UNIVERSIDADE ESTADUAL DO CEARÁ
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
INSTITUTO FEDERAL DE EDUCAÇÃO,
CIÊNCIA E TECNOLOGIA DO CEARÁ
PRÓ-REITORIA DE PÓS-GRADUAÇÃO - PROPG

MESTRADO PROFISSIONAL EM COMPUTAÇÃO APLICADA

TACIANO PINHEIRO DE ALMEIDA ALCÂNTARA

PAOLA: UMA PLATAFORMA PARA O DESENVOLVIMENTO DE APLICAÇÕES BASEADAS EM ONTOLOGIAS PARA O PROJETO LARIISA

FORTALEZA

2012

TACIANO PINHEIRO DE ALMEIDA ALCÂNTARA

PAOLA: UMA PLATAFORMA PARA O DESENVOLVIMENTO DE APLICAÇÕES BASEADAS EM ONTOLOGIAS PARA O PROJETO LARIISA

Dissertação submetida à Coordenação do Curso de Mestrado Profissional em Computação Aplicada da Universidade Estadual do Ceará e do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, como requisito parcial para a obtenção do grau de Mestre em Computação Aplicada.

Orientador: Prof. Dr. Antônio Mauro Barbosa de Oliveira.

FORTALEZA

2012

(MODELO)

Dados Internacionais de Catalogação na Publicação

Universidade Estadual do Ceará

Biblioteca Central Prof. Antônio Martins Filho

C999m Alcântara, Taciano Pinheiro de Almeida

Paola: Uma Plataforma Para O Desenvolvimento De Aplicações Baseadas Em Ontologias Para O Projeto Lariisa / Taciano Pinheiro de Almeida Alcântara – 2012.

99f. : il. color., enc. ; 30cm.

Dissertação (mestrado) – Universidade Estadual do Ceará, Centro de Ciências e Tecnologia, Curso de Mestrado Profissional em Computação Aplicada, Fortaleza, 2012.

Área de concentração: Redes de Computadores

Orientação: Prof. Dr. Antônio Mauro Barbosa de Oliveira

1. Contexto. 2. Sistemas sensíveis ao contexto. 3. Desenvolvimento de sistemas.

CDD: 001.64

TACIANO PINHEIRO DE ALMEIDA ALCANTARA

PAOLA: UMA PLATAFORMA PARA O DESENVOLVIMENTO DE APLICAÇÕES BASEADAS EM ONTOLOGIAS PARA O PROJETO LARIISA

Dissertação submetida à Coordenação do Curso de Mestrado Profissional em Computação Aplicada da Universidade Estadual do Ceará e do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, como requisito parcial para a obtenção do grau de Mestre em Computação Aplicada.

Orientador: Prof. Dr. Antônio Mauro Barbosa de Oliveira.

Aprovada em: ___ / ___ / _____

BANCA EXAMINADORA

_____

Prof. Dr. Antônio Mauro Barbosa de Oliveira (IFCE)
Presidente (Orientador)

_____

Prof. Dr. Marcos José Negreiros Gomes (UECE)
1º Membro Interno

_____

Prof. Dr. Anilton Salles Garcia (UFES)
2º Membro Interno

_____

Prof. Dr. Luiz Odorico Monteiro de Andrade (UFC)
Membro Externo

Dedico este trabalho à minha família.

AGRADECIMENTOS

A Deus, pelas oportunidades.

Ao meu orientador, Dr. Mauro Oliveira, pelos ensinamentos.

A todos os professores do MPCOMP, pelo relevante trabalho que realizam.

A UECE e ao IFCE, por viabilizarem este curso.

Aos meus pais, Cazuza e Seabranira, pelo apoio incondicional.

Ao meu irmão Emanuel, minha cunhada Angeliene e minha sobrinha Luma, por tudo.

Ao meu irmão Aureliano, pelos momentos de descontração.

A minha namorada, Dionizia, pela compreensão.

Ao grande amigo e primo MSc. Yuri Lacerda, pelo incentivo.

Ao grande amigo, primo, companheiro e médico Dr. Thales Cavalcante, pela companhia e por cuidar da minha saúde. A sua mãe, Sônia, pelos cuidados.

