# Bảng báo cáo đồ án

# WEBSITE KINH DOANH GÀ RÁN

**Nhóm thực hiện đề tài**

Dương Thiên Khôi - 18DH110901

HCMC, 7/01/2022

# Mục lục

# 1. Giới thiệu

## 1.1. Mục tiêu

Bản phân tích và thiết kế này cung cấp bản mô tả chi tiết về hệ thống "website kinh doanh gà rán", bản phân tích các chức năng chủ yếu và bản thiết kế các chức năng thiết kế chính yếu của hệ thống được xây dựng.

## 1.2. Các định nghĩa và từ viết tắt trong báo cáo

| # | Thuật ngữ/ Từ viết tắt | Mô tả |
|---|---|---|
| 1 | SRS | Software Requirement Specification |
| 2 | BR | Business Rule |
| 3 | SC | Screen |
| 4 | UC | Use Case |
| 5 | CRUD | Create/Read/Update/Delete a record in database |
| 6 | QL | Quản lý |

**Bảng 1: Các từ viết tắt và thuật ngữ**

## 1.3. Danh sách tham khảo trong đồ án

| # | Tên | Mô tả |
|---|---|---|
| | | |

**Bảng 2: Các tham khảo**

# 2. Phân tích yêu cầu

## 2.1. Mô tả dự án

Dự án website bán gà rán bao gồm hai actor sử dụng chính là khách hàng và nhân viên, khách hàng thể sử dụng các chức năng mua sắm cơ bản và nhân viên có thể sử dụng các chức năng về quản lý.

## 2.2. Actors và Use Cases

### 2.2.1. Các Use Case Diagram

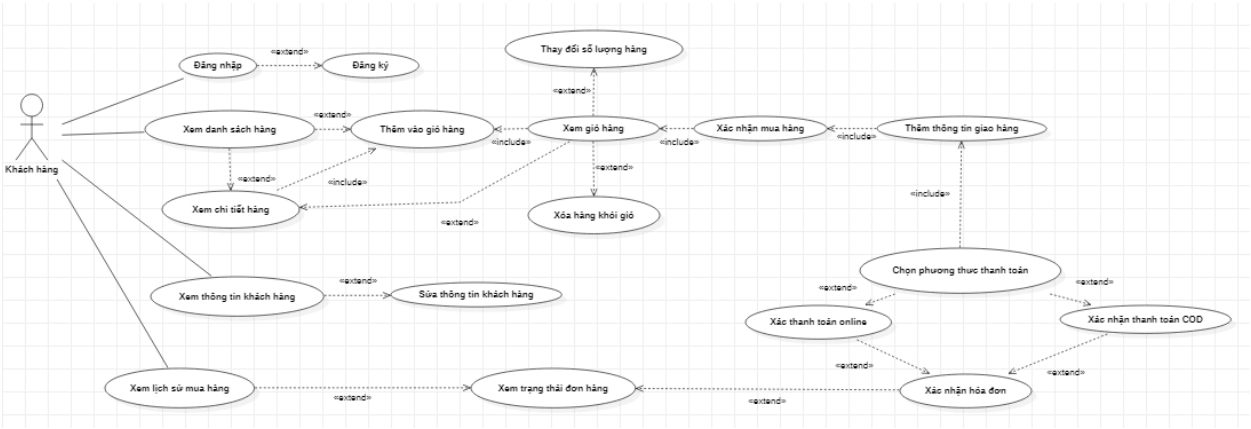



**Diagram 01 – Khách hàng**

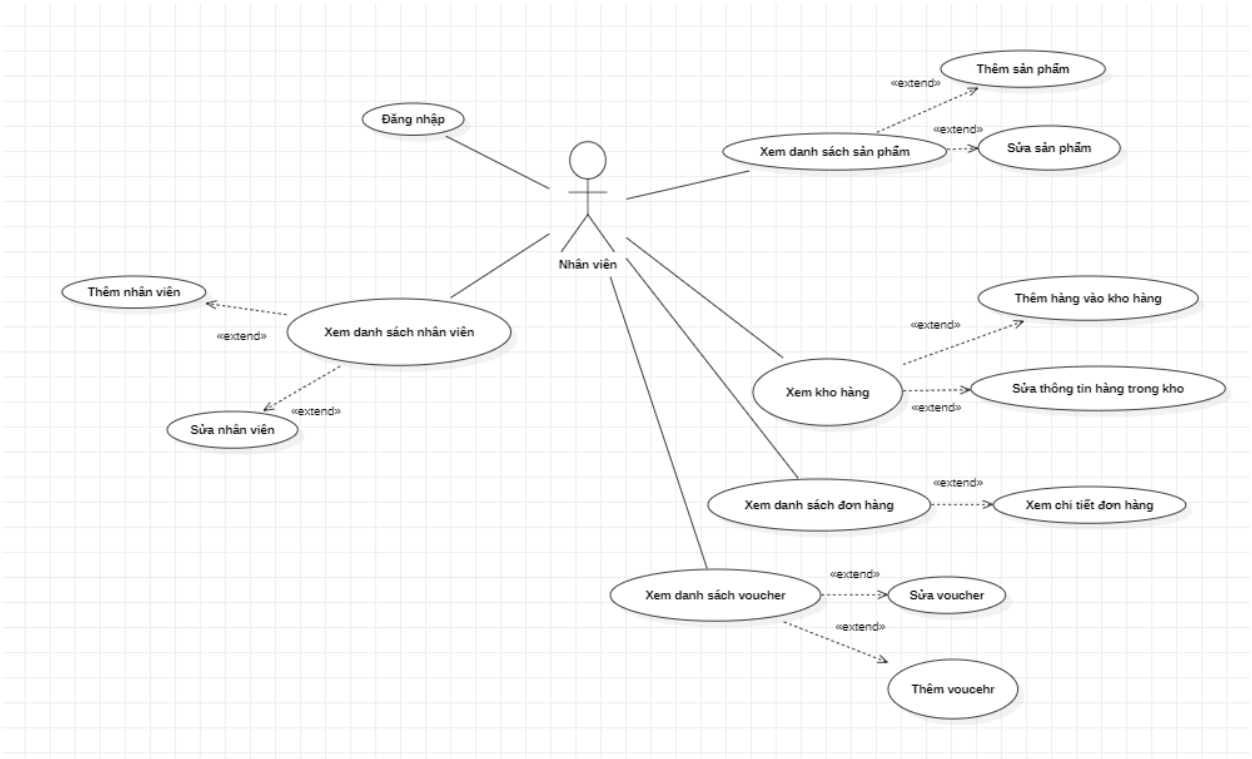



**Diagram 02 – Nhân viên**

### 2.2.2. Mô tả Actors

| # | Tên Actor | Mô tả |
|---|---|---|
| 1 | Nhân viên | Quản lý hệ thống, kiểm tra doanh thu, quản lý nhân viên ,đơn hàng |
| 2 | Khách hàng | Thực hiện các hành vi mua hàng cơ bản |

### 2.2.3. Mô tả Use Cases

| # | Code | Name | Brief Description |
|---|---|---|---|
| 1 | UC01 | Đăng nhập | Cho phép actor đăng nhập vào hệ thống |
| 2 | UC02 | Đăng ký | Cho phép actor đăng ký trên hệ thống |
| 3 | UC03 | Xem danh sách hàng | Cho phép actor view toàn bộ hàng trong hiện có |
| 4 | UC04 | Xem chi tiết | Cho phép actor xem chi tiết 1 sản phẩm |
| 5 | UC05 | Thêm vào giỏ hàng | Cho phép actor thêm hàng vào giỏ |
| 6 | UC06 | Xem giỏ hàng | Cho phép actor xem những sản phẩm đã thêm vào giỏ |
| 7 | UC07 | Thay đổi số lượng hàng | Cho phép actor thay đổi số lượng hàng trong giỏ |
| 8 | UC08 | Xóa khỏi giỏ | Cho phép actor xóa hàng trong giỏ |
| 9 | UC09 | Xác nhận mua hàng | Cho phép actor xác nhận hàng trong giỏ, bắt đầu các bước thanh toán |
| 10 | UC10 | Thêm thông tin giao hàng | Cho phép actor thêm thông tin giao hàng |
| 11 | UC11 | Chọn phương thức thanh toán | Cho phép actor kiểm tra lại tổng tiền, sau đó chọn phương thức thanh toán |
| 12 | UC12 | Xác nhận thanh toán online | Cho actor thanh toán qua paypal |
| 13 | UC1E | Xác nhận thanh toán COD | Ghi nhận actor sẽ tanh toán cod |
| 14 | UC14 | Xác nhận hóa đơn | Cho phép actor xác nhận thông tin hoá đơn |
| 15 | UC15 | Xem trạng thái đơn hàng | Cho phép actor xem trạng thái đơn hàng |
| 16 | UC16 | Xem thông tin khách hàng | Cho phép actor xem thông tin bản thân |
| 17 | UC17 | Sửa thông tin khách hàng | Cho phép actor sửa thông tin bản thân |
| 18 | UC18 | Xem lịch sử mua hàng | Cho phép actor xem lịch sử mua hàng |
| 19 | UC19 | Xem danh sách sản phẩm | Cho phép actor xem danh sách các sản phẩm hệ thống |
| 20 | UC20 | Thêm sản phẩm | Cho phép actor thêm sản phẩm |
| 21 | UC21 | Sửa sản phẩm | Cho phép actor sửa sản phẩm |
| 22 | UC22 | Xem danh sách kho hàng | Cho phép actor xem danh sách hàng trong kho |
| 23 | UC23 | Thêm hàng vào kho hàng | Cho phép actor thêm hàng vào kho |
| 24 | UC24 | Sửa thông tin hàng trong kho | Cho phép actor sửa hàng trong kho |

| | | | |
|---|---|---|---|
| 25 | UC25 | Xem danh sách voucher | Cho phép actor xem danh sách các voucher hệ thống |
| 26 | UC26 | Thêm voucher | Cho phép actor thêm voucher |
| 27 | UC27 | Sửa voucher | Cho phép actor sửa voucher |
| 28 | UC28 | Xem danh sách nhân viên | Cho phép actor xem danh sách các nhân viên hệ thống |
| 29 | UC29 | Thêm nhân viên | Cho phép actor thêm nhân viên |
| 30 | UC30 | Sửa nhân viên | Cho phép actor sửa nhân viên |
| 31 | UC31 | Xem danh sách đơn hàng | Cho phép actor xem danh sách các đơn hàng hệ thống |
| 32 | UC32 | Xem chi tiết đơn hàng | Cho phép actor xem chi tiết đơn hàng |

**Table 3: Use Case List**

# 3. Yêu cầu chức năng

(Viết description đầy đủ các Use cases. Đối với các usecases quan trọng của hệ thống, sẽ vẽ kèm theo activity diagram hoặc sequences diagram (tối thiểu 5 activity/sequence diagram)

## 3.1. UC01:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Đăng nhập | **Code** | UC01 |
| **Description** | Cho phép actor đăng nhập vào hệ thống | | |
| **Actor** | Khách hàng/ Nhân viên | **Trigger** | Actor bấm nút login |
| **Pre-condition** | | | |
| **Post condition** | Chuyển tới trang default với role tương ứng | | |

