**Organización de Clases**

La organización adecuada de las clases es clave para la mantenibilidad del código. En el código que se desarrolló, cada clase está definida dentro de paquetes específicos, lo que mejora la estructura y la legibilidad del proyecto. Los paquetes models y controllers separan las responsabilidades del modelo de datos y la lógica de control, respectivamente, lo que facilita la navegación y comprensión del código.

Declaración de Variables: Las variables de instancia son privadas, asegurando el principio de encapsulación. Esto se ve en clases como Error, User, y Product, donde las variables se declaran al inicio de la clase. El uso de anotaciones como @Getter y @Setter de Lombok simplifica la gestión de accesores, manteniendo el código limpio y reduciendo la repetición.

Métodos Públicos: Los métodos públicos siguen a las variables y ofrecen funcionalidades claramente definidas. Por ejemplo, en las clases de controladores como UserController y AuthController, los métodos públicos proporcionan puntos de entrada para las operaciones de la API, siguiendo una estructura lógica que mejora la legibilidad.

**Encapsulación**

El principio de encapsulación fue implementado, asegurando que las variables y métodos solo sean accesibles donde realmente se necesitan:

Privacidad de Datos: Las clases User y Product mantienen la privacidad de sus variables usando modificadores de acceso privado. Esto protege los datos de cambios externos no deseados y garantiza que las interacciones con estas clases se realicen a través de métodos bien definidos.

Métodos de Acceso: Las clases utilizan métodos públicos para acceder a los datos encapsulados. Este enfoque permite la validación y transformación de datos antes de ser expuestos o modificados, como se observa en los métodos createUser y updateUserById en UserController.

**Tamaño y Responsabilidad de Clases**

**Clases Pequeñas y Enfocadas**: Cada clase tiene una única responsabilidad, como Product para representar productos, User para usuarios, y RequestExchange para las solicitudes de intercambio. Esta separación de responsabilidades facilita el mantenimiento y la comprensión del código.

**Controladores y Servicios**: Los controladores están enfocados en manejar las solicitudes HTTP y delegar la lógica de negocio a los servicios correspondientes. Esto sigue el principio de responsabilidad única y mejora la estructura del código.