UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

BENE LEMUEL DANTAS GONDIM

BORIS: UM BROKER PARA INTEGRAÇÃO DE SERVIÇOS DE SAÚDE

MOSSORÓ
2020

BENE LEMUEL DANTAS GONDIM

BORIS: UM BROKER PARA INTEGRAÇÃO DE SERVIÇOS DE SAÚDE

Dissertação apresentada ao Programa de Pós-Graduação em Ciências da Computação – associação ampla entre a Universidade Federal Rural do Semi-Árido e a Universidade do Estado do Rio Grande do Norte, para a obtenção do título de Mestre em Ciências da Computação.

Orientador: Francisco Milton Mendes Neto, D.Sc – UFERSA
Coorientador: Paulo Gabriel Gadelha Queiroz, D.Sc – UFERSA

MOSSORÓ
2020

G637b Gondim, Bene Lemuel Dantas.
BORIS: UM BROKER PARA INTEGRAÇÃO DE SERVIÇOS DE SAÚDE / Bene Lemuel Dantas Gondim. - 2020.
66 f. : il.

Orientador: Francisco Milton Mendes Neto.
Coorientador: Paulo Gabriel Gadelha Queiroz.
Dissertação (Mestrado) - Universidade Federal Rural do Semi-árido, Programa de Pós-graduação em Ciência da Computação, 2020.

1. Saúde. 2. Integração de sistemas. 3. Broker. 4. Ontologia. I. Neto, Francisco Milton Mendes , orient. II. Queiroz, Paulo Gabriel Gadelha , co-orient. III. Título.

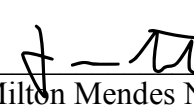
BENE LEMUEL DANTAS GONDIM

BORIS: UM BROKER PARA INTEGRAÇÃO DE SERVIÇOS DE SAÚDE

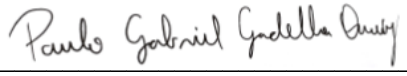
Dissertação apresentada ao Programa de Pós-Graduação em Ciências da Computação – associação ampla entre a Universidade Federal Rural do Semi-Árido e a Universidade do Estado do Rio Grande do Norte, para a obtenção do título de Mestre em Ciências da Computação.

Defendida em: 19 / 08 / 2020.

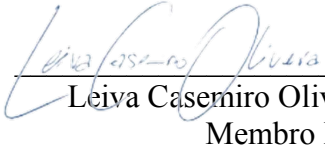
BANCA EXAMINADORA



Francisco Milton Mendes Neto, D.Sc – UFERSA
Presidente



Paulo Gabriel Gadelha Queiroz, D.Sc – UFERSA
Membro Examinador



Leiva Casemiro Oliveria, D.Sc – UFERSA
Membro Examinador

Cecilia Dias Flores, D. Sc – UFCSPA
Membro Examinador

A minha esposa, pelo amor e constante apoio durante o período de elaboração desse trabalho.

AGRADECIMENTOS

Agradeço a Deus pela vida e por ter me abençoado durante todo o processo.

Agradeço a minha esposa Dinara, companheira de todos os momentos, por todo o amor e estímulo que me deu para que eu conseguisse concluir o mestrado.

Agradeço aos meus pais, Aldemir e Maria Dalva, por todo o apoio que sempre me deram durante minha vida escolar.

Agradeço ao meu orientador, Prof. Dr. Milton Mendes, assim como ao meu coorientador, Prof. Dr. Paulo Gabriel, pela paciência e por toda ajuda a mim disposta durante essa trajetória.

Agradeço a Banca Examinadora por ter aceitado o convite de contribuir com esse trabalho.

Agradeço também aos meus amigos e familiares que torceram e acreditam.

Deus é bom em todo tempo. Em todo tempo Deus é bom.

Salmos 136:1

RESUMO

Diversos hospitais, clínicas e desenvolvedores oferecem serviços web aplicados à Saúde por meio de páginas na internet, aplicativos para *smartphones* ou pela utilização de sistemas físico cibernéticos. Cada um desses serviços dispõe de recursos específicos para o ambiente em que está inserido. Nesse trabalho é proposto um *broker* que possibilite a interoperabilidade entre sistemas de saúde existentes, de forma que os consumidores possam acessar recursos independente da representação e interface de um sistema particular. O *broker* possibilita que o usuário final possa ter acesso a mais recursos dentro de um mesmo sistema, assim como simplifica o processo de integração entre os sistemas. O *broker* foi desenvolvido utilizando como base uma ontologia, na qual são descritas as aplicações que desejam compartilhar seus recursos e uma interface *web*, por meio da qual os consumidores podem ver quais recursos estão disponíveis e detalhes sobre cada um.

Palavras-chave: Saúde, integração de sistemas, *broker*, ontologia

ABSTRACT

Many hospitals, clinics and developers use web services applicable to health through websites, smartphone applications or through the use of cyber physical systems. Each of these services provides resources to the environment in which it operates. In this paper, a broker is proposed that allows interoperability among existing health systems, so that consumers can access resources independent of the representation and interface of a specific system. The broker allows the end user to access more resources within the same system, as well as simplifying the integration process among systems. The broker was developed based on an ontology, in which are described the applications that share their resources, and a web interface, through which consumers can look which resources are available and details about each one.

Keywords: Health, systems integration, broker, ontology

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

A2A	Aplication-to-Aplication
API	Application Programming Interface
B2C	Business-to-Consumer
B2B	Business-to-Business
CORBA	Common Object Request Broker Architecture
EAI	Enterprise Application Integration
EDI	Electronic Data Interchange
FTP	File Transfer Protocol
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
MOM	Message Oriented Middleware
OIL	Ontology Interference Layer
OWL	Ontology Web Language
PDQ	Patient Demographics Query
PIX	Patient Identifier Cross-referencing
P2P	Peer-to-Peer
REST	Representational State Transfer
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
TCP/IP	Transmission Control Protocol / Internet Protocol
UDDI	Universal Discovery Description and Integration
URI	Uniform Resource Identifier
WAP	Wireless Application Protocol
WDSL	Web Service Description Language
XML	Extensible Markup Language
XOL	XML-Based Ontology Exchange Language

SUMÁRIO

1. INTRODUÇÃO

A saúde se põe em um lugar de destaque na história, ajudando, nos bastidores, as civilizações a evoluírem ao ponto em que se encontram. Melhorar a qualidade de vida implica em garantir maior expectativa de vida e melhores condições de trabalho, tanto física como intelectual, movendo, assim, as indústrias e a economia (Dogaru, et al., 2015). Tratar um indivíduo e gerenciar esses dados isoladamente foi efetivo no passado, porém é preferível que os dados de uma pessoa sejam analisados e comparados com as demais, para que seja possível estabelecer padrões, diferenças e similaridades para se conseguir encontrar a solução com maior precisão.

A indústria dos dispositivos médicos está passando por uma rápida transformação, incorporando a potência dos *softwares* embarcados e da conectividade das redes (Lee, et al., 2010). Ao invés de dispositivos isolados que são projetados, certificados e usados para tratar pacientes de forma isolada, os sistemas distribuídos controlam múltiplos aspectos do fisiologia do paciente, através de compartilhamento de dados e recursos (Lee, et al., 2006).

A evolução da tecnologia, como os Sistemas Físico-Cibernéticos, provê a oportunidade de usar essas tecnologias para interagir com o mundo físico e automatizar processos físicos em diferentes campos, como a medicina, para introduzir um novo nível de interações quando sistemas interagem com pessoas (Sultanovs, et al., 2016).

A primeira transição para a evolução dos serviços de saúde foi a telemedicina, no começo do século XX, em que médicos podiam tratar paciente através do uso de telecomunicações e tecnologia da informação. Logo após, com o rápido desenvolvimento da internet, a medicina cibernética começou a oferecer serviços (Dogaru, et al., 2015).

O crescimento da tecnologia e da indústria médica contribuiu para o interesse crescente em Sistemas Médicos Físico-Cibernético (Grispos, et al., 2017). Os denominados Sistemas Médicos Físico-Cibernéticos são sistemas de informação que integram dispositivos médicos com o intuito de prover tomada de decisão automatizada. Tais sistemas podem melhorar a vida dos pacientes, usando uma abordagem que possibilita, de forma automatizada, a obtenção de dados e recursos médicos disponíveis para localizar o tratamento para um paciente.

1.1. MOTIVAÇÃO

Muitos hospitais, clínicas e até mesmo desenvolvedores oferecem serviços web aplicados à saúde, seja por meio de páginas na internet, aplicativos para *smartphones* e, especialmente, por meio da utilização de sistemas físico-cibernéticos no qual pacientes podem marcar consultas ou exames, obter seus resultados, informações sobre quadros clínicos, além de poder ter um acompanhamento por meio de aplicativos específicos. Cada um desses serviços dispõe de recursos específicos para cada ambiente em que está inserido, mas muitas vezes se faz necessário acesso a recursos que estão disponíveis em serviços de terceiros. Porém, cada uma dessas aplicações disponibilizadas possui arquitetura própria, como também sua própria plataforma de acesso, dificultando, assim, o compartilhamento desses recursos e dados.

Esse problema pode ocorrer dentro de uma mesma empresa, é o caso de empresas de tamanho médio e grande. Por diversas vezes, essas empresas necessitam utilizar várias aplicações para poder dar suporte às operações do negócio, pois uma única aplicação não é capaz de cobrir todas as funcionalidades que são necessárias.

O Ministério da Saúde do Brasil, por meio da Portaria Nº 2.073, de 31 de agosto de 2011, tratou sobre a regulamentação do uso de padrões de informação em saúde e interoperabilidade entre sistemas de informação do SUS, tanto para sistemas das instituições pública como privadas. Por meio dessa portaria, o Ministério definiu o uso de tecnologia *Web Service*, identificados por meio de URIs. Também foi definido o uso de ontologias para representação de conceitos, terminologias e classificações relacionadas ao domínio da Saúde.

Com isso, a criação de uma ferramenta que possibilite a integração dos sistemas existentes, de forma que os consumidores possam, independente da representação e interface de um sistema particular, acessar seus recursos, apresenta-se como uma solução para que cada sistema possa oferecer mais serviços e de forma otimizada.

Existem dois aspectos principais que justificam a criação de tal ferramenta. O primeiro, centrado nos usuários finais das aplicações de saúde, refere-se aos benefícios da integração entre as aplicações que estes utilizam com outras aplicações de saúde. Com isso, objetiva-se disponibilizar para o usuário final mais funcionalidades dentro de um mesmo sistema. O segundo aspecto é direcionado aos desenvolvedores e concentra-se na simplificação do processo de integração entre seus serviços com outros serviços. Pretende-se minimizar o trabalho dos desenvolvedores, possibilitando que estes não precisem implementar recursos que já são compartilhados por outros sistemas, como também facilitando o acesso a recursos exclusivos de sistemas de terceiros.

1.2. OBJETIVOS

1.2.1. Objetivo geral

A finalidade deste trabalho é possibilitar a interoperabilidade entre sistemas de saúde, por meio do desenvolvimento de um *broker* que possibilite e simplifique tal integração, para permitir que os sistemas consumidores possam acessar recursos dos demais sistemas independentes da representação e interfaces de um sistema particular, assim como, sem ter o conhecimento de onde o recurso está localizado.

1.2.2. Objetivos específicos

Os objetivos específicos descrevem as metas a serem alcançadas no decorrer da implementação do trabalho, assim como os benefícios esperados, tendo em vista o objeto de estudo. Com o desenvolvimento deste trabalho, espera-se obter os seguintes resultados:

I. Definição do estado da arte da integração de sistemas;
II. Desenvolvimento de um *broker* de recursos para possibilitar a integração de sistemas de saúde;
III. Possibilitar que sistemas possam utilizar recurso de outros sistemas de maneira facilitada;
IV. Possibilitar que os usuários acessem recursos de diferentes fontes de forma unificada.

1.3. ORGANIZAÇÃO DO DOCUMENTO

O presente trabalho está organizado conforme a seguinte estrutura: No capítulo 2 são apresentados os procedimentos e métodos utilizados para a elaboração deste trabalho. No capítulo 3 é apresentado o referencial teórico que trata dos conceitos relevantes para a elaboração do trabalho. Os aspectos e implementação do *broker*, incluindo seu funcionamento, arquitetura desenvolvida e ontologia são demonstrados no capítulo 4. No capítulo 5 é mostrada a validação do *broker*. Por fim, no capítulo 6 são feitas as considerações finais, bem como são apresentadas propostas de trabalhos futuros.

2. PROCEDIMENTOS E MÉTODOS

Para a realização deste projeto de pesquisa foi necessário seguir as seguintes etapas:

I. Revisão bibliográfica: inicialmente foi realizada uma revisão bibliográfica com o objetivo de fazer um levantamento sobre o "estado da arte" dos trabalhos relacionados à integração de sistemas;

II. Estudo das tecnologias: foram estudadas as tecnologias relacionadas ao assunto, como arquiteturas e padrões de desenvolvimento através da leitura de artigos e publicações referentes ao assunto;

III. Levantamento e análise dos requisitos: foi feito o levantamento e análise dos requisitos funcionais e não funcionais necessários para o desenvolvimento do *broker,* com base no que foi levantado na revisão bibliográfica e em entrevistas realizadas com especialistas em desenvolvimento de sistemas de saúde. Em sequência, foi elaborada a arquitetura da aplicação, com base em publicações que classificam e definem padrões de requisitos e arquiteturas, como os presentes em Samtani, et al., (2002) e Moltchanov, et al., (2008);

IV. Prototipação: com base no conhecimento extraído da revisão bibliográfica, foi implementado, de forma incremental, um protótipo do *broker*, que foi validado com base nos padrões encontrados na revisão bibliográfica;

V. Desenvolvimento da ontologia: foi desenvolvida a ontologia que será usada para armazenar as informações das APIs dos serviços que irão compartilhar seus recursos. A ontologia foi construída com o auxílio da ferramenta Protégé[1]. Seu objetivo é descrever os serviços e seus recursos que estejam disponíveis no *broker*. Os detalhes da construção da ontologia são descritos na Seção 4.2;

VI. Validação do protótipo: a validação do *broker* foi feita com a utilização de dois sistemas simples, elaborados com o objetivo único de validar o *broker,* implementados em linguagem PHP. As APIs destes sistemas foram cadastradas no *broker* de forma que um sistema podia, através da API, acessar os recursos disponibilizados pelo outro sistema, após a solicitação ao *broker*;

VII. Aplicação no MobiLEHealth (Sombra, 2015) e no DOAR (Do Carmo, et al., 2019): com base na validação feita no protótipo, foram feitas melhorias no

[1] Protégé é uma ferramenta desenvolvida pela Universidade de Stanford, que oferece suporte para construção, teste e manutenção de ontologias e um *framework* para construção de sistemas inteligentes.

broker e assim o MobiLEHealth e o DOAR foram utilizados como um estudo de caso, no qual o *broker* passou por uma validação em um sistema real.

3. REFERENCIAL TEÓRICO

Neste capítulo são apresentados os conceitos necessários para o desenvolvimento do trabalho. A organização se dá da seguinte forma: primeiramente, na seção 3.1, é apresentado o conceito de integração de sistemas. Em seguida, na seção 3.2, é apresentado o conceito de *Broker*. Na seção 3.3, o conceito de ontologia é apresentado. Na seção 3.4, é apresentado o conceito de *web service.* Na seção 3.5 é apresentado o conceito de interoperabilidade de sistemas de saúde. Por fim, na seção 3.6 são apresentados os trabalhos correlatos.

3.1. INTEGRAÇÃO DE SISTEMAS

Segundo Fernandez (2004), a integração de sistemas pode ser dividida em níveis ou camadas. Cada camada oferece possibilidades distintas de integração, custos, recursos e formas de implementação.

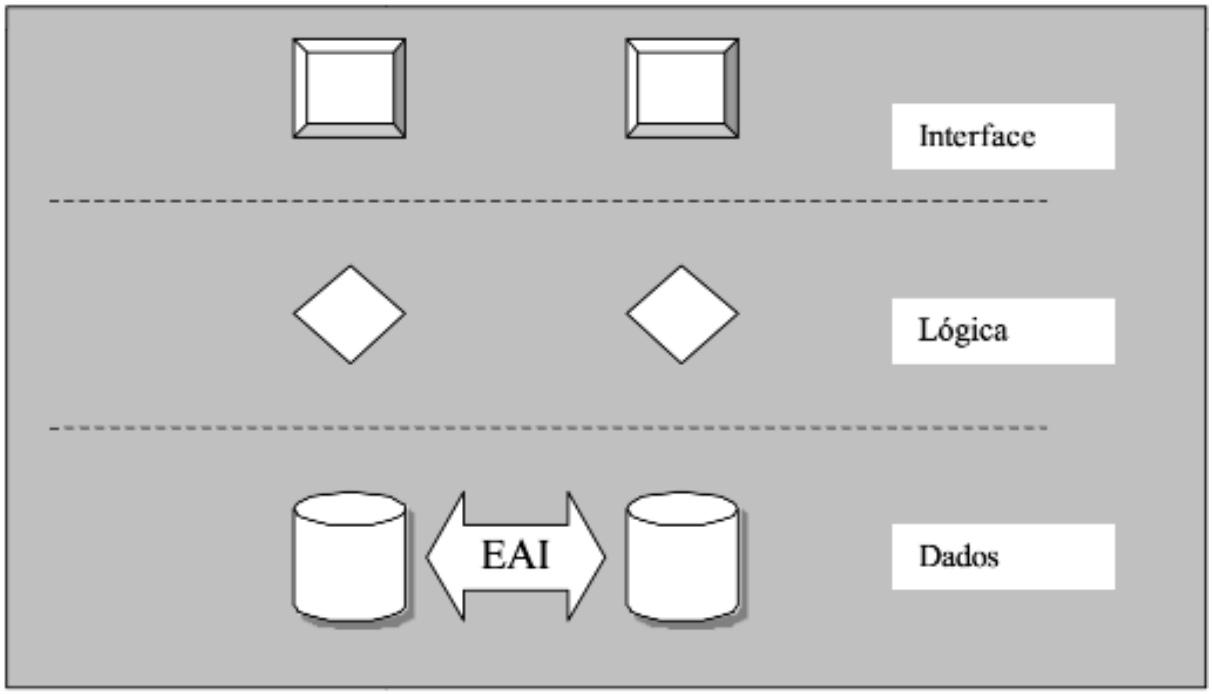
3.1.1. Integração das Plataformas Operacionais

A integração das plataformas operacionais apresenta-se como um pré-requisito para obtenção de outros níveis de integração. Tal nível de integração encontra seu maior desafio na interoperabilidade de sistemas legados, em que não há mais suporte dos fabricantes.

3.1.2. Integração por Dados

A integração no nível de dados é realizada pela troca direta entre depósitos de dados, sem envolver alterações na lógica das aplicações (Linthicum, 1999). Por não afetar diretamente as aplicações, é o nível de integração mais fácil de ser obtido. Quando as aplicações possuem suas camadas de dados, lógica e interface com usuário desacopladas (Figura 1), o processo de integração de dados é simplificado, tornando possível trabalhar somente com a camada de dados.

