**Técnicas Avanzadas de Programación**

**Modelado Red Social**

**Integrantes:**

**Alan Yoel Gutman**

**Turno: miércoles – Tarde**

**Docente: María Belén Alegre**

**Fecha: 28 de abril de 2021**

Contenido del cuestionario

*Se pide generar el modelo de una red social, que consiste en un muro donde los usuarios pueden publicar su estado y los amigos pueden realizar comentarios y poner me gusta. Según las siguientes consideraciones:*

* *Crear las clases mínimas necesarias involucradas*
* *Determinar las relaciones entre los objetos elegidos para la implementación*
* *Determinar el modelo de datos a utilizar. (Motor de base de datos, tablas)*
* *Determinar si es posible implementar inyección de dependencias en alguna de las relaciones que haya considerado.*

*Consideraciones:*

*Teniendo en cuenta que podrá ser un sistema del tipo cliente - servidor, ver el modelado orientado a la separación de capas y funcionalidades aisladas.*

Entregables

* *Documento con el análisis del proyecto*
* *Detalle de todas las consideraciones realizadas*
* *Estructura de las clases involucradas*
* *Modelo de datos*
* *Justificación de cada decisión*

Análisis del proyecto

*La solución propuesta consiste en realizar un sistema donde la persona pueda ser capaz de visualizar una publicación particular junto a todos los comentarios que compongan dicha publicación. Para ello el usuario deberá previamente ingresar el código del posteo en el sistema.*

*La solución dispondrá de la funcionalidad necesaria para que los usuarios puedan cargar publicaciones y comentarios, como también la posibilidad de dar me gusta a los mismos.*

*Cabe aclarar que no se contempla para esta primera etapa del proyecto un servicio de registración de usuarios, siendo que los mismos deberán enviar un correo electrónico y un administrador los dará de alta en el sistema en un lapso de 48 horas hábiles.*

*Tampoco se tendrá en cuenta la validación de los datos ingresados por el usuario para la carga de las publicaciones o comentarios.*

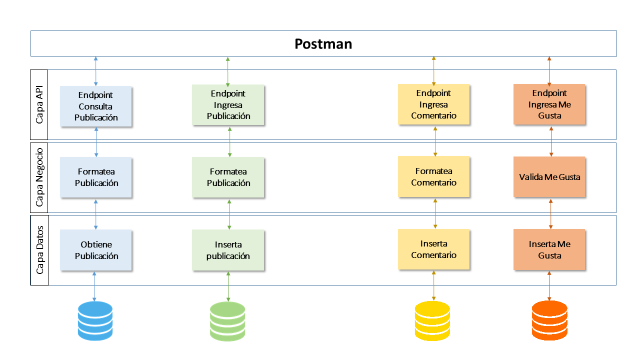
Arquitectura

*Teniendo en cuenta que podrá ser un sistema del tipo cliente - servidor, haré uso del patrón de arquitectura MVC porque me ayudará a desacoplar en módulos claramente identificables el acceso a los datos de la capa de lógicas y reglas de negocio, y la de interacción con el usuario. Con esto logramos un código que sea más fácil de entender, de codificar y de mantener; nos permite un mejor manejo de errores y que el sistema resulte fácilmente escalable.*

 *Ciclo de vida del MVC*

*Los servicios detectados y dispuestos por la API serán los siguientes:*

* *Consulta de las publicaciones: Será el encargado de construir y devolver la estructura JSON con los datos de la publicación solicitada por el usuario.*
* *Carga de las publicaciones: Será el encargado de guardar la publicación en el sistema a partir de los datos ingresados por el usuario.*
* *Carga de los comentarios: Será el encargado de agregar los comentarios a la publicación solicitada por el usuario en el sistema.*
* *Carga de me gusta: Será el encargado de agregar me gusta a la publicación o al comentario indicado por el usuario.*



Tecnologías

*Para el desarrollo se utilizará una tecnología orientada a micro servicios utilizando como herramienta* ***Spring Framework 4 (Spring Tool Suite 4)*** *porque según veo me permite generar una configuración inicial de base y luego ir agregando módulos de forma incremental a medida que vayamos necesitando. También se encuentra mucha documentación y tutoriales lo que me permite aprenderlo relativamente fácil.*

