

Universidad ORT Uruguay

Facultad de ingeniería

Diseño de aplicaciones 2

Obligatorio 1

<https://github.com/ORT-DA2/SobralGoni>

210361 - José Pablo Goñi

192247 - Juan Pablo Sobral

ÍNDICE

	2
Justificación de clean code	3
Nombres nemotécnicos	3
Funciones	3
Comentarios	4
Las variables deben comenzar con minúscula.	4
Los métodos deben comenzar con mayúscula.	5
Las llaves de apertura y clausura sean utilizadas exclusivamente en una línea aparte.	5
Las properties deben de comenzar con mayúscula.	5
Las interfaces deben de comenzar con la letra I.	6
Testing y pruebas unitarias	6

Justificación de clean code

+ Nombres nemotécnicos

Se busca que los nombres sean intencionados y descriptivos, evitando abreviaciones, prefijos, secuencia de números en variables y palabras redundantes. Los nombres de las clases no deben ser verbos y se deben evitar sufijos. A su vez para los nombres de los métodos se deben usar verbos. Cada nombre debe ser corto, explícito y claro. Si la intención es técnica se deben usar nombres técnicos, por ejemplo cuando estamos aplicando algún patrón.

Evidencias:

```
public class User
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    32 referencias | Juanpa, Hace 19 días | 1 autor, 1 cambio | 0 excepciones
    public long Id { get; set; }

    16 referencias | Juanpa, Hace 19 días | 1 autor, 1 cambio | 0 excepciones
    public string FirstName { get; set; }

    11 referencias | Juanpa, Hace 19 días | 1 autor, 1 cambio | 0 excepciones
    public string LastName { get; set; }

    19 referencias | Juanpa, Hace 19 días | 1 autor, 1 cambio | 0 excepciones
    public string Username { get; set; }

    14 referencias | Juanpa, Hace 19 días | 1 autor, 1 cambio | 0 excepciones
    public string Password { get; set; }

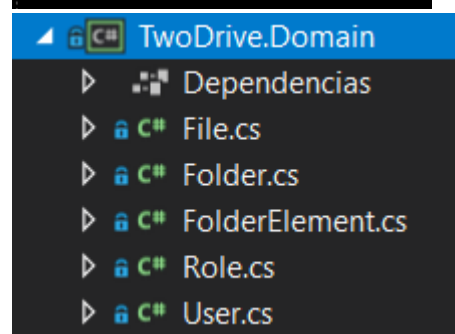
    11 referencias | Juanpa, Hace 19 días | 1 autor, 1 cambio | 0 excepciones
    public string Email { get; set; }

    13 referencias | Juanpa, Hace 11 días | 1 autor, 1 cambio | 0 excepciones
    public string Role { get; set; }

    1 referencia | Juanpa, Hace 11 días | 1 autor, 1 cambio | 0 excepciones
    public string Token { get; set; }

    12 referencias | Juanpa, Hace 11 días | 1 autor, 4 cambios | 0 excepciones
    public List<User> FriendList { get; set; }

    6 referencias | Juanpa, Hace 18 días | 1 autor, 1 cambio | 0 excepciones
    public Folder RootFolder { get; set; }
}
```



```
public void AddFolder(Folder folder)
{
    Folders.Add(folder);
}
```

+ Funciones

Se busca que las mismas sean reducidas (20 líneas) y con nombres descriptivos. Los parámetros que se le pasan a una función no deberían ser más de 3 como

máximo. Cada función debe hacer lo que dice su nombre y nada más que eso. Se recomienda extraer funciones en los try y catch.

Evidencias:

```
public void Delete(Folder entity)
{
    var regex = new Regex("$-rootFolder");
    if (regex.IsMatch(entity.Name))
    {
        throw new Exception("The name format you specified is reserved for user root folders, you can not delete it");
    }
    FolderElementExists(entity.Id);
    DeleteFolder(entity);
}
```

```
public Folder Add(Folder entity)
{
    ValidateFormat(entity);
    entity.Parent = GetFolderId(entity.Parent.Id);
    Folder folderBefore = entity.Parent;
    Folder folderAfter = entity.Parent;
    FoldersNull(folderAfter);
    _repository.Add(entity);
    folderAfter.AddFolder(entity);
    _repository.Update(folderBefore, folderAfter);
    return entity;
}
```

+ Comentarios

No debe haber comentarios salvo que sean aclaratorios y que sumen a la solución como una advertencia.

