**Patrones de Diseño**

**1. Patrón MVC (Modelo-Vista-Controlador)**

* **Contexto:** Este patrón es ideal para organizar la arquitectura general de la plataforma, separando la lógica de negocio (modelo), la interfaz de usuario (vista), y la gestión de las interacciones del usuario (controlador).
* **Aplicación:**
  + **Modelo:** Manejo de la lógica de gestión de usuarios (estudiantes y fundaciones), proyectos, aplicaciones y postulaciones.
  + **Vista:** Las interfaces de usuario para los estudiantes y fundaciones, asegurando que se ajusten a múltiples dispositivos (patrón responsivo).
  + **Controlador:** Procesamiento de las acciones de los usuarios, como aplicar a proyectos, publicar proyectos o gestionar aplicaciones.

**2. Patrón Singleton**

* **Contexto:** Es útil para asegurar que ciertas clases tengan una única instancia en todo el sistema, como los componentes que manejan configuraciones o conexiones con la base de datos.
* **Aplicación:**
  + **Gestión de Configuración:** Una única instancia que maneje la configuración global de la plataforma, como parámetros de escalabilidad y configuraciones de seguridad.
  + **Conexión a la Base de Datos:** Controlar la conexión a la base de datos para que solo exista una conexión activa centralizada.

**3. Patrón Factory Method**

* **Contexto:** Este patrón es útil para la creación de objetos sin especificar la clase exacta del objeto que será creado.
* **Aplicación:**
  + **Creación de Perfiles:** Diferenciar la creación de perfiles de estudiantes y fundaciones, asegurando que las instancias correspondientes se generen dinámicamente de acuerdo con los datos y tipo de usuario.
  + **Proyectos:** Generar diferentes tipos de proyectos (presenciales, remotos, híbridos) sin modificar el código base.

**4. Patrón Observer**

* **Contexto:** Este patrón puede aplicarse para notificar a ciertos componentes del sistema cuando ocurre algún cambio relevante, como la publicación de nuevos proyectos o la aprobación/rechazo de postulaciones.
* **Aplicación:**
  + **Notificaciones:** El sistema puede enviar notificaciones a los estudiantes cuando una fundación publique un nuevo proyecto, o cuando una postulación sea aceptada o rechazada. El patrón Observer permite implementar esta funcionalidad de manera eficiente.
  + **Actualización de Estado:** Las fundaciones pueden ser notificadas cuando los estudiantes apliquen a sus proyectos.

**5. Patrón Strategy**

* **Contexto:** Es útil cuando se necesitan varias maneras de realizar una tarea, permitiendo que el algoritmo utilizado varíe según las necesidades.
* **Aplicación:**
  + **Filtrado de Proyectos:** Dependiendo de los criterios de búsqueda (sector, ubicación, modalidad, duración), se puede aplicar un enfoque estratégico para realizar el filtrado de proyectos de diferentes maneras.
  + **Autenticación:** Puedes tener diferentes estrategias de autenticación, por ejemplo, mediante correo y contraseña, integración con redes sociales o SSO.

**6. Patrón Template Method**

* **Contexto:** Este patrón es útil cuando quieres definir la estructura general de un algoritmo o proceso, pero permitir que ciertos pasos puedan ser redefinidos o personalizados por las subclases.
* **Aplicación:**
  + **Proceso de Aplicación a Proyectos:** Las fundaciones pueden definir ciertos pasos específicos para el proceso de aplicación (como requerir entrevistas, subir portafolio, etc.), pero el flujo general (subir CV, llenar formulario, enviar) sería gestionado por el método base.

**7. Patrón Decorator**

* **Contexto:** Este patrón se utiliza para agregar responsabilidades adicionales a un objeto de manera dinámica.
* **Aplicación:**
  + **Mejora de Perfiles:** Permitir a los estudiantes agregar módulos o secciones adicionales a su perfil (por ejemplo, portafolios, certificaciones) sin alterar el perfil base.
  + **Proyectos:** Las fundaciones pueden decorar sus proyectos con características adicionales, como etiquetas o descripciones adicionales, para destacarlos dentro de las búsquedas de estudiantes.

**8. Patrón Command**

* **Contexto:** Se aplica cuando es necesario encapsular las solicitudes como objetos, lo que permite deshacer operaciones o realizar acciones en cola.
* **Aplicación:**
  + **Gestión de Aplicaciones:** Permitir a las fundaciones realizar operaciones como aprobar o rechazar aplicaciones, con la posibilidad de deshacer esas decisiones.
  + **Postulaciones de Estudiantes:** Encapsular las acciones de los estudiantes (postularse a proyectos, cancelar postulaciones) de forma que puedan gestionarse como comandos.

**9. Patrón Proxy**

* **Contexto:** Este patrón es útil para controlar el acceso a objetos de manera controlada, como cuando se manejan datos sensibles.
* **Aplicación:**
  + **Control de Acceso:** Las fundaciones solo deberían poder ver los perfiles completos de los estudiantes después de que estos apliquen a un proyecto. Un proxy puede controlar este acceso y limitar la información visible según las reglas establecidas.

**10. Patrón Adapter**

* **Contexto:** Si se necesita integrar sistemas externos (por ejemplo, servicios de autenticación de terceros o plataformas externas para gestionar CV), el patrón Adapter puede ser útil para que la plataforma se comunique con estos sistemas sin modificar su código.
* **Aplicación:**
  + **Integración con Sistemas Externos:** Implementar adaptadores para permitir el uso de servicios de autenticación, verificación de identidades, o incluso bases de datos externas.