TEST DE DESARROLLO

Microsoft Fakes es un nuevo framework de aislamiento que nos permite aislar el código a testear reemplazando otras partes de la aplicación con stubs o shims. Nos permite testear partes de nuestra solución incluso si otras partes de nuestra aplicación no han sido implementadas o aún no funcionan.

Las pruebas Mock son pruebas unitarias que utilizan objetos simulados ("mock") que sustituyen a los objetos reales utilizados por la clase o fragmento de código a probar.

TEST DE SERVICE ACCIONISTA

namespace TriunfadoresSysTest.Services

{

[TestFixture]

public class AccionistaTest

{

private Mock<TrinfadoresSysDataBase> entitiesMock;

[SetUp]

public void SetUp()

{

var db = AccionistaFakeDB();

var mockDbset = new Mock<IDbSet<Accionista>>();

mockDbset.Setup(x => x.Provider).Returns(db.Provider);

mockDbset.Setup(x => x.Expression).Returns(db.Expression);

mockDbset.Setup(x => x.ElementType).Returns(db.ElementType);

mockDbset.Setup(x => x.GetEnumerator()).Returns(db.GetEnumerator);

entitiesMock = new Mock<TriunfadoresSysDataBase>();

entitiesMock.Setup(x => x.accionista).Returns(mockDbset.Object);

}

[Test]

public void TestAddNewAccionista()

{

var service = new AccionistaService(entitiesMock.Object);

service.addAccionista (new Accionista {id=1, DNI= “22222222” Apellidos = "Mercado Carrera", nombreAccionista = "Lucía" Celular="975864212", Direccion= “Jr Huanuco 111”, Email=[flor@gmail.com](mailto:flor@gmail.com), fecha\_ingreso= “12/05/2012”, Nro\_Hijos= “1”, ocupación= “vendedor”, centro\_trabajo= “Mercado”, lugar\_nacimiento= “Cajamarca”});

}

public void TestReturnUnAccionista()

{

var service = new AccionistaService(entitiesMock.Object);

var result = service.getAccionistaById(1);

Assert.AreEqual("Lucía", result.nombreAccionista);

Assert.AreEqual("22222222", result.DNI);

}

TEST CONTROLLER ACCIONISTA

namespace TriunfadoresSysTest.Controller

{

[TestFixture]

public class AccionistaTest

{

[Test]

public void TestReturnViewCreateAccionista()

{

var mock = new Mock<IAccionistaService>();

var mockVal = new Mock<IValidacion>();

var mock\_ci = new Mock<ICiudadService>();

var controller = new AccionistaController(mock.Object, mockVal.Object, mock\_ci.Object);

var view = controller.Create() as ViewResult;

Assert.IsNotNull(view);

Assert.AreEqual("Create", view.ViewName);

Assert.IsNull(view.Model);

}

[Test]

public void TestAccionistaGuardaRedirectToIndex()

{

var mock = new Mock<IAccionistaService>();

var mockVal = new Mock<IValidacion>();

var accionista = new Accionista { nombreAccionista = "Vainsa" };

mockVal.Setup(x => x.validarDni(accionista)).Returns(true);

var controller = new AccionistaController(mock.Object, mockVal.Object, null);

var redirect = controller.Create(accionista) as RedirectToRouteResult;

Assert.IsNotNull(redirect);

}

[Test]

public void TestAccionistaReturnViewWhenFails()

{

var mock = new Mock<IAccionistaService>();

var mockVal = new Mock<IValidacion>();

var accionista = new Accionista { nombreAccionista = "Lucía" };

mockVal.Setup(x => x.validarDni(accionista)).Returns(false);

var controller = new AccionistaController(mock.Object, mockVal.Object, null);

var view = controller.Create(accionista) as ViewResult;

// Assert.IsNull(redirect);

Assert.IsNotNull(view);

Assert.AreEqual("Create", view.ViewName);

}

}

}

REGISTRAR PRÉSTAMO

namespace TriunfadoresSysTest.Services

{ [TestFixture]

public class PrestamoTest

{

private Mock<TriunfadoresSysDataBase> entitiesMock;

[SetUp]

public void SetUp()

{

var db = PrestamoFakeDB();

var mockDbset = new Mock<IDbSet<Prestamo>>();

var mockDbset = new Mock<IDbSet<GenerarCuota>>();

mockDbset.Setup(x => x.Provider).Returns(db.Provider);

mockDbset.Setup(x => x.Expression).Returns(db.Expression);

mockDbset.Setup(x => x.ElementType).Returns(db.ElementType);

mockDbset.Setup(x => x.GetEnumerator()).Returns(db.GetEnumerator);

entitiesMock = new Mock<TriunfadoresSysDataBase>();

entitiesMock.Setup(x => x.prestamo).Returns(mockDbset.Object);

}

[Test]

public void TestAddNewPrestamo ()

{

var service = new PrestamoService(entitiesMock.Object);

service.addPrestamo(new Prestamo { id=1,monto\_prestamo=1000, tasa\_interes=0.05, fecha\_registro=DateTime.Parse("2015-05-05”), cant\_cuotas= 2, fecha\_cuota1=DateTime.Parse("2015-06-05”), monto\_cuota1=500, interes\_cuota1=50, monto\_total\_cuota1=550, fecha\_cuota2=DateTime.Parse("2015-07-05”), monto\_cuota2=500 , interes\_cuota2=25, monto\_total\_cuota2=525,estado="pendiente" });

}

}

}