Registrar Pago

Permite a una pasarela de pagos autenticada (por ejemplo, Palomma) **registrar el pago** de uno o varios ítems pagables en la inmobiliaria: una factura, un movimiento (concepto / interés) o una factura junto con sus intereses adjuntos. El endpoint es idempotente, devuelve un cuerpo en formato de arreglo y aplica automáticamente la configuración de la pasarela (forma de pago, envío DIAN).

> **¿Para qué sirve este servicio?**
> Está pensado exclusivamente para integraciones con pasarelas de pago. La pasarela lo invoca cuando el deudor confirma el pago en línea de los ítems devueltos previamente por **GET /gateways/portfolio**. El sistema persiste el pago, genera el recibo de caja y, cuando la configuración lo requiera, dispara el envío del documento electrónico a la DIAN.

> **Endpoint restringido a pasarelas registradas**
> Este endpoint **solo** puede ser consumido por clientes OAuth2 cuyo `client_name` esté registrado como una pasarela activa en la inmobiliaria. Cualquier otro cliente recibirá `403 Forbidden`.

---

1. El Endpoint (La dirección web)

Apunta tu sistema a la siguiente dirección. Recuerda reemplazar **{{instancia}}** por la dirección web completa que utilizas para ingresar a tu plataforma.

```
POST https://{{instancia}}/service/v2/public/gateways/payments
```

> **¿Qué debes colocar en {{instancia}}?**
> Es muy sencillo: corresponde a la **dirección web principal** que utilizas a diario para ingresar a tu plataforma (incluyendo la terminación `.nuby.app` o `.arrendasoft.co`).
> Por ejemplo, si para entrar a tu sistema escribes `inmobiliaria.nuby.app` o `inmobiliaria.arrendasoft.co` en tu navegador, esa será exactamente tu instancia. Solo asegúrate de no incluir el "https://" ni barras diagonales ("/") al final.

2. La Petición (¿Qué debes enviarnos?)

Debes enviar una petición **POST** con el detalle del pago en el cuerpo de la solicitud en formato JSON. La pasarela se identifica automáticamente a partir del `client_name` del token; **no debes enviar el nombre de la pasarela ni la forma de pago en el body**: ambos se derivan de la configuración registrada.

Método	POST
Content-Type	application/json
Authorization	**Bearer token**, Token obtenido al consumir el servicio **Login** con un cliente OAuth2 registrado como pasarela activa.

> **Autenticación requerida**
> Este servicio requiere un Token de autenticación válido. Debes incluir el encabezado `Authorization: Bearer TU_TOKEN` en cada petición. El token se obtiene consumiendo el servicio de **Login**. El cliente OAuth además debe estar registrado como pasarela activa; en caso contrario el endpoint responderá `403`.

Cuerpo de la petición (Body):

Campo	Tipo	Requerido	Descripción
factura_id	integer	Condicional	Identificador de la factura a pagar. Debe existir en el sistema. Es **obligatorio** si no se envía `movimiento_id`.
movimiento_id	integer | array	Condicional	Identificador (o lista de identificadores) de los movimientos a pagar. Es **obligatorio** si no se envía `factura_id`. Cuando se envía sin `factura_id`, por defecto solo se admite **un único** movimiento; sin embargo, si la pasarela tiene activo el filtro **«Agrupar conceptos del mismo periodo»**, se aceptan **varios movimientos del mismo tercero** y se procesan en **modo GROUP** (un solo recibo de caja + cuando aplique una factura agrupada por contrato).

Campo	Tipo	Requerido	Descripción
monto	number	Sí	Valor pagado por el deudor. Debe ser un número mayor a cero. Si excede el saldo adeudado, el excedente se registra como **anticipo**.
fecha_pago	string	Sí	Fecha y hora del pago. Formato estricto: `YYYY-MM-DD HH:MM:SS`.
n_comprobante	string	No	Número de comprobante de la pasarela. Máximo **100** caracteres. Solo letras, números, guiones, guiones bajos, puntos y espacios. **Recomendado**: garantiza la idempotencia ante reintentos.

> **Modos de operación**
> El endpoint clasifica el pago en uno de cuatro modos según los campos enviados y la configuración de la pasarela:
> — **Modo movimiento único**: `factura_id = null` y `movimiento_id = [n]`.
> — **Modo GROUP**: `factura_id = null` y `movimiento_id = [m1, m2, ...]` con varios movimientos del mismo tercero. Solo se admite cuando la pasarela tiene activo el filtro avanzado **«Agrupar conceptos del mismo periodo»**. Genera un único recibo de caja y, cuando aplique, una factura agrupada por contrato.
> — **Modo factura**: `factura_id = X` y `movimiento_id = []` o ausente.
> — **Modo factura + intereses adjuntos**: `factura_id = X` y `movimiento_id = [m1, m2, ...]`. Los movimientos que pertenezcan a la factura X se ignoran (ya están cubiertos); los demás se procesan como intereses sueltos en la misma transacción.

Ejemplo 1: Pago de una sola factura

```
{
    "factura_id": 4521,
    "monto": 1750000.00,
    "fecha_pago": "2026-04-25 14:30:00",
    "n_comprobante": "PALM-2026-04-25-998877"
}
```

Ejemplo 2: Pago de un movimiento (concepto suelto)

```
{
    "movimiento_id": 98515,
    "monto": 280000.00,
    "fecha_pago": "2026-04-25 14:30:00",
    "n_comprobante": "PALM-2026-04-25-998878"
}
```

Ejemplo 3: Pago de una factura junto con sus intereses adjuntos

```
{
    "factura_id": 4521,
    "movimiento_id": [98410, 98411],
    "monto": 1820000.00,
    "fecha_pago": "2026-04-25 14:30:00",
    "n_comprobante": "PALM-2026-04-25-998879"
}
```

3. La Respuesta (¿Qué te entregaremos?)

El sistema responde con HTTP `200` cuando el pago se registra correctamente. La clave `body` es **siempre un arreglo**, incluso cuando el pago resulta en un solo registro. Cada elemento del arreglo describe un documento generado por la operación: la factura pagada, una factura nueva creada por intereses facturables, o un anticipo si hubo excedente.

```
{
    "success": true,
    "status": 200,
    "message": "El pago fue registrado exitosamente.",
    "body": [
        {
            "factura_id": 4521,
            "movimiento_id": null,
            "pago_id": 87123,
            "recibo_id": 56412,
            "documento_contable_id": 102345,
            "monto_pagado": 1750000.00,
            "fecha_pago": "2026-04-25 14:30:00",
            "forma_pago_id": 7,
            "forma_pago": "Pasarela Palomma",
```

```
            "n_comprobante": "PALM-2026-04-25-998877",
            "saldo_anterior": 1750000.00,
            "saldo_actual": 0.00,
            "estado": "pagada",
            "mensaje": "El pago cubrió el total de la factura.",
            "estado_dian": "pendiente",
            "gateway": "palomma",
            "client_name": "palomma_inmobiliaria_xyz"
        }
    ],
    "alertas": [],
    "data": {
        "factura_id": 4521,
        "pago_id": 87123,
        "recibo_id": 56412,
        "documento_contable_id": 102345,
        "confirm_pay_id": 9921,
        "gateway": "palomma",
        "client_name": "palomma_inmobiliaria_xyz",
        "documentos_generados": [
            { "tipo": "recibo", "id": 56412, "numero": "REC-56412" },
            { "tipo": "factura", "id": 4521, "numero": "FAC-4521" }
        ]
    }
}
```


Ejemplo de respuesta con excedente (anticipo):


```
{
    "success": true,
    "status": 200,
    "message": "El pago fue registrado exitosamente.",
    "body": [
        {
            "factura_id": 4521,
            "movimiento_id": null,
            "pago_id": 87124,
            "recibo_id": 56413,
            "documento_contable_id": 102346,
            "monto_pagado": 1750000.00,
```

```
            "fecha_pago": "2026-04-25 14:30:00",
            "forma_pago_id": 7,
            "forma_pago": "Pasarela Palomma",
            "n_comprobante": "PALM-2026-04-25-998880",
            "saldo_anterior": 1750000.00,
            "saldo_actual": 0.00,
            "estado": "pagada",
            "mensaje": "El pago cubrió el total de la factura.",
            "estado_dian": "pendiente",
            "gateway": "palomma",
            "client_name": "palomma_inmobiliaria_xyz"
        },
        {
            "factura_id": null,
            "movimiento_id": null,
            "pago_id": null,
            "recibo_id": null,
            "documento_contable_id": 102347,
            "monto_pagado": 50000.00,
            "fecha_pago": "2026-04-25 14:30:00",
            "forma_pago_id": 7,
            "forma_pago": "Pasarela Palomma",
            "n_comprobante": "PALM-2026-04-25-998880",
            "saldo_anterior": null,
            "saldo_actual": null,
            "estado": "anticipo",
            "mensaje": "Se generó un anticipo por valor de 50000.00 con el excedente del
pago.",
            "estado_dian": null,
            "gateway": "palomma",
            "client_name": "palomma_inmobiliaria_xyz"
        }
    ],
    "alertas": [],
    "data": {
        "factura_id": 4521,
        "pago_id": 87124,
        "recibo_id": 56413,
        "documento_contable_id": 102346,
        "anticipo_documento_contable_ids": [102347],
```

```
        "confirm_pay_id": 9922,
        "gateway": "palomma",
        "client_name": "palomma_inmobiliaria_xyz"
    }
}
```

Campos de la respuesta

Clave	Descripción
success	Indica si la operación fue exitosa (`true`) o no (`false`).
status	Código HTTP devuelto.
message	Mensaje descriptivo del resultado.
body	Arreglo con uno o más registros generados por la operación (ver tabla detallada abajo).
alertas	Arreglo de alertas no bloqueantes generadas durante el proceso (por ejemplo, fallas en el envío DIAN).
data	Objeto con la información consolidada del pago (IDs principales, identificador de confirmación, gateway, cliente). Adicionalmente puede incluir `documentos_generados`: un arreglo de objetos `{tipo, id, numero}` que lista todos los documentos creados u afectados por el pago (recibo de caja y factura(s)). El campo `tipo` puede ser `"recibo"` o `"factura"`; `numero` es el consecutivo legible para el usuario final.
error_code	Solo presente en respuestas de error. Códigos posibles: `"DUPLICATE_PAYMENT"`, `"VALIDATION_ERROR"`, `"GATEWAY_NOT_FOUND"`, `"GATEWAY_NOT_ACTIVE"`, `"GATEWAY_PAYMENT_METHOD_REQUIRED"`, `"INTERNAL_ERROR"`.
duplicate	Solo presente en respuestas `409 Conflict`. Contiene el snapshot del pago previamente registrado.

Estructura de cada elemento de `body[]`

Clave	Descripción
factura_id	Identificador de la factura pagada o generada. `null` para registros de anticipo.
movimiento_id	Identificador (o arreglo de IDs) de los movimientos asociados al registro. Puede ser `null`.
pago_id	Identificador del pago creado en el sistema. `null` para anticipos.
recibo_id	Identificador del recibo de caja generado.
documento_contable_id	Identificador del documento contable asociado (recibo, anticipo, etc.).

Clave	Descripción
monto_pagado	Valor efectivamente aplicado a la factura/movimiento. `null` para anticipos sin asignación específica.
fecha_pago	Fecha y hora del pago. Formato: `YYYY-MM-DD HH:MM:SS`.
forma_pago_id	Identificador interno de la forma de pago contable (derivada de la configuración de la pasarela).
forma_pago	Nombre legible de la forma de pago.
n_comprobante	Número de comprobante enviado por la pasarela.
saldo_anterior	Saldo de la factura/movimiento antes de aplicar el pago.
saldo_actual	Saldo restante después de aplicar el pago.
estado	Estado del registro tras la operación. Valores posibles: • `"pagada"`: el monto cubrió el saldo total de la factura (`saldo_actual = 0`). • `"pendiente"`: el monto fue inferior al saldo y la factura sigue con saldo pendiente. El campo `mensaje` describe el saldo restante. • `"anticipo"`: la fila representa un excedente registrado como anticipo (sin factura asociada).
mensaje	Mensaje legible que describe el resultado del pago a nivel de cada fila. Valores típicos: • `"El pago cubrió el total de la factura."` cuando `estado = "pagada"`. • `"El pago fue registrado parcialmente. La factura aún tiene un saldo pendiente de {monto}."` cuando `estado = "pendiente"`. • `"Se generó un anticipo por valor de {monto} con el excedente del pago."` cuando `estado = "anticipo"`.
estado_dian	Estado del envío DIAN. Valores posibles: `"pendiente"` (envío en cola o fallido) o `null` (no aplica). El detalle del fallo se reporta en `alertas`.
gateway	Slug de la pasarela autenticada (por ejemplo, `"palomma"`).
client_name	Nombre del cliente OAuth2 autenticado.

Escenarios especiales del cuerpo

Escenario	Comportamiento del `body`
Pago simple	Un único elemento que describe el recibo generado.
Intereses facturables	Al menos dos elementos: uno por la factura original pagada y uno por la factura nueva creada para los intereses facturables.

Escenario	Comportamiento del `body`
Excedente / anticipo	Cuando el monto pagado supera el saldo de los ítems enviados, el sistema genera un **anticipo** con el sobrante. La respuesta incluirá una fila adicional con `estado = "anticipo"`, `documento_contable_id` poblado, `monto_pagado` con el valor del anticipo, y los campos `fecha_pago`, `forma_pago_id`, `forma_pago` y `n_comprobante` heredados del pago original. Los campos `factura_id`, `movimiento_id`, `pago_id`, `recibo_id`, `saldo_anterior` y `saldo_actual` serán `null`. **Importante:** el sistema garantiza que primero se cubre el saldo de cada item enviado (factura y/o movimientos) y solo el sobrante real se transforma en anticipo; nunca se desvía a anticipo el monto destinado a un movimiento explícitamente declarado.
Falla en envío DIAN	El pago queda persistido (`estado = "pagada"`), `estado_dian = "pendiente"`, y `alertas` contiene una entrada `{ "level": "warning", "code": "DIAN_SEND_FAILED", "message": "...", "factura_id": ... }`.

4. Idempotencia

El endpoint detecta automáticamente reintentos de un mismo pago y responde `409 Conflict` sin volver a registrarlo. Las reglas de idempotencia son:

Disparador	Comportamiento
Mismo `n_comprobante`	Cuando el `n_comprobante` ya fue registrado para la misma pasarela, se devuelve `409` con el registro previo.
Sin `n_comprobante`, mismos datos esenciales	Si los datos `(factura_id, movimiento_ids ordenados, monto, fecha_pago, client_name)` coinciden con un pago previo, se devuelve `409`.
`n_comprobante` distinto, mismos datos esenciales	Se aplica un hash de respaldo sobre los datos esenciales y se devuelve `409` si coincide con un pago previo.

> **Cómo interpretar el 409**
> Una respuesta `409 Conflict` de este endpoint **no es un fallo**: significa que el pago **ya existe** en nuby. La pasarela debe tratarla como confirmación idempotente y **no reintentar**. El cuerpo incluye `error_code = "DUPLICATE_PAYMENT"`, `is_business_error = true` y, en `data`, los identificadores del pago original (`confirm_pay_id`, `confirm_pay_reference_code`, `factura_id`, `pago_id`, `recibo_id`, `documento_contable_id`).

Ejemplo de respuesta 409:

```
{
    "success": false,
    "status": 409,
    "message": "El pago ya fue registrado previamente para la pasarela
palomma_inmobiliaria_xyz.",
    "body": [ /* snapshot del body original */ ],
    "error_code": "DUPLICATE_PAYMENT",
    "data": {
        "factura_id": 4521,
        "pago_id": 87123,
        "recibo_id": 56412,
        "documento_contable_id": 102345,
        "confirm_pay_id": 9921,
        "confirm_pay_reference_code": "PALM-2026-04-25-998877",
        "gateway": "palomma",
        "client_name": "palomma_inmobiliaria_xyz"
    },
    "duplicate": {
        "payload": { /* payload original recibido en el primer intento */ }
    }
}
```

5. Seguridad y Posibles Errores

El sistema realiza validaciones de autenticación, autorización por pasarela, configuración y consistencia del payload. Si alguna falla, devolverá un error con su respectivo código HTTP. La estructura siempre incluye `success: false`, `status`, `message` y, cuando aplica, `error_code`.

Código HTTP	Descripción
400	**Token faltante o inválido.** Posibles causas: — No se envió el encabezado `Authorization`. — El encabezado no tiene el formato `Bearer {token}`. — El token no fue encontrado en el sistema.
401	El token ha expirado. Debes generar uno nuevo consumiendo el servicio de **Login**.
403	El cliente OAuth autenticado no está registrado como pasarela activa en la inmobiliaria.
409	**Pago duplicado.** El pago ya fue registrado previamente. `error_code = "DUPLICATE_PAYMENT"`. La pasarela debe tratar la respuesta como confirmación idempotente y **no reintentar**. Ver sección de Idempotencia.

Código HTTP	Descripción
422	**Error de validación.** `error_code = "VALIDATION_ERROR"`. Posibles causas: — No se envió ni `factura_id` ni `movimiento_id`. — `factura_id` no es un entero positivo o no existe. — Algún `movimiento_id` no es un entero positivo o no existe. — Sin `factura_id` se envió más de un `movimiento_id`. — `monto` no es numérico o es menor o igual a cero. — `fecha_pago` no cumple el formato `YYYY-MM-DD HH:MM:SS`. — `n_comprobante` supera los 100 caracteres o contiene caracteres no permitidos. — La `factura_id` y los `movimiento_id` pertenecen a terceros distintos. — La pasarela no tiene una forma de pago configurada (`error_code = "GATEWAY_PAYMENT_METHOD_REQUIRED"`). — El cliente no corresponde a una pasarela registrada (`error_code = "GATEWAY_NOT_FOUND"`) o la pasarela está inactiva (`error_code = "GATEWAY_NOT_ACTIVE"`). — Errores de negocio retornados por el motor de ingresos (por ejemplo, factura ya pagada).
500	Error interno al registrar el pago. `error_code = "INTERNAL_ERROR"`. El detalle se registra en el canal de log `PASARELAS`.

6. Ejemplos de integración

Aquí tienes ejemplos de código listos para que tus desarrolladores los adapten:

cURL

```
# Pago de una factura
curl -X POST "https://{{instancia}}/service/v2/public/gateways/payments" \
-H "Content-Type: application/json" \
-H "Authorization: Bearer TU_TOKEN_AQUI" \
-d '{
  "factura_id": 4521,
  "monto": 1750000.00,
  "fecha_pago": "2026-04-25 14:30:00",
  "n_comprobante": "PALM-2026-04-25-998877"
}'

# Pago de una factura junto con sus intereses adjuntos
curl -X POST "https://{{instancia}}/service/v2/public/gateways/payments" \
```

```
-H "Content-Type: application/json" \
-H "Authorization: Bearer TU_TOKEN_AQUI" \
-d '{
  "factura_id": 4521,
  "movimiento_id": [98410, 98411],
  "monto": 1820000.00,
  "fecha_pago": "2026-04-25 14:30:00",
  "n_comprobante": "PALM-2026-04-25-998879"
}'
```

