Como el proyecto es para usar como portafolio la idea de este documento es mostrar algunos detalles y características sobre la implementación. También al final del documento se muestran desde swagger los endpoints creados y se muestran algunas pruebas realizadas sobre la API.

**Diagrama de clases**

Antes de empezar a escribir código realice el diagrama de clases pensando en obtener todo lo mínimo para que la aplicación funcione. Como resultado obtuve el siguiente diagrama UML (para que sea mas claro quitaron los métodos e identificadores de acceso).

Diagrama

Descripción generada automáticamente

La relación entre post y comentario se hizo bidireccional ya que era útil para obtener los comentarios desde la entidad post.

**Paquetes/Estructura**

Con el fin de aplicar buenas practicas el código fue dividido en capas claramente distinguibles y cada una con una funcionalidad especifica.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Modelo Post y validaciones**

Al ser muchos modelos solo muestro el de posts que es el que mas relaciones tiene. Como se puede observar, la relación con los comentarios se configuro como FetchType.LAZY, esto es para que los comentarios relacionados se traigan de la base de datos solo cuando sea necesario. Esto aumentara el rendimiento de la aplicación, ya que en primer momento solo se deben mostrar los posts, luego si el usuario quiere se muestran los comentarios. También se puede ver la anotación @JsonIgnore para que a la hora de obtener un post no se genere un loop con los comentarios debido a que es una relación bidireccional.

Texto

Descripción generada automáticamente

Para evitar inconsistencias en la base de datos se aplicaron validaciones a cada formulario recibido. En este caso se verifica principalmente que el post tenga todas las relaciones necesarias. En caso de que no pase las validaciones se retorna una lista con todas las validaciones en las que fallo.

Texto

Descripción generada automáticamente

**Manejo de excepciones**

Spring con la anotación @RestControllerAdvice nos permite captar todas las interrupciones lanzadas por la aplicación en tiempo de ejecución y tratar cada tipo de ellas de manera personalizada. En la siguiente imagen se puede ver como se tratan las excepciones lanzadas por el validador de formularios y cuando no se encuentra un recurso.

Texto

Descripción generada automáticamente

**Custom querys**

Con el fin de buscar posts específicos se crearon consultas personalizadas con el fin de filtrar desde la base de datos y no tener hacer procesamiento de datos innecesarios.

Interfaz de usuario gráfica, Texto, Chat o mensaje de texto

Descripción generada automáticamente

**Endpoints**

**Patrón de fondo

Descripción generada automáticamente**

**Patrón de fondo

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Aplicación, Correo electrónico

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

**Test para agregar un usuario**

A continuación, se muestran las validaciones al hacer un post.

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente**

Ahora se agrega un usuario que respeta todas las reglas.

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

Se intenta agregar un usuario con un mail que ya esta en la base de datos

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

**Test para agregar un post**

A la hora de agregar un post deben existir todas las demás relaciones como el local, el usuario, el producto y la categoría. Si una de ellas no existe lanza una excepción.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto, Chat o mensaje de texto

Descripción generada automáticamente

Si se agrega un post donde las relaciones existen

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Si bien no nos retorna los datos de las entidades relacionadas el post se crea de manera correcta.

Texto

Descripción generada automáticamente

Si hacemos un get del post con id 4

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente