Universidad ORT Uruguay
Facultad de Ingeniería

Diseño de Aplicaciones 2
Obligatorio 1

Evidencia de Clean Code y de la aplicación de TDD

Diego Asadurian (198874)
Dominique Lachaise(214233)

Docentes: Gabriel Piaretti, Nicolas Fierro y Nicolas Hernandez

Declaraciones de autoría

Nosotros Diego Asadurian y Dominique Lachaise, declaramos que el trabajo que se presenta es este entregable es de nuestra autoridad, en lo cual podemos asegurar que:

- El entregable fue realizado en su totalidad mientras se cursaba la materia Diseño de Aplicaciones 2

Resumen

El presente documento se enmarca en un proyecto para implementar una WebApi para el Ministerio de Salud Pública. En dicho documento se implementarán capturas correspondientes a las pruebas unitarias y la evidencia de clean code.

Índice

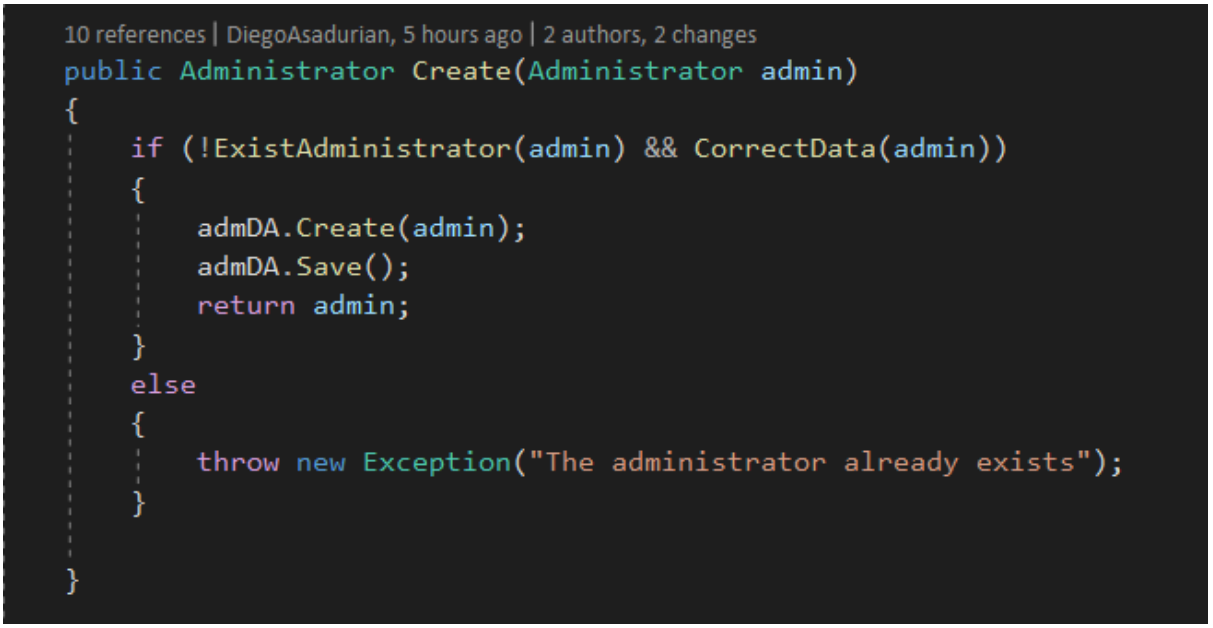
Evidencia de Clean Code

Para obtener un proyecto de alta calidad y mantener el mismo de forma ordenada y prolija se aplicaron los estándares de codigo de C# basandonos en las tecnicas de CleanCode, algunos de los cuales son:

- Métodos comienzan con Mayúscula
- Utilizamos CamelCase
- Utilizamos nombres nemotécnicos tanto para métodos como para variables
- Nombre de atributos o variables con minúscula
- Métodos que realizan una única función
- Funciones que no reciben más de 2 parámetros
- Etc..

