Heritage Bookshop Sales Integration using Mulesoft

Author
Faith Ann Baraclan

1. Solution Overview

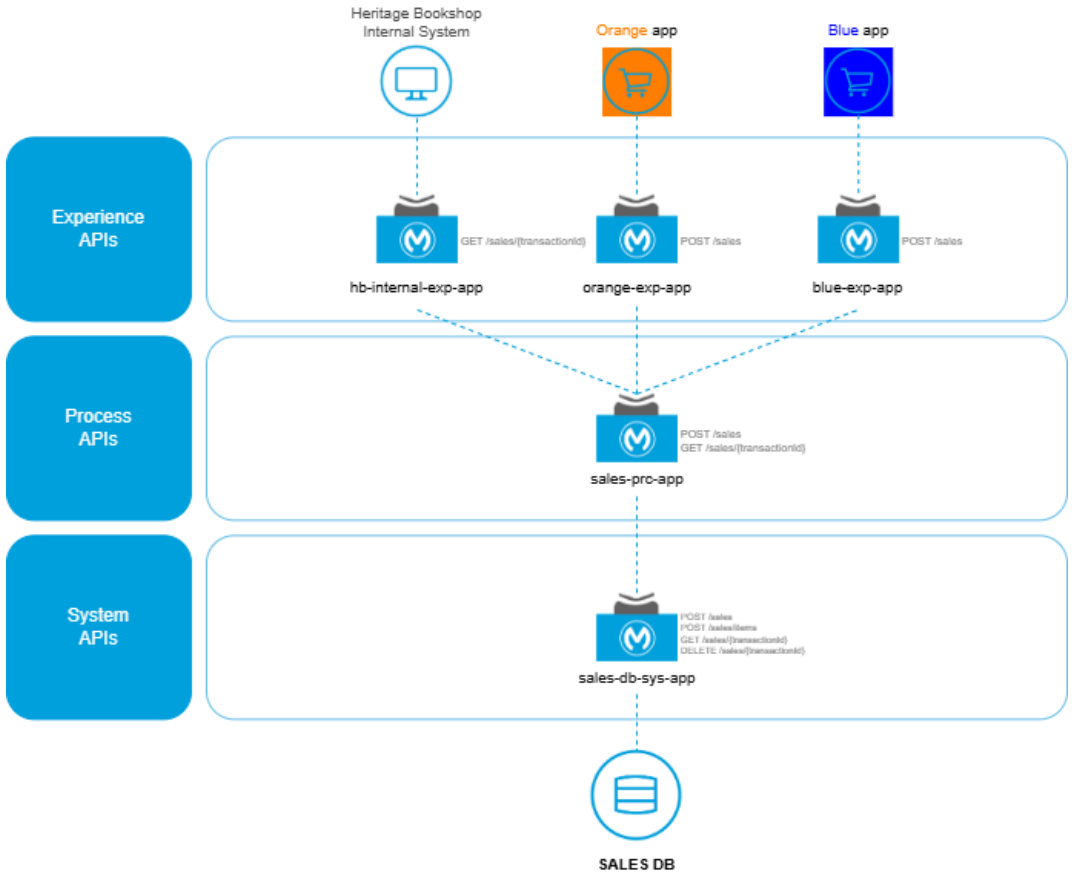
This solution follows Mulesoft's API-Led Connectivity approach, organizing integration logic into three key layers: Experience APIs, Process APIs, and System APIs.

The overall objective is to expose sales-related capabilities from the **Sales Database** to multiple consuming channels, including

- Heritage Bookshop's internal system
- *Orange* app
- *Blue* app

2. Architecture Overview

API-Led Connectivity: Diagram



Created using draw.io

Experience layer: orange-exp-app, blue-exp-app, hb-internal-exp-app
Process layer: sales-prc-app
System layer: sales-db-sys-app

3. Experience APIs

Experience APIs provide consumer-specific representations of the underlying business data. They tailor the data format, orchestration, and logic to meet the needs of each consuming application.

hb-internal-exp-app

- Consumer: Heritage Bookshop internal system
- Exposes: `GET /sales/{transactionId}`
- Purpose: Retrieves sales information to support internal business process

orange-exp-app

- Consumer: *Orange* app
- Exposes: `POST /sales`
- Purpose: Allows *Orange* app to create new sales to Heritage Bookshop's database

blue-exp-app

- Consumer: *Blue* app
- Exposes: `GET /sales/{transactionId}`
- Purpose: Allows *Blue* app to create new sales to Heritage Bookshop's database

Remarks: Created one experience API for each consumer as each consumer sends different payload formats. By isolating them, in the event of changes for one consumer, the other consumers won't be affected. This follows Mulesoft's API-led best practice wherein Experience APIs must be consumer-specific.

4. Process API

Process APIs centralizes business logic across channels/consumers. It coordinates data flows between the Experience APIs and System APIs.

sales-prc-app

- Exposes:
  - `POST /sales`
  - `GET /sales/{transactionId}`
- Responsibilities:
  - Transform experience API payloads into internal or standard data sales structure
  - Validate and route incoming sales data
  - Invoke the system API to interact with the Sales Database

Remarks: This process API ensures consistency across all channels.

5. System API

System APIs provide direct and reusable access to backend systems, i.e. database for this project.

sales-db-sys-app

- Exposes:
  - `POST /sales`
  - `POST /sales/items`
  - `GET /sales/{transactionId}`
  - `DELETE /sales/{transactionId}`
- Responsibilities:
  - Perform CRUD operations against the **Sales Database**
  - `POST /sales` - insert new sales transactions into ***sales*** *Table*
  - `POST /sales/items` - insert new sales transactions into ***sales_items*** *Table*
  - `GET /sales/{transactionId}`- retrieves a single sales detail from the **Sales Database**
  - `DELETE /sales/{transactionId}`- deletes a transaction from the ***sales*** *Table*

Remarks: This layer ensures security, reusability, and controlled access to backend systems.

6. Database

6.1 Sales Database Overview

The **Sales Database** serves as the system of record for all sales-related information in the Heritage Bookshop ecosystem.

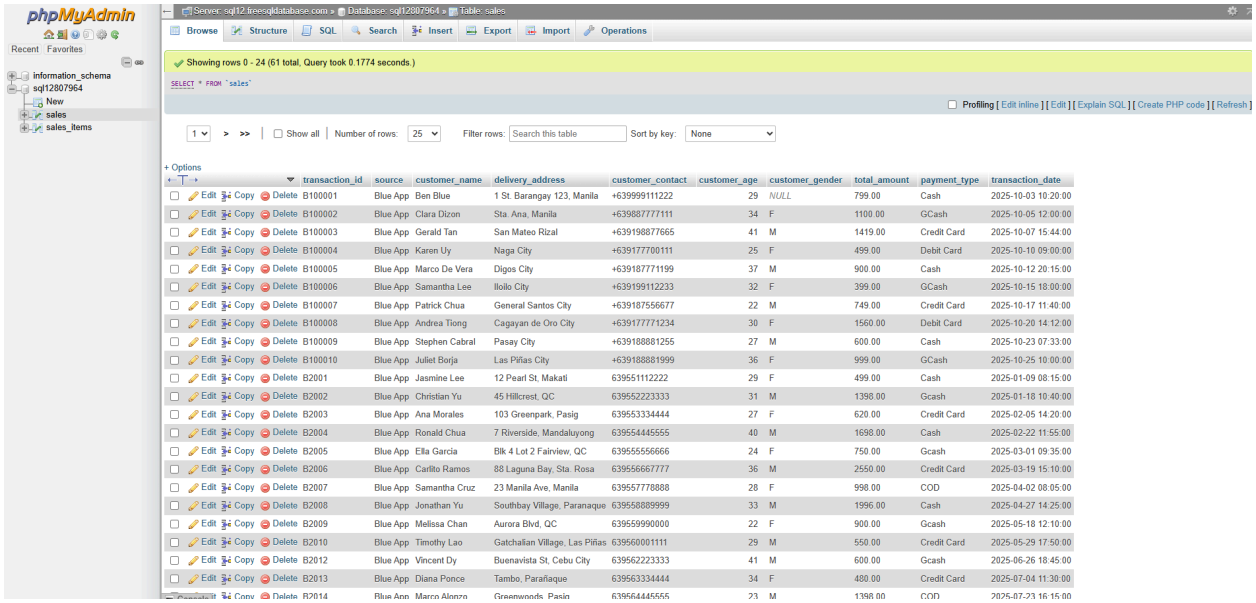
As per requirements, there are 2 tables created:

- Sales Table
- Sales Items Table

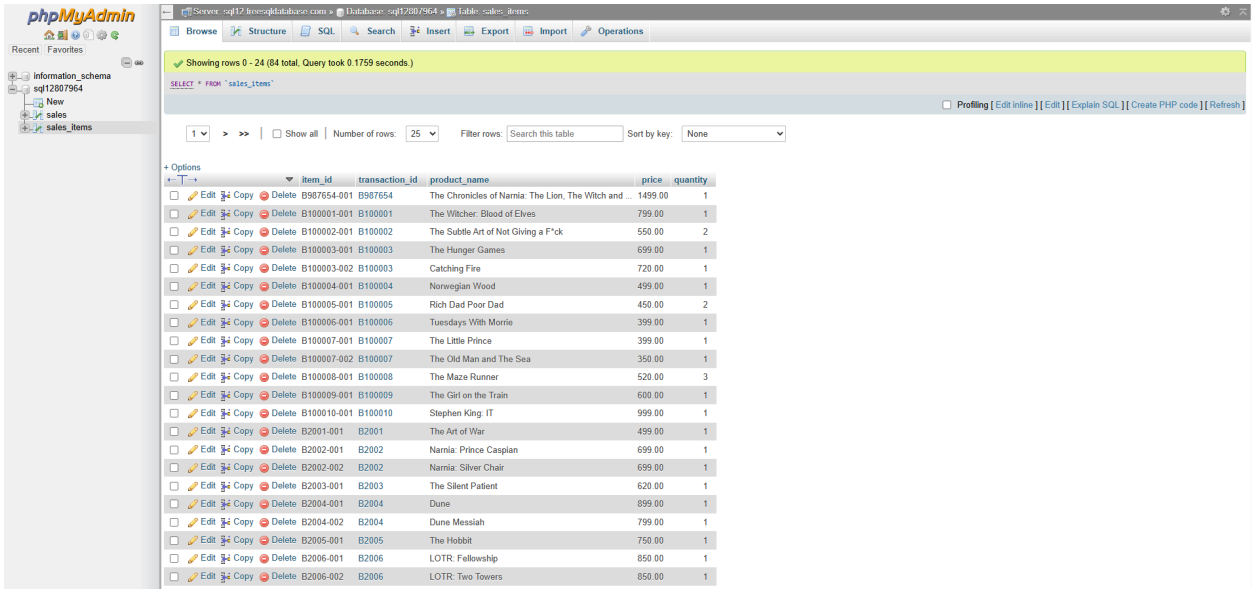
6.2 Setup

- Created a free database account here: https://www.freesqldatabase.com/
- Created the tables directly in the database using the provided SQL for table creation

