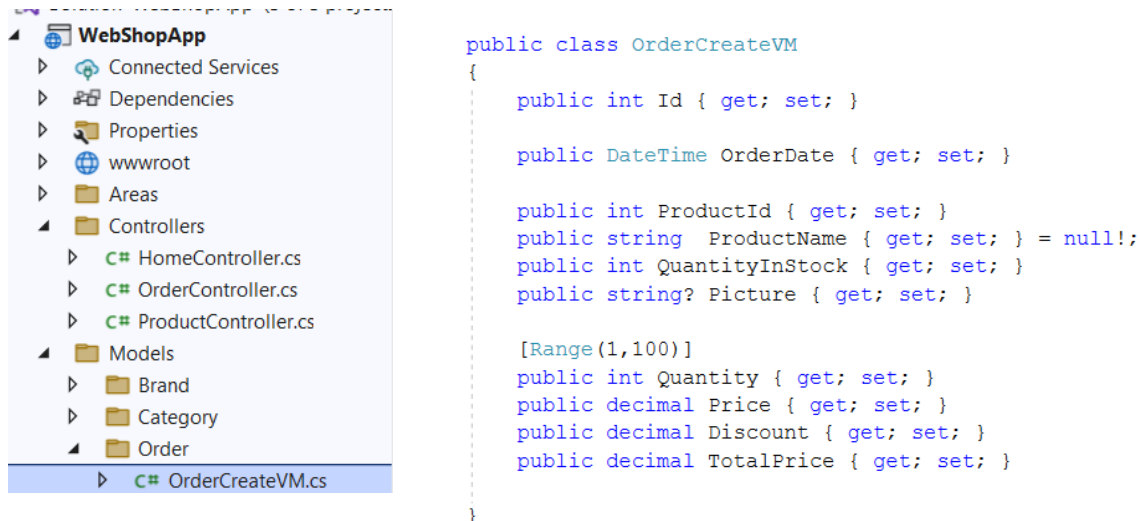


## WebShopApp - Част 5 - Имплементация на поръчки

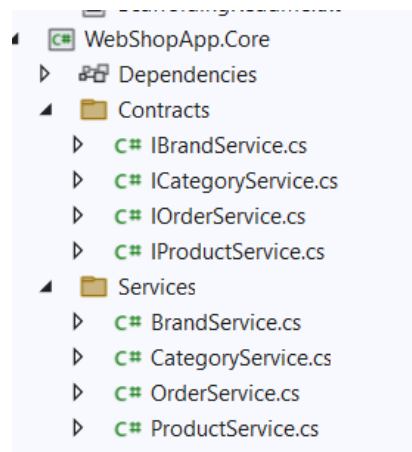
Правим нова папка в проекта WebShopApp в папка Models с име Order.

В нея ще създадем вю модела OrderCreateVM.



Тъй като през този модел ще се въвежда единствено количеството на поръчаните продукти, само там е зададено ограничение с атрибут. Приемаме, че не може да се поръчат онлайн повече от 100 броя от продукта.

В папката Contracts създаваме нов интерфейс IOrderService, а в папката Services – нов сървис OrderService. В него ще инжектираме ApplicationDbContext и ProductService.



```
public interface IOrderService
{
    bool Create(int productId, string userId, int quantity);

    List<Order> GetOrders();
}
```

```

public class OrderService : IOrderService
{
    private readonly ApplicationDbContext _context;
    private readonly IProductService _productService;

    public OrderService(ApplicationDbContext context, IProductService productService)
    {
        _context = context;
        _productService = productService;
    }
}

```

Имплементацията на метода Create() е следната:

```

public bool Create(int productId, string userId, int quantity)
{
    //намираме продукта по неговото id
    var product = this._context.Products.SingleOrDefault(x => x.Id == productId);

    //проверяваме дали има такъв продукт
    if (product == null)
    {
        return false;
    }

    //създаване на поръчка
    Order item = new Order
    {
        OrderDate = DateTime.Now,
        ProductId = productId,
        UserId = userId,
        Quantity = quantity,
        Price = product.Price,
        Discount = product.Discount
    };

    //намаляване на количеството на продукта
    product.Quantity -= quantity;

    //отразяване на промените в колекциите
    this._context.Products.Update(product);
    this._context.Orders.Add(item);

    //запис на промените в БД
    return _context.SaveChanges() != 0;
}

```

За да се използва успешно новия сървис, трябва да го добавим след останалите в Program.cs.

```
.AddDbContext<ApplicationDbContext>();  
builder.Services.AddControllersWithViews();  
  
builder.Services.AddTransient<ICategoryService, CategoryService>();  
builder.Services.AddTransient<IBrandService, BrandService>();  
builder.Services.AddTransient<IProductService, ProductService>();  
builder.Services.AddTransient<IOrderService, OrderService>();
```

В папка Controllers създаваме нов контролер с име **OrderController**. На първо време му сложете атрибут **[Authorize]**, за да бъде достъпен само за логнат потребител. Инжектирайте двата сървиса, необходими за имплементацията на екшъните.

```
[Authorize]  
public class OrderController : Controller  
{  
    private readonly IProductService _productService;  
    private readonly IOrderService _orderService;  
  
    public OrderController(IProductService productService, IOrderService orderService)  
    {  
        _productService = productService;  
        _orderService = orderService;  
    }  
}
```

Да имплементираме екшъна Create() по GET и по POST. Обърнете внимание на начина за достъпване на текущия потребител.

```
// GET: OrderController/Create  
public ActionResult Create(int id)  
{  
    Product product = _productService.GetProductById(id);  
    if (product == null)  
    {  
        return NotFound();  
    }  
    //Ако има продукт с това id, го зареждаме във формата за поръчка  
    OrderCreateVM order = new OrderCreateVM()  
    {  
        ProductId = product.Id,  
        ProductName = product.ProductName,  
        QuantityInStock = product.Quantity,  
        Price = product.Price,  
        Discount = product.Discount,  
        Picture = product.Picture,  
    };  
    return View(order);  
}
```

```
// POST: OrderController/Create
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create(OrderCreateVM bindingModel)
{
    string currentUserId = this.User.FindFirstValue(ClaimTypes.NameIdentifier);

    var product = this._productService.GetProductById(bindingModel.ProductId);
    if (currentUserId == null || product == null || product.Quantity < bindingModel.Quantity ||
        product.Quantity == 0)
    {
        // ако потребителят не съществува или продуктът не съществува или няма достатъчно наличност
        return RedirectToAction("Denied", "Order");
    }
    if (ModelState.IsValid)
    {
        _orderService.Create(bindingModel.ProductId, currentUserId, bindingModel.Quantity);
    }
    //при успешна поръчка се връща в списъка на продуктите
    return this.RedirectToAction("Index", "Product");
}
```

Добавете в контролера и екшъна Denied(), който просто ще връща съответното вю.

```
// GET: OrderController/Denied
public ActionResult Denied()
{
    return View();
}
```

Да направим двата изгледа Create.cshtml и Denied.cshtml, като първо създадем папка Order в папка Views.

```
Create.cshtml*  ProductService.cs  OrderCreateVM.cs*  Product.cs  Program.cs  IOrderService.cs*  IBrandService.cs  OrderService.cs
1  @model WebShopApp.Models.Order.OrderCreateVM
2  @{
3      ViewData["Title"] = "Create";
4  }
5
6  <h1>Order</h1>
7
8  <hr />
9  <div class="row">
10     <div class="col-md-4">
11         <form asp-action="Create">
12             <div asp-validation-summary="ModelOnly" class="text-danger"></div>
13             <div class="form-group">
14                 <input type="hidden" asp-for="ProductId" class="form-control" readonly="readonly" />
15             </div>
16             <div class="form-group">
17                 <label asp-for="ProductName" class="control-label"></label>
18                 <input asp-for="ProductName" class="form-control" readonly="readonly" />
19             </div>
20             <div class="form-group">
21                 <label asp-for="Picture" class="control-label"></label>
22                 <input type="hidden" asp-for="Picture" class="form-control" readonly="readonly" />
23                 <div></div>
24             </div>
25             <div class="form-group">
26                 <label asp-for="Price" class="control-label"></label>
27                 <input asp-for="Price" class="form-control" readonly="readonly" />
28             </div>
29             <div class="form-group">
30                 <label asp-for="Discount" class="control-label"></label>
31                 <input asp-for="Discount" class="form-control" readonly="readonly" />
32             </div>

