a-яА-Я]+.\s([0-
9]+)".format(city_number), blok)
```

Рисунок 77 – Перебор блоков и сбор информации

Бот собирает следующую информацию: домашний номер телефона, полное имя, улицу на которой проживает человек, дом в котором проживает человек и квартиру, рисунок 78. На рисунке видно, что бот нашел сразу несколько пользователей с одной фамилией и вывел их в отдельные блоки. В программе предусмотрено то, что в одном городе может проживать сразу несколько человек с одинаковой фамилией. Процесс вывода информации о пользователе зациклен, выделив каждого пользователя в отдельный блок. Все это позволило боту собирать информацию о каждом человеке отдельно, не затрагивая другого человека, что предотвращает смешивание информации. В результате бот выдает всю информацию о найденных людях в более чем читабельном виде. Так же исключено, что информация одного человека попадет в блок другого человека.

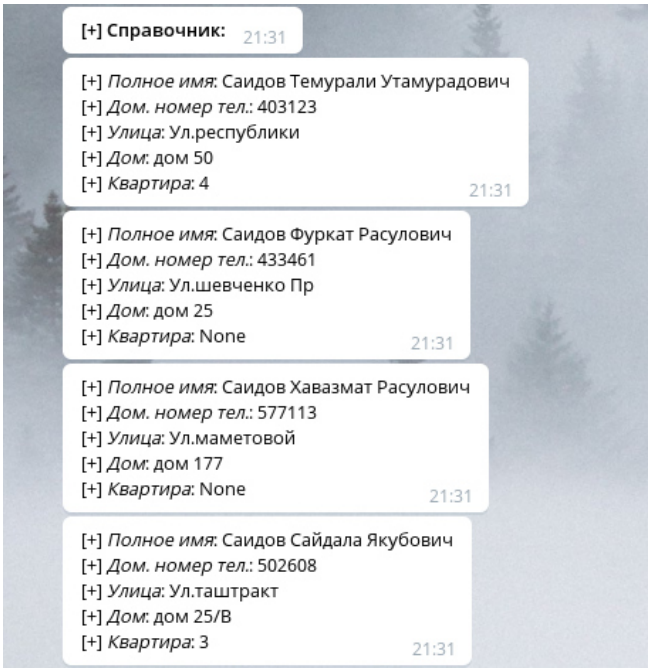


Рисунок 78 – Собранная информация

На рисунке 79 показан сбор информации в переменные с обработкой ошибок.

```
try:
        number = number.group(1)
except Exception as error:
        pass
        number = "None"
try:
        full_name = full_name.group(1)
except Exception as error:
        pass
        full_name = "None"
try:
        street = street.group(1)
except Exception as error:
        pass
        street = "None"
try:
        home = home.group(1)
except Exception as error:
        home = "None"
try:
        kv = kv.group(1)
except Exception as error:
        pass
        kv = "None"
```

Рисунок 79 – Обработка ошибок в случае их возникновения

Как только бот обработал первый блок, бот собирает всю полученную информацию в словарь, используя в качестве ключа параметры people1, people2, people3 и так далее, в зависимости от количества найденных людей, рисунок 80. После того как все данные собраны, бот возвращает словарь главному модулю для вывода информации пользователю.

```
full_data = str(number) + ";" + str(full_name) + ";" + str(street) + ";" + str(home) + ";" + str(kv)

data_from_spravka['people{}'.format(i)] = full_data

i += 1
```

Рисунок 80 – Формирование словаря с данными

**Вывод**

В данной главе подробно рассмотрены предпринятые меры безопасности, система логирования, система обработок ошибок, система приватного доступа и структура каждого модуля бота.

Как показывает практика, система безопасности — это наиболее важный аспект, к разработке которого нужен тщательный подход. В программе предусмотрены все возможные вектора атаки на бот. Разработана система фильтрации сообщений от пользователя, предотвращена возможность атаки типа удаленное выполнение кода и так далее.

Для отслеживания запросов пользователя разработана система логирования, которая позволяет в реальном времени следить за запросами пользователей и в нужный момент зафиксировать и предотвратить попытки атаки на бота.

Для предотвращения прекращения работы бота использованы специальные структуры для обработки ошибок, которые позволяют предотвратить остановку бота, даже если произошла непредвиденная ошибка. Это позволило реализовать хороший уровень непрерывности бота, что является довольно важным параметром.

Разработана система приватного доступа, которая предоставляет доступ только тем пользователям, которые имеют на это право. Доступ предоставляется администратором, посредством добавления уникального идентификатора пользователя в доверенные. Это надежный способ, позволяющий легко регулировать доступность бота.

Разработан приветливый и интуитивно понятный интерфейс, который позволяет пользователю легко управлять ботом и получать результаты в удобно читаемом виде.

Подробно рассмотрена структура каждого модуля и подробно описано функционирование бота на всех этапах его работы.

Стоит подчеркнуть что, для использования данного бота по отношению к искомому человеку необходимо для начала получить разрешение искомого человека.

## 4 Технико-экономическое обоснование

Цель данного дипломного проекта заключается в разработке программного обеспечения, позволяющего в максимально короткие сроки раскрыть личность человека, владея при этом минимальным набором идентификаторов.

На данный момент существует множество различных программ позволяющих собрать информацию о человеке. Разработанная программа работает только с открытыми источниками. Для того, чтобы воспользоваться разработанным программным обеспечением, владелец должен удостовериться, что искомый человек разрешает автоматизированный сбор информации. Только после того как искомый человек даст согласие, владелец программного обеспечения сможет воспользоваться программой, не нарушая личное пространство искомого человека. Под открытыми источниками подразумеваются социальные сети и иные ресурсы, где пользователи добровольно оставляют о себе информацию.

Данное программное обеспечение может использоваться компаниями, которые регулярно взаимодействуют с различными людьми и им необходимо быть уверенными в личности человека.

В разработке программного обеспечения будет участвовать группа специалистов, в состав которой входит технический руководитель, программист-разработчик. В обязанности технического руководителя входит соблюдение и разработка рабочих графиков, их контроль и оптимизация. В обязанности программиста-разработчика входит разработка технического обоснования, разработка программного обеспечения, его тестирование и сопровождение. Следовательно, основная работа ложится на плечи программиста-разработчика, в то время как технический руководитель занимается организационными вопросами.

Технико-экономическое обоснование содержит следующие пункты:

- определение сложности разработки программного обеспечения;
- расчет затрат на разработку ПО;
- определение ценности готового продукта;
- оценка результатов работы программного обеспечения.

### 4.1 Определение сложности разработки ПО

Для того, чтобы точно определить сложность разработки программного обеспечения, необходимо произвести деление всей задачи на более простые этапы. Это позволит эффективно следить за прогрессом разработки программного обеспечения, за счет деления сложной задачи на более легкие подзадачи. Такой подход, с точки зрения автора, считается более эффективным и позволяет результативно и быстро обрабатывать подзадачи. Модель распределения сложности разработки ПО и стадии разработки представлены в таблице 4.1.

