**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №5-6

«Телеграмм-бот на Aiogram»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| Студент группы ИУ5-33Б |  | Преподаватель каф. ИУ5 |
| Нагапетян Валерий |  | Гапанюк Ю. Е. |
|  |  |  |

Москва, 2023 г.

**Описание работы**

Создание телеграмм-бота для анализа рынка, позволяющего пользователю узнать курс обмена фиатных валют. [API](https://coinmarketcap.com/api/documentation/v1/) для получения данных о курсе криптовалют.

Пользователь может ввести базовую и котируемую валюты и указать сумму для перевода. Также пользователь может изменить базовую или котируемую валюты, а также сумму для перевода.

Для написания телеграмм-бота используется библиотека **Aiogram**. Хранение данных, вводимых пользователем, осуществляется с помощью механизма конечных автоматов (**FSM**).

Для надежного хранения конфиденциальной информации, такой как токена бота, API-ключи, используются dotenv файлы и библиотека **Pydantic**.

**Файл bot.py**

**Текст программы**

*import* asyncio  
*import* logging  
  
*from* aiogram *import* Bot, Dispatcher, types, F  
*from* aiogram.filters *import* Command, StateFilter  
*from* aiogram.fsm.context *import* FSMContext  
*from* configs.config *import* configuration  
*from* keyboards *import* keyboards  
*from* FSM.StateMachine *import* ExchangeCurrency  
*from* FSM.StateMachine *import* Menu  
*from* api.api *import* api\_crypto  
  
logging.basicConfig(filename="../static/logger.txt", level=logging.INFO)  
dp = Dispatcher()  
bot = Bot(token=configuration.BOT\_TOKEN.get\_secret\_value(), parse\_mode="html")  
  
  
@dp.message(Command("start"))  
*async def* cmd\_start(message: types.Message, state: FSMContext):  
 *await* message.answer(f"Hello, {message.from\_user.username}! Welcome to NIVACryptoBot 📈\n\n"  
 f"Choose an option on your keyboard 📲",  
 reply\_markup=keyboards.main\_keyboard)  
 *await* state.set\_state(Menu.option)  
  
  
@dp.message(F.text.lower().in\_(['back']))  
*async def* back(message: types.Message, state: FSMContext, change\_flag: list[bool]):  
 *await* state.clear()  
 *if* change\_flag[0]:  
 change\_flag[0] = *False  
 await* state.set\_state(Menu.option)  
 *return await* message.answer(text='Back to Menu', reply\_markup=keyboards.main\_keyboard)  
  
  
@dp.message(Menu.option, F.text.in\_("Currency exchange prices"))  
*async def* menu\_option(message: types.Message, state: FSMContext):  
 *await* state.update\_data(option=message.text)  
 *await* message.answer("Choose a <i>base currency</i> on your keyboard 💱",  
 reply\_markup=keyboards.currency\_exchange\_keyboard())  
 *await* state.set\_state(ExchangeCurrency.base\_currency)  
  
  
@dp.message(ExchangeCurrency.base\_currency, F.text.in\_(keyboards.currencies))  
*async def* exchange\_target\_currency(message: types.Message, state: FSMContext, change\_flag: list[bool]):  
 *await* state.update\_data(chosen\_base\_currency=message.text.upper())  
 exchange = *await* state.get\_data()  
 *if not* change\_flag[0]:  
 *await* message.reply(f"You've chosen <b>{exchange['chosen\_base\_currency']}</b> as base currency. "  
 f"Now choose a <i>target currency</i> on your keyboard 💱",  
 reply\_markup=keyboards.currency\_exchange\_keyboard())  
 *await* state.set\_state(ExchangeCurrency.target\_currency)  
 *else*:  
 *await* message.reply("Please, set the currency amount for converting ⬇️")  
 *await* state.set\_state(ExchangeCurrency.amount)  
  
  
@dp.message(ExchangeCurrency.target\_currency, F.text.in\_(keyboards.currencies))  
*async def* exchange\_procedure(message: types.Message, state: FSMContext):  
 *await* state.update\_data(chosen\_target\_currency=message.text.upper())  
 exchange = *await* state.get\_data()  
 base\_currency = exchange['chosen\_base\_currency']  
 target\_currency = exchange['chosen\_target\_currency']  
 *await* message.reply(  
 f"You've chosen <b>{base\_currency}</b> as base currency"  
 f" and <b>{target\_currency}</b> as target currency.")  
 *await* message.answer(text='Please, set the currency amount for converting ⬇️')  
 *await* state.set\_state(ExchangeCurrency.amount)  
  
  
@dp.message(ExchangeCurrency.amount)  
*async def* currency\_amount(message: types.Message, state: FSMContext, change\_flag: list[bool]):  
 *await* state.update\_data(amount=message.text)  
 exchange = *await* state.get\_data()  
 amount\_for\_converse = exchange['amount']  
 *if* message.text.isdigit():  
 base\_currency = exchange['chosen\_base\_currency']  
 target\_currency = exchange['chosen\_target\_currency']  
  
 parameters = {  
 "amount": int(amount\_for\_converse),  
 "symbol": base\_currency,  
 "convert": target\_currency  
 }  
  
 response = api\_crypto(parameters)  
 conversion = response["data"][0]["quote"][target\_currency]["price"]  
 *await* message.answer(  
 f'You are going to converse <b>{float(amount\_for\_converse):,}</b> units of <b>{base\_currency}</b> into'  
 f' <b>{target\_currency}</b>\n\n'f"{float(amount\_for\_converse):,} <b>{base\_currency}</b> equals {conversion:,.2f} <b>{target\_currency}</b>",  
 reply\_markup=keyboards.currency\_exchange\_keyboard\_expanded())  
 change\_flag[0] = *False  
 return await* state.set\_state(ExchangeCurrency.next\_step)  
 *else*:  
 change\_flag[0] = *False  
 await* message.answer('Wrong data. Please, try again')  
 *return await* state.set\_state(ExchangeCurrency.amount)  
  
  
@dp.message(ExchangeCurrency.next\_step)  
*async def* next\_step(message: types.Message, state: FSMContext, change\_flag: list[bool]):  
 *await* state.update\_data(step=message.text)  
 *if* message.text.lower() == "change base currency":  
 change\_flag[0] = *True  
 await* state.set\_state(ExchangeCurrency.base\_currency)  
 *return await* message.answer(text='Choose a new base currency on your keyboard',  
 reply\_markup=keyboards.currency\_exchange\_keyboard())  
 *if* message.text.lower() == 'change target currency':  
 change\_flag[0] = *True  
 await* state.set\_state(ExchangeCurrency.target\_currency)  
 *return await* message.answer(text='Choose a new target currency on your keyboard',  
 reply\_markup=keyboards.currency\_exchange\_keyboard())  
  
  
@dp.message()  
*async def* wrong\_input(message: types.Message):  
 *await* message.answer('Wrong data. Please, try again')  
  
  
*async def* main():  
 *await* bot.delete\_webhook(drop\_pending\_updates=*True*)  
 *await* dp.start\_polling(bot, change\_flag=[*False*])  
  
  
*if* \_\_name\_\_ == "\_\_main\_\_":  
 asyncio.run(main())

**Файл keyboards.py**

**Текст программы**

*from* aiogram.types *import* (  
 InlineKeyboardMarkup,  
 InlineKeyboardButton,  
 ReplyKeyboardMarkup,  
 KeyboardButton  
)  
*from* aiogram.utils.keyboard *import* ReplyKeyboardBuilder  
  
main\_keyboard = ReplyKeyboardMarkup(  
 keyboard=[  
 [  
 KeyboardButton(text="Currency exchange prices"),  
 KeyboardButton(text="Cryptocurrency info"),  
 ]  
 ],  
 resize\_keyboard=*True*,  
 selective=*True*,  
 one\_time\_keyboard=*True*,  
 input\_field\_placeholder="Choose an option from menu..."  
)  
  
currencies = [  
 "USD", "EUR", "RUB",  
 "CNY", "JPY", "QAR",  
 "XAU", "XAG", "XPT"  
]  
  
  
*def* currency\_exchange\_keyboard():  
 keyboard = ReplyKeyboardBuilder()  
 [keyboard.button(text=fiat) *for* fiat *in* currencies]  
 keyboard.button(text='Back')  
 keyboard.adjust(\*[3] \* 3, 1)  
 *return* keyboard.as\_markup(resize\_keyboard=*True*)  
  
  
*def* currency\_exchange\_keyboard\_expanded():  
 keyboard = ReplyKeyboardBuilder()  
  
 [keyboard.button(text=fiat) *for* fiat *in* currencies]  
 keyboard.button(text='Change base currency')  
 keyboard.button(text='Change target currency')  
 keyboard.button(text='Back')  
 keyboard.adjust(\*[3] \* 4)  
 *return* keyboard.as\_markup(resize\_keyboard=*True*)

**Файл StateMachine.py**

**Текст программы**

*from* aiogram.fsm.state *import* StatesGroup, State  
  
  
*class* ExchangeCurrency(StatesGroup):  
 base\_currency: str = State()  
 target\_currency: str = State()  
 amount: str = State()  
 next\_step : str = State()  
  
  
*class* Menu(StatesGroup):  
 option: str = State()  
 menu = ["Currency exchange prices", "Cryptocurrency info"]

**Файл api.py**

**Текст программы**

*import* json  
*from* typing *import* Dict  
  
*from* requests *import* Session  
*from* configs.config *import* configuration  
  
api\_key\_coin = configuration.API\_KEY\_COIN.get\_secret\_value()  
api\_key\_crypto = configuration.API\_KEY\_CRYPTO.get\_secret\_value()  
  
  
*def* api\_crypto(parameters: Dict):  
 url = "https://pro-api.coinmarketcap.com/v2/tools/price-conversion"  
  
 headers = {  
 'Accepts': 'application/json',  
 'X-CMC\_PRO\_API\_KEY': api\_key\_crypto  
 }  
  
 session = Session()  
 session.headers.update(headers)  
 response = session.get(url, params=parameters)  
  
 *return* json.loads(response.text)

**Файл config.py**

**Текст программы**

*from* pydantic\_settings *import* BaseSettings, SettingsConfigDict  
*from* pydantic *import* SecretStr  
  
  
*class* Settings(BaseSettings):  
 BOT\_TOKEN: SecretStr  
 API\_KEY\_COIN: SecretStr  
 API\_KEY\_CRYPTO: SecretStr  
  
 model\_config = SettingsConfigDict(  
 env\_file="../static/.env", env\_file\_encoding="utf-8"  
 )  
  
  
configuration = Settings()

**Результаты**













