**‘Capítulo 7. Serviços Web em Ação**

Nos capítulos anteriores, construímos um panorama das tecnologias e metodologias em torno dos serviços web SOAP. Neste capítulo, aplicamos a discussão a uma implementação real de um serviço web SOAP. Você verá como SOAP e WSDL são implantados e também como integrar outras tecnologias XML para resolver problemas que SOAP e WSDL não abordam.

O serviço que desenvolveremos é o **CodeShare Service Network**, um conjunto simples de serviços web ponto a ponto para compartilhamento de código-fonte de aplicativos. Enquanto desenvolvemos esse código, faremos uma pausa para analisar segurança e como implementá-la quando SOAP e WSDL não a cobrem.

A implementação do CodeShare mostrada aqui fornece um meio para que pessoas compartilhem código-fonte. Usamos assinaturas digitais para verificar a identidade dos clientes e mantemos um registro central dos arquivos que as pessoas estão oferecendo. Em vez de um único serviço web, o aplicativo CodeShare é composto por várias pequenas interfaces diferentes — um design comum de serviços web. Cada interface pode ser implementada em qualquer linguagem que suporte SOAP, e usamos uma mistura de Perl e Java para demonstrar isso. O CodeShare é um exemplo de serviço web entre pares (peer web service). No modelo ponto a ponto (P2P), a Internet não é vista como uma rede de clientes acessando recursos de um servidor, mas sim como uma rede cooperativa de pares compartilhando recursos igualmente. As linhas entre o provedor e o consumidor do serviço se confundem, sem que seja exigido de um aplicativo apenas um único papel.

Serviços web entre pares usam tecnologias de serviços web já implantadas para fornecer serviços P2P.

**7.1 A Rede de Serviços CodeShare**

A **CodeShare Service Network** é um exemplo muito simples de serviços web entre pares. Ela fornece um ambiente onde desenvolvedores podem facilmente compartilhar código-fonte com o restante do mundo.

**7.1.1 Visão geral**

Existem três componentes importantes no CodeShare:

* o **proprietário** do código sendo compartilhado,
* o **solicitante** do código,
* e o **servidor CodeShare**, que atua como central de distribuição do código e como autoridade de autenticação que os proprietários de código podem usar para controlar o acesso ao código que estão compartilhando.

As relações entre os componentes são mostradas na Figura 7-1.

**Figura 7-1.** A arquitetura do CodeShare

**Cenário típico de uso:**

1. Os desenvolvedores de algum código decidem compartilhá-lo publicamente. Para isso, atualizam o arquivo local index.xml do projeto, indicando os arquivos que desejam compartilhar.
2. Os desenvolvedores conectam-se ao servidor CodeShare para atualizar sua entrada no índice mestre mantido no servidor.
3. Os desenvolvedores iniciam seu serviço proprietário CodeShare (um daemon local SOAP HTTP).
4. Quando usuários desejam encontrar código sendo compartilhado, eles têm duas opções:
   * conectar-se diretamente ao serviço proprietário e executar quatro operações básicas: **search**, **list**, **info** e **get**;
   * ou conectar-se ao servidor CodeShare e pesquisar no índice mestre. Isso resultará em uma lista de todos os serviços proprietários CodeShare compartilhando código que corresponda à pesquisa. Todas as operações **get** apontam diretamente para o serviço proprietário para recuperar o código-fonte compartilhado.
5. Às vezes, os desenvolvedores podem querer restringir quem pode acessar o código que estão compartilhando. Para isso, basta adicionar os nomes de todos os usuários autorizados ao seu index.xml (todos os usuários são registrados no servidor CodeShare). Sempre que um usuário tentar recuperar código restrito, o serviço proprietário verificará primeiro se o usuário fez login no servidor CodeShare e, em caso afirmativo, se tem permissão para acesso.

**7.1.2 Pré-requisitos**

Há algumas coisas que você precisa configurar no seu sistema antes de poder executar este exemplo:

**SOAP::Lite Versão 5.1 e todos os pré-requisitos**  
As instruções para instalar estão no Capítulo 3.

**DBI e DBD:CSV**  
Estes são módulos de banco de dados SQL em Perl usados pelo servidor proprietário do CodeShare. Instale-os digitando install DBI e install DBD::CSV no shell do CPAN.

**Um servidor web habilitado para Servlets**  
Recomendamos o Jakarta Tomcat da Apache, Versão 3.22. O Tomcat pode ser baixado de <http://jakarta.apache.org/>.

**Apache Xerces 1.4 ou qualquer outro parser XML habilitado para JAXP**  
O JAXP é a API Java para Processamento de XML (<http://xml.apache.org/xerces-j>).

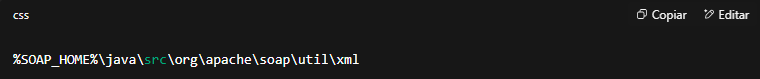
**Apache SOAP**  
No momento da redação, a última versão era a 2.2, que possui um bug que você precisará corrigir. Baixe a distribuição de código-fonte do Apache SOAP. As alterações e o processo de compilação são descritos na próxima seção deste capítulo.

**A versão mais recente do IBM XML Security Suite**  
Está disponível no site alphaWorks da IBM (<http://alphaworks.ibm.com/tech/xmlsecuritysuite>).  
Após o download, descompacte a distribuição e coloque o arquivo XSS4J.jar no classpath do seu servidor de aplicações.

**7.1.2.1 Corrigindo o bug no Apache SOAP 2.2**

O problema na Versão 2.2 do Apache SOAP causa a produção de XML inválido em algumas situações. O CodeShare acaba gerando uma dessas situações. A correção detalhada aqui foi enviada e não deve ser necessária em versões posteriores à 2.2.

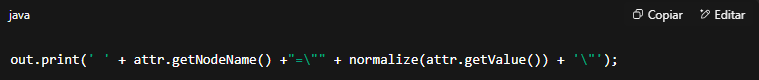
Assumindo que você já baixou a distribuição de código-fonte do Apache SOAP, localize o arquivo chamado **DOM2Writer.java** na pasta:



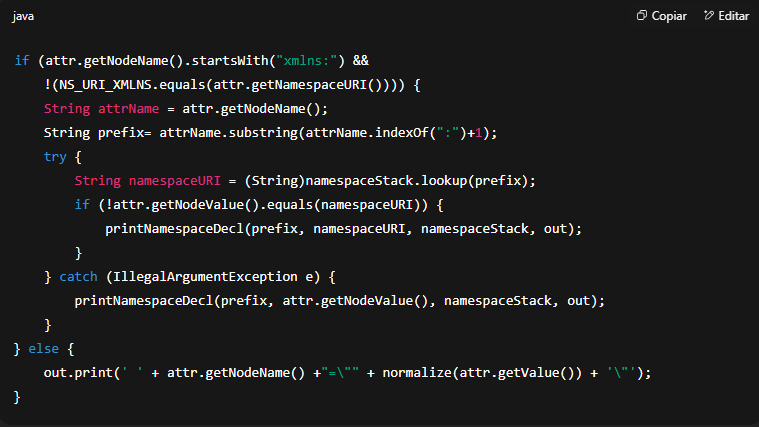
(%SOAP\_HOME% será o diretório onde o conteúdo descompactado da distribuição foi baixado).

Na **linha 172**, substitua as linhas do **Exemplo 7-1** pelas do **Exemplo 7-2**.

