Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЙ УНИВЕРСИТЕТ»

Факультет математики и информационных технологий
Кафедра компьютерной безопасности и математического обеспечения информационных систем

РАСЧЕТНО-ГРАФИЧЕСКОЕ ЗАДАНИЕ

по дисциплине «Технология создания прикладного программного обеспечения»

«Учебное расписание студентов компьютерной безопасности с помощью бота Telegram»

ОГУ 090301.65.3017.006 ОО

Руководитель
канд. техн. наук, доцент
_____ Влацкая И.В.
“_____”_____2017г.
Исполнитель
студент гр. 13КБ(с)РЗПО
_____Казакова М.А.
“_____”_____2017г.

Оренбург 2017

Содержание

Введение

Чат-боты представляют собой программы-собеседники, в функционал которых входит способность общаться с человеком, используя человеческий язык, а также выполнение разнообразных команд-просьб.

С помощью ботов можно узнавать абсолютно всю имеющуюся в сети информацию, от поиска новостей и заказа товаров, до перевода текстов и поиска спутника жизни.

На данный момент существует большое количество различных ботов на базе Telegram, но их функционал не соответствует поставленной задаче. Таким образом, на базе Telegram будет создан программный продукт, позволяющий просмотреть учебное расписание с официального сайта ОГУ, для группы 13КБ.

Разрабатываемая система изначально будет предоставлять также ограниченный набор функций, но со временем позволит выйти за рамки задачи и позволить хранить информацию и о других вещах, со временем превращаясь в удобного помощника.

Целью *данного расчетно-графического задания является помощь студентам при просмотре учебного расписания.*

1 Разработка модели демонстрационной программы «Учебное расписание студентов компьютерной безопасности с помощью бота Telegram»

1.1 Концептуальная модель программы «Учебное расписание студентов компьютерной безопасности с помощью бота Telegram»

Программа предназначена для студентов для просмотра учебного расписания ОГУ. Главной целью является оптимизация просмотра расписания. Для этого будет реализован бот на базе Telegram API.

Пользователь добавляет бота в свои контакты. После выбирает временной промежуток для отображения расписания или справку по вызову команд бота. В качестве выходного параметра будет сообщение бота содержащее расписание группы в определённый промежуток времени. Пользователь не ограничен в количестве действий с ботом.

Планируется, что программа будет работать с сайтом ОГУ и выполнять синтаксический анализ расписания группы.

Это показано на диаграмме прецедентов (рис. 1).

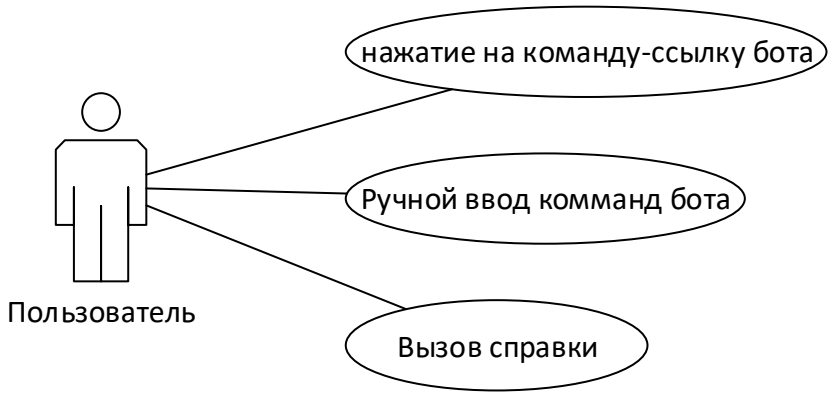

Рисунок 1 – Диаграмма прецедентов

Диаграмма активности показывает программный алгоритм как последовательность действий (рис. 2).

