A black background with blue and purple text

Description automatically generated

**BÁO CÁO ĐỒ ÁN**

**HỌC PHẦN: CÔNG NGHỆ LẬP TRÌNH HIỆN ĐẠI**

**Giảng viên hướng dẫn: Đỗ Như Tài**

Nhóm: 3

Tên thành viên:

+ Huỳnh Anh Tuấn MSSV: 3121411220

+ Trần Gia Phú MSSV: 3121411166

+ Phùng Tấn Nhựt MSSV: 3121411159

**Năm học: 2024 - 2025**

**BÁO CÁO BÀI TẬP 3 - KIẾN TRÚC CLEAN TRONG ỨNG DỤNG ASP.NET CORE**

### ****1. Giải thích kiến trúc Clean trong ứng dụng ASP.NET Core****

**Clean Architecture** là mô hình kiến trúc phần mềm do Robert C. Martin (Uncle Bob) đề xuất, nhằm tách biệt rõ ràng giữa các phần logic nghiệp vụ và các yếu tố hạ tầng (như UI, cơ sở dữ liệu). Mục tiêu chính là **hạn chế sự phụ thuộc ngược**, tức là **các lớp bên ngoài phụ thuộc vào lớp bên trong** chứ không ngược lại, giúp ứng dụng dễ bảo trì, mở rộng, và kiểm thử.

Trong ASP.NET Core, Clean Architecture thường bao gồm 4 tầng chính:

**Domain Layer**: Chứa các entity và logic nghiệp vụ cốt lõi.

**Application Layer**: Chứa các use case, định nghĩa luồng nghiệp vụ cụ thể.

**Infrastructure Layer**: Chứa logic tương tác với hệ thống bên ngoài như database, API, email, v.v.

**Presentation Layer (Web)**: Cung cấp giao diện người dùng hoặc Web API, nhận yêu cầu và chuyển đến Application Layer.

### ****2. Mô tả các tầng chính****

#### ****a. Domain Layer****

**Thư mục:** Domain/

**Thành phần chính:** Entities, Interfaces, ValueObjects

**Mục đích:**

Định nghĩa các **Entity** cốt lõi đại diện cho đối tượng trong hệ thống (Product, Order, User, ...).

Không phụ thuộc vào bất kỳ framework hay thư viện nào.

Được tái sử dụng dễ dàng ở mọi tầng khác.

#### ****b. Application Layer****

**Thư mục:** Application/

**Thành phần chính:** UseCases, DTOs, Interfaces, Service

**Mục đích:**

Định nghĩa các **Use Case** (ví dụ: CreateOrder, GetProducts).

Sử dụng **interface** để giao tiếp với hạ tầng (cơ sở dữ liệu, email, file, ...).

Xử lý logic nghiệp vụ cụ thể, điều phối giữa Domain và Infrastructure.

#### ****c. Infrastructure Layer****

**Thư mục:** Infrastructure/

**Thành phần chính:** Repository Implementations, EF Core, External Services

**Mục đích:**

Cung cấp triển khai cụ thể cho các interface đã khai báo trong Application Layer.

Tương tác với **database**, hệ thống file, API bên ngoài.

Phụ thuộc vào tầng Application (theo nguyên lý Dependency Inversion).

#### ****d. Presentation Layer (Web Layer)****

**Thư mục:** WebUI/, API/

**Tệp chính:** Program.cs, Startup.cs (hoặc builder trong .NET 6+)

**Mục đích:**

Tiếp nhận các request từ client (trình duyệt, Postman,...).

Gọi các Use Case từ Application Layer.

Trả về response thích hợp (HTML/JSON).

Cấu hình DI, middleware, và route.

### ****3. Hướng dẫn build và chạy dự án****

#### ****Bước 1: Clone repository****

bash

CopyEdit

git clone https://github.com/aspnetrun/run-aspnetcore-basics.gitcd run-aspnetcore-basics

#### ****Bước 2: Build và run ứng dụng****

bash

CopyEdit

dotnet restore

dotnet build

dotnet run --project AspnetRun.Web

#### ****Bước 3: Truy cập giao diện ứng dụng****

Mở trình duyệt và vào địa chỉ:  
👉 [http://localhost:5000/](http://localhost:5000/" \t "_new)

#### ****Kiểm tra Web API****

Truy cập:  
👉 [http://localhost:5000/api/products](http://localhost:5000/api/products" \t "_new)  
Kết quả trả về sẽ là danh sách sản phẩm ở định dạng JSON.

### ****4. Hướng dẫn triển khai với Docker****

#### ****Bước 1: Cài đặt Docker****

Tải tại: [https://www.docker.com/](https://www.docker.com/" \t "_new)

Kiểm tra cài đặt:

bash

CopyEdit

docker --version

#### ****Bước 2: Viết Dockerfile****

Tạo file Dockerfile trong thư mục gốc với nội dung:

Dockerfile

CopyEdit

FROM mcr.microsoft.com/dotnet/core/aspnet:3.1-nanoserver-1709 AS base

WORKDIR /app

EXPOSE 80

EXPOSE 443

FROM mcr.microsoft.com/dotnet/core/sdk:3.1-nanoserver-1709 AS build

WORKDIR /src

COPY ["AspnetRunBasics/AspnetRunBasics.csproj", "AspnetRunBasics/"]

RUN dotnet restore "AspnetRunBasics/AspnetRunBasics.csproj"

COPY . .

WORKDIR "/src/AspnetRunBasics"

RUN dotnet build "AspnetRunBasics.csproj" -c Release -o /app/build

FROM build AS publish

RUN dotnet publish "AspnetRunBasics.csproj" -c Release -o /app/publish

FROM base AS final

WORKDIR /app

COPY --from=publish /app/publish .

ENTRYPOINT ["dotnet", "AspnetRunBasics.dll"]

#### ****Bước 3: Build và chạy container****

bash

CopyEdit

docker build -t myapp .

docker run -d -p 5000:80 myapp

### ****5. Kết luận****

**Clean Architecture** giúp tổ chức ứng dụng ASP.NET Core một cách rõ ràng, **tách biệt giữa business logic và các thành phần hạ tầng**, tăng tính mô-đun, dễ kiểm thử, dễ bảo trì. Các tầng trong Clean Architecture hoạt động độc lập và chỉ phụ thuộc vào chiều từ ngoài vào trong, giúp bạn dễ dàng thay thế hoặc mở rộng hệ thống trong tương lai.