Proyecto SocialMeli 2

ÍNDICE

El proyecto consta de compradores van a poder seguir a sus vendedores favoritos y enterarse de todas las novedades que los mismos posteen.

Se plantea creación de una API Rest que permita:

1. Poder realizar la acción de “Follow” (seguir) a un determinado usuario.
2. Obtener el resultado de la cantidad de usuarios que siguen a un determinado **vendedor.**
3. Obtener un listado de todos los usuarios que siguen a un determinado **vendedor** (¿Quién me sigue?).
4. Obtener un listado de todos los vendedores a los cuales sigue un determinado usuario (¿A quién sigo?).
5. Dar de alta una nueva publicación.
6. Obtener un listado de las publicaciones realizadas en las últimas dos semanas, por los vendedores que un usuario sigue (para esto tener en cuenta ordenamiento por fecha, publicaciones más recientes primero).
7. Poder realizar la acción de “Unfollow” (dejar de seguir) a un determinado vendedor.
8. Alfabético Ascendente y Descendente.
9. Fecha Ascendente y Descendente.
10. Llevar a cabo la publicación de un nuevo producto en promoción.
11. Obtener la cantidad de productos en promoción de un determinado vendedor.

Mejoras del proyecto SocialMeli

12. —

13. Buscar un producto por nombre o por marca. Opcionalmente, se puede pasar un user id para especificar más la búsqueda.

14. Obtener un listado de las publicaciones realizadas por los vendedores según las fechas ingresadas.

15. Activar la promoción o modificar el porcentaje de descuento.

16. Eliminar una publicación.

17. Obtener un listado de todos los posts de un determinado vendedor. Con la opción de filtrar los posts de acuerdo a si tienen descuento o no tienen descuento.

Test Unitarios

se solicita una serie de test unitarios a llevar a cabo

T-0001 Verificar que el usuario a seguir exista. **(US-0001)**

T-0002 Verificar que el usuario a dejar de seguir exista. **(US-0007)**

T-0003 Verificar que el tipo de ordenamiento alfabético exista **(US-0008)**

T-0004 Verificar el correcto ordenamiento ascendente y descendente por nombre. **(US-0008)**

T-0005 Verificar que el tipo de ordenamiento por fecha exista **(US-0009)**

T-0006 Verificar el correcto ordenamiento ascendente y descendente por fecha. **(US-0009)**

T-0007 Verificar que la cantidad de seguidores de un determinado usuario sea correcta. **(US-0002)**

T-0008 Verificar que la consulta de publicaciones realizadas en las últimas dos semanas de un determinado vendedor sean efectivamente de las últimas dos semanas. **(US-0006)**

	Situaciones de entrada	Comportamiento Esperado
T-0001	Verificar que el usuario a seguir exista. **(US-0001)**	**Se cumple:** Permite continuar con normalidad. **No se cumple:** Notifica la no existencia mediante una excepción.
T-0002	Verificar que el usuario a dejar de seguir exista. **(US-0007)**	**Se cumple:** Permite continuar con normalidad. **No se cumple:**

		Notifica la no existencia mediante una excepción.
T-0003	Verificar que el tipo de ordenamiento alfabético exista **(US-0008)**	**Se cumple:** Permite continuar con normalidad. **No se cumple:** Notifica la no existencia mediante una excepción.
T-0004	Verificar el correcto ordenamiento ascendente y descendente por nombre. **(US-0008)**	Devuelve la lista ordenada según el criterio solicitado
T-0005	Verificar que el tipo de ordenamiento por fecha exista **(US-0009)**	**Se cumple:** Permite continuar con normalidad. **No se cumple:** Notifica la no existencia mediante una excepción.
T-0006	Verificar el correcto ordenamiento ascendente y descendente por fecha. **(US-0009)**	Verificar el correcto ordenamiento ascendente y descendente por fecha. **(US-0009)**
T-0007	Verificar que la cantidad de seguidores de un determinado usuario sea correcta. **(US-0002)**	Devuelve el cálculo correcto del total de la cantidad de seguidores que posee un usuario.
T-0008	Verificar que la consulta de publicaciones realizadas en las últimas dos semanas de un determinado vendedor sean efectivamente de las últimas dos semanas. **(US-0006)**	Devuelve únicamente los datos de las publicaciones que tengan fecha de publicación dentro de las últimas dos semanas a partir del día de la fecha.