*Para el almacenamiento y gestión de los datos utilizaremos una base de datos relacional* ***Microsoft SQL Server 2019*** *por mi familiaridad con dicho motor de base de datos junto con la herramienta* ***JPA & Hibernate*** *la cual ayuda a generar la estructura de base de datos de la aplicación a partir del modelo de clases declarado de forma automática permite trabajar con la base de datos por medio de entidades en lugar de tener que estar generando Querys específicas.*

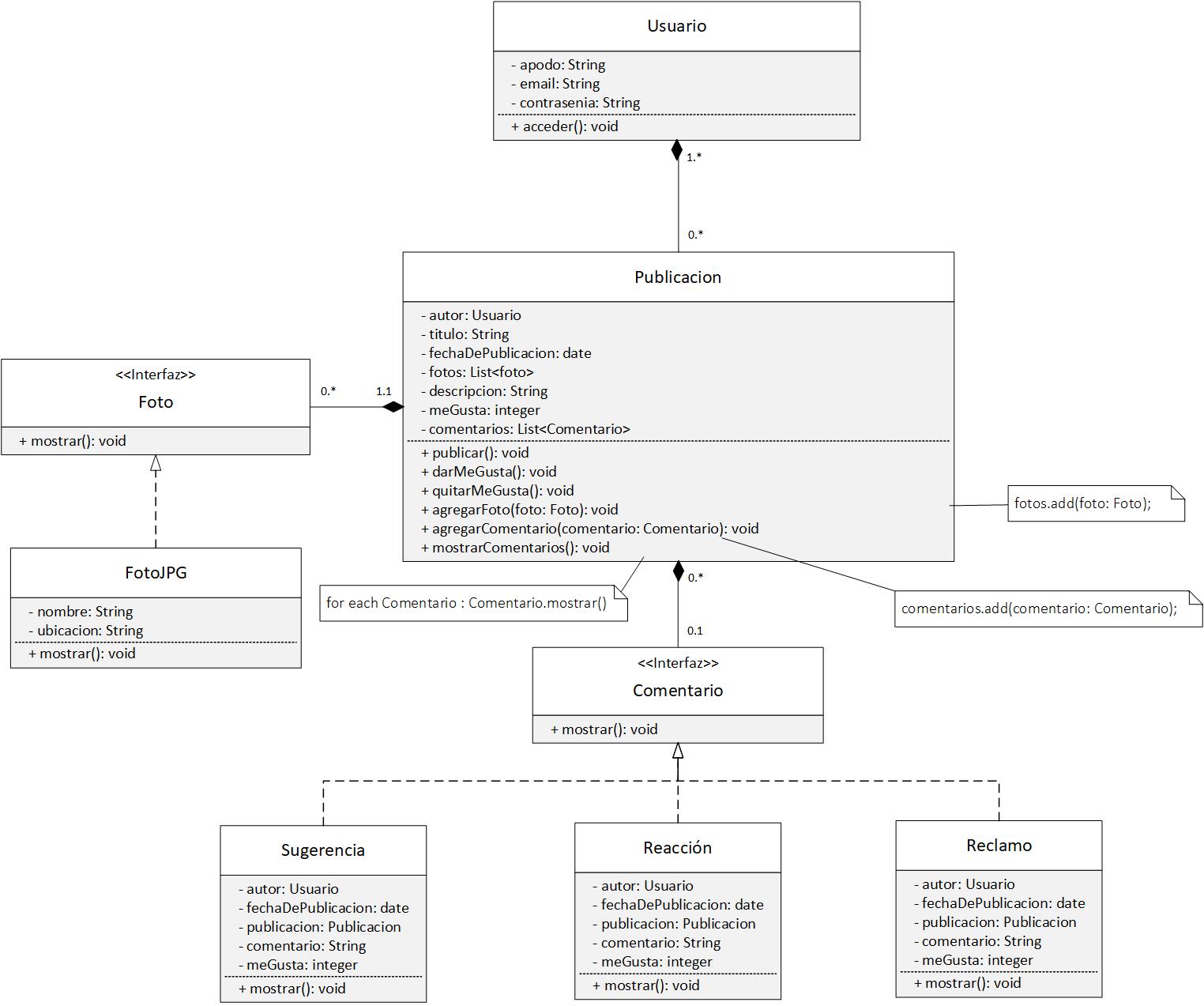
*Para el versionado de nuestro desarrollo utilizaremos* ***GIT Hub*** *ya que el mismo es gratuito y tiene una amplia comunidad de usuarios, además de ser simple de manejar.*