Ao grande amigo Alex Penha, pela amizade.

As empresas Stefanini, BRQ, Faculdade Leão Sampaio e UFC, e todos seus funcionários, especialmente representados por Wabber Filho, João Barros, Jorge Salvador, Eva Campos, Thiago Bessa e Wellington Sarmento, pela ajuda.

Aos colegas do MPCOMP, Rita de Castro, Michelle Queiroz, Claudemir Queiroz, Hedwio Carvalho, Marcondes Alexandre, Herbert Novais e Márcio Correia, pela união.

“Dá-me sabedoria e conhecimento, pois confio nos teus mandamentos.”

(Salmos 199,66).

RESUMO

O Lariisa é um projeto para tomada de decisão em governança de sistemas públicos de saúde. Ele integra dados de pacientes em suas residências para a construção de sistemas inteligentes capazes de apoiar a tomada de decisão em suas diversas instâncias. O projeto Lariisa prevê a representação de informação contextual para ser utilizada por aplicações sensíveis ao contexto (*context-aware concept*) e representação de conhecimento mediante o uso de ontologias. O desenvolvimento de aplicações no projeto Lariisa tem certa complexidade por envolver a representação do conhecimento (informações capturadas dos diversos atores do sistema, domínio representado, diversas bases de dados relacionadas, entre outros) e a captura de informações de contexto, via mecanismos de *hardware*, que alimentam os mecanismos de inferência existentes. A inexistência de mecanismos que apoiem o desenvolvimento dessas aplicações dificulta a utilização do projeto Lariisa, resultando em soluções *ad-hoc* com escopo específico, com maior tempo dispendido e elevado custo. Esta dissertação apresenta a Paola (Plataforma Aplicações Ontologia LAriisa). Trata-se de um mecanismo que facilita o desenvolvimento de aplicações no projeto Lariisa, levando em consideração suas especificidades. A plataforma Paola integra as informações de contexto provenientes da arquitetura Lisa (Lariisa *Integration System Architecture*) com Editores de Ontologias e *frameworks* de base de conhecimento (ex.: Protégé-OWL). Assim, é disponibilizado ao desenvolvedor do projeto Lariisa funcionalidades como a busca (em bases diversas), o reuso e a edição de ontologias e definição de regras para a criação de novas aplicações. A plataforma Paola oferece, naturalmente, funcionalidades inerentes ao projeto Lariisa, necessárias à nova aplicação sensível ao contexto, além de outras facilidades que possam ajudar o desenvolvedor da definição de sua aplicação (ex.: exemplos contextuais) e no desenvolvimento (ex.: *templates*, *tutorial*).

Palavras-chave: Sistemas sensíveis ao contexto; Ontologia; Governança de saúde; Lariisa.

ABSTRACTS

Lariisa is a decision making project for public health systems governance. It integrates patient data from their homes for the construction of intelligent systems capable of supporting the making of decisions in different situations. The Lariisa project has representation of contextual information to be used by context-aware applications and knowledge representation through the use of ontologies. The application development in the Larissa project is complex because it involves the knowledge representation (information gathered from diverse actors in the system, represented domain, diverse related data bases, among others) and the gathering of contextual information through hardware mechanisms that feed the existing inference mechanisms. The lack of mechanisms that support the development of these applications makes it difficult to use the Lariisa project in satisfying its objectives, resulting in ad-hoc solutions with specific scopes that take longer and cost more. This dissertation presents PAOLA (Platform for Lariisa Ontological Applications). It is a mechanism that facilitates the application development in the Lariisa project, taking into consideration their specificities. The Paola platform integrates contextual information originating from Lisa architecture (Lariisa Integration System Architecture) with ontological editors and frameworks based on knowledge (ex.: Protégé-OWL). Because of this, it enables the Lariisa project developer to use functions such as searching (in different data bases), the reuse and the edition of ontologies and the definition of rules for the creation of new applications. Of course the Paola platform offers functions inherent to the Lariisa project necessary to new applications sensitive to context as well as other tools that can help to develop the definition of its applications (ex.: contextual examples) and in development (ex.: templates, tutorial).

Key words: *Context-aware systems*; *Ontology*; *Health governance*; Lariisa.

