WebShopApp - Част 4 - Създаване на сървиси, контролер и вюта за продукт

В папка **Contracts** добавяме интерфейс за продукт **IProductService.cs**

```
public interface IProductService
{
    bool Create(string name, int brandId, int categoryId, string picture, int quantity, decimal price, decimal
      discount);

    bool Update(int productId, string name, int brandId, int categoryId, string picture, int quantity, decimal
      price, decimal discount);

    List<Product> GetProducts();

    Product GetProductById(int productId);

    bool RemoveById(int dogproductId);

    List<Product> GetProducts(string searchStringCategoryName, string searchStringBrandName);
}
```

В папка **Services** трябва да имплементираме интерфейса **IProductService** и неговите методи в клас **ProductService.cs**

```
public class ProductService : IProductService
{
    private readonly ApplicationDbContext _context;

    public ProductService(ApplicationDbContext context)
    {
        _context = context;
    }
    public bool Create(string name, int brandId, int categoryId, string picture, int quantity, decimal price,
      decimal discount)
    {
        Product item = new Product
        {
            ProductName = name,
            Brand = _context.Brands.Find(brandId),
            Category = _context.Categories.Find(categoryId),

            Picture = picture,
            Quantity = quantity,
            Price = price,
            Discount = discount
        };

        _context.Products.Add(item);
        return _context.SaveChanges() != 0;

    }
```

```
public Product GetProductById(int productId)
{
    return _context.Products.Find(productId);

}

public List<Product> GetProducts()
{
    List<Product> products = _context.Products.ToList();
    return products;
}
```

```
public List<Product> GetProducts(string searchStringCategoryName, string searchStringBrandName)
{
    List<Product> products = _context.Products.ToList();
    if (!String.IsNullOrEmpty(searchStringCategoryName) && !String.IsNullOrEmpty(searchStringBrandName))
    {
        products = products.Where(x => x.Category.CategoryName.ToLower().Contains
          (searchStringCategoryName.ToLower())
        && x.Brand.BrandName.ToLower().Contains(searchStringBrandName.ToLower())).ToList();
    }
    else if (!String.IsNullOrEmpty(searchStringCategoryName))
    {
        products = products.Where(x => x.Category.CategoryName.ToLower().Contains
          (searchStringCategoryName.ToLower())).ToList();
    }
    else if (!String.IsNullOrEmpty(searchStringBrandName))
    {
        products = products.Where(x => x.Brand.BrandName.ToLower().Contains(searchStringBrandName.ToLower
          ())).ToList();
    }
    return products;

}
```

```
public bool RemoveById(int productId)
{
    var product = GetProductById(productId);
    if (product == default(Product))
    {
        return false;
    }
    _context.Remove(product);
    return _context.SaveChanges() != 0;

}
```

```
public bool Update(int productId, string name, int brandId, int categoryId, string picture,
    int quantity, decimal price, decimal discount)
{
    var product = GetProductById(productId);
    if (product == default(Product))
    {
        return false;
    }
    product.ProductName = name;

    //product.BrandId = brandId;
    //product.CategoryId = categoryId;

    product.Brand = _context.Brands.Find(brandId);
    product.Category = _context.Categories.Find(categoryId);

    product.Picture = picture;
    product.Quantity = quantity;
    product.Price = price;
    product.Discount = discount;
    _context.Update(product);
    return _context.SaveChanges() != 0;

}
```

Когато сме готови с имплементацията на ProductServise трябва да го добавим в Program.cs така както добавихме сървисите за категории и производители.

```
builder.Services.AddControllersWithViews();

builder.Services.AddTransient<ICategoryService, CategoryService>();
builder.Services.AddTransient<IBrandService, BrandService>();
builder.Services.AddTransient<IProductService, ProductService>();

var app = builder.Build();
```

Билднете проекта и се уверете, че не сте допуснали грешки.

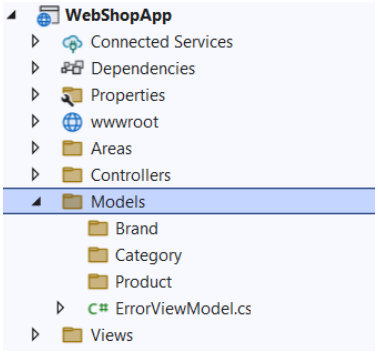
Имплементиране на CRUD операции за продуктите

Сега може да пристъпим към създаване на моделите, вютата и контролерите на приложението.

Първо ще направим моделите, които ще използваме за вютата в приложението.

В папка Models ще направим отделни папки за моделите към всяко ентити.

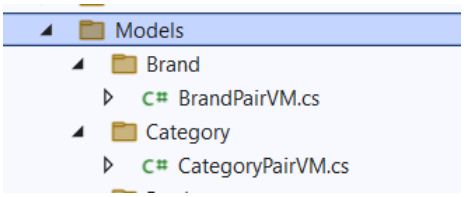
Нека направим три папки за категориите, производителите и продуктите съответно Category, Brand и Product.



Създаване на модели за Brand.

Ще ни трябват два модел : BrandPairVM и CategoryPairVM. С тяхна помощ при зареждане на формата за създаване на продукт, ще можем от падащ списък да избираме бранд и категория на продукта.

```
public class BrandPairVM
{
    public int Id { get; set; }

    [Display(Name = "Brand")]
    public string Name { get; set; } = null!;
}
```



Аналогично правим CategoryPairVM.cs за Category.

```
public class CategoryPairVM
{
    public int Id { get; set; }

    [Display(Name = "Category")]
    public string Name { get; set; } = null!;
}
```

За Brand и Category няма да създаваме контролери и вюта, т.к. сиднахме данните за тях и те се попълват при стартиране на приложението.

