Aplikacje WWW
(studia niestacjonarne)
Laboratorium 5

Service Binding, Identity, User registration, Controllers & Views

1. Service Binding (Service registration), Depedency Injection Container

Posiadając utworzone na poprzednich zajęciach usługi (services) możemy powiązac abstrakcje (interfejsy) z konkretną implementacją (klasy). W naszym przypadku są to interfejsy usług z implementującymi je klasami. Należy to wykonać w pliku ***Startup.cs***, analogicznie jak w przypadku projektu ***SchoolRegister.Tests.***

```
public void ConfigureServices (IServiceCollection services) {
    services.AddAutoMapper (typeof (Startup));
    services.AddDbContext<ApplicationDbContext> (options =>
        options.UseSqlServer (Configuration.GetConnectionString ("DefaultConnection")) //here you ca
    );
    services.AddDatabaseDeveloperPageExceptionFilter ();
    services.AddDefaultIdentity<User> (options => options.SignIn.RequireConfirmedAccount = false)
        .AddRoles<Role> ()
        .AddRoleManager<RoleManager<Role>> ()
        .AddUserManager<UserManager<User>> ()
        .AddEntityFrameworkStores<ApplicationDbContext> ();
    services.AddTransient (typeof (ILogger), typeof (Logger<Startup>));
    services.AddScoped<ISubjectService, SubjectService> ();
    services.AddScoped<IEmailSender, EmailSenderService>();
    services.AddScoped<IEmailSenderService, EmailSenderService>();
    services.AddScoped<IGradeService, GradeService>();
    services.AddScoped<IGroupService, GroupService>();
    services.AddScoped<IStudentService, StudentService>();
    services.AddScoped<ITeacherService, TeacherService>();
    services.AddScoped ((serviceProvider) => {
        var config = serviceProvider.GetRequiredService<IConfiguration> ();
        return new SmtpClient () {
            Host = config.GetValue<string> ("Email:Smtp:Host"),
                Port = config.GetValue<int> ("Email:Smtp:Port"),
                EnableSsl = true,
                DeliveryMethod = SmtpDeliveryMethod.Network,
                UseDefaultCredentials = false,
                Credentials = new NetworkCredential (
                    config.GetValue<string> ("Email:Smtp:Username"),
                    config.GetValue<string> ("Email:Smtp:Password")
                )
        };
    });
    services.AddControllersWithViews ();
}
```

W procesie bindowania usług możemy wykorzystać poniższe metody definiujące sposób ich tworzenia:

- ***AddTransient*** – metoda rejestrująca usługę w trybie transient. Obiekty usług zarejestrowanych w ten sposób będą tworzone dla każdego użycia, inaczej mówiąc: Mechanizm stworzy nowy obiekt w pamięci, a następnie zwróci lub przekaże referencję do obiektu.
- ***AddScoped*** – metoda rejestrująca usługę w trybie scoped. Obiekty usług zarejestrowanych w ten sposób będą tworzone na nowo w obrębie każdego żądania HTTP. Jeśli w obszarze jednego żądania HTTP wielokrotnie poprosimy o dany serwis, zostanie nam zwrócona referencja do tego samego obiektu w pamięci.
- ***AddSingleton*** – metoda rejestrująca usługę w trybie singleton. Obiekty usług zarejestrowanych w ten sposób będą tworzone jedynie przy pierwszym użyciu. W przypadku kolejnych zwrócona nam zostanie referencja do tego samego obiektu w pamięci.

Podobnie jak w przypadku testów jednostkowych, aby nasza usługa wysyłająca maila działała poprawnie proszę podać dane wrażliwe w pliku ***appsettings.json***

```
{
  "buildOptions": {
    "copyToOutput": {
      "include": [ "appsettings.Development.json" ]
    }
  },
  "ConnectionStrings": {
    "DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database={Imie_Nazwisko}AppDb;Tru
sted_Connection=True;MultipleActiveResultSets=true"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "Email": {
    "Smtp": {
      "Host": "{host_domain_name}",
      "Port": 587,
      "Username": "{username}",
      "Password": "{password}"
    }
  },
  "AllowedHosts": "*"
}
```

2. ASP.NET Identity - customization

ASP.NET Identity jest frameworkiem umożliwiającym szybkie uwierzytelnianie (Authentication + Autorization) użytkowników w aplikacjach ASP.NET. Framework wspiera: role, custom data stores, indywidualne bazy danych, OAuth, AD, Azure AD. Umożliwia dwuetapową autentykację (two-factor authentication). Składa się następujących typów komponentów:

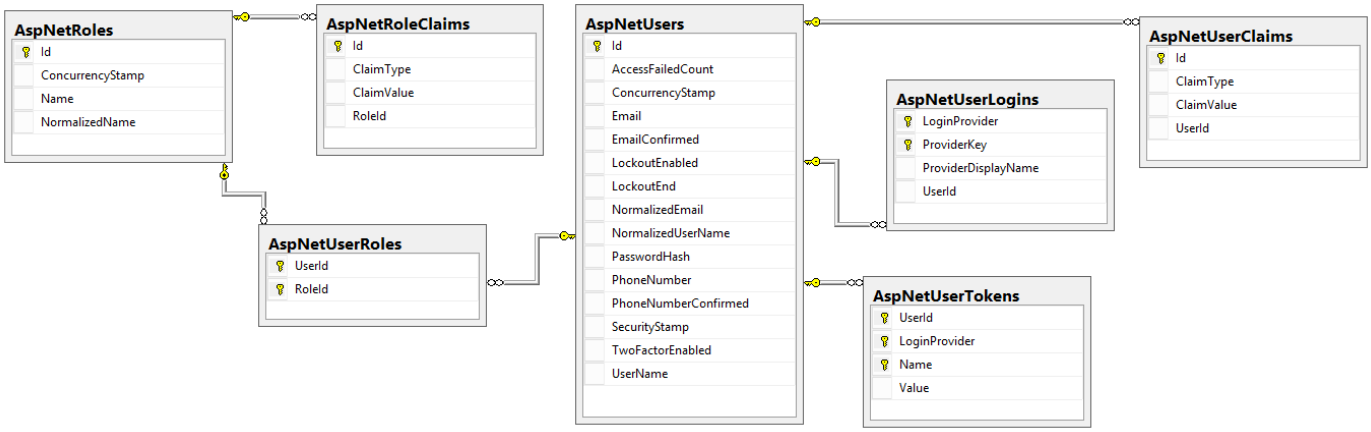
- Managers
- Stores

Managers:

- Są to klasy nie posiadające informacji w jaki sposób dane użytkownika są przechowywane.
- Wykonują następujące operacje:
  - rejestracja nowych użytkowników,
  - walidacja poświadczeń,
  - pobieranie informacji o użytkowniku,
  - tworzenie oraz edycja ról,
  - generowanie user claim.
- Przykłady: ***SignInManager***, ***RoleManager***, ***UserManager***

Stores:

- Są to klasy odpowiedzialne za dostęp do danych oraz CRUD,
- Domyślnie wykorzystywane jest podejście EF Code First aby wygenerować schemat tabel w bazie danych (np. SQL Server, MySQL, PostgesSQL),
- Istnieją implementacje dla baz NoSQL (np. RavenDB czy MongoDB)
- Przykład: ***UserStore***



Lokalna autentykacja użytkowników – wykorzystuje jedynie (bez zewnętrznych providerów) bazę danych aplikacji w celu dokonania uwierzytelniania. Konfigurowalne / rozszerzalne – komponenty frameworka umożliwiają rozszerzanie (dziedziczenie) domyślnych klas. Identity umożliwia również autentykację przy pomocy zewnętrznych serwisów (third party providers) przy pomocy serwerów OAuth. OAuth jest protokołem umożliwiającym dostęp do zasobów zewnętrznych serwisów / providerów (Google, Facebook, Twitter, Amazon, Microsoft) bez dostępu do samego serwisu – oczywiście przy zgodzie użytkownika.

W raz nadejściem frameworka 2.1, który wprowadził biblioteki razor, całe Identity zostało przeniesione do osobnej bliblioteki razora (Razor Library). Stąd aby dokonać jej customizacji należy wykonać następujące kroki:

1) Proszę zainstalować globalnie tzw. scaffolder, do generowania kodu o nazwie ***dotnet-aspnet-codegenerator***

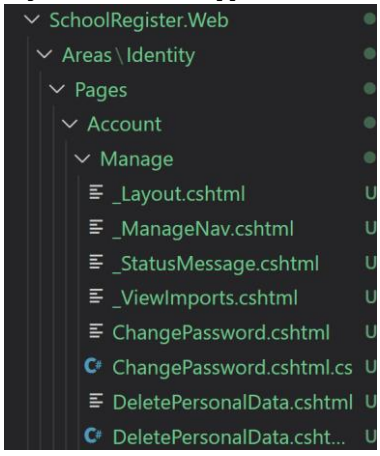
```
dotnet tool install -g dotnet-aspnet-codegenerator --version 5.0.2
```

2) Następnie proszę zainstalować poniższą biblotekę w dla projektu **SchoolRegister.Web**

