ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

GVHD: TS. Đỗ Bá Lâm

Chủ đề thực hiện:

XÂY DỰNG WEB BÁN HÀNG ONLINE SỬ DỤNG KIẾN TRÚC MICROSERVICE

Mã lớp: **162307** Nhóm: **1**

Môn học: **Công nghệ Web và dịch vụ trực tuyến** Mã môn học: **IT4409**

Các thành viên trong nhóm:

STT	Họ và tên	MSSV	Nhiệm vụ
1	Nguyễn Thế Anh	20224921	Trưởng nhóm, phân công và theo dõi tiến độ, phát triển Backend, xây dựng CI/CD
2	Hồ Lương An	20224821	Viết báo cáo, phát triển Frontend
3	Bùi Khắc Anh	20225246	Phát triển Backend, tester, phân tích yêu cầu
4	Lê Đình Hùng Anh	20224917	Làm slide, phát triển Frontend

Hà Nội, tháng 12 năm 2025

Lời nói đầu

Trong bối cảnh thương mại điện tử phát triển mạnh mẽ, việc xây dựng một hệ thống web bán hàng online hiệu quả, dễ bảo trì và mở rộng là vô cùng quan trọng. Báo cáo này trình bày quá trình thiết kế và triển khai website HUSTBuy - một nền tảng thương mại điện tử áp dụng kiến trúc Microservice.

Kiến trúc Microservice cho phép phân chia hệ thống thành các dịch vụ độc lập, giúp tăng tính linh hoạt, khả năng mở rộng và dễ dàng bảo trì. Qua đó, nhóm chúng em đã nghiên cứu và áp dụng các công nghệ hiện đại để xây dựng một giải pháp toàn diện từ phân tích, thiết kế đến triển khai.

Báo cáo bao gồm các nội dung chính: giới thiệu bài toán và mục tiêu hệ thống, phân tích và thiết kế chi tiết theo kiến trúc Microservice, các công nghệ được sử dụng, cùng với hình ảnh minh họa sản phẩm thực tế.

Nhóm xin chân thành cảm ơn thầy, TS. Đỗ Bá Lâm đã tận tình hướng dẫn trong suốt quá trình thực hiện project này.

Nhóm 1

Mục lục

Danh mục hình ảnh

Danh mục bảng biểu

Danh mục mã nguồn

1 Giới thiệu bài toán

Trong thời đại công nghệ số hiện nay, thương mại điện tử đã trở thành xu hướng không thể thiếu trong cuộc sống. Người tiêu dùng ngày càng ưu tiên mua sắm trực tuyến nhờ sự tiện lợi, đa dạng sản phẩm và khả năng so sánh giá dễ dàng. Tuy nhiên, để xây dựng một hệ thống thương mại điện tử đáp ứng được nhu cầu ngày càng cao của người dùng về hiệu năng, tính sẵn sàng và khả năng mở rộng là một thách thức lớn.

Để giải quyết vấn đề này, kiến trúc Microservice đã nổi lên như một giải pháp tối ưu. Thay vì xây dựng hệ thống theo mô hình monolithic truyền thống, kiến trúc Microservice cho phép chia nhỏ ứng dụng thành các dịch vụ độc lập, mỗi dịch vụ đảm nhận một chức năng cụ thể. Điều này mang lại nhiều lợi ích như tăng khả năng mở rộng từng phần, dễ dàng bảo trì, triển khai linh hoạt và tối ưu hóa hiệu suất.

Dự án HUSTBuy được nhóm chúng em triển khai nhằm áp dụng kiến trúc Microservice vào thực tế, xây dựng một nền tảng thương mại điện tử hoàn chỉnh với đầy đủ các tính năng từ quản lý sản phẩm, giỏ hàng, thanh toán đến xác thực người dùng và quản lý đơn hàng.

1.1 Bối cảnh và nhu cầu thực tiễn

Sự phát triển của thương mại điện tử:

Theo báo cáo của các tổ chức nghiên cứu thị trường, ngành thương mại điện tử tại Việt Nam đang tăng trưởng với tốc độ hơn 20% mỗi năm. Người tiêu dùng, đặc biệt là thế hệ trẻ, ngày càng quen thuộc với việc mua sắm trực tuyến. Điều này tạo ra nhu cầu cấp thiết về các nền tảng thương mại điện tử đáng tin cậy, nhanh chóng và dễ sử dụng.

Thách thức của kiến trúc truyền thống:

Các hệ thống thương mại điện tử truyền thống thường được xây dựng theo mô hình monolithic, nơi tất cả các chức năng được tích hợp trong một ứng dụng duy nhất. Mô hình này gặp phải nhiều hạn chế:

- Khó mở rộng khi lưu lượng truy cập tăng cao
- Một lỗi nhỏ có thể làm sập toàn bộ hệ thống
- Khó khăn trong việc áp dụng công nghệ mới cho từng module
- Thời gian triển khai và bảo trì kéo dài
- Không tận dụng được tài nguyên một cách hiệu quả

Nhu cầu áp dụng Microservice:

Kiến trúc Microservice ra đời để giải quyết các vấn đề trên bằng cách:

- Phân tách hệ thống thành các dịch vụ nhỏ, độc lập
- Mỗi dịch vụ có thể được phát triển, triển khai và mở rộng riêng biệt
- Tăng tính sẵn sàng của hệ thống nhờ cơ chế fault isolation
- Cho phép các team phát triển song song, tăng năng suất
- Dễ dàng áp dụng các công nghệ phù hợp cho từng dịch vụ

Từ những nhu cầu và thách thức thực tiễn trên, nhóm chúng em quyết định xây dựng HUSTBuy theo kiến trúc Microservice để không chỉ đáp ứng yêu cầu của đề tài mà còn nghiên cứu sâu về một xu hướng công nghệ đang được ứng dụng rộng rãi trong các doanh nghiệp lớn.

1.2 Mục tiêu hệ thống

Mục tiêu tổng quát:

Xây dựng một nền tảng thương mại điện tử hoàn chỉnh áp dụng kiến trúc Microservice, đảm bảo tính mở rộng, bảo mật và hiệu năng cao, phục vụ nhu cầu mua bán trực tuyến của cộng đồng.

Mục tiêu cụ thể:

Về mặt *chức năng*, dự án tập trung xây dựng một hệ thống quản lý sản phẩm toàn diện với đầy đủ thông tin chi tiết, hình ảnh và phân loại khoa học. Hệ thống sẽ triển khai các tính năng thương mại điện tử cốt lõi bao gồm giỏ hàng, đặt hàng trực tuyến và tích hợp các phương thức thanh toán an toàn. Bên cạnh đó, các module về xác thực và phân quyền người dùng sẽ được thiết lập chặt chẽ cùng với hệ thống quản lý đơn hàng, cho phép theo dõi trạng thái giao hàng theo thời gian thực. Để hỗ trợ công tác vận hành, dự án cũng phát triển các dashboard quản trị chuyên sâu dành riêng cho người bán và quản trị viên.

Về *kiến trúc và công nghệ*, hệ thống được định hướng phát triển dựa trên kiến trúc Microservices với các dịch vụ hoạt động độc lập, sử dụng API Gateway để quản lý và định tuyến các yêu cầu. Dự án sẽ triển khai các cơ chế hiện đại như Service Discovery, Load Balancing và áp dụng linh hoạt các design pattern phù hợp như Database per service, Distributed Caching hoặc Saga để đảm bảo tính nhất quán của dữ liệu. Ngoài ra, một hệ thống monitoring và logging tập trung sẽ được xây dựng để theo dõi sức khỏe và hiệu suất của toàn bộ các service.

Đối với *hiệu năng và bảo mật*, mục tiêu trọng tâm là đảm bảo hệ thống có khả năng xử lý tốt trong điều kiện số lượng người dùng đồng thời cao, đồng thời tối ưu hóa thời gian phản hồi và tốc độ tải trang. Các biện pháp bảo mật tiên tiến như giao thức HTTPS, cơ chế xác thực JWT và kiểm soát dữ liệu đầu vào (input validation) sẽ được áp dụng triệt để. Hệ thống cũng được thiết kế để chống lại các cuộc tấn công phổ biến như SQL Injection, XSS, CSRF và thực hiện mã hóa các thông tin nhạy cảm để bảo vệ quyền riêng tư của khách hàng.

Cuối cùng, về *trải nghiệm người dùng*, dự án hướng tới việc thiết kế giao diện thân thiện, dễ sử dụng và có tính tương thích (responsive) trên nhiều loại thiết bị và nền tảng khác nhau. Trải nghiệm mua sắm sẽ được tối ưu hóa thông qua hệ thống tìm kiếm và bộ lọc sản phẩm hiệu quả, kết hợp với các thông báo thời gian thực (real-time) về tình trạng đơn hàng, giúp nâng cao mức độ hài lòng của người dùng.

