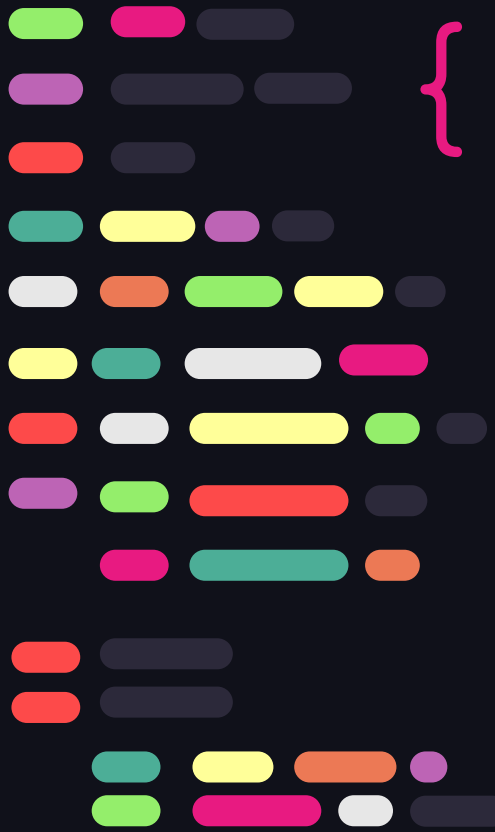


Arquitectura por Capas en Software

Programación Web 1

Jose Castellón - Nathan Castillo - Pablo Chavarría

Fundamentos y Principios

Definición de la arquitectura por capas

- Modelo estructurado que organiza una aplicación en distintos niveles.
- Cada capa tiene una función específica y se comunica con las capas adyacentes.
- Facilita la modularidad, el mantenimiento y la escalabilidad del software.

Fundamentos y Principios

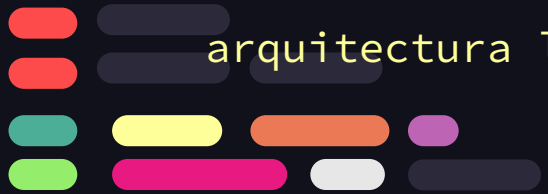
Características de la arquitectura por capas

- Organización estructurada
- Separación de responsabilidades
- Bajo acoplamiento y alta cohesión
- Abstracción y encapsulamiento
- Independencia tecnológica
- Mejor mantenibilidad y escalabilidad

Fundamentos y Principios

Evolución de la arquitectura por capas

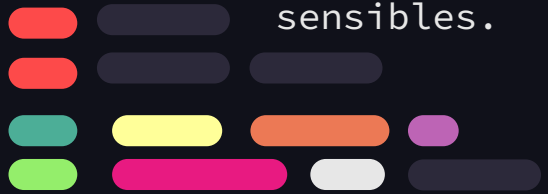
- Origen en sistemas monolíticos: Código integrado sin separación clara.
- Adopción del modelo de 3 capas: Presentación, lógica de negocio y acceso a datos.
- Expansión a modelos más avanzados: Inclusión de capas adicionales (servicios, persistencia, etc.).
- Aparición de nuevos enfoques: Arquitectura hexagonal, arquitectura limpia y microservicios.



Fundamentos y Principios

Beneficios de la arquitectura por capas

- Organización del código => Estructura clara y comprensible.
- Mantenimiento eficiente => Permite modificar componentes sin afectar todo el sistema.
- Reutilización de código => Posibilita aprovechar módulos en distintos proyectos.
- Separación de responsabilidades => Cada capa cumple una función específica.
- Facilidad de escalabilidad => Se pueden añadir nuevas funciones sin alterar toda la aplicación.
- Mayor seguridad => Restringe el acceso directo a datos y procesos sensibles.



Fundamentos y Principios

Desventajas de la arquitectura por capas

- Impacto en el rendimiento => La comunicación entre capas puede generar sobrecarga.
- Mayor complejidad => Requiere una planificación más detallada desde el inicio.
- Dificultad en la depuración => Problemas pueden originarse en múltiples capas.
- Posible sobre-ingeniería => Uso excesivo de capas puede complicar el desarrollo innecesariamente.
- Desafíos en la integración => Puede requerir esfuerzos adicionales para comunicarse con otros sistemas.

