



**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 ИУ5 «Системы обработки информации и управления»**

**Лабораторная работа №6  
по дисциплине «Базовые компоненты интернет-технологий»**

**Выполнил:  
студент группы ИУ5-35Б  
Емельянова Т.И.**

## Задание:

1. Разработайте бота для Telegram. Бот должен реализовывать конечный автомат из трех состояний.

telegram bot/t bot.py

```
import telebot
from telebot import types
from telegram_bot import config, dbworker
from burger import JustBurger, CheeseBurger, DoubleCheeseBurger,
MiniCheeseBurger, Director, UserBuilder

# Создание бота
bot = telebot.TeleBot(config.TOKEN)

def create_buttons():
    markup = types.ReplyKeyboardMarkup(row_width=2)
    itembtn1 = types.KeyboardButton('Бургер')
    itembtn2 = types.KeyboardButton('Чизбургер')
    itembtn3 = types.KeyboardButton('Двойной чизбургер')
    itembtn4 = types.KeyboardButton('Маленький чизбургер')
    itembtn5 = types.KeyboardButton('Мой бургер')
    markup.add(itembtn1, itembtn2, itembtn3, itembtn4, itembtn5)
    return markup

# Начало диалога
@bot.message_handler(commands=['start'])
def cmd_start(message):
    bot.send_message(message.chat.id, 'Я умею делать целый бургер!')
    dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_SELECTED_BURGER.value)
    bot.send_message(message.chat.id, 'Начнём создание!')
    markup = create_buttons()
    bot.send_message(message.chat.id, 'Выберите пожалуйста действие',
reply_markup=markup)

@bot.message_handler(commands=['menu'])
def cmd_menu(message):
    text_start = 'Я выведу список всех возможных ингредиентов, которые вы можете
добавить\n'
    text_add = '🥒 - Огурцы, 5\n🍅 - Помидор, 5\n🧀 - Сыр чеддер, 5\n🧄 - Лук,
3\n🥗 - Салат, 7\n🍷 - Кетчуп, 5\n'
    text_end = 'Котлета всегда будет из курицы, а булочка из белого хлеба'
    text = text_start + text_add + text_end
    bot.send_message(message.chat.id, text)
```

```
dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_SELECTED_BURGER.value)
```

# По команде /reset будем сбрасывать состояния, возвращаясь к началу диалога

```
@bot.message_handler(commands=['reset'])
```

```
def cmd_reset(message):
```

```
    bot.send_message(message.chat.id, 'Сбрасываем результаты предыдущего  
ввода.')
```

```
    dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_SELECTED_BURGER.value)
```

```
    bot.send_message(message.chat.id, 'Начнём создание!')
```

```
    markup = create_buttons()
```

```
    bot.send_message(message.chat.id, 'Выберите пожалуйста действие',  
reply_markup=markup)
```

```
@bot.message_handler(func=lambda message: dbworker.get(
    dbworker.make_key(message.chat.id, config.CURRENT_STATE)) ==
config.States.STATE_SELECTED_BURGER.value)
```

```
def selected_burger(message):
```

```
    # Текущее действие
```

```
    op = message.text
```

```
    director = Director()
```

```
    builder = None
```

```
    if op == 'Бургер':
```

```
        builder = JustBurger()
```

```
    elif op == 'Чизбургер':
```

```
        builder = CheeseBurger()
```

```
    elif op == 'Двойной чизбургер':
```

```
        builder = DoubleCheeseBurger()
```

```
    elif op == 'Маленький чизбургер':
```

```
        builder = MiniCheeseBurger()
```

```
    elif op == 'Мой бургер':
```

```
        dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_MY_BURGER.value)
```

```
        bot.send_message(message.chat.id, 'Введите ингредиенты с помощью  
эмодзи')
```

```
        return
```

```
    else:
```

```
        bot.send_message(message.chat.id, 'Вы ввели данные не из списка.  
Попробуйте снова')
```

```
        return
```

```
    director.builder = builder
```

```
    director.create_burger()
```

```
    text = builder.burger.get_description(1)
```

```
    markup = types.ReplyKeyboardRemove(selective=False)
```

```
    text_all = ""
```

```
    for line in text:
```

```
        text_all += line + '\n'
```

```

bot.send_message(message.chat.id, text_all)
dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_SELECTED_BURGER.value)
bot.send_message(message.chat.id, 'Начнём создание!')
markup = create_buttons()
bot.send_message(message.chat.id, 'Выберите пожалуйста действие',
reply_markup=markup)