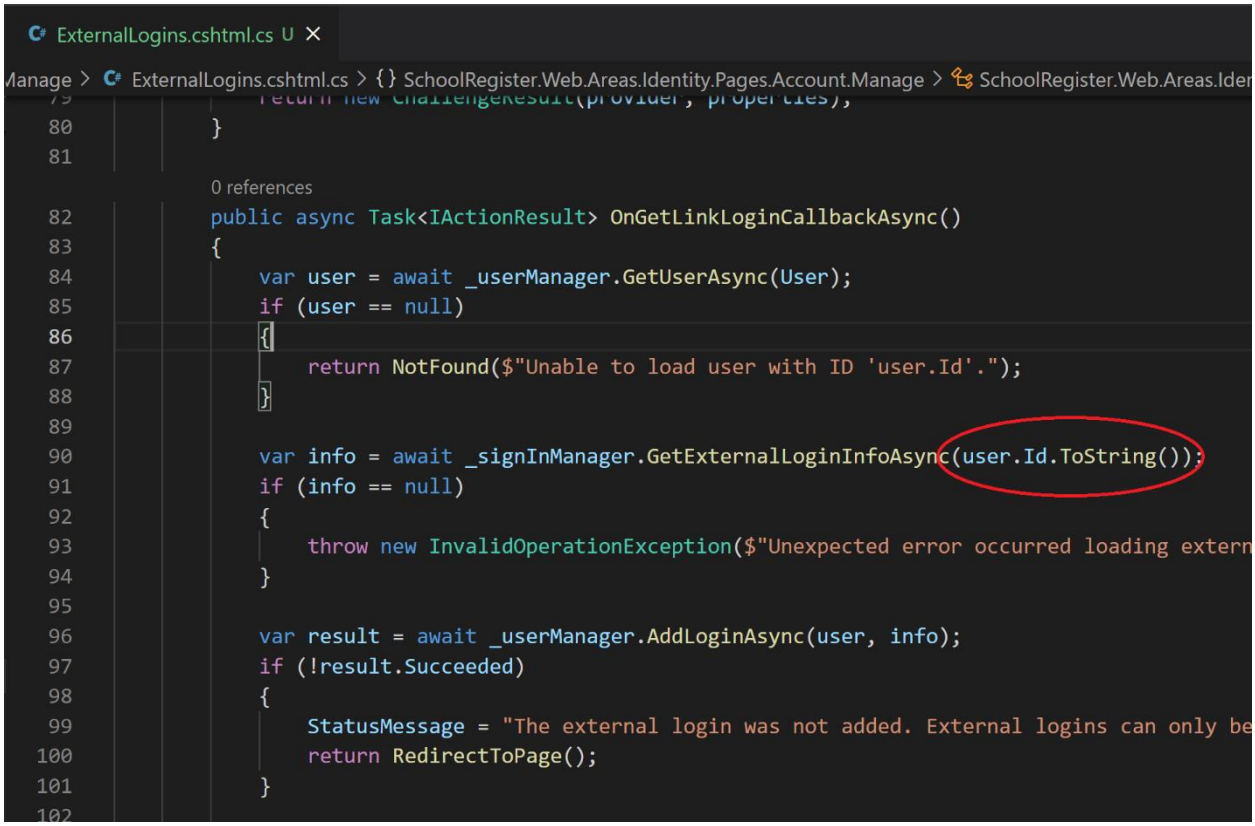
```
dotnet add .\SchoolRegister.Web\SchoolRegister.Web.csproj package Microsoft.VisualStudio.Web.CodeGeneration.Design
--version 5.0.2
```

3) W celu scaffoldingu wszystkich ***Razor Pages*** we frameworku ***Identity*** proszę w terminalu wykonać poniższy kod:

```
dotnet aspnet-codegenerator identity --project .\SchoolRegister.Web\SchoolRegister.Web.csproj -dc
SchoolRegister.DAL.EF.ApplicationDbContext
```



4) Następnie proszę w pliku ***SchoolRegister.Web/Areas/Identity/Account/Manage/ExtenalLogins.cshtml.cs*** zmienic poniższą linijkę kodu.



5) Nastepnie proszę skompilować i uruchomic aplikację.

```
dotnet run -p .\SchoolRegister.Web\SchoolRegister.Web.csproj
```

3. User registration

Posiadając już działającą aplikację należy zmodyfikować funkcjonalność rejestracji nowych użytkowników. Powinna ona być dostępna jedynie dla administratorów systemu (rola "Admin"). Rejestracja użytkowników w obecnej formie zostanie wyłączona. W tym celu należy zmodyfikować predefiniowane elementy biblioteki Identity.

