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 Домашнему заданию
«Telegram-бот Магазин Товаров»

Выполнил:
студент группы ИУ5-33Б
Сикоринский Артемий

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2024 г.

Постановка задачи

Реализовать telegram-бота с возможность размещения объявлений о продаже товаров и просмотром размещенных объявлений.

Код программы

Main.py

```
from bot import src

from token import token

from token import token
if __name__ == '__main__':
    TOKEN = token
    product_bot = src.ProductBot(TOKEN)
    product_bot.run()
```

Src.py

```
import telebot
import time
import logging

from data import sql_data
from telebot import types

from bot import callback_handler
from bot import message_handler
from bot import users
from bot import state_handler

class ProductBot:
    def __init__(self, token):
        self.bot = telebot.TeleBot(token)
        self.products = sql_data.sql_manager("data\products.db")
        self.user = users.User()
        self.CallbackHandlers = callback_handler.CallbackHandlers(self.bot,
self.products, self.user)
        self.MessageHandlers = message_handler.MessageHandlers(self.bot,
self.products, self.user)
        self.StateHandlers = state_handler.StateHandlers(self.bot,
self.products, self.user)


self.bot.message_handler(commands=['start'])(self.MessageHandlers.start)

self.bot.message_handler(commands=['add'])(self.MessageHandlers.add_product)

        self.bot.message_handler(func=lambda message:message.text =="Список
```

```
объявлений")(self.MessageHandlers.list_products)
        self.bot.message_handler(func=lambda message:message.text =="Мои
объявления")(self.MessageHandlers.list_users_ads)
        self.bot.message_handler(func=lambda message:message.text
=="Разместить объявление")(self.MessageHandlers.add_user_product)

        self.bot.message_handler(func=lambda message:
self.bot.get_state(message.from_user.id) ==
"PRICE")(self.StateHandlers.edit_product_price)
        self.bot.message_handler(func=lambda message:
self.bot.get_state(message.from_user.id) ==
"NAME")(self.StateHandlers.edit_product_name)
        self.bot.message_handler(func=lambda message:
self.bot.get_state(message.from_user.id) ==
"COMMENT")(self.StateHandlers.edit_product_comment)
        self.bot.message_handler(func=lambda message:
self.bot.get_state(message.from_user.id) ==
"CONTACT")(self.StateHandlers.edit_product_contact_user)

        self.bot.message_handler(func=lambda message:
self.bot.get_state(message.from_user.id) == "ADD-
PRICE")(self.StateHandlers.add_product_price)
        self.bot.message_handler(func=lambda message:
self.bot.get_state(message.from_user.id) == "ADD-
NAME")(self.StateHandlers.add_product_name)
        self.bot.message_handler(func=lambda message:
self.bot.get_state(message.from_user.id) == "ADD-
COMMENT")(self.StateHandlers.add_product_comment)
        self.bot.message_handler(func=lambda message:
self.bot.get_state(message.from_user.id) == "ADD-
CONTACT")(self.StateHandlers.add_product_contact_user)

        self.bot.message_handler(func=lambda
message:True)(self.MessageHandlers.check_text)
        #self.bot.message_handler(func=lambda message:
self.bot.get_state(message.from_user.id) in
["USER_PRODUCTS"])(self.StateHandlers.open_page)

        self.bot.callback_query_handler(func=lambda
call:call.data.split('_')[0] in
['back'])(self.CallbackHandlers.callback_meny_keyboards)
        self.bot.callback_query_handler(func=lambda
call:call.data.split('_')[0] in ['info-product', 'edit-
product'])(self.CallbackHandlers.callback_get_info_product)
        self.bot.callback_query_handler(func=lambda
call:call.data.split('_')[0] in ['nextpage',
'prevpage'])(self.CallbackHandlers.callback_page_creator)
        self.bot.callback_query_handler(func=lambda
call:call.data.split('_')[0] in ['delete',
'edit'])(self.CallbackHandlers.callback_user_ads)
        self.bot.callback_query_handler(func=lambda
call:call.data.split('_')[0] in ['setprice','setname', 'setcomment',
'setcontact'])(self.CallbackHandlers.callback_set_details)

    def run(self):
        #self.bot.polling(none_stop=True)
        while True:
            try:
                #self.bot.polling(none_stop=True)
                self.bot.infinity_polling(timeout=10, long_polling_timeout =
10)
            except Exception as e:
                logging.error(e)
                time.sleep(2)
```

