Лабораторна робота № 6

Розробка клієнтських додатків з використанням Blazor WebAssembly.

Мета: отримати навички створення та інтеграції клієнтського додатку з RESTful Web API за допомогою Blazor WebAssembly.

Хід роботи

Завдання 1.

Створіть у своєму рішенні проект Blazor WebAssembly, який взаємодіятиме з WebAPI. Розробіть сторінки та необхідні компоненти для виконання CRUD-операцій з існуючими сутностями. Додайте валідацію для створених форм.

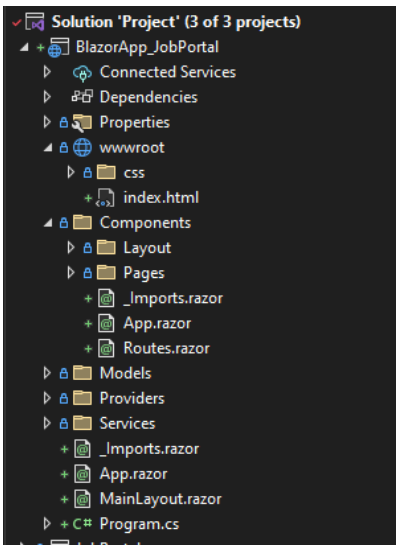

Рис. 1.1. Створений проект Blazor WebAssembly

Лістинг (Program.cs) налаштувань створеного проекту:

```
using BlazorApp_JobPortal;
using Blazored.LocalStorage;
using JobPortal.BlazorApp.Providers;
using JobPortal.BlazorApp.Services;
using Microsoft.AspNetCore.Components.Authorization;
using Microsoft.AspNetCore.Components.Web;
using Microsoft.AspNetCore.Components.WebAssembly.Hosting;

var builder = WebAssemblyHostBuilder.CreateDefault(args);

builder.Services.AddBlazoredLocalStorage();
builder.Services.AddAuthorizationCore();
```

```
builder.Services.AddScoped<AuthenticationStateProvider,
ApiAuthenticationStateProvider>();

builder.RootComponents.Add<App>("#app");
builder.RootComponents.Add<HeadOutlet>("head::after");

builder.Services.AddScoped(sp => new HttpClient { BaseAddress = new
Uri(builder.HostEnvironment.BaseAddress) });
builder.Services.AddScoped(sp =>
    new HttpClient { BaseAddress = new Uri("https://localhost:7283/api/") });

builder.Services.AddScoped<JobsService>();
builder.Services.AddScoped<AuthService>();

await builder.Build().RunAsync();
```

Лістинг моделей створеного проекту (з серверною валідацією):

```
namespace JobPortal.BlazorApp.Models
{
    public class UserDto
    {
        public string Id { get; set; }
        public string UserName { get; set; }
        public string Email { get; set; }
        public string Role { get; set; }
        public DateTime CreatedAt { get; set; }
    }
}


using System.ComponentModel.DataAnnotations;

namespace JobPortal.BlazorApp.Models
{
    public class RegisterDto
    {
        [Required(ErrorMessage = "Введіть повне ім'я")]
        public string FullName { get; set; } = string.Empty;

        [Required(ErrorMessage = "Введіть Email")]
        [EmailAddress(ErrorMessage = "Некоректний формат Email")]
        public string Email { get; set; } = string.Empty;

        [Required(ErrorMessage = "Введіть пароль")]
        [MinLength(6, ErrorMessage = "Пароль має бути мінімум 6 символів")]
        public string Password { get; set; } = string.Empty;

        [Required(ErrorMessage = "Підтвердіть пароль")]
        [Compare(nameof(Password), ErrorMessage = "Паролі не співпадають")]
        public string ConfirmPassword { get; set; } = string.Empty;

        [Required(ErrorMessage = "Оберіть роль")]
        public string Role { get; set; } = "Candidate";
    }
}

using System.ComponentModel.DataAnnotations;

namespace BlazorApp_JobPortal.Models
{
    public class LoginDto
    {
        [Required(ErrorMessage = "Введіть Email")]
        [EmailAddress(ErrorMessage = "Некоректний формат Email")]
        public string Email { get; set; } = string.Empty;

        [Required(ErrorMessage = "Введіть пароль")]
        public string Password { get; set; } = string.Empty;
```

```
        }
    }

    namespace BlazorApp_JobPortal.Models
    {
        public class AuthResponseDto
        {
            public string Token { get; set; } = string.Empty;
            public bool IsSuccess { get; set; }
            public string? ErrorMessage { get; set; }
        }
    }

    using System.ComponentModel.DataAnnotations;

    namespace JobPortal.BlazorApp.Models
    {
        public class JobDto
        {
            public int Id { get; set; }
            [Required(ErrorMessage = "Введіть назву вакансії")]
            [MinLength(3)]
            public string Title { get; set; }

            [Required(ErrorMessage = "Введіть опис")]
            public string Description { get; set; }

            [Required(ErrorMessage = "Введіть назву компанії")]
            public string Company { get; set; }
        }
    }
```