Thông qua việc đạt được các mục tiêu trên, dự án không chỉ tạo ra một sản phẩm thực tế mà còn giúp nhóm nắm vững kiến thức về kiến trúc Microservice và các công nghệ web hiện đại.

1.3 Quản lý tiến độ công việc

Nhóm đã lựa chọn áp dụng quy trình **Agile** (theo mô hình Scrum) để đảm bảo tính linh hoạt và khả năng ứng biến nhanh với các thay đổi trong quá trình phát triển dự án. Công cụ **Jira** được sử dụng làm nền tảng chính để quản lý và theo dõi tiến độ công việc một cách trực quan.

Cụ thể, nhóm thực hiện quản lý công việc thông qua các hoạt động sau:

- **Product Backlog:** Toàn bộ các yêu cầu chức năng và phi chức năng được lưu trữ tập trung trên Jira.
- **Sprint Planning:** Nhóm chia dự án thành các giai đoạn nhỏ (Sprints), mỗi giai đoạn kéo dài từ 1-2 tuần để hoàn thành một số lượng công việc nhất định.
- **Quản lý Task:** Mỗi công việc được chia thành các *Task* cụ thể gắn với từng thành viên, kèm theo thời hạn (deadline) và mức độ ưu tiên.

Tiến độ được theo dõi với các trạng thái:

- **To Do:** Các công việc đã được lập kế hoạch nhưng chưa bắt đầu.
- **In Progress:** Các công việc đang trong quá trình thực hiện.
- **In Review:** Các công việc đã xong phần code và đang được kiểm thử hoặc đợi review.
- **Done:** Các công việc đã hoàn thành.

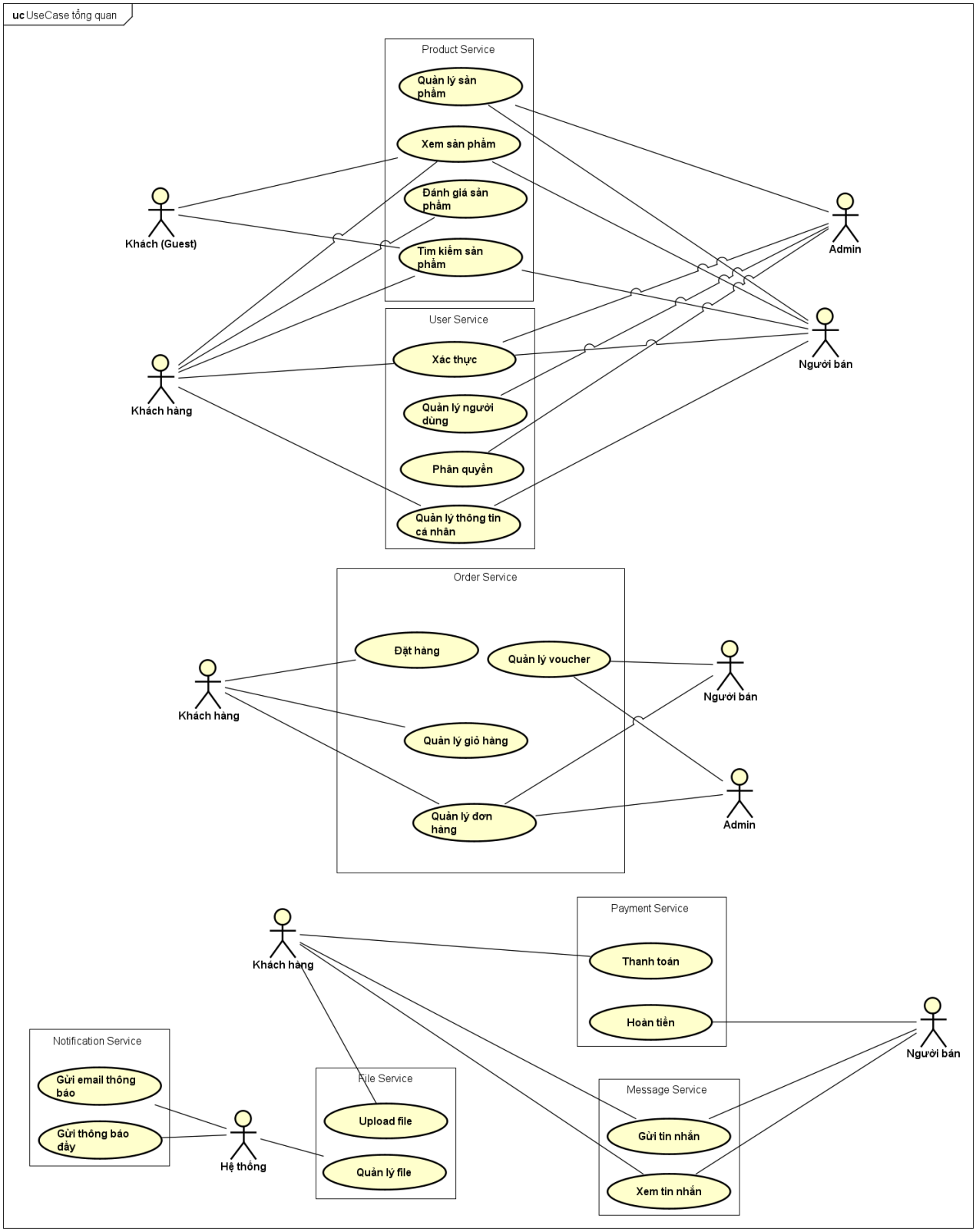
2 Phân tích và thiết kế hệ thống

2.1 Phân tích yêu cầu

2.1.1 Biểu đồ Use Case tổng quan

Qua phân tích các yêu cầu nghiệp vụ của hệ thống thương mại điện tử đa dịch vụ dựa trên kiến trúc Microservices, nhóm đã xây dựng biểu đồ Use Case tổng quan như sau:

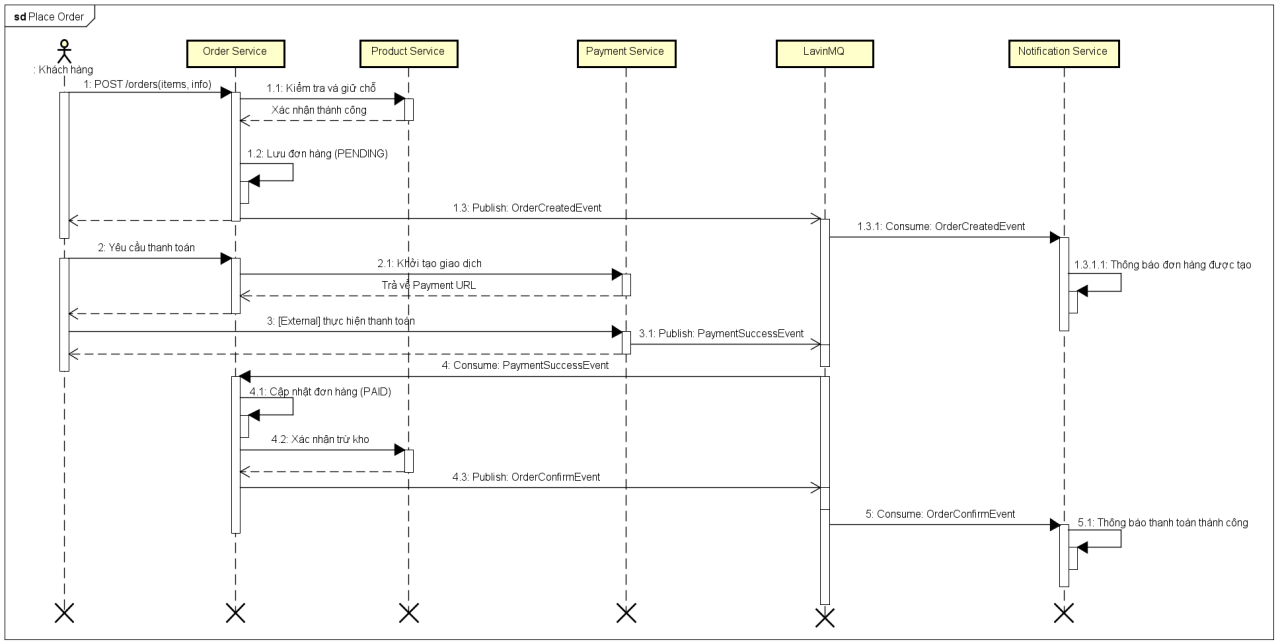
- **Khách vãng lai (Guest):** Có thể tìm kiếm và xem sản phẩm trước khi quyết định sử dụng hệ thống.
- **Khách hàng/Người mua (Buyer):** Thực hiện các chức năng cốt lõi như tìm kiếm sản phẩm với hỗ trợ gợi ý, trao đổi trực tiếp với người bán qua hệ thống chat real-time, quản lý giỏ hàng, đặt hàng và nhận thông báo tức thời về trạng thái đơn hàng.
- **Người bán (Seller):** Quản lý thông tin cửa hàng, danh mục sản phẩm, theo dõi và xử lý các yêu cầu từ phía khách hàng.
- **Quản trị viên (Admin):** Thực hiện quản trị toàn bộ hệ thống, bao gồm quản lý tài khoản và thực thi chính sách phân quyền nghiêm ngặt dựa trên vai trò (RBAC) để đảm bảo an toàn tài nguyên.