User.py

```
class User:
    def __init__(self):
        self.data_user = {}

    def get_user_data(self, user_id):
        if user_id not in self.data_user:
            self.create_data(user_id)
        return self.data_user[user_id]

    def create_data(self, user_id):
        if user_id not in self.data_user:
            self.data_user[user_id] = {
                'prev_state':'START',
                'id':0,
                'page':0
            }
    def get_prev_state(self, user_id):
        return self.get_user_data(user_id)['prev_state']

    def set_data(self, user_id, key, value):
        self.get_user_data(user_id)[key] = value

    def get_id(self, user_id):
        return self.get_user_data(user_id)['id']

    def set_id(self, user_id, id_value):
        self.get_user_data(user_id)['id'] = id_value

    def set_prev_state(self, user_id, prev_state):
        self.get_user_data(user_id)['prev_state'] = prev_state
```

State_handler.py

```
from telebot import types

class StateHandlers:
    def __init__(self, bot, products, user):
        self.bot = bot
        self.products = products
        self.user = user
    def edit_product_price(self, message):
      price = message.text
      user_data = self.user.get_user_data(message.from_user.id)
      if price.isdigit():
          self.products.set_product("price", int(price), user_data['id'])
          self.bot.delete_state(message.from_user.id)
          self.bot.send_message(message.from_user.id, 'Цена на товар
измена.')
      else:
          self.bot.send_message(message.from_user.id, 'Неверный формат цены.
Пожалуйста, введите число.')
```

```
    def edit_product_name(self, message):
      name = message.text
      user_data = self.user.get_user_data(message.from_user.id)
      self.products.set_product("name", name, user_data['id'])
      self.bot.delete_state(message.from_user.id)
      self.bot.send_message(message.from_user.id, 'Название товара измено.')

    def edit_product_comment(self, message):
      comment = message.text
      user_data = self.user.get_user_data(message.from_user.id)
      self.products.set_product("comment", comment, user_data['id'])
      self.bot.delete_state(message.from_user.id)
      self.bot.send_message(message.from_user.id, 'Описание товара измено.')

    def edit_product_contact_user(self, message):
        contact_user = message.text
        user_data = self.user.get_user_data(message.from_user.id)
        self.products.set_product("contact_user", contact_user,
user_data['id'])
        self.bot.delete_state(message.from_user.id)
        self.bot.send_message(message.from_user.id, 'Ваши данные для связи
изменены.')

    def add_product_name(self, message):
        name = message.text
        user_data = self.user.get_user_data(message.from_user.id)
        self.products.set_product("name", name, user_data['id'])
        self.bot.send_message(message.from_user.id, 'Добавьте фото товара:')
        self.bot.set_state(message.from_user.id, "ADD-PHOTO")

    @self.bot.message_handler(content_types=['photo'])
    def add_product_photo(self, message):
        photo = message.text
        user_data = self.user.get_user_data(message.from_user.id)
        self.products.set_product("photo_id", photo, user_data['id'])
        self.bot.send_message(message.from_user.id, 'Укажите цену товара:')
        self.bot.set_state(message.from_user.id, "ADD-PRICE")

    def add_product_price(self, message):
        price = message.text
        user_data = self.user.get_user_data(message.from_user.id)
        if price.isdigit():
            self.products.set_product("price", int(price), user_data['id'])
            self.bot.delete_state(message.from_user.id)
            self.bot.send_message(message.from_user.id, 'Укажите описание
товара:')
            self.bot.set_state(message.from_user.id, "ADD-COMMENT")
        else:
            self.bot.send_message(message.from_user.id, 'Неверный формат
цены. Пожалуйста, введите число.')
    def add_product_comment(self, message):
        comment = message.text
        user_data = self.user.get_user_data(message.from_user.id)
        self.products.set_product("comment", comment, user_data['id'])
        self.bot.delete_state(message.from_user.id)
        self.bot.send_message(message.from_user.id, 'Укажите контакты для
связи с вами:')
        self.bot.set_state(message.from_user.id, "ADD-CONTACT")

    def add_product_contact_user(self, message):
        contact_user = message.text
        user_data = self.user.get_user_data(message.from_user.id)
        self.products.set_product("contact_user", contact_user,
user_data['id'])
```

```
        self.bot.delete_state(message.from_user.id)
        self.bot.send_message(message.from_user.id, 'Ваши данные для связи
изменены.')
```