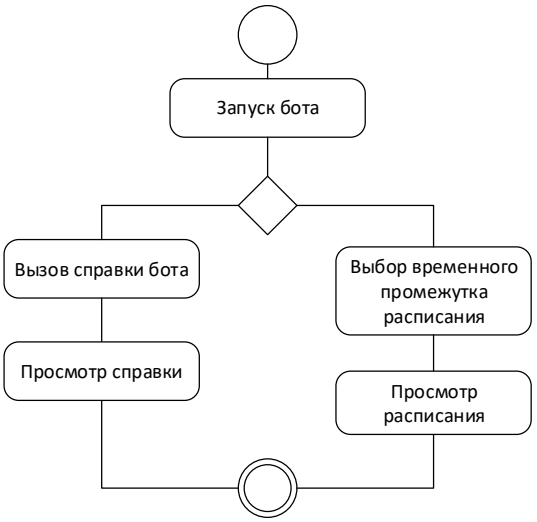

Рисунок 2 – Диаграмма активности

Диаграмма состояний показывает набор состояний системы и представляет собой конечный автомат.

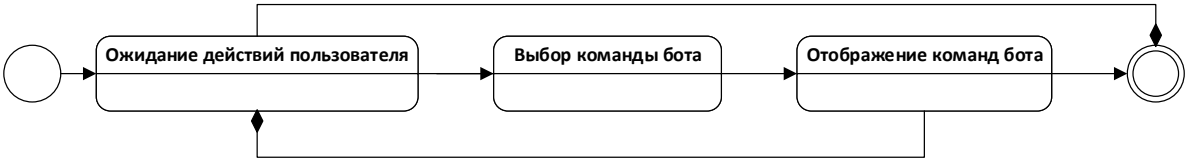

Рисунок 3 – Диаграмма состояний

Для поставленной задачи программист выбрал каскадную модель жизненного цикла потому, что:

– задача проста;

– требования окончательно согласованы;

– переход на очередную стадию проекта возможен только после полного завершения работ на предыдущей стадии.

В качестве метода программирования будет использоваться архитектурный подход, т.к. для данной предметной области можно выделить примитивные функции, которые будут использоваться несколько раз при решении задачи.

1.2 Описание параметров качества

Разрабатываемое программное средство обязательно должно обладать следующими параметрами качества:

– пригодность для применения, чтобы программа выполняла объявленный набор функции и имела внешнее описание;

– надежность, чтобы программа работала безотказно с достаточно большой вероятностью и время выполнения не превышало разумных норм;

– эффективность для того, чтобы программа обеспечивала требуемую производительность решения функциональных задач с учетом количества используемых вычислительных ресурсов в установленных условиях.

– применимость означает, что разрабатываемая программа должна обладать легким, понятным и интуитивным интерфейсом.

Программа должна работать на OS Windows, Android, IOS. Поэтому параметр «переносимость», также, как и «сопровождаемость» в данном проекте имеет место.

2 Реализация демонстрационной программы «Учебное расписание студентов компьютерной безопасности с помощью бота Telegram»

Для реализации проекта будет бот для Telegram — это приложение, запущенное на вашей стороне и осуществляющее запросы к Telegram Bot API. Причем API довольное простое — бот обращается на определенный URL с параметрами, а Telegram отвечает JSON объектом. Для описания тела бота будет использоваться язык Python3.

Архитектура программного обеспечения – цельная программа, поскольку задача в целом несложная, а функция программы ярко выраженна.

Итоговый продукт - программа, которая должна отображать расписание группы.

2.1 Организация структуры программного средства

На рисунке 4 представлена SADT-диаграмма программного средства. Она показывает функциональную структуру программы, т.е. производимые им действия и связи между ними.

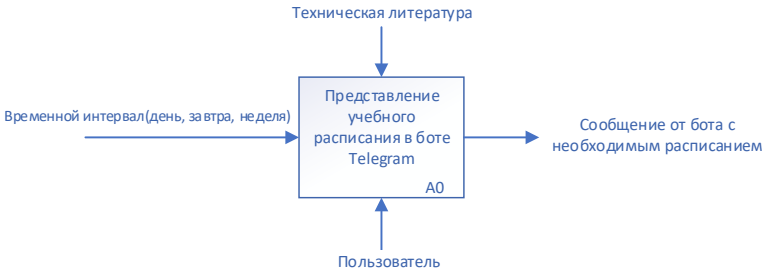

Рисунок 4 — SADT-диаграмма A0

Была произведена детализация программного средства (рис.5), которая отображает подфункции основного блока программы и движение информационных потоков между ними.

