Universitatea Tehnică a Moldovei

Facultatea *Calculatoare, Informatică și Microelectronică*

Specialitatea *Tehnologii Informaționale*

Raport

la lucrarea de laborator nr. 6

Tema:*"Utilizarea unui CMS | Framework pentru dezvoltarea unei aplicații Web"*

Disciplina: "Tehnologii web"

A efectuat:	Student grupa TI-231 FR	Apareci Aurica
A verificat:	Asistent universitar	Rusu Viorel

Chișinău 2025

Cuprins

1. Cadru teoretic

Tema lucrării: Utilizarea unui CMS sau Framework pentru dezvoltarea unei aplicații Web.

Sarcina practică:

a) Alegeti un CMS sau Framework (JS sau pe oricare script server) pentru dezvoltarea unei aplicatii Web. Motivati alegerea.

b) Dezvoltati o aplicatie functionala in baza alegerii facute.

c) Puneti in raport PrintScreen-uri la paginile create. In cazul CMS-ului ilustrati modul de interactiune cu partea administrativa unde ati intervenit pentru adaptari, iar in cazul Framework-ului - citeva secvente de cod creat de voi pe care-l considerati important sau interesant.

2. Repere teoretice

În cadrul dezvoltării aplicației web, am utilizat următoarele tehnologii atât pe latura serverului (server-side) cât și pe latura clientului (client-side):

Server-side

ASP.NET Core: Am ales să utilizăm framework-ul ASP.NET Core pentru dezvoltarea serverului datorită performanței ridicate, scalabilității și suportului nativ pentru construirea de aplicații web moderne. ASP.NET Core ne oferă un mediu de dezvoltare robust și ușor de utilizat, precum și o integrare eficientă cu alte tehnologii și servicii.

Razor Pages: Am utilizat limbajul de templating Razor Pages, care face parte din framework-ul ASP.NET Core, pentru generarea paginilor web dinamice și gestionarea logicii serverului. Acesta ne-a permis să construim pagini web cu cod C# încorporat, facilitând manipularea datelor și generarea de conținut dinamic.

Entity Framework Core: Pentru gestionarea bazei de date, am ales să utilizăm Entity Framework Core, o tehnologie ORM (Object-Relational Mapping) puternică și ușor de utilizat. Entity Framework Core ne-a oferit un set de instrumente pentru interacțiunea cu baza de date, inclusiv crearea, citirea, actualizarea și ștergerea datelor. Acesta ne-a permis să lucrăm cu obiecte de tipul "code-first", facilitând dezvoltarea și întreținerea structurii bazei de date.

Client-side:

HTML, CSS și JavaScript: Acestea sunt tehnologiile de bază utilizate pentru construirea interfeței utilizatorului într-un browser web. HTML (HyperText Markup Language) este utilizat pentru structurarea și definirea conținutului paginilor web, CSS (Cascading Style Sheets) este folosit pentru stilizarea și formatarea acestora, iar JavaScript este utilizat pentru a adăuga interactivitate și funcționalități dinamice.

jQuery: Am inclus biblioteca jQuery pentru a simplifica manipularea DOM-ului (Document Object Model) și pentru a oferi un set de funcționalități și metode utile pentru dezvoltarea aplicației web. jQuery ne-a ajutat să gestionăm evenimente, să efectuăm animații și să interacționăm cu elementele paginii într-un mod ușor și eficient.

Bootstrap: Pentru crearea unei interfețe web responsive și estetice, am folosit framework-ul Bootstrap. Acesta ne-a furnizat un set de componente și stiluri predefinite, care ne-au permis să construim rapid și ușor un aspect profesional și coeziv pentru aplicația noastră. Bootstrap a oferit, de asemenea, un sistem de grilă flexibil, care a facilitat aranjarea și alinierea elementelor pe pagină.

Motivele alegerii acestor tehnologii au fost:

Maturitate și suport extins: Tehnologiile ASP.NET Core și Entity Framework Core sunt bine stabilite și au o comunitate de dezvoltatori activă, ceea ce ne-a oferit acces la resurse și documentație bogată.

Integrare și compatibilitate: ASP.NET Core a facilitat integrarea cu alte servicii și tehnologii, permițându-ne să construim aplicații web complexe și extensibile. De asemenea, HTML, CSS și JavaScript sunt tehnologii standard în dezvoltarea web, oferind interoperabilitate și compatibilitate între diferite platforme și browsere.