message_handler.py

```
from telebot import types
import re
from datetime import datetime
from data import sql_data
from bot import create_message_data
class MessageHandlers():
    def __init__(self, bot, products, user):
        self.bot = bot
        self.create_message = create_message_data.CreateMessage(products,
user)
        self.products = products
        self.user = user

    def start(self, message):
        menu = types.ReplyKeyboardMarkup(resize_keyboard=True)
        list = types.KeyboardButton("Список объявлений")
        my_ads = types.KeyboardButton("Мои объявления")
        add_ads = types.KeyboardButton("Разместить объявление")
        menu.add(list,my_ads, add_ads)
        self.bot.reply_to(message, 'Привет! Я бот для размещения товаров. ',
reply_markup=menu)

    def add_product(self, message):
      match = re.match(r'/add\s+(.+?)\s+(\d+(\.\d+)?)$', message.text)
      if match:
          product_name = match.group(1)
          product_price = int(match.group(2))
          product_comment = 'Нет описания'
          product_contact = "Нет контактных данных"
          message_date = datetime.fromtimestamp(message.date)
          product = {
               'user': message.from_user.id,
               'name': product_name,
               'comment': product_comment,
               'price': product_price,
               'date': str(message_date.strftime('%Y-%m-%d %H:%M:%S')),
               'contact_user': product_contact
          }
          id = self.products.add_new_product(product)
          keyboard = types.InlineKeyboardMarkup()
          keyboard.add(types.InlineKeyboardButton(text="Цена",
callback_data=f"setprice_{id}"),
          types.InlineKeyboardButton(text=f'Название',
callback_data=f"setname_{id}"),
          types.InlineKeyboardButton(text=f'Описание',
callback_data=f"setcomment_{id}"),
          types.InlineKeyboardButton(text=f'Контакты',
callback_data=f"setcontact_{id}"))
          self.bot.reply_to(message, f'Товар "{product_name}" добавлен! Если
хотите добавить или изменить информацию о нем, то воспользуйтесь кнопками:',
reply_markup=keyboard)
      else:
          self.bot.reply_to(message, 'Пожалуйста, укажите название товара и
его цену. Пример: /add Товар1 100')
```

```
    def check_text(self, message):
        print(message.from_user.id, message.text)
    def add_user_product(self, message):
        product_name = "Нет имя"
        product_price = 0
        product_comment = 'Нет описания'
        product_contact = "Нет контактных данных"
        product_photo_id = "None"
        message_date = datetime.fromtimestamp(message.date)
        product = {
             'user': message.from_user.id,
             'name': product_name,
             'comment': product_comment,
             'price': product_price,
             'date': str(message_date.strftime('%Y-%m-%d %H:%M:%S')),
             'contact_user': product_contact,
             'photo_id': product_photo_id

        }
        id = self.products.add_new_product(product)
        self.user.set_id(message.from_user.id, id)
        self.bot.reply_to(message, 'Введите название товара: ')
        self.bot.set_state(message.from_user.id, "ADD-NAME")

    def list_products(self, message):
        page_index = 0
        promt, markup =
self.create_message.mess_generate_page('nextpage',page_index)
        self.user.set_prev_state(message.from_user.id, f"all-
products_{page_index}")
        self.bot.reply_to(message, text = promt, reply_markup = markup)

    def list_users_ads(self, message):
        promt, markup =
self.create_message.mess_list_users_ads(message.from_user.id)
        self.bot.reply_to(message, text=promt, reply_markup=markup)
        self.user.set_prev_state(message.from_user.id, "user-products_0")
```