Рисунок 5 — Декомпозиция SADT-диаграммы A0

Рисунок 5 показывает наличие коммуникационной (4) и последовательной (5) связности модулей.

2.2 Модули

Программа содержит несколько модулей. Модуль для хранения констант, который, например, содержит уникальный токен бота, чтобы защитить бота от перехвата управления им.

Модуль синтаксического анализа сайта. Создаётся «парсер», для корректного отображения элементов сайта в боте.

Основной модуль программы выполняет непосредственную работу с библиотеками Telegram API, а также содержит тело бота.

Для работы с ботом используется библиотека telebot, которая имеет простой интерфейс для взаимодействия с пользователем. Она включает в себя некоторые функции обработки запросов и ответов бота, такие как @bot.message_handler(параметр) – обработка сообщений бота, bot = telebot.TeleBot(bot_private_constants.token) – функция получения уникального токена бота для работы с ним. Для корректной работы необходима установка пакетов pyTelegramBotAPI.

Таким образом, если рассматривать характеристики модулей по Майерсу, данные модули вызывают друг друга, поэтому их тип сцепления «По данным», а уровень значимости 1. Модули справки и внешних библиотек являются рутинными, поскольку результат их работы зависит только от значений параметров.

Функционально прочным модулем является модуль хранения констант, он реализует одну определенную функцию. Информационно прочный модуль – модуль синтаксического анализа, он работает с одной структурой данных и

выполняет множество функций. Определить прочность внешних библиотек не представляется возможным.

2.3 Метрики качества программного средства

В общем случае применение метрик позволяет изучить сложность разработанного, оценить объем работ, стилистику разрабатываемой программы и усилия, потраченные для реализации того или иного решения.

Количественные метрики вычисляются на основе исходного кода программы. Метрика Холстеда показывает уровень качества программирования сложность понимания программы (таблица 1).

Таблица 1 – Расчет метрики Холстеда

Название	n1	n2	n	N1	N2	N	V
Основное окно	6	49	55	63	98	171	297,6

Метрика Джилба так же относится к количественным метрикам и показывает сложность программы (таблица 2).

Таблица 2 – Расчет метрики Джилба

Название	CL	n	cl
Окно справки	18	63	0,28

Метрики сложности потока управления программ основаны уже не на количественных показателях, а на анализе управляющего графа программы. Метрика МакКейба показывает требуемое количество проходов для покрытия всех контуров сильно связанного графа или, другими словами, количество тестовых прогонов программы, необходимых для исчерпывающего тестирования по критерию «работает каждая ветвь». Для расчета метрики необходимо знать следующие параметры: количество вершин графа (функций) V=28, количество ребер E=57, число компонентов связности графа p=3; тогда цикломатическое число Маккейба Z=15. Метрика граничных значений определяется по формуле S0=1-(V-1)/Sa, где V – число вершин, Sa – абсолютная граничная сложность программы (в данном случае она равна 3). Число вершин уже известно, поэтому можно найти относительную граничную сложность программы, она равна 9,3.

Метрика Чепина является мерой трудности понимания программ на основе входных и выходных данных и относится к метрикам сложности потока данных (таблица 3).

Таблица 3 – Расчет метрики Чепина

Название	P	M	C	T	Q
Основное окно	2	26	7	6	78

Рассчитав данные метрики, можно сделать вывод, что данное программное средство несколько сложное для понимания и, возможно, его можно сделать более простым.

3 Тестирование системы

Был протестирован как пользовательский интерфейс, так и прямое обращение к сервису. Позитивные тесткейсы представлены в таблице 1. Негативные тесткейсы представлены в таблице 2.

