

# Universidad ORT Uruguay

## Facultad de Ingeniería

### Obligatorio 1 - Diseño de Aplicaciones 2

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

 **GitHub** <https://github.com/IngSoft-DA2-2023-2/266628-255981-271568>

Federico Rodriguez - 255981

Santiago Salinas - 266628

Manuel Morandi - 271568

**Tutores:** Daniel Acevedo

Rafael Alonso

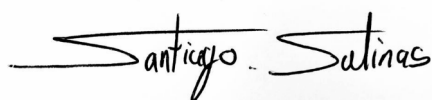
Franco Galeano

Septiembre 2023

# Declaración de autoría

Federico Rodriguez, Santiago Salinas y Manuel Morandi declaramos que el trabajo que se presenta en esta obra es de nuestra propia mano. Podemos asegurar que:

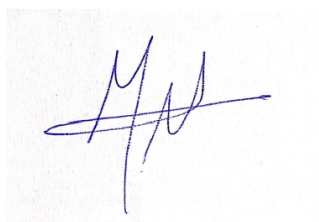
- La obra fue producida en su totalidad mientras construimos el primer obligatorio de Diseño de Aplicaciones 2
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra 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



Santiago Salinas 26/9/2023



Federico Rodriguez 26/9/2023



Manuel Morandi 26/9/2023

## Abstract

El documento provee explicación y evidencia de la aplicación de técnicas de TDD a través del desarrollo del obligatorio y seguimiento de las normas dadas por Clean Code.

Incluye a su vez evidencia del resultado de la ejecución de las pruebas escritas, evidencia del código de pruebas automáticas y reporte de la cobertura de código de dichas pruebas.

<b>Abstract</b>	<b>2</b>
<b>Test Driven Development</b>	<b>4</b>
<b>Clean Code</b>	<b>6</b>
<b>Informe de cobertura para todas las pruebas desarrolladas</b>	<b>7</b>
<b>Evidencia de TDD</b>	<b>9</b>
Promos	9
Usuario	10
Admin	11

# Test Driven Development

La idea del trabajo es desarrollar el sistema aplicando Test Driven Development (TDD). Este enfoque propone escribir los tests unitarios de la funcionalidad, para posteriormente escribir el código de producción mínimo y necesario para que esas pruebas pasen. De esta manera, logramos saber que el código escrito es correcto y necesario, evitando gold plating (codificar cosas innecesarias) y la aparición de bugs.

Entonces, podemos dividir el proceso de desarrollo con TDD en 3 etapas distintas. En una primera instancia, las pruebas unitarias fueron escritas, pero al correrlas fallan porque aún no fue escrito el código de producción; se trata de la etapa “Red”. Luego pasamos a la fase “Green”, donde, tras codificar la funcionalidad, las pruebas pasan correctamente. Finalmente, llegamos al momento de “Refactor”, cambiar el código (sin modificar su comportamiento) para emprolijar y cumplir estándares de calidad. Estas etapas se llevan a cabo de manera cíclica.

Las pruebas escritas deben ser FIRST. En palabras, deben ser rápidas de ejecutar, independientes las unas de las otras, repetibles en cualquier ambiente, deben validarse ellas mismas y deben ser escritas antes del código que van a probar. Esto presenta un problema, ya que nuestro sistema debe, por ejemplo, acceder a la base de datos, cosa que haría imposible los primeros tres puntos. Para solucionar esto, las pruebas hacen uso de mocks. En vez de acceder a la base de datos, asumimos que esta funcionará correctamente y nos limitamos a simular que a ella accedemos. Creamos un objeto de mock y le indicamos cómo debe comportarse en cada situación (cómo se comportaría la base de datos), para poder probar la funcionalidad en cuestión de manera rápida y repetible, sin tener que depender de otras pruebas para funcionar.

Ahora, para implementar TDD existen distintas estrategias. La primera estrategia es outside-in, la llamada escuela de Londres. Esta piensa desde los usuarios, lo que implica comenzar desde arriba, desde lo de más alto nivel de abstracción e ir “bajando” hacia el dominio y los elementos más específicos. Este no fue el camino tomado en esta oportunidad, sino que se llevó a cabo un desarrollo

más inside-out, de la escuela de Chicago donde primero se definen los objetos y la lógica de negocio para luego ir “subiendo” en niveles de abstracción hasta llegar a los elementos con los que los usuarios están más cerca. En otras palabras, se construyó desde la base, para llegar a la funcionalidad.