Create_message_data.py

```
from telebot import types

from datetime import datetime
from data import sql_data

class CreateMessage():
    def __init__(self, products, user):
        self.products = products
        self.user = user

    def mess_list_users_ads(self, user):
        user_id = user
        product_list = self.products.get_products_from_user(user_id, 0, 30)
        if product_list:
            promt = []
            product_keys = types.InlineKeyboardMarkup()
            for i in range(len(product_list)):
                product = product_list[i]
                text_message = f"{i+1}) {product['name']} за
{product['price']} руб."
```

```
product_keys.add(types.InlineKeyboardButton(text=text_message,
callback_data=f"edit-product_{product['id']}"))
            text =  "Ваши объявления:"
            reply_markup = product_keys
        else:
            text =  "Нет доступных товаров."
            reply_markup = None
        return text, reply_markup

    def mess_generate_page(self, command, page_index, count=6):
        product_list = self.products.get_products_from_user('%', page_index,
page_index+count)
        navigation = types.InlineKeyboardMarkup()
        product_buttons = types.InlineKeyboardMarkup()

        if product_list:
            #promt = []
            for i in range(len(product_list)):
                product = product_list[i]
                text_message = f"{i+1+page_index}) {product['name']} за
{product['price']} руб."

product_buttons.add(types.InlineKeyboardButton(text=text_message,
callback_data=f"info-product_{product['id']}"))
                #promt.append(f"{i+1+page_index}) {product['name']} Цена:
{product['price']} руб. \n {product['comment']}\n Контакты:
{product['contact_user']} \n Размещено: {product['date']}\n")
            text = "Список товаров:" #+ '\n'.join(promt)
        else:
            text = "Нет доступных товаров."

        if command == "nextpage":
            if product_list==0 or len(product_list) < count:
                product_buttons.add(types.InlineKeyboardButton(text=f'<<',
callback_data=f"prevpage_{page_index-count}"))
            elif page_index ==0:
                product_buttons.add(types.InlineKeyboardButton(text=f'>>',
callback_data=f"nextpage_{page_index+count}"))
            else:
                product_buttons.add(types.InlineKeyboardButton(text=f'<<',
callback_data=f"prevpage_{page_index-count}"),
                types.InlineKeyboardButton(text=f'>>',
callback_data=f"nextpage_{page_index+count}"))
        elif command == "prevpage":
            if page_index < count:
                product_buttons.add(types.InlineKeyboardButton(text=f'>>',
callback_data=f"nextpage_{page_index+count}"))
            else:
                product_buttons.add(types.InlineKeyboardButton(text=f'<<',
callback_data=f"prevpage_{page_index-count}"),
                types.InlineKeyboardButton(text=f'>>',
callback_data=f"nextpage_{page_index+count}"))

        reply_markup = product_buttons

        return text, reply_markup
```

Callback_handler.py

```
from telebot import types
from bot import create_message_data

class CallbackHandlers():
    def __init__(self, bot, products, user):
        self.bot = bot
        self.create_message = create_message_data.CreateMessage(products,
user)
        self.products = products
        self.user = user

    def callback_set_details(self, call):
        #state = bot.get_state(message.from_user.id)
        command, id = tuple(call.data.split('_'))
        if command == 'setname':
            self.bot.send_message(call.message.chat.id, 'Введите новое имя
продукта:')
            self.bot.set_state(call.from_user.id, "NAME")
        elif command == 'setprice':
             self.bot.send_message(call.message.chat.id, 'Введите новую цену
продукта:')
             self.bot.set_state(call.from_user.id, "PRICE")

        elif command == 'setcomment':
            self.bot.send_message(call.message.chat.id, 'Введите новое
описание продукта:')
            self.bot.set_state(call.from_user.id, "COMMENT")
            #print(self.bot.get_state(call.message.from_user.id))

        elif command == 'setcontact':
            self.bot.send_message(call.message.chat.id, 'Введите новое
контакты для связи с вами:')
            self.bot.set_state(call.from_user.id, "CONTACT")

        self.user.set_id(call.from_user.id, id)

    def callback_page_creator(self, call):
        command, call_page_index = tuple(call.data.split('_'))
        page_index = int(call_page_index)
        promt, markup = self.create_message.mess_generate_page(command,
page_index)
        self.user.set_data(call.from_user.id, 'prev_state', f"all-
products_{page_index}")
        self.bot.edit_message_text(text=promt, reply_markup = markup,
chat_id=call.message.chat.id, message_id=call.message.message_id)

    def callback_get_info_product(self, call):
        command, id = tuple(call.data.split('_'))
        if command == 'edit-product':
            product = self.products.get_product_from_id(id)

            edit_buttons = types.InlineKeyboardMarkup()
            edit_buttons.add(types.InlineKeyboardButton(text="Удалить",
callback_data=f"delete_{product['id']}"),
                types.InlineKeyboardButton(text='Редактировать',
callback_data=f"edit_{product['id']}"),
                types.InlineKeyboardButton(text='Назад',
callback_data="back"))

            text = f"{product['name']} Цена: {product['price']} руб. \n
{product['comment']}\n Контакты: {product['contact_user']} \n Размещено:
{product['date']}\n"
            self.bot.edit_message_text(text, reply_markup = edit_buttons,
chat_id=call.message.chat.id, message_id=call.message.message_id)
```

