Bu uygulamanın amacı, çeşitli ürünlerin bilgileriyle birlikte listelenmesi ve filtreleme ile arama yapmak olacak.

// kategori sınıfı

// kategorilerin id ve adını tutuyor

public class Category

{

public int CategoryId { get; set; }

public string? Name { get; set; } = string.Empty;

}

// ürün sınıfı

// ürünlerin id, ad, fiyat, resim, aktiflik durumu ve kategori id'sini tutuyor

// kategori id'si ile kategorilerle ilişkilendirilecek

// aktiflik durumu ile ürünlerin aktif olup olmadığını belirleyeceğiz vs.

public class Product

{

public int ProductId { get; set; }

public string? Name { get; set; } = string.Empty;

public int Price { get; set; }

public string Image { get; set; } = string.Empty;

public bool IsActive { get; set; }

public int CategoryId { get; set; }

}

// Veri tabanı yerine geçiyor

// Repository Türkçe’de depo anlamına gelir.

public class Repository

{

// ürünlerin ve kategorilerin listesi

// ürünleri ve kategorileri buradan ekleyeceğiz, ve burayı geçici veri tabanıymış gibi kullanacağız

private static readonly List<Product> \_products = new();

private static readonly List<Category> \_categories = new();

// ürünlerin ve kategorilerin eklenmesini bu kurucu metotta yapacağız

static Repository()

{

\_categories.Add(new Category { CategoryId = 1, Name = "Telefon" });

\_categories.Add(new Category { CategoryId = 2, Name = "Bilgisayar" });

\_products.Add(new Product { ProductId = 1, Name = "iPhone 14", Price = 2000, Image = "1.jpg", IsActive = true, CategoryId = 1 });

\_products.Add(new Product { ProductId = 2, Name = "iPhone 15", Price = 3000, Image = "2.jpg", IsActive = true, CategoryId = 1 });

\_products.Add(new Product { ProductId = 3, Name = "Samsung S7", Price = 4000, Image = "3.jpg", IsActive = true, CategoryId = 1 });

\_products.Add(new Product { ProductId = 4, Name = "Samsung S8", Price = 5000, Image = "4.jpg", IsActive = true, CategoryId = 1 });

\_products.Add(new Product { ProductId = 6, Name = "Macbook Air", Price = 7000, Image = "5.jpg", IsActive = true, CategoryId = 2 });

\_products.Add(new Product { ProductId = 7, Name = "Macbook Pro", Price = 8000, Image = "6.jpg", IsActive = true, CategoryId = 2 });

}

// ürünlerin çağırılması için oluşturduğumuz metot

public static List<Product> Products

{

get { return \_products; }

}

// kategorilerin çağırılması için oluşturduğumuz metot

public static List<Category> Categories

{

get { return \_categories; }

}

}

public IActionResult Index()

{

//Ürün listesi View sayfasına gönderiliyor

return View(Repository.Products);

}

<tr>

@\* İlgili Product classında yer alan Display(Name) özelliği kullanılarak tablo başlıkları doğrudan buraya geliyor \*@

<th>@Html.DisplayNameFor(m => m.ProductId)</th>

<th>@Html.DisplayNameFor(m => m.Name)</th>

<th>@Html.DisplayNameFor(m => m.Price)</th>

<th style="width: 50px">@Html.DisplayNameFor(m => m.Image)</th>

<th>@Html.DisplayNameFor(m => m.IsActive)</th>

<th>@Html.DisplayNameFor(m => m.CategoryId)</th>

<th>Düzenle</th>

</tr>

Product sayfası

public class Product

{

[Display(Name = "Ürün Id")]

public int ProductId { get; set; }

[Display(Name = "Ürün Adı")]

public string? Name { get; set; } = string.Empty;

[Display(Name = "Fiyat")]

public int Price { get; set; }

[Display(Name = "Resim")]

public string Image { get; set; } = string.Empty;

[Display(Name = "Durum")]

public bool IsActive { get; set; }

[Display(Name = "Kategori")]

public int CategoryId { get; set; }

}

[HttpGet]

public IActionResult Index(string searchString)

{

// products değişkenine tüm ürünler koyuluyor

var products = Repository.Products;

// searchstring null veya boş değilse eğer bu koşul çalışıyor

if (!String.IsNullOrEmpty(searchString))

{

// Ürünlerin içinde arama yapılıyor

// Mevcut ürünler arasından where koşuluyla linq expression kullanılıyor

// p denilen değişken products içerisinde yer alan her bir ürünü temsil ediyor

// bu, products içerisindeki ürünlerin adları içerisinde, bizim search stringimiz var mı kontrol ediliyor

// varsa eğer, listeye çevirilerek, productsa kaydediliyor

products = products.Where(p => p.Name.ToLower().Contains(searchString)).ToList();

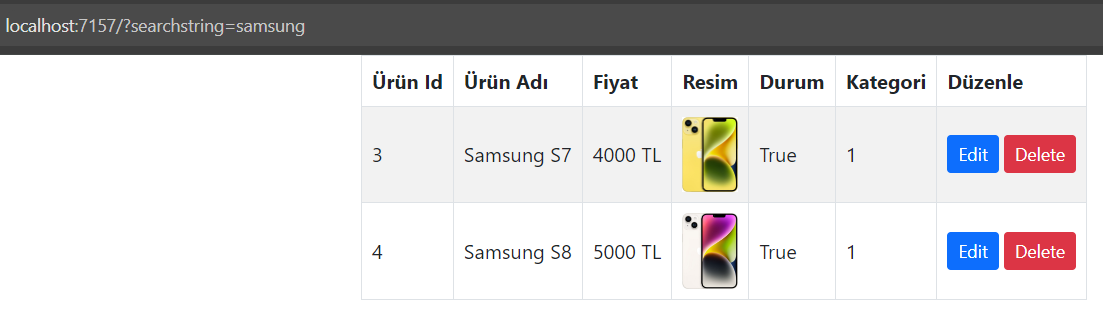
}

// bundan sonra tarayıcıda linkin sonuna ?searchString=iphone yazarak arama yapabiliriz

return View(products);

}

Sadece samsung kelimesini içeren ürünler:



// bind edilecek alanları burada da tanımlayabiliyoruz

// bind edilmesini istemediğimiz alanlara giderek BindNever yazarsak da bind edilmelerini engellemiş oluyoruz

[Bind("Name", "Price")]

public class Product

{

// Product Id'nin namenin çağırılmasıyla tabloda nasıl görünmesi gerektiğini belirtiyoruz

[Display(Name = "Ürün Id")]

[Required]

// BindNever attribute'u ile bu alanın bind edilmemesini sağlıyabiliyoruz

[BindNever]

public int ProductId { get; set; }

Yukarıdaki yöntemlere ek olarak controller üzerinde de bind işlemi gerçekleştirebiliyoruz.

// bind kullanarak, viewden sadece Name ve Price alanlarını alıyoruz

[HttpPost]

public IActionResult Create([Bind("Name", "Price")]Product model)

{

ViewBag.Categories = new SelectList(Repository.Categories, "CategoryId", "Name");

return View();

}

@\* kategori id için asp for kullanıldı

kategori dropdown listesi için viewbag kullanıldı \*@

<div class="mb-3">

<label class="form-label" for="CategoryId">Category</label>

<**select** **asp-for**="CategoryId" **asp-items**="@ViewBag.Categories" class="form-select">

<**option** **value**="0">Hepsi</**option**>

</**select**>

</div>