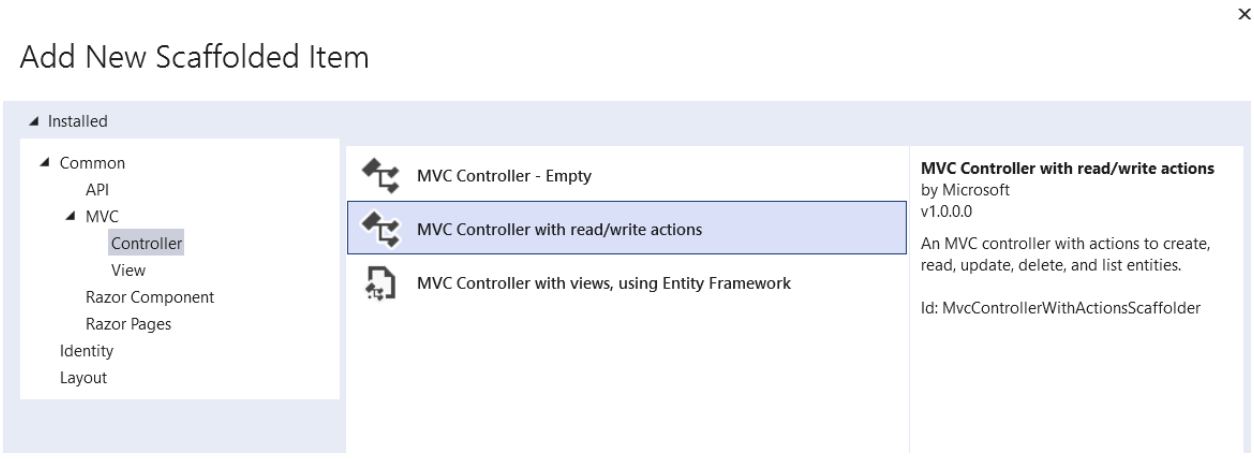
Модели, вюта и контролер за Product

В папка Models/Product създаваме съответните модели за Product. Започваме със създаването на продукт.

ProductCreateVM.cs

```
public class ProductCreateVM
{
    [Key]
    public int Id { get; set; }

    [Required]
    [MaxLength(30)]
    [Display(Name = "Product Name")]
    public string ProductName { get; set; } = null!;

    [Required]
    [Display(Name = "Brand")]
    public int BrandId { get; set; }
    public virtual List<BrandPairVM> Brands { get; set; } = new List<BrandPairVM>();

    [Required]
    [Display(Name = "Category")]
    public int CategoryId { get; set; }
    public virtual List<CategoryPairVM> Categories { get; set; } = new List<CategoryPairVM>();

    [Display(Name = "Picture")]
    public string Picture { get; set; } = null!;

    [Range(0, 5000)]
    [Display(Name = "Quantity")]
    public int Quantity { get; set; }

    [Display(Name = "Price")]
    public decimal Price { get; set; }

    [Display(Name = "Discount")]
    public decimal Discount { get; set; }
}
```

Да направим ProductController и в него да инжектираме през конструктора ProductService, BrandServise и CategoryServise. Последните две ще ни трябват за създаване на дроп-даун полета с производителите и категориите.



Имплементираме първо екшъна Create(). Екшънът по метод GETслужи за зареждане на формата, а този по POST служи за изпращане на въведените от потребителя данни към сървъра. Във формата трябва да намерим заредени категориите и брандовете.

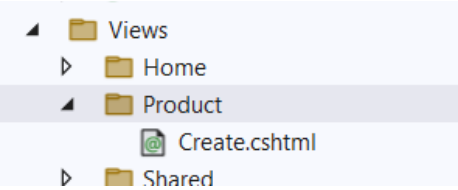
```
public class ProductController : Controller
{
    private readonly IProductService _productService;
    private readonly ICategoryService _categoryService;
    private readonly IBrandService _brandService;

    public ProductController(IProductService productService, ICategoryService categoryService, IBrandService brandService)
    {
        this._productService = productService;
        this._categoryService = categoryService;
        this._brandService = brandService;
    }
```

Ето как трябва да изглежда методът Create(). Още при появата на формата трябва да се заредят производителите и категориите, извлечени от БД.

```
// GET: ProductController/Create
public ActionResult Create()
{
    var product = new ProductCreateVM();
    product.Brands = _brandService.GetBrands()
        .Select(x => new BrandPairVM()
        {
            Id = x.Id,
            Name = x.BrandName
        }).ToList();
    product.Categories = _categoryService.GetCategories()
       .Select(x => new CategoryPairVM()
       {
           Id = x.Id,
           Name = x.CategoryName
       }).ToList();
    return View(product);
}

// POST: ProductController/Create
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([FromForm] ProductCreateVM product)
{
    if (ModelState.IsValid)
    {
        var createdId = _productService.Create(product.ProductName, product.BrandId,
        product.CategoryId, product.Picture,
        product.Quantity, product.Price, product.Discount);
        if (createdId)
        {
            return RedirectToAction(nameof(Index));
        }
    }

    return View();
}
```

