Opción 1

Guillermo Marion López, Karl Christian Deilmann, Louka David Vanhoucke

1.- Definición del Dominio:

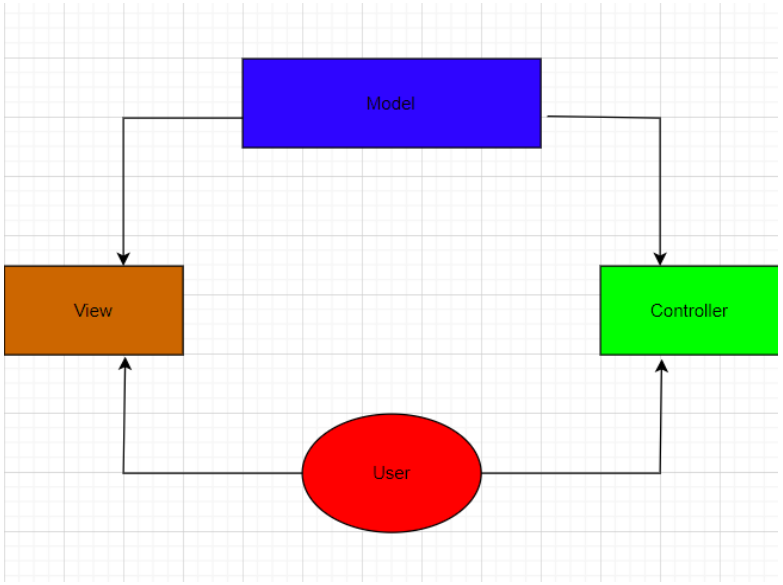
- El dominio de nuestra aplicación será un buscador de zonas culturales.
- Las principales funcionalidades son:
    - Buscador de zonas culturales
    - Planificador de rutas culturales
    - Mapa interactivo
    - Sistema de Feedback

2.- Identificación de los principales elementos de la arquitectura elegida:

Para nuestra app hemos elegido la arquitectura en 3 capas llamada MVC (Modelo vista y controlador), que está conformada por la capa de presentación (front-end), la capa de la lógica de negocio (back-end) y por último la capa de almacenamiento de datos.

- **Modelo:** Representa el dominio de la aplicación. Es responsable de almacenar los datos y la lógica de negocios.
- **Vista:** Representa la interfaz de usuario de la aplicación. Es responsable de mostrar la información al usuario y de responder a las acciones del usuario.
- **Controlador:** Es el intermediario entre el modelo y la vista. Es responsable de recibir las solicitudes del usuario, de interactuar con el modelo y de actualizar la vista.

3.- Diseño de la Arquitectura:



- **Arquitectura de tipo “Model-view-controller”**
  - **“View”:** El front-end parte → todo el usuario va ver
  - **“Controller”:** Servicios que sirven para conectar el “view” con nuestro back-end (como base de datos, autentificación, …).
  - **“Model” :** Presenta la lógica de negocio, se encargará del manejo de datos y de su almacenamiento.

4.- Caso de Uso:

Caso de Uso 1: Búsqueda de Zonas Culturales por Nombre

Este caso de uso implica que un usuario ingrese el nombre de una zona cultural (como un museo o una galería de arte) en un formulario de búsqueda en la interfaz de usuario y solicite resultados relacionados con ese nombre.

Capa de Presentación (Vista):

El usuario ingresa el nombre de la zona cultural en un formulario de búsqueda y hace clic en "Buscar".
La interfaz de usuario envía una solicitud de búsqueda al controlador en la capa de lógica de negocio (back-end).

Capa de Lógica de Negocio (Controlador):

El controlador recibe la solicitud de búsqueda y procesa el término de búsqueda.

Utiliza la lógica de negocio para buscar en la base de datos cualquier zona cultural cuyo nombre coincida con el término de búsqueda.

Recopila los resultados de la búsqueda.

Capa de Almacenamiento de Datos (Modelo):

La capa de datos recibe la solicitud de búsqueda de la capa de lógica de negocio.

Realiza una consulta a la base de datos para buscar zonas culturales que coincidan con el término de búsqueda.

Devuelve los resultados de la consulta al controlador.

Capa de Lógica de Negocio (Controlador):

El controlador recibe los resultados de la búsqueda de la capa de datos.

Formatea los resultados en un formato adecuado para la presentación.

Envía los resultados de vuelta a la capa de presentación.

Capa de Presentación (Vista):

La interfaz de usuario muestra los resultados de la búsqueda al usuario, que incluyen el nombre de las zonas culturales encontradas y detalles adicionales.

Caso de Uso 2: Visualización de Detalles de una Zona Cultural

Este caso de uso implica que un usuario haga clic en un resultado de búsqueda para ver detalles específicos de una zona cultural.

5.- Conclusión:

El patrón MVC es un patrón de arquitectura de software que se utiliza a menudo para aplicaciones móviles. Este patrón divide la aplicación en tres componentes principales: el modelo, la vista y el controlador.

Nuestro patrón elegido tiene algunas ventajas sobre las otras, aquí mencionamos algunos:

- **Separación de responsabilidades:** El modelo, la vista y el controlador tienen responsabilidades claramente definidas. Esto facilita el mantenimiento y la evolución de la aplicación.
- **Testabilidad:** Cada componente de la aplicación se puede probar de forma independiente.
- **Flexibilidad:** El patrón MVC se puede adaptar a diferentes dominios y aplicaciones.

Ventajas de otras arquitecturas

- **Patrón MVP:** El patrón MVP ofrece una mayor flexibilidad que el patrón MVC. El presentador se puede utilizar para separar la lógica de la aplicación de la interfaz de usuario.
- **Patrón MVVM:** El patrón MVVM ofrece una mayor flexibilidad que el patrón MVC. El modelo de vista se puede utilizar para separar la lógica de la aplicación de la interfaz de usuario.
- **Patrón MVVM-C:** El patrón MVVM-C ofrece una mayor flexibilidad que el patrón MVC. El controlador se puede utilizar para separar la lógica de la aplicación de la interfaz de usuario.

El patrón MVC es una buena opción para aplicaciones móviles que requieren una separación clara de responsabilidades. También lo hemos escogido ya que tenemos experiencia con trabajar con ese modelo y para nuestra aplicación es la más adecuada