

Universidad ORT Uruguay
Facultad de Ingeniería

**Obligatorio 1 -Diseño de aplicaciones 2 -
Evidencia de Clean Code y de la
aplicación de TDD**

Nicolas Hernandez – 229992
Francisco Aguilar – 230143

2023

Índice

Descripción de la estrategia de TDD seguida	3
Evidencia de TDD	3
Usuarios	3
Comentarios	4
Artículos	5
Evidencia de Clean Code	7
Informe de cobertura	10
Cobertura de código de Registro, modificación, logueo y deslogueo de usuarios	10
Cobertura de código de comentarios	10
Cobertura de código de buscar artículo a partir de un texto	11
Cobertura de administración de usuarios	12
Análisis de las métricas de cobertura	12
Cobertura general del proyecto	12

Abstract

En este documento se encuentra toda la información necesaria sobre la evidencia del uso de las prácticas de clean code y de TDD utilizadas en el obligatorio de diseño de aplicaciones 2, aquí se van a poder observar las diferentes evidencias de tanto TDD como clean code mediante capturas de pantallas de los commits de github y del código.

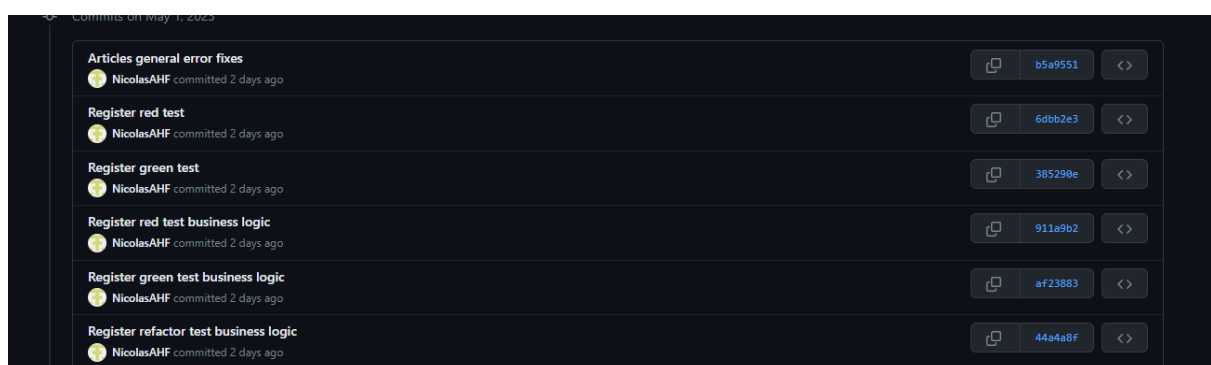
Descripción de la estrategia de TDD seguida

La estrategia que escogimos fue la de outside-in ya que primero creamos el test red para que fallara luego el green, después el refactor y continuamos el bucle hasta obtener la funcionalidad requerida, como se puede ver en los diferentes commits, se observa que íbamos poniendo en qué etapa del bucle nos encontrábamos para cada caso, hasta culminar con el redactor final de cada implementación y el test en green.





Evidencia de TDD



Se utilizó TDD en la mayoría de implementaciones se dejan capturas a continuación:

Usuarios















Comentarios


	NicolasAHF committed 2 weeks ago
	Added DeleteCommentById Method
	New test for commentController
	Refactored commentController

	Added DeleteCommentById Method
	Reply comment red test
	Reply comment green test
	Reply comment refactor test


Artículos

	Commits on Apr 23, 2023
GetArticleById red test controller	
 NicolasAHF committed last week	
GetArticleById green test controller	
 NicolasAHF committed last week	
ArticleController Refactor test	
 NicolasAHF committed last week	
GetAllArticles Red test controller	
 NicolasAHF committed last week	
GetAllArticles Green test controller	
 NicolasAHF committed last week	
GetAllArticles FailTest red controller	
 NicolasAHF committed last week	
GetAllArticles FailTest green controller	
 NicolasAHF committed last week	
GetAllArticles Refactor Test controller	
 NicolasAHF committed last week	
CreateArticle Red Test Controller	
 NicolasAHF committed last week	
CreateArticle Green Test Controller	
 NicolasAHF committed last week	
CreateInvalidArticle Red Test Controller	
 NicolasAHF committed last week	


