

Universidad ORT Uruguay

Facultad de Ingeniería

Diseño de aplicaciones 2

Descripción del diseño

Obligatorio 1

Emiliano Barboza (147067)
Mauricio Dalgalarondo (189280)

Docentes:
Ignacio Valle
Daniel Acevedo

Entregado como requisito de la materia Diseño de
aplicaciones 2
<https://github.com/ORT-DA2/barboza-dalgalarondo>

15 de octubre de 2020

Declaraciones de autoría

Nosotros, Emiliano Barboza y Mauricio Dalgarrondo, 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 la materia
- 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

Este documento del obligatorio 1 plantea poner en práctica los conocimientos vistos hasta el momento en el curso del punto de vista la aplicación de TDD y Clean Code visto en diseño de aplicaciones 1:

1. TDD
2. Clean code
3. Code coverage

Palabras claves

- TDD, Test Development Driven, consta de una técnica para el desarrollo de software mediante pruebas unitarias.
- Code coverage, herramienta de reporte de cobertura de los test unitarios.
- Outside-In, se trata de una metodología aplicada a TDD. Consiste en que el desarrollador comience creando la funcionalidad final del sistema y luego mediante refactorizaciones ir creando pequeños componentes que dan soporte. El camino resulta más como ir explorando las necesidades. Esto resulta bueno, cuando se conoce la idea general, pero no los detalles de la implementación.

Índice general

1. Descripción del Proyecto	5
1.1. Introducción y Objetivos	5
1.2. Alcance de la Aplicación	5
2. Descripción de la estrategia de TDD	7
2.1. Evidencia cantidad de test unitarios creados en el proceso	8
2.2. Nomenclatura para nombrar test unitarios	8
2.3. Pruebas de aplicación de TDD para los casos marcados con *	9
2.3.1. (*)Buscar hospedajes para un cierto punto turístico con los parámetros especificados.	9
2.3.2. (*) Realizar una reserva de un hospedaje.	10
2.3.3. (*) Dar de alta un nuevo hospedaje o borrar uno existente, para un punto turístico existente.	11
2.3.4. (*) Modificar la capacidad actual de un hospedaje.	12
2.3.5. (*) Cambiar el estado de una reserva, indicando una descripción.	12
3. Clean Code	13
3.1. Nomenclatura	13
3.2. Largo de métodos	13
3.3. Parámetros	13
3.4. Nombres	13
3.5. Excepciones	13
3.6. Lugares donde se infringieron criterios de Clean code	13
3.7. Paquetes de Test	14
4. Code coverage	15
4.1. Paquetes y clases excluidas	15

1. Descripción del Proyecto

1.1. Introducción y Objetivos

El objetivo de este proyecto es crear una api rest para dar soporte a la marca Uruguay Natural.

Las tecnologías involucradas son:

- Web Services (REST API) - Se expondrá las apis en un IIS.
- Microsoft Visual Studio Code (lenguaje C)
- Microsoft SQL Server Express 2017
- Postman
- NET Core SDK 3.1 / ASP.NET Core 3
- Entity Framework Core 3.1.3

1.2. Alcance de la Aplicación

El nuevo sistema, deberá dar acceso a 3 tipos de usuarios posibles en la plataforma (turistas, administradores y super administradores). Para los cuales se listan los requerimientos funcionales para los mismos.

Requerimientos funcionales:

- RF1 - Búsqueda de puntos turísticos por región y por categoría:.
- RF2 - Elegir un punto turístico y realizar una búsqueda de hospedajes.
- RF3 - Dado un hospedaje, realizar una reserva.

Requerimientos no funcionales

- RNF1 - Independencia de librerías.

Se debe diseñar la solución que al modificar el código fuente minimice el impacto del cambio en los componentes físicos de la solución.

Cada paquete lógico debe ser implementado en un assembly independiente.

- RNF2 - Acceso a las funcionalidades mediante HTTP

Acceso mediante web service.

- RNF3 - Persistencia en base de datos.

El diseño debe contemplar el modelado de una solución de persistencia adecuada para el problema utilizando Entity Framework (Code First).

Se espera que como parte de la entrega se incluya dos respaldos de la base de datos: uno vacío y otro con datos de prueba.

Se debe entregar el archivo .bak y también el script .sql para ambas bases de datos.

- RNF4 - Mantenibilidad

Estar en un repositorio Git.

Haber sido escrito utilizando TDD

Se debe utilizar el framework Moq para realizar los Mocking.

