

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ  
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)  
МАИ

---

Институт № 8 «Информационные технологии и прикладная математика»  
Кафедра 806 «Вычислительная математика и программирование»

КУРСОВОЙ ПРОЕКТ

по дисциплине  
БАЗЫ ДАННЫХ

Работу выполнил  
студент группы М8О-307Б-21  
Пирогов М. Д.  
Казанцев Д. И.  
Кажекин Д. А.

Работу приняла  
кандидат технических наук,  
доцент Киндинова В. В.

МОСКВА, 2023

## **1. Техническое задание**

Требуется разработать телеграм-бота онлайн-магазина с 3 интерфейсами. Через пользовательский интерфейс можно выбрать товар из списка, его количество, положить в корзину, оформить заказ и пополнить баланс. О каждом товаре имеется следующая информация: название, фото, страна производителя и цена. Вся эта информация записывается в момент добавления товара через интерфейс владельца. Через интерфейс курьера можно посмотреть актуальные заказы на доставку и принять заказ на доставку. Через интерфейс владельца можно добавить товар и посмотреть информацию по 3 метрикам: общая выручка, количество всех заказов, самый активный пользователь.

Каждому пользователю и курьеру присваивается уникальный ID, который добавляется в базу данных. Пользователь добавляет товары в корзину. Затем создается пользователем заказ и добавляется в базу данных, где потом будет отображаться в интерфейсе курьера как актуальный заказ. Затем любой курьер можно принять этот заказ в своем интерфейсе и доставит.

Функционал должен быть следующим:

- Интерфейс пользователя: содержит кнопки для отображения общей информации об аккаунте пользователя, добавления товаров в корзину, очистки корзины, пополнения баланса, истории покупок, совершения заказа;
- Интерфейс курьера: содержит кнопки для просмотра и принятия заказов;
- Интерфейс владельца: содержит кнопки, которые могут добавить новый товар в базу данных, отображения базовых метрик (выручка, число заказов, самый активный пользователь);

Для обеспечения связи между интерфейсами нужна база данных, состоящая из 7 таблиц:

1. products

в ней хранится информация о каждом товаре (номер продукта, название, цена, страна производителя);

2. orders

в ней хранится информация о каждом заказе (номер заказа и время создания);

3. order\_body

связывает таблица products и orders (номер заказа, номер продукта и его количество);

4. users

в ней хранится информация о каждом пользователе;

5. user\_actions

связывает таблицу users и orders (номер пользователя, номер заказа, действие и время);

6. couriers

в ней хранится информация о каждом курьере;

7. courier\_actions

связывает таблицу couriers и orders (номер курьера, номер заказа, действие и время).

## 2. Проект системы

В разработанной базе данных присутствуют таблицы: order\_body, orders, products, user\_actions, courier\_actions, users, couriers. Эти таблицы связаны между собой логической схемой, представленной на рис. 1.

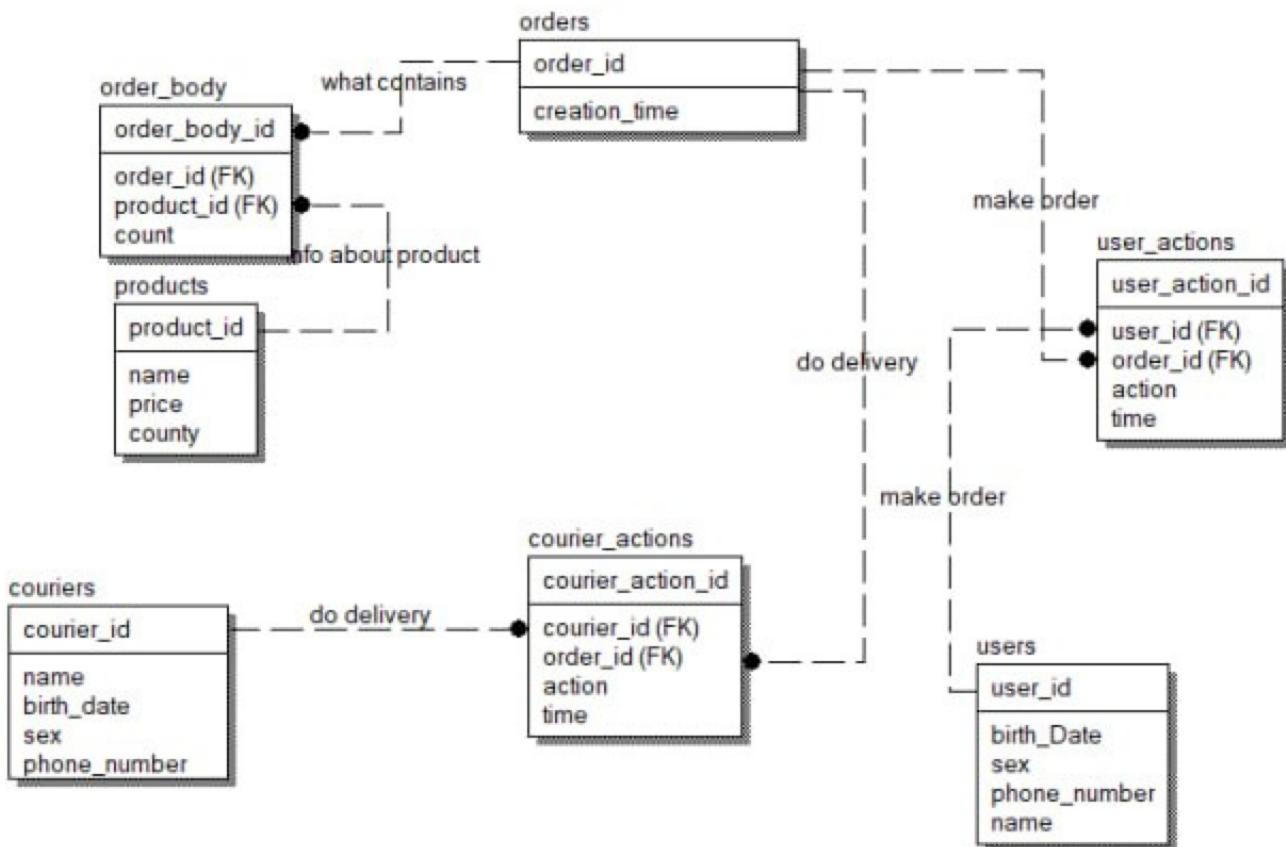


Рис. 1. Логическая схема базы данных, выполненная в ERwin

Поля таблиц и их характеристики представлены в таблице 2.

Таблица 1. Поля таблиц базы данных и их характеристики

Поле	Тип данных	Длина	Дополнительно
<b>«order body»</b>			
<code>order body_id</code>	Serial		Первичный ключ
<code>order id</code>	Integer		Номер заказа
<code>product id</code>	Integer		ID продукта
<code>count</code>	Integer		Количество данного продукта
<b>«products»</b>			
<code>product id</code>	Serial		Первичный ключ
<code>name</code>	Character	50	Название
<code>price</code>	Integer		Цена
<code>country</code>	Character	50	Страна производителя
<b>«orders»</b>			

order_id	Serial	4	Первичный ключ
creation_time	Timestamp		Ключ связи с таблицей «Книга»
<b>«users»</b>			
User_id	Serial		Первичный ключ
name	Character	20	Имя пользователя
Birth_date	Timestamp		Дата рождения
sex	Character	10	Пол
Phone_number	Character	20	Номер телефона
<b>«couriers»</b>			
courier_id	Serial		Первичный ключ
name	Character	20	Имя курьера
Birth_date	Timestamp		Дата рождения
sex	Character	10	Пол
Phone_number	Character	20	Номер телефона
<b>«user_actions»</b>			
User_action_id	Serial		Первичный ключ
User_id	Integer		Номер пользователя
Order_id	Integer		Номер заказа
action	Character	10	Действие
time	Timestamp		Время
<b>«courier_actions»</b>			
courier_action_id	Serial		Первичный ключ
courier_id	Integer		Номер курьера
Order_id	Integer		Номер заказа
action	Character	10	Действие
time	Timestamp		Время

Физическая схема с указанием типов данных приведена на рис. 2.

