**Estándar de nombramiento para web en Angular**

A continuación, se define el estándar de nombramiento para el proyecto Angular app golpi. El idioma utilizado para el proyecto será el español, con excepción de las palabras más comunes en inglés, como lo son las palabras: Login, Generic, Pages. Repository etc.

**Directorios**

* Nombre en minúscula.
* Si el nombre está compuesto por dos o más palabras, se separa por guión.

**Archivos**

* Nombre en minúscula.
* Si el nombre está compuesto por dos o más palabras se separa por guión.
* Para la mayoría de archivos se añade un sufijo de acuerdo al tipo de archivo que pertenece. El sufijo es precedido de un punto, por ejemplo, clientes.component.ts| html| css| spec.ts clientes.service.ts| spec.ts

Este estándar se sigue de acuerdo a la guía de estilos propuesta por Angular: [https://angular.dev/style-guide#naming](https://angular.dev/style-guide" \l "naming)

**Clases**

* Utilizando UpperCamelCase donde cada palabra inicia en mayúscula.
* El nombre de la clase es un sustantivo.
* Para la creación de dto’s se define el nombre del objeto y se añade al final el sufijo DTO en mayúscula.
* En caso de utilizar acrónimos se escribe totalmente en mayúscula.
* En caso de utilizar abreviaturas de más de 2 letras se utiliza UpperCamelCase.
* Deben ser claros y dicientes.

**Métodos**

* Inicia con minúscula. Utilizando lowerCamelCase.
* Su nombre inicia con un verbo en infinitivo.
* Deben ser claros y dicientes.

**Variables y valores Enum**

* Inician con minúscula. Utilizando lowerCamelCase.
* No utilizar prefijos.

**Constantes**

* Utilizando UpperCase.
* Si el nombre está compuesto por dos o más palabras, se separa por guión bajo.

**Selectores**

* Nombre en minúscula.
* Si el nombre está compuesto por dos o más palabras, se separa por guión.

**Orden de las directivas**

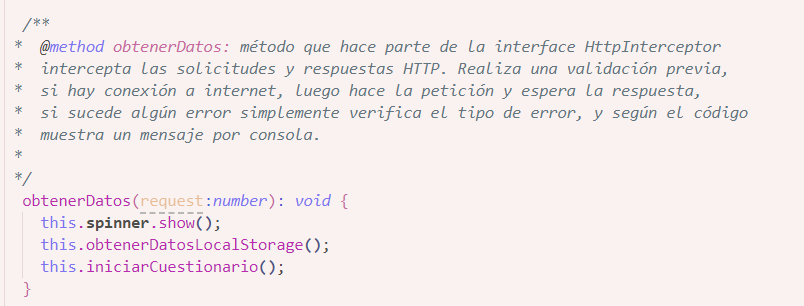
* A continuación, se especificará el orden de las importaciones:
* Las importaciones propias de angular.
* Las importaciones de los componentes.
* Las importaciones de los DTO’s.
* Las importaciones de los servicios.
* Las importaciones de las utilidades.
* Las importaciones de las constantes.

**Documentar código**

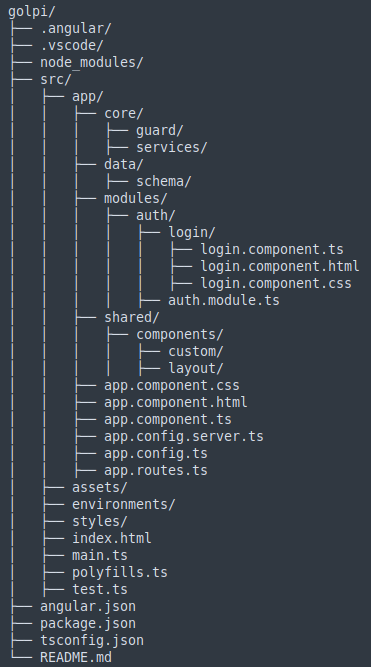
* Para comentar las importaciones de los archivos se usará //



* Para comentar los métodos complejos se utilizara /\*\* @method \*/



**Estructuración de las carpetas:**



1. **app**/

* **core**/ Generalmente se usa para servicios y utilidades centrales que se comparten en toda la aplicación.

1. **guard**/ Aquí se colocan los archivos relacionados con la protección de rutas, como los "route guards" que permiten o bloquean el acceso a ciertas partes de la aplicación según las condiciones (por ejemplo, si el usuario está autenticado).
2. **services**/ Contiene servicios globales, como autenticación, manejo de errores, o cualquier lógica compartida entre diferentes partes de la app.

* **data**/ Se utiliza para manejar la estructura de datos y modelos compartidos en la aplicación.

1. **schema**/ Define los esquemas o interfaces que modelan la estructura de los datos utilizados en la aplicación. Por ejemplo, aquí podrías definir la estructura de los datos de un usuario, un calendario, o una cancha.

* **modules**/ Este directorio organiza los diferentes módulos funcionales de la aplicación. La modularización ayuda a organizar el código por características y a manejar mejor la escalabilidad.

1. **auth**/ Módulo encargado de la autenticación.

* **shared**/ Esta carpeta contiene componentes y servicios que se comparten entre diferentes partes de la aplicación.

1. **components**/ Contiene componentes reutilizables en varias secciones de la aplicación.
2. **custom**/ Componentes personalizados específicos.
3. **layout**/ Componentes que se relacionan con el diseño o estructura visual, como menús, headers o footers.
4. **assets**/ Esta carpeta contiene recursos estáticos como imágenes, fuentes, íconos, etc., que pueden ser utilizados dentro de la aplicación.
5. **environments**/ Contiene configuraciones separadas por ambientes, como desarrollo (development) y producción (production). Permite gestionar variables de entorno específicas para cada caso.

**Versión de Angular:**

* Angular 17.3.11

**Estilos:**

* Css

**Arquitectura:**

* Arquitectura Hexagonal