- RNF5 - Control de versiones

GitFlow

2. Descripción de la estrategia de TDD

Elegimos la estrategia de Outside-In, debido a que nuestro conocimiento mayor consistía en saber qué apis eran necesarias, por lo tanto decidimos empezar desde afuera.

Siguiendo con las reglas de TDD (toda implementación requiere ser una necesidad generada por un test), comenzamos creando nuestra `WebApi.Test`. En la misma fuimos primeramente creando el controller junto al test que fuimos necesitando. Luego que ese test estaba escrito y fallaba (red), podíamos pasar a la siguiente etapa que era la implementación (green).

Finalmente, luego de varios ciclos aplicando esta técnica fuimos viendo las oportunidades de refactorio (refactor), dado a que fuimos viendo que en varias ocasiones el código se repetía.

Por ende este fue el flujo de trabajo aplicado en todo el ciclo de desarrollo de la api.

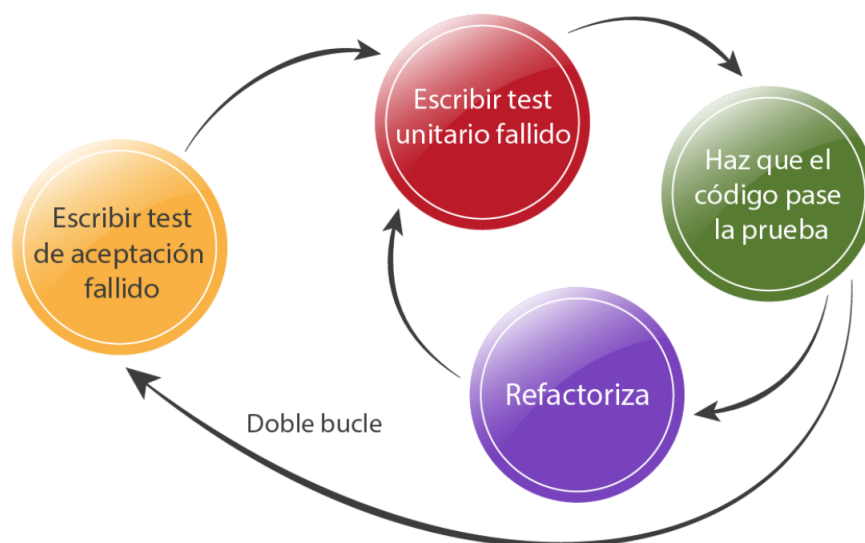


Figura 2.1: Ciclo TDD

2.1. Evidencia cantidad de test unitarios creados en el proceso

Se crearon un total de 177 test unitarios, los cuales se dividieron de la siguiente forma por proyecto:

- AuthorizationDataAccess.Test - 14
- AuthorizationEngine.Test - 5
- Helper.Test - 11
- NaturalUruguayDataAccess.Test - 60
- NaturalUruguayEngine.Test - 48
- WebApi.Test - 39

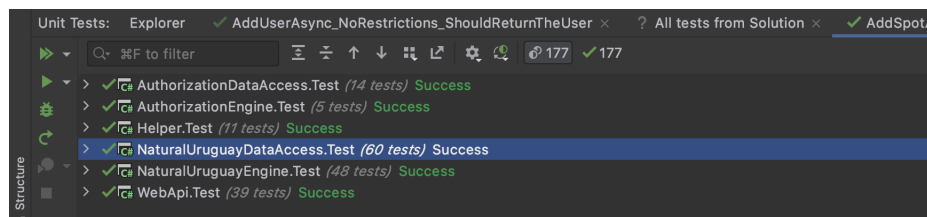


Figura 2.2: Evidencia TDD.

2.2. Nomenclatura para nombrar test unitarios

Seguimos las recomendaciones del artículo, donde especifica como buena práctica para el nombre de un test:

- NombreMetodo_EstadoActual_EstadoFinalEsperado

2.3. Pruebas de aplicación de TDD para los casos marcados con *

2.3.1. (*)Buscar hospedajes para un cierto punto turístico con los parámetros especificados.

The screenshot displays a GitHub Pull Request (PR) interface. At the top, a navigation bar shows 'Conversation 0', 'Commits 17', 'Checks 0', and 'Files changed 30'. A status bar on the right indicates '+1,055 -118' with a progress indicator. The main content area shows a list of commits by 'EmilianoBarbozaOjeda' added 16 days ago. The commits include tests, refactors, and integrations. A comment by 'Emiliano-Barboza' is visible at the top. The right sidebar contains sections for 'Reviewers' (MauriDalga), 'Assignees' (No one), 'Labels' (None yet), 'Projects' (None yet), 'Milestone' (No milestone), 'Linked issues' (Successfully merging this pull request may close these issues), 'Notifications' (Unsubscribe), and '1 participant'.

