Бот-байер

Выполнили студенты
НММбд-02-23:
Гозенко Артем,
Овчинников Антон,
Иванов Александр

Актуальность

В связи с введенными санкциями, жители нашей страны столкнулись с проблемой покупок вещей с иностранных платформ. Наш бот позволяет сделать заказ наиболее простым способом для покупателя

Главной целью является комфортное взаимодействие пользователя с ботом при совершение покупок. Бот позволяет оптимизировать процесс заказа вещей через параллельный импорт

Обзор библиотек и пакетов

Бот реализован при использовании библиотеки Telebot, которая отлично подходит для несложных проектов работающих с Telegram

Кроме того код парсит курс валюты с сайта центробанка, для этого мы использовали библиотеку request

Для хранения данных пользователей используются методы sqlite3

При прохождение капчи выбирается произвольный вариант из списка при помощи math.randint

Капча

```python
def get_cny_to_rub_rate():
    # URL Центробанка для получения курсов
    url = "https://www.cbr.ru/scripts/XML_daily.asp"

    # Отправляем запрос
    response = requests.get(url)
    if response.status_code != 200:
        raise Exception("Ошибка при получении данных от Центробанка")

    # Парсим XML
    tree = ET.fromstring(response.content)

    # Ищем курс для юаня (код валюты: CNY)
    for currency in tree.findall("Valute"):
        if currency.find("CharCode").text == "CNY":
            rate = currency.find("Value").text.replace(",", ".")
            return float(rate)

    raise Exception("Курс юаня не найден")
```

Парсинг курса

```
def ChekUser(message):

    markup = types.InlineKeyboardMarkup()

    capcha_one = types.InlineKeyboardButton(text=callback_capcha[0], callback_data=callback_capcha[0])
    capcha_two = types.InlineKeyboardButton(text=callback_capcha[1], callback_data=callback_capcha[1])
    capcha_tree = types.InlineKeyboardButton(text=callback_capcha[2], callback_data=callback_capcha[2])
    capcha_four = types.InlineKeyboardButton(text=callback_capcha[3], callback_data=callback_capcha[3])
    capcha_five = types.InlineKeyboardButton(text=callback_capcha[4], callback_data=callback_capcha[4])

    markup.add(capcha_one, capcha_two, capcha_tree, capcha_four, capcha_five)

    markup.row_width = 2

    capcha_id = randint(0, 4)

    capcha = callback_capcha[capcha_id]

    bd.SetCapcha(message.chat.id, capcha_id)

    print("[Set Capcha]", capcha_id, capcha)

    bot.send_message(message.chat.id, text="Для обеспечения безопастности, необходимо пройти проверку\n"
                                           "Для этого, найдите и выберите одинаковый изобраения\n"
                                           f"{capcha}",
                     reply_markup=markup
    )
```

База данных

```python
import sqlite3

# Подключение к базе данных
# conn = sqlite3.connect('BotDataBase.db')  # Укажите путь, если база в другом месте
# cursor = conn.cursor()

# Функция для создания соединения с базой данных
def create_connection():
    try:
        # Устанавливаем соединение с базой данных
        conn = sqlite3.connect('BotDataBase.db', check_same_thread=False)  # Для многопоточных приложений
        return conn
    except sqlite3.Error as e:
        print(f"Ошибка соединения с базой данных: {e}")
        return None

# Функция для выполнения операций с базой данных в рамках контекста
def execute_query(query, params=()):
    conn = create_connection()
    if conn:
        try:
            with conn:
                cursor = conn.cursor()
                cursor.execute(query, params)
                return cursor.fetchall()
        except sqlite3.Error as e:
            print(f"Ошибка при выполнении запроса: {e}")
        finally:
            conn.close()

def GetAcsess(id_tg):
    # Проверяем, существует ли пользователь в базе данных
    query = "SELECT EXISTS(SELECT 1 FROM base_user WHERE telegramm_id = ?)"
```

```python
def GetAcsess(id_tg):
    # Проверяем, существует ли пользователь в базе данных
    query = "SELECT EXISTS(SELECT 1 FROM base_user WHERE telegramm_id = ?)"
    print("[GetAcsess]", id_tg)
    result = execute_query(query, (id_tg,))
    print("[GetAcsess]", result)
    if result:
        print("[GetAcsess]", result[0][0])
        return result[0][0] == 1
    return False


def NewUserNFT(id_tg, teg, name = "", clothing_size = "", shoe_size = ""):  # Добавление нового пользователя
    if GetAcsess(id_tg):
        query = """
        UPDATE base_user
        SET name = ?, clothing_size = ?, shoe_size = ?
        WHERE telegramm_id = ?
        """
        params = (name, clothing_size, shoe_size, id_tg)
        execute_query(query, params)
        return False
    else:
        query = """
        INSERT INTO `base_user`(`telegramm_id`, `telegramm_url`, `name`, `clothing_size`, `shoe_size`) VALUES (?, ?, ?, ?, ?)
        """
        params = (id_tg, teg, name, clothing_size, shoe_size)
        execute_query(query, params)
        return True

def GetCapcha(id):
    query = "SELECT capcha_id FROM desired_purchase WHERE telegramm_id = ?"
    result = execute_query(query, (id,))
```

```python
def GetCapcha(id):  1 usage
    query = "SELECT capcha_id FROM desired_purchase WHERE telegramm_id = ?"
    result = execute_query(query, params: (id,))
    if result:
        return result[0][0]  # Возвращаем capcha_id из первой строки результата


def SetCapcha(id, capcha_id):  1 usage

    query = "SELECT EXISTS(SELECT telegramm_id FROM desired_purchase WHERE telegramm_id = ?)"
    result = execute_query(query, params: (id,))
    if result and result[0][0] == 0:
        # Запись не существует, делаем INSERT
        query = "INSERT INTO desired_purchase (telegramm_id, capcha_id) VALUES (?, ?)"
        execute_query(query, params: (id, capcha_id))
    else:
        # Запись существует, делаем UPDATE
        query = "UPDATE desired_purchase SET capcha_id = ? WHERE telegramm_id = ?"
        execute_query(query, params: (capcha_id, id))

```

```python

def GetDataUser(id):  1 usage
    query = "SELECT name, clothing_size, shoe_size FROM base_user WHERE telegramm_id = ?"
    result = execute_query(query,  params: (id,))
    if result:
        name, clothing_size, shoe_size = result[0]
        return name, clothing_size, shoe_size
    return None

def NewOrder(id, parset_data, coast_in_rub):  1 usage
    query = "SELECT MAX(number_order) FROM orders"
    result = execute_query(query)
    if result and result[0][0]:
        last_order_id = result[0][0]
    else:
        last_order_id = 0

    new_order_id = last_order_id + 1
    query = """
        INSERT INTO orders (number_order, telegramm_id, size, type_clothes, url, coast_in_yuan, coast_in_rub, status_order)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?)
    """
    params = (new_order_id, id, parset_data["size"], parset_data["clothing_type"], parset_data["link"], parset_data["price_in_yuan"], coast_in_rub, "NEW ORDER")
    execute_query(query, params)
    return new_order_id
```

СПАСИБО ЗА ВНИМАНИЕ!!!