Anypoint Platform Architecture: Solution Design Exercises

Introduction

This document contains descriptions of the exercises that are part of the Solution Design class. More details on the training can be found on the MuleSoft Training website: https://training.mulesoft.com/instructor-led-training/aparch-solution-design

For these exercises, you will work on the digital transformation of a fictional bank, the ACME Bank. You will be the architect in charge of defining the architecture of the new digital platform for ACME Bank.

This platform will enable the bank to migrate their legacy systems and expand their business by enabling developers to quickly build new components and integrations.

Prerequisites & requirements

For these exercises you will be creating architectural blueprints and designs. You may use a simple text editor or word processing application to describe the application design and design decisions.

Also, consider using a drawing/diagramming tool to graphically design the application. There are freely available online drawing tools such as draw.io and lucidcharts or commercial diagramming tools such as Visio or Omnigraffle which provide stencils or templates for diagramming. If you do not wish to use a diagramming tool, you can use Webex's whiteboarding functionalities.

Exercise 1: Architectural blueprint

In this exercise you will draft an architectural blueprint of the digital transformation platform.

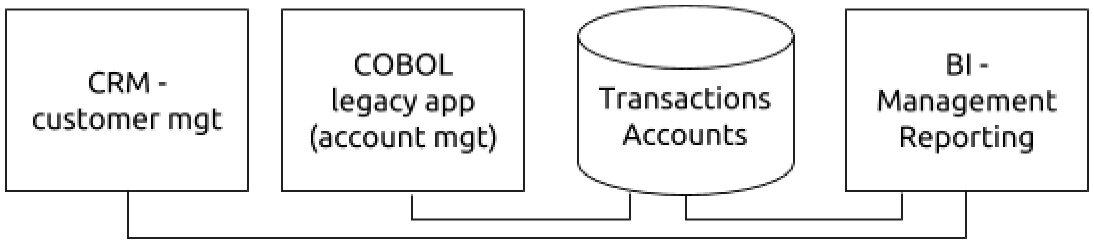
Background: ACME Bank's heterogeneous IT landscape consists of a variety of systems, applications and databases. The systems lack proper integration, though there is some point-to-point integration for specific purposes. It recently became clear that the current landscape does not scale and does not provide enough flexibility and capacity to expand. It also becomes increasingly difficult to maintain some old legacy systems.

The newly appointed CTO's main responsibility is to lead ACME Bank into the digital era by providing new and innovative applications and functionalities. The focus will be on quickly enabling web based and mobile applications, which are yet to be developed. Also, in due time several legacy systems are to be replaced by custom Java applications.

These systems are part of the first step in the transformation project:

- A custom developed set of COBOL mainframe applications that contain the main business logic for account management and transaction management. These applications are difficult to use in an integrated environment as they do not provide proper interfacing methods. Bank employees use this application directly using a terminal. ACME wishes to have better interfacing capabilities for these applications.
- An Oracle database that contains transactions and transaction related data and is being used by the COBOL application.
- A CRM system from a commercial vendor with customizations for customer management. Bank employees use the web based UI on top of this Java based CRM system for accessing and managing customer data.
- A commercial BI suite for providing management reporting, insight in transactions and customer data. The BI suite connects directly to the Oracle database and CRM system.

Partial overview of ACME's applications and systems:



Your task is to draft an architectural blueprint for the digital transformation platform. Make sure the following requirements are met:

- Integration of all mentioned systems.
- Legacy modernization of existing (and soon to be replaced) COBOL applications

- Exposure of main business functionalities through platform or technology independent interfaces (wherever possible).
- Offloading the old database so more requests for data can be served, knowing that the existing database has reached its performance limitations.

Recommendations:

- Make sure your initial blueprint is future proof, thus enabling the rapid development of new applications and services on top of the platform.
- Clearly identify an architectural style (traditional SOA, microservices, API-led, custom)
- Use the MuleSoft approach and recommended architectural concepts in your design.
- Try to use a common and commonly understandable diagramming style.
- You can use a drawing/diagramming tool to graphically design the application and optionally use a text editor or word processor to describe the application design.

Exercise 2: API Design

The goal of this exercise is to design a set of APIs that will allow developers to easily build applications for both customers and bank employees.

Part 1: API basics
Your task is to design an API based on a set of functional requirements, recorded as user stories. Break into groups and start whiteboarding the basics of the API. Focus only on identifying resources and operations.

You may use any notation you like, but refrain from using an actual API definition language or tools.