GetArticleByTextValidTest red test controller

 **NicolasAHF** committed last week


GetArticleByTextValidTest green test controller

 **NicolasAHF** committed last week

GetArticleByTextValidTest refactor test controller

 **NicolasAHF** committed last week


UpdateValidArticle red test controller

 **NicolasAHF** committed last week


UpdateValidArticle green test controller

 **NicolasAHF** committed last week


UpdateValidArticle refactor test controller

 **NicolasAHF** committed last week


DeleteArticle red test controller

 **NicolasAHF** committed last week


DeleteArticle green and refactor test test controller and refactor of... ...

 **NicolasAHF** committed last week


CreateArticle red test Logic

 **NicolasAHF** committed last week


CreateArticle green test Logic

 **NicolasAHF** committed last week


GetAllArticles red test Logic

 **NicolasAHF** committed last week


GetAllArticles green test Logic

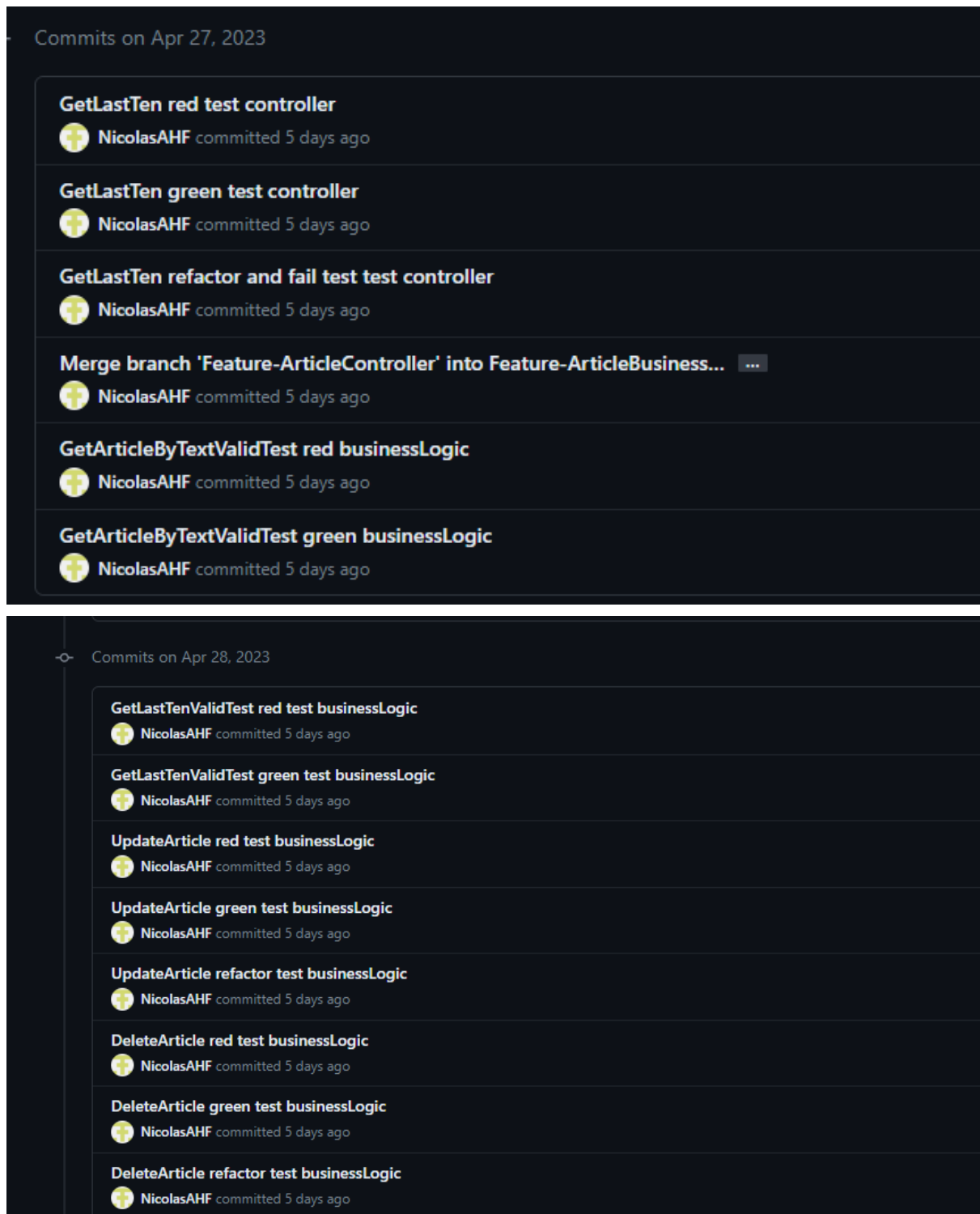
 **NicolasAHF** committed last week

GetArticleById red test Logic

 **NicolasAHF** committed last week

GetArticleById green test Logic

 **NicolasAHF** committed last week

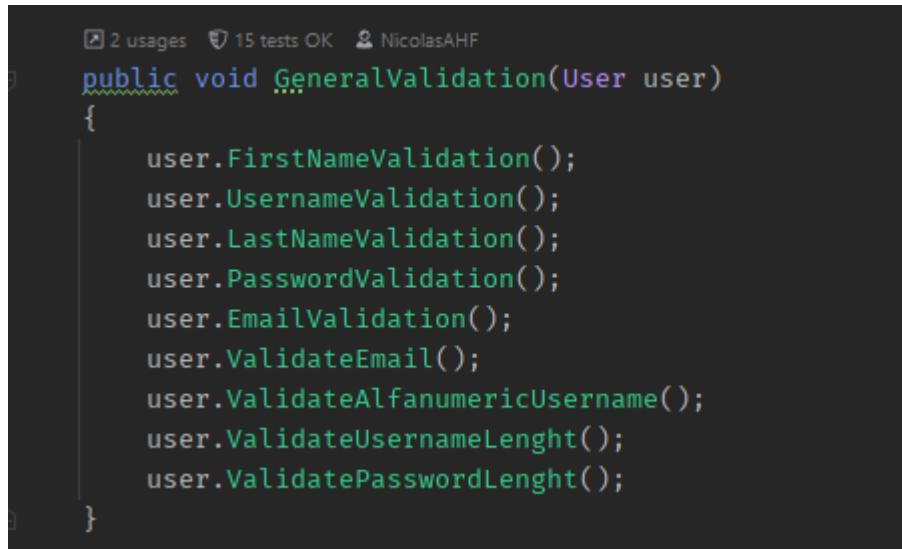


Evidencia de Clean Code

Durante nuestro proyecto buscamos guiarnos siempre por los diferentes principios de clean code para lograr un código fácil de leer, entendible para otros desarrolladores, reutilizable y con las mejores prácticas posibles. Uno de los principales puntos que seguimos de clean code fue evitar el uso de comentarios, se

puede apreciar que en nuestra solución es casi inexistente el uso de comentarios en el código, ya que tratamos que el mismo sea lo más descriptivo posible.

