Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Pós-Graduação em Engenharia de Eletricidade

Framework de Aplicações Móveis com Segurança em SOA

Johnneth de Sene Fonsêca

São Luís
2009

Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Pós-Graduação em Engenharia de Eletricidade

Framework de Aplicações Móveis com Segurança em SOA

Johnneth de Sene Fonsêca

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Eletricidade da UFMA como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

São Luís
2009

Fonsêca, Johnneth de Sene

Framework para Desenvolvimento de Aplicações baseados em SOA em Ambiente de Computao Móvel com Agregação de Serviços de Segurança / Johnneth de Sene Fonsêca.

126f.

Dissertação (Mestrado) - Programa de Pós Graduação em Engenharia de Eletricidade, Universidade Federal do Maranhão, São Luís, 2009.

1. SOA 2.Computação Móvel 3. Segurança. I.Título. Framework para Desenvolvimento de Aplicações baseados em SOA em Ambiente de Computao Móvel com Agregação de Serviços de Segurança

CDU:004.652.5

FRAMEWORK DE APLICAÇÕES MÓVEIS COM SEGURANÇA EM SOA

Johnneth de Sene Fonsêca

Dissertação aprovada em 16 de fevereiro de 2009.

Prof. Zair Abdelouahab, Ph.D.
(Orientador)

Prof. Djamel Fawzi Hadj Sadok, Ph.D.
(Membro da Banca Examinadora)

Prof. Francisco José da Silva e Silva, Dr.
(Membro da Banca Examinadora)

Prof. Denivaldo Cicero Pavão Lopes, Dr.
(Membro da Banca Examinadora)

"Com o critério com que julgardes, sereis julgados; e, com a medida com que tiverdes medido, vos medirão também." **Jesus Cristo**

"Tudo aquilo que nós precisamos, Deus tem e deseja que tenhamos. A nós, basta aceitar o que Ele nos oferece." **Autor desconhecido**

"O que eu faço hoje é importante, porque estou trocando um dia de minha vida por isso." **Bob Marley**

"Aquele que conhece os outros é sábio, aquele que conhece a si próprio é iluminado.". Lao-Tsé

Dedicatória

A minha mãe Maurina, a meu pai João, meus irmãos Johnrimá e Johberth e as minhas queridas sobrinhas, Camile, Vivian e Carina, aos meus avós in Memorian Crispiando, Raimundo Nonato, Senhorinha e Dinorá. Aos meus tios e tias, em especial a Tia Concinha, que sempre me deram conselhos sobre a vida. Aos meus primos e primas, sempre presentes nas diferentes fases da minha vida. Aos amigos que fiz ao longo da minha vida, em especaial a Luís, Líliam, Josélia, Guilherme, Nilson, Leandro, Cristina, Letícia, Newton, Geraldo e Eduardo, mesmo os mais distantes cuja amizade sempre se faz presente.

Agradecimentos

Em primeiro lugar a Deus, a quem sempre recorri nos momentos difíceis.

Ao meu orientador, o professor PhD. Zair Abdelouahab.

Aos professores Djamel SAdok, Francisco Silva e Denivaldo Lopes que aceitaram prontamente participar da Banca Examinadora desta dissertação e pelas contribuições feitas ao trabalho.

A minha família, , pela confiança e motivação.

Aos amigos e colegas do curso, em especial a Alfredo, Lindonete, Simone, Ricardo, Osvaldo, Irlandino, Flávio, Helaine, Aline e Falker pela força e pela vibração em relação a esta jornada.

Aos Professores do PPGEE, Colegas de Curso e do Laboratório de Pesquisa, pois juntos vencemos grandes desafios em nossas vidas.

A todos que ajudaram direta ou indiretamente na realização deste trabalho e a todos aqueles que sempre me apoiaram em todos os momentos da minha vida.

Resumo

A constante evolução das tecnologias utilizadas em dispositivos móveis permitiu o aumento das suas capacidades no que diz respeito ao seu armazenamento, processamento e transmissão de dados, inclusive com mais de um tipo de tecnologia de transmissão em um mesmo dispositivo, e também do acesso a internet de forma mais eficiente. Estes fatores permitiram que um maior número de aplicações e serviços possam ser disponibilizados neles. Com isso surgiu à necessidade de se encontrar um modelo de desenvolvimento de serviços para os dispositivos móveis e sua disponibilização de forma mais rápida e eficiente, além de que os dados sejam transmitidos de forma segura. Uma das melhores opções existentes atualmente são os SOAs (Arquitetura Orientada a Serviços), um modelo de desenvolvimento em grande evidência atualmente. Esta dissertação visa apresentar um *Framework* que permite o desenvolvimento de SOA no ambiente móvel, dando ao desenvolvedor todas as ferramentas necessárias para provisão de serviços neste tipo de ambiente. Também é proposta uma ferramenta para o uso de mecanismo de segurança pelo *Framework*.

Palavras-Chave: 1. SOA 2.Computação Móvel 3. Segurança

Abstract

The constant evolution of technologies used in mobile devices has increased its capabilities with respect to its storage, processing and transmission of data, including more of one kind of technology transfer in a single device, and also access Internet more efficiently. These factors have a greater number of applications and services may be provided therein. With this came the need to find a model for developing services and making them available more quickly and efficient, and that data is transmitted more securely. One of the best options currently existing are the SOAS (Services Oriented Architecture) a development model in great evidence today. The aim of this dissertation is present a framework that allows the development of SOA in the mobile environment, giving the developer all the tools necessary for provision of services in this type of environment. A tool for the use of security mechanism for the Framework is also proposed.

Keywords: 1. SOA 2.Mobile Computing 3. Security

Lista de Tabelas

Lista de Figuras

Lista de Abreviaturas e Siglas

3G	Terceira Geração
AP	Access Point
API	Application Programming Interface
CORBA	Common Object Request Broker Architecture
DCOM	Distributed Component Object Model
GET	Get Execute Trigger
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	General System Mobile Communication
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Security
ICP-Brasil	Infra-Estrutura de Chaves Públicas Brasileiras
IETF	Internet Engineering Task Force
MANET	Mobile Ad Hoc Network
OLSRd	Optimized Link State Routing Protocol
OMA	Open Mobile Alliance Group
P2P	Peer-to-Peer
PDA	Personal Digital Assistant
PICO	Pervasive Information Communities Organization
POST	Power On Self Test
QoS	Quality of Service
SAML	Security Assertion Markup Language
SeSCO	Seamless Service Composition in Pervasive Environments
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
UMtS	Universal Mobile Telecommunications System
TLS	Transport Layer Security
UDDI	Universal Description, Discovery and Integration
WAP	Wireless Application Protocol
WS-I	Web Services Interoperability Organization

WS-Security	Web Services Security
XKMS	eXtensible Markup Language Key Management Specification
XML	eXtensible Markup Language
XML Encryption	eXtensible Markup Language Encryption
XML Signature	eXtensible Markup Language Signature

Sumário

Capítulo 1

Introdução

1.1 Cenários e Definição do Problema

A evolução tecnológica das redes de comunicação de dados sem fio, a possibilidade de integração destas ao mundo IP e à Internet, associada à adequada especificação de sistemas e às necessidades de mercado, permitiram o crescimento exponencial do mercado das comunicações móveis.

A mobilidade possibilita a extensão do ambiente de trabalho da empresa às áreas externas, levando o acesso remoto às informações corporativas para os seus colaboradores, permitindo-lhes a aplicação de ações imediatas e integrando-os melhor em ações de trabalho colaborativo. Também, permite selecionar a informação a ser disponibilizada ao usuário, onde somente o conteúdo relevante naquele momento seja considerado.

Além disso, a evolução do meio de comunicação entre os dispositivos possibilita que eles se comuniquem diretamente de maneira mais eficiente, onde há troca de qualquer tipo de dado, como vídeo, fotos, entre outros. Outra característica existente traz o dispositivo móvel, que possui uma maior capacidade de processamento e armazenamento, tenha o papel de provedor de serviços, ou seja, o serviço a ser utilizado por um dispositivo se encontra em outro dispositivo, que estando em qualquer lugar e utilizando uma tecnologia qualquer para ser invocado.

Mesmo com todas as suas vantagens, há uma série de características nos dispositivos móveis que limitam a possibilidade do uso de padrões de sistemas tradicionais, como a variação de localização, a restrição de capacidades e a di-

versidades das aplicações. Além disso, as limitações de poder de memória e processamento dos dispositivos móveis tornam seu uso mais restrito. Ou seja, apenas aplicações simples e pequenas podem executar nesses aparelhos. Entretanto, aplicações móveis podem prover adição de valores para serem usados, o que os torna muito interessantes para diferentes tipos de corporações, como bancos, empresas de telefonia, lojas, aplicações baseadas em contexto e localização, etc.. Conseqüentemente, seu uso se torna bem atrativo, desde que se haja um modelo de desenvolvimento, que permita sua rápida implementação e disponibilização para o seu público alvo.

Outro ponto no advento das redes wireless é o surgimento de uma série de novas ameaças a segurança, cujo tratamento não pode ser realizado através das contramedidas normalmente aplicadas em rede cabeadas. Logo, no ambiente de computação móvel a preocupação com a segurança se torna mais acentuada, uma vez que o meio pelo qual as informações trafegam torna os dados mais passíveis de serem obtidos por usuários indevidos. Desta forma, percebe-se que um ponto fundamental no processo de desenvolvimento de serviços móveis é a preocupação com a segurança, não apenas na captura de dados através de pontos maliciosos que estejam agindo na rede, mas também na preocupação do serviço disponibilizado aos usuários aos quais estejam destinados, garantindo a integridade e a confidencialidade dos seus dados e qualquer outro tipo de informação, levando a necessidade de criar um mecanismo para proteção destes dados. Portanto, mecanismos que forneçam soluções de confidencialidade, integridade, não-repúdio e autenticação são indispensáveis para esses sistemas.

1.2 Justificativa

Durante os últimos anos houve um considerável aumento das capacidades dos aparelhos portáteis, tanto na sua capacidade de armazenamento quanto relativo ao seu processamento. Esta evolução permitiu que estes aparelhos tornarem-se mais populares e passarem a oferecer um grande número de aplicações e serviços aos seus usuários finais. Entretanto, os dispositivos móveis não permitem o desenvolvimento de qualquer aplicação, pois o sistema controla os módulos de software presentes no aparelho, logo somente as aplicações que atendam a determinados

pre-requisitos podem ser instalados no dispositivo de acordo com as suas especificações. Além disso, novas tecnologias de comunicação permitiram que estes aparelhos acessem a Internet de forma mais eficiente, além de se comunicarem entre si.

Existe, ainda, a possibilidade de instalar nestes aparelhos outras aplicações e/ou serviços além daqueles previamente instalados de fábrica. Pode-se perceber que estes dispositivos atuam não apenas como consumidores, mas também como provedores de serviços. Desta forma, um padrão de desenvolvimento que permita aos desenvolvedores criarem e disponibilizarem seus serviços de forma rápida e eficiente torna-se necessário.

A melhor opção para este tipo de cenário é utilizar Arquitetura Orientada a Serviços (SOA) um modelo de desenvolvimento que propõe um uso de um conjunto de padrões para descrever, disponibilizar, publicar e invocar serviços. Por isso, busca-se neste trabalho, agregar essas duas visões, ou seja, lançar mão dos SOA para a integração de Sistemas Móveis a outros sistemas, assunto bastante recente, e de extrema relevância. Logo, os principais ganhos são a diminuição do tempo de desenvolvimento, onde o desenvolvedor estaria focado somente em desenvolver a sua aplicação e não mais em como disponibilizar a sua aplicação, e permitir que equipamentos heterogêneos se comuniquem, já que SOA possui uma interface única de comunicação

A proposta descrita nesta dissertação é um framework que permite o desenvolvimento de SOAs em ambiente de computação móvel extraindo a complexidade do seu desenvolvimento, pois já possui mecanismos para realizar todas as funcionalidades necessárias para provisão de serviços, como descrever os serviços, realizar parser das mensagens de/para um formato específico, criar um canal de comunicação para receber e enviar mensagens. Além disso, um estudo da necessidade de adicionar propriedades de segurança foi feito, bem como a implementação de um mecanismo de segurança que permita o uso dessas propriedades, integrando o suporte de segurança em SOA para dispositivos móveis.

1.3 Objetivos

1.3.1 Objetivos Gerais

Esta dissertação tem por objetivo apresentar um estudo sobre arquitetura orientada a serviços voltados para o ambiente móvel e identificando as características relevantes necessárias para o desenvolvimento de uma aplicação orientada a serviços e quais suas vantagens e funcionalidades neste tipo de ambiente. Desta forma, identificando as necessidades de segurança para este cenário específico, e elaborar um framework que possibilite o desenvolvimento de serviços neste tipo de ambiente.

1.3.2 Objetivos Específicos

No sentido de alcançar o objetivo geral pretendido, os seguintes objetivos devem ser atingindos:

a) Verificar os possíveis cenários existentes para o uso desta tecnologia;

b) Estudar os Requisitos de Segurança necessários para implementar uma Arquitetura Orientada a Serviços voltado para o ambiente móvel, considerando as limitações deste ambiente;

c) Desenvolver uma ferramenta para o desenvolvimento de aplicações baseadas na Arquitetura Orientada a Serviços no ambiente móvel abranjendo algum destes cenários estudados;

d) Elaborar uma arquitetura que possua todas as propriedades desejadas que permita o desenvolvimento de serviços com a mínima interação do desenvolvedor;

e) Construir uma ferramenta para agregar propriedades de segurança previamente levantadas ao framework proposto;

f) Avaliar a ferramenta proposta através de testes e estudos de caso;

1.4 Organização da Dissertação

Esta dissertação está estruturada em seis capítulos os quais são descritos a seguir:

O primeiro capítulo trata de uma breve introdução sobre objetivos e da motivação para o desenvolvimento deste trabalho.

O segundo trata do referencial teórico necessário para o entendimento deste trabalho.

No terceiro capítulo é feito um breve resumo de algumas soluções apresentadas na literatura semelhantes ao que está sendo proposto.

No quarto capítulo é feita a descrição da arquitetura proposta e a modelagem do framework.

No quinto capítulo são apresentados alguns estudos de casos e uma comparação entre a Arquitetura proposta e os trabalhos relacionados anteriormente apresentados.

No sexto capítulo são feitas as considerações finais e sugestões para trabalhos futuros.

Capítulo 2

Referêncial Teórico

2.1 SOA

SOA (OASIS 2008) é um padrão que descreve os principais conceitos de uma arquitetura de software e suas relações, onde um serviço e sua utilização são os conceitos fundamentais implícitos de SOAs, seguindo um modelo de publicação de serviços e de aplicações e do seu acesso universal.

SOA emergiu na computação distribuída devido à necessidade de distribuir os serviços computacionais para requisitantes de serviços heterogêneos (Dokovski 2004), sendo uma interface que descreve uma coleção de operações que são acessíveis pela rede através de algum formato padronizado (ex. XML(W3C 2006)). Estes requisitos são ativados em qualquer lugar em um ambiente computacional dinâmico e/ou móveis onde provedores de serviços computacionais oferecem um conjunto de serviços. Um SOA facilita aos requisitantes identificarem, procurarem e utilizarem serviços apropriados em qualquer momento entre o conjunto de serviços oferecidos pelos provedores de serviços.

SOA cria um ambiente distribuído no qual aplicações e componentes podem interoperabilizar de forma independente de linguagem e plataforma, estando focado na utilização de um padrão de comunicação bastante difundido entre as operações, viabilizando, assim, um modelo homogêneo para distribuição e composição de componentes. Estas características fazem com que as aplicações baseadas em SOA sejam fracamente acopladas e orientadas a serviços, o que permite o uso de vários serviços em conjunto para executar operações complexas.

SOA é um modelo de componentes, apresentando um ambiente para construção sistemas distribuídos (Srirama 2007). SOA comunica aplicações funcionalmente como serviços para o usuário final da aplicação e outros serviços, levando os benefícios de baixo acoplamento e encapsulamento para a integração de aplicações coorporativas. SOA define as regras dos participantes como, provedor de serviços, consumidor de serviços e registro de serviços, a arquitetura de um SOA e seus componentes serão descritos nas próximas seções. SOA não é uma abordagem nova e muitas tecnologias como CORBA e DCOM já apresentaram uma idéia semelhante ao que SOA apresenta. Além disso, o padrão que mais se aproxima do modelo SOA são os Web Services.

2.1.1 Serviços

Um serviço é uma função independente que, normalmente, não possui estado (stateless), pois um serviço somente é carregado na memória principal quando este for solicitado. O serviço aceita uma ou mais requisições e devolve uma ou mais respostas através de uma interface padronizada e bem definida, ou seja, um serviço não precisa está presente na memória principal quando não está sendo utilizado, desta forma libera os recursos do dispositivo. Entretanto, a operação de busca e recarga do serviço na memória principal é uma operação que leva algum tempo e consome os recursos do dispositivo. Serviços podem também realizar partes discretas de um processo tal como editar ou processar uma transação. Serviços não devem depender do estado de outras funções ou processos. A tecnologia utilizada para prover o serviço, tal como uma linguagem de programação, não pode fazer parte da definição do serviço.

Os serviços (Papazoglou 2003) são componentes abertos e auto-descritivos que oferecem apoio à composição de aplicações/soluções distribuídas de uma forma rápida e com baixo custo. A descrição de um serviço é usada para apresentar - a organizações potencialmente interessadas - suas capacidades, sua interface, seu comportamento e sua qualidade, um SOA deve possuir as seguintes características:

- **Orquestração**: Processo de seqüenciar serviços e prover uma lógica adicional para processar dados. Não inclui uma representação de dados;
- **Provedor**: O recurso que executa o serviço em resposta a uma requisição

de um consumidor;

- **Consumidor**: Entidade que consome ou pede o resultado de um serviço fornecido por um provedor;
- **Descoberta**: SOA se baseia na capacidade de identificar serviços e suas características. Conseqüentemente, esta arquitetura depende de um diretório que descreva quais os serviços existentes dentro de um domínio;
- **Binding**: A relação entre os serviços do provedor e do consumidor deve ser idealmente dinâmica, ela é estabelecida em tempo de execução através de um mecanismo de Binding;

2.1.2 Componentes

A arquitetura básica do terminal móvel como provedor de serviços pode ser estabelecido como mostra a Figura 2.1. Ainda que o provedor de serviços seja implementado em um dispositivo móvel um padrão para descrição, como o WSDL, pode ser usado para descrever o serviço, e o padrão de registro, como UDDI, pode ser usado para publicar e tornar o serviço indisponível. Sendo um desafio desenvolver no terminal móvel arquiteturas semelhantes como encontrados em sistema de desktop padrão, levando em consideração os baixos recursos do dispositivo móvel (Pawar 2007). A arquitetura básica para SOA é construída através de três componentes (Amorim 2004):

- **Service Requestor** - entidade que requisita certas funcionalidades para realizar alguma tarefa, aplicação ou serviço que invoca ou inicializa uma interação com algum serviço;
- **Service Provider** - é a plataforma acessada na solicitação do serviço, entidade que cria e disponibiliza o serviço, também por fazer a descrição do serviço e por publicá-lo em um registro central;
- **Service Registry (Broker)** - localização da descrição do serviço, ou seja, local onde o *Service Provider* publicou a descrição do seu serviço;

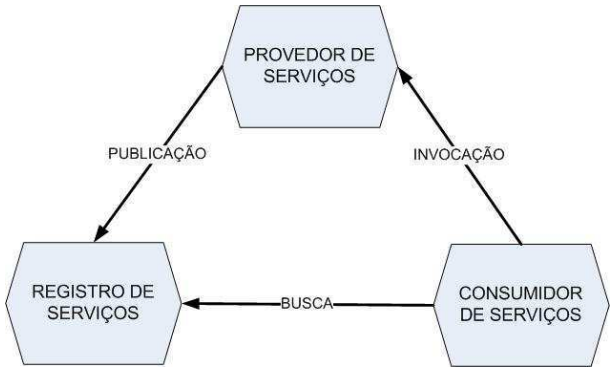

Figura 2.1: Arquitetura Básica de um SOA

2.1.3 Operações

Os componentes de SOA interagem entre si através de um conjunto de operações (Leymann 2002), conforme podemos observar na Figura 2.1, e que são descritas a seguir:

- Publicação - registro da descrição do serviço em diretórios de serviços, abrangendo o registro de suas capacidades, interface, comportamento e da qualidade oferecida por ele;
- Descoberta - busca automática por serviços registrados em diretórios de serviços, desde que satisfaçam os critérios desejados e possam ser usados em um processo de negócio, levando em consideração a descrição do serviço publicado;
- Invocação - nesta operação o serviço requisitado invoca ou inicializa uma interação com o serviço em tempo de execução usando os detalhes obtidos na descoberta do serviço;

2.2 Mobile Host (Mobile Web Service Provisioning)

Tradicionalmente, os sistemas móveis têm sido concebidos como sistemas cliente-servidor em que ***thin clients*** tais como PDAs ou telefones são capazes de usar conexões sem fio para obterem acesso a recursos (dados e serviços) fornecidos por

servidores centrais (Srirama 2006*b*). Com o surgimento de redes sem fio Ad-Hoc e poderosos dispositivos móveis torna-se possível a concepção de sistemas móveis usando uma arquitetura P2P.

Mobile Host é um provedor de Serviços *Light Weight* construído para ser executado em dispositivos móveis como smart-phones e PDAs(Srirama 2006*a*), desenvolvidos como um **Web Service Handler** sendo construído no topo de um Servidor Web normal, o esquema básico de um Mobile Host pode ser visto na Figura 2.2: cujos componentes são descritos a seguir:

- **Interface de Rede**: responsável por receber as requisições e enviar as respostas geradas pelo sistema para o consumidor de serviços;
- **Handler**: verifica o tipo de mensagem recebida e verifica o tipo de tratamento mais adequada a ela;
- **Parser**: extrai o contéudo da mensagem SOAP;
- **Gerenciador de Serviços**: gerencia todas as atividades referentes aos serviços, como descrição, publicação e execução;
- **Base de Dados de Serviços**: local onde os serviços são armazenados para posterior utilização;

Mobile Host abre um novo conjunto de aplicações ainda pouco exploradas (Srirama 2006*b*). Podendo ser usados em domínios como serviços baseados em localização, suporte a comunidade móvel e jogos pervasivos, etc. Também, permitem que pequenos operadores móveis aumentem seus próprios negócios sem recorrer a uma infra-estrutura estacionária. Entretanto, estas novas possibilidades de uso destes dispositivos geram um grande número de questões interessantes para pesquisas as quais necessitam de maiores investigações.

A criação de um "Mobile Host"passa por vários aspectos, onde certas existem poucas pesquisa, o que torna o provisionamento de serviço limitado nos dispositivos. (Srirama 2006*c*) descreve um modelo para o desenvolvimento de um sistema Mobile Host em termos gerais, devendo possuir as seguintes características (Srirama 2006*b*):

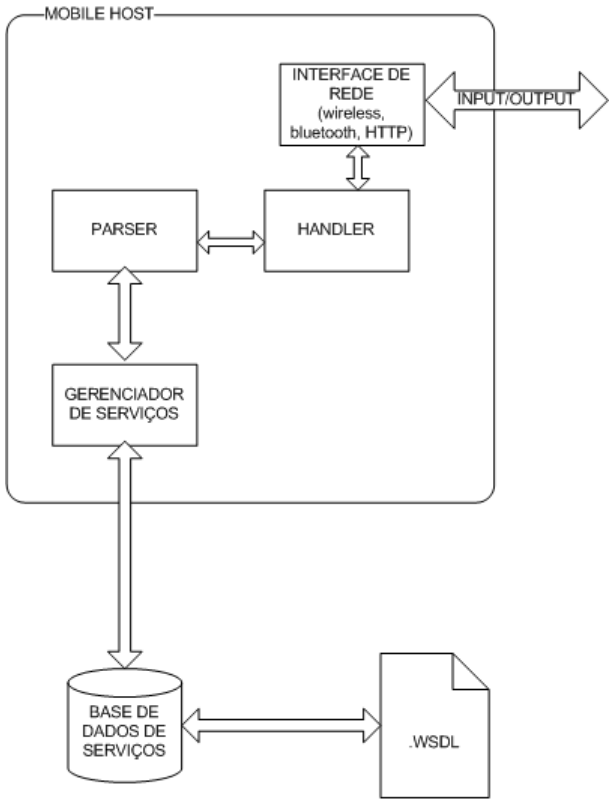

Figura 2.2: Núcleo do Mobile Host Proposto

- Mobile Host deve se mantiver totalmente compatível com as habituais interfaces de SOAs para os clientes não notarem a diferença;
- Conceber ao "Mobile Host"um espaço pequeno em relação ao que está disponível no dispositivo móvel;
- O funcionamento do serviço no dispositivo não deve interferir no funcionamento do aparelho;
- Um Servidor Web normal que manuseie as solicitações de rede;
- Um provedor de serviços básico para tratamentos das requisições dos SOA;
- Habilidade para lidar com pedidos concorrentes;
- Apoio na implantação de serviços em tempo de execução;

- Os Web Services desenvolvidos como um "Mobile Host"devem possuir capacidade para acessar o sistema de arquivos local, ou de quaisquer dispositivos externos como um receptor de GPS, receptor infravermelho, Bluetooth etc., e podem implementar lógica do negócio;

- Capacidade para criar uma interface para sua publicação e distribuição;

2.2.1 Outros Cenários de SOA em Computação Móvel

OMA identifica as arquiteturas direta e indireta de Web Service móveis como modelos para provisão de mobile web services (Alliance 2006). A Figura 2.3 ilustra a diferença entre essas arquiteturas.

A arquitetura indireta é apropriada para os solicitantes sem a plena funcionalidade necessária para interagir com um provedor (Alliance 2006). Esta arquitetura está intimamente associada com o Wireless Application Protocol (WAP), que fornece acesso à Internet dispositivos móveis que não possuem capacidade de se conectar a Internet usando os tradicionais protocolos (Gupta 2001). Esta arquitetura pode ser aplicada nos serviços web, para que um requisitante sem a capacidade de intervir em operações de serviços da web pode se comunicar com um provedor através de um proxy (Alliance 2006). O proxy traduz normalmente entre um protocolo específico do dispositivo móvel e do protocolo SOAP.

A Figura 2.3-A ilustra a utilização da arquitetura indireta quando um requisitante móvel pode processar mensagens SOAP, mas é incapaz de aplicar WS-Security (OASIS 2004) para eles. O requisitante assegura que as mensagens que passam entre ele e o Proxy com um mecanismo de segurança dentro do seu conjunto de capacidades, por exemplo, TLS. O Proxy, por sua vez, assegura estas mensagens de acordo com o WS-Security requisitos do provedor. A arquitetura indireta sofre com alguns inconvenientes, que inclui a exigência de ter confiança no Proxy (Gupta 2001), pois ele é responsável por toda a comunicação entre provedor e consumidor. Os dois modos de segurança empregado com esta arquitetura são a procuração sempre quebrada devido a um modo para o outro é arquivar a ser instanciado. Isso permite que o proxy dê acesso ilimitado a todos os dados que converte a segurança de um modo para o outro. A arquitetura é uma opção direta para a implantação WS-Security para um SOA móvel quando um proxy não pode

ser confiável.

