

ENTERPRISE SERVICE BUS

ESB

Basicamente é um EAR do JEE tendo o JMS (Java Message Service) = **Middleware**

Através da API JMS duas ou mais aplicações podem se comunicar por mensagens.

JMS suporta dois modelos de troca de mensagens:

- **ponto a ponto ou modelo de filas:** um "produtor" (producer) envia mensagens para uma fila e um "consumidor" (consumer) as lê.
- **modelo publish/subscribe:** O(s) "assinante(s)" (subscriber) podem registrar interesse em receber ("em assinar") mensagens de um tópico. Neste modelo, nem o "publicador" (publisher) ou o "assinante" sabem um do outro.

<http://pt.wikipedia.org/wiki/JMS>

Normalmente baseado no reconhecimento de padrões, que fornecem uma base de serviços para arquiteturas mais complexas via um driver de evento e padrões baseados em mensagens (BUS). ESB devem ser baseados em padrões flexíveis, suportando vários meios de [transportes](#).

Arquitetura ESB

A palavra "bus" é a referência para o meio físico que carrega bits entre dispositivos em um computador. O ESB serve a uma função análoga a alto nível de abstração. Em uma arquitetura empresarial fazendo uso de um ESB, uma aplicação irá comunicar via barramento, que atua como um message broker entre aplicações. A principal vantagem de com uma aproximação é a redução de conexões ponto a ponto necessárias para permitir a comunicação entre aplicações. Isto por sua vez afeta diretamente na simplificação das mudanças de sistema. Por reduzir o número de conexões ponto a ponto para uma aplicação específica, o processo de adaptar um sistema às mudanças em um de seus componentes torna-se mais fácil.

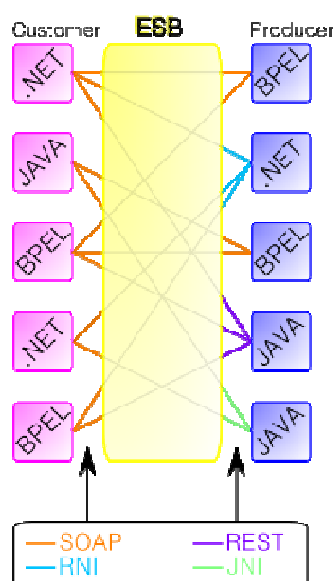


Figura 1 - ESB

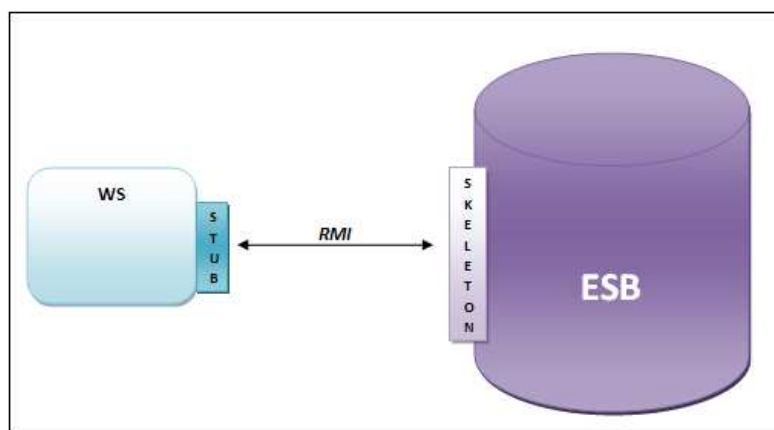
























Figura 2 - Sutb > Skeleton


Action	Descrição
 Dynamic Publish	Publicar uma mensagem para um serviço identificado por uma expressão XQuery
 Publish	Colocar uma mensagem para um serviço de informação estaticamente.
 Publish Table	Colocar uma mensagem para zero ou mais serviços especificados estaticamente. Lógica da condição de estilo interruptor é usada para determinar em tempo de execução quais serviços serão utilizados para a publicação.
 Routing Options	Modificar qualquer uma ou todas as seguintes propriedades do pedido de saída: URI, qualidade de serviço, de modo, Retry parâmetros, prioridade da mensagem.
 Service Callout	Configurar um síncrona (bloqueio) chamada para um serviço de proxy Bus-registrada da Oracle ou serviço do negócio.




