SocialMeli

Bootcamp TI wave 8

Bootcamp Backend Java
Desafío Spring

Índice

Contexto

Resumen

Comencemos entendiendo de qué trata el desafío Spring, al visualizar el crecimiento de Mercado Libre, se pretende implementar más herramientas que provean de más funcionalidades a los compradores y vendedores. Para que su experiencia sea más innovadora.

SocialMeli será llamada la herramienta que contiene estas nuevas funcionalidades.

En pocas palabras, este conjunto de nuevas herramientas y experiencias es una API, que proporciona información que podremos mostrar a los usuarios finales.

El beta de SocialMeli fue desarrollado con el framework Spring Boot y su lenguaje de programación respectivo Java.

Estructura del proyecto

Introducción

En primera instancia para facilitar el desarrollo de cada requerimiento, se pensó en la estructura completa que debía tener el proyecto, para esto nos guiamos por lo visto dentro del bootcamp, y lo mostraremos a continuación.

Carpetas

Package Principal: meli.desafio_spring

Contiene el proyecto en su totalidad.

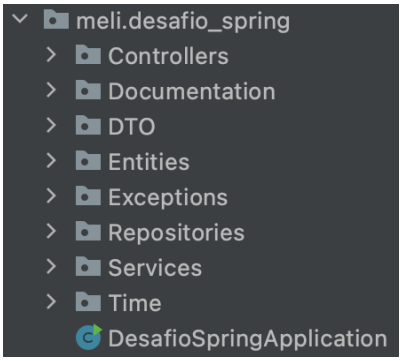

Figura 1: Estructura del proyecto

Controllers:

Consta de 2 clases las cuales contienen todos los endpoint vistos como requerimientos, para productos y usuarios, cada uno tiene un controlador.

Documentation:

Mantiene una clase que corresponde a la configuración de la librería Swagger para facilitar las pruebas de la API.

DTO:

Contenemos todas las respuestas que entregaremos a cada solicitud que se realice a los endpoint respectivos.

Entities:
Es el listado de cada modelo que deberíamos considerar, para trabajar los datos, se contienen las clases que se obtienen al interactuar con la DB(en este caso un archivo JSON creado de manera dinámica).

Exceptions:
Se presenta una clase que contiene distintos métodos para contener excepciones customizadas.

Repositories:
Aquí se mantienen las clases con sus interfaces respectivas que obtienen y escriben la data en la DB (archivos json).

Services:
Se separa en 2 packages, los cuales tienen relación directa con los controladores. Mantenemos un servicio para usuarios y otro para productos donde se mantiene gran parte de la lógica.

Carpeta Resource:
La carpeta resource está fuera del package principal, pero contiene los archivos PostList.json y UserList.json que son la DB utilizada. También contiene una carpeta llamada documentation donde está el código de cada diagrama y su imagen.