Requirements for a bank employee's desktop application, written down as user stories:

As a bank employee I want to:

- Update bank account information
- Replace a customer's information
- Retrieve details on a particular incoming transfer
- Retrieve a list of all transfers, both incoming and outgoing.
- Retrieve a full list of customers
- Add an account to a customer
- Remove an account from a customer
- Retrieve a list of outgoing transfers
- Create an outgoing transfer from a customer's account to another account
- Add a customer
- Retrieve a detailed account report
- Get a list of accounts belonging to a particular customer
- Retrieve a list of incoming transfers for a particular bank account
- Retrieve details on a particular outgoing transfer

Step 1: Identify requirements for your API
Ask yourselves questions about what kinds of capabilities should be available through your API.

Step 2: Translate requirements
Now that you've identified what you want your API to do, translate that functionality into resources and methods.

Remember: Resources = nouns, methods = HTTP verbs. Remain at this high-level.

Part 2: API definition
Based on the identified resources and operations, start working on a more elaborate API design. You may start using a formal API modeling language such as RAML (recommended but not limited to).

Create an actual API definition in the Anypoint API Designer, API Workbench or any other text editor of choice. Use the identified resources and operations as a starting point and add the following elements:

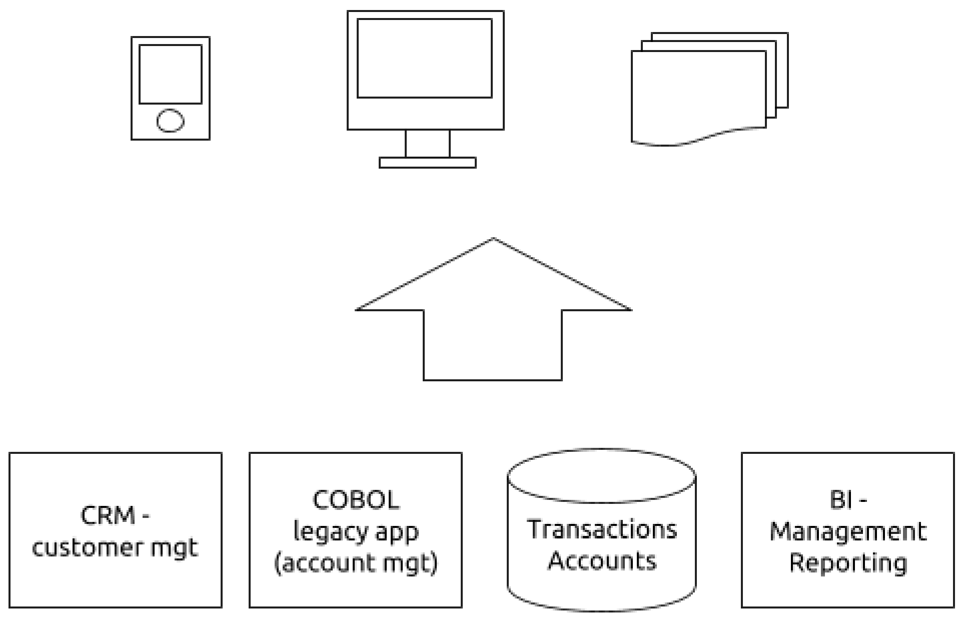
- Documentation/API descriptions
- Mediatypes
- HTTP response types
- Sample responses

Part 3: API definition refinement
The last part of this exercise consists of refactoring the API design you created in the previous steps into a full-fledged RAML API:

- Identify possible traits (reusable behavioural elements). Think of applying paging to large sets of data or searching options. Try to refactor code duplicates into reusable elements.
- Identify data types and data models. Assume all data will be in JSON format. Provide examples in JSON and define resourcetypes.
- Review the structure and hierarchy of your API. Apply design recommendations and best practices.

Exercise 3: Application Design

As part of the digital transformation journey, ACME Bank wants to support the Open Bank initiative and forms a task group for this purpose. The task group controls a development team that has been assembled to implement all necessary infrastructure and application components.



Goals
Your task is to design -in detail- the integration platform components that will allow access to the bank's data coming from various sources.

The data must be made available in a uniform, easy to consume format.

Deliverables
Write an architectural design document that contains all components and details you consider relevant. Feel free to use any documenting/diagramming style or template you prefer.

For your reference, check slides 5-11 of module 5. Those slides can help you identifying the most important components of your architectural design.

Note:
Your design must be detailed enough for a developer to be able to implement it independently. However, the main focus of this exercise is on solution design, and your design decisions and their rationales.

Recommendations
Make sure the platform is capable of serving multiple different clients, such as mobile apps, web apps, reporting systems, custom clients etc. Keep in consideration that the Open Bank program assumes the availability of APIs.