Action	Descrição
 Transport Headers	Defina os valores de cabeçalho de transporte em mensagens
 Dynamic Routing	Atribuir uma rota para uma mensagem com base em informações de roteamento disponível em um recurso XQuery.
 Routing	Identificar um serviço de destino para a mensagem e configurar a forma como a mensagem é encaminhada para o serviço:
 Routing Table	Atribuir um conjunto de rotas envolto em uma condição de estilo interruptor rotas table.Different são selecionados com base nos resultados de uma única expressão XQuery.

Action	Descrição
 For each	Iterar sobre uma sequência de valores e executar um bloco de ações
 If ... then ...	Executar uma ação ou conjunto de ações condicional, com base no resultado booleano de uma expressão XQuery.
 Raise error	Gerar uma exceção com um código de erro especificado (a string) e descrição.

Action	Descrição
 Reply	Especificar que uma resposta imediata ser enviado para o solicitante.
 Resume	Retomar o fluxo de mensagens após um erro é tratado por um manipulador de erro.
 Skip	Especificar que em tempo de execução, a execução do estágio atual é ignorado eo processamento prossegue para a próxima fase no fluxo de mensagens.

Action	Descrição
 Assign	Atribuir o resultado de uma expressão XQuery para uma variável de contexto.
 Delete	Excluir uma variável de contexto ou um conjunto de nós especificados por uma expressão XPath.
 Insert	Insira o resultado de uma expressão XQuery em um local identificado em relação a nós selecionados por uma expressão XPath.
 Java callout	Invocar um método Java a partir do pipeline.
 MFL transform	Converter os não-XML para XML ou XML para não-XML no pipeline.
 Rename	Mudar elementos seleccionados por uma expressão XPath sem modificar o conteúdo do elemento.
	Substituir um nó ou o conteúdo de um nó especificado por uma expressão XPath.

Action	Descrição
Replace	
 Validate	Validar os elementos selecionados por uma expressão XPath contra um elemento do esquema XML ou um recurso WSDL.

Action	Descrição
 Alert	Enviar uma notificação de alerta com base no contexto mensagem pipeline.
 Log	Construir uma mensagem a ser registrada.
 Report	Ativar relatório de mensagem para um serviço de proxy.

PipelinePar => qd usa? vc precisa manipular fazer varias chamadas.

Exemplo: Chama Serasa/ Chama Casas Bahia => precisa manipular os retornos no response.

Route => qd vc? Não precisa quebrar as coisas.

Exemplo: Um unico sistema que resolve seus problemas, vc tem um request verifica credito.. e ele ja responde o que vc precisa. Ele é o end de um processo que vc nao precisa manipular as respostas. Ele encerra o processo.

Service Callout (Síncrono): Expõe Req/Reply chamadas para BU.

Publish (assíncrono): uma fila de messageira, ele não encerra o processo.

Proxy: Expõe o endpoint, expondo a api.

Business: Responsável por comunicador com o legado, faz a chamada dos legados, cuida do load balance sendo exposto por um PROXY. Ele que é o Stub dos WS.

Quando é criado ele encapsula as chamadas para os legados (cobol/.net/Java/etc..).

Na imagem abaixo, vemos que um Proxy contém um determinado fluxo, fazendo chamadas a WS/BU:

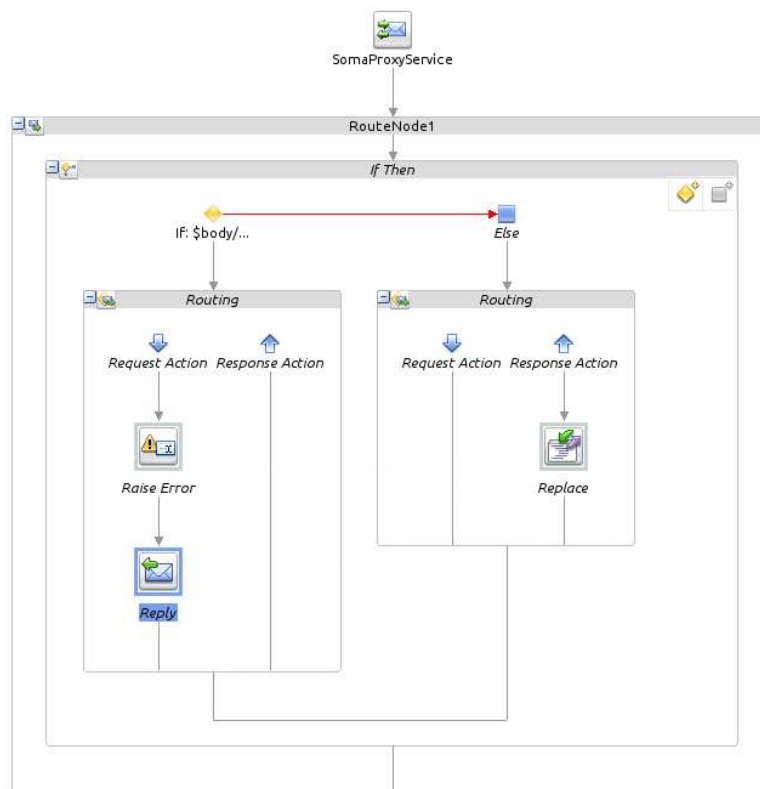


Figura 3 - Exemplo DES

MULE

<http://www.mulesoft.org/>

Mule ESB é um leve middleware de código aberto que fornece integração de aplicações abrangente. O Enterprise Service Bus (ESB) no núcleo Mule facilita conexões de intranet dentro de uma organização, bem como as ligações externas seguras com APIs baseadas em Web e outros recursos em nuvem.

Mule permite a fácil integração de sistemas existentes, independentemente das diferentes tecnologias que o uso aplicações, incluindo JMS, Web Services, JDBC, HTTP, e muito mais.

A idéia do Mule é usar tudo na NUVEM, por isso ele conta com a CLOUD HUB, onde a ideia é você desenvolveu > disponibiliza sua app na NUVEM.

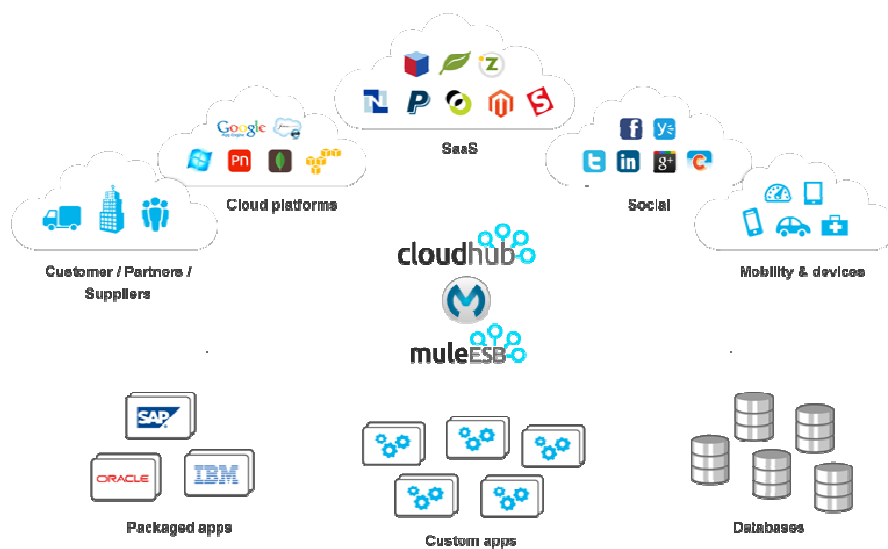


Figura 4 - MuleESB to CLOUD

[<http://www.mulesoft.org/documentation/display/current/Meet+Mule>]

Ele contém muitas topologias, conseguindo fazer também um Peer-toPeer Network, o que o OSB e o CAMEL não fazem o mesmo. Pois ele baseia-se em ORQUESTRAÇÃO. Baseado em EVENTO, arquitetura SEDA, sendo mais escalável, atendendo mais de 100 mil requisições, porém isso não quer dizer que ele seja mais rápido, quer dizer que ele consegue se preparar para isso.

