

BÀI 3: XÂY DỰNG ỨNG DỤNG WEBSITE BÁN HÀNG VỚI ASP.NET CORE MVC (PHẦN 1)

Sau khi học xong bài này, sinh viên có thể nắm được:

- Học cách sử dụng Entity Framework Core để tương tác với CSDL MS SQL Server.
- Xây dựng ứng dụng web bán hàng dựa trên ASP.NET Core MVC, kết hợp với cơ sở dữ liệu và Entity Framework Core.

3.1 Bài tập

3.1.1 Yêu cầu

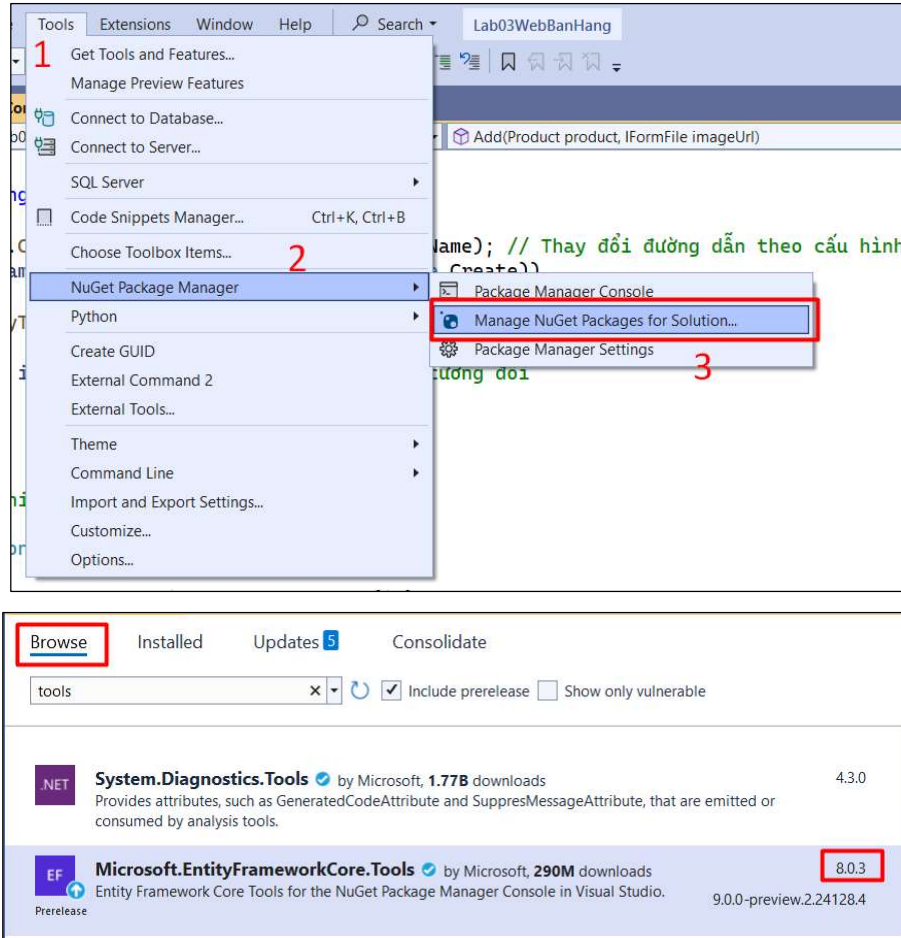
Xây dựng web bán hàng có các chức năng Hiển thị/ Thêm/ Xóa/ Sửa với Entity Framework Core và Cơ sở dữ liệu MS SQL Server

3.1.2 Hướng dẫn thực hiện

- *Cài Đặt Gói NuGet Cần Thiết*

Cài đặt các gói sau bằng NuGet:

- `Microsoft.EntityFrameworkCore` → phiên bản 8.0.3
- `Microsoft.EntityFrameworkCore.SqlServer` → phiên bản 8.0.3 (hoặc provider cơ sở dữ liệu khác)
- `Microsoft.EntityFrameworkCore.Tools` → phiên bản 8.0.3



Cài đặt Tương tự cho các Nuget khác.

