Universidad ORT Uruguay
Facultad de Ingeniería

Obligatorio 1
Diseño de Aplicaciones 2

Evidencia de la aplicación de TDD y Clean Code

Repositorio: https://github.com/IngSoft-DA2-2023-2/242493-260956-281651
Video detallando pruebas: https://youtu.be/Mo39Tbi6L1s

Romina Arour - 242493
Daiana Aysa - 281651
Ana Gutman - 260956

Tutores:

2023

INDICE

Evidencia de Clean Code y de Aplicación de TDD

Clean Code

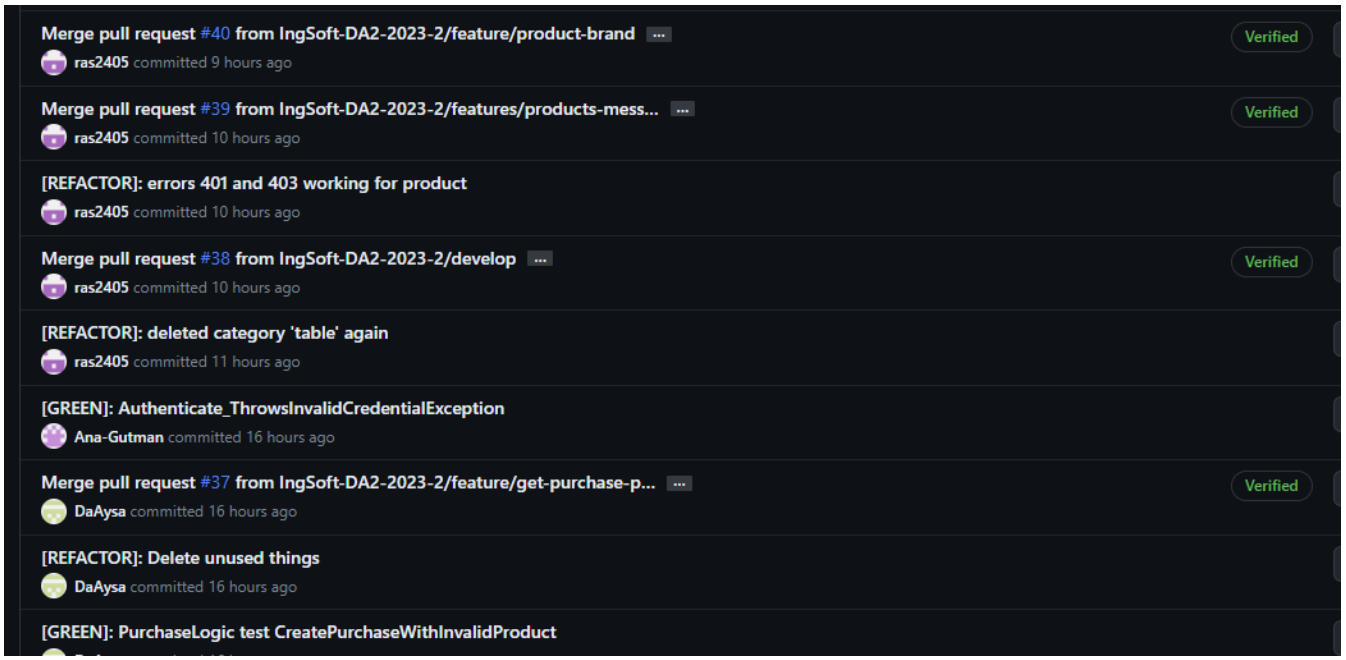
Nos basamos en los principios de clean code para garantizar que nuestro código mantuviera una legibilidad, comprensión y mantenibilidad sencillas. Inicialmente, nos aseguramos de seguir una estructura clara y consistente en el código, optando por nombres de variables y funciones que fueran descriptivos, facilitando así la comprensión tanto para quien programaba, como para el resto de las integrantes del equipo.

Para cumplir con lo indicado como buenas prácticas de clean code, nos aseguramos que todos los métodos tengan 3 o menos parámetros. También tuvimos en cuenta de no tener más de 2 enters entre las líneas del código y procuramos evitar el uso de comentarios innecesarios o complicados. Otra práctica implementada fue una gestión apropiada de excepciones, proporcionando mensajes de error claros y útiles para una comprensión efectiva.

Adicionalmente, incorporamos la modularidad del código para asegurarnos de que cada componente de nuestra solución fuera independiente y pudiera someterse a pruebas de forma individual. Esto nos habilita para identificar errores de manera más rápida y efectuar cambios con mayor facilidad en el futuro.