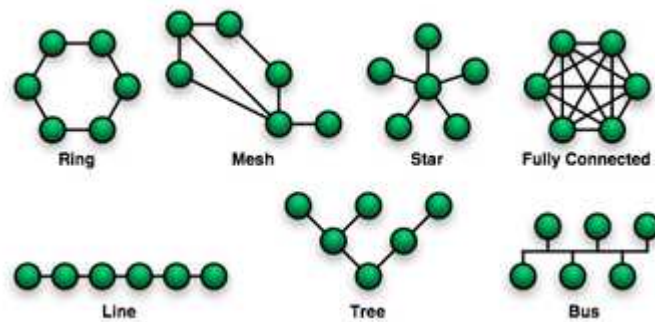


Figura 5 - topologia

Principais vantagens do Mule:

- Possibilidade de validar a sua arquitetura antes de partir para um investimento em uma solução comercial (se for o caso).
- The simplicity of installation and deployment of Mule. Simplicidade de instalação e implantação.
- The availability of the broadest scope of supported transports, and Inversion-of-Control containers. Disponibilidade da mais ampla margem de transportes apoiados, e Inversão de Controle contentores.
- The availability of active community support, as well as commercial support. Disponibilidade de apoio ativo da comunidade, bem como o apoio comercial.

Principais desvantagens do Mule:

- Falta de integração com os produtos de outros fabricantes
- Complexidade para implementar "serviços complementares" necessários para administrar um ESB:
 - a) service monitoring
 - b) administration
 - c) contract management
 - d) security policies

