



Kauno technologijos universitetas

Informatikos Fakultetas

Programų Sistemų Testavimas (T120B162)

2-ojo laboratorinio darbo ataskaita

Atliko IFF-2/1 grupės studentai:

Rokas Gudžiūnas

Gedmantas Šilinskas

Rugilė Jovaišaitė

Simona Gerikaitė

Priėmė:

doc. prakt. Guogis Evaldas

Turinys

Laboratorinio darbo tikslas.....	3
Uždaviniai	3
Aprašymas	3
Pasiruošimas testavimui	3
Testuojamas metodas	7
Pirmas testas	8
Antras testas	9
Trečias testas	9
Ketvirtas testas	10
Testų rezultatai	11
Išvados.....	11

Laboratorinio darbo tikslas

Šio laboratorinio darbo tikslas yra ištestuoti kuriamos sistemos komponentus, sukuriant vienetų (unit) testus.

Uždaviniai

1. Išsiaiškinti, kaip aprašomi bei sukuriami vienetų testai ABP aplinkoje.
2. Sukurti vienetų testus, kurie tikrina programinės įrangos funkcionalumą.

Aprašymas

Programinės įrangos testavimas – tai procesas, skirtas nustatyti programinės įrangos, kurioje vis dar vyksta korekcijos, tikslumą. Testuojant siekiama aptikti programinės įrangos defektus. Šie testai užtikrina, kad siunčiami duomenys grąžina tikėtus rezultatus.

Vienetų testai, žinomi kaip „unit tests“, yra testai, skirti patikrinti atskirų programos komponentų funkcionalumą. Šie vienetai gali būti funkcijos, metodai arba klasės. Vienetų testavimo tikslas yra užtikrinti, kad kiekvienas programos komponentas veiktų kaip numatyta atskirai nuo kitų sistemų dalių.

Pasiruošimas testavimui

Prieš pradėdant rašyti vienetų testus ABP aplinkoje, buvo svarbu tinkamai susipažinti su dokumentacija apie vienetų rašymą ABP karkase. Pasirinkome testuoti ExerciseAppService serviso metodą `CreateOrUpdateExerciseAsync`, kuris yra atsakingas už naujų pratimų sukūrimą ar egzistuojančių pratimų atnaujinimą mūsų sistemoje. Šis metodas yra svarbi mūsų sistemos dalis, todėl yra svarbu, kad būtų užtikrintas jo tinkamas veikimas.

Vienetų testai buvo generuojami generatyvinio dirbtinio intelekto “ChatGPT” (Generative AI) pagalba. Sugeneruoti vienetų testai atrodė šitaip:

```
using Shouldly;  
  
using System;  
  
using System.Threading.Tasks;  
  
using Volo.Abp.Domain.Repositories;  
  
using Volo.Abp.Identity;  
  
using Volo.Abp.Modularity;  
  
using Xunit;  
  
using Moq;
```

```

namespace Gymzii.Exercises;

/* This is just an example test class.
 * Normally, you don't test code of the modules you are using
 * (like IIdentityUserAppService here).
 * Only test your own application services.
 */

public class ExerciseAppService_Tests
{
    [Fact]
    public async Task Should_Update_Existing_Exercise_If_New_MaxWeight_Is_Greater()
    {
        // Arrange

        var input = new CreateUpdateExerciseDto { Name = "Test Exercise", MaxWeight =
100 };

        var existingExercise = new Exercise { Name = "Test Exercise", MaxWeight = 50 }; //
Existing exercise with lower weight

        var mockRepository = new Mock<IRepository<Exercise, Guid>>();

        mockRepository.Setup(r =>
r.FirstOrDefaultAsync(It.IsAny<Expression<Func<Exercise, bool>>>()))
.ReturnsAsync(existingExercise);

        var service = new ExerciseAppService(mockRepository.Object);

        // Act

        var result = await service.CreateOrUpdateExerciseAsync(input);

        // Assert

        result.ShouldNotBeNull();

        result.MaxWeight.ShouldBe(input.MaxWeight); // Should be the updated max weight
    }
}