Una vez explicado el concepto de TDD, veamos cómo lo aplicamos. Se siguió el ciclo recomendado de desarrollo, antes de implementar una función se escribieron los tests, para luego implementar y refactorizar. Antes de pasar de una fase a otra, se realizaba un commit para poder revertir los cambios en caso de que sea necesario, en los casos que amerite. Utilizamos una convención para nombrar estos commits, incluyendo al inicio de su nombre la etapa en la que nos encontramos ([Red], [Green] o [Refactor]). También hemos utilizado un cuarto tag, [Fix], para cuando una funcionalidad posterior rompe una previa y requiere de cambios sencillos inmediatos.

Pese a que en general se siguió TDD en toda etapa del proceso, hubo algunas excepciones. En la creación del primer endpoint Products, no hicimos TDD, ya que al momento de hacerlo aún no habíamos dado mocks. Además este no es más que el ejemplo de ContosaPizza de Microsoft Learn, cambiando la clase tratada Pizza por Product. En futuros commits, y seguimiento del desarrollo, se crearon los debidos Tests de este controlador utilizando mocks, y agregando funcionalidades y protecciones de autenticación y autorización debidas siguiendo TDD.

# Clean Code

Se intentó que el código entregado cumpliera las prácticas recomendadas por Clean Code. Con esto se busca que el código producido sea entendible y fácilmente mantenible. Esto implica que cualquier otro desarrollador pueda ver el código y comprenderlo sencillamente, para poder modificarlo o expandirlo. Idealmente, debería ser tan simple y claro que se lee como prosa.

Para trabajar este aspecto, los nombres (de clases, métodos, paquetes y variables) buscan ser lo más explícitos posibles, dejando claro cuál es su objetivo o función. Los métodos y clases no deberían ser muy extensos ni tener muchos niveles de indentación, ya que es tedioso de leer y comprender. Adicionalmente, esto puede ser un indicio de que esa pieza de código está tomando muchas responsabilidades, por lo que incumple SRP y debería ser dividido en trozos más pequeños que tengan solo un objetivo (esto escapa de esta área de la documentación, se habla de esto y el resto de principios SOLID en “Descripción de diseño”).

En el punto anterior de TDD se habló de la etapa Refactor. Una de las metas de esta fase es asegurarnos que el código implementado cumple TDD. Así, podemos ver la aplicación de de prácticas de clean code en aquellos commits categorizados como [Refactor], pudiendo ver, por ejemplo, que seguimos los lineamientos de mantener variables privadas con `_nombre` , hacer uso de funciones auxiliares, cuidar la indentación, entre otras convenciones.

Es complicado saber si las técnicas de Clean Code aplicadas surten efecto, ya que lo que para el desarrollador que implementó algo puede ser claro, para el siguiente en usar el código puede resultar confuso. Para poder probar si el código está prolijo, por ejemplo, se puede dejar de lado el código implementado durante un par de días para luego, al revisarlo, ver si comprendemos su funcionamiento con facilidad. Otro método aplicado fue leer el código que los otros miembros del equipo escribieron, para ver si lo comprendemos.

# Informe de cobertura para todas las pruebas desarrolladas

Realizando TDD con las clases pertenecientes a los paquetes de Services y de Rest Api, se consiguió terminar con una cobertura mayor al 90% de todas las líneas de código para cada paquete. Dentro de cada paquete, la mayoría de clases tienen una cobertura mayor al 90%. Aquellas que no, es porque consideramos que eran métodos que no requerían tests especializados, nótese los métodos {get; set;} para las propiedades de las clases que simplemente asignan o retornan un valor guardado.

Hierarchy	Covered (%Lines)
Fede_DESKTOP-NQMBKEG_2023-10-03.21_49_14.coverage	33,10%
services.dll	95,08%
{ } Services.Models.Promos	92,00%
{ } Services.Models.Exceptions	100,00%
{ } Services.Models	94,42%
{ } Services.Exceptions	100,00%
{ } Services	96,18%
rest api.dll	75,93%
{ } Rest_Api.Filters	100,00%
{ } Rest_Api.DTOs	93,33%
{ } Rest_Api.Controllers	94,35%
{ } Clases globales	0,00%

La categoría de Clases globales abarca el Main del archivo Program.cs, el cual no se considera amerita un test:

{ } Clases globales	0,00%
Program	0,00%
<Main> \$(string[])	0,00%
Program.<>c	0,00%
Program.<>c__DisplayClass0_0	0,00%

