Praca Magisterska

# PORÓWNANIE APLIKACJI WEBOWYCH TYPU API + KLIENT ORAZ APLIKACJI MVC

Marcin Drelewski

Promotor: dr Wojciech Gwizdała

# Motywacja podjęcia tematu pracy

- Słabo dostępne informacje na temat różnic w obu podejściach
- Poszerzanie wiedzy w zakresie tworzenia aplikacji webowych przy wykorzystaniu różnych metod
- Chęć porównania ich z uwagi na zainteresowanie tematyką

# Cel Pracy

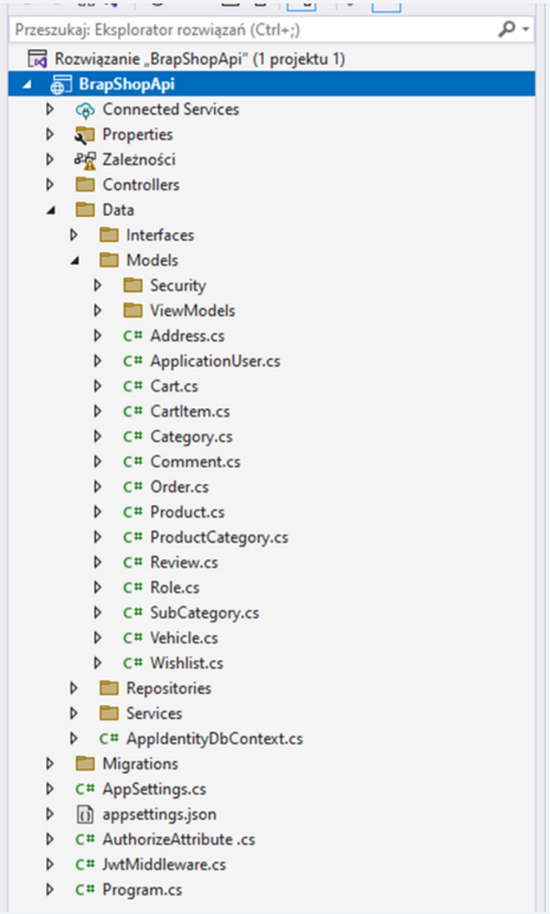
Zaprezentowanie i omówienie wybranych różnic w obu podejściach do tworzenia aplikacji internetowych na przykładzie aplikacji podzielonej na warstwę serwerową, opartą o platformę .Net oraz warstwę kliencką na platformie Angular oraz aplikacji MVC stworzonej przy wykorzystaniu nowoczesnej platformy Microsoftu o nazwie Blazor

# Zastosowane Technologie

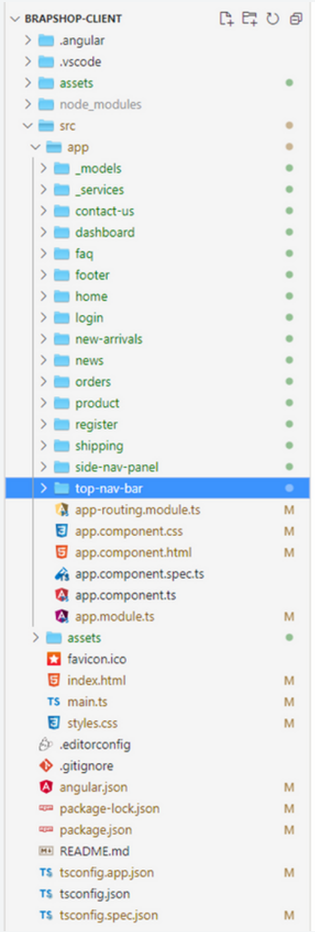
- .NET
- Angular
- Blazor
- Linq
- Entity Framework Core
- Bootstrap