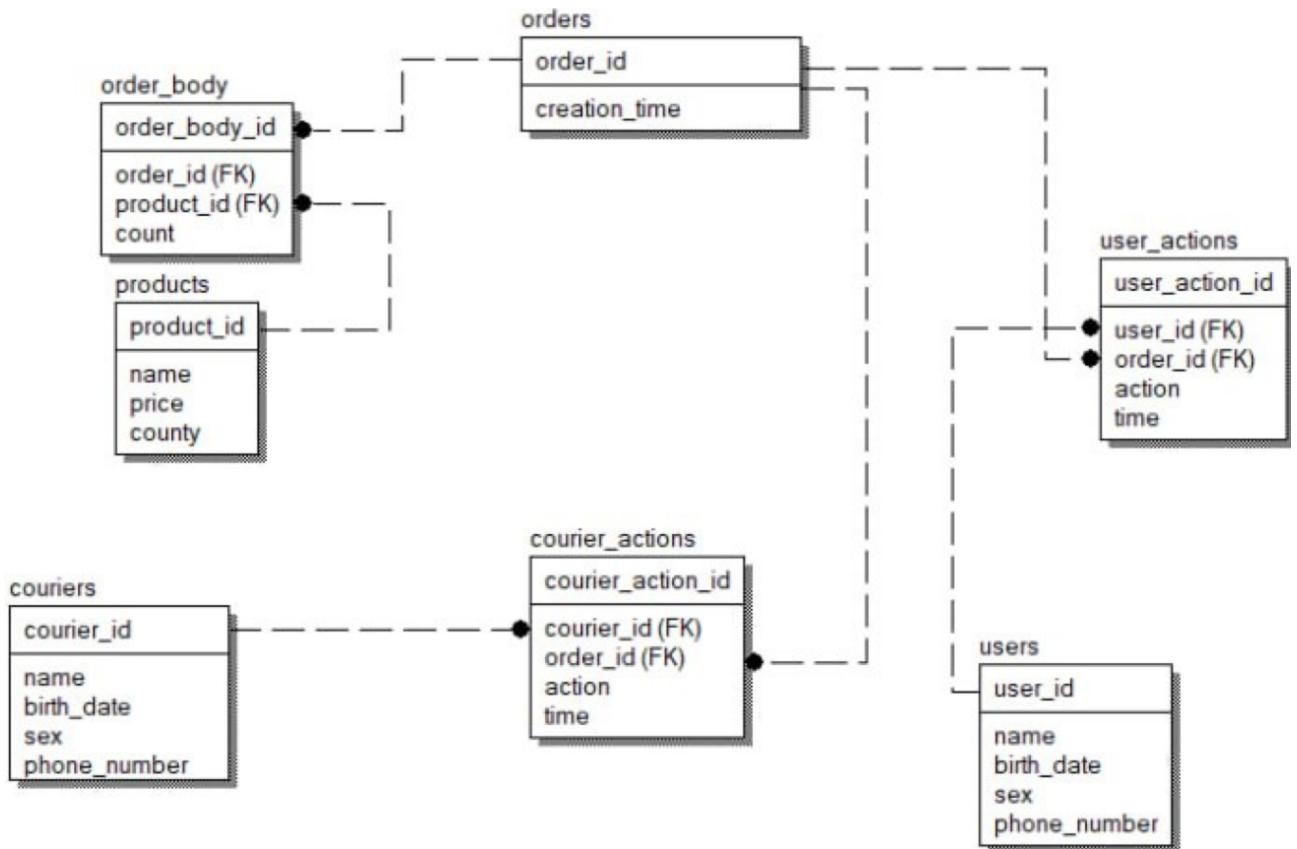


Рис. 2. Физическая схема базы данных, выполненная в ERwin

Описание сгенерированной в ERwin базы данных на SQL:

```

CREATE TABLE courier_actions
(
    courier_id      INTEGER NOT NULL,
    order_id        INTEGER NOT NULL,
    action          VARCHAR(20) NULL,
    courier_action_id  INTEGER NOT NULL,
    time            TIMESTAMP NULL
);
    
```

```

ALTER TABLE courier_actions
ADD PRIMARY KEY (courier_action_id);
    
```

```

CREATE TABLE couriers
(
    courier_id      INTEGER NOT NULL,
    name            VARCHAR(20) NULL,
    birth_date      DATE NULL,
    sex             VARCHAR(20) NULL,
    phone_number    VARCHAR(20) NULL
);
    
```

```

ALTER TABLE couriers
ADD PRIMARY KEY (courier_id);
    
```

```

CREATE TABLE order_body
    
```

```
(  
    order_id      INTEGER NOT NULL,  
    product_id    INTEGER NOT NULL,  
    count         INTEGER NULL,  
    order_body_id INTEGER NOT NULL  
)
```

```
ALTER TABLE order_body  
ADD PRIMARY KEY (order_body_id);
```

```
CREATE TABLE orders  
(  
    order_id      INTEGER NOT NULL,  
    creation_time TIMESTAMP NULL  
)
```

```
ALTER TABLE orders  
ADD PRIMARY KEY (order_id);
```

```
CREATE TABLE products  
(  
    product_id    INTEGER NOT NULL,  
    name          VARCHAR(50) NULL,  
    price         INTEGER NULL,  
    county        VARCHAR(50) NULL  
)
```

```
ALTER TABLE products  
ADD PRIMARY KEY (product_id);
```

```
CREATE TABLE user_actions  
(  
    user_id       INTEGER NOT NULL,  
    order_id      INTEGER NOT NULL,  
    action        VARCHAR(20) NULL,  
    time          TIMESTAMP NULL,  
    user_action_id INTEGER NOT NULL  
)
```

```
ALTER TABLE user_actions  
ADD PRIMARY KEY (user_action_id);
```

```
CREATE TABLE users  
(  
    user_id       INTEGER NOT NULL,  
    birth_date    DATE NULL,  
    sex           VARCHAR(20) NULL,  
    phone_number  VARCHAR(20) NULL,  
    name          VARCHAR(20) NULL  
)
```

```
ALTER TABLE users
```

ADD PRIMARY KEY (user\_id);

ALTER TABLE courier\_actions

ADD FOREIGN KEY R\_4 (order\_id) REFERENCES orders (order\_id);

ALTER TABLE courier\_actions

ADD FOREIGN KEY R\_7 (courier\_id) REFERENCES couriers (courier\_id);

ALTER TABLE order\_body

ADD FOREIGN KEY R\_5 (order\_id) REFERENCES orders (order\_id);

ALTER TABLE order\_body

ADD FOREIGN KEY R\_6 (product\_id) REFERENCES products (product\_id);

ALTER TABLE user\_actions

ADD FOREIGN KEY R\_2 (user\_id) REFERENCES users (user\_id);

ALTER TABLE user\_actions

ADD FOREIGN KEY R\_3 (order\_id) REFERENCES orders (order\_id);

### 3. Реализация

На рисунке 11 продемонстрирована построенная схема базы данных.