La clase CredentialsDTO no tiene mayor cobertura simplemente por el atributo Token:

rest api.dll	75,93%
{ } Rest_Api.Filters	100,00%
{ } Rest_Api.DTOs	93,33%
UserDTO	100,00%
CredentialsDTO	81,82%
set_Token(string)	0,00%
set_Password(string)	100,00%
set_Email(string)	100,00%
get_Token()	0,00%
get_Password()	100,00%
get_Email()	100,00%
CredentialsDTO(string, string)	100,00%

```
namespace Rest_Api.DTOs
{
    8 referencias | Santiago, Hace 6 días | 1 autor, 1 cambio
    public class CredentialsDTO
    {
        2 referencias | Santiago, Hace 6 días | 1 autor, 1 cambio
        public string Email { get; set; }
        2 referencias | Santiago, Hace 6 días | 1 autor, 1 cambio
        public string Password { get; set; }
        0 referencias | Santiago, Hace 6 días | 1 autor, 1 cambio
        public string? Token { get; set; }

        3 referencias | 3/3 pasando | Santiago, Hace 6 días | 1 autor, 1 cambio
        public CredentialsDTO(string email, string password)
        {
            Email = email;
            Password = password;
        }
    }
}
```



Dentro de los modelos también se ven clases con menor cobertura únicamente por simples métodos {get; set;}

totalBookPromo.<>C_DisplayClass_0	0,00%
Purchase	75,00%
get_Cart()	0,00%
get_Date()	0,00%
get_Id()	100,00%
set_Id(int)	100,00%
Promo	83,33%
get_Condition()	0,00%
get_Discount()	0,00%
Promo(string, string, string)	100,00%
Product	90,48%
set_PriceUYU(double)	66,67%
Product()	100,00%
get_Id()	100,00%

En el caso de User, también falta cubrir los métodos Equals y StringIsNullOrEmpty, pero nuevamente se observa que la trivialidad de los métodos no dan necesidad de un test.

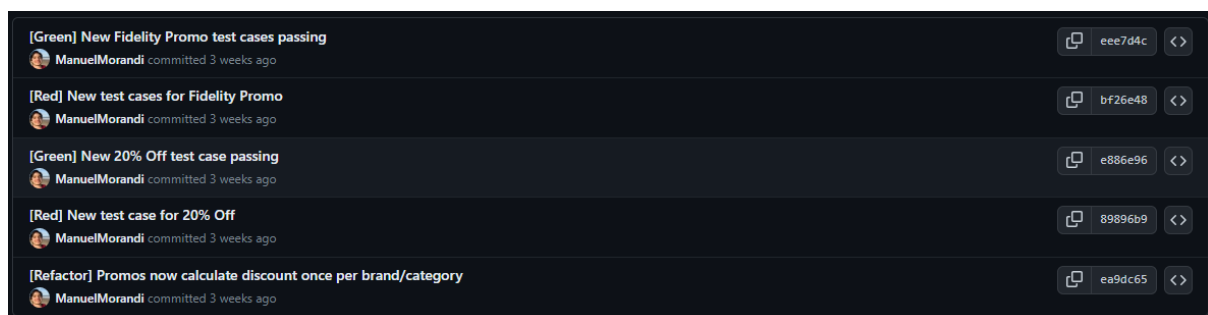
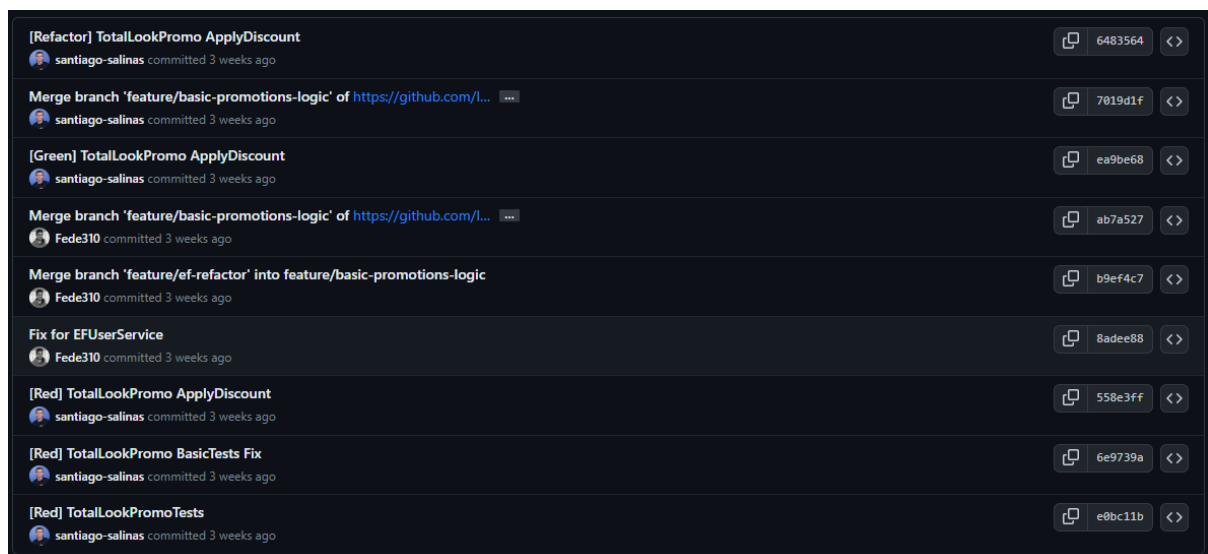
User	89,19%
get_Email()	0,00%
get_Password()	0,00%
get_Token()	0,00%
get_Address()	0,00%
Equals(object)	50,00%
StringIsNullOrEmpty(string)	75,00%
User(string, string, string, System.Collections.Generic.List<Sei	100,00%

