UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE PROTOCOLO DE APLICAÇÃO PARA TROCA DE DOCUMENTOS DA ÁREA EXTRAJUDICIAL

FABRÍCIO BENTO

BLUMENAU
2005

2005/1-15

FABRÍCIO BENTO

PROTÓTIPO DE PROTOCOLO DE APLICAÇÃO PARA TROCA DE DOCUMENTOS DA ÁREA EXTRAJUDICIAL

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Paulo Fernando da Silva, Orientador

BLUMENAU
2005

2005/1-15

PROTÓTIPO DE PROTOCOLO DE APLICAÇÃO PARA TROCA DE DOCUMENTOS DA ÁREA EXTRAJUDICIAL

Por

FABRÍCIO BENTO

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Paulo Fernando da Silva, Orientador, FURB
Membro:	Prof. Sérgio Stringari, Banca 2 – FURB
Membro:	Prof. Alexander Roberto Valdameri, Banca 3 – FURB

Blumenau, 22 de junho de 2005

Dedico este trabalho a meus pais, minha esposa e a todos os amigos, especialmente aqueles que me ajudaram diretamente na realização deste.

AGRADECIMENTOS

À Deus, pelo seu imenso amor e graça.

Aos meus pais, que mesmo longe, sempre estiveram presentes em meus pensamentos.

À minha esposa Vanessa e sua família, por seu amor, compreensão e apoio nas horas difíceis.

Ao meu orientador, Prof. Paulo Fernando da Silva, que foi capaz de me guiar durante a difícil caminhada que foi o desenvolvimento desse trabalho.

Aos meus amigos, pelos empurrões e cobranças, especialmente Jean Carlos Zimmermann, Júlio César Zimmermann e Eduardo Teixeira que me auxiliaram de alguma forma para que esse trabalho se tornasse realidade.

A todas as outras pessoas que me apoiaram durante essa longa jornada pelo curso de Bacharelado em Ciências da Computação.

.

RESUMO

Este trabalho descreve a análise, especificação e implementação de um protótipo de protocolo de aplicação para realizar a troca de documentos entre sistemas de cartórios distintos, utilizando serviços web. Este protocolo disponibiliza métodos em uma classe Java armazenada em um contêiner HTTP. Os métodos são invocados através de mensagens SOAP. Para fins de teste foi implementado um aplicativo cliente que acessa dados em uma base de dados SQL remota através de métodos de uma classe servidora de serviços.

Palavras-chave: Protocolo. Cartórios. Serviços web.

ABSTRACT

This work describes the analisys, specification and implementation of an application protocol prototype to realize the document interchange among different offices system, using web services. This protocol make available methods at a Java class stored into HTTP container, The methods is invoked by SOAP messages. For test ends, it was implemented a client application of the service. It access data in a remote SQL database by methods of a service server class.

Key-Words: Protocol. Offices. Web services.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

ACAFE - Associação Catarinense das Fundações Educacionais

AXIS – *Apache eXtensible Interaction System*

B2B – *Business to Business*

CML – *Chemical Markup Language*

DTD – *Document Type Definition*

FTP – *File Transfer Protocol*

HTTP – *Hypertext Transfer Protocol*

IBGE – Instituto Brasileiro de Geografia e Estatística

INSS – Instituto Nacional de Seguridade Social

JDBC – *Java Database Connectivity*

MathML – *Mathematical Markup Language*

PC – *Personal Computer*

PDA – *Personal Digital Assistant*

RCPN – Registro Civil das Pessoas Naturais

SGML – *Standart Generalized Markup Language*

SMTP – *Simple Mail Transfer Protocol*

SOA – *Service-Oriented Architecture*

SOAP – *Simple Object Access Protocol*

UML – *Unified Modeling Language*

UDDI – *Universal Description, Discover, and Integration*

XML – *eXtensible Markup Language*

WSDL – *Web Service Description Language*

W3C – *World Wide Web Consortium*

SUMÁRIO

1 INTRODUÇÃO

Atualmente, a utilização de meios computacionais para armazenamento e troca de informações têm trazido inúmeros benefícios à população em geral. Empresas, indústrias, instituições, governos, pessoas, entre outras, dependem do processamento direto ou indireto de dados através do uso de computadores.

A popularização do uso da internet em todo o mundo proporciona o intercâmbio incessante de informações num espaço de tempo muito curto, não importando a distância física entre origem e destino.

Atualmente as serventias extra judiciais, conhecidas popularmente como cartórios, utilizam computadores em substituição às máquinas datilográficas. Com isso houve uma enorme evolução quanto à qualidade e agilidade dos serviços prestados pelos cartórios, que compõem os registros públicos. Apesar disto, nem todas as serventias extra judiciais utilizam mesmos aplicativos para gerenciamento dos registros armazenados, aplicativos estes que são fornecidos por vários fabricantes diferentes.

Entre os cartórios existe a necessidade de envio de dados através dos serviços prestados pelos correios. São enviados editais, pedidos de alterações de registros relacionados à casamento de pessoas, notificações extra judiciais à serem cumpridas em outras comarcas, entre outros. Existe também a obrigatoriedade do fornecimento de relatórios baseados nos registros feitos para órgãos como Receita Federal, Instituto Brasileiro de Geografia e Estatística (IBGE), Instituto Nacional de Seguridade Social (INSS), entre outros. Estes órgãos utilizam as informações na fiscalização de declarações de imposto de renda, no levantamento de dados estatísticos e controle de pagamento de aposentadorias.

Como os sistemas dos cartórios não são integrados, existe a dependência direta dos serviços dos correios no envio e recebimento de documentos, além da obrigatoriedade de

utilizar sistemas fornecidos pelos órgãos públicos citados anteriormente, para envio de dados.

Diante destes obstáculos emergiu a intenção de se desenvolver um protocolo de aplicação para a área extra judicial para promover a integração consistente entre os diferentes sistemas utilizados pelos cartórios. A utilização deste protocolo é independente de arquitetura de sistema operacional, banco de dados e linguagem de programação.

Com a realização deste trabalho desenvolveu-se um protótipo de protocolo de aplicação para um dos registros públicos da área extra judicial. O registro público escolhido foi o Registro Civil das Pessoas Naturais (RCPN).

O protocolo utilizará a *eXtensible Markup Language* (XML) em conjunto com *Simple Object Access Protocol* (SOAP), que serão responsáveis pela confecção e transporte dos documentos respectivamente. Serão utilizados também recursos de implementação de *web services* para disponibilizar serviços invocados através do uso do protocolo.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é a criação de um protótipo de protocolo de aplicação para realizar a troca de documentos entre sistemas computacionais heterogêneos utilizados pelos serviços extra judiciais, serviços esses que compõem os registros públicos existentes no país. Dentre os principais registros públicos foi escolhido o registro civil das pessoas naturais como base para realização desta integração entre aplicativos distintos.

Os objetivos específicos do trabalho são:

a) utilizar a XML *Schema* como base para o desenvolvimento de uma linguagem de marcação para a confecção dos documentos que serão utilizados no intercâmbio de informações do registro civil das pessoas naturais;
b) criar um *web service* para disponibilizar o serviço de troca de documentos;

c) criar um aplicativo cliente do serviço para demonstração do funcionamento do protocolo.

1.2 ESTRUTURA DO TRABALHO

No primeiro capítulo são apresentados objetivos e a estrutura deste trabalho.

No segundo capítulo são apresentados os registros públicos, as tecnologias utilizadas neste trabalho, tais como XML, XML *Schema*, SOAP, *Apache eXtensible Interaction System* (AXIS) e alguns trabalhos correlatos.

No terceiro capítulo são apresentados os requisitos, a especificação e a implementação do protótipo de protocolo de aplicação.

No quarto capítulo são apresentadas as conclusões e sugestões para o desenvolvimento de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nas próximas seções são apresentados conceitos e técnicas que foram relevantes no desenvolvimento deste trabalho.

2.1 REGISTROS PÚBLICOS

Segundo Filho (2004), os registros públicos são feitos em serventias pertencentes as comarcas e reconhecidas por lei, que são controladas por um oficial que tem a função de realizar os registros.

O oficial citado anteriormente é o registrador público, detentor de um conjunto de formalidades prescritas por lei no registro de atos que atestam juridicamente fatos sociais. O registro dos atos é formalizado nas serventias extra judiciais, ou popularmente conhecidas como cartórios, expressão que caiu em desuso após a entrada em vigor da lei federal nº 10.169/2000, sendo substituído por serviços registrais e notariais.

2.1.1 HISTÓRIA DOS REGISTROS PÚBLICOS

De acordo com Junior (1963) a origem dos registros públicos confunde-se com a origem da própria civilização. Desde os primórdios podemos citar alguns exemplos da prática registral. Na antiga Mesopotâmia foram encontrados os primeiros sinais de transações imobiliárias. Os negócios eram lavrados pelos escribas em tabuletas de argila que eram entregues aos compradores e cópias eram conservadas por autoridades públicas.

Na época dos faraós, no antigo Egito, foi criado um modo sofisticado de publicidade registral, onde os registros públicos, chamados de *katagrafe*, eram incumbidos de registrar

contratos e cobrar impostos.

A civilização grega contava com os *mnemons* (notários) e os *hieromnemons* (arquivistas), que eram responsáveis pelos registros e conservação de contratos de transmissão imobiliária respectivamente.

No século VII, nos mosteiros e igrejas da Alemanha foram introduzidos livros de registro de propriedades imobiliárias, antepassados dos registros paroquiais.

O modelo de registro público atual foi instituído por voltado século XIX em diversos países como Brasil, Espanha e Portugal. No Brasil, durante quatro séculos, a igreja católica foi responsável pelos registros pessoais até que o registro civil começou a estabelecer-se, e tornou-se obrigatório com a proclamação da república, resultando na separação por completo da igreja e do estado.

2.1.2 PUBLICIDADE

"A publicidade é a alma dos Registros Públicos. É a oportunidade que o legislador quer dar ao povo de conhecer tudo o que lhe interessa a respeito de determinados atos. Deixa-o a par de todo o movimento de pessoas e bens." (FILHO, 1978, p. 28).

Em função deste fato os atos de registro podem ser fornecidos pelos serviços notariais e registrais através da emissão de certidões, que são documentos que comprovam a existência dos registros. Qualquer pessoa pode requisitar certidões sem ter que expressar o motivo do pedido. Estas são retiradas do conteúdo dos livros ou documentos das serventias.

2.1.3 FÉ PÚBLICA

Segundo o artigo terceiro da lei federal nº 8.935/94: "Notário ou tabelião, e o oficial

de registro, ou registrador, são profissionais do direito, dotados de fé pública, a quem é designado o exercício da atividade notarial e registral".

Segundo Ceneviva (1996), a fé pública afirma a certeza e a verdade dos assentamentos que o notário e o oficial de registro pratiquem e das certidões que emitam nessa condição. Ela assegura a eficácia dos atos jurídicos baseando-se no que é declarado ou praticado pelos registradores e notários.

2.1.4 SERVIÇOS EXTRA JUDICIAIS

Segundo Filho (2004), os serviços extra judiciais são serviços prestados por cartórios notariais e de registro que são os seguintes:

a) registro civil das pessoas naturais: responsável pelos registros das pessoas físicas;
b) registro civil das pessoas jurídicas: responsável pelos atos constitutivos e contratos;
c) registro de títulos e documentos: responsável pela conservação de documentos em caso de extravio e registro de contratos que criam obrigações entre as partes envolvidas;
d) registro de imóveis: responsável pelo registro da documentação de imóveis para assegurar o direito de propriedade;
e) tabelionato de notas e protestos: que procedem a intimação de cobrança de títulos vencidos e fornecem certidões de protesto utilizadas na análise de fichas cadastrais.

2.1.5 REGISTRO CIVIL DAS PESSOAS NATURAIS (RCPN)

Segundo Scochi (2004) o registro civil de nascimento possibilita o reconhecimento jurídico dos indivíduos e estabelece provas familiares perante a sociedade. O início da

cidadania de uma pessoa acontece através do assento de seu nascimento no registro civil das pessoas naturais. Esse registro público é o responsável pela existência das pessoas perante a sociedade civil.

A vida civil de uma pessoa começa e termina no RCPN, desde o momento que nasce até a hora de sua morte. Após o nascimento de uma pessoa, no decorrer da sua vida podem ser feitas várias anotações e averbações no seu registro. Essa pessoa pode obter emancipação dos pais, pode contrair matrimônio, pedir divórcio, ser interditada por impossibilidade de prática dos direitos civis, pode ter seu nome alterado, dentre outros casos. Todos os casos citados anteriormente resultam em alterações do registro da pessoa, isso significa que o RCPN representa o histórico de cada pessoa durante seu convívio na sociedade civil.

Durante a colonização do Brasil, a igreja católica foi responsável pelos registros vitais. Os registros eram mantidos em livros separados que tinham um formato padronizado conforme o conselho de Trento.

Em 1870, o estado passou a organizar esses registros, através da lei 1.829, criada pelo imperador D. Pedro II, que instituiu a Diretoria Geral de Estatística, órgão responsável pelos registros de nascimentos, casamentos e óbitos. A regulamentação dos registros ocorreu em 1888, através do decreto nº 9.886, passando por muitas alterações com o passar dos anos.

A principal função do registro civil das pessoas naturais é cuidar dos assuntos referentes às pessoas físicas. Ele é responsável por registrar os nascimentos, casamentos, conversões de uniões estáveis em casamento, óbitos, emancipações de menores de idade, interdições, sentenças declaratórias de ausência, opções de nacionalidade e sentenças que constituem vínculo de adoção. Além disso, são feitas averbações de sentenças judiciais de anulação de casamento, divórcio, reconhecimento de filhos ilegítimos e alterações ou correções de nome.

Os registros são armazenados em livros específicos, correspondentes a cada tipo de

registro. No livro “A” são registrados os nascimentos, no livro “B” os casamentos, o livro “B auxiliar” serve para registro de casamento religioso para efeitos civis. No livro “C” são registrados os óbitos e no “C auxiliar” os registros de crianças nascem sem vida ou natimortos. No livro “D” são feitos os registros de proclamas e no livro “E” as inscrições dos demais atos relativos ao estado civil.

O registro civil fornece as certidões de nascimento, casamento e óbitos, que podem ser solicitadas por qualquer pessoa em qualquer tempo, sem que seja questionado o motivo pelo qual é feita a solicitação. Muitos órgãos públicos utilizam relatórios fornecidos obrigatoriamente pelo registro civil. O IBGE necessita dos dados sobre nascimentos para cálculo de vários coeficientes para planejamento de atividades a serem desenvolvidas na área da saúde. O INSS recebe relatórios dos óbitos ocorridos para que com esses dados, possa cancelar o pagamento indevido de benefícios. Muitas outras instituições utilizam dados do registro civil, dentre elas estão a Receita Federal, Serviço Militar, Secretaria Estadual da Fazenda e Polícia Federal.

2.2 EXTENSIBLE MARKUP LANGUAGE (XML)

De acordo com Daconta (2003) a XML representa um conjunto de regras de sintaxe utilizadas para a criação de novas linguagens de marcação, chamadas de aplicações XML.

A XML foi desenvolvida pelo *XML Working Group* que pertence ao *World Wide Web Consortium* (W3C) e é uma tecnologia aberta para troca de dados. É uma linguagem de marcação descendente da *Standart Generalized Markup Language* (SGML) que possui as seguintes características:

a) poder de armazenamento e organização das informações em um formato adequado;

b) possui unicode como conjunto de caracteres padrão;

c) oferece várias maneiras de validação de documentos;

d) fácil entendimento para seres humanos e programas;

e) formato puro dos dados não dificulta as conversões de formato;

f) tem a capacidade de ter seu vocabulário extensível.

Os documentos XML são compostos apenas por dados puros, por isso os aplicativos que processam esses documentos decidem de qual forma eles serão apresentados. Esses aplicativos podem ser executados em *Personal Digital Assistants* (PDAs), telefones celulares, *Personal Computers* (PCs) ou qualquer outro tipo de dispositivo computacional.

Segundo Deitel et al (2003) a possibilidade de expansão da XML permite definição de novas linguagens de marcação dos mais variados tipos de dados. Existem alguns tipos variados de linguagens de marcação fundamentadas em XML, tais como, MathXML para matemática, VoiceML para reconhecimento de voz, *Chemical Markup Language* (CML) para química, entre outras.

“Uma vasta área de aplicação para XML está nas comunicações, mais especificamente na área de protocolos de nível de aplicação – protocolos que habilitam as aplicações a falar uma com as outras” (AHMED et al, 2001 p. 781, tradução nossa). Como desenvolvimento de protocolos modernos através da XML, aplicações heterogêneas que a utilizam podem comunicar-se independentemente da plataforma, banco de dados ou linguagem de programação na qual foram implementadas.

Para processar arquivos XML é necessário um analisador sintático para os documentos. Este analisador pode ser baseado em *Document Type Definition* (DTD) ou XML *Schemas*, para modelagem dos documentos. Ao utilizar um analisador sintático é garantida a boa formação dos documentos.

