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](#_Toc485010980)

[1 Человеко-машинное взаимодействие 8](#_Toc485010981)

[1.1что то типо введения как то надо назвать сказку эту 8](#_Toc485010982)

[1.2 История развития человеко-компьютерного взаимодействия 9](#_Toc485010983)

[1.3 Кризис email-маркетинга 14](#_Toc485010984)

[1.4 Рождение чат ботов 15](#_Toc485010985)

[1.5 Боты в мессенджерах 18](#_Toc485010986)

[1.6 Кому и для чего нужны боты 19](#_Toc485010987)

[1.7 Чем бот лучше реального пользователя 20](#_Toc485010988)

[1.8 Самые популярные боты 20](#_Toc485010989)

[1.9 Почему боты не заменят мобильные приложения 22](#_Toc485010990)

[1.10 Тенденции развития чат ботов в 2017 году 23](#_Toc485010991)

[2 Системы поддержки принятия решений 25](#_Toc485010992)

[2.1 История развития систем поддержки принятия решений 25](#_Toc485010993)

[2.2 Описание и применение систем поддержки принятия решений 27](#_Toc485010994)

[3 Публичные API 33](#_Toc485010995)

[3.1 API один из новых каналов взаимодействия 33](#_Toc485010996)

[3.2 История API 36](#_Toc485010997)

[3.3 Применение публичных API 36](#_Toc485010998)

[4 Выбор среды для разработки 40](#_Toc485010999)

[4.1Причины использования Python 41](#_Toc485011000)

[4.1.1 Качество программного обеспечения 41](#_Toc485011001)

[4.1.2 Высокая скорость разработки 41](#_Toc485011002)

[4.1.3 Переносимость программ 42](#_Toc485011003)

[4.1.4 Библиотеки поддержки 42](#_Toc485011004)

[4.1.5 Интеграция компонентов 42](#_Toc485011005)

[4.1.6 Влияние других языков 43](#_Toc485011006)

[4.2 История создания 44](#_Toc485011007)

[4.3 Функциональные возможности 44](#_Toc485011008)

[4.4 Прикладное применение Python 47](#_Toc485011009)

[4.5 Преимущество Python перед другими языками высокого уровня 48](#_Toc485011010)

[4.6 Реализации Python 49](#_Toc485011011)

[4.7 Недостатки 50](#_Toc485011012)

[5 Телеграм(мессенджер) 52](#_Toc485011013)

[5.1 История создания телеграмма 52](#_Toc485011014)

[5.2Технология 54](#_Toc485011015)

[5.3 Основные возможности приложения Telegram 55](#_Toc485011016)

[5.4 Преимущества и недостатки использования Telegram 56](#_Toc485011017)

[5.5 Секретный чат 56](#_Toc485011018)

[5.6 Аудитория телеграмм 57](#_Toc485011019)

[6. Выбор Субд 62](#_Toc485011020)

[6.1 История PostgresQL 62](#_Toc485011021)

[6.2 PostgreSQL 65](#_Toc485011022)

[6.3 Сравнение PostgreSQL c другими СУБД 67](#_Toc485011023)

[7 Процесс создания бота для телеграмм 72](#_Toc485011024)

[7.1 Регистрация 72](#_Toc485011025)

[7.2 Подготовка перед написание 73](#_Toc485011026)

[7.3 Листинг процедур бота 75](#_Toc485011027)

[4.3 Проверка работы бота 77](#_Toc485011028)

[4.152 Подключение к базе Дамиркиной 77](#_Toc485011029)

[4.4 Декоративные моменты 77](#_Toc485011030)

[Добавим различные эмодзи к нашим подписям в командах и оформим красиво приветствие бота, добавим подсказки к командам 77](#_Toc485011031)

[Добавление красивой иконки бота и изменение его описание 77](#_Toc485011032)

[Источники 77](#_Toc485011033)

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.** В век информатизации, гаджеты вошли в нашу жизнь и остаются очень нужными и удобными вещами, смартфоны приобретают всё более мощные характеристики и используются как замена в иной раз компьютеров. Развиваются мобильные операционные системы и для каждой создаются мобильные приложения, для различного рода деятельности или развлечения. одно из таких приложений телеграмм. Это приложение представляет собой оболочку для общения клиента с клиентом по защищённому соединению, далее оно приобрело ещё одну функцию и такую как бот информер, с помощью таких ботов можно упростить некоторые рутинные функции в ежедневном быту.

**Цель исследования.** Создание системы поддержки принятия решений с интерфейсом в виде бота для мессенджера telegram.

**Объект исследования.** СППР, бот в мессенджере telegram.

**Предмет исследования**. Применение СППР к созданию бота информера для мессенджера telegram.

**Методы исследования.** Анализ литературы и материалов, моделирование СППР.

**Научная новизна исследования:** Системы поддержки принятия решений приобретают значительную популярность в своём применении к организации и выбору необходимых решений в трудных ситуация. С постоянным развитием информационных систем, применение СППР становится всё более популярным, разместить такого рода систему возможно и на смартфонах, так как рост программных и аппаратных мощностей на 2017 год позволяет это сделать.

Научная новизна: новый интерфейс человеко-машинного взаимодействия и применение СППР в бытовых целях, т.е. новая область и новая реализация.

Задачи.

Не окончательный вариант, необходима доработка.

1 Человеко-машинное взаимодействие

1.1 Электронно-вычислительные машины

Электронно-вычислительные машины заняли место рядом с человеком с середины двадцатого века, со временем взаимодействие человека и компьютера только растёт. Практически не осталось сферы человеческой деятельности, где не используются какие-либо вычислительные системы. И если раньше на этапе развития электронно-вычислительной техники общаться с компьютера могли только «посвященные», человек был вынужден использовать язык понятные машине, то в современной действительности, вычислительные устройства становятся всё более дружелюбны и для взаимодействия с компьютерными системами человеку уже не требуется серьёзной подготовки, разумеется если это взаимодействие не предполагает решения сложной профессиональных задач.

Поэтому вполне логично стремление разработчиков сделать взаимодействие с вычислительными устройствами, как можно более естественным и простым для человека. Развитие этого направления сдерживалось недостаточный технологическим уровнем. Однако в последнее время в деле гуманизации пользовательских интерфейсов видны серьёзные продвижения. Появление сенсорных экранов, функций касания (multitouch) позволяют уже взаимодействовать с компьютерной системой на интуитивном уровне, это подтверждается быстрым освоением подобных устройств детьми. Следовательно логично дальше ожидать от компьютеров способность понимать словесные команды, реагировать на мимику и жесты людей, анализировать поведение человека. Эти ожидания не напрасны, современное развитие технологий обработки речи, видео и изображений, выводит взаимодействие человека с компьютерной системой на новый этап.

Появляются идеи естественно-интуитивного взаимодействия человека с компьютерной системой, предполагающей способность вычислительных устройств понимать человека через словесные команды и жесты, какой либо физический контакт. Разработчикам программного обеспечения использования новых идей позволит создавать приложения, вносящие ощущения присутствия и погружения. Такого эффекта можно добиться реализацией возможностей отслеживания жестов рук и пальце, анализа мимики лица, распознавания речи, отслеживания перемещения 2D/3D объектов. Умение разрабатывать такие приложения конечно же будет востребовано в будущем.

1.2 История развития человеко-компьютерного взаимодействия

Идеи самих цифровых вычислительных машин получили своё развитие ещё в ⅩⅧ-ⅩⅨ веках, а компьютерные технологии стали доступны в середине ⅩⅩ века. Самые первые электронно-вычислительне машины были ориентированы, в первую очередь, на выполнение трудоёмких расчётов и на взаимодействие со специалистами. Например для первой электронной машины ENIAC (выпущенно й в 1943 году) ввод программы выполнялся с помошью различных переключателей и гибких кабелей со штекерами, вставляемых в нужные разъёмы, в соответствии с рисунком 1.

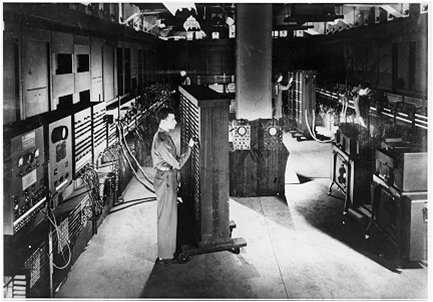


Рисунок 1 – Взаимодействие человека и ЭВМ ENIAC

Например в ЭВМ Mark 1 (1945 г.) уже использовались перфоленты с нанесённой на них программой в соответствии с рисунком 2, что немного облегчило труд оператором. Перфоленты и перфокарты долго оставались носителями информации, которые использовались для хранения программ и ввода их в ЭВМ для исполнения.

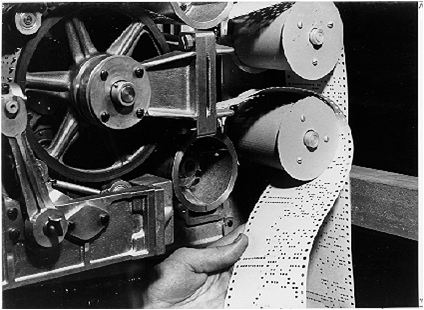


Рисунок 2 - устройство для чтения перфолент ЭВМ Mark 1

В 1960 году Джордж Ликлайдер выдвинул идею «симбиоза человека и компьютера» - объединения человеческого интеллекта и вычислительной техники для управления информацией. Он предположил промежуточные цели, достижение которых предпологает реализацию данной идеи.

Ближайшие цели:

- разделение времени компьютера между пользователями;

- крупномасштабные системы хранения и поиска информации;

- интерактивные системы реального времени для обработки информации и программирования;

- электронный ввод/вывод символьной информации и графической информации.

Среднесрочные цели:

- координации объединения разработчиков для проектирования и программирования больших систем;

- способность ЭВМ распознавать речь оператора;

- способность ЭВМ распознавать рукописные тексты;

- возможность использования светового пера, в качестве устройства ввода координат и указки (световое перо – светочувствительное устройство, позволяющее выбрать точку экрана дисплея, указывая на нее).

Долгосрочные цели:

- понимание ЭВМ естественного языка;

- способность ЭВМ распознавать речь произвольного пользователя;

- эвристическое программирование, т.е. "интеллектуализация" работы программы путем придания ей большей гибкости и эвристичности "мышления".

Развитие вычислительной техники, во многом, пошло по пути достижения целей, поставленных Ликлайдером, уже в середине 60-х годов появились вычислительные машины, поддерживающие большое количество пользователей, каждый из которых получал в свое распоряжение выделенный интерфейс к системе (терминал) и мог работать в интерактивном режиме.

В 1963 году Айвен Сазерленд (Ivan Sutherland) разработал SketchPad – графический комплекс, прообраз будущих САПР, оказавший огромное влияние на формирование базовых принципов графических пользовательских интерфейсов. Основные идеи: использование объектно-ориентированной модели, любой нарисованный элемент представлялся n-компонентной структурой, его можно было копировать, перемещать, поворачивать или масштабировать, сохраняя основные свойства. Впервые были реализованы алгоритм прорисовки окон и алгоритм обрезки.