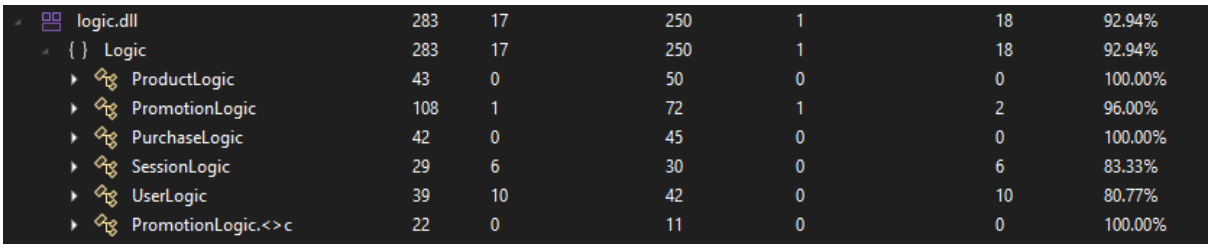
Evidencia del uso de TDD en las funcionalidades

Se puede visualizar en GitHub el uso de TDD a lo largo de todo el proyecto, por medio de commits los cuales tienen especificadas las fases: “[GREEN]” o “[REFACTOR]”, los cuales se encuentran dentro del historial de nuestro repositorio.
Se puede ver un ejemplo a continuación del uso de esta metodología:

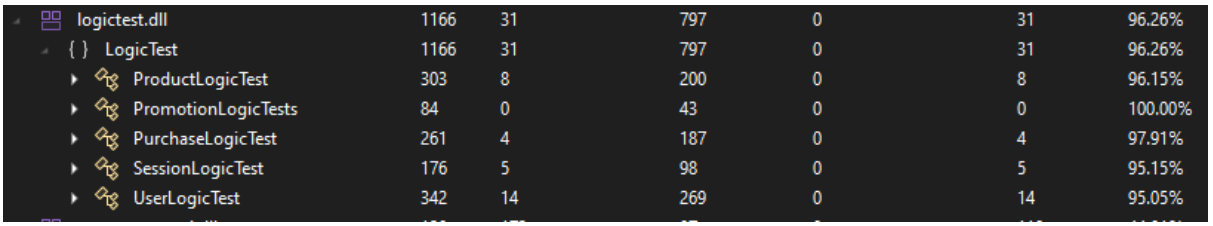


Informe de cobertura de las pruebas unitarias

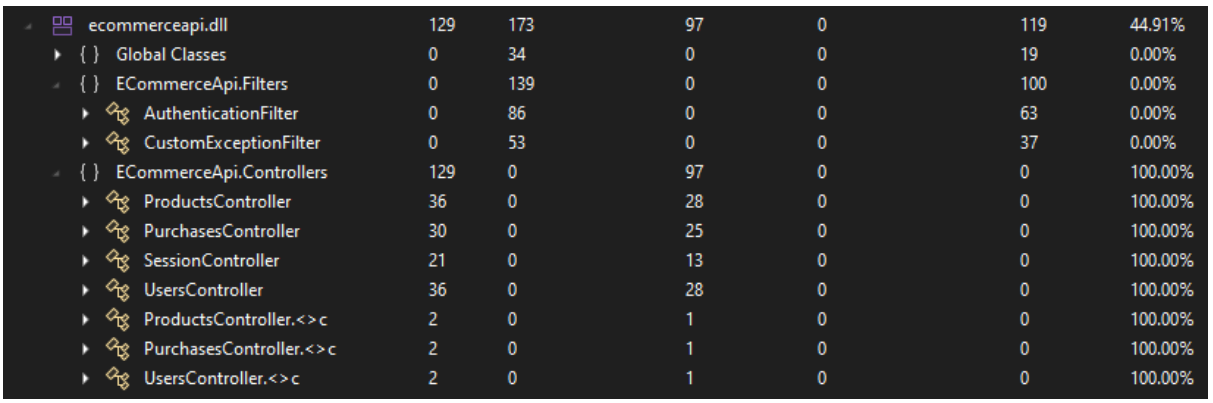
En la siguientes imágenes podemos visualizar la cobertura. Se puede apreciar como la cobertura es casi total en los módulos abordados con TDD.