Hình 1: Biểu đồ Use Case tổng quan

2.1.2 Biểu đồ tuần tự

Để minh họa luồng tương tác phức tạp giữa các Microservices, nhóm lựa chọn biểu đồ tuần tự cho chức năng **Đặt hàng và thanh toán**. Quy trình này được chia thành nhiều giai đoạn nhằm đảm bảo tính toàn vẹn dữ liệu và trải nghiệm người dùng:



Hình 2: Biểu đồ tuần tự chức năng Đặt hàng

Việc tách biệt quy trình thành các bước bất đồng bộ giúp hệ thống không bị treo khi chờ đợi thanh toán hoặc gửi thông báo.

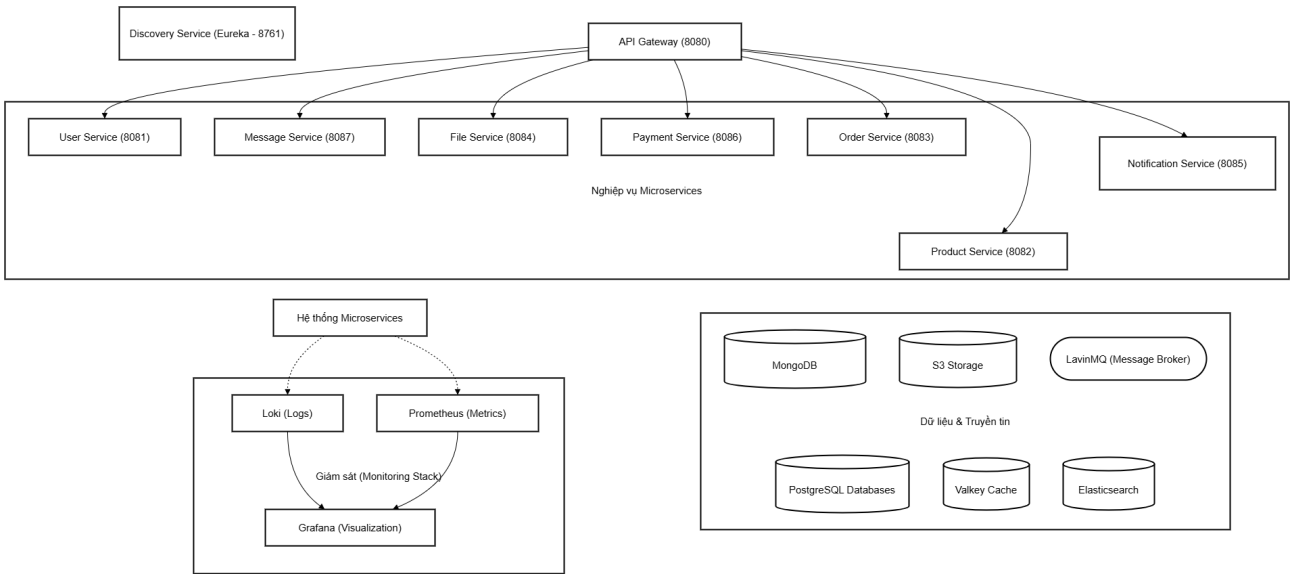
2.2 Thiết kế kiến trúc hệ thống

2.2.1 Mô hình tổng quan các Microservices

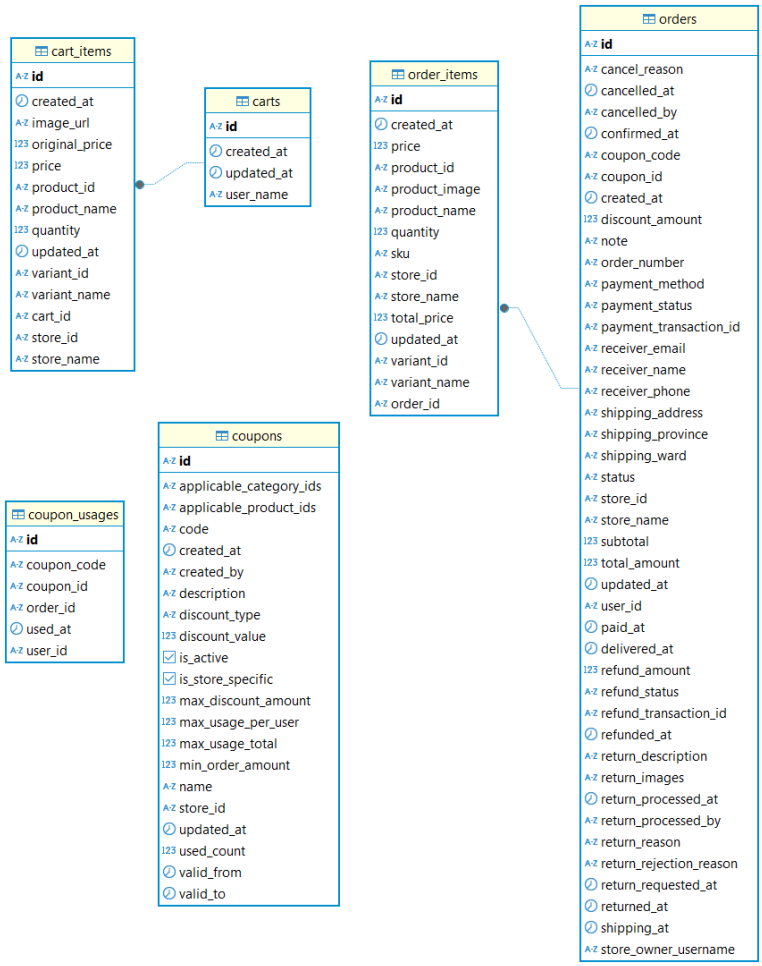
Hệ thống được xây dựng trên nền tảng ngôn ngữ lập trình **Java** và framework **Spring Boot**.

Nhóm dịch vụ	Tên Service	Database	Chức năng chính
Hạ tầng	API Gateway	-	Cổng vào (Port 8080), định tuyến, bảo mật.
	Discovery	-	Đăng ký dịch vụ qua Eureka.
	Monitoring	Prometheus	Giám sát hệ thống.
Nghiệp vụ	User	PostgreSQL	Quản lý người dùng, RBAC và Store.
	Product	ElasticSearch, PostgreSQL	Quản lý sản phẩm, tồn kho và tìm kiếm.
	Order	PostgreSQL	Xử lý đơn hàng, giỏ hàng và Coupon.
	Payment	PostgreSQL	Thanh toán ZaloPay và hoàn tiền.
Tiện ích	Message	MongoDB	Chat real-time qua WebSocket.
	Notification	PostgreSQL	Gửi Email (SendGrid) và Push notification.
	File	Space Object Storage	Lưu trữ file trên S3.
	Common DTO	-	Thư viện dùng chung.

Bảng 1: Tổng hợp các services trong kiến trúc hệ thống



Hình 3: Sơ đồ hệ thống



Hình 4: Database Diagram cho Order-service

2.2.2 Cơ chế giao tiếp và Lưu trữ

Để đảm bảo hiệu suất và tính nhất quán, hệ thống triển khai đa dạng các phương thức giao tiếp và caching:

- **Giao tiếp nội bộ:** Kết hợp giữa **Feign Client** cho các yêu cầu đồng bộ (như kiểm tra tồn kho) và **LavinMQ** cho các luồng bất đồng bộ (như xử lý sau thanh toán).
- **Chiến lược Caching:** Sử dụng **Valkey** (bản fork của Redis) để lưu trữ thông tin phiên làm việc, giỏ hàng và kết quả tìm kiếm nhằm giảm tải cho cơ sở dữ liệu chính.
- **Bảo mật Gateway:** Mọi yêu cầu đi qua Port 8080 đều được kiểm tra **JWT Token** và phân quyền trước khi chuyển tiếp tới các service bên trong.

