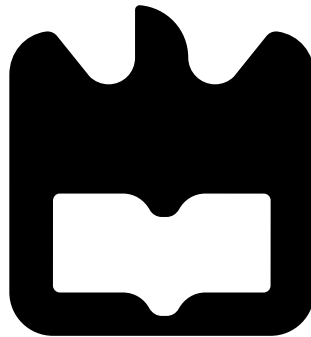




**Luís Miguel  
Borges Pinto**

**Serviços para integração regional de dados clínicos**







**Luís Miguel  
Borges Pinto**

## **Serviços para integração regional de dados clínicos**

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor João Paulo T. S. Cunha, Professor Associado com Agregação do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e do Mestre Ilídio Fernando de Castro Oliveira, Assistente Convocado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.



**júri / the jury**

presidente / president

**Prof. Doutor José Luís Guimarães Oliveira**

Professor Associado da Universidade de Aveiro

vogais / examiners committee

**Prof. Doutor João Paulo Trigueiros da Silva Cunha**

Professor Associado com Agregação da Universidade de Aveiro (Orientador)

**Prof. Doutor João Gonçalo Gomes de Paiva Dias**

Professor Coordenador da Universidade de Aveiro

**Mestre Ilídio Fernando de Castro Oliveira**

Assistente Convidado da Universidade de Aveiro (Coorientador)



## Resumo

Os sistemas de informação clínica têm sido reconhecidos como uma tecnologia chave no suporte à prestação de cuidados de saúde. Dada a complexidade do domínio clínico e o largo espectro de requisitos, vários sistemas heterogêneos têm sido introduzidos, criando aquilo a que se chama "silos de informação". A plataforma Rede Telemática da Saúde (RTS) tem vindo a ser desenvolvida por forma a providenciar uma rede regional de informação clínica eletrónica na região de Aveiro, permitindo que instituições parceiras troquem dados clínicos. Ao contrário de protocolos ponto-a-ponto que assentam na troca de mensagens ou documentos, a RTS está desenvolvida numa abordagem arquitetural, disponibilizando um modelo da informação partilhado e coerente, acessível através de um portal web.

No presente trabalho, abordamos as necessidades básicas na expansão da RTS para a colaboração com sistemas externos. Consideram-se para tal dois grandes requisitos: a expansão das ferramentas semânticas aplicadas às correspondências entre conceitos e uma interface estável para a invocação por serviços.

O modelo de dados de referência da RTS assenta num modelo de objetos partilhado. Um novo componente foi introduzido na aplicação empresarial já existente com o intuito de expor este modelo através de serviços SOAP, que implementam a especificação WS-Security.

As transformações semânticas na RTS são suportadas na sua maioria por módulos de transformação, conhecidos como "Wrappers". Propusemos e implementamos um componente de semântica capaz de aplicar transformações ao nível dos serviços, tirando partido de uma ferramenta do tipo servidor de vocabulário - LexEVS - essencialmente para mapear correspondências entre diferentes sistemas de classificação de informação clínica. O cenário para a aplicação dos mapeamentos foi escolhido indo ao encontro das necessidades do projeto internacional Smart Open Services for European Patients (epSOS), tendo sido criado para tal um serviço que disponibiliza informação com dados codificados com terminologias usadas no epSOS.

As extensões à RTS aqui propostas, nomeadamente a camada de serviços de invocação segura e a adoção de ferramentas para o mapeamento interno de conceitos, foram desenvolvidas com sucesso e integradas na plataforma. Cenários de teste foram realizados com a ajuda de grupos de alunos. Como prova de conceito (*proof of concept*), os serviços foram testados por duas aplicações externas, desenvolvidas em ambiente académico.

Conseguimos com sucesso implementar um conjunto de serviços, configurados para serem acedidos de forma segura, e um servidor de vocabulário para a interoperabilidade semântica. O que possibilita que no futuro sistemas externos possam consultar dados na RTS.





## Abstract

Health information systems have been acknowledged as a key technology to support care workflows. Given the complexity of the health domain and the large spectrum of requirements, many heterogeneous systems have been introduced, originating the so called “information silos”. The Rede Telemática da Saúde (RTS) platform has been developed to offer a regional electronic health information network in the region of Aveiro, allowing the partner institutions to exchange clinical data. Unlike point-to-point message or documents exchanging protocols, RTS builds on an architectural approach, exposing a coherent shared information model accessible through a Portal system.

In this work, we address the needs rooted to extending RTS to collaborate with external systems. Two major capabilities are deemed required: extending semantic tools for concept matching and a stable services invocation interface.

RTS reference information model builds on a shared object model. A new component was introduced on the existing enterprise application to expose the existing model through stable SOAP services, implementing the WS-Security specification.

The semantic transformations in RTS are mainly supported by the work of adapter modules, called Wrappers. We have proposed and implemented semantic transformation capability on services layer, taking advantage of a vocabulary server tool LexEVS, especially to map different medical classification systems. The mapping scenarios were demonstrated taking as an example the needs of the Smart Open Services for European Patients (epSOS) international specifications, by creating a service exposing information with data encoded with epSOS terminologies.

The proposed extensions to RTS, namely the secure services invocation layer and the adoption of semantic tools for internal concept mapping, were successfully developed and integrated in the platform.

Validation test cases were run with the assistance of student teams. As a proof of concept, the services have been tested by two external applications developed in the academic environment.

We successfully implemented a set of services configured to only be accessed in a safe way, and a vocabulary server to the semantic interoperability. This provides the means for in the future external systems can consult the RTS data.



# Conteúdo

<b>Conteúdo</b>	<b>i</b>
<b>Lista de Figuras</b>	<b>iii</b>
<b>Lista de Tabelas</b>	<b>v</b>
<b>Acrónimos</b>	<b>vii</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Enquadramento do tema . . . . .	1
1.2 Objetivos e âmbito da dissertação . . . . .	2
1.2.1 Objetivo principal . . . . .	2
1.2.2 Objetivos específicos . . . . .	3
1.3 Estrutura da Dissertação . . . . .	3
<b>2 Síntese do estado da arte</b>	<b>5</b>
2.1 Aplicações distribuídas baseadas em serviços . . . . .	5
2.1.1 Conceito . . . . .	5
2.1.2 Service-Oriented Architecture . . . . .	5
2.1.3 Simple Object Access Protocol . . . . .	6
2.1.4 Simple Object Access Protocol em Java . . . . .	8
2.2 Segurança em Web Service . . . . .	11
2.2.1 Controlo de acesso . . . . .	11
2.2.2 Segurança na comunicação . . . . .	12
2.3 Interoperabilidade semântica em saúde . . . . .	13
2.3.1 Interoperabilidade semântica . . . . .	13
2.3.2 Semântica no contexto da informação clínica . . . . .	15
<b>3 Cenários de integração na Rede Telemática da Saúde</b>	<b>17</b>
3.1 A RTS como um repositório regional acessível por serviços . . . . .	17
3.1.1 Cenários possíveis . . . . .	19
3.2 Cenário a implementar . . . . .	24
3.3 Casos de uso . . . . .	25
<b>4 Modelo computacional do sistema</b>	<b>27</b>
4.1 Interface de serviços . . . . .	27
4.1.1 Nova Arquitetura da aplicação . . . . .	27

4.1.2	Serviços a disponibilizar . . . . .	29
4.1.3	Modelo de Dados . . . . .	29
4.2	Serviços seguros . . . . .	31
4.2.1	Controlo da utilização . . . . .	31
4.2.2	Comunicação Segura . . . . .	32
4.3	Módulo de semântica . . . . .	32
4.3.1	Servidor de vocabulário . . . . .	32
4.3.2	Casos de utilização . . . . .	33
<b>5</b>	<b>Implementação</b>	<b>35</b>
5.1	Web Services (WSs) . . . . .	35
5.1.1	Modulo de serviços da Rede Telemática da Saúde (RTS) . . . . .	35
5.1.2	Serviços disponibilizados . . . . .	36
5.2	Modelo de dados . . . . .	36
5.3	Segurança nos serviços da RTS . . . . .	40
5.3.1	Controlo de acesso . . . . .	40
5.3.2	Comunicação segura . . . . .	43
5.4	Modulo de Semântica . . . . .	43
5.5	Garantia de funcionamento . . . . .	45
5.5.1	jUnit . . . . .	46
5.5.2	Aplicação Cliente . . . . .	46
5.5.3	Teste com aplicações de alunos . . . . .	47
<b>6</b>	<b>Conclusões</b>	<b>49</b>
6.1	Discussão de resultados . . . . .	49
6.2	Aspetos em aberto e trabalho Futuro . . . . .	50
	<b>Referências</b>	<b>51</b>
<b>A</b>	<b>Tabelas técnicas dos métodos disponibilizados na API de WSs</b>	<b>53</b>
A.1	DirectoryAndAuthority . . . . .	53
A.2	HealthRecord . . . . .	53
A.3	SysDiagnosis . . . . .	58
A.4	PatientIndex . . . . .	58

# Lista de Figuras

1.1	Principais conceitos da Dissertação . . . . .	2
2.1	Principais categorias das normas WS-* . . . . .	7
2.2	Arquitetura Metro . . . . .	9
2.3	Mecanismo de atuação do JAXB . . . . .	10
2.4	Modelo WSIT . . . . .	10
2.5	Utilização de chaves assimétricas . . . . .	11
3.1	Representação geográfica dos parceiros da RTS . . . . .	18
3.2	Arquitetura da RTS . . . . .	19
3.3	A RTS como fonte de dados para ATs . . . . .	20
3.4	Cenário com federação de RTSs . . . . .	21
3.5	Várias RTSs podem ser as fontes para o RSE . . . . .	22
3.6	A RTS a disponibilizar dados no formatos epSOS . . . . .	23
3.7	Casos de uso para os WSs da RTS . . . . .	25
4.1	Arquitetura da RTS. Antes (a) e depois (b). . . . .	28
4.2	<i>Patient Summary</i> . . . . .	30
5.1	Diagrama de classes - Utente . . . . .	37
5.2	Diagrama de classes - Episódio . . . . .	38
5.3	Diagrama de classes - Registo Clínico . . . . .	39
5.4	Utilização normal do sistema . . . . .	42
5.5	Aplicação <i>standalone</i> de interação com o LexEVS . . . . .	45



# Lista de Tabelas

1.1	Conceitos principais em contexto . . . . .	2
4.1	Serviços a criar na RTS . . . . .	29
4.2	Elementos do <i>Patient Summary</i> . . . . .	30
4.3	Elementos do <i>PatientEHR</i> . . . . .	30
5.1	Serviços a criar na RTS . . . . .	36
5.2	Directory and Authority . . . . .	40
5.3	initSession . . . . .	41
A.1	DirectoryAndAuthority . . . . .	53
A.2	retrieveHealthCareUnits . . . . .	53
A.3	HealthRecord . . . . .	54
A.4	retrieveEpisode . . . . .	54
A.5	retrieveEpisodeEssentials . . . . .	55
A.6	findEpisodesByPatientUID . . . . .	55
A.7	findEpisodesByPatientNID . . . . .	55
A.8	findEpisodesByPatientNIDandStartDate . . . . .	56
A.9	findEpisodesByPatientNIDStartDateAndType . . . . .	56
A.10	findEpisodesByDiagnosis . . . . .	57
A.11	findSubEpisodesByEpisodeUIDAndPatientUID . . . . .	57
A.12	findSubEpisodesForParentEpisode . . . . .	58
A.13	SysDiagnosis . . . . .	58
A.14	isRTSAliveAndReady . . . . .	58
A.15	PatientIndex . . . . .	59
A.16	retrievePDbyPDE . . . . .	59
A.17	retrievePDEbyPatientUId . . . . .	59
A.18	retrievePDEbyPatientNId . . . . .	60
A.19	findPatientsByPostalCode . . . . .	60
A.20	findPatientsByNameAndGender . . . . .	61
A.21	findPatientsByNameAndBirthday . . . . .	61





# Acrónimos

<b>AMRIA</b>	Associação de Municípios da Ria de Aveiro
<b>API</b>	Application Programming Interface
<b>AT</b>	Aplicação Terceira
<b>BD</b>	Base de Dados
<b>CA</b>	Credential Authority
<b>caCORE</b>	<a href="#">NCI</a> - Common Ontologic Representation Environment
<b>CDA</b>	Clinical Document Architecture
<b>EHR</b>	Electronic Health Record
<b>epSOS</b>	Smart Open Services for European Patients
<b>HCI</b>	Health Care Institution
<b>HIP</b>	Hospital Infante D. Pedro
<b>HL7</b>	Health Level Seven International
<b>HTTP</b>	Hypertext Transfer Protocol
<b>ICD</b>	International Classification of Diseases
<b>ICPC</b>	International Classification of Primary Care
<b>Java EE</b>	Java Platform, Enterprise Edition
<b>Java SE</b>	Java Platform, Standard Edition
<b>JAXB</b>	Java Architecture for XML Binding
<b>JAX-RPC</b>	Java API for XML-Based RPC
<b>JAX-WS</b>	Java API for XML Web Services
<b>JAX-WS RI</b>	Java API for XML Web Services Reference Implementation
<b>JDK</b>	java Development Kit
<b>MeSH</b>	Medical Subjects Heading

<b>MVC</b>	Model View Controller
<b>NCI</b>	Natioonal Cancer Institute
<b>OMS</b>	Organização Mundial de Saúde
<b>PCE</b>	Processo Clínico Electrónico
<b>RPC</b>	Remote Procedure Call
<b>RSE</b>	Registo de Saúde Electrónico
<b>RTS</b>	Rede Telemática da Saúde
<b>SGBD</b>	Sistema de gestão bases de dados
<b>SMTP</b>	Simple Mail Transfer Protocol
<b>SNOMED</b>	Systematized Nomenclature of Medicine-Clinical Terms
<b>SOA</b>	Service-Oriented Architecture
<b>SOAP</b>	Simple Object Access Protocol
<b>UML</b>	Unified Modeling Language
<b>W3C</b>	World Wide Web Consortium
<b>WS</b>	Web Service
<b>WSIT</b>	Web Services Interoperability Technologies
<b>WSDL</b>	Web Service Definition Language
<b>XML</b>	eXtensible Markup Language

# Capítulo 1

## Introdução

### 1.1 Enquadramento do tema

Os sistemas de informação clínica encontram-se sobre grande desenvolvimento em diferentes áreas do software. A natureza colaborativa da prestação de cuidados médicos entre instituições e pontos de serviço requer a articulação de diferentes sistemas de informação, entre diferentes instituições, surgindo assim, perante este fluxo de dados, alguns requisitos, tais como:

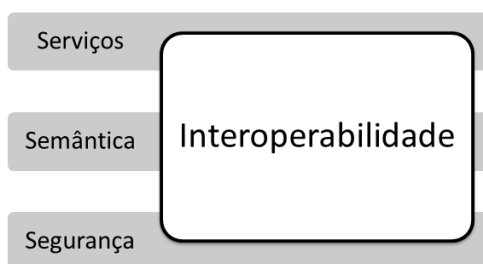
- A informação deve estar acessível remotamente;
- A disponibilidade da informação deve ser independente dos sistemas clientes;
- A informação neste contexto é sensível; a comunicação tem de ser segura;
- A informação deve estar disponível num formato universal, ou pelo menos sobre um ou mais formatos bem definidos;
- A troca de dados deve realizar-se sem ambiguidade quanto ao seu significado.

A base de estudo para a presente dissertação é a Rede Telemática da Saúde (RTS) [16] [2], uma plataforma de centralização e uniformização de dados clínicos. Desenvolvido no contexto do programa AveiroDigital e implementado na mesma região, este sistema ao qual algumas instituições de cuidados de saúde primários e secundários se encontram associadas, permite que estas possam aceder de forma unificada a dados relacionados com os episódios dos utentes. Providencia assim, de forma segura, a disponibilização de informação clínica a profissionais do sector, mediante o seu acesso a um portal- "Portal do Profissional". A RTS proporciona então uma melhoria no sistema de informação entre profissionais de saúde e utentes, contribuindo para o envolvimento do cidadão na gestão da sua saúde.

O Smart Open Services for European Patients (epSOS) [1] é um projeto a nível europeu, único no seu género, para a troca de informação clínica entre diferentes países. Apresenta-se assim como uma referência de forte relevo no que diz respeito a muitas das necessidades de desenvolvimento identificadas. A troca de informação estabelece-se utilizando documentos, estando estes num formato Health Level Seven International (HL7) Clinical Document Architecture (CDA). Decidimos então que apesar de não fazer parte do âmbito principal da Dissertação, o epSOS é uma referência a ter em conta neste trabalho, e que seria de todo pertinente orientar a evolução da RTS no sentido do epSOS.

Este trabalho apresenta um estudo sobre uma abordagem possível para a colmatação das barreiras identificadas, bem como a implementação de uma extensão da **RTS** que o materializa.

O enquadramento tecnológico principal, são os sistemas de informação, aplicados à informação clínica. O projeto **RTS** como base para este trabalho, é a plataforma de informação clínica que servirá de suporte à presente dissertação. O projeto **epSOS** é a referencia no que à interoperabilidade semântica diz respeito, e por conseguinte, é utilizado como referência quanto ao modelo de informação a integrar. Resumem-se na figura 1.1 os conceitos a abordar e na tabela 1.1 apresenta-se uma descrição contextualizada dos mesmos.



**Figura 1.1:** Principais conceitos da Dissertação

**Tabela 1.1:** Conceitos principais em contexto

Conceito	Contexto no projeto
Serviços	Os serviços representam neste trabalho uma camada de software que permite a interação entre aplicação, por forma a tornar a <b>RTS</b> acessível remotamente. Assim, neste contexto, um serviço é a implementação de cada um dos mecanismo de acesso a desenvolver.
Segurança	A segurança neste conceito refere-se a dois princípios de aplicação: o controlo de acesso aos serviços e a garantia de confidencialidade no acesso.
Semântica	Semântica é a capacidade de entendimento mútuo entre dois interlocutores. No cenário desta dissertação, representa a capacidade que a <b>RTS</b> tem de se adaptar e disponibilizar a informação que disponibiliza segundo diferentes normas de representação de dados clínicos.