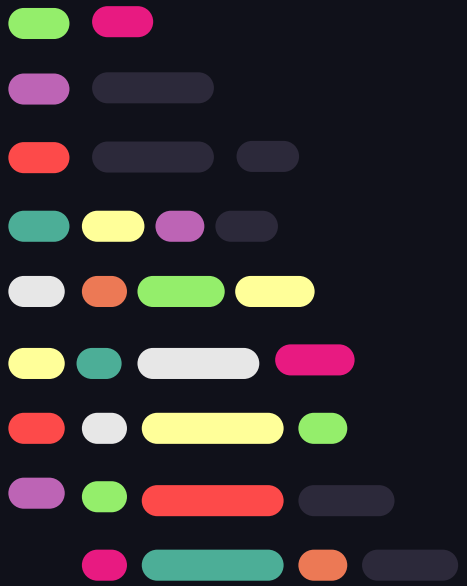
Fundamentos y Principios

Comparación entre arquitecturas

CRITERIA	MONOLÍTICA	POR CAPAS
Estructura	Código unificado	Dividida en niveles
Escalabilidad	Limitada	Alta
Mantenimiento	Difícil en proyectos grandes	Mas sencillo y modular
Desempeño	Puede ser más rápido	Puede generar sobrecarga
Reutilización	Baja	Alta
Complejidad inicial	Menor	Mayor planificación

Capas y su Funcionalidad

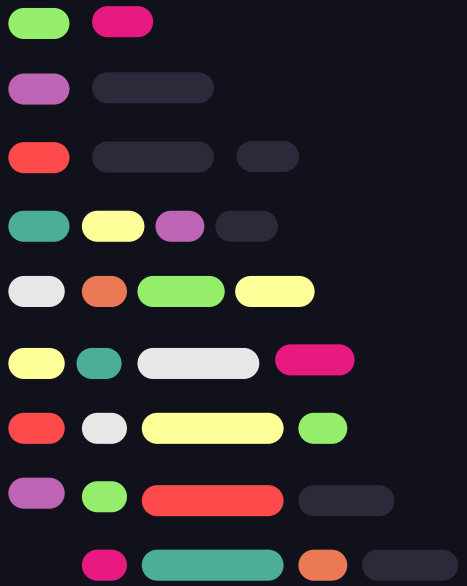
Capas de la arquitectura



- Presentación => UX/UI
- Aplicación o lógica de negocio
- Acceso a datos

Capas y su Funcionalidad

Capa de presentación



- Interacción con el Usuario.
- Gestión de la Experiencia del Usuario (UX/UI).
- Comunicación con la Capa de Lógica de Negocio.
- Manejo de Errores y Notificaciones.
- Implementación de Seguridad en el Frontend.

