Aplikacje WWW
(studia niestacjonarne)
Laboratorium 4

Software testing, Unit Tests

1. Testowanie oprogramowania (Software testing)

Testowanie oprogramowanie jest oraz powinno być nieodłącznym elementem w trakcie procesu tworzenia aplikacji. Niestety w wielu przypadkach ten proces jest całkowicie pomijany, co często prowadzi do poważnych luk w oprogramowaniu. Często również w celu oszczędności w procesie developmentu stosuje się testowanie jedynie najbardziej witalnych modułów systemu. Jednakże profesjonalne systemy oraz projekty opierają się na wielu modelach testowania oprogramowania. Docelowo dążymy do pokrycia testami jak największej ilości kodu. Można wyróżnić następujące poziomy testów:

- Testy jednostkowe – pozwalają na testowanie pojedynczych elementów systemu tj. metody, obiekty.
- Testy integracyjne – testy tego typu pozwalają na wykrycie luk w poszczególnych modułach lub elementach systemu jak również, umożliwiają wykrycie problemów pomiędzy poszczególnymi wersjami oprogramowania.
- Testy GUI – testy tego typu pozwalają na sprawdzenie poprawności działania interfejsu użytkownika. Przykładem narzędzia do testów GUI, jest Selenium.

Obecnie, praktycznie każdy framework webowy korzysta z frameworka do testów. Nie inaczej jest w przypadku .NET 5. Dostępne są frameworki testowe:

- MSTest – framework do testów dostarczany przez Microsoft.
- NUnit – jest framework zaczerpniętym (porting) z JUnit, dedykowanego do Javy.
- xUnit – framework napisany przez twórcę NUnita, dedykowany do rozwiązań .NET, wspierający rożne narzędzia, np. ReSharper.

Podczas laboratorium będziemy wykorzystywać xUnita. Jednakże możliwe jest wykorzystywanie pozostałych.

Podczas poprzednich laboratoriów tworzone były usługi, których celem jest realizacja poszczególnych funkcjonalności w oparciu o dostarczone dane poprzez view modele. Jednakże do tej pory nie było możliwe sprawdzenie czy dana metoda/klasa działa poprawnie i zgodnie z założonymi wymaganiami. Po zastosowaniu testów jednostkowych będzie to możliwe. W celu konfiguracji projektu do testów jednostkowych proszę wykonać następujące kroki:

1) Proszę stworzyć nowy projekt o nazwie ***SchoolRegister.Tests***:

```
dotnet new xunit -n SchoolRegister.Tests -f net5.0
```

2) Następnie proszę dodać projekt do solucji:

```
dotnet sln SchoolRegister.sln add (ls -r **/*.csproj)
```

3) W kolejnym kroku proszę dodać poniższe referencje:

- `dotnet add SchoolRegister.Tests/SchoolRegister.Tests.csproj reference SchoolRegister.Services/SchoolRegister.Services.csproj`
- `dotnet add SchoolRegister.Tests/SchoolRegister.Tests.csproj reference .\SchoolRegister.Model\SchoolRegister.Model.csproj`
- `dotnet add SchoolRegister.Tests/SchoolRegister.Tests.csproj reference .\SchoolRegister.DAL\SchoolRegister.DAL.csproj`
- `dotnet add SchoolRegister.Tests/SchoolRegister.Tests.csproj reference .\SchoolRegister.Web\SchoolRegister.Web.csproj`

4) Kolejno proszę zainstalować następujące biblioteki:

- `dotnet add .\SchoolRegister.Tests\SchoolRegister.Tests.csproj package Xunit.DependencyInjection --version 7.1.0`
- `dotnet add .\SchoolRegister.Tests\SchoolRegister.Tests.csproj package Microsoft.EntityFrameworkCore.InMemory --version 5.0.3`

5) W projekcie ***SchoolRegister.Tests*** proszę stworzyć nowy plik o nazwie ***appsettings.json*** a następnie wkleić poniższą zawartość. **Proszę zwrócić uwagę na fakt że jeśli chcemy przetestować funkcjonalność wysyłania e-maili to należy w tym miejscu podać dane serwera *SMTP* oraz poświadczenia. Dla niektórych serwerów pocztowych tj. Gmail, jest również konieczne włączenie tzw. aplikacji niebezpiecznych. Należy to zrobić na poziomie skrzynki pocztowej.**

