UNIVERSIDAD TECNOLÓGICA NACIONAL

FACULTAD REGIONAL CÓRDOBA
INGENIERÍA EN SISTEMAS DE INFORMACIÓN

INGENIERÍA Y CALIDAD DE SOFTWARE

TRABAJO PRÁCTICO N.º 6: "REQUERIMIENTOS ÁGILES - IMPLEMENTACIÓN DE USER STORIES"

Curso: 4K4

Grupo 12:

78996, Aimar, Pablo

83548, Capobianco, Francisco

84056 Diaz Posse, Felipe

78733, Gatica, Federico

78490, Toledano Gonzalez, Mateo

Docentes:

- Ing. Laura Covaro - lcovaro@gmail.com
- Ing. Gerardo Boiero - gboiero@gmail.com
- Ing. Mickaela Crespo - mickaelacrespo@gmail.com
- Ing. Cecilia Massano - ceciliamassano@gmail.com

Fecha de presentación: 5/05/23

Documento de estilo de código

Reglas para codificación en TypeScript

1) Organización de los ficheros

Ficheros de código fuente

Mantener las clases y los ficheros cortos, con no más de 2.000 líneas de código y que estén claramente divididas en estructuras.

Estructura de directorios

No utilizar puntos en el nombre de los directorios. Esto hará más fácil la asociación entre directorios y espacios de nombres. Ejemplo: usar MiEmpresa/Proyecto/Datos.

2) Convenios de nombres

Directivas para asignación de nombres

- Un nombre debe describir la función semántica del elemento, es decir, qué hace, o qué valor representa.

Nombres de clases

- Usar el estilo PasCal.
- Utilizar la notación generada por defecto como parte del framework Angular.

Nombres de variables

- Usar el estilo Camel Case.
- Utilizar i, j, k, l, m, n, etc. para los contadores locales cuando se utilizan para bucles triviales.

Nombres de métodos

- Usar el estilo Camel Case.
- Aquellos métodos de procedimientos (sin retorno) se nombrarán comenzando con verbos infinitivos que expresen la acción que se quiere realizar.

- Aquellos métodos con tipo de retorno booleano se nombrarán comenzando ‘es’ o ‘esta’ seguido de una descripción clara y corta del tipo de propiedad que está validando.

Nombres de propiedades

- Usar el estilo Camel Case.

3) Indentación

Espacios en blanco

- Utilizar Tab para indentar el código. Nunca utilizar espacios.

Ajuste de línea

Cuando una expresión no quepa en una sola línea de código, dividirla de acuerdo con estos principios:

- Nueva línea después de una coma.
- Nueva línea después de un operador aritmético.
- Alinear la nueva línea con el comienzo de la sección en la que se encuentra el código. Para mantener las líneas alineadas usar Tab y complementar con espacios.

4) Comentarios

Comentarios de bloque

Los comentarios de bloque deben ser evitados para simplificar las descripciones.

Comentarios de línea

Estos comentarios deben tener el mismo nivel de indentación que el código que describen

5) Sentencias

Sentencias simples

- Cada línea debe contener sólo una sentencia.

Sentencias de retorno

- Una sentencia de retorno no debe utilizar paréntesis para encerrar el valor de retorno.

6) Declaraciones

Declaraciones de variables locales

- Sólo una declaración por línea.

- La declaración de los atributos de cada clase se hace antes de su constructor.
- De ser posible, si existen grupos de variables con características comunes, reunir sus declaraciones en líneas contiguas precedidas de un comentario de línea con una breve descripción de su utilidad, separado de otras variables mediante una línea vacía.

Inicializaciones

- Inicializar las variables locales durante la inicialización.

7) Espaciado

Líneas en blanco

Usar una línea en blanco entre:

1. Secciones lógicas de un fichero
2. Definiciones de clases.
3. Métodos o Propiedades.
4. Sentencias y métodos definidos a continuación en forma anidada.

Espacios entre términos

- Debe haber un espacio luego de una coma o un punto y coma.
- Debe haber un espacio alrededor de los operadores (excepto los unarios, como el de incremento o la negación lógica).