| **Họ và Tên** | **MSSV** | **Chức vụ** |
| --- | --- | --- |
| Nguyễn Minh Kiên | 3122411103 | Nhóm trưởng |
| Lê Nhựt Huy | 3122411063 | Thành viên |
| Phạm Nhật Phương | 3122411162 | Thành viên |
| Phạm Nguyễn Thế Hào | 3122411048 | Thành viên |



**MỤC LỤC**

[PHÂN CÔNG](#_k9v6robfq1ck) 3

[1. Sơ đồ khối](#_ujz5wzi5ing2) 4

[2. Decomposition View: C4](#_73x323t3g8d2) 8

[2.1. sơ đồ C1 - System Context](#_fwe5ewcmww2n) 8

[2.2. sơ đồ C2 – Container](#_ahsltxnje8r0) 9

[2.3. sơ đồ C3 – Component (High-Level)](#_i2h7ensb54us) 12

[2.4. Sơ đồ C4: Implementation](#_vgi56ulvu2zj) 15

[3.Communication View](#_le69qioep21m) 40

[3.1. Xem và Lọc/Sắp xếp Sản phẩm](#_5zat3t3nmnrd) 40

[3.3. Quản Lý Giỏ Hàng](#_rfumm5ogophf) 41

[3.4. Thanh toán đơn hàng](#_yhve3o9e4s1o) 42

[3.5. Quản lý sản phẩm ( Admin)](#_78i503pkceqj) 43

[3.6. Xem Lịch sử Đơn hàng](#_jmbfqeyqvtfz) 43

[3.7. Đánh giá sản phẩm](#_4nkn5goraabz) 43

[4. Deployment View](#_xgthi1d2tge7) 44

[5. Sơ đồ kiến trúc SPA (Single Page Application):](#_ix986mfn28fj) 45

[6. Sơ đồ triển khai CI/CD:](#_5c6kejx8dxl7) 45

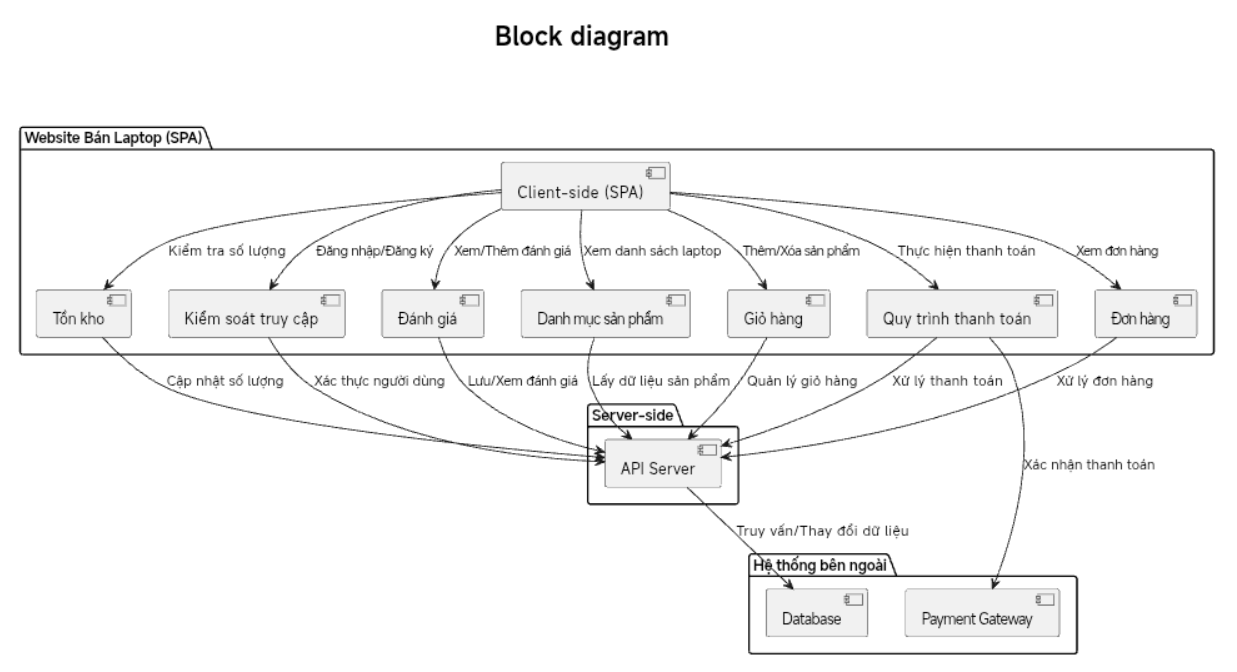
[7. Sơ đồ API của hệ thống:](#_t4kmgvniac72) 46

[8. sơ đồ xử lý 1 request:](#_2nmyj33ts95h) 47

# PHÂN CÔNG

| STT | Họ tên | Email | Phân công |
| --- | --- | --- | --- |
| 1 | Nguyễn Minh Kiên | Nguyenminhkien5577@gmail.com | Sơ đồ triển khai CI/CD, sơ đồ System Context, Communication View |
| 2 | Lê Nhựt Huy | lenhuthuy345@gmail.com | Sơ đồ Component, Deployment View |
| 3 | Phạm Nguyễn Thế Hào | phamnguyen.thehao2703@gmail.com | Communication View¸sơ đồ API, sơ đồ Container |
| 4 | Phạm Nhật Phương | pnhatphuong2011@gmail.com | Sơ đồ khối, sơ đồ SPA, sơ đồ xử lý Request, sơ đồ Implementation diagram |

## 1. Sơ đồ khối



## 

## 1.1. Giới thiệu đôi chút về sơ đồ khối

## Sơ đồ khối trên mô tả kiến trúc tổng thể của hệ thống Website bán laptop (SPA – Single Page Application). Đây là sơ đồ ở mức khái quát hóa kiến trúc, cho thấy các thành phần chính của hệ thống, cách chúng được tổ chức và cách chúng liên kết với nhau.

## Khác với sơ đồ khái niệm (conceptual diagram) vốn chỉ mô tả các đối tượng nghiệp vụ và mối quan hệ giữa chúng (ví dụ: Người dùng – Đơn hàng – Sản phẩm), sơ đồ khối này đi thêm một bước, thể hiện cách các khối chức năng vận hành trên từng lớp: Client-side, Server-side, và Hệ thống bên ngoài. Nó giúp ta hiểu rõ không chỉ có “cái gì tồn tại” trong hệ thống, mà còn “các khối đó giao tiếp với nhau như thế nào”.

## 

## 1.2 Các thành phần chính

### a. Client-side (SPA)

## Đây là lớp giao diện người dùng, chạy trên trình duyệt. SPA (Single Page Application) nghĩa là toàn bộ trải nghiệm của người dùng được xử lý trong một trang web duy nhất, không phải tải lại toàn bộ trang khi chuyển đổi chức năng. Các module chính gồm:

## Danh mục sản phẩm: cung cấp cho người dùng khả năng duyệt qua toàn bộ danh sách laptop có trong hệ thống. Dữ liệu này không tự chứa trên client mà sẽ được lấy từ server qua API.

## Giỏ hàng: nơi người dùng thêm hoặc xóa sản phẩm. Thông tin giỏ hàng có thể tạm thời lưu trên client, nhưng để đồng bộ và duy trì trạng thái giữa nhiều thiết bị, nó cần gửi về server.

## Tồn kho: hiển thị số lượng sản phẩm hiện có. Đây là tính năng quan trọng để đảm bảo khách không đặt mua vượt quá số lượng sẵn có.

## Kiểm soát truy cập: hỗ trợ người dùng đăng nhập, đăng ký, hoặc quản lý tài khoản. Cơ chế này thường đi kèm với session hoặc token (JWT).

## Đánh giá: cho phép người dùng xem nhận xét từ khách hàng khác và thêm nhận xét mới cho sản phẩm đã mua.

## Quy trình thanh toán: tập trung các bước thanh toán, từ nhập thông tin giao hàng, chọn phương thức thanh toán, đến xác nhận đơn hàng.

## Đơn hàng: nơi người dùng có thể theo dõi trạng thái đơn hàng đã đặt (ví dụ: đang xử lý, đã giao hàng, đã hủy).

## Như vậy, Client-side đóng vai trò là cửa ngõ cho mọi tương tác của người dùng với hệ thống.

### b. Server-side (API Server)

## Đây là tầng trung gian, giữ vai trò “bộ não” xử lý logic nghiệp vụ. API Server tiếp nhận các yêu cầu từ Client-side, phân tích và xử lý, sau đó giao tiếp với hệ thống bên ngoài. Các nhiệm vụ chính:

## Xử lý yêu cầu dữ liệu: ví dụ, khi Client cần danh sách sản phẩm, API Server sẽ truy vấn Database và trả về kết quả.

## Đảm bảo tính toàn vẹn dữ liệu: API Server đảm nhận việc kiểm soát dữ liệu (giỏ hàng, tồn kho, đơn hàng, đánh giá) để không xảy ra sai lệch.

## Xác thực và phân quyền: khi Client gửi yêu cầu có liên quan đến tài khoản người dùng, API Server sẽ xác minh token hoặc session để đảm bảo tính hợp lệ.

## Tích hợp các dịch vụ bên ngoài: API Server chính là cầu nối để Client có thể gián tiếp tương tác với Database và Payment Gateway.

## Nói cách khác, Client-side không bao giờ truy cập trực tiếp vào Database hay Payment Gateway, mà tất cả đều phải đi qua API Server.

### c. Hệ thống bên ngoài

## Database: lưu trữ toàn bộ thông tin cốt lõi của hệ thống, bao gồm người dùng, sản phẩm, tồn kho, giỏ hàng, đơn hàng, đánh giá. Đây là nền tảng dữ liệu để mọi module có thể hoạt động ổn định.

## Payment Gateway: cổng thanh toán trung gian, chịu trách nhiệm xác thực và xử lý các giao dịch tài chính trực tuyến. Hệ thống website không tự thực hiện giao dịch, mà chỉ gửi yêu cầu đến Payment Gateway để đảm bảo an toàn và tuân thủ pháp lý.

## 1.3. Luồng hoạt động

## Người dùng thao tác trên Client-side (SPA), ví dụ: duyệt sản phẩm, thêm laptop vào giỏ, hoặc nhấn thanh toán.

## Ứng dụng SPA gửi yêu cầu (HTTP request, thường ở dạng JSON qua REST API hoặc GraphQL) đến API Server.

## API Server xử lý logic nghiệp vụ:

## Nếu yêu cầu là đọc dữ liệu (xem sản phẩm, tồn kho, đơn hàng), nó truy vấn Database và trả kết quả về.

## Nếu yêu cầu là ghi dữ liệu (thêm giỏ hàng, tạo đơn hàng, gửi đánh giá), nó cập nhật Database và phản hồi trạng thái cho Client.

## Với thao tác thanh toán, Quy trình thanh toán từ Client sẽ gọi đến Payment Gateway qua API Server. Payment Gateway xác nhận giao dịch, sau đó kết quả được trả lại API Server.

## API Server ghi nhận kết quả thanh toán và cập nhật lại Database để đảm bảo dữ liệu đơn hàng chính xác.

## Cuối cùng, Client-side (SPA) nhận phản hồi và hiển thị kết quả cho người dùng (ví dụ: “Thanh toán thành công”, “Đơn hàng #12345 đang được xử lý”).

## 1.4. Ý nghĩa và mối liên hệ với sơ đồ khái niệm

## Sơ đồ khái niệm: tập trung vào nghiệp vụ (ai, cái gì, quan hệ gì). Ví dụ: Người dùng → tạo → Đơn hàng → chứa → Sản phẩm.

## Sơ đồ khối: tập trung vào kiến trúc triển khai, tức là những nghiệp vụ đó được hiện thực bằng module nào, chạy ở lớp nào, giao tiếp qua đâu.

## Trong thực tế, sơ đồ khối này là bước nối giữa sơ đồ khái niệm và sơ đồ chi tiết (như sequence diagram hoặc activity diagram). Nó giúp người thiết kế hệ thống và người phát triển có cái nhìn tổng thể, từ đó dễ dàng phân công công việc, thiết kế API, và đảm bảo rằng các chức năng nghiệp vụ đã được ánh xạ đầy đủ vào các khối kỹ thuật.