- *Thiết Kế Models*

Trong thư Models Tạo các models `Product` và `Category` :

```
public class Product
{
    public int Id { get; set; }
    [Required, StringLength(100)]
    public string Name { get; set; }
    [Range(0.01, 10000.00)]
    public decimal Price { get; set; }
    public string Description { get; set; }
    public string? ImageUrl { get; set; }
    public List<ProductImage>? Images { get; set; }
    public int CategoryId { get; set; }
}
```

```

    public Category? Category { get; set; }
}

public class Category
{
    public int Id { get; set; }
    [Required, StringLength(50)]
    public string Name { get; set; }
    public List<Product>? Products { get; set; }
}

public class ProductImage
{
    public int Id { get; set; }
    public string Url { get; set; }
    public int ProductId { get; set; }
    public Product? Product { get; set; }
}

```

- *Cấu Hình Entity Framework Core*

Tạo một lớp **ApplicationDbContext** trong thư mục Models và cấu hình:

```

using Microsoft.EntityFrameworkCore;

public class ApplicationDbContext : DbContext
{
    Public ApplicationDbContext(DbContextOptions<ApplicationDbContext>
options) : base(options)
    {
    }

    public DbSet<Product> Products { get; set; }
    public DbSet<Category> Categories { get; set; }
    public DbSet<ProductImage> ProductImages { get; set; }
}

```

- *Cấu hình EF Core trong file **Program.cs**:*

```

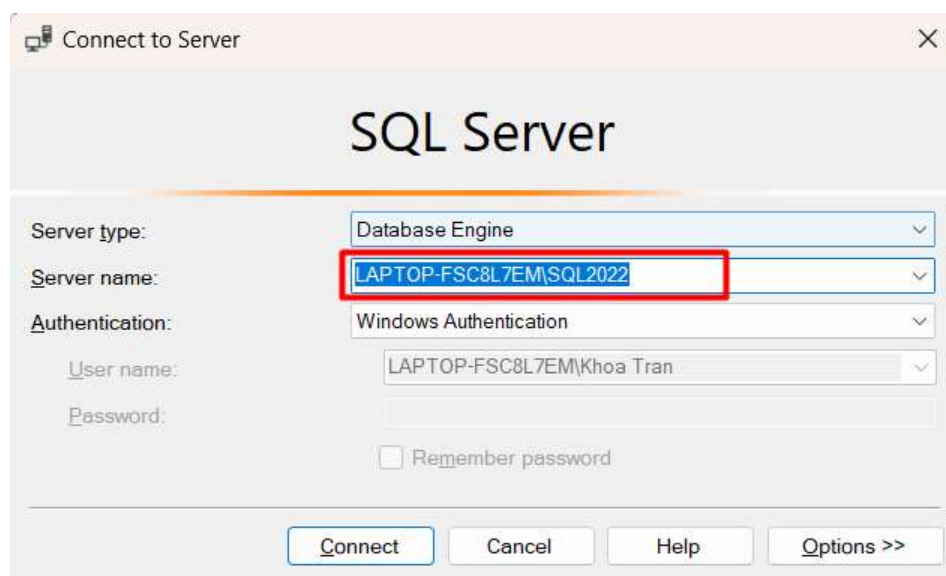
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext<ApplicationDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConne
ction")));

// Các cấu hình khác
...

```

- Cấu hình connection string trong `appsettings.json`.
- Xem tên Server name trong SQL Server



```
"ConnectionStrings": {  
  "DefaultConnection": "Server=LAPTOP-FSC8L7EM\\SQL2022;  
Database=Webbanhang;Trusted_Connection=True;TrustServerCertificate=True  
"
```