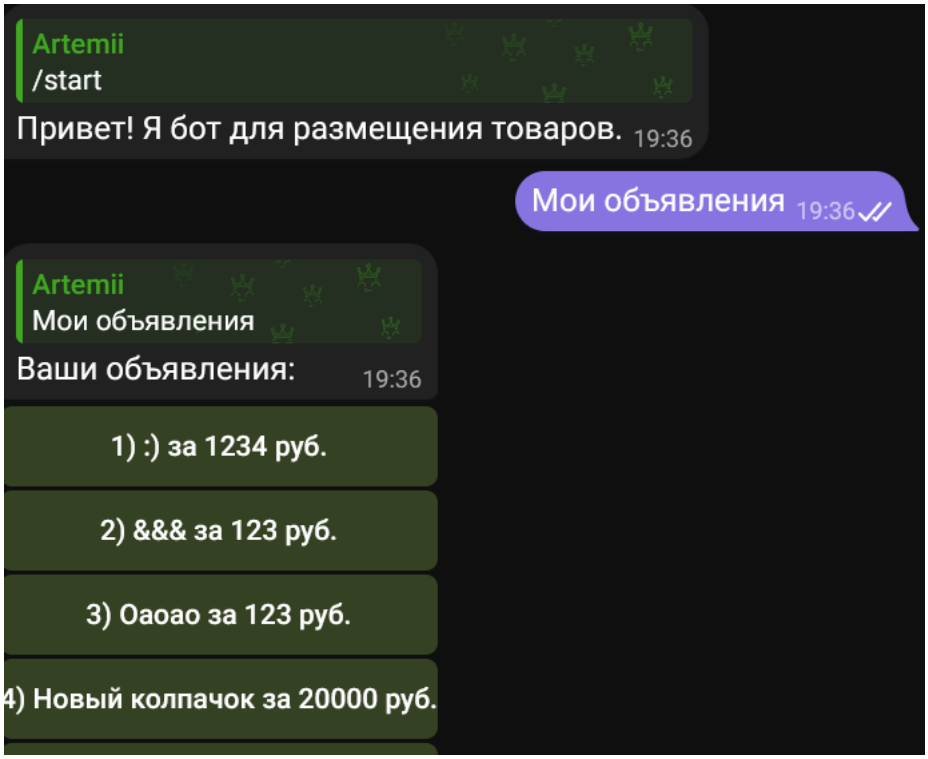
```
        elif command =='info-product':
            product = self.products.get_product_from_id(id)

            edit_buttons = types.InlineKeyboardMarkup()
            edit_buttons.add(types.InlineKeyboardButton(text='Назад',
callback_data="back"))
            text = f"{product['name']} Цена: {product['price']} руб. \n
{product['comment']}\n Контакты: {product['contact_user']} \n Размещено:
{product['date']}\n"
            self.bot.edit_message_text(text, reply_markup = edit_buttons,
chat_id=call.message.chat.id, message_id=call.message.message_id)

    def callback_meny_keyboards(self, call):
        current_state_data = self.user.get_prev_state(call.from_user.id)
        current_state, page_index_str = tuple(current_state_data.split('_'))
        page_index = int(page_index_str)
        if current_state == "user-products":
            promt, markup =
self.create_message.mess_list_users_ads(call.from_user.id)
            self.bot.edit_message_text(promt, reply_markup = markup,
chat_id=call.message.chat.id, message_id=call.message.message_id)
        elif current_state == "all-products":
            promt, markup =
self.create_message.mess_generate_page('nextpage', page_index)

            self.user.set_data(call.from_user.id, 'prev_state', f"all-
products_{page_index}")
            self.bot.edit_message_text(text=promt, reply_markup = markup,
chat_id=call.message.chat.id, message_id=call.message.message_id)

    def callback_user_ads(self, call):
        command, id = tuple(call.data.split('_'))
        if command == "edit":
            keyboard = types.InlineKeyboardMarkup()
            keyboard.add(types.InlineKeyboardButton(text="Цена",
callback_data=f"setprice_{id}"),
            types.InlineKeyboardButton(text=f'Название',
callback_data=f"setname_{id}"),
            types.InlineKeyboardButton(text=f'Описание',
callback_data=f"setcomment_{id}"),
            types.InlineKeyboardButton(text=f'Контакты',
callback_data=f"setcontact_{id}"))

            self.bot.send_message(call.message.chat.id, f'Что хотите
изменить?', reply_markup=keyboard)
        elif command == "delete":
            self.products.delete_product(id)
            self.callback_meny_keyboards(call)
```

