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 Хакимов Дамир Рашитович

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

**(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 В.С./

**АННОТАЦИЯ**

ХАКИМОВ ДАМИР РАШИТОВИЧ

РАЗРАБОТКА СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ В ЕЖЕДНЕВНОМ БЫТУ

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

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

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

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

**ПРИМЕР.** В магистерской диссертации анализ о системах поддержки принятия решений и методу их применения к созданию бота информера.

**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](#_Toc484777878)

[1 Системы поддержки принятия решений 9](#_Toc484777879)

[1.1 Преимущества СППР 13](#_Toc484777880)

[1.2 Метод анализа иерархий 13](#_Toc484777881)

[1.3 Пример задачи многокритериального выбора с простейшей иерархией 15](#_Toc484777882)

[1.4 Сфера образования и научных исследований 16](#_Toc484777883)

[1.5 Методика применения МАИ 16](#_Toc484777884)

[1.6 Моделирование проблемы в виде иерархии 17](#_Toc484777885)

[1.7 Определение иерархической структуры 17](#_Toc484777886)

[1.8 Объяснение иерархических структур, используемых в МАИ 18](#_Toc484777887)

[1.9 Расстановка приоритетов 19](#_Toc484777888)

[1.10 Определение приоритетов и пояснения 20](#_Toc484777889)

[1.11 История развития систем поддержки принятия решений 22](#_Toc484777890)

[2 Публичные API 25](#_Toc484777891)

[2.1 История API 25](#_Toc484777892)

[3 Выбор СУБД 35](#_Toc484777893)

[3.1 PostgreSQL 38](#_Toc484777894)

[4 Среда разработки. 48](#_Toc484777895)

[4.1 Причины использования Python 49](#_Toc484777896)

[4.1.1 Качество программного обеспечения 49](#_Toc484777897)

[4.1.2 Высокая скорость разработки 49](#_Toc484777898)

[4.1.3 Переносимость программ 50](#_Toc484777899)

[4.1.4 Библиотеки поддержки 50](#_Toc484777900)

[4.1.5 Интеграция компонентов 50](#_Toc484777901)

[4.2 История создания 51](#_Toc484777902)

[4.3 Влияние других языков 52](#_Toc484777903)

[4.4 Функциональные возможности 53](#_Toc484777904)

[4.5 Прикладное применение Python 54](#_Toc484777905)

[4.6 Преимущество Python перед другими языками высокого уровня 56](#_Toc484777906)

[4.7 Реализации Python 57](#_Toc484777907)

[4.8 Недостатки 58](#_Toc484777908)

[5 Основные причины выбора мессенджера Telegram 60](#_Toc484777909)

[5.1 Telegram (мессенджер) 61](#_Toc484777910)

[5.2 История 62](#_Toc484777911)

[5.3 Технология 64](#_Toc484777912)

[5.4 Боты 65](#_Toc484777913)

[5.5 Конкурсы поиска уязвимостей 65](#_Toc484777914)

[5.6 Критика 67](#_Toc484777915)

[5.7 Цензура 68](#_Toc484777916)

[6 Моделирование системы поддержки принятия решений при обращении к боту мессенджеру 69](#_Toc484777917)

[Заключение 71](#_Toc484777918)

[Источники 72](#_Toc484777919)

Введение

В той или иной степени, системы поддержки принятия решений (СППР) присутствуют в любой информационной системе (ИС). Поэтому к задаче создания системы поддержки принятия решений организации приступают сразу после приобретения вычислительной техники и установки программного обеспечения. По мере развития бизнеса, упорядочения структуры организации и налаживания межкорпоративных связей, проблема разработки и внедрения СППР становится особенно актуальной. Одним из подходов к созданию таких систем стало использование хранилищ данных.

СППР можно, в зависимости от данных, c которыми они работают, разделить на оперативные, предназначенные для немедленного реагирования на текущую ситуацию, и стратегические - основанные на анализе большого количества информации из разных источников с привлечением сведений, содержащихся в системах, аккумулирующих опыт решения проблем.

СППР первого типа получили название Информационных Систем Руководства (ИСР). По сути, они представляют собой конечные наборы отчетов, построенные на основании данных из транзакционной информационной системы предприятия или OLTP-системы, в идеале адекватно отражающей в режиме реального времени все аспекты производственного цикла предприятия. Для ИСР характерны следующие основные черты:

- отчеты, как правило, базируются на стандартных для организации запросах; число последних относительно невелико;

- ИСР представляет отчеты в максимально удобном виде, включающем, наряду с таблицами, деловую графику, мультимедийные возможности и т. п.;

- как правило, ИСР ориентированы на конкретный вертикальный рынок, например финансы, маркетинг, управление ресурсами.

СППР второго типа предполагают достаточно глубокую проработку данных, специально преобразованных так, чтобы их было удобно использовать в ходе процесса принятия решений. Неотъемлемым компонентом СППР этого уровня являются правила принятия решений, которые на основе агрегированных данных подсказывают менеджерскому составу выводы и придают системе черты искусственного интеллекта. Такого рода системы создаются только в том случае, если структура бизнеса уже достаточно определена и имеются основания для обобщения и анализа не только данных, но и процессов их обработки. Если ИСР есть не что иное как развитие системы оперативного управления производственными процессами, то СППР в современном понимании - это механизм развития бизнеса, который включает в себя некоторую часть управляющей информационной системы, обширную систему внешних связей предприятия, а также технологические и маркетинговые процессы развития производства.

**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 Системы поддержки принятия решений

Система поддержки принятия решений имеет не одно, а несколько видов определений, варианты представлены ниже.

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 др.

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 типо-вые 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 также 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 со-голосования 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.

Теоретические исследования в области разработки первых систем поддержки принятия решений проводились в технологическом институте Карнеги в конце 50-х начале 60-х годов XX века. Объединить теорию с практикой удалось специалистам из Массачусетского технологического института в 60-х годах. В середине и конце 80-х годов XX столетия стали появляться такие системы, как EIS, GDSS, ODSS. В 1987 году компания Texas Instruments разработала для United Airlines Gate Assignment Display System. Это позволило значительно снизить убытки от полетов и отрегулировать управление различными аэропортами, начиная от Международного аэропорта O’Hare в Чикаго и заканчивая Stapleton в Денвере, штат Колорадо. В 90-х годах сфера возможностей СППР расширялась благодаря внедрению хранилищ данных и инструментов OLAP. Появление новых технологий отчетности сделало СППР незаменимой в менеджменте.

1.1 Преимущества СППР

СППР позволяет облегчить работу руководителям предприятий и повысить ее эффективность. Они значительно ускоряют решение проблем в бизнесе. СППР способствуют налаживанию межличностного контакта. На их основе можно проводить обучение и подготовку кадров. Данные информационные системы позволяют повысить контроль над деятельностью организации. Наличие четко функционирующей СППР дает большие преимущества по сравнению с конкурирующими структурами. Благодаря предложениям, выдвигаемым СППР, открываются новые подходы к решению повседневных и нестандартных задач.

1.2 Метод анализа иерархий

Метод Анализа Иерархий (МАИ, иногда МетАнИе) — математический инструмент системного подхода к сложным проблемам принятия решений. МАИ не предписывает лицу, принимающему решение (ЛПР), какого-либо «правильного» решения, а позволяет ему в интерактивном режиме найти такой вариант (альтернативу), который наилучшим образом согласуется с его пониманием сути проблемы и требованиями к её решению. Этот метод разработан американским математиком Томасом Саати, который написал о нем книги, разработал программные продукты и в течение 20 лет проводит симпозиумы ISAHP (англ. *International Symposium on Analytic Hierarchy Process*). МАИ широко используется на практике и активно развивается учеными всего мира. В его основе наряду с математикой заложены и психологические аспекты. МАИ позволяет понятным и рациональным образом структурировать сложную проблему принятия решений в виде иерархии, сравнить и выполнить количественную оценку альтернативных вариантов решения. Метод Анализа Иерархий используется во всем мире для принятия решений в разнообразных ситуациях: от управления на межгосударственном уровне до решения отраслевых и частных проблем в бизнесе, промышленности, здравоохранении и образовании. Для компьютерной поддержки МАИ существуют программные продукты, разработанные различными компаниями. Анализ проблемы принятия решений в МАИ начинается с построения иерархической структуры, которая включает цель, критерии, альтернативы и другие рассматриваемые факторы, влияющие на выбор. Эта структура отражает понимание проблемы лицом, принимающим решение. Каждый элемент иерархии может представлять различные аспекты решаемой задачи, причем во внимание могут быть приняты как материальные, так и нематериальные факторы, измеряемые количественные параметры и качественные характеристики, объективные данные и субъективные экспертные оценки. Иными словами, анализ ситуации выбора решения в МАИ напоминает процедуры и методы аргументации, которые используются на интуитивном уровне. Следующим этапом анализа является определение приоритетов, представляющих относительную важность или предпочтительность элементов построенной иерархической структуры, с помощью процедуры парных сравнений. Безразмерные приоритеты позволяют обоснованно сравнивать разнородные факторы, что является отличительной особенностью МАИ. На заключительном этапе анализа выполняется синтез (линейная свертка) приоритетов на иерархии, в результате которой вычисляются приоритеты альтернативных решений относительно главной цели. Лучшей считается альтернатива с максимальным значением приоритета.

