APLICANDO CLEAN ARCHITECTURE

CICLO 4b: Desarrollo de Aplicaciones Móviles

En cada proyecto desarrollado durante este ciclo trabajaremos clean architecture como patrón de diseño que nos facilitará el proceso de construcción de software.

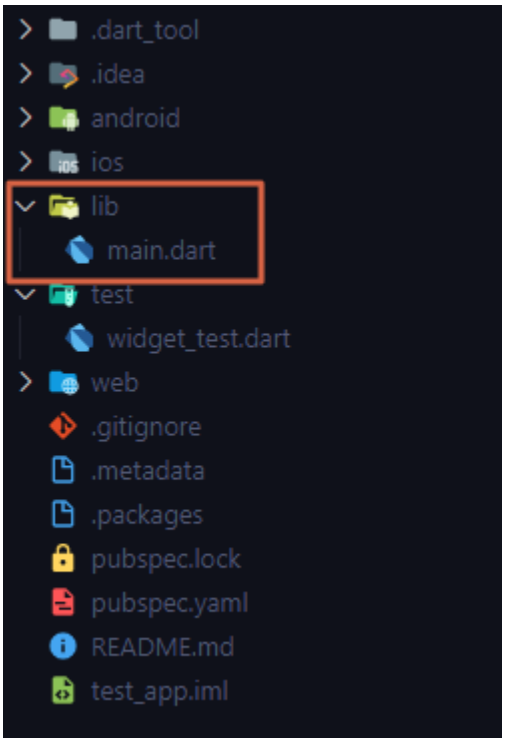
¿Qué es clean architecture?

Clean Architecture es un nombre popularizado por Robert Cecil Martin, conocido como “Uncle Bob” con la base de estructura del código en capas o “layers” que solo se comunican con sus capas contiguas. Es decir, todas las capas tienen diferentes enfoques, pero comparten la idea de que cada nivel tiene su responsabilidad y se comunica únicamente con sus niveles inmediatamente contiguos.

Una capa es un conjunto de clases, paquetes o ficheros que tienen unas responsabilidades relacionadas dentro del sistema. Estas capas están organizadas de forma jerárquica unas encima de otras y las dependencias siempre van hacia abajo. Es decir, que una capa dependerá solamente de las capas inferiores, pero nunca de las superiores.

¿Cómo implementar clear architecture en nuestros proyectos?

1. Al crear un proyecto Flutter con “Flutter Create” podemos obtener un árbol de carpetas como el siguiente:

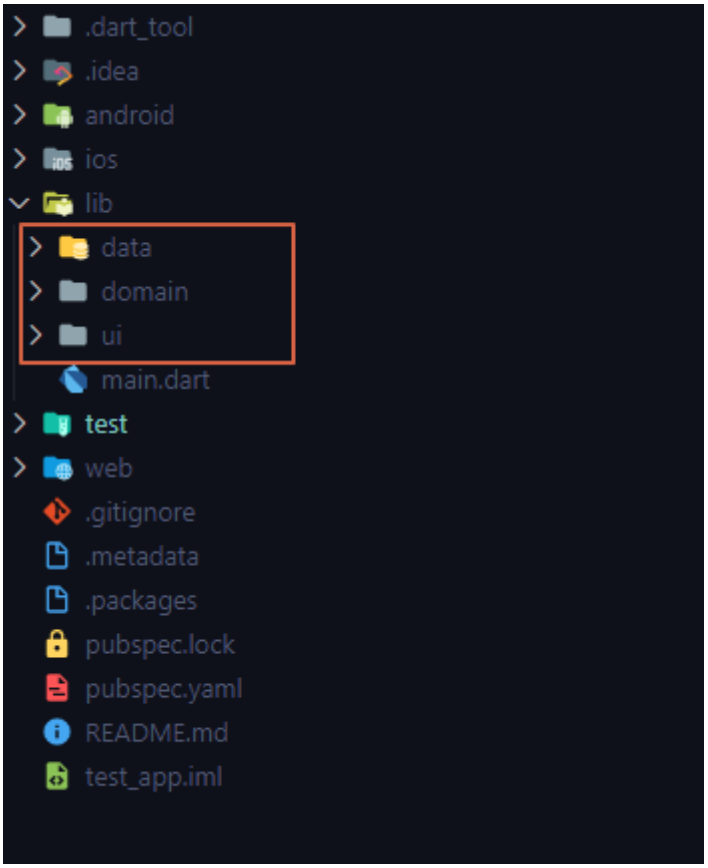


2. Como podemos observar en la carpeta /lib únicamente es creado el archivo main.dart de nuestra App. La carpeta lib es la encargada de almacenar todos los ficheros que harán que la aplicación funcione y se debe dividir en 3 capas (o