Otro de los puntos en los que nos enfocamos es en que cada función realice solo una tarea, en el ejemplo que se puede ver a continuación mostramos como para la validación de los campos del usuario, cada método valida un campo y luego realizamos una validación general para llamar a todos los campos.



```
2 usages 15 tests OK NicolasAHF
1 public void GeneralValidation(User user)
2 {
3     user.FirstNameValidation();
4     user.UsernameValidation();
5     user.LastNameValidation();
6     user.PasswordValidation();
7     user.EmailValidation();
8     user.ValidateEmail();
9     user.ValidateAlphanumericUsername();
10    user.ValidateUsernameLength();
11    user.ValidatePasswordLength();
12 }
```

```

3 usages 12 tests OK NicolasAHF
public void FirstNameValidation()
{
    if (String.IsNullOrEmpty(FirstName))
        throw new ArgumentException(message: "Empty FirstName");
}

2 usages 11 tests OK NicolasAHF
public void LastNameValidation()
{
    if (String.IsNullOrEmpty(LastName))
        throw new ArgumentException(message: "Empty LastName");
}

2 usages 11 tests OK NicolasAHF
public void UsernameValidation()
{
    if (String.IsNullOrEmpty(Username))
        throw new ArgumentException(message: "Empty Username");
}

2 usages 11 tests OK NicolasAHF
public void PasswordValidation()
{
    if (String.IsNullOrEmpty>Password))
        throw new ArgumentException(message: "Empty Password");
}

2 usages 11 tests OK NicolasAHF
public void EmailValidation()
{
    if (String.IsNullOrEmpty>Email))
        throw new ArgumentException(message: "Empty Email");
}

3 usages 12 tests OK NicolasAHF
public void ValidateAlphanumericUsername()
{
    Regex regex = new Regex(pattern: @"^w+$", RegexOptions.IgnoreCase);
    if (!regex.IsMatch(Username))
    {
        throw new ArgumentException(message: "Username must be alphanumeric");
    }
}

```

```

3 usages 12 tests OK NicolasAHF
public void ValidateAlphanumericUsername()
{
    Regex regex = new Regex(pattern: @"^w+$", RegexOptions.IgnoreCase);
    if (!regex.IsMatch(Username))
    {
        throw new ArgumentException(message: "Username must be alphanumeric");
    }
}

1 usage 10 tests OK NicolasAHF
public void ValidatePasswordLenght()
{
    if (Password.Length > 16 || Password.Length < 5)
    {
        throw new ArgumentException(message: "Password must be between 16 and 5 characters");
    }
}

3 usages 12 tests OK NicolasAHF
public void ValidateUsernameLenght()
{
    if (Username.Length > 12 || Username.Length < 4)
    {
        throw new ArgumentException(message: "Username must be between 12 and 4 characters");
    }
}

3 usages 12 tests OK NicolasAHF
public void ValidateEmail()
{
    Regex regex = new Regex(pattern: @"^w+([\.-]?\w+)*@w+([\.-]?\w+)*(\.\w{2,3})+$", RegexOptions.IgnoreCase);
    if (!regex.IsMatch>Email))
    {
        throw new ArgumentException(message: "Invalid Email");
    }
}

```

Otro de los puntos de clean code que respetamos con mucha fuerza fue el número de parámetros por método, ya que el número máximo con el que cuentan nuestras funciones es 3 y ninguna pasa ese número de parámetros, logrando tener un código más limpio y funciones más cortas.

Informe de cobertura

Cobertura de código de Registro, modificación, logueo y deslogueo de usuarios

AuthController	100%	0/18
AuthController(ISessionLogic, IUserLogic)	100%	0/5
Register(RegisterDto)	100%	0/5
Login(LoginDto)	100%	0/4
Logout(Guid)	100%	0/4
SessionLogic	100%	0/28
SessionLogic(IRepository<Session>, IRepository<User>)	100%	0/5
GetLoggedUser(Guid)	100%	0/7
Login(string, string)	100%	0/11
Logout(Guid)	100%	0/5
IRepository<T>	100%	0/23
Repository(BlogDbContext)	100%	0/4
GetAll()	100%	0/3
GetBy(Expression<Func<T, bool>>)	100%	0/3
Insert(T)	100%	0/3
Delete(T)	100%	0/3
Update(T)	100%	0/3
Save()	100%	0/3
SessionRepository	100%	0/6
SessionRepository(BlogDbContext)	100%	0/3
GetBy(Expression<Func<Session, bool>>)	100%	0/3

Cobertura de código de comentarios

CommentController	100%	0/18
CommentController(ICommentLogic)	100%	0/4
PostNewComment(CommentInModel, Guid)	100%	0/6
DeleteCommentById(Guid)	100%	0/4
ReplyComment(ReplyCommentDto)	100%	0/4

