BÁO CÁO KIẾN TRÚC PHẦN MỀM

Đề tài: Hệ thống Web Bán Sách (Book Store Web System)

Môn học: Software Architecture and Design

1. Giới thiệu chung

Kiến trúc phần mềm là cách tổ chức tổng thể của một hệ thống, xác định các thành phần chính và mối quan hệ giữa chúng. Một kiến trúc tốt giúp hệ thống dễ phát triển, dễ bảo trì và có khả năng mở rộng trong tương lai.

Trong báo cáo này, sinh viên nghiên cứu và trình bày **ba phong cách kiến trúc phần mềm phổ biến**:

- Monolithic Architecture
- Clean Architecture
- Microservices Architecture

Các kiến trúc này được áp dụng vào **hệ thống Web Bán Sách** sử dụng **Django Framework** và **MySQL Database**.

2. Monolithic Architecture

2.1 Khái niệm

Monolithic Architecture là kiến trúc truyền thống, trong đó toàn bộ hệ thống được xây dựng như **một khối thống nhất**. Tất cả các chức năng (giao diện, xử lý nghiệp vụ, truy cập dữ liệu) đều nằm trong cùng một dự án và được triển khai cùng nhau.

Trong hệ thống Web Bán Sách, kiến trúc Monolithic gồm:

- Một Django project duy nhất
- Các app: accounts, books, cart
- Sử dụng chung một cơ sở dữ liệu MySQL

2.2 Đặc điểm

- Một codebase duy nhất
- Các module phụ thuộc chặt chẽ vào nhau
- Chia theo app nhưng vẫn nằm trong cùng project
- Dễ triển khai trên một server

2.3 Ưu điểm

- Dễ thiết kế và phát triển
- Phù hợp với dự án nhỏ và vừa
- Dễ debug và test
- Thời gian phát triển nhanh

2.4 Nhược điểm

- Khó mở rộng khi hệ thống lớn
- Thay đổi một chức năng có thể ảnh hưởng toàn hệ thống
- Khó bảo trì lâu dài
- Không linh hoạt khi scale từng phần riêng lẻ

3. Clean Architecture

3.1 Khái niệm

Clean Architecture là kiến trúc tập trung vào **tách biệt logic nghiệp vụ khỏi framework và hạ tầng**. Mục tiêu chính là đảm bảo **logic cốt lõi không phụ thuộc vào Django, database hay UI**.

3.2 Các tầng trong Clean Architecture

1. Domain Layer

- Chứa các Entity (Customer, Book, Cart, CartItem)
- Chỉ chứa logic nghiệp vụ cốt lõi
- Không phụ thuộc framework

2. Use Cases Layer

- Chứa các nghiệp vụ như:
    - RegisterCustomer
    - LoginCustomer
    - AddToCart
- Điều phối luồng xử lý nghiệp vụ

3. Interfaces Layer

- Định nghĩa interface (Repository, Service)

- Kết nối UseCase với hạ tầng

4. Infrastructure Layer

- Cài đặt cụ thể của repository
- Giao tiếp với MySQL, ORM

5. Framework Layer

- Django
- Views, serializers, URLs

3.3 Nguyên tắc Dependency Rule

Phụ thuộc chỉ được hướng **từ ngoài vào trong**

Framework → Infrastructure → Interfaces → UseCases → Domain

3.4 Ưu điểm

- Logic nghiệp vụ độc lập với framework
- Dễ test (unit test không cần Django)
- Dễ mở rộng và bảo trì
- Code rõ ràng, có cấu trúc

3.5 Nhược điểm

- Thiết kế ban đầu phức tạp
- Nhiều file và layer
- Không phù hợp với dự án nhỏ, gấp deadline

4. Microservices Architecture

4.1 Khái niệm

Microservices Architecture chia hệ thống thành **nhiều service độc lập**, mỗi service đảm nhận một chức năng riêng và giao tiếp với nhau thông qua **REST API**.

4.2 Các service trong hệ thống Book Store

1. Customer Service

- Đăng ký, đăng nhập
- Quản lý khách hàng

- Database riêng

2. Book Service

- Quản lý sách
- Xem catalog
- Kiểm tra tồn kho

3. Cart Service

- Thêm sách vào giỏ hàng
- Xem giỏ hàng
- Gọi API từ Book Service

4.3 Đặc điểm

- Mỗi service là một Django project riêng
- Mỗi service có database riêng
- Giao tiếp qua HTTP (REST API)
- Có thể triển khai độc lập

4.4 Ưu điểm

- Dễ mở rộng theo từng service
- Triển khai độc lập
- Phù hợp hệ thống lớn
- Dễ tích hợp CI/CD

4.5 Nhược điểm

- Phức tạp khi triển khai
- Khó debug do nhiều service
- Tốn tài nguyên
- Cần quản lý API, network

5. So sánh ba kiến trúc

Tiêu chí	Monolithic	Clean Architecture	Microservices
Độ phức tạp	Thấp	Trung bình	Cao
Mở rộng	Kém	Tốt	Rất tốt
Bảo trì	Khó	Dễ	Trung bình
Phù hợp	Dự án nhỏ	Trung bình	Dự án lớn

6. Kết luận

Ba kiến trúc phần mềm đều có ưu và nhược điểm riêng:

- **Monolithic** phù hợp cho hệ thống nhỏ, học tập
- **Clean Architecture** phù hợp để học tư duy kiến trúc chuẩn
- **Microservices** phù hợp cho hệ thống lớn, thực tế doanh nghiệp