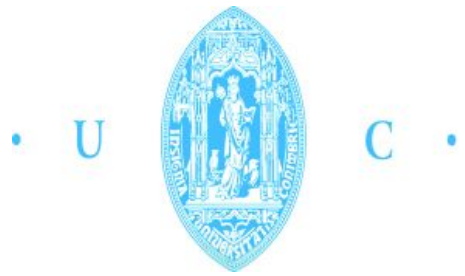


Faculdade de Ciências e Tecnologia

Universidade de Coimbra - DEI

IS Project 3 - Application integration with an Enterprise Service Bus(ESB)

Mestrado em Engenharia Informática

autores: João Batanete 2009113460, Ricardo Rei 2014233736

Introdução

Nesta secção iremos fazer uma breve descrição das tecnologias e metodologias utilizadas ao longo do projeto.

Service Oriented Architecture(SOA):

SOA é um estilo de desenvolvimento de software em que os serviços oferecidos por cada componente de um sistema são disponibilizados às outras componentes seguindo um protocolo de comunicação pré-definido. Um serviço pode ser entendido como a unidade básica de recursos do sistema acessíveis a partir de outra componente.

Caso um sistema seja implementado utilizando esta arquitetura, todos os serviços que o constituem irão obedecer às seguintes propriedades:

- representa uma entidade de "negócio" auto contida, que processa as suas funções de forma independente dos restantes serviços.
- é uma *black box* para as componentes do sistema que acedem a ela(ou seja, estes apenas possuem visibilidade dos *inputs* e *outputs* das suas funcionalidades).
- pode consistir ele mesmo noutros serviços internos.

Representational state transfer(RESTful) web services:

REST é um tipo de *web service* que utiliza um conjunto de operações *stateless* para satisfazer clientes*,* ou seja, não guarda informações de estado ou sessão entre chamadas ao serviço.

REST pode utilizar uma variedade de formatos para responder a pedidos(HTML,XML,JSON,...), bem como utilizar HTTP. Neste projeto iremos contudo utilizar apenas os formatos XML e JSON.

Simple Object Access Protocol(SOAP):

SOAP é um protocolo para troca de informação em *web services* baseado em XML que possui como principal vantagem a portabilidade simplificada para várias máquinas e sistemas operativos diferentes, devido ao uso de XML e protocolos como HTTP e SMTP, que praticamente todas as máquinas reconhecem nos dias de hoje. Uma vez que XML é relativamente simples de verificar em qualquer dos pontos de acesso, também é altamente seguro na validação de mensagens.

No entanto, o uso de XML impõe limitações de *performance* em relação a outras implementações de SOA menos rígidas, devido ao *overhead* extra do *parsing* das mensagens recebidas.

Enterprise Service Bus(ESB):

Um ESB é uma ferramenta baseada em SOA utilizada na integração de componentes de um sistema, atuando como ponto central nas trocas de informação de cada um, utilizando conectores de vários tipos, desde *sockets* TCP simples a serviços REST e SOAP, consoante as necessidades do sistema global. A utilização de um ESB pode reduzir exponencialmente a complexidade geral do sistema, evitando a utilização de comunicações *point-to-point* entre cada aplicação.

Neste projeto, o ESB utilizado será o Mule, um dos ESB's mais populares.

Mule:

Mule é um ESB baseado em Java produzido pela Mulesoft, um dos mais utilizados na indústria. Possui este nome devido a ser a "mula" que carrega o *overhead* extra de integrar vários componentes de uma aplicação pelos *developers*.

O desenvolvimento em Mule baseia-se na utilização de *flows* de informação constituídos por conectores(HTTP,SOAP,REST,...) que transportam as mensagens entre si conforme necessário. Também possibilita a utilização de scripts em várias linguagens de programação, como Java, Python ou Groovy. É também utilizada uma arquitetura orientada a eventos, uma vez que os flows habitualmente iniciam a execução aquando da receção de um evento, seja por receberem uma mensagem de um outro componente do sistema, um evento temporal ou meramente um referência a partir de outro *flow*(conector *flow reference*).

Os conectores disponibilizados pelo Mule também possibilitam interagir com uma variedade de websites e aplicações na internet de forma simples, como o gmail. twitter e facebook.