**System Message**

| | |
|---|---|
| MS01 | "Tên Đăng Nhập và/hoặc Mật Khẩu của bạn không chính xác. Vui lòng kiểm tra và thử lại."<br>Message thông báo khi actor nhập sai tên đăng nhập/mật khẩu |

## 3.2. UC02:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Đăng ký | **Code** | UC02 |
| **Description** | Cho phép actor đăng ký trên hệ thống | | |
| **Actor** | Khách hàng | **Trigger** | Nhấn nút đăng ký |
| **Pre-condition** | | | |
| **Post condition** | Đăng ký thành công | | |

## 3.3. UC03:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Đăng ký | **Code** | UC03 |
| **Description** | Xem danh sách hàng | | |
| **Actor** | Khách hàng | **Trigger** | Nhấn nút xem danh sách hàng |
| **Pre-condition** | Đã đăng nhập role khách hàng | | |
| **Post condition** | Xem được danh sách hàng | | |

## 3.4. UC04:

**Use Case Description**

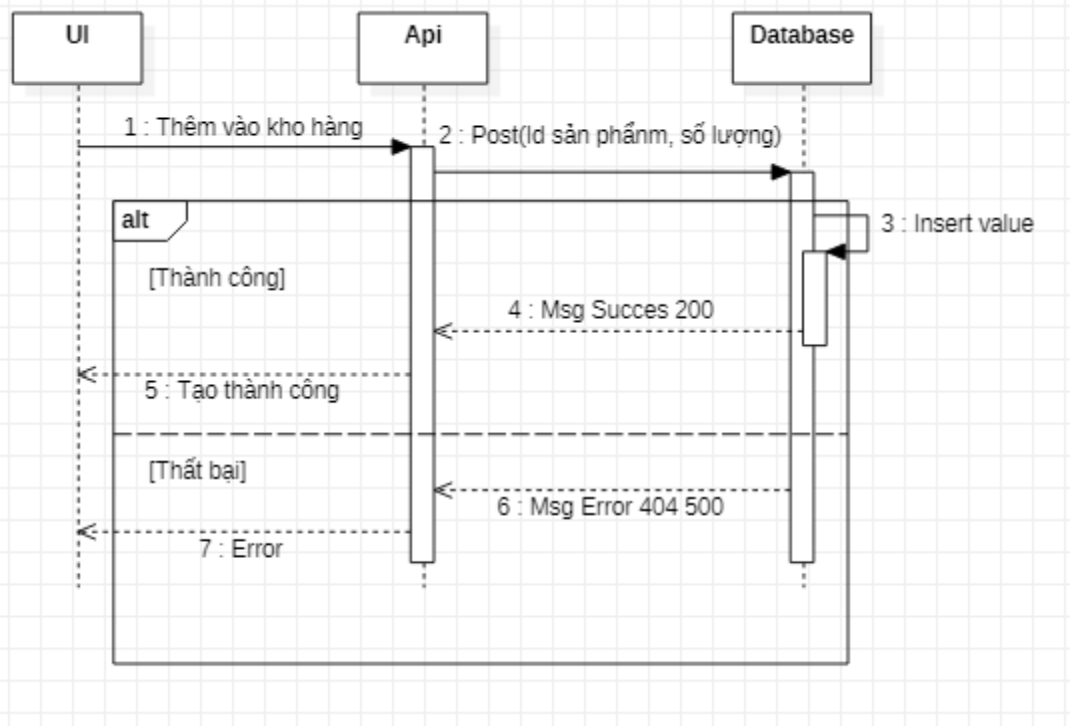
| **Name** | Xem chi tiết | **Code** | UC04 |
|---|---|---|---|
| **Description** | Cho phép actor xem chi tiết 1 sản phẩm | | |
| **Actor** | Khách hàng | **Trigger** | Nhấn vào 1 mặt hàng |
| **Pre-condition** | Đã đăng nhập role khách hàng | | |
| **Post condition** | Xem được chi tiết hàng | | |

## 3.5. UC05:

**Use Case Description**

| **Name** | Thêm vào giỏ hàng | **Code** | UC05 |
|---|---|---|---|
| **Description** | Cho phép actor thêm hàng vào giỏ | | |
| **Actor** | Khách hàng | **Trigger** | Nhấn vào nút thêm vào giỏ |
| **Pre-condition** | Đã đăng nhập role khách hàng | | |
| **Post condition** | Hàng được thêm vào giỏ | | |

**Sequence diagram**




## 3.6. UC06:

**Use Case Description**

| **Name** | Xem giỏ hàng | **Code** | UC06 |
|---|---|---|---|
| **Description** | Cho phép actor xem những sản phẩm đã thêm vào giỏ | | |
| **Actor** | Khách hàng | **Trigger** | Nhấn vào nút giỏ hàng |
| **Pre-condition** | Đã đăng nhập role khách hàng | | |

| | |
|---|---|
| **Post condition** | Xem được danh sách hàng trong giỏ |

## 3.7. UC07:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Thay đổi số lượng giỏ | **Code** | UC07 |
| **Description** | Cho phép actor thay đổi số lượng hàng trong giỏ | | |
| **Actor** | Khách hàng | **Trigger** | Nhấn vào nút cộng trừ |
| **Pre-condition** | Đã đăng nhập role khách hàng | | |
| **Post condition** | Thay đổi số lượng trong giỏ | | |

## 3.8. UC08:

**Use Case Description**

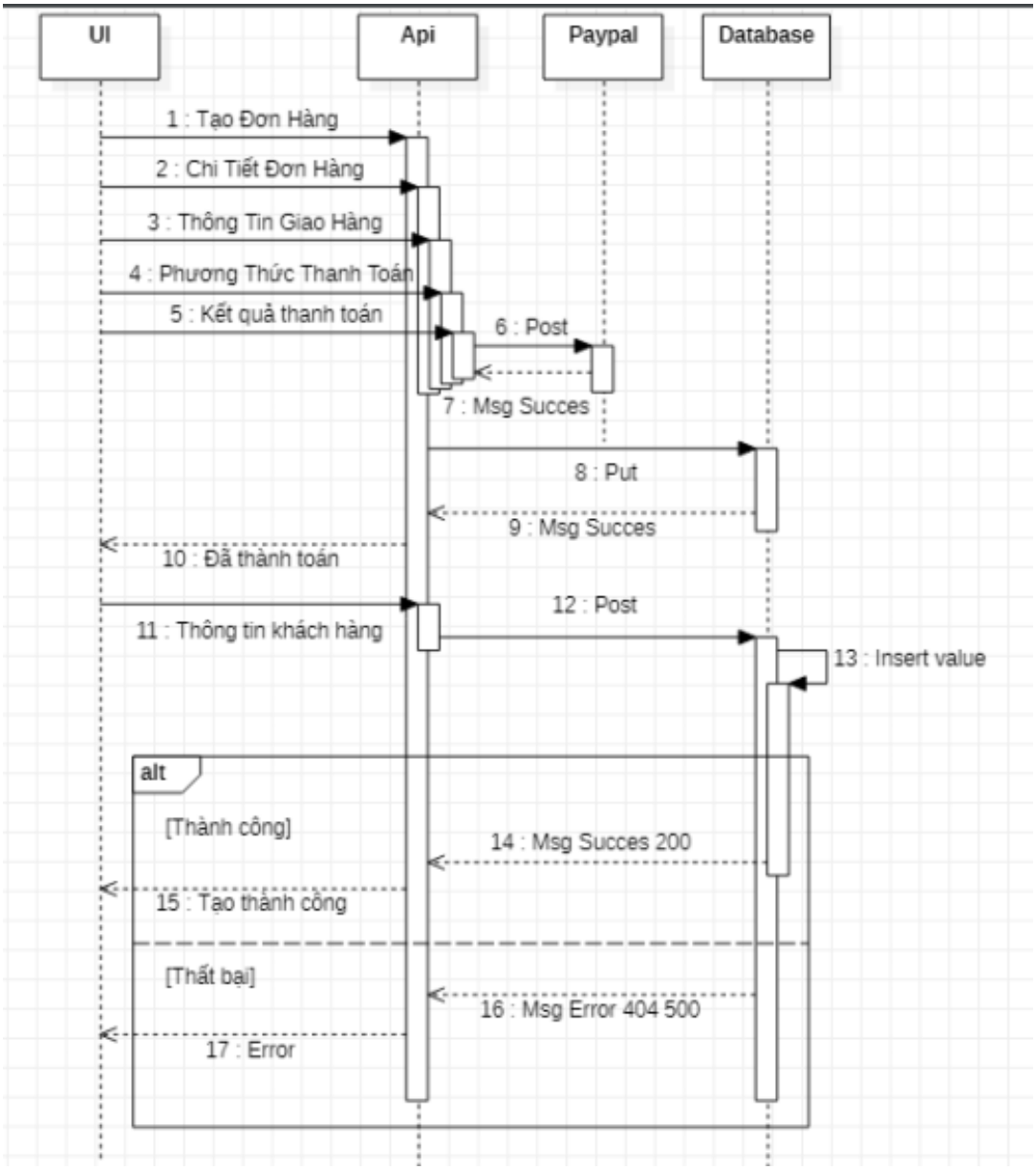
| | | | |
|---|---|---|---|
| **Name** | Xem giỏ hàng | **Code** | UC08 |
| **Description** | Cho phép actor xóa hàng trong giỏ | | |
| **Actor** | Khách hàng | **Trigger** | Nhấn vào nút xóa |
| **Pre-condition** | Đã đăng nhập role khách hàng | | |
| **Post condition** | Xóa khỏi giỏ | | |

## 3.9. UC09:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Xác nhận mua hàng | **Code** | UC09 |
| **Description** | Cho phép actor xác nhận hàng trong giỏ, bắt đầu các bước thanh toán | | |
| **Actor** | Khách hàng | **Trigger** | Nhấn vào nút mua hàng |
| **Pre-condition** | Đã đăng nhập role khách hàng | | |
| **Post condition** | Bắt đầu quá trình thanh toán | | |

**Sequence diagram**




## 3.10. UC010:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Thêm thông tin giao hàng | **Code** | UC10 |
| **Description** | Cho phép actor thêm thông tin giao hàng | | |
| **Actor** | Khách hàng | **Trigger** | Nhập thông tin sao đó nhấn xác nhận |
| **Pre-condition** | Đã đăng nhập role khách hàng | | |
| **Post condition** | Thêm thông tin giao hàng | | |

## 3.11. UC011:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Chọn phương thức thanh toán | **Code** | UC11 |
| **Description** | Cho phép actor kiểm tra lại tổng tiền, sau đó chọn phương thức thanh toán | | |
| **Actor** | Khách hàng | **Trigger** | Chọn 1 trong hai hình thức tương ứng |
| **Pre-condition** | Đã đăng nhập role khách hàng | | |
| **Post condition** | Chuyển đến màn hình tương ứng | | |

## 3.12. UC012:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Xác nhận thanh toán online | **Code** | UC12 |
| **Description** | Cho actor thanh toán qua paypal | | |
| **Actor** | Khách hàng | **Trigger** | Thực hiện các thao tác paypal |
| **Pre-condition** | Đã đăng nhập role khách hàng | | |
| **Post condition** | Chuyển sang xác nhận hóa đơn | | |

