Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Отчет по лабораторной работе №5-6
«Разработка бота на основе конечного автомата для Telegram с использованием языка Python»

Выполнил: Ким Алексей Максимович ИУ5-32Б
Дата: 20.12.2021

Москва, 2021 г.

Постановка задачи:

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

Код программы:
Sub.py

```python
import sqlite3

class sub:

    def __init__(self, database):
        self.connection = sqlite3.connect(database)
        self.cursor = self.connection.cursor()

    def get_subscriptions(self, status = True):
        with self.connection:
            return self.cursor.execute("SELECT * FROM `subscriptions` WHERE `status` = ?", (status,)).fetchall()

    def subscriber_exists(self, user_id):
        with self.connection:
            result = self.cursor.execute('SELECT * FROM `subscriptions` WHERE `user_id` = ?', (user_id,)).fetchall()
            return bool(len(result))

    def subscriber_check(self, user_id):
        with self.connection:
            result = self.cursor.execute('SELECT `status` FROM `subscriptions` WHERE `user_id` = ?', (user_id,)).fetchall()
            return bool(len(result))

    def add_subscriber(self, user_id, status):
        with self.connection:
            return self.cursor.execute("INSERT INTO `subscriptions` (`user_id`, `status`) VALUES(?,?)", (user_id,status))

    def update_subscription(self, user_id, status):
        with self.connection:
            return self.cursor.execute("UPDATE `subscriptions` SET `status` = ? WHERE `user_id` = ?", (status, user_id))

    def close(self):
        self.connection.close()
```

Translate.py

```python
async def translate(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data[2] = float(message.text)
        await chosen_curr.next()
    async with state.proxy() as data:
        if data[0] == "USD":
            d = float(a[1].replace(',', '.'))
            if data[1] == "EUR":
                e = float(a[2].replace(',', '.'))
            else:
                e = 1
            f = data[2] * (d / e)
        elif data[0] == "EUR":
            d = float(a[2].replace(',', '.'))
            if data[1] == "USD":
                e = float(a[1].replace(',', '.'))
            else:
                e = 1
            f = data[2] * (d / e)
        elif data[0] == "RUB" and data[1] == "USD":
            d = float(a[1].replace(',', '.'))
            f = data[2]/d
        elif data[0] == "RUB" and data[1] == "EUR":
            d = float(a[2].replace(',', '.'))
            f = data[2] / d
    await message.answer(f'Переводим {data[2]} {data[0]} в {data[1]} и получаем {f} {data[1]}')
    await state.finish()


def register_handlers_translate(dp: Dispatcher):
    dp.register_message_handler(transtale_start, commands=['translate'], state=None)
    dp.register_message_handler(translate_curr1_chosen, state=chosen_curr.waiting_for_translate_curr1)
    dp.register_message_handler(translate_curr2_chosen, state=chosen_curr.waiting_for_translate_curr2)
    dp.register_message_handler(translate, state=chosen_curr.waiting_for_translate_num)
```

```python
from aiogram import Bot, Dispatcher, executor, types
from aiogram.dispatcher.filters import state

from bot import dp
from pasring import parsing
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters.state import State, StatesGroup

available_currency_names = ['USD', 'EUR', 'RUB']
data = []
a = parsing()

class chosen_curr(StatesGroup):
    waiting_for_translate_curr1 = State()
    waiting_for_translate_curr2 = State()
    waiting_for_translate_num = State()


async def transtale_start(message: types.Message):
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for name in available_currency_names:
        keyboard.add(name)
    await message.answer("Выберите валюту с какой хотите поменять: ", reply_markup=keyboard)
    await chosen_curr.waiting_for_translate_curr1.set()


async def translate_curr1_chosen(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data[0] = message.text
    await chosen_curr.next()
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for name in available_currency_names:
        keyboard.add(name)
    await message.answer("Выберите вторую валюту: ")
    await chosen_curr.waiting_for_translate_curr2.set()

async def translate_curr2_chosen(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data[1] = message.text
    await chosen_curr.next()
    await message.answer("Напишите сумму, которое нужно перевести: ", reply_markup=types.ReplyKeyboardRemove())
    await chosen_curr.waiting_for_translate_num.set()
```

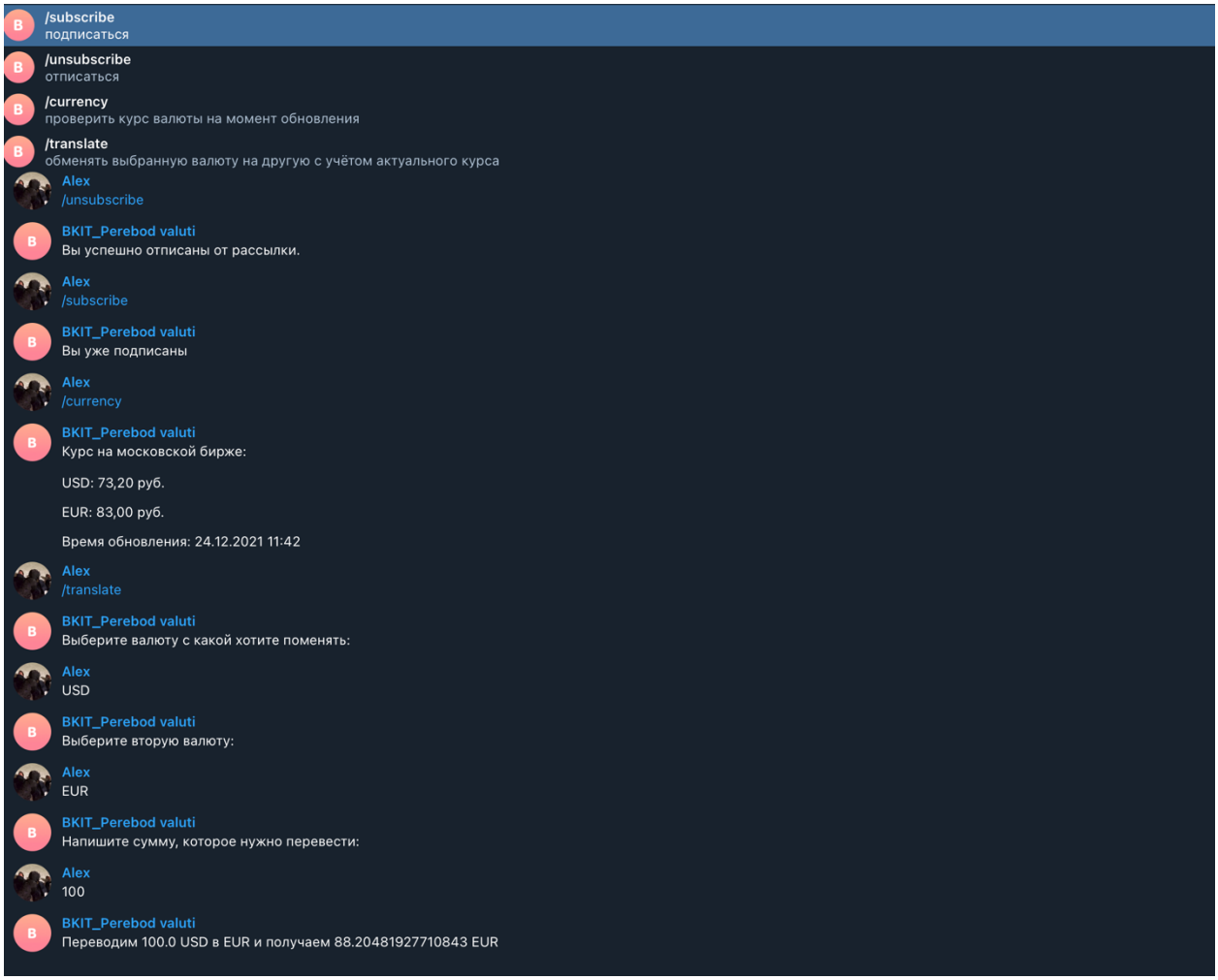
Pasring.py

```python
import aiogram.utils.executor
import requests
from bs4 import BeautifulSoup as BS
def parsing():
    a = []
    r = requests.get("https://www.banki.ru/products/currency/cash/moskva/")
    html = BS(r.content, 'html.parser')
    for el in html.select(".font-bold"):
        currency = el.find('span')
        try:
            a.append(currency.get_text())
        except AttributeError:
            continue
    for el in html.select(".table-flex__th"):
        time = el.find('div', class_='text-align-center')
        try:
            a.append(time.get_text())
        except AttributeError:
            continue
    return a
```

Bot.py

```python
import config
import logging
import translate
from aiogram.contrib.fsm_storage.memory import MemoryStorage
from pasring import parsing
from aiogram.types import BotCommand
from sub import sub
from aiogram import Bot, Dispatcher, executor, types
logging.basicConfig(level=logging.INFO)
storage = MemoryStorage()
bot = Bot(token=config.API_TOKEN)
dp = Dispatcher(bot, storage=storage)
db = sub('db.db')
async def set_commands(bot: Bot):
    commands = [
        BotCommand(command="/subscribe", description="Подписаться"),
        BotCommand(command="/unsubscribe", description="Отписаться"),
        BotCommand(command="/currency", description="Актуальный курс"),
        BotCommand(command="/translate", description="Обмен валюты")
    ]
    await bot.set_my_commands(commands)

@dp.message_handler(commands=['subscribe'])
async def subscribe(message: types.Message):
    if not db.subscriber_exists(message.from_user.id):
        db.add_subscriber(message.from_user.id)
        await message.answer(
            "Вы успешно подписались на рассылку!\n Теперь вы можете получать актуальные курсы валюты")
    elif not db.subscriber_check(message.from_user.id):
        db.update_subscription(message.from_user.id, True)
        await message.answer("Вы подписались снова!")
    else:
        await message.answer("Вы уже подписаны")

# Команда отписки
@dp.message_handler(commands=['unsubscribe'])
async def unsubscribe(message: types.Message):
    if not db.subscriber_exists(message.from_user.id):
        db.add_subscriber(message.from_user.id, False)
        await message.answer("Вы итак не подписаны.")
    else:
        db.update_subscription(message.from_user.id, False)
        await message.answer("Вы успешно отписаны от рассылки.")


@dp.message_handler(commands=['currency'])
async def currency(message: types.Message):
    if db.subscriber_exists(message.from_user.id) and db.subscriber_check(message.from_user.id):
        a = parsing()
        a[0] += ':'
        a[1] = 'USD: ' + a[1] + ' руб.'
        a[2] = 'EUR: ' + a[2] + ' руб.'
        for i in a:
            await message.answer(i)
    else:
        await message.answer(
            'Вы должны подписаться для получения актуальной информации.\nПропишите команду /subscribe и пользуйтесь любым предоставленным функционал


if __name__ == '__main__':
    translate.register_handlers_translate(dp)
    executor.start_polling(dp, skip_updates=False)
```

Результаты кода:


/subscribe
подписаться
/unsubscribe
отписаться
/currency
проверить курс валюты на момент обновления
/translate
обменять выбранную валюту на другую с учётом актуального курса
Alex
/unsubscribe
BKIT_Perebod valuti
Вы успешно отписаны от рассылки.
Alex
/subscribe
BKIT_Perebod valuti
Вы уже подписаны
Alex
/currency
BKIT_Perebod valuti
Курс на московской бирже:
USD: 73,20 руб.
EUR: 83,00 руб.
Время обновления: 24.12.2021 11:42
Alex
/translate
BKIT_Perebod valuti
Выберите валюту с какой хотите поменять:
Alex
USD
BKIT_Perebod valuti
Выберите вторую валюту:
Alex
EUR
BKIT_Perebod valuti
Напишите сумму, которое нужно перевести:
Alex
100
BKIT_Perebod valuti
Переводим 100.0 USD в EUR и получаем 88.20481927710843 EUR