Usualmente, e no decorrer do projeto, aplicações Mule são desenvolvidas utilizando o IDE AnyPoint Studio, baseado em Eclipse.

Flow

Um *flow* representa uma linha de execução no Mule, iniciada após a ocorrência de um certo evento(exemplo: receção de um pedido de um cliente exterior). Pode ser feita facilmente uma analogia entre os *flows* e os métodos/funções numa linguagem de programação convencional. De notar que um evento pode atravessar vários flows ao ser tratado, através do uso do conector *Flow Reference*, que redireciona a linha de execução e o *payload* para outro *flow*.

Payload

O payload representa a “mensagem” trocada pelos vários conectores do *flow*. A execução destes depende, por norma, do conteúdo exato da mensagem.

Mule Conectors

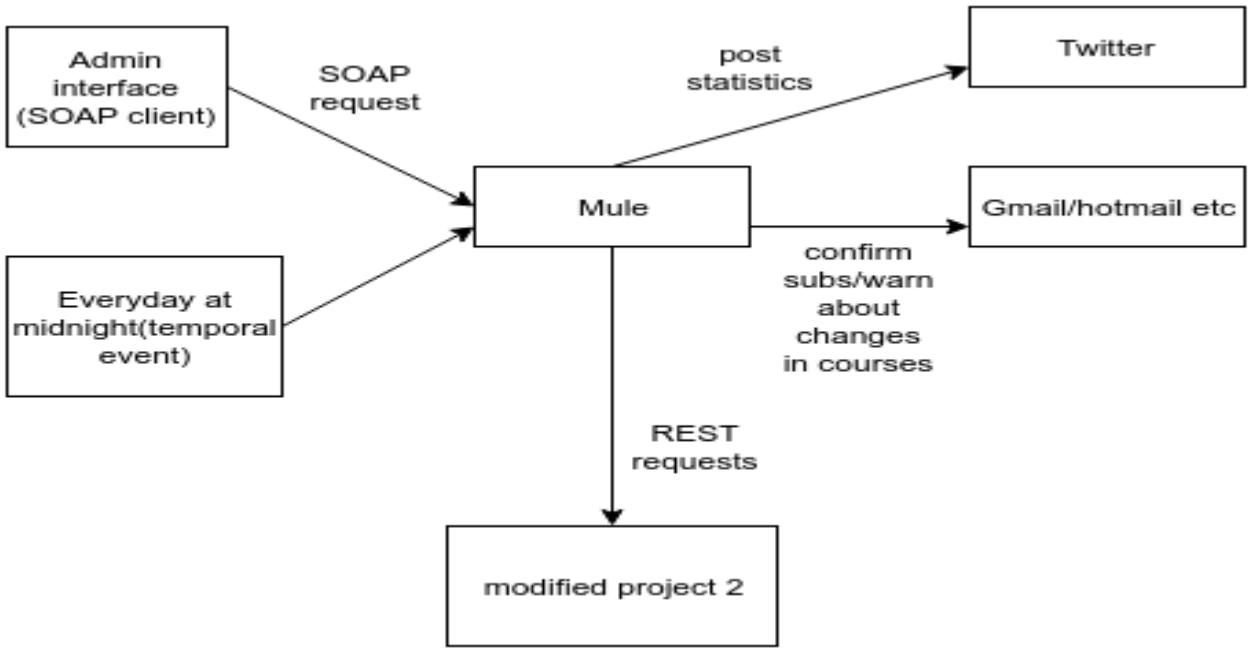
Os conectores Mule realizam uma certa fase de tratamento da mensagem(*payload*). Isto pode implicar vários tipos de tarefas diferentes, tais como a extração de informação de uma base de dados(conector *Database*) ou o tratamento de pedidos HTTP. Outros conetores utilizados, como o *Choice*, destinam-se a simular instruções *if* e *for* de linguagens de programação convencionais.

Session Variable

Variáveis de estado mantidas ao longo da execução de uma linha de execução do Mule(possivelmente, ao longo de vários *flows*) e apagadas automaticamente após o tratamento do evento que a despoletou, utilizadas extensivamente ao longo do projeto.

Arquitetura

O sistema encontra-se estruturado da seguinte forma:



Nas secções seguintes iremos descrever o funcionamento dos vários flows de Mule, bem como as alterações feitas ao projeto 2 que possibilitam as chamadas REST.

