Cài đặt các package sau:

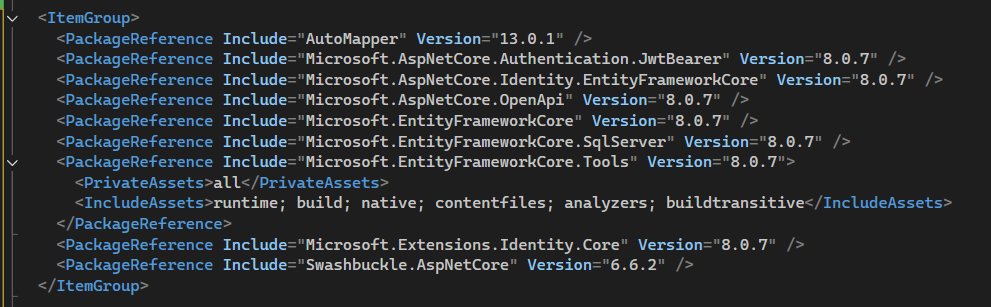
Automapper :

* cái này tương tự DTO, dùng để mapping 2 Object (đối tượng) với nhau. Thay vì tạo lớp DTO thì dùng package này đỡ làm thủ công mất thời gian và việc duplicate sẽ xảy ra nếu nhiều lần cần chuyển đổi dữ liệu

Microsoft.Extensions.Identity.Core:

* Cái này để tạo nhanh model đăng ký đăng nhập

Và các package linh tinh khác:



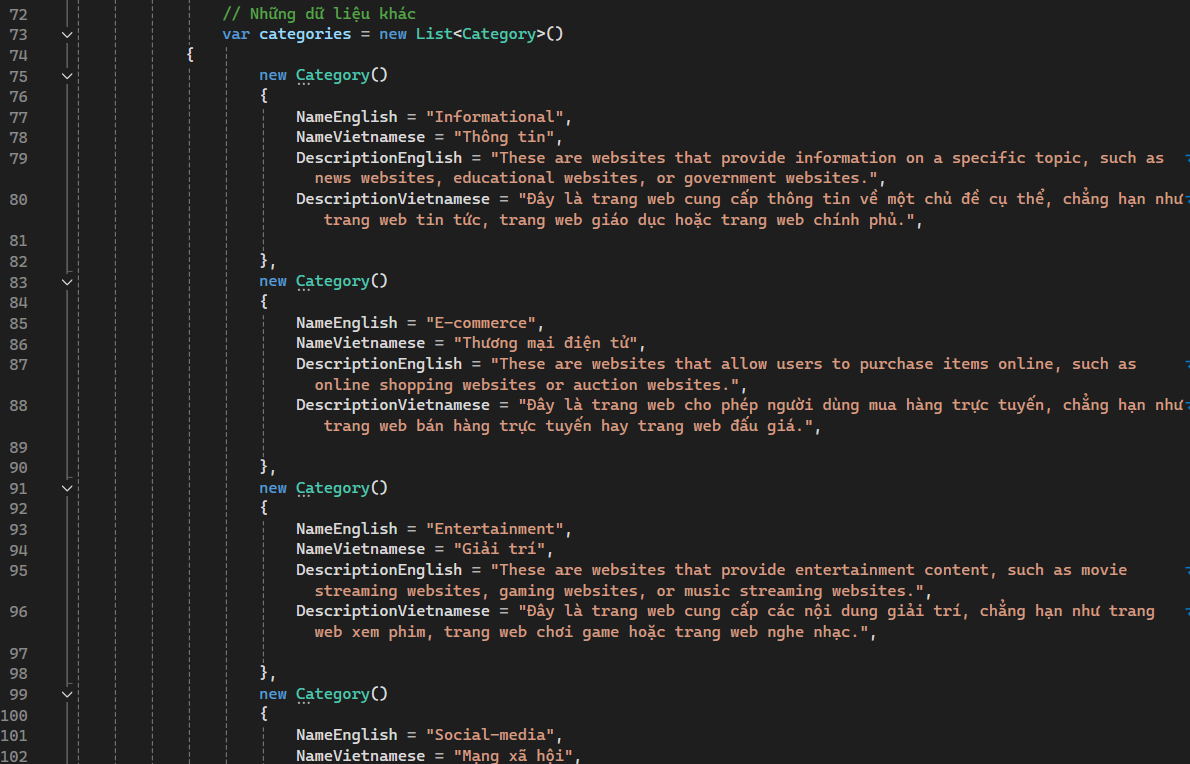
Chia clean architect như MVC thôi

Đầu tiên Persistence: đây là tầng chứa Data những gì làm việc liên quan tới data thì ném vào.

* Persistence layer: chịu trách nhiệm giao tiếp với data layer (thường là một hệ quản trị cơ sở dữ liệu quan hệ – Relational DBMS)

Trong đó có chứa 3 mục sau:

* Data nơi chứa DbContext
* DbInitializer nơi chứa các dataSeed ( dataSeed có thể hiểu là dữ liệu mẫu khi run PJ thì luôn có sẵn)



(ví dụ minh họa)

