

НП "Обучение за ИТ кариера"

Сайт за музикален магазин

Антон Христов, Хасан Хасан, Реджеб Реджеб

Група 08

Гр.Хасково – 2025г.

1.Цели на проекта:

- Създаване на уебсайт, чрез който клиентите да закупуват музикални инструменти от ToniStore

2.Разпределение на ролите:




















- На всеки участник в проекта му бе възложено да направи по 2 модела и по 1 контролер

3.Основни етапи в разработването на проекта:

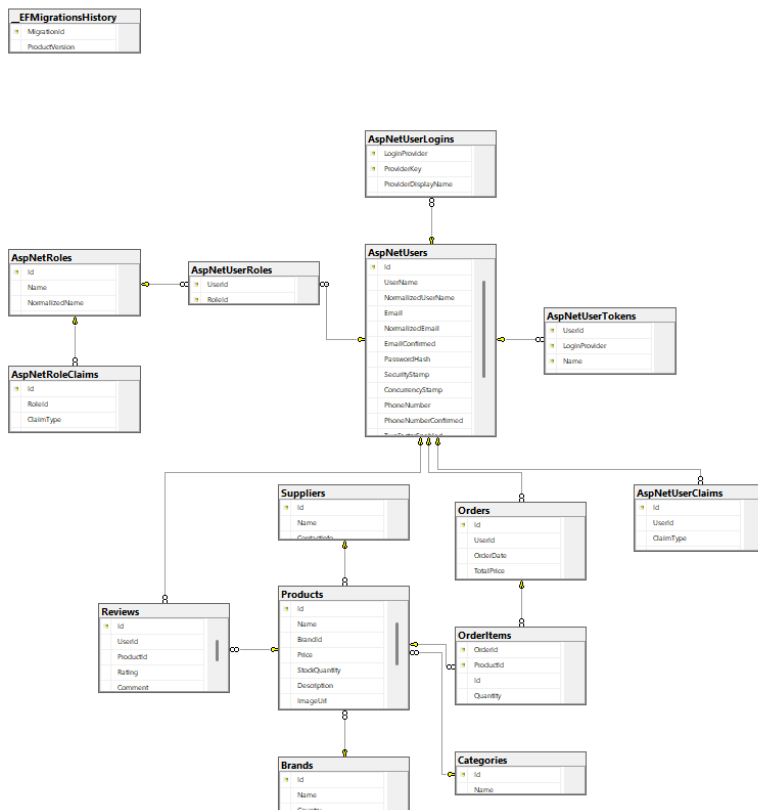
- Използване на Code First за изграждане на базата данни
- Създаване на модели
- Създаване на контролери
- Създаване на Views(изгледи)
- Модифициране на StartUp за създаване на роли за участници и администратори (Member и Admin)

4.Реализация:

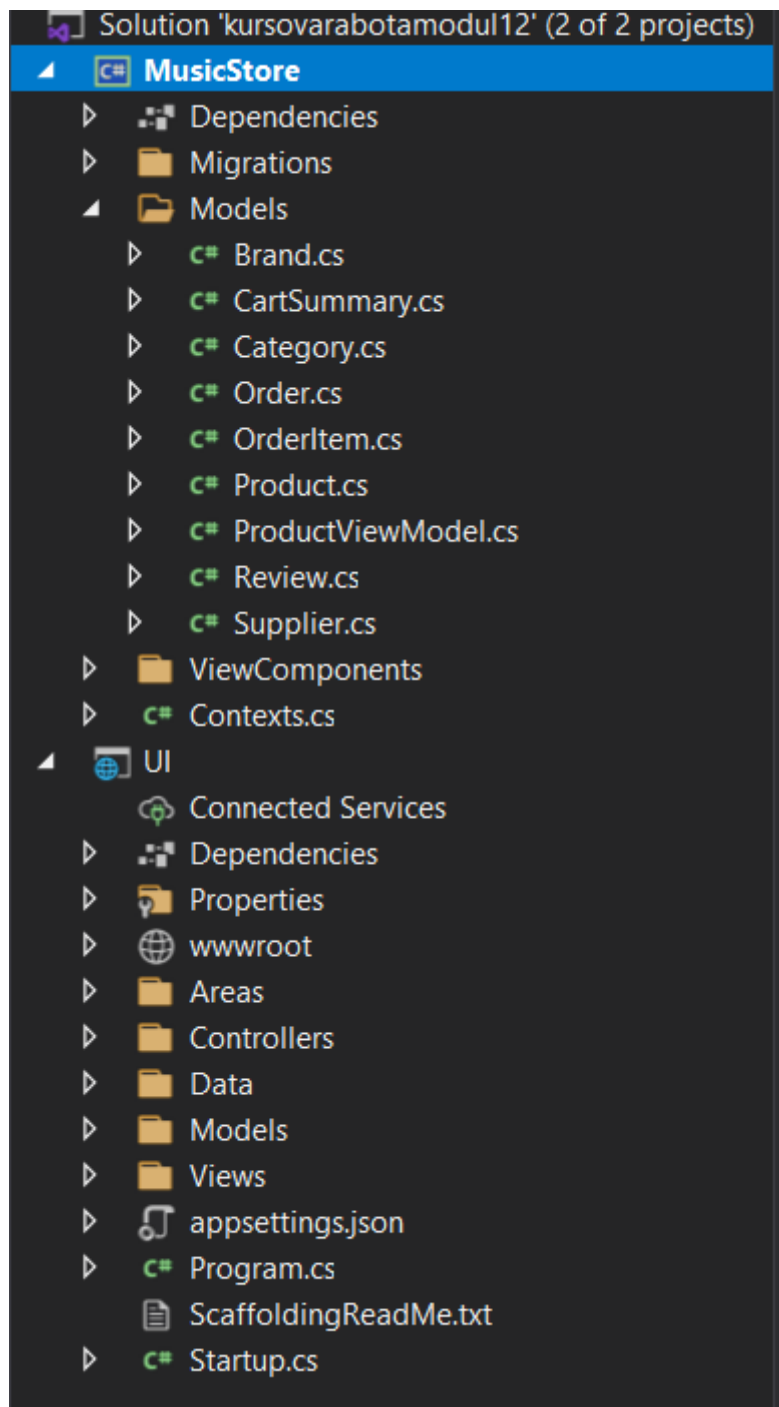
- а) За реализирането на проекта са използвани следните nugget пакети:

	Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore by Microsoft	 5.0.17
	ASP.NET Core middleware for Entity Framework Core error pages. Use this middleware to detect and diagnose errors with Entity Framework Core migra...	9.0.2
	Microsoft.AspNetCore.Identity.EntityFrameworkCore by Microsoft	 5.0.17
	ASP.NET Core Identity provider that uses Entity Framework Core.	9.0.2
	Microsoft.AspNetCore.Identity.UI by Microsoft	 5.0.17
	ASP.NET Core Identity UI is the default Razor Pages built-in UI for the ASP.NET Core Identity framework.	9.0.2
	Microsoft.EntityFrameworkCore by Microsoft	 5.0.17
	Entity Framework Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations. EF...	9.0.2
	Microsoft.EntityFrameworkCore.Design by Microsoft	 5.0.17
	Shared design-time components for Entity Framework Core tools.	9.0.2
	Microsoft.EntityFrameworkCore.Proxies by Microsoft	 5.0.17
	Lazy-loading proxies for Entity Framework Core.	9.0.2
	Microsoft.EntityFrameworkCore.Sqlite by Microsoft	 5.0.17
	SQLite database provider for Entity Framework Core.	9.0.2
	Microsoft.EntityFrameworkCore.SqlServer by Microsoft	 5.0.17
	Microsoft SQL Server database provider for Entity Framework Core.	9.0.2
	Microsoft.EntityFrameworkCore.Tools by Microsoft	 5.0.17
	Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.	9.0.2
	Microsoft.VisualStudio.Web.CodeGeneration.Design by Microsoft	5.0.2
	Code Generation tool for ASP.NET Core. Contains the dotnet-aspnet-codegenerator command used for generating controllers and views.	9.0.0

б) Диаграма на проекта:



в) Снимки от програмата:



```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace MusicStore.Models
8  {
9      public class Product
10     {
11         public virtual int Id { get; set; }
12         public virtual string Name { get; set; }
13         public virtual int BrandId { get; set; }
14         public virtual decimal Price { get; set; }
15         public virtual int StockQuantity { get; set; }
16         public virtual string Description { get; set; }
17         public virtual string ImageUrl { get; set; }
18
19
20         public virtual Brand Brand { get; set; }
21         public virtual int CategoryId { get; set; }
22         public virtual Category Category { get; set; }
23         public virtual int SupplierId { get; set; }
24         public virtual Supplier Supplier { get; set; }
25
26         public virtual ICollection<OrderItem> OrderItems { get; set; } = new List<OrderItem>();
27         public virtual ICollection<Review> Reviews { get; set; } = new List<Review>();
28
29
30         public Product() { }
31     }
32 }
33

