Implementando múltiples APIs en Rails

Jaime Zuberbuhler

Problema

Diferentes APIs

Diferentes formatos de solicitud/ respuesta

Solución

1. Patrón Adaptador y Patrón de Mapeo de Respuestas:

2. El Patrón Adaptador en Rails:

→ El propósito del adapter es intermediar y traducir lo que nuestro backend quiere decir a como cada servicio lo interpreta

Beneficio: Mantiene la lógica de negocio limpia y agnóstica a la API.

3. El Patrón de Mapeo de Respuestas en Rails:

→ El propósito del mapper es intermediar y traducir lo que el servicio responde a lo que nuestro backend interpreta

Beneficio: Garantizar datos normalizados para el backend de Rails.

Implementación

1. Definir si vamos a usar objetos o módulos
2. Definir las acciones
3. Definir estructura backend
4. Definir estructura adapter
5. Definir estructura mapper

Beneficios

Escalabilidad:

→ Agregar nuevos exchanges fácilmente.

Consistencia:

→ La aplicación siempre recibe respuestas predecibles.

Mantenibilidad:

→ No hay lógica específica de exchange en los controladores/servicios.

Flexibilidad:

→ Las API pueden manejar incluso diferentes productos (spot/perps en nuestro caso).

Casos de uso

Ruby on Rails:

→ Base de datos: active records, etc

Sistemas de scrapping:

→ Scrapping de múltiples sitios

Proveedores de datos:

→ Múltiples proveedores de datos

Proveedores de pagos:

→ Múltiples proveedores de pagos

Preguntas y respuestas

Moony PRO