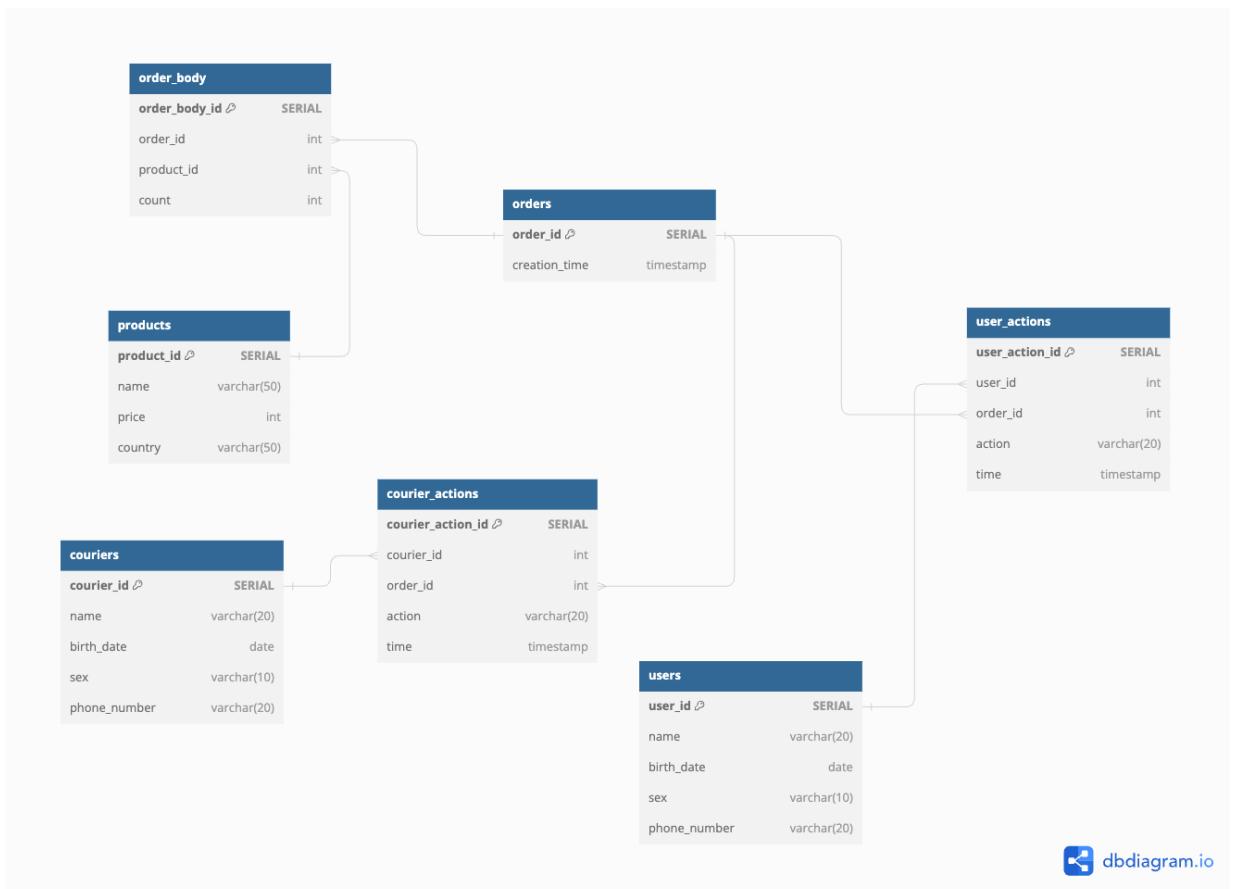


Рис. 11. Схема базы данных

На рисунках 12 – 16 приведены примеры заполнения таблиц базы данных.

	product_id [PK] integer	name character varying (50)	price real	country character varying (50)	url character varying (1024)
1	1	Air Force 1	149.99	China	<a href="https://cdn1.ozone.ru/s3/multimedia/e/6486597578.jpg">https://cdn1.ozone.ru/s3/multimedia/e/6486597578.jpg</a>
2	2	Air Jordan 1 Mid	259.99	Thailand	<a href="https://www.newjordans2022.com/wp-content/uploads/2019/03/Air-Jordan-1-Mid-Chicago-Red-Black-For-Sale-6.jpg">https://www.newjordans2022.com/wp-content/uploads/2019/03/Air-Jordan-1-Mid-Chicago-Red-Black-For-Sale-6.jpg</a>
3	9	Supreme White T	200	Netherlands	<a href="https://cdn1.lyst.com/photos/stadiumgoods/c317143d/supreme-White-Paris-Box-Logo-Tee.jpeg">https://cdn1.lyst.com/photos/stadiumgoods/c317143d/supreme-White-Paris-Box-Logo-Tee.jpeg</a>
4	3	Adidas x Bape Superstar	200.99	China	<a href="https://godmeetsfashion.com/wp-content/uploads/2021/04/bape-a-bathing-ape-adidas-superstar-80s-abc-camo-gz8981-release-20210508-20.jpg">https://godmeetsfashion.com/wp-content/uploads/2021/04/bape-a-bathing-ape-adidas-superstar-80s-abc-camo-gz8981-release-20210508-20.jpg</a>
5	4	Bape Zip Hoodie	159.99	China	<a href="https://i.ebayimg.com/0/s/MTAwMfgxNDAw/z/wEcAOSwk9bfwIw/_/s_5.JPG?set_id=8800005007">https://i.ebayimg.com/0/s/MTAwMfgxNDAw/z/wEcAOSwk9bfwIw/_/s_5.JPG?set_id=8800005007</a>
6	10	Dickies Loose Jeans	139.99	Saint-P	<a href="https://www.venuestore.com.au/media/catalog/product/cache/1/image/2000x/9df7eab33525d08d6e5fb8d27136e95/d/dickies_1994_relaxed_straight_fit_c">https://www.venuestore.com.au/media/catalog/product/cache/1/image/2000x/9df7eab33525d08d6e5fb8d27136e95/d/dickies_1994_relaxed_straight_fit_c</a>
7	11	Thrasher White T	100	China	<a href="https://prokedu.ru/content/img/45/futbolka-thrasher-skate-goat-white_2982301.jpg">https://prokedu.ru/content/img/45/futbolka-thrasher-skate-goat-white_2982301.jpg</a>
8	13	Stone Island Balacalva	259.99	Thailand	<a href="https://images.are.na/eyJidWNrZXQjOjhcmVuYV9pbWFrZXMyLCJrZXkiOizNjg4NzgzL29yaWdpdmFsX2VlYzE2NzliNjUyMnQwZGJzTlwZDZlYmU1NTA3NTViL">https://images.are.na/eyJidWNrZXQjOjhcmVuYV9pbWFrZXMyLCJrZXkiOizNjg4NzgzL29yaWdpdmFsX2VlYzE2NzliNjUyMnQwZGJzTlwZDZlYmU1NTA3NTViL</a>

Рис. 12. Заполненная таблица «products»

	user_action_id [PK] integer	user_id integer	order_id integer	action character varying (20)	time timestamp without time zone
1		19	766015334	32	delivered
2		20	529319238	33	delivered
3		21	766015334	34	delivered

