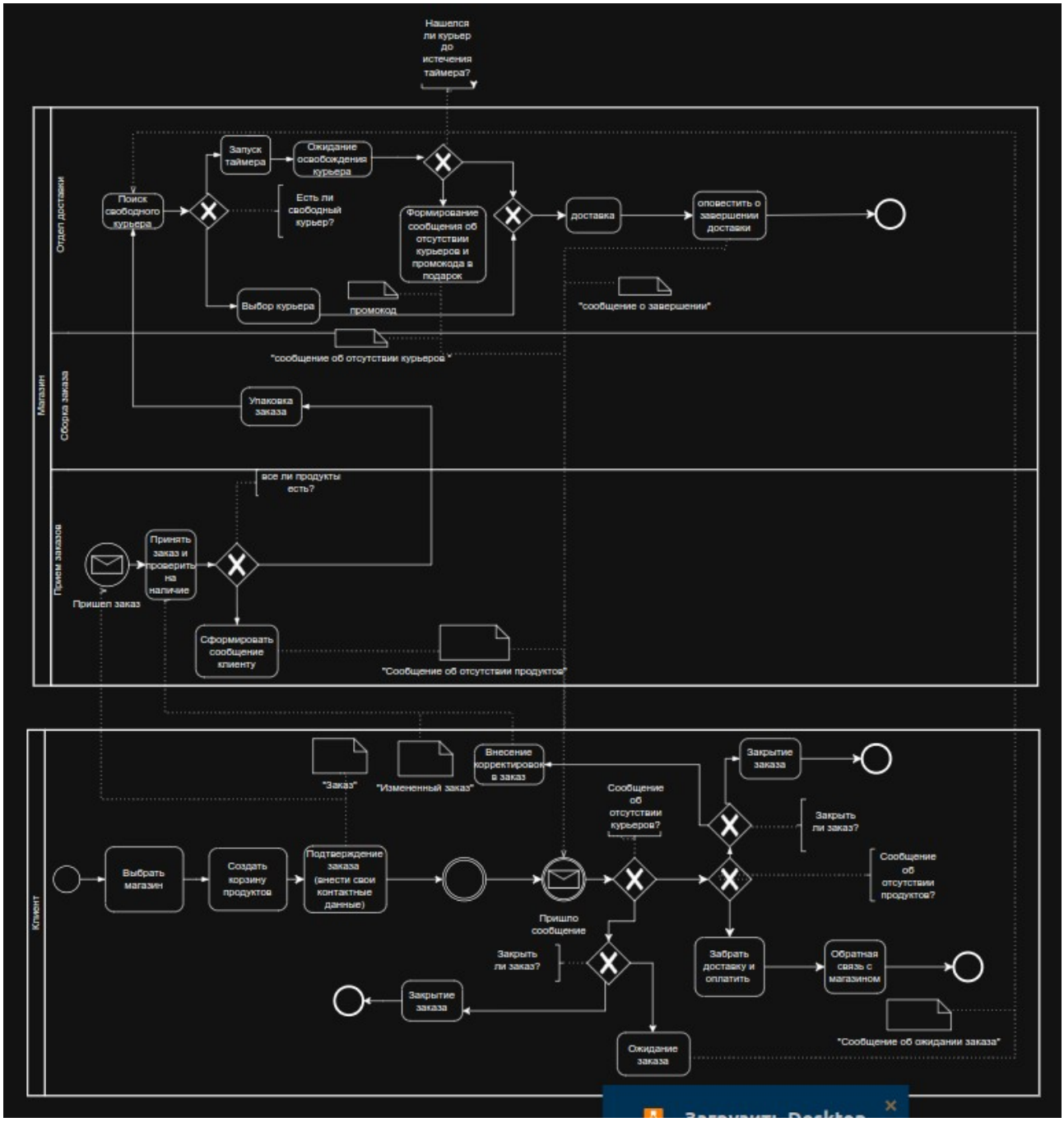
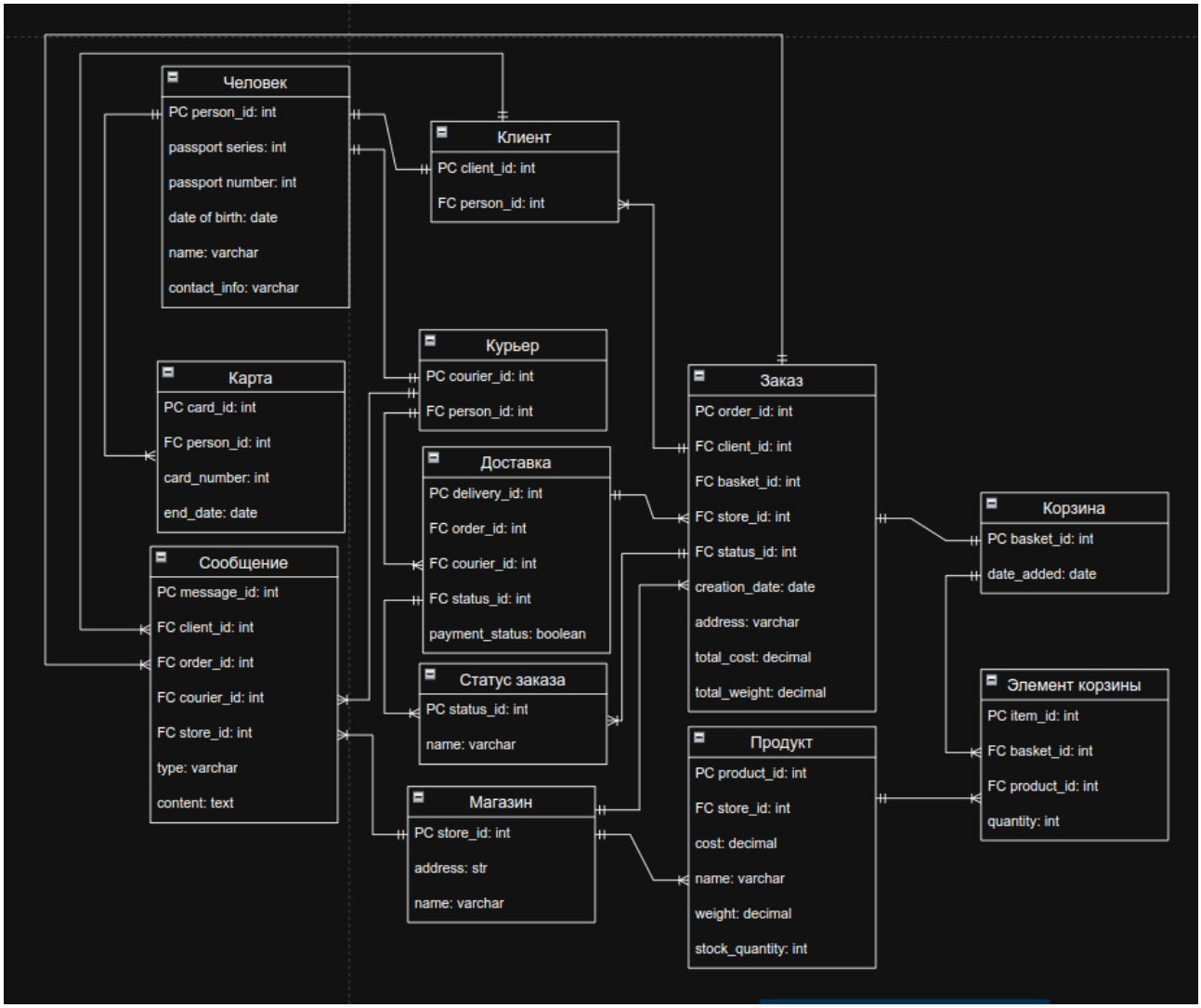
Мое приложение работает по такой логике:



Как открыть такую диаграмму:

1. Зайти в браузер и вбить в поисковик draw.io (← ссылка)
2. Далее нажать открыть существующий
3. Выбрать BPMN.draw.io

Как выглядит моя база данных:
(открывается также, как и верхняя, но только в третьем пункте ER.drawio выбираешь)

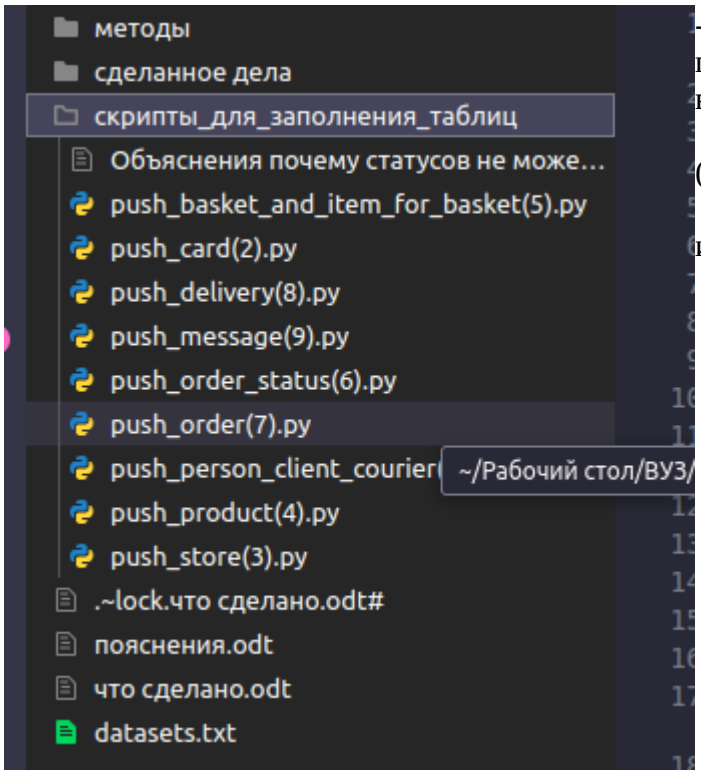


Далее ниже код для создания этой базы данных:

```
-- 1. Создаём таблицы
CREATE TABLE order_status(
status_id INT PRIMARY KEY,
name VARCHAR(50) NOT NULL UNIQUE
);

-- CREATE TABLE city(
-- city_id INT PRIMARY KEY,
-- name VARCHAR(255) NOT NULL
-- );

CREATE TABLE basket(
basket_id INT PRIMARY KEY,
date_added TIMESTAMP NOT NULL
);

CREATE TABLE person(
person_id INT PRIMARY KEY,
passport_series INT CHECK(passport_series BETWEEN 1000 AND 9999),
passport_number INT CHECK(passport_number BETWEEN 100000 AND 999999),
date_of_birth DATE NOT NULL,
name VARCHAR(255),
contact_info VARCHAR(255) NOT NULL
);

CREATE TABLE card(
card_id INT PRIMARY KEY,
person_id INT REFERENCES person(person_id),
card_number BIGINT NOT NULL CHECK(card_number > 0),
end_date TIMESTAMP NOT NULL
);

ALTER TABLE "card" ADD COLUMN "cvc" INTEGER NOT NULL ;

-- ALTER TABLE person ADD COLUMN card_id INT REFERENCES card(card_id);

CREATE TABLE client(
client_id INT PRIMARY KEY,
person_id INT REFERENCES person(person_id)
);

CREATE TABLE courier(
courier_id INT PRIMARY KEY,
person_id INT REFERENCES person(person_id)
);

CREATE TABLE store(
store_id INT PRIMARY KEY,
-- city_id INT REFERENCES city(city_id),
address TEXT NOT NULL,
```

```
name VARCHAR(255) NOT NULL
);

CREATE TABLE product(
product_id INT PRIMARY KEY,
store_id INT REFERENCES store(store_id),
cost DECIMAL(10, 2) NOT NULL CHECK(cost > 0),
name VARCHAR(255) NOT NULL,
weight DECIMAL(10, 2) NOT NULL,
stock_quantity INT NOT NULL
);

CREATE TABLE customer_order(
order_id INT PRIMARY KEY,
client_id INT REFERENCES client(client_id),
basket_id INT REFERENCES basket(basket_id),
store_id INT REFERENCES store(store_id),
status_id INT REFERENCES order_status(status_id),
creation_date TIMESTAMP NOT NULL,
address VARCHAR(255) NOT NULL,
cost DECIMAL(10, 2) NOT NULL CHECK(cost > 0),
total_weight DECIMAL(10, 2) NOT NULL
);

CREATE TABLE item(
item_id INT PRIMARY KEY,
basket_id INT REFERENCES basket(basket_id),
product_id INT REFERENCES product(product_id),
quantity INT NOT NULL
);

CREATE TABLE delivery(
delivery_id INT PRIMARY KEY,
order_id INT REFERENCES customer_order(order_id),
courier_id INT REFERENCES courier(courier_id),
status_id INT REFERENCES order_status(status_id),
payment_status BOOLEAN NOT NULL
);

CREATE TABLE message(
message_id INT PRIMARY KEY,
client_id INT REFERENCES client(client_id),
order_id INT REFERENCES customer_order(order_id),
courier_id INT REFERENCES courier(courier_id),
store_id INT REFERENCES store(store_id),
type VARCHAR(255) CHECK(type in (
'order_created', 'order_confirmed', 'payment_received',
'order_assembled', 'courier_assigned', 'on_the_way',
'arrived', 'delivered', 'delivery_problem', 'cancelled'
)),
content TEXT NOT NULL
);
```

