Санкт-Петербургский политехнический университет

Институт компьютерных наук и технологий

Высшая школа киберфизических систем и управления

Итоговый проект

по дисциплине «Базы данных»

Тема: «Создание базы данных для службы доставки товаров»

Выполнил:
студент гр. 3530902/70201 _____Т. И. Матченко

Проверил:
к.т.н., доцент _____ С. А. Нестеров

Санкт-Петербург

2020г

Содержание

Описание проекта

В данном проекте была создана база данных для службы доставки товаров. Она включает в себя таблицы

Модели базы данных созданы в инструменте Oracle SQL Developer Data Modeler

База данных реализована в системе управления базами данных MySQL.

Клиентское приложение написано на Java.

Логическая модель базы данных

В Data Modeler была создана логическая модель базы данных (рисунок 1).

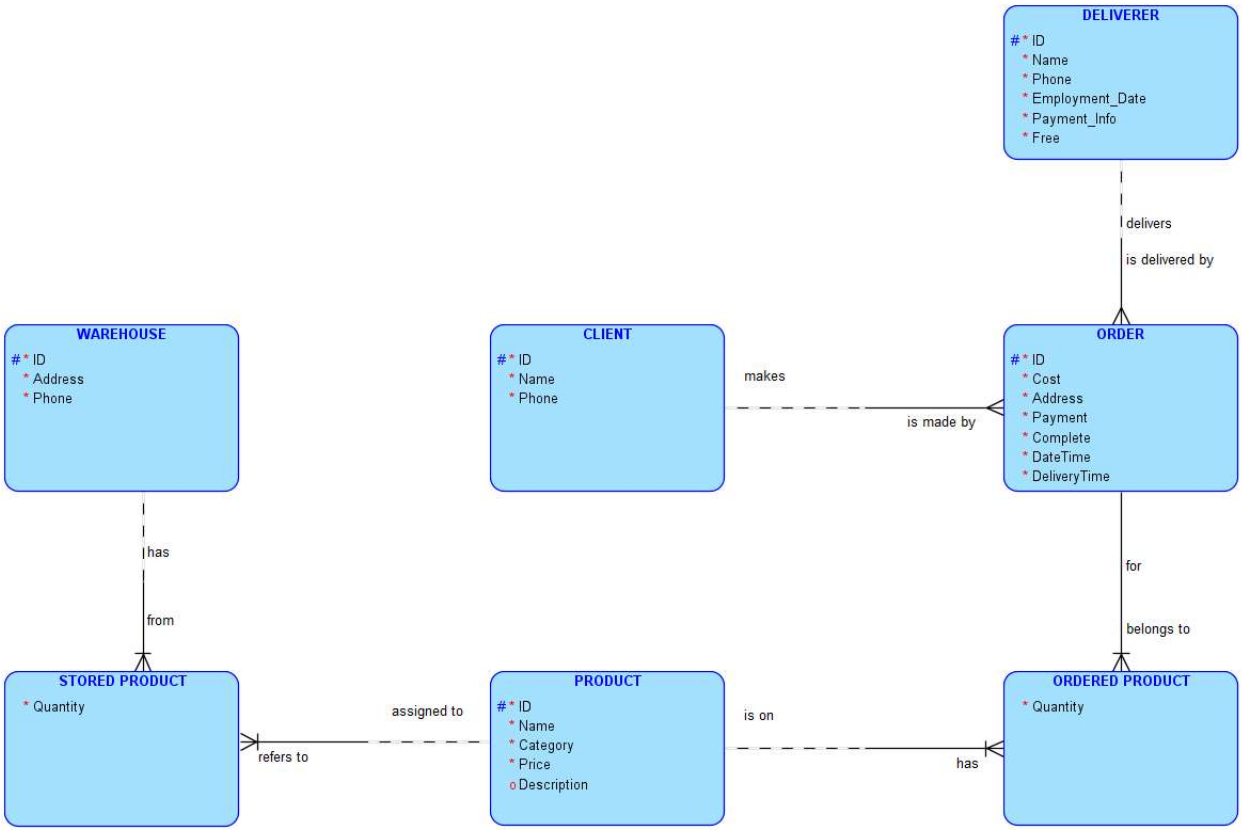


Рисунок 1 — Логическая модель базы данных

Реляционная модель базы данных

Логическая модель была преобразована в реляционную (рисунок 2). Сущности были преобразованы в таблицы, атрибуты - в поля таблиц.

- CLT – CLIENT
- DLV – DELIVERERER
- ORD – ORDER
- PRD – PRODUCT
- OPR – ORDERED PRODUCT
- WRH – WAREHOUSE
- SPR – STOCKED PRODUCT

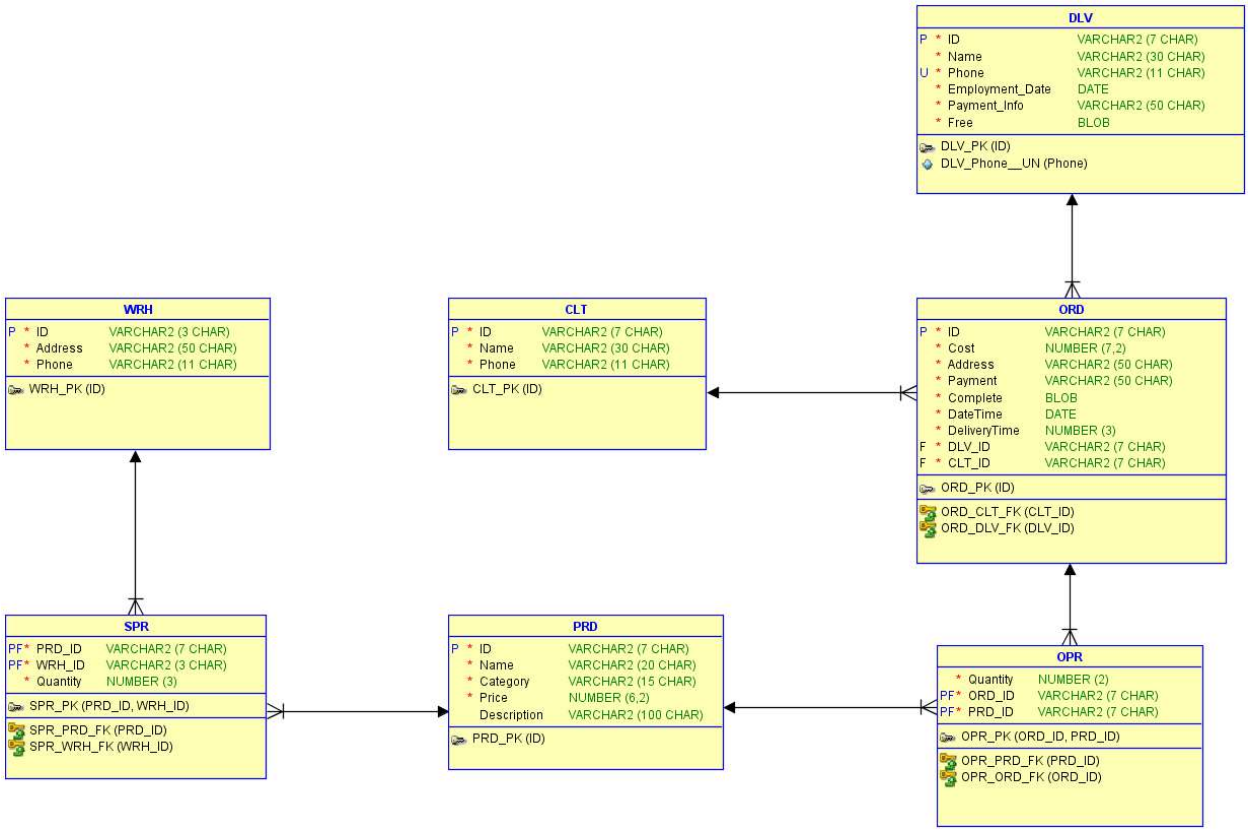


Рисунок 2 — Реляционная модель базы данных

Создание базы данных в MySQL

Проанализируем правила предметной области.

- Цена товаров/заказа должна быть больше 0;
- Нельзя заказать больше товаров, чем есть на складе;
- Количество товара на складе должно быть >= 0;
- Количество товара в заказе должно быть больше 0;
- Цена заказа складывается из суммы цен товаров в заказе;
- Номера телефонов должны быть в формате 8xxxxxxxxxx;

С помощью Data Modeler сгенерировали DDL-скрипт для создания таблиц и адаптировали его под работу в MySQL (приложение 1). Добавляем ограничения и заполняем таблицы с помощью скриптов с DML командами (приложение 2 и 3).

