

**ĐỀ KIỂM TRA KẾT THÚC HỌC PHẦN**  
**MÔN: PHÁT TRIỂN ỨNG DỤNG PHÍA SERVER**

**Đề 1**

Xây dựng ứng dụng ASP .NET Core WEB API có tên <Họ và tên><Mã số sinh viên> (vd: **NguyenVanA1234**) thỏa mãn: (thời gian làm bài 135 phút)

**1. Yêu cầu chung:**

**a. Tổ chức code**

- Coding convention sử dụng PascalCase với tên các trường, class, interface. Sử dụng camelCase với các biến. Sử dụng \_camelCase với các trường có access modifier là private.
- Cây thư mục code dạng như sau:
  - Properties: các config chạy ứng dụng
  - Constants: các hằng số
  - Controllers: các API Controller
  - DbContexts: các context entity framework
  - Dtos: chia các thư mục con theo nghiệp vụ
  - Entities: các entity map với bảng trong cơ sở dữ liệu
  - Exceptions: các class xử lý ngoại lệ
  - Migrations: các migration tạo các cấu trúc bảng quan hệ
  - Services: xử lý các logic nghiệp vụ
    - Implements: triển khai các interface
    - Interfaces: khai báo các interface
  - Utils: các hàm hữu ích dùng chung
  - appsetting.json: các setting của ứng dụng dạng json
  - Program.cs

**b. Yêu cầu từng thành phần**

- Các class Dto phải xử lý **trim** đối với các trường kiểu dữ liệu **string**.
- Validate model bằng các built-in annotation attribute với các class Dto phục vụ cho việc create, update, delete (bao gồm bắt buộc nhập, số lượng ký tự với trường kiểu chuỗi, giá trị tối thiểu tối đa với trường kiểu số, ...).
- Kiểu dữ liệu trong các entity sử dụng phù hợp với bài toán.
- Các API Controller trả về kiểu dữ liệu phù hợp với interface IActionResult.
- Các API Controller kế thừa từ class ApiControllerBase (tự tạo) xử lý các hàm dùng chung ví dụ như xử lý ngoại lệ trả về.

- Xử lý logic trả ra các ngoại lệ thông qua class `UserFriendlyException`:

```
public class UserFriendlyException : Exception
{
    14 references
    public UserFriendlyException(string message) : base(message)
    {
    }
}
```

- Truy vấn cơ sở dữ liệu quan hệ sử dụng Entity Framework Core cho Sql Server tiếp cận theo hướng Code First (tạo các migration update cấu trúc bảng vào database).
- Các bảng phải có khoá chính (primary key) dạng số nguyên và tự tăng.
- Các bảng có quan hệ khoá ngoại phải được tạo liên kết khoá ngoại trong migration.
- Các câu lệnh truy vấn cơ sở dữ liệu sử dụng các method Linq, keyword Linq.
- Xử lý logic nghiệp vụ theo design pattern Dependency injection (DI): dưới dạng các class service được inject vào controller khi sử dụng.
- Tổ chức các class theo hướng kế thừa (tùy chọn).
- Đặt tên các class theo dạng <Tên class><Mã số sinh viên><Đề thi số> (vd: StudentDto1234De1).

## 2. Bài toán cụ thể:

Cho quan hệ n-n Cửa hàng (tên, địa chỉ, giờ mở cửa, giờ đóng cửa - tên không được trùng) và nhà cung cấp (tên, địa chỉ, số điện thoại - tên không được trùng). Trong bảng quan hệ thêm một trường lưu mức độ thân thiết trên thang 100 giữa cửa hàng và nhà cung cấp dạng số thực.

- Thực hiện các chức năng sau:
  - Tạo migration và update vào database cấu trúc các bảng của bài toán trên.
  - Tạo các API **thêm, sửa, xóa** cửa hàng (các hàm thêm và sửa lưu ý kiểm tra trùng).
  - Tạo API xem **danh sách có phân trang** (gợi ý phân trang bằng PageSize và PageIndex) danh sách cửa hàng có cho phép lọc gần đúng theo tên hoặc theo địa chỉ (gợi ý dùng trường Keyword trong class filter Dto).
  - Tạo API liệt kê danh sách những nhà cung cấp có mức độ thân thiết cao nhất với đầu vào là id cửa hàng đầu ra là danh sách nhà cung cấp gồm **tên và số điện thoại** (Kiểm tra lại API bằng cách nhập các dữ liệu mẫu vào các bảng trong cơ sở dữ liệu).

*Ghi chú: Sinh viên đọc kỹ đề, cán bộ coi thi không giải thích gì thêm.*

----- Kết thúc -----