

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

**Laboratorinis darbas Nr. 2**

**Programų sistemų testavimas (T120B162)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | Darbą atliko:  Žilvinas Abromavičius IFF-4/3  Ernestas Venckus IFF-4/3  Darbą priėmė:  Doc. Šarūnas Packevičius  Dėst. Petras Tamošiūnas |

KAUNAS, 2017

Turinys

[1. Užduotis 3](#_Toc499994223)

[2. Darbo eiga 3](#_Toc499994224)

[2.1. Modulių testų automatinis generavimas ir kūrimas su grafine vartotojo sąsaja 3](#_Toc499994225)

[2.2. Programos kodo padengimas 3](#_Toc499994226)

[2.3. Modulių testų kūrimas rankiniu būdu 4](#_Toc499994227)

[2.4. Programos testų padengimas 6](#_Toc499994228)

[3. Išvados 6](#_Toc499994229)

# Užduotis

Pagrindinis šio laboratorinio darbo tikslas – susipažinti su modulių testavimu (angl. unit testing) bei pritaikyti tai pirmame laboratoriniame darbe pasirinktai testuojamai sistemai. Sugeneruoti testus, kad programos kodo padengimas pasiektų 100%. Kiekvienam projekto nariui sukurti modulių testus su grafine vartotojo sąsaja vienai pasirinktai klasei bei parašyti rankiniu būdu modulių testus vienai pasirinktai klasei. Įvertinti programos testų padengimą.

# Darbo eiga

Žemiau pateikta darbo eiga. Rankinio modulių testų rašymo darbo pasidalinimas:

* Žilvinas – „MovieRepository“, „ShowRepository“ klasės.
* Ernestas – „VoteRepository“, „MovieCreatorRepository“, „VotingRepository“ klasės.

## Modulių testų automatinis generavimas ir kūrimas su grafine vartotojo sąsaja

Automatiniam modulių testų generavimui buvo pasirinktas įrankis „IntelliTest“, kuris yra integruotas į Microsoft Visual Studio Enterprise Edition, tačiau šio įrankio pritaikyti nepavyko dėl šios priežasties:

* Projektas naudoja .NET Core 1.0, tuo tarpu IntelliTest įrankis veikia tik su .NET Framework.

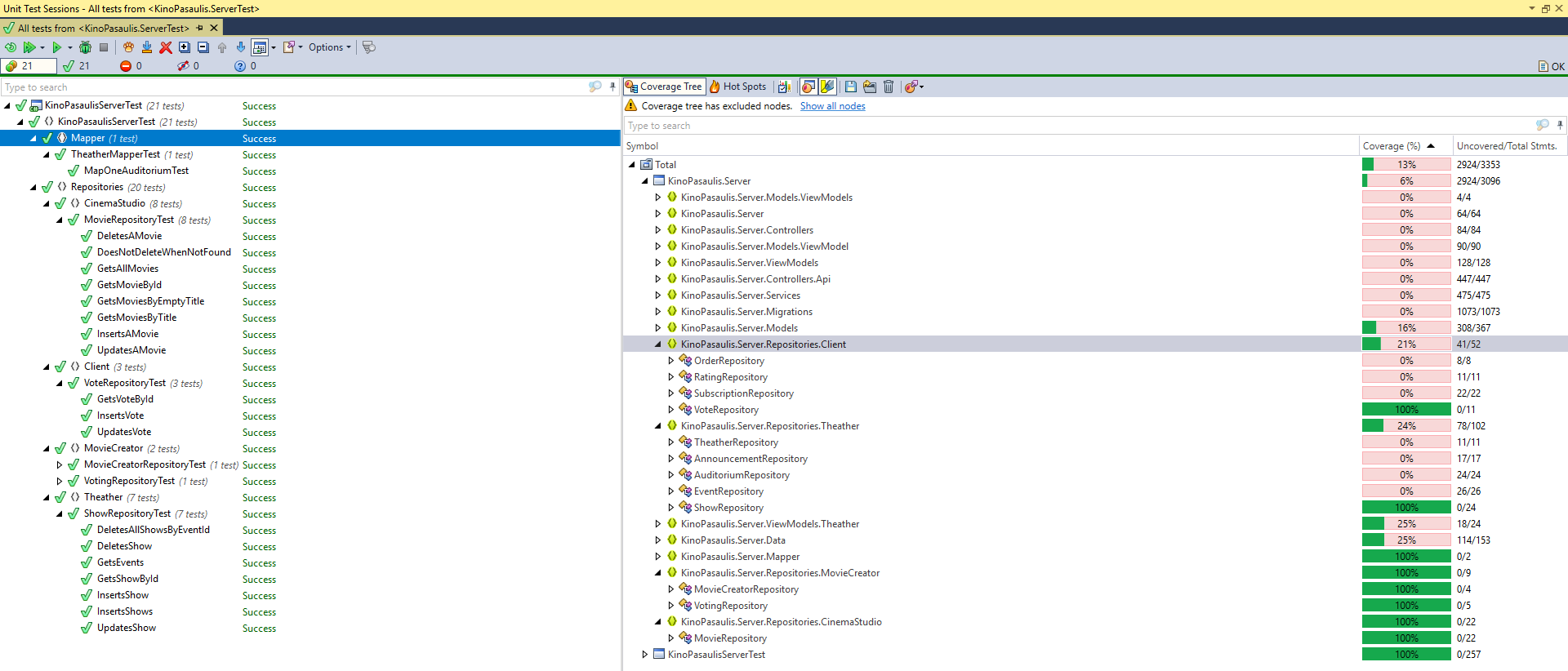
Buvo išsamiai ieškoma alternatyvių įrankių modulių testų automatiniam generavimui, tačiau veikiančių sprendimų nebuvo rasta.