## 3.13. UC013:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Xác nhận thanh toán COD | **Code** | UC13 |
| **Description** | Ghi nhận actor sẽ tanh toán cod | | |
| **Actor** | Khách hàng | **Trigger** | Nhấn nút xác nhận |
| **Pre-condition** | Đã đăng nhập role khách hàng | | |
| **Post condition** | Chuyển sang xác nhận hóa đơn | | |

## 3.14. UC014:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Xác nhận hóa đơn | **Code** | UC14 |
| **Description** | Cho phép actor xác nhận thông tin hoá đơn | | |
| **Actor** | Khách hàng | **Trigger** | Nhấn nút xác nhận |
| **Pre-condition** | Đã đăng nhập role khách hàng | | |
| **Post condition** | Lưu hóa đơn lên hệ thống | | |

## 3.15. UC015:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Xem trạng thái đơn hàng | **Code** | UC15 |
| **Description** | Cho phép actor xem trạng thái đơn hàng | | |

| | | | |
|---|---|---|---|
| **Actor** | Khách hàng | **Trigger** | Nhấn nút trạng thái |
| **Pre-condition** | Đã đăng nhập role khách hàng | | |
| **Post condition** | Xem dược trạng thái đơn hàng | | |

## 3.16. UC016:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Xem thông tin khách hàng | **Code** | UC16 |
| **Description** | Cho phép actor xem thông tin bản thân | | |
| **Actor** | Khách hàng | **Trigger** | Nhấn nút hồ sơ |
| **Pre-condition** | Đã đăng nhập role khách hàng | | |
| **Post condition** | xem thông tin bản thân | | |

## 3.17. UC017:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Sửa thông tin khách hàng | **Code** | UC17 |
| **Description** | Cho phép actor sửa thông tin bản thân | | |
| **Actor** | Khách hàng | **Trigger** | Nhấn nút sửa |
| **Pre-condition** | Đã đăng nhập role khách hàng | | |
| **Post condition** | Sửa thông tin thành công | | |

## 3.18. UC018:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Xem lịch sử mua hàng | **Code** | UC18 |
| **Description** | Cho phép actor xem lịch sử mua hàng | | |
| **Actor** | Khách hàng | **Trigger** | Nhấn nút lịch sửa mua |
| **Pre-condition** | Đã đăng nhập role khách hàng | | |
| **Post condition** | Xem được lịch sử mua hàng | | |

## 3.19. UC019:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Xem danh sách sản phẩm | **Code** | UC19 |
| **Description** | Cho phép actor xem danh sách các sản phẩm hệ thống | | |
| **Actor** | Nhân viên | **Trigger** | Nhấn nút danh sách sản phẩm |
| **Pre-condition** | Đã đăng nhập role nhân viên | | |
| **Post condition** | Xem được danh sách sản phẩm | | |

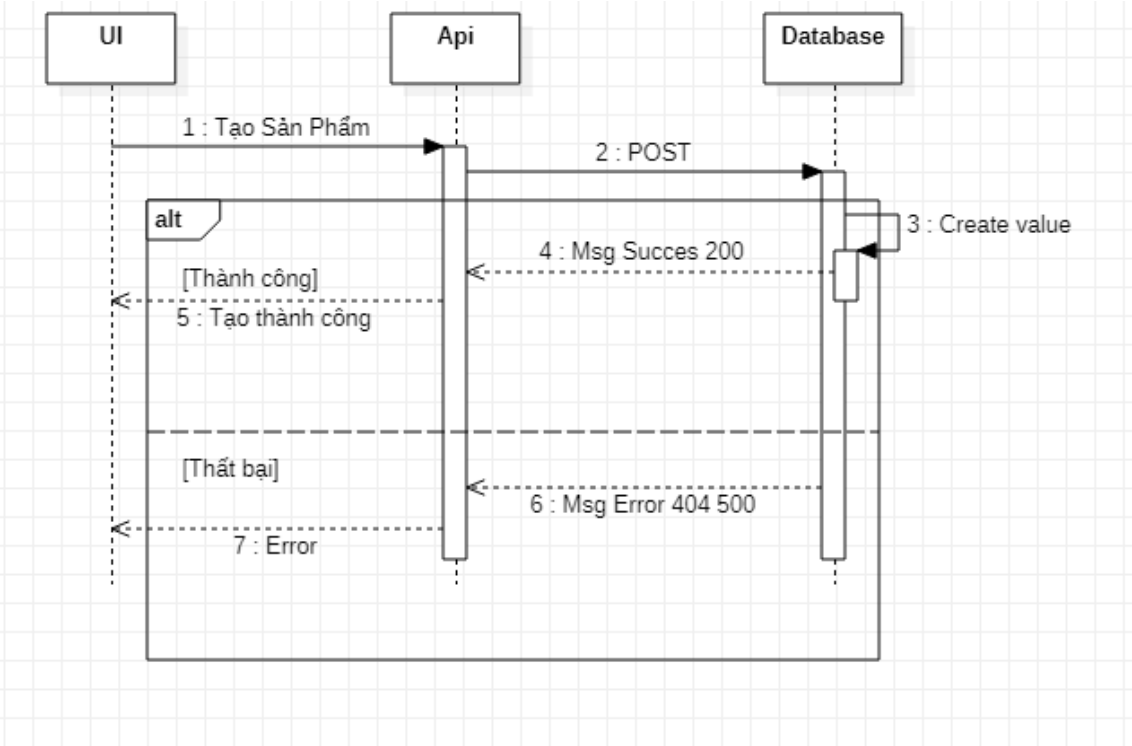
## 3.20. UC020:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Thêm sản phẩm | **Code** | UC20 |

| **Description** | Cho phép actor thêm sản phẩm | | |
|---|---|---|---|
| **Actor** | Nhân viên | **Trigger** | Nhấn nút thêm |
| **Pre-condition** | Đã đăng nhập role nhân viên | | |
| **Post condition** | Thêm thành công sản phẩm | | |

**Sequence diagram**

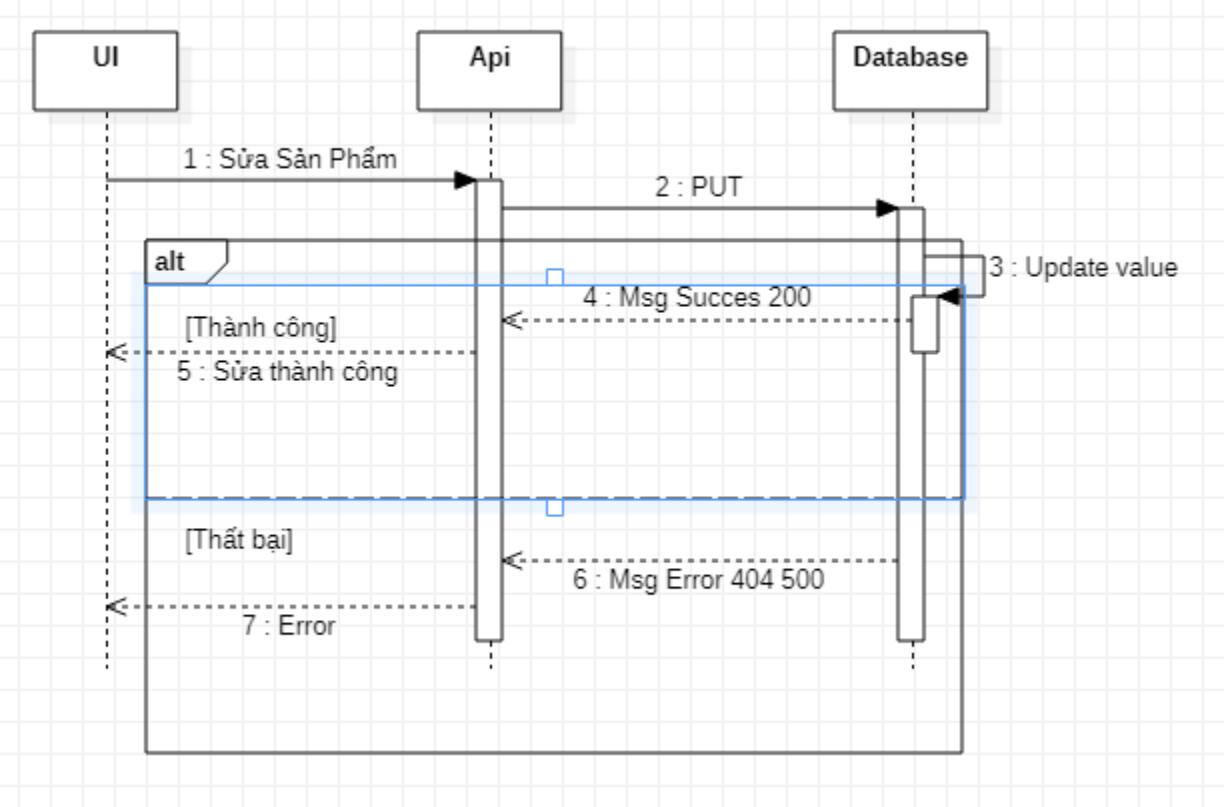



## 3.21. UC021:

**Use Case Description**

| **Name** | Sửa sản phẩm | **Code** | UC21 |
|---|---|---|---|
| **Description** | Cho phép actor sửa sản phẩm | | |
| **Actor** | Nhân viên | **Trigger** | Nhấn nút sửa |
| **Pre-condition** | Đã đăng nhập role nhân viên | | |
| **Post condition** | Sửa thành công sản phẩm | | |

**Sequence diagram**




## 3.22. UC22:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Xem danh sách kho hàng | **Code** | UC22 |
| **Description** | Cho phép actor xem danh sách các sản phẩm kho hàng | | |
| **Actor** | Nhân viên | **Trigger** | Nhấn nút danh sách sản phẩm |
| **Pre-condition** | Đã đăng nhập role nhân viên | | |
| **Post condition** | Xem được danh sách kho hàng | | |

## 3.23. UC23:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Thêm hàng trong kho | **Code** | UC23 |
| **Description** | Cho phép actor thêm hàng vào kho | | |
| **Actor** | Nhân viên | **Trigger** | Nhấn nút thêm |
| **Pre-condition** | Đã đăng nhập role nhân viên | | |
| **Post condition** | Thêm thành công hàng | | |

## 3.24. UC24:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Sửa hàng trong kho | **Code** | UC24 |

| | | | |
|---|---|---|---|
| **Description** | Cho phép actor sửa hàng | | |
| **Actor** | Nhân viên | **Trigger** | Nhấn nút sửa |
| **Pre-condition** | Đã đăng nhập role nhân viên | | |
| **Post condition** | Sửa thành công hàng | | |