```

```

    }

    [Fact]
    public async Task Should_Return_Existing_Exercise_If_New_MaxWeight_Is_Not_Greater()
    {
        // Arrange

        var input = new CreateUpdateExerciseDto { Name = "Test Exercise", MaxWeight =
50 }; // Same max weight as existing exercise

        var existingExercise = new Exercise { Name = "Test Exercise", MaxWeight = 50 };

        var mockRepository = new Mock<IRepository<Exercise, Guid>>();

        mockRepository.Setup(r =>
r.FirstOrDefaultAsync(It.IsAny<Expression<Func<Exercise, bool>>>()))

            .ReturnsAsync(existingExercise);

        var service = new ExerciseAppService(mockRepository.Object);

        // Act

        var result = await service.CreateOrUpdateExerciseAsync(input);

        // Assert

        result.ShouldNotBeNull();

        result.MaxWeight.ShouldBe(input.MaxWeight); // Should be the same as existing max
weight
    }

    [Fact]
    public async Task
Should_Insert_New_Exercise_If_No_Existing_Exercise_With_Same_Name()
    {
        // Arrange

        var input = new CreateUpdateExerciseDto { Name = "New Exercise", MaxWeight =

```

```

100 };

        Exercise insertedExercise = null; // For verifying InsertAsync is called

        var mockRepository = new Mock<IRepository<Exercise, Guid>>();

        mockRepository.Setup(r =>
r.FirstOrDefaultAsync(It.IsAny<Expression<Func<Exercise, bool>>>()))

                .ReturnsAsync((Exercise)null); // No existing exercise with the same name

        mockRepository.Setup(r => r.InsertAsync(It.IsAny<Exercise>()))

                .Callback<Exercise>(exercise => insertedExercise = exercise)

                .Returns(Task.CompletedTask);

        var service = new ExerciseAppService(mockRepository.Object);

        // Act

        var result = await service.CreateOrUpdateExerciseAsync(input);

        // Assert

        result.ShouldNotBeNull();

        insertedExercise.ShouldNotBeNull();

        insertedExercise.Name.ShouldBe(input.Name);

        insertedExercise.MaxWeight.ShouldBe(input.MaxWeight);

    }
}

```

Vis dėlto sugeneruoti vienetų testai buvo neteisingi, turėjo daug klaidų ir reikėjo atlikti pakeitimų – reikėjo testus pritaikyti ABP aplinkai.

```

using Gymzii.Exercises.Tests;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

```

```

using System.Threading.Tasks;

using Xunit;

namespace Gymzii.EntityFrameworkCore.Applications.Exercises;

[Collection(GymziiTestConsts.CollectionDefinitionName)]

public class EfCoreBookAppService_Tests :
    ExercisesAppService_Tests<GymziiEntityFrameworkCoreTestModule>

{

}

```

Testuojamas metodas

Toliau pateikiamas metodas sukuria naują pratimą, kuris susideda iš pratimo pavadinimo, raumenų grupės ir maksimalaus svorio. Jei sukuriamas visiškai naujas pratimas, jis yra įterpiamas tarp jau egzistuojančių pratimų. Jei sukurto pratimo pavadinimas sutampa su jau esančiu pratimu, naujas pratimas nepridedamas, tik pakeičiama maksimalaus svorio reikšmė, jei ji naujame pratime yra didesnė. Jei maksimalaus svorio reikšmė mažesnė arba lygi, duomenys nekeičiami ir grąžinamas senasis metodas.

Testuojamas metodas `CreateOrUpdateExerciseAsync(CreateUpdateExerciseDto input)`:

```

public async Task<ExerciseDto>
CreateOrUpdateExerciseAsync(CreateUpdateExerciseDto input)
{
    Exercise exercise;
    var existingExercise = await _exerciseRepository.FirstOrDefaultAsync(e
=> e.Name == input.Name);

    if (existingExercise != null)
    {
        if (input.MaxWeight > existingExercise.MaxWeight)
        {
            existingExercise.MaxWeight = input.MaxWeight;
            exercise = await
            _exerciseRepository.UpdateAsync(existingExercise);
        }
        else
        {
            return ObjectMapper.Map<Exercise,
            ExerciseDto>(existingExercise);
        }
    }
}

```

```

        else
        {
            exercise = ObjectMapper.Map<CreateUpdateExerciseDto,
Exercise>(input);
            await _exerciseRepository.InsertAsync(exercise);
        }

        return ObjectMapper.Map<Exercise, ExerciseDto>(exercise);
    }
}