Lưu ý: Thay bằng thông tin **Server name** và **Database** của mình

- Tạo Migrations

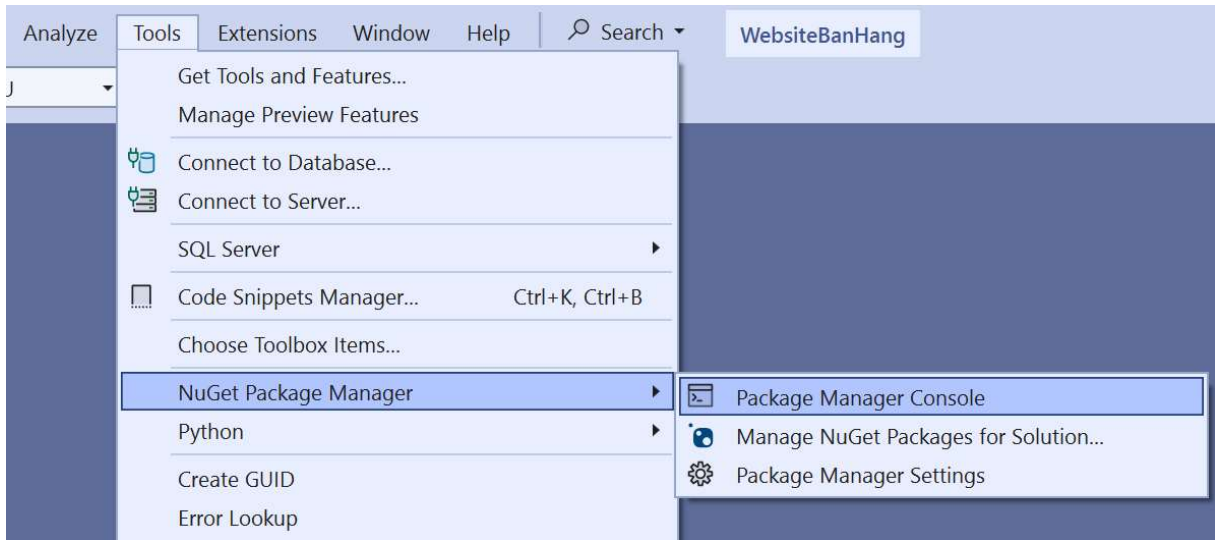
Cài đặt **dotnet-ef** tool bằng cách mở **Command Line (Admin)**

```
PS C:\Users\Khoa Tran> dotnet tool install --global dotnet-ef
```

dotnet-ef là một công cụ thiết yếu trong việc quản lý cơ sở dữ liệu trong các dự án sử dụng Entity Framework Core, giúp tự động hóa và quản lý các tác vụ liên quan đến cơ sở dữ liệu.

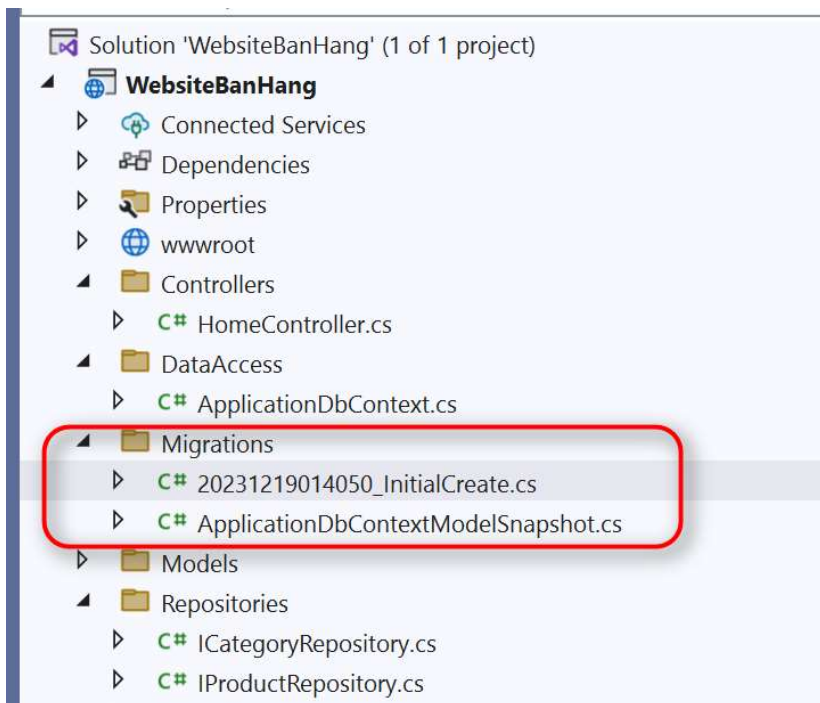
Gõ lệnh bên dưới để cài đặt

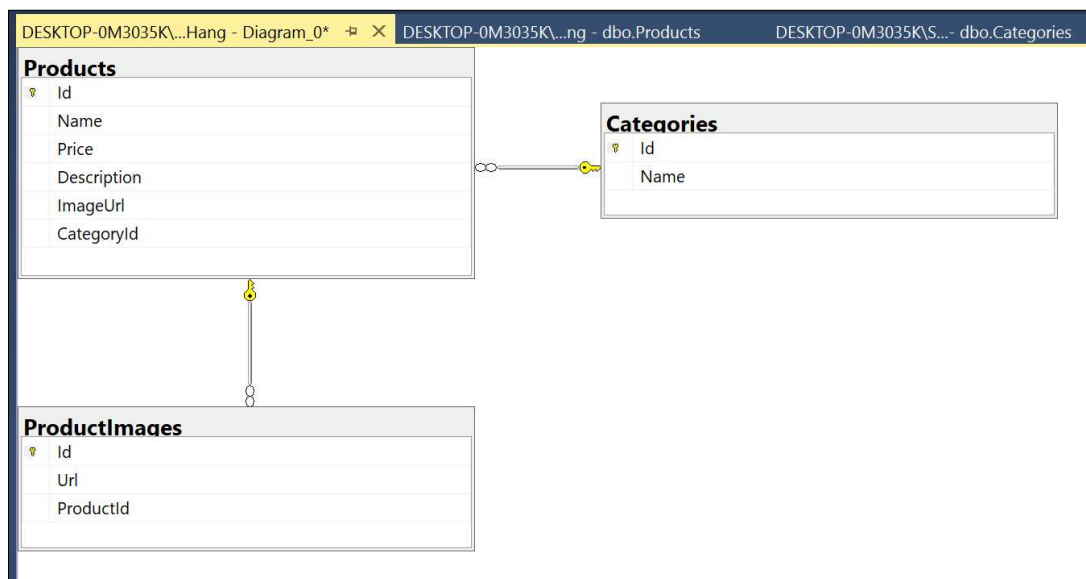
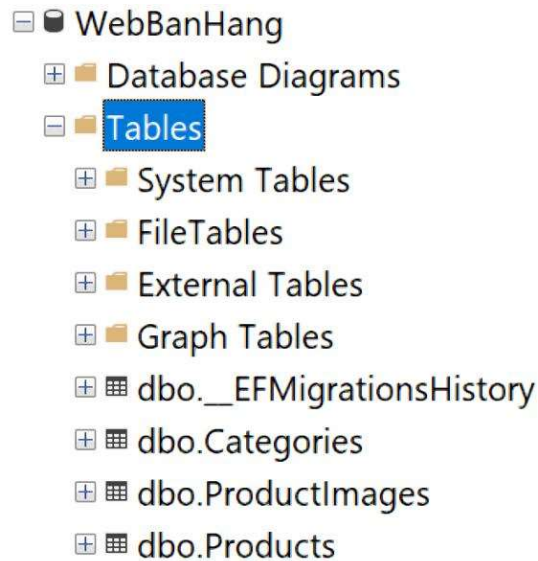
```
dotnet tool install --global dotnet-ef
```



Tạo migrations để tạo cơ sở dữ liệu:

```
dotnet ef migrations add InitialCreate --project {TenDuAn}  
dotnet ef database update --project {TenDuAn}
```





Tạo thư mục **Repositories** và thêm các file sau:

- *IproductRepository.cs* và *IcategoryRepository.cs*

Đây là các interface định nghĩa các phương thức cần thiết để tương tác với cơ sở dữ liệu cho Product và Category.

```
public interface IProductRepository
{
    Task<IEnumerable<Product>> GetAllAsync();
    Task<Product> GetByIdAsync(int id);
    Task AddAsync(Product product);
    Task UpdateAsync(Product product);
    Task DeleteAsync(int id);
}
```

```
public interface ICategoryRepository
{
    Task<IEnumerable<Category>> GetAllAsync();
    Task<Category> GetByIdAsync(int id);
    Task AddAsync(Category category);
    Task UpdateAsync(Category category);
    Task DeleteAsync(int id);
}
```

- *EFProductRepository* và *EFCategoryRepository*

```
public class EFProductRepository : IProductRepository
{
    private readonly ApplicationDbContext _context;

    public EFProductRepository(ApplicationDbContext context)
    {
        _context = context;
    }

    public async Task<IEnumerable<Product>> GetAllAsync()
    {
        // return await _context.Products.ToListAsync();
        return await _context.Products
            .Include(p => p.Category) // Include thông tin về category
            .ToListAsync();
    }

    public async Task<Product> GetByIdAsync(int id)
    {
        // return await _context.Products.FindAsync(id);
        // lấy thông tin kèm theo category
        return await _context.Products.Include(p =>
p.Category).FirstOrDefaultAsync(p => p.Id == id);
    }

    public async Task AddAsync(Product product)
    {
        _context.Products.Add(product);
        await _context.SaveChangesAsync();
    }

    public async Task UpdateAsync(Product product)
    {
        _context.Products.Update(product);
        await _context.SaveChangesAsync();
    }

    public async Task DeleteAsync(int id)
    {