2.3 Thiết kế giao diện hệ thống

Giao diện người dùng của hệ thống được thiết kế tập trung vào trải nghiệm người dùng (UX) và khả năng phản hồi linh hoạt (Responsive Design). Hệ thống đảm bảo tính nhất quán về màu sắc, phông chữ và các thành phần điều hướng trên toàn bộ các trang chức năng.

Để xây dựng một giao diện hiệu năng cao và dễ bảo trì, nhóm áp dụng các công nghệ và nguyên tắc sau:

- **Công nghệ:** Sử dụng **ReactJS** kết hợp với **JavaScript** và **CSS Modules** để quản lý các thành phần giao diện một cách độc lập. Sử dụng thư việc UI **Ant Design** để tiết kiệm thời gian thiết kế. **Websocket** và **SockJS** cũng được sử dụng để phục vụ tính năng chat real-time và gửi thông báo.
- **Bảo mật phía Client:** Tích hợp thư viện **DOMPurify** để làm sạch dữ liệu HTML (Rich Text) từ mô tả sản phẩm, ngăn chặn triệt để các lỗ hổng XSS trước khi hiển thị lên trình duyệt. Ngoài ra, sử dụng **Protected Routes** để ngắn người dùng truy cập vào các trang không được phép.
- **Nguyên tắc Responsive:** Giao diện được tối ưu hóa để hiển thị tốt trên nhiều loại thiết bị từ máy tính để bàn đến các thiết bị di động, đảm bảo luồng đặt hàng không bị gián đoạn.

3 Các tính năng nổi bật

3.1 Bảo mật

3.1.1 Chống tấn công SQL Injection

SQL Injection là một kỹ thuật tấn công mạng phổ biến, trong đó kẻ tấn công chèn các đoạn mã SQL độc hại vào các trường nhập liệu của ứng dụng. Mục đích của cuộc tấn công là làm thay đổi cấu trúc câu lệnh truy vấn ban đầu, từ đó có thể truy cập trái phép, chỉnh sửa hoặc xóa dữ liệu nhạy cảm trong cơ sở dữ liệu.

Để phòng chống loại hình tấn công này, nhóm đã triển khai các biện pháp kỹ thuật sau:

- **Sử dụng thư viện ORM (Object-Relational Mapping):** Nhóm sử dụng **Spring Data JPA** làm tầng tương tác dữ liệu.
- **Sử dụng Named Parameters (Tham số có tên):** Thay vì sử dụng cơ chế nối chuỗi, nhóm sử dụng các trình giữ chỗ được định danh bằng dấu hai chấm.

Cụ thể, mã nguồn tại tầng Repository được triển khai như sau:

```
/**
 * Tim don hang theo Id và fetch cac item lien quan
 */
@Query("SELECT o FROM Order o LEFT JOIN FETCH o.items WHERE o.id = :
    orderId")
Optional<Order> findByIdWithItems(@Param("orderId") String orderId);

/**
 * Tim don hang theo ma giao dich thanh toan
 */
List<Order> findByPaymentTransactionId(String paymentTransactionId);
```

Mã nguồn 1: Sử dụng Named Parameters trong Repository

Khi sử dụng kỹ thuật này, hệ thống sẽ thực hiện **Parameter Binding**. Các tham số truyền vào như `orderId` sẽ được trình điều khiển cơ sở dữ liệu xử lý như một giá trị thuần túy.

3.1.2 Chống tấn công XSS

Cross-Site Scripting (XSS) là lỗ hổng cho phép kẻ tấn công chèn các đoạn mã script độc hại vào trang web.

Trong dự án này, do yêu cầu hiển thị các mô tả sản phẩm có định dạng (Rich Text), nhóm đã triển khai giải pháp sau:

- **Sử dụng thư viện DOMPurify:** Quét và loại bỏ các thẻ nguy hiểm hoặc các thuộc tính thực thi sự kiện.
- **Cơ chế Whitelisting (Danh sách trắng):** Cấu hình cụ thể các thẻ HTML an toàn được phép hiển thị.

Cụ thể, mã nguồn xử lý tại Frontend (ReactJS) được triển khai như sau:

```
// Su dung DOMPurify de loai bo cac script tags va event handlers doc
    hai
{product.description ? (
```

```
  <div
    className={styles.richDescription}
    dangerouslySetInnerHTML={{
      __html: DOMPurify.sanitize(product.description, {
        // Chi cho phep cac the dinh dang van ban an toan
        ALLOWED_TAGS: ["p", "br", "strong", "em", "u", "s"]
      })
    }}
  />
) : null}
```

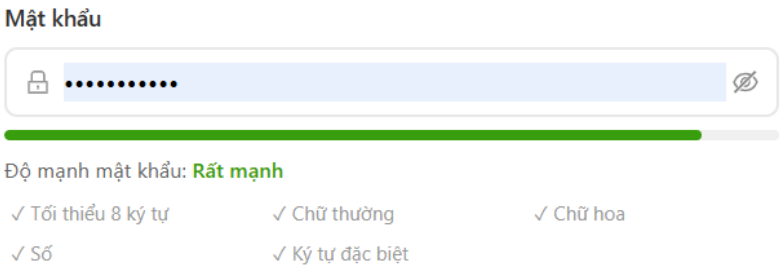
Mã nguồn 2: Sử dụng DOMPurify để làm sạch dữ liệu HTML tại Frontend

Việc sử dụng thuộc tính `dangerouslySetInnerHTML` kết hợp với `DOMPurify.sanitize` giúp hệ thống chỉ chấp nhận các thẻ định dạng văn bản cơ bản.

3.1.3 Chống tấn công Brute Force mật khẩu

Tấn công Brute Force là hình thức kẻ tấn công thử tất cả các tổ hợp mật khẩu có thể cho đến khi tìm đúng. Để bảo vệ thông tin tài khoản của người dùng, nhóm đã triển khai các cơ chế sau:

- **Sử dụng thư viện BCrypt để mã hóa mật khẩu:**
  - **Cơ chế Salt:** BCrypt tự động tạo ra một chuỗi ngẫu nhiên (salt) và kết hợp với mật khẩu trước khi băm. Điều này vô hiệu hóa hoàn toàn kiểu tấn công bằng bảng tra cứu sẵn (*Rainbow Tables*).
  - **Cost Factor:** Làm chậm quá trình tính toán của kẻ tấn công mà không ảnh hưởng đến trải nghiệm người dùng.
- **Yêu cầu độ phức tạp của mật khẩu:** Hệ thống tích hợp bộ đo độ mạnh mật khẩu ngay tại giao diện đăng ký.



Hình 5: Giao diện kiểm tra độ mạnh mật khẩu và các tiêu chí bảo mật

Cụ thể, việc cấu hình bộ mã hóa mật khẩu trong Spring Security được triển khai như sau:

```
@Bean
public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}
```

Mã nguồn 3: Cấu hình Bean PasswordEncoder sử dụng BCrypt

Nhờ sự kết hợp giữa mã hóa mạnh mẽ và ràng buộc chính sách mật khẩu, hệ thống giảm thiểu tối đa rủi ro bị chiếm quyền điều khiển tài khoản.

3.1.4 Chống tấn công Authorization Bypass

Authorization Bypass xảy ra khi kẻ tấn công có thể truy cập vào các tài nguyên hoặc thực hiện các hành động mà họ không có quyền. Để ngăn chặn rủi ro này, nhóm đã triển khai hệ thống phân quyền dựa trên vai trò (*Role-Based Access Control - RBAC*) một cách chặt chẽ ở cả Backend và Frontend:

- **Kiểm soát tại Backend:** Sử dụng các Annotation bảo mật để bảo vệ các API. Chỉ những người dùng có vai trò phù hợp mới có thể kích hoạt các hàm xử lý dữ liệu.
- **Kiểm soát tại Frontend:** Sử dụng các kỹ thuật định tuyến bảo vệ (*Protected Routes*) để ngăn chặn người dùng truy cập vào các trang giao diện không thuộc thẩm quyền của họ.

Cụ thể, mã nguồn bảo vệ API tại tầng Controller (Backend) được triển khai như sau:

```
@Override
@PreAuthorize("hasRole('ADMIN')")
public RoleResponse createRole(RoleRequest request) {
    // Logic tao role chi danh cho Admin
}
```

Mã nguồn 4: Sử dụng @PreAuthorize để phân quyền truy cập API

Đồng thời, tại phía Frontend (ReactJS), hệ thống sử dụng các Component bọc ngoài để bảo vệ các tuyến đường (routes):

```
{
  path: PROTECTED_ROUTES.USER_CHECKOUT,
  element: (
    <SuspenseWrapper>
      <ProtectedRoute allowedRoles={[ROLES.USER, ROLES.ADMIN]}>
        <CheckoutPage />
      </ProtectedRoute>
    </SuspenseWrapper>
  ),
},
```

Mã nguồn 5: Bảo vệ tuyến đường truy cập tại Frontend

Sự phối hợp kiểm tra quyền ở cả hai phía giúp đảm bảo rằng ngay cả khi kẻ tấn công tìm cách vượt qua lớp bảo vệ giao diện, họ vẫn sẽ bị chặn lại bởi tầng nghiệp vụ tại Server.

