**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и системы управления»

Кафедра ИУ5. Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №6

«Разработка простого телеграм-бота на языке Python»

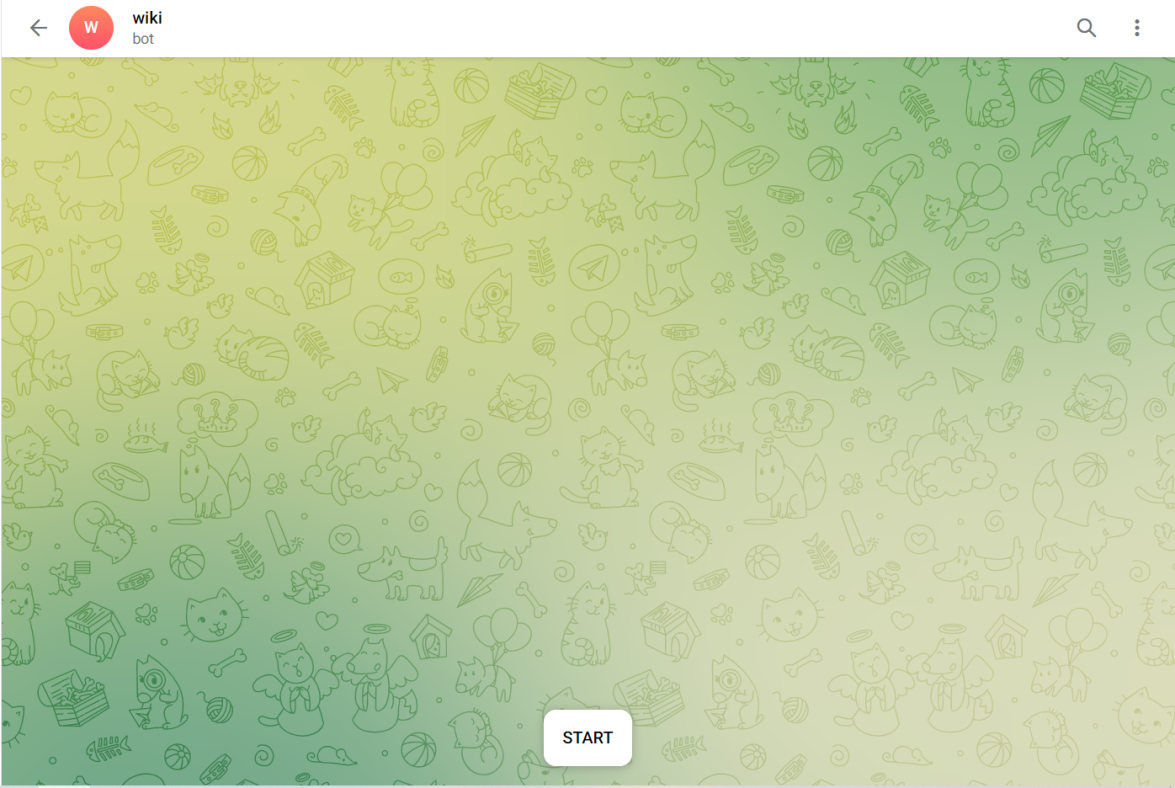
|  |  |  |
| --- | --- | --- |
| Выполнил:  Каженец Д.Н.  ИУ5-31Б |  | Проверил:  Гапанюк Е.Ю. |
| Подпись и дата: |  | Подпись и дата: |

В рамках лабораторной работы осуществлено программирование телеграм-бота на языке Python. Бот может выполнять различные команды, получаемые от пользователя. В табл. 1 представлены список команд и их описание.

