**Actividad 2: Identificación y Justificación de Patrones de Diseño**

**Introducción**

Para diseñar una plataforma inteligente de gestión de proyectos que sea flexible y fácil de mantener, es fundamental elegir patrones de diseño que resuelvan problemas comunes y se adapten a nuestros requerimientos. Después de evaluar las necesidades del sistema, he seleccionado un patrón creacional, uno estructural y uno de comportamiento, que en conjunto permitirán que el software crezca y evolucione de manera orgánica.

**1. Patrón Creacional: Factory Method**

El patrón Factory Method es perfecto para la creación de objetos de forma flexible. En nuestra plataforma, este patrón permitirá instanciar diferentes componentes (como módulos de análisis predictivo o gestión de tareas) sin depender de clases concretas. Esto no solo simplifica la incorporación de nuevas funcionalidades, sino que además desacopla el proceso de creación del objeto de su utilización, haciendo que el código sea más fácil de mantener y extender en el futuro.

**2. Patrón Estructural: Facade**

El patrón Facade nos ayuda a ocultar la complejidad interna del sistema detrás de una interfaz sencilla. Imagina que tenemos varios subsistemas, como el motor de inteligencia artificial, la gestión de bases de datos y el sistema de notificaciones; Facade se encargará de unificar su interacción, permitiendo a los desarrolladores y usuarios trabajar con una única interfaz amigable. Esto simplifica la experiencia de uso y facilita el mantenimiento, ya que cualquier cambio interno no afectará la manera en que se interactúa con el sistema.

**3. Patrón de Comportamiento: Strategy**

El patrón Strategy es ideal para encapsular diferentes algoritmos y hacerlos intercambiables según la necesidad. En el contexto de nuestra plataforma, este patrón permitirá implementar y cambiar estrategias de análisis predictivo o recomendaciones de productividad sin alterar la estructura principal del software. Esto significa que, si en el futuro queremos ajustar o mejorar nuestros algoritmos, podremos hacerlo de forma dinámica y sin complicaciones.

**Conclusión**

La elección de estos patrones se basa en la búsqueda de simplicidad y flexibilidad. Con el **Factory Method** aseguramos una creación desacoplada y extensible de objetos; el **Facade** nos brinda una interfaz clara y amigable para interactuar con sistemas complejos; y el **Strategy** nos permite adaptar el comportamiento del sistema de forma dinámica. En conjunto, estos patrones no solo responden a las necesidades actuales de la plataforma, sino que también la preparan para futuros desafíos, garantizando un crecimiento sostenible y un mantenimiento eficiente.