Autentikacija pomoću WS-FED (ADFS) protokola

Pored SAML i OIDC protokola sustav AAI@EduHr za ASP.NET aplikacije omogućuje korištenje i WS-FED (ADFS) protokola.

Potrebno je instalirati nuget paket microsoft.aspnetcore.authentication.wsfederation i slijediti upute koje su dostupne na Microsoftovim stranicama:
https://learn.microsoft.com/en-us/aspnet/core/security/authentication/ws-federation?view=aspnetcore-8.0#add-ws-federation-as-an-external-login-provider-for-aspnet-core-identity

U verzijama .NET 7 i 8 postoji bug koji do pisanja ovih uputa nije ispravljen pa je potrebno primijeniti zakrpu koja je opisana ovdje:
https://github.com/AzureAD/azure-activedirectory-identitymodel-extensions-for-dotnet/issues/1258

Slijedi primjer koda za ASP.NET 8 Core Web c# aplikacije:

```
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Authentication.WsFederation;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens.Saml;
using Microsoft.IdentityModel.Tokens;
using primjer.Data;
using System.Security.Claims;
using Microsoft.Extensions.DependencyInjection;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddAuthentication(sharedOptions =>
{
    sharedOptions.DefaultScheme = CookieAuthenticationDefaults.AuthenticationScheme;
    sharedOptions.DefaultChallengeScheme = WsFederationDefaults.AuthenticationScheme;
})
    .AddWsFederation(wo =>
    {
    wo.MetadataAddress = "https://fed-lab.aaiedu.hr/sso/module.php/adfs/idp/metadata.php";
    wo.Wtrealm = "https://testna-aplikacija.hr";
    wo.TokenHandlers.Clear();
    wo.TokenHandlers.Add(new CustomSamlSecurityTokenHandler());
        wo.Events = new WsFederationEvents()
        {
            OnSecurityTokenValidated = async context =>
            {
                // Ukoliko su potrebne dodatne provjere korisnika
                var ci = context.Principal.Identity as ClaimsIdentity;

            }
        };
    })
     .AddCookie();

// Add services to the container.
var connectionString = builder.Configuration.GetConnectionString("DefaultConnection") ?? throw new
InvalidOperationException("Connection string 'DefaultConnection' not found.");
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(connectionString));
builder.Services.AddDatabaseDeveloperPageExceptionFilter();

builder.Services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = true)
    .AddEntityFrameworkStores<ApplicationDbContext>();
builder.Services.AddRazorPages();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseMigrationsEndPoint();
}
else
{
    app.UseExceptionHandler("/Error");
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();
```

```
app.UseAuthorization();

app.MapRazorPages();

app.Run();



public class CustomSamlSecurityTokenHandler : SamlSecurityTokenHandler
{
    public override async Task<TokenValidationResult> ValidateTokenAsync(string token, TokenValidationParameters
validationParameters)
    {
        if (token.Contains("&#xD;"))
        {
            token = token.Replace("&#xD;", "");
        }
        try
        {
            var configuration = await validationParameters.ConfigurationManager.GetBaseConfigurationAsync
(CancellationToken.None).ConfigureAwait(false);
            var issuers = new[] { configuration.Issuer };
            validationParameters.ValidIssuers = validationParameters.ValidIssuers == null ? issuers :
validationParameters.ValidIssuers.Concat(issuers);
            validationParameters.IssuerSigningKeys = validationParameters.IssuerSigningKeys == null ?
configuration.SigningKeys : validationParameters.IssuerSigningKeys.Concat(configuration.SigningKeys);
            validationParameters.IssuerSigningKey = configuration.SigningKeys.First();
            validationParameters.ValidIssuer = configuration.Issuer;
        }
        catch (Exception e)
        {
            Log.Error(e, "Greška u custom handleru: {token}", token);
        }
        return await base.ValidateTokenAsync(token, validationParameters);
    }
}
```

Hvala kolegi Vidu Kašiću na pomoći pri izradi ovih uputa.