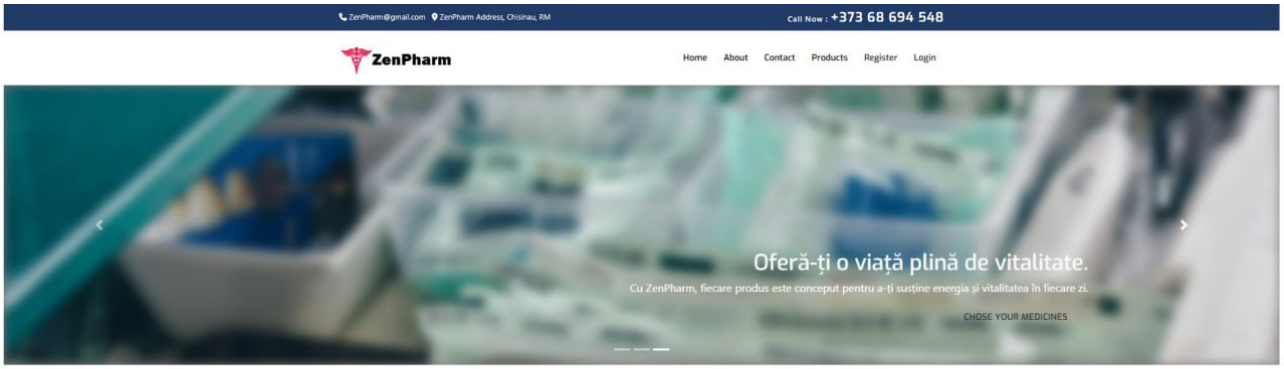
Productivitate și eficiență: Utilizarea Razor Pages ne-a permis să dezvoltăm rapid pagini web dinamice, evitând complexitatea arhitecturilor bazate pe MVC (Model-View-Controller). Entity Framework Core a simplificat gestionarea bazei de date și a oferit un set de instrumente puternice pentru interacțiunea cu datele.

Reutilizabilitate și consistență: jQuery și Bootstrap sunt tehnologii populare și bine documentate, care ne-au oferit un set de funcționalități predefinite și stiluri coezive. Utilizarea acestora ne-a permis să creăm interfețe web consistente și să reutilizăm codul existent, sporind eficiența dezvoltării.

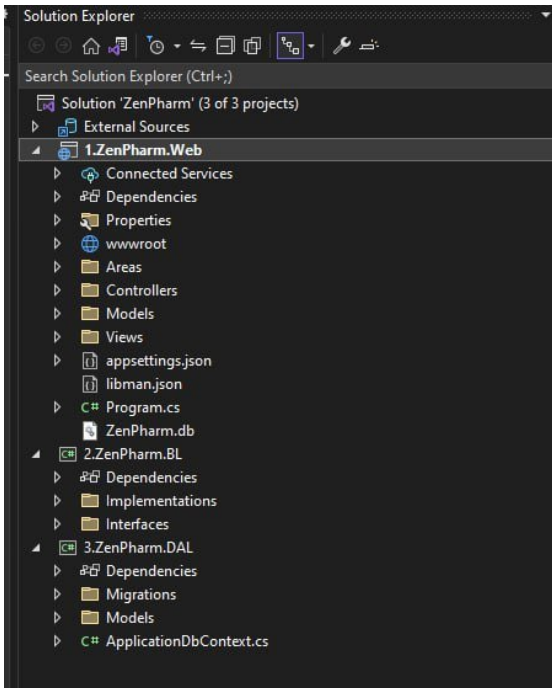
Prin utilizarea acestor tehnologii, am putut dezvolta o aplicație web modernă, scalabilă și ușor de întreținut, care oferă o experiență interactivă și plăcută utilizatorilor.

3. Sarcini practice

În cadrul acestui laborator, am realizat o aplicație web destinată gestionării activităților unei farmacii, denumită *ZenPharm*. Aplicația a fost dezvoltată folosind platforma .NET și are o arhitectură modulară, structurată în trei proiecte principale: **ZenPharm.Web**, **ZenPharm.BL** (Business Logic) și **ZenPharm.DAL** (Data Access Layer). Această structură reflectă separarea clară a responsabilităților în cadrul aplicației, favorizând scalabilitatea și mentenanța.