* Entity nơi chứa các thực thể (class),
  + phần BASE bên trong là tạo 1 class để lặp đi lặp lại những thuộc tính trùng lặp có ở nhiều bảng (như là ngày tạo, ngày kết thúc, tạo bởi ai và cập nhật bởi ai)

các tầng clean architect

Web api (Presentation) : hiển thị giao hiện -> OnlineFood

UseCase (Application) : xử lý business logic của tầng application \*\*

Domain: chứa các Entity

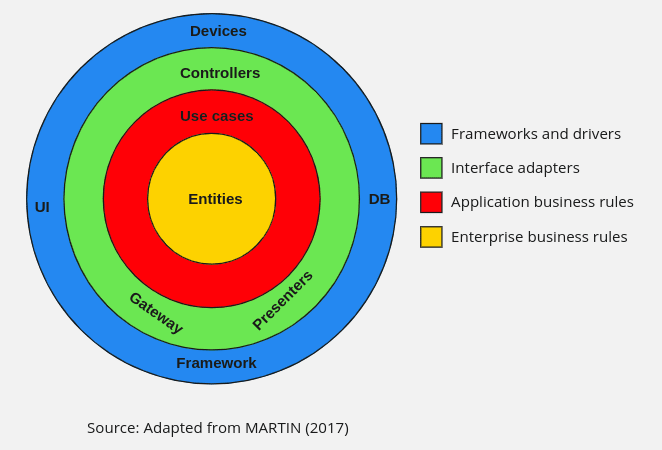
Infrastructure (cơ sở hạ tầng): chứa dataAccess hoặc Persistence (có thể hiểu là repository). Sau này nơi này sẽ chứa thêm các third party như Redis, Kafka, RabbitMQ,…

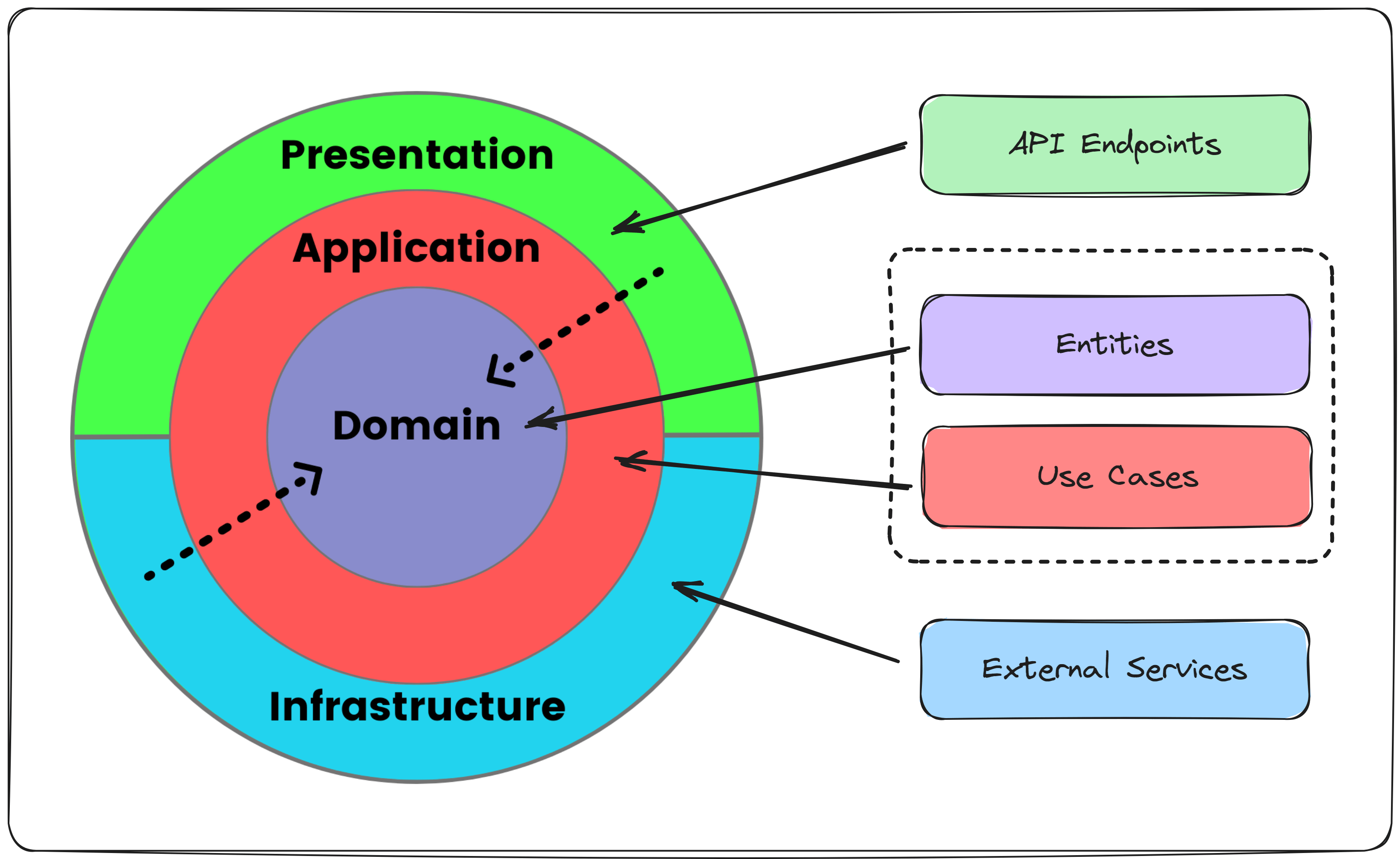
**Cách refference:**

UseCase ~> domain

Infrastructure ~> UseCase

* WebAPI (UI) ~> Infrastructure





* Thực tế Presentation (UI) và Infrastructure là ngang cấp, nhưng bởi vì UI có chứa Program, còn **Infrastructure chứa DataAccess(Repository) và Persistence.**
* Mà Application của mình cần persistence nên reference vào Persistence và Use case
* Còn Riêng Repository thì cần Presentation (UI) nên sẽ reference tới OnlineFood

!!! ở đây có 1 chút khác biệt vì mình tách ra cụ thể hơn là **Infrastructure chứa DataAccess(Repository) và Persistence.**

