Respuestas Teóricas Final Libre Web 2

Joaquín de la Iglesia

TUDAI – Olavarría

Cohorte 2021

Legajo 250993

2. a. Se decidió implementar el servicio web de lista de publicaciones del portal para poder obtener las nuevas publicaciones sin necesidad de refrescar la página, por ejemplo, y para aligerar su carga.

Los servicios WEB tienen varias ventajas, una de ellas es que se puede usar el mismo servicio para varias aplicaciones WEB, por ejemplo, puedo hacer una aplicación que muestre el clima en Olavarría utilizando el servicio WEB de OpenWeather, de la misma forma que hacen miles y miles de desarrolladores por todo el mundo. También, el hecho de que los datos se entreguen en formato texto, permite entender más fácilmente cómo es su funcionamiento interno y, de esa forma, desarrollar aplicaciones más fácil e intuitivamente. Por otro lado, también permiten que una misma aplicación, ubicada en distintos servidores, pueda acceder al mismo servicio WEB y trabajar de forma sincronizada con todos los servidores, este es el enfoque que siguen grandes aplicaciones WEB como Facebook, Instagram, Google, Twitter, entre tantas otras (por este motivo, cuando se cae un servidor, el problema suele ser localizado en una región y no a nivel global.) Además, este enfoque reduce al mínimo las pérdidas de datos, ya que, si un servidor de la aplicación ve afectado su almacenamiento, toda la información proviene del servicio WEB y no de sí mismo (todo esto exceptuando, claro está, aquellos casos en los que el servicio WEB es el afectado).

2. b. Personalmente implementaría los siguientes servicios WEB: Uno que permita borrar publicaciones, otro que permita activar y desactivar publicaciones y otro que permita activar y desactivar el Premium de un usuario.

Los endpoints serían /borrarPublicacion/:ID y se accedería mediante el método de petición DELETE, /alternarPublicacion/:ID y se accedería mediante el método PATCH, y /alternarPremiumUsuario/:ID y se accedería también mediante PATCH.

Creo que serían útiles ya que /borrarPublicación permitiría ser llamado con un botón desde CSR con Vue y así borrar las publicaciones más rápidamente, /alternarPublicacion tendría la misma ventaja, además de poder automatizar la desactivación de una publicación (creando una aplicación WEB que todos los días verifique desde cuándo se crearon todas las publicaciones, y si tiene más de 30 días, se desactive automáticamente). Con /alternarPremiumUsuario la idea es la misma que con /alternarPublicacion, que luego de 30 días el Premium se desactive automáticamente, y poniéndome creativo, creo que sería un enfoque interesante para por ejemplo activar automáticamente el premium luego de recibir un pago por MercadoPago, por ejemplo (utilizando la API de dicha plataforma).

3. i. Las ventajas de usar **CSR** son varias, entre las cuales destaca principalmente la posibilidad de mostrar datos luego de que la página terminó de procesarse del lado del servidor (como, por ejemplo, mostrar un comentario nuevo en una sección de comentarios). También se obtienen páginas más livianas y, por ende, que cargan más rápidamente después de la carga inicial. También la experiencia del usuario suele ser mejor. Suele utilizarse principalmente para aplicaciones web ya que permite que las acciones del usuario se realicen sin necesidad de refrescar el navegador, resultando más cómodo.

3. ii. Las ventajas de usar **PDO** son mayor compatibilidad entre engines de bases de datos, ya que, con el driver adecuado, se puede utilizar en la mayoría de los motores más utilizados. También es más seguro, ya que con las funciones prepare(), bind() y execute() protegemos las solicitudes de varios tipos de ataques, como por ejemplo, SQL Injection, ya que podemos preparar las solicitudes de antemano y evitar así que los atacantes inyecten su propio código.