```

```
        var product = await _context.Products.FindAsync(id);
        _context.Products.Remove(product);
        await _context.SaveChangesAsync();
    }
}
```

```
public class EFCategoryRepository : ICategoryRepository
{
    private readonly ApplicationDbContext _context;

    public EFCategoryRepository(ApplicationDbContext context)
    {
        _context = context;
    }

    // Tương tự như EFProductRepository, nhưng cho Category
}
```

- Đăng Ký trong *Program.cs*

```
builder.Services.AddScoped<IProductRepository, EFProductRepository>();
builder.Services.AddScoped<ICategoryRepository, EFCategoryRepository>();
```

- Tạo và Cập Nhật Controllers

Sử Dụng Repository trong Controllers

```
using ProjectName.Models;
using ProjectName.Repositories;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace ProjectName.Controllers
{
    public class ProductController : Controller
    {

        private readonly IProductRepository _productRepository;
        private readonly ICategoryRepository _categoryRepository;

        public ProductController(IProductRepository productRepository,
            ICategoryRepository categoryRepository)
        {
            _productRepository = productRepository;
        }
    }
}
```



```

        _categoryRepository = categoryRepository;
    }

    // Hiển thị danh sách sản phẩm
    public async Task<IActionResult> Index()
    {
        var products = await _productRepository.GetAllAsync();
        return View(products);
    }

    // Hiển thị form thêm sản phẩm mới
    public async Task<IActionResult> Add()
    {
        var categories = await _categoryRepository.GetAllAsync();
        ViewBag.Categories = new SelectList(categories, "Id", "Name");

        return View();
    }

    // Xử lý thêm sản phẩm mới
    [HttpPost]
    public async Task<IActionResult> Add(Product product, IFormFile
imageUrl)
    {
        if (ModelState.IsValid)
        {
            if (imageUrl != null)
            {
                // Lưu hình ảnh đại diện tham khảo bài 02 hàm SaveImage
                product.ImageUrl = await SaveImage(imageUrl);
            }

            await _productRepository.AddAsync(product);
            return RedirectToAction(nameof(Index));
        }

        // Nếu ModelState không hợp lệ, hiển thị form với dữ liệu đã nhập
        var categories = await _categoryRepository.GetAllAsync();
        ViewBag.Categories = new SelectList(categories, "Id", "Name");
        return View(product);
    }

    // Viết thêm hàm SaveImage (tham khảo bài 02)

    // Hiển thị thông tin chi tiết sản phẩm
    public async Task<IActionResult> Display(int id)
    {
        var product = await _productRepository.GetByIdAsync(id);
        if (product == null)
        {
            return NotFound();
        }
        return View(product);
    }

```

```
// Hiển thị form cập nhật sản phẩm
public async Task<IActionResult> Update(int id)
{
    var product = await _productRepository.GetByIdAsync(id);
    if (product == null)
    {
        return NotFound();
    }

    var categories = await _categoryRepository.GetAllAsync();
    ViewBag.Categories = new SelectList(categories, "Id", "Name",
product.CategoryId);
    return View(product);
}
// Xử lý cập nhật sản phẩm
[HttpPost]
public async Task<IActionResult> Update(int id, Product product,
IFormFile imageUrl)
{
    ModelState.Remove("ImageUrl"); // Loại bỏ xác thực ModelState cho
ImageUrl
    if (id != product.Id)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        var existingProduct = await
_productRepository.GetByIdAsync(id); // Giả định có phương thức GetByIdAsync
tải lên
        // Giữ nguyên thông tin hình ảnh nếu không có hình mới được
        if (imageUrl == null)
        {
            product.ImageUrl = existingProduct.ImageUrl;
        }
        else
        {
            // Lưu hình ảnh mới
            product.ImageUrl = await SaveImage(imageUrl);
        }
        // Cập nhật các thông tin khác của sản phẩm
        existingProduct.Name = product.Name;
        existingProduct.Price = product.Price;
        existingProduct.Description = product.Description;
        existingProduct.CategoryId = product.CategoryId;
        existingProduct.ImageUrl = product.ImageUrl;

        await _productRepository.UpdateAsync(existingProduct);

        return RedirectToAction(nameof(Index));
    }
}
```

```

    }
    var categories = await _categoryRepository.GetAllAsync();
    ViewBag.Categories = new SelectList(categories, "Id", "Name");
    return View(product);
}

// Hiển thị form xác nhận xóa sản phẩm
public async Task<IActionResult> Delete(int id)
{
    var product = await _productRepository.GetByIdAsync(id);
    if (product == null)
    {
        return NotFound();
    }
    return View(product);
}

// Xử lý xóa sản phẩm
[HttpPost, ActionName("Delete")]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    await _productRepository.DeleteAsync(id);
    return RedirectToAction(nameof(Index));
}
}
}

```

- *Tạo và Cập nhật Views*

Xây dựng các view của `ProductController` trong ASP.NET Core MVC, bao gồm các trang hiển thị danh sách sản phẩm, thêm sản phẩm mới, cập nhật thông tin sản phẩm, và xóa sản phẩm.

