

Diseño del software y modularidad



Francisco Coya Abajo
Rubén Caño Domínguez

Principios de un buen diseño de Software

- + Simplicidad.
- + Claridad.
- + Abstracción para manejar la complejidad.
- + Anticiparse al cambio.
- + ¿Cuándo diseñar?
- + Definir errores fuera de existencia
- + Modularidad

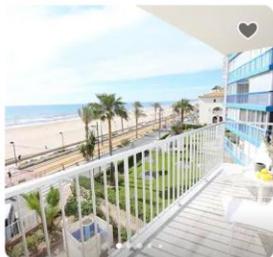


1. Simplicidad

- + Evitar implementar funcionalidades que no son necesarias.
- + Código sencillo es más fácil de probar.
- + Software fácil de entender y mantenible.
- + Interfaces de usuario fáciles de usar y “minimalistas”.
- + Evitar sobre-diseñar.



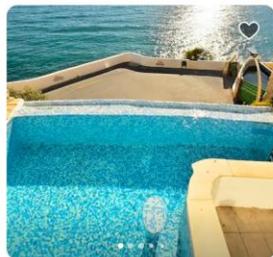
El Campello, España ★ 4,75
 Playa de Mutxavista
 10-16 abr · Anfitrión particular
 201 € noche



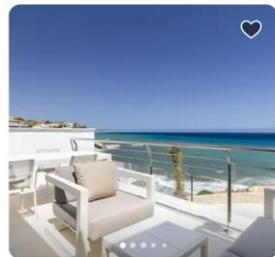
El Campello, España ★ 5,0
 Mutxavista Beach
 31 mar - 5 abr · Anfitrión profesional
 138 € noche



Villajoyosa, España ★ 4,95
 Playa Centro La Vila Joiosa
 2-7 mar · Anfitrión particular
 67 € noche



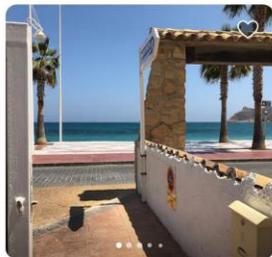
Benidorm, España ★ 4,73
 Finestrat
 12-17 mar · Anfitrión profesional
 276 € noche



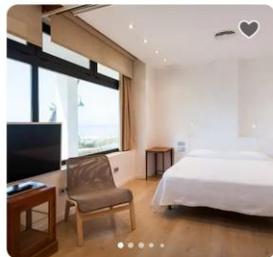
El Campello, España ★ 4,63
 Anfitrión profesional
 3-8 mar
 101 € noche



Calpe, España ★ 4,84
 Cala La Manzanera
 19-24 mar · Anfitrión particular
 149 € noche



Altea, España ★ 4,76
 Lp
 11-16 mar · Anfitrión particular
 74 € noche



Altea, España ★ 4,84
 Cala de la Barra Grande
 6-12 mar · Anfitrión particular
 86 € noche

Fuente: Airbnb



Visitantes

- Ventajas de introducir el código postal:
 - Informarle de si hacemos repartos en su código postal.
 - Adaptarle su surtido de productos local.
 - Tener opción a compra al finalizar su visita.

Registrarse como Cliente

- A través de esta opción accederá al formulario para convertirse en cliente del Supermercado on-line no teniendo que volver a escribir sus datos. Además podrá confeccionar sus listas personales.

Cliente registrado

Condiciones generales de compra

Código Postal:

Usuario:
 Contraseña:

Fuente: Mercadona

2. Claridad

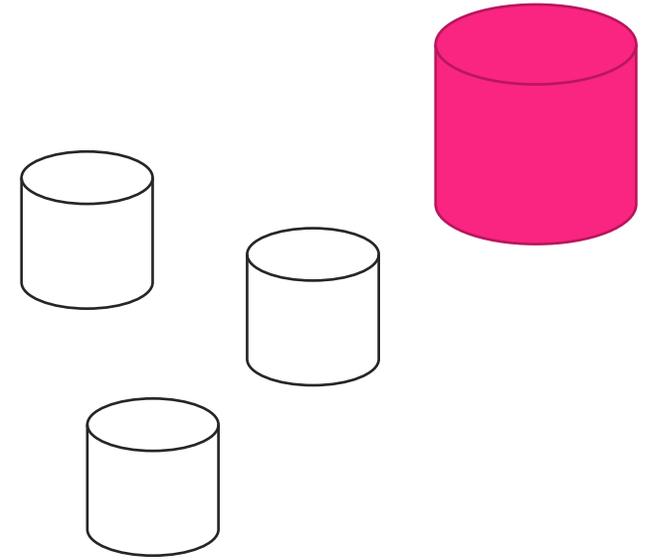
- + Mantener el código lo más simple y legible posible.
Fácil de entender, modificar y mantener.
- + Nombrado de variables, funciones y clases coherente y descriptivo.
- + Buena organización.
- + *Comunicación.

3. Abstracción para manejar la complejidad

- + Ocultar detalles de la implementación.
- + Si un diseño parece complicado, enfocarlo desde otro punto de vista.
- + Conocer el dominio del problema y su contexto.
- + Programar para interfaces, no para implementaciones.
Abstraer los detalles de la implementación a una interfaz.

4. Anticiparse al cambio

- + El software está en continuo cambio.
- + Modularidad + Flexibilidad.
- + Conocer lo que no sabes que no sabes.
- + Analizar puntos de mejora actuales.
- + Refactorización de código.
- + Minimizar el acoplamiento. Produce cambios en todas las dependencias.



“ This is a problem for future Homer ”



Fuente: 20th Century Fox Television

DEUDA
TÉCNICA

5. ¿Cuándo diseñar?

+ 3 posibles métodos:

- Modelo cascada: Diseño final antes de implementar.
- Extremo contrario: Implementar sin diseñar.
 - Diseños pobres y sistemas difíciles de entender y ampliar.
- Diseño iterativo:
 - Diseño inicial, implementación, rediseño, implementación...

+ 3 iteraciones hasta un buen diseño.

6. Definir errores fuera de existencia

- + Considerar que un caso especial es un error cuando no lo es
- + Ejemplo: Método que elimina un archivo:
 - Es lógico pensar en lanzar una excepción si el archivo no existe
 - Mejor finalizar el método, el archivo ya está eliminado
- + Evitar usar el principio en exceso
 - Ignorar errores importantes es una mala interpretación

7. Modularidad para reducir la complejidad

- + Forma sencilla de pensar en algo complicado.
- + Encapsular la funcionalidad reduce la complejidad.
- + Dos tipos de módulos:
 - Módulos profundos.
 - Módulos superficiales.
- + Ejemplo recolectores de basura. Son ejemplo de módulo que ni siquiera tiene interfaz.

Preguntas

Arquitectura del Software. Curso 2022/23

Francisco Coya Abajo · UO257239
Rubén Caño Domínguez · UO284647