1) W pierwszym kroku proszę przy pomocy Azure Data Studio wykonać poniższy skrypt SQL na utworzonej w porzednich laboratoriach bazie danych:

```
-- DELETE ALL DATA FROM TABLES
DELETE FROM [dbo].[Grades];
DELETE FROM [dbo].[AspNetUserTokens];
DELETE FROM [dbo].[SubjectGroups];
DELETE FROM [dbo].[Subjects];
DELETE FROM [dbo].[AspNetUserRoles];
DELETE FROM [dbo].[AspNetUsers];
DELETE FROM [dbo].[Groups];
DELETE FROM [dbo].[AspNetRoles];
GO
SET IDENTITY_INSERT [dbo].[AspNetRoles] ON
INSERT [dbo].[AspNetRoles] ([Id], [Name], [NormalizedName], [ConcurrencyStamp], [RoleValue]) VALUES
(3, N'Teacher', N'TEACHER', N'097087b9-f582-442c-8c98-7c5a7795098f', 3)
INSERT [dbo].[AspNetRoles] ([Id], [Name], [NormalizedName], [ConcurrencyStamp], [RoleValue]) VALUES
(1, N'Student', N'STUDENT', N'ae41df1a-6e54-465f-b52e-847fcc0029c7', 1)
INSERT [dbo].[AspNetRoles] ([Id], [Name], [NormalizedName], [ConcurrencyStamp], [RoleValue]) VALUES
(2, N'Parent', N'PARENT', N'1a7eb639-3a74-4041-a6b0-100666902232', 2)
INSERT [dbo].[AspNetRoles] ([Id], [Name], [NormalizedName], [ConcurrencyStamp], [RoleValue]) VALUES
(4, N'Admin', N'ADMIN', N'1a7eb639-3a74-4041-a6b0-100666902255', 4)
SET IDENTITY_INSERT [dbo].[AspNetRoles] OFF
GO
SET IDENTITY_INSERT [dbo].[Groups] ON
INSERT [dbo].[Groups] ([Id], [Name]) VALUES (1, N'IO')
```

```
INSERT [dbo].[Groups] ([Id], [Name]) VALUES (2, N'PAI')
INSERT [dbo].[Groups] ([Id], [Name]) VALUES (3, N'AIP_Erasmus+')
SET IDENTITY_INSERT [dbo].[Groups] OFF
GO
SET IDENTITY_INSERT [dbo].[AspNetUsers] ON
--PASSWORD FOR ALL USERS: User1234
INSERT [dbo].[AspNetUsers] ([Id], [UserName], [NormalizedUserName], [Email], [NormalizedEmail],
[EmailConfirmed], [PasswordHash], [SecurityStamp], [ConcurrencyStamp], [PhoneNumber],
[PhoneNumberConfirmed], [TwoFactorEnabled], [LockoutEnd], [LockoutEnabled], [AccessFailedCount],
[FirstName], [LastName], [RegistrationDate], [UserType], [GroupId], [ParentId], [Title]) VALUES (1,
N't1@eg.eg', N'T1@EG.EG', N't1@eg.eg', N'T1@EG.EG', 0,
N'AQAAAAEAACcQAAAAEL5luXXr+auYHroKazQLIljJqou2ERCuex8Hwco9xkjwBuE3y7IkN0IMf2XX5YGylg==',
N'2ZS2NADB4BES3WUMYKTDGU72KCZ2W7VP', N'efb5fcaf-87da-4010-9a1d-57677910e0fa', NULL, 0, 0, NULL, 1, 0,
N'Adam', N'Bednarski', CAST(N'2010-01-01T00:00:00.0000000' AS DateTime2), 3, NULL, NULL, N'mgr inż.')
INSERT [dbo].[AspNetUsers] ([Id], [UserName], [NormalizedUserName], [Email], [NormalizedEmail],
[EmailConfirmed], [PasswordHash], [SecurityStamp], [ConcurrencyStamp], [PhoneNumber],
[PhoneNumberConfirmed], [TwoFactorEnabled], [LockoutEnd], [LockoutEnabled], [AccessFailedCount],
[FirstName], [LastName], [RegistrationDate], [UserType], [GroupId], [ParentId], [Title]) VALUES (2,
N't2@eg.eg', N'T2@EG.EG', N't2@eg.eg', N'T2@EG.EG', 0,
N'AQAAAAEAACcQAAAAEL5luXXr+auYHroKazQLIljJqou2ERCuex8Hwco9xkjwBuE3y7IkN0IMf2XX5YGylg==',
N'B6E24AVVZ4PGOYSXOCBCZG7JDFQQDRJU', N'da94c1dc-9e31-4e0d-b9b5-2d11e042b1a5', NULL, 0, 0, NULL, 1, 0,
N'Jan', N'Nowak', CAST(N'2010-11-12T00:00:00.0000000' AS DateTime2), 3, NULL, NULL, N'mgr')
INSERT [dbo].[AspNetUsers] ([Id], [UserName], [NormalizedUserName], [Email], [NormalizedEmail],
[EmailConfirmed], [PasswordHash], [SecurityStamp], [ConcurrencyStamp], [PhoneNumber],
[PhoneNumberConfirmed], [TwoFactorEnabled], [LockoutEnd], [LockoutEnabled], [AccessFailedCount],
[FirstName], [LastName], [RegistrationDate], [UserType], [GroupId], [ParentId], [Title]) VALUES (3,
N'p1@eg.eg', N'P1@EG.EG', N'p1@eg.eg', N'P1@EG.EG', 0,
N'AQAAAAEAACcQAAAAEL5luXXr+auYHroKazQLIljJqou2ERCuex8Hwco9xkjwBuE3y7IkN0IMf2XX5YGylg==',
N'VQFU65V4PHFHHKBJAPLHE7ECGJ562XZ2', N'1f4e7556-0970-4d3b-bce9-e231cb9eb988', NULL, 0, 0, NULL, 1, 0,
N'Zbigniew', N'Kowalski', CAST(N'2014-03-20T00:00:00.0000000' AS DateTime2), 2, NULL, NULL, NULL)
INSERT [dbo].[AspNetUsers] ([Id], [UserName], [NormalizedUserName], [Email], [NormalizedEmail],
[EmailConfirmed], [PasswordHash], [SecurityStamp], [ConcurrencyStamp], [PhoneNumber],
[PhoneNumberConfirmed], [TwoFactorEnabled], [LockoutEnd], [LockoutEnabled], [AccessFailedCount],
[FirstName], [LastName], [RegistrationDate], [UserType], [GroupId], [ParentId], [Title]) VALUES (4,
N'p2@eg.eg', N'P2@EG.EG', N'p2@eg.eg', N'P2@EG.EG', 0,
N'AQAAAAEAACcQAAAAEL5luXXr+auYHroKazQLIljJqou2ERCuex8Hwco9xkjwBuE3y7IkN0IMf2XX5YGylg==',
N'VQFU65V4PHFHHKBJAPLHE7ECGJ562XZ2', N'1f4e7556-0970-4d3b-bce9-e231cb9eb988', NULL, 0, 0, NULL, 1, 0,
N'Zbigniew', N'Stamowski', CAST(N'2014-06-21T00:00:00.0000000' AS DateTime2), 2, NULL, NULL, NULL)
INSERT [dbo].[AspNetUsers] ([Id], [UserName], [NormalizedUserName], [Email], [NormalizedEmail],
[EmailConfirmed], [PasswordHash], [SecurityStamp], [ConcurrencyStamp], [PhoneNumber],
[PhoneNumberConfirmed], [TwoFactorEnabled], [LockoutEnd], [LockoutEnabled], [AccessFailedCount],
[FirstName], [LastName], [RegistrationDate], [UserType], [GroupId], [ParentId], [Title]) VALUES (5,
N's1@eg.eg', N'S1@EG.EG', N's1@eg.eg', N'S1@EG.EG', 0,
N'AQAAAAEAACcQAAAAEL5luXXr+auYHroKazQLIljJqou2ERCuex8Hwco9xkjwBuE3y7IkN0IMf2XX5YGylg==',
N'EYKWZQO43N3MUR74O4AQAKCZ3BKSA3NX', N'd4ed1021-52bc-4c47-972b-6ecc97b6e58f', NULL, 0, 0, NULL, 1, 0,
N'Tomasz', N'Kowalski', CAST(N'2016-05-11T00:00:00.0000000' AS DateTime2), 1, 1, 3, NULL)
INSERT [dbo].[AspNetUsers] ([Id], [UserName], [NormalizedUserName], [Email], [NormalizedEmail],
[EmailConfirmed], [PasswordHash], [SecurityStamp], [ConcurrencyStamp], [PhoneNumber],
[PhoneNumberConfirmed], [TwoFactorEnabled], [LockoutEnd], [LockoutEnabled], [AccessFailedCount],
[FirstName], [LastName], [RegistrationDate], [UserType], [GroupId], [ParentId], [Title]) VALUES (6,
N's2@eg.eg', N'S2@EG.EG', N's2@eg.eg', N'S2@EG.EG', 0,
N'AQAAAAEAACcQAAAAEL5luXXr+auYHroKazQLIljJqou2ERCuex8Hwco9xkjwBuE3y7IkN0IMf2XX5YGylg==',
N'4KCBZNJAASKS4WUOQJEGNBSPZ6KRPSAX', N'478b51c7-0055-4574-8ffb-7a876765bfcf', NULL, 0, 0, NULL, 1, 0,
N'Krzysztof', N'Kowalski', CAST(N'2015-09-18T00:00:00.0000000' AS DateTime2), 1, 1, 3, NULL)
INSERT [dbo].[AspNetUsers] ([Id], [UserName], [NormalizedUserName], [Email], [NormalizedEmail],
[EmailConfirmed], [PasswordHash], [SecurityStamp], [ConcurrencyStamp], [PhoneNumber],
[PhoneNumberConfirmed], [TwoFactorEnabled], [LockoutEnd], [LockoutEnabled], [AccessFailedCount],
[FirstName], [LastName], [RegistrationDate], [UserType], [GroupId], [ParentId], [Title]) VALUES (7,
N's3@eg.eg', N'S3@EG.EG', N's3@eg.eg', N'S3@EG.EG', 0,
N'AQAAAAEAACcQAAAAEL5luXXr+auYHroKazQLIljJqou2ERCuex8Hwco9xkjwBuE3y7IkN0IMf2XX5YGylg==',
N'4KCBZNJAASKS4WUOQJEGNBSPZ6KRPSAX', N'478b51c7-0055-4574-8ffb-7a876765bfcf', NULL, 0, 0, NULL, 1, 0,
N'Natalia', N'Kowalska', CAST(N'2017-07-16T00:00:00.0000000' AS DateTime2), 1, 2, 3, NULL)
INSERT [dbo].[AspNetUsers] ([Id], [UserName], [NormalizedUserName], [Email], [NormalizedEmail],
[EmailConfirmed], [PasswordHash], [SecurityStamp], [ConcurrencyStamp], [PhoneNumber],
[PhoneNumberConfirmed], [TwoFactorEnabled], [LockoutEnd], [LockoutEnabled], [AccessFailedCount],
[FirstName], [LastName], [RegistrationDate], [UserType], [GroupId], [ParentId], [Title]) VALUES (8,
N's4@eg.eg', N'S4@EG.EG', N's4@eg.eg', N'S4@EG.EG', 0,
N'AQAAAAEAACcQAAAAEL5luXXr+auYHroKazQLIljJqou2ERCuex8Hwco9xkjwBuE3y7IkN0IMf2XX5YGylg==',
N'4KCBZNJAASKS4WUOQJEGNBSPZ6KRPSAX', N'478b51c7-0055-4574-8ffb-7a876765bfcf', NULL, 0, 0, NULL, 1, 0,
N'Magdalena', N'Stamowska', CAST(N'2018-05-14T00:00:00.0000000' AS DateTime2), 1, 2, 4, NULL)
INSERT [dbo].[AspNetUsers] ([Id], [UserName], [NormalizedUserName], [Email], [NormalizedEmail],
[EmailConfirmed], [PasswordHash], [SecurityStamp], [ConcurrencyStamp], [PhoneNumber],
[PhoneNumberConfirmed], [TwoFactorEnabled], [LockoutEnd], [LockoutEnabled], [AccessFailedCount],
[FirstName], [LastName], [RegistrationDate], [UserType], [GroupId], [ParentId], [Title]) VALUES (9,
N's5@eg.eg', N'S5@EG.EG', N's5@eg.eg', N'S5@EG.EG', 0,
N'AQAAAAEAACcQAAAAEL5luXXr+auYHroKazQLIljJqou2ERCuex8Hwco9xkjwBuE3y7IkN0IMf2XX5YGylg==',
N'4KCBZNJAASKS4WUOQJEGNBSPZ6KRPSAX', N'478b51c7-0055-4574-8ffb-7a876765bfcf', NULL, 0, 0, NULL, 1, 0,
N'Janusz', N'Stamowski', CAST(N'2019-02-19T00:00:00.0000000' AS DateTime2), 1, 3, 4, NULL)
```

```
INSERT [dbo].[AspNetUsers] ([Id], [UserName], [NormalizedUserName], [Email], [NormalizedEmail],
[EmailConfirmed], [PasswordHash], [SecurityStamp], [ConcurrencyStamp], [PhoneNumber],
[PhoneNumberConfirmed], [TwoFactorEnabled], [LockoutEnd], [LockoutEnabled], [AccessFailedCount],
[FirstName], [LastName], [RegistrationDate], [UserType], [GroupId], [ParentId], [Title]) VALUES (10,
N's6@eg.eg', N'S6@EG.EG', N's6@eg.eg', N'S6@EG.EG', 0,
N'AQAAAAEAACcQAAAAEL5luXXr+auYHroKazQLIljJqou2ERCuex8Hwco9xkjwBuE3y7IkN0IMf2XX5YGylg==',
N'4KCBZNJAASKS4WUOQJEGNBSPZ6KRPSAX', N'478b51c7-0055-4574-8ffb-7a876765bfcf', NULL, 0, 0, NULL, 1, 0,
N'Krystian', N'Stamowski', CAST(N'2013-02-18T00:00:00.0000000' AS DateTime2), 1, 3, 4, NULL)
INSERT [dbo].[AspNetUsers] ([Id], [UserName], [NormalizedUserName], [Email], [NormalizedEmail],
[EmailConfirmed], [PasswordHash], [SecurityStamp], [ConcurrencyStamp], [PhoneNumber],
[PhoneNumberConfirmed], [TwoFactorEnabled], [LockoutEnd], [LockoutEnabled], [AccessFailedCount],
[FirstName], [LastName], [RegistrationDate], [UserType], [GroupId], [ParentId], [Title]) VALUES (11,
N'a1@eg.eg', N'A1@EG.EG', N'a1@eg.eg', N'A1@EG.EG', 0,
N'AQAAAAEAACcQAAAAEL5luXXr+auYHroKazQLIljJqou2ERCuex8Hwco9xkjwBuE3y7IkN0IMf2XX5YGylg==',
N'5X4DP3KINUFVF5APSXYLC3P3S7AJIR6L', N'90f466f7-d4a9-4cd8-8a1f-f3c18bc36ab9', NULL, 0, 0, NULL, 1, 0,
N'Jacek', N'Miłowski', CAST(N'2019-04-20T00:00:00.0000000' AS DateTime2), 0, NULL, NULL, NULL)
SET IDENTITY_INSERT [dbo].[AspNetUsers] OFF
GO
INSERT [dbo].[AspNetUserRoles] ([UserId], [RoleId]) VALUES (1, 3)
INSERT [dbo].[AspNetUserRoles] ([UserId], [RoleId]) VALUES (2, 3)
INSERT [dbo].[AspNetUserRoles] ([UserId], [RoleId]) VALUES (3, 2)
INSERT [dbo].[AspNetUserRoles] ([UserId], [RoleId]) VALUES (4, 2)
INSERT [dbo].[AspNetUserRoles] ([UserId], [RoleId]) VALUES (5, 1)
INSERT [dbo].[AspNetUserRoles] ([UserId], [RoleId]) VALUES (6, 1)
INSERT [dbo].[AspNetUserRoles] ([UserId], [RoleId]) VALUES (7, 1)
INSERT [dbo].[AspNetUserRoles] ([UserId], [RoleId]) VALUES (8, 1)
INSERT [dbo].[AspNetUserRoles] ([UserId], [RoleId]) VALUES (9, 1)
INSERT [dbo].[AspNetUserRoles] ([UserId], [RoleId]) VALUES (10, 1)
INSERT [dbo].[AspNetUserRoles] ([UserId], [RoleId]) VALUES (11, 4)
GO
SET IDENTITY_INSERT [dbo].[Subjects] ON
INSERT [dbo].[Subjects] ([Id], [Name], [Description], [TeacherId]) VALUES (1, N'Aplikacje WWW',
N'Aplikacje webowe', 1)
INSERT [dbo].[Subjects] ([Id], [Name], [Description], [TeacherId]) VALUES (2, N'Programowanie
obiektowe', N'Programowanie obiektowe jest przedmiotem realizującym przykłady programowanie
obiektowego', 1)
INSERT [dbo].[Subjects] ([Id], [Name], [Description], [TeacherId]) VALUES (3, N'Advanced Internet
Programming', N'Advanced Internet Programming is a course for ERASMUS+ students', 2)
INSERT [dbo].[Subjects] ([Id], [Name], [Description], [TeacherId]) VALUES (4, N'Administracja
Intenetowymi Systemami Baz Danych', N'Administracja Intenetowymi Systemami Baz Danych jest kontynuacją
przedmiotu Bazy danych na studiach stacjonarnych I-go stopnia spec. PAI', 2)
SET IDENTITY_INSERT [dbo].[Subjects] OFF
GO
INSERT [dbo].[SubjectGroups] ([SubjectId], [GroupId]) VALUES (1, 1)
INSERT [dbo].[SubjectGroups] ([SubjectId], [GroupId]) VALUES (1, 2)
INSERT [dbo].[SubjectGroups] ([SubjectId], [GroupId]) VALUES (2, 1)
INSERT [dbo].[SubjectGroups] ([SubjectId], [GroupId]) VALUES (2, 2)
INSERT [dbo].[SubjectGroups] ([SubjectId], [GroupId]) VALUES (2, 3)
INSERT [dbo].[SubjectGroups] ([SubjectId], [GroupId]) VALUES (3, 3)
INSERT [dbo].[SubjectGroups] ([SubjectId], [GroupId]) VALUES (4, 1)
INSERT [dbo].[SubjectGroups] ([SubjectId], [GroupId]) VALUES (4, 2)
INSERT [dbo].[SubjectGroups] ([SubjectId], [GroupId]) VALUES (4, 3)
GO
INSERT [dbo].[Grades] ([DateOfIssue], [GradeValue], [SubjectId], [StudentId]) VALUES (CAST(N'2019-03-
31T17:46:38.1887824' AS DateTime2), 4, 1, 5)
GO
```

