# 📑 Phân tích và Thiết kế Hệ thống ZylPelox (ZPX) - VERSION 1.3

**Mã dự án:** ZPX-ECOM **Ngày khởi tạo:** 10-02-2023 **Ngày cập nhật:** 14-04-2025 **Tác giả:** phamtiendungcw **Phiên bản:** 1.3

## 1. TỔNG QUAN DỰ ÁN

### 1.1. Mục tiêu

Xây dựng nền tảng cửa hàng bán lẻ trực tuyến ZylPelox (ZPX) với kiến trúc hiện đại, tích hợp đầy đủ tính năng quản lý từ đầu đến cuối, đáp ứng nhu cầu kinh doanh đa kênh với khả năng mở rộng cao và bảo trì dễ dàng.

***Chú thích:*** *Dự án nhằm tạo ra một nền tảng eCommerce toàn diện, không chỉ đáp ứng nhu cầu hiện tại mà còn có khả năng mở rộng trong tương lai.*

### 1.2. Phạm vi

* **Hệ thống quản lý bán hàng và tồn kho toàn diện**: Theo dõi sản phẩm, đơn hàng và tồn kho theo thời gian thực
* **Nền tảng eCommerce đầy đủ chức năng**: Trải nghiệm mua sắm trực tuyến liền mạch cho khách hàng
* **Tích hợp thanh toán và vận chuyển**: Hỗ trợ nhiều phương thức thanh toán và đối tác vận chuyển
* **Hệ thống CRM và quản lý khách hàng**: Quản lý thông tin và tương tác với khách hàng
* **Dashboard quản trị và báo cáo**: Công cụ phân tích dữ liệu và báo cáo toàn diện

***Chú thích:*** *Dự án sẽ bao gồm cả backend và frontend, API được thiết kế để dễ dàng mở rộng và tích hợp với các hệ thống khác.*

### 1.3. Giới hạn

* **Khối lượng dữ liệu và truy cập**: Hệ thống được thiết kế để xử lý khối lượng dữ liệu và người dùng từ trung bình đến lớn (lên đến 100,000 sản phẩm, 50,000 người dùng đồng thời)
* **Thời gian phát triển dự kiến**: 6-9 tháng cho phiên bản MVP, với lộ trình phát triển tiếp theo được chia thành các giai đoạn
* **Quy mô nhóm phát triển**: Yêu cầu đội ngũ 5-8 người bao gồm backend, frontend, QA và DevOps

***Chú thích:*** *Các giới hạn được xác định để đặt kỳ vọng rõ ràng và đảm bảo việc lập kế hoạch tài nguyên phù hợp.*

### 1.4. Yêu cầu phi chức năng

* **Hiệu năng**:
  + Response time < 200ms cho API calls
  + Page load time < 3s
  + Hỗ trợ 1000 concurrent users
* **Khả dụng**: 99.9% uptime
* **Bảo mật**: Tuân thủ OWASP Top 10, PCI DSS
* **Khả năng mở rộng**: Horizontal scaling ready

***Chú thích:*** *Yêu cầu phi chức năng được thiết kế để đảm bảo hệ thống hoạt động hiệu quả và an toàn trong môi trường sản xuất.*

## 2. YÊU CẦU CHỨC NĂNG (TÍNH NĂNG HỆ THỐNG)

### 2.1. Quản lý Catalog

* **Sản phẩm và biến thể**
  + Tạo/sửa/xóa/khôi phục sản phẩm với giao diện quản trị trực quan
  + Quản lý biến thể không giới hạn (màu sắc, kích thước, v.v.)
  + Thông số kỹ thuật và mô tả đa dạng với định dạng phong phú (rich text)
  + Giá cả và thuế có thể cấu hình theo khu vực

***Chú thích:*** *Hệ thống sẽ hỗ trợ sản phẩm đơn giản và sản phẩm có nhiều biến thể. Mỗi biến thể có thể có giá, mã SKU và tồn kho riêng.*

* **Danh mục**
  + Cấu trúc danh mục đa cấp không giới hạn độ sâu
  + Sản phẩm có thể thuộc nhiều danh mục khác nhau
  + SEO tối ưu cho từng danh mục (meta title, description, URL)

***Chú thích:*** *Danh mục sẽ được thiết kế với hiệu suất cao, đảm bảo thời gian phản hồi nhanh ngay cả khi có hàng nghìn sản phẩm.*

* **Quản lý phương tiện**
  + Thư viện hình ảnh/video tập trung với bộ lọc và tìm kiếm
  + Tối ưu hóa hình ảnh tự động (nén, thay đổi kích thước, chuyển đổi định dạng)
  + Hỗ trợ CDN để tăng tốc tải trang

***Chú thích:*** *Hệ thống sẽ sử dụng dịch vụ lưu trữ đám mây cho media để đảm bảo hiệu suất và khả năng mở rộng.*

### 2.2. Quản lý Khách hàng

* **Hồ sơ khách hàng chi tiết**: Thông tin cá nhân, lịch sử mua hàng, sở thích
* **Phân nhóm khách hàng và tags**: Phân loại khách hàng để tiếp thị và phân tích
* **Lịch sử mua hàng và tương tác**: Theo dõi toàn bộ hành trình khách hàng
* **Hệ thống đánh giá và điểm thưởng**: Khuyến khích khách hàng quay lại
* **Quản lý địa chỉ giao hàng/thanh toán**: Hỗ trợ nhiều địa chỉ cho mỗi khách hàng

***Chú thích:*** *Module này sẽ tích hợp với hệ thống CRM để cung cấp cái nhìn 360 độ về khách hàng.*

### 2.3. Quản lý Kho hàng

* **Theo dõi tồn kho theo thời gian thực**: Cập nhật số lượng tồn kho ngay khi có thay đổi
* **Nhập/xuất kho với lịch sử đầy đủ**: Theo dõi mọi thay đổi tồn kho
* **Quản lý nhiều kho và vị trí**: Hỗ trợ mô hình kinh doanh phân tán
* **Cảnh báo tồn kho thấp**: Thông báo khi sản phẩm gần hết hàng
* **Báo cáo tồn kho và giá trị hàng hóa**: Phân tích giá trị tồn kho theo thời gian

***Chú thích:*** *Hệ thống kho sẽ được thiết kế để xử lý cả quy trình thủ công và tự động, tích hợp với các thiết bị quét mã vạch nếu cần.*

### 2.4. Quản lý Vận chuyển

* **Tích hợp nhiều đối tác vận chuyển**: API-based integration với các đơn vị vận chuyển phổ biến
* **Ma trận phí vận chuyển tùy chỉnh**: Xác định phí vận chuyển dựa trên khu vực, trọng lượng, giá trị
* **Theo dõi trạng thái vận chuyển**: Cập nhật theo thời gian thực về tình trạng đơn hàng
* **In ấn vận đơn và đóng gói**: Tự động tạo và in vận đơn
* **Xử lý hoàn hàng**: Quy trình hoàn hàng và hoàn tiền đầy đủ

***Chú thích:*** *Module này sẽ cung cấp API mở để dễ dàng tích hợp với các nhà cung cấp dịch vụ vận chuyển mới.*

### 2.5. Quản lý Thanh toán

* **Đa dạng phương thức thanh toán**: Hỗ trợ thẻ tín dụng, chuyển khoản, ví điện tử, COD
* **Xử lý thanh toán an toàn**: Tuân thủ PCI DSS và tiêu chuẩn bảo mật
* **Lịch sử giao dịch và hoàn tiền**: Theo dõi đầy đủ các giao dịch tài chính
* **Hỗ trợ trả góp và thanh toán một phần**: Linh hoạt trong các phương thức thanh toán
* **Xác thực thanh toán**: Tích hợp 3D Secure và các phương thức xác thực hiện đại

***Chú thích:*** *Hệ thống thanh toán sẽ được thiết kế với tính mở rộng cao, cho phép thêm mới các cổng thanh toán dễ dàng.*

### 2.6. Quản lý Đơn hàng

* **Quy trình xử lý đơn hàng linh hoạt**: Tùy chỉnh quy trình theo nhu cầu kinh doanh
* **Trạng thái đơn hàng tùy chỉnh**: Tạo và quản lý các trạng thái đơn hàng theo yêu cầu
* **Lịch sử thay đổi đơn hàng**: Theo dõi mọi thay đổi trong vòng đời đơn hàng
* **Đơn hàng dự trữ và đặt trước**: Hỗ trợ đặt hàng trước khi sản phẩm có sẵn
* **Bán hàng chéo và bán thêm**: Đề xuất sản phẩm liên quan khi đặt hàng

***Chú thích:*** *Module đơn hàng là trung tâm của hệ thống, tích hợp với tất cả các module khác để cung cấp trải nghiệm xuyên suốt.*

### 2.7. Khuyến mãi & Giảm giá

* **Mã giảm giá và voucher**: Tạo và quản lý các mã giảm giá với nhiều điều kiện
* **Chương trình khuyến mãi theo thời gian**: Lên lịch khuyến mãi tự động
* **Giảm giá theo nhóm khách hàng**: Giá đặc biệt cho khách hàng VIP
* **Giảm giá theo số lượng**: Giảm giá khi mua số lượng lớn
* **Quà tặng và sản phẩm kèm theo**: Thêm sản phẩm quà tặng vào đơn hàng

***Chú thích:*** *Hệ thống khuyến mãi được thiết kế để hỗ trợ các chiến dịch marketing phức tạp với nhiều quy tắc kết hợp.*

### 2.8. Nội dung & Marketing

* **Blog và bài viết**: Hệ thống quản lý nội dung tích hợp
* **Landing page và banner quảng cáo**: Tạo và quản lý các trang đích cho chiến dịch
* **SEO tools và metadata**: Tối ưu hóa nội dung cho công cụ tìm kiếm
* **Email marketing templates**: Tạo và gửi email marketing từ hệ thống
* **Đánh giá và bình luận sản phẩm**: Thu thập phản hồi của khách hàng

***Chú thích:*** *Module này sẽ cung cấp các công cụ để quản lý nội dung và tối ưu hóa SEO một cách hiệu quả.*

### 2.9. Nhân viên & Phân quyền

* **Quản lý người dùng và vai trò**: Tạo và quản lý tài khoản nhân viên
* **Phân quyền chi tiết theo chức năng**: Kiểm soát quyền truy cập đến cấp độ hành động
* **Theo dõi hoạt động người dùng**: Ghi lại mọi thay đổi và hành động trong hệ thống
* **Xác thực hai lớp**: Tăng cường bảo mật cho tài khoản quan trọng
* **Giới hạn theo IP và thời gian**: Kiểm soát thời gian và địa điểm truy cập

***Chú thích:*** *Hệ thống phân quyền sẽ được thiết kế để linh hoạt và chi tiết, cho phép tạo các vai trò tùy chỉnh theo nhu cầu cụ thể của doanh nghiệp.*

## 2.10. Wishlist & So sánh sản phẩm

* **Danh sách yêu thích**: Lưu sản phẩm để mua sau
* **So sánh sản phẩm**: So sánh tối đa 4 sản phẩm cùng lúc
* **Thông báo giảm giá**: Alert khi sản phẩm trong wishlist giảm giá
* **Chia sẻ wishlist**: Tạo link chia sẻ cho người khác