Emiliano-Barboza commented 14 days ago

No description provided.

EmilianoBarbozaOjeda added 17 commits 16 days ago

- red - GetLodgments_Success and GetLodgments_IfThrowsException_ShouldR...
- green - GetLodgments_Success and GetLodgments_IfThrowsException_Shoul...
- refactor - removed dead code
- refactor - moved spot calls inside to region service
- refactor - removed dependency service in controller and moved into th...
- refactor - added test GetRegionByIdAsync_CallingTheRepository_ShouldR...
- refactor - added missing tests
- green - GetLodgmentsAsync_ThereAreLodgments_ShouldReturnTheLodgments ...
- red - GetLodgmentsAsync_ThereAreLodgments_ShouldReturnTheLodgments an...
- green - GetLodgmentsAsync_ThereAreLodgments_ShouldReturnTheLodgments ...
- refactor - integrate repository
- refactor - fixed circular dependency injection
- green - fixed tests after refactor
- refactor - improved code coverage
- red - GetLodgmentsAsync_ThereAreLodgmentsNotCalculateTotalStay_Should...
- green - GetLodgmentsAsync_ThereAreLodgmentsNotCalculateTotalStay_Shou...
- refactor - integrate calculator

Emiliano-Barboza requested a review from MauriDalga 14 days ago

Emiliano-Barboza merged commit 959e0cd into develop 14 days ago

Revert

Reviewers

MauriDalga

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked issues

Successfully merging this pull request may close these issues.

None yet

Notifications

Unsubscribe

You're receiving notifications because you're watching this repository.

1 participant

Lock conversation

Figura 2.3: Buscar hospedajes. Pull Request

2.3.2. (*) Realizar una reserva de un hospedaje.

The screenshot displays a GitHub Pull Request (PR) interface. At the top, a navigation bar shows 'Conversation' (0), 'Commits' (24), 'Checks' (0), and 'Files changed' (40). A status bar on the right indicates '+2,128 -182' changes. The main content area is divided into two columns. The left column, titled 'Commits', lists 24 commits by 'MauriDalga' added 24 days ago. The commits include tests, merges, fixes, and refactors. The right column contains a 'Conversation' section with a comment from 'Emiliano-Barboza' stating 'Redoing merge to the correct branch, since this was merged to master'. Below the comment are sections for 'Reviewers' (MauriDalga), 'Assignees' (None), 'Labels' (None), 'Projects' (None), 'Milestone' (None), 'Linked issues' (None), 'Notifications' (Unsubscribe), and '2 participants' (Emiliano-Barboza, MauriDalga). At the bottom, a 'Revert' button is visible.

Emiliano-Barboza commented 15 hours ago

Redoing merge to the correct branch, since this was merged to master

MauriDalga added 24 commits 24 days ago

- green - AddBookingAtLodgment_Success, AddBookingAtLodgment_Fail in L.o...
- green - AddBookingAtLodgmentAsync_LodgmentExist_ShouldReturnBookingCo...
- green - AddBookingAtLodgmentAsync_LodgmentNotExist_ShouldThrowExcept...
- Merge branch 'develop' into feature/List-regions
- fix - solved circular reference problems into booking tests
- green - GetLodgmentByIdAsync tests added in LodgmentsService.cs
- green - AddBookingAsync in BookingsRepository
- Merge branch 'develop' into feature/create-bookings
- green - add 'created' booking status at AddBookingAtLodgmentAsync
- green - add date manage to price calculator
- green - LodgmentOptionModel Validation in LodgmentOptionModelTest
- refactor - LodgmentOptionModel instancing into tests
- refactor - updated GetLodgments postman
- refactor - updated Booking data access Migration
- refactor - updated AddBooking postman
- Merge branch 'develop' into feature/create-bookings
- green - GetBookingStatusAsync_BookingNotExists_ShouldReturnException - ...
- refactor - updated booking db schema, fixed typo in property
- green - AddBookingAtLodgmentAsync_LodgmentIsInactive_ShouldReturnExce...
- refactor - error messages translated into English
- refactor - deleted async keyword on synchronous tests
- refactor - updated Directory.Build.props
- refactor - Authorization token required error normalize
- refactor - updated postman requests for get booking status

Emiliano-Barboza requested a review from MauriDalga 15 hours ago

Emiliano-Barboza merged commit 6d49b5e into develop 15 hours ago

Revert

Reviewers

MauriDalga

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked issues

Successfully merging this pull request may close these issues.