Рис. 13. Заполненная таблица «user\_actions»

	order_body_id [PK] integer	order_id integer	product_id integer	counts integer
1		39	32	3
2		40	32	10
3		41	32	13
4		42	33	4
5		43	34	3

Рис. 14. Заполненная таблица «order\_body»

	order_id [PK] integer	creation_time timestamp without time zone
1		2023-11-26 22:36:11.08342
2		2023-11-27 13:19:40.162557
3		2023-11-27 14:44:23.232415

Рис. 15. Заполненная таблица «orders»

	user_id [PK] integer	username character varying (20)	balance integer
1	529319238	gswwnrs	5188977
2	766015334	famdex	999996612
3	534934446	AxAndrei	200000
4	777346449	putilin21dn	0
5	611857481	trapdn1	187

Рис. 16. Заполненная таблица «users»

	<b>id</b> [PK] integer	<b>user_id</b> integer	<b>item_name</b> character varying (50)	<b>item_count</b> integer
1	44	529319238	Bape Zip Hoodie	52

Рис. 17. Заполненная таблица «cart»

	<b>courier_action_id</b> [PK] integer	<b>courier_id</b> integer	<b>order_id</b> integer	<b>action</b> character varying (20)	<b>time</b> timestamp without time zone
1		7	766015334	32	delivered
2		8	611857481	32	delivered
3		9	529319238	33	delivered
4		10	766015334	34	delivered

Рис. 18. Заполненная таблица «courier\_actions»

	<b>courier_id</b> [PK] integer	<b>name</b> character varying (20)	<b>birth_date</b> date	<b>sex</b> character varying (10)	<b>phone_number</b> character varying (20)
1	611857481	Danila	1952-07-17	male	+7952812
2	766015334	Denis	1952-07-17	male	+7952812
3	529319238	Mark	1952-07-17	male	+7952812

Рис. 19. Заполненная таблица «couriers»

Далее приведены описания (рисунок и программный код элементов) всех форм созданной программы.

## Интерфейс клиентского бота

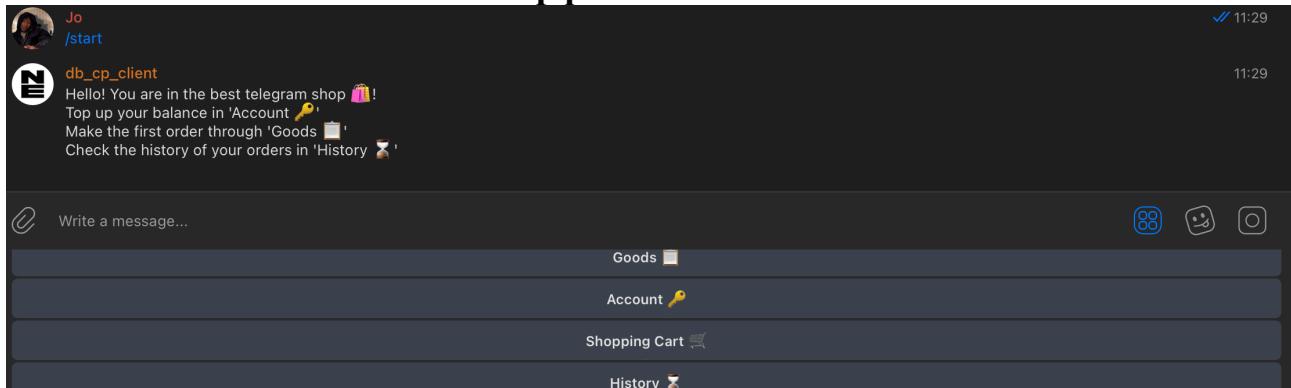


Рис. 20. Клиентский интерфейс. Начальное сообщение

Код события “Start message”:

```
# Start button
@dp.message(Command("start"))
async def cmd_start(message: types.Message):
    kb = [
        [types.KeyboardButton(text="Goods 📦")],
        [types.KeyboardButton(text="Account 💰")],
        [types.KeyboardButton(text="Shopping Cart 🛒")],
        [types.KeyboardButton(text="History ⏳")]
    ]
    keyboard = types.ReplyKeyboardMarkup(keyboard=kb, resize_keyboard=True)
    await message.answer("Hello! You are in the best telegram shop 🛍! \n"
                        "Top up your balance in 'Account 💰' \n"
                        "Make the first order through 'Goods 📦' \n")
```

"Check the history of your orders in 'History ⏳'", reply\_markup=keyboard)

```
await add_new_user(message)
```

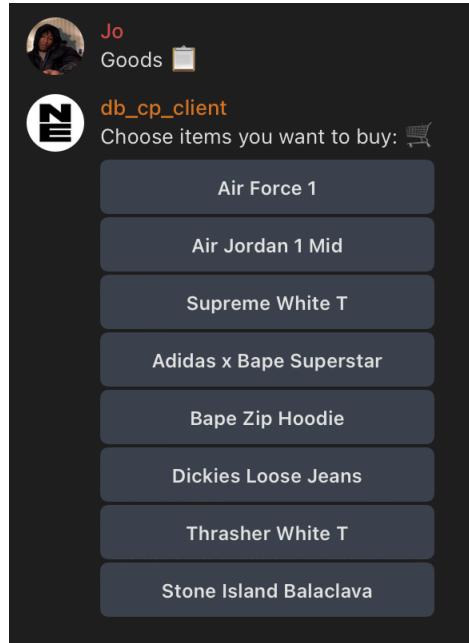


Рис. 21. Клиентский интерфейс. Кнопка “Goods”

Код события “Goods”:

```
@dp.message(F.text == "Goods 📝")
async def menu_button(message: types.Message, state: FSMContext):
    conn = await db_connection()
    data = await conn.fetch(
        "SELECT * FROM products"
    )
    keyboard = [[InlineKeyboardButton(text=record['name'], callback_data=record['name'])] for
               record in data]
    keyboard = types.InlineKeyboardMarkup(inline_keyboard=keyboard)
    await message.answer(text="Choose items you want to buy: 🛒", reply_markup=keyboard)
    await state.set_state(choose_button.button)
    await conn.close()
```



Рис. 22. Клиентский интерфейс. Выбор кнопки товара из списка “Goods”

Код события выбора кнопки товара из списка “Goods”:

```
@dp.message(choose_button.button)
@dp.callback_query((F.data != "Back") & (F.data[:3] != "Add"))
async def query_buttons(callback: types.CallbackQuery, state: FSMContext):
    await callback.message.delete()
    price, country, url = await get_info(callback.data)
    bbutton = InlineKeyboardButton(text="Back ⏪", callback_data="Back")
    add_button = InlineKeyboardButton(text="Add to cart 🖊",
                                    callback_data="Add:{}".format(callback.data))
    keyboard = InlineKeyboardMarkup(inline_keyboard=[[add_button], [bbutton]])
    await callback.message.answer_photo(
        photo=url,
        caption='Item: {} \n'
                'Price: {}$ \n'
                'Country: {} \n'.format(callback.data, round(price, 2), country),
        reply_markup=keyboard
    )
    await state.clear()
```

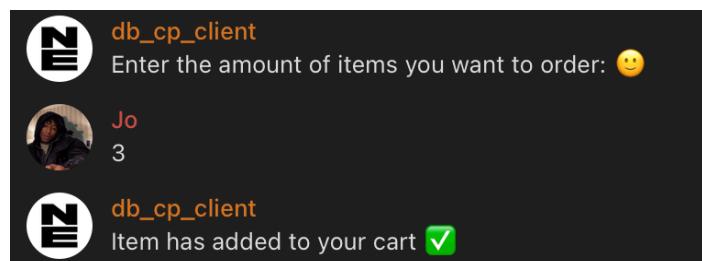


Рис. 23. Клиентский интерфейс. Кнопка “Add to cart”

