МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет *компьютерных наук*

Кафедра *информационные технологии управления*

*Telegram Bot*

*Курсовая работа*

09.03.02 *Информационные системы и технологии*

*Информационные системы и технологии в управлении предприятием*

Допущен к защите

Зав. Кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_ *М. Г. Матвеев, д.т.н., профессор* \_\_.\_\_.20\_\_

Обучающиеся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*И. А. Фирсова, А. Г. Толчеева 3 курс, д/о*

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*, ст. преподаватель*

Воронеж 2018

Содержание

[1. Введение 3](#_Toc9766305)

[2. Анализ 4](#_Toc9766306)

[2.1. Анализ предметной области 4](#_Toc9766307)

[2.2. Анализ задач 4](#_Toc9766308)

[2.3. Анализ средств реализации 5](#_Toc9766309)

[2.4. Диаграммы 6](#_Toc9766310)

[**2.4.1. Диаграмма прецедентов 6**](#_Toc9766311)

[**2.4.2. Диаграмма классов 7**](#_Toc9766312)

[**2.4.3. Диаграмма объектов 8**](#_Toc9766313)

[**2.4.4. Диаграмма развертывания 8**](#_Toc9766314)

[**2.4.5. Диаграмма последовательности 9**](#_Toc9766315)

[**2.4.6. Диаграмма состояний 10**](#_Toc9766316)

[**2.4.7. Диаграмма активности 11**](#_Toc9766317)

[**2.4.8. Диаграмма коммуникации 12**](#_Toc9766318)

[2.5. Отчет по ролям 13](#_Toc9766319)

[2. Основная часть 14](#_Toc9766320)

[3. Тестирование 16](#_Toc9766321)

1. **Введение**

Чат-боты – это помощник, который общается с пользователями посредством сообщений и обладает множеством специфичных функций. Чатбота можно использовать как для рассылки информации, так и для ее сбора. На сегодняшний день мессенджеры пользуются большим спросом, это связано c изменением в области мобильного интернета: высокие скорости, низкая цена и широкое распространение смартфонов. Прогрессивность мессенджеров можно сравнить с явлением десятилетней давности – взрывом социальных медиа. Уже сейчас приложениями для обмена сообщениями пользуются 2 миллиарда человек, и если верить прогнозам, к 2021 году число пользователей увеличится до 2,48 миллиардов.

Изначально Telegram пользовался популярностью преимущественно у людей интеллектуальных профессий. Широкая публика уже успела распробовать WhatsApp и Viber, а новинка, у которой не было русскоязычной версии, оставалась на долю IT-специалистов и зарубежных стран — в основном развивающихся государствах, Италии, Испании и Бразилии.

AI (Artificial intelligence) используется в основном для сокращения издержек и повышения качества услуг. Наиболее популярными направлениями являются распознавание голоса и текстовое общение при помощи ботов. В качестве примера последнего можно привести знаменитые всплывающие окошки "консультантов" на многих сайтах. Конечно, о качестве таких консультаций можно поспорить, но тенденция налицо и от неѐ не собираются отказываться.

Примером автоматизации можно назвать огромное количество ботов для приема заявок на доставку еды, заказа столиков в ресторанах, рассылки рекламы и многое другое. Такие боты позволяют увеличить прибыль компаний, так как бот может обрабатывать заявки с большей скоростью, чем человек, и уменьшить нагрузку на рабочий персонал.

# 2. Анализ

## 2.1. Анализ предметной области

В современном мире люди хотят быстро получать нужную им информацию. Для многих сайтов уже разработаны боты для более удобной навигации по сайту.

Наш бот позволит пользователю в формате диалога получить интересующую информацию и сократить время поиска по сайту ВГУ.

Основным плюсом разработанного продукта является то, что он размещен в Telegram, что позволит пользователю в любой момент найти ответ на свой вопрос. Наиболее запрашиваемая информация уже содержится в боте в виде реализованных команд на основе быстрых ссылок.

## 2.2. Анализ задач

**Постановка задачи**

Основной целью создания Системы является предоставление пользователям удобного и бесплатного сервиса, позволяющего оперативно находить информацию и узнавать о новостях ВГУ.

**Задачи проекта**

* + - автоматизация процесса приёма обращений от пользователей;
    - разработка интерфейса чат-бота;
    - создание механизма информационного взаимодействия между чат-ботом и сайтом ВГУ;
    - демонстрация прототипа чат-бота, обеспечивающего:
      * приём обращения от пользователя;
      * интеграцию с существующей информацией;
      * обеспечение пользователю повсеместным доступом к информации – с любого мобильного устройства (смартфона), подключенного к сети Интернет.
    - ввод Системы в действие:
      * проведение испытаний Системы;
      * организация работы эксплуатационного персонала;
      * формирование отчетности по результатам испытаний.

