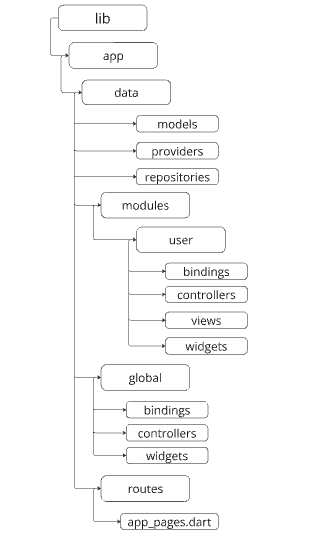
**Distribución de carpetas Flutter**

Para el proyecto Flutter o móvil de la aplicación picao se sugiere realizar la utilización de una distribución de carpetas modular con el fin de que el código se manejable y tenga una alta escalabilidad a futuro, adicional se utilizara GetX como manejador de estados, gestor de dependencias y gestor de rutas.

**Diagrama.**



**Descripción.**

1. **app/data/**
   1. **models**: Aquí se definen las clases que representan los modelos de datos que utiliza la aplicación.
   2. **providers**: Se colocan las clases responsables de hacer peticiones HTTP, manejar conexiones a bases de datos u otras fuentes de datos.
   3. **repositories**: Aquí se encapsulan las operaciones que involucran el manejo de datos, como la lógica de negocio y la interacción entre modelos y proveedores.
2. **app/modules/**
   1. **bindings**: Las clases de bindings enlazan los controladores con las vistas. Se usa para la inyección de dependencias al iniciar una vista.
   2. **controllers**: Controladores para manejar la lógica de negocio y la gestión del estado.
   3. **views**: Vistas de la UI que utilizan los controladores para mostrar datos.
   4. **widgets**: Componentes UI específicos de cada módulo, reutilizables dentro del módulo.
3. **app/routes/**
   1. **app\_pages.dart**: Aquí se definen las rutas de la aplicación usando GetX, gestionando las transiciones entre vistas y la navegación.
4. **app/global**/:
   1. **bindings**: Binding generales que se utilizan en toda la aplicación.
   2. **controllers**: Controladores globales (por ejemplo, controladores de sesión o de configuración).
   3. **themes**: Configuración de temas y estilos globales.

**Estándar de nombramiento para Flutter**

A continuación, se define el estándar de nombramiento para el proyecto Flutter, App móvil. El idioma utilizado para el proyecto será ingles esto para el nombramiento de archivos, clases, métodos y variables, para los comentarios se usará el idioma español.

Con excepción también en las palabras más comunes en inglés, por ejemplo: Login, Generic, Pages. Repository etc.

**Directorios**

* Nombre en minúscula.
* Si el nombre es compuesto por dos o más palabras, se separa por guion bajo.

**Archivos**

* Nombre en minúscula.
* Si el nombre es compuesto por dos o más palabras, permanece junto y en minúscula.
* Para la mayoría de archivos se añade un identificador de acuerdo al directorio que pertenece. El identificador es precedido de guion bajo junto al identificador elegido. Ejemplo: nombrearchivo\_identificador.dart. Para la mayoría de casos el nombre del identificador se relaciona con el nombre del directorio.

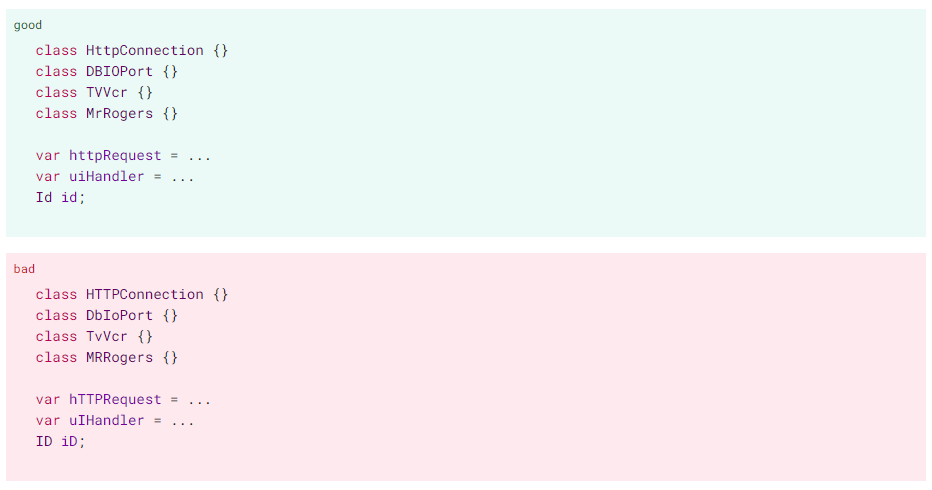
Todo lo demás aquí:

<https://dart.dev/guides/language/effective-dart>

<https://dart.dev/tools/linter-rules#directives_ordering>

**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.

[](https://dart.dev/guides/language/effective-dart/style#do-name-libraries-and-source-files-using-lowercase_with_underscores)

*Effective Dart: Style*

**Métodos**

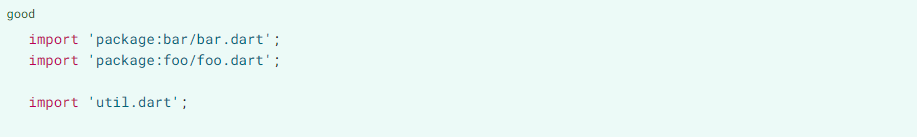
* Inicia con minúscula. Utilizando lowerCamelCase.
* Su nombre inicia con un verbo en infinitivo.