Код события выбора кнопки “Add to cart”:

```
class ItemAmount(StatesGroup):
    amount = State()
```

```

@dp.callback_query(F.data.split(sep=":")[0] == "Add")
async def add_2_cart_get_item(callback: types.CallbackQuery, state: FSMContext):
    await callback.message.delete()
    await state.set_state(ItemAmount.amount)
    await state.update_data(item=callback.data.split(":")[1])
    await callback.message.answer(text="Enter the amount of items you want to order: 😊")

@dp.message(ItemAmount.amount)
async def add_2_cart_take_quantity(message: types.Message, state: FSMContext):
    amount = message.text
    try:
        amount = int(amount)
        item_name = await state.get_data()
        await add_2_cart(item_name['item'], amount, message.from_user.id)
        await message.answer(text="Item has added to your cart ✅")
    except:
        await message.reply(text="Please provide correct number! ❌")
    finally:
        await state.clear()

```

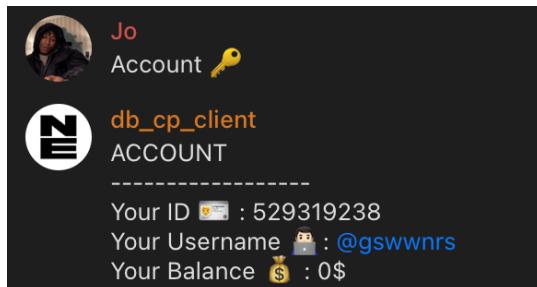


Рис. 24. Клиентский интерфейс. Кнопка “Account”

Код события выбора кнопки “Account”:

```

@dp.message(F.text == "Account 🔑")
async def account_button(message: types.Message):
    top_up_button = types.KeyboardButton(text="Top Up Balance 💰")
    back_button_ = types.KeyboardButton(text="Back ←")
    kb = [
        [top_up_button],
        [back_button_]
    ]
    keyboard = types.ReplyKeyboardMarkup(keyboard=kb, resize_keyboard=True)
    await message.answer(" ACCOUNT \n"
                         "----- \n"
                         "Your ID 📱 : {} \n"
                         "Your Username 🎙 : @{} \n"
                         "Your Balance 💰 : ${} \n".format(message.from_user.id,
                                              message.from_user.username,
                                              (await get_balance(message))[0]['balance']),
                         reply_markup=keyboard)

```

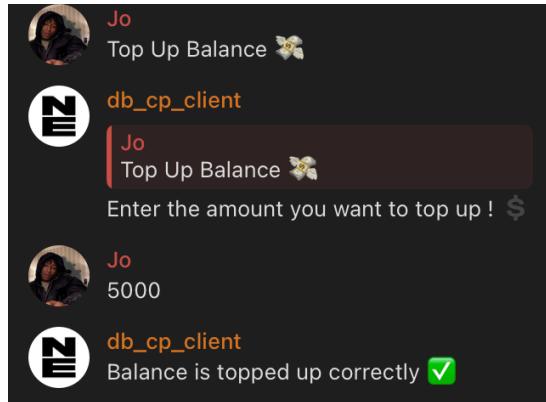


Рис. 25. Клиентский интерфейс. Кнопка “Top up balance”

Код события выбора кнопки “Top up balance”:

```
@dp.message(F.text == "Top Up Balance 💰")
async def top_up_balance_button(message: types.Message, state: FSMContext):
    await state.set_state(TopUpBalance.amount)
    await message.reply(text="Enter the amount you want to top up ! $")
```

```
@dp.message(TopUpBalance.amount)
async def balance_top_upped(message: types.Message, state: FSMContext):
    amount = message.text
    try:
        float(amount)
        await top_up_balance(message, int(amount))
        await message.answer(text="Balance is topped up correctly ✅")
    except ValueError:
        await message.reply(text="Please provide correct number to top up! ❌")
    finally:
        await state.clear()
```

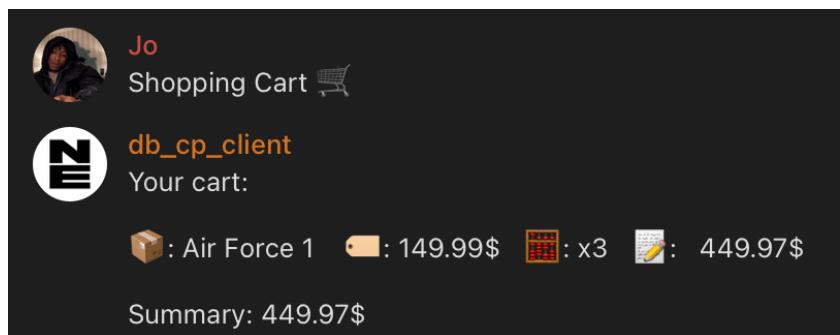


Рис. 26. Клиентский интерфейс. Кнопка “Shopping Cart”

Код события выбора кнопки “Shopping Cart”:

```
@dp.message(F.text == "Shopping Cart 🛒")
async def cart_button(message: types.Message) -> None:
    keyboard = [
        [types.KeyboardButton(text="Back 🔙")],
```

```

[types.KeyboardButton(text="Clean Up Cart 🛒")],
[types.KeyboardButton(text="Make Order ⭐")]
]
kb = types.ReplyKeyboardMarkup(keyboard=keyboard)

conn = await db_connection()
data = await conn.fetch(
    "SELECT * FROM cart WHERE user_id = $1", message.from_user.id
)
output = "Your cart: \n\n"
Summary = 0
for record in data:
    if record['user_id'] == message.from_user.id:
        price_query = await conn.fetch(
            "SELECT price FROM products WHERE name = $1", record['item_name']
        )
        price = price_query[0]['price']
        line = "📦: {} 🎁: {} $ 🎄: x{} 📋: {}$ \n".format(record['item_name'], round(price, 2),
                                                          record['item_count'],
                                                          round(price * record['item_count'], 2))
        output += line
        Summary += round(price * record['item_count'], 2)
output += "\n" "Summary: " + str(Summary) + "$\n"
await message.answer(text=output, reply_markup=kb)
await conn.close()

```

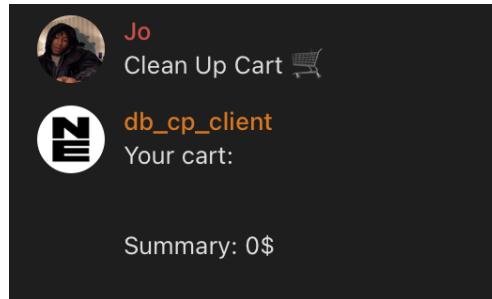


Рис. 27. Клиентский интерфейс. Кнопка “Clean Up Cart”

Код события выбора кнопки “Clean Up Cart”:

```

@dp.message(F.text == "Clean Up Cart 🛒")
async def clean_up_cart_button(message: types.Message) -> None:
    conn = await db_connection()
    await conn.execute(
        "DELETE FROM cart WHERE user_id = $1", message.from_user.id
    )
    await conn.close()
    await cart_button(message)

