



Universidad ORT Uruguay

Facultad de Ingeniería

Diseño de Aplicaciones 2

Obligatorio 1

BlogsApp

Fernando Spillere - 274924

Gimena Alamón - 243518

Carmela Sotuyo - 186554

Mayo 2023

Índice

| | | |
|----------|-------------------------------------|----------|
| 1 | Aplicación de TDD | 3 |
| 1.1 | Descripción General | 3 |
| 1.2 | Definición de Tests | 3 |
| 1.2.1 | User | 4 |
| 1.2.2 | Article - Comment - Reply | 4 |
| 1.2.3 | Session | 5 |
| 1.3 | Cobertura | 6 |
| 1.3.1 | Lógica de negocio | 6 |
| 1.3.2 | Acceso a datos | 6 |
| 1.3.3 | API | 7 |
| 2 | Clean Code | 8 |

Aplicación de TDD

1.1 Descripción General

Para el desarrollo de la aplicación se utilizó la metodología TDD (Test Driven Development). Al principio no se tuvo muy claro qué enfoque aplicar, ya que se empezó por los tests del dominio, pero luego resultó una mejor estrategia para el equipo tomar el enfoque Outside-in y comenzar por las pruebas de la API, es decir los tests de los controllers, luego por los tests de la lógica de negocio y por último por el acceso a datos.

Como estrategia para implementar la metodología TDD se utilizó outside - in. Se comenzó realizando los tests de los controllers, luego la de la lógica y el dominio y por último la de la base de datos. Al comienzo del desarrollo se dejó lista la estructura de la aplicación en cuanto a proyectos, y luego se fueron añadiendo funcionalidades en base a los tests desarrollados.

1.2 Definición de Tests

Para definir los tests se crearon proyectos que los contuvieran. Los mismos son:

- WebApi.Test
- BusinessLogic.Test
- DataAccess.Test
- Domain.Test

Para poder simular objetos y comportamiento en las pruebas, es decir realizar pruebas sobre la lógica y acceso a datos, se utilizó el framework de Mocking "Moq".

Detalles de commits realizados aplicando TDD así como evidencia de las pruebas a continuación:

1.2.1 User

- 9e8fc90
- d466455
- 7c77273
- d466455
- e107976
- f48a359
- 6cb8cdd

1.2.2 Article - Comment - Reply

- 1cc3483
- c05fc97
- 10b35ce
- d1393fa
- 457f5ea
- 625d443
- 121423b
- 14a23d3
- eb00587
- 58c0386
- 025c362

- faabb1c
- c1a4f46
- 9f45c14
- 62d7ee8
- 65f9f66
- d53d0e8
- c83a0e6
- ed54645
- f167d44
- 0dacc54
- f04067f
- 6f47574

1.2.3 Session

- 40ed439
- 5a761d5
- cb27125
- 9f5d924
- 58781d1
- bee65e8

1.3 Cobertura

1.3.1 Lógica de negocio

En este proyecto tenemos una cobertura del 96%, dejando un 4% sin cubrir. Este restante se debe a pequeñas secciones de la lógica para las cuales no se desarrollaron tests al considerarlo innecesario.

A continuación se muestra la evidencia de dicha cobertura:

| | Covered (%Blocks) | Not Covered (%Blocks) |
|----------------|-------------------|-----------------------|
| ▶ ArticleLogic | 92,79% | 7,21% |
| ▶ CommentLogic | 100,00% | 0,00% |
| ▶ ReplyLogic | 100,00% | 0,00% |
| ▶ SessionLogic | 92,31% | 7,69% |
| ▶ UserLogic | 96,74% | 3,26% |

Figure 1.1: Cobertura Business Logic

1.3.2 Acceso a datos

En este proyecto tenemos una cobertura del 93%, dejando un 7% sin cubrir. Este restante se debe a que surgieron algunos inconvenientes con el uso de Moqs y detalles menores quedaron sin cubrir.