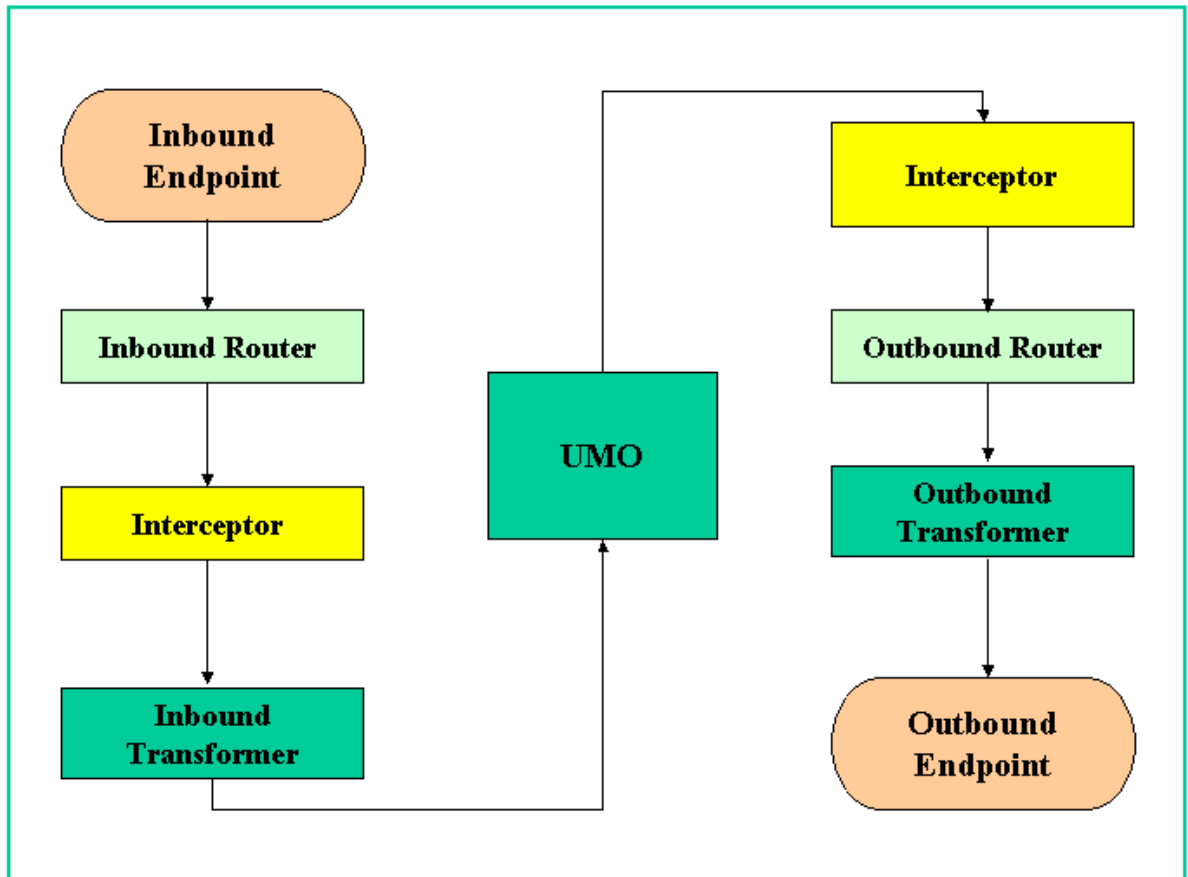


Figura 6 - Exemplo de funcionamento

UMO (Universal Messaging Object)

Event-driven software cliente. Componente que recebe e processa uma mensagem.

Inbound transformer

Componente que fornece mensagem de transformações a partir do transporte do cliente em formato imediatamente anteriores à invocação método UMO.

Outbound transformer

Componente que fornece mensagem de transformações do cliente em formato de transportes. Fica entre o emissor de mensagem e o encaminhador da mensagem.

Endpoint

É uma configuração do objeto que define a forma como os dados serão recebido e transformado pelo Mule.

Inbound router

Componente que controla o fluxo em mensagem-endpoint a direção-UMO.

Outbound router

Componente que controla o fluxo na mensagem UMO-to-end point direção; pode incluir vários tipos de filtros para testar se a mensagem pode ser enviada para um determinado parâmetro.

Interceptor

Oferece serviços adicionais, como a exploração da mensagem, coletando dados estatísticos, etc.

Um grande diferencial: Por exemplo, se você faz a centralização com um MuleESB , exemplo.: Você dentro da sua empresa consegue ver qual área usa mais dados, Area Adm/ RH, no mês a área ADM usa determinados serviços mais que o RH portanto, ela deve pagar mais por consumir os dados do que o RH, visando lucro da área de TI aumentando seu Bandit.

Possui dois Tipos

1. Baseado a AOP (***programação orientada a aspectos*** ou ***POA***, é um paradigma de programação de computadores que permite aos desenvolvedores de software separar e organizar o código de acordo com a sua importância para a aplicação (separation of concerns). Todo o programa escrito no paradigma orientado a objetos possui código que é alheio a implementação do comportamento do objeto.) Um interceptador poderia oferecer grande flexibilidade no momento de autorizar o acesso aos seus serviços, inclusive podendo consultar o BD para isso.
2. Baseado por direção, encadeando os filtros, Change Of responsibility.

Conector

Prevê a implementação do conector de ligação ao sistema exterior.

IMPORTANTE

É necessário sempre TRANSFORMAR os dados, por exemplo, entro STRING e vai sair XML, ele possui muitos tipos prontos.



Figura 7 - Conversão Tipos

Ele suporta várias linguagens como PYthon/PHP etc.. não é obrigado a usar somente XQuery e XPath. => lembrando que isso dificulta a integração já que você pode usar muitos tipos.

Dúvida?

Vale a pena pagar o MULE!?

O Mule Pago oferece suporte, contem um Manager (um console) para você controlar suas app, e ainda podendo fazer diretamente deploy na Nuvem ☺

Tabela 1 – Comparativo Preço

MULE PAGO	OSB
50 mil Dólares	1 Milhão Dólares

p.s. preço não são reais, foi somente um comparativo!