LISTA DE FIGURAS

Figura 1 - Tipos de Ontologias (GUARINO, 1998).....28
Figura 2 – Exemplo de configuração de componentes de contexto. Setas indicam fluxo. (SALBER; DEY; ABOWD, 1999) 37
Figura 3 – Esquema de criação de regras do VadeMecum (FIGUEIREDO, 2009) 38
Figura 4 – Outdoor Virtual – aplicação móvel multimídia desenvolvida com o VadeMecum (FIGUEIREDO, 2009).....39
Figura 5 – *Sentient object model* proposto por Biegel e Cahill (2004) 41
Figura 6 - Protégé 42
Figura 7 – Modelo Global de Saúde do projeto Lariisa (OLIVEIRA et al., 2010) 46
Figura 8 – Modelo Local de Saúde do projeto Lariisa (OLIVEIRA et al., 2010) 47
Figura 9 – Lariisa *Framework Core* 54
Figura 10 – Funcionalidades do Diga Saúde (SANTOS, 2011) 57
Figura 11 – Visão Geral do Diga Saúde (SANTOS, 2011) 58
Figura 12 – Diagrama de Blocos do Projeto Lariisa acrescido da arquitetura Lisa (FROTA, 2011) 59
Figura 13 – Integração de aplicativos na arquitetura Lisa/Lariisa (FROTA, 2011) 59
Figura 14 – Arquitetura da Lisa (FROTA, 2011) 60
Figura 15 – Visão geral da aplicação Sisa (ANTUNES, 2011) 61
Figura 16 – Diagrama de Atividades do Sisa (ANTUNES, 2011) 62
Figura 17 – Os caminhos do conhecimento dentro do projeto Lariisa (BASTOS, 2012) 63
Figura 18 – Diagrama de blocos dos módulos da plataforma Paola.....68
Figura 19 – Abstração de detalhes do projeto Lariisa através da plataforma Paola 69
Figura 20 – Diagrama de Caso de Uso das funcionalidades da plataforma Paola 70

Figura 21 – Tela principal da interface gráfica da plataforma Paola 71
Figura 22 – Diagrama de Caso de Uso de edição de Bases de Conhecimento da plataforma Paola 73
Figura 23 – Tela de edição de Bases de Conhecimento da plataforma Paola74
Figura 24 – Diagrama de Caso de Uso de edição de Provedores de Contexto da plataforma Paola 75
Figura 25 – Tela de edição de Provedores de Contexto da plataforma Paola76
Figura 26 – Diagrama de Caso de Uso de edição de Regras na plataforma Paola 78
Figura 27 – Tela de edição de Regras da plataforma Paola.....79
Figura 28 – Diagrama de caso de uso de Gerenciamento da Informação da plataforma Paola 80
Figura 29 – Tela de Gerenciamento de Informação (consulta a dados contextuais) 81
Figura 30 – Diagrama de caso de uso do módulo Gerador de artefato executável & Simulação 82
Figura 31 – Tela do Gerador de Artefato Executável & Simulação de Aplicação Sensível ao Contexto 83
Figura 32 – Diagrama de Atividades do fluxo principal de execução da plataforma Paola 84
Figura 33 – Arquitetura da plataforma Paola 87
Figura 34 – Tela de edição de ontologia do Protégé 90
Figura 35 – Aspectos de implementação do Módulo de Gerenciamento de Bases de Conhecimento 90
Figura 36 – Tela de desenvolvimento da manipulação de bases de conhecimento 91
Figura 37 – Integração de provedores de contexto ao projeto Larissa com o Lisa.....93
Figura 38 – Exemplo de integração de Provedores de Contexto à plataforma Paola utilizando a arquitetura Lisa 94
Figura 39 – Aspectos de implementação das Regras do Módulo de Gerenciamento de Regras e Ações 95
Figura 40 – Aspectos de implementação de Ações do Módulo de Gerenciamento de Regras e Ações 97

Figura 41 – Aspectos de implementação do Módulo de Gerenciamento de Informação 98
Figura 42 – Gráfico da ontologia.....102
Figura 43 – Desenvolvimento de ontologias utilizando o Protégé 103
Figura 44 – Plataforma Paola e Mapas Conceituais (BASTOS, 2012) independentes.....110
Figura 45 – Mapas Conceituais e integrados às regras e à plataforma Paola 111

