**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**INFORMATIKOS FAKULTETAS**

**PROGRAMŲ SISTEMŲ TESTAVIMAS (T120B162)**

**Unit testai**

Atliko: IFF – 4/3 gr. studentas

Dovydas Petrutis

**KAUNAS 2017**

Turinys

[Turinys 2](#_Toc499391610)

[1. Įvadas 3](#_Toc499391611)

[2. Tikslas 3](#_Toc499391612)

[3. Ranka generuoti testai 3](#_Toc499391613)

[4. Testai 3](#_Toc499391614)

[4.1 CustomerController testai 3](#_Toc499391615)

[4.1.1 Testas 1 3](#_Toc499391616)

[4.1.2 Testas 2 4](#_Toc499391617)

[4.1.3 Testas 3 5](#_Toc499391618)

[4.2 Customer API testai 6](#_Toc499391619)

[4.2.1 Testas 1 6](#_Toc499391620)

[4.2.2 Testas 2 7](#_Toc499391621)

[4.2.3 Testas 3 8](#_Toc499391622)

[4.2.4 Testas 4 9](#_Toc499391623)

[4.2.5 Testas 5 10](#_Toc499391624)

[5. Generuoti testai 11](#_Toc499391625)

[5.1 CustomerController sugeneruotas testas 11](#_Toc499391626)

[5.2 Api.CustomerController sugeneruotas kodas 12](#_Toc499391627)

[6. Testų dengimas 13](#_Toc499391628)

[7. Išvados 13](#_Toc499391629)

# Įvadas

Užduotis yra prašyti ir sugeneruoti testus. Taip pat patikrinti kiek kodo dengia tie testai. Laboratorinio darbo metu parašyti 8 testai ir 2 testai sugeneruoti.

# Tikslas

Susipažinti su testų rašymų ir automatiniu jų generavimu.

# Ranka generuoti testai

Testai parašyti testuoti .net projekto veikimą, tai yra patikrinti kontrolerio ir API veikima, ar sukuriami atitinkami duomenys ir ar gaunami teisingi pranešimai naudojanti API. Testavimui naudotasi „Visual studio testing framework“ tai testavimo karkasas, kuris jau būna įdiegtas į visual studio, todėl papildomų diegimų nereikalauja ir juo paprasta naudotis, bei yra daug infomacijos. Ranka parašyti visi testai, kurie buvo aprašyti testavimo plane.

# Testai

## CustomerController testai

### Testas 1

1. Testas patikrina ar metodas “Details” grąžina sukurto kliento duomenis

|  |  |
| --- | --- |
| 1. Testavimo žingsniai | 1. Duomenys |
| Sukurti klientą ir jį paimti iš atmintyje laikomos duomenų bazės ir patikrinti ar jo duomenis. | Customer objektas kurį sudaro teisingai duomenys |

1. Tikėtinas rezultatas:

Sukurtas klientas randamas duomenų bazėje ir jo duomenys atitinka duomenis kurių tikimasi.

1. Gautas rezultatas:

Toks koks ir tikėtasi.

|  |
| --- |
| Testas |
| [TestClass]  public class CustomersControllerTests  {  [TestMethod]  public void Details\_Returns\_DBCustomer()  {  //Arange  DateTime dd = new DateTime(1995, 06, 10);  DateTime dd2 = new DateTime(1996, 06, 10);  DateTime dd3 = new DateTime(1997, 06, 10);  var cust = new Customer  {  Id = 1,  Name = "Jonas",  IsSubscribedToNewsletter = false,  MembershipTypeId = 2,  Birthdate = dd  };  Mock <ICustomerRepository> mock = new Mock<ICustomerRepository>();  mock.Setup(m => m.Customers).Returns(new Customer[]  {  new Customer {Id = 1, Name="Jonas", IsSubscribedToNewsletter=false,MembershipTypeId=2,Birthdate=dd },  new Customer {Id = 2, Name="Petras", IsSubscribedToNewsletter=true,MembershipTypeId=1,Birthdate=dd2 },  new Customer {Id = 3, Name="Antanas", IsSubscribedToNewsletter=false,MembershipTypeId=3,Birthdate=dd3 }  }.AsQueryable());  CustomersController controller = new CustomersController(mock.Object);  //Act  var actual = (Customer)controller.Details(1).Model;  string expectedStr = cust.Id +" "+cust.Name+" "+cust.IsSubscribedToNewsletter+" "+cust.MembershipTypeId+" "+cust.Birthdate;  string actualStr = actual.Id + " " + actual.Name + " " + actual.IsSubscribedToNewsletter + " " + actual.MembershipTypeId + " " + actual.Birthdate;  //Assert  Assert.AreEqual(expectedStr, actualStr);    }  } |

