Arquitectura Limpia

Tras años de desarrollo siguiendo la arquitectura multicapa, la modernización de las aplicaciones y las desventajas de dicha arquitectura provocaron el surgimiento de nuevas arquitecturas que intentan ser la solución a los problemas de la multicapa y además intentan traer nuevas ventajas y facilidades al mundo del desarrollo. Algunas de ellas son:

- La arquitectura hexagonal
- La arquitectura de cebolla
- DCI de James Coplien y Trygve Reenskaug
- BCE de Ivar Jacobson

En 2012, Robert C. Martin, también conocido como Uncle Bob, unifica estas arquitecturas con una nueva llamada "Clean Architecture", manteniendo los propiedades y objetivos de las ya mencionadas. Para lograr ese objetivo su nueva arquitectura se basa en una estructura de capas, donde la capa central es la de las *entidades* que, junto con la siguiente capa mas concéntrica, la de *casos de usos*, forman la capa de dominio. Esta capa es seguida por la capa de *adaptadores* y finalmente por la de f*rameworks y drivers.*

Para que la arquitectura funcione y tenga las propiedades deseadas, las capas deben seguir una regla de dependencia. Esta regla dicta que las dependencias del código solo pueden apuntar hacia capas interiores. Nada de un circulo interior puede saber nada de un circulo exterior.

ESTRUCTURA

Clean Architecture se divide en las siguientes capas:

DOMINIO

El dominio es el núcleo de la aplicación y por ello, es recomendable que este se encuentre totalmente aislado de cualquier tipo de dependencia. El dominio consta de dos subcapas, las entidades y los casos de uso.

ENTIDADES

Están formadas por la lógica de negocio empresarial, que es aquella lógica que existiría en una compañía independientemente de que exista o no una aplicación para automatizar los procesos de dicha empresa. Las entidades pueden ser compartidas entre varias aplicaciones de la compañía.

CASOS DE USO

Están formados por la lógica de la aplicación, es decir, existe debido a la creación de una aplicación para la automatización de procesos. Son individuales e inherentes a la aplicación a la que pertecen.

ADAPTADORES

En esta capa se encuentran mecanismos que convierten la información obtenida de una capa a otro lenguaje que sea interpretable por la capa de destino (bases de datos, servidores, etc.)

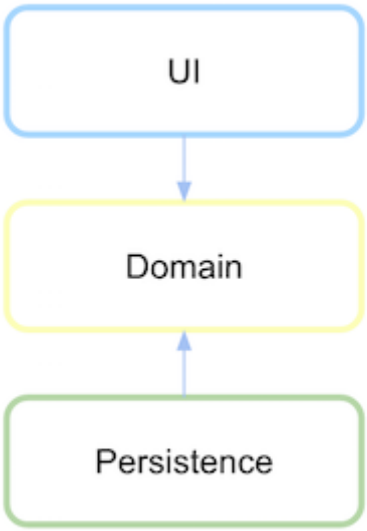
DETALLES DE IMPLEMENTACIÓN

Es todo aquello que no sea lógica, las partes menos fundamentales de una aplicación. Están formados por los frameworks, librerías, bases de datos, …

Si seguimos correctamente la implementación de Clean Architecture, esta capa debería ser fácilmente sustituibles.

VENTAJAS

- Baja acoplación a frameworks, librerías, etc. Estas herramientas son intercambiables fácilmente sin afectar a ninguna de las otras capas.
- La interfaz de usuario se puede sustituir en cualquier momento. Si implementamos la interfaz de usuario utilizando frameworks la podemos cambiar en cualquier momento.
- Independencia de la base de datos. El cambio de base de datos no afecta a las reglas de negocio.
- El dominio gana importancia y no depende de nadie. Tomando el principio SOLID de inversión de dependencia conseguimos que el dominio no dependa de ninguna capa.



- Las reglas de negocio son más fáciles de testear. Eliminando las dependencias se consigue que podamos testear solo las reglas de negocio sin usar ninguna de las otras capas.