Mule(explicação dos *flows*)

Nota geral: uma vez que vários componentes do Mule são, devido ao design do mesmo, bastante intuitivos e com funcionalidade *self-explanatory*, optamos por uma descrição mais generalizada da utilidade de cada flow, explicitando as opções tomadas onde tal se justificar.

SOAPFlow:

Este flow destina-se a processar os requests SOAP feitos a partir da **Admin CLI**.

Embora exista a possibilidade de implementar o *web service* utilizando Java puro, optámos pela implementação direta no Mule com conectores, utilizando apenas uma interface com as assinaturas dos métodos. Os métodos na interface(SubscriberService) são os seguintes:

- createSub: cria uma nova subscrição, para um certo email e nome de curso(necessita de confirmação pelo email).
- listSubs: devolve uma lista de todas as subscrições existentes.
- delSub: apaga uma subscrição(necessita de confirmação pelo email).

Utilizando o conector CXF associado à interface Java, o *payload* é transformado no envelope XML com o método chamado e os argumentos utilizados. Após o parsing destes, é utilizado um componente *Choice* para realizar a operação pretendida, e devolver o *output* ao cliente.

Confirmop:

Este flow destina-se a confirmar as operações de criação ou deleção de subscrições a cursos efetuadas anteriormente pelo admin.

Os pedidos são inicialmente recebidos por um conector HTTP, e em seguida os parâmetros são guardados em *Session Variables* e é verificado se na base de dados se a confirmação é válida(exemplo: caso estejamos a tentar confirmar a deleção de uma subscrição, deverá existir na base de dados uma entrada com essa subscrição e com o campo *forDeletion* a 1). Caso a operação seja válida, será processada e o cliente receberá uma mensagem a informá-lo de que foi atendido com sucesso no browser. Caso contrário, receberá uma mensagem de erro.

Emails:

O flow "emails" tem como objetivo o envio de um email de confirmação a um dado subscritor para que este aceite a sua respectiva adição ou remoção da lista de subscritores.

Este flow começa com um componente *choice*, que recebe o tipo de acção pretendida (adição ou remoção) e encaminha a mensagem para a opção correta. No caso da operação pretendida ser de confirmação de adição à lista de subscritores, é enviado um email com o respectivo link de confirmação de adição ao subscritor. No caso da operação

pretendida ser de remoção da lista de subscrição, é enviado um email com um link para que o subscritor possa confirmar a sua remoção da lista. Após o envio do email, o último componente deste flow tem como objetivo atualizar o número de emails enviados para que seja possível mostrar esta informação posteriormente no Twitter.

//TODO(OUTROS FLOWS)

Outros componentes do projeto

Admin CLI(contido no projeto Mule):

Este programa é bastante simples, e implementa uma interface baseada em texto que o admin pode utilizar para aceder ao *web service* SOAP implementados no Mule. Para tal são utilizadas classes geradas automaticamente a partir do ficheiro WSDL(Web Service Descriptor Language) do WS utilizando o comando wsimport.

Projeto 2(modificado):

Esta aplicação(EAR) é equivalente à submetida anteriormente, com as seguintes modificações:

- Foi adicionado um novo projeto, ISproj3-rest, com um servidor REST implementado. Este servidor possui apenas dois métodos que são detalhados abaixo.
- O projeto IS-proj2-CRUD sofreu ligeiras alterações por forma a ser possível obter POJO's(Plain Old Java Objects) representantes das entidades JPA a que o servidor REST pretende aceder.

Os dois métodos mencionados são os seguintes:

- getAllStudents(acedido a partir de getstudents no browser): devolve uma representação JSON da lista dos estudantes do curso com id dado, criada a partir de POJO's.
- getAllMaterials(acedido a partir de getmaterials no browser): devolve uma representação XML da lista de cursos e os materiais que estes contém, criada a

Base de dados:

Neste projeto, a base de dados é muito simples e foi apenas utilizada como auxiliar para

aceder a dados persistentes de sessões anteriores de Mule. As tabelas utilizadas encontram-se detalhadas abaixo.
///TODO

Nota: a tabela EmailCount apenas possui uma entrada, uma vez que não necessitamos de aceder aos dados de emails de dias anteriores.

Testes realizados

Nesta secção iremos descrever os principais testes realizados ao funcionamento da aplicação antes da sua submissão.

