Cenaria – API de Despensa (Pantry) – v0.1

Fecha: 2025-08-17

Resumen

Esta documentación describe los endpoints REST del módulo **Despensa** para gestionar el inventario de ingredientes por usuario. La API está protegida con **Cognito (JWT)** y utiliza **DynamoDB** como almacén. Incluye patrones de paginación, idempotencia, y control de concurrencia optimista (OCC) por versión.

Base URL & Autenticación

Base URL (dev): `https://y3sh6vq335.execute-api.us-east-2.amazonaws.com/dev`

Autenticación: Cognito User Pool Authorizer. Enviar `Authorization: Bearer <ID_TOKEN>`

Formato de error: `{ code, message, details? }`

Headers comunes

Header	Valor / Ejemplo	Obligatorio
Authorization	Bearer <ID_TOKEN>	Sí (todas las rutas)
Content-Type	application/json	Sí en POST/PUT/PATCH
If-Match	Entero con versión esperada (OCC)	Recomendado en PUT/PATCH, opcional en DELETE

Esquemas (DTOs)

```
// PantryItem
{
  "id": "string (ulid)",
  "name": "string (2..80)",
  "quantity": number (>= 0),
  "unit": "g|kg|ml|l|pieza|taza|cda|cdta",
  "category": "verduras|frutas|carnes|lácteos|granos|especias|enlatados|otros",
  "perishable": boolean (opcional),
  "notes": "string" (opcional),
  "createdAt": epoch_ms,
  "updatedAt": epoch_ms,
  "version": integer (OCC)
}

// NewPantryItem (POST)
{
  "name": "string",
  "quantity": number,
  "unit": "Unit",
  "category": "Category" (opcional),
  "perishable": boolean (opcional),
  "notes": "string" (opcional)
}
```

Paginación & Idempotencia

• Paginación: cursor con base64 en `nextCursor`. Enviar `?cursor=<valor>` en llamadas subsiguientes. Parámetro `limit` (1..100, default 20).

• Idempotencia (POST /pantry): Se recomienda un encabezado o campo `Idempotency-Key` si se habilita en el servicio. Actualmente, el POST crea ítems y retorna la lista creada.

GET /v1/pantry

Lista los ingredientes de la despensa del usuario autenticado.

Query params: `search?` (prefix), `category?`, `limit?`, `cursor?`

200 OK: `{ items: PantryItem[], nextCursor?: string }`

```
curl -sS "https://y3sh6vq335.execute-api.us-east-2.amazonaws.com/dev/v1/pantry?limit=50" \
  -H "Authorization: Bearer $IDTOKEN"
```

POST /v1/pantry

Crea uno o varios ingredientes.

Body: `{ items: NewPantryItem[] }`

201 Created: `{ items: PantryItem[] }`

```
curl -sS -X POST "https://y3sh6vq335.execute-api.us-east-2.amazonaws.com/dev/v1/pantry" \
  -H "Authorization: Bearer $IDTOKEN" \
  -H "content-type: application/json" \
  -d '{"items":[{"name":"Arroz","quantity":1,"unit":"kg","category":"granos"}]}'
```

GET /v1/pantry/{id}

Obtiene un ingrediente por ID.

200 OK: `PantryItem` | **404 Not Found**

```
ID="01K2ABCDEF123..."  # ejemplo
curl -sS "https://y3sh6vq335.execute-api.us-east-2.amazonaws.com/dev/v1/pantry/$ID" \
  -H "Authorization: Bearer $IDTOKEN"
```

PUT /v1/pantry/{id}

Reemplaza completamente el ítem (body completo requerido). Usa **OCC**: enviar `If-Match: <version>`. **200 OK** con ítem actualizado, **409 Conflict** si la versión no coincide.

```
curl -sS -X PUT "https://y3sh6vq335.execute-api.us-east-2.amazonaws.com/dev/v1/pantry/$ID" \
  -H "Authorization: Bearer $IDTOKEN" \
  -H "If-Match: 2" \
  -H "content-type: application/json" \
  -d '{"name":"Arroz","quantity":2,"unit":"kg","category":"granos","perishable":false}'
```

PATCH /v1/pantry/{id}

Actualiza parcialmente los campos enviados. Admite **OCC opcional** con `If-Match`. Campos permitidos: `name, quantity, unit, category, perishable, notes`.

```
curl -sS -X PATCH "https://y3sh6vq335.execute-api.us-east-2.amazonaws.com/dev/v1/pantry/$ID" \
  -H "Authorization: Bearer $IDTOKEN" \
  -H "If-Match: 2" \
  -H "content-type: application/json" \
  -d '{"quantity":3,"perishable":true,"notes":"comprado hoy"}'
```

DELETE /v1/pantry/{id}

Elimina un ingrediente. Soporta **OCC opcional** con `If-Match`. **204 No Content** si se elimina; **409** si versión no coincide; **404** si no existe (cuando no se usa OCC).

```
curl -i -sS -X DELETE "https://y3sh6vq335.execute-api.us-east-2.amazonaws.com/dev/v1/pantry/$ID"
  -H "Authorization: Bearer $IDTOKEN" \
  -H "If-Match: 3"
```

Códigos de estado

Código	Descripción
200	OK
201	Created
204	No Content
400	Bad Request (validación de payload, parámetros)
401/403	Unauthorized/Forbidden (JWT inválido/ausente)
404	Not Found
409	Conflict (OCC – versión no coincide)
413	Payload muy grande
429	Rate limit
500	Error interno

Notas & Reglas de validación

• **quantity** ≥ 0; **name** longitud 2..80; **notes** ≤ 240. Las listas válidas de **unit** y **category** se validan del lado del servidor.

• **Normalización** de nombre: minúsculas + sin acentos para búsqueda por prefijo. • **Paginación**: usar `nextCursor` hasta agotar resultados.

• **Seguridad**: todas las llamadas requieren JWT de Cognito del mismo User Pool configurado en el authorizer.

Anexo — Ejemplo de respuesta `GET /v1/pantry`


```
{
  "items": [
    {
      "id": "01K2T1WEV87BR9254JVMTG0D1W",
      "name": "Arroz",
      "quantity": 3,
      "unit": "kg",
      "category": "granos",
      "perishable": true,
      "notes": "comprado hoy",
      "createdAt": 1755368536936,
      "updatedAt": 1755445423486,
      "version": 3
    }
  ],
  "nextCursor": "eyJwayI6ICJVU0VSIy4uLiJ9"
}
```