Sistemas Distribuídos - 2ª Parte do Projecto

Grupo: 44

Alunos: Duarte Pompeu, 73008

Requisito SD-STORE-A

Gestão de Réplicas

O servidor **SDStoreMain** regista-se no JUDDI com o nome *SD-STORE-X*, sendo X um inteiro de 0 a *n*. Devido a falhas ocasionais ao remover o registo no UDDI, implementei no sd-store um método de remover todos os nomes do UDDI com o nome **SD-STORE-X** (`mvn exec:java -Dexec.args="kill"`).

No cliente, a classe **ClientFrontEnd** faz uma pesquisa com *wildcard* no UDDI por ***SD-STORE-%***, inicializando um número de **StoreClients** consoante o numero de *endpoints* encontrados. Ou seja, o *frontend* gere vários clientes virtuais, e cada cliente comunica com um servidor. Isto é interessante pois mantém a relação de 1 cliente - 1 servidor, tal como na 1ª entrega. Esta funcionalidade individual é testada em **WebServiceRemoteTestIT**.

A ligação a várias réplicas é testada através em **FrontEndRemoteTestIT**. Por razões desconhecidas, este teste pode falhar na bateria de testes completa, mas tem sucesso quando executado individualmente (`mvn -Dtest=FrontEndRemoteTestIT test`).

Quorum Consensus / Handler com Tag

Cada réplica mantém registo de nº de versões. Há dois tipos de versões: a versão do **documento** e a versão do **repositório**, que é a soma das versões dos documentos, necessária para o pedido de listDocs. Também há a noção de ID do frontend que fez última escrita. Tal como com as versões, há um ID por **documento** e um por **repositório**, igual ao do ID último documento a ser editado.

O servidor responde a pedidos do cliente e escreve no header do SOAP a tag respectiva. O cliente utiliza esta tag para fazer um quorum segundo o protocolo ensinado. Para este efeito, são usadas as classes **QuorumFactory, Quorum, Response e Tag**. A fábrica de quorums permite editar os thresholds **RT** e **WT** dinamicamente, sendo estes reflectidos nos quorums obtidos através dos métodos *getWriteQuorum()* e *getReadQuorum()*.

Após a operação *loadDoc*, o frontend faz **writeback** quando necessário. Para isso, guarda todas as tags obtidas tal como o cliente/servidor que gerou o resultado, obtém a de maior peso e o conteúdo do documento associado à tag, e executa a operação *store* em todos os clientes/servidores com tags de menor peso.

Esta componente foi algo problemática pois não tinha percebido bem o protocolo inicialmente. Pensava que o frontend inquiria todas as réplicas e escolhia a resposta mais comum (uma espécie de votação). Este protocolo foi corrigido, mas mantive algumas partes antigas para referência/histórico, apesar de não estarem a ser usadas de todo.

Considerações

Por falta de tempo, não implementei mensagens assíncronas.

Requisito SD-ID-B

Confidencialidade

Nas operações *store* e *load*, o cliente gera uma chave com base no ID de utilizadores que recebe. A partir desta chave, cifra um documento antes de enviar, e decifra depois de o rebecer.

O servidor não precisa de saber o significado do conteúdo do documento, pelo que não tenta (nem sequer consegue) decifrar estes dados. Isto é interessante do ponto de privacidade, pois garante ao cliente que não há acessos indevidos à informação alojada remotamente

Integridade

Existe uma chave *hardcoded* no cliente e servidor, usada para gerar e verificar MACs. Quando o cliente executa a operação *store*, gera um MAC com base na chave e no resumo do documento cifrado, passando-o ao servidor no header da mensagem SOAP.

O servidor recebe o documento cifrado e o MAC. Sabendo a chave, compara o resumo do MAC decifrado e do resumo do documento. Caso não seja igual, pode concluir que houve uma falha de integridade desde o cliente até ao servidor.

Considerações

O plano era começar com soluções simples e fazer melhorias incrementais para obter soluções mais flexíveis e eficazes. Tal não foi possível, mas estas são algumas melhorias planeadas.

Confidencialidade:

1. Obter resumo da senha do utilizador a partir do ID recebido
2. Cifrar e decifrar documentos segundo o resumo (em vez do ID)
3. Desenvolver solução que contemplasse mudanças de senha

Integridade:

1. Melhorar geração e partilha de chave MAC
2. Integrar MAC com outros serviços (*create*, *list*, *load)*
3. Utilizar MAC nas mensagens Servidor → Cliente