**Система работает следующим образом:**

1. Для начала работы пользователь запускает бота. В ответ на запуск бот выдаёт ему список функций, которые он может выполнить.
2. Затем пользователь может выбрать:

* вопрос из шаблона:  
  Тогда бот отправит выбранный запрос в базу данных, получит на него ответ и покажет его клиенту.
* подписку на рассылку уведомлений:  
  В этом случае бот отправит этот запрос в базу данных и id клиента будет добавлено в базу для рассылок, что обеспечит отправку уведомлений о новых записях.
* отписку от рассылки:  
  Бот отправит в базу запрос об отписке и id пользователя удалится из базы для рассылок.
* возможность задать свой вопрос для получения развёрнутого ответа, которого нет в базе:  
  Бот передаст это сообщение менеджеру, который напишет ответ на вопрос и добавит его в базу данных. После этого бот пришлёт ответ пользователю (Обращение к пользователю осуществляется по id).

1. Помимо функций пользователь может завершить чат, тогда бот проведёт закрытие соединения.

## 2.3. Анализ средств реализации

На этапе проектирования для создания диаграмм и схем использовались следующие средства:

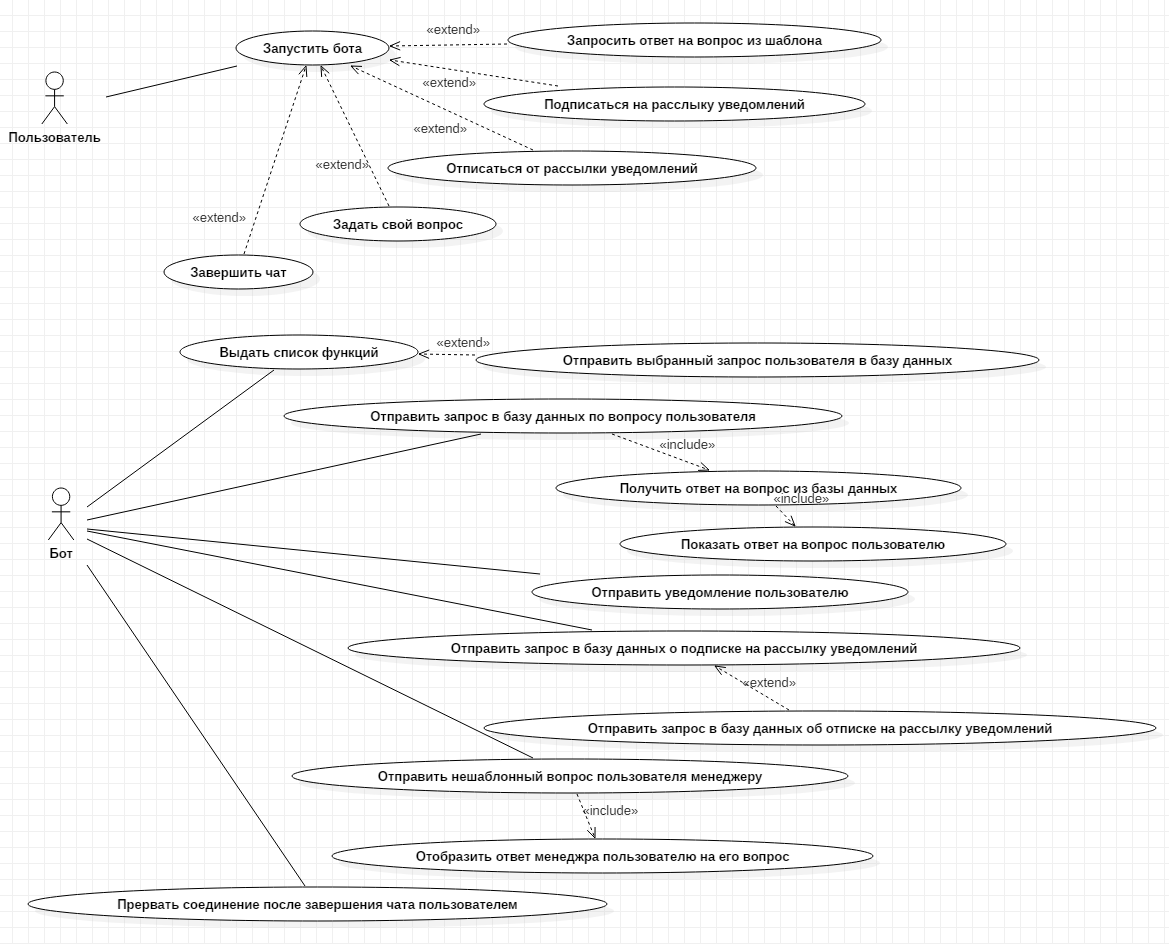
- StarUML;

