Mercado Pago Orders API Integration

Overview

The billing service now uses the Mercado Pago Orders API (`/v1/orders`) instead of the Payments API for processing PIX payments. This provides better support for order-based payment flows.

Key Features

1. Orders API Implementation

- **Endpoint**: `https://api.mercadopago.com/v1/orders`
- **Method**: POST
- **Authentication**: Bearer token in Authorization header
- **Idempotency**: Each request includes a unique `X-Idempotency-Key` UUID header

2. Static Request Data

All order requests use standardized static values except for:

- **Email**: Provided from payment request or defaults to `test@testuser.com`
- **Amount**: Provided from payment request

Static values:

```
{
  "type": "online",
  "external_reference": "order_ref_<UUID>",
  "payer": {
    "first_name": "APRO"
  },
  "transactions": {
    "payments": [
      {
        "payment_method": {
          "id": "pix",
          "type": "bank_transfer"
        }
      }
    ]
  }
}
```

PROF

3. Response Persistence

Payment responses from Mercado Pago are automatically:

- Persisted to the PostgreSQL database
- Sent to the appropriate SQS queue based on status:
  - **Success Queue**: `payment-response-success-queue` (for APPROVED/PROCESSING statuses)
  - **Failure Queue**: `payment-response-failure-queue` (for REJECTED/FAILED statuses)

Configuration

Access Token

Configure your Mercado Pago access token in the application configuration:

```
mercadopago:
  access-token: APP_USR-your-access-token-here
```

Note: The Orders API requires a production access token (starting with `APP_USR-`), not test tokens.

SQS Queues

Ensure the following queues are configured:

```
aws:
  sqs:
    payment-request-queue: payment-request-queue
    payment-response-success-queue: payment-response-success-queue
    payment-response-failure-queue: payment-response-failure-queue
```

Testing

1. Send a Payment Request

PROF

Send a message to the `payment-request-queue` with the following structure:

```
{
  "order_id": "a1b2c3d4-e5f6-4a7b-8c9d-0e1f2a3b4c5d",
  "client_id": "f1e2d3c4-b5a6-4d7e-8f9a-0b1c2d3e4f5a",
  "amount": 280.0,
  "customer_email": "test@testuser.com",
  "customer_name": "John Doe",
  "description": "Payment for order #1234"
}
```

2. Monitor the Response

The service will:

1. Create a payment record in the database
2. Call Mercado Pago Orders API with a unique idempotency key
3. Update the payment record with the response
4. Send the response to the appropriate SQS queue

3. Check the Response Queue

Monitor the success queue for the response:

```
{
  "paymentId": "...",
  "workOrderId": "a1b2c3d4-e5f6-4a7b-8c9d-0e1f2a3b4c5d",
  "clientId": "f1e2d3c4-b5a6-4d7e-8f9a-0b1c2d3e4f5a",
  "status": "APPROVED",
  "amount": 280.0,
  "externalPaymentId": "mercadopago-order-id",
  "paymentMethod": "pix",
  "qrCode": "00020101021243650016COM.MERCADOLIBRE...",
  "qrCodeBase64": "iVBORw0KGgoAAAANSUhEUgAA...",
  "createdAt": "2026-02-13T10:30:00",
  "processedAt": "2026-02-13T10:30:05"
}
```

API Request Details

Headers

```
Authorization: Bearer {access-token}
X-Idempotency-Key: {random-uuid}
Content-Type: application/json
```

PROF

Request Body Structure

```
{
  "type": "online",
  "external_reference": "order_ref_{uuid}",
  "total_amount": "280.00",
  "payer": {
    "email": "test@testuser.com",
    "first_name": "APRO"
  },
  "transactions": {
    "payments": [
      {
        "amount": "280.00",
        "payment_method": {
          "id": "pix",
          "type": "bank_transfer"
        }
      }
    ]
  }
}
```

Implementation Details

Components Modified

1. **MercadoPagoAdapter** (`infrastructure/adapter/out/payment/MercadoPagoAdapter.java`)
    - Replaced SDK-based payment creation with direct REST API calls
    - Implements Orders API endpoint
    - Generates random UUID for each idempotency key
    - Uses RestTemplate for HTTP communication
2. **PaymentResponseMessageAdapter** (`infrastructure/adapter/out/messaging/PaymentResponseMessageAdapter.java`)
    - Implements actual SQS message sending
    - Routes messages to success/failure queues based on status
    - Serializes payment data to JSON
3. **New DTOs**
    - `MercadoPagoOrderRequest`: Request structure for Orders API
    - `MercadoPagoOrderResponse`: Response structure from Orders API

Flow Diagram

PROF

```
┌─────────────┐
│ SQS Request │
│   Queue     │
└──────┬──────┘
       │
       ▼
┌─────────────────┐
│ Payment Queue   │
│   Listener      │
└──────┬──────────┘
       │
       ▼
┌─────────────────┐      ┌────────────────┐
│ Process Payment │─────▶│ MercadoPago    │
│   Service       │      │ Orders API     │
```

```
└─────────────────┘     └──────────────────┘
         │
         │
         ▼
┌─────────────────┐
│    Save to DB   │
└─────────────────┘
         │
         │
         ▼
┌─────────────────┐
│  Send Response  │
│   to SQS        │
└─────────────────┘
```

Error Handling

- API errors are logged and wrapped in `RuntimeException`
- Payment status is marked as FAILED on errors
- Failed payments trigger error messages to the failure queue
- SQS sending errors are logged but don't break the payment flow

Troubleshooting

Common Issues

1. **401 Unauthorized**
    - Check if access token is valid
    - Ensure token is production token (APP_USR-*)
2. **Queue not found**
    - Verify queue names in configuration
    - Check AWS region configuration (us-east-2)
    - Ensure queues exist in AWS

PROF

3. **Missing QR Code**
    - Check response from Mercado Pago
    - Verify PIX is enabled in your Mercado Pago account
    - Check logs for transaction data structure

Monitoring

Key log messages to monitor:

- `Processing PIX payment through Mercado Pago Orders API`
- `Calling Mercado Pago Orders API with idempotency key`
- `Mercado Pago order created: id={}, status={}`
- `Payment response sent successfully to queue`