1. ¿Porque piensa que se decidió implementar algunas de las necesidades como Servicios WEB? ¿Cuál es su ventaja?
2. ¿Qué otros servicios web pensaría que pueden ser útiles en este sistema? De al menos dos ejemplos, defina los endpoints de cada uno y explique brevemente cual es su función y porque pueden ser útiles

3. Explique brevemente las ventajas de usar:

1. CSR (Client Side Rendering)
2. PDO

4.  Explique el **patrón de diseño MVC**. Describa cada una de sus partes y las responsabilidades de las mismas. Ventajas y desventajas del uso de este patrón. ¿En qué tipo de sistema no sería útil usar este patrón?

1. La necesidad de implementar servicios web se dio cuando los navegadores dejaron de ser los únicos capaces de conectarse a la internet surgieron otros dispositivos/servicios. El navegador puede interpretar HTML en cambio el resto de los dispositivos necesita de otros formatos. En el caso de las API Rest, se pasan los datos en forma de JSON. La ventaja es que independiente del dispositivo, uno puede obtener la misma información proveniente del mismo lado y el cliente la interpreta como quiere.
2. Otros servicios web posibles seria el de los usuarios, en donde uno podría agregar, modificar, eliminar, traer todos los usuarios existentes. Funcionarían como el ABM tanto de usuarios como de categorías, además de poder retornar el listado de todos o de uno en específico.

Los posibles endpoints serían:

/user GET me trae todos los usuarios

/user POST publico un nuevo usuario

/user/:ID GET traigo un usuario con id específico

/user/:ID DELETE elimino un usuario con un id específico

/user/:ID PUT modifico un usuario con un id específico.

Otro servicio posible seria el las categorías, funcionando de la misma manera

/categoria GET me lista las categorías con su información.

/categoria POST agregar una categoría nueva

/categoria/:ID GET me lista una única categoría por su id

/categoria/:ID DELETE elimino una categoría por su id

/categoria/:ID PUT modifico una categoría por su id

Client side rendering, es mas fácil para aquellas páginas que requieren alta interacción y dinamismo con el usuario al poder usar JS. Ofrece además mejores experiencias UX UI y de usuario mas integrales y mas cómodas.

PDO es una layer que funciona como nexo entre la aplicación que estamos programando y la base de datos que utilizamos. Es un intermediario, al utilizarla no necesitamos sabes las características especificas de nuestra db que estemos usando en el momento ya que usa un lenguaje en común para todas.

4

El patrón de diseño de mvc consta de tres partes principales: vista, modelo y controlador.

Vista: es en donde van a estar todas las funciones o llamados involucrados en mostrarle al usuario (o también puede ser al que consume la API) el resultado de los diferentes pedidos. Es donde se renderiza el HTML que luego es lo que el usuario termina visualizando.

Modelo: la conección a la base de datos se realiza desde el modelo y todos los llamados ella. El modelo se encarga de devolver la información de la base datos pudiendo ser en forma de arreglos, de objetos etc.

Controlador: el encargado de realizar toda la lógica del sistema, es el único que se comunica tanto con el modelo como con la vista. La vista y modelo no se comunican entre si.

El router se comunica con el controlador y a partir de este se realizan todas las acciones o peticiones por parte del usuario.

Una desventaja podría ser que en caso de error puede ser mas complicado ver en donde esta el problema al no tener cada una de las páginas divididas o explícitas en distintos archivos.

Quizás no seria muy necesario en aplicaciones o páginas web muy chicas o aquellas que solo contengan un par de links incluidos con información de solo texto y muy poca interacción con el cliente.