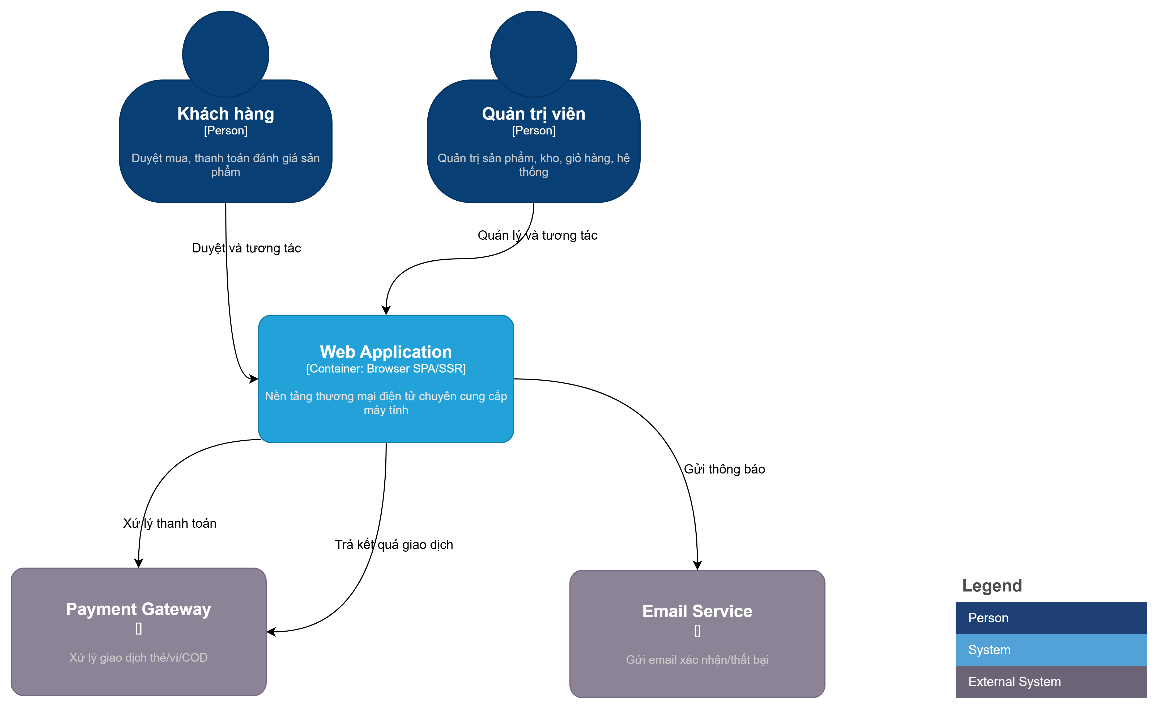
## Nói cách khác, sơ đồ khối chính là “bản đồ giao tiếp” cho toàn bộ website: ai (client) nói chuyện với ai (server, database, payment gateway), và nói chuyện như thế nào.

## 

## 

## 2. Decomposition View: C4

### 2.1. sơ đồ C1 - System Context



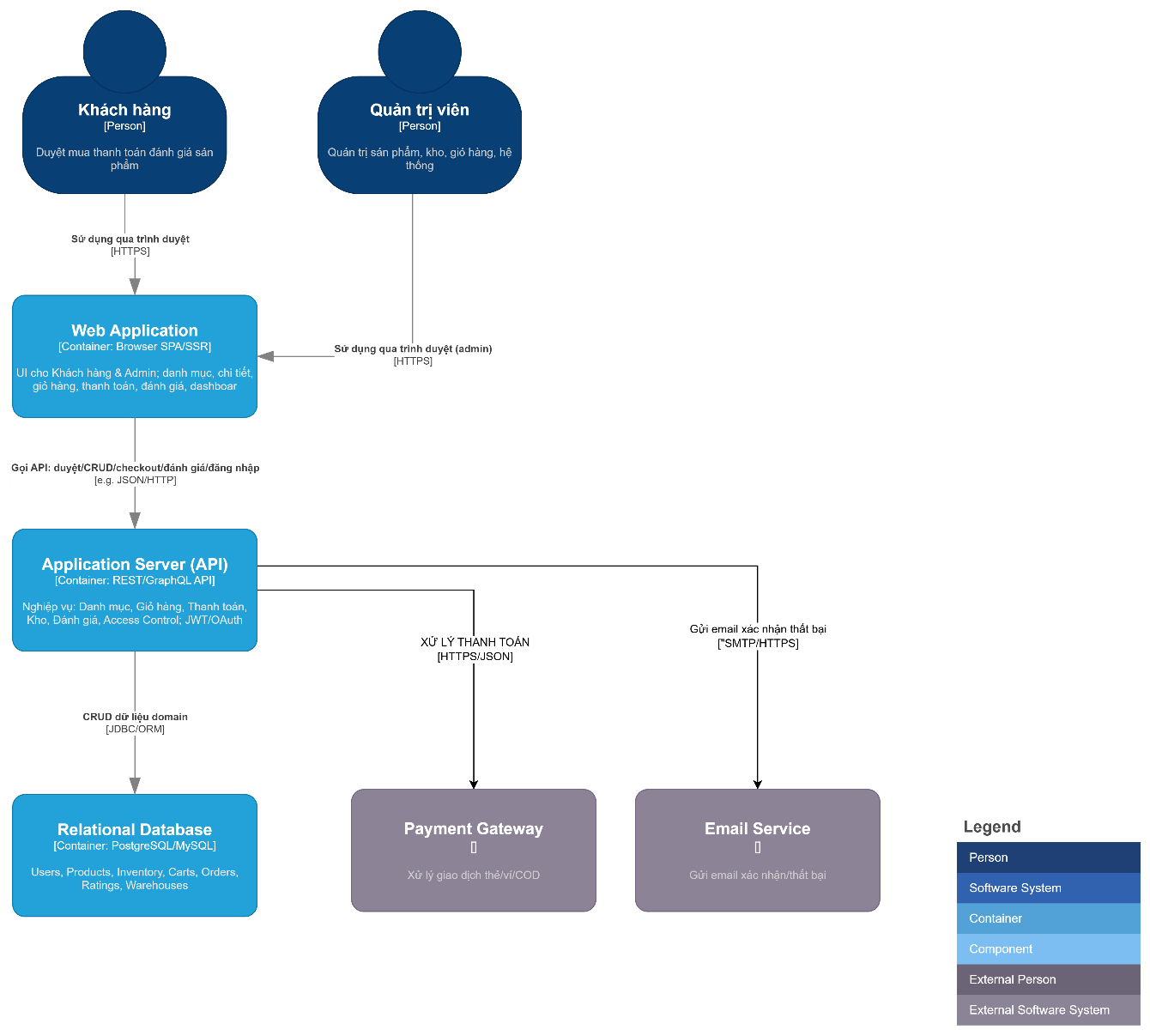
**1. Giải thích**

* My Meetings (Hệ thống): Là hệ thống chính đang được phân tích. Nó có vai trò quản lý các cuộc họp, nhóm họp và các thành viên.
* Member: Là người dùng thông thường của hệ thống. Có thể tổ chức, tham gia và quản lý các cuộc họp và là người thực hiện các giao dịch thanh toán.
* Administrator: Là người quản trị hệ thống, có quyền cao nhất để quản lý các thành viên, nhóm họp và cuộc họp.
* Payment Gateway: Là một dịch vụ bên ngoài, chuyên xử lý các giao dịch thanh toán.
* Email Service: Là một dịch vụ bên ngoài, dùng để gửi các email thông báo.

**2. Luồng hoạt động**

* Luồng từ Thành viên (Member):
  + Tổ chức & Tham gia cuộc họp**:** Thành viên tương tác trực tiếp với hệ thống "My Meetings" để tổ chức các cuộc họp, nhóm họp hoặc tham gia vào các cuộc họp đã có.
  + Thanh toán: Khi cần thanh toán, thành viên gửi yêu cầu "Pay in" đến "My Meetings". Hệ thống này sau đó sẽ "ủy thác việc thanh toán" cho Payment Gateway. Cổng thanh toán sẽ xử lý giao dịch và "trả về thông tin về việc thanh toán"
  + Gửi email: Thành viên có thể yêu cầu "My Meetings" "gửi email" cho họ hoặc người khác. Hệ thống sẽ chuyển yêu cầu này đến Email Service, và dịch vụ email sẽ thực hiện "gửi email".
* Luồng từ Quản trị viên (Administrator):
  + Quản lý hệ thống: Quản trị viên có quyền "quản lý các thành viên, nhóm họp, cuộc họp" thông qua hệ thống "My Meetings".
  + Gửi email: Tương tự như thành viên, quản trị viên cũng có thể sử dụng dịch vụ Email Service để gửi email.
* Luồng giữa các hệ thống:
  + My Meetings và Payment Gateway: My Meetings gửi yêu cầu thanh toán và nhận lại kết quả.
  + My Meetings và Email Service: My Meetings gửi yêu cầu gửi email và nhận thông báo trạng thái từ Email Service.

### 2.2. sơ đồ C2 – Container



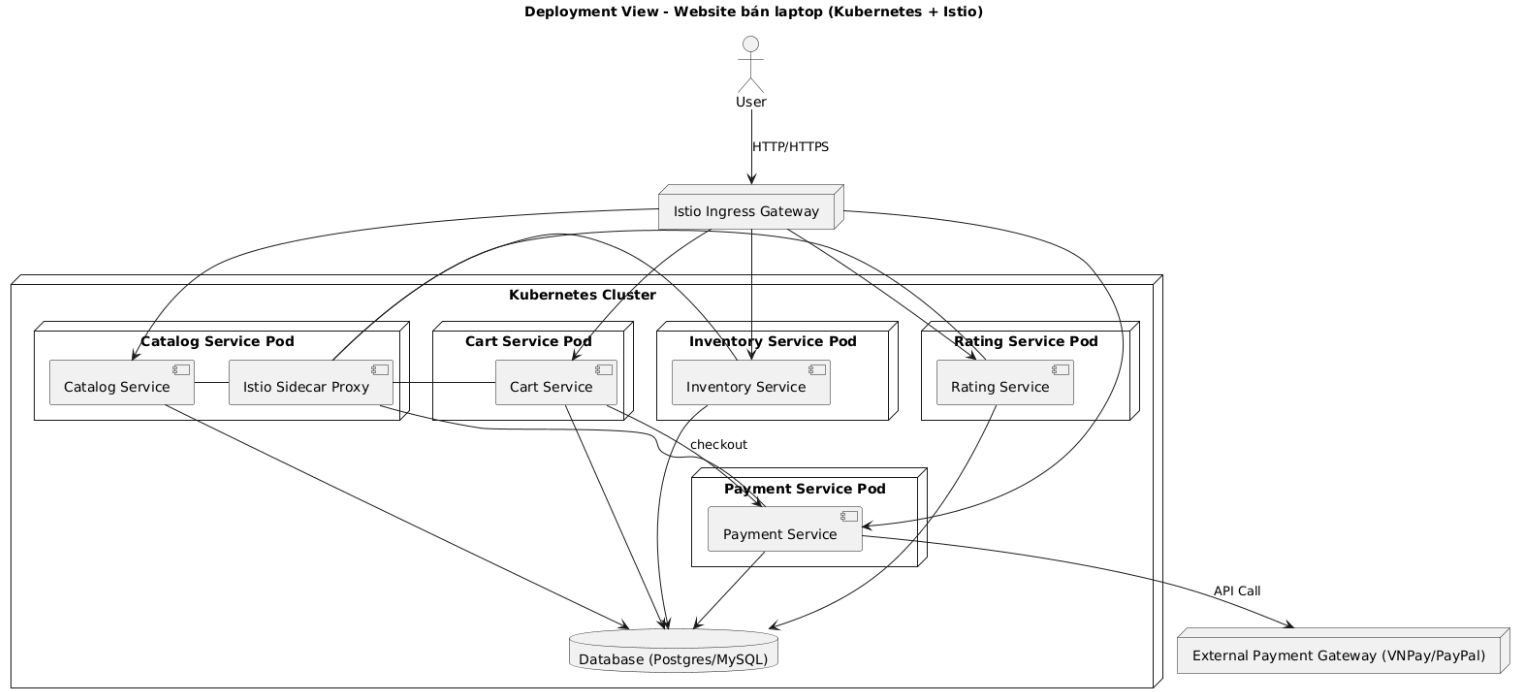
**Giải thích**

* Web Application (Ứng dụng Web): Là giao diện người dùng, nơi khách hàng và quản trị viên tương tác với hệ thống. Được xây dựng dưới dạng một ứng dụng trang đơn (SPA) hoặc kết xuất phía máy chủ (SSR).
* Application Server (API): Xử lý logic nghiệp vụ, giỏ hàng, thanh toán, quản lý kho, và kiểm soát truy cập.
* Relational Database (Cơ sở dữ liệu quan hệ): Lưu trữ tất cả dữ liệu của hệ thống, bao gồm thông tin người dùng, sản phẩm, kho hàng, giỏ hàng, đơn hàng và đánh giá.
* Payment Gateway (Cổng thanh toán): Một hệ thống bên ngoài xử lý các giao dịch thanh toán.
* Email Service (Dịch vụ email): Một hệ thống bên ngoài dùng để gửi email xác nhận hoặc thông báo.