- онлайн-сервис Draw.io;

В качестве языка программирования для реализации приложения был выбран язык Python, встроенные библиотеки pyTelegramBotAPI. СУБД – SQLite3.

**2.4. Диаграммы**

2.4.1. Диаграмма прецедентов



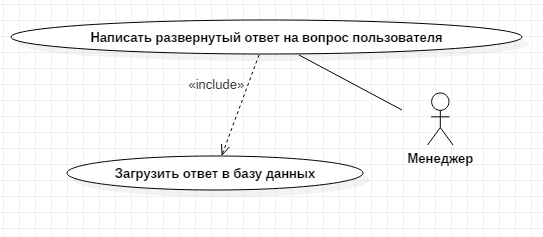


Рис. 1 Use-case диаграмма

2.4.2. Диаграмма классов

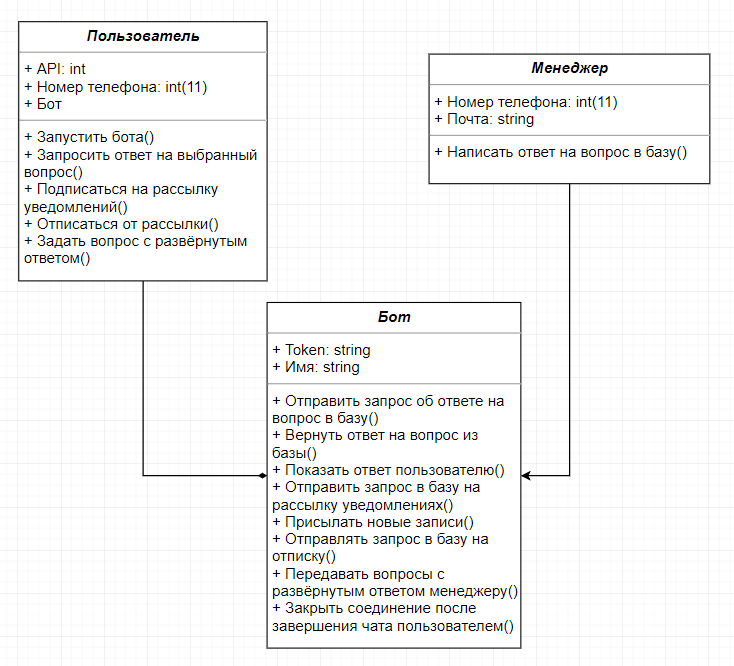


Рис. 2 Диаграмма классов

Описание предполагаемых классов:

Класс пользователь - пользователь может запустить бота, запросить ответ на выбранный вопрос по шаблону, подписаться или отписаться от рассылки, задать свой вопрос.

Класс бот- бот может отправить запрос, вернуть ответ из базы данных и показать его пользователю, отправить запрос в базу данных о подписки на рассылку и отписки, присылать новости каждый день, передать вопросы с развёрнутым ответом в базу данных и вернуть отвеет менеджера, закрыть соединение после завершения чата.

Класс Менеджер – менеджер отвечает на развёрнутые вопросы, отправляет новости с сайта каждый день в базу данных.

### 2.4.3. Диаграмма объектов



Рис. 3 Диаграмма объектов

### 2.4.4. Диаграмма развертывания



Рис. 4 Диаграмма развёртывания

### 

### 2.4.5. Диаграмма последовательности

Здесь приведена диаграмма последовательностей для ситуаций запуска бота, запроса на выбранный вопрос по шаблону, подписки на рассылку, отписки от рассылки, вопрос с развёрнутым ответом и завершение чата.