1.3 Пример задачи многокритериального выбора с простейшей иерархией

В данной задаче необходимо выбрать из трех кандидатов одного на должность руководителя. Кандидаты оцениваются по критериям: возраст, опыт, образование и личные качества. В соответствии с рисунком 1.3.1 показана иерархия для этой задачи. Простейшая иерархия содержит три уровня: цель, критерии и альтернативы. Числа на рисунке показывают приоритеты элементов иерархии с точки зрения цели, которые вычисляются в МАИ на основе парных сравнений элементов каждого уровня относительно связанных с ними элементами вышерасположенного уровня. Приоритеты альтернатив относительно цели (глобальные приоритеты) вычисляются на заключительном этапе метода путём линейной свертки локальных приоритетов всех элементов. В данном примере лучшим кандидатом является Дик, так как имеет максимальное значение глобального приоритета.

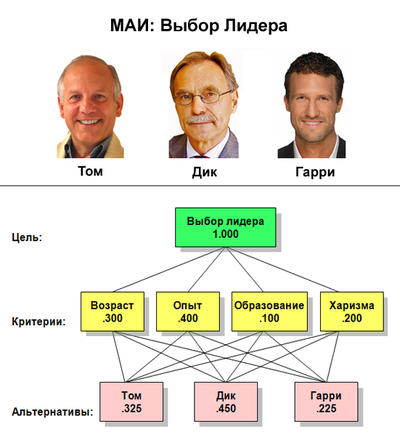


Рисунок 1.3.1 – МАИ. Выбор лидера

1.4 Сфера образования и научных исследований

Хотя для практического применения МАИ отсутствует необходимость специальной подготовки, основы метода преподают во многих учебных заведениях. Кроме того, этот метод широко применяется в сфере управления качеством и читается в рамках многих специализированных программ, таких как Six Sigma, Lean Six Sigma, и QFD. Около ста китайских университетов предлагают курсы по основам МАИ, и многие соискатели научных степеней выбирают МАИ в качестве объекта научных и диссертационных исследований. Опубликовано более 900 научных статей по данной тематике. Существует китайский научный журнал, специализирующийся в области МАИ. Раз в два года проводится Международный симпозиум, посвященный МАИ (International Symposium on Analytic Hierarchy Process, ISAHP), на котором встречаются как ученые, так и практики, работающие с МАИ. В 2007 году симпозиум проходил в Вальпараисо, Чили, где было представлено более 90 докладов ученых из 19 стран, включая США, Германию, Японию, Чили, Малайзию, и Непал.

1.5 Методика применения МАИ

Метод анализа иерархий содержит процедуру синтеза приоритетов, вычисляемых на основе субъективных суждений экспертов. Число суждений может измеряться дюжинами или даже сотнями. Математические вычисления для задач небольшой размерности можно выполнить вручную или с помощью калькулятора, однако гораздо удобнее использовать программное обеспечение (ПО) для ввода и обработки суждений. Самый простой способ компьютерной поддержки — электронные таблицы, самое развитое ПО предусматривает применение специальных устройств для ввода суждений участниками процесса коллективного выбора. Порядок применения Метода Анализа Иерархий:

1. Построение качественной модели проблемы в виде иерархии, включающей цель, альтернативные варианты достижения цели и критерии для оценки качества альтернатив.
2. Определение приоритетов всех элементов иерархии с использованием метода парных сравнений.
3. Синтез глобальных приоритетов альтернатив путём линейной свертки приоритетов элементов на иерархии.
4. Проверка суждений на согласованность.
5. Принятие решения на основе полученных результатов.

1.6 Моделирование проблемы в виде иерархии

Первый шаг МАИ — построение иерархической структуры, объединяющей цель выбора, критерии, альтернативы и другие факторы, влияющие на выбор решения. Построение такой структуры помогает проанализировать все аспекты проблемы и глубже вникнуть в суть задачи.

1.7 Определение иерархической структуры

Иерархическая структура — это графическое представление проблемы в виде перевернутого дерева, где каждый элемент, за исключением самого верхнего, зависит от одного или более выше расположенных элементов. Часто в различных организациях распределение полномочий, руководство и эффективные коммуникации между сотрудниками организованы в иерархической форме.

Иерархические структуры используются для лучшего понимания сложной реальности: мы раскладываем исследуемую проблему на составные части; затем разбиваем на составные части получившиеся элементы и т. д. На каждом шаге важно фокусировать внимание на понимании текущего элемента, временно абстрагируясь от всех прочих компонентов. При проведении подобного анализа приходит понимание всей сложности и многогранности исследуемого предмета.

В качестве примера можно привести иерархическую структуру, которая используется при обучении в медицинских вузах. В рамках изучения анатомии отдельно рассматривается костно-мышечная система (которая включает такие элементы, как руки и их составляющие: мышцы и кости), сердечнососудистая система (и её множественные уровни), нервная система (и её компоненты и подсистемы) и т. д. Степень детализации доходит до клеточного и молекулярного уровня. В конце изучения приходит понимание системы организма в целом, а также осознание того, какую роль играет в нем занимает каждая часть. С помощью подобного иерархического структурирования студенты приобретают всесторонние знания об анатомии.

Аналогичным образом, когда мы решаем сложную проблему, мы можем использовать иерархию как инструмент для обработки и восприятия больших объемов информации. По мере проектирования этой структуры у нас формируется все более полное понимание проблемы.

Чтобы избежать беспорядка в диаграммах МАИ, связи, соединяющие Альтернативы и их покрывающие Критерии, часто опускаются, или их количество искусственно уменьшается. Несмотря на такие упрощения в диаграмме, в самой иерархии каждая в соответствии с рисунком 1.7.1 Альтернатива связана с каждым из покрывающих её Критериев.

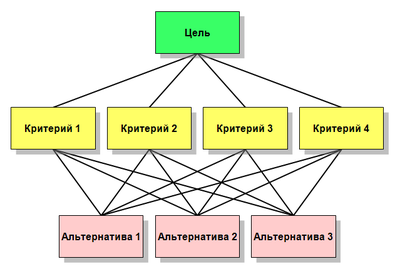


Рисунок 1.7.1 – Простейшая иерархия МАИ

1.8 Объяснение иерархических структур, используемых в МАИ

Иерархические структуры, используемые в МАИ, представляют собой инструмент для качественного моделирования сложных проблем. Вершиной иерархии является главная цель; элементы нижнего уровня представляют множество вариантов достижения цели (альтернатив); элементы промежуточных уровней соответствуют критериям или факторам, которые связывают цель с альтернативами. Существуют специальные термины для описания иерархической структуры МАИ. Каждый уровень состоит из узлов. Элементы, исходящие из узла, принято называть его детьми (дочерними элементами). Элементы, из которых исходит узел, называются родительскими. Группы элементов, имеющие один и тот же родительский элемент, называются группами сравнения. Родительские элементы Альтернатив, как правило, исходящие из различных групп сравнения, называются покрывающими Критериями. Используя эти термины для описания представленной ниже диаграммы, можно сказать, что четыре Критерия — это дети Цели; в свою очередь, Цель — это родительский элемент для любого из Критериев. Каждая Альтернатива — это дочерний элемент каждого из включающих её Критериев. Всего на диаграмме присутствует две группы сравнения: группа, состоящая из четырех Критериев и группа, включающая три Альтернативы. Вид любой иерархии МАИ будет зависеть не только от объективного характера рассматриваемой проблемы, но и от знаний, суждений, системы ценностей, мнений, желаний и т. п. участников процесса. Опубликованные описания применений МАИ часто включают в себя различные схемы и объяснения представленных иерархий. Последовательное выполнение всех шагов МАИ предусматривает возможность изменения структуры иерархии, с целью включения в неё вновь появившихся, или ранее не считавшихся важными, Критериев и Альтернатив

1.9 Расстановка приоритетов

После построения иерархии участники процесса используют МАИ для определения приоритетов всех узлов структуры. Информация для расстановки приоритетов собирается со всех участников и математически обрабатывается. В данном разделе приведена информация, на простом примере поясняющая процесс вычисления приоритетов.

1.10 Определение приоритетов и пояснения

Приоритеты — это числа, которые связаны с узлами иерархии. Они представляют собой относительные веса элементов в каждой группе. Подобно вероятностям, приоритеты — безразмерные величины, которые могут принимать значения от нуля до единицы. Чем больше величина приоритета, тем более значимым является соответствующий ему элемент. Сумма приоритетов элементов, подчиненных одному элементу выше лежащего уровня иерархии, равна единице. Приоритет цели по определению равен 1.0. Рассмотрим простой пример в соответствии с рисунком 1.10.1, поясняющий методику вычисления приоритетов.

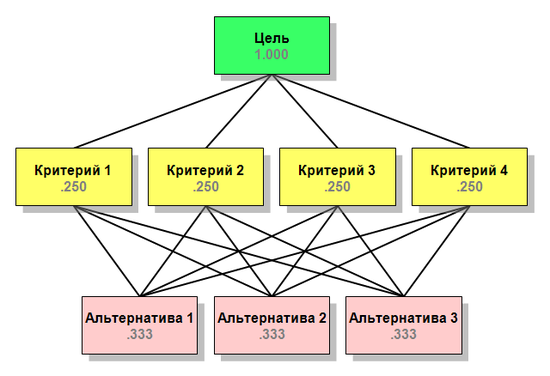


Рисунок 1.10.1 – Простейшая иерархическая структура МАИ с приоритетами, определенными по умолчанию

В соответствии с рисунком 1.10.1 показана иерархия, в которой приоритеты всех элементов не устанавливались ЛПР. В таком случае по умолчанию приоритеты элементов считаются одинаковыми, то есть все четыре критерия имеют равную важность с точки зрения цели, а приоритеты всех альтернатив равны по всем критериям. Другими словами, альтернативы в этом примере неразличимы. Заметим, что сумма приоритетов элементов любого уровня, равна единице. Если бы альтернатив было две, то их приоритеты были бы равны 0.500, если бы критериев было 5, то приоритет каждого был бы равен 0.200. В этом простом примере приоритеты альтернатив по разным критериям могут не совпадать, что обычно и бывает на практике. Приведем пример, в котором локальные приоритеты альтернатив по разным критериям не совпадают. В соответствии с рисунком 1.10.2 глобальные приоритеты альтернатив относительно цели вычисляются путём умножения локального приоритета каждой альтернативы на приоритет каждого критерия и суммирования по всем критериям.

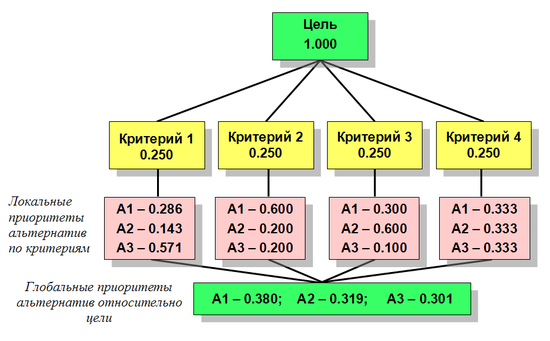


Рисунок 1.10.2 – Более сложная иерархическая структура, содержащая глобальные и локальные значения приоритетов по умолчанию

Если приоритеты критериев изменятся, то изменятся значения глобальных приоритетов альтернатив, следовательно, может измениться их порядок. В соответствии с рисунком 1.10.3 показано решение данной задачи с изменившимися значениями приоритетов критериев, при этом наиболее предпочтительной альтернативой становится A3.

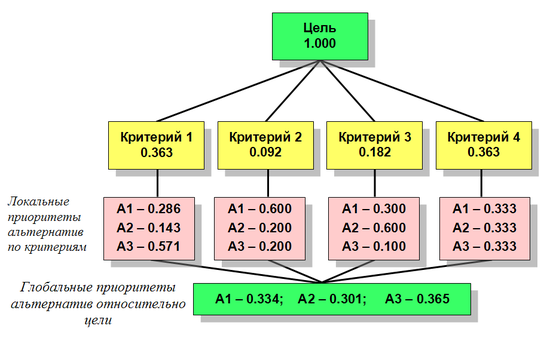


Рисунок 1.10.3 – Изменившиеся значения приоритетов критериев

1.11 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.

2 Публичные API

2.1 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.

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.

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 2.1.1 данные должны быть введены правильно и не содержать иных знаков.

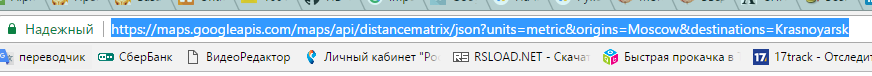


Рисунок 2.1.1 – Запрос в браузерной строке

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"

}

В соответствии с рисунком 2.1.2 изображены запрос и полученный ответ в браузере.



Рисунок 2.1.2 – Ответ, полученный от запроса

Из полученного ответа следует, что расстояние между городами составляет 4153 км, эта информация отображена в элементе *distance* со значением *"text" : "4 153 км"*. А также из полученного ответа показана продолжительность пути *duration* со значение *"text" : "2 дн. 6 ч."*.

Пример получения информации о пользователе с сервиса api.github.

Структура запроса:

<domain>/<endpoint>/<id>

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

Посылаем запрос на ресурс и ожидаем ответа.

Ответ с ресурс api.github:

*{*

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

*"id": 22838877,*

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

*"gravatar\_id": "",*

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

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

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

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

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

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

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

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

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

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

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

*"type": "User",*

*"site\_admin": false,*

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

*"company": null,*

*"blog": "",*

*"location": "Krasnoyarsk region",*

*"email": null,*

*"hireable": null,*

*"bio": null,*

*"public\_repos": 1,*

*"public\_gists": 0,*

*"followers": 0,*

*"following": 0,*

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

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

*}*

Из данных полученных от запроса, можно увидеть небходимую информацию, допустим такую как *"public\_repos": 1*, которая говорит о имеющемся репозитории в количестве равной одному. Изменим данные о пользователе в соответствии с рисунком 2.1.4 и добавим ещё один репозиторий и отправим запрос снова. В соответствии с рисунком 2.1.3. отображена практически вся информация с запроса. В данный момент видно, о наличии одного репозитория с именем *dplm*.

