МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И. ВЕРНАДСКОГО» ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра компьютерной инженерии и моделирования

**Разработка ТЕЛеГРАММ-БОТА**

**"Reminder"**

Курсовая работа  
по дисциплине «Программирование»

студента 1 курса группы ПИ-б-о-202(2)

Абдураманова Ибраима Арсеновича

направления подготовки 09.03.04 «Программная инженерия»

(код и наименование)

|  |  |  |
| --- | --- | --- |
| старший преподаватель кафедры  компьютерной инженерии и моделирования |  | Тимофеева С. В. |
| (оценка) |
|  |
|  |
| (подпись, дата) |

Симферополь, 2021

# Реферат

Абдураманов И. А. Разработка Телеграмм бота «Reminder» // Курсовая работа (уровень бакалавриата) по специальности 09.03.04 Программная инженерия / Кафедра компьютерной инженерии и моделирования Физико-технического института Крымского федерального университета им. В. И. Вернадского. – Симферополь, 2021. – 23 с., 0 табл., 14 рис., 0 прил., 15 ист.

Объект разработки данной работы – Телеграмм-бот «Reminder», сервер для хранения заметок.

Цель работы – создание удобного Телеграмм-бота для напоминаний о предстоящих делах. Реализация клиентского приложения на языке программирования Python, с помощью библиотеки «Python Telegram Bot», которая очень хорошо справляется со своими задачами. Инструментом взаимодействия клиента с сервером используется библиотека «requests». Для хранения данных пользователя был написан сервер на языке C++, благодаря библиотеке «cpp\_httlib». Обмен данными происходит с помощью POST запросов.

ПРОГРАММИРОВАНИЕ, БОТ, КЛИЕНТ-СЕРВЕР, СИНХРОНИЗАЦИЯ

# Оглавление

[Реферат 2](#_Toc73659129)

[Оглавление 3](#_Toc73659130)

[Введение 4](#_Toc73659131)

[Глава 1 ПОСТАНОВКА ЗАДАЧИ 5](#_Toc73659132)

[1.1 Цель проекта 5](#_Toc73659133)

[1.2 Задачи и технологий реализации серверной части 5](#_Toc73659134)

[1.2.1 Форматы данных JSON 5](#_Toc73659135)

[Глава 2 Разработка Телеграмм-Бота 9](#_Toc73659136)

[2.1 Логика работы бота 9](#_Toc73659137)

[2.1.2 Анализ инструментальных средств 9](#_Toc73659138)

[2.2.1. Клиентская часть 10](#_Toc73659139)

[2.3 Описание алгоритмов 14](#_Toc73659140)

[2.4 Серверная часть. 17](#_Toc73659141)

[Глава 3 Тестирование программы 19](#_Toc73659142)

[Глава 4 Перспективы развития приложения 20](#_Toc73659143)

[4.1 Перспективы технического развития 20](#_Toc73659144)

[Для дальнейшего технического развития необходимо будет сделать веб версию данного проекта, для того чтобы расширить круг возможных пользователей. 20](#_Toc73659145)

[4.2 Перспективы монетизации 20](#_Toc73659146)

[Заключение 21](#_Toc73659147)

[Список литературы 22](#_Toc73659148)

# Введение

Что такое боты в Telegram?

Боты – это специальные программы, выполняющие различные функции и упрощающие жизнь их пользователей. Написанные для платформы Telegram, они предназначены для выполнения самых разных функций: от получения новостей до поиска информации и даже торговли акциями. Главное задачей бота является автоматический ответ после введенной ему пользователем команды. При этом, работая непосредственно через интерфейс Telegram, программа имитирует действия живого юзера, за счет чего пользование таким ботом гораздо удобно и понятно.

Именно поэтому, многие компании, развивающие бизнес через интернет, используют возможности ботов по нескольким причинам:

Они быстро выполняют однообразную работу, позволяя разгрузить наемных сотрудников, тем самым экономя деньги компании.

Стоит отметить ограничения определенные в начале работы – проект должен использовать логику на языках программирования C++ и Python. Допускается использование других языков программирования и разметки, однако суммарная доля C++ и Python должна составлять не менее 60% от всего написанного кода.