```

```

33 <div class="form-group">
34 <label asp-for="Quantity" class="control-label"></label>
35 <input asp-for="Quantity" class="form-control" onclick="calcSum()" />
36 </div>
37 <div class="form-group">
38 <label asp-for="TotalPrice" class="control-label"></label>
39 <input asp-for="TotalPrice" class="form-control" readonly="readonly" />
40 </div>
41 <br/>
42 <div class="form-group">
43 <input type="submit" value="Order" class="btn btn-primary" />
44 <a asp-action="Index" asp-controller="Product" class="btn btn-warning">Back to Products</a>
45 </div>
46 </form>
47 </div>
48 </div>
49
50 <script>
51 function calcSum() {
52 let q = document.getElementsByName('Quantity')[0].value;
53 let p = document.getElementsByName('Price')[0].value;
54 let d = document.getElementsByName('Discount')[0].value;
55 let sum = Number(q) * (Number(p) - Number(p) * Number(d) / 100);
56 document.getElementsByName('TotalPrice')[0].value = sum.toFixed(2);
57 }
58 document.getElementById('Quantity').max = @Model.QuantityInStock;
59 document.getElementById('Quantity').min = 1;
60 </script>
61
62 @section Scripts {
63 @{}
64 await Html.RenderPartialAsync("_ValidationScriptsPartial");
65 }
66 }

```

С атрибута `type="hidden"` някои полета са скрити, с атрибута `readonly="readonly"` не е позволено редактирането им, но се виждат във формата. Например задължително е да се предаде стойността за полето `ProductId`, но не е необходимо потребителят да я вижда.

За единственото достъпно за промяна поле `Quantity` чрез атрибута `onclick="calcSum()"` е закачена функция за изчисляване на дължимата сума. В скрипта са направени допълнително ограничения за въвежданото количество – то не може да е по-малко от 1 и да надвишава наличното количество за продукта.

```

@model WebShopApp.Models.Order.OrderCreateVM
@{
    ViewData["Title"] = "Create";
}

<h1>Order</h1>

<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">
            <input type="hidden" asp-for="ProductId" class="form-control"
readonly="readonly" />
        </div>
        <div class="form-group">
            <label asp-for="ProductName" class="control-label"></label>
            <input asp-for="ProductName" class="form-control" readonly="readonly"
/>
        </div>
        <div class="form-group">
            <label asp-for="Picture" class="control-label"></label>
            <input type="hidden" asp-for="Picture" class="form-control"
readonly="readonly" />
            <div></div>
        </div>
        <div class="form-group">
            <label asp-for="Price" class="control-label"></label>
            <input asp-for="Price" class="form-control" readonly="readonly" />
        </div>
        <div class="form-group">
            <label asp-for="Discount" class="control-label"></label>
            <input asp-for="Discount" class="form-control" readonly="readonly" />
        </div>
        <div class="form-group">
            <label asp-for="Quantity" class="control-label"></label>
            <input asp-for="Quantity" class="form-control" onclick="calcSum()" />
            <span asp-validation-for="Quantity" class="text-danger"></span>
        </div>
        <div class="form-group">
            <label asp-for="TotalPrice" class="control-label"></label>
            <input asp-for="TotalPrice" class="form-control" readonly="readonly" />
        </div>
        <br/>
        <div class="form-group">
            <input type="submit" value="Order" class="btn btn-primary" />
            <a asp-action="Index" asp-controller="Product" class="btn btn-
warning">Back to Products</a>
        </div>
    </form>
</div>
</div>

<script>
    function calcSum() {
        let q = document.getElementsByName('Quantity')[0].value;
        let p = document.getElementsByName('Price')[0].value;
        let d = document.getElementsByName('Discount')[0].value;
        let sum = Number(q) * (Number(p) - Number(p) * Number(d) / 100);
        document.getElementsByName('TotalPrice')[0].value = sum.toFixed(2)
    }
    document.getElementById('Quantity').max = @Model.QuantityInStock;
    document.getElementById('Quantity').min = 1;
</script>

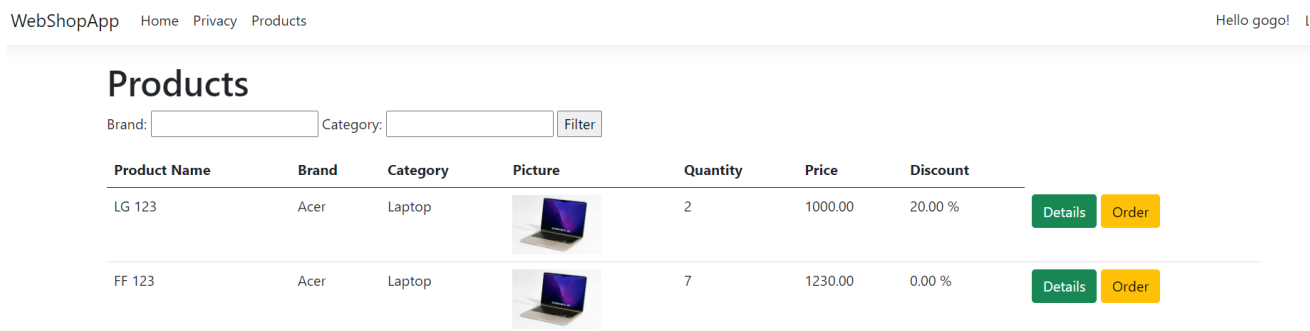
@section Scripts {
    @{
        await Html.RenderPartialAsync("_ValidationScriptsPartial");
    }
}

