Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia de Redes de Comunicação e Multimédia

4º Semestre Lectivo 2013/2014

Disciplina de

Infra-estruturas Computacionais Distribuídas

2º Trabalho Prático

Alunos:

João Sousa, nº 35749

Jorge Fernandes, nº 39372

Índice

Índice de Figuras

Objectivos

Este trabalho prático tem como ponto de partida o primeiro trabalho prático da unidade curricular de Infra-estruturas Computacionais distribuídas, no qual se desenvolveu um sistema para gestão de informação espacial capaz de armazenar, gerir e representar Pontos de Interesse Geográficos. Com a nova iteração sobre o sistema desenvolvido, pretende-se que a informação anteriormente mostrada ao utilizador através de uma consola passe a ser apresentada no *browser*, continuando a informação de POIs alocada no servidor.

Para que tal seja exequível, dar-se-á uso a tecnologias para a *World Wide Web* quer *client side*, quer *server side*, nomeando-se *Applets*, *Servlets* e *JavaServer Pages*. De modo a conseguir a comunicação entre os clientes e o servidor utilizar-se-ão *Sockets*.

Dado o exposto, seria objectivo proceder à alteração da aplicação apenas do lado do cliente, parte responsável pela interface gráfica. Contudo, surgiu a necessidade de repensar todo o sistema e comunicação.

Como se pode verificar no relatório da primeira versão do trabalho, o protocolo de comunicação era bastante rudimentar, incapaz de processar pedidos independentemente do conhecimento do estado do sistema. Deste modo, para a realização da nova abordagem surgiu a necessidade de reconsiderar o protocolo de comunicação a utilizar, visto que utilizando as tecnologias citadas é necessário tratar de cada pedido de forma independente. Através do protocolo a implementar, o servidor terá de saber impreterivelmente qual o pedido e responder a esse com toda a informação necessária a apresentar ao cliente.

Deste modo, para além da redefinição do cliente e de como o servidor trata os pedidos, recorrer-se-á às alterações necessárias no modo como a informação é enviada e recebida de e para o servidor.

Relativamente ao tratamento da informação alocada em ficheiros do lado do servidor, este dar-se-á de modo idêntico ao primeiro trabalho.

Desenvolvimento do Sistema

Ficheiros XML

Relativamente aos ficheiros XML nos quais os dados são armazenados não houveram alterações significativas, apresentando-se de seguida os diagramas de nós representativos das estruturas de dados. Sendo que agora cada Poi pode ter uma imagem associada, acrescentou-se essa informação à estrutura representada na *Figura 2* - Estrutura dos ficheiros de *POIs* da região. Referencia-se o relatório da primeira parte do trabalho prático para uma abordagem mais detalhada sobre estas estruturas.