- *Index.cshtml (Danh sách sản phẩm)*

```

@model IEnumerable<YourNamespace.Models.Product>

<h2>Products</h2>

<table class="table">
    <thead>
        <tr>
            <th>Name</th>
            <th>Price</th>
            <th>Description</th>
            <th>Category</th>
            <th>Actions</th>
        </tr>
    </thead>

```

```

</thead>
<tbody>
@foreach (var product in Model)
{
    <tr>
        <td>@product.Name</td>
        <td>@product.Price</td>
        <td>@product.Description</td>
        <td>@product.Category?.Name ?? "No Category"</td>
        <td>
            <a asp-action="Display" asp-route-
id="@product.Id">View</a> |
            <a asp-action="Update" asp-route-
id="@product.Id">Edit</a> |
            <a asp-action="Delete" asp-route-
id="@product.Id">Delete</a>
        </td>
    </tr>
}
</tbody>
</table>

```

- *Add.cshtml (Thêm sản phẩm mới)*

```

@model YourNamespace.Models.Product
@using Microsoft.AspNetCore.Mvc.Rendering
@{
    ViewData["Title"] = "Add Product";
}

<h2>Add Product</h2>

<form asp-action="Add" method="post" enctype="multipart/form-data">
    <div class="form-group">
        <label asp-for="Name"></label>
        <input asp-for="Name" class="form-control" />
        <span asp-validation-for="Name" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Price"></label>
        <input asp-for="Price" class="form-control" />
        <span asp-validation-for="Price" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Description"></label>
        <textarea asp-for="Description" class="form-control"></textarea>
        <span asp-validation-for="Description" class="text-
danger"></span>
    </div>
</form>

```

```

    </div>
    <div class="form-group">
        <label asp-for="CategoryId">Category</label>
        <select asp-for="CategoryId" asp-items="ViewBag.Categories"
class="form-control"></select>
    </div>
    <div class="form-group">
        <label asp-for="ImageUrl">Product Image</label>
        <input type="file" asp-for="ImageUrl" class="form-control" />
    </div>
    <button type="submit" class="btn btn-primary">Add</button>
</form>

```

- *Display.cshtml (Hiển thị thông tin chi tiết sản phẩm)*

```

@model YourNamespace.Models.Product

<h2>@Model.Name</h2>

<div>
    <h3>Price: @Model.Price</h3>
    <p>@Model.Description</p>
    @if (!string.IsNullOrEmpty(Model.ImageUrl))
    {
        
    }
</div>

<a asp-action="Index">Back to List</a>

```

- *Update.cshtml (Cập nhật thông tin sản phẩm)*

```

@model YourNamespace.Models.Product
@using Microsoft.AspNetCore.Mvc.Rendering
@{
    ViewData["Title"] = "Edit Product";
}

<h2>Edit Product</h2>

<form asp-action="Update" method="post" enctype="multipart/form-data">
    <input type="hidden" asp-for="Id" />
    <div class="form-group">
        <label asp-for="Name"></label>
        <input asp-for="Name" class="form-control" />
        <span asp-validation-for="Name" class="text-danger"></span>
    </div>

```

```

<div class="form-group">
  <label asp-for="Price"></label>
  <input asp-for="Price" class="form-control" />
  <span asp-validation-for="Price" class="text-danger"></span>
</div>
<div class="form-group">
  <label asp-for="Description"></label>
  <textarea asp-for="Description" class="form-control"></textarea>
  <span asp-validation-for="Description" class="text-
danger"></span>
</div>
<div class="form-group">
  <label asp-for="CategoryId">Category</label>
  <select asp-for="CategoryId" asp-items="@ViewBag.Categories"
class="form-control"></select>
</div>
<div class="form-group">
  <label asp-for="ImageUrl">Product Image</label>
  <input type="file" asp-for="ImageUrl" class="form-control" />
  
</div>
<button type="submit" class="btn btn-primary">Update</button>
</form>