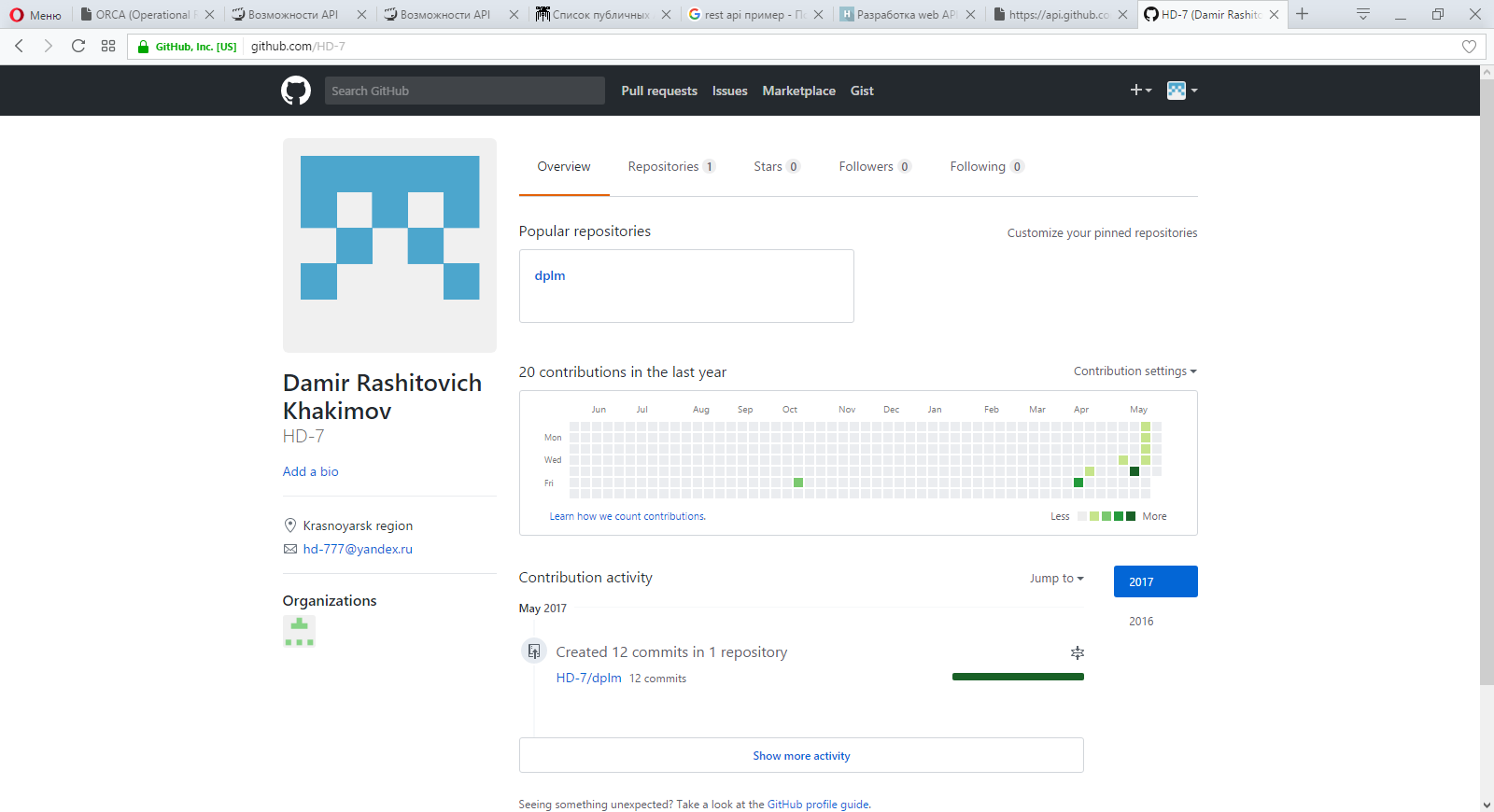


Рисунок 2.1.3 – Данные о пользователе HD-7 на github.

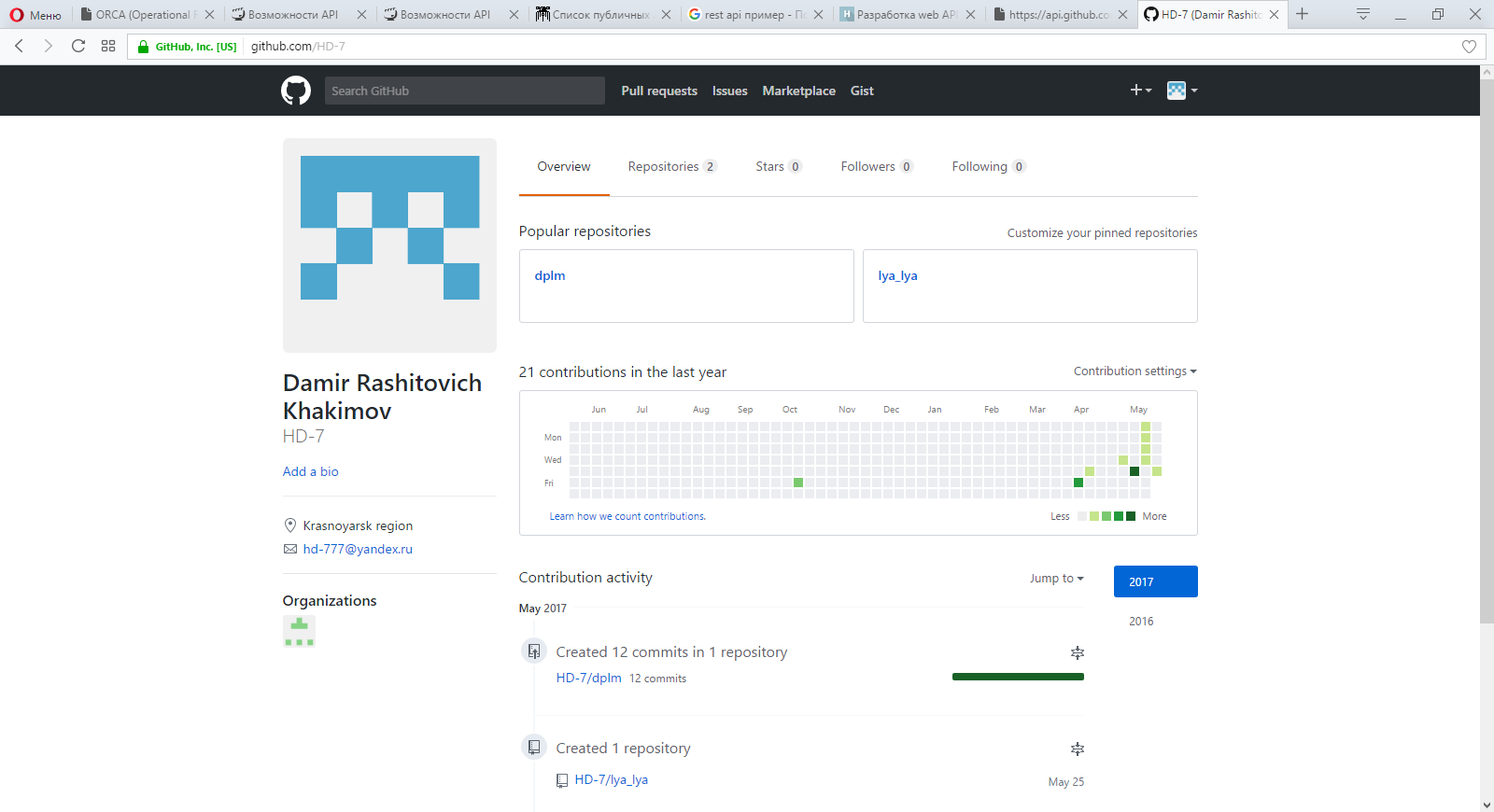


Рисунок 2.1.4– Наличие двух репозиториев.

Повтор запроса о пользователе. Ответ:

*{*

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

*"id": 22838877,*

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

*"gravatar\_id": "",*

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

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

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

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

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

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

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

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

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

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

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

*"type": "User",*

*"site\_admin": false,*

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

*"company": null,*

*"blog": "",*

*"location": "Krasnoyarsk region",*

*"email": null,*

*"hireable": null,*

*"bio": null,*

*"public\_repos": 2,*

*"public\_gists": 0,*

*"followers": 0,*

*"following": 0,*

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

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

*}*

Из данного ответа видно, что *"public\_repos": 2* изменилось, значит запрос работает должным образом, в соответствии с рисунком 2.1.4 указано два репозитория, *dplm* и *lya\_lya*. Так же можно получить данные о пользователе, которые находятся в публичном доступе. Допустим *"email": null* принимает значение *null* в том случае, если нет данных либо это информация скрыта и имеет значение приватности.

3 Выбор СУБД

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

Проект POSTGRES, под руководством профессора Майкла Стоунбрейкера (Michael Stonebraker), был спонсирован Агентством Расширенных Оборонных Исследовательских Проектов (Defense Advanced Research Projects Agency (DARPA)), Офисом Армейских Исследований (Army Research Office (ARO)), Национальным Научным Фондом (National Science Foundation (NSF)), а также ESL, Inc. Реализация реляционной СУБД POSTGRES началась в 1986. Начальные концепции для этой системы были представлены в The design of POSTGRES , а определение начальной модели данных было осуществлено в The POSTGRES data model . Устройство системы правил на тот момент, было описано в The design of the POSTGRES rules system. Обоснование архитектуры и менеджеры хранения были детально описаны в The design of the POSTGRES storage system .