Заполненные таблицы:

```
mysql> select * from clt;
+----+-------+-------------+
| id | name  | phone       |
+----+-------+-------------+
| 11 | Nina  | 88118876532 |
| 12 | Tanya | 88117653979 |
| 13 | Maria | 88119871352 |
| 14 | Anna  | 88110962856 |
| 15 | Klava | 88118594738 |
+----+-------+-------------+
5 rows in set (0.00 sec)
```

Рисунок 3 – таблица CLT (клиентов)

```
mysql> select * from dlv;
+----+--------+-------------+---------------------+------------------+------+
| id | name   | phone       | employment_date     | payment_info     | free |
+----+--------+-------------+---------------------+------------------+------+
| 21 | Iosif  | 89117462056 | 2015-01-01 10:01:15 | 4716998573225739 | 1    |
| 22 | Kolya  | 89117689834 | 2017-02-01 10:01:15 | 4532896698742905 | 1    |
| 23 | Zina   | 89118926547 | 2020-03-24 09:12:15 | 4916965832084304 | 1    |
| 24 | Polina | 89118876543 | 2001-12-01 06:01:15 | 4929746864519044 | 1    |
| 25 | Frosia | 89116657493 | 2016-07-05 07:12:59 | 4929420584685895 | 1    |
+----+--------+-------------+---------------------+------------------+------+
5 rows in set (0.01 sec)
```

Рисунок 4 – таблица DLV (доставщиков)

```
mysql> select * from ord;
+----+-------+-------------------+---------+----------+---------------------+--------------+--------+--------+
| id | cost  | address           | payment | complete | datetime            | deliverytime | dlv_id | clt_id |
+----+-------+-------------------+---------+----------+---------------------+--------------+--------+--------+
| 31 | 35.00 | street 1, house 1 | cash    | 1        | 2020-03-25 07:12:59 |           40 | 21     | 11     |
| 32 |  3.00 | street 2, house 2 | cash    | 1        | 2020-03-27 09:12:59 |           30 | 22     | 12     |
| 33 | 50.00 | street 3, house 3 | cash    | 1        | 2020-03-15 12:44:01 |          150 | 23     | 13     |
| 34 | 16.00 | street 4, house 4 | cash    | 1        | 2020-03-17 15:13:43 |           90 | 24     | 14     |
| 35 | 21.00 | street 5, house 5 | cash    | 1        | 2020-03-20 07:12:59 |           45 | 25     | 15     |
+----+-------+-------------------+---------+----------+---------------------+--------------+--------+--------+
5 rows in set (0.00 sec)
```

Рисунок 5 – таблица ORD (заказов)

```
mysql> select * from prd;
+----+-----------+-----------------+-------+---------------+
| id | name      | category        | price | description   |
+----+-----------+-----------------+-------+---------------+
| 41 | potato    | vegetables      |  5.00 | fresh         |
| 42 | carrot    | vegetables      |  3.00 | fresh         |
| 43 | chocolate | sweet           | 10.00 | tasty but bad |
| 44 | soap      | household goods | 15.00 | always need   |
| 45 | candy     | sweet           |  4.00 | tasty but bad |
+----+-----------+-----------------+-------+---------------+
5 rows in set (0.00 sec)
```

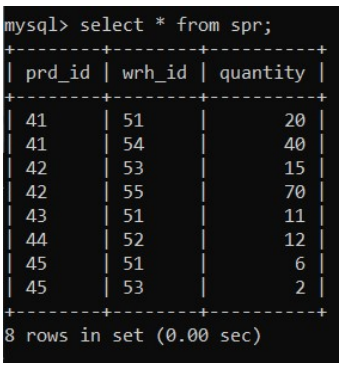
Рисунок 6 – таблица PRD (продуктов)

```
mysql> select * from opr;
+----------+--------+--------+
| quantity | ord_id | prd_id |
+----------+--------+--------+
|        3 | 31     | 41     |
|        2 | 31     | 43     |
|        1 | 32     | 42     |
|        5 | 33     | 43     |
|        4 | 34     | 45     |
|        3 | 35     | 41     |
|        2 | 35     | 42     |
+----------+--------+--------+
7 rows in set (0.02 sec)
```

Рисунок 7 – таблица OPR (заказанных товаров)

```
mysql> select * from wrh;
+----+--------------------+-------------+
| id | address            | phone       |
+----+--------------------+-------------+
| 51 | street 6, house 6  | 89116573848 |
| 52 | street 7, house 7  | 89113456274 |
| 53 | street 8, house 8  | 89116592746 |
| 54 | street 9, house 9  | 89116342856 |
| 55 | street 10, house 10 | 89116482064 |
+----+--------------------+-------------+
5 rows in set (0.00 sec)
```

Рисунок 8 – таблица WRH (складов)



```
mysql> select * from spr;
+--------+--------+----------+
| prd_id | wrh_id | quantity |
+--------+--------+----------+
| 41     | 51     |       20 |
| 41     | 54     |       40 |
| 42     | 53     |       15 |
| 42     | 55     |       70 |
| 43     | 51     |       11 |
| 44     | 52     |       12 |
| 45     | 51     |        6 |
| 45     | 53     |        2 |
+--------+--------+----------+
8 rows in set (0.00 sec)
```

Рисунок 9 – таблица SPR (товаров на складе)

Соответствие нормальным формам:

1НФ:

Все поля таблиц имеют только одно значение.

2НФ:

Все таблицы соответствуют 1НФ и каждое поле каждой таблицы зависит от полного первичного ключа.

Таблицы client (CLT), deliverer (DLV), order (ORD), warehouse (WRH), products (PRD) – первичный ключ состоит из id. От него зависят все атрибуты.

Таблицы ordered product (OPR), stocked product (SPR) – первичный ключ состоит из id заказа или склада соответственно и id продукта. От него зависят остальные атрибуты (количество).

3НФ:

Все таблицы соответствуют 2НФ и поля таблиц не составляют транзитивных зависимостей.

Ни в одной таблице поле не зависит ни от какого другого поля, кроме полей, входящих в первичный ключ.

НФБК:

Во всех таблицах отсутствуют функциональных зависимости между полями, входящими в первичный ключ таблицы.

Таблицы client (CLT), deliverer (DLV), order (ORD), warehouse (WRH), products (PRD) – первичный ключ состоит только из одного атрибута (id).

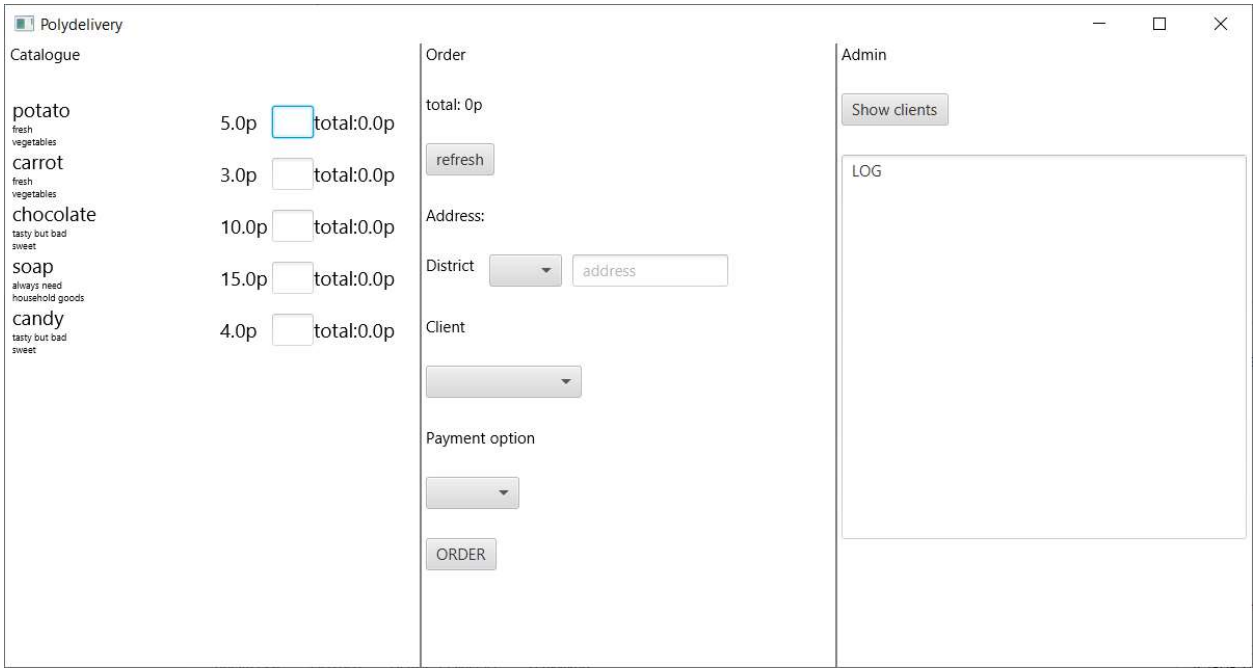
Таблицы ordered product (OPR), stocked product (SPR) – при изменении id заказа/склада может остаться неизменным id товара.