6.3 Sales Table

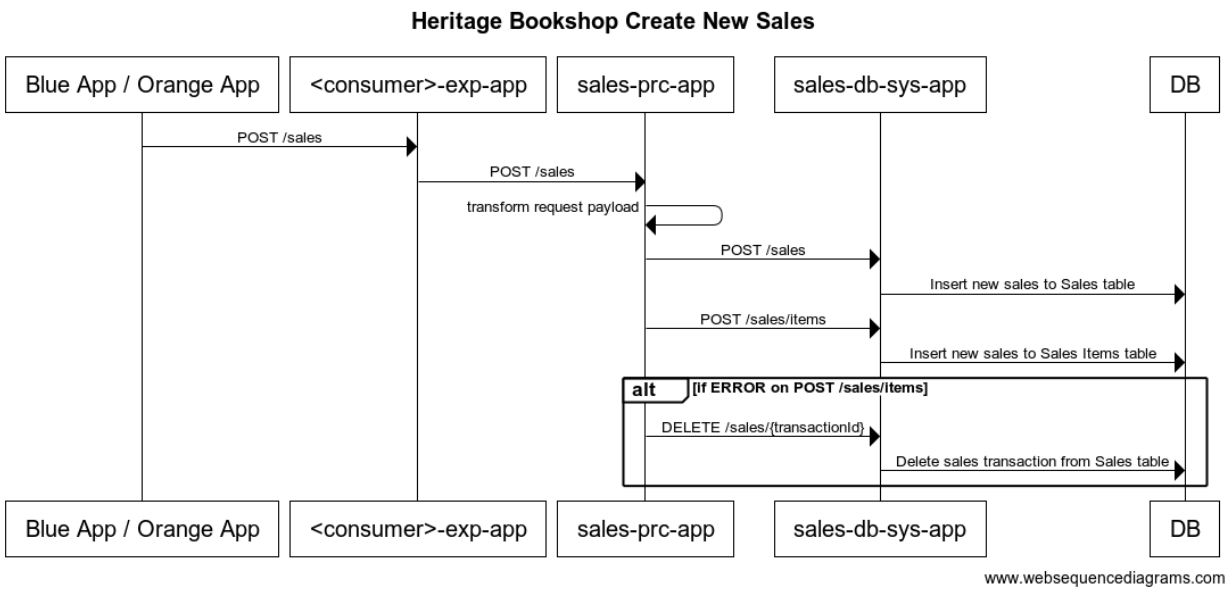


6.4 Sales Items Table



7. Sequence Diagrams

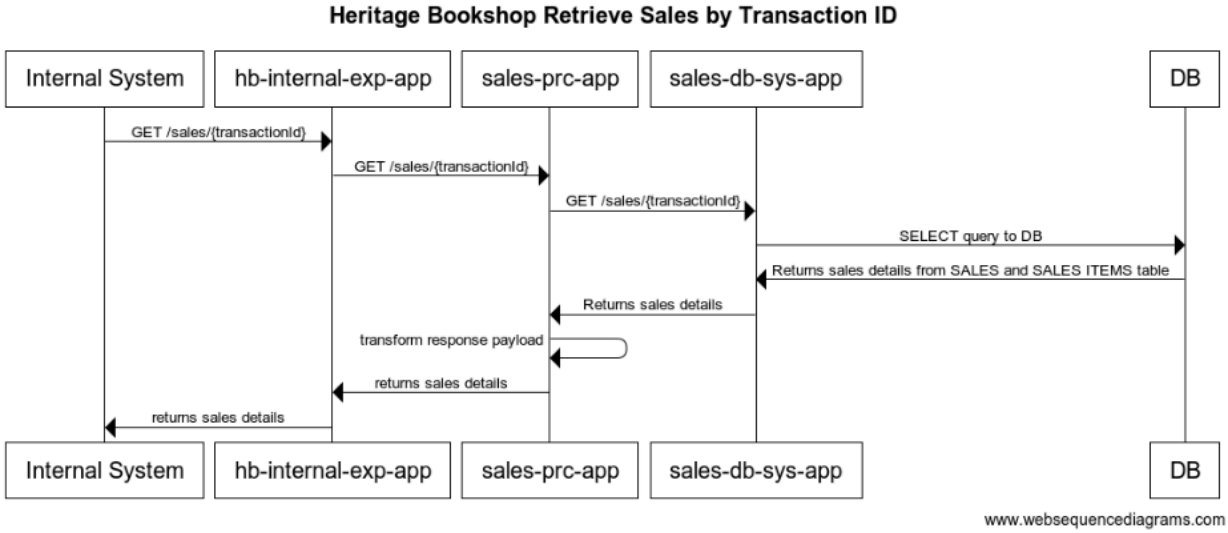
7.1 Create New Sales



Remarks: Due to time constraints, DELETE operation has been implemented as a short-term solution to prevent "orphan" sales and maintain database consistency.

In the current setup, the Sales record and Sales items record are inserted through separate operations. If the step on inserting sales items fails, the system would create partial or inconsistent data. Since we do not have yet failed-state handling mechanism, DELETE provides a simple and reliable way to clean up incomplete transactions and ensure the database remains consistent.

7.2 Retrieve Sales Details

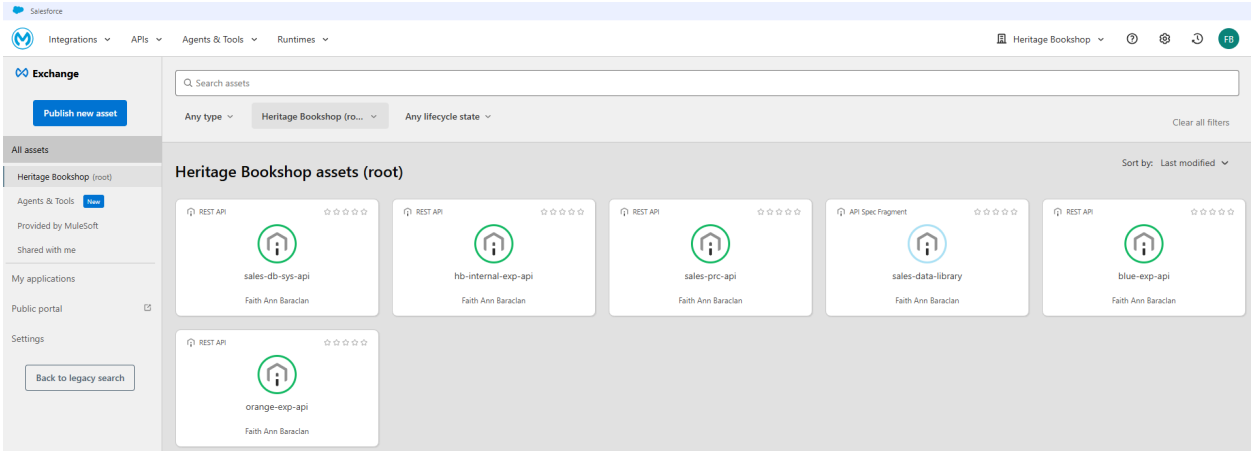


Remarks: This retrieves sales details from both Sales table and Sales Items table returning a single sales details from the database based on the transaction ID.

8. Anypoint Platform

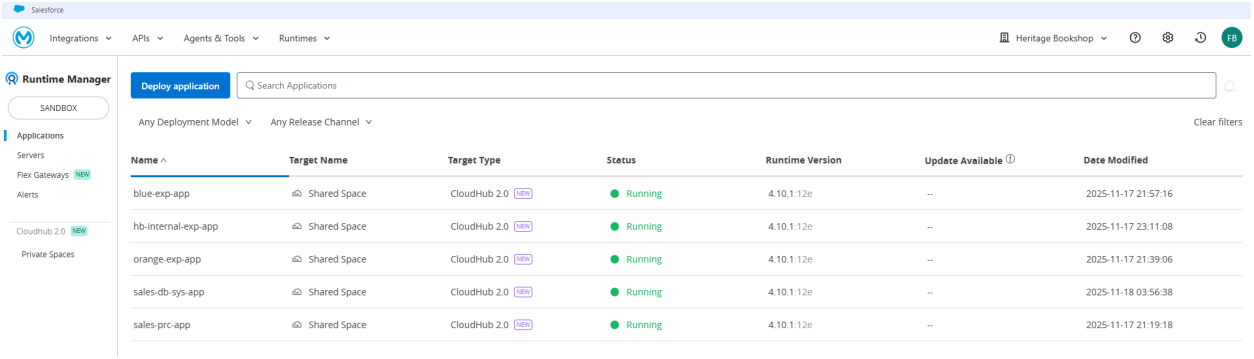
8.1 Anypoint Exchange

Below are the assets published to Anypoint Exchange.