Лістинг сервісів створеного проекту:

```
    using BlazorApp_JobPortal.Models;
    using Blazored.LocalStorage;
    using JobPortal.BlazorApp.Models;
    using JobPortal.BlazorApp.Providers;
    using Microsoft.AspNetCore.Components.Authorization;
    using System.Net.Http.Json;

    namespace JobPortal.BlazorApp.Services
    {
        public class AuthService
        {
            private readonly HttpClient _http;
            private readonly ILocalStorageService _localStorage;
            private readonly AuthenticationStateProvider _authStateProvider;

            public AuthService(HttpClient http, ILocalStorageService localStorage,
AuthenticationStateProvider authStateProvider)
            {
                _http = http;
                _localStorage = localStorage;
                _authStateProvider = authStateProvider;
            }

            public async Task<bool> Register(RegisterDto model)
            {
                var result = await _http.PostAsJsonAsync("account/register", model);
                return result.IsSuccessStatusCode;
            }

            public async Task<bool> Login(LoginDto model)
            {
                var result = await _http.PostAsJsonAsync("account/login", model);

                if (!result.IsSuccessStatusCode)
```

```
                    return false;

                var response = await result.Content.ReadFromJsonAsync<AuthResponseDto>();

                if (response == null || string.IsNullOrEmpty(response.Token))
                    return false;

                await _localStorage.SetItemAsync("authToken", response.Token);


((ApiAuthenticationStateProvider)_authStateProvider).MarkUserAsAuthenticated(response.Token);

                return true;
            }

            public async Task Logout()
            {
                await _localStorage.RemoveItemAsync("authToken");
                ((ApiAuthenticationStateProvider)_authStateProvider).MarkUserAsLoggedOut();
                _http.DefaultRequestHeaders.Authorization = null;
            }
        }
    }

    using Blazored.LocalStorage;
    using JobPortal.BlazorApp.Models;
    using System.Net.Http.Headers;
    using System.Net.Http.Json;

    namespace JobPortal.BlazorApp.Services
    {
        public class JobsService
        {
            private readonly HttpClient _http;
            private readonly ILocalStorageService _localStorage;

            public JobsService(HttpClient http, ILocalStorageService localStorage)
            {
                _http = http;
                _localStorage = localStorage;
            }

            private async Task SetAuthHeader()
            {
                var token = await _localStorage.GetItemAsync<string>("authToken");

                if (!string.IsNullOrEmpty(token))
                {
                    _http.DefaultRequestHeaders.Authorization =
                        new AuthenticationHeaderValue("Bearer", token);
                }
            }

            public async Task<List<JobDto>> GetJobs()
            {
                // await SetAuthHeader();
                return await _http.GetFromJsonAsync<List<JobDto>>("jobs");
            }

            public async Task<JobDto> GetJob(int id)
            {
                // await SetAuthHeader();
                return await _http.GetFromJsonAsync<JobDto>($"jobs/{id}");
            }

            public async Task CreateJob(JobDto job)
            {
                await SetAuthHeader();
                await _http.PostAsJsonAsync("jobs", job);
```

```
        }

        public async Task UpdateJob(JobDto job)
        {
            await SetAuthHeader();
            await _http.PutAsJsonAsync($"jobs/{job.Id}", job);
        }

        public async Task DeleteJob(int id)
        {
            await SetAuthHeader();
            await _http.DeleteAsync($"jobs/{id}");
        }
    }
}
```