```

Създайте и изгледа Denied

```
Denied.cshtml  Create.cshtml  Create.cshtml*  ProductService.cs  OrderCreateVM.cs  Program.cs
1  @{
2      ViewData["Title"] = "Denied";
3  }
4
5  <h1>Insufficient availability</h1>
6
7  <div>
8      <a asp-controller="Product" asp-action="Index">Back to list of products</a>
9  </div>
10
```

Остана да променим вюто за преглед на продуктите, като добавим бутон за поръчка.



Направете и малко реорганизиране на бутоните: Details е достъпан за всички посетители на сайта, Order - за логнатите, Edit и Delete – само за администратора.

```
<td>
    @Html.DisplayFor(modelItem => item.Discount) %
</td>

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

    @if (this.User.Identity.IsAuthenticated)
    {
        <a asp-action="Create" asp-controller="Order" asp-route-id="@item.Id" class="btn btn-warning">Order</a>
    }

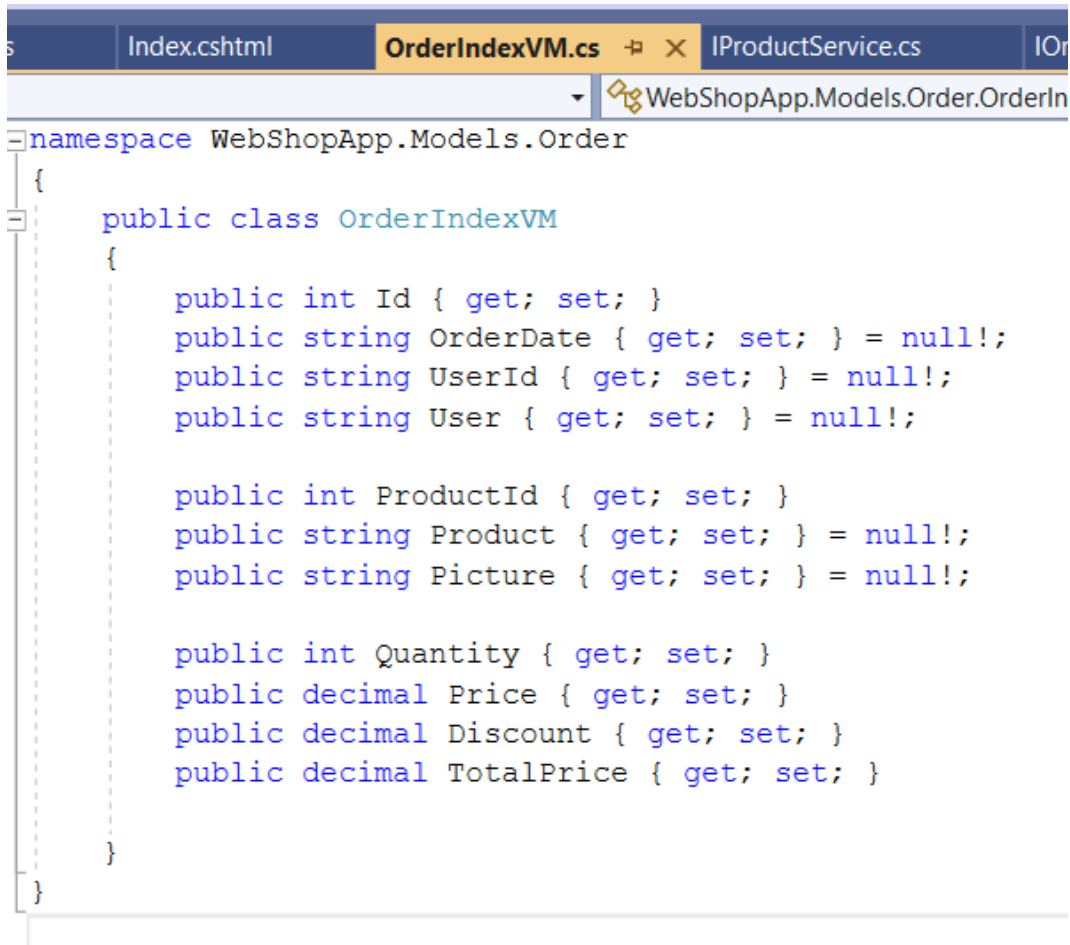
    @if ((this.User.Identity.IsAuthenticated) && (this.User.IsInRole("Administrator")))
    {
        <a asp-action="Edit" asp-route-id="@item.Id" class="btn btn-warning">Edit</a>
        <a asp-action="Delete" asp-route-id="@item.Id" class="btn btn-danger">Delete</a>
    }
</td>
```