Создание клиентского приложения

Написано на Java. Код в приложении 4.

Позволяет делать заказы, просматривать таблицу клиентов, товаров.

Финальная версия приложения выглядит так:



Тестирование приложения

Неверные входные данные

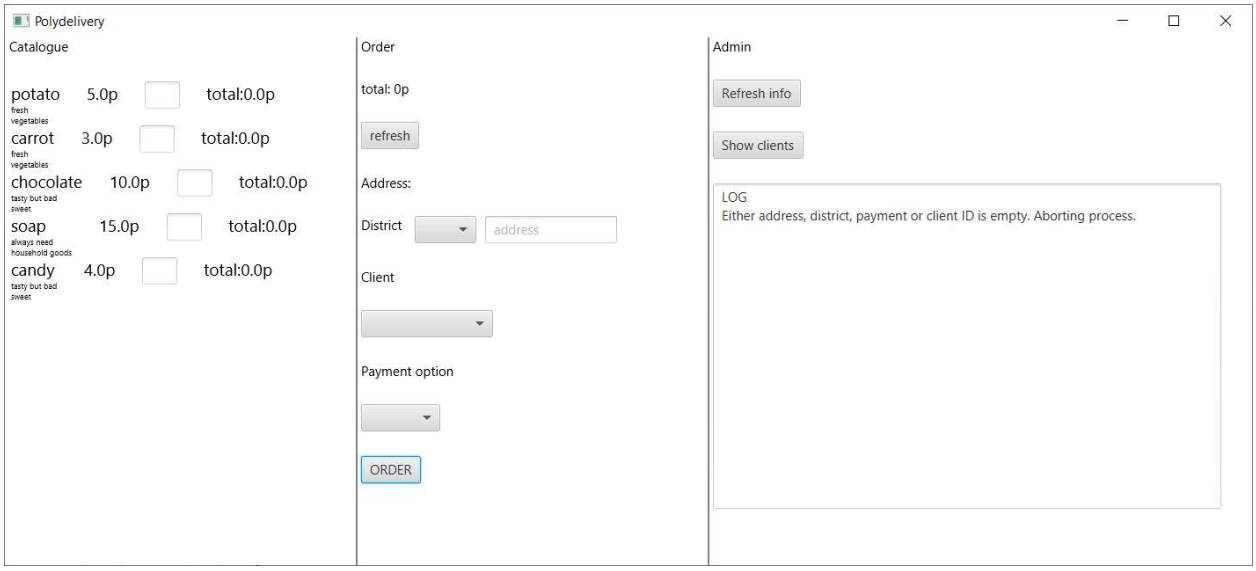
Несоответствие типов

Программно отключена возможность ввести больше двух цифр (и только цифр) в поле количества товара.

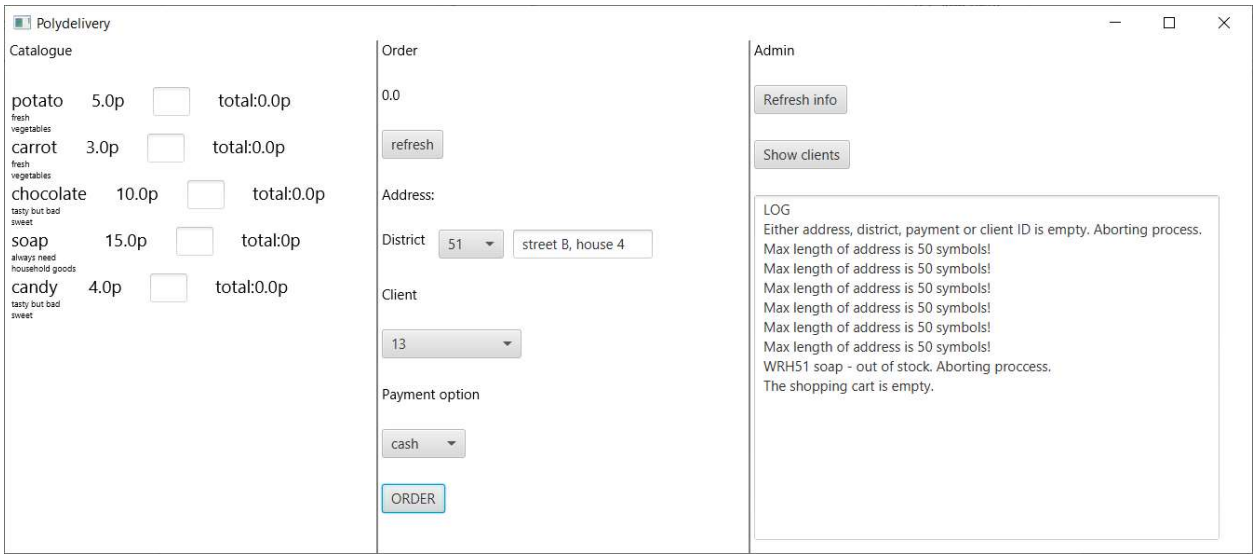
В тестировании внешний вид приложения немного отличается от финальной версии.

Заполнены не все поля

В лог выйдет сообщение об ошибке при отсутствии информации о заказчике…

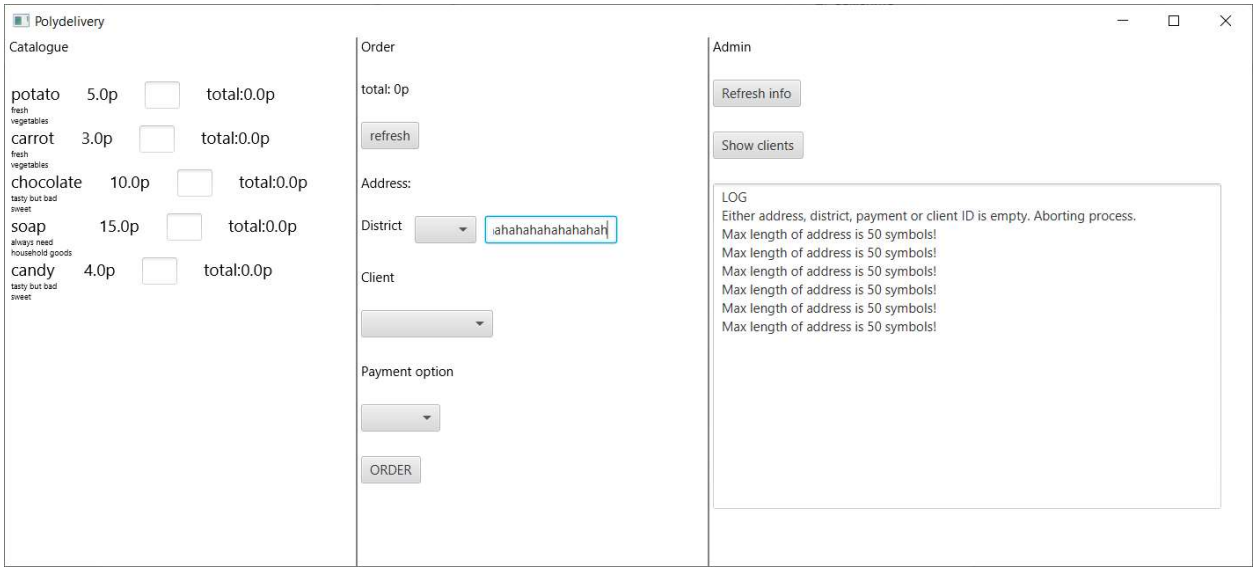


…или товаров.



