Prueba técnica Frontend – AXPE Consulting

**Ejercicio 1:**1- Los problemas que detecté en la operación propuesta, son que no se está respetando el principio de responsabilidad única, no es una solución escalable ni modular, ya que la función getTotal accede directamente a los precios (streamingPrice, downloadPrice y additionalFee) de las clases de contenido multimedia (MultimediaContent y PremiumContent). Esto es problemático porque mezcla la lógica de precios dentro de la clase RegisteredUser, creando una dependencia innecesaria entre el usuario y los detalles internos de los contenidos. Otra observación que se detecté es que tampoco se respeta el principio abierto/cerrado ya que el comportamiento de una clase debe poder extenderse y no quedar sujeta a modificación, por lo que en la solución que propongo a continuación es utilizar el método de polimorfismo para respetar este principio, y permitir extensibilidad de servicios.  
  
Propongo la siguiente solución:  
  
// Clase base para todos los servicios

class Service {

constructor(content) {

this.content = content;

}

calculatePrice() {

throw new Error("calculatePrice must be implemented by subclasses");

}

}

// Servicio de Streaming

class StreamingService extends Service {

calculatePrice = () => this.content.streamingPrice + (this.content instanceof PremiumContent ? this.content.additionalFee : 0);

}

// Servicio de Descarga

class DownloadService extends Service {

calculatePrice = () => this.content.downloadPrice + (this.content instanceof PremiumContent ? this.content.additionalFee : 0);

}

// Clase para el usuario registrado

class RegisteredUser {

constructor(services = []) {

this.services = services;

}

getTotal = () => this.services.reduce((total, service) => total + service.calculatePrice(), 0);

}  
  
En esta solución, propongo utilizar un enfoque escalable en el que se separan las responsbilidades de los cálculos de precios según cada servicio, utilizando también operadores ternarios para simplificar la lógica condicional en lugar de utilizar “If/else”. De esta manera la clase RegisteredUser contiene instancias de ambos servicios (streaming, download), y queda abierto a otros servicios futuros.  
  
Con esta solución busqué la simplicidad, escalabilidad y respetar el principio de responsabilidad única y principio abierto/cerrado utilizando polimorfismo.  
  
  
  
Ejercicio 2:  
  
Se puede encontrar en el directorio del proyecto, un archivo Readme.md que contiene las instrucciones y características de como se implementó la prueba técnica, como también el enlace al repositorio remoto en Github.