## 3.25. UC25:

**Use Case Description**

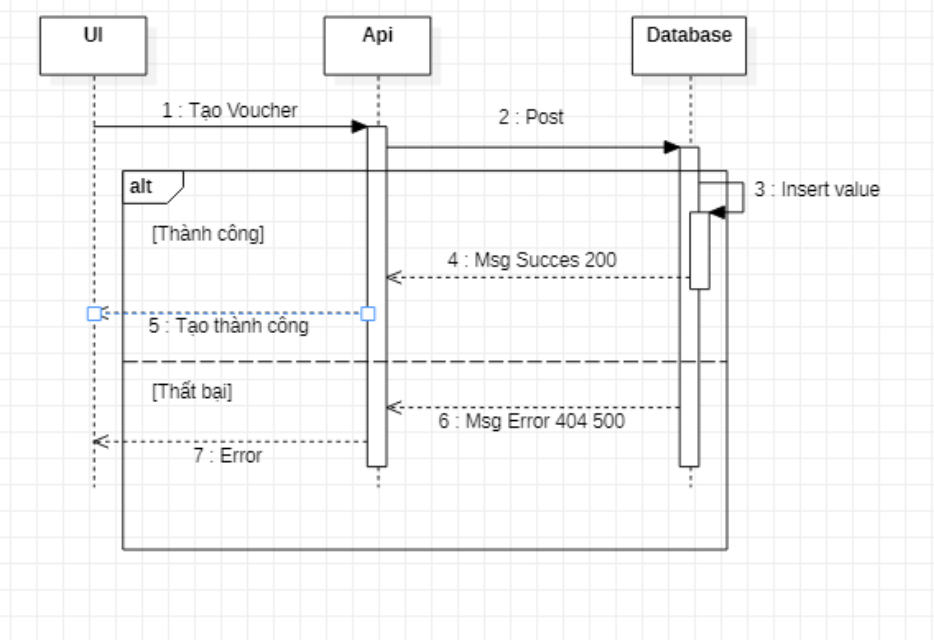
| | | | |
|---|---|---|---|
| **Name** | Xem danh sách voucher | **Code** | UC25 |
| **Description** | Cho phép actor xem danh sách các sản phẩm voucher | | |
| **Actor** | Nhân viên | **Trigger** | Nhấn nút danh sách sản phẩm |
| **Pre-condition** | Đã đăng nhập role nhân viên | | |
| **Post condition** | Xem được danh sách voucher | | |

## 3.26. UC26:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Thêm voucher | **Code** | UC26 |
| **Description** | Cho phép actor thêm voucher | | |
| **Actor** | Nhân viên | **Trigger** | Nhấn nút thêm |
| **Pre-condition** | Đã đăng nhập role nhân viên | | |
| **Post condition** | Thêm thành công hàng | | |

**Sequence diagram**




## 3.27. UC27:

**Use Case Description**

| **Name** | Sửa hàng voucher | **Code** | UC27 |
|---|---|---|---|
| **Description** | Cho phép actor sửa voucher | | |
| **Actor** | Nhân viên | **Trigger** | Nhấn nút sửa |
| **Pre-condition** | Đã đăng nhập role nhân viên | | |
| **Post condition** | Sửa thành công voucher | | |

## 3.28. UC28:

**Use Case Description**

| **Name** | Xem danh sách khách hàng | **Code** | UC28 |
|---|---|---|---|
| **Description** | Cho phép actor xem danh sách khách hàng | | |
| **Actor** | Nhân viên | **Trigger** | Nhấn nút danh sách sản phẩm |
| **Pre-condition** | Đã đăng nhập role nhân viên | | |
| **Post condition** | Xem được danh sách khách hàng | | |

## 3.29. UC29:

**Use Case Description**

| **Name** | Thêm hàng khách hàng | **Code** | UC29 |
|---|---|---|---|
| **Description** | Cho phép actor thêm khách hàng | | |
| **Actor** | Nhân viên | **Trigger** | Nhấn nút thêm |
| **Pre-condition** | Đã đăng nhập role nhân viên | | |
| **Post condition** | Thêm thành công khách hàng | | |

## 3.30. UC30:

**Use Case Description**

| **Name** | Sửa hàng khách hàng | **Code** | UC30 |
|---|---|---|---|
| **Description** | Cho phép actor sửa khách hàng | | |
| **Actor** | Nhân viên | **Trigger** | Nhấn nút sửa |
| **Pre-condition** | Đã đăng nhập role nhân viên | | |
| **Post condition** | Sửa thành công khách hàng | | |

## 3.31. UC31:

**Use Case Description**

| **Name** | Xem danh sách đơn hàng | **Code** | UC31 |
|---|---|---|---|
| **Description** | Cho phép actor xem danh sách đơn hàng | | |
| **Actor** | Nhân viên | **Trigger** | Nhấn nút danh sách sản phẩm |
| **Pre-condition** | Đã đăng nhập role nhân viên | | |
| **Post condition** | Xem được danh sách đơn hàng | | |

## 3.32. UC32:

**Use Case Description**

| | | | |
|---|---|---|---|
| **Name** | Xem chi tiết đơn hàng | **Code** | UC31 |
| **Description** | Cho phép actor xem chi tiết đơn hàng | | |
| **Actor** | Nhân viên | **Trigger** | Nhấn nút chi tiết sản phẩm |
| **Pre-condition** | Đã đăng nhập role nhân viên | | |
| **Post condition** | Xem được chi tiết đơn hàng | | |

# 4. Thiết kế hệ thống

## 4.1. Kiến trúc hệ thống

**Kiến trúc chính:**

Giao diện: React

Api: TypeScript

CSDL: mysql

**Cấu hình hệ thống:**

Server: https://servertmdt.herokuapp.com/

## 4.2. Thiết kế cơ sở dữ liệu:

DonHang.js

```
import database from 'mongoose'

const donhangSchema = new database.Schema({
    ChiTietDonHang: [
        {
            ten: {type:String, required: true},
            sl: {type:String, required: true},
            hinhanh: {type:String, required: true},
            gia: {type:String, required: true},
            sanpham: {
                type: database.Schema.Types.ObjectId,
                ref: 'Sanpham',
                required: true,
            },
        },
    ],
    ThongTinGiaoHang: {
        hoten: {type:String, required: true},
        diachi: {type:String, required: true},
        phuong: {type:String, required: true},
        quan: {type:String, required: true},
        sodienthoai: {type:String, required: true},
        lat: Number,
        lng: Number,
    },
    PhuongThucThanhToan: {type:String, required: true},
    paymentResult: {
        id: String,
        status: String,
        update_time: String,
        email_address: String,
    },
    itemsPrice: {type:Number, required: true},
    shippingPrice: {type:Number, required: true},
```

```
    taxPrice: {type:Number, required: true},
    totalPrice: {type:Number, required: true},
    user: {type: database.Schema.Types.ObjectId, ref: 'User', required: true},
    isPaid: {type:Boolean, default: false},
    paidAt: {type: Date},
    isDelivered: {type:Boolean, default: false},
    deliveredAt: {type: Date},

},
{
    timestamps: true,
}
);

const DonHang = database.model('DonHang', donhangSchema);
export default DonHang;
```

KhoHang.js

```
import mongoose from 'mongoose';
import Sanpham from './SanPham.js';

const khohangSchema= mongoose.Schema({
    sanpham:{Sanpham},
    sanpham: {
        type: mongoose.Schema.Types.ObjectId,
        ref: 'Sanpham',
        required: true,
    },
    soluong: Number,
},
{
    timestamps:true
})
const KhoHang= mongoose.model("KhoHang", khohangSchema);
export default KhoHang;
```

```
import database from 'mongoose';


const reviewSchema = new database.Schema(
  {
    ten: { type: String, required: true },
    binhluan: { type: String, required: true },
    rating: { type: Number, required: true },
  },
  {
    timestamps: true,
  }
);

const sanphamSchema = new database.Schema({
    ten: { type: String, required: true, unique: true},
    hinhanh: { type: String, required: true},
    thuonghieu: { type: String, required: true},
    loai: { type: String, required: true},
    mota: { type: String, required: true},
    gia: { type: Number, required: true},
    countInStock: { type: Number, required: true},
    rating: { type: Number, required: true},
    numReviews: { type: Number, required: true},
    reviews: [reviewSchema],
},
{
    timestamps: true,
});

const Sanpham = database.model('Sanpham', sanphamSchema);

export default Sanpham;
```

SanPham.js

```js
import database from 'mongoose';


const reviewSchema = new database.Schema(
  {
    ten: { type: String, required: true },
    binhluan: { type: String, required: true },
    rating: { type: Number, required: true },
  },
  {
    timestamps: true,
  }
);

const sanphamSchema = new database.Schema({
    ten: { type: String, required: true, unique: true},
    hinhanh: { type: String, required: true},
    thuonghieu: { type: String, required: true},
    loai: { type: String, required: true},
    mota: { type: String, required: true},
    gia: { type: Number, required: true},
    countInStock: { type: Number, required: true},
    rating: { type: Number, required: true},
    numReviews: { type: Number, required: true},
    reviews: [reviewSchema],
},
{
    timestamps: true,
});

const Sanpham = database.model('Sanpham', sanphamSchema);

export default Sanpham;
```

User.js

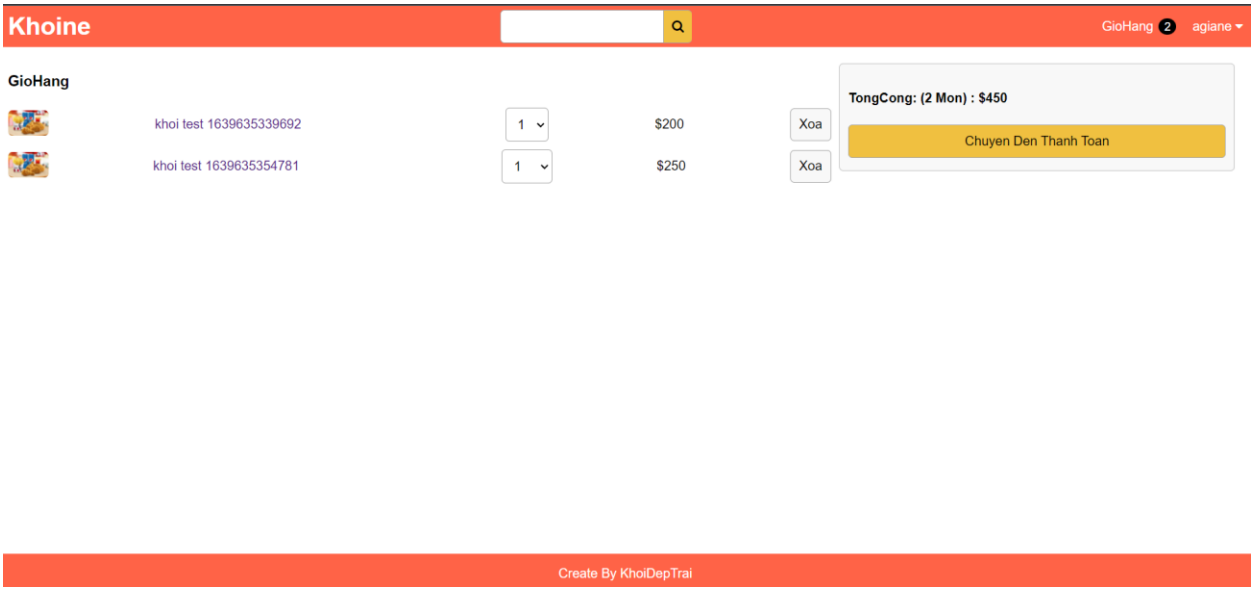
```js
import database from 'mongoose';

const userSchema = new database.Schema({
    name: {type: String, required: true},
    email: {type: String, required: true, unique: true},
    password: {type: String, required: true},
    isAdmin: {type: Boolean, default: false, required: true},
},
{
    timestamps: true,
});
const User = database.model('User', userSchema);
export default User;
```