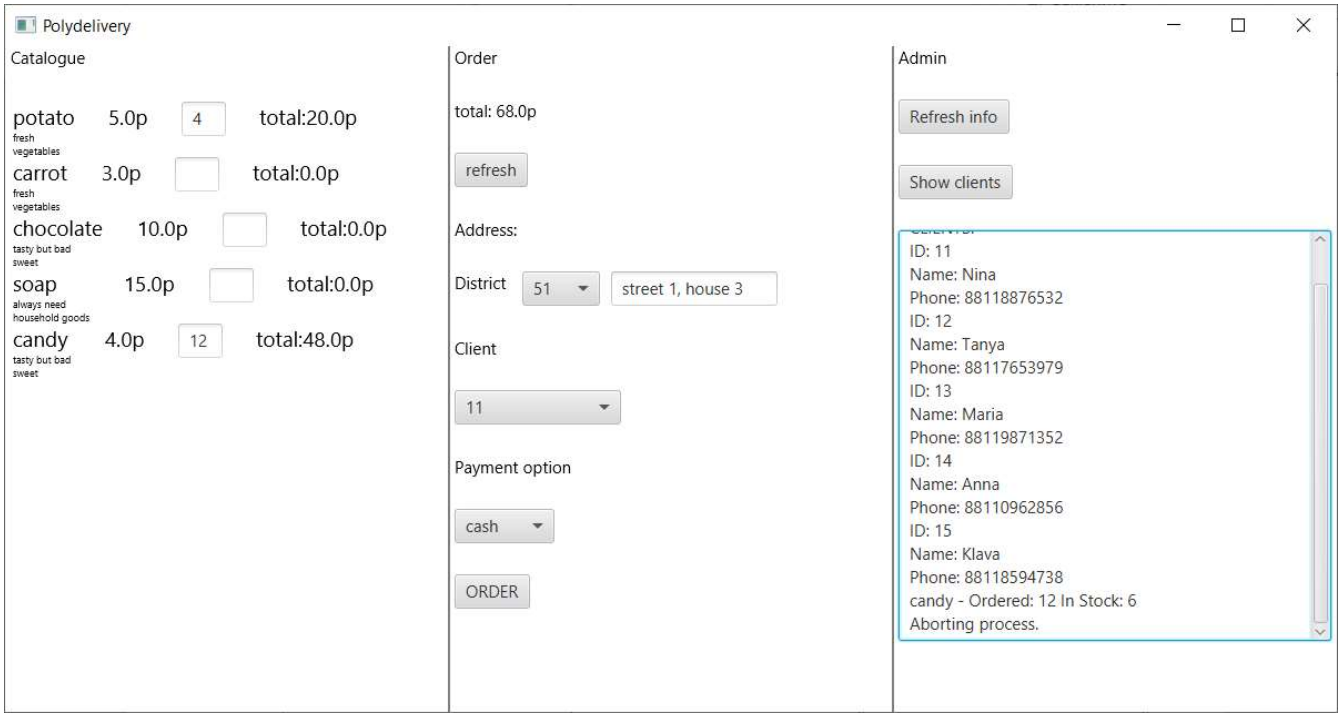
Ввод данных больше допустимой длины

Максимальная длина адреса – 50 символов. Возможность ввести новый символ будет программно заблокирована, останется только удалять предыдущие. В логе будет об этом сообщение.

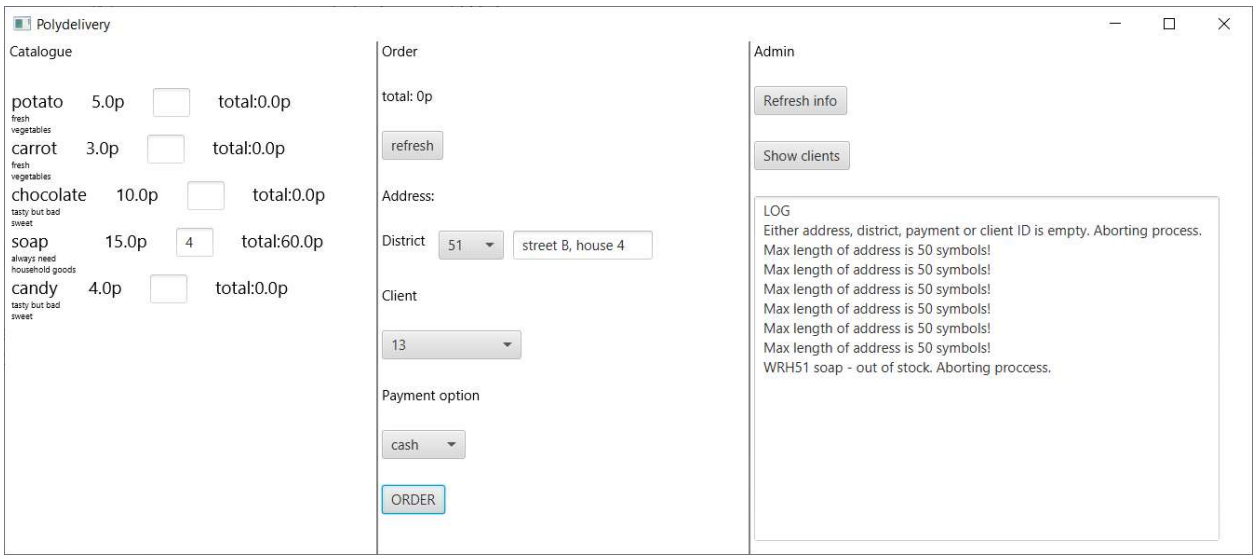


Создание заказа

На складе может быть недостаточно товаров…



…или вовсе не быть. Различие в том, что в первом случае запрос к SQL вернет количество, а во втором случае – придет пустой ответ, поэтому handling разный.



И действительно, мыло (prd_id = 44) есть только на складе в районе 52. Чтобы заказать мыло, адрес должен принадлежать 52 району.

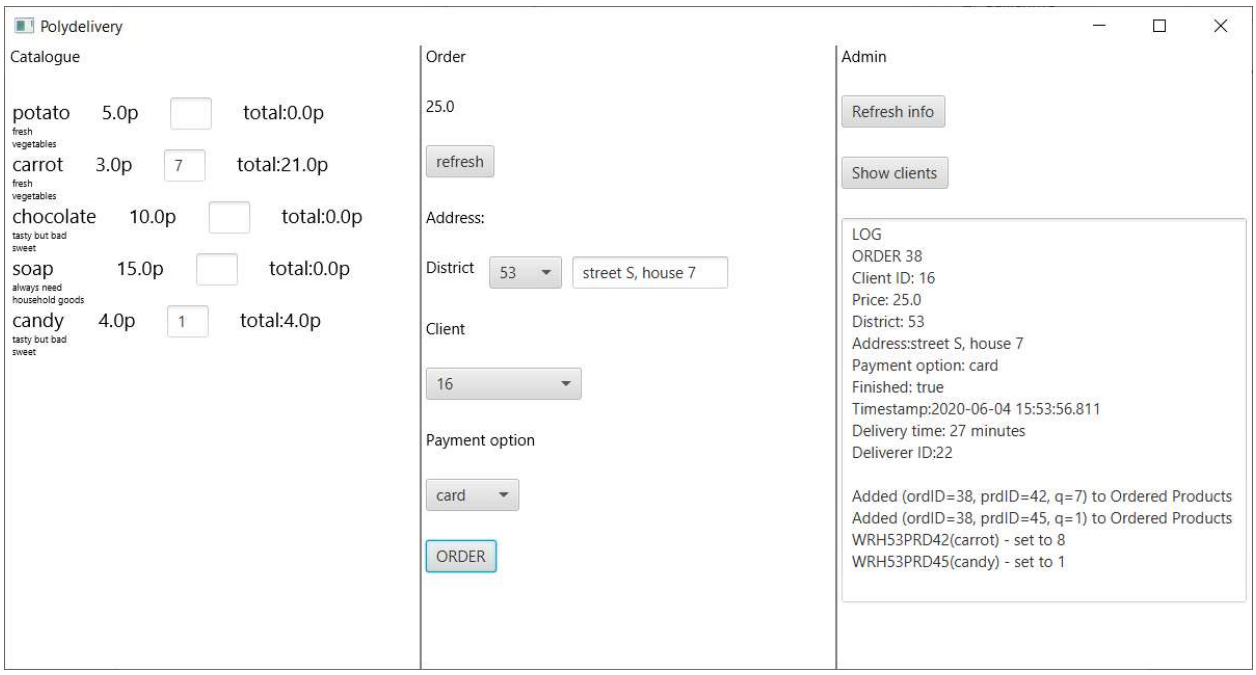
```
mysql> select * from spr where prd_id=44;
+---------+--------+----------+
| prd_id | wrh_id | quantity |
+---------+--------+----------+
| 44     | 52     |       12 |
+---------+--------+----------+
1 row in set (0.00 sec)
```