*Por último, al no contar con una interfaz de usuario, utilizaremos* ***Advanced Rest Client*** *para validar la correcta funcionalidad de nuestro desarrollo.*

Diagrama de Clases

*En esta red social tendremos las siguientes entidades:*

* *Los usuarios. Éstos pueden publicar su estado, y escribir un comentario y poner me gusta en una publicación.*
* *Las publicaciones. Las mismas pueden tener un título, 1 o varias fotos y una descripción. A su vez los usuarios pueden agregar comentarios o poner me gusta a los mismos. Necesito contabilizar la cantidad de “Me Gusta”.*
* *Los comentarios. Éstos son textos que publica un usuario en una publicación y por ende no pueden existir por sí mismos, sino que deben estar asociados a una publicación. Necesito contabilizar la cantidad de “Me Gusta”.*



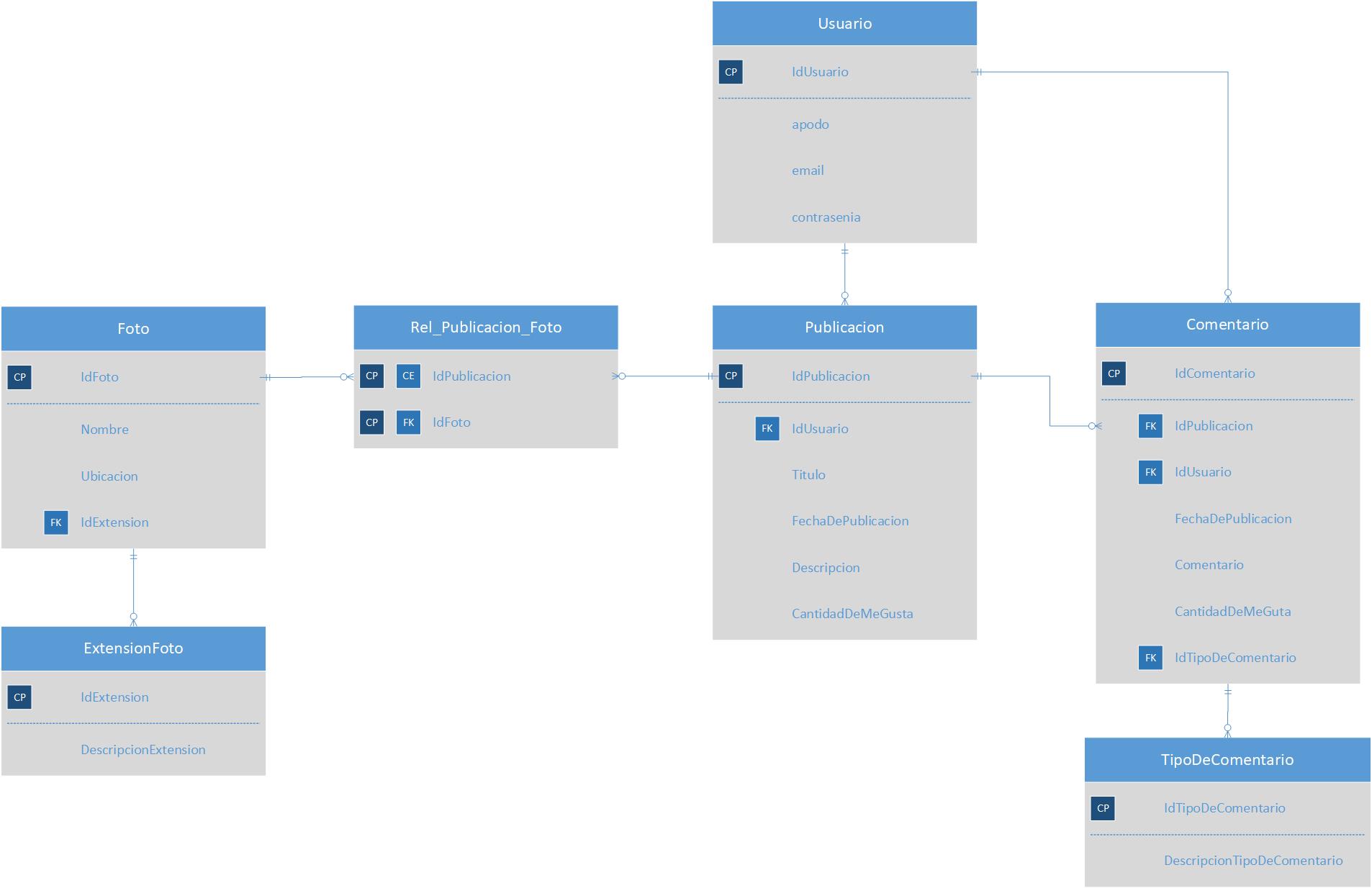
Inyección de dependencias

*Es posible encontrar el patrón de inyección de dependencias en las siguientes relaciones del diagrama de clases:*

* *La publicación y las fotos de la publicación.*
* *En un principio la red social sólo admite fotos en formato JPG. Sin embargo, si al tiempo quiero eliminar las fotos JPG y usar algún otro tipo de formato que tenga una mejor compresión y me ayude a ahorrar espacio, no puedo ya que tengo la clase FotoJPG en todas las publicaciones. Para solucionar este posible problema haría que las publicaciones conozcan la interfaz Fotos, desacoplando las implementaciones.*
* *La publicación y los comentarios de la publicación.*
* *En un caso similar al anterior, si yo quiero modificar mi reacción, por un reclamo o una sorpresa (que se implementará en una versión posterior), podría hacerlo ya que la publicación conoce a la interfaz comentario y sería simple modificar el objeto en tiempo de ejecución sin modificar la clase publicación ni sus instancias de la clase.*

Modelo de Datos

*Se presenta a continuación el modelo de datos diseñado para soportar el diagrama de clases de la aplicación propuesto:*

