Patrones de Diseño

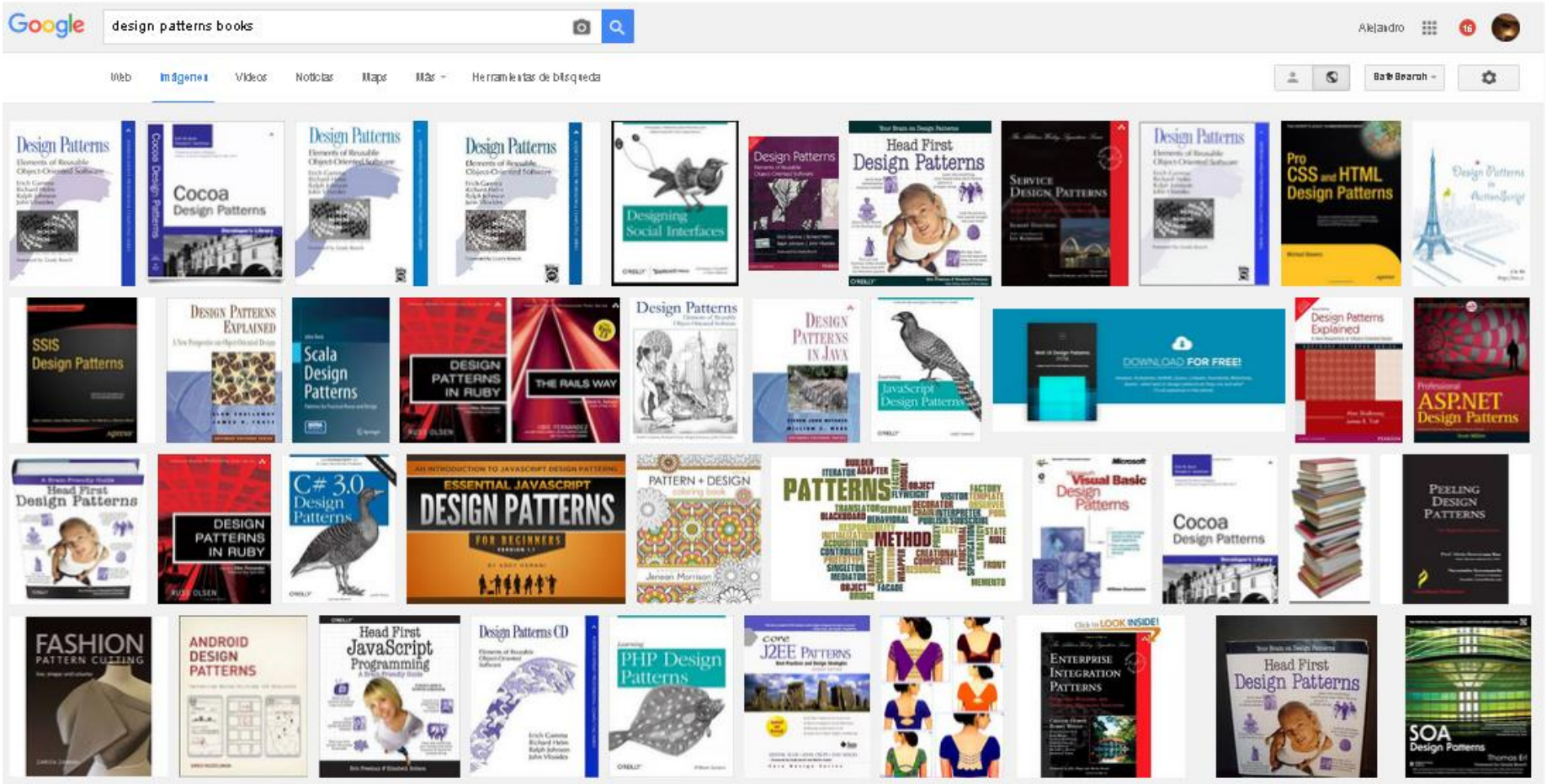
Alejandro Mancilla
@alxmancilla

Agenda del curso

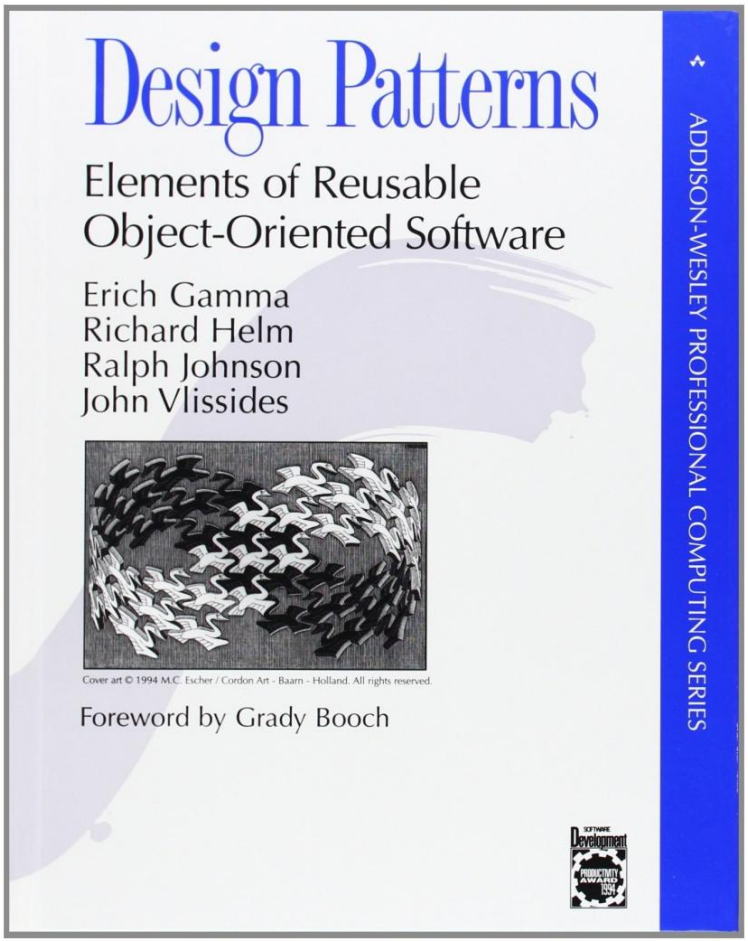
- Patrones ¿qué son y para qué sirven?
- Categorías de Patrones
- Clasificación de Patrones de Diseño
- Patrones de Creación - ¿Cómo creo un objeto?
- Patrones de Estructura - ¿Cómo pueden trabajar una clase y un objeto?