Figura 1 - Camada de Dados das Aplicações



Fonte: Linthicum, (1999).

A integração de dados é analisada sob dois aspectos, o sintático e o semântico (McGoveran, 2002). O aspecto sintático indica o formato que a informação deve ter em sua fonte e seu destino, de forma a decidir sobre a necessidade de tradução ou adaptação dos dados. O aspecto semântico trata do significado, integridade e valor da informação para o negócio.

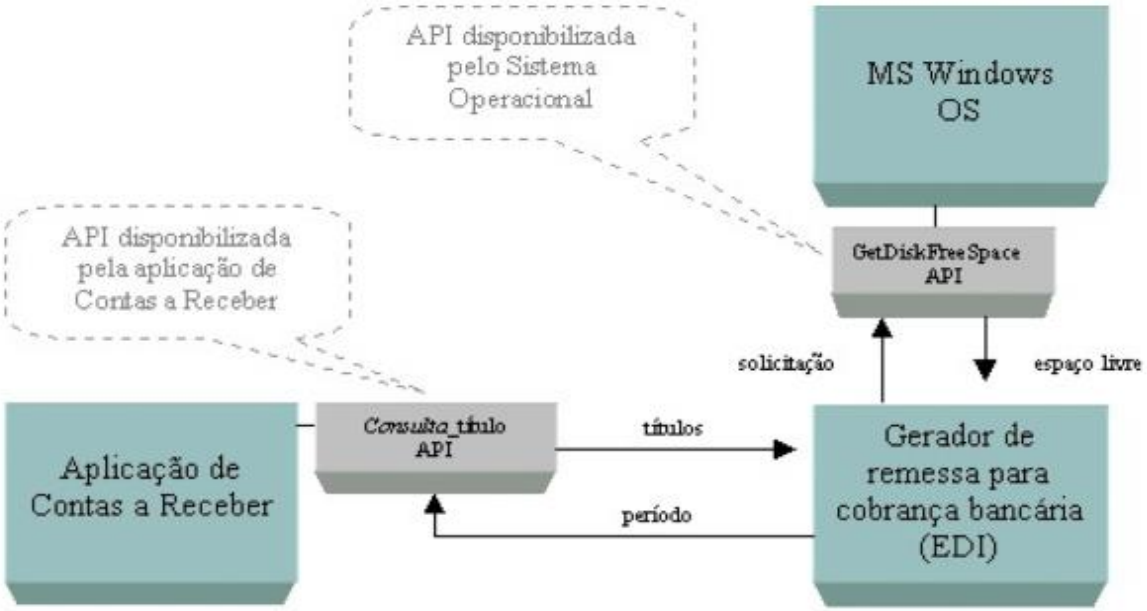
3.1.3. Integração pela Interface da Aplicação

A integração pela interface da aplicação é um tipo de integração de alto nível, em que uma aplicação utiliza funcionalidades disponibilizadas por outras aplicações (Juric, et al., 2001). As interfaces das aplicações são disponibilizadas pelos próprios desenvolvedores das aplicações e podem permitir acesso aos processos de negócio, aos dados ou a ambos, de forma a permitir o compartilhamento de dados e serviços. Segundo Lublisky (2001), esse tipo de integração produz uma troca de informações mais controlada do que no nível de integração por dados, porém é mais invasivo por exigir modificações nas aplicações.

Para se obter a integração de aplicações, são utilizadas APIs, que podem ser utilizadas no desenvolvimento de aplicações e integração. Segundo Juric, et al., (2001), APIs possibilitam uma integração de mais alto nível, possibilitando que aplicações utilizem funcionalidades existentes em outras aplicações.

API `s são mecanismos disponibilizados por desenvolvedores para permitir que outras aplicações utilizem bancos de dados, sistemas operacionais, servidores de aplicação e outros, de forma que se possa, de maneira facilitada, integrar sistemas diferentes com um custo reduzido. Além deste fator, a utilização de APIs promove a reutilização de tais funcionalidades, como mostrado na Figura 2.

Figura 2 - Exposição de interfaces nas aplicações - APIs

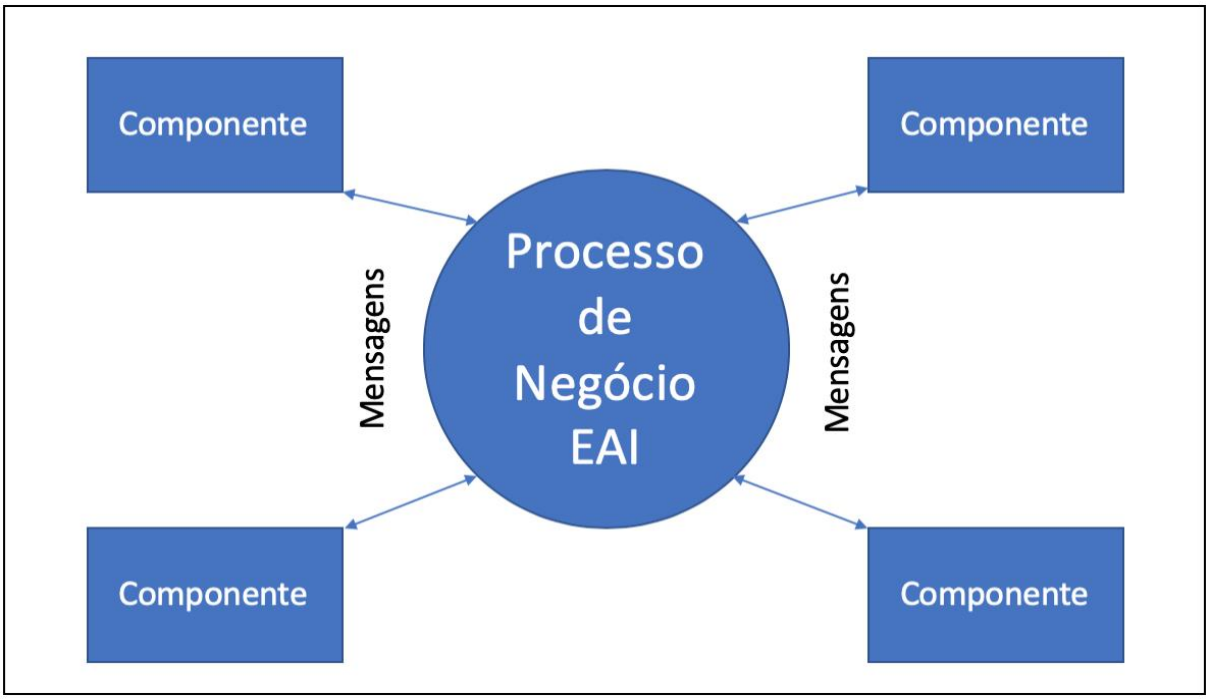


Autor: Fernandez (2004)

3.1.4. Integração por Métodos

Segundo Lithicum (2000), o nível de integração por métodos ocorre pelo compartilhamento dos processos de negócio. Lublinsky (2001) classifica esse nível como nível de processos e o classifica como uma forma orquestrada de troca de mensagens, na qual um *middleware* atua como um mecanismo de controle de execução de processos (Figura 3). Essa solução de integração é a mais invasiva, pois as aplicações precisam ter seu código alterado.

Figura 3 - Nível de processo EAI



Fonte: Lublinsky (2001)

3.1.5. Integração via Interface de Usuário

A integração via interface de usuário é utilizada quando não há outra possibilidade de integração devido à tecnologia da aplicação ser proprietária (Fernandez, 2004). Esse nível de integração, de acordo com Linthicum (2000), é o esquema de integração mais primitivo ou improvisado, sendo assim o mais limitado, restringindo-se às características da interface. Porém é o que apresenta maior facilidade por não ser intrusivo com relação à aplicação integrada.

Ainda segundo Linthicum (2000), a comunicação entre as aplicações ou sistemas se dá por meio de um *middleware*. O *middleware* faz a supressão das diferenças entre as partes comunicantes (Fernandez, 2004). O *middleware* pode ser de dois tipos: lógico e físico. O modelo lógico trata o movimento da informação pela organização do ponto de vista conceitual. Já o modelo físico trata o movimento da informação como realmente acontece.

Para que a integração de sistemas seja possível, é necessário um formato de descrição que possa ser entendido pelo computador. A ontologia vem sendo alvo de estudos em aplicações que necessitam de uma descrição formal do conhecimento. A seção 3.3 apresenta os conceitos relacionados às ontologias.

3.2. BROKER

O modelo Cliente-Servidor é um paradigma bem conhecido entre os serviços que estão disponíveis na *web*. Cada componente pode ter o papel de provedor ou consumidor de recursos (Moltchanov, et al., 2008). Essas entidades podem se conectar através da técnica *Peer-to-Peer (P2P)* ou por meio de um *broker*, que provê um diretório e busca por serviços (Kiani, et al., 2010).

Um *broker*, em suas várias formas, trabalha como um *middleware*[2], gerenciando comunicações e trocas de dados entre objetos ou entidades. Um *broker* prover uma plataforma de integração fim-a-fim, abordando os componentes críticos de um negócio, necessários para automatizar completamente o processo entre os parceiros (Samtani, et al., 2002).

Um *broker* armazena, transforma e roteia as informações extraídas de um nó para o nó destino. O nó pode ser uma aplicação, um sistema ou uma pessoa, na forma que tenha sido definido no processo de negócio (Samtani, et al., 2002). Um *broker* também dispõe de um repositório, para armazenar, buscar e obter essas informações.

Broker se define como um sistema capaz de gerenciar mensagens, fazendo a ligação entre os consumidores e fornecedores e normalmente são desenvolvidos sobre um *middleware*.

[2] *Middleware* é um tipo de processamento distribuído, baseado em componentes de *software* independentes ou serviços com interfaces e protocolos de programação padrão (ACM/IFIP/USENIX, 2009). Com o uso de um *middleware* pode ser realizado o compartilhamento de informações, bem como a interoperabilidade de aplicações entre diferentes plataformas de *hardware* e *software*.

Os *middlewares* fornecem a base da comunicação entre as aplicações, provendo também a persistência e garantia de entrega. No *middleware* existem muitas comunicações internas, onde cada componente compartilha dados com os demais.

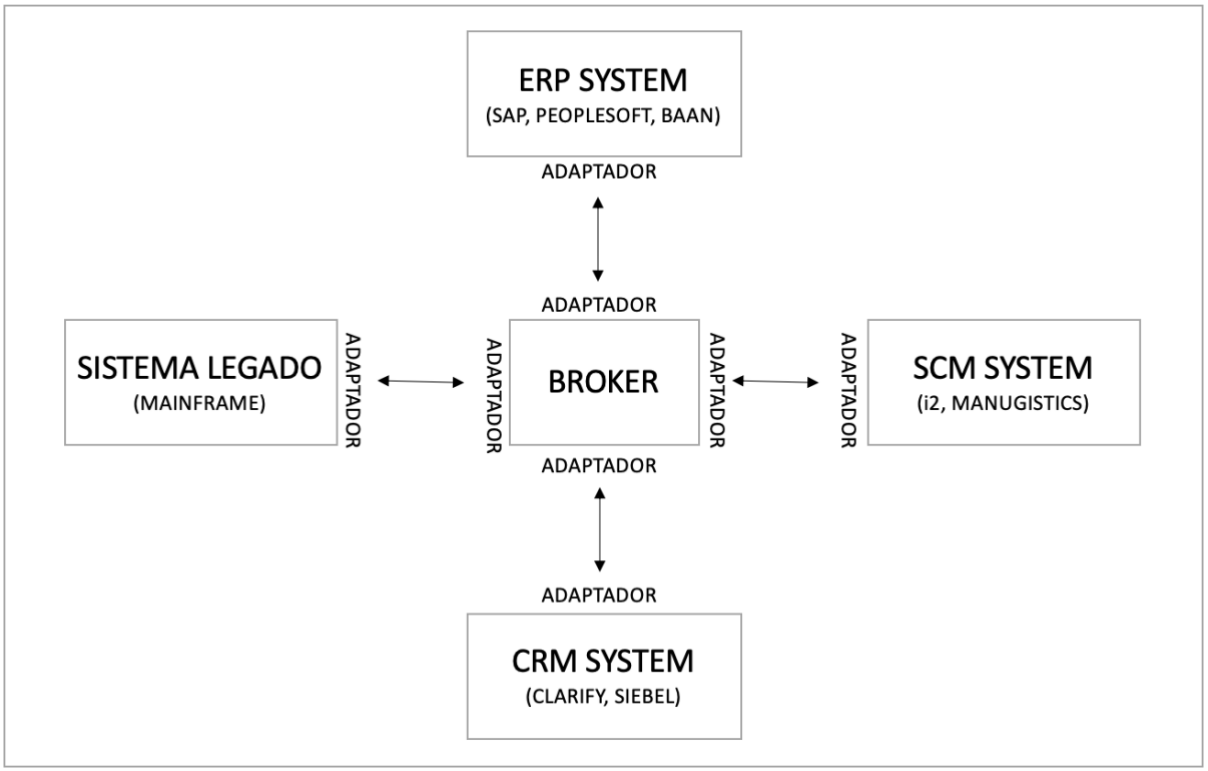
3.2.1. Arquiteturas

Segundo Samtani, et al. (2002), *brokers* podem ser baseados em duas arquiteturas físicas distintas: *hub-and-spoke* ou *message bus*. Existe ainda uma arquitetura derivada, conhecida como *multi-hub*, que conecta vários *brokers*, podendo cada um ser baseado em arquitetura *hub-and-spoke* ou *message bus*.

3.2.1.1. Arquitetura *Hub-and-Spoke*

Na arquitetura *Hub-and-Spoke*, existe um servidor central (*hub*) no qual todas as aplicações (*spoke*), internas e externas, estão conectadas (Alonso, et al., 2004). O nó central é o *broker*, que fornece os serviços de integração. A adição de uma nova aplicação é extremamente simples, sendo necessário que apenas seja conectada ao *hub*. A partir desse ponto, essa nova aplicação pode comunicar-se com as demais através do *broker*. A administração do *broker* também é simplificada nessa arquitetura, pois tudo é gerenciado de forma centralizada (Figura 4).

Figura 4 – Exemplo da Arquitetura *Hub-and-Spoke*



Fonte: Adaptado de Samtani, et al. (2002)

A natureza centralizada dessa arquitetura, entretanto, é também sua maior desvantagem (Samtani, et al., 2002). Se, por exemplo, a conexão de rede do *broker* cair, todo o sistema irá parar. Nenhuma aplicação, interna ou externa, será capaz de se comunicar com as demais. Esse tipo de situação pode ser evitada com o uso de uma solução em *cluster*, em que múltiplas instâncias do *broker* rodam em diferentes máquinas.

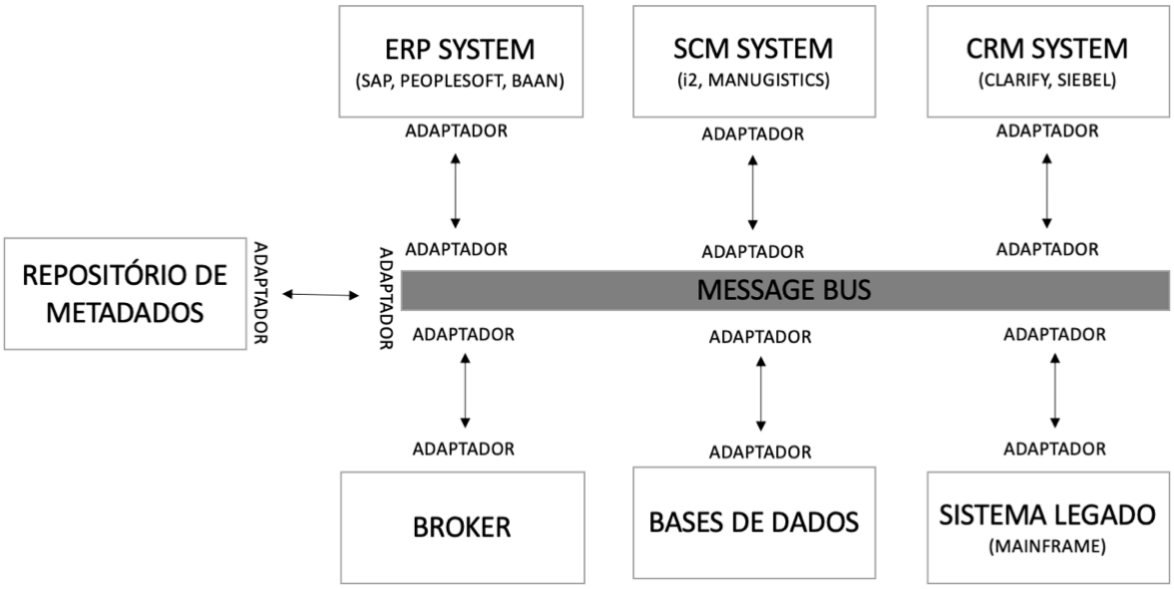
3.2.1.2. Arquitetura *Message Bus*

Na arquitetura *Message Bus*, o *message bus* forma o canal de comunicação no qual todas as aplicações são conectadas. Cada mensagem que é trocada entre as aplicações trafega pelo *bus* até o *broker*, que transforma, traduz e roteia a mensagem para a aplicação destinatária (Samtani, et al., 2002)

Nessa arquitetura, o *broker* deve ser visto como um serviço no *bus*, e não como um *hub*. Como a arquitetura é distribuída, ela oferece melhor escalabilidade e performance.

O processo de adição de uma nova aplicação também é simplificado. O *broker* deve fornecer um adaptador para a aplicação ou uma API para que a aplicação possa desenvolver o adaptador. Após a aplicação ser conectada no *bus*, ela pode se conectar com as aplicações que estão conectadas ao *bus*. Na Figura 5 é mostrada um exemplo da arquitetura *Message Bus*.

Figura 5 – Exemplo da Arquitetura *Message bus*

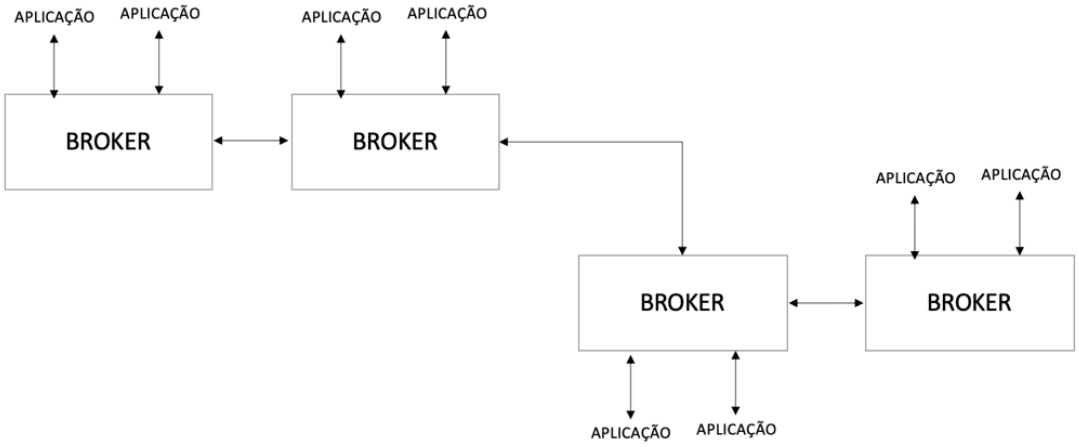


Fonte: Adaptado de Samtani, et al. (2002)

3.2.1.3. Arquitetura *Multi-Hub*

A arquitetura *multi-hub* é caracterizada pela presença de múltiplos *brokers*, cada um com várias aplicações conectadas a ele. A conexão entre os *brokers* é transparente para as aplicações que estão conectadas (Samtani, et al., 2002). Através da conexão com um *broker*, as aplicações são integradas como as outras aplicações conectadas aos diversos *brokers* no sistema. Na Figura 6 é mostrado um exemplo da arquitetura *Multi-Hub*.