```

Pirmas testas

Toliau pateikiamas pirmas testas `Should_Update_Existing_Exercise_If_New_MaxWeight_Is_Greater()`, kuris tikrina, ar atnaujinama egzistuojančio pratimo informacija, jei maksimalus svoris yra didesnis, negu buvo prieš tai:

```

namespace Gymzii.Exercises.Tests;
public abstract class ExercisesAppService_Tests<TStartupModule> :
Gymzii.ApplicationTestBase<TStartupModule>
    where TStartupModule : IAbpModule
{
    private readonly IExerciseAppService _userAppService;
    private readonly IRepository<Exercise, Guid> _userAppRepository;
    protected ExercisesAppService_Tests()
    {
        _userAppService = GetRequiredService<IExerciseAppService>();
        _userAppRepository = GetRequiredService<IRepository<Exercise,
Guid>>();
    }
    [Fact]
    public async Task
Should_Update_Existing_Exercise_If_New_MaxWeight_Is_Greater()
    {
        // Arrange
        //var exerciseAppService =
GetRequiredService<IExerciseAppService>();
        //var exerciseRepository =
GetRequiredService<IRepository<Exercise, Guid>>();

        var exerciseAppService = _userAppService;
        var exerciseRepository = Substitute.For<IRepository<Exercise,
Guid>>();

        var input = new CreateUpdateExerciseDto { Name = "Test
Exercise", Type = MuscleType.Biceps, MaxWeight = 100 };
        var existingExercise = new Exercise()
        {
            Name = "Test Exercise",
            Type = MuscleType.Biceps,
            MaxWeight = 50
        };
        exerciseRepository.FirstOrDefaultAsync(e => e.Name ==
input.Name).Returns(existingExercise);

        // Act
    }
}

```



```

        var result = await
exerciseAppService.CreateOrUpdateExerciseAsync(input);

        // Assert
        result.ShouldNotBeNull();
        result.MaxWeight.ShouldBe(input.MaxWeight);
    }

```

Antras testas

Toliau pateikiamas antras testas `Should_Return_Existing_Exercise_If_New_MaxWeight_Is_Not_Greater()`, kuris tikrina, ar yra grąžinamas egzistuojantis pratimas, jei maksimalus svoris nėra didesnis nei buvo prieš tai:

```

[Fact]
public async Task
Should_Return_Existing_Exercise_If_New_MaxWeight_Is_Not_Greater()
{
    // Arrange
    var exerciseAppService =
GetRequiredService<IExerciseAppService>();
    var exerciseRepository = Substitute.For<IRepository<Exercise,
Guid>>();

    var input = new CreateUpdateExerciseDto { Name = "Test
Exercise", Type = MuscleType.Triceps, MaxWeight = 50 };
    var existingExercise = new Exercise()
    {
        Name = "Test Exercise",
        Type = MuscleType.Triceps,
        MaxWeight = 50
    };
    exerciseRepository.FirstOrDefaultAsync(e => e.Name ==
input.Name).Returns(existingExercise);

    // Act
    var result = await
exerciseAppService.CreateOrUpdateExerciseAsync(input);

    // Assert
    result.ShouldNotBeNull();
    result.MaxWeight.ShouldBe(existingExercise.MaxWeight);
}

```

Trečias testas

Toliau pateikiamas trečias testas `Should_Insert_New_Exercise_If_No_Existing_Exercise_With_Same_Name()`, kuris tikrina, ar tarp jau egzistuojančių pratimų įterpiamas naujas pratimas, jei sutapties pavadinime nebuvo:

```

[Fact]
public async Task
Should_Insert_New_Exercise_If_No_Existing_Exercise_With_Same_Name()
{
    // Arrange
    var exerciseAppService =
GetRequiredService<IExerciseAppService>();
    var exerciseRepository = Substitute.For<IRepository<Exercise,
Guid>>();

    var input = new CreateUpdateExerciseDto { Name = "New
Exercise", MaxWeight = 100 };
    Exercise nullExercise = null;
    exerciseRepository.FirstOrDefaultAsync(e => e.Name ==
input.Name).Returns(nullExercise);

    // Act
    var result = await
exerciseAppService.CreateOrUpdateExerciseAsync(input);

    // Assert
    result.ShouldNotBeNull();
    result.Name.ShouldBe(input.Name);
    result.MaxWeight.ShouldBe(input.MaxWeight);
}
}

```

Ketvirtas testas

Toliau pateikiamas ketvirtas testas `Should_Correctly_Map_Fields_When_Inserting_New_Exercise()`, kuris tikrina, ar teisingai įterpiami duomenys į `Exercise` pratimą:

```

[Fact]
public async Task
Should_Correctly_Map_Fields_When_Inserting_New_Exercise()
{
    // Arrange
    var exerciseAppService = _userAppService;
    var exerciseRepository = Substitute.For<IRepository<Exercise,
Guid>>();

    var input = new CreateUpdateExerciseDto { Name = "Unique
Exercise", Type = MuscleType.Legs, MaxWeight = 150 };
    Exercise nullExercise = null;
    exerciseRepository.FirstOrDefaultAsync(e => e.Name ==
input.Name).Returns(nullExercise);

    exerciseRepository.InsertAsync(Arg.Do<Exercise>(exercise =>
{
    exercise.Name.ShouldBe(input.Name);
    exercise.Type.ShouldBe(input.Type);
    exercise.MaxWeight.ShouldBe(input.MaxWeight);
})).Returns(Task.FromResult(new Exercise()));
}
}