Voucher.js

```js
import mongoose from 'mongoose';

export const VoucherSchema= mongoose.Schema({
    ngaybatdau: {type: Date, require:true},
    ngayketthuc: {type: Date, require:true},
    mota: {type: String, require:true},
    giamgia: {type: Number, require: true},
    toida: {type: Number, require: true}
})

const Voucher= mongoose.model('Voucher', VoucherSchema)

export default Voucher;
```

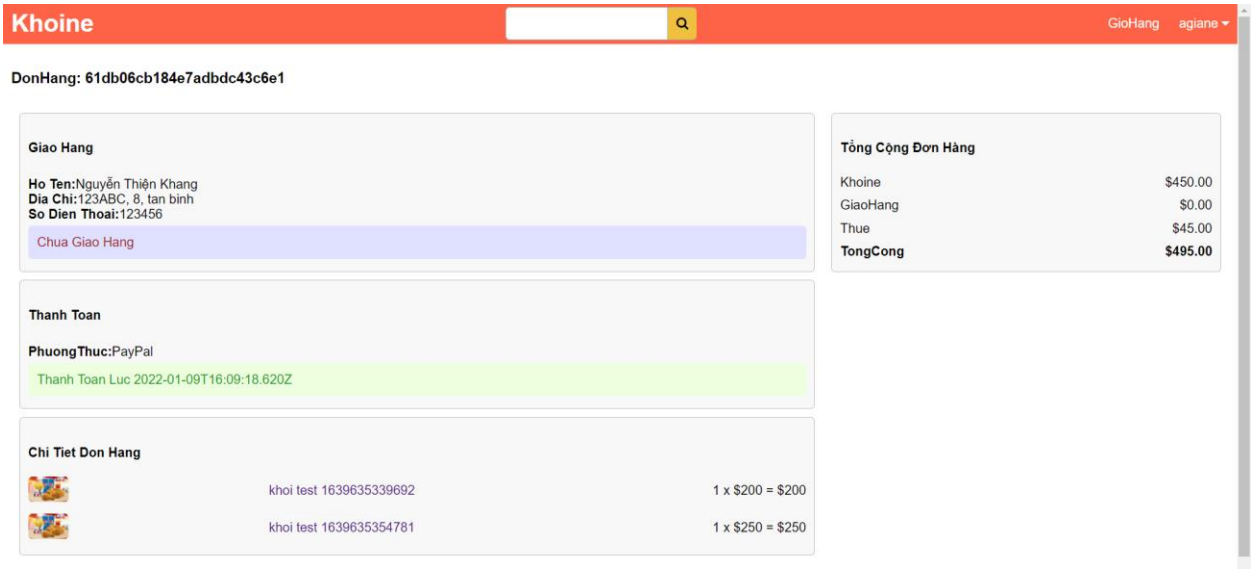
# 5. Hệ thống đã được xây dựng

## 5.1. Giỏ hàng




- Hàng sau khi được thêm vào giỏ sẽ được sắp xếp theo thứ tự thêm vào
- Có thể thay đổi số lượng từng món
- Có thể bỏ hàng khỏi giỏ
- Chuyển đến thanh toán để bắt đầu quá trình thanh toán
- Chức năng này thuộc về tài khoản khách hàng

## 5.2. Đơn hàng

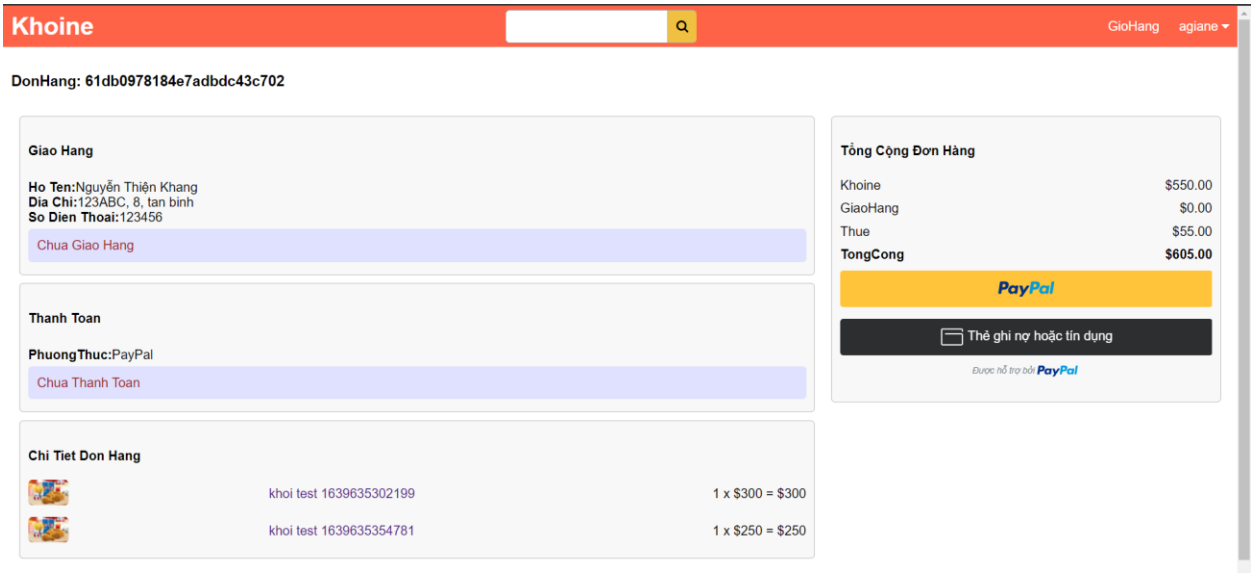



- Chức năng này thuộc về tài khoản khách hàng
- Đơn hàng sau khi đã thanh toán xong
- Bao gồm thông tin giao hàng đã nhập, xác nhận thanh toán, tổng tiền và danh sách