Test de Integración

Bonus

Fernando Nicolas Baldrich

TB-03 - Verificar el correcto funcionamiento del endpoint para obtener los seguidores de un vendedor
INTEGRATION US 10 - Verificar el correcto funcionamiento del endpoint para crear posts de productos en promoción
INTEGRATION US 13 - Verificar el correcto funcionamiento del buscador de posts

Matias Gregorat

INTEGRATION US 15 - Verificar el correcto funcionamiento de la activacion de la promo.
Falla con usuario inexistente
Falla con Post inexistente
Falla con todos los parámetros nulos
Falla con todos los parámetros negativos

María Emilia Lascano

INTEGRATION US 17
Happy Path: Get promo posts history
Happy Path: Get promo posts history with promo
Happy Path: Get promo posts history without promo

Sad Path: User ID no tiene promos
Sad Path: User ID no existe

Delfina Brenda Glavas

INTEGRATION US 4 - FIND BY FOLLOWED

Sad Path: El usuario ingresado no sigue a nadie.
Sad Path: User ID no existe.
Sad Path: Order ingresado inválido.

INTEGRATION US 9 - GET RECENT POST FROM FOLLOWED USERS

Sad Path: Order ingresado inválido.

INTEGRATION US 16 - DELETE POST

Happy Path: Eliminar post con usuario y post existentes.
Sad Path: User ID no existe.
Sad Path: Post ID no existe.

Stephanie Castillo

INTEGRATION US 01: Poder realizar la acción de "Follow" (Seguir) a un determinado vendedor.

Coverage

Element	Class, %	Method, %	Line, %	Branch, %
ar.com.mercadolibre.socialmeli	83% (10/12)	94% (64/68)	94% (288/305)	83% (115/138)
dto	0% (0/1)	0% (0/1)	0% (0/1)	100% (0/0)
SocialmeliApplication	0% (0/1)	0% (0/1)	0% (0/1)	100% (0/0)
exception	100% (3/3)	77% (7/9)	84% (22/26)	100% (0/0)
entity	100% (1/1)	100% (4/4)	100% (8/8)	100% (0/0)
utils	100% (1/1)	100% (5/5)	100% (35/35)	100% (0/0)
repository	100% (1/1)	100% (11/11)	96% (32/33)	75% (3/4)
controller	100% (2/2)	100% (16/16)	100% (19/19)	100% (0/0)
service	100% (2/2)	100% (21/21)	94% (172/182)	83% (112/134)

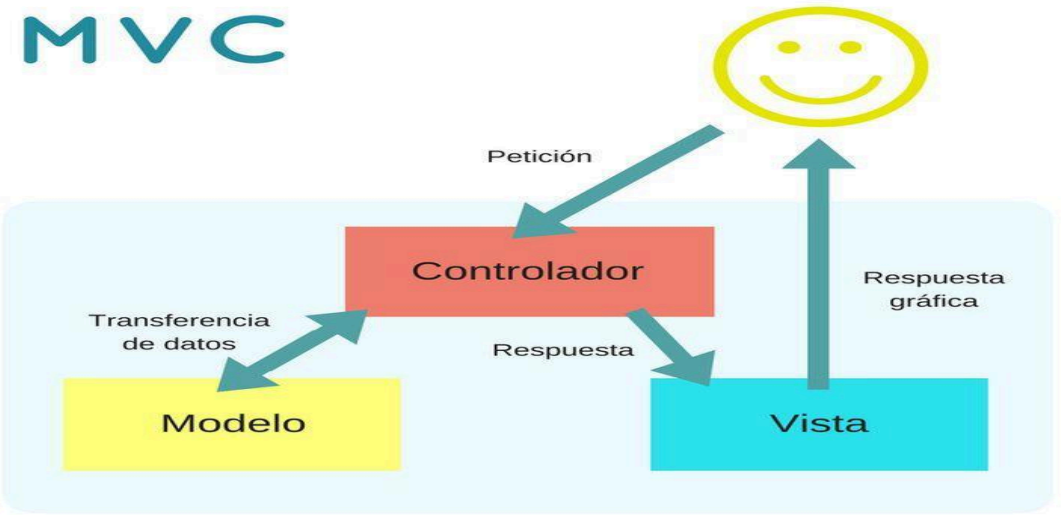
Tecnologías utilizadas

- **Lenguaje de programación:**
  - **JAVA (v 21).**
- **Framework:**
  - **Spring Framework (Spring Boot).**
  - **Maven**

