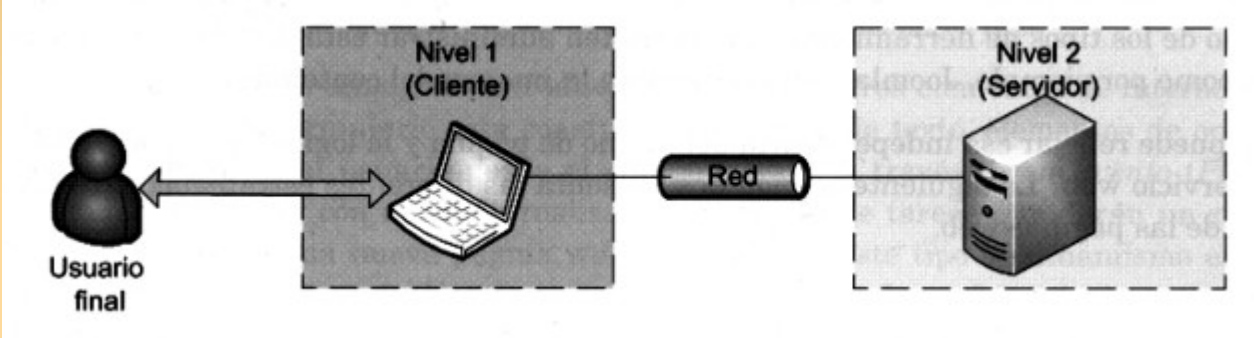
Arquitecturas para aplicaciones web: el patrón MVC y otros

(UD 4)

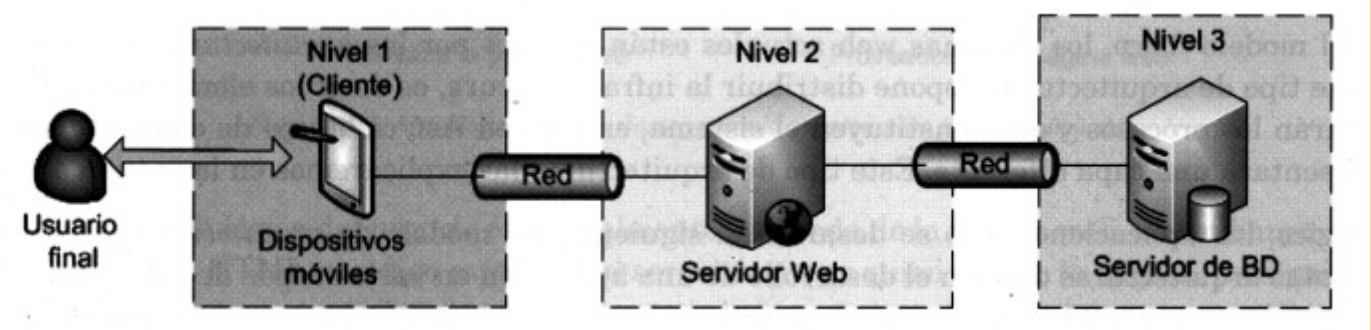
1. La arquitectura física

Arquitecturas FÍSICAS multinivel (*multitier*)

Arquitectura en 2 niveles:

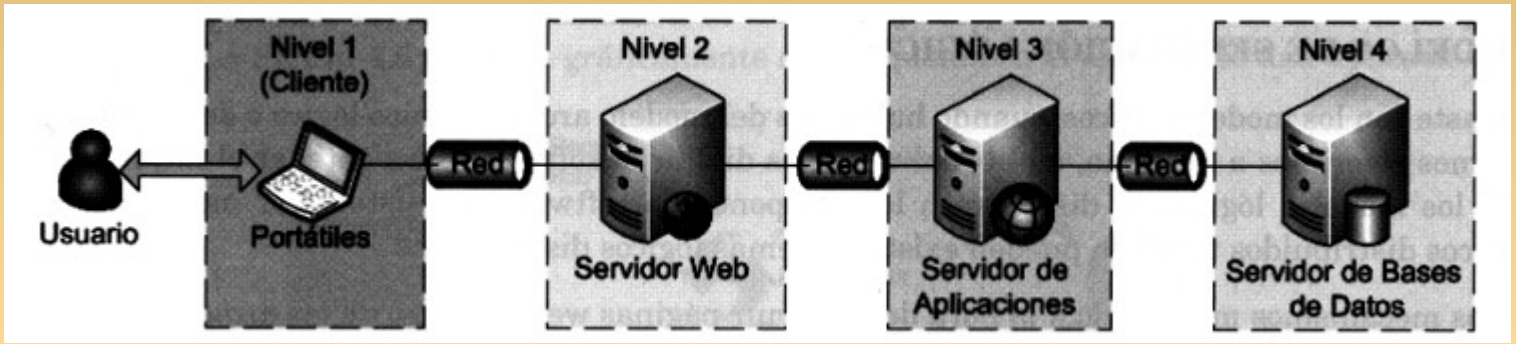


Arquitectura en 3 niveles:

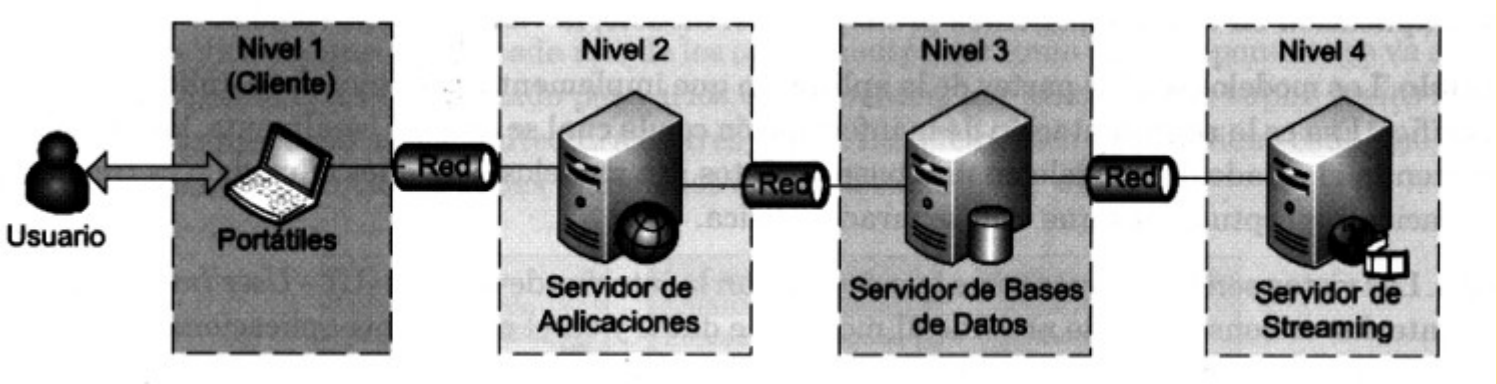


1. La arquitectura física

Un ejemplo de arquitectura en 4 niveles:



Otro ejemplo de arquitectura en 4 niveles:



2. La arquitectura lógica

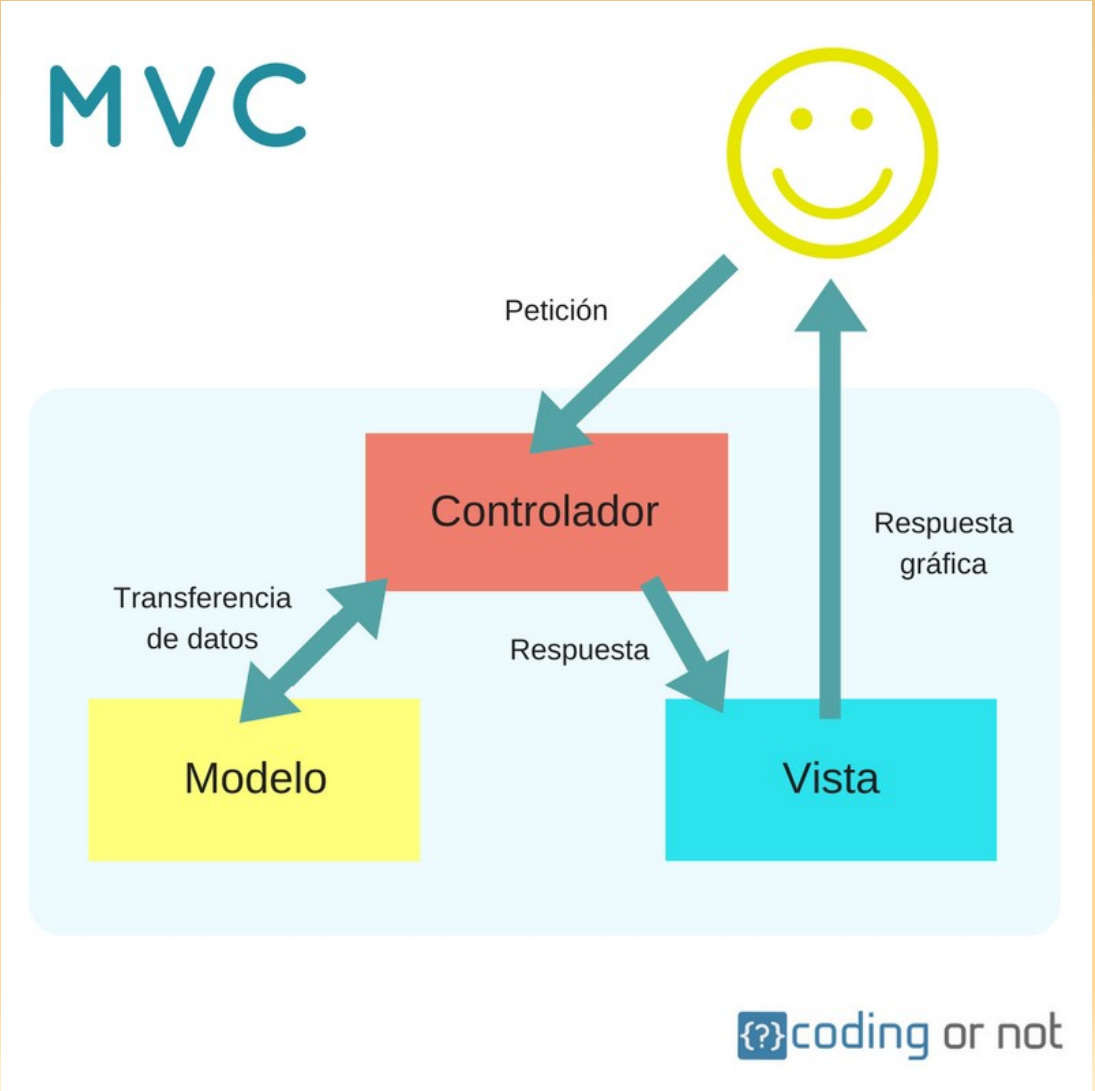
Arquitecturas LÓGICAS multicapa (*multilayer*)

Ventajas:

- Desarrollos paralelos en cada capa
- Aplicaciones robustas (encapsulamiento)
- Matenimiento más sencillo
- Más flexibilidad para añadir módulos
- Más escalabilidad para aumentar rendimiento