Таблица 4.1 – Этапы разработки ПО

| Этапы разработки ПО | Вид работы | Трудоемкость, чел. час. |
|---|---|---|
| Этап 1 | Постановка задач | 10 |
| Этап 2 | Разработка и утверждение ТЗ на разработку ПО | 20 |
| Этап 3 | Поиск и изучение подобных программ | 15 |
| Этап 4 | Поиск и изучение сопутствующей литературы | 10 |
| Этап 5 | Составление аналитических графиков ПО | 5 |
| Этап 6 | Оформление теоретической части дипломной работы | 15 |
| Этап 7 | Разработка практической части дипломного проекта | 25 |
| Этап 8 | Реализация проекта | 50 |
| Этап 9 | Отладка и устранение недоработок | 20 |
| Этап 10 | Организация отчета и результатов работы | 10 |
| Этап 11 | Тестирование ПО | 20 |
| Этап 12 | Подведение итогов по разработанному ПО | 10 |
| Этап 13 | Внедрение | 15 |
| Итого: трудоемкость выполнения дипломного проекта | | 225 |

Продолжительность рабочего дня равна 8 часам. В результате для реализации программного обеспечения необходимо 28 рабочих дней.

### 4.2 Расчет затрат на разработку ПО

Определение затрат необходимых для разработки программного обеспечения производится на основе имеющейся сметы, которая включает следующие элементы:

- материальные затраты;
- затраты на оплату труда;
- социальный налог;
- амортизация основных фондов;
- прочие затраты.

Материальные затраты делятся на основные и вспомогательные затраты на материалы, энергию и другие затраты необходимые для разработки ПО. Расчет материальных затрат происходит по форме, предоставленной в таблице 4.2.

Таблица 4.2 – Затраты на материальные ресурсы

| Наименование материала | Марка | Ед. измерения | Количество | Цена за ед. в тенге | Сумма в тенге |
|---|---|---|---|---|---|
| Бумага для офиса | International Paper | Упаковка | 3 | 1 000 | 3 000 |
| Тетрадь (96 листов) | Маяк Канц | Штук | 2 | 190 | 380 |
| Блокнот | КТС-ПРО | Штук | 2 | 400 | 800 |
| Ручки | Parker Jotter | Штук | 2 | 90 | 180 |
| Компьютерная мышь | TECH | Штук | 1 | 3 000 | 3 000 |
| Итого: | | | | | 7 360 |

Для разработки программного обеспечения будет использоваться ноутбук Lenovo Ideapad 330 81FK00CXRK Gray, мощности ноутбука достаточно для выполнению поставленных задач. Так как ноутбук содержит установленную операционную систему Windows 10 x64 и программное обеспечение необходимое для разработки ПО, нет нужды производить дополнительные затраты на новую ОС и ПО. Альтернативой операционной системе Windows 10 x64, может служить бесплатная операционная система Linux семейства Unix. Данная операционная система распространяется по лицензией GNU General Public License, распространяясь под данной лицензией, операционная система имеет открытый код и возможность произвести любую конфигурацию, вплоть до конфигурации самого ядра операционной системы. Данная операционная система полностью подходит для разработки программного обеспечения, так как обладает большим функционалом и возможностью конфигурации самой системы под нужны разработчика.

Общую сумму, необходимую на материальные средства ($З_м$) можно рассчитать по следующей формуле:

$$З_м = \sum P_i * Ц_i, \quad (4.1)$$

где $P_i$ - расход i-го вида материального ресурса, натуральные единицы;

$Ц_i$ - цена за единицу i-го вида материального ресурса, тг;

i - вид материального ресурса;

n - количество видов материальных ресурсов.

Расчет затрат на необходимое оборудование и программное обеспечение производится по форме, приведенной в таблице 4.3.

Таблица 4.3 – Расчет затрат на оборудование и ПО, необходимое для проекта

| Наименование материала | Марка | Ед. измерения | Количество | Цена за ед. в тенге | Сумма в тенге |
|---|---|---|---|---|---|
| Ноутбук | Lenovo Ideapad 330 81FK00CXRK | Штук | 1 | 390 000 | 390 000 |
| Принтер | Samsung SCX-3400 | Штук | 1 | 52 874 | 52 874 |
| Хостинг | PS.kz | Штук | 2 | 2 350 | 4 700 |
| Модем | Ericsson T073G | Штук | 1 | 14 000 | 14 000 |
| ОС | Windows 10 | Штук | 1 | - | - |
| Домен | PS.kz | Штук | 1 | 3 338 | 3 338 |
| Итого: | | | | | 464 912 |

$$Зм = 7\ 360 + 464\ 912 = 472\ 272\ (тг)$$

Для реализации программного обеспечения необходимы материалы на сумму 472 272 тенге.

### 4.3 Расчет затрат на электроэнергию

Так как при разработке программного обеспечения не обойтись без потребления электроэнергии, имеет смысл произвести расчет затрат на электроэнергию.

Согласно таблице 4.1, для разработки программного обеспечения необходимо порядка 225 часов, теперь необходимо рассчитать стоимость электроэнергии, которая будет потрачена в течении 225 часов [10].

$$Э = З_{эл.эн.обор.} + З_{доп.нужды.}, \quad (4.2)$$

где $З_{эл.эн.обор.}$ – затраты на электроэнергию оборудования;

$З_{доп.нужды.}$ – затраты электроэнергии на дополнительные нужды.

Расчет электроэнергии, которая необходима для оборудования, определяется по следующей формуле:

$$З_{эл.эн.обор.} = \sum W * К_{исц} * S * T, \quad (4.3)$$

где W – потребляемая мощность, Вт;

$К_{исц}$ – коэффициент использования ($К_{исц}$ = 0,7..0,9);

Т – время работы;

S – тариф (1кВт/ч = 23,85 тг).

Итоги по расчетам стоимости затрачиваемой электроэнергии представлены в таблице 4.4.

Таблица 4.4 – Затраты на электроэнергию

| Наименование приборов | Паспортная мощность, кВт | Коэффициент мощности | Время работы оборудования, ч | Цена ЭЭ тг/кВтч | Сумма, тг. |
|---|---|---|---|---|---|
| Ноутбук | 0,6 | 0,7 | 225 | 23,85 | 2 254,9 |
| Модем | 0,08 | 0,9 | 225 | 23,85 | 386,4 |
| Принтер | 0,5 | 0,9 | 24 | 23,85 | 257,6 |
| Кондиционер | 0,8 | 0,9 | 180 | 23,85 | 3 091 |
| Освещение | 0,3 | 0,7 | 225 | 23,85 | 1 127 |
| Итого: | | | | | 7 117 |

$$З_{эл.эн.обор.} = 7\ 117\ (\text{тенге})$$

На дополнительные потребности расходы подсчитываются на основе повышенного показателя в объеме 5% от расходов на электроэнергию:

$$З_{доп.нужды} = 5\% * З_{эл.эн.обор.} \quad (4.4)$$

Определим затраты на дополнительные потребности согласно формуле (4.4):

$$З_{доп.нужды} = 0.05 * 7117 = 355{,}85\ (\text{тенге})$$

Исходя из всех расчетов, полные расходы на электроэнергию составляют:

$$Э = 355{,}85 + 7117 = 7\ 472{,}85\ (\text{тенге})$$

Для принтера расчет проводился для периода в 24 часа, так как нет необходимости постоянно использовать принтер.