- Patrones de Comportamiento - ¿Cómo interactúo entre objetos?
- Patrones de Arquitectura - ¿Cómo diseño una aplicación?

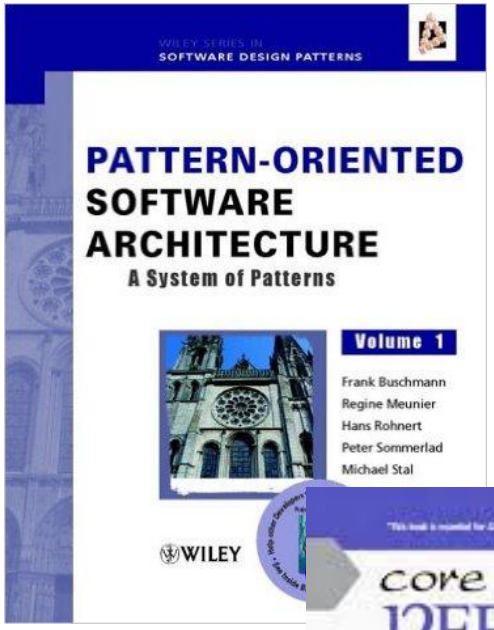
Patrones ¿qué son y para qué sirven?

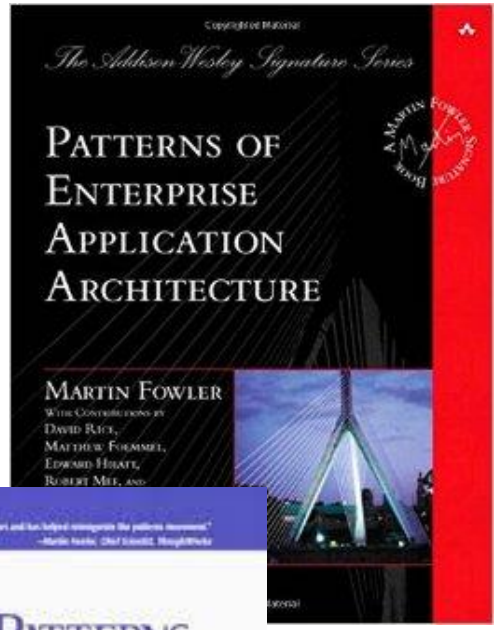
Búsqueda: “design patterns books”

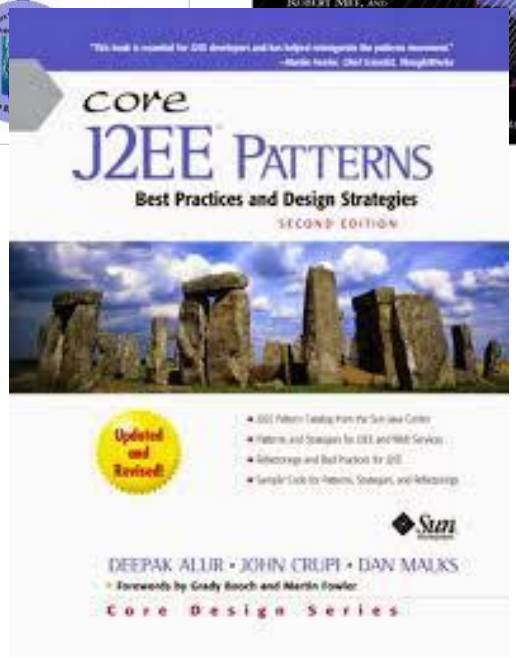


Libros más relevantes









¿Qué es un patrón?

