**Elementos evaluados en esta prueba práctico (elementos teóricos evaluados implícitamente):**

- Instalación y configuración de Symfony como Framework PHP (a)

- Implementación de conceptos como: herencia, polimorfismo, POO y patrones. Elementos que son imprescindibles para desarrollar el caso de estudio que se plantea. (a)

- Uso de patrones y principios SOLID. (e)

- Elementos conceptuales y prácticos de DDD, como son agregados, identidad, objetos de valor, “*boundaries”*, consistencia de invariantes, lenguaje ubicuo, eventos, etc. (a, e)

- Arquitectura hexagonal en la estructuración del proyecto y la relación entre elementos. Mantenimiento de dominio separado de infraestructura. Garantizar bajo acoplamiento. (a, b, c, d)

-Diseño de API REST y CQRS. (d)

**1. Comprobación práctica. DDD y Hexagonal en SF**

1. Supongamos que estás diseñando un sistema de gestión de inventario para una tienda en línea y que la entidad central del sistema es el Producto. Un Producto tiene un nombre, una descripción, un precio y una cantidad en stock. Además, un Producto puede tener múltiples **variantes**, como **diferentes tamaños**, colores, etc. y cada **variante tiene su propio precio y cantidad en stock**. Cada **variante también tiene una imagen asociada**.
   * Implemente en un proyecto nuevo con Symfony 6.2 estas entidades y sus invariantes, deberás diseñar una estructura de clases utilizando el enfoque de DDD “*inside-out"* y arquitectura hexagonal.
   * Describa con palabras la estrategia que ha seguido.
2. Implemente la interacción de la base de datos (MySQL) con el dominio para la persistencia de Productos. Un **puerto** que defina las funcionalidades que podrá realizar nuestro dominio con la dependencia externa y un **adaptador** que implemente la relación entre la dependencia externa y nuestro dominio.
3. Implemente en la capa de aplicación las acciones necesarias para hacer uso de los elementos de infraestructura anteriores para persistir el Producto y sus relaciones.
4. Diseñe y desarrolle un punto de acceso REST para crear un nuevo producto en la aplicación. Asegúrese de utilizar el formato de intercambio JSON y de tener en cuenta el patrón CQRS durante el desarrollo. Construya un nuevo adaptador para persistir en otro sistema de persistencia de datos (el que decida).
5. Diseñe un evento de dominio que se lance cuando se cree un producto nuevo e implemente un servicio que represente un sistema de envío de correos electrónicos cada vez que se persista un producto notifique a la dirección de correo pepe@up-spain.com. El servicio debe permitir enviar correos electrónicos utilizando distintos servicios de correo electrónico, como SMTP, Amazon SES, SendGrid, etc.
   * Se debe aplicar los principios SOLID:
     1. responsabilidad única (SRP),
     2. abierto-cerrado (OCP)
     3. sustitución de Liskov (LSP).