

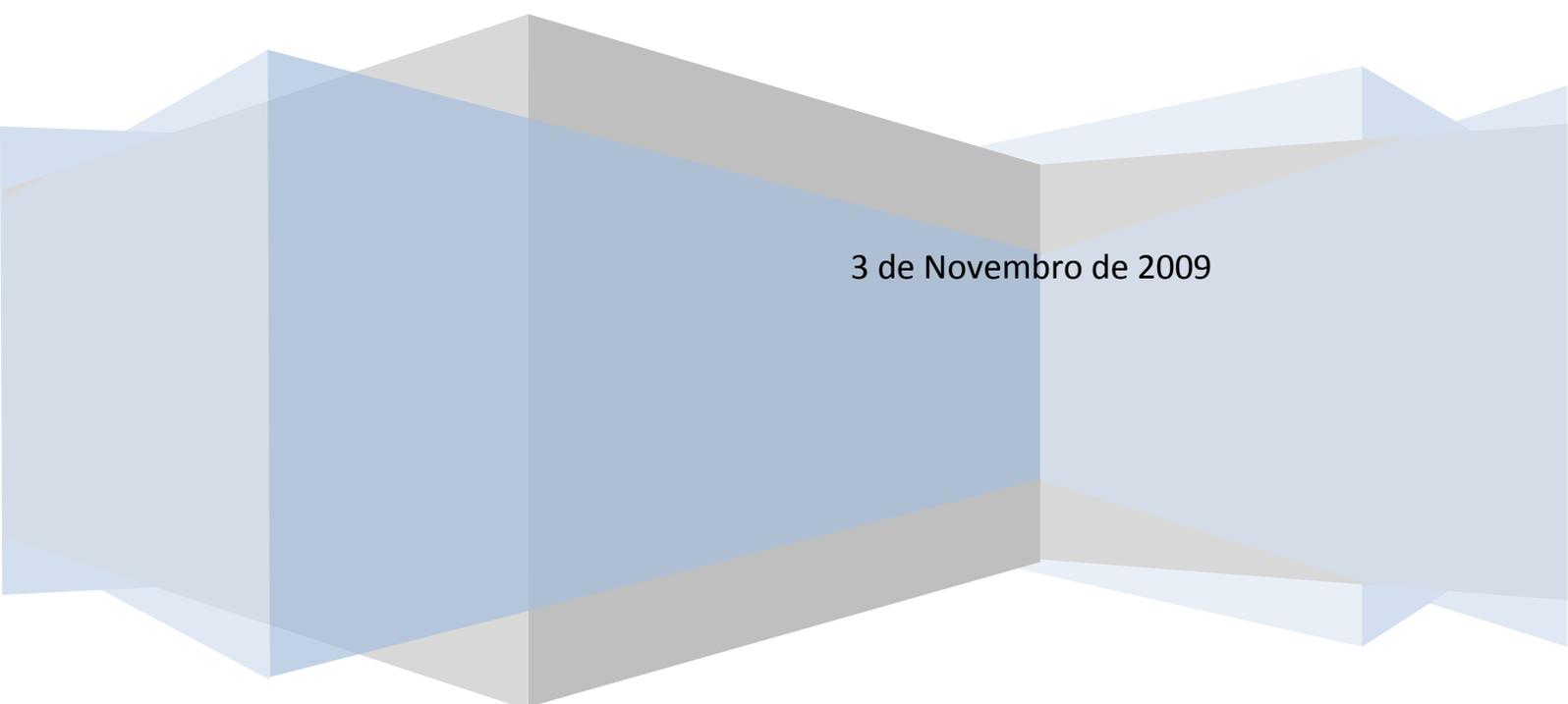
Departamento de Engenharia Informática
Mestrado em Engenharia Informática

Relatório - Trabalho Prático 2

Integração de Sistemas

Francisco Ferreira, João Lopes e Marco Simões

3 de Novembro de 2009



Introdução

Este segundo trabalho prático tem como objectivo a criação de uma aplicação que permita a compra de câmaras fotográficas. Este serviço irá ser disponibilizado ao utilizador final num Web-Site, que irá comunicar com a camada Web da nossa aplicação, que irá responder aos pedidos do utilizador. Esta camada comunica com um *container* de Enterprise Java Beans, que por sua vez utiliza alguns serviços externos, através de chamadas a Web Services.