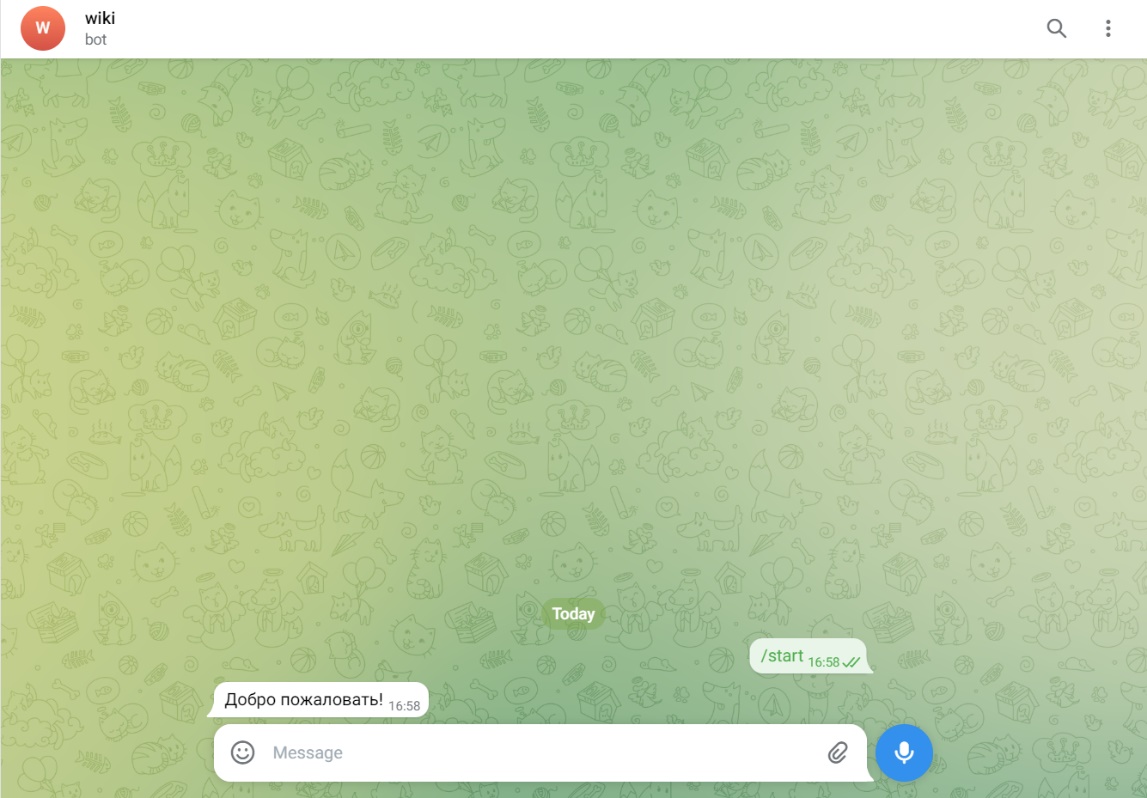
Таблица 1.

|  |  |  |
| --- | --- | --- |
| № п.п. | Название команды | Описание работы команды |
| 1. | /start | Запускает бота |
| 2. | /help | Отправляет пользователю список всех команд |
| 3. | /spent 2000 | Делает запись о расходе на указанную сумм |
| 4. | /earned 2000 | Делает запись о доходе на указанную сумму |
| 5. | /history | Отправляет пользователю историю операций за день |
| 6. | /history day | Отправляет пользователю историю операций за день |
| 7. | /history month | Отправляет пользователю историю операций за месяц |
| 8. | /history year | Отправляет пользователю историю операций за год |

