FCTUC – DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Integração de Sistemas
TP3 - Relatório

Relatório do Projecto

João Nuno Duro, Luís Silva Perdigão, Pedro Miguel Macedo

{jduro,perdigao,pmosm}@student.dei.uc.pt

1. Introdução

Este projecto têm em vista reestruturar a arquitectura do *Phasebook,* anteriormente desenvolvido, para arquitectura *SOA* (*Service Oriented Architecture).* De modo a obter sucesso na implementação desta nova arquitectura foram utilizadas novas tecnologias, tais como *Web Services(WS)*, Enterprise Service Bus *(ESB)* e *Entreprise JavaBeans(EJB)*, de modo a completar as necessidades existentes na transição entre a arquitectura antiga para a nova.

Para tal, uma das necessidades primárias, foi a separação a base de dados utilizada. Esta separação consiste em cortar todas a relações existentes entre as tabelas e recriando, assim, um ambiente em que existem bases de dados independentes. Isto permite simular a separação entre departamentos, por exemplo, o departamente de Clientes tem a sua própria base de dados e só tem acesso aos dados referentes a clientes.

De seguida será descrita a arquitectura e opções escolhidas no desenvolvimento deste projecto.

2. Arquitectura

2.1. Descrição Geral

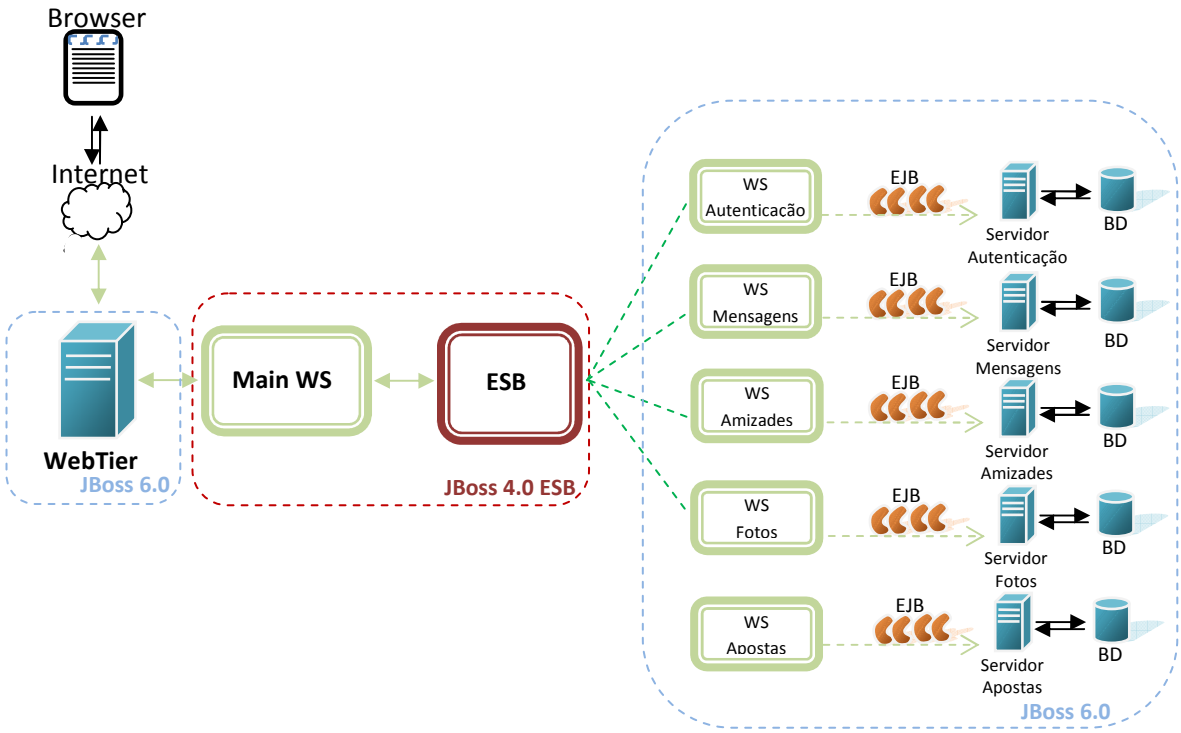

Figura 1

Como é possível observar na *Figura 1*, a arquitectura viza utilizar *WebServices* de modo a poder dar resposta a todas as funcionalidades existentes. No nosso caso não foi possível conseguir que todas as funcionalidades tivessem o seu *WebService*, em especial o servidor de apostas.

São utilizados dois servidores Jboss: O Jboss 6.0 está a hospedar o *WebTier* (onde se faz o parse dos *JSP* e onde todos os serviços são invocados) e todos os *WebServices* que podemos observar a direita no esquema. No Jboss *ESB* 4.0 será hospedado um *Main WebService* e o Enterprise Service Bus (ESB) que irá invocar os diferentes serviços, funcionando como um Entreprise Appication Integration(EAI).

Uma das tecnologias exigidas neste projecto era o uso de um projecto *jBPM* de modo a implementar um Orquestrador que controlasse um ou dois processos, na nossa arquitectura não é visível esse módulo pois não foi implementado devido às dificuldades deparadas durante a implementação/configuração dos restantes módulos.

