La prueba técnica implica la creación de un sistema de gestión de inventario para una tienda en línea utilizando Symfony (Laravel), DDD (Domain Driven Design), Hexagonal Architecture, CQRS (Command Query Responsibility Segregation), y principios SOLID. A continuación, una guía paso a paso para cada parte de la prueba:

**a) Diseño de Entidades con DDD y Arq. Hexagonal:**

1. **Estructura de Clases:**
   * Define las entidades: **Producto**, **Variante**, etc.
   * Utiliza el enfoque "inside-out" de DDD.
   * Establece las invariantes del dominio para garantizar la consistencia de los datos.
2. **Arquitectura Hexagonal:**
   * Separa claramente las capas de dominio, aplicación e infraestructura.
   * Define puertos y adaptadores para la interacción con la base de datos.
3. **Estrategia Desarrollada:**
   * Describe la estrategia seguida para el diseño de las entidades y la aplicación de DDD y Hexagonal.

**b) Persistencia de Productos en MySQL:**

1. **Puerto y Adaptador:**
   * Crea un puerto que defina las funciones del dominio para interactuar con la base de datos.
   * Implementa un adaptador que conecte el dominio con la base de datos MySQL.

**c) Capa de Aplicación:**

1. **Acciones para la Persistencia:**
   * Implementa las acciones necesarias en la capa de aplicación para utilizar el puerto y adaptador anteriores.

**d) Diseño de API REST con CQRS:**

1. **Punto de Acceso REST:**
   * Diseña un punto de acceso REST para crear un nuevo producto.
   * Utiliza el formato JSON para el intercambio de datos.
2. **Patrón CQRS:**
   * Divide las operaciones de lectura y escritura.
   * Implementa un adaptador para persistir en otro sistema de persistencia de datos.

**e) Evento de Dominio y Servicio de Correo:**

1. **Evento de Dominio:**
   * Diseña un evento de dominio que se lance cuando se crea un nuevo producto.
2. **Servicio de Correo:**
   * Implementa un servicio de envío de correos electrónicos.
   * Notifica a la dirección de correo [pepe@up-spain.com](mailto:pepe@up-spain.com) cuando se persiste un nuevo producto.
3. **Principios SOLID:**
   * i. **SRP (Responsabilidad Única):** Asegúrate de que cada clase tenga una única razón para cambiar.
   * ii. **OCP (Abierto-Cerrado):** Las entidades, módulos, etc., deben estar abiertos para la extensión pero cerrados para la modificación.
   * iii. **LSP (Sustitución de Liskov):** Las instancias de una clase base deben poder ser reemplazadas por instancias de una de sus clases derivadas sin afectar la correctitud del programa.

**TODO:** Recordar realizar pruebas.