# Глава 1 ПОСТАНОВКА ЗАДАЧИ

## Цель проекта

Целями моего проекта являются:

1. Создание удобного телеграмм бота, с клиент-серверной архитектурой;
2. Улучшение навыков работы с языками программирования C++ и Python;
3. Получение опыта в сфере разработке проектов;

## 1.2 Задачи и технологий реализации серверной части

Исходя из сказанного в пункте 1.1, серверная часть приложения выполняет ряд ключевых функций, в числе которых:

1. хранения данных пользователей;
2. обработка приходящих запросов;
3. рассылка информации по подключенным клиентам;
4. хранения индивидуального кода пользователя.

Хранения данных является важным аспектом работы программы, таким образом следует определить технологию, которую стоит использовать для передачи и хранения данных.

Существует два кардинально разных подхода: хранения данных с помощью какого-либо формата или же с использованием готовой базы данных.

В своем проекте я использовал JSON.

### 1.2.1 Форматы данных JSON

JSON — текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми. Формат JSON был разработан Дугласом Крокфордом [2].

Несмотря на происхождение от JavaScript (точнее, от подмножества языка стандарта ECMA-262 1999 года), формат считается независимым от языка и может использоваться практически с любым языком программирования. Для многих языков существует готовый код для создания и обработки данных в формате JSON.

Поскольку формат JSON является подмножеством синтаксиса языка JavaScript, то он может быть быстро десериализован встроенной функцией eval() [3].

JSON-текст представляет собой (в закодированном виде) одну из двух структур:

Набор пар ключ: значение. В различных языках это реализовано как запись, структура, словарь, хеш-таблица, список с ключом или ассоциативный массив. Ключом может быть только строка (регистрозависимость не регулируется стандартом, это остаётся на усмотрение программного обеспечения. Как правило, регистр учитывается программами — имена с буквами в разных регистрах считаются разными, например), значением — любая форма [3]. Повторяющиеся имена ключей допустимы, но не рекомендуются стандартом; обработка таких ситуаций происходит на усмотрение программного обеспечения, возможные варианты — учитывать только первый такой ключ, учитывать только последний такой ключ, генерировать ошибку.

Упорядоченный набор значений. Во многих языках это реализовано как массив, вектор, список или последовательность.

Структуры данных, используемые JSON, поддерживаются любым современным языком программирования, что и позволяет применять JSON для обмена данными между различными языками программирования и программными системами [4].

В качестве значений в JSON могут быть использованы:

* запись — это неупорядоченное множество пар ключ:значение, заключённое в фигурные скобки «{ }». Ключ описывается строкой, между ним и значением стоит символ «:». Пары ключ-значение отделяются друг от друга запятыми.
* массив (одномерный) — это упорядоченное множество значений. Массив заключается в квадратные скобки «[ ]». Значения разделяются запятыми. Массив может быть пустым, то есть не содержать ни одного значения. Значения в пределах одного массива могут иметь разный тип.
* число (целое или вещественное).
* литералы true (логическое значение «истина»), false (логическое значение «ложь») и null.
* строка — это упорядоченное множество из нуля или более символов юникода, заключённое в двойные кавычки. Символы могут быть указаны с использованием escape-последовательностей, начинающихся с обратной косой черты «\» (поддерживаются варианты \", \\, \/, \t, \n, \r, \f и \b), или записаны шестнадцатеричным кодом в кодировке Unicode в виде \uFFFF [2].

Строка очень похожа на литерал одноимённого типа данных в языке Javascript. Число тоже очень похоже на Javascript-число, за исключением того, что используется только десятичный формат (с точкой в качестве разделителя). Пробелы могут быть вставлены между любыми двумя синтаксическими элементами [3].

Следующий пример показывает JSON-представление данных об объекте, описывающем человека. В данных присутствуют строковые поля имени и фамилии, информация об адресе и массив, содержащий список телефонов. Как видно из примера, значение может представлять собой вложенную структуру.

{

"firstName": "Иван",

"lastName": "Иванов",

"address": {

"streetAddress": "Московское ш., 101, кв.101",

"city": "Ленинград",

"postalCode": 101101

},

"phoneNumbers": [

"812 123-1234",

"916 123-4567"

]

}

Обратите внимание на пару "postalCode": 101101. В качестве значений в JSON могут быть использованы как числа, так и строки. Поэтому запись "postalCode": "101101" содержит строку, а "postalCode": 101101 — уже числовое значение. Из-за слабой типизации в Javascript и PHP строка может быть приведена к числу и не влиять на логику программы. Тем не менее, рекомендуется аккуратно обращаться с типом значения, так как JSON служит для межсистемного обмена [4].

# Глава 2 Разработка Телеграмм-Бота

## 2.1 Логика работы бота

Следующим этапом разработки бота является определение основных функций, которые должны присутствовать для корректной работы. Такими функциями являются:

1. Идентификация пользователя по индивидуальному коду;
2. Хранение данных о пользователе в «Json» файле;
3. Реализация хранения Заметок на сервере;
4. Реализация встроенного календаря для телеграмм-ботов

## 2.1.2 Анализ инструментальных средств

На начальной стадии разработки проекта был поставлен вопрос: на каком языке писать Телеграмм-бота. Мой выбор остановился на языке программирования Python. Почему же я выбрал именно его для написания клиентской части приложения, а не С++? Я выделил следующие причины:

* Понятный синтаксис и функциональность
* Множество простых и полезных библиотек для изучения
* Отличная читаемость
* Python портативен. Он также кроссплатформенный, что позволяет запускать код на разных устройствах.
* Предлагается с крупной библиотекой, включающей различные парсеры и многое другое

В качестве среди разработки клиентского приложения был выбрать редактор Sublime Textв виду преимущества в виде понятности и комфортности использования относительно конкурентов (например, в Sublime Text нет встроенного отладчика).

В свою очередь, для написания серверной части я выбрал язык программирования C++ из-за ряда причин:

* удобная и уже знакомая мне простая в использовании библиотека «cpp\_httlib», которая очень хороша справляется с задачей обработки запросов, посылаемых с клиентской стороны;
* скорость вычислительной возможности у компиляторов быстрее, чем у интерпретаторов. В качестве среды разработки серверного приложения был выбран редактор VS Community 2019

Для работы с данными в «Json» формате была изучена библиотека «Json» на Python. Чтобы анализировать запросы, получаемые с клиентской части на сервер, и сохранять данные в файл, в С++ использовалась библиотека «JSON for Modern C++».

* Интуитивно понятный синтаксис. В таких языках, как Python, JSON воспринимается как тип данных первого класса. В этой библиотеке использовали всю операторную магию современного C++, чтобы добиться того же ощущения в вашем коде.
* Тривиальная интеграция. Весь наш код состоит из одного заголовочного файла json.hpp. Вот и все. Ни библиотеки, ни подпроекта, ни зависимостей, ни сложной системы сборки. Класс написан на языке C++11. В целом, все не должно требовать корректировки флагов компилятора или настроек проекта.

## 2.2.1. Клиентская часть

**Подмодуль «telegram.ext»**

Подмодуль «telegram.ext» построен поверх чистой реализации API. Он обеспечивает простой в использовании интерфейс и снимает некоторую работу с программиста, так что вам не придется повторяться.

Он состоит из нескольких классов, но двумя наиболее важными из них являются «telegram.ext.Updater» и «telegram.ext.Dispatcher».

Класс Updater непрерывно извлекает новые обновления из telegram и передает их классу Dispatcher. Если вы создадите объект обновления, он создаст для вас Диспетчер и свяжет их вместе с очередью. Затем вы можете зарегистрировать обработчики различных типов в диспетчере, который будет сортировать обновления, полученные средством обновления, в соответствии с зарегистрированными обработчиками и доставлять их в определенную вами функцию обратного вызова.

Каждый обработчик является экземпляром любого подкласса класса «telegram.ext.Handler». Библиотека предоставляет классы обработчиков почти для всех случаев использования, но, если вам нужно что-то очень конкретное, вы также можете самостоятельно создать подкласс обработчика.

Для начала вам понадобится маркер доступа. Если вы уже прочитали и следили за введением в API, вы можете использовать тот, который вы создали тогда. Если нет: Чтобы сгенерировать токен доступа, вам нужно поговорить с @BotFather и выполнить несколько простых шагов. Хотя вам действительно следует сначала прочитать введение.