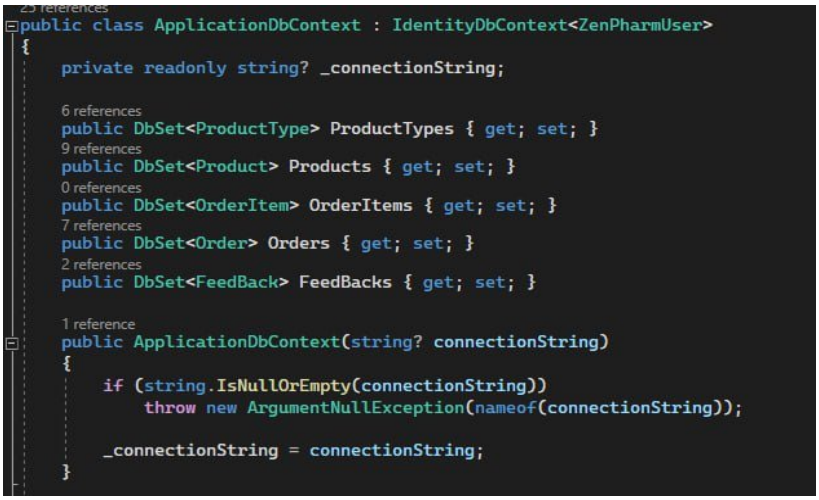
Proiectul **ZenPharm.Web** reprezintă partea de interfață cu utilizatorul (frontend și backend), incluzând directoare specifice aplicațiilor ASP.NET MVC precum Controllers, Views și Models. De asemenea, conține fișiere de configurare precum appsettings.json și Program.cs, alături de resurse statice în wwwroot.



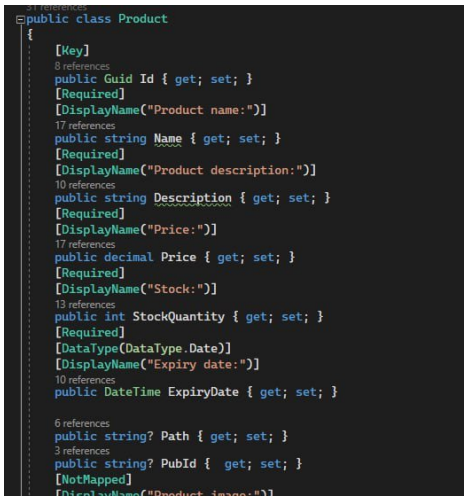
Componenta **ZenPharm.BL** este responsabilă de logica de afaceri a aplicației și este organizată în două subdirectoare: Interfaces (pentru definirea contractelor serviciilor) și Implementations (pentru clasele concrete). Această abordare permite o decuplare eficientă și facilitează testarea.

ZenPharm.DAL gestionează accesul la baza de date, conținând clasele de tip DbContext, modele de entități și migrări pentru configurarea bazei de date prin Entity Framework. Această arhitectură contribuie la o dezvoltare robustă, clar structurată și ușor extensibilă.

În cadrul aplicației ZenPharm, componenta de acces la date joacă un rol esențial în gestionarea informațiilor persistente ale farmaciei. Clasa *ApplicationDbContex*t, reprezintă contextul principal pentru interacțiunea cu baza de date și integrează identitatea utilizatorului prin moștenirea de la *IdentityDbContext<ZenPharmUser>*. În acest context sunt definite colecțiile DbSet, corespunzătoare entităților cheie din sistem: produse, tipuri de produse, comenzi, elemente ale comenzilor și feedbackuri. Fiecare entitate este mapată la o tabelă din baza de date, permițând realizarea operațiilor CRUD prin intermediul Entity Framework.