Figura 6 – Exemplo da Arquitetura *Multi-Hub*



Fonte: Adaptado de Samtani, et al. (2002)

Esta arquitetura se faz muito útil em soluções escaláveis, em que múltiplas instâncias do mesmo *broker* podem ser instaladas em diferentes máquinas, sendo que mais instâncias podem ser instaladas ao passo que novas aplicações forem adicionadas ao sistema, para evitar que o sistema fique sobrecarregado.

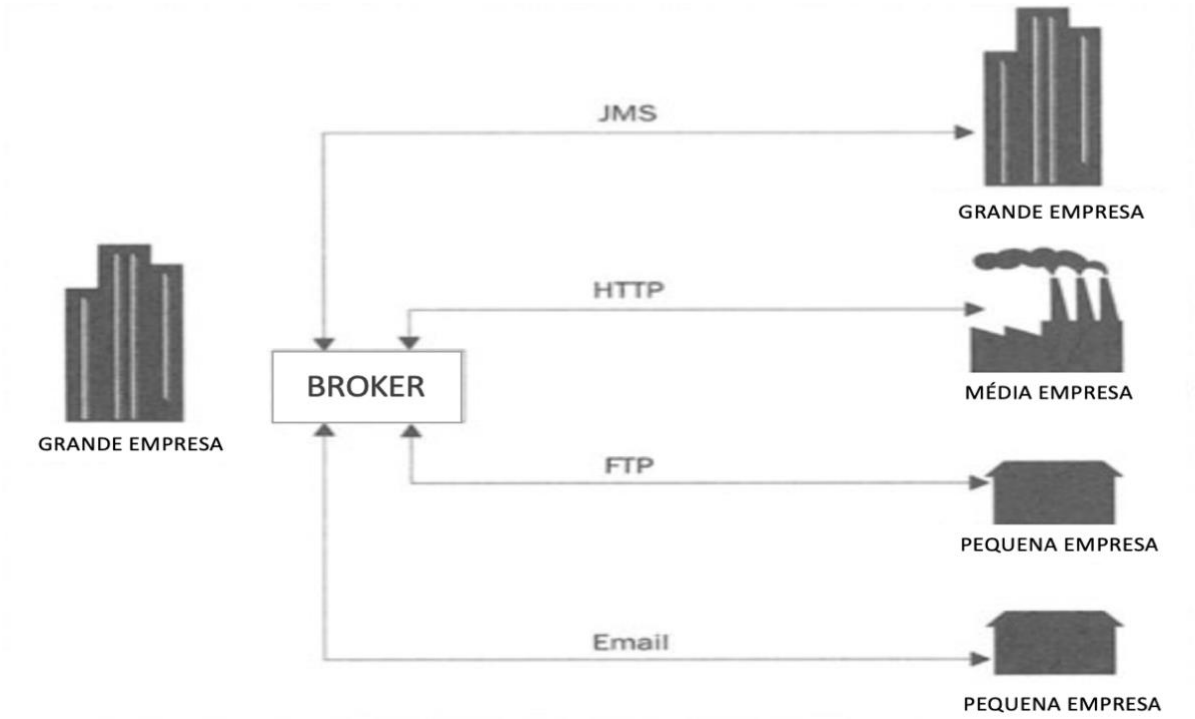
3.2.2. Serviços essenciais de um *Broker*

Ainda de acordo com Samtani, et al. (2002), um *broker* deve oferecer os seguintes serviços essenciais:

- **Possibilitar todos os tipos de integração:** O *broker* deve possibilitar integração entre Aplicações (A2A – Aplication-to-Aplication), entre negócio e consumidor (B2C – Business-to-Consumer) e entre negócios (B2B – Business-to-Business);
- **Interoperabilidade**: O *broker* deve fornecer interoperabilidade perfeita com as aplicações existentes, independente da linguagem de programação utilizada ou da plataforma em que é utilizada. Isso normalmente é obtido com uso de tecnologia baseadas em padrões abertos.

- **Arquitetura aberta:** A arquitetura do *broker* deve ser não-invasiva, escalável e aberta e que suporte arquiteturas de computação distribuídas;
- **Suporte para os protocolos de comunicação:** O *broker* deve ser capaz de suportar vários protocolos de comunicação B2B. Os mais comumente utilizados são: FTP, HTTP, HTTPS, EDI, WAP e TCP/IP, como mostrado na Figura 7;
- **Serviço de diretório:** O serviço de diretório funciona como um índice para todos os recursos e aplicações, juntamente com sua localização, protocolo de comunicação e uso. Ele fornece o ponto de entrada e busca pelas aplicações e recursos conectados ao *broker*;
- **Gerenciamento de troca entre parceiros:** o *Broker* deve fornecer um repositório de meta-dados, que pode armazenar as definições, preferencias, especificações técnicas (protocolo de comunicação, padrões XML, canais de distribuição e requerimentos de segurança) únicas de cada relação entre parceiros. Armazenar essas informações possibilita que serviços possam oferecer recursos personalizados para cada consumidor, fornecedor ou distribuidor;
- **Segurança:** *Brokers* devem proporcionar soluções seguras, usando criptografia, autenticação e certificados digitais;
- **Escalabilidade:** O *broker* também deve ser escalável, sendo capaz de suportar os serviços que estão cadastrados hoje e os que estão por vir;
- **Integridade transacional:** Por fim, o *broker* deve prover integridade nas transações, a cada passo, atividade e nó.

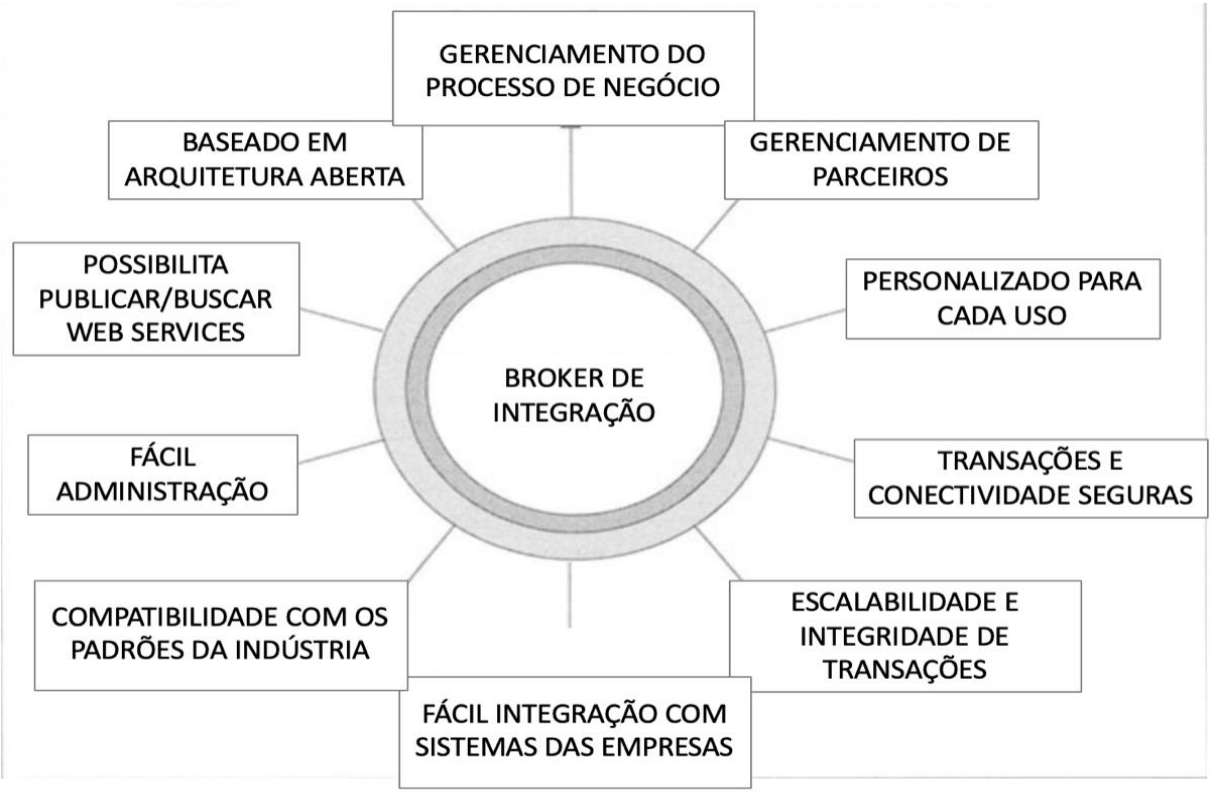
Figura 7 – Protocolos de comunicação que devem ser suportados em um *broker*



Fonte: Adaptado de Samtani, et al., (2002)

Na Figura 8 são mostradas as vantagens e as funções essenciais de um *broker*.

Figura 8 – Serviços essenciais de um *broker*



Fonte: Adaptado de Samtani, et al. (2002)

3.3. ONTOLOGIA

Do ponto de vista filosófico, ontologia pode ser definida como um ramo da metafísica que estuda os tipos de coisas que existem no mundo (Almeida, et al., 2003). O objetivo é fornecer um sistema de categorização para organizar a realidade (Breitman, 2005). A palavra ontologia deriva do grego *ontos*, que significa ser, e *logos* que significa palavra. Blackman (2005) afirma que ontologia é a parte da filosofia que lida com a natureza do ser.

No contexto da ciência da computação, uma ontologia permite definir primitivas de representação que podem modelar um determinado domínio de conhecimento (Gruber, 2009). Segundo Morais, et al. (2007), as principais áreas de utilização de ontologias são: recuperação de informação na internet, processamento de linguagem natural, gestão de conhecimento, web-semântica, educação, comunicação, formalização, representação de conhecimento e reutilização.

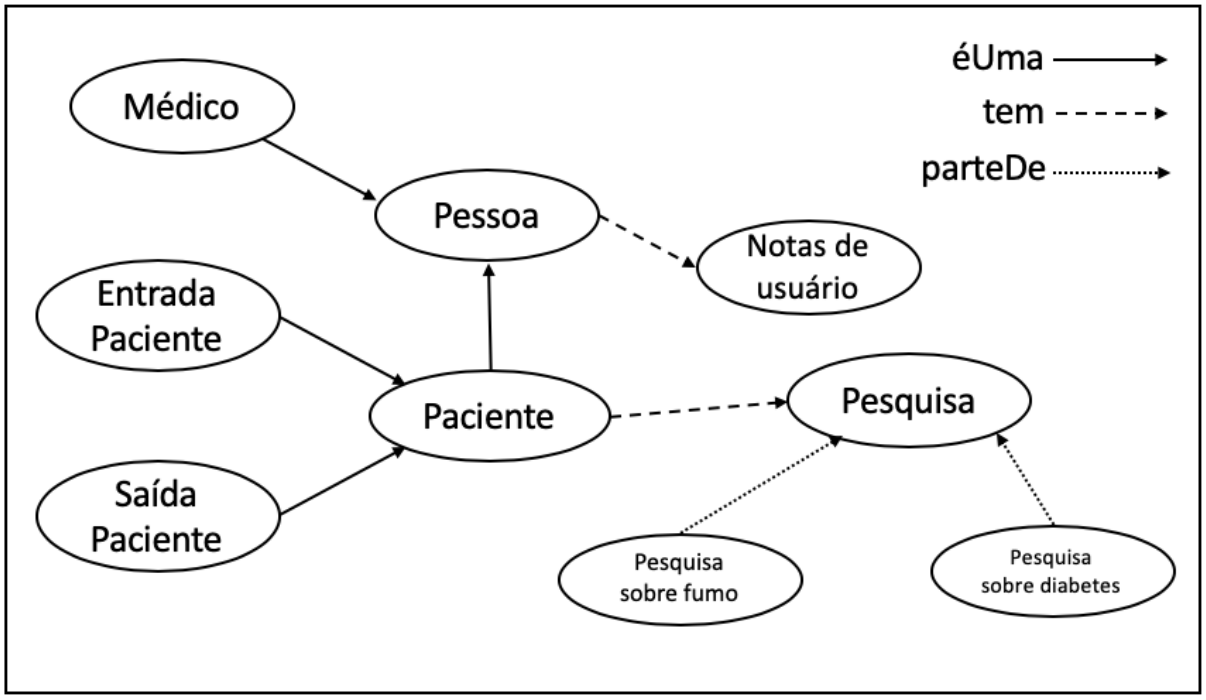
As principais vantagens de se utilizar ontologias, segundo Albarrak, et al. (2011), são: possibilitar a partilha e a interoperabilidade do conhecimento entre diferentes domínios; estruturar o domínio de forma que se permita a sua compreensão com maior clareza e objetividade; e permitir a reutilização de conceitos em diferentes domínios.

Morais, et al. (2007) classificam as ontologias quanto ao seu formalismo, aplicação, conteúdo ou função, em: ontologias genéricas, ontologias de domínio, ontologias de tarefa, ontologias de aplicação e ontologias de representação.

Na Figura 9 é ilustrado um exemplo de ontologia em que é definido que “Médico é uma Pessoa”, “Pessoa tem Notas de usuário” e “Pesquisa sobre diabetes é parte de uma Pesquisa”. Os relacionamentos entre os dados são baseados na percepção do mundo real, que consiste em três noções básicas: conjunto de entidades, conjunto de relações e atributos.

Uma ontologia é especificada com o uso de uma linguagem. As linguagens podem ser divididas em três tipos: linguagens de ontologias tradicionais, como Cycl, Ontolíngua ou F-Logic; linguagem padrão Web, como XML e RDF; e linguagens de ontologias *Web-Based,* como OIL, XOL e OWL (Morais, et al., 2007). O OWL é a linguagem recomendada pela W3C desde fevereiro de 2014. OWL é uma linguagem utilizada quando as informações contidas em documentos *Web* precisam ser processadas por aplicações e que seu conteúdo precisa ser entendido por humanos e computadores.

Figura 9 - Exemplo de Ontologia



Fonte: Autoria própria

3.4. WEB SERVICES

De acordo com W3C (2004), *web service* é um programa de computador projetado para suportar interações entre máquinas através de uma rede, fornecendo um padrão de interoperabilidade para que diferentes *softwares* possam consumir seus serviços. Tal interação é obtida por meio do envio e recebimento de dados, independentemente da plataforma utilizada para o desenvolvimento dos *softwares*.

As principais características de um *web service*, segundo Breitman (2005), são:

- Não depende de plataforma nem da linguagem de programação do solicitante;
- Não depende da localidade de onde o serviço está sendo solicitado;
- Não exige que o solicitante conheça a estrutura física do provedor de serviço.

Existem dois tipos básicos de *web service*, os *big web services* e os *web services* baseados na arquitetura *REST*.

Os *big web services* utilizam-se de uma pilha de tecnologias, conhecida como "WS-*", para prover interoperabilidade tanto por troca de mensagens quanto por chamadas a procedimentos remotos (Pautasso, et al., 2008).

Os *big web services* utilizam-se de SOAP e WSDL. O SOAP é um protocolo de comunicação de aplicativos que fornece uma estrutura padrão para empacotamento e troca de mensagens XML em ambientes descentralizados e distribuídos (W3C, 2004). Enquanto o WSDL consiste em uma linguagem de descrição para *web services* baseada em XML.

Os *web services* baseados na arquitetura *REST* constituem uma alternativa mais leve aos *web services* baseados em WS-*. O REST foi designado originalmente como um estilo arquitetônico para a construção de sistemas hipermídia distribuídos em larga escala (Pautasso, et al., 2008). O *REST* aproveita padrões já definidos na W3C/IETF, como: HTTP, XML e URI, de maneira que sua implementação é simplificada. Pautasso, et al. (2008) definem o estilo arquitetural REST baseado em quatro princípios básicos, são eles:

- Identificação de recursos por meio de URI;
- Os recursos são manipulados usando um conjunto fixo de quatro operações de criação, leitura, atualização e exclusão: PUT, GET, POST e DELETE;
- Os recursos são dissociados de suas representações para que seu conteúdo possa ser acessado em uma variedade de formatos (XML, HTML, Json, entre outros);
- As interações com estado são baseadas no conceito de transferência de estado explícita através de *hiperlinks*.

3.5. INTEROPERABILIDADE DE SISTEMAS DE SAÚDE

A interoperabilidade tem um valor importante no aumento da segurança do paciente ao permitir o acesso e a disponibilidade aos dados clínicos dos pacientes. Por outro lado, o acesso aos dados clínicos do paciente, em tempo real, permite ao sistema de saúde atender a um paciente a partir de qualquer local de atendimento do sistema melhorando, desta forma, a qualidade e a eficiência desse atendimento. Para conseguir a troca de informações da maneira mais natural possível, é imprescindível a adoção de padrões sobre os quais os diferentes sistemas de saúde possam coincidir. Por este motivo, surgem várias organizações que

pretendem unificar critérios em favor da interoperabilidade, tais como HL7 International[3], HIMSS[4] o NEMA[5] (Montón, 2017).

No Brasil, o Ministério da Saúde regulamentou por meio da Portaria 2.073 de 31 de agosto de 2011 o uso de padrões de interoperabilidade e informação em saúde para sistemas de informação em saúde no âmbito do Sistema Único de Saúde, nos níveis Municipal, Distrital, Estadual e Federal, e para os sistemas privados e do setor de saúde suplementar. A portaria define o uso dos padrões *PIX – Patient Identifier Cross-referencing HL7 V3 (PIXV3)* – que possibilita que múltiplas aplicações distribuídas possam correlacionar informações sobre um único paciente a partir de fontes que conhecem este paciente por diferentes identificadores, bem como o *PDQ – Patient Demographics Query HL7 V3 (PDQV3)* – possibilita que múltiplas aplicações distribuídas consultem os dados demográficos de pessoas armazenados num servidor central (no caso Brasileiro, o servidor do Cadastro Nacional de Usuários de SUS) a partir de um conjunto de dados demográficos pré-definidos (Ministério da Saúde, 2020).

3.6. TRABALHOS CORRELATOS

Neste tópico são apresentados trabalhos encontrados na literatura que tratam sobre integração de sistemas e o uso de ontologias, *web services* e *brokers*.

