Sprint I

MARINA SANTISO FILIPPI, Bootcamp Backend W13

Tema

Módulo 8, Sprint I

Índice

Sistema

El sistema contará con usuarios vendedores y compradores, donde un comprador también puede ser un vendedor dada la naturaleza de MercadoLibre, por ende, tanto vendedor como comprador se pueden seguir mutuamente.
El **comprador** que sigue a un vendedor es llamado **follower** y, por otro lado, el **vendedor** que es seguido por un comprador se denomina **followed**.

En mi sistema, los usuarios son representados a través de un solo modelo “Usuario” el cual cuenta con una lista de followers y una lista de followed, por ende, no hubo una separación estricta entre ambos tipos de usuarios, sino que maneje todo a través de la entidad nombrada anteriormente.

Desde el punto de vista de las publicaciones, tomé la decisión de que las publicaciones en promoción y las publicaciones comunes son parte del mismo modelo “Publicación”, por ende al listar las publicaciones de los followed de un usuario (US 0006) vendrán todas. Por otro lado, las publicaciones con productos en promoción no pueden ser cargadas si el booleano que determina si está en promoción o no está en falso o si el descuento no es mayor a 0, dado que si cualquiera de las dos se cumple, no sería una publicación con productos en promoción, sería solo una publicación y para este caso ya existe un método para insertar dichas publicaciones.

US 0001

El sistema debe ser capaz de realizar la acción de que un comprador pueda seguir (**follow**) a un vendedor.

Método

Método	POST
Firma	/users/{user_id}/follow/{user_id_to_follow}
Ejemplo de firma	/users/1235/follow/1569
Response OK	Status Code 200
Response Bad Request	Status Code 400

Parámetros a utilizar

Parámetro	Tipo de dato	Descripción
user_id	int	Número que identifica al usuario actual
user_id_to_follow	int	Número que identifica al usuario a seguir

Lógica de negocio controlada a través de Bad Request:

- ➔ Se controla que el usuario follower exista.
- ➔ Se controla que el usuario followed exista.
- ➔ Se controla que el usuario follower no esté siguiendo actualmente al usuario followed.
- ➔ Se controla que el usuario follower y followed no sean el mismo usuario.

US 0002

El sistema debe ser capaz de obtener la cantidad de usuarios que siguen a un determinado usuario.

Método

Método	GET
Firma	/users/{user_id}/followers/count
Ejemplo de firma	/users/1569/followers/count/

Response	{
	"user_id": 1569,
	"user_name": "vendedor1",
	"followers_count": 35
	}

Parámetros a utilizar

Parámetro	Tipo de dato	Descripción
user_id	int	Número que identifica al usuario actual
user_name	String	Nombre de usuario asociado al user_id

Lógica de negocio controlada a través de Bad Request:

- ➔ Se controla que el usuario exista.
- ➔ Se controla que el usuario tenga el contador de seguidores diferente de 0.

US 0003

El sistema debe ser capaz de obtener un listado de todos los usuarios followers que tiene determinado usuario.

Método

Método	GET
Firma	/users/{user_id}/followers/list
Ejemplo de firma	/users/1569/followers/list

Response	{
	"user_id": 1569,
	"user_name": "vendedor1",
	"followers": [
	{
	"user_id": 4698,
	"user_name": "usuario1"
	},
	{
	"user_id": 1536,
	"user_name": "usuario2"
	},
	{
	"user_id": 2236,
	"user_name": "usuario3"
	}
	]
	}

Parámetros a utilizar

Parámetro	Tipo de dato	Descripción
user_id	int	Número que identifica al usuario actual
user_name	String	Nombre de usuario asociado al user_id

Lógica de negocio controlada a través de Bad Request:

- ➔ Se controla que el usuario exista.
- ➔ Se controla que la lista de followers no retorne vacía.

US 0004

El sistema debe ser capaz de obtener un listado de todos los usuarios followed que tiene determinado usuario.

Método

Método	GET
Firma	/users/{user_id}/followed/list
Ejemplo de firma	/users/4698/followed/list

Response	{
	"user_id": 4698,
	"user_name": "usuario1",
	"followed": [
	{
	"user_id": 1569,
	"user_name": "vendedor1"
	},
	{
	"user_id": 6932,
	"user_name": "vendedor2"
	},
	{
	"user_id": 6631,
	"user_name": "vendedor3"
	}
	]
	}

Parámetros a utilizar

Parámetro	Tipo de dato	Descripción
user_id	int	Número que identifica al usuario actual
user_name	String	Nombre de usuario asociado al user_id

Lógica de negocio controlada a través de Bad Request:

- ➔ Se controla que el usuario exista.
- ➔ Se controla que la lista de followed no retorne vacía.

US 0005

El sistema debe ser capaz de dar de alta una nueva publicación.