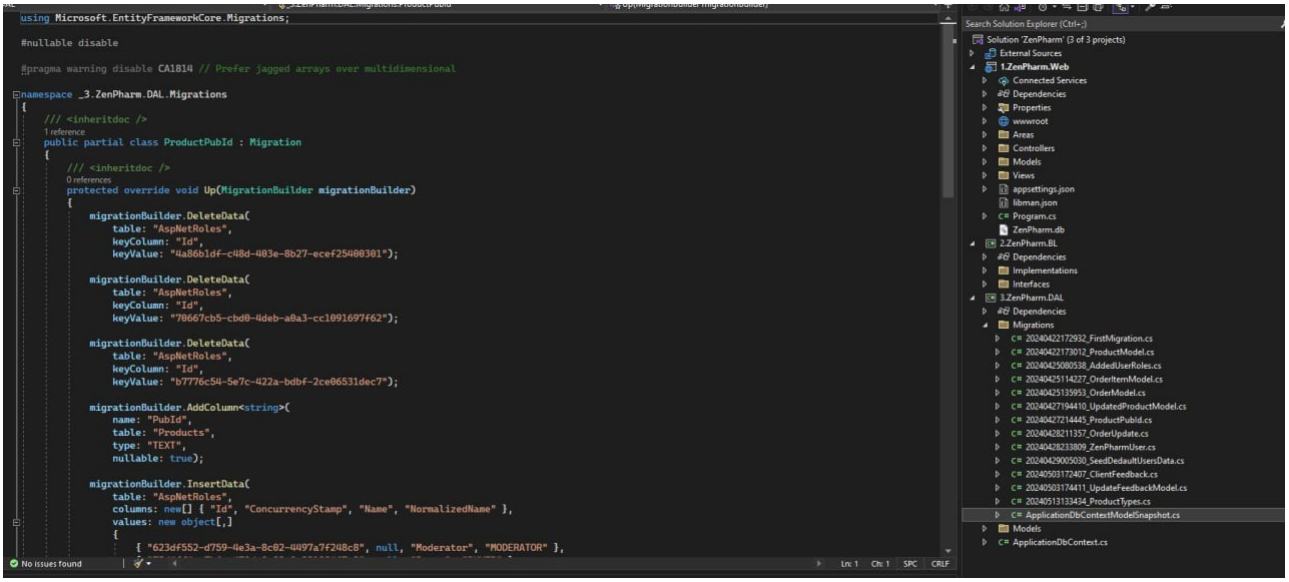


```
public class ApplicationDbContext : IdentityDbContext<ZenPharmUser>
{
    private readonly string? _connectionString;

    6 references
    public DbSet<ProductType> ProductTypes { get; set; }
    9 references
    public DbSet<Product> Products { get; set; }
    0 references
    public DbSet<OrderItem> OrderItems { get; set; }
    7 references
    public DbSet<Order> Orders { get; set; }
    2 references
    public DbSet<FeedBack> FeedBacks { get; set; }

    1 reference
    public ApplicationDbContext(string? connectionString)
    {
        if (string.IsNullOrEmpty(connectionString))
            throw new ArgumentNullException(nameof(connectionString));

        _connectionString = connectionString;
    }
```



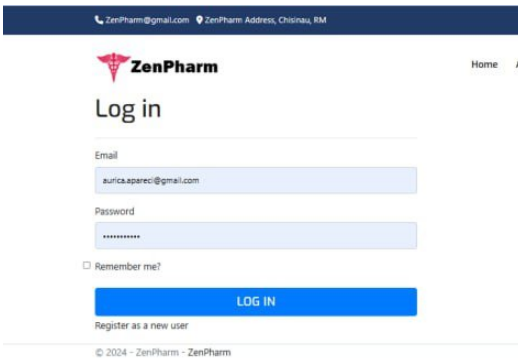
```
public class Product
{
    [Key]
    8 references
    public Guid Id { get; set; }
    [Required]
    [DisplayName("Product name:")]
    17 references
    public string Name { get; set; }
    [Required]
    [DisplayName("Product description:")]
    10 references
    public string Description { get; set; }
    [Required]
    [DisplayName("Price:")]
    17 references
    public decimal Price { get; set; }
    [Required]
    [DisplayName("Stock:")]
    13 references
    public int StockQuantity { get; set; }
    [Required]
    [DataType(DataType.Date)]
    [DisplayName("Expiry date:")]
    10 references
    public DateTime ExpiryDate { get; set; }

    6 references
    public string? Path { get; set; }
    3 references
    public string? PubId {  get; set; }
    [NotMapped]
```

