

# MELI CHALLENGE: CARRITO DE COMPRAS



# CONTEXT

Deseamos que los usuarios puedan gestionar sus carritos de compras utilizando Python para el backend y ReactJS para el frontend. Las funcionalidades que queremos implementar son las siguientes:

- Agregar productos al carrito.
- Eliminar productos del carrito.
- Modificar la cantidad de un producto.
- Mostrar el número total de productos en el carrito.
- Calcular y mostrar el precio total.
- Calcular el subtotal por producto.
- Mostrar las opciones de envío.





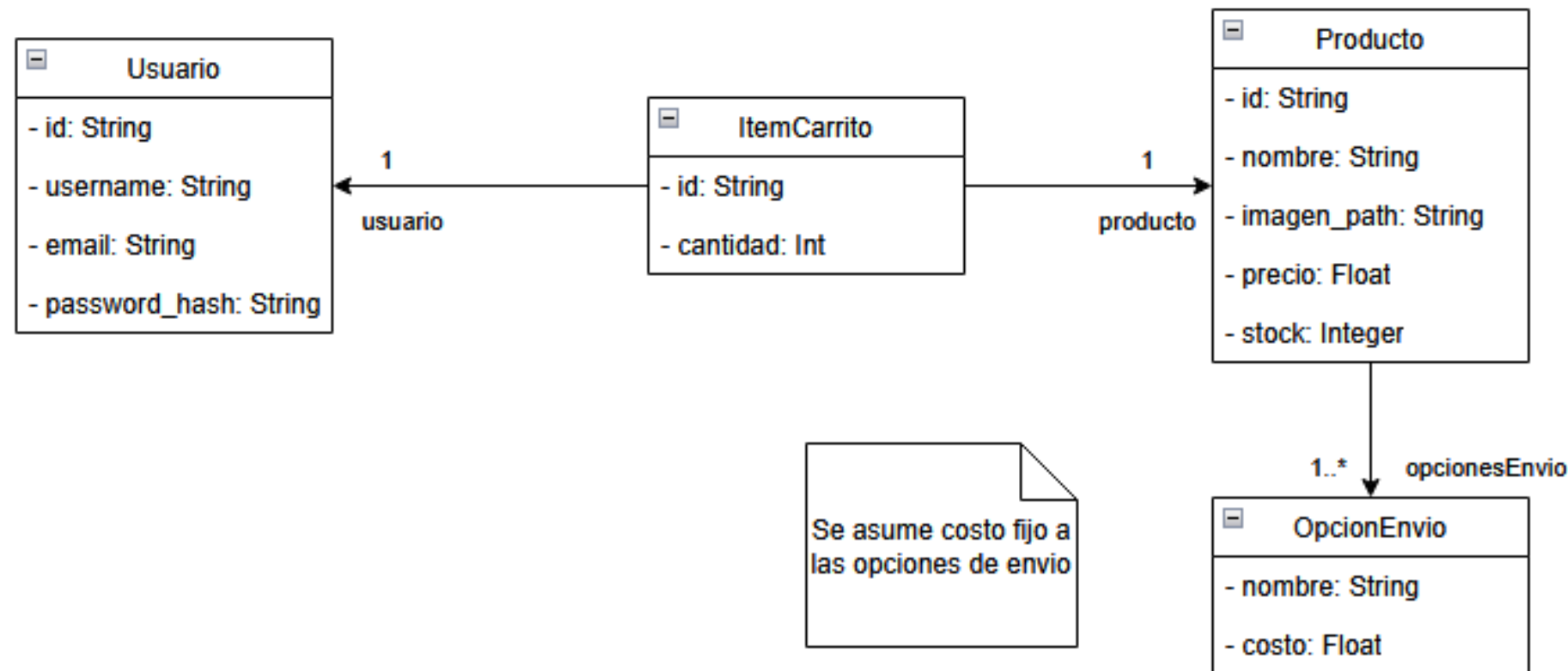
# DEFINICIÓN DE ELEMENTOS

Como primer paso, definí las tecnologías que utilizaré. Para ello, consideré aquellas que ya he usado anteriormente y que me han dado buenos resultados, con el objetivo de facilitar el desarrollo en un periodo corto de tiempo. Además, incluí las tecnologías solicitadas en el enunciado.

Las tecnologías seleccionadas son:

- Flask para el pequeño backend en Python.
- SQLAlchemy como ORM.
- PostgreSQL como base de datos relacional.
- ReactJS para el frontend usando el Context API para mantener el estado.
- Bootstrap para aprovechar elementos visuales y funcionales ya existentes.
- Fetch API de JS para realizar las simples peticiones. HTTP.
- NGINX para el servidor del frontend por su facilidad de implementación.
- JWT para la autenticación.
- Docker para la contenedorización de la aplicación.

