МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

**«Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева»**

**(СибГУ)**

ИНСТИТУТ (ФАКУЛЬТЕТ) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

НАПРАВЛЕНИЕ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ПРОФИЛЬ ПОДГОТОВКИ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ**

РАЗРАБОТКА СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ В ЕЖЕДНЕВНОМ БЫТУ

Студент Жудрак Антон Петрович

Руководитель Коморовский Витольд Станиславович

Рецензент (Ф.И.О.)

Нормоконтроль (Ф.И.О.)

Допускается к защите

Зав. Кафедрой (Ф.И.О.)

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г.

Красноярск 20\_\_\_ г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

**«Сибирский государственныйуниверситет науки и технологий имени академика М. Ф. Решетнёва»**

**(СибГУ)**

УТВЕРЖДАЮ

Зав. кафедрой ИУС

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_г.

**ЗАДАНИЕ**

**на магистерскую диссертацию**

Студенту\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_группы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

направления\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

магистерской программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тема диссертации\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Утверждена приказом по университету №\_\_\_\_\_\_\_от «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_\_\_г.

Срок сдачи студентом первого варианта диссертации «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_\_\_г.

Срок сдачи студентом законченной диссертации «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_г.

Исходные данные к диссертации\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Содержание диссертации (перечень вопросов, подлежащих разработке)  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Перечень графического материала с указанием обязательных чертежей

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нормоконтроль\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания «\_\_\_»\_\_\_\_\_\_ 20\_\_г. \_\_\_\_\_\_\_\_\_\_ /Коморовский Витольд Станиславович/

**АННОТАЦИЯ**

ЖУДРАК АНТОН ПЕТРОВИЧ

РАЗРАБОТКА СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ В ЕЖЕДНЕВНОМ БЫТУ

НАПРАВЛЕНИЕ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

МАГИСТЕРСКАЯ ПРОГРАММА \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

УЧЕБНАЯ ГРУППА \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ГОД ЗАЩИТЫ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ПРИМЕР.** В магистерской диссертации исследован вопрос и далее по работе.

**THE SUMMARY**

DEVELOPMENT OF THE SYSTEM OF SUPPORT OF DECISION-MAKING DURING EVERYDAY LIFE.

**EXAMPLE.** In master’s thesis investigated the issue of strategic planning and etc.

Содержание

[Введение 6](#_Toc483487247)

[1 Человеко-машинное взаимодействие 7](#_Toc483487248)

[1.1 Боты в мессенджерах 7](#_Toc483487249)

[1.2 Кому и для чего нужны боты 8](#_Toc483487250)

[1.3 Чем бот лучше реального пользователя 9](#_Toc483487251)

[1.4 Самые популярные боты 9](#_Toc483487252)

[1.5 Почему боты не заменят мобильные приложения 11](#_Toc483487253)

[1.6 Стоимость разработки бота 12](#_Toc483487254)

[2 Системы поддержки принятия решений 13](#_Toc483487255)

[2.1 История развития «систем поддержки принятия решений» 17](#_Toc483487256)

[3 Публичные API 19](#_Toc483487257)

[4 Выбор среды для разработки 24](#_Toc483487258)

[4.1Причины использования Python 25](#_Toc483487259)

[4.1.1 Качество программного обеспечения 25](#_Toc483487260)

[4.1.2 Высокая скорость разработки 25](#_Toc483487261)

[4.1.3 Переносимость программ 26](#_Toc483487262)

[4.1.4 Библиотеки поддержки 26](#_Toc483487263)

[4.1.5 Интеграция компонентов 26](#_Toc483487264)

[4.2 История создания 27](#_Toc483487265)

[4.1.6 Влияние других языков 28](#_Toc483487266)

[4.3 Функциональные возможности 29](#_Toc483487267)

[4.4 Прикладное применение Python 31](#_Toc483487268)

[4.5 Преимущество Python перед другими языками высокого уровня 32](#_Toc483487269)

[4.6 Реализации Python 33](#_Toc483487270)

[4.7 Недостатки 34](#_Toc483487271)

[5 Телеграм(мессенджер) 36](#_Toc483487272)

[5.1 История создания телеграмма 36](#_Toc483487273)

[5.2Технология 38](#_Toc483487274)

[5.3 Основные возможности приложения Telegram 39](#_Toc483487275)

[5.4 Преимущества и недостатки использования Telegram 40](#_Toc483487276)

[5.5 Секретный чат 40](#_Toc483487277)

[6 Процесс создания бота для телеграмм 42](#_Toc483487278)

[6.1 Подготовка перед написание 42](#_Toc483487279)

[6.2 Регистрация 43](#_Toc483487280)

[4.2 Листинг процедур бота 44](#_Toc483487281)

[4.3 Проверка работы бота 44](#_Toc483487282)

[4.4 Декоративные моменты 44](#_Toc483487283)

[Источники 44](#_Toc483487284)

Введение

Стандартизация во все большей и большей степени определяет деятельность человека в профессиональной сфере. Это касается и информатики. В частности, в связи с развитием систем поддержки принятия решений. Почему приходится обращать внимание на стандартизацию? В первую очередь потому, что всех интересует коммерческое применение прикладных решений. Продукт (товар, услуга) становится массовым только после унификации требований к нему, пусть даже с учетом пожеланий «узких» групп потребителей

**Актуальность исследования.**

**Цель исследования.**

**Объект исследования.**

**Предмет исследования**.

**Методы исследования**

**Научная новизна исследования:**

Не окончательный вариант, необходима доработка.

1 Человеко-машинное взаимодействие

1.1 Боты в мессенджерах

Бот – специальная программа, выполняющая автоматически и/или по заданному расписанию какие-либо действия через интерфейсы, предназначенные для людей. При обсуждении компьютерных программ термин употребляется в основном в применении к интернету[1].

Мода на ботов не появилась – она вернулась. Первые боты были придуманы в начале 90х годов и использовались в основном для чатов, работающий на базе протокола IRC.

IRC (англ. Internet Relay Chat) — протокол прикладного уровня для обмена сообщениями в режиме реального времени. Разработан в основном для группового общения, также позволяет общаться через личные сообщения и обмениваться данными, в том числе файлами.

Например, «Телевикторина» – чат, где бот задает группе людей вопросы, а те должны угадать слово. Такой бот был особенно популярен более 10 лет назад в локальных домашних сетях и в вебовских чатах. С течением времени бот потерял свою популярность, т.к. он просто банально всем наскучил. С развитием кроссплатформенных мессенджеров таких как Telegram, Facebook, Viber, WhatsApp, Skype, Вконтакте, боты снова обрели свою популярность. Теперь благодаря новым платформам таким как iOS и Android у каждого владельца гаджета есть установленный мессенджер, которым он пользуется постоянно.

На данный момент 85% всех ботов – это боты однодневки, жизненный цикл, который очень короткий. В России он не более двух недель, это связано с падением интереса разработчика к тому что он создал. Основную массу ботов создают программисты одиночки.

1.2 Кому и для чего нужны боты

У ботов сегодня две основные аудитории - те, кто хочет сэкономить время и силы и те, кому по разным причинам недоступны альтернативные варианты.

Например, сейчас в России большой спрос на недорогие смартфоны. Они стоят 2-3 тысячи рублей, но из-за технической слабости на них невозможно установить 30 разных приложений. На такой смартфон не поставить отдельное мобильное приложение с прогнозом погоды, отдельное с курсом валюты или с игрой – просто не хватит места. А вот боты с аналогичными функциями могут легко удовлетворить все эти потребности пользователя прямо в чате, занимая крайне мало места на телефоне.

Вторая часть аудитория — те, кто хочет сэкономить время и силы. Боты предоставляют нужную информацию в фоновом режиме и в привычной среде — там, где пользователь общается с друзьями. В результате не нужно переключаться между разными приложениями, чтобы совершить ту или иную операцию - к примеру, одолжить другу деньги в том же чате, где вы обсуждаете последние новости футбола.[2]

Боты – это маленькие умные помощники. Их прелесть в том, что вам не нужно покидать экосистему мессенджера, чтобы решить элементарную задачу: спросить, уточнить, узнать, посчитать. Каждый бот сам по себе, но вместе они образуют вашу личную армию ассистентов, каждому из которых вы даете поручение, а они безропотно и четко выполнят ваши пожелания.

Боты — это небольшие шестеренки и приводные ремни к вашим задачам, которые могут действовать слаженно, если с умом подогнать их друг к другу. Скажем, на какие-то конкретные действия вы можете настроить ботов к определенному времени. Например, доставлять вам с утра свежие новости, прогноз погоды, ситуацию по пробкам. А вечером – рекомендацию по фильму, смешные анекдоты и рецепт дня.

В умелых руках боты могут стать очень мощным ресурсом для организации своего времени и автоматизации повторяющихся действий.

1.3 Чем бот лучше реального пользователя

Бот, выполняя однообразную работу, не только экономит силы и время человека, но делает работу на более высоких скоростях.

У ботов лучше реакция и точность действий по сравнению с человеком – это находит применение в компьютерных играх, интернет-аукционах, рекламе, электронной биржевой торговле и так далее. Боты применяются для имитации человеческой деятельности, в частности, в чатах. Это так называемые «чат-боты».

Чат-бота можно настроить на выдачу адекватных ответов на человеческом языке. Конечно, при условии, что он распознает и поддерживает такого рода команды.

1.4 Самые популярные боты

На данный момент самые популярные боты, которые интересны массовой аудитории создаются в мессенджере телеграмм, так как там открытый исходный код, очень удобный API-инструментарий, и самые обширные возможности для их создания. Боты создаваемые в Telegram – это половина создаваемых всех ботов в мире.