De modo a tentar manter a ideia inicial foi, de certa forma, simulado a sequência de pedidos independentes aos serviços por parte do *Main WebService.* Isto é, no caso do *getFriends* é invocado o serviço do departamento das relações de modo a obter todas as relações respectivas a um utilizador. Depois, para apresentação destes mesmo dados na página *web*, é necessário

conter informação do nome do cliente, e a sua foto de perfil. Para isso no *Main WebService*, após receber a informação das relações (pendentes e actuais), segue-se o pedido de informação sobre o cliente, ao departamento de clientes, de quem a relação de amizade se destina e a foto de perfil (departamento de fotos) desse mesmo cliente. Assim, ao encaminhar os pedidos de dados especificos de forma sequencial aos diferentes departamentos, é possível simular o comportamento do controlo do Orquestrador e garantir que um departamento não aceda a dados de outro departamento.

Cada departamento contem o seu *WebService* disponível que, utilizando os *EJBs*, elabora a operação do pedido acedendo a cada base de dados. O resultado é retornado e tratado em todas as fazes do caminho de retorno até que é possível dispor a informação de sucesso ou insucesso do pedido efectuado.

2.2. Comunicação entre WebTier e MainWebService

Cada acção disponivél nas páginas do *PhasebookWebTier* correspondem a uma invocação do método respectivo disponivel no *PhasebookMainWS*. Assim foi necessário alterar, em todas as páginas, o acesso ao serviço disponível no *MainWebService* de modo a cortar a anterior comunicação entre a página e os *EJB*.

Devido às dificuldades, anteriormente referidas, não foi possivel conseguir uma conversão total das invocações directas aos *EJB*, mas sim das mais importantes e centrais. O processo de registo, login, editar informações pessoais, adicionar e responder ao pedido de amizade, procurar utilizadores, enviar mensagens, entre outros, foram reformulados de modo a invocar o *WebService* respectivo.

Onde foi escolhido manter o acesso directo ao *EJB* foi por exemplo, na opção de visualizar os amigos está presente, entre parêntises, o numero actual de amigos para essa contagem está a ser feito o pedido directamente ao *EJB*. Do mesmo género é feito o pedido de informação do nome e da fotografia nos casos em que é listada a lista de utilizadores da rede e amigos.

Em relação a visualizar a galeria de fotogafias e apostas não é utilizado nenhum *WebService*, tento assim só invocações directas aos *EJBs* respectivos.