### 4.4 Расчет затрат на оплату труда

Для разработки программного обеспечения, необходимо два работника:

- руководитель проекта – управление рабочим временем, корректировка рабочих процессов, координация, изучение предметной области;
- разработчик – разработка ПО, тестирование и сопровождение.

Сумму расходов на оплату труда рассчитывает по следующей формуле:

$$З_{тр} = \sum ЧС_i * T_i \quad (4.5)$$

где $ЧС_i$ - часовая ставка i-го работника, тг;

$T_i$ - трудоемкость разработки модели, чел.×ч; i - категория работника.

Во время реализации проекта рабочее время участников не равномерно, поэтому имеет смысл установить часовую ставку каждого работника и общий объем заработной платы.

Часовую ставку сотрудника можно рассчитать по следующей формуле:

$$ЧС_i = \frac{ЗП_i}{ФРВ_i} \quad (4.6)$$

где $ЗП_i$ - месячная заработная плата i-го работника, тг;

$ФРВ_i$ - месячный фонд рабочего времени i-го работника, час.

Месячная заработная плата руководителя равняется 180 000 тенге и месячная заработная плата разработчика равняется 150 000 тенге. Рассчитаем часовую ставку каждого работника согласно формуле (4.6):

$$ЧС_{руководитель} = \frac{180\,000}{22 * 8} = 1\,022{,}72 \text{ тг/ч}$$

$$ЧС_{разработчик} = \frac{150\,000}{22 * 8} = 852{,}3 \text{ тг/ч}$$

Часовая ставка руководителя составляет 1 022,72 (тг/ч), трудоемкость разработки равняется 100 часам. Часовая ставка разработчика составляет 852,3 (тг/ч), трудоемкость разработки равняется 225 часам. Согласно формуле (4.5) можно рассчитать сумму расходов на заработную плату работников:

$$З_{тр} = 1022{,}72 * 100 + 852{,}3 * 225 = 102\,000{,}28 + 191\,767{,}5 = 293\,767{,}8$$

Таблица 4.5. – Расчет заработной платы

| Категория работника | Квалификация | Трудоемкость разработки ПП, час. | Часовая ставка, тг/ч | Сумма, тг. |
|---|---|---|---|---|
| Руководитель | Проектный руководитель | 100 | 1022,72 | 102 000,28 |
| Разработчик | Программист | 225 | 852,3 | 191 767,5 |
| Итого: | | | | 293 767,8 |

### 4.5 Расчет затрат по социальному налогу

Социальный налог можно рассчитать по следующей формуле:

$$С_н = (ФОТ - ПО) * 0{,}095 \quad (4.7)$$

где ПО - отчисления в пенсионный фонд, они составляют 10% от ФОТ.

$$ПО = \ 293\ 767{,}8 * 0{,}1 = 29\ 376{,}8 \text{ тенге}$$

$$С_н = (293\ 767{,}8 - 29\ 376{,}8) * 0{,}095 = 25\ 117{,}2 \text{ тенге}$$

Результаты расчетов представлены в таблице (4,6):

Таблица 4.6 – Начисление социального налога

| Категория работника | Количество человек | Заработная плата, тг | Пенсионные отчисления, тг | Социальный налог, тг |
|---|---|---|---|---|
| Руководитель | 1 | 102 000,28 | 10 273 | 8 714,1 |
| Разработчик | 1 | 191 767,5 | 19 177 | 16 396, 1 |
| Итого: | | | | 25 117,2 |

Согласно Налоговому кодексу Республики Казахстан социальный налог составляет 9,5% от фонта оплаты труда.

### 4.6 Амортизация основных фондов и прочие затраты

Нормы амортизации ОФ необходимо определить в соответствии с налоговым кодексом РК. Амортизацию ОФ можно определить по следующей формуле:

$$A_г = \frac{С_{об} * Н_а}{100} \quad (4.8)$$

где $С_{об}$ – стоимость оборудования;

$Н_а$ – норма амортизации (норма амортизация = 25);

Формула (4.8) позволяет рассчитать нужную сумму для амортизационных отчислений за год для ноутбука:

$$A_г = \frac{390\ 000 * 25}{100} = 97\ 500 \text{ тенге}$$

Теперь необходимо рассчитать норму амортизации за период разработки:

$$A_{\text{г}} = \frac{97\,500 * 34}{365} = 9\,082{,}2 \text{ тенге}$$

Подобным образом необходимо рассчитать норму амортизации для всего оборудования. Результаты расчетов приведены в таблице (4.7).

Таблица 4.7 – Амортизация ОФ

| Наименование оборудования и ПО | Стоимость оборудования и ПО, тг | Годовая норма амортизации, % | Сумма амортизации за год, тг | Сумма амортизации за время разработки, тг |
|---|---|---|---|---|
| Ноутбук | 390 000 | 25 | 97 500 | 9 082,2 |
| Принтер | 52 874 | 25 | 13 218 | 108,64 |
| Модем | 14 000 | 20 | 2 800 | 172,6 |
| Хостинг | 4 700 | 20 | 940 | 105,98 |
| Домен | 3 338 | 15 | 500,7 | 52,28 |
| Итого: | | | 114 958,7 | 9 521,67 |

Смета расходов на разработку ПО.

На основе всех представленных расчетов необходимо оформить смету расходов на разработку ПО согласно форме, которая приведена в таблице 4.8. На рисунке 81 продемонстрирована диаграмма рабочих расходов.

Таблица 4.8 – Смета затрат на разработку ПО

| Статьи затрат | Сумма, тг |
|---|---|
| Затраты на оборудование | 464 912 |
| Затраты на программное обеспечение | 0 |
| Затраты на оплату труда | 293 767,8 |
| Социальные налоги | 25 117,2 |
| Затраты на электроэнергию | 7 117 |
| Амортизация основных фондов | 9 521,67 |
| Прочие расходы | 27 600 |
| Итого по смете: | 828 035,76 |

Затраты на программное обеспечение равняются нулю так как, в качестве операционной системы на ноутбуке установлен Linux семейства Unix, дистрибутив Debian. Данная операционная система распространяется под лицензией GNU General Public License. Согласно лицензии, пользователь может конфигурировать операционную систему, под свои нужды. Так же данная операционная система часто используется разработчика, при написании программного обеспечения. Данная операционная система полностью подходит для разработки программного обеспечения, так как

обладает большим функционалом и возможностью конфигурации самой системы под нужны разработчика [11].