2) W następnym kroku w projekcie Web proszę przejść pod poniższą scieżkę i otworzyć poniższy plik:

SchoolRegister.Web/Areas/Identity/Pages/Account/Manage/ManageNavPages.cs

Następnie proszę zmodyfikować go do poniższej postaci:

```
public static class ManageNavPages {

        public static string Index => "Index";

        public static string Email => "Email";

        public static string ChangePassword => "ChangePassword";

        public static string DownloadPersonalData => "DownloadPersonalData";
```

```
        public static string DeletePersonalData => "DeletePersonalData";

        public static string ExternalLogins => "ExternalLogins";

        public static string PersonalData => "PersonalData";

        public static string TwoFactorAuthentication => "TwoFactorAuthentication";

        public static string RegisterNewUsers => "RegisterNewUsers";

        public static string IndexNavClass (ViewContext viewContext) => PageNavClass (viewContext, Index);

        public static string EmailNavClass (ViewContext viewContext) => PageNavClass (viewContext, Email);

        public static string ChangePasswordNavClass (ViewContext viewContext) => PageNavClass (viewContext, ChangePassword);

        public static string DownloadPersonalDataNavClass (ViewContext viewContext) => PageNavClass (viewContext, DownloadPersonalData);

        public static string DeletePersonalDataNavClass (ViewContext viewContext) => PageNavClass (viewContext, DeletePersonalData);

        public static string ExternalLoginsNavClass (ViewContext viewContext) => PageNavClass (viewContext, ExternalLogins);

        public static string PersonalDataNavClass (ViewContext viewContext) => PageNavClass (viewContext, PersonalData);

        public static string TwoFactorAuthenticationNavClass (ViewContext viewContext) => PageNavClass (viewContext, TwoFactorAuthentication);

        public static string RegisterNewUsersNavClass (ViewContext viewContext) => PageNavClass (viewContext, RegisterNewUsers);

        private static string PageNavClass (ViewContext viewContext, string page) {

            var activePage = viewContext.ViewData["ActivePage"] as string ??

                System.IO.Path.GetFileNameWithoutExtension (viewContext.ActionDescriptor.DisplayName);

            return string.Equals (activePage, page, StringComparison.OrdinalIgnoreCase) ? "active" : null;

        }

}
```

3) W folderze ***SchoolRegister.Web/Areas/Identity/Pages/Account/Manage/*** proszę stworzyć nowy widok o nazwie ***RegisterNewUsers.cshtml***

```
@page
@model RegisterNewUsersDataModel
@{
    ViewData["Title"] = "Register new users";
    ViewData["ActivePage"] = ManageNavPages.RegisterNewUsers;
}
<h4>@ViewData["Title"]</h4>
<partial name="_StatusMessage" for="StatusMessage" />
<div class="row">
    <div class="col-md-4">
        <form method="post" >
            <h4>Create a new account.</h4>
            <hr />
            <div asp-validation-summary="All" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Input.Email"></label>
                <input asp-for="Input.Email" class="form-control" />
                <span asp-validation-for="Input.Email" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Input.Password"></label>
                <input asp-for="Input.Password" class="form-control" />
                <span asp-validation-for="Input.Password" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Input.ConfirmPassword"></label>
                <input asp-for="Input.ConfirmPassword" class="form-control" />
                <span asp-validation-for="Input.ConfirmPassword" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Input.FirstName"></label>
                <input asp-for="Input.FirstName" class="form-control" />
                <span asp-validation-for="Input.FirstName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Input.LastName"></label>
                <input asp-for="Input.LastName" class="form-control" />
                <span asp-validation-for="Input.LastName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label>Role</label>
```

```
                <select asp-for="Input.RoleId" class="form-control" asp-
items="@ViewData["Roles"] as IEnumerable<SelectListItem>"></select>
                <span asp-validation-for="Input.RoleId" class="text-danger"></span>
            </div>
            <div class="student-selected">
                <div class="form-group">
                    <label>Group</label>
                    <select asp-for="Input.GroupId" class="form-control" asp-
items="@ViewData["Groups"] as IEnumerable<SelectListItem>"></select>
                    <span asp-validation-for="Input.GroupId" class="text-danger"></span>
                </div>
                <div class="form-group">
                    <label>Parent</label>
                    <select asp-for="Input.ParentId" class="form-control" asp-
items="@ViewData["Parents"] as IEnumerable<SelectListItem>"></select>
                    <span asp-validation-for="Input.ParentId" class="text-danger"></span>
                </div>
            </div>
            <div class="teacher-selected">
                <div class="form-group">
                    <label asp-for="Input.TeacherTitles"></label>
                    <input asp-for="Input.TeacherTitles" class="form-control" />
                    <span asp-validation-for="Input.TeacherTitles" class="text-danger"></span>
                </div>
            </div>
            <button type="submit" class="btn btn-primary">Register</button>
        </form>
    </div>
</div>
@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}
```