```

```

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using MusicStore.Models;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;

namespace MusicStore.Controllers
{
    public class ProductsController : Controller
    {
        private readonly Contexts _context;

        public ProductsController(Contexts context)
        {
            _context = context;
        }

        // GET: Products
        [AllowAnonymous]
        public async Task<IActionResult> Index(string searchString, int? categoryId)
        {
            var query = _context.Products
                .Include(p => p.Brand)
                .Include(p => p.Category)
                .Include(p => p.Supplier)
                .AsQueryable();

            if (!string.IsNullOrEmpty(searchString))
            {
                query = query.Where(p => p.Name.Contains(searchString) ||
                    p.Description.Contains(searchString));
            }

            if (categoryId.HasValue)
            {
                query = query.Where(p => p.CategoryId == categoryId);
            }

            ViewData["Categories"] = new SelectList(_context.Categories, "Id", "Name");
            return View(await query.ToListAsync());
        }

        // GET: Products/Details/5
        [AllowAnonymous]
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null) return NotFound();

            var product = await _context.Products
                .Include(p => p.Brand)
                .Include(p => p.Category)
                .Include(p => p.Supplier)
                .Include(p => p.Reviews)
                .ThenInclude(r => r.User)
                .SingleOrDefaultAsync(p => p.Id == id);
            if (product == null) return NotFound();

            return View(product);
        }
    }
}

```

```

// GET: Products/Details/5
[AllowAnonymous]
public async Task<IActionResult> Details(int? id)
{
    if (id == null) return NotFound();

    var product = await _context.Products
        .Include(p => p.Brand)
        .Include(p => p.Category)
        .Include(p => p.Supplier)
        .Include(p => p.Reviews)
        .ThenInclude(r => r.User)
        .FirstOrDefaultAsync(m => m.Id == id);

    if (product == null) return NotFound();

    return View(product);
}

[Authorize(Roles = "Admin")]
public IActionResult Create()
{
    ViewBag.Suppliers = new SelectList(_context.Suppliers, "Id", "Name");
    ViewBag.ExistingBrands = _context.Brands.Select(b => b.Name).ToList();
    ViewBag.ExistingCategories = _context.Categories.Select(c => c.Name).ToList();
    return View();
}

[HttpPost]
[ValidateAntiForgeryToken]
[Authorize(Roles = "Admin")]
public async Task<IActionResult> Create(ProductViewModel model)
{
    if (ModelState.IsValid)
    {
        using var transaction = await _context.Database.BeginTransactionAsync();
        try
        {
            // Get or create brand
            var brand = await _context.Brands
                .FirstOrDefaultAsync(b => b.Name.ToLower() == model.BrandName.Trim().ToLower());

            if (brand == null)
            {
                brand = new Brand { Name = model.BrandName.Trim() };
                _context.Brands.Add(brand);
                await _context.SaveChangesAsync();
            }

            // Get or create category
            var category = await _context.Categories
                .FirstOrDefaultAsync(c => c.Name.ToLower() == model.CategoryName.Trim().ToLower());

            if (category == null)
            {
                category = new Category { Name = model.CategoryName.Trim() };
                _context.Categories.Add(category);
            }

```

```

        if (category == null)
        {
            category = new Category { Name = model.CategoryName.Trim() };
            _context.Categories.Add(category);
            await _context.SaveChangesAsync();
        }

        // Create product
        var product = new Product
        {
            Name = model.Name,
            BrandId = brand.Id,
            CategoryId = category.Id,
            SupplierId = model.SupplierId,
            Price = model.Price,
            StockQuantity = model.StockQuantity,
            Description = model.Description,
            ImageUrl = model.ImageUrl
        };

        _context.Products.Add(product);
        await _context.SaveChangesAsync();
        await transaction.CommitAsync();

        return RedirectToAction(nameof(Index));
    }
    catch
    {
        await transaction.RollbackAsync();
        ModelState.AddModelError("", "Error saving product. Please try again.");
    }
}

// Repopulate needed data if validation fails
ViewBag.Suppliers = new SelectList(_context.Suppliers, "Id", "Name");
ViewBag.ExistingBrands = _context.Brands.Select(b => b.Name).ToList();
ViewBag.ExistingCategories = _context.Categories.Select(c => c.Name).ToList();
return View(model);
}

private void PopulateDropdowns()
{
    ViewBag.Brands = new SelectList(_context.Brands, "Id", "Name");
    ViewBag.Categories = new SelectList(_context.Categories, "Id", "Name");
    ViewBag.Suppliers = new SelectList(_context.Suppliers, "Id", "Name");
}

// GET: Products/Edit/5
[Authorize(Roles = "Admin")]
public async Task<IActionResult> Edit(int? id)
{
    if (id == null) return NotFound();

    var product = await _context.Products.FindAsync(id);
    if (product == null) return NotFound();
}