***Chú thích:*** *Các tính năng này giúp tăng cường trải nghiệm người dùng, khuyến khích họ quay lại mua hàng.*

## 2.11. Tìm kiếm & Lọc nâng cao

* **Elasticsearch integration**: Full-text search với typo tolerance
* **Faceted search**: Lọc đa tiêu chí động
* **Search suggestions**: Gợi ý realtime khi gõ
* **Visual search**: Tìm kiếm bằng hình ảnh (AI-powered)
* **Voice search**: Tìm kiếm bằng giọng nói

***Chú thích:*** *Tính năng tìm kiếm và lọc nâng cao sẽ giúp khách hàng tìm thấy sản phẩm nhanh chóng, cải thiện trải nghiệm mua sắm.*

## 2.12. Hệ thống Recommendation

* **Personalized recommendations**: Dựa trên lịch sử và hành vi
* **Collaborative filtering**: “Khách hàng khác cũng mua”
* **Content-based filtering**: Sản phẩm tương tự
* **Trending products**: Xu hướng theo thời gian thực

***Chú thích:*** *Hệ thống gợi ý sẽ sử dụng machine learning để cải thiện độ chính xác theo thời gian, giúp tăng doanh thu và giữ chân khách hàng.*

## 2.13. Multi-channel Support

* **POS Integration**: Đồng bộ với cửa hàng offline
* **Social commerce**: Bán hàng qua Facebook, Instagram
* **Marketplace integration**: Kết nối Shopee, Lazada, Tiki
* **B2B Portal**: Cổng riêng cho khách hàng doanh nghiệp

***Chú thích:*** *Hỗ trợ đa kênh giúp mở rộng thị trường và tăng cường khả năng tiếp cận khách hàng.*

## 2.14. Analytics & Business Intelligence

* **Real-time dashboard**: Theo dõi KPIs quan trọng
* **Cohort analysis**: Phân tích nhóm khách hàng
* **Funnel analysis**: Phân tích chuyển đổi
* **A/B Testing framework**: Tối ưu hóa liên tục
* **Predictive analytics**: Dự báo doanh thu, tồn kho

***Chú thích:*** *Module phân tích sẽ cung cấp cái nhìn sâu sắc về hiệu suất kinh doanh, giúp đưa ra quyết định dựa trên dữ liệu.*

## 2.15. Customer Service

* **Live chat**: Tích hợp chatbot AI
* **Ticket system**: Quản lý yêu cầu hỗ trợ
* **FAQ & Knowledge base**: Tự phục vụ
* **Video call support**: Hỗ trợ trực tiếp cho VIP

***Chú thích:*** *Hệ thống dịch vụ khách hàng sẽ giúp nâng cao trải nghiệm và giữ chân khách hàng, đồng thời giảm tải cho đội ngũ hỗ trợ.*

## 2.16. Quản lý Identity Database và Persistence Database

* **Identity Database**: Quản lý thông tin người dùng, xác thực và phân quyền
* **Persistence Database**: Lưu trữ dữ liệu ứng dụng, bao gồm sản phẩm, đơn hàng, khách hàng
* **Tách biệt rõ ràng**: Hai cơ sở dữ liệu này sẽ được tách biệt để đảm bảo hiệu suất và bảo mật, đồng thời dễ dàng mở rộng trong tương lai

## 3. KIẾN TRÚC & CÔNG NGHỆ

### 3.1. Stack Công nghệ

* **Backend**: ASP.NET Core 6 (SDK 6.0.428)
* **Frontend**: Angular 16 (Angular CLI 16.2.16)
* **Database**: SQL Server
* **Cache**: Redis + HybridCache (In-Memory + Distributed)
* **Search (Optional)**: Elasticsearch
* **Message Queue (Optional)**: RabbitMQ

***Chú thích:*** *Stack được chọn để đảm bảo hiệu suất, bảo mật, khả năng mở rộng và hỗ trợ cộng đồng tốt.*

### 3.2. Kiến trúc Phần mềm

#### 3.2.1. Nguyên tắc cốt lõi

* **Clean Architecture**
  + Domain-centric: Đặt logic nghiệp vụ ở trung tâm
  + Tách biệt các mối quan tâm: Domain, Application, Infrastructure, Presentation
  + Dependency Rule: Các phụ thuộc chỉ hướng vào trong (domain core)

***Chú thích:*** *Clean Architecture giúp tạo ra code dễ bảo trì, dễ test và độc lập với các framework, database hay UI.*

* **Domain-Driven Design**
  + Aggregate Roots: Xác định ranh giới nhất quán của dữ liệu
  + Bounded Context: Phân chia hệ thống thành các ngữ cảnh có ý nghĩa
  + Entity và Value Objects: Phân biệt rõ ràng giữa các đối tượng có danh tính và không có danh tính

***Chú thích:*** *DDD giúp mô hình hóa các domain phức tạp và đảm bảo rằng code phản ánh đúng business rules.*

* **Modular Monolith**
  + Khả năng chuyển đổi sang Microservices sau này
  + Module separation: Tách biệt rõ ràng giữa các module
  + Shared kernel: Chia sẻ các khái niệm chung giữa các module

***Chú thích:*** *Modular Monolith là lựa chọn cân bằng giữa tính đơn giản của monolith và tính linh hoạt của microservices.*

#### 3.2.2. Sơ đồ Kiến trúc Tổng thể

---  
title: Tổng thể kiến trúc hệ thống ZPX  
---

flowchart LR %% Define Nodes subgraph UserFacing ["User Facing"]  
 direction LR USERS["Users<br/>(Web/Mobile)"]  
 FRONTEND["zylpelox-zpx-ui<br><span style='font-size:small; opacity:0.8;'>Angular SPA</span>"]  
 GATEWAY["API Gateway<br><span style='font-size:small; opacity:0.8;'>Optional</span>"]  
 end  
  
 subgraph BackendServices ["Backend Services (ZPX System)"]  
 direction TB API["ZylPelox.ZPX.Server<br/><span style='font-size:small; opacity:0.8;'>ASP.NET Core API</span>"]  
 APP["ZylPelox.ZPX.Application<br/><span style='font-size:small; opacity:0.8;'>Business Logic, CQRS</span>"]  
 DOMAIN["ZylPelox.ZPX.Domain<br/><span style='font-size:small; opacity:0.8;'>Entities, Core Logic</span>"]  
 INFRA["ZylPelox.ZPX.Infrastructure<br/><span style='font-size:small; opacity:0.8;'>Implementation Details</span>"]  
 end  
  
 subgraph ExternalDependencies ["External Dependencies"]  
 direction TB subgraph Persistence ["Persistence"]  
 DB[(SQL Server)]  
 CACHE[(Redis Cache)]  
 SEARCH[(Elasticsearch)]  
 end subgraph Messaging ["Messaging"]  
 MQ[(RabbitMQ)]  
 end subgraph ThirdParty ["3rd Party Services"]  
 direction TB PAYMENT[Payment Service]  
 SHIPPING[Shipping Service]  
 EMAIL[Email Service]  
 OTHER[...]  
 end end  
  
 %% Define Connections (Following Clean Architecture Dependencies)  
 USERS --> FRONTEND FRONTEND --> GATEWAY GATEWAY --> API  
  
 API --> APP APP --> DOMAIN APP --> INFRA  
  
 INFRA --> DB INFRA --> CACHE INFRA --> SEARCH INFRA --> MQ INFRA -.-> PAYMENT INFRA -.-> SHIPPING INFRA -.-> EMAIL INFRA -.-> OTHER  
  
 %% Styling (Optional - makes it visually distinct)  
 classDef user fill:#f0f9ff,stroke:#0ea5e9,stroke-width:1px,color:#0369a1;  
 classDef backend fill:#fdf2f8,stroke:#db2777,stroke-width:1px,color:#831843;  
 classDef external fill:#fefce8,stroke:#ca8a04,stroke-width:1px,color:#713f12;  
 classDef db fill:#eff6ff,stroke:#60a5fa,stroke-width:1.5px,color:#1e40af;  
 classDef thirdparty fill:#fffbeb,stroke:#f59e0b,stroke-width:1.5px,color:#78350f;  
  
 class USERS,FRONTEND,GATEWAY user;  
 class API,APP,DOMAIN,INFRA backend;  
 class DB,CACHE,SEARCH,MQ db;  
 class PAYMENT,SHIPPING,EMAIL,OTHER thirdparty;  
  
 style UserFacing fill:none,stroke:#0ea5e9,stroke-dasharray: 5 5;  
 style BackendServices fill:none,stroke:#db2777,stroke-dasharray: 5 5;  
 style ExternalDependencies fill:none,stroke:#ca8a04,stroke-dasharray: 5 5;  
 style Persistence fill:none,stroke:#60a5fa,stroke-width:1px;  
 style Messaging fill:none,stroke:#60a5fa,stroke-width:1px;  
 style ThirdParty fill:none,stroke:#f59e0b,stroke-width:1px;

2.

### 3.3. Design Patterns Chính

* **CQRS với MediatR**
  + Commands (Write): Thay đổi trạng thái hệ thống
  + Queries (Read): Truy vấn dữ liệu, không thay đổi trạng thái
  + Notifications (Events): Thông báo sự kiện đến các handler khác nhau

***Chú thích:*** *CQRS giúp tách biệt logic đọc và ghi, cho phép tối ưu hóa riêng cho mỗi loại, đặc biệt hữu ích cho hệ thống có tỉ lệ đọc/ghi chênh lệch.*

* **Repository & Specification Pattern**
  + Generic Repository: Interface chung cho việc truy xuất dữ liệu
  + Specification abstraction: Đóng gói logic truy vấn

***Chú thích:*** *Repository pattern giúp tách biệt logic truy cập dữ liệu khỏi domain logic, trong khi Specification pattern giúp tái sử dụng và kết hợp các điều kiện truy vấn.*

* **Domain Events**
  + Event publishing: Phát hành sự kiện khi có thay đổi trong domain
  + Event handling: Xử lý các tác động phụ do sự kiện gây ra

***Chú thích:*** *Domain Events cho phép các phần khác nhau của hệ thống phản ứng với những thay đổi mà không cần kết nối trực tiếp, giúp giảm sự phụ thuộc.*

### 3.4. Packages & Libraries Chính

#### 3.4.1. Backend (.NET)

* **MediatR**: Triển khai CQRS và xử lý sự kiện
* **FluentValidation**: Validation dựa trên fluent interface
* **AutoMapper**: Mapping giữa các đối tượng
* **FluentResults**: Xử lý kết quả với nhiều trạng thái
* **Serilog**: Logging có cấu trúc
* **Ardalis.Specification**: Triển khai pattern Specification
* **EFCore.BulkExtensions**: Tối ưu các thao tác hàng loạt
* **BCrypt.Net-Next**: Mã hóa mật khẩu
* **Swashbuckle**: Tài liệu API (Swagger)

***Chú thích:*** *Các thư viện được chọn dựa trên độ tin cậy, hiệu suất và mức độ hỗ trợ tốt từ cộng đồng.*

#### 3.4.2. Frontend (Angular)

* **Angular Material**: Thành phần UI
* **NgRx**: Quản lý state
* **RxJS**: Lập trình reactive
* **Angular Flex Layout**: Layout phản hồi
* **NgxTranslate**: Đa ngôn ngữ
* **Tailwind CSS**: Thiết kế giao diện