Um documento XML consiste de elementos e atributos que implementam uma

hierarquia em árvore. Por se basear em texto os documentos podem ser lidos tanto por seres humanos quanto por computadores. Um modelo de documento XML é mostrado no quadro 1.

```
<?xml version="1.0" ?>
- <BookOnCars xmlns="http://www.publishing.org" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.publishing.org ScopeTest.xsd">
  - <Chapter>
      <Title>My Car</Title>
    - <Section>
        <Title>Thunderbird</Title>
      </Section>
    </Chapter>
    <Title>A car</Title>
  </BookOnCars>
```

Quadro 1 – Modelo de documento XML

2.2.1 ELEMENTOS E ATRIBUTOS

De acordo com Skonnard (2002) os documentos XML são constituídos na sua maioria de elementos. Todo documento XML tem um elemento de nível mais alto conhecido como elemento raiz que serve de contêiner para todos os outros elementos. Os elementos têm nomes e podem ter outros elementos filhos.

Os elementos são serializados como um par de *tags*, uma *tag* de abertura (<) e uma *tag* de fechamento (>), onde entre essas *tags* é informado o nome do elemento, conhecido também como *tagname*. Os elementos filhos são serializados entre as *tags* do seu elemento ancestral. Quando os elementos não possuem elementos filhos, são ditos como elementos vazios.

A XML não define qualquer nome de elemento que fica a critério do desenvolvedor escolher os nomes da maneira que desejar, lembrando que a XML é *case-sensitive* e os nomes de elementos devem iniciar com uma letra ou um *underscore*.

O quadro 2 mostra o elemento *Person* com os elementos filhos *name* e *age*.

```
<Person>
  <name>Martin</name>
  <age>33</age>
</Person>
```

Fonte: Skonnard (2002, p. 2)

Quadro 2 – Elemento com elementos filhos

Segundo Ray (2001) os atributos permitem o detalhamento dos elementos. Podem ser usados para descrever propriedades de um elemento ou algum aspecto de comportamento do elemento. Um elemento pode conter inúmeros atributos que são separados por espaços em branco. Os atributos aparecem após o nome do elemento em qualquer ordem e seus valores devem estar entre apóstrofos (‘) ou aspas (“). O quadro 3 demonstra o elemento *Person* com os atributos *name* e *age*.

```
<Person name='Martin' age='33' />
```

Fonte: Skonnard (2002, p. 5)

Quadro 3 - Elemento com atributos

2.2.2 MODELAGEM DE DOCUMENTOS XML

De acordo com Ray (2001) a XML permite criar linguagens de marcação próprias onde são definidos elementos e atributos que se ajustam às necessidades de manter informações desejadas. Mas existe a necessidade de uma definição formal para restrição do vocabulário de elementos e controle de sua gramática, que pode ser obtida através da utilização de uma modelagem de documento.

A construção de um modelo de documento pode ser feita através do uso de *Document Type Definitions* (DTD) ou XML *Schemas*. Os DTDs consistem em regras para especificação das *tags* que podem ser utilizadas pelos documentos e o conteúdo das mesmas. Os XML

Schemas utilizam modelos baseados em XML para demonstrar a exibição dos documentos. Como os XML *Schema* são uma forma de XML eles podem ser editados com as mesmas ferramentas utilizadas para edição dos documentos.

2.3 XML SCHEMAS

"XML *Schemas* é uma linguagem de definição que permite restrição em documentos XML para vocabulários e estruturas hierárquicas específicas" (DACONTA, 2003, p. 37, tradução nossa).

De acordo com Vlist (2002) a XML *Schema* formaliza restrições, expressadas como regras ou modelo de estruturas que são aplicadas a documentos XML. A XML *Schema* foi desenvolvida para suprir as limitações dos DTDs. Ela oferece ao desenvolvedor muitas características e versatilidades na solução de problemas como:

a) capacidade de gerenciamento de dados melhorada por causa da habilidade de estender e restringir tipos de dados;
b) capacidade de criar tipos de dados definidos pelo usuário;
c) natureza intuitiva de objetos orientados para programadores orientados à objetos.

O documento XML do quadro 4 representa dados de um arquivo de biblioteca que descreve livros, seus autores e suas características.

```
<?xml version="1.0" ?>
- <BookStore xmlns="http://www.books.org" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.books.org BookStore.xsd">
  - <Book>
      <Title>My Life and Times</Title>
      <Author>Paul McCartney</Author>
      <Date>1998</Date>
      <ISBN>1-56592-235-2</ISBN>
      <Publisher>McMillin Publishing</Publisher>
    </Book>
  - <Book>
      <Title>Illusions The Adventures of a Reluctant Messiah</Title>
      <Author>Richard Bach</Author>
      <Date>1977</Date>
      <ISBN>0-440-34319-4</ISBN>
      <Publisher>Dell Publishing Co.</Publisher>
    </Book>
  - <Book>
      <Title>The First and Last Freedom</Title>
      <Author>J. Krishnamurti</Author>
      <Date>1954</Date>
      <ISBN>0-06-064831-7</ISBN>
      <Publisher>Harper & Row</Publisher>
    </Book>
  </BookStore>
```

Quadro 4 – Documento XML que utiliza um XML *Schema*

No quadro 5 é demonstrado o XML *Schema* que contém os elementos e atributos utilizados na confecção do documento XML anterior.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.books.org"
    targetNamespace="http://www.books.org" elementFormDefault="qualified">
  - <xsd:element name="BookStore">
    - <xsd:complexType>
      - <xsd:sequence>
          <xsd:element ref="Book" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  - <xsd:element name="Book">
    - <xsd:complexType>
      - <xsd:sequence>
          <xsd:element ref="Title" />
          <xsd:element ref="Author" />
          <xsd:element ref="Date" />
          <xsd:element ref="ISBN" />
          <xsd:element ref="Publisher" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="Title" type="xsd:string" />
    <xsd:element name="Author" type="xsd:string" />
    <xsd:element name="Date" type="xsd:string" />
    <xsd:element name="ISBN" type="xsd:string" />
    <xsd:element name="Publisher" type="xsd:string" />
  </xsd:schema>
```

Quadro 5 – XML Schema utilizado para validação de um documento XML

2.4 WEB SERVICES

De acordo com Daconta (2003) os *web services* representam aplicações de software que podem ser descobertos, descritos e acessados. Os *web services* baseados em XML e protocolos padrão web e utilizados na internet. Além da internet, pode-se utilizar *web services* em redes internas entre aplicativos internos e também em parcerias entre organizações através de pequenas soluções *Business-to-Business* (B2B).

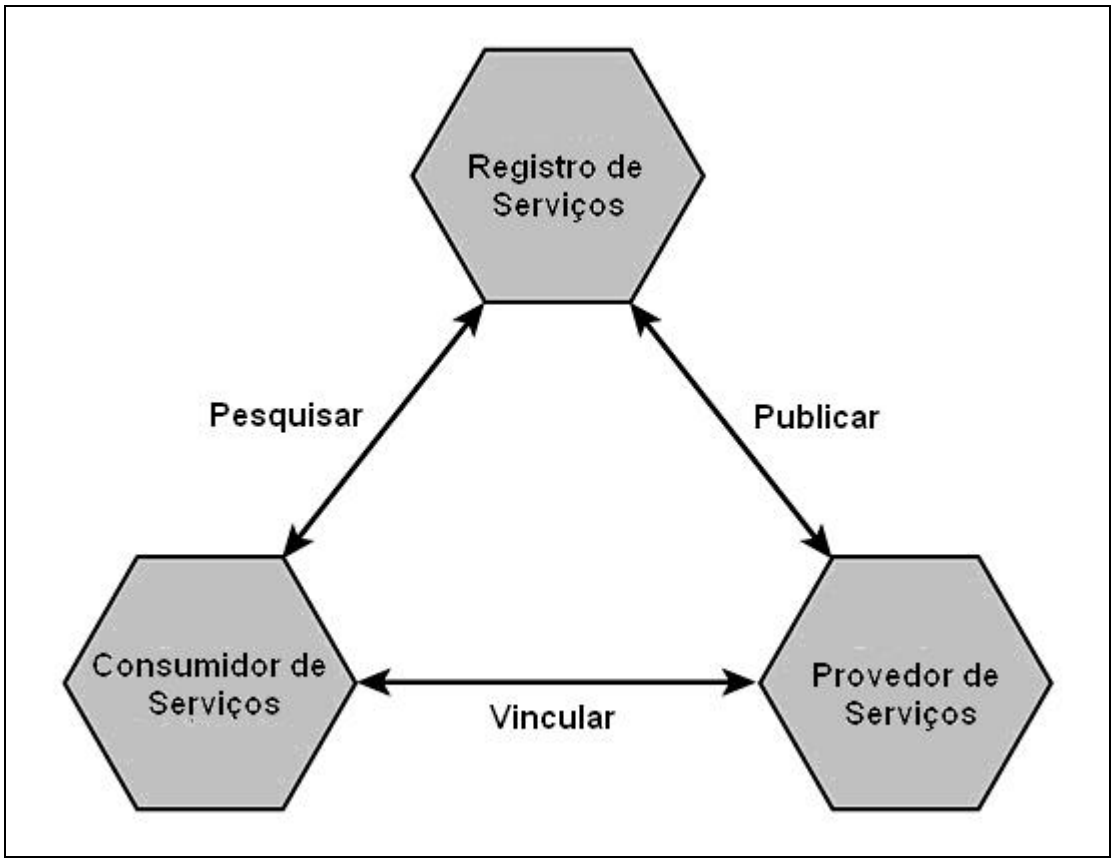
Segundo Neto (2004), os *web services* têm como principal característica a integração entre sistemas heterogêneos, utilizando padrões de protocolos independentes da plataforma e linguagem de programação em que foram desenvolvidos. Utilizam o *Hypertext Transfer Protocol* (HTTP) para transportar os serviços, mas podem utilizar outros protocolos tais como *File Transfer Protocol* (FTP) e *Simple Mail Transfer Protocol*(SMTP).

2.4.1 ARQUITETURA DE SERVIÇOS ORIENTADOS

De acordo com Graham et al (2002) com a evoluação tecnológica dos *web services*, percebeu-se o surgimento de um modelo, este modelo foi chamado de *Service-Oriented Architecture* (SOA), um simples conceito que é largamente utilizado em várias situações na qual os *web services* são utilizados.

Segundo Hendricks et al (2002) deve ser feita uma analogia sobre a arquitetura conceitual de *web services*, destacando papéis e operações. Os papéis são tipos de entidades distintos e operações atuam com funções realizadas por essas entidades, que conseqüentemente possibilitam o funcionamento do serviço.

A figura 1 demonstra a arquitetura dos serviços web que consiste em papéis e operações.



Fonte: GRAHAM (2002)

Figura 1 – Arquitetura de serviços orientados

Os principais papéis da arquitetura de serviços orientados são:

a) provedor de serviços: responsável pela descrição e publicação dos serviços web;
b) consumidor de serviços: tem a função de descobrir o serviço web, implementar sua interface de comunicação e consumir os serviços oferecidos;
c) registro de serviços: repositório que contém as informações sobre os serviços web.

As principais operações da arquitetura de serviços web são:

a) pesquisar: pesquisar os locais onde estão publicados os serviços web;
b) víncular: ligação entre o consumidor e o provedor de serviços;
c) publicar: liberar a publicação dos serviços web disponíveis.

2.4.1.1 CAMADAS

Segundo Hendricks et al (2002) os *web services* que são utilizam uma pilha de serviços dividida em camadas. A figura 2 mostra a pilha de camadas utilizadas pelos *web services*.

Figura 2 – Pilha de camadas dos *web services*

Os principais métodos utilizados nas camadas da pilha de *web services* são as seguintes:

a) *Web Service Description Language* (WSDL): utilizado na camada de descrição de serviços web. O WSDL é o responsável por fornecer o documento que contém a interface do serviço para que possa ser implementada pelos clientes;
b) *Universal Description, Discover, and Integration* (UDDI): atua na camada de descoberta, publica dados sobre as empresas fornecedoras e descreve os serviços para implementação da interface de comunicação por parte de seus clientes;
c) XML: atua na camada de dados, responsável pelo formato das mensagens;
d) SOAP: protocolo padrão para vínculo dos serviços web utilizado na camada de mensagens;
e) HTTP: atua na camada de transporte do ponto de vista de *web services*, utilizado

para encapsular as mensagens SOAP em requisições do tipo POST.

2.5 SOAP

De acordo com Hendricks et al (2002), SOAP é um protocolo superficial distribuído para a troca de informações em um ambiente distribuído. Utiliza XML baseado em texto diferentemente de outros protocolos que utilizam formatos binários. Por conter menos recursos o SOAP é um protocolo menos complexo do que outros protocolos utilizados em computação distribuída como *Common Object Request Broker Architecture* (CORBA), *Remote Method Invocation* (RMI) e *Distributed Component Object Model* (DCOM).

"SOAP é a sintaxe de envelope para envio e recebimento de mensagens XML com *web services*. Isto é, SOAP é o *envelope* que empacota as mensagens XML que são enviadas sobre HTTP entre clientes e *web services*." (DACONTA, 2003, p. 65).

2.5.1 MENSAGENS SOAP

As mensagens SOAP são representadas através de documentos XML e contém os seguintes elementos:

a) envelope SOAP: elemento raiz obrigatório de uma mensagem SOAP, define um documento XML como uma mensagem SOAP;
b) cabeçalho SOAP: elemento opcional, utilizado para adição de recursos de autenticação, gerenciamento de transações e serviços de pagamento;
c) corpo SOAP: elemento obrigatório, contém a mensagem SOAP destinada ao ponto final da mensagem.

A figura 3 demonstra as partes que compõem a estrutura de uma mensagem SOAP.

Figura 3 – Estrutura de mensagem SOAP

De acordo com Hendricks et al (2002) o protocolo SOAP apresenta muitas vantagens em comparação com outras aplicações de processamento distribuído. Algumas dessas vantagens são:

a) capacidade de transpor *firewalls* facilmente ao ser utilizado sobre o HTTP;

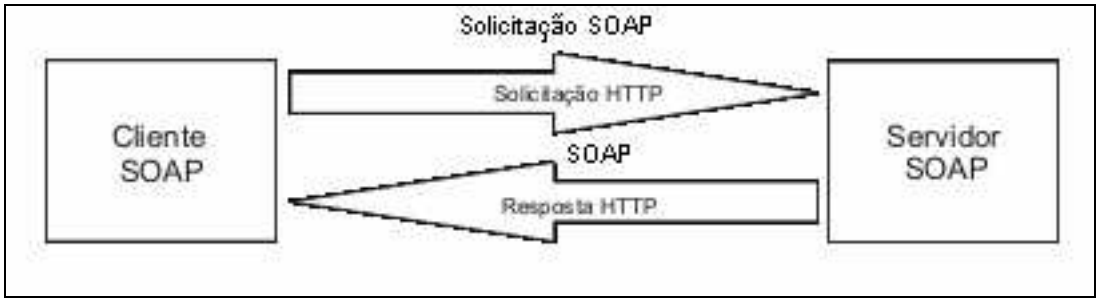
b) utiliza a estruturação de dados em XML;

c) é satisfatoriamente mapeado no padrão solicitação/resposta do HTTP;

d) superficial como um protocolo;

e) existe suporte para SOAP por muitos fornecedores como Microsoft, IBM e a Sun.

A figura 4 demonstra o processo de troca de mensagens utilizando SOAP e HTTP.



Fonte: Hendricks et al (2002)

Figura 4 – Troca de mensagens utilizando SOAP sobre HTTP

Na figura acima um cliente SOAP faz uma solicitação SOAP incorporada em uma

mensagem de solicitação HTTP. O servidor SOAP retorna uma mensagem de resposta SOAP incorporada em uma mensagem de resposta HTTP.

2.6 APACHE EXTENSIBLE INTERACTION SYSTEM (AXIS)

Segundo Hendricks et al (2002) o Axis representa o nome do projeto da versão 3.0 do SOAP Apache e tem o objetivo de estender as funcionalidades do SOAP Apache v2.0. Ele estende as funcionalidades como:

a) definição de um nó de mensagem que pode ser empacotado para utilização por clientes, servidores de serviço e intermediários;
b) suporta o protocolo XML para permitir camadas de protocolo de mensagens conectáveis;
c) fornecimento de um *framework* conectável para componentes como o transporte de objetos *listener, router, serializer/deserializer, dispatcher* e *handler*.

2.6.1 CARACTERÍSTICAS E VANTAGENS

O Axis possui algumas características e vantagens que são as seguintes:

a) velocidade: utiliza *Simple Api for* XML (SAX) ao invés de *Document Object Model* (DOM) que era utilizada na versão antiga do Apache SOAP. Com a utilização do SAX aumenta a velocidade de processamento de um documento XML;
b) flexibilidade: disponibiliza recursos para recursos afim de facilitar a inserção de novas extensões no mecanismo de processamento de cabeçalhos e gerenciamento de sistema;

c) organização orientada a componentes: apresenta o conceito de encadeamentos e controladores na implementação de padrões comuns de processamento de aplicações;
d) framework de transporte: oferecido ao fornecer remetentes e sintonizadores com o SOAP para vários protocolos como HTTP, FTP, dentre outros.

2.7 TRABALHOS CORRELATOS

Dantas (2001) implementou um protocolo criptográfico para emissão de certidões de nascimento através da internet em sua dissertação. Com o objetivo de dar agilidade aos novos registros de nascimento. O protocolo foi implementado utilizando redes de Petri e foi utilizado um protótipo de aplicativo para demonstração do funcionamento do protocolo.

Bortoli (2002) elaborou uma dissertação sobre a possibilidade do registro civil das pessoas naturais utilizar documentos eletrônicos para emissão e registro de documentos. O objetivo do trabalho foi a utilização do documento eletrônico para prática do registro civil e melhorar o atendimento ao público que utiliza o serviço dos cartórios.

Um trabalho utilizando XML para comunicação e armazenamento de dados da área contábil foi proposto por Kracik (2002). O objetivo principal foi o desenvolvimento de um protocolo de comunicação baseado nas tecnologias XML *Schema*, SOAP e *web services* para integrar sistemas contábeis heterogêneos. O trabalho consiste de um ambiente de transmissão de dados contábeis utilizando a arquitetura cliente/servidor. Foram desenvolvidos dois aplicativos, um servidor do serviço e um cliente para demonstrar o funcionamento do protocolo.

Colpani (2002) desenvolveu um protótipo de software para troca de dados entre aplicações de comércio eletrônico. Foi utilizado SOAP como base para intercâmbio dos

dados. O aplicativo desenvolvido utiliza um servidor HTTP utilizando *web services* e implementações do protocolo SOAP e XML.

Outro trabalho para troca eletrônica de dados é apresentado em Jesus (2004). Foi implementada a criação do sistema integrado de bibliotecas do sistema da Associação Catarinense das Fundações Educacionais (ACAFE), utilizando Java e XML. O objetivo principal do trabalho foi disponibilizar um catálogo coletivo dos acervos das bibliotecas do sistema ACAFE. Foi criado um sistema de recuperação centralizado dos acervos em uma *interface* web, para facilitar a localização física e geográfica dos materiais bibliográficos.

3 DESENVOLVIMENTO DO AMBIENTE PROPOSTO

O presente trabalho resultou na construção de um protótipo de protocolo de aplicação baseado em XML para troca de documentos entre cartórios que gerenciam os registros públicos referentes ao registro civil das pessoas naturais.

O protocolo é utilizado no fornecimento de *web services*, através da troca de mensagens entre um aplicativo servidor (provedor de serviços) e um aplicativo cliente (consumidor dos serviços). A troca de mensagens entre os aplicativos citados anteriormente baseia-se na implementação do protocolo SOAP.

A figura 5 mostra um exemplo de ambiente que será construído.

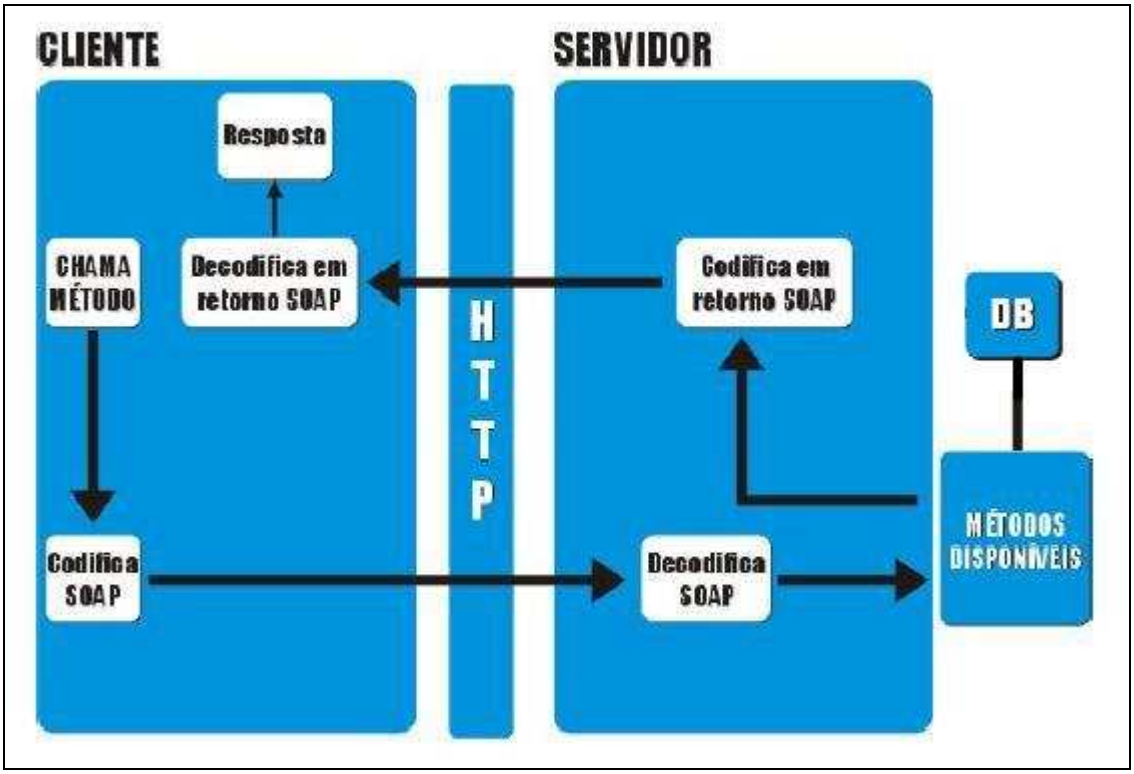


Figura 5 – Demonstração do ambiente cliente/servidor

Na seqüência serão apresentados os requisitos principais para solucionar o problema aqui explanado.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O objetivo do desenvolvimento desse trabalho é a criação de um protótipo de protocolo de aplicação (requisito funcional) para troca de documentos utilizados pelos cartórios de registro civil. O protocolo consiste em um XML *Schema* que define o formato e as restrições que existem em documentos do RCPN para a confecção de arquivos XML. Através da troca dos arquivos citados anteriormente é possível o intercâmbio dos sistemas dos cartórios de RCPN, onde pode-se consultar registros e trocar certidões.

O protocolo é utilizado na implementação de um *web service* (requisito não funcional) que disponibiliza seus métodos através requisições remotas invocadas por aplicativos clientes através do uso do protocolo SOAP. Os métodos são implementados em uma classe Java que fica residente no *framework* AXIS, responsável gerenciar as mensagens do protocolo SOAP.

O servidor de serviços fica disponível em um contêiner HTTP que com isso pode ser acessado por clientes remotos, que fazem consultas e solicitam certidões referentes a dados registrados em uma base SQL, neste caso no banco de dados *Firebird* utilizando acesso via *Java Database Connectivity*(JDBC).

É necessária a utilização do protocolo SOAP (requisito não funcional) para que seja feita a chamada de funções entre as partes. Com isso é possível realizar a integração entre sistemas construídos em plataformas distintas, pois o SOAP representa uma camada comum entre os aplicativos.

Com a utilização do arquivo RCPN.wsdl foi implementado um aplicativo cliente do *web service*, pois este arquivo contém a descrição dos métodos que devem ser implementados para obtenção dos dados.

As próximas seções descrevem a especificação e implementação do protocolo de aplicação proposto, do servidor e do cliente dos serviços utilizados para demonstração do

funcionamento do protocolo.

3.2 ESPECIFICAÇÃO

Nesta seção são apresentadas especificações do protótipo de protocolo de aplicação e dos aplicativos cliente e servidor, que serão utilizados para demonstração.

O protótipo de protocolo desenvolvido nesse trabalho visa atender a necessidade de integração de sistemas distintos de cartórios de registro civil. O protocolo é utilizado para acessar métodos remotos disponibilizados através de *web services* para consulta de registros e emissão de certidões. As entidades utilizadas pelo protocolo são especificadas através de diagramas de classes que representam os documentos que podem ser trocados entre usuários do protocolo de aplicação.

Para ilustrar a funcionalidade do protocolo são especificados casos de uso dos aplicativos cliente e servidor. Os casos de uso do servidor demonstram as respostas aos métodos invocados pelo aplicativo cliente, e os casos de uso do aplicativo cliente demonstram as solicitações feitas ao aplicativo servidor de serviços. São utilizados também diagramas de classes utilizados pelo servidor, e também os diagramas de seqüência que ilustram a funcionalidade de seus métodos.

Para especificação dos requisitos foram utilizadas as técnicas da *Unified Modeling Language* (UML) através da ferramenta *Poseidon for UML Community Edition 3.0*, para descrição dos casos de uso, diagramas de classes e diagramas de seqüências.

3.2.1 DIAGRAMA DE CLASSES DO PROTOCOLO DE APLICAÇÃO

O protocolo de aplicação consiste em classes que fazem referência aos documentos

utilizados pelo registro civil das pessoas naturais. As classes serão apresentadas através de diagramas UML. O XML *Schema* que especifica o modelo de documentos XML do protocolo encontra-se no Apêndice A. O diagrama de classes da figura 6 mostra as classes principais utilizadas pelo protocolo de aplicação.

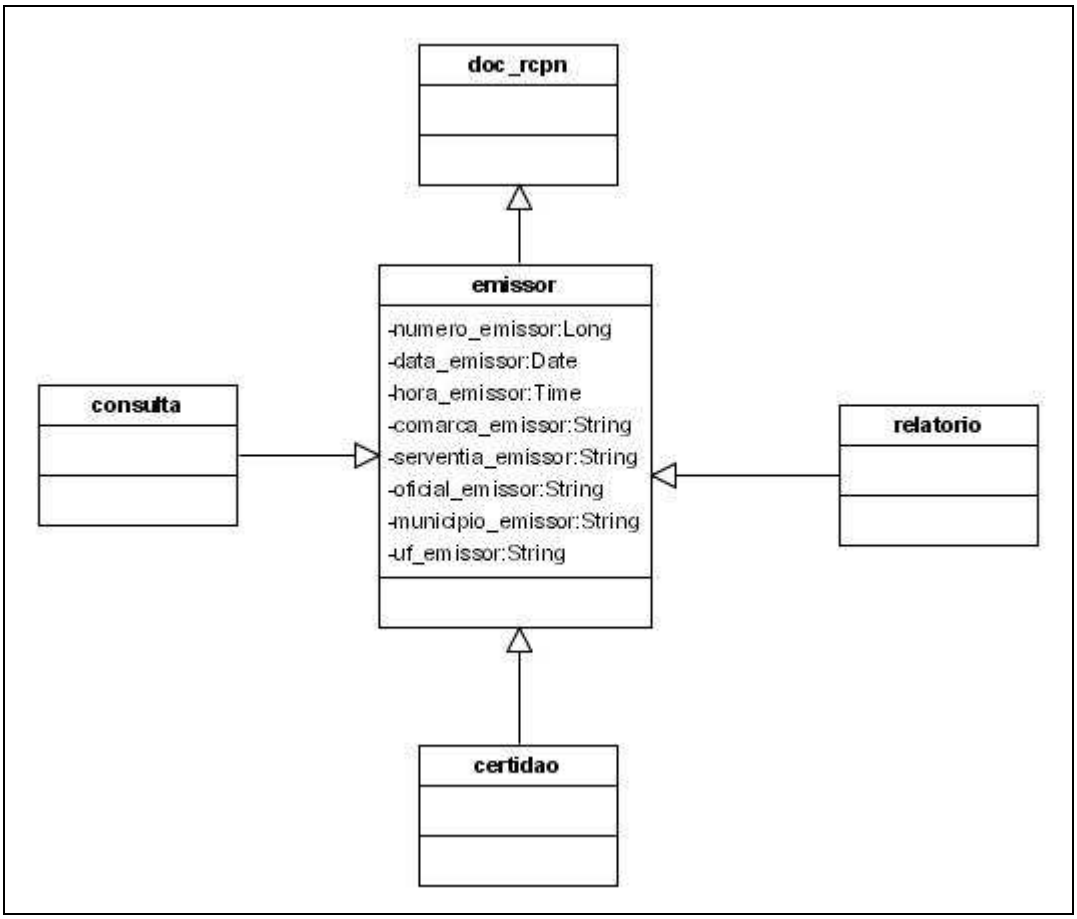


Figura 6 – Visão geral das classes principais do protocolo de aplicação

As principais classes mostradas na figura 6 são as seguintes:

A classe *doc_rcpn* é a classe abstrata ancestral dos documentos referentes ao registro civil das pessoas naturais modelados pelo protocolo de aplicação.

A classe *emissor* é uma especialização da classe *doc_rcpn* e representa os dados do cartório responsável pelo fornecimento dos documentos. Esta classe contém os seguintes atributos:

a) *numero_emissor* – número único gerado para cada documento;

b) *data_emissor* – data de emissão do documento;

c) *hora_emissor* – hora de envio do documento;

d) *comarca_emissor* – nome da comarca a qual pertence o cartório responsável pela emissão do documento;

e) *serventia_emissor* – nome do cartório que emite o documento;

f) *oficial_emissor* – nome do notário ou registrador responsável pelo cartório;

g) *municipio_emissor* – nome do município onde encontra-se o cartório emissor do documento;

h) *uf_emissor* – unidade da federação onde localiza-se o município do cartório.

No modelo é observada a existência de três classes que são especializações da classe *emissor*, que são as classes *consulta*, *certidao* e *relatorio*. Estas classes são mostradas em detalhes nas próximas seções.

3.2.1.1 A CLASSE CONSULTA

A classe *consulta* representa informações sobre registros que são encontrados através de uma busca. Ela possui três especializações que são *consulta_nascimento*, *consulta_casamento* e *consulta_obito*. Estas classes representam dados de registros encontrados em livros distintos do registro civil. A figura 7 apresenta o diagrama com a classe *consulta* e suas classes subordinadas.

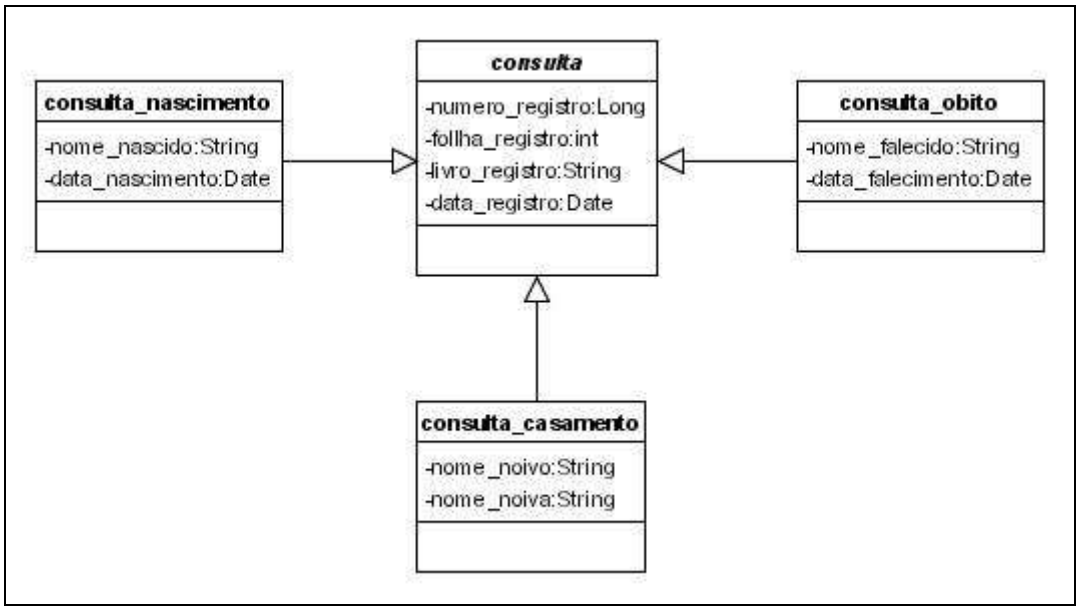


Figura 7 – Classe *consulta* e especializações

A classe *consulta_nascimento* representa uma consulta de registro de nascimento e é composta dos seguintes atributos:

a) *nome_nascido* – nome da criança registrada;

b) *data_nascimento* – data de nascimento da criança;

c) *hora_nascimento* – hora de nascimento da criança;

d) *numero_registro* – número do registro da criança;

e) *folha_registro* – número da folha do livro de registro de nascimentos onde consta o registro da criança;

f) *livro_registro* – número do livro de registro de nascimentos onde consta o registro da criança;

g) *data_registro* – data de registro do nascimento da criança.

A classe *consulta_casamento* representa dados sobre registros de casamentos e consiste nos seguintes atributos:

a) *nome_noivo* – nome do noivo do casamento;

b) *nome_noiva* – nome da noiva do casamento;

c) *numero_registro* – número do registro de casamento;

d) *folha_registro* – número da folha do livro de registro de casamentos onde consta o registro do matrimônio;

e) *livro_registro* – número do livro de registro de casamentos onde consta o registro do matrimônio;

f) *data_registro* – data do casamento.

A classe *consulta_obito* representa dados sobre registros de óbito e é composta dos atributos a seguir:

a) *nome_falecido* – nome da pessoa que faleceu;

b) *data_falecimento* – data de falecimento da pessoa;

c) *numero_registro* – número do registro de óbito;

d) *folha_registro* – número da folha do livro de registro de óbitos onde consta o registro do óbito;

e) *livro_registro* – número do livro de registro de óbitos onde conta o registro de óbito.

3.2.1.2 A CLASSE CERTIDAO

A classe *certidao* representa um documento que comprova a existência de um registro nos livros dos cartórios. Esta classe serve como base para certidões de tipos distintos e é demonstrada na figura 8.

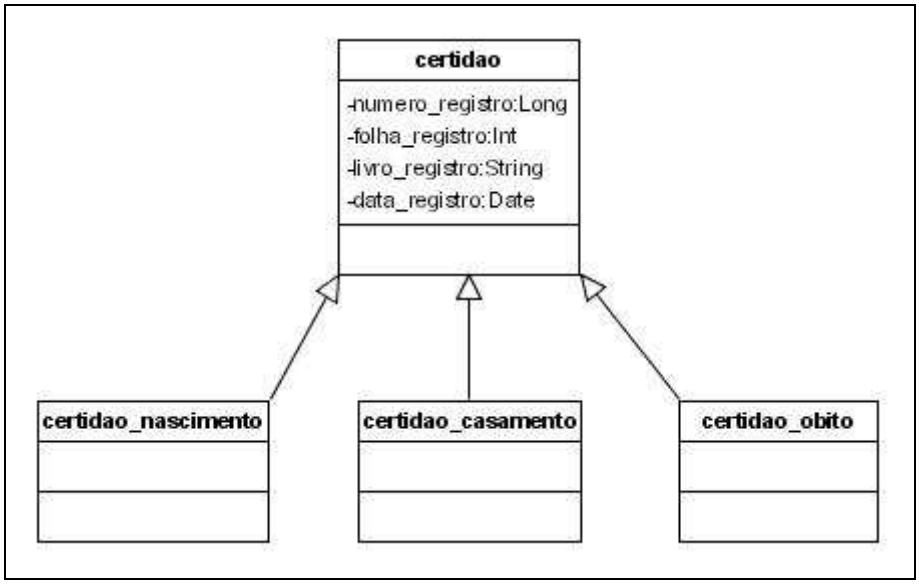


Figura 8 – Classe *certidao* e suas especializações

A classe que representa documentos do tipo certidão é composta pelos seguintes atributos:

a) *numero_registro* – número do registro que consta no livro de registros específico (nascimentos, casamentos ou óbitos);

b) *folha_registro* – número da folha onde consta o registro específico;

c) *livro_registro* – número do livro onde consta o registro específico;

d) *data_registro* – data do registro específico.

3.2.1.2.1 A CLASSE CERTIDAO_NASCIMENTO

A classe *certidao_nascimento* é uma especialização da classe *certidao* e consiste em dados correspondentes a um registro de nascimento contido no livro de registro de nascimentos do registro civil. As classes que compõem a classe *certidao_nascimento* são mostradas na figura 9.

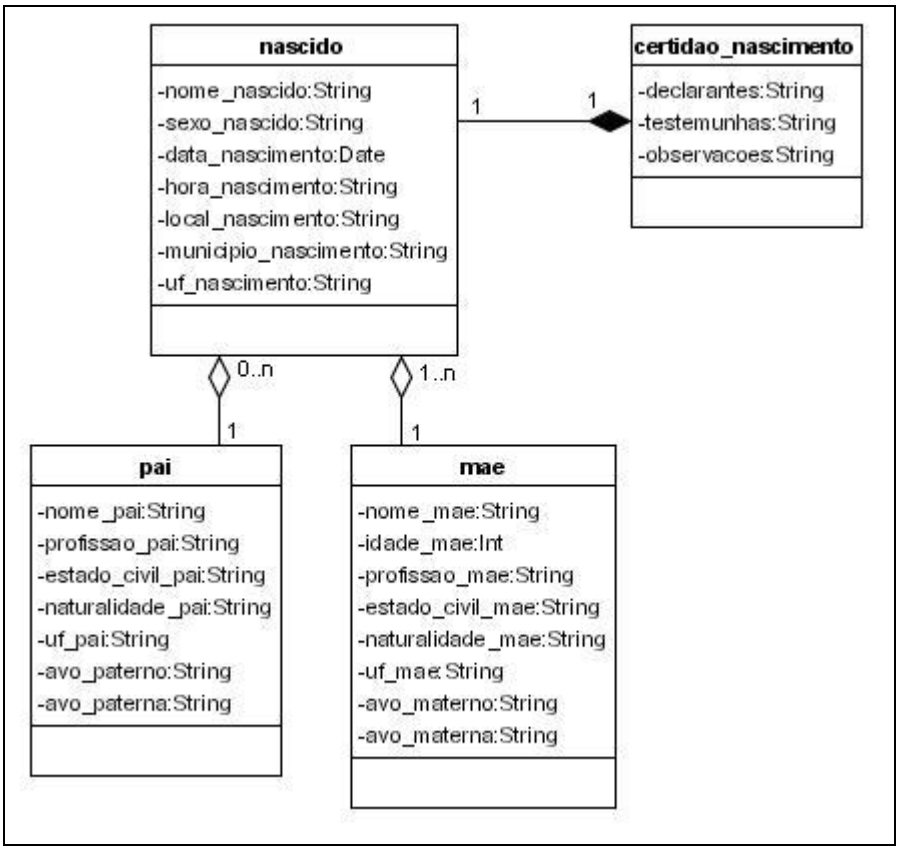


Figura 9 – Classe *certidao_nascimento* e outras classes que a compõem

A classe *certidao_nascimento* possui os seguintes atributos:

a) *declarantes* – nome da pessoa ou pessoas que declaram o nascimento;

b) *testemunhas* – nome das pessoas que testemunharam o nascimento ou que podem confirmar a existência de uma criança;

c) *observacoes* – observações referentes ao registro de nascimento.

A classe *nascido* é uma composição da classe *certidao_nascimento* e representa os dados de nascimento de uma criança e. Esta classe possui os seguintes atributos:

a) *nome_nascido* – nome da criança registrada;

b) *sexo_nascido* – descrição do sexo da criança;

c) *data_nascimento* – data de nascimento da criança;

d) *hora_nascimento* – hora de nascimento da criança;

e) *local_nascimento* – local onde ocorreu o nascimento da criança (maternidade, hospital, entre outros);

f) *municipio_nascido* – município onde ocorreu o nascimento;

g) *uf_nascido* – unidade da federação a qual pertence o município onde ocorreu o nascimento.

A classe *nascido* possui duas classes agregadas que são as classes *pai* e *mae*. A classe *pai* é composta pelos seguintes atributos:

a) *nome_pai* – nome do pai da criança;

b) *profissao_pai* – profissão exercida pelo pai na data do nascimento da criança;

c) *estado_civil_pai* – descrição do estado civil do pai da criança;

d) *naturalidade_pai* – nome do município onde nasceu o pai da criança;

e) *uf_pai* – unidade da federação a qual pertence o município de nascimento do pai da criança;

f) *avo_paterno* – nome do avô paterno da criança;

g) *avo_paterna* – nome da avó paterna da criança.

A classe *mae* consiste nos seguintes atributos:

a) *nome_mae* – nome da mãe da criança;

b) *idade_mae* – idade da mãe na data do nascimento da criança;

c) *profissao_mae* – profissão exercida pela mãe na data de nascimento da criança;

d) *estado_civil_mae* – descrição do estado civil da mãe da criança;

e) *naturalidade_mae* – nome do município onde nasceu a mãe da criança;

f) *uf_mae* – unidade da federação a qual pertence o município de nascimento da mãe da criança;

g) *avo_materno* – nome do avô materno da criança;

h) *avo_materna* – nome da avó materna da criança.

3.2.1.2.2 A CLASSE CERTIDAO_CASAMENTO

A classe *certidao_casamento* representa um registro contido no livro de registro de casamentos do registro civil. A figura 10 demonstra as classes que representam uma certidão de casamento.

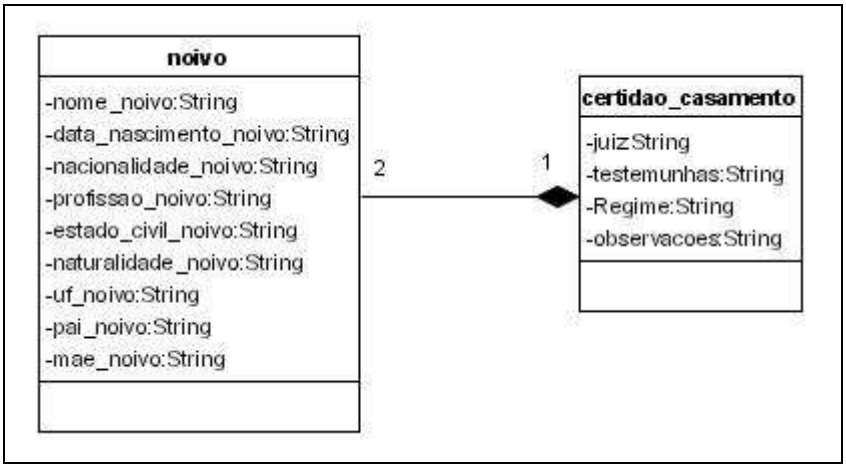


Figura 10 – Classes que representam uma certidão de casamento

A classe *certidao_nascimento* contém os seguintes atributos:

a) *juiz* – nome do juiz de paz que realizou o casamento;

b) *testemunhas* – nome das testemunhas do casamento;

c) *regime* – descrição do regime escolhido para realização do casamento;

d) *observacoes* – observações do casamento.

A classe *noivo* representa os noivos do casamento e compõe a classe *certidao_casamento*. Os atributos da classe *noivo* são os seguintes:

a) *nome_noivo* – nome do noivo ou da noiva do casamento;

b) *data_nascimento* – data de nascimento do noivo ou da noiva;

c) *nacionalidade_noivo* – descrição da nacionalidade do noivo ou da noiva;

d) *profissao_noivo* – profissão do noivo ou da noiva;

e) *estado_civil_noivo* – descrição do estado civil do noivo ou da noiva;

f) *naturalidade_noivo* – nome do município de nascimento do ou da noiva;

g) *uf_noivo* – unidade da federação a qual pertence o município de nascimento do noivo ou da noiva;

h) *pai_noivo* – nome do pai do noivo ou da noiva;

i) *mae_noivo* – nome da mãe do noivo ou da noiva.

3.2.1.2.3 A CLASSE CERTIDAO_OBITO

A classe *certidao_obito* representa uma certidão baseada em dados de um registro de óbito. A classe é composta pela agregação de outras classes que representam partes constantes de uma certidão de óbito.

Esta classe é composta pela agregação de outras classes demonstradas na figura 11.

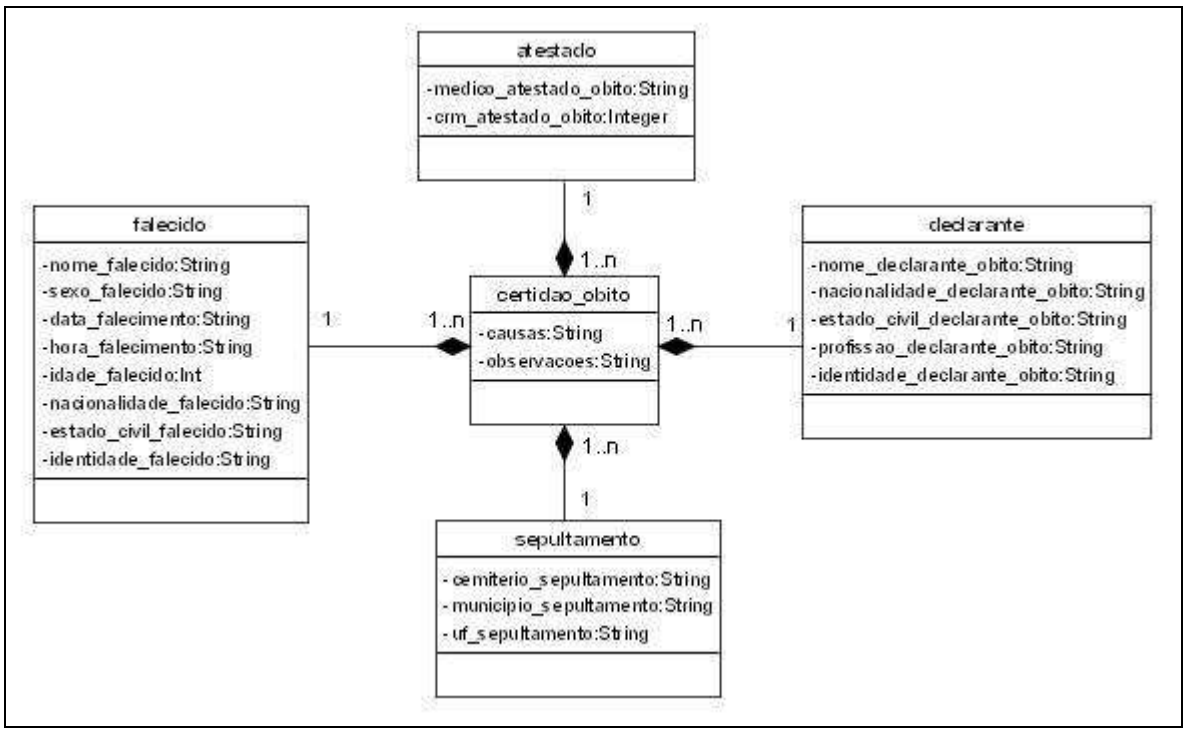


Figura 11 – Classe *certidao_obito* e suas classes agregadas

A classe *certidao_obito* contém os seguintes atributos:

a) *causas* – descrição das causas do óbito;

b) *observacoes* – observações relacionadas ao registro do óbito.

Na classe *certidao_obito* existem quatro classes agregadas que são as classes *falecido* que contém dados sobre a pessoa que faleceu, a classe sepultamento que detém os dados sobre o local do sepultamento da pessoa falecida, a classe *atestado* que consiste em dados sobre o médico que atestou o óbito e finalmente a classe *declarante*, que representa os dados da pessoa que levou as informações do óbito para serem registradas. A seguir são detalhadas as classes agregadas que compõem a classe *certidao_obito*.

A classe *falecido* é descrita com os seguintes atributos:

a) *nome_falecido* – nome da pessoa que consta no registro de óbito;
b) *sexo_falecido* – descrição do sexo da pessoa que faleceu;
c) *data_falecimento* – data em que ocorreu o óbito;
d) *hora_falecimento* - hora em que ocorreu o óbito;
e) *idade_falecido* – idade da pessoa que faleceu;
f) *nacionalidade_falecido* – descrição da nacionalidade da pessoa que faleceu;
g) *estado_civil_falecido* – descrição do estado civil da pessoa que faleceu;
h) *identidade_falecido* – descrição de documento de identificação da pessoa que faleceu.

A classe *sepultamento* contém os seguintes atributos:

a) *cemiterio_sepultamento* – nome do cemitério onde ocorreu o sepultamento;
b) *municipio_sepultamento* – nome do município onde ocorreu o sepultamento;
c) *uf_sepultamento* – unidade da federação a qual pertence o município onde ocorreu o sepultamento.

A classe *atestado* possui os seguintes atributos:

a) *medico_atestado_obito* – nome do médico que emitiu o atestado de óbito;
b) *crm_atestado_obito* – número de cadastro do médico no Conselho Regional de Medicina.

Os atributos da classe *declarante* são os seguintes:

a) *nome_declarante_obito* – nome da pessoa que declarou o óbito;

b) *nacionalidade_declarante_obito* – descrição da nacionalidade do declarante que declarou o óbito;

c) *estado_civil_declarante_obito* – descrição do estado civil do declarante do óbito;

d) *profissao_declarante_obito* – profissão do declarante do óbito;

e) *identidade_declarante_obito* – documento de identidade do declarante do óbito.

3.2.1.3 A CLASSE RELATORIO

A classe *relatorio* demonstrada no diagrama de classes da figura 6 é citada apenas como uma forma de extensão do protocolo de aplicação. Esta classe não é implementada nesse trabalho. O protocolo de aplicação pode ser estendido para outros documentos como editais de proclamas, notificações extrajudiciais entre outros.

3.2.2 MÓDULO SERVIDOR

O módulo servidor corresponde ao *web service* e gerencia o acesso a base de dados de registros. Consiste em uma classe que contém os métodos que são acessados pelo módulo cliente. Utiliza a classe agregada JDOM para confecção de documentos baseados no protocolo de aplicação. Na figura 12 mostra o diagrama de classes do módulo servidor.

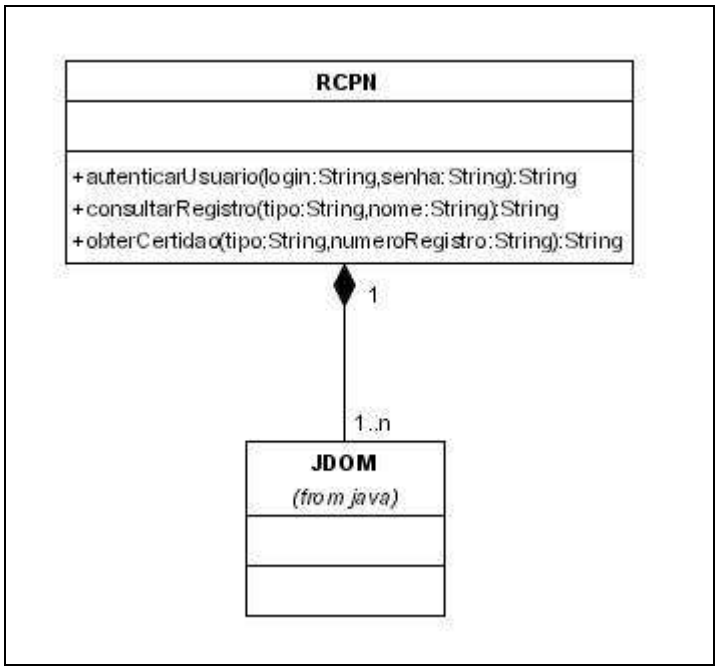


Figura 12 – Classes utilizadas pelo módulo servidor

Os documentos criados servem de retorno à invocação dos métodos feita pelos clientes do *web service*. O retorno desses métodos corresponde à autenticação de usuários, lista de registros encontrados e envio de documentos solicitados respectivamente.

A utilização do módulo servidor pode ser compreendida através dos diagramas de caso de uso mostrados na seção 3.2.4.

3.2.3 MÓDULO CLIENTE

O módulo cliente equivale a um consumidor de serviços disponibilizados pelo módulo servidor. Consiste em classes que fazem chamadas aos métodos do módulo servidor e utiliza o retorno desses para emitir documentos do tipo certidão. A figura 13 mostra as classes utilizadas no módulo cliente.

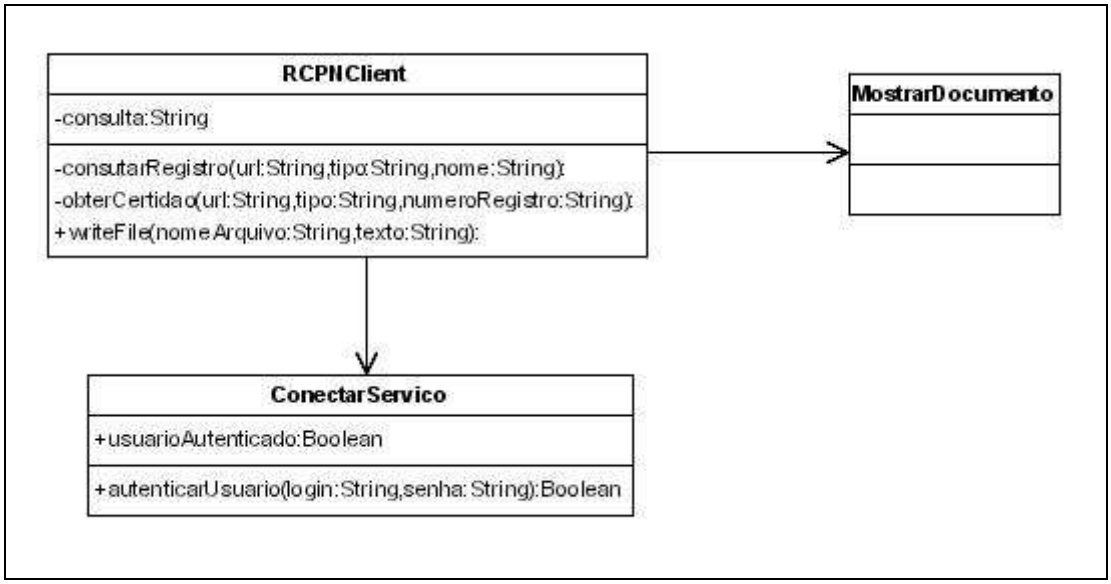


Figura 13 – Classes utilizadas pelo módulo cliente

Os diagramas de caso de uso da seção 3.2.4 exemplificam a utilização do módulo cliente para acesso aos métodos do *web service*.

3.2.4 CASOS DE USO E DIAGRAMAS DE SEQÜÊNCIA DOS MÓDULOS CLIENTE E SERVIDOR

A utilização do protocolo de aplicação pode ser demonstrada através de alguns casos de uso e diagramas de seqüência. Estes representam as ações realizadas entre os módulos cliente e servidor utilizados na demonstração do protocolo de aplicação. A figura 14 mostra os casos de uso utilizados entre cliente e servidor do protocolo.

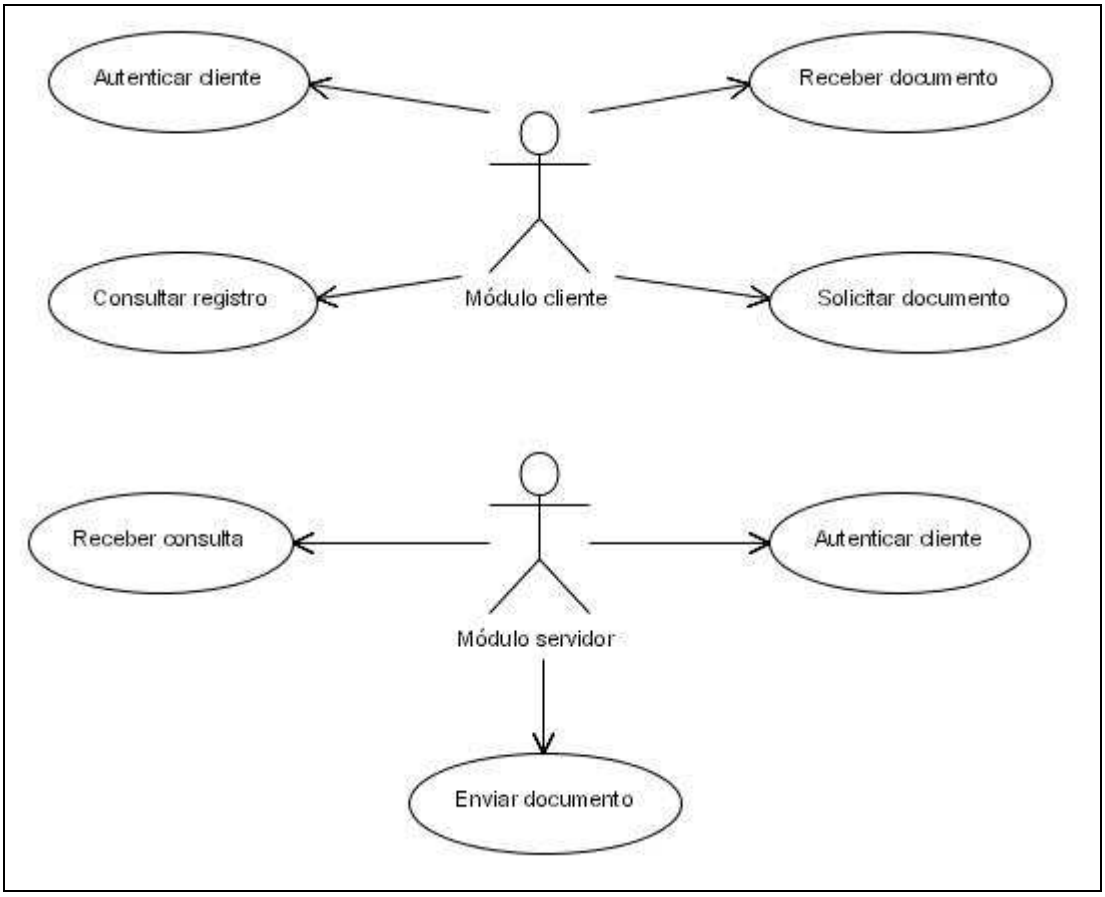


Figura 14 – Casos de uso do protocolo de aplicação

Os casos de uso mostrados na figura 14 demonstram a utilização do protocolo através dos módulos cliente e servidor que são utilizados na implementação de um *web service* para realizar o intercâmbio de dados entre os aplicativos.

O caso de uso *Autenticar cliente* do módulo cliente representa uma forma de identificação do usuário do protocolo. Através dessa autenticação de usuários é possível o acesso à base de dados que contém os registros do cartório. A figura 15 demonstra o diagrama de seqüência do caso de uso *Autenticar cliente.*

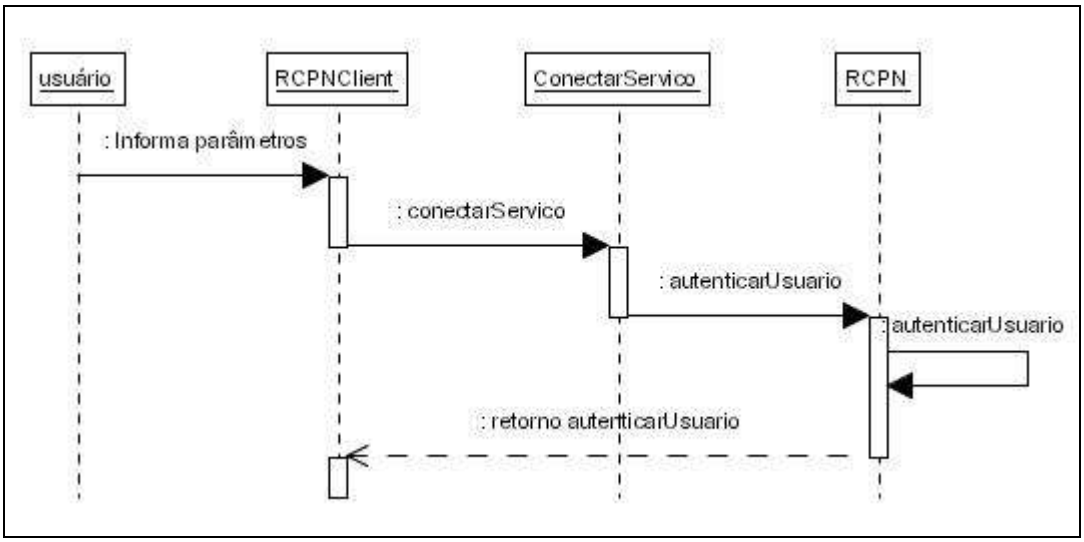


Figura 15 – Diagrama de seqüência *Autenticar cliente*

A possibilidade de consulta aos registros é demonstrada no caso de uso *Consultar registros* do módulo cliente, onde o usuário pode buscar dados sobre registros de nascimento, casamento ou óbito. O usuário informa o tipo de registro e o nome da pessoa que consta no registro que deseja localizar. A figura 16 mostra o diagrama de seqüência *Consultar registros*.

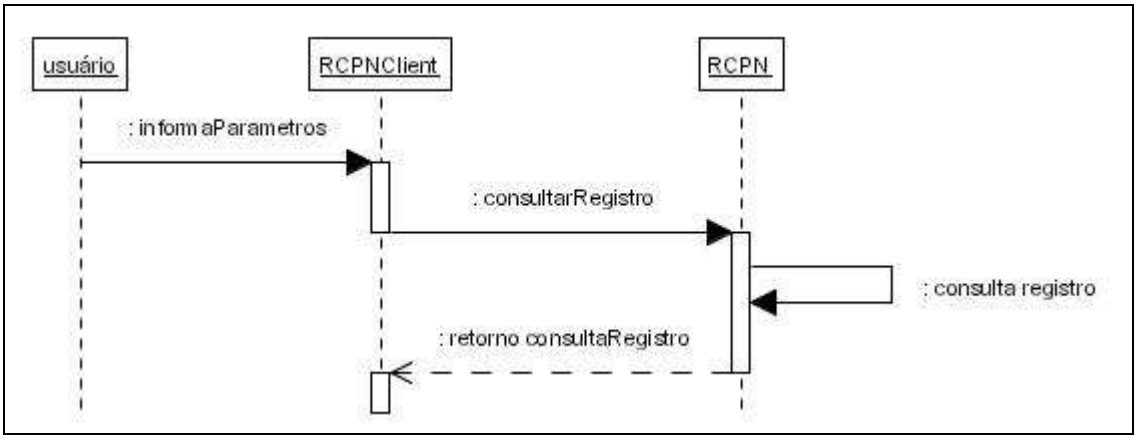


Figura 16 -Diagrama de seqüência *Consultar registros*

O usuário do módulo cliente pode solicitar documentos representados pelo caso de uso *Solicitar documento*, onde ele passa o tipo e o número do registro desejado. A figura 17 mostra o diagrama de seqüência do caso de uso *Solicitar documento*.

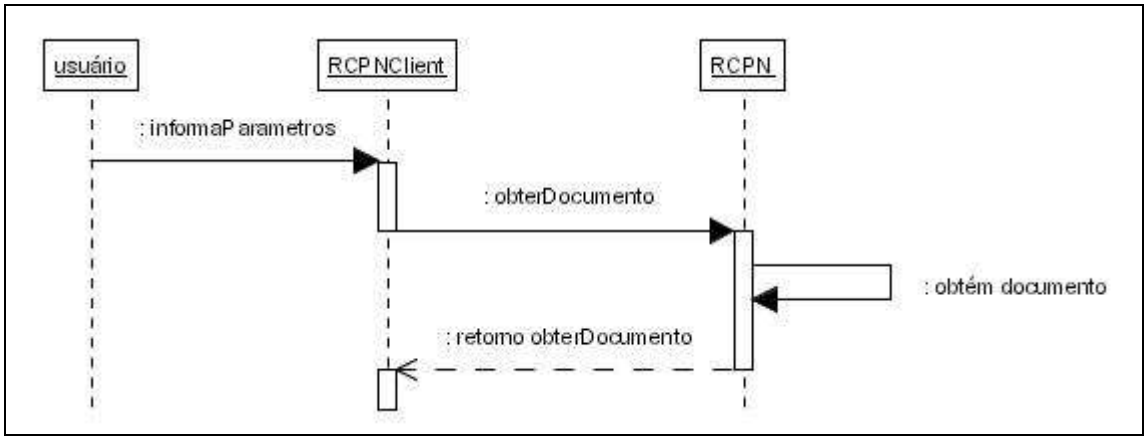


Figura 17 – Diagrama de seqüência *Solicitar documento*

E finalmente receber documentos que são formatados de acordo com o protocolo de aplicação, essa interação se dá através do caso de uso *Receber documento* do módulo cliente. O cliente do sistema recebe certidões de nascimento, casamento ou óbito.

Os casos de uso do módulo servidor representam a execução dos métodos invocados pelo módulo cliente. O caso *Autenticar cliente* representa uma validação do nome e senha do usuário que deseja utilizar os métodos do *web service*. Após a autenticação do cliente do serviço pode-se fazer buscas ou solicitar certidões.

As consultas recebidas pelo módulo servidor são representadas pelo caso de uso *Receber consulta*, nesse caso é invocado o método de busca, feita pelo cliente do serviço na base de dados de registros, retornando uma lista dos registros encontrados.

O caso de uso *Enviar documento* representa o envio de uma certidão que foi solicitada por um cliente do serviço. É montado o documento baseado nos parâmetros passados pelo cliente do serviço e enviado para o mesmo.

3.3 IMPLEMENTAÇÃO

Nesta seção são discutidos todos os aspectos relacionados ao desenvolvimento do protocolo de aplicação. Além disso, os aplicativos cliente e servidor também têm seus aspectos técnicos demonstrados, pois, eles são partes integrantes deste trabalho.

3.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

Para desenvolvimento do protocolo de aplicação foi utilizada a ferramenta *Altova* XML *Spy 2005*, para confecção dos XML *Schemas*. A ferramenta foi útil na definição dos tipos de elementos usados no protocolo.

O quadro 6 mostra um exemplo de um elemento de tipo complexo criado pela ferramenta.

```
<xs:complexType name="ct_emissor">
  <xs:annotation>
    <xs:documentation>Tipo complexo de descrição do emissor do documento</xs:documentation>
  </xs:annotation>
  <xs:attribute name="numero_emissor" type="xs:long" use="required"/>
  <xs:attribute name="data_emissor" type="xs:date" use="required"/>
  <xs:attribute name="hora_emissor" type="xs:time" use="required"/>
  <xs:attribute name="comarca_emissor" type="xs:string" use="required"/>
  <xs:attribute name="serventia_emissor" type="xs:string" use="required"/>
  <xs:attribute name="oficial_emissor" type="xs:string" use="required"/>
  <xs:attribute name="municipio_emissor" type="xs:string" use="required"/>
  <xs:attribute name="uf_emissor" type="xs:string" use="required"/>
</xs:complexType>
```

Quadro 6 – Elemento de tipo complexo criado pela ferramenta *Altova* XML *Spy 2005*

Na implementação do módulo servidor do *web service* foram utilizadas as seguintes ferramentas:

a) *Jakarta Tomcat 5.0*: contêiner HTTP, utilizado para armazenar o *framewok* Axis;

b) Axis 1.1: *framework* que implementa funções do SOAP, armazena as classes que fornecem os serviços e gerencia a troca de mensagens;

c) banco de dados *Firebird*: armazena base de dados utilizada para consulta e obtenção dos documentos.

O servidor é o responsável por disponibilizar o *web service* aos clientes que desejarem utilizar seus métodos. O *web service* consiste em uma classe Java que contém os métodos, esses métodos são invocados através de mensagens SOAP enviadas pelo aplicativo cliente.

Ao receber uma requisição SOAP o servidor faz a validação do login e senha do usuário do serviço. Após isso é então liberado o acesso ao método que está implícito no corpo da mensagem.

O aplicativo servidor consiste na configuração de um ambiente que tem como base o contêiner HTTP *Jakarta Tomcat*. Dentro do contêiner devem ser instalados os arquivos que pertencem ao Axis. No Axis são colocadas as classes que contém os métodos que serão acessados remotamente por clientes do serviço. Essas classes conectam-se em uma base de dados *Firebird*, de onde trazem registros armazenados. A base de dados citada anteriormente não representa a base de dados utilizada pelo sistema do cartório. Os dados contidos na base utilizada pelo protocolo são gerados a partir dos dados do sistema de RCPN do cartório, representando assim uma replicação de dados. Na figura 18 é mostrado o modelo de dados que contém as tabelas utilizadas pelo protocolo.

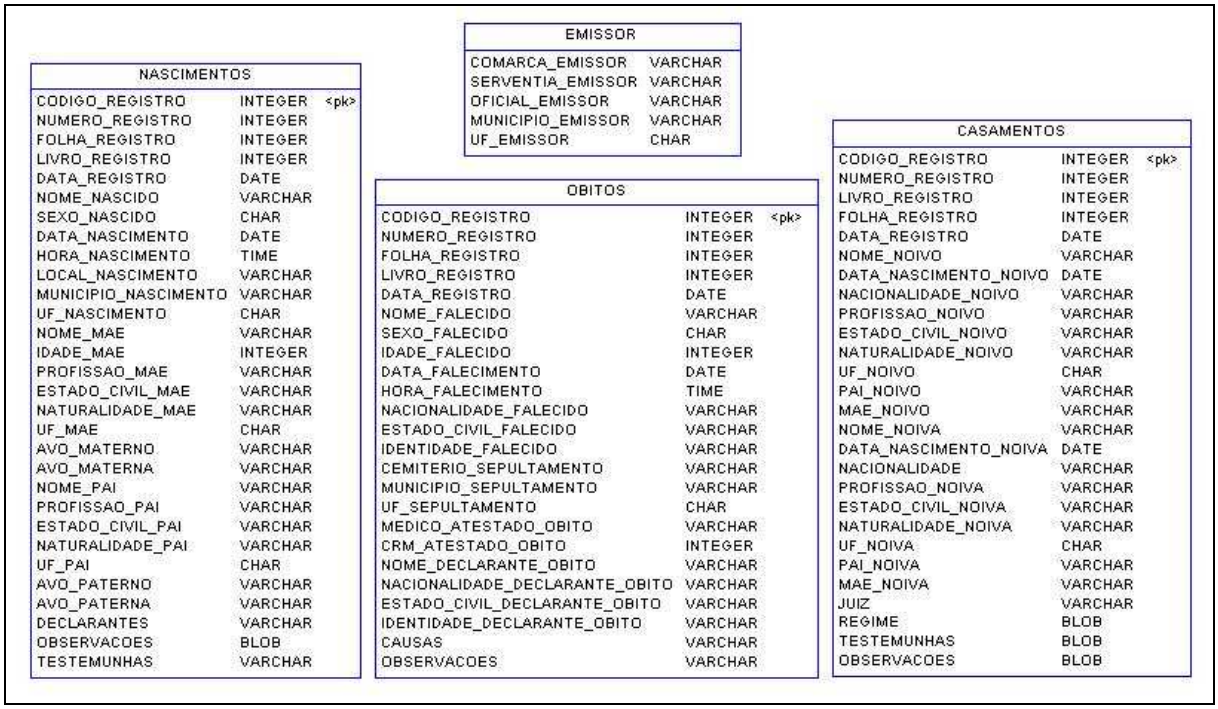


Figura 18 – Modelo de dados utilizado pelo protocolo

Para que o *web service* esteja disponível é preciso inicializar o *Jakarta Tomcat*.

Pelo fato de ser um ambiente configurado, o servidor não possui telas gráficas, apenas a classe que é disponibilizada para acesso pelos clientes do *web service*.

No aplicativo cliente foi utilizada a plataforma de desenvolvimento *Eclipse 3.0* em conjunto com a linguagem de programação Java. Esse aplicativo foi desenvolvido baseando-se no arquivo RCPN.wsdl, que contém a descrição dos métodos e parâmetros disponíveis para acesso ao *web service*. O conteúdo do arquivo RCPN.wsdl encontra-se no apêndice B.

O aplicativo cliente realiza consultas e solicitação de certidões através da invocação dos métodos do servidor. O quadro 7 mostra o método de autenticação de usuários pelo cliente.

```
//método do módulo cliente para autenticação de usuário do web service
private boolean autenticarUsuario(String login, String senha){
    boolean resultado = false;
    try {
        Service service = new Service();
        Call call = (Call) service.createCall();
        call.setTargetEndpointAddress( new java.net.URL(ConectarServico.this.url));
        call.setOperationName("autenticarUsuario");
        String response = (String) call.invoke(new Object[]{login,senha});
        if (response.equals("true")){
            resultado = true;
        }

    } catch (Exception e) {
        JOptionPane.showMessageDialog( null, "Não foi possível conectar ao serviço!!",
                                       "Conexão RCPN", JOptionPane.ERROR_MESSAGE);
    }
    return resultado;
}
```

Quadro 7 – Método para autenticação de usuários do *web service*

A interface do aplicativo cliente consiste em uma tela para informação dos dados que são descritos abaixo:

a) *provedor de serviços* – representa o endereço onde está localizado o servidor dos serviços;

b) *armazenar documentos em* – representa o local onde devem ser salvos os documentos XML enviados pelo servidor;

c) *procura por* – campo onde devem ser informados os nomes para consulta de registros;

d) *resultados* – grade que reúne a lista de registros encontrados através da consulta.

Na figura 19 é apresentado à interface gráfica do aplicativo cliente do *web service*.

Figura 19 – Interface do aplicativo cliente

3.3.2 IMPLEMENTAÇÃO DO PROTOCOLO DE APLICAÇÃO

A implementação do protocolo de aplicação consiste em um arquivo XML *Schema*, chamado doc_rcpn.xsd. O esquema contém uma estrutura XML que utiliza tipos complexos na representação dos objetos que compõem documentos utilizados no RCPN. Alguns exemplos de tipos criados são *ct_emissor* que representa dados do cartório emissor dos documentos, *ct_nascido* que modela os atributos de uma criança em uma certidão de nascimento entre outros. O quadro 7 mostra o tipo complexo *ct_emissor*.

```
<xs:complexType name="ct_emissor">
  <xs:annotation>
    <xs:documentation>Tipo complexo de descrição do emissor do documento</xs:documentation>
  </xs:annotation>
  <xs:attribute name="numero_emissor" type="xs:long" use="required"/>
  <xs:attribute name="data_emissor" type="xs:date" use="required"/>
  <xs:attribute name="hora_emissor" type="xs:time" use="required"/>
  <xs:attribute name="comarca_emissor" type="xs:string" use="required"/>
  <xs:attribute name="serventia_emissor" type="xs:string" use="required"/>
  <xs:attribute name="oficial_emissor" type="xs:string" use="required"/>
  <xs:attribute name="municipio_emissor" type="xs:string" use="required"/>
  <xs:attribute name="uf_emissor" type="xs:string" use="required"/>
</xs:complexType>
```

Quadro 7 – Tipo complexo *ct_emissor*

O tipo complexo *ct_emissor* contém todos os dados definidos na especificação do protocolo de aplicação. Este tipo identifica a origem de envio dos documentos.

A consulta e busca de documentos através do protocolo acontece pela invocação de métodos disponibilizados pelo aplicativo servidor. Os métodos são acessados através de mensagens SOAP. O quadro 8 mostra a requisição de uma certidão.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body>
    <m:obterCertidao xmlns:m="http://DefaultNamespace" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <tipo xsi:type="xsd:string">String</tipo>
      <numeroRegistro xsi:type="xsd:int">0</numeroRegistro>
    </m:obterCertidao>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Quadro 8 – Mensagem SOAP de solicitação de certidão

No quadro 8 representa uma mensagem de solicitação de documento baseada em SOAP. Ela contém os parâmetros *tipo* e *numeroRegistro* que representam o tipo de certidão e o número de registro desejado respectivamente.

Como retorno de uma solicitação de certidão tem-se o documento XML certidão, documento este que representa um registro de nascimento, casamento ou óbito existente no RCPN. No quadro 9 é mostrado um documento XML que representa uma certidão de nascimento.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<doc_rcpn>
  <emissor numero_emissor="12345" data_emissor="2005-05-07" hora_emissor="15:00:38" comarca_emissor="Florianópolis"
   serventia_emissor="Ofício de Registro Civil das Pessoas Naturais" oficial_emissor="Pedro Almeida de Araújo" municipio_emissor="Florianópolis" uf_emissor="SC"/>
   <certidao numero_registro="3265" folha_registro="187" livro_registro="A-236" data_registro="1985-07-15">
      <nascimento>
         <nascido nome_nascido="João Vitor da Silva" sexo_nascido="M" hora_nascimento="16:15" loca_nascimento="Maternidade Carmela Dutra">
            municipio_nascimento="Florianópolis" uf_nascimento="SC">
            <pai nome_pai="João da Silva" profissao_pai="Operário" estado_civil_pai="casado" naturalidade_pai="São José" uf_pai="SC">
               <avo_paterno>Joaquim da Silva</avo_paterno>
               <avo_paterna>Luiza da Silva</avo_paterna>
            </pai>
            <mae nome_mae="Luiza da Silva" idade_mae="28" profissao_mae="Do lar" estado_civil_mae="Casada" naturalidade_mae="Blumenau"  uf_mae="SC">
               <avo_materno>Pedro da Silva</avo_materno>
               <avo_materna>Joana da Silva</avo_materna>
            </mae>
         </nascido>
         <declarantes>O Pai</declarantes>
         <observacoes>Nada consta.</observacoes>
      </nascimento>
   </certidao>
</doc_rcpn>
```

Quadro 9 – Documento XML que representa uma certidão de nascimento

No quadro acima são demonstrados todos os elementos e atributos que compõem uma certidão de nascimento contida no RCPN.

3.3.3 OPERACIONALIDADE

O processo de troca de documentos entre as aplicações cliente e servidor, implementados neste trabalho, podem ser divididas em três partes distintas:

a) configurar cliente – são informados os dados como endereço do servidor e local onde serão armazenados os documentos recebidos;

b) consultar registros – onde o sistema cliente envia uma solicitação ao sistema servidor sobre os registros referentes aos parâmetros informados;

c) obter documentos – onde o sistema cliente solicita o documento desejado ao invocar os métodos do *web service* passando os parâmetros necessários.

Para exemplificar a funcionalidade das partes integrantes do sistema, o usuário da aplicação cliente realizará consultas e a busca de documentos XML para emissão de certidões do registro civil que existem em uma base de dados remota.

3.3.3.1 CENÁRIO “CONFIGURAR CLIENTE”

O usuário utiliza os *web services* através da interface do sistema cliente. Para utilizar o aplicativo é necessário informar alguns parâmetros para execução dos métodos remotos. Os parâmetros são descritos abaixo:

a) *provedor de serviços* – endereço onde está localizado o servidor de web *services*. Por exemplo o endereço *http://201.3.253.45:8080/axis/RCPN.jws* representa o local onde encontra-se o *web service*;

b) *armazenar documentos em* – local onde serão salvos os documentos obtidos.

A figura 20 mostra os respectivos parâmetros informados pelo usuário.

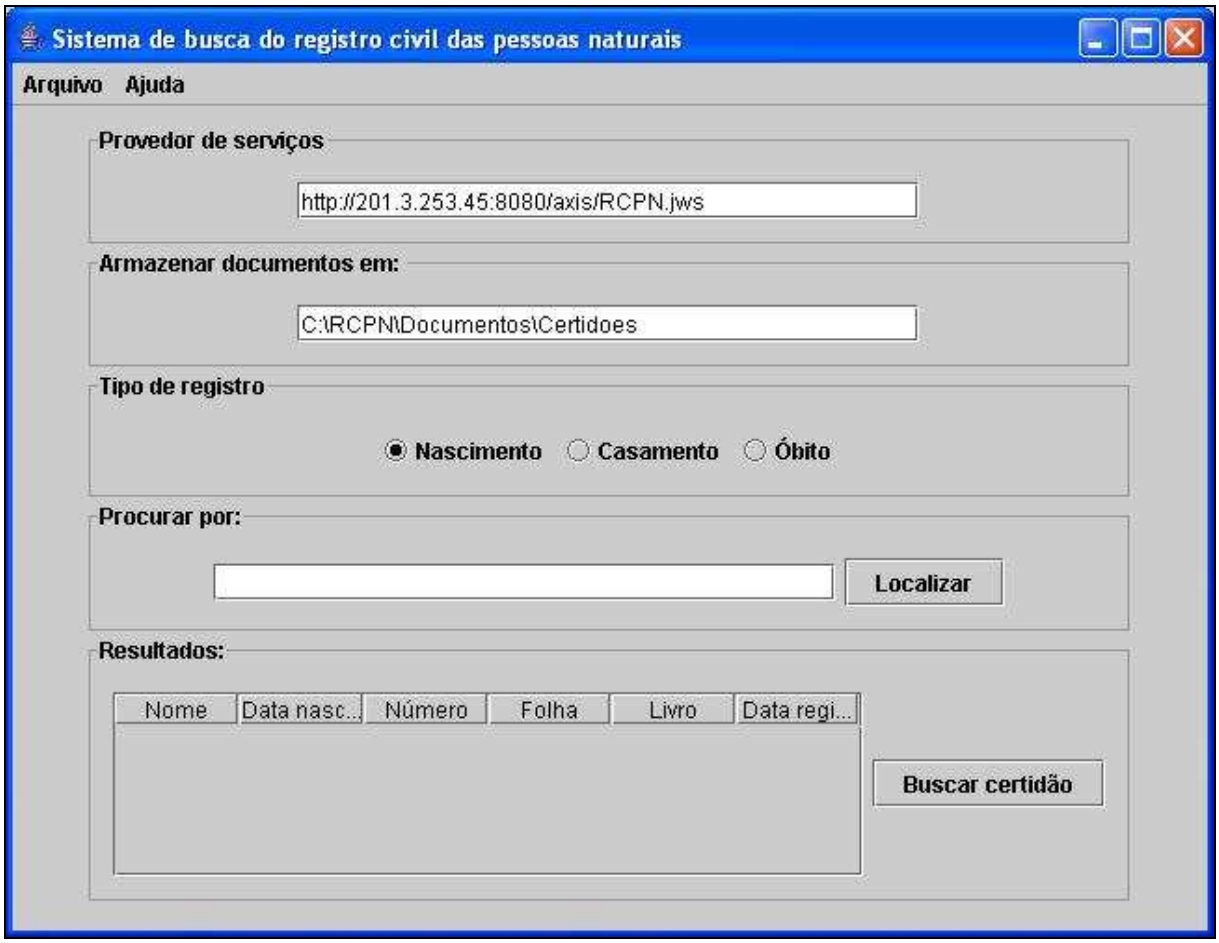


Figura 20 – Cenário *Configurar cliente*

3.3.3.2 CENÁRIO “CONSULTAR REGISTROS”

Para utilizar o método *consultar registros* deve-se proceder da seguinte maneira:

a) marcar a opção *tipo de registro* a ser consultado. As opções são nascimento, casamento ou óbito;

b) no campo *procurar por*, informar o nome da pessoa na qual deseja-se procurar o registro. Clicando no botão *localizar*, é mostrada a tela para autenticação do cliente do serviço onde devem ser informados os parâmetros *login* e *senha*. Os parâmetros citados anteriormente são previamente cadastrados no arquivo users.xml que encontra-se no servidor remoto. Através da tela de autenticação é executado o método *autenticarUsuario* para que seja permitido o acesso aos outros métodos disponibilizados pelo protocolo. O método *consultarRegistro* é executado no servidor remoto e retorna um documento XML que representa uma lista de registros encontrados com o valor informado anteriormente. Essa lista é carregada no objeto do tipo tabela. A figura 21 mostra a tela de autenticação de usuário do *web service*.

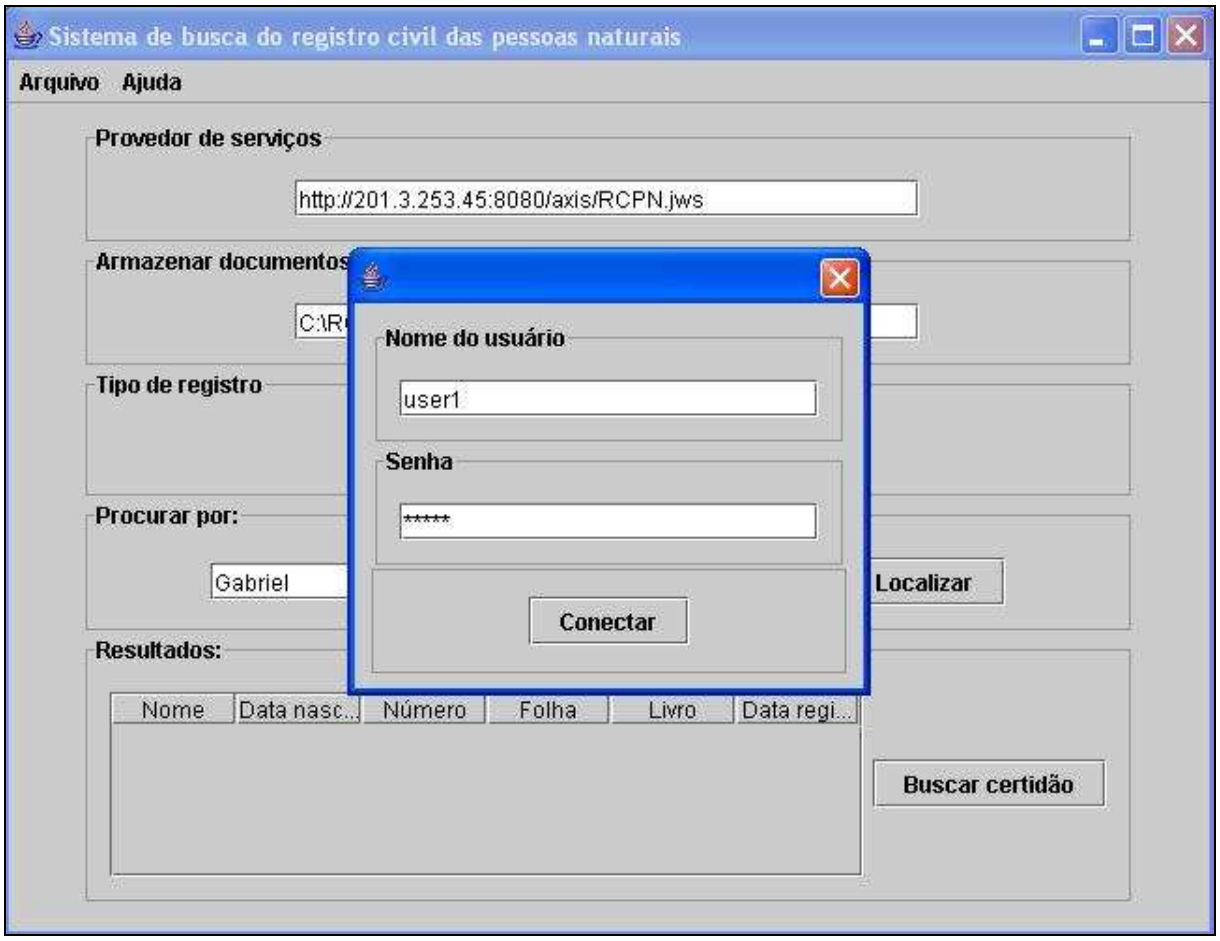


Figura 21 – Autenticação de usuário do *web service*

Devem ser informados o nome do usuário do serviço e sua respectiva senha de acesso. Após a autenticação é possível fazer consultas e buscar certidões.

A figura 22 mostra o resultado de uma consulta de registros de nascimentos feita após a autenticação do usuário pelo *web service*.

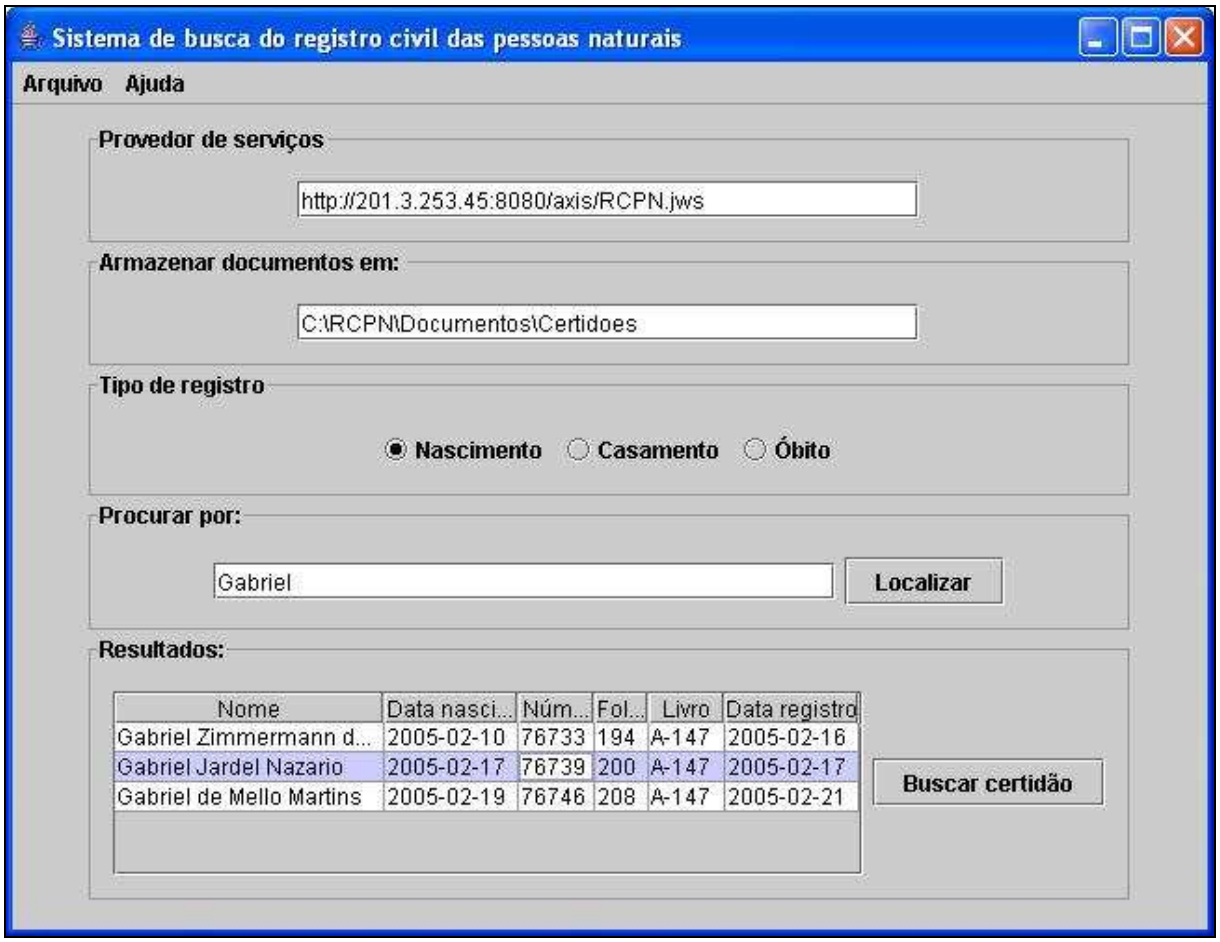


Figura 22 – Cenário *Consultar registros*

Nesta figura é mostrada uma lista de registros encontrados referentes ao nome da pessoa informado na busca de registros de nascimento.

3.3.3.3 CENÁRIO “OBTER DOCUMENTO”

Para busca um documento contido na lista de registros encontrados, deve-se selecionar o registro na grade de resultados e clicar no botão *buscar certidão*. Após a busca da certidão é mostrada uma caixa de diálogo onde pode-se ou não ver o documento XML que é armazenado no local informado no campo *armazenar documentos em*.

A figura 23 mostra o resultado da busca de uma certidão de nascimento.

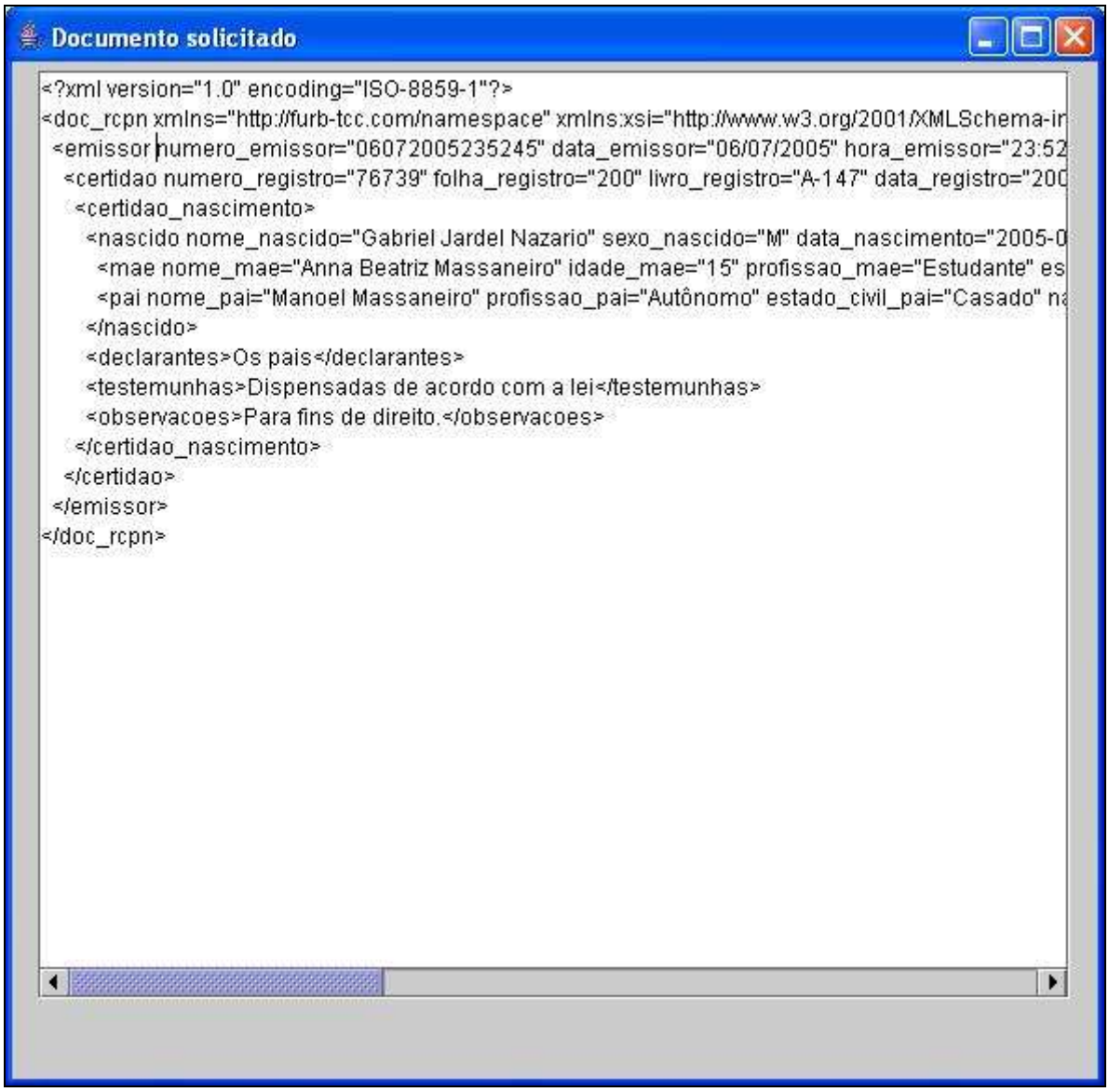


Figura 23 - Documento solicitado pelo cliente do *web service*

3.4 RESULTADOS E DISCUSSÃO

O protótipo de protocolo de aplicação modela documentos utilizados pelo registro civil das pessoas naturais. Este protocolo pode ser utilizado na integração dos sistemas das serventias extra judiciais.

O protocolo tem a capacidade de realizar consultas sobre registros e envio de certidões de nascimento, casamento e óbito entre sistemas remotos.

Para demonstrar a possibilidade de extensão do protocolo, foi especificada a classe relatório, não implementada neste trabalho, mas que pode ser desenvolvida para aumentar a utilização entre órgãos públicos que utilizam os dados dos registros públicos.

A XML *Schema* pode ser utilizada para criação de outros modelos de protocolos de aplicação para diversas áreas. Em conjunto com o protocolo SOAP pode integrar sistemas distintos mais facilmente se comparada com outros padrões de integração existentes atualmente, que baseiam-se em tecnologias com maior nível de complexidade de implementação. Este trabalho forneceu suporte adequado a modelagem dos documentos e pode facilmente estender o mesmo para outros tipos de registros públicos, tais como registro civil das pessoas jurídicas, registro de títulos e documentos, registro de imóveis entre outros.

4 CONCLUSÕES

O principal objetivo desse trabalho que foi a criação de um protótipo de protocolo de aplicação para a área extra judicial foi conseguido com êxito. O intercâmbio de documentos baseados no esquema XML implícitos em mensagens SOAP através do uso de *web services* demonstrou uma funcionalidade satisfatória.

A implementação do ambiente permite que os métodos de uma classe Java sejam acessados remotamente através de requisições de um aplicativo cliente, utilizando o protocolo SOAP. Esses métodos realizam consultas e retornam documentos XML baseados em registros armazenados em uma base de dados padrão SQL. O Axis implementa as funcionalidades do protocolo SOAP de uma forma satisfatória.

Com a implementação do protótipo de protocolo de aplicação, existe a possibilidade de utilização do mesmo na integração de sistemas de gerenciamento de dados de cartórios distintos. Embora o protocolo seja limitado, existe a possibilidade de extensão do mesmo para abranger outros documentos e outros tipos de cartórios que compõem os registros públicos. Para isso deve-se fazer um estudo aprofundado da prática dos registros públicos na área extra judicial.

4.1 EXTENSÕES

Seguem algumas sugestões para trabalhos futuros:

a) o desenvolvimento de uma extensão do protocolo para os outros tipos de cartórios que praticam os registros públicos;
b) a implementação de protocolos de aplicação para outras áreas que necessitem de integração entre sistemas heterogêneos;

c) um estudo comparativo entre tecnologias de integração de sistemas.

REFERÊNCIAS BIBLIOGRÁFICAS

AHMED, Kal et al. **Professional java xml**. Birmingham: Wrox Press Ltd., 2001. 1159 p.

BORTOLI, Dejane Luiza. **O documento eletrônico no ofício de registro civil das pessoas naturais**. 2002. 117 p. Dissertação (Mestrado em Ciências da Computação) – Centro de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis.

CENEVIVA, Walter. **Lei dos notários e dos registradores comentada**. São Paulo: Saraiva, 1996. 234 p.

COLPANI, Cristiano Fornari. **Protótipo de software para troca de dados entre aplicações de comércio eletrônico utilizando o protocolo SOAP**. 2002. 68 p. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

DANTAS, Laurentino Augusto. **ECN: protocolo criptográfico para emissão de certidões de nascimento na internet**. 2001. 96 p. Dissertação (Mestrado em Ciências da Computação) – Centro de Pós Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis.

DACONTA, Michael C., ORBST, Leo J., SMITH, Kevin T. **The semantic web**. Indianapolis: Wiley Publishing, Inc., 2003. 281 p.

DEITEL, H. M. et al. **XML como programar**. Tradução Luiz Augusto Salgado, Edson Furmankiewicz. Porto Alegre: Bookman., 2003. 972 p.

FILHO, Lair da Silva Loureiro; LOUREIRO, Cláudia R. M. **Notas e registros públicos**. São Paulo: Saraiva, 2004. 433 p.

FILHO, Nicolau Balbino. **Registro de imóveis**. 4. ed. São Paulo: Atlas, 1978. 411 p.

GRAHAM, Steve et al. **Building web services with java: making sense of XML, SOAP, WSDL, and UDDI**. Indianápolis: Sams, 2002. 581 p.

HENDRICKS, Mack et al. **Profissional java web services**. Tradução Marcos Rolo. Rio de Janeiro: Alta Books, 2002. 536 p.

JESUS, Alberto Pereira de; GOMES, Jefferson José. **Criação do sistema integrado de bibliotecas do sistema ACAFE: utilizando java e XML**. Blumenau, [2004]. Disponível em: <http://www.inf.furb.br/seminco/2004/artigos/106-vf.pdf>. Acesso em: 04 out. 2004.

JUNIOR, João Mendes de Almeida. **Órgãos da fé pública**. São Paulo: Saraiva, 1963. 186 p.

KRACIK, João. **Protótipo de comunicação e armazenamento de dados contábeis usando XML**. 2002. 99 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

NETO, Agostinho Campos. **Web services em java com axis**. [2004]. Disponível em: <http://www.guj.com.br/java.tutorial.artigo.132.1.guj>. Acesso em: 20 abr. 2005.

RAY, Erick T. **Aprendendo XML**. Tradução Daniel Vieira. Rio de Janeiro: Campus, 2001. 372 p.

SCOCHI, C. G. S. et al. **Intervalo entre o nascimento e o registro civil: situação no município de Riberão Preto, São Paulo, Brasil**. São Paulo, [2004]. Disponível em: <http://www.scielo.br/pdf/rbsmi/v4n2/21003.pdf>. Acesso em: 05 mar. 2005.

SKONNARD, Aaron; GUDGIN, Martin. **Essential XML quick reference**. Indianápolis: Pearson-Education, Inc., 2002. 430 p.

VLIST, Erik van der. **XML schema**. Sebastopol: O'Reilly & Associates, Inc., 2002. 400 p.

Apêndice A – XML *Schema* que representa o protocolo de aplicação

```
     <?xml version="1.0" encoding="ISO-8859-1" ?>
     -     <xs:schema     xmlns="http://furb-tcc.com/namespace"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://furb-tcc.com/namespace"
elementFormDefault="qualified"
attributeFormDefault="unqualified">
     - <xs:element name="doc_rcpn">
     - <xs:annotation>
       <xs:documentation>esquema de documento básico para
todos os outros documentos do registro civil das pessoas
naturais</xs:documentation>
       </xs:annotation>
     - <xs:complexType>
     - <xs:sequence>
     - <xs:element name="emissor">
     - <xs:complexType>
     - <xs:complexContent>
     - <xs:extension base="ct_emissor">
     - <xs:choice>
     - <xs:element name="consulta">
     - <xs:complexType>
     - <xs:choice>
```

```
     - <xs:element name="consulta_nascimento" minOccurs="0"
     maxOccurs="unbounded">
     - <xs:complexType>
       <xs:attribute    name="nome_nascido"    type="xs:string"
use="required" />
       <xs:attribute   name="data_nascimento"   type="xs:date"
use="required" />
       <xs:attribute  name="numero_registro"  type="xs:integer"
use="required" />
       <xs:attribute   name="folha_registro"   type="xs:integer"
use="required" />
       <xs:attribute   name="livro_registro"   type="xs:string"
use="required" />
       <xs:attribute    name="data_registro"    type="xs:date"
use="required" />
       </xs:complexType>
       </xs:element>
     -   <xs:element   name="consulta_casamento"   minOccurs="0"
maxOccurs="unbounded">
     - <xs:complexType>
       <xs:attribute     name="nome_noiva"     type="xs:string"
use="required" />
       <xs:attribute     name="nome_noivo"     type="xs:string"
use="required" />
```

```
     <xs:attribute    name="livro_registro"    type="xs:string"
use="required" />
       <xs:attribute   name="folha_registo"   type="xs:integer"
use="required" />
       <xs:attribute   name="numero_registo"   type="xs:integer"
use="required" />
       <xs:attribute    name="data_registro"    type="xs:date"
use="required" />
       </xs:complexType>
       </xs:element>
     -    <xs:element    name="consulta_obito"    minOccurs="0"
maxOccurs="unbounded">
     - <xs:complexType>
       <xs:attribute   name="nome_falecido"   type="xs:string"
use="required" />
       <xs:attribute   name="data_falecimento"   type="xs:date"
use="required" />
       <xs:attribute  name="numero_registro"  type="xs:integer"
use="required" />
       <xs:attribute   name="folha_registro"   type="xs:integer"
use="required" />
       <xs:attribute   name="livro_registro"   type="xs:string"
use="required" />
       <xs:attribute   name="data_registro"   type="xs:integer"
use="required" />
```

```
      </xs:complexType>
       </xs:element>
       </xs:choice>
       </xs:complexType>
       </xs:element>
     - <xs:element name="certidao">
     - <xs:complexType>
     - <xs:complexContent>
     - <xs:extension base="ct_registro">
     - <xs:choice>
     - <xs:element name="certidao_nascimento">
     - <xs:complexType>
     - <xs:sequence>
     - <xs:element name="nascido">
     - <xs:complexType>
     - <xs:complexContent>
     - <xs:extension base="ct_nascido">
     - <xs:sequence>
       <xs:element name="pai" type="ct_pai" minOccurs="0" />
       <xs:element name="mae" type="ct_mae" />
       </xs:sequence>
       </xs:extension>
       </xs:complexContent>
       </xs:complexType>
       </xs:element>
```

```
      <xs:element name="declarantes" type="xs:string" />
       <xs:element name="testemunhas" type="xs:string" />
       <xs:element name="observacoes" type="xs:string" />
       </xs:sequence>
       </xs:complexType>
       </xs:element>
     - <xs:element name="certidao_casamento">
     - <xs:complexType>
     - <xs:sequence>
       <xs:element name="noivo" type="ct_noivo" />
       <xs:element name="noiva" type="ct_noiva" />
       <xs:element name="juiz" type="xs:string" />
       <xs:element name="testemunhas" type="xs:string" />
       <xs:element name="regime" type="xs:string" />
       <xs:element name="observacoes" type="xs:string" />
       </xs:sequence>
       </xs:complexType>
       </xs:element>
     - <xs:element name="certidao_obito">
     - <xs:complexType>
     - <xs:sequence>
       <xs:element name="falecido" type="ct_falecido" />
       <xs:element name="causas" type="xs:string" />
       <xs:element  name="sepultamento"  type="ct_sepultamento"
/>
```

```
     <xs:element    name="atestado"    type="ct_atestado_obito"
minOccurs="0" />
       <xs:element                              name="declarante"
type="ct_declarante_obito" />
       <xs:element name="observacoes" type="xs:string" />
       </xs:sequence>
       </xs:complexType>
       </xs:element>
       </xs:choice>
       </xs:extension>
       </xs:complexContent>
       </xs:complexType>
       </xs:element>
       <xs:element name="relatorio" />
       </xs:choice>
       </xs:extension>
       </xs:complexContent>
       </xs:complexType>
       </xs:element>
       </xs:sequence>
       </xs:complexType>
       </xs:element>
     - <xs:complexType name="ct_emissor">
     - <xs:annotation>
```

```
       <xs:documentation>Tipo complexo de descrição do emissor
do documento</xs:documentation>
       </xs:annotation>
       <xs:attribute    name="numero_emissor"    type="xs:long"
use="required" />
       <xs:attribute     name="data_emissor"     type="xs:date"
use="required" />
       <xs:attribute     name="hora_emissor"     type="xs:time"
use="required" />
       <xs:attribute   name="comarca_emissor"   type="xs:string"
use="required" />
       <xs:attribute  name="serventia_emissor"  type="xs:string"
use="required" />
       <xs:attribute   name="oficial_emissor"   type="xs:string"
use="required" />
       <xs:attribute  name="municipio_emissor"  type="xs:string"
use="required" />
       <xs:attribute     name="uf_emissor"     type="xs:string"
use="required" />
       </xs:complexType>
     - <xs:complexType name="ct_registro">
     - <xs:annotation>
       <xs:documentation>Tipo            complexo           de
registro</xs:documentation>
       </xs:annotation>
```

```
      <xs:attribute   name="numero_registro"   type="xs:integer"
use="required" />
       <xs:attribute   name="folha_registro"   type="xs:integer"
use="required" />
       <xs:attribute    name="livro_registro"    type="xs:string"
use="required" />
       <xs:attribute     name="data_registro"     type="xs:date"
use="required" />
       <xs:attribute    name="hora_registro"    type="xs:string"
use="optional" />
       </xs:complexType>
     - <xs:complexType name="ct_nascido">
     - <xs:annotation>
       <xs:documentation>Tipo   complexo   de   descrição   de
criança</xs:documentation>
       </xs:annotation>
       <xs:attribute    name="nome_nascido"    type="xs:string"
use="required" />
       <xs:attribute    name="sexo_nascido"    type="xs:string"
use="required" />
       <xs:attribute    name="data_nascimento"    type="xs:date"
use="required" />
       <xs:attribute   name="hora_nascimento"   type="xs:string"
use="required" />
```

```
     <xs:attribute   name="local_nascimento"   type="xs:string"
use="required" />
       <xs:attribute               name="municipio_nascimento"
type="xs:string" use="required" />
       <xs:attribute   name="uf_nascimento"   type="xs:string"
use="required" />
       </xs:complexType>
     - <xs:complexType name="ct_mae">
     - <xs:annotation>
       <xs:documentation>Tipo   complexo   de   descrição   de
mãe</xs:documentation>
       </xs:annotation>
       <xs:attribute      name="nome_mae"      type="xs:string"
use="required" />
       <xs:attribute       name="idade_mae"       type="xs:int"
use="required" />
       <xs:attribute   name="profissao_mae"   type="xs:string"
use="required" />
       <xs:attribute  name="estado_civil_mae"  type="xs:string"
use="required" />
       <xs:attribute  name="naturalidade_mae"  type="xs:string"
use="required" />
       <xs:attribute       name="uf_mae"       type="xs:string"
use="required" />
```

```
      <xs:attribute     name="avo_materno"     type="xs:string"
use="required" />
       <xs:attribute    name="avo_materna"    type="xs:string"
use="required" />
       </xs:complexType>
     - <xs:complexType name="ct_pai">
     - <xs:annotation>
       <xs:documentation>Tipo   complexo   de   descrição   de
pai</xs:documentation>
       </xs:annotation>
       <xs:attribute      name="nome_pai"      type="xs:string"
use="required" />
       <xs:attribute   name="profissao_pai"   type="xs:string"
use="required" />
       <xs:attribute  name="estado_civil_pai"  type="xs:string"
use="required" />
       <xs:attribute  name="naturalidade_pai"  type="xs:string"
use="required" />
       <xs:attribute       name="uf_pai"       type="xs:string"
use="required" />
       <xs:attribute    name="avo_paterno"    type="xs:string"
use="required" />
       <xs:attribute    name="avo_paterna"    type="xs:string"
use="required" />
       </xs:complexType>
```

```
     - <xs:complexType name="ct_noivo">
     - <xs:annotation>
       <xs:documentation>Tipo complexo de descrição de
noivo</xs:documentation>
       </xs:annotation>
       <xs:attribute name="nome_noivo" type="xs:string"
use="required" />
       <xs:attribute name="data_nascimento_noivo"
type="xs:date" use="required" />
       <xs:attribute name="nacionalidade_noivo"
type="xs:string" use="required" />
       <xs:attribute name="profissao_noivo" type="xs:string"
use="optional" />
       <xs:attribute name="estado_civil_noivo"
type="xs:string" use="required" />
       <xs:attribute name="naturalidade_noivo"
type="xs:string" use="required" />
       <xs:attribute name="uf_noivo" type="xs:string"
use="required" />
       <xs:attribute name="pai_noivo" type="xs:string"
use="optional" />
       <xs:attribute name="mae_noivo" type="xs:string"
use="required" />
       </xs:complexType>
     - <xs:complexType name="ct_noiva">
```

```
     - <xs:annotation>
       <xs:documentation>Tipo complexo de descrição de
noiva</xs:documentation>
       </xs:annotation>
       <xs:attribute name="nome_noiva" type="xs:string"
use="required" />
       <xs:attribute name="data_nascimento_noiva"
type="xs:date" use="required" />
       <xs:attribute name="nacionalidade_noiva"
type="xs:string" use="required" />
       <xs:attribute name="profissao_noiva" type="xs:string"
use="optional" />
       <xs:attribute name="estado_civil_noiva"
type="xs:string" use="required" />
       <xs:attribute name="naturalidade_noiva"
type="xs:string" use="required" />
       <xs:attribute name="uf_noiva" type="xs:string"
use="required" />
       <xs:attribute name="pai_noiva" type="xs:string"
use="optional" />
       <xs:attribute name="mae_noiva" type="xs:string"
use="required" />
       </xs:complexType>
     - <xs:complexType name="ct_falecido">
     - <xs:annotation>
```

```
      <xs:documentation>Tipo complexo de descrição de
falecido</xs:documentation>
       </xs:annotation>
       <xs:attribute name="nome_falecido" type="xs:string"
use="required" />
       <xs:attribute name="sexo_falecido" type="xs:string"
use="required" />
       <xs:attribute name="data_falecimento" type="xs:date"
use="required" />
       <xs:attribute name="hora_falecimento" type="xs:string"
use="required" />
       <xs:attribute name="idade_falecido" type="xs:int"
use="required" />
       <xs:attribute name="nacionalidade_falecido"
type="xs:string" use="required" />
       <xs:attribute name="estado_civil_falecido"
type="xs:string" use="required" />
       <xs:attribute name="identidade_falecido"
type="xs:string" use="required" />
       </xs:complexType>
     - <xs:complexType name="ct_sepultamento">
     - <xs:annotation>
       <xs:documentation>Tipo complexo de descrição de
sepultamento</xs:documentation>
       </xs:annotation>
```

```
      <xs:attribute                name="cemiterio_sepultamento"
type="xs:string" use="required" />
       <xs:attribute              name="municipio_sepultamento"
type="xs:string" use="required" />
       <xs:attribute  name="uf_sepultamento"  type="xs:string"
use="required" />
       </xs:complexType>
     - <xs:complexType name="ct_atestado_obito">
     - <xs:annotation>
       <xs:documentation>Tipo   complexo   de   descrição   de
atestado de óbito</xs:documentation>
       </xs:annotation>
       <xs:attribute               name="medico_atestado_obito"
type="xs:string" use="required" />
       <xs:attribute                  name="crm_atestado_obito"
type="xs:integer" use="required" />
       </xs:complexType>
     - <xs:complexType name="ct_declarante_obito">
     - <xs:annotation>
       <xs:documentation>Tipo   complexo   de   declarante   de
óbito</xs:documentation>
       </xs:annotation>
       <xs:attribute               name="nome_declarante_obito"
type="xs:string" use="required" />
```

```
       <xs:attribute      name="nacionalidade_declarante_obito"
type="xs:string" use="required" />
       <xs:attribute       name="estado_civil_declarante_obito"
type="xs:string" use="required" />
       <xs:attribute          name="profissao_declarante_obito"
type="xs:string" use="required" />
       <xs:attribute         name="identidade_declarante_obito"
type="xs:string" use="required" />
       </xs:complexType>
     </xs:schema>
```

Apêndice B – Arquivo wsdl que contém a descrição dos métodos do *web service*

```
<?xml version="1.0" encoding="UTF-8" ?>
-<wsdl:definitions
targetNamespace="http://201.3.202.241:8080/axis/RCPN.jws"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://201.3.202.241:8080/axis/RCPN.jws"
xmlns:intf="http://201.3.202.241:8080/axis/RCPN.jws"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     - <wsdl:message name="autenticarUsuarioResponse">
       <wsdl:part                name="autenticarUsuarioReturn"
type="xsd:string" />
       </wsdl:message>
     - <wsdl:message name="obterCertidaoRequest">
       <wsdl:part name="tipo" type="xsd:string" />
       <wsdl:part name="numeroRegistro" type="xsd:string" />
       </wsdl:message>
     - <wsdl:message name="consultarRegistroResponse">
       <wsdl:part                name="consultarRegistroReturn"
type="xsd:string" />
       </wsdl:message>
     - <wsdl:message name="obterCertidaoResponse">
```

```
       <wsdl:part  name="obterCertidaoReturn"  type="xsd:string"
/>
       </wsdl:message>
     - <wsdl:message name="autenticarUsuarioRequest">
       <wsdl:part name="login" type="xsd:string" />
       <wsdl:part name="senha" type="xsd:string" />
       </wsdl:message>
     - <wsdl:message name="consultarRegistroRequest">
       <wsdl:part name="tipo" type="xsd:string" />
       <wsdl:part name="nome" type="xsd:string" />
       </wsdl:message>
     - <wsdl:portType name="RCPN">
     -         <wsdl:operation         name="autenticarUsuario"
parameterOrder="login senha">
       <wsdl:input      message="impl:autenticarUsuarioRequest"
name="autenticarUsuarioRequest" />
       <wsdl:output    message="impl:autenticarUsuarioResponse"
name="autenticarUsuarioResponse" />
       </wsdl:operation>
     -         <wsdl:operation         name="consultarRegistro"
parameterOrder="tipo nome">
       <wsdl:input      message="impl:consultarRegistroRequest"
name="consultarRegistroRequest" />
       <wsdl:output    message="impl:consultarRegistroResponse"
name="consultarRegistroResponse" />
       </wsdl:operation>
```

```
     -           <wsdl:operation           name="obterCertidao"
parameterOrder="tipo numeroRegistro">
       <wsdl:input          message="impl:obterCertidaoRequest"
name="obterCertidaoRequest" />
       <wsdl:output        message="impl:obterCertidaoResponse"
name="obterCertidaoResponse" />
       </wsdl:operation>
       </wsdl:portType>
     - <wsdl:binding name="RCPNSoapBinding" type="impl:RCPN">
       <wsdlsoap:binding                              style="rpc"
transport="http://schemas.xmlsoap.org/soap/http" />
     - <wsdl:operation name="autenticarUsuario">
       <wsdlsoap:operation soapAction="" />
     - <wsdl:input name="autenticarUsuarioRequest">
       <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://DefaultNamespace" use="encoded" />
       </wsdl:input>
     - <wsdl:output name="autenticarUsuarioResponse">
       <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://201.3.202.241:8080/axis/RCPN.jws"
use="encoded" />
       </wsdl:output>
       </wsdl:operation>
     - <wsdl:operation name="consultarRegistro">
```

```
       <wsdlsoap:operation soapAction="" />
     - <wsdl:input name="consultarRegistroRequest">
       <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://DefaultNamespace" use="encoded" />
       </wsdl:input>
     - <wsdl:output name="consultarRegistroResponse">
       <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://201.3.202.241:8080/axis/RCPN.jws"
use="encoded" />
       </wsdl:output>
       </wsdl:operation>
     - <wsdl:operation name="obterCertidao">
       <wsdlsoap:operation soapAction="" />
     - <wsdl:input name="obterCertidaoRequest">
       <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://DefaultNamespace" use="encoded" />
       </wsdl:input>
     - <wsdl:output name="obterCertidaoResponse">
       <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://201.3.202.241:8080/axis/RCPN.jws"
use="encoded" />
       </wsdl:output>
```

```
        </wsdl:operation>
        </wsdl:binding>
      - <wsdl:service name="RCPNService">
      - <wsdl:port binding="impl:RCPNSoapBinding" name="RCPN">
        <wsdlsoap:address
location="http://201.3.202.241:8080/axis/RCPN.jws" />
        </wsdl:port>
        </wsdl:service>
        </wsdl:definitions>
```