Таблица 1 – Позитивные тесткейсы

Действие	Компонент	Ожидаемый результат	Полученный результат
Нажатие на кнопку	Кнопка /start	Добавление бота	Произошло добавление бота в список контактов (разовое), а также вывод информации о боте. Последующие нажатия на эту кнопку будут открывать справку.
Нажатие на кнопку	Кнопка /now	Вывод ближайшей или уже начавшейся пары	Вывод сообщения бота с названием пары и номером кабинета.
Нажатие на кнопку	Кнопка /today	Вывод расписания на день	Вывод сообщения бота с расписанием на текущий день.
Нажатие на кнопку	Кнопка /tomorrow	Вывод расписания на завтрашний день	Вывод сообщения бота с расписанием на завтрашний день.
Нажатие на кнопку	Кнопка /timetable	Вывод расписания на семестр	Вывод ссылки на полное расписание на сайте ОГУ.
Нажатие на кнопку	Кнопка /week	Вывод расписания на неделю	Вывод сообщения бота с расписанием на неделю.
Ввод команды, не прописанной в клиенте /command1	Отображение сообщения: «Введите правильную команду».	Ввод команды, не прописанной в клиенте /command1	Отображение сообщения: «Введите правильную команду».

Таблица 2 – Негативные тесткейсы

Действие	Полученный результат
Попытка обращения при недоступном сервисе или отсутствии сети	Отображается ошибка о проблемах с подключением. Программа не завершается аварийно.
Ввод команды, не прописанной в клиенте /command1	Отображалась ошибка на сервере о необработанной команде. Программа завершается аварийно. Ошибка исправлена.

Тестирование показало, что пользователь не получает сообщения об ошибке, но на сервере отображалась ошибка о необработанной команде. Программа завершается аварийно. Данная ошибка была исправлена.

Заключение

Разработанное программное средство помогает студентам в быстром поиске расписания. Особенностью программы является быстрота использования из любого места, где есть подключение к интернету. Программа имеет свою справку с описанием всех процессов, происходящих с деревьями и кратким руководством пользования.

Для разработки данного программного средства были изучены научные статьи, посвященные проектированию и разработке программных средств. Были рассмотрены основные возможности языка Python3, такие как работа со сторонними библиотеками, корректного отображения информации. Так как дополнительно для работы бота использовалась внешняя библиотека telebot, были изучены ее основные функции.

Тестирование программного средства выявило небольшие ошибки, которые были исправлены.

В будущем планируется улучшение данного программного средства – доработка пользовательского интерфейса, ускорение обработки больших объемов данных, добавление нескольких новых функций, а также устранение найденных при тестировании багов.

Список использованных источников

1 Дж. Макконелл Анализ алгоритмов. Активный обучающий подход. — 3-е дополненное издание. М: Техносфера, 2009. -416с.

2 Скиена С. Алгоритмы. Руководство по разработке. 2-е изд.: Пер. с англ. — СПб.: БХВ-Петербург. 2011. — 720 с.: ил.

3 Влацкая И. В., Заельская Н. А., Надточий Н. С. Проектирование и реализация прикладного программного обеспечения: учебное пособие //Оренбург: ОГУ. – 2015. – С. 11.

4 Дональд Э. Кнут. Глава 2.3. Деревья // Искусство программирования = The Art of Computer Programming. — 3-е изд. — М.: Вильямс, 2000. — Т. 2. Основные алгоритмы. — 832 с. — 7000 экз. — ISBN 5-8459-0081-6 (рус.) ISBN 0-201-89684-2 (англ.).

5 Майерс Г. Надежность программного обеспечения - М.: Мир, 1980

Приложение А
Исходный код

Bothelper.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import bot_private_constants
import timetable
import telebot
import json
import time
import random
import os.path
import re
bot = telebot.TeleBot(bot_private_constants.token)
S = json.load(open("strings.json"))
commands_with_description = [
        "/group_list — Список группы",
        "/today — Расписание на сегодня",
        "/tomorrow — Расписание на завтра",
   "/timetable — Расписание на неделю",
   "/now — А что сейчас?"]
@bot.message_handler(commands = ['group_list'])
def group_list(message):
        group_list_string = ""
        for index, person in enumerate(bot_private_constants.group_list, start = 1):
                group_list_string += str(index) + ". " + person + "\n"
        bot.send_message(message.chat.id, group_list_string)
@bot.message_handler(commands = ['help', 'start'])
def introduction(message):
        intro_message = "Я бот-помощник. Умею отвечать на команды:\n\n"
        for command in commands_with_description:
                intro_message += command + "\n"
        bot.send_message(message.chat.id, intro_message, True)
@bot.message_handler(commands = ['today'])
def today(message):
   timetable_string = "\n".join(
      map(timetable.make_long_subject,
         timetable.get_today_subjects()))
   bot.send_message(message.chat.id, timetable_string)
@bot.message_handler(commands = ['tomorrow'])
def tomorrow(message):
   timetable_string = "\n".join(
      map(timetable.make_long_subject,
         timetable.get_tomorrow_subjects()))
   bot.send_message(message.chat.id, timetable_string)
@bot.message_handler(commands = ['timetable'])
def show_timetable(message):
   timetable_string = ""
   for day, dayname in enumerate(S['weekdays'][:-1]):
      timetable_string += "{}\n{}\n\n".format(
            dayname,
            "\n".join(
               map(timetable.make_long_subject,
```

