Capítulo 3: principios SOLID

Es uno de los acrónimos más famosos de la programación. Se compone de 5 principios de la Programación orientada a objetos:

Principio de responsabilidad única

Dentro de una clase debe de haber sólo una funcionalidad. Si hay más fun. separarlas en mas clases.

Principio de ser abierto o cerrado

Abierto a extensiones
Cerrado a modificaciones

Principio de sustitución de liskov

Toda clase q extienda la funcionalidad de una clase padre.base debe implementar la funcionalidad de la clase base sin alterrar el funcionamiento.

Principio de segregación de interfaz

Evitar tener métodos no implementados de interfaces q implementamos

Principio de inversión de dependencia.

Utilizar interfaces ó abstracciones en el dominio de nuestro sistema. Elementos q cambian con poca frecuencia.

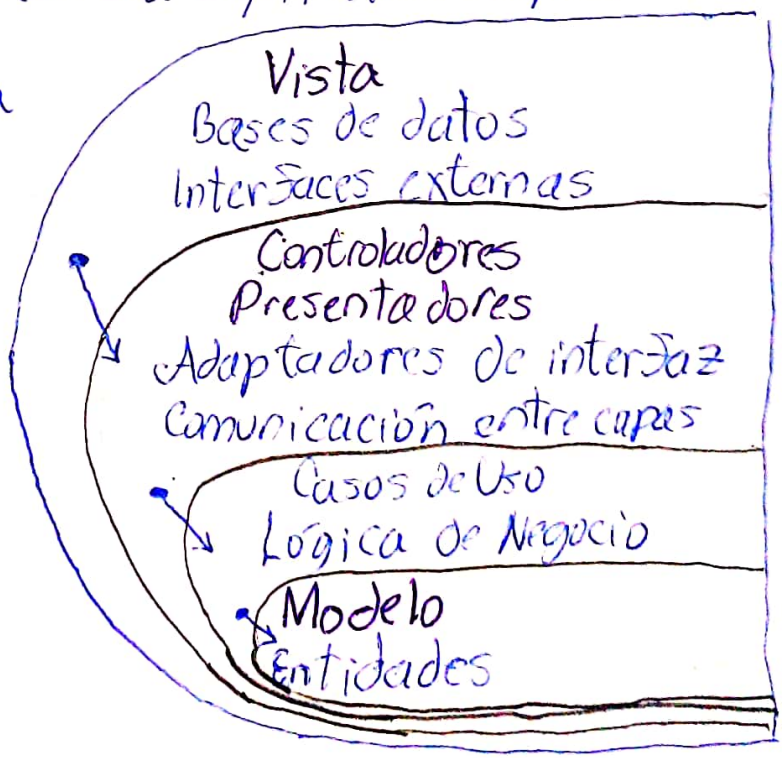
Capitulo 2: Arquitectura Clean

Una arquitectura limpia es aquella que pretende conseguir unas estructuras modulares bien separadas, de fácil lectura, limpieza del código y testeabilidad. Basándose en el artículo de Uncle Bob, los sistemas construidos con una arquitectura limpia han de ser:

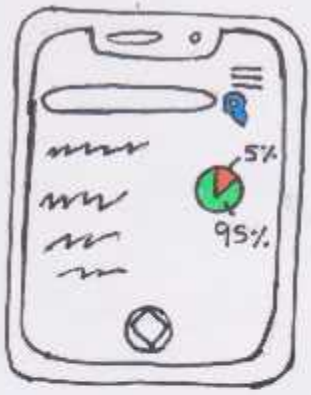
1. Independientes del Framework utilizado.
2. Testeables.
3. Independientes de la interfaz gráfica.
4. Independientes de los datos.
5. Independientes de los factores externos.

A continuación se expone la idea de la separación de responsabilidades y las diferentes capas de nuestros sistemas construidos con una arquitectura limpia.

↑ Regla dependencia



CAPITULO 4: PATRONES DE DISEÑO.



