




Консультант Вячеслав (z... 22:23

Погода в moscow: Температура:
15.83 °C Состояние: пасмурно...

 Курсы валют

 Перейти в магазин 

/start 22:23 ✓✓

Используйте кнопки ниже или меню: 22:23

 Показать погоду

 Курсы валют


 Перейти в магазин 

Введите название города: 22:23

moscow 22:23 ✓✓

Погода в moscow:
Температура: 15.83 °C
Состояние: пасмурно
Влажность: 79 %
Ветер: 2.14 м/с

22:23

 Назад в меню




Консультант Вячеслав (z... 22:24

Город не найден, попробуйте еще раз:

Главное меню:

22:23

 Показать погоду

 Курсы валют

 Перейти в магазин



Введите название города: 22:24

gsfgsfghs 22:24 ✓✓

Город не найден, попробуйте еще раз: 22:24

moscow 22:23 ✓✓

К

Консультант Вячеслав (z... 22:25
Курсы ЦБ РФ: Доллар USA: 78.72 ₽
Евро: 90.31 ₽ Юань: 10.87 ₽

Курсы валют

Перейти в магазин ↗

Введите название города: 22:24

gsfgsfghs 22:24 ✓✓

Город не найден, попробуйте еще раз: 22:24

new york 22:25 ✓✓

Курсы ЦБ РФ:
Доллар USA: 78.72 ₽
Евро: 90.31 ₽
Юань: 10.87 ₽

22:25

Назад в меню

```

1 import logging
2 from my_token import TOKEN, OPENWEATHER_API_KEY, CBR_API_URL
3 import requests #Для выполнения HTTP-запросов к внешним API
4 from telegram.ext import Updater, CommandHandler, CallbackQueryHandler, CallbackContext, MessageHandler, Filters, Convers
5 from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup, ReplyKeyboardMarkup, KeyboardButton, ReplyKey
6
7
8 logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s', level=logging.INFO)
9 level=logging.INFO #Уровень логирования
10 logger = logging.getLogger(__name__) # Создание объекта логгера для текущего модуля
11
12 # Определение состояний для бота (Conversation Handler)
13 WAIT_CITY, SHOW_INFO = range(2) # Состояние диалога: ожидание города и показ информации
14
15 def create_main_menu_keyboard():
16     keyboard = [
17         [InlineKeyboardButton("☀ Показать погоду", callback_data='weather')],
18         [
19             InlineKeyboardButton("💱 Курсы валют", callback_data='currency'),
20             InlineKeyboardButton("🛒 Перейти в магазин", url="https://zerocoder.ru")
21         ]
22     ]
23
24     return InlineKeyboardMarkup(keyboard) # Создаем разметку клавиатуры
25
26 def start(update: Update, context: CallbackContext) -> None:
27     user = update.effective_user
28     update.message.reply_text(
29         "Используйте кнопки ниже или меню: ",
30         reply_markup=create_main_menu_keyboard()
31     )
32
33 def button_click(update: Update, context: CallbackContext) -> None:
34     query = update.callback_query # Получаем информацию о нажатии
35     query.answer() # Подтверждаем получение callback (убирает "часики" у кнопки)
36

```



```

37     if query.data == "weather":
38         query.message.reply_text(
39             |     "Введите название города:",
40             |     reply_markup=ReplyKeyboardRemove() # Удаление клавиатуры
41         )
42         return WAIT_CITY # Переход в состояние ожидания города
43     elif query.data == 'currency':
44         show_currency_rates(query) # Вызов функции показа курсов валют
45         return SHOW_INFO # Переход в состояние показа информации
46     elif query.data == 'back_to_menu':
47         query.edit_message_text(
48             |     text="Главное меню: ",
49             |     reply_markup=create_main_menu_keyboard()
50         )
51         return ConversationHandler.END # Завершение диалога
52     elif query.data == 'close':
53         query.delete_message()
54
55 def cancel (update:Update, context:CallbackContext) ->int:
56     """Отмена текущего действия"""
57     update.message.reply_text("Действие отменено", reply_markup=create_main_menu_keyboard()) # Возврат основной клавиатуры
58
59
60 def get_weather(update:Update, context:CallbackContext) ->int:
61     city = update.message.text # Получение города из сообщения пользователя
62     url = f"https://api.openweathermap.org/data/2.5/weather?q={city}&appid={OPENWEATHER_API_KEY}&units=metric&lang=ru"
63     try:
64         response = requests.get(url) # Отправка запроса
65         data = response.json() # Парсинг ответа в словарь
66         if response.status_code ==200: # Проверка на успешность запроса
67             weather_info = (
68                 |     f" Погода в {city}:\n"
69                 |     f" Температура: {data['main']['temp']} °C\n"
70                 |     f" Состояние: {data['weather'][0]['description']} \n"
71                 |     f" Влажность: {data['main']['humidity']} %\n"
72                 |     f" Ветер: {data['wind']['speed']} м/с"
73             )

```

```

74         # Кнопка возврата
75         keyboard = [[InlineKeyboardButton("⬅️ Назад в меню", callback_data="back_to_menu")]]
76         update.message.reply_text(weather_info, reply_markup=InlineKeyboardMarkup(keyboard))
77         return SHOW_INFO # Переход в состояние показа информации
78     else:
79         # Если город не найден
80         update.message.reply_text(" Город не найден, попробуйте еще раз: ")
81         return WAIT_CITY # Повторный запрос города
82 except Exception as e:
83     logger.error(f"Ошибка при получении информации о погоде: {e}") # Логирование ошибки
84     update.message.reply_text("Произошла ошибка. Попробуйте позже")
85     return ConversationHandler.END #Завершение диалога
86
87 > def show_currency_rates(query): --
105
106
107 def main():
108     updater = Updater(TOKEN)
109     dispatcher = updater.dispatcher
110
111     conv_handler = ConversationHandler(
112         entry_points=[CallbackQueryHandler(button_click, pattern='^weather$')],
113         states={
114             WAIT_CITY: [MessageHandler(Filters.text & ~Filters.command, get_weather)],
115             # Ожидание города
116             SHOW_INFO: [CallbackQueryHandler(button_click)] # Состояние показа информации
117         },
118         fallbacks=[CommandHandler('cancel', cancel)], # Резервный обработчик отмены
119         allow_reentry=True # Разрешение на повторный диалог
120     )
121     dispatcher.add_handler(conv_handler) # Диалог погоды
122     dispatcher.add_handler(CallbackQueryHandler(button_click, pattern='^(currency|back_to_menu|close)$')) # через кноп
123     dispatcher.add_handler(CommandHandler("start", start)) # через /start
124     dispatcher.add_handler(CallbackQueryHandler(button_click)) # через кнопку
125

```



```
125
126
127     updater.start_polling()
128     logger.info("Бот запущен и готов к работе")
129
130 if __name__ == '__main__':
131     main()
```