***Chú thích:*** *Các thư viện frontend được chọn để cung cấp trải nghiệm người dùng nhất quán và hiệu suất cao.*

### 3.5. Cấu trúc Solution

📁 ZylPelox  
 ├── 📁 Core  
 │ ├── 📁 ZylPelox.ZPX.Domain // Logic nghiệp vụ cốt lõi  
 │ │ ├── 📁 Entities // Các entity cơ bản  
 │ │ ├── 📁 Aggregates // Các aggregate roots  
 │ │ │ ├── 📁 Product  
 │ │ │ ├── 📁 Order  
 │ │ │ ├── 📁 Customer  
 │ │ │ ├── 📁 Inventory  
 │ │ │ └── 📁 ...  
 │ │ ├── 📁 JunctionEntities // Entity mối quan hệ nhiều-nhiều  
 │ │ ├── 📁 ValueObjects // Value objects  
 │ │ ├── 📁 Events // Domain events  
 │ │ ├── 📁 Exceptions // Domain exceptions  
 │ │ └── 📁 Services // Domain services  
 │ │  
 │ ├── 📁 ZylPelox.ZPX.Application // Logic ứng dụng  
 │ │ ├── 📁 Common  
 │ │ │ ├── 📁 Behaviors // Pipeline behaviors (validation, logging)  
 │ │ │ ├── 📁 Specifications // Query specifications  
 │ │ │ └── 📁 Exceptions // Application exceptions  
 │ │ ├── 📁 Contracts  
 │ │ │ ├── 📁 Persistence // Persistence interfaces  
 │ │ │ │ └── 📁 Repositories // Repository interfaces  
 │ │ │ ├── 📁 Infrastructure // Infrastructure interfaces  
 │ │ │ └── 📁 Identity // Identity interfaces  
 │ │ ├── 📁 Features // CQRS features  
 │ │ │ ├── 📁 Products // Product-related commands/queries  
 │ │ │ ├── 📁 Orders // Order-related commands/queries  
 │ │ │ ├── 📁 Customers // Customer-related commands/queries  
 │ │ │ ├── 📁 Inventory // Inventory-related commands/queries  
 │ │ │ └── 📁 ... // Other features  
 │ │ └── 📁 EventHandlers // Domain event handlers  
 | │  
 │ └── 📁 ZylPelox.ZPX.SharedKernel // Shared utilities and DTOs  
 │ ├── 📁 Constants // Shared constants  
 │ ├── 📁 Extensions // Extension methods  
 │ └── 📁 Utilities // Utility classes  
 │  
 ├── 📁 Infrastructure  
 │ ├── 📁 ZylPelox.ZPX.Infrastructure // Dịch vụ cơ sở hạ tầng  
 │ │ ├── 📁 Cache // Caching implementation  
 │ │ ├── 📁 Email // Email service  
 │ │ ├── 📁 FileStorage // File storage service  
 │ │ ├── 📁 MessageBroker // Message broker integration  
 │ │ └── 📁 ThirdPartyServices // Third-party integrations  
 │ │  
 │ ├── 📁 ZylPelox.ZPX.Persistence // Truy cập dữ liệu  
 │ │ ├── 📁 Configurations // EF Core configurations  
 │ │ ├── 📁 Repositories // Repository implementations  
 │ │ └── 📁 Migrations // Database migrations  
 │ │  
 │ └── 📁 ZylPelox.ZPX.Identity // Xác thực và phân quyền  
 │ ├── 📁 Models // Identity models  
 │ ├── 📁 Services // Identity services  
 │ └── 📁 Configurations // Identity configurations  
 │  
 ├── 📁 Presentation  
 │ ├── 📁 ZylPelox.ZPX.Server // Web API  
 │ │ ├── 📁 Controllers // API controllers  
 │ │ ├── 📁 Filters // Action filters  
 │ │ ├── 📁 Middleware // Custom middleware  
 │ │ └── 📁 Configurations // API configurations  
 │ │  
 │ └── 📁 ZylPelox.ZPX.UI // Angular frontend  
 │ ├── 📁 src  
 │ │ ├── 📁 app  
 │ │ │ ├── 📁 core // Core functionality  
 │ │ │ ├── 📁 shared // Shared components  
 │ │ │ └── 📁 features // Feature modules  
 │ │ ├── 📁 assets // Static assets  
 │ │ └── 📁 environments // Environment configurations  
 │ └── 📁 angular.json // Angular configuration  
 │  
 └── 📁 Tests // Tests  
 ├── 📁 ZylPelox.ZPX.UnitTests // Unit tests  
 ├── 📁 ZylPelox.ZPX.IntegrationTests // Integration tests  
 └── 📁 ZylPelox.ZPX.E2ETests // End-to-end tests

***Chú thích:*** *Cấu trúc tuân theo Clean Architecture, phụ thuộc hướng vào Core.*

## 3.6. Caching Architecture (Chi tiết hơn)

// Hybrid Cache Configuration  
public class CacheConfiguration  
{  
 public class CacheSettings  
 {  
 public int L1CacheSizeLimit { get; set; } = 100\_000; // 100MB  
 public TimeSpan L1DefaultExpiration { get; set; } = TimeSpan.FromMinutes(5);  
 public TimeSpan L2DefaultExpiration { get; set; } = TimeSpan.FromHours(1);  
  
 public Dictionary<string, CachePolicy> Policies { get; set; } = new()  
 {  
 ["Product"] = new() { L1TTL = TimeSpan.FromMinutes(10), L2TTL = TimeSpan.FromHours(2) },  
 ["Category"] = new() { L1TTL = TimeSpan.FromHours(1), L2TTL = TimeSpan.FromDays(1) },  
 ["User"] = new() { L1TTL = TimeSpan.FromMinutes(5), L2TTL = TimeSpan.FromMinutes(30) }  
 };  
 }  
}

## 3.7. Event-Driven Architecture

graph LR  
 subgraph "Domain Events"  
 A[OrderPlaced] --> B[Event Dispatcher]  
 C[PaymentReceived] --> B  
 D[InventoryUpdated] --> B  
 end  
  
 subgraph "Event Handlers"  
 B --> E[Email Service]  
 B --> F[Inventory Service]  
 B --> G[Analytics Service]  
 B --> H[Notification Service]  
 end  
  
 subgraph "External Systems"  
 E --> I[SendGrid]  
 F --> J[ERP System]  
 G --> K[Google Analytics]  
 H --> L[Firebase FCM]  
 end

***Chú thích:*** *Kiến trúc Event-Driven cho phép các phần của hệ thống phản ứng với các sự kiện mà không cần kết nối trực tiếp, giúp giảm độ phụ thuộc và tăng tính mở rộng.*

## 3.8. Security Architecture

### Authentication & Authorization Flow

sequenceDiagram  
 participant C as Client  
 participant API as API Gateway  
 participant Auth as Auth Service  
 participant Redis as Redis Cache  
 participant DB as Database  
  
 C->>API: Login Request  
 API->>Auth: Validate Credentials  
 Auth->>DB: Check User  
 DB-->>Auth: User Data  
 Auth->>Auth: Generate JWT + Refresh Token  
 Auth->>Redis: Store Refresh Token  
 Auth-->>API: Tokens  
 API-->>C: Access + Refresh Tokens  
  
 Note over C,API: Subsequent Requests  
 C->>API: Request + Access Token  
 API->>API: Validate JWT  
 API->>API: Check Permissions  
 API-->>C: Response

***Chú thích:*** *Quy trình xác thực sử dụng JWT và Refresh Token để đảm bảo bảo mật và khả năng mở rộng. Redis được sử dụng để lưu trữ Refresh Token, giúp giảm tải cho cơ sở dữ liệu.*

## 3.9. Resilience Patterns

### Circuit Breaker Implementation

public class PaymentServiceCircuitBreaker  
{  
 private readonly ICircuitBreaker \_circuitBreaker;  
  
 public PaymentServiceCircuitBreaker()  
 {  
 \_circuitBreaker = new CircuitBreaker(  
 failureThreshold: 5,  
 samplingDuration: TimeSpan.FromMinutes(1),  
 minimumThroughput: 10,  
 durationOfBreak: TimeSpan.FromMinutes(5)  
 );  
 }  
  
 public async Task<PaymentResult> ProcessPaymentAsync(PaymentRequest request)  
 {  
 return await \_circuitBreaker.ExecuteAsync(async () =>  
 {  
 // Call external payment service  
 return await ExternalPaymentService.ProcessAsync(request);  
 });  
 }  
}

***Chú thích:*** *Circuit Breaker giúp ngăn chặn các cuộc gọi đến dịch vụ bên ngoài khi nó gặp lỗi liên tục, bảo vệ hệ thống khỏi quá tải.*

### Retry Policy

services.AddHttpClient<IShippingService, ShippingService>()  
 .AddPolicyHandler(GetRetryPolicy())  
 .AddPolicyHandler(GetCircuitBreakerPolicy());  
  
static IAsyncPolicy<HttpResponseMessage> GetRetryPolicy()  
{  
 return HttpPolicyExtensions  
 .HandleTransientHttpError()  
 .WaitAndRetryAsync(  
 3,  
 retryAttempt => TimeSpan.FromSeconds(Math.Pow(2, retryAttempt)),  
 onRetry: (outcome, timespan, retryCount, context) =>  
 {  
 var logger = context.Values["logger"] as ILogger;  
 logger?.LogWarning($"Retry {retryCount} after {timespan} seconds");  
 });  
}

***Chú thích:*** *Retry Policy giúp tự động thử lại các cuộc gọi HTTP khi gặp lỗi tạm thời, cải thiện độ tin cậy của hệ thống.*

## 3.10. Performance Optimization

### Database Query Optimization

// Sử dụng Specification Pattern với EF Core  
public class ProductWithCategorySpecification : Specification<Product>  
{  
 public ProductWithCategorySpecification(bool includeVariants = false)  
 {  
 Query.Include(x => x.Category)  
 .Include(x => x.Brand)  
 .Where(x => x.IsActive && !x.IsDeleted);  
  
 if (includeVariants)  
 {  
 Query.Include(x => x.Variants.Where(v => v.IsActive));  
 }  
  
 // Sử dụng AsSplitQuery cho multiple includes  
 Query.AsSplitQuery();  
 }  
}  
  
// Bulk operations  
public async Task BulkUpdateInventoryAsync(List<InventoryUpdate> updates)  
{  
 await \_context.BulkUpdateAsync(updates, options =>  
 {  
 options.BatchSize = 1000;  
 options.UseTempDB = true;  
 });  
}

***Chú thích:*** *Sử dụng Specification Pattern giúp tái sử dụng logic truy vấn, trong khi Bulk Operations giúp tối ưu hóa các thao tác cập nhật hàng loạt.*

### Response Compression

// Program.cs  
builder.Services.AddResponseCompression(options =>  
{  
 options.EnableForHttps = true;  
 options.Providers.Add<BrotliCompressionProvider>();  
 options.Providers.Add<GzipCompressionProvider>();  
});  
  
builder.Services.Configure<BrotliCompressionProviderOptions>(options =>  
{  
 options.Level = CompressionLevel.Optimal;  
});

***Chú thích:*** *Response Compression giúp giảm kích thước dữ liệu trả về, cải thiện tốc độ tải trang và giảm băng thông.*

## 4. THIẾT KẾ CHI TIẾT

### 4.1. Domain Design

* **Entities & Base Classes**: BaseEntity (Guid Id), BaseAuditableEntity (thêm CreatedAt, CreatedBy, LastModifiedAt, LastModifiedBy, IsDeleted…).
* **Aggregate Roots**: để quản lý các entity liên quan như một đơn vị nhất quán.
* **Junction/Intermediate Entities**: ProductCategory, ProductTag, OrderDiscount… (quản lý quan hệ N-N).
* **Value Objects**: Money, Address, DateRange, ContactInformation… (bất biến, định nghĩa bởi thuộc tính).
* **Domain Services**: OrderProcessingService, PricingService… (chứa logic nghiệp vụ phức tạp không thuộc entity nào).
* **Domain Events**: OrderPlacedEvent, PaymentReceivedEvent… (thông báo các thay đổi quan trọng).