## 1.2 Objetivos e âmbito da dissertação

### 1.2.1 Objetivo principal

O objetivo principal desta dissertação é o desenvolvimento de novos módulos para a **RTS**, nomeadamente uma camada de Web Services (**WSs**) e um servidor de vocabulário.

A camada de serviços, deve disponibilizar o acesso à informação disponível na **RTS** através

de uma Application Programming Interface (API) de WS expondo um modelo de informação bem definido.

A comunicação estabelecida entre as aplicações-cliente e o servidor deve ser efetuada de forma segura, visto a informação trocada ter um cariz privado.

O servidor de vocabulário permitirá o mapeamento entre diferentes terminologias, possibilitando que a informação disponibilizada seja codificada segundo diferentes sistemas de representação.

### 1.2.2 Objetivos específicos

O objetivo da dissertação pode decompor-se em sub-objetivos mais específicos:

- Desenvolvimento de WSs para a criação de uma API para acesso remoto à RTS;
- Implementação de mecanismos de segurança de forma a garantir que a comunicação providencia privacidade;
- A disponibilização da informação pode ser feita segundo diferentes terminologias, com a finalidade da interoperabilidade.
- Utilização de um servidor de vocabulário para o armazenamento e manutenção das terminologias, permitindo consultas aos conceitos e também às suas relações e mapeamentos.

## 1.3 Estrutura da Dissertação

O presente documento está dividido em seis secções principais, mais duas auxiliares e anexos.

### Secções principais

- **Introdução** - onde se apresenta e contextualiza a temática do projeto, bem como os principais objetivos.
- **Síntese do estado da arte** - onde estão presentes, de forma estruturada por áreas de estudo as opções tecnológicas, as ferramentas e metodologias a aplicar no trabalho.
- **Cenários de integração na RTS** - nesta secção são discutidos alguns cenários de aplicações para a nova API de acesso à RTS.
- **Modelo computacional do sistema** - relata-se a forma como se planeou o projeto; paradigmas e metodologias.
- **Implementação** - descreve-se como se realizou a implementação dos módulos definidos no capítulo anterior.
- **Conclusão** - apresentam-se os resultados, observações e conclusões sobre o trabalho realizado. Nesta secção aborda-se também possíveis trabalhos futuros.



## Capítulo 2

# Síntese do estado da arte

### 2.1 Aplicações distribuídas baseadas em serviços

#### 2.1.1 Conceito

Atualmente é comum uma empresa sentir a necessidade de expôr para os seus clientes, ou apenas para outras partes desta, um conjunto de funcionalidades computacionais, acessíveis remotamente. Esta necessidade acarreta diferentes problemas arquitetônicos que têm por base o acesso o armazenamento e acesso distribuído a informação. No paradigma da computação distribuída esta necessidade recebe uma atenção especial, visto ser um elemento fulcral na sua realização. No entanto, não é apenas na computação distribuída que esta necessidade existe e está presente na arquitetura de todos os sistemas computacionais, que necessitam de disponibilizar informação a terceiros ou de comunicar por qualquer outro motivo. Foram criados vários paradigmas que abordam a operacionalização da comunicação entre máquinas, apresentando na sua maioria uma essência comum - a interação através de serviços.

O conceito de serviço é algo bem definido quando se abstrai a sua aplicação. No entanto a definição de serviço tendo em conta a sua aplicação já é variável, dependendo do contexto onde vai ser aplicado.

Num cenário de comunicação entre máquinas ligadas através da internet, os serviços que efetuam este contacto são chamados de serviços web (Web Services).

Um Web Service (**WS**) neste contexto pode ser definindo como sendo:

- um componente de *software* com uma interface bem definida;
- detentor uma interface simples;
- de preferência, tecnologicamente independente;
- uma abstração dos mecanismos internos;
- um ponto de acesso para a restante instituição;
- acessível remotamente.

#### 2.1.2 Service-Oriented Architecture

A comunicação entre aplicações a correr em diferentes máquinas é uma necessidade antiga. Para se conseguir a interligação entre diferentes sistemas é necessária uma forma metódica de

o fazer. Uma dessas formas é a Service-Oriented Architecture (SOA).

”SOA is a kind of architecture that uses services as building blocks to facilitate enterprise integration and component reuse through loose coupling.” (E.Hewitt) [14]

Com o objetivo de facilitar a conexão cooperativa entre aplicações colocadas em diferentes máquinas, é um modelo completo de programação com especificação de normas e tecnologias. Os seus constituintes, em oposição a subsistemas ou componentes, são serviços.

SOA é uma arquitetura na medida em que propõe a implementação de um estilo e de um conjunto de práticas a implementar-se nos componentes de um sistema. Tal como o nome indica, tais práticas são definidas para serem aplicadas a serviços, definindo a forma como estes devem ser incorporados no modelo computacional.

É importante realçar que um conjunto de serviços *per se* não implica a aplicação de da arquitetura SOA. A funcionalidade dos serviços, a forma como são expostos e a sua orquestração, é algo específico a cada modelo de negócio, mas tem de constituir sempre a implementação das boas práticas definidas por esta arquitetura.

A frequência com que surgem sistemas onde a elaboração dos web serviços é feita de uma forma menos cuidada (ou não cuidada) dá origem a serviços que se apresentam de uma forma desconexa, de difícil utilização, e frequentemente de fraca eficiência. Tais serviços não se podem caracterizar uma SOA. Este tipo de abordagem é tão recorrente que foram detetados alguns padrões de má utilização. Estes padrões são conhecidos como anti-padrões, e os mais conhecidos são: Just a Bunch of Web Services (JaBoWS) e Rogue Service. JaBoWS define um conjunto de serviços, desconexos que não constituindo uma solução de negócio também não fazem parte de uma SOA. Rogue Service descreve um serviço marginal, um serviço que não é anunciado, e inútil no sistema onde foi definido.

Por outro lado, a utilidade de um serviço quando este é parte integrante numa SOA, não deve ser limitada a um modelo de negócio, ou a uma função demasiado específica no contexto em que foi desenvolvido, sem qualquer relevância na restante infra-estrutura da empresa. Um dos princípios base nesta arquitetura, é a reusabilidade dos seus componentes, devendo estes ser planeados tendo em conta possíveis variações no modelo de negócio, não abrangendo uma funcionalidade demasiado ampla, ou restritiva. A reusabilidade é uma característica chave neste domínio e talvez a mais importante.

A facilidade com que um serviço pode ser reutilizável está diretamente relacionada com a dependência com que este foi concebido, relativamente ao sistema onde se insere. A dependência entre componentes computacionais é chamada de *Coupling*, e é definida como o nível de conhecimento que um cliente de um serviço tem sobre este. No caso dos WSs e apesar de disponibilizarem uma interface bem definida, mais nenhuma informação tem de ser fornecida ao cliente, o que tem evidentes vantagens, no que diz respeito à dependência na implementação.

### 2.1.3 Simple Object Access Protocol

Os WSs constituem um paradigma da comunicação que define um mecanismo de interoperabilidade entre aplicações. No entanto, este conceito é independente da tecnologia aplicada na sua implementação. Uma destas tecnologias é o Simple Object Access Protocol (SOAP); um protocolo que descreve uma norma para a troca de mensagens em WSs. Este protocolo é atualmente a recomendação World Wide Web Consortium (W3C) para a implementação de

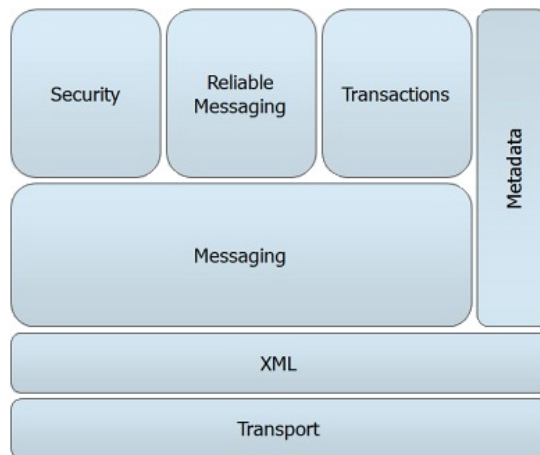


WSs.

*(A Web Service is) "... a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format... Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."* (W3C)

Para assegurar a independência relativamente às plataformas, tecnologias e linguagens de programação subjacentes às aplicações envolvidas, este protocolo baseia-se na definição de uma mensagens em eXtensible Markup Language (XML). [7] Pelo facto de estar largamente espalhado na programação para a internet e dispor também de uma enorme variedade de ferramentas para a sua manipulação, o XML é uma mais valia para o SOAP, transformando-o num protocolo simples e altamente expansível. A estrutura base de uma mensagem é chamada de *envelop* (envelope), contendo este um *header* (cabeçalho) e um *body* (corpo). Cada um destes componentes é mapeado num elemento em XML, tal como o são os seus conteúdos. Contudo, o uso de XML acarreta algumas desvantagens, devido ao formato estas mensagens podem tornar-se pesadas para a rede, e dispendiosas em recursos na sua manipulação.

Varias empresas que desenvolvem tecnologias relacionadas com WS têm vindo a criar especificações quanto ao seu uso de forma a poderem garantir interoperabilidade com tecnologias de outras empresas que proporcionem as mesmas funcionalidades. O nome da maioria destas normas começa por "WS-" o que de origem à abreviatura "WS-\*" como referencia ao conjunto de normas criadas para os WS. Estas normas estão apresentadas na figura 2.1.



**Figura 2.1:** Principais categorias das normas WS-\*

## Transporte

Esta categoria diz respeito aos protocolos de transporte que podem ser usados para a troca de mensagens SOAP, havendo para tal uma especificação (SOAP Binding Framework). [12] Esta especificação define como qualquer protocolo pode ser utilizado para transportar as mensagens. Com suporte oficial documentado estão definidos para transporte de mensagens SOAP o protocolo Simple Mail Transfer Protocol (SMTP) e o Hypertext Transfer Protocol

(HTTP). O primeiro é usado em situações de necessidades bastante específicas, enquanto o segundo é usado na grande maioria das implementações.

## XML

Normas tais como a própria linguagem XML ou XML Schema são a base da manipulação de dados no formato XML, tornando-se portanto também a base do próprio SOAP. Outras especificações relacionadas com a utilização de XML ganham relevo noutras categorias do SOAP, de se salientar XML Digital Signature e XML Encryption [11], que foram repescadas para a implementação de segurança em SOAP, fazendo assim uso das ferramentas já existentes.

## Segurança

A segurança é uma das necessidades básicas na comunicação, por forma a garantir privacidade e integridade dos dados trocados. A segurança foi desde cedo uma categoria de desenvolvimento ativo quanto às especificações da sua implementação em mensagens SOAP, o que levou à elaboração de diversas especificações. Estas serão abordadas com algum detalhe na secção 2.2.2.

## Metadados

O recurso a WS pressupõe que o cliente apenas necessita de conhecer a interface de acesso ao serviço e o modelo de dados a ele adjacente. No entanto, nada é definido quanto à forma a informação é codificada e transportada até ao cliente. Segundo a especificação do protocolo SOAP um serviço pode ser anunciado fazendo-se uso da linguagem XML para criar um documento onde se descreve um serviço. A esta descrição de um serviço deu-se o nome de Web Service Definition Language (WSDL). [13] Nele é apresentada uma descrição da interface do serviço, definindo, entre outras coisas, o seu nome e o seu endereço. Apresenta também as estruturas de dados necessárias à sua invocação e respetivo retorno.

### 2.1.4 Simple Object Access Protocol em Java

O suporte para a invocação de serviços remotos em Java surgiu primariamente na implementação empresarial, Java Platform, Enterprise Edition (Java EE) na sua versão 1.4, com o nome de: invocação remota de procedimentos - Remote Procedure Call (RPC). Esta tecnologia estava sobre a alçada da ferramenta Java API for XML-Based RPC (JAX-RPC), que definia um mecanismo para a invocação de métodos remotamente, fazendo uso de mensagens em XML. Na versão seguinte do Java (Java EE 1.5), a definição do uso desta tecnologia evoluiu e tornou-se a primeira implementação do conceito de WS. Esta mudança de paradigma levou à criação de uma nova Application Programming Interface (API) de comunicação e por conseguinte a um novo nome: Java API for XML Web Services (JAX-WS).

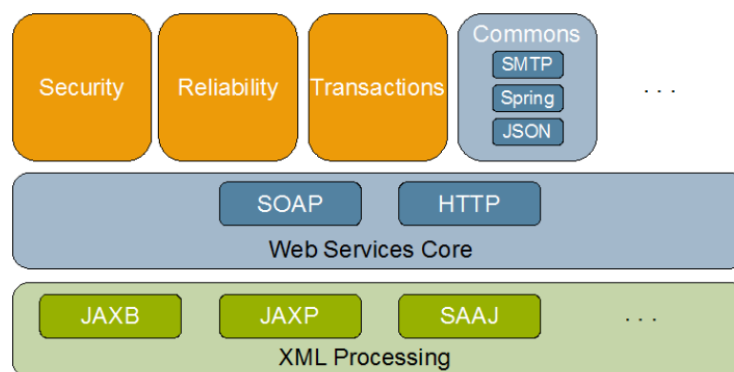
A utilização de WSs em Java deixou de fazer parte apenas do ambiente de desenvolvimento empresarial, e passa a fazer parte da implementação comum do java - Java Platform, Standard Edition (Java SE) - na sua versão 1.6. Este lançamento veio não só liberalizar a aplicação desta tecnologia, como também trazer a sua nova versão: 2.0. [3]

**JAX-WS** é a definição de uma **API** para a utilização de **WS** mas não disponibiliza a sua implementação. Para desenvolver aplicação que façam uso de **WS** é necessário complementar o uso das bibliotecas Java, com uma que disponibilize esta implementação.

Ao longo do tempo foram surgindo várias plataformas de suporte a **WSs**, sendo algumas implementações da **API** definida pelo Java, mas não todas. Como exemplos de implementações de **JAX-WS** são de salientar Apache CXF e Metro. Por outro lado, Spring WebServices é um exemplo de um plataforma que não segue esta interface normalizada. É também de referir que como parte do java Development Kit (**JDK**) 1.6 é disponibilizada a plataforma Java API for XML Web Services Reference Implementation (**JAX-WS RI**) que é a parte do metro responsável pela implementação estrita de **JAX-WS**.

## Metro

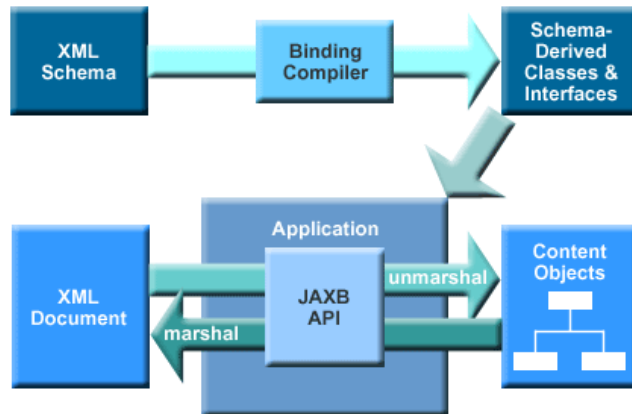
Metro é uma plataforma para a implementação de **WS** que vai mais além que a simples implementação de **JAX-WS**. Esta plataforma é composta pela aplicação de diferentes tecnologias e outras plataformas, para permitir a implementação de **WS** e mecanismos adicionais como segurança e interoperabilidade. Esta arquitetura está demonstrada na figura 2.2.



**Figura 2.2:** Arquitetura Metro

[http://weblogs.java.net/blog/arungupta/archive/2007/06/announcing\\_metr.html](http://weblogs.java.net/blog/arungupta/archive/2007/06/announcing_metr.html)

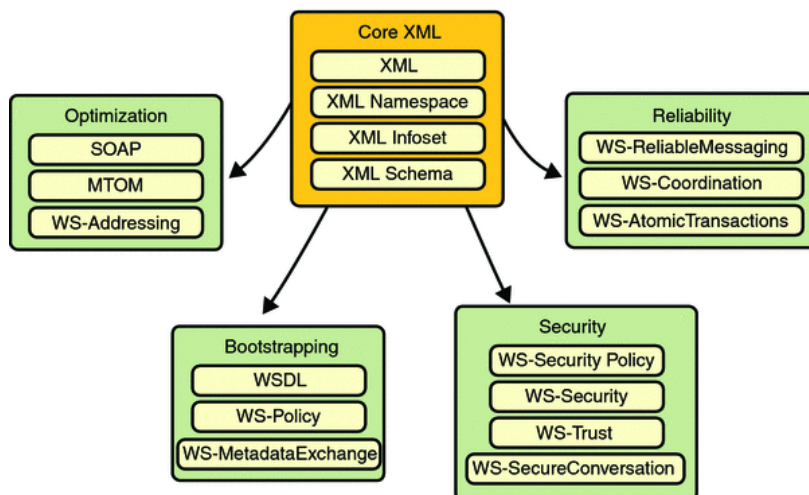
Para a manipulação de **XML** no Metro recorreu-se à ferramenta Java Architecture for XML Binding (**JAXB**), a sua finalidade é a manipulação automatizada de **XML** em ambiente Java. A forma base de atuação com **JAXB** é a transformação de dados em formato **XML** em objetos java (*marshal*) e *vice versa* (*unmarshal*). O processo de gerar classes partindo de um **XML Schema** é também assegurado por esta ferramenta, que para além de declarar as classes ainda as vai dotar das anotações necessárias para transformar o seu conteúdo de novo em **XML**. [4] Este mecanismo encontra-se exemplificado na figura 2.3.



**Figura 2.3:** Mecanismo de atuação do JAXB

<http://www.oracle.com/technetwork/articles/javase/index-140168.html>

Para além de disponibilizar uma plataforma para a implementação de **WS**, o projeto Metro é também constituído por mecanismos que garantem a interoperabilidade entre **WSs**, sendo esta parte do projeto conhecida por Web Services Interoperability Technologies (**WSIT**). Durante anos, tem havido contacto entre empresas que desenvolvem tecnologias com o objetivo de proporcionar interoperabilidade entre as suas implementações. Neste sentido, a Sun e a Microsoft iniciaram há alguns anos uma parceria com a finalidade de criar normas que garantam interoperabilidade entre as suas tecnologias para **WSs** principalmente as referentes a segurança, fiabilidade da comunicação e criação de transações atómicas. Os vários componentes do **WSIT** estão representados na figura- 2.4.