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "Email": {
    "Smtp": {
      "Host": "host_domain_name",
      "Port": 587,
      "Username": "username",
      "Password": "password"
    }
  },
  "AllowedHosts": "*"
}
```

6) Proszę w projekcie ***SchoolRegister.Tests*** stworzyć plik ***Extensions.cs*** a następnie wkleić poniższą zawartość:

```
using System;
using Microsoft.AspNetCore.Identity;
using Microsoft.Extensions.DependencyInjection;
using SchoolRegister.DAL.EF;
using SchoolRegister.Model.DataModels;

public static class Extensions {

    // Create sample data
    public static async void SeedData (this IServiceCollection services) {
        var serviceProvider = services.BuildServiceProvider ();
        var dbContext = serviceProvider.GetRequiredService<ApplicationDbContext> ();
        var userManager = serviceProvider.GetRequiredService<UserManager<User>> ();
        var roleManager = serviceProvider.GetRequiredService<RoleManager<Role>> ();

        // Roles
        var teacherRole = new Role () {
            Id = 3,
            Name = "Teacher",
            RoleValue = RoleValue.Teacher
        };
        await roleManager.CreateAsync (teacherRole);

        var studentRole = new Role () {
            Id = 1,
            Name = "Student",
            RoleValue = RoleValue.Student
        };
        await roleManager.CreateAsync (studentRole);

        var parentRole = new Role () {
```

```
            Id = 2,
            Name = "Parent",
            RoleValue = RoleValue.Parent
        };
        await roleManager.CreateAsync (parentRole);

        var adminRole = new Role () {
            Id = 4,
            Name = "Admin",
            RoleValue = RoleValue.Admin
        };
        await roleManager.CreateAsync (adminRole);

        // Groups
        var groupIo = new Group () {
            Id = 1,
            Name = "IO"
        };
        await dbContext.Groups.AddAsync (groupIo);
        var groupPai = new Group () {
            Id = 2,
            Name = "PAI"
        };
        await dbContext.Groups.AddAsync (groupPai);
        var groupAipErasmus = new Group () {
            Id = 3,
            Name = "AIP Erasmus"
        };
        await dbContext.Groups.AddAsync (groupAipErasmus);

        // All users, including teachers, students and parents
        var userPassword = "User1234"; // in lab env we use the same pass for all users :)
        var t1 = new Teacher () {
            Id = 1,
            FirstName = "Adam",
            LastName = "Bednarski",
            UserName = "t1@eg.eg",
            Email = "real_email@eg.eg",
            Title = "mgr inż.",
            RegistrationDate = new DateTime (2010, 1, 1),
        };
        await userManager.CreateAsync (t1, userPassword);
        await userManager.AddToRoleAsync (t1, teacherRole.Name);

        var t2 = new Teacher () {
            Id = 2,
            FirstName = "Jan",
            LastName = "Nowak",
            UserName = "t2@eg.eg",
            Email = "t2@eg.eg",
            Title = "mgr",
            RegistrationDate = new DateTime (2010, 11, 12),
        };
        await userManager.CreateAsync (t2, userPassword);
        await userManager.AddToRoleAsync (t2, teacherRole.Name);

        var t3 = new Teacher () {
            Id = 12,
            FirstName = "Stanisław",
            LastName = "Nowakowski",
            UserName = "t11@eg.eg",
            Email = "t11@eg.eg",
            Title = "mgr inż.",
            RegistrationDate = new DateTime (2010, 11, 12),
        };
        await userManager.CreateAsync (t3, userPassword);
        await userManager.AddToRoleAsync (t3, teacherRole.Name);

        var p1 = new Parent () {
            Id = 3,
            FirstName = "Zbigniew",
            LastName = "Kowalski",
            UserName = "p1@eg.eg",
            Email = "real_email@eg.eg",
            RegistrationDate = new DateTime (2014, 03, 20),
        };
        await userManager.CreateAsync (p1, userPassword);
        await userManager.AddToRoleAsync (p1, parentRole.Name);

        var p2 = new Parent () {
            Id = 4,
            FirstName = "Anna",
            LastName = "Nowakowska",
            UserName = "p2@eg.eg",
            Email = "p2@eg.eg",
```