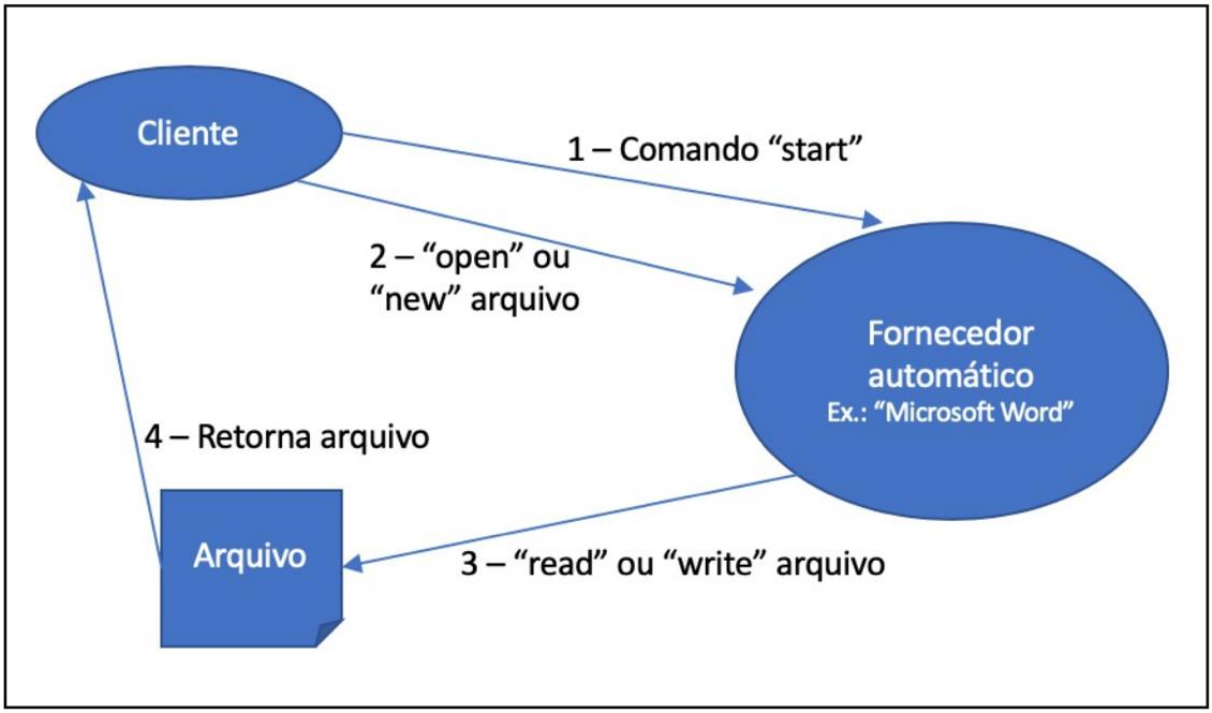
Segundo Fernandez (2004), os pontos de integração são divididos em: pontos permanentes, que são as interfaces que permanecerão ativas no ambiente de produção; pontos de integração temporária, pontos provisórios, relativos a projetos executados por etapas e vão sendo eliminados no decorrer do projeto; e múltiplos pontos centrais de integração.

Santana, et al. (2009) apresentam um estudo de caso sobre a integração de sistemas por meio de *web services*. No trabalho foi realizada uma avaliação das perspectivas de integração, identificando os pontos de integração que são os tipos de interfaces que estarão ativas na pós-implantação ou que simplesmente realizarão o papel de integração das transações do ambiente de produção.

[3] https://www.hl7.org
[4] https://www.himss.org
[5] https://www.nema.org

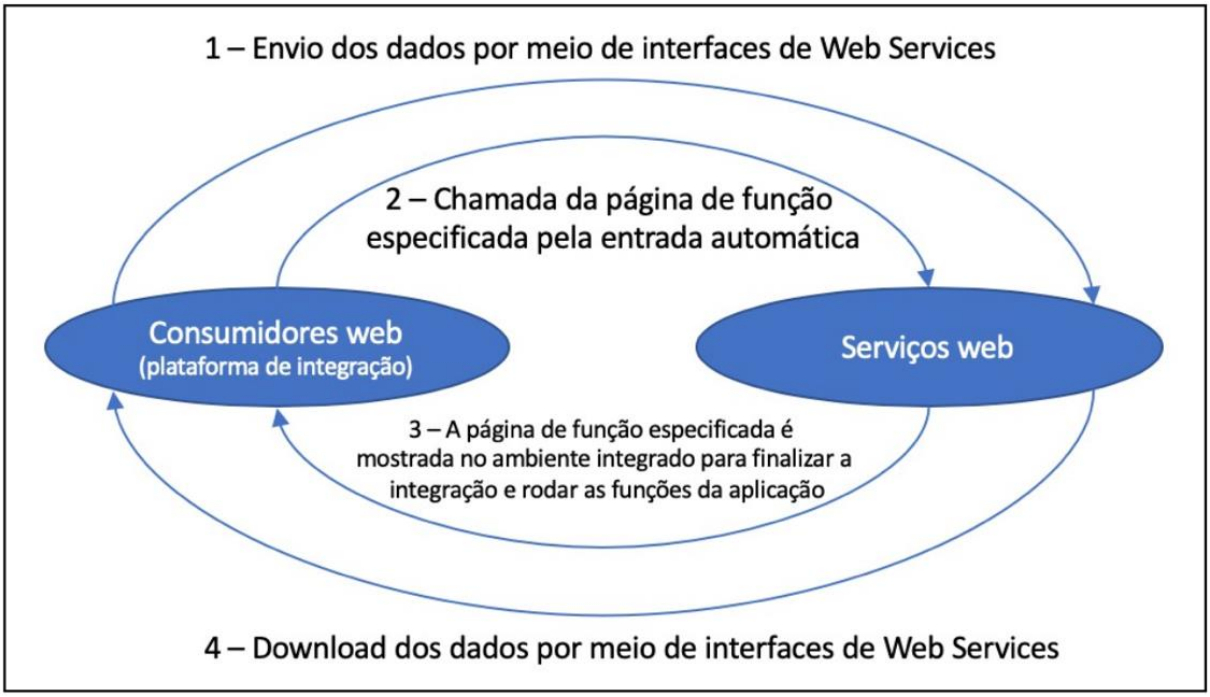
Xu, et al. (2016) propõem uma arquitetura unificada de integração de aplicações e uma tecnologia de automação de integração de sistemas baseada em *web services*. Tal tecnologia é aplicada em um projeto real de integração de sistemas entre várias plataformas. Os autores utilizam a ideia do OLE automation. OLE automation é um mecanismo para que os aplicativos do Windows possam manipular outros programas. Definindo-se um programa cliente e um servidor, pode-se definir uma série de comandos como “start”, “stop” e “new”, e envia-los a outros programas, como o Microsoft Word. O processo geral do OLE automation é mostrado na Figura 10. Estes comandos agem como uma aplicação chamando uma função ou procedimento. Utilizando a ideia do OLE automation, é possível realizar a interação entre duas aplicações Web, como mostrado na Figura 11. Os resultados obtidos com o trabalho mostram que sistemas de diferentes plataformas podem ser efetivamente integrados de forma a se obter um sistema íntegro, paralelo e facilmente acessível.

Figura 10 – Processo automático do OLE automation



Fonte: XU, et al., (2016)

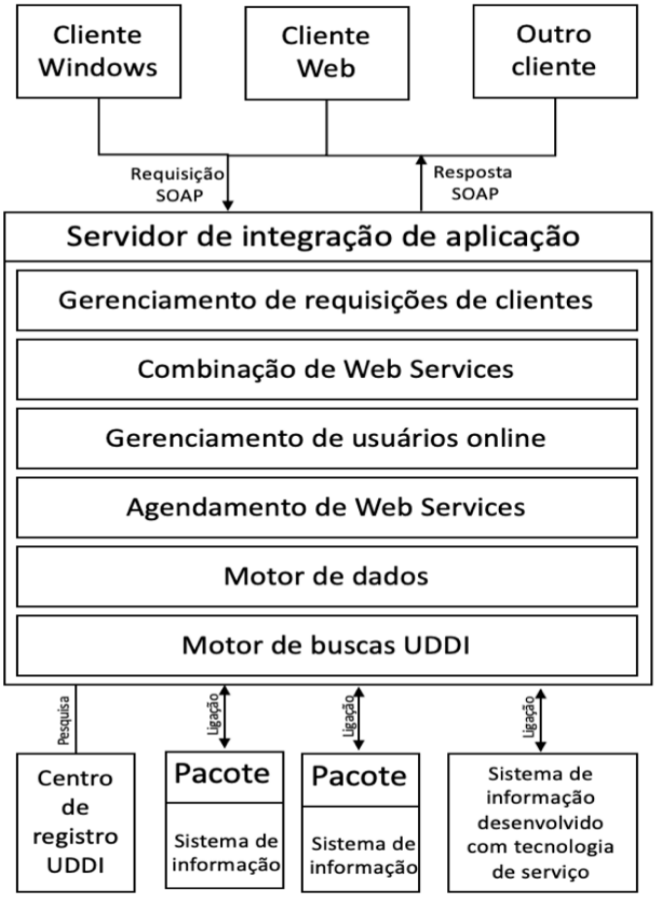
Figura 11 – Interação entre duas aplicações Web



Fonte: XU, et al., (2016)

Li (2010) utiliza a tecnologia de *web services* para obter compartilhamento de recurso de maneira flexível, segura e coordenada. Os sistemas são encapsulados baseados em uma tecnologia multiagente, possibilitando o reuso de funcionalidades de diferentes sistemas, de forma que vários provedores de serviço e conteúdo possam contribuir para o desenvolvimento de uma plataforma integrada de sistemas da informação. Devido ao encapsulamento dos sistemas de informação, a estabilidade e a segurança dos sistemas são garantidas na plataforma de integração. Os sistemas são definidos como *web service* e depois são registrados no centro de registro UDDI. Ao receber requisições de diferentes usuários, a aplicação irá pesquisar no centro de registro UDDI, e irá encontrar o *web service* correto para responder a requisição. O framework de integração é mostrado na Figura 12. A troca de mensagens entre as diferentes aplicações, rodando em diferentes sistemas operacionais, é feita em XML. No servidor de integração das aplicações estão presentes diversos módulos-núcleo que podem ser chamados para responder as diversas requisições dos usuários. As descrições desses módulos são mostradas na Tabela 1.

Figura 12 – Framework de integração de sistemas baseado em web services



Fonte: Li (2010)

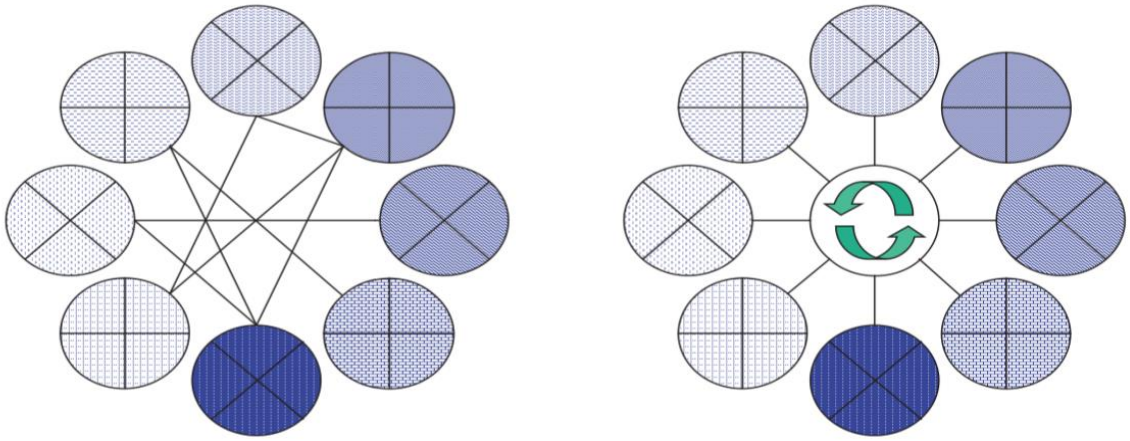
Tabela 1 - Módulos-núcleo do servidor de integração

Web_Service_Combination	**Agendamento Web_Service**	**Mecanismo de pesquisa UDDI**
Buscar/Listar Web Services de relações Definição do fluxo do Web Service Buscar/Listar Serviços alternativos Definição do fluxo dos serviços alternativos	Analise do serviço Verificação do serviço Agendamento do serviço Controle do fluxo de operação Controle de transação Monitoramento do serviço	Web Service query Legação Web Service Criação do agente de Web Service Web Service de ligação Gerenciamento da informação UDDI
Gerenciamento da requisição de cliente	**Mecanismo de dados**	**Gerenciamento de usuário online**
Verificação do cliente Validação da requisição Lista de criação de requisição do cliente Lista de atualização de requisição do cliente Lista de gerenciamento de requisição do cliente Retorno da mensagem de resposta	Criação do dicionário de dados Gerenciamento da informação global Conversão do formato de dados Conversão do modelo de dados Compressão de dados Descompressão de dados	Criação da informação de usuário online Atualização da informação de usuários online Logout de usuário online Verificação de usuários online Gerenciar lista de usuários

Fonte: Li, (2010)

Süß et. al. (2001) apresentam uma definição de um *broker* de mensagem, enumerando critérios críticos de uma empresa e descrevendo uma arquitetura de referência para atender tais critérios. É feita uma comparação entre o *broker* e *middlewares* como CORBA[6] e MOM/DAD[7], em que são apontadas as vantagens e desvantagens do uso de um *broker*. Ao final, uma arquitetura de referência em Java, XML e XSL é descrita, que permite atender os critérios de configuração, execução e extensão definidos. Os autores afirmam que o uso de *brokers* é mais indicado para problemas N_2, pois suportam mudanças que modifique ou criem novas interfaces dinamicamente, sem que o serviço precise ser interrompido. Para um *broker,* uma interface é simplesmente um tipo de documento de interesse comum para o negócio. Cada sistema atua como um produtor de mensagem, independente do consumidor. Isso permite que cada sistema inicie com um conjunto de mensagens bem definido que sejam importantes para o negócio. Outros sistemas, ao buscarem alguma informação, podem busca-las junto ao *broker.* O *broker,* ao aplicar regras de transformação, pode fornecer saídas formatadas para as necessidades de cada um. Na Figura 13 é apresentada a integração para problemas N_2 e a solução utilizando um *broker*.

Figura 13 – Integração para problemas N_2 (esquerda) e a solução utilizando um *broker* (direita)



Fonte: Süß et. al. (2001)

6 CORBA (*Common Object Request Broker Architecture*) é a arquitetura padrão criada pelo Object Management Group para estabelecer e simplificar a troca de dados entre sistemas distribuídos heterogêneos. CORBA foi projetado com o objetivo de oferecer um ambiente de desenvolvimento com transparência de localização, o que permite que o acesso aos métodos oferecidos pela interface de um objeto sejam invocados da mesma maneira, não importando se o objeto CORBA se encontra no espaço de endereçamento local ou em um ambiente remoto. Além disso, o padrão CORBA pode interoperar com múltiplas linguagens de programação, permitindo que clientes e servidores sejam desenvolvidos em linguagens diferentes (Mello, et al., 2006).

7 *Message Oriented Middleware* (MOM) trata-se de uma infraestrutura de software cliente/servidor que cria uma camada entre as aplicações de alto nível e as plataformas onde estão instaladas, substituindo a comunicação direta entre aplicações por um sistema de troca de mensagens. Tipicamente, uma implementação MOM fornece um conjunto de API's capaz de funcionar com um número relativamente vasto de plataformas e redes que varia entre implementações (Gonçalves, 2010).

Armitage, et al, (2009) apresenta uma revisão sistemática sobre a Integração de Sistemas de Saúde. A revisão sistemática destaca alguns modelos, ferramentas de medição e resultados esperados da integração, que podem ajudar na modelagem e implementação de sistemas de saúde integrados. A revisão também mostrou algumas lacunas significantes, especialmente quando se trata de ferramentas padronizadas para medir as respostas da integração no que se refere à eficiência e eficácia tanto a nível de sistema, como de programas, provedores e clientes.

Diferente dos trabalhos relacionados encontrados, este trabalho apresenta o projeto e desenvolvimento de um *broker* para integração de *web services* no domínio da saúde. O *broker* é desenvolvido com base em uma ontologia em que se são descritos todos serviços e recursos que estão disponíveis.

A Tabela 2 abaixo, apresenta uma comparação entre os trabalhos relacionados.

Tabela 2 - Comparação entre trabalho relacionados

	Integração de sistemas	**Web services**	**Broker**
Santana, et al, (2009)	X	X	
Xu, et al. (2016)	X	X	
Li, (2010)		X	X
Süß, et al. (2001)			X
Armitage, et al, (2009)	X		
BORIS	X	X	X

Fonte: Autoria própria

4. ASPECTOS DE IMPLEMENTAÇÃO

Neste capítulo são apresentadas as informações relacionadas aos aspectos de implementação do *broker*. Na Seção 4.1 é apresentada a arquitetura, o funcionamento e as funcionalidades do componente *broker*. Na Seção 4.2 é mostrada construção e descrição da ontologia desenvolvida.

4.1. BROKER

Neste trabalho é apresentado o projeto e implementação de um *broker* que possibilita a integração entre aplicações de saúde. O *broker* foi desenvolvido utilizando como base uma ontologia, na qual são descritas as aplicações que desejam compartilhar seus recursos e uma interface *web*, por meio da qual os consumidores podem ver quais recursos estão disponíveis e detalhes sobre cada um. A arquitetura REST é utilizada para prover os serviços, pois garante a alta coesão e baixo acoplamento entre serviços. A escolha pela arquitetura REST se deu pelo fato de os serviços poderem ser acessados por meio do protocolo HTTP.