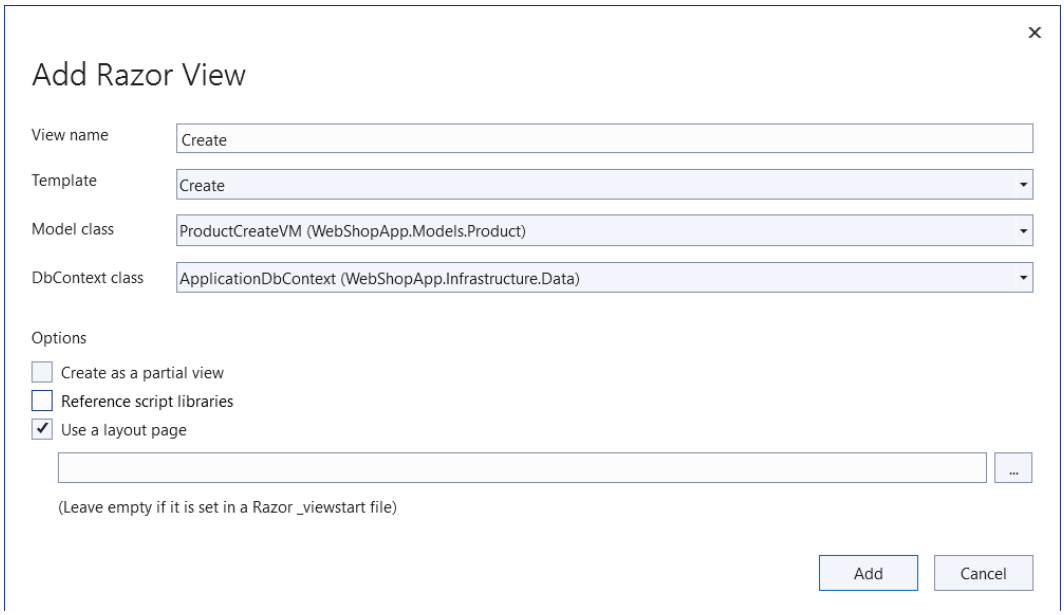
За създаване на продукт ни остава само да направим вюто, в което ще поставим дроп-даун меню на мястото на производителите и категориите.

В папка Views правим папка с името на контролера Product и в нея създаваме вюта с имена, съответстващи на всеки екшън от контролера, като на вюто му подаваме съответния вю модел от папка Models.



Създаване на Create.cshtml

Опитайте да създадете Razor View, а ако не се получи успешно генериране на код, копирайте дадения по-долу.

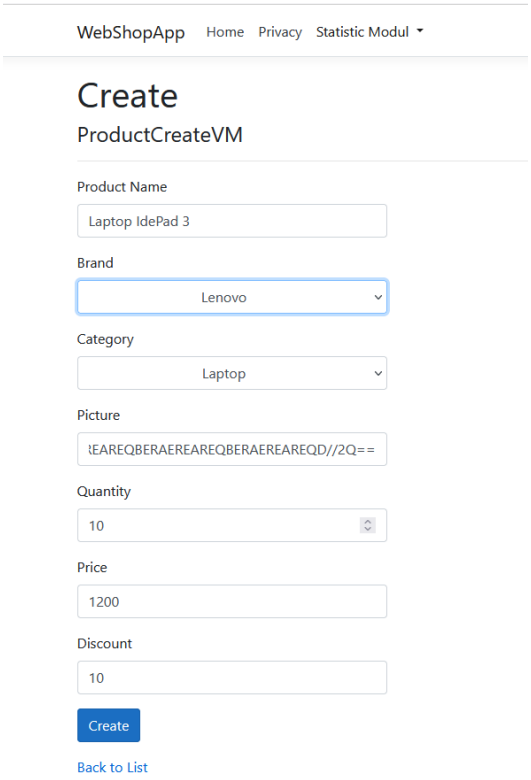


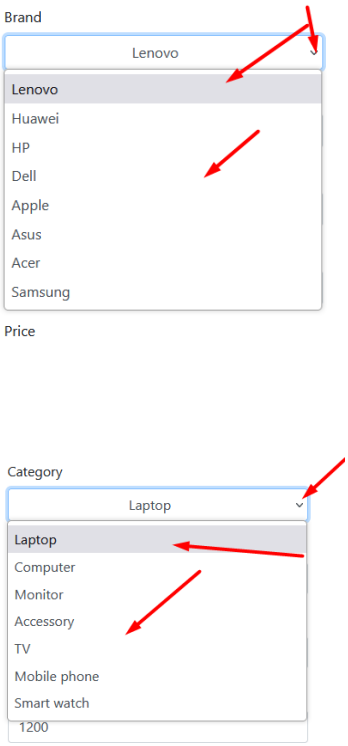
Изглед Create

```
@model WebShopApp.Models.Product.ProductCreateVM

@{
    ViewData["Title"] = "Create";
}

<h1>Create</h1>

<h4>ProductCreateVM</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="ProductName" class="control-label"></label>
                <input asp-for="ProductName" class="form-control" />
                <span asp-validation-for="ProductName" class="text-danger"></span>
```

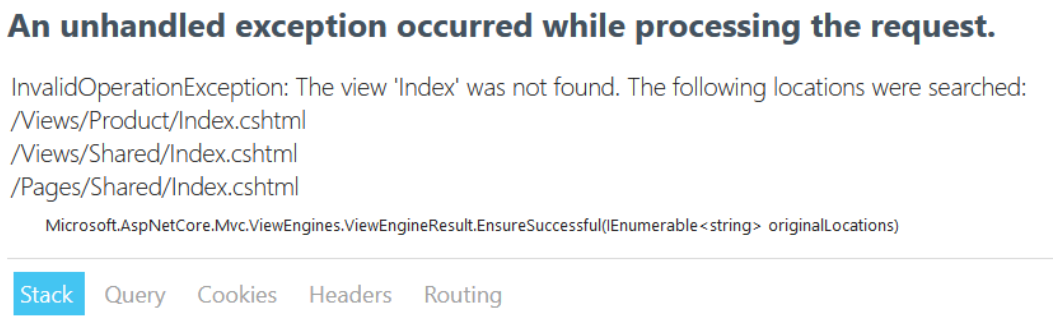
```
            </div>
            <div class="form-group">
                <label asp-for="BrandId" class="control-label"></label>
                @*<input asp-for="BrandId" class="form-control" />*@
                <select asp-for="BrandId" class="form-control text-center">
                    @foreach (var brand in Model.Brands)
                    {
                        <option value=" @brand.Id">@brand.Name</option>
                    }
                </select>
                <span asp-validation-for="BrandId" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="CategoryId" class="control-label"></label>
                @*<input asp-for="CategoryId" class="form-control" />*@
                <select asp-for="CategoryId" class="form-control text-center">
                    @foreach (var category in Model.Categories)
                    {
                        <option value="@category.Id">@category.Name</option>
                    }
                </select>
                <span asp-validation-for="CategoryId" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Picture" class="control-label"></label>
                <input asp-for="Picture" class="form-control" />
                <span asp-validation-for="Picture" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Quantity" class="control-label"></label>
                <input asp-for="Quantity" class="form-control" />
                <span asp-validation-for="Quantity" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Price" class="control-label"></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="Discount" class="control-label"></label>
                <input asp-for="Discount" class="form-control" />
                <span asp-validation-for="Discount" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>
@section Scripts {
    @{
        await Html.RenderPartialAsync("_ValidationScriptsPartial");
    }
}
```

Тествайте приложението, като създадете продукт и се уверите, че всичко работи.

