TÉCNICO LISBOA

Relatório de Projetode

Engenharia de Software

&

Sistemas Distribuídos

Última Entrega

Grupo A_04_17_31

Grupo A_04_17_31

Tiago Sampaio nº 72648 (Grupo 17 SD)

Francisco Peixeiro nº 72745 (Grupo 17 SD)

Paulo Martins nº 72976 (Grupo 17 SD)

Fábio Antunes nº 66979 (Grupo 31 SD)

Miguel Deus nº 72879 (Grupo 31 SD)

Rodrigo Monteiro nº 73701 (Grupo 31 SD)

Índice

Cheque Refeição (Segurança)

Grupo 17

Introdução

Nesta segunda entrega foi-nos pedido para conceber um sistema de segurança para a utilização de cheques, através de uma política de segurança que assegure que o sistema de cheques é resistente a ataques de clientes mal-intencionados. Nesta segunda fase, o número de cheque pode ser visto como uma sequência de caracteres que inclui um identificador público e um segredo. O conhecimento do número de um cheque (identificador público + segredo) constitui uma capacidade que dá a quem conhece a permissão de usar o cheque armazenado no servidor.

Quando um utilizador faz um pagamento usando um cheque, o número de cheque deve ser fornecido à entidade que beneficiará do cheque. Usando esse número, a entidade benificiária poderá depois sacar o cheque.

Por outro lado, operações como "listar", que não se pretende que concedam permissões de uso a que as invoca, devem devolver apenas o identificador público de cada cheque.

Opções de desenho e implementação

Para conseguir implementar as funcionalidades pedidas nesta segunda fase do projeto, optámos por introduzir o conceito de criptografia que nos permite cifrar e decifrar as mensagens, utilizando Handlers para conseguir interceta-las.

Existem dois tipos de cifras, as simétricas e as assimétricas. As cifras simétricas têm como objetivo confundir, ou seja, operações não destrutivas que permitam alterar o significado da mensagem em aberto misturando-o com a chave, e difundir, isto é, fazer com que as alterações se difundam a toda a mensagem cifrada para não ser alvo de análise estatística de padrões. A utilização destas duas cifras deve-se ao facto da chave perder dados se utilizarmos só a cifra assimétrica, garantindo assim que todo o conteúdo é transmitido.

A comunicação entre cliente e servidor é feita cifrando a mensagem/pedido no cliente e decifrada pelo servidor, de seguida esse mesmo servidor cifra a mensagem/resposta e envia para o cliente que vai decifrar essa mesma mensagem/resposta. Para garantir a confidencialidade temos as diretorias "keys" tanto no cliente como no servidor para garantir que ao gerar as chaves ambos consigam aceder às chaves que necessitam, tanto para encriptar como para decifrar. Como está no enunciado o servidor tem a sua chave privada e a pública de todos os clientes e o cliente vice-versa.

Requisitos implementados de forma insatisfatória / não implementados

Para esta entrega, os requisitos implementados corresponde à parte do Protocolo Base com todas as funcionalidades que lhes estão associadas, respeitando a segurança relativa à parte do Cheque Refeição. No que diz respeito ao Endosso, não nos foi possível implementar essa funcionalidade no nosso projeto, como era inicialmente previsto, devido à falta de tempo com o excesso de trabalho relativo a outras cadeiras com projetos e testes, ficando a funcionar a segurança base e essencial para o bom funcionamento do sistema.

O requisito em falta consiste na delegação da capacidade de utilização do cheque do Titular para um terceiro, chamado Endosso e na criação de um novo segredo sem revelar o antigo. Para concretizar o requisito em falta, seria necessário criar uma função criptográfica de hash para gerar um novo segredo a partir do antigo. Para garantir que o titular do cheque esteja envolvido na operação de Endosso, seria necessário associar um novo segredo à assinatura do segredo original utilizando a chave privada do titular.