```

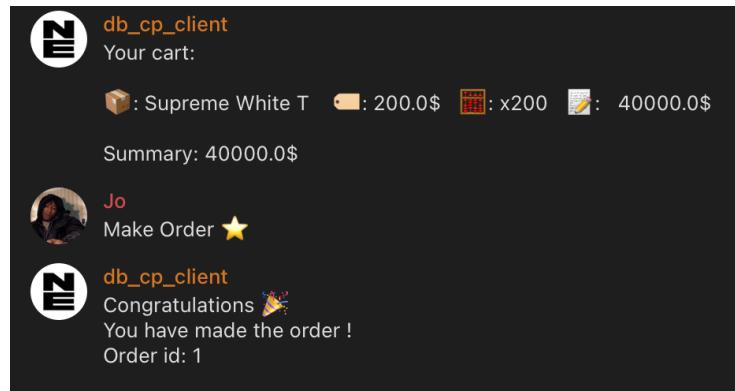


Рис. 28. Клиентский интерфейс. Кнопка “Make order”

Код события выбора кнопки “Make order”:

```
@dp.message(F.text == "Make Order ★")
async def make_order(message: types.Message) -> None:
    conn = await db_connection()
    balance = await conn.fetch(
        "SELECT balance FROM users WHERE user_id = $1", message.from_user.id
    )

    cart = await conn.fetch(
        "SELECT * FROM cart WHERE user_id = $1", message.from_user.id
    )

    summary = 0
    # Считаем общую сумму заказа
    for record in cart:
        if record['user_id'] == message.from_user.id:
            price_query = await conn.fetch(
                "SELECT price FROM products WHERE name = $1", record['item_name']
            )
            price = price_query[0]['price']
            summary += round(price * record['item_count'], 2)

    # Если денег на балансе достаточно
    if balance[0]['balance'] >= summary:
        # Если корзина пуста
        if summary == 0:
            await message.answer(text="Please, add something to your cart to make the order 🙏\n")
            return
        # Заполняем orders
        order_id = await conn.fetch(
            "INSERT INTO orders(creation_time) VALUES (CURRENT_TIMESTAMP)
RETURNING order_id, creation_time"
        )
        # Заполняем order_body
        for record in cart:
            product_id = await conn.fetch(
                "SELECT product_id FROM products WHERE name = $1", record['item_name']
            )
```

```

        await conn.execute(
            "INSERT INTO order_body(order_id, product_id, counts) VALUES($1, $2, $3)",
            int(order_id[0]['order_id']),
            int(product_id[0]['product_id']),
            record['item_count']
        )
        await message.answer(text="Congratulations 🎉\n"
            "You have made the order !\n"
            "Order id: {}".format(order_id[0]['order_id']))

# Делаем запись в таблицу user_actions
await conn.execute(
    "INSERT INTO user_actions(user_id, order_id, action, time) VALUES($1, $2, $3, $4)",
    message.from_user.id,
    int(order_id[0]['order_id']), 'create_order', order_id[0]['creation_time']
)
# Снимаем деньги с баланса
await conn.execute(
    "UPDATE users SET balance = balance - $1 WHERE user_id = $2", summary,
    message.from_user.id
)
else:
    await message.answer(text="Unfortunately, you haven't got enough credits to make this
purchase 😢\n"
    "Top up your balance in 'Account 💰'\n")
await conn.close()

```

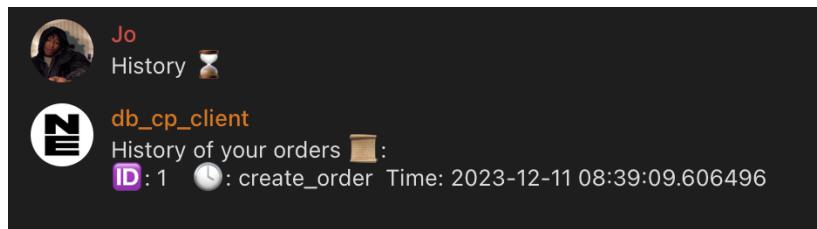


Рис. 29. Клиентский интерфейс. Кнопка “History”

Код события выбора кнопки “History”:

```

@dp.message(F.text == "History 🕒")
async def history_button(message: types.Message) -> None:
    conn = await db_connection()
    data = await conn.fetch(
        "SELECT * FROM user_actions WHERE user_id = $1", message.from_user.id
    )
    output = "History of your orders 📜:\n"
    for record in data:
        time = record['time']
        order_id = record['order_id']
        status = record['action']
        line = "ID: {} ⏱: {} Time: {} \n".format(order_id, status, time)
        output += line
    await message.answer(text=output)

```

## Интерфейс владельца

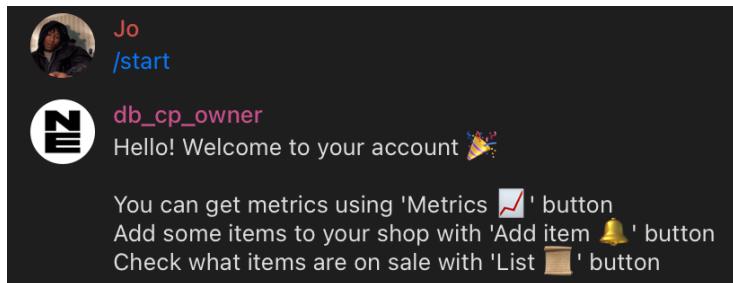


Рис. 30. Интерфейс владельца. Начальное сообщение

Код события начального сообщения:

```
@dp.message(Command("start"))
async def cmd_start(message: types.Message):
    kb = [
        types.KeyboardButton(text="Metrics 📈"),
        types.KeyboardButton(text="Add 🎙"),
        types.KeyboardButton(text="List 📃")
    ]
    keyboard = types.ReplyKeyboardMarkup(keyboard=kb, resize_keyboard=True)
    await message.answer("Hello! Welcome to your account 🎉\n\n"
                         "You can get metrics using 'Metrics 📈' button \n"
                         "Add some items to your shop with 'Add item 🎙' button \n"
                         "Check what items are on sale with 'List 📃' button \n", reply_markup=keyboard)
```

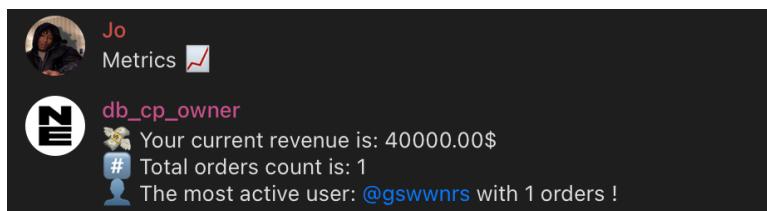


Рис. 31. Интерфейс владельца. Кнопка “Metrics”

Код события выбора кнопки “Metrics”:

```
@dp.message(F.text == "Metrics 📈")
async def metrics_button(message: types.Message):
    conn = await db_connection()
    revenue = (await conn.fetch(
        "SELECT SUM(ROUND((counts * price)::DECIMAL, 2)) AS revenue FROM orders INNER
        JOIN order_body USING(order_id) INNER JOIN products USING(product_id)"
    ))[0]['revenue']

    orders_count = (await conn.fetch(
        "SELECT COUNT(*) FROM orders"
    ))[0]['count']

    best_user = (await conn.fetch(
        "WITH group_ AS (SELECT user_id, COUNT(*) AS count_order FROM user_actions
        GROUP BY user_id) SELECT * FROM group_ INNER JOIN users USING(user_id) WHERE
        count_order = (SELECT MAX(count_order) FROM group_)"
    ))[0]
```

```

)))[0]

text = ("💸 Your current revenue is: {}$ \n"
        "#📝 Total orders count is: {} \n"
        "👤 The most active user: @{} with {} orders !").format(revenue, orders_count,
best_user['username'],
                           best_user['count_order'])

await message.answer(text=text)
await conn.close()