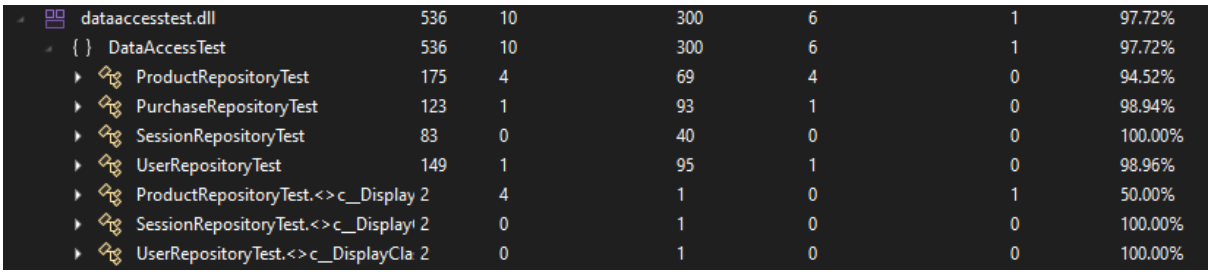
logic.dll	283	17	250	1	18	92.94%
{ } Logic	283	17	250	1	18	92.94%
ProductLogic	43	0	50	0	0	100.00%
PromotionLogic	108	1	72	1	2	96.00%
PurchaseLogic	42	0	45	0	0	100.00%
SessionLogic	29	6	30	0	6	83.33%
UserLogic	39	10	42	0	10	80.77%
PromotionLogic.<>c	22	0	11	0	0	100.00%



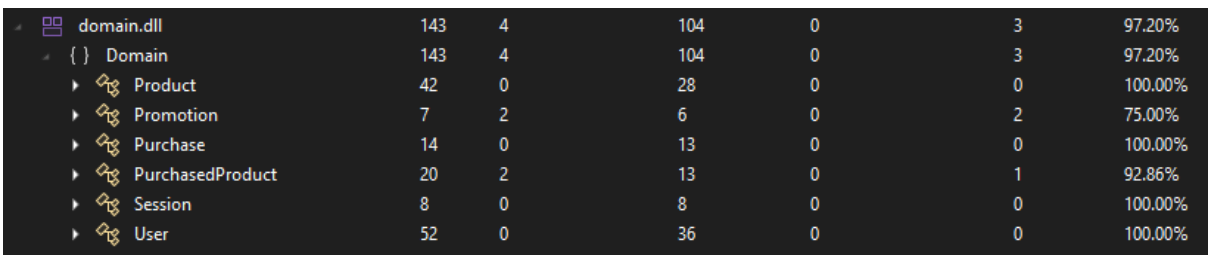
logictest.dll	1166	31	797	0	31	96.26%
{ } LogicTest	1166	31	797	0	31	96.26%
ProductLogicTest	303	8	200	0	8	96.15%
PromotionLogicTests	84	0	43	0	0	100.00%
PurchaseLogicTest	261	4	187	0	4	97.91%
SessionLogicTest	176	5	98	0	5	95.15%
UserLogicTest	342	14	269	0	14	95.05%



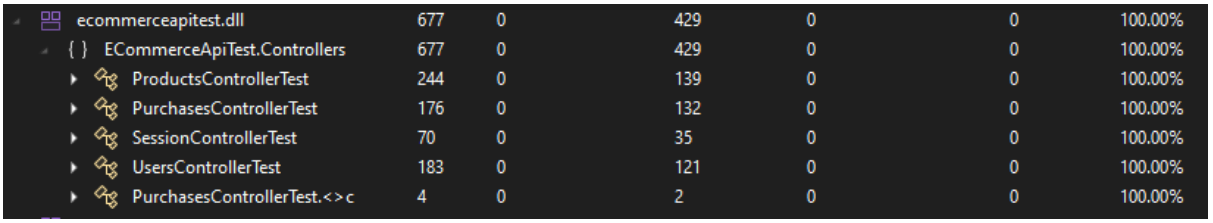
ecommerceapi.dll	129	173	97	0	119	44.91%
{ } Global Classes	0	34	0	0	19	0.00%
{ } ECommerceApi.Filters	0	139	0	0	100	0.00%
AuthenticationFilter	0	86	0	0	63	0.00%
CustomExceptionFilter	0	53	0	0	37	0.00%
{ } ECommerceApi.Controllers	129	0	97	0	0	100.00%
ProductsController	36	0	28	0	0	100.00%
PurchasesController	30	0	25	0	0	100.00%
SessionController	21	0	13	0	0	100.00%
UsersController	36	0	28	0	0	100.00%
ProductsController.<>c	2	0	1	0	0	100.00%
PurchasesController.<>c	2	0	1	0	0	100.00%
UsersController.<>c	2	0	1	0	0	100.00%