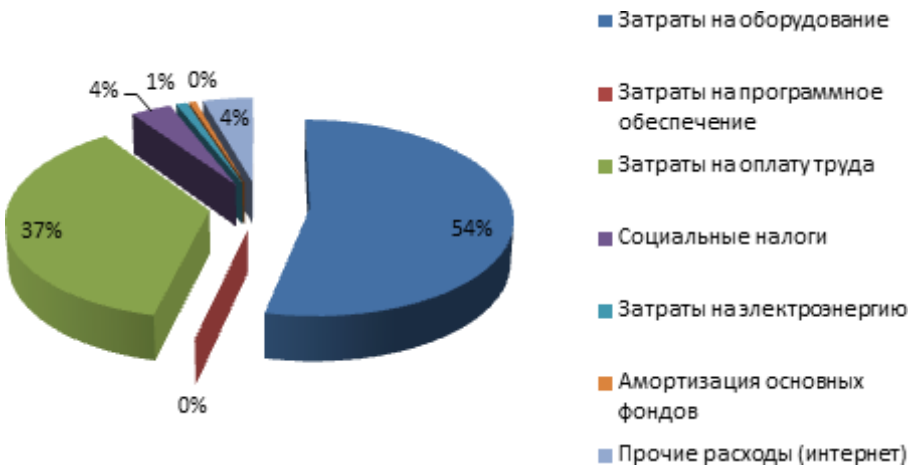


Рисунок 81 – Диаграмма затрат

Основными расходами исходя из графика выше, являются затраты на оборудование необходимое для разработки программного продукта.

### 4.7 Определение возможной (договорной) цены ПО

Стоимость программного обеспечения определяется на основе качества разработанного продукта, сроков его разработки и производительности продукта. Стоимость $Ц_Д$ для программного обеспечения можно рассчитать по следующей формуле:

$$Ц_Д = З_{нир}\left(1 + \frac{P}{100}\right) \quad (4.9)$$

где $З_{нир}$ – затраты на разработку программного обеспечения, тг;

Р – средний уровень рентабельности ПО, (%). Данный параметр принят равным 25%.

$$Ц_Д = 828\,035{,}76\left(1 + \frac{25}{100}\right) = 1\,031\,294{,}7 \text{ тенге}$$

Далее необходимо определить стоимость реализации с учетом НДС, ставка НДС устанавливается законодательством РК. На 2019 года ставка НДС составляет 12%. Стоимость реализации учитывая НДС можно рассчитать по следующей формуле:

$$Ц_р = Ц_д + Ц_д * НДС \quad (4.10)$$

$$Ц_р = 1\,031\,294{,}7 + 1\,031\,294{,}7 * 0{,}12 = 1\,155\,050{,}06 \text{ тенге}$$

**Вывод**

Данная глава дипломного проекта содержит экономические расчеты, которые позволяют определить затраты необходимые для разработки программного продукта. Расчеты включают в себя:

- расчет трудоемкости разработки программного продукта;
- расчет затрат на разработку программного продукта;
- расчет затрат на электроэнергию;
- расчет затрат на оплату труда;
- расчет затрат по социальному налогу;
- амортизация основных фондов и прочие затраты.

Для покупателя основным показателем будет считаться оптимальная цена программного продукта и его производительность. Цена и полезность программного продукта должна обладать равновесием, чтобы покупатель был заинтересован в приобретении разработки. Качественным результатом для покупателя считается, что приобретенное программное обеспечение полностью покрывает все необходимые задачи, которые встают перед покупателем. Так же в последней главе был произведен расчет договорной цены программного продукта, который равняется 1 155 050 тенге, данное значение является рациональным с точки зрения экономической эффективности.

## 5 Безопасность жизнедеятельности

Основной целью данного дипломного проекта является разработка автоматизированного программного обеспечения, которое позволяет в максимально короткие сроки собрать информацию о человеке, опираясь на минимальный набор знаний.

Актуальность данной темы заключается в том, что на текущий момент информация является ценным ресурсом, с которым необходимо правильно взаимодействовать. Жизненный цикл информации собранной посредством программного обеспечения будет включать следующие этапы: сбор информации, эксплуатация полученной информации, расположение полученной информации в специальное хранилище и ее долговременное хранение.

Данное программное обеспечение может использоваться различными компаниями, которым приходится часто работать с новыми людьми. В результате взаимодействия с человек, компания должна быть уверена в нем и владеть необходимым перечнем информации о человеке, с которым взаимодействует. Для работы программы ей необходимо предоставить фамилию, имя и отчество субъекта.

Разработанная программа работает только с открытыми источниками. Для того чтобы воспользоваться разработанным программным обеспечением, владелец должен удостовериться, что искомый человек разрешает автоматизированный сбор информации. Только после того как искомый человек даст согласие, владелец программного обеспечения сможет воспользоваться программой, не нарушая личное пространство искомого человека. В качестве открытых источников используются социальные сети: Yandex, Вконтакте и так далее.

### 5.1 Анализ условий труда

При разработке программного обеспечения, разработчик вынужден долгое время взаимодействовать с компьютерной техникой. Рабочее место – это место временного или постоянного пребывания сотрудника. В связи с тем, что работник должен долгое время проводить в сидячем положении у компьютерной техники, должны быть обеспечены необходимые меры, которые позволять работнику максимально комфортно проводить это время без вредного воздействия на его организм. Данные меры включают в себя: компьютерное оборудование и мебель должны быть размещены оптимальным образом, достаточное рабочее пространство, которые позволит работнику проделывать все необходимые действия и перемещения, работник должен получать необходимое количество световых лучей, чтобы максимально снизить нагрузку на зрение, на рабочем месте должна соблюдаться комфортная комнатная температура и так далее.

Окраска помещения и мебели так же должна быть благоприятной для зрительного восприятия, отражающие поверхности могут привести к

попаданию отраженных световых лучей в область глаз работника, что будет вызывать раздражение и снижение работоспособности. Для предотвращения избыточной яркости и отражающего эффекта, необходимо использовать матовые поверхности спокойных цветов, так же необходимо использовать шторы и экраны. В помещениях с ПЭВМ должны соблюдаться следующие величины отражения: для потолка 60 – 70%, стен 40 – 50%, пола примерно 30%, для иной поверхности 30 – 40%.

Существует несколько типов освещения, одни из них: естественное и искусственное.

Естественное освещение – это освещение, основанное на дневном свете, который проникает через световые проемы. Данный тип освещения характеризуется различными факторами, такими как: время года, погодные условия, текущее время суток и другие факторы.

Искусственное освещение – это освещение, которое применяться в темное время суток и моменты когда не достаточно естественного освещения. Комбинирование естественного и искусственного освещения называется комбинированным освещением.

Правильный выбор светильников и корректное расположение рабочих мест по отношению к естественному и искусственному освещению, обеспечит ограниченную отраженную блёскость на рабочих поверхностях, при этом яркость бликов на экране рабочей станции не должна превышать 40 $кд/м^2$.

Для искусственного освящения должны применяться люминесцентные лампы белого света. В производственной среде и административно-общественных помещениях можно использовать металлогалогенные лампы мощностью до 250 Вт. В общем освещением должно быть расположено прерывистыми линиями светильников или же в виде сплошной линии, которые должно располагаться сбоку от мест рассадки работников, параллельно линии зрения.

