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-DA2/233126-203942.git

10 de octubre de 2019

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 demostrar el uso de TDD y Clean Code en el desarrollo del sistema

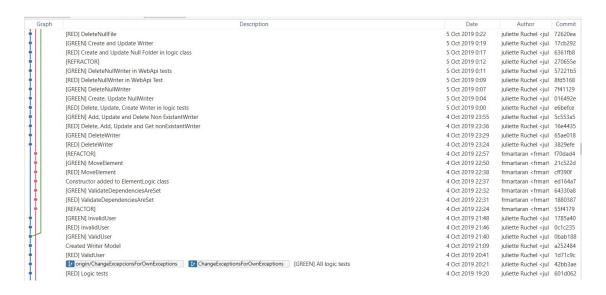
Índice general

1.	Test	Driven Development	2
	1.1.	Evidencia TDD	2
	1.2.	Cobertura de pruebas	2
2.	Clea	an Code	4

1. Test Driven Development

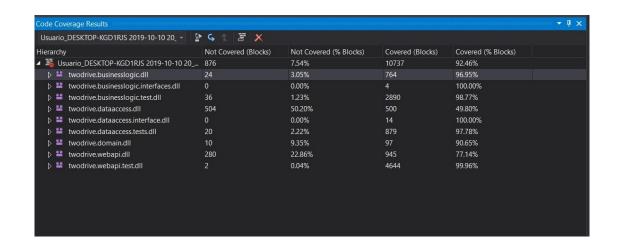
Implementamos TDD mientras desarrollabamos de manera inside - out. Ya que comenzamos programando los niveles mas bajos del sistema hasta llegar a la capa API.

1.1. Evidencia TDD



1.2. Cobertura de pruebas

Observando la cobertura obtuvimos alrededor de un 92% de cobertura del sistema. Tomando en cuenta que las migraciones y los filtros no pudieron ser testeados por su complejidad.



2. Clean Code

Para seguir los implementar clean code seguimos los siguientes estandares:

- Usamos nombres nemotécnicos para variables y metodos
- Tramamos de reducir la cantidad de parametros que reciben los metodos
- Evitamos hacer RTTI
- No usamos comentarios para hacer le codigo mas mantenible
- Realizamos refractor a medida que crecian los metodos
- Preferimos mayor cantidad de metodos por clase para mantener una sola responsabilidad y nivel de abstracción de los mismos.
- Tratamos de evitar la repeticion de codigo utilizando interfaces y herencias.