```

```

def get_additives(op):
    additives = []
    for ing in op:
        if ing == '🥒':
            additives.append('Огурцы')
        if ing == '🍅':
            additives.append('Помидор')
        if ing == '🧀':
            additives.append('Сыр чеддер')
        if ing == '🧄':
            additives.append('Лук')
        if ing == '🥬':
            additives.append('Салат')
        if ing == '🍷':
            additives.append('Кетчуп')
    return additives

```

```

@bot.message_handler(func=lambda message: dbworker.get(
    dbworker.make_key(message.chat.id, config.CURRENT_STATE)) ==
config.States.STATE_MY_BURGER.value)
def my_burger(message):
    op = message.text
    if len(op) > 20:
        bot.send_message(message.chat.id, 'Лимит ингредиентов превышен! Введите
данные снова')
        return
    additives = get_additives(op)
    director = Director()
    builder = UserBuilder(additives)
    director.builder = builder
    director.create_burger()
    text = builder.burger.get_description(1)
    markup = types.ReplyKeyboardRemove(selective=False)
    bot.send_message(message.chat.id, 'Итак, ваш бургер')
    text_all = ""
    for line in text:
        text_all += line + '\n'
    bot.send_message(message.chat.id, text_all)

```

```

        dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_SELECTED_BURGER.value)
        bot.send_message(message.chat.id, 'Начнём создание!')
        markup = create_buttons()
        bot.send_message(message.chat.id, 'Выберите пожалуйста действие',
reply_markup=markup)

```

```

if __name__ == '__main__':
    bot.infinity_polling()

```

telegram bot/config.py

```

from enum import Enum

```

```

# Токент бота

```

```

TOKEN = "5042483811:AAGLzGDse8j6tEBGypMEdfcHQC0I4Iqq7ZM"

```

```

# Файл базы данных Vedis

```

```

db_file = "../db.vdb"

```

```

# Ключ записи в БД для текущего состояния

```

```

CURRENT_STATE = "CURRENT_STATE"

```

```

# Состояния автомата

```

```

class States(Enum):

```

```

    STATE_START = "STATE_START" # Начало нового диалога

```

```

    STATE_SELECTED_BURGER = "STATE_SELECTED_BURGER"

```

```

    STATE_MY_BURGER = "STATE_MY_BURGER"

```

telegram bot/dbworker.py

```

from redis import Redis

```

```

from telegram_bot import config

```

```

# Чтение значения

```

```

def get(key):

```

```

    with Redis(config.db_file) as db:

```

```

        try:

```

```

            return db[key].decode()

```

```

        except KeyError:

```

```

            # в случае ошибки значение по умолчанию - начало диалога

```

```

            return config.States.STATE_START.value

```

```

# Запись значения

```

```

def set(key, value):

```

```

    with Redis(config.db_file) as db:

```

```

        try:

```

```

        db[key] = value
        return True
    except:
        # тут желательно как-то обработать ситуацию
        return False

```

**# Создание ключа для записи и чтения**

```

def make_key(chatid, keyid):
    res = str(chatid) + '__' + str(keyid)
    return res

```

telegram bot/requirements.txt

```

certifi==2021.10.8
charset-normalizer==2.0.8
Cython==0.29.24
idna==3.3
pyTelegramBotAPI==4.2.2
requests==2.26.0
urllib3==1.26.7
redis==4.3.4

```

burger.py

```

from abc import ABC, abstractmethod
from additives import *
from buns import TopBun, BottomBun

```

```

class Burger:
    def __init__(self):
        self._kind = bread_kinds.find("Белый хлеб")
        self._name = ''
        self._parts = []

