Capitulo 2.

"Arquitectura CLEAN".

Arquitectura limpia, esta tuvo su origen en un articulo escrito por Robert C. Martin, también Conocido como Uncle Bob. el nombre del articulo fue titulado como "The Clean Architecture".

Una arquitectura limpia es aquella la cual pretende conseguir una estructura modular bien separadas, de facil lectura, limpieza del código y testabilidad.

Si se toma como base dicho articulo creado por Robert C. Martin los sistemas que han sido creado por la Arquitectura limpia dube de tener ciertas Caracteristicas las Cuales Son:

- **Independientes del Framework Utilizado:** esto nos dice que las librerias a emplear en el momento del desarrollo o construcción del sistema no deben de ser Condicionarnos más bien dichas librerias deben de Tomase como una herramienta más.

- **Testeables:** La cual se nos dice que independientemente de la interfaz grafica a usar ó modelo, base de datos. ó Peticiones a una API, esta dube de ser testeable en cualquier entorno indiscutiblemente.

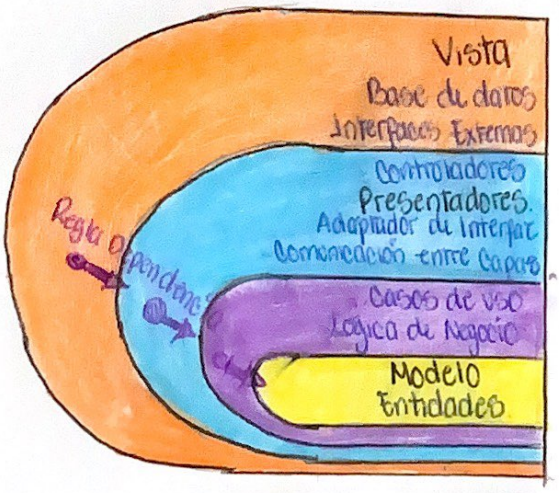
- **Independientes de la interfaz grafica**: En este se dube emplear Patrones que nos permitan Cambiar fácilmente la interfaz gráfica, en otras palabras debemos de evitar el acoplar el funcionamiento de la vista con el modelo y para lograr esto se emplean patrones como MVP, MVVM, ó el clasico MVC.

- **Independientes de los orígenes de datos**: Se podrán sustituir fácilmente el origen de los datos, sin importar si este esta disponible en una base de datos local, ficheros ó una base de datos relacional ó no relacional.

- **Independientes de factores externos**: Se debe aislar las reglas del negocio de tal forma que estas no se involucren a otros factores.

Hay que aclarar que no todo lo mencionado anteriormente describe a totalidad la arquitectura limpia, el articulo escrito por Uncle Bob, nos presenta un diagrama en el que se expone la idea de la separación de responsabilidades y las diferentes capas de nuestro sistema.

Arquitectura CLEAN.



Nuestro modelo ó logica de negocio está en el interior, en el corazón del diagrama; hacia el exterior tenemos la capa de control, transporte o comunicación entre capas y finalmente la capa correspondiente a la vista, base de datos, interfaces externas.

Además de las diferentes capas, la regla de dependencia nos indica que existe un sentido para atravesarlas y es que las dependencias estas deben apuntar desde el extremo hacia el interior, esto quiere decir que si todo se aplica o se emplea correctamente las capas interiores no deben de ser relacionadas con las capas externas.

Nuestra lógica de negocio ó casos de uso tienen que ser independientes, no deben utilizar ningún elementos software de cualquier otra capa.

Algunos de los elementos el cual se muestra en el diagrama son:

- Entidades: Son objetos de negocio de nuestra aplicación, estas deben poseer propedades y métodos capaces de ser reutilizables y que esta posean una estructura que varié con poca frecuencia.

- **Casos de Uso**: Ó más bien conocidos como interactores, estas se encargan de implementar las reglas de negocio de la aplicación

- Adaptadores de interfaz: son adaptadores encargados de transformar los datos desde el formato mas conveniente para los casos de uso y entidades al formato que mejor convenga a Base de datos ó Interfaz de usuario.

- **Frameworks y Drivers**: En esta capa se encuentran plataformas externas, interfaz de usuario y Base de datos. Es la capa más externa, que debe comunicarse hacia las capas interiores.