Например:

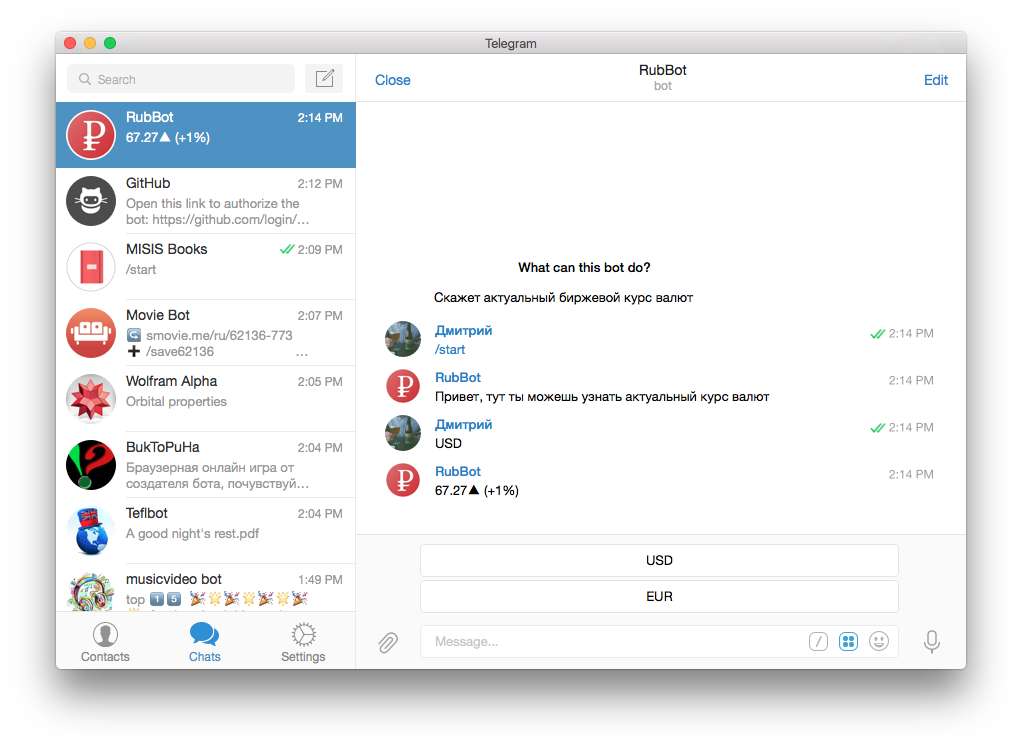


Рисунок 1.4.1 – RubBot (@rubbot)

Бот, который показывает актуальный курс валют

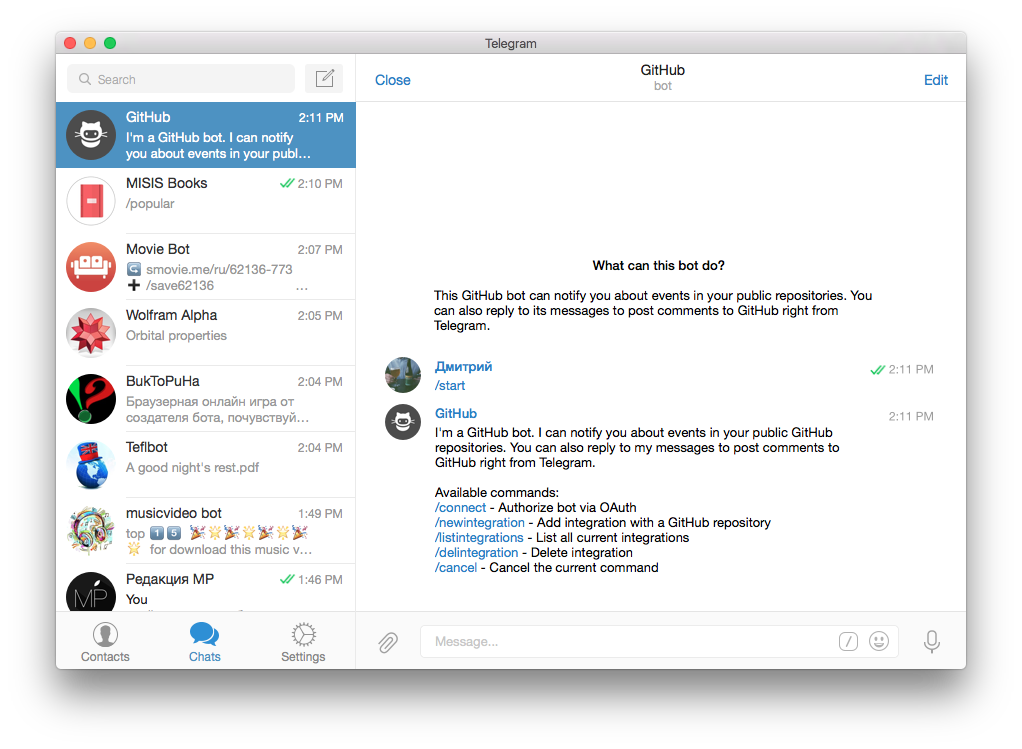


Рисунок 1.4.2 – GitHub(@githubbot)

Бот соединяется с репозиторием GitHub для получения уведомлений о правках файлов в режиме реального времени

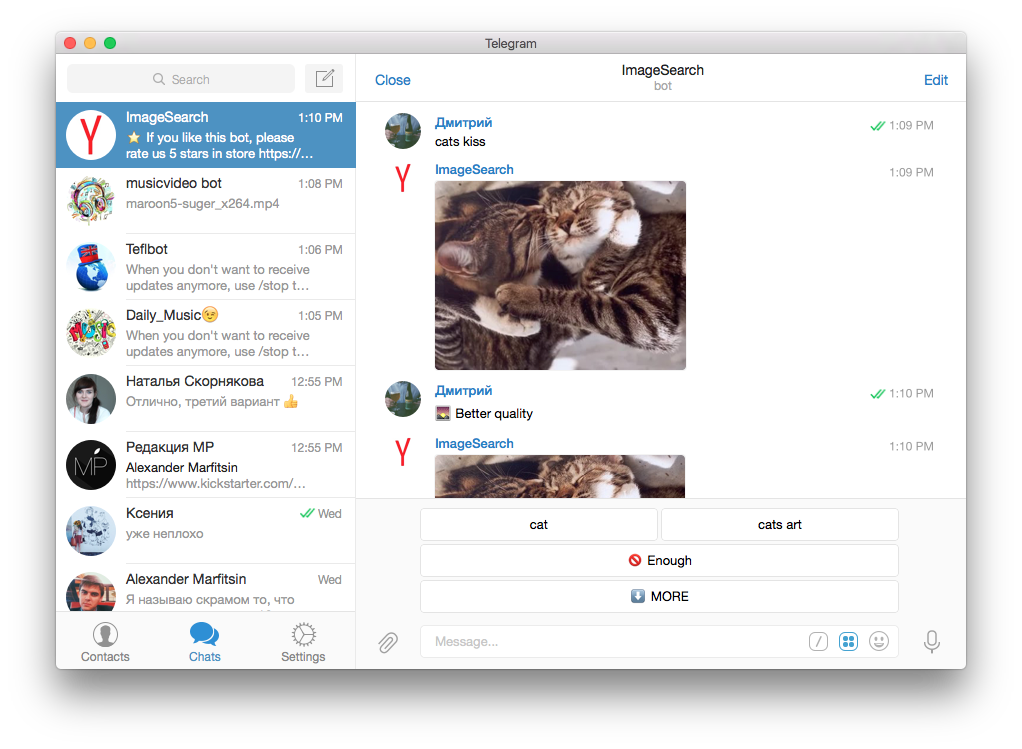


Рисунок 1.4.3 – ImageSearch (@imageSearchBot)

Бот находит любое изображение по вашему запросу. В опциях можно запросить улучшенное качество или похожие картинки

1.5 Почему боты не заменят мобильные приложения

У ботов, безусловно, будет своя аудитория. Причем, не маленькая. Чаты, WhatsApp, Telegram, Twitter — очень зрелые решения. Эти платформы со временем будут становиться только лучше, что даст почву для создания новых ботов.

А это еще больше привлечет внимание новых пользователей. Но нужно понимать, что боты — не та технология, которая завтра перевернет мир настолько, что все откажутся от мобильных приложений.

Боты всегда будут ограничены возможностями чата или мессенджера, которые существенно меньше возможностей полноценного мобильного приложения. Скорее все идет к формированию гибридного информационного пространства, в котором приложения на компьютере и на мобильных устройствах, в браузере и в мессенджерах представляют собой единое целое, а переход между ними для пользователя будет абсолютно прозрачным.

1.6 Стоимость разработки бота

Если заказывать в столичной ИТ-компании, то разработка сложного бота может обойтись и в 500 тысяч рублей. В региональной компании дешевле — от 50 тысяч до 150 тысяч рублей. Но дешевле всего заказывать у фрилансера — он попросит около 20 тысяч рублей.

2 Системы поддержки принятия решений

Системы поддержки принятия решений – это экспертные и автоматизированные системы управления, которые позволяют принимать необходимые решения, сложившиеся в той или иной ситуации.

Системы поддержки принятия решений – это компьютерные системы, предназначенные для сбора и анализа больших объёмов информации с целью оценивания возможных вариантов решений.

По факту любой механизм, помогающий человеку принимать какие-либо решения, на основе не только интуиции, можно назвать системой поддержки принятия решений. Поэтому элементы систем поддержки принятия решений есть в системе, где есть возможность построить какой-либо график либо получить отчёт. К сожалению СППР – это всего лишь система и она не может заменить человека, но тем не менее информационная сложность определяется необходимостью учёта большого объёма данных, обработка которых без помощи современных вычислительных машин практически невыполнима. В этих условиях число возможных решений, как правило, весьма велико, и выбор наилучшего из них «на глаз» без всестороннего анализа может приводить к грубым ошибкам. [3] Основные функции таких систем следующие:

* оценивание проблемной ситуации и имеющихся ограничений;
* выявление приоритетов и их ранжирование;
* формулировка предпочтений лица принимающего решения (ЛПР);
* генерирование альтернатив;
* оценивание альтернатив;
* анализ возможных последствий и рисков;
* выбор наилучшей альтернативы.

Для реализации указанных функций в системах поддержки принятия решений используются следующие методы:

* информационный поиск;
* анализ данных;
* поиск знаний в базах данных;
* имитационное моделирование;
* когнитивное моделирование;
* ситуационное моделирование и д.р.

Основными компонентами систем поддержки принятия решений являются:

* хранилища данных;
* средства обработки данных;
* многомерные базы данных;
* data mining.

Лицо принимающее решение может использовать системы поддержки принятия решений на всех основных этапах разработки и принятия решений:

* распознавания проблемы;
* генерирования и анализа альтернатив;
* выбор наилучшей альтернативы;
* реализации решений.

Системы поддержки принятия решений помогают решать такие типовые задачи:

* анализ примеров;
* идентификация зависимостей;
* анализ чувствительности;
* анализ возможностей;
* анализ влияния;
* анализ данных;
* сравнение и агрегирование;
* анализ рисков;
* оптимизация.