# Struktura aplikacji

API

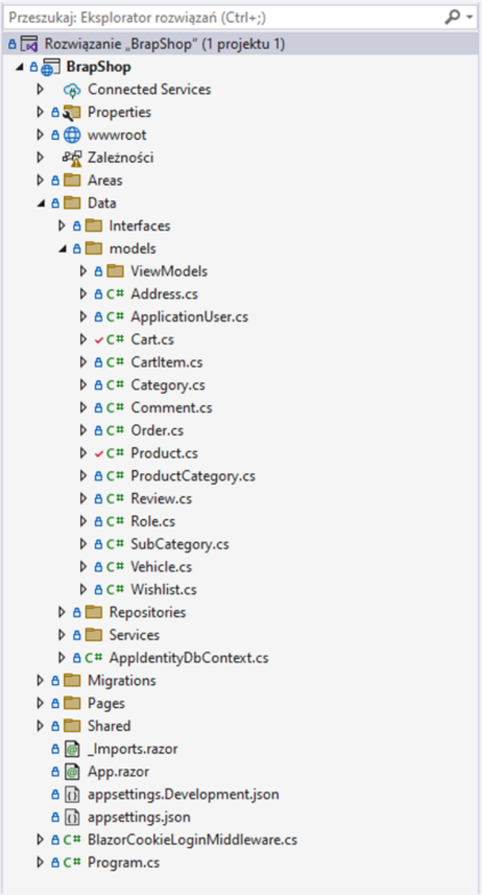



Klient




VS

Blazor



# Plik Program.cs

## API

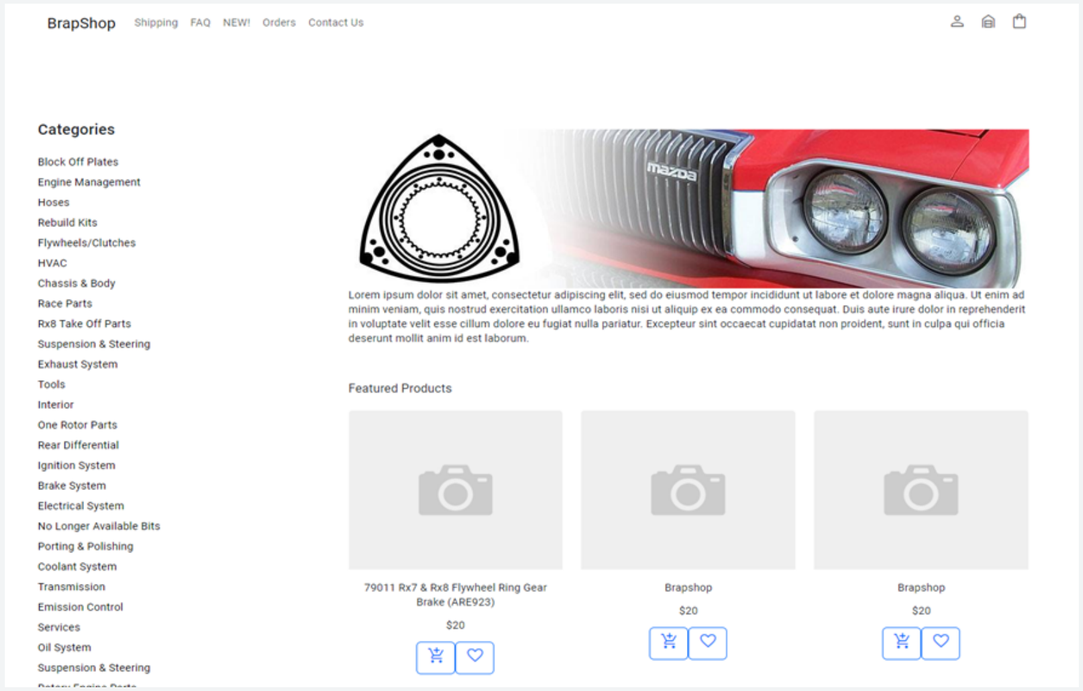
```
var builder = WebApplication.CreateBuilder(args);

var connectionStrings = new Dictionary<string, string>()
{
    { "identity", builder.Configuration.GetConnectionString("IdentityDBConnection") },
    { "data", builder.Configuration.GetConnectionString("DataDBConnection") }

};

builder.Services.AddDbContext<AppIdentityDbContext>(options =>
    options.UseSqlServer(connectionStrings["identity"]));
builder.Services.AddControllers();
builder.Services.AddScoped < ICartRepository, CartRepository > ();
builder.Services.AddScoped < IUserRepository, UserRepository > ();
builder.Services.AddScoped<ITokenService, TokenService>();
builder.Services.AddScoped<IUserService, UserService>();
builder.Services.AddScoped<ProductService>();
builder.Services.AddScoped<CategoryService>();
builder.Services.AddScoped<CartService>();
builder.Services.AddCors(options =>
{
    options.AddDefaultPolicy(builder =>
    {
        builder.AllowAnyOrigin()
               .AllowAnyMethod()
               .AllowAnyHeader();
    });
});

var app = builder.Build();

app.UseAuthorization();
app.MapControllers();
app.UseCors();

app.Run();
```