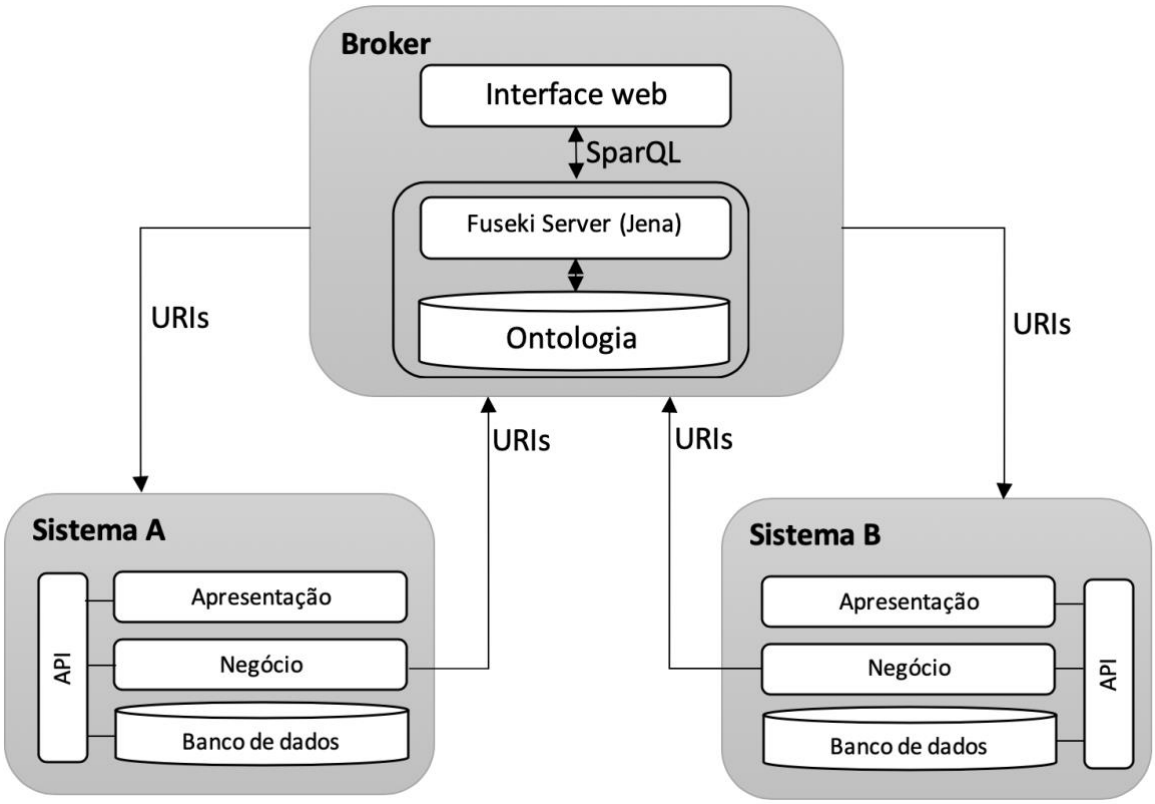
4.1.1. Arquitetura do *broker*

O *broker* é composto por dois módulos internos: interface web, na qual os interessados acessam para visualizar quais sistemas e recursos estão disponíveis e os detalhes de acesso de cada um destes, como *links*, parâmetros, retornos e formato do retorno; e uma ontologia, na qual são descritos os sistemas cadastrados, assim como recursos e informações necessárias para o acesso. A interface w*eb* realiza consultas SparQL em um *end-point* que é executado pelo *Fuseki Server (Jena)*[8], no qual a ontologia se mantém ativa, escutando uma porta definida pelo

[8] Apache Jena Fuseki é um servidor SPARQL. O Fuseki roda como um serviço do Sistema operacional. O Fuseki também provê segurança e tem uma interface de usuário para monitoração e administração.

servidor. O *broker* acessa as URIs dos recursos e armazena na ontologia, ao passo que os sistemas consumidores, em sua camada de negócio, acessam as URIs armazenadas no *broker*. Na Figura 14 ilustra-se a arquitetura do *broker*.

Figura 14 – Arquitetura do *broker*



Fonte: Autoria própria

4.1.2. Funcionalidades

Algumas funcionalidades que o *broker* deveria prover foram identificadas e selecionadas. Na Tabela 3 são elencadas tais funcionalidades.

Tabela 3 - Funcionalidades do *broker*

Funcionalidade	Método	Descrição
Listar serviços	GetListServicos	Por meio desta funcionalidade, é possível saber quais são os serviços que estão cadastrados no *broker*.
Listar recursos de um serviço	PostListRecursos	Ao informar ao *broker* um determinado serviço, este irá responder com os recursos que o compõem.
Listar detalhes de um recurso	PostListDetalhes	O *broker*, ao receber um recurso, lista os detalhes relacionados a este recurso.

Fonte: Autoria própria

Cabe ressaltar que outras funcionalidades podem, posteriormente, serem adicionadas ao *broker*, a fim de enriquecer a integração.

4.2. ONTOLOGIA

A ontologia foi construída com o auxílio da ferramenta Protégé. Seu objetivo é descrever os serviços e seus recursos que estejam disponíveis no *broker*. A seguir é apresentada a construção e descrição da ontologia.

4.2.1. Definição do Domínio e Escopo

O Domínio da ontologia é referente a sistemas/aplicações no âmbito da saúde, fornecendo o mapeamento dos recursos que estão disponíveis, facilitando a integração entre aplicações.

O responsável pela adição e manutenção da ontologia é o administrador, que poderá incluir ou excluir recursos de outros sistemas interessados.

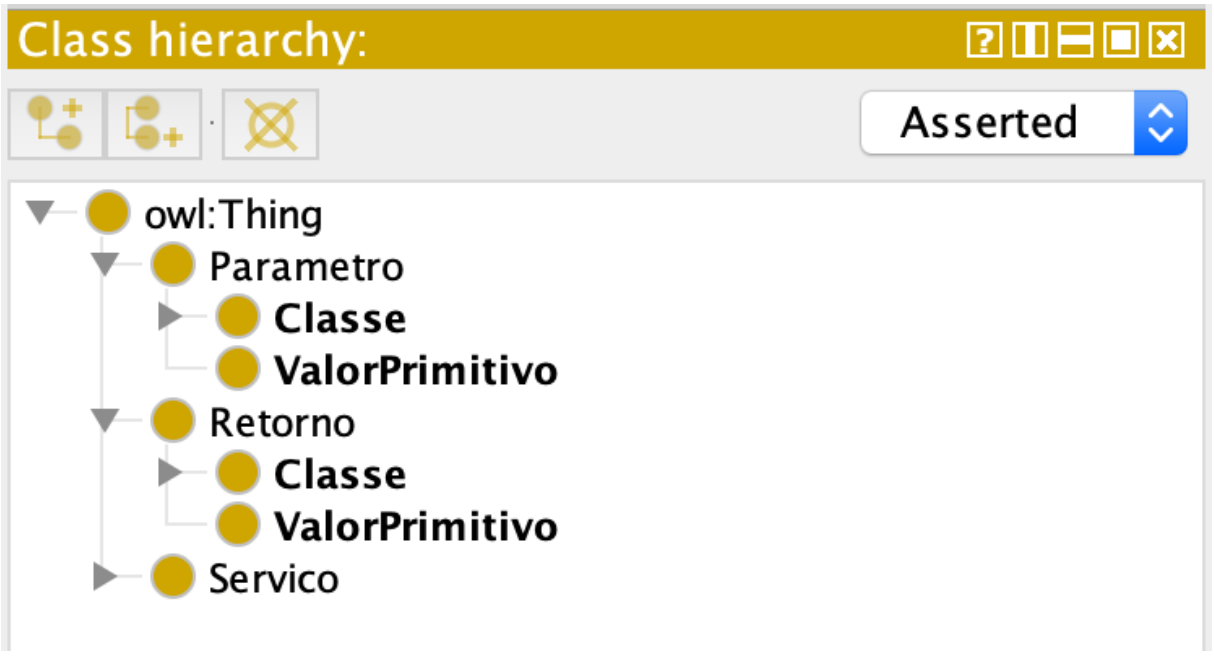
Umas das formas de se determinar o escopo de uma ontologia é esboçar uma lista de questões que a ontologia precisa ter competência de responder (Gruninger, et al., 1995). Com isso, foram elaboradas as seguintes questões:

1. Quais os serviços/sistemas que estão cadastrados no *broker*?
2. Quais os recursos de um determinado serviço cadastrado no *broker*?
3. Quais os parâmetros, retornos, formatos, *links* de acesso de um determinado recurso cadastrado no *broker*?

4.2.2. Definição da Taxonomia (Classes e Hierarquia)

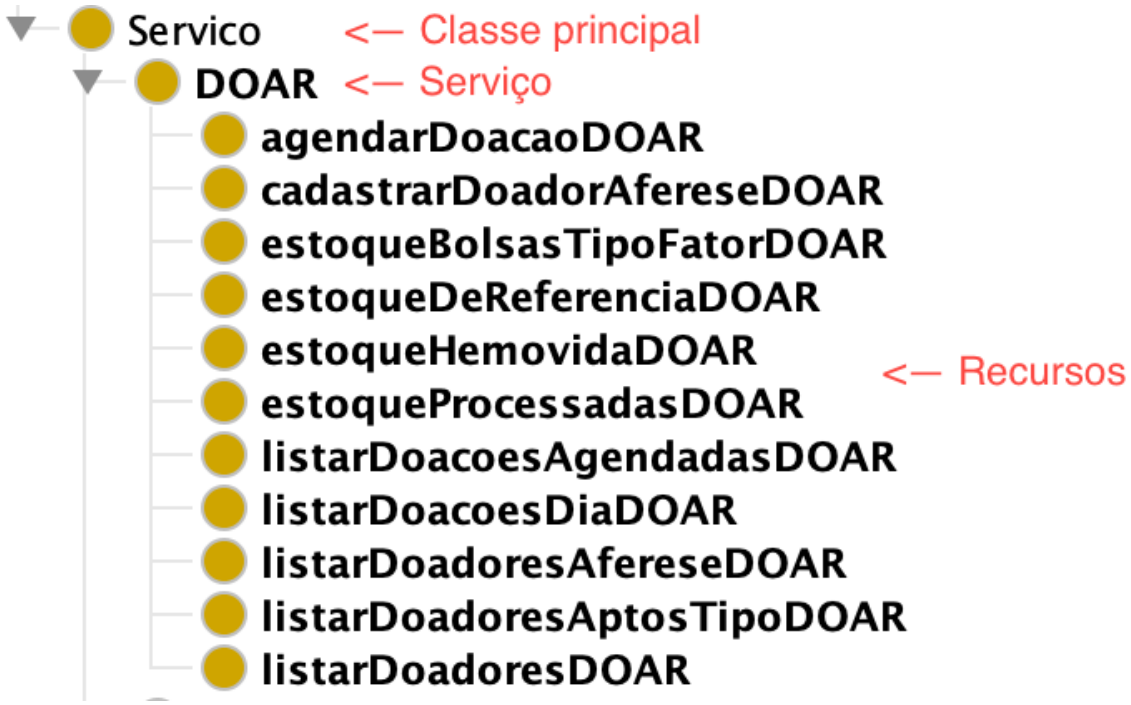
A ontologia definida é composta de três classes principais: “Parametro”, “Retorno” e “Servico”, como mostrado na Figura 15. Na classe “Servico” são criadas subclasses referentes a cada serviço que é cadastrado no *broker*. Para cada recurso de um serviço, é criada uma subclasse dentro da classe referente ao serviço, como mostrado na Figura 16.

Figura 15 – Classes principais da ontologia



Fonte: Autoria própria

Figura 16 – Recursos de um serviço

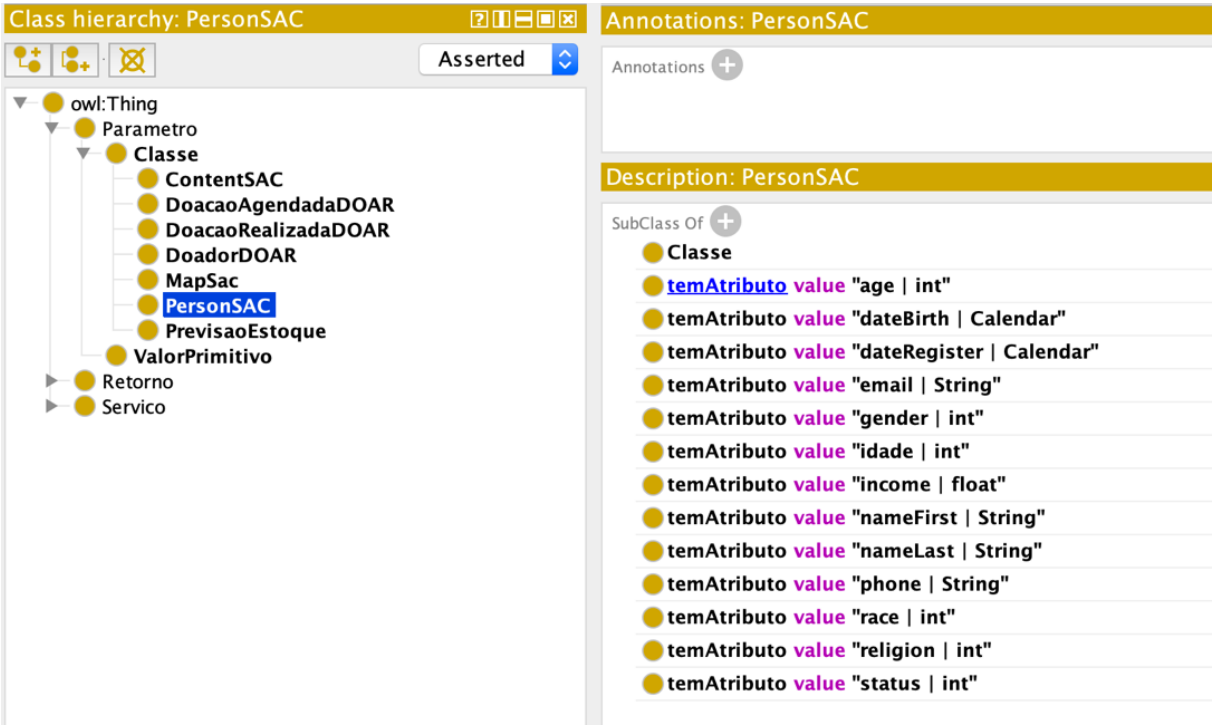


Fonte: Autoria própria

Na classe “Parametro” são definidos os tipos que cada recurso de um serviço utiliza como parâmetro de entrada. A classe “Parametro” é subdividida em duas subclasses: “Classe” e “ValorPrimitivo”. Quando um recurso utiliza como parâmetro um valor primitivo (inteiro, real, string), é definido que este utiliza como parâmetro a subclasse “ValorPrimitivo”, nesta são especificados indivíduos que correspondem a cada tipo primitivo. Caso algum recurso utilize um tipo não primitivo como parâmetro, por exemplo tipo Pessoa (composto de: *string* nome, *int* idade, *string* telefone), esse parâmetro é definido como uma subclasse criada em “Classe”, referente a este tipo utilizado, como mostrado na Figura 17.

A classe “Retorno” segue os mesmos princípios da classe “Parâmetro”, porém relativos aos valores de retorno de cada recurso.

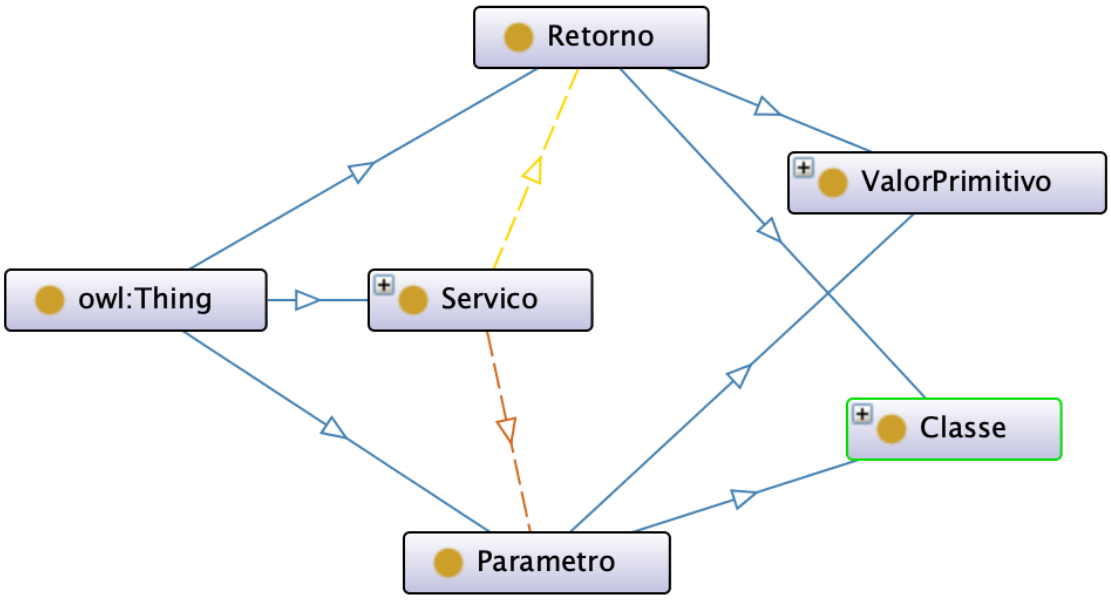
Figura 17 – Definição de um tipo não primitivo



Fonte: Autoria própria

Na Figura 18 é mostrada a taxonomia básica da ontologia.

Figura 18 – Taxonomia da ontologia

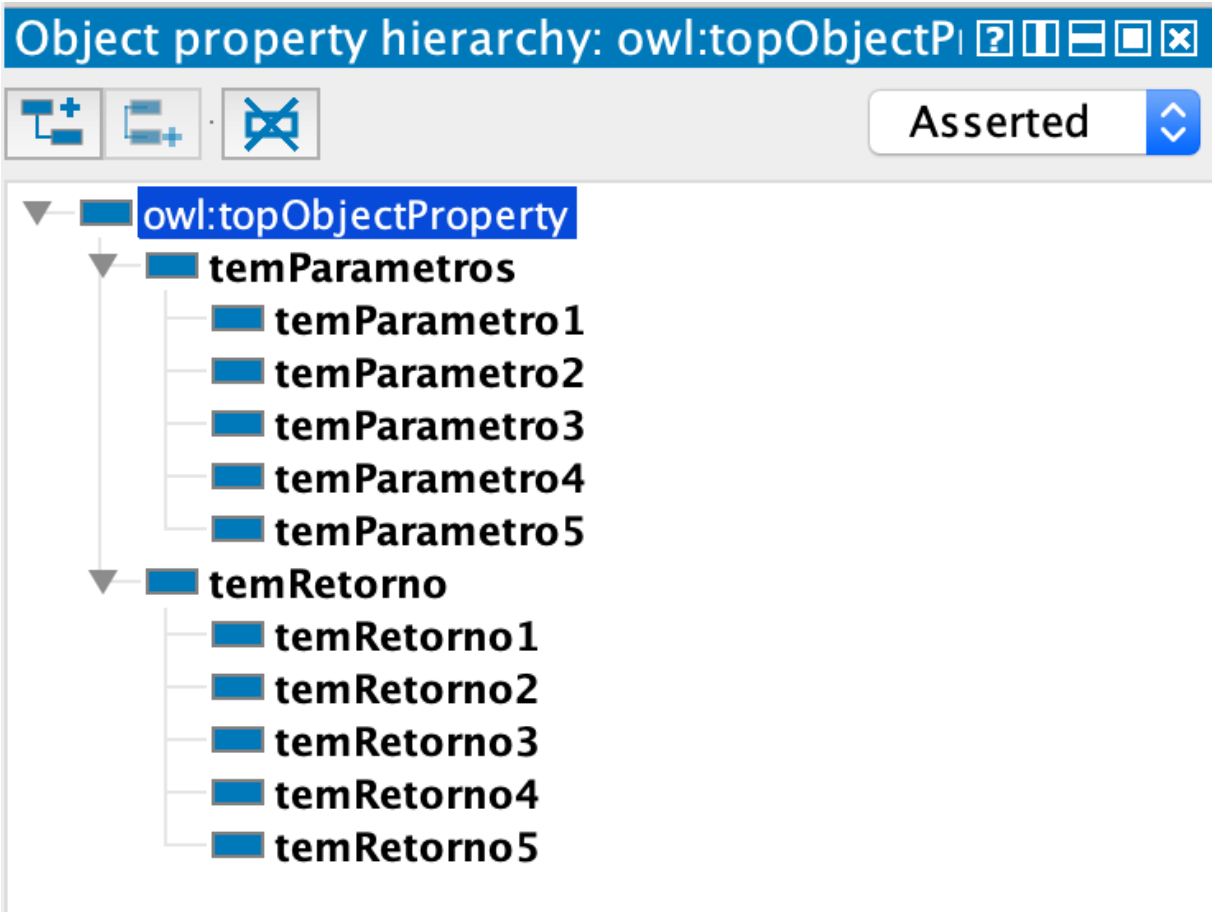


Fonte: Autoria própria

4.2.3. Definição das propriedades dos Objetos

Após definir a taxonomia da ontologia, o próximo passo foi definir as propriedades dos objetos. Foram definidas duas propriedades gerais: temParametros e temRetorno. Cada uma dessas propriedades podem ser subdivida em subpropriedades (temParametro1, temParametro2, temRetorno1, temRetorno2), a depender da quantidade de parâmetros e retornos que cada recurso define. Na Figura 19 são mostradas as propriedades dos objetos definidas na ontologia.

