** Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет МГТУ им. Н.Э. Баумана)**

**Факультет «Информатика и системы управления» Кафедра «Системы обработки информации и управления»**

Лабораторная работа №6

**«Telegram бот на Python»**

по предмету

«Базовые компоненты интернет-технологий»

Выполнил:

студент группы № ИУ5-35Б

Обухов Антон

Проверил:

Преподаватель кафедры ИУ-5

Гапанюк Юрий

2022 г.

Постановка задачи

* **Цель лабораторной работы:** изучение разработки ботов в Telegram.

Текст программы

**baza.py**

import sqlite3

import config

import telebot

from telebot import types

from config import token

from datetime import date

bot = telebot.TeleBot(token)

class SQLt():

def \_\_init\_\_(self):

self.connection = sqlite3.connect("data.db")

self.cursor = self.connection.cursor()

def nachalo(self):

with self.connection:

self.cursor.execute('''CREATE TABLE user (id int)''')

self.cursor.execute("ALTER TABLE user ADD column 'uroven1' 'int'")

self.cursor.execute("ALTER TABLE user ADD column 'date' 'str'")

self.cursor.execute("ALTER TABLE user ADD column 'name' 'str'")

self.cursor.execute("ALTER TABLE user ADD column 'pmes' 'int'")

self.cursor.execute("ALTER TABLE user ADD column 'balance' 'int'")

self.cursor.execute('''CREATE TABLE oplata (id int)''')

self.cursor.execute("ALTER TABLE oplata ADD column 'code' 'int'")

self.cursor.execute("ALTER TABLE oplata ADD column 'status' 'int'")

self.cursor.execute("ALTER TABLE oplata ADD column 'summ' 'int'")

self.cursor.execute('''CREATE TABLE oplatac (id int)''')

self.cursor.execute("ALTER TABLE oplatac ADD column 'summ' 'int'")

def new\_oplata\_insert(self, user\_id, commet, skolko):

with self.connection:

self.cursor.execute(f"INSERT INTO oplata (id, code,status,summ) VALUES({user\_id},{commet},{0},{skolko})")

def counts\_users\_for(self, message):

with self.connection:

return self.cursor.execute(f"SELECT count(\*) FROM user WHERE id = {message.chat.id}").fetchone()[0]

def dobav(self):

self.cursor.execute("ALTER TABLE user ADD column 'date' 'date'")

def insert\_new\_user(self, id, name):

with self.connection:

q=str(date.today())

self.cursor.execute(f"INSERT INTO user (id,name,uroven1,date,pmes,balance)"

f"VALUES ({id},\"{name}\",{0},\"1,1,1\",0,0)")

def delete(self,ids):

with self.connection:

self.cursor.execute(f"DELETE FROM user WHERE id ={ids}")

def get\_uroven(self, ids):

with self.connection:

return self.cursor.execute(f"SELECT uroven1 from user where id ={ids}").fetchone()[0]

def get\_name(self, ids):

with self.connection:

return self.cursor.execute(f"SELECT name from user where id ={ids}").fetchone()[0]

def get\_date(self,ids):

with self.connection:

return self.cursor.execute(f"SELECT date from user where id ={ids}").fetchone()[0]

def pmes(self,ids):

with self.connection:

return self.cursor.execute(f"SELECT pmes from user where id ={ids}").fetchone()[0]

def set\_pmes(self,pmes,ids):

with self.connection:

self.cursor.execute("UPDATE user SET pmes = ? WHERE id = ?",(str(pmes),str(ids)))

def insert\_new\_oplatac(self, user\_id, skolko):

with self.connection:

self.cursor.execute(f"INSERT INTO oplatac (id,summ) VALUES({user\_id},{skolko})")

