



**Universidad ORT Uruguay**

Facultad de Ingeniería

## **Documentación Obligatorio 1**

### **Evidencia de la aplicación de TDD y Clean Code**

#### **Diseño de Aplicaciones 2**

**Link al Repositorio:** [IngSoft-DA2/301178\\_231810\\_280070 \(github.com\)](https://github.com/IngSoft-DA2/301178_231810_280070)

**Integrantes:**

- Angelina Maverino - 280070
- Yliana Otero - 301178
- María Belén Rywaczuk - 231810

**Tutores:**

- Juan Ignacio Irabedra De Maio
- Juan Pablo Barrios García
- Ignacio Valle Dubé

# ÍNDICE

Descripción de la estrategia de TDD seguida (inside – out o outside - in).	3
Informe de cobertura para todas las pruebas desarrolladas.	5
Funcionalidades especificadas como prioritarias (*) evidencia mediante los commits de que se aplicó TDD mostrando la evolución del ciclo de TDD	6
Mantenimiento de cuentas de administrador	6
Creación de una empresa	8
Evidencia de TDD en service:	10
Evidencia de TDD en controller:	10
Detección de movimiento	10
Aquí se incorpora la funcionalidad inicial. Sin embargo, posteriormente realizamos una refactorización al identificar que se añadirían otras funcionalidades.	11
Asociar dispositivos al hogar	12

## Descripción de la estrategia de TDD seguida (inside – out o outside - in).

En el desarrollo guiado por pruebas (TDD), hay dos enfoques principales: inside-out y outside-in:

### Inside-Out:

Comienzas desarrollando las partes más internas o fundamentales del sistema, como los modelos o los servicios.

Luego, vas construyendo el resto de la aplicación, conectando estas capas internas hacia el exterior (controladores, interfaces de usuario).

Ejemplo: En una API, primero escribirías pruebas para los métodos de servicio o lógica de negocio, y después para los controladores.

### Outside-In:

Empiezas con los componentes más externos (como la interfaz de usuario o los controladores), escribiendo pruebas que describen cómo se espera que funcione la aplicación desde una perspectiva de usuario.

A partir de ahí, vas construyendo las capas internas necesarias para que la funcionalidad pase las pruebas externas.

Ejemplo: Primero creamos una prueba para verificar si una llamada a un endpoint de la API devuelve la respuesta esperada, y luego implementar la lógica del servicio para pasar esa prueba.

### Nuestra estrategia

Para el desarrollo de la API, comenzamos creando el **domain**, lo cual nos permitió definir claramente la lógica de negocio y las entidades fundamentales sin depender de la infraestructura o la interfaz externa. Esto facilitó la evolución y reutilización del código en futuras implementaciones.

Nos reunimos para elaborar un Excel con la lista de endpoints, lo cual nos proporcionó una visión clara de lo que íbamos a implementar y de los recursos necesarios para lograrlo.

Aunque los endpoints se desarrollaron al final del proceso, esta planificación anticipada nos permitió estructurar mejor el proyecto y priorizar las tareas.

Primero nos enfocamos en implementar los **services** y **models**, lo que permitió una base sólida antes de crear los endpoints. Por lo tanto, seguimos un enfoque **inside-out**, donde la lógica interna se desarrolla primero y luego se conecta con los controladores y puntos de entrada.

## Uso de mocks

En nuestro proyecto utilizamos **mocks** para simular el comportamiento de dependencias externas durante las pruebas unitarias, lo que nos permite centrarnos en probar el código en sí, sin depender de la implementación real de esas dependencias. Usamos la biblioteca **Moq** para crear estos mocks de manera sencilla.

Los mocks se generan definiendo el comportamiento esperado de ciertos métodos o propiedades de una clase o interfaz que queremos simular. Por ejemplo, si un servicio depende de una base de datos o una API externa, en lugar de acceder a la implementación real (que podría ser lenta o no estar disponible), creamos un mock que "imita" ese comportamiento con respuestas predefinidas. Esto nos permite aislar y probar el código bajo condiciones controladas.

Usamos **Moq** en nuestro proyecto principalmente para simular servicios como el acceso a datos o la lógica de negocio. Esto nos asegura que las pruebas sean consistentes y rápidas, y que se puedan ejecutar en cualquier entorno sin depender de conexiones reales o datos externos.

Además, los mocks son útiles para **test-driven development (TDD)**, ya que permiten validar los comportamientos esperados de las dependencias desde el inicio del desarrollo, antes de que las implementaciones reales estén completas.

## Informe de cobertura para todas las pruebas desarrolladas.

Dado que todo el desarrollo se realizó siguiendo la metodología TDD (Desarrollo Guiado por Pruebas), garantizamos que la cobertura de las pruebas unitarias alcanzara entre el 90% y el 100%. A continuación, se muestra una imagen que detalla el nivel de cobertura logrado.

▼  Total	<div><div>79%</div></div>	894/4...
>  IDataAccess	<div><div>100%</div></div>	0/7
>  CustomExceptions	<div><div>100%</div></div>	0/15
>  TestDataAccess	<div><div>99%</div></div>	1/131
>  TestWebApi	<div><div>99%</div></div>	8/647
>  Domain	<div><div>97%</div></div>	10/392
>  Model	<div><div>93%</div></div>	36/539
>  TestDomain	<div><div>92%</div></div>	37/476
>  TestService	<div><div>91%</div></div>	48/547
>  BusinessLogic	<div><div>90%</div></div>	32/335
>  WebApi	<div><div>60%</div></div>	156/389
>  DataAccess	<div><div>23%</div></div>	557/724
>  ServiceFactory	<div><div>0%</div></div>	9/9

▼  WebApi	<div><div>60%</div></div>	156/389
▼  WebApi	<div><div>64%</div></div>	131/364
>  Controllers	<div><div>99%</div></div>	1/234
>  Filters	<div><div>0%</div></div>	121/121
>  Attributes	<div><div>0%</div></div>	9/9
>  Program	<div><div>0%</div></div>	25/25

▼  DataAccess	<div><div>23%</div></div>	557/724
▼  DataAccess	<div><div>23%</div></div>	557/724
>  SqlRepository<T>	<div><div>100%</div></div>	0/84
>  SmartHomeContext	<div><div>83%</div></div>	17/100
>  Migrations	<div><div>0%</div></div>	540/540

Como se puede observar, todos los puntos alcanzan entre un 90% y 100% de cobertura. Es importante señalar que no se realizaron pruebas para las migraciones en la capa de acceso a datos (Data Access), ni para los filtros y atributos (Filter y Attributes).

## Funcionalidades especificadas como prioritarias (\*) evidencia mediante los commits de que se aplicó TDD mostrando la evolución del ciclo de TDD

Nuestro flujo de trabajo comenzó desarrollando las clases del dominio, seguido de los servicios que considerábamos necesarios para el endpoint a implementar, y finalmente procedimos con el desarrollo del propio endpoint. Este enfoque nos permitió ir de lo más específico a lo más general o abstracto, brindándonos una visión más clara de las necesidades del proyecto y permitiéndonos trabajar de manera organizada.















Todo este proceso se realizó utilizando Test-Driven Development (TDD). Prestamos especial atención a las funcionalidades marcadas con un asterisco, aunque todas las funcionalidades fueron implementadas bajo TDD. El proceso comenzaba escribiendo las pruebas unitarias utilizando mocks, lo que correspondía a la etapa RED del ciclo TDD. Luego, implementamos la funcionalidad esperada en la etapa GREEN, asegurándonos de cumplir con los requisitos. En muchos casos, las etapas GREEN y de refactorización se realizaron en el mismo commit, salvo algunas excepciones.

A lo largo del proyecto, realizamos commits separados para distinguir entre la etapa RED y la etapa GREEN/refactorización, evidenciando el uso de TDD. A continuación, se adjuntan ejemplos de commits con imágenes y descripciones que detallan el flujo de trabajo seguido.

### Mantenimiento de cuentas de administrador

Pruebas de uso de TDD en dominio Administrator:

#### DOMAIN

<b>[GREEN] administrator password</b>  MariaBelenR committed last month	bcfcd06  <>
<b>[RED] administrator password</b>  MariaBelenR committed last month	81e6bf0  <>
<b>[GREEN] administrator surname and email</b>  MariaBelenR committed last month	aad05cd  <>
<b>[RED] administrator email</b>  MariaBelenR committed last month	d06e4b8  <>
<b>[RED] administrator surname</b>  MariaBelenR committed last month	bed09c9  <>
<b>[GREEN] administrator name</b>  MariaBelenR committed last month	35342ea  <>
<b>[RED] administrator name</b>  MariaBelenR committed last month	1b689bb  <>

Ejemplo de commit [RED] (se realiza el Test que va estar en rojo ya que la funcionalidad todavía no está realizada)

```
SmartHome/DomainTest/AdministratorTest.cs
+19

1 + namespace DomainTest;
2 +
3 + [TestClass]
4 + public class AdministratorTest
5 + {
6 +     [TestMethod]
7 +     public void TestAddNameToAdministrator()
8 +     {
9 +         // Arrange
10 +         Administrator administrator = new Administrator();
11 +
12 +         // Act
13 +         administrator.Surname = "Juan";
14 +
15 +         // Assert
16 +         Assert.AreEqual("Juan", administrator.Surname);
17 +     }
18 +
19 + }
```

Ejemplo de commit [GREEN] (se implementa la funcionalidad mínima para que el test pase, luego se refactoriza)

```
SmartHome/Domain/Administrator.cs
+1

1 namespace Domain;
2
3 public class Administrator
4 {
5     public string Name { get; set; }
6 +     public string Surname { get; set; }
7 }
```

Se creó el UserService, el cual se conecta con el repositorio de usuarios. Para agregar o eliminar un administrador, se deben seguir procedimientos específicos dentro de este servicio.

<b>[GREEN] user service</b> AngelinaMaverino committed 3 weeks ago	8ea7a43		
<b>[RED] user service</b> AngelinaMaverino committed 3 weeks ago	1664e0f		

Se realizó el endpoint para agregar un administrador

<b>[GREEN] create admin</b> AngelinaMaverino committed 3 weeks ago	44acfb8d		
<b>[RED] create admin</b> AngelinaMaverino committed 3 weeks ago	6b8da21		
<b>[GREEN] create admin response</b> AngelinaMaverino committed 3 weeks ago	448d4a6		
<b>[RED] create admin response</b> AngelinaMaverino committed 3 weeks ago	3ea6936		
<b>[GREEN] create admin request</b> AngelinaMaverino committed 3 weeks ago	8b45ba2		

Commits on Sep 18, 2024

<b>[RED] create admin request</b> AngelinaMaverino committed 3 weeks ago	7fd9cfb		
<b>[RED] create admin</b> AngelinaMaverino committed 3 weeks ago	dab3c82		

# Creación de una empresa

## DOMAIN

Aquí hay un ejemplo claro en donde realizamos REFACTOR, ya que mientras hacíamos device, nos dimos cuenta que sería mejor tener una clase Company, en vez de tener los atributos como nombre de company en la clase de Device.

add: CompanyName, CompanyRUT & CompanyLogoURL in SecurityCamera - GREEN	7997f57		
add: TestAddCompanyDataToSecurityCamera - RED	fabef5a		

▼ SmartHome/TestDomain/DevicesTest.cs		+17	■■■■■	...
60	60	// Assert		
61	61	Assert.AreEqual(photos, device.PhotoURLs);		
62	62	}		
63	+			
64	+	[TestMethod]		
65	+	public void TestAddCompanyDataToSecurityCamera()		
66	+	{		
67	+	// Arrange		
68	+	SecurityCamera device = new SecurityCamera();		
69	+			
70	+	// Act		
71	+	device.CompanyName = "SecurityCameras & Co.";		
72	+	device.CompanyRUT = "123456789";		
73	+	device.CompanyLogoURL = "https://example.com/logo.jpg";		
74	+			
75	+	// Assert		
76	+	Assert.AreEqual("SecurityCameras & Co.", device.CompanyName);		
77	+	Assert.AreEqual("123456789", device.CompanyRUT);		
78	+	Assert.AreEqual("https://example.com/logo.jpg", device.CompanyLogoURL);		
79	+	}		
63	80	}		

▼ SmartHome/Domain/SecurityCamera.cs		+4	■■■■■	...
↑...	@@ -8,4 +8,8 @@	public class SecurityCamera : IDevice		
8	8	public long Model { get; set; }		
9	9	public List<string> PhotoURLs { get; set; }		
10	10	public bool IsConnected { get; set; }		
11	+	public string CompanyName { get; set; }		
12	+	public string CompanyRUT { get; set; }		
13	+	public string CompanyLogoURL { get; set; }		
14	+			
11	15	}		

Aquí es donde se hace el REFACTOR:



add: Company is a class on its own - REFACTOR f7d9fb1 <>

YilianaOtero committed last month

```
SmartHome/Domain/Company.cs  +9  ...  
@@ -0,0 +1,9 @@  
1 + namespace Domain;  
2 +  
3 + public class Company  
4 + {  
5 +     public string Name { get; set; }  
6 +     public string RUT { get; set; }  
7 +     public string LogoURL { get; set; }  
8 +  
9 + }
```

```
SmartHome/Domain/SecurityCamera.cs   +1 -3  ...  
@@ -8,8 +8,6 @@ public class SecurityCamera : IDevice  
8 8     public long Model { get; set; }  
9 9     public List<string> PhotoURLs { get; set; }  
10 10    public bool IsConnected { get; set; }  
11 -    public string CompanyName { get; set; }  
12 -    public string CompanyRUT { get; set; }  
13 -    public string CompanyLogoURL { get; set; }  
11 +    public Company Company { get; set; }  
.. ..
```

Evidencia de TDD en service:

[REFACTOR] company and companyOwnerTest and [RED] update MariaBelenR committed last week	a5096eb  <>
[GREEN] createCompany MariaBelenR committed 2 weeks ago	22aa886  <>
[RED] createCompany MariaBelenR committed 2 weeks ago	6ac4da0  <>

Evidencia de TDD en controller:

[RED] post company MariaBelenR committed last week	54ade34  <>
[GREEN] post company MariaBelenR committed last week	99b919a  <>

## Detección de movimiento

### DOMAIN

add: TestAddMovementDetectionFunctionalityToSecurityCamera - GREEN

1d3fa18

YlianaOtero committed last month

Nota: Aquí hay un leve error de tipeo, este commit se refiere a [RED]

1 file changed +1 -0 lines changed



SmartHome/Domain/SecurityCamera.cs

+1

@@ -17,5 +17,6 @@ public class SecurityCamera : IDevice

```
17 17      public Company Company { get; set; }
18 18      public string Description { get; set; }
19 19      public LocationType LocationType { get; set; }
20 20      + public bool HasMovementDetection { get; set; }
20 21
21 22  }
```

```
104 +      bool hasMovementDetection = true;
105 +
106 +      device.HasMovementDetection = hasMovementDetection;
107 +
108 +      Assert.AreEqual(hasMovementDetection, device.HasMovementDetection);
```

add: SecurityCamera has a list of functionalities - REFACTOR

f37ec2e

YlianaOtero committed last month

add: HasMovementDetection is a property in SecurityCamera - GREEN

ac42356

YlianaOtero committed last month

Aquí se incorpora la funcionalidad inicial. Sin embargo, posteriormente realizamos una refactorización al identificar que se añadirían otras funcionalidades.

SmartHome/TestDomain/DevicesTest.cs

+3 -3

```
@@ -101,11 +101,11 @@ public void TestAddOutdoorLocationTypeToSecurityCamera()
101 101      [TestMethod]
102 102      public void TestAddMovementDetectionFunctionalityToSecurityCamera()
103 103      {
104 104      -      bool hasMovementDetection = true;
104 104      +      List<string> functionalities = new List<string> { "Movement Detection" };
105 105
106 106      -      device.HasMovementDetection = hasMovementDetection;
106 106      +      device.Functionalities = functionalities;
107 107
108 108      -      Assert.AreEqual(hasMovementDetection, device.HasMovementDetection);
108 108      +      Assert.AreEqual(functionalities, device.Functionalities);
109 109      }
110 110
111 111  }
```

## Asociar dispositivos al hogar

Aquí se muestra cómo se crea un test para cada flujo, incluyendo los casos en los que se lanzan excepciones.

<b>[GREEN] cannot delete device</b> AngelinaMaverino committed 3 weeks ago	1188302		
<b>[RED] cannot delete device</b> AngelinaMaverino committed 3 weeks ago	cfcdd08		
<b>[GREEN] cannot find device</b> AngelinaMaverino committed 3 weeks ago	52dc29f		
<b>[RED] cannot find device</b> AngelinaMaverino committed 3 weeks ago	c854f4e		
<b>[RED] find device</b> AngelinaMaverino committed 3 weeks ago	9d0ee60		
<b>[GREEN] try to add existing device</b> AngelinaMaverino committed 3 weeks ago	4312584		
<b>[RED] try to add existing device</b> AngelinaMaverino committed 3 weeks ago	42fa760		
<b>[GREEN] add device</b> AngelinaMaverino committed 3 weeks ago	f7dbd6c		
<b>[RED] add device</b> AngelinaMaverino committed 3 weeks ago	a8e87ae		

<b>[REFACTOR] homeControllerTest and add putdevices and tests</b> MariaBelenR committed last week	a656ac8		
--	---------	--	--

Commits on Sep 27, 2024

<b>[RED] put devices in homeservice</b> MariaBelenR committed last week	08ea5a1		
--	---------	--	--

Aquí se adjunta la parte [RED] con la [GREEN] en el service.

```

SmartHome/TestService/HomeServiceTest.cs
+20

211 + [TestMethod]
212 + public void TestPutDevicesInHome()
213 + {
214 +     HomeService homeService = new HomeService(_mockHomeRepository.Object);
215 +     Home home = new Home(homeOwnerId, Street, DoorNumber, Latitude, Longitude);
216 +     _mockHomeRepository.Setup(x => x.GetById(1)).Returns(home);
217 +
218 +     List<Device> homeDevices = new List<Device>
219 +     {
220 +         new WindowSensor
221 +         {
222 +             Id = 1, Name = "Sensor de ventana", Model = 456, Description = "Sensor para ventanas", IsConnected = false ,
223 +         },
224 +         new SecurityCamera { Id = 1, Name = "Cámara de seguridad", Model = 123, Description = "Cámara para exteriores",
225 +             IsConnected = true }
226 +     };
227 +     homeService.PutDevicesInHome(1, homeDevices);
228 +     Assert.AreEqual(home.Devices, homeDevices);
229 + }
230 +

```

```
SmartHome/BusinessLogic/HomeService.cs  +14 5/5 ...
@@ -75,4 +75,18 @@ public Home GetHomeById(long id)
75 75
76 76         return home;
77 77     }

78 +
79 +     public Home PutDevicesInHome(long homeId, List<Device> homeDevices)
80 +     {
81 +         Home home = homeRepository.GetById(homeId);
82 +         if (home == null)
83 +         {
84 +             throw new ElementNotFound/HomeNotFoundMessage);
85 +         }
86 +
87 +         home.Devices = homeDevices;
88 +
89 +         homeRepository.Update(home);
90 +         return home;
91 +     }
78 92     }
```

Aquí se adjunta la parte [RED] con la [GREEN] en el endpoint.

```
SmartHome/TestWebApi/Controllers/HomesControllerTest.cs  +15 5/5 ...
@@ -169,6 +169,21 @@ public void TestGetMembersFromHomeNotFoundStatusCode()
169 169
170 170         Assert.IsInstanceOfType(result, typeof(NotFoundObjectResult));
171 171     }

172 +
173 +     [TestMethod]
174 +     public void TestPutDevicesInHomeOkStatusCode()
175 +     {
176 +         HomeDevicesRequest request = new HomeDevicesRequest()
177 +         {
178 +             Devices = new List<DeviceRequest>(),
179 +             WindowSensors = new List<WindowSensorRequest>(),
180 +             SecurityCameras = new List<SecurityCameraRequest>()
181 +         };
182 +
183 +         ObjectResult? result = _homeController.PutDevicesInHome(request) as OkObjectResult;
184 +
185 +         Assert.AreEqual(200, result!.StatusCode);
186 +     }

172 187
173 188     private HomeResponse DefaultHomeResponse()
```

SmartHome/WebApi/Controllers/HomeController.cs +9 -1

...

@@ -74,5 +74,13 @@ public IActionResult GetHomeById([FromRoute] long id)

74 74

75 75       return Ok(homeResponse);

76 76     }

77 -

77 +

78 +     [HttpPut]

79 +     [Route("{id}/devices")]

80 +     [RolesWithPermissions(RoleWithPermissions)]

81 +     public IActionResult PutDevicesInHome([FromRoute] long id, [FromBody] HomeDevicesRequest request)

82 +     {

83 +         \_homeService.PutDevicesInHome(id, request.ToEntity());

84 +         return Ok();

85 +     }

78 86     }