**Figura 2.4:** Modelo WSIT

[http://metro.java.net/guide/What\\_is\\_WSIT\\_.html](http://metro.java.net/guide/What_is_WSIT_.html)

## 2.2 Segurança em Web Service

### 2.2.1 Controlo de acesso

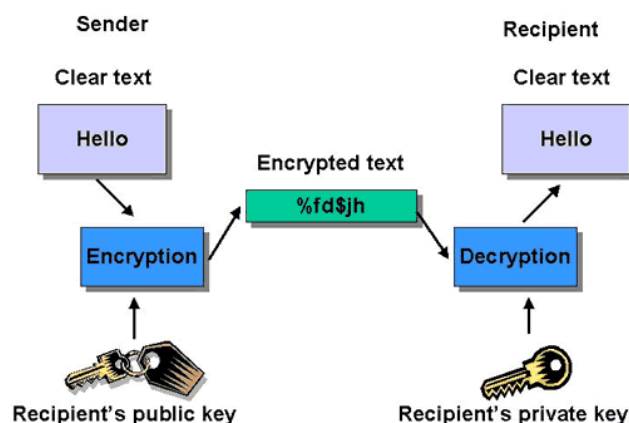
É fundamental que a comunicação inter-aplicação seja feita de forma segura, para que se possa garantir a integridade dos dados trocados e também a sua autoria. A comunicação segura pressupõe que sejam asseguradas várias características, entre elas:

- Autenticação do cliente;
- Autenticação do servidor;
- Comunicação cifrada;
- Mensagens (ou pelo menos partes) assinadas.

Por forma a garantir que uma mensagem é trocada entre dois interlocutores, de forma impercetível a terceiros, é necessário que esta seja cifrada pelo seu emissor, e posteriormente decifrado pelo receptor. Isto implica que ambos partilhem um segredo – a forma de cifrar e decifrar o texto. Levanta-se um problema de troca deste segredo quando um o define e necessita de o partilhar com o outro. A solução para esta limitação de partilha de segredo, é a utilização de duas chaves distintas para o efeito.

A arquitetura PKI (criptografia de chave pública), apresenta um mecanismo de cifra que, permite a utilização de duas chaves distintas para cifrar e posterior decifrar um texto. Segundo este mecanismo, cada entidade está munida de um par de chaves, a que se chama chaves assimétricas. Este é constituído por uma chave pública e uma privada. A relação matemática existente entre um par de chaves assimétricas, garante que quando uma é usada para cifrar um texto, apenas o seu par o pode decifrar. [20]

Num cenário de utilização de chaves assimétricas, o servidor possui um par de chaves (i.e. uma chave privada e uma pública), podendo o cliente ter também o seu próprio par de chaves. Quando um deles pretende enviar uma mensagem esta é cifrada com a chave pública do receptor. Este irá aquando da recepção decifrá-la usando a sua chave privada, tal como está representado na figura 2.5.



**Figura 2.5:** Utilização de chaves assimétricas

[www.crypticbox.com/images/pgp\\_encryption.gif](http://www.crypticbox.com/images/pgp_encryption.gif)

De forma a garantir que o servidor possui a chave pública correta de um determinado cliente, está é-lhe fornecida dentro de um certificado. Um certificado é um ficheiro que autentica uma chave e contém também alguma informação sobre o dono desta. Assim, na posse de tal informação é sempre possível saber-se a que entidade pertence uma determinada chave. Um certificado é sempre assinado digitalmente pela entidade certificadora e basta que o servidor confie na entidade certificadora para passar a confiar em todas as entidades por ela certificadas.

## 2.2.2 Segurança na comunicação

As normas relacionadas com segurança foram das primeiras a surgir, pela necessidade dos fabricantes de interoperabilidade na forma como as credenciais dos utilizadores eram codificadas e enviadas nas mensagens. Posteriormente a atenção virou-se para cenários típicos, como por exemplo a segurança das mensagens ou a identificação federativa de utilizadores.

Para se implementar segurança na comunicação é necessário que as mensagens sejam cifradas e assinadas. Quando uma mensagem é cifrada o seu conteúdo fica seguro garantindo-se assim privacidade na comunicação, para além disso deve ser assinada para se garantir a autoria e também a integridade da mesma.

As questões de privacidade, autoria e integridade levaram à criação da norma WS-Security [18], um formato aberto para a assinatura e cifra de partes de mensagens em XML. Esta norma faz uso de outras tais como XML Digital Signature e XML Encryption protocols, para criar credenciais nas mensagens sob a forma de *tokens* seguros. Entre estes constam: par username-password, credenciais kerberos ou um *token* de identidade SAML. Entre dos diferentes cenários atualmente implementados, os mais recorrentes são:

- UserName token over transport security;
- UserName token over message security;
- Mutual certificate authentication;
- SAML token authentication.

### UserName token over transport security

No passado a segurança na comunicação sobre SOAP aplicava-se fazendo uso de protocolos como o SSL. Este protocolo define a implementação de um canal de comunicação seguro ponto-a-ponto, no qual toda a informação é cifrada à saída da máquina e é decifrada quando chega à máquina destino. Esta forma de realizar segurança tem algumas limitações, principalmente por a segurança ser apenas ponto-a-ponto, ou seja entre duas máquinas e não entre a máquina origem e a destino.

Esta é a forma mais popular de implementar segurança em WS, pois para a maioria daqueles que desenvolvem software ainda é uma forma suficientemente boa. Neste cenário a segurança é imposta ao nível da camada de transporte e as credenciais de acesso ao serviço são transportadas num *token* com um par username/password.

## **UserName token over message security**

Deve favorecer-se a implementação da segurança ao nível da mensagem em detrimento da segurança apenas ao nível do canal de comunicação. Ao garantir-se que é a mensagem que é segura (e não apenas a conexão usada) garante-se que esta chega intacta e é apenas perceptível para a aplicação a que é destinada. Estando a mensagem cifrada até chegar à aplicação, impossibilita-se que a mensagem seja interceptada em trânsito, entre a recepção no servidor e a chegada à aplicação, ou mesmo entre uma máquina *proxy* e o verdadeiro servidor. Este nível mais alto na segurança deve ser implementado sempre que possível, garantindo-se assim uma menor suscetibilidade a ataques de escuta de conversas privadas (*eavesdropping*).

Este cenário é então aplicado em situações onde a segurança é colocada ao nível da mensagem, e as credenciais de acesso ao serviço são transportadas num *token* com um par *username/password*.

## **Mutual certificate authentication**

No cenário anterior as credenciais identificam um utilizador do sistema. Contudo, em comunicações entre sistemas no panorama empresarial, é muitas vezes necessário identificar o sistema cliente ao invés de um utilizador específico. Isto permite restrição de acesso apenas a aplicações registadas e credenciadas.

Este mecanismo de segurança é então aplicado ao nível da mensagem e faz uso de um certificado X.509 para identificação do cliente. [15]

## **2.3 Interoperabilidade semântica em saúde**

### **2.3.1 Interoperabilidade semântica**

Quando se registam ou armazenam dados os campos são tipicamente de um tipo ou estrutura bem definida, tais como números ou datas, no entanto estes também podem ser de texto "livre". Quando um valor numérico é lido (não entrando em questões sobre métodos de codificação ou representação) numa primeira impressão parece não haver dúvidas quanto ao seu significado, e o mesmo acontece com uma data. Na realidade isto não é verdade, não há dúvidas quanto ao valor que representam, porém, o seu valor não transporta o significado. Este carece de um contexto, e mesmo assim, diferentes utilizadores podem ter diferentes interpretações.

Na interpretação de um texto podem surgir problemas de:

- Léxico;
- Sintaxe;
- Semântica.

Uma língua define-se por um conjunto de palavras e uma gramática que define o uso correto das mesmas. O conjunto das palavras chama-se o léxico, a forma como estas podem ser organizadas para criar frases válidas é a sintaxe, e o significado de cada uma delas (ou de um conjunto, i.e. expressão) é a semântica.

Quando um texto é escrito numa determinada língua, é impossível para alguém que não a conheça, interpretar o texto. Isto porque o segundo não compreende as palavras - léxico.

No caso de ambos os interlocutores falarem a mesma língua se o emissor não souber utilizar de forma correta as palavras, as frases que este criar não serão válidas para o recetor - sintaxe. Numa situação em que dois interlocutores falam a mesma língua e ambos sabem criar frases de forma correta, se um deles utilizar palavras, com um significado, que o outro desconhece a comunicação fica inviabilizada - semântica. É importante também salientar que a comunicação pode falhar se os interlocutores partilham contextos diferentes (e.g. culturais).

Uma frase, apesar de gramaticalmente correta, pode ter interpretações distintas, para diferentes pessoas. A expressão "vou já!" pode significar a adesão a um pedido, ou justamente o contrário, conforme o contexto em que é dita (num sentido literal ou num sentido irónico). Pode ainda ser ainda mais difícil para um falante de português não nativo perceber o sentido idiomático desta expressão.

No registo de informação surge ainda uma dificuldade adicional. Por exemplo, se alguém escrever uma palavra com todas as suas letras em minúsculas e depois a escrever novamente com pelo menos uma das letras em maiúscula, as duas palavras não são iguais; a isto chama-se um problema de redundância.

No contexto clínico, não se pode assumir que todos os profissionais de saúde falem a mesma língua, mas fazendo-o, podem ainda existir problemas semânticos ou por redundância. Para evitar que tal aconteça, foram desenvolvidos sistemas de codificação de conceitos. Ao conjunto de pares código/conceito dá-se o nome de terminologia. A primeira utilização de um sistema neste formato remonta ao séc. XIX, quando a primeira terminologia clínica foi criada para representar causas de morte (International List of Causes of Death), tendo dado mais tarde origem ao sistema International Classification of Diseases (ICD).

O registo de informação clínica num formato de texto "livre" padece dos problemas referidos. Os sistemas de registo de informação clínica têm de persistir os dados utilizando codificação, através de terminologias, de todos os campos que possam sofrer de tais problemas; e. g. identificação de doenças, alergias, procedimentos clínicos, etc.

Uma terminologia no contexto da representação de conhecimento é uma especificação formal de uma conceptualização. Em Bioinformática, é uma linguagem comum, que representa um conjunto de entidades biomédicas, os seus termos e as suas relações, normalmente na área da taxionomia. As ontologias agilizam a representação, troca e compreensão da informação, permitindo:

- Uma estrutura comum para a informação, entre todos os interlocutores.
- Reusabilidade de conceitos num domínio de conhecimento.
- Simplificação na atualização do conhecimento, através da explicitação de suposições de informação.
- Separação do conhecimento de domínio, do conhecimento operacional.

Para que um sistema possa armazenar dados numa determinada terminologia, mas depois disponibiliza-los para o exterior numa outra, precisa de conhecê-las e de saber relacioná-las. Conhecê-las, significa ter armazenadas de alguma forma no sistema todas as terminologias a usar. Relacioná-las, implica conhecer o significado dos conceitos e de saber interligá-los.

As terminologias resolvem o problema de redundância, mas o problema da troca de dados mantém-se. Diferentes sistemas de informação usam diferentes sistemas de codificação, o que impossibilita a interoperabilidade entre estes. Para se possibilitar as relações entre aplicações,



é necessário que um dos sistemas compreenda as terminologias usadas pelo outro. Surgem assim sistemas de representação de estruturas de dados clínicos, que definem não só a sintaxe destas, como também quais as terminologias a adotar para cada campo.

### 2.3.2 Semântica no contexto da informação clínica

Foram desenvolvidas terminologias nas várias áreas do conhecimento, muitas vezes com características específicas dependentes da sua finalidade.

No contexto da informação clínica entre as várias terminologias existentes, são de salientar pela sua presença alargada nos registos eletrónicos de saúde, as seguintes:

- Systematized Nomenclature of Medicine-Clinical Terms (**SNOMED**) - é um sistema de classificação de dados médicos, publicada pelo International Health Terminology Standards Development Organization, uma organização sem fins lucrativos, criada para o desenvolvimento desta terminologia. Contem conceitos sobre: sinais e sintomas, diagnósticos e procedimentos. Pretende a integração completa de toda a informação do registo médico eletrónico numa única estrutura de dados.
- International Classification of Diseases (**ICD**) - é utilizada na classificação de doenças e outros problemas de saúde. Criada no séc. XIX, foi adotada pelo Instituto Internacional de Estatística em 1893, tendo passado a responsabilidade da Organização Mundial de Saúde (**OMS**) em 1948, aquando da sua 6ª edição (**ICD-6**). Atualmente está na versão 11 e é a terminologia estipulada pela **OMS** para a representação de causas de morte e morbidez.
- LOINC - foi criada para colmatar uma necessidade de normalização na identificação de observações clínicas e laboratoriais. Está sob a alçada do Instituto Regenstrief, uma organização internacional de investigação sem fins lucrativos.
- International Classification of Primary Care (**ICPC**) - encontra-se atualmente na versão número 2, é também uma terminologia da responsabilidade da **OMS**. Esta terminologia classifica dados dos utentes e de atividade médica ao nível dos cuidados de saúde primários, isto é, medicina geral e/ou médico de família.

No que diz respeito à normalização das estruturas de dados para a representação de informação, foram criados vários padrões de representação de dados médicos. Entre estes, os mais influentes são:

- Health Level Seven International (**HL7**) Clinical Document Architecture (**CDA**) - é um modelo para representação formal de informação clínica. Tal modelo inclui por exemplo observações médicas ou administração medicamentosa. Na sua segunda versão desta arquitetura, foram associadas ao modelo de dados do **HL7** ontologias tais como **SNOMED** ou RxNorm.
- OpenEHR - apresenta um conjunto de arquétipos que definem um modelo de dados normalizado para a troca de informação clínica. Estas normas são de especificação aberta.



## Capítulo 3

# Cenários de integração na Rede Telemática da Saúde

### 3.1 A **RTS** como um repositório regional acessível por serviços

Após a apresentação feita no capítulo introdutório sobre a Rede Telemática da Saúde (**RTS**), é importante expor agora uma descrição com mais detalhe sobre a plataforma.

A **RTS** foi desenvolvida e implementada por um consórcio com o mesmo nome, que integra o Hospital Infante D. Pedro, o Hospital Distrital de Águeda, a Sub-Região de Saúde de Aveiro e a Universidade de Aveiro - este último como parceiro tecnológico; no âmbito de um projeto parcialmente financiado pelo Programa Aveiro Digital que teve início em 2003 e decorreu até 2006. O principal objetivo é facilitar a troca de informação clínica entre as instituições parceiras. Tal objetivo tem como finalidade primária, a disponibilização do acesso à informação clínica e promover a comunicação entre os profissionais de saúde da Rede. Na figura 3.1 pode ver-se num mapa o conjunto de parceiros da **RTS**.

Os principais utilizadores da rede são os profissionais de saúde e foi criado para eles um portal web, chamado "Portal dos Profissionais", sendo este a interface entre a **RTS** e as instituições prestadoras de serviços clínicos. O portal dá suporte à colaboração entre os profissionais de saúde, através de um sistema de troca de mensagens construído como parte da plataforma. O portal disponibiliza ainda, um conjunto mínimo de informação clínica sobre o utente, permanentemente atualizado e com integridade garantida.

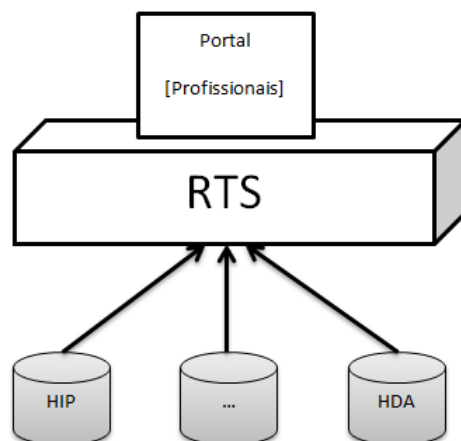
A comunicação entre os pontos intervenientes na rede, devido ao cariz sensível da informação trocada, efetua-se de forma segura. Para tal, criou-se um canal de comunicação seguro entre as fontes de informação clínica e a plataforma agregadora de informação. De igual forma, o profissional de saúde necessita de ter credenciais específicas para poder fazer uso do portal.



**Figura 3.1:** Representação geográfica dos parceiros da [RTS](#)

A arquitetura da [RTS](#), ilustrada na figura 3.2, foi baseada no desenvolvimento de um Processo Clínico Electrónico ([PCE](#)). O processo eletrónico agrega informação clínica do utente, proveniente de várias entidades clínicas da Rede que funcionam como fontes de informação. O Hospital Infante D. Pedro ([HIP](#)), para além de parceiro responsável deste projeto, é a maior instituição de saúde da Associação de Municípios da Ria de Aveiro ([AMRIA](#)), e conseqüentemente será o principal fornecedor de dados para a geração dos processos clínicos. Entre outras, a [RTS](#) disponibiliza acesso a:

- Cartas de alta de episódios;
- Boletins de análises clínicas;
- Relatórios de exames de imagiologia.



**Figura 3.2:** Arquitetura da [RTS](#)

A plataforma [RTS](#) não se limita a agregar a informação, e a disponibilizá-la num ponto de acesso centralizado. O principal pilar da rede é a capacidade que a [RTS](#) tem de uniformizar os dados recolhidos nas fontes clínicas, para os apresentar no portal sempre num formato normalizado. Esta capacidade é a base para impulsionar a interoperabilidade, permitindo que os profissionais clínicos, acedam a informação que se encontra sempre num formato conhecido, mesmo que esse não seja o seu formato original.

Qualquer pessoa que se tenha cruzado com a [RTS](#), sabe que esta plataforma aumenta de forma significativa a relação interinstitucional, bem como a facilidade e qualidade com que um profissional de saúde pode atender um utente. A plataforma tem sido objeto de melhoria contínua através de colaborações com a Universidade de Aveiro que, envolvendo os seus alunos, tem contribuído no plano técnico para o enriquecimento da solução. Com o intuito de criar mais uma melhoria, o consórcio de parceiros responsável pelo projeto, decidiu que seria de todo oportuno expandir a plataforma, criando as condições necessárias para o acesso remoto a esta.