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

with self.connection:

return self.cursor.execute(f"select id from oplatac where id = {user\_id}").fetchone()[0]

def select\_summ\_oplatac(self, user\_id):

with self.connection:

return self.cursor.execute(f"SELECT summ FROM oplatac WHERE id = {user\_id}").fetchone()[0]

def info\_all\_user(self, user\_id):

with self.connection:

return self.cursor.execute(f"SELECT \* FROM user where id = {user\_id}").fetchone()

def select\_count\_oplatac(self, user\_id):

with self.connection:

return self.cursor.execute(f"select count(\*) from oplatac where id = {user\_id}").fetchone()[0]

def get\_numn(self, chat\_id):

with self.connection:

return self.cursor.execute(f"select summ from oplatac WHERE id = {chat\_id}").fetchone()[0]

def delete\_oplatac(self, user\_id):

with self.connection:

self.cursor.execute(f"DELETE from oplatac where id = {user\_id}")

def status\_from\_oplata(self, user\_id):

with self.connection:

return self.cursor.execute(f"select status from oplata where id = {user\_id}").fetchone()[0]

def update\_oplata\_status(self, user\_id):

with self.connection:

self.cursor.execute(f"UPDATE oplata SET status = {1} where id = {user\_id}")

def oplata\_select\_all(self, user\_id):

with self.connection:

return self.cursor.execute(f"SELECT \* FROM oplata WHERE id = {user\_id}").fetchone()

def get\_numn1(self, user\_id):

with self.connection:

return self.cursor.execute(f"select summ from oplata WHERE id = {user\_id}").fetchone()[0]

def updatestatuspay(self, ids, status):

with self.connection:

self.cursor.execute("UPDATE `users` SET `statuspay` = ? WHERE `id` = ?", (status, ids))

def update\_balance(self, user\_id, balance\_up):

with self.connection:

self.cursor.execute(f"UPDATE user SET balance = {balance\_up} WHERE id = {user\_id}")

def delete\_oplata(self, user\_id):

with self.connection:

self.cursor.execute(f"DELETE FROM oplata WHERE id = {user\_id}")

def getbalance(self, msid):

with self.connection:

self.cursor.execute(f"select balance from user WHERE id = {msid}")

return self.cursor.fetchone()[0]

def close(self):

self.connection.close()

**bot1.py**

import telebot

import requests

from telebot import types

from telebot.types import InputMediaPhoto

import sqlite3

import random

import string

import time

from random import randint, choice

import json

from random import randint

import threading

import config

from baza import SQLt

from datetime import date

from config import minimalka,maximalka,admins, pravila, user, token, soglashenie, userbtn1, userbtn2, userbtn3, userbtn4, userbtn5, userbtn6, nazad, tarifs, nazad\_s, free\_chan,otzyvy,otzyvy1,free\_chan1, tarifus, qiwinumber, token\_qiwi,oplata,proverit

bot = telebot.TeleBot(token)

admin = admins[0]

BD = SQLt()

@bot.message\_handler(commands=['start'])

def send\_welcome(message):

BD = SQLt()

if BD.counts\_users\_for(message) == 0:

BD.insert\_new\_user(message.chat.id, message.chat.username)

bot.send\_message(message.chat.id, "Тут типо должны быть правила, но пока их нет.", reply\_markup=soglashenie(pravila))

BD.close()

else:

gh = open('photo/Nachalo.jpg', "rb")

BD = SQLt()

bot.send\_photo(message.chat.id, gh, caption="Что тебе нужно?", reply\_markup=user())

BD.close()

@bot.message\_handler(content\_types=['text'])

def main\_message(message):

print(message)

if message.text == userbtn1:

BD = SQLt()

urov=BD.get\_uroven(message.chat.id)

if urov==0:

ger="У тебя нет тарифов"

elif urov==-1:

ger="У вас вечный тариф"

elif urov==1:

ger="Тариф: Доступ на 1 месяц"

elif urov==3:

ger="Тариф: Доступ на 3 месяца"

bot.send\_message(message.chat.id,ger, reply\_markup=nazad())

BD.close()

elif message.text == userbtn2:

BD = SQLt()

dat=list(map(int,BD.get\_date(message.chat.id).split(",")))

print(dat)

if dat==[1,1,1] and BD.get\_uroven(message.chat.id)!=-1:

ger="У вас нет тарифов, значит нет и времени действия"

elif dat==[1,1,1] and BD.get\_uroven(message.chat.id)==-1:

ger="У вас вечная подписка"

else:

ger="Доступ есть до "+str(date(list(dat)[0],list(dat)[1]+BD.get\_uroven(message.chat.id),list(dat)[2]))

bot.send\_message(message.chat.id,ger, reply\_markup=nazad())