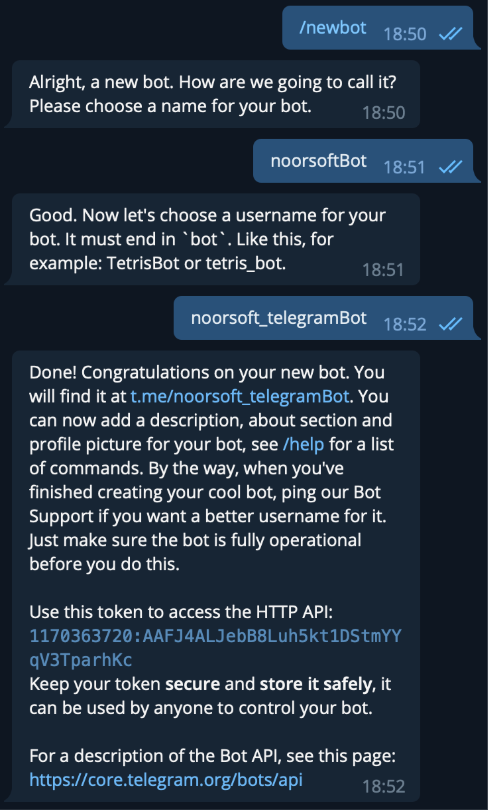


Рисунок 2.2.1.1 Получение индивидуального

«TOKEN» для доступа к телеграмм-боту.

**Класс «telegram.ext.ConversationHandler»**

Обработчик для ведения разговора с одним или несколькими пользователями с помощью обновлений Telegram, управляя четырьмя коллекциями других обработчиков.

Первая коллекция, список с именем entry\_points, используется для инициирования разговора, например, с помощью «telegram.ext.CommandHandler» или «telegram.ext.MessageHandler».

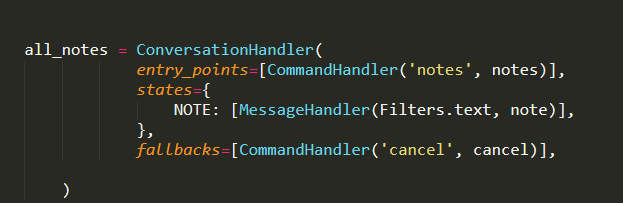


Рисунок 2.2.1.2. Список с именем entry\_points для активации разговора

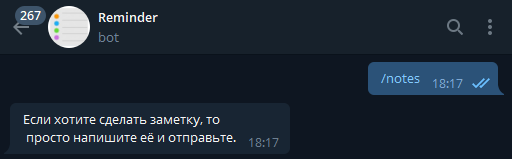


Рисунок 2.2.1.3. Пример реализации объекта entry\_points

Вторая коллекция, словаря с именем *states*, содержит различные сценарии диалога и один или несколько связанных обработчиков, которые следует использовать, если пользователь отправляет сообщение, когда разговор с ними в данный момент находится в этом состоянии. Здесь также можно определить состояние ТАЙМ-аута для определения поведения при превышении conversation\_timeout и состояние ОЖИДАНИЯ для определения поведения при получении нового обновления, в то время как предыдущий обработчик @run\_async не завершен.

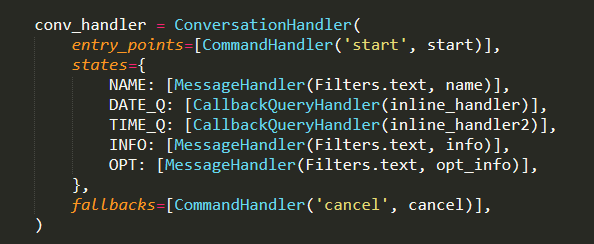


Рисунок 2.2.1.4. Реализация списка states

В одном из моих обработчиков «conv\_handler», содержаться пять сценариев Name отвечает за название напоминания, Date\_Q создает интерактивный календарь, реализованный в файле «telegramcalendar.py», Time\_Q создаёт интерактивные кнопки для выбора времени, Info сохраняет информацию о созданном напоминании, OPT – опциональная информация.