Design the platform as an enterprise grade solution. Specify in detail your architectural approach and implementation components. Make sure to pay attention to :

- Interface specifications (how can the clients consume data from the bank's systems?)
- Data formats and protocols (how is the data presented?)
- Application layers and components (what are the implementation details?)
- Data transformation (what is needed to transform from specific to generic format?)

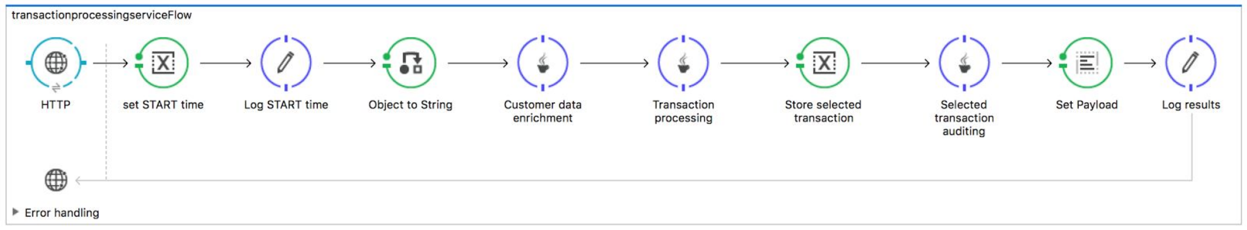
Exercise 4: Performance optimizations

The ACME bank's legacy COBOL application processes a set of transactions each night. The set of transactions is extracted from the transactions database and exported as a single large file, which is transferred to a shared file system. The COBOL application loads the file and reads the contents. Every transaction is processed sequentially. As part of the reporting requirements, the largest transaction amount must logged and investigated more closely after processing all transactions.

Your task is to design a Mule application that will replace the legacy processing application. One of your developers has already built a prototype application that mimics the business logic of the old application, but it does not perform and scale very well and does not meet the non-functional requirements.

Part 1
Design a new version of the application that is optimized for performance. Think of the most efficient way of maximizing the performance of the application. As a reference, use the design recommendations and best practices of Module 8.

The instructor will demonstrate the prototype application and show the code. The application's main flow is depicted below:



If you have Anypoint Studio installed, you can import the prototype application, which can be found in the student files.

Deliverables:

- Application design of the transaction processing application that is optimized for performance. You may use a simple text editor to describe the application design, including all relevant Mule flows or use a drawing/diagramming tool to graphically design the application. Alternatively, you may use Anypoint Studio to actually build out the application. Note that Anypoint Studio is not a requirement for this course, so this is optional.
- The design must be detailed enough for a developer to be able to implement it independently.
- Provide rationales for your design decisions that will improve performance. Describe how your design will improve the performance.

- Describe endpoints and relevant Mule flow components.

Please note that this is not a development class, and you will not be graded on delivering working code. You may use Anypoint Studio, though this is not required. The main focus is on application design, and the rationales of the design decisions you made for optimizing the application's performance.

Part 2
Improve your existing design and optimize it for reliability and scalability. The transaction processing service is a crucial component for ACME, and needs to be available 24x7.

ACME Bank expects to grow steadily in the next two years in terms of new customers, but also with regard to services, transactions and new products. The newly appointed CTO defined to following non-functional requirements:

- The transaction processing service must be available 24x7.
- The transaction processing service must be highly reliably. In case of a disaster, no data should be lost.

Your task is to describe the changes that need to be made to the current architecture in order to support high availability, but also scalability.

- Clearly describe your recommended approach to scalability and how to make sure the applications and infrastructure will be ready for it.
- Clearly describe the application architecture and infrastructure architecture.
- Take into consideration that the annual license renewal means that a new license file needs to be installed. Installing a new license file usually requires the Mule runtime to be restarted to activate the new license. Come up with a solution to address this, as it conflicts with the requirement of 24x7 availability.

Part 3
Take the solution application from the student files as a reference.
Your task is to evaluate the application architecture and define a set of development standards and best practices for your development team.

Also, the newly hired CTO appointed a Chief Security Officer who asked you to come up with a security plan. Your second task for the exercise is to define security standards that will be part of the development standards. Check the best practices provided in the various modules of this training for reference.

Recommendations:

- How can the application be secured? There is sensitive data, unsecured communication etc.

- What code elements/components are duplicates. How can this be refactored?
- What elements/part of the code can be made configurable?
- Testability
- Coding standards
- Transformations?
- Data formats, usage of canonical data format?