2. La arquitectura lógica

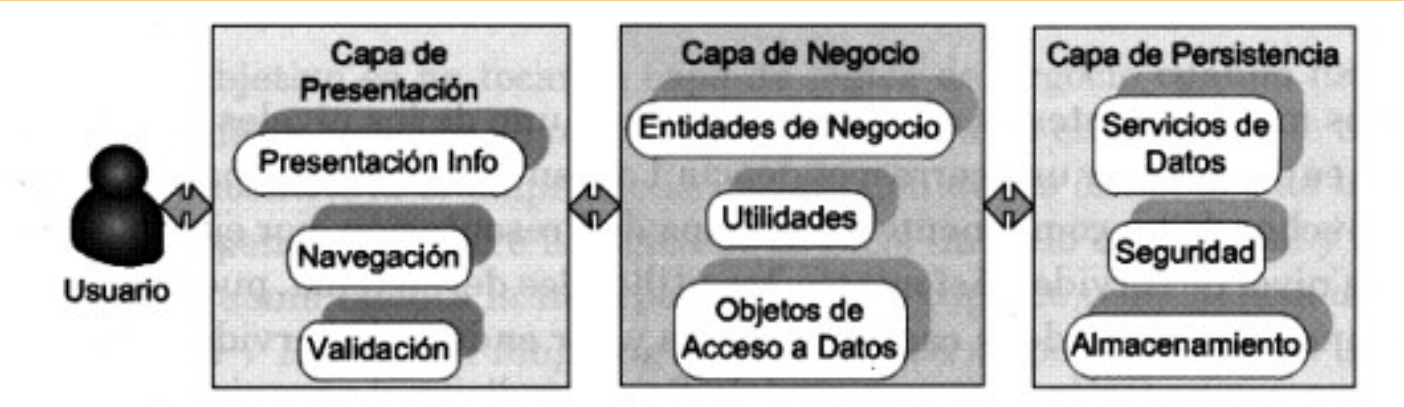
Esquema Modelo-Vista-Controlador (MVC)



2. La arquitectura lógica

Arquitectura en 3 capas

Es una generalización del patrón MVC.