None yet

Notifications

Customize

Unsubscribe

You're receiving notifications because you're watching this repository.

2 participants

Emiliano-Barboza, MauriDalga

Lock conversation

Figura 2.4: Realizar una reserva. Pull Request

2.3.3. (*) Dar de alta un nuevo hospedaje o borrar uno existente, para un punto turístico existente.

feature/add-re...	
Commits on Sep 27, 2020	
refactor - moved part of the logic to spots controller since we need ...	7c7f519a
refactor - data access integration	3fac22a
green - DeleteLodgmentAsync_LodgmentNotExists_ShouldThrowException I...	85c2282
green - DeleteLodgmentAsync_LodgmentNotExists_ShouldReturnTheLodgment...	5d88f4c
red - DeleteLodgmentAsync_LodgmentNotExists_ShouldReturnTheLodgment I...	9495389
green - DeleteLodgment_Success and DeleteLodgment_IfThrowsException_S...	5b7b496
red - DeleteLodgment_Success and DeleteLodgment_IfThrowsException_Sho...	edd77fc
refactor - set user role dynamically	899325e
refactor - seeding user role and user	5ac71e1
refactor - moved lodgment services to new class	325994a
refactor - include Spot in the lodgment object	e86a1a8
green - AddLodgmentAsync_TouristSpotNotExists_ShouldThrowException I...	67b2896
red - AddLodgmentAsync_TouristSpotNotExists_ShouldThrowException in ...	b6742d3
Commits on Sep 26, 2020	
refactor - integrate data base and some renamings	e3187c3
green - AddLodgmentAsync_LodgmentExists_ShouldThrowException in Spot...	b93908a
green - AddLodgmentAsync_LodgmentNotExists_ShouldReturnTheLodgment in...	2644c48
red - AddLodgmentAsync_LodgmentNotExists_ShouldReturnTheLodgment in S...	58ac218
refactor - applied nomenclature in variables	9666366
green - AddLodgment_Success in SpotsControllerTest	40862b8
red - AddLodgment_Success and AddLodgment_IfThrowsException_ShouldRet...	6820563
Commits on Sep 25, 2020	
refactor - created class domain diagram	56f8c78
Merge pull request #2 from ORT-DA2/feature/create-crud-actions-for-ad...	Verified 1233166
refactor - added paging control in postman	e1baa11
refactor - update postman json	35caaf2
refactor - repository integration	849817f
green - GetUsers_NoRestrictions_ShouldReturnTheUsers in UsersServiceTest	17b8cfa
green - GetUsers_NoRestrictions_ShouldReturnTheUsers in UsersService	438cd1d
red - GetUsers_NoRestrictions_ShouldReturnTheUsers in UsersServiceTest	8c63e3d
green - GetUsers_IfThrowsException_ShouldReturnBadRequest in UsersCon...	68b693e
green - GetUsers_Success in UsersControllerTest	579b086
red - GetUsers_Success in UsersControllerTest	34324d9
green - UpdateUser_UserExistsAndTriesToChangeEmailThatAlreadyExists_S...	4a12ad1
refactor - integrate update	7e4af5b
green - UpdateUser_UserExists_ShouldReturnTheUpdatedUser and UpdateUs...	b22fbce
red - UpdateUser_UserExists_ShouldReturnTheUpdatedUser and UpdateUser...	29625e1
Newer Older	

Figura 2.5: Dar de alta un nuevo hospedaje o borrar uno existente. Pull Request

2.3.4. (*) Modificar la capacidad actual de un hospedaje.