3.1.5 Chống tấn công CSRF

Cross-Site Request Forgery (CSRF) là kỹ thuật tấn công buộc người dùng thực thi các hành động không mong muốn trên một ứng dụng web mà họ đã xác thực. Nhóm triển khai các biện pháp sau:

- **Sử dụng JWT Token Authentication:** Token được gửi qua header `Authorization`. Trình duyệt không tự động đính kèm header này giống như cookie, ngăn chặn kẻ tấn công từ website khác có thể giả mạo yêu cầu.
- **Không sử dụng Session-based Authentication:** Hệ thống không dùng session cookie, loại bỏ hoàn toàn bề mặt tấn công dựa trên cookie truyền thống.

- **Cấu hình CORS (Cross-Origin Resource Sharing) chặt chẽ:** Chỉ cho phép các yêu cầu từ danh sách các nguồn (origin) được tin tưởng (whitelist).

Cụ thể, cấu hình CORS tại Backend được triển khai như sau:

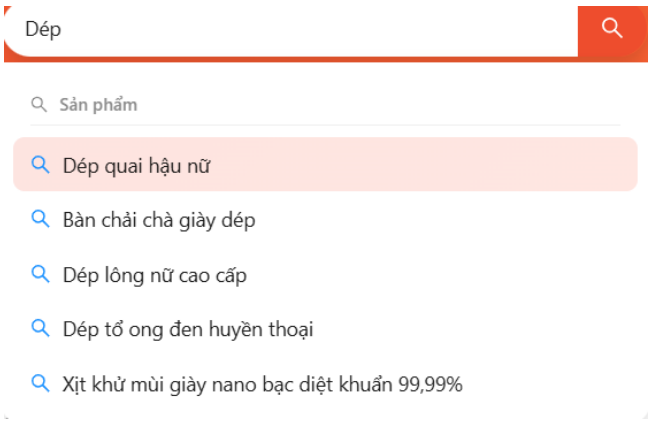
```
corsConfig.setAllowedOrigins(Arrays.asList(
    "http://localhost:5173",
    "http://localhost:3000",
    "https://nguyentheanh-nta.id.vn",
    "https://www.nguyentheanh-nta.id.vn",
    "http://nguyentheanh-nta.id.vn"
));

corsConfig.setAllowedMethods(Arrays.asList(
    "GET", "POST", "PUT", "DELETE", "PATCH", "OPTIONS"
));
```

Mã nguồn 6: Cấu hình CORS Whitelist và HTTP Methods

3.2 Hệ thống tìm kiếm và gợi ý

Để tối ưu hóa trải nghiệm người dùng trong việc tìm chọn sản phẩm, nhóm đã triển khai hệ thống tìm kiếm thông minh:

- **Công nghệ lưu trữ:** Nhóm sử dụng **ElasticSearch** để lưu trữ và lập chỉ mục (indexing) dữ liệu sản phẩm và cửa hàng.
- **Ưu điểm:** Việc sử dụng ElasticSearch cho phép hệ thống thực hiện các truy vấn tìm kiếm thông minh, hỗ trợ gợi ý từ khóa ngay khi người dùng đang nhập, mang lại tốc độ và độ chính xác vượt trội so với các phương pháp truy vấn SQL truyền thống.



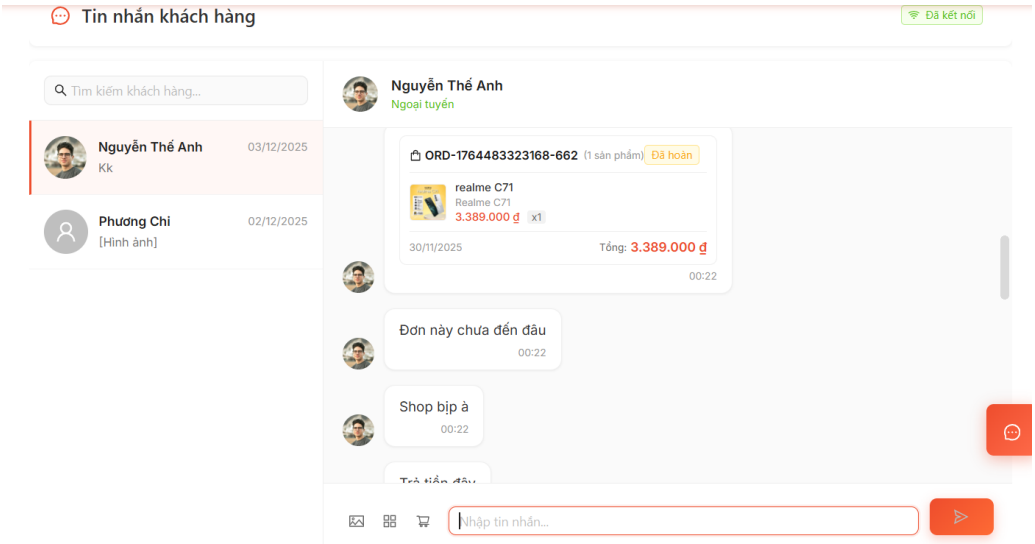
Hình 6: Minh họa tính năng gợi ý thông minh khi tìm kiếm sản phẩm

3.3 Hệ thống nhắn tin giữa người mua và người bán

Nhằm tăng cường khả năng tương tác và hỗ trợ khách hàng, hệ thống tích hợp tính năng nhắn tin trực tuyến:

- **Kết nối Real-time:** Nhóm sử dụng công nghệ **WebSocket** để duy trì kết nối hai chiều liên tục, cho phép gửi và nhận tin nhắn tức thời giữa người mua và người bán mà không cần tải lại trang.

- **Hỗ trợ đa phương tiện:** Ngoài văn bản thuần túy, người dùng có thể đính kèm thông tin đơn hàng cụ thể, chi tiết sản phẩm hoặc hình ảnh để nhận được sự hỗ trợ chính xác nhất từ phía người bán.

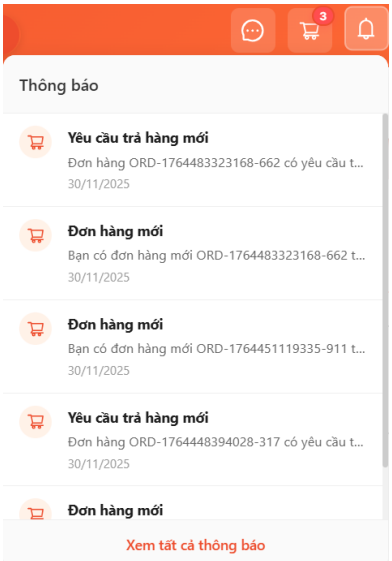


Hình 7: Giao diện nhắn tin hỗ trợ giữa người dùng và cửa hàng

3.4 Hệ thống thông báo đẩy và thông báo email

Hệ thống đảm bảo người dùng luôn nhận được các cập nhật quan trọng thông qua cơ chế thông báo đa kênh:

- **Thông báo đẩy (Push Notification):** Sử dụng **WebSocket** để gửi các thông báo trực tiếp lên giao diện người dùng ngay khi có sự kiện mới như đơn hàng mới hoặc thay đổi trạng thái yêu cầu.
- **Thông báo Email:** Nhóm sử dụng **RabbitMQ** làm hàng đợi thông điệp để xử lý các tác vụ gửi mail một cách không đồng bộ. Kết hợp với dịch vụ **SendGrid**, hệ thống đảm bảo việc gửi email thông báo (như xác nhận đơn hàng, thông tin tài khoản) luôn ổn định và tin cậy.



Hình 8: Giao diện danh sách thông báo và nhật ký gửi email từ hệ thống

4 Triển khai

4.1 Nhà cung cấp dịch vụ Cloud

Dịch vụ Cloud (Điện toán đám mây) là mô hình cung cấp các tài nguyên máy tính bao gồm máy chủ, lưu trữ, cơ sở dữ liệu và mạng lưới thông qua Internet. Thay vì phải tự vận hành hạ tầng vật lý tốn kém, nhóm có thể thuê và quản lý tài nguyên một cách linh hoạt theo nhu cầu thực tế.

Nhóm quyết định lựa chọn nhà cung cấp dịch vụ cloud **Digital Ocean** vì những lý do sau:

- **200$ miễn phí:** Digital Ocean cung cấp gói 200$ cho sinh viên đã xác thực tài khoản Github Education.
- **Tính đơn giản:** Giao diện quản lý trực quan, dễ dùng.
- **Hiệu năng ổn định:** Các Droplets và Managed Databases cung cấp hiệu suất cao.