```

```

        PopulateDropdowns(product);
        return View(product);
    }

    // POST: Products/Edit/5
    [HttpPost]
    [ValidateAntiForgeryToken]
    [Authorize(Roles = "Admin")]
    public async Task<IActionResult> Edit(int id, Product product)
    {
        if (id != product.Id) return NotFound();

        if (ModelState.IsValid)
        {
            try
            {
                _context.Update(product);
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!ProductExists(product.Id)) return NotFound();
                throw;
            }
            return RedirectToAction(nameof(Index));
        }
        PopulateDropdowns(product);
        return View(product);
    }

    // GET: Products/Delete/5
    [Authorize(Roles = "Admin")]
    public async Task<IActionResult> Delete(int? id)
    {
        if (id == null) return NotFound();

        var product = await _context.Products
            .Include(p => p.Brand)
            .Include(p => p.Category)
            .Include(p => p.Supplier)
            .FirstOrDefaultAsync(m => m.Id == id);

        if (product == null) return NotFound();

        ViewBag.CanDelete = !await _context.OrderItems.AnyAsync(oi => oi.ProductId == id) &&
            !await _context.Reviews.AnyAsync(r => r.ProductId == id);

        return View(product);
    }

    // POST: Products/Delete/5
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    [Authorize(Roles = "Admin")]
    public async Task<IActionResult> DeleteConfirmed(int id)
    {
        var product = await _context.Products.FindAsync(id);
    }

```



```

namespace MusicStore.Controllers
{
    [Authorize(Roles = "Admin")]
    public class SuppliersController : Controller
    {
        private readonly Contexts _context;

        public SuppliersController(Contexts context)
        {
            _context = context;
        }

        public async Task<IActionResult> Index()
        {
            return View(await _context.Suppliers.ToListAsync());
        }

        public async Task<IActionResult> Details(int? id)
        {
            if (id == null) return NotFound();

            var supplier = await _context.Suppliers
                .FirstOrDefaultAsync(m => m.Id == id);

            return supplier == null ? NotFound() : View(supplier);
        }

        public IActionResult Create()
        {
            ViewBag.Brands = _context.Brands.ToList();
            ViewBag.Categories = _context.Categories.ToList();
            ViewBag.Products = _context.Products.ToList();
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Create(Supplier supplier,
            List<int> SelectedProductIds,
            List<string> NewProductNames,
            List<decimal> NewProductPrices,
            List<int> NewProductQuantities,
            List<int> SelectedBrandIds,
            List<string> NewBrandNames,
            List<int> SelectedCategoryIds,
            List<string> NewCategoryNames)
        {
            if (ModelState.IsValid)
            {
                _context.Suppliers.Add(supplier);
                _context.SaveChanges(); // Save supplier first

                // Handle existing products
                if (SelectedProductIds != null)
                {