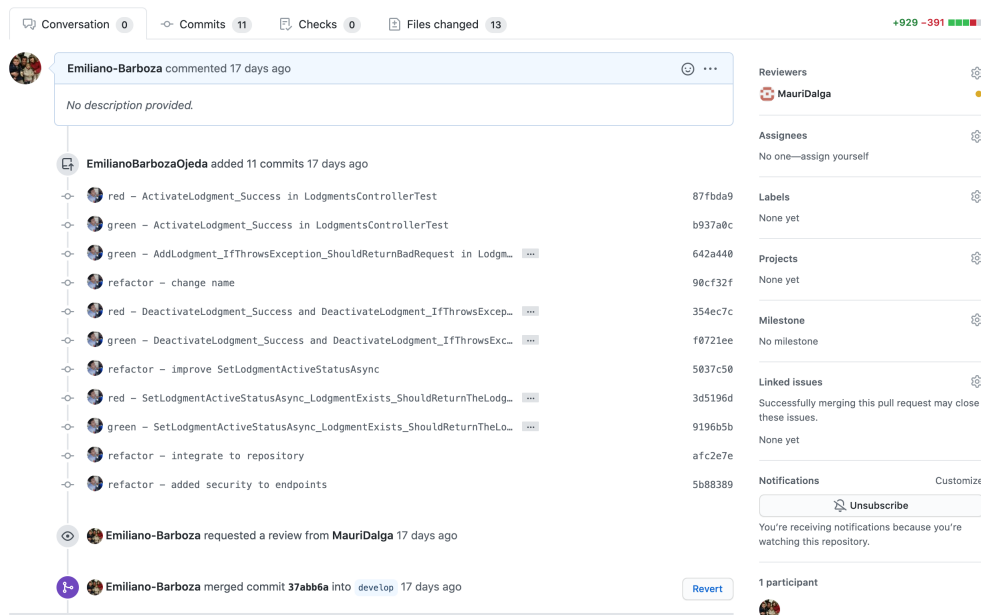


Figura 2.6: Modificar la capacidad actual de un hospedaje. Pull Request

2.3.5. (*) Cambiar el estado de una reserva, indicando una descripción.

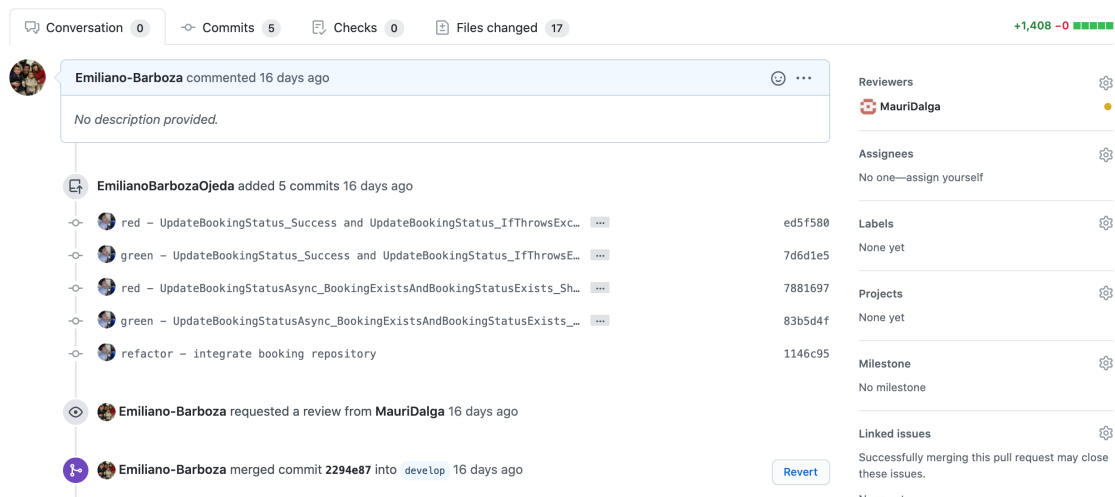


Figura 2.7: Cambiar el estado de una reserva. Pull Request

3. Clean Code

3.1. Nomenclatura

Se siguió el estandar de C sharp declarado en el siguiente repositorio con el fin de seguir buenas prácticas.

3.2. Largo de métodos

Los métodos no superan las 30 líneas en los métodos más largos, respetando la legibilidad en la laectura.

3.3. Parámetros

Declaramos un máximo de 3 parámetros en los métodos.

3.4. Nombres

Los nombres son nemotécnicos y hacen referencia a los objetos que conyevan.

3.5. Excepciones

Se manejan las expcepciones tiradas por base de datos de forma que las mismas no lleguen al cliente de forma cruda

3.6. Lugares donde se infringieron criterios de Clean code

Lugares como el repositorio, Setup de web api y los paquetes de test, fueron excluidos para este criterio debido a que en algunos casos es muy conveniente la nomenclatura Lambda, la cual va en contra de Clean Code.

3.7. Paquetes de Test

Como estrategia de test unitarios se decidió que los proyectos de test se nombren de forma tal que sea el "NombreDelProyecto.Test".

