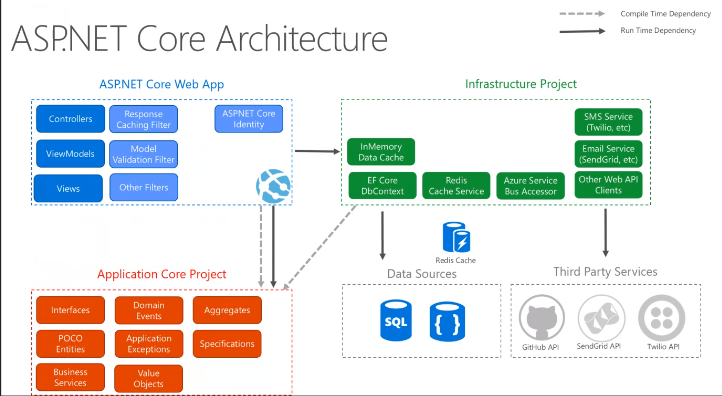
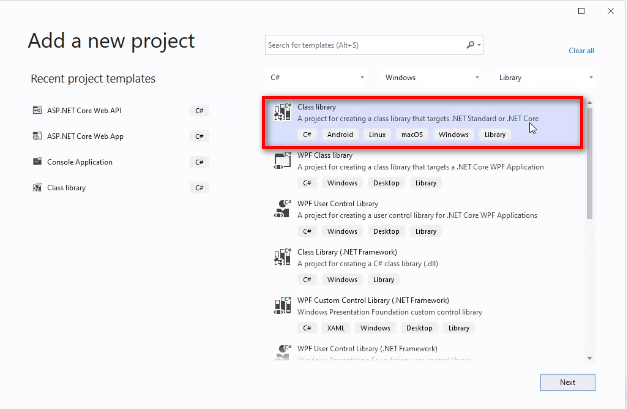
Tìm hiểu thêm về **Microservices**

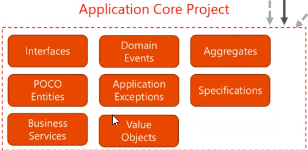


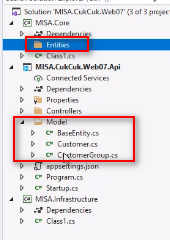
Chia **thành 3 project** khác nhau để tương tác và đảm nhiệm những nhiệm vụ khác nhau.

1. **Tạo 2 project** với dạng **Library:** MISA.Core, MISA.Infrastructure và reference 2 project vào project chính.



**MISA.Core** sẽ chứa các thành phần dùng chung



1. **Dựng các Entities** của MISA.Core 

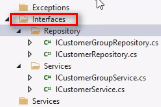
**Chi tiết:** Chuyển các Entities từ Model sang hoặc tạo mới hoàn toàn ở File Entities trong MISA.Core

1. **Dựng các Interface**

**Chi tiết:** chứa các công việc mà có thể làm được (Chỉ là dạng liệt kê chứ không phải đối tượng để thực thi)

**Chú ý:**

* 1. 2 class không bao giờ giao tiếp trực tiếp với nhau mà mọi việc phải giao tiếp qua Interface
  2. Interface chuẩn convention phải bắt đầu bằng chữ **“I”**



+ **Repository:** Các vấn đề cơ bản (Thêm, sửa, xoá, v.v)

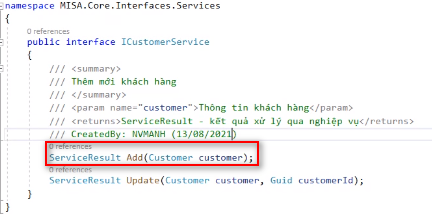
+ **Services:** Xử lý các vấn đề về nghiệp vụ (Validate thông tin, xử lý chuỗi, v.v)

1. **Tạo một class ServiceResult** chứa thông tin trả về

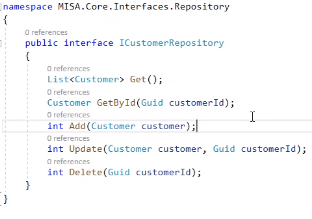


1. **Thêm ServiceResult** vào các service cần thiết





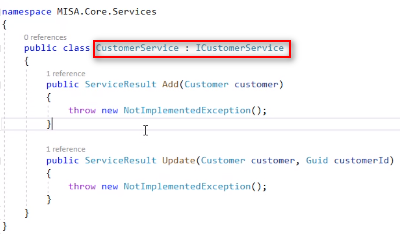
1. **Dựng CustomerRepository**



1. **Tạo class CustomerService trong service**

**Chi tiết:** Để xử lý nghiệp vụ cho Interface **ICustomerService**

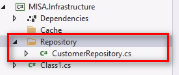




Có thể tạo nhiều class để xử lý công việc của một interface



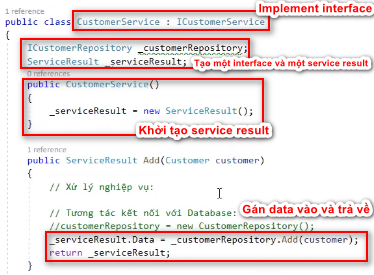
1. **Tạo các chức năng của MISA.Infrastructure**