- **RESTful API.**
- **Herramienta de integración continua:**
  - **GitHub**

Arquitectura del sistema

Se utilizó la arquitectura MVC o Modelo-Vista-Controlador es un patrón de arquitectura de software que, utilizando 3 componentes (Vistas, Models y Controladores) separa la lógica de la aplicación de la lógica de la vista en una aplicación.



Requisitos del sistema

- **Software:**
  - **JDK 21**
- **Dependencias:**
  - **Lombok**
  - **Spring boot starter WEB**
  - **Spring boot DevTools**
  - **Spring boot Starter test**
  - **Model Mapper**
  - **Jackson DataType JSR-310**

Cronograma y etapas de desarrollo:

Se puede visualizar el cronograma en el trello de integración del proyecto.

Requerimientos técnicos funcionales

A. Requerimientos Iniciales

US 0001: Poder realizar la acción de “Follow” (seguir) a un determinado vendedor

Sign:

Method	SIGN
POST	/users/{userId}/follow/{userIdToFollow}
Ejemplo: /users/123/follow/234	
Response	Status Code 200 (todo OK) - bodyless or dto Status Code 400 (Bad Request) - bodyless or dto

Filtros/Parámetros:

Parámetros	Tipo	Descripción/Ejemplo
userId	int	Número que identifica al usuario actual
userIdToFollow	int	Número que identifica al usuario a seguir

US 0002: Obtener el resultado de la cantidad de usuarios que siguen a un determinado vendedor

Sign:

Method	SIGN
GET	/users/{userId}/followers/count
Ejemplo: /users/234/followers/count/	

Response	{
	"user_id": 234,
	"user_name": "vendedor1",
	"followers_count": 35
	}

Filtros/Parámetros:

Parámetros	Tipo	Descripción/Ejemplo
userId	int	Número que identifica a cada usuario

US 0003: Obtener un listado de todos los usuarios que siguen a un determinado vendedor (¿Quién me sigue?)

Sign:

Method	SIGN
GET	/users/{userId}/followers/list
Ejemplo: /users/234/followers/list	

Response	{
	"user_id": 234,
	"user_name": "vendedor1",
	"followers": [
	{
	"user_id": 4698,
	"user_name": "usuario1"
	},
	{
	"user_id": 1536,
	"user_name": "usuario2"
	},
	{
	"user_id": 2236,
	"user_name": "usuario3"
	}
	]
	}

Filtros/Parámetros:

Parámetros	Tipo	Descripción/Ejemplo
userId	int	Número que identifica a cada usuario

US 0004: Obtener un listado de todos los vendedores a los cuales sigue un determinado usuario (¿A quién sigo?)

Sign:

Method	SIGN
GET	/users/{userId}/followed/list
Ejemplo: /users/4698/followed/list	

Response	{
	"user_id": 4698,
	"user_name": "usuario1",
	"followed": [
	{
	"user_id": 234,
	"user_name": "vendedor1"
	},
	{
	"user_id": 6932,
	"user_name": "vendedor2"
	},
	{
	"user_id": 6631,
	"user_name": "vendedor3"
	}
	]
	}

Filtros/Parámetros:

Parámetros	Tipo	Descripción/Ejemplo
userId	int	Número que identifica a cada usuario

US 0005: Dar de alta una nueva publicación

Sign:

Method	SIGN
POST	/products/post

PAYLOAD:	{
	"user_id": 123,
	"date": "29-04-2021",
	"product": {
	"product_id": 1,
	"product_name": "Silla Gamer",
	"type": "Gamer",
	"brand": "Racer",
	"color": "Red & Black",
	"notes": "Special Edition"
	},
	"category": 100,
	"price": 1500.50
	}