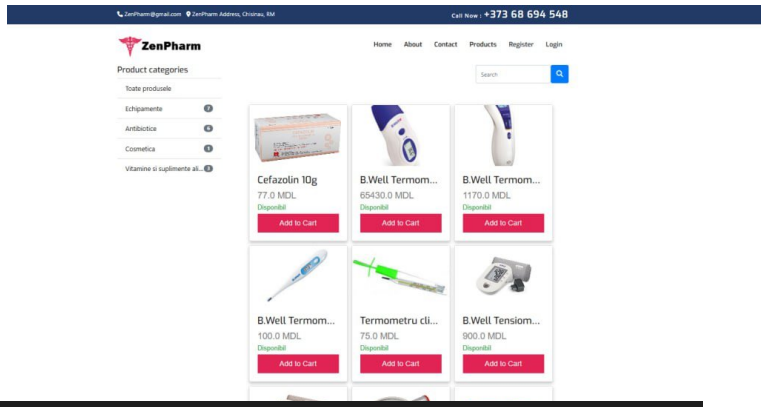
Un exemplu central este clasa Product, care modelează un produs farmaceutic. Aceasta include atribute precum nume, descriere, preț, stoc, dată de expirare și un câmp special pentru imagine, gestionat printr-un IFormFile. Atributele sunt adnotate cu validatori ([Required], [DisplayName], [DataType]) pentru a asigura integritatea datelor la nivelul aplicației. Evoluția structurii bazei de date este controlată prin sistemul de migrări EF Core, după cum se observă în fișierul ProductPubId, unde sunt realizate modificări asupra tabelei Products, precum adăugarea unui nou câmp.



```
using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

#pragma warning disable CA1814 // Prefer jagged arrays over multidimensional

namespace _3.ZenPharm.DAL.Migrations
{
    /// <inheritdoc />
    1 reference
    public partial class ProductPubId : Migration
    {
        /// <inheritdoc />
        0 references
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DeleteData(
                table: "AspNetRoles",
                keyColumn: "Id",
                keyValue: "4a86b1df-c48d-403e-8b27-ecef25400301");

            migrationBuilder.DeleteData(
                table: "AspNetRoles",
                keyColumn: "Id",
                keyValue: "78667cb5-cbd0-4deb-a0a3-cc1091697f62");

            migrationBuilder.DeleteData(
                table: "AspNetRoles",
                keyColumn: "Id",
                keyValue: "b7776c54-5e7c-422a-bdbf-2ce06531dec7");

            migrationBuilder.AddColumn<string>(
                name: "PubId",
                table: "Products",
                type: "TEXT",
                nullable: true);

            migrationBuilder.InsertData(
                table: "AspNetRoles",
                columns: new[] { "Id", "ConcurrencyStamp", "Name", "NormalizedName" },
                values: new object[,]
                {
                    { "623df552-d759-4e3a-8c02-4497a7f248c8", null, "Moderator", "MODERATOR" },
```

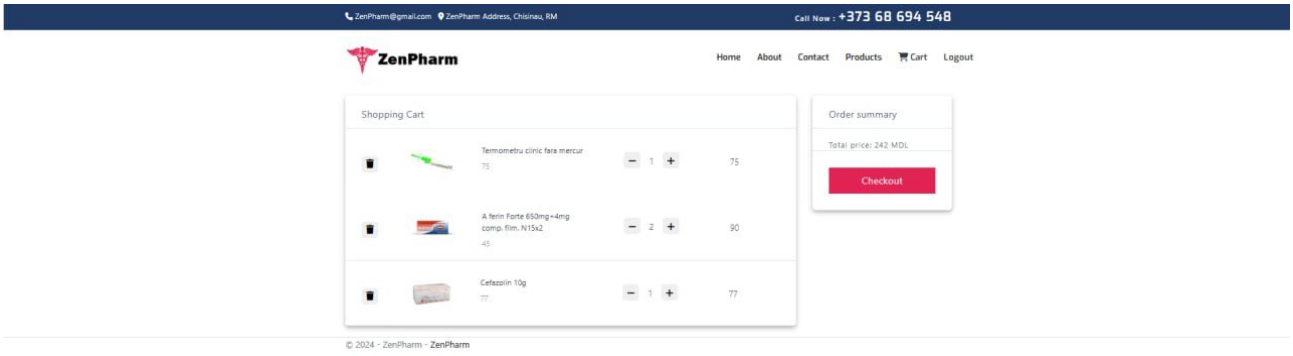
Fluxul de utilizare al aplicației ZenPharm este conceput pentru a oferi o experiență intuitivă și eficientă. Utilizatorul începe prin a-și crea un cont prin intermediul paginii de înregistrare, furnizând datele necesare pentru autentificare. Odată înregistrat, se poate autentifica pentru a accesa funcționalitățile complete ale platformei. După autentificare, utilizatorul navighează prin catalogul de produse disponibile și poate adăuga în coș articolele dorite. Coșul de cumpărături permite vizualizarea, actualizarea cantităților sau eliminarea produselor.