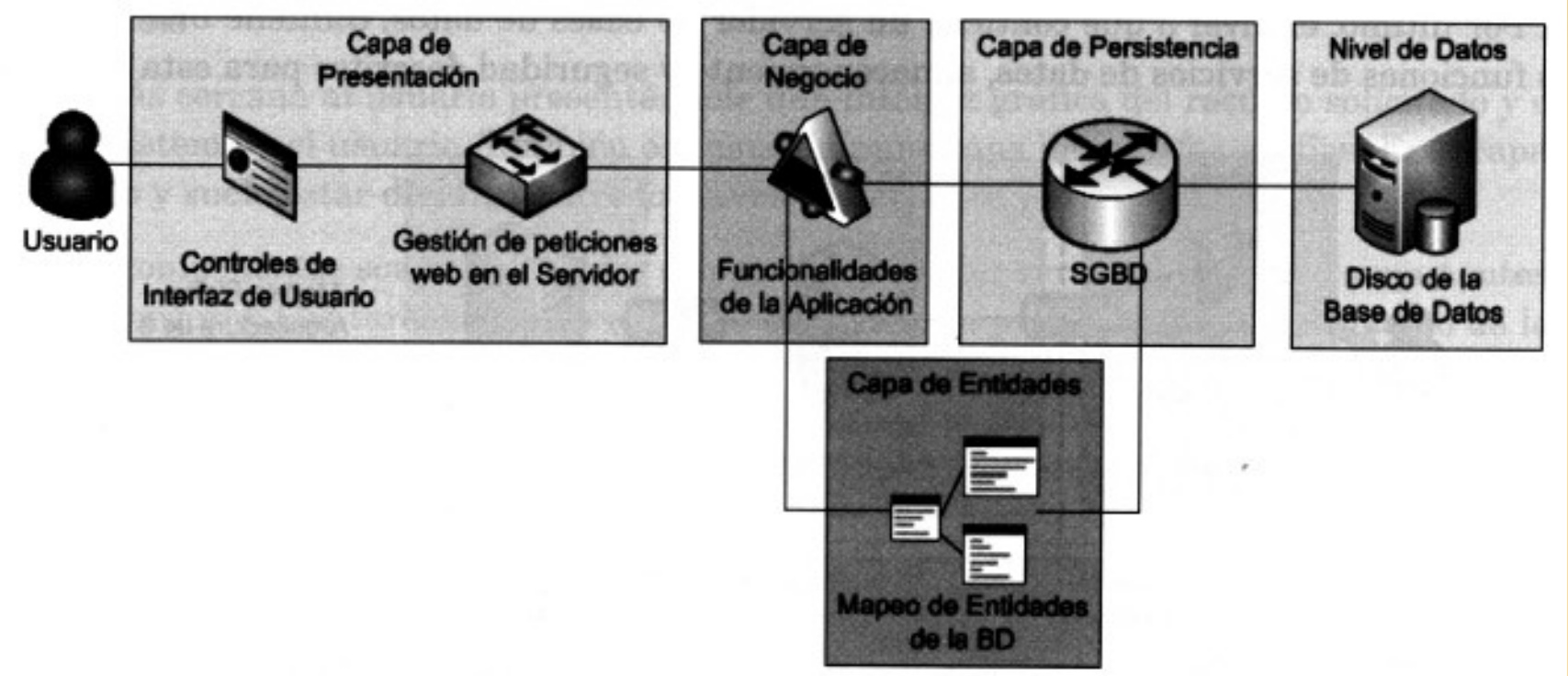


Hay que decidir qué componentes hay, sus funciones, y en qué capa y qué nivel físico encaja cada uno.

2. La arquitectura lógica

Arquitecturas multicapa

Modelo de capas mejorado con una capa de entidades



2. La arquitectura lógica

Aspectos estructurales

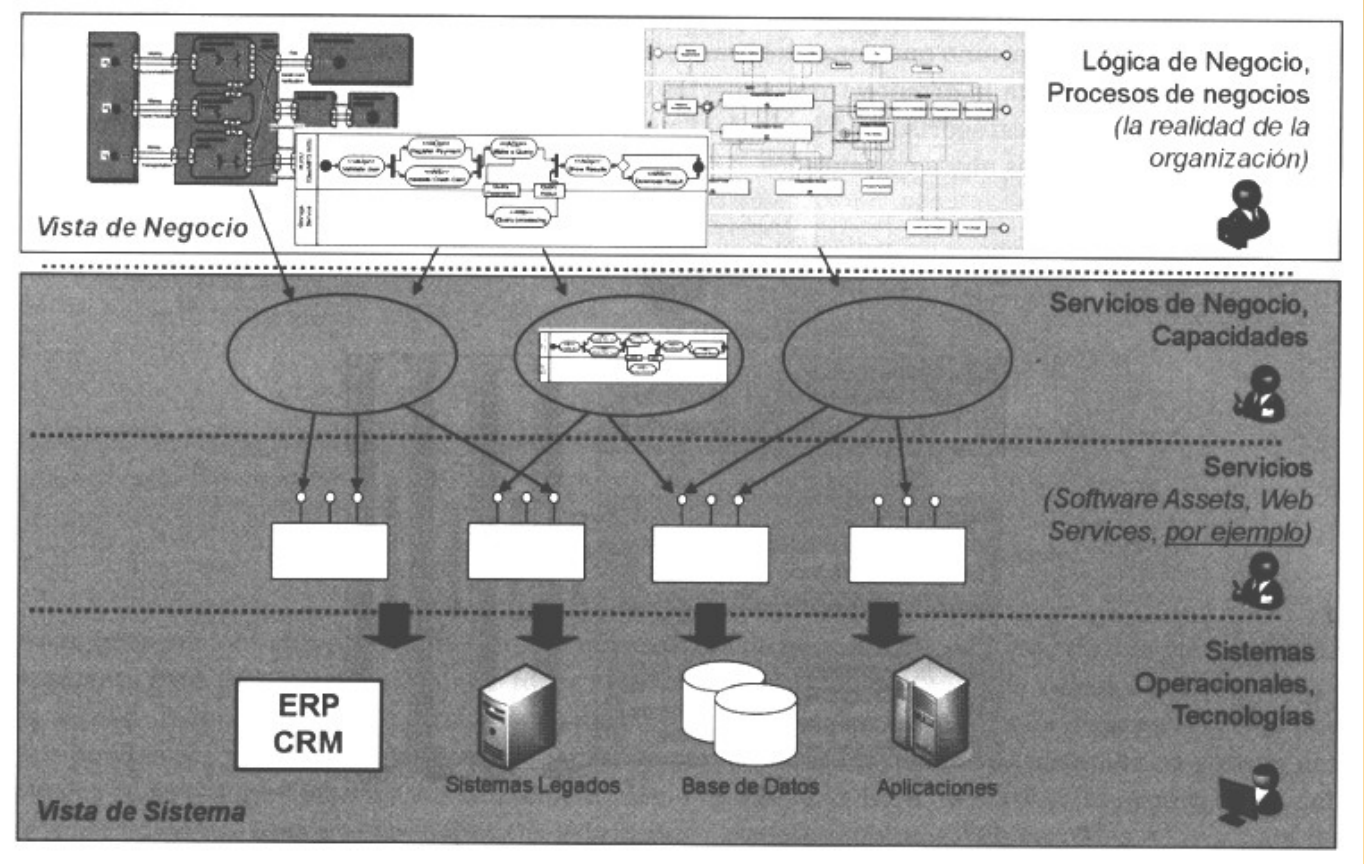
Funcionalidades transversales presentes en todas las capas:

- Seguridad
- Escalabilidad
- Rendimiento
- Comunicaciones
- Control de calidad

2. La arquitectura lógica

Arquitecturas orientadas a los servicios (SOA)

Permiten comunicar sistemas diferentes mediante protocolos bien definidos en XML.



Los estudiaremos en un tema posterior.

3. Patrones de software

Los patrones de software: Soluciones comprobadas a problemas comunes en el desarrollo de software.

Características de un patrón:

- Debe haber sido comprobado en otros sistemas.
- Debe ser fácilmente reutilizable.
- Debe ser aplicable a diferentes circunstancias.
- Debe estar bien documentado.

3. Patrones de software

Documentación típica de un patrón de software:

- Nombre.
- Problema que resuelve.
- Contexto en el que es aplicable.
- Fuerzas, objetivos y restricciones.
- Solución que propone.
- Ejemplos.
- Contexto resultante.
- Exposición razonada.
- Otros patrones relacionados.

3. Patrones de software

Tipos de patrones:

- De arquitectura (p. ej: MVC)
- De diseño
  - De creación de objetos
  - De estructura de clases
  - De comportamiento
- De dialectos
- De interacción o interfaz de usuario
- De análisis
- De dominio

3. Patrones de software

Ejemplo de patrón: el patrón Singleton

Algunos recursos en una aplicación son de tal naturaleza que sólo puede existir una instancia de ese tipo de recurso. Por ejemplo, la conexión a la base de datos a través de un manejador de base de datos. A veces interesa compartir un manejador de base de datos para que el resto de recursos no tengan que conectarse y desconectarse continuamente de la BD, y sólo debería existir una instancia de ese manejador.

El patrón Singleton cubre esta necesidad. Un objeto es “singleton” si la aplicación puede generar una y sólo una instancia del mismo.

```
<?php
require_once("DB.php");

class DatabaseConnection
{
  public static function get()
  {
    static $db = null;
    if ( $db == null )
      $db = new DatabaseConnection();
    return $db;
  }
```

3. Patrones de software

```
private $_handle = null;

  private function __construct()
  {
    $dsn = 'mysql://root:password@localhost/photos';
    $this->_handle =& DB::Connect( $dsn, array() );
  }

  public function handle()
  {
    return $this->_handle;
  }
}

print( "Handle = ".DatabaseConnection::get()->handle()."\n" );
print( "Handle = ".DatabaseConnection::get()->handle()."\n" );
?>
```