### 4.2. Application Layer Details

* **CQRS Implementation**: Commands (CreateProductCommand…), Queries (GetProductsQuery…), Handlers tương ứng.
* **DTOs & Mapping**: Request DTOs (CreateProductRequest…), Response DTOs (ProductDto…), Mapping Profiles (AutoMapper).
* **Validation**: Validators (CreateProductValidator…) với FluentValidation, Validation Behavior (MediatR Pipeline).
* **Application Services Interfaces**: IEmailService, IFileStorageService, ICacheService… (định nghĩa contracts cho infrastructure).

### 4.3. API Design & Endpoints

* **Nguyên tắc RESTful**: Resource-based URLs, HTTP methods đúng, status codes phù hợp, hỗ trợ Paging/Filtering/Sorting.
* **API Versioning**: URL-based (/api/v1/...).
* **API Documentation**: Tích hợp Swagger/OpenAPI, ví dụ request/response, changelog.
* **Controller Example**:
* [ApiController]  
  [Route("api/v1/[controller]")]  
  public class ProductsController : ControllerBase // Kế thừa từ ControllerBase hoặc BaseApiController tùy chỉnh  
  {  
   private readonly IMediator \_mediator; // Hoặc ISender/IPublisher  
    
   public ProductsController(IMediator mediator) => \_mediator = mediator;  
    
   [HttpGet]  
   [ProducesResponseType(typeof(PaginatedList<ProductDto>), StatusCodes.Status200OK)]  
   public async Task<IActionResult> GetProducts([FromQuery] GetProductsQuery query)  
   {  
   var result = await \_mediator.Send(query);  
   // Có thể thêm xử lý lỗi chung hoặc trả về Ok trực tiếp nếu dùng middleware/filter  
   return Ok(result);  
   }  
    
   // ... các endpoints khác (GET by Id, POST, PUT, DELETE) với xử lý lỗi và trả về ActionResult phù hợp  
  }

### 4.4. Database Design

#### 4.4.1. Tổng quan Schema

* Nhóm bảng liên quan đến Product (Products, Categories, Brands, ProductImages…).
* Nhóm bảng liên quan đến Order (Orders, OrderItems, OrderStatuses…).
* Nhóm bảng liên quan đến Customer (Customers, Addresses…).
* Nhóm bảng Inventory (Inventory, Warehouses, InventoryHistory…).
* Nhóm bảng Identity (AspNetUsers, AspNetRoles…).
* Nhóm bảng Marketing (Discounts…).
* Nhóm bảng Reviews (ProductReviews…).
* Bảng trung gian (Junction tables).

#### 4.4.2. Sơ đồ Quan hệ Thực thể (ERD)

---  
title: ERD (Entity Relationship Diagram) cho database  
---  
  
erDiagram  
 CUSTOMER ||--o{ ORDER : places  
 CUSTOMER ||--o{ ADDRESS : has  
 PRODUCT ||--o{ PRODUCT\_VARIANT : has  
 PRODUCT }o--o{ PRODUCT\_CATEGORY : "in (via junction)"  
 %% Many-to-Many requires junction table PRODUCT\_CATEGORY  
 CATEGORY }o--o{ PRODUCT\_CATEGORY : "contains (via junction)"  
  
 ORDER ||--o{ ORDER\_ITEM : includes  
 PRODUCT\_VARIANT ||--o{ ORDER\_ITEM : ordered  
  
 CUSTOMER {  
 GUID Id PK "Primary Key"  
 string Name  
 string Email "Unique identifier"  
 datetime CreatedAt  
 datetime UpdatedAt  
 }  
 ADDRESS {  
 GUID Id PK  
 GUID CustomerId FK "Links to CUSTOMER"  
 string Line1  
 string Line2 "Optional"  
 string City  
 string PostalCode  
 string Country  
 bool IsDefault  
 }  
 PRODUCT {  
 GUID Id PK  
 string Name  
 string Description "Optional"  
 string SKU "Base SKU"  
 decimal Price "Default price if no variants"  
 datetime CreatedAt  
 datetime UpdatedAt  
 }  
 PRODUCT\_VARIANT {  
 GUID Id PK  
 GUID ProductId FK "Links to PRODUCT"  
 string SKU "Unique Variant SKU"  
 string AttributesJSON "Optional, e.g., {'Color': 'Red', 'Size': 'L'}"  
 decimal Price "Specific variant price"  
 int StockQuantity  
 datetime CreatedAt  
 datetime UpdatedAt  
 }  
 CATEGORY {  
 GUID Id PK  
 string Name  
 string Slug "URL-friendly name"  
 GUID ParentCategoryId FK "Optional, For sub-categories"  
 datetime CreatedAt  
 datetime UpdatedAt  
 }  
 %% Junction table for Many-to-Many between Product and Category  
 PRODUCT\_CATEGORY {  
 GUID ProductId FK "Links to PRODUCT"  
 GUID CategoryId FK "Links to CATEGORY"  
 %% Composite Primary Key (ProductId, CategoryId) typically handled by DB constraint  
 }  
 "ORDER" {  
 GUID Id PK  
 GUID CustomerId FK "Links to CUSTOMER"  
 string OrderNumber "User-friendly identifier"  
 datetime OrderDate  
 string Status "e.g., Pending, Processing, Shipped, Delivered, Cancelled"  
 decimal TotalAmount  
 datetime CreatedAt  
 datetime UpdatedAt  
 }  
 ORDER\_ITEM {  
 GUID Id PK  
 GUID OrderId FK "Links to ORDER"  
 GUID ProductVariantId FK "Links to PRODUCT\_VARIANT"  
 int Quantity  
 decimal UnitPrice "Price at time of order"  
 decimal TotalPrice  
 }

***Chú thích:*** *Đây là ERD giản lược các thực thể và mối quan hệ chính. Xem* ***Phụ lục B*** *để có chi tiết đầy đủ về schema.*

#### 4.4.3. Entity Framework Core Configuration Example

public class ProductConfiguration : IEntityTypeConfiguration<Product>  
{  
 public void Configure(EntityTypeBuilder<Product> builder)  
 {  
 builder.ToTable("Products"); // Đặt tên bảng rõ ràng  
 builder.HasKey(p => p.Id);  
  
 builder.Property(p => p.Name).IsRequired().HasMaxLength(200);  
 builder.Property(p => p.Price).HasColumnType("decimal(18,2)"); // Định nghĩa kiểu dữ liệu DB  
  
 // Quan hệ với Variants  
 builder.HasMany(p => p.Variants)  
 .WithOne(v => v.Product)  
 .HasForeignKey(v => v.ProductId)  
 .OnDelete(DeleteBehavior.Cascade); // Hoặc Restrict tùy logic nghiệp vụ  
  
 // Quan hệ với Category (1-N)  
 builder.HasOne(p => p.Category)  
 .WithMany(c => c.Products)  
 .HasForeignKey(p => p.CategoryId)  
 .OnDelete(DeleteBehavior.SetNull); // Hoặc Restrict  
  
 // Cấu hình Index  
 builder.HasIndex(p => p.Name);  
 builder.HasIndex(p => p.Sku).IsUnique(); // Nếu SKU là duy nhất  
  
 // Cấu hình Soft Delete Filter  
 builder.HasQueryFilter(p => !p.IsDeleted);  
 }  
}

#### 4.4.4. Migration Strategy

* Code-First approach, Incremental migrations, Schema versioning, Data seeding cho lookup tables.

#### 4.4.5. Performance Considerations

* Indexing phù hợp, Batch operations (BulkExtensions), Tối ưu truy vấn đọc (CQRS), Tránh N+1 problem (Eager/Explicit loading).

### 4.5. Frontend Architecture (ZylPelox.ZPX.UI)

* **Terminal**:
* ng new zylpelox-zpx-ui --routing --style=scss --skip-git --package-manager=npm --directory=src/Presentation/ZylPelox.ZPX.UI
* **Cấu trúc**: Core Module, Shared Module, Feature Modules (lazy-loaded).
* **State Management**: NgRx (cho state phức tạp/toàn cục), Services + RxJS (cho state đơn giản), Local component state.
* **UI Components**: Base trên Angular Material, Custom components mở rộng, Responsive (Flex Layout).
* **Routing & Navigation**: Feature-based, Route Guards, Lazy-loading.
* **Internationalization (I18n)**: NgxTranslate, định dạng currency/date theo locale, hỗ trợ RTL.
* **Tailwind CSS**: Sử dụng Tailwind CSS cho styling, Responsive design, Utility-first approach.

## 5. CÁC VẤN ĐỀ XUYÊN SUỐT (CROSS-CUTTING CONCERNS)

### 5.1. Error Handling & Logging

* **Exception Handling**: Custom domain exceptions with prefix name [Zpx]NameExceptions.cs (ZpxNotFoundException…), Global exception middleware, Structured error responses.
* **Logging Strategy**: Structured logging (Serilog), Log levels, Log enrichment, Log storage/rotation.
* **Monitoring**: Health checks, Application metrics, Performance counters, API usage statistics.

### 5.2. Security Considerations

* **Authentication & Authorization**: JWT, Role-based, Policy-based, Refresh token strategy.
* **Data Protection**: HTTPS, Encryption (at rest/in transit), Input validation/sanitization, CSRF protection.
* **API Security**: Rate limiting, Request validation, API keys, CORS configuration.

### 5.3. Caching Strategy

* **Cache Levels**: Memory cache (L1), Distributed cache - Redis (L2), Browser caching.
* **Cache Invalidation**: Time-based, Event-based, Cache dependencies.
* **Cacheable Resources**: Product catalog, User preferences, Lookup data, Session data.

### 5.4. E2ETests

* **End-to-End Testing**: Cypress, Test critical user flows, UI/UX validation, API contract tests.
* **Test Scenarios**: User registration, Product search, Order placement, Payment processing, Admin operations.

### 5.5. Registration Service

* **Name Static class**: [ProjectName]ServiceRegistration.cs
* **Register Services**: Use IServiceCollection to register all services, repositories, and configurations.

### 5.6. Nuget Package

* **Domain Layer**: Ardalis.GuardClauses, MediatR.Contracts,…
* **Application Layer**: MediatR, FluentValidation, AutoMapper, FluentResults,…
* **Infrastructure Layer**: Serilog,…
* **Server Layer**: Serilog.AspNetCore,…

## 6. QUY TRÌNH PHÁT TRIỂN & VẬN HÀNH

### 6.1. Testing Approach

* **Unit Testing**: Domain logic, Application handlers, Services (xUnit, Moq).
* **Integration Testing**: Repositories, API endpoints, Database interactions (xUnit, TestContainers/In-Memory DB).
* **E2E Testing**: Critical user flows, UI/UX (Cypress/Playwright).

### 6.2. Deployment & DevOps

* **CI/CD Pipeline**: Automated builds (GitHub Actions/Azure DevOps), automated testing, static code analysis, deployment automation.
* **Environments**: Development, Testing/QA, Staging, Production.
* **Deployment Strategy**: Docker containerization, Azure App Service/Kubernetes, Blue-green/Canary deployment.

