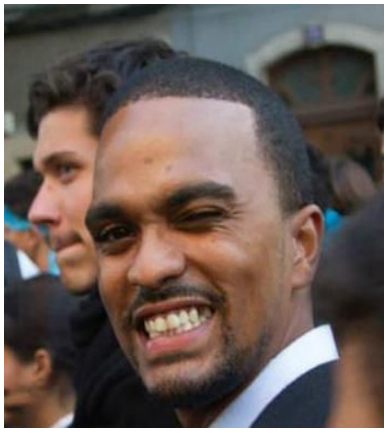
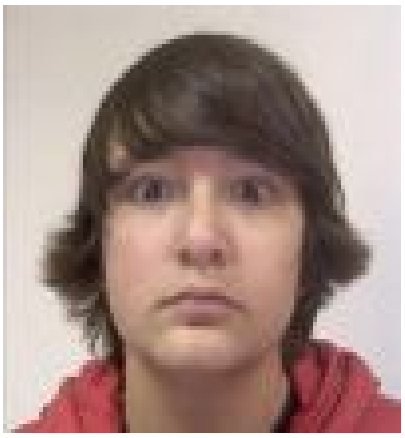
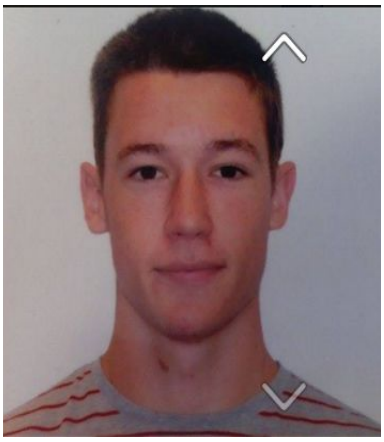
Relatório Final do Projecto:
Upa Transportes

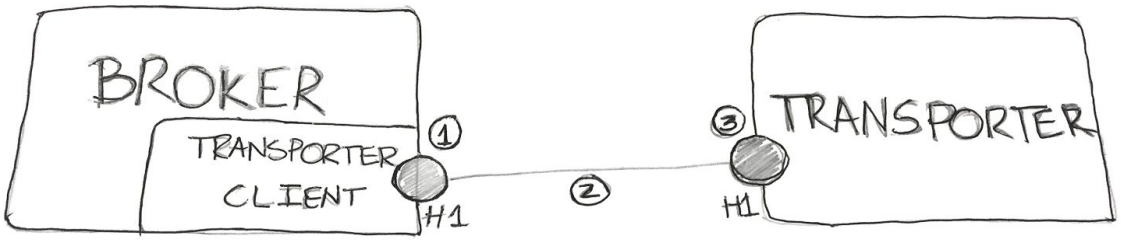

77928 - Frederico Teixeira


78015 - David Silva


78052 - Miguel Carvalho

Grupo T27
GIT URL: https://github.com/tecnico-distsys/T_27-project

Segurança



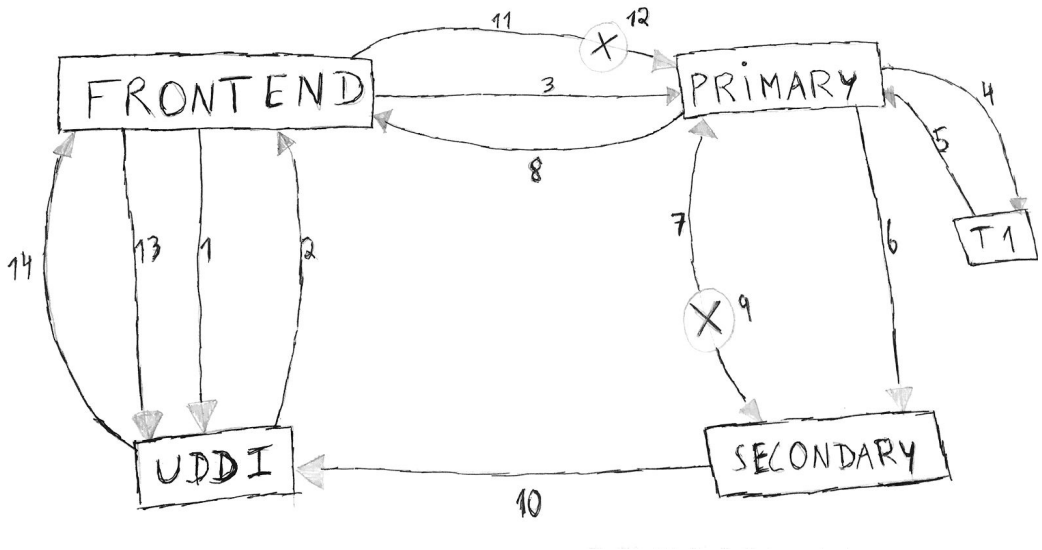
H1 - Handler-Chain: LoggingHandler->AuthenticationHandler->LoggingHandler

Descrição:

- 1- Broker envia mensagem para um Transporter, após esta passar pela Handler-Chain H1 (handleOutboundMessage);
- 2- Mensagem assinada passa pela rede;
- 3-Transporter recebe a mensagem enviada pelo Broker, após esta passar pela Handler-Chain H1 (handleInboundMessage);

Racional: Nesta parte pretendemos garantir alguns requisitos não-funcionais: frescura, autenticidade e não-repúdio.
Com isto em mente desenvolvemos um AuthenticationHandler pelo qual passam todas as mensagens SOAP que entram ou saem do Broker e dos Transporters. Para além do Body da mensagem original acrescentámos três elementos ao Header, <Freshness>,<SenderName> e <Message Signature>. No primeiro temos um identificador único e um timestamp. Utilizamos o segundo para permitir ao receptor (inbound) aceder ao certificado emitido pela CA, e verificar se este é válido. O terceiro contém o resultado do Hash da mensagem original, Hash(M), que depois é assinada com a chave privada do remetente antes de ser enviada, {Hash(M)}PrivateKeySender. Do lado do recetor, verificamos se a mensagem não foi alterada, comparando o Hash recebido com o realizado no receptor, garantindo a autenticidade e o não-repúdio. Analisamos também o identificador e o timestamp recebidos, de acordo com regras definidas por nós, garantindo a frescura.

Tolerância a Faltas



Descrição:1,2-Conhecer localização do Broker perguntando ao UDDI
3,11- Pedido ao BrokerPrimary
4,5-Pedido e resposta a Transportadora
6-Actualizar estado no Secundário
7-Proof of Life do Primário
8-Resposta para FrontEnd
9-Secundário apercebe-se da morte do Primário
10-Secundário publica-se no UDDI
12-Front End não consegue alcançar Primário
13,14-Conhecer localização do novo Primário

Racional: Para garantirmos a tolerância a faltas do Broker utilizamos uma abordagem de Replicação primário-secundário, ou seja, criamos um servidor secundário para além do que já existe. Este é uma réplica cujo estado deve ser alterado pelo primário para manter-se actualizado e coerente. Caso o servidor primário falhe, o secundário deve detectar essa mesma falha e assumir o seu papel. Optámos por colocar o Secundário a verificar se o primário está vivo através do uso de pings. Este raciocínio é válido pois a comunicação entre os dois é por HTTP (TCP), e este protocolo garante uma comunicação bidireccional, logo prova de vida de A->B é equivalente a B->A.