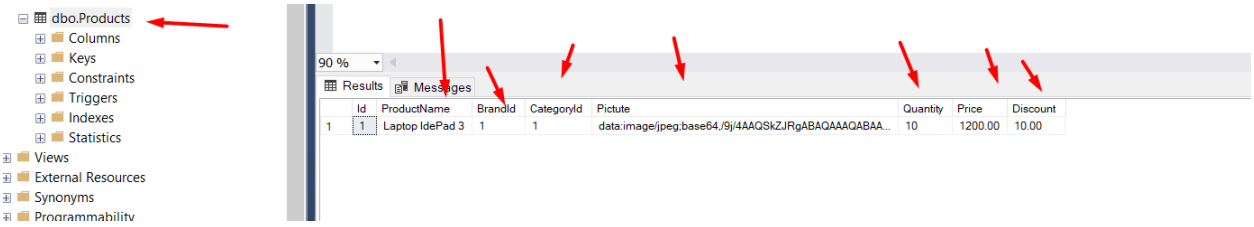




След натискане на бутона Create би трябвало да получите следната грешка, т.к. все още не сме създали вю за екшъна Index() на ProductController



Затова проверете в БД дали се е запазил продуктът.



Имплементация на Index() за Product

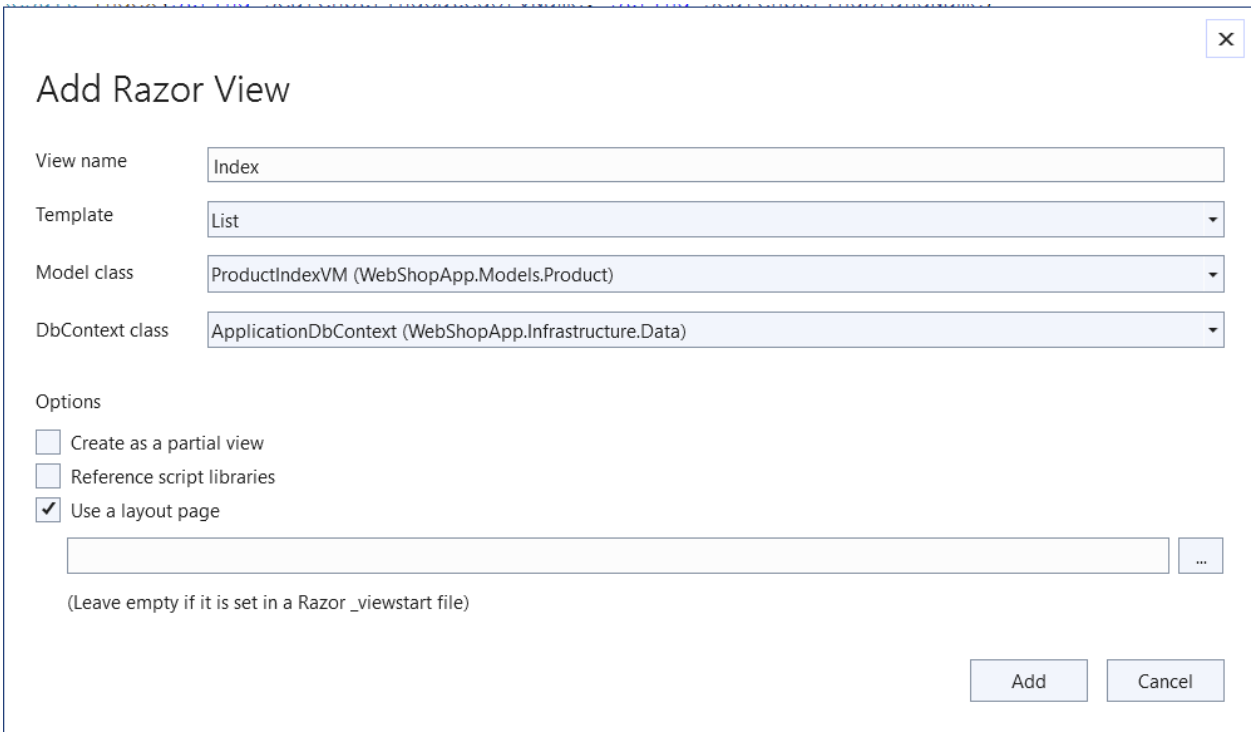
Първо създаваме модел ProductIndexVM в папка Models/Product

```
6 references
public class ProductIndexVM
{
    [Key]
    4 references
    public int Id { get; set; }
    [Display(Name = "Product Name")]
    3 references
    public string ProductName { get; set; }

    1 reference
    public int BrandId { get; set; }
    [Display(Name = "Brand")]
    3 references
    public string BrandName { get; set; }
    1 reference
    public int CategoryId { get; set; }
    [Display(Name = "Category")]
    3 references
    public string CategoryName { get; set; }

    [Display(Name = "Picture")]
    3 references
    public string Picture { get; set; }

    [Display(Name = "Quantity")]
    3 references
    public int Quantity { get; set; }

    [Display(Name = "Price")]
    3 references
    public decimal Price { get; set; }
    [Display(Name = "Discount")]
    3 references
    public decimal Discount { get; set; }
}
```