Лістинг сторінки виведення вакансій проекту:

```
@page "/jobs"
@using JobPortal.BlazorApp.Models
@using JobPortal.BlazorApp.Services
@using Microsoft.JSInterop
@inject JobsService JobsService
@inject IJSRuntime JS

<h3>Вакансії</h3>

@if (jobs == null)
{
    <div class="d-flex justify-content-center">
        <div class="spinner-border text-primary" role="status">
            <span class="visually-hidden">Завантаження...</span>
        </div>
    </div>
}
else
{
    <div class="mb-3 text-end">
        <a href="/jobs/create" class="btn btn-success">
            <span class="oi oi-plus" aria-hidden="true"></span> Створити нову
        </a>
    </div>

    <table class="table table-hover table-striped">
        <thead class="table-dark">
            <tr>
                <th>Назва</th>
                <th>Компанія</th>
                <th style="width: 300px;">Дії</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var j in jobs)
            {
                <tr>
                    <td>
                        <a href="/jobs/@j.Id" class="text-decoration-none fw-bold">
                            @j.Title
                        </a>
                    </td>
                    <td>@j.Company</td>
                    <td>
                        <div class="btn-group" role="group">
                            <a  href="/jobs/@j.Id"  class="btn  btn-info  btn-sm  text-
white">
                                Деталі
                            </a>
                            <a href="/jobs/edit/@j.Id" class="btn btn-warning btn-sm">
```

```
                                    Редагувати
                                </a>
                                <button class="btn btn-danger btn-sm" @onclick="() =>
Delete(j.Id, j.Title)">
                                    Видалити
                                </button>
                            </div>
                        </td>
                    </tr>
                }
            </tbody>
        </table>
    }

    @code {
        List<JobDto>? jobs;

        protected override async Task OnInitializedAsync()
        {
            await LoadJobs();
        }

        private async Task LoadJobs()
        {
            try
            {
                jobs = await JobsService.GetJobs();
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Помилка завантаження: {ex.Message}");
            }
        }

        async Task Delete(int id, string title)
        {
            bool confirmed = await JS.InvokeAsync<bool>("confirm", $"Ви впевнені, що хочете
видалити вакансію '{title}'?");

            if (confirmed)
            {
                await JobsService.DeleteJob(id);
                await LoadJobs();
            }
        }
    }
```