- En 1977, en el libro *A Pattern Language,* Christopher Alexander dice:
  - "Cada patrón describe un problema que ocurre una y otra vez en nuestro medio ambiente y, a continuación describe el núcleo de la solución a ese problema, de tal manera que se puede utilizar esta solución un millón de veces, sin tener que hacerlo de la misma manera dos veces”

¿Qué es un patrón?

- En Ingeniería de Software, describe una *solución* reutilizable a un *problema* común en un *contexto* dado.
- Se enfoca en la **solución**, no en el problema
- Identifica responsabilidades e interacciones entre los participantes
- "Un patrón es principalmente una forma de masticar consejos sobre un tema" – Martin Fowler

¿Qué NO es un patrón de diseño?

- No es un paradigma de programación
- No es un *silver bullet*.
- No es una solución inefectiva y riesgosa
- No resuelve un problema específico
- No depende del lenguaje de implementación

¿Para qué sirven los patrones de diseño?

- Vocabulario y entendimiento común para el diseño de software
- Alternativas de diseño para que sea flexible y reutilizable
- Construir arquitecturas de software complejas y heterogéneas
- Favorece la vida y mantenibilidad de una aplicación
- Incrementa tu experiencia profesional

Patrón: Layers

- Problema:
  - Diseñar un sistema cuya característica dominante sea una mezcla de problemas de alto nivel y de bajo nivel
  - El flujo de comunicación consta de peticiones que van del nivel más superior al nivel inferior, y las respuestas en sentido inverso
- Contexto:
  - Una aplicación grande que requiere descomposición
- Solución:
  - Estructurar aplicaciones que puedan ser descompuestas en grupos de subtareas, en las que cada grupo de subtareas está en un nivel particular de abstracción

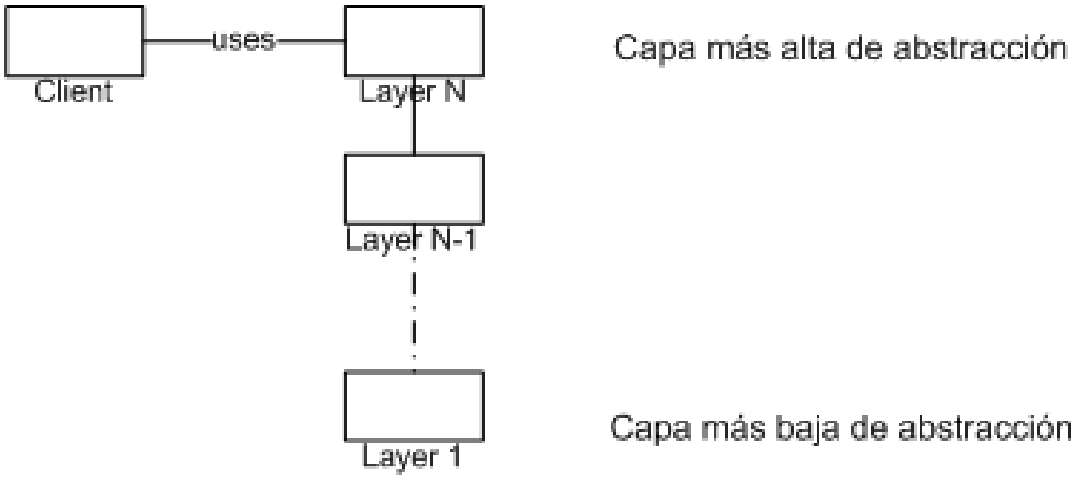
Estructura del patrón Layers

- Estructura:
  - Una capa individual se puede describir con la siguiente tabla

Clase: • Layer J	**Colaborador**: • Layer J-1
Responsabilidad: • Provee servicios usados por Layer J+1 • Delega subtareas a Layer J-1	