Взаимодействие с ЛПР выделяют три основных класса систем поддержки принятия решений:

* пассивные;
* активные;
* кооперативные.

По способу поддержки различают такие классы систем:

* модельно-ориентированные СППР. Они используют для своей работы: статистические, экономические и др. типы моделей;
* коммуникационно-ориентированные СППР. Обеспечивают взаимодействие нескольких пользователей в процессе разработки решения;
* СППР, ориентированные на данные. Имеют доступ к необходимым базам данных, а также средства для работы с этими базами;
* СППР, ориентированные на документы. Содержат средства обработки, имеющихся документов деструктурированной информации;
* СППР, ориентированные на знания. Обеспечивают решение определенных классов задач на основе знаний, которые накоплены в соответствующей предметной области.

По сфере использования выделяют общесистемные и настольные СППР.

Общесистемные СППР ориентированы на взаимодействие с большими системами хранения данных и могут одновременно работать с несколькими, а иногда и со многими пользователями.

Настольные СППР имеют более ограниченные возможности по хранению и обработке информации и предназначены для работы одного пользователя.

С точки зрения архитектуры выделяют-таки типы системы поддержки принятия решений:

* функциональные СППР. Предназначены для небольших организаций, такие СППР обеспечивают компактность и оперативность работы. Обычно используются для узкого круга проблем, а качество получаемых результатов окажется недостаточным из-за необеспечения предварительной очистки данных;
* СППР с независимыми витринами данных. Включает несколько витрин данных, каждая из которых предназначена для своего класса задач и ориентирована на определенную группу пользователей, это дает возможность повысить производительность, но приводит к необходимости дублировать данные. Недостаток таких систем, это сложность наполнения витрин и отсутствия консолидации данных, что ведет к отсутствию у ЛПР целостной картины объекта управления;
* СППР на основе двухуровневых хранилищ данных. Применяются там, где данные консолидированы в единую систему. Это дает возможность предложить унифицированные способы обработки информации, однако такие СППР не могут обеспечить возможность структурирования данных для отдельных групп пользователей, а также ограничить доступ пользователя для той или иной информации. Кроме того, необходимо создание группы специалистов для обслуживания данной системы;
* СППР на основе трехуровневых хранилищ данных. Позволяет осуществить консолидацию информации и ее предварительную очистку, а так же сформировать витрину данных, которая будет использоваться отдельными группами пользователей. Такие СППР содержат единую корпоративную модель данных и характеризуется гарантированной производительностью, однако данные в таких системах являются избыточными и это ведёт к осложнению хранения этих данных, а также к возникновению проблемы согласования архитектуры с различающимися запросами различных групп пользователей.

Некоторые примеры задач, решаемые с помощью систем поддержки принятия решений:

* выбор маршрутов перевозок;
* управление пакетами акций;
* управление производственными ресурсами предприятий;
* составление расписаний;
* управление проектами и др.

2.1 История развития «систем поддержки принятия решений»

До середины 60-х годов прошлого века создание больших информационных систем (ИС) было чрезвычайно дорогостоящим, поэтому первые ИС менеджмента (так называемые Management Information Systems – MIS) были созданы в эти годы лишь в достаточно больших компаниях. MIS предназначались для подготовки периодических структурированных отчетов для менеджеров.

В конце 60-х годов появляется новый тип ИС – модель-Ориентированные СППР (Model-oriented Decision Support Systems – DSS) или системы управленческих решений (Management Decision Systems - MDS).

По мнению первооткрывателей СППР Keen P. G. W., Scott Morton M.S. (1978), концепция поддержки решений была развита на основе «теоретических исследований в области принятия решений и технических работ по созданию интерактивных компьютерных систем».

В 1971 г. – опубликована книга Scott Morton‘а, в которой впервые были описаны результаты внедрения СППР, основанной на использовании математических моделей.

1974 г. – в работе дано определение первые ИС менеджмента - MIS: «MIS – это интегрированная человеко-машинная система обеспечения информацией, поддерживающая функции операций, менеджмента и принятия решений в организации. Системы используют компьютерную технику и программное обеспечение, модели управления и принятия решений, а также базу данных».

1975 г. - J.D.C.Little в своей работе предложил критерии проектирования СППР в менеджменте.

1978 г. – опубликован учебник по СППР, в котором исчерпывающе описаны аспекты создания СППР: анализ, проектирование, внедрение, оценка и разработка.

1980 г. – опубликована диссертация S. Alter, в которой он дал основы классификации СППР.

1981 г. – Bonczek, Holsapple и Whinston в книге создали теоретические основы проектирования СППР. Они выделили 4 необходимых компонента, присущих всем СППР: 1) Языковая система (Language System - LS) – СППР может принимать все сообщения; 2) Система презентаций (Presentation System (PS)) (СППР может выдавать свои сообщения); 3) Система знаний (Knowledge System - KS) – все знания СППР сохраняет; 4) Система обработки задач (Problem-Processing System (PPS)) – программный «механизм», который пытается распознать и решить задачу во время работы СППР.

1981 г. – В книге R.Sprague и E.Carlson описали, каким образом на практике можно построить СППР. Тогда же была разработана информационная система руководителя (Executive Information System (EIS)) – компьютерная система, предназначенная для обеспечения текущей адекватной информации для поддержки принятия управленческих решений менеджером.

Начиная с 1990-х, разрабатываются так называемыеData Warehouses - хранилища данных. Хранилище данных – это очень большая предметно- ориентированная информационная корпоративная база данных, предназначенная для подготовки отчётов, анализа бизнес-процессов и поддержки принятия решений. Строится на базе клиент-серверной архитектуры, реляционной СУБД и утилит поддержки принятия решений. Данные, поступающие в хранилище данных, становятся доступны только для чтения.

27 октября 2005 года в Москве на Международной конференции «Информационные и телемедицинские технологии в охране здоровья» (ITTHC 2005), А. Пастухов (Россия) представил СППР нового класса - PSTM (Personal Information Systems of Top Managers). Основным отличием PSTM от существующих СППР является построение системы для конкретного лица, принимающее решение, с предварительной логико-аналитической обработкой информации в автоматическом режиме и выводом информации на один экран.

3 Публичные API

API (Application Programming Interface) — это интерфейс программирования, интерфейс создания приложений.

Из определения API – это в первую очередь интерфейс, который позволяет разработчикам использовать готовые блоки для создания приложений. Например, с разработкой мобильных приложений в роли API может выступать библиотека для работы с «умным домом», всё практически реализовано в этой библиотеке, а программист лишь обращается к этому API в своем коде.

API представляет собой REST приложение, принимающее и возвращающее данные в формате JSON. С его помощью можно получить список событий, категорий, организаций и связанной с ними информации.

Есть публичные API и большинство таких на западном сегменте, которые предоставляют доступ к своим базам. На просторах рунета ощущается их явный недостаток, однако имеется некоторый список таких сервисов.

Популярные публичные API в сегменте РФ:

* yandex.ru;
* vk.com;
* mail.ru;
* ok.ru.

В мире:

* google.com;
* twitter.com;
* facebook.com;
* sandspace.com.

Публичная информация отдается через API без необходимости авторизации. Для получения доступа к непубличной информации, либо для изменения данных, нужна авторизация по протоколу OAuth2.

Пример получения информации о пользователе с сервиса api.github.

Структура запроса: <domain>/<endpoint>/<id>

[*https://api.github.com/users/HD-7*](https://api.github.com/users/HD-7)

Посылаем запрос на ресурс и ожидаем ответа.

Ответ с ресурс api.github:

*{*

*"login": "HD-7",*

*"id": 22838877,*

*"avatar\_url": "https://avatars2.githubusercontent.com/u/22838877?v=3",*

*"gravatar\_id": "",*

*"url": "https://api.github.com/users/HD-7",*

*"html\_url": "https://github.com/HD-7",*

*"followers\_url": "https://api.github.com/users/HD-7/followers",*

*"following\_url": "https://api.github.com/users/HD-7/following{/other\_user}",*

*"gists\_url": "https://api.github.com/users/HD-7/gists{/gist\_id}",*

*"starred\_url": "https://api.github.com/users/HD-7/starred{/owner}{/repo}",*

*"subscriptions\_url": "https://api.github.com/users/HD-7/subscriptions",*

*"organizations\_url": "https://api.github.com/users/HD-7/orgs",*

*"repos\_url": "https://api.github.com/users/HD-7/repos",*

*"events\_url": "https://api.github.com/users/HD-7/events{/privacy}",*

*"received\_events\_url": "https://api.github.com/users/HD-7/received\_events",*

*"type": "User",*

*"site\_admin": false,*

*"name": "Damir Rashitovich Khakimov",*

*"company": null,*

*"blog": "",*

*"location": "Krasnoyarsk region",*

*"email": null,*

*"hireable": null,*

*"bio": null,*

*"public\_repos": 1,*

*"public\_gists": 0,*

*"followers": 0,*

*"following": 0,*

*"created\_at": "2016-10-14T13:28:01Z",*

*"updated\_at": "2017-05-03T05:16:34Z"*

*}*

Из данных полученных от запроса, можно увидеть небходимую информацию, допустим такую как *"public\_repos": 1*, которая говорит о имеющемся репозитории в количестве раной одному. Изменим данные о пользователе и добавим ещё один репозиторий и отправим запрос снова. На рисунке 3.1. отображена практически вся информация с запроса. В данным момент видно, о наличии одного репозитория.

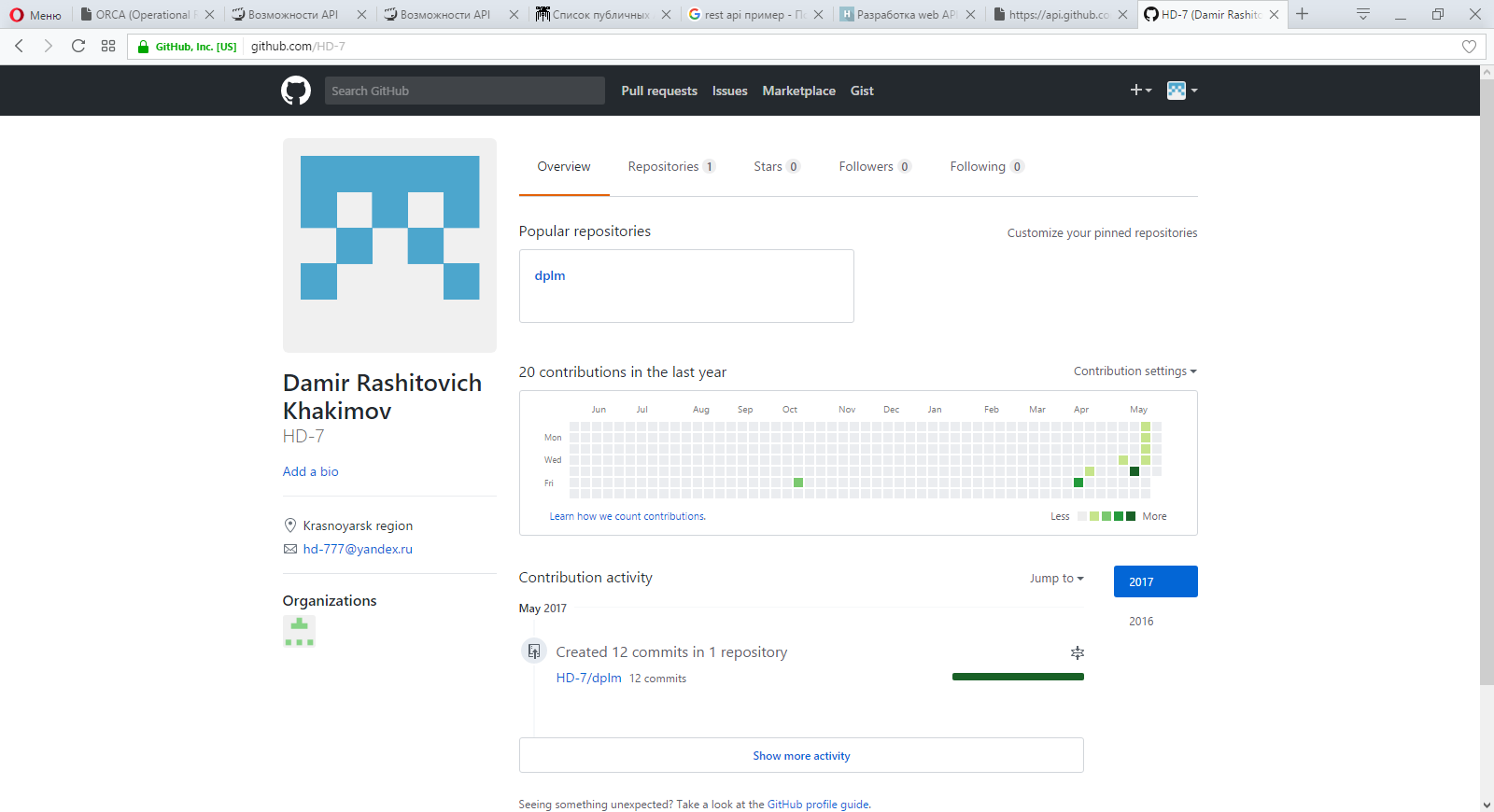


Рисунок 3.1 – Данные о пользователе HD-7 на github.

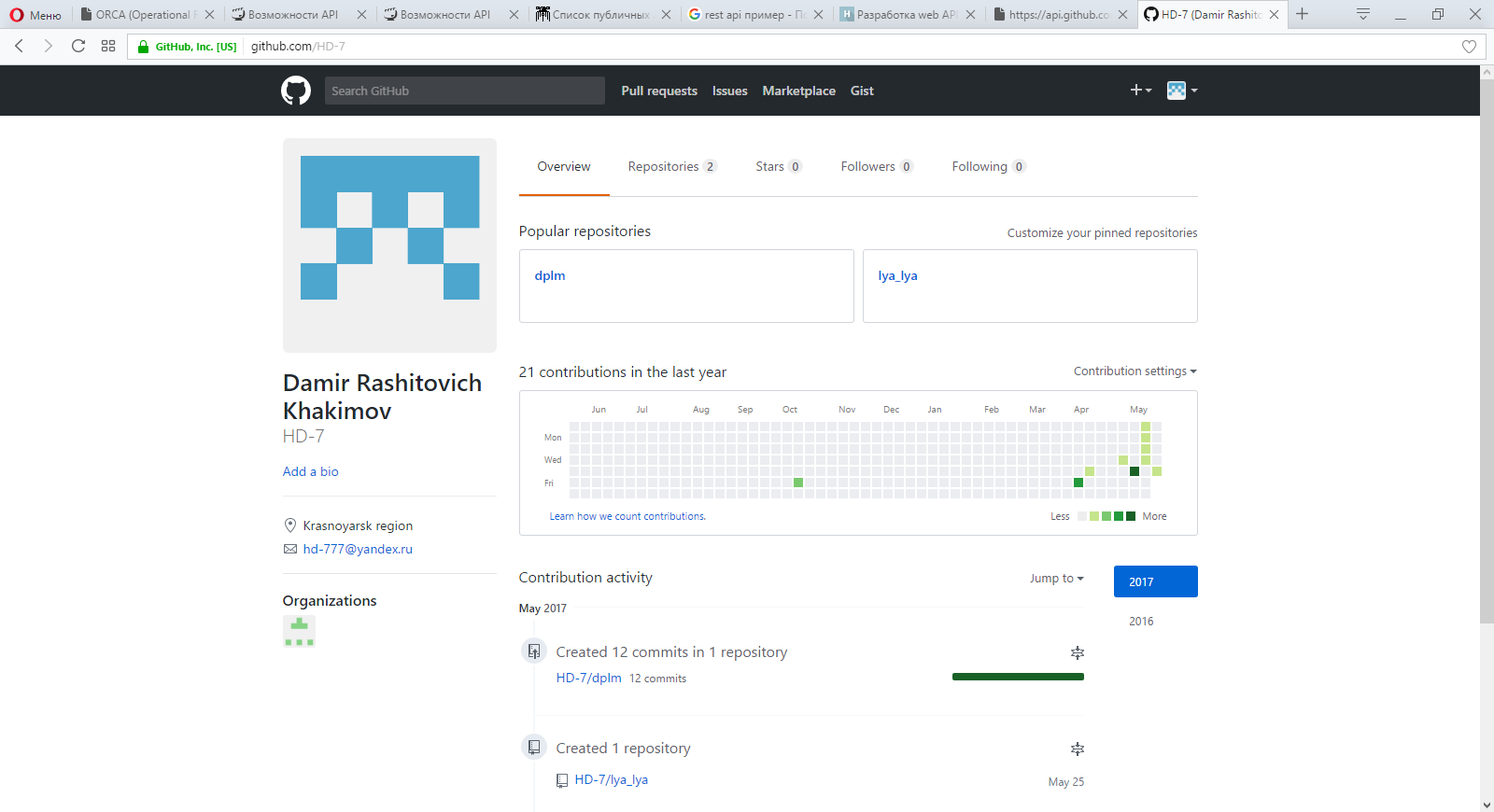


Рисунок 3.2– Наличие двух репозиториев.

Повтор запроса о пользователе. Ответ:

*{*

*"login": "HD-7",*

*"id": 22838877,*

*"avatar\_url": "https://avatars2.githubusercontent.com/u/22838877?v=3",*

*"gravatar\_id": "",*

*"url": "https://api.github.com/users/HD-7",*

*"html\_url": "https://github.com/HD-7",*

*"followers\_url": "https://api.github.com/users/HD-7/followers",*

*"following\_url": "https://api.github.com/users/HD-7/following{/other\_user}",*

*"gists\_url": "https://api.github.com/users/HD-7/gists{/gist\_id}",*

*"starred\_url": "https://api.github.com/users/HD-7/starred{/owner}{/repo}",*

*"subscriptions\_url": "https://api.github.com/users/HD-7/subscriptions",*

*"organizations\_url": "https://api.github.com/users/HD-7/orgs",*

*"repos\_url": "https://api.github.com/users/HD-7/repos",*

*"events\_url": "https://api.github.com/users/HD-7/events{/privacy}",*

*"received\_events\_url": "https://api.github.com/users/HD-7/received\_events",*

*"type": "User",*

*"site\_admin": false,*

*"name": "Damir Rashitovich Khakimov",*

*"company": null,*

*"blog": "",*

*"location": "Krasnoyarsk region",*

*"email": null,*

*"hireable": null,*

*"bio": null,*

*"public\_repos": 2,*

*"public\_gists": 0,*

*"followers": 0,*

*"following": 0,*

*"created\_at": "2016-10-14T13:28:01Z",*

*"updated\_at": "2017-05-03T05:16:34Z"*

*}*

Из данного ответа видно, что *"public\_repos": 2* изменилось, значит запрос работает должным образом. Так же можно получить данные о пользователе, которые находятся в публичном доступе. Допустим *"email": null* принимает значение *null* в том случае, если нет данных либо это информация скрыта и имеет значение приватности.

4 Выбор среды для разработки

В связи с наблюдаемым в настоящее время стремительным развитием персональной вычислительной техники, происходит постепенное изменение требований, предъявляемых к языкам программирования. Все большую роль начинают играть интерпретируемые языки, поскольку возрастающая мощь персональных компьютеров начинает обеспечивать достаточную скорость выполнения интерпретируемых программ. А единственным существенным преимуществом компилируемых языков программирования является создаваемый ими высокоскоростной код. Когда скорость выполнения программы не является критичной величиной, наиболее правильным выбором будет интерпретируемый язык, как более простой и гибкий инструмент программирования.

В связи с этим, определенный интерес представляет рассмотрение сравнительно нового языка программирования Python, который был создан его автором Гвидо ван Россумом в начале 90-х годов.

Отличительные характеристики языка:

* очень низкий порог вхождения, уже после одного дня изучения можно начать писать простые программы;
* минималистичный язык, с небольшим количеством конструкций;
* краткий код;
* прекрасно подходит для создания программ-обёрток, поддерживается импорт Си-библиотек;
* существует большое количество реализаций: CPython (основная реализация); Jython (реализация для JVM); IronPython (CLR); PyPy;
* очень хорошая поддержка математических вычислений (библиотеки NumPy, SciPy);
* используется для обработки естественных языков (NLTK);
* большое количество развитых web-фреймворков (Django, TurboGear, CherryPy, Flask).

Язык можно рекомендовать всем, кто только начинает программировать, как первый язык программирования в жизни.