BD.close()

elif message.text == userbtn3:

gh = open('photo/tarifs.jpg', "rb")

BD = SQLt()

rg=bot.send\_photo(message.chat.id, gh, caption="Выберите тариф для более подробной информации:", reply\_markup=tarifs())

BD.set\_pmes(rg.message\_id,message.chat.id)

BD.close()

elif message.text == userbtn4:

gh = open('photo/tarifs.jpg', "rb")

BD = SQLt()

rg=bot.send\_photo(message.chat.id, gh, caption="Ссылка на бесплатный канал:", reply\_markup=free\_chan(otzyvy))

BD.set\_pmes(rg.message\_id,message.chat.id)

BD.close()

elif message.text == userbtn5:

gh = open('photo/tarifs.jpg', "rb")

BD = SQLt()

rg=bot.send\_photo(message.chat.id, gh, caption="Ссылка на пополнение кошелька:", reply\_markup=free\_chan1(otzyvy1))

BD.set\_pmes(rg.message\_id,message.chat.id)

BD.close()

elif message.text == userbtn6:

markup = types.InlineKeyboardMarkup()

markup.add(types.InlineKeyboardButton(text="💳 Пополнить через банковскую карту", callback\_data="balanceqiwi"))

BD = SQLt()

gh = open('photo/tarifs.jpg', "rb")

rg=bot.send\_photo(message.chat.id, gh, caption="Ссылка на канал с отзывами:", reply\_markup=markup)

BD.set\_pmes(rg.message\_id,message.chat.id)

BD.close()

elif message.text == "Назад":

gh = open('photo/Nachalo.jpg', "rb")

BD = SQLt()

bot.send\_photo(message.chat.id, gh, caption="Что тебе нужно?", reply\_markup=user())

gh.close()

BD.close()

elif message.text=="Удались":

bot.send\_message(message.chat.id,"Я тебя удалил")

BD=SQLt()

BD.delete(message.chat.id)

BD.close()

elif message.text=="Qwerty":

bot.send\_message(message.chat.id,"Выполнено")

BD=SQLt()

BD.nachalo()

BD.close()

@bot.callback\_query\_handler(func=lambda call: True)

def callback\_inline(call):

if call.message:

if call.data == "prinyal":

gh = open('photo/Nachalo.jpg', "rb")

BD = SQLt()

bot.send\_photo(call.message.chat.id, gh, caption="Что тебе нужно?", reply\_markup=user())

BD.close()

elif call.data == "tarif1":

BD = SQLt()

bot.delete\_message(call.message.chat.id, BD.pmes(call.message.chat.id))

gh = open('photo/tarif1.jpg', "rb")

rg=bot.send\_photo(call.message.chat.id, gh, caption=f"Доступ к каналу на месяц, ничего необычного\nСтоимость {tarifus[0]} руб.", reply\_markup=nazad\_s())

BD.set\_pmes(rg.message\_id,call.message.chat.id)

BD.close()

elif call.data == "tarif2":

BD = SQLt()

bot.delete\_message(call.message.chat.id, BD.pmes(call.message.chat.id))

gh = open('photo/tarif2.jpg', "rb")

rg=bot.send\_photo(call.message.chat.id, gh, caption=f"Доступ к каналу на 3 месяца, ничего необычного \nСтоимость {tarifus[1]} руб.", reply\_markup=nazad\_s())

BD.set\_pmes(rg.message\_id,call.message.chat.id)

BD.close()

elif call.data == "nazad":

BD = SQLt()

bot.delete\_message(call.message.chat.id, BD.pmes(call.message.chat.id))

gh = open('photo/Nachalo.jpg', "rb")

rg=bot.send\_photo(call.message.chat.id, gh, caption="Что тебе нужно?", reply\_markup=user())

BD.set\_pmes(rg.message\_id,call.message.chat.id)

gh.close()

BD.close()

elif call.data == "nazad\_s":

gh = open('photo/tarifs.jpg', "rb")