Tal como requerido, e como é objectivo deste trabalho conseguir integrar e utilizar vários sistemas com a nossa aplicação, tivemos de utilizar vários *web-services* externos. Esses serviços são *shipping-department*, que trata do envio das encomendas, e o *camara-supplier*, que é um sistema com uma base de dados com todas as câmaras disponíveis. A este último, a nossa aplicação faz pedidos de pesquisa, caso o cliente queira uma câmara que não esteja no nosso servidor.

Implementação

Na implementação foram bem definidas as várias camadas da aplicação:

- A *presentation-layer* serve para apresentar o site sob o formato *html* com *javascript*, através do protocolo *http*, que, de uma forma clara e simples, mostra ao utilizador as suas opções e escolhas e tudo o que poderá fazer no sistema. Esta camada é suportada por um Dynamic Java Web Project, a correr no servidor jboss.

- A *logic-layer* é a implementação central do projecto, permitindo à camada anterior utilizar todas as funções necessárias para o funcionamento do sistema e interactividade do utilizador final com o serviço. Esta comunicação é feita através de RMI. Nesta fase, é também utilizado JAX para comunicar com os outros provedores de serviços externos, como o *shipping-department* e o *camara-supplier*, como descrito anteriormente. É também neste servidor que é feito o acesso a uma base de dados (*data-layer*) para certos serviços como validação dos utilizadores ou registo de compras.

- Na *data-layer* estão localizados os dados sobre os utilizadores, as suas compras e a informação das câmaras importadas do outro servidor. A comunicação a esta informação é feita via JPA sobre Hibernate. Na implementação da *camara-supplier* utilizamos JDOM na pesquisa da câmara. A lista de câmaras apresenta-se sobre a forma de ficheiros XML. Na figura 1 está representado o esquema de todo este projecto, e todas as suas componentes.