Лістинг сторінки реєстрації користувача:

```
    @page "/register"
    @using JobPortal.BlazorApp.Models
    @using JobPortal.BlazorApp.Services
    @using Microsoft.AspNetCore.Components.Forms
    @inject AuthService AuthService
    @inject NavigationManager Navigation

    <div class="row justify-content-center mt-5">
        <div class="col-md-6 col-lg-4">
            <div class="card shadow">
                <div class="card-header bg-primary text-white">
                    <h3 class="card-title text-center mb-0">Реєстрація</h3>
                </div>
                <div class="card-body">

                    @if (!string.IsNullOrEmpty(ErrorMessage))
                    {
                        <div class="alert alert-danger">@ErrorMessage</div>
                    }
```

```
                    <EditForm Model="registerModel" OnValidSubmit="HandleRegistration">
                        <DataAnnotationsValidator />

                        <div class="mb-3">
                            <label class="form-label">ПІБ (Повне ім'я)</label>
                            <InputText   @bind-Value="registerModel.FullName"   class="form-
control" placeholder="Іван Іванов" />
                            <ValidationMessage For="@(() => registerModel.FullName)" />
                        </div>

                        <div class="mb-3">
                            <label class="form-label">Email</label>
                            <InputText     @bind-Value="registerModel.Email"     class="form-
control" placeholder="example@mail.com" />
                            <ValidationMessage For="@(() => registerModel.Email)" />
                        </div>

                        <div class="mb-3">
                            <label class="form-label">Хто ви?</label>
                            <InputSelect    @bind-Value="registerModel.Role"    class="form-
select">
                                <option value="Candidate">Шукач роботи</option>
                                <option value="Employer">Роботодавець</option>
                            </InputSelect>
                        </div>

                        <div class="mb-3">
                            <label class="form-label">Пароль</label>
                            <InputText  @bind-Value="registerModel.Password"  type="password"
class="form-control" />
                            <ValidationMessage For="@(() => registerModel.Password)" />
                        </div>

                        <div class="mb-3">
                            <label class="form-label">Підтвердження паролю</label>
                            <InputText            @bind-Value="registerModel.ConfirmPassword"
type="password" class="form-control" />
                            <ValidationMessage  For="@(()  =>  registerModel.ConfirmPassword)"
/>
                        </div>

                        <div class="d-grid gap-2">
                            <button        type="submit"       class="btn       btn-success"
disabled="@IsProcessing">
                                @if (IsProcessing)
                                {
                                    <span  class="spinner-border  spinner-border-sm"></span>
<span> Реєстрація...</span>
                                }
                                else
                                {
                                    <span>Зареєструватися</span>
                                }
                            </button>
                        </div>
                    </EditForm>
                </div>
                <div class="card-footer text-center">
                    <small>Вже є акаунт? <a href="/login">Увійти</a></small>
                </div>
            </div>
        </div>
    </div>

    @code {
        private RegisterDto registerModel = new RegisterDto();
        private bool IsProcessing = false;
        private string? ErrorMessage;
```

```
        private async Task HandleRegistration()
        {
            IsProcessing = true;
            ErrorMessage = null;

            try
            {
                var isSuccess = await AuthService.Register(registerModel);
                if (isSuccess)
                {
                    Navigation.NavigateTo("/login");
                }
                else
                {
                    ErrorMessage = "Помилка реєстрації. Перевірте дані.";
                }
            }
            catch (Exception ex)
            {
                ErrorMessage = $"Помилка: {ex.Message}";
            }
            finally
            {
                IsProcessing = false;
            }
        }
    }
```

Лістинг Program.cs API проекту (додаємо дозвіл використання Blazor):

```
            builder.Services.AddCors(options =>
            {
                options.AddPolicy("AllowBlazorClient",
                    policy =>
                    {
                        policy.WithOrigins("https://localhost:5288",
"http://localhost:5288")
                              .AllowAnyHeader()
                              .AllowAnyMethod();
                    });
            });
```

Результат виконаного завдання:

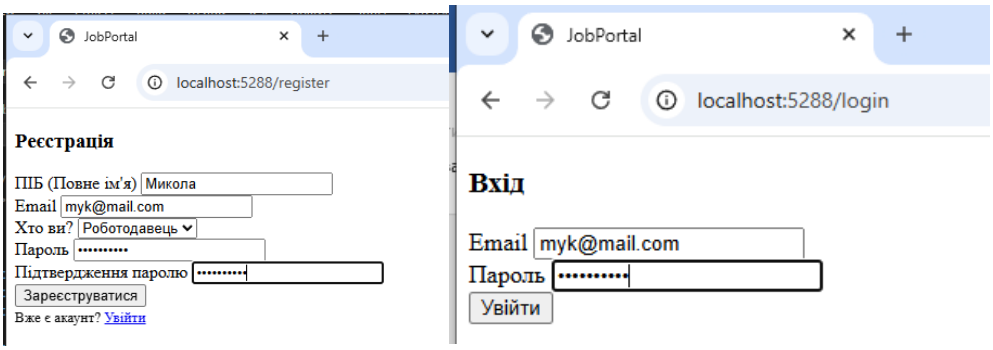

Рис. 1.2-3. Реєстрація та авторизація користувача

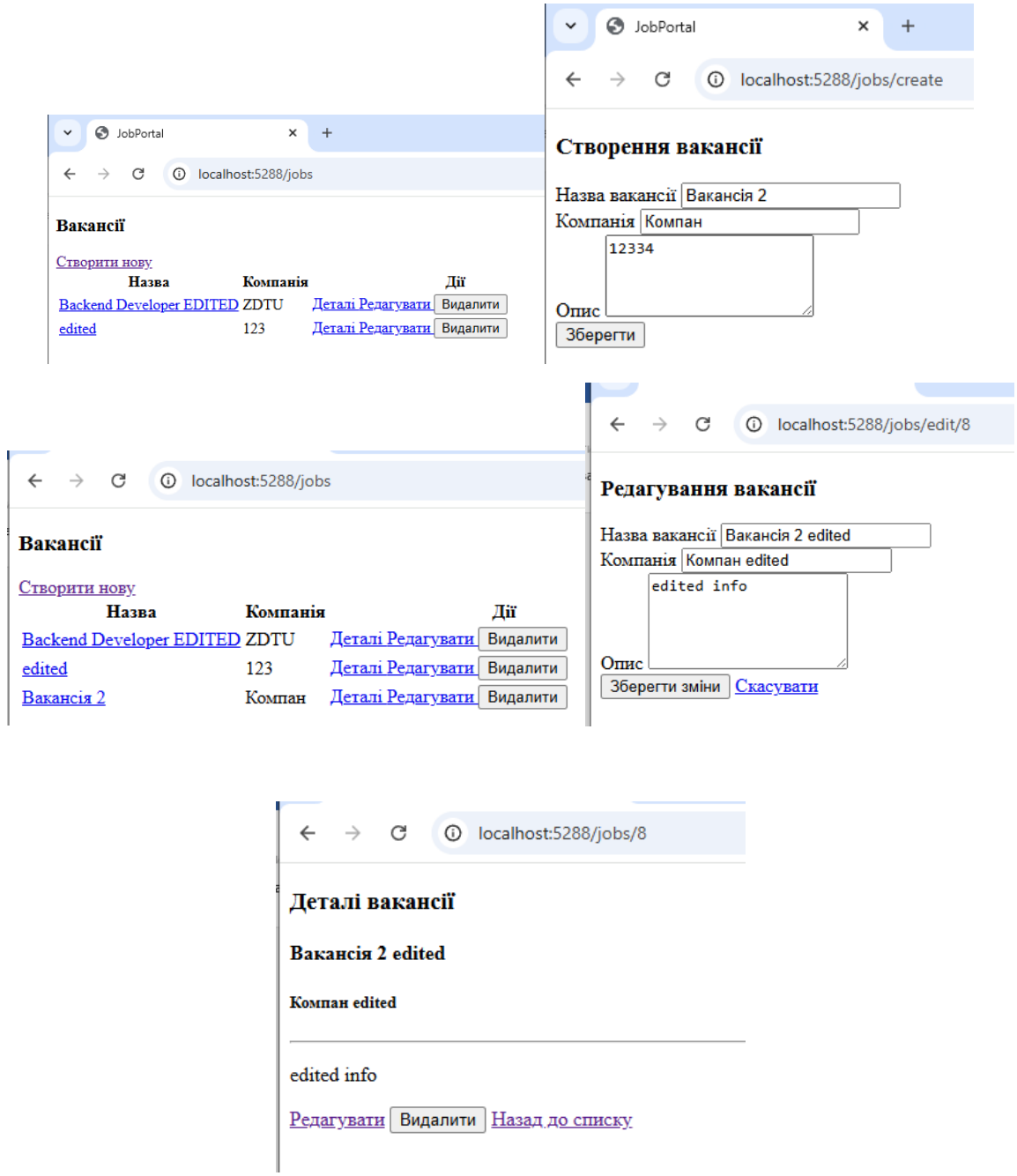

Рис. 1.4-8. CRUD операції з вакансіями

Завдання 2.