Еще одно важное имя в истории развития вычислительной техники: Дуглас Энгельбарт (Douglas C. Engelbart). В середине 60-х годов командой Дугласа Энгельбарта разработана среда NLS (oN-LineSystem), включающая в себя принципиально новую операционную систему, универсальный язык программирования, электронную почту, разделенные экраны телеконференций, систему контекстной помощи. Представлен прототип т.н. WIMP-интерфейса, т. е. интерфейса, использующего понятия окон (windows), пиктограмм (icons), меню (menus) и указателей (pointers), являющихся ключевыми и для сегодняшних графических пользовательских программ и сред. Среда широкого распространения не получила, но как побочный эффект проекта NLS был изобретен первый манипулятор типа мышь, без которого сложно представить любое взаимодействие с компьютером в современной действительности. К оконной среде NLS существующие манипуляторы (джойстики, световые перья и прочие) категорически не подходили, в этой области было проведено целое исследование, итогом которого и стала мышка в соответствии с рисунком 3 показано изображение первой мыши.



Рисунок 3 – Первая мышь 1964 год

В конце 60-х годов XX века, технологический уровень позволил задумываться о создании персонального компьютера. В 1969 году американский математик Алан Кей защитил докторскую диссертацию, в которой разработал принципы создания персонального компьютера. С 1971 года Алан Кей занимался теоретической разработкой прототипа персонального компьютера, названного им Dynabook, в исследовательском центре фирмы Xerox в Пало-Альто (Palo Alto Research Center, PARC). Этот компьютер, не превышающий размер блокнота, должен был обладать возможностями для обработки текстов и графической информации, а также служить средством связи с удаленными базами данных. Кроме того, этот компьютер должен был быть недорогим и доступным широкому кругу покупателей.

Концепция Dynabook описывала то, что сейчас известно как ноутбук, или планшетный ПК. Для Dynabook был спроектирован и смоделирован графический интерфейс Star GUI, одним из основных принципов управления новым компьютером должен был стать не ввод команд с клавиатуры, а выбор их с помощью "мыши" из предлагаемого меню. Графический интерфейс Star стал прототипом интерфейса Macintosh.

Несмотря на то, что графический интерфейс был описан еще в начале 70-х годов, а идеи появились еще раньше, в реальности взаимодействие пользователя с ЭВМ обеспечивалось за счет, так называемого, интерфейса командной строки (CLI, Command Line Interface). В процессе взаимодействия человек вводил команды, а компьютер реагировал соответствующим образом, разумеется, ни о каком дружественном интерфейсе речи не шло. Пользователь должен был точно знать, какая команда приведет к выполнению нужных ему действий и правильно ввести ее в командную строку.

К концу 70-х годов не только стало понятно, что при создании персональных компьютеров необходимо учитывать удобство пользователей, но и накопились технологии, позволяющие реализовать, так называемое, эргономическое проектирование вычислительной техники. Стали появляться персональные компьютеры с графическим интерфейсом, спроектированные с учетом удобства пользователя. В связи с этим назрела необходимость изучения человеко-компьютерного взаимодействия в университетах при подготовке специалистов в области компьютерных наук.

Человеко-компьютерное взаимодействие (HCI, Human-Computer Interaction) – это дисциплина, имеющая дело с проектированием, оцениванием и реализацией интерактивных вычислительных систем для использования человеком, а также с изучением основных явлений, связанных с этими вопросами.

Такое определение было сформулировано в отчете группы, ответственной за разработку рекомендаций к образовательной программе в области человеко-компьютерного взаимодействия (август, 1988). Группа была сформирована из членов ассоциации по вычислительной технике (ACM, Association for Computing Machinery). ACM и IEEE Computer Society, крупнейшие научно-профессиональные сообщества специалистов по вычислительной технике, играют ключевую роль в разработке образовательных программ в области компьютерных наук. После этого отчета модуль HCI (человеко-компьютерное взаимодействие) включается как обязательная часть в курс компьютерные науки.Эволюцию устройств ввода/вывода можно представить таблицей 1.

Таблица 1 – Эволюция устройств ввода/вывода

|  |  |  |
| --- | --- | --- |
| Время | Устройство ввода | Устройство вывода |
| На заре эры вычислительной техники | Переключатели и гибкие кабели со штекерами, перфолента и перфокарты, клавиатура | Экран дисплея, телетайп, принтеры |
| Недавнее прошлое | Клавиатура, управление курсором, световое перо, джостик, мышь, микрофон | Прокручиваемый «стеклянный телетайп», текстовый терминал, аудио |
| Наши дни | Цифровые перчатки и костюмы, естественный язык, интерактивная камера | Наголовный дисплей, повсеместный вычисления, автономные агенты |
| Ближайшее будущее | ? | ? |

1.3 Кризис email-маркетинга

По меркам интернет-маркетинга, история взаимодействия с аудиторией посредством email-рассылок весьма продолжительна. Проверенный временем инструмент и сегодня пользуется большой популярностью. Проблема состоит в том, что получать адреса электронной почты потенциальных клиентов с каждым днем все сложнее, а click-through rate рассылок постепенно падает в соответствии с рисунком 4. А значит, email-маркетинг становится дороже.

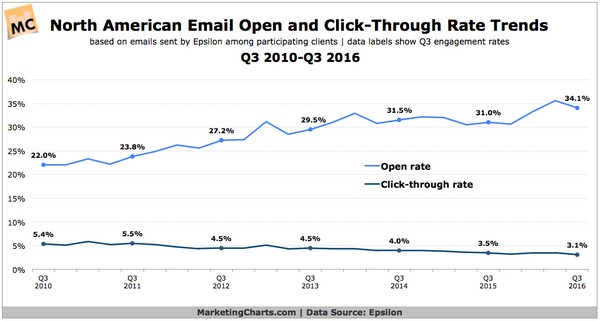


Рисунок 4 - Кризис email-маркетинга

Представленный выше график наглядно демонстрирует отрицательную динамику коэффициента кликабельности. Рост такого показателя, как Open Rate, происходит за счет пользователей мобильных устройств. Политика почтовых сервисов (к примеру, создание Google отдельной вкладки для писем коммерческого характера) также не упрощает маркетологам жизнь.

Безусловно, email-маркетинг все еще является важным коммуникационным каналом. Однако едва ли кто-то станет отрицать, что пик его эффективности пройден.

Когда речь идет об органических каналах — будь то электронная почта, Facebook или ВКонтакте — относительная досягаемость аудитории часто имеет тенденцию к уменьшению. И бороться с этим практически невозможно

1.4 Рождение чат ботов

На фоне кризиса email-рассылок маркетологи ищут новые перспективные инструменты взаимодействия с аудиторией. Один из таких инструментов — чат-боты, которые представляют собой сервисы, способные вести диалог с пользователем по определенному алгоритму (некоторые чат-боты даже наделены искусственным интеллектом).

В последнее время интерес к программируемым собеседникам стабильно растет. Запуск в середине 2016 года платформы Facebook Messenger с чат-ботами спровоцировал дополнительный всплеск интереса к этой тем.

Рождение чат-ботов

Как и искусственный интеллект, чат-боты - типичный вернувшийся тренд. Первую бот-программу написал профессор Массачусетского технологического института Джозеф Вайзенбаум в далеком 1964 году. Его детище по имени Eliza пародировала поведение психотерапевта: выделяла ключевые слова в репликах человека и отвечала уточняющими вопросами, а если варианта ответа не находилось то она говорила «Понятно» и переводила разговор на другую тему.

1972 год порадовал другой эпичной разработкой – Parry. Этот чат-бот мог имитировать речь параноидального шизофреника и часто выдавал себя за пациента. Иногда его описывали, как «Элиза с мнением». Видимо психиатра от шизофреника отличает лишь отсутствие этого самого мнения. В любом случае чат-бот был более умным, чем его предшественница-терапевт, так как мог неплохо обманывать профессиональных психиатров. 48% из них так и не поняли, что они общаются с машиной.

Через 16 лет после Parry учёные попытались создать бота с искусственным интеллектом, и на это у них ушло 9 лет. Программа Jabberwacky способна имитировать человеческий разговор в развлекательной манере. Бот запоминает всё, что ему говорят, а после этого находит нужный ответ с помощью метода контекстных шаблонов. Он быстро учится сленгу, шутит и даже формирует собственный характер. К слову, сегодня робот Джордж помогает российским студентам учить английский язык в проекте LiveEnglish.

Следующей важной вехой в развитии смарт-ботов стала программа A.L.I.C.E. (Artificial Linguistic Internet Computer Entity), разработанная в 1995 году. Кто такая Элис? Это первое интернет-существо, которое смогло обрабатывать естественную речь пользователя. Она использует эвристические шаблоны, полученные от собеседника, и благодаря этому может вести полноценный разговор. Тест Тьюринга, к сожалению, так и не прошла, но была неоднократно признана самым «человечным» ботом среди уже существующих.

2001 год заложил фундамент для всеми любимой Siri. Именно в это время появился бот Smarterchild, который доказал, что людям действительно в кайф общаться с умной машиной, которая имеет свой характер и может дать дельные советы. С ботом болтало более 30 тысяч пользователей AIM и MSN.

С появлением больших данных понадобилась система, которая может обработать, проанализировать и извлечь нужную информацию. В 2006 году IBM принялась за разработку Watson. В итоге получился чат-бот, способный обрабатывать естественный язык и обучаться в процессе общения. Он может мгновенно ответить на любой вопрос, например, как открыть счёт в банке или что подарить маме на день рождения.

Google Now возник через два года после Siri. Разработчики создали его для мобильного поисковика. Бот от Google может отвечать на вопросы, давать дельные советы и обрабатывать запросы пользователя.

В 2015-м появились знаменитые Alexa от Amazon и Cortana от Microsoft. И в принципе уже стало ясно, что мы вступили в эпоху чат-ботов. Умные программы научились без проблем распознавать человеческую речь, обучаться, отвечать на всевозможные вопросы, реагировать на голосовые команды, заказывать товары на дом и многое другое.

Ну, а в 2016 сеть покорил знаменитый Tay – самообучаемый бот от Microsoft, который должен был перенять манеру общения подростков. Буквально за 16 часов бот из милой девочки превратился в агрессивного параноика-расиста, который ненавидит весь мир. Проект в срочном порядке прикрыли, но мемы о нём разошлись по всей сети.

Также в прошлом году Facebook запустила платформу Messenger, которая позволила разработчикам создавать умных ботов для контакта с пользователями. Примерно за полтора месяца после запуска на платформе было создано 11 тысяч ботов.

Как видно, эволюция чат-ботов охватывает 50 лет. А сейчас они обрели популярность, потому что для них появилась «питательная среда» – мессенджеры, в которых некоторые люди стали проводить больше времени, чем в соцсетях и приложениях. Сегодня мессенджерами пользуется более 2,5 миллиарда пользователей, и данная цифра будет только увеличиваться. Соответственно, интернет вскоре заселится чат-ботами и вполне возможно, что в скором времени мы не сможем отличить, с кем общаемся – с другим пользователем или машиной.

1.5 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.

1.6 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.

1.7 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.