Nhóm đã triển khai và sử dụng các dịch vụ Cloud cụ thể sau đây:

4.1.1 Cơ sở dữ liệu

Digital Ocean cung cấp các dịch vụ **Managed Databases**, giúp hệ thống tự động hóa các tác vụ quản trị phức tạp như sao lưu dữ liệu hàng ngày, cập nhật các bản vá lỗi bảo mật và tối ưu hóa cấu hình phần cứng. Điều này giúp nhóm tập trung hoàn toàn vào việc phát triển nghiệp vụ thay vì quản lý hạ tầng.

Nhóm sử dụng ba loại cơ sở dữ liệu chính để đáp ứng các nhu cầu lưu trữ khác nhau:

- **PostgreSQL:** Được sử dụng để lưu trữ các dữ liệu có cấu trúc, quan hệ chặt chẽ như thông tin người dùng, danh mục sản phẩm và quản lý đơn hàng.
- **MongoDB:** Được sử dụng để lưu trữ lịch sử tin nhắn trong hệ thống Chat.
- **Valkey:** Được sử dụng để lưu cache tập trung

Hình 9: Cơ sở dữ liệu PostgreSQL

Hình 10: Cơ sở dữ liệu MongoDB

4.1.2 Máy chủ

Nhóm sử dụng máy chủ ảo (Droplet) chạy hệ điều hành **Ubuntu 22.04 LTS**. Để đảm bảo tính nhất quán giữa môi trường phát triển (Local) và môi trường thực tế (Production), nhóm triển khai công nghệ **Docker**.

Việc cài đặt Docker trên Ubuntu cho phép nhóm container hóa các microservices như *product-service*, *order-service*, và *api-gateway*. Mọi dịch vụ đều được quản lý thông qua Docker Compose, giúp quá trình triển khai (Deployment) diễn ra nhanh chóng, dễ dàng mở rộng và giảm thiểu sai sót do khác biệt về môi trường hệ thống.

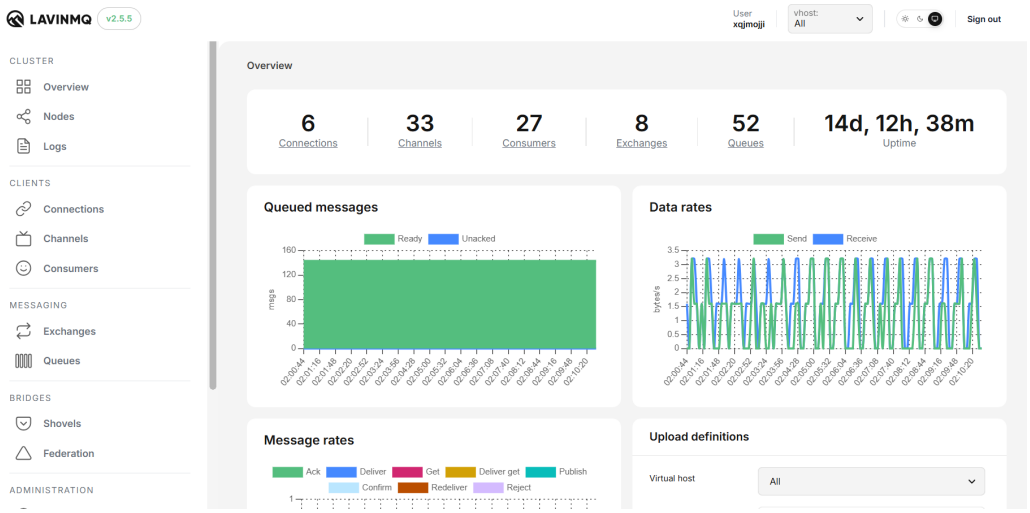
Hình 11: Droplet chạy Docker trên nền tảng Ubuntu 22.04

4.1.3 Message Broker

Để giải quyết các tác vụ chạy ngầm và đảm bảo tính bất đồng bộ (*asynchronous*) cho hệ thống, nhóm sử dụng **LavinMQ** (phiên bản nhẹ hơn của RabbitMQ) thông qua dịch vụ lưu trữ đám mây của **CloudAMQP**.

Việc sử dụng Message Broker mang lại các lợi ích sau:

- **Giảm độ trễ cho người dùng:** Các tác vụ tiêu tốn thời gian như gửi email xác nhận sẽ được đẩy vào hàng đợi (*queue*) để xử lý sau, giúp phản hồi phía Client ngay lập tức.
- **Tăng tính ổn định:** LavinMQ kết hợp với dịch vụ **SendGrid** giúp đảm bảo các thông báo email không bị mất mát ngay cả khi có lượng lớn yêu cầu cùng lúc.



Hình 12: Giao diện quản lý hàng đợi LavinMQ trên CloudAMQP

4.1.4 Spaces Object Storage

Để quản lý các tệp tin tĩnh và dữ liệu phi cấu trúc với dung lượng lớn, nhóm sử dụng **DigitalOcean Spaces**. Đây là một dịch vụ lưu trữ đối tượng (Object Storage) tương thích với giao thức S3, giúp hệ thống tách biệt hoàn toàn phần lưu trữ mã nguồn và lưu trữ dữ liệu người dùng.

Việc tích hợp Spaces mang lại những lợi thế quan trọng cho nền tảng:

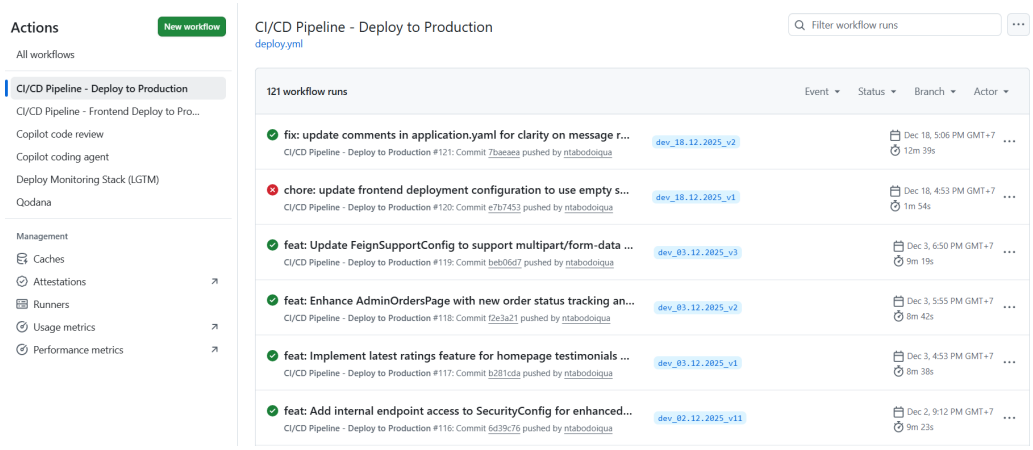
- **Lưu trữ tài nguyên tĩnh:** Toàn bộ hình ảnh sản phẩm, ảnh đại diện người dùng, các biểu ngữ quảng cáo và hồ sơ được lưu trữ tại đây thay vì lưu trực tiếp trên máy chủ Droplet. Điều này giúp giữ cho máy chủ luôn "stateless" và dễ dàng mở rộng.
- **Tích hợp CDN (Content Delivery Network):** Spaces đi kèm với mạng lưới phân phối nội dung giúp giảm độ trễ khi tải hình ảnh cho người dùng ở các vị trí địa lý khác nhau, đồng thời tiết kiệm băng thông cho máy chủ chính.
- **Khả năng mở rộng vô hạn:** Hệ thống có thể lưu trữ hàng triệu tệp tin mà không cần lo lắng về giới hạn dung lượng ổ cứng của máy chủ ảo.
- **Bảo mật dữ liệu:** Hỗ trợ các quyền truy cập riêng tư (Private) và tạo đường dẫn truy cập có thời hạn (Presigned URLs) để đảm bảo an toàn cho các tệp tin nhạy cảm.

4.2 CI/CD với Github Actions

Github Actions là một nền tảng tự động hóa quy trình (*Workflow*) giúp tích hợp và triển khai mã nguồn liên tục (CI/CD). Thay vì phải thực hiện các bước đóng gói và đẩy mã lên máy chủ một cách thủ công, nhóm đã thiết lập các luồng công việc tự động để tối ưu hóa quy trình phát triển.