A continuación se muestra la evidencia de dicha cobertura:

| | Covered (%Blocks) | Not Covered (%Blocks) |
|--------------------------------------|-------------------|-----------------------|
| { } BlogsApp.DataAccess.Repositories | 92,52% | 7,48% |
| ▶ ArticleRepository | 100,00% | 0,00% |
| ▶ CommentRepository | 85,00% | 15,00% |
| ▶ ReplyRepository | 78,05% | 21,95% |
| ▶ SessionRepository | 100,00% | 0,00% |
| ▶ UserRepository | 100,00% | 0,00% |

Figure 1.2: Cobertura DataAccess

1.3.3 API

Se adjunta evidencia de la ejecución de los tests de la API:







| | Covered (%Blocks) | Not Covered (%Blocks) |
|--|-------------------|-----------------------|
| { } BlogsApp.WebAPI.Controllers | 98,32% | 1,68% |
| ▶  ArticleController | 100,00% | 0,00% |
| ▶  CommmentController | 100,00% | 0,00% |
| ▶  LogController | 100,00% | 0,00% |
| ▶  ReplyController | 100,00% | 0,00% |
| ▶  SessionController | 100,00% | 0,00% |
| ▶  UserController | 93,94% | 6,06% |

Figure 1.3: Cobertura WebAPI

Clean Code

Se cumplen con los lineamientos de Clean Code (capítulos 1 al 10, y el 12), utilizando las técnicas y metodologías ágiles presentadas para crear código limpio y que así puede ser comprendido con facilidad por un tercero, así como ser mantenido de sin mayor esfuerzo y escalable a nuevas funcionalidades.

Se aplican principios de diseño como el Principio de segregación de interfaz y Principio de inyección de dependencias. También se busca que los métodos sean específicos, es decir, tengan una única responsabilidad, y que tanto los nombres de los mismos y de las variables sean nemotécnicos para ser fácilmente comprendidos.

Como por ejemplo este método que cumple ser breve y no realizar controles que no pasan bajo su responsabilidad. A la vez en que su firma es igual a la firma del controlador que lo usa, lo cual hace fácilmente comprensible el hecho de donde es invocado.

```
public Article GetArticleById(int id, User loggedInUser)
{
    return _articleRepository.Get(ArticleById(id, loggedInUser));
}
```

Figure 2.1: Método GetArticleById

Para una buena organización del equipo se utilizó la herramienta Notion, con el fin de definir los requerimientos solicitados y hacer un seguimiento del trabajo. Se utiliza GitFlow como flujo de trabajo en GitHub, en conjunto con Pull Requests cada vez que se desea mergear a dev una nueva funcionalidad, con el fin de que todos los integrantes pudieran hacer Code Review, realizando sugerencias o aprobando el PR.

Evidencia de Pull requests:















| <input type="checkbox"/> |  0 Open | <input checked="" type="checkbox"/> 42 Closed | Author ▾ | Label ▾ | Projects ▾ | Milestones ▾ | Reviews ▾ | Assignee ▾ | Sort ▾ |
|--------------------------|--|---|---|---------|------------|--------------|-----------|------------|--------|
| <input type="checkbox"/> |  | Bug/test fixes | #42 by carmesotuyo was merged 30 minutes ago • Approved | | | | | | |
| <input type="checkbox"/> |  | Bug/article dtos | #41 by ferspi was merged 2 hours ago • Approved | | | | | | |
| <input type="checkbox"/> |  | se arregla bug de crear article | #40 by ferspi was merged 4 hours ago | | | | | | |
| <input type="checkbox"/> |  | Bug/log entry search query not nulleable | #39 by ferspi was merged 6 hours ago • Approved | | | | | | |
| <input type="checkbox"/> |  | Feature/reply repository | #38 by ferspi was merged 15 hours ago • Approved | | | | | | |
| <input type="checkbox"/> |  | Feature/comment logic | #37 by carmesotuyo was merged 7 hours ago • Approved | | | | | | |
| <input type="checkbox"/> |  | Feature/comment repository | #36 by ferspi was merged 17 hours ago • Approved | | | | | | |
| <input type="checkbox"/> |  | Feature/logger service | #35 by ferspi was merged 18 hours ago • Approved | | | | | | |
| <input type="checkbox"/> |  | Bug/session fixes | #34 by carmesotuyo was merged 20 hours ago • Approved | | | | | | |
| <input type="checkbox"/> |  | added exception catch and messages | #33 by carmesotuyo was merged yesterday • Approved | | | | | | |
| <input type="checkbox"/> |  | Bug/corregir articles test | #32 by ferspi was merged yesterday • Approved | | | | | | |
| <input type="checkbox"/> |  | Feature/fix get logged user | #31 by carmesotuyo was merged yesterday • Approved | | | | | | |
| <input type="checkbox"/> |  | Feature/session logic | #30 by carmesotuyo was merged yesterday | | | | | | |

Figure 2.2: Pull requests