LISTA DE TABELAS

Tabela 1 – Quantitativo de referências de pesquisa.....22
Tabela 2 – Comparação entre os trabalhos relacionados:.....43

LISTA DE QUADROS

Quadro 1 – Atividades dos profissionais do PSF 34
Quadro 2 – Exemplo de regra de decisão global 48
Quadro 3 – Exemplo de regra de decisão global utilizando SWRL 48
Quadro 4 – Exemplo de regra de decisão local 49
Quadro 5 - Exemplo de regra de decisão local utilizando SWRL 49
Quadro 6 – Exemplo de regra de decisão local em Normatização Sistêmica. 50
Quadro 7 – Exemplo de regra de decisão global (Clínico-Epidemiológico) 50
Quadro 8 – Exemplo de regra de decisão global (Clínico-Epidemiológica) utilizando SWRL 50
Quadro 9 – Exemplo de regra de decisão em administração 51
Quadro 10 – Exemplo de regra de decisão em administração utilizando SWRL 51
Quadro 11 – Exemplo de regra de decisão em Gerenciamento Compartilhado 52
Quadro 12 – XML *Schema* do Repositório de Bases de Ontologias 89
Quadro 13 – XML Schema do repositório de bases de ontologias 92
Quadro 14 – Exemplo de inserção da base de dados contextual 94
Quadro 15 – Exemplo de regra na Jena/GPRE 96
Quadro 16 – Exemplo de consulta a dados em base de conhecimento 98
Quadro 17 – Exemplo de resultado de consulta a dados em base de conhecimento 99
Quadro 18 - Pré-condições da regra do Sisa 105
Quadro 19 – Exemplo de consulta a dados em base de conhecimento 105
Quadro 20 – Utilização do artefato executável no Sisa 106

LISTA DE ABREVIATURAS E SIGLAS

3G	*3rd Generation Mobile Telecommunications*
ANS	Agência Nacional de Saúde Suplementar (Brasil)
API	*Application Programming Interface*
CA	*Context Aggregator*
CARE	*Context-Aware Rule Editor*
CAS	*Context-Aware Service*
CASC	CAS *Container*
CDN	*Code Development Network*
CI	Ciência da Informação
CP	*Context Provider*
CR	*Context Reasoner*
DAML	DARPA *Agent Markup Language*
DARPA	*Defense Advanced Research Project Agency (USA)*
Datasus	Departamento de Informática do SUS
DCS	*Distributed Classification System*
ECA	*Event-Condition-Action*
ESB	*Enterprise Service Bus*
ETICE	Empresa de Tecnologia da Informação do Estado do Ceará
FIOCRUZ	Fundação Oswaldo Cruz
GeoRSS	*Geographically Encoded Objects for* RSS
GPRE	*General Purpose Rule Engine*
GPRS	*General Packet Radio Service*
GPS	Sistema de Posicionamento Global (*Global Positioning System*)
GSM	*Global System for Mobile Communications*
HTML	*Hypertext Markup Language*
IA	Inteligência Artificial
IBGE	Instituto Brasileiro de Geografia e Estatística
IDE	*Integrated Development Environment*
KTA	*Knowledge to Action*
LAR	Laboratório de Redes do IFCE