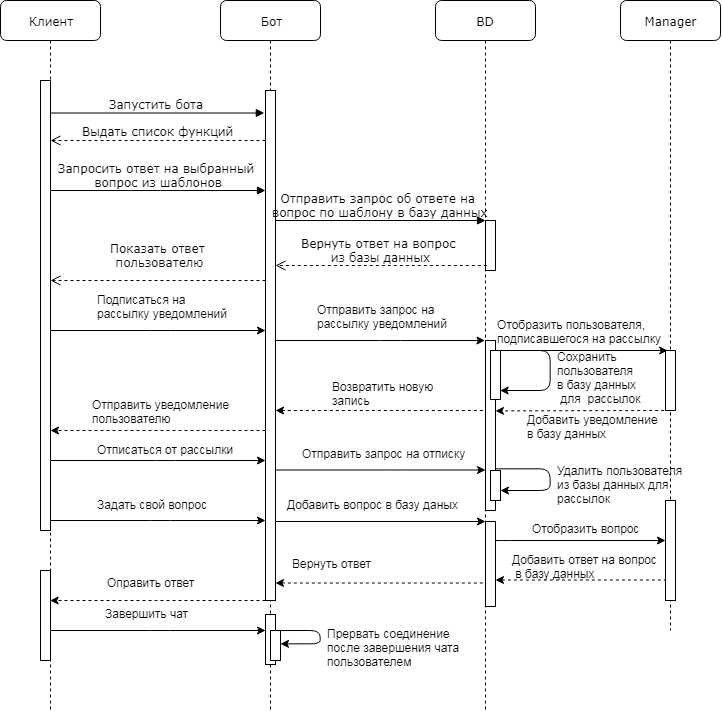


Рис. 5 Диаграмма последовательности

### 2.4.6. Диаграмма состояний

На диаграмме состояний приводится набор состояний бота при определённом выборе функций и последовательность переходов от одного состояния к другому.

