Carrito de Compras (Ecommerce)

Esta guía documenta los **endpoints del módulo de carrito y checkout** de la API de **Laravel**.
Permiten crear, consultar, actualizar, vaciar y finalizar carritos de compra.

> ⚙️ **Autenticación:** todos los endpoints requieren un **Bearer Token**
> (`Authorization: Bearer <TOKEN>`)
> El token se obtiene tras el login (ver documentación de **Auth**).

Resumen de Endpoints

Acción	Método	Ruta	Body / Query	Auth
Obtener carrito activo	GET	`/cart`	—	✅
Ver carrito por ID	GET	`/cart/{cart}`	—	✅
Actualizar carrito (bulk)	PUT	`/cart/{cart}`	✅ JSON	✅
Vaciar carrito	DELETE	`/cart/{cart}`	—	✅
Checkout (crear orden)	POST	`/cart/{cart}/checkout`	—	✅
Agregar ítem al carrito	POST	`/cart/items`	✅ JSON	✅
Actualizar cantidad de ítem	PUT	`/cart/items/{item}`	✅ JSON	✅
Eliminar ítem del carrito	DELETE	`/cart/items/{item}`	—	✅

1️⃣ Obtener carrito activo — `GET /cart`

Obtiene el carrito activo del usuario.
Si no existe, el sistema lo crea automáticamente.

Ejemplo JS

```
const token = localStorage.getItem('auth_token');

fetch(`${BASE_URL}/cart`, {
  headers: { 'Authorization': `Bearer ${token}` }
})
  .then(r => r.json())
  .then(console.log)
  .catch(console.error);
```

Ejemplo de respuesta


```
{
  "success": true,
  "message": "Carrito actual",
  "data": {
    "id": 8,
    "status": "active",
    "items": [
      {
        "id": 21,
        "product": "Camisa Blanca",
        "quantity": 2,
        "unit_price": 199,
        "subtotal": 398
      }
    ],
    "total": 398
  }
}
```


2️⃣ Ver carrito por ID — `GET /cart/{cart}`

Consulta un carrito específico (solo si pertenece al usuario autenticado).

Ejemplo JS

```
fetch(`${BASE_URL}/cart/8`, {
  headers: { 'Authorization': `Bearer ${token}` }
})
  .then(r => r.json())
  .then(console.log);
```

Ejemplo de respuesta


```
{
  "success": true,
  "message": "Detalle del carrito",
  "data": {
    "id": 8,
    "status": "active",
    "items": [
      {
        "id": 21,
        "product": "Camisa Blanca",
        "quantity": 2,
        "price": 199
      }
    ]
  }
}
```


3️⃣ Agregar ítem al carrito — `POST /cart/items`

Agrega un producto (por `price_id`) al carrito activo del usuario.
Si ya existe ese `price_id`, se incrementa la cantidad.

Body (JSON)

```
{
  "price_id": 5,
  "quantity": 2
}
```

Ejemplo JS

```
fetch(`${BASE_URL}/cart/items`, {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': `Bearer ${token}`
  },
  body: JSON.stringify({
    price_id: 5,
    quantity: 2
  })
})
  .then(r => r.json())
  .then(console.log)
  .catch(console.error);
```

Respuesta (201 Created)


```
{
  "success": true,
  "message": "Producto agregado al carrito",
  "data": {
    "id": 8,
    "items": [
      {
        "id": 33,
        "product": "Zapatillas deportivas",
        "quantity": 2,
        "unit_price": 299,
        "subtotal": 598
      }
    ]
  }
}
```


4️⃣ Actualizar carrito (bulk) — `PUT /cart/{cart}`

Actualiza el carrito completo en una sola petición.
Permite reemplazar o fusionar ítems (modo `replace` o `merge`).

Body (JSON)


```
{
  "mode": "replace",
  "items": [
    { "price_id": 5, "quantity": 1 },
    { "price_id": 7, "quantity": 3 }
  ]
}
```

Ejemplo JS

```
fetch(`${BASE_URL}/cart/8`, {
  method: 'PUT',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': `Bearer ${token}`
  },
  body: JSON.stringify({
    mode: 'replace',
    items: [
      { price_id: 5, quantity: 1 },
      { price_id: 7, quantity: 3 }
    ]
  })
})
  .then(r => r.json())
  .then(console.log);
```

Respuesta

```
{
  "success": true,
  "message": "Carrito actualizado",
  "data": {
    "id": 8,
    "items": [
      { "price_id": 5, "quantity": 1 },
      { "price_id": 7, "quantity": 3 }
    ],
    "total": 896
  }
}
```

5️⃣ Actualizar cantidad de ítem — `PUT /cart/items/{item}`

Modifica la cantidad de un ítem individual.

Body (JSON)

```
{
  "quantity": 4
}
```

Ejemplo JS

```
fetch(`${BASE_URL}/cart/items/33`, {
  method: 'PUT',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': `Bearer ${token}`
  },
  body: JSON.stringify({ quantity: 4 })
})
  .then(r => r.json())
  .then(console.log);
```

Ejemplo de respuesta

```
{
  "success": true,
  "message": "Cantidad actualizada",
  "data": {
    "item_id": 33,
    "quantity": 4,
    "subtotal": 1196
  }
}
```

6️⃣ Eliminar ítem del carrito — `DELETE /cart/items/{item}`

Elimina un ítem específico del carrito.

Ejemplo JS

```
fetch(`${BASE_URL}/cart/items/33`, {
  method: 'DELETE',
  headers: { 'Authorization': `Bearer ${token}` }
})
  .then(r => r.json())
  .then(console.log);
```

Ejemplo de respuesta

```
{
  "success": true,
  "message": "Ítem eliminado del carrito"
}
```

7️⃣ Vaciar carrito — `DELETE /cart/{cart}`

Elimina **todos los ítems** del carrito especificado.

Ejemplo JS

```
fetch(`${BASE_URL}/cart/8`, {
  method: 'DELETE',
  headers: { 'Authorization': `Bearer ${token}` }
})
  .then(r => r.json())
  .then(console.log);
```

Ejemplo de respuesta

```
{
  "success": true,
  "message": "Carrito vaciado",
  "data": {
    "id": 8,
    "items": []
  }
}
```

8 Checkout — `POST /cart/{cart}/checkout`

Cierra el carrito actual, crea una **orden de compra**, descuenta stock y genera el siguiente carrito activo.

Ejemplo JS

```
fetch(`${BASE_URL}/cart/8/checkout`, {
  method: 'POST',
  headers: { 'Authorization': `Bearer ${token}` }
})
  .then(r => r.json())
  .then(console.log);
```

Ejemplo de respuesta (éxito)

```
{
  "success": true,
  "message": "Checkout exitoso",
  "data": {
    "order_id": 123,
    "status": "paid",
    "total": 1196,
    "created_at": "2025-11-12T10:32:45Z"
  }
}
```

Posibles errores

Tipo	Código	Ejemplo
Validación	422	`{ "errors": { "price_id": ["El campo es obligatorio."] } }`
No autorizado	403	`{ "message": "No tienes permiso para modificar este carrito." }`
Carrito no encontrado	404	`{ "message": "Carrito no encontrado." }`

Notas adicionales

- Todos los endpoints retornan el campo `"success": true/false` .
- Los carritos están asociados al usuario autenticado.
- Los precios (`price_id`) deben existir en la tabla `prices` .
- El `checkout` genera automáticamente una nueva orden y deja el carrito cerrado (`status: closed`).