**Luồng hoạt động**

* Khách hàng và Quản trị viên sử dụng trình duyệt web (HTTP) để tương tác với Ứng dụng Web.
* Ứng dụng web sau đó gọi các API của Application Server (sử dụng JSON/HTTP) để thực hiện các chức năng như duyệt sản phẩm, thêm vào giỏ hàng, thanh toán, v.v.
* Luồng dữ liệu nội bộ:
* Application Server thực hiện các thao tác CRUD (tạo, đọc, cập nhật, xóa) trên dữ liệu trong Relational Database để quản lý thông tin.
* Tương tác với hệ thống bên ngoài:
* Khi có giao dịch thanh toán, Application Server sẽ gửi yêu cầu xử lý thanh toán (thường là qua JSON/HTTP) đến Payment Gateway.
* Khi cần gửi thông báo, Application Server sẽ gửi yêu cầu gửi email (thường qua SMTP/HTTPS) đến Email Service.

### 2.3. sơ đồ C3 – Component (High-Level)



**Các thành phần chính**

**1. Catalog Component**

* Chức năng: Quản lý danh mục sản phẩm laptop (CRUD sản phẩm, xem chi tiết).
* Giao tiếp:
  + Lưu/truy vấn dữ liệu từ Products Data schema.
  + Được Cart gọi khi người dùng thêm sản phẩm vào giỏ.
  + Được Rating gọi khi cần hiển thị sản phẩm kèm đánh giá.

**2. Cart Component**

* Chức năng: Quản lý giỏ hàng của người dùng (thêm/xóa/sửa, hiển thị giỏ).
* Giao tiếp:
  + Lưu/truy xuất dữ liệu từ Cart Data schema.
  + Gọi Catalog để kiểm tra thông tin sản phẩm.
  + Gọi Order khi người dùng bấm checkout.

**3. Order Component**

* Chức năng: Quản lý đơn hàng (tạo đơn, trạng thái, lịch sử đơn hàng).
* Giao tiếp:
  + Lưu/truy xuất dữ liệu từ Orders Data schema.
  + Nhận dữ liệu từ Cart khi checkout.
  + Gọi Payment để xử lý thanh toán.

**4. Payment Component**

* Chức năng: Quản lý thanh toán (phí thành viên, phí trễ hạn, thanh toán đơn hàng).
* Giao tiếp:
  + Lưu dữ liệu vào Payments Data schema.
  + Nhận yêu cầu từ Order để xử lý thanh toán.
  + Gọi External Payment Gateway (VNPay, PayPal) qua API.

**5. Inventory Component**

* Chức năng: Quản lý kho (số lượng tồn kho của từng laptop).
* Giao tiếp:
  + Lưu/truy vấn dữ liệu từ Inventory Data schema.
  + Giảm số lượng tồn kho khi Order thành công.

**6. Rating Component**

* Chức năng: Người dùng đánh giá sản phẩm (rating, comment).
* Giao tiếp:
  + Lưu/truy xuất dữ liệu từ Ratings Data schema.
  + Gọi Catalog để hiển thị đánh giá theo sản phẩm.

**7. Access Control Component**

* Chức năng: Kiểm soát truy cập (login, đăng ký, phân quyền User/Admin).
* Giao tiếp:
  + Lưu/truy vấn từ Users Data schema.
  + Các service khác sẽ dùng nó để xác thực người dùng (Cart, Order, Payment…).

**Các thành phần hỗ trợ**

**a. Database schemas**

* Mỗi component có schema riêng (Products, Cart, Orders, Payments, Inventory, Ratings, Users).
* Điều này tuân theo nguyên tắc Database-per-service trong microservices.

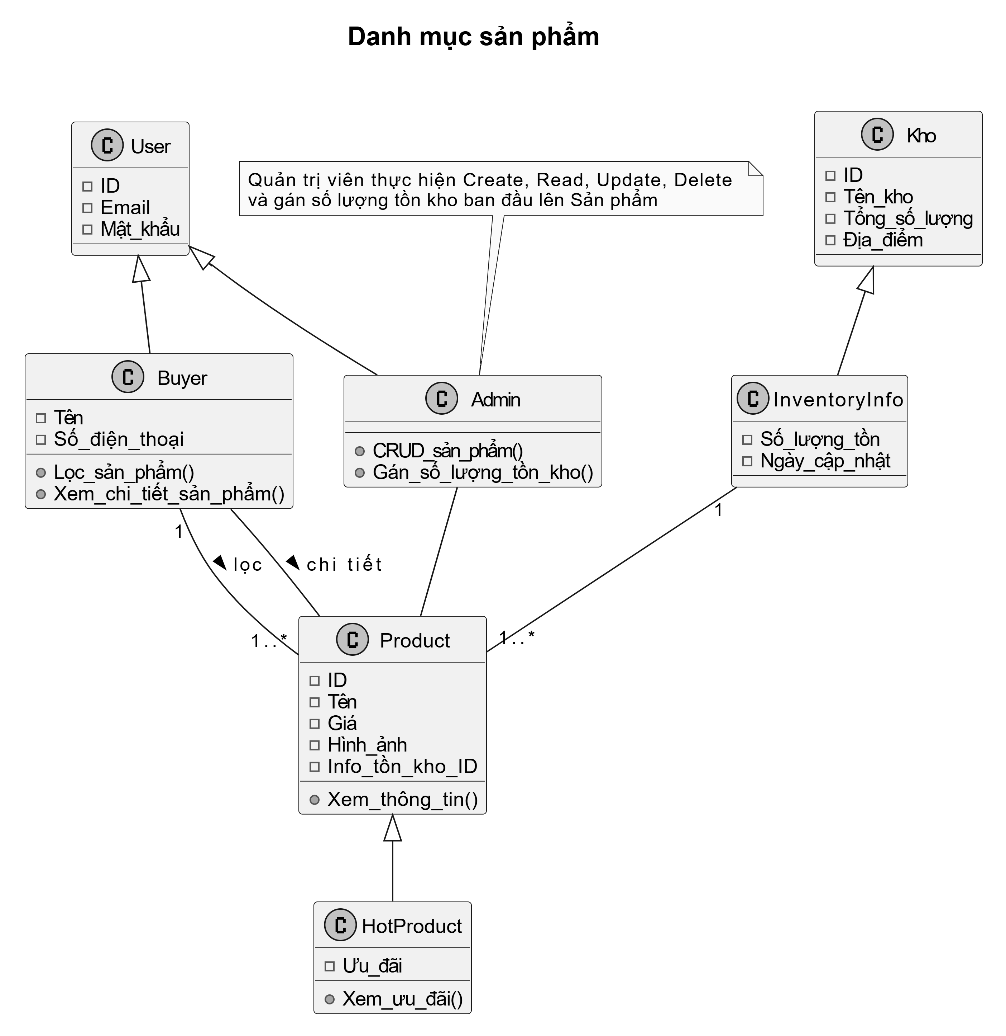
**b. Events Bus (In memory)**

* Đảm bảo giao tiếp bất đồng bộ giữa các component.

**c. External Payment Gateway**

* Dịch vụ bên ngoài (VNPay, PayPal).
* Payment gọi API để xác thực và xử lý giao dịch.

### 2.4. Sơ đồ C4: Implementation



## 1. Nhóm thực thể chính

### User (Người dùng)

* Thuộc **module Kiểm soát truy cập**.
* Thuộc tính:  
  + ID: định danh người dùng.
  + Email: dùng để đăng nhập.
  + Mật\_khẩu: thông tin bảo mật.
* Đây là lớp cha, đóng vai trò **trừu tượng** cho các loại người dùng khác.

### Buyer (Người mua)

* **Kế thừa từ User**.
* Thuộc tính:  
  + Tên, Số\_điện\_thoại.
* Phương thức:  
  + Lọc\_sản\_phẩm(): lọc sản phẩm theo tiêu chí.
  + Xem\_chi\_tiết\_sản\_phẩm(): xem thông tin chi tiết sản phẩm.
* Vai trò: đại diện cho khách hàng mua laptop.

### Admin (Quản trị viên)

* **Kế thừa từ User**.
* Phương thức:  
  + CRUD\_sản\_phẩm(): thực hiện thêm, sửa, xóa, xem sản phẩm.
  + Gán\_số\_lượng\_tồn\_kho(): gán số lượng tồn ban đầu cho sản phẩm.
* Vai trò: quản lý sản phẩm và số lượng tồn trong kho.

## 2. Nhóm thực thể về kho và sản phẩm

### Kho (Warehouse)

* Thuộc **module Tồn kho**.
* Thuộc tính:  
  + ID, Tên\_kho, Tổng\_số\_lượng, Địa\_điểm.
* Là lớp cha cho các thông tin chi tiết tồn kho.

### InventoryInfo (Thông tin tồn kho)

* **Kế thừa từ Kho**.
* Thuộc tính:  
  + Số\_lượng\_tồn: số lượng hiện tại.
  + Ngày\_cập\_nhật: ngày cập nhật tồn kho gần nhất.
* Vai trò: cung cấp thông tin tồn kho cụ thể cho từng sản phẩm.

### Product (Sản phẩm)

* Thuộc tính:  
  + ID, Tên, Giá, Hình\_ảnh, Info\_tồn\_kho\_ID.
* Phương thức:  
  + Xem\_thông\_tin(): hiển thị chi tiết sản phẩm.
* Vai trò: đối tượng trung tâm mà cả Buyer và Admin đều thao tác.

### HotProduct (Sản phẩm hot)

* **Kế thừa từ Product**.
* Thuộc tính:  
  + Ưu\_đãi.
* Phương thức:  
  + Xem\_ưu\_đãi().
* Vai trò: thể hiện sản phẩm đặc biệt, có khuyến mãi/ưu đãi.