Graças à abstração criada sobre a informação da região, um novo paradigma de utilização pode ser aplicado aos dados clínicos. Sendo a [RTS](#) a camada de abstração, se esta disponibilizar serviços que permitam a consulta dos dados (para além do uso do portal), a informação clínica da região estaria centralizada, uniformizada, e acessível a partir de qualquer sistema computacional. Isto permitiria criar novas funcionalidades, como por exemplo:

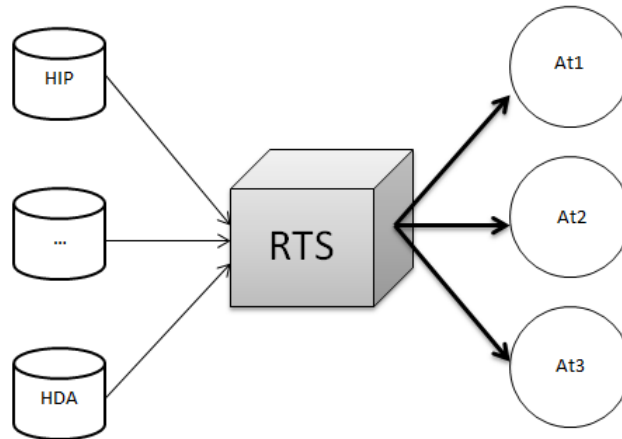
- um catálogo dos profissionais de saúde da região;
- referenciação de utentes;
- extração de dados para suporte a investigação;
- disponibilizar informação de interesse para as entidades reguladoras.

### 3.1.1 Cenários possíveis

Ao disponibilizar-se a informação em serviços, em alternativa ao acesso com recurso a um *browser*, esta passa a estar acessível a partir de novas plataformas. Isto permitiria que, não

só pessoas, mas também sistemas autónomos (e automáticos) pudessem consultar os dados clínicos. Surgindo assim novos cenários de utilização para a nova arquitetura da [RTS](#).

### Cenário 1: Disponibilizar informação para Aplicação Terceira



**Figura 3.3:** A [RTS](#) como fonte de dados para [ATs](#)

Um dos cenários mais desejáveis entre aqueles que podem surgir é a disponibilização de uma *Application Programming Interface* (*API*) para acesso remoto por parte de *Aplicações Terceiras* (*ATs*). Servindo a *RTS* como uma camada de abstração sobre os vários sistemas de informação clínica da região, qualquer aplicação pode assim através da *RTS* aceder de forma completamente transparente à informação clínica da região, sem ter que conhecer a forma de operar da *RTS*, mas apenas a *API* de acesso, e inevitavelmente o esquema de segurança e modelo de dados utilizados.

Para que uma qualquer *AT* se possa ligar à *RTS* é necessário que esta última esteja acessível a contactos externos. Neste cenário achamos que a melhor abordagem seria o uso de *Web Services* (*WSs*). [5]

## Cenário 2: Modelo federativo

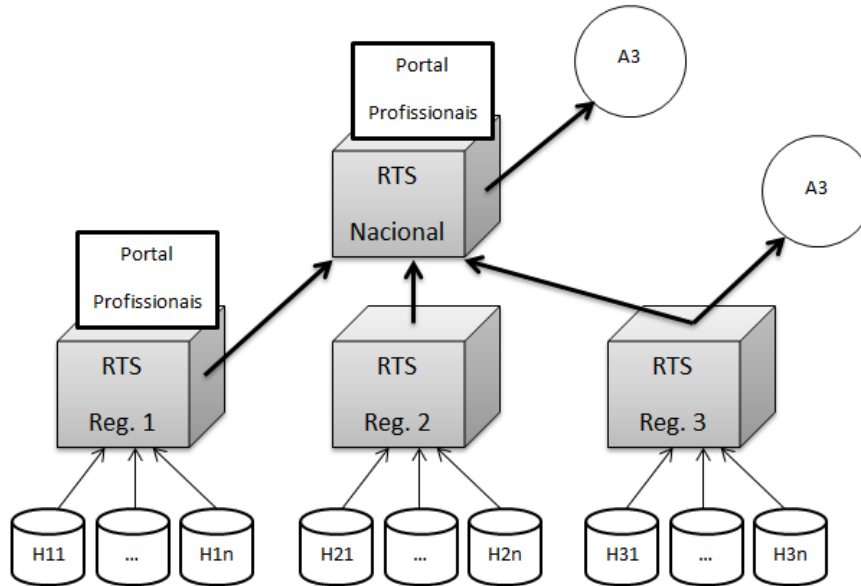


Figura 3.4: Cenário com federação de RTSs

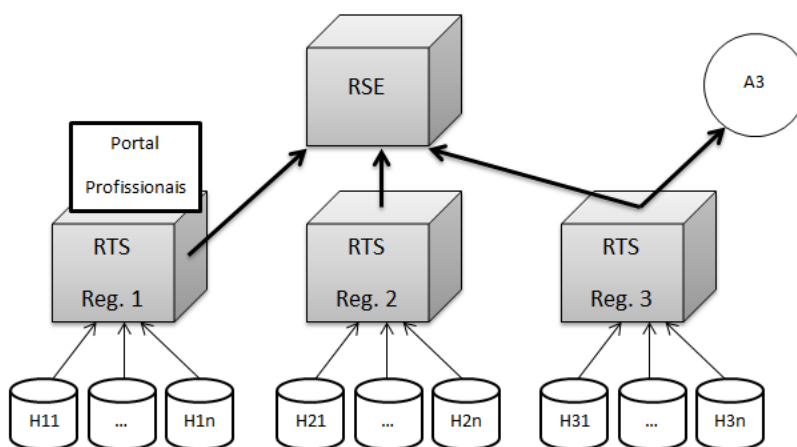
A RTS foi criada como uma rede regional, pois disponibiliza de forma transparente informação clínica que recolhe das instituições de cuidados de saúde da região. Assim, poderia fazer sentido a instalação de uma plataforma RTS em cada região do país, o que seria bastante benéfico, pois levaria as vantagens deste sistema a todo o país. No entanto seria ainda melhor se a informação de todas essas novas instâncias estivesse disponível para os profissionais de saúde das restantes regiões, num único ponto central e não num portal para cada instância. Isto significaria a disponibilização dos dados dos utentes a nível nacional, num único "Portal dos Profissionais", onde a informação das várias regiões era centralizada.

Através da criação de um modelo federativo, seria possível criar-se uma arquitetura piramidal de instâncias de RTSs. Numa arquitetura deste género, um sistema poderia ter como fonte de informação, para além das instituições de cuidados de saúde, outra sistema RTS.

Para se criar um "Portal dos Profissionais" nacional uma das opções seria instalar uma RTS, que tivesse como fonte de dados todas as instâncias nacionais, sendo estas alimentadas pelas instituições da respetiva região.

Este cenário implica a implementação do cenário 1, pois implica também, que a RTS seja dotada de uma API de RTSs para que possa contactar com outras instâncias. Para além desta alteração, é necessário ainda efetuar todas as alterações necessárias para que a RTS possa usar a sua API como uma fonte de informação clínica, tal como faz uso das instituições que já estão a ser utilizadas.

### Cenário 3: Integração com o RSE



**Figura 3.5:** Várias **RTS**s podem ser as fontes para o **RSE**

Em vários países de União Europeia existem sistemas de informação nacionais baseados nos dados clínicos dos seus cidadãos. Por orientações estratégicas da Comissão Europeia, pelas políticas do Governo Português e pelas entidades envolvidas na prestação de cuidados de saúde, surge o interesse da implementação em Portugal de um registo da informação clínica nacional. No despacho do Secretário de Estado da Saúde, publicado no Diário da República (i.e. n.º 10 864/2009, de 20/03/2009), redigiu-se a necessidade de que "a informação clínica de um cidadão esteja ao dispor do próprio e do profissional de saúde que lhe presta um qualquer serviço, de modo adequado mas independente do momento e do local de prestação" [9] Refere-se também no mesmo despacho que a criação de um registo de saúde eletrónico aumentaria a qualidade do serviço prestado aos utentes, bem como a sua celeridade.

Foi criado um grupo de trabalho com o desígnio de estudar a implementação de tal sistema. Este, desenvolveu ao longo dos anos de 2009 e 2010 alguns documentos referentes ao tema, como por exemplo, o estado de arte, a especificação funcional do sistema ou o plano de operacionalização. O serviço foi planeado em torno do cidadão como utente, na ótica da disponibilização da sua informação clínica em suporte digital.

A planificação do que seria o Registo de Saúde Electrónico (**RSE**) [10] foi baseada na ideia de que este deveria cumprir os desígnios de:

- Partilha de informação clínica, orientada para os profissionais de saúde e apoiando-os no cumprimento da sua missão;
- Acompanhamento virtual do cidadão, na sua mobilidade espaço-temporal, materializando-se sempre que o seu acesso é requerido num dado ponto.

Em traços muito gerais o **RSE** tem vindo a ser planeado como um sistema de informação de âmbito nacional, para dar resposta à necessidade incontornável da disponibilização em tempo real da informação de saúde dos utentes, independentemente da sua origem ou da localização dos mesmos.

A recolha de dados de várias (e distintas) fontes, levanta um conjunto de problemas já bem conhecidos. A receção de informação originária de múltiplos sistemas impõe à priori

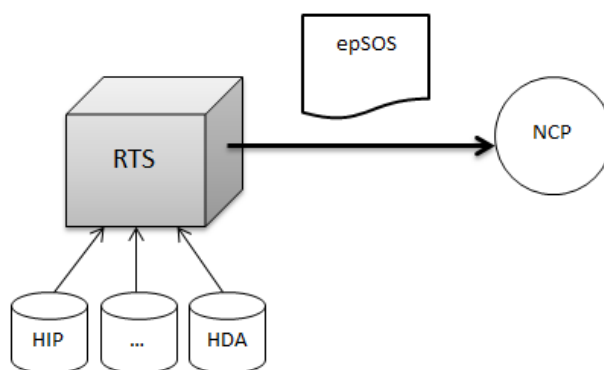


um problema de escalabilidade. Está também sempre presente a problemática sintática e semântica na troca de informação. É na resposta a esta categoria de problemas em arquiteturas distribuídas que a **RTS** se apresenta como uma forma viável de diminuir a complexidade do modelo. A **RTS** apresenta-se como uma plataforma de consulta de informação clínica da região de forma transparente relativamente às entidades providenciadoras dos dados. Surge como um sistema que permite uma abstração à informação regional, resumindo os vários sistemas possuidores de dados, a um único ponto de acesso.

Num cenário em que o **RSE** esteja efectivamente a operar, recebendo informação das várias entidades prestadoras de serviços de saúde, a **RTS** poderia ser uma mais valia nesta arquitetura. Apresentando-se como um ponto único de acesso à região, permite diminuir consideravelmente o número de fontes de informação para o registo nacional. Se se replicar a rede telemática nas restantes regiões, fazendo-se a instalação de uma **RTS** em cada região do país, e fossem estas representantes da informação clínica das suas regiões, então a quantidade de fontes para o **RSE** seria extraordinariamente reduzida.

A dificuldade sintática e semântica na aquisição de informação, poderia ser diminuída fazendo-se uso da **RTS** como um adaptador regional. Assim a rede recolheria de forma transparente para o **RSE** a informação com diferentes formatos, estruturas e significados; disponibilizando-lhos de seguida de num estado uniformizado.

#### Cenário 4: **RTS** como fonte de dados para plataformas de interoperabilidade transfronteiriça



**Figura 3.6:** A **RTS** a disponibilizar dados no formato **epSOS**

A mobilidade dos cidadãos, quer por motivos profissionais ou apenas de férias, acontece já há alguns anos a uma escala internacional. Por este motivo os profissionais de saúde prestam, cada vez mais, cuidados a utentes estrangeiros e conseqüentemente o atendimento acontece sem que haja forma de aceder a informação clínica do doente. É então uma necessidade a implementação de uma plataforma europeia para a troca de informação clínica.

Vários esforços aconteceram já no sentido da disponibilização da informação médica entre os países da União Europeia. Uma das entidades a intervir neste processo foi a Comissão Europeia que emitiu em 2008 recomendações referentes à interoperabilidade transfronteiriça entre sistemas de registo eletrónicos de informação clínica. [8] Nesse mesmo ano estabeleceu-se o projeto Smart Open Services for European Patients (**epSOS**) que visa a implementação de uma arquitetura piloto para a troca de informação entre os sistemas de registo clínico

eletrónico dos países europeus. [1].

O projeto [epSOS](#) acarreta a promessa da prestação de cuidados médicos, mais eficientes e principalmente mais seguros. Este projeto baseia-se na troca de informação transfronteiriça, com base em documentos clínicos. Foram definidas duas áreas de intervenção, o que levou à definição de dois documentos para a troca de informação:

- Patient Summary; [19]
- e-Prescription (inclui e-Dispensation); [6]

A arquitetura do [epSOS](#) é uma arquitetura distribuída baseada no paradigma da orientação a serviços. Cada país participante implementa um componente de comunicação a que foi dado o nome de *National Contact Point* (NCP). Este componente tem a função de servir de ponto de encontro entre a infra-estrutura de informação clínica de um país, e a restante rede, efetuando a transformação necessária à elaboração dos documentos transmitidos no [epSOS](#). Os serviços de comunicação foram implementados como [WSs](#), num paradigma de cliente-servidor.

Sendo a arquitetura do [epSOS](#) baseada no pressuposto da troca de informação entre plataformas nacionais, a [RTS](#) como sistema de registo eletrónico de informação clínica, poderia representar esse papel.

## 3.2 Cenário a implementar

Qualquer um dos cenários apresentados requer que a [RTS](#) seja dotada de novas capacidades, passando tais pela implementação de uma base comum nas áreas dos serviços, segurança e capacidade de interoperabilidade semântica.

A ordem pela qual os presentes cenários foram apresentados está indexada à sua complexidade e dificuldade de execução. Acharmos que deveria ser feita uma análise aos requisitos de cada um deles, e que após enquadramento como os objetivos da dissertação e período de desenvolvimento disponível, poder-se-ia então escolher um deles.

Foi definido que havendo uma ordem de complexidade, e acima de tudo, pela interdependência entre as diferentes áreas a desenvolver, seria o âmbito inicial deste trabalho a criação dos serviços necessários para que [ATs](#) possam consultar a informação da [RTS](#) no seu modelo de dados atual e sobre as terminologias usadas no sistema. Definiu-se também como imperativo que a comunicação teria de ser feita de forma segura.

No âmbito da interoperabilidade semântica, achamos interessante utilizar o projeto [epSOS](#) como base para a criação de um serviço que disponibilize a informação da [RTS](#) segundo o modelo de dados, e terminologias do mesmo. Isto levanta desafios sintáticos e semânticos. Dotada de mecanismo para a implementação de tal serviço, a [RTS](#) ficaria apta a integrar outras arquiteturas, tanto no papel de cliente como no de distribuidor de informação.

O mapear deste *roadmap* nos cenários implementados é:

- Cenário 1 - a ser implementado na totalidade;
- Cenário 4 - criação de um *proof of concept* (prova de conceito) para a interoperabilidade semântica, segundo o modelo [epSOS](#);
- Cenários 2/3 - não sendo implementados, ficam disponíveis as base para a sua implementação.

Todo o desenvolvimento que se irá levar a cabo para a realização destes planos, serve de base a outros teatros de utilização, e não inviabiliza de forma alguma quaisquer cenários de integração.

A integração base com o sistema passará pela consulta da informação clínica dos utentes. Surge neste panorama um conjunto de casos de uso bem definidos, apresentados na secção 3.3 e ilustrados na figura 3.7.

### 3.3 Casos de uso

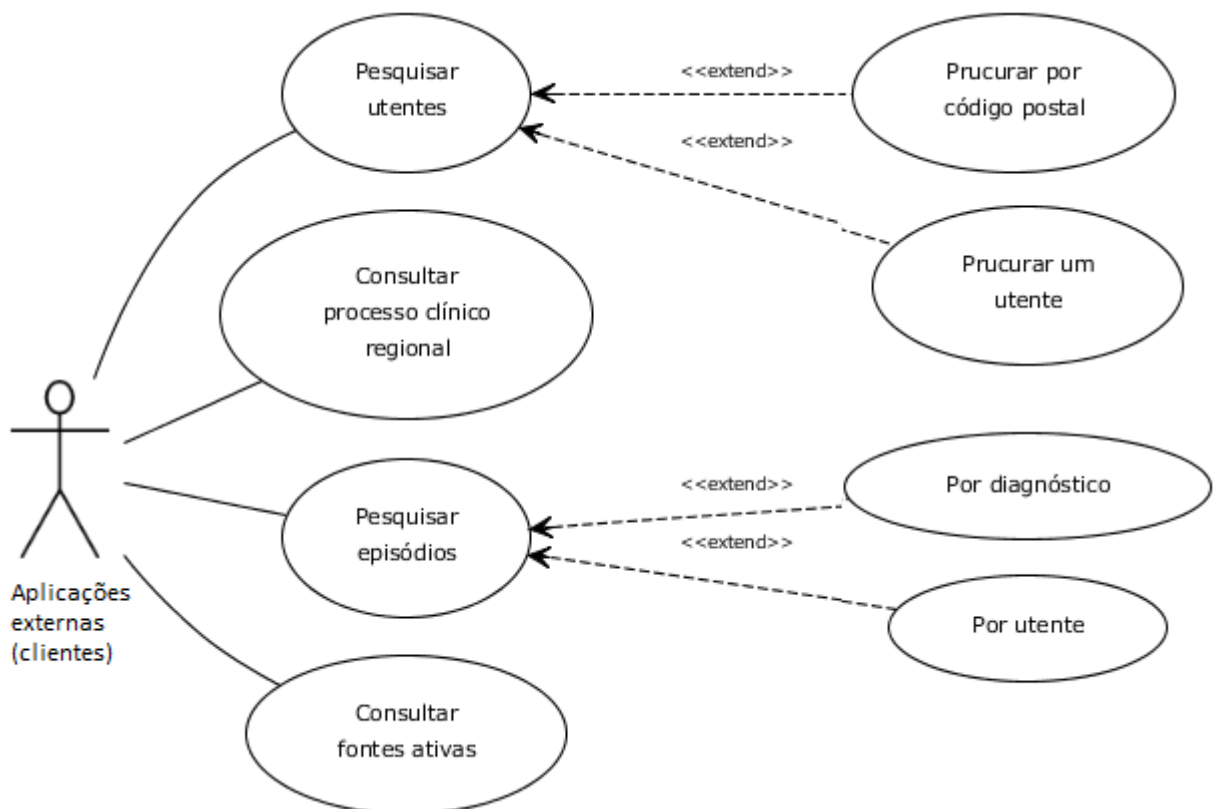


Figura 3.7: Casos de uso para os WS da RTS

#### Pesquisar utentes

O utilizador deve ter a possibilidade de pesquisar no sistema por um utente ou por um conjunto deles. Esta pesquisa de ser realizada mediante um identificador único do utente ou por um conjunto de parâmetros identificativos de um conjunto de utentes.