Por otro lado con lo que lleva a la aplicación de TDD, comenzamos aplicando de forma correcta respetando las 3 fases que tiene dicha metodología. Siempre creando la prueba primero, y luego de fallar, comenzar a realizar el refactor y por último el código.
Si bien más sobre el final del proyecto tuvimos problemas con git y el tiempo nos empezó a jugar en contra y nos obligó a no aplicar TDD un 100%.

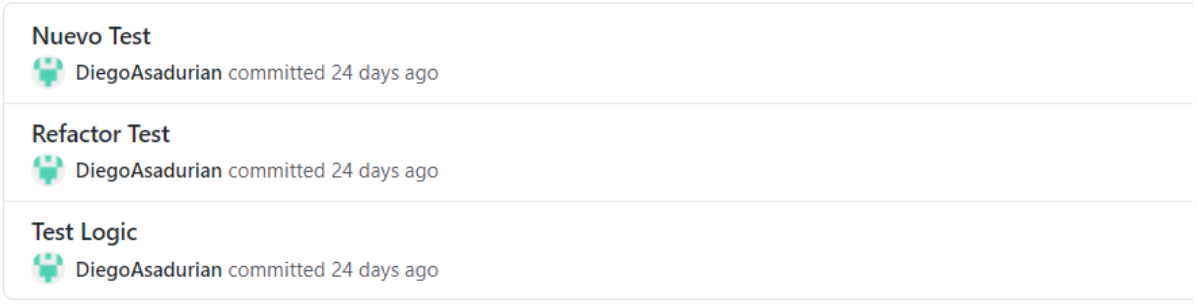
A continuación se deja constancia de cómo aplicamos técnicas de clean code, en este caso mostramos el metodo Create de administrador en la clase AdministratorLogic del paquete MSP.BetterCalm.BusinessLogic



Como se puede ver en dicho método el mismo comienza con Mayúscula y es una función que recibe un parámetro y realiza una única función.
Como se puede ver el mismo método llama a dos funciones auxiliares las cuales devuelven si el administrador no existe en el sistema y si tanto el mail como el password son correctos.
Dicho método fue refactoreado para poder dejarlo de la manera en la que está, ya que antes de refactorear las validaciones del mail como password las realizabamos dentro del mismo método, lo cual no cumpliamos con clean code, fue por eso que decidimos separarlo en métodos auxiliares, para cumplir con las normas de clean code.

Evidencia de TDD

Como explicamos al principio de este documento al principio comenzamos utilizando TDD tratando de crear la prueba, luego refactorear y luego crear nuevos casos.
Creemos que es de suma respetar el proceso de la construcción de la prueba utilizando TDD, ya que el mismo permite reducir Bugs en la codificación y permite en un futuro poder ingresar nuevas modificaciones de una forma más sencilla.



Diseño de casos de prueba

Pruebas unitarias y cobertura de las mismas

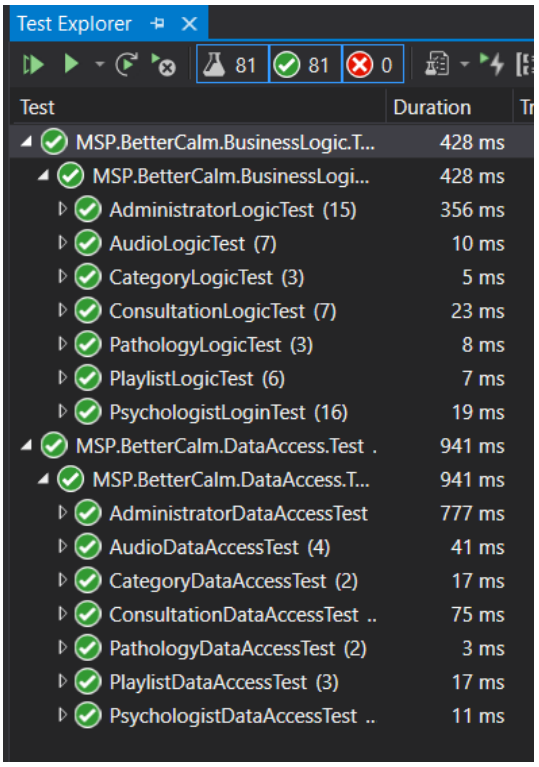
Para llevar a cabo los test finales de las pruebas se utilizo Visual Studio Enterprise 2019 ya que brinda la posibilidad de analizar toda la cobertura de las pruebas.