El uso de patrones de diseño facilita la solución de problemas comunes que se encuentran en el desarrollo de software. Los patrones de diseño tratan de resolver los problemas relacionados con la interacción entre interfaz de usuario, lógica de negocio y todo lo relacionado con los datos.

Dos de los patrones más utilizados para la resolución de este tipo de problemas son los patrones MVC (Modelo Vista Controlador) y MVP (Modelo Vista Presentador).

MVC – Modelo Vista Controlador.

Es uno de los más conocidos por la comunidad de desarrolladores de software. Plantea el uso de 3 capas para separar la interfaz de usuario de los datos y la lógica de negocio. Estas capas son:

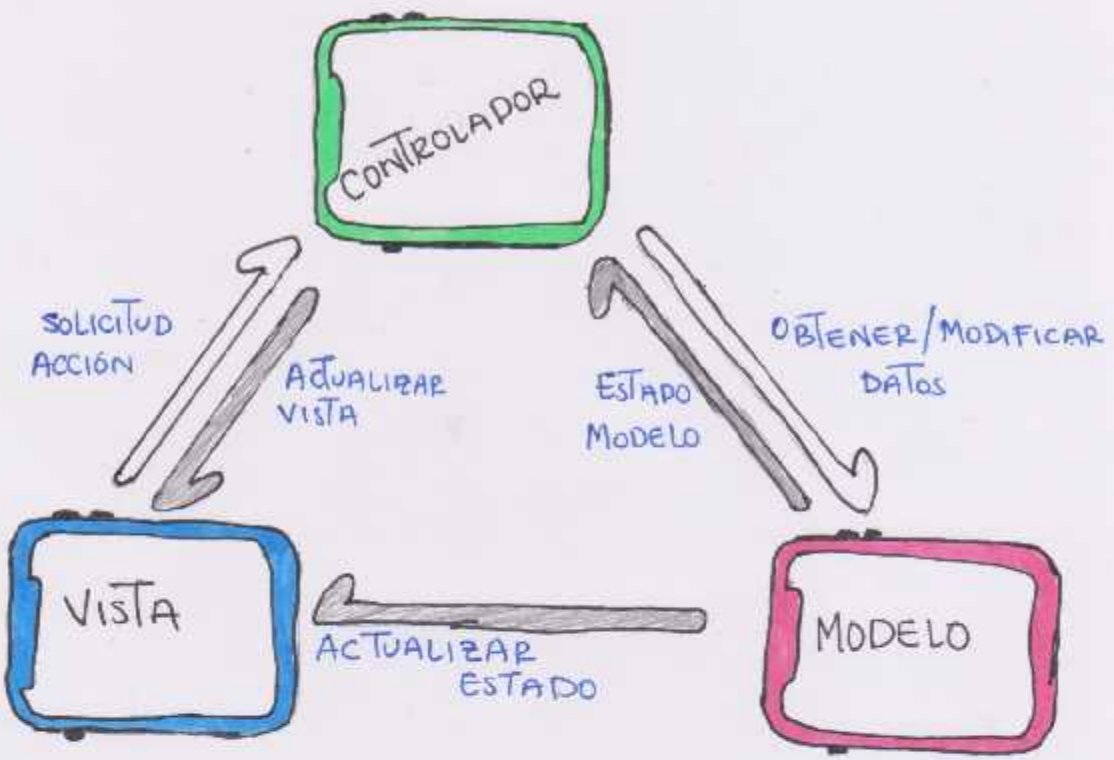
- MODELO:
Esta capa contiene el conjunto de clases que definen la estructura de datos con los que vamos a trabajar en el sistema.

• VISTA:

Esta capa contiene la interfaz de usuario de nuestra aplicación. Maneja la interacción del usuario con la interfaz de usuario para enviar peticiones al controlador.

