

# Arquitectura CLEAN

Es muy importante entender la arquitectura CLEAN aunque se desida no ocuparla ya que puede ayudarte a comprender de mejor manera la programación orientada a objetos.

Permite tener separadas unidades de un solo código de manera organizada de tal forma que se hace mas practico entender, modificar y testear dicho código realizandolo de forma o manera adecuada

## Las capas de clean architecture

- 1) **Presentación:** Es la capa que interactua con la interzas del usuario (activities, fragments, views)
- 2) **Caso de uso:** o interactores, son las acciones que el usuario puede desencadenar (Hacer click en un boton o navegar a una página o pantalla)
- 3) **Domínio:** o logica de negocio, son las reglas o instrucciones para dar vida a la funcionalidad para que realice un trabajo en específico



4) Datos: Es una definición abstracta de los diferentes puentes de datos y la forma en la que se debe de utilizar

5) Framework: lo que hace esta capa a grandes rasgos es que encapsula la interacción con el framework

Interacción entre capas

Presentación  $\xrightarrow{\quad}$  Caso de uso  $\xrightarrow{\quad}$  Dominio  $\xrightarrow{\quad}$  Datos  $\xrightarrow{\quad}$  Framework

Todo siempre debe pasar por el dominio que es lo que queremos ver así es la parte con mayor importancia en nuestra aplicación.



## Principios SOLID

Es aplicable a POO y ayudara a escribir software de calidad con los siguientes

- ▶ Principio de responsabilidad unica: un modulo tiene una unica razon para cambiar
- ▶ Principio de Abierto/Cerrado: una entidad debe estar abierta a expansion pero cerrada a modificaciones
- ▶ Principio de Sustitucion de Liskov: trata sobre todo de coherencia que se realiza en la aplicacion
- ▶ Principio de segregacion de interfaces: Ninguna clase debe depender de metodos que no utiliza
- ▶ Principio de Inversion de dependencias: tener bien visible que dependencia ocupa cada modulo y no complicar los tests



## Patrones de Diseño

Son soluciones generales y reutilizables a problemas comunes en el desarrollo de software

Más utilizados en el desarrollo android son patrones creacionales

- ▶ Singleton
- ▶ Builder
- ▶ Factory

Estructurales

- ▶ Adapter
- ▶ Facade

Comportamiento

- ▶ Observer
- ▶ Iterator



```
public class HomeActivity extends AppCompatActivity {
```

```
    private static HomeActivity instanciaUnica;
```

```
    private synchronized static void createInstance() {
```

```
        if (instanciaUnica == null) {
```

```
            instanciaUnica = new HomeActivity();
```

```
        }
```

```
    }
```

```
    public static HomeActivity getInstance() {
```

```
        createInstance();
```

```
        return instanciaUnica;
```

```
    }
```

```
    public void printName() {
```

```
        System.out.println("Marco Antonio Hernandez");
```

```
    }
```



HomeActivity.java × EcuacionActivity.java × activity\_ecuacion.xml

```
package sv.com.edu.ldb.dsm.activities;
```

```
import ...
```

```
public class EcuacionActivity extends AppCompatActivity {
```

```
    HomeActivity user = HomeActivity.getInstance();
```



```
private void updateRoots(String[] roots){  
    lblX1.setText(Html.fromHtml( source: "X<  
    lblX2.setText(Html.fromHtml( source: "X<  
    user.printName();  
}
```



2/21/11 After code cache collection, code=94KB, data

I/System.out: Marco Antonio Hernandez

D/EGl emulation: eg1MakeCurrent: 0x00705000, user 0