**Exemplo 7-1.** Código em DOM2Writer.java a substituir

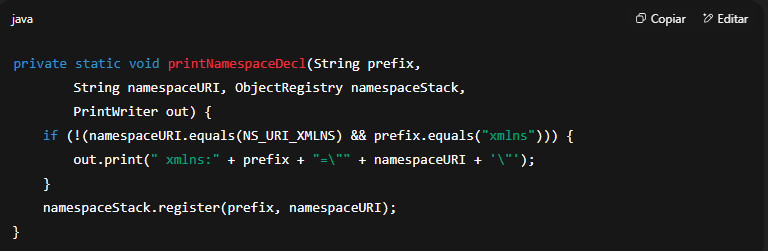


**Exemplo 7-2.** Novo código para DOM2Writer.java



Agora, adicione o novo método do **Exemplo 7-3** à classe DOM2Writer.

**Exemplo 7-3.** Novo método para a classe DOM2Writer



Em seguida, compile o pacote Apache SOAP.

**7.1.2.2 Compilando o Apache SOAP**

Para compilar o Apache SOAP, você precisará usar o **Ant**, uma ferramenta de gerenciamento de build para Java lançada pela Apache. O Ant está disponível em <http://jakarta.apache.org/> e é oficialmente parte do projeto Jakarta Tomcat.  
Depois de baixá-lo, siga as instruções detalhadas incluídas no pacote para instalá-lo.

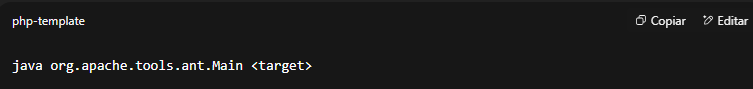
O Ant usa um script baseado em XML (build.xml) para definir como compilar o código. O arquivo build.xml do Apache SOAP está localizado em:



Antes de compilar, você deve garantir que todos os pré-requisitos estão presentes. Estes estão listados no início do arquivo build.xml:

* Qualquer parser XML habilitado para JAXP (o Xerces é preferido)
* O pacote **JavaMail**, disponível em <http://java.sun.com/products/javamail/>
* O pacote **Java Activation Framework**, disponível em <http://java.sun.com/products/beans/glasgow/jaf.html>

Todos esses pacotes devem estar no seu classpath antes de tentar compilar.  
Uma vez configurado, inicie a compilação usando o seguinte comando:



Onde <target> pode ser uma destas quatro opções:

* **compile** – Cria o pacote soap.jar
* **javadocs** – Cria a documentação JavaDocs do soap.jar
* **dist** – Cria a distribuição binária completa
* **srcdist** – Cria a distribuição completa do código-fonte

Para nosso propósito, use a opção **compile**. Isso criará um novo soap.jar com a classe DOM2Writer.java modificada incluída.  
Uma vez compilado, substitua todos os outros arquivos soap.jar que possam estar no classpath do seu servidor de aplicações pelo novo soap.jar.

**7.2 O Índice Code Share**

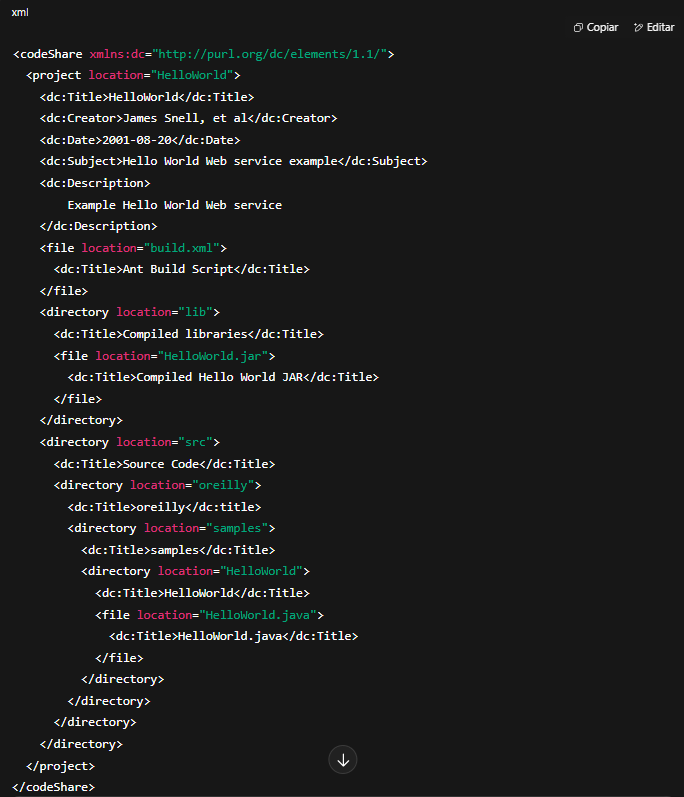
O código-fonte compartilhado através da rede CodeShare é organizado em torno de uma estrutura de índice simples que preserva a hierarquia original de diretórios e arquivos.  
Todos que desejam compartilhar código-fonte por meio do CodeShare devem criar um índice.

Como exemplo, vamos supor que estamos compartilhando o seguinte projeto Java:



Há um total de seis diretórios e três arquivos sendo compartilhados. Dentro do índice do CodeShare, representamos este projeto conforme o **Exemplo 7-4**.

**Exemplo 7-4.** Índice CodeShare para projeto de exemplo



Como você pode ver, a estrutura do índice é muito básica.  
O elemento <codeShare> é a raiz de todo o índice.  
O elemento <project> define um projeto compartilhado.  
O elemento <directory> define um diretório sendo compartilhado dentro de um projeto.  
O elemento <file> define um arquivo sendo compartilhado.

A característica mais interessante do índice é o uso de elementos de metadados **Dublin Core** (dc:Title, por exemplo) para adicionar propriedades descritivas a cada um dos itens compartilhados.

O projeto de metadados Dublin Core é uma iniciativa para definir tipos padrão de metadados (dados sobre dados) capazes de descrever conteúdo na Internet.  
Nós o usamos aqui para oferecer opções de busca mais flexíveis quando as pessoas procuram por tipos específicos de código.  
Sem esses elementos descritivos, a capacidade de busca do CodeShare ficaria limitada apenas à pesquisa baseada no nome do arquivo ou diretório.  
Mais adiante, veremos exatamente como esses dados adicionais são usados.

A especificação Dublin Core (<http://www.dublincore.org/documents/dces/>) define um conjunto de 15 elementos de metadados, todos utilizáveis dentro do índice do CodeShare.  
Os elementos são descritos na **Tabela 7-1**.

**Tabela 7-1.** Conjunto de elementos Dublin Core

| **Nome do elemento** | **Descrição do elemento** |
| --- | --- |
| **Title** | O nome dado ao recurso |
| **Creator** | A entidade responsável por criar o recurso |
| **Subject** | Um tópico curto que descreve o recurso |
| **Description** | Uma descrição textual detalhada do recurso |
| **Publisher** | A entidade responsável por disponibilizar o recurso |
| **Contributor** | Uma entidade responsável por contribuições ao recurso |
| **Date** | Tipicamente, a data em que o recurso foi criado |
| **Type** | O tipo genérico de recurso (não o tipo de conteúdo MIME) |
| **Format** | O tipo de conteúdo MIME ou outro formato físico do recurso |
| **Identifier** | Uma referência inequívoca ao recurso |
| **Source** | Uma referência ao recurso do qual este recurso foi derivado |
| **Language** | O idioma (não a linguagem de programação) no qual o recurso é apresentado |
| **Relation** | Uma referência a um recurso relacionado |
| **Coverage** | A extensão ou escopo do recurso |
| **Rights** | Informações sobre direitos mantidos sobre o recurso |

**7.3 Segurança em Serviços Web**

O que significa adicionar segurança a serviços web?  
No caso do exemplo CodeShare, nosso objetivo é permitir que os proprietários do código especifiquem direitos de acesso para indivíduos específicos.  
Se um usuário não estiver na lista de usuários aprovados, ele não poderá fazer download do código.