Hệ thống triển khai tự động lên server được kích hoạt mỗi khi thành viên trong nhóm thực hiện thao tác **push tags** theo định dạng quy định:

- **Đối với Backend:** Kích hoạt khi đẩy tag có định dạng: `dev_dd.mm.yy_v*` (ví dụ: `dev_18.12.2025_v2`).
- **Đối với Frontend và NGINX:** Kích hoạt khi đẩy tag có định dạng: `fe_dev_dd.mm.yy_v*`.



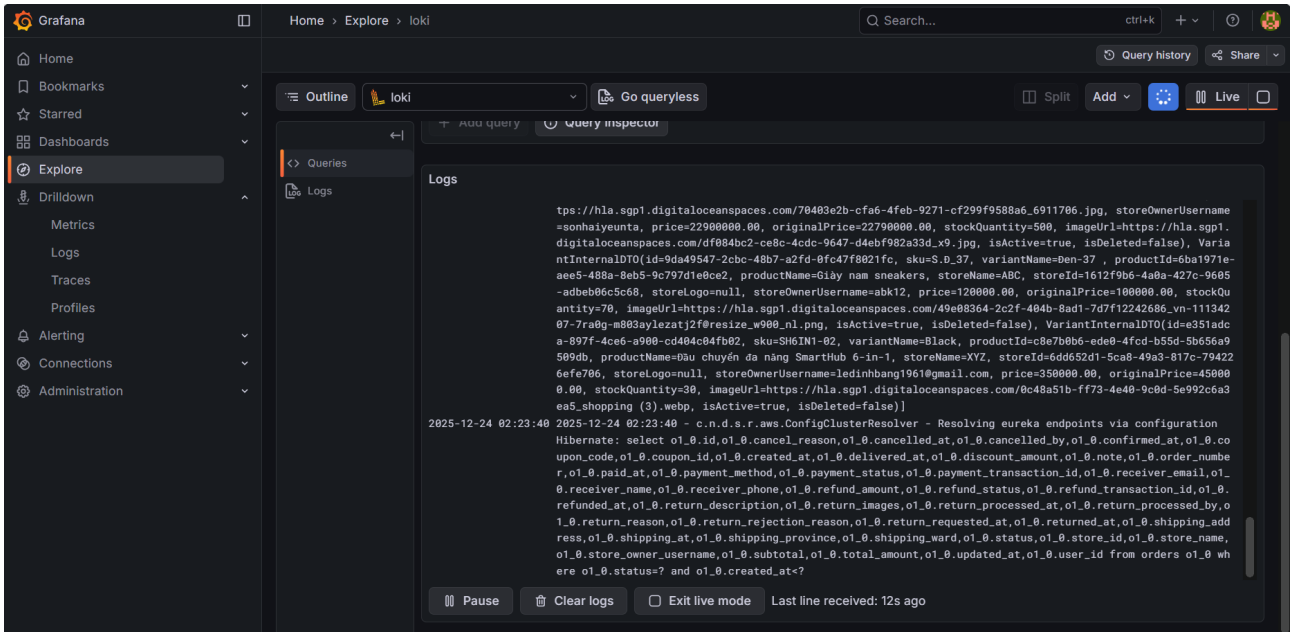
Hình 13: Lịch sử các phiên bản triển khai tự động thông qua Github Actions

Việc áp dụng CI/CD giúp nhóm rút ngắn thời gian phát hành tính năng mới, đồng thời đảm bảo mã nguồn luôn được kiểm tra và triển khai đồng nhất trên môi trường thực tế.

4.3 Hệ thống Monitoring với PLG Stack

Để đảm bảo hệ thống luôn hoạt động ổn định và có thể phát hiện sớm các sự cố, nhóm đã triển khai giải pháp giám sát tập trung sử dụng bộ công cụ **PLG Stack** (Prometheus, Loki, Grafana).

- **Prometheus (Metrics):** Đóng vai trò là trung tâm thu thập các dữ liệu chuỗi thời gian (*Time-series data*) như CPU, RAM, số lượng Request, thời gian phản hồi API.
- **Loki (Logs):** Là hệ thống tập trung hóa nhật ký (*Log aggregation*). Nhóm sử dụng Loki kết hợp với các Agent để thu thập toàn bộ log từ các Docker Container đang chạy trên Server.
- **Grafana (Visualization):** Là tầng hiển thị dữ liệu cuối cùng.



Hình 14: Hệ thống Logs tập trung từ Loki trên giao diện Grafana

5 Kết quả thực hiện

5.1 Các yêu cầu chức năng đã hoàn thiện

Dựa trên kết quả triển khai, hệ thống đã hoàn thiện các quy trình nghiệp vụ cốt lõi cho ba nhóm đối tượng chính: Khách hàng, Người bán và Quản trị viên.

Về phía khách hàng:

Chức năng	Trạng thái	Chi tiết triển khai
Đăng ký / Đăng nhập	Hoàn thành	Hỗ trợ đăng ký, đăng nhập, xác thực email, quên mật khẩu sử dụng JWT.
Quản lý hồ sơ	Hoàn thành	Cập nhật thông tin cá nhân, địa chỉ (Tỉnh/Xã), đổi mật khẩu, nộp hồ sơ thành người bán.
Tìm kiếm & Duyệt	Hoàn thành	Tìm kiếm sản phẩm, lọc theo danh mục và thương hiệu.
Giỏ hàng & Đặt hàng	Hoàn thành	Quy trình Checkout, chọn địa chỉ và phương thức thanh toán.
Thanh toán	Hoàn thành	Tích hợp thành công cổng thanh toán ZaloPay và xem lịch sử giao dịch.
Tương tác	Hoàn thành	Chat trực tiếp (Real-time) và nhận thông báo về đơn hàng/hệ thống.

Bảng 2: Đánh giá chức năng phân hệ Khách hàng

Về phía quản trị viên và người bán:

- **Đối với Người bán:**
  - Hệ thống Dashboard thống kê doanh thu và đơn hàng theo thời gian thực.
  - Quản lý sản phẩm đa biến thể (variants) và các thuộc tính động (attributes).
  - Công cụ cấu hình cửa hàng, banner và giao diện chăm sóc khách hàng.
- **Đối với Quản trị viên:**
  - Quản lý tập trung người dùng, phân quyền Role-Based Access Control (RBAC).
  - Kiểm duyệt sản phẩm, cửa hàng và quản lý danh mục/thương hiệu toàn hệ thống.
  - Theo dõi toàn bộ luồng đơn hàng và giao dịch thanh toán trên sàn.

5.2 Các yêu cầu phi chức năng đã hoàn thiện

Hệ thống được thiết kế để đảm bảo tính ổn định, bảo mật và khả năng mở rộng thông qua các tiêu chuẩn kỹ thuật hiện đại.

- **Kiến trúc và Hiệu năng:**
  - Triển khai kiến trúc **Microservices** với Java Spring Boot và Spring Cloud.
  - Áp dụng mô hình **Database per service** giúp tách biệt hoàn toàn dữ liệu giữa các dịch vụ.
  - Sử dụng **API Gateway** (Spring Cloud Gateway) để định tuyến và cân bằng tải (Load Balancing).

  - Tích hợp **Service Discovery** (Eureka) cho phép mở rộng (scaling) các dịch vụ linh hoạt.

- **Bảo mật và Giám sát:**
  - Hệ thống được thiết kế để chống nhiều hình thức tấn công Web phổ biến (SQLi, XSS, Authorization Bypass, ...).
  - Hệ thống giám sát chỉ số (Metrics) tập trung thông qua **Prometheus**.
  - Quản lý log tập trung bằng bộ công cụ **Loki** và **Promtail**.
  - Đóng gói và triển khai đồng nhất bằng công nghệ **Docker** và **Docker Compose**.
- **Giao diện:**
  - Giao diện đẹp, trực quan, dễ sử dụng, tuân thủ các nguyên tắc UX-UI.
  - Được thiết kế Responsive để có thể sử dụng tốt trên nhiều thiết bị

6 Tổng kết

6.1 Đánh giá đóng góp các thành viên

Tên thành viên	MSSV	Đóng góp	Đánh giá
Nguyễn Thế Anh	20224921	• Phân công, theo dõi tiến độ các thành viên. • Phát triển Backend: Message, Notification, File, Payment và Order services. • Xây dựng quy trình CI/CD và Monitoring hệ thống. • Tích hợp công nghệ: ElasticSearch và Valkey. • Tham gia tinh chỉnh báo cáo, slide.	Đầy đủ, đúng hạn
Hồ Lương An	20224821	• Viết báo cáo. • Phát triển Frontend cho các trang trang đăng ký đăng nhập, quản trị Admin, thêm các components footer, searchbar, navigation bar.	Đầy đủ, đúng hạn
Lê Đình Hùng Anh	20224917	• Làm slide. • Phát triển Frontend cho các trang người bán, người dùng, thêm các components footer.	Đầy đủ, đúng hạn
Bùi Khắc Anh	20225246	• Phát triển Backend cho các Service: User Service, Product Service. • Phát triển các tính năng thống kê (statistics). • Phân tích thiết kế hệ thống, vẽ biểu đồ.	Đầy đủ, đúng hạn

Bảng 3: Bảng phân công nhiệm vụ và đánh giá mức độ hoàn thành

6.2 Ưu và nhược điểm

6.2.1 Ưu điểm

Hệ thống được xây dựng với các ưu điểm nổi bật:

- **Kiến trúc Microservices:** Phân tách các services độc lập giúp tăng khả năng mở rộng, cô lập lỗi và phát triển song song.
- **Tech Stack hiện đại:** Sử dụng Spring Boot 3, Java 21, React 18 và hệ sinh thái Spring Cloud (Gateway, Eureka, OpenFeign) đảm bảo hiệu năng và tính ổn định.