### 6.3. Coding Standards & Quality

* **Naming Conventions**: Nhất quán (DTOs, Specs, Exceptions…).
* **Code Quality Tools**: SonarLint/SonarQube, StyleCop, EditorConfig, Quality gates in CI/CD.
* **Documentation**: Code comments (XML Doc), Architecture docs, API docs (Swagger), User guides.
* **Development Practices**: Code reviews, Pull request workflow, Feature branch strategy, TDD (khi phù hợp).

## 7. QUẢN LÝ DỰ ÁN

### 7.1. Roadmap & Phasing

* **Phase 0: Planning & Setup (Month 0)**: Xác định yêu cầu chi tiết, Thiết lập môi trường phát triển, Kiến trúc tổng thể.
* **Phase 1: MVP (Month 1-3)**: Catalog, Order cơ bản, Customer accounts, Admin dashboard, 1-2 Payment providers.
* **Phase 2: Enhanced Features (Month 4-6)**: Inventory, Advanced search, Promotions, Reviews, Thêm payment options.
* **Phase 3: Advanced Features (Month 7-9)**: Analytics, Marketing tools, Multi-vendor (opt), Mobile app (opt), 3rd Party API.
* **Phase 4: Optimization & Scaling (Month 10+)**: Performance tuning, Security hardening, Scalability improvements, Monitoring setup.
* **Phase 5: Maintenance & Support (Ongoing)**: Bug fixes, Feature requests, Technical debt management, Regular updates.
* **Phase 6: Review & Retrospective (Ongoing)**: Đánh giá quy trình phát triển, Học hỏi từ các vấn đề gặp phải, Cải tiến quy trình.
* **Phase 7: Future Planning (Ongoing)**: Lên kế hoạch cho các tính năng mới, Cập nhật công nghệ, Mở rộng hệ thống.
* **Phase 8: Community Engagement (Ongoing)**: Tương tác với cộng đồng người dùng, Thu thập phản hồi, Cải tiến dựa trên nhu cầu thực tế.
* **Phase 9: Documentation & Training (Ongoing)**: Cập nhật tài liệu, Hướng dẫn sử dụng, Đào tạo cho người dùng và nhà phát triển mới.
* **Phase 10: Long-term Support (LTS) Planning (Ongoing)**: Lên kế hoạch cho các phiên bản hỗ trợ lâu dài, Bảo trì và cập nhật định kỳ.
* **Phase 11: Scalability & Performance Enhancements (Ongoing)**: Tối ưu hóa hiệu suất, Mở rộng quy mô hệ thống, Cải tiến kiến trúc khi cần thiết.
* **Phase 12: Innovation & New Features (Ongoing)**: Nghiên cứu công nghệ mới, Thêm các tính năng sáng tạo, Cập nhật theo xu hướng thị trường.
* **Phase 13: Community Contributions (Ongoing)**: Khuyến khích cộng đồng đóng góp mã nguồn, Tích hợp các tính năng từ cộng đồng, Quản lý pull requests.
* **Phase 14: Legacy Code Management (Ongoing)**: Quản lý mã nguồn cũ, Cải tiến và refactor khi cần thiết, Đảm bảo tính tương thích ngược.
* **Phase 15: Compliance & Legal Considerations (Ongoing)**: Đảm bảo tuân thủ các quy định pháp lý, Bảo vệ dữ liệu người dùng, Quản lý giấy phép phần mềm.
* Use **Agile/Scrum**: Sprints, Backlog grooming, Daily standups, Sprint reviews/retrospectives.

### 7.2. Resource Requirements & Timeline

#### 7.2.1. Development Team

* 1 Tech Lead/Architect, 3-4 Backend Devs, 2 Frontend Devs, 1 QA Engineer, 1 DevOps (part-time).

#### 7.2.2. Infrastructure

* Dev, Test/QA, Staging, Production environments, CI/CD pipeline, Source control.

#### 7.2.3. Estimated Timeline

* Planning & Setup: 2-3 weeks
* MVP Development: 2-3 months
* Testing & Refinement: 1 month
* Enhanced Features: 3 months
* Advanced Features: 3 months

## 7.3. Agile Implementation Details

### Sprint Planning Template

## Sprint [Number] Planning - [Start Date] to [End Date]  
  
### Sprint Goal  
  
[Mô tả mục tiêu chính của sprint]  
  
### Team Capacity  
  
- Total capacity: [X] story points  
- Reserved for bugs/support: [Y] story points  
- Available for new features: [Z] story points  
  
### Committed User Stories  
  
| ID | Story | Priority | Points | Assignee |  
| ------ | --------------------------- | -------- | ------ | -------- |  
| US-001 | As a customer, I want to... | High | 5 | Dev1 |  
| US-002 | As an admin, I want to... | Medium | 3 | Dev2 |  
  
### Technical Tasks  
  
| ID | Task | Related Story | Hours | Assignee |  
| ------ | -------------------- | ------------- | ----- | -------- |  
| TT-001 | Setup Redis caching | US-001 | 8 | Dev3 |  
| TT-002 | Implement unit tests | US-001 | 4 | Dev1 |  
  
### Definition of Done  
  
- [ ] Code reviewed by at least 1 peer  
- [ ] Unit tests written and passing (coverage > 80%)  
- [ ] Integration tests passing  
- [ ] Documentation updated  
- [ ] No SonarLint critical/major issues  
- [ ] Deployed to staging environment  
- [ ] Acceptance criteria verified by PO

### Risk Management Matrix

## Project Risks  
  
| Risk | Probability | Impact | Mitigation Strategy | Owner |  
| ------------------------------------- | ----------- | --------- | ----------------------------------------------------------------------- | ----------------- |  
| Third-party payment gateway downtime | Medium | High | Implement multiple payment providers, circuit breaker pattern | Tech Lead |  
| Team member leaves project | Low | High | Knowledge sharing sessions, comprehensive documentation | PM |  
| Performance issues with large catalog | Medium | Medium | Early performance testing, caching strategy, database optimization | Backend Lead |  
| Security vulnerabilities | Low | Very High | Regular security audits, OWASP compliance, penetration testing | Security Champion |  
| Scope creep | High | Medium | Clear requirements, change control process, regular stakeholder reviews | PO |

## 7.4. Team Structure & Communication

### RACI Matrix

| Activity | Product Owner | Tech Lead | Backend Dev | Frontend Dev | QA | DevOps |  
| ----------------------- | ------------- | --------- | ----------- | ------------ | --- | ------ |  
| Requirements Definition | R/A | C | I | I | C | I |  
| Architecture Design | C | R/A | C | C | I | C |  
| API Development | I | C | R/A | C | C | I |  
| UI Development | I | C | I | R/A | C | I |  
| Testing | I | I | C | C | R/A | I |  
| Deployment | I | C | I | I | C | R/A |  
| Monitoring Setup | I | C | I | I | I | R/A |  
  
R = Responsible, A = Accountable, C = Consulted, I = Informed

### Communication Plan

## Communication Schedule  
  
### Daily  
  
- \*\*Stand-up Meeting\*\*: 9:30 AM (15 min)  
 - Format: What I did, What I'll do, Blockers  
 - Platform: MS Teams/Zoom  
  
### Weekly  
  
- \*\*Sprint Review\*\* (End of Sprint): Friday 2:00 PM (1 hour)  
- \*\*Sprint Retrospective\*\*: Friday 3:30 PM (45 min)  
- \*\*Sprint Planning\*\*: Monday 10:00 AM (2 hours)  
- \*\*Technical Sync\*\*: Wednesday 2:00 PM (30 min)  
  
### Bi-weekly  
  
- \*\*Stakeholder Update\*\*: Every other Thursday 3:00 PM  
- \*\*Architecture Review\*\*: Every other Tuesday 2:00 PM  
  
### Communication Channels  
  
- \*\*Urgent Issues\*\*: Phone/SMS  
- \*\*Daily Communication\*\*: Slack/Teams  
- \*\*Documentation\*\*: Confluence/Wiki  
- \*\*Code Review\*\*: GitHub/GitLab  
- \*\*Task Tracking\*\*: Jira/Azure DevOps

## 7.5. Quality Metrics & KPIs

### Development Metrics

quality\_metrics:  
 code\_quality:  
 - metric: Code Coverage  
 target: "> 80%"  
 measurement: "SonarQube"  
 - metric: Technical Debt Ratio  
 target: "< 5%"  
 measurement: "SonarQube"  
 - metric: Duplicated Lines  
 target: "< 3%"  
 measurement: "SonarQube"  
  
 performance:  
 - metric: API Response Time (P95)  
 target: "< 200ms"  
 measurement: "Application Insights"  
 - metric: Page Load Time  
 target: "< 3s"  
 measurement: "Google PageSpeed"  
 - metric: Database Query Time (P95)  
 target: "< 100ms"  
 measurement: "SQL Profiler"  
  
 reliability:  
 - metric: Uptime  
 target: "> 99.9%"  
 measurement: "Pingdom/UptimeRobot"  
 - metric: Error Rate  
 target: "< 0.1%"  
 measurement: "Application Insights"  
 - metric: Mean Time To Recovery (MTTR)  
 target: "< 30 minutes"  
 measurement: "Incident Reports"  
  
business\_metrics:  
 - metric: Sprint Velocity  
 target: "Stable or increasing"  
 measurement: "Jira/Azure DevOps"  
 - metric: Defect Escape Rate  
 target: "< 5%"  
 measurement: "Bug tracking system"  
 - metric: Customer Satisfaction (CSAT)  
 target: "> 4.5/5"  
 measurement: "User surveys"

## 7.6. Release Management

### Release Checklist

## Release [Version] Checklist  
  
### Pre-Release (1 week before)  
  
- [ ] Feature freeze announced  
- [ ] All features code complete  
- [ ] Integration tests passing  
- [ ] Performance testing completed  
- [ ] Security scan completed  
- [ ] Release notes drafted  
  
### Release Day  
  
- [ ] Final regression testing  
- [ ] Database migration scripts tested  
- [ ] Rollback plan documented  
- [ ] Monitoring alerts configured  
- [ ] Communication sent to stakeholders  
  
### Post-Release  
  
- [ ] Smoke tests on production  
- [ ] Monitor error rates (first 24 hours)  
- [ ] Gather user feedback  
- [ ] Document lessons learned  
- [ ] Update documentation

### Version Numbering Strategy

MAJOR.MINOR.PATCH-PRERELEASE+BUILD  
  
Examples:  
- 1.0.0 - First stable release  
- 1.1.0 - New features added  
- 1.1.1 - Bug fixes  
- 2.0.0-beta.1 - Breaking changes in beta  
- 1.2.0-rc.1+20230615 - Release candidate with build info

## 7.7. Documentation Standards

### Code Documentation Template

/// <summary>  
/// Processes an order and handles payment transaction.  
/// </summary>  
/// <param name="orderId">The unique identifier of the order to process.</param>  
/// <param name="paymentDetails">Payment information including method and credentials.</param>  
/// <returns>A task that represents the asynchronous operation.  
/// The task result contains the order processing result.</returns>  
/// <exception cref="ZpxNotFoundException">Thrown when the order is not found.</exception>  
/// <exception cref="ZpxPaymentException">Thrown when payment processing fails.</exception>  
/// <example>  
/// <code>  
/// var result = await orderService.ProcessOrderAsync(  
/// orderId: Guid.Parse("123e4567-e89b-12d3-a456-426614174000"),  
/// paymentDetails: new PaymentDetails { ... }  
/// );  
/// </code>  
/// </example>  
public async Task<OrderProcessingResult> ProcessOrderAsync(  
 Guid orderId,  
 PaymentDetails paymentDetails)  
{  
 // Implementation  
}

### API Documentation Standard

openapi: 3.0.3  
info:  
 title: ZylPelox API  
 version: 1.0.0  
 description: |  
 # Introduction  
 ZylPelox API provides comprehensive e-commerce functionality.  
  