BD = SQLt()

bot.delete\_message(call.message.chat.id, BD.pmes(call.message.chat.id))

rg=bot.send\_photo(call.message.chat.id, gh, caption="Выберите тариф для более подробной информации:", reply\_markup=tarifs())

BD.set\_pmes(rg.message\_id,call.message.chat.id)

BD.close()

elif call.data == "balanceqiwi":

bot.delete\_message(call.from\_user.id, call.message.message\_id)

bot.send\_message(call.message.chat.id, f"💰 Введите сумму пополнения: \n Минимальная сумма - {tarifus[0]} RUB", reply\_markup=nazad())

bot.register\_next\_step\_handler(call.message, popolni)

elif "zaplatit" in call.data:

user\_id=int(str(call.data).split("\_")[1])

try:

BD = SQLt()

inn = BD.status\_from\_oplata(int(user\_id))

if inn == 1:

BD.close()

bot.send\_message(call.message.chat.id, "ID Платежа не найден")

bot.register\_next\_step\_handler(call.message, prinyatieplateja)

else:

BD=SQLt()

isumm = BD.get\_numn1(user\_id)

ibn = BD.getbalance(user\_id)

BD.update\_oplata\_status(user\_id)

BD.update\_balance(user\_id, ibn + isumm)

skolko = isumm

mamont = BD.get\_name(user\_id)

bot.register\_next\_step\_handler(call.message, main\_message)

BD.close()

except:

pass

elif call.data == "prov":

try:

BD = SQLt()

paystatus = BD.status\_from\_oplata(call.message.chat.id)

if paystatus == 0:

user\_id = call.message.chat.id

QIWI\_TOKEN = token\_qiwi

QIWI\_ACCOUNT = str(qiwinumber)

s = requests.Session()

s.headers['authorization'] = 'Bearer ' + QIWI\_TOKEN

parameters = {'rows': '50'}

h = s.get('https://edge.qiwi.com/payment-history/v1/persons/' + QIWI\_ACCOUNT + '/payments', params=parameters)

try:

req = json.loads(h.text)

result = BD.oplata\_select\_all(user\_id)

comment = str(result[1])

for x in req['data']:

if str(x['comment']) == comment:

skolko = (x['sum']['amount'])

balancenow = BD.getbalance(call.message.chat.id)

BD.updatestatuspay(x['personId'],call.message.chat.id)

BD.update\_balance(call.message.chat.id, balancenow + skolko)

#cur.execute(f"SELECT username FROM users WHERE id = {wk}")

try:

bot.send\_message(admins[0], f"[{call.message.chat.first\_name}](tg://user?id={call.message.chat.id}) пополнил баланс на {skolko}RUB", parse\_mode='Markdown')

except:

pass

try:

bot.send\_message(admins[1], f"[{call.message.chat.first\_name}](tg://user?id={call.message.chat.id}) пополнил баланс на {skolko}RUB", parse\_mode='Markdown')

except:

pass

try:

bot.send\_message(call.message.chat.id, f"Ваш баланс пополнен.\n\nБаланс {balancenow+skolko} RUB", reply\_markup=user())

except:

pass

BD.close()

break

else:

bot.send\_message(call.message.chat.id, "⚠️Вы не оплатили⚠️\n\nОплатите заказ после чего нажмите \"Проверить оплату\"")

BD.close()

except:

bot.send\_message(call.message.chat.id, "⚠️Вы не оплатили⚠️\n\nОплатите заказ после чего нажмите \"Проверить оплату\"")

BD.close()

pass

else:

balancenow = BD.getbalance(call.message.chat.id)

skolko = BD.get\_numn1(call.message.chat.id)

BD.update\_balance(call.message.chat.id, balancenow + skolko)

BD.delete\_oplata(call.message.chat.id)

bot.send\_message(call.message.chat.id, f"Ваш баланс пополнен.\n\nБаланс {balancenow+skolko} RUB", reply\_markup=user())

BD.close()

except:

pass

else:

print(call.data)