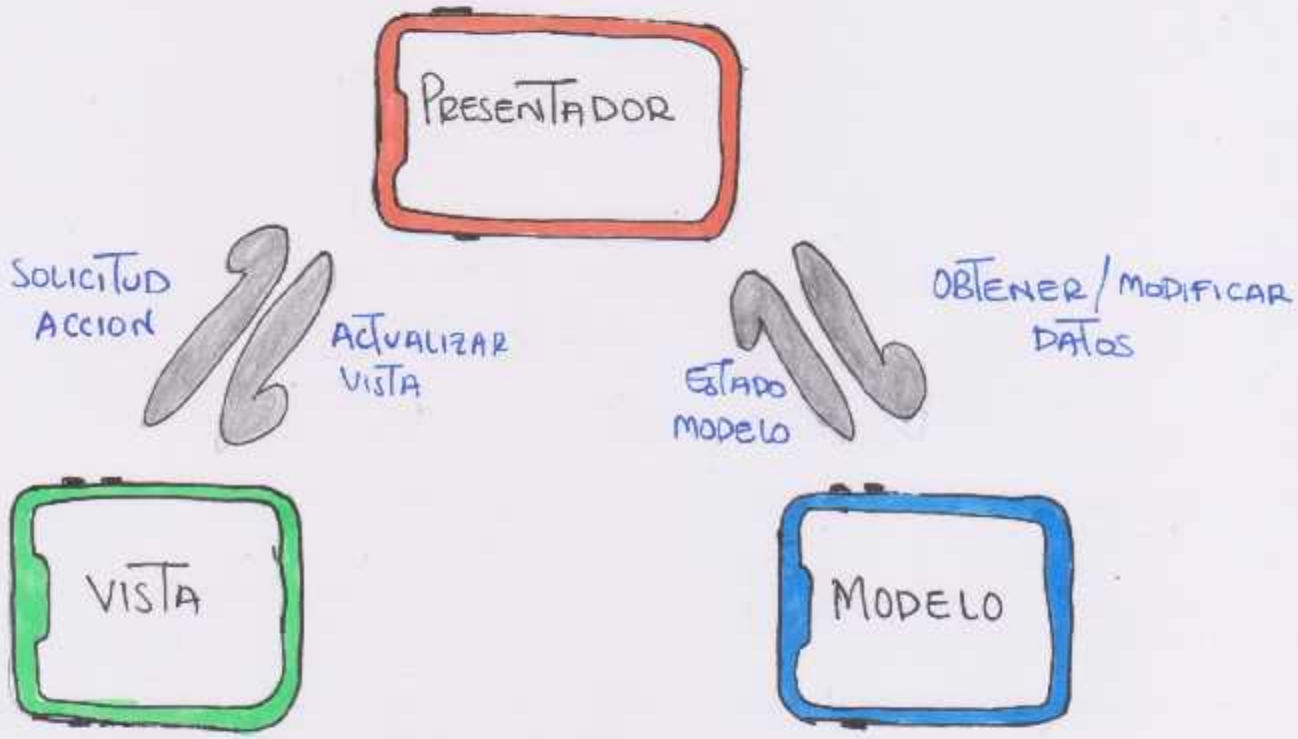
• CONTROLADOR:

Esta capa es la intermediaria entre la Vista y el Modelo. Es capaz de responder a eventos, capturándolos por la interacción de usuarios en la interfaz, para procesar la petición y solicitar o modificarlos en el modelo.



MVP- MODELO VISTA PRESENTADOR.

Este patrón se deriva del MVC y nos permite separar aún más la vista de la lógica de negocio y de los datos. Toda la lógica de la presentación de la interfaz reside en el Presentador, de forma que este da el formato a los datos.



- MODELO:

Es la capa encargada de gestionar los datos; su principal responsabilidad es la persistencia y almacenamiento de datos.

- VISTA:

La vista no es un Activity ovn Fragment, simplemente es una interfaz de comportamiento de lo que podemos realizar con la vista.

- PRESENTADOR:

Es la capa que actúa como intermediaria entre el modelo y la vista.