- **Bảo mật đa lớp:** Triển khai JWT, OAuth2, RBAC và bảo mật tập trung tại API Gateway, chống nhiều hình thức tấn công.
- **Quản lý dữ liệu tối ưu:** Áp dụng mô hình Database-per-service với PostgreSQL, MongoDB, kết hợp Redis Caching và Elasticsearch cho tìm kiếm nâng cao.
- **Khả năng quan sát:** Tích hợp PLG để giám sát hệ thống và quản lý log tập trung.

6.2.2 Nhược điểm

Bên cạnh các ưu điểm, hệ thống vẫn tồn tại một số hạn chế cần khắc phục:

- **Độ bao phủ kiểm thử (Testing Coverage) thấp:** Thiếu hụt các bài kiểm thử Unit, Integration và E2E, dẫn đến rủi ro khi thay đổi mã nguồn.
- **Quản lý cơ sở dữ liệu:** Chưa sử dụng các công cụ migration như Flyway hoặc Liquibase để kiểm soát phiên bản schema.
- **Thiếu tài liệu API:** Thiếu Swagger gây khó khăn cho việc tích hợp và bàn giao.
- **Vận hành (Ops):** Thiếu cơ chế Distributed Tracing (Zipkin/Jaeger) để theo dõi luồng request, chưa có cơ chế roll-back khi CICD gặp lỗi.

6.3 Kiến thức đã học được

Thông qua dự án, nhóm đã tích lũy được các kiến thức chuyên môn quan trọng:

- **Kiến trúc phần mềm:** Hiểu và áp dụng thành công Microservices patterns, API Gateway, Service Discovery và Event-Driven Architecture với RabbitMQ.
- **Phát triển Backend:** Nắm vững hệ sinh thái Spring, tối ưu hóa truy vấn và triển khai bảo mật với Spring Security.
- **Phát triển Frontend:** Xây dựng ứng dụng SPA với React, thư viện UI Ant Design, quản lý State và xử lý luồng giao tiếp RESTful API qua Axios, Real-time qua WebSocket/STOMP.
- **Kỹ năng DevOps:** Sử dụng Github Actions, Docker/Docker Compose để container hóa và deploy ứng dụng. Cấu hình hệ thống giám sát với Prometheus, Grafana, Loki.

6.4 Hướng phát triển tương lai

Để hoàn thiện hệ thống trở thành một sản phẩm Enterprise-grade, định hướng phát triển bao gồm:

- **Nâng cao chất lượng mã nguồn:** Triển khai Unit/Integration Tests (target 70% coverage), tích hợp Flyway và SpringDoc OpenAPI.
- **Tối ưu hóa vận hành:** Triển khai Distributed Tracing và tập trung quản lý cấu hình qua Config Server.
- **Mở rộng nền tảng:** Phát triển ứng dụng Mobile (React Native) và di chuyển hệ thống lên điều phối bằng Kubernetes (K8s).
- **Tích hợp AI/ML:** Xây dựng công cụ gợi ý sản phẩm cá nhân hóa, tìm kiếm thông minh và hệ thống phân tích dữ liệu kinh doanh (BI).

Tài liệu tham khảo

[1] Đỗ Bá Lâm, *Bài giảng môn Công nghệ Web và dịch vụ trực tuyến*, Đại học Bách Khoa Hà Nội.

[2] Chris Richardson, *Microservices Patterns*, 2nd Edition, Manning Publications.

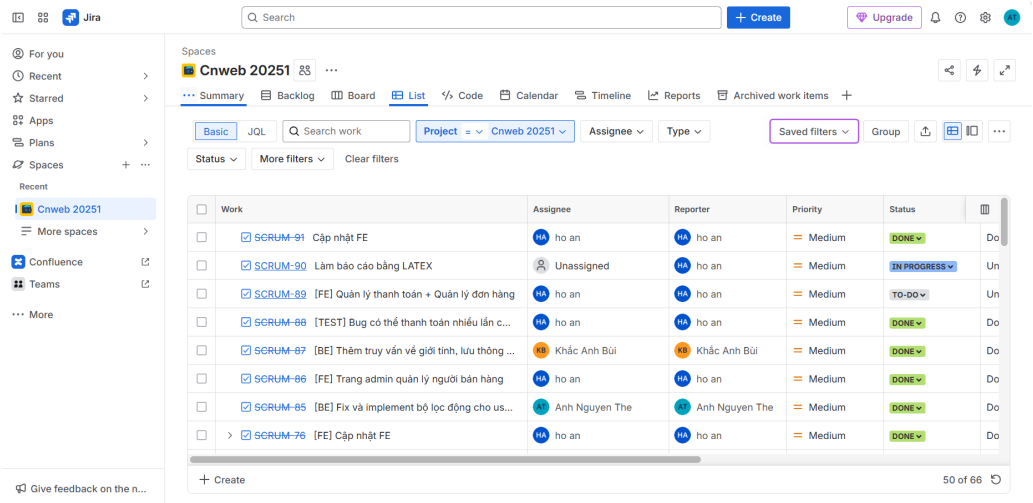
[3] Chris Richardson, "Microservices Architecture", *Microservices.io*.
`https://microservices.io/`.

[4] Spring Projects, "Spring Boot Documentation", *Spring.io*.
`https://docs.spring.io/spring-boot/index.html`.

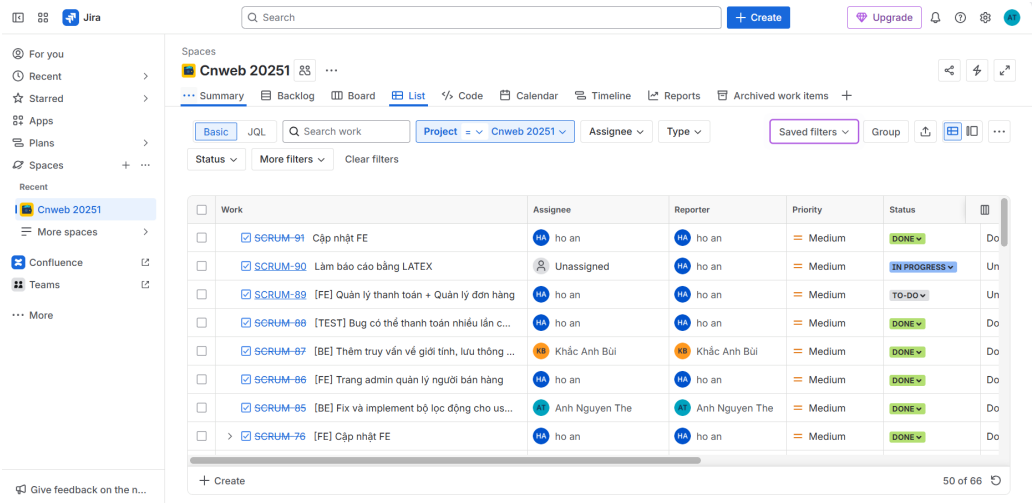
[5] Meta Platforms, Inc., "Quick Start - Learn React", *React.dev*.
`https://react.dev/learn`.

Phụ lục

1. **Mã nguồn dự án (Github Repository):**
   `https://github.com/ntabodoiqua/cnweb_20251`
2. **Tài liệu báo cáo, Slide và Demo:**
   `https://drive.google.com/drive/folders/1MyE1HaZQWU9Q3x9vgnKUuMEbEk0ISfUe?usp=sharing`
3. **Địa chỉ website dự án:**
   `https://nguyentheanh-nta.id.vn/`
   - *Tài khoản Admin:* `admin` / `admin`
4. **Hệ thống giám sát (Grafana):**
   `https://nguyentheanh-nta.id.vn/monitor/`
   - *Tài khoản monitor:* `admin` / `Theanh@3007`
5. **Jira và Github Contributions**



Hình 15: Phân công công việc trên Jira



Hình 16: Github Contributions