**Начало работы**

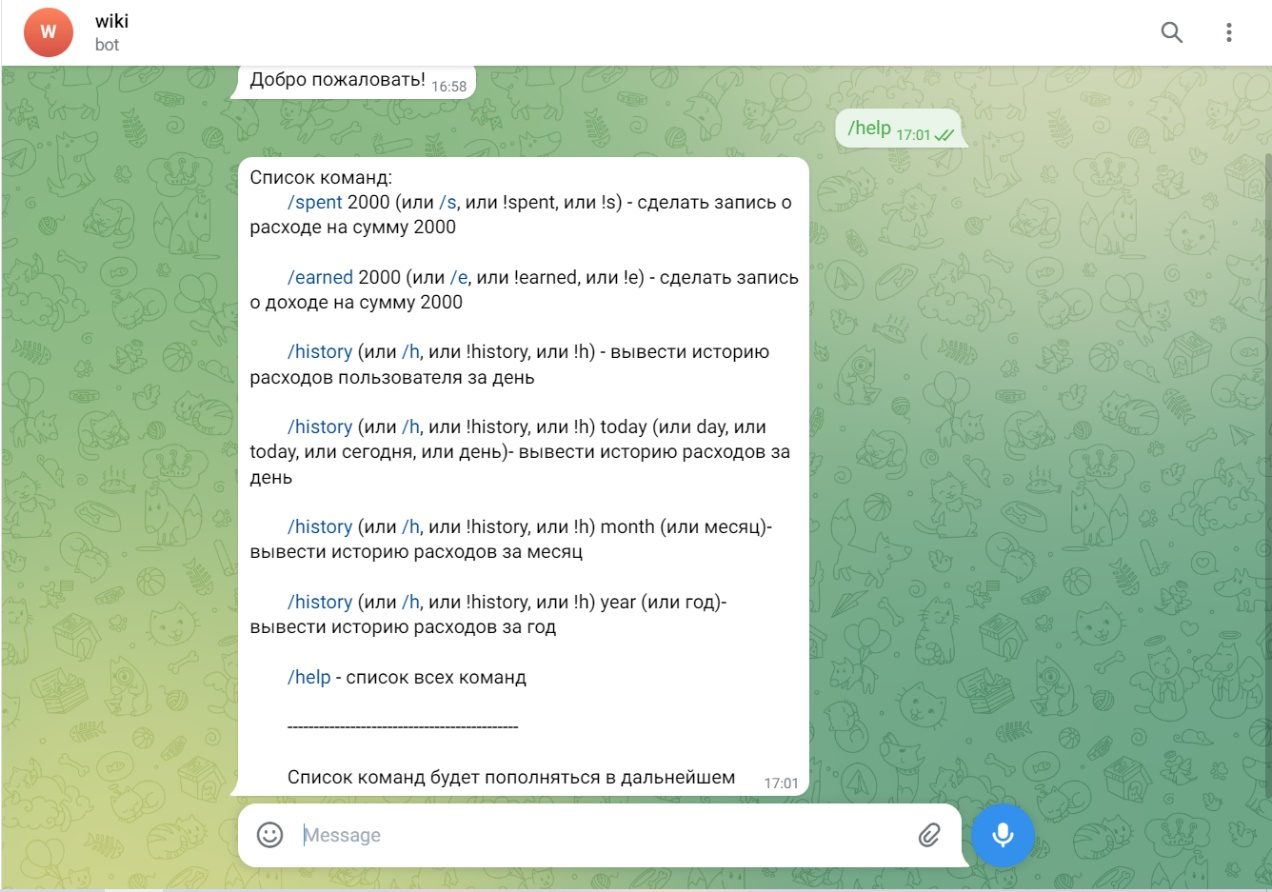


Для того чтобы запустить бота, необходимо нажать на кнопку «START».



После этого бот здоровается с пользователем и начинает работу

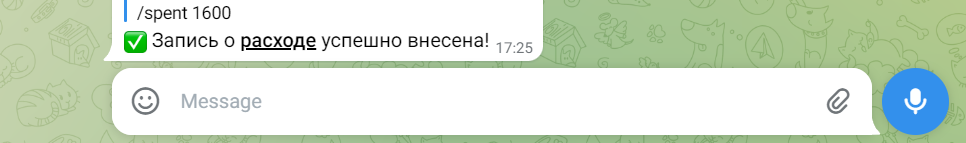
**Команда «/help»**

****

При вызове команды «/help» пользователю отправляется список всех доступных команд бота с их описанием.

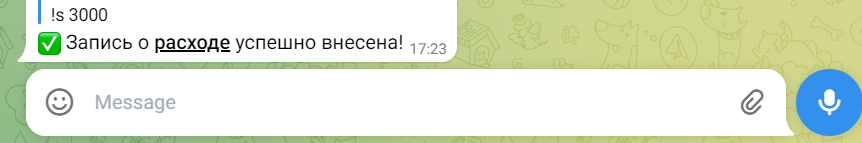
**Команда «/spent»**

При вызове команды «/spent 2000» делается запись о расходе на сумму, указанную пользователем (также возможны варианты «/s», «!spent», «!s»)