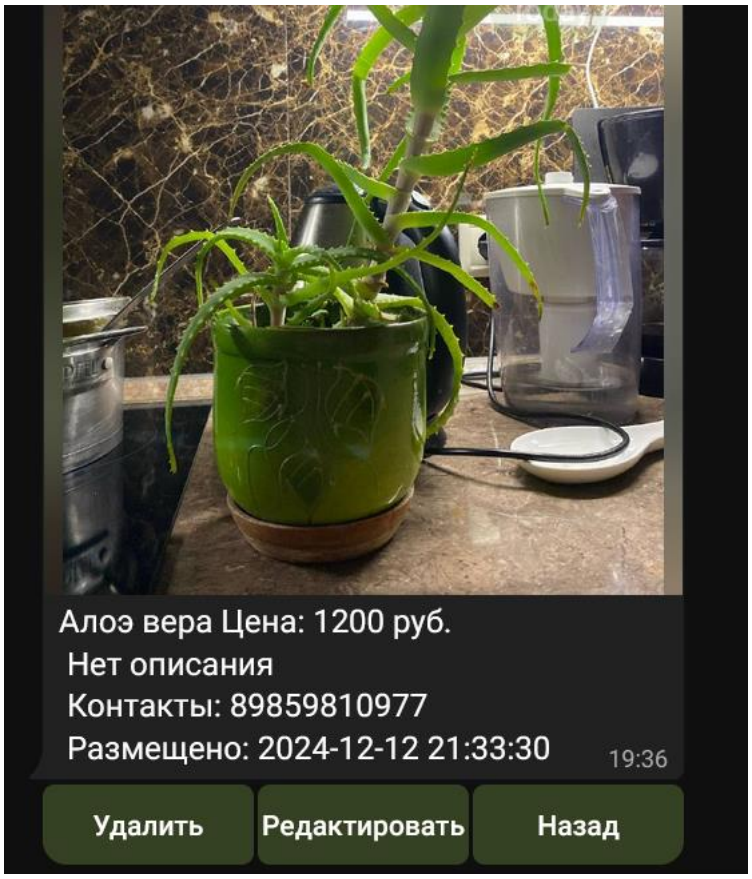
sql_data.py

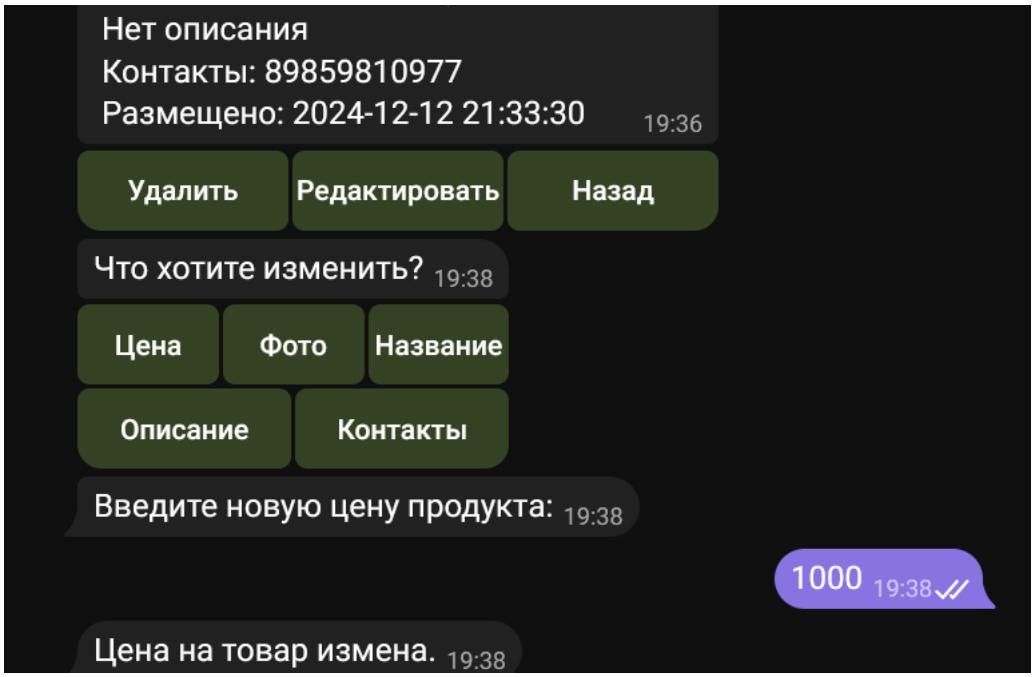
```
import sqlite3
class sql_manager:
    def __init__(self, path):
        self.path = path
        self.name = path.split("\\")[-1].replace(".db", "", 1)
        self.connection = sqlite3.connect(f'{self.path}',
check_same_thread=False)
        self.cursor = self.connection.cursor()
        self.cursor.execute(f""" CREATE TABLE IF NOT EXISTS {self.name}(
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            user INTEGER,
```

