

Integração de Sistemas

1º Semestre

**XML and XML Manipulation, Java Message Service and Message Oriented Middleware**



Trabalho realizado por:

* Daniel Coelho Bastos 2009109605
* Tiago Martins Andrade 2010165760

2015/2016

# Índice

[Índice 2](#_Toc432698085)

[Introdução 3](#_Toc432698086)

[Aplicações 4](#_Toc432698087)

[1. Web Crawler 4](#_Toc432698088)

[2. HTML Summary Creator 4](#_Toc432698089)

[3. Price Keeper 5](#_Toc432698090)

[3.1. Queue/Temporary Queue 5](#_Toc432698091)

[4. Cliente/Price Requester 5](#_Toc432698092)

[Testes e Conclusões 6](#_Toc432698093)

# Introdução

Este projecto tem como objetivo a aprendizagem e compreensão de competências para a implementação de um sistema de parsing e comunicação entre servidores para aceitar resquests de preços de smartphones dos sites <http://www.pixmania.pt> e <http://www.worten.pt>.

As linguagens de programação utilizadas são o Java, XML, XSD e XSL. Os IDE’s utilizados foram o IntelliJ e o Eclipse. Como Servidor de Aplicação foi utilizado o Wildfly que fornece a API, o modelo de arquitetura e o ambiente de execução definidos na especificação JAVA EE que inclui o container web e diversos outros subsistemas Java EE.

O sistema divide-se em 4 partes:

* [Web Crawler](#_Web_Crawler)
* [HTML Summary Creator](#_HTML_Summary_Creator)
* [Price Keeper](#_Price_Keeper)
* [Price Requester](#_Clients/Requesters)

A informação e preço dos smartphones é obtida pelo WebCrawler e enviada para o Price Keeper através do Java Message Service Wildfly. O Price Keeper faz a gestão entre a Queue dos requests feitos pelos Clientes/Price Requesters e as Temporary Queues criadas para lhes enviar as respostas. Existe ainda o HTML Summary Creator que cria uma página HTML para apresentar a informação obtida pelo WebCrawler.