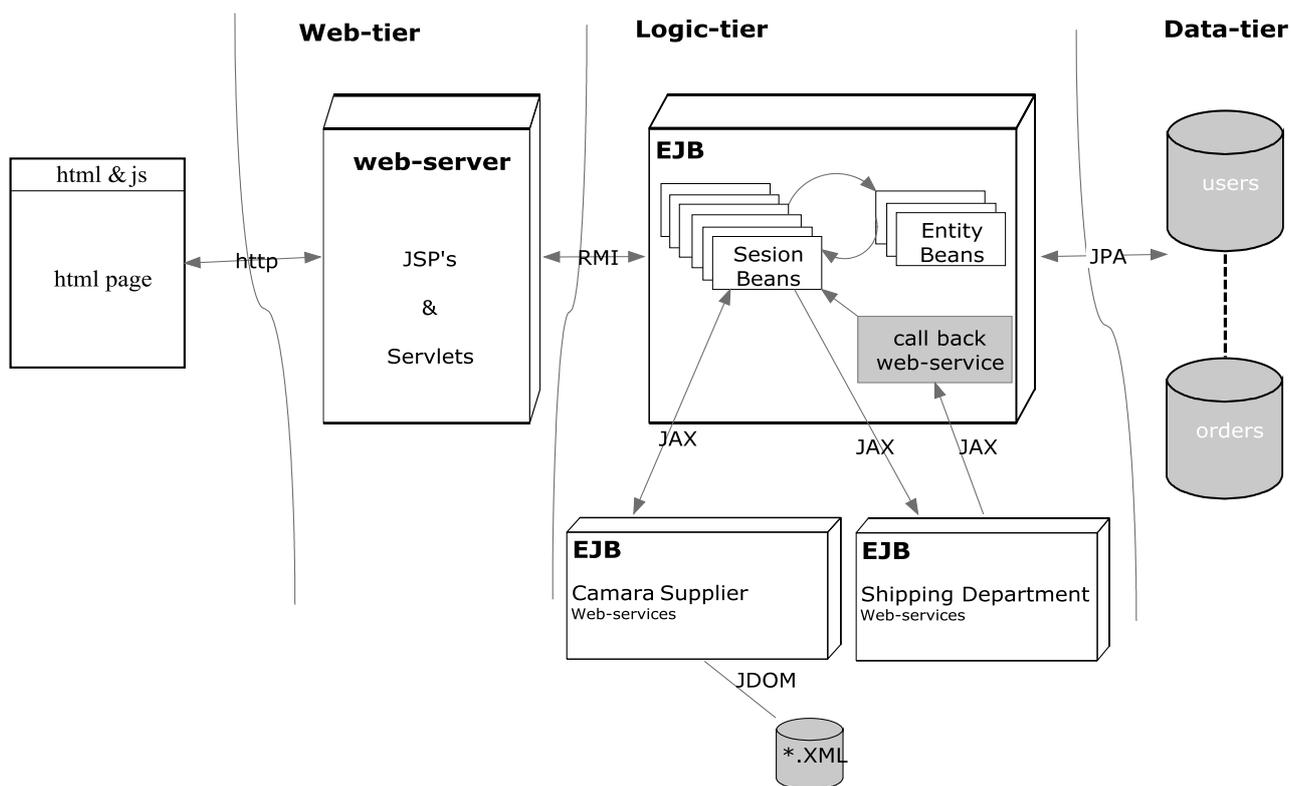


Figura 1 - Arquitetura geral do projecto

Como podemos observar temos as várias camadas (*layers ou tiers*) bem definidas e separadas, o que permitiu uma fácil distribuição de tarefas assim como uma implementação por metas. Em cada meta era possível testar cada fase que fosse implementada no período. Numa última parte do projecto procedeu-se à integração de todas as camadas, deixando assim o sistema funcional.

Preferimos focarmo-nos mais numa implementação robusta do que numa interface "bonita", seguindo, no entanto, os princípios de *Web-development* estruturado. Temos por isso uma interface visual um pouco simplificada, mas que qualquer designer consegue facilmente trabalhar, editando simplesmente o ficheiro de *cascading style sheet*.

De seguida está explicado cada uma destas *layers*, a sua arquitectura e implementação.

Presentation-Layer

A camada de apresentação consiste num conjunto de páginas JSP (Java Server Pages), que utilizam um *bean* de sessão local para guardar todos os dados necessários.

As páginas apresentadas ao utilizador contêm *javascript* para um melhor controlo e ajuda à interacção do utilizador com os serviços disponíveis. Implementamos esta camada de maneira a que o uso dos serviços apresentados fosse feito de forma simples e clara.