### 

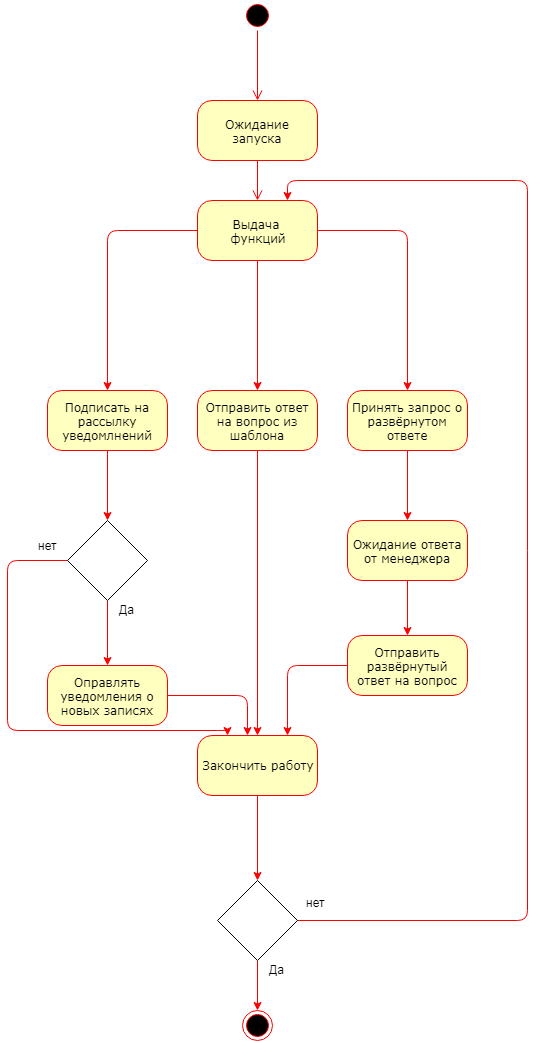


Рис. 6 Диаграмма состояний

### 2.4.7. Диаграмма активности

Диаграмма активности является расширением диаграммы состояния

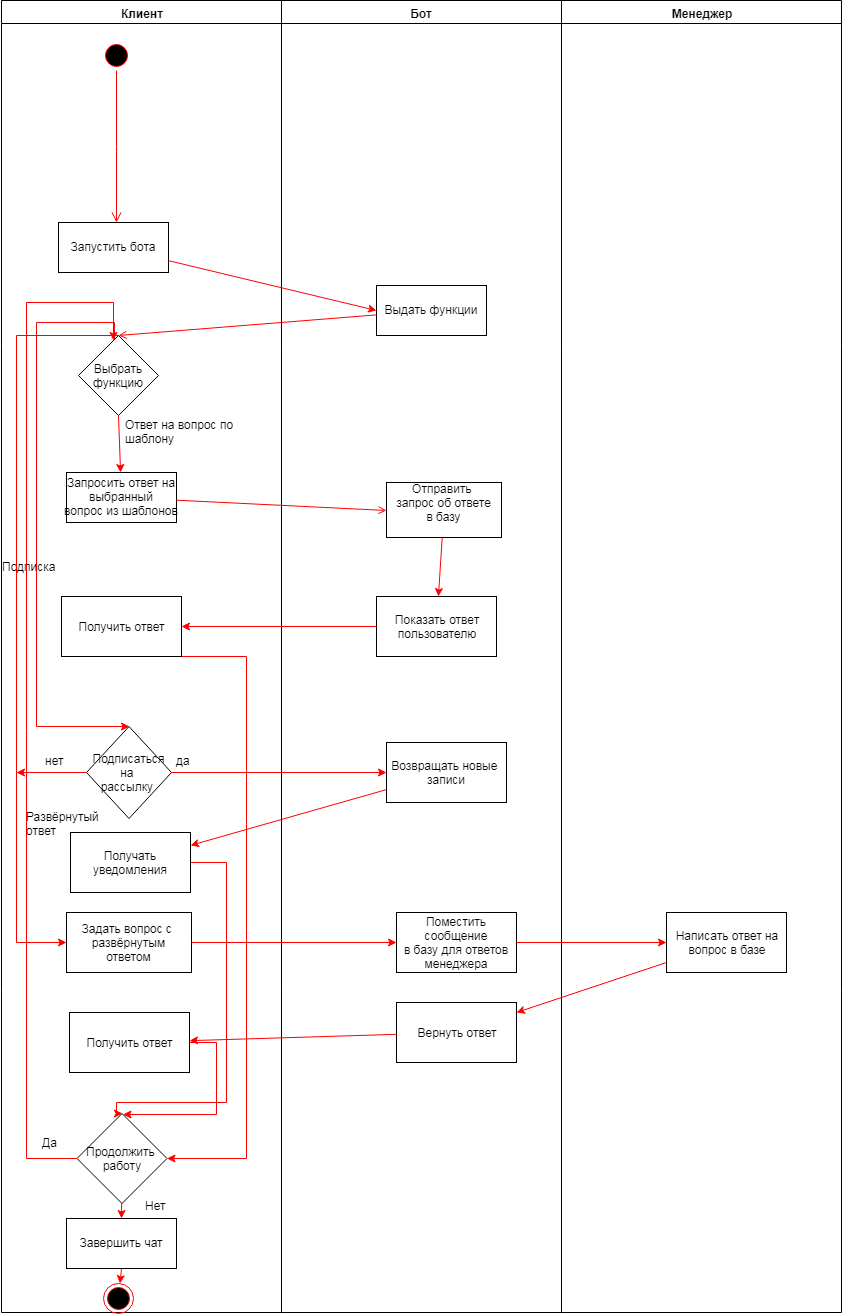


Рис. 7 Диаграмма активности

На диаграмме активности присутствуют 3 части (дорожки):

* Клиент (пользователь),
* Бот (отвечающий на функции),
* Менеджер(отвечающий на развёрнутые вопросы)

### 2.4.8. Диаграмма коммуникации

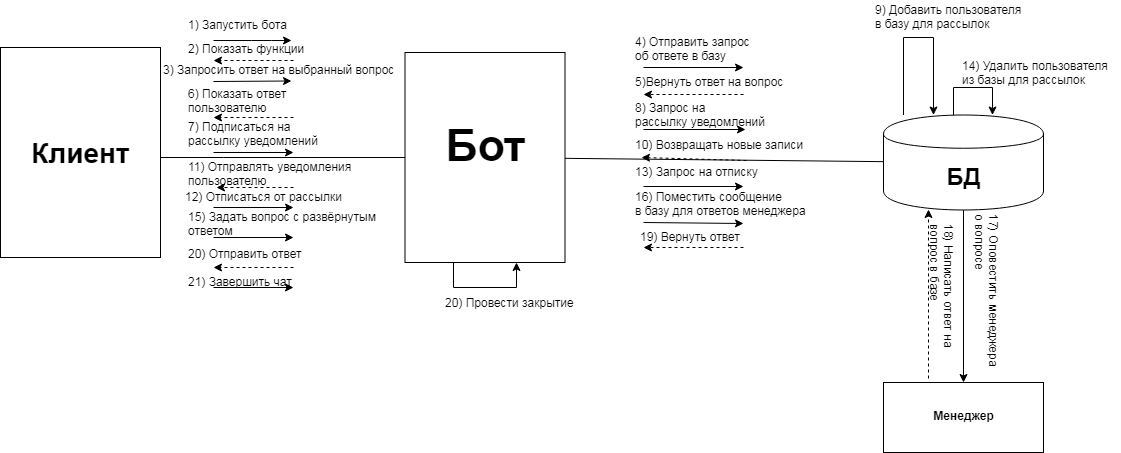


Рис. 8 Диаграмма коммуникаций

На диаграмме коммуникаций также приведена для ситуаций запуска бота, запроса на выбранный вопрос по шаблону, подписки на рассылку, отписки от рассылки, вопрос с развёрнутым ответом и завершение чата