Segurança em serviços web significa adicionar recursos básicos de segurança às tecnologias que tornam os serviços web possíveis.  
Isso significa ter a capacidade de **criptografar mensagens SOAP**, **assinar digitalmente descrições de serviço WSDL**, adicionar confiabilidade aos protocolos de transporte que usamos para transmitir essas informações, **afirmar a identidade de um usuário**, definir políticas que regem **como** as informações devem ser usadas, **por quem** podem ser usadas e **para que fins** podem ser usadas — e qualquer outra quantidade de itens de uma lista extensa.

Poderia ser necessário quase um livro inteiro apenas para descrever como implementar todos esses requisitos.  
Infelizmente, embora esforços estejam sendo feitos em cada uma dessas áreas, ainda estamos longe de ter padrões definidos (de fato ou não) sobre como tudo isso acontecerá no ambiente de serviços web.  
No exemplo CodeShare, focamos apenas em um: **autenticação de usuário**.

A autenticação em serviços web baseados em SOAP pode ocorrer de várias maneiras.  
O serviço pode optar por usar métodos tradicionais de autenticação em nível de transporte, como **HTTP Basic** ou **Digest Authentication**.  
Alternativamente, o serviço pode optar por implementar um mecanismo de autenticação em nível de serviço, tornando o próprio serviço responsável por validar a identidade de um usuário.

A segunda abordagem é a que vemos emergir na forma do **serviço de autenticação Passport da Microsoft**, que fornece autenticação baseada em **Kerberos** sobre protocolos de serviços web.  
O Kerberos é um mecanismo popular de autenticação padrão da Internet baseado na troca de *tickets*.  
Esses *tickets* são usados de maneira semelhante a um ingresso de cinema.  
O portador do ticket o apresenta como um passe para entrar no filme ou, no nosso caso, para acessar um serviço.

O Capítulo 8 discute o esquema de autenticação Passport e várias outras abordagens alternativas em mais detalhes.

**7.3.1 A *Security Assertions Markup Language* (SAML)**

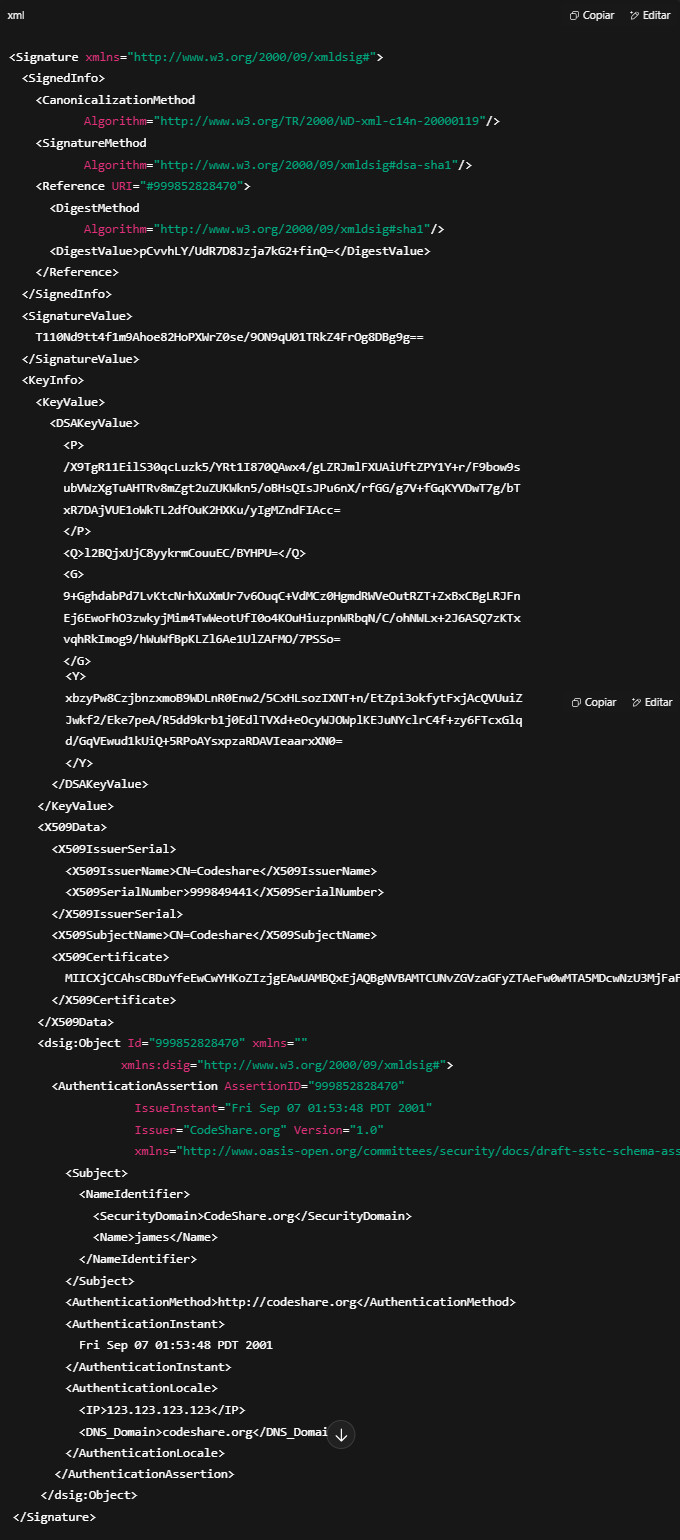
Uma das muitas tecnologias emergentes de serviços web é projetada especificamente para ser usada como método de implementação de login global em nível de serviço (*service-layer global sign-on*) para serviços web.  
A especificação, chamada **Security Assertions Markup Language**, ou **SAML**, define uma sintaxe XML para expressar fatos relacionados à segurança.

Por exemplo, o SAML pode ser usado para expressar o fato de que *Pavel Kulchenko* foi autenticado às 10h00 e que a autenticação expira às 14h00.

As afirmações SAML, como são chamadas, são criadas e assinadas digitalmente pela autoridade de autenticação que realiza o processo real de autenticação.  
Por exemplo, quando um usuário invoca a operação login na interface do cliente CodeShare, o servidor CodeShare (que valida o ID de usuário e a senha) emite a afirmação SAML indicando que o login foi bem-sucedido.  
Ao assinar digitalmente essa afirmação, qualquer pessoa que a receba pode validar que ela foi, de fato, criada e emitida pelo servidor CodeShare.