1.8 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 в соответствии с рисунком 5 показан бот, который показывает актуальный курс валют

:

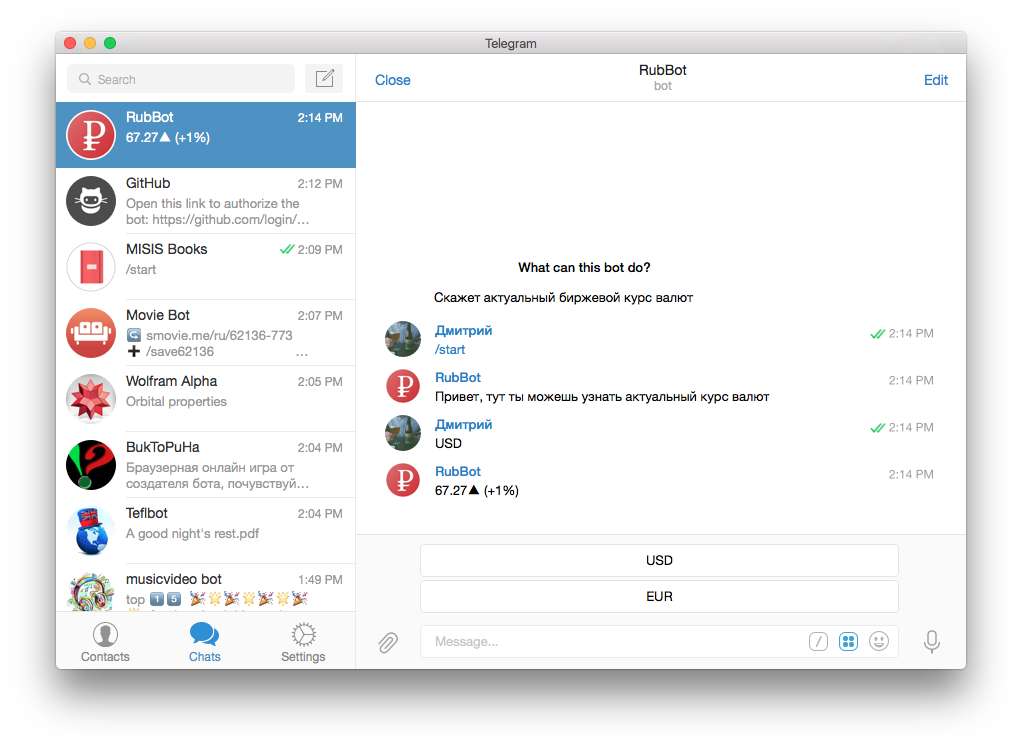


Рисунок 5 – RubBot (@rubbot)

Или например в соответствии с рисунком 6, бот, который соединяется с репозиторием GitHub для получения уведомлений о правках файлов в режиме реального времени



Рисунок 6 – GitHub(@githubbot)

И наконец в соответствии с рисунком 7, бот находит любое изображение по вашему запросу. В опциях можно запросить улучшенное качество или похожие картинки

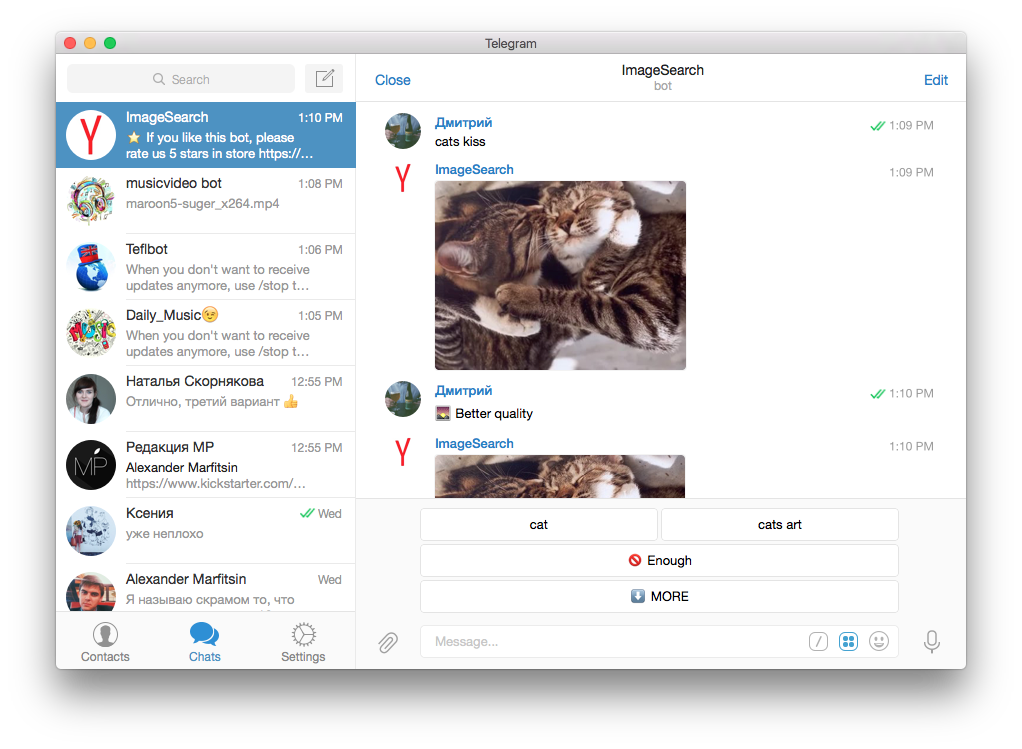


Рисунок 7 – ImageSearch (@imageSearchBot)

1.9 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.

1.10 Тенденции развития чат ботов в 2017 году

1. Переписка только по делу

Пользователи уже устали от бессмысленного общения и огромных потоков информации. Поэтому чат-боты и ценны – они могут предоставить конкретный ответ на заданный вопрос. Боты смогут обрабатывать естественный язык, однако копирование привычного общения станет скорее исключением, а не правилом.

2. Возможность представления веб-страниц в мобильном виде

Интеграция версий страничек для мобильных устройств делает поиск намного удобнее и позволяет чат-боту в полной мере задействовать возможности мобильных платформ. Так, бот может просто показать пользователю веб-страничку с результатами поиска нужного товара, а не генерировать сложное сообщение с избранной информацией.

3. Выполнение важных социальных функций

Функциональные боты могут хорошо выполнять множество важных повседневных задач, а значит, пользователь будет чаще общаться с виртуальным помощником и советовать его родным, друзьям и знакомым. К слову, эксперты ожидают в скором времени появления вирусных ботов.

4. Выход за пределы экосистемы

С развитием отрасли чат-ботов у виртуальных собеседников появятся новые инструменты и функции, которые можно будет использовать за пределами экосистемы.

5. Полезный инструмент для каждой сферы

Универсальных ботов постепенно сменят специализированные программы. Ожидается, что уже в этом году появятся инструменты для ритейла, медиа, сферы обслуживания и т.д.

6. Новые способы взаимодействия с клиентами

Разработчики платформы Messenger реализовали отличное решение – автоматизировали наиболее частые запросы с помощью ботов, а когда те не могли ответить на вопрос пользователя, подключали оператора. С развитием данной отрасли появится намного больше гибридных моделей поддержки клиентов.

7. «Гонка вооружения» ботами

С появлением ботов многие компании стремятся внедрить их первыми для получения конкурентного преимущества. В свою очередь разработчики и маркетологи также исследуют новые возможности чат-ботов, чтобы внедрить уникальные инструменты. Подобная «гонка вооружения» приведёт к появлению интересных и полезных решений

2 Системы поддержки принятия решений

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.

В начале нового тысячелетия была создана СППР на основе Web - сематический Web, которая обеспечивала поиск информации с использованием формализации представления содержания, семантики, смысла искомых объектов, что позволило обеспечить необходимые условия сходимости поискового процесса к желаемому результату, существенно сократить время поиска при «расплывчатом» формулировании запроса и «недетерминированной» постановки целей.

Таким образом, СППР - в большинстве случаев - это интерактивная автоматизированная система, которая помогает менеджеру (ЛПР) использовать данные и модели для идентификации и решения задач и принятия решений. Система должна обладать возможностью работать с интерактивными запросами с достаточно простым для изучения языком запросов.

Дальнейшее развитие СППР происходит по пути усложнения интеллектуальных информационных технологий (ИИТ), позволяющих более глубоко описывать проблемные ситуации. Описание проблемной ситуации опирается как на саму выделенную ситуацию, так и на индивидуальное восприятие ее человеком, т.е. проблемная ситуация описывается внешними и внутренними факторами, пропорция между которыми меняется с изменением ситуации. ИИТ, применяемые для описания проблемной ситуации, как правило, реализуют методы: решения обратных задач, нечетких топологических пространств, эволюционных вычислений, теории катастроф, рефлексивной и когнитивной психологии, нейролингвистического программирования, квантовых вычислений и д

2.2 Описание и применение систем 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.

Системы поддержки принятия решений могли использовать в основном крупные компании, так как разработка подобных систем требует очень огромных ресурсов и времени. С течение некоторого времени эти системы стали доступным и мелким организациям, так как рост информационных технологий не стоит на месте. На сегодняшний день уже абсолютно каждый может использовать и применять подобные системы в любой сфере деятельности для принятия тех или иных решений.

Сейчас системы поддержки принятия решений изменили свой термин на BI-системы. В качестве синонимов систем поддержки принятия решений оперируют понятиями «аналитическая система» и «информационно-аналитическая система» (ИАС). Для обозначения аналитический технологий и средств в целом теперь принято использовать термин "Business Intelligence". Понятие BI объединяет различные средства и технологии анализа и обработки данных масштаба предприятия, на их основе создаются BI-системы. Их цель – повысить качество информации для принятия управленческих решений.

3 Публичные API

3.1 API один из новых каналов взаимодействия

Для того чтобы реализовать систему поддержки принятия решений, необ-ходимо использовать публичные API. Такой метод с применением API позволя-ет получать доступ к различным сервисам и брать информацию для принятия решений из публичного доступа, что значительно облегчает работу при про-граммной реализации человеко-машинного интерфейса.

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.

Стоит отметить, что не все публичные API дают полный доступ к своим сервисам, например, сервис «OpenWeatheMap.com» из которого берётся информация о погоде в реализованной среде человеко-машинного взаимодействия, позволяет делать 60 запросов в минуту. В соответствии с рисунком 8, выделен пункт о предоставлении данной услуги.