Ето как трябва да изглежда екшъна Index() в ProductController

```
// GET: ProductController
3 references
public ActionResult Index(string searchStringCategoryName, string searchStringBrandName)
{
    List<ProductIndexVM> products = _productService.GetProducts(searchStringCategoryName, searchStringBrandName)
    .Select(product => new ProductIndexVM
    {
        Id = product.Id,
        ProductName=product.ProductName,
        BrandId=product.BrandId,
        BrandName=product.Brand.BrandName,
        CategoryId=product.CategoryId,
        CategoryName=product.Category.CategoryName,
        Picture=product.Picture,
        Quantity=product.Quantity,
        Price=product.Price,
        Discount=product.Discount

    }).ToList();
   return this.View(products);

}
```

Създаваме вю в папка Views/Product със същото име като екшъна. Ако пробвате автоматично генериране на изглед, трябва да настроите полетата по следния начин:



Променяме вюто Index.cshtml

```
@model IEnumerable<WebShopApp.Models.Product.ProductIndexVM>

@{
    ViewData["Title"] = "Index";
}
<h1>Index</h1>
<p>
    <a asp-action="Create">Create New</a>
</p>
<form asp-controller="Product" asp-action="Index" method="get">
    <p>
        Brand: <input type="text" name="SearchStringBrandName" />
        Category: <input type="text" name="SearchStringCategoryName" />

        <input type="submit" value="Filter" />
    </p>
</form>
<table class= table >
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.ProductName)
            </th>
            @*<th>
                @Html.DisplayNameFor(model => model.BrandId)
            </th>*@
            <th>
                @Html.DisplayNameFor(model => model.BrandName)
            </th>
            @*<th>
                @Html.DisplayNameFor(model => model.CategoryId)
            </th>*@
            <th>
                @Html.DisplayNameFor(model => model.CategoryName)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Picture)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Quantity)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Price)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Discount)
            </th>
            <th></th>
        </tr>
    </thead>
```

```
    <tbody>
        @foreach (var item in Model)
        {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.ProductName)
                </td>
                @*<td>
                        @Html.DisplayFor(modelItem => item.BrandId)
                    </td>*@
                <td>
                    @Html.DisplayFor(modelItem => item.BrandName)
                </td>
                @*<td>
                        @Html.DisplayFor(modelItem => item.CategoryId)
                    </td>*@
                <td>
                    @Html.DisplayFor(modelItem => item.CategoryName)
                </td>
                <td>
                    <img src="@Html.DisplayFor(modelItem => item.Picture)" alt="Image" width="100" />
                    @*@Html.DisplayFor(modelItem => item.Picture)*@
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Quantity)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Price)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Discount) %
                </td>
                <td>
                    <a asp-action="Edit" asp-route-id="@item.Id">Edit</a> |
                    <a asp-action="Details" asp-route-id="@item.Id">Details</a> |
                    <a asp-action="Delete" asp-route-id="@item.Id">Delete</a>
                </td>
            </tr>
        }
    </tbody>
</table>
@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Може да копирате кода на изгледа Index от тук:

```
@model IEnumerable<WebShopApp.Models.Product.ProductIndexVM>

@{
    ViewData["Title"] = "Index";
}
<h1>Index</h1>

    <p>
        <a asp-action="Create">Create New</a>
    </p>

<form asp-controller="Product" asp-action="Index" method="get">
    <p>
        Brand: <input type="text" name="SearchStringBrandName" />
        Category: <input type="text" name="SearchStringCategoryName" />

        <input type="submit" value="Filter" />
    </p>
</form>

<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.ProductName)
            </th>
            @*<th>
            @Html.DisplayNameFor(model => model.BrandId)
```

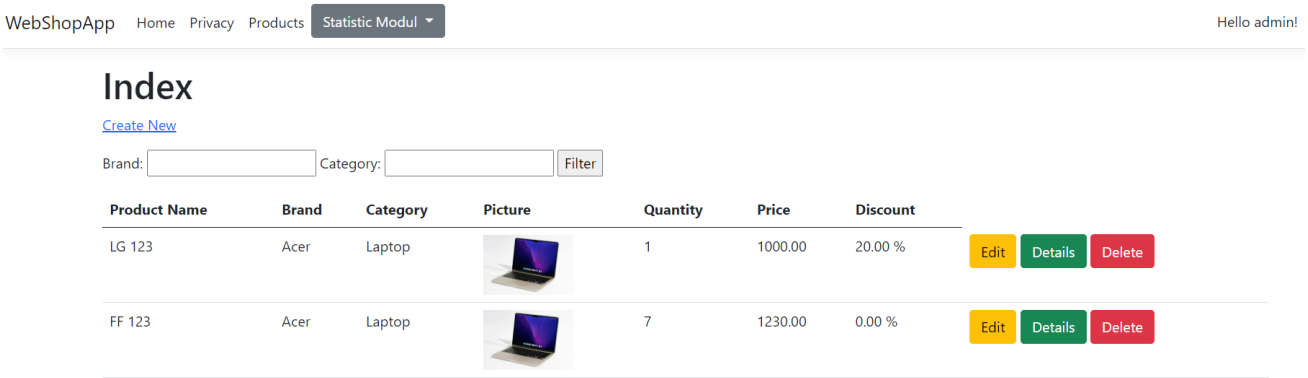
```
            </th>*@
            <th>
                @Html.DisplayNameFor(model => model.BrandName)
            </th>
            @*<th>
            @Html.DisplayNameFor(model => model.CategoryId)
            </th>*@
            <th>
                @Html.DisplayNameFor(model => model.CategoryName)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Picture)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Quantity)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Price)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Discount)
            </th>