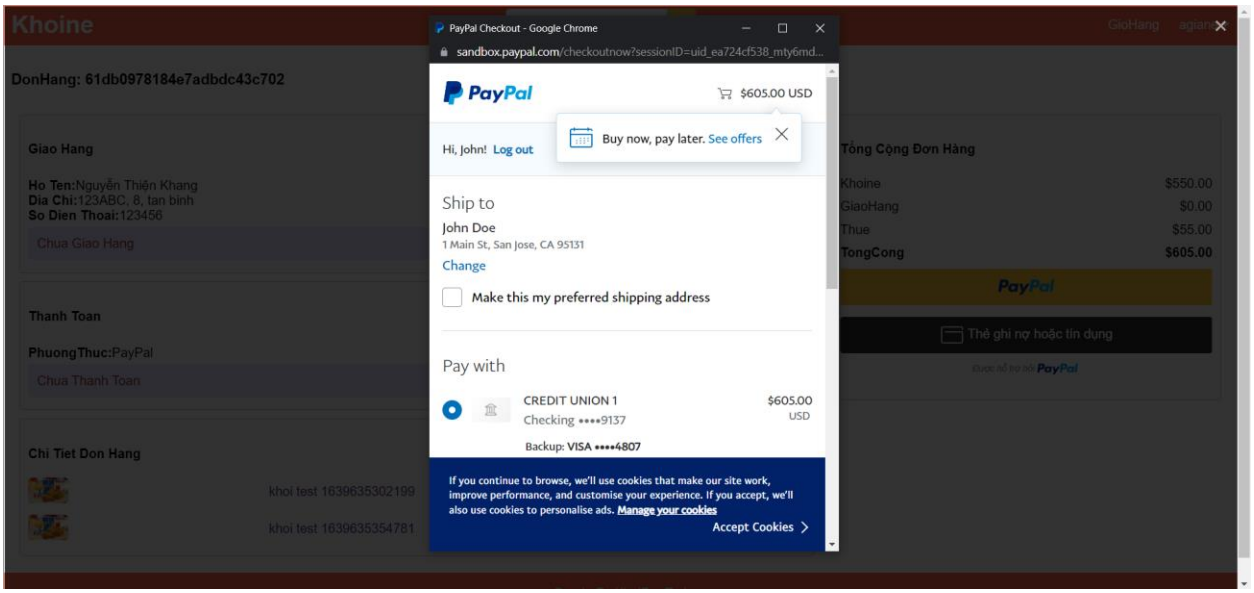
sản phẩm

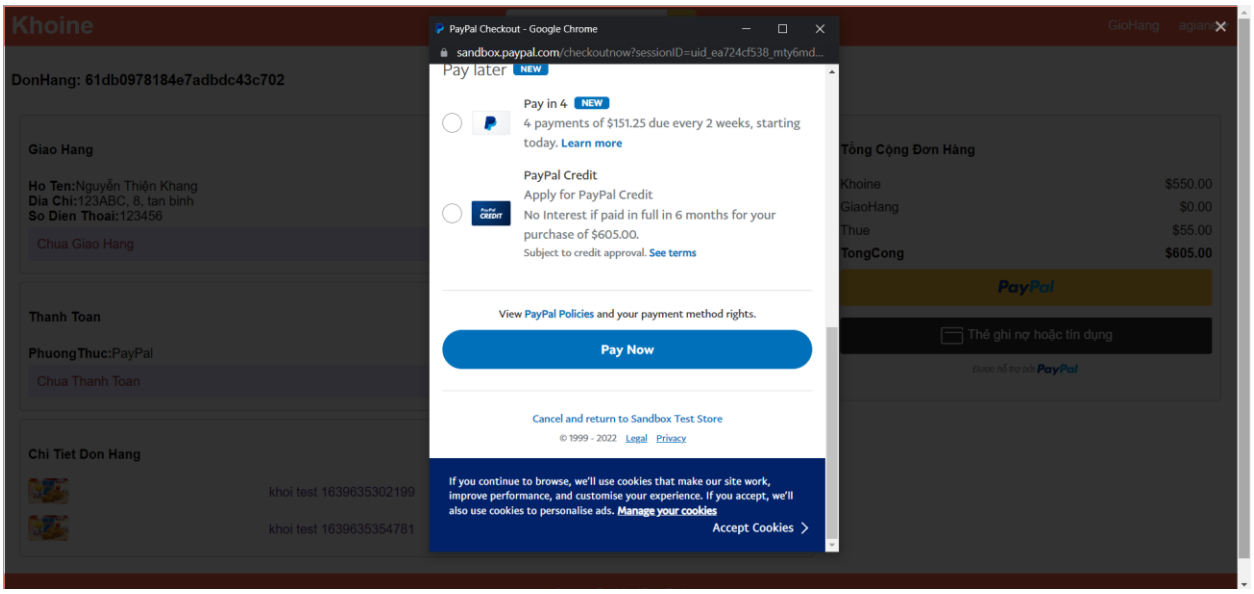
- Có thể xem lại trong lịch sử mua hàng

## 5.3. Thanh toán qua paypal




Trước khi bắt đầu thanh toán

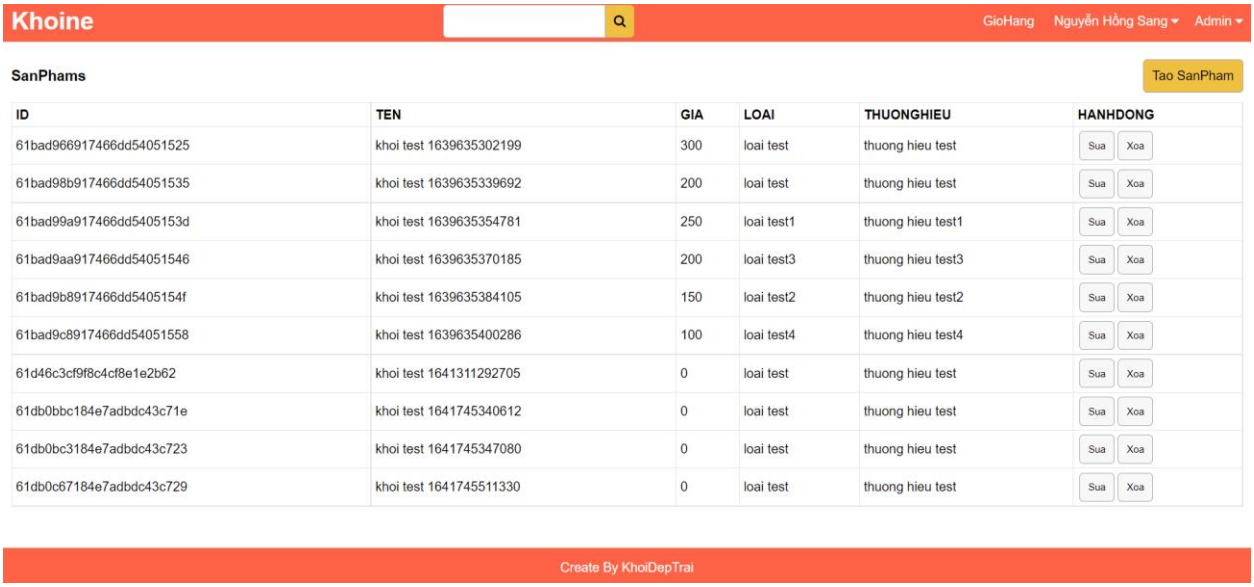




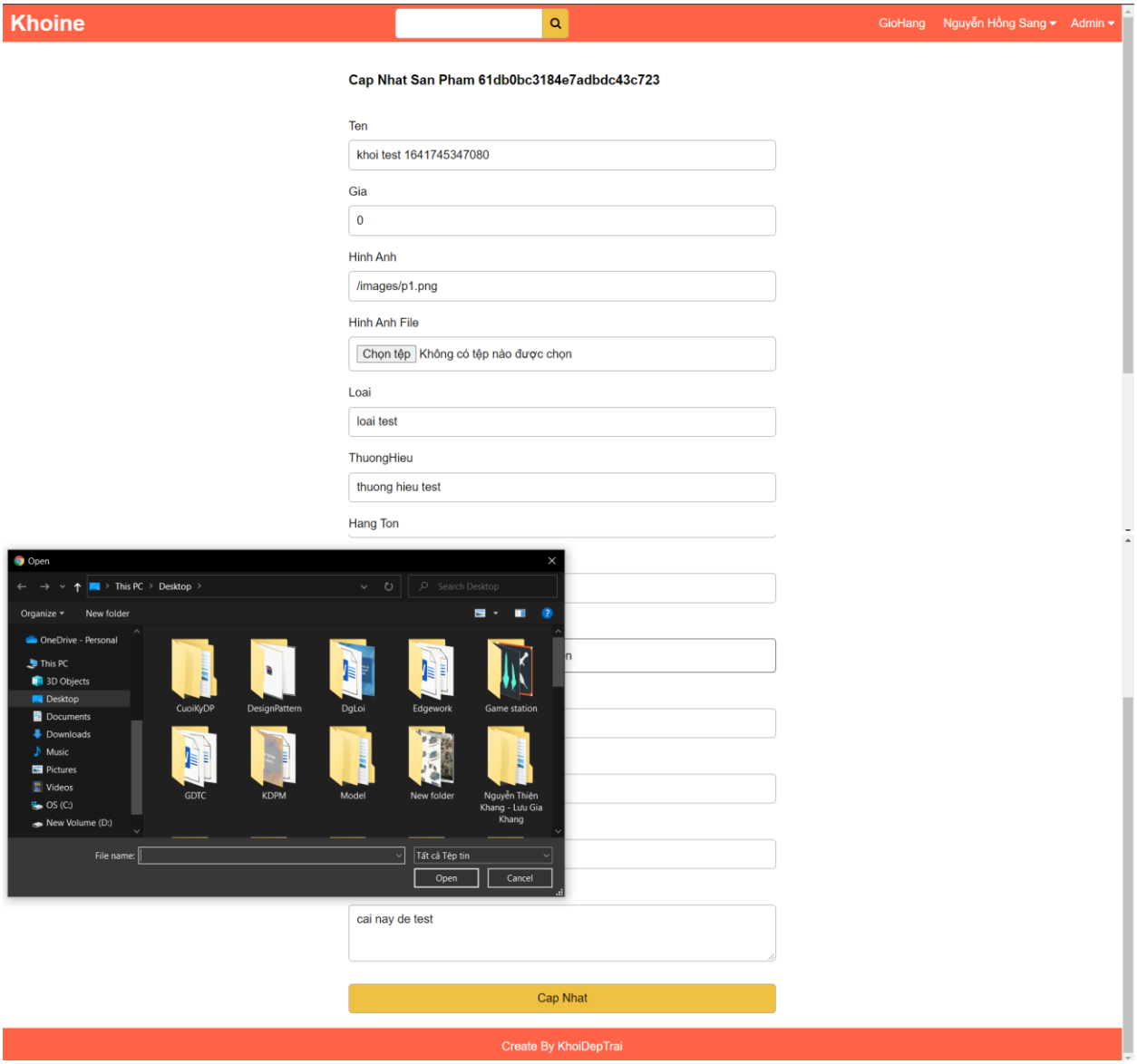

Giao diện thanh toán

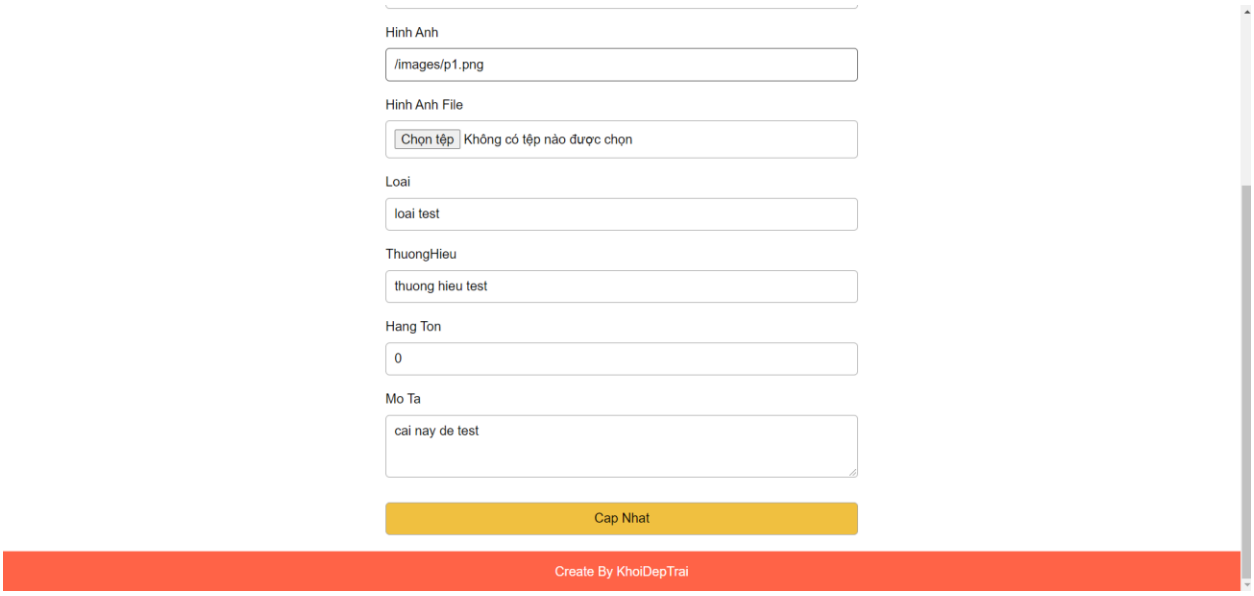
- Tự động gi lại số tiền thanh toán, người dùng không cần nhập lại
- Sau khi thanh toán thành công, đơn hàng sẽ trả về đơn hàng cùng dòng trạng thái thông báo đã thanh toán thành công

## 5.4. Tạo sản phẩm




Nhấn tạo sản phẩm để kích hoạt

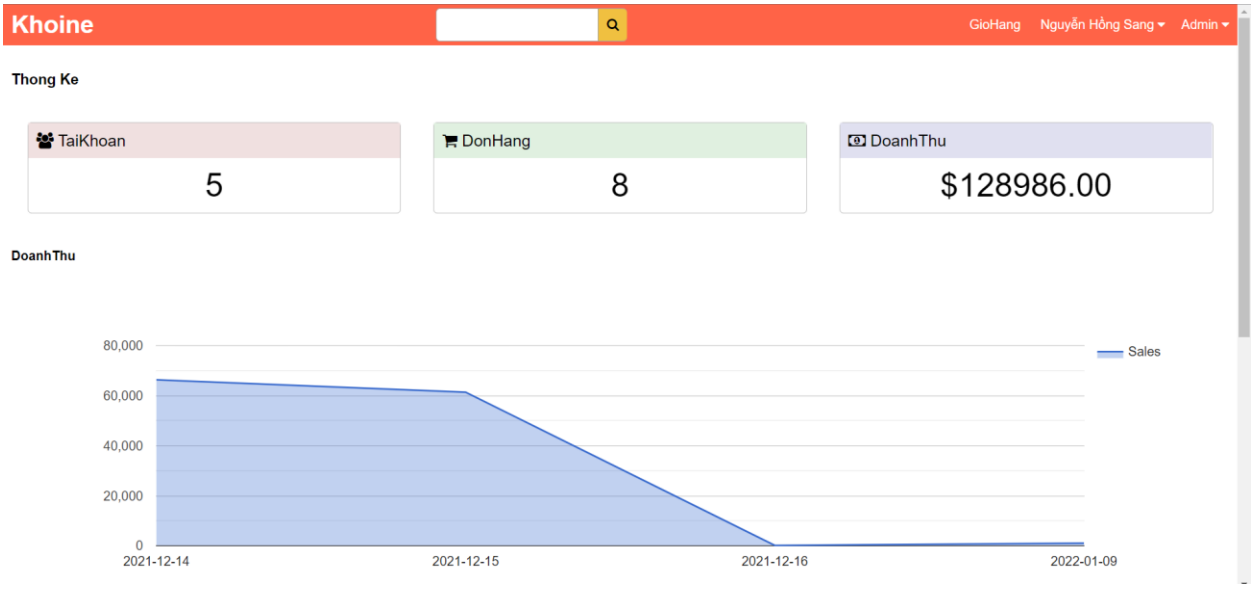

Khoine
GioHang
Nguyễn Hồng Sang
Admin
Cap Nhat San Pham 61db0bc3184e7adbdc43c723
Ten
khoi test 1641745347080
Gia
0
Hinh Anh
/images/p1.png
Hinh Anh File
Chọn tệp Không có tệp nào được chọn
Loai
loai test
ThuongHieu
thuong hieu test
Hang Ton
Open
This PC
Desktop
Search Desktop
Organize
New folder
OneDrive - Personal
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
OS (C:)
New Volume (D:)
CuoiKyDP
DesignPattern
DgLoi
Edgework
Game station
GDTC
KDPM
Model
New folder
Nguyễn Thiên Khang - Lưu Gia Khang
File name:
Tất cả Tệp tin
Open
Cancel
cai nay de test
Cap Nhat
Create By KhoiDepTrai