IDE de DESENVOLVIMENTO

<http://www.mulesoft.org/documentation/display/current/Mule+Studio>

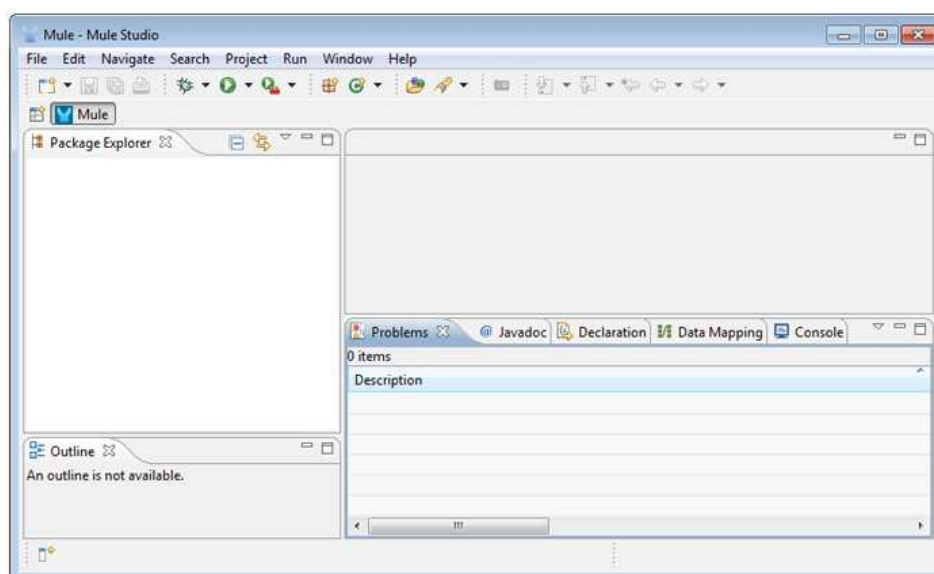


Figura 8 - Mule Studio

Apache Camel

<http://camel.apache.org>

O Apache Camel é um framework de integração baseado em padrões empresariais de integração (EIP). Traduzindo em termos simples, ele pretende simplificar a integração entre sistemas. Com centenas de componentes disponíveis (<http://camel.apache.org/components>), é possível criar fluxos entre aplicativos de maneira prática e rápida.

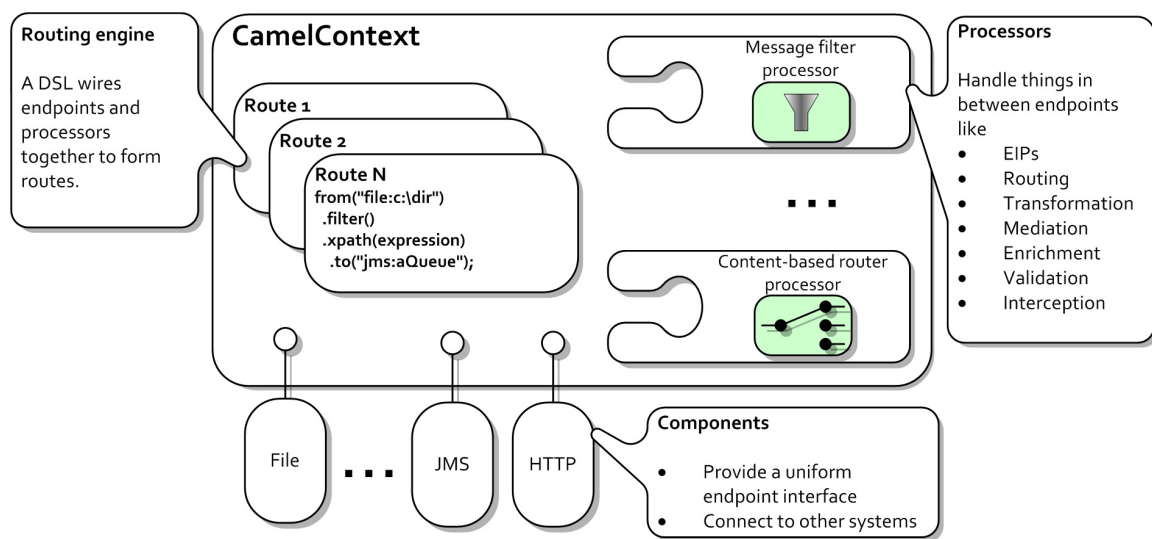


Figura 9 - Apache Camel

Como funciona?