RESPONSE	Status Code 200 (todo OK) Status Code 400 (Bad Request)

Filtros/Parámetros:

Parámetros	Tipo	Descripción/Ejemplo
user_id	int	Número que identifica a cada usuario
date	LocalDate	Fecha de la publicación en formato dd-MM-yyyy
product_id	int	Número identificatorio de un producto asociado a una publicación
product_name	String	Cadena de caracteres que representa el nombre de un producto
type	String	Cadena de caracteres que representa el tipo de un producto
brand	String	Cadena de caracteres que representa la marca de un producto

color	String	Cadena de caracteres que representa el color de un producto
notes	String	Cadena de caracteres para colocar notas u observaciones de un producto
category	int	Identificador que sirve para conocer la categoría a la que pertenece un producto. Por ejemplo: 100: Sillas, 58: Teclados
price	double	Precio del producto

US 0006: Obtener un listado de las publicaciones realizadas por los vendedores que un usuario sigue en las últimas dos semanas (para esto tener en cuenta ordenamiento por fecha, publicaciones más recientes primero).

Sign:

Method	SIGN
GET	/products/followed/{userId}/list
Ejemplo: /products/followed/4698/list	
RESPONSE:	{ "user_id": 4698, "posts": [{ “user_id”: 123,El "post_id": 32, "date": "01-05-2021", "product": { "product_id": 62, "product_name": "Headset RGB Inalámbrico", "type": "Gamer", "brand": "Razer", "color": "Green with RGB", "notes": "Sin Batería" }, "category": 120, "price": 2800.69 }, { “user_id”: 234, "post_id": 18, "date": "29-04-2021", "product": { "product_id": 1, "productName": "Silla Gamer", "type": "Gamer", "brand": "Racer", "color": "Red & Black", "notes": "Special Edition" }, "category": 100, "price": 15000.50

	}] }
RESPONSE	Status Code 200 (todo OK) Status Code 400 (Bad Request)

Filtros/Parámetros:

Parámetros	Tipo	Descripción/Ejemplo
userId	int	Número que identifica a cada usuario

US 0007: Poder realizar la acción de “Unfollow” (dejar de seguir) a un determinado vendedor.

Sign:

Method	**SIGN**
POST	/users/{userId}/unfollow/{userIdToUnfollow}
Ejemplo: /users/234/unfollow/123	
RESPONSE	Status Code 200 (todo OK) Status Code 400 (Bad Request)

Filtros/Parámetros:

Parámetros	**Tipo**	**Descripción/Ejemplo**
userId	int	Número que identifica al usuario actual
userIdToUnfollow	int	Número que identifica al usuario a dejar de seguir

US 0008: Ordenamiento alfabético ascendente y descendente

Sign:

Method	SIGN
GET	**Ejemplos:** /users/{UserID}/followers/list?order=name_asc /users/{UserID}/followers/list?order=name_desc /users/{UserID}/followed/list?order=name_asc /users/{UserID}/followed/list?order=name_desc

order	Description
name_asc	Alfabético ascendente.
name_desc	Alfabético descendente.

*__Nota:__ Este ordenamiento aplica solo para US-003 y US-004.

US 0009: Ordenamiento por fecha ascendente y descendente

Sign:

Method	SIGN
GET	**Ejemplos:** /products/followed/{userId}/list?order=date_asc /products/followed/{userId}/list?order=date_desc

order	Description
date_asc	Fecha ascendente (de más antigua a más nueva)
date_desc	Fecha descendente (de más nueva a más antigua)

*__Nota:__ Este ordenamiento aplica solo para la US-006

US 0010: Llevar a cabo la publicación de un nuevo producto en promoción

Sign:

Method	SIGN
POST	/products/promo-post

PAYLOAD:	{
	"user_id": 234,
	"date": "29-04-2021",
	"product": {
	"product_id": 1,
	"product_name": "Silla Gamer",
	"type": "Gamer",
	"brand": "Racer",
	"color": "Red & Black",
	"notes": "Special Edition"
	},
	"category": 100,
	"price": 1500.50,
	"has_promo": true,
	"discount": 0.25
	}