 ## Authentication  
 Use Bearer token authentication. Obtain token via /auth/login endpoint.  
  
 ## Rate Limiting  
 - Anonymous: 100 requests/hour  
 - Authenticated: 1000 requests/hour  
  
 ## Pagination  
 Use `pageNumber` and `pageSize` query parameters.  
  
paths:  
 /api/v1/products:  
 get:  
 summary: Get products list  
 operationId: getProducts  
 tags:  
 - Products  
 parameters:  
 - $ref: "#/components/parameters/PageNumber"  
 - $ref: "#/components/parameters/PageSize"  
 - name: categoryId  
 in: query  
 schema:  
 type: string  
 format: uuid  
 responses:  
 "200":  
 description: Success  
 content:  
 application/json:  
 schema:  
 $ref: "#/components/schemas/ProductListResponse"

## 7.8. Budget & Resource Planning

### Development Cost Estimation

## Project Budget Breakdown  
  
### Human Resources (6 months)  
  
| Role | Count | Monthly Cost | Total |  
| ------------------ | ----- | ------------ | ------------ |  
| Tech Lead | 1 | $8,000 | $48,000 |  
| Backend Developer | 3 | $6,000 | $108,000 |  
| Frontend Developer | 2 | $5,500 | $66,000 |  
| QA Engineer | 1 | $4,500 | $27,000 |  
| DevOps (part-time) | 0.5 | $6,000 | $18,000 |  
| \*\*Subtotal\*\* | | | \*\*$267,000\*\* |  
  
### Infrastructure & Tools  
  
| Item | Monthly Cost | 6-Month Total |  
| --------------------- | ------------ | ------------- |  
| Azure (Dev/Test/Prod) | $2,500 | $15,000 |  
| Third-party services | $500 | $3,000 |  
| Development tools | $300 | $1,800 |  
| \*\*Subtotal\*\* | | \*\*$19,800\*\* |  
  
### Contingency (15%)  
  
$43,020  
  
### \*\*Total Project Budget\*\*  
  
$329,820

## 7.9. Success Criteria

### MVP Success Metrics

## Definition of Success - MVP Phase  
  
### Technical Success  
  
- [ ] All critical features implemented and tested  
- [ ] Performance meets defined SLAs  
- [ ] Security audit passed with no critical issues  
- [ ] 99.9% uptime achieved in first month  
  
### Business Success  
  
- [ ] 1000+ registered users in first month  
- [ ] 100+ orders processed successfully  
- [ ] < 2% cart abandonment rate improvement  
- [ ] Positive feedback from beta users (>80% satisfaction)  
  
### Team Success  
  
- [ ] On-time delivery (±10% of estimated timeline)  
- [ ] Within budget (±15% of allocated budget)  
- [ ] Team satisfaction score > 4/5  
- [ ] Knowledge transfer completed

## 8. PHỤ LỤC

### Phụ lục A: Diagram Quy trình Nghiệp vụ

#### A.1. Sequence Diagram: Luồng “Đặt hàng” (Order Placement)

---  
title: Luồng "Đặt hàng" (Order Placement)  
---  
sequenceDiagram  
 actor User  
 participant FE as "Frontend (ui)"  
 participant GW as "API Gateway (Optional)"  
 participant API as "API Server (Server)"  
 participant APP as "Application (Application)"  
 participant DM as "Domain (Domain)"  
 participant DB as "Persistence (Persistence)"  
 participant PAY as "Payment Service (Infrastructure)"  
 participant EMAIL as "Email Service (Infrastructure)"  
  
 User->>+FE: 1. Chọn sản phẩm, điền thông tin, nhấn "Đặt hàng"  
 FE->>+GW: 2. POST /api/v1/orders (Auth Token, Order Data)  
 GW->>+API: 3. Forward POST /api/v1/orders  
 API->>+APP: 4. Execute CreateOrderCommand (CQRS)  
 APP->>+DM: 5. Validate Order Data (Business Rules)  
 DM-->>-APP: 6. Validation Result (Success/Fail)  
 alt Order is Valid  
 APP->>DM: 7. Create Order Aggregate (Entities: Order, OrderItems)  
 APP->>+DB: 8. SaveChangesAsync() (Unit of Work Pattern)  
 DB-->>-APP: 9. Order Saved (ID generated)  
 APP->>+PAY: 10. ProcessPayment(OrderId, Amount, PaymentDetails)  
 PAY-->>-APP: 11. PaymentResult (Success/Fail)  
 alt Payment Successful  
 APP->>+DB: 12. Update Order Status ('Processing' or 'Paid')  
 DB-->>-APP: 13. Status Updated  
 APP->> DM: 14. Raise OrderPlaced Domain Event  
 APP->>+EMAIL: 15. SendOrderConfirmationEmail(OrderId, CustomerEmail)  
 EMAIL-->>-APP: 16. Email Queued/Sent  
 APP-->>API: 17. Order Creation Success (Order Details)  
 else Payment Failed  
 APP->>+DB: 12a. Update Order Status ('PaymentFailed')  
 DB-->>-APP: 13a. Status Updated  
 APP-->>API: 17a. Order Creation Failed (Payment Error)  
 end  
 else Order is Invalid  
 APP-->>API: 17b. Order Creation Failed (Validation Errors)  
 end  
 API-->>-GW: 18. API Response  
 GW-->>-FE: 19. Forward API Response  
 FE->>-User: 20. Hiển thị trạng thái đặt hàng (Success/Error Message)

#### A.2. Sequence Diagram: Luồng “Quản lý tồn kho tự động” (Auto Inventory Adjustment)

---  
title: Luồng "Quản lý tồn kho tự động" (Auto Inventory Adjustment via Domain Event)  
---  
sequenceDiagram  
 participant APP as "Application Layer (Order Service Context)"  
 participant DM as "Domain Layer"  
 participant DB as "Persistence (Persistence)"  
 participant MQ as "Event Bus (e.g., RabbitMQ via Infrastructure)"  
 participant InventoryHandler as "InventoryEventHandler (App Layer - Inventory Context)"  
 participant Notify as "NotificationService (Infrastructure)"  
 participant Admin as "Admin User/System"  
  
 Note over APP, DB: Order has been successfully placed and saved.  
 APP->>+DM: 1. Raise OrderPlaced Domain Event  
 DM-->>-APP: (Event raised within Domain)  
  
 Note over APP, MQ: Application Layer (or Domain Event Dispatcher) handles publishing events.  
 APP->>+MQ: 2. Publish(OrderPlacedEvent)  
  
 Note right of MQ: Event Bus delivers event to interested subscribers.  
 MQ->>+InventoryHandler: 3. Receive(OrderPlacedEvent)  
  
 InventoryHandler->>+DB: 4. Get ProductVariant data for OrderItems  
 DB-->>-InventoryHandler: 5. ProductVariant details (incl. current stock)  
  
 loop For Each OrderItem in Event  
 InventoryHandler->>InventoryHandler: 6. Calculate new stock level  
 InventoryHandler->>+DB: 7. Update ProductVariant Stock (e.g., Stock = Stock - Quantity)  
 DB-->>-InventoryHandler: 8. Stock updated  
 alt Stock Level is Low (below threshold)  
 InventoryHandler->>+MQ: 9. Publish(LowStockDetectedEvent)  
 MQ->>+Notify: 10. Receive(LowStockDetectedEvent)  
 Notify->>Admin: 11. Send Low Stock Notification (Email, SMS, etc.)  
 end  
 end  
 InventoryHandler-->>-MQ: (Processing Complete)

### Phụ lục B: Chi tiết Schema Database

**Ngày hiện tại:** 2024-06-15 19:10:01 **Người dùng:** phamtiendungcw

#### Tổng quan các nhóm bảng

1. **Identity Tables** - Quản lý người dùng và xác thực
2. **Core eCommerce Tables** - Sản phẩm, danh mục, đơn hàng
3. **Customer Management Tables** - Khách hàng, địa chỉ
4. **Inventory & Stock Tables** - Quản lý hàng tồn kho
5. **Review & Rating Tables** - Đánh giá sản phẩm
6. **Marketing Tables** - Khuyến mãi, mã giảm giá
7. **Logging & Audit Tables** - Ghi log hệ thống

#### Identity Tables

##### 1. AspNetUsers (Bảng Identity mặc định)

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | nvarchar(450) | PK | Định danh người dùng |
| UserName | nvarchar(256) |  | Tên đăng nhập |
| NormalizedUserName | nvarchar(256) | INDEX | Tên đăng nhập chuẩn hóa |
| Email | nvarchar(256) |  | Email |
| NormalizedEmail | nvarchar(256) | INDEX | Email chuẩn hóa |
| EmailConfirmed | bit |  | Email đã xác nhận |
| PasswordHash | nvarchar(max) |  | Mật khẩu đã hash |
| SecurityStamp | nvarchar(max) |  | Dùng cho bảo mật |
| ConcurrencyStamp | nvarchar(max) |  | Kiểm soát đồng thời |
| PhoneNumber | nvarchar(max) |  | Số điện thoại |
| PhoneNumberConfirmed | bit |  | SĐT đã xác nhận |
| TwoFactorEnabled | bit |  | Xác thực 2 yếu tố |
| LockoutEnd | datetimeoffset |  | Thời gian khóa hết hạn |
| LockoutEnabled | bit |  | Khóa tài khoản |
| AccessFailedCount | int |  | Số lần đăng nhập sai |
| FirstName | nvarchar(100) |  | Tên |
| LastName | nvarchar(100) |  | Họ |
| ProfilePictureUrl | nvarchar(500) |  | URL ảnh đại diện |

##### 2. AspNetRoles

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | nvarchar(450) | PK | Định danh vai trò |
| Name | nvarchar(256) |  | Tên vai trò |
| NormalizedName | nvarchar(256) | INDEX | Tên vai trò chuẩn hóa |
| ConcurrencyStamp | nvarchar(max) |  | Kiểm soát đồng thời |

##### 3. AspNetUserRoles

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| UserId | nvarchar(450) | PK, FK | ID người dùng |
| RoleId | nvarchar(450) | PK, FK | ID vai trò |

##### 4. AspNetUserClaims, AspNetRoleClaims, AspNetUserLogins, AspNetUserTokens

***Chú thích:*** *Các bảng Identity khác được tự động tạo bởi EFCore, không cần cấu hình thêm.*

#### 🛒 Core eCommerce Tables

##### 5. Products

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | uniqueidentifier | PK | Định danh sản phẩm |
| Name | nvarchar(200) | NOT NULL, INDEX | Tên sản phẩm |
| Description | nvarchar(max) |  | Mô tả sản phẩm |
| ShortDescription | nvarchar(500) |  | Mô tả ngắn |
| Price | decimal(18,2) | NOT NULL | Giá gốc |
| OldPrice | decimal(18,2) |  | Giá cũ (khi giảm giá) |
| SpecialPrice | decimal(18,2) |  | Giá đặc biệt |
| SpecialPriceStartDate | datetime2 |  | Ngày bắt đầu giá đặc biệt |
| SpecialPriceEndDate | datetime2 |  | Ngày kết thúc giá đặc biệt |
| Sku | nvarchar(100) | INDEX, UNIQUE | Stock Keeping Unit |
| Gtin | nvarchar(100) | INDEX | Global Trade Item Number |
| CategoryId | uniqueidentifier | FK, INDEX | ID danh mục |
| BrandId | uniqueidentifier | FK, INDEX | ID thương hiệu |
| IsActive | bit | NOT NULL, DEFAULT(1) | Trạng thái kích hoạt |
| IsFeatured | bit | NOT NULL, DEFAULT(0) | Sản phẩm nổi bật |
| IsDeleted | bit | NOT NULL, DEFAULT(0) | Trạng thái xóa mềm |
| CreatedAt | datetime2 | NOT NULL, DEFAULT(GETUTCDATE()) | Ngày tạo |
| CreatedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người tạo |
| ModifiedAt | datetime2 |  | Ngày sửa đổi |
| ModifiedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người sửa đổi |
| DeletedAt | datetime2 |  | Ngày xóa |
| DeletedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người xóa |
| StockQuantity | int | NOT NULL, DEFAULT(0) | **(Moved to ProductVariant)** |
| Weight | decimal(18,4) |  | **(Moved to ProductVariant)** |
| Length | decimal(18,4) |  | **(Moved to ProductVariant)** |
| Width | decimal(18,4) |  | **(Moved to ProductVariant)** |
| Height | decimal(18,4) |  | **(Moved to ProductVariant)** |