4.1Причины использования Python

4.1.1 Качество программного обеспечения

Для многих основное преимущество языка Python заключается в удобочитаемости, ясности и более высоком качестве, отличающими его от других инструментов в мире языков программирования. Программный код на языке Python читается легче, а значит, многократное его использование и обслуживание выполняется гораздо проще, чем использование программного кода на других языках сценариев. Единообразие оформления программного кода на языке Python облегчает его понимание даже для тех, кто не участвовал в его создании. Кроме того, Python поддерживает самые современные механизмы многократного использования программного кода, каким является объектно-ориентированное программирование (ООП).

4.1.2 Высокая скорость разработки

По сравнению с компилирующими или строго типизированными языками, такими как C, C++ и Java, Python во много раз повышает производительность труда разработчика. Объем программного кода на языке Python обычно составляет треть или даже пятую часть эквивалентного программного кода на языке C++ или Java. Это означает меньший объем ввода с клавиатуры, меньшее количество времени на отладку и меньший объем трудозатрат на сопровождение. Кроме того, программы на языке Python запускаются сразу же, минуя длительные этапы компиляции и связывания, необходимые в некоторых других языках программирования, что еще больше увеличивает производительность труда программиста.

4.1.3 Переносимость программ

Большая часть программ на языке Python выполняется без изменений на всех основных платформах. Перенос программного кода из операционной системы Linux в Windows обычно заключается в простом копировании файлов программ с одной машины на другую. Более того, Python предоставляет массу возможностей по созданию переносимых графических интерфейсов, программ доступа к базам данных, веб-приложений и многих других типов программ. Даже интерфейсы операционных систем, включая способ запуска программ и обработку каталогов, в языке Python реализованы переносимым способом.

4.1.4 Библиотеки поддержки

В составе Python поставляется большое число собранных и переносимых функциональных возможностей, известных как *стандартная библиотека*. Эта библиотека предоставляет массу возможностей, востребованных в прикладных программах, начиная от поиска текста по шаблону и заканчивая сетевыми функциями. Кроме того, Python допускает расширение как за счёт ваших собственных библиотек, так и за счет библиотек, созданных сторонними разработчиками. Из числа сторонних разработок можно назвать инструменты создания веб-сайтов, программирование математических вычислений, доступ к последовательному порту, разработку игровых программ и многое другое. Например, расширение NumPy позиционируется как свободный и более мощный эквивалент системы программирования математических вычислений Mathlab.

4.1.5 Интеграция компонентов

Сценарии Python легко могут взаимодействовать с другими частями приложения благодаря различным механизмам интеграции. Эта интеграция позволяет использовать Python для настройки и расширения функциональных возможностей программных продуктов. На сегодняшний день программный код на языке Python имеет возможность вызывать функции из библиотек на языке C/C++, сам вызываться из программ, написанных на языке C/C++, интегрироваться с программными компонентами на языке Java, взаимодействовать с такими платформами, как COM и .NET, и производить обмен данными через последовательный порт или по сети с помощью таких протоколов, как SOAP, XML-RPC и CORBA.

4.2 История создания

Разработка языка Python была начата в конце 1980-х годов сотрудником голландского института CWI Гвидо ван Россумом. Для распределённой ОС Amoeba требовался расширяемый скриптовый язык, и Гвидо начал писать Python на досуге, позаимствовав некоторые наработки для языка ABC (Гвидо участвовал в разработке этого языка, ориентированного на обучение программированию). В феврале 1991 года Гвидо опубликовал исходный текст в ньюсгруппе alt.sources. С самого начала Python проектировался как объектно-ориентированный язык.

Название языка произошло вовсе не от вида пресмыкающихся. Автор назвал язык в честь популярного британского комедийного телешоу 1970-х "Летающий цирк Монти Пайтона". Впрочем, всё равно название языка чаще ассоциируют именно со змеёй, нежели с передачей — пиктограммы файлов в KDE или в Microsoft Windows и даже эмблема на сайте python.org (до выхода версии 2.5) изображают змеиные головы.

Наличие дружелюбного, отзывчивого сообщества пользователей считается наряду с дизайнерской интуицией Гвидо одним из факторов успеха Python. Развитие языка происходит согласно чётко регламентированному процессу создания, обсуждения, отбора и реализации документов PEP (англ. *Python Enhancement Proposal*) — предложений по развитию Python.

3 декабря 2008 года, после длительного тестирования, вышла первая версия Python 3000 (или Python 3.0, также используется сокращение Py3k). В Python 3000 устранены многие недостатки архитектуры с максимально возможным (но не полным) сохранением совместимости со старыми версиями Python. На сегодня поддерживаются обе ветви развития (Python 3.x и 2.x)

4.1.6 Влияние других языков

Появившись сравнительно поздно, Python создавался под влиянием множества языков программирования:

* ABC — отступы для группировки операторов, высокоуровневые структуры данных (map) (Python фактически создавался как попытка исправить ошибки, допущенные при проектировании ABC);
* Modula-3 — пакеты, модули, использование else совместно с try и except, именованные аргументы функций (на это также повлиял Common Lisp);
* С, C++ — некоторые синтаксические конструкции (как пишет сам Гвидо ван Россум — он использовал наиболее непротиворечивые конструкции из С, чтобы не вызвать неприязнь у С-программистов к Python);
* Smalltalk — объектно-ориентированное программирование;
* Lisp — отдельные черты функционального программирования (lambda, map, reduce, filter и другие);
* Fortran — срезы массивов, комплексная арифметика;
* Miranda — списочные выражения;
* Java — модули logging, unittest, threading (часть возможностей оригинального модуля не реализована), xml.sax стандартной библиотеки, совместное использование finally и except при обработке исключений, использование @ для декораторов;
* Icon — генераторы.

Большая часть других возможностей Python (например, байт-компиляция исходного кода) также была реализована ранее в других языках

4.3 Функциональные возможности

С точки зрения функциональных возможностей Python можно назвать гибридом. Его инструментальные средства укладываются в диапазон между традиционными языками сценариев (такими как Tcl, Scheme и Perl) и языками разработки программных систем (такими как C, C++ и Java). Python обеспечивает простоту и непринужденность языка сценариев и мощь, которую обычно можно найти в компилирующих языках. Превышая возможности других языков сценариев, такая комбинация делает Python удобным средством разработки крупномасштабных проектов. Ниже приводится список основных возможностей, которые есть в арсенале Python:

*Динамическая типизация*

Python сам следит за типами объектов, используемых в программе, благодаря чему не требуется писать длинные и сложные объявления в программном коде. В действительности, в языке Python вообще отсутствуют понятие типа и необходимость объявления переменных. Так как программный код на языке Python не стеснен рамками типов данных, он автоматически может обрабатывать целый диапазон объектов.

*Автоматическое управление памятью*

Python автоматически распределяет память под объекты и освобождает ее ("сборка мусора"), когда объекты становятся ненужными. Большинство объектов могут увеличивать и уменьшать занимаемый объем памяти по мере необходимости.

*Модульное программирование*

Для создания крупных систем Python предоставляет такие возможности, как модули, классы и исключения. Они позволяют разбить систему на составляющие, применять ООП для создания программного кода многократного пользования и элегантно обрабатывать возникающие события и ошибки.

*Встроенные типы объектов*

Python предоставляет наиболее типичные структуры данных, такие как списки, словари и строки, в виде особенностей, присущих самому языку программирования. Эти типы отличаются высокой гибкостью и удобством. Например, встроенные объекты могут расширяться и сжиматься по мере необходимости, могут комбинироваться друг с другом для представления данных со сложной структурой.

*Встроенные инструменты*

Для работы со всеми этими типами объектов в составе Python имеются мощные и стандартные средства, включая такие операции, как конкатенация (объединение коллекций), получение срезов (извлечение части коллекции), сортировка, отображение и многое другое.

*Библиотеки утилит*

Для выполнения более узких задач в состав Python также входит большая коллекция библиотечных инструментов, которые поддерживают практически все, что только может потребоваться, – от поиска с использованием регулярных выражений до работы в сети. Библиотечные инструменты языка Python – это то место, где выполняется большая часть операций.

*Утилиты сторонних разработчиков*

Python – это открытый программный продукт и поэтому разработчики могут создавать свои предварительно скомпилированные инструменты поддержки задач, решить которые внутренними средствами невозможно.

4.4 Прикладное применение Python

Python используется не только отдельными пользователями, он также применяется компаниями для создания продуктов, приносящих настоящую прибыль. Например:

− Компания Google широко использует Python в своей поисковой системе и оплачивает труд создателя Python.

− Служба коллективного использования видеоматериалов YouTube в значительной степени реализована на языке Python.

− Популярная программа BitTorrent для обмена файлами в пиринговых сетях (peer-to-peer) написана на языке Python.

− Популярный веб-фреймворк App Engine от компании Google использует Python в качестве прикладного языка программирования.

− Такие компании, как EVE Online и Massively Multiplayer Online Game (MMOG), широко используют Python в своих разработках.

− Мощная система трехмерного моделирования и создания мультипликации Maya поддерживает интерфейс для управления из сценариев на языке Python.

− Такие компании, как Intel, Cisco, Hewlett-Packard, Seagate, Qualcomm и IBM, используют Python для тестирования аппаратного обеспечения.

Такие компании, как Industrial Light & Magic, Pixar и другие, используют Python в производстве анимационных фильмов.

− NASA, Los Alamos, Fermilab, JPL и другие используют Python для научных вычислений.

− iRobot использует Python в разработке коммерческих роботизированных устройств.

− ESRI использует Python в качестве инструмента настройки своих популярных геоинформационных программных продуктов под нужды конечного пользователя.

− NSA использует Python для шифрования и анализа разведданных.

− В реализации почтового сервера IronProt используется более 1 миллиона строк программного кода на языке Python.

− Проект "ноутбук каждому ребенку" (One Laptop Per Child, OLPC) строит свой пользовательский интерфейс и модель функционирования на языке Python.

Универсальная природа языка обеспечивает возможность его применения в самых разных областях. Фактически с определенной долей уверенности можно утверждать, что Python так или иначе используется практически каждой достаточно крупной организацией, занимающейся разработкой программного обеспечения, – как для решения краткосрочных тактических задач, так и для разработки долгосрочных стратегических проектов.

