

**Sulution Item:**

* Thư mục ảo để lưu các file cần thiết

+ docker-compase.yml

+ docker-compose.override.yml

+ global.json: Xác định version (Khi build sẽ chạy dự án với Core được cấu hình trong file này)

**Src:**

**ApiGetways:**

* OcelotApiGw:

**BuildingBlocks:**

* Các phần bao nhất của toàn bộ dự án, Library

+ Common.Logging:

+ Contracts: Chứa các định nghĩa của dự án, toàn bộ interface. Các service phải triển khai đúng với định nghĩa trong Contracts

+ EventBus.Message: Định nghĩa các event, message, interface

+ Infrastructure: Refrence tới Contracts, sẽ triển khai 1 số phương thức mà Contracts đưa ra

Ex: triển khai interface cho reponsitory => Các service sẽ Refrence tới Infrastructure

+ Shared: Chứa các Dto, cấu hình chung, địa chỉ, url liên lạc với các service khác

**Transaction:**

* Thường có 2 cách giải quyết:

+ Cách 1: Log dữ liệu của 1 đối tượng, 1 tác vụ (1 api ) ở 1 service lại => Qua service đợi service này xử lý thành công thì service kia sẽ tiếp tục chạy.

+ Cách 2: Cho chạy thành công, nếu 1 trong các service xảy ra lỗi gọi event 1 rabbitmq/kafka thông báo tác vụ liên quan đến transaction (đơn hàng, hóa đơn) này bị lỗi. Và xóa hết transaction đó

**Store string values as json strong SQL Server:**

* Lưu tất cả các dòng dữ liệu thành 1 dòng dạng string

+ Ưu: load nhanh, chỉ lưu 1 dòng, hiển thị, thống kê, báo cáo tất cả cùng 1 lúc nhanh cho FE

+ Nhược: Update phải load tất cả ra và update 1 cái

**Order:**

**https://github.com/jasontaylordev/CleanArchitecture**