Que es MVC?

El paradigma modelo vista controlador (MVC) es un patrón de diseño que separa el código en tres capas o modulos (Model,View,Controller).

Modelo: Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio).

Controlador: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta el 'modelo' (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto, se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo'.

Vista: Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario), por tanto, requiere de dicho 'modelo' la información que debe representar como salida.

Dentro del proyecto en la carpeta Controllers/Http definiremos los controladores asociados a cada vista, por ejemplo, tenemos el controlador de usuarios, el cual puede obtener datos, actualizar datos y eliminarlos. Este comportamiento deberá manejarse mediante eventos, es decir que si el usuario desea obtener información o realizar alguna actualización tendrá un método correspondiente.

De igual manera el modelo será el único que podrá interactuar con la BD y este modelo será consumido por su controlador correspondiente, a cada módulo deberá corresponderle un modelo. La información que le llegue al modelo ya debió haber sido procesado tanto en el front-end como en el controlador.

Dentro del proyecto estaremos implementando el principio de responsabilidad única, esto implica que una clase debe tener una y una sola razón para cambiar, lo que significa que una clase debe tener solo un trabajo y/o responsabilidad, es decir que para seguir este principio deberíamos usar el controlador para delegar sus funciones, lo que significaría que la validación de la información entrante debería realizarle en una clase independiente.

Para más información revise:

<https://jairogarciarincon.com/clase/programacion-orientada-a-objetos-en-php/patron-mvc>

<https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design-es>