Capas y su Funcionalidad

Capa de presentación

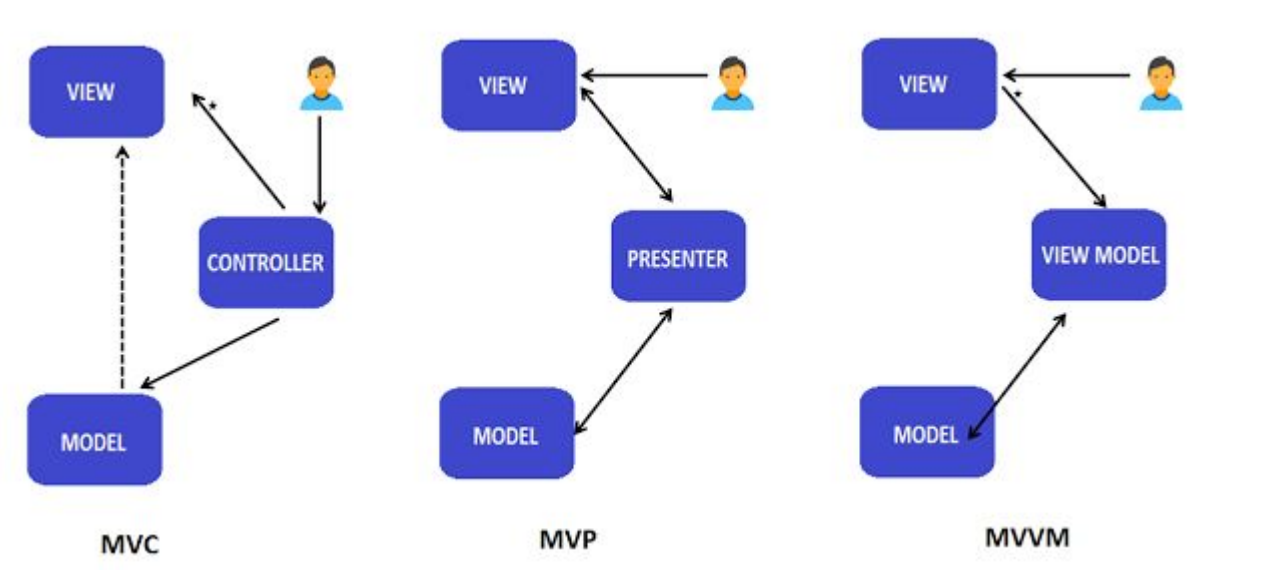
```
 import React, { useState } from "react";
 function LoginForm() {
   const [email, setEmail] = useState("");
   const [password, setPassword] = useState("");
   const handleSubmit = async (e) => {
     e.preventDefault();
     const response = await fetch("/api/login", {
       method: "POST",
       headers: { "Content-Type": "application/json" },
       body: JSON.stringify({ email, password }),
     });

     if (response.ok) {
       alert("Inicio de sesión exitoso");
     } else {
       alert("Error en las credenciales");
     }
   };
return (
     <form onSubmit={handleSubmit}>
       <input
         type="email"
         onChange={(e) => setEmail(e.target.value)}
         placeholder="Correo"
         required
       />
       <input
         type="password"
         onChange={(e) => setPassword(e.target.value)}
         placeholder="Contraseña"
         required
       />
       <button type="submit">Ingresar</button>
     </form>
   );
 }

 export default LoginForm;
```

Implementación y Patrones Relacionados

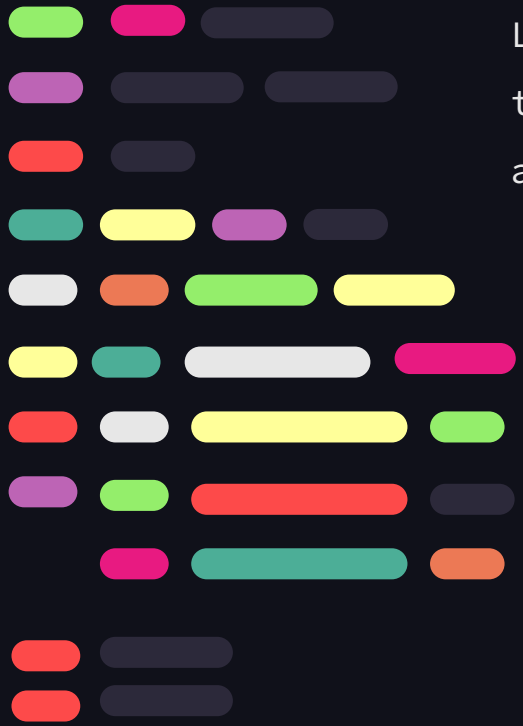
Uso de Patrones de Diseño en cada Capa.



Arquitectura: Hexagonal Vs. Tradicional por Capas

Características	Arquitectura Hexagonal	Arquitectura por Capas
Acoplamiento	Bajo	Medio
Flexibilidad	Alta	Media
Testeabilidad	Alta	Media
Complejidad Inicial	Alta	Media

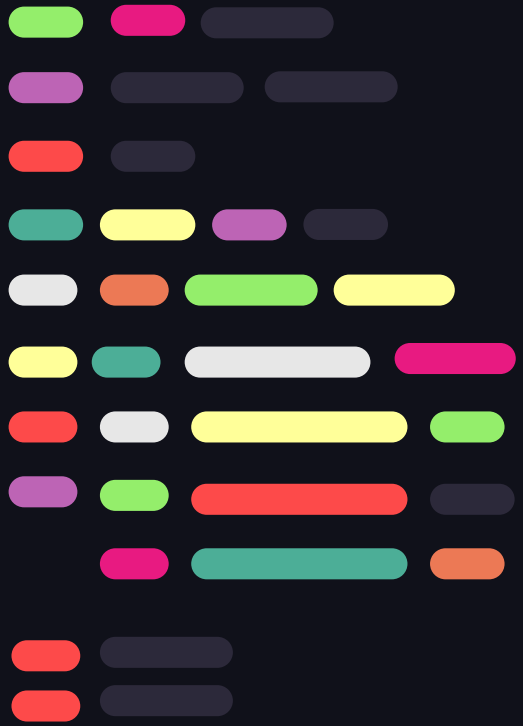
¿Cómo se diferencian del enfoque monolítico?



Los microservicios pueden ser organizados en capas pero tienen una diferencia clave con respecto a las arquitecturas del tipo Monolíticas*

- Cada servicio maneja una parte específica del dominio.
- La separación de responsabilidades es más granular.
- La comunicación entre microservicios suele realizarse mediante APIs REST o mensajería asíncrona.
- Se facilita el escalado independiente de cada servicio

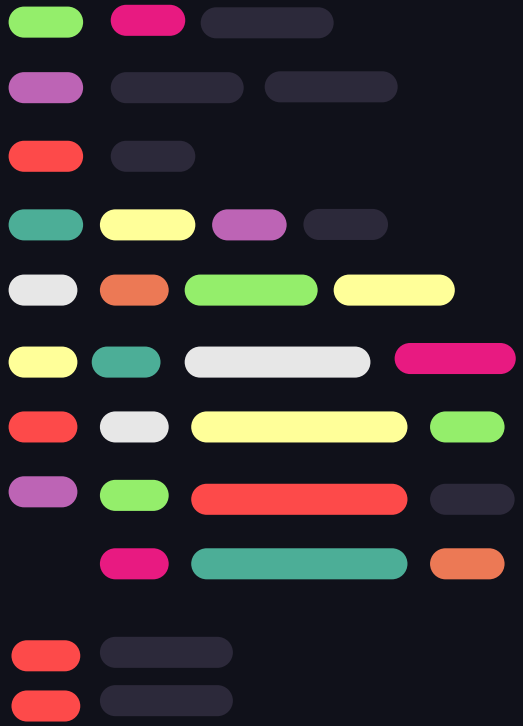
SPRING BOOT



Spring Boot es un framework de Java basado en Spring que simplifica la creación de aplicaciones web y de backend. Es un facilitador de implementación de arquitecturas por capas que se realiza mediante:

- Capa de Presentación: Controladores con Spring MVC.
- Capa de Negocio: Lógica de negocio y servicios.
- Capa de datos: Uso de JPA e Hibernate para la persistencia.