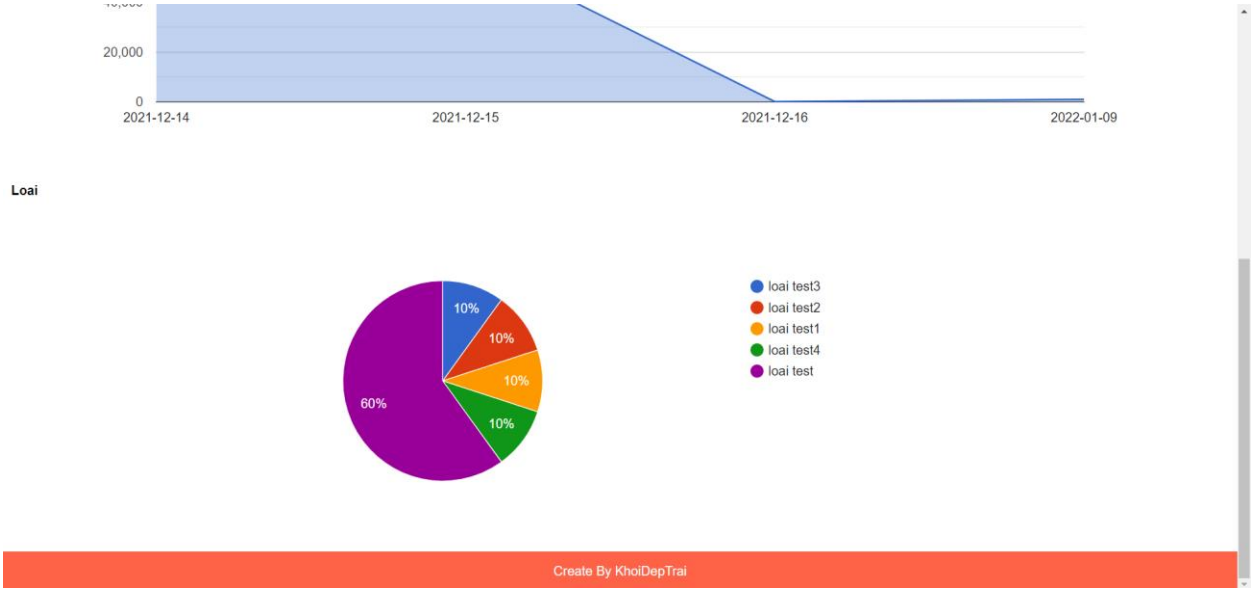



Giao diện chính

- Chỉ có tài khoản nhân viên (admin) mới có tính năng tạo mới
- Mã sản phẩm là tự sinh
- Nhập thông tin sản phẩm vào các ô
- Hình ảnh sản phẩm có thể chọn từ máy tính

## 5.5. Thống kê






- Chỉ có tài khoản nhân viên (admin) mới có tính năng xem thống kê
- Chức năng thống kê bao gồm các thông tin cơ bản như số lượng khách hàng,
- Doanh thu theo loại (hãng) và theo thời gian
- Thống kê tự động theo dữ liệu từ hệ thống

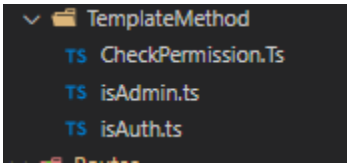
# 6. Ứng dụng mẫu thiết kế

## 6.1. Template method

### 1. Khái niệm:

Mẫu Template method định nghĩa một bộ khung của một thuật toán trong một chức năng, chuyển giao việc thực hiện nó cho các lớp con. Mẫu Template Method cho phép lớp con định nghĩa lại cách thực hiện của một thuật toán, mà không phải thay đổi cấu trúc thuật toán, rất thích hợp cho việc tạo ra các chủng loại robot khác nhau.

### 2. Cấu Trúc:

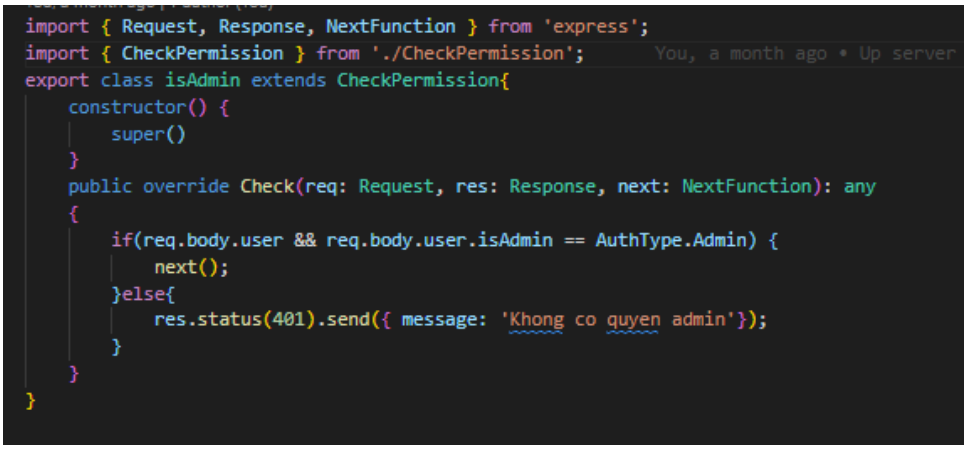



### 3. Vận dụng:

- Bước 1: Tạo abstract class với Template Method

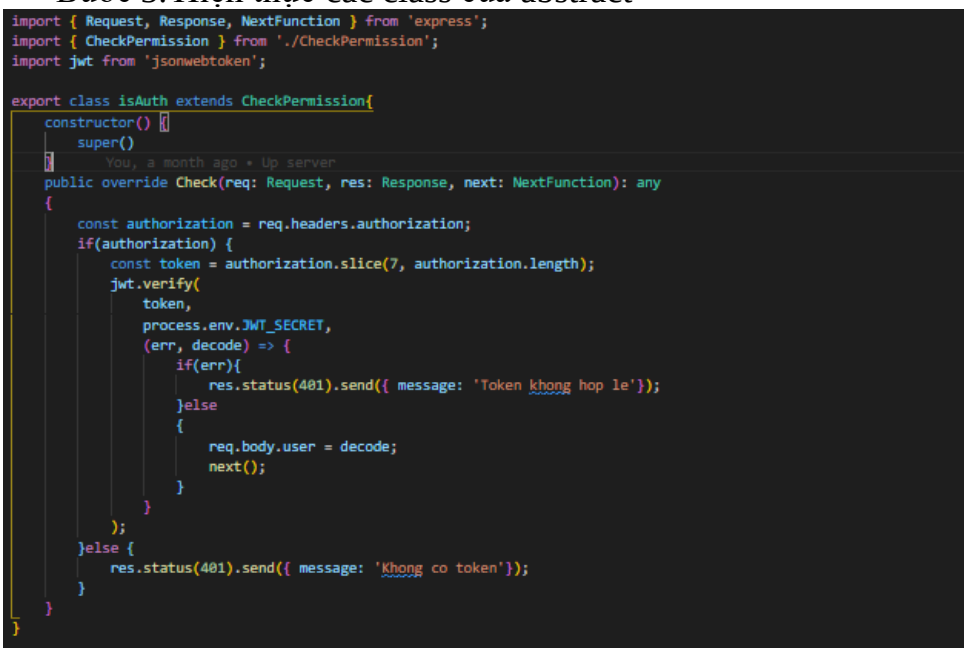
```
import { Request, Response, NextFunction } from 'express';
export abstract class CheckPermission{
    abstract Check(req: Request, res: Response, next: NextFunction): any
    public test(req: Request, res: Response, next: NextFunction): any{
        this.Check(req, res, next);
    }
}
```

- Bước 2: kế thừa abstract class



```
import { Request, Response, NextFunction } from 'express';
import { CheckPermission } from './CheckPermission';      You, a month ago • Up server
export class isAdmin extends CheckPermission{
    constructor() {
        super()
    }
    public override Check(req: Request, res: Response, next: NextFunction): any
    {
        if(req.body.user && req.body.user.isAdmin == AuthType.Admin) {
            next();
        }else{
            res.status(401).send({ message: 'Khong co quyen admin'});
        }
    }
}
```

- Bước 3: Hiện thực các class của abstract



```
import { Request, Response, NextFunction } from 'express';
import { CheckPermission } from './CheckPermission';
import jwt from 'jsonwebtoken';

export class isAuth extends CheckPermission{
    constructor() {
        super()
    }      You, a month ago • Up server
    public override Check(req: Request, res: Response, next: NextFunction): any
    {
        const authorization = req.headers.authorization;
        if(authorization) {
            const token = authorization.slice(7, authorization.length);
            jwt.verify(
                token,
                process.env.JWT_SECRET,
                (err, decode) => {
                    if(err){
                        res.status(401).send({ message: 'Token khong hop le'});
                    }else
                    {
                        req.body.user = decode;
                        next();
                    }
                }
            );
        }else {
            res.status(401).send({ message: 'Khong co token'});
        }
    }
}
```

- Bước 4: Ứng dụng

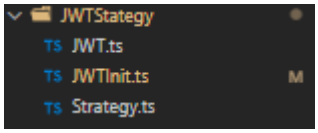
```
import { JWT } from './../MiddleWare/JWTStategy/JWT';
import { isAdmin } from './../MiddleWare/TemplateMethod/isAdmin';
import { isAuth } from './../MiddleWare/TemplateMethod/isAuth';
import { UserService } from './../services/UserService';
import {Request, Response, NextFunction} from 'express';
import bcryptjs from 'bcryptjs';
const get = async (req: Request, res: Response, next: NextFunction):Promise<any> =>{
    var checkAdmin = new isAdmin();
    var checkAuth = new isAuth();
    checkAdmin.test(req, res, next);
    checkAuth.test(req, res, next);      You, seconds ago • Uncommitted changes
    try{
        var UserSV = new UserService();
        UserSV.getAllUsers((data: any)=>{
            console.log(data)
        });
        res.send(UserSV.cache);
    }catch(err)
    {
        res.status(500).send({
            msg: err.message
        })
    }
}
```

## 6.2. Stategy

### 2. Khái niệm:

Strategy là mẫu thiết kế dùng để định nghĩa một họ các thuật toán, đóng gói mỗi thuật toán đó và làm cho chúng có khả năng thay đổi dễ dàng.Strategy cho phép giả thuật tùy biến một cách độc lập tại các Client sử dụng nó..