# DISEÑO



Posterior a la definición de tecnologías, realicé un diagrama de clases/conceptual con el fin de comprender los conceptos involucrados y sus relaciones, lo cual me permitió estructurar adecuadamente el modelo de datos para la base de datos.

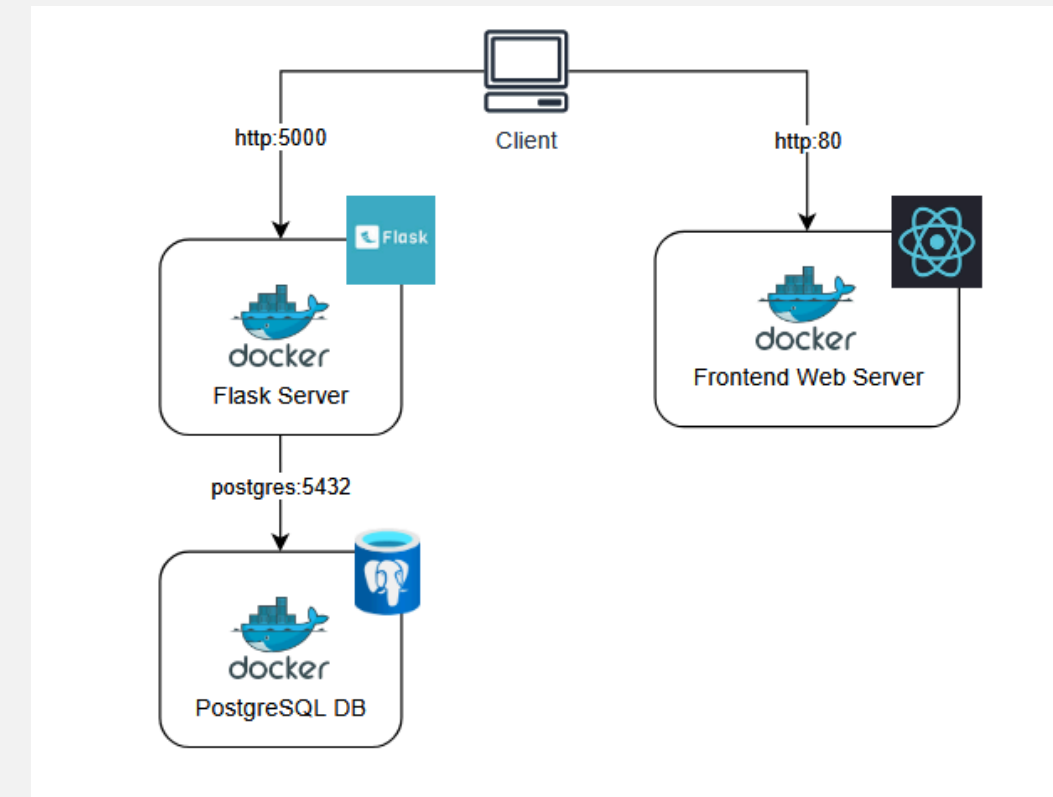
Adicionalmente, diseñé los endpoints necesarios para cumplir con los requerimientos del reto. Estos endpoints se encuentran documentados en el archivo README del repositorio correspondiente.

# DISEÑO

Después de diseñar los endpoints, se definió un pequeño diagrama de despliegue que muestra los componentes involucrados en la solución.

Se optó por una arquitectura monolítica, dado que se trata de un proyecto pequeño que no se espera que maneje un alto número de usuarios concurrentes. El servidor Flask sigue una aproximación basada en el patrón MVC (Modelo-Vista-Controlador), en el cual:

- La vista está representada por los endpoints del API.
- El controlador está integrado en la lógica contenida en las vistas, ya que son funciones cortas y específicas.
- El modelo gestiona la persistencia de datos a través de la base de datos, exponiendo la funcionalidad necesaria mediante SQLAlchemy.





### 1. Agregar productos del carrito

Método	POST
Ruta	/users/cart
Parámetros	N/A
Encabezados	Authorization: Bearer token
Cuerpo	<pre>{   "cantidad": Cantidad del elemento a insertar,   "id_producto": Identificador del producto, }</pre>

# REQUERIMIENTO 1





# REQUERIMIENTO 1

La solución del requerimiento involucra tres entidades: Producto, Usuario e ItemCarrito. Este endpoint crea un ItemCarrito que asocia un producto y su cantidad a un usuario, formando así el “Carrito”.


El token JWT en el encabezado permite identificar al usuario. El endpoint valida:


- El token.
- El formato de los datos.
- Que la cantidad sea mayor a 0 y menor o igual al stock.
- La existencia del producto y del usuario.

Si el producto ya está en el carrito, se devuelve un error indicando lo que ocurrió ya que se debe usar el endpoint para modificar la cantidad en caso de querer hacer una actualización. En caso de éxito, se retorna un mensaje confirmando la operación.