        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model)
        {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.ProductName)
                </td>
                @*<td>
            @Html.DisplayFor(modelItem => item.BrandId)
            </td>*@
                <td>
                    @Html.DisplayFor(modelItem => item.BrandName)
                </td>
                @*<td>
            @Html.DisplayFor(modelItem => item.CategoryId)
            </td>*@
                <td>
                    @Html.DisplayFor(modelItem => item.CategoryName)
                </td>
                <td>
                    <img src="@Html.DisplayFor(modelItem => item.Picture)" alt="Image"
width="100" />
                    @*@Html.DisplayFor(modelItem => item.Pictute)*@
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Quantity)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Price)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Discount) %
                </td>

                <td>
                     <a asp-action="Edit" asp-route-id="@item.Id" class="btn btn-
warning">Edit</a>
```

```
                    <a asp-action="Details" asp-route-id="@item.Id" class="btn btn-
success">Details</a>

                    <a asp-action="Delete" asp-route-id="@item.Id" class="btn btn-
danger">Delete</a>

                </td>
            </tr>
        }
    </tbody>
</table>
@section Scripts {
    @{
        await Html.RenderPartialAsync("_ValidationScriptsPartial");
    }
}
```

В изгледа _Layuot.cshtml, който се намира в споделената папка Shared в папка Views, добавете в навигационната лента Products.

```
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
</li>
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Product" asp-action="Index">Products</a>
</li>
```

Тествайте приложението



По подобен начин създайте останалите екшъни и вюта към тях. Edit() трябва да прилича на Create с тази разлика, че в началото формата на вюто не е празна, а е заредена с данни по съответното Id.

Имплементация на Edit

```
public class ProductEditVM
{
    [Key]
    public int Id { get; set; }

    [Required]
    [MaxLength(30)]
    [Display(Name = "Product Name")]
    public string ProductName { get; set; } = null!;

    [Required]
    [Display(Name = "Brand")]
    public int BrandId { get; set; }
    public virtual List<BrandPairVM> Brands { get; set; } = new List<BrandPairVM>();

    [Required]
    [Display(Name = "Category")]
    public int CategoryId { get; set; }
    public virtual List<CategoryPairVM> Categories { get; set; } = new List<CategoryPairVM>();

    [Display(Name = "Picture")]
    public string Picture { get; set; } = null!;

    [Range(0, 5000)]
    [Display(Name = "Quantity")]
    public int Quantity { get; set; }

    [Display(Name = "Price")]
    public decimal Price { get; set; }

    [Display(Name = "Discount")]
    public decimal Discount { get; set; }
}
```

```
// GET: ProductController/Edit/5
public ActionResult Edit(int id)
{
    Product product = _productService.GetProductById(id);
    if (product == null)
    {
        return NotFound();
    }

    ProductEditVM updatedProduct = new ProductEditVM()
    {
        Id = product.Id,
        ProductName = product.ProductName,
        BrandId = product.BrandId,
        //BrandName = product.Brand.BrandName,
        CategoryId = product.CategoryId,
        // CategoryName = product.Category.CategoryName,
        Picture = product.Picture,
        Quantity = product.Quantity,
        Price = product.Price,
        Discount = product.Discount
    };
```

```
    updatedProduct.Brands = _brandService.GetBrands()
        .Select(b => new BrandPairVM()
        {
            Id = b.Id,
            Name = b.BrandName
        })
        .ToList();

    updatedProduct.Categories = _categoryService.GetCategories()
        .Select(c => new CategoryPairVM()
        {
            Id = c.Id,
            Name = c.CategoryName
        })
        .ToList();
    return View(updatedProduct);
}

 // POST: ProductController/Edit/5
 [HttpPost]
 [ValidateAntiForgeryToken]
 public ActionResult Edit(int id, ProductEditVM product)
 {
     {
         if (ModelState.IsValid)
         {
             var updated = _productService.Update(id, product.ProductName, product.BrandId,
                                                  product.CategoryId, product.Picture,
                                                  product.Quantity, product.Price, product.Discount);
             if (updated)
             {
                 return this.RedirectToAction("Index");
             }

         }
         return View(product);
     }
 }
```

Изглед Edit

```
@model WebShopApp.Models.Product.ProductEditVM

@{
    ViewData["Title"] = "Edit";
}

<h1>Edit</h1>

<h4>ProductEditVM</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Edit">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="Id" />
            <div class="form-group">
                <label asp-for="ProductName" class="control-label"></label>
                <input asp-for="ProductName" class="form-control" />
                <span asp-validation-for="ProductName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="BrandId" class="control-label"></label>
                @*<input asp-for="BrandId" class="form-control" />*@
```

```
                <select asp-for="BrandId" class="form-control text-center">
                    @foreach (var brand in Model.Brands)
                    {
                        <option value=" @brand.Id">@brand.Name</option>
                    }
                </select>
                <span asp-validation-for="BrandId" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="CategoryId" class="control-label"></label>
                @*<input asp-for="CategoryId" class="form-control" />*@
                <select asp-for="CategoryId" class="form-control text-center">
                    @foreach (var category in Model.Categories)
                    {
                        <option value="@category.Id">@category.Name</option>
                    }
                </select>
                <span asp-validation-for="CategoryId" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Picture" class="control-label"></label>
                <input asp-for="Picture" class="form-control" />
                <span asp-validation-for="Picture" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Quantity" class="control-label"></label>
                <input asp-for="Quantity" class="form-control" />
                <span asp-validation-for="Quantity" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Price" class="control-label"></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="Discount" class="control-label"></label>
                <input asp-for="Discount" class="form-control" />
                <span asp-validation-for="Discount" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Save" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>
@section Scripts {
    @{
        await Html.RenderPartialAsync("_ValidationScriptsPartial");
    }
}
```

Имплементация на Details