Условия по организации деятельность за компьютером:

- наличие в рабочем помещении естественного и искусственного освещения;
- оборудование помещения системами кондиционирования воздуха или эффективной вентиляцией; помещение должно проветриваться ежечасно;
- ежедневная влажная уборка помещения;
- использование штор или жалюзи для избегания прямого попадания солнечных лучей;
- равномерное искусственное освещение.

Для соблюдения всех норм труда необходимо произвести вспомогательные расчёты, которые приведены далее.

### 5.2 Характеристики рабочего помещения

Рабочее помещение оборудовано на одно рабочее место. Рабочее помещение расположено в высокоэтажном здании на 10 этаже. Модель оборудованного помещения, в котором ведется разработка программного

обеспечения, приставлена на рисунке 82. Рабочее помещение имеет следующие характеристики: длина помещения L = 4 метра, ширина помещения B = 3,5 метров, высота помещения H = 3 метра. Само помещение было оборудовано и построение на основании санитарных требований от 01.12.2011 года, согласно требования, площадь рабочего места для одного пользователя должно составлять 4,5 метра в квадрате.

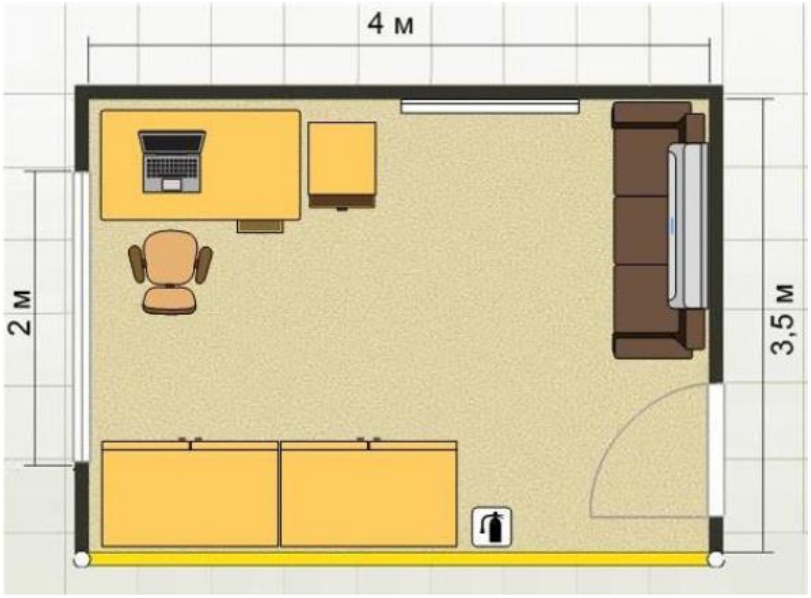


Рисунок 82 – Рабочее помещение

Общее рабочее пространство удовлетворяет требованиям и составляет 14 квадратных метров. Согласно требования, монитор должен быть отдален от глаз пользователя на расстояние 60 сантиметров.

### 5.3 Используемое оборудование и его характеристики

Ноутбук Acer Aspire 3. Технические характеристики устройства:

- Intel(R) Core i3 (CPU 2.3 GHz);
- UMA Intel HD;
- ОЗУ 4 ГБ;
- HDD 500 ГБ;
- электропитание: 220-250В, 50Гц, 400 Вт;
- габариты(мм): 385 - 258 - 32.

Модем: 4-х портовый с коммутатором 10/100 Мбит/с.
Стул: высота 0,43 м.
Стол: высота – 0,7 м, длина – 2 м, ширина – 3,5 м.

### 5.4 Расчет естественного освещения

Одним из основных показателей, который должен держаться на определенном уровне, является освящение, освящение играет важную роль для создания комфортных условий в рабочей обстановке. Среда для работы должна иметь комфортно освящение, что бы рабочий не испытывал

напряжения и дискомфорта, в последствии которые могут вызвать симптомы (головная или глазная боль) приостанавливающие работу. Согласно нормам освещенности (СНиП 11-4-79) и отраслевым нормам, работа инженера относится к четвертому разряду зрительной работы. Расчет по освещению необходимы для того, чтобы определить площадь для световых проемов при естественном освещении, а так же характеристики искусственного освещения. Ниже представлена формула для расчета площади световых проемов при естественном освещении:

$$S = \frac{(S_n * e_n * K_n * h_0 * K_{зо})}{(100 * t_0 * r_1)} \quad (5.1)$$

где $S_n$ – площадь помещения, м$^2$;

$e_n$ – нормированное значение КЕО, %;

$K_n$ – коэффициент запаса;

$h_0$ – световая характеристика окон (6,5 - 29);

$K_{зо}$ – коэффициент затемнение окон противостоящими зданиями (1,0-1,7);

$r_1$ – коэффициент повышение КЕО за счет отраженного света от поверхности помещения (1,05 - 1,7);

$t_0$ - общий коэффициент светопропускания равен от 0,1-0,8.

Согласно тому что, длина помещения равна 4 метра, а ширина равно 3,5 метра, можно найти площадь пола по следующей формуле:

$$S_n = L * B \quad (5.2)$$

$$S_n = 4 * 3{,}5 = 14 \text{ м}^2$$

Для текущего рабочего помещения площадь световых проемов естественного бокового освещения можно определить по формуле (5.1), для этого понадобятся следующие значения коэффициентов:

$e_n = 1{,}5$ %;

$K_n = 1{,}5$;

$h_0 = 29$;

$K_{зо} = 1$;

$r_1 = 1{,}05$;

$t_0 = 0{,}7$.

Теперь необходимо подставить значение данных коэффициентов в формулу (5.1) и вычислить площадь световых проемов:

$$S = \frac{14 * 1{,}5 * 1{,}5 * 29 * 1}{100 * 0{,}7 * 1{,}05} = 12{,}43 \text{ м}^2$$

Согласно расчетом площадь оконного пространства должна составлять $12{,}43\ м^2$, учитывая что, в помещение окно площадью $4\ м^2$, необходимо дополнительное искусственное освещение.

### 5.5 Расчет искусственного освещения

Основываясь на норму СНиП 11-4-79 для четвертого класса зрительных работ освещенность помещения должно быть не менее 200 Лк. Определить номинальную освещенность рабочего места можно по формуле:

$$E = \frac{Ф_{св} * n * N * 1}{s * К_з * Z} \quad (5.3)$$

где $Ф_{св}$ – световой поток от ламп, Лк;
N – количество светильников;
$К_з$ – коэффициент учитывающий запыленность светильников;
n – коэффициент использования светильников;
s – площадь помещения, $м^2$;
z – коэффициент неравномерности освещения.

Основываясь на норму СНиП 11-4-79 для типа ламп, который будут использоваться в ЛАЗ КДП (светильник типа УСП-35): $К_з = 1{,}4{:}\,1{,}5$ при нормальной эксплуатации светильников; z = 1,1:1,2 при оптимальном их размещении. В данном случае коэффициент n полностью зависит от светильников и их типа, коэффициенты отражения светового потока от потолка - $p_2$, от пола $p_3$, от стен $p_1$, зависят от размера помещения, учитывающих величиной I, где I это индекс помещения.