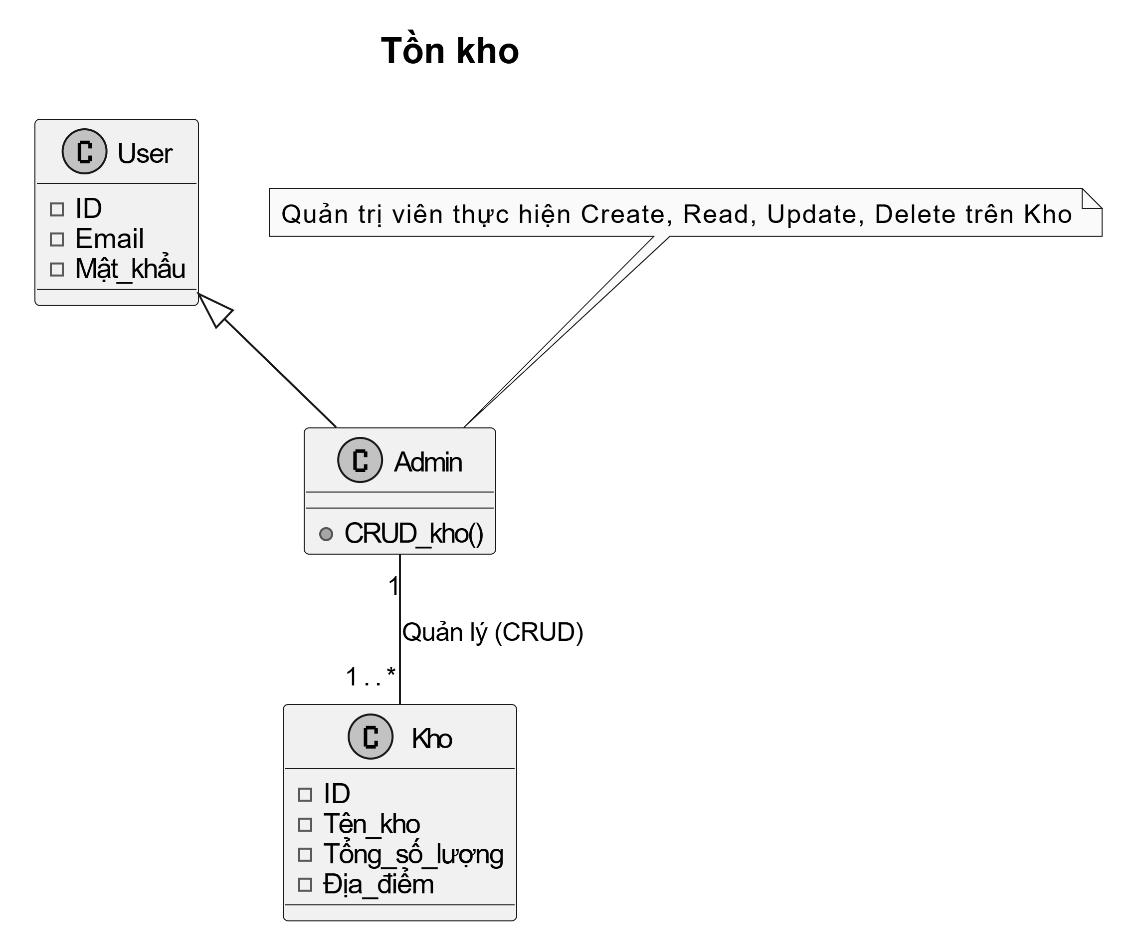
## 3. Quan hệ giữa các lớp

* **User <|-- Buyer** và **User <|-- Admin**: thể hiện kế thừa.
* **Kho <|-- InventoryInfo**: kế thừa để chuyên biệt hóa thông tin kho.
* **Product <|-- HotProduct**: kế thừa để mở rộng sản phẩm thường thành sản phẩm hot.
* **InventoryInfo "1" -- "1..\*" Product**: một thông tin tồn kho gắn với nhiều sản phẩm.
* **Buyer "1" -- "1..\*" Product**: Buyer có thể lọc nhiều sản phẩm.
* **Buyer -- Product (chi tiết)**: Buyer có thể xem chi tiết sản phẩm.
* **Admin -- Product**: Admin quản lý sản phẩm (CRUD và gán số lượng tồn).

## 4. Ý nghĩa sơ đồ

Sơ đồ này cho thấy:

* **User** là thực thể gốc, chia thành **Buyer** và **Admin** với vai trò khác nhau.
* **Product** là trung tâm, kết nối với cả Buyer (xem, lọc) và Admin (quản lý).
* **Kho** và **InventoryInfo** gắn liền với sản phẩm để quản lý tồn kho.
* Các mối quan hệ kế thừa và kết nối giúp hệ thống vừa có tính **tái sử dụng** (User chung), vừa có tính **mở rộng** (HotProduct từ Product).



## 1. Thực thể chính

### User (Người dùng)

* Nằm trong **module Kiểm soát truy cập**.
* Thuộc tính:  
  + ID: định danh duy nhất cho mỗi người dùng.
  + Email: thông tin đăng nhập.
  + Mật\_khẩu: dùng để xác thực tài khoản.
* Đây là lớp cha, từ đó sinh ra **Admin**.

### Admin (Quản trị viên)

* **Kế thừa từ User**.
* Phương thức:  
  + CRUD\_kho(): thực hiện các thao tác cơ bản với kho:  
    - **C**reate: Tạo kho mới.
    - **R**ead: Xem thông tin kho.
    - **U**pdate: Cập nhật thông tin kho.
    - **D**elete: Xóa kho.

### Kho (Inventory/Warehouse)

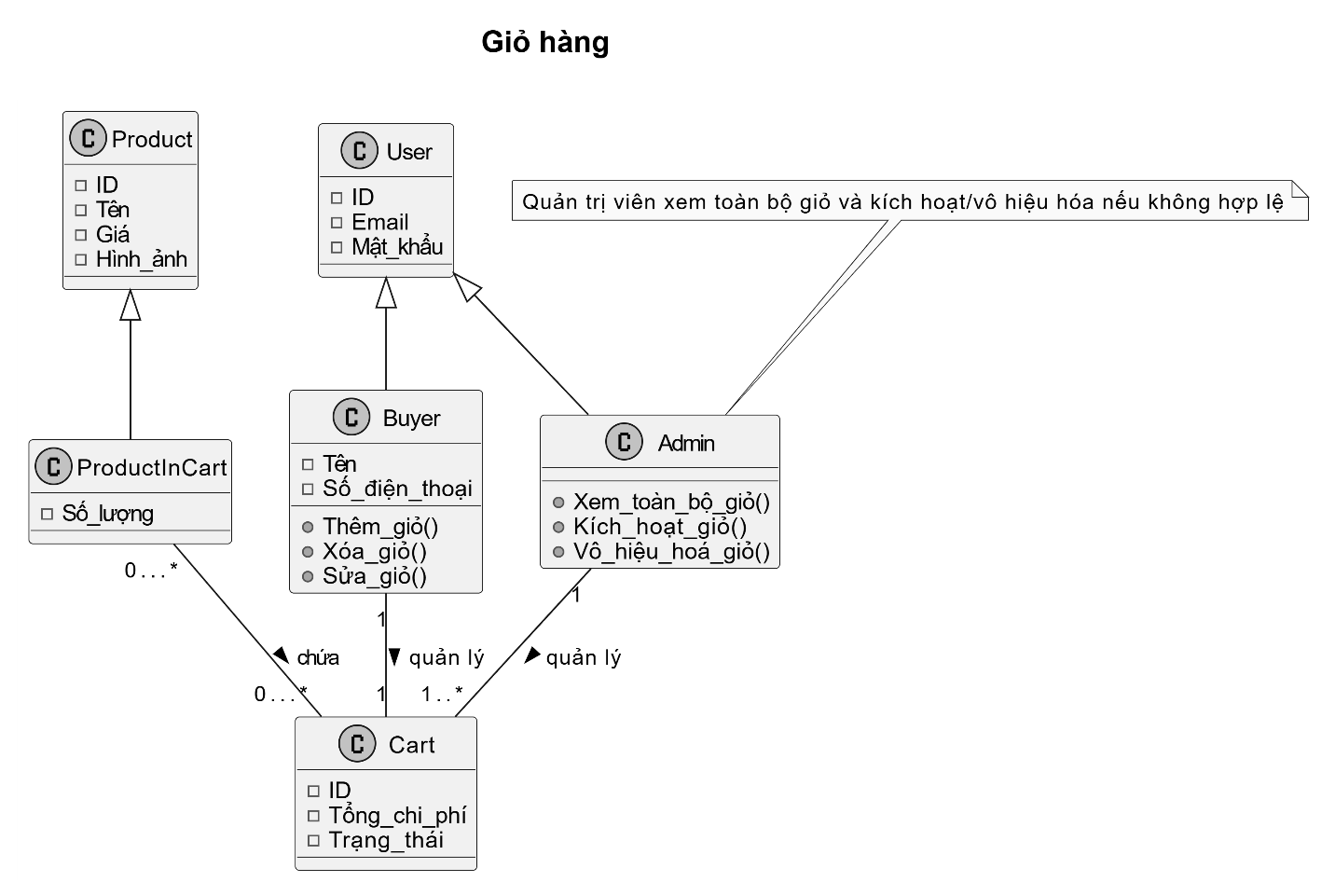
* Thuộc tính:  
  + ID: mã định danh của kho.
  + Tên\_kho: tên gọi của kho.
  + Tổng\_số\_lượng: tổng số lượng sản phẩm/đơn vị đang có trong kho.
  + Địa\_điểm: vị trí kho hàng.
* Đây là thực thể được quản trị viên thao tác trực tiếp.

## 2. Quan hệ giữa các lớp

* **User <|-- Admin**: Admin là một loại người dùng, được kế thừa từ lớp User.
* **Admin "1" -- "1..\*" Kho**: Một quản trị viên có thể quản lý nhiều kho (ít nhất một).
* Mối quan hệ được gắn nhãn: **Quản lý (CRUD)** để nhấn mạnh rằng Admin có quyền đầy đủ thao tác trên kho.

## 3. Ý nghĩa sơ đồ

* Sơ đồ mô tả **chức năng quản lý kho** trong hệ thống:  
  + **User** là cơ sở cho các loại người dùng khác nhau.
  + **Admin** có quyền cao nhất với kho, đảm nhận các thao tác CRUD.
  + **Kho** là tài nguyên trọng tâm cần quản lý.
* Ghi chú ở trên Admin làm rõ chức năng CRUD mà quản trị viên có thể thực hiện.



**1. Nhóm thực thể chính**

### User (Người dùng)

* Nằm trong **module Kiểm soát truy cập**.
* Thuộc tính:  
  + ID: định danh người dùng.
  + Email: dùng để đăng nhập.
  + Mật\_khẩu: thông tin bảo mật.
* Là lớp cha, từ đó sinh ra hai vai trò chính: **Buyer** và **Admin**.

### Buyer (Người mua)

* **Kế thừa từ User**.
* Thuộc tính:  
  + Tên: tên người mua.
  + Số\_điện\_thoại: thông tin liên hệ.
* Phương thức:  
  + Thêm\_giỏ(): thêm sản phẩm vào giỏ.
  + Xóa\_giỏ(): xóa sản phẩm trong giỏ.
  + Sửa\_giỏ(): thay đổi số lượng sản phẩm trong giỏ.
* Vai trò: khách hàng thao tác trực tiếp với giỏ hàng.

### Admin (Quản trị viên)

* **Kế thừa từ User**.
* Phương thức:  
  + Xem\_toàn\_bộ\_giỏ(): xem tất cả giỏ hàng trong hệ thống.
  + Kích\_hoạt\_giỏ(): cho phép giỏ hoạt động.
  + Vô\_hiệu\_hoá\_giỏ(): tạm ngưng hoặc xóa giỏ không hợp lệ.
* Vai trò: giám sát và quản lý các giỏ hàng.

## 2. Nhóm thực thể sản phẩm và giỏ

### Product (Sản phẩm)

* Nằm trong **module Danh mục sản phẩm**.
* Thuộc tính:  
  + ID, Tên, Giá, Hình\_ảnh.
* Là lớp cơ bản cho mọi loại sản phẩm.

### ProductInCart (Sản phẩm trong giỏ)

* **Kế thừa từ Product**.
* Thuộc tính:  
  + Số\_lượng: số lượng sản phẩm người mua chọn.
* Vai trò: cụ thể hóa sản phẩm khi nó đã được đưa vào giỏ.

### Cart (Giỏ hàng)

* Thuộc tính:  
  + ID: mã giỏ hàng.
  + Tổng\_chi\_phí: tổng giá trị sản phẩm trong giỏ.
  + Trạng\_thái: cho biết giỏ đang hoạt động, bị vô hiệu, hay đã thanh toán.
* Vai trò: trung tâm chứa các sản phẩm được chọn bởi người mua.