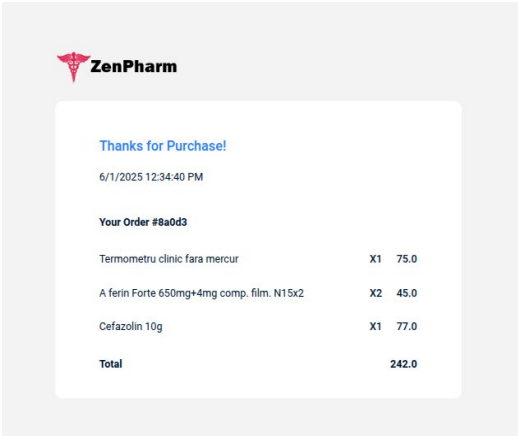



```
<br />
<div class="items py-4">
    <div class="d-flex flex-wrap justify-content-center">
        @foreach (var product in Model.Products)
        {
            <div class="card">
                @if (!string.IsNullOrEmpty(product.Path))
                {
                    <img src="@product.Path" alt="product" style="width:100%; max-height: 150px;object-fit:cover;">
                }
                else
                {
                    <img src="~/Images/placeholder.png" alt="product" style="width:100%; max-height: 150px;object-fit:cover;">
                }
                <div class="card-body">
                    <h4 class="title"><a href="@Url.Action("ProductDetails", "products", new {@productId = product.Id})">@product.Name</a></h4>
                    <p style="margin:0;" class="price">@product.Price MDL</p>
                    @if (product.StockQuantity > 0)
                    {
                        <p style="margin:0;" class="text-success">Disponibil</p>
                        <button class="to-cart" onclick="addToCart('@product.Id')">Add to Cart</button>
                    }
                    else
                    {
                        <p style="margin:0;" class="text-danger">Nu este in stoc</p>
                    }
                </div>
            </div>
```

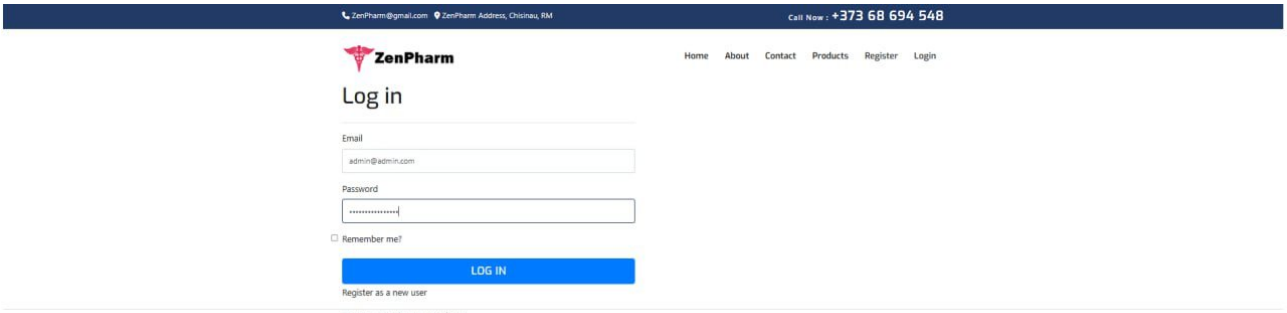
După finalizarea selecției, utilizatorul accesează pagina de checkout, unde confirmă comanda și introduce datele de livrare. După plasarea comenzii, sistemul procesează cererea, înregistrează detaliile în baza de date și generează un email de confirmare care este trimis automat către adresa utilizatorului, conținând informații despre produse, prețuri și detalii de livrare. Acest flux complet asigură o interacțiune coerentă și sigură cu platforma, acoperind toate etapele esențiale ale unui proces de cumpărare online într-o farmacie digitală.