```

```

@property
def kind(self):
    return self._kind

```

```

@kind.setter
def kind(self, kind):
    self._kind = kind

```

```

@property
def name(self):
    return self._name

```

```

@name.setter
def name(self, name):
    self._name = name

def add(self, part):
    self._parts.append(part)

def get_description(self, telegram=0):
    telegram_text = []
    if telegram == 0:
        print(self._name)
    for part in self._parts:
        if telegram == 0:
            print(part.name, part.get_price())
        telegram_text.append(part.name + ', ' + str(part.get_price()))
    if telegram == 0:
        print('Итого:', self.get_price())
    telegram_text.append('Итого: ' + str(self.get_price()))
    return telegram_text

def get_price(self):
    price = 0
    for part in self._parts:
        price += part.get_price()
    return price

class Builder(ABC):
    @property
    @abstractmethod
    def burger(self, **params):
        pass

    @abstractmethod
    def get_name(self):
        pass

    @abstractmethod
    def create_base(self, **params):
        pass

    @abstractmethod
    def create_ingredients(self, **params):
        pass

    @abstractmethod
    def create_bottom(self, **params):
        pass

```

```
class JustBurger(Builder):
    def __init__(self):
        self._burger = Burger()

    def reset(self):
        self._burger = Burger()

    @property
    def burger(self):
        burger = self._burger
        self.reset()
        return burger

    def get_name(self):
        self._burger.name = 'Простой бургер'

    def create_base(self):
        self._burger.add(TopBun(self._burger.kind, seasoning.find("Кунжут")))

    def create_ingredients(self):
        self._burger.add(base_additives.find("Салат"))
        self._burger.add(base_additives.find("Помидор"))
        self._burger.add(base_additives.find("Лук"))
        self._burger.add(base_additives.find("Огурцы"))
        self._burger.add(sauces.find("Кетчуп"))
        self._burger.add(base_additives.find("Сыр чеддер"))

    def create_bottom(self):
        self._burger.add(cutlets.find("Куриная котлета"))
        self._burger.add(BottomBun(self._burger.kind))

class CheeseBurger(JustBurger):
    def get_name(self):
        self._burger.name = 'Чизбургер'

    def create_base(self):
        self._burger.add(TopBun(self._burger.kind, seasoning.find("Кунжут")))

    def create_ingredients(self):
        self._burger.add(base_additives.find("Салат"))
        self._burger.add(base_additives.find("Помидор"))
        self._burger.add(base_additives.find("Лук"))
        self._burger.add(base_additives.find("Огурцы"))
        self._burger.add(sauces.find("Кетчуп"))
        self._burger.add(sauces.find("Горчица"))
        self._burger.add(base_additives.find("Сыр чеддер"))
```



```
def create_bottom(self):
self._burger.add(cutlets.find("Котлета из говядины"))
self._burger.add(BottomBun(self._burger.kind))
```

```
class MiniCheeseBurger(CheeseBurger):
def get_name(self):
self._burger.name = 'Маленький чизбургер'
```

```
def create_base(self):
self._burger.add(TopBun(self._burger.kind))
```

```
def create_ingredients(self):
self._burger.add(base_additives.find("Огурцы"))
self._burger.add(sauces.find("Кетчуп"))
self._burger.add(base_additives.find("Лук"))
self._burger.add(base_additives.find("Сыр чеддер"))
```

```
class DoubleCheeseBurger(CheeseBurger):
def get_name(self):
self._burger.name = 'Двойной чизбургер'
```

```
def create_ingredients(self):
self._burger.add(base_additives.find("Салат"))
self._burger.add(base_additives.find("Помидор"))
self._burger.add(base_additives.find("Лук"))
self._burger.add(base_additives.find("Огурцы"))
self._burger.add(sauces.find("Кетчуп"))
self._burger.add(sauces.find("Горчица"))
self._burger.add(base_additives.find("Сыр чеддер"))
self._burger.add(cutlets.find("Котлета из говядины"))
self._burger.add(base_additives.find("Лук"))
self._burger.add(base_additives.find("Сыр чеддер"))
```

```
class VeryCheeseBurger(JustBurger):
def get_name(self):
self._burger.name = 'Сырный бургер'
```

```
def create_ingredients(self):
self._burger.add(base_additives.find("Салат"))
self._burger.add(base_additives.find("Помидор"))
self._burger.add(base_additives.find("Лук"))
self._burger.add(base_additives.find("Огурцы"))
self._burger.add(sauces.find("Кетчуп"))
self._burger.add(sauces.find("Горчица"))
self._burger.add(base_additives.find("Сыр чеддер"))
```

```
self._burger.add(cutlets.find("Котлета из говядины"))
self._burger.add(base_additives.find("Лук"))
self._burger.add(base_additives.find("Сыр чеддер"))
```

```
class UserBuilder(Builder):
    def __init__(self, parts):
        self._burger = Burger()
        self.parts = parts
```

```
def reset(self):
    self._burger = Burger()
```

```
@property
def burger(self):
    burger = self._burger
    self.reset()
    return burger
```

```
def get_name(self):
    self._burger.name = 'Ваш бургер'
```

```
def create_base(self):
    self._burger.add(TopBun(self._burger.kind))
```

```
def create_ingredients(self):
    for part in self.parts:
        base_add = base_additives.find(part)
        sauce = sauces.find(part)
        if base_add.get_price() != 0:
            self._burger.add(base_additives.find(part))
        if sauce.get_price() != 0:
            self._burger.add(sauces.find(part))
    self._burger.add(cutlets.find('Куриная котлета'))
```

```
def create_bottom(self):
    self._burger.add(BottomBun(self._burger.kind))
```

```
class Director:
    def __init__(self):
        self._builder = None
```

```
@property
def builder(self):
    return self._builder
```

```
@builder.setter
def builder(self, builder):
```

```
self._builder = builder

def create_burger(self):
    self._builder.get_name()
    self._builder.create_base()
    self._builder.create_ingredients()
    self._builder.create_bottom()
```

ingredient.py

```
import sys
from abc import ABC, abstractmethod
```

```
class Ingredient(ABC):
```

```
@abstractmethod
def get_price(self):
    pass
```

burger additive.py

```
from ingredient import Ingredient
```

```
class BurgerAdditive(Ingredient):
    def __init__(self, name, price):
        self._name = name
        self._price = price
```

```
def get_price(self):
    return self._price
```

```
@property
def name(self):
    return self._name
```

buns.py

```
from ingredient import Ingredient
```

```

class Bun(Ingredient):
    def __init__(self, weight, kind, name):
        self._weight = weight
        self._kind = kind
        self._name = name

    def get_price(self):
        return self._weight * self._kind.get_price()

    @property
    def name(self):
        return self._name

```

```

class TopBun(Bun):
    def __init__(self, kind, additive=None):
        name = 'Верхняя булочка, ' + kind.name
        if additive is not None:
            name += ', ' + additive.name
        super().__init__(1, kind, name)
        self._additive = additive

    def get_price(self):
        price = super().get_price()
        if self._additive is not None:
            price += self._additive.get_price()
        return price

```

```

class BottomBun(Bun):
    def __init__(self, kind):
        super().__init__(0.5, kind, 'Нижняя булочка, ' + kind.name)

    def get_price(self):
        return super().get_price()

```

additives.py

```

from burger_additive import BurgerAdditive
from abc import ABC, abstractmethod
import itertools

```

```

class Components:
    def __init__(self, file_name):
        self.All = []
        self.file_name = file_name

```

```

with open(self.file_name, "r", encoding="utf-8") as file:
for name, cost in itertools.zip_longest(*[file]*2):
self.All.append(BurgerAdditive(name.rstrip(), int(cost)))

def print_all(self, telegram=0):
telegram_text = []
index = 1
for element in self.All:
text = "{}. {}, {}".format(index, element.name, element.get_price())
if telegram == 0:
print(text)
telegram_text.append(text)
index += 1
return telegram_text

def find(self, name):
for element in self.All:
if element.name == name:
return element
return BurgerAdditive("", 0)

```

```

base_additives = Components("additives/base_additives.txt")
sauces = Components("additives/sauces.txt")
bread_kinds = Components("additives/bread_kinds.txt")
cutlets = Components("additives/cutlets.txt")
seasoning = Components("additives/seasoning.txt")

```

additives/base additives.txt, telegram bot/additives/base additives.txt

```

Помидор
5
Огурцы
5
Лук
3
Салат
7
Сыр чеддер
5
Сыр пармезан
6
Бекон
7
Халапеньо
5

```

additives/bread kinds.txt, telegram bot/additives/bread kinds.txt

Белый хлеб  
10  
Чёрный хлеб  
20

Additives/cutlets.txt, telegram bot/additives/cutlets.txt

Куриная котлета  
20  
Котлета из говядины  
30

Additives/sauces.txt, telegram bot/additives/sauces.txt

Сырный соус  
5  
Кетчуп  
5  
Горчица  
5

Additives/seasoning.txt, telegram bot/additives/seasoning.txt

Кунжут  
5  
Перец  
5

Результаты выполнения работы t\_bot.py









