UNIVERSIDAD TECNOLÓGICA NACIONAL – FACULTAD REGIONAL CÓRDOBA

DEPARTAMENTO DE SISTEMAS

INGENIERIA Y CALIDAD DE SOFTWARE

TRABAJO PRÁCTICO N°1: *User Story*

CURSO: 4K1

GRUPO: 4

ESTUDIANTES: Aguilera, Guadalupe - 85352 - guadi1307@gmail.com

Benegas, Catalina - 85616 - catalinabenegas@gmail.com

Garnero, Constanza - 85648 - constanzagarnero@gmail.com

Lavra, Andrés - 86147 - andresnex@gmail.com

Romero, Manuel - 87210 - manuromero0801@gmail.com

FECHA DE ENTREGA: 12 de septiembre de 2023

Documento de estilo de código

Realizamos este documento para establecer un conjunto básico de reglas para escribir código que usaremos en el equipo. Usaremos mayormente estándares de C#.

Convenciones de nomenclatura

Los espacios de nombres serán todos PascalCase.

NO:

```
ingenieriasoftware.tp6.principal
```

SI:

```
IngenieriaSoftware.TP6.Principal
```

Las clases serán PascalCase, los métodos y los atributos serán camelCase.

NO:

```
public class miclase
{
    private String soyunatributo;
    public void soyunmetodo()
    {
        // código muy bueno…
    }
}
```

SI:

```
public class MiClase
{
    private String soyUnAtributo;
    public void soyUnMetodo()
```

```
        {
            // código muy bueno…
        }
    }
```

Los campos estáticos serán PascalCase.

NO:

```
public static int _algunValorEstatico = 10;
```

SI:

```
public static int AlgunValorEstatico = 10;
```

Los parámetros deben ser camelCase.

NO:

```
public void soyUnMetodoDistinto(bool AlgunaCondicion)
```

SI:

```
public void soyUnMetodoDistinto(bool algunaCondicion)
```

Añadir prefijo On a eventos y acciones.

```
public UnityAction OnClick;
```

Añadir prefijo I a interfaces.

```
public interface ISoyUnaInterfaz;
```

Mejores prácticas básicas

Declaraciones

Escribir siempre modificadores de nivel de acceso.

NO:

```
int privateVariable;
```

SI:

```
private int privateVariable;
```

Utilizar una sola declaración por línea.

NO:

```
private int primeraVariable, segundaVariable;
```

SI:

```
private int primeraVariable;
private int segundaVariable;
```

Declarar sólo una clase / interfaz por archivo fuente, con la excepción de las clases internas.

Espaciado

Colocar una **línea en blanco entre los métodos** para visualizar mejor la estructura de la clase y colocar una **línea en blanco dentro de los métodos para separar la funcionalidad** (por ejemplo: un método que crea un objeto debe tener su parte de creación y su parte de inicialización separadas).

Estilo de los corchetes

Todas las llaves tienen su propia línea.

NO:

```
public void hagoAlgo(){
    // código muy bueno…
}
```

SI:

```
public void hagoAlgo()
{
    // código muy bueno…
```

```
}
```

Las sentencias "case" deben ser indentadas desde la sentencia switch de esta manera:

```
switch (algunaExpresión)
{
    case 0:
        {
            hacerAlgo();
        }
        break;
    caso 1:
        {
            hacerAlgoMás();
        }
        break;
    caso 2:
        {
            int n = 1;
            hacerOtraCosa(n);
        }
        break;
}
```

Usar siempre llaves.

NO:

```
for(int i=0; i<10; i++)
```

```
            ejecutaAlgo();
    SI:
        for(int i=0; i<10; i++)
        {
            ejecutaAlgo();
        }
```

Campos públicos

Definiremos todos los atributos de los objetos como **privados**. Para acceder a ellos crearemos propiedades públicas, que contienen los métodos set y get de los atributos. Las propiedades se escribirán con **PascalCase**.

```
private string nombre;
public string Nombre
{
    get
    {
        return this.name;
    }
    set
    {
        this.name = value;
    }
}
```

Estructura de clase:

Cada clase llevará la siguiente estructura:

using [*librería*]

[*variablesConstantes*]

public class [*NombreClase]*

private [*tipo*] [*nombreAtributo*]

public [*retorno*] [*método*]

Comentarios:

// única línea

/*

párrafo

*/