Затем вышло несколько версий Postgres. Первая "demoware" система заработала в 1987 и была продемонстрирована в 1988 на Конференции ACM-SIGMOD. Версия 1, описанная в The implementation of POSTGRES была выпущена в июне 1989 года и могла работать с несколькими внешними пользователями. В ответ на критику первого варианта системы правил, был сделан следующий вариант (вариант A commentary on the POSTGRES rules system ) был переделан как ( On Rules, Procedures, Caching and Views in Database Systems ) и Версия 2, выпущенная в Июне 1990 года была основана на новой системе правил. Версия 3 выпущенная в 1991, включала в себя поддержку нескольких менеджеров хранения, улучшенный обработчик запросов и вновь переписанную систему правил. Большинство следующих версий до появления Postgres95 (см. ниже) были сфокусированы на вопросах переносимости и стабильности.

СУБД POSTGRES была использована для реализации многих различных исследований и написания приложений. Сюда вошли: система анализа финансовых данных, пакет мониторинга производительности реактивных двигателей, база данных перемещений астероидов, база данных медицинской информации и несколько географических информационных систем. СУБД POSTGRES также использовлась как средство обучения в нескольких университетах. Наконец компания Illustra Information Technologies (позднее влившаяся в компанию Informix, которой теперь владеет IBM) взяла код этой СУБД и коммерциализировала его. POSTGRES стал приоритетным менеджером данных для проекта научных вычислений Sequoia 2000 после 1992 года.

Размер сообщества пользователей этого продукта удвоился в 1993 году. Стало весьма очевидно, что обслуживание прототипа кода и его поддержка занимают гораздо больше времени, чем сами исследования в области баз данных. Пытаясь снизить нагрузку, связанную с поддержкой, проект Беркли POSTGRES официально прекратил своё существование с выходом версии 4.2.

Postgres95. В 1994, Эндрю Ю (Andrew Yu) и Джолли Чен (Jolly Chen) добавили в POSTGRES интерпретатор языка SQL. Затем Postgres95 был выложен в Интернет, чтобы найти свой собственный путь в мире продуктов с открытым исходным кодом, как потомок, основанный на оригинальном коде Беркли POSTGRES.

Postgres95 был полностью приведён к стандарту ANSI C и сократил свой размер на 25%. Были внесены многие внутренние изменения, которые увеличили производительность и обслуживаемость кода. Postgres95 версий 1.0.x был быстрее на 30-50% согласно Wisconsin Benchmark по сравнению с POSTGRES, Version 4.2. За исключением исправления ошибок, были сделаны следующие серьёзные расширения:

* язык запросов PostQUEL был заменен на SQL (реализованный в этом сервере). Подзапросы не поддерживались вплоть до выхода PostgreSQL (см. ниже), но в Postgres95 их можно было иммитировать с помощью функций SQL, определяемых пользователем. Агрегаты были переписаны. Также в запросы была добавлена поддержка GROUP BY;
* для интерактивных SQL запросов была предоставлена новая программа (psql), которая использовала библиотеку GNU Readline. Она вытеснила старую программу monitor;
* была создана новая front-end библиотека, libpgtcl, поддерживающая клиентов, основанных на Tcl. Простая оболочка pgtclsh, предоставила новые команды Tcl для обеспечения взаимодействия Tcl программ и Postgres95;
* была тщательно пересмотрена работа с большими объектами. Инверсионные большие объекты представляли собой только механизм для хранения больших объектов. (Инверсионная файловая система была удалена);
* была удалена instance-level система правил. Правила стали доступны как переписываемые правила;
* вместе с исходным кодом стал поставляться краткий учебник по особенностям работы с SQL в Postgres95;
* для построения проекта стал использоваться GNU make (вместо BSD make). Также, Postgres95 был скомпилирован со стандартной версией GCC (выравнивание данных типа double было исправлено).

PostgreSQL. В 1996 году было решено, что имя "Postgres95" не соответствует настоящему времени. Мы выбрали новое имя PostgreSQL чтобы подчеркнуть отличие от оригинального POSTGRES и выход множества версий с поддержкой SQL. В тоже время, мы установили нумерацию версий начиная с 6.0, вернувшись обратно к нумерации, которую начали в проекте Беркли POSTGRES.

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.

3.1 PostgreSQL

PostgreSQL — свободная объектно-реляционная система управления базами данных (СУБД). Существует в реализациях для множества UNIX-подобных платформ, включая AIX, различные BSD-системы, HP-UX, IRIX, Linux, macOS, Solaris/OpenSolaris, Tru64, QNX, а также для Microsoft Windows.

Объектно-реляционная СУБД, в настоящее время известная как PostgreSQL ведёт свое происхождение от пакета POSTGRES, который был написан в департаменте Беркли, Калифорнийского Университета. Более чем двадцатилетняя разработка PostgreSQL сделала этот продукт одной из наиболее продвинутых СУБД с открытым исходным кодом.

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.

Стоунбрейкер и его студенты разрабатывали новую СУБД в течение восьми лет с 1986 по 1994 год. За этот период в синтаксис были введены процедуры, правила, пользовательские типы и другие компоненты. В 1995 году разработка снова разделилась: Стоунбрейкер использовал полученный опыт в создании коммерческой СУБД Illustra, продвигаемой его собственной одноимённой компанией (приобретённой впоследствии компанией Informix), а его студенты разработали новую версию Postgres — Postgres95, в которой язык запросов POSTQUEL — наследие Ingres — был заменен на SQL.

Разработка Postgres95 была выведена за пределы университета и передана команде энтузиастов. Новая СУБД получила имя, под которым она известна и развивается в текущий момент — PostgreSQL.

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.

В данной дипломной работе использована PostgreSQL-9.6.3-1. В соответствии с рисунком 3.1.1 запущено окно с установкой и дальнейшей настройкой СУБД.



Рисунок 3.1.1 – Начальное окно установщика

После указания директории установки каталога хранения БД и пользовательский файлов в соответствии с рисунком 3.1.2 необходимо установить пароль на данную БД, самое главное не забыть пароль. В этом случае пароль указан с наличием одних лишь цифр и является следующим 1814536.

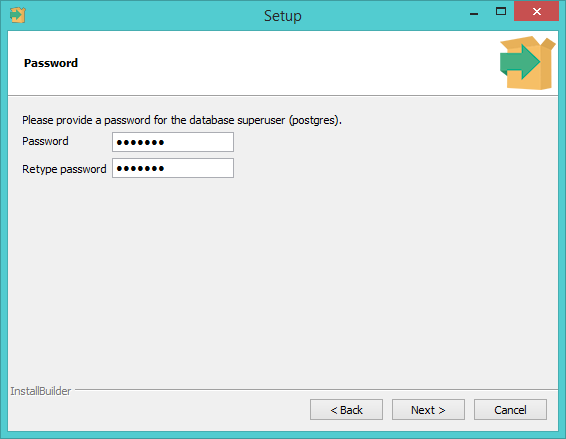


Рисунок 3.1.2 – Установка пароля для БД

В следующем окне в соответствии с рисунком 3.1.3 идёт указание порта для данной БД, по стандарту порт является 5432 и таким и останется, нет необходимости, пока что изменять на другой.

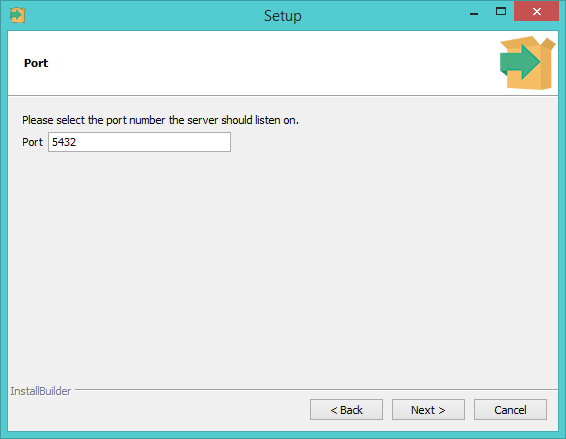


Рисунок 3.1.3 – Установка своего порта для БД

Далее в соответствии с рисунком 3.1.4 необходимо указать локализацию данной БД, и она следующая Russian, Russia.

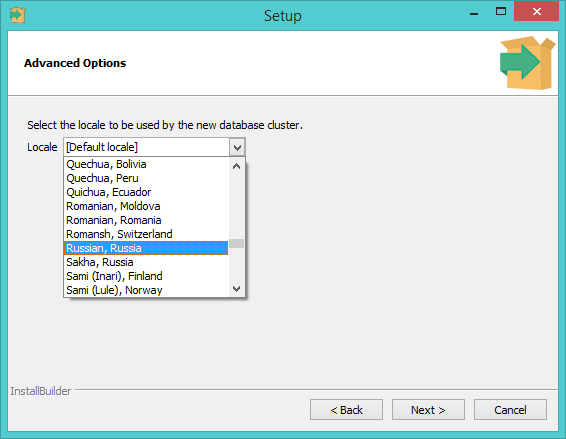


Рисунок 3.1.4 – Указание локализации своей БД

После указания всех нужных данных в соответствии с рисунком 3.1.5 идёт процесс установки.

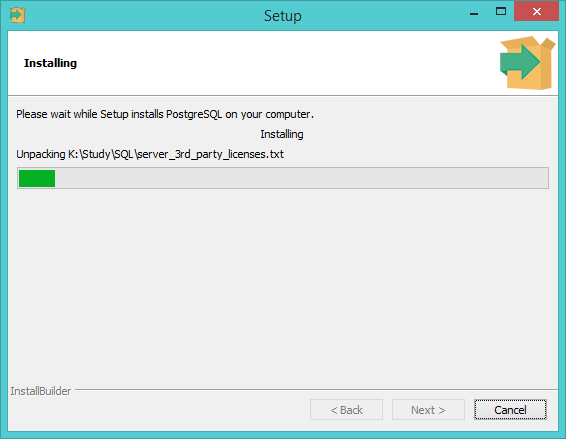


Рисунок 3.1.5 – Процесс установки

Следующий этап, настройка и установка необходимых приложений. В соответствии с рисунком 3.1.6 выбрать соответствующую БД с указанным портом.

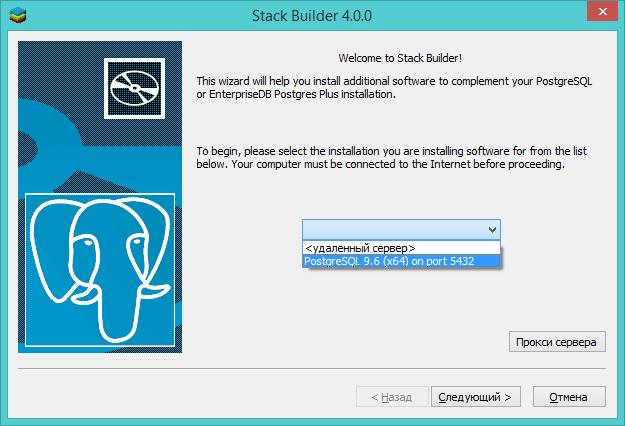


Рисунок 3.1.6 – Окно установщика дополнительных программ

Далее необходимо указать какую категорию дополнительных программ нужно установить для данной БД. В соответствии с рисунком 3.1.7 изображены категории, выбор только самых необходимых. И все выбранные категории в соответствии с рисунком 3.1.8 будут загружены, далее в соответствии с рисунком 3.1.9 указан процесс загрузки этих компонентов.

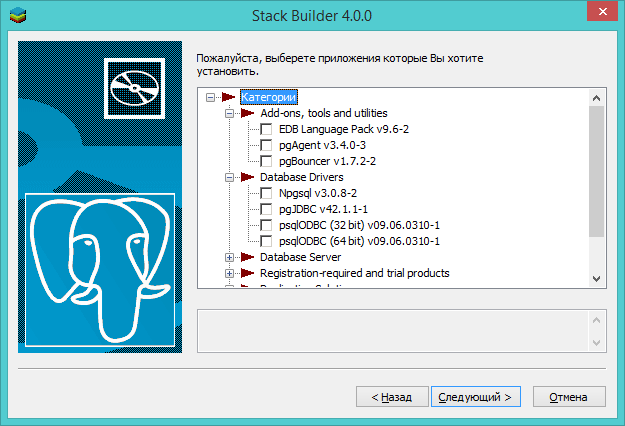


Рисунок 3.1.7 – Выбор категорий для установки

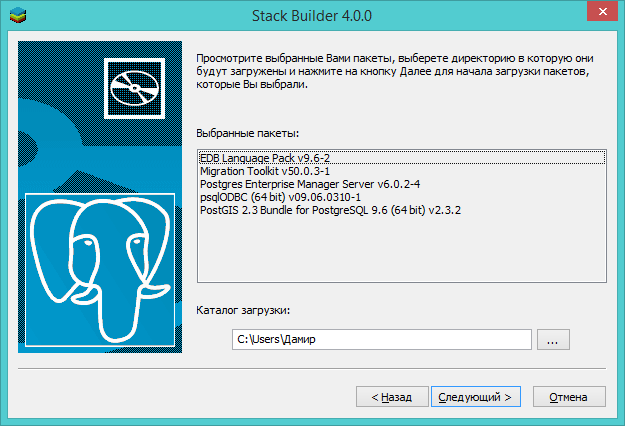


Рисунок 3.1.8 – Выбраны для загрузки необходимые пакеты

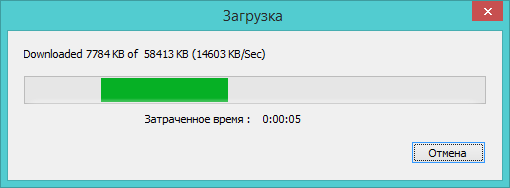


Рисунок 3.1.9 – Процесс загрузки

После загрузки в соответствии с рисунком 3.1.10 выйдет окно о завершении и предложено установить эти компоненты.

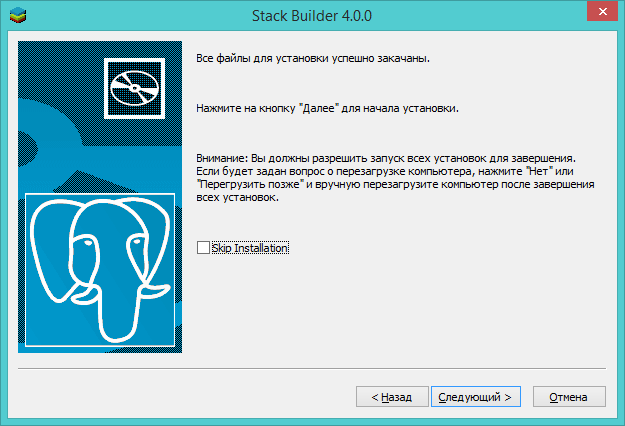


Рисунок 3.1.10 – Завершение загрузки

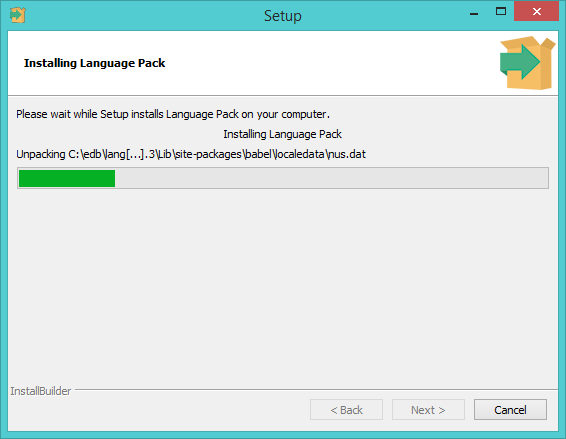


Рисунок 3.1.11 – Процесс установки компонента

В pgAdmin происходит создание базы данных, в соответствии с рисунком 3.1.13 бд называется question\_answer (вопрос - ответ).

