**DECODER**

**JOSHUA ABRIL**

**PATRONES DE SOFTWARE**

**UNINPAHU**

**2025**

1. **¿Qué es el patrón de diseño Decoder?:**

El patrón **Decoder** es una estructura de diseño que se encarga de **traducir o transformar datos codificados** (como **JSON, XML, binario**, etc.) en estructuras de datos comprensibles y utilizables por un programa (como **objetos o clases**).

1. **¿Para qué se utiliza comúnmente el patrón Decoder?:**

Se usa para **interpretar datos de entrada** que provienen de formatos estructurados o serializados, y convertirlos en objetos del lenguaje anfitrión, permitiendo al sistema trabajar con ellos de manera nativa.

1. **¿Cómo ayuda el patrón Decoder en la interpretación de datos?:**

Ayuda al **aislar la lógica de conversión** en una clase o módulo especializado, separando la interpretación del formato del resto del sistema, lo que mejora la mantenibilidad y facilita el soporte de múltiples formatos.

1. **¿Cuál es la estructura típica de un patrón Decoder?:**

* **Algoritmo =**

[Input codificado] --> [Decoder] --> [Objeto/Modelo nativo]

Suele tener:

* Una interfaz común (**IDecoder**)
* Varias implementaciones específicas (**JsonDecoder, XmlDecoder**, etc.)
* Un modelo de datos destino (clases a las que se traduce la información)

1. **Un lenguaje o biblioteca donde se emplea el patrón Decoder:**

En **Swift**, la biblioteca **Codable** utiliza el protocolo **Decoder** para convertir datos **JSON o plist** en estructuras o clases nativas.

1. **¿Cómo se puede extender un patrón Decoder para nuevos formatos de datos?:**

Creando nuevas **clases que implementen la interfaz base** del **decoder** y que contengan la lógica para interpretar el nuevo formato. Por ejemplo, si ya existe y se tiene un **JsonDecoder**, se podría crear un **YamlDecoder** que use la misma interfaz.

1. **¿Cuál es la diferencia entre un Decoder y un Parser?:**

* Un **Parser** analiza la estructura sintáctica de una entrada (por ejemplo, un **JSON** válido).
* Un **Decoder** convierte esa estructura ya analizada en objetos concretos del sistema (como **instancias de clases**).

Un **parser** trabaja sobre la **forma**, un **decoder** sobre el **significado**.

1. **¿Qué retos presenta el diseño de un Decoder robusto?:**

Manejo de **errores y formatos inválidos**.

Soporte para **versiones diferentes** del mismo formato.

Rendimiento al procesar grandes **volúmenes de datos.**

Validación de **campos** requeridos o anidados.

1. **Un ejemplo donde un Decoder es esencial en una aplicación real:**

En una **API RESTful**, un **decoder** es esencial para convertir los **datos JSON recibidos** en objetos que la aplicación puede manipular internamente, como convertir un **POST /usuario** en una instancia de **Usuario**.

1. **¿Qué técnicas se pueden usar para probar un Decoder?:**

* **Pruebas unitarias** con entradas válidas e inválidas.
* **Tests de regresión** para asegurar que cambios futuros no rompan formatos antiguos.
* **Validación por esquema**, comparando el resultado contra una estructura esperada.
* **Fuzz testing** para detectar fallos con datos aleatorios o corruptos.