**Aplicación de precios**

El proyecto se centra en el desarrollo del backend de una plataforma similar a un foro, donde los usuarios pueden compartir y buscar precios favorables de productos que han encontrado en diferentes lugares. Al ingresar a la aplicación, los usuarios pueden ver un listado de los precios publicados. A partir de este listado, es posible buscar publicaciones relacionadas con un producto o lugar específico.

**Motivación**

Debido a la situacion economica y el bajo control de precios la gente empieza a recorrer distintos lugares con el fin de encontrar los prodcutos que busca al mejor precio posible. En pueblos y ciudades chicas se suelen crear grupos de Facebook o Whatsapp donde se comparte precios entre las personas con el fin de ayudarse entre todos. Si bien suele funcionar se puede volver tedioso tanto para los administradores como para los usuarios. La idea de la aplicación es centralizar todas las publicaciones en un solo lugar y dar la posibilidad de aplicar filtros de busqueda y poder valorar las publicaciones para corroborar la veracidad de las mismas.

**Surgimiento**

La idea del proyecto surge de aplicar lo aprendido a lo largo de un curso que realice sobre desarrollo de API’s con Spring Boot y poder utilizarlo como portafolio. Algunas de las técnologias y patrones aplicados a lo largo del proyecto fueron las siguientes:

* Java 21
* Framework Spring Boot v3.2.2
* Base de datos mysql con Wampserver64
* Manejo de relaciones con Spring Data JPA (ORM)
* Lombock para creacion de getters, setters y constructores.
* Arquitectura multicapas (model, controler, service, repository, etc)
* Patrón DTO para respuestas personalizadas y validaciones
* Inyeccion de dependencias e inversión de control
* Manejo de excepciones con codigos HTTP y mensajes personalizados

**Alguno detalles sobre la implementación**

**Diagrama de clases**

Antes de empezar a escribir codigo realice el diagrama de clases pensando en obtener todo lo minimo para que la aplicación funcione. Como resultado obtuve el siguiente diagrama UML.

**Paquetes/Estructura**

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

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Endpoints

**Modelo con relaciones**

Como son muchos modelos solo mostrare el de post que es el que mas relaciones tiene.

Texto

Descripción generada automáticamente

**Validaciones**

Para evitar inconsistencias en la base de datos se aplicaron validaciones a cada formulario recibido. 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 anotacion @RestControllerAdvice nos permite captar todas las interrupciones lanzadas por la aplicación en tiempo de ejecucion y tratar cada tipo de ellas de manera personalizada. En la siguiente imagen se puede ver como se tratan la 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 especificos 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

**Post**

Crear: /posts/crear | POST

Borrar: /posts/eliminar/id |DELETE

Editar: /posts/editar/id | UPDATE

Buscar: /posts/id | GET

Todos: /posts | GET

Posts de un usuario: /posts/usuario/id\_usuario | GET

Posts de un producto: /posts/producto/id\_prodcuto | GET

Posts de una categoria: /posts/categoria/id\_categoria **???**

Posts de un local: /posts/local/id\_local | GET

Buscar posts: /posts/búsqueda?atributo=””&buscar=”” | GET

**Usuarios**

Crear: /usuarios/crear | POST

Borrar: /usuarios/eliminar/id |DELETE

Editar: /usuarios/editar/id | UPDATE

Buscar: /usuarios/id | GET

Todos: /usuarios | GET

**Productos**

Crear: /productos/crear | POST

Borrar: /productos/eliminar/id |DELETE

Editar: /productos/editar/id | UPDATE

Buscar: /productos/id | GET

Todos: /productos | GET

**Categorías**

Crear: /categorias/crear | POST

Borrar: /categorias/eliminar/id |DELETE

Editar: /categorias/editar/id | UPDATE

Buscar: /categorias/id | GET

Todos: /categorias | GET

**Comentarios**

Crear: /comentarios/crear | POST

Borrar: /comentarios/eliminar/id |DELETE

Editar: /comentarios/editar/id | UPDATE

Buscar: /comentarios/id | GET

Todos: /comentarios | GET

Comentarios de un post: /comentarios/post/id\_post

**Locales**

Crear: /locals/crear | POST

Borrar: /locals/eliminar/id |DELETE

Editar: /locals/editar/id | UPDATE

Buscar: /locals/id | GET

Todos: /locals | GET