4.5 Преимущество Python перед другими языками высокого уровня

* Имеет более широкие возможности, чем Tcl. Язык Python поддерживает "программирование в целом", что делает его применимым для разработки крупных систем.
* Имеет более четкий синтаксис и более простую архитектуру, чем Perl, что делает программный код более удобочитаемым, простым в сопровождении и снижает вероятность появления ошибок.
* Проще и удобнее, чем Java. Python – это язык сценариев, а Java унаследовала сложный синтаксис от таких языков программирования, как C++.
* Проще и удобнее, чем C++, но нередко он не может конкурировать с C++, поскольку, будучи языком сценариев, Python предназначен для решения другого круга задач.
* Более мощный и более переносимый, чем Visual Basic. Открытая природа Python также означает, что нет какой-то отдельной компании, которая его контролирует.
* Более удобочитаемый и более универсальный, чем PHP. Иногда Python используется для создания веб-сайтов, но он способен решать гораздо более широкий круг задач, от управления роботами до создания анимационных фильмов.
* Более зрелый и имеет более ясный синтаксис, чем Ruby. В отличие от Ruby и Java, объектно-ориентированный стиль программирования является необязательным в Python – он не вынуждает использовать ООП в проектах, где этот стиль неприменим.

Обладает динамическими особенностями таких языков, как SmallTalk и Lisp, но имеет более простой и традиционный синтаксис, доступный как для разработчиков, так и для конечных пользователей настраиваемых систем.

4.6 Реализации Python

CPython является основной, но не единственной реализацией языка программирования Python. Существуют также следующие реализации:

* Jython — реализация Python, использующая JVM в качестве среды исполнения. Позволяет прозрачно использовать Java-библиотеки.
* PyS60 — реализация языка для смартфонов фирмы Nokia на платформе Series 60.
* IronPython — Python для .NET Framework и Mono. Компилирует Python программы в MSIL, таким образом предоставляя полную интеграцию с .NET-системой.
* Stackless — также написанная на Си реализация Python. Это не полноценная реализация, а патчи к CPython. Предоставляет расширенные возможности многопоточного программирования и значительно большую глубину рекурсии.
* Python for .NET — ещё одна реализация Python для .NET. В отличие от IronPython эта реализация не компилирует Python код в MSIL, а только предоставляет интерпретатор, написанный на C#. Позволяет использовать .NET-сборки из Python кода.
* PyPy — реализация Python, написанная на Python. Позволяет легко проверять новые возможности. В PyPy кроме стандартного CPython включены возможности Stackless, Psyco. В проект интегрированы возможности анализа Python кода и трансляция в другие языки и байткоды виртуальных машин.
* python-safethread — версия CPython без GIL, что позволяет одновременно исполнять Python потоки на всех доступных процессорах. Внесены также некоторые другие изменения.
* Unladen Swallow — начатый Google проект по разработке высокоэффективного, максимально совместимого с CPython JIT-компилятора, а на базе LLVM. Согласно планам по развитию Python, планировалось перенести исходный код Unladen Swallow в CPython в версии 3.3. Но PEP-3146 был отменён в связи с отсутствием интереса к Unladen Swallow со стороны Google, основного спонсора разработки.
* tinypy — минималистическая версия Python. Часть возможностей CPython не реализована.

Micro Python — эффективная реализация Python 3 для встроенных систем с малым объёмом оперативной памяти.

4.7 Недостатки

*Низкое быстродействие*

Классический Python, как и многие другие интерпретируемые языки, не применяющие, например, JIT-компиляторы, имеют общий недостаток — сравнительно невысокую скорость выполнения программ. Сохранение байт-кода позволяет интерпретатору не тратить лишнее время на перекомпиляцию кода модулей при каждом запуске, в отличие, например, от языка Perl. Кроме того, существует специальная JIT-библиотека psyco, позволяющая ускорить в выполнение программ (однако приводящая к увеличению потребления оперативной памяти). Эффективность psyco сильно зависит от архитектуры программы.

Существуют реализации языка Python, вводящие высокопроизводительные виртуальные машины (ВМ) в качестве бэк-энда компилятора. Примерами таких реализаций может служить PyPy, базирующийся на LLVM; более ранней инициативой является проект Parrot. Ожидается, что использование ВМ типа LLVM приведёт к тем же результатам, что и использование аналогичных подходов для реализаций языка Java, где низкая вычислительная производительность в основном преодолена.

Множество программ/библиотек для интеграции с другими языками программирования предоставляют возможность использовать другой язык для написания критических участков.

*Невозможность модификации встроенных классов*

По сравнению с Ruby и некоторыми другими языками, в Python отсутствует возможность модифицировать встроенные классы, такие, как int, str, float, list и другие, что, однако, позволяет Python потреблять меньше оперативной памяти и быстрее работать. Ещё одной причиной введения такого ограничения является необходимость согласования с модулями расширения. Многие модули (в целях оптимизации быстродействия) преобразуют Python-объекты элементарных типов к соответствующим Си-типам вместо манипуляций с ними посредством Си-API. Также это избавляет от многих потенциальных ошибок при неконтролируемом динамическом переопределении встроенных типов

5 Телеграм(мессенджер)

Мессенджер – (IM – Instant Messenger) – это программа мобильное приложение или веб сервис для мгновенного обмена сообщениями. Чаще всего под мессенджером понимают программу, в которой вы пишите сообщения и где вы их читаете. Однако, за каждой такой программой входит понятие «мессенджер».

Telegram – бесплатный кроссплатформенный мессенджер для смартфонов и других устройств, позволяющий обмениваться текстовыми сообщениями и медиафайлами различных форматов. Используются проприетарная сервераня часть с закрытым кодом, работающая на мощностях нескольких компаниях США и Германии, финансируемых Павлом Дуровым в объёме порядка 12 млн долларов США ежегодно, и несколько клиентов с открытым кодом, в том числе под GNU GPL. Количество активных пользователей сервиса на февраль 2016 года составляло более 100 млн человек, а количество ежедневно пересылаемых сообщений достигло 10 миллиардов на август 2015.

5.1 История создания телеграмма

Проект создан Павлом Дуровым, основателем социальной сети «ВКонтакте». В интервью The New York Times Павел рассказал, что первоначальная идея приложения пришла ему ещё в 2011-м, когда к его двери приходили спецназовцы. Когда последние всё-таки ушли, Дуров сразу же написал своему брату Николаю. Тогда же он и осознал, что у него нет безопасного способа коммуникации с братом. Сервис построен на технологии шифрования переписки MTProto, разработанной братом Павла — Николаем. Сам «Телеграм» изначально был экспериментом принадлежащей Павлу компании Digital Fortress с целью протестировать MTProto на больших нагрузках.

14 августа 2013 года представлен первый клиент Telegram для устройств на платформе iOS.

22 августа 2013 года один из участников конкурса Durov’s Android Challenge написал и выложил в открытый доступ первое приложение для операционной системы Android, совместимое с Telegram, (использует тот же протокол MTProto).

В октябре у проекта открылся веб-сайт и была представлена официальная версия Telegram под Android с открытым исходным кодом (GPL2). Предыдущая версия программы доступна под названием «Unofficial Telegram S».

7 ноября 2013 года появились сторонние клиенты сервиса для Windows и OS X с ограниченным функционалом. Также был разработан концепт веб-версии клиента.

В ноябре у программы насчитывалось, по данным «TJournal», около 1 миллиона установок.

В январе 2014 года вышла неофициальная веб-версия Webogram от бывшего разработчика ВКонтакте Игоря Жукова.

21 июля 2014 года в App Store появилось приложение Telegram HD для iPhone и iPad, которое загрузила компания Telegram Messenger LLP.

Новое приложение получило специальную версию для Apple iPad, улучшило поддержку видео и фотографий высокого разрешения, добавило возможность пересылки анимированных изображений в формате gif. На официальном сайте мессенджера в качестве клиента для iOS указано именно это приложение.

15 октября 2014 года в Telegram была добавлена поддержка псевдонимов, по которым возможно связываться с пользователями, даже не зная их телефонного номера, а также запущен веб-клиент.

2 января 2015 года в Telegram была добавлена поддержка стикеров. Изначально в приложении 14 стикеров, но любой пользователь может модифицировать их или добавить свои собственные. В отличие от многих приложений, в Telegram стикеры полностью бесплатные.

В феврале 2016 года один из создателей Telegram Павел Дуров заявил, что мессенджером пользуются уже более 100 миллионов человек, при этом сервис доставляет около 15 миллиардов сообщений ежедневно. Еще в сентябре 2015 года Telegram передавал 12 миллиардов посланий в день.

В апреле 2016 года стало известно, что в мае 2015 года корпорация Google рассматривала возможность покупки мессенджера за более чем 1 млрд долларов США.

В мае 2016 года появилась возможность редактирования отправленных сообщений. Внести изменения возможно в течение двух суток с момента отправки. В этом случае в сообщении появится специальная метка.

22 ноября 2016 года разработчиками был запущен проект Telegraph — блог-платформа, бесплатный издательский инструмент, который позволяет создавать публикации, обзоры, вставлять фотографии и все виды embedded-кода. Telegraph — гибрид блог-платформы, мессенджера и платишера (подобного Medium), с концепцией анонимных имиджборд. Проект Павла Дурова, основателя социальной сети «ВКонтакте»

3 января 2017 года один из разработчиков добавил возможность удаления своих отправленных сообщений. После удаления отправителем сообщения, собеседник не сможет увидеть удаленное сообщение.

В марте 2017 года В. Д. Соловей со ссылкой на анонимный источник сообщил, что российские спецслужбы получили доступ к сообщениям пользователей и их архиву за три года. Павел Дуров назвал это заявление уткой.

15 мая 2017 года стало известно о том, что декстопная версия Telegram получила возможность совершать звонки.

16 мая 2017 года администрация Telegram заявила, что не будет предоставлять информацию российским госорганам.

5.2Технология

Для мессенджера был создан протокол MTProto, предполагающий использование нескольких протоколов шифрования. При авторизации и аутентификации используются алгоритмы RSA-2048, DH-2048 для шифрования, при передаче сообщений протокола в сеть они шифруются AES с ключом, известным клиенту и серверу. Также применяются криптографические хеш-алгоритмы SHA-1 и MD5.

Безопасность от перехвата пересылаемых сообщений со стороны сервера Telegram обеспечивается лишь в режиме «секретных» чатов (Secret Chats), доступном с 8 октября 2013 года. Этот режим реализует шифрование, при котором лишь отправитель и получатель обладают общим ключом (end-to-end шифрование), с применением алгоритма AES-256 в режиме IGE (англ. Infinite Garble Extension) для пересылаемых сообщений. В отличие от обычного режима, сообщения в секретных чатах не расшифровываются сервером, история переписки сохраняется лишь на тех двух устройствах, на которых был создан чат.