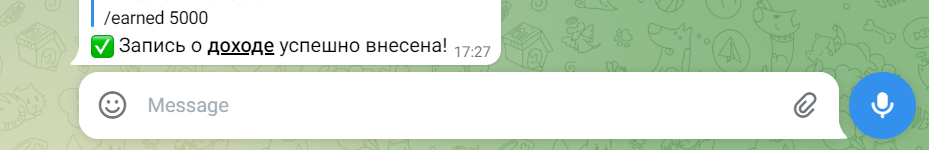
Если боту не удалось найти статью, связанную с отправленным пользователем текстом, то отправляется соответствующее сообщение

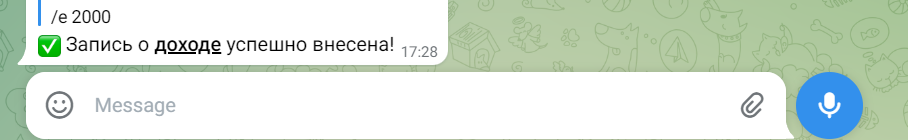
Если боту удалось найти статью, связанную с отправленным пользователем текстом, то отправляются первые 1000 символов этой статьи



**Команда «/earned»**

При вызове команды «/earned 2000» делается запись о доходе на сумму, указанную пользователем (также возможны варианты «/e», или «!earned», или «!e»)





db.py

import sqlite3

class BotDB:

def \_\_init\_\_(self, db\_file):

self.conn = sqlite3.connect(db\_file)

self.cursor = self.conn.cursor()

def user\_exists(self, user\_id):

result = self.cursor.execute("SELECT `id` FROM `users` WHERE `user\_id` = ?", (user\_id,))

return bool(len(result.fetchall()))

def get\_user\_id(self, user\_id):

result = self.cursor.execute("SELECT `id` FROM `users` WHERE `user\_id` = ?", (user\_id,))

return result.fetchone()[0]

def add\_user(self, user\_id):

self.cursor.execute("INSERT INTO `users` (`user\_id`) VALUES (?)", (user\_id,))

return self.conn.commit()

def add\_record(self, user\_id, operation, value):

self.cursor.execute("INSERT INTO `records` (`users\_id`, `operation`, `value`) VALUES (?, ?, ?)",

(self.get\_user\_id(user\_id),

operation == "+",

value))

return self.conn.commit()

def get\_records(self, user\_id, within = "all"):

if within == "day":

result = self.cursor.execute("SELECT \* FROM `records` WHERE `users\_id` = ? AND `date` BETWEEN datetime('now', 'start of day') AND datetime('now', 'localtime') ORDER BY `date`",

(self.get\_user\_id(user\_id),))

elif within == "week":

result = self.cursor.execute("SELECT \* FROM `records` WHERE `users\_id` = ? AND `date` BETWEEN datetime('now', '-6 days') AND datetime('now', 'localtime') ORDER BY `date`",

(self.get\_user\_id(user\_id),))

elif within == "month":

result = self.cursor.execute("SELECT \* FROM `records` WHERE `users\_id` = ? AND `date` BETWEEN datetime('now', 'start of month') AND datetime('now', 'localtime') ORDER BY `date`",

(self.get\_user\_id(user\_id),))

else:

result = self.cursor.execute("SELECT \* FROM `records` WHERE `users\_id` = ? ORDER BY `date`",

(self.get\_user\_id(user\_id),))

return result.fetchall()

def close(self):

self.connection.close()

dispatcher.py

import logging

from aiogram import Bot, Dispatcher

from filters import IsOwnerFilter, IsAdminFilter, MemberCanRestrictFilter

import config

# Configure logging

logging.basicConfig(level=logging.INFO)

# prerequisites

if not config.BOT\_TOKEN:

exit("No token provided")

# init

bot = Bot(token=config.BOT\_TOKEN, parse\_mode="HTML")

dp = Dispatcher(bot)