Evidencia:

```
//GET: /api/files/view?fileName=a&lastName=b
[HttpGet("view")]
0 referencias | Juanpa, Hace 1 día | 1 autor, 1 cambio | 0 solicitudes | 0 excepciones
public IActionResult GetSorted(string fileName, string sortOrder)
{
    try
    {
        List<File> files = _fileLogic.GetSortedFiles(int.Parse(User.Identity.Name), sortOrder, fileName);
        return Ok(files);
    }
    catch (Exception e)
    {
        return BadRequest(e.Message);
    }
}
```

+ Las variables deben comenzar con minúscula.

```
var regex = new Regex("$-rootFolder");
```

- + Los métodos deben comenzar con mayúscula.

```
private void IsFolderRoot(long folderId)
{
    if(_repository.Get(folderId).Parent == null)
        throw new Exception("La carpeta RAIZ no puede ser movida.");
}
```

- + Las llaves de apertura y clausura sean utilizadas exclusivamente en una línea aparte.

```
private void DeleteFolder(Folder folder)
{
    foreach (File f in folder.Files)
    {
        _fileRepository.Delete(f);
    }
    foreach (Folder f in folder.Folders)
    {
        DeleteFolder(f);
    }
    Delete(folder);
}
```

- + Las properties deben de comenzar con mayúscula.

```
public long OwnerId { get; set; }

50 referencias | Juanpa, Hace 18 días | 1 autor, 1 cambio | 0 excepciones
public string Name { get; set; }

55 referencias | Juanpa, Hace 18 días | 1 autor, 2 cambios | 0 excepciones
public Folder Parent { get; set; }

43 referencias | Juanpa, Hace 18 días | 1 autor, 1 cambio | 0 excepciones
public List<User> Readers { get; set; }
```

- + Las interfaces deben de comenzar con la letra I.

```
public interface ILogic<T>
{
    2 referencias | Jose Pablo Goñi, Hace 6 días | 2 autores, 2 cambios | 0 excepciones
    User Authenticate(string username, string password);
    3 referencias | Jose Pablo Goñi, Hace 6 días | 1 autor, 1 cambio | 0 excepciones
    long GetUserId(string username);
    2 referencias | Jose Pablo Goñi, Hace 6 días | 1 autor, 1 cambio | 0 excepciones
    long GetUserRootFolderId(string username);
    2 referencias | Jose Pablo Goñi, Hace 16 días | 1 autor, 1 cambio | 0 excepciones
    IEnumerable<T> GetAll();
    8 referencias | Jose Pablo Goñi, Hace 16 días | 1 autor, 1 cambio | 0 excepciones
    T Get(long id);
    2 referencias | Jose Pablo Goñi, Hace 6 días | 1 autor, 2 cambios | 0 excepciones
    T Add(T entity);
    2 referencias | Jose Pablo Goñi, Hace 16 días | 1 autor, 1 cambio | 0 excepciones
    void Update(T Entity, T newEntity);
    2 referencias | Jose Pablo Goñi, Hace 16 días | 1 autor, 1 cambio | 0 excepciones
    void Delete(T Entity);
    2 referencias | Jose Pablo Goñi, Hace 23 horas | 1 autor, 1 cambio | 0 excepciones
    void AddFriend(T user, T userFriend);
    2 referencias | Jose Pablo Goñi, Hace 23 horas | 1 autor, 1 cambio | 0 excepciones
    void RemoveFriend(T user, T userFriend);
}
```



















Testing y pruebas unitarias

Pruebas unitarias.

Se debe cumplir en todo el desarrollo con las 3 leyes de TDD:

- No hay que crear código hasta que haya fallado un unit test
- No hay que crear nunca más de una prueba que falle
- El código creado debe ser el mínimo para que la prueba pase

Evidencia:

AddRemoveReaders	JosePabloGoni committed 13 days ago	 280cad4	
DelteFolder	JosePabloGoni committed 13 days ago	 96fb464	
UpdateFolder	JosePabloGoni committed 13 days ago	 7a84ad8	
AddFolder	JosePabloGoni committed 13 days ago	 738313b	
getAll	JosePabloGoni committed 13 days ago	 782de0e	
GetFolder	JosePabloGoni committed 13 days ago	 02058f4	
GetFolderNotExist	JosePabloGoni committed 13 days ago	 365365e	
MoveFileToRootCorrectly	JosePabloGoni committed 13 days ago	 6ace1a1	
AddFolderNullToRootTest	JosePabloGoni committed 13 days ago	 5ea84b8	

