Na análise da solicitação identifiquei que serão necessários cinco endpoints, são eles:

* Cadastrar cliente
* Listar cliente
* Editar cliente
* Atualizar cliente
* Deletar cliente

Normalmente os comandos usados seriam CADASTRAR = POST, LISTAR = GET, EDITAR = PUT / PATCH, ATUALIZAR = PUT / PATCH e DELETAR = DELETE.

Por experiência decidi usar o PUT para o cadastro de cliente ao invés do POST, o motivo é que o comando POST permite duplicidade de registros, se encontrar um registro igual ele irá criar um novo registro com os mesmos dados, PUT se existir um registro igual ele irá atualizar e se não existir ele cria um novo.

Aparentemente EDITAR e ATUALIZAR parecem o mesmo processo, mas entendi que EDITAR seria um a um, ATUALIZAR por ser lista o processo seria por lotes.

EDITAR decidi usar o PATCH, o comando PATCH atualiza o registro se existir e retorna EXCEPTION se não existir, assim permitindo ao sistema solicitante enviar uma mensagem de erro para usuário.

ATUALIZAR por considerar que seria feito em lotes decidi usar o PUT, o comando PUT atualiza o registro se existir e cria um novo se não existir, permitindo assim que o usuário até incluir um novo registro na atualização se o mesmo desejar.

Os demais foram usados os comandos padrões LISTAR = GET e DELETAR = DELETE.

No desenvolvimento usei o padrão recomendado como melhores práticas sendo:

* Desing no Anypoint Plataform documentando assim todo o processo.
* Nomenclatura foi usada a mais indicada, baseada em API Led, como o desenvolvimento e focado na camada SYSTEM foi usado SAPI, ficando assim builders-sapi-customer.

Builders = Cliente

SAPI = System API

Customer = Objetivo do desenvolvimento

* Importação do desing no Anypoint Studio.
* Cada ENDPOINT foi criado um Mule Configuration File(XML) de implementação, assim facilitando futuras manutenções e entendimento do desenvolvedor responsável.
* Um Mule Configuration File(XML) como GLOBAL, para serem colocados os objetos compartilhados.
* Um Mule Configuration FIle(XML) como Interface, onde está o flow principal facilitando o entendimento geral do projeto.
* Banco de dados usei o Apache Derby Database usando o armazenamento em memória temporária.
* Configurar o Apache Derby Database eu usei o Spring para usar as bibliotecas e dependências necessárias.
* Finalizar o desenvolvimento no Anypoint Studio conforme a solicitação.
* Efetuar teste usando o POSTMAN.
* Usar o DEBUG MODE para analisar os problemas de desenvolvimento.
* Criar o collection e exportar no POSTMAN.
* Exportar o projeto no Anypoint Studio.
* Criar URL de um repositório GIT na nuvem.
* Entregar projeto com relatório.