### 3. Cấu Trúc:




### 4. Vận dụng:

- Bước 1: Tạo abstract class với Stategy

```
You, a month ago | 1 author (You)
export abstract class JWTStrategy{      You, a month ago • Up server
    public abstract sign(data: any): any;
}
```

- Bước 2: kế thừa abstract class

```
import jwt from 'jsonwebtoken';
import {JWTStrategy} from './Strategy';

export class Token extends JWTStrategy{
    public sign(data: any): any{
        return jwt.sign({
            id: data.id,        You, a minute ago • Uncommitted changes
            name: data.name,
            email: data.email,
            isAdmin: data.isAdmin,
        }, process.env.JWT_SECRET,
        {
            expiresIn: '30d'
        });
    }
}
```

- Bước 3: class hiện thực của stategy

```
You, a month ago | 1 author (You)
import {JWTStrategy} from './Strategy';
import {Token} from './JWTInit';

export class JWT{
    Strategy: JWTStrategy;
    data: any
    constructor(data: any)
    {       You, a month ago • Up server
        this.data = data;
        this.Strategy = new Token();
    }
    public getToken():any{
        this.Strategy.sign(this.data);
    }
}
```

- Bước 4: Ứng dụng

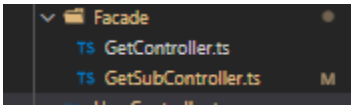
```
const Login = (req: Request, res: Response) => {
    try{
        UserService.UserLogin(req.body.email, req.body.password, (data: any) => {
            var token = new JWT(data);
            res.send({
                token:token.getToken(),
            });
        });
    }catch(error)
    {
        res.status(500).send({msg: error.message});
    }
}
```

## 6.3. Facade

### 3. Khái niệm:

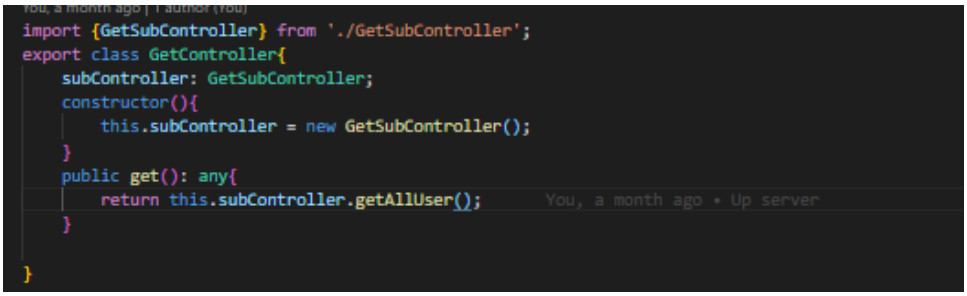
Facade Pattern là một trong những Pattern thuộc nhóm cấu trúc (Structural Pattern). Mẫu thiết kế này cung cấp một giao diện chung đơn giản thay cho một nhóm các giao diện có trong một hệ thống phức tạp. Hệ thống phức tạp này có thể là 1 thư viện (library), 1 framework, hay tập hợp các class phức tạp.

### 4. Cấu Trúc:




### 5. Vận dụng:

- Bước 1: Tạo class cha với Facade

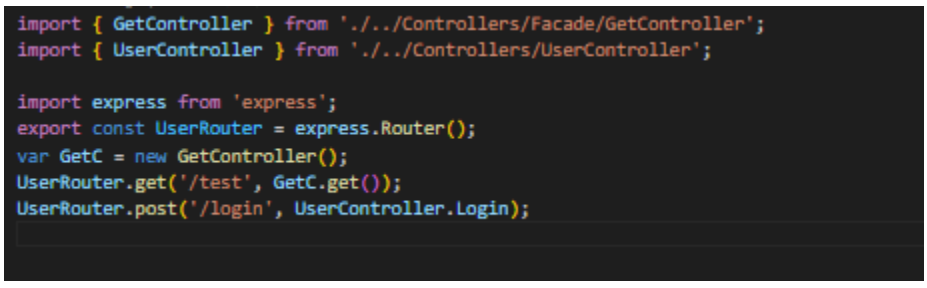


```
import {GetSubController} from './GetSubController';
export class GetController{
    subController: GetSubController;
    constructor(){
        this.subController = new GetSubController();
    }
    public get(): any{
        return this.subController.getAllUser();
    }

}
```

- Bước 2: class con của Facade

```
import { UserService } from './../../services/UserService';
import {Request, Response, NextFunction} from 'express';
const get = async (req: Request, res: Response, next: NextFunction):Promise<any> =>{
    try{
        var UserSV = new UserService();
        UserSV.getAllUsers((data: any) => {
            res.send(data);
        });
    }catch(err)
    {
        res.status(500).send({
            msg: err.message
        })
    }
}
export class GetSubController{
    constructor(){
    }
    public getAllUser(){
        return get;
    }
}
```

- Bước 3: Ứng dụng



```
import { GetController } from './../Controllers/Facade/GetController';
import { UserController } from './../Controllers/UserController';

import express from 'express';
export const UserRouter = express.Router();
var GetC = new GetController();
UserRouter.get('/test', GetC.get());
UserRouter.post('/login', UserController.Login);
```

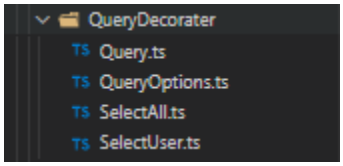
## 6.4. Decorater

### 4. Khái niệm:

Decorator là gì? Decorator là một trong 23 mẫu thiết kế Gang of Four mô tả làm thế nào để giải quyết vấn đề thiết kế các phần mềm mang tính hướng đối tượng một các linh hoạt và có tính tái sử dụng. Decorator cho phép người dùng thêm các

tính năng mới vào một đối tượng đã có mà không làm thay đổi cấu trúc lớp của nó.

## 5. Cấu Trúc:




## 6. Vận dụng:

- Bước 1: Tạo abstract class với Decorater

```ts
export abstract class Query{
    public getSql: string
    public getData(): string{
        return this.getSql;
    }
}
```

- Bước 2: class abstract thứ 2

```ts
import {Query} from './Query';

export abstract class QueryOptions extends Query{
    decoratedQuery: Query;
    public abstract getData():string;
}
```

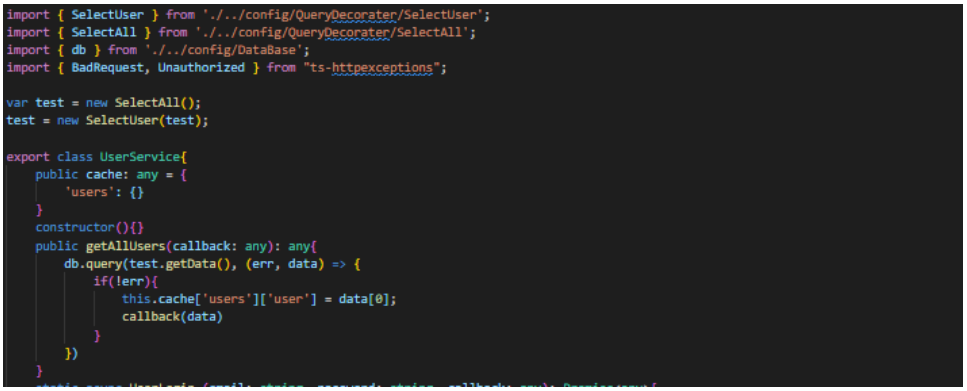
- Bước 3: Hiện thực class abstract thứ nhất

```ts
import {Query} from './Query';

export class SelectAll extends Query{
    public getSql = "SELECT * ";
}
```

- Bước 4: Hiện thực class abstract thứ hai

```ts
import {QueryOptions} from './QueryOptions';
import {Query} from './Query';


export class SelectUser extends QueryOptions{
    decoratedQuery: Query;
    constructor(query: Query){
        super();
        this.decoratedQuery = query;
    }
    public getData():string{
        return this.decoratedQuery.getData() + 'FROM user';
    }
}
```

- Bước 5: Ứng dụng



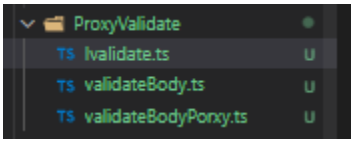
```
import { SelectUser } from './../config/QueryDecorater/SelectUser';
import { SelectAll } from './../config/QueryDecorater/SelectAll';
import { db } from './../config/DataBase';
import { BadRequest, Unauthorized } from "ts-httpexceptions";

var test = new SelectAll();
test = new SelectUser(test);

export class UserService{
    public cache: any = {
        'users': {}
    }
    constructor(){}
    public getAllUsers(callback: any): any{
        db.query(test.getData(), (err, data) => {
            if(!err){
                this.cache['users']['user'] = data[0];
                callback(data)
            }
        })
    }
```

## 6.5. Proxy

### 5. Khái niệm:

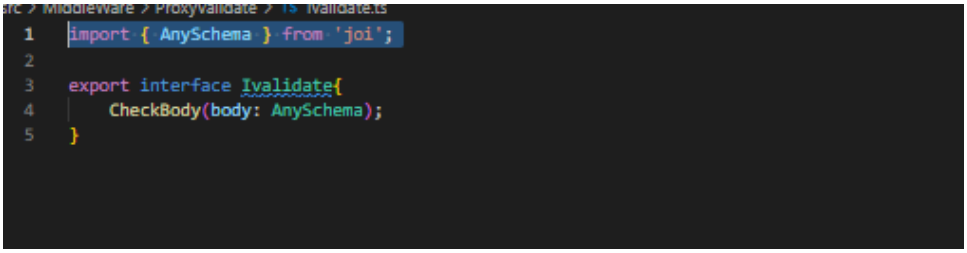
Mẫu Proxy (người đại diện) đại diện cho một đối tượng khác thực thi các phương thức, phương thức đó có thể được định nghĩa lại cho phù hợp với múc đích sử dụng. Để đơn giản hơn bạn có thể nghĩ đến khái niệm HTTP proxy trong mạng máy tính, nó là một gateway giữa trình duyệt (client) và máy chủ (server).

### 6. Cấu Trúc:




### 7. Vận dụng:

- Bước 1: Tạo interface với Proxy



```
1  import { AnySchema } from 'joi';
2
3  export interface Ivalidate{
4      CheckBody(body: AnySchema);
5  }
```

- Bước 2: kế thừa interface

src > MiddleWare > ProxyValidate > TS validateBody.ts

```ts
import { validateBodyProxy } from './validateBodyPorxy';
import { Ivalidate } from './Ivalidate';
import { AnySchema } from 'joi';

export class validateBody implements Ivalidate{
    public override CheckBody(body: AnySchema){
        validateBodyProxy.CheckBody(body);
    }
}
```

- Bước 3: class Proxy

MiddleWare > ProxyValidate > TS validateBodyPorxy.ts

```ts
import { Ivalidate } from './Ivalidate';
import { validateBody } from './validateBody';
import { Request, Response, NextFunction } from 'express';
import { AnySchema } from 'joi';
import { StatusCodes } from 'http-status-codes';


export class validateBodyProxy implements Ivalidate{
    private validateBd: validateBody;
    constructor(validateBd: validateBody){
      this.validateBd = validateBd;
    }
    publics static CheckBody(schema: AnySchema){
        return (req: Request, res: Response, next: NextFunction) => {
      const value = req.body;
      const result = schema.validate(value);
      if (result.error) {
        const firstError = result.error.details[0];
        return res.status(StatusCodes.BAD_REQUEST).json({
          msg: firstError.message,
        });
      }
      return next();
      };
    }
}

export { validateBody };
```

# 7. Tổng kết

## 7.1. Các chức năng đã hoàn thành

- Đã hoàn thành các usercase đã đề ra ở phần 2.2

## 7.2. Các chức năng có thể phát triển

- Định vị bản đồ khi nhận địa chỉ giao hàng
- Kết nối với ứng dụng trên smartphone