Diagramas

Diagrama de clases

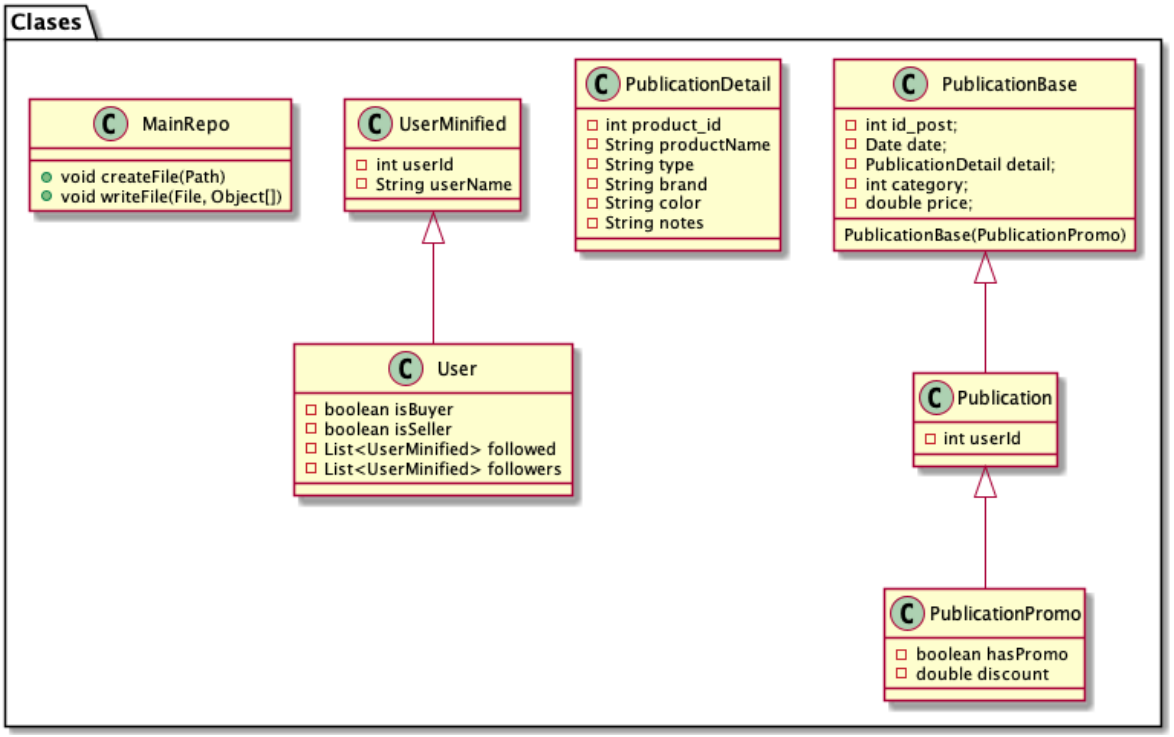

Figura 2: Diagrama de Clases

Diagrama de Controladores

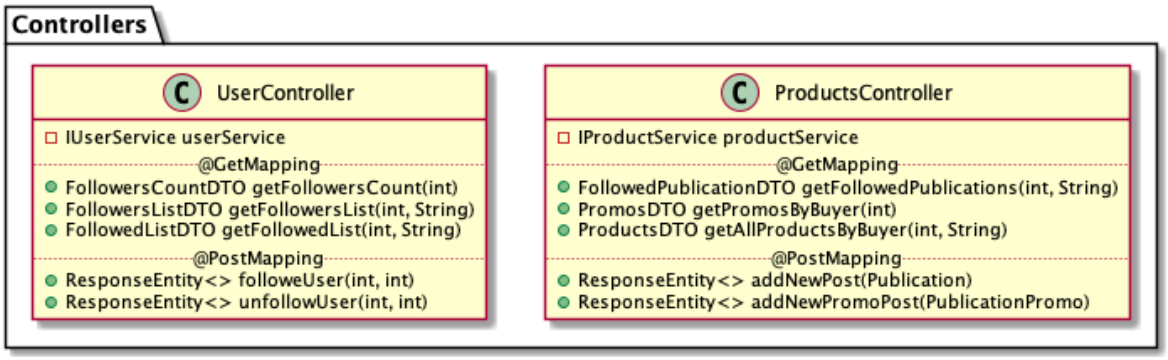

Figura 3: Diagrama de Controladores

Diagrama DTOs

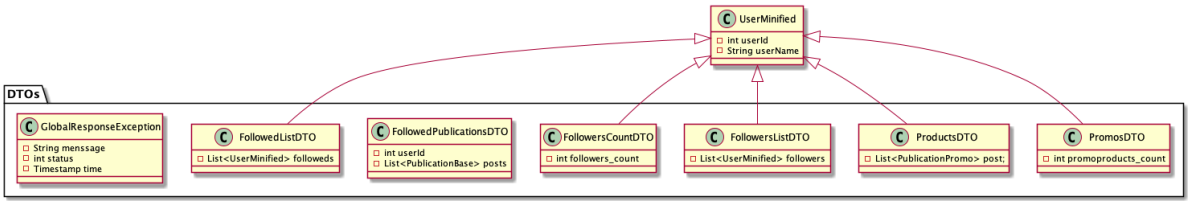

Figura 4: Diagrama de DTOs

Diagrama Repositorios

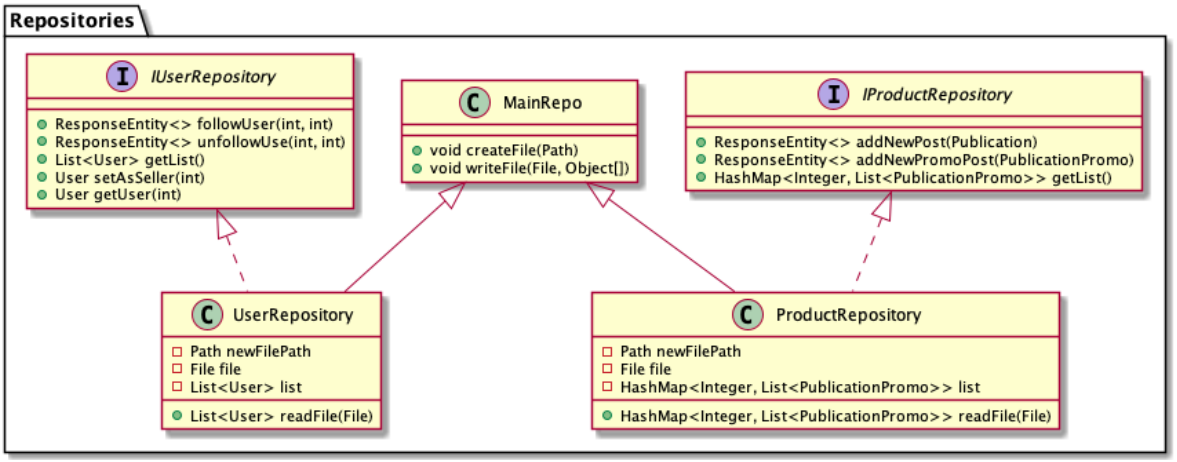

Figura 5: Diagrama de Repositorios

Diagrama Servicios

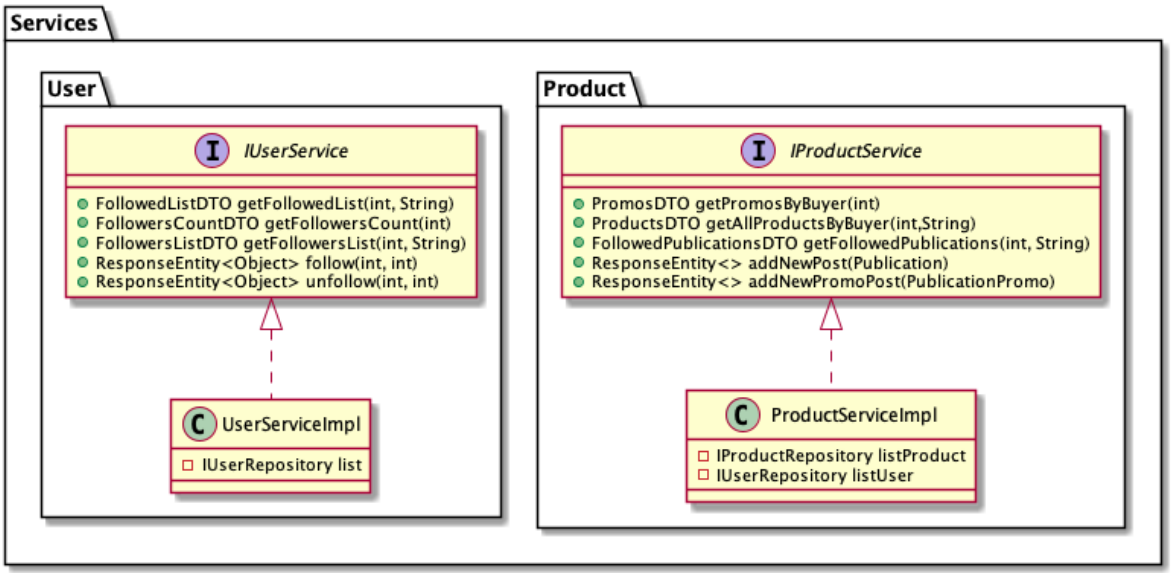

Figura 6: Diagrama de Servicios

Diagrama completo

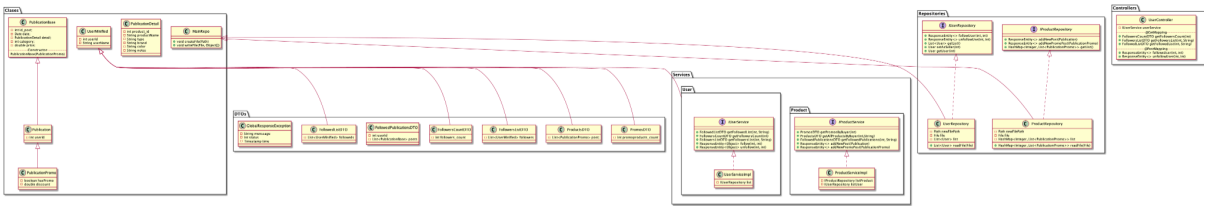

Figura 7: Diagrama Completo

Pruebas

Guia

Para llevar a cabo las pruebas y verificar que el proyecto funciona de manera correcta, debemos inicializarlo y dirigirnos a la ruta:

localhost:8080/swagger-ui.html

Esto nos redirigirá a la siguiente página:

Figura 8: Página de API documentada

*Swagger es una biblioteca que nos permite documentar la API que estemos desarrollando.

Desde este punto, es totalmente factible realizar cualquier tipo de prueba a los endpoints desarrollados.

Para hacer esto podemos hacer un click directamente en un controlador, el cual nos indicará los endpoint de ese controlador.

Figura 9: Selección de controlador

En este caso estamos visualizando los endpoint del controlador de usuarios. Podemos ingresar directamente al Endpoint que queramos probar, en este caso agregar un nuevo usuario (funcionalidad extra).