$$I = \frac{(A * B)}{h_c * (A + B)} \quad (5.4)$$

где A, B - параметры помещения, м;
$h_c$ - высота светильников над рабочей поверхностью.

Высота светильников над рабочей поверхность определяется по формуле:

$$h_c = Н_{помещения} - Н_{свеса} - Н_{р.п.} \quad (5.5)$$

где $Н_{свеса} = 0{,}3$ - высота свеса ламп, м;
$Н_{р.п.} = 0{,}7$ - расстояние рабочей поверхности над полом, м;
$Н_{помещения} = 3$ - высота помещения, м.

Согласно формуле (5.5) определим высоту светильников над рабочей поверхностью:

$$h_{расч} = 3 - 0{,}3 - 0{,}7 = \ 2 \text{ м}$$

Учитывая, что помещение имеет размеры 3 м × 4,5 м и высота светильников над рабочей поверхностью $h_c = 2$ м, по формуле (5.4):

$$I = \frac{(3 * 4{,}5)}{2 * (3 + 4{,}5)} = 0{,}9$$

По таблице (5.1) можно определить коэффициент использования светового потока n, учитывая что коэффициенты $p_1$ = 30%, $p_2 = 50\%$, $p_3 = 10\%$.

Таблица 5.1 - Значения коэффициента использования светового потока

| Коэффициент I | 0,5 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| Коэффициент использования светового X потока, h | 0,22 | 0,36 | 0,48 | 0,54 | 0,59 |

Коэффициент n = 0,3 для рабочего места. От лампы типа ЛБ-40 световой поток равняется 3120 Лк, в совокупности от 3 ламп световой поток будет равняться 9360 Лк. Основываясь на все вычисления и данные можно определить номинальную освещенность рабочего места по формуле (5.3).

$$E = \frac{9360 \cdot 0{,}3}{14 \cdot 1{,}4 \cdot 1{,}2} = 119{,}39 \text{ (Лк)}$$

Полученное значение не соответствует нормам, необходимо взять лампы с более высоким световым потоком. Для повышения светового потока возьму лампы “Светодиодный светильник 24W "круг" накладной Bellson” со световым потоком равным 6000 Лк.

Следующая формула позволяет определить количество светильников для создания освещенности в 200 лк, используя значение светового потока, равного 6000 лк.

$$N = \frac{S_{помещ} K_З Z E_H}{\Phi_л n \eta} \qquad (5.6)$$

Согласно приведенной выше формуле рассчитывает необходимое количество светильников:

$$N = \frac{12{,}43 * 1{,}5 * 1{,}1 * 200}{6000 * 1 * 0{,}23} = 2{,}97 \approx 3 \text{ светильника}$$

То есть, для создания освещенности в 200 лк с разрядом зрительных работ 3, необходимо 3 светодиодных светильника со световым потоком равным 6000 лк.

Следующая формула позволяет определить высоту свеса светильников:

$$h_{\text{расч}} = H_{\text{пом}} - \left(H_{\text{св}} + H_{\text{р.п.}}\right) \quad (5.7)$$

где $H_{\text{р.п.}} = 0{,}7$ – расстояние рабочей поверхности над полом, м;

$H_{\text{пом}} = 3$ – высота помещения, м.

Тогда высота подвеса светильников равна:

$$h_{\text{расч}} = 3 - (1{,}5 + 0{,}7) = 0{,}9 \text{ м}$$

Следующая формула позволяет определить расстояние между светильниками:

$$L = \lambda * h \quad (5.8)$$

где L – расстояние между соседними светильниками;

h – высота подвеса светильника над рабочей поверхностью;

$\lambda = 1{,}2 \div 2{,}4$.

По формуле (5.8) определяем необходимое расстояние между светильниками:

$$L = 0{,}9 * 1{,}3 = 1{,}17 \text{ м}$$

Расстояние между светильником и стеной можно определить по следующей формуле:

$$l_a = l_b = \frac{L_b}{3} + [0{,}3; 0{,}5] \quad (5.9)$$

По формуле (5.9) определяем расстояние между светильниками и стеной:

$$l_a = \frac{0{,}72}{3} + 0{,}47 = 0{,}71 \text{ м}$$

Расстояние между светильником и стеной сбоку можно определить по следующей формуле:

$$l_b = L_b - 1 \quad (5.10)$$

По формуле (5.10) рассчитываем расстояние между светильником и стеной сбоку:

$$l_b = 2 - 0{,}8 = 1{,}2 \text{ м}$$

Схема размещения светильников и световых проемов представлена на рисунке 83.

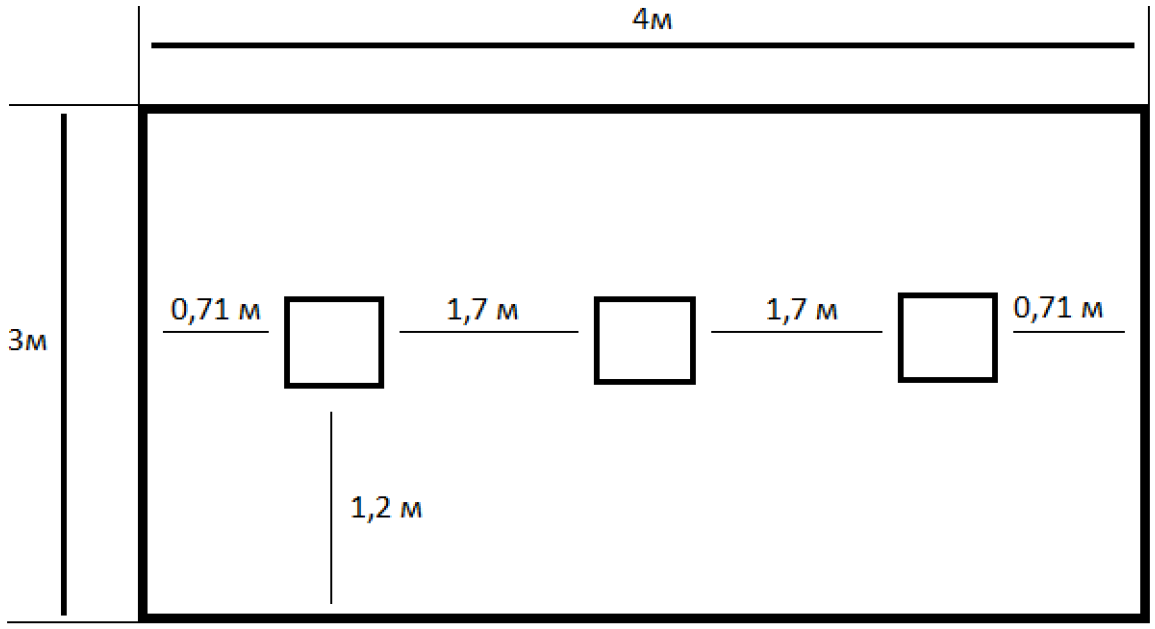


Рисунок 83 – Расположение светильников помещении

Так-как значение полученное ранее не соответствовало нормам, ниже приведен повторный расчет номинального освещения рабочего места. В данном случае используются лампы “Светодиодный светильник 24W "круг" накладной Bellson” со световым потоком равным 6000 Лк.