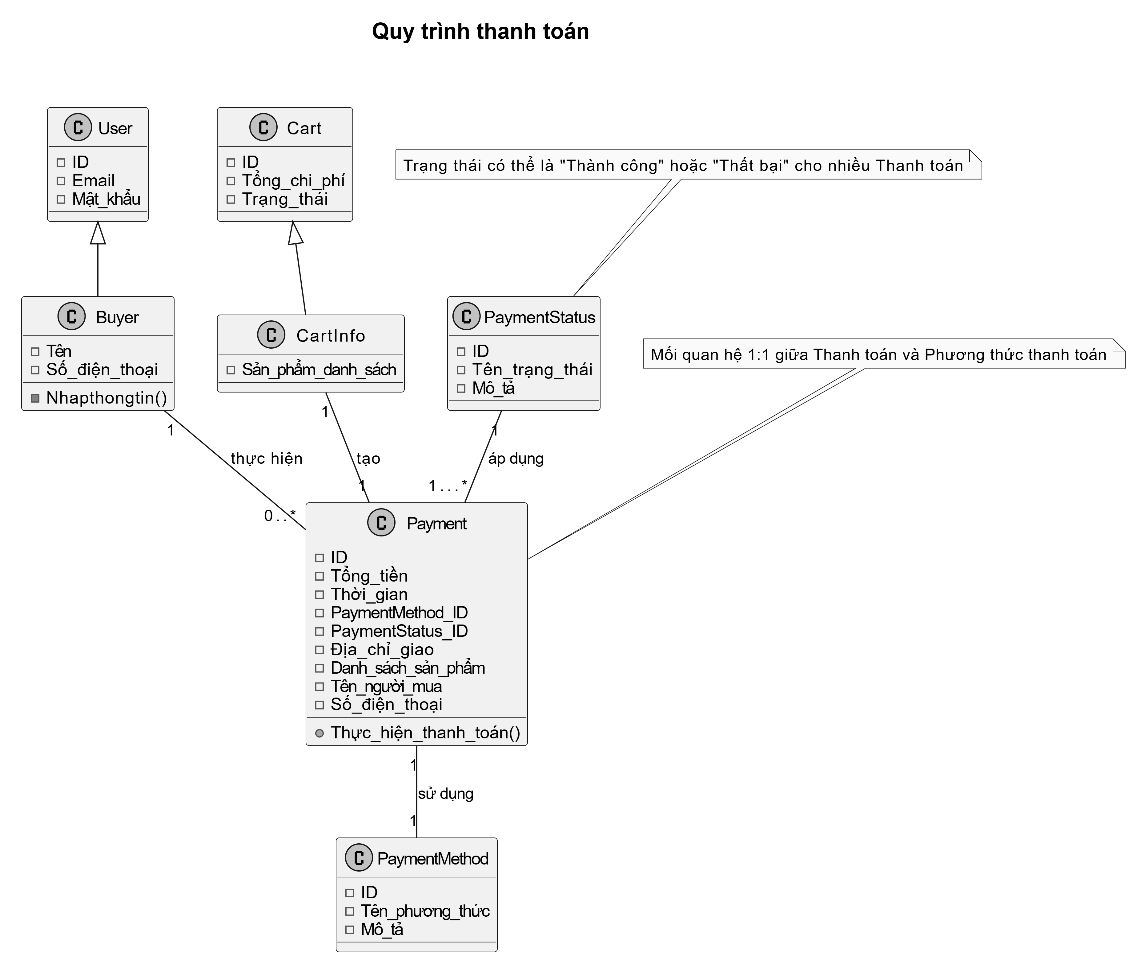
## 3. Quan hệ giữa các lớp

* **Product <|-- ProductInCart**: sản phẩm trong giỏ là một loại mở rộng của sản phẩm.
* **User <|-- Buyer**, **User <|-- Admin**: Buyer và Admin đều kế thừa từ User.
* **ProductInCart "0..*" -- "0..*" Cart**: một giỏ chứa nhiều sản phẩm, và một sản phẩm có thể nằm trong nhiều giỏ.
* **Buyer "1" -- "1" Cart**: một người mua gắn với đúng một giỏ hàng.
* **Admin "1" -- "1..\*" Cart**: một quản trị viên có thể quản lý nhiều giỏ hàng.

## 

## 4. Ý nghĩa sơ đồ

* **Buyer** đại diện cho khách hàng: thêm, sửa, xóa giỏ hàng.
* **Admin** có quyền cao hơn: giám sát toàn bộ giỏ, kích hoạt/vô hiệu hóa khi cần.
* **Cart** là trung tâm kết nối Buyer và các sản phẩm.
* **ProductInCart** cho thấy sự khác biệt giữa “sản phẩm” (Product) và “sản phẩm được thêm vào giỏ” (có số lượng cụ thể).



## 1. Thực thể chính

### User (Người dùng)

* Thuộc module **Kiểm soát truy cập**.
* Thuộc tính:  
  + ID: định danh duy nhất.
  + Email: email đăng nhập.
  + Mật\_khẩu: thông tin bảo mật.
* Đây là lớp cơ sở cho các loại người dùng.

### Buyer (Người mua)

* **Kế thừa từ User**.
* Thuộc tính:  
  + Tên: tên của người mua.
  + Số\_điện\_thoại: liên hệ.
* Phương thức:  
  + Nhapthongtin(): nhập thông tin cần thiết để đặt hàng/thanh toán.

### Cart (Giỏ hàng)

* Thuộc module **Giỏ hàng**.
* Thuộc tính:  
  + ID: mã giỏ.
  + Tổng\_chi\_phí: tổng giá trị sản phẩm trong giỏ.
  + Trạng\_thái: trạng thái giỏ (mở, chờ thanh toán, đã thanh toán).

### CartInfo (Thông tin giỏ)

* **Kế thừa từ Cart**.
* Thuộc tính:  
  + Sản\_phẩm\_danh\_sách: chứa danh sách sản phẩm trong giỏ.

### Payment (Thanh toán)

* Là trung tâm của module thanh toán.
* Thuộc tính:  
  + ID: mã thanh toán.
  + Tổng\_tiền: số tiền cần trả.
  + Thời\_gian: thời điểm giao dịch.
  + PaymentMethod\_ID: phương thức thanh toán (liên kết với PaymentMethod).
  + PaymentStatus\_ID: trạng thái thanh toán (liên kết với PaymentStatus).
  + Địa\_chỉ\_giao: nơi giao hàng.
  + Danh\_sách\_sản\_phẩm: sản phẩm được thanh toán.
  + Tên\_người\_mua: tên người thực hiện thanh toán.
  + Số\_điện\_thoại: số liên hệ khi giao hàng.
* Phương thức:  
  + Thực\_hiện\_thanh\_toán(): tiến hành xử lý thanh toán.

### PaymentMethod (Phương thức thanh toán)

* Thuộc tính:  
  + ID: mã phương thức.
  + Tên\_phương\_thức: ví dụ: thẻ tín dụng, ví điện tử, COD.
  + Mô\_tả: chi tiết thêm.

### PaymentStatus (Trạng thái thanh toán)

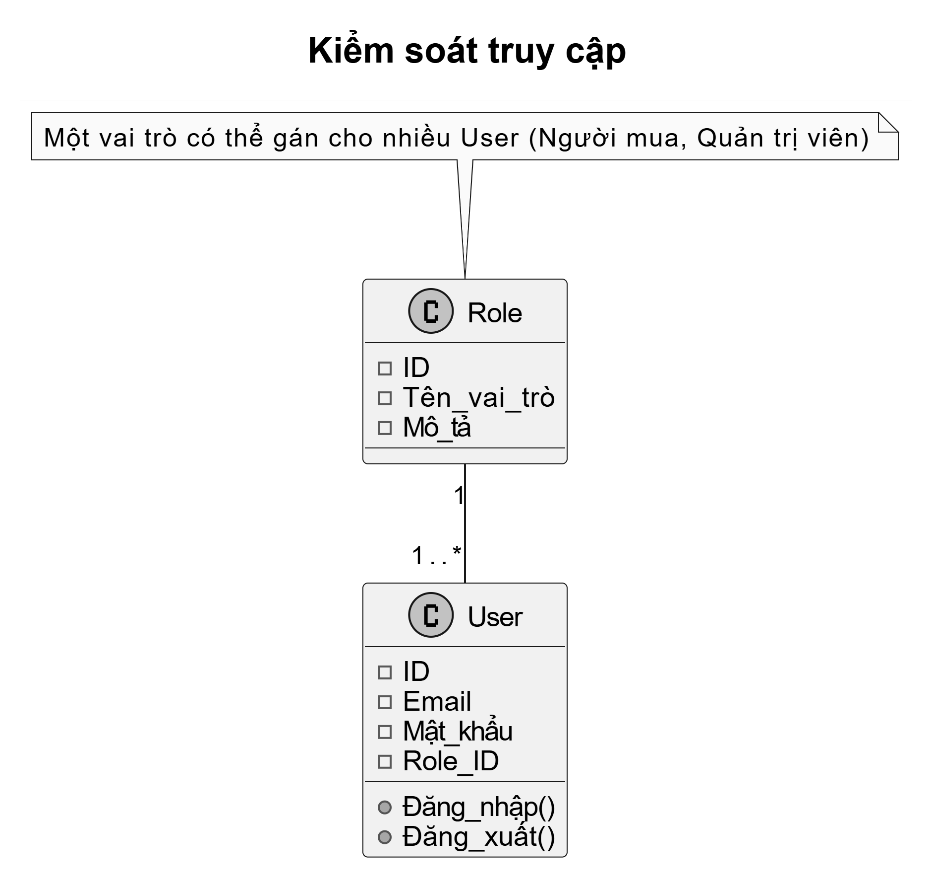
* Thuộc tính:  
  + ID: mã trạng thái.
  + Tên\_trạng\_thái: ví dụ: Thành công, Thất bại, Đang chờ.
  + Mô\_tả: diễn giải chi tiết.

## 2. Mối quan hệ giữa các lớp

* **Buyer "1" -- "0..\*" Payment**: Một người mua có thể thực hiện nhiều giao dịch thanh toán.
* **CartInfo "1" -- "1" Payment**: Mỗi thanh toán được tạo từ một giỏ hàng.
* **Payment "1" -- "1" PaymentMethod**: Một giao dịch thanh toán chỉ sử dụng duy nhất một phương thức.
* **PaymentStatus "1" -- "1..\*" Payment**: Một trạng thái có thể áp dụng cho nhiều giao dịch khác nhau (ví dụ: nhiều giao dịch cùng ở trạng thái “Thành công”).

## 3. Ý nghĩa sơ đồ

* Sơ đồ thể hiện rõ luồng **từ người mua → giỏ hàng → thanh toán → phương thức + trạng thái**.
* Nhấn mạnh tính **kế thừa** (User → Buyer, Cart → CartInfo) và **ràng buộc quan hệ 1-1, 1-nhiều** giữa các thực thể.
* Hệ thống đảm bảo:  
  + Mỗi giao dịch luôn gắn với **một giỏ hàng**.
  + Thanh toán chỉ dùng **một phương thức** tại một thời điểm.
  + Trạng thái thanh toán được tái sử dụng, không cần tạo mới cho từng giao dịch.
* Nhờ vậy, mô hình vừa rõ ràng về **vai trò thực thể**, vừa đảm bảo **toàn vẹn dữ liệu** trong quy trình thanh toán.



## 1. Thực thể chính

### User (Người dùng)

* Đại diện cho mọi đối tượng đăng nhập vào hệ thống.
* Thuộc tính:  
  + ID: định danh duy nhất cho người dùng.
  + Email: dùng để đăng nhập hoặc liên lạc.
  + Mật\_khẩu: thông tin bảo mật cho việc xác thực.
  + Role\_ID: liên kết đến vai trò mà người dùng đảm nhận.
* Phương thức:  
  + Đăng\_nhập(): xác thực thông tin và cho phép truy cập.
  + Đăng\_xuất(): kết thúc phiên làm việc của người dùng.