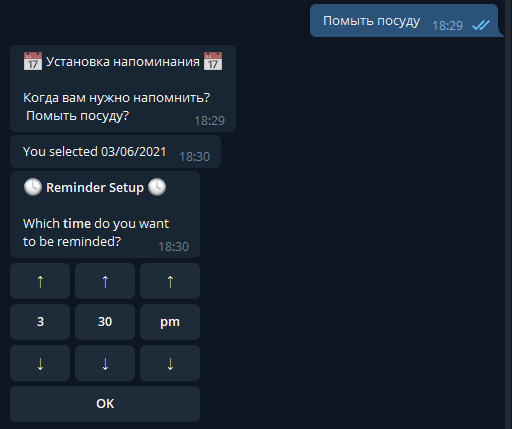


Рисунок 2.2.1.5. Реализация интерактивных кнопок.

Третья коллекция, список с резервными командами, они используются, если пользователь в данный момент находится в разговоре, но не имеет правильного связанного обработчика, либо обработчик, связанный с состоянием, не подходит для обновления, например, если обновление содержит команду, но ожидается обычное текстовое сообщение. Вы можете использовать это для команды /cancel или для того, чтобы пользователь знал, что его сообщение не было распознано.

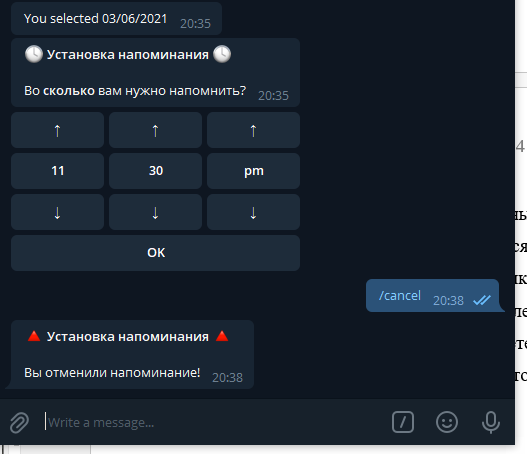


Рисунок 2.2.1.6. Реализация «fallback» обработчика.

## 2.3 Описание алгоритмов

**Функция, которая записывает название напоминания.**

Она принимает сообщение, отправленное пользователем. Записывает в файл JSON имя пользователя, а также id его чата. После этого вызывается календарь для выбора числа.

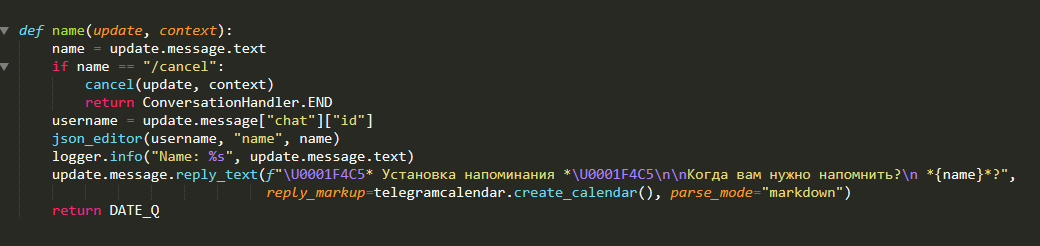


Рисунок 2.3.1. Функция name

**Алгоритм выбора даты.**

Идёт вызов календаря из файла telegramcalendar.py, также обновляется Json файл, а после выбора числа идёт вызов имитации часов.

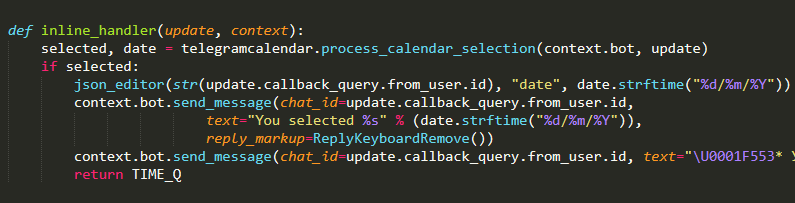


Рисунок 2.3.2. Функция выбора времени напоминания.