No que diz respeito aos testes, como só implementámos o protocolo base, apenas se baseiam na funcionalidade de emitir e sacar. O emitir é testado automaticamente através de testes criados pelo grupo, por outro lado o sacar só é testado manualmente, devido à alteração provocada com a implementação do segredo, isto é, teríamos que concatenar esse mesmo segredo com o ID do cheque, requisito este que não nos foi possível concretizar devido às razões acima explicadas.

Registo Fatura (Replicação)

Grupo 31

Introdução

Para garantir uma melhor disponibilidade e escalabilidade foi-nos proposta a introdução de replicação (passiva) nesta última entrega do projeto. Os clientes interagem com o servidor principal e o servidor de backup torna-se o primário quando deteta que o primário falhou. Com a implementação deste conceito garantimos a existência de cópias dos dados em vários sítios, precavendo a existência de eventuais falhas.

Utilizámos ainda o conceito de transparência de replicação, promovendo a ilusão de que o utilizador está a aceder a um único objeto lógico, não se apercebendo que na realidade existem várias cópias físicas dos seus dados.

Adotámos o mecanismo de linearizibilidade, havendo uma serialização virtual que é correta segundo a especificação dos objetos, respeita o tempo em que as operações foram invocadas e a consistência entre as execuções observadas por cada cliente, i.e. os valores retornados e estado final são os mesmos.

Opções de desenho e implementação

Nesta 2ª entrega de Sistemas Distribuídos optámos por criar suporte para uma réplica de servidor que vê a sua base de dados e parâmetros definidos no ficheiro *build.xml*usando uma porta diferente daquela usada pelo servidor principal/primário para se publicar e registar no serviço de naming UDDI.

Quando a aplicação é lançada são criadas e populadasambas as bases de dados e lançados ambos os servidores, sendo que apenas o servidor primário fica registado no UDDI (inicialmente). Este envia periodicamente (com um período de 4 segundos), mensagens *I'malive*para o servidor de suporte (secundário) que, por sua vez interpreta as mensagens recebidas, analisando o tempo de intervalo entre cada uma delas, sendo que se este ficar 5 segundos sem receber uma das mensagens supracitadas, interpreta isso como falha do servidor primário e regista-se no UDDI, assumindo ele o papel de servidor primário.

A troca de mensagens entre os servidores primário (registado no UDDI) e de backup é conseguida utilizando a estrutura de *sockets,* estando um associado a uma *TaskTime*para o envio da mensagem *I'm alive* a cada 4 segundos e o segundo que utiliza um *timeout*de 5 segundos para controlar o tempo de resposta do primeiro e aferir o seu estado como cativo ou não.

Enquanto o servidor primário se mantiver ativo, garantimos a consistência dos dados nas bases de dados de todos os servidores (primário e de backup), propagando as alterações feitas na base de dados do servidor primário para a do servidor secundário através da utilização de uma interface de comunicação entre os dois servidores, previamente estabelecida.

Requisitos implementados de forma insatisfatória / não implementados

Os requisitos que nos propusemos a fazer para esta entrega foram quase todos implementados satisfatoriamente, a nosso ver, sendo que, por falta de tempo, não conseguimos por em prática uma solução com *N*servidores de backup, como era inicialmente nosso desejo, ficando assim apenas um servidor para esse efeito, o que torna a solução mais simples mas talvez menos robusta e não tão elegante, do nosso ponto de vista.

Para implementar a solução descrita acima seria necessário implementar uma lógica de multi*sockets*para garantir a comunicação entre o servidor primário e os *N* servidores de reserva (backup), sendo necessária a criação prévia de várias bases de dados e utilizar um sistema de prioridades para a comutação de servidores que determina aquando da falha do servidor primário qual dos servidores de backup assume esse papel.

Também não nos foi possível, pelos motivos já mencionados, a criação de testes para o projeto, pelo que todos os testes que fizemos para aferir a qualidade do projeto foram realizados manualmente, passo a passo.