Figura 19 – Propriedades dos objetos



Fonte: Autoria própria

4.2.4. Definição das propriedades de dados

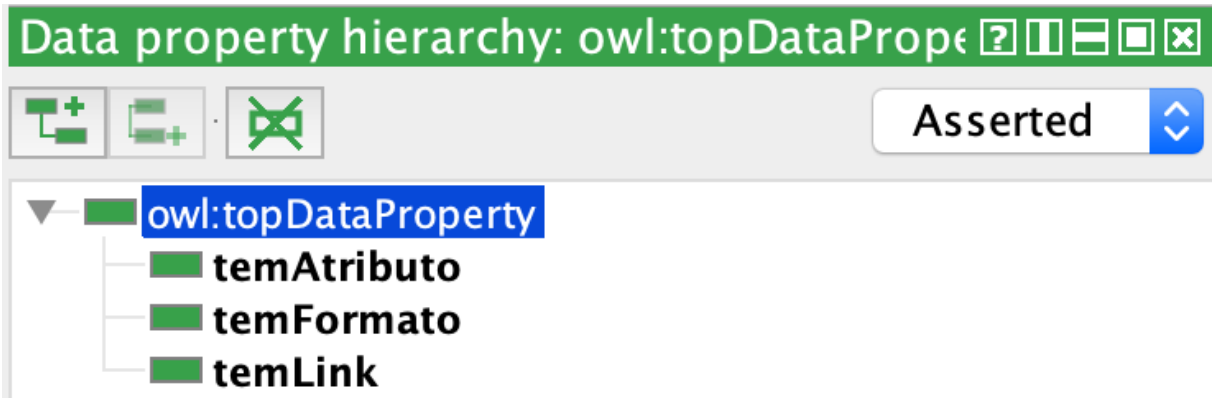
No próximo passo foram definidas as propriedades de dados da ontologia. Foram definidos dois dados básicos que devem ser informados no cadastro de cada recurso: temFormato e temLink. A propriedade temFormato define o formato em que o recurso responde a uma requisição (JSON, XML, entre outros). Já a propriedade temLink define o *link* de acesso desse recurso.

A propriedade temAtributo é utilizada quando um determinado recurso tem como parâmetro ou retorno um tipo específico de dados, por exemplo: tipo Pessoa (composto de: *string* nome, *int* idade, *string* telefone). Um exemplo de uso é mostrado na Figura 17.

As propriedades de dados definidas são mostradas na Figura 20.

Na Tabela 4 são apresentadas as propriedades da ontologia, definindo seu tipo, domínio e imagem. As propriedades filhas têm as mesmas definições que as propriedades mães.

Figura 20 – Propriedades de dados



Fonte: Autoria própria

Tabela 4 - Propriedades da ontologia

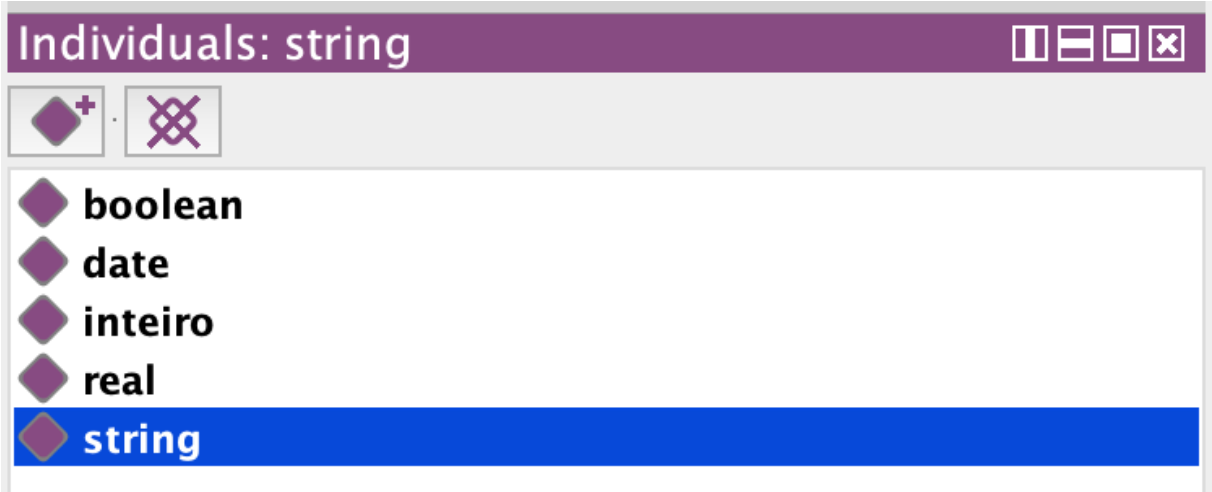
Propriedade	Tipo	Domain	Range
temParametros	Object Property	Servico	Parametro
temRetorno	Object Property	Servico	Retorno
temAtributo	Data Property	Parametro	string
temFormato	Data Property	Parametro ou Retorno	string
temLink	Data Property	Servico	string

Fonte: Autoria própria

4.2.5. Criação dos indivíduos

Na ontologia definida, os indivíduos correspondem aos tipos de dados primitivos definidos para serem usados como parâmetro e retorno de um recurso. Na Figura 21 são mostrados os indivíduos criados.

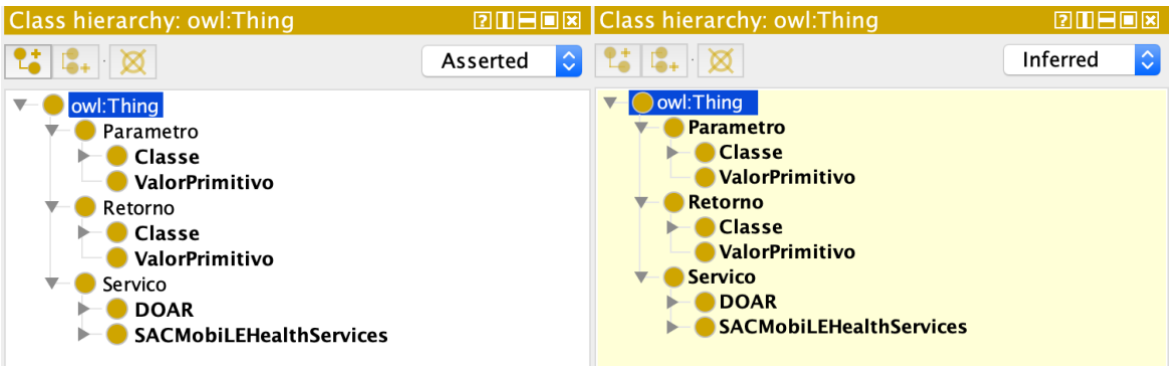
Figura 21 – Indivíduos criados



Fonte: Autoria própria

Ao se executar o *reasoner*[9], notou-se que a hierarquia de classes inferida era igual a hierarquia de classes definida. Isto indica que a ontologia está consistente não apresentando nenhum erro nas definições que causem comportamento anormal da mesma. Na Figura 22 é ilustrada a hierarquia de classes definida e inferida.

Figura 22 – Hierarquia de classes definida e inferida



Fonte: Autoria própria

[9] Ferramenta disponível no Protégé utilizada para inferência de classes e indivíduos

5. ESTUDO DE CASO

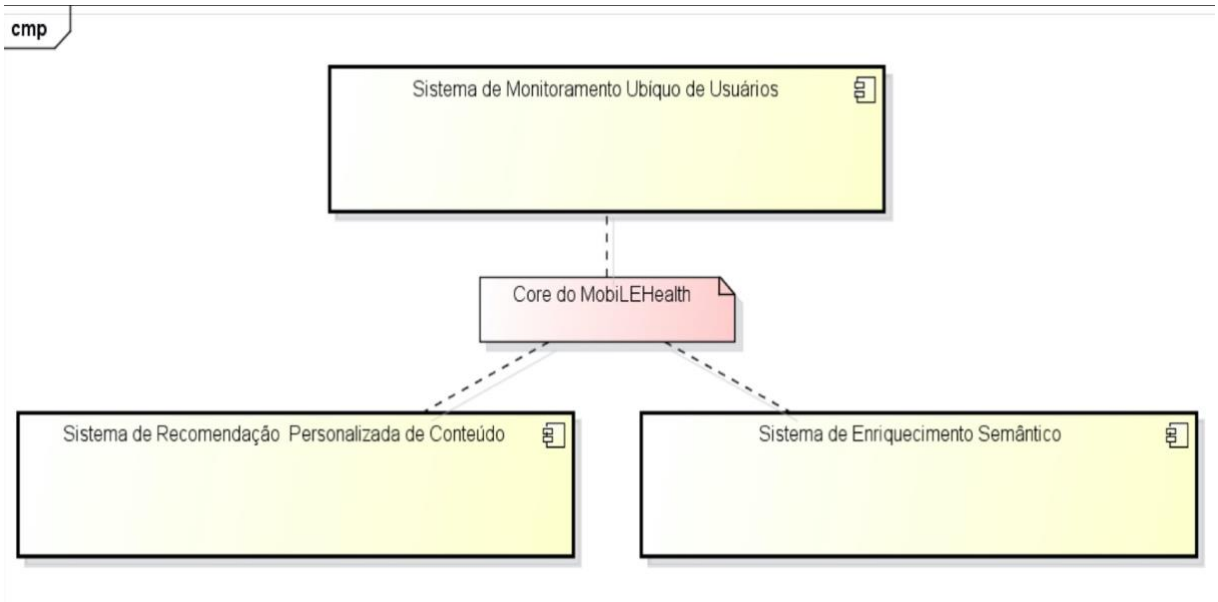
Para validar a solução proposta por esse trabalho, foram adicionados dois exemplos de aplicações, o MobiLEHealth e o DOAR. As seções a seguir apresentam as aplicações utilizadas como estudo de caso e o detalhamento das análises realizadas.

5.1. MOBILEHEALTH

O MobiLEHealth – *Mobile Learning Environment for Health* (Sombra, 2015) é um ambiente de aprendizagem ubíqua no contexto da Saúde 4.0, possibilitando o acesso, a visualização e o fornecimento de conteúdo independente do local e horário. O MobiLEHealth se utiliza de tecnologias e ferramentas da Web 4.0, como redes sociais, registro pessoais de saúde, *blogs*, vídeos, etc., como forma de monitorar, através da experiência do usuário, informações referentes à saúde e, pela combinação desses dados e informações, fornecer conteúdo personalizado. O ambiente é destinado a pessoas com doenças crônicas e fornece aos usuários uma interface de interação com conteúdos Web e mídias sociais, monitorando e capturando as informações geradas e apresentado os conteúdos recomendados às suas necessidades.

Originalmente o MobiLEHealth consiste em um conjunto de três sistemas (Sistema de Enriquecimento Semântico; Sistema de Recomendação Híbrida; e Sistema de Monitoramento Ubíquo de Usuários) (Sombra, 2015). Tais sistemas atuam de forma independente, porém de modo integrado. Na Figura 23 é mostrado o conjunto dos três sistemas que compõem o ambiente MobiLEHealth.

Figura 23 – Conjunto de sistemas que compõem o MobiLEHealth



Fonte: Sombra (2015)

O Sistema de Enriquecimento Semântico (SES) tem a função de enriquecer semanticamente o perfil de usuário com os dados coletados pelo sistema de Monitoramento Ubíquo do Usuário, que relaciona estas informações a domínios de conhecimentos modelados em ontologia específica da doença crônica em questão e gera um perfil semântico do usuário (Moreira, et al., 2014).

O Sistema de Recomendação Personalizada de Conteúdo (SRPC) tem a finalidade de recomendar conteúdos personalizados a partir dos interesses do usuário por meio de perfis semânticos, gerados a partir de sua experiência cotidiana e informações referentes à sua saúde (Costa, et al., 2014).

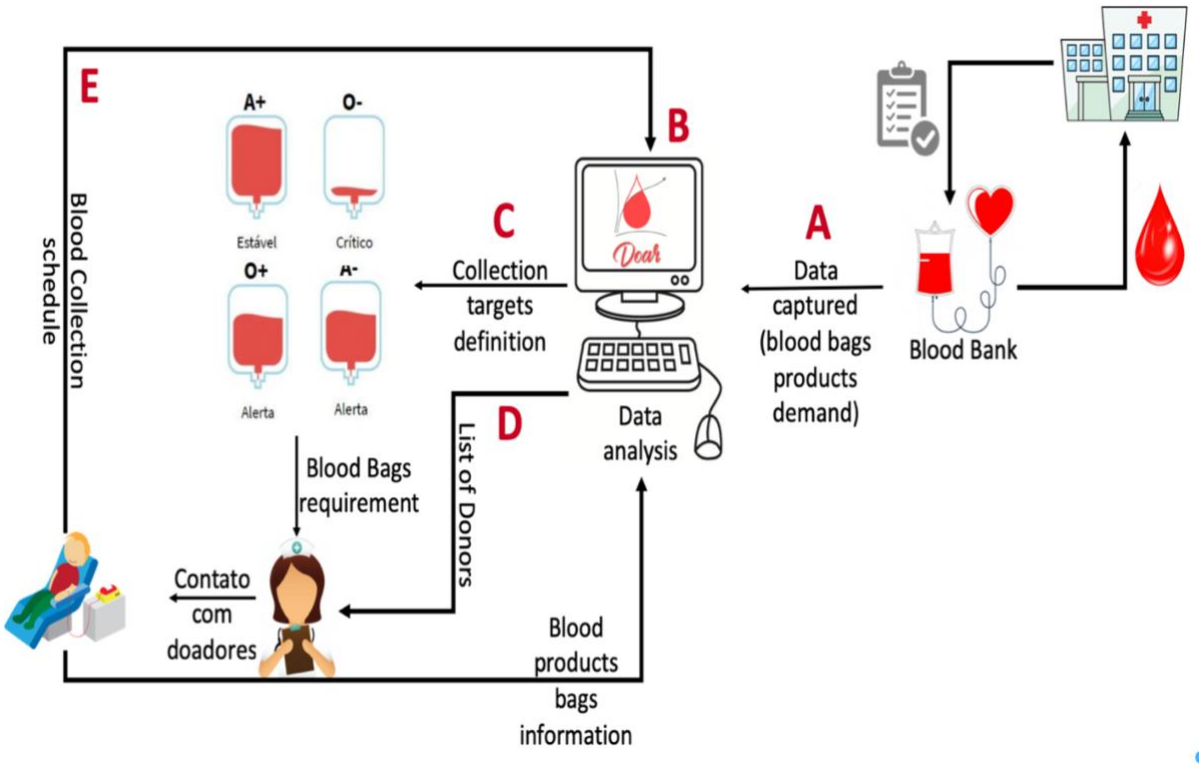
O Sistema de Monitoramento Ubíquo de Usuários (SMUU) consiste em um ambiente que possui a responsabilidade de realizar o monitoramento ubíquo de atividades cotidianas do usuário de forma dinâmica, autônoma e transparente (Sombra, 2015).

5.2. DOAR

DOAR (Do Carmo, et al., 2019) é um sistema de gerenciamento de estoque de bancos de sangue, capaz de prever demanda para minimizar o desperdício de bolsas de sangue.

O DOAR tem por objetivo promover o equilíbrio entre a oferta e a demanda por componentes de sangue, permitindo a alta disponibilidade de produtos de sangue com baixo desperdício, resultando em melhor gerenciamento de bolsas de sangue e dos componentes do sangue, promovendo o uso otimizado de seus produtos, para evitar desperdícios. A concepção do sistema é mostrada na Figura 24.

Figura 24 – Concepção do DOAR



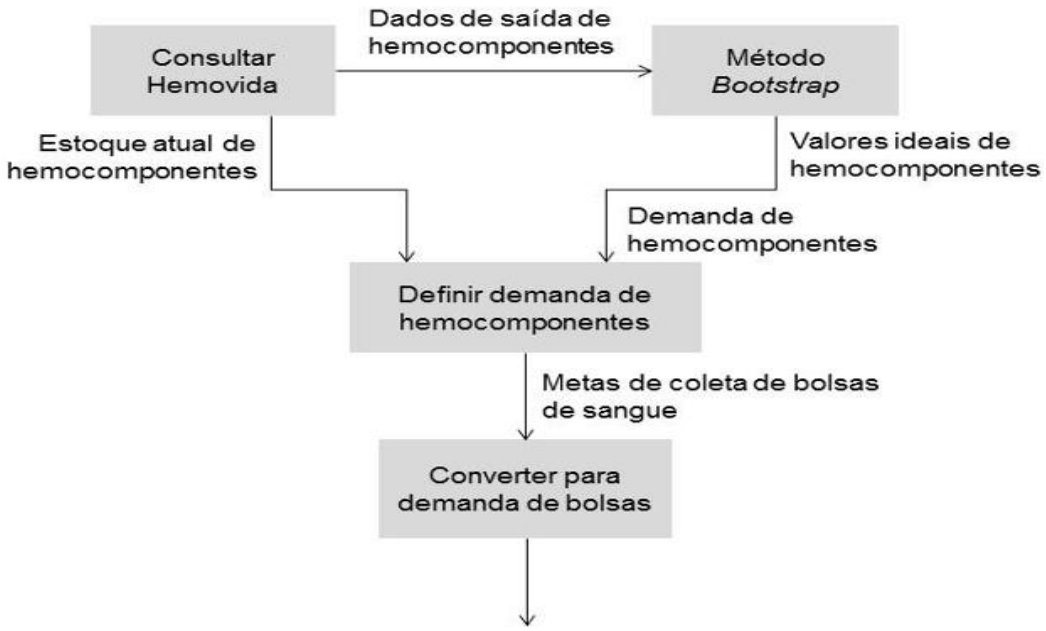
Fonte: Do Carmo, et al. (2019)

O sistema DOAR é composto por dois módulos: o de gestão de inventário é capaz de estabelecer níveis de estoque ótimos para cada hemocomponente e definir uma meta de coleta para cada tipo de sangue; e o de gestão de doadores é responsável por promover o contato entre o Hemocentro e os doadores, considerando o tipo de sangue e a aptidão para doação (Do Carmo, et al., 2019).