Las pruebas fueron escritas respetando el principio de las mismas FIRST que dicta por sus siglas. Las mismas estan compuestas por 3 fases bien definidas como lo son el Arrange, Act y el Assert. Para que vean un ejemplo de como lo aplicamos le dejamos una imagen del mismo.

Como se puede ver en la imagen se aplica correctamente los métodos recientemente mencionados.

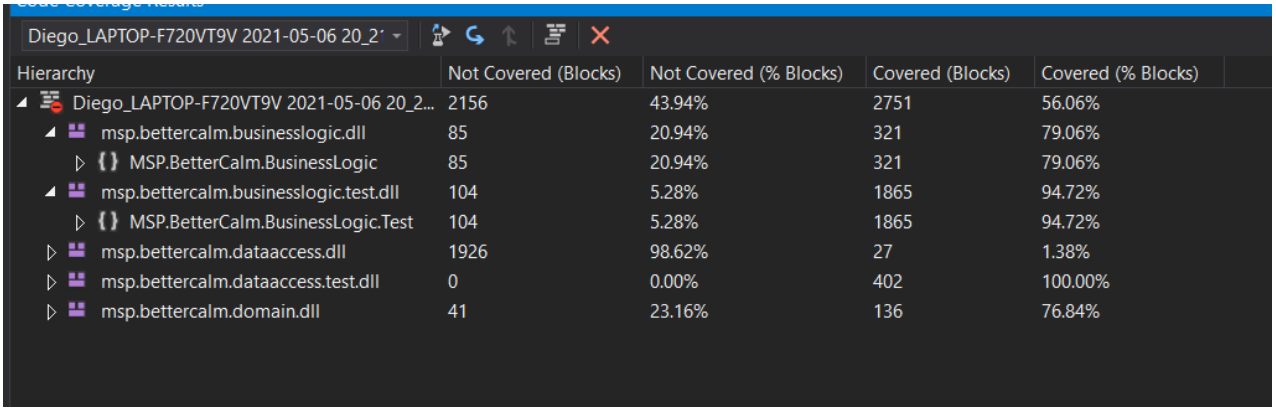
Por otro lado para llevar a cabo la cobertura de pruebas y ver la cantidad de código que tenemos probado utilizamos la herramienta de visual Code Coverage.

A continuación se deja captura de todos los test ejecutados, para que se pueda visualizar la velocidad de ejecución de cada test, como también observar que los mismo fueron satisfactorios.



Como se puede ver en la imagen la velocidad de los test son muy ágiles.

A continuación se brinda la cobertura de código que poseemos en el proyecto



Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Diego_LAPTOP-F720VT9V 2021-05-06 20_2...	2156	43.94%	2751	56.06%
msp.bettercalm.businesslogic.dll	85	20.94%	321	79.06%
MSP.BetterCalm.BusinessLogic	85	20.94%	321	79.06%
msp.bettercalm.businesslogic.test.dll	104	5.28%	1865	94.72%
MSP.BetterCalm.BusinessLogic.Test	104	5.28%	1865	94.72%
msp.bettercalm.dataaccess.dll	1926	98.62%	27	1.38%
msp.bettercalm.dataaccess.test.dll	0	0.00%	402	100.00%
msp.bettercalm.domain.dll	41	23.16%	136	76.84%

Como podemos observar lo que es la lógica si bien tenemos más del 50% probado de negocio no logramos completar el 100%, ya que a último momento del proyecto logramos solucionar determinados errores, los cuales no nos dio el tiempo para poder probarlo en su totalidad.