#### Consultar processo clínico regional

Deverá ser disponibilizado através da nova API um mecanismo para a consulta do Processo Clínico Regional.

### **Pesquisar episódios**

Sendo os episódios clínicos a base da informação recolhida pela [RTS](#), um caso de uso essencial para a plataforma será a capacidade de pesquisa sobre estes. A consulta de episódios pode ser de dois tipos, a pesquisa da informação de um determinado utente, ou a consulta de todos os episódios de uma determinada característica, por exemplo o diagnóstico.

### **Consultar fontes ativas**

O utilizador deverá ter a possibilidade de consultar quais as fontes da [RTS](#) que estão ativas e a ser utilizadas como fontes de dados para a informação a ser devolvida pelo sistema.

## Capítulo 4

# Modelo computacional do sistema

### 4.1 Interface de serviços

#### 4.1.1 Nova Arquitetura da aplicação

Esta dissertação propõe uma nova versão da Rede Telemática da Saúde (RTS), que disponibiliza em relação ao estado de implementação da RTS existente, duas áreas principais a desenvolver:

- exposição dos dados da plataforma através de serviços;
- suporte para operações semânticas, designadamente de relacionar diferentes modelos de informação.

Alcançar estas novas funcionalidades, requer a implementação de uma camada de Web Services (WSs) e a adição à plataforma de um subsistema para a gestão de terminologias.

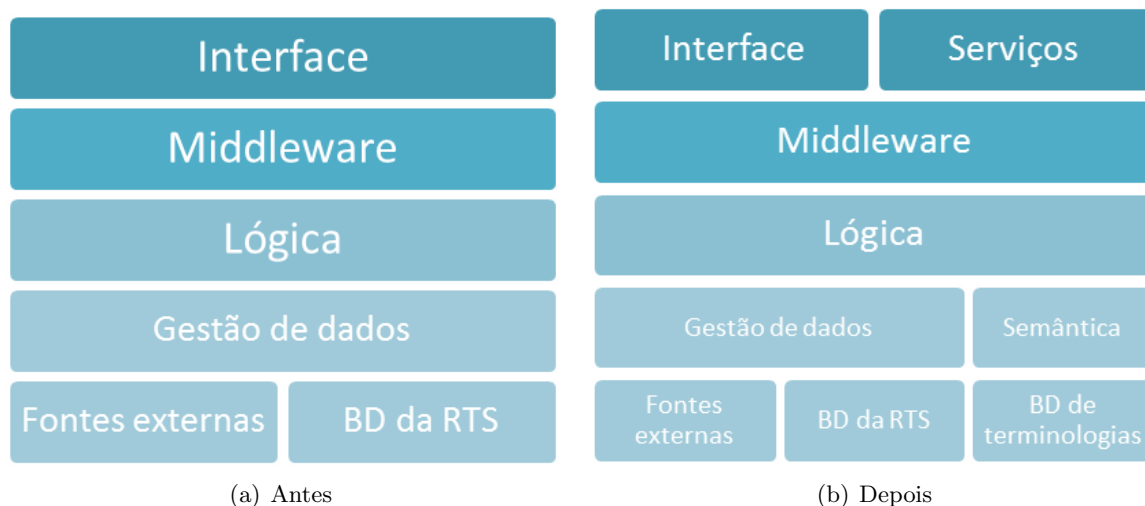
A arquitetura atual da RTS (figura 4.1(a)) foi planeada como uma sucessão de camadas de software, que em conjunto criam o sistema de informação. Estas camadas são:

- Fontes externas de dados - wrappers;
- Camada de acesso/manipulação de dados;
- Implementação lógica do modelo de negócio - camada lógica;
- Camada de engenharia do sistema, para abstração à utilização dos diversos módulos - camada de middleware;
- Camada de interface com o utilizador.

Por forma a acomodar as novas necessidades de desenvolvimento, devem surgir dois novos módulos nesta plataforma, tal como representado na figura 4.1(b). Transversal às camadas que compõem o middleware deverá surgir o módulo de semântica, para a persistência e consulta de terminologias. Surgirá uma nova camada, complementar à camada de interface, a camada de serviços.

Do modelo a várias camadas apresentado, apenas algumas necessitam de intervenção. Ao nível da lógica, é necessário instalar um servidor de vocabulário, e na camada de serviços serão

implementados os [WSs](#). Define-se então a arquitetura a desenvolver como um acrescento da arquitetura atual, pela adição de um servidor de vocabulário, dotação de mecanismos semânticos, e a exposição do modelo de negócio, ou parte dele, por serviços.



**Figura 4.1:** Arquitetura da [RTS](#). Antes (a) e depois (b).

## Serviços

Fazem parte desta categoria todos os componentes necessários para se poder expor os serviços. Como elemento nuclear temos o conjunto dos [WSs](#). Mas para se construir os serviços, não basta montá-los, é preciso criar um modelo de dados que sirva de suporte à comunicação com o cliente. Surge por isso a necessidade de desenvolver um componente de transformação de dados, que faça a transformação dos dados internos do sistema, para este novo modelo de dados.

## Lógica

Como se definiu no início desta secção, a implementação atual da [RTS](#) representa o componente responsável pela implementação da lógica do sistema, sendo assim o componente principal desta categoria. Note-se que para este trabalho este componente deve ser considerado uma "caixa negra", sobre a qual apenas se conhece a Application Programming Interface ([API](#)), e assim sendo nenhum tipo de desenvolvimento será levado a cabo nele. Neste grupo de componentes o desenvolvimento/instalação acontecerá com o servidor de vocabulário. Este servidor será utilizado para efetuar a transcodificação dos dados que estão na [RTS](#) num sistema de codificação interno, para um outro que se pretenda expor nos serviços. Para que tal seja exequível a este componente deverá também apresentar uma [API](#) que a camada de serviços poderá utilizar para realizar transcodificação e possivelmente também a tradução, de dados resultantes de consultas na [RTS](#).

## Bases de Dados (BDs)

Tal como acontece com o componente [RTS](#), as suas bases de dados, bem como as suas fontes de dados externas, não serão alvo de nenhum tipo de intervenção neste trabalho. No entanto como a informação que vai ser trocada com os clientes vem da camada lógica, e por conseguinte de tais destas fontes, deverá ser útil ter presente a sua existência. E à semelhança com o que acontece na lógica, também na persistência é preciso complementar a implementação atual com o suporte à persistência das terminologias, a serem usadas pelo servidor de vocabulário.

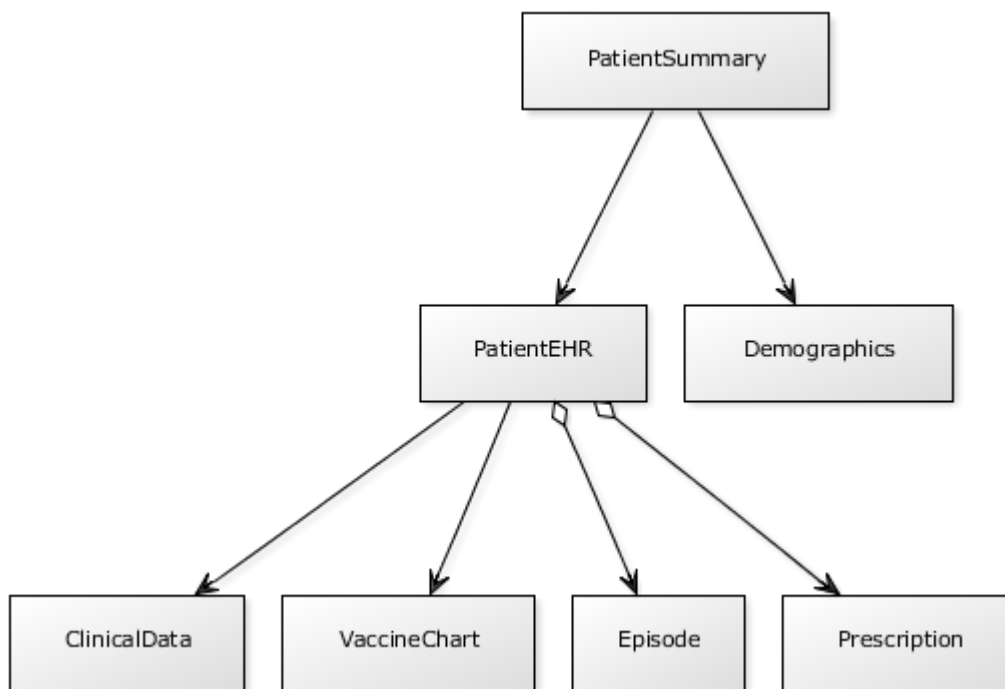
### 4.1.2 Serviços a disponibilizar

**Tabela 4.1:** Serviços a criar na [RTS](#)

Serviço	Descrição
Directory and Authority	Apresenta mecanismos de gestão de contas, e pesquisa de informação relativa as entidades de prestação de cuidados de saúde.
Extended	Disponibiliza informação pertinente de outros serviços mas agora em formatos ou codificações conhecidas de outros sistemas de informação de relevo.
Health record	Onde se consulta a informação sobre os episódios de um utente.
Patient index	Permite pesquisar por utentes, bem como aceder aos seus dados demográficos.
System diagnosis	Para aferir se o sistema se encontra disponível, e se todos os seus componentes estão operacionais.

### 4.1.3 Modelo de Dados

A [RTS](#) foi desenvolvida como um *middleware* de abstração para a informação clínica. Pode portanto definir-se sobre o sistema um conjunto de vistas, e diferentes componentes para as apresentar. Decidimos que uma evolução natural deste paradigma de utilização da plataforma, seria disponibilizar nos serviços, a informação segundo uma ótica de resumo clínico. A estrutura básica que se pretende partilhar a partir dos novos serviços é algo a que chamamos "Resumo clínico do utente", que do ponto de vista técnico tomará o nome de "*Patient Summary*". Na ótica deste trabalho, um resumo clínico é composto pelos elementos presentes na figura [4.1.3](#).



**Figura 4.2:** *Patient Summary*.

Nas tabelas 4.2 e 4.3 pode ver-se o significado de cada uma das classes representadas na figura 4.1.3.

**Tabela 4.2:** Elementos do *Patient Summary*

Classe	Descrição
PatientDemographics	Contem os dados pessoais do utente.
PatientEHR	Agregado de diferentes elementos de informação clínicos

**Tabela 4.3:** Elementos do *PatientEHR*

Classe	Descrição
ActivePrescription	Listagem de prescrições atuais
Episode	Apresenta informação sobre um episódio clínico
EssentialClinicalData	Contem dados médicos relevantes
vacinesChart	Registo de vacinas do utente

Os serviços usam um modelo de dados coerente com a representações internas da [RTS](#). Como operam sobre informação agregada e resumida, o modelo de informação é simplificado, quando comparado com as necessidades de um sistema completo de Electronic Health Record ([EHR](#)).

O modelo de dados exposto nos [WSs](#) é um modelo reduzido, na perspetiva de partilha de informação na região.



Este modelo tem analogias com o que, noutros sistemas, se designa por sumário clínico do doente (Patient Summary), na medida em que, da mesma maneira, procura apresentar um subconjunto de dados mais relevante para a mobilidade do doente.

Mas as estruturas disponíveis não estão orientadas ao documento, como em outros sistemas se encontra, por exemplo, em sistemas baseados em Health Level Seven International ([HL7](#)) Clinical Document Architecture ([CDA](#)).

## 4.2 Serviços seguros

### 4.2.1 Controlo da utilização

A utilização dos novos serviços da [RTS](#) deve ser condicionado a apenas os profissionais de saúde, previamente registados na plataforma. Apesar de haver o requisito de identificação, não serão disponibilizados métodos para o registo de novos utilizadores, mas apenas para controlo do acesso ao sistema.

Temos como requisitos de segurança:

- Autenticação;
- Autorização;
- Auditoria.

Para cumprir tais requisitos será implementada uma arquitetura de segurança conhecida como "AAA" [[17](#)].

#### **Autenticação**

Foram criadas, para controlar o acesso ao "Portal dos Profissionais", credenciais para cada utilizador, que o identificam univocamente perante o sistema. Estas credenciais são constituídas por um par nome de utilizador e palavra chave.

As credenciais do portal deverão ser reutilizadas nos serviços, uniformizando-se assim o acesso ao sistema independentemente do ponto de entrada.

#### **Autorização**

À imagem da autenticação, deve fazer-se mímica do sistema de autorização à existente no portal. Os utilizadores estão divididos em grupos, tendo cada grupo diferentes privilégios, consequentemente, cada utilizador pode apenas aceder à informação disponível para o seu nível de acesso. A este mecanismo de controlo chama-se Role-based authorization (autorização baseada em perfis).

No caso dos serviços a autorização baseada em grupos, traduz-se na necessidade de adicionar a cada serviço, uma lista que identifique os grupos com os privilégios necessários à sua utilização.

#### **Auditoria**

Atualmente os acessos ao "Portal do Profissional" são registados, guardando-se alguns dados sobre o acesso a atividade. Estes registos são persistidos numa base de dados criada para este fim.

Visto existir uma base de dados para registo de Auditoria, os serviços deverão também fazer uso desta. Ao partilhar este armazenamento, não só se diminui o esforço de implementação (não sendo necessário criar-se uma **BD** com este propósito), mas também se permite que um sistema que consulte estes dados, adquira informação sobre ambos os tipos de acesso ao sistema.

## 4.2.2 Comunicação Segura

É importante garantir que a comunicação entre uma aplicação cliente e os serviços da **RTS** é feita de forma segura, isto é, respeitando todas as características apresentadas anteriormente.

A segurança em **WSs** é um assunto que está bem estudado e documentado. A uma escala tão vasta quanto a utilização dos mesmos, apresentam-se também os mecanismos para os segurar. Foram definidas um conjunto de normas com o intuito de uniformizar as configurações de segurança a aplicar em **WSs**, entre estas é de salientar a **WS-Security**, onde se definem as bases da segurança para as mensagens em Simple Object Access Protocol (**SOAP**).

Os serviços a desenvolver para a **RTS** devem seguir esta norma. Dos diferentes mecanismos possíveis o que mais se adequa aos requisitos é mecanismo conhecido como *Mutual Certificate Authentication* (certificação mutua por certificados).

Todos os **WSs** devem ser configurados de forma a trocar apenas mensagens cifradas, e mais importante que isso, a responder apenas a pedidos vindos de aplicações confiáveis.

Com o intuito de certificar e identificar as aplicações terceiras, foi gerado para cada uma destas um certificado, tendo este de ser apresentado perante a **RTS** a cada invocação aos seus serviços. Por conseguinte, cada aplicação possui um certificado que a identifica univocamente perante o servidor. Sempre que uma aplicação envia uma mensagem a um dos serviços envia também o seu certificado, sendo este utilizado pelo mecanismo que impõe a segurança na comunicação.

A utilização de um certificado por parte de uma aplicação cliente não se restringe apenas à sua identificação, este está também associado a um par de chaves assimétricas. Estas chaves são utilizadas para cifrar e assinar as mensagens trocadas com os serviços.

Quando se gera um certificado para uma Aplicação Terceira (**AT**), não se cria somente um certificado, mas também um par de chaves assimétricas. Este par é composto por uma chave privada e uma chave pública, sendo o certificado um "documento" que atesta a posse de uma chave pública por parte de um indivíduo ou aplicação. Quando uma **AT** é credenciada é-lhe disponibilizado o seu certificado acompanhado pelo seu par de chaves.

Dotada de um par de chaves, uma **AT** quando envia uma mensagem, para além de a fazer acompanhar do seu certificado cifra-a e assina-a.

Uma mensagem é cifrada para garantir que será impercetível para qualquer leitor desta, que não seja o seu destinatário esperado. Para além disso, todas as mensagens são também assinadas por forma a identificar a sua autoria.

## 4.3 Módulo de semântica

### 4.3.1 Servidor de vocabulário

Pretendemos que seja adicionado à arquitetura atual da **RTS** um componente responsável pelo armazenamento e manutenção das terminologias usadas no sistema.

Atualmente alguns dos dados recolhidos junto das instituições prestadores de cuidados de saúde, estão codificados. Tais códigos são parte de um conjunto de terminologias que estão a ser utilizadas nos sistemas de registo de informação médica. Estas terminologias, não são nem universais, nem imutáveis, sendo assim necessário adicionar à plataforma, um componente onde as diferentes terminologias estariam armazenadas, e seriam geridas de forma a garantir a sua correção e validade, de forma centralizada.

A **RTS** vai ser dotada de um componente de terminologia, com o propósito de permitir tais capacidades. A instalação de uma instancia do servidor de vocabulário LexEVS, permite a manipulação de terminologias, num ambiente controlado, e independente da implementação interna da **RTS**. Este componente de gestão semântica é composto por duas camadas:

- armazenamento - armazena as terminologias, mantendo num ponto centralizado a gestão das terminologias. Pode conter várias versões de cada.
- utilização - consiste na tradução dos códigos nos nomes de conceitos que estes representam, e *vice versa*. Aqui também se faz transcodificação entre terminologias.