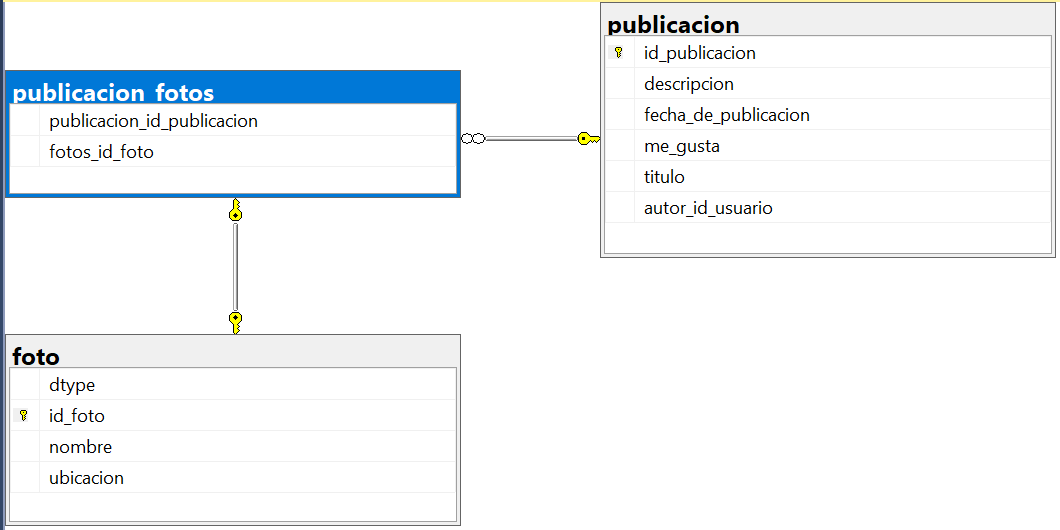
****

Log de cambios

*03/05/2021 – Cambio de relación entre las clases Publicación y Foto*

*Antes*

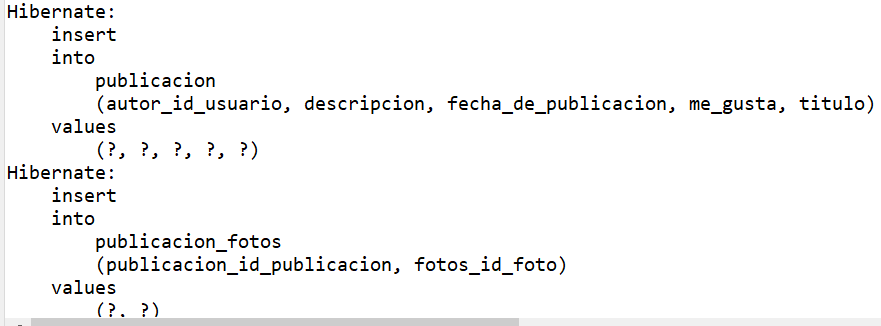
*El comportamiento deseado era que una misma foto pueda reutilizarse en varias publicaciones distintas. Para ello se le indicó a Spring Boot que en la base de datos genere y luego administre una relación como la siguiente:*

**

*El problema ocurre cuando hago el PUT de la publicación pasándole la lista de fotos (creadas previamente) desde el JSON:*



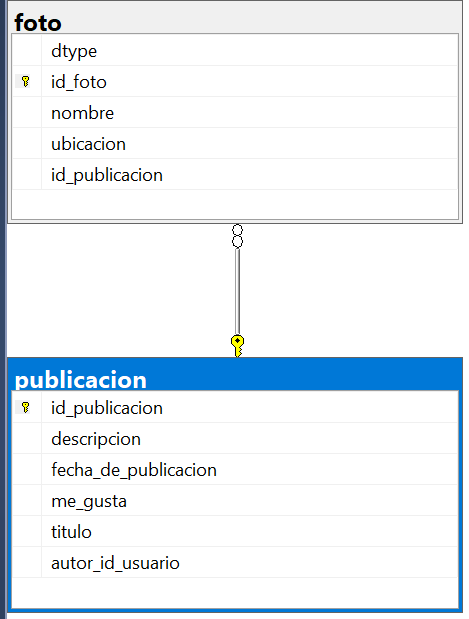
*Spring Boot primero intenta crear la publicación y luego generar la relación de la publicación con las fotos:*



*El problema es que al no commitear entre ambos inserts, la aplicación arroja el siguiente error: HHH000010: On release of batch it still contained JDBC statements.*

*Después*

*Cambié el comportamiento de la implementación para que cada foto esté fuertemente asociados a 1 publicación en lugar de poder reutilizar la misma foto en varias publicaciones:*

**

*Con esto cada vez que se carga una publicación con una o varias fotos asociadas desde el JSON, Spring Boot mediante Hibernate y JPA entienden que debe primero crear la publicación y luego crear y guardar la o las fotos asociadas.*

*Conclusión*

*La ventaja es que logro el comportamiento esperado, pero al hacer esto podría llegar a tener la misma foto duplicada ocupando espacio en el servidor.*

*24/05/2021 – Se hace que el campo fecha de publicación se genere automáticamente en lugar de tener que ser ingresado en el body del request.*

*Antes*

*El usuario ingresaba la fecha de publicación al crear o modificar una publicación.*

*Después*

*Se mantiene el campo fecha de publicación para cuando se crea la publicación y se agrega el campo fecha de última actualización para poder persistir el último cambio realizado. Tuve que agregar este campo adicional ya que las anotaciones* ***@CreationTimestamp*** *y* ***@UpdateTimestamp*** *se deben utilizar en 2 campos distintos.*