# Evidencia de TDD

A continuación, se presentan algunos extractos donde se evidencia el uso de TDD. La letra pide evidenciar el uso de TDD en funcionalidades prioritarias. Esta evidencia será dada en forma de capturas de Github, mostrando los commits en las distintas etapas.

## Promos

El primer requerimiento prioritario es la implementación de las promos. Se adjuntan todos los commits referentes a esta funcionalidad, notando el uso y la aparición de las 3 etapas de TDD. Se crean las pruebas de la promoción para posteriormente implementarla y refactorizar, si es pertinente.



[Green] 20% Off tests passing	ManuelMorandi committed 3 weeks ago	9bbf3e2	<>
[Green] Fidelity Promo tests passing	ManuelMorandi committed 3 weeks ago	8f20b37	<>
[Red] Fidelity Promo tests	ManuelMorandi committed 3 weeks ago	6f06d7a	<>
[Refactor] Minor changes in 3x2Promo class and tests	ManuelMorandi committed 3 weeks ago	51f5c02	<>
[Green] 3x2 tests passing	ManuelMorandi committed 3 weeks ago	6d45773	<>
[Red] 3x1 tests added	ManuelMorandi committed 3 weeks ago	f78de90	<>

## Usuario

Las siguientes capturas refieren a la creación del filtro Auth, que permite llevar a cabo el Login de usuarios.

[Refactor] Placed filter in controllers	santiago-salinas committed last week	98ebe6e	<>
[Green] Can Create User and Get Token With Credentials	santiago-salinas committed last week	696b953	<>
[Red] Problems with id being generated by DB and not program	santiago-salinas committed last week	09d6ac1	<>
Migration	santiago-salinas committed last week	35dc802	<>
[Refactor] Run Code Cleanup Profile	santiago-salinas committed last week	b9491b3	<>
[Green] LoginController Tests	santiago-salinas committed last week	4364e90	<>

[Red] LoginController Tests	santiago-salinas committed last week	68b26b9	<>
[Refactor] Clean Coded nullable parameters in User	santiago-salinas committed last week	2b6ddaf	<>
[Green] Added password tests	santiago-salinas committed last week	4f25d92	<>
[Red] Added password tests	santiago-salinas committed last week	0591342	<>
Commits on Sep 23, 2023			
[Green] AuthFilter Tests	santiago-salinas committed last week	68db8a1	<>

A continuación vemos la definición de la clase UserController, que nos brinda el endpoint de usuarios, junto con toda su funcionalidad.

[Green] Deleted unnecessary role logic	ManuelMorandi committed last month	5461f72	<>
[Green] User Controller Tests passing	ManuelMorandi committed last month	d5c372c	<>

Added User Controller ManuelMorandi committed last month	4a35922	<>
[Green] User Service Tests passing ManuelMorandi committed last month	beae266	<>
[Red] User Service Tests ManuelMorandi committed last month	af5055c	<>
[Green] Deleted Both role ManuelMorandi committed last month	0819e2e	<>

## Admin

Las funciones exclusivas para administradores se implementaron para llevarse a cabo a través de un endpoint Admin. Antes de usar cualquiera de estos métodos, el AuthorizationFilter chequea que el usuario que hizo la request es Admin, por lo que está autorizado.

Merge branch 'develop' into feature/admin-authorization ManuelMorandi committed 4 days ago	Verified	8b07680	<>
[Green] Removed repeated operations not available for customers ManuelMorandi committed 4 days ago		09e35bc	<>

[Green] Added endpoint for Admin exclusive operations ManuelMorandi committed 5 days ago	2a4069e	<>
Exception filter Fede310 committed 5 days ago	356f0b9	<>
[Green] Added dependency injection for UserRepository ManuelMorandi committed 5 days ago	ebee8a0	<>
[Refactor] Changed the way UserEntity manages Roles ManuelMorandi committed 5 days ago	fd3bded	<>
[Green] Added AuthorizationFilter ManuelMorandi committed 5 days ago	9477daf	<>