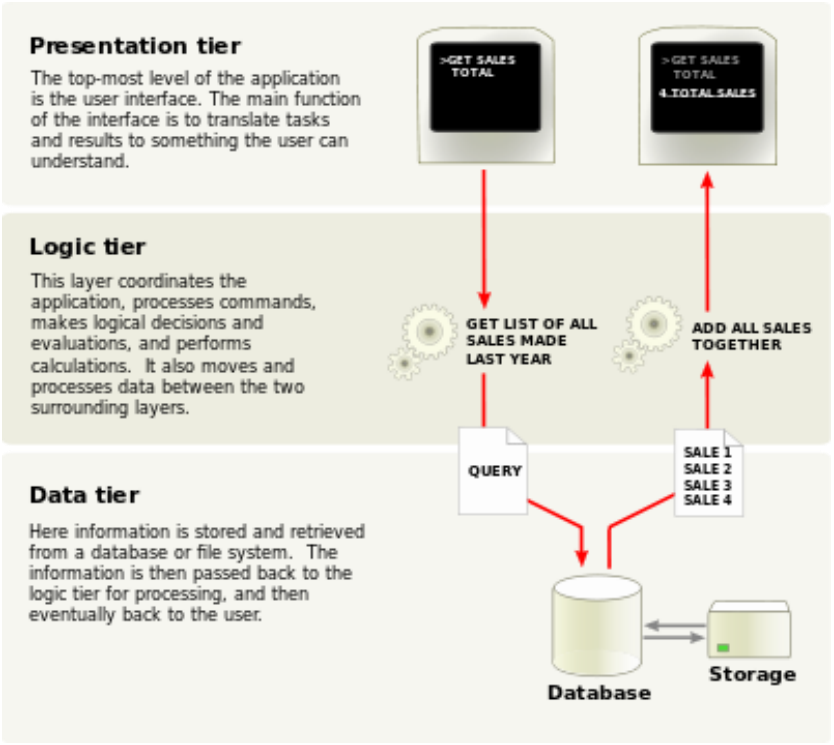
Estructura del patrón Layers

- Estructura:
  - Los servicios de capa Layer J son sólo usados por capa Layer J+1, no existen más dependencias directas entre capas.



Ejemplo del patrón Layers

- La arquitectura de tres capas es un patrón de arquitectura de software cliente-servidor en el que la interfaz de usuario (PresentationLayer), la lógica de negocio funcional (LogicLayer), el acceso y almacenamiento de datos(DataLayer) en están desarrollados y mantenidos como módulos independientes, con mucha frecuencia en plataformas separadas.



Patrón Layers en Python

```
class DataLayer():
    def l1Service(self):
        print("Executing %s Service" % self.name);

class DomainLayer():
    def l2Service(self):
        print("Starting %s Service" % self.name)
        self.lowerLayer.l1Service()
        print("Finishing %s Service" % self.name)

class PresentationLayer():
    def l3Service(self):
        print("Starting %s Service" % self.name)
        self.lowerLayer.l2Service()
        print("Finishing %s Service" % self.name)
```

Categorías y clasificación de Patrones de Diseño

Síntomas de un mal diseño

- Rigidez
- Fragilidad
- Inmovilidad
- Viscosidad

Principios SOLID de DOO

- The Single Responsibility Principle
- The Open Closed Principle
- The Liskov Substitution Principle
- The Interface Segregation Principle
- The Dependency Inversion Principle

Categorías de Patrones (POSA)

- Patrones de Arquitectura
  - Ejemplos*: Layers, MVC, EDA, etc.*
- Patrones de Diseño
  - Ejemplos*: Factory Method, Façade*, *Strategy, Observer, etc*
- Idioms/Modismos
  - Ejemplos*: Manejo de memoria, uso del lenguaje, convenciones de nombrado, etc*

Clasificación de Patrones de Diseño (GoF)

- Propósito
  - Refleja lo que hace un patrón
  - Patrones de Creación , Estructura y Comportamiento

- Alcance
  - Especifica si el patrón se aplica principalmente a clases u objetos
  - Patrones de Clase y de Objeto

Clasificación de Patrones de Diseño (GoF)

		Propósito		
		Creación	Estructura	Comportamiento
Alcance	Clase	Factory Method	Adapter	Interpreter Template Method
	Objeto	Abstract Factory Builder Prototype Singleton	Adapter Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

Patrones de Diseño

- Factory Method
  Crea una instancia de varias clases derivadas
- Abstract Factory
  Crea una instancia de varias familias de clases
- Builder
  Separa la construcción de un objeto de su representación
- Prototype
  A fully initialized instance to be copied or cloned
- Singleton
  A class of which only a single instance can exist

Patrón: Builder

- Problema:
  - The more complex an application is the complexity of classes and objects used increases.
- Contexto:
  - An application might need a mechanism for building complex objects that is independent from the ones that make up the object.
- Solución:
  - Defines an instance for creating an object but letting subclasses decide which class to instantiate

Participantes del patrón: Builder

- **Builder** class
  - specifies an abstract interface for creating parts of a Product object.
- **ConcreteBuilder**
  - constructs and puts together parts of the product by implementing the Builder interface.
- **Director** class
  - constructs the complex object using the Builder interface.
- **Product**
  - represents the complex object that is being built.

Participantes del patrón: Builder