### Role (Vai trò)

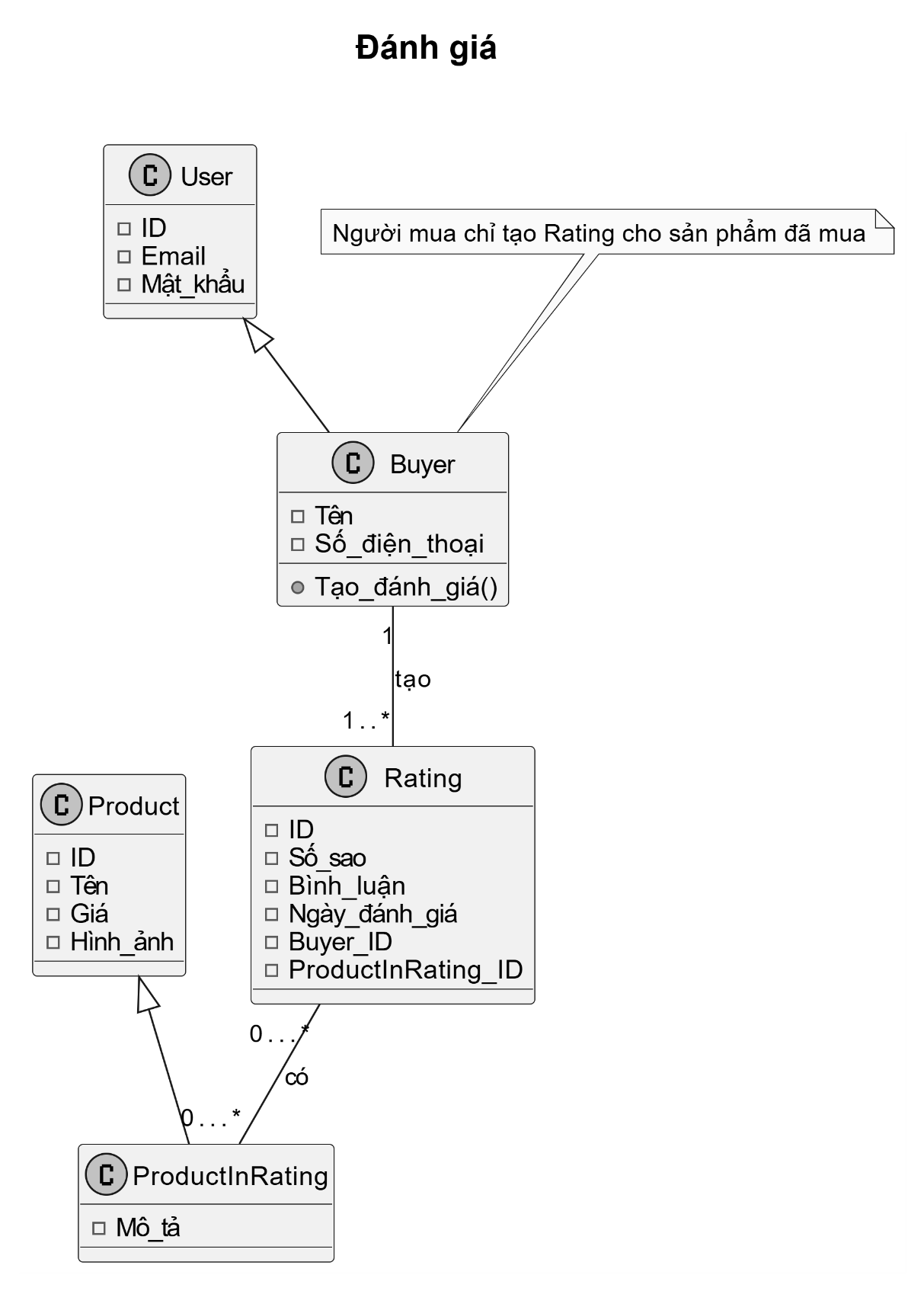
* Đại diện cho **nhóm quyền hạn** trong hệ thống.
* Thuộc tính:  
  + ID: định danh vai trò.
  + Tên\_vai\_trò: tên gọi, ví dụ: *Buyer*, *Admin*.
  + Mô\_tả: diễn giải chi tiết về vai trò.

## 2. Mối quan hệ

* **Role "1" -- "1..\*" User**
  + Một vai trò có thể gán cho nhiều người dùng khác nhau.
  + Ví dụ:  
    - Vai trò **Buyer** có thể được gán cho hàng nghìn người mua.
    - Vai trò **Admin** có thể gán cho một vài quản trị viên.
  + Ngược lại, **mỗi User chỉ có duy nhất một Role** trong mô hình này (thông qua Role\_ID).

## 3. Ý nghĩa sơ đồ

* Sơ đồ này nằm trong **module kiểm soát truy cập (Access Control)**.
* Nó giúp quản lý hệ thống theo hướng **Role-based Access Control (RBAC)**:  
  + Người dùng không được gắn trực tiếp quyền, mà gián tiếp thông qua **vai trò**.
  + Quản lý dễ dàng hơn: chỉ cần thay đổi quyền trong Role, tất cả User thuộc Role đó sẽ thay đổi theo.
* Đây là **thiết kế chuẩn** trong nhiều hệ thống web/app hiện đại.



## 1. Thực thể chính

### Product (Sản phẩm)

* Thuộc module **Danh mục sản phẩm**.
* Thuộc tính:  
  + ID: mã sản phẩm duy nhất.
  + Tên: tên sản phẩm (ví dụ: Laptop Dell XPS).
  + Giá: giá bán.
  + Hình\_ảnh: hình minh họa cho sản phẩm.

### User (Người dùng)

* Thuộc module **Kiểm soát truy cập**.
* Thuộc tính:  
  + ID: định danh duy nhất cho người dùng.
  + Email: thông tin đăng nhập/liên lạc.
  + Mật\_khẩu: bảo mật.

### Buyer (Người mua)

* **Kế thừa từ User**.
* Thuộc tính:  
  + Tên: tên đầy đủ của người mua.
  + Số\_điện\_thoại: liên lạc.
* Phương thức:  
  + Tạo\_đánh\_giá(): thực hiện việc viết đánh giá cho sản phẩm.

### ProductInRating (Sản phẩm trong đánh giá)

* **Kế thừa từ Product**.
* Thuộc tính:  
  + Mô\_tả: phần mô tả chi tiết sản phẩm trong ngữ cảnh đánh giá (có thể khác với mô tả chính thức).

### Rating (Đánh giá)

* Đại diện cho một đánh giá mà người mua để lại.
* Thuộc tính:  
  + ID: mã định danh đánh giá.
  + Số\_sao: số sao (ví dụ: 1–5).
  + Bình\_luận: nội dung bình luận.
  + Ngày\_đánh\_giá: thời điểm viết đánh giá.
  + Buyer\_ID: liên kết đến người mua đã viết đánh giá.
  + ProductInRating\_ID: liên kết đến sản phẩm được đánh giá.

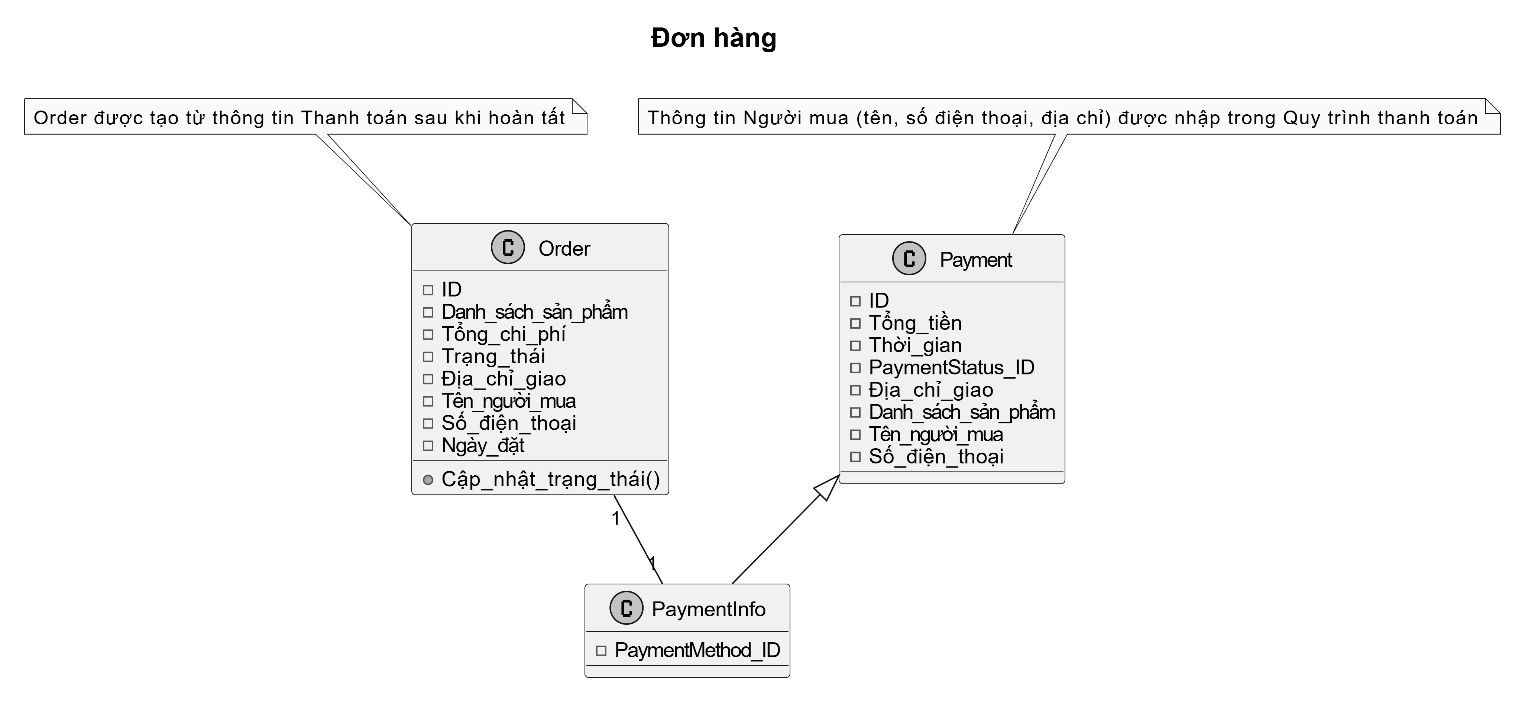
## 2. Mối quan hệ

* **Buyer "1" -- "1..\*" Rating**:  
  + Mỗi người mua có thể tạo nhiều đánh giá cho các sản phẩm khác nhau.
  + Nhưng một đánh giá chỉ thuộc về duy nhất một người mua.
* **Rating "0..*" -- "0..*" ProductInRating**:  
  + Một sản phẩm có thể nhận nhiều đánh giá từ các người mua khác nhau.
  + Ngược lại, một đánh giá gắn với một sản phẩm cụ thể.

## 

## 3. Ý nghĩa sơ đồ

* Sơ đồ mô tả module **Đánh giá sản phẩm**, trong đó:  
  + **Buyer** đóng vai trò chủ động tạo đánh giá.
  + **Rating** là đối tượng trung gian chứa thông tin sao và bình luận.
  + **ProductInRating** cho thấy rằng việc đánh giá gắn liền với sản phẩm trong ngữ cảnh đánh giá (kế thừa từ Product).
* Mô hình giúp quản lý được:  
  + Ai đánh giá.
  + Sản phẩm nào được đánh giá.
  + Nội dung và chất lượng đánh giá.
* Nhờ mối quan hệ rõ ràng, hệ thống có thể dễ dàng thống kê mức độ hài lòng của khách hàng, lọc đánh giá theo sản phẩm, và ngăn chặn đánh giá giả mạo (vì Buyer phải mua hàng trước).



## 1. Thực thể chính

### Payment (Thanh toán)

* Thuộc module **Quy trình thanh toán**.
* Thuộc tính:  
  + ID: mã định danh giao dịch thanh toán.
  + Tổng\_tiền: số tiền phải thanh toán.
  + Thời\_gian: thời điểm giao dịch diễn ra.
  + PaymentStatus\_ID: trạng thái thanh toán (thành công, thất bại, đang xử lý).
  + Địa\_chỉ\_giao: nơi nhận hàng.
  + Danh\_sách\_sản\_phẩm: danh sách sản phẩm được thanh toán.
  + Tên\_người\_mua: thông tin người thực hiện giao dịch.
  + Số\_điện\_thoại: số điện thoại liên hệ khi giao hàng.

### PaymentInfo (Thông tin thanh toán)

* **Kế thừa từ Payment**.
* Thuộc tính bổ sung:  
  + PaymentMethod\_ID: phương thức thanh toán được chọn (VD: thẻ tín dụng, COD, ví điện tử).
* Đây là lớp mở rộng để lưu **chi tiết phương thức thanh toán**, ngoài những thông tin cơ bản của Payment.

### Order (Đơn hàng)

* Đại diện cho đơn hàng được tạo sau khi thanh toán thành công.
* Thuộc tính:  
  + ID: mã định danh đơn hàng.
  + Danh\_sách\_sản\_phẩm: các sản phẩm thuộc đơn hàng.
  + Tổng\_chi\_phí: tổng số tiền của đơn hàng (có thể trùng với Payment).
  + Trạng\_thái: trạng thái đơn hàng (đang xử lý, đã giao, bị hủy).
  + Địa\_chỉ\_giao: địa chỉ giao hàng (lấy từ Payment).
  + Tên\_người\_mua: tên khách hàng.
  + Số\_điện\_thoại: số điện thoại liên hệ.
  + Ngày\_đặt: ngày mà đơn hàng được tạo.
* Phương thức:  
  + Cập\_nhật\_trạng\_thái(): thay đổi trạng thái đơn hàng trong quá trình xử lý.

