GUÍA DE ESTILOS DE CÓDIGO

Índice

Índice **2**

Introducción **3**

Estructura de carpetas y archivos **4**

Nombrado de archivos **4**

Escritura de código fuente **4**

Typescript 4

HTML 5

CSS 6

Introducción

Esta guía de estilos y buenas prácticas de código fue elaborada para tener un estándar de calidad y legibilidad a seguir durante la implementación del trabajo práctico 6 de la materia Ingeniería de Software.
Este documento definirá reglas de nombrado, estructura de carpetas y distintos lineamientos a seguir a la hora de escribir archivos de código fuente dependiendo del lenguaje utilizado.
Muchas de las prácticas derivan de la guía de estilo de código oficial de Angular y de las mismas prácticas que la herramienta TSLint incluida con el transpilador de Typescript trata de hacer cumplir al trabajar con archivos de tipo ts.

Estructura de carpetas y archivos

- En un principio se respetará la estructura de carpetas ya generada por el cliente de Angular, que realiza cosas como la creación de carpetas para cada componente creado, separación de las librerías importadas del código fuente del proyecto en sí, separación del código del proyecto y de los archivos distribuibles generados con el comando ‘ng build’, etc.
- Partiendo de esta estructura separaremos componentes, interfaces, servicios, pipes, clases y archivos estáticos en carpetas separadas (una carpeta por cada tipo de componente de código).
- Al tratarse de una implementación pequeña, no haremos una separación de los distintos componentes en módulos como si se realizaría en aplicaciones de Angular más grandes y complejas. Todo se mantendrá dentro de un mismo módulo (El módulo AppModule creado por defecto).
- Para mayor simplicidad en el desarrollo no realizaremos testing automáticos, por lo que los componentes generados no contarán con los archivos generados por Angular dedicados a esta tarea.

Nombrado de archivos

- Respecto a componentes generados por Angular, respetaremos el nombrado definido por el cliente de Angular al momento de su generación, haciendo uso del estilo kebab-case. Usaremos nombres claros que representen el contenido del componente.
- Respecto a archivos estáticos, trataremos de darles nombres claros y representativos de su contenido, haciendo uso del estilo snake_case para esto.

Escritura de código fuente

Typescript

Se respetará el estilo impuesto y controlado por el mismo intellisense de typescript, además de algunos otros acordados con el equipo:

- Se usarán comillas simples para las cadenas de caracteres.

- Los nombres de las clases e interfaces se escribirán con PascalCase.
- Todas las variables y parámetros tendrán definido su tipo.
- Todas las funciones y métodos tendrán definido un tipo de retorno.
- Todos los imports necesarios se escribirán al principio del archivo y estarán separados del resto del código por al menos 2 saltos de línea.
- Las variables se nombrarán utilizando camelCase.
- Los componentes de tipo servicio tendrán definida la URL del recurso a obtener como un atributo privado.
- Se hará uso de un indentado de 2 espacios para mayor legibilidad.
- Se agregará un nivel de indentado cada vez que se abra un bloque de código (función, clase, paréntesis de más de una línea, etc) y se quitará uno cada vez que se cierre.
- Después de cada punto y coma se realizará un salto de línea, no habrá más de una instrucción de código por línea.
- Todas las instrucciones de código terminarán con punto y coma por más que el lenguaje estrictamente no lo requiera en todos los casos.
- Los objetos definidos en JSON respetarán una estructura como la siguiente:

```
objeto: ClaseObjeto = {
     'atributo1': valor1,
     'atributo2': valor2,
     'atributo3': valor3,
     'atributo4': valor4,
};
```

  Respetando espacios, saltos de línea e indentado.
- Después de cada coma se pondrá un espacio para una mejor distinción entre los valores separados por esta.

HTML

- Los nombres de las clases e identificadores se escribirán utilizando kebab-case
- Los elementos que no requieren un tag de cierre (tales como <img>, <input>,
, etc) no los incluirán.
- Se utilizarán tabulaciones de 2 espacios para representar niveles de anidación de elementos.

CSS

- Las propiedades se escribirán en notación corta o ‘shorthand’.
- Implementaremos la filosofía ‘mobile first’, escribiendo primero los estilos como queremos que se vean en un dispositivo móvil y luego, en caso de que sea necesario un ajuste, los estilos para otros tamaños de pantalla dentro de media queries ubicados al final del archivo.