Проверка нового заказа

База до:

```
mysql> select * from ord;
+----+-------+------------------+---------+----------+---------------------+--------------+--------+--------+
| id | cost  | address          | payment | complete | datetime            | deliverytime | dlv_id | clt_id |
+----+-------+------------------+---------+----------+---------------------+--------------+--------+--------+
| 31 | 35.00 | street 1, house 1 | cash    | 1        | 2020-03-25 07:12:59 |           40 | 21     | 11     |
| 32 |  3.00 | street 2, house 2 | cash    | 1        | 2020-03-27 09:12:59 |           30 | 22     | 12     |
| 33 | 50.00 | street 3, house 3 | cash    | 1        | 2020-03-15 12:44:01 |          150 | 23     | 13     |
| 34 | 16.00 | street 4, house 4 | cash    | 1        | 2020-03-17 15:13:43 |           90 | 24     | 14     |
| 35 | 21.00 | street 5, house 5 | cash    | 1        | 2020-03-20 07:12:59 |           45 | 25     | 15     |
| 36 | 24.00 | street 1, house 3 | cash    | 1        | 2020-06-04 14:15:41 |           37 | 24     | 11     |
| 37 |  9.00 | street B, house 4 | cash    | 1        | 2020-06-04 15:03:00 |           25 | 24     | 11     |
+----+-------+------------------+---------+----------+---------------------+--------------+--------+--------+
7 rows in set (0.00 sec)
```

Делаем заказ:



База после:

```
mysql> select * from ord;
+----+-------+-------------------+---------+----------+---------------------+--------------+--------+--------+
| id | cost  | address           | payment | complete | datetime            | deliverytime | dlv_id | clt_id |
+----+-------+-------------------+---------+----------+---------------------+--------------+--------+--------+
| 31 | 35.00 | street 1, house 1 | cash    | 1        | 2020-03-25 07:12:59 |           40 | 21     | 11     |
| 32 |  3.00 | street 2, house 2 | cash    | 1        | 2020-03-27 09:12:59 |           30 | 22     | 12     |
| 33 | 50.00 | street 3, house 3 | cash    | 1        | 2020-03-15 12:44:01 |          150 | 23     | 13     |
| 34 | 16.00 | street 4, house 4 | cash    | 1        | 2020-03-17 15:13:43 |           90 | 24     | 14     |
| 35 | 21.00 | street 5, house 5 | cash    | 1        | 2020-03-20 07:12:59 |           45 | 25     | 15     |
| 36 | 24.00 | street 1, house 3 | cash    | 1        | 2020-06-04 14:15:41 |           37 | 24     | 11     |
| 37 |  9.00 | street B, house 4 | cash    | 1        | 2020-06-04 15:03:00 |           25 | 24     | 11     |
| 38 | 25.00 | street S, house 7 | card    | 1        | 2020-06-04 15:53:56 |           27 | 22     | 16     |
+----+-------+-------------------+---------+----------+---------------------+--------------+--------+--------+
8 rows in set (0.00 sec)
```

Доставщик выбирается случайный из доступных, время доставки – случайное число от 25 до 44.

Тестирование базы данных

Каким образом предотвращается insert/update неверных значений?

Работа производится на MySQL 5.0.67, поэтому вместо CONSTRAINT… CHECK (это поддерживается только на MySQL 8.0.16 и старше) используется TRIGGER… BEFORE INSERT ON…

Пример триггера (предотвращает добавления товаров, чья цена меньше или равна 0):

triggers.sql

```
delimiter $$
create trigger prd_price before insert on PRD
for each row
begin
if new.price <= 0 then
CALL raise_error;
end if;
end;$$
```

При срабатывании условия можно было бы использовать, например, signal sqlstate ‘45000’, что вызвало бы пользовательскую ошибку 45000, но signal sqlstate поддерживается только на MySQL 5.5 и старше, поэтому используем костыль: вызываем несуществующую функцию raise_error. Возникает ошибка, и новая строчка не вставляется, что и было нашей целью.

У товаров должна быть положительная цена. Пробуем добавить товар с ценой 0 и -1:

```
mysql> insert into prd values(46, 'juice','liquid',0,'apple');
ERROR 1305 (42000): PROCEDURE tanya.raise_error does not exist
mysql> insert into prd values(46, 'juice','liquid',-1,'apple');
ERROR 1305 (42000): PROCEDURE tanya.raise_error does not exist
```

У заказа должна быть положительная цена. Пробуем добавить заказ с ценой (второй атрибут) 0:

```
mysql> insert into ord values(39,0,'ha','cash',1,'2020-06-04 15:53:57',31,22,16);
ERROR 1305 (42000): PROCEDURE tanya.raise_error does not exist
```

Количество товара на складе неотрицательно. Пробуем добавить продукт №45 на склад №52 с количеством -30:

```
mysql> insert into spr values (45,52,-30);
ERROR 1305 (42000): PROCEDURE tanya.raise_error does not exist
```

Количество заказанного товара положительно. Пытаемся поместить 0 штук товара №44 в заказ №38.

```
mysql> insert into opr values (0,38,44);
ERROR 1305 (42000): PROCEDURE tanya.raise_error does not exist
```

Номера телефонов должны быть в формате 8xxxxxxxxxx.

Присутствует символ:

```
mysql> insert into clt values(17,'Ivan','8123456789t');
ERROR 1305 (42000): PROCEDURE tanya.raise_error does not exist
```

Длина меньше 11 (здесь 10):

```
mysql> insert into clt values(17,'Ivan','8123456789');
ERROR 1305 (42000): PROCEDURE tanya.raise_error does not exist
```

Начинается не с 8:

```
mysql> insert into clt values(17,'Ivan','71234567890');
ERROR 1305 (42000): PROCEDURE tanya.raise_error does not exist
```

Длина больше 11:

```
mysql> insert into clt values(17,'Ivan','712345678905');
ERROR 1406 (22001): Data too long for column 'phone' at row 1
```

Приложение 1

inittables.ddl

```
CREATE TABLE clt (
    id     VARCHAR(7) NOT NULL,
    name   VARCHAR(30) NOT NULL,
    phone  VARCHAR(11) NOT NULL
);

ALTER TABLE clt ADD CONSTRAINT clt_pk PRIMARY KEY ( id );

CREATE TABLE dlv (
    id               VARCHAR(7) NOT NULL,
    name             VARCHAR(30) NOT NULL,
    phone            VARCHAR(11) NOT NULL,
    employment_date  DATETIME NOT NULL,
    payment_info     VARCHAR(50) NOT NULL,
    free             BLOB NOT NULL
);

ALTER TABLE dlv ADD CONSTRAINT dlv_pk PRIMARY KEY ( id );

ALTER TABLE dlv ADD CONSTRAINT dlv_phone__un UNIQUE ( phone );

CREATE TABLE opr (
    quantity  SMALLINT NOT NULL,
    ord_id    VARCHAR(7) NOT NULL,
    prd_id    VARCHAR(7) NOT NULL
);
```

```
ALTER TABLE opr ADD CONSTRAINT opr_pk PRIMARY KEY ( ord_id,
                                                    prd_id );

CREATE TABLE ord (
    id            VARCHAR(7) NOT NULL,
    cost          DECIMAL(7, 2) NOT NULL,
    address       VARCHAR(50) NOT NULL,
    payment       VARCHAR(50) NOT NULL,
    complete      BLOB NOT NULL,
    datetime      DATETIME NOT NULL,
    deliverytime  SMALLINT NOT NULL,
    dlv_id        VARCHAR(7) NOT NULL,
    clt_id        VARCHAR(7) NOT NULL
);

ALTER TABLE ord ADD CONSTRAINT ord_pk PRIMARY KEY ( id );

CREATE TABLE prd (
    id           VARCHAR(7) NOT NULL,
    name         VARCHAR(20) NOT NULL,
    category     VARCHAR(15) NOT NULL,
    price        DECIMAL(6, 2) NOT NULL,
    description  VARCHAR(100)
);

ALTER TABLE prd ADD CONSTRAINT prd_pk PRIMARY KEY ( id );

CREATE TABLE spr (
```

```
    prd_id    VARCHAR(7) NOT NULL,
    wrh_id    VARCHAR(3) NOT NULL,
    quantity  SMALLINT NOT NULL
);

ALTER TABLE spr ADD CONSTRAINT spr_pk PRIMARY KEY ( prd_id,
                                                    wrh_id );

CREATE TABLE wrh (
    id       VARCHAR(3) NOT NULL,
    address  VARCHAR(50) NOT NULL,
    phone    VARCHAR(11) NOT NULL
);

ALTER TABLE wrh ADD CONSTRAINT wrh_pk PRIMARY KEY ( id );

ALTER TABLE opr
    ADD CONSTRAINT opr_ord_fk FOREIGN KEY ( ord_id )
        REFERENCES ord ( id );

ALTER TABLE opr
    ADD CONSTRAINT opr_prd_fk FOREIGN KEY ( prd_id )
        REFERENCES prd ( id );

ALTER TABLE ord
    ADD CONSTRAINT ord_clt_fk FOREIGN KEY ( clt_id )
        REFERENCES clt ( id );

ALTER TABLE ord
```

```
    ADD CONSTRAINT ord_dlv_fk FOREIGN KEY ( dlv_id )

        REFERENCES dlv ( id );


ALTER TABLE spr

    ADD CONSTRAINT spr_prd_fk FOREIGN KEY ( prd_id )

        REFERENCES prd ( id );


ALTER TABLE spr

    ADD CONSTRAINT spr_wrh_fk FOREIGN KEY ( wrh_id )

        REFERENCES wrh ( id );
```