Далее надо заполнить базу данных



- последовательно выполнения файлов, показано в скобках (1) — первым выполнить код

(2) — вторым выполнить код

и т. д.

Еще после заполнения таблиц можно запустить триггеры, которые будут вызываться после происхождения некоторых событий (у каждого триггера свое событие указывается)

BOT код для создания триггеров:

```
BEGIN;
-- 1) Функция для провепки возраста клиента, должен быть 18+
CREATE OR REPLACE FUNCTION check_age()
RETURNS TRIGGER AS $$
DECLARE
client_age INTERVAL;
BEGIN
SELECT age(p.date_of_birth) INTO client_age
FROM person p
JOIN client c ON p.person_id = c.person_id
WHERE c.client_id = NEW.client_id;
IF client_age < INTERVAL '18 years' THEN
RAISE EXCEPTION 'Клиент младше 18 лет. Заказ невозможен.';
END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER age_verification
```

```
BEFORE INSERT ON customer_order
FOR EACH ROW
EXECUTE FUNCTION check_age();

-- 2) Триггер для уменьшения количества товара на складе при добавлении товара в заказ
CREATE OR REPLACE FUNCTION update_stock()
RETURNS TRIGGER AS $$
DECLARE
current_stock INT;
BEGIN
SELECT stock_quantity INTO current_stock
FROM product
WHERE product_id = NEW.product_id
FOR UPDATE; -- Блокировка строки
IF current_stock < NEW.quantity THEN
RAISE EXCEPTION 'Недостаточно товара на складе. Доступно: %', current_stock;
END IF;
UPDATE product
SET stock_quantity = stock_quantity - NEW.quantity
WHERE product_id = NEW.product_id;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER stock_control
AFTER INSERT ON item
FOR EACH ROW
EXECUTE FUNCTION update_stock();

-- 3) Триггер для проверки срока действия карты (Нельзя заказывать если срок действия
истек)
CREATE OR REPLACE FUNCTION check_card_expiry()
RETURNS TRIGGER AS $$
DECLARE
card_expired BOOLEAN;
no_card BOOLEAN;
BEGIN
SELECT
COALESCE(c.end_date < CURRENT_DATE, TRUE),
c.card_id IS NULL
INTO card_expired, no_card
FROM client cl
JOIN person p ON cl.person_id = p.person_id
LEFT JOIN card c ON p.card_id = c.card_id
WHERE cl.client_id = NEW.client_id;
IF no_card THEN
RAISE EXCEPTION 'У клиента нет привязанной карты';
ELSIF card_expired THEN
RAISE EXCEPTION 'Карта клиента просрочена. Оплата невозможна.';
END IF;
RETURN NEW;
END;
```

```
$$ LANGUAGE plpgsql;

CREATE TRIGGER card_expiry_check
BEFORE INSERT ON delivery
FOR EACH ROW
EXECUTE FUNCTION check_card_expiry();

-- 4) Триггер для расчета общего веса заказа (при добавления продукта в корзину, вес заказа увеличивается)
CREATE OR REPLACE FUNCTION calculate_order_weight()
RETURNS TRIGGER AS $$
BEGIN
IF EXISTS (
SELECT 1 FROM customer_order
WHERE basket_id = NEW.basket_id
) THEN
UPDATE customer_order
SET total_weight = (
SELECT COALESCE(SUM(p.weight * i.quantity), 0)
FROM item i
JOIN product p ON i.product_id = p.product_id
WHERE i.basket_id = NEW.basket_id
)
WHERE basket_id = NEW.basket_id;
END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER weight_calculator
AFTER INSERT OR UPDATE OR DELETE ON item
FOR EACH ROW
EXECUTE FUNCTION calculate_order_weight();

COMMIT;
```