```

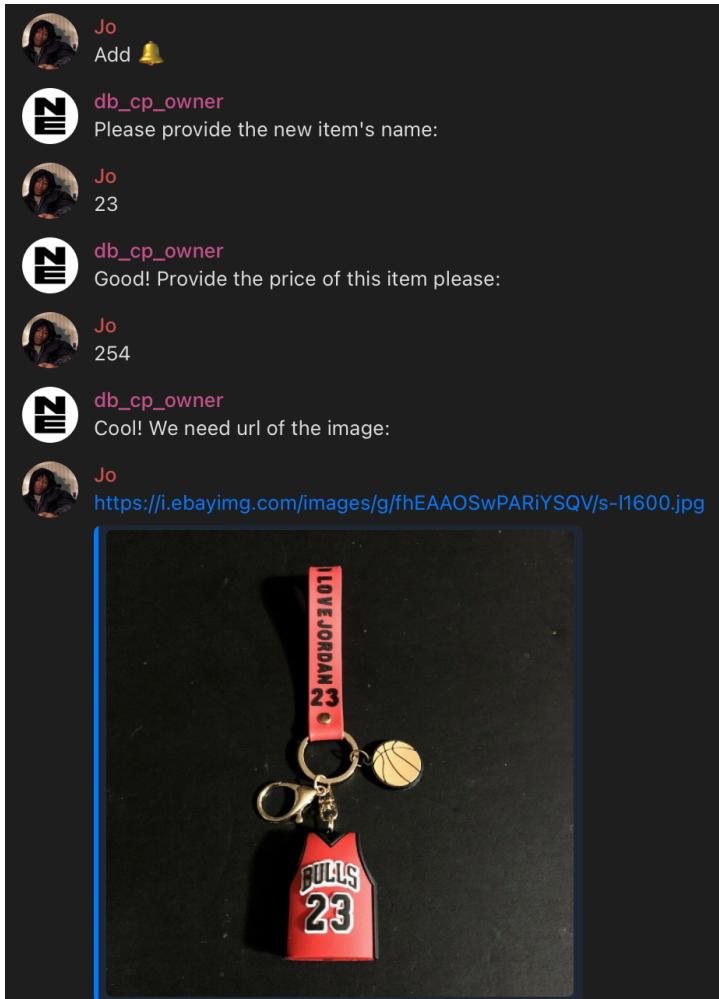


Рис. 32а. Интерфейс владельца. Кнопка “Add”

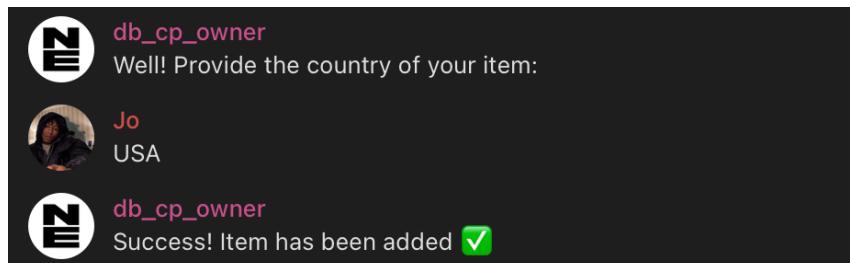


Рис. 32б. Интерфейс владельца. Кнопка “Add”

Код события выбора кнопки “Add”:

```

class add_item(StatesGroup):
    item_name = State()
    price = State()
    url = State()
    country = State()

    @dp.message(F.text == "Add 🎙")
    async def add_item_button(message: types.Message, state: FSMContext):
        await state.set_state(add_item.item_name)
        await message.answer(text="Please provide the new item's name:")

    @dp.message(add_item.item_name)
    async def item_name_input(message: types.Message, state: FSMContext):
        await state.update_data(name=message.text)
        await state.set_state(add_item.price)
        await message.answer("Good! Provide the price of this item please:")

    @dp.message(add_item.price)
    async def item_price_input(message: types.Message, state: FSMContext):
        await state.update_data(price=message.text)
        await state.set_state(add_item.url)
        await message.answer("Cool! We need url of the image:")

    @dp.message(add_item.url)
    async def item_url_input(message: types.Message, state: FSMContext):
        await state.update_data(url=message.text)
        await state.set_state(add_item.country)
        await message.answer("Well! Provide the country of your item:")

    @dp.message(add_item.country)
    async def item_country_input(message: types.Message, state: FSMContext):
        saved_data = await state.get_data()
        conn = await db_connection()
        await conn.execute(
            "INSERT INTO products(name, price, url, country) VALUES($1, $2, $3, $4)",
            saved_data['name'],
            round(float(saved_data['price']), 2), saved_data['url'], message.text
        )
        await message.answer(text="Success! Item has been added ✅")
        await conn.close()

```

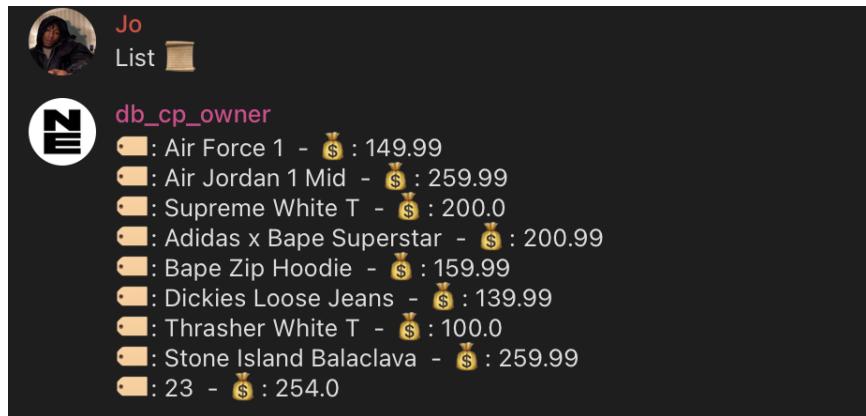


Рис. 33. Интерфейс владельца. Кнопка “List”

Код события выбора кнопки “List”:

```
@dp.message(F.text == "List 📜")
async def list_items_button(message: types.Message):
    text = ""
    conn = await db_connection()
    data = await conn.fetch(
        "SELECT * FROM products"
    )
    for record in data:
        item_name = record['name']
        price = record['price']
        line = "•: {} - ₽: {} \n".format(item_name, round(price, 2))
        text += line

    await message.answer(text=text)
    await conn.close()
```

1	•: Air Force 1 - ₽: 149.99
2	•: Air Jordan 1 Mid - ₽: 259.99
3	•: Supreme White T - ₽: 200.0
4	•: Adidas x Bape Superstar - ₽: 200.99
5	•: Bape Zip Hoodie - ₽: 159.99
6	•: Dickies Loose Jeans - ₽: 139.99
7	•: Thrasher White T - ₽: 100.0
8	•: Stone Island Balaclava - ₽: 259.99
9	•: 23 - ₽: 254.0
10	