Lariisa	Laboratório de Redes Inteligentes e Integradas de Saúde
Lisa	Lariisa *Integration System Architecture*
OIL	*Ontology Inference Layer*
OWL	*Web Ontology Language*
Paola	Plataforma de Aplicações baseadas em Ontologias do Projeto Lariisa
PSF	Programa Saúde da Família
QA	*Query Adapter*
QoC	*Quality of Context*
QoCE	QoC *Evaluator*
RDF	*Resource Description Framework*
RDFS	RDF *Schema*
Ripass	Rede Interdisciplinar de Pesquisa e Avaliação em Sistemas de Saúde
RNP	Rede Nacional de Ensino e Pesquisa
RSS	*Really Simple Syndication feeds*
SA	*Service Adapter*
SE	Sala de Emergência
SQL	*Structured Query Language*
SUS	Sistema Único de Saúde
SWRL	*Semantic Web Rule Language*
TI	Tecnologia da Informação
TIC	Tecnologia da Informação e Comunicação
TV	Televisão
TVD	TV Digital
TVDI	TVD Interativa
UECE	Universidade Estadual do Ceará
UFC	Universidade Federal do Ceará
UML	Linguagem de Modelagem Unificada (*Unified Modeling Language)*
URI	Identificador Uniforme de Recursos (*Uniform Resource Identifier)*
URL	Localização-Padrão de Recursos (*Uniform Resource Locator)*
W3C	*World Wide Web Consortium*
WiMAX	*Worldwide Interoperability for Microwave Access*
XHTML	*Extensible Hypertext Language*
XML	*Extensible Markup Language*

XSD XML *Schema*

SUMÁRIO

1. INTRODUÇÃO 20

2. FUNDAMENTAÇÃO TEÓRICA 24

2.1. Contexto 24

2.2. Ontologias 26

2.2.1. Classificação 27

2.2.2. Padrões 28

2.2.3. Motivação para o uso de ontologias 30

2.2.4. Ferramentas ontológicas 31

2.3. Governança em Saúde no Brasil 32

2.4. Considerações Finais do Capítulo 35

3. TRABALHOS RELACIONADOS 36

3.1. The Context Toolkit 36

3.2. VadeMecum 38

3.3. A Framework for Developing Mobile, Context-aware Applications 40

3.4. Protégé 41

3.5. Comparação 43

3.6. Considerações Finais do Capítulo 44

4. LARIISA 45

4.1. Modelo de Contexto de Saúde 45

4.2. Configuração de Governança do projeto Lariisa 48

4.2.1. Governança de Gerenciamento de Conhecimento 48

4.2.2. Governança de Normatização Sistêmica 49

4.2.3. Governança Clínico-Epidemiológica 50

4.2.4. Governança Administrativa 51

4.2.5. Governança de Gerenciamento Compartilhado 51

4.3. Knowledge to Action Model 52

4.4. Lariisa Framework Core 53

4.5. Diga Saúde 56

4.6. Lisa: *Lariisa Integration System Architecture* 58

4.7. Sisa 60

4.8. Caminhos do Conhecimento 63

4.9. Considerações Finais do Capítulo 64

5. PAOLA: UMA PLATAFORMA PARA O DESENVOLVIMENTO DE APLICAÇÕES BASEADAS EM ONTOLOGIAS PARA O PROJETO LARIISA 66

5.1. Funcionalidades do Sistema 68

5.1.1. Funcionalidades do módulo de Manipulação de Bases de Conhecimento 72

5.1.2. Funcionalidades do módulo de Manipulação de Provedores de Contexto 74

5.1.3. Funcionalidades do módulo de Manipulação de Regras e Ações 76

5.1.4. Funcionalidades do módulo de Gerenciamento de Informação 79

5.1.5. Funcionalidades do módulo Gerador de Executável & Simulação 81

5.2. Considerações Finais do Capítulo 83

6. ASPECTOS DE IMPLEMENTAÇÃO 86

6.1. Arquitetura do Sistema 86

6.2. Aspectos de Implementação do módulo de Gerenciamento de Base de Conhecimento 88

6.2.1. Manipulação de Bases de Ontologias 88

6.2.2. Manipulação de Bases de Conhecimento 89

6.3. Aspectos de Implementação do módulo de Gerenciamento de Provedores de Contexto 91

6.3.1. Manipulação de Provedores de Contexto 92

6.3.2. Integração com a arquitetura Lisa 93

6.3.3. Captura de Dados 93

6.4. Aspectos de Implementação do módulo de Gerenciamento de Regras e Ações 95

6.4.1. Regras 95

6.4.2. Ações 96

6.5. Aspectos de Implementação do módulo de Gerenciamento de Informação 98

6.6. Considerações Finais do Capítulo 99

7. ESTUDO DE CASO: APLICAÇÃO SISA 101

7.1. Criação de Ontologias 101

7.2. Definição de Provedores de Contexto 104

7.3. Criação de Regras e