```
            RegistrationDate = new DateTime (2014, 06, 21),
        };
        await userManager.CreateAsync (p2, userPassword);
        await userManager.AddToRoleAsync (p2, parentRole.Name);

        var s1 = new Student () {
            Id = 5,
            FirstName = "Tomasz",
            LastName = "Kowalski",
            UserName = "s1@eg.eg",
            Email = "s1@eg.eg",
            RegistrationDate = new DateTime (2016, 05, 11),
            GroupId = 1,
            ParentId = 3
        };
        await userManager.CreateAsync (s1, userPassword);
        await userManager.AddToRoleAsync (s1, studentRole.Name);

        var s2 = new Student () {
            Id = 6,
            FirstName = "Krzysztof",
            LastName = "Kowalski",
            UserName = "s2@eg.eg",
            Email = "s2@eg.eg",
            RegistrationDate = new DateTime (2015, 09, 18),
            GroupId = 1,
            ParentId = 3
        };
        await userManager.CreateAsync (s2, userPassword);
        await userManager.AddToRoleAsync (s2, studentRole.Name);

        var s3 = new Student () {
            Id = 7,
            FirstName = "Natalia",
            LastName = "Kowalska",
            UserName = "s3@eg.eg",
            Email = "s3@eg.eg",
            RegistrationDate = new DateTime (2017, 07, 16),
            GroupId = 2,
            ParentId = 3
        };
        await userManager.CreateAsync (s3, userPassword);
        await userManager.AddToRoleAsync (s3, studentRole.Name);

        var s4 = new Student () {
            Id = 8,
            FirstName = "Magdalena",
            LastName = "Wiśniewska",
            UserName = "s4@eg.eg",
            Email = "s4@eg.eg",
            RegistrationDate = new DateTime (2018, 05, 14),
            GroupId = 2,
            ParentId = 4
        };
        await userManager.CreateAsync (s4, userPassword);
        await userManager.AddToRoleAsync (s4, studentRole.Name);

        var s5 = new Student () {
            Id = 9,
            FirstName = "Jan",
            LastName = "Wiśniewski",
            UserName = "s5@eg.eg",
            Email = "s5@eg.eg",
            RegistrationDate = new DateTime (2019, 02, 19),
            GroupId = 3,
            ParentId = 4
        };
        await userManager.CreateAsync (s5, userPassword);
        await userManager.AddToRoleAsync (s5, studentRole.Name);

        var s6 = new Student () {
            Id = 10,
            FirstName = "Krystian",
            LastName = "Wiśniewski",
            UserName = "s6@eg.eg",
            Email = "s6@eg.eg",
            RegistrationDate = new DateTime (2019, 05, 1),
            GroupId = 3,
            ParentId = 4
        };
        await userManager.CreateAsync (s6, userPassword);
        await userManager.AddToRoleAsync (s6, studentRole.Name);

        var a1 = new User () {
            Id = 11,
```

```
            FirstName = "Jacek",
            LastName = "Kowalczyk",
            UserName = "a1@eg.eg",
            Email = "a1@eg.eg",
            RegistrationDate = new DateTime (2009, 1, 1)
        };
        await userManager.CreateAsync (a1, userPassword);
        await userManager.AddToRoleAsync (a1, adminRole.Name);

        // Subject
        var subject1 = new Subject () {
            Id = 1,
            Name = "Aplikacje WWW",
            Description = "Aplikacje webowe",
            TeacherId = 1
        };
        await dbContext.AddAsync (subject1);

        var subject2 = new Subject () {
            Id = 2,
            Name = "Programowanie obiektowe",
            Description = "Programowanie obiektowe jest przedmiotem realizującym przykłady programowanie obiektowe
go",
            TeacherId = 1
        };
        await dbContext.AddAsync (subject2);

        var subject3 = new Subject () {
            Id = 3,
            Name = "Advanced Internet Programming",
            Description = "Advanced Internet Programming is a course for ERASMUS+ students",
            TeacherId = 2
        };
        await dbContext.AddAsync (subject3);

        var subject4 = new Subject () {
            Id = 4,
            Name = "Administracja Intenetowymi Systemami Baz Danych",
            Description = "Administracja Intenetowymi Systemami Baz Danych jest kontynuacją przedmiotu Bazy danych
 na studiach stacjonarnych I-go stopnia spec. PAI",
            TeacherId = 2,
        };
        await dbContext.AddAsync (subject4);

        var subject5 = new Subject () {
            Id = 5,
            Name = "Programowanie interaktywnej grafiki dla stron WWW",
            TeacherId = 12
        };
        await dbContext.AddAsync (subject5);

        //SubjectGroups
        var subjectGroup1 = new SubjectGroup () {
            SubjectId = 1,
            GroupId = 1,
        };
        await dbContext.SubjectGroups.AddAsync (subjectGroup1);

        var subjectGroup2 = new SubjectGroup () {
            SubjectId = 1,
            GroupId = 2,
        };
        await dbContext.SubjectGroups.AddAsync (subjectGroup2);

        var subjectGroup3 = new SubjectGroup () {
            SubjectId = 2,
            GroupId = 1,
        };
        await dbContext.SubjectGroups.AddAsync (subjectGroup3);

        var subjectGroup4 = new SubjectGroup () {
            SubjectId = 2,
            GroupId = 2,
        };
        await dbContext.SubjectGroups.AddAsync (subjectGroup4);

        var subjectGroup5 = new SubjectGroup () {
            SubjectId = 2,
            GroupId = 3,
        };
        await dbContext.SubjectGroups.AddAsync (subjectGroup5);

        var subjectGroup6 = new SubjectGroup () {
            SubjectId = 3,
            GroupId = 3,
```

```
        };
        await dbContext.SubjectGroups.AddAsync (subjectGroup6);

        var subjectGroup7 = new SubjectGroup () {
            SubjectId = 4,
            GroupId = 2,
        };
        await dbContext.SubjectGroups.AddAsync (subjectGroup7);

        var subjectGroup8 = new SubjectGroup () {
            SubjectId = 4,
            GroupId = 3,
        };
        await dbContext.SubjectGroups.AddAsync (subjectGroup8);

        var grade1 = new Grade () {
            DateOfIssue = new DateTime (2019, 03, 21, 17, 46, 38),
            StudentId = 5,
            SubjectId = 1,
            GradeValue = GradeScale.DB
        };
        await dbContext.Grades.AddAsync (grade1);

        await dbContext.SaveChangesAsync ();
    }
}
```

7) W ostatnim kroku proszę stworzyć plik ***Startup.cs*** a następnie do niego wkleić poniższy kod:

```
using System;
using System.IO;
using System.Net;
using System.Net.Mail;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;
using SchoolRegister.DAL.EF;
using SchoolRegister.Model.DataModels;
using SchoolRegister.Services.Interfaces;
using SchoolRegister.Services.Services;
using SchoolRegister.Web.Configuration.Profiles;

namespace SchoolRegister.Tests {
    public class Startup {
        public Startup () {

        }
        public void ConfigureServices (IServiceCollection services) {
            services.AddAutoMapper (typeof (MainProfile));
            services.AddEntityFrameworkInMemoryDatabase ()
                .AddDbContext<ApplicationDbContext> (options =>
                    options.UseInMemoryDatabase ("InMemoryDb")
                );
            services.AddIdentity<User, Role> (options => {
                    options.SignIn.RequireConfirmedAccount = false;
                    options.Password.RequiredLength = 6;
                    options.Password.RequiredUniqueChars = 0;
                    options.Password.RequireNonAlphanumeric = false;
                })
                .AddRoleManager<RoleManager<Role>> ()
                .AddUserManager<UserManager<User>> ()
                .AddEntityFrameworkStores<ApplicationDbContext> ();
            services.AddTransient (typeof (ILogger), typeof (Logger<Startup>));
            services.AddTransient<ISubjectService, SubjectService> ();
            services.AddTransient<IEmailSenderService, EmailSenderService> ();
            services.AddTransient<IGradeService, GradeService> ();
            services.AddTransient<IGroupService, GroupService> ();
            services.AddTransient<IStudentService, StudentService> ();
            services.AddTransient<ITeacherService, TeacherService> ();
            services.AddSingleton<IConfiguration>(x=> {
```