Response	Status Code 200 (OK) Status Code 400 (Bad request)

Filtros/Parámetros:

Parámetros	Tipo	Descripción/Ejemplo
user_id	int	Número que identifica a cada usuario
date	LocalDate	Fecha de la publicación en formato dd-MM-yyyy
product_id	int	Número identificatorio de un producto asociado a una publicación
product_name	String	Cadena de caracteres que representa el nombre de un producto
type	String	Cadena de caracteres que representa el tipo de un producto
brand	String	Cadena de caracteres que representa la marca de un

		producto
color	String	Cadena de caracteres que representa el color de un producto
notes	String	Cadena de caracteres para colocar notas u observaciones de un producto
category	int	Identificador que sirve para conocer la categoría a la que pertenece un producto. Por ejemplo: 100: Sillas, 58: Teclados
price	double	Precio del producto
has_promo	boolean	Campo true o false para determinar si un producto está en promoción o no
discount	double	En caso de que un producto estuviese en promoción ,establece el monto de descuento.

US 0011: Obtener la cantidad de productos en promoción de un determinado vendedor

Sign:

Method	SIGN
GET	/products/promo-post/count?user_id={userId}

Response	{
	"user_id" : 234,
	"user_name": "vendedor1",
	"promo_products_count": 23
	}

Filtros/Parámetros:

Parámetros	Tipo	Descripción/Ejemplo
user_id	int	Número que identifica a cada usuario
user_name	String	Cadena de caracteres que representa el nombre del usuario
promo_products_count	int	Cantidad numérica de productos en promoción de un determinado usuario.

US 0013: Buscar un producto por nombre o por marca. Opcionalmente, se puede pasar un user id para especificar más la búsqueda.

Sign:

Method	SIGN
GET	localhost:8080/products/search?query=ams localhost:8080/products/search?query=ams&user_id=2

RESPONSE:	[
	{
	"post_id": 3,
	"user_id": 2,
	"product": {
	"type": "Monitor",
	"brand": "Samsung",
	"color": "Negro",
	"notes": "Ultra HD",
	"product_id": 3,
	"product_name": "Monitor 4K"
	},
	"date": "18-09-2024",
	"category": 300,
	"price": 30000.0,
	"discount": 0.3,
	"has_promo": true
	}
	]

RESPONSE	Status Code 200 (todo OK) Status Code 404 (No se encontró al usuario con el user_id)

Parámetros del Response:

Parámetros	Tipo	Descripción/Ejemplo
user_id	int	Número que identifica a cada usuario
post_id	int	Número identificatorio de cada una de las publicaciones

date	LocalDate	Fecha de la publicación en formato dd-MM-yyyy
product_id	int	Número identificatorio de un producto asociado a una publicación
product_name	String	Cadena de caracteres que representa el nombre de un producto
type	String	Cadena de caracteres que representa el tipo de un producto
brand	String	Cadena de caracteres que representa la marca de un producto
color	String	Cadena de caracteres que representa el color de un producto
notes	String	Cadena de caracteres para colocar notas u observaciones de un producto
category	int	Identificador que sirve para conocer la categoría a la que pertenece un producto. Por ejemplo: 100: Sillas, 58: Teclados
price	double	Precio del producto
has_promo	boolean	Campo true o false para determinar si un producto está en promoción o no
discount	double	En caso de que un producto estuviese en promoción, establece el monto de descuento.

US 0014: Obtener un listado de las publicaciones realizadas por los vendedores según las fechas ingresadas.

Sign:

Method	SIGN
GET	/products/search/date?date_start={date_start]&date_end={date_end}
Ejemplo:	/products/search/date?date_start=16/09/2021&date_end=

RESPONSE	[
	{
	"user_id": 2,
	"post_id": 1,
	"date": "16-09-2021",
	"product": {
	"type": "Gamer",
	"brand": "Racer",
	"color": "Red",
	"notes": "Special Edition",
	"product_id": 1,
	"product_name": "Silla gamer"
	},
	"category": 100,
	"price": 15000.0
	}
	]

Filtros/Parámetros:

Parámetros	Tipo	Descripción/Ejemplo
date_start	Local Date	Fecha que marca el inicio temporal del cúal el usuario quiere empezar la búsqueda de posteos de los vendedores. Formato `dd-MM-yyyy.`
date_end	Local Date	Fecha que marca el fin temporal del cúal el usuario quiere finalizar la búsqueda de posteos de los vendedores. (En caso de que el usuario no ingrese este dato se tomará como fin de búsqueda la fecha actual). Formato `dd-MM-yyyy` .

