KULINA TEST

Name : Dhiya Ulhaq Rifqi

Basic concepts

1. Coding

A. Smell code OOP

Temporary field

Temporary field is one of the smell code in OOP, Temporary field is a variable that isn't really needed, or used in a very limited situation.

For example:

```
function sum (a, b) {
  var total = a + b;
  return total
}
```

Variable total is a temporary field.

Solution

Extract temporary fields be extracted into their own SI, or replace them by passing them as parameters to methods.

Refactoring from code before.

```
function sum (a, b) {
  return a + b;
}
```

B. Dependency injection

Dependency injection is a technique in which an object receives other objects that it depends on. Why used this method?

Because dependency injection does not require any change in code behavior it can be applied to legacy code as a refactoring. The result is clients that are more independent and that are easier to unit test, and dependency injection allows a client the flexibility of being configurable.

2. Rest API

POST and GET handle request

GET

- Do
  - when you want to get something from server.
- Don’t
  - when you want to change something in server (create, update, or delete) or send private information to server.

POST

- Do
  - When you want to change something in server (create, update, or delete).
- Don’t
  - When you just want to see something from server and idempotent requests (operation is one in which the result will be the same no matter how many times you request it).

Basic coding

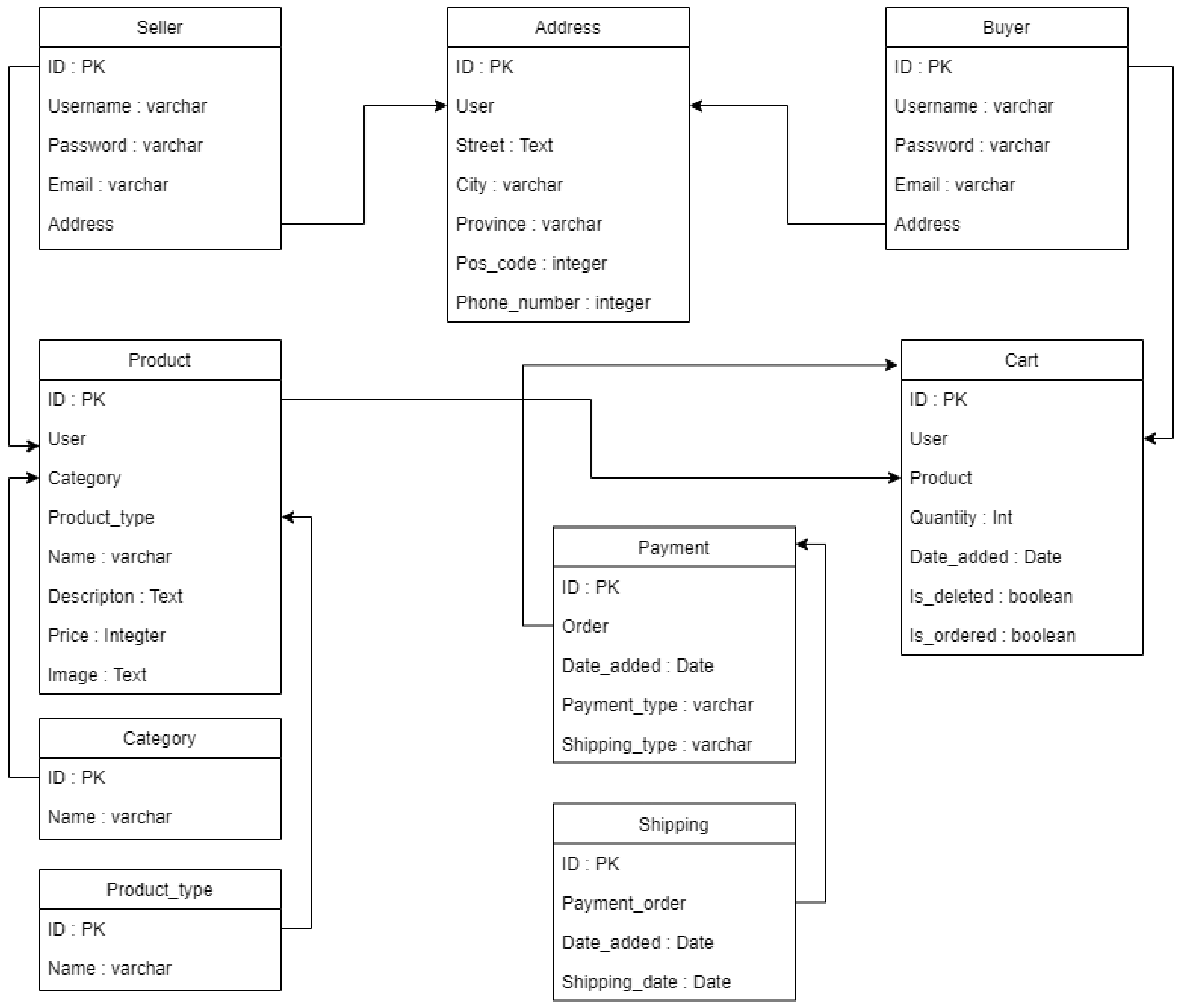

Figure1. Diagram

User

- User can register
- User can input address (Address table, figure 1)
- User the products to be purchased with subscription and/or one-time purchase (Product_type in Product Table, Figure1)
- User can pay the bill
- User can skip delivery.