Приложение 2

triggers.sql

```
delimiter $$
create trigger prd_price before insert on PRD
for each row
begin
if new.price <= 0 then
CALL raise_error;
end if;
end;$$

create trigger prd_price_upd before update on PRD
for each row
begin
if new.price <= 0 then
CALL raise_error;
end if;
end;$$

CREATE TRIGGER ord_cost before insert on ORD
for each row
begin
if new.cost <= 0 then
CALL raise_error;
end if;
end;$$

CREATE TRIGGER ord_cost_upd before update on ORD
for each row
begin
if new.cost <= 0 then
CALL raise_error;
end if;
end;$$

create trigger spr_quantity before insert on spr
for each row
begin
if new.quantity < 0 then
CALL raise_error;
end if;
end;$$

create trigger spr_quantity_upd before update on spr
for each row
begin
if new.quantity < 0 then
CALL raise_error;
```

```
end if;
end;$$

create trigger opr_quantity before insert on opr
for each row
begin
if new.quantity <= 0 then
CALL raise_error;
end if;
end;$$

create trigger opr_quantity_upd before update on opr
for each row
begin
if new.quantity <= 0 then
CALL raise_error;
end if;
end;$$

create trigger clt_phone before insert on clt
for each row
begin
if (new.phone REGEXP '8[0-9]{10}' = 0) then
CALL raise_error;
end if;
end;$$

create trigger clt_phone_upd before update on clt
for each row
begin
if (new.phone REGEXP '8[0-9]{10}' = 0) then
CALL raise_error;
end if;
end;$$

delimiter ;
```

Приложение 3

filltables.sql

```
insert into clt(id, name, phone)

values(11, 'Nina', 88118876532);

insert into clt(id, name, phone)

values(12, 'Tanya', 88117653979);

insert into clt(id, name, phone)

values(13, 'Maria', 88119871352);

insert into clt(id, name, phone)

values(14, 'Anna', 88110962856);

insert into clt(id, name, phone)

values(15, 'Klava', 88118594738);




insert into dlv (id, name, phone, employment_date, payment_info, free)

values(21, 'Iosif', 89117462056, '2015-01-01 10:01:15', '4716998573225739',
true);



insert into dlv (id, name, phone, employment_date, payment_info, free)

values(22, 'Kolya', 89117689834, '2017-02-01 10:01:15', '4532896698742905',
true);



insert into dlv (id, name, phone, employment_date, payment_info, free)

values(23, 'Zina', 89118926547, '2020-03-24 9:12:15', '4916965832084304', true);



insert into dlv (id, name, phone, employment_date, payment_info, free)

values(24, 'Polina', 89118876543, '2001-12-01 6:01:15', '4929746864519044',
true);
```

```
insert into dlv (id, name, phone, employment_date, payment_info, free)

values(25, 'Frosia', 89116657493, '2016-07-05 07:12:59', '4929420584685895',
true);



insert into ord (id, cost, address, payment, complete, datetime, deliverytime,
dlv_id, clt_id)

values(31, 35, 'street 1, house 1', 'cash', true, '2020-03-25 07:12:59', 40, 21,
11);



insert into ord (id, cost, address, payment, complete, datetime, deliverytime,
dlv_id, clt_id)

values(32, 3, 'street 2, house 2', 'cash', true, '2020-03-27 09:12:59', 30, 22,
12);



insert into ord (id, cost, address, payment, complete, datetime, deliverytime,
dlv_id, clt_id)

values(33, 50, 'street 3, house 3', 'cash', true, '2020-03-15 12:44:01', 150,
23, 13);



insert into ord (id, cost, address, payment, complete, datetime, deliverytime,
dlv_id, clt_id)

values(34, 16, 'street 4, house 4', 'cash', true, '2020-03-17 15:13:43', 90, 24,
14);



insert into ord (id, cost, address, payment, complete, datetime, deliverytime,
dlv_id, clt_id)

values(35, 21, 'street 5, house 5', 'cash', true, '2020-03-20 07:12:59', 45, 25,
15);





insert into prd  (id, name, category, price, description)

values(41, 'potato', 'vegetables', 5, 'fresh');
```

```
insert into prd  (id, name, category, price, description)
values(42, 'carrot', 'vegetables', 3, 'fresh');

insert into prd  (id, name, category, price, description)
values(43, 'chocolate', 'sweet', 10, 'tasty but bad');

insert into prd  (id, name, category, price, description)
values(44, 'soap', 'household goods', 15, 'always need');

insert into prd  (id, name, category, price, description)
values(45, 'candy', 'sweet', 4, 'tasty but bad');



insert into wrh (id, address, phone)
values(51, 'street 6, house 6', 89116573848);

insert into wrh (id, address, phone)
values(52, 'street 7, house 7', 89113456274);

insert into wrh (id, address, phone)
values(53, 'street 8, house 8', 89116592746);

insert into wrh (id, address, phone)
values(54, 'street 9, house 9', 89116342856);

insert into wrh (id, address, phone)
```

```
values(55, 'street 10, house 10', 89116482064);



insert into opr (quantity, ord_id, prd_id)
values(3, 31, 41);

insert into opr (quantity, ord_id, prd_id)
values(2, 31, 43);

insert into opr (quantity, ord_id, prd_id)
values(1, 32, 42);

insert into opr (quantity, ord_id, prd_id)
values(5, 33, 43);

insert into opr (quantity, ord_id, prd_id)
values(4, 34, 45);

insert into opr (quantity, ord_id, prd_id)
values(3, 35, 41);

insert into opr (quantity, ord_id, prd_id)
values(2, 35, 42);



insert into spr (prd_id, wrh_id, quantity)
values(41, 51, 20);
```

```
insert into spr (prd_id, wrh_id, quantity)
values(42, 53, 15);

insert into spr (prd_id, wrh_id, quantity)
values(41, 54, 40);

insert into spr (prd_id, wrh_id, quantity)
values(42, 55, 70);

insert into spr (prd_id, wrh_id, quantity)
values(43, 51, 11);

insert into spr (prd_id, wrh_id, quantity)
values(44, 52, 12);

insert into spr (prd_id, wrh_id, quantity)
values(45, 53, 2);

insert into spr (prd_id, wrh_id, quantity)
values(45, 51, 6);
```