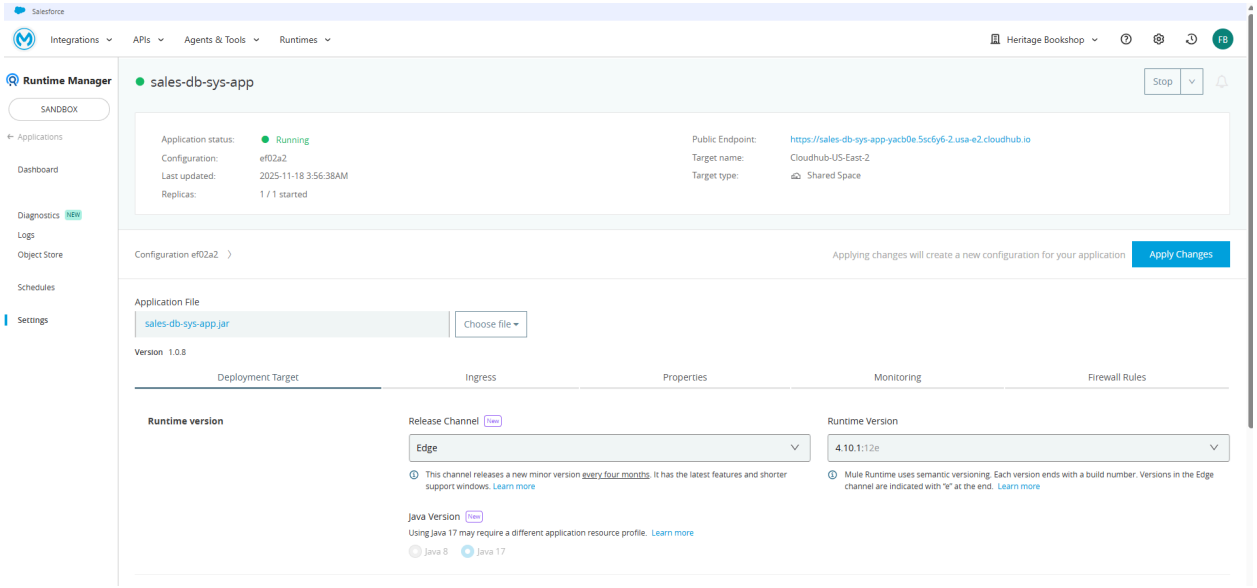


8.2 Runtime Manager

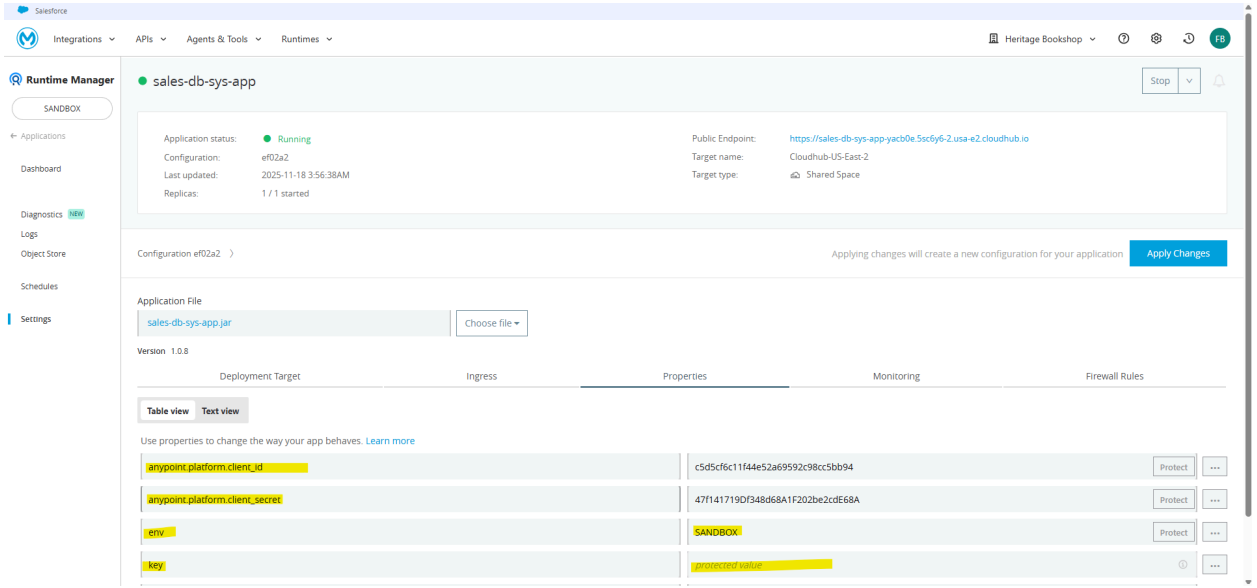
All Mulesoft APIs are deployed in Cloudhub 2.0 with Runtime version 4.10.