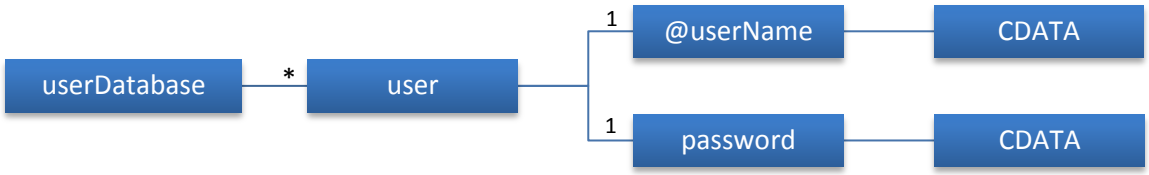

Figura 1 – Estrutura do ficheiro de dados de Utilizadores

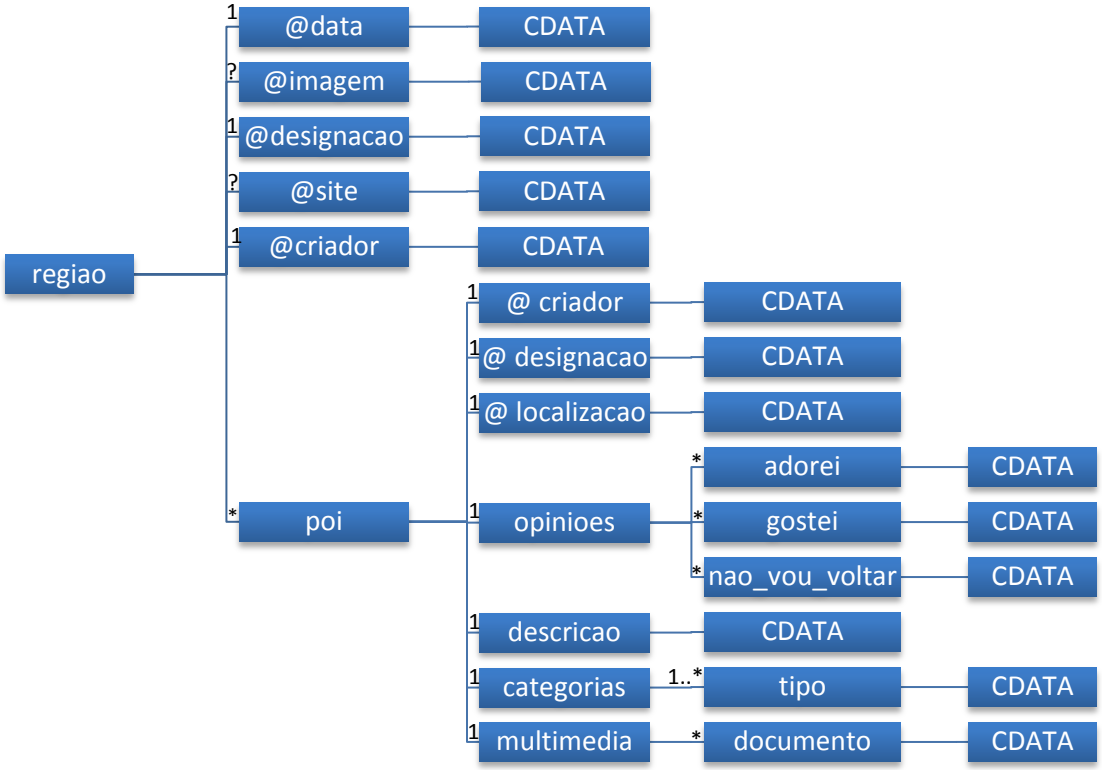

Figura 2 - Estrutura dos ficheiros de *POIs* da região

Mensagens XML

Tal como já foi referido nos objectivos, foi necessário redefinir uma estrutura de dados que permita a representação dos diversos pedidos e respostas de forma concisa.

Repare-se que se necessitará apenas de dois tipos de mensagens:

- Listas de parâmetros textuais sem hierarquia:
  - Utilizado em todos os pedidos do cliente, que envia tipo de pedido e os *n* parâmetros necessários para detalhar esse pedido;
  - Utilizado pelo cliente também para enviar os dados de login e registo;
  - Utilizado nas respostas do servidor sempre que não necessita de enviar estruturas hierárquicas complexas para o cliente. São exemplos a listagem de todos os utilizadores, POIs, regiões e categorias.
- Listas de POIs:
  - Há excepção das mensagens abordadas anteriormente, todas as restantes são listas de POIs. Para tal, bastará enviar-se uma estrutura semelhante à de armazenamento dos POIs em disco, contendo todos os POIs que constituem a resposta para o cliente.

É de notar que quando a mensagem é uma lista de parâmetros textuais há que dar a conhecer qual o parâmetro a que cada valor está associado. Para que tal seja possível, neste tipo de estruturas XML, cada valor está dentro de um nó com o nome do parâmetro respectivo.

Apresentam-se de seguida as estruturas dos dois XML trocados entre cliente e servidor.

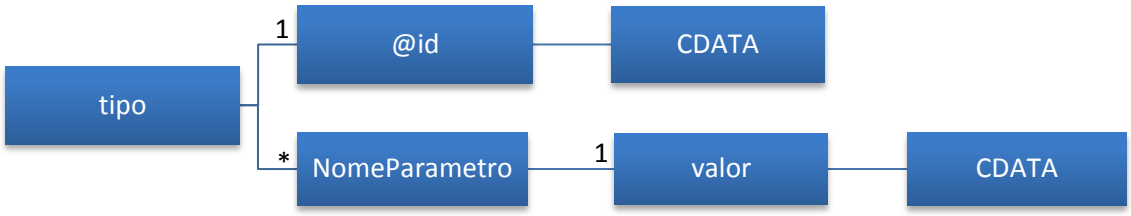

Figura 3 - Estrutura das mensagens de listas de parâmetros textuais sem hierarquia

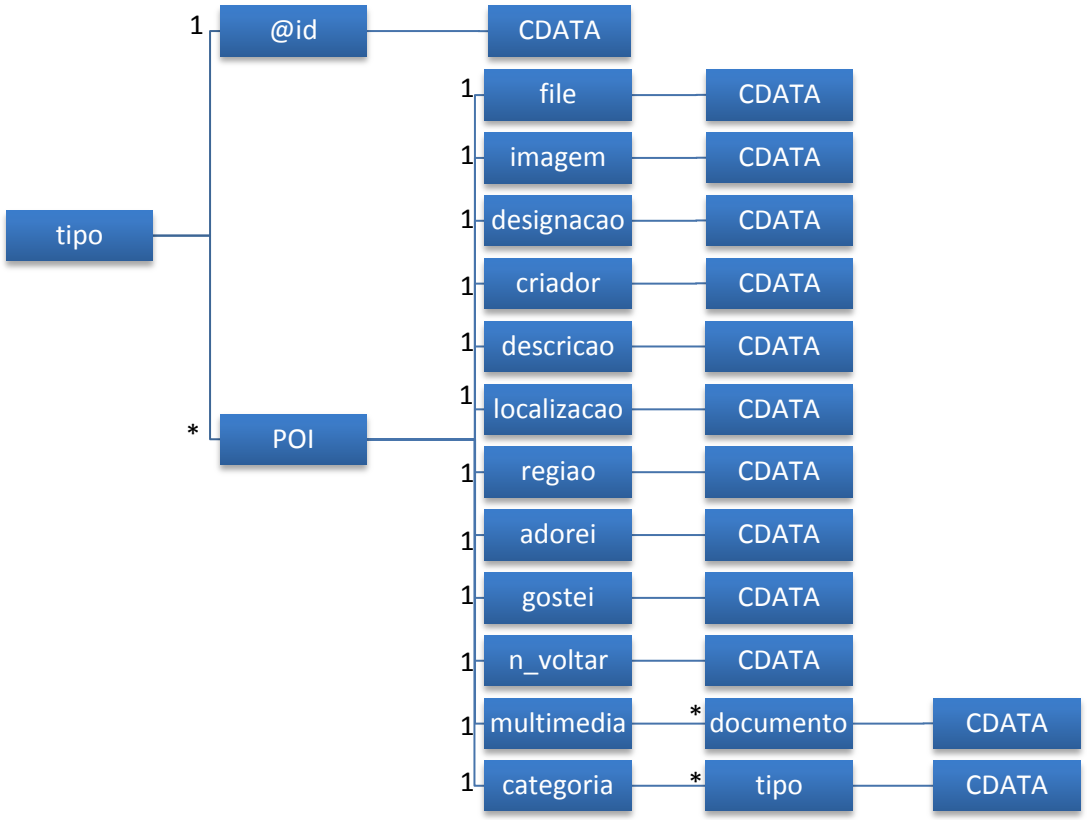

Figura 4 - Estrutura das mensagens de Listas de POIs

Repare-se que em ambas as estruturas o caminho a percorrer na árvore de nós de modo a obter o tipo de pedido ou resposta é o mesmo. Deste modo é possível obter o tipo de formulário de forma inequívoca de forma a proceder ao correcto tratamento dos dados.

Estrutura do *software*

Embora a maioria dos métodos tenham sido aproveitados do primeiro trabalho prático, esses foram reorganizados em classes de modo a facilitar lidar com a nova arquitectura. Criou-se ainda a classe *LeituraStringsXML*, responsável pela criação e leitura dos string trocadas entre clientes e servidor, assim como duas classes auxiliares que facilitam o manuseamento das estruturas de dados. A abordagem das diversas classes será efectuada adiante.

Acrescentou-se ainda o suporte para imagens de POIs, imagens essas submetidas pelo cliente e posteriormente armazenadas no servidor. Foi necessário também proceder à conversão de imagem para string, do lado do servidor, de modo a ser enviada no xml, assim como fazer a reconstrução da imagem do lado do servidor.

Apresentam-se agora os diagramas de classes do lado do cliente e servidor, Figura 5 - Diagrama de classes do lado do cliente e Figura 6 - Diagrama de classes do lado do servidorrespectivamente:

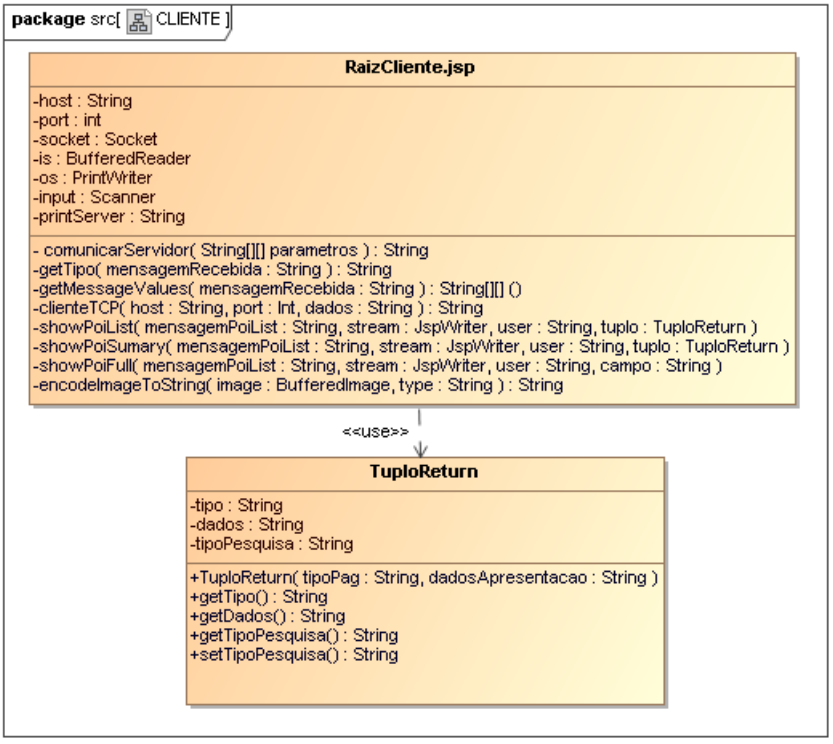

Figura 5 - Diagrama de classes do lado do cliente

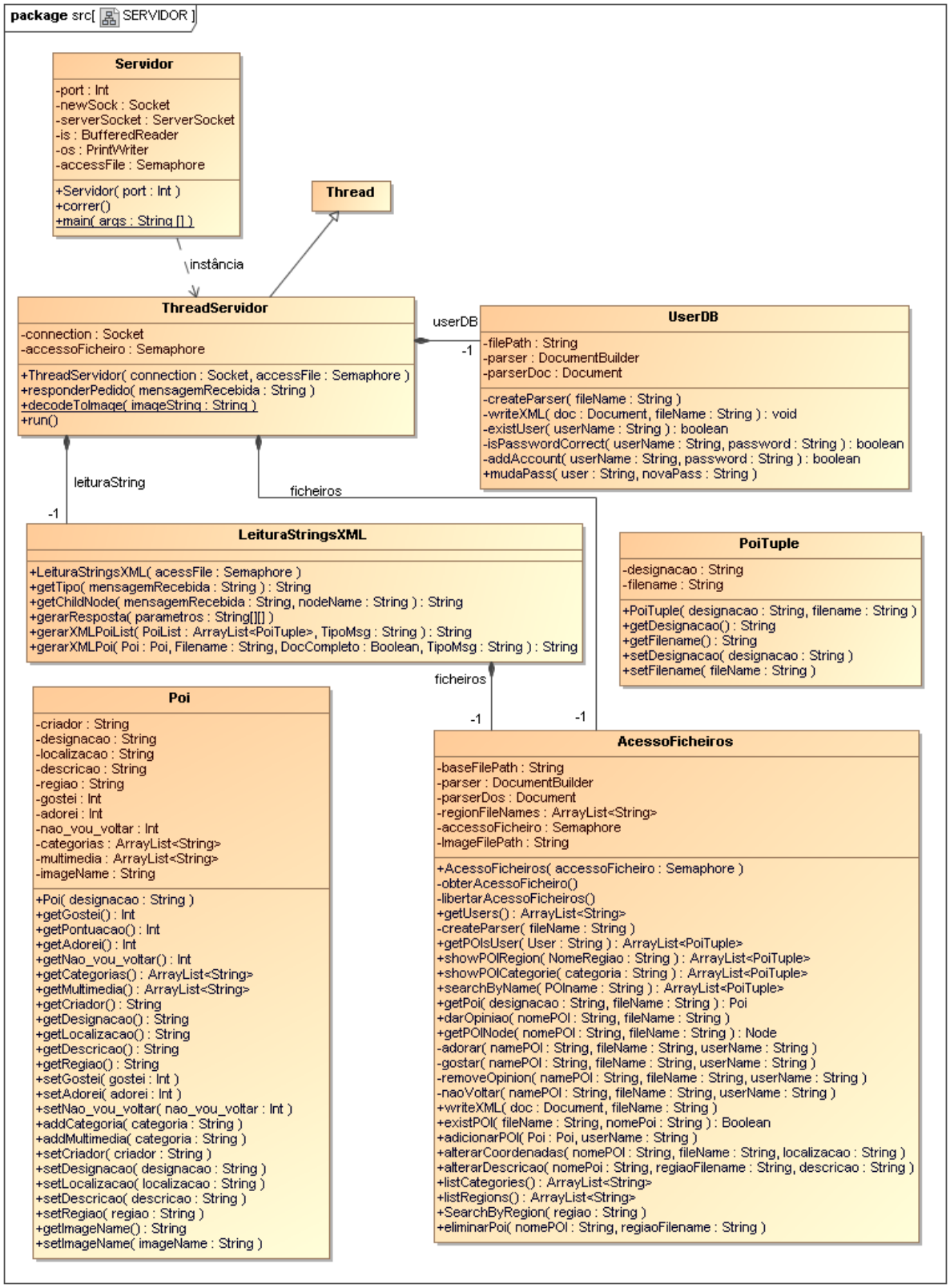

Figura 6 - Diagrama de classes do lado do servidor

Foi também criada uma Applet com a funcionalidade de alterar a palavra-chave do utilizador, cuja estrutura é demonstrada na Figura 7 - Diagrama de classes da Applet.

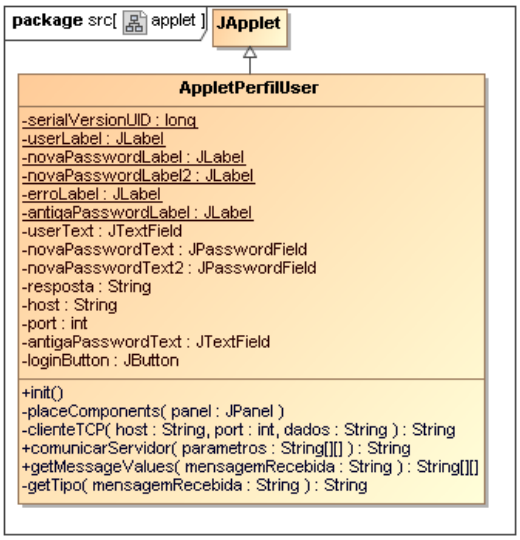

Figura 7 - Diagrama de classes da Applet

Apresenta-se ainda na Figura 8 – Diagrama ilustrativo do funcionamento do *software* um diagrama que permite simplificar a compreensão do fluxo de informação no sistema.

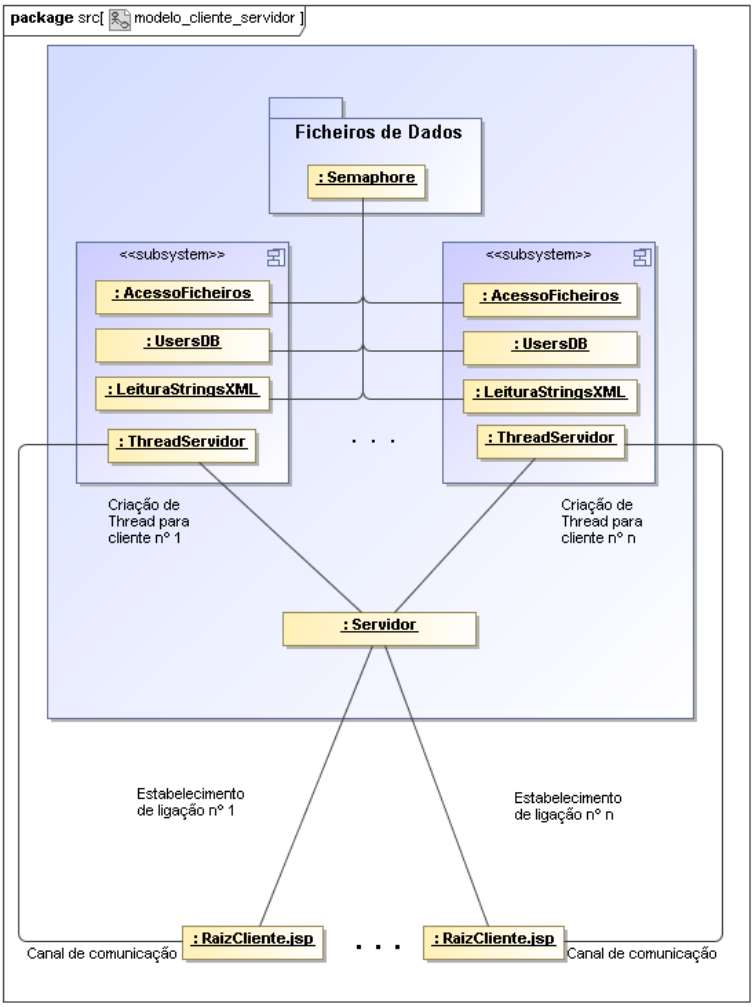

Figura 8 – Diagrama ilustrativo do funcionamento do *software*

Modelo Cliente-Servidor

Há semelhança do trabalho anterior o servidor é concorrente, instanciando-se uma *thread* para responder a cada pedido do utilizador. Contudo, a nova arquitectura distingue-se da anterior no funcionamento das *threads* que respondem aos pedidos. Enquanto no trabalho anterior a *thread* era instanciada aquando se estabelecia a ligação cliente-servidor e eliminada quando o cliente terminava a sessão, na nova abordagem a cada pedido do cliente é instanciada uma nova *thread* que responde apenas a essa solicitação. Sumariamente, a cada pedido o cliente estabelece ligação com o servidor, este instancia uma *thread* que responde ao pedido e para a sua execução após essa tarefa. Deste modo os clientes exercem uma muito menor carga computacional no servidor, dado que este não precisa de ter uma *thread* a correr por cada cliente logado nem tem que manter informação de estado do cliente.

Sendo uma aplicação a correr no *browser*, esta continua a ser *thin Client*, dado que toda a lógica do sistema de gestão de POIs e utilizadores está do lado do servidor. O cliente tem apenas as estruturas de dados e métodos necessários à descodificação e apresentação dos formulários recebidos do servidor. A aplicação cliente limita-se a direccionar pedidos do cliente efectuados através da nova interface gráfica e a apresentar as respostas a esses pedidos, fruto do processamento do servidor.

Há semelhança do trabalho anterior, a classe Servidor tem apenas como função esperar ligações de clientes, instanciar uma *thread ThreadServidor* e redireccionar para essa a ligação, que atenderá o pedido e enviará a resposta para o respectivo cliente.

Aplicação Servidora

Classes Auxiliares

Sendo que se mantiveram as estruturas dos dados guardados em ficheiros, os métodos de acesso aos mesmos preservaram também a sua estrutura. Recorreu-se apenas á reorganização do código e à alteração de alguns das estruturas de dados de parâmetros e ou retornos da funções implementadas, dado que agora os métodos não imprimem na consola mas passaram a devolver objectos com os dados que o cliente requisita.

Com a nova organização o código passou a dividir-se em três classes, sendo elas:

- A classe *AcessoFicheiros*, constituída pelos métodos que permitem o manuseamento dos ficheiros XML contendo POIs;
- A classe *UsersDB*, constituída pelos métodos que permitem o manuseamento dos ficheiros XML contendo as informações de utilizadores;
- A classe *LeituraStringXML*, com a qual se introduz um nível de abstracção no tratamento das *strings* trocadas entre servidor e cliente. A criação desta classe deu-se apenas neste trabalho pois no primeiro não ocorria troca de informação entre o cliente e o servidor em formato XML;

Criaram-se ainda as classes *Poi* e *PoiTuple* de modo a faz um melhor manuseamento dos Pois e respectivas informações no contexto do servidor.

As classes principais, *Servidor* e *ThreadServidor*, serão abordadas adiante.

Aplicação De Gestão de POI's

A aplicação principal encontra-se na classe Servidor, a qual instancia *threads* da classe ThreadServidor.

Tendo em conta as alterações já enunciadas relativamente ao primeiro trabalho, é óbvia a necessidade de reestruturar a aplicação do lado do servidor. Foi a classe relativa à *thread* que teve de assumir uma estrutura completamente diferente.

Note-se que se alterou apenas a forma como é feito cada pedido e cada resposta, sendo toda a lógica de processamento da informação de POIs idêntica à implementada anteriormente. Para tal, caso se pretenda conhecer em detalhe o modo de funcionamento de acesso aos dados e quais as funções do sistema à que consultar o relatório da trabalho anterior.

Sendo que por cada conexão é efectuado um pedido apenas, após receber a ligação da parte do Servidor e a redireccionar para uma nova instância de *ThreadServidor*, essa recebe o pedido, identificando o tipo de solicitação e tratando-o concordantemente com as expectativas do utilizador. Para tal, a *thread* é constituída por um *switch*, cuja variável de teste é uma String identificadora do estado. É da responsabilidade da *thread* enviar sempre uma resposta em formato XML ao cliente indicando o estado para o qual deve transitar, assim como a informação de resposta ao pedido. A execução da *thread* é parada após o envio da resposta ao cliente.

Aplicação Cliente

Sendo que a aplicação cliente apenas necessita construir uma string XML com os parâmetros dos formulários para enviar ao servidor e ler e interpretar as *strings XML* recebidas, assim como proceder à representação da informação do lado do *browser*, todas as funcionalidades foram compiladas numa única *Java Server Page*.

De modo semelhante ao tratamento dos dados do lado do servidor, a JSP referida tem como base a identificação do tipo de resposta recebida pelo servidor à pergunta efectuada e a partir dessa fazer a representação da página, recorrendo aos dados recebidos na mensagem. A priori, a JSP já sabe como tratar cada tipo de resposta do servidor, assim como contém a definição dos métodos necessários para lidar com as estruturas XML recebidas e a construir.

Imagens Demonstrativas

Figura 9 – Página de login

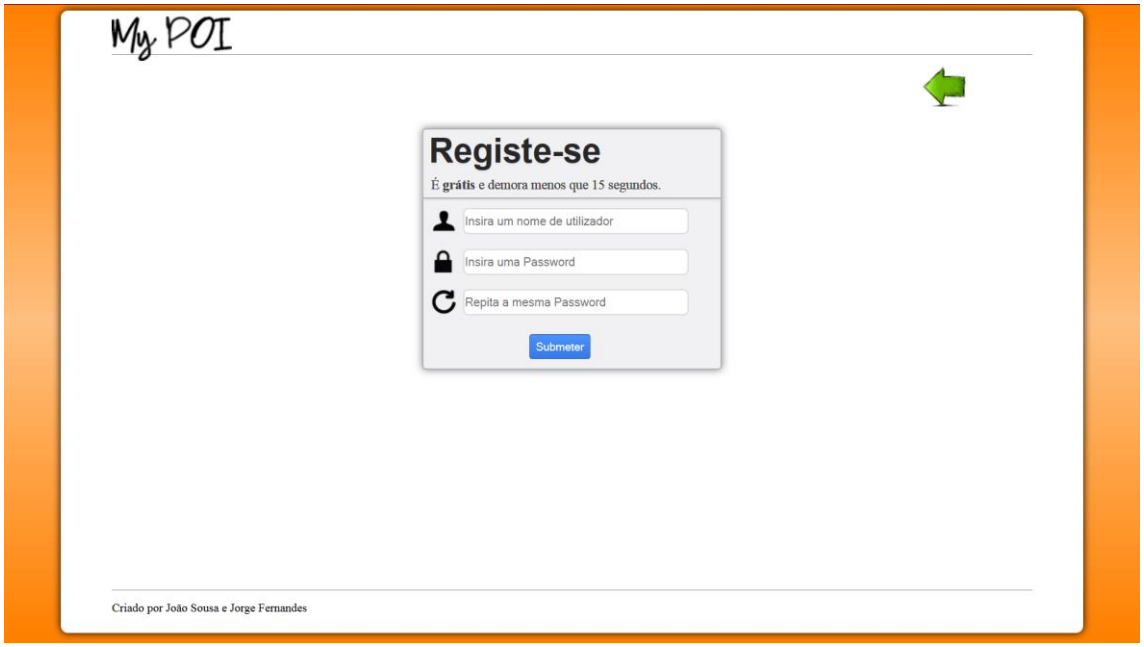

Figura 10 – Página de registo

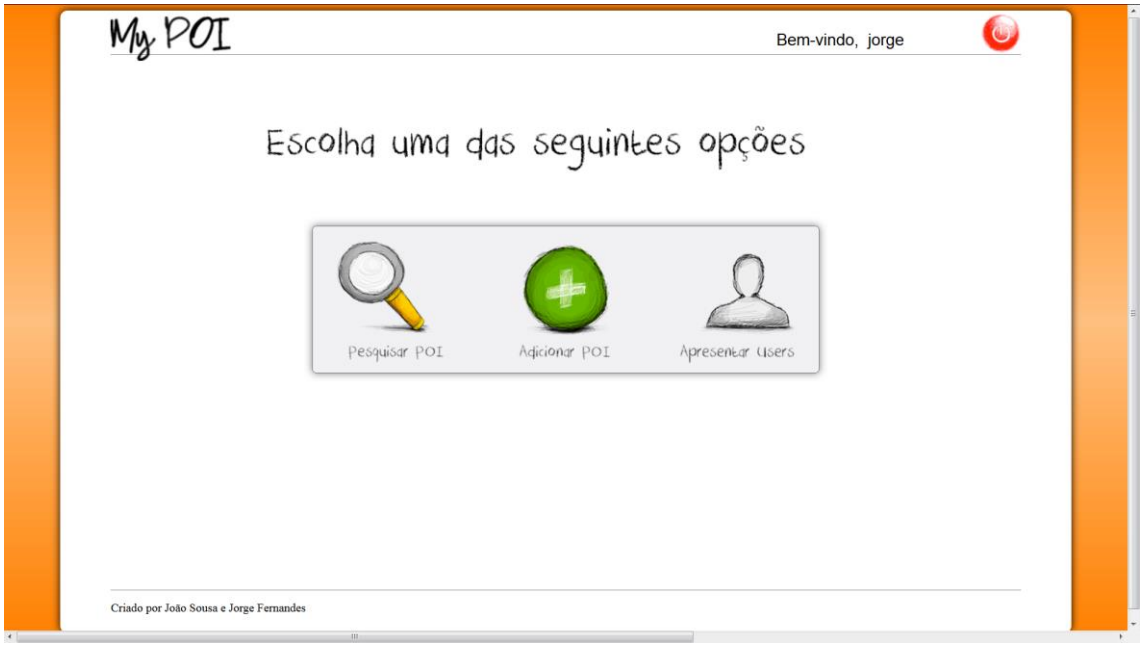

Figura 11 – Menu inicial

Figura 12 – Página de pesquisa de POIs

Figura 13 – Adição de POI

Figura 14 – Listagem dos utilizadores registados

Figura 15 –POIs de um utilizador (semelhante a qualquer listagem de POIs)

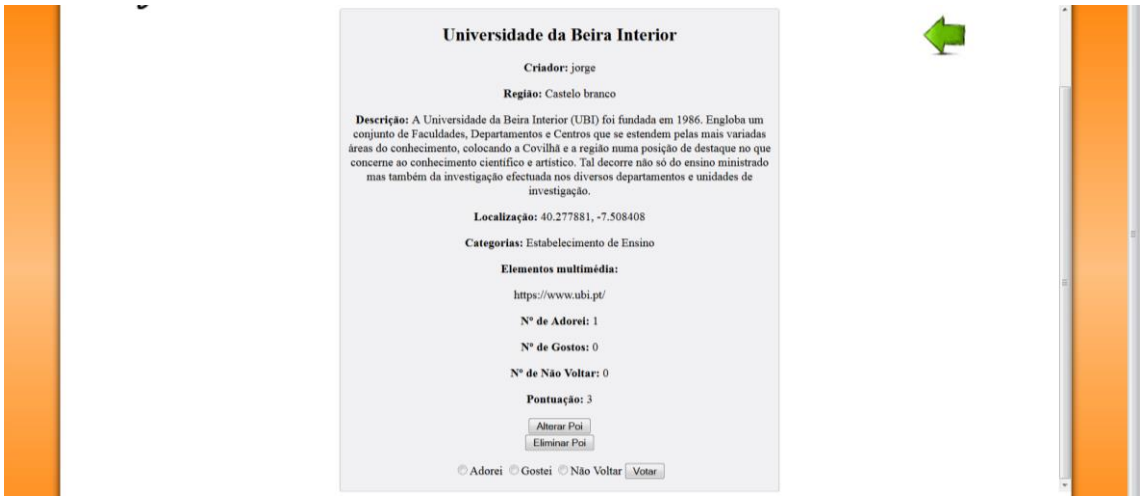

Figura 16 – Apresentação de um POI

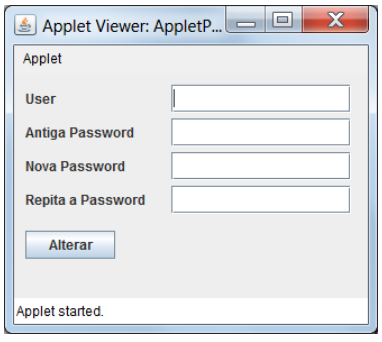

Figura 17 – Applet de alteração de palavra-chave

Conclusões

Com a execução abordada pudemos verificar que a execução do primeiro trabalho embora cumprisse com os objectivos propostos não seguia a metodologia estudada na Unidade Curricular. A estrutura do sistema anteriormente conseguido embora cumprisse os objectivos exercia uma carga computacional elevada no servidor, deixava-o responsável de lidar com o estado de todos os clientes o que aumentava a tolerância a falhas, para além de ter uma portabilidade muito fraca. Exemplo de tal é o facto de termos de repensar toda a estrutura de comunicação de modo a na nova iteração sobre o trabalho se poder utilizar o *browser* como interface.

Denotámos também os diversos benefícios em adoptar uma arquitectura na qual a comunicação se baseasse em pares pergunta-resposta através dos quais fosse possível deduzir a alteração entre estados. Deste modo, o servidor não necessita de ser bloqueante, não necessita de guardar informações adicionais para além dos registos da informação da plataforma, guardados em disco, e fica liberto da responsabilidade de cessar conexão convenientemente quando o cliente desejar, ou até de fazer a limpeza de variáveis relativas à sessão do utilizador. Com a nova arquitectura torna-se tudo muito mais simples. O servidor limita-se a receber uma ligação, responder a um pedido e cessar essa ligação.

Podemos ainda verificar o quão importante é a definição da estrutura dos dados transmitidos e armazenados, primando por um lado pela simplicidade da solução e por outro pela expansibilidade. No nosso caso, optámos pela definição de duas estruturas distintas precisamente pelo facto de lidarmos com dois tipos de formulários bastantes diferentes. Para que fosse possível abarcar ambas numa única estrutura XML, essa seria demasiado vaga, a qual complicaria o *parsing* do ficheiro. Assumindo a existência de duas estruturas XML distintas, pôde perfeitamente responder-se às exigências, sem contudo lidar com um grau de complexidade elevado. Foi apenas necessário ter o cuidado de conseguir um processamento idêntico sobre a árvore com vista à obtenção do tipo de mensagem e estrutura de ficheiro XML.

Relativamente ao trabalho, há ainda a apontar o facto de a estrutura de dados trocados entre clientes e servidor detendo listagens de parâmetros textuais sem hierarquia não terem sido bem conseguidas. Na tentativa de definir uma estrutura global, criamos uma lista de nós, filhos do nó raiz, cujos nomes e os valores são os mesmos da variável discriminada nesse nó. Denotámos posteriormente a má prática associada a esta definição, dado que não é possível estabelecer regras DTD ou XSD para uma estrutura deste género. Como alternativa aponta-se a possibilidade de manter a estrutura hierárquica de profundidade 1, contudo definindo-se um nome fixo para os nós filhos da raiz do documento, passando a indicar-se o tipo de parâmetro como atributo de cada nó. Assim, não se descarta a possibilidade de existirem infinidades de parâmetros distintos, tornando-se possível definir regras que validem as mensagens XML comutadas entre servidor e cliente.