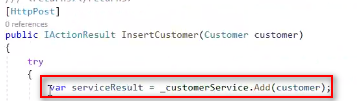
1. **Tạo class CustomerRepository** trong thư mục Repository của MISA.Infrastructure

**Chi tiết:** để xử lý nghiệp vụ cho các Interface Repository

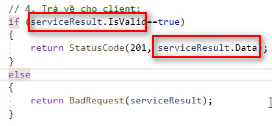
1. **Tạo Interface chung để dùng cho các service**



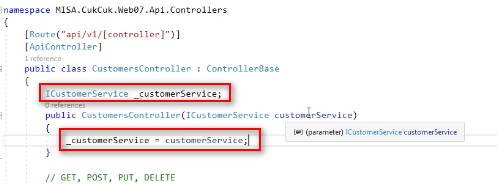
1. **Sử dụng Service trong các method của API Controller**



Sau khi service chạy xong thì trả về giá trị là **serviceResult**

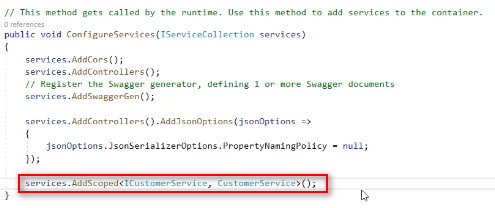


1. **Cho Interface chỉ định ra một object để làm việc (Dependency injection)**

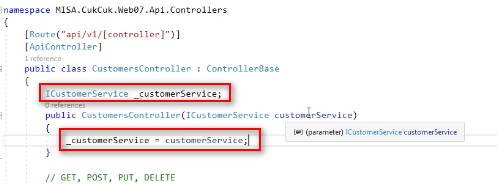


**Chú ý:** Tạo một interface và trong hàm khởi tạo controller tạo đối tượng interface từ dependency

1. **Cấu hình DI (Dependency injection) trong Startup.cs**

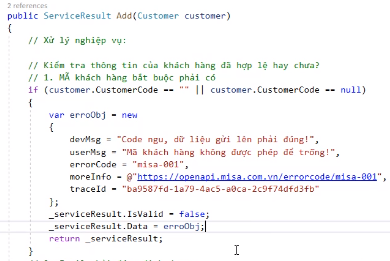


1. **Tương tự tiêm cho Service dependency đến Repository ở MISA.Infrastructure**

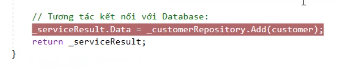




1. **Hoàn thiện các logic code trong class**

**+ Trong service xử lý các nghiệp vụ** 

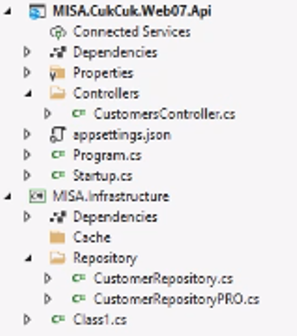
**+ và gọi đến repository của Infrastructure**



**Cốt lõi kiến trúc ứng dụng**

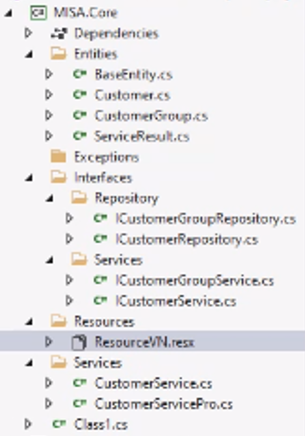
Chia thành 3 project cũng chính là những thành phần chính của ứng dụng:

1. **Project chính: .NET Core**



Chứa các controller api chính và các class config đường dẫn, v.v

1. **Project kiểu library:**



**Core:**

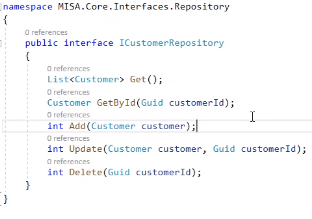
Là project chứa các thư mục để xử lý chung và tạo đối tượng như:

**+ Entities:** Khởi tạo các đối tượng với các thuộc tính cần thiết (tương tự file model trong project thông thường). Đặc biệt là có lớp ServiceResult để tạo ra một kết quả trả về chuẩn cho một Service (giải thích bên dưới)

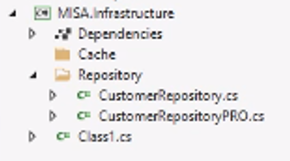
**+ Interface:** Tiếp đến là thư mục interface để tạo ra các interface liệt kê ra các phương thức có thể có của các đối tượng. Trong interface sẽ được chia ra thành Repository và Service

* **Repository:** là các interface chứa các phương thức cơ bản như thêm, xoá, sửa, v.v
* **Service:** là các interface chứa các phương thức nghiệp vụ như (xác thực thông tin, v.v)

+ **Service:** thư mục service chứa những class để thực hiện những nghiệp vụ tương ứng với interface , gọi đến Repository của Project thứ 3 là **Infrastructure** chứ service không làm công việc của repository



1. Project kiểu library:



**Infrastructure:**

Chứa các nghiệp vụ Repository cơ bản (còn các chức năng khác chưa học)

Sẽ được service gọi đến để thực hiện

**Directory Overview**