4) W kolejnym kroku proszę stworzyć nową klasę w tym samym folderze o nazwie: **RegisterNewUsers.cshtml.cs** o poniższej zawartości:

```
using System;
using System.Linq;
using System.Threading.Tasks;
using AutoMapper;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.Extensions.Logging;
using SchoolRegister.DAL.EF;
using SchoolRegister.Model.DataModels;
using SchoolRegister.ViewModels.VM;

namespace SchoolRegister.Web.Areas.Identity.Pages.Account.Manage
{
    [Authorize(Roles = "Admin")]
    public class RegisterNewUsersDataModel : PageModel
    {
        private readonly UserManager<User> _userManager;
        private readonly ILogger _logger;
        private readonly IMapper _mapper;
        private readonly ApplicationDbContext _dbContext;

        [BindProperty]
        public RegisterNewUserVm Input { get; set; }

        [TempData]
        public string StatusMessage { get; set; }

        public RegisterNewUsersDataModel(UserManager<User> userManager,
            ILogger logger,
            ApplicationDbContext dbContext,
            IMapper mapper)
        {
            _userManager = userManager;
            _logger = logger;
            _dbContext = dbContext;
            _mapper = mapper;
        }
        public IActionResult OnGet()
        {
            ViewData["Roles"] = new SelectList(_dbContext.Roles.Select(t => new
            {
                Text = t.Name,
```

```
                Value = t.Id
            }), "Value", "Text", _dbContext.Roles.FirstOrDefault(x => x.RoleValue == RoleValue.Parent).Id);
            ViewData["Groups"] = new SelectList(_dbContext.Groups.Select(t => new
            {
                Text = t.Name,
                Value = t.Id
            }), "Value", "Text");
            ViewData["Parents"] = new SelectList(_dbContext.Users.OfType<Parent>().Select(t => new
            {
                Text = $"{t.FirstName} {t.LastName}",
                Value = t.Id
            }), "Value", "Text");
            return Page();
        }
        public async Task<IActionResult> OnPostAsync()
        {
            if (ModelState.IsValid)
            {
                var tupleUserRole = CreateUserBasedOnRole(Input);
                var result = await _userManager.CreateAsync(tupleUserRole.Item1, Input.Password);
                if (result.Succeeded)
                {
                    _logger.LogInformation("User created a new account with password.");
                    result = await _userManager.AddToRoleAsync(tupleUserRole.Item1,
                                                               tupleUserRole.Item2.Name);
                    if (result.Succeeded)
                    {
                        OnGet();
                        StatusMessage = "User created";
                        return RedirectToPage();
                    }
                }
                foreach (var error in result.Errors)
                {
                    ModelState.AddModelError(string.Empty, error.Description);
                }
            }
            OnGet();
            return RedirectToPage();
        }
        private Tuple<User, Role> CreateUserBasedOnRole(RegisterNewUserVm inputModel)
        {
            var role = _dbContext.Roles.FirstOrDefault(r => r.Id == inputModel.RoleId);
            if (role == null)
                throw new InvalidOperationException("Role not exists.");
            switch (role.RoleValue)
            {
                case RoleValue.Student:
                    return new Tuple<User, Role>(_mapper.Map<Student>(inputModel), role);
                case RoleValue.Parent:
                    return new Tuple<User, Role>(_mapper.Map<Parent>(inputModel), role);
                case RoleValue.Teacher:
                    return new Tuple<User, Role>(_mapper.Map<Teacher>(inputModel), role);
                case RoleValue.Admin:
                case RoleValue.User:
                    return new Tuple<User, Role>(_mapper.Map<User>(inputModel), role);
                default:
                    return null;
            }
        }
    }
}
```

5) Następnie proszę w projekcie ***SchoolRegister.ViewModels/VM/RegisterNewUserVm.cs*** dodać poniższy kod:

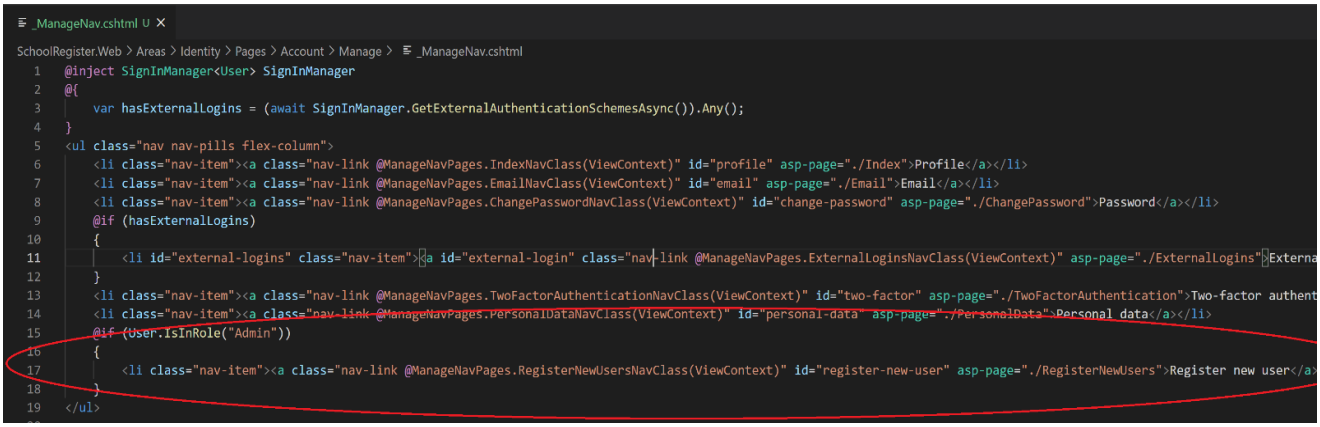
```
public class RegisterNewUserVm
        {
            [Required]
            [EmailAddress]
            [Display(Name = "Email")]
```

```
            public string Email { get; set; }

            [Required]
            [StringLength(100, ErrorMessage = "The {0} must be at least {2} and at max {1} characters
long.", MinimumLength = 6)]
            [DataType(DataType.Password)]
            [Display(Name = "Password")]
            public string Password { get; set; }

            [DataType(DataType.Password)]
            [Display(Name = "Confirm password")]
            [Compare("Password", ErrorMessage = "The password and confirmation password do not
match.")]
            public string ConfirmPassword { get; set; }

            [Required]
            [StringLength(100, ErrorMessage = "The {0} must be at least {2} and at max {1} characters
long.", MinimumLength = 6)]
            [DataType(DataType.Text)]
            [Display(Name = "First name")]
            public string FirstName { get; set; }

            [Required]
            [StringLength(100, ErrorMessage = "The {0} must be at least {2} and at max {1} characters
long.", MinimumLength = 6)]
            [DataType(DataType.Text)]
            [Display(Name = "Last name")]
            public string LastName { get; set; }

            [Required]
            [Display(Name = "Role")]
            public int RoleId { get; set; }

            [Display(Name = "Parent")]

            public int? ParentId { get; set; }

            [Display(Name = "Teacher titles")]
            public string TeacherTitles { get; set; }

            public int? GroupId { get; set; }

        }
```

6) Proszę dodać poniższe mapy AutoMappera ***SchoolRegister.Web/Configuration/Profiles/MainProfile.cs***

```
            CreateMap<RegisterNewUserVm, User>()
                .ForMember(dest => dest.UserName, y => y.MapFrom(src => src.Email))
                .ForMember(dest => dest.RegistrationDate, y => y.MapFrom(src => DateTime.Now));
            CreateMap<RegisterNewUserVm, Parent>()
                .ForMember(dest => dest.UserName, y => y.MapFrom(src => src.Email))
                .ForMember(dest => dest.RegistrationDate, y => y.MapFrom(src => DateTime.Now));
            CreateMap<RegisterNewUserVm, Student>()
                .ForMember(dest => dest.UserName, y => y.MapFrom(src => src.Email))
                .ForMember(dest => dest.RegistrationDate, y => y.MapFrom(src => DateTime.Now));
            CreateMap<RegisterNewUserVm, Teacher>()
                .ForMember(dest => dest.UserName, y => y.MapFrom(src => src.Email))
                .ForMember(dest => dest.RegistrationDate, y => y.MapFrom(src => DateTime.Now))
                .ForMember(dest => dest.Title, y => y.MapFrom(src => src.TeacherTitles));
```

7) Proszę zmienić plik ***SchoolRegister.Web/Areas/Identity/Pages/Account/Manage/_ManageNav.cshtml*** aby wyglądał w następujący sposób:

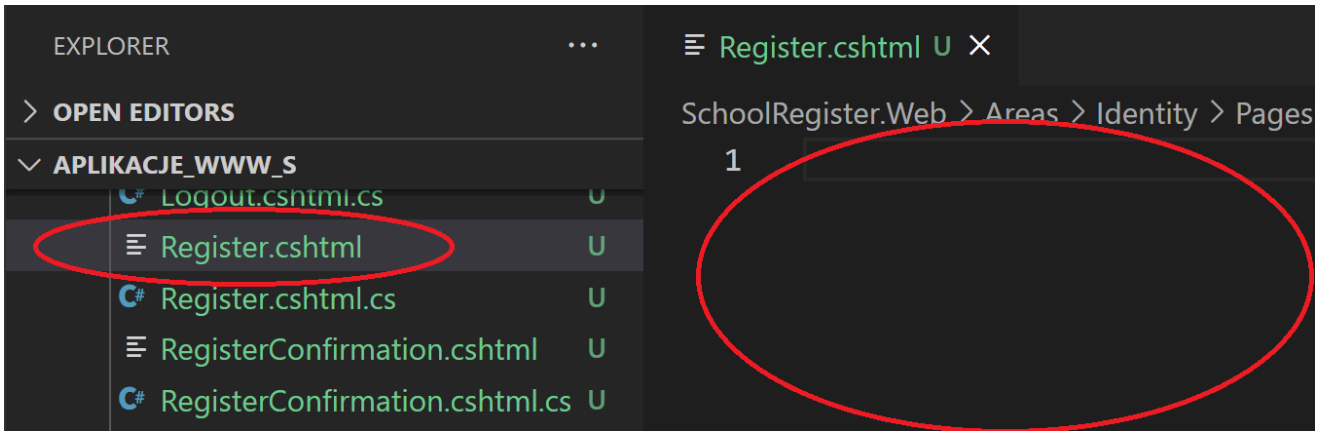


8) W pliku ***SchoolRegister.Web/Areas/Identity/Pages/Account/Register.cshtml.cs*** proszę wkleić poniższy kod:

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace SchoolRegister.Web.Areas.Identity.Pages.Account {
    public class RegisterModel : PageModel {
        public IActionResult OnGet () {
            return NotFound ();
        }
        public IActionResult OnPost () {
            return NotFound ();
        }
    }
}
```

9) Następnie proszę usunąc cały kod z pliku:

SchoolRegister.Web/Areas/Identity/Pages/Account/Register.cshtml



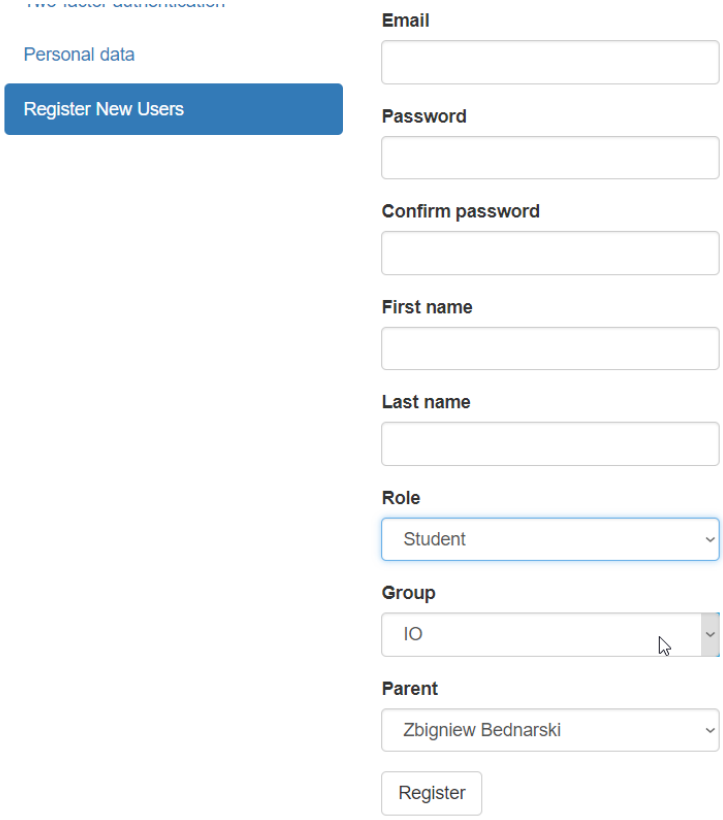
10) W ostatnim kroku proszę dodać poniższy skrypt do pliku:

SchoolRegister.Web/Areas/Identity/Pages/_ValidationScriptsPartial.cshtml

```
<environment include="Development">
    <script src="~/Identity/lib/jquery-
validation/dist/jquery.validate.js"></script>
    <script src="~/Identity/lib/jquery-validation-
unobtrusive/jquery.validate.unobtrusive.js"></script>
</environment>
<environment exclude="Development">
    <script src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.17.0/jquery.vali
date.min.js"
```

```
            asp-fallback-src="~/Identity/lib/jquery-
validation/dist/jquery.validate.min.js"
            asp-fallback-test="window.jQuery && window.jQuery.validator"
            crossorigin="anonymous"
            integrity="sha384-
rZfj/ogBloos6wzLGpPkkOr/gpkBNLZ6b6yLy4o+ok+t/SAKlL5mvXLr0OXNi1Hp">
    </script>
    <script src="https://ajax.aspnetcdn.com/ajax/jquery.validation.unobtrusive/3.2.
9/jquery.validate.unobtrusive.min.js"
            asp-fallback-src="~/Identity/lib/jquery-validation-
unobtrusive/jquery.validate.unobtrusive.min.js"
            asp-fallback-
test="window.jQuery && window.jQuery.validator && window.jQuery.validator.unobtrusi
ve"
            crossorigin="anonymous"
            integrity="sha384-
ifv0TYDWxBHzvAk2Z0n8R434FL1Rlv/Av18DXE43N/1rvHyOG4izKst0f2iSLdds">
    </script>
</environment>
<script>
    jQuery(document).ready(function () {
        $(".student-selected, .teacher-selected").hide();
    });
    jQuery("#Input_RoleId").change(function (data) {
        $(".student-selected, .teacher-selected").hide(500);
        if ($(this).find(":selected").text() === "Teacher") {
            $(".teacher-selected").show(500);
        }
        if ($(this).find(":selected").text() === "Student") {
            $(".student-selected").show(500);
        }
    });
</script>
```

Po uruchomieniu powinniśmy uzyskać poniższy efekt:



4. Controllers & Views (kontrolery i widoki)

Mając zbudowane oraz zarejestrowane usługi, możemy je wykorzystać do stworzenia kontrolerów, a na ich podstawie również widoków.

1) Przed utworzeniem kontrolerów należy najpierw podmienić zawartość pliku pliku ***Startup.cs*** w projekcie ***SchoolRegister.Web*** na poniższą zawartość. Czerwoną czcionką oznaczono dodane elementy kodu, głownie dotyczące procesu internacjonalizacji, który będziemy implementować na kolejnych laboratoriach. Jednakże te elementy będą potrzebne na tym etapie w kontrolerach.

```
using System.Globalization;
using System.Net;
using System.Net.Mail;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.UI.Services;
using Microsoft.AspNetCore.Localization;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Localization;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Options;
using SchoolRegister.DAL.EF;
using SchoolRegister.Model.DataModels;
using SchoolRegister.Services.Interfaces;
using SchoolRegister.Services.Services;
using SchoolRegister.Web.Controllers;

namespace SchoolRegister.Web {
    public class Startup {
        public IConfiguration Configuration { get; }
        public Startup (IConfiguration configuration) {
            Configuration = configuration;
        }

        // This method gets called by the runtime. Use this method to add services to the container.
        public void ConfigureServices (IServiceCollection services) {
```

```
            services.AddAutoMapper (typeof (Startup));
            services.AddDbContext<ApplicationDbContext> (options =>
                options.UseSqlServer (Configuration.GetConnectionString ("DefaultConnection")) //here
you can define a database type.
            );
            services.AddDatabaseDeveloperPageExceptionFilter ();
            services.AddDefaultIdentity<User> (options => options.SignIn.RequireConfirmedAccount = fal
se)
                .AddRoles<Role> ()
                .AddRoleManager<RoleManager<Role>> ()
                .AddUserManager<UserManager<User>> ()
                .AddEntityFrameworkStores<ApplicationDbContext> ();
            services.AddTransient (typeof (ILogger), typeof (Logger<Startup>));
            services.AddTransient<IStringLocalizer, StringLocalizer<BaseController>>();
            services.AddScoped<ISubjectService, SubjectService> ();
            services.AddScoped<IEmailSender, EmailSenderService> ();
            services.AddScoped<IEmailSenderService, EmailSenderService> ();
            services.AddScoped<IGradeService, GradeService> ();
            services.AddScoped<IGroupService, GroupService> ();
            services.AddScoped<IStudentService, StudentService> ();
            services.AddScoped<ITeacherService, TeacherService> ();
            services.AddScoped ((serviceProvider) => {
                var config = serviceProvider.GetRequiredService<IConfiguration> ();
                return new SmtpClient () {
                    Host = config.GetValue<string> ("Email:Smtp:Host"),
                        Port = config.GetValue<int> ("Email:Smtp:Port"),
                        EnableSsl = true,
                        DeliveryMethod = SmtpDeliveryMethod.Network,
                        UseDefaultCredentials = false,
                        Credentials = new NetworkCredential (
                            config.GetValue<string> ("Email:Smtp:Username"),
                            config.GetValue<string> ("Email:Smtp:Password")
                        )
                };
            });
            services.Configure<RequestLocalizationOptions> (options => {
                var supportedCultures = new [] {
                new CultureInfo ("en"),
                new CultureInfo ("pl-PL")
                };

                options.DefaultRequestCulture = new RequestCulture (culture: "en", uiCulture: "en");
                options.SupportedCultures = supportedCultures;
                options.SupportedUICultures = supportedCultures;
            });
            services.AddLocalization (options => options.ResourcesPath = "Resources");
            services.AddControllersWithViews ()
                .AddViewLocalization ()
                .AddDataAnnotationsLocalization ();
        }

        // This method gets called by the runtime. Use this method to configure the HTTP request pipel
ine.
        public void Configure (IApplicationBuilder app, IWebHostEnvironment env) {
            if (env.IsDevelopment ()) {
                app.UseDeveloperExceptionPage ();
                app.UseMigrationsEndPoint ();
            } else {
                app.UseExceptionHandler ("/Error");
                app.UseHsts ();
            }
            app.UseHttpsRedirection ();
            app.UseStaticFiles ();
            app.UseRouting ();
            app.UseAuthentication ();
            app.UseAuthorization ();
            var localizationOption = app.ApplicationServices.GetService<IOptions<RequestLocalizationOp
tions>>();
            app.UseRequestLocalization(localizationOption.Value);
            app.UseEndpoints (endpoints => {
                endpoints.MapControllerRoute (
                    name: "default",
                    pattern: "{controller=Subject}/{action=Index}/{id?}");
```

```
                endpoints.MapRazorPages ();
            });
        }
    }
}
```

2) Proszę stworzyć nowy plik w ***SchoolRegister.Web/Contollers/BaseController.cs*** **Wybierając nazwę, pamiętajmy o sufiksie *Controller*.** Następnie proszę skopiować zawrtość pliku ***BaseController.cs***

```
using AutoMapper;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Localization;
using Microsoft.Extensions.Logging;

namespace SchoolRegister.Web.Controllers
{
    public abstract class BaseController : Controller
    {
        protected readonly IStringLocalizer Localizer;
        protected readonly ILogger Logger;
        protected readonly IMapper Mapper;
        public BaseController(ILogger logger, IMapper mapper, IStringLocalizer localizer)
        {
            Localizer = localizer;
            Logger = logger;
            Mapper = mapper;
        }
    }
}
```

3) Proszę stworzyć nowy plik w ***SchoolRegister.Web/Contollers/SubjectController.cs***. Następnie dodajemy pola o typie interfejsowym, dla odpowiednich usług. Dzieki temu będziemy mogli z nich korzystać w naszym kontrolerze. Jak mozna zauważyć do kontrukora równiez podajemy parametry interfejsowe. Wykorzystajmy w ten sposób ***_subjectService*** aby pobrać wszystkie przedmioty. Następnie zwrócimy widok z przekazanym view modelem.

```
using System;
using System.Linq;
using System.Linq.Expressions;
using AutoMapper;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.Extensions.Localization;
using Microsoft.Extensions.Logging;
using SchoolRegister.Model.DataModels;
using SchoolRegister.Services.Interfaces;
using SchoolRegister.ViewModels.VM;

namespace SchoolRegister.Web.Controllers
{
    [Authorize(Roles = "Teacher, Admin, Student")]
    public class SubjectController : BaseController
    {
        private readonly ISubjectService _subjectService;
        private readonly ITeacherService _teacherService;
        private readonly UserManager<User> _userManager;
        public SubjectController(ISubjectService subjectService,
            ITeacherService teacherService,
            UserManager<User> userManager,
            IStringLocalizer localizer,
            ILogger logger,
            IMapper mapper) : base(logger, mapper, localizer)
        {
            _subjectService = subjectService;
            _teacherService = teacherService;
            _userManager = userManager;
        }

        public IActionResult Index()
        {
            var user = _userManager.GetUserAsync(User).Result;
            if (_userManager.IsInRoleAsync(user, "Admin").Result)
                return View(_subjectService.GetSubjects());
            else if (_userManager.IsInRoleAsync(user, "Teacher").Result)
            {
                var teacher = _userManager.GetUserAsync(User).Result as Teacher;
                return View(_subjectService.GetSubjects(x => x.TeacherId == teacher.Id));
            }
            else if (_userManager.IsInRoleAsync(user, "Student").Result)
                return RedirectToAction("Details", "Student", new { studentId = user.Id });
```

```
            else
                return View("Error");

        }

        [HttpGet]
        [Authorize(Roles = "Admin")]
        public IActionResult AddOrEditSubject(int? id = null)
        {
            var teachersVm = _teacherService.GetTeachers();
            ViewBag.TeachersSelectList = new SelectList(teachersVm.Select(t => new
            {
                Text = $"{t.FirstName} {t.LastName}",
                Value = t.Id
            }), "Value", "Text");
            if (id.HasValue)
            {
                var subjectVm = _subjectService.GetSubject(x => x.Id == id);
                ViewBag.ActionType = "Edit";
                return View(Mapper.Map<AddOrUpdateSubjectVm>(subjectVm));
            }
            ViewBag.ActionType = "Add";
            return View();
        }
        public IActionResult Details(int id)
        {
            var subjectVm = _subjectService.GetSubject(x => x.Id == id);
            return View(subjectVm);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        [Authorize(Roles = "Admin")]
        public IActionResult AddOrEditSubject(AddOrUpdateSubjectVm addOrUpdateSubjectVm)
        {
            if (ModelState.IsValid)
            {
                _subjectService.AddOrUpdateSubject(addOrUpdateSubjectVm);
                return RedirectToAction("Index");
            }
            return View();
        }
    }
}
```

4) Mając utworzony akcję kontrolera (Index) możemy dla niej zbudować widok (View).

```
dotnet aspnet-codegenerator view -p .\SchoolRegister.Web\SchoolRegister.Web.csproj Index List -m SubjectVm -outDir Views\Subject -scripts -udl
```

5) Następnie w folderze ***SchoolRegister.Web/Views*** został utworzony folder ***Subjects*** a w nim plik ***Index.cshtml.*** Proszę zmodyfikować utworzony widok w poniższy sposób:

```
@using SchoolRegister.ViewModels.VM
@model IEnumerable<SubjectVm>
@{
    ViewData["Title"] = "Index";
}
<h2>Subject</h2>
@if (User.IsInRole("Admin"))
{
    <p>
        <a asp-action="AddOrEditSubject">Create New</a>
    </p>
}
<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Name)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Description)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.TeacherName)
            </th>
            <th>
                <label>Groups Count</label>
            </th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model)
        {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.Name)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Description)
                </td>
```

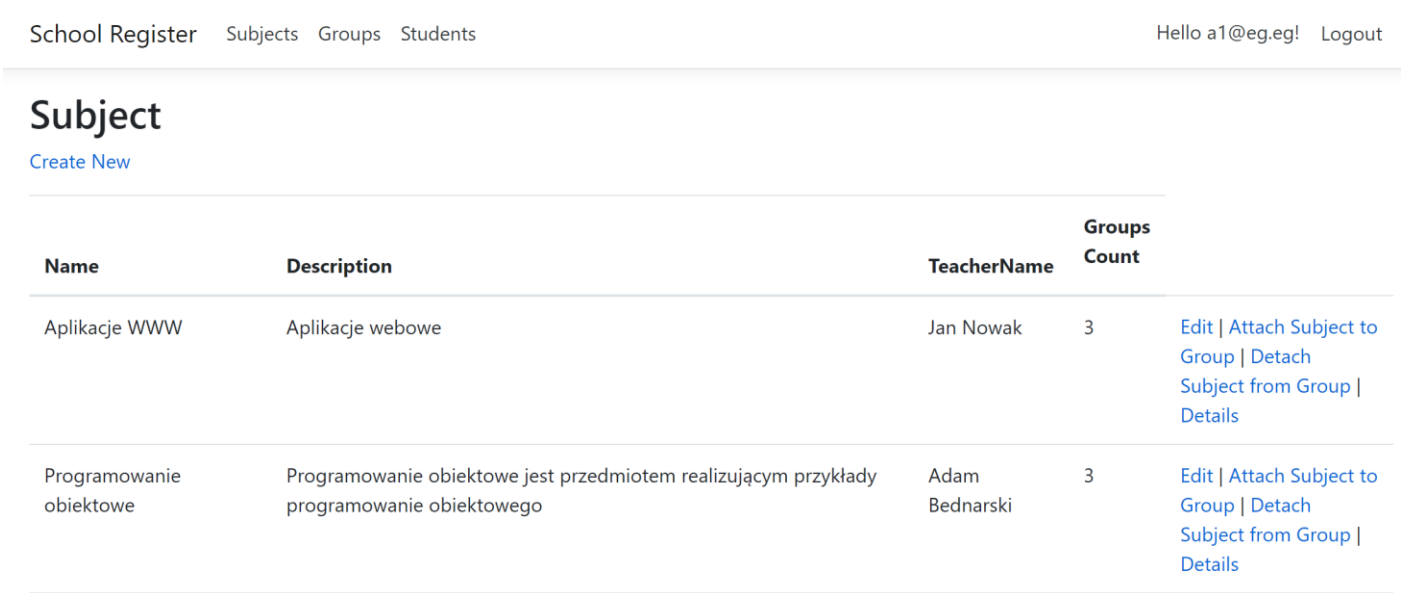
```
                <td>
                    @Html.DisplayFor(modelItem => item.TeacherName)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Groups.Count)
                </td>
                <td>
                    @if (User.IsInRole("Admin"))
                    {
                        <a asp-action="AddOrEditSubject" asp-route-id="@item.Id">Edit</a> @:|
                        <a asp-controller="Group" asp-action="AttachSubjectToGroup" asp-route-
subjectId="@item.Id">Attach Subject to Group</a> @:|
                        <a asp-controller="Group" asp-action="DetachSubjectToGroup" asp-route-
subjectId="@item.Id">Detach Subject from Group</a> @:|
                    }
                    <a asp-action="Details" asp-route-id="@item.Id">Details</a>
                </td>
            </tr>
        }
    </tbody>
</table>
```

6) Kolejno przechodzimy do ***SchoolRegister.Web/Views/Shared/_Layout.cshtml***, a następnie dokonujemy poniższych zmian:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - School Register</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-
shadow mb-3">
            <div class="container">
                <a class="navbar-brand" asp-area="" asp-controller="Subject" asp-
action="Index">School Register</a>
                <button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-
collapse" aria-controls="navbarSupportedContent"
                        aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
                    <ul class="navbar-nav flex-grow-1">
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-controller="Subject" asp-
action="Index">Subjects</a>
                        </li>
                    </ul>
                    <partial name="_LoginPartial" />
                </div>
            </div>
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>
    <footer class="border-top footer text-muted">
        <div class="container">
            &copy; @DateTime.Now.Year - School Register
        </div>
    </footer>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @await RenderSectionAsync("Scripts", required: false)
</body>
```

```
</html>
```

7) Po uruchomieniu aplikacji powinniśmy uzyskać następujący efekt.



Na tym etapie możemy wyświetlić wszystkie przedmioty dodane do bazy danych przy pomocy skryptu SQL, Należy zbudować mechanizmy pozwlające na tworzenie oraz modyfikację przedmiotów. Czesto stosuje się podejście w którym metody add oraz edit tworzone są osobno, lecz zgodnie z regułą DRY zbudujemy metodę która pozwoli kontekstowo modyfikować oraz tworzyć przedmioty.

8) **W pierwszym kroku tworzymy metodę typu GET. Możemy oznaczyć metodę jako GET poprzez dodania atrybutu [HttpGet] lub nie podając jej w ogóle. Wszystkie metody bez atrybutu traktowane są jako GET.** Metoda ***AddOrEditSubject*** została już zdefiniowana w kontrolerze stąd należy jedynie użyć mechanizmu scaffolding w celu wygenerowania widoku.

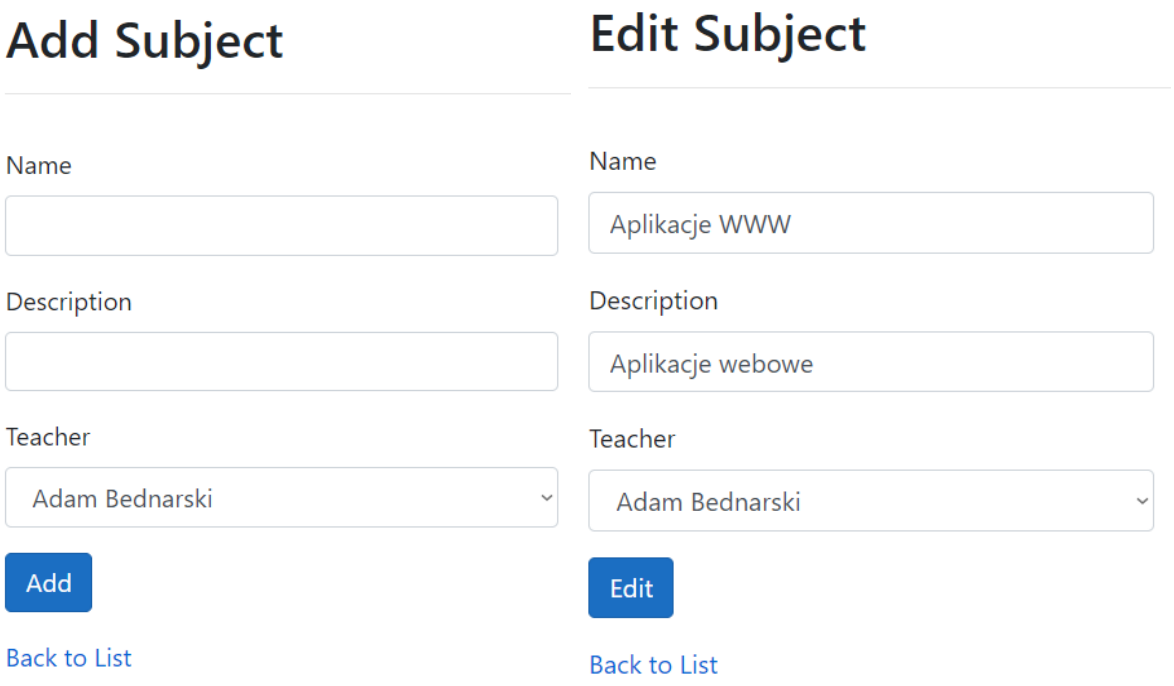
```
dotnet aspnet-codegenerator view -p .\SchoolRegister.Web\SchoolRegister.Web.csproj AddOrEditSubject
Create -m AddOrUpdateSubjectVm -outDir Views\Subject -scripts -udl
```

Po wygenerowaniu kodu widoku proszę dokonać następujących zmian:

```
@model SchoolRegister.ViewModels.VM.AddOrUpdateSubjectVm
@{
    ViewData["Title"] = $"{ViewBag.ActionType} Subject";
}
<h2>@ViewData["Title"]</h2>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="AddOrEditSubject">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <input asp-for="Id" type="hidden" class="form-control" />
            </div>
            <div class="form-group">
                <label asp-for="Name" class="control-label"></label>
                <input asp-for="Name" class="form-control" />
                <span asp-validation-for="Name" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Description" class="control-label"></label>
                <input asp-for="Description" class="form-control" />
                <span asp-validation-for="Description" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label  class="control-label">Teacher</label>
                <select asp-for="TeacherId" class="form-control"
                        asp-items="@ViewBag.TeachersSelectList"></select>
                <span asp-validation-for="TeacherId" class="text-danger"></span>
```

```
            </div>
            <div class="form-group">
                <input type="submit" value="@ViewBag.ActionType" class="btn btn-primary" />

            </div>
        </form>
    </div>
</div>
<div>
    <a asp-action="Index">Back to List</a>
</div>
@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

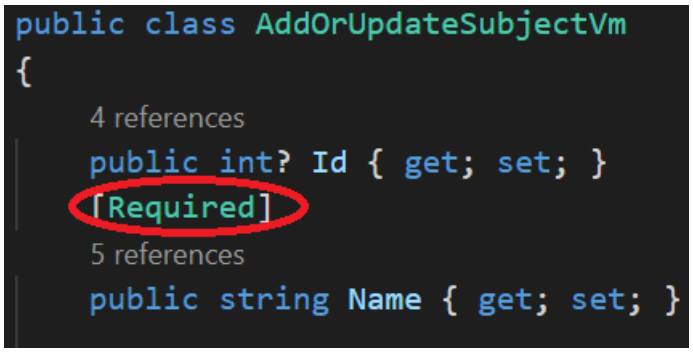
Funkcjonalność metody (GET) ***AddOrEditSubject*** jest następująca: Gdy wartość parametru ***id*** jest pusta (null) zwracany jest pusty widok inaczej mówiąc w takim przypadku chcemy wykonać akcję ***add***. W przeciwnym przypadku pobierana jest encja z serwisu o podanym ***id*** oraz zwracany jest widok z view modelem jako wartość parametru. Tworzona jest również lista nauczycieli ***teachersSelectList***. Jest ona wykorzystywana do wyświetlenia nauczycieli w liście rozwijanej. **Dzięki utworzeniu tej metody *AddOrEditSubject* możemy jednoczenie przechodzić do formularzy tworzenia oraz edycji.**



W kontrolerze istnieje druga metoda *AddOrEditSubject* typu POST o tej samej nazwie zatwierdzająca formularz. Warto podkreślić że musi zawierać atrybut [HttpPost] nad swoją sygnaturą.

```
[HttpPost]
[ValidateAntiForgeryToken]
[Authorize(Roles = "Admin")]
0 references
public IActionResult AddOrEditSubject(AddOrUpdateSubjectVm addOrUpdateSubjectVm)
{
    if (ModelState.IsValid)
    {
        _subjectService.AddOrUpdateSubject(addOrUpdateSubjectVm);
        return RedirectToAction("Index");
    }
    return View();
}
```

Powyższa metoda przyjmuje dedykowany view model, nastepnie sprawdza (waliduje) czy jest on zgodny z dodanymi adotacjami. Jeśli model jest zgodny wywoływana jest metoda ***AddOrUpdateSubject*** (POST) z serwisu a następnie zwracane jest przekierowanie do akcji Index. Teraz możemy dodawać oraz modyfikować dane przedmiotów.



Atrybut **[ValidateAniForgeryToken]** umożliwia przeciwdziałanie atakom Cross-site request forgery (XSRF,CSRF). Natomiast atrybut **[Authorize]** pozwala na ograniczenie dostępu do poszczgólnych akcji kontrolera dla użytkowników niezalogowanych lub nienależących do określonej roli.

5. Zadanie

Po wykonaniu powyższej instrukcji proszę stworzyć pozostałe kontrolery oraz widoki aby spełnic poniższe funkcjonalności systemu. **Przy wykorzystaniu utworzonych wcześniej usług (serwisów), należy pamiętać, aby NIE dodawać logiki biznesowej do kontrolerów - jest to zadanie oddelegowane do serwisów. Zadaniem kontrolerów jest:**

- **Odebranie danych od widoku oraz ich walidacja,**
- **Przekazanie danych do serwisu – wywołanie metody z serwisu,**
- **Wyświetlenie danych otrzymanych z serwisu do widoku.**

Funkcjonalności systemu:

- Dodawanie, modyfikacja, oraz wyświetlanie przedmiotów.
- Wystawianie ocen studentom przez użytkowników z rolą *Teacher*.
- Możliwość wyświetlania swoich ocen (lub ocen swoich dzieci) przez użytkowników z rolą *Parent* lub *Student*.
- Użytkownicy z rolą *Teacher* posiadają możliwość wysyłania emaila do użytkowników z rolą *Parent*.
- Zarządzanie grupami (klasami) (tworzenie oraz modyfikacja grup)
- Dodawanie oraz usuwanie studentów do i z grup.

Po wykonaniu dzisiejszych laboratoriów powinniście Państwo uzyskać w pełni funkcjonalny system webowy spełniajacy powyższe funkcjonalności.