```
public class ProductDetailsVM
{
    [Key]
    public int Id { get; set; }

    [Display(Name = "Product Name")]
    public string ProductName { get; set; } = null!;

    public int BrandId { get; set; }
    [Display(Name = "Brand")]
    public string BrandName { get; set; } = null!;

    public int CategoryId { get; set; }
    [Display(Name = "Category")]
    public string CategoryName { get; set; } = null!;

    [Display(Name = "Picture")]
    public string Picture { get; set; } = null!;

    [Display(Name = "Quantity")]
    public int Quantity { get; set; }

    [Display(Name = "Price")]
    public decimal Price { get; set; }

    [Display(Name = "Discount")]
    public decimal Discount { get; set; }
}
```

```
// GET: ProductController/Details/5
public ActionResult Details(int id)
{
    Product item = _productService.GetProductById(id);
    if (item == null)
    {
        return NotFound();
    }
    ProductDetailsVM product = new ProductDetailsVM()
    {
        Id = item.Id,
        ProductName = item.ProductName,
        BrandId = item.BrandId,
        BrandName = item.Brand.BrandName,
        CategoryId = item.CategoryId,
        CategoryName = item.Category.CategoryName,
        Picture = item.Picture,
        Quantity = item.Quantity,
        Price = item.Price,
        Discount = item.Discount
    };
    return View(product);
}
```

Изглед Details

```
@model WebShopApp.Models.Product.ProductDetailsVM

@{
    ViewData["Title"] = "Details";
}

<h1>Details</h1>

<div>
    <h4>ProductDetailsVM</h4>
    <hr />
    <dl class="row">
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.ProductName)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.ProductName)
        </dd>

        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.BrandName)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.BrandName)
        </dd>

        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.CategoryName)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.CategoryName)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Picture)
        </dt>
        <dd class="col-sm-10">
            <img src="@Html.DisplayFor(model => model.Picture)" alt="Image" width="100"
/>
            @*@Html.DisplayFor(model => model.Picture)*@
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Quantity)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Quantity)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Price)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Price)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Discount)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Discount)
        </dd>
    </dl>
</div>
```

```
<div>
    <a asp-action="Edit" asp-route-id="@Model.Id">Edit</a> |
    <a asp-action="Index">Back to List</a>
</div>
@section Scripts {
    @{
        await Html.RenderPartialAsync("_ValidationScriptsPartial");
    }
}
```

```
@model WebShopApp.Models.Product.ProductDetailsVM

@{
    ViewData["Title"] = "Details";
}

<h1>Details</h1>

<div>
    <h4>ProductDetailsVM</h4>
    <hr />
    <dl class="row">
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.ProductName)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.ProductName)
        </dd>
        @*<dt class = "col-sm-2">
                @Html.DisplayNameFor(model => model.BrandId)
            </dt>
            <dd class = "col-sm-10">
                @Html.DisplayFor(model => model.BrandId)
            </dd>*@
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.BrandName)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.BrandName)
        </dd>
        @*<dt class = "col-sm-2">
                @Html.DisplayNameFor(model => model.CategoryId)
            </dt>
            <dd class = "col-sm-10">
                @Html.DisplayFor(model => model.CategoryId)
            </dd>*@
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.CategoryName)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.CategoryName)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Picture)
        </dt>
        <dd class="col-sm-10">
            <img src="@Html.DisplayFor(model => model.Picture)" alt="Image" width="100" />
            @*@Html.DisplayFor(model => model.Picture)*@
        </dd>
```

```
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Quantity)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Quantity)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Price)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Price)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Discount)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Discount)
        </dd>
    </dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model.Id">Edit</a> |
    <a asp-action="Index">Back to List</a>
</div>
@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Имплементация на Delete

Вю модел ProductDeleteVM

```
public class ProductDeleteVM
{

    [Key]
    public int Id { get; set; }

    [Display(Name = "Product Name")]
    public string ProductName { get; set; } = null!;

    public int BrandId { get; set; }
    [Display(Name = "Brand")]
    public string BrandName { get; set; } = null!;
    public int CategoryId { get; set; }
    [Display(Name = "Category")]
    public string CategoryName { get; set; } = null!;
    [Display(Name = "Picture")]
    public string Picture { get; set; } = null!;
    [Display(Name = "Quantity")]
    public int Quantity { get; set; }
    [Display(Name = "Price")]
    public decimal Price { get; set; }
    [Display(Name = "Discount")]
    public decimal Discount { get; set; }
}
```

Екшъни в ProductController

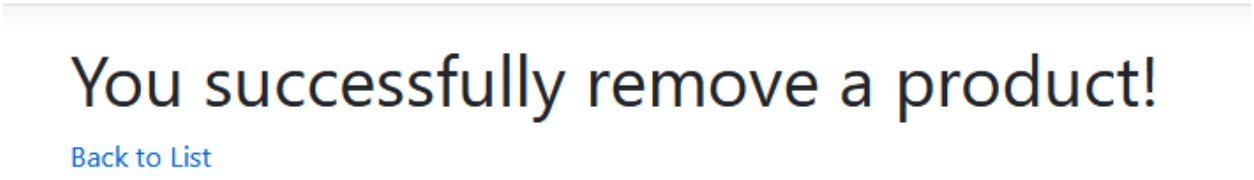
```
// GET: ProductController/Delete/5
public ActionResult Delete(int id)
{
    Product item = _productService.GetProductById(id);
    if (item == null)
    {
        return NotFound();
    }
    ProductDeleteVM product = new ProductDeleteVM()
    {
        Id = item.Id,
        ProductName = item.ProductName,
        BrandId = item.BrandId,
        BrandName = item.Brand.BrandName,
        CategoryId = item.CategoryId,
        CategoryName = item.Category.CategoryName,
        Picture = item.Picture,
        Quantity = item.Quantity,
        Price = item.Price,
        Discount = item.Discount
    };
    return View(product);
}

  // POST: ProductController/Delete/5
  [HttpPost]
  [ValidateAntiForgeryToken]
    0 references
    public ActionResult Delete(int id, IFormCollection collection)


  {
      var deleted = _productService.RemoveById(id);

      if (deleted)
      {
          return this.RedirectToAction("Success");
      }
      else
      {
          return View();
      }
  }

  0 references
  public IActionResult Success()
  {

      return View();
  }
```