#### 4.3.2 Casos de utilização

- Adicionar uma terminologia;
- Editar uma terminologia armazenada;
- Remover uma terminologia armazenada;
- Pesquisar conceitos por código;
- Pesquisar conceitos por descrição;
- Pesquisar as relações de um conceito;
- Recuperar os conceitos relacionados com um em específico;
- Transcodificar um código num "sinónimo" de outra terminologia;



# Capítulo 5

## Implementação

### 5.1 Web Services (WSs)

#### 5.1.1 Módulo de serviços da Rede Telemática da Saúde (RTS)

A implementação das várias camadas da [RTS](#) está dividida num conjunto de projetos Maven, cujos nomes são sempre precedidos de 'RTS.', criamos então mais um para o desenvolvimento da camada de serviços - RTS.WS.

O novo módulo interage com a restante plataforma apenas através do módulo RTS.IENGINE - a implementação da camada de integração (middleware). Este projeto permite abstrair a funcionalidade da [RTS](#) disponibilizando uma Application Programming Interface ([API](#)) como ponto de entrada único para a plataforma. Estes métodos disponibilizam a informação no modelo de dados de transporte da [RTS](#).

Para a comunicação com as aplicações externas, foi desenvolvido um modelo de dados especificamente para este propósito. Este modelo é um refinamento do modelo exposto pelo módulo de integração. A elaboração deste modelo teve em consideração apenas a informação que se queria disponibilizar, as boas práticas quanto à modelação de objetos, e foi inspirado no modelo dos documentos do Smart Open Services for European Patients ([epSOS](#)).

A interação base dos serviços com a [RTS](#) baseia-se na invocação dos serviços internos necessários, seguida da transformação dos objetos de transporte para o modelo de comunicação. Neste sentido, e tendo também em conta as necessidades de segurança, foram definidos três *packages* primários:

- *referencemodel* - do qual fazem parte todas as classes que compõem o modelo de dados para a comunicação com os sistemas exteriores. Este modelo de dados representa toda a informação da [RTS](#) que é disponibilizada para o exterior, não contemplando classes desenvolvidas especificamente para auxiliar a comunicação.
- *innerproxy* - contem duas categorias de classes: serviços de transformação de dados, e serviços de proxy. As primeiras contêm classes de utilidades, para a transformação dos objetos, entre o modelo de transporte da [RTS](#) e o modelo de comunicação para os serviços. As últimas são constituídas por classes que pretendem abstrair a utilização da [RTS](#), fornecendo uma adaptador para as funcionalidades disponíveis no módulo de integração. Os métodos destas classes recebem como parametros e retornam instancias do modelo de dados comunicação, abstraindo-se assim também os [WSs](#) da especificação interna para a representação de dados.

- aaa - disponibiliza classes com funcionalidades de autenticação, autorização e auditoria. Estas classes são também uma abstração aos mecanismos internos do sistema, permitindo que as classes da camada de serviços, possam utilizar estas funcionalidades de uma forma mais cómoda.

Nas sub-seccões seguintes serão apresentados cada um dos serviços individualmente, e os respetivos métodos.

### 5.1.2 Serviços disponibilizados

Tal como especificado foram desenvolvidos os seguintes serviços presentes na tabela-5.1.

**Tabela 5.1:** Serviços a criar na [RTS](#)

Serviço	Descrição
Directory Authority and	Apresenta mecanismos de gestão de contas, e pesquisa de informação relativa as entidades de prestação de cuidados de saúde.
Extended	Disponibiliza pertinente de outros serviços mas agora em formatos ou codificações conhecidas de outros sistemas de informação de relevo.
Health record	Onde se consulta a informação sobre os episódios de um utente.
Patient index	Permite pesquisar por utentes, bem como aceder aos seus dados demográficos.
System diagnosis	Para aferir se o sistema se encontra disponível, e se todos os seus componentes estão operacionais.

## 5.2 Modelo de dados

Nesta secção será apresentado (em suporte gráfico) o modelo de dados que foi concebido para o transporte de informação sobre os serviços. É de salientar que o modelo é bastante complexo, e que apenas algumas das classes criadas, serão aqui apresentadas.

As figuras 5.1, 5.2 e 5.3 foram selecionadas para representar partes do modelo, estando representadas em cada uma delas as classes relevantes para o contexto apresentado.

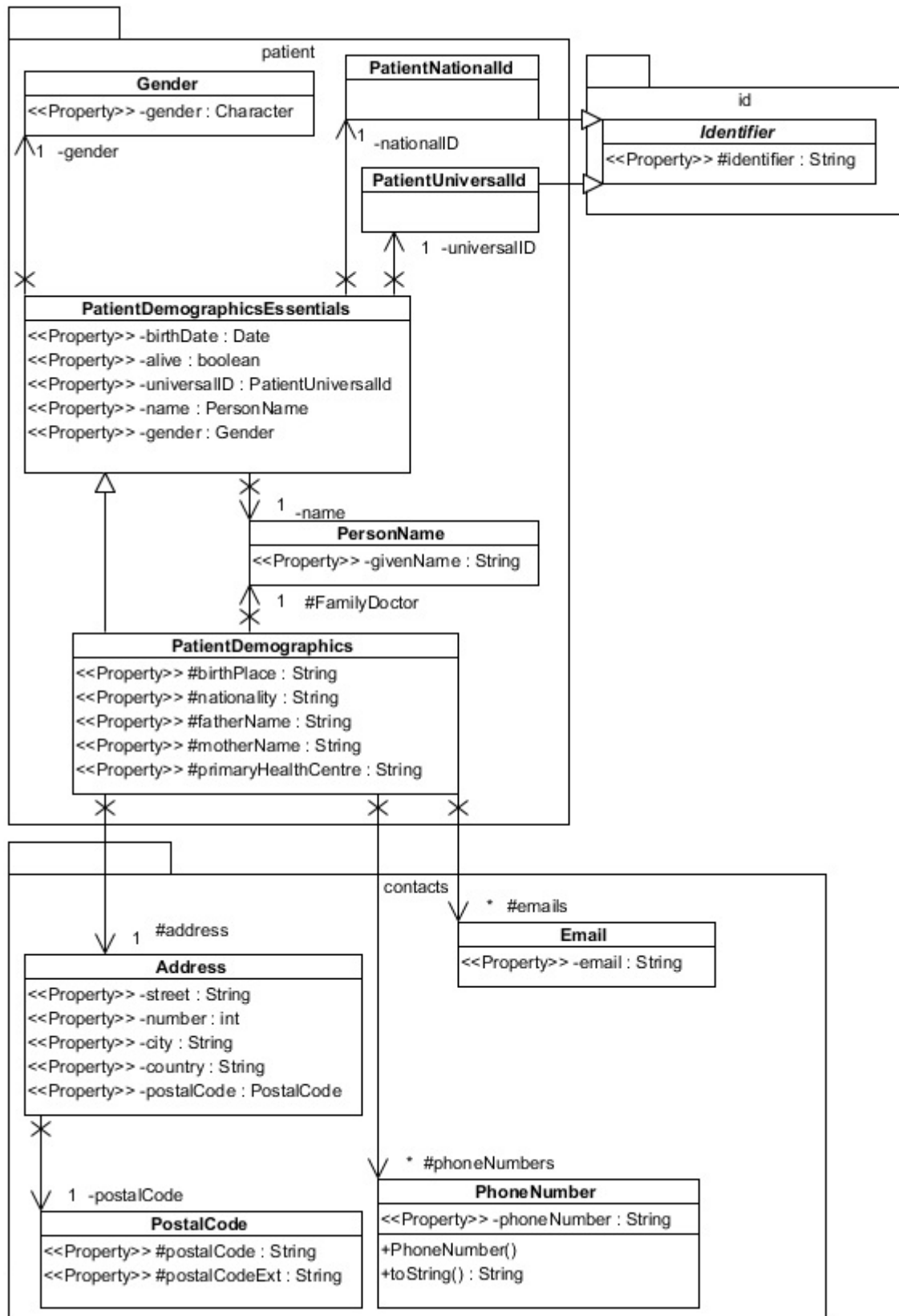


Figura 5.1: Diagrama de classes - Utente

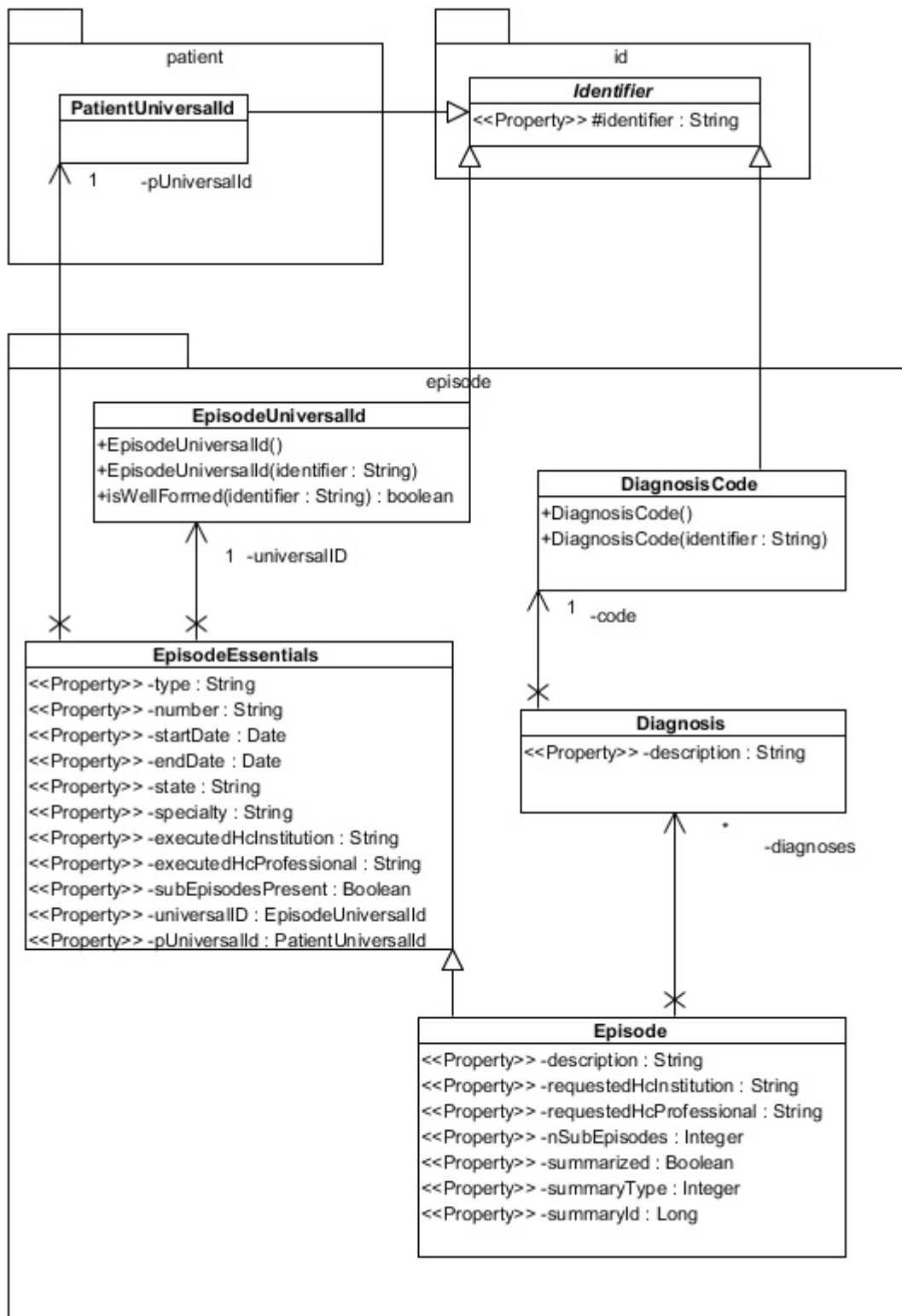


Figura 5.2: Diagrama de classes - Episódio



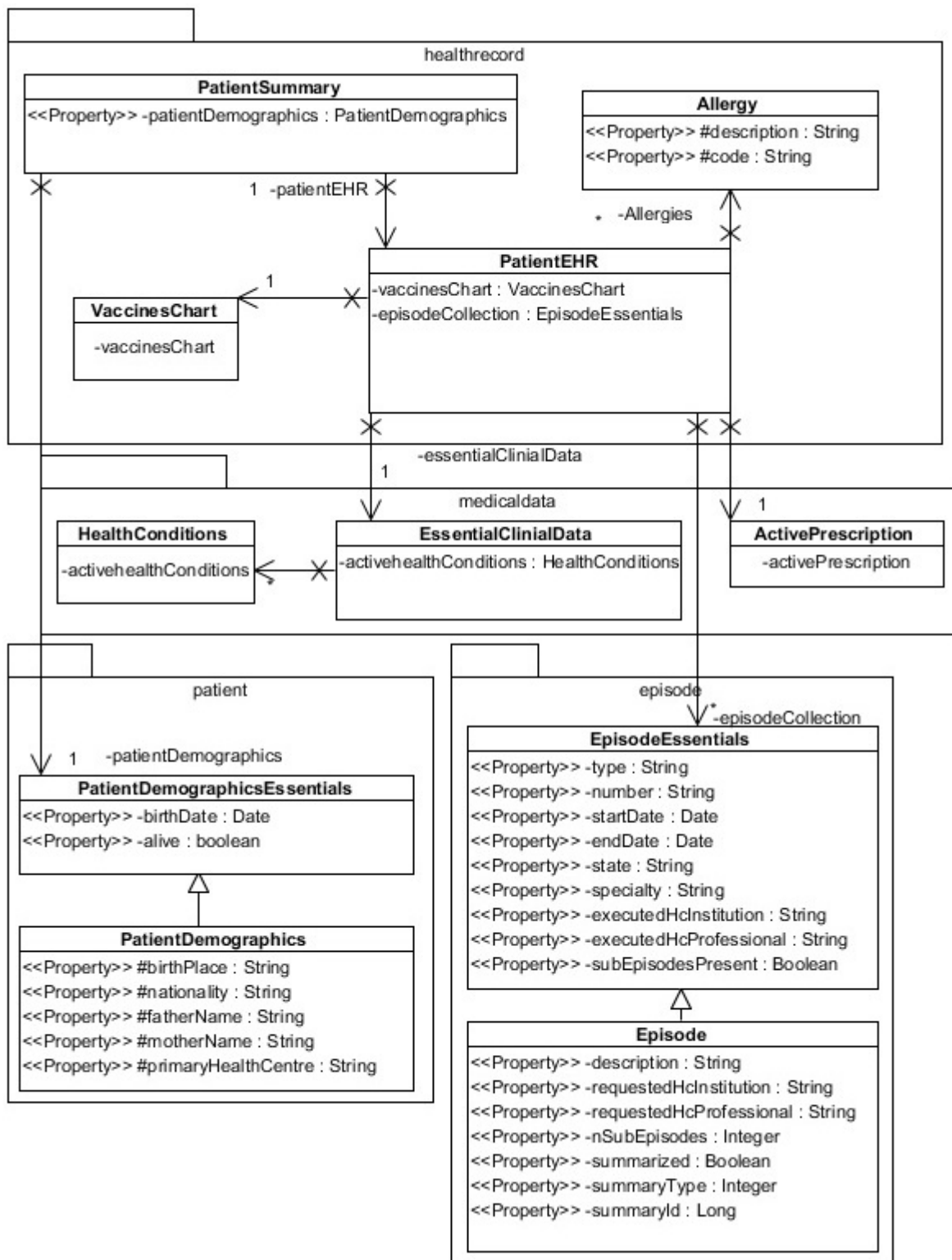


Figura 5.3: Diagrama de classes - Registo Clínico

## 5.3 Segurança nos serviços da RTS

### 5.3.1 Controlo de acesso

O controlo de acesso à aplicação é feito a três níveis, autenticação, autorização e auditoria. Um dos requisitos do sistema era que a autenticação ocorresse uma só vez, e que as chamadas subsequentes fossem acompanhadas de um identificador de autorização. Esse identificador seria utilizado pelo mecanismo de autorização para validar se o utilizador faz parte dos grupos com acesso ao serviço pretendido. Os requisitos definiam também que toda a utilização do sistema deveria ser auditada.

#### Autenticação

De forma a garantir acesso apenas a utilizadores conhecidos pelo sistema, foram criados métodos para autenticação, e pelo facto de existir um identificador de autorização, há também um método para invalidar este identificador.

**Tabela 5.2:** Directory and Authority

Serviço	Descrição
<code>initSession</code>	Autentica um utilizador, dado um conjunto <code>username</code> e <code>password</code> .
<code>endSession</code>	Termina uma sessão, invalidando o contexto da sessão.
<code>initSessionWithInstitution</code>	Autentica um utilizador, dado um conjunto <code>username</code> , <code>password</code> e uma instituição.