dataaccesstest.dll	536	10	300	6	1	97.72%
{ } DataAccessTest	536	10	300	6	1	97.72%
ProductRepositoryTest	175	4	69	4	0	94.52%
PurchaseRepositoryTest	123	1	93	1	0	98.94%
SessionRepositoryTest	83	0	40	0	0	100.00%
UserRepositoryTest	149	1	95	1	0	98.96%
ProductRepositoryTest.<>c__Display	2	4	1	0	1	50.00%
SessionRepositoryTest.<>c__Display	2	0	1	0	0	100.00%
UserRepositoryTest.<>c__DisplayCla	2	0	1	0	0	100.00%

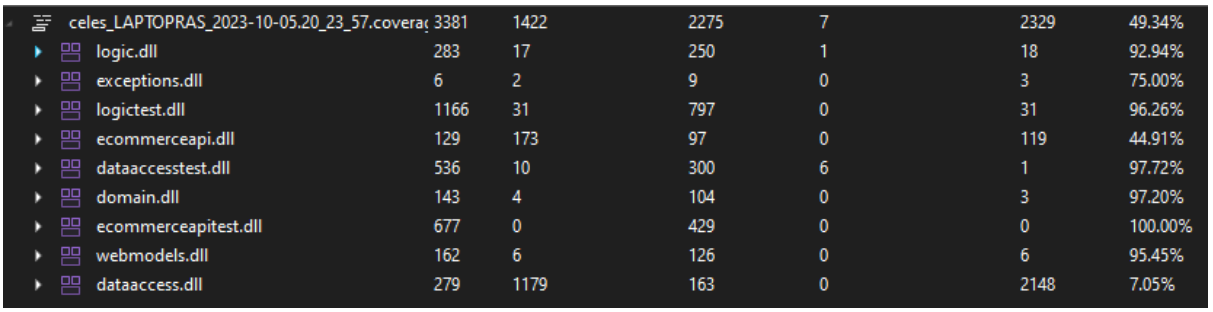


domain.dll	143	4	104	0	3	97.20%
{ } Domain	143	4	104	0	3	97.20%
Product	42	0	28	0	0	100.00%
Promotion	7	2	6	0	2	75.00%
Purchase	14	0	13	0	0	100.00%
PurchasedProduct	20	2	13	0	1	92.86%
Session	8	0	8	0	0	100.00%
User	52	0	36	0	0	100.00%



ecommerceapitest.dll	677	0	429	0	0	100.00%
{ } ECommerceApiTest.Controllers	677	0	429	0	0	100.00%
ProductsControllerTest	244	0	139	0	0	100.00%
PurchasesControllerTest	176	0	132	0	0	100.00%
SessionControllerTest	70	0	35	0	0	100.00%
UsersControllerTest	183	0	121	0	0	100.00%
PurchasesControllerTest.<>c	4	0	2	0	0	100.00%

webmodels.dll	162	6	126	0	6	95.45%
{ } WebModels.Models.Out	70	6	55	0	6	90.16%
{ } WebModels.Models.In	92	0	71	0	0	100.00%



celes_LAPTOPRAS_2023-10-05.20_23_57.covera	3381	1422	2275	7	2329	49.34%
logic.dll	283	17	250	1	18	92.94%
exceptions.dll	6	2	9	0	3	75.00%
logictest.dll	1166	31	797	0	31	96.26%
ecommerceapi.dll	129	173	97	0	119	44.91%
dataaccesstest.dll	536	10	300	6	1	97.72%
domain.dll	143	4	104	0	3	97.20%
ecommerceapitest.dll	677	0	429	0	0	100.00%
webmodels.dll	162	6	126	0	6	95.45%
dataaccess.dll	279	1179	163	0	2148	7.05%

El porcentaje de cobertura total se ve fuertemente afectado por los módulos como ECommerceApi.Filters que fueron excluidos del Test Driven Development por exceder el alcance del curso. Sin embargo, al ver el desglose de todos los porcentajes podemos apreciar que esta metodología tiene la clara consecuencia positiva de generar exhaustividad a la hora de probar el código. De esta manera se genera la confianza de que cada funcionalidad implementada, y dentro de ella cada línea de código, se pone a prueba antes de entrar en producción.