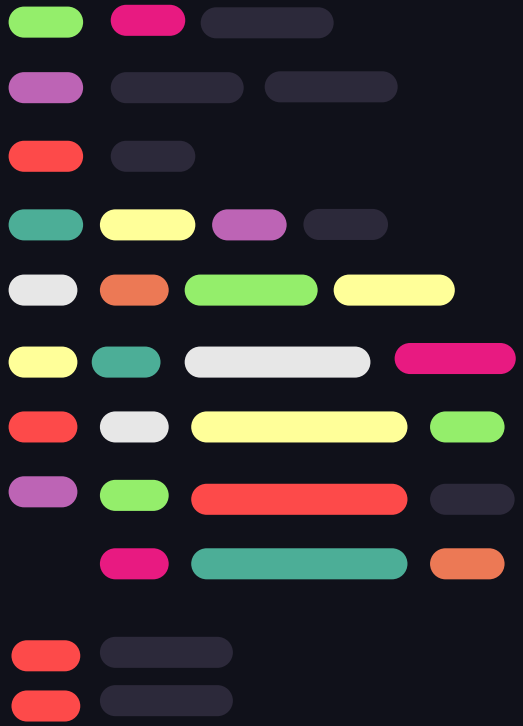
ASP.NET



ASP.NET es un framework de desarrollo web que permite construir aplicaciones web dinámicas, servicios y APIs usando .NET, basado en C# o VB.NET, proporcionando un entorno estructurado y optimizado para crear aplicaciones escalables y seguras. Realiza una implementación MVC por medio de capas con las siguientes características:

- Controladores en la capa de presentación.
- Servicios y Repositorios en la capa de negocio y datos.
- Uso de Entity Framework Core para acceso a datos.

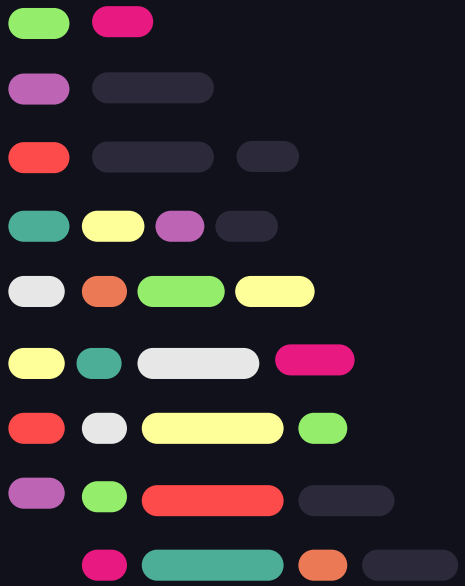
FLASK y Django



- **Flask:** Es un micro framework de desarrollo web para Python, que se caracteriza por ser simple y flexible. Lo que permite a los desarrolladores construir aplicaciones web sin imponer una estructura rígida.
- **Django:** Es un framework web completo, de alto nivel para Python que sigue el principio de “baterías incluidas” lo cual significa que tiene muchas funcionalidades listas para su utilización. Diseñado para facilitar el desarrollo rápido de aplicaciones web seguras y escalables.

Desafíos y Optimización

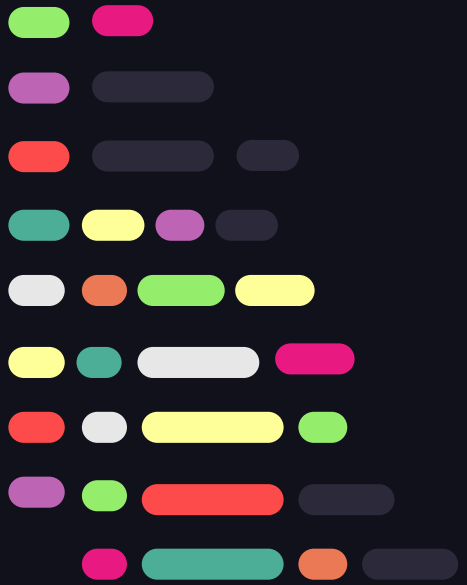
Problemas comunes en la implementación de Arquitecturas por capas



Dependencia Excesiva
Object-Relational Impedance Mismatch
Complejidad Excesiva
Sobrecarga de Comunicación

Desafíos y Optimización

Rendimiento y Escalabilidad en Arquitecturas por capas



Sobrecarga de Serialización/deserialización

Gestión de caché

Escalabilidad horizontal y vertical

Cuellos de botella

Balanceo de cargas

Desafíos y Optimización

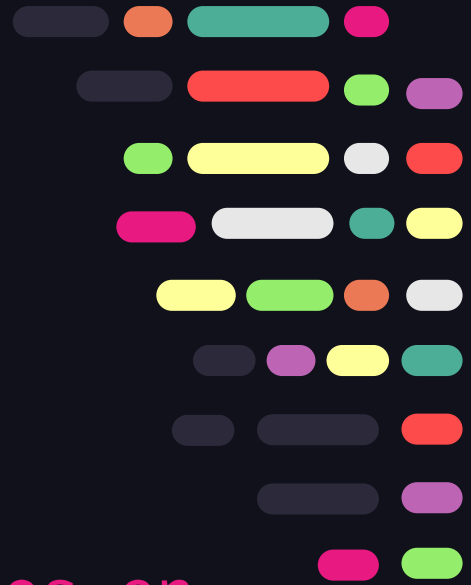
Cargar datos solo cuando sea necesario o anticipar necesidades.