@bot.message\_handler(content\_types=['text'])

def prinyatieplateja(message):

try:

if message.text == otmena:

bot.send\_message(message.chat.id, "Отменено", reply\_markup=user())

bot.register\_next\_step\_handler(message, main\_message)

else:

if message.text.isdigit():

BD = SQLt()

inn = BD.select\_count\_oplatac(int(message.text))

if inn == 0:

BD.close()

bot.send\_message(message.chat.id, "ID Платежа не найден\nНапишите правильный айди")

bot.register\_next\_step\_handler(message, prinyatieplateja)

else:

BD=SQLt()

user\_id = int(message.text)

isumm = BD.select\_summ\_oplatac(user\_id)

ibn = BD.getbalance(user\_id)

BD.update\_balance(user\_id, ibn + isumm)

skolko = isumm

wk = BD.worker\_code(user\_id)

workerusername = BD.get\_username(wk)

workername = BD.get\_name(wk)

mamont = BD.get\_name(user\_id)

bot.register\_next\_step\_handler(message, main\_message)

BD.close()

else:

bot.send\_message(message.chat.id, "Напишите число")

bot.register\_next\_step\_handler(message, prinyatieplateja)

except Exception as e:

raise

@bot.message\_handler(content\_types=['text'])

def popolni(message):

try:

if message.text.isdigit():

skolko = int(message.text)

if skolko >= minimalka and skolko <= maximalka:

BD = SQLt()

try:

BD.delete\_oplata(message.chat.id)

except Exception as e:

raise

comment = randint(10000, 9999999)

BD.new\_oplata\_insert(message.chat.id, comment, skolko)

wb = types.InlineKeyboardMarkup()

wb1 = types.InlineKeyboardButton(text="Заплатить", callback\_data=f'zaplatit\_{message.chat.id}')

wb.add(wb1)

bot.send\_message(admins[0], f"ID: `{message.chat.id}`\n\nЮзер [{message.chat.first\_name}](tg://user?id={message.chat.id}) создал заявку на пополнение\n\nСумма: {skolko}", reply\_markup=wb, parse\_mode='Markdown')

bot.send\_message(admins[1], f"ID: `{message.chat.id}`\n\nЮзер [{message.chat.first\_name}](tg://user?id={message.chat.id}) создал заявку на пополнение\n\nСумма: {skolko}", reply\_markup=wb, parse\_mode='Markdown')

statwusername12 = BD.info\_all\_user(message.chat.id)

BD.close()

link = f"https://qiwi.com/payment/form/99?extra%5B%27account%27%5D={qiwinumber}&amountInteger={skolko}&amountFraction=0&currency=643&extra%5B%27comment%27%5D={comment}&blocked[0]=sum&blocked[1]=account&blocked[2]=comment"

kb = types.InlineKeyboardMarkup()

kb1 = types.InlineKeyboardButton(text=oplata, callback\_data="site", url=link)

kb2 = types.InlineKeyboardButton(text=proverit, callback\_data='prov')

kb.add(kb1)

kb.add(kb2)

texttt = f'♻️ Оплата QIWI/банковской картой:\n[ОПЛАТА]({link})\n\n\*Сумма\* {skolko}₽\nКомментарий `{comment}`\n\n\_ВАЖНО! Обязательно после пополнения, не забудьте нажать кнопку «проверить оплату» для пополнения баланса.\_)'

gh = open("photo/popoln.jpg", "rb")

bot.send\_photo(message.from\_user.id, gh, caption=texttt, parse\_mode='Markdown', reply\_markup=kb)

gh.close()

else:

bot.send\_message(message.chat.id, f"❗️ Сумма пополнения должна быть от {tarifus[0]}")

bot.register\_next\_step\_handler(message, popolni)

elif message.text == "Назад":

bot.send\_message(message.chat.id, "Отменено", reply\_markup=user())

bot.register\_next\_step\_handler(message, main\_message)

else:

bot.send\_message(message.chat.id, "Напишите число")

bot.register\_next\_step\_handler(message, popolni)

except Exception as e:

raise

if \_\_name\_\_ == '\_\_main\_\_':

bot.polling(none\_stop=True)