Изглед Delete

```
@model WebShopApp.Models.Product.ProductDeleteVM

@{
    ViewData["Title"] = "Delete";
}

<h1>Delete</h1>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>ProductDeleteVM</h4>
    <hr />
    <dl class="row">
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.ProductName)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.ProductName)
        </dd>

        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.BrandName)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.BrandName)
        </dd>

        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.CategoryName)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.CategoryName)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Picture)
        </dt>
        <dd class="col-sm-10">
            <img src="@Html.DisplayFor(model => model.Picture)" alt="Image" width="100"
/>
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Quantity)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Quantity)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Price)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Price)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Discount)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Discount)
        </dd>
    </dl>
```

```
    <form asp-action="Delete">
        <input type="hidden" asp-for="Id" />
        <input type="submit" value="Delete" class="btn btn-danger" /> |
        <a asp-action="Index">Back to List</a>
    </form>
</div>
@section Scripts {
    @{
        await Html.RenderPartialAsync("_ValidationScriptsPartial");
    }
}
```

При успешно изтриване на продукт се появява вю с надпис: **You successfully remove a product!**



Изглед Success

Сега трябва да направим, така че само администраторът на сайта да може да добавя, редактира и изтрива продукти.

Гостите и регистрираните потребители могат само да разглеждат и виждат детайлите за продуктите.

Първо слагаме на целия контролер атрибут `[Authorize(Roles = "Administrator")]`

Така всички екшъни вече са достъпни само за администратора.

Сега трябва да позволим на гостите и регистрираните потребители да имат достъп до екшъните Index() и Details()

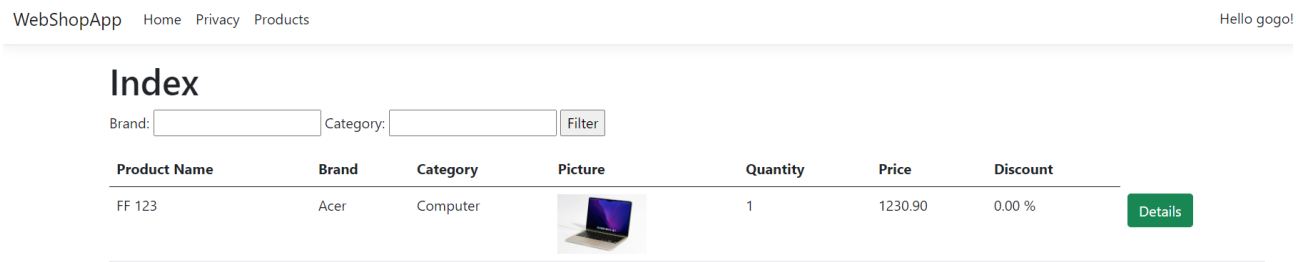
Над всеки от двата екшъна поставяме атрибут [AllowAnonymous]

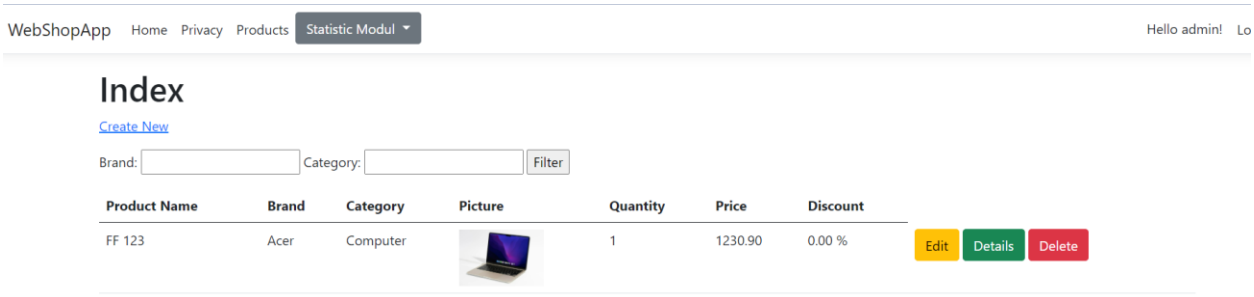
```
    // GET: ProductController
    [AllowAnonymous]
    1 reference
    public ActionResult Index(string searchStringCategoryName, string searchStringBrandName)
    {
}

[AllowAnonymous]
// GET: ProductController/Details/5
0 references
public ActionResult Details(int id)
{
```

Трябва да съобразим, че във вютата на тези екшъни има препратки към Create, Edit и Delete. Затова променяме Index.cshtml

```
@if ((this.User.Identity.IsAuthenticated) && (this.User.IsInRole("Administrator")))
{
    <p>
        <a asp-action="Create">Create New</a>
    </p>
}
    <form asp-controller="Product" asp-action="Index" method="get">
</td>
<td>
    @if ((this.User.Identity.IsAuthenticated) && (this.User.IsInRole("Administrator")))
    {
        <a asp-action="Edit" asp-route-id="@item.Id">Edit</a>
    } |
    <a asp-action="Details" asp-route-id="@item.Id">Details</a> |
    @if ((this.User.Identity.IsAuthenticated) && (this.User.IsInRole("Administrator")))
    {
        <a asp-action="Delete" asp-route-id="@item.Id">Delete</a>
    }
</td>
```

Подобни действия трябва да предприемем за всички **контролери, екшъни и менюта** като се съобразим с изискванията за сигурност и правата, които нашето приложение предоставя на своите потребители.





Така, ако не сме се логнали като администратор, дори и да напишем правилния раутинг – няма да имаме достъп до съответния екшън и вю.