```

```

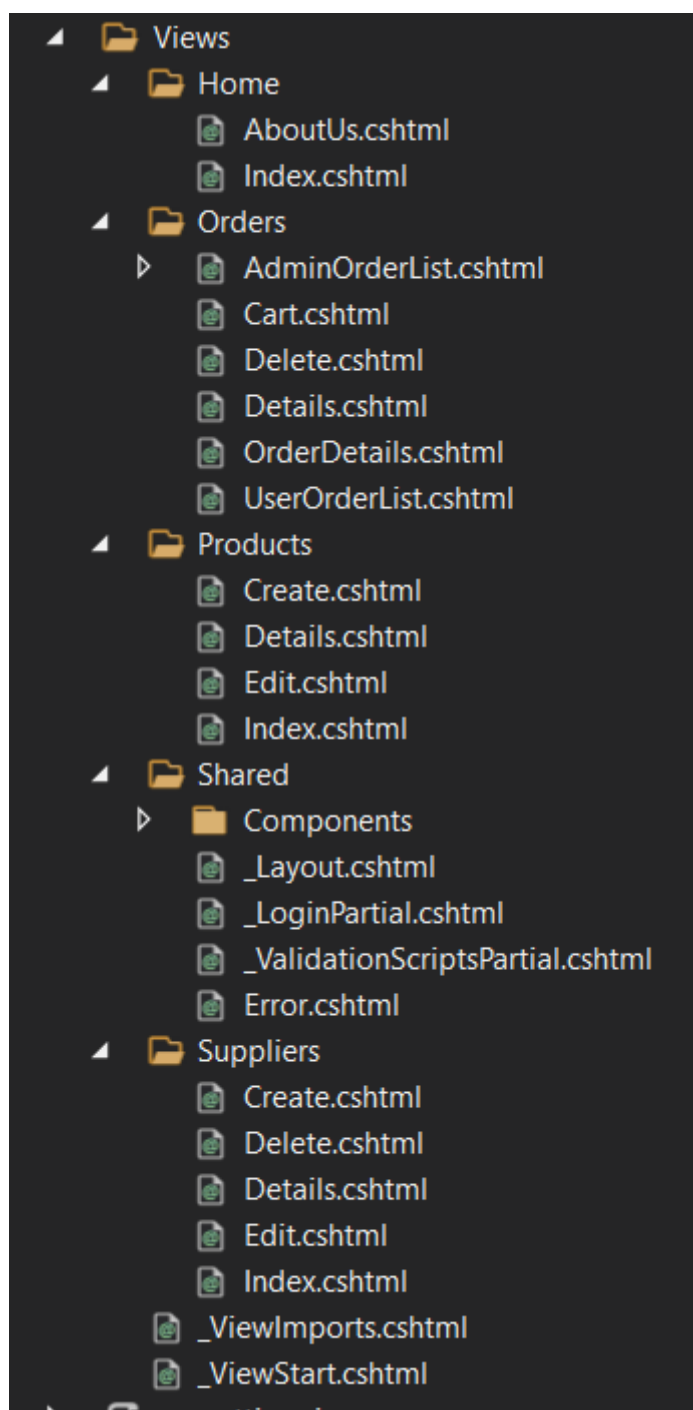
// Create new categories
var createdCategoryIds = new List<int>();
if (NewCategoryNames != null)
{
    foreach (var categoryName in NewCategoryNames)
    {
        var newCategory = new Category { Name = categoryName };
        _context.Categories.Add(newCategory);
        _context.SaveChanges();
        createdCategoryIds.Add(newCategory.Id);
    }
}

// Create new products
if (NewProductNames != null)
{
    for (int i = 0; i < NewProductNames.Count; i++)
    {
        var newProduct = new Product
        {
            Name = NewProductNames[i],
            Price = NewProductPrices[i],
            StockQuantity = NewProductQuantities[i],