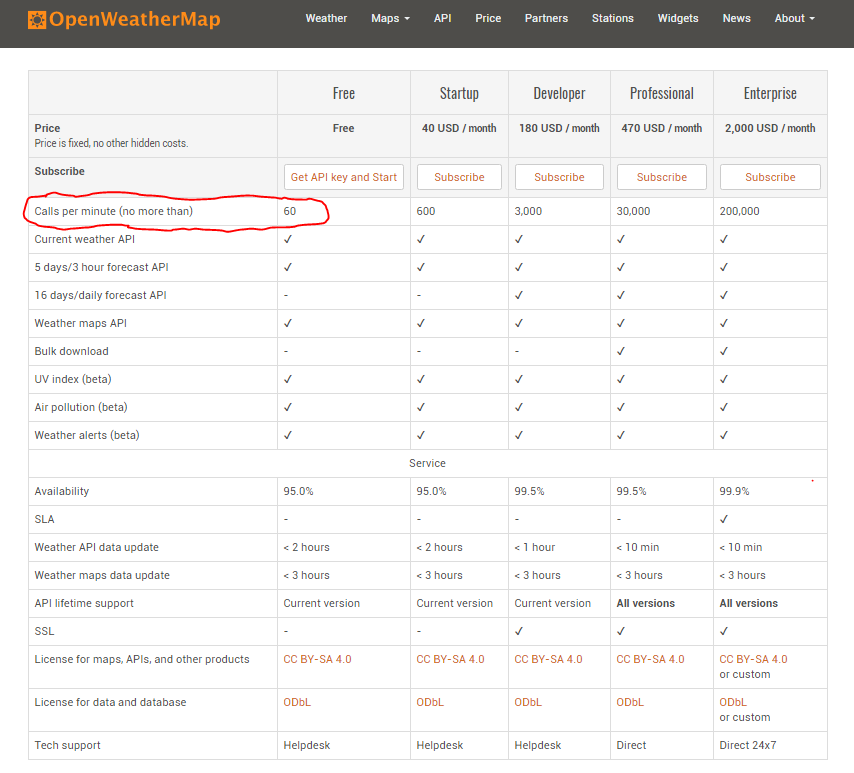


Рисунок 8 – Предоставление 60 запросов в минуту для публичного API

Просмотрев остальные сервисы, стоит отметить, что у всех стоит ограничение по запросам. Сервис «yandex.ru» предоставляет доступ к своим API и количество запросов в сутки 25000 в соответствии с рисунком 9, однако, ограничено некоторое количество функций. Так же предоставлены тарифы по использованию данного сервиса по предоставлению полного списка услуг. В соответствии с рисунком 10 предоставлена ценовая категории, также можно посчитать стоимость того, сколько запросов необходимо использовать. Данная ценовая категория относится к «яндекс картам», но не сильно отличается от других сервисов данного ресурса.

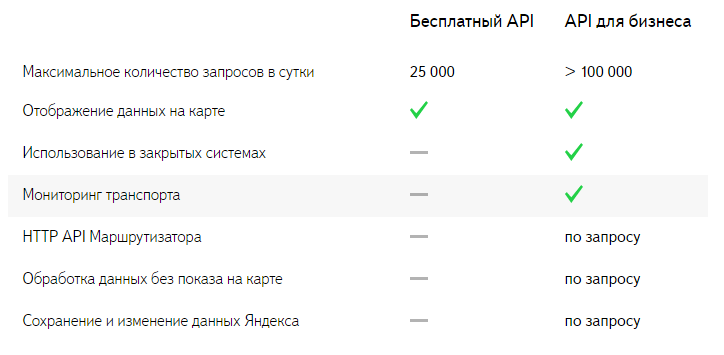


Рисунок 9 – Предоставление бесплатного API c сервиса yandex.ru

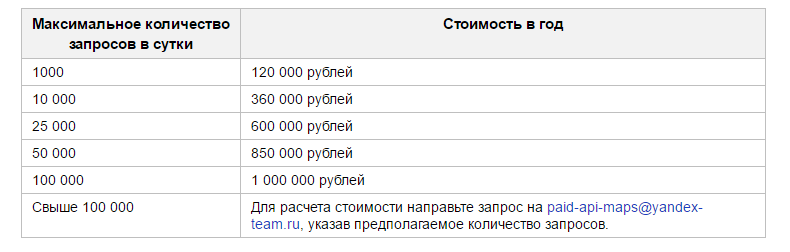


Рисунок 10 – Стоимость услуг сервиса «яндекс карты»

3.2 История API

«Концепция API предваряет даже появление персональных компьютеров, не говоря уже о Web, очень долгое время! Принцип хорошо документированного набора общедоступных «точек входа», которые позволяют приложению взаимодействовать с другой системой является неотъемлемой частью разработки программного обеспечения с самых ранних дней обработки служебных данных. Однако появление распределенных систем, а затем самой сети показало, что важность и полезность этих же базовых концепций резко возрастает». - Мартин Бартлетт

В 2017 году, есть некоторые очевидные технические причины того, почему веб-API находят успех в компаниях всех форм и размеров и даже внутри правительства, но не все причины этого успеха являются техническими. Есть много других, менее очевидных аспектов веб-APis, которые способствовали их успеху, чему можно только учиться, внимательно изучая прошлое и глядя на то, почему некоторые из первопроходцев веб-API были успешными, и продолжают добиваться успеха из года в год.

3.3 Применение публичных API

Для наглядности работы API предложено рассмотреть пример работы Google Maps Distance Matrix API.

Google Maps Distance Matrix API возвращает информацию на основе рекомендованного маршрута между исходной и конечной точками в соответствии с расчетами Google Maps API, которая состоит из строк, содержащих значения duration и distance для каждой пары точек.

Эта служба не возвращает подробной информации о маршруте. Информацию о маршруте можно получить, передав одну желаемую исходную точку и пункт назначения в Google Maps Directions API.

Запрос Google Maps Distance Matrix API имеет следующий вид:

*https://maps.googleapis.com/maps/api/distancematrix/outputFormat?parameters*

где *outputFormat* может принимать одно из следующих значений:

* *json* (рекомендуется) – задает вывод в формате JavaScript Object Notation (JSON); либо
* *xml* – задает вывод в формате XML.

Параметры запроса.

Некоторые параметры являются обязательными, другие – дополнительными. Все параметры разделяются амперсандами *(&)* в соответствии со стандартом URL-адресов.

Обязательные параметры.

*origins* – исходная точка для вычисления расстояния и времени пути. Можно задавать одно или несколько мест, отделяемых вертикальной чертой *(|)*, в виде адреса, координат широты и долготы или идентификатора места.

При отправке адреса служба геокодирует строку, преобразуя ее в координаты широты и долготы, чтобы вычислить расстояние. Данные координаты могут отличаться от координат, полученных от Google Maps Geocoding API, например, может быть указан вход в здание, а не его центр.

*origins=Bobcaygeon+ON|24+Sussex+Drive+Ottawa+ON*

Если передать координаты широты и долготы, они используются без изменений в расчетах расстояния. Убедиться, что между значениями широты и долготы отсутствует пробел.

*origins=41.43206,-81.38992|-33.86748,151.20699*

Если указать идентификатор места, к нему необходимо добавлять префикс *place\_id*:. Указать идентификатор места можно лишь в том случае, если запрос содержит ключ API или идентификатор клиента Google Maps APIs Premium Plan. Идентификаторы мест можно извлекать из Google Maps Geocoding API и Google Places API (включая подсказки мест).

*origins=place\_id:ChIJ3S-JXmauEmsRUcIaWtf4MzE*

Также можно указать совокупность кодированных координат, используя алгоритм кодирования ломаной линии. Он особенно полезен при наличии большого количества исходных точек, так как URL-адрес значительно короче при использовании кодированной ломаной линии.

Кодированные ломаные линии должны использоваться с префиксом *enc:* и последующим двоеточием *(:)*. Например, *origins=enc:gfo}EtohhU:*

Также можно включить несколько кодированных ломаных линий, разделив их вертикальной чертой *(|)*.

Например: *origins=enc:wc~oAwquwMdlTxiKtqLyiK:|enc:c~vnAamswMvlTor@tjGi}L:|enc:udymA{~bxM:*

*destinations* – одно или несколько мест, используемых как конечные точки при вычислении расстояния и времени пути. Параметры поля *destinations* аналогичны описанным выше параметрам поля *origins*.

*key* – ключ API необходим приложению. Этот ключ используется для идентификации приложения в целях управления квотами.

Системы единиц.

Результаты Distance Matrix содержат параметр *text* в поле *distance*, которое указывает расстояние для рассчитанного маршрута. Используемая система единиц может быть указана как:

* *units=metric* (по умолчанию) – указывает расстояния в метрах и километрах,
* *units=imperial* – указывает расстояния в милях и футах.

Пример запроса и ответа Google Maps Distance Matrix API.

Доступ к Google Maps Distance Matrix API осуществляется с помощью интерфейса HTTP. Для этого используются запросы, состоящие из строки URL-адреса, включающей исходные точки и точки назначения.

В следующем примере запрашиваются данные матрицы расстояний в формате JSON между столицей РФ городом Москвой и городом Красноярском.

*https://maps.googleapis.com/maps/api/distancematrix/json?****units=metric****&origins=****Moscow****&destinations=****Krasnoyarsk***

Для проверки необходимо прописать данный запрос в браузерную строку и нажать клавишу ввод. В соответствии с рисунком 11 данные должны быть введены правильно и не содержать иных знаков.

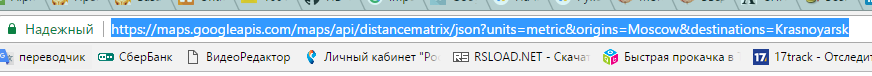


Рисунок 11 – Запрос в браузерной строке

Полученный ответ после выполнения запроса:

{

"destination\_addresses" : [

"Красноярск, Красноярский край, Россия"

],

"origin\_addresses" : [ "Москва, Россия" ],

"rows" : [

{

"elements" : [

{

"distance" : {

"text" : "4 153 км",

"value" : 4153124

},

"duration" : {

"text" : "2 дн. 6 ч.",

"value" : 196111

},

"status" : "OK"

}

]

}

],

"status" : "OK"

}

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 офоурмления 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.

4.1.6 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

4.2 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)

4.3 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.

Для реализации человеко-машинного взаимодействия, данная особенность низкого быстродействия не сильно скажется на работе бота информера, так как на данном этапе разработки, система не подразумевает большого количества пользователей, что значит бот сможет давать ответ пользователю за коротки промежуток времени. Тем не менее такой недостаток не сильно скажется на пользователе, чем на разработчике.

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.

5.6 Аудитория телеграмм

Все информационные ресурсы в сети, рассуждающие о популярности Telegram в той или иной стране, ссылаются на единственное масштабное исследование, которое провели веб-аналитики компании SimilarWeb. В этом исследовании использовались данные Android-устройств из 187 государств мира, с целью выявления самого популярного приложения для обмена сообщениями. Тем, что использовались показатели только устройств на базе Android в целом можно пренебречь. Для владельцев Iphone соотношение будет примерно таким же — в пределах одного региона и лингвистической группы результат не критично зависит от платформы.

В соответствии с рисунком 12, согласно полученным результатам, аудиторию Телеграмм составляют в основном жители таких республик, как Иран и Узбекистан. На этих территориях используются практически идентичные наречия, и спрос в Узбекистане стал, скорее всего, следствием популярности в Иране.



Рисунок 12 – Аудитория телеграмм по странам

Платформа SimilarWeb позволяет измерять и анализировать статистические данные о поведении посетителей по любым веб-сайтам и мобильным приложениям, которые имеют хоть какой-то минимальный трафик. Цифры довольно условны в разрезе одного объекта исследования и абсолютно точных реальных значений вам не узнать, но сервис позволяет хорошо сравнить 2-3 конкурента в одной области.

В соответствии с рисунком 13, приведенная статистика показывает лидеров в отрасли программ для обмена сообщениями в различных государствах. Но не дает представление о распределении процентов по языкам и странам в пределах одного мессенджера. То есть, из этих цифр мы не сможем выяснить, что среди пользователей проекта Павла Дурова 30% — русских, 30% — итальянцев и 40% — иранцев.

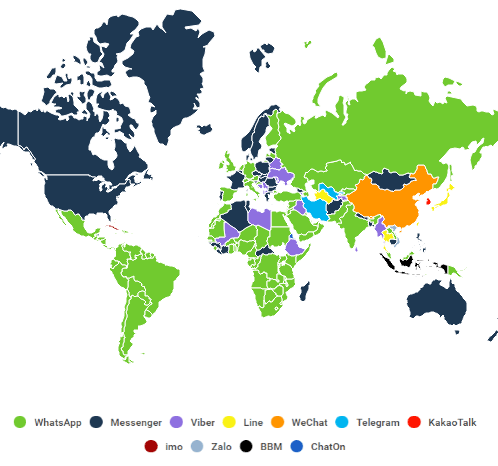


Рисунок 13 – Итоги исследования SimularWeb

Один из довольно популярных русскоязычных каналов об интернет-аналитике @internetanalytics проводил опрос с целью выявить социально-демографическую картину российских подписчиков. Этот опрос получил много критики, так как в нем принимала участие очень узкоспециализированная и малочисленная группа людей. Всего дали ответы около трех с половиной тысяч человек и в основном был охвачен профессиональный сегмент интернет-аналитиков. По итогам сформировался такой портрет: житель столицы в зрелом возрасте (25-30 лет), являющийся работником IT-сферы со средним заработком около 100 тысяч рублей в месяц.

Но отчасти эти результаты можно принимать на веру, поскольку именно в России Telegram активно использует как раз этот довольно узкий сегмент: более продвинутая, не реакционная аудитория. Более старшая, менее образованная и социально-обеспеченная прослойка общества активно использует мессенджеры-старички, такие как WhatsApp и Viber.

Большое сообщество из России помнит Павла Дурова еще по социальной сети ВКонтакте, и активно следит за его деятельностью. В “дуровскую” эпоху ВКонтакте была заложена мощная платформа для сторонней разработки, этот же паттерн переносится и на Телеграмм. Активное развитие платформы Telegram для сторонних разработчиков и СМИ привлекает программистов, специалистов в интернет-маркетинге и специалистов различных социальных СМИ и SMM. Они, в свою очередь, вовлекают все больше людей смежных специализаций. Многие из них пытаются сделать свои продукты: боты, каналы, большие групповые чаты.

То, что делают представители этого активного сегмента, прочно связано с их сферой деятельности и кругом общения: специализированные и профессиональные паблики и т.п. В следствие этого продвинутая аудитория растет. Поклонники Telegram отмечают, что мессенджер стал незаменимым рабочим инструментом, и находят его очень удобным для решения профессиональных задач.

Не секрет, что работники IT-сферы имеют довольно высокий доход и хорошее образование, то есть это очень качественная выборка. В подтверждение достаточно посмотреть на статистику по самой дорогой рекламе в пабликах на русском языке. Выигрывают более интеллектуальные паблики по сравнению с пабликами, которые содержат контент смешных видео и приколов в стиле MDK.

Признание мессенджера в России увеличивается, о чем хорошо свидетельствует возрастающая популярность ключевой фразы “телеграмм на русском” в Яндексе. Вышеописанной категории специалистов русифицировать приложение вряд ли нужно, они скорей всего знают английский язык на уровне интерфейсов. Как следствие, с увеличением популярности Telegram в СНГ, в ближайшем будущем аудитория будет становиться менее образованной и платежеспособной.

Касательно соотношения по половой принадлежности отсутствуют источник данных, которые могли бы точно прояснить эту ситуацию.

В результате получается что на сегодняшний день среднестатистический русскоговорящий пользователь использующий телеграмм обладает следующими характеристиками:

- возраст от 20 до 35 лет

- имеет высшее образование

- имеет работу интеллектуального характера (большой процент it-специалистов)

Стоит конечно отметить, что дальнейшее распространение телеграмм, доля прогрессивного меньшинства будет размываться другими социальными категориями

6. Выбор Субд

6**.1 История PostgresQL**

Немаловажно выбрать нужную СУБД для реализации бота мессенджера, необходимо иметь такую БД, которая будет работать с большим количество подключений, поэтому подходящий вариант это PostgreSQL.

PostgreSQL, первоначально названный Postgres, был создан в UCB профессором компьютерной науки по имени Майкл Стоунбрейкер, который стал техническим директором Informix Corporation. Стоунбрейкер начал разрабатывать Postgres в 1986 году в качестве последующего проекта своему предшественнику, Ingres, которому теперь принадлежит Computer Associates. Название Postgres таким образом отрывается от своего предшественника (как в «после Ingres»). Ingres, разработанный с 1977 по 1985 год, был упражнением в создании системы баз данных в соответствии с классической теорией РСУБД. Postgres, разработанный в период между 1986-1994 годами, был проектом, призванным открыть новые основы в концепциях базы данных, таких как исследование «объектно-реляционных» технологий.

Стоунбрейкер и его аспиранты активно развивали Postgres в течение восьми лет. В течение этого времени Postgres вводили правила, процедуры, расширяемые типы с индексами и объектно-реляционными концепциями. Postgres позже был коммерциализирован, чтобы стать Illustra, который позже был куплен Informix и интегрирован в свой Universal Server. Informix был приобретен IBM в 2001 году на один миллиард долларов.

В 1995 году два кандидата наук. Студенты из лаборатории Стоунбрейкера, Эндрю Ю и Джоли Чэн, заменили язык запросов Postgres POSTQUEL расширенным подмножеством SQL. Они переименовали систему в Postgres95.

В 1996 году Postgres95 отошла от академических кругов и начала новую жизнь в мире с открытым исходным кодом, когда группа преданных разработчиков за пределами Беркли увидела работу этой системы и посвятила себя ее дальнейшему развитию. Содействуя огромному количеству времени, навыкам, труду и технической экспертизе, эта глобальная группа развития радикально трансформировала Postgres. В течение следующих восьми лет они обеспечивали согласованность и единообразие базы кода, создавали подробные регрессионные тесты для обеспечения качества, настраивали списки рассылки для отчетов об ошибках, фиксировали неисчислимые ошибки, добавляли невероятные новые функции и дополняли систему, заполняя различные пробелы таких как документация для разработчиков и пользователей.

Результатом их труда стала новая база данных, получившая репутацию и солидную стабильность. С началом своей новой жизни в мире с открытым исходным кодом, с множеством новых функций и улучшений, система базы данных заняла свое текущее имя: PostgreSQL. («Postgres» по-прежнему используется как старое прозвище начальной системы).

PostgreSQL начинался с версии 6.0, давая кредит на многолетнюю предшествующую разработку. С помощью сотен разработчиков со всего мира система была изменена и улучшена практически во всех областях. В течение следующих четырех лет (версии 6.0 - 7.0) были сделаны значительные улучшения и новые возможности, такие как:

* Мультиверсионный контроль параллелизма. Блокировка на уровне таблиц была заменена сложной системой управления параллельными параллелизмами, которая позволяет читателям продолжать чтение согласованных данных во время активности записи и позволяет делать резервные копии во время работы базы данных в режиме реального времени;
* Важные функции SQL. Было сделано много усовершенствований SQL, включая подзапросы, значения по умолчанию, ограничения, первичные ключи, вторичные ключи, цитируемые идентификаторы, принудительное строковое кодирование, распределение типов ролей, двоичные и шестнадцатеричные целочисленные значения, и другие.
* Улучшены встроенные типы. Были добавлены новые типы, включая широкий диапазон типов дата-время и дополнительных геометрических типов.
* Скорость. Большой прирост производительности размере 20-40 процентов, а время запуска бэкэнда было уменьшено на 80 процентов.

За последующие четыре года (версии с 7.0 по 7.4) создано: журнал опережающей записи (WAL), схемы БД, подготовленные запросы, внешние соединения, сложные запросы, синтаксис join стандарта SQL92, методика хранения сверхбольших атрибутов (TOAST), поддержка IPv6 протокола, стандартную информационную схему SQL, полнотекстовое индексирование, автоматическое вакуумирование, процедурные языки Perl / Python / TCL, улучшенная поддержка SSL, оптимизация SQL запросов, информация о статистике базы данных, добавлена безопасность, функции таблицы и усовершенствования ведения журнала, а также значительные улучшения скорости.

Сегодня пользовательская база PostgreSQL больше, чем когда-либо, и она включена значительную группу крупных корпораций, которые используют ее в сложных условиях. Некоторые из этих компаний, Afilias и Fujitsu, внесли значительный вклад в развитие PostgreSQL. PostgreSQL продолжает улучшаться как в изощренности, так и в производительности более чем когда-либо. Версия 8.0 - долгожданный дебют PostgreSQL на рынке корпоративных баз данных, который включает такие функции, как табличные пространства, хранимые процедуры Java, восстановление времени и вложенные транзакции (точки сохранения). С ним появилась долгожданная функция - портирования Windows.

Многие организации, государственные учреждения и компании используют PostgreSQL.

Многие продолжают говорить о PostgreSQL как о "Postgres" (часто полностью заглавными буквами) потому что так привыкли или потому что так легче произносится. Это название широко используется в качестве сокращения или варианта полного названия PostgreSQL.

При разработке Postgres95 акцент ставился на обнаружение и понимание существующих проблем в коде сервера. В PostgreSQL акцент сместился на расширение возможностей и совместимости при продолжении работы во всех других областях.

Благодаря своему открытому коду, в России появилась своя система на базе PostgreSQL с именем PostgreSQL Pro. Postgres Pro - Российская СУБД, разработанная компанией Postgres Professional на основе свободно-распространяемой СУБД PostgreSQL. Postgres Pro входит в реестр российского ПО. [12]

6.2 PostgreSQL

PostgreSQL - это мощная система объектно-реляционных баз данных с открытым исходным кодом. Она имеет более чем 20-летнюю активную разработку и проверенную архитектуру, которая заслужила высокую репутацию за надежность, целостность данных и правильность. PostgreSQL работает на всех основных операционных системах, включая Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, macOS, Solaris, Tru64) и Windows, а также полностью совместим с ACID, имеет полную поддержку внешних ключей, соединений, просмотров, триггеров и хранимых процедур (на нескольких языках). Она включает большинство типов данных SQL: типы данных 2008 года, включая INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL и TIMESTAMP. А также поддерживает хранение двоичных больших объектов, включая изображения, звуки или видео. Имеет встроенные программные интерфейсы для C / C ++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, среди прочих, и исключительную документацию.

Сильными сторонами PostgreSQL считаются:

* высокопроизводительные и надёжные механизмы транзакций и репликации;
* расширяемая система встроенных языков программирования: в стандартной поставке поддерживаются PL/pgSQL, PL/Perl, PL/Python и PL/Tcl;
* дополнительно можно использовать PL/Java, PL/PHP, PL/Py, PL/R, PL/Ruby, PL/Scheme, PL/sh и PL/V8, а также имеется поддержка загрузки C-совместимых модулей;
* наследование;
* легкая расширяемость.

PostgreSQL создана на основе некоммерческой СУБД Postgres, разработанной как open-source проект в Калифорнийском университете в Беркли. К разработке Postgres, начавшейся в 1986 году, имел непосредственное отношение Майкл Стоунбрейкер, руководитель более раннего проекта Ingres, на тот момент уже приобретённого компанией Computer Associates. Название расшифровывалось как «Post Ingres», и при создании Postgres были применены многие уже ранее сделанные наработки.