Стартирайте приложението и проверете създаването на поръчка.

Пробвайте работа като администратор, гост и клиент. Проверете дали се намалява количеството на продуктите. Пробвайте да въведете по-голямо количество или да направите поръчка при липса на количество. За първото имате двойна защита – през скрипта и през сървиса.

Да направим възможно администраторът да вижда всички поръчки.

Създаваме вю модел



```
namespace WebShopApp.Models.Order
{
    public class OrderIndexVM
    {
        public int Id { get; set; }
        public string OrderDate { get; set; } = null!;
        public string UserId { get; set; } = null!;
        public string User { get; set; } = null!;

        public int ProductId { get; set; }
        public string Product { get; set; } = null!;
        public string Picture { get; set; } = null!;

        public int Quantity { get; set; }
        public decimal Price { get; set; }
        public decimal Discount { get; set; }
        public decimal TotalPrice { get; set; }
    }
}
```

Добавяме в интерфейса IOrderService нови методи, имплементираме методите в сървиса .

```
public interface IOrderService
{
    bool Create(int productId, string userId, int quantity);

    List<Order> GetOrders();

    List<Order> GetOrdersByUser(string userId);

    Order GetOrderById(int orderId);

    bool RemoveById(int orderId);

    bool Update(int orderId, int productId, string userId, int quantity);
}
```



Имплементация на List<Order> GetOrders() в OrderService.cs

```
public List<Order> GetOrders()
{
    return _context.Orders.OrderByDescending(x=>x.OrderDate).ToList();
}
```

Имплементираме екшъна Index() в контролера. Екшънът да бъде достъпен само за администратора.

```
// GET: OrderController
[Authorize(Roles = "Administrator")]
public ActionResult Index()
{
    // string userId = this.User.FindFirstValue(ClaimTypes.NameIdentifier);
    // var user = context.Users.SingleOrDefault(u => u.Id == userId);

    List<OrderIndexVM> orders = _orderService.GetOrders()
        .Select(x => new OrderIndexVM
        {
            Id = x.Id,
            OrderDate = x.OrderDate.ToString("dd-MMM-yyyy hh:mm", CultureInfo.InvariantCulture),
            UserId = x.UserId,
            User = x.User.UserName,
            ProductId = x.ProductId,
            Product = x.Product.ProductName,
            Picture = x.Product.Picture,
            Quantity = x.Quantity,
            Price = x.Price,
            Discount = x.Discount,
            TotalPrice = x.TotalPrice,
        }).ToList();
    return View(orders);
}
```

Да създадем изгледа Index.cshtml в папка View->Order

```
@model IEnumerable<WebShopApp.Models.Order.OrderIndexVM>

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

<h1>All Orders</h1>

<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.OrderDate)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.User)
            </th>
            <th>
```

```

        @Html.DisplayNameFor(model => model.Product)
    </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>
        @Html.DisplayNameFor(model => model.TotalPrice)
    </th>
    <th></th>
</tr>
</thead>
<tbody>
    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.OrderDate)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.User)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Product)
            </td>
            <td>
                
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Quantity)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Price)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Discount)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.TotalPrice)
            </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>*@
            </td>
        </tr>
    }
</tbody>
</table>