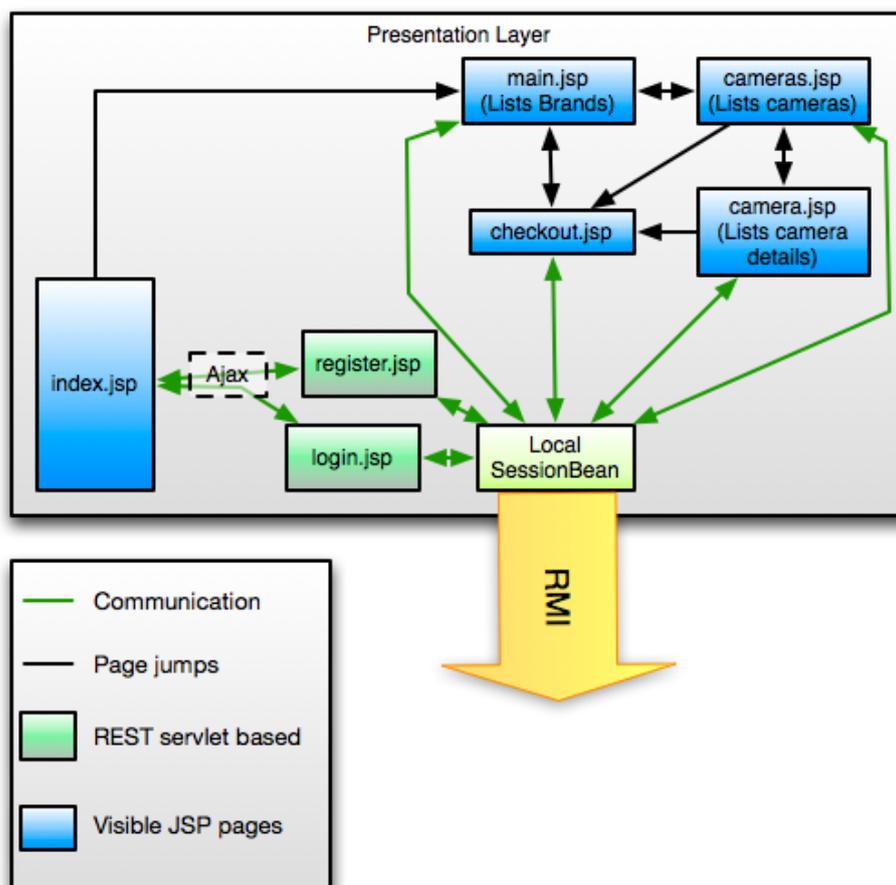


Figura 2 - Arquitectura da presentation-layer

Como podemos observar esta *layer* contém três tipos de ficheiros.

- Páginas visíveis de JSP, que o utilizador usa para navegação e introdução de dados;
- Páginas "invisíveis" de JSP, que funcionam como serviços REST. As páginas principais comunicam com estes *servlets* através de *HTTPPOSTs*. Utilizam AJAX através do seu *javascript* interno;
- *Bean* de sessão local, que é instanciado a cada login. Tem ligações RMI com o sistema da próxima camada. É este *bean* que faz o tratamento de todos os dados, bem como mantém a informação sobre o Cart do utilizador, com todas as câmaras que este deseja adquirir.

A comunicação via AJAX usa JSON (JavaScript Object Notation), facilitando os callbacks. São efectuadas ligações por AJAX de dois conjuntos de páginas para três *servlets*.

A comunicação com os outros layers da aplicação é feita a partir de chamadas RMI, existentes no *session-bean* local, ao *web-server* principal da camada lógica.

Logic-Layer

É nesta camada que é definida e implementada toda a estrutura e arquitectura da aplicação.

Usamos dois *web-servers* externos. O *shipment-department* e o *camara-supplier*. O **shipment-department** é um serviço que simula o envio de encomendas feitas pelos utilizadores. É feito um pedido, através de JAX, para envio de uma encomenda. Este pedido é registado no serviço, que inicia uma thread para efectuar o *shipping* e retorna. Esta thread, passado algum tempo (3-7 segundos), avisa que a encomenda chegou. Durante este tempo obviamente que o servidor não se encontra bloqueado a espera da resposta, pois num caso real poderia demorar vários dias. Este aviso de recepção é feito através de um pedido a um *web-server* existente no EJB (Enterprise Java Bean) principal.

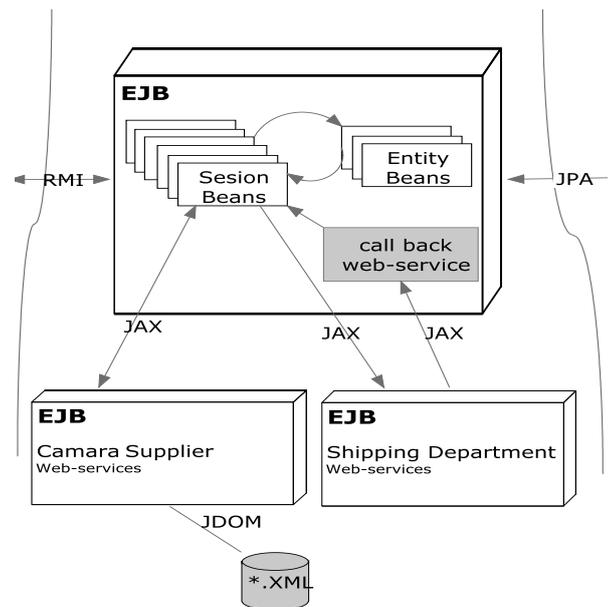


Figura 3 - Arquitectura da Logic Layer

Este Web-service (**call back Web-service**), apesar de estar dentro do container principal, utiliza **rmi** para comunicação com os **Session Beans**. Esta decisão de implementação foi tomada para garantir a independência do serviço, permitindo assim, por questões de escalabilidade, a qualquer momento ser desintegrado do EJB e passar a correr numa máquina diferente, por exemplo. Este Web-service, ao receber o pedido do shipment department, comunica com um Session Bean para que este actualize o estado da ordem na base de dados. Envia também um e-mail ao utilizador com a confirmação da entrega da encomenda.