            SupplierId = supplier.Id,
            BrandId = SelectedBrandIds[i] > 0 ? SelectedBrandIds[i] : createdBrandIds[i], // Use existing or new brand
            CategoryId = SelectedCategoryIds[i] > 0 ? SelectedCategoryIds[i] : createdCategoryIds[i] // Use existing or new category
        };
        _context.Products.Add(newProduct);
    }
}

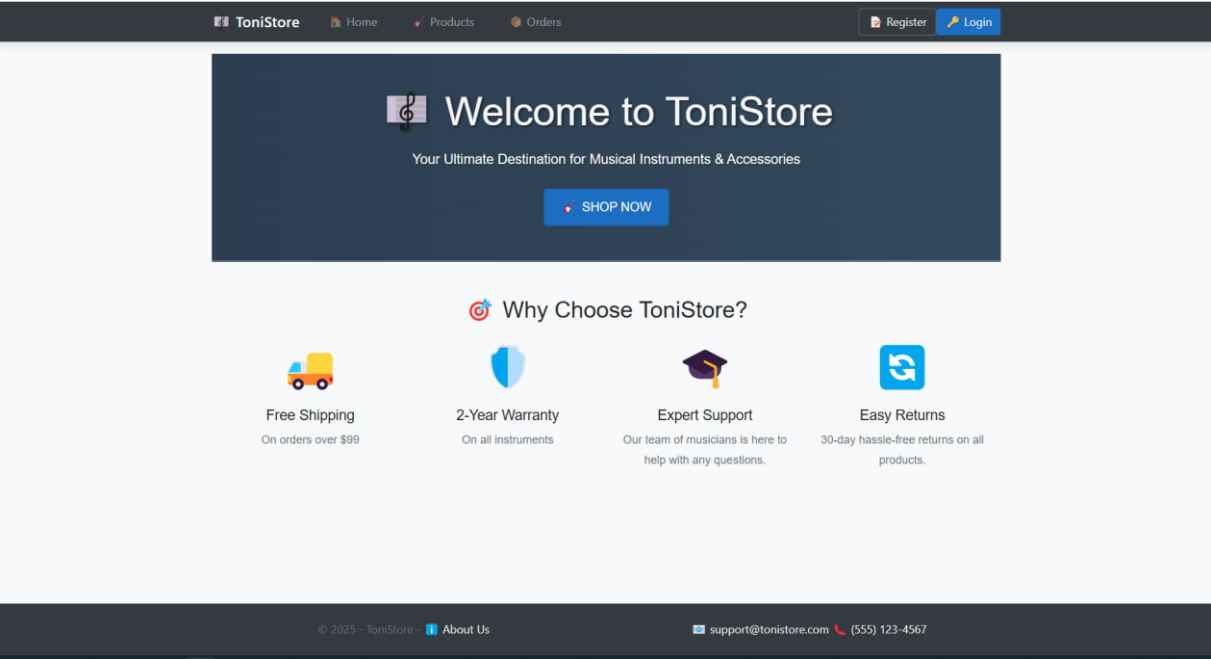
_context.SaveChanges();
return RedirectToAction(nameof(Index));
}

ViewBag.Brands = _context.Brands.ToList();
ViewBag.Categories = _context.Categories.ToList();
ViewBag.Products = _context.Products.ToList();
return View(supplier);
}

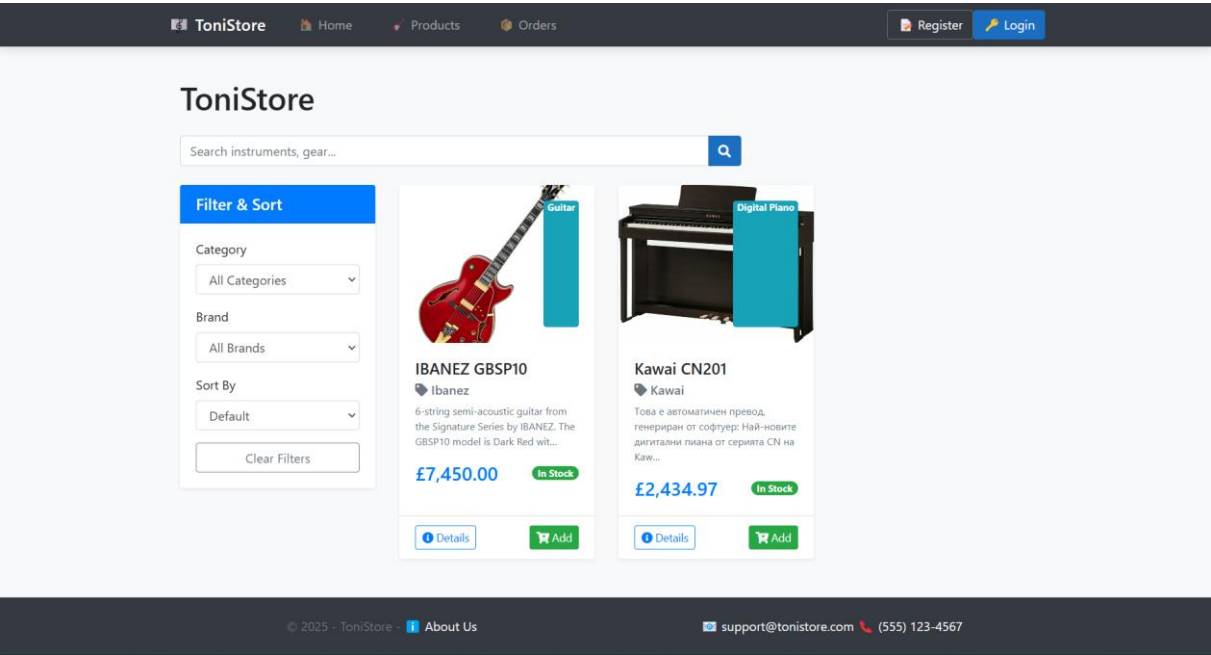
```