Ele é diferente do MULE e do OSB, não contém um container, portanto não é indicado para usar quando dentro da empresa você precisa se comunicar com muitas aplicações, pois ele se carrega para dentro das aplicações, fazendo uma topologia Peer to Peer, sendo recomendado para quando sua aplicação precisa de um Payload/FaceBook/ EJB/Email etc..:

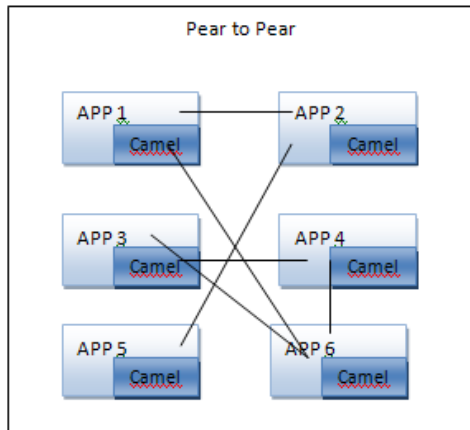


Figura 10 – Inviável

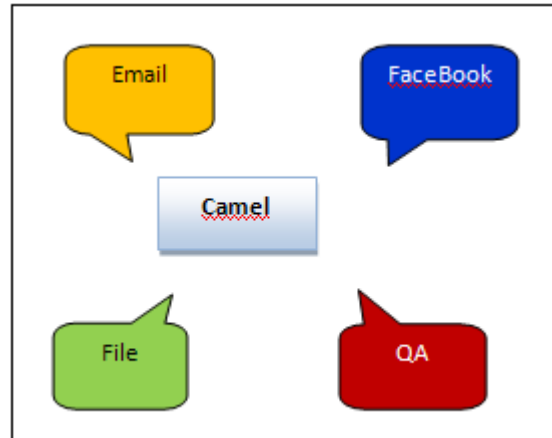


Figura 11 - Viável

IDE

Eclipse/Neatbeans, usando maven basta adicionar na dependência:

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-core</artifactId>
  <version>2.11.0</version>
</dependency>
```

Faz DE => PARA

FROM > TO qd vc faz um FROM ele vai trafegar uma cartinha EXCHANGE(Binario) o ESB trafega um SOAP(XML).

Mule X Camel

Ambos Mule e Camel oferecem soluções leves, cada um tem uma abordagem distinta para a integração. Mule ESB é uma plataforma de ready-to-use com componentes estrategicamente alinhado, e é mais adequado para empresas que necessitam de um ESB ágil com um time-to-market rápida. Por outro lado, Camel escolheu para fornecer uma estrutura de esqueleto, colocando o peso técnico sobre a empresa para a construção de uma plataforma funcional. Esta abordagem DIY pode apelar para o desejo de um desenvolvedor para construir e consertar, mas rotineiramente provoca atrasos inesperados, perda de funcionalidade e pivôs atenção de necessidades de negócio.

Tabela 2- Camel X Mule

	Camel	Mule
Fácil Aprender	X	X
Diferentes Domain Specific Languages (DSLs)	X	X
Enterprise Integration Patterns (PEI)	X	X
Ambiente de desenvolvimento visual intuitivo		X
Conectividade: SAP/ETC..		X
Dezenas de Transportes	X	
Console Manager		X
Deploys Nas Nuvens integrado com IDE		X
Contanier JEE / TomCat		X

Ler mais

<http://www.devmedia.com.br/servicos-restful-com-apache-cxf-e-camel/26849>

<http://www.devmedia.com.br/introducao-ao-apache-cxf/26250>

<http://www.devmedia.com.br/introducao-ao-apache-camel/26278>