A arquitetura direta, A Figura 2.3-B, exige um requisitante móvel que possua plenamente os requisitos para comunicar-se com um provedor (Alliance 2006). Todos os tratamentos, incluindo a obtenção de mensagens SOAP, devem ser realizados pelo solicitante. Para esse fim, a arquitetura direta não pode suportar tantas requisições concorrentes como a arquitetura indireta porque ela depende da capacidade requisitante. No entanto, a arquitetura direta é a que melhor se direciona para o modelo *end-to-end*, porque a segurança da mensagem SOAP nunca é quebrada durante o seu trânsito entre o requisitante e o provedor. Assim sendo, esta dissertação se baseia sua segurança Web Services móveis sobre a configuração arquitetura direta

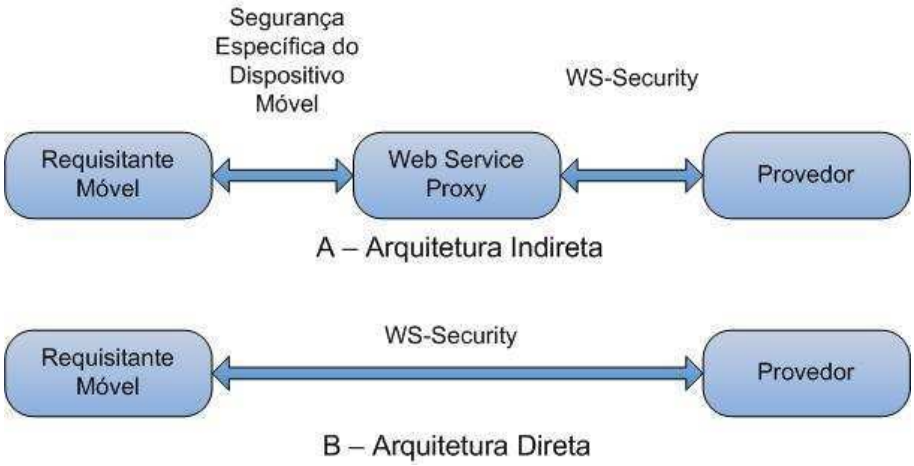

Figura 2.3: Arquitetura de Mobile Web Services

2.2.2 A necessidade de segurança *end-to-end*

Redes celulares podem proporcionar confidencialidade, integridade e autenticação de mensagens transportadas por elas. Um exemplo é a segurança proporcionada pela General Packet Radio Service (GPRS), em Redes GSM (G. Sanders 2003). No entanto, essa proteção dura apenas a extensão da rede celular móvel e no serviço de mensagens da web móvel que pode trafegar para além desta rede (Moyo 2006). As mensagens são, portanto, vulneráveis à atividade maliciosa desde que sai da rede móvel. Logo, percebemos que as mensagens devem ser protegidas além do âmbito da rede móvel.

Segurança de uma rede celular pode ser inadequada devido o uso de protocolos proprietários que não são utilizados fora das redes de telefonia móvel. Logo, para manter a comunicação segura fora dessas redes requer que os SOAs exijam dos participantes a confiança dos provedores de rede móvel na oferta de segurança. Vulnerabilidades relatadas dentro dos mecanismos proprietários de segurança em redes celulares mostram que essa confiança é injustificada (Biryukov 2001). Segurança de mensagens SOAs na web permite que os participantes possam apropriar-se de serviços de proteção de mensagem e selecionem o mecanismo de segurança que ele confia (Moyo 2006).

A questão de confiar cegamente em segurança de redes celulares não é universalmente aplicável porque a segurança de algumas redes celulares está disponível para revisão pública, por exemplo, a especificação aberta do algoritmo Kasumi utilizado por redes UMTS (3G 2007). No entanto, a segurança que opera em um nível inferior ao transporte camada de segurança e sofre com a mesma incapacidade de fornecer segurança end-to-end quando existem intermediários. Entre as entidades e os SOAs exsite dependência nas ligações de comunicação que requerem intermediários confiáveis para a prestação de segurança end-to-end.

Uma vez que um SOA esteja implementado como um Mobile Host, os serviços estão propensos a diferentes tipos de violações de segurança: como ataques de negação de serviço, ataques man-in-a-middle, de intrusão e de spoofing, etc. SOAs em ambiente móvel utilizam tecnologias baseadas em mensagem (como SOAP sobre HTTP) para operações complexas em vários domínios. Entretanto, este tipo de tecnologia por si só não especifica os meios de segurança para a prestação do serviço de comunicação em uma rede de comunicação. Também, podem existir muitos serviços intermediários legítimos na comunicação entre os SOAs que compõe um determinado serviço, o que torna o contexto de segurança um requisito fim-a-fim.

2.3 Segurança

A segurança em redes sem fio é um trabalho em constante evolução. Com tempo e acesso suficientes, um *cracker* persistente provavelmente conseguirá invadir um sistema sem fio. Ainda assim, algumas atitudes precisam ser tomadas para difi-

cultar ao máximo possível o trabalho do intruso, de forma a permitir que serviços básicos de segurança sejam atendidos.

Além dos riscos já comuns nas redes cabeadas que foram incorporadas às redes sem fio, novos[1] surgiram devido às diferenças na estrutura física destas e modo como operam. Assim, toda e qualquer solução direcionados para redes sem fio deve ser construída observando-se estes novos riscos exclusivos a redes sem fio.

2.3.1 Níveis de segurança

A segurança, Figura 2.4, pode ser dividida em três níveis (Candolin 2007):

- **Rede** - é concedida com a proteção da infra-estrutura da rede e incluem serviços como controle de acesso e proteção a ataques de negação de serviços;
- **Comunicação** - provê segurança fim-a-fim entre os nós que estão se comunicando, seus serviços incluem autenticação dos dados originais, confidencialidade e verificação de integridade;
- **Conteúdo** - é concedido com a proteção do conteúdo, tal como informação e serviços, durante o armazenamento e transmissão;

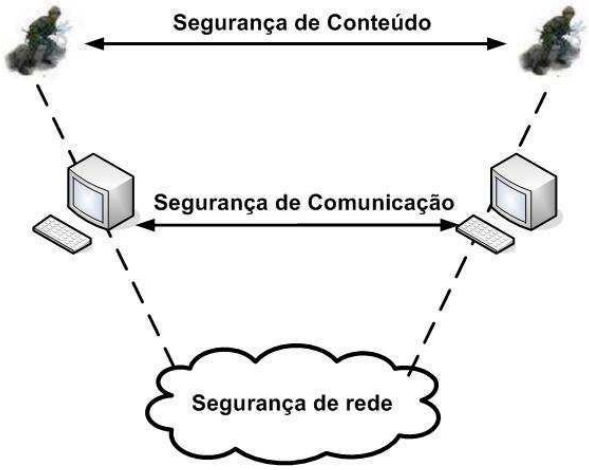

Figura 2.4: Níveis de Segurança

[1] seção 2.3.3

2.3.2 Aspectos de Segurança em redes sem fio

A utilização de uma rede sem fios implica em alguns aspectos especiais em relação à segurança, quando comparada com uma rede com fios. As redes que utilizam fios para interconexão dos computadores possuem características de segurança física inexistentes em redes sem fio, tais como (Buttyán 2002):

- **Vulnerabilidade de canais** - Como em qualquer rede sem fio, as mensagens podem ser escutadas através de portas e mensagens falsas podem ser injetadas na rede sem a dificuldade de ter o acesso físico aos componentes de rede;

- **Vulnerabilidade de nós** - Uma vez que os nodos da rede normalmente não residem nos locais fisicamente protegidos, tais como salas trancadas, eles podem ser mais facilmente captadas e cair sob o controle de um invasor;

- **Ausência de infra-estrutura** - Redes Ad-Hoc, presume-se, operar de forma independente de qualquer infra-estrutura fixa. Isso torna soluções clássicas de segurança baseadas em autoridades de certificação e servidores on-line inútil;

- **Escolha dinâmica de topologia** - Em redes móveis Ad-Hoc, as alterações permanentes da topologia requerem sofisticados roteamentos, a segurança que é um desafio adicional. Uma dificuldade especial que é o roteamento incorreto de informações que podem ser gerados por nodos comprometidos ou como resultado de algumas mudanças topologia, e é difícil distinguir entre os dois casos;

O problema de segurança é tão amplo que não há nenhuma maneira de conceber uma solução geral. Além disso, as diferentes aplicações possuem diferentes requisitos de segurança é uma característica evidente. A complexidade e a diversidade do campo levaram a uma multiplicidade de propostas, que se concentram em diferentes partes do problema domínio. Portanto, para que uma rede sem fios possua as mesmas características de segurança de uma rede com fios, existe a necessidade de inclusão de mecanismos de autenticação de dispositivos e confidencialidade de dados.

2.3.3 Principais problemas em redes sem fio

A - Ameaças comuns a redes cabeadas

Umas das maiores ameaças existentes em uma rede móvel é a possibilidade da instalação de escutas através de portas para o monitoramento de chamadas telefônicas e tráfego de dados, decorrente ao meio pelo qual os dados são transmitidos. Esta ameaça pode ser solucionada em parte com o uso de encriptação. Conseqüentemente, a probabilidade de ameaça depende na resistência de encriptação do algoritmo. Entretanto, esta resistência é uma saída que se torna questionável no sistema GSM (Moyo 2006). Outra ameaça crítica, embora mais hipotética, é alteração do tráfego móvel original. Neste caso o intruso sobrescreve partes dos dados com suas próprias informações. Além disso, existe a possibilidade da criação de mensagens falsas que são enviadas para algum nó da rede móvel como se fosse uma mensagem original e também existe a possibilidade de uma mensagem não ser entregue ao seu destinatário. As principais formas de ameaças existentes em transmissões sem fio são mostradas na figura 2.5.

B - Ameaças próprias das redes sem fio

Análises não autorizadas do tráfego de telecomunicações móveis são facilmente implementadas, mas isto ainda requer dispositivos especiais, conhecimentos específicos e rápido funcionamento. O monitoramento do tráfego entre os dispositivos e a estação base pode-se obter a posição, velocidade, tempo de tráfego, duração, identificação de um dispositivo móvel qualquer. Entretanto, os cenários de exploração pelos intrusos são limitados e o maior benefício de informações são possivelmente informações de localização e perfil do usuário.

C - Ataques mais comuns

O WS-I identifica um conjunto de ameaças que podem impedir a realização dos desafios da confidencialidade integridade, autenticidade e mensagem originais (Schwarz 2007). Estas ameaças são definidos da seguinte forma (Schwarz 2007), cujas contramedidas são apresentadas na Tabela 2.1:

- Alteração da Mensagem: A mensagem é modificada ao se inserir informações, remover ou modificar as informações criadas por outra a origem da informação e enganados pelo receptor como sendo a intenção do autor;

- Confidenlialidade: Informação dentro da mensagem pode ser visualizada por

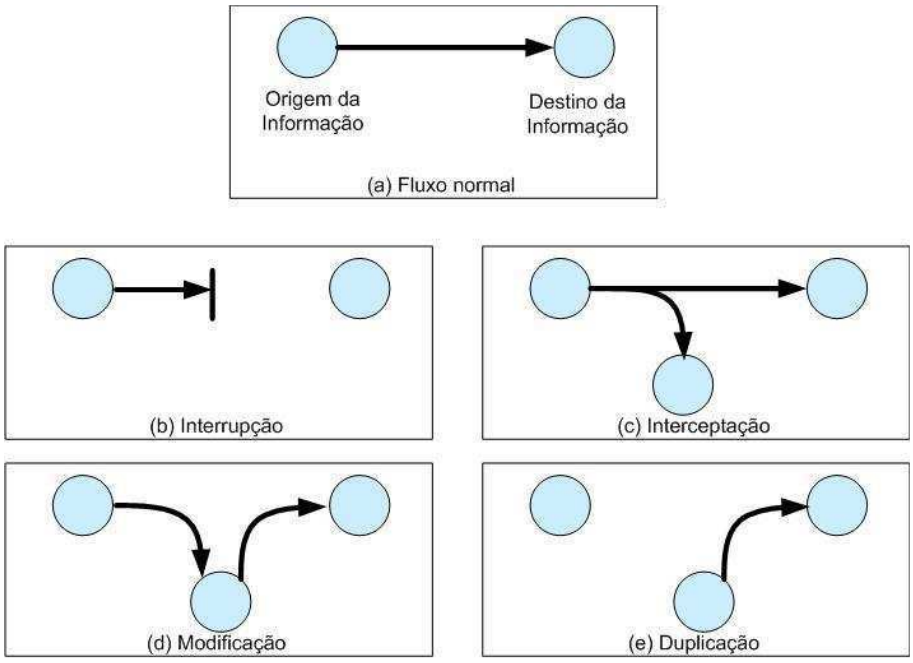

Figura 2.5: Ameaças a redes sem fio

não intencional e não autorizado participantes;

- Man-in-the-Middle: Um participante que como o outro participante para o verdadeiro remetente e o receptor, a fim de enganar os dois participantes;
- Principal Spoofing: A mensagem enviada é o que parece ser de outro principal;
- Forged claims: Uma mensagem é enviada na qual as alegações de segurança são forjados em um esforço para obter acesso a outras informações não autorizadas;
- Replay of Message Parts: Uma mensagem é enviada, que inclui porções de outra mensagem em um esforço para obter acesso não autorizado a informações ou outra forma de provocar o receptor para ter alguma ação;
- Replay: Uma mensagem inteira, é mandada novamente por um atacante;
- Denial of Service: (O) atacante cria uma grande quantidade de trabalho e força (o) sistema sujeito ao ataque realize um grande volume de trabalho e se torne indisponível;

	Data Confidentiality	Data Integrity	Peer Authentication	Data Authentication	Message Uniqueness
Message Alteration		*			
Confidentiality	*				
Falsified Messages			*	*	
Man in the Middle			*	*	
Principal Spoofing			*	*	
Forged Claims			*	*	
Replay of Message Parts			*	*	*
Replay					*
Denial of Service					*

Tabela 2.1: Mapeamento de mensagens de Web Services: Metas e Ameaças, Adaptado de (Schwarz 2007)

A identificação de ameaças para integridade e confidencialidade são simples, dado que as ameaças são contraditórias com os desafios. O problema de ameaças para a autenticação surgem na criação da identificação durante uma operação SOA. As ameaças à mensagem são baseadas na ausência de discriminação entre as mensagens. A Tabela 2.1 pode dar a percepção incorreta de que ameaças podem ser tratadas de forma isolada no âmbito de cada desafio. No entanto, alguma sobreposição entre os desafios possam ser encontrados ao amenizar certas ameaças.

Serviço	Descrição	Ferramenta
Confidencialidade	Serviço usado para prevenir descoberta de informação por entidades não autorizadas;	Cifragem
Integridade de Dados	A integridade de dados se preocupa com a alteração não autorizada da informação, incluindo a proteção contra inserção de dados adicionais, destruição ou modificação de todo ou parte dos dados. Seu objetivo é verificar se um dado não foi alterado;	Resumo Digital
Autenticação	Serviço usado para estabelecer a origem da informação, ou seja, o serviço de autenticação verifica a identidade do usuário ou sistema que criou estainformação (uma transação ou mensagem);	Senhas
Não-Repúdio	Serviço que é usado para fornecer a prova da integridade e a origem do dado de tal maneira que a integridade e a origem possam ser verificadas através de terceiros;	Certificado Digital

Tabela 2.2: Aspectos de Segurança tratados pela Criptografia

2.4 Criptografia

Criptografia é a área da matemática e da computação que trata de problemas relacionados à segurança da informação. A criptografia pode ser usada para executar vários serviços básicos de segurança tais como: confidencialidade, integridade dos dados, autenticação e não repúdio, conforme podemos observar na Tabela 2.2. Estes serviços podem também ser requisitados para projetos de chaves criptográficas. Dentre estas, encontra-se a infra-estrutura de chaves públicas, tecnologia intimamente relacionada com a ferramenta de validação de certificados digitais a qual está relacionada com este trabalho. Nos próximos tópicos será mostrado como cada aspecto de segurança pode ser utilizado.

2.4.1 Criptografia Simétrica

Na criptografia simétrica ou de chave privada existe apenas uma chave secreta que é compartilhada pelo emissor e pelo receptor da mensagem, ou seja, a chave

que cifra é a mesma que decifra, Figura 2.6. Para manter a confidencialidade a chave deve ser de conhecimento somente desses atores e que não se repita para pares de atores diferentes. Esta forma de criptografia tem como principal vantagem a rapidez no cálculo das operações de cifra/decifra. No entanto, tem como desvantagem o fato de requerer n *(n -1) / 2 chaves para n atores e o problema do modo como as chaves devem ser distribuídas pelos vários intervenientes sem que seja quebrada (Bernardino 2004).

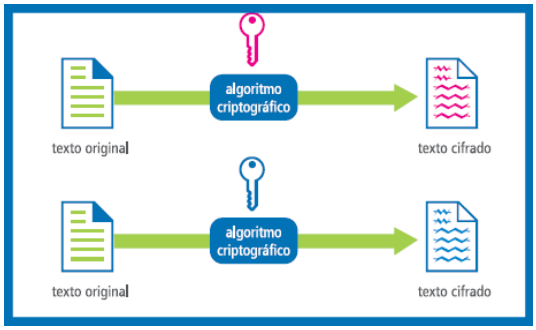

Figura 2.6: Criptografia Simétrica

2.4.2 Criptografia Assimétrica

A criptografia assimétrica ou de chave pública utiliza um par de chaves distintas em que, embora não se consiga obter uma chave a partir de outra, estas se encontram matematicamente relacionadas, conseguindo uma decifrar aquilo que a outra cifrou, Figura 2.7. Esta característica vai permitir que uma das chaves seja publicada, a chave pública, pelo que só serão necessárias n chaves para n atores, uma vantagem em relação à criptografia simétrica. O fator de sucesso deste tipo de cifra é manter-se a chave privada protegida e só do conhecimento do seu titular. Preenchendo este requisito, é possível obter a autenticação de conteúdos e de autoria. A grande vantagem deste sistema é permitir que qualquer pessoa possa enviar uma mensagem secreta, apenas utilizando a chave pública de quem irá recebê-la. Como a chave pública está amplamente disponível, não há necessidade do envio de chaves como é feito no modelo simétrico. Uma desvantagem que este tipo de cifra apresenta em relação à simétrica, resume-se ao fato do seu desempenho ser mais lento, por utilizar um processo algorítmico mais complexo.

Por este motivo muitas vezes estes dois métodos de cifragem operam em conjunto (Bernardino 2004).

Figura 2.7: Criptografia Assimétrica

2.4.3 Confidencialidade

A cifragem de um documento garante que só o destinatário do documento o consegue ler na sua forma original, sendo incompreensível para terceiros que o consigam interceptar.

Com o sistema de chave pública, tanto o remetente como o destinatário podem, de uma forma segura, trocar informação privada numa transação eletrônica. Portanto, para proteger a informação a ser enviada, o remetente utiliza a chave pública do destinatário para cifrar os dados. Quando o destinatário recebe os dados, pode decifrar essa informação com a sua chave privada, e assim obter acesso à informação original em formato legível. Enquanto o destinatário mantiver a sua chave privada segura a informação que é transmitida muito dificilmente será acessível a outros.

Desta forma, com o objetivo de ninguém mais ter acesso ao conteúdo da mensagem, a não ser o destinatário, isto é, para garantir a confidencialidade na transmissão de informação, o emissor da mensagem adiciona a assinatura digital criada à mensagem original, que cifra, por sua vez, com a chave pública do receptor. Assim. uma mensagem eletrônica confidencial e assinada digitalmente é criada.

Após a recepção da mensagem, caso a mensagem esteja cifrada, o receptor acede ao texto utilizando a sua chave privada. A fim de obter a assinatura digital original em formato legível, o receptor decifra o código Hash recebido com a chave

pública do emissor (Valor A). Para verificar a exatidão da assinatura digital e a integridade da informação, o receptor gera um código Hash 2.4.4 com a mensagem original que recebeu (Valor B), e compara este código com o que obteve da decifragem da assinatura digital recebida (Valor A). Se o receptor validar positivamente a assinatura digital com a chave pública do emissor, temos a garantia de que a mensagem foi enviada pelo dono da chave privada e que, simultaneamente, não foi alterada no seu trajeto.

2.4.4 Função *Hashing*

O termo hash significa misturar, confundir. As funções hash calculam um valor digital para uma mensagem ou *stream*[2]. São muito utilizadas em assinaturas digitais porque oferecem agilidade, pois os algoritmos assimétricos são lentos. Garantem integridade aos dados visto que não existe um mesmo valor de hash para textos distintos, isto é qualquer alteração que ocorra no texto original gerará um valor modificado. Este método funciona como uma impressão digital da mensagem ou do documento, as funções hashing mais utilizadas são:

- **(Message-Digest algorithm) MD5**: Criado em 1991 por Ron Rivest e trata-se de uma função de espalhamento unidirecional. É um algoritmo rápido, simples e seguro. Gera um valor hash de 128 bits, o que pode ocasionar uma certa preocupação em relação a sua segurança, pois se trata de um valor pequeno;

- **Secure Hash Algorithm**: Denominado de SHA-1, consiste em uma função de espalhamento unidirecional que produz um valor hash de 160 bits. Internamente se assemelha com o MD5, oferecendo um nível de segurança maior devido ao tamanho da hash;

2.4.5 Assinatura Digital

O mesmo método de autenticação dos algoritmos criptográficos de chave pública opera em conjunto com uma função resumo, também conhecido como função de hash, é chamada de assinatura digital. O processo de geração de uma Assinatura

[2]seqüência de caracteres qualquer

Digital pode ser visto na Figura 2.8. A assinatura digital surge como uma medida de segurança contra fraudes eletrônicas como:

Figura 2.8: Processo de criação de uma assinatura digital

- **Personificação** - Um elemento se fazendo passar por outro, perante terceiros, numa troca de informação;
- **Alteração de dados** - Modificação de alguns ou de todos os dados transmitidos numa sessão;

Desta forma, a assinatura digital de um documento garante:

- A autenticação da identidade da entidade que assinou o documento (o emissor do documento;
- A não alteração (acidental ou maliciosa) do documento durante a sua transmissão, i.e., protege a integridade do documento;
- O não repúdio do documento por parte do emissor, i.e., o emissor não pode reclamar que não foi ele que assinou o documento;

O resumo criptográfico é o resultado retornado por uma função de hash. Esse pode ser comparado a uma impressão digital, pois cada documento possui um valor único de resumo e até mesmo uma pequena alteração no documento, como a inserção de um espaço em branco, resulta em um resumo completamente diferente.

A vantagem da utilização de resumos criptográficos no processo de autenticação é o aumento de desempenho, pois os algoritmos de criptografia assimétrica são muito lentos. A submissão de resumos criptográficos ao processo de cifragem com a chave privada reduz o tempo de operação para gerar uma assinatura por serem os resumos, em geral, muito menores que o documento em si. Assim, consomem um tempo baixo e uniforme, independente do tamanho do documento a

ser assinado. Na assinatura digital, o documento não sofre qualquer alteração e o hash cifrado com a chave privada é anexado ao documento.

Para comprovar uma assinatura digital é necessário inicialmente realizar duas operações: calcular o resumo criptográfico do documento e decifrar a assinatura com a chave pública do signatário. Se forem iguais, a assinatura está correta, o que significa que foi gerada pela chave privada corresponde à chave pública utilizada na verificação e que o documento está íntegro. Caso sejam diferentes, a assinatura está incorreta, o que significa que pode ter havido alterações no documento ou na assinatura pública.

Figura 2.9: Processo de validação das mensagens assinadas digitalmente

2.4.6 Certificado Digital

A certificação digital atesta a identidade de uma pessoa ou instituição na internet por meio de um arquivo eletrônico assinado digitalmente. Seu objetivo é atribuir um nível maior de segurança nas transações eletrônicas, permitindo a identificação inequívoca das partes envolvidas, bem como a integridade e a confidencialidade dos documentos e dados da transação, a figura 2.10 mostra um exemplo de Certificado que pode ser instalado em uma computador.

Um certificado digital normalmente é usado para ligar uma entidade a uma chave pública. Para garantir digitalmente, no caso de uma Infra-estrutura de Chaves Públicas (ICP), o certificado é assinado pela Autoridade Certificadora que o emitiu e no caso de um modelo de Rede de Confiança (Web of Trust) como o PGP[3], o certificado é assinado pela própria entidade e assinado por outros que

[3]**Pretty Good Privacy** utiliza criptografia para proteger a privacidade de e-mail e de

dizem confiar naquela entidade. Em ambos os casos as assinaturas contidas em um certificado são atestados por uma entidade que confirma os dados contidos naquele certificado.

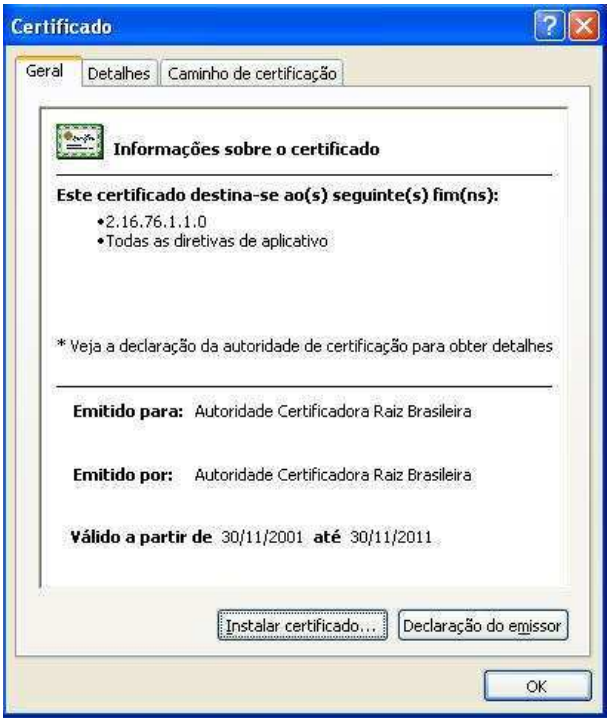

Figura 2.10: Exemplo de Certificado Digital emitido pela ICP-Brasil

Um certificado normalmente inclui:

1. Informações referentes à entidade para o qual o certificado foi emitido;
2. A chave pública referente à chave privada de posse da entidade especificada no certificado;
3. O período de validade;
4. A localização do "centro de revogação";
5. A(s) assinatura(s) da(s) AC/entidade(s) que afirma que a chave pública contida naquele certificado confere com as informações contidas no mesmo;

arquivos guardados em um computador de um usuário. O PGP pode, ainda, ser utilizado como um sistema à prova de falsificações de assinaturas digitais permitindo, desta forma, a comprovação de que arquivos ou e-mails não foram modificados;

Formato do certificado	Versão 3
Nº de Série do certificado	123456789
Identificador do algoritmo de assinatura da EC	RSA com MD5 RSA com MD5
Endereço X.500 do emissor	C=PT, o=EC
Período de validade	start=01/08/2008, expiry=01/08/2009
Nome X.500 do utilizador	c=PT, o=organização, cn=João Silva, ...
Chave pública do utilizador e método criptográfico utilizado	C.P.U. + RSA com MD5 C.P.U. + RSA com MD5

Tabela 2.3: Informação básica de um certificado digital X.509 V3

Podemos verificar exemplos destes dados na Tabela 2.3.

2.5 Padrões de Segurança com XML

O protocolo SOAP não implementa nenhum mecanismo segurança, apenas utiliza mecanismos já existentes na infra-estrutura de rede que desempenhem essa função. SSL é o principal desses mecanismos, mas o fato de não prover segurança fim-a-fim, criptografia seletiva, autenticação, integridade de dados e não-rejeição, o torna não aplicável a todas as situações em que SOAs precisem ser seguros. Por essas razões novas especificações de segurança têm sido propostas com o intuito de promover segurança específica a Web Services. Mas falta ainda um acordo sobre qual desses mecanismos devem ser adotados como padrão (de Carvalho Ferreira 2005). Alguns desses padrões são listados a seguir:

- XML Signature: É um padrão aberto que permite especificar regras de processamento para criação e representação de assinaturas digitais em porções selecionadas de um documento XML, através de tags XML específicas e de criptografia assimétrica, com o intuito de promover integridade e não-repúdio a uma mensagem SOAP (et al Potts 2003). O XML Signature garante que os dados presentes no pacote não foram alterados antes de chegar ao seu destino. Ele utiliza certificados X.509 para assinar os pacotes SOAP.

- XML Encryption: Especifica um processo para criptografia de dados e sua representação em formato XML. Os dados podem ser arbitrários (incluindo um documento XML completo), elementos XML, ou conteúdos de elementos XML. Um documento XML que utiliza a XML Encryption pode ser visto por qualquer utilizador, mas apenas o proprietário da chave de decodificação conseguirá compreender o conteúdo do que foi criptografado (et al Potts 2003).

- SAML: É um padrão emergente para a troca de informação sobre autenticação e autorização, solucionando um importante problema das aplicações, que é a possibilidade de consumidores transportarem seus direitos entre diferentes Web Services. Isto é importante para aplicações que integrem um número de Web Services para formar uma aplicação unificada (de Carvalho Ferreira 2005).