```

- *Delete.cshtml (Xác nhận xóa sản phẩm)*

```

@model YourNamespace.Models.Product

<h2>Delete Product</h2>

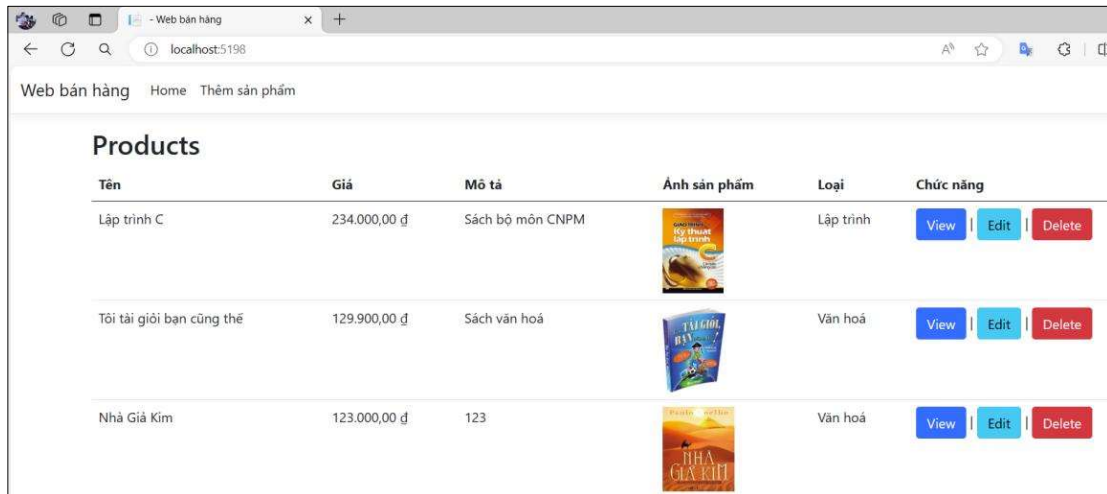
<form asp-action="DeleteConfirmed" method="post">
  <input type="hidden" asp-for="Id" />
  <p>Are you sure you want to delete @Model.Name?</p>
  <button type="submit" class="btn btn-danger">Delete</button>
</form>

```

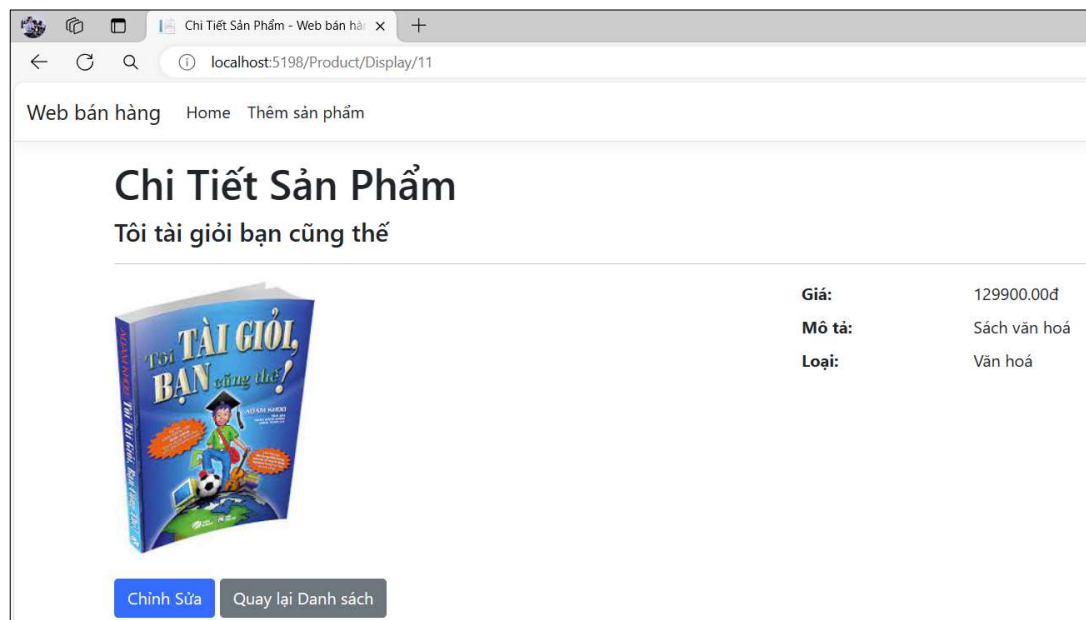
3.1.3 Kết quả

Kết quả, hướng dẫn trên chỉ cơ bản. Sinh viên hoàn thiện thêm để được kết quả bên dưới. Áp dụng bài số 2 (thiết kế giao diện) để hoàn thiện thêm.