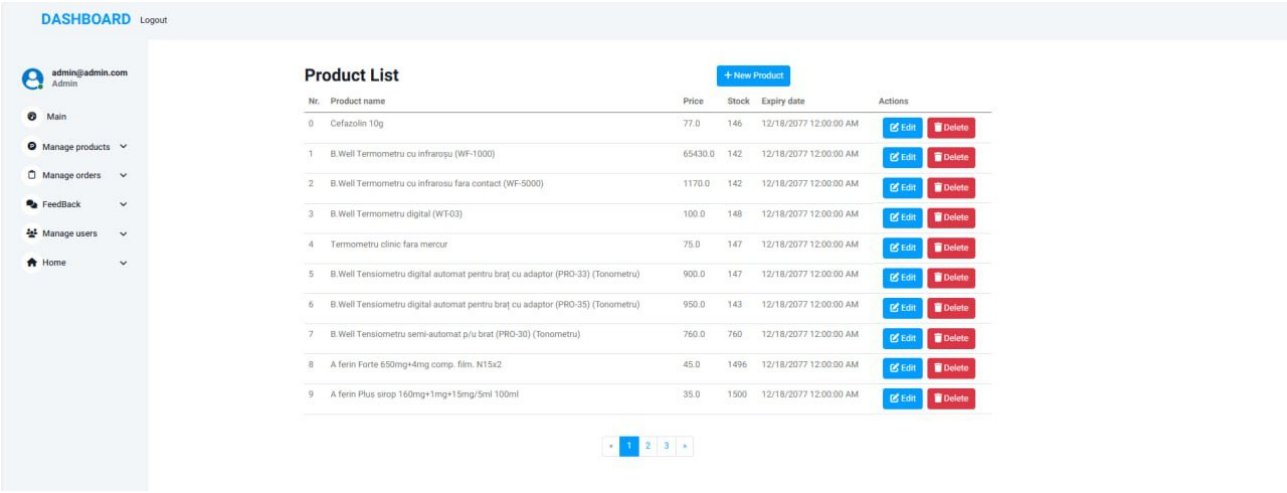


```
[HttpPost]
[Authorize]
0 references
public async Task<IActionResult> PlaceOrder([FromBody] OrderItemsViewModel order)
{
    if (!ModelState.IsValid)
        return await Task.FromResult(BadRequest(ModelState));

    var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);

    if (!(!string.IsNullOrEmpty(userId) && Guid.TryParse(userId, out var userIdGuid)))
        return await Task.FromResult(BadRequest("Invalid user ID."));

    var newOrder = new Order
    {
        UserID = userIdGuid,
        OrderItems = order.OrderItems.Select(x =>
            new OrderItem
            {
                OrderItemProductID = x.ProdId,
                Quantity = x.Quantity
            }).ToList()
    };
    await _orderService.AddOrder(newOrder);
    return Ok("Order placed successfully!");
}
```



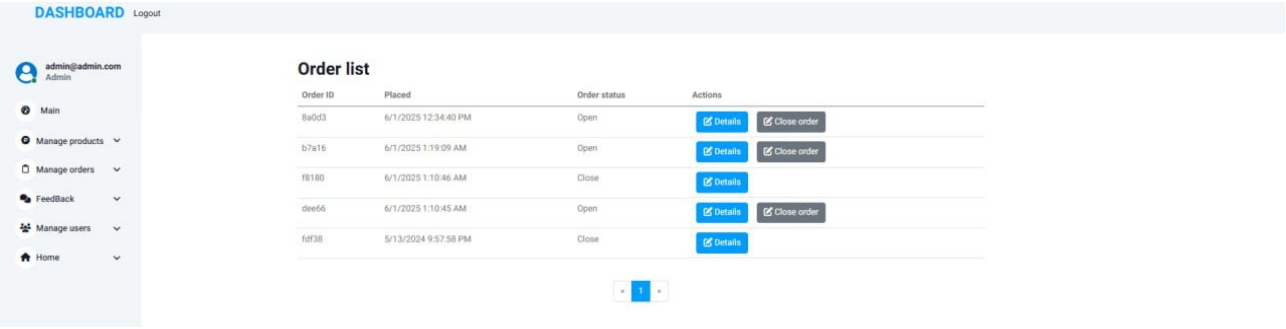
În calitate de administrator al aplicației ZenPharm, fluxul de utilizare este orientat către gestionarea completă a resurselor din sistem. După autentificare cu un cont de tip admin, utilizatorul este direcționat către un Dashboard dedicat, unde are acces la mai multe secțiuni printr-un meniu lateral. Administratorul poate vizualiza lista completă a produselor farmaceutice, afișate într-un tabel detaliat ce include numele produsului, prețul, stocul disponibil și data de expirare. De aici, poate adăuga un nou produs prin butonul „+ New Product", completând un formular cu informațiile necesare. Totodată, fiecare produs din listă poate fi modificat folosind butonul Edit, iar dacă este necesar, eliminat definitiv cu Delete.





Pe lângă gestionarea produselor, administratorul are acces la o secțiune dedicată comenzilor, unde poate vizualiza, actualiza sau anula comenzi în funcție de statusul acestora. În plus, în zona Manage users, poate gestiona utilizatorii înregistrați, inclusiv atribuirea sau modificarea rolurilor (ex. utilizator → moder-

ator sau admin), ceea ce permite un control flexibil asupra nivelului de acces. Acest flux oferă administratorului control deplin asupra aplicației și resurselor sale, asigurând un management eficient al platformei farmaceutice.




```
[HttpPost]
[Authorize(Roles = "Admin")]
0 references
public async Task<IActionResult> UpdateUserRole([FromBody] UpdateUserRoleViewModel update)
{
    var user = await _userManager.FindByIdAsync(update.userId);
    if (user == null)
        return NotFound();
    var role = await _roleManager.FindByIdAsync(update.newRoleId);
    if (role == null)
        return NotFound();
    var userRoles = await _userManager.GetRolesAsync(user);
    var result = await _userManager.RemoveFromRolesAsync(user, userRoles);
    if (!result.Succeeded)
        return Ok(result);
    result = await _userManager.AddToRoleAsync(user, role.Name);
    if (!result.Succeeded)
        return BadRequest(result);
    return RedirectToAction("AdminUserRoles");
}
```

În cadrul aplicației ZenPharm, pentru optimizarea gestionării fișierelor media, în special a imaginilor aferente produselor, a fost implementată funcționalitatea de încărcare în cloud. Această abordare permite stocarea imaginilor într-un serviciu cloud extern (precum Azure Blob Storage, Amazon S3 sau Cloudinary), în locul serverului local, ceea ce aduce multiple avantaje: reducerea spațiului ocupat pe server, creșterea vitezei de livrare a conținutului și scalabilitate sporită. La momentul adăugării unui produs nou, utilizatorul poate atașa o imagine folosind un câmp de tip IFormFile, iar fișierul este procesat și încărcat automat în cloud. Ulterior, în baza de date este salvat doar linkul de acces la imagine, nu și fișierul în sine. Această metodă contribuie la îmbunătățirea performanței aplicației și la o gestionare mai eficientă a resurselor multimedia, fiind totodată compatibilă cu standardele moderne de securitate și disponibilitate a datelor.

5. Concluzii

Aplicația ZenPharm a fost dezvoltată folosind tehnologia ASP.NET MVC, un framework robust care permite separarea clară a responsabilităților în cadrul aplicației, prin organizarea logicii în componente distincte: Model, View și Controller. Acest model arhitectural facilitează dezvoltarea, testarea și întreținerea codului. Pentru gestionarea datelor, s-a utilizat Entity Framework Core, un ORM (Object-Relational Mapper) care permite lucrul cu baze de date relaționale într-un mod obiectual, fără a fi necesară scrierea manuală a interogărilor SQL. Acesta oferă suport pentru migrații, actualizări ale schemei bazei de date și maparea entităților într-un mod flexibil și eficient. De asemenea, integrarea cu ASP.NET Identity a permis gestionarea autentificării și autorizării utilizatorilor, oferind un mecanism securizat și personalizabil pentru înregistrare, login și managementul rolurilor. Împreună, aceste tehnologii au oferit un mediu solid și scalabil pentru dezvoltarea aplicației farmaceutice, contribuind la realizarea unui sistem modern, sigur și ușor de extins.

Din perspectiva utilizatorului final, aplicația oferă o experiență fluentă și intuitivă: înregistrarea și autentificarea se realizează rapid, iar procesul de adăugare a produselor în coș și finalizarea comenzii este clar și eficient. Confirmarea comenzii prin email sporește încrederea utilizatorului în serviciile oferite și adaugă un plus de profesionalism.

Pentru administrator, platforma asigură control complet asupra conținutului și utilizatorilor. Posibilitatea de a adăuga, edita sau șterge produse, de a gestiona comenzile și de a modifica rolurile utilizatorilor face ca sistemul să fie adaptabil și ușor de administrat, chiar și în condiții dinamice.
În plus, integrarea mecanismului de încărcare a imaginilor în cloud contribuie semnificativ la optimizarea performanței aplicației și la scalabilitatea acesteia. Astfel, ZenPharm reprezintă un exemplu solid de aplicație web funcțională, securizată și bine structurată, destinată domeniului farmaceutic.

6. Webografie

1. https://else.fcim.utm.md/course/view.php?id=2318
2. https://stackoverflow.com/
3. https://www.w3schools.com/js/DEFAULT.asp
4. https://chatgpt.com/