O **Exemplo 7-5** é uma afirmação SAML assinada digitalmente retornada pela operação login.  
A própria afirmação está destacada em **negrito**.  
A primeira parte desta estrutura é a **Assinatura Digital XML**, que valida que a afirmação SAML é autêntica.  
As Assinaturas Digitais XML estão sendo padronizadas por meio de um esforço conjunto do W3C e do IETF.  
A estrutura dessas assinaturas é muito complexa para explicar aqui, então fornecemos links para informações suplementares no Capítulo 8.  
Felizmente, não precisamos criar essas assinaturas manualmente.  
Este exemplo em particular foi criado usando o **IBM XML Security Suite**.

**Exemplo 7-5.** Afirmação SAML



O objetivo deste exemplo de afirmação SAML é declarar que o usuário **james**, do domínio **CodeShare.org**, foi autenticado na **sexta-feira, 7 de setembro de 2001**, às **13:53** no horário **Pacific Daylight Time (PDT)**, usando o método de autenticação padrão do CodeShare (a operação login).  
A autenticação em si foi fornecida por um servidor localizado no endereço IP **123.123.123.123**, com o nome de domínio **codeshare.org**.

Essa declaração é assinada digitalmente usando o certificado digital X509 do servidor CodeShare, garantindo sua autenticidade.

Quando um usuário apresenta este token a um proprietário CodeShare, o proprietário pode verificar se ele é autêntico perguntando ao servidor CodeShare se realmente emitiu a declaração.  
A **Figura 7-2** ilustra o fluxo de mensagens.

**Figura 7-2.** Um fluxo ilustrando a conversa típica entre o proprietário do CodeShare e o servidor CodeShare

As afirmações SAML podem ser criadas e validadas por qualquer pessoa, tornando-as um mecanismo muito bom para implementar funcionalidades de login único (*single sign-on*).  
Mais adiante neste capítulo, demonstraremos como esta afirmação SAML foi criada e assinada.

**7.4 Definições e Descrições**

Como serviços web são totalmente sobre as interfaces que conectam aplicativos, a primeira coisa que precisamos fazer é definir como essas interfaces serão.

Neste exemplo, existem quatro interfaces de interesse:

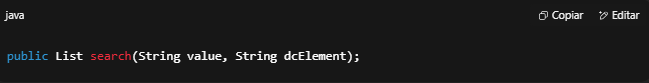
* **A interface do proprietário**, implementada pelos proprietários CodeShare e pelo servidor CodeShare, para permitir que usuários pesquisem e recuperem código-fonte.
* **A interface do cliente**, implementada pelo servidor CodeShare, para permitir que usuários se registrem e façam login.
* **A interface de verificação de login**, implementada pelo servidor CodeShare, para permitir que os proprietários CodeShare garantam que os usuários tenham feito login.
* **A interface de índice mestre**, que permite aos proprietários CodeShare atualizar suas entradas no índice mestre mantido pelo servidor CodeShare.

Cada uma dessas interfaces é expressa usando descrições de interface de serviço WSDL.

**7.4.1 A Interface do Proprietário**

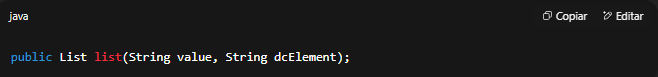
A interface do proprietário consiste em quatro operações fundamentais:

* **search**  
  Pesquisa no arquivo index.xml por elementos que correspondam a um determinado valor. Por padrão, esta operação pesquisa apenas o elemento **Dublin Core Title**, mas outros elementos Dublin Core também podem ser alvo.  
  Se fosse uma função Java, seria algo como:



 Nesse exemplo, dcElement corresponde ao nome do elemento Dublin Core que você deseja pesquisar.

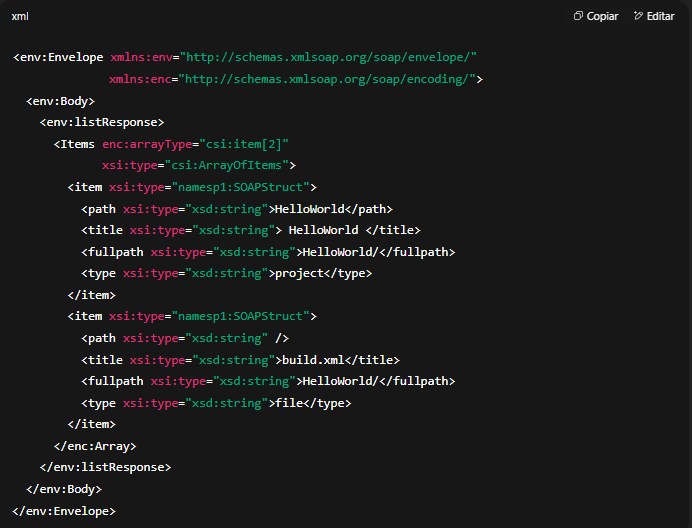
 **list**  
Lista todos os projetos e itens sendo compartilhados pelo proprietário. Apenas informações básicas sobre os itens são retornadas, incluindo o local e o título dos itens.  
Filtros podem ser aplicados para retornar apenas itens que correspondam a um determinado valor de um elemento Dublin Core especificado. Assim como na operação search, o título é o elemento padrão sobre o qual os filtros são aplicados.  
Em Java, a assinatura seria:



* Diferente da operação search, os parâmetros value e dcElement são opcionais.
* **info**  
  Retorna informações detalhadas sobre cada um dos projetos e itens sendo compartilhados. Assim como na operação list, filtros podem ser aplicados para limitar o número de itens retornados. A assinatura dessa operação é exatamente a mesma da operação list.
* **get**  
  Retorna todos os arquivos sendo compartilhados para um projeto ou projetos especificados. As estruturas de diretórios serão recriadas exatamente. A assinatura desta operação também é a mesma que a da operação list.

Todas essas operações retornam um array codificado em SOAP de elementos item, semelhante ao **Exemplo 7-6**.

**Exemplo 7-6.** Array de exemplo retornado por operações do proprietário



**7.4.1.1 Definição de tipo de porta WSDL**

A definição completa do tipo de porta WSDL está no Apêndice C.  
Aqui, cobriremos os destaques: os tipos de dados, as mensagens, o tipo de porta (*port type*) e o vínculo de protocolo (*protocol binding*).

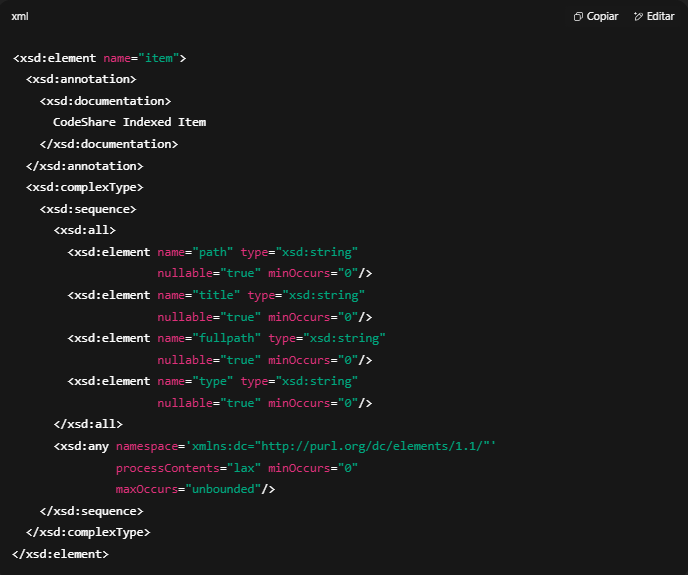
**7.4.1.2 Tipos de dados**

Os tipos de dados são definidos com um *schema* XML incorporado.  
Este *schema* define dois tipos de dados:

* um **item**, que, como vimos no envelope SOAP anterior, representa um item de projeto dentro do índice CodeShare,
* e um **array de itens**.

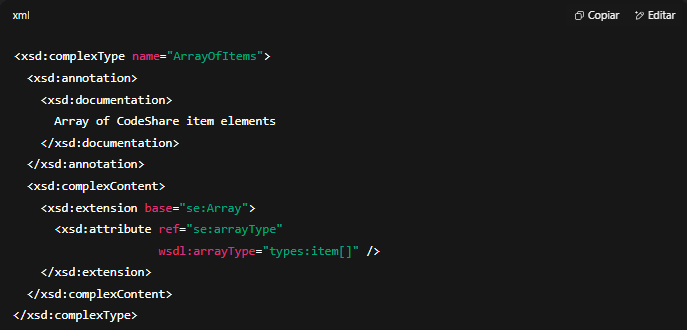
A definição de item, mostrada no **Exemplo 7-7**, é direta.  
É um elemento de tipo complexo com quatro elementos filhos e um marcador para incluir quaisquer elementos Dublin Core que o serviço CodeShare possa querer adicionar.

**Exemplo 7-7.** A definição de item



O tipo de dado **ArrayOfItems**, mostrado no **Exemplo 7-8**, é derivado do tipo de dados Array definido pelo estilo de codificação SOAP Seção 5.  
Com esta definição, declaramos que este é um *array* de elementos item, conforme especificado pelas regras de codificação da Seção 5.

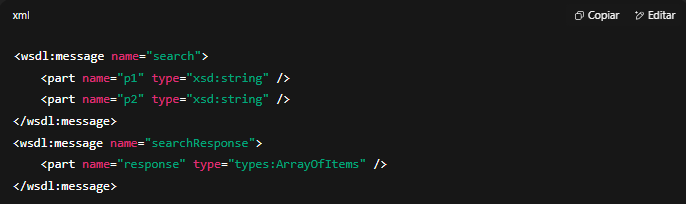
**Exemplo 7-8.** A definição de ArrayOfItems



**7.4.1.3 Mensagens**

Há exatamente duas mensagens definidas para cada operação.  
As mensagens de uma operação de exemplo são definidas no **Exemplo 7-9**.  
A listagem completa está no Apêndice C.

**Exemplo 7-9.** Definições de mensagens search

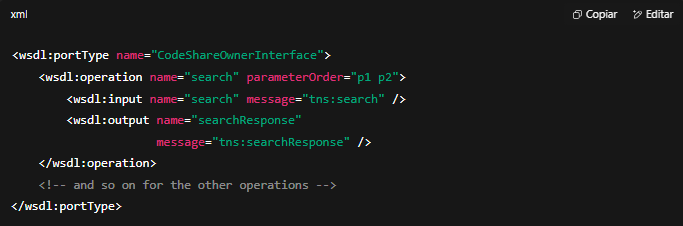


**7.4.1.4 Tipo de porta (*Port type*)**

O tipo de porta da interface do proprietário é definido em termos das mensagens que acabamos de criar.  
O **Exemplo 7-10** mostra o elemento portType com uma operação representativa definida.  
Veja no Apêndice C a listagem completa.

O atributo parameterOrder é um mecanismo WSDL para especificar a ordem em que as partes de uma mensagem devem aparecer dentro do corpo da mensagem SOAP.

**Exemplo 7-10.** A definição de portType



**7.4.1.5 Vinculação de protocolo (*Protocol binding*)**

O **Exemplo 7-11** especifica que o tipo de porta da interface do proprietário é acessado via mensagens SOAP transportadas sobre HTTP.  
Cada mensagem SOAP seguirá o estilo de codificação da Seção 5 (como indicado pelos elementos soap:body).  
Como antes, apenas uma operação é mostrada aqui.  
Para o conjunto completo, veja o WSDL completo no Apêndice C.

**Exemplo 7-11.** Vinculando a interface ao portType



**7.4.2 A Interface do Cliente**

A interface do cliente consiste em duas operações: **register** e **login**.  
Como a interface WSDL é semelhante, em estrutura, à definida para a interface do proprietário, não a mostraremos aqui.  
Ela está impressa na íntegra no Apêndice C.

**register**

Recebe um par simples de ID de usuário e senha para criar uma nova conta de usuário no servidor CodeShare.

**login**

Recebe uma string codificada em Base64 composta pelo ID de usuário e senha, e retorna uma afirmação SAML assinada digitalmente indicando que o usuário fez login com sucesso.  
Se o login não for bem-sucedido, uma *fault* SOAP com o tipo "Client.Authentication" será retornada.

**7.4.2.1 Operação de login do CodeShare**

A operação de login do CodeShare valida o ID de usuário e a senha, e gera uma afirmação SAML assinada, como mostrado na **Figura 7-3**.  
Não é uma solução de segurança perfeita — a especificação SAML ainda carece de algumas partes muito importantes (por exemplo, é muito fácil para alguém interceptar uma afirmação SAML assinada e fingir ser a pessoa para quem ela foi emitida).  
Vai levar algum tempo até que todas essas questões sejam resolvidas.  
Para nossos propósitos, precisamos apenas de algo simples, apenas para demonstrar a ideia básica.

**Figura 7-3.** Um fluxo ilustrando a conversa típica entre o cliente CodeShare e o servidor CodeShare

**7.4.3 A Interface de Verificação de Login**

O usuário apresenta as afirmações SAML do servidor CodeShare para recuperar o código sendo compartilhado.  
A afirmação é entregue ao proprietário do código, que deve validar se ela realmente veio do serviço CodeShare.  
A interface de verificação de login do CodeShare fornece essa funcionalidade.

Há apenas uma única operação **verify** definida por esta interface.  
Ela recebe a afirmação SAML como entrada e retorna um valor simples verdadeiro (*true*) ou falso (*false*) indicando a validade da afirmação.  
Veja o Apêndice C para o WSDL completo.

**7.4.4 A Interface do Índice Mestre**

Além de fornecer as funções de gerenciamento de usuários e autenticação para a rede CodeShare, o serviço CodeShare também fornece um índice mestre de todo o código sendo compartilhado.  
Os proprietários de código atualizam este índice mestre por meio da interface do índice mestre.

Esta interface define duas operações:

* **register**, pela qual novos proprietários participam da rede CodeShare,
* e **update**, que permite aos proprietários atualizar suas entradas no índice mestre.

A operação update recebe o ID do proprietário, a senha e o índice do projeto atualizado.

**7.5 Implementando o Servidor CodeShare**

O servidor CodeShare é implementado como um conjunto de classes Java que mantêm um índice mestre de todos os proprietários CodeShare que estão compartilhando código e todos os usuários registrados que podem ter acesso a esse código.  
O servidor é dividido em quatro serviços web distintos, cada um sendo uma implementação das quatro interfaces que já definimos:

* o **serviço de índice mestre**,
* o **serviço de proprietário**,
* o **serviço de cliente**,
* e o **serviço de verificação**.

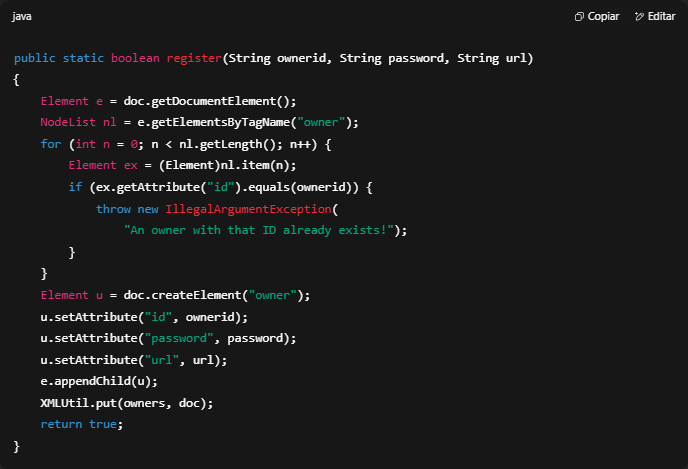
**7.5.1 O Serviço de Índice Mestre**

O serviço de índice mestre permite que os proprietários CodeShare atualizem suas entradas no índice mantido pelo servidor CodeShare.  
A classe codeshare.OwnerService implementa o serviço de índice.

**7.5.1.1 Operações**

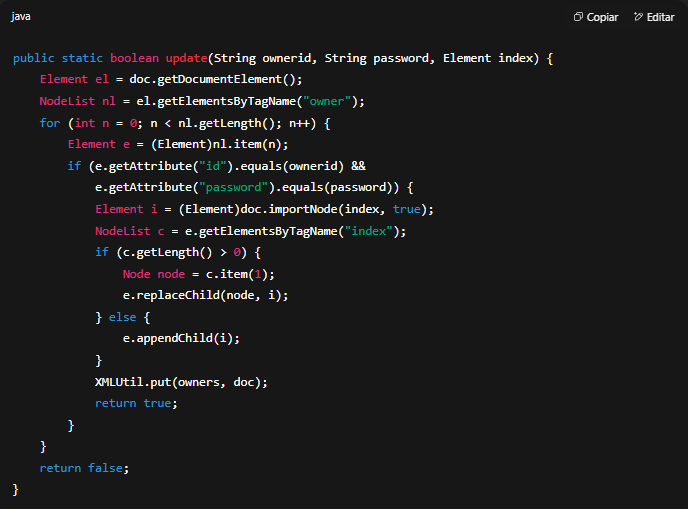
A lista de proprietários registrados é armazenada como um arquivo XML.  
A operação register, no **Exemplo 7-12**, simplesmente adiciona um novo elemento a esse arquivo XML.

**Exemplo 7-12.** A operação register



O próprio índice mestre (a lista de todo o código sendo compartilhado através da rede CodeShare) também é mantido como um arquivo XML.  
Assim como na operação register, a operação update mostrada no **Exemplo 7-13** não faz nada além de atualizar esse arquivo XML, inserindo o índice passado pelo proprietário ou substituindo uma parte existente do índice que tenha sido atualizada anteriormente.

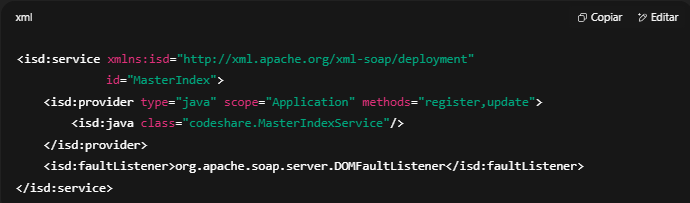
**Exemplo 7-13.** A operação update



**7.5.1.2 Implantação**

O serviço de índice mestre é implantado no servidor Apache SOAP usando o seguinte arquivo de implantação no formato de implementação Java, mostrado no **Exemplo 7-14**.

**Exemplo 7-14.** Arquivo de implantação do serviço de índice mestre



**7.5.2 O Serviço de Proprietário**

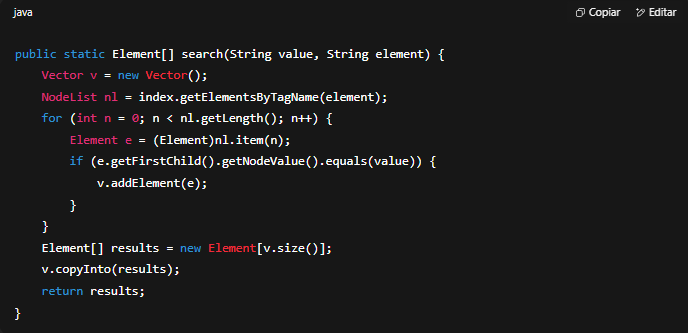
O serviço de proprietário é implementado pela classe codeshare.OwnerService.  
Ele fornece acesso ao índice do proprietário para que os usuários possam procurar e recuperar código.  
Cada proprietário CodeShare executa uma instância desse serviço.

**7.5.2.1 Operações**

O índice do proprietário é mantido em um arquivo XML.  
As operações search, list e info retornam subconjuntos desse índice, enquanto a operação get retorna o próprio código.

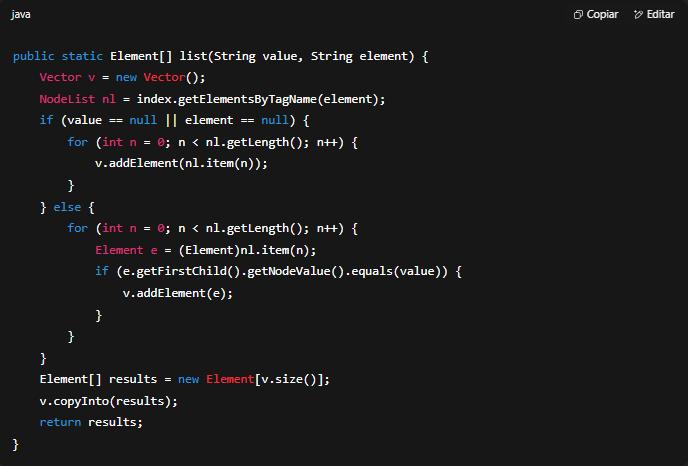
A operação search, mostrada no **Exemplo 7-15**, percorre o índice procurando por um valor específico dentro de um elemento Dublin Core especificado.

**Exemplo 7-15.** A operação search



A operação list, mostrada no **Exemplo 7-16**, retorna todos os elementos do índice, possivelmente filtrados por um elemento Dublin Core e um valor.

**Exemplo 7-16.** A operação list



A operação info é idêntica à operação list, exceto que retorna informações mais detalhadas sobre cada item.  
A operação get, mostrada no **Exemplo 7-17**, retorna o próprio arquivo solicitado.

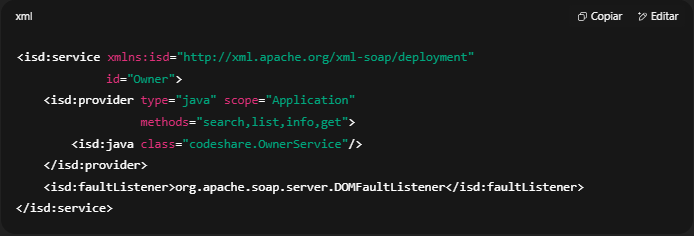
**Exemplo 7-17.** A operação get



**7.5.2.2 Implantação**

O serviço de proprietário é implantado no servidor Apache SOAP usando o seguinte arquivo de implantação, mostrado no **Exemplo 7-18**.

**Exemplo 7-18.** Arquivo de implantação do serviço de proprietário



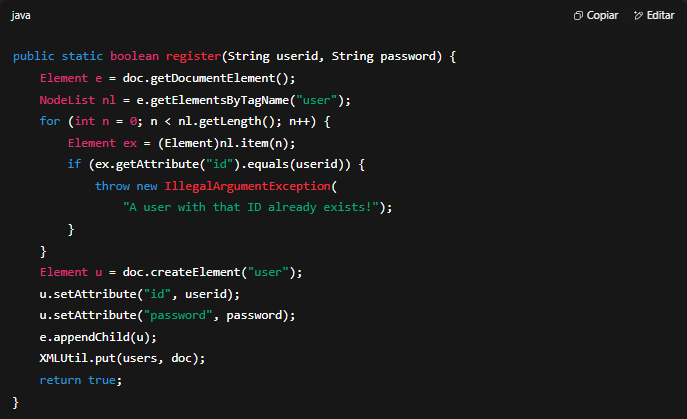
**7.5.3 O Serviço de Cliente**

O serviço de cliente é implementado pela classe codeshare.ClientService.  
Ele fornece as funções de registro e login para os usuários CodeShare.

**7.5.3.1 Operações**

A operação register simplesmente adiciona um novo usuário ao arquivo XML de usuários registrados, conforme mostrado no **Exemplo 7-19**.

**Exemplo 7-19.** A operação register



A operação login, mostrada no **Exemplo 7-20**, valida o ID de usuário e a senha e gera uma afirmação SAML assinada.

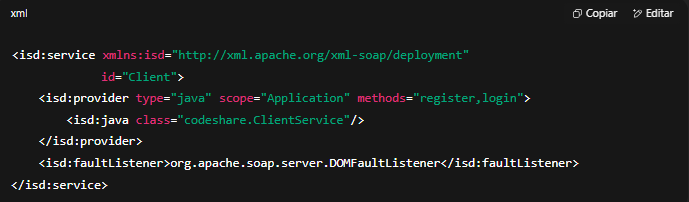
**Exemplo 7-20.** A operação login



**7.5.3.2 Implantação**

O serviço de cliente é implantado no servidor Apache SOAP usando o seguinte arquivo de implantação, mostrado no **Exemplo 7-21**.

**Exemplo 7-21.** Arquivo de implantação do serviço de cliente



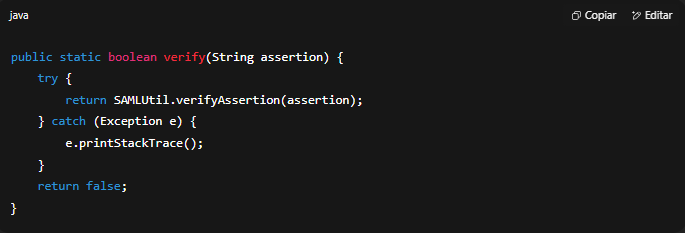
**7.5.4 O Serviço de Verificação**

O serviço de verificação é implementado pela classe codeshare.VerifyService.  
Ele fornece a funcionalidade de validação das afirmações SAML recebidas dos clientes.

**7.5.4.1 Operações**

A operação verify, mostrada no **Exemplo 7-22**, verifica se a afirmação SAML apresentada é válida e foi realmente emitida pelo serviço CodeShare.

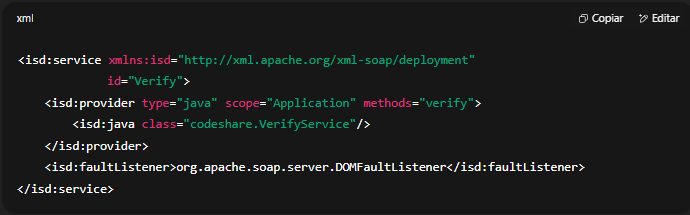
**Exemplo 7-22.** A operação verify



**7.5.4.2 Implantação**

O serviço de verificação é implantado no servidor Apache SOAP usando o seguinte arquivo de implantação, mostrado no **Exemplo 7-23**.

**Exemplo 7-23.** Arquivo de implantação do serviço de verificação



**7.6 Implementando o Cliente CodeShare**

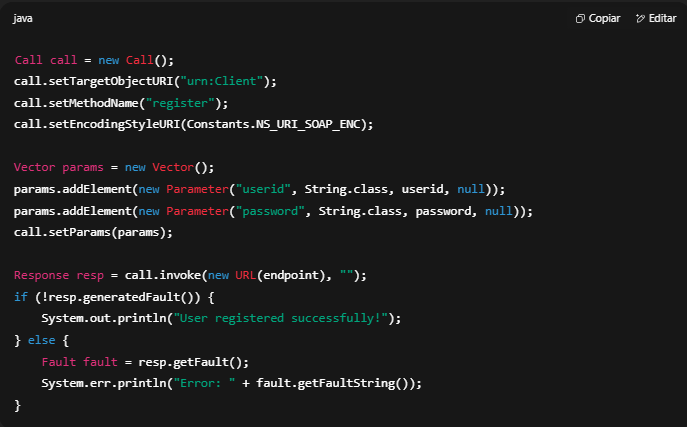
O cliente CodeShare é uma aplicação Java que se comunica com os quatro serviços web CodeShare que implementamos.  
Ele permite que um usuário se registre, faça login, pesquise e recupere código de qualquer proprietário registrado na rede CodeShare.

O cliente é construído como uma interface gráfica simples usando Swing, mas a parte importante é a forma como ele invoca remotamente os métodos desses serviços SOAP.

**7.6.1 Registrando um Novo Usuário**

Para registrar um novo usuário, o cliente chama a operação register do **serviço de cliente**.  
O código mostrado no **Exemplo 7-24** cria uma chamada SOAP usando a API do Apache SOAP, define os parâmetros e invoca o serviço.

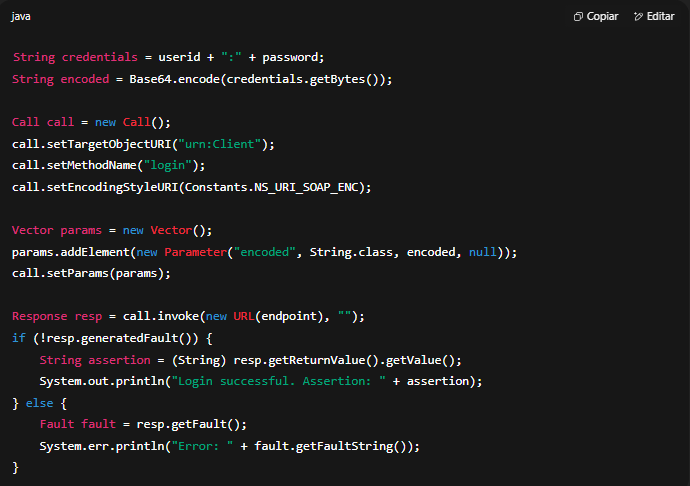
**Exemplo 7-24.** Registrando um novo usuário



**7.6.2 Fazendo Login**

O login é feito enviando uma string codificada em Base64 para a operação login do **serviço de cliente**.  
Se o login for bem-sucedido, uma afirmação SAML assinada será retornada.  
O **Exemplo 7-25** mostra como isso é feito no cliente.

**Exemplo 7-25.** Fazendo login



**7.6.3 Pesquisando Código**

Para procurar código no índice de um proprietário, o cliente invoca a operação search do **serviço de proprietário**.  
O **Exemplo 7-26** mostra como uma pesquisa é realizada.

**Exemplo 7-26.** Pesquisando no índice do proprietário



**7.6.4 Obtendo Código**

Quando o usuário seleciona um item de código para baixar, o cliente invoca a operação get do **serviço de proprietário**.  
O **Exemplo 7-27** mostra como o arquivo é recuperado e salvo localmente.

**Exemplo 7-27.** Obtendo código



**7.6.5 Verificando Afirmações SAML**

Antes que um proprietário de código envie o código solicitado para um cliente, ele deve verificar se a afirmação SAML apresentada é válida.  
Isso é feito chamando a operação verify do **serviço de verificação**.  
O **Exemplo 7-28** mostra como o cliente realiza essa verificação.

**Exemplo 7-28.** Verificando uma afirmação SAML



**7.7 Considerações de Segurança**

O sistema CodeShare que acabamos de implementar é, evidentemente, simplista e não deve ser usado em um ambiente de produção real sem modificações substanciais.  
Algumas das limitações mais sérias incluem:

* As afirmações SAML podem ser interceptadas e reutilizadas por terceiros mal-intencionados, uma vez que não há proteção contra *replay attacks*.
* A autenticação e a transmissão de dados não são protegidas por criptografia de transporte, como SSL/TLS.
* As senhas são armazenadas em texto simples nos arquivos XML.

Em um sistema real, deveríamos:

* Usar SSL/TLS para criptografar todas as comunicações.
* Armazenar senhas usando hashes seguros com *salt*.
* Adicionar um mecanismo de expiração e *nonce* às afirmações SAML para evitar reutilização.

**7.8 Resumo**

Neste capítulo, exploramos um sistema de compartilhamento de código simples, mas funcional, construído usando serviços web baseados em SOAP.  
Projetamos e implementamos quatro serviços distintos:

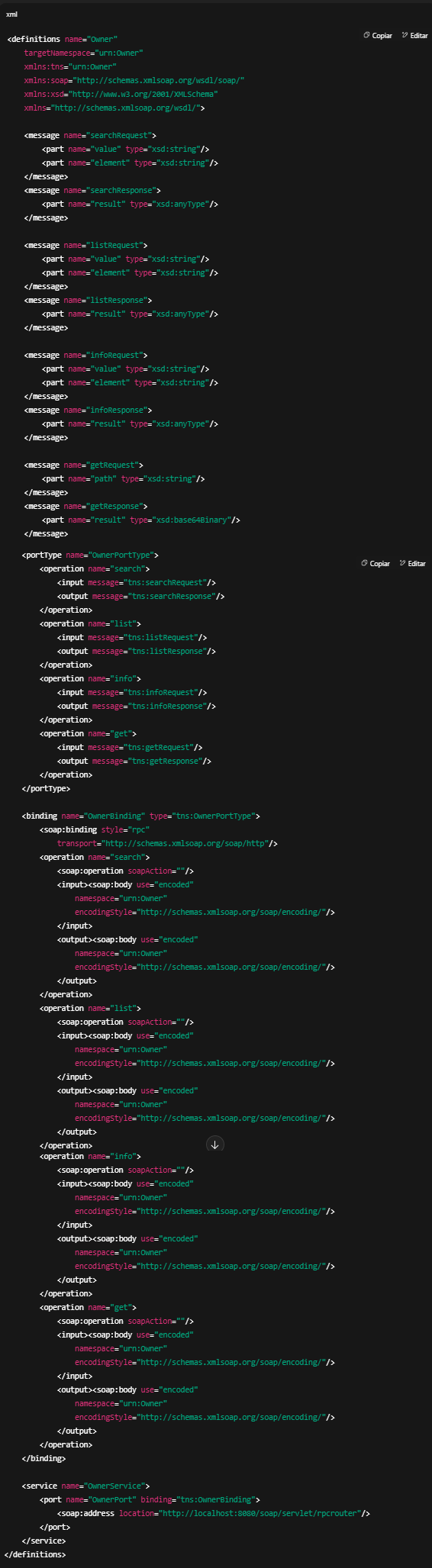
1. **Serviço de índice mestre** – mantém a lista de todos os proprietários e seus índices de código.
2. **Serviço de proprietário** – fornece acesso ao código e às informações de índice de um proprietário específico.
3. **Serviço de cliente** – gerencia o registro e o login de usuários.
4. **Serviço de verificação** – valida afirmações SAML apresentadas para autenticação.

Também implementamos um cliente gráfico simples que interage com esses serviços para registrar usuários, autenticar logins, procurar e obter código.  
Embora a segurança do exemplo seja propositalmente básica, a estrutura apresentada pode ser estendida para construir sistemas distribuídos robustos e seguros.

**Apêndice C – WSDLs do CodeShare**

A seguir estão os documentos WSDL completos para as quatro interfaces definidas no sistema CodeShare: **Proprietário**, **Cliente**, **Verificação de Login** e **Índice Mestre**.

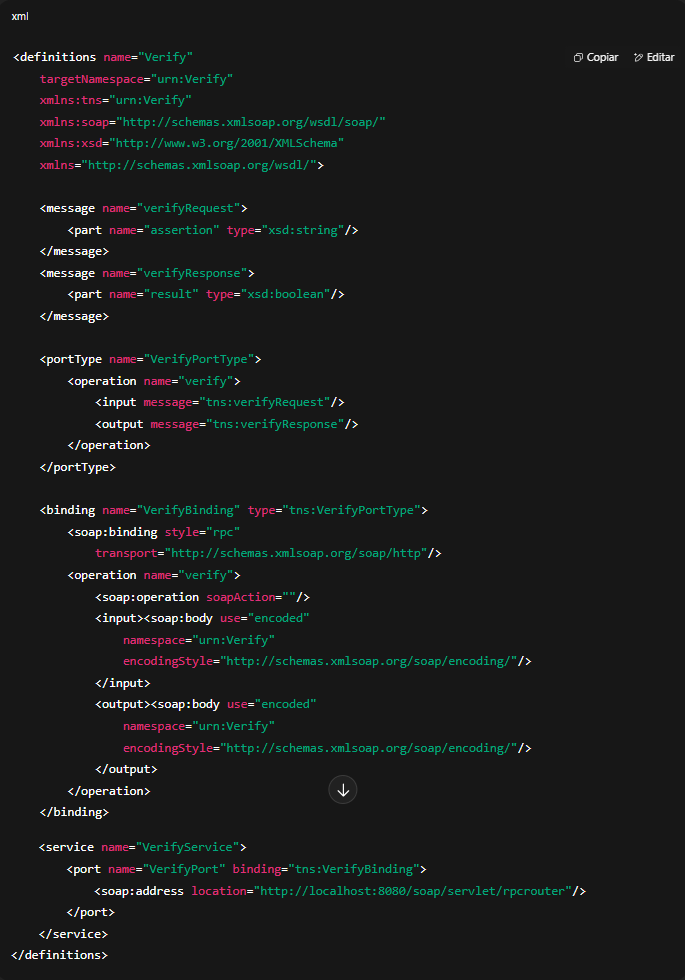
**C.1 Interface do Proprietário**



**C.2 Interface do Cliente**



**C.3 Interface de Verificação de Login**



**C.4 Interface do Índice Mestre**