### Testas 2

1. Testas patikrina ar pagrindinis klientų puslapis grąžina klientų sarašo objektą

|  |  |
| --- | --- |
| 1. Testavimo žingsniai | 1. Duomenys |
| Sukurti vartotojus ir juos peržiūrėti | Sąrašas sudarytas ir customer ojbektų |

1. Tikėtinas rezultatas:

Klientai atvaizduojami o jų tipas klientų sarašas.

1. Gautas rezultatas:

Toks koks ir tikėtasi.

|  |
| --- |
| Testas |
| [TestClass]  public class CustomersControllerTests  {  [TestMethod]  public void Index\_Returns\_CustomerList()  {  //Arange  DateTime dd = new DateTime(1995, 06, 10);  DateTime dd2 = new DateTime(1996, 06, 10);  DateTime dd3 = new DateTime(1997, 06, 10);  Mock<ICustomerRepository> mock = new Mock<ICustomerRepository>();  mock.Setup(m => m.Customers).Returns(new Customer[]  {  new Customer {Id = 4, Name="Jonas", IsSubscribedToNewsletter=false,MembershipTypeId=2,Birthdate=dd },  new Customer {Id = 5, Name="Petras", IsSubscribedToNewsletter=true,MembershipTypeId=1,Birthdate=dd2 },  new Customer {Id = 6, Name="Antanas", IsSubscribedToNewsletter=false,MembershipTypeId=3,Birthdate=dd3 }  }.AsQueryable());  CustomersController controller = new CustomersController(mock.Object);  //Act    var actual = (List<Customer>)controller.Index().Model;  //Assert  Assert.IsInstanceOfType(actual,typeof(List<Customer>));  }  } |

### Testas 3

1. Testas patikrina ar buvo atnaujinti kliento duomenys.

|  |  |
| --- | --- |
| 2. Testavimo žingsniai | 3.Duomenys |
| Sukurti vartotoją ir redaguoti jo duomenys | Customer objektas kurį sudaro su teisingais duomenimis. |

1. Tikėtinas rezultatas:

Sukurtas klientas, jį redaguojant, duomenų bazėje pasikeičia jo duomenys

1. Gautas rezultatas:

Toks koks ir tikėtasi.

|  |
| --- |
| Testas |
| [TestClass]  public class CustomersControllerTests  {  [TestMethod]  public void Updating\_Customer()  {  //Arange  DateTime dd = new DateTime(1995, 06, 10);  DateTime dd2 = new DateTime(1996, 06, 10);  DateTime dd3 = new DateTime(1997, 06, 10);  var cust = new Customer {Id = 4, Name = null, IsSubscribedToNewsletter = false, MembershipTypeId = 10, Birthdate = dd};  var cust2 = new Customer {Id = 3, Name = "JonasKitas", IsSubscribedToNewsletter = true, MembershipTypeId = 4, Birthdate = dd2 };  Mock<ICustomerRepository> mock = new Mock<ICustomerRepository>();  mock.Setup(m => m.Customers).Returns(new Customer[]  {  new Customer {Id = 4, Name="Jonas", IsSubscribedToNewsletter=false,MembershipTypeId=2,Birthdate=dd },  new Customer {Id = 5, Name="Petras", IsSubscribedToNewsletter=true,MembershipTypeId=1,Birthdate=dd2 },  new Customer {Id = 6, Name="Antanas", IsSubscribedToNewsletter=false,MembershipTypeId=3,Birthdate=dd3 }  }.AsQueryable());  CustomersController controller = new CustomersController(mock.Object);  //Act  controller.Save(cust);  var actual = (Customer)controller.Details(4).Model;  string expectedStr = cust.Id + " " + cust.Name + " " + cust.IsSubscribedToNewsletter + " " + cust.MembershipTypeId + " " + cust.Birthdate;  string actualStr = actual.Id + " " + actual.Name + " " + actual.IsSubscribedToNewsletter + " " + actual.MembershipTypeId + " " + actual.Birthdate;    //Assert  Assert.AreEqual(expectedStr, actualStr);    }  } |