**2.5. Отчет по ролям**

Фирсова Ирина:

* Написала первую часть ТЗ
* Спланировала список задач в Real Time Board
* Создала Backend и Frontend в Real Time Board
* Построила диаграммы (классов, последовательности, коммуникации, активности, развёртывания, состояния)
* Построила диаграмму Ганта

Толчеева Алёна:

* Написала вторую часть ТЗ
* Написала анализ предметной области с UseCase диаграммой
* Построила диаграммы (прецедентов, объектов, классов)

Выполнили командой:

* Ознакомились с задачей и придумали назначение бота
* Ознакомились с аналогами нашего бота
* Установили PyCharm, Anaconda и нужные библиотеки
* Начали строить скелет бота с его функциями

# Основная часть

Язык программирование Python

Python это язык программирования общего назначения, нацеленный в первую очередь на повышение продуктивности самого программиста, нежели кода, который он пишет. Говоря простым языком, на Python можно написать практически что угодно (веб-/настольные приложения, игры, скрипты по автоматизации, комплексные системы расчѐта, системы управления жизнеобеспечением и многое другое) без ощутимых проблем. Более того, порог вхождения низкий, а код во многом лаконичный и понятный даже тому, кто никогда на нѐм не писал. За счѐт простоты кода, дальнейшее сопровождение программ, написанных на Python, становится легче и приятнее по сравнению с Java или C++. А с точки зрения бизнеса это влечѐт за собой сокращение расходов и увеличение производительности труда сотрудников.

Несомненным достоинством является то, что интерпретатор Python реализован практически на всех платформах и операционных системах. Первым таким языком был Cи, однако его типы данных на разных машинах могли занимать разное количество памяти и это служило некоторым препятствием при написании действительно переносимой программы. Python же таким недостатком не обладает.

Так же, немаловажная черта - расширяемость языка, этому придается большое значение и, как пишет сам автор Гвидо ван Россум, язык был задуман именно как расширяемый. Это означает, что имеется возможность совершенствования языка всеми заинтересованными программистами. Интерпретатор написан на Си и исходный код доступен для любых манипуляций. В случае необходимости, можно вставить его в свою программу и использовать как встроенную оболочку. Или же, написав на Cи свои дополнения к Python и скомпилировав программу, получить "расширенный" интерпретатор с новыми возможностями.

Следующее достоинство - наличие большого числа подключаемых к программе модулей, обеспечивающих различные дополнительные возможности. Такие модули пишутся на Си и на самом Python и могут быть разработаны всеми достаточно квалифицированными программистами

Модуль TeleBot

Этот модуль является оболочкой над запросами к Telegram Bot API, используется для упрощения и минимизации написанного кода. Все типы определены в types.py. Все они полностью соответствуют определению типов API Telegram, за исключением from поля Message, которое переименовано в from\_user(поскольку from это зарезервированный токен Python). Таким образом, к таким атрибутам как message\_id можно обращаться напрямую, например: message.message\_id. Стоит обратить внимание, что атрибут message.chat может принадлежать как определенному пользователю, так и групповому чату.

Объект Message также имеет content\_type атрибут, который определяет тип сообщения. Атрибут content\_type может быть одним из следующих строк: text, audio, document, photo, sticker, video и так далее. Можно использовать несколько типов в одной функции. Все методы API расположены в классе TeleBot. Они переименованы, чтобы следовать общим соглашениям об именах Python. Например: sendMessage переименован в send\_message, editMessageText в edit\_message\_text .

Обработчик сообщений - это функция, украшенная декоратором экземпляра TeleBot. Обработчики сообщений состоят из одного или нескольких фильтров. Каждый фильтр возвращает True или False для определенного сообщения, и если возвращается True, то обработчик получает разрешение на обработку сообщения. Пример объявления обработчика сообщений приведен в листинге .

Модуль Requests Requests – библиотека Python, которая элегантно и просто выполняет HTTP-запросы (HyperText Transfer Protocol). В листинге приведен пример получения информации о веб странице через запрос.

import requests

resp = requests.get(‘htthps://api.github.com/events’)

После выполнения этого кода, в объекте resp хранится вся необходимая информация об этом объекте. Простой API Requests означает, что все формы HTTP запросов являются очевидными.

1. **Тестирование**

**Цели:**

1. Выявление дефектов на ранних стадиях.
2. Проверить, все ли требования были выполнены.
3. Убедиться, что приложение работает так, как ожидает пользователь.
4. Уверенность в качестве.

**Почему тестирование необходимо:**

1. Помогает избежать риски, так как мы прописываем условия, при которых могут возникнуть трудности.
2. Выявить дефекты, тем самым повысить качество продукта.

**Нами были проведены такие тестовые мероприятия:**

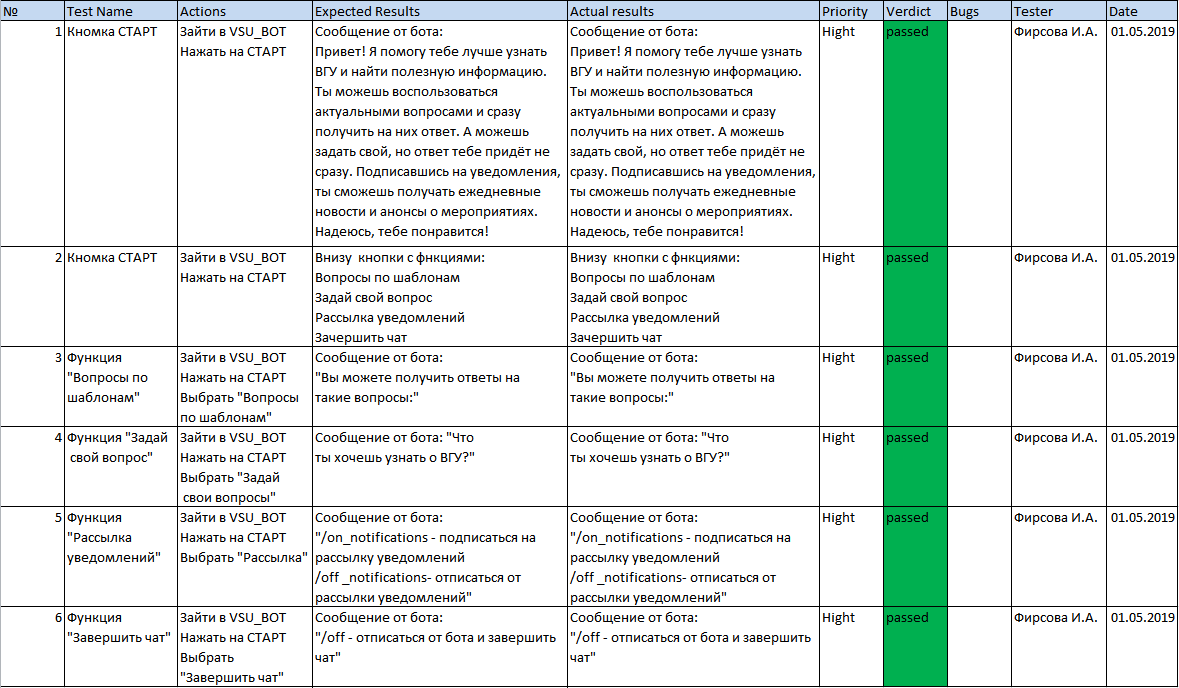
1. Планирование. Определить цели тестирования и подход для их достижения.
2. Контроль и управление. Составить план тестирования, что выполнено и будет выполнено на определённых стадиях разработки.
3. Тест Анализ. Рассмотреть особенности текст – кейсов, расставить приоритеты.
4. Тест дизайн. Разработать тест – кейсы. Прописать ожидаемые результаты.
5. Выполнение тестов. Протестировать приложение. Сравнить полученные результаты с ожидаемыми. Проверить, чтобы тесты соответствовали документации
6. Завершение тестов. Сбор данных после тестирования. Тест-report (доклады о тестах). Сохранить информацию для дальнейших проектов.

Так как нами была использована Цикличный жизненный цикл проекта, то тестирование будет тоже проходить по определённым итерациям.