 MeLi Challenge [Productos](#) [Carrito](#) [Logout](#)




iPhone

**Precio:** \$10    **Stock:** 5




Agregar al carrito




PS5

**Precio:** \$20    **Stock:** 10




Agregar al carrito



RTX 4090

**Precio:** \$30    **Stock:** 3



Agregar al carrito

# REQUERIMIENTO 1







# REQUERIMIENTO 1

Para el usuario, se diseñó un listado de productos accesible tras registrarse e iniciar sesión. Una vez autenticado, puede añadir artículos al carrito en distintas cantidades. Si el usuario no ha iniciado sesión, las funciones para añadir productos estarán deshabilitadas.



2. Eliminar productos del carrito.

Método	DELETE
Ruta	/users/cart/{id}
Parámetros	id: identificador del item del cart
Encabezados	Authorization: Bearer token
Cuerpo	N/A

REQUERIMIENTO 2









## REQUERIMIENTO 2

Para este requerimiento, se utiliza nuevamente el token de autenticación, tanto para validar como para identificar al usuario. El endpoint verifica que el producto esté en su carrito y, si es así, elimina la entidad ItemCarrito correspondiente usando su identificador, recibido como parámetro. En caso de éxito, el endpoint no retorna contenido (status 204).



MeLi Challenge <a href="#">Productos</a> <a href="#">Carrito</a> <a href="#">Logout</a>						
# de productos: 2				Total: \$40		
	<b>Nombre:</b> RTX 4090 <b>Precio:</b> \$30	<ul style="list-style-type: none"><li>• Envio Normal \$5</li><li>• Envio Express \$10</li></ul>	<div>1</div>	<b>Subtotal:</b> \$30		
	<b>Nombre:</b> iPhone <b>Precio:</b> \$10	<ul style="list-style-type: none"><li>• Envio Normal \$5</li><li>• Envio Express \$10</li></ul>	<div>1</div>	<b>Subtotal:</b> \$10		

# REQUERIMIENTO 2





## REQUERIMIENTO 2

Para este requerimiento, se añadió un botón de borrado junto a cada producto en el carrito del usuario. Al presionarlo, el producto se elimina del carrito y se actualizan automáticamente la cantidad total de productos y el precio total.



### 3. Cambiar la cantidad de cada producto en el carrito

Método	PUT
Ruta	/users/cart/{id}
Parámetros	id: identificador del item del cart
Encabezados	Authorization: Bearer token
Cuerpo	<pre>{   "cantidad": Cantidad del elemento a actualizar, }</pre>

## REQUERIMIENTO 3





## REQUERIMIENTO 3





Este requerimiento expone un endpoint que permite actualizar la cantidad de un producto en el carrito, identificado por un parámetro. Se utiliza un token para validar al usuario y asegurar que el carrito le pertenece.

El endpoint verifica:

- La validez del token.
- La existencia del usuario.
- Que el producto esté en el carrito.
- Que la cantidad sea mayor o igual a 1 y menor o igual al stock disponible.





MeLi Challenge <a href="#">Productos</a> <a href="#">Carrito</a> <a href="#">Logout</a>					
# de productos: 2				Total: \$40	
	<b>Nombre:</b> RTX 4090 <b>Precio:</b> \$30	<ul style="list-style-type: none"><li>• Envio Normal \$5</li><li>• Envio Express \$10</li></ul>	<div>1</div>	<b>Subtotal:</b> \$30	
	<b>Nombre:</b> iPhone <b>Precio:</b> \$10	<ul style="list-style-type: none"><li>• Envio Normal \$5</li><li>• Envio Express \$10</li></ul>	<div>1</div>	<b>Subtotal:</b> \$10	

# REQUERIMIENTO 3





## REQUERIMIENTO 3

Este requerimiento se refleja en la interfaz como un menú desplegable que permite seleccionar una cantidad válida, limitada por el stock disponible. Esto facilita el uso de la funcionalidad y hace que la actualización de la cantidad en el carrito sea intuitiva y rápida para el usuario.



#### 4. Obtener el carrito de un usuario

Método	GET
Ruta	<code>/users/cart</code>
Parámetros	N/A
Encabezados	<code>Authorization: Bearer token</code>
Cuerpo	N/A

## REQUERIMIENTOS 4 A 7





## REQUERIMIENTOS 4 A 7





Estos cuatro requerimientos se implementaron en un solo endpoint, ya que se utilizan simultáneamente en una misma pantalla. El endpoint recibe y valida un token para autenticar e identificar al usuario, y luego le retorna su carrito.

Se realiza lo siguiente:

- Se obtienen los items del carrito asociados al usuario, junto con los datos de los productos y las opciones de envío.
- Se calcula el número total de productos como la suma de las cantidades solicitadas.
- Se calcula el precio total, multiplicando el precio de cada producto por su cantidad y realizando la suma.
- Se obtiene el subtotal por producto mediante la misma multiplicación individual.