Рис. 34. Полученный список актуальных товаров с их ценами по запросу

Код события:

```
async def dump_items_button(message: types.Message):
    text = ""
    conn = await db_connection()
    data = await conn.fetch(
```

```

    "SELECT * FROM products"
)
for record in data:
    item_name = record['name']
    price = record['price']
    line = "◦: {} - $: {} \n".format(item_name, round(price, 2))
    text += line

```

with open("dump.txt", "w") as file:  
 file.write(text)

```

await message.answer(text="Success! Dump has been created ✅")
await conn.close()

```

```

@dp.message(F.text == "Dump 🔍")
async def dump_items_button(message: types.Message):
    text = ""
    conn = await db_connection()
    data = await conn.fetch(
        "SELECT * FROM products"
    )
    for record in data:
        item_name = record['name']
        price = record['price']
        line = "◦: {} - $: {} \n".format(item_name, round(price, 2))
        text += line

```

with open("dump.txt", "w") as file:  
 file.write(text)

```

await message.answer(text="Success! Dump has been created ✅")
await conn.close()

```

## Интерфейс курьера

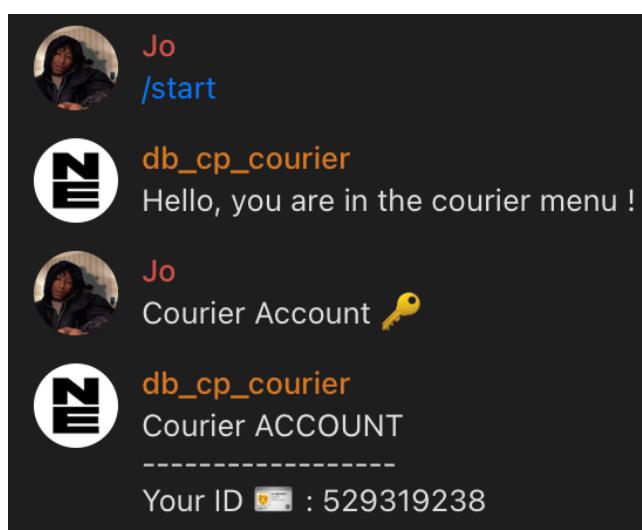


Рис. 35. Интерфейс курьера. Начальное сообщение

Код события начального сообщения:

```
@dp.message(Command("start"))
async def cmd_start(message: types.Message):
```

```
    kb = [[types.KeyboardButton(text="Courier Account 🔑")]]
    keyboard = types.ReplyKeyboardMarkup(keyboard=kb, resize_keyboard=True)
    await message.answer("Hello, you are in the courier menu !", reply_markup=keyboard)
```

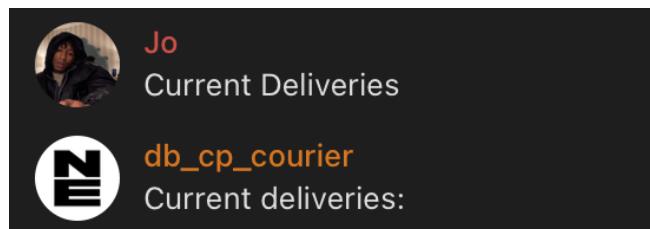


Рис. 36. Интерфейс курьера. Кнопка “Current Deliveries”

Код события выбора кнопки “Current Deliveries”:

```
@dp.message(F.text == "Current Deliveries")
async def current_deliveries_button(message: types.Message):
    data_list = await get_inprocess_action_courier(message)
    data_list = [str(i) for i in data_list]
    keyboard = types.InlineKeyboardMarkup(inline_keyboard=[
        [types.InlineKeyboardButton(text=data, callback_data="button2:" + data)] for data in data_list
    ])
    await message.answer("Current deliveries:\n",
                        reply_markup=keyboard
    )
```

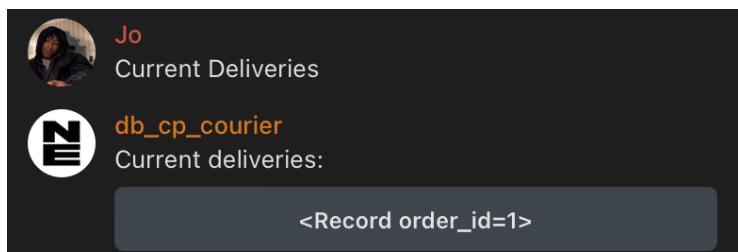


Рис. 37. Интерфейс курьера. Кнопка “Available Deliveries”

Код события выбора кнопки “Available Deliveries”:

```
@dp.message(F.text == "Available Deliveries")
async def available_deliveries_button(message: types.Message):
    data_list = await get_available_deliveries(message)
    data_list = [str(i) for i in data_list]
    keyboard = types.InlineKeyboardMarkup(inline_keyboard=[
        [types.InlineKeyboardButton(text=data, callback_data="button1:" + data)] for data in data_list
    ])
    await message.answer("Available deliveries:\n",
                        reply_markup=keyboard
    )
```

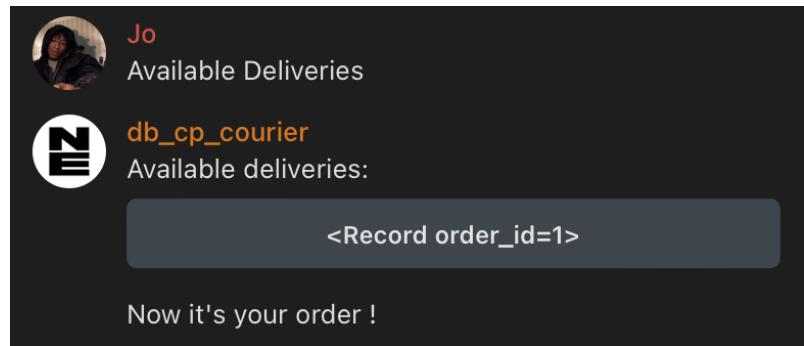


Рис. 38. Интерфейс курьера. Принятие доступного заказа

Код события выбора кнопки заказа:

```
@dp.callback_query(lambda c: c.data.startswith('button1:'))
async def available_deliveries_button_callback(callback_query: types.CallbackQuery):
    conn = await db_connection()
    await conn.execute("UPDATE user_actions SET action = 'in_process', time =
CURRENT_TIMESTAMP WHERE order_id = $1",
                      int(callback_query.data[25:-1]))
    await conn.execute(
        "INSERT INTO courier_actions (courier_id, order_id, action, time) VALUES ($1, $2,
'in_process', CURRENT_TIMESTAMP)",
        callback_query.from_user.id, int(callback_query.data[25:-1]))
    await bot.answer_callback_query(callback_query.id)
    await bot.send_message(callback_query.from_user.id, "Now it's your order !")
```



Рис. 39. Интерфейс курьера. Окончание доставки заказа

Код события выбора кнопки заказа:

```
@dp.callback_query(lambda c: c.data.startswith('button2:'))
async def available_deliveries_button_callback(callback_query: types.CallbackQuery):
    conn = await db_connection()
    await conn.execute("UPDATE user_actions SET action = 'delivered', time =
CURRENT_TIMESTAMP WHERE order_id = $1",
                      int(callback_query.data[25:-1]))
    await conn.execute("UPDATE courier_actions SET action = 'delivered', time =
CURRENT_TIMESTAMP WHERE order_id = $1",
                      int(callback_query.data[25:-1]))
    await bot.answer_callback_query(callback_query.id)
    await bot.send_message(callback_query.from_user.id, "You delivered the order!")
```