$$E = \frac{18000 \cdot 0{,}3}{14 \cdot 1{,}4 \cdot 1{,}2} = 229{,}6 (\text{Лк})$$

Полученное значение соответствует нормальным условиям и оказывает благоприятную рабочую обстановку. Согласно полученным значениям разработчик и руководитель не будут испытывать дискомфорт находясь в помещении [12].

### 5.6 Расчет воздухообмена в рабочем помещении

Не менее важной характеристикой на рабочем месте является температура, для регулирования температурой используется такой оборудование как вентиляторы, кондиционеры и естественная вентиляция посредством окон и дверей. Воздухообмен на рабочем месте определяется по формуле:

$$Q = \frac{g}{X - X_n} \tag{5.11}$$

где Q - потребный воздухообмен, м$^3$/ч;

g - количество вредных веществ, л/ч;

X – предельно допустимая концентрация вредных веществ в помещения, л/м$^3$;

$X_n$ – предельно допустимая концентрация вредных веществ в наружном воздухе, л/м$^3$.

Кратность воздухообмена n, показывает количества раз в течении часа воздух обязан смениться.

$$n = \frac{Q}{Q_{пот}} \tag{5.12}$$

где $Q_{пот}$ - это объем помещения.

Объем помещения $Q_{пот}$ можно найти по формуле:

$$Q_{пот} = L * H * B \tag{5.13}$$

$$Q_{пот} = 4 * 3 * 3{,}5 = 42 м^3$$

Количество углекислого газа, который выделяет человек при труде в расслабленном состоянии равен 23 л/ч, в помещение предельная концентрация не должна превышать 1 л/м$^3$, предельно допустимая концентрация в наружном воздухе 0,5 л/м$^3$. По формуле (5.6) определим потребный воздухообмен:

$$Q = \frac{23}{1 - 0{,}5} = 46 \left(\frac{м^3}{ч}\right)$$

По формуле (5.7) определяем кратность воздухообмена n:

$$n = \frac{46}{42} = 1{,}1$$

Рассчитаем воздухообмен для удаления избыточного тепла по формуле:

$$Q_{изб} = \frac{L_{изб}}{G_B * C_B * d_t} \quad (5.14)$$

где $L_{изб}$ - избыточное тепло, ккал/ч;

$G_B$ = 1,206 кг/м$^3$ - удельная масса приточного воздуха;

$C_B$ = 0,24 ккал/кг·°С - теплоемкость воздуха;

$d_t$ - разность температур удаленного и приточного воздуха;

$d_t$ зависит от теплонапряжённости воздуха - $L_H$. Если $L_H$ больше 20 ккал/ч, то $d_t$ = 8°С. Если $L_H$ меньше 20 ккал/ч, то $d_t$ = 6°С.

Теплонапряжённости воздуха можно определить по формуле:

$$L_H = \frac{L_{изб}}{Q_{пот}} \quad (5.15)$$

Количество избыточного тепла можно определить по формуле:

$$L_{изб} = L_{об} + L_{ос} + L_л + L_р + L_{отд} \quad (5.16)$$

где $L_{об}$ - тепло от оборудования, ккал/ч;

$L_{ос}$ - тепло от системы освещения, ккал/ч;

$L_л$ - тепло, выделяемое людьми, ккал/ч;

$L_р$ - тепло от солнечной радиации, ккал/ч;

$L_{отд}$ – теплоотдача естественным путем, ккал/ч.

Тепло от оборудования можно найти по следующей формуле:

$$L_{об} = 860 * P_{об} * f \quad (5.17)$$

где $P_{об}$ - номинальная мощность оборудования, Вт;

f - коэффициент передачи, f = 0,25.

Тепло от системы освещения можно рассчитать по формуле:

$$L_{ос} = 860 * P_{ос} * a * b * cof(f) \quad (5.18)$$

где $P_{ос}$ - номинальная мощность освещения, кВт;

a – коэффициент перевода электрической энергии в световую, 0,46;

b – коэффициент одновременной работы ламп, b = 1;
cos(f) – коэффициент мощности, cos(f) = 0,3.

Тепло выделяемое людьми можно рассчитать по формуле:

$$L_л = n * g \quad (5.19)$$

где n - количество человек;
g - тепловыделение одного человека, g=50 (ккал/ч).

Тепло от солнца для одного окна можно рассчитать по формуле:

$$L_р = F * D_{ос} \quad (5.20)$$

где F - площадь окна, м$^2$;
$D_{ос}$ - солнечная радиация, $D_{ос}$ = 65 (ккал/ч).

Для помещения тепло изучающее системой, оборудованием, людьми и солнцем рассчитываем по формулам (0.12 – 0.15):

$$L_{об} = 860 \cdot 1 \cdot 0{,}25 = 215 \text{ (ккал/ч)}$$

$$L_{ос} = 860 \cdot (0{,}04 \cdot 3) \cdot 0.46 \cdot 1 \cdot 0.3 = 14{,}24 \text{ (ккал/ч)},$$

$$L_л = 1 \cdot 50 = 50 \text{ (ккал/ч)},$$

$$L_р = 4 \cdot 65 = 260 \text{ (ккал/ч)}.$$

Естественную теплоотдачу можно приравнять к $L_р$ в холодные времена года и считать равной нулю в теплое время года, по формуле (5.11) можно узнать количество избыточного тепла:

$$L_{изб} = 215+14{,}24+50+260+0 = 539{,}24 \text{ (ккал/ч)}.$$

Воздушную теплонапряженность можно рассчитать по формуле (5.21):

$$L_Н = \frac{539{,}24}{40} = 13{,}5 \text{ (ккал)}$$

Так как теплонапряженность воздуха меньше 20, $d_t = 6°C$. Используем формулу (5.9) чтобы вычислить значение потребного воздухообмена для удаления избыточного тепла:

$$Q_{изб} = \frac{539{,}24}{(1{,}206 \cdot 0{,}24 \cdot 6)} = 310{,}5\ (м3/ч)$$

Следуя из полученных значений, для удаления лишнего тепла и очистки воздуха необходимо использовать вентиляционную систему, которая способна обеспечить требуемую подачу воздуха $Q_{изб} = 310{,}5\ (м3/ч)$. В данном случае подойдет кондиционер Ditreex 24 F12 (R410). Данный кондиционер способен обеспечить подачу воздуха до 900 м$^3$ /ч, рисунок 84 [13].