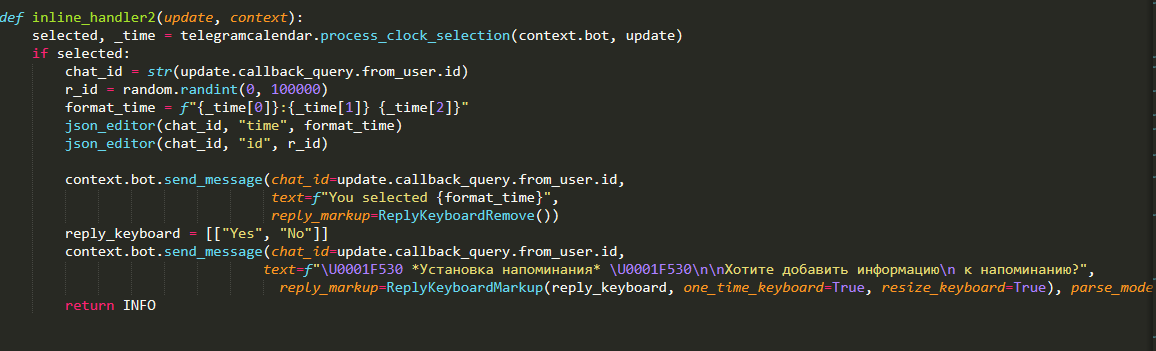


Рисунок 2.3.3. Функция выбора часов.

Функция «inline\_handler» отвечает за выбор числа напоминания, а «inline\_handler2» за выбор часов, когда придёт уведомление.

**Функция установки напоминания.**

Данная функция, с помощью оператора ветвления проверяет, хочет ли пользователь добавить какую-нибудь дополнительную информацию? Если да, то он перенаправляет на функцию «opt\_info» типа (optional info). Если же пользователю это не нужно, то он переводит текущее время, в секунды и создает очередь с помощью класса «telegram.ext.JobQueue». Этот класс позволяет периодически выполнять задачи с ботом. Это удобная оболочка для библиотеки APScheduler.



Рисунок 2.3.4. Функция info

Также в данном боте реализованы, две функции для выбора UTC — это всемирное координированное время, стандарт времени, принятый на Земле. От UTC отсчитываются часовые пояса. UTC заменил устаревшее время по Гринвичу (GMT). UTC расшифровывается как Coordinated Universal Time.

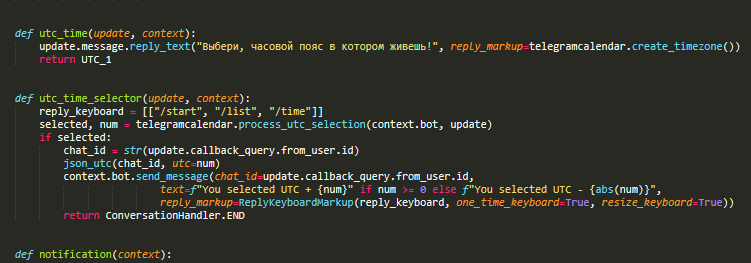


Рисунок 2.3.5. Функции выбора часового пояса

## 2.4 Серверная часть.

В сервере я реализовал две функции. Одна отправляет заметки в Телеграмм-бот, другая принимает и сохраняет их.



Рисунок 2.4.1 Функция отправляет заметки на клиент.

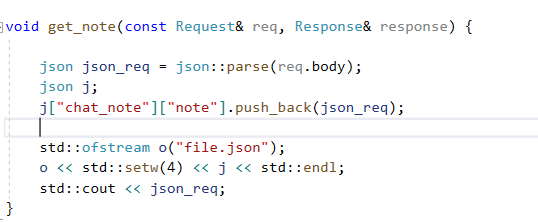


Рисунок 2.4.2 Функция принимает заметки и сохраняет их на сервере.

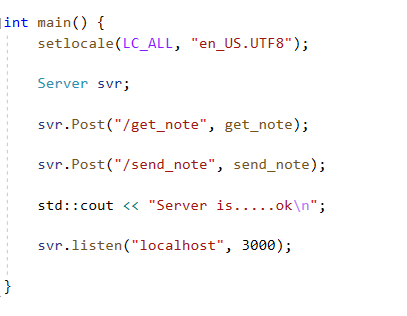


Рисунок 2.4.3 Сервер отправляет POST запросы.

# Глава 3 Тестирование программы

Тут будет описание тест-кейсов и отчетов тестирования.

# Глава 4 Перспективы развития приложения

## 4.1 Перспективы технического развития

## Для дальнейшего технического развития необходимо будет сделать веб версию данного проекта, для того чтобы расширить круг возможных пользователей.

Также огромным шагом будет в развитие этого проекта будет, внедрение возможности устанавливать напоминание с помощью голосовых сообщений. За пример можно будет взять такой бот как «@voicos\_bot», который конвертирует голосовые сообщения в текст.