PHP

```
<?php

$instance = 'tu_instancia';
$token = 'TU_TOKEN_AQUI'; // Token de un cliente OAuth registrado como pasarela

$url = "https://{$instance}/service/v2/public/gateways/payments";

$payload = [
    'factura_id'     => 4521,
    'monto'          => 1750000.00,
    'fecha_pago'     => '2026-04-25 14:30:00',
    'n_comprobante'  => 'PALM-2026-04-25-998877',
];

$ch = curl_init($url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($payload));
curl_setopt($ch, CURLOPT_HTTPHEADER, [
    'Content-Type: application/json',
    "Authorization: Bearer {$token}",
]);

$response = curl_exec($ch);
$http_code = curl_getinfo($ch, CURLINFO_HTTP_CODE);
```

```
curl_close($ch);

echo "Código HTTP: {$http_code}\n";
$data = json_decode($response, true);

if ($http_code === 200) {
    echo "Pago registrado. Recibo: {$data['data']['recibo_id']}\n";
} elseif ($http_code === 409) {
    echo "Pago ya existía previamente. confirm_pay_id:
{$data['data']['confirm_pay_id']}\n";
    // No reintentar: el pago ya está registrado en nuby.
} else {
    echo "Error registrando el pago:\n{$response}\n";
}

?>
```

Python

```
import requests

# 1. Configura tus datos de acceso
instancia = 'mi-inmobiliaria.nuby.app'
token = 'TU_TOKEN_AQUI'  # Token de un cliente OAuth registrado como pasarela

# 2. Prepara el body del pago
url = f"https://{instancia}/service/v2/public/gateways/payments"

payload = {
    "factura_id": 4521,
    "monto": 1750000.00,
    "fecha_pago": "2026-04-25 14:30:00",
    "n_comprobante": "PALM-2026-04-25-998877",
}

headers = {
    "Content-Type": "application/json",
```

```
    "Authorization": f"Bearer {token}",
}

# 3. Envía la petición POST y procesa la respuesta
try:
    response = requests.post(url, json=payload, headers=headers)
    print(f"Código HTTP: {response.status_code}")
    data = response.json()

    if response.status_code == 200:
        print(f"Pago registrado. Recibo: {data['data']['recibo_id']}")
    elif response.status_code == 409:
        print(f"Pago ya existía previamente. confirm_pay_id:
{data['data']['confirm_pay_id']}")
        # No reintentar: el pago ya está registrado en nuby.
    else:
        print("Error registrando el pago.")
        print(response.text)
except Exception as e:
    print(f"Error de conexión: {e}")
```

JavaScript

```
// 1. Configura tus datos de acceso
const instancia = 'mi-inmobiliaria.nuby.app';
const token = 'TU_TOKEN_AQUI'; // Token de un cliente OAuth registrado como pasarela

// 2. Prepara el body del pago
const url = `https://${instancia}/service/v2/public/gateways/payments`;

const payload = {
    factura_id: 4521,
    monto: 1750000.00,
    fecha_pago: '2026-04-25 14:30:00',
    n_comprobante: 'PALM-2026-04-25-998877',
};

// 3. Función para registrar el pago
```

```
async function registrarPago() {
    try {
        const response = await fetch(url, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${token}`,
            },
            body: JSON.stringify(payload),
        });

        console.log(`Código HTTP: ${response.status}`);
        const data = await response.json();

        if (response.status === 200) {
            console.log(`Pago registrado. Recibo: ${data.data.recibo_id}`);
        } else if (response.status === 409) {
            console.log(`Pago ya existía previamente. confirm_pay_id:
${data.data.confirm_pay_id}`);
            // No reintentar: el pago ya está registrado en nuby.
        } else {
            console.log('Error registrando el pago.');
            console.log(data);
        }
    } catch (error) {
        console.error(`Error de conexión: ${error}`);
    }
}

registrarPago();
```

Revisión #8
Creado el 27 abril 2026 14:17:29 por Isabel Higuita
Actualizado el 27 mayo 2026 13:54:17 por Isabel Higuita