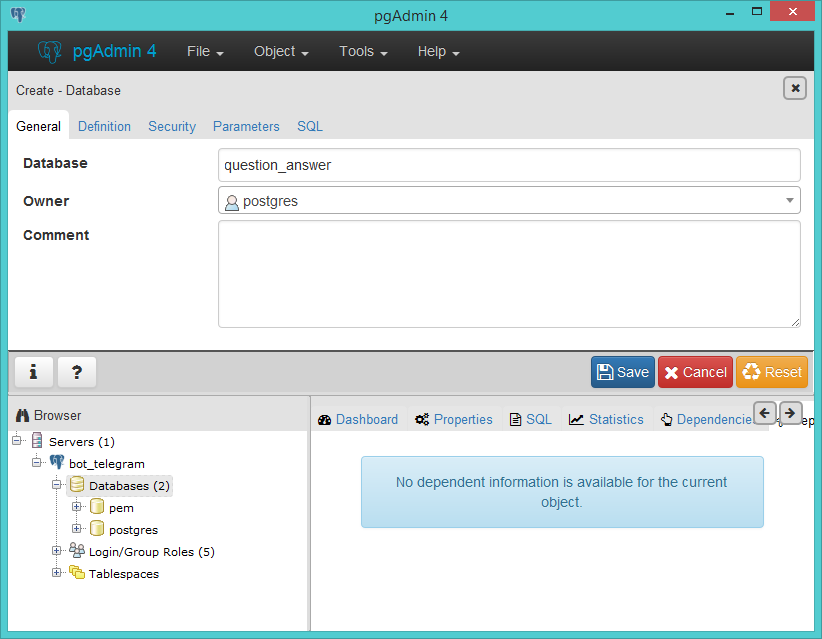


Рисунок 3.1.13 – Создание базы данных

Для работы необходимо создать две простые таблицы, одна будет хранить вопросы имя этой таблицы (question), другая будет содержать ответы (answer). С помощью команды *CREATE* создаются эти самые таблицы с указанием типов данных.

Таблица «вопрос».

*CREATE TABLE question (*

*id\_question integer,*

*question text*

*);*

Таблица «ответ».

CREATE TABLE answer (

id\_answer integer,

answer text

);

Добавление данных в таблицу осуществляется с помощью команды *INSERT* или *COPY* (для большого количества записей за раз). Используется команда *INSERT* для добавления записей в таблицу «вопрос».

INSERT INTO question (id\_question, question) VALUES

(1, 'Как дела?'),

(2, 'Какая сейчас погода?'),

(3, 'Кто ты?');

Проверка занесённых данных в таблицу выполняется командой *SELECT \* FROM public.question*, в соответствии с рисунком 3.1.14 отображена информация о всех записях в таблице question.

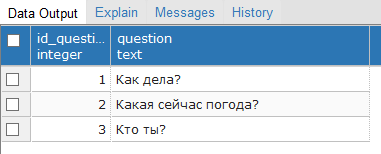


Рисунок 3.1.14 – Данные в таблице question

4 Среда 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.

4.1 Причины использования Python

4.1.1 Качество программного обеспечения

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

4.1.2 Высокая скорость разработки

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

4.1.3 Переносимость программ

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

4.1.4 Библиотеки поддержки

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

4.1.5 Интеграция компонентов

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

4.2 История создания

Разработка языка Python была начата в конце 1980-х годов сотрудни-ком 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 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

4.4 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.

4.5 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, Qual-comm 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.

4.6 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.

4.7 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.

4.8 Недостатки

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

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

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

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

5 Основные причины выбора мессенджера Telegram

1. Скорость работы. В telegram скорость отправки сообщений моментальная, даже при низкой скорости интернета. Все передается за секунды, да и сервера падают крайне редко. Это очень большой плюс для данного мессенджера.
2. Дизайн. Он очень эстетичный, все подогнано под один стиль, ни что не выделяется. По канонам Material Design, быстро и в минимальное количество кликов. Так же плюсом является размер приложения, он меньше чем у Whatsapp, Viber and etc.



Рисунок 5.1 – Скриншот дизайна мессенджера

1. Всеядность. Абсолютно везде применим данный мессенджер – Android, iOS, Windows, Mac OS и даже Web-клиент. Причем работает одинаково хорошо во всех платформах.
2. Облачность. Все переписки, фото, видео, хранятся в облаке. Все данные сжаты(урезаны), но они не потеряются. Это конечно не ново, но реализовано лучше, чем у WhatsApp, где вас спрашивают сохранять или нет. В данном случае, здесь всё реализовано по стандарту и ничего не пропадёт.
3. Безопасность. Секретные чаты с авто удалением, пароли на вход и двухэтапная авторизация. То есть, он уже более защищен чем остальные, и это может стать решающим фактором для многих, но работает все как часы, пароль никак не обойти, двухэтапная авторизация при попытке взлома.
4. Боты, каналы и стикеры.

Возможность создания бота с помощью клиента мессенджера, которого можно использовать по своему усмотрению. Создание большого количества каналов для общения и делиться информацией в виде массовой рассылки. Ещё огромное количество разнообразных стикеров, которые доступны всем бесплатно, а также можно добавлять свои стикеры в коллекцию.

Для реализации цели данной магистерской работы, является возможность создания бота в мессенджере telegram.

5.1 Telegram (мессенджер)

Telegram — бесплатный кроссплатформенный мессенджер для смартфонов и других устройств, позволяющий обмениваться текстовыми сообщениями и медиафайлами различных форматов. Используются проприетарная серверная часть c закрытым кодом, работающая на мощностях нескольких компаний США и Германии, финансируемых Павлом Дуровым в объёме порядка 12 млн долларов США ежегодно, и несколько клиентов с открытым исходным 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.

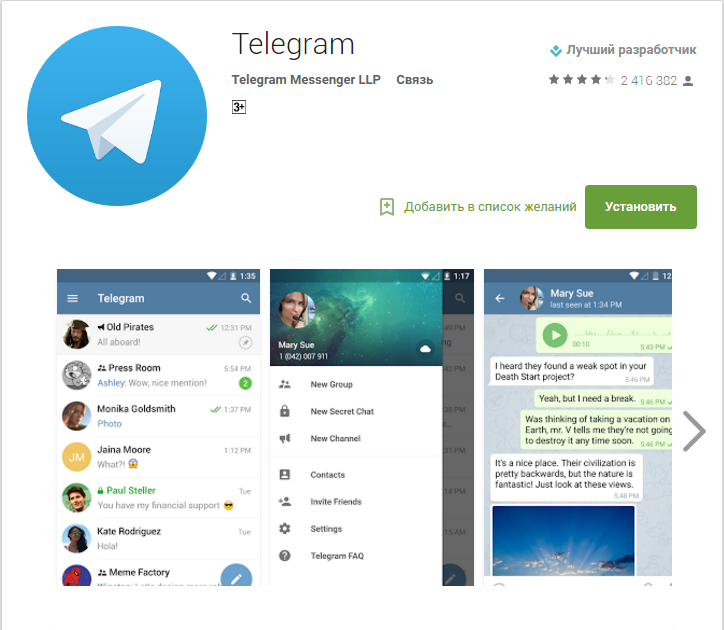


Рисунок 5.1.1 – Скриншот Telegram для Android

5.2 История

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.

5.3 Технология

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.

5.4 Боты

При помощи специального API сторонние разработчики могут создавать «ботов», специальные аккаунты, управляемые программами. Типичные боты отвечают на специальные команды в персональных и групповых чатах, также они могут осуществлять поиск в интернете или выполнять иные задачи, применяются в развлекательных целях или в бизнесе.

В сентябре 2015 года Павел Дуров заявил о скором появлении возможностей монетизации и размещения рекламы в ботах.

5.5 Конкурсы поиска уязвимостей

В декабре 2013 года Павел Дуров объявил конкурс до 1 марта 2014 года на «взлом» защиты Telegram с призовым фондом в 200 тысяч долларов. Условиями конкурса ставилась задача расшифровать личную переписку Павла с его братом Николаем через «секретные чаты», используя зашифрованные данные, которыми обменивались приложения и сервер. В их сообщениях, отправляемых ежедневно, содержался секретный адрес электронной почты, расшифровка которого позволила бы получить приз.

Требуемая для такого «взлома» модель атаки, атака на основе шифротекста, является самой слабой и, в то же время, наиболее сложной и неудобной для криптоаналитика. Существуют чрезвычайно слабые алгоритмы, которые могут быть устойчивыми в данной модели, но уязвимыми для других методов. Обычно при анализе новых криптографических алгоритмов используются более сильные модели атаки, в которых атакующему может быть известен текст до шифрования, дана возможность отправлять на шифрование любые тексты или возможность изменения данных, пересылаемых по сети. Таким образом, в случае, если конкурс никто не выиграет, это не докажет криптографическую безопасность протокола.