▼ CommentLogic	100%	0/35
* CommentLogic(IRepository<Comment>,IArticleLogic)	100%	0/6
AddNewComment(Comment,Guid,Guid)	100%	0/10
DeleteCommentByld(Guid)	100%	0/6
ReplyComment(Guid,string)	100%	0/8
ValidateNull(Comment)	100%	0/5
▼ CommentRepository	100%	0/6
* CommentRepository(BlogDbContext)	100%	0/3
GetBy(Expression<Func<Comment,bool>>)	100%	0/3

Cobertura de código de buscar articulo a partir de un texto

▼ ArticlesController	100%	0/43
* ArticlesController(IArticleLogic)	100%	0/4
GetArticleByld(Guid)	100%	0/4
> GetAllPublicArticles()	100%	0/6
> GetAllUserArticles(string,Guid)	100%	0/6
> GetArticleByText(string)	100%	0/6
CreateArticle(CreateArticleDTO,Guid)	100%	0/5
UpdateArticle(Guid,CreateArticleDTO,Guid)	100%	0/5
DeleteArticle(Guid,Guid)	100%	0/4
GetLastTen()	100%	0/3
ArticleLogic	76%	19/80
* ArticleLogic(IRepository<Article>,ISessionLogi)	100%	0/5
GetArticleByld(Guid)	100%	0/5
GetAllArticles()	100%	0/5
GetLastTen()	100%	0/5
CreateArticle(Article,Guid)	100%	0/8
GetArticleByText(string)	100%	0/5
UpdateArticle(Guid,Article,Guid)	100%	0/12
▼ ArticleRepository	100%	0/21
* ArticleRepository(BlogDbContext)	100%	0/3
GetAll()	100%	0/3
GetBy(Expression<Func<Article,bool>>)	100%	0/3
GetByText(string)	100%	0/3
GetLastTen()	100%	0/3
GetPublicAll()	100%	0/3
GetUserArticles(string)	100%	0/3

Cobertura de administracion de usuarios

GetUserById(Guid)	100%	0/4
GetAllUsers()	100%	0/3
CreateUser(CreateUserDTO)	100%	0/5
UpdateUser(Guid,CreateUserDTO,Guid)	100%	0/5
DeleteUser(Guid)	100%	0/4

▼ UserLogic	71%	33/112
UserLogic(IRepository<User>,ISessionLogic,IRe	100%	0/6
GetUserById(Guid)	100%	0/5
GetAllUsers()	100%	0/5
CreateUser(User)	100%	0/9
UpdateUser(Guid,User,Guid)	100%	0/20
DeleteUser(Guid)	100%	0/6

▼ UserRepository	100%	0/9
UserRepository(BlogDbContext)	100%	0/3
GetAll()	100%	0/3
GetBy(Expression<Func<User,bool>>)	100%	0/3

Análisis de las métricas de cobertura

Cobertura general del proyecto

▼ Total	92%	166/2034
> Blog.DataAccess.Test	100%	0/235
> Blog.DataAccess	96%	3/76
> Blog.WebApi	96%	4/112
> Blog.BusinessLogic.Test	96%	13/369
> Blog.Models	95%	10/184
> Blog.Domain	94%	9/141
> Blog.Tests	94%	35/576
> Blog.BusinessLogic	81%	52/277
> Blog.Filters	38%	40/64

Como se puede observar en la imagen y en el punto anterior, se logró el objetivo de una cobertura entre 90% y 100% en todos los requerimientos marcados como fundamentales para este punto, en los únicos casos que se bajó del 90% fue en el paquete general de lógica de negocio, ya que para algunos métodos complementarios no realizamos los test correspondientes, porque fueron parte de un refactor posterior y también un paquete al que no se alcanzó un 90% de cobertura fue al de filtros, ya que tuvimos dificultades al momento de crear los test unitarios de

los filtros de autenticación por roles y autenticación, pudiendo solamente crear dichos test para el filtro de excepciones, decidimos enfocarnos en llegar a la cobertura de los puntos marcados como fundamentales y lograr dicho objetivo.