Приложение 4

ProductInCart.java

```
import javafx.scene.control.TextField;
import javafx.scene.layout.ColumnConstraints;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.scene.text.Text;

/**
 * Created by Tanya Matchenko.
 * A class for box in the catalogue - a product.
 */
public class ProductInCart {
    private GridPane box = new GridPane();
    private String name;
    private String description;
    private String category;
    private double price;
    private double totalPrice;
    private int id;
    private int quantity;

    private TextField fldQuantity;
    private Text textName;
    private Text textDesc;
    private Text textPrice;
    private Text textTotalPrice;
```

```
public ProductInCart(int id, String name, String description, String category, double price) {
    this.name = name;
    this.description = description;
    this.category = category;
    this.price = price; totalPrice = 0;
    this.id = id;
    quantity = 0;

    //name, description, price, total price...
    textName = new Text(name); textName.setFont(Font.font(19));
    textDesc = new Text(description+"\n"+category); textDesc.setFont(Font.font(9));
    textPrice = new Text(price + "p"); textPrice.setFont(Font.font(19));
    textTotalPrice = new Text("total:"+totalPrice+"p"); textTotalPrice.setFont(Font.font(19));

    //setting up a field for quantity input
    fldQuantity = new TextField(); fldQuantity.setMaxSize(40, 30);
    fldQuantity.textProperty().addListener( (observable, oldValue, newValue) -> {
        //validating int, setting value
        if ( newValue.matches("\\d{1,2}")){
            fldQuantity.setText(newValue);
            quantity = Integer.parseInt(newValue);
            totalPrice = price*quantity;
            textTotalPrice.setText("total:"+totalPrice+"p");
        }
        else if (newValue.length()==0) {
            quantity = 0;
            totalPrice = 0;
            textTotalPrice.setText("total:0p");
        }
        else fldQuantity.setText(oldValue);
```

```
        });

        textTotalPrice.setLayoutX(300);
        box.add(new VBox(textName,textDesc), 0, 0);
        box.add(textPrice, 1, 0);
        box.add(fldQuantity, 2, 0);
        box.add(textTotalPrice, 3, 0);
        ColumnConstraints colCon0 = new ColumnConstraints();
        colCon0.setPrefWidth(200);
        ColumnConstraints colCon1 = new ColumnConstraints();
        colCon1.setPrefWidth(50);
        box.getColumnConstraints().addAll(colCon0,colCon1);
    }

    /**
     * returns the product's wrapper - GridPane
     * @return
     */
    public GridPane getBox(){
        return this.box;
    }
    public int getId() {
        return id;
    }
    public String getName(){
        return name;
    }
    public double getTotalPrice() {
        return totalPrice;
    }
    public int getQuantity(){
```

```
        return quantity;
    }
    /**
     * sets quantity
     * @param quantity
     */
    public void setQuantity(int quantity){
        fldQuantity.setText(quantity+"");
        this.quantity = quantity;
        totalPrice = price*quantity;
        textTotalPrice.setText("total:"+totalPrice+"p");
    }
    /**
     * sets info
     * @param name
     * @param description
     * @param category
     * @param price
     */
    public void setInfo(String name, String description, String category, double price) {
        this.name = name; textName.setText(name);
        this.description = description; textDesc.setText(description+"\n"+category);
        this.price = price; textPrice.setText(price+"p");
        totalPrice = price*quantity;
        textTotalPrice.setText("total:"+totalPrice+"p");
    }
}
```