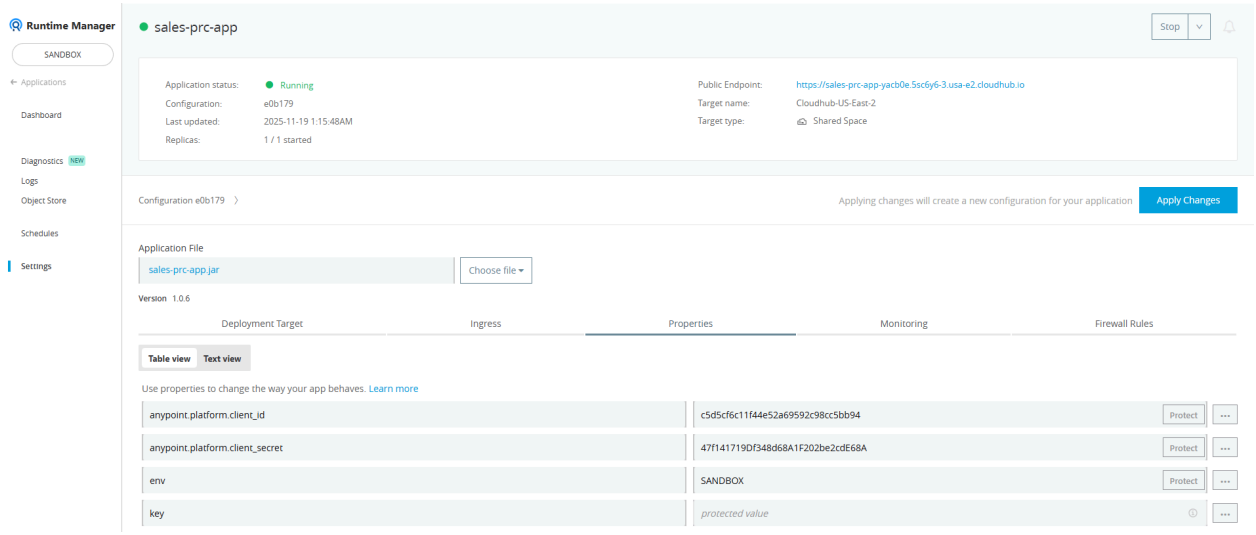
sales-db-sys-app - env: SANDBOX



Runtime properties

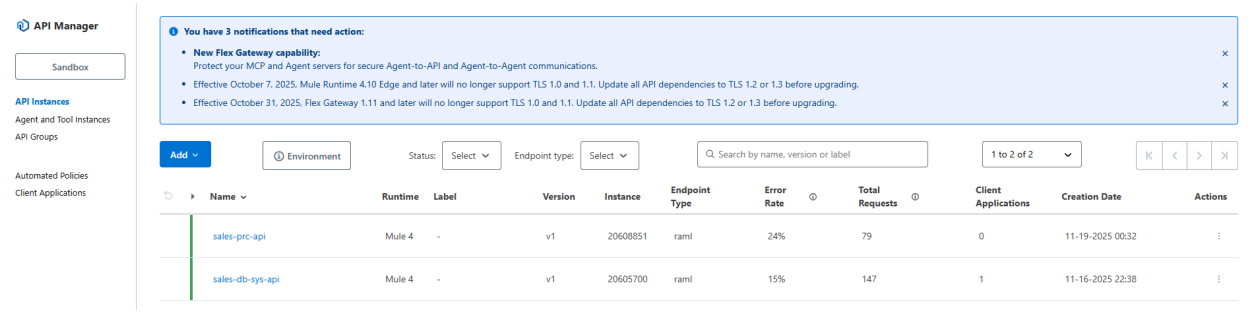


sales-prc-app - env: SANDBOX

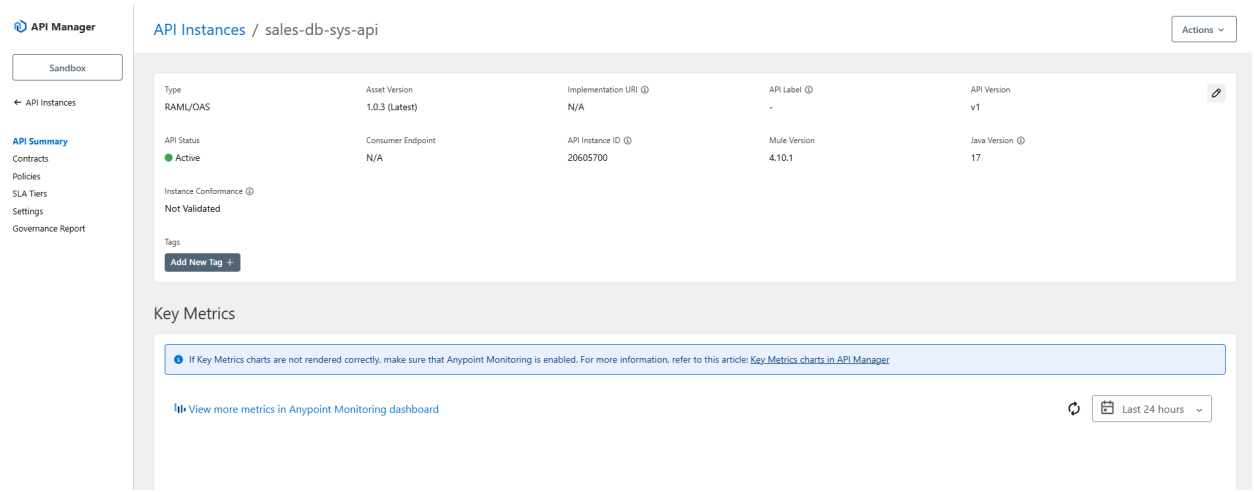


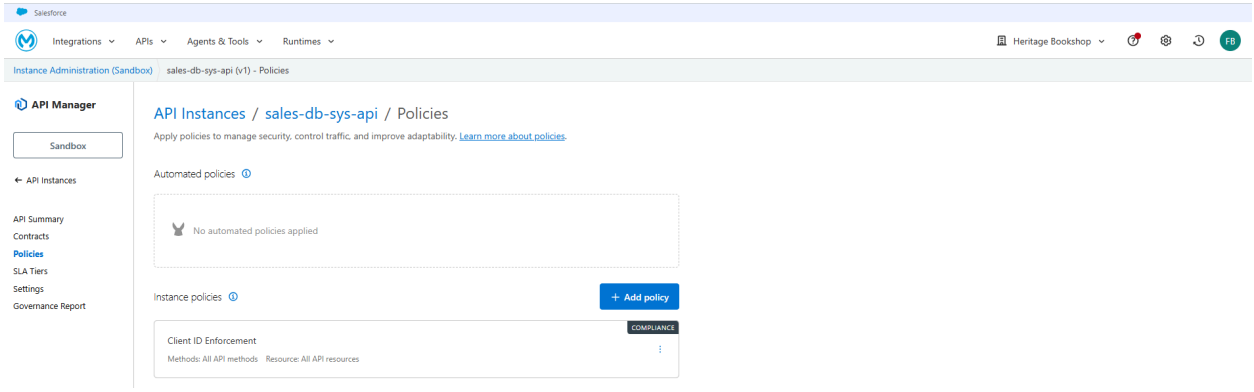
8.3 API Manager

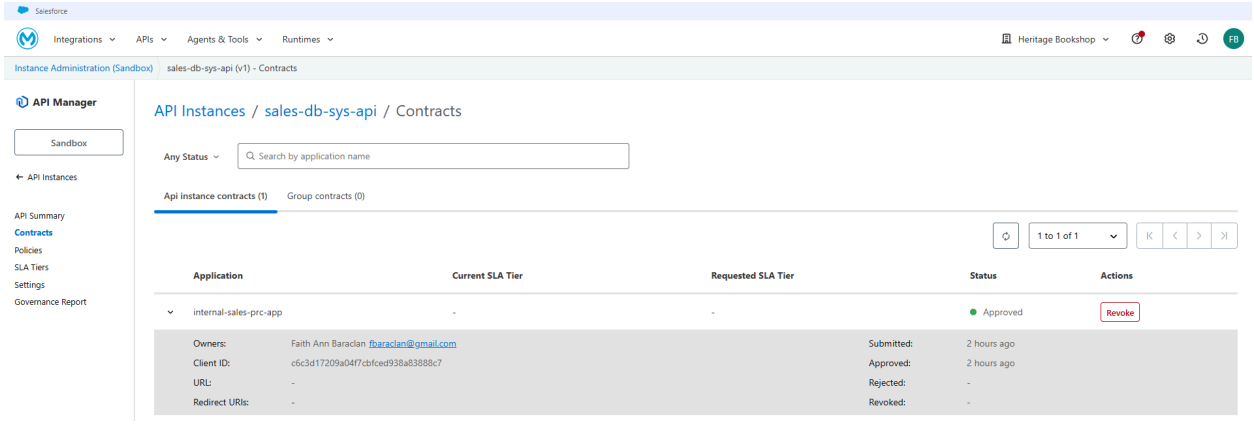
For demonstration purposes, **sales-db-sys-app** & **sales-prc-app** has API instances, and only **sales-db-sys-app** has policy applied, that is, Client ID Enforcement policy.