23 декабря 2013 года, всего через несколько дней после начала конкурса, пользователь «Хабрахабра», не являющийся экспертом в криптографии, обнаружил уязвимость, заключавшуюся в том, что клиент получал параметры для генерации ключей DH (константы для определения поля вычетов) от сервера без проверки, благодаря чему проприетарный сервер MTProto мог передать некорректные параметры, не обеспечивающие криптографическую стойкость, и скрытно провести MITM-атаку на секретные чаты. Так как ему не удалось прочитать переписку, размер выигрыша составил только 100 тысяч долларов. После этого клиент был обновлён, в нём была добавлена проверка параметров, получаемых от сервера, с целью значительно уменьшить вероятность подобной атаки.

В ноябре 2014 года был организован новый трёхмесячный конкурс, в котором модель атаки была расширена, у атакующего появилась возможность выступать в качестве сервера MTProto, изменяя пересылаемые данные. По условиям конкурса требуется взломать «секретный чат», при этом участники чата проводят сверку ключей, согласованных при открытии чата, по независимым каналам связи.

По оценкам исследователя Moxie Marlinspike и других подобные конкурсы не могут доказать безопасность шифрования и лишь вводят в заблуждение. Отсутствие выигравших не означает безопасности продукта, многие такие конкурсы в целом нечестны, анализ не контролируется и проводится случайными людьми, а вознаграждения зачастую слишком малы, чтобы оправдать многолетнюю работу нескольких компетентных криптоаналитиков.

5.6 Критика

Учётные записи пользователей привязываются к телефонным номерам, что является одним из самых существенных аргументов критиков Telegram, поскольку это не обеспечивает полной анонимности при общении. При регистрации в сервисе и последующих авторизациях новых устройств, производится проверка телефонного номера через отправку SMS-сообщения с кодом (на некоторых ОС — перехватывается приложением) или телефонный вызов.

Основатель WhatsApp Ян Кум указал в комментарии для Cossa.ru, что идеи, реализованные в его приложении, используются в Telegram.

Серверы Telegram не сохраняют сообщения из секретных чатов, но сохраняют историю обычных чатов и содержимое адресной книги пользователей на срок использования сервиса и на срок неактивности, указанный в настройках аккаунта (от одного месяца до года). Используемое в мессенджере шифрование не обеспечивает PFS во всех случаях.

Официальные клиенты Telegram по умолчанию активно рассылают всем контактам метаинформацию об открытии и закрытии приложения, при этом подписаться на эту метаинформацию может любой пользователь. Для отключения подобной рассылки требуется изменить настройки аккаунта.

Также неоднократно высказывались сомнения в безопасности протокола MTProto.

Имеются сообщения, что мессенджером могут пользоваться различные террористические группы как для общения, так и для пропаганды. В частности, террористическая группировка ИГ (ИГИЛ) использовала Телеграм для распространения своих заявлений более чем 14 тысячам подписчиков в более чем 30 каналах на различных языках. Однако команда Telegram активно занимается поиском и дальнейшей блокировкой подобных каналов.

5.7 Цензура

Телеграм выборочно применял цензуру. В частности, некоторое время мессенджер использовался в Иране для распространения порнографии и сатирических комментариев о правительстве. Руководство Telegram ограничило деятельность некоторых ботов и запретила определённые наборы изображений-«стикеров» по запросу правительства Ирана. В то же время чаты Телеграма не подвергались цензуре. В октябре 2015 года Дуров заявил, что компания Telegram Messenger LLP отказалась помогать Ирану в слежке за гражданами и в цензуре, из-за чего работа приложения некоторое время блокировалась.

Мессенджер блокировался властями в некоторых регионах Китая, где он мог использоваться для координации антиправительственных выступлений.

6 Моделирование системы поддержки принятия решений при обращении к боту мессенджеру

Порядок выполнения действий бота мессенджера при запросе пользователя:

1. Определение корректности вопроса;
2. Обработка данного запроса;
3. Вывод результата пользователю.

Алгоритм может быть составлен по блочной схеме или по логической схеме. Блочная схема алгоритма представляет собой графическое построение операций и логических условий. Простейшие алгоритмы применялись с давних времен в математике. Смысл этого понимается как точное предписание выполнения в заданном порядке некоторой системы операции для решения задач определенного типа.

Под алгоритмом необходимо понимать последовательность математических, логических, технологических или вместе взятых операций, отличающихся детерминированностью, массовостью, направленностью и приводящих к решению всех задач данного класса за конечное число шагов.

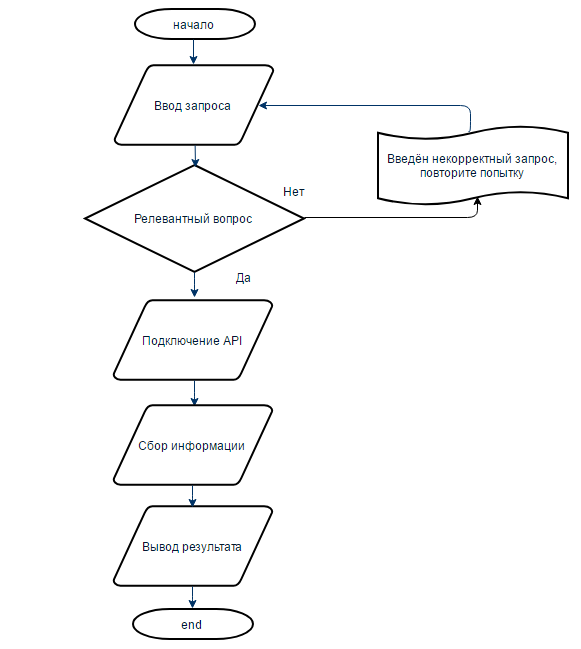


Рисунок 6.1 – Блок-схема алгоритма

Заключение

Системы поддержки принятия решений являются мощными инструментами, объединяющими научные методы поддержки сложных решений с методами, разработанными в области информатики, и становятся все более популярными во многих областях.

Они особенно ценны в ситуациях, когда объем доступной информации является неподъемным для интуиции невостребованного лица, принимающего решения, и в котором важны точность и оптимальность.

Системы поддержки принятия решений помогают человеческим когнитивным недостаткам путем интеграции различных источников информации, обеспечения интеллектуального доступа к соответствующим знаниям, содействия процессу структурирования и оптимизации решений.

Источники

1. <http://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BF%D0%BE%D0%B4%D0%B4%D0%B5%D1%80%D0%B6%D0%BA%D0%B8_%D0%BF%D1%80%D0%B8%D0%BD%D1%8F%D1%82%D0%B8%D1%8F_%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B9#.D0.9F.D1.80.D0.B5.D0.B8.D0.BC.D1.83.D1.89.D0.B5.D1.81.D1.82.D0.B2.D0.B0>
2. <http://bourabai.ru/tpoi/dss.htm#1>
3. <http://elar.urfu.ru/bitstream/10995/1676/5/1335843_schoolbook.pdf>
4. <http://pro-spo.ru/erp/1816-dss>
5. <https://www.youtube.com/watch?v=dDwTKNsrv54>
6. <http://mirznanii.com/a/111739/sozdanie-sistem-podderzhki-prinyatiya-resheniy>
7. <http://www.treko.ru/show_article_763>
8. <https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0_%D0%B8%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D0%B9>
9. <https://ru.wikipedia.org/wiki/Python>
10. <https://pythonworld.ru>
11. <https://habrahabr.ru/post/31180/>
12. <https://www.ibm.com/developerworks/ru/library/l-python_part_1/>
13. <https://ru.wikipedia.org/wiki/Telegram_(мессенджер)>
14. <http://telegram-free.ru/kak-iskat-kanaly-v-telegram.html>
15. <https://telegram-store.com/blog/anonymous-telegram/>
16. <https://telegram.org>
17. <https://keddr.com/2016/07/6-prichin-pochemu-telegram-top/>
18. <http://telegram-online.ru>
19. <https://ru.wikipedia.org/wiki/PostgreSQL>
20. <http://postgresql.ru.net/manual/history.html>
21. <http://elar.urfu.ru/bitstream/10995/1676/5/1335843_schoolbook.pdf>