- User cancel the order (if the user has not paid the order, user can cancel the order, cart table in figure1. “Is_deleted” change to “true” if user cancel order, default is “false”)

Supplier

- Supplier can register as seller
- Supplier can create the store and complete the address (from figure 1, seller can sell product and fill the address)
- Supplier can create products that can be purchased either daily or one-time purchase (seller can choose product type in product table from figure 1)
- Supplier can determine the price of each product.
- Supplier can determine the selling area. (from the address filled in by the seller)

Additional

Case 1.

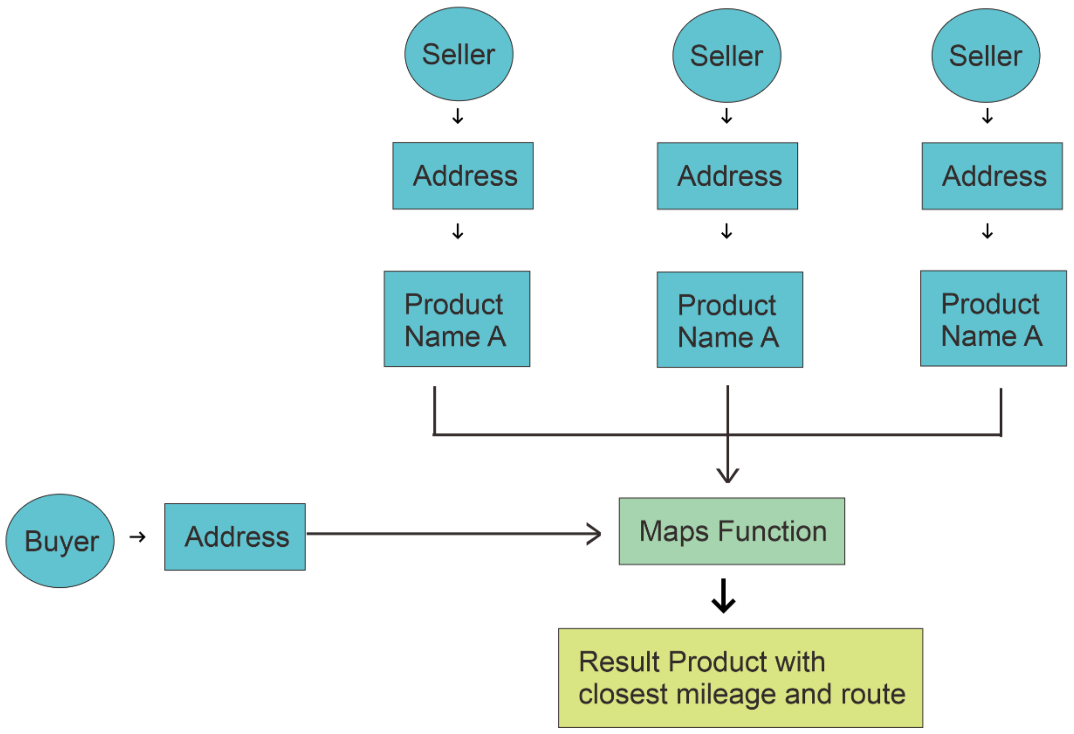

Figure 2. Case 1 Flow Chart

If we want to know what is product with closet mileage and route from seller and buyer, we can use address buyer and seller (address table in figure 1), we create like maps-function, which calls up the map API, and compares the distances from each buyer to the seller on the same product. In the end, it shows the product with closet mileage and route from seller and buyer.

Case 2.

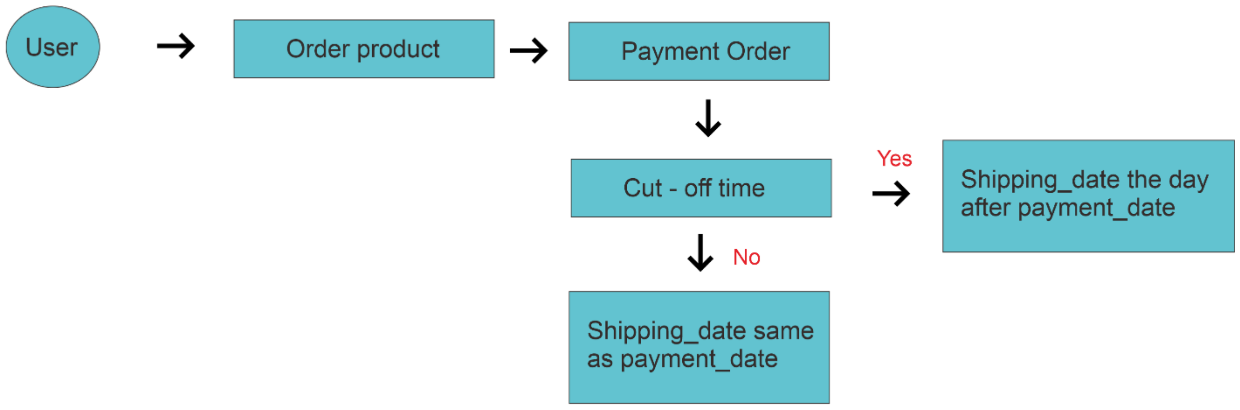
User
Order product
Payment Order
Cut - off time
Yes
Shipping_date the day after payment_date
No
Shipping_date same as payment_date

Figure 3. Case 2 Flow Chart.

We can create a cut-off function that check whether an order is in cut-off time or not, and fills in the shipping_date table from shipping table in Figure 1.

Algorithm

In main.go files or my github repo (https://github.com/Dhiiyaur/kulina-test/blob/main/main.go)

Github repo:
https://github.com/Dhiiyaur/kulina-test