Microservicio de Pagos: Franco Veggiani

Casos de uso

CU: Ver métodos de pago

- Precondición: Usuario autenticado con métodos de pago vigentes o no vigentes.
- Camino normal:
  - El usuario consulta sus métodos vigentes mediante `GET /get_payment_methods/{usuario_id}`.
  - El servicio devuelve una lista con información consolidada por método: tipo (débito/crédito/billetera), marca o gateway, últimos 4 dígitos, nombre del titular, banco, identificador de wallet y si es predeterminado.
- Caminos alternativos:
  - Si el usuario no tiene métodos vigentes, retorna lista vacía (200).
  - Si el tipo no puede inferirse por catálogo, se intenta resolver por el detalle asociado (tarjeta o billetera); si no hay detalle, el método no se incluye.

CU: Gestionar métodos de pago

- Precondición: El usuario autenticado necesita administrar los métodos de pago disponibles para cobros posteriores.
- Camino normal:
  - Alta: `POST /create_payment_method` con `tipo_pago_id` y `moneda_id`.
    - Tarjeta: requiere `marca_pago_id`, `nro_tarjeta`, `nombre_titular`, `fecha_vencimiento`, `banco_id`. Crea un detalle en `metodos_pago_tarjeta` y lo vincula por `metodo_pago_detalle_id`. Se guardan los últimos 4 dígitos.
    - Billetera: requiere `gateway_pago_id`, `identificador_wallet`, `moneda_id`. Crea el detalle en `metodos_pago_billetera` y lo vincula.
  - Default: si `es_default = true`, desactiva el default previo del usuario y marca el nuevo.
  - Modificación: `PATCH /update_payment_method/{pm_id}`. Puede actualizar campos principales y del detalle; si cambia el tipo, se crea un nuevo detalle y se da de baja el anterior.
  - Baja lógica: `DELETE /delete_payment_method/{pm_id}` para métodos que no son default.
- Caminos alternativos:
  - Faltan campos obligatorios según tipo: `400` con campos faltantes.
  - Si ya existe un método igual para el usuario (misma tarjeta/billetera), se reactiva y actualiza; se devuelve el método reactivado.
  - Intento de borrar un método default: `400`.

CU: Crear pago

- Endpoint: `POST /create_payment`
- Body: `{ nro_cuenta, monto_pagado, order_id, metodo_pago_id }`
- Resultado: crea un pago en estado “En Proceso” y publica evento de estado en la cola de salida.

CU: Cancelar pago

- Endpoint: `GET /cancel_payment/{payment_id}`
- Resultado: cambia el estado a “Cancelado” si el estado actual lo permite y publica evento de estado en la cola de salida.
- Nota (asíncrono relacionado): también puede actualizarse el estado vía cola `payments_set_status` con `status_id` o `new_status` ("Realizado" | "Fallido").

Modelo de datos

metodos_pago

- id
- usuario_id
- tipo_pago_id
- moneda_id
- es_default
- fecha_hora_alta
- fecha_hora_baja
- metodo_pago_detalle_id

metodos_pago_tarjeta

- id
- marca_pago_id
- numero_tarjeta
- ultimos_cuatro_digitos
- fecha_vencimiento
- nombre_titular
- banco_id
- fecha_hora_alta
- fecha_hora_baja

metodos_pago_billetera

- id
- proveedor_id
- wallet_id
- moneda_id
- fecha_hora_alta
- fecha_hora_baja

pagos

- id
- fecha_hora_creacion
- estado_actual
- metodo_pago_id
- orden_id

estado_pago

- id
- fecha_hora_alta
- fecha_hora_baja
- nombre_estado

tipos_metodos_pago

- id
- fecha_hora_alta
- fecha_hora_baja
- nombre_metodo

marcas_metodos_pago

- id
- fecha_hora_alta
- fecha_hora_baja
- nombre_marca

proveedores_billetera

- id
- fecha_hora_alta
- fecha_hora_baja
- nombre_gateway

monedas

- id
- moneda_nombre
- fecha_hora_baja
- fecha_hora_alta

bancos

- id
- nombre_banco
- fecha_hora_baja
- fecha_hora_alta

Interfaz REST (FastAPI)

- `POST /create_payment`
  - Body: `create_payment_request` → `{ order_id: int, metodo_pago_id: int }`
  - Respuesta: `create_payment_out` → `{ estado, order_id, payment_id }`
- `GET /cancel_payment/{payment_id}`
  - Respuesta: `{ detail: "Pago cancelado correctamente." }`
- `POST /create_payment_method`
  - Body: `create_payment_method_req` (tarjeta o billetera según `tipo_pago_id`).

    - Respuesta: `create_payment_method_out` (incluye ids de catálogos y datos resumidos como últimos 4 dígitos).
- `GET /get_payment_methods/{usuario_id}`
    - Respuesta: lista de `get_payment_method` (campos enriquecidos: tipo/brand/gateway, últimos 4, titular, banco, wallet, es_default).
- `PATCH /update_payment_method/{pm_id}`
    - Body: `update_payment_method_req`.
    - Respuesta: método actualizado.
- `DELETE /delete_payment_method/{pm_id}`
    - Respuesta: `200` al dar de baja lógica (si no es default).

Interfaz asíncrona (RabbitMQ)

- Consumidores (`app/rabbit_consumer.py`):
    - Cola `payments`: crea pagos. Payload: `{ "order_id": 42, "metodo_pago_id": 1 }`.
    - Cola `payments_cancel`: cancela pagos. Payload: `{ "payment_id": 1 }`.
    - Cola `payments_set_status` (env `PAYMENTS_SET_STATUS_QUEUE`): cambia estado del pago.
        - Por id: `{ "payment_id": 123, "status_id": 2 }`.
        - Por nombre: `{ "payment_id": 123, "new_status": "Realizado" }` o `"Fallido"`.
- Publicación de eventos (`app/rabbit_publisher.py` + `app/services/payments.py`):
    - Cola `payments_status` (env `PAYMENTS_STATUS_QUEUE`): emite `{ event: "payment_status_changed", payment_id, order_id, previous_status, new_status, changed_at }` al crear/cancelar/actualizar estado.
- Ejemplos rápidos en `mensajes_rabbit`.