MVC

¿Qué es el MVC?

El MVC (Modelo-Vista-Controlador) es un patrón de arquitectura utilizado comúnmente para implementar interfaces de usuario, datos y lógica de control. Enfatiza una separación entre la lógica de negocio y lo que se muestra al usuario. Esta separación de labores proporciona una mejor división de labores y mejor mantenimiento. Además, permite la reusabilidad de los componentes y promueve a un desarrollo modular del software.

¿Cómo se divide?

Las tres partes del MVC son:

1. Modelo: maneja los datos y la lógica de negocio.
2. Vista: maneja el diseño y la presentación.
3. Controlador: dirige los comandos al modelo y a las partes de la vista.

Ya que conocemos los conceptos de manera general, ahondemos en cada uno de ellos.

- El modelo define qué datos debe contener y la lógica de negocio de la aplicación. Es responsable de manejar los datos de la aplicación, procesar las reglas del negocio y responder a peticiones de información de los otros componentes.
- La vista es donde se define cómo deben ser mostrados los datos y manda las entradas de los usuarios al controlador. No interactúa directamente con el modelo; es decir, las peticiones pasan por el controlador y el modelo le responde a través del controlador.
- El controlador es el intermediario entre el modelo y la vista. Maneja la entrada y actualiza el modelo acorde a lo recibido. También se encarga de actualizar a la vista en caso de que el modelo haya tenido algún cambio. Contiene la lógica de la aplicación, como puede ser la validación de la entrada que insertan los usuarios y la transformación de los datos.

¿Cómo se comunican entre sí los componentes?

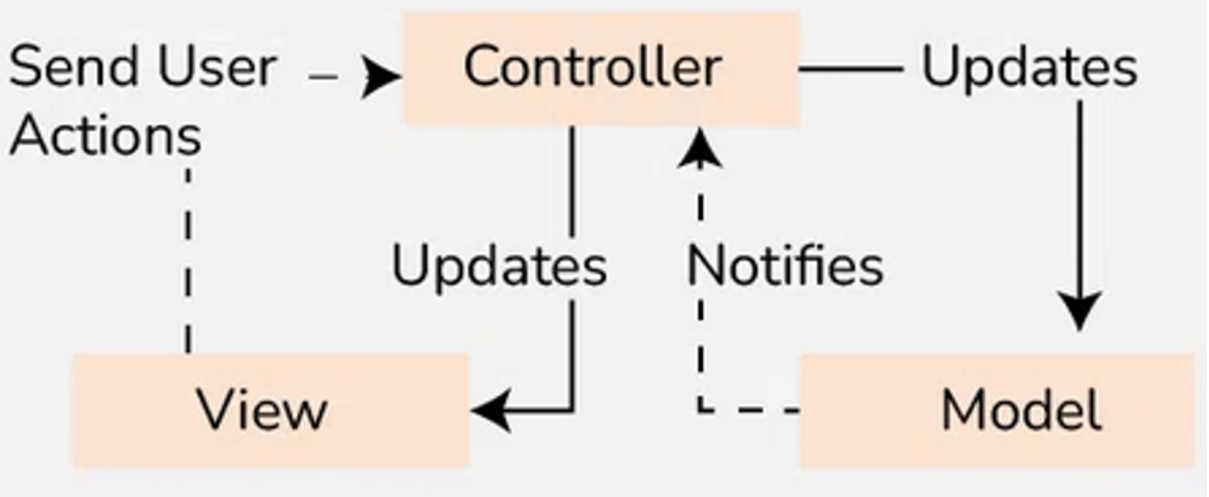

Diagrama que resume el funcionamiento del MVC

Como podemos observar en el diagrama y como se mencionó anteriormente, la vista y el modelo no se comunican directamente, sino que tienen al controlador como intermediario.

Podemos resumir las comunicaciones de esta forma:

- Usuario-Vista: el usuario ingresa texto o hace clic a un botón.
- Vista-Controlador: una vez recibida la entrada, se manda al controlador que la interpreta y actúa acorde a lo recibido.
- Controlador-Modelo: el controlador actualiza el modelo de acuerdo a la entrada recibida o la lógica de la aplicación.
- Modelo-Controlador: en caso de que se hayan hecho cambios en los datos, se tiene que actualizar la vista y, para lograrlo, se le notifica al controlador.
- Controlador-Vista: actualiza cualquier cambio hecho y son mandados a la vista para que los renderice.
- Vista-Usuario: la vista renderiza la información obtenida y la organiza para mostrarla al usuario.

Ventajas

- Separación de tareas. Ésto hace al código más fácil de entender, mantener y modificar.
- Modularidad: cada componente es desarrollado y probado por separado, lo cual promueve su reusabilidad y escalabilidad.
- Flexibilidad: ya que los componentes son independientes, cambiar uno no afecta a los otros dos, permitiendo así actualizaciones y modificaciones más fácilmente.
- Desarrollo paralelo: múltiples desarrolladores pueden trabajar en diferentes componentes a la vez, acelerando el proceso de desarrollo de la aplicación.

Desventajas

- Complejidad: su implementación puede añadir complejidad que conlleva a una sobrecarga en el desarrollo.
- Curva de aprendizaje: el hecho de aprender un patrón como éste conlleva tiempo y recursos adicionales a los presupuestados inicialmente en el desarrollo de una aplicación si no se conoce de antemano.
- Sobrecarga: si no se implementa una buena comunicación entre los componentes, puede haber una sobrecarga que afecte el desempeño de la aplicación, sobre todo en entornos de recursos limitados.
- Gran cantidad de archivos: este patrón se presta a realizar grandes cantidades de *scripts* comparado con otros patrones, lo cual puede crear una estructura de proyecto muy compleja y difícil de navegar.

¿Para qué se utiliza?

Este patrón suele ser utilizado en el desarrollo web. El modelo de datos está contenido en una base de datos, la cual puede estar implementada, por ejemplo, en MySQL si es relacional o MongoDB si no lo es; el código de control puede ser escrito en el lenguaje (o lenguajes) que maneje el proyecto; y vista (que conocemos como interfaz de usuario) podría estar escrita en CSS o HTML, por dar unos ejemplos.

Referencias:

- GeeksforGeeks. (2024, 19 febrero). *MVC Design pattern*. GeeksforGeeks. https://www.geeksforgeeks.org/mvc-design-pattern/
- *MVC - MDN Web Docs Glossary: Definitions of Web-related terms | MDN*. (2023, 20 diciembre). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Glossary/MVC