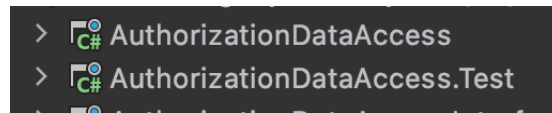


Figura 3.1: Convención nombres de paquetes de test.

4. Code coverage

Declaramos haber logrado un 95 % de code coverage. Identificamos que faltaron casos de prueba para lograr un mejor desempeño en el paquete de Domain el cual quedó con un 70 %, siendo así el menos cubierto.

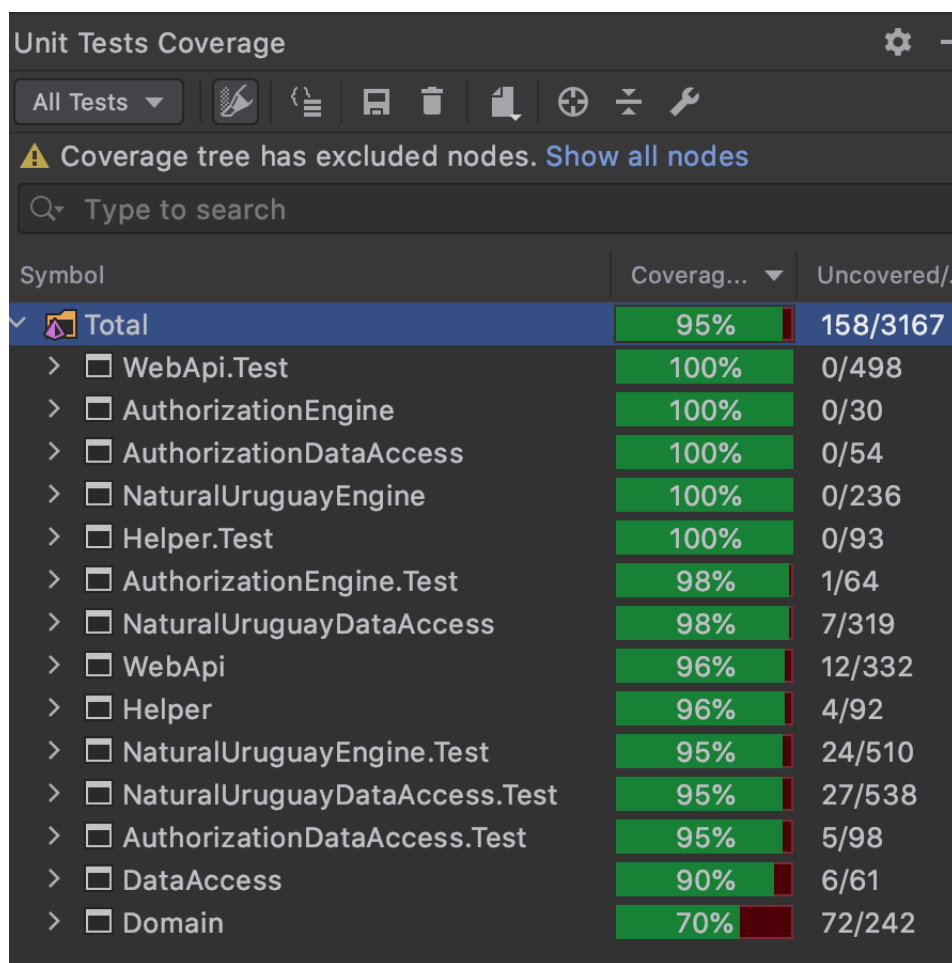


Figura 4.1: Convención nombres de paquetes de test.

4.1. Paquetes y clases excluidas

- Migrations dentro del DataAccess, estas con las migraciones que se corren con el comando dotnet.

- Migrations, este es el proyecto del cual arrancamos las migraciones.
- Factory, aquí únicamente ocurren la inyección de dependencias.
- Clase Program y Setup del proyecto WebApi, estas no tienen sentido testearlas ya que es el arranque de la aplicación y configuraciones de la misma.
- WrappedDbException, implementa la clase Exception y nada más, por lo cual no tiene sentido testearla ya que sería igual que testear Exception.
- NaturalUruguayContext, es el acceso a la base de datos.

Bibliografía

- [1] Universidad ORT Uruguay. (2013) Documento 302 - Facultad de Ingeniería. [Online]. Available: <http://www.ort.edu.uy/fi/pdf/documento302facultaddeingenieria.pdf>