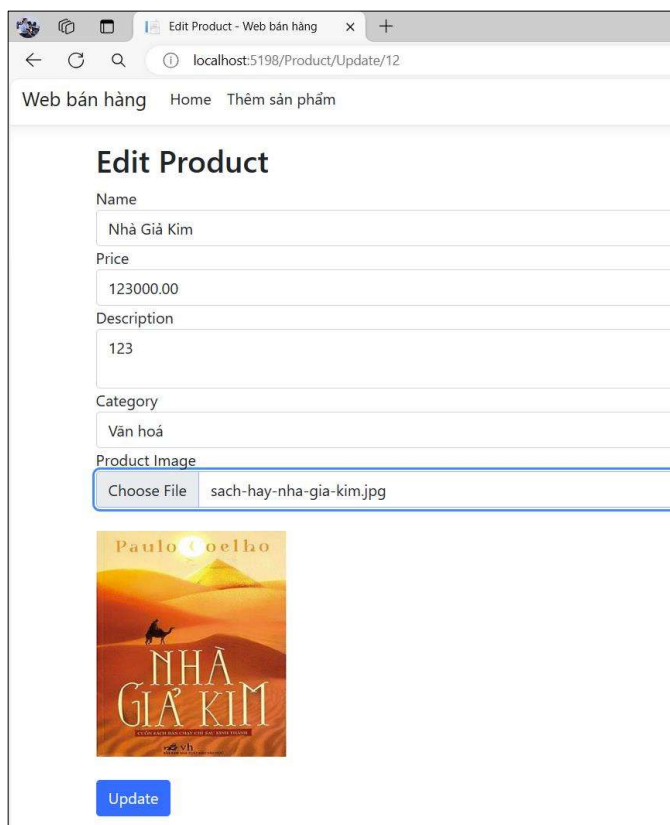
Trang chủ website.



Trang chi tiết sản phẩm.



Trang Edit sản phẩm.



Web bán hàng Home Thêm sản phẩm

Edit Product

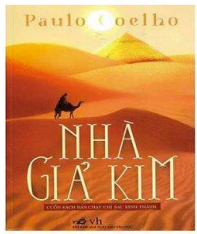
Name
Nhà Giả Kim

Price
123000.00

Description
123

Category
Văn hoá

Product Image
Choose File sach-hay-nha-gia-kim.jpg



Update

3.1.4 Yêu cầu bổ sung

Yêu cầu:

Ở trang chỉnh sửa sản phẩm **Update.cshtml**, khi thay đổi ảnh khác thì ảnh sẽ cập nhật hình ảnh hiện ngay trên trang.

Gợi ý dùng:

```
30 <div class="form-group">
31 <label asp-for="ImageUrl">Product Image</label>
32 <input type="file" asp-for="ImageUrl" class="form-control" id="imageInput" />
33 <br />
34 <img id="previewImage" style="max-width: 200px; display: none;" alt="Image preview" />
35 </div>
36 <br />
37 <button type="submit" class="btn btn-primary">Update</button>
38 </form>
39
40 <script>
41 document.getElementById('imageInput').addEventListener('change', function (event) {
42     var output = document.getElementById('previewImage');
43     if (this.files && this.files[0]) {
44         var reader = new FileReader();
45
46         reader.onload = function (e) {
47             output.src = e.target.result;
48             output.style.display = 'block'; // Show the image
49         };
50
51         reader.readAsDataURL(this.files[0]);
52     }
53 });
54 </script>
```

Thực hiện cho các chức năng **CRUD** tương tự cho **Category**

Lưu ý:

- Thay thế `YourNamespace.Models.Product` với namespace thực tế của model `Product` trong dự án của bạn.
- Đảm bảo rằng `ViewBag.Categories` được đúng cách gán giá trị trong controller trước khi gửi đến `Add.cshtml` và `Update.cshtml`.
- Thêm logic để xử lý đường dẫn hình ảnh trong `Display.cshtml` và `Update.cshtml` tùy theo cách bạn lưu trữ và truy xuất hình ảnh.