subcarpetas) con cada parte que la componen. Por lo tanto, se deben crear las siguientes carpetas:

data
domain
ui

Resultado:



¿Qué significa cada capa creada?

Data: Capa encargada de comunicarse con las dependencias externas que necesita nuestra app para obtener los datos. Por ejemplo, la implementación concreta de los repositories con llamadas HTTP, Firebase…

Domain: Capa que engloba toda la lógica de negocio de nuestra app, donde se encuentra el código que debe ser agnóstico de cualquier otra parte de nuestro software. En este caso, es puro código Dart.

UI: Capa encargada de la representación de los datos en un dispositivo o plataforma. En nuestro caso, será todo el código de Flutter correspondiente a nuestros Widgets, pages, navegación…

3. Cada capa estará compuesta por otras subcapas que permitirán dividir los diferentes módulos de la aplicación. Se debe crear un árbol como el siguiente:

data

repositories *//diferentes implementaciones de cada interfaz de los Repositories*
services *//diferentes implementaciones de cada servicio*

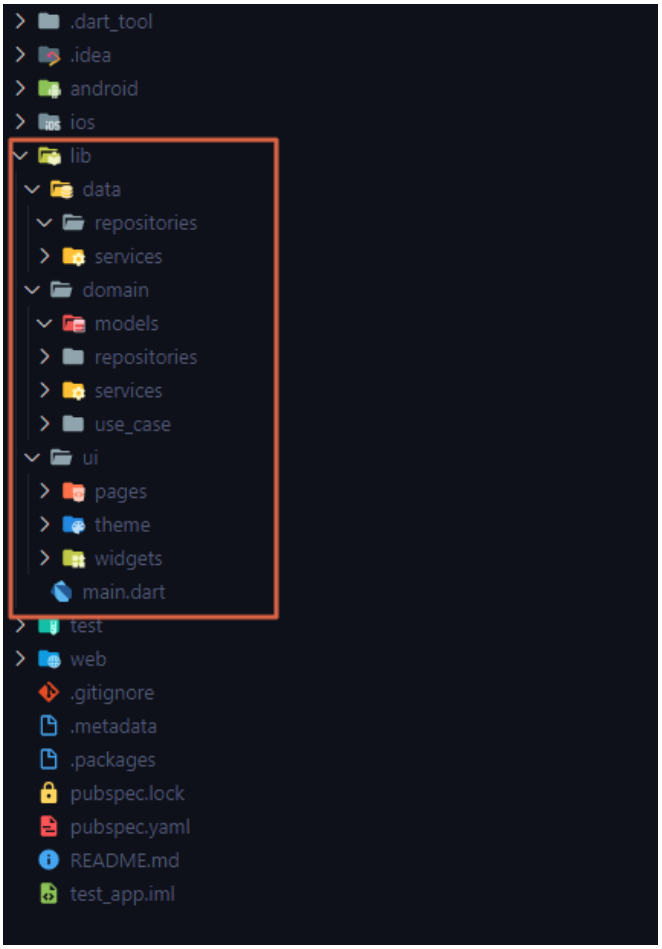
domain

models *// clases que representan nuestras entidades de la capa de negocio*
repositories *// componentes para obtener información de servicios externos*
services *// interacción con servicios externos de nuestra app*
use_case *// lógica de negocio de cada componente y su definición de evento/estado*

ui

pages *//contiene los ficheros que representan nuestras pantallas*
themes *//contiene la definición de los temas de la aplicación*
widgets *//carpeta que contiene los componentes que se usan a través de toda la aplicación*

Resultado:



Referencias:
https://alfredobs97.medium.com/clean-architecture-en-flutter-ee028a6379a5