```

```

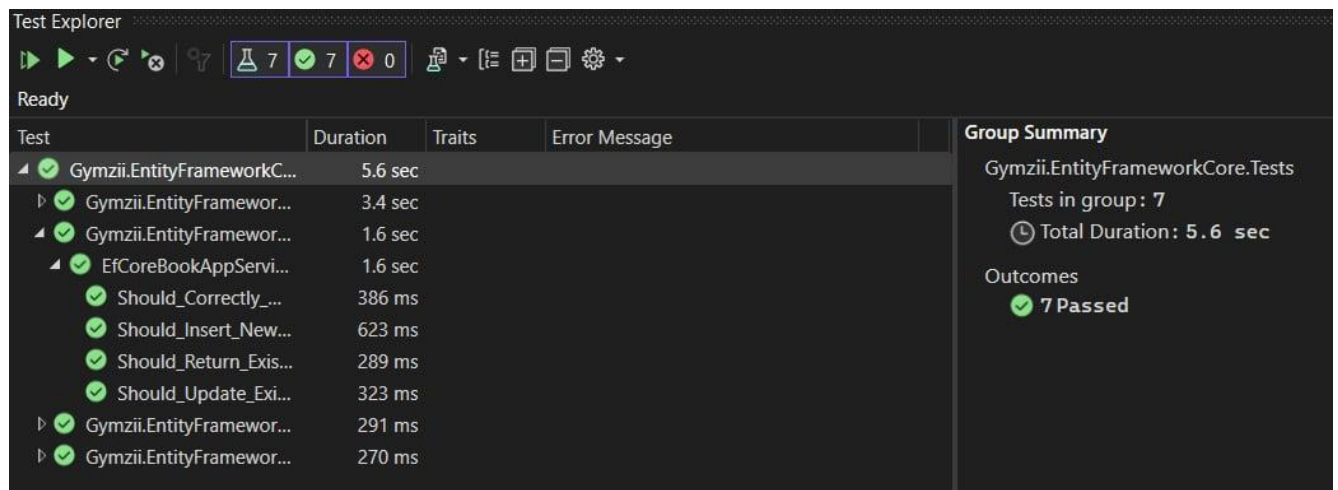
    // Act
    var result = await exerciseAppService.CreateOrUpdateExerciseAsync(input);

    // Assert
    result.ShouldNotBeNull();
}

```

Testų rezultatai

Vienetų testų sėkmingumą galime matyti 1 paveiksle. Matydami rezultatus, galime teigti, jog visi testai patvirtina metodo `CreateOrUpdateExerciseAsync(CreateUpdateExerciseDto input)` korektiškumą.



Test	Duration	Traits	Error Message	Group Summary
✓ Gymzii.EntityFrameworkC...	5.6 sec			Gymzii.EntityFrameworkCore.Tests Tests in group: 7 ⌚ Total Duration: 5.6 sec Outcomes ✓ 7 Passed
▶ ✓ Gymzii.EntityFramework...	3.4 sec			
▶ ✓ Gymzii.EntityFramework...	1.6 sec			
▶ ✓ EfCoreBookAppServi...	1.6 sec			
✓ Should_Correctly_...	386 ms			
✓ Should_Insert_New...	623 ms			
✓ Should_Return_Exis...	289 ms			
✓ Should_Update_Exi...	323 ms			
▶ ✓ Gymzii.EntityFramework...	291 ms			
▶ ✓ Gymzii.EntityFramework...	270 ms			

1 pav. Vienetų testų rezultatai

Išvados

1. Laboratorinio darbo metu buvo išsiaiškinta, kokius ABP automatizuotus vienetų testus naudoti.
2. Prieš testų realizavimą buvo išsiaiškinta, kaip reikia tinkamai aprašyti vienetų testus.
3. Testai buvo generuojami generatyvinio dirbtinio intelekto (angl. generative AI) „ChatGPT“ įrankio, tačiau pateiktas kodas nebuvo tikslus ir veikiantis, todėl buvo atlikta daug pakeitimų ranka.
4. Parašytas metodas veikia teisingai ir vienetų testai tai patvirtina.

5. Kadangi turėjome tik vieną su logika susijusį metodą, kurį pilnai ištestavome, galime teigti, jog kodo padengimas testais yra 100%.
6. Kurdami vienetų testus pastebėjome, jog šitoks testavimo tipas nėra pats efektyviausias mūsų kuriamam puslapiui, nes naudodamiesi tik juo galime ištestuoti tik mažą dalį viso kodo.