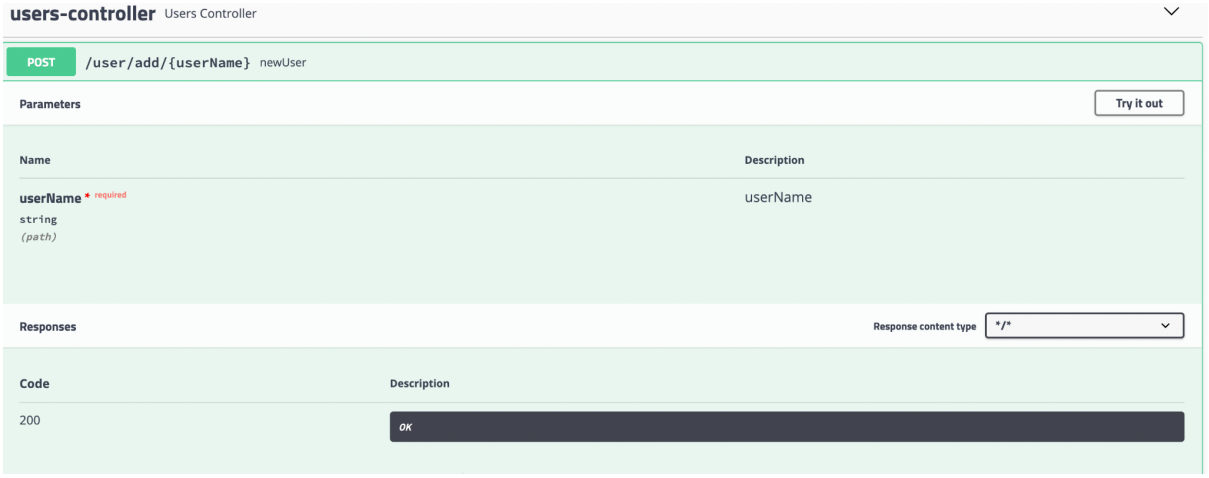

Figura 10: Selección de endpoint

En la parte superior existe un botón llamado Try it out, donde al clickear, nos muestra la siguiente ventana:

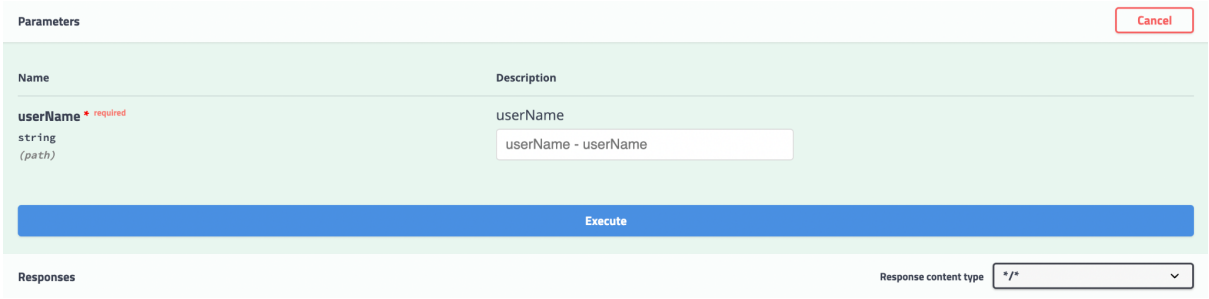

Figura 11: Prueba de endpoint

donde podremos agregar los parámetros respectivos y luego visualizar la respuesta en la parte inferior, mostrando el código que retorna y un json o cualquier otro tipo de respuesta en la descripción.

Pasos

Para verificar los endpoint, existen 2 usuarios en los archivos JSON (DB) con los cuales pueden probar directamente los Endpoint de listar seguidores,

seguidos o hacer follow y unfollow, debemos considerar también que un usuario debe seguir a otro para poder ver las publicaciones.

Como mejor práctica lo ideal es crear 2 usuarios a estos 2 usuarios hacer que suban un producto, para que se conviertan en vendedores y posteriormente podemos hacer follow, la interacción y prueba de los Endpoint es bastante didáctica y entretenida.

Espero se entretengan realizando pruebas de este desafío.

Conclusión

En primera instancia fue bastante complejo la estructuración del proyecto pero al cabo de tener armado los cimientos esto facilitó mucho el trabajo posterior, pudimos aplicar absolutamente todos los conceptos comprendidos durante estas semanas de bootcamp.

Algunos temas que me gustaría comentar, que me fueron de bastante ayuda para realizar este desafío es el uso de herencia, interfaces e inyección de dependencias. A mi parecer esto amenizó en gran medida lo que debíamos realizar.

Como nueva funcionalidad, se agregó un endpoint para crear usuarios, el cual será visible al momento de visualizar la documentación de la API.

En síntesis, el proyecto fue culminado con funcionalidades extra, con una estructura y funcionalidad prolija.

Muchas gracias.
Felipe Andrade Valenzuela