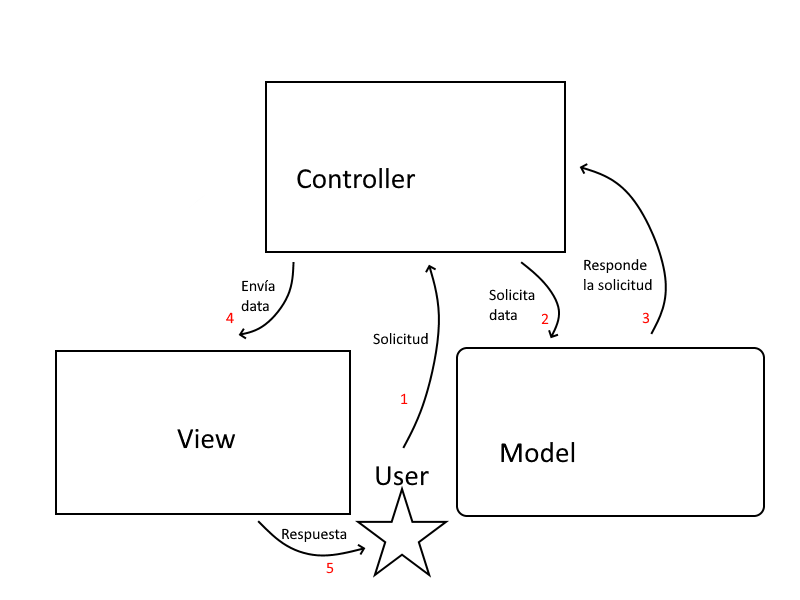
4. El patrón **MVC**, como la sigla indica, se compone de tres partes: el Model, el View y el Controller. El **Model** se encarga de gestionar la transferencia de datos con el motor de base de datos: solicita, actualiza, crea y elimina entradas y/o tablas de la base de datos, y devuelve el resultado al Controller.

El **View** se encarga de recibir los datos que envía el Controller y mostrarlos en pantalla al usuario. Asimismo, también le permite al usuario interactuar con la aplicación WEB, ya que el View puede mostrar, por ejemplo, un formulario donde se cargue un e-mail que desea enviar el usuario, o un botón para leer un e-mail entrante. Cuando se recibe una solicitud del usuario a través del cliente, ésta se envía al Controller.

El **Controller** se encarga de procesar los datos que serán enviados al Model y al View, como también los datos que son recibidos del Model (por ejemplo, cuando se solicita una entrada determinada) y también los datos que se reciben del cliente (cuando se envía un formulario, por ejemplo). Aquí suele hacerse la mayor cantidad de procesamiento de datos. Un ejemplo de su funcionamiento es, por ejemplo: Se desea ver un e-mail guardado en la base de datos, el Controller solicita a la base de datos la entrada de dicho e-mail, procesa los datos recibidos, y envía los datos ya procesados al View, que se encarga de mostrarlos al usuario. Luego, envía al Model una modificación a la entrada, para indicar que el e-mail ya fue leído.

En síntesis, cuando un usuario solicita, por ejemplo, que el index de una aplicación WEB se muestre, el Controller recibe la solicitud, el cual solicita al Model los datos necesarios para hacerlo, éste devuelve los datos solicitados, y el Controller los procesa, para poder enviarlos al View, que se encarga, finalmente, de mostrar el index solicitado.

En este burdo pero conciso dibujo se puede observar con más claridad este proceso. Los números en rojo representan el orden en el que van sucediendo las distintas acciones.



Algunas ventajas de la utilización del patrón **MVC** es que resulta muy rápido para desarrollar aplicaciones WEB, debido a su modularidad. Asimismo, esto permite que varios desarrolladores colaboren en conjunto sin entorpecerse unos con otros. La aplicación también es muy fácil de actualizar y depurar, ya que su estructura es bastante concisa y se puede deducir, por los errores, en qué módulo está la falla.

Las desventajas son pocas, pero algunas de ellas es que, para que sea útil, hay que ser muy estricto con la funcionalidad de cada módulo, para evitar que se desordene y la implementación de nuevas funciones sea caótica, como así su depuración y actualización.