- XKMS: Fornece um protocolo, baseado em SOAP, para gerenciamento de chaves públicas, incluindo funções para obtenção de informações sobre chaves, registro, verificação e revogação de permissões. O XKMS tem a função principal de fornecer suporte ao gerenciamento de chaves para tecnologias como **XML Signature** e **XML Encryption**, mas também pode ser utilizado para suportar outras tecnologias de chaves públicas (et al Potts 2003).

- WS-Security: WS-Security propõe um conjunto de extensões no cabeçalho do envelope SOAP para inserir dados relativos a segurança através de especificações como**XML Signature** e **XML Encryption**, bem como os padrões que surgirem futuramente. O objetivo do WS-Security é fornecer a estrutura de uma solução de segurança de Web Services completa, que possa ser instanciada às necessidades do implementador (et al O'Neill 2003). Mas o maior valor que a especificação WS-Security traz é a organização do cabeçalho SOAP, imprescindível para o estabelecimento de um padrão que transforme a segurança de Web Services de uma série de soluções personalizadas em um método mais unificado de transferir informações com segurança por meio da Internet (et al Potts 2003).

- UsernameToken: este elemento provê uma construção que pode ser utilizada para acrescentar tokens com informações específicas para autorização e aut-

enticação de usuários. Tipicamente os elementos filhos desta construção são Username e Password, mas elementos customizados podem ser adicionados (Erl 2005). UsernameToken é uma das formas de identificar o requisitante (consumidor) pelo seu username e por meio de uma senha para autenticar aquela identidade com o sistema provedor;

2.6 Resumo

Neste capítulo foram apresentados os principais conceitos que serão discutidos no decorrer desta dissertação, tais conceitos são imprescindíveis para o bom entendimento deste trabalho. No próximo capítulo serão descritos um conjunto de trabalhos que foram levantados durante as pesquisas iniciais e que têm ralação com o trabalho que está sendo aqui descrito.

Capítulo 3

Trabalhos Relacionados

3.1 SOA em MANET

O objetivo de (Halonen 2006) é fornecer SOA em uma MANET (Mobile Ad Hoc Network), onde sua implementação se baseou no protocolo de roteamento OLSR (Optimized Link State Routing) [1]. Como ilustra a Figura 3.1, a diferença entre a abordagem tradicional (a), onde todos os serviços são publicados em um servidor central, e Ad Hoc ou Mesh (b), sem um servidor central, o protocolo de roteamento, no caso OLSR, é de extrema importância.

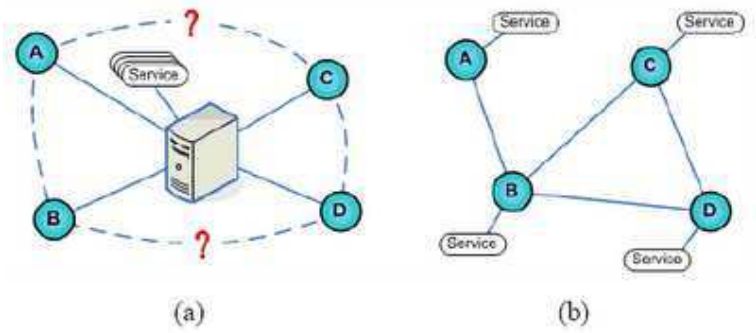

Figura 3.1: SOA como um serviço de registros convencional centralizado (a) e em uma completa rede distribuída mesh (b), extraído de (Halonen 2006)

[1]Protocolo pró-ativo, guiado por tabela e que utiliza uma técnica chamada de *multipoint relaying for message flooding* (Veiculação Multiponto para Inundamento de Mensagens) (e P. Jacquet 2003)

A Figura 3.2 mostra os três principais componentes para seu padrão, o protocolo de roteamento ad hoc (OLSR), a camada SOA, e a camada Services/Appplication. O protocolo de roteamento pro-ativo OLSR que permite de realizar a descoberta de serviços proativamente, tão logo os serviços estejam visíveis durante o tempo. A implementação da camada SOA é feita com um olsrd plug-in, sendo responsável somente por distribuir a arquitetura orientada a serviços na rede mesh atual, transferindo as mensagens de acordo com o protocolo OLSR, irrelevante ao conteúdo. A camada SOA não é responsável por fornecer nenhum serviço ou aplicação usuário, sua principal tarefa é criar as mensagens SOAP, processar recebimento de mensagens e manter a representação dos serviços remotos. Além disso, o trabalho realizado pelos autores provê a criação da descrição dos serviços que rodam em um ponto da rede, e assim permite saber quais os serviços estão disponíveis em cada nó.

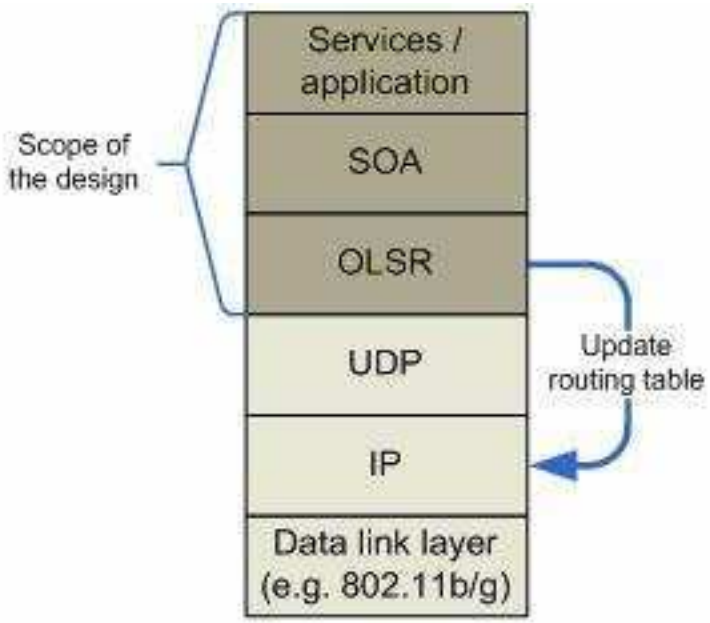

Figura 3.2: Pilha de protocolos da Implementação do protótipo, extraído de (Halonen 2006)

3.2 SOA para suporte em Dispositivos Móveis

(SánchezNilsen 2006) investiga os mecanismos para suporte dinâmico para o contexto de m-SOA com o objetivo de prover disponibilidade de serviços e descoberta dinâmica para usuários móveis em qualquer lugar e qualquer hora discutindo as características dos SOAs em redes sem-fio e correntes infra-estrutura de descoberta de serviços, onde os dipositivos atuam somente como consumidores de serviços. Neste contexto, os autores propõem uma arquitetura que atenda aos seguintes requisitos: (1) integração dinâmica de novos serviços por provedores, (2) descoberta dinâmica de serviços disponíveis e (3) o uso de um software de fonte aberto para desenvolver a solução.

Além disso, (SánchezNilsen 2006) investiga a disponibilidade de tecnologias para dispositivos móveis e paradigmas SOAP para redes sem-fio; propondo uma arquitetura m-services para suporte dinâmicos e escaláveis serviços móveis. (Sánchez-Nilsen 2006) propõe uma entidade de gerenciamento de serviços com um serviço de registro como uma camada intermediária entre provedores de serviços e usuários móveis. Este gerenciamento é responsável pela coordenação de interações entre provedores de serviços e usuários móveis. Essa interação envolve como produzir serviços de informação, disponibilizando e entregando os serviços para os usuários móveis em qualquer momento. (SánchezNilsen 2006) também investiga o usos de interfaces de invocação dinâmica como mecanismo de comunicação entre serviços de descrição e invocação em tempo de execução.

Basicamente a arquitetura proposta se divide em quatro componentes principais: (1) provedores de serviço, (2) o gerenciador de serviços, (3) os clientes e (4) o registro de UDDI, que são descritos a seguir. A Figura 3.3 apresenta o modelo da arquitetura proposta pelos autores.

- Provedores de Serviço: são os principais componentes que implementam e oferecem os serviços disponíveis. Estes serviços são descritos utilizando a linguagem WSDL;

- Clientes: representam os usuários de dispositivos móveis que estão interessados nos serviços disponíveis no ambiente em que ele se encontra, estes serviços devem ser fornecidos aos clientes de uma forma transparente para si;

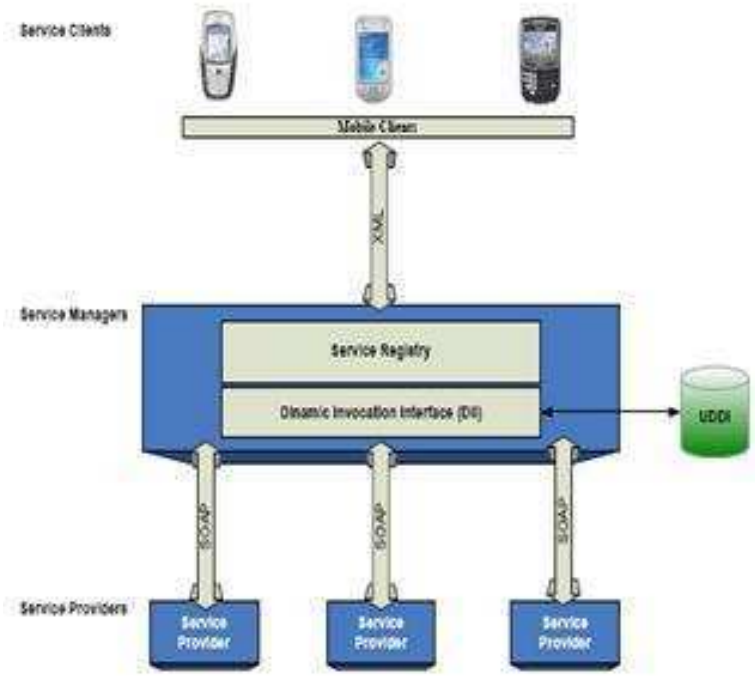

Figura 3.3: Suporte em Dispositivos Móveis, extraído de (SánchezNilsen 2006)

- Gerenciador de Serviço: é a camada intermediária entre os dispositivos móveis e o provedor de serviços. É o responsável pelo fluxo das informações entre ambos os componentes. Um gerenciador de serviços é um Web Service que utiliza uma invocação dinâmica de serviços como mecanismo de comunicação entre os diferentes provedores de serviços;
- Registro UDDI: estes registros são utilizados para a localização de novos serviços. A descoberta de serviços é computada em tempo de execução pelo gerenciador de serviços, quando o usuário envia uma requisição de um novo serviço;

3.3 Arquitetura SeSCO: Composição de Serviços em Ambientes Pervasivos

(Kalasapur 2005) propõe um mecanismo hierárquico para superposição de serviços, baseado nas capacidades dos dispositivos que hospedam serviços. (Kalasapur 2005) construiu seu Middleware chamado PICO (Pervasive Information Com-

munities Organization) para habilitar a transformação de recursos em serviços. Baseando-se na generalização hierárquica "service overlay", desenvolvendo um mecanismo de composição de serviços que é capaz de tecer dinamicamente serviços complexos utilizando a disponibilidade de serviços básicos.

Basicamente os elementos que foram representados na arquitetura são os seguintes: (1) ***camileuns*** - entidade que representa um dispositivos de forma abstrata, (2) ***delegents*** - entidades de software que representam as funcionalidades presentes nos dispositivos como se fossem serviços, e (3) as ***comunidades*** - organização lógica de um número de delegents que trabalham para um objetivo comum, ou um serviço composto.

Na arquitetura cada *Camileus* possui caracteristicas, que são os serviços que ele disponibiliza, que são disponibilizadas para os demais *Camileus* durante abstração para representar suas características na rede, criando uma comuidade de recursos disponíveis na rede. Logo, a arquitetura depende do seu contexto em determinado instante. A consciência de contexto não estará ligada a uma infraestrutura de software específica ou ambiente físico. A consciência de contexto estará disponível nos dispositivos ambientes, através da transmissão em uma rede Ad-Hoc, que contém as fontes de contexto e os consumidores. A Figura 3.4 apresenta o middleware construído para a arquitetura.

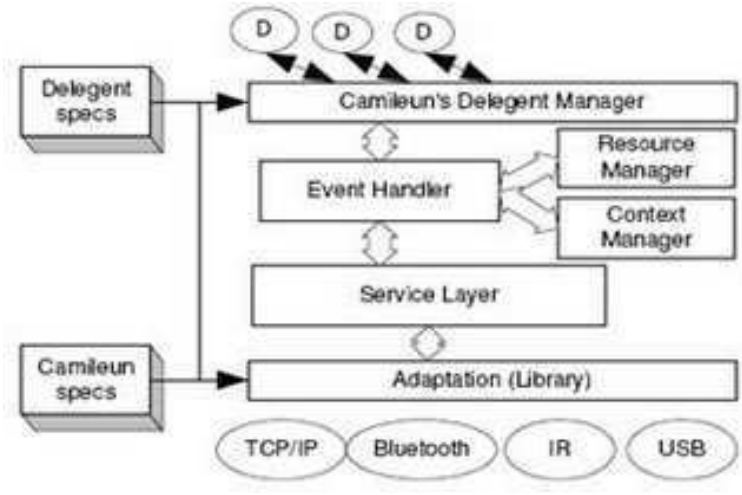

Figura 3.4: Pilha do middleware PICO, extraído de (Kalasapur 2005)

O middleware para a arquitetura PICO é dividido em camadas, com operações diferentes que podem ser executadas em diferentes camadas do middleware. Basea-

dos na capacidade dos dispositivos podem existir três versões diferentes de execução, que essencialmente se diferem nas operações que eles podem executar. Para dispositivos com poucos recursos, pode ser instalada uma versão mínima do middleware, que contém uma pilha reduzida de instruções. Esta versão mínima é capaz de avisar e participar da descoberta de serviços, porém não é possível administrar tarefas mais complexas como orquestração e composição de serviços. A segunda versão contém as mesmas operações da versão anterior, porém é instalado em dispositivos que são possivelmente móvel, como PDAs, laptops etc. A versão completa do middleware é instalada em dispositivos que possuem mais recursos, como PCs, servidores. Estes dispositivos podem executar tarefas mais complexas como descoberta e composição de serviços.

O foco dos autores no trabalho está relacionado a camada de serviços, a Figura 3.5 apresenta os componentes que compõem a camada de serviços desenvolvida, esta figura detalha a camada de serviços representada no middleware da Figura 3.4. A camada de serviços do middleware é responsável por serviços básicos relacionados com as operações do sistema, como exposição do serviço, descoberta e composição.

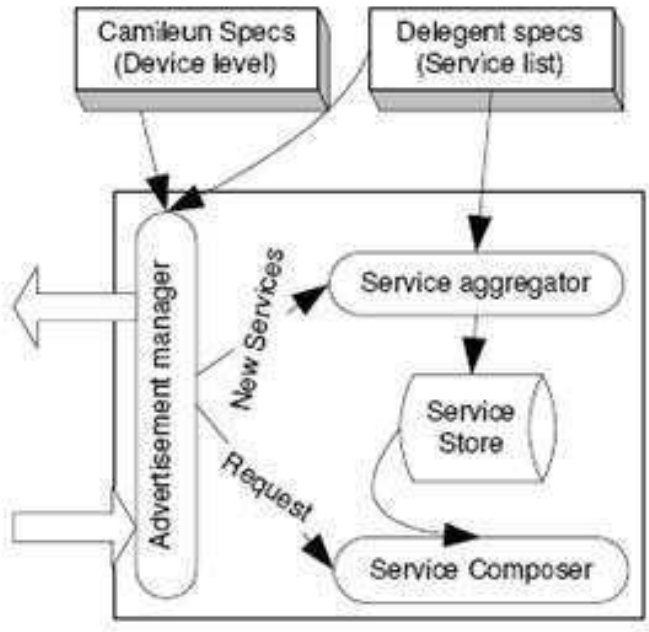

Figura 3.5: Componentes da Camada de Serviços, extraído de (Kalasapur 2005)

Conforme mostra a Figura, o gerenciamento de publicação envia os serviços

disponíveis periodicamente e também responde a publicações externas e a pedidos. O serviço de agregação é responsável por armazenar os novos serviços que estão disponíveis. Já o serviço de composição é responsável por gerenciar a composição dos serviços e as informações pertinentes a cada serviço. De forma simplificada os autores apresentaram uma arquitetura para a composição de serviços, principalmente a serviços inerentes a multimídia em ambientes ubíquo.

3.4 Arquitetura *Mobile Host*

Em (Srirama 2006*c*) é explorado a idéia que o provisionamento de SOA pode ser feito em dispositivos móveis. Segundo (Srirama 2006*c*), a arquitetura básica de um Terminal móvel como um provedor de serviços pode ser estabelecido através de um: Service Request, Service Provider (Mobile Host) e Service Registry. Sendo o Service Provider implementado no dispositivo móvel onde é utilizado um padrão para descrever o serviço (WSDL), e um padrão (UDDI) é usado para publicar e remover publicação os serviços. Logo se pode perceber que esta arquitetura segue o padrão de sistemas desktops, mas levando em conta os baixos recursos do dispositivo.

De acordo com (Srirama 2006*c*) com o advento das redes wireless Ad-Hoc e aumento das capacidades dos aparelhos permite que sejam criadas redes P2P totalmente puras. Logo, a sua arquitetura apresenta uma abordagem para o fornecimento Web Services móveis. A Figura 3.6 traz a configuração básica para sua construção.

O núcleo da arquitetura proposta por (Srirama 2006*c*) consiste em: uma *interface de rede* responsável por receber as requisições e enviar as respostas aos consumidores, *Request Handler* que extrai o conteúdo da requisição e o envio para o *Service handler* que acessa a base de dados de serviços e executa a solicitação, conforme podemos observar na Figura 3.7. Na interface HTTP, Mobile Host fica esperando por requisições HTTP GET/POST em soquete do servidor. Quando Mobile Host recebe uma requisição, o soquete do servidor a aceita cria um soquete para a comunicação e inicia uma nova thread de execução criando uma instância do manipulador de requisições (*request handler*). Este extrai a mensagem de entrada do canal de entrada do soquete e checa por requisições para

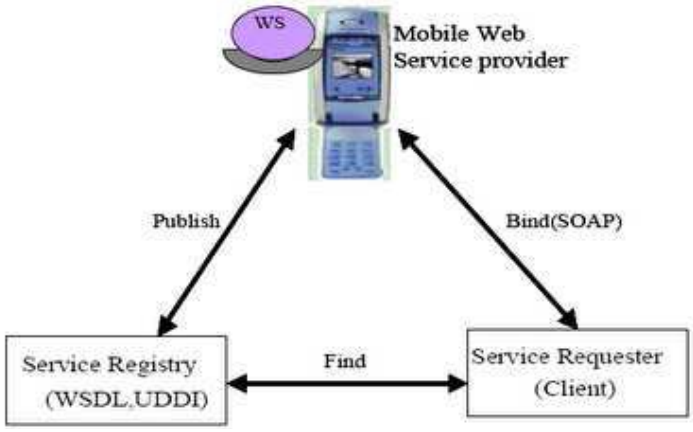

Figura 3.6: Configuração Básica de um Mobile Host, extraído de (Srirama 2006b)

o Web Service. Em caso afirmativo, o manipulador de requisições as processa e envia as respostas escrevendo no canal de saída do soquete.

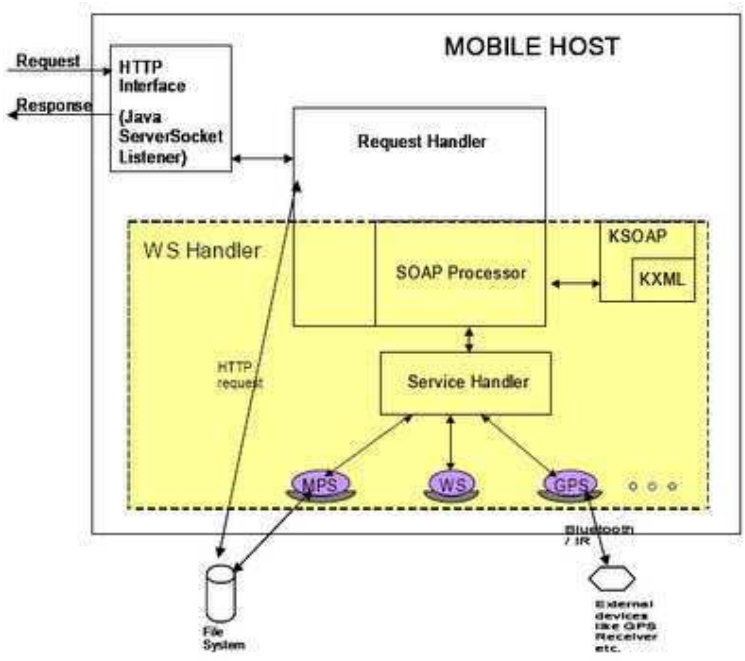

Figura 3.7: Núcleo da Arquitetura Mobile Host, extraído de (Srirama 2006b)

Se a mensagem comprime uma requisição Web Service, o componente manipulador de Web Service (Web Service Handler) processa a mensagem, lendo o corpo da mensagem HTTP e deserializando a requisição SOAP de objetos Java através do processador SOAP. Esses objetos são passados para o manipulador de

serviço (service handler), que extrai os detalhes do serviço e invoca o respectivo serviço. Os Web Services implementados no Mobile Host podem acessar o sistema de arquivos local ou qualquer dispositivo externo, como um receptor GPS, usar infravermelho, bluetooth, etc., e podem implementar lógica de negócio. O manipulador de requisições serializa a resposta e prepara uma mensagem de resposta HTTP, que retorna ao cliente como uma resposta HTTP pelo canal de saída do soquete.

3.5 Resumo das Tecnologias Apresentadas

A Tabela 3.1 mostra um resumo dos trabalhos aqui apresentados, para tanto um conjunto de critérios foram tomados como base nas principais características que um SOA deve possuir, que são: Provisão de Serviços, Publicação e Descoberta de Serviços, Descrição do serviço em formato WSDL, uso do padrão XML, utilização de algum tipo de Serviço de Segurança, criação de serviços pela própria ferramenta, gerenciamento entre o consumidor e o provedor de serviços, parser das mensagens recebidas para que o serviço solicitado seja executado, Composição de serviços em tempo de execução e criação e envio de mensagens do provedor para o consumidor. Observamos que entre os diversos trabalhos encontrados, a maioria dos pontos necessários para uso desta tecnologia são abordados, além do uso de padrões abertos, o que mostra uma preocupação com a portabilidade. Entretanto percebe-se que um ponto não tratado por eles é o foco na segurança dos serviços que serão disponibilizados, requisito que será enfatizado pela arquitetura que será apresentada [2].

3.6 Resumo

Neste capítulo foram apresentados um conjunto de trabalhos que possuem certa relação com o problema abordado nesta dissertação, com o intuito de ter ciência com as pesquisas que estão sendo feitas, para efeito de comparação com a proposta de está sendo apresentada e avaliar as contribuições que estão sendo inseridas por este trabalho. Também, uma comparação entre estes trabalhos e a solução pro-

[2] vide Capítulo 4

	Halonen	Sánchez-Nilsen	Kalasapur	Srirama
Provisão	Sim	Sim	Sim	Sim
Descoberta e Publicação	Sim	Sim	Sim	Sim
Descrição	Sim	Sim	Sim	Sim
Formato XML	Sim	Sim	Sim	Sim
Serviço de Segurança	Não	Não	Não	Não
Gerenciamento	Sim	Sim	Sim	Sim
Criação de Serviços no dispositivo	Não	Não	Não	Não
Parser de Mensagens	Sim	Sim	Sim	Sim
Composição	Não	Sim	Sim	Não
Criação de Mensagens	Sim	Sim	Sim	Sim

Tabela 3.1: Resumo dos Trabalhos Relacionados

posta foi apresentada. No próximo capitulo será descrita a arquitetura proposta e os principais pontos da sua modelagem.

Capítulo 4

Arquitetura Proposta

Este capítulo apresenta a proposta de uma arquitetura para um Framework voltado para construção de serviços, baseado na arquitetura SOA, em ambiente de computação móvel, onde será criado um arcabouço onde estes serviços serão instalados. Uma visão geral da arquitetura proposta e do seu funcionamento será apresentada neste capítulo, bem como as principais interações entre cada um dos módulos.

4.1 Visão Geral da Arquitetura

O Framework proposto está sendo baseado na arquitetura de um Mobile Host [1], como mostra a Figura 2.2, com a adição de mecanismos de segurança. O Framework foi concebido para ser utilizado em um ambiente Ad-hoc totalmente puro, entretanto determinadas atividades exigem a utilização de uma rede estacionária como a publicação dos serviços e a validação dos Certificados Digitais por uma Autoridade Certificadora. Todavia a comuicação entre o consumidor dos serviços e o provedor (disponibilizado pelo Framework) ocorre um ambiente Ad-Hoc.

A arquitetura permite ao desenvolvedor ao utilizar o framework proposto torne os seus serviços (inclusive mais de um serviço em um mesmo dispositivo) disponíveis ao público em geral sem a necessidade de implementações adicionais para as atividades necessárias para o provisionamento de serviço, como: receber e enviar mensagens, realizar leitura e conversão das mensagens de/para o

[1] Descrito na seção 2.2

formato SOAP/XML (o uso de mensagens XML influi no desempenho do dispositivo, pois as mensagens torna-se maiores, logo o dispositivo requer mais tempo para envio e recebimento da mensagem, e o seu tratamento é mais complexo, pois sua composição possui muitas informações que devem ser identificadas pelo dispositivo para definir qual seu contéudo e qual ação deve ser tomada); publicação de serviços; geração de documentos WSDL contendo as informações do serviço publicado; criação e monitoramento de uma interface de comunicação para transmissão de mensagens em diferentes tipos tecnologias existente (Bluetooth, Wireless, HTTP); identificar e executar de um serviço solicitado. Além disso, a arquitetura, também, provê um modelo de comunicação seguro fim-a-fim, sendo baseado em chaves públicas, conforme será visto nas próximas seções. A Figura 4.1 mostra os principais componentes da arquitetura proposta, os quais são descritos a seguir:

- Web Server - componente que provê as funcionalidades de um Servidor Web ao dispositivo móvel, por meio deste componente o framework recebe e envia mensagens de acordo com o tipo de tecnologia que está sendo empregada pelo desenvolvedor (bluetooth, HTTP ou wireless). Também, o Web Server é quem verifica se a mensagem recebida é uma mensagem SOAP ou uma requisição web normal, transferindo-a Gerente de Serviços caso seja uma requisição SOAP ou tratando-a caso seja uma requisição web normal;

- Controle de Acesso e Privacidade - este componente é responsável por verificar se um consumidor pode acessar os serviços disponíveis no dispositivo móvel, este controle é feito por intermédio do uso de senhas ou a partir de informações do próprio aparelho que pode receber permissão para acessar os serviços em tempo de execução e escolher se permite ou não seu acesso ser armazenado para verificações posteriores;

- Gerente de Serviços - componente responsável em dar as funcionalidades necessárias para a provisão de um serviço no dispositivo móvel. Este componente realiza atividades como a descrição do serviço após a obtenção das informações do mesmo e da geração de um documento WSDL relacionado a ele; armazenagem do serviço em uma base dados local, denominada Base de Dados de Serviços; geração de documento para sua publicação. O Gerente

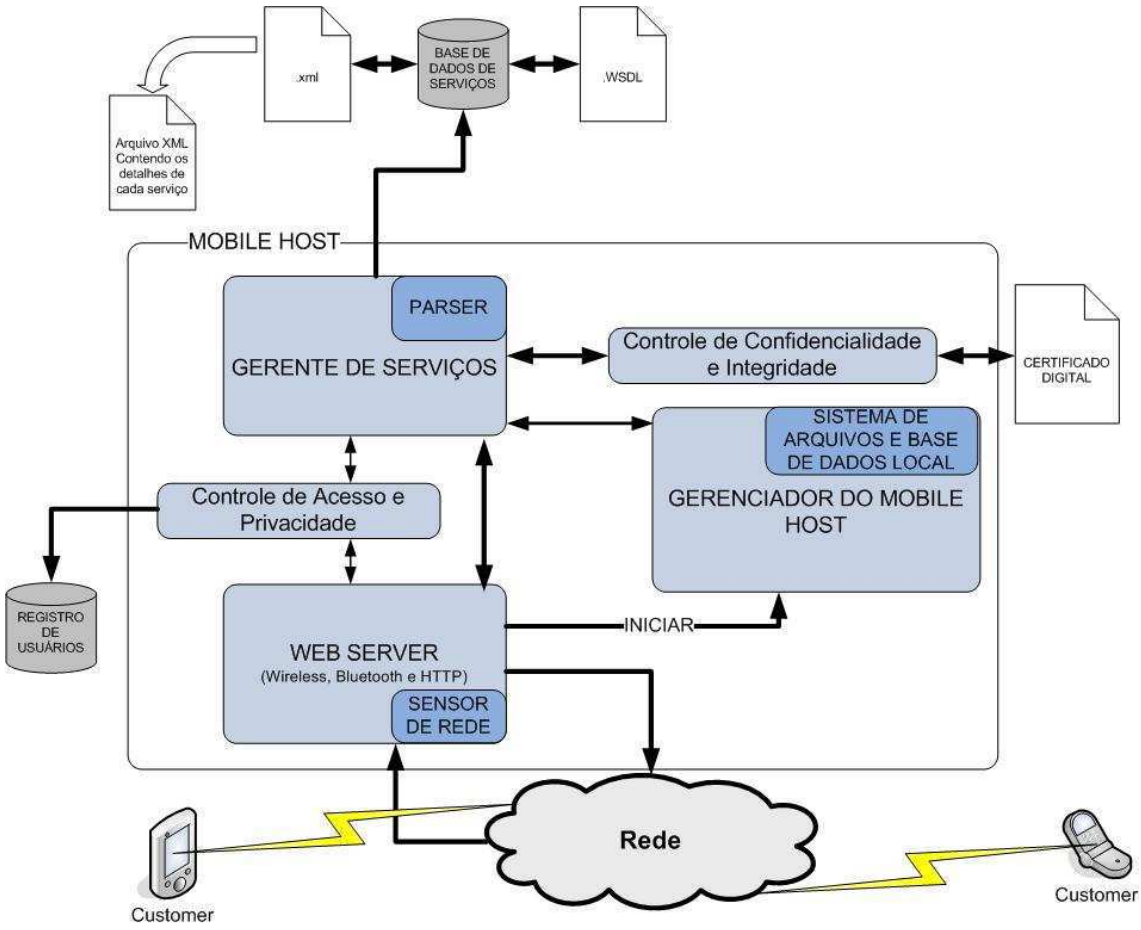

Figura 4.1: Arquitetura Prosposta

de Serviços também recebe as requisições SOAP e as processa, verificando possíveis erros, recuperando e executando algum serviço que foi solicitado;

- Gerenciador do Mobile Host - este módulo é responsável pela configuração e iniciação dos demais componentes do middleware. Neste módulo as bases de dados utilizadas são criadas e disponibilizadas para utilização dos demais componentes. Também, este componente é responsável por verificar se um serviço já foi criado e enviar os dados do novo serviço para serem processados pelo Gerente de Serviços;

- Controle de Confidencialidade e Integridade - adiciona mecanismos que permitem que as mensagens enviadas pelos consumidores não possam ser interpretados por terceiros por meio da geração de um par de chaves assimétricas

para cada serviço disponibilizado, cuja chave privada é armazenada no próprio dispositivo. Os consumidores podem obter a chave pública por meio de um certificado digital armazenada no dispositivo ou em algum receptáculo específico. Além disso, este módulo permite que o framework possa gerar uma assinatura digital para as mensagens que são criadas, permitindo que caso o contéudo transmitido seja alterado possa ser identificado.

- Base de Dados: O framework utiliza algumas bases de dados para armazenar algumas informações que são importantes para realização de suas atividades[2]. Estas bases, bem como as suas funcionalidades, são listadas a seguir:

  1. Base de Dados de Serviços - esta base é utilizada para armazenar os serviços criados e que estão disponíveis no dispositivo;
  2. Base de Registro de Usuários - armazena os dados para validação dos consumidores feita por intermédio do registro do login e do hash da senha. Os dados que permitem identificar um aparelho e se eles têm permissão para acessar os serviços também são armazenados nesta base;

4.1.1 Diagrama de Casos de Uso

A Figura 4.2 mostra o diagrama de caso de uso elaborado a partir das necessidades levantadas que devem ser suportados pelo sistema.

Os objetivos de cada caso de uso levantado são descritos a seguir:

1. **Criar Novo Serviço** - processo de criação de um novo serviço a ser oferecido;
2. **Gerar Descrição do serviço** - descrever o serviço e gerar um documento WSDL contendo os detalhes da descrição do serviço;
3. **Publicar Serviço** - gerar um modelo para registrá-lo em algum repositório;
4. **Armazenar Serviço** - processo de armazenagem do serviço localmente para posterior utilização;

[2]A criação das Bases de Dados do Framework foi feita por meio de da manipulação de arquivos no próprio dispositivo utizando-se o *RecordStore* do *J2ME*.

5. **Criar Certificado Digital** - geração de certificado digital para cada serviço criado;

6. **Gerar par de Chaves** - criar par de chaves Assimétricas que serão utilizadas para cifrar/decifrar as mensagens e geração de assinatura digital;

7. **Aguardar Solicitações** - aguardar o recebimento de mensagens de algum cliente;

8. **Criar Conexões de Rede** - criar uma conexão para aguardar solicitações na rede;

9. **Tratar Requisição SOAP** - verificar quando uma nova mensagem recebida for uma requisição SOAP e tratá-la devidamente;

10. **Efetuar Parser da Mensagem** - extraem da mensagem recebida as informações necessárias para sua execução;

11. **Obter Serviço** - verifica se o serviço invocado existe e se a composição da mensagem está correta;

12. **Executar Serviço** - executa o serviço solicitado;

13. **Decifrar Conteúdo da Mensagem** - caso o serviço solicitado tenha adicionado serviços de segurança é feita a decifragem do conteúdo da mensagem antes da sua execução;

14. **Responder Solicitação** - gera a mensagem de resposta da solicitação e/ou de erro caso algum tenha ocorrido;

15. **Gerar Assinatura Digital** - gera uma assinatura digital da resposta da solicitação;

4.1.2 Diagrama de Classes

A Figura 4.3 mostra o diagrama de classes do framework, com os seus respectivos relacionamentos. As funcionalidades das principais classes do diagrama apresentado serão detalhadas nesta sub-sessão.

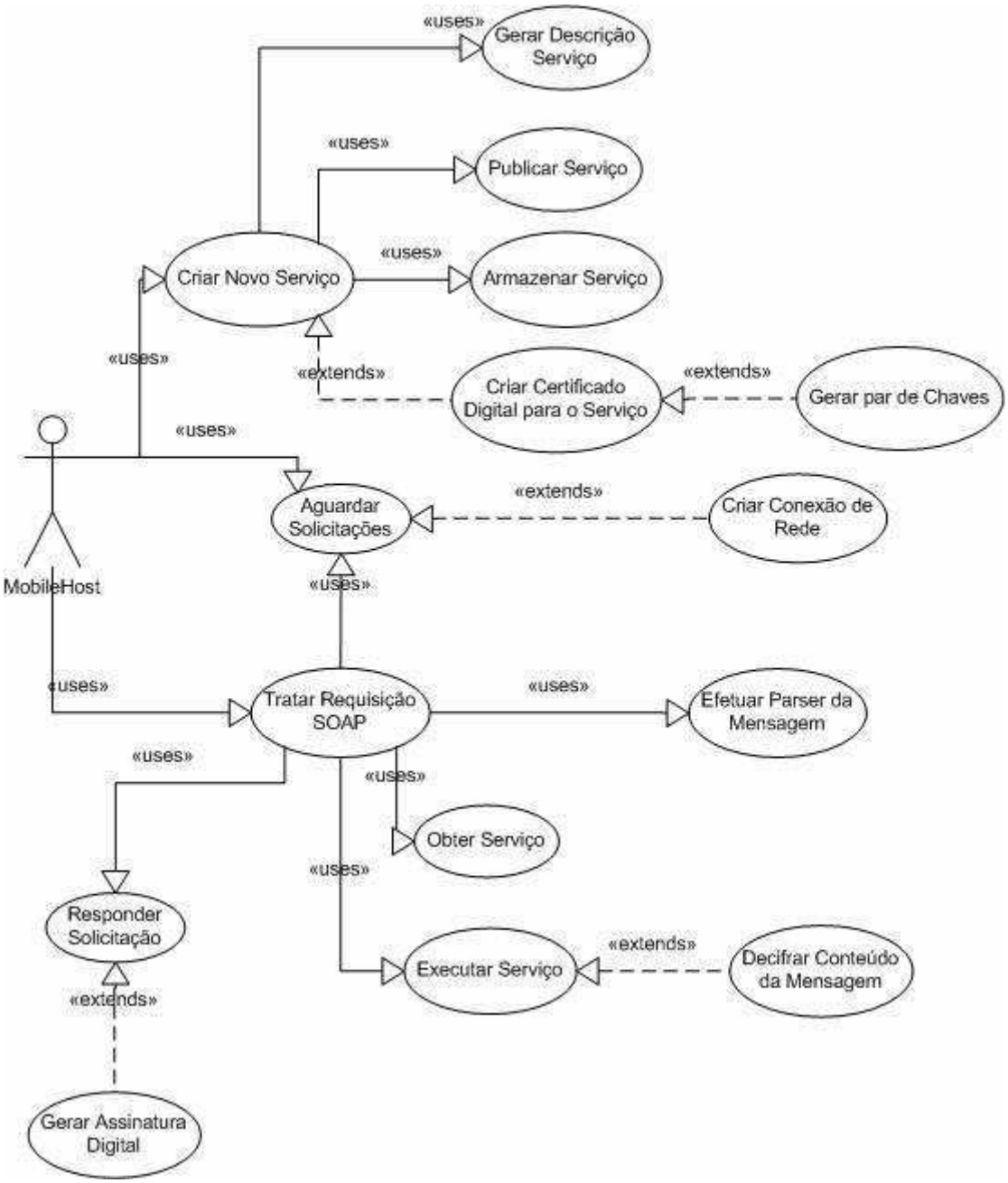

Figura 4.2: Diagrama de Casos de Uso

MobileHost é uma classe que herda a classe *MIDlet* para que a aplicação contendo o serviço proposto seja iniciada, por meio desta classe os demais componentes do Framework são iniciados e o desenvolver informa quais serão os serviços que estarão sendo disponibilizados naquele dispositivo. Além disso, nesta classe é definida qual tecnologia de comunicação será empregada pelo desenvolvedor.

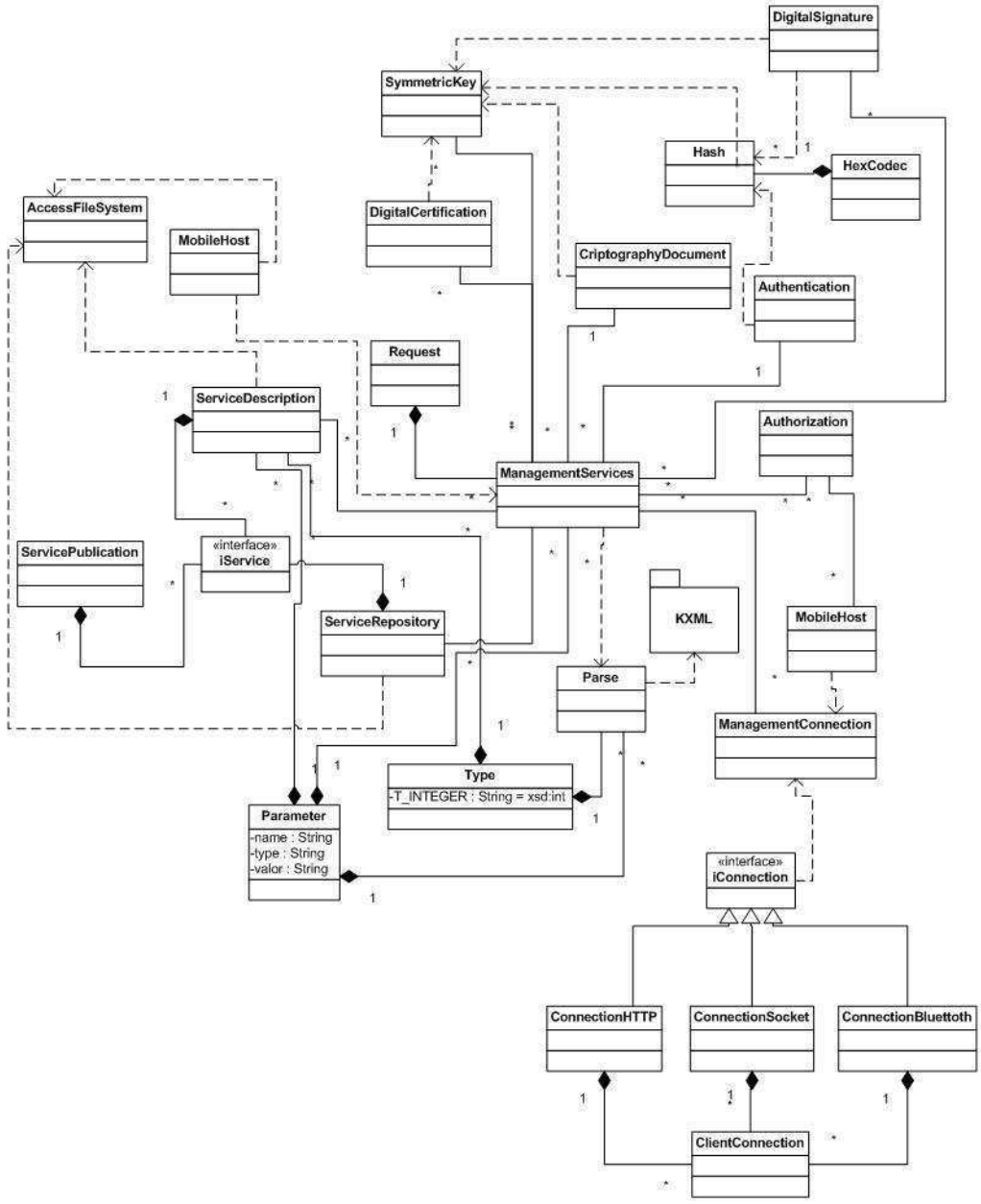

Figura 4.3: Diagrama de Classes do Framework

A interface *iService*, a qual pode-se ver na listagem a seguir, possui a assinatura dos métodos que ao serem implementados pelo desenvolvedor permitem que os dados do serviço sejam conhecidos, tais como nome dos métodos, tipos de parâmetros e retorno, a descrição dos seus métodos é feita a seguir:

- **describeService** permite a obter as informações referentes a um serviço e

a descrição de seus métodos;

- **executeMethod** permite a execução dos métodos do serviço;
- **checkRequest** faz a verificação se os dados de um método invocado estão de acordo com a sua descrição feita previamente;

Os demais métodos existentes são métodos Getters/Setters que permitem o acesso a um conjunto de atributos que são definidos para a utilização do Framework, os quais são listados a seguir:

- **getNameService()** e **setNameService()** permitem obter e setar o nome do serviço;
- **addMethod()** e **getAllMethod()** permitem adicionar os métodos a uma Hashtable e obtê-la para seu uso para um determinado fim;
- **getCert()** e **setCert()** permitem acesso ao certificado quando este for gerado pelo framework;
- **getServiceDescription()** e **setServiceDescription()** permitem o acesso a descrição de serviço que também é armenado em disco e que contém todas a informações obtidas por meio da interface **iService**;
- **setSecurityServiceActive()** e **isSecurityServiceActive()** permite informar se o serviço irá gerar o par de chaves assiméticas e se irá criar o certificado digital;
- **setServiceAuthentication()** e **isServiceAuthentication()** informa se o serviço irá realizar autenticação de consumidores;
- **isServiceAutorization()** e **setServiceAutorization()** informa se o serviço exigir a necessidade de autorização dos consumidores para executar uma determinada tarefa;
- **setCipher()** e **isCipher()** verifica se as mensagens enviadas devem está cifradas ou não; e
- **getSymmetric()** e **setSymmetric()** permitem acesso a classe que dá acesso ao par de chaves gerados e ao serviço de Criptografia das mensagens

```
public interface iService {

public abstract void describeService();
public abstract String executeMethod(String nameService,
Vector params);
public abstract boolean checkRequest (Request request);
public abstract String getNameService();
public abstract setNameService (String nameService);
public abstract boolean addMethod(String nameMethod, Vector params);
public abstract Hashtable getAllMethod();
public abstract CertificateDigital getCert ();
public abstract void setCert (CertificateDigital cer);
public abstract String getServiceDescription();
public abstract void setServiceDescription(String serviceDescription);
public abstract boolean  isSecurityServiceActive();
public abstract void setSecurityServiceActive(boolean status);
public abstract boolean isServiceAuthentication();
public abstract void setServiceAuthentication(boolean serviceAuthentication);
public abstract boolean isServiceAutorization();
public abstract void setServiceAutorization(boolean serviceAutorization);
public abstract boolean isCipher() ;
public abstract void setCipher(boolean cipher);
public abstract SymmetricKey getSymmetric();
public abstract void setSymmetric(SymmetricKey symmetric);
}
```

A classe *ServiceDescription* é responsável por gerar o documento WSDL referente ao serviço que está sendo criado de acordo com os dados informados pelo desenvolvedor, também é nesta classe que é criado o arquivo contendo todas as informações referentes ao serviço que será armazenado. A classe *ServiceRepository* armazena os dados dos serviço em tempo de execução, sendo responsável, também por recarregar um serviço da base de dados quando este for invocado e não estiver pronto para ser utilizado. Os dados para publicação são criados pela classe *ServicePublication*, além de envia e retirar esSas informações dos repositórios específicos

para publicação de serviços.

AccessFileSystem é uma classe que permite a criação, alteração, leitura e exclusão de arquivos, utilizada pelo framework para acessar as bases de dados utilizados por ele e dos arquivos manipulados por ele, como os documentos **WSDL** e de certificados digitais referentes aos serviços que são armazenados em disco. A classe *Parser* é responsável por extrair as informações de um serviço invocado do formato **SOAP** para a classe *Request* que mantém os dados de um requisição em tempo de execução.

A interface *iConnection* possui a assinatura dos métodos que permitem a criação de uma interface de rede para ser utilizada pelo Framework de acordo com a tecnologia empregada: a *ConnectionHTTP* implementa a interface *iConnection* para criar um serviço de rede que utilize a internet como meio de comunicação por intermédio do uso de SOAP sobre HTTP; a classe *ConnectionSocket* implementa a *interface* iConnection para permitir que os dispositivos se comuniquem diretamente por meio de sockets que envia os documentos SOAP entre eles e a classe *ConnectionBluetooth* implementa a interface *iConnection* permitindo que o protocolo Bluetooth seja utilizado para estabelecer a comunicação entre os dispositivos.

A classe *ManagementService* gerencia todas as atividades que envolvem os serviços que estão disponibilizados pelo dispositivo, como a criação, a invocação e execução. O gerenciamento da interface de rede é feito pela classe *ManagementConnection*, ou seja, gerência o recebimento e o envio de mensagens no dispositivo, para cada requisição que chega ela cria uma nova thread por meio da classe *ClientConnection*, que identifica que tipo de mensagem foi recebida e a trata da maneira adequada.

As funcionalidades do serviço de segurança são dadas por intermédio de um conjunto de classes, cujas principais são descritas a seguir: a classe *SymmetricKey* é responsável pela geração de um par de chaves assimétricas, onde para cada um dos serviços disponibilizados um novo par é gerado, e pela cifragem e decifragem de mensagens; a classe *DigitalCertificate* cria um Certificado Digital para cada serviço com base nas informações geradas pelo framework e pelo par de chaves criadas; *DigitalSignature* cria assinaturas digitais e verifica a veracidade de um documento assinado digitalmente; *Authentication* valida os dados de autenticação

feitos pelo consumidor de acordo com as informações armazenadas na **Base de Dados de Usuários** e *Authorization* verifica se determinado consumidor tem permissão para utilizar determinado serviço.

4.2 Gerenciador do Mobile Host

O componente **Gerenciador Mobile Host do Framework**, como o próprio nome sugere, realiza atividades de coordenação e gerenciamento do contexto geral do framework, interagindo com os demais componentes do middleware. O Gerenciador é responsável pelo gerenciamento, geração e inicialização das atividades executados pelos outros componentes do framework. A Figura 4.4 mostra a estrutura interna e os sub-componentes do módulo Gerenciador Mobile Host e a sua interação com os demais módulos. A descrição de seus sub-componentes é feita a seguir:

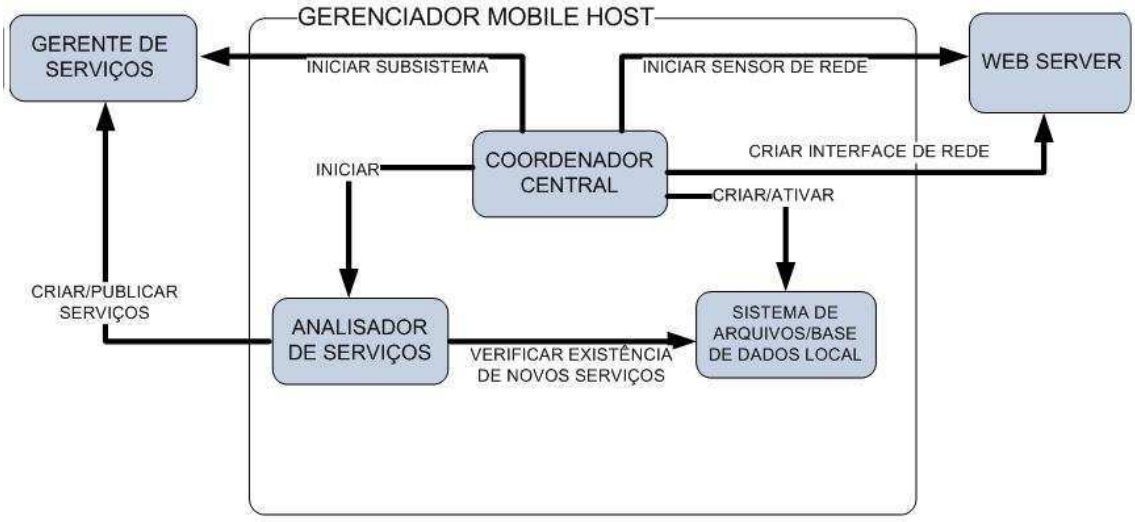

Figura 4.4: Gerenciador do Mobile Host

- Coordenador Central - responsável pela coordenação dos demais componentes do Framework. Por intermédio dele os serviços disponibilizados pelo framework são criados e/ou configurados e posteriormente iniciados. Neste sub-componente é definido o tipo de tecnologia de rede que será empregada pelo Framework e os detalhes para sua configuração;

- Analisador de Serviços - Este sub-componente verifica se um novo serviço será inserido no dispositivo, passando os dados do serviço, informados pelo desenvolvedor, para o Gerente de Serviços para este efetuar a sua criação e armazenamento;

- Sistema de Arquivos/Base de Dados Local - a criação e gerenciamento das bases de dados locais e do sistema de arquivo utilizado pelo framework é de responsabilidade deste sub-componente que possui uma interface que permite o acesso aos serviços que ele gerencia. Esta interface inclui métodos para criar e excluir arquivos e diretórios, atualizar e lêem arquivos, verificar se determinado arquivo existe, etc.;

4.2.1 Inicialização do *Mobile Host* e de seus serviços

A Figura 4.5 mostra um diagrama de sequência com processo de inicialização do Mobile Host e dos serviços que estão associados a ele. O Mobile Host inicia o sistema de arquivos e cria os diretórios onde serão armazenados as descrições dos serviços (passo 1) onde o arquivo XML e o documento WSDL referente a cada serviço são criados e os certificados digitais criados quando os serviços de segurança são adicionados (passo 2 e 3) respectivamente, a criação destes diretórios ocorre somente uma vez quando o Framework é executado pela primeira vez. Outra tarefa desempenha é a criação e inicialização do **Gerenciador de Serviços** (passo 4), componente que gerência todas as atividades referentes aos serviços criados, conforme visto na seção 4.4, também um repositório de serviços utilizado em tempo de execução é criado 9 (passo 5). O outro componente a ser criado é o gerenciador de conexões (passo 6), responsável pelo controle do tráfego da rede no Mobile Host, nesta etapa o tipo de tecnologia utilizada para comunicação é definido (passo 7) e o serviço correspondente é instanciado (passo 8), além disso o gerenciador de serviço para o qual as requisições devem ser repassadas é definido (passo 9) e por fim o serviço é iniciado (passo 10).

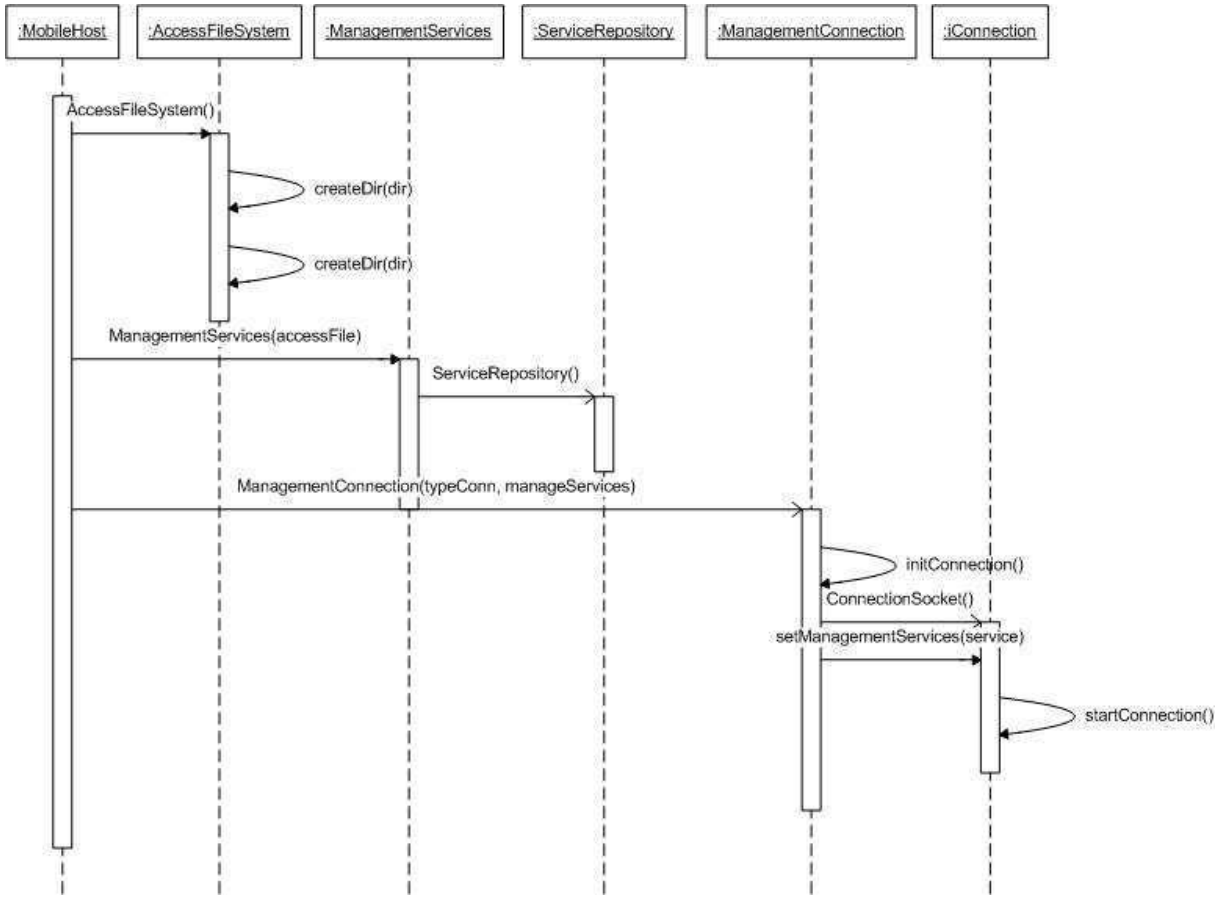

Figura 4.5: Diagrama de Sequência - Inicialização do Mobile Host

4.3 Web Server

Neste tópico são apresentados os sub-componentes do *Web Server*, conforme veremos a seguir, além da forma como eles interagem entre si e com os demais módulos do framework pode ser observado na Figura 4.6:

- Gerador de Interface de Rede - Para que exista comunicação entre o provedor de serviços e o consumidor final, o framework possui um mecanismo que provê as funcionalidades necessárias para a sua comunicação. Este subcomponente é responsável pela criação deste mecanismo, que permite que diferentes tecnologias de comunicação possam ser empregadas (como Bluetooth, HTTP ou Sockets), cabendo ao desenvolvedor escolher a qual melhor que se adapta as suas necessidades. O *Gerenciador do Mobile Host* é responsável por identificar qual tecnologia está sendo utilizada e por fornecer

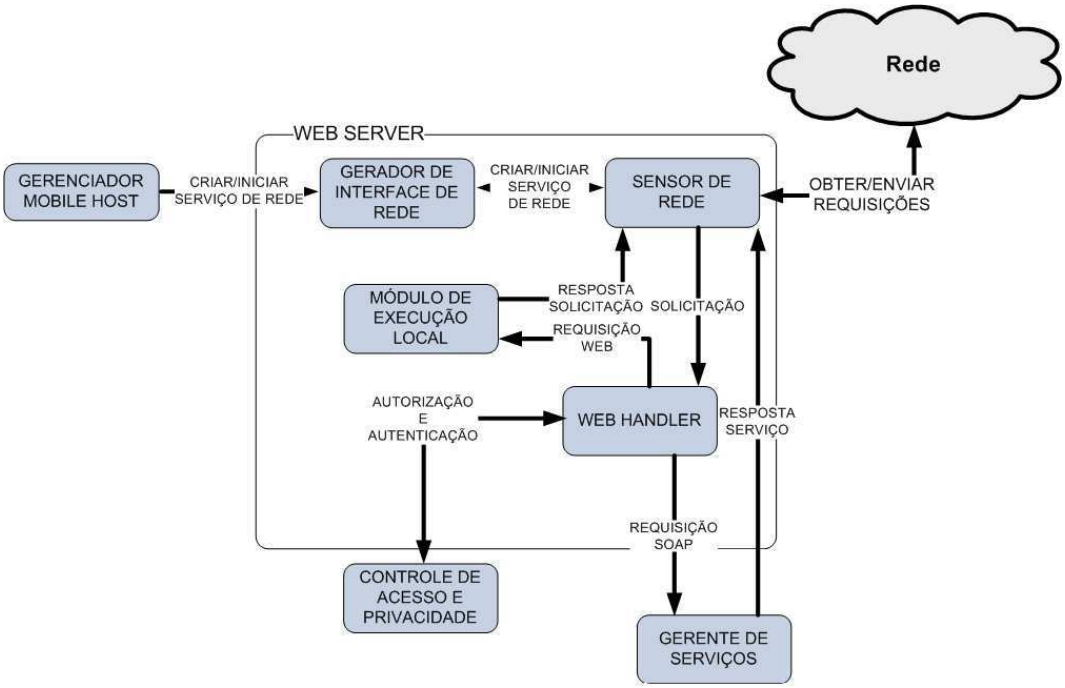

Figura 4.6: Componentes do Web Server Utilizado pela Arquitetura

os dados para a sua configuração. O Framework implementa diferentes tecnologias, entretanto não trabalham de forma coordenada, ou seja, somente uma pode ser utilizada quando o Framework está executando. Entretanto, o Framework ainda não possui meios para identificar quando um dispositivo está fora do seu raio de atuação, ou seja, quando ele está impossibilitado de se comunicar com o consumidor, o que causaria a perda da mensagem enviada nestes casos;

- Sensor de Rede - Este sub-componente é responsável por receber e enviar as mensagens no do Mobile Host de acordo com a tecnologia empregada, ou seja, apenas uma tecnologia pode ser usada para comunição; o Framework não realiza gerenciamento interno para verificar qual tecnologia pode ser usada em dado instante, mesmo se estiver ociosa. O Mobile Host instância este sub-componente que fica constantemente esperando por solicitações de consumidores chegarem ao dispositivo e enviar as mensagens geradas pelo framework, desta forma o funcionamento do Framework se assemelha a um Servidor Web normal;

- Web Handler - Quando uma nova requisição é recebida este sub-componente é acionado e uma nova linha de execução é criada para atender essa solicitação. Este componente é responsável por verificar se o pedido recebido é uma requisição SOAP ou uma requisição web normal e posteriormente enviá-lo para o tratamento adequado. No caso de ser uma requisição SOAP ela é enviada ao *Gerente de Serviços*, caso contrário é tratada pelo próprio *Web Server* por meio do *Módulo de Execução Local*. Controle de acesso feito pelo framework é realizado neste sub-componete que utiliza as interfaces criadas no módulo de *Controle de Acesso e Privacidade* para validar as permissões de acesso que o requisitante possui;

- Módulo de Execução Local - as solicitações web normais são executadas por este sub-componente que também gera uma mensagem de resposta e a envia para o requisitante por intermédio do *Sensor de Rede*;

4.3.1 Processamento de Mensagem Recebida

A Figura 4.7 mostra o processo de tratamento de uma mensagem recebida pelo framework, o qual ocorre da seguinte forma: o Sensor de Rede recebe uma mensagem em algum canal de comunicação (passo 1), isso ativa o Web Handler que cria uma nova linha de execução para tratar a mensagem recebida (passo 2), em seguida é verificado a natureza da mensagem: caso seja uma mensagem SOAP ou uma requisição web normal (passo 3), sendo uma requisição normal é tratada localmente, senão é enviada para o Gerenciador de Serviços (passo 4). O Gerenciador de Serviços extrai o conteúdo da mensagem (passo 5) e obtém o serviço requisitado no repositório de serviços (passo 6), este componente verifica se o serviço já está carregado na memória (passo 7), se não estiver o serviço é recarregado da Base de Dados de Serviços (passo 8), caso o serviço não seja encontrado uma mensagem de erro é enviada para o requisitante informando que o serviço não existe.

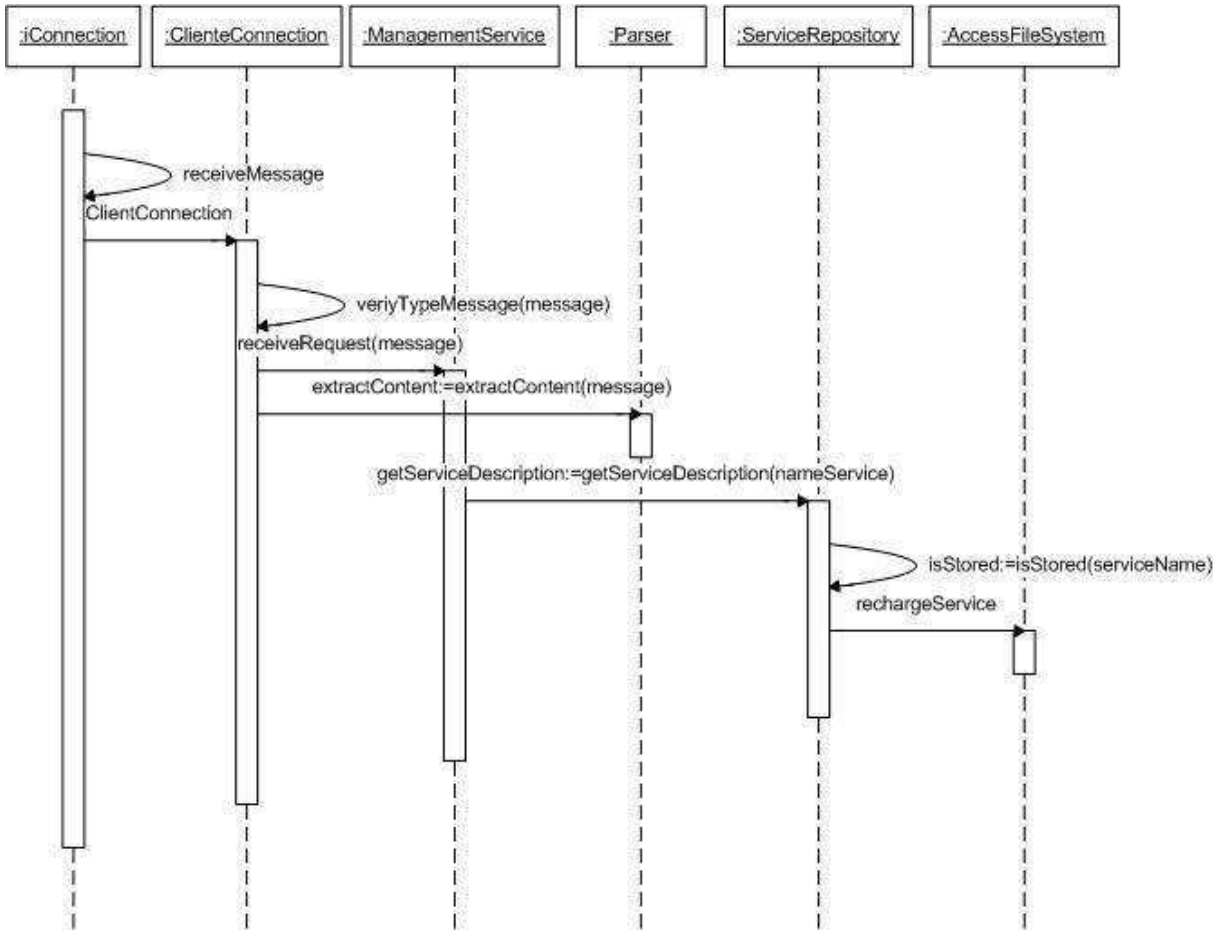

Figura 4.7: Diagrama de Sequência - Processamento de Requisição Recebida

4.4 Gerente de Serviços

O Gerente de Serviços é responsável pelas atividades de geração, descrição, publicação, busca e execução de serviços. A Figura 4.8 apresenta o Gerente de Serviços e seus sub-componentes e sua interação com os demais componentes do sistema. A descrição dos seus sub-componentes é feita a seguir:

- Gerador de Descrição - O *Gerenciador do Mobile Host* envia os dados do serviço a ser criado para este sub-componente que gera a descrição do serviço para ser armazenado como um documento WSDL ao ser enviando para a *Base de Dados de Serviços*, ou seja, é armazenado no próprio dispositivo, entretanto o documento WSDL pode ser enviado para um *Binder* para que ele se torne público. Também, esse módulo é responsável por criar as chaves as-

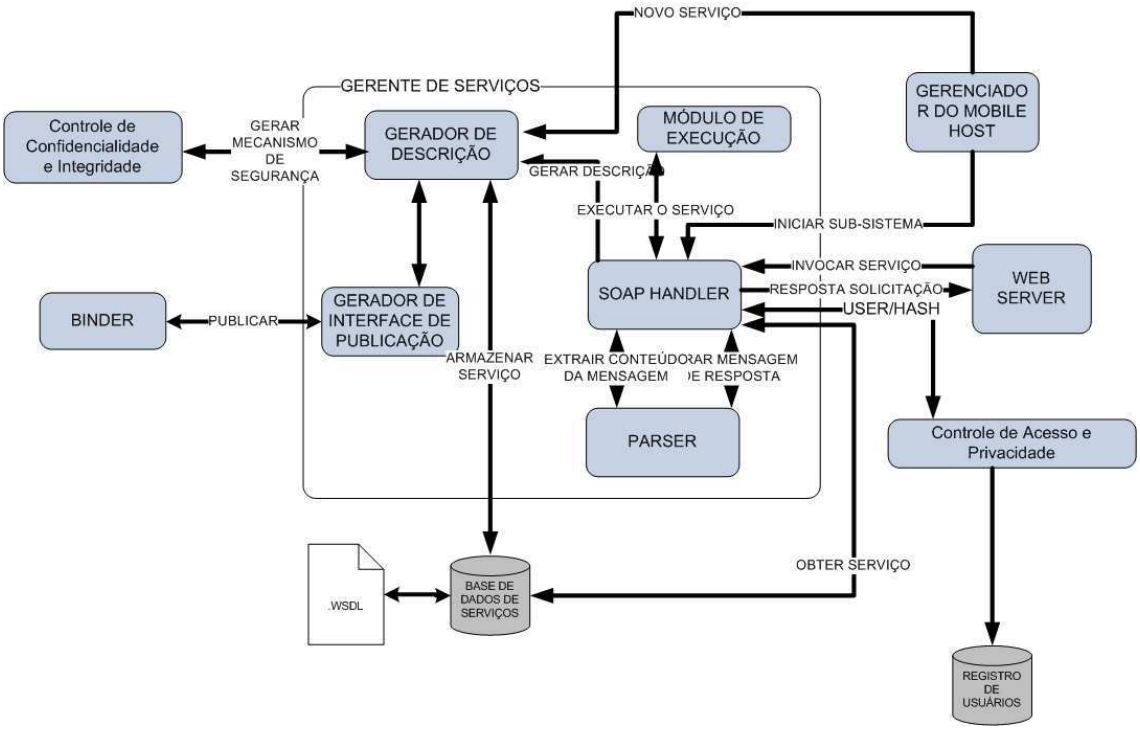

Figura 4.8: Gerente de Serviços

simétricas e de um certificado digital para o serviço, caso seja solicitado pelo desenvolvedor, utilizando a interface disponibilizada pelo módulo de *Controle de Confidencialidade e Integridade*. Paralelamente com a descrição este sub-componente cria um documento XML com as informações do serviço que está sendo armazenado, como nome do serviço, métodos e tipos de retorno, chaves assimétricas do serviço, referência do certificado digital criado;

- Gerador de Interface de Publicação - sub-componente responsável pelo anúncio das descrições dos serviços. Um documento UDDI é gerado neste componente para ser enviado a um repositório de serviços (Binder) utilizado os dados obtidos pelo *Gerador de Descrição*. O módulo verifica a validade da descrição a ser publicada, e, em caso afirmativo, o envia para publicação, também é responsável por solicitar a retirada de um serviço anteriormente publicado, quando algum serviço não for mais disponibilizado pelo dispositivo;

- SOAP Handler - quando uma requisição SOAP chega pela rede e é transmitida para o *Gerente de Serviços* é o **SOAP Handler** que irá tratá-la. Este sub-componente inicialmente aciona o *Parser* para extrair o conteúdo da mensagem e verificar se contém algum erro. Posteriormente obtém o serviço soliticitado na *Base de Dados de Serviços*, caso não exista uma mensagem de erro é gerada e enviada ao solicitante. Com as duas etapas anteriores concluídas o **SOAP Handler** aciona o *Módulo de Execução* para executar o serviço, caso execute normalmente o **SOAP Handler** gerará uma mensagem de resposta por meio do *Parser* e a enviará para o solicitante, caso contrário uma mensagem de erro será criada e enviada. Este módulo, também é responsável por gerar a assinatura digital da mensagem utilizando a chave privada do serviço para ser enviada com a resposta da solicitação para o solicitante, garantindo a integridade da mensagem;

- Módulo de Execução - O serviço solicitado é executado por este sub-componente que aciona as ações pertinentes a execução do serviço, como acessar sistema de arquivos ou bases de dados locais e realizar requisições na rede quando o serviço o necessita, caso um serviço não esteja disponível na memória principal ele é carregado da **Base de Dados de Serviços**. As atividades são disparadas em virtude da especificidade de cada serviço e de acordo com as características das requisições e dos perfis dos consumidores envolvidos. Caso o sistema esteja trabalhando com os mecanismos de criptografia a decifragem do conteúdo da mensagem é feita neste sub-componente;

- Parser - Uma importante funcionalidade do Mobile Host é sua capacidade de realizar o parser das mensagens recebidas na forma de um envelope SOAP para um objeto que contém as informações da requisição e que é utilizado para obter as informações necessárias para execução da requisição. Também, o processo inverso é feito com os dados da resposta concluídos estes são transformados em um documento SOAP antes de ser enviado para o requisitante. O framework também gera mensagens de erro (SOAP FAULT) caso alguma falha seja identificada na composição da requisição ou durante a sua execução.

4.4.1 Geração de Mensagem de Resposta

A Figura 4.9 mostra o processo de geração e envio da mensagem de resposta ao solicitante de um serviço. Inicialmente, uma mensagem SOAP é gerada (passo 1) de acordo com as informações obtidas durante os processos anteriores (execução da requisição [3], geração de Assinatura Digital [4]), além das informações do serviço requisitado. Em seguida, a mensagem é enviada para o Web Server (passo 2), que gera uma mensagem de acordo com a tecnologia utilizada (passo 3) encaminha a resposta para consumidor que solicitou o serviço (passo 4) e por fim fecha a conexão com o dispositivo consumidor (passo 5).

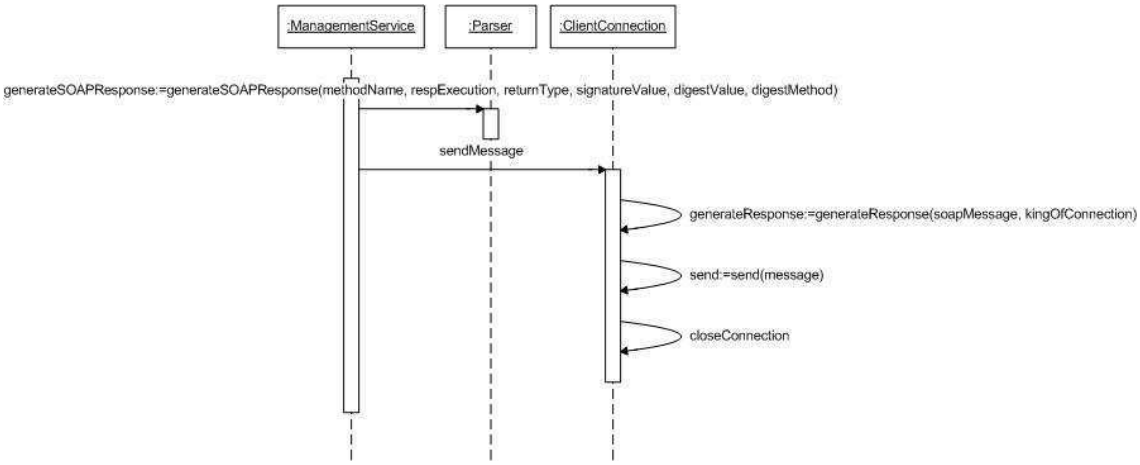

Figura 4.9: Diagrama de Sequência - Geração de Mensagem de Resposta

4.4.2 Execução de Serviço

O processo de execução de um serviço solicitado pode se visto na Figura 4.10 que consiste, inicialmente, em verificar as informações da requisição estão corretas de acordo com o serviço que está sendo invocado(passo 1), ou seja, se os dados do corpo da mensagem SOAP correspondem à composição do serviço invocado conforme foi informado durante o processo de criação do serviço. Uma vez essas informação estejam corretas o serviço é executado (passo 2), caso contrário o sistema gera uma mensagem de erro.

[3] ver seção 4.4.2

[4] ver seção 4.6.2

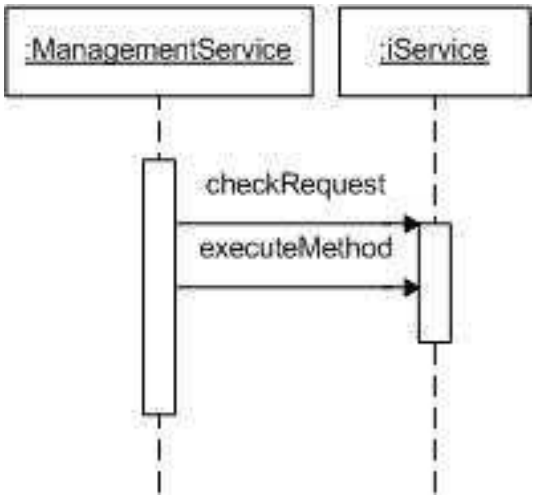

Figura 4.10: Diagrama de Sequência - Execução de Serviço

4.5 Controle de Acesso e Privacidade

Em determinados casos existe a necessidade de que sejam empregados mecanismos que permitam estabelecer controle de acesso aos serviços disponibilizados pelo Framework. Logo, um sistema de controle de acesso foi criado e esse proveém uma forma de controlar quais consumidores de serviços terão acesso aos serviços. Este serviço foi projetado para realizar a autenticação dos usuários cadastrados no sistema por meio de login e senha e também, de acordo com as informações sobre o dispositivo que está requisitando um serviço. Logo, antes de requisitar um serviço, o consumidor precisa se autenticar no sistema. A **Controle de Acesso e Privacidade** assegura que o cliente se autenticou em um determinado instante usando o método de autenticação aqui especificado. Durante o processo de autenticação, o consumidor recupera atributos que o credencia e que estão armazenados no sistema (i.e. *login* e *hash da senha*) e que são usados para determinar se o mesmo possui direito de acesso ao recurso.

O módulo de Controle de Acesso e Privacidade atua como um agente verificador sobre os acessos feitos pelos consumidores aos serviços, ou seja, ele é responsável por verificar se um determinado consumidor e/ou dispositivo tem direito de acessar os serviços no dispositivo. O controle é feito para cada mensagem que chega no dispositivo, assim sempre que um consumidor deseja utilizar um serviço ele deve se autenticar no sistema. Para tanto, ele conta com uma base de

dados própria onde são armazenados os dados para a validação do consumidor, onde são armazenados o login e o hash da senha de cada consumidor (usuário) que têm permissão para se autenticar no sistema. Outro controle é feito quando uma requisição é feita ao sistema pedindo ao usuário do dispositivo a confirmação do uso do recurso que está sendo solicitado caso o usuário negue a requisição não é executada, caso contrário ela é executada. Também, uma verificação é feita para saber se o usuário deseja armazenar o tipo de opção escolhida para determinado consumidor, e se for novamente solicitado não irá pedir a confirmação novamente ao usuário e sim utilizar o histórico de permissões.

Senhas são utilizadas para autenticação de usuários, mas por questões de segurança, apenas os hash dessas senhas são armazenados em uma base de dados, o que evita que o sistema seja comprometido em caso de violação da base de dados. O processo de geração de hash é descrito na seção 4.5.1. O Serviço de Autenticação responde a cada uma das mensagens recebidas pelo framework e, assim, esse pode saber quais podem continuar acessando o sistema. Quando uma requisição chega ao sistema, é feita uma busca do hash na base de dados e este é comparado com o hash da senha informada.

O Serviço de Autenticação envia uma resposta que pode ser uma confirmação de que o dispositivo está autenticado até a resposta da solicitação, no caso do hash recebido e o armazenado serem idênticos, no caso de não serem idênticos é negado o acesso aos serviços. O funcionamento deste esquema pode ser visto na Figura 4.11, onde após obter os valores do nome e do hash da senha o sistema verifica a veracidade dessas informações e, consequentemente, permite ou nega o acesso ao serviços solicitado.

As mensagens SOAP devem ser criadas já contendo os dados para a sua autenticação, ou seja, para transmitir os dados usados na validação da aplicação que consome os recursos o consumidor deve criar uma mensagem SOAP de acordo com modelo ***WS-Security*** para autenticação, conforme podemos observar no exemplo a seguir:

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Header>
      <wsse:Security
       xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext">
```

```
        <wsse:UsernameToken>
           <wsse:Username></wsse:Username>
           <wsse:Password Type="wsse:PasswordDigest">
           </wsse:Password>
        </wsse:UsernameToken>
     </wsse:Security>
  </Header>

  <Body>
     ...
  </Body>
</Envelope>
```

Pode-se observar que o elemento ***UsernameToken*** é usado para construção de um nó que contém as informações necessárias para autenticação e autorização, os sub-elementos ***Username*** e ***Password*** são usados para transmitir o nome do usuário e a senha respectivamente, onde um hash da senha do consumidor é enviado O framework é capaz de processar este tipo de documento SOAP. A mensagem SOAP que chega pela rede é enviada para o **SOAP Handler**, cujo conteúdo é extraindo pelo **PARSER** que gera um objeto que contém os dados da mensagem inclusive os dados para a autenticação, nesta etapa o serviço que está sendo solicitado também é identificado pelo **PARSER**. Se a mensagem contiver algum erro na composição da mensagem o **PARSER** o identificará e uma mensagem de erro será criada e enviada ao consumidor. O **SOAP Handler** obtém os dados do serviço e verifica se ele exige o uso de serviços de autenticação, se exigir os valores de login e senha (hash) são enviados para o módulo ***Controle de Acesso e Privacidade*** que é responsável por validar o acesso ao obter do sistema um resultado positivo ou negativo de acordo com sodados enviados. Este componente por intermédio *Serviço de Autenticação* realiza uma consulta na **Base de Registro de Usuários** para verificar se o usuário em questão possui permissão para acessar o serviço e se sua senha (hash) está correta, passando esta informação ao **SOAP Handler** que executa ou não a solicitação recebida, de acordo com a resposta do Framework.

O processo de autorização pode ocorre sem o uso da validação por senhas.

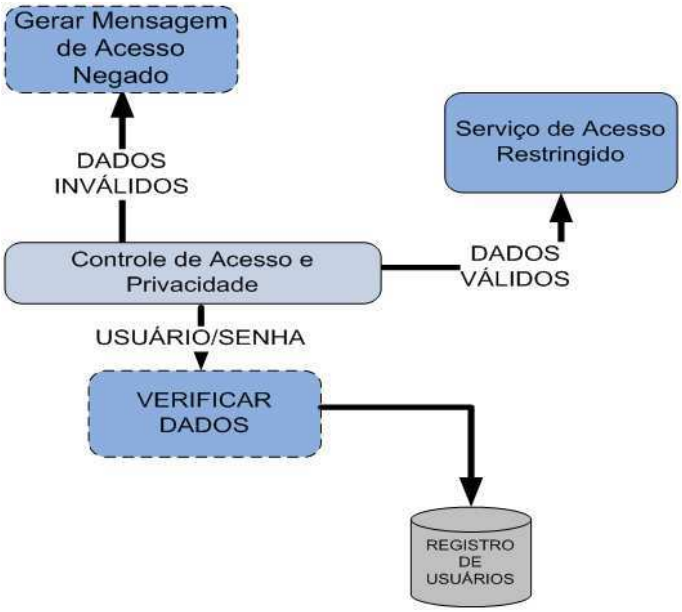

Figura 4.11: Processo de Validação de Mensagem Recebida

Neste caso, ao receber uma mensagem o **SOAP Handler** também a envia para o **PARSER** para obter seus dados. O **SOAP Handler** verifica o serviço solicitado exige a autorização de sua execução, em caso positivo uma mensagem é exibida ao usuário do dispositivo móvel, onde o **Mobile Host** está instalado. Essa mensagem pede a confirmação da execução do serviço e contém informações sobre o aparelho que está efetuando a requisição, que é executado ou não dependendo da opção escolhida pelo usuário. Além disso, outra mensagem é exibida para o usuário, onde é solicitado se ele deseja registrar a opção escolhida para o dispositivo solicitante e esta informação é armazenada na **Base de Registro de Usuários** para posterior verificação sem a necessidade de interação com o usuário.

O Framework foi desenvolvido para restringir o acesso para o uso dos serviços instalados no dispositivo. Entretanto, a criação de outros mecanismos é possível, como, por exemplo, criar níveis de acesso aos serviços, de acordo com o consumidor isoladamente ou com um grupo de consumidores que acesso determinado serviço.

4.5.1 Geração de *Hash*

Os usuários que realizam autenticação são registrados previamente, para tanto cada um deles deve possuir um nome de usuário e sua respectiva senha, a Tabela

Servico	**User**	**Hash**
Restaurante	USER1	93246038d91f02b45aefd4b883edff31b67a00ce
Restaurante	USER2	942d06800727d9f025a8ab493adbc16f42a40fd3
Restaurante	USER2	61130e4f25bbcfde7b22784ef60954aefc691e85

Tabela 4.1: Exemplo da Tabela de Usuários

4.1 mostra um exemplo da Base de Dados de Usuários, esta base é carregada pela aplicação que usar o serviço e o Framework possui ferramentas para prover esta funcionalidade. O campo **Servico** é nome do serviço o qual determinado usuário tem permissão de acesso, já o campo **user** é o login do usuário que tem a permissão e o **Hash** a armazena a senha do usuário a qual passa por um processo para que em vez de armazenar a senha diretamente, somente o valor de seu hash por ser único é armazenado. O código que gera o valor de hash pode ser visto a seguir:

```
public static bytes[] createHash (bytes[] password) {

    Digest digest = new SHA1Digest();
    digest.update(password, 0, password.length);
    byte[] digestValue = new byte[digest.getDigestSize()];
    digest.doFinal(digestValue, 0);

    return HexCodec.bytesToHex(digestValue);
}
```

4.5.2 Autenticar Requisição

A Figura 4.12 mostra o processo autenticação e autorização de uma solicitação feita quando o serviço solicitado requer este tipo de tratamento, que ocorre da seguinte forma: O Gerenciador de Serviços verifica se o serviço solicitado necessita da autenticação do consumidor do recurso para ser executado (passo 1). Se o serviço exigir autenticação o Gerenciador irá validar a o login do consumidor (passo 2), cujo login e senha já devem constar na mensagem enviada e essas informações são extraídas durante o processo de tratamento de mensagem recebida,

verificando se os dados do solicitante correspondem aos dados existente na Base da Dados de Usuários (passo 3), se os dados não forem correspondentes ou os dados para a autenticação não forem informados uma mensagem de erro é gerada e enviada para o solicitante. Quando as informações estiverem corretas o tratamento da mensagem prossegue normalmente.

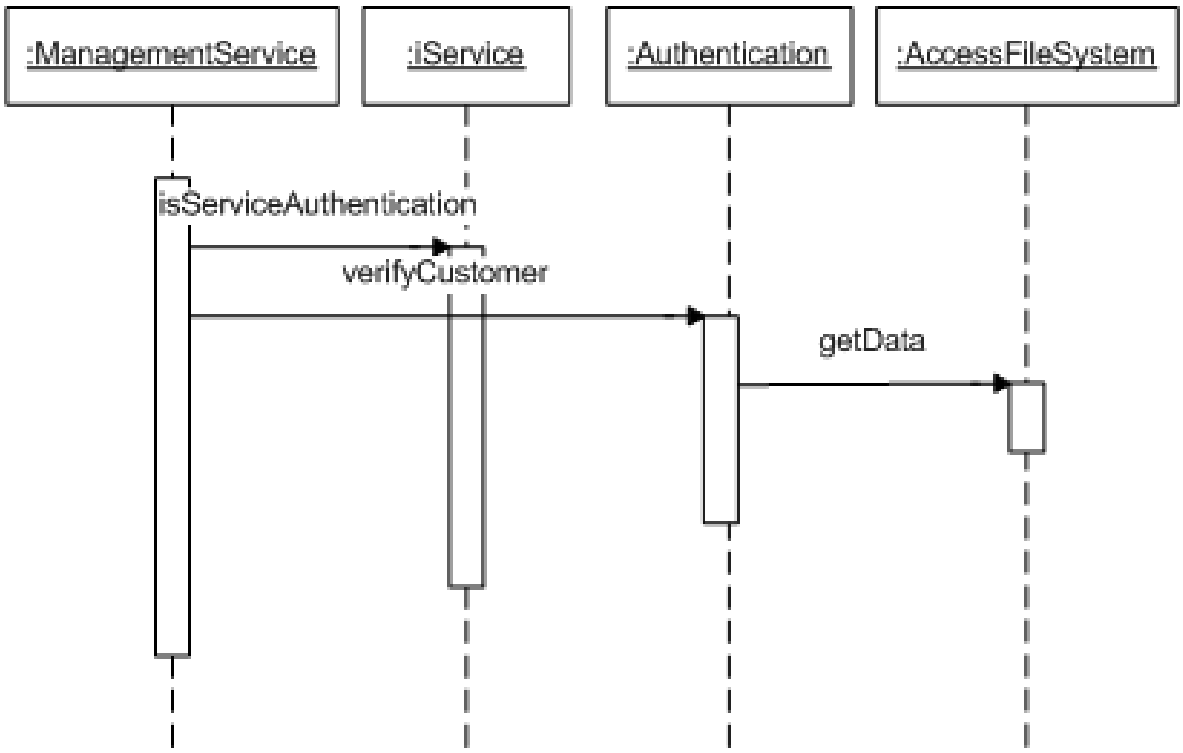

Figura 4.12: Diagrama de Sequência - Autenticação de uma requisição

4.5.3 Autorizar Requisição

A próxima fase consiste em verificar se o serviço necessita da autorização do usuário do dispositivo móvel, conforme se observa na Figura 4.13, para ser executado (passo 1), ou seja, quando uma requisição chega e tenta executar algum serviço cabe ao usuário do dispositivo autorizar ou não o acesso aos seus serviços. Caso necessite o processo de autorização será iniciado (passo 2) onde o Gerenciador verifica se o solicitante possui permissão de acesso (passo 3), informação que é armazenada em uma base de dados local, senão a ação a ser seguida consiste em criar uma mensagem para ser exibida ao usuário do dispositivo pedindo que a execução do serviço seja autorizada (passo 4) e também se ele deseja se a opção escolhida nesse caso seja armazenada para futura verificação (passo 5 e 6).

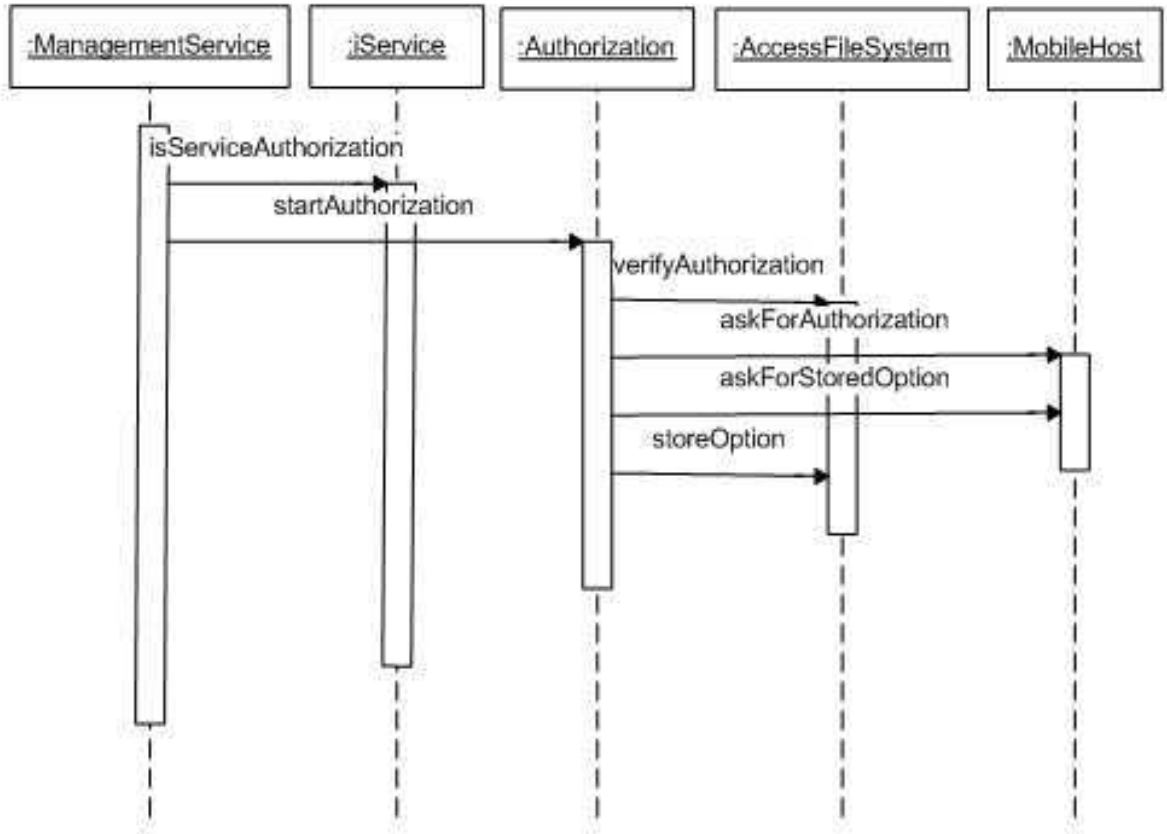

Figura 4.13: Diagrama de Sequência - Autenticação de uma requisição

4.6 Controle de Confidencialidade e Integridade

Para assegurar que um conteúdo não seja modificado durante a sua transmissão pela rede e que sua origem seja legítima este módulo foi concebido, ou seja, em sua proposta o framework possui um mecanismo de segurança baseado no conteúdo das mensagens se dedicando em proteger a comunicação fim-a-fim entre o consumidor e o provedor de serviço, garantindo que as mensagens trocadas entre ambos e não sejam conhecidas ou alteradas por terceiros. Ele contém ferramentas que permitem tanto a transmissão segura de mensagens quanto a integridade dos dados. Esta garantia se dá pelo uso de um par de chaves assiméticas que são geradas pelo próprio módulo e que podem ser usadas para criptografar de mensagens pelos clientes. Também, esse módulo é capaz de criar assinaturas digitais que são anexadas às mensagens e caso o seu conteúdo seja alterado a sua assinatura não irá corresponder ao seu conteúdo o que invalidará a mensagem. A garantia ocorre, porque o módulo de integridade (assinatura digital) aplica o algoritmo de criptografia sobre o hash do conteúdo. Dessa forma, se o conteúdo for alterado, seu hash não será o mesmo afetando o valor da assinatura gerada. Além dos funções descritas acima este mecanismo provê ferramentas para geração de certificados digitais.

Com esse intuito para cada serviço disponibilizado pelo **Mobile Host** para cada serviço que é disponibilizado pode ser gerado um par de chaves (pública e privada) durante o processo de criação do serviço[5] (embora o uso dos recursos de segurança não seja obrigatório cabendo ao desenvolvedor final decidir se agrega ou não este serviço ao seu sistema). O **Gerador de Descrição** é que cria as chaves que serão utilizadas pelo serviço, usando as interfaces disponibilizadas pelo **Controle de Confidencialidade e Integridade** e essas são armazenadas junto do seu serviço correspondente na **Base de Dados de Serviços**.

As chaves podem ser usadas no processo de assinatura digital, permitindo a garantia da autenticidade das mensagens que são enviadas, pois o mecanismo de integridade está associado ao seu conteúdo. A construção da mensagem SOAP para resposta de uma requisição com uso de assinatura segue o modelo definido pelo ***XML-Signature***, conforme podemos observar na listagem a seguir:

[5]Descrito na seção 4.7

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<Header>
<wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext">
<Signature Id="RailCo333" xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
          <SignatureMethod
           Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
          <Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
           <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
           <DigestValue>...</DigestValue>
          </Reference>
       </SignedInfo>
       <SignatureValue>...</SignatureValue>
       <KeyInfo>...</KeyInfo>
     </Signature>
   </wsse:Security>
  </Header>

<Body>
 ...
</Body>
</Envelope>
```

Para que a chave pública possa ser disponibilizada para um possível consumidor, o sistema agrega a geração do certificado digital que pode ser enviado para os consumidores móveis. O Certificado é armazenado em um repositório específico criado pelo Framework no próprio dispositivo onde ele está executando. Com o certificado um consumidor pode obter a chave pública dos serviços o qual ele deseja se comunicar, podendo cifrar as suas mensagens e verificar a autenticidade das assinaturas digitais recebidas por eles. O certificado pode ser enviado para uma Autoridade Certificadora para efetuar a sua validação. Um exemplo de um Certificado Digital gerado pelo sistema pode ser visto no Apêndice C.

Além de possibilitar o uso de Assinaturas Digitais, o Framework foi concebido para trabalhar com o envio e recebimento de mensagens seguras, por meio do uso do padrão XML-Encryption. Este padrão possui uma estrutura que permite que mensagens cifradas sejam adicionadas a um documento XML, conforme podemos observar no trecho de código a seguir, cujo valor cifrado está localizado no campo ***CipherValue***. O Framework proposto é capaz de gerar e realizar a leitura de mensagens que contenha este tipo de estrutura.

```
<EncryptedData
xmlns="http://www.w3.org/2001/04/xmlenc#"
  Type="http://www.w3.org/2001/04/xmlenc#Element">
<CipherData>
<CipherValue>R5J7UUI78</CipherValue>
    </CipherData>
</EncryptedData>
```

A Figura 4.14 mostra o processo de uso do Serviço de Segurança pelo Framework. Para tanto, o certificado digital gerado estará disponível para o público em geral que o obtém e tem acesso a chave pública do serviço o qual ele deseja utilizar. O Certificado Digital pode ser obtido diretamente no dispositivo onde o serviço está disponobilizado, por meio de uma simples requisição HTTP que enviado junto do documento WSDL, mas uma Autoridade Certificadora deve validá-lo. Logo, o consumidor pode usar a chave pública para cifrar mensagens que ele deseja enviar e também verificar a origem da mensagem recebida, pois, como as mensagens deverão está assinadas digitalmente ele pode fazer a sua validação com a chave que ele contém. Somente o Framework, consequentemente, pode decifrar as mensagens destinadas a determinado serviço com a sua chave privada correspondente além de assinar digitalmente as mensagens que serão enviadas para o consumidor.

4.6.1 Chaves Assimétricas

O processo de geração das chaves, que pode ser vista na Figura 4.11, consiste no desenvolvedor informar o tamanho da chave a ser gerada (512, 1024, 2048, etc.) em bits e também o tipo de algoritmo que será utilizado (**RSA**, **AES**, **DES**, etc.), com posse dessas informações o Framework cria os dados (*modulus* e *public*

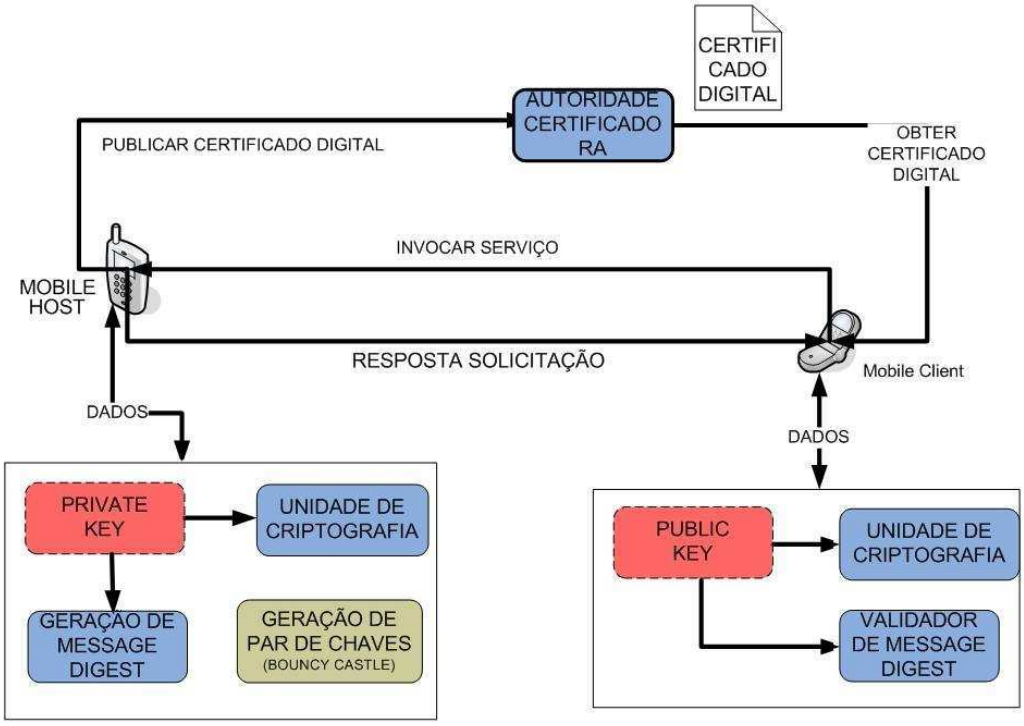

Figura 4.14: Serviço de Segurança: Confidencialidade e Integridade

exponent) para geração da chaves de forma aleatória. Estes dados também estão presentes no certificado digital correspondente ao serviço para que possam ser utilizados para gerar a sua chave pública na aplicação consumidora dos serviços. Para geração das chaves utiliza-se uma API denominada *Bouncy Castle*[6]. Uma vez concluído o processo, o par de chaves é armazenado no aparelho para posterior utilização. O trecho de código a seguir mostra o método responsável pela geração de chaves nos termos acima citados.

```
/**
 * Geração de par de chaves RSA
 * @param tamChave Tamanho da Chave
 * @throws Exception Erro na geração das chaves
```

[6]Bouncy Castle: API open source que permite o desenvolvimento de aplicações seguras, por meio da agregação de funcionalidades de segurança contidas em sua biblioteca. Bouncy Castle é uma coleção de APIs usadas em criptografia. Ela inclui APIs para as linguagem de programação Java e C#. Devido ser um projeto originalmente Australiano, portanto as restrições de exportação de software de criptografia dos Estados Unidos não se aplicam a ela.

```
 */
public AsymmetricCipherKeyPair generateRSAKeyPair (int tamChave)
throws Exception {
  SecureRandom modulus = new SecureRandom ();
  BigInteger exponent = new BigInteger ("10001", 16);
  RSAKeyGenerationParameters RSAKeyGenPara =
   new RSAKeyGenerationParameters (exponent, modulus, tamChave, 80);
  RSAKeyPairGenerator RSAKeyPairGen = new RSAKeyPairGenerator();
  RSAKeyPairGen.init(RSAKeyGenPara);
  AsymmetricCipherKeyPair keyPair = RSAKeyPairGen.generateKeyPair();
  return keyPair;
}
```

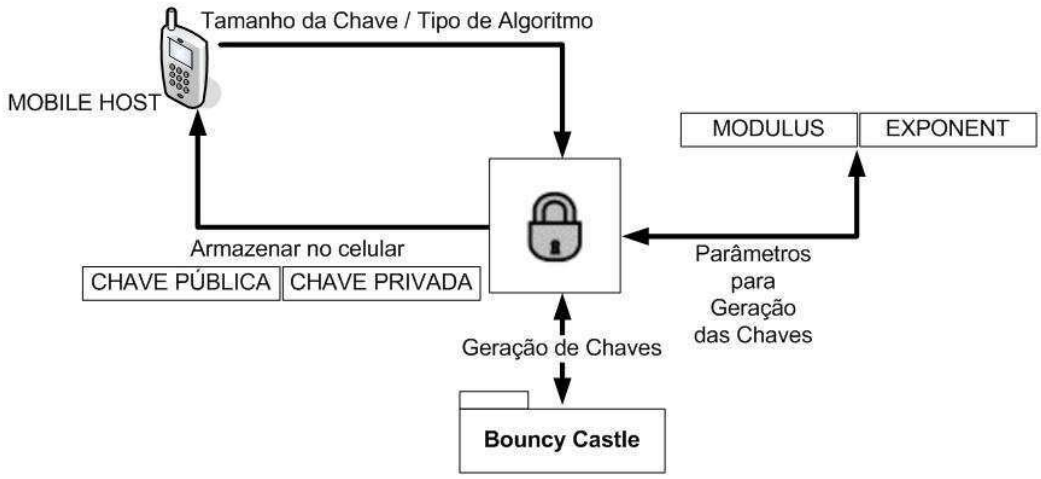

Figura 4.15: Geração de Chaves pelo Framework

4.6.2 Geração de Assinatura Digital

O framework é capaz de gerar uma assinatura digital para, posteriormente, adicioná-la a mensagem final a ser enviada para o requisitante, este processo pode ser visto na Figura 4.16, e seu funcionamento ocorre da seguinte maneira: inicialmente o Gerenciador de Serviços verifica se o serviço utiliza serviços de segurança (passo 1). Caso positivo, o processo de geração da assinatura é iniciado (passo 2), o que ativa a geração de um hash do corpo da mensagem (passo 3), o hash

gerado é convertido para o formato Hexadecimal (passo 4) e o resultado obtido é utilizado na composição final da mensagem de resposta.

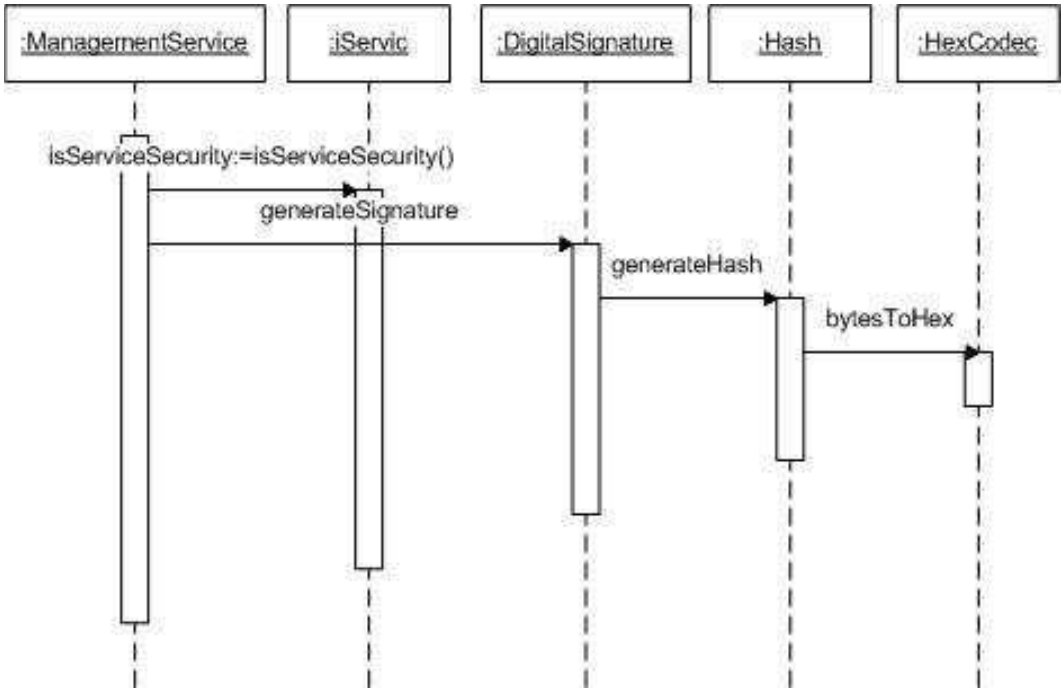

Figura 4.16: Diagrama de Sequência - Geração de Assinatura Digital

4.6.3 Decifrar Mensagem Recebida

Uma mensagem recebida pelo Mobile Host pode ter sido cifrada de acordo com o padrão *XML-Encryption* estabelecido pelo *WS-Secutity*, todavia o framework é apto para tratar este tipo de mensagem, conforme pode ser visto na Figura 4.17. Neste caso, os dados do método requisitado são cifrados usando a chave pública do serviço, a qual o consumidor obteve anteriormente, logo o Mobile Host precisa identificar qual é o serviço requisitado, para poder obter a sua chave privada e decifrar o conteúdo da mensagem.

4.7 Processo de Criação de Serviços

Quando um desenvolvedor deseja utilizar o framework proposto para criar um serviço ele deve fornecer os dados necessários para descrever o seu serviço, onde

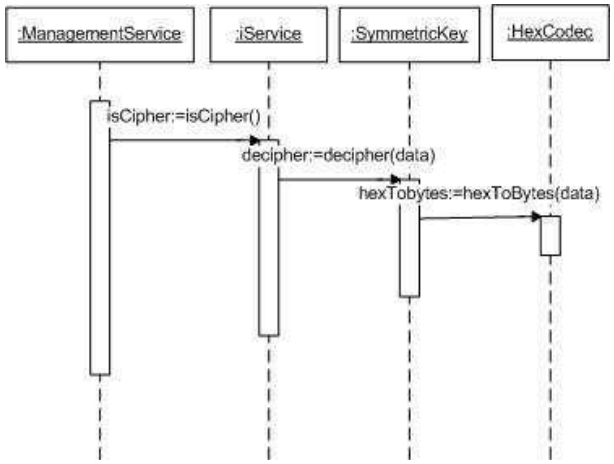

Figura 4.17: Diagrama de Sequência - Decifrar Mensagem Recebida

irá informar quais os seus métodos com seus respectivos parâmetros e o tipo de retorno do método criado. Além disso, deve identificar o modo de execução dos métodos criados para serem utilizados quando estes forem criados. Essas informações são passados ao **Gerente de Serviços**, que irá gerar automaticamente um documento WSDL, que contém os detalhes do serviço seguindo um padrão já bem conhecido, e o armazena em uma pasta específica no próprio dispositivo para posterior verificação, ou ser enviado para um possível consumidor. A Figura 4.18 mostra o mapeamento de uma classe java para um documento WSDL que contém a sua descrição.

Caso queria agregar serviços de segurança o desenvolvedor pode utilizar as interfaces pelo módulo de *Controle de Confidencialidade e Integridade* que é responsável pela geração de um par de chaves assimétricas (pública e privada) que serão utilizadas para criação de um Certificado Digital (que pode ser enviado para os consumidores finais) para o serviço e criar uma Assinatura Digital para cada mensagem transmitida, o que garante a sua integridade.

Após as etapas descritas anteriormente o serviço será armazenado na *Base de Dados de Serviços* para sua posterior utilização. Vários serviços podem ser criados e armazenados, permitindo que o desenvolvedor detenha um conjunto de serviços para disponibilizar aos seus consumidores, mas limitado as restrições de memória e capacidade do dispositivo onde eles serão alocados.

Paralelamente as atividades acima descritas um arquivo XML contendo os da-

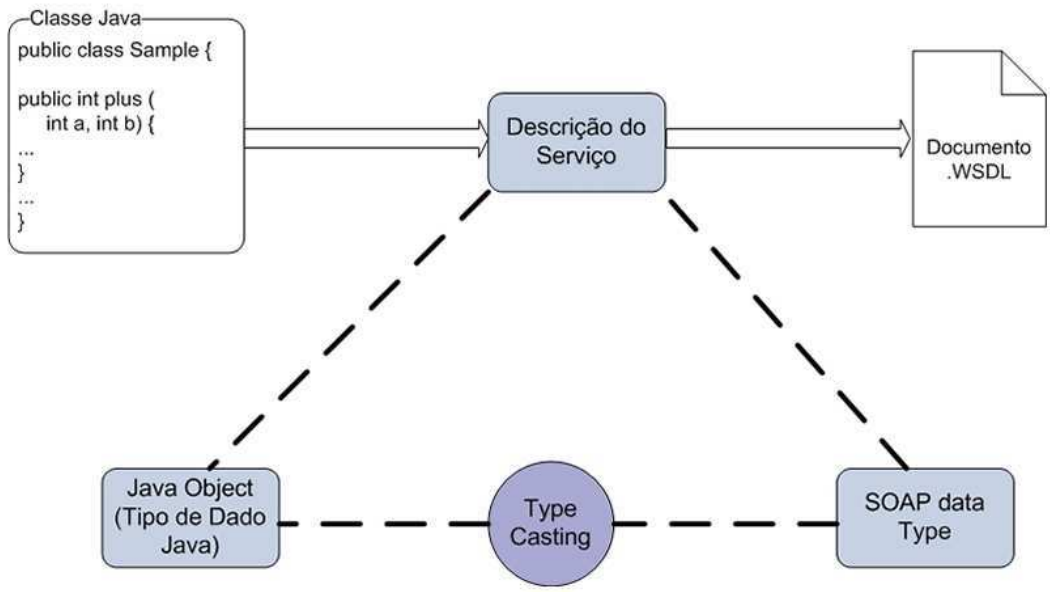

Figura 4.18: Mapeamento classe Java para documento WSDL

dos do serviço é criado, contendo informações como: nome do serviço, localização do documento WSDL, dados dos método do serviço (nome do método, tipos e nomes de parâmetros e tipo de retorno) e dados da configuração do serviço de segurança (chaves assimétricas, serviço de autenticação e autorização ativos e localização do certificado digital), no código a seguir podemos ver um exemplo deste arquivo.

```
<service>
   <serviceName value=""/>
   <address>
<protocol />
<url />
<port />
</address>
   <serviceDescription location="" />
  <methods>
  <method name="">
   <parameter name="" type="" />
   <parameter name="" type="" />
```

```
 <return type="" />
 </method>
 </methods>
<serviceSecurity>
 <active />
<symmetricKey>
 <publicKey value=""/>
 <privateKey value=""/>
 </symmetricKey>
 <authenticationService active="" />
 <authorizationService active="" />
 <certificateDigital />
</serviceSecurity>
</service>
```

A Figura 4.19 mostra o processo de criação de um serviço pelo framework proposto, o seu funcionamento ocorre da seguinte forma: o sistema verifica quais os serviços que devem ser criados (passo 1), verificando quais aqueles que já existem (passo 2), caso o serviço ainda não tenha sido criado os dados do novo serviço são enviados para o **Gerenciador de Serviços** executar esta tarefa (passo 3). Para que esta etapa possa ser executada alguns requisitos são necessários: o serviço deve implementar a interface **iService** e consequentemente os seus métodos, principalmente os métodos **describeService** e **executeMethod**, pois estes métodos são utilizados para, respectivamente: informar os dados dos métodos do serviço criado (como: tipo de retorno, quantidade de parâmetros, o tipo de cada parâmetro, e o seu nome) e a forma de como os métodos deverão ser executados após terem sido validados pelo sistema quando uma requisição é recebida, mesmo várias tarefas desempenhadas pelo framework serem automáticas a definição deste dois métodos é necessária. A próxima etapa é a geração da descrição do serviço por meio de um documento WSDL (passo 4 e 5) e o seu armazenamento no sistema de arquivo local (passo 6), posteriormente o Gerente de Serviços verifica se o desenvolvedor irá utilizar o serviço de segurança do framework (passo 7), em caso positivo o sistema irá gerar um par de Chaves Assimétricas para o serviço (passo 8) e também um Certificado Digital (passo 9) para publicação e sua Chave Pública e o ar-

mazenamento do certificado no sistema de arquivos local (passo 10). Concluídas todas as etapas citas anteriormente, o serviço está pronto para ser usado e as suas informações são armazenadas (passo 11) para serem utilizadas quando solicitado.

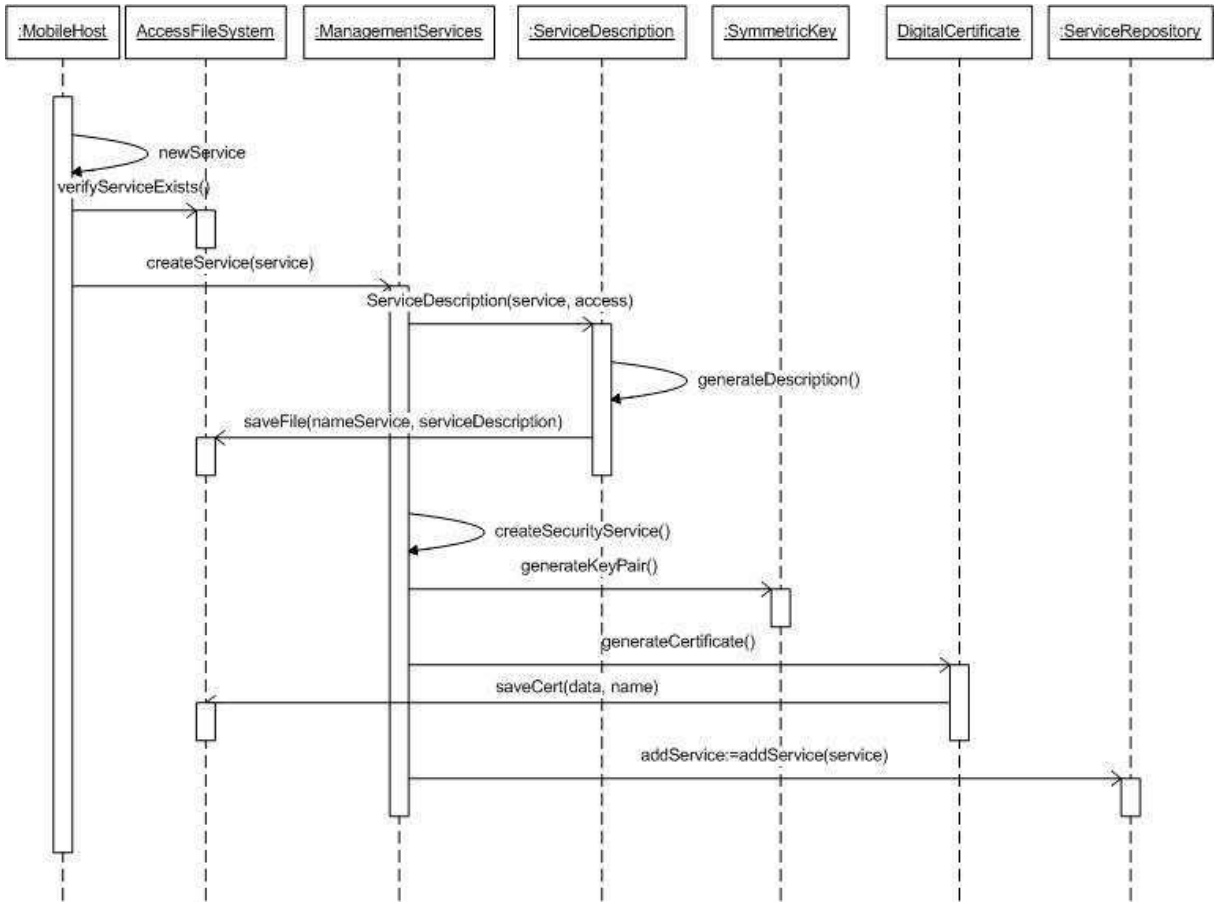

Figura 4.19: Diagrama de Sequência - Processo de Criação de Serviços

4.8 Análise Comparativa com os Trabalhos Relacionados

Uma análise comparativa dos principais trabalhos apresentados e a Arquitetura Proposta é feita nesta seção, conforme observa-se na Tabela 4.2 que mostra uma comparação entre a Arquitetura Proposta e outras soluções existentes, anteriormente apresentada. Na tabela pode-se observar que o principal ponto não observado pelos demais trabalhos é a ausência de uma solução para a segurança

	Arquitetura Proposta	Halonen	Sánchez-Nilsen	Kalasapur	Srirama
Provisão	Sim	Sim	Sim	Sim	Sim
Descoberta e Publicação	Sim	Sim	Sim	Sim	Sim
Descrição	Sim	Sim	Sim	Sim	Sim
Formato XML	Sim	Sim	Sim	Sim	Sim
Serviço de Segurança	Sim	Não	Não	Não	Não
Gerenciamento	Sim	Sim	Sim	Sim	Sim
Criação de Serviços no dispositivo	Sim	Não	Não	Não	Não
Parser de Mensagens	Sim	Sim	Sim	Sim	Sim
Composição	Não	Não	Sim	Sim	Não
Criação de Mensagens	Sim	Sim	Sim	Sim	Sim

Tabela 4.2: Tabela Comparativa da Arquitetura Proposta com outras Soluções Existentes

dos serviços que serão disponibilizados nos aparelhos. Logo, este aspecto foi o principal ponto abordado na solução proposta, que agregou um Serviço de Segurança para ser utilizado por SOA no Ambiente de Computação Móvel. Além disso, o Framework provê a criação de serviços no próprio dispositivo descrevendo o serviço que será disponibilizado. Entretanto, um ponto que não foi abordado pela Arquitetura proposta é a Composição de Serviços em tempo de execução para criação de serviços mais sofisticados e a adição de novos serviços por meio da sua migração de um dispositivo para outro, característica cuja solução foi proposta apenas por (Halonen 2006) e (Kalasapur 2005); a Composição de Serviço não foi um tema abordado pelo Framework devido a sua proposta está focada inicialmente em desenvolver um mecanismo de segurança para ser utilizado pelo Framework.

4.9 Resumo

Este capítulo se dedicou em abordar a Arquitetura do Framework que está sendo proposto, descrevendo os seus principais componentes e como eles estão relacionados. Além disso, a modelagem do sistema é apresentada focando na apresentação do funcionamento das suas principais funcionalidades, além de uma comparação com algumas das soluções existentes. O próximo capítulo mostrará um conjunto de estudos de casos que foram feitos com o intuito de avaliar o funcionamento do sistema.

Capítulo 5

Estudo de Casos

Neste capítulo serão apresentados alguns estudos de caso que foram desenvolvidos com o intuito de analisar o protótipo do Framework proposto, onde foram realizadas diversas simulações com os estudos de casos criados. Os serviços foram criados de acordo com os passos apresentados na seção 4.7, onde a implementação a interface *iService* permite a criação dos serviços, conforme podemos observar na Figura 5.1 que possui a modelagem de todos os estudos de caso aqui mensionados.

5.1 Ambiente de Teste

Na Tabela 5.1 é mostrada a configuração do computador onde as simulações onde os testes foram feitos, enquanto na Tabela 5.2 apresenta a configuração do Sun Java Wireless Tookit, ambiente que permitiu que as simulações no ambiente móvel fossem realizadas.

Categoria	Computador Pessoal
Placa-Mãe	Asus A8V-E SE
Processador	AMD ATHLON 60 3700+(2.2GHz)
Memória	1 GB RAM DDR
HD	80 GB
Sistema Operacional	Microsoft Windows XP SP2
Sistema Operacional Version	5.1

Tabela 5.1: Ambiente de Teste

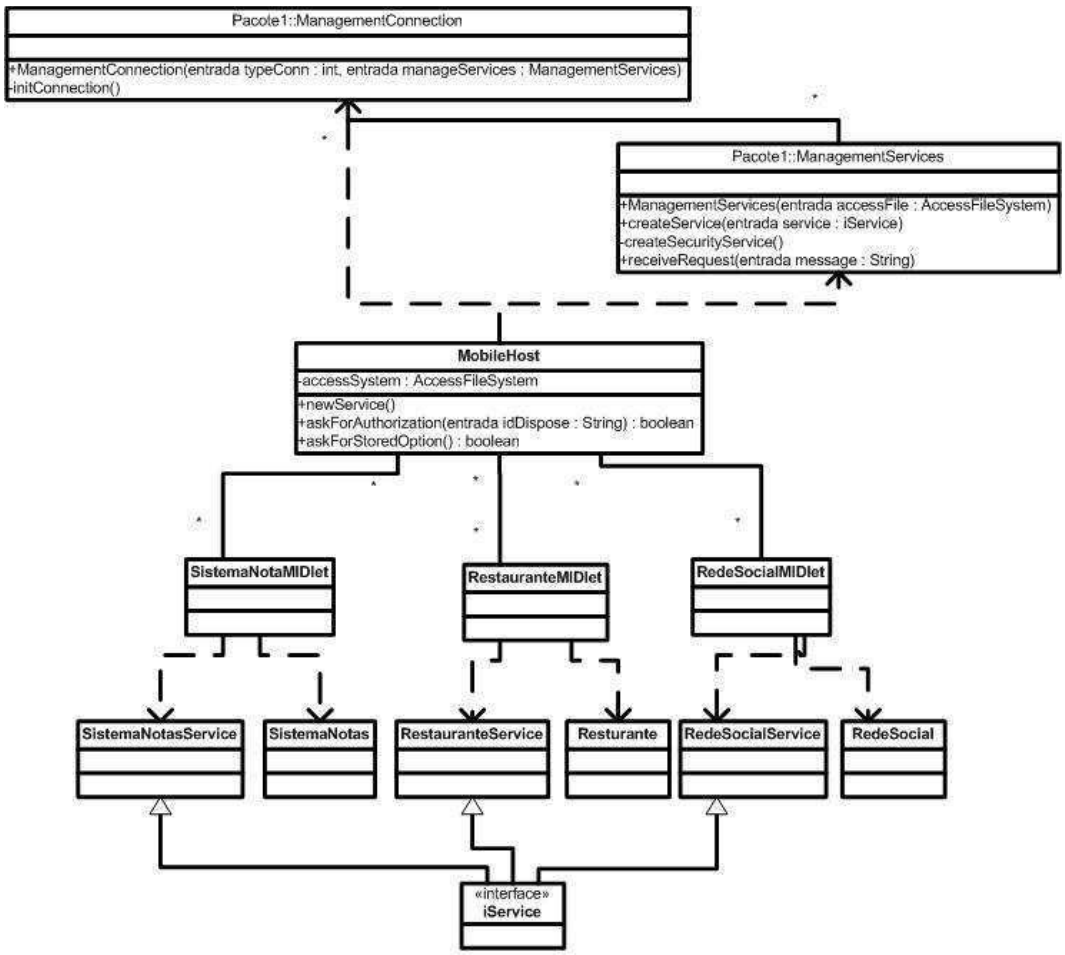

Figura 5.1: Diagrama de Classes dos Estudos de Caso

5.2 Estudo de Caso 1 - Controle Escolar

Este estudo de caso apresenta um Serviço que disponibiliza as notas de um conjunto de alunos de determinadas disciplinas que são armazenadas em um dispositivo móvel por intermédio de uma aplicação instalada no próprio dispositivo, a qual pode ser vista na Figura 5.3 (esta aplicação é executada no Mobile Host).

A Figura 5.2 contém os Casos de Uso deste Estudo de Caso que é composto de dois atores. O Ator **Professor**, que é a aplicação servidora e portanto instalada no Mobile Host, é responsável por cadastrar as disciplinas (Caso de Uso *Cadastrar Disciplina*) e e os alunos (Caso de Uso *Cadastrar Aluno*), também insere as notas de cada aluno relacionado a uma disciplina (Caso de Uso *Inserir Nota*). O Ator **Aluno**, que é a aplicação consumidora, obtém as displinas (Caso de Uso *Obter*

Versão J2ME	Version: 1.7.0.a
Versão WTK	2.5.1 for CLDC
Java vendor	Sun Microsystems Inc.
Java version	1.5.0_08
Edição	Developer Edition

Tabela 5.2: *Sun Java Wireless Toolkit*

Lista Disciplinas) e as suas notas de determinada disciplina (Caso de Uso *Obter Notas*).

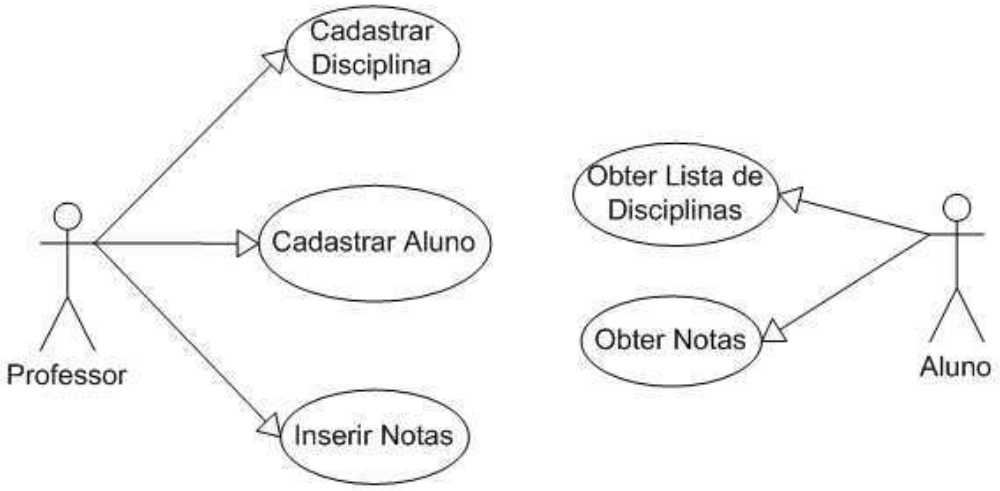

Figura 5.2: Diagrama de Caso de Usos - Sistema de Notas

A implementação do serviço é feito pela classe *SistemaNotas*, enquanto o serviço é criado através da implementação da interface *iService* pela classe *SistemaNotasService*, de acordo no trecho de código a seguir, podemos ver a implementação dos métodos **describeService** (para descrição do serviço e dos seus métodos) e **executeMethod** (para a execução dos métodos definidos pelo serviço onde são executados os métodos da classe *SistemaNotas* referentes ao método que foi invocado). Conforme mencionado anteriormente para cada serviço criado um arquivo contendo a descrição do mesmo é gerado, este arquivo referente ao serviço **SistemaNotas** pode ser visto no Apêndice A.

```
public class SistemaNotas extends iService {
```

```
/**
 * Implementação do método abstrato describeService pela
 * classe SistemaNotasService
 */
public void describeService() {

/* Informando o nome do serviço */
super.setNameService("SistemaNotas");

/* Serviço de Segurança */
super.setCipher(false);
super.setServiceAuthentication(false);
super.setServiceAutorization(false);

/* Método cadDisciplina */
String nameMethodCadDisc = "cadDisciplina";

/* Parametros de cadDisciplina */
Parameter param1CadDisc = new Parameter("codDisciplina",  Type.getT_STRING());
Parameter param2CadDisc = new Parameter("nomeDisciplina", Type.getT_STRING());

/* Tipo de retorno */
Parameter paramReturnCadDisc = new Parameter("cadDisciplina",
Type.getT_BOOLEAN());

/* Adiciona os parametros a um vetor */
Vector paramCadDisc = new Vector();
paramCadDisc.addElement(param1CadDisc);
paramCadDisc.addElement(param2CadDisc);
paramCadDisc.addElement(paramReturnCadDisc);

this.addMethod(nameMethodCadDisc, paramCadDisc);
```

```
/* Descrição de outros médotos */
}

/**
 * Implementação do método abstrato executeMethod pela
 * classe SistemaNotasService
 */
public String executeMethod(String nameService, Vector params) {

SistemaNotas sistemaNotas = new SistemaNotas ();

/*-------------------------------*/
if (nameService.equals("cadDisciplina")) {
Parameter p1 = (Parameter) params.elementAt(0);
Parameter p2 = (Parameter) params.elementAt(1);

String codDisciplina  = p1.getValue();
String nomeDisciplina = p2.getValue();

if (sistemaNotas.cadDisciplina(codDisciplina, nomeDisciplina)) {
resp = "true";
} else {
resp = "false";
}
}
/* Modo de execução dos outros métodos */
}

}
```

Quando um cliente deseja obter as notas de um determinado aluno ele deve criar uma requisição SOAP e enviá-la ao dispositivo onde a serviço está instalado. Na Figura 5.4, podemos observar o consumidor informando os dados que serão

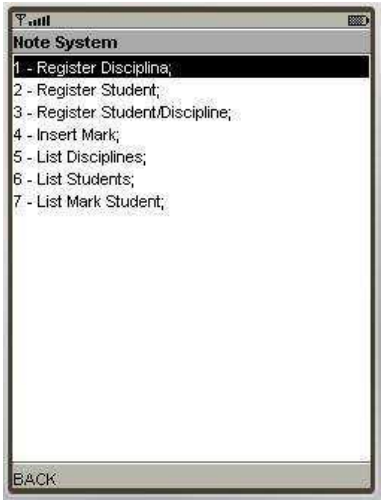

Figura 5.3: Aplicação de Sistema de Notas

utilizados na requisição.

Figura 5.4: Consumidor de serviço informando os dados para criar uma requisição.

Com estas informações uma requisição SOAP pode ser gerada conforme vemos na Figura 5.5. Assim esta mensagem pode ser enviada ao Móbile Host, onde o serviço está instalado.

Quando esta mensagem chega ao Mobile Host ela é identificada como uma requisição SOAP e é enviada para o *Gerenciador de Serviços* que é responsável pelo seu tratamento que inicialmente é a realização de um parser para obter o serviço solicitado e extrair os seus dados. Caso o serviço exista e efetuada uma análise dos dados para verificar a sua conformidade com serviço invocado, se

```
- <SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  - <SOAP-ENV:Body SOAP-
      ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    - <obterNotas xmlns="http://localhost:5000/CadastroEscolar.jws"
        id="o0" SOAP-ENC:root="1">
        <codAluno xmlns="" xsi:type="xsd:string">A001</codAluno>
        <codDisciplina xmlns=""
          xsi:type="xsd:string">D002</codDisciplina>
      </obterNotas>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Figura 5.5: Mensagem SOAP com a Solicitação das Notas

estiver de acordo o serviço é executado e uma mensagem (Figura 5.6) é gerada para ser enviada de volta para o requisitante. Caso o serviço não exista, os dados do método estejam incorretos ou ocorra alguma falha durante a execução do serviço uma mensagem de erro é gerada e enviada ao requisitante.

```
<?xml version="1.0" encoding="utf-8" ?>
- <soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  - <soap:Body>
    - <obterNotasResponse xmlns="http://www.dee.ufma.br/">
        <obterNotasResult
          xsi:type="xsd:string">#A001;D002;LACKS:;0#A001;D002;FINAL
          TEST:;0#A001;D002;REPLACEMENT:;0#A001;D002;NOTE
          3:;98#A001;D002;NOTE 2:;95#A001;D002;NOTE
          1:;100#</obterNotasResult>
      </obterNotasResponse>
    </soap:Body>
  </soap:Envelope>
```

Figura 5.6: Mensagem de resposta

Ao receber a mensagem de resposta a aplicação consumidora do serviço extrai o seu conteúdo e exibe as informações obtidas para o usuário do dispositivo, como podemos ver na Figura 5.7. A aplicação também deve está preparada para receber e exibir uma mensagem de erro, caso ocorra algum durante o tratamento da sua solicitação.

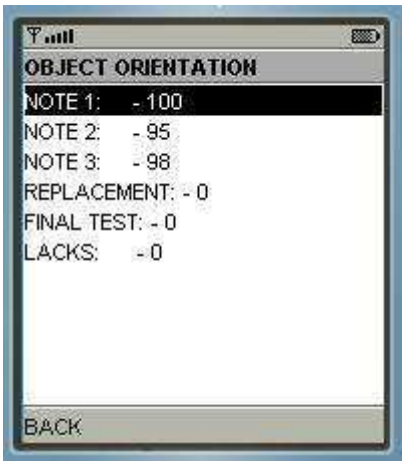

Figura 5.7: Informações Obtidas com a invocação do serviço sendo exibidas para o usuário do dispositivo

5.3 Estudo de Caso 2 - Restaurante

Este estudo de caso apresenta um Serviço de Cozinha de um Restaurante. Neste serviço, os pedidos são enviados por um garçom usando uma aplicação específica em qualquer lugar no restaurante para a cozinha, onde existe alguém com um aparelho para receber os pedidos, de acordo com a Figura 5.9. Entretanto, deve-se garantir que somente pessoas previamente registradas possam executar este serviço, para isso o serviço de Autenticação utilizado pelo sistema é utilizado.

A Figura 5.8 contém os Casos de Uso deste Estudo de Caso que é composto de dois atores. O Ator **Cozinheiro**, que é a aplicação servidora e portanto instalada no Mobile Host, é responsável por cadastrar as pratos (Caso de Uso *Cadastrar Pratos*) e os garçom com seus respectivos login e senha (Caso de Uso *Cadastrar Garçom*), também listar os pedidos existents (Caso de Uso *Listar Pedidos*). O Ator **Garçom**, que é a aplicação consumidora, obtém os pratos disponíveis naquele dia (Caso de Uso *Obter Pratos*) e envia os pedidos para o cozinheiro (Caso de Uso *Fazer Pedido*).

A implementação do serviço é feito pela classe *Restaurante*, enquanto o serviço é criado através da implementação da interface *iService* pela classe *RestauranteService*, no trecho de código que segue pode-se observar que o serviço criado irá utilizar o módulo de autenticação do Framework através do método **setServiceAuthentication**, no apendice B é mostrado o documento WSDL gerado pelo

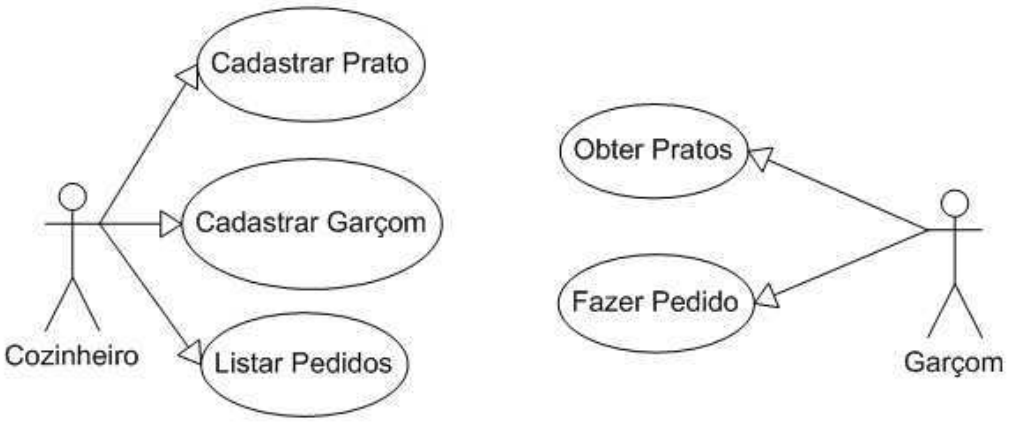

Figura 5.8: Diagrama de Caso de Usos - Restaurante

framework referente a este serviço.

```
public class RestauranteService extends iService {

public void describeService() {

/* Informando o nome do serviço */
super.setNameService("Restaurante");

/* Serviço de Segurança */
super.setCipher(false);
super.setSecurityServiceActive(false);
super.setServiceAuthentication(true);
super.setServiceAutorization(false);
...
```

Quando um consumidor do serviço deseja invocar um serviço, este deve criar uma mensagem SOAP de acordo com a especificação WS-Security a qual o Framework está apto a trabalhar. O usuário informa os dados de seu login, conforme vemos na Figura 5.10, e estes são utilizados para gerar a mensagem SOAP em questão. Para que a senha utilizada se mantenha confidencial enquanto trafega

Figura 5.9: Aplicação Restaurante que recebe as Solicitações de Pedidos

pela rede, deve ser gerado um hash desta senha e esta é quem deve ser enviado pela rede. A mensagem SOAP deve ser criada usando o elemento do UsernameToken do WS-Secutiry, pois provê uma construção que pode ser usada para enviar dados de autenticação e autorização (neste estudo de caso são utilizados os elementos filhos *Username* e *Password*). A mensagem completa da requisição pode se vista na Figura 5.11.

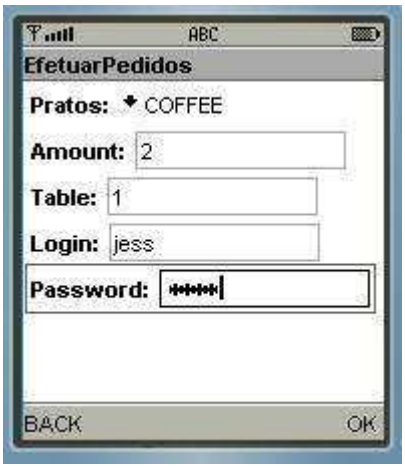

Figura 5.10: Usuário informando os dados para Solicitar um Serviço

Quando o Mobile Host identifica o serviço solicitado e que este exige a Autenticação do usuário, o sistema efetua a validação dos dados de Autenticação recebidos (que foram extraídos durante o processo de parser da mensagem) com os que estão armazenados no sistema - comparando o login e o hash da mensagem

e com o que está armazenado. Uma vez este processo concluído com êxito a requisição é continua a ser tratada, em caso contrário sua execução é abortada e uma mensagem informada falha na Autenticação é enviada ao requisitante.

```
- <SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:SOAP-
    ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/envelope/">
  - <Header>
    - <wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext">
      - <wsse:UsernameToken>
          <wsse:Username>jess</wsse:Username>
          <wsse:Password
            Type="wsse:PasswordDigest">136e3cf1ab6a4dc9cd25784ffe7ab05af45d9f77</wsse:Password>
        </wsse:UsernameToken>
      </wsse:Security>
    </Header>
  - <SOAP-ENV:Body SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    - <efetuarPedido xmlns="http://localhost:5000/Restaurante.jws" id="o0" SOAP-ENC:root="1">
        <cod xmlns="" xsi:type="xsd:string">P003</cod>
        <quantidade xmlns="" xsi:type="xsd:int">2</quantidade>
        <mesa xmlns="" xsi:type="xsd:int">1</mesa>
      </efetuarPedido>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Figura 5.11: Mensagem SOAP com o uso de WS-Security para Autenticação

5.4 Estudo de Caso 3 - Rede Social

Este estudo de caso trata de uma Aplicação de uma Rede Social, onde o usuário tem a possibilidade de estabelecer um canal de comunicação segura com os seus contatos. Para tanto, esta aplicação foi desenvolvida utilizado o Serviço de Segurança do Framework, mais precisamente com o uso de Assinatura Digitais e Cifragem das Mensagens transmitidas.

A Figura 5.12 contém o Caso de Uso deste Estudo de Caso que é o Ator **Usuário da Rede Social**, que é a aplicação servidora e consumidora dependendo da atividade que esteja exercendo. Nessa aplicação é possível criar cartões de visitas (Caso de Uso *Criar Cartão*) e enviá-lo para outro usuário (Caso de Uso *Enviar Cartão*), mas para garantir que sua origem seja verdadeira é possível assinar digitalmente o cartão (Caso de Uso *Assinar Cartão*). O um usuário pode enviar mensagens (Caso de Uso *Enviar Mensagem*) para seus contatos que estão armazenadosno dispositivo (Caso de Uso *Listar Contatos*) cujo conteúdo é criptografado (Caso de Uso *Cifrar Mensagem*).

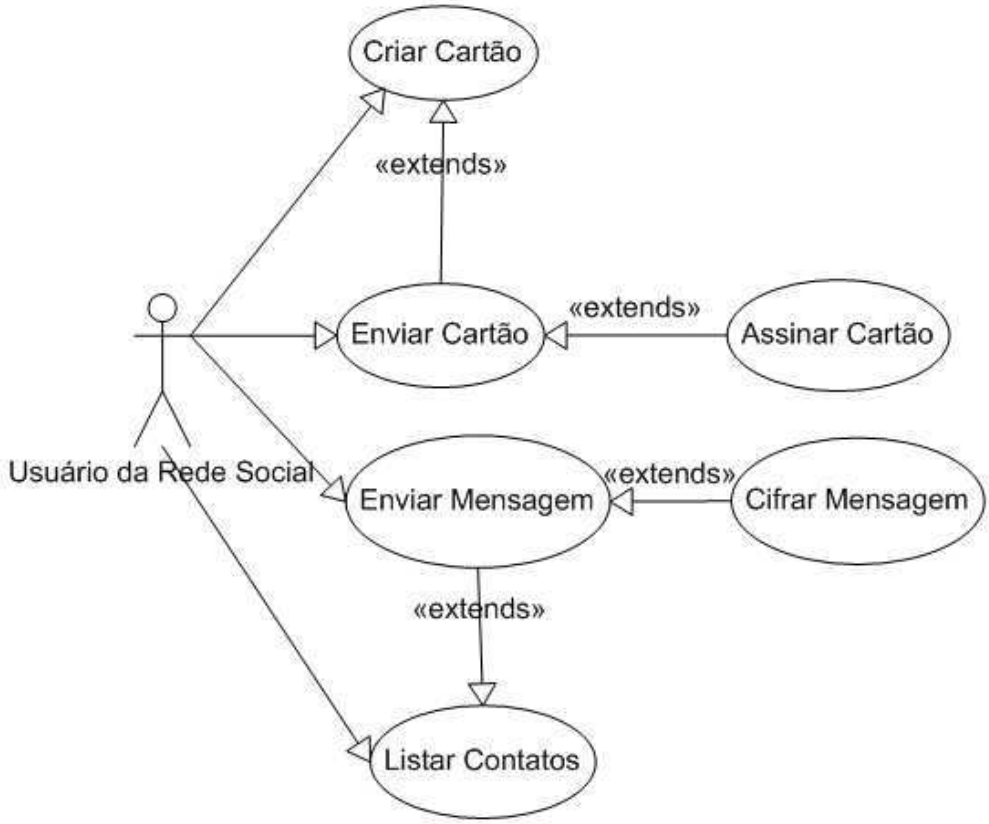

Figura 5.12: Diagrama de Caso de Usos - Rede Social

A implementação deste serviço é feito pela classe *RedeSocial*, enquanto o serviço é criado através da implementação da interface *iService* pela classe *RedeSocialService*, no techo de código que segue pode-se observar que o serviço criado irá utilizar o módulo de segurança, ou seja, irá criar o par de Chaves Assimétricas e gerará um Certificado Digital o qual pode ser visto no Apêndice C.

```
public class RedeSocialService extends iService {

RedeSocial rede;

public void describeService() {

/* Informando o nome do serviço */
super.setNameService("RedeSocial");
```

```
/* Serviço de Segurança */
super.setCipher(true);
super.setSecurityServiceActive(true);
super.setServiceAuthentication(false);
super.setServiceAutorization(false);

...
```

Cada usuário da aplicação pode criar um Cartão de Visitas que contém informações sobre ele e pode enviá-los para outros usuários, que podem adicioná-lo a sua lista de contatos. Um exemplo de mensagem transmitida pode se visto na Figura 5.13. Entretanto, a comprovação da autenticidade da mensagem transmitida, com os dados do emissor, torna-se necessária. Por isso, a mensagem é assinada digitalmente, utilizando os seus dados e a sua Chave Privada, desta foram, existe a comprovação da origem dos dados após o destinatário obter a Chave Pública (Através do seu Certificado Digital que é disponibilizado para o público geral) correspondente ao emissor e validar a mensagem recebida.

A Figura 5.14 mostra a lista de contatos de um usuário após os seus dados terem sido validados.

Além do envio de cartões um usuário pode enviar mensagens para os seus contatos, a Figura 5.15 mostra uma mensagem sendo criada para ser transmitida para um contato. Mas, para manter o sigilo do seu conteúdo seus dados devem passar, previamente, por um processo de encriptação.

Logo, a menagem SOAP correspondente estará com o seu conteúdo seguro, conforme se pode observar na Figura 5.16, que contém os valores dos seus campos cifrados de acordo com o padrão WS-Encrpty.

5.5 Resumo

Neste capítulo, um conjunto de estudos de casos criados foram apresentados com o intuito de verificar o funcionamento do Framework desenvolvido e assim, verificar se a solução proposta se adéqua à problemática apresentada. As Considerações finais serão feitas no próximo capítulo bem como sugestões para trabalhos futuros.

```
<SOAP-ENV:Envelope  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

    <header>
        <wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext">
            <Signature Id="RailCo333" xmlns="http://www.w3.org/2000/09/xmldsig#">
                <SignedInfo>
                    <CanonicalizationMethod
                        Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
                    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa"/>
                    <Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
                        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa"/>
                        <DigestValue></DigestValue>
                    </Reference>
                </SignedInfo>
                <SignatureValue>
                    4844dbf11f96305ba3d16221b8f8157cf58d0467d5
                    41f4883b0584ade2c4dc9af380fc3151cb4ab1302f
                    dd57819e6914de1444b6164acdfce8c02e8d263731
                    9106cd82ebb589cd6835b299b8910d8e906a246b54
                    22d0e87f51badecbaeb2b2c50b012f91ec9eef00d2
                    7a0adf78fb53271cf80c78b8cb2bca1d612a38284c
                    5f58
                </SignatureValue>
                <KeyInfo></KeyInfo>
            </Signature>
        </wsse:Security>
    </header>

    <SOAP-ENV:Body
            SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <enviarCartaoVisita xmlns="http://localhost:5001/RedeSocial.jws"
                id="o0" SOAP-ENC:root="1">
            <nome xmlns="" xsi:type="xsd:string">Alice</nome>
            <email xmlns="" xsi:type="xsd:string">alice@alice.com</email>
            <telefone xmlns="" xsi:type="xsd:string">1234-4567</telefone>
            <endereco xmlns="" xsi:type="xsd:string">localhost:5000</endereco>
        </enviarCartaoVisita>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 5.13: Mensagem SOAP com o uso de WS-Signature para Assinatura de Mensagem

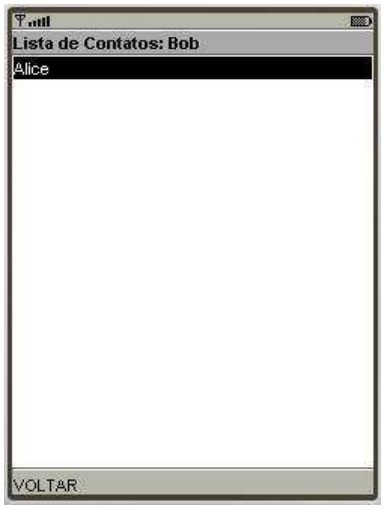

Figura 5.14: Lista de Contatos

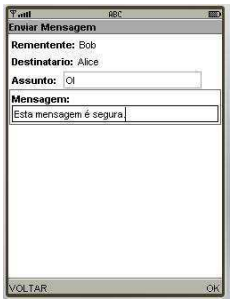

Figura 5.15: Aplicação exibindo Mensagem que será enviada

```
<SOAP-ENV:Envelope
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <enviarMensagem xmlns="http://localhost:5001/RedeSocial.jws" id="o0" SOAP-ENC:root="1">
        <rementente xmlns="" xsi:type="xsd:string">bob@bob.com</rementente>
        <assunto xmlns="" xsi:type="xsd:string">
            <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
                Type="http://www.w3.org/2001/04/xmlenc#Element">
                <CipherData>
                    <CipherValue>
                        5cf619edad7c464ae5080d62522df29ee9e4e7cb21ddb1c7641c6
                        68053701a961207f432f4834e6f3e51d1de8683e73f287c933f7f
                        f338c99211a0de79b2e0949a12041774d73955e2ae524ee2eb823
                        3343e7c5d914b233a85d4d17ecb62b4c3f275ecbfea977d06ca5a
                        e87886f58e2f55146090c301723573f56583b3a8ff9b
                    </CipherValue>
                </CipherData>
            </EncryptedData>
        </assunto>
        <mensagem xmlns="" xsi:type="xsd:string">
            <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
                Type="http://www.w3.org/2001/04/xmlenc#Element">
                <CipherData>
                    <CipherValue>
                        79d343a950f40a6ca327ad545754c5aa1895af59ae1585f53e111
                        f7825b4f10341ecf0fe5ad2210de87874bbea26fa676f532bc4f6
                        6355276aa7a6eefc9fcf16648a1f17f963b6865f0147a146e9602
                        1b3ec850b5bf7f867022176d15d8fc2e7780d5bba4db8407aadb8
                        558eff0885a4aa29305ecb7f30479c2b9c9ee18fde13
                    </CipherValue>
                </CipherData>
            </EncryptedData>
        </mensagem>
    </enviarMensagem>
</SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Figura 5.16: Mensagem SOAP que será envida cujo conteúdo está Cifrado

Capítulo 6

Conclusões e Sugestões para Trabalhos Futuros

Este capítulo apresenta as principais contribuições obtidas com o desenvolvimento deste trabalho, algumas considerações finais sobre o trabalho realizado, além de sugestões para trabalhos futuros.

6.1 Contribuições

A utilização de SOA para suportar aplicações destinadas a dispositivos móveis possibilita o aumento da oferta de novos serviços, tornando possível obter qualquer tipo de dados em qualquer dispositivo e em qualquer lugar. Logo, o principal objetivo deste trabalho foi realizar um estudo abrangente sobre o desenvolvimento orientado a serviços que tem um grande potencial para ser referência no desenvolvimento de software nos próximos anos.

Os estudos sobre aplicações orientadas a serviços permitiu que fossem identificadas as principais características para a construção desse tipo de aplicação no ambiente móvel.

Com base nessas informações, a construção de um Framework que fornecesse suporte ao desenvolvimento orientado a serviços tornou-se possível, onde os problemas levantados fossem solucionados. Além disso, o framework foi construído para funcionar como um Servidor Web e com isso, deixar que seus serviços pudessem utilizar todas as funcionalidades que este tipo de servidor pode fornecer.

Portando, a utilização de dispositivos móveis não apenas como requisitantes de serviços, mas também como provedores de serviços foi proposta nesta dissertação. Entretanto, para prover estas funcionalidades, seu desenvolvimento requer um maior esforço de desenvolvimento que pode se tornar extremamente complexo. Logo, percebemos a necessidade de uma ferramenta para o desenvolvimento e provisionamento de serviços em dispositivos móveis de forma mais automatizada.

A partir deste cenário, o Middleware foi introduzido para construção de SOA em ambiente móvel. Sua arquitetura visa fornecer a um desenvolvedor uma ferramenta para o rápido desenvolvimento de serviços no ambiente móvel, agregando todos os serviços necessários para a disponibilização do serviço, sendo disponibilizado em uma rede Ad-Hoc. Com isso, o desenvolvedor retira grande parte do seu desenvolvimento, pois a ferramenta proposta já possui mecanismos para realizar todas as funcionalidades necessárias para provisão de serviços, como descrever os serviços, realizar leitura e a conversão (parser) das mensagens de/para um formato específico, criar um canal de comunicação para receber e enviar mensagens.

A segurança é um ponto fundamental para desenvolvido de qualquer aplicação. Logo, a sua importância vai desde a camada de transporte (HTTPS, TCP/IP) até a camada SOAP. A incorporação de criptografia, simétrica ou assimétrica, nas mensagens XML transmitidas é opção para garantir a segurança "Fim a Fim", da comunicação. Logo, tanto o cabeçalho da mensagem XML como o conteúdo da mensagem podem ser transformados em um conteúdo criptografado, proporcionando assim um maior grau de segurança. Assim, um serviço de segurança, também foi incluído no projeto do framework, onde o desenvolvedor tem a opção de agregá-lo ao seu serviço final.

A escolha dos padrões baseados em XML, na solução proposta, foi devido não existir problemas de dependência de plataforma, servidores web ou navegadores, pois XML é um padrão aberto e totalmente compatível com os Web Services (SOAs). Além disso, os padrões de segurança baseados em XML oferecem uma proteção dos dados fim-a-fim (diferentemente de técnicas não XML). Conseqüentemente, elas podem ser utilizadas pelos Web Services (SOAs), fornecendo mecanismos para Confidencialidade e Integridade do conteúdo das mensagens e capacidade de Autenticação e Autorização de Usuários, o que possibilita uma solução completa para comunicação segura entre os consumidores, possíveis inter-

mediários e provedores de serviços.

6.2 Considerações Finais

Embora seja um requisito extremamente importante a adição do mecanismo de segurança exige um maior uso das capacidades de processamento do dispositivo, principalmente com o uso do par de chaves assimétricas. Portanto deve-se avaliar quais de suas funções se adequam a um serviço que será desenvolvido, pois:

- tempo de geração de chaves cresce de acordo com o tamanho que a mesma irá ter, comprometendo os recursos do dispositivo durante a sua geração, o que contrasta com os requisitos de um Mobile Host que não pode sobrecarregar o dispositivo onde ele está executando;
- tempo de cifragem/decifragem também aumenta, o que conseqüentemente aumenta o tempo de resposta de uma solicitação;
- o tamanho do bloco a ser cifrado deveria ser igual ou inferior ao tamanho da chave o que exige a geração de chaves maiores dependendo do conteúdo a ser transmitido;

6.3 Sugestões para Tabalhos Futuros

Como sugestão para trabalhos futuros, bem como para melhoria deste, temos:

Publicação de certificado digital em um repositório específico para sua obtenção pelos consumidores, pois nas simulações feitas o consumidor o obtém diretamente do provedor de serviços (Mobile Host).

Geração de uma ferramenta que permita a composição de serviços em tempo de execução e desta forma à criação de serviços mais complexos. Além da criação de um mecanismo que permita a Migração de Serviços de um aparelho para outro.

Criação de um mecanismo de detecção de intrusão que possibilite a verificação de ataques ao Mobile Host, o que pode ser feito através da agregação do Framework ao projeto NIDIA.

Submeter o protótipo criado a um conjunto de ataques para avaliação do desempenho do Serviço proposto em um ambiente hostil, bem como as contramedidas para solução dos problemas apresentados.

Detecção dos pontos de falha do Framework e adição de um mecanismo de Tolerância a Falhas para permitir maior confiabilidade ao Framework.

Adição de um Mediador de Serviços, cuja função é transmitir as requisições entre consumidor e o provedor de serviços, quando o consumidor se encontra fora da área de abrangência do provedor em um ambiente P2P.

Identificar quando um consumidor está fora da área de atuação do Mobile Host e desta forma impedir a perda de mensagens destinadas a esse dispositivo.

Criar um mecanismo que identifique as tecnologias existentes no dispositivo em tempo de execução. Assim, verificar qual o melhor canal para envio de mensagens, levando em consideração aspectos como largura de banda disponível, ociosidade do canal, etc.

Apêndice A

Arquivo gerado pelo Framework contendo os detalhes do Serviço Criado

```
- <service>
  <serviceName value="CadastroEscolar" />
   <address>
<protocol>UDP</protocol>
<url>127.0.0.1</url>
<port>5000</port>
</address>
  <serviceDescription location="" />
- <methods>
- <method name="cadAlunoDisc">
  <parameter name="codAluno" type="xsd:string" />
  <parameter name="codDisc" type="xsd:string" />
  <parameter name="turma" type="xsd:string" />
  <parameter name="cadAluno" type="xsd:boolean" />
  </method>
- <method name="obterNotas">
  <parameter name="codAluno" type="xsd:string" />
  <parameter name="codDisciplina" type="xsd:string" />
```

```
  <return type="xsd:string" />
  </method>
- <method name="obterAlunos">
  <parameter name="alunos" type="xsd:string" />
  <return type="xsd:string" />
  </method>
- <method name="cadDisciplina">
  <parameter name="codDisciplina" type="xsd:string" />
  <parameter name="nomeDisciplina" type="xsd:string" />
  <return type="xsd:boolean" />
  </method>
- <method name="obterDisciplinas">
  <parameter name="disciplinas" type="xsd:string" />
  <return type="xsd:string" />
  </method>
- <method name="cadAluno">
  <parameter name="codAluno" type="xsd:string" />
  <parameter name="nome" type="xsd:string" />
  <return type="xsd:boolean" />
  </method>
- <method name="inserirNota">
  <parameter name="codAluno" type="xsd:string" />
  <parameter name="codDiscplina" type="xsd:string" />
  <parameter name="tipo" type="xsd:string" />
  <parameter name="valor" type="xsd:string" />
  <return type="xsd:boolean" />
  </method>
  </methods>
- <serviceSecurity active="false">
  <authenticationService active="false" />
  <authorizationService active="false" />
- <assymmetricKey>
  <publicKey value="" />
```

```
<privateKey value="" />
</assymmetricKey>
<certificateDigital value="" />
</serviceSecurity>
</service>
```

Apêndice B

WSDL - Serviço de Restaurante

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
targetNamespace="urn:Restaurante"
xmlns:impl="urn:Restaurante-impl"
xmlns:intf="urn:Restaurante"

<types>
<schema targetNamespace="http://abcom.com/Restaurante.xsd" xmlns="http://www.w3.
</schema></types>
<wsdl:message name="getPratosRequest">
<wsdl:part name="pratos" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getPratosResponse">
<wsdl:part name="getPratosReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="efetuarPedidoRequest">
<wsdl:part name="cod" type="xsd:string"/>
<wsdl:part name="quantidade" type="xsd:int"/>
<wsdl:part name="mesa" type="xsd:int"/>
</wsdl:message>
<wsdl:message name="efetuarPedidoResponse">
<wsdl:part name="efetuarPedidoReturn" type="xsd:boolean"/>
```

```
</wsdl:message>
<wsdl:message name="cadastrarPratoRequest">
<wsdl:part name="cod" type="xsd:string"/>
<wsdl:part name="descricao" type="xsd:string"/>
<wsdl:part name="preco" type="xsd:float"/>
</wsdl:message>
<wsdl:message name="cadastrarPratoResponse">
<wsdl:part name="cadastrarPratoReturn" type="xsd:boolean"/>
</wsdl:message>
<wsdl:message name="cadastrarGarcomRequest">
<wsdl:part name="cod" type="xsd:string"/>
<wsdl:part name="nome" type="xsd:string"/>
<wsdl:part name="login" type="xsd:string"/>
<wsdl:part name="senha" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="cadastrarGarcomResponse">
<wsdl:part name="cadastrarGarcomReturn" type="xsd:boolean"/>
</wsdl:message>


<wsdl:portType name="Restaurante">
<wsdl:operation name="getPratos" parameterOrder="pratos">
<wsdl:input  message="impl:getPratosRequest"
name="getPratosRequest" />
<wsdl:output message="impl:getPratosResponse"
name="getPratosResponse" />
</wsdl:operation>
<wsdl:operation name="efetuarPedido" parameterOrder="cod quantidade mesa">
<wsdl:input  message="impl:efetuarPedidoRequest" name="efetuarPedidoRequest" />
<wsdl:output message="impl:efetuarPedidoResponse" name="efetuarPedidoResponse" /
</wsdl:operation>
<wsdl:operation name="cadastrarPrato" parameterOrder="cod descricao preco">
<wsdl:input  message="impl:cadastrarPratoRequest"
```

```
name="cadastrarPratoRequest" />
<wsdl:output message="impl:cadastrarPratoResponse"
name="cadastrarPratoResponse" />
</wsdl:operation>
<wsdl:operation name="cadastrarGarcom" parameterOrder="cod nome login senha">
<wsdl:input  message="impl:cadastrarGarcomRequest"
name="cadastrarGarcomRequest" />
<wsdl:output message="impl:cadastrarGarcomResponse"
name="cadastrarGarcomResponse" />
</wsdl:operation>
</wsdl:portType>

<wsdl:binding name="RestauranteSoapBinding" type="impl:Restaurante">
<wsdlsoap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="getPratos">
<wsdlsoap:operation soapAction="" />
<wsdl:input name="getPratosRequest">
<wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://DefaultNamespace" use="encoded" />
</wsdl:input>
<wsdl:output name="getPratosResponse">
<wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost/services/Restaurante.java"
use="encoded" />
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="efetuarPedido">
<wsdlsoap:operation soapAction="" />
<wsdl:input name="efetuarPedidoRequest">
<wsdlsoap:body
```

```
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://DefaultNamespace" use="encoded" />
</wsdl:input>
<wsdl:output name="efetuarPedidoResponse">
<wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost/services/Restaurante.java"
use="encoded" />
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="cadastrarPrato">
<wsdlsoap:operation soapAction="" />
<wsdl:input name="cadastrarPratoRequest">
<wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://DefaultNamespace" use="encoded" />
</wsdl:input>
<wsdl:output name="cadastrarPratoResponse">
<wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost/services/Restaurante.java"
use="encoded" />
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="cadastrarGarcom">
<wsdlsoap:operation soapAction="" />
<wsdl:input name="cadastrarGarcomRequest">
<wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://DefaultNamespace" use="encoded" />
</wsdl:input>
<wsdl:output name="cadastrarGarcomResponse">
<wsdlsoap:body
```

```
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost/services/Restaurante.java"
use="encoded" />
</wsdl:output>
</wsdl:operation>
</wsdl:binding>

<wsdl:service name="RestauranteService">
<wsdl:port binding="impl:RestauranteSoapBinding" name="Restaurante">
<wsdlsoap:address location="http://localhost/services/Restaurante"/>
</wsdl:port>
</wsdl:service>

</wsdl:definitions>
```

Apêndice C

Cetificado Digital Referente ao Serviço Rede Social

Type: X.509v1

Serial number: 37:30:39:38:37:39

SubjectDN: MobileHost/

IssuerDN: MobileHost/

Start Date: Wed Jan 28 19:20:09 UTC 2009

Final Date:Sat Feb 07 19:20:09 UTC 2009

Public Key: RSA

modulus:
1031664852432424125064287537258174701724197322680481908255627
9229584820385511015237215039792565075682135132278955127100542
0335160795811607399198895160890626983380141099275635060667228
9907913709832133413787247613560338839353978427691677702348710
9752188166663497513629121821811506812853619016584726258826008

```
1497

public exponent:65537

Signature Algorithm: RSA

Signature:
442342479522874134863916217424860821272251392225065533828082 2
22855214313024345177246527322818182242479817112525019822517 5
23118387106483248598639254601602197882401853512239220126227 16
28294682235667104601861271253812412541923724798252223317533 20
32141835718721424596124251192511332021902371023816665353213 24
508221471271017616237646
```

Referências Bibliográficas

3G (2007). Securing mobile commerce interactions through secure mobile web services security.

Alliance, Open Mobile (2006). Oma web services enabler (owser) : Overview. technical specification.

Amorim, S. (2004). A tecnologia web service e sua aplicação num sistema de gerência de telecomunicações.

Bernardino, Anabela Moreira (2004). Autenticação: Mestrado e curso de especialização em sistemas de informação tecnologias para o comércio electrônico. Universidade do Minho.

Biryukov, Alex (2001). Real time cryptanalysis of a5/1 on a pc. In: *In FSE'00: Proceedings of the 7th International Workshop on Fast Software Encryption.* SpringerVerlag. p. 118.

Buttyán, Levente (2002). Report on a working session on security in wireless ad hoc networks. In: *Mobile Computing and Communications Review* (7, Ed.). Vol. I.

Candolin, Catharina (2007). A security framework for service oriented architectures. IEEE.

de Carvalho Ferreira, Lucas (2005). Relatório técnico segurança de web services.

Dokovski, Nikolai (2004). Paradigm: Service oriented computing. University of Twente.

e P. Jacquet, T. Clausen (2003). Olsr. In: *Optimized Link State Routing Protocol, RFC 3626* (The Internet Society, Ed.). Network Working Group.

Erl, T. (2005). Services-oriented architecture, concepts, technology, and design. In: *Services-Oriented Architecture, Concepts, Technology, and Design*.

et al O'Neill, Mark (2003). Web services security. In: *Web Services Security*.

et al Potts, Stephen (2003). Aprenda em 24 horas web services. In: *Aprenda em 24 horas Web Services*.

G. Sanders, L. Thorens, M. Reisky O. Rulik e S. Deylitz (2003). Gprs networks. In: *John Wiley and Sons*.

Gupta, V. (2001). Securing the wireless internet. In: *Communications Magazine, IEEE*.

Halonen, T. (2006). Cross-layer design for providing service oriented architecture in a mobile ad hoc network. In: *MUM06* (Stanford, Ed.).

Kalasapur, Swaroop (2005). Seamless service composition (sesco) in pervasive environments. In: *MSC 2005*. ACM. pp. 11–20.

Leymann, F. (2002). Web services and business process managment. In: *IBM Systems Journal*. IBM. pp. 198-211.

Moyo, T. (2006). Securing mobile commerce interactions through secure mobile web services. In: *In 8th Annual Conference on WWW Applications: South Africa*.

OASIS (2004). Web services security.

OASIS (2008). Reference architecture for service oriented architecture version 1.0.

Papazoglou, M. (2003). Serviceoriented computing. In: *Commmunications of the ACM: ServiceOriented Computing, Services and Technologies*. ACM. pp. 25-28.

Pawar, Pravin (2007). A comparative study of nomadic mobile service provisioning approachess. In: *The 2007 International Conference on Next Generation Mobile Applications, Services and Technologies*. IEEE.

Schwarz, Jerry (2007). Security challenges, threats and countermeasures version 1.0. In: *Technical Specification, The Web Services-Interoperability Organization,*.

SánchezNilsen, Elena (2006). An open and dynamical service oriented architecture for supporting mobile service. In: *ICWE'06*. ACM.

Srirama, Satish N. (2006*a*). A mediation framework for mobile web service provisioning. In: *10th IEEE International Enterprise Distributed Object Computing Conference Workshops*. IEEE.

Srirama, Satish N. (2006*b*). Mobile host: A feasibility analysis of mobile web service provisioning. In: *Ubiquitous Mobile Information and Collaboration Systems*.

Srirama, Satish N. (2006*c*). Mobile web service provisioning. In: *Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services*.

Srirama, Satish N. (2007). Security analysis of mobile web service provisioning. In: *Int. J. Internet Technology and Secured Transactions*. Inderscience Enterprises 2007.

W3C (2006). Extensible markup language (xml) 1.0 (fourth edition).