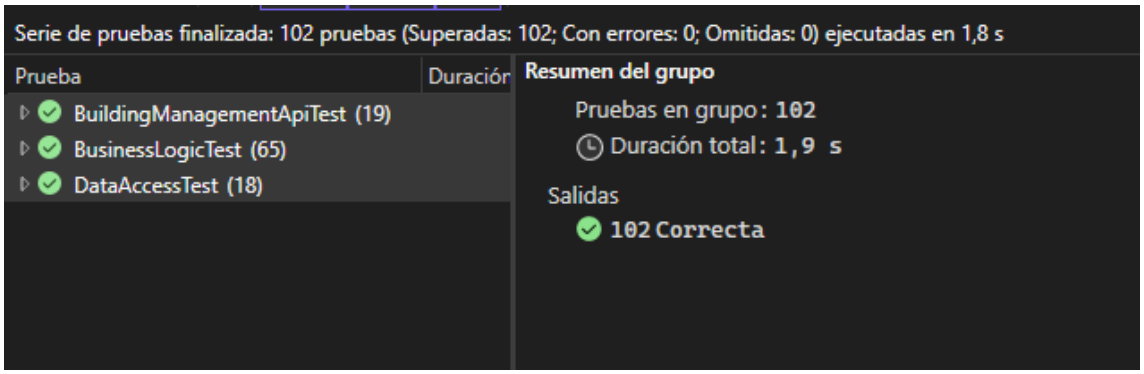
Evidencia de Clean Code y de la aplicación de TDD

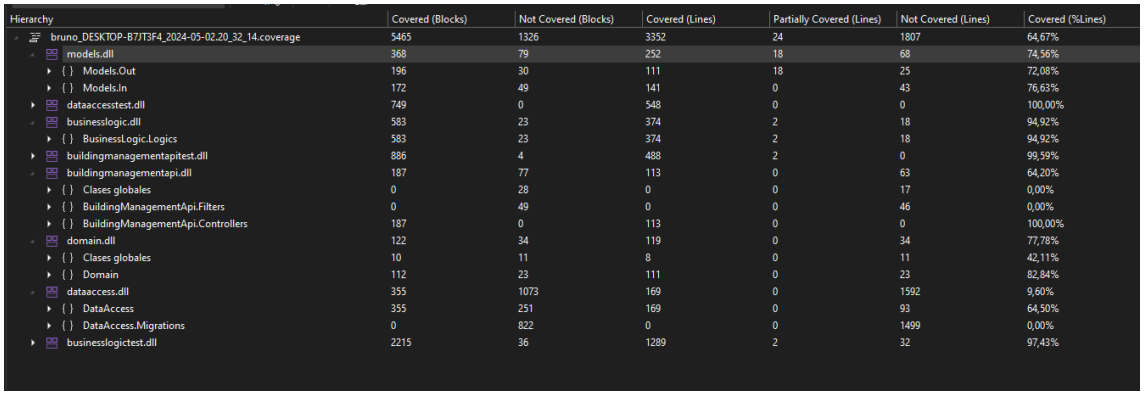
Estrategia de las pruebas

El desarrollo de la aplicación se llevó a cabo siguiendo el enfoque de Test-Driven Development (TDD). Si se ven los primeros commits, se pueden ver GREENs y REDs respectivamente. A su vez, mas tarde se dejo de poner los commits de esta forma y decidimos hacerlo de la forma de TDD avanzado, programar las pruebas y no hacer el commit hasta asegurarnos que corran.

Cobertura de código y resultado de las pruebas.

Las pruebas corrían bien y tenían esta cobertura de código,





Hierarchy	Covered (Blocks)	Not Covered (Blocks)	Covered (Lines)	Partially Covered (Lines)	Not Covered (Lines)	Covered (%Lines)
bruno_DESKTOP-B7JT3F4_2024-05-02.20_32_14.coverage	5465	1326	3352	24	1807	64,67%
models.dll	368	79	252	18	68	74,56%
{ } Models.Out	196	30	111	18	25	72,08%
{ } Models.In	172	49	141	0	43	76,63%
dataaccesstest.dll	749	0	548	0	0	100,00%
businesslogic.dll	583	23	374	2	18	94,92%
{ } BusinessLogic.Logics	583	23	374	2	18	94,92%
buildingmanagementapitest.dll	886	4	488	2	0	99,59%
buildingmanagementapi.dll	187	77	113	0	63	64,20%
{ } Clases globales	0	28	0	0	17	0,00%
{ } BuildingManagementApi.Filters	0	49	0	0	46	0,00%
{ } BuildingManagementApi.Controllers	187	0	113	0	0	100,00%
domain.dll	122	34	119	0	34	77,78%
{ } Clases globales	10	11	8	0	11	42,11%
{ } Domain	112	23	111	0	23	82,84%
dataaccess.dll	355	1073	169	0	1592	9,60%
{ } DataAccess	355	251	169	0	93	64,50%
{ } DataAccess.Migrations	0	822	0	0	1499	0,00%
businesslogictest.dll	2215	36	1289	2	32	97,43%

pero debido a correcciones de último momento, se rompieron y no andan. Nos comprometemos a arreglarlos para la siguiente entrega. Somos conscientes de que la cobertura de código no era la ideal. Más usando TDD, pero nos comprometemos a mejorarla para la siguiente entrega

Uso de mocks y frameworks de testing

Uso de Mocks
Durante el desarrollo del proyecto, se hizo uso de mocking para probar tanto los controladores como la lógica de negocio de manera aislada. El uso de mocks permite simular el comportamiento de las dependencias que se dan por el hecho de tener interfaces. Esto es esencial para garantizar que las pruebas sean rápidas, confiables y no dependan de factores externos.

Uso de EF Core InMemory
Para probar la capa de acceso a datos, se utilizó la base de datos InMemory de Entity Framework Core. Esto proporciona una forma eficaz de ejecutar pruebas que involucran operaciones de base de datos sin la necesidad de una base de datos real.