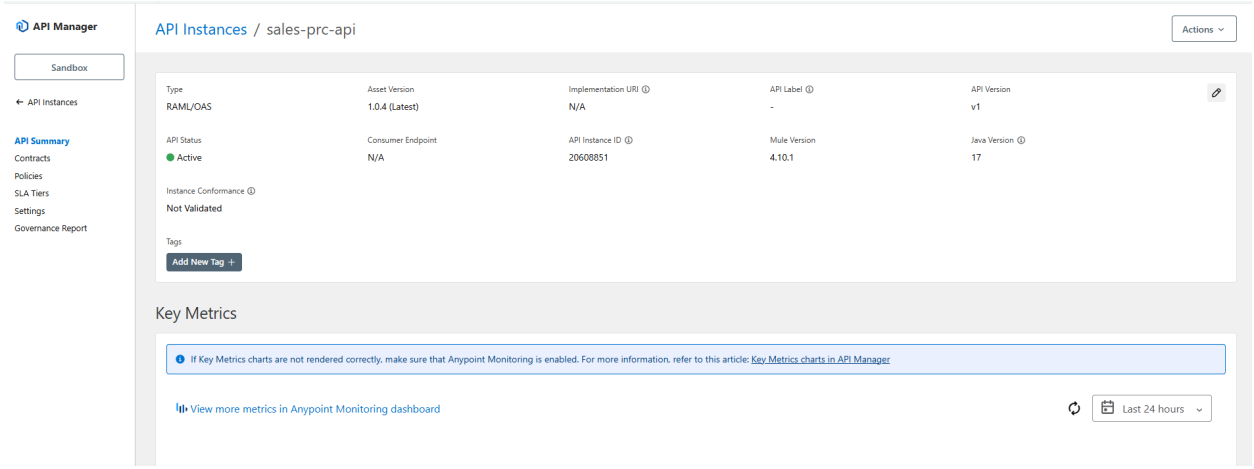
sales-db-sys-app - env: SANDBOX


API Manager
Sandbox
API Instances
API Summary
Contracts
Policies
SLA Tiers
Settings
Governance Report
API Instances / sales-db-sys-api
Actions
Type
RAML/OAS
Asset Version
1.0.3 (Latest)
Implementation URI
N/A
API Label
-
API Version
v1
API Status
Active
Consumer Endpoint
N/A
API Instance ID
20605700
Mule Version
4.10.1
Java Version
17
Instance Conformance
Not Validated
Tags
Add New Tag
Key Metrics
If Key Metrics charts are not rendered correctly, make sure that Anypoint Monitoring is enabled. For more information, refer to this article: Key Metrics charts in API Manager
View more metrics in Anypoint Monitoring dashboard
Last 24 hours


Salesforce
Integrations
APIs
Agents & Tools
Runtimes
Heritage Bookshop
FB
Instance Administration (Sandbox)
sales-db-sys-api (v1) - Policies
API Manager
Sandbox
API Instances
API Summary
Contracts
Policies
SLA Tiers
Settings
Governance Report
API Instances / sales-db-sys-api / Policies
Apply policies to manage security, control traffic, and improve adaptability. Learn more about policies.
Automated policies
No automated policies applied
Instance policies
Add policy
Client ID Enforcement
COMPLIANCE
Methods: All API methods
Resource: All API resources

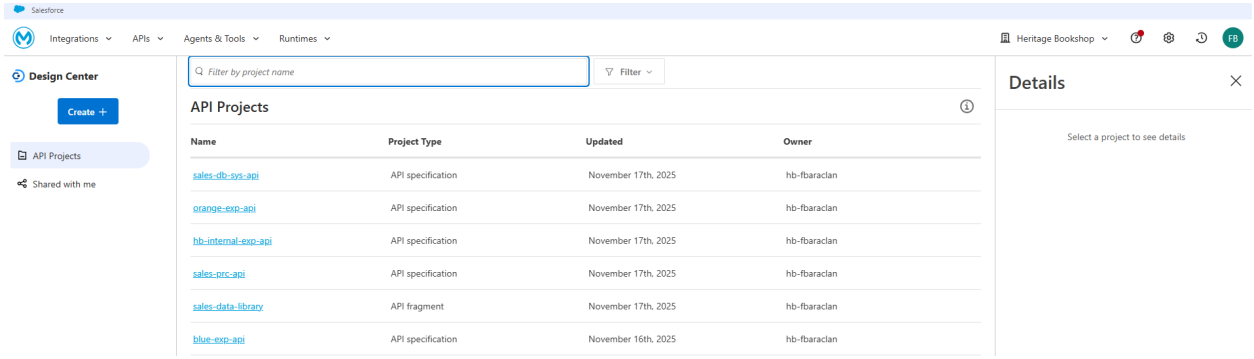

Salesforce
Integrations
APIs
Agents & Tools
Runtimes
Heritage Bookshop
FB
Instance Administration (Sandbox)
sales-db-sys-api (v1) - Contracts
API Manager
Sandbox
API Instances
API Summary
Contracts
Policies
SLA Tiers
Settings
Governance Report
API Instances / sales-db-sys-api / Contracts
Any Status
Search by application name
Api instance contracts (1)
Group contracts (0)
1 to 1 of 1
Application
Current SLA Tier
Requested SLA Tier
Status
Actions
internal-sales-prc-app
-
-
Approved
Revoke
Owners:
Faith Ann Baraclan fbaraclan@gmail.com
Client ID:
c6c3d17209a04f7cbfced938a83888c7
URL:
-
Redirect URIs:
-
Submitted:
2 hours ago
Approved:
2 hours ago
Rejected:
-
Revoked:
-