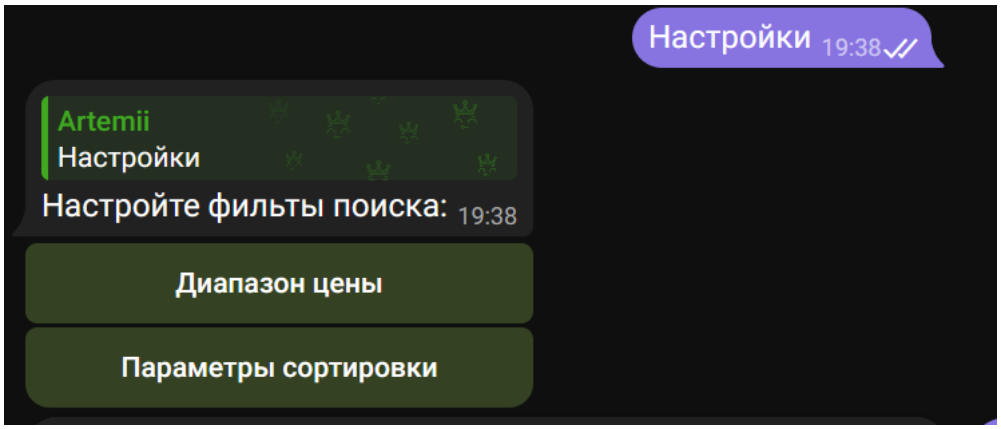
```
            name TEXT,
            comment TEXT,
            price INTEGER,
            date TEXT,
            contact_user TEXT,
            phot_id TEXT
        ) """)
        self.connection.commit()
    def add_new_product(self, product):
        self.cursor.execute(f'INSERT INTO {self.name} (user, name, comment,
price, date, contact_user, photo_id) VALUES (?, ?, ?, ?, ?, ?, ?)',
tuple(map(lambda x: x[1], product.items())))
        id = self.cursor.lastrowid
        self.connection.commit()
        return id

    def set_product(self, name_val, val, id):
        self.cursor.execute(f"UPDATE {self.name} SET {name_val} = ? WHERE id
LIKE ?", (val, id, ))
        self.connection.commit()

    def get_product_from_id(self, id):
        self.cursor.execute(f'SELECT id, name, comment, price, date,
contact_user FROM {self.name} WHERE id LIKE ?', (id, ))
        item = self.cursor.fetchone()
        if item:
            product = {
                'id':item[0],
                'name': item[1],
                'comment': item[2],
                'price': item[3],
                'date': item[4],
                'contact_user': item[5],
                'photo_id':item[6]
            }
            return product
        return 0

    def get_products_from_user(self, user, start, end):
        count = end-start
        self.cursor.execute(f'SELECT id, name, comment, price, date,
contact_user FROM {self.name} WHERE user LIKE ? LIMIT ? OFFSET ?', (user,
count, start))
        products = self.cursor.fetchmany(count)
        if products:
            product_list = []
            for item in products:
                product = {
                    'id':item[0],
                    'name': item[1],
                    'comment': item[2],
                    'price': item[3],
                    'date': item[4],
                    'contact_user': item[5],
                    'photo_id': item[6]
                }
                product_list.append(product)
            return product_list
        return 0
    def delete_product(self, id):
        self.cursor.execute(f"DELETE FROM {self.name} WHERE id LIKE ?",
(id,))
        self.connection.commit()
```

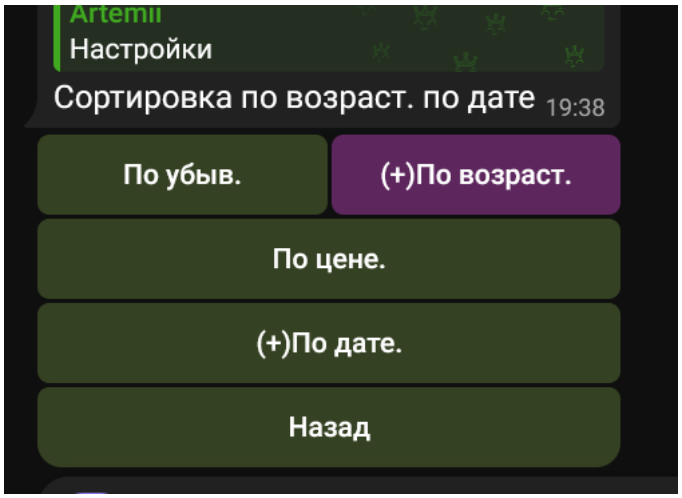
Пример работы:

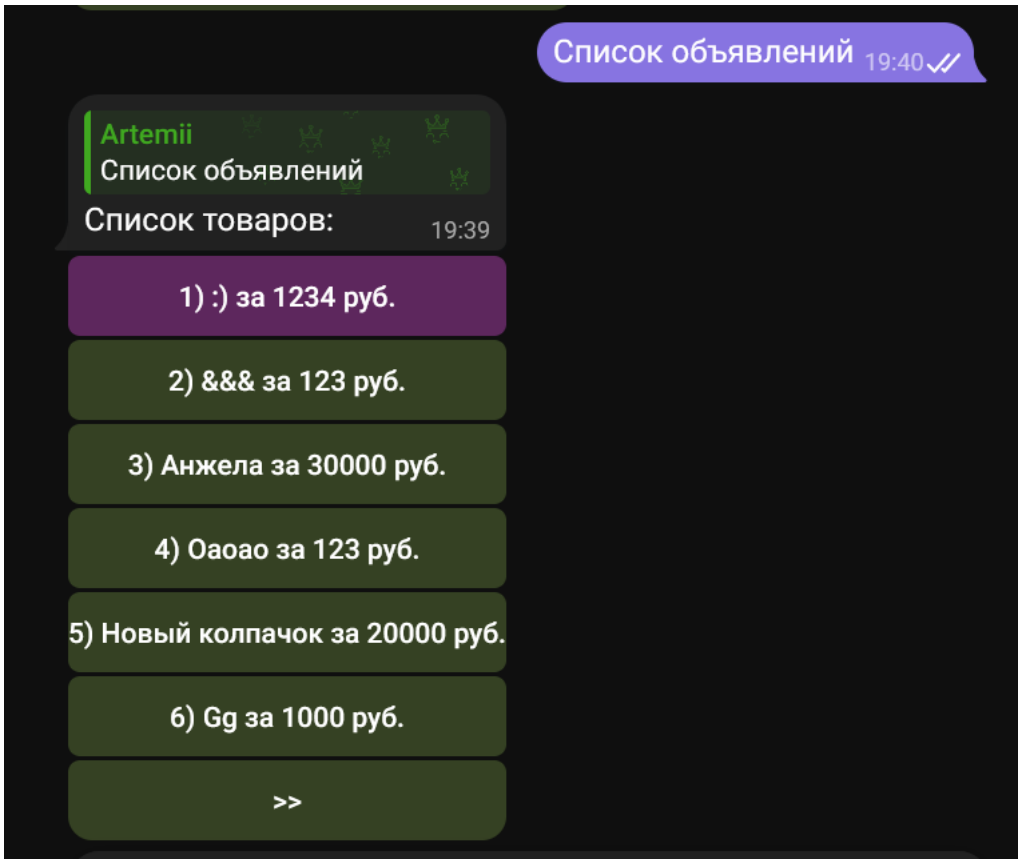





Нет описания
Контакты: 89859810977
Размещено: 2024-12-12 21:33:30 19:36
Удалить
Редактировать
Назад
Что хотите изменить? 19:38
Цена
Фото
Название
Описание
Контакты
Введите новую цену продукта: 19:38
1000 19:38
Цена на товар измена. 19:38


Настройки 19:38
Artemii
Настройки
Настройте фильты поиска: 19:38
Диапазон цены
Параметры сортировки


Artemii
Настройки
Сортировка по возраст. по дате 19:38
По убыв.
(+)По возраст.
По цене.
(+)По дате.
Назад


Список объявлений 19:40
Artemii
Список объявлений
Список товаров: 19:39
1) :) за 1234 руб.
2) &&& за 123 руб.
3) Анжела за 30000 руб.
4) Оаоао за 123 руб.
5) Новый колпачок за 20000 руб.
6) Gg за 1000 руб.
>>