Main.java

```
import java.util.ArrayList;
import java.util.Random;
import java.sql.*;
import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.ScrollPane;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.control.ScrollPane.ScrollBarPolicy;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.shape.Line;
import javafx.scene.text.Text;
import javafx.stage.Stage;

/**
 * created by Tanya Matchenko. Semester project.
 * Client for delivery service. Works with DB.
 */
public class Main extends Application {

    // поля для соединения с бд
    private static Connection conn;
    private static Statement stmt;
```

```
    // остальное
    public static Group root = new Group();
    private static Scene scene;
    private static ArrayList<ProductInCart> catalogueList = new ArrayList<>();
    private static VBox boxCatalogue = new VBox();
    private static ObservableList<String> districts = FXCollections.observableArrayList();
    private static ObservableList<Integer> clients = FXCollections.observableArrayList();
    private static TextArea fieldLog;
    private static int currentOrderId = 0;
    private static Random rnd = new Random();

    /**
     * Connects with database
     */
    private static void initSql() {
        try {
            String url = "jdbc:mysql://localhost:3306/tanya?&serverTimezone=UTC";
            String user = "root";
            // register
            Class.forName("com.mysql.jdbc.Driver");
            // создаем подключение к бд
            conn = DriverManager.getConnection(url, user, "");
            stmt = conn.createStatement();
            conn.setAutoCommit(false);
        } catch (SQLException e) {
            log(e.getMessage());
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            log(e.getLocalizedMessage());
            e.printStackTrace();
        }
```

```
    }

    /**
     * Spawns UI
     */
    private static void initMenu() {
        scene = new Scene(root, 1200, 600);

        // граница
        Line border1 = new Line(400, 0, 400, 600);
        Line border2 = new Line(800, 0, 800, 600);
        // делаем границу менюшки всегда до конца экрана
        scene.heightProperty().addListener((obs, oldValue, newValue) -> border1.setEndY(newValue.doubleValue()));
        scene.heightProperty().addListener((obs, oldValue, newValue) -> border2.setEndY(newValue.doubleValue()));

        // заголовки
        Text textHeadCatalogue = new Text("Catalogue");
        Text textHeadOrder = new Text("Order");
        Text textHeadAdmin = new Text("Admin");
        // коробки
        VBox boxCataloguePart = new VBox(28, textHeadCatalogue);
        boxCataloguePart.setLayoutX(5);
        VBox boxOrderPart = new VBox(28, textHeadOrder);
        boxOrderPart.setLayoutX(405);
        VBox boxAdminPart = new VBox(28, textHeadAdmin);
        boxAdminPart.setLayoutX(805);

        // #region catalogue
        // catalogue
        ScrollPane spCatalogue = new ScrollPane(boxCatalogue);
        spCatalogue.setStyle("-fx-background: white; -fx-border-color: white;");
```

```
        spCatalogue.setMaxHeight(560);
        spCatalogue.setPrefWidth(390);
        spCatalogue.setHbarPolicy(ScrollBarPolicy.NEVER);
        spCatalogue.setVbarPolicy(ScrollBarPolicy.AS_NEEDED);
        boxCataloguePart.getChildren().addAll(spCatalogue);
        // #endregion

        // #region order
        // total price
        Text textTotalPrice = new Text("total: 0p");
        Button btnRefreshPrice = new Button("refresh");
        btnRefreshPrice.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                textTotalPrice.setText("total: " + getTotalPrice() + "p");
            }
        });

        // address
        Text textAddress = new Text("Address:");
        // district
        Text textDistrict = new Text("District ");
        ComboBox<String> fieldDistrict = new ComboBox<>(districts);
        fieldDistrict.setVisibleRowCount(4);
        fieldDistrict.setPrefWidth(70);
        TextField fieldAddress = new TextField();
        fieldAddress.setMaxWidth(150);
        fieldAddress.setPromptText("address");
        // set max address length
        fieldAddress.textProperty().addListener((observable, oldValue, newValue) -> {
            if (newValue.length() < 50)
```

```
            fieldAddress.setText(newValue);
        else {
            fieldAddress.setText(oldValue);
            log("Max length of address is 50 symbols!");
        }
    });
    HBox boxAddress = new HBox(10, textDistrict, fieldDistrict, fieldAddress);

    // client
    Text textClient = new Text("Client");
    ComboBox<Integer> fieldClientId = new ComboBox<>(clients);
    fieldClientId.setVisibleRowCount(4);
    fieldClientId.setPrefWidth(150);

    // payment
    Text textPaymentOption = new Text("Payment option");
    ObservableList<String> paymentOptions = FXCollections.observableArrayList("cash", "card");
    ComboBox<String> fieldPayment = new ComboBox<>(paymentOptions);
    fieldPayment.setVisibleRowCount(4);
    fieldPayment.setPrefWidth(90);

    // make order
    Button btnMakeOrder = new Button("ORDER");
    btnMakeOrder.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            // #region ORDER
            // check address, district, payment, client
            String address = fieldAddress.getText();
            String payment = fieldPayment.getValue();
            String wrhId = fieldDistrict.getValue();
```

```
if (address.equals("") || payment == null || wrhId == null || fieldClientId.getValue() == null) {
    log("Either address, district, payment or client ID is empty. Aborting process.");
    return;
}
int cltId = fieldClientId.getValue();

// check if in stock
try {
    for (ProductInCart productInCart : catalogueList)
        if (productInCart.getQuantity() > 0) {
            ResultSet res = stmt.executeQuery("select quantity from spr where prd_id = "
                    + productInCart.getId() + " and wrh_id = " + wrhId);
            if (!res.next()) {
                log("WRH" + wrhId + " " + productInCart.getName()
                        + " - out of stock. Aborting proccess.");
                return;
            }
            int stocked = res.getInt("quantity");
            if (stocked < productInCart.getQuantity()) {
                log(productInCart.getName() + " - Ordered: " + productInCart.getQuantity()
                        + " In Stock: " + stocked);
                log("Aborting process.");
                return;
            }
        }
} catch (SQLException e) {
    log("Failed to check stocked products. Aborting process. " + e.getSQLState());
    e.printStackTrace();
    return;
}
```

```
                double totalPrice = getTotalPrice();
                textTotalPrice.setText(totalPrice + "");
                if (totalPrice == 0) {
                    log("The shopping cart is empty.");
                    return;
                }
                Timestamp stamp = new Timestamp(System.currentTimeMillis());
                int deliverytime = rnd.nextInt(20) + 25;

                // getting random deliverer
                int dlvId = 0;
                try {
                    ResultSet res = stmt.executeQuery("select id from dlv order by rand() limit 1");
                    res.next();
                    dlvId = res.getInt("id");
                } catch (SQLException e) {
                    log("Couldn't obtain a random deliverer. Aborting process.\n" + e.getSQLState());
                    e.printStackTrace();
                    return;
                }

                // ORDERING
                try {
                    stmt.executeUpdate(
                            "insert into ord (id, cost, address, payment, complete, datetime, deliverytime, dlv_id, clt_id)
"
                                    + "values(" + currentOrderId + "," + totalPrice + ",'" + address + "','" + payment
                                    + "','" + "1" + "','" + stamp + "'," + deliverytime + "," + dlvId + "," + cltId
                                    + ")");

                    log("ORDER " + currentOrderId + "\nClient ID: " + cltId + "\nPrice: " + totalPrice + "\nDistrict: "
```

```
                + wrhId + "\nAddress:" + address + "\nPayment option: " + payment
                + "\nFinished: true\nTimestamp:" + stamp + "\nDelivery time: " + deliverytime
                + " minutes\nDeliverer ID:" + dlvId + "\n");
        } catch (SQLException e) {
            log("Failed to make a new order. Aborting process. " + e.getSQLState());
            e.printStackTrace();
            try {
                conn.rollback();
            } catch (SQLException e1) {
                log("Failed to rollback. " + e1.getSQLState());
                e1.printStackTrace();
            }
            return;
        }
        // #endregion

        // #region ORDERED PRODUCT
        try {
            for (ProductInCart product : catalogueList) {
                if (product.getQuantity() > 0) {
                    stmt.executeUpdate("insert into opr (quantity, ord_id, prd_id) values("
                            + product.getQuantity() + "," + currentOrderId + "," + product.getId() + ")");
                    log("Added (ordID=" + currentOrderId + ", prdID=" + product.getId() + ", q="
                            + product.getQuantity() + ") to Ordered Products");
                }
            }
        } catch (SQLException e) {
            log("Failed to add ordered products. " + e.getSQLState());
            e.printStackTrace();
            try {
                conn.rollback();
```

```
            } catch (SQLException e1) {
                log("Failed to rollback. " + e1.getSQLState());
                e1.printStackTrace();
            }
            return;
        }
        // #endregion

        // #region UNSTOCK
        try {
            for (ProductInCart product : catalogueList) {
                if (product.getQuantity() > 0) {

                    ResultSet res = stmt.executeQuery("select quantity from spr where prd_id = "
                            + product.getId() + " and wrh_id = " + wrhId);
                    res.next();
                    int stocked = res.getInt("quantity");

                    stmt.executeUpdate("update spr set quantity=" + product.getQuantity() + " where prd_id="
                            + product.getId() + " and wrh_id=" + wrhId);
                    log("WRH" + wrhId + "PRD" + product.getId() + "(" + product.getName() + ")" + " - set to "
                            + (stocked - product.getQuantity()));
                }
            }
        } catch (SQLException e) {
            log("Failed to unstock products. Aborting proccess. " + e.getSQLState());
            e.printStackTrace();
            return;
        }
        // #endregion
```

```
            currentOrderId++;
            try {
                conn.commit();
            } catch (SQLException e) {
                log("Failed comitting changes. "+e.getSQLState());
                e.printStackTrace();
            }
        }
    });

    boxOrderPart.getChildren().addAll(textTotalPrice, btnRefreshPrice, textAddress, boxAddress, textClient,
            fieldClientId, textPaymentOption, fieldPayment, btnMakeOrder);
    // #endregion

    // #region admin
    Button btnRefresh = new Button("Refresh info");
    btnRefresh.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            try {
                refreshDistricts();
                refreshClients();
            } catch (SQLException e) {
                log("Failed to obtain Districts/Clients. "+e.getSQLState());
                e.printStackTrace();
            }
        }
    });

    //setting up the log field. it's not editable. it adapts to app's width.
    fieldLog = new TextArea("LOG");
```

```
        fieldLog.editableProperty().set(false);
        fieldLog.setPrefSize(390, 370);
        scene.widthProperty().addListener((obs, oldValue, newValue) -> fieldLog.setPrefWidth(newValue.doubleValue()-840));

        Button btnClients = new Button("Show clients");
        btnClients.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                try {
                    log("CLIENTS:");
                    ResultSet res = stmt.executeQuery("select * from clt");
                    while (res.next()) {
                        log("ID: " + res.getInt("id") + "\nName: " + res.getString("name") + "\nPhone: "
                                + res.getString("phone"));
                    }
                } catch (SQLException e) {
                    log("Failed to obtain clients. "+e.getSQLState());
                    e.printStackTrace();
                }
            }
        });

        boxAdminPart.getChildren().addAll(btnClients, fieldLog);
        // #endregion

        root.getChildren().addAll(border1, border2, boxCataloguePart, boxOrderPart, boxAdminPart);
    }

    /**
     * узнаем номер последнего заказа.
     * когда будем создавать новый заказ, id будет увеличен
```

```
     * @return номер последнего заказа
     * @throws SQLException
     */
    private static int getLastOrderId() throws SQLException {
        ResultSet res = stmt.executeQuery("select max(id) from ord");
        res.next();
        return res.getInt("max(id)");
    }

    /**
     * оставить сообщение в окне лога
     * @param msg сообщение
     */
    public static void log(String msg) {
        fieldLog.setText(fieldLog.getText() + "\n" + msg);
    }

    /**
     * refresh observable list for combobox
     * @throws SQLException
     */
    private static void refreshDistricts() throws SQLException {
        districts.clear();
        ResultSet res = stmt.executeQuery("select id from wrh");
        while (res.next()) {
            districts.add(res.getString("id"));
        }
    }

    /**
     * refresh observable list for combobox
```

```
     * @throws SQLException
     */
    private static void refreshClients() throws SQLException {
        clients.clear();
        ResultSet res = stmt.executeQuery("select id from clt");
        while (res.next()) {
            clients.add(res.getInt("id"));
        }
        res.close(); stmt.close(); stmt = conn.createStatement();
    }

    /**
     * @return total price
     */
    private static double getTotalPrice() {
        double total = 0;
        for (ProductInCart productInCart : catalogueList) {
            if (productInCart.getQuantity() > 0) {
                total += productInCart.getTotalPrice();
            }
        }
        return total;
    }

    /**
     * спрашиваем у бд, какие есть товары. заполняем каталог.
     */
    private static void initCatalogue() {
        try {
            ResultSet res = stmt.executeQuery("select * from prd");
            while (res.next()) {
```

```
                ProductInCart tmp = new ProductInCart(res.getInt("id"), res.getString("name"),
                        res.getString("description"), res.getString("category"), res.getDouble("price"));
                catalogueList.add(tmp);
                boxCatalogue.getChildren().add(tmp.getBox());
            }
        } catch (SQLException e) {
            log(e.getSQLState());
            e.printStackTrace();
        }
    }

    @Override
    public void start(Stage primaryStage) {
        initMenu();
        initSql();
        initCatalogue();
        try {
            refreshClients();
            refreshDistricts();
        } catch (SQLException e) {
            log(e.getSQLState());
            e.printStackTrace();
        }
        try {
            currentOrderId = getLastOrderId()+1;
        } catch (SQLException e) {
            currentOrderId = 0;
            log("Couldn't get last order's ID!\n" + e.getSQLState());
            e.printStackTrace();
        }
```

```
        primaryStage.setTitle("Polydelivery");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```