Método

Método	POST
Firma	/products/post

Payload	{
	"user_id": 1235,
	"id_post": 18,
	"date": "29-04-2021",
	"detail": {
	"product_id": 1,
	"product_name": "Silla Gamer",
	"type": "Gamer",
	"brand": "Racer",
	"color": "Red & Black",
	"notes": "Special Edition"
	},
	"category": 100,
	"price": 1500.50
	}

Response OK	Status Code 200
Response Bad Request	Status Code 400

Parámetros a utilizar

Parámetro	Tipo de dato	Descripción
user_id	int	Número que identifica al usuario
id_post	int	Número que identifica a la publicación
date	LocalDate	Fecha de publicación en formato dd-MM-yyyy
product_id	int	Número que identifica a los productos pertenecientes a las publicaciones
product_name	String	String que representa el nombre del producto
type	String	String que representa el tipo del producto

brand	String	String que representa la marca del producto
color	String	String que representa el color del producto
notes	String	String que representa las observaciones del producto
category	int	Número que representa la categoría del producto
price	double	Número decimal que representa el precio del producto

Lógica de negocio controlada a través de Bad Request:

- ➔ Se controla que el usuario exista.
- ➔ Se controla que la publicación todavía no exista.

US 0006

El sistema debe retornar un listado de las publicaciones realizadas por los followeds de un usuario en particular, dichas publicaciones deben estar ordenadas de más reciente a menos reciente y **deben haber sido hechas en las últimas dos semanas**.

Método

Método	GET
Firma	/products/followed/{user_id}/list
Ejemplo de firma	/products/followed/4698/list

Response	{
	"user_id": 4698,
	"posts": [{
	"id_post": 32,
	"date": "01-05-2021",
	"detail": {
	"product_id": 62,
	"product_name": "Headset RGB Inalámbrico",
	"type": "Gamer",
	"brand": "Razer",
	"color": "Green with RGB",
	"notes": "Sin Batería"
	},
	"category": 120,
	"price": 2800.50
	}, {
	"id_post": 18,
	"date": "29-04-2021",
	"detail": {
	"product_id": 1,
	"product_name": "Silla Gamer",
	"type": "Gamer",
	"brand": "Racer",
	"color": "Red & Black",
	"notes": "Special Edition"
	},
	"category": 100,
	"price": 15000.50
	}
	]
	}

Parámetros a utilizar

Parámetro	Tipo de dato	Descripción
user_id	int	Número que identifica al usuario actual

Lógica de negocio controlada a través de Bad Request:

- ➔ Se controla que el usuario exista.
- ➔ Se controla que la lista de publicaciones de usuarios followed no esté vacía.
- ➔ Se controla que la lista de publicaciones de usuarios followed no esté vacía a raíz del filtro de fechas.
- ➔ Se controla que el usuario tenga followed (utiliza la excepción de la US 0004).

US 0007

El sistema debe ser capaz de realizar la acción de unfollow (dejar de seguir) a un determinado usuario, siendo otro usuario.

Método

Método	POST
Firma	/users/{user_id}/unfollow/{user_id_to_unfollow}
Ejemplo de firma	/users/1569/unfollow/1235

Parámetros a utilizar

Parámetro	Tipo de dato	Descripción
user_id	int	Número que identifica al usuario actual
user_id_to_follow	int	Número que identifica al usuario a dejar de seguir

Lógica de negocio controlada a través de Bad Request:

- ➔ Se controla que el usuario follower exista.
- ➔ Se controla que el usuario followed exista.
- ➔ Se controla que el usuario follower ya esté siguiendo al usuario follower.
- ➔ Se controla que el usuario follower y followed no sean el mismo usuario.

US 0008

El sistema debe ser capaz de:

- ➔ Ordenar de manera ascendente la lista de followers, en base a sus nombres, de un determinado usuario.
- ➔ Ordenar de manera descendente la lista de followers, en base a sus nombres, de un determinado usuario.
- ➔ Ordenar de manera ascendente la lista de followed, en base a sus nombres, de un determinado usuario.
- ➔ Ordenar de manera descendente la lista de followed, en base a sus nombres, de un determinado usuario.

Método

Método	GET
Firma	/users/{user_id}/followers/list?order=name_asc /users/{user_id}/followers/list?order=name_desc /users/{user_id}/followed/list?order=name_asc /users/{user_id}/followed/list?order=name_desc

Parámetros a utilizar

Order	Descripción
name_asc	Ordena de manera ascendente en base a los nombres de los usuarios.
name_desc	Ordena de manera descendente en base a los nombres de los usuarios.

Lógica de negocio controlada a través de Bad Request:

- ➔ Se controla que el usuario exista.
- ➔ Se controla que la lista de followed no retorne vacía.
- ➔ Se controla que la lista de followers no retorne vacía.

US 0009

El sistema debe ser capaz de:

- ➔ Ordenar de manera ascendente la lista de publicaciones de los followed, en base a sus fechas, de un determinado usuario.
- ➔ Ordenar de manera descendente la lista de publicaciones de los followed, en base a sus fechas, de un determinado usuario.