PostgreSQL поддерживает большой набор встроенных типов данных:

* численные типы
* целые
* с фиксированной точкой
* с плавающей точкой
* денежный тип (отличается специальным форматом вывода, а в остальном аналогичен числам с фиксированной точкой с двумя знаками после запятой);
* символьные типы произвольной длины;
* двоичные типы (включая blob);
* типы «дата/время» (полностью поддерживающие различные форматы, точность, форматы вывода, включая последние изменения в часовых поясах);
* булев тип;
* перечисление;
* геометрические примитивы;
* сетевые типы;
* ip и ipv6-адреса;
* cidr-формат;
* mac-адрес;
* uuid-идентификатор;
* xml-данные;
* массивы;
* json;
* идентификаторы объектов бд;
* псевдотипы.

Более того, пользователь может самостоятельно создавать новые требуемые ему типы и программировать для них механизмы индексирования с помощью GiST.

PostgreSQL может быть расширен пользователем для собственных нужд практически в любом аспекте. Есть возможность добавлять собственные:

* преобразования типов;
* типы данных;
* домены (пользовательские типы с изначально наложенными ограничениями);
* функции (включая агрегатные);
* индексы;
* операторы (включая переопределение уже существующих);
* процедурные языки.

6.3 Сравнение PostgreSQL c другими СУБД

Для проведения сравнения приведены такие системы управления базами данных которые основаны на открытом исходном коде:

1. SQLite;
2. MySQL;
3. PostgreSQL.

1. SQLite представляет собой библиотеку, скомпонованную вместе с использующим ее приложением. Как и все встраиваемые базы данных, SQLite предоставляет внушительный набор инструментов для работы с самыми разными наборами данных при уменьшенных накладных расходах сравнительно с базами данных, построенных по клиент-серверной модели.

Благодаря тому, что вся база данных хранится в единственном файле, SQLite представляет собой очень быструю и эффективную СУРБД, которая используется как во встраиваемых системах, так и на выделенных машинах.

*Преимущества SQLite.*

* Вся база состоит из единственного файла данных, что обеспечивает высочайший уровень переносимости;
* Несмотря на свою облегченную структуру, в основе SQLite лежит SQL и его стандарты;
* SQLite удобна для разработчиков приложений, поскольку состоит всего из одного файла, а ее библиотека написана на Си и обладает множеством привязок к другим языкам программирования.

*Недостатки SQLite.*

* Отсутствие разграничения прав доступа
* Невозможность использования в крупных проектах
* Невозможность повышения эффективности

*В каких случаях следует использовать SQLite.*

* Для встроенных приложений
* Для прямого доступа к диску
* В целях тестирования

*В каких случаях не использовать SQLite.*

* При проектировании многопользовательских приложений
* При записи больших объемов данных

2. MySQL – самая популярная база данных для серверов с высокой нагрузкой. Это многофункциональный программный продукт с открытым кодом, взаимодействующий с огромным числом сайтов и веб-приложений.

Кроме того, MySQL характеризуется сравнительно низким порогом вхождения.

Несмотря на неполную имплементацию стандартов SQL, MySQL может похвастаться очень высоким функционалом.

MySQL, в отличие от SQLite, реализована согласно спецификациям клиент-серверной архитектуры, поэтому приложения обращаются к базе данных посредством службы MySQL.

*Преимущества MySQL.*

* Легкость установки и наличие визуальных средств управления;
* Поддержка, прямо или косвенно, обширного SQL-функционала;
* Наличие встроенных средств безопасности;
* Масштабируемость и работа с большими объемами данных;
* Игнорирование определенных стандартов увеличивает скорость работы и производительность MySQL.

*Недостатки MySQL.*

* MySQL далеко не универсальна и обладает определенными ограничениями функционала, которые «нравятся» далеко не всем приложениям;
* MySQL не настолько надежна, как хотелось бы, по сравнению с другими СУРБД;
* Несмотря на то, что технически MySQL представляет собой программное обеспечение с открытым кодом, процесс ее разработки фактически прекращен.

*В каких случаях использовать MySQL.*

* В случае необходимости совершения большого количества распределенных транзакций;
* Реализации требований высокой надежности;
* Повышенных требований к гибкости и универсальности для работы с сайтами и веб-приложениями;
* В случае соответствия требованиям индивидуального проекта.

*В каких случаях не использовать MySQL.*

* Если существует необходимость точного соответствия стандартам SQL;
* При необходимости многопоточности и параллелизма;
* При повышенных требованиях к функционалу – например, необходимости наличия полноценного текстового поиска.

3. PostgreSQL – это наиболее продвинутая объектно-реляционная база данных. PostgreSQL ставит перед собой задачу полного соответствия как стандартам SQL, так и международным стандартам ANSI/ISO.

Отличием PostgreSQL от других систем управления базами данных является интеграция объектно-ориентированной и/или реляционной моделей.

Благодаря своей внутренней структуре, PostgreSQL обладает высоким уровнем параллелизма и впечатляющей производительностью.

Несмотря на то, что PostgreSQL не настолько популярна, как MySQL, существует множество приложений сторонних производителей, значительно облегчающих работу с этой чрезвычайно мощной СУБД.

*Преимущества PostgreSQL.*

* PostgreSQL – это ПО с открытым исходным кодом, отвечающее всем стандартам СУРБД;
* Обширное сообщество пользователей;
* Наличие большого числа приложений сторонних производителей для настройки и управления;
* Расширение функционала за счет создания хранимых процедур;
* PostgreSQL – не реляционная, а объектно-реляционная база данных, поддерживающая все свойства ООП.

*Недостатки PostgreSQL.*

* Неудовлетворительная производительность на простых операциях чтения;
* PostgreSQL не пользуется особой популярностью, несмотря на развитое сообщество;
* Мало хостеров, предоставляющих поддержку PostgreSQL.

*Когда следует использовать PostgreSQL.*

* В случае, когда приоритетом является надежность хранения данных и их целостность;
* В случае необходимости выполнения большого количества пользовательских процедур;
* При запланированной миграции на платные СУРБД (например, Oracle);
* При наличии сложных структур данных.

*Когда не нужно использовать PostgreSQL.*

* Скорость чтения не относится к сильным сторонам PostgreSQL. Если данное условие существует, то PostgreSQL – не самый лучший выбор;
* Настройка PostgreSQL – утомительное занятие. Если многие из ее возможностей вам не нужны, то лучше подыскать другую СУРБД;
* В PostgreSQL существуют определенные сложности с репликацией, поэтому, если вы к ним не готовы, лучше использовать MySQL.

Исходя из данного сравнения для реализации системы человеко-машинного взаимодействия в виде бота для мессенджера telegram и делая упор на сильные стороны СУБД, PostgreSQL является отличным решением в данной работе. Так как необходима повышенная целость данных и отличная интеграция в среде Python. Не маловажно использование данной БД для подключения большого числа пользователей.

7 Процесс создания бота для телеграмм

7.1 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 в* соответствии с рисунком 14, после чего нам открывается подсказка с командами, нам нужно создать нового ботам следовательно пишем команду /*newbot*, которая отвечает за создание нового бота.



Рисунок 14– начало работы с @BotFather

Далее @BotFather предлагает нам придумать имя (имя бота должно обязательно оканчиваться на “bot”, в соответствии с рисунком 15 сделано это для того что бы можно было отличить живых людей от ботов.

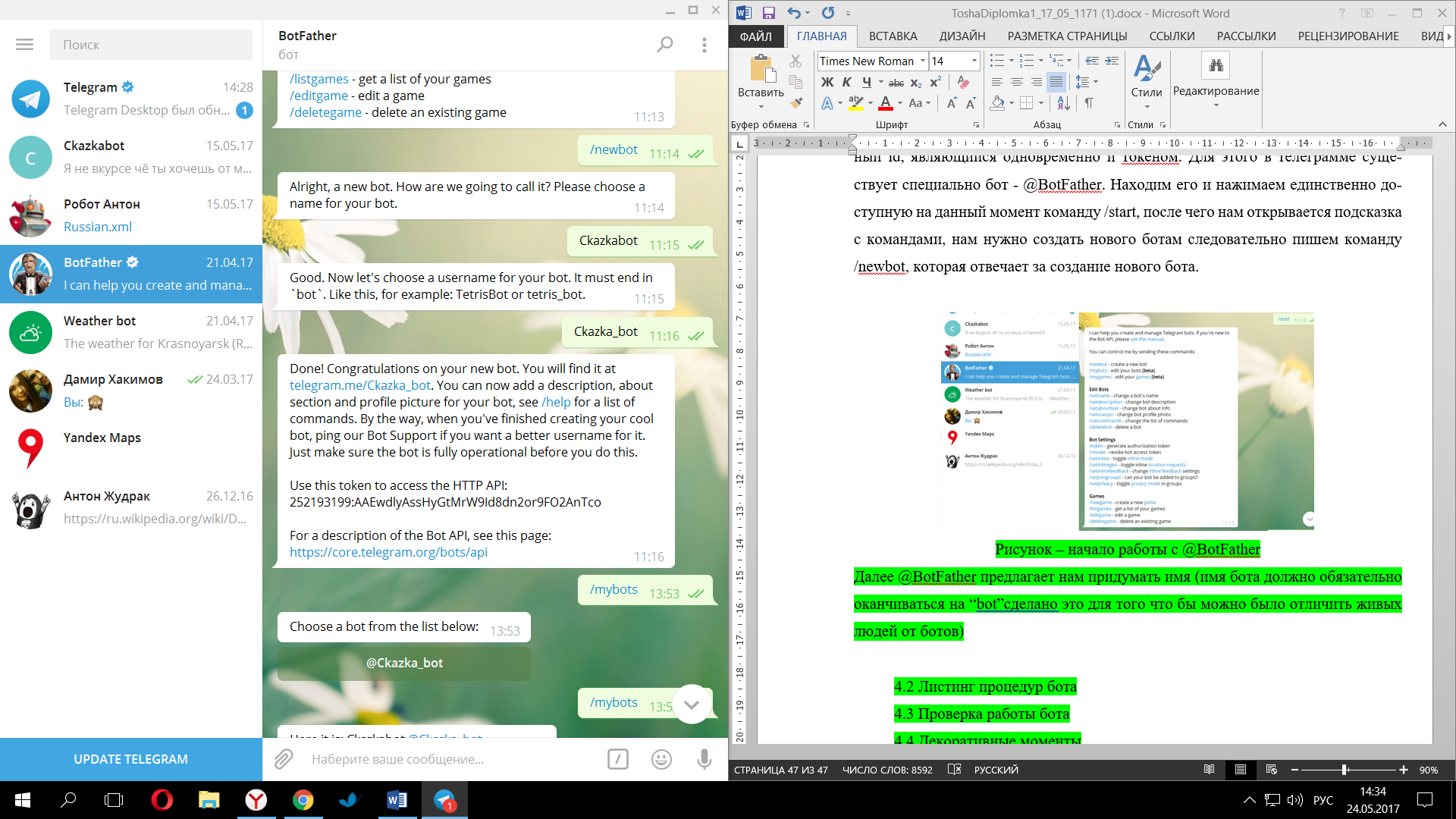


Рисунок 15 – Создание имени бота и присвоение токена

После создания имени @BotFather автоматически присвоит HTTP API token нашему боту и вернёт его вместе ссылкой для быстрого добавления бота в контакты.

7.2 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 в соответствии с рисунком 16.

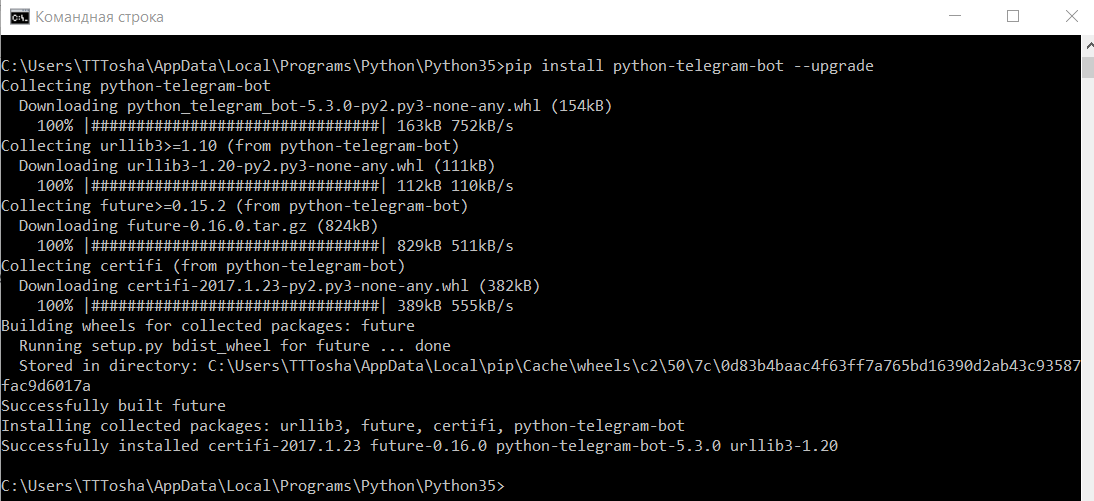


Рисунок 16 – Установки библиотеки *python-telegram-bot*

Библиотека установлена можно приступать к работе. Для начала создаём новый проект в PyCharm в соответствии с рисункмо 17. Внутрь проекта помещаем два основных файла bot55.py и config.py. В файле bot55.py будет написан основный код программы, а в файле config.py будут содержатся необходимые константы и токен нашего бота.

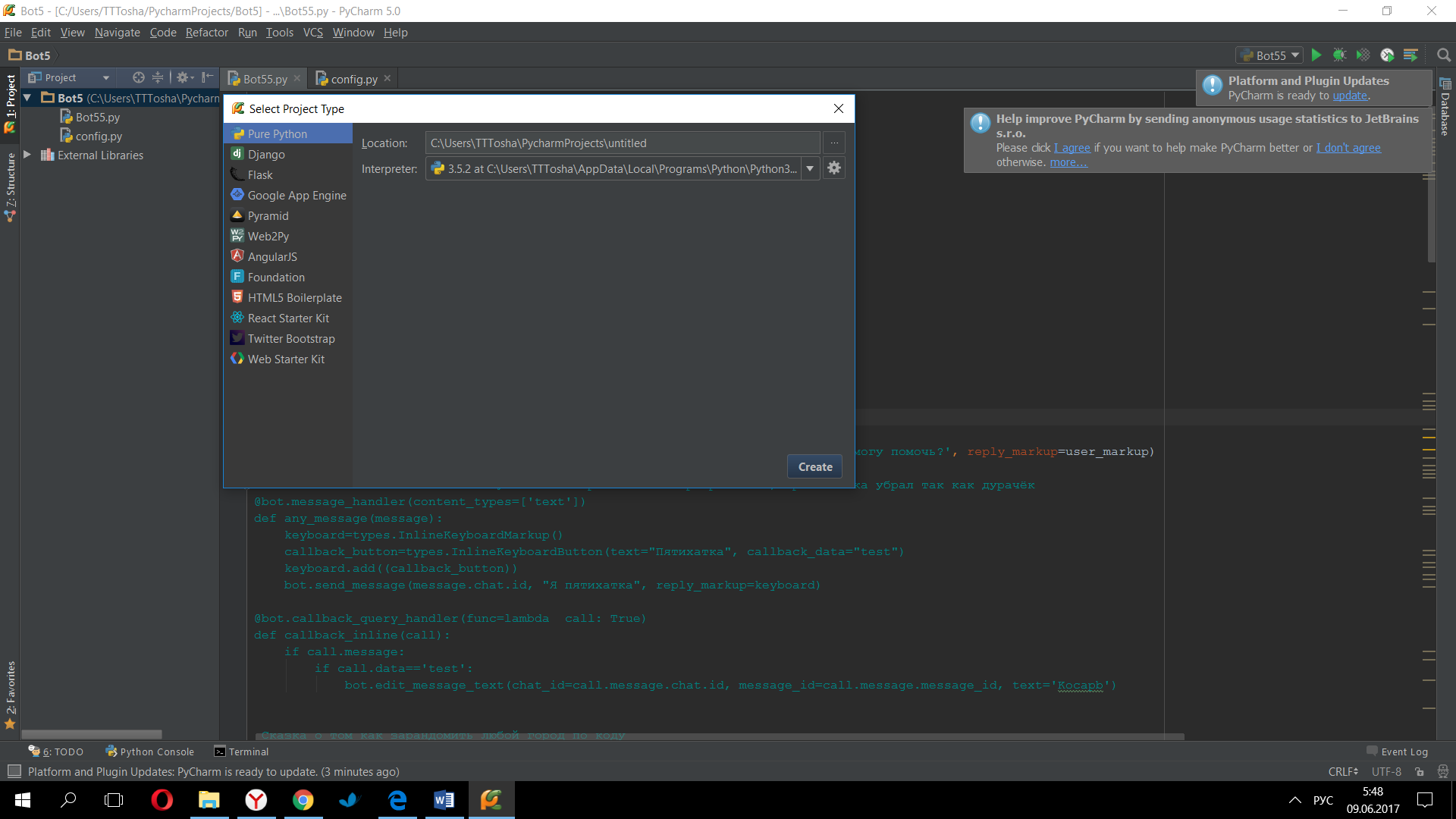


Рисунок 17 – создание нового проекта в PyCharm

Для начала создаём сам объект бота, через который будет осуществляться обращение к API Telegram с использованием методой описанных на сайте telegram в описании bot API. Для его создания нужно вызвать конструктор и передать токен бота в качестве его параметра.

Bot=telebot.Telebot(config.token)

По мере расширений функций бота, код будет начинать расти что может стать проблемой. Чтобы не загромождать код большим объёмом циклов и конструкций if/else/elif в библиотеке pyTelegramBotAPI существуют декораторы, которые помогают избежать этой проблемы. Декораторы указывают библиотеке, что следующая дальше функция должна быть вызвана когда боту проходит сообщение с подходящим типом содержимого, они существуют для всех типов сообщений, что очень удобно и позволяют разделить код на логичные части, не нагромождая его в одном месте, что в свою очередь повышает читабельность кода. Декораторы будут проверяться в порядке следования их в коде, будет использоваться тот декоратор который подошёл под тип сообщения первым. Внутри декораторов будет производится работа с текстом входящих сообщений.

7.3 Листинг процедур бота

Напишем первый декоратор, который будет отвечать за отправку вступительного сообщения пользователю при начале работы, с ботом. Декоратор сработает когда пользователь отправит боту команду /start.

@bot.message\_handler(commands=['start'])

def handle\_start(message):

bot.send\_message(message.chat.id, 'Привет, чем могу помочь?)

Второй декоратор будет отвечать за отправку сообщения при получении от пользователя команды /help.

@bot.message\_handler(commands=['help'])

def handle\_help(message):

bot.send\_message(message.chat.id, config.help\_message())

И наконец третий декоратор с помощью подключенной нами библиотеки pywapi будет получать погоду с сайта weather.com и преобразовывать полученный json-объект в удобный для пользователя вид.

@bot.message\_handler(content\_types=['text'])

def handle\_text(message):

#получаем словарь всех в мире городов, подходящий под запрос пользователя

city=pywapi.get\_location\_ids(message.text)

#перебираем словарь для получения последнего элемента

for i in city:

cityCode=i

#отправляем запрос на сайте weather.com для получения погоды

weather\_com\_result=pywapi.get\_weather\_from\_weather\_com(cityCode)

#формируем сообщения с погодой из json-объекта, полученного нами от сайта

weatherReport="Сейчас в городе" +weather\_com\_result['current\_conditions']['text'].lower()+" и "+weather\_com\_result['current\_conditions']['temperature'] +" °С +"\n"+weather\_com\_result['current\_conditions']['last\_updated']

bot.send\_message(message.chat.id, weatherReport)

Для того чтобы наш код не останавливался после выполнения, нужно постоянно спрашивать сервер об обновлениях. Для этого будет использовать функцию polling. Функция polling запускает Long Polling, а параметры none\_stop=True и interval=0, будут говорить боту что бы он не прекращал работу при возникновении каких либо ошибок

bot.polling(none\_stop=True, interval=0)

Протестируем нашего бота в соответствии с рисунком 18, всё отлично работает.

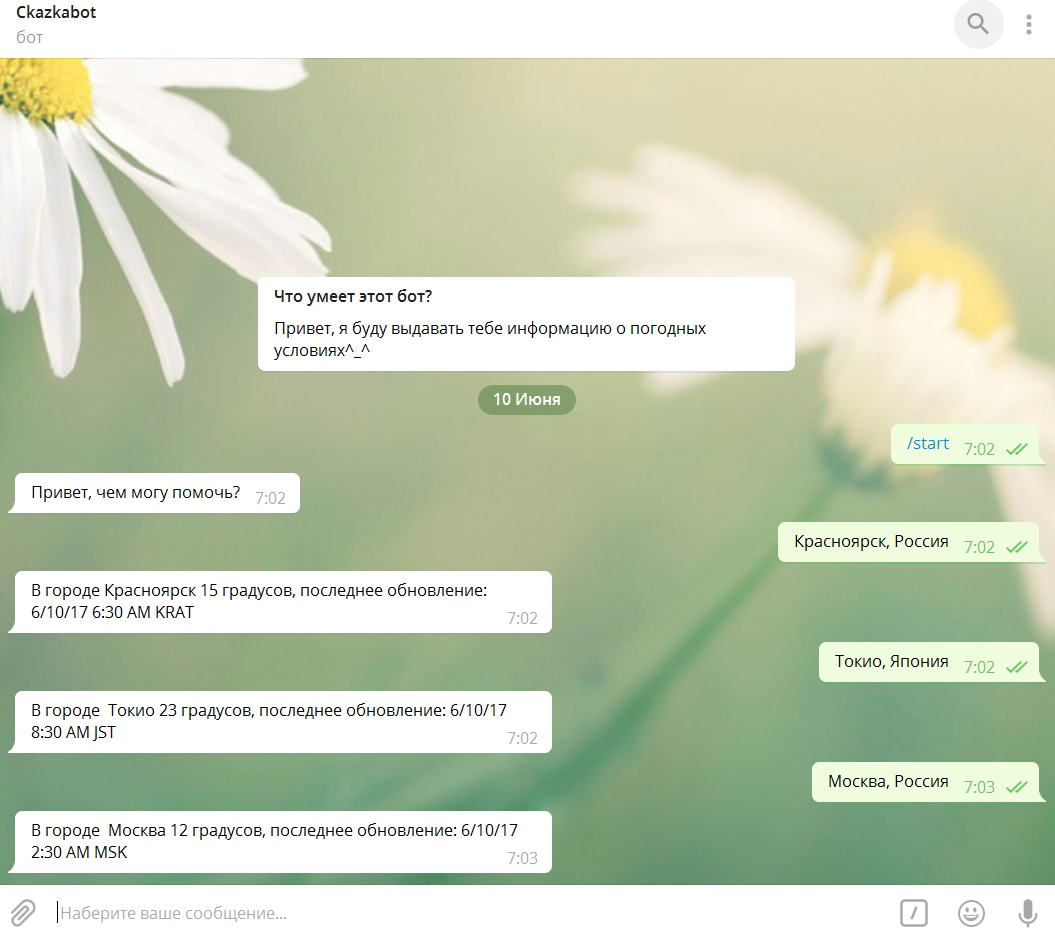


Рисунок 18– Проверка бота

Теперь перейдём к кастомным клавиатурам. Кастомные клавиатуры это одна из особенностей Bot API, такие клавиатуры представляют собой шаблоны сообщений. Суть в том что команду которая отправляется кнопкой клавиатуры, полностью соответствует её названию. Например если кнопка на кастомной клавиартура называется /help то при нажатии на неё в чат отправится команда /help. Для создания такой клавиатуры нужно создать объект ReplyKeyboardMarkup(). Далее добавить построчно элементы либо же выбрать автоматическое разделение строки (параметры row,width) и передать в метод send\_message().