Agrupar operaciones para reducir viajes a la base de datos.

Optimizar consultas y usar índices adecuadamente.

Reutilizar conexiones para minimizar overhead.

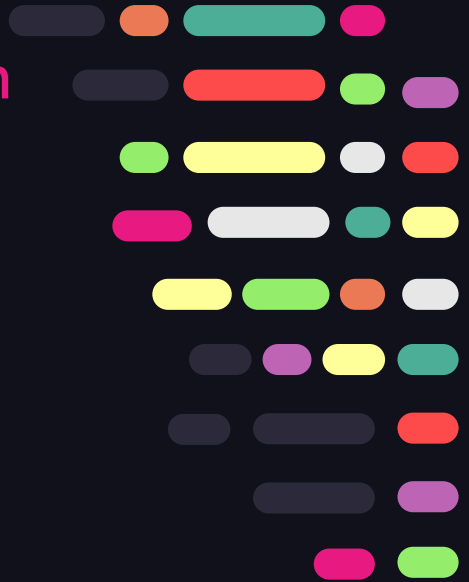
Facilitan la gestión eficiente de datos.



Optimización del acceso a los datos en arquitecturas por capas

Desafíos y Optimización

Cómo evitar la sobre ingeniería en una arquitectura por capas



{

Principio YAGNI (You Ain't Gonna Need It)

Iniciar con arquitectura simple

Abstracciones con propósito

Evaluación constante de valor vs complejidad

Domain-Driven Design

}

Casos de Uso y Aplicaciones reales

Estudio de caso: Implementación de una arquitectura por capas en una aplicación empresarial

Un ejemplo sería un sistema ERP:

- **Capa de presentación**
- **Capa de aplicación**
- **Capa de dominio**
- **Capa de infraestructura**

Casos de Uso y Aplicaciones reales

Arquitectura por capas en sistemas financieros y bancarios

Los sistemas financieros se benefician enormemente:

- **Seguridad por capas**
- **Auditabilidad**
- **Cumplimiento normativo**
- **Alta disponibilidad**
- **Integridad transaccional**

Casos de Uso y Aplicaciones reales

Aplicación de la arquitectura por capas en sistema de e-commerce

Capa de experiencia de usuario

Capa de catálogo y precios

Capa de procesamiento de pedidos

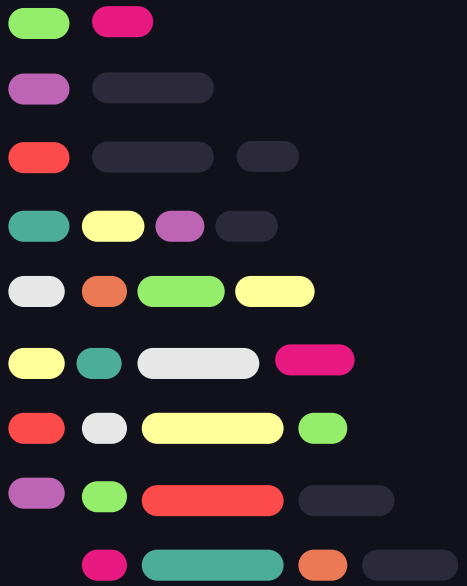
Capa de integración

Capa de análisis de datos

Casos de Uso y Aplicaciones reales

Como la arquitectura por capas influye en el desarrollo de software en la nube

Microservicios organizados por capas
Elasticidad diferenciada
Servicios gestionados
Observabilidad entre capas



Universidad
CENFOTEC
SOMOS LO QUE SABEMOS