sales-prc-app - env: SANDBOX



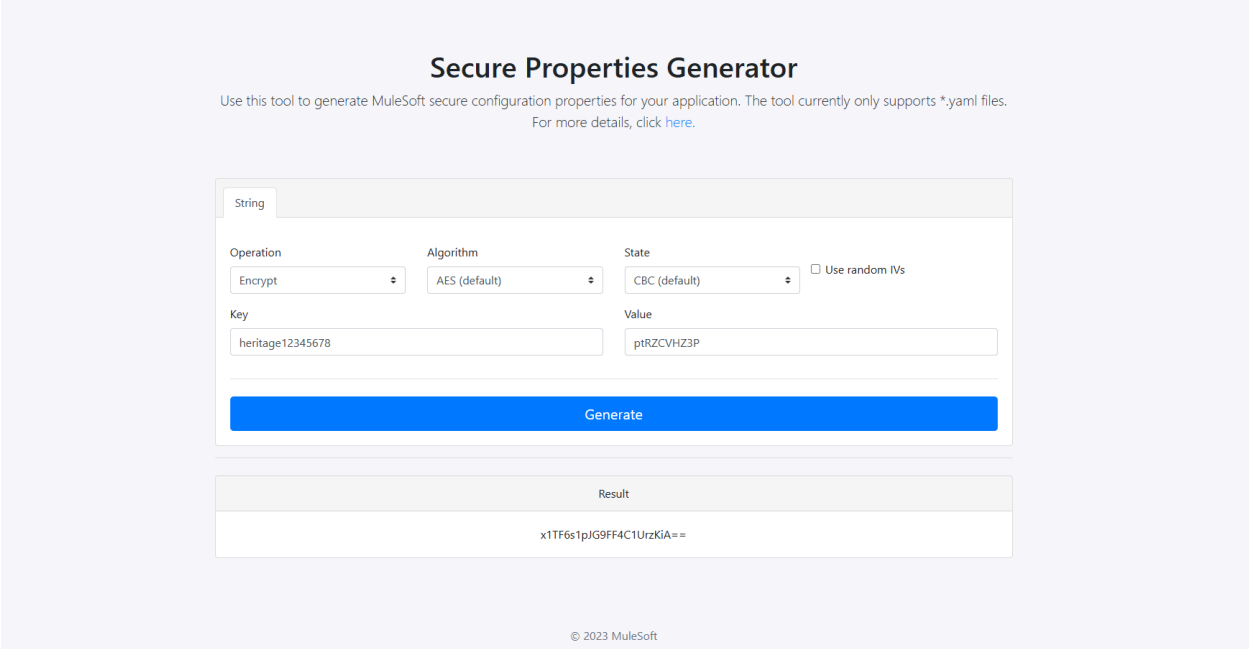
8.24 Design Center

API specifications are created in the Design Center, and then published to Anypoint Exchange.



9. Secure Properties Generator

Used secure properties generator to encrypt/decrypt passwords and client credentials.
Link: https://secure-properties-api.us-e1.cloudhub.io/