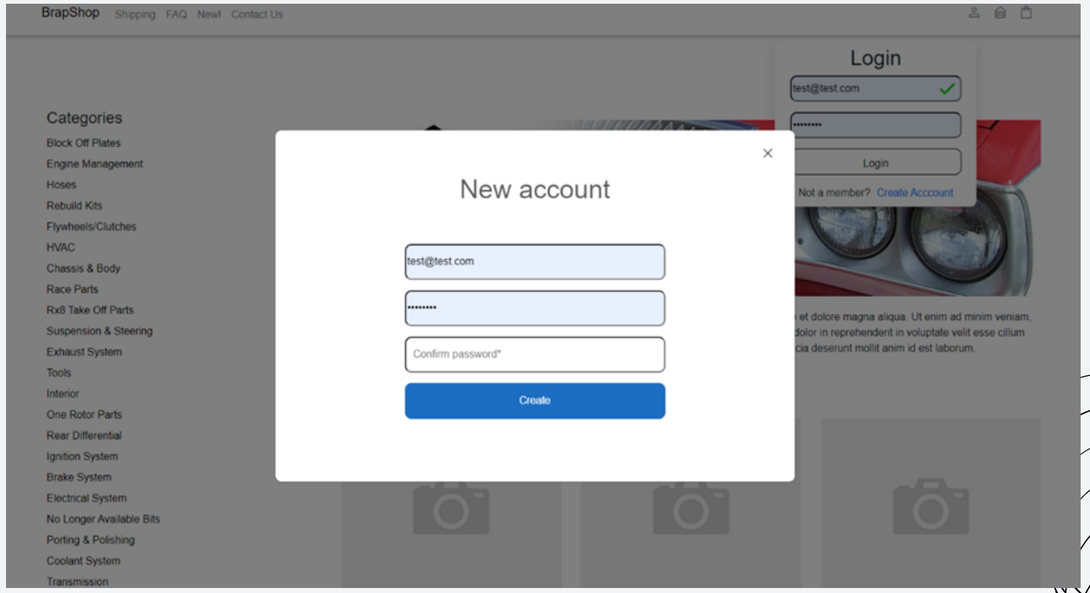
VS

## Blazor

```
var builder = WebApplication.CreateBuilder(args);

var connectionStrings = new Dictionary<string, string>()
{
    { "identity", builder.Configuration.GetConnectionString("IdentityDBConnection") },
    { "data", builder.Configuration.GetConnectionString("DataDBConnection") }

};

builder.Services.AddDbContext<AppIdentityDbContext>(options =>
    options.UseSqlServer(connectionStrings["identity"]));
builder.Services.AddDatabaseDeveloperPageExceptionFilter();
builder.Services.AddDefaultIdentity<ApplicationUser>(options => options.SignIn.RequireConfirmedAccount = true)
    .AddRoles<IdentityRole>()
    .AddEntityFrameworkStores<AppIdentityDbContext>().AddDefaultUI();
builder.Services.AddRazorPages();
builder.Services.AddServerSideBlazor();
builder.Services.AddScoped<AuthenticationStateProvider, RevalidatingIdentityAuthenticationStateProvider<IdentityUser>>();
builder.Services.AddScoped<ProductService>();
builder.Services.AddScoped<CategoryService>();
builder.Services.AddScoped<CartService>();
builder.Services.AddScoped<ICartRepository, CartRepository>();
builder.Services.AddScoped<IUserRepository, UserRepository>();
builder.Services
    .AddBlazorise(options =>
    {
        options.Immediate = true;
    })
    .AddBootstrapProviders()
    .AddFontAwesomeIcons();

var app = builder.Build();

if (app.Environment.IsDevelopment())
{
    app.UseMigrationsEndPoint();
}
else
{
    app.UseExceptionHandler("/Error");
    app.UseHsts();
}

app.UseMiddleware<BlazorCookieLoginMiddleware>();
app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();
app.UseAuthentication();
app.UseAuthorization();
app.MapControllers();
app.MapBlazorHub();
app.MapFallbackToPage("/_Host");

app.Run();
```

# Własna aplikacja wykorzystana w pracy



# Możliwości dalszego rozwoju

W przyszłości można rozszerzyć pracę nad różnicami między tymi, jak widać, mocno odmiennymi podejściami, o badania nad wydajnością obu aplikacji bądź dodać do porównania również aplikację opartą o architekturę mikroserwisów.