## 2. Mối quan hệ

* **Order "1" -- "1" PaymentInfo**:  
  + Mỗi **Order** được sinh ra từ đúng một **PaymentInfo** (nghĩa là một giao dịch thanh toán).
  + Ngược lại, một PaymentInfo chỉ tạo ra duy nhất một Order tương ứng.
  + Đây là mối quan hệ **1-1**, bảo đảm rằng một giao dịch thanh toán chỉ tạo ra đúng một đơn hàng, và mỗi đơn hàng phải có một giao dịch hợp lệ.

## 3. Ý nghĩa sơ đồ

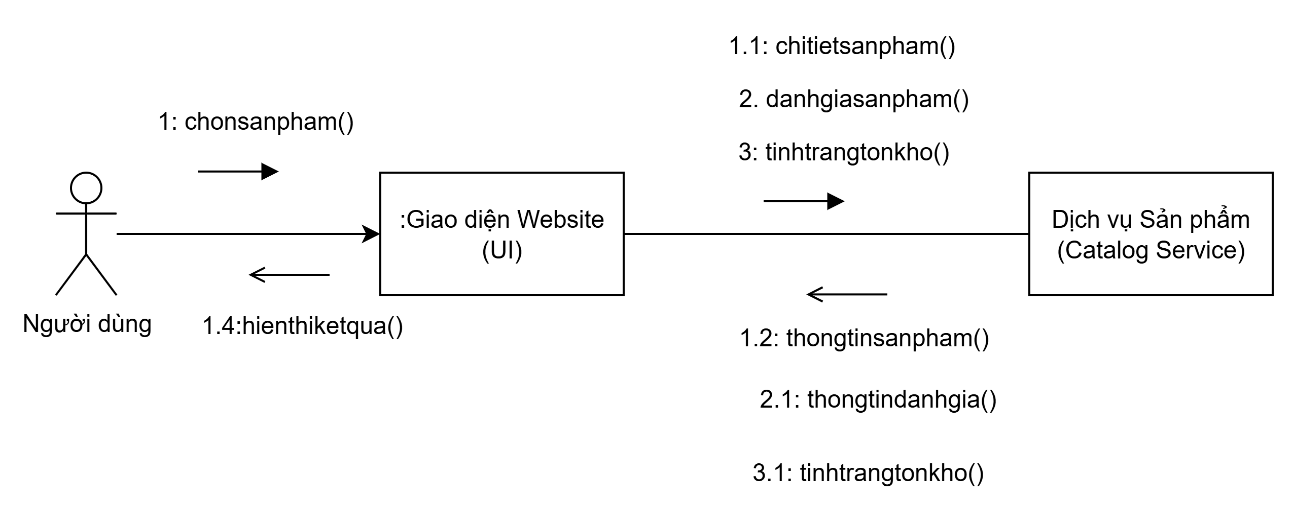
* Sơ đồ này thể hiện **mối quan hệ trực tiếp giữa Thanh toán và Đơn hàng**.
* Các điểm chính:  
  + Quy trình thanh toán diễn ra trước, khi thành công thì sinh ra **Order**.
  + Thông tin người mua và địa chỉ giao hàng **được nhập ở bước thanh toán**, sau đó được **chuyển nguyên sang Order**.
  + Order đóng vai trò theo dõi quá trình sau thanh toán: vận chuyển, cập nhật trạng thái, quản lý sản phẩm.
* Mô hình này giúp đảm bảo:  
  + **Dữ liệu đồng nhất** (Order và Payment chia sẻ cùng thông tin người mua, sản phẩm, địa chỉ).
  + **Quy trình nghiệp vụ rõ ràng**: Payment thành công → Order được tạo → Quản lý đơn hàng tiếp tục xử lý.