Método

Método	GET
Firma	/products/followed/{user_id}/list?order=date_asc /products/followed/{user_id}/list?order=date_desc

Parámetros a utilizar

Order	Descripción
date_asc	Ordena de manera ascendente en base a las fechas de las publicaciones de los usuarios, es decir, de la más antigua a la más nueva.
date_desc	Ordena de manera descendente en base a las fechas de las publicaciones de los usuarios, es decir, de la más nueva a la más antigua.

Lógica de negocio controlada a través de Bad Request:

- ➔ Se controla que el usuario exista.
- ➔ Se controla que la lista de followed no retorne vacía.

US 0010

El sistema debe ser capaz de dar de alta una publicación de un nuevo producto en promoción.

Método

Método	POST
Firma	/products/promo-post

Payload	{
	"user_id": 1569,
	"id_post": 18,
	"date": "29-04-2021",
	"detail": {
	"product_id": 1,
	"product_name": "Silla Gamer",
	"type": "Gamer",
	"brand": "Racer",
	"color": "Red & Black",
	"notes": "Special Edition"
	},
	"category": "100",
	"price": 1500.5,
	"has_promo": true,
	"discount": 0.25
	}

Response OK	Status Code 200
Response Bad Request	Status Code 400

Parámetros a utilizar

Parámetro	Tipo de dato	Descripción
user_id	int	Número que identifica al usuario
id_post	int	Número que identifica a la publicación
date	LocalDate	Fecha de publicación en formato dd-MM-yyyy
product_id	int	Número que identifica a los productos pertenecientes a las publicaciones
product_name	String	String que representa el nombre del producto

type	String	String que representa el tipo del producto
brand	String	String que representa la marca del producto
color	String	String que representa el color del producto
notes	String	String que representa las observaciones del producto
category	int	Número que representa la categoría del producto
price	double	Número decimal que representa el precio del producto
has_promo	boolean	Campo booleano que determina si un producto está o no en promoción.
discount	double	Número decimal que representa el monto de descuento para productos en promoción.

Lógica de negocio controlada a través de Bad Request:

- ➔ Se controla que el usuario exista.
- ➔ Se controla que la publicación todavía no exista.
- ➔ Se controla que se pase true por has_prom, dado que si no desea dar de alta un producto en promoción entonces se debería cargar el producto por el post de la US 0005.
- ➔ Se controla que el descuento aplicado no sea menor o igual a 0, dado que sino no se estaría aplicando ningún descuento.

US 0011

El sistema debe ser capaz de poder obtener la cantidad de publicaciones con productos en promoción, dado determinado vendedor.

Método

Método	GET
Firma	/products/{user_id}/promo-post/count

Response	{
	"user_id" : 1569,
	"user_name": "vendedor1",
	"promo_products_count": 23
	}

Parámetros a utilizar

Order	Descripción
user_id	Número que identifica al usuario actual.
user_name	Nombre de usuario asociado al user_id.
promo_products_count	Cantidad numérica de productos en promoción de un determinado usuario.

Lógica de negocio controlada a través de Bad Request:

- ➔ Se controla que el usuario exista.
- ➔ Se controla que el usuario tenga publicaciones con productos en promoción.

US 0012

El sistema debe ser capaz de obtener todas las publicaciones con productos en promoción de determinado vendedor.

Método

Método	GET
Firma	/products/{user_id}/list

Response	{
	"user_id": 1569,
	"user_name": "vendedor1",
	"posts": [{
	"id_post": 18,
	"date": "29-04-2021",
	"detail": {
	"product_id": 1,
	"product_name": "Silla Gamer",
	"type": "Gamer",
	"brand": "Racer",
	"color": "Red & Black",
	"notes": "Special Edition"
	},
	"category": "100",
	"price": 15000.5,
	"has_promo": true,
	"discount": 0.25
	}, {
	"id_post": 32,
	"date": "01-05-2021",
	"detail": {
	"product_id": 62,
	"product_name": "Headset RGB Inalámbrico",
	"type": "Gamer",
	"brand": "Razer",
	"color": "Green with RGB",
	"notes": "Sin Batería"
	},
	"category": "120",
	"price": 2800.69,
	"has_promo": true,
	"discount": 0.1
	}]
	}

Parámetros a utilizar

Parámetro	Tipo de dato	Descripción
user_id	int	Número que identifica al usuario
user_name	String	Nombre de usuario asociado al user_id.
id_post	int	Número que identifica a la publicación
date	LocalDate	Fecha de publicación en formato dd-MM-yyyy
product_id	int	Número que identifica a los productos pertenecientes a las publicaciones
product_name	String	String que representa el nombre del producto
type	String	String que representa el tipo del producto
brand	String	String que representa la marca del producto
color	String	String que representa el color del producto
notes	String	String que representa las observaciones del producto
category	int	Número que representa la categoría del producto
price	double	Número decimal que representa el precio del producto
has_promo	boolean	Campo booleano que determina si un producto está o no en promoción.
discount	double	Número decimal que representa el monto de descuento para productos en promoción.

Lógica de negocio controlada a través de Bad Request:

- ➔ Se controla que el usuario exista.
- ➔ Se controla que el usuario tenga publicaciones con productos en promoción.