Додайте автентифікацію та авторизацію до клієнта. Додайте сторінки та відповідні форми для реєстрації та входу. Реалізуйте автентифікацію на основі JWT. Приклад проекту, який реалізує автентифікацію та авторизацію за допомогою WebAPI та Blazor WebAssembly, можна знайти за наступним посиланням. Докладніше з механізмами автентифікації та авторизації в Blazor WebAssembly можна ознайомитись тут.

Лістинг зміненого Program.cs:

```
using Microsoft.AspNetCore.Components.Web;
using Microsoft.AspNetCore.Components.WebAssembly.Hosting;
using Blazored.LocalStorage;
using Microsoft.AspNetCore.Components.Authorization;
using JobPortal.BlazorApp.Services;
using JobPortal.BlazorApp.Providers;
using BlazorApp_JobPortal;

var builder = WebAssemblyHostBuilder.CreateDefault(args);
builder.RootComponents.Add<App>("#app");
builder.RootComponents.Add<HeadOutlet>("head::after");

builder.Services.AddScoped(sp => new HttpClient { BaseAddress = new
Uri("https://localhost:7283/api/") });

builder.Services.AddAuthorizationCore();

builder.Services.AddBlazoredLocalStorage();

builder.Services.AddScoped<AuthenticationStateProvider,
ApiAuthenticationStateProvider>();

builder.Services.AddScoped<AuthService>();
builder.Services.AddScoped<JobsService>();

await builder.Build().RunAsync();
```

Лістинг зміненого Login.razor:

```
@page "/login"
@inject AuthService AuthService
@inject NavigationManager Nav
@using BlazorApp_JobPortal.Models
@using JobPortal.BlazorApp.Models
@using JobPortal.BlazorApp.Services
@using Microsoft.AspNetCore.Components.Forms

<h3>Вхід</h3>
@if(Error) { <div class="alert alert-danger">Помилка входу</div> }

<EditForm Model="model" OnValidSubmit="HandleLogin">
    <DataAnnotationsValidator />
    <div class="mb-3">
        <label>Email</label>
        <InputText @bind-Value="model.Email" class="form-control" />
    </div>
    <div class="mb-3">
        <label>Пароль</label>
```

```
            <InputText @bind-Value="model.Password" type="password" class="form-control" />
        </div>
        <button class="btn btn-primary">Увійти</button>
    </EditForm>

    @code {
        LoginDto model = new();
        bool Error;

        async Task HandleLogin()
        {
            var result = await AuthService.Login(model);
            if (result) Nav.NavigateTo("/jobs");
            else Error = true;
        }
    }
```

Лістинг зміненого ApiAuthenticationStateProvider:

```
    using Blazored.LocalStorage;
    using Microsoft.AspNetCore.Components.Authorization;
    using System.Net.Http.Headers;
    using System.Security.Claims;
    using System.Text.Json;

    namespace JobPortal.BlazorApp.Providers
    {
        public class ApiAuthenticationStateProvider : AuthenticationStateProvider
        {
            private readonly ILocalStorageService _localStorage;
            private readonly HttpClient _http;

            public     ApiAuthenticationStateProvider(ILocalStorageService     localStorage,
HttpClient http)
            {
                _localStorage = localStorage;
                _http = http;
            }

            public override async Task<AuthenticationState> GetAuthenticationStateAsync()
            {
                var token = await _localStorage.GetItemAsync<string>("authToken");

                if (string.IsNullOrWhiteSpace(token))
                {
                    // Немає токена -> Анонім
                    return       new       AuthenticationState(new       ClaimsPrincipal(new
ClaimsIdentity()));
                }

                // Є токен -> Встановлюємо його в заголовок HTTP для майбутніх запитів
                _http.DefaultRequestHeaders.Authorization                =                new
AuthenticationHeaderValue("Bearer", token);

                return        new        AuthenticationState(new        ClaimsPrincipal(new
ClaimsIdentity(ParseClaimsFromJwt(token), "jwt")));
            }

            public void MarkUserAsAuthenticated(string token)
            {
                var        authenticatedUser        =        new        ClaimsPrincipal(new
ClaimsIdentity(ParseClaimsFromJwt(token), "jwt"));
                var authState = Task.FromResult(new AuthenticationState(authenticatedUser));
                NotifyAuthenticationStateChanged(authState);
            }

            public void MarkUserAsLoggedOut()
            {
                var anonymousUser = new ClaimsPrincipal(new ClaimsIdentity());
```

```
                var authState = Task.FromResult(new AuthenticationState(anonymousUser));
                NotifyAuthenticationStateChanged(authState);
            }

            // Парсинг JWT (технічний метод)
            private IEnumerable<Claim> ParseClaimsFromJwt(string jwt)
            {
                var payload = jwt.Split('.')[1];
                var jsonBytes = ParseBase64WithoutPadding(payload);
                var     keyValuePairs     =     JsonSerializer.Deserialize<Dictionary<string,
object>>(jsonBytes);

                return       keyValuePairs!.Select(kvp       =>       new       Claim(kvp.Key,
kvp.Value.ToString()!));
            }

            private byte[] ParseBase64WithoutPadding(string base64)
            {
                switch (base64.Length % 4)
                {
                    case 2: base64 += "=="; break;
                    case 3: base64 += "="; break;
                }
                return Convert.FromBase64String(base64);
            }
        }
    }
```

Лістинг зміненого AuthService.cs:

```
    using Blazored.LocalStorage;
    using JobPortal.BlazorApp.Providers;
    using Microsoft.AspNetCore.Components.Authorization;
    using System.Net.Http.Json;
    using System.Net.Http.Headers;
    using BlazorApp_JobPortal.Models;
    using JobPortal.BlazorApp.Models;

    namespace JobPortal.BlazorApp.Services
    {
        public class AuthService
        {
            private readonly HttpClient _http;
            private readonly AuthenticationStateProvider _authStateProvider;
            private readonly ILocalStorageService _localStorage;

            public       AuthService(HttpClient       http,       AuthenticationStateProvider
authStateProvider, ILocalStorageService localStorage)
            {
                _http = http;
                _authStateProvider = authStateProvider;
                _localStorage = localStorage;
            }

            public async Task<bool> Register(RegisterDto registerModel)
            {
                var result = await _http.PostAsJsonAsync("account/register", registerModel);
                return result.IsSuccessStatusCode;
            }

            public async Task<bool> Login(LoginDto loginModel)
            {
                var result = await _http.PostAsJsonAsync("account/login", loginModel);

                if (!result.IsSuccessStatusCode) return false;

                var response = await result.Content.ReadFromJsonAsync<AuthResponseDto>();

                if (response == null || string.IsNullOrEmpty(response.Token)) return false;
```

```
                await _localStorage.SetItemAsync("authToken", response.Token);

((ApiAuthenticationStateProvider)_authStateProvider).MarkUserAsAuthenticated(response.Token);

                _http.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", response.Token);

                return true;
            }

            public async Task Logout()
            {
                await _localStorage.RemoveItemAsync("authToken");
                ((ApiAuthenticationStateProvider)_authStateProvider).MarkUserAsLoggedOut();
                _http.DefaultRequestHeaders.Authorization = null;
            }
        }

        public class AuthResponseDto { public string Token { get; set; } }
    }
```

Висновок: в рамках цієї лабораторної роботи були отримані навички створення та інтеграції клієнтського додатку з RESTful Web API за допомогою Blazor WebAssembly. Роботу виконано у повному обсязі.