//TODO

Operação/requerimento	**Testes realizados e output esperado**
1-script para criar/apagar admins	● criar admin que não existe->admin deve ser criado ● apagar admin que existe->admin deve ser apagado ● tentar criar admin que já existe->erro ● tentar apagar admin que não existe->erro ● tentar logar como admin inexistente ou previamente apagado->erro
2-como admin, criar novos alunos e professores	● criar um estudante ou professor que não existe->utilizador deve ser criado ● apagar estudante ou professor que existe-> deve ser apagado ● tentar criar estudante ou professor que já existe->erro ● tentar apagar estudante ou professor que não existe->erro ● tentar logar como estudante ou professor inexistente ou previamente apagado->erro
3-como admin, editar campos de informação de um utilizador	● editar campo existente e comum aos 3 tipos de Person, quando a pessoa existe->campo editado com sucesso ● editar campo que só existe num tipo de Person, numa person desse tipo->campo editado com sucesso ● editar campo que só existe num tipo de Person, numa person diferente desse tipo->erro ● editar qualquer campo numa Person que não existe->erro ● editar campo que não existe->erro
4-como utilizador não autenticado, apenas ter	● procurar entrar em qualquer página da app

acesso à página de login	sem estar logado->erro ● procurar entrar em qualquer página da app para que não temos permissão(exemplo:entrar como prof no menu do admin)->erro ● entrar numa página para a qual temos permissão->sucesso
5-realizar o login e com o email da instituição e a password	● user e pass corretas->aceder a página que corresponde ao tipo de utilizador logado ● user e pass errados->refresh na página de login
6-realizar logout a partir de qualquer localização	● fazer logout->cookies apagadas e página de login mostrada
7-como admin, criar um curso com uma lista de alunos e professor, e mudar qualquer informação presente nele	● criar curso(curso não existe, professor e alunos existem)->sucesso ● criar curso(curso existe)->erro ● criar curso(professor não existe)->erro ● criar curso(pelo menos um aluno não existe)->erro ● adicionar student a curso(student existe, não está no curso)->sucesso ● adicionar student a curso(student não existe)->erro ● adicionar student a curso(student existe, já está no curso)->sucesso ● renomear curso(curso existe, não existe curso com o novo nome)->sucesso ● renomear curso(curso não existe)->erro ● renomear curso(curso existe, curso com o novo nome existe)->erro
8-como professor, fazer upload a materiais de um curso lecionado por mim	● upload de um material(curso existe,material não existe no curso)->sucesso ● upload de um material(curso existe,material existe no curso)->erro ● upload de um material(curso não existe)->erro ● upload de um material(professor não leciona curso)->erro
9-como professor, remover materiais de um curso lecionado por mim	● apagar um material(curso existe,material não existe no curso)->sucesso ● apagar um material(curso existe,material existe no curso)->erro ● apagar um material(curso não existe)->erro ● apagar um material(professor não leciona curso)->erro
10-como qualquer utilizador logado, listar e fazer o download de materiais de qualquer curso	(materiais foram listados e descarregados sem problemas)
11-como professor, listar os alunos de um determinado curso lecionado por mim ordenados por	● professor leciona curso->lista corretamente ordenada

ordem crescente ou decrescente	• professor não leciona curso->erro
12-pesquisar por alunos segundo uma lista de critérios	• sempre sucesso(caso não exista nenhum aluno é mostrada uma lista vazia)
13-como administrador, apagar qualquer tipo de dados do sistema	• apagar qualquer entidade inexistente->erro • apagar professor->todos os cursos lecionados por ele metem o campo professor_id a NULL. • apagar administrador->sucesso • apagar aluno->removido da lista de estudantes de todos os cursos que estudava • apagar curso->removido de todas as listas de cursos dos seus estudantes e professor, todos os seus materiais são apagados • apagar material->removido da lista de materiais do seu curso

Referências

- slides da disciplina
- http://eai-course.blogspot.pt (blog do professor)
- https://en.wikipedia.org/wiki/Enterprise_service_bus
- https://en.wikipedia.org/wiki/Service-oriented_architecture
- https://en.wikipedia.org/wiki/Representational_state_transfer
- https://en.wikipedia.org/wiki/SOAP
- https://en.wikipedia.org/wiki/Mule_(software)
- https://www.draw.io