```
                    timetable.get_subjects(day))))
    bot.send_message(message.chat.id, timetable_string)
@bot.message_handler(commands = ['now'])
def tomorrow(message):
    subject, tm = timetable.get_next_subject()
    if subject == None:
        if tm == 0:
            fmt = S["no_subject"]
        else:
            fmt = S["weekend"]
    else:
        subject = timetable.make_inline_subject(subject)
        if tm > 0:
            fmt = S["next_subject"]
        else:
            fmt = S["current_subject"]
    interval = timetable.make_human_time("{:02}:{:02}".format(
        abs(int(tm)) // 3600, abs(int(tm)) // 60 % 60))
    message_string = fmt.format(subject=subject, interval=interval)
    bot.send_message(message.chat.id, message_string)
bot.polling(none_stop=True)
while True:
    time.sleep(100)
```

Parser.py

```
import pandas as pd
import datetime, locale
import numpy
from prettytable import PrettyTable
locale.setlocale(locale.LC_ALL, "ru_RU.UTF-8")
def tb(x):
    table = PrettyTable()
    table.add_column("Дата", [x[i][0] for i in range(len(x))])
    table.add_column("1 пара 08:00-09:30", [x[i][1] for i in range(len(x))])
    table.add_column("2 пара 09:40-11:10", [x[i][2] for i in range(len(x))])
    table.add_column("3 пара 11:20-12:50", [x[i][3] for i in range(len(x))])
    table.add_column("4 пара 13:20-14:50", [x[i][4] for i in range(len(x))])
    table.add_column("5 пара 15:00-16:30", [x[i][5] for i in range(len(x))])
    print(table)
def today(df):
    today = datetime.date.today()
    return df.values[df['Дата'] == today.strftime("%d.%m.%Y(%A)").lower()]
def week(df):
    week = [datetime.date.today() + datetime.timedelta(days=i) for i in range(7+1)]
    sp = []
    for date in week:
        sp += [df.values[df['Дата'] == date.strftime("%d.%m.%Y(%A)").lower()][0].tolist()]
    return sp
def semestr(df):
    sp = []
    def daterange(date1, date2):
        for n in range(int ((date2-date1).days)+1):
            yield date1 + datetime.timedelta(n)
    for date in daterange(datetime.date(2017,9,1), datetime.date(2017,12,31)):
        sp += [df.values[df['Дата'] == date.strftime("%d.%m.%Y(%A)").lower()][0].tolist()]
```

```
    return sp
def main():
    calls_df, =
pd.read_html("http://www.osu.ru/pages/schedule/?who=1&what=1&facult=5683&potok=2013&group=686
1&mode=full",header=0)
    for index in range(calls_df.index.argmax()):
        if index <= calls_df.index.argmax():
            if calls_df.values[index][0] == 'Дата':
                calls_df.drop(calls_df.index[index], inplace=True)
                calls_df.reset_index(drop=True, inplace=True)
    calls_df = calls_df.replace(numpy.nan, ' ', regex=True)
    query = input("День, неделя, семестр? ")
    if query == "день":
        res = today(calls_df)
    elif query == "неделя":
        res = week(calls_df)
    elif query == "семестр":
        res = semestr(calls_df)
    else:
        print("И всё таки?")
    tb(res)
if __name__ in "__main__":
    main()
```