```

Да добавим в навигационната лента за администратора линк за разглеждане на поръчките /\_Layout.cshtml/. Поръчките ще се разглеждат от статистическия модул.

```
}
@if ((this.User.Identity.IsAuthenticated) && (this.User.IsInRole("Administrator")))
{
    <li class="dropdown">
        <button class="btn btn-secondary dropdown-toggle" type="button" id="dropdownMenuButton1"
            data-bs-toggle="dropdown" aria-expanded="false">
            Statistic Modul
        </button>

        <ul class="dropdown-menu" aria-labelledby="dropdownMenuButton1">
            <li><a class="dropdown-item" href="/Statistics/Clients">Clients</a></li>
            <li><a class="dropdown-item" href="/Order/Index">Orders</a></li>

        </ul>
    </li>
}
```

Сега да имплементираме нова функционалност – преглед на собствените поръчки. За това ще ни послужи метод `GetOrdersByUser()`, който трябва да имплементирате по следния начин:

```
public List<Order> GetOrdersByUser(string userId)
{
    return _context.Orders.Where(x => x.UserId == userId)
        .OrderByDescending(x => x.OrderDate)
        .ToList();
}
```

В контролера ще създадем нов екшън `MyOrders()`

```
public ActionResult MyOrders()
{
    string currentUserId = this.User.FindFirstValue(ClaimTypes.NameIdentifier);
    // var user = context.Users.SingleOrDefault(u => u.Id == userId);

    List<OrderIndexVM> orders = _orderService.GetOrdersByUser(currentUserId)
        .Select(x => new OrderIndexVM
        {
            Id = x.Id,
            OrderDate = x.OrderDate.ToString("dd-MMM-yyyy hh:mm", CultureInfo.InvariantCulture),
            UserId = x.UserId,
            User = x.User.UserName,
            ProductId = x.ProductId,
            Product = x.Product.ProductName,
            Picture = x.Product.Picture,
            Quantity = x.Quantity,
            Price = x.Price,
            Discount = x.Discount,
            TotalPrice = x.TotalPrice,
        }).ToList();
    return View(orders);
}
```

Ще използваме същия вю модел OrderIndexVM и ще създадем изгледа MyOrders.cshtml. Няма да има нужда да се вижда потребителското име, но е добре в началото да има бутон за добавяне на нова поръчка.

```
@model IEnumerable<WebShopApp.Models.Order.OrderIndexVM>
```

```
@{  
    ViewData["Title"] = "Index";  
}
```

```
<h1>My Orders</h1>
```

```
<table class="table">  
    <thead>  
        <tr>  
            <th>  
                @Html.DisplayNameFor(model => model.OrderDate)  
            </th>  
            <th>  
                @Html.DisplayNameFor(model => model.Product)  
            </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>  
                @Html.DisplayNameFor(model => model.TotalPrice)  
            </th>  
        <th></th>  
    </tr>  
    </thead>
```

```

<tbody>
  @foreach (var item in Model)
  {
    <tr>
      <td>
        @Html.DisplayFor(modelItem => item.OrderDate)
      </td>
      <td>
        @Html.DisplayFor(modelItem => item.Product)
      </td>
      <td>
        
      </td>
      <td>
        @Html.DisplayFor(modelItem => item.Quantity)
      </td>
      <td>
        @Html.DisplayFor(modelItem => item.Price)
      </td>
      <td>
        @Html.DisplayFor(modelItem => item.Discount)
      </td>
      <td>
        @Html.DisplayFor(modelItem => item.TotalPrice)
      </td>
    </tr>
  }
</tbody>
</table>

```

Накрая да добавим в навигационната лента MyOrders след Products.

```

<li class="nav-item">
  <a class="nav-link text-dark" asp-area="" asp-controller="Product" asp-action="Index">Products</a>
</li>
@if (this.User.Identity.IsAuthenticated)
{
  <li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Order" asp-action="MyOrders">My Orders</a>
  </li>
}
@if ((this.User.Identity.IsAuthenticated) && (this.User.IsInRole("Administrator")))
{
  <li class="nav-item">

```

Клиентът няма да има възможност да редактира или изтрива поръчка, но може да позволи на администраторът да има възможност да редактира или изтрива поръчки.