O método `initSession` é usado pelo utilizador para se autenticar perante o sistema, recebendo como retorno um objeto de contexto. Definimos que este contexto seria concebido como uma forma de transportar configuração de sistema, e dados do utilizador entre invocações do sistema. Na prática, o único conteúdo que é adicionado a este objeto, é o identificador de utilização. Um utilizador da [RTS](#) possui para fins de autenticação, um par: identificador e palavra chave. Assim, este par compõe os parâmetros de entrada deste método.

```
SessionResult initSession(String username, String password)
```

**Tabela 5.3:** `initSession`

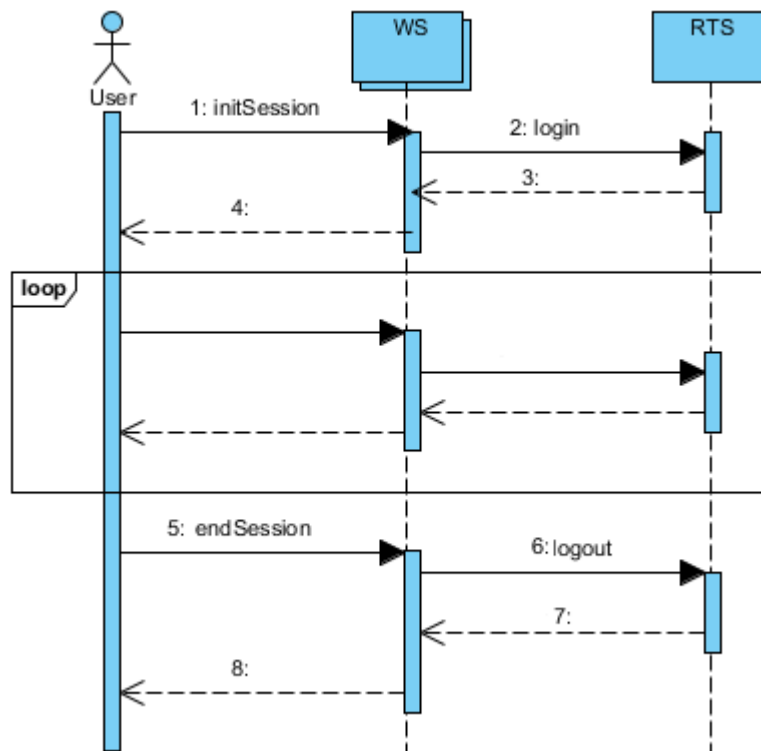
Operação	<code>initSession</code>	
Descrição	Autentica um utilizador, dado um conjunto <code>username</code> e <code>password</code> .	
Entradas	<code>username</code> <code>password</code>	identificador do utilizador palavra chave
Retorno	<code>RtsSessionContext</code>	Um objeto de contexto, que deve ser apresentado nas invocações subsequentes.
Pré-condições	<code>username != null;</code> <code>username.isEmpty() == false;</code> <code>password != null;</code> <code>password.isEmpty() == false;</code>	

Os profissionais de saúde, como utilizadores do "Portal dos Profissionais", têm um identificador por cada instituição a que pertencem (e.g. `HIPMedico01`). Para facilitar a autenticação dos profissionais de saúde já registados, foi criado um segundo método: `initSessionWithInstitution`, este é redundante com o anterior, tendo sido criado apenas por uma questão de semântica. A implementação deste serviço, é a mesma que o anterior, com a pequena diferença que a primeira coisa que se faz neste método é a concatenação dos parâmetros `institution` e `username`, para criar o identificador.

```
SessionResult initSessionWithInstitution(String username, String institution, String←  
password)
```

O método `endSession` tal como o nome indica, deve ser usado para terminar um contexto de sessão. Neste método o identificador de autorização, é invalidado.

```
RtsWsResult endSession(RtsSessionContext context)
```



**Figura 5.4:** Utilização normal do sistema

## Autorização

Os utilizadores da **RTS** estão divididos em grupos, servindo estes grupos para o controlo de acesso aos conteúdos disponíveis na páginas web. Estes grupos foram utilizados para implementar o controlo de acesso ao nível dos **WSs**. Cada **WS** pode apenas ser utilizado pelos utilizadores que fazem parte do conjunto de grupos associados.

Todas as mensagens trocadas entre os clientes e o servidor, transportam um objeto de contexto, contendo este um identificador de autorização. Como este contexto necessita de existir em todos os pedidos, este objeto foi adicionado a todos os métodos, como mais um parâmetro.

Deve separar-se o código que implementa a lógica daquele que impõe as restrições de segurança; e assim, retirar-se do código dos métodos, todas as questões relacionadas com o controlo de utilização destes. Este desacoplamento permite que as restrições de acesso, e a forma como se impõe, possam mudar sem que haja necessidade de se alterar nada no código dos serviços.

Foi criado um SOAP Handler para interceptar todas as mensagens que chegam servidor, com o intuito de realizar o controlo das autorizações. Um Handler é uma classe que possui um método que é invocado sempre que uma mensagem SOAP chega ou deixa o servidor.

A implementação da verificação da autorização consistiu em:

- Identificar qual o método a ser invocado;
- Extrair o identificador de autorização;

- Validar o identificador;
- Confirmar se o utilizador, pertence aos grupos autorizados.

## Auditoria

Segundo o mesmo paradigma que foi utilizado no controlo de autorização, foi criado um SOAP Handler, para realizar a recolha de dados das mensagens para auditoria. Para auditoria são registados vários dados de acesso aos serviços, entre eles:

- IP do cliente;
- Hora do acesso;
- Serviço invocado;
- Parâmetros utilizados;
- Utilizador, e respetivo perfil;

### 5.3.2 Comunicação segura

As Aplicação Terceira (**AT**) necessitam de ter certificados para apresentar aos serviços da **RTS** de forma a serem identificadas. Este certificado tem de ser emitido por uma entidade que seja de confiança no contexto e segurança da **RTS**, isto é, tem de ser uma entidade certificadora na qual se confia.

Foi criada uma entidade certificadora para a **RTS**. Esta entidade serviu apenas para criar um certificado para a própria **RTS**, e um conjunto de certificados para serem utilizados pelo seus clientes.

O módulo de **WS** necessita apenas de confiar na entidade certificadora que foi criada para este efeito, pois consequentemente irá confiar em todos os certificados que por ela tenham sido emitidos.

A **RTS** tem um par de chaves, e conhece o certificado da Credential Authority (**CA**) no qual confia. Uma **AT** tem um par de chaves próprio, e conhece o certificado da **RTS** que utiliza para cifrar as mensagens.

Cada **WS** tem de ser configurado individualmente. Foi criado para cada um deles um ficheiro com a descrição do mecanismo de segurança, que será utilizado pela biblioteca que o implementará, para o configurar.

O ficheiro de configuração define, que tipo de segurança se aplica e qual o mecanismo. No caso Mutual Certificate, configurou-se que as mensagens são cifradas recorrendo a chaves assimétricas. A cifra das mensagens será apoiada nos certificados dos intervenientes. Para cada um dos métodos definiu-se quais as partes da mensagens são cifradas, e quais têm de ser assinadas.

## 5.4 Modulo de Semântica

A utilização do LexEVS como servidor de vocabulário requer a utilização de uma base de dados para o armazenamento das terminologias. Assim o primeiro ato na implementação do módulo semântico da **RTS** foi a criação de uma base de dados para tal. Este sistema apresenta

algumas restrições quanto aos Sistemas de gestão de bases de dados (SGBDs) que podem ser utilizados, isto é, garante-se apenas compatibilidade com um pequeno conjunto destes. A RTS faz uso de PostgreSQL, e como este é um dos sistemas compatíveis, foi criada uma base de dados neste SGBD. A utilização desta base de dados, será abordada posteriormente, aquando da configuração do servidor.

A instalação do LexEVS é realizada pela execução de um ficheiro do tipo *jar*, isto vai criar uma estrutura de diretórios, abaixo do escolhido durante a instalação, onde serão colocados todos os ficheiros necessários à execução do servidor de vocabulário. Entre estes encontra-se o ficheiro de configuração do sistema, que é necessário preencher. Entre os diversos campos que são necessários definir encontram-se as configurações da base de dados. Um excerto deste documento encontra-se em 5.1.

**Listing 5.1:** Ficheiro de configuração do LexEVS

```
# The location of the file that will store information about all loaded ↔
  terminologies.
REGISTRY_FILE=registry.xml

# The folder that will hold all of the generated system indexes.
INDEX_LOCATION=../lbIndex

##### LexGrid System performance variables

# The maximum number of SQL connections to open (and pool) per database.
MAX_CONNECTIONS_PER_DB=25

# Size of the Dynamic Cache (this is used to cache frequently accessed information)
CACHE_SIZE=500

# Length of idle time before invalidating Iterators (in minutes)
ITERATOR_IDLE_TIME=5

# Max number of results that can be returned at once by any query operation.
MAX_RESULT_SIZE=1000

##### Database Configuration

# The current persistence scheme used to persist to an underlying store.
CURRENT_PERSISTENCE_SCHEME=2.0

# true - all ontologies in one common set of tables
# false - each ontology in a separate set of tables
SINGLE_TABLE_MODE=false

# JDBC Connection URL string used to connect to the database
DB_URL=jdbc:postgresql://heartbeat.ieeta.pt:5432/lexevs

...
```

Este servidor de vocabulário, pode ser instalado como uma aplicação Web acessível por serviços, ou então como uma biblioteca a ser incluída numa aplicação que necessita de a usar. Por motivos desempenho, simplicidade de utilização, e facilidade de implementação, optamos pela utilização da biblioteca.

No diretório de instalação, para além de todos os ficheiros instalados, é depositado um arquivo do tipo jar, que quando adicionada numa aplicação, fornece os mecanismos necessários

à utilização do LexEVS.

Está também disponível uma aplicação independente que permite através de uma interface gráfica a utilização de uma sub-parte das funcionalidades do LexEVS. Através desta aplicação é possível carregar terminologias, e fazer pesquisas sobre elas. Na figura 5.5.

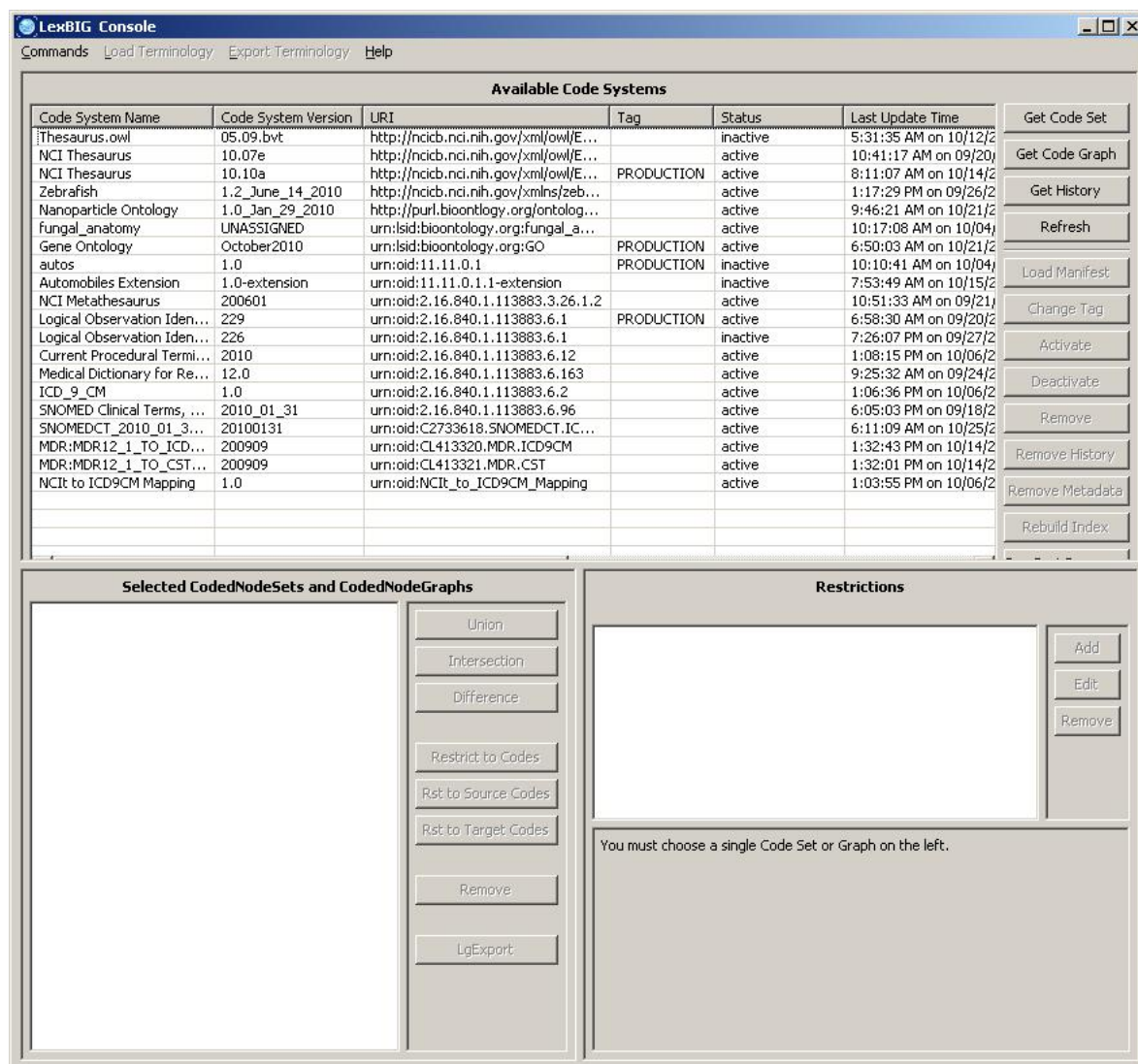


Figura 5.5: Aplicação *standalone* de interação com o LexEVS

## 5.5 Garantia de funcionamento

A programação orientada a objetos, tal como o nome indica, baseia-se em entidades (os objetos) e as relações entre elas. Em qualquer sistema criado sobre este paradigma, a arquitetura dos seus algoritmos é sempre uma composição de vários objetos, onde estes de forma orquestrada se utilizam uns aos outros. Por vezes testar um algoritmo inteiro, pode dificultar a depuração do código como um todo ou tornar-se mesmo ser impossível. É necessário implementar o paradigma "Dividir para Conquistar", testando diferentes pedaços do código e/ou

da aplicação de cada vez, pode permitir uma verificação mais fácil da validade dos mesmos. Desde cedo surgiu a necessidade de executar testes sobre os serviços que iam sendo desenvolvidos, para garantir o seu funcionamento correto. Por correto funcionamento entende-se que a camada de serviços opera corretamente sobre a camada de integração.

### 5.5.1 jUnit

#### Teste unitários

Os mecanismos criados para a camada de serviços apoia-se na camada de integração e não apresentam complexidade que obrigue à decomposição do fluxo de implementação em diferentes métodos (de diferentes classes).

Os testes unitários poder-se-iam aplicar aos métodos de cada serviços, no entanto do ponto de vista da invocação estrita, não há necessidade de desenvolver testes, porque se implementaram, como parte do código de cada serviço disponibilizado, um conjunto de verificações das pré-condições. Estas condições como parte integrante de cada serviço garantem a consistência e validade dos parâmetros de entrada, despoletando em caso negativo o termino do serviço e o retorno de um código e uma mensagem de erro.

#### Testes funcionais

Pretendia com recurso a esta ferramenta efetuar testes funcionais sobre o sistema como um todo. Através desta forma seriam criados vários testes, onde cada um deles serviria para testar uma funcionalidade do sistema. O problema com esta abordagem é a dependência que a [RTS](#) tem de um *container* no servidor de aplicações. Por esta razão, apenas se pode invocar os serviços após a aplicação estar a correr no servidor, não sendo possível instanciá-los e testá-los num ambiente comum de uma aplicação Java.

### 5.5.2 Aplicação Cliente

Quando se desenvolvem qualquer tipo de recurso a ser disponibilizado a terceiros, é importante garantir que do ponto de vista destes, é possível fazer uso do recurso.

No cenário dos serviços web, um cliente necessita de conseguir contactar os serviços, e de comunicar com estes de forma inteligível. Foi uma das tarefas teste trabalho, garantir que tal acontecia. A única forma de garantir com certeza que uma aplicação se pode ligar aos serviços da [RTS](#) e utilizá-los com sucesso é desenvolver uma. Foi então criada uma aplicação cliente, com o intuito de comprovar que é possível consumir os serviços disponibilizados pela [RTS](#).

A utilização de um serviço implementado segundo o protocolo Simple Object Access Protocol ([SOAP](#)), impõe a capacidade da aplicação cliente, aceder ao respetivo [WSDL](#), de o compreender e de conseguir criar o código para o invocar. Pressupõe também, visto as mensagens serem trocadas no formato eXtensible Markup Language ([XML](#)), que as mensagens sejam interpretadas de forma correta, ou seja, que não existam problemas de comunicação.

Foi criada uma aplicação com estes objetivos. Foram todos cumpridos com sucesso!

Dispondo de uma aplicação cliente, dispõe-se também de uma ferramenta de testes para os serviços. Depois de se criarem mecanismos de invocação para todos os serviços disponibilizados, estes podem ser utilizados, para testar os requisitos funcionais dos serviços.



Através da aplicação cliente é possível testar individualmente cada serviço, comprovar que cada um deles comunica de forma correta com a camada de integração e que cria um objeto para retorno, coerente com o esperado.

Este mecanismo acarreta a intervenção de um humano para se realizarem os testes, o que é um fator bastante limitador. Se estes se realizassem de forma automática, poder-se-iam fazer com uma periodicidade previamente definida, poderiam realizar relatórios também automáticos, e poderiam estar incluídos num qualquer mecanismo de controlo de qualidade, de forma transparente para o programador. Por outro lado, permite de uma forma bastante simples, realizar testes aos serviços. Permitindo ainda que os testes sejam executados por alguém que não saiba programar, ou não esteja familiarizado com os requisitos para se utilizarem os serviços.

Recorrendo a esta aplicação todos os serviços que iam sendo desenvolvidos, iam também sendo testados, garantindo-se assim, que o sistema evoluía sempre sobre a premissa do restante sistema estar estável.

### 5.5.3 Teste com aplicações de alunos

No âmbito da disciplina de Engenharia de Software de 2011, os alunos foram divididos em grupos, com o intuito de que cada grupo desenvolvesse um sistema de informação. Alguns desses grupos criaram aplicações que tinham a **RTS** como fonte de dados. Tais aplicações contactavam com a **RTS** através da **API** de serviços que estava então em desenvolvimento.

O período de desenho e modelação da arquitetura de tais projetos, coincidiu com o período de implementação dos serviços deste trabalho. Foi então criado um canal de comunicação entre mim, e os alunos dos grupos. Esta comunicação tinha dois objetivos:

- Perceber as necessidades que os alunos tinham quanto aos serviços;
- Prestar apoio à implementação das aplicações terceiras.

Foi então concedido ao alunos um período de alguns dias, durante o qual, poderiam solicitar aos seus professores, a adição de serviços específicos, de acordo com as suas necessidades. Todos os pedidos foram por nós ponderados, e todos foram consentidos. A lista ser serviços a implementar foi então adaptada a estes novos requisitos. Assim foram criadas as condições necessárias ao desenvolvimento das aplicações terceiras.

Para facilitar a criação de uma aplicação cliente por parte dos alunos, parte da aplicação de teste que tinha sido previamente criada para testar os serviços, foi-lhes disponibilizada. Sentiu-se então a necessidade de uma plataforma de disponibilização não só da aplicação, mas também de informações úteis (e.g. quais os serviços disponíveis ou como configurar a segurança).

Está disponível, para toda a comunidade da Universidade de Aveiro, a plataforma *Redmine*. *Redmine* é uma plataforma web para a gestão de projetos, permitindo entre outras coisas, rastrear problemas (*issues*), associação a um repositório de código e ainda a criação de uma *wiki*. A esta instanciação na universidade deu-se o nome de Code.UA e está disponível em <http://code.ua.pt>. Decidimos recorrer a esta plataforma como plataforma de distribuição de conteúdos para o alunos.

Criei um novo projeto no Code.UA, onde foi colocada a aplicação a disponibilizar para os grupos. Para além do código fonte da aplicação, os alunos tinham também acesso a um

documento que explicava, o modelo de dados utilizado, a [API](#) disponível e ainda um passo-a-passo sobre como dotar as suas aplicações de mecanismos para comunicarem com os serviços da [RTS](#) de forma segura.

Como o desenvolvimento das aplicações terceiras, se fazia ao mesmo tempo que se terminava o desenvolvimento dos serviços, estava também disponível no Code.UA uma wiki, que continha informação sobre o estado de implementação. Na wiki podia consultar-se qual a ultima versão disponível, as alterações desde a versão anterior, e mais alguns dados técnicos sobre os serviços.

Os alunos desenvolveram aplicações cliente, que serviram do ponto de vista deste trabalho como provas de conceito, mostrando que é possível criarem-se aplicações clientes dos serviços da [RTS](#), e mais importante, que os serviços se comportavam da forma pretendida.

## Capítulo 6

# Conclusões

### 6.1 Discussão de resultados

Este trabalho apresentava-se como um desafio tecnológico, o qual requeria a aquisição de conhecimento especializado.

O desafio proposto era a interoperabilidade em plataformas de informação clínica, estando esta associada a duas problemáticas bem identificadas, a comunicação entre os sistemas computacionais, e capacidade de entendimento mutuo. Este teve como base a Rede Telemática da Saúde (RTS) onde havia interesse de expandir a plataforma dotando-a de uma camada de serviços para acesso remoto, e da capacidade de comunicar sobre diferentes sistemas de representação de dados clínicos.

Este desafio foi superado com êxito. Conseguimos implementar um conjunto de serviços, configurados para serem acedidos de forma segurança, e um servidor de vocabulário para a interoperabilidade semântica.

O sucesso nesta tarefa significa que agora é possível para sistemas externos à plataforma aceder à sua informação. Esta melhoria retira a obrigatoriedade da utilização do "Portal dos Profissionais", permitindo a elaboração de aplicações de interface para profissionais de saúde, libertando-os da necessidade de usar o browser. Para além da possibilidade de se criar novas formas de interagir com o sistema, esta nova forma de comunicação, permite também que sejam desenvolvidas aplicações que disponibilizam a informação de novas formas: transformando dados, reorganizando-os ou mesmo complementando-os com outros dados.

A nova capacidade demonstrada de disponibilização dos dados segundo diferentes terminologias, permite que, se forem criados serviços para tal, seja possível disponibilizar a informação contida na RTS preparada para qualquer sistema semântico, bastando para tal que haja compatibilidade semântica nos dados trocados.

## 6.2 Aspetos em aberto e trabalho Futuro

Apesar do objetivo da dissertação ter sido alcançado, alguns aspetos podem ser melhorados, e algumas adições podem ser feitas ao trabalho desenvolvido.

Pelo facto da utilização de um qualquer serviço da [RTS](#) ser independente dos restantes, não há necessidade de se manter um mecanismo de sessões, podendo este ser substituído pela adição de um *token* de autenticação presente em todas as mensagens. A não adoção desta medida, mantendo-se o mecanismo atual, pode mesmo assim ser melhorada. A segurança de um serviço é um requisito não funcional, não devendo constar na interface deste, nem mesmo na sua implementação (se assim for possível). Neste sentido o identificador de sessão que é utilizado para controlo de acesso deve deixar de fazer parte da interface dos serviços, e passar a ser um elemento do header (cabeçalho) das mensagens.

A implementação dos serviços faz-se valer nas situações de erro de códigos de erro (e respetivas mensagens), para dar a conhecer ao utilizador a ocorrência de situações anómalas. Estas mensagens devem ser revistas por alguém que não o programador dos serviços. As próprias situações que levam ao reportamento de uma situação de erro, são diversificadas, e seria por exemplo útil, que os erros fossem categorizados.

Os serviços implementados ainda não disponibilizam a totalidade da informação disponível na [RTS](#), assim mantem-se a oportunidade de desenvolvimento de serviços para a plataforma. Mais especificamente o serviço de extensão (Extensions) pode ser alargado à restante informação disponível, mas para além disso, pode ainda ganhar novos métodos que disponibilizem dos dados fazendo uso de terminologias novas (ao sistema).

O âmbito do servidor de vocabulário pode ser alargado para além da [RTS](#), pode ser carregado com novas terminologias, e estar acessível remotamente, para que outros sistemas possam usufruir deste.

O modelo de dados atual da [RTS](#) pode não permitir o registo de qualquer tipo de valor, isto é, ter a capacidade de armazenar dados de qualquer terminologia. Um desafio de futuro poderia passar pela implementação de um novo modelo de dados, mais flexível e dinâmico. Este modelo de alta abrangência, tornar-se-ia o novo modelo de dados canónico do sistema, capaz de armazenar informação clínica, "independente" da fonte.

# Referências

- [1] Smart Open Services for European Patients (epSOS). <http://www.epsos.eu>. Accessed Oct 30, 2011.
- [2] RTS. <http://rtsaude.org>. Accessed Oct 30, 2011.
- [3] Java ee 6 - web services. <http://download.oracle.com/javase/6/tutorial/doc/bnayk.html>. Accessed June 1, 2011.
- [4] Jaxb reference implementation project. <http://jaxb.java.net>. Accessed June 1, 2011.
- [5] Why web services? [http://www.w3schools.com/webservices/ws\\_why.asp](http://www.w3schools.com/webservices/ws_why.asp). Accessed Oct 10, 2011.
- [6] Marta Bonilla and María José Piña. Final definition of functional service requirements – eprescription. 2010.
- [7] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. Web services architecture, w3c working group note 11 february 2004. *World Wide Web Consortium*, 2004.
- [8] European Commission. Eu commission recommendation of 2 july 2008 on cross-border interoperability of electronic health record systems. 2008.
- [9] Ministério da Saúde. Despacho n.º 27311/2009. 2009.
- [10] Grupo de Trabalho do Registo de Saúde Electrónico (RSE). *Plano de Operacionalização*. Gabinete do Secretário de Estado Adjunto e da Saúde, 2009.
- [11] D.E. EASTLAKE and K. NILES. Secure xml: The new syntax for signatures and encryption, paperback. 2002.
- [12] M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, H. Nielsen, A. Karmarkar, and Y. Lafon. Soap version 1.2 part 1: Messaging framework . w3c recommendation. april. *World Wide Web Consortium (W3C)*. Web site: [www.w3.org](http://www.w3.org), 2007.
- [13] H. Haas and A. Brown. Web services glossary. w3c working group note. *World Wide Web Consortium (W3C)*, 2004.
- [14] E. Hewitt. *Java SOA cookbook*. O'Reilly Media, 2009.
- [15] R. Housley, W. Polk, W. Ford, and D. Solo. Internet x. 509 public key infrastructure certificate and certificate revocation list (crl) profile. *RFC3280*, 66, 2002.

- [16] IC Oliveira and JP Cunha. Integration services to enable regional shared electronic health records. *Studies in health technology and informatics*, page 310, 2011.
- [17] N. Papatheodoulou and N. Sklavos. Architecture & system design of authentication, authorization, & accounting services. In *EUROCON 2009, EUROCON'09. IEEE*, pages 1831–1837. IEEE.
- [18] OASIS Web Services Security TC. Web services security v1.1 (ws-security). 2006.
- [19] Clara Vaquerizo. Final definition of functional service requirements - patient summary. 2010.
- [20] Andre Zúquete. *Segurança em Redes Informáticas*. FCA, 2010.

## Apêndice A

# Tabelas técnicas dos métodos disponibilizados na **API** de **WSs**

### A.1 DirectoryAndAuthority

**Tabela A.1:** DirectoryAndAuthority

Serviço	Descrição
retrieveHealthCareUnits	Retorna o conjunto de entidades de cuidados de saúde registradas no sistema

**Tabela A.2:** retrieveHealthCareUnits

Operação	retrieveHealthCareUnits	
Descrição	Retorna o conjunto de entidades de cuidados de saúde registradas no sistema	
Entradas	context	contexto associado à sessão atual
Retorno	HealthCareUnitsResult	O conjunto de HealthCareUnits registrados
Pré-condições	context != null; context - must provide access;	

### A.2 HealthRecord

**Tabela A.3:** HealthRecord

Método	Descrição
RetrieveEpisode	Retorna episódio detalhado dado um episódio resumido
retrieveEpisodeEssentials	Retorna um episódio resumido dado o seu identificador universal
FindEpisodesByPatientUId	Procura os episódios raiz de um utente dado o seu identificador universal
FindEpisodesByPatientNId	Procura os episódios raiz de um utente dado o seu identificador nacional
findEpisodesByPatientNIDandStartDate	Procura os episódios raiz de um utente dado o seu identificador nacional e a data de ocorrência mais antiga
findEpisodesByPatientNIDStartDateAndType	Procura os episódios raiz de um utente dado o seu identificador nacional, a data de ocorrência mais antiga e o tipo de episódio
FindEpisodesByDiagnosis	Procura os episódios raiz de um utente dado um diagnóstico
FindSubEpisodesByEpisodeUIdAndPatientUId	Procura os sub-episódios de um episódio dado o identificador universal do seu episódio raiz e o identificador universal do utente
FindEubEpisodesForParentEpisode	Procura os sub-episódios para um dado episódio
RetrievePatientSummary	Devolve um resumo da informação clínica disponível na Rede Telemática da Saúde ( <a href="#">RTS</a> )

**Tabela A.4:** retrieveEpisode

Operação	retrieveEpisode	
Descrição	Retorna episódio detalhado dado um episódio resumido	
Entradas	context eEssentials	contexto associado à sessão atual episódio resumido
Retorno	EpisodeResult	The Episode
Pré-condições	context != null; context - must provide access; eEssentials != null;	



**Tabela A.5:** retrieveEpisodeEssentials

Operação	retrieveEpisodeEssentials	
Descrição	Retorna um episódio resumido dado o seu identificador universal	
Entradas	context eUniversalId	contexto associado à sessão atual identificador universal do episódio
Retorno	EpisodeResult	Um episódio
Pré-condições	context != null; context - must provide access; eUniversalId != null; eUniversalId.isWellFormed();	

**Tabela A.6:** findEpisodesByPatientUID

Operação	findEpisodesByPatientUID	
Descrição	Procura os episódios raiz de um utente dado o seu identificador universal	
Entradas	context pUniversalId	contexto associado à sessão atual identificador universal do utente
Retorno	EpisodeEssentialsListResult	Um conjunto de episódios resumidos
Pré-condições	context != null; context - must provide access; pUniversalId != null; pUniversalId.isWellFormed();	

**Tabela A.7:** findEpisodesByPatientNID

Operação	findEpisodesByPatientNID	
Descrição	Procura os episódios raiz de um utente dado o seu identificador nacional	
Entradas	context pNationalId	contexto associado à sessão atual identificador nacional do episódio
Retorno	EpisodeEssentialsListResult	Um conjunto de episódios resumidos
Pré-condições	context != null; context - must provide access; pNationalId != null; pNationalId.isWellFormed();	

**Tabela A.8:** findEpisodesByPatientNIDandStartDate

Operação	findEpisodesByPatientNIDandStartDate	
Descrição	Procura os episódios raiz de um utente dado o seu identificador nacional e a data de ocorrência mais antiga	
Entradas	context pNationalId startDate	contexto associado à sessão atual identificador nacional do episódio data de ocorrência mais antiga
Retorno	EpisodeEssentialsListResult	Um conjunto de episódios resumidos
Pré-condições	context != null; context - must provide access; pNationalId != null; pNationalId.isWellFormed(); startDate != null;	

**Tabela A.9:** findEpisodesByPatientNIDStartDateAndType

Operação	findEpisodesByPatientNIDStartDateAndType	
Descrição	Procura os episódios raiz de um utente dado o seu identificador nacional, a data de ocorrência mais antiga e o tipo de episódio	
Entradas	context pNationalId startDate type	contexto associado à sessão atual identificador nacional do episódio a data de ocorrência mais antiga tipo de episódio
Retorno	EpisodeEssentialsListResult	Um conjunto de episódios resumidos
Pré-condições	context != null; context - must provide access; pNationalId != null; pNationalId.isWellFormed(); startDate != null; type != null; type.isEmpty() == false;	

**Tabela A.10:** findEpisodesByDiagnosis

Operação	findEpisodesByDiagnosis	
Descrição	Procura os episódios raiz de um utente dado um diagnóstico	
Entradas	context diagnosis startDate endDate	contexto associado à sessão atual diagnostico a procurar data da ocorrência mais antiga a mostrar data da ocorrência mais recente a mostrar
Retorno	EpisodeEssentialsListResult	Um conjunto de episódios resumidos
Pré-condições	context != null; context - must provide access; diagnosis != null; diagnosis.isWellFormed(); startDate != null; endDate != null;	

**Tabela A.11:** findSubEpisodesByEpisodeUIDAndPatientUID

Operação	findSubEpisodesByEpisodeUIDAndPatientUID	
Descrição	Procura os sub-episódios de um episódio dado o identificador universal do seu episódio raiz e o identificador universal do utente	
Entradas	context eUniversalId pUniversalId	contexto associado à sessão atual identificador universal do episódio identificador universal do utente
Retorno	EpisodeEssentialsListResult	Um conjunto de episódios resumidos
Pré-condições	context != null; context - must provide access; eUniversalId != null; eUniversalId.isWellFormed(); pUniversalId != null; pUniversalId.isWellFormed();	

**Tabela A.12:** findSubEpisodesForParentEpisode

Operação	findSubEpisodesForParentEpisode	
Descrição	Procura os sub-episódios para um dado episódio	
Entradas	context eEssentials	contexto associado à sessão atual episódio resumido
Retorno	EpisodeEssentialsListResult	Um conjunto de episódios resumidos
Pré-condições	context != null; context - must provide access; eEssentials != null;	

### A.3 SysDiagnosis

**Tabela A.13:** SysDiagnosis

Método	Descrição
isRTSAliveAndReady	Testa se todo sistema está operacional

**Tabela A.14:** isRTSAliveAndReady

Operação	isRTSAliveAndReady	
Descrição	Testa se todo sistema está operacional	
Entradas	context	contexto associado à sessão atual
Retorno	BooleanResult	<i>true</i> em caso afirmativo, <i>false</i> caso contrário
Pré-condições	context != null; context - must provide access;	

### A.4 PatientIndex

**Tabela A.15: PatientIndex**

Método	Descrição
retrievePDbyPDE	Retorna os dados demográficos completos dado um resumo dos dados demográficos do utente
retrievePDEbyPatientUID	Retorna o resumo dos dados demográficos do utente, dado um identificador universal do utente
retrievePDEbyPatientNId	Retorna o resumo dos dados demográficos do utente, dado um identificador nacional do episódio
findPatientsByPostalCode	Procura quais os utentes que moram na área de um dado código-postal
findPatientsByNameAndGender	Procura pelos utentes que correspondam a um dado nome e de um dado sexo
findPatientsByNameAndBirthday	Procura pelos utentes que correspondam a um dado nome e de uma dada data de nascimento

**Tabela A.16: retrievePDbyPDE**

Operação	retrievePDbyPDE	
Descrição	Retorna os dados demográficos completos dado um resumo dos dados demográficos do utente	
Entradas	context pde	contexto associado à sessão atual resumo dos dados demográficos do utente
Retorno	PatientDemographicsResult	dados demográficos do utente
Pré-condições	context != null; context - must provide access; pde != null;	

**Tabela A.17: retrievePDEbyPatientUID**

Operação	retrievePDEbyPatientUID	
Descrição	Retorna o resumo dos dados demográficos do utente, dado um identificador universal do utente	
Entradas	context pUniversalID	contexto associado à sessão atual identificador universal do utente
Retorno	PdeResult	resumo dos dados demográficos do utente
Pré-condições	context != null; context - must provide access; pUniversalID != null; pUniversalID.isWellFormed();	

**Tabela A.18:** retrievePDEbyPatientNId

Operação	retrievePDEbyPatientNId	
Descrição	Retorna o resumo dos dados demográficos do utente, dado um identificador nacional do episódio	
Entradas	context pNationalID	contexto associado à sessão atual identificador nacional do episódio
Retorno	PdeResult	resumo dos dados demográficos do utente
Pré-condições	context != null; context - must provide access; pNationalID != null; pNationalID.isWellFormed();	

**Tabela A.19:** findPatientsByPostalCode

Operação	findPatientsByPostalCode	
Descrição	Procura quais os utentes que moram na área de um dados código-postal	
Entradas	context postalCode	contexto associado à sessão atual código-postal a procurar
Retorno	PdeListResult	uma lista com o resumo dos dados demográficos do utente, para cada um encontrado
Pré-condições	context != null; context - must provide access; postalCode != null;	

**Tabela A.20:** findPatientsByNameAndGender

Operação	findPatientsByNameAndGender	
Descrição	Procura pelos utentes que correspondam a um dado nome e de um dado sexo	
Entradas	context pName pGender	contexto associado à sessão atual nome do utente sexo do utente
Retorno	PdeListResult	uma lista com o resumo dos dados demográficos do utente, para cada um encontrado
Pré-condições	context != null; context - must provide access; pName != null; pName.matches("[a-zA-Z]+([a-zA-Z]+)"); pGender != null;	

**Tabela A.21:** findPatientsByNameAndBirthday

Operação	findPatientsByNameAndBirthday	
Descrição	Procura pelos utentes que correspondam a um dado nome e de uma dada data de nascimento	
Entradas	context pName pGender	contexto associado à sessão atual nome do utente data de nascimento do utente
Retorno	PdeListResult	uma lista com o resumo dos dados demográficos do utente, para cada um encontrado
Pré-condições	context != null; context - must provide access; pName != null; pName.matches("[a-zA-Z]+([a-zA-Z]+)"); pGender != null;	