**tests.py**

from behave import Given, When, Then

from equation import calculate

@Given("equation with coef A {A} B {B} C {C}")

def given\_increment(context, A: {str}, B: {str}, C: {str}):

context.A = int(A)

context.B = int(B)

context.C = int(C)

@When("we calculate {object}")

def given\_increment(context, object: {str}):

context.results = calculate(context.A, context.B, context.C)

@Then("we should see root1 {root1} root2 {root2} root3 {root3} root4 {root4}")

def then\_results(context, root1: {str}, root2: {str}, root3: {str}, root4: {str}):

if (root1 == "empty"):

assert (context.results == ())

elif (root2 == "empty"):

assert(context.results == (int(root1)))

elif (root3 == "empty"):

assert (context.results == (int(root1), int(root2)))

elif (root4 == "empty"):

assert (context.results == (int(root1), int(root2), int(root3)))

else:

assert(context.results == (int(root1), int(root2), int(root3), int(root4)))

**config.py**

from telebot import types

from telebot.types import InlineKeyboardMarkup, InlineKeyboardButton

token="Токен от бота"

admins=[Id\_admin1,Id\_admin2]

qiwinumber="Ваш номер"

token\_qiwi="Зачем он вам?"

qiwi=""

otzyvy="Ваш канал"

otzyvy1="Ваш канал"

pravila='Принять'

userbtn1 = "🖥Мой тариф🖥"

userbtn2 = "⏱Сколько осталось до конца⏱"

userbtn3 = "🤑Выбор тарифов🤑"

userbtn4 = "😘Мой бесплатный канал😘"

userbtn5 = "💬Отзывы💬"

userbtn6 = "🦾Покупка тарифов🦾"

oplata = "💳 Оплатить"

proverit = "📲 Проверить платёж"

tarifus = [1, 2, 3, 4]

minimalka=1

maximalka=3

def user():

k1 = types.ReplyKeyboardMarkup(True)

k1\_btn1 = types.KeyboardButton(userbtn1)

k1\_btn2 = types.KeyboardButton(userbtn2)

k1\_btn3 = types.KeyboardButton(userbtn3)

k1\_btn4 = types.KeyboardButton(userbtn4)

k1\_btn5 = types.KeyboardButton(userbtn5)

k1\_btn6 = types.KeyboardButton(userbtn6)

k1.add(k1\_btn1, k1\_btn2)

k1.add(k1\_btn3)

k1.add(k1\_btn4, k1\_btn5)

k1.add(k1\_btn6)

return k1

def free\_chan(otzyvy):

markup = types.InlineKeyboardMarkup()

markup.add(types.InlineKeyboardButton(text="Мы в телеграмм", url=otzyvy))

return markup

def free\_chan1(otzyvy):

markup = types.InlineKeyboardMarkup()

markup.add(types.InlineKeyboardButton(text="Мы в телеграмм", url=otzyvy))

return markup

def soglashenie(prinyat):

prinyatpravila = types.InlineKeyboardMarkup()

prinyatpravila\_btn1 = types.InlineKeyboardButton(text=prinyat, callback\_data="prinyal")

prinyatpravila.add(prinyatpravila\_btn1)

return prinyatpravila

def nazad():

k1 = types.ReplyKeyboardMarkup(True)

k1\_btn1 = types.KeyboardButton("Назад")

k1.add(k1\_btn1)

return k1

def nazad\_s():

act = types.InlineKeyboardMarkup()

activ\_btn3 = types.InlineKeyboardButton("Назад", callback\_data="nazad\_s")

act.add(activ\_btn3)

return act

def tarifs():

act = types.InlineKeyboardMarkup()

activ\_btn1 = types.InlineKeyboardButton("Тариф 1 месяц", callback\_data="tarif1")

activ\_btn2 = types.InlineKeyboardButton("Тариф 3 месяца", callback\_data="tarif2")

activ\_btn3 = types.InlineKeyboardButton("Назад", callback\_data="nazad")

act.add(activ\_btn1)

act.add(activ\_btn2)

act.add(activ\_btn3)

return act

Анализ результатов

Бот способен автоматически принимать оплату и выставлять счет, так что обслуживание дополнительным человеком нужно лишь в редких случаях.