Finalmente, se retorna un objeto JSON con toda esta información estructurada.



MeLi Challenge <a href="#">Productos</a> <a href="#">Carrito</a> <a href="#">Logout</a>					
# de productos: 2				Total: \$40	
	<b>Nombre:</b> RTX 4090 <b>Precio:</b> \$30	<ul style="list-style-type: none"><li>• Envio Normal \$5</li><li>• Envio Express \$10</li></ul>	<div>1</div>	<b>Subtotal:</b> \$30	
	<b>Nombre:</b> iPhone <b>Precio:</b> \$10	<ul style="list-style-type: none"><li>• Envio Normal \$5</li><li>• Envio Express \$10</li></ul>	<div>1</div>	<b>Subtotal:</b> \$10	

# REQUERIMIENTO 4-7





# FEATURE EXTRAS

Se desarrollaron una serie de funcionalidades adicionales para mejorar la demostración de los requerimientos solicitados. Estas son:

- Listado de productos
- Registro de usuarios
- Inicio de sesión (Login)
- Cierre de sesión (Logout)



## 1. Obtener los productos

Método	GET
Ruta	/products
Parámetros	N/A
Encabezados	N/A
Cuerpo	N/A

## 1. Registrarse

Método	POST
Ruta	/users
Parámetros	N/A
Encabezados	N/A
Cuerpo	<pre>{   "username": Nombre de usuario alfanumerico y de minimo 5 caracteres unico,   "email": Email del usuario unico,   "password": Contraseña, }</pre>

# FUNCIONALIDADES ADICIONALES







# FUNCIONALIDADES ADICIONALES

Para hacer el proyecto más completo y fácil de probar, se implementaron varios endpoints adicionales que complementan las funcionalidades originales.

- El primer endpoint permite obtener el listado de productos, lo que facilita su visualización en la página principal y habilita la opción de añadirlos al carrito.
- El segundo endpoint permite a los usuarios registrarse, lo que les da acceso a un carrito persistente en la base de datos. Este endpoint recibe username, email y password, y realiza las siguientes validaciones:
  - Que el email tenga un formato válido.
  - Que el username sea un string sin espacios ni caracteres especiales, y tenga al menos 5 caracteres.
  - Que el email y el username no estén registrados previamente.
  - Que la contraseña tenga al menos 8 caracteres.



iPhone

**Precio:** \$10

**Stock:** 5

1



Agregar al carrito



PS5

**Precio:** \$20

**Stock:** 10

1



Agregar al carrito



RTX 4090


**Precio:** \$30

**Stock:** 3

1



Agregar al carrito

 MeLi Challenge

[Productos](#) [Login](#) [Register](#)

Registro

Email

Username

Contraseña

Registrarse



# **FUNCIONALIDADES ADICIONALES**

El listado de productos se desarrolló simulando el diseño comúnmente utilizado en sitios de compras en línea, es decir, mediante agrupaciones de tarjetas. Cada tarjeta muestra la información del producto, un menú desplegable para seleccionar la cantidad deseada y un botón para añadir el producto al carrito.

### Ingrese a su cuenta

Username

Contraseña

Entrar



# **FUNCIONALIDADES ADICIONALES**

Para el registro e inicio de sesión del usuario se implementaron formularios simples. En el caso del formulario de registro, se realizan validaciones en los campos ingresados, como el formato y la longitud. Si alguno de los valores no cumple con las condiciones requeridas, se bloquea el envío al backend, evitando así intentos inválidos de registro.

Cuando se comete un error en algún campo, este se resalta en rojo y se muestra un mensaje indicando el tipo de error, mejorando la experiencia del usuario y facilitando la corrección.



## 2. Login

Método	POST
Ruta	/users/login
Parámetros	N/A
Encabezados	N/A
Cuerpo	<pre>{   "username": Nombre de usuario alfanumerico y de minimo 5 caracteres unico,   "password": Contraseña, }</pre>

# FUNCIONALIDADES ADICIONALES







# **FUNCIONALIDADES ADICIONALES**

Para la funcionalidad de inicio de sesión, simplemente se verifica que los datos proporcionados por el usuario correspondan a un usuario existente y que las credenciales sean correctas.



# DEMOSTRACIÓN, EXPLORACIÓN Y RECORRIDO DE LA SOLUCIÓN

```
> __pycache__
└─ __init__.py
└─ itemCarrito.py
└─ model.py
└─ opcionEnvio.py
└─ producto.py
└─ usuario.py
└─ views
    > __pycache__
    └─ __init__.py
    └─ cart.py
    └─ products.py
    └─ users.py
└─ __init__.py
└─ main.py
└─ session.py
$ .env.development
```