## Customer API testai

### Testas 1

1. Testas patikrinti ar gaunamas teisingas rezultatas, kai bandomas gauti /api/customers

|  |  |
| --- | --- |
| 2. Testavimo žingsniai | 3.Duomenys |
| Sukurti klientą  Bandyti gauti jo api duomenis | Customer objektas kurį sudaro su teisingais duomenimis. |

1. Tikėtinas rezultatas:

Sukurtas klientas, bandant gauti jo duomenis gaunamas OK http request pranšimas

1. Gautas rezultatas:

Toks koks ir tikėtasi.

|  |
| --- |
| Testas |
| [TestClass]  public class APITests  {  [TestMethod]  public void API\_GetCustomers\_OK()  {  //Arrange  var fakeCustomers = new List<CustomerDto>{  new CustomerDto{ Id = 1, Birthdate=DateTime.Now, IsSubscribedToNewsletter=false, MembershipTypeId = 1,Name="Jonas" },  new CustomerDto{ Id = 21, Birthdate=DateTime.Now, IsSubscribedToNewsletter=true, MembershipTypeId = 2,Name="Antanas" },  new CustomerDto{ Id = 3, Birthdate=DateTime.Now, IsSubscribedToNewsletter=false, MembershipTypeId = 3,Name="Pentas" }    };  var repository = new Mock<IAPICustomerRepository>();  repository  .Setup(\_ => \_.GetCustomers(It.IsAny<string>()))  .Returns(fakeCustomers)  .Verifiable();  var controller = new CustomersController(repository.Object);  OkResult ok = new OkResult(controller);  //Act  var result = controller.GetCustomers();  //Assert  repository.Verify();  Assert.ReferenceEquals(ok, result);  }  } |

### Testas 2

1.Testas patikrinti ar gaunamas teisingas rezultatas, kai bandomas gauni /api/customer/{id}

|  |  |
| --- | --- |
| 2. Testavimo žingsniai | 3.Duomenys |
| Sukurti klientą  Bandyti gauti jo api duomenis | Customer objektas kurį sudaro su teisingais duomenimis. |

4.Tikėtinas rezultatas:

Sukurtas klientas, bandant gauti jo duomenis gaunamas OK http request pranšimas

5. Gautas rezultatas:

Toks koks ir tikėtasi.

|  |
| --- |
| Testas |
| [TestClass]  public class APITests  {  [TestMethod]  public void API\_GetCustomer\_By\_Id\_OK()  {  //Arrange  var fakeCustomers = new List<CustomerDto>{  new CustomerDto{ Id = 1, Birthdate=DateTime.Now, IsSubscribedToNewsletter=false, MembershipTypeId = 1,Name="Jonas" },  new CustomerDto{ Id = 2, Birthdate=DateTime.Now, IsSubscribedToNewsletter=true, MembershipTypeId = 2,Name="Antanas" },  new CustomerDto{ Id = 3, Birthdate=DateTime.Now, IsSubscribedToNewsletter=false, MembershipTypeId = 3,Name="Pentas" }  };  var repository = new Mock<IAPICustomerRepository>();  repository  .Setup(\_ => \_.GetCustomer(1))  .Returns(fakeCustomers[1])  .Verifiable();  var controller = new CustomersController(repository.Object);  OkResult ok = new OkResult(controller);  //Act  var result = controller.GetCustomer(1);  //Assert  repository.Verify();  Assert.ReferenceEquals(ok, result);  }  } |

### Testas 3

1.Testas patikrinti ar gaunamas klaidingas pranešimas, kai bandoma gauti /api/customer/{id} su vienu id bet tikimasi kitokių rezultatų. (Pvz po duomenų atnaujinimo, patikrinti ar jie atsinaujino)

|  |  |
| --- | --- |
| 2. Testavimo žingsniai | 3.Duomenys |
| Sukurti klientą  Bandyti gauti jo api duomenis | Customer objektas kurį sudaro su teisingais duomenimis. Taip pat kitas objektas, kurio kitokie duomenys. |

4.Tikėtinas rezultatas:

Tikrinant kliento duomenis gaunamas http request pranešimas not found

5. Gautas rezultatas:

Toks koks ir tikėtasi.

|  |
| --- |
| Testas |
| [TestClass]  public class APITests  {  [TestMethod]  public void API\_GetCustomer\_By\_Id\_NOT\_OK()  {    //Arrange  var fakeCustomers = new List<CustomerDto>{  new CustomerDto{ Id = 1, Birthdate=DateTime.Now, IsSubscribedToNewsletter=false, MembershipTypeId = 1,Name="Jonas" },  new CustomerDto{ Id = 2, Birthdate=DateTime.Now, IsSubscribedToNewsletter=true, MembershipTypeId = 2,Name="Antanas" },  new CustomerDto{ Id = 3, Birthdate=DateTime.Now, IsSubscribedToNewsletter=false, MembershipTypeId = 3,Name="Pentas" }    };  var repository = new Mock<IAPICustomerRepository>();  repository  .Setup(\_ => \_.GetCustomer(2))  .Returns(fakeCustomers[2])  .Verifiable();  var controller = new CustomersController(repository.Object);  NotFoundResult NFR = new NotFoundResult(controller);  //Act  var result = controller.GetCustomer(1);  //Assert  Assert.ReferenceEquals(NFR, result);    }  } |

### Testas 4

1.Testas patikrinti ar gaunamas geras pranešimas, kai bandoma atnaujinti kažkokio kliento duomenis

|  |  |
| --- | --- |
| 2. Testavimo žingsniai | 3.Duomenys |
| Atnaujinti kliento duomenis, suvesti juos teisingai. | Customer objektas kurį sudaro su teisingais duomenimis. |

4.Tikėtinas rezultatas:

Atnaujinami kliento duomenys ir gaunamas OK pranešimas

5. Gautas rezultatas:

Toks koks ir tikėtasi.

|  |
| --- |
| Testas |
| [TestClass]  public class APITests  {  [TestMethod]  public void API\_UpdateCustomer\_OK()  {  //Arrange  var fakeCustomers = new List<CustomerDto>{  new CustomerDto{ Id = 1, Birthdate=DateTime.Now, IsSubscribedToNewsletter=false, MembershipTypeId = 1,Name="Jonas" },  new CustomerDto{ Id = 2, Birthdate=DateTime.Now, IsSubscribedToNewsletter=true, MembershipTypeId = 2,Name="Antanas" },  new CustomerDto{ Id = 3, Birthdate=DateTime.Now, IsSubscribedToNewsletter=false, MembershipTypeId = 3,Name="Pentas" },  new CustomerDto{ Id = 1, Birthdate=DateTime.Now, IsSubscribedToNewsletter=false, MembershipTypeId = 3,Name="Pentas" }  };  var repository = new Mock<IAPICustomerRepository>();  repository  .Setup(\_ => \_.UpdateCustomer(1,It.IsAny<CustomerDto>()))  .Returns(fakeCustomers[1])  .Verifiable();  var controller = new CustomersController(repository.Object);  OkResult ok = new OkResult(controller);  //Act  var result = controller.UpdateCustomer(1,fakeCustomers[3]);  //Assert  Assert.ReferenceEquals(ok, result);  }  } |

### Testas 5

1.Testas patikrinti ar gaunamas geras pranešimas, kai yra ištrinimas klientas.

|  |  |
| --- | --- |
| 2. Testavimo žingsniai | 3.Duomenys |
| Ištrinti egzistuojantį klientą | Reikalinga, kad būtų sukurtas nors vienas klientas. |

4.Tikėtinas rezultatas:

Ištrynus klientą gaunamas OK pranešimas

5. Gautas rezultatas:

Toks koks ir tikėtasi.

|  |
| --- |
| Testas |
| [TestClass]  public class APITests  {  [TestMethod]  public void API\_DeleteCustomer\_OK()  {  //Arrange  var fakeCustomers = new List<CustomerDto>{  new CustomerDto{ Id = 1, Birthdate=DateTime.Now, IsSubscribedToNewsletter=false, MembershipTypeId = 1,Name="Jonas" },  new CustomerDto{ Id = 2, Birthdate=DateTime.Now, IsSubscribedToNewsletter=true, MembershipTypeId = 2,Name="Antanas" },  new CustomerDto{ Id = 3, Birthdate=DateTime.Now, IsSubscribedToNewsletter=false, MembershipTypeId = 3,Name="Pentas" }  };  var repository = new Mock<IAPICustomerRepository>();  repository  .Setup(\_ => \_.DeleteCustomer(2))  .Returns(true)  .Verifiable();  var controller = new CustomersController(repository.Object);  OkResult ok = new OkResult(controller);  //Act  var result = controller.DeleteCustomer(fakeCustomers[1].Id);  //Assert  Assert.ReferenceEquals(ok, result);  }  } |

# Generuoti testai

Testų generavimui pasirinkau Unit Test Boilerplate Generator plėtinį Visual Studio. Šis įskiepis automatiškai sukuria testus nurodytoms klasėms.

## CustomerController sugeneruotas testas

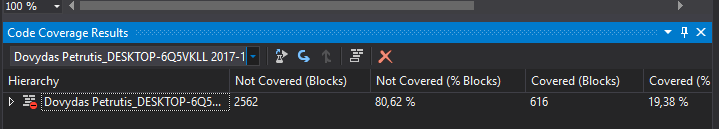
|  |
| --- |
| Testas |
| using Microsoft.VisualStudio.TestTools.UnitTesting;  using Moq;  using Vidly.Controllers;  using Vidly.Models;  namespace UnitTestProject2.Controllers  {  [TestClass]  public class CustomersControllerTests  {  private MockRepository mockRepository;  private Mock<ICustomerRepository> mockCustomerRepository;  [TestInitialize]  public void TestInitialize()  {  this.mockRepository = new MockRepository(MockBehavior.Strict);  this.mockCustomerRepository = this.mockRepository.Create<ICustomerRepository>();  }  [TestCleanup]  public void TestCleanup()  {  this.mockRepository.VerifyAll();  }  [TestMethod]  public void TestMethod1()  {  // Arrange  // Act  CustomersController customersController = this.CreateCustomersController();  // Assert  }  private CustomersController CreateCustomersController()  {  return new CustomersController(  this.mockCustomerRepository.Object);  }  }  } |

## Api.CustomerController sugeneruotas kodas

|  |
| --- |
| Testas |
| using Microsoft.VisualStudio.TestTools.UnitTesting;  using Moq;  using Vidly.Controllers.Api;  using Vidly.Models;  namespace UnitTestProject2.Controllers.Api  {  [TestClass]  public class CustomersControllerTests  {  private MockRepository mockRepository;  private Mock<IAPICustomerRepository> mockAPICustomerRepository;  [TestInitialize]  public void TestInitialize()  {  this.mockRepository = new MockRepository(MockBehavior.Strict);  this.mockAPICustomerRepository = this.mockRepository.Create<IAPICustomerRepository>();  }  [TestCleanup]  public void TestCleanup()  {  this.mockRepository.VerifyAll();  }  [TestMethod]  public void TestMethod1()  {  // Arrange  // Act  CustomersController customersController = this.CreateCustomersController();  // Assert  }  private CustomersController CreateCustomersController()  {  return new CustomersController(  this.mockAPICustomerRepository.Object);  }  }  } |

# Testų dengimas

Atlikus testus ir pažiūrėjus kiek jie padengia projekto matoma (1 pav.), kad testais patikrinama 19,38% kodo, kas yra ne daug, tačiau žinant, kad testuojamoje programoje yra 7 kontrolieriai, o iš jų buvo testuojamas tik vienas. Tai padengiama ganetinai nemažai kodo. Parašius testus ir kitiems kontroleriams, kurie būtų beveik identiški, testų kodo padengimo procentas smarkiai išaugtų,



1 pav. Test code coverage

# Išvados

Galbūt būtų įmanoma sugeneruoti 100% kodą dengiančius testus su Unit Test Boilerplate Generator, tačiau šių testų sukurtas šablonas nelabai padeda. Be to pagrindiniai šio projekto testai turėtų būti integraciniai, nes jie padeda ištestuoti visus programos sluoksnius ir bendra jos veikimą. Taigi, rašant testus ranka prireiktų daug laiko, o generavimas šiuo atveju neišeitis, todėl 100% dengiantys testai yra per didelė investicija. Taip pat man kol kas sunku prašyti testus, kuris dengtų didelę dalį kodo, nes kodo bazė nuolatos kinta.