2.3. Main WebService

No *Main Web Service* estão disponíveis os métodos que reenviam os pedidos para o *PhasebookESB*. Em cada chamada são tratados dos dados a enviar como argumentos e a invocação do serviço disponibilizado pelo *ESB*. Após a invocação é feito o tratamento dos dados da resposta e enviados para o *WebTier*.

2.4. Phasebook *ESB*

Neste módulo estão definidos os diversos serviços disponíveis implementados. Para cada serviço são criadas duas *queues* uma por onde as mensagens de pedidos chegam e outra para o envio da resposta.

Os serviços implementados são so seguintes:

Client

- checkLogIn
- createUser
- editProfile
- getFriendsInfo
- getClientInfo
- getSearch

Message

- getPosts
- sendNewMessage

Photo

- addPhoto

Relation

- getRelations
- getPendingRelations
- addFriends
- acceptFriend
- declineFriend
- removeFriend

Em cada serviço existe uma acção que encaminha o fluxo para a action myInitializer que contém um método para cada serviço de modo a preparar os dados para o envido desses mesmos para o *WebService*.

Depois dos dados preparados o W*ebService* é invocado e depois existe uma acção que encaminha o fluxo para a action *responseHandler* que faz o tratamento dos objectos retornados pelo *WebService* de modo a criar o objecto/lista de e retorna para a *queue* de modo a chegar ao ponto de partida (*Main Web Service)*.

2.5. Phasebook *Web Services*

Aqui estão todos os web services criados de modo a aceder ao respectivo *EJB* e proceder a execução da funcionalidade. Existe um *web service* para cada departamento com os seus métodos de modo a manter a simulação da separação fisica entre os departamentos.

2.6. Trabalho pendente

Como já foi sendo referido anteriormente existem aspectos que não foram possíveis implementar. Deste modo pode-se considerar trabalho futuro a implementação do sistema de autenticação e verificação em cada serviço se o *token* de sessão é válido e está dentro dos limites de tempo atribuido. Na transição de arquitectura foi forçado a desmantelar o sistema de segurança antigo ficando assim, neste momento, a aplicação desprotegida.

Algo a implementar seria, então, o processo de orquestração para, por exemplo, o *getPosts e* para a listagem das amizades correntes/pendentes.

Acabar a transição dos restantes pedido directos ao *EJB* por parte do *WebTier* para *WebServices*.

3. Desempenho dos elementos

Foi possível fazer uma divisão de responsabilidades e de trabalho uniforme pelos diversos elementos do grupo, onde cada elemento acabou por trabalhar em todas as áreas, desde a a separação da base de dados em módulos, na criação do mecanismo de serviços desde a sua invocação à criação de todos os nós por onde passam, mudança nas funcionalidades e métodos nos *EJB*, entre outros.

Podemos concluir que em termos de trabalho colectivo foi um grupo bem sucedido no ponto de vista que todos os elementos tiveram a sua parte em qualquer aspecto e parte do projecto, podendo assim ter a possibilidade de lidar com as diferentes tecnologias utilizadas.

Embora tenha existido uma boa coordenação do grupo de trabalho, as inúmeras complicações encontradas na configuração e execução do Jboss e Jboss ESB, levaram a que o grupo tivesse que investir uma carga de esforço excessivo na execução do projecto e sem os resultados esperados na avaliação deste mesmo.

O número de horas dispendidas por elemento no trabalho:

João Nuno Duro: ~ 110 horas

Luís Silva Perdigão: ~ 110 horas

Pedro Miguel Macedo: ~ 110 horas

4. Conclusão

As arquitecturas baseadas em serviços são muito complexas e complicadas de implementar. É necessário um cuidado excepcional para conseguirmos um panorama eficiente, sem falhas e seguro. É fácil de ver que empresas com visitas diárias superiores a um certo limite têm que adoptar uma arquitectura baseada em serviços uma vez que se torna insuportável dar resposta a todos os acessos tendo só máquina ou serviço central.