##### 6. ProductVariants (NEW/MODIFIED Table for Variants)

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | uniqueidentifier | PK | Định danh biến thể |
| ProductId | uniqueidentifier | FK (Products.Id), NOT NULL, INDEX | ID sản phẩm cha |
| Name | nvarchar(200) |  | Tên biến thể (nếu có) |
| Sku | nvarchar(100) | INDEX, UNIQUE | SKU của biến thể |
| Price | decimal(18,2) | NOT NULL | Giá của biến thể |
| OldPrice | decimal(18,2) |  | Giá cũ của biến thể |
| StockQuantity | int | NOT NULL, DEFAULT(0) | Số lượng tồn kho |
| ReservedQuantity | int | NOT NULL, DEFAULT(0) | Số lượng đặt trước |
| Weight | decimal(18,4) |  | Trọng lượng |
| Length | decimal(18,4) |  | Chiều dài |
| Width | decimal(18,4) |  | Chiều rộng |
| Height | decimal(18,4) |  | Chiều cao |
| AttributesJSON | nvarchar(max) |  | Thuộc tính (JSON) |
| ImageUrl | nvarchar(500) |  | Ảnh đại diện biến thể |
| IsActive | bit | NOT NULL, DEFAULT(1) | Trạng thái kích hoạt |
| CreatedAt | datetime2 | NOT NULL, DEFAULT(GETUTCDATE()) | Ngày tạo |
| CreatedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người tạo |
| ModifiedAt | datetime2 |  | Ngày sửa đổi |
| ModifiedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người sửa đổi |

##### 7. ProductImages

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | uniqueidentifier | PK | Định danh ảnh |
| ProductId | uniqueidentifier | FK (Products.Id), NOT NULL, INDEX | ID sản phẩm |
| ProductVariantId | uniqueidentifier | FK (ProductVariants.Id), INDEX | ID biến thể (nếu ảnh riêng cho variant) |
| ImageUrl | nvarchar(500) | NOT NULL | URL ảnh |
| AltText | nvarchar(200) |  | Văn bản thay thế |
| DisplayOrder | int | NOT NULL, DEFAULT(0) | Thứ tự hiển thị |
| CreatedAt | datetime2 | NOT NULL, DEFAULT(GETUTCDATE()) | Ngày tạo |
| CreatedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người tạo |

##### 8. ProductAttributes (Consider using JSON in Product or Variant)

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | uniqueidentifier | PK | Định danh thuộc tính |
| ProductId | uniqueidentifier | FK (Products.Id), NOT NULL, INDEX | ID sản phẩm |
| AttributeName | nvarchar(100) | NOT NULL | Tên thuộc tính |
| AttributeValue | nvarchar(500) | NOT NULL | Giá trị thuộc tính |
| DisplayOrder | int | NOT NULL, DEFAULT(0) | Thứ tự hiển thị |
| CreatedAt | datetime2 | NOT NULL, DEFAULT(GETUTCDATE()) | Ngày tạo |
| CreatedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người tạo |

##### 9. Categories

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | uniqueidentifier | PK | Định danh danh mục |
| Name | nvarchar(200) | NOT NULL, INDEX | Tên danh mục |
| Description | nvarchar(500) |  | Mô tả |
| ParentId | uniqueidentifier | FK (Categories.Id), INDEX | ID danh mục cha |
| DisplayOrder | int | NOT NULL, DEFAULT(0) | Thứ tự hiển thị |
| IsActive | bit | NOT NULL, DEFAULT(1) | Trạng thái kích hoạt |
| ImageUrl | nvarchar(500) |  | URL hình ảnh |
| Slug | nvarchar(200) | NOT NULL, INDEX, UNIQUE | URL thân thiện |
| MetaTitle | nvarchar(200) |  | Tiêu đề SEO |
| MetaKeywords | nvarchar(500) |  | Từ khóa SEO |
| MetaDescription | nvarchar(500) |  | Mô tả SEO |
| IsDeleted | bit | NOT NULL, DEFAULT(0) | Trạng thái xóa mềm |
| CreatedAt | datetime2 | NOT NULL, DEFAULT(GETUTCDATE()) | Ngày tạo |
| CreatedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người tạo |
| ModifiedAt | datetime2 |  | Ngày sửa đổi |
| ModifiedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người sửa đổi |
| DeletedAt | datetime2 |  | Ngày xóa |
| DeletedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người xóa |

##### 10. Brands

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | uniqueidentifier | PK | Định danh thương hiệu |
| Name | nvarchar(200) | NOT NULL, INDEX | Tên thương hiệu |
| Description | nvarchar(500) |  | Mô tả |
| LogoUrl | nvarchar(500) |  | URL logo |
| IsActive | bit | NOT NULL, DEFAULT(1) | Trạng thái kích hoạt |
| DisplayOrder | int | NOT NULL, DEFAULT(0) | Thứ tự hiển thị |
| Slug | nvarchar(200) | NOT NULL, INDEX, UNIQUE | URL thân thiện |
| IsDeleted | bit | NOT NULL, DEFAULT(0) | Trạng thái xóa mềm |
| CreatedAt | datetime2 | NOT NULL, DEFAULT(GETUTCDATE()) | Ngày tạo |
| CreatedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người tạo |
| ModifiedAt | datetime2 |  | Ngày sửa đổi |
| ModifiedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người sửa đổi |
| DeletedAt | datetime2 |  | Ngày xóa |
| DeletedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người xóa |

##### 11. Orders

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | uniqueidentifier | PK | Định danh đơn hàng |
| OrderNumber | nvarchar(50) | NOT NULL, INDEX, UNIQUE | Số đơn hàng |
| CustomerId | uniqueidentifier | FK (Customers.Id), NOT NULL, INDEX | ID khách hàng |
| OrderDate | datetime2 | NOT NULL, DEFAULT(GETUTCDATE()) | Ngày đặt hàng |
| OrderStatusId | int | FK (OrderStatuses.Id), NOT NULL, INDEX | Trạng thái |
| ShippingAddressId | uniqueidentifier | FK (Addresses.Id), NOT NULL | Địa chỉ giao hàng |
| BillingAddressId | uniqueidentifier | FK (Addresses.Id) | Địa chỉ thanh toán |
| PaymentMethodId | int | FK (PaymentMethods.Id) | Phương thức thanh toán |
| ShippingMethodId | int | FK (ShippingMethods.Id) | Phương thức vận chuyển |
| SubTotal | decimal(18,2) | NOT NULL | Tổng tiền hàng |
| ShippingCost | decimal(18,2) | NOT NULL, DEFAULT(0) | Phí vận chuyển |
| TaxAmount | decimal(18,2) | NOT NULL, DEFAULT(0) | Thuế |
| DiscountAmount | decimal(18,2) | NOT NULL, DEFAULT(0) | Giảm giá |
| TotalAmount | decimal(18,2) | NOT NULL | Tổng cộng |
| Notes | nvarchar(max) |  | Ghi chú của KH |
| AdminNotes | nvarchar(max) |  | Ghi chú của Admin |
| CouponCode | nvarchar(50) | INDEX | Mã giảm giá đã dùng |
| IsDeleted | bit | NOT NULL, DEFAULT(0) | Trạng thái xóa mềm |
| CreatedAt | datetime2 | NOT NULL, DEFAULT(GETUTCDATE()) | Ngày tạo |
| CreatedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người tạo (Nếu KH tạo) |
| ModifiedAt | datetime2 |  | Ngày sửa đổi |
| ModifiedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người sửa đổi |
| DeletedAt | datetime2 |  | Ngày xóa |
| DeletedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người xóa |

##### 12. OrderItems

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | uniqueidentifier | PK | Định danh mục đơn hàng |
| OrderId | uniqueidentifier | FK (Orders.Id), NOT NULL, INDEX | ID đơn hàng |
| ProductVariantId | uniqueidentifier | FK (ProductVariants.Id), NOT NULL, INDEX | ID biến thể sản phẩm |
| Quantity | int | NOT NULL | Số lượng |
| UnitPrice | decimal(18,2) | NOT NULL | Đơn giá tại thời điểm mua |
| TotalPrice | decimal(18,2) | NOT NULL | Thành tiền |
| ProductName | nvarchar(200) | NOT NULL | Tên SP (snapshot) |
| ProductVariantName | nvarchar(200) |  | Tên biến thể (snapshot) |
| Sku | nvarchar(100) |  | Mã SKU (snapshot) |
| CreatedAt | datetime2 | NOT NULL, DEFAULT(GETUTCDATE()) | Ngày tạo |

##### 13. OrderStatuses (Lookup Table)

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | int | PK | Định danh trạng thái |
| Name | nvarchar(50) | NOT NULL | Tên trạng thái |
| Description | nvarchar(200) |  | Mô tả |
| DisplayOrder | int | NOT NULL | Thứ tự hiển thị |
| IsDefault | bit | DEFAULT(0) | Trạng thái mặc định |

##### 14. PaymentMethods (Lookup Table)

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | int | PK | Định danh phương thức |
| Name | nvarchar(100) | NOT NULL | Tên phương thức |
| Description | nvarchar(500) |  | Mô tả |
| IsActive | bit | NOT NULL, DEFAULT(1) | Trạng thái kích hoạt |
| DisplayOrder | int | NOT NULL | Thứ tự hiển thị |
| AdditionalSettings | nvarchar(max) |  | Cấu hình bổ sung (JSON) |

##### 15. ShippingMethods (Lookup Table)

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | int | PK | Định danh phương thức |
| Name | nvarchar(100) | NOT NULL | Tên phương thức |
| Description | nvarchar(500) |  | Mô tả |
| Price | decimal(18,2) | NOT NULL | Giá |
| DeliveryTimeMin | int |  | Thời gian giao tối thiểu |
| DeliveryTimeMax | int |  | Thời gian giao tối đa |
| IsActive | bit | NOT NULL, DEFAULT(1) | Trạng thái kích hoạt |
| DisplayOrder | int | NOT NULL | Thứ tự hiển thị |