На первом этапе нами был проведён **Functional testing:**

**Smoke testing.**

Тестирование охватывает основные функций программного обеспечения, но ни одну из них в глубине. Результат этого теста используется, чтобы решить, следует ли продолжить дальнейшее тестирование. Если испытание проходит успешно, продолжайте дальнейшие испытания. Если это не помогло, остановите дальнейшие тесты и попросите новую сборку с необходимыми исправлениями. Если приложение сильно сломано, детальное тестирование может быть пустой тратой времени и усилий.

****



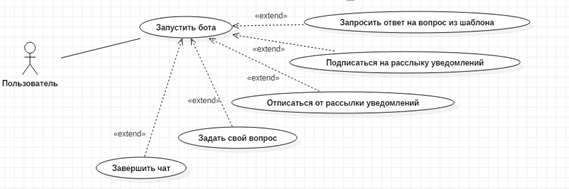








А так же **Test Design Techniques**: **Use-Case Testing**, чтобы проверить доступность всех необходимых пользователю функций. Мы использовали **Use-Case** и сверили все наши функции и запланированными.



Design Techniques бывает двух видов:

Static Test Design Techniques

1. Informal Reviews – пробежаться глазами по приложению, без документации о особых знаний о функционале.
2. Walkthroughs – Поочерёдное выполнение шагов для сбора информации.
3. Technical Review – Процесс обнаружения ошибок, дискуссия в команде и достижение единого мнения.
4. Inspection – Визуальная экспертиза, нарушение стандартов, несоответствие документации.

Dynamic Test Design

Black-Box. Основан на анализе соответствующей тестовой базы (например, документов формальных требований, спецификаций, вариантов использования, пользовательских историй или бизнес-процессов). Применимы как для функционального, так и нефункционального тестирования. Концентрируются на «входных» и «выходных» значениях.

White – Box. Основывается на внутренней структуре (код, архитектура, рабочие потоки и / или потоки данных в системе) или реализации системы. Тщательность тестирования белого ящика может быть измерена через структурное покрытие. Структурное покрытие - это степень, в которой определенный тип структурного элемента был испытан тестами, и выражается в процентах от типа охватываемого элемента.

Experience – based. Используют опыт разработчиков, тестировщиков и пользователей; для разработки, реализации и выполнения тестов. Часто сочетаются с методами белого и черного ящика.

Specification based (black - box)

* Equivalence Partitioning ( Эквивалентное разбиение) Классы эквивалентности делят данные на разделы таким образом, что все данные должны обрабатываться одинаково.

Любой раздел может быть разделен на подразделы, если требуется.

Каждое значение должно принадлежать одному и только одному разделу эквивалентности.

Когда недопустимые разделы эквивалентности используются в тестовых примерах, они должны тестироваться индивидуально, то есть не объединяться с другими недопустимыми разделами эквивалентности, чтобы гарантировать, что сбои не маскируются. Сбои могут быть замаскированы, когда происходит несколько сбоев одновременно, но виден только один из них, в результате чего другие сбои не обнаруживаются.

* Boundary Value Analysis. Анализ граничных значений является расширением эквивалентного разбиения, но может использоваться только при упорядочении раздела, состоящего из числовых или последовательных данных. Минимальные и максимальные значения (или первое и последнее значения) раздела являются его граничными значениями.
* Decision Table. Таблицы решений - хороший способ записать сложные бизнес-правила, которые должна реализовать система.

При создании таблиц решений тестер определяет условия (чаще input) и результирующие действия (чаще output) системы.

Они формируют строки таблицы, обычно с условиями вверху и действиями внизу.

Каждый столбец соответствует правилу принятия решения, которое определяет уникальную комбинацию условий, которая приводит к выполнению действий, связанных с этим правилом.

Значения условий и действий обычно отображаются как логические значения (true или false) или дискретные значения (например, красный , зеленый , синий), но также могут быть числами или диапазонами чисел.

Эти различные типы условий и действий объединены в одной таблице.

* Use-Case Testing. Тестирование вариантов использования. Вариант использования может включать в себя возможные варианты его основного поведения, включая исключительное поведение и обработку ошибок (ответ системы и восстановление после ошибок программирования, приложений и связи, например, приводящих к ошибке сообщение). Тесты предназначены для осуществления определенных поведений (базовых, исключительных или альтернативных и обработки ошибок). Охват может быть измерен как процент протестированного поведения вариантов использования, деленный на общее количество поведений вариантов использования, обычно выраженное в процентах

На втором этап мы воспользовались **Test Design Techniques** и **Decision Table**, чтобы проверить правильность ответов на вопросы по шаблонам.