# activate filters

dp.filters\_factory.bind(IsOwnerFilter)

dp.filters\_factory.bind(IsAdminFilter)

dp.filters\_factory.bind(MemberCanRestrictFilter)

filters.py

from aiogram import types

from aiogram.dispatcher.filters import BoundFilter

import config

class IsOwnerFilter(BoundFilter):

key = "is\_owner"

def \_\_init\_\_(self, is\_owner):

self.is\_owner = is\_owner

async def check(self, message: types.Message):

return message.from\_user.id == config.BOT\_OWNER

class IsAdminFilter(BoundFilter):

key = "is\_admin"

def \_\_init\_\_(self, is\_admin: bool):

self.is\_admin = is\_admin

async def check(self, message: types.Message):

member = await message.bot.get\_chat\_member(message.chat.id, message.from\_user.id)

return member.is\_chat\_admin() == self.is\_admin

class MemberCanRestrictFilter(BoundFilter):

key = 'member\_can\_restrict'

def \_\_init\_\_(self, member\_can\_restrict: bool):

self.member\_can\_restrict = member\_can\_restrict

async def check(self, message: types.Message):

member = await message.bot.get\_chat\_member(message.chat.id, message.from\_user.id)

return (member.is\_chat\_creator() or member.can\_restrict\_members) == self.member\_can\_restrict

personal\_actions.py

from aiogram import types

from dispatcher import dp

import config

import re

from bot import BotDB

@dp.message\_handler(commands = "start")

async def start(message: types.Message):

if(not BotDB.user\_exists(message.from\_user.id)):

BotDB.add\_user(message.from\_user.id)

await message.bot.send\_message(message.from\_user.id, "Добро пожаловать!")

@dp.message\_handler(commands = "help")

async def start(message: types.Message):

if(not BotDB.user\_exists(message.from\_user.id)):

BotDB.add\_user(message.from\_user.id)

await message.bot.send\_message(message.from\_user.id, '''Список команд:

/spent 2000 (или /s, или !spent, или !s) - сделать запись о расходе на сумму 2000

/earned 2000 (или /e, или !earned, или !e) - сделать запись о доходе на сумму 2000

/history (или /h, или !history, или !h) - вывести историю расходов пользователя за день

/history (или /h, или !history, или !h) today (или day, или today, или сегодня, или день) - вывести историю расходов за день

/history (или /h, или !history, или !h) month (или месяц) - вывести историю расходов за месяц

/history (или /h, или !history, или !h) year (или год) - вывести историю расходов за год

/help - список всех команд

--------------------------------------------

Список команд будет пополняться в дальнейшем

''')

@dp.message\_handler(commands = ("spent", "earned", "s", "e"), commands\_prefix = "/!")

async def start(message: types.Message):

cmd\_variants = (('/spent', '/s', '!spent', '!s'), ('/earned', '/e', '!earned', '!e'))

operation = '-' if message.text.startswith(cmd\_variants[0]) else '+'

value = message.text

for i in cmd\_variants:

for j in i:

value = value.replace(j, '').strip()

if(len(value)):

x = re.findall(r"\d+(?:.\d+)?", value)

if(len(x)):

value = float(x[0].replace(',', '.'))

BotDB.add\_record(message.from\_user.id, operation, value)

if(operation == '-'):

await message.reply("✅ Запись о <u><b>расходе</b></u> успешно внесена!")

else:

await message.reply("✅ Запись о <u><b>доходе</b></u> успешно внесена!")

else:

await message.reply("Не удалось определить сумму!")

else:

await message.reply("Не введена сумма!")

@dp.message\_handler(commands = ("history", "h"), commands\_prefix = "/!")

async def start(message: types.Message):

cmd\_variants = ('/history', '/h', '!history', '!h')

within\_als = {

"day": ('today', 'day', 'сегодня', 'день'),

"month": ('month', 'месяц'),

"year": ('year', 'год'),

}

cmd = message.text

for r in cmd\_variants:

cmd = cmd.replace(r, '').strip()

within = 'day'

if(len(cmd)):

for k in within\_als:

for als in within\_als[k]:

if(als == cmd):

within = k

records = BotDB.get\_records(message.from\_user.id, within)

if(len(records)):

answer = f"🕘 История операций за {within\_als[within][-1]}\n\n"

for r in records:

answer += "<b>" + ("➖ Расход" if not r[2] else "➕ Доход") + "</b>"

answer += f" - {r[3]}"

answer += f" <i>({r[4]})</i>\n"

await message.reply(answer)

else:

await message.reply("Записей не обнаружено!")