**Variables, Constantes y valores Enum**

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

**Orden de las directivas**

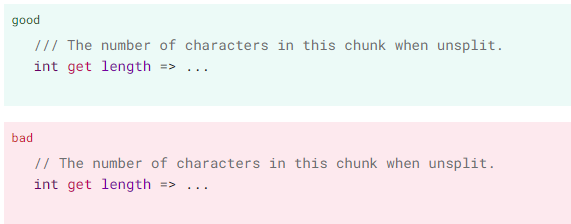
* Las importaciones propias de dart deben ir antes que otras importaciones.
* Las importaciones de los paquetes van antes de las importaciones relativas.



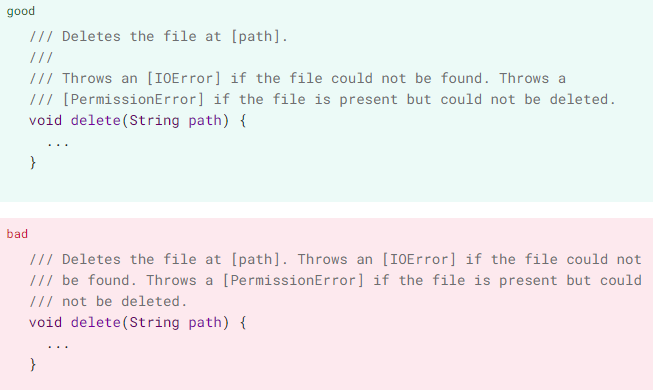
* Las exportaciones se especifican en una sección aparte al final de todas las importaciones.
* Separar en secciones las directivas de Widgets, Service, Repository, Pages.

**Documentar código**

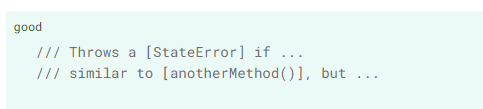
* Para comentar solo se usa ///.

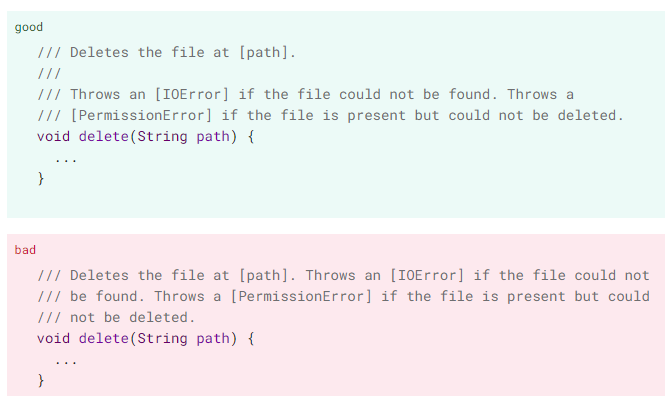


* Separar la primera oración de un comentario sobre un documento en su propio párrafo.

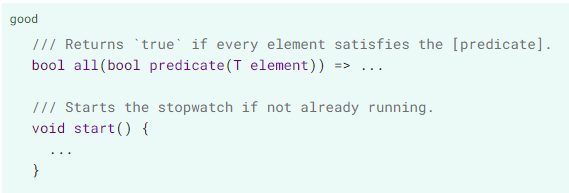


* Usar corchetes para variables, throws o métodos.

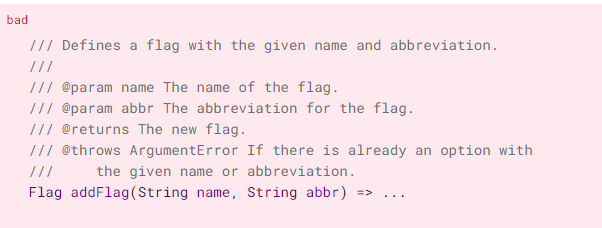




* En los comentarios para definir los valores que se retorna, se pone entre backticks(`valor`).



* Use la prosa para explicar parámetros, valores de retorno y excepciones.Otros lenguajes usan etiquetas y secciones detalladas para describir cuáles son los parámetros y los retornos de un método.



La convención en Dart es integrar eso en la descripción del método y resaltar los parámetros usando corchetes.