Por ejemplo, entramos a uno de los commits que se hicieron:
Tenemos que primero se realizó la prueba necesaria, que nos dio que no se había implementado aún nada. Evidenciamos esto a continuación:

```

7 TwoDrive.Test/BusinessLogic/LogicFolderTest.cs
@@ -129,5 +129,18 @@ public void DeleteFolder()
129 logicFolder.Delete(logicFolder.Get(4));
130 logicFolder.Get(4);
131 }
132 + [TestMethod]
133 + public void AddReaders()
134 + {
135 +     logicFolder.AddReader(logicFolder.Get(3), user2.Id);
136 +     Assert.IsTrue(logicFolder.Get(3).Readers.Count == 2);
137 + }
138 +

```

Y luego de darnos cuenta de que no se había implementado aún la lógica para agregar lectores a un usuario, nos dirigimos a la lógica de carpetas e implementamos la misma. Evidenciamos esto:

```

7 TwoDrive.BusinessLogic/LogicFolder.cs
@@ -6,7 +6,7 @@
40 +
41 + public void AddReader(Folder Entity, User user)
42 + {
43 +     throw new NotImplementedException();
44 + }
45 +

```

Invitamos a que se verifique esto en los otros commits generados a lo largo del proyecto.

Jerarquía	No cubiertos (bloques)	No cubiertos (% de bloques)	Cubiertos (bloques)	Cubiertos (% de bloques)
PC_JOSEPABLOGONI 2019-10-08 ...	532	19.76%	2160	80.24%
twodrive.businesslogic.dll	309	44.85%	380	55.15%
TwoDrive.BusinessLogic	309	44.85%	380	55.15%
FileLogic	19	12.42%	134	87.58%
FolderLogic	55	25.94%	157	74.06%
ReportLogic	141	100.00%	0	0.00%
ReportLogic.<>c	37	100.00%	0	0.00%
ReportLogic.<>c_Dis...	3	100.00%	0	0.00%
ReportLogic.<>c_Dis...	5	100.00%	0	0.00%
ReportLogic.<>c_Dis...	11	100.00%	0	0.00%
ReportLogic.<>c_Dis...	11	100.00%	0	0.00%
UserLogic	27	24.77%	82	75.23%
UserLogic.<>c_Displa...	0	0.00%	3	100.00%
UserLogic.<>c_Displa...	0	0.00%	4	100.00%
twodrive.businesslogic.interfa...	29	38.16%	47	61.84%
TwoDrive.BusinessLogic.In...	29	38.16%	47	61.84%
FolderElementLogic.Us...	6	100.00%	0	0.00%
FolderElementLogic<T>	23	32.86%	47	67.14%
twodrive.dataaccess.dll	66	22.45%	228	77.55%
TwoDrive.DataAccess	66	22.45%	228	77.55%
FileRepository	2	2.86%	68	97.14%
FolderRepository	2	2.17%	90	97.83%
LogRepository	16	100.00%	0	0.00%
UserRepository	46	39.66%	70	60.34%
twodrive.dataaccess.interface....	29	93.55%	2	6.45%
TwoDrive.DataAccess.Inter...	29	93.55%	2	6.45%
TwoDriveContext	29	93.55%	2	6.45%
twodrive.domain.dll	6	8.11%	68	91.89%
TwoDrive.Domain	6	8.11%	68	91.89%
File	0	0.00%	6	100.00%
Folder	0	0.00%	16	100.00%
FolderElement	0	0.00%	16	100.00%
LogItem	4	40.00%	6	60.00%
User	2	7.69%	24	92.31%

Por falta de tiempo se priorizo la funcionalidad del sistema, se creó la lógica para ordenar los archivos y carpetas, como también la funcionalidad para poder ver cuantas modificaciones se tuvo en un dia(La parte de reportes del sistema). En todo lo otro se realizó tdd correctamente, las interfaces se crearon refactorizando por eso se muestra en la imagen que no tienen mucha cobertura, eso hace que el nivel de casos cubiertos también baje.