O **camara-supplier** é um *web-server* que recebe um pedido de procura e retorna a primeira câmara cujo nome contém todas as palavras da pesquisa introduzida. Sincronamente, responde ao servidor com o resultado. Se não existir resultados, retorna null.

Para a comunicação entre os *Beans* e os dados persistentes, foi utilizada a API JPA (Java Persistent API) sobre *hibernate*. A lógica propriamente dita é mantida pelos Session Beans. A nossa aplicação utiliza três Session Beans para gerir os Entity Beans. Assim, estes estabelecem uma interface pública para gestão do catálogo, dos utilizadores, das encomendas e dos itens das encomendas.

Optámos por utilizar apenas Stateless Session Beans, pois estes garantem uma melhor performance, desenhando assim a aplicação de modo a não precisar de guardar informações de sessão do lado do EJB container. A informação de sessão foi guardada na camada de apresentação, no Web-server, de maneira a o Cart ser preenchido e mantido desse lado, enviado para o EJB aquando do checkout.

Data-Layer

No que toca a **persistência** de dados, o nosso server mantém uma estrutura de objectos relacionados que são mapeados para uma base de dados Hypersonic, através de JPA.

O diagrama de classes seguinte mostra a estrutura de dados guardada em persistência (Entity Beans):

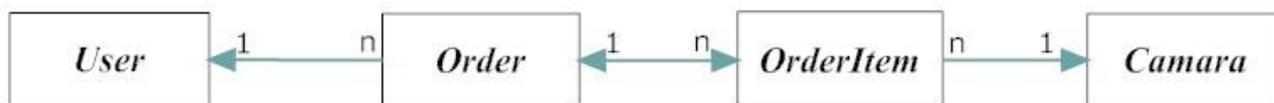


Figura 4- Diagrama de Entidades (com respectivas relações)

Para cada Entity Bean existia um Stateless Session Bean responsável por todos os métodos de persistência (adicionar, obter, editar, etc). Estes Session Beans utilizavam o Entity Manager da JPA para garantir a consistência das operações.

Os **três** Stateless Session Beans utilizados para esta gestão foram:

- UserManager – gestão de utilizadores;
- CatalogManager – gestão do catálogo das câmaras;
- OrdersManager – gestão das encomendas e respectivos items.

Existe ainda um quarto Stateless Session Bean que utilizamos. Este serve para criar uma interface de pesquisa de câmaras (chama depois o Web service camera supplier) para a camada de apresentação.

Conclusão

Tivemos alguns problemas a colocar o servidor JBOSS a funcionar correctamente. Conseguimos, no entanto, coloca-lo a funcionar de forma correcta e robusta na versão final do projecto.

Com todos os problemas encontrados e resolvidos, em conjunto com os objectivos do projecto foi-nos possível aprender a desenvolver aplicações com os *tiers* bem definidos, utilizando o modelo Java 2 Enterprise Edition, assim como as várias tecnologias envolventes em cada camada do projecto.

Horas de Trabalho:

Developer	Pesquisa	Desenvolvimento	Relatório
Marco Simões	10	18	2
João Lopes	12	10	8
Francisco Ferreira	2	20	3

3 de Novembro de 2009

Francisco Ferreira
fmsf@student.dei.uc.pt
2006124182

João Lopes
jmlopes@student.dei.uc.pt
2006125131

Marco Simões
msimoes@student.dei.uc.pt
2006125287

Integração de Sistemas
Mestrado em Engenharia Informática no
Departamento de Engenharia Informática da
Faculdade de Ciências e Tecnologia da Universidade de Coimbra