**Estándares De Desarrollo FrontEnd**  
  
Para mayor detalle sobre los estándares y mejores prácticas en Angular consultar:

<https://angular.io/guide/styleguide>  
   
**Estándares:**

1. **Nombramiento:**
   1. En general las nombres de variables o clases deben ser en español a menos de que sea un nombre técnico sin traducción o el framework requiera que el nombre sea en inglés (ej: Component).

* 1. Si el nombre de una variable o clase contiene la letra ‘ñ’, debe ser reemplazada por ‘ni’ (ej: Cumpleaños -> Cumpleanios) para evitar problemas durante la compilación y ejecución del código.

* 1. Los nombres de los artefactos del framework deben ser en Kebab Case y singular (ej: cuenta-bancaria.component.ts, lista-cuentas-bancarias.component.ts).
  2. Por facilidad de integración con la herramienta CLI los nombres de tipos de archivo deben ser en inglés (ej: cumpleaños.**component**.ts, cumpleaños.**service**.ts).
  3. Clases: Las clases deben nombrarse usando Pascal Case.
  4. Interfaces: Las interfaces deben nombrarse usando Pascal Case y deben empezar con una letra ‘i’ mayúscula. (ej: IComponenteBase).
  5. Propiedades: Las propiedades de una clase deben ser en Camel Case (ej: listaUsuarios) y deben ir antes del constructor.
  6. Carpetas: Las carpetas dentro del proyecto deben ser nombradas en Kebab Case (ej: lista-usuarios)
  7. Métodos: Los métodos y los parámetros de éstos deben ser nombrados en Camel Case (ej: calcularSalario)
  8. Variables: El nombramiento de variables internas de un método debe ser Camel Case y se deben declarar usando **let** o **const**, dependiendo de su mutabilidad. Evitar a toda costa el uso de **var.**
  9. El nombre del selector de un componente debe ser en Kebab Case y debe iniciar por sl (ej: sl-lista-usuarios).

1. **Mejores prácticas**
   1. Se prefiere el uso de funciones flecha para las funciones pasadas como parámetros o callbacks. (ej: *elemento.subscribe(datos => return datos)* );
   2. Se sugiere el uso de forEach, map, filter y reduce para la manipulación de arreglos por encima de ciclos for tradicionales.
   3. Se recomienda importar solamente los elementos del módulo que serán utilizados y no todo el módulo completo. Hacer uso de *import { elemento } from ‘modulo’* y no de *require.*
   4. Los elementos que tengan un decorador (@Input, @Output, @ViewChild) deben ir al inicio de la clase, antes de los atributos.

* 1. Hacer uso de @Input y @Output para la comunicación entre servicios hijos y padre. En caso de que sea necesario comunicar 2 componentes sin parentesco se debe usar un servicio.

* 1. Declarar el tipo de la variable siempre que sea posible.
  2. Realizar la inicialización del componente en el método ngOnInit y no en el constructor.

* 1. Hacer uso de comillas simples (‘ ’) para las cadenas de texto y *backtick (` `)* para la interpolación de cadenas (ej: ‘hola mundo’, `hola ${usuario.nombre}`).

* 1. Se prefiere el uso de interpolación de cadenas sobre la concatenación de cadenas.

* 1. Hacer uso de === y !== para las comparaciones. Evitar el uso de == y != ya que puede generar comportamiento no esperado.

* 1. La plantilla y los estilos deben ser archivos externos al componente. No usar estilos *inline* en la plantilla, usar el archivo scss.