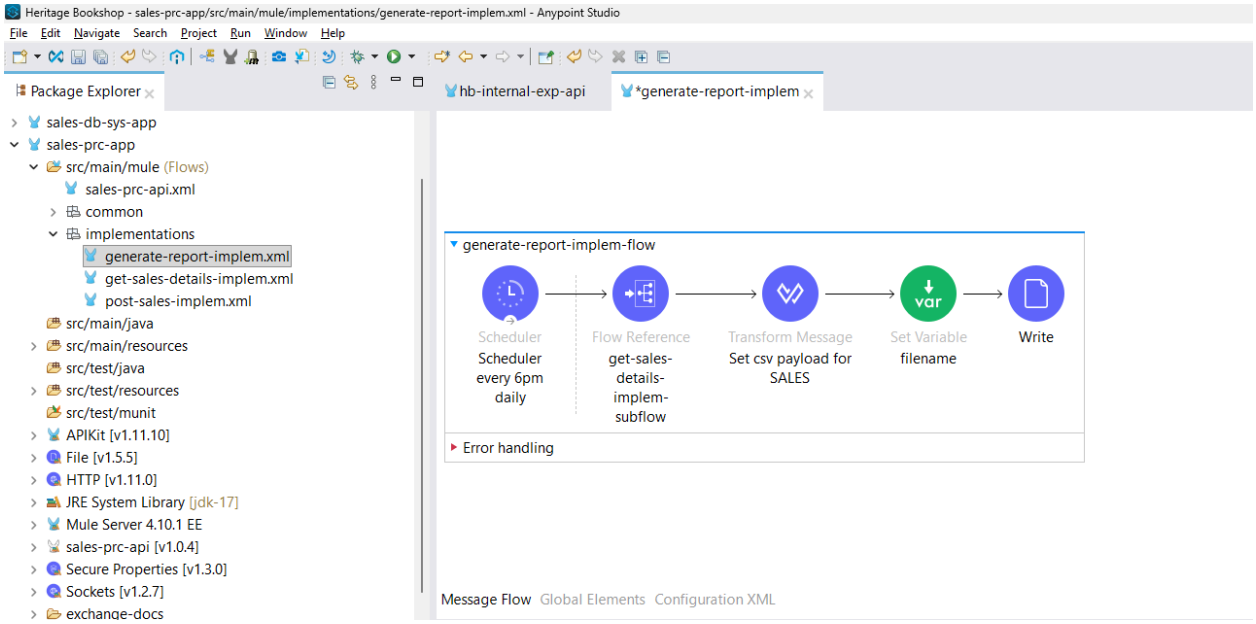


10. End-to-End Flow Summary

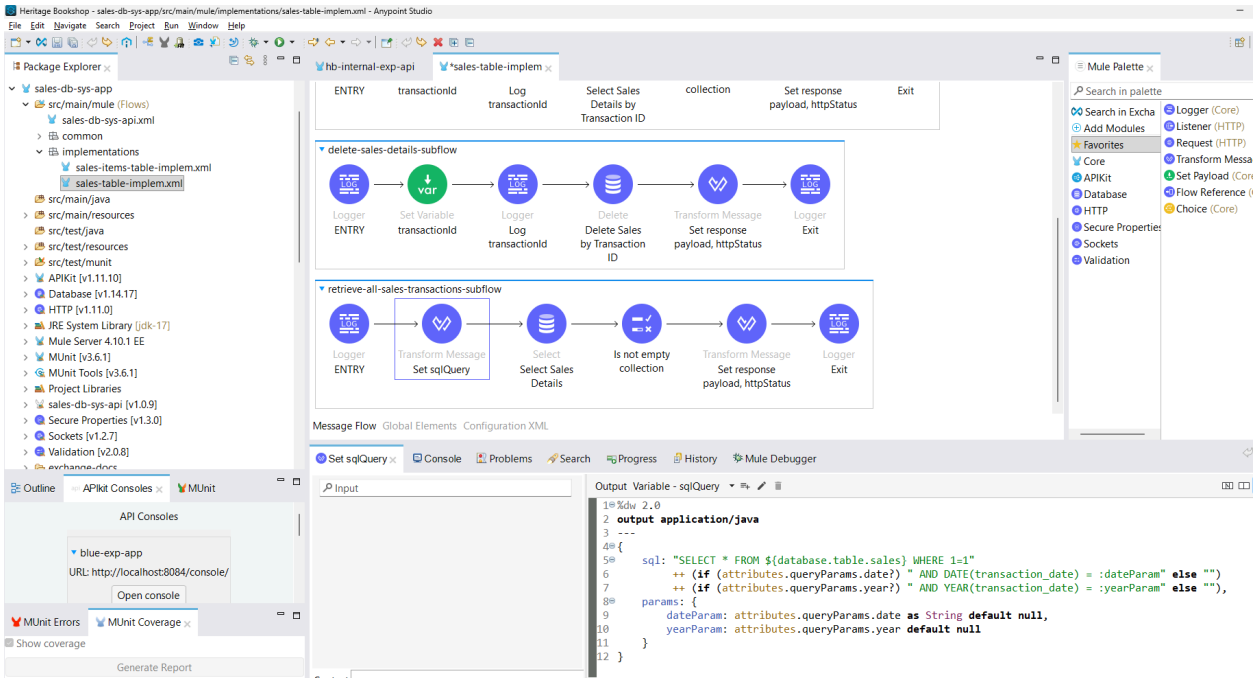
1. A client application (*Orange* app, *Blue* app, or internal system) submits a request to its dedicated **Experience API**.
2. The Experience API validates request format and calls the **sales-pro-app**.
3. The Process API handles business logic, applies transformations, and calls the **sales-db-sys-app**.
4. The System API interacts with the **Sales DB**, returns the result, and propagates it back up through the Process and Experience layers to the client.

11. Data Pipeline

A data pipeline is a series of processes that moves raw data to a destination. The most common or the one I'm familiar with is the ETL (Extract, Transform, Load) model. While I haven't fully deployed a complete solution yet, I have created a flow using Mulesoft that extracts data from the database and writes it as a CSV file to a designated directory.



This flow is designed to run on a scheduler, for example every day at 6PM, to generate daily sales reports from the **Sales** table in Sales Database.

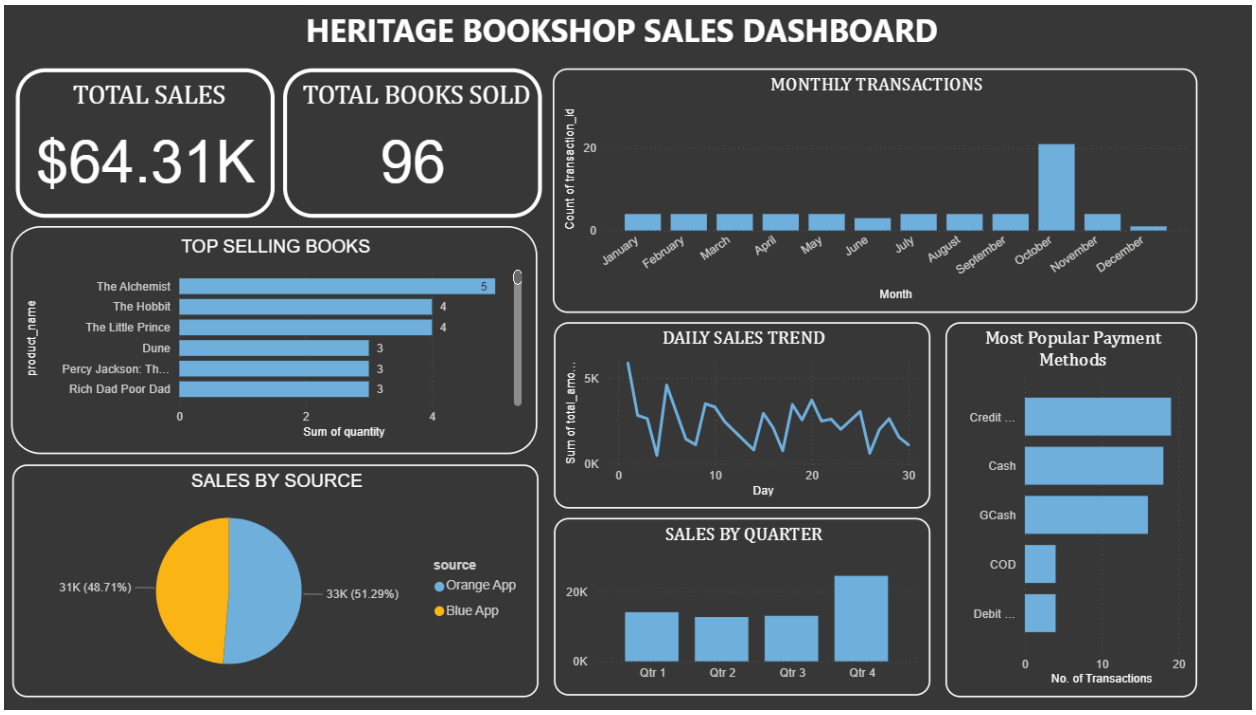


Additionally, I have implemented a partial flow in the sales-db-sys-app to retrieve sales transactions using query parameters such as **date** or **year**. For instance, calling the endpoint /api/sales?date=2025-11-02, it is expected to return all sales transactions for that specific date (or year).

Based on my research, there are other different tools available to build data pipelines. I may not be familiar with all of them, but I believe Mulesoft can fulfill the requirements of a data pipeline as per its standard definition.

Currently, no CSV files have been generated, as this is only a partial implementation.

12. Reports



Remarks: The data used for this dashboard comes from the extract of **Sales** table and **Sales items** table of the Sales database. The data was manually exported from the database server.

This dashboard has been created using Power BI. This is my first time working with Power BI, and I was able to use it to visualize sales data and generate insights for reporting. My background in Mathematics has helped me approach the data logically and create visualizations. Given more time to explore the tool, I hope to further enhance the dashboard and utilize other features.

13. Role of AI in the Solution

Artificial intelligence (or AI) played a supporting role in the development of this integration solution and its documentation. AI did not generate or implement the technical system itself;

rather it provided descriptive support to refine communication and structuring the documentation. Specifically, AI support was used in the following areas:

- Documentation Generation
    - assisted in creating documentation from scratch by providing documentation template with heading and subheadings
    - generated clear and professional descriptions
    - Standardized responsibilities of endpoints
- Testing
    - Generated sample payloads used for testing

Future Enhancements

If given the luxury of time, the following enhancements can be done but not limited to:

- Anypoint Exchange Documentation for all API specs created
- Proper logging on all the APIs, masking personally identifiable information (PII) - preferably with JSON logger
- Munit for ALL APIs
- Additional security for the APIs by applying additional policies