Добавим кастомную клавиатуру состоящую из 4 клавиш: ‘Погода на сегодня’, команды /help, ‘Задать вопрос’ и ‘’Расскажи анекдот. Немного обновим декоратор команды /start.

@bot.message\_handler(commands=['start'])

def handle\_start(message):

user\_markup=telebot.types.ReplyKeyboardMarkup(True, False)

user\_markup.row('Погода на сегодня', '/help')

user\_markup.row('Расскажи анекдот', 'Задать вопрос')

bot.send\_message(message.chat.id, 'Привет, '+' чем могу помочь?', reply\_markup=user\_markup)

4.3 Проверка работы бота

4.152 Подключение к базе Дамиркиной

4.4 Декоративные моменты

Добавим различные эмодзи к нашим подписям в командах.

Эмодзи – это стиль иконок наподобие смайликов, появившихся в Японии и получивших популярность в социальных сетях по всему миру благодаря интернету. Японское слово «эмодзи» состоит из «э» (картинка) и «модзи» (буква, письменный знак). То есть получается что-то наподобие «буква-картинка». Основное отличие эмодзи от стандратного набора смайликов – это более крупный размер и более широкий набор эмоций, стилей, начертаний и образов в которые входят иконки.

Для добавления эмодзи к нашему меню и подписям будем использовать базу эмодзи расшифрованный юникодом, находится такая база на сайте charbase.com. Находим нужную нам эмодзи и копируем юникод в соответствии с рисунком 19.

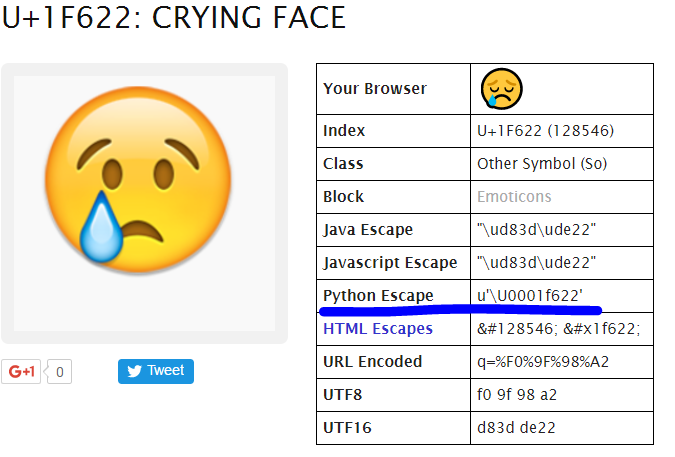


Рисунок 19 – Юникод выбранной нами эмодзи

Осталось только найти нужные нам эмодзи и добавить в код, покажем на примере команды /start, выглядеть это будет таким образом:

@bot.message\_handler(commands=['start'])

def handle\_start(message):

user\_markup=telebot.types.ReplyKeyboardMarkup(True, False)

user\_markup.row('\u2601 Погода на сегодня', '/help \U0001f631')

user\_markup.row('\U0001f61c Расскажи анекдот', 'Задать вопрос \U0001f4ac')

bot.send\_message(message.chat.id, 'Привет, '+message.chat.first\_name+' чем могу помочь?', reply\_markup=user\_markup)

Теперь оформим красивое приветствие нашего бота добавим подсказки к командам изменим его описание и изменим иконку бота. Для этого обратимся к первоистокам к @BotFather изначальный бот, который выдал нам токен нашего бота и зная токен позволяет нам изменять нашего бота.

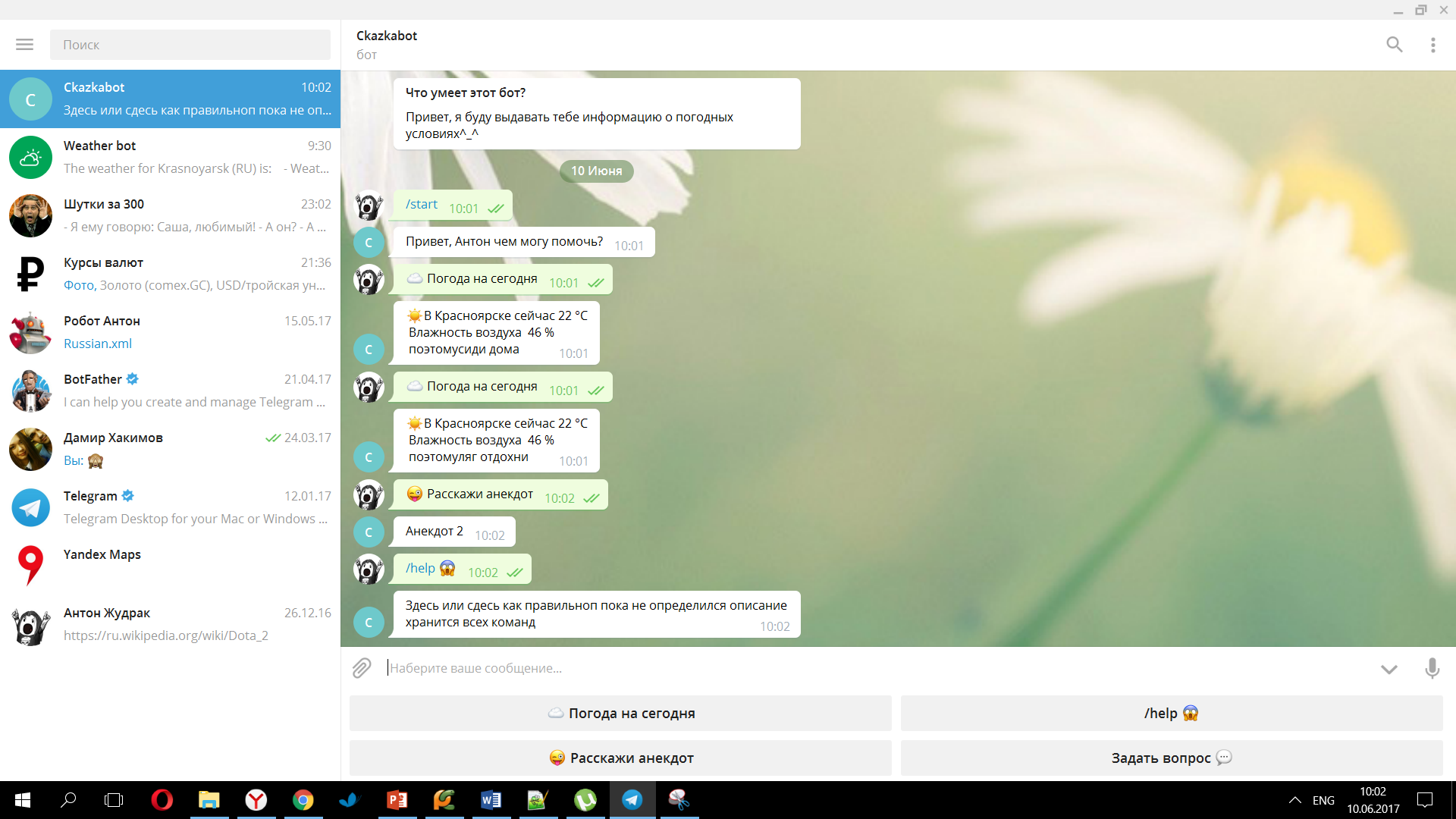
Добавление красивой иконки бота и изменение его описание

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>



- Пункт про аудиторию телеграмм: <https://rb.ru/opinion/russian-telegram-inside/>

- Аудиториия телеграмм общая статистика и анализ: <https://ru.telegram-store.com/blog/auditorya_telegram/>

- 1.7 тенденции развития чат ботов в 2017 году. Источник: https://chatbotconf.ru/ru/article/7-osnovnih-tendentsiy-v-sfere-razrabotki-chat-botov-63457 © ChatBot Conference RU

Как погода?

Как сам?

Займи денег? (я студент лучше сам мне закинсь и номер кошелька)

Вопросы

Патч 08.06.2017

- Изменения добавлен пункт 5.6 Аудитория ботов

- Примечания измененены (историю только оставил там)

- Добавлено описание бота это про пункт 6

Добавить что можно? Вот это

-3 пункт как у Дамирки взять

-Можно добавить про JSON объекты что такое чутка страничку например)

Антиплагиат в основном только в истории питона

3.) выбор субд мб потом коротко добавить где нибудь если страниц мало будет

4.) критику может взять и цензуру (3.6, 3.7 у Дамирки)

аудитория телеграмм по странам ещё с графиками



Актуальность(спёр не подходит ко мне конечно, ну ничего)

Информационно-коммуникационные технологии становятся неотъемлемой частью жизни человека, меняется сам характер взаимодействия между ними. Технологии призваны поддерживать определенные человеческие ценности, но нередко вступают в противоречие с другими, негативно влияя на качество жизни. Возможно ли сохранить ключевые ценности под натиском перемен? Ответ на этот жизненно важный вопрос во многом зависит от исследований и разработок в сфере взаимодействия человека и компьютера — междисциплинарной области, динамично развивающейся в последние десятилетия. В данном направлении активно действует компания Microsoft, результаты многолетних наработок которой обобщаются в статье. Рассматриваются история развития соответствующих исследований, их современное состояние и будущие задачи. Анализируются сущность базовых тенденций, радикально преобразующих отношения человека с технологиями, обусловленные ими возможности, риски и актуальные направления развития науки. Приводятся рекомендации по преобразованию рассматриваемой области, которые позволят сформировать адекватный ответ на будущие вызовы. Прежде всего, необходимо расширить научно-проектный цикл за счет введения дополнительной стадии — концептуального анализа, касающегося этических, социальных и политических эффектов складывающихся тенденций. Следует усилить внимание к междисциплинарным аспектам взаимодействия человека с компьютером, что, в свою очередь, предполагает формирование нового профессионального языка, который дал бы возможность проанализировать зарождающиеся перемены и способы управления. Центральное место в меняющемся калейдоскопе технологий должны занять базовые ценности, поддерживающие человеческую идентичность.