US 0015: Activar la promoción o modificar el porcentaje de descuento.

Sign:

Method	SIGN
PUT	/products/posts/activate-promo

PAYLOAD:	{
	"user_id": 4,
	"post_id": 2,
	"discount": 0.2
	}

RESPONSE	Status Code 200 (todo OK) Status Code 400 (Bad Request) Status Code 404 (Not Found)

Filtros/Parámetros:

Parámetros	Tipo	Descripción/Ejemplo
user_id	int	Número que identifica a cada usuario
post_id	int	Número identificatorio de una publicación
discount	double	Porcentaje de descuento

US 0016: Eliminar una publicación.

Sign:

Method	SIGN
PUT	/products/post/{userId}/{postId}
RESPONSE	Status Code 200 (todo OK) Status Code 404 (Not Found)

Filtros/Parámetros:

Parámetros	Tipo	Descripción/Ejemplo
userId	int	Número que identifica a cada usuario
postId	int	Número identificatorio de una publicación asociado a una lista de publicaciones en un usuario.

US 0017: Obtener un listado de todos los posts de un determinado vendedor. Con la opción de filtrar los posts de acuerdo a si tienen descuento o no tienen descuento.

Sign:

Method	SIGN
GET	/products/promo-post/{userId}/history /products/promo-post/{userId}/history?with_promo=true /products/promo-post/{userId}/history?with_promo=false
RESPONSE:	{ "user_id": 234, "user_name": "vendedor1", "posts": [{ "post_id": 18, "date": "29-04-2021", "product": { "product_id": 1, "product_name": "Silla Gamer", "type": "Gamer", "brand": "Racer", "color": "Red & Black", "notes": "Special Edition" }, "category": "100", "price": 15000.50, "has_promo": true, "discount": 0.25 }, { “user_id”: 234 "post_id": 32, "date": "01-05-2021", "product": { "product_id": 62, "product_name": "Headset RGB Inalámbrico", "type": "Gamer", "brand": "Racer", "color": "Green with RGB", "notes": "Sin Batería" },

	"category": "120", "price": 2800.69, "has_promo": false, "discount": 0.0 }] }
RESPONSE	Status Code 200 (todo OK) Status Code 400 (Bad Request)

Filtros/Parámetros:

Parámetros	Tipo	Descripción/Ejemplo
user_id	int	Número que identifica a cada usuario
user_name	String	Cadena de caracteres que representa el nombre del usuario
post_id	int	Número identificatorio de cada una de las publicaciones
date	LocalDate	Fecha de la publicación en formato dd-MM-yyyy
product_id	int	Número identificatorio de un producto asociado a una publicación
product_name	String	Cadena de caracteres que representa el nombre de un producto
type	String	Cadena de caracteres que representa el tipo de un producto
brand	String	Cadena de caracteres que representa la marca de un producto
color	String	Cadena de caracteres que representa el color de un producto
notes	String	Cadena de caracteres para colocar notas u observaciones de un producto
category	int	Identificador que sirve para conocer la categoría a la que pertenece un producto. Por ejemplo: 100: Sillas, 58:

		Teclados
price	double	Precio del producto
has_promo	boolean	Campo true o false para determinar si un producto está en promoción o no
discount	double	En caso de que un producto estuviese en promoción, establece el monto de descuento.

Observaciones:

Durante la implementación del proyecto todos los integrantes tomaron los diversos roles para el correcto uso de las herramientas. En la trazabilidad de los PR se podrán observar que en algunas ocasiones un desarrollador hacia merge sobre su propio PR a la rama develop esto se produjo en los momentos que el grupo se encontraba en reunión de equipo y desarrollo grupal.