При обмене файлами можно как отправить файлы с устройства, так и искать медиаконтент в интернете, в том случае, если используется мобильная версия для iOS или Android. Размер передаваемых файлов ограничен 1,5 Гб. Программа использует систему докачки файлов после обрыва связи.

Имеется возможность организовывать мультичаты до 200 участников, начиная с ноября 2015 года, супергруппы до 1000 участников, с 14 марта 2016 — супергруппы до 5000 участников.

5.3 Основные возможности приложения Telegram

* Программа поддерживает большую часть распространенных операционных систем, такие как Windows, Linux, Mac OS;
* доступна для мобильных устройств, работающих на Android, Windows Phone и iOS;
* автоматическая синхронизация программы между всеми устройствами пользователя;
* применение специальной системы шифрования для передачи и хранения личной информации;
* самостоятельная очистка всей истории переписки по истечению выбранного промежутка времени;
* возможность сохранять мультимедийные объекты в «облаке»;
* открытый доступ к API для создания собственных интегрируемых модулей.

Начало работы с месседжером Телеграмм подразумевает регистрацию мобильного аккаунта в приложении. Единственное обязательное условие для авторизации – указание номера мобильного телефона.

5.4 Преимущества и недостатки использования Telegram

В пользу использования этой программы можно отметить следующие характеристики:

* бесплатное пользование приложением на неограниченный срок;
* высококачественное шифрование информации;
* мгновенная доставка пересланных данных;
* отсутствие рекламных блоков и микро-транзакций;
* возможность самостоятельной настройки и регулирования приложения «под себя».

Скромные, но тем не менее, недостатки:

* отсутствие интерфейса с переводом на русский язык;
* исходный код серверной части закрыт для пользователей.

Несмотря на это, Телеграмм остается одним из самых удобных, а главное безопасных способов передачи информации в сети. Ведь разработчики приложения доказали, что обеспечить конфиденциальность в интернете возможно.

5.5 Секретный чат

Мессенджер Telegram обладает некоторыми отличительными положительными чертами, относящимися к безопасности пользования данной программой. Одной из основных черт считается секретный чат в Telegram.

Если подробнее ознакомиться с этой функцией, то можно увидеть ряд особенностей её применения. Переписка не подвергается сохранению на сервере и моментально удаляется сразу после отправления сообщения. Остальные пользователи, естественно, не могут прочитать её. А для повышения мощности защиты реализуется конечное шифрование сообщения. Таким образом, отправленное сообщение остается доступным лишь для его адресата и адресанта. Применяемые системой алгоритмы гарантируют высокий уровень безопасности, а расшифровать чужую личную переписку не могут даже сами разработчики мессенджера.

Для каждого сообщения или отдельной переписки можно подобрать и установить отдельный таймер самоликвидации, который составляет минимум две секунды и максимум семь дней. Как только срок завершается, сообщение самостоятельно удаляется.

Получатель читает новое сообщение и через несколько секунд оно попросту исчезает. При проявлении смекалки, конечно, можно успеть сделать снимок экрана или скриншот. Но в таком случае программа зафиксирует скриншот и перешлет отправителю уведомление об этой операции.

ТУТ ДОЛЖНА БЫТЬ КАРТИНКА ТЕЛЕГРАММА ЛИБО РЕКЛАМКА)

<https://texterra.ru/blog/pochemu-stoit-poprobovat-telegram-obzor-messendzhera.html> про телеграм топчик просто глянуть как будет сказка может быть,

<http://telegram-online.ru/sekretnyj-chat-v-telegram.html> ссылка про секретный чатик телеграммма

6 Процесс создания бота для телеграмм

6.1 Подготовка перед написание

Непосредственно вот мы и подошли к написанию нашего бота, для начала откроем командную строку от имени администратора, для того чтобы мы могли вносить изменения и установим библиотеку *python-telegram-bot*

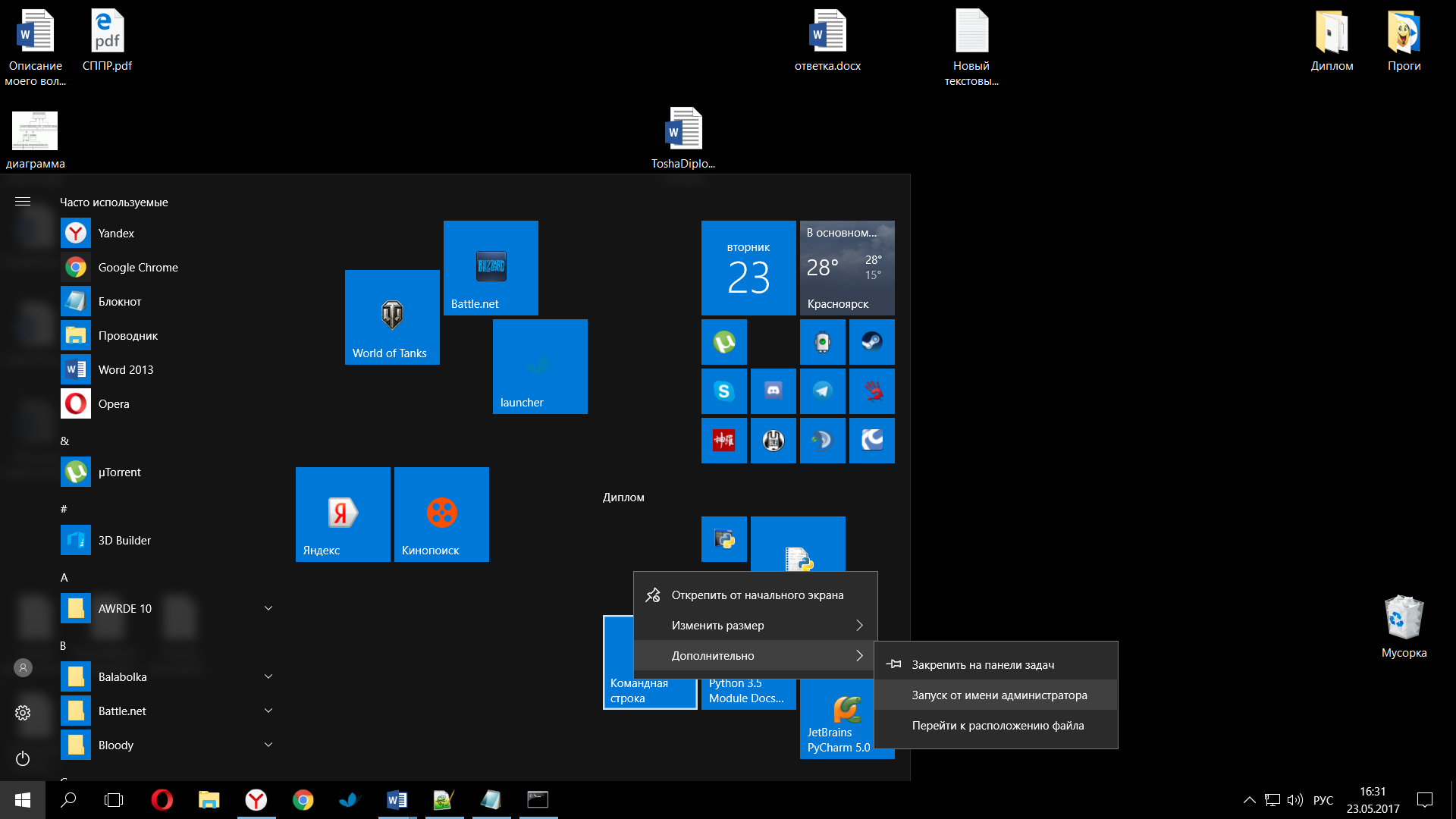


Рисунок 6.1.1 – Запуск командной строки от имени администратора

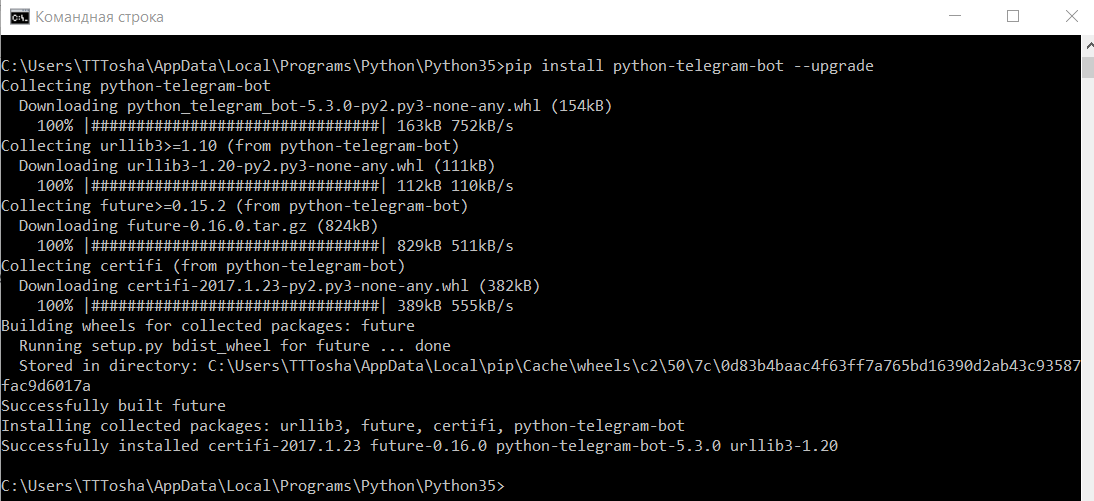


Рисунок 6.1.2 – Установки библиотеки *python-telegram-bot*

И т.д. опишу дальше как всё писал если так можно либо лучше кодом

6.2 Регистрация

Перед начало разработки бота, необходимо установить приложение Telegram. Далее необходимо зарегистрировать нашего бота и получить его уникальный id, являющийся одновременно и токеном. Для этого в телеграмме существует специально бот - @*BotFather*. Находим его и нажимаем единственно доступную на данный момент команду /*start*, после чего нам открывается подсказка с командами, нам нужно создать нового ботам следовательно пишем команду /*newbot*, которая отвечает за создание нового бота.

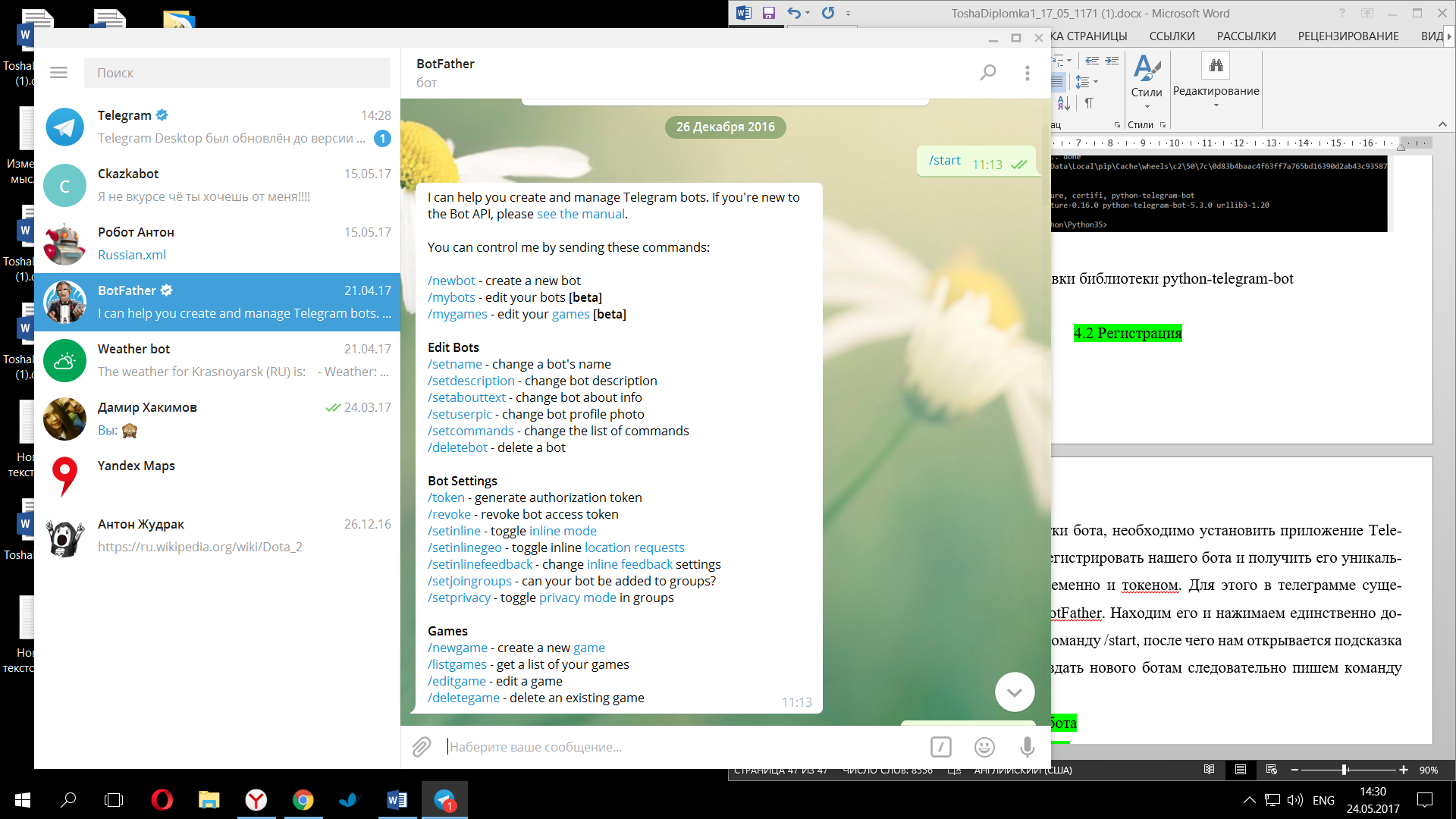


Рисунок 6.2.1– начало работы с @BotFather

Далее @BotFather предлагает нам придумать имя (имя бота должно обязательно оканчиваться на “bot” сделано это для того что бы можно было отличить живых людей от ботов)

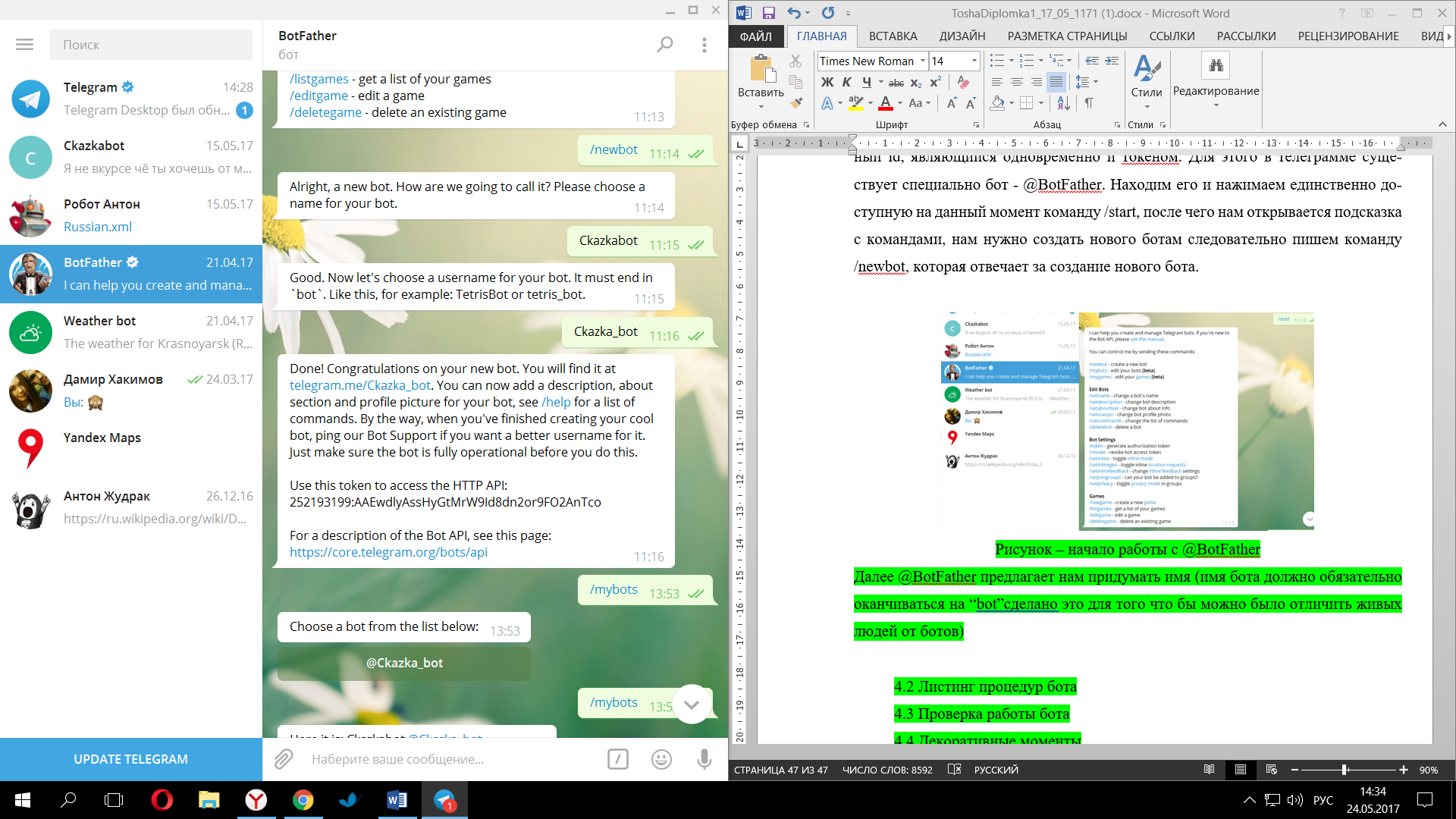


Рисунок 6.2.2 – Создание имени бота и присвоение токена

После создания имени @BotFather автоматически присвоит токен нашему боту, зная токен любой человек сможет управлять ботом поэтому при создании бота токен нужно хранить в отдельном файле чтобы злоумышленники не воспользовались им в корыстных целях.

4.2 Листинг процедур бота

4.3 Проверка работы бота

4.4 Декоративные моменты

Источники

1 <https://ru.wikipedia.org/wiki/Бот_(программа)> Википедия типо

2 <https://vc.ru/p/ym-bot-hackaton> Аудитория ботв зачем они нужны

3 <http://www.abc.org.ru/it_spec.html#4>

Патч 25.05 изменения

1. Добавлен пункт 1 "Человеко машинное взаимодействие"

с пункта 1.1 до 1.4 в основном всё своими словами

пункт 1.5 и пункт 1.6 ctrl+c --> ctrl+v

2.Добавлен пункт 2 Системы поддержки принятий решений (заново написали)

2.1 История развития (оставили ничего не меняли пока что)

3. Добавлен пунк 3 Публичные API (всё своими словамиб даже запрос свой)

4.Пункты с питоном пока оставил т.к. диплом по госту должен быть 80-100 страниц как мне сказали, в любой момент можно убрать пункты

4.2 История создания питон (-1,5 страницы)

4.2.1 Влияние других языков (-1 страница)

4.3 Функциональные возможности (-2 страницы)

4.4 Прикладное применение Питон (-1,5 страницы)

4.6 Реализации Питона (-1 страница)

минус 7-8 страниц будет)

5. Пункт про секретный чат сокращен до 0,5 страницы

6. Добавлен незавершенный пункт 6. Процесс создания бота (доделаю, пойдёт ли по оформлению или лучше код писать курсивом и так подробно не расписывать)

Что можно добавить:

В пункт 6 я бы добавил процесс установки самого пакета питон с картинками (2 страницы)

https://pythonworld.ru/osnovy/skachat-python.html

Не реализованно:

как весь процесс происходит отправки ключа и ответа бота пользователю

Про базу данных какую использую, в данным момент это просто блокнотик)

И описание по факту самого бота, его листинг и весь процесс, что по факту самое основное.Как то так

Создание модели СППР, создание диаграммы классов т.е. какие классы писать там чтото наподобие (бот(айди,токен), пользователь(айди,имя)

Отчёт https://www.antiplagiat.ru/: 49% антиплагиата, и всё в основном по питону