## 4.2 Перспективы монетизации

Сейчас многие Телеграмм-боты монетизируются, и при достаточном количестве пользователей, возможна продажа рекламы.

Также можно будет продавать, какие-нибудь эксклюзивные корпоративные функции для компаний и коллективов.

# Заключение

В результате выполнения курсовой работы, был реализован рабочий Телеграмм-бот, который справляется с поставленными задачами.

Поставленная цель была достигнута посредством решения следующих задач:

1. Реализация выбора времени напоминания с помощью интерпретации календаря.
2. Сервер реализует логику обработки запросов посредством библиотеки cpp\_httplib.
3. В качестве формата данных выбран JSON.
4. Хранение заметок на сервере, и дальнейшее их использование.

В работе используются следующие методы исследования: анализ, синтез, программирование и тестирование.

Анализ выражается в изучении современных методов разработки клиент-серверных приложений и выбор наиболее подходящих под требования проекта.

Дальнейшими перспективами развития проекта могут быть множество дополнений, которые ещё более упростят взаимодействие с данным ботом.

# Список литературы

1. Многоуровневые системы клиент-сервер [электронный ресурс] / Валерий Коржов – 17.06.1997 – Режим доступа: https://www.osp.ru/nets/1997/06/142618, дата обращения: 18.05.2021;
2. JSON [электронный ресурс] / JSON Standart – Режим доступа: http://json.org/json-ru.html, дата обращения: 18.05.2021;
3. Бьёрн Страуструп. Язык программирования C++ = The C++ Programming Language / Пер. с англ. – 3-е изд. – СПб.; М.: Невский диалект – Бином, 1999. – 991 с. – 3000 экз. – ISBN 5-7940-0031-7 (Невский диалект), ISBN 5-7989-0127-0 (Бином), ISBN 0-201-88954-4 (англ.).
4. Бьёрн Страуструп. Язык программирования C++. Специальное издание = The C++ programming language. Special edition. – М.: Бином-Пресс, 2007. – 1104 с. – ISBN 5-7989-0223-4.
5. Бьёрн Страуструп. Программирование: принципы и практика использования C++, исправленное издание = Programming: Principles and Practice Using C++. – М. : Вильямс, 2011. – С. 1248. – ISBN 978-5-8459-1705-8.
6. Бьёрн Страуструп. Дизайн и эволюция C++ = The Design and Evolution of C++. – СПб. : Питер, 2007. – 445 с. – ISBN 5-469-01217-4.
7. *Лучано Рамальо.* Python. К вершинам мастерства : [рус.] = Fluent Python. O’Reilly, 2015 : [пер. с англ.]. – ДМК Пресс, 2016.
8. *Кеннет Рейтц, Таня Шлюссер.* Автостопом по Python : [рус.] = The Hitchhiker`s Guide to Python : [пер. с англ.]. – Издательский дом «Питер», 2017. – ISBN 9785496030236.
9. *David M. Beazley.* Python Essential Reference. – 4th Edition. – Addison-Wesley Professional, 2009. – 717 с. – ISBN 978-0672329784.
10. Лучано Рамальо. Python. К вершинам мастерства : [рус.] = Fluent Python. O’Reilly, 2015 : [пер. с англ.]. – ДМК Пресс, 2016.
11. Кеннет Рейтц, Таня Шлюссер. Автостопом по Python : [рус.] = The Hitchhiker`s Guide to Python : [пер. с англ.]. – Издательский дом «Питер», 2017. – ISBN 9785496030236.
12. David M. Beazley. Python Essential Reference. – 4th Edition. – Addison-Wesley Professional, 2009. – 717 с. – ISBN 978-0672329784.
13. Лучано Рамальо. Python. К вершинам мастерства : [рус.] = Fluent Python. O’Reilly, 2015 : [пер. с англ.]. – ДМК Пресс, 2016.
14. Кеннет Рейтц, Таня Шлюссер. Автостопом по Python : [рус.] = The Hitchhiker`s Guide to Python : [пер. с англ.]. – Издательский дом «Питер», 2017. — ISBN 9785496030236.
15. David M. Beazley. Python Essential Reference. – 4th Edition. – Addison-Wesley Professional, 2009. – 717 с. – ISBN 978-0672329784.