#### Customer Management Tables

##### 16. Customers

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | uniqueidentifier | PK | Định danh khách hàng |
| UserId | nvarchar(450) | FK (AspNetUsers.Id), INDEX, UNIQUE | ID người dùng liên kết |
| FirstName | nvarchar(100) | NOT NULL | Tên |
| LastName | nvarchar(100) | NOT NULL | Họ |
| Email | nvarchar(256) | NOT NULL, INDEX, UNIQUE | Email (có thể khác AspNetUser) |
| PhoneNumber | nvarchar(20) | INDEX | Số điện thoại |
| DateOfBirth | date |  | Ngày sinh |
| Gender | nvarchar(10) |  | Giới tính |
| CustomerTypeId | int | FK (CustomerTypes.Id) | Loại khách hàng |
| IsSubscribedToNewsletter | bit | NOT NULL, DEFAULT(0) | Đăng ký newsletter |
| IsDeleted | bit | NOT NULL, DEFAULT(0) | Trạng thái xóa mềm |
| CreatedAt | datetime2 | NOT NULL, DEFAULT(GETUTCDATE()) | Ngày tạo |
| CreatedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người tạo |
| ModifiedAt | datetime2 |  | Ngày sửa đổi |
| ModifiedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người sửa đổi |
| DeletedAt | datetime2 |  | Ngày xóa |
| DeletedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người xóa |

##### 17. Addresses

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | uniqueidentifier | PK | Định danh địa chỉ |
| CustomerId | uniqueidentifier | FK (Customers.Id), NOT NULL, INDEX | ID khách hàng |
| FirstName | nvarchar(100) | NOT NULL | Tên người nhận |
| LastName | nvarchar(100) | NOT NULL | Họ người nhận |
| AddressLine1 | nvarchar(200) | NOT NULL | Địa chỉ dòng 1 |
| AddressLine2 | nvarchar(200) |  | Địa chỉ dòng 2 |
| City | nvarchar(100) | NOT NULL | Thành phố |
| State | nvarchar(100) |  | Tỉnh/Bang |
| PostalCode | nvarchar(20) |  | Mã bưu điện |
| CountryId | int | FK (Countries.Id), NOT NULL | ID quốc gia |
| PhoneNumber | nvarchar(20) |  | Số điện thoại người nhận |
| IsDefaultShipping | bit | NOT NULL, DEFAULT(0) | Địa chỉ giao hàng mặc định |
| IsDefaultBilling | bit | NOT NULL, DEFAULT(0) | Địa chỉ thanh toán mặc định |
| CreatedAt | datetime2 | NOT NULL, DEFAULT(GETUTCDATE()) | Ngày tạo |
| CreatedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người tạo |
| ModifiedAt | datetime2 |  | Ngày sửa đổi |
| ModifiedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người sửa đổi |

##### 18. Countries (Lookup Table)

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | int | PK | Định danh quốc gia |
| Name | nvarchar(100) | NOT NULL | Tên quốc gia |
| IsoCode | nvarchar(2) | NOT NULL, UNIQUE | Mã ISO2 |
| IsoCode3 | nvarchar(3) | UNIQUE | Mã ISO3 |
| IsActive | bit | NOT NULL, DEFAULT(1) | Trạng thái kích hoạt |
| DisplayOrder | int | NOT NULL | Thứ tự hiển thị |

##### 19. CustomerTypes (Lookup Table)

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | int | PK | Định danh loại khách hàng |
| Name | nvarchar(100) | NOT NULL | Tên loại |
| Description | nvarchar(500) |  | Mô tả |
| DiscountPercentage | decimal(5,2) |  | % giảm giá mặc định |

#### Inventory & Stock Tables (Đơn giản hóa - Tồn kho theo ProductVariant)

##### 20. InventoryHistory (Theo dõi thay đổi tồn kho)

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | uniqueidentifier | PK | Định danh lịch sử |
| ProductVariantId | uniqueidentifier | FK (ProductVariants.Id), NOT NULL, INDEX | ID biến thể |
| WarehouseId | uniqueidentifier | FK (Warehouses.Id), NOT NULL, INDEX | ID kho hàng |
| QuantityChange | int | NOT NULL | Thay đổi số lượng (+/-) |
| QuantityBefore | int | NOT NULL | Số lượng trước |
| QuantityAfter | int | NOT NULL | Số lượng sau |
| Note | nvarchar(500) |  | Ghi chú |
| ReferenceType | nvarchar(50) | INDEX | Loại tham chiếu (Order, Stock Adjustment…) |
| ReferenceId | nvarchar(128) | INDEX | ID tham chiếu |
| CreatedAt | datetime2 | NOT NULL, DEFAULT(GETUTCDATE()) | Ngày tạo |
| CreatedBy | nvarchar(450) | FK (AspNetUsers.Id), NOT NULL | Người tạo |

##### 21. Warehouses

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | uniqueidentifier | PK | Định danh kho hàng |
| Name | nvarchar(200) | NOT NULL, INDEX | Tên kho hàng |
| AddressLine1 | nvarchar(200) | NOT NULL | Địa chỉ dòng 1 |
| AddressLine2 | nvarchar(200) |  | Địa chỉ dòng 2 |
| City | nvarchar(100) | NOT NULL | Thành phố |
| State | nvarchar(100) |  | Tỉnh/Bang |
| PostalCode | nvarchar(20) |  | Mã bưu điện |
| CountryId | int | FK (Countries.Id), NOT NULL | ID quốc gia |
| PhoneNumber | nvarchar(20) |  | Số điện thoại |
| Email | nvarchar(256) |  | Email |
| IsActive | bit | NOT NULL, DEFAULT(1) | Trạng thái kích hoạt |
| IsDeleted | bit | NOT NULL, DEFAULT(0) | Trạng thái xóa mềm |
| CreatedAt | datetime2 | NOT NULL, DEFAULT(GETUTCDATE()) | Ngày tạo |
| CreatedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người tạo |
| ModifiedAt | datetime2 |  | Ngày sửa đổi |
| ModifiedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người sửa đổi |
| DeletedAt | datetime2 |  | Ngày xóa |
| DeletedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người xóa |

#### Review & Rating Tables

##### 22. ProductReviews

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | uniqueidentifier | PK | Định danh đánh giá |
| ProductId | uniqueidentifier | FK (Products.Id), NOT NULL, INDEX | ID sản phẩm |
| CustomerId | uniqueidentifier | FK (Customers.Id), NOT NULL, INDEX | ID khách hàng |
| OrderItemId | uniqueidentifier | FK (OrderItems.Id), INDEX | ID mục đơn hàng |
| Rating | int | NOT NULL, CHECK(Rating BETWEEN 1 AND 5) | Điểm (1-5) |
| ReviewText | nvarchar(max) |  | Nội dung đánh giá |
| Title | nvarchar(200) |  | Tiêu đề đánh giá |
| IsApproved | bit | NOT NULL, DEFAULT(0) | Đã duyệt |
| IsVerifiedPurchase | bit | NOT NULL, DEFAULT(0) | Đã xác minh mua hàng |
| CreatedAt | datetime2 | NOT NULL, DEFAULT(GETUTCDATE()) | Ngày tạo |
| CreatedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người tạo (nếu có) |
| ModifiedAt | datetime2 |  | Ngày sửa đổi |
| ModifiedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người sửa đổi |

##### 23. ReviewHelpfulness

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| ReviewId | uniqueidentifier | PK, FK (ProductReviews.Id), NOT NULL | ID đánh giá |
| CustomerId | uniqueidentifier | PK, FK (Customers.Id), NOT NULL | ID khách hàng |
| IsHelpful | bit | NOT NULL | Có hữu ích |
| CreatedAt | datetime2 | NOT NULL, DEFAULT(GETUTCDATE()) | Ngày tạo |

#### Marketing Tables

##### 24. Discounts

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | uniqueidentifier | PK | Định danh giảm giá |
| Name | nvarchar(200) | NOT NULL | Tên chương trình |
| Description | nvarchar(500) |  | Mô tả |
| DiscountType | int | NOT NULL, INDEX | Loại (Percentage, FixedAmount…) |
| DiscountValue | decimal(18,2) | NOT NULL | Giá trị giảm giá (% hoặc số tiền) |
| MinimumOrderAmount | decimal(18,2) |  | Giá trị đơn hàng tối thiểu |
| MaximumDiscountAmount | decimal(18,2) |  | Giảm giá tối đa (cho loại %) |
| StartDate | datetime2 | NOT NULL | Ngày bắt đầu |
| EndDate | datetime2 |  | Ngày kết thúc |
| CouponCode | nvarchar(50) | INDEX, UNIQUE | Mã giảm giá (nếu có) |
| UsesPerCustomer | int |  | Số lần sử dụng/KH |
| UsesPerCoupon | int |  | Tổng số lần sử dụng |
| IsActive | bit | NOT NULL, DEFAULT(1) | Trạng thái kích hoạt |
| CreatedAt | datetime2 | NOT NULL, DEFAULT(GETUTCDATE()) | Ngày tạo |
| CreatedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người tạo |
| ModifiedAt | datetime2 |  | Ngày sửa đổi |
| ModifiedBy | nvarchar(450) | FK (AspNetUsers.Id) | Người sửa đổi |

##### 25. DiscountUsageHistory

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | uniqueidentifier | PK | Định danh sử dụng |
| DiscountId | uniqueidentifier | FK (Discounts.Id), NOT NULL, INDEX | ID giảm giá |
| OrderId | uniqueidentifier | FK (Orders.Id), NOT NULL, INDEX | ID đơn hàng |
| CustomerId | uniqueidentifier | FK (Customers.Id), NOT NULL, INDEX | ID khách hàng |
| CreatedAt | datetime2 | NOT NULL, DEFAULT(GETUTCDATE()) | Ngày sử dụng |

#### Logging & Audit Tables

##### 26. AuditLogs (Theo dõi thay đổi dữ liệu)

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | uniqueidentifier | PK | Định danh log |
| UserId | nvarchar(450) | FK (AspNetUsers.Id), INDEX | ID người dùng thực hiện |
| EntityName | nvarchar(100) | NOT NULL, INDEX | Tên entity |
| EntityId | nvarchar(128) | NOT NULL, INDEX | ID entity |
| Action | nvarchar(50) | NOT NULL, INDEX | Hành động (Create, Update, Delete) |
| OldValues | nvarchar(max) |  | Giá trị cũ (JSON) |
| NewValues | nvarchar(max) |  | Giá trị mới (JSON) |
| ClientIp | nvarchar(50) |  | Địa chỉ IP client |
| CreatedAt | datetime2 | NOT NULL, DEFAULT(GETUTCDATE()) | Thời gian |

##### 27. ActivityLogs (Theo dõi hoạt động người dùng/hệ thống)

| Column Name | Data Type | Constraints | Description |
| --- | --- | --- | --- |
| Id | uniqueidentifier | PK | Định danh log |
| UserId | nvarchar(450) | FK (AspNetUsers.Id), INDEX | ID người dùng |
| ActivityType | nvarchar(100) | NOT NULL, INDEX | Loại hoạt động (Login, View Product…) |
| Description | nvarchar(500) |  | Mô tả chi tiết |
| EntityName | nvarchar(100) | INDEX | Tên entity liên quan |
| EntityId | nvarchar(128) | INDEX | ID entity liên quan |
| IpAddress | nvarchar(50) |  | Địa chỉ IP |
| UserAgent | nvarchar(500) |  | Trình duyệt/thiết bị |
| CreatedAt | datetime2 | NOT NULL, DEFAULT(GETUTCDATE()) | Thời gian |

## 9. PHÊ DUYỆT

| Vai trò | Tên | Chữ ký | Ngày |
| --- | --- | --- | --- |
| Người yêu cầu | phamtiendungcw | ✓ | 2023-02-10 |
| Quản lý dự án | phamtiendungcw | ✓ | 2023-02-10 |
| Developer | phamtiendungcw | ✓ | 2023-02-10 |