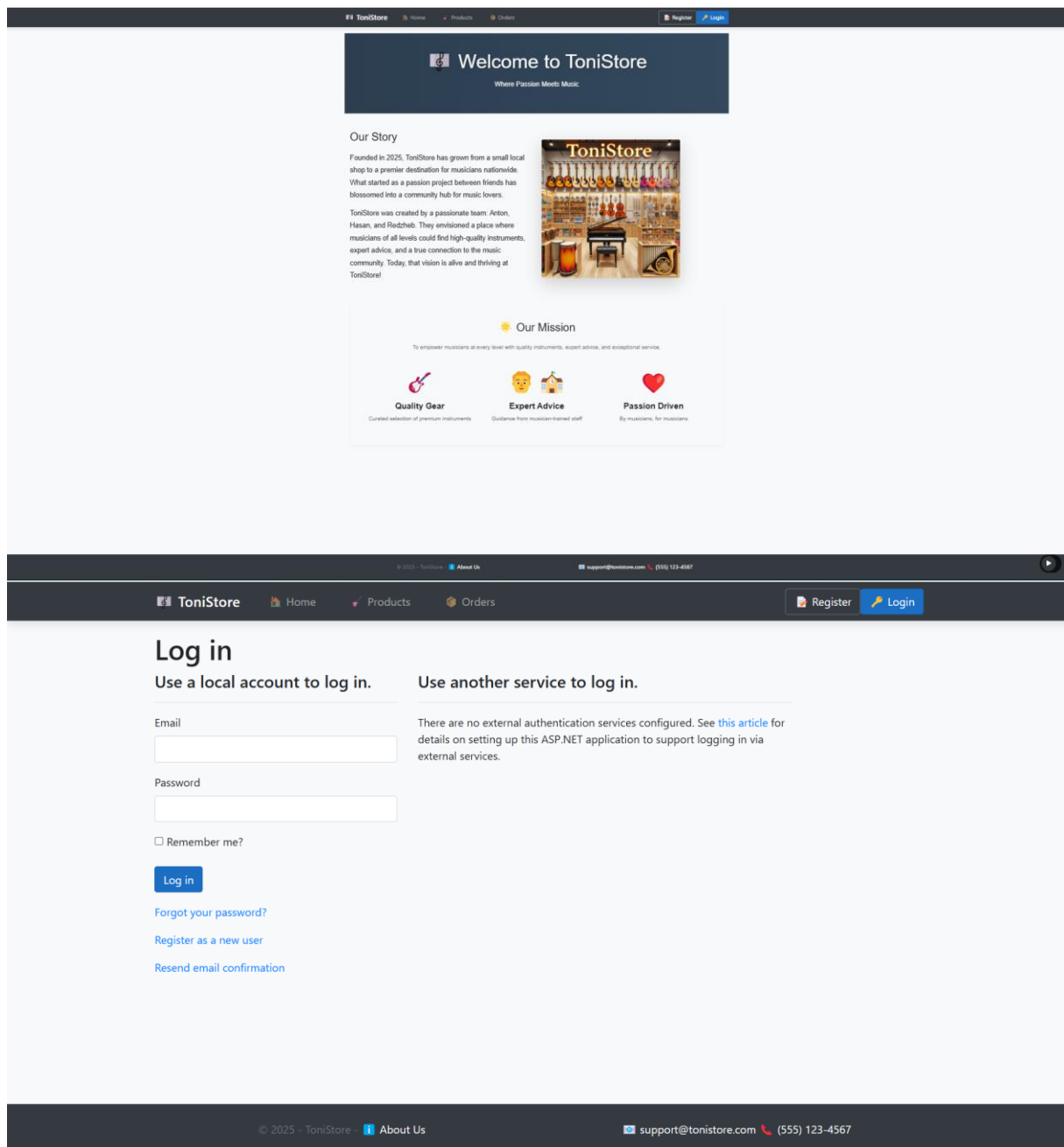


г) Снимки от уебсайта:



5.Заклучение:





- Създадохме ASP.NET проект, с който направихме уебсайт за музикален магазин

6.Използвана литература:

-Stack Overflow

-SoftUni

- GitHub

- YouTube

- rockshock

- muzikalen

- musicmag

- muziker
- music-store
- gemamusic