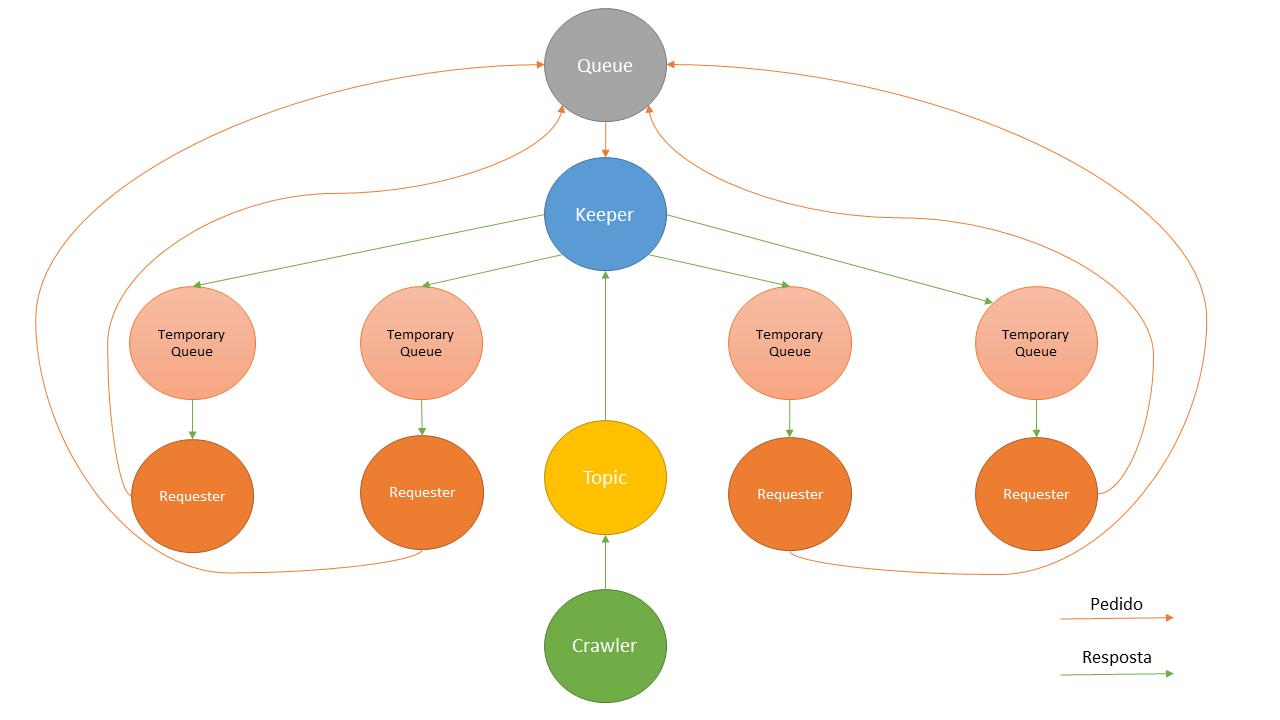


Figura 1 - Organização das Aplicações

# Aplicações

## Web Crawler

O Web Crawler é a aplicação que faz o parsing de um website e envia a informação para o servidor Wildfly. Utilizámos a biblioteca Jsoup para fazer o parsing das várias páginas HTML do site.

Para a eventualidade de os websites estarem offline fizemos uma cópia dos mesmos localmente através da Extensão para Firefox *Scrapbook*. Assim que acaba o parsing do site este faz Marshal à informação (Java Object -> xml) para esta ser enviada para o JMS (PlayTopic).

## HTML Summary Creator

A aplicação HTML Summary Creator (*MainHtmlCreator.java*) recebe o xml e gera uma página HTML (*full\_smartphones\_list.html*).

Para validação do XML criado adicionámos o ficheiro *smartphones.xsd*. Usámos o site <http://www.freeformatter.com/xsd-generator.html> para gerar o xsd correspondente ao xml correto.

Ao iniciar primeiro valida o xml recebido através do xsd. De seguida, caso a validação seja positiva, transforma-se o xml usando a stylesheet xsl (*xsl\_stylesheet.xsl*) numa página HTML.

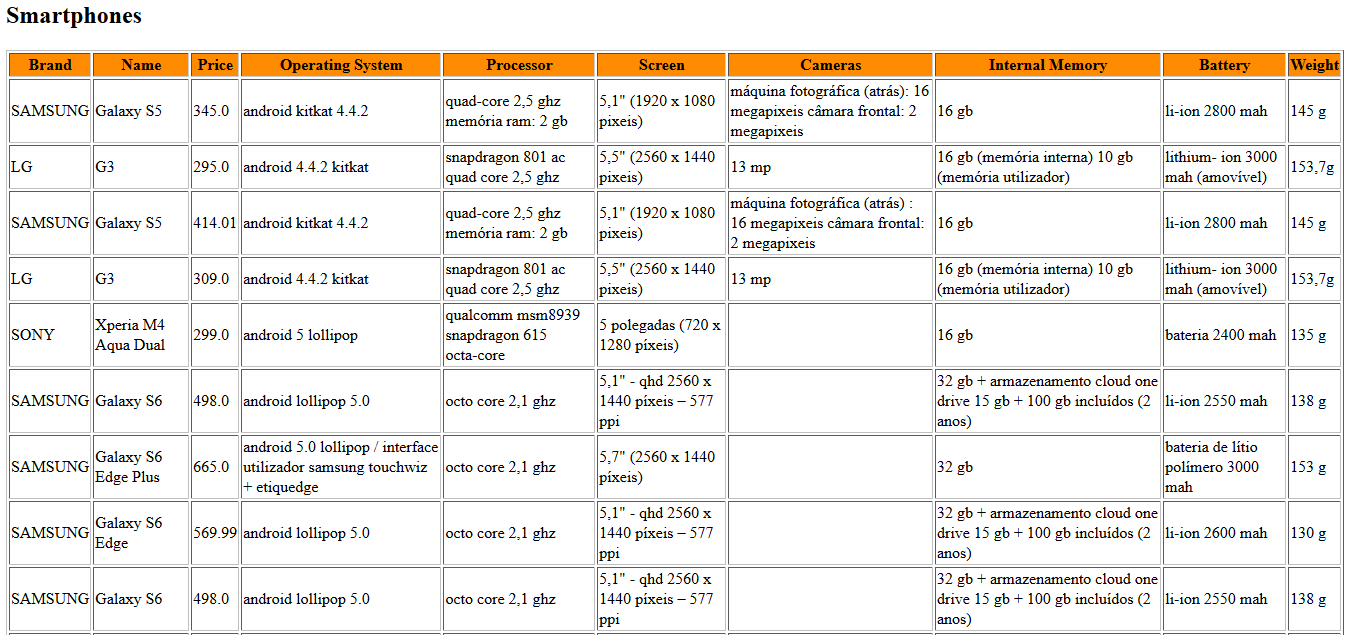


Figura 2 - Página HTML gerada pelo HTML Summary Creator

## Price Keeper

Price Keeper é a aplicação que mantém em memória os preços dos smartphones e comunica com o WebCrawler e os Clientes/Price Requesters.

Ao iniciar o MainKeeper põe a correr o Father Keeper, que por sua vez inicia dois processos filho (*TopicListener.java*, *QueueListener.java*). O FatherKeeper mantém uma lista atualizada dos smartphones e fica à espera de ações dos seus filhos.

### Queue/Temporary Queue

TopicListener é a aplicação que comunica com o JMS (PlayTopic), ficando à escuta de mensagens vindas do WebCrawler. Ao receber o xml do WebCrawler este faz unmarshal (XML -> Java Object) e envia para o FatherKeeper que guarda a informação em memória.

QueueListener é a aplicação que comunica com o JMS (PlayQueue) para receber as mensagens/requests dos Clientes. Uma vez que recebe a mensagem cria uma *Thead* (*Responder.java*) para resposta onde se liga à Temporary Queue do Cliente em escuta e envia-lhe a resposta (preço do smartphone procurado).

## Cliente/Price Requester

A aplicação Price Requester (*MainRequester.java*) implementa uma pesquisa pelo nome de um smartphone com o objetivo de saber qual o seu preço.

Ao iniciar a aplicação o utilizador introduz o nome do smartphone que quer saber o preço. Após a receção do pedido a aplicação cria uma Temporary Queue para ficar à escuta de uma resposta.

Entretanto liga-se ao JMS (PlayQueue) e envia a pesquisa com a informação de qual é a sua Temporary Queue para receber a resposta. Ao receber a resposta termina a Temporary Queue dedicada a essa pesquisa e fica à espera de novo pedido do utilizador.

# Testes e Conclusões

Para avaliar o funcionamento das várias aplicações fizemos uma série de testes.

**Webcrawler:**

* Se a conexão com o website falhar utilizar a cópia local do site;
* Se a conexão com o servidor Wildfly falhar criar o ficheiro xml e guardar lá a informação;
* Se ao iniciar existir um ficheiro xml guardado em sistema enviar esse ficheiro em vez de fazer o parsing e de seguida apagá-lo.

**HTML Summary Creator:**

* Receber a mensagem (xml);
* Validar o xml através do schema (xsd);
* Criar o ficheiro HTML através do xsl com as informações recebidas no xml.

**Price Keeper:**

* Criação das Threads e comunicação entre elas;
* Comunicação entre as Theads e o JMS (PlayTopic e PlayQueue);
* Aceder ao preço correto do smartphone mediante a pesquisa recebida;
* Manter atualizada a informação recebida do WebCrawler.

**Cliente/Requester:**

* Fazer pesquisa por um termo;
* Fazer pesquisa por mais do que um termo;
* Receber o preço correto em relação ao smartphone pesquisado.

Ao realizar este projecto ficámos com conhecimentos sólidos sobre a edição e manipulação de documentos xml. Também nos deu a oportunidade para explorar o serviço de comunicação de mensagens Wildfly.