## 3.Communication View

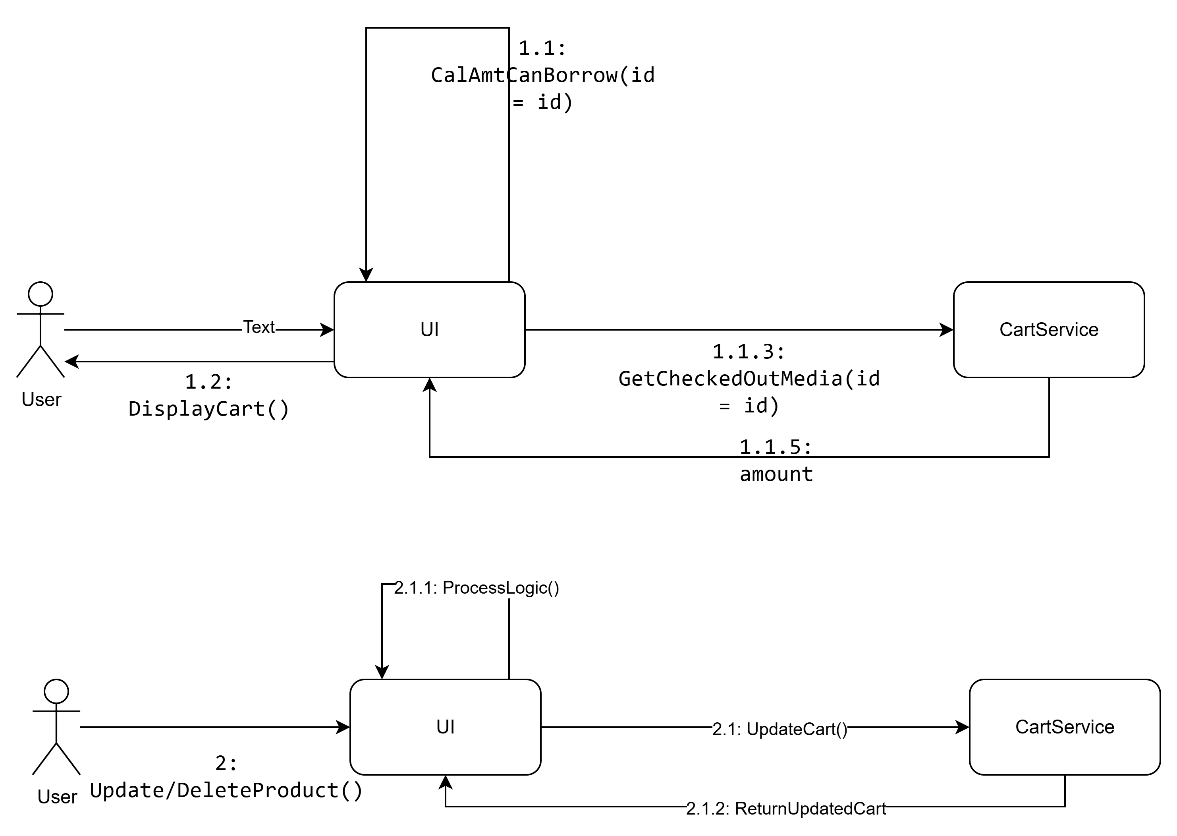
### 3.1. Xem và Lọc/Sắp xếp Sản phẩm

****

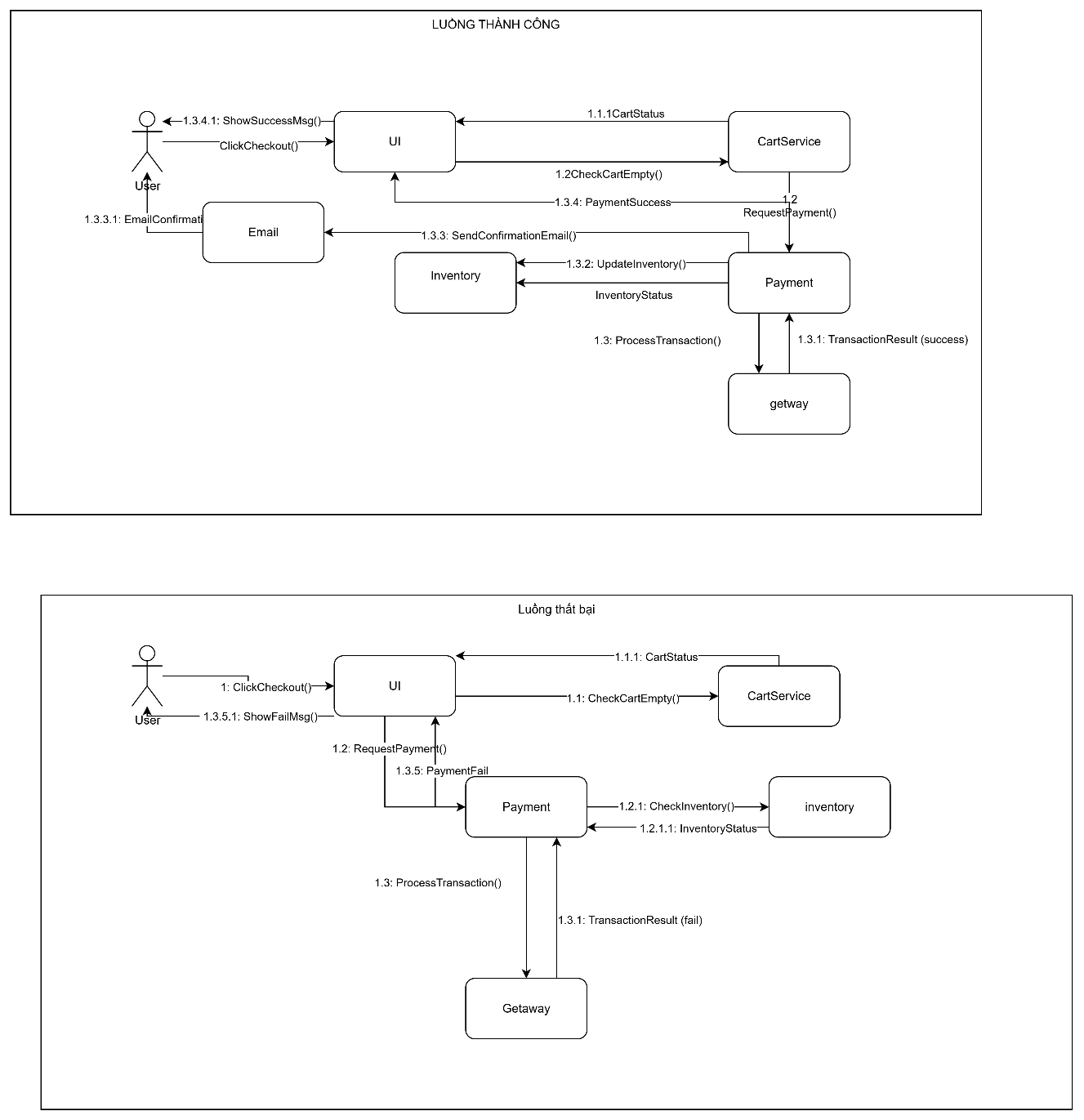
**3.2. Thêm sản phẩm vào giỏ hang**



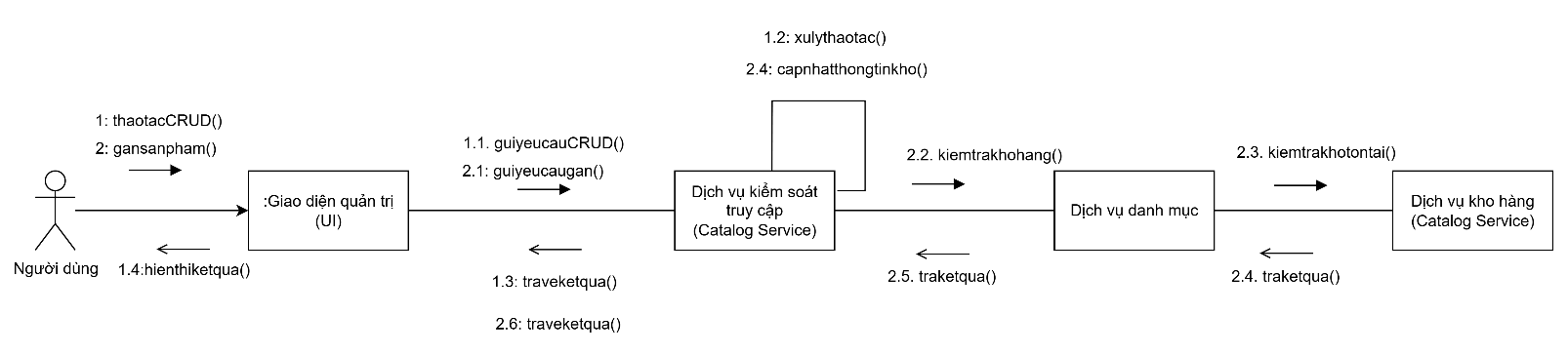
### 3.3. Quản Lý Giỏ Hàng



### 3.4. Thanh toán đơn hàng



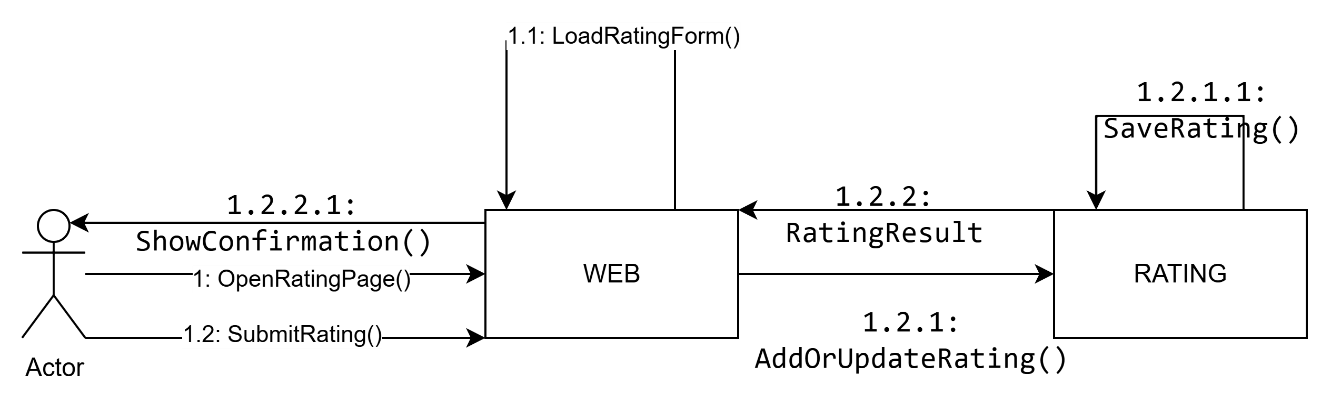
### 3.5. Quản lý sản phẩm ( Admin)



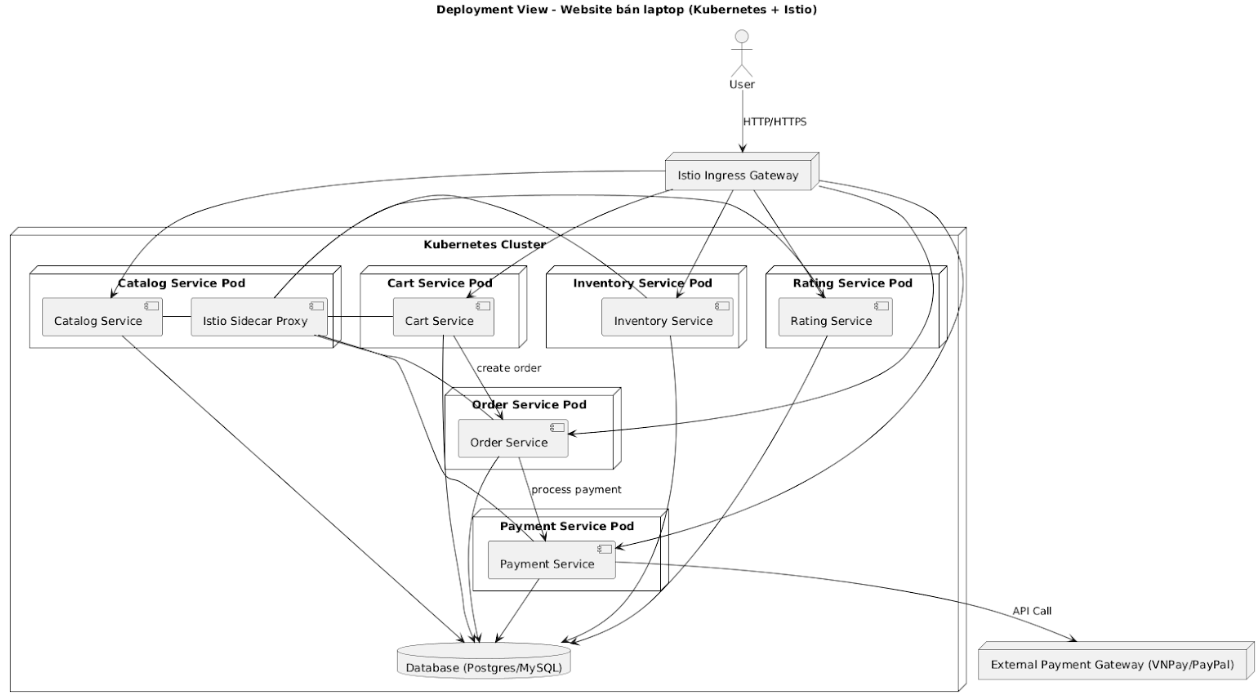
### 3.6. Xem Lịch sử Đơn hàng



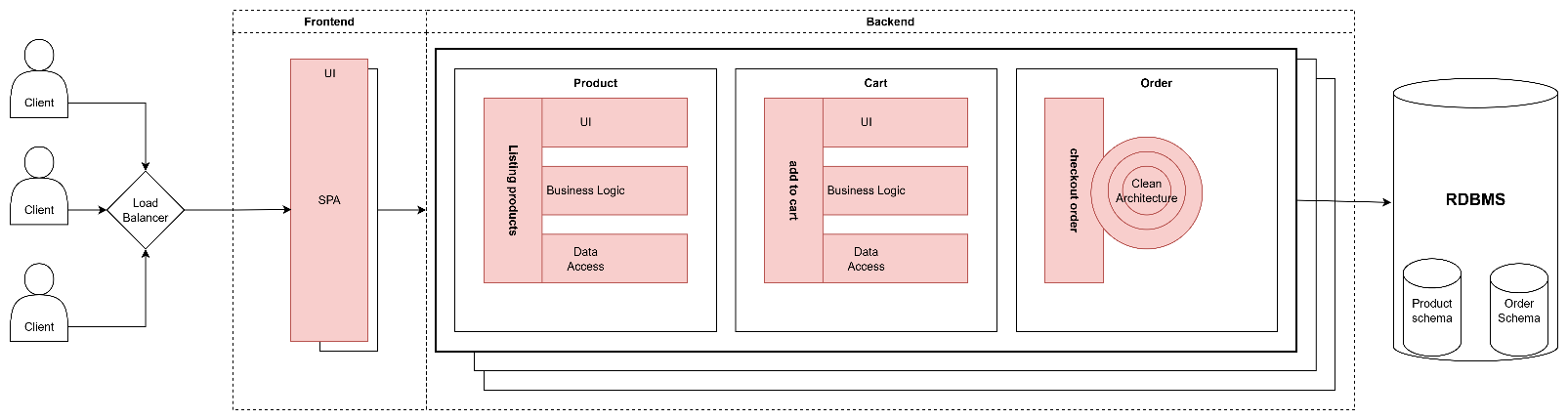
### 3.7. Đánh giá sản phẩm



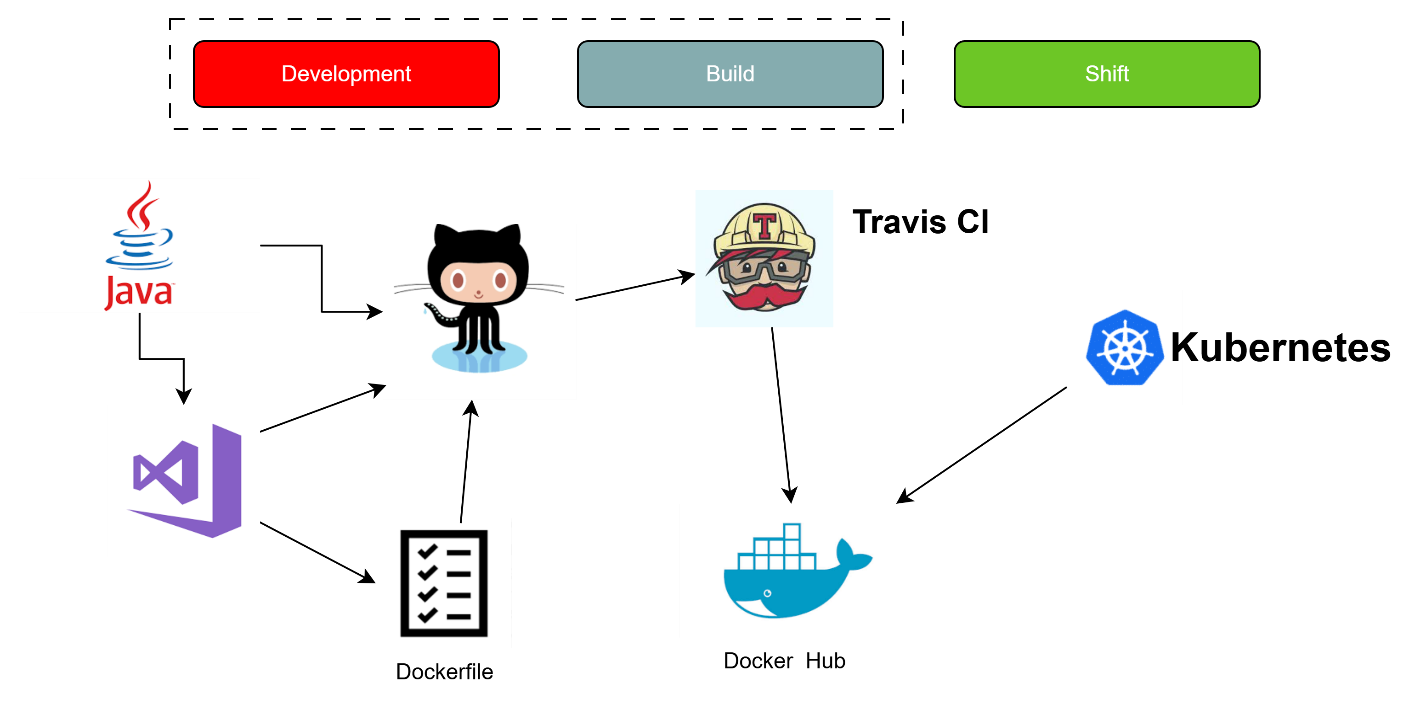
## 4. Deployment View



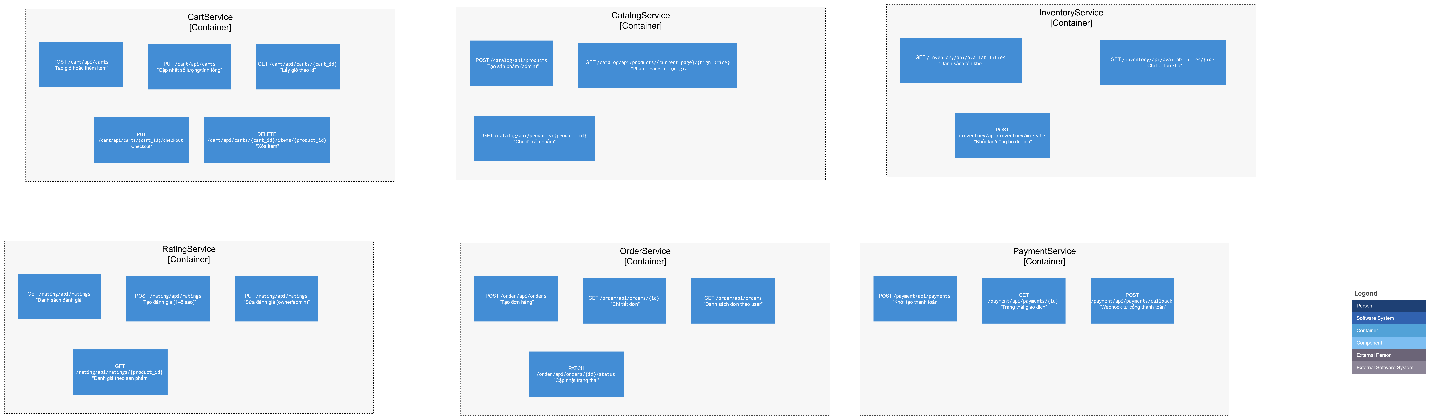
## 5. Sơ đồ kiến trúc SPA (Single Page Application):



## 6. Sơ đồ triển khai CI/CD:



## 7. Sơ đồ API của hệ thống:



## 8. sơ đồ xử lý 1 request:

