

Universidad ORT Uruguay

Facultad de Ingeniería

Diseño de Aplicaciones 2

Obligatorio 1:

Evidencia TDD y Clean Code

Juliette Ruchel - 203942

Francisco Martinez - 233126

Docente: **Ignacio Valle**

Entregado como requisito de la materia Diseño de
Aplicaciones 2

<https://github.com/ORT-DA2/203942-233126>

14 de mayo de 2020

Declaraciones de autoría

Nosotros, Juliette Ruchel y Francisco Martinez, declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizábamos Diseño de aplicaciones 2;
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad;
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;
- En la obra, hemos acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Resumen

El presente documento tiene el propósito de exponer el usos de buenas practicas de codigo limpio, el desarrollo utilizando la metodologia de *Test Driven Development (TDD)* y sus resultados respecto a la cobertura del sistema.

Índice general

1. Test Driven Development	2
1.1. Evidencia TDD	2
1.2. Cobertura de las pruebas	3
2. Clean Code	6

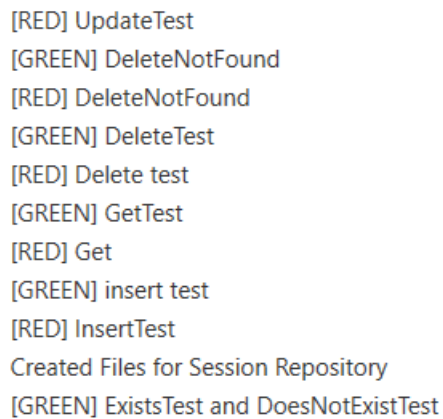
1. Test Driven Development

Como metodología de desarrollo se utilizo Test Driven Development, lo que implica desarrollar pruebas unitarias previamente al desarrollo del código funcional, para obtener un sistema de mayor confiabilidad y robustez. Si bien TDD no garantiza que no existan bugs, ayuda a reducir su existencia y verificar que las funcionalidades requeridas sean cumplidas por el código desarrollado.

Nuestro desarrollo se realizo de manera inside-out, es decir, nos enfocamos en las capas de menor nivel primero hasta llegar a las de mas alto nivel. Empezamos por el Dominio, luego el Data Access, la Lógica y finalmente la capa de Web API.

A continuación se pueden observar imágenes de los commits realizados sobre esas capas.

1.1. Evidencia TDD



```
[RED] UpdateTest  
[GREEN] DeleteNotFound  
[RED] DeleteNotFound  
[GREEN] DeleteTest  
[RED] Delete test  
[GREEN] GetTest  
[RED] Get  
[GREEN] insert test  
[RED] InsertTest  
Created Files for Session Repository  
[GREEN] ExistsTest and DoesNotExistTest
```

Figura 1.1: Commits en Data Access

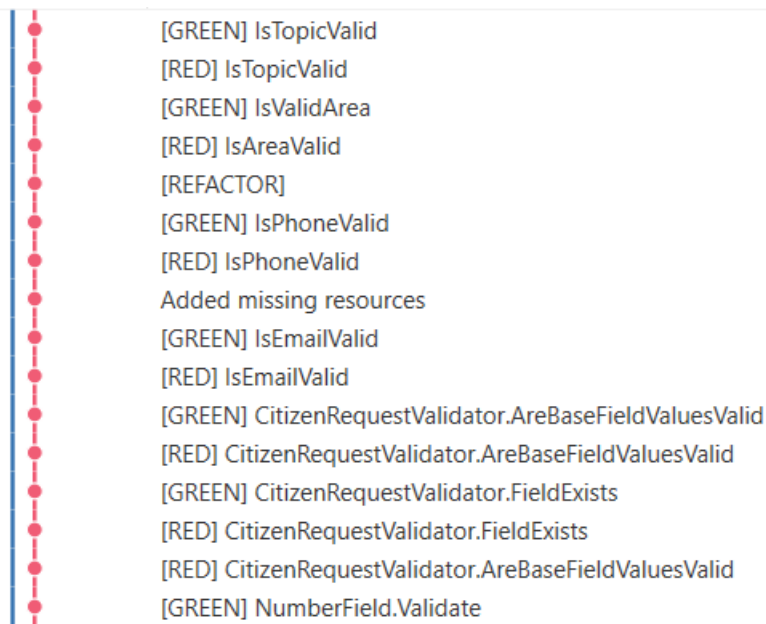


Figura 1.2: Commits en Logica

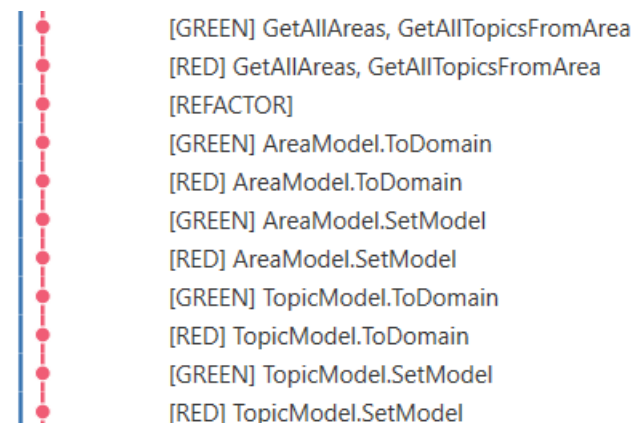


Figura 1.3: Commits en Web Api

1.2. Cobertura de las pruebas

Para obtener la cobertura del código se utilizó la funcionalidad de Visual Studio 2019 Enterprise, herramienta provista por la universidad. Se agregó un archivo de configuración para poder personalizar el análisis y evitar que se analicen los paquetes de Tests y excluimos los paquetes de Migrations y Resources.

La razón para excluir los paquetes de Migrations y Resources del análisis de cobertura, es que al ser código auto generado por herramientas desarrolladas por un tercero reconocido dentro de la industria y autor de las mismas -Microsoft-, nos sentimos en la libertad de confiar que es código confiable y funcional. Sin embargo, no significa que no se debería probar, pero creemos que el modo de prueba no deberían ser pruebas unitarias, sino otro tipo de pruebas como de integración que evalúen

que nuestro sistema se adapte correctamente a la herramienta. En conclusión, al determinar que estos paquetes no serian probados, se decidió excluirlos del análisis, ya que no realizan un aporte significativo al resultado.

A continuación el análisis general y por proyecto de cobertura.

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Usuario_DESKTOP-KGD1RJS 2020-05-13 18_31_48.co...	137	4.97%	2618	95.03%
immrequest.businesslogic.dll	10	1.57%	627	98.43%
immrequest.dataaccess.dll	0	0.00%	887	100.00%
immrequest.dataaccess.interfaces.dll	0	0.00%	2	100.00%
immrequest.domain.dll	10	3.44%	281	96.56%
immrequest.webapi.dll	117	12.47%	821	87.53%

Figura 1.4: Cobertura general

ierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Usuario_DESKTOP-KGD1RJS 2020-05-13 18_31_48.co...	137	4.97%	2618	95.03%
immrequest.businesslogic.dll	10	1.57%	627	98.43%
ImmRequest.BusinessLogic.Exceptions	0	0.00%	6	100.00%
ImmRequest.BusinessLogic.Extensions	0	0.00%	10	100.00%
ImmRequest.BusinessLogic.Helpers	0	0.00%	48	100.00%
ImmRequest.BusinessLogic.Logic	7	3.61%	187	96.39%
ImmRequest.BusinessLogic.Logic.Finders	0	0.00%	60	100.00%
ImmRequest.BusinessLogic.Validators	3	0.94%	316	99.06%

Figura 1.5: Cobertura Business Logic

erarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Usuario_DESKTOP-KGD1RJS 2020-05-13 18_31_48.co...	137	4.97%	2618	95.03%
immrequest.businesslogic.dll	10	1.57%	627	98.43%
immrequest.dataaccess.dll	0	0.00%	887	100.00%
ImmRequest.DataAccess.Context	0	0.00%	142	100.00%
ImmRequest.DataAccess.Repositories	0	0.00%	745	100.00%

Figura 1.6: Cobertura Data Access

erarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Usuario_DESKTOP-KGD1RJS 2020-05-13 18_31_48.co...	137	4.97%	2618	95.03%
immrequest.businesslogic.dll	10	1.57%	627	98.43%
immrequest.dataaccess.dll	0	0.00%	887	100.00%
immrequest.dataaccess.interfaces.dll	0	0.00%	2	100.00%
ImmRequest.DataAccess.Interfaces.Exceptions	0	0.00%	2	100.00%

Figura 1.7: Cobertura Data Access Interfaces

erarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Usuario_DESKTOP-KGD1RJS 2020-05-13 18_31_48.co...	137	4.97%	2618	95.03%
immrequest.businesslogic.dll	10	1.57%	627	98.43%
immrequest.dataaccess.dll	0	0.00%	887	100.00%
immrequest.dataaccess.interfaces.dll	0	0.00%	2	100.00%
immrequest.domain.dll	10	3.44%	281	96.56%
immrequest.webapi.dll	117	12.47%	821	87.53%
ImmRequest.WebApi	96	100.00%	0	0.00%
ImmRequest.WebApi.Controllers	7	2.26%	303	97.74%
ImmRequest.WebApi.Exceptions	0	0.00%	2	100.00%
ImmRequest.WebApi.Filters	2	5.00%	38	95.00%
ImmRequest.WebApi.Helpers	1	2.56%	38	97.44%
ImmRequest.WebApi.Helpers.Binders	8	16.67%	40	83.33%
ImmRequest.WebApi.Models	3	0.91%	326	99.09%
ImmRequest.WebApi.Models.UserManagement	0	0.00%	74	100.00%

Figura 1.8: Cobertura Web Api

erarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Usuario_DESKTOP-KGD1RJS 2020-05-13 18_31_48.co...	137	4.97%	2618	95.03%
immrequest.businesslogic.dll	10	1.57%	627	98.43%
immrequest.dataaccess.dll	0	0.00%	887	100.00%
immrequest.dataaccess.interfaces.dll	0	0.00%	2	100.00%
immrequest.domain.dll	10	3.44%	281	96.56%
immrequest.webapi.dll	117	12.47%	821	87.53%
ImmRequest.WebApi	96	100.00%	0	0.00%
ImmRequest.WebApi.Controllers	7	2.26%	303	97.74%
ImmRequest.WebApi.Exceptions	0	0.00%	2	100.00%
ImmRequest.WebApi.Filters	2	5.00%	38	95.00%
ImmRequest.WebApi.Helpers	1	2.56%	38	97.44%
ImmRequest.WebApi.Helpers.Binders	8	16.67%	40	83.33%
ImmRequest.WebApi.Models	3	0.91%	326	99.09%
ImmRequest.WebApi.Models.UserManagement	0	0.00%	74	100.00%

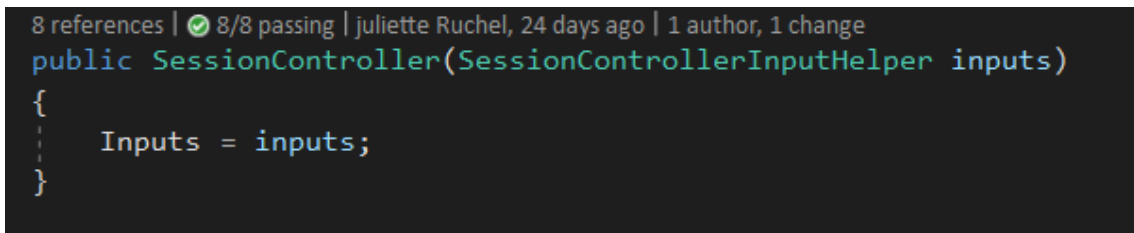
Figura 1.9: Cobertura Dominio

2. Clean Code

Se decidió utilizar como guía el Libro de Clean Code escrito por Robert C. Martin para los estándares utilizados durante la etapa de desarrollo. Se hizo énfasis en los siguientes puntos:

- Nombres de funciones y variables que dieran sentido y contexto al lector del código.
- Limitar la cantidad de parámetros que recibe un método o constructores de clases. En caso de necesitarlos todos, creamos DTOs para hacer que el código sea mas limpio.
- Se evito hardcodear valores de mensajes y usaron constantes y resources.
- Evitamos hacer RTTI
- Evitamos el uso de comentarios.
- Refactoreamos métodos a medida que crecían y buscamos que tuvieran una única responsabilidad. Procuramos mantener el mismo nivel de abstracción dentro de los métodos.
- Preferimos mayor cantidad de métodos por clase para mantener una sola responsabilidad y nivel de abstracción de los mismos.
- Aprovechamos las herramientas de polimorfismo del lenguaje para evitar y reducir el reuso de código utilizando herencias e interfaces.

A continuación se dan ejemplos de uso en el sistema:



```
8 references | 8/8 passing | juliette Ruchel, 24 days ago | 1 author, 1 change
public SessionController(SessionControllerInputHelper inputs)
{
    Inputs = inputs;
}
```

Figura 2.1: Uso de DTOs para reducir parámetros

```
1 reference | juliette Ruchel, 2 days ago | 1 author, 1 change
private void ValidateEmail(Administrator objectToValidate)
{
    ValidateEmailsIsNotEmpty(objectToValidate);
    ValidateEmailFormat(objectToValidate);
    ValidateEmailIsUnique(objectToValidate);
}
```

Figura 2.2: Ej. Mismo nivel de abstracción en métodos

```
6 references | 🟢 4/4 passing | juliette Ruchel, 26 days ago | 1 author, 3 changes
public bool IsValidToken(Guid token)
{
    return Repository.GetAll().Any(s => s.Token == token);
}
```

Figura 2.3: Ej. Única responsabilidad de un método