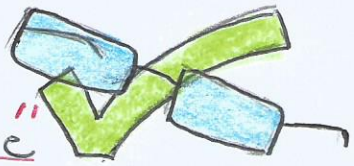
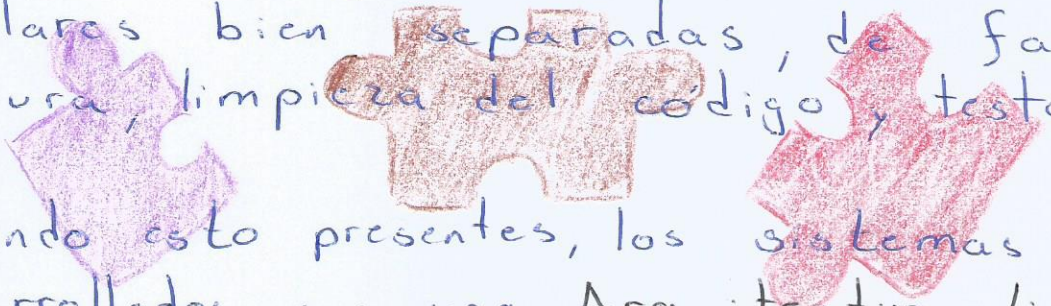

Arquitectura
Clean

La denominación "Arquitectura Limpia" tiene su origen en un artículo escrito por Robert C. Martin, también conocido como Uncle Bob, titulado como: "The Clean Architecture"



Toda arquitectura limpia es aquella que pretende conseguir unas estructuras modulares bien separadas, de fácil lectura, limpieza del código y testabilidad.



Teniendo esto presentes, los sistemas desarrollados con una Arquitectura Limpia deben ser:

Las librerías empleadas en la construcción del sistema no deben condicionarnos, han de ser una herramienta más por utilizar.

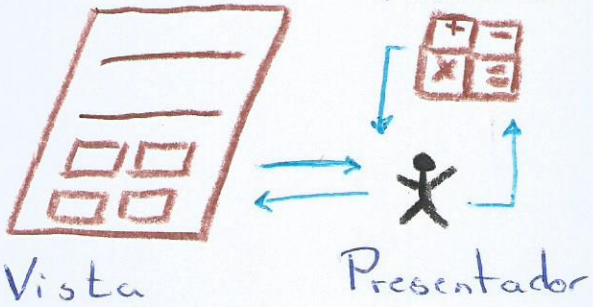
class Testeables (modelo.Model):

La lógica de negocio de nuestra aplicación ha de ser testeable indistintamente de la interfaz gráfica, modelo, base de datos o peticiones a una API empleadas.

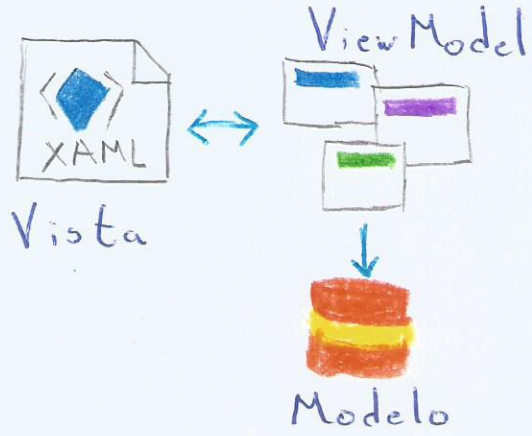
#Independientes_de_la_interfaz_gráfica {

Depende del patron que sea indicado usar, que nos permita cambiar fácilmente la interfaz gráfica, siendo estos:

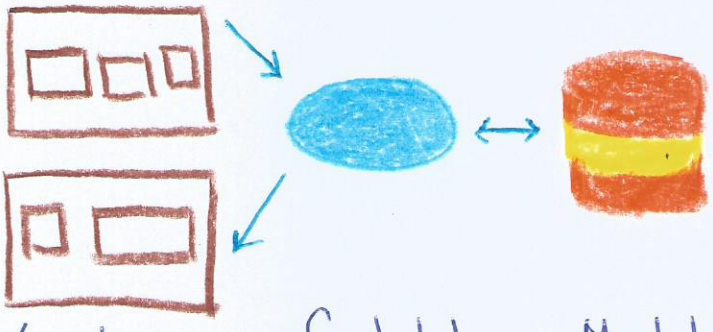
■ MVP ■



■ MVVM ■



Clásico MVC

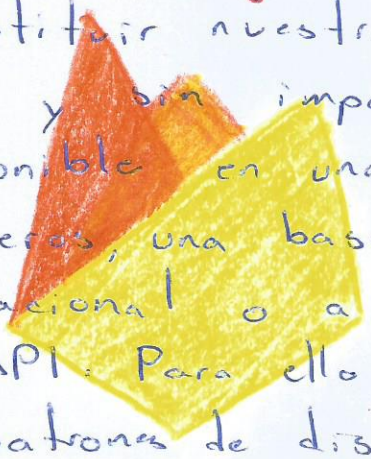


}

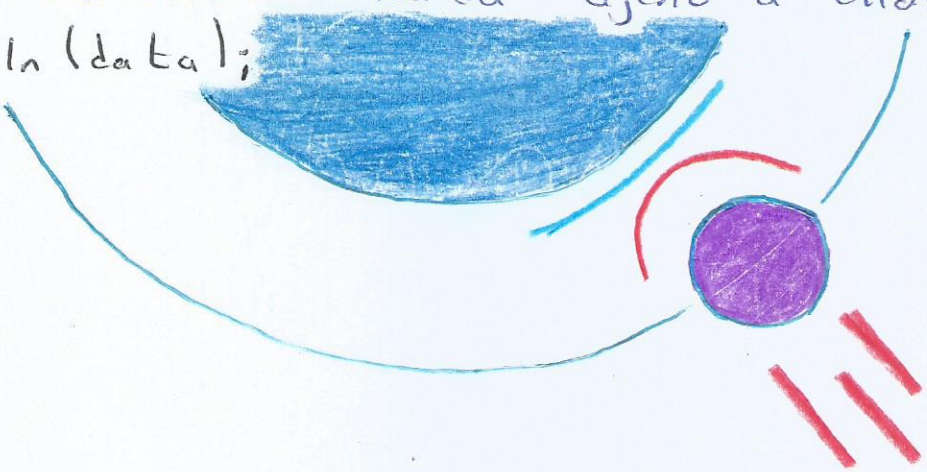
Independientes_de_los_orígenes_de_datos : [

1. Podremos sustituir nuestro origen de datos
2. fácilmente y sin importarnos si este
3. está disponible en una base de datos
4. local, ficheros, una base de datos relacional
5. o no relacional o a través de peticiones
6. a una API. Para ello se puede hacer
7. uso de patrones de diseño como el patrón
8. Repositorio.

]



```
public void Independientes_de_factores_externos (String data){
  data= "Debemos aislar nuestras reglas de
  negocios en un mundo a parte, de tal forma
  que no conozcan nada ajeno a ellas";
println(data);
}
```



Uncle Bob expone la idea de la separación de responsabilidades y las diferentes capas de nuestros sistemas construidos con una arquitectura limpia.

Los elementos de estas capas:

- Entidades. Los objetos y métodos deben ser reutilizables y poseer una estructura que varía con poca frecuencia.
- Casos de uso. Implementan las reglas de negocio, orquestando el flujo de datos desde y hacia las entidades.
- Adaptadores de interfaz. Encargados de transformar los datos desde el formato más conveniente para los Casos de uso y Entidades al formato que mejor convenga a Base de datos o Interfaz de usuario.
- Frameworks y Drivers. Son las plataformas externas, Interfaz de usuario y Base de Datos; la encargada de comunicarse hacia las capas interiores.

<!-- Además de estas, la regla de dependencia nos indica que existe un sentido para atravesarlas y es que las dependencias deben apuntar desde el extremo de la figura hacia su corazón -->

/*
Con esta separación, si comparamos dos capas, la capa interior no debe conocer nada de las capas exteriores a ella.
*/