Рисунок 84 – Кондиционер Ditreex

Таблица 5.2 – Характеристики кондиционера

| Параметр | Значение |
|---|---|
| Производит. по холоду | 7100 (Вт) |
| Потреб. мощность при охлаждении | 2510 (Вт) |
| Рекоменд. площадь охлаждения/обогрева | 60 (м$^2$) |
| Количество конденсата | 2 (l/h) |
| EER/C.O.P. при охлаждении | 2,81/3,21 ( Вт/Вт) |
| Производит. по теплу | 7300 (Вт) |
| Потреб. мощность при обогреве | 2280 (Вт) |
| Потреб. ток при обогреве | 11,2 (А) |
| EER/C.O.P. при обогреве | 2,8 (Вт/Вт) |
| Расход воздуха внутренним блоком | 900/1050/1150 (м$^3$/ч) |
| Уровень шума внут. блока | 41/45/48 (дБ (А)) |
| Длина внут. блока | 1045 (мм) |
| Высота внут. блока | 315 (мм) |
| Глубина внут. блока | 235 (мм) |
| Вес внут. блока | 12 (кг) |
| Уровень шума наруж. блока | 58 (дБ (А) |
| Длина наруж. блока | 845 (мм) |
| Высота наруж. блока | 700 (мм) |
| Глубина наруж. блока | 320 (мм) |
| Вес наруж. блока | 49 (кг) |
| Напряжение питания | 220-240/50 (В/Гц) |
| Потребляемый ток при охлаждении | 11,2 (А) |

**Вывод**

В данном разделе дипломной работе автор рассмотрел и рассчитал световые и воздушные условия на рабочем месте. Это два показателя, которые должны соблюдаться в норме для комфортной и продуктивной работы. Согласно расчетам, чтобы осветить комнату размером 14 $м^2$, недостаточно окна размерами в два метра в высоту и в два метра в ширину. Для комфортного освещения необходимо комбинированное освящение, включающее в себя как естественное и искусственное освящение. Согласно расчетам искусственное освящение должно включать в себя 3 лапы с определенными параметрами, основным из этих параметров является световой поток излучения, который должен быть равен 3120 Лк. При соблюдении данных условий, в данном помещении можно будет работать в темное время суток.

Так же была рассмотрена и рассчитана система вентиляции. Согласно расчетам для поддержания благоприятного для работы воздушного окружения, необходим один кондиционер, который способен обеспечить подачу воздуха минимум 310,5 $м^3$ /ч. В данной работе для поддержания воздушной атмосферы автор выбрал кондиционер Ditreex (до 900 $м^3$ /ч).

## Заключение

В ходе выполнения дипломного проекта были рассмотрены различные типы ботов и их функционал. Был проведен анализ их функционала, интерфейса и систем безопасности. Исходя из произведенных анализов, разработан бот, который позволяет автоматизировать процесс сбора персональных данных. Бот работает с открытыми источниками, в качестве открытых источников подразумеваются социальные сети. Важно отметить то, что для работы с ботом пользователь обязан получить разрешение на сбор информации искомого человека.

Для разработки был использован язык программирования Python 3.6. Данный язык был выбран, поскольку содержит все необходимые библиотеки для работы с Telegram ботами.

Были выполнены следующие задачи:

– рассмотрены существующие боты и их виды;

– проведен анализ сильных и слабых сторон различных ботов;

– выявлены критерии, характеризующие бота как качественного;

– проанализированы техники разработки различных ботов и их структура;

– разработана система защиты;

– ручное тестирование бота;

– устранение несоответствий, связанных в процессе ручного тестирования;

– рассчитана экономическая эффективность, цена реализации продукта составила 1 155 050 тенге;

– произведены расчеты, согласно стандартам БЖД;

– подведены итоги.

Система защиты включает в себя следующие разработки:

– разграничение доступа;

– ведение лог-журналов и логирование запросов;

– фильтрация данных, входящих от пользователя в процессе запроса;

– реализована система обработки ошибок.

Целью работы является разработка бота, позволяющего автоматизировать процесс сбора информации.

Достоинством темы автора заключается в том, что разработанная автором программа объединяет в себе множество положительных критерий, которые были выявлены в процессе изучения различных ботов, это позволило разработать качественного, надежного, полезного и безопасного с точки зрения информационной безопасности бота. Бот включает в себя удобный и интуитивно понятный интерфейс.

Весь бот реализован с использованием деления одной большой задачи на более маленькие, это позволило с легкостью изменять функционал бота, не

переписывая его полностью. Это так же позволяет внедрять в код бота любые новые модули без многочисленных изменений.

## Перечень сокращений

ПО – Программное обеспечение.
ПП – Программный продукт.
ПК – Персональный компьютер.
ИС – Информационная система.

## Перечень терминов

Парсинг – автоматизированный процесс сбора информации.

Брутфорс – автоматизированный метод подбора пароля к аккаунту.

RCE – удаленное выполнение кода.

LFI – локальное включение файлов.

VPN – виртуальная персональная сеть.

XSS – межсайтовое внедрение Java Script.

Token – уникальный идентификатор пользователя.

Cookies – уникальные идентификаторы пользователя.

Хеширование – процесс шифрования данных, посредством хеш-функции.

Хеш – уникальные данные полученные в процессе хеширования.

Регулярное выражение – структура позволяющая на основе сигнатур определить информацию для парсинга.

## Список литературы

1 Что такое боты и как они работают URL: https://ru.epicstars.com/botyi-telegram/ (дата обращения 12.01.2019)

2 Виды ботов URL: https://psycho.ru/library/3791 (дата обращения 13.01.2019)

3 Разработка Telegram бота URL: https://khashtamov.com/ru/create-telegram-bot-in-python/ (дата обращения 10.01.2019)

4 Разработка Telegram бота URL: https://habr.com/ru/post/262247/ (дат обращения 11.01.2019)

5 Бот GetContact URL: https://pikabu.ru/story/getcontact__kto_kak _zapisan_ v_chuzhikh_kontaktlistakh_5748737 (дата обращения 08.02.2019)

6 Бот DrWeb для Telegram URL: https://news.drweb.ru/show/?i=9852&lng=ru (дата обращения 10.02.2019)

7 Самоучитель Python URL: https://pythonworld.ru/samouchitel-python (дата обращения 10.01.2019 - 15.04.2019)

8 Средства анонимности в сети URL: https://habr.com/ru/post/190396/ (дата обращения 03.03.2019)

9 Requests + proxy в Python URL: https://stackoverflow.com/questions/8287628/ proxies-with-python-requests-module (дата обращения 04.03.2019)

10 Голубицкая Е. А., Жигульская Г. М. Экономика связи. – М. Радио и связь, 2000.

11 Аманжолова К. Б., Алибаева С. А. Экономика предприятии телекоммуникации: Учебное пособие. - Алматы: АИЭС, 2003.

12 СНиП II-4-79. Естественное и искусственное освещение. Нормы проектирования. -М.: Стройиздат, 1980.- 48 с.

13 Нормы микроклимата URL: http://adilet.zan.kz/rus/docs /V050003789 (дата обращения: 21.02.2019).