Dėl šios priežasties sukurti modulių testų su grafine vartotojo sąsaja taip pat nepavyko.

## Programos kodo padengimas

Kadangi nebuvo įmanoma sugeneruoti modulių testų, tai 100% kodo padengimo buvo nuspręsta nesiekti, nes tam būtų reikėję investuoti labai daug laiko: būtų reikėję sukurti visai sistemai modulių testus rankiniu būdu.

Kodo padengimo tyrimui buvo naudojamas dot Cover įrankis. Buvo pasiektas 100% kodo padengimas tik tose klasėse, kurioms rankiniu būdu buvo sukurti modulių testai. Bendras programos padengimas 6%:



1 pav. Programos kodo padengimas (projektas KinoPasaulis.Server) yra 6%

## Modulių testų kūrimas rankiniu būdu

Rankiniu būdu kiekvienas komandos narys sukūrė modulių testus vienai pasirinktai klasėj. Buvo pasirinkta sukurti testus „MovieRepository“, „VoteRepository“, „ShowRepository“, „MovieCreatorRepository“ bei „VotingRepository“ klasėms. Visoms testuojamoms klasėms buvo pasiektas 100% kodo padengimas. Žemiau pateikti keletas testų pavyzdžių.

Modulių testų kūrimui buvo naudojamas xUnit karkasas. Taip pat buvo naudojamas Moq „mockinimo“ karkasas.

„MovieRepository“ klasei buvo sėkmingai sukurti modulių testai ir pasiektas šios klasės 100% kodo padengimas. Keletas testų pavaizduota žemiau.

|  |
| --- |
| [Fact]  public void GetsAllMovies()  {  var mockContext = GetApplicationDbContextMock();  var movieRepository = new MovieRepository(mockContext.Object);  var movies = movieRepository.GetMovies().ToList();  Assert.Equal(4, movies.Count);  Assert.Equal("Movie 1", movies[0].Title);  Assert.Equal("Movie 2", movies[1].Title);  Assert.Equal("Movie 3", movies[2].Title);  Assert.Equal("Another One", movies[3].Title);  }  [Fact]  public void InsertsAMovie()  {  var mockContext = GetApplicationDbContextMock();  var movieRepository = new MovieRepository(mockContext.Object);  var movie = new Movie {Title = "Movie to be inserted"};  movieRepository.InsertMovie(movie);  Assert.Equal(5, mockContext.Object.Movies.Count());  Assert.Equal("Movie to be inserted", mockContext.Object.Movies.SingleOrDefault(m => m.Title == "Movie to be inserted").Title);  } |

„VoteRepository“ klasei taip pat buvo sėkmingai sukurti modulių testai ir pasiektas šios klasės 100% kodo padengimas. Keletas testų pavaizduota žemiau.

|  |
| --- |
| [Fact]  public void GetsVoteById()  {  var mockContext = GetApplicationDbContextMock();  var voteRepository = new VoteRepository(mockContext.Object);  var vote = voteRepository.GetVoteById(2);  Assert.NotNull(vote);  Assert.Equal(2, vote.Id);  }  [Fact]  public void InsertsVote()  {  var mockContext = GetApplicationDbContextMock();  var dateTimeNow = DateTime.Now;  var voteRepository = new VoteRepository(mockContext.Object);  var vote = new Vote {Id = 4, VotedOn = dateTimeNow};  voteRepository.InsertVote(vote);  Assert.Equal(4, mockContext.Object.Votes.Count());  Assert.Equal(dateTimeNow, mockContext.Object.Votes.SingleOrDefault(v => v.Id == vote.Id).VotedOn);  } |

## Programos testų padengimas

Naudojant dot Cover įrankį buvo pasižiūrėta, koks yra testų padengimas. Buvo pastebėtas 100% modulių testų padengimas.



2 pav. Pasiektas 100% testų padengimas

# Išvados

Atliekant laboratorinį darbą buvo susidurta su įvairiomis problemomis, todėl buvo pasiekta tik dalis išsikeltų tikslų. Buvo atlikta kodo bei testų padengimo analizė su dot Cover įrankiu, rankiniu būdu parašyti modulių testai pasirinktoms klasėms, pasiektas tų klasių 100% kodo padengimas. Nepavyko sugeneruoti modulių testų automatiškai su IntelliTest įrankiu, dėl laiko stokos pasiekti 100% kodo padengimo, sugeneruoti modulių testų su grafine sąsaja.