```
                    var builder = new ConfigurationBuilder ()
                .SetBasePath (Directory.GetCurrentDirectory ())
                .AddJsonFile ("appsettings.json", optional : false, reloadOnChange : true)
                .AddEnvironmentVariables ();
                return builder.Build ();
            });
            services.AddTransient ((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.SeedData ();
        }
    }
}
```

2. Tworzenie testów jednostkowych (Unit tests)

Aby stworzyć testy jednostkowe posłużymy się klasą bazową, dzięki której zaoszczędzimy ciągłego tworzenia obiektów typu ***ApplicationDbContext***. Aby stworzyć testy jednostkowe proszę wykonać następujące kroki:

1) Proszę w projekcie ***SchoolRegister.Tests*** stworzyć nowy folder o nazwie ***UnitTests,*** a następnie w w tym folderze stworzyć nowy plik o nazwie ***BaseUnitTests.cs***, a następnie proszę wkleić poniższy kod:

```
using AutoMapper;
using SchoolRegister.DAL.EF;

namespace SchoolRegister.Tests.UnitTests {
    public abstract class BaseUnitTests {
        protected readonly ApplicationDbContext DbContext;

        public BaseUnitTests (ApplicationDbContext dbContext) {
            DbContext = dbContext;;
        }
    }
}
```

2) W kolejnym kroku proszę stworzyć nowy plik o nazwie ***GradeServiceUnitTests.cs.*** Następnie proszę wkleić poniższy kod. **W przypadku gdy nie posiadacie Państwo danej metody w swoich serwisach proszę odpowiednio zmodyfikować poniższy kod.**

```
using System;
using System.Linq;
using SchoolRegister.DAL.EF;
using SchoolRegister.Model.DataModels;
using SchoolRegister.Services.Interfaces;
using SchoolRegister.ViewModels.VM;
using Xunit;

namespace SchoolRegister.Tests.UnitTests {
    public class GradeServiceUnitTests : BaseUnitTests {
        private readonly IGradeService _gradeService;
        public GradeServiceUnitTests (ApplicationDbContext dbContext, IGradeService gradeService) : base (dbContext) {
            _gradeService = gradeService;
        }
        [Fact]
        public void AddGradeToStudent () {
```

```
            var gradeVm = new AddGradeToStudentVm () {
                StudentId = 5,
                SubjectId = 1,
                GradeValue = GradeScale.DB,
                TeacherId = 1
            };
            var grade = _gradeService.AddGradeToStudent (gradeVm);
            Assert.NotNull(grade);
            Assert.Equal(2, DbContext.Grades.Count());
        }
        [Fact]
        public void GetGradesReportForStudentByTeacher () {
            var getGradesReportForStudent = new GetGradesReportVm(){
                StudentId = 5,
                GetterUserId = 1
            };
            var gradesReport = _gradeService.GetGradesReportForStudent(getGradesReportForStudent);
            Assert.NotNull(gradesReport);
        }
        [Fact]
        public void GetGradesReportForStudentByStudent () {
            var getGradesReportForStudent = new GetGradesReportVm(){
                StudentId = 5,
                GetterUserId = 5
            };
            var gradesReport = _gradeService.GetGradesReportForStudent(getGradesReportForStudent);
            Assert.NotNull(gradesReport);
        }
        [Fact]
        public void GetGradesReportForStudentByParent () {
            var getGradesReportForStudent = new GetGradesReportVm(){
                StudentId = 5,
                GetterUserId = 3
            };
            var gradesReport = _gradeService.GetGradesReportForStudent(getGradesReportForStudent);
            Assert.NotNull(gradesReport);
        }
    }
}
```

3) W celu uruchomienia wszystkich testów jednostkowych w całym projekcie proszę w terminalu wykonać poniższe polecenie.

```
dotnet test
```

```
Passed!  - Failed:     0, Passed:    27, Skipped:     0, Total:    27, Duration: 817 ms
```

Aby wykonać pojedynczy test należy wykorzystać parametr `-filter`

```
dotnet test --filter {nazwa_metody}
```

Przykład:

```
dotnet test --filter GetGradesReportForStudentByTeacher
```

Aby wykonać wszystkie testy dla danej klasy testowej, należy podać nazwę klasy, np.

```
dotnet test --filter GradeServiceUnitTests
```

3. Zadanie

Po zapoznaniu się z powyższą instrukcją, proszę zaprojektować oraz zaimplementować pozostałe testy jednostkowe, w taki sposób, aby pokryć testami wszystkie klasy oraz metody usług. Proszę wziąć pod uwagę wszystkie potencjalne scenariusze testowe.