O módulo de gestão de inventário designa quais hemocomponentes devem ser produzidos com base na meta de bolsas a serem coletadas para cada grupo sanguíneo. Os valores ideais para cada hemocomponente de cada grupo sanguíneo são estabelecidos semanalmente de acordo com os resultados do método de gestão de estoques em reposição periódica com estoque de segurança definido por meio da técnica *bootstrap*, tendo como entrada

valores históricos de um ano de demanda. Na Figura 25, ilustra-se o fluxo do módulo de gestão de inventário.

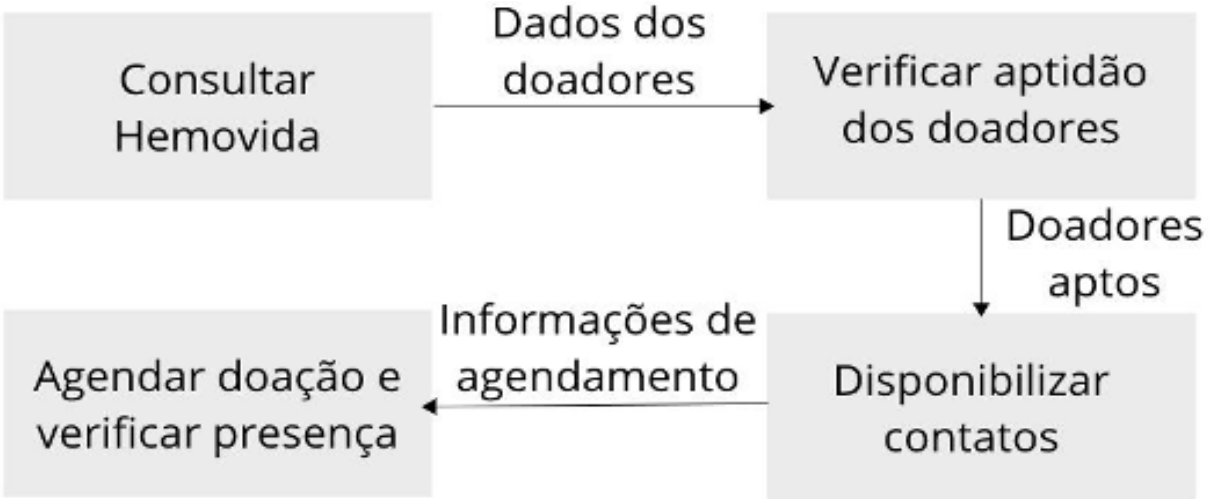
Figura 25 – Gestão de inventário – DOAR



Fonte: Do Carmo, et al. (2019)

O módulo de gestão de doadores disponibiliza os contatos, disponíveis no cadastro do doador no Hemovida, para que o Hemocentro estabeleça uma comunicação e solicite que faça a doação. Na Figura 26, são ilustradas as etapas que o módulo passa para disponibilizar as informações de doadores aptos.

Figura 26 – Gestão de doadores – DOAR



Fonte: Do Carmo, et al. (2019)

5.3. ANÁLISE DA ONTOLOGIA

Para ser possível analisar se a ontologia está consistente e respondendo as perguntas de maneira correta, foram elaboradas consultas SparQL, utilizando o Apache Jena Fuseki.

1. Quais os serviços/sistemas que estão cadastrados no *broker*?

A resposta a esta questão deve ser uma lista de serviços que estão cadastrados no *broker*. A consulta foi elaborada para que a ontologia responda quais são os serviços que estão cadastrados e o comentário (*annotation*) que está definido para cada serviço.

Na Figura 27 é apresentada a consulta SparQL e a resposta da ontologia.

Figura 27 – Consulta SparQL – Questão 1

```
PREFIX ont: <http://purl.org/net/ns/ontology-annot#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ont: <http://www.semanticweb.org/les/ontologies/2020/2/untitled-ontology-85#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT DISTINCT ?servicos ?comentarios
WHERE {
  ?servicos rdfs:subClassOf ont:Servico .
  OPTIONAL { ?servicos rdfs:comment ?comentarios}
}
```

QUERY RESULTS

Table | Raw Response

Showing 1 to 2 of 2 entries Search: Show 50 entries

	servicos	comentarios
1	ont:SACMobiLEHealthServices	"Os serviços do SAC do MobiLEHealth"
2	ont:DOAR	"Sistema de gerenciamento de estoque e doadores de sangue do Hemocentro Mossoró/RN."

Fonte: Autoria própria

2. Quais os recursos de um determinado serviço cadastrado no *broker*?

Essa questão espera ter como resposta uma lista de recursos de um determinado serviço que está cadastrado no *broker*. A consulta foi elaborada para que, ao ser informado o serviço, a ontologia responda quais são os recursos desse serviço e o comentário (*annotation*) que está definido para cada recurso.

Na Figura 28 é apresentada a consulta SparQL e a resposta da ontologia para o serviço DOAR.

Figura 28 – Consulta SparQL – Questão 2

```
PREFIX ont: <http://purl.org/net/ns/ontology-annot#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ont: <http://www.semanticweb.org/les/ontologies/2020/2/untitled-ontology-85#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT DISTINCT ?recursos ?description
WHERE {
  ?recursos rdfs:subClassOf ont:DOAR .
  OPTIONAL { ?recursos rdfs:comment ?description}
}
```

QUERY RESULTS

Table | Raw Response

Showing 1 to 11 of 11 entries

Search: Show 50 entries

	recursos	description
1	ont:listarDoadoresAptosTipoDOAR	"Listar doadores aptos de um tipo sanguíneo específico; Parâmetro 1: Tipo sanguíneo"
2	ont:estoqueDeReferenciaDOAR	"Previsão do número de bolsas que serão necessárias num mês."
3	ont:cadastrarDoadorAfereseDOAR	"Cadastrar doador por aférese; Parâmetro 1: ID do doador; Retorno 1: "err: Descrição do erro" - Em caso de erro; Retorno 2: "success: true: - Em caso de sucesso;"
4	ont:estoqueBolsasTipoFatorDOAR	"Estoque de bolsas de sangue por Tipo e Fator específico; Retorno 1: ID da Doação Realizada"
5	ont:estoqueProcessadasDOAR	"Lista todas as doações processadas que não estão vencidas"
6	ont:listarDoacoesAgendadasDOAR	"Listar todas as doações agendadas"
7	ont:agendarDoacaoDOAR	"Agendar uma doação de sangue. Parâmetro 1: Data escolhida pra marcar; Parâmetro 2: Turno (M/T/N); Parâmetro 3: ID do doador; Retorno 1: "err: Descrição do erro" - Em caso de erro; Retorno 2: "success: true: - Em caso de sucesso."
8	ont:listarDoadoresAfereseDOAR	"Listar todos os doadores por aférese"
9	ont:listarDoacoesDiaDOAR	"Listar as doações realizadas em um determinado dia"
10	ont:listarDoadoresDOAR	"Listar todos os doadores cadastrados."
11	ont:estoqueHemovidaDOAR	"Lista todas as doações que não estão vencidas"

Showing 1 to 11 of 11 entries

Fonte: Autoria própria

3. **Quais os parâmetros, retornos, formatos, *links* de acesso de um determinado recurso cadastrado no *broker*?**

Essa questão espera ter como resposta uma lista de parâmetros, retornos, formato do retorno e *link* de um determinado recurso que está cadastrado no *broker*. A consulta foi elaborada para que, ao ser informado o recurso, a ontologia responda todas as informações deste recurso.

Na Figura 29 é apresentada a consulta SparQL e a resposta da ontologia para o recurso “listarDoacoesDia” do serviço DOAR. No exemplo em tela, o recurso tem como retorno um tipo próprio (DoacaoRealizada). A ontologia apresenta esse valor na coluna tipoClasse e, na coluna values, apresenta cada atributo que faz parte desse tipo (*int doador, string hrinicio, string hrtermino, int doação, date dtaend*).

Figura 29 – Consulta SparQL – Questão 3

```
PREFIX ont: <http://purl.org/net/ns/ontology-annot#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ont: <http://www.semanticweb.org/les/ontologies/2020/2/untitled-ontology-85#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?propertyservices ?tipoClasse ?values
WHERE {
  {

    ont:listarDoacoesDiaDOAR rdfs:subClassOf ?subcservices .
      ?subcservices owl:onProperty ?propertyservices.
      ?subcservices owl:someValuesFrom ?tipoClasse .
      ?tipoClasse rdfs:subClassOf ?subargs .
      ?subargs owl:hasValue ?values
  }
      UNION
      {

      ont:listarDoacoesDiaDOAR rdfs:subClassOf ?subcservices .
      ?subcservices owl:onProperty ?propertyservices .
      ?subcservices owl:hasValue ?values
  }
  }
```

Table | Raw Response

Showing 1 to 8 of 8 entries — Search: — Show 50 entries

	propertyservices	tipoClasse	values
1	ont:temRetorno	ont:DoacaoRealizadaDOAR	"doador | int"
2	ont:temRetorno	ont:DoacaoRealizadaDOAR	"hrinicio | string"
3	ont:temRetorno	ont:DoacaoRealizadaDOAR	"hrtermino | string"
4	ont:temRetorno	ont:DoacaoRealizadaDOAR	"doacao | int"
5	ont:temRetorno	ont:DoacaoRealizadaDOAR	"dtaend | date"
6	ont:temParametro1		ont:date
7	ont:temFormato		"JSON"
8	ont:temLink		"POST /api/doacoes/dia"

Showing 1 to 8 of 8 entries

Fonte: Autoria própria

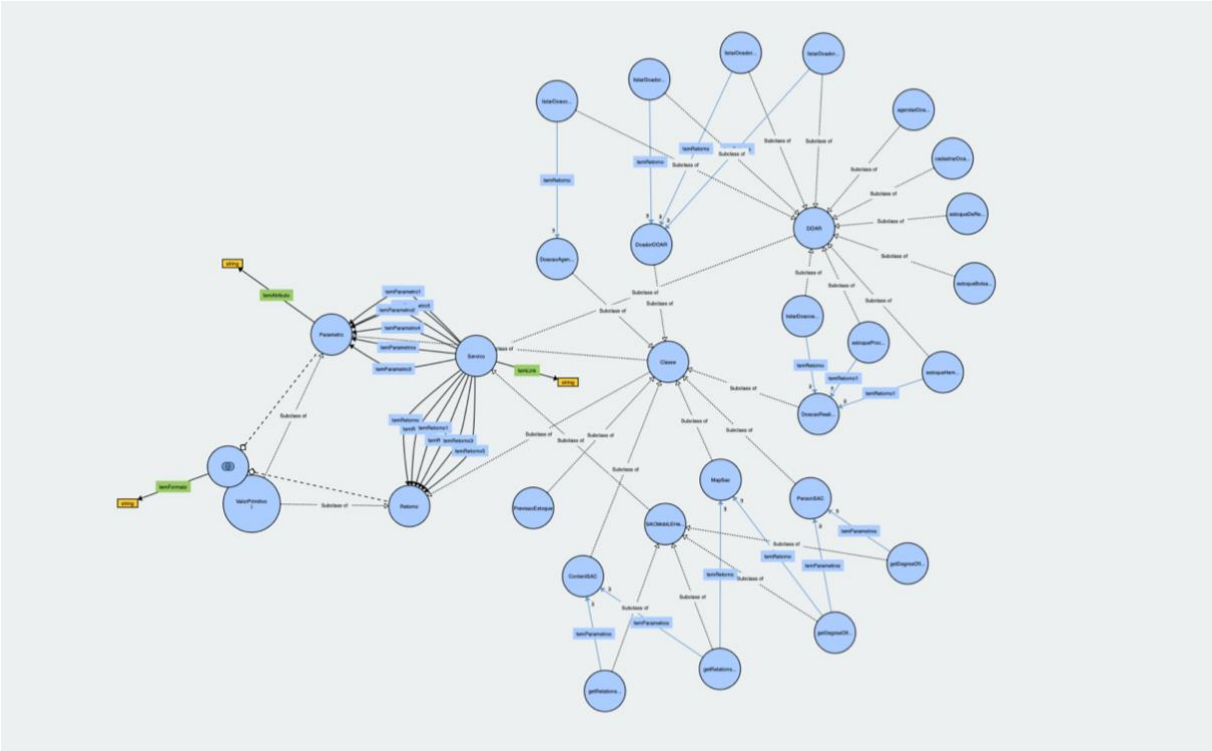
Os resultados obtidos mostram que não há inconsistência na ontologia e Os resultados tinham a forma esperada. Com isso, é possível observar que a ontologia é capaz de responder todas as perguntas de competência propostas, confirmando, assim, que a ontologia é eficaz.

Para ser avaliada a completude e concisão da ontologia, foi verificado se a ontologia está completamente definida e que não possui definições desnecessárias ou redundantes. Para esse passo, foi utilizado o visualizador gráfico WebVOLW[10], esta ferramenta foi desenvolvida para propiciar a visualização gráfica dos elementos pertencentes à ontologia.

De acordo com o que é mostrado na Figura 30, é possível observar que não há classes e propriedades que não se encontrem conectadas. De forma que garante-se que a ontologia é hiperconectada, isto é, todas as informações inseridas podem ser recuperadas por meio de consultas na ontologia, garantindo assim sua completude e concisão.

[10] Visual Notation for OWL Ontologies. Disponível em: http://www.visualdataweb.de/webvowl/

Figura 30 – Visualização da ontologia



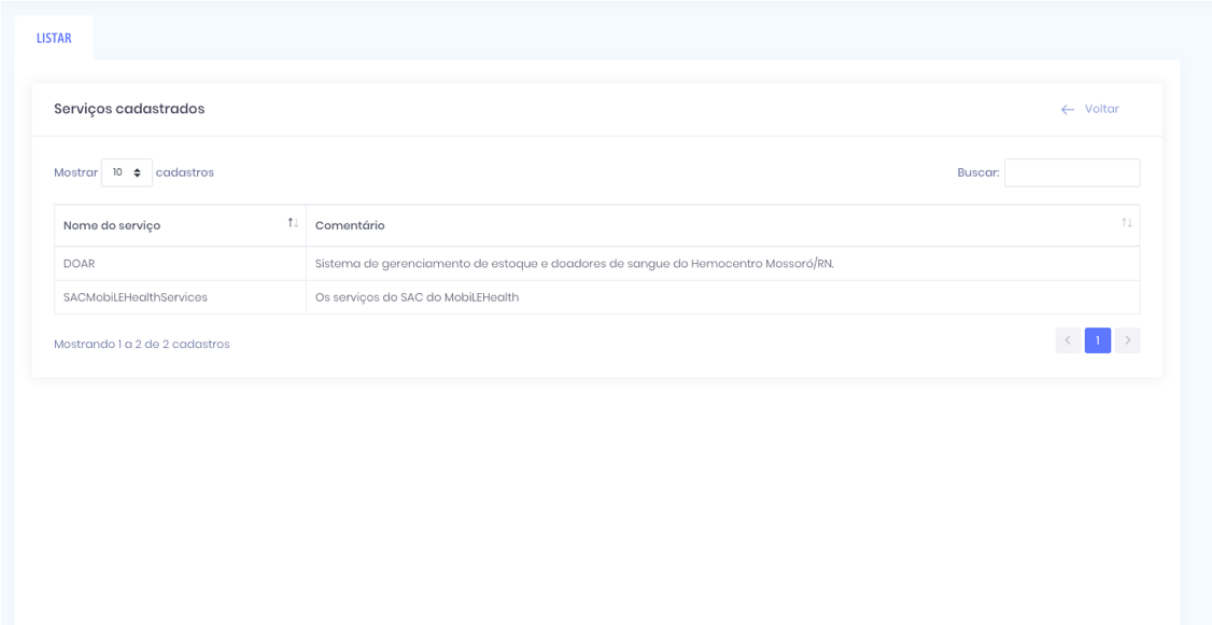
Fonte: Autoria própria

5.4. INTERFACE WEB

Para visualização das funcionalidades do *broker*, foi implementada uma interface *web*, utilizando a linguagem PHP, com o auxílio de HTML, CSS, Javascript, *framework* Bootstrap e da biblioteca *BorderCloud*, que possibilita inserir consultas SparQL em uma página PHP.

Na página inicial é mostrada uma lista dos serviços que estão cadastrados no *broker*, como mostrado na Figura 31.

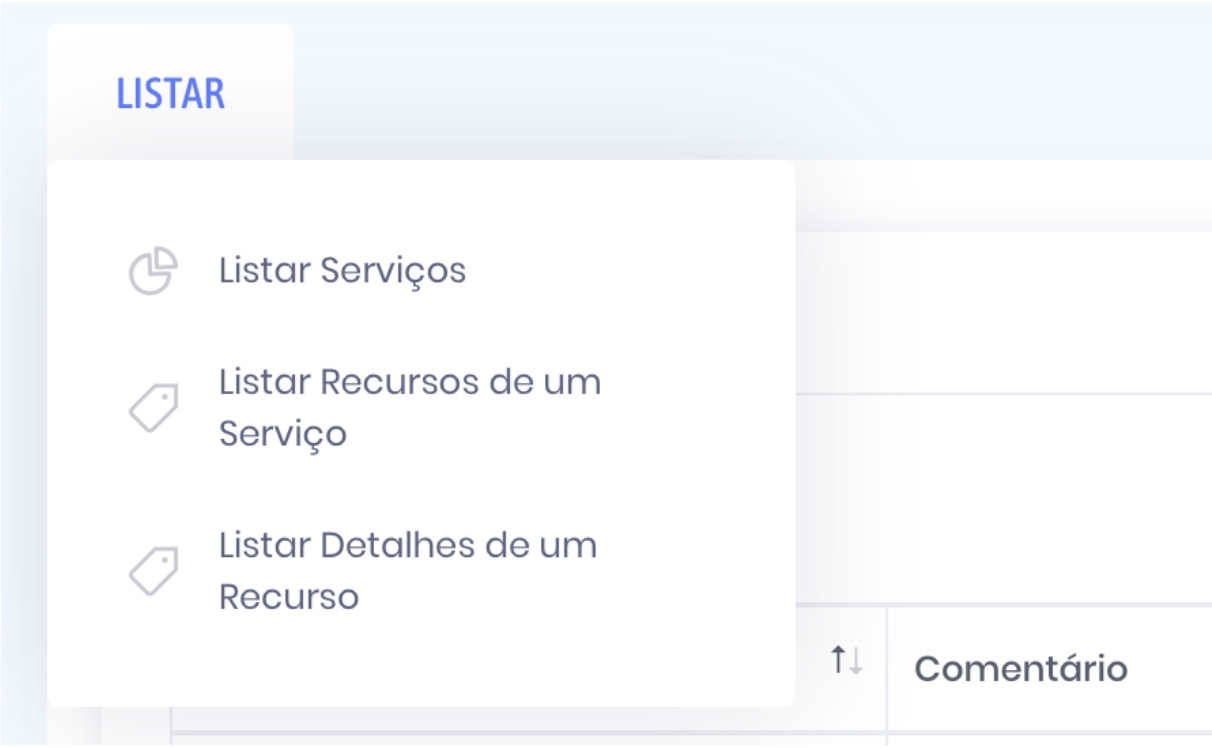
Figura 31 – Serviços cadastrados



Fonte: Autoria própria

No menu superior, são mostradas as opções: Listar Serviços (página inicial), Listar Recursos de um Serviço e Listar Detalhes de um Recurso, conforme Figura 32.

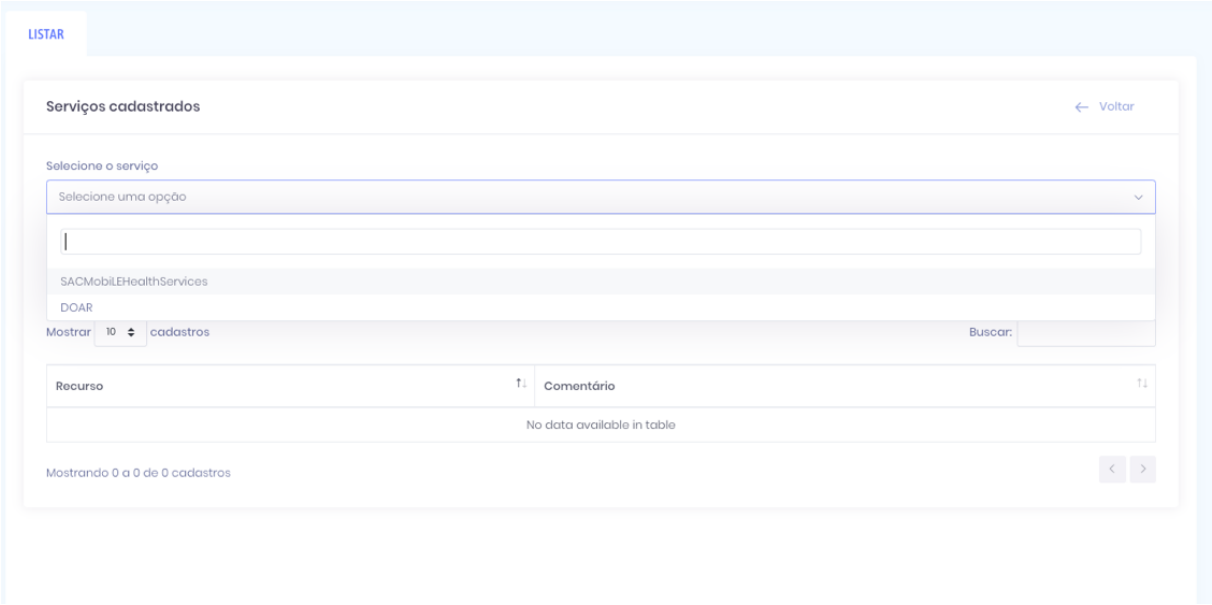
Figura 32 – Menu superior



Fonte: Autoria própria

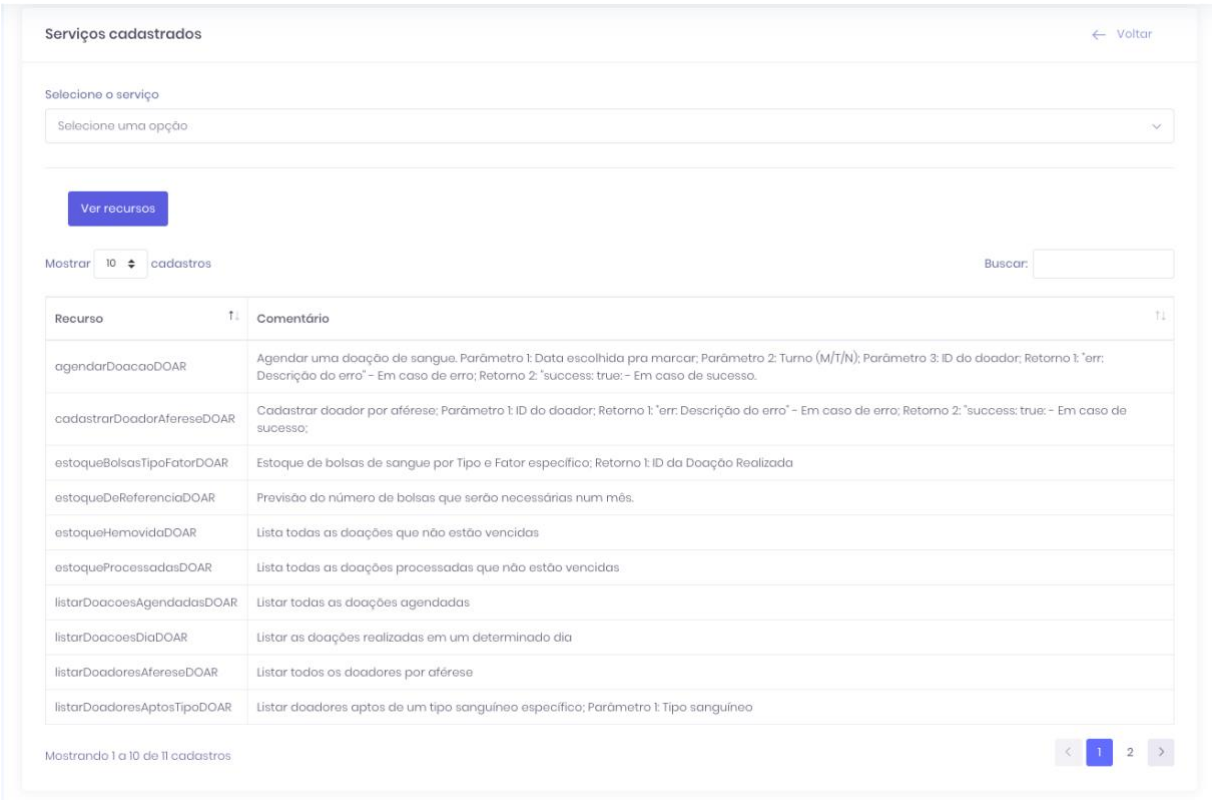
Na página “Listar Recursos de um Serviço” é exibido um Elemento HTML *select*, em que são listados os serviços disponíveis e, ao selecionar uma opção, são listados os recursos desse determinado serviço. Essa página é mostrada na Figura 33 e Figura 34.

Figura 33 – Opção *select*



Fonte: Autoria própria

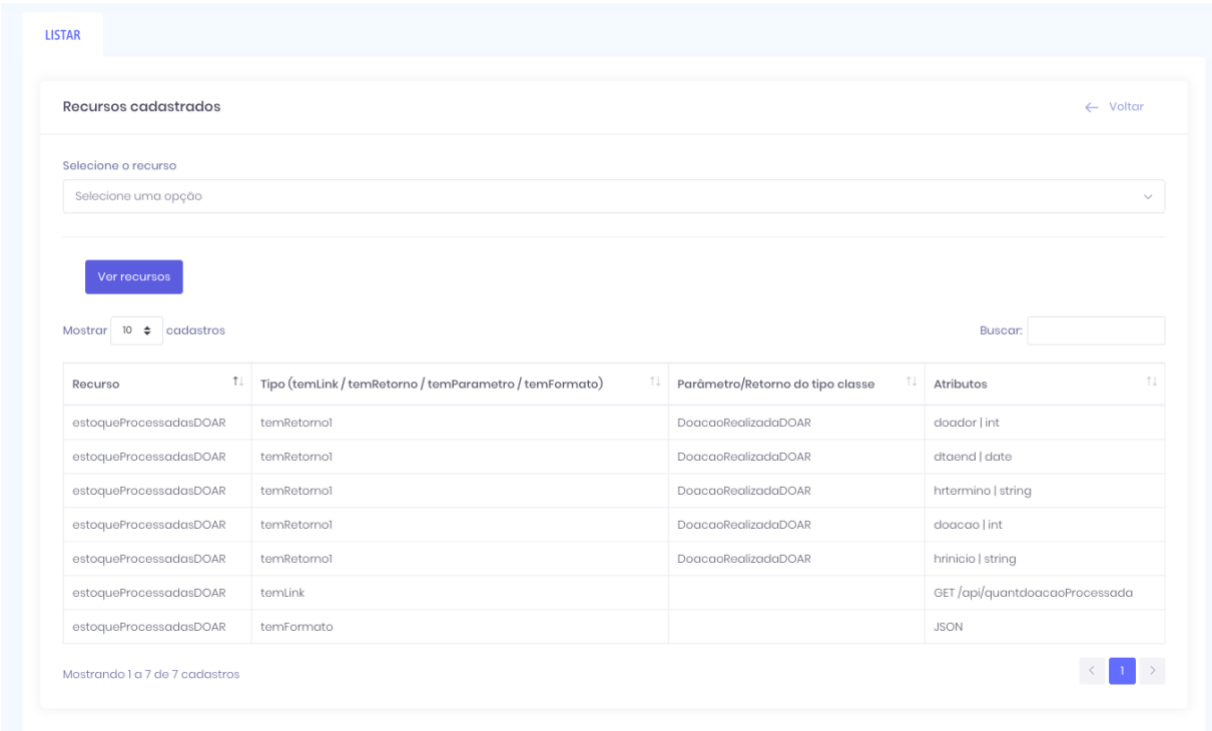
Figura 34 – Lista de recursos



Fonte: Autoria própria

Na página “Listar Detalhes de um Recurso” são exibidos os detalhes como: parâmetros, retornos, formato de retorno e *link* de um recurso específico, que é escolhido na opção *select*. No exemplo da Figura 35, o recurso estoqueProcessadas tem como retorno um tipo específico “DoacaoRealizadaDOAR”, que é especificado na coluna “Atributos” (*int doador, date dtaend, string hrtermino, int doacao, string hrinicio*). Também é possível extrair o *link* de acesso (GET /api/quantdoacaoProcessada) e o formato de retorno (JSON).

Figura 35 – Detalhes de um recurso



Fonte: Autoria própria

6. CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

As aplicações de saúde são ferramentas fundamentais no contexto atual. Tais ferramentas exercem um papel fundamental na qualidade de vida das pessoas. A evolução da tecnologia possibilitou integração dos sistemas computacionais com o mundo físico, através da utilização dos Sistemas Físico-Cibernéticos, introduzindo, assim, um novo nível de interação entre sistemas e pessoas. Os Sistemas Médicos Físico-Cibernético são sistemas da informação que proveem a tomada de decisão de forma automatizada. Tais sistemas podem melhorar a vida dos pacientes, usando uma abordagem que possibilita, de forma automatizada, a obtenção de todos os dados e recursos médicos disponíveis para localizar o tratamento para um paciente.

Existem diversas aplicações com características específicas disponíveis aos usuários. Por diversas vezes, o usuário necessita utilizar várias dessas aplicações para poder dar suporte às operações que necessita, pois uma única aplicação não é capaz de cobrir todas as funcionalidades necessárias.

Com base nisso, neste trabalho foi apresentado o projeto e desenvolvimento de um *broker*, que permite que serviços de saúde compartilhem recursos presentes em suas aplicações. O *broker* fornece ao solicitante as informações necessárias para que este possa ter acesso aos recursos disponibilizados por outros serviços.

O *broker* é composto por uma ontologia, em que são descritos os serviços disponíveis, além de uma interface *web*, por meio da qual os consumidores podem ver quais recursos estão disponíveis e detalhes sobre cada um.

Para validar a solução proposta, foram utilizados dois sistemas: o MobiLEHealth e o DOAR. A validação realizada foi bem sucedida, pois os recursos de ambos os sistemas foram acessados pelo *broker* de forma eficaz.

Algumas limitações na solução proposta foram identificadas após a implementação do *broker*:

- A adição de um novo serviço de saúde com recursos depende que o mapeamento seja feito diretamente na ontologia;
- Pequeno número de funcionalidades, visto que apresenta apenas funcionalidades de exibir as informações já inseridas na ontologia.

Com base nessas limitações, pode-se identificar algumas melhorias possíveis. Sendo assim, como trabalhos futuros, sugere-se:

- Integrar outros serviços de saúde à ontologia;
- Expandir as funcionalidades, possibilitando, por exemplo, que seja possível inserir um novo serviço pela interface *web*;
- Implementar uma busca semântica de serviços cadastrados no *broker;*
- Melhorar o *broker* e a ontologia, de forma que seja possível estabelecer um *middleware* que, ao ser requisitado, realize o acesso aos diversos recursos, de diversos sistemas, que possam responder a esta requisição e devolva, ao serviço requisitante, uma resposta concisa, combinando todas as respostas dos diferentes serviços.

REFERÊNCIAS

ACM/IFIP/USENIX 10th International Conference, Urbana, IL, USA, "Middle2009" [Periódico].

ACM/IFIP/USENIX Middle2009 [Artigo] // 10th International Conference, Urbana, IL, USA. - 2009.

ALBARRAK K. M. e SIBLEY E. H. A survey of methods that transform data models into Ontology models [Artigo] // IEEE International Conference on Information Reuse and Integration (IRI). - 2011.

Almeida M e Bax M Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção [Artigo] // Revista Ciência da Informação. - 2003.

Alonso G. [et al.] Web Services: Concepts, Architectures and Applications [Seção do Livro]. - 2004.

Armitage Gail D. [et al.] Health systems integration: state of the evidence [Periódico]. - [s.l.] : International journal of integrated care 9, 2009.

BLACKMAN S Serious Games... and Less [Artigo] // ACM Siggraph Computer Graphics. - 2005.

Breitman K. Web Semântica: A Internet do Futuro. [Artigo]. - 2005.

Costa A. A. L. [et al.] Um sistema de enriquecimento semântico de perfil de usuário baseado em traços digitais para apoio à aprendizagem informal no contexto da saúde [Artigo] // RENOTE 12. - 2014.

Do Carmo B. B. T. [et al.] Blood Inventory Management System: Reducing Wastage and Shortage [Seção do Livro] // In International Joint conference on Industrial Engineering and Operations Management (pp. 24-30). - [s.l.] : Springer, Cham, 2019.

Dogaru Delia e Dumitrache Ioan Cyber-Physical Systems in Healthcare Networks [Artigo] // The 5th IEEE International Conference on E-Health and Bioengineering - EHB 2015. - 2015.

Fernandez Fernando Henrique Discussão de um modelo conceitual para Enterprise Application Integration - EAI [Artigo]. - 2004.

Gonçalves B. M. S. MOM – Message Oriented Middleware [Online] // INESC-ID. - 2010. - 07 de 04 de 2020. - https://www.gsd.inesc-id.pt/~ler/docencia/tm0405/slides/MessageOrientedMiddleware-MOM.pdf.

Grispos George, Glisson William e Choo Kim-Kwang Medical Cyber-Physical Systems Development: A Forensics-Driven Approach [Artigo] // 2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE). - 2017.

Gruber T. Ontology in Encyclopedia of Database Systems [Artigo]. - 2009.

Gruninger M e Fox M Methodology for the Design and Evaluation of Ontologies [Artigo]. - 1995.

Juric Matjaz B. [et al.] Professional J2EE EAI. [Artigo] // Wrox Press Ltd.. - 2001.

Kiani S. A. [et al.] A Federated Broker Architecture for Large Scale Context Dissemination [Artigo] // 10th IEEE International Conference on Computer and Information Technology (CIT 2010). - 2010.

Lee I [et al.] High-confidence medical device software and systems [Artigo]. - 2006.

Lee Insup e Sokolsky Oleg Medical Cyber Physical Systems [Artigo] // ACM. - 2010.

Li Xiangyu An integration framework for information system based on web service [Artigo]. - 2010.

LINTHICUM David S Enterprise Application Integration. Information technology series [Artigo]. - 2000.

Linthicum David S. Data Level Integration (Part I) [Artigo]. - 1999.

LUBLINSKY Boris The Ultimate EAI Implementatio [Artigo] // EAI Journal. - 2001.

McGoveran David Data Integration I, II, III, IV, V, VI, VII, VIII, IX. [Artigo]. - 2002.

Mello R. X. e Cerqueira R. F. G. A COLLABORATIVE MODEL FOR EVENT SCHEDULING USING COROUTINES: CHAPTER 4 [Online]. - 04 de 04 de 2006. - 07 de 04 de 2020. - https://www.maxwell.vrac.puc-rio.br/8079/8079_5.PDF.

Ministério da Saúde DataSUS [Online] // Padrões e normas. - 2020. - 01 de 09 de 2020. - https://datasus.saude.gov.br/interoperabilidade-padroes-e-normas/.

Moltchanov B., Knappmeyer M. e Licciardi C. A. Context-Aware Content Sharing and Casting [Artigo] // 12th ICIN. - 2008.

Montón José Manuel Gómez Montón José Manuel Gómez Interoperabilidade nos sistemas de saúde [Online] // ehCOS. - 2017. - 01 de 09 de 2020. - https://www.ehcos.com/pt-br/interoperabilidade-nos-sistemas-de-saude/.

MORAIS E A M e AMBROSIO A P L Ontologias: conceitos, usos, tipos, metodologias, ferramentas e linguagens [Relatório]. - 2007.

Morais E. A. M. e Ambrósio A. P. L. Ontologias: conceitos, usos, tipos, metodologias, ferramentas e linguagens [Artigo]. - 2007.

Moreira J. D. C. [et al.] Um sistema de enriquecimento semântico de perfil de usuário baseado em traços digitais para apoio à aprendizagem informal no contexto da saúde [Artigo] // RENOTE 12. - 2014.

Pautasso C, Zimmermann O e Leymann F RESTful Web Services versus 'Big' Web Services: Making the Right Architectural Decision [Artigo] // World Wide Web Conf. (WWW '08). - 2008.

Süß J. G. e Mewes M. An Architecture Proposal for Enterprise Message Brokers [Artigo]. - 2001.

Samtani G. e Sadhwani D. Integration Brokers and Web Services [Seção do Livro]. - 2002.

Santana Luiz Henrique e Da Silva Leandro Libério Integração de Sistemas: um estudo de caso do Sistema de Agendamento de Relatórios de uma Instituição Financeira [Artigo] // e-Tec, v. 2, n. 1. - 2009.

Sombra Enio Lopes MOBILEHEALTH: Um ambiente de apoio a saúde 2.0 [Artigo]. - 2015.

Sultanovs E, Skorobogatjko A e Romanovs A Centralized Healthcare Cyber-Physical System's Architecture Development [Artigo] // RTUCON 2016 – Proceedings of the 57th Annual international scientific conference on power and electrical engineering. - 2016.

W3C World Wide Web Consultoriu Architecture of the World Wide Web [Online]. - 2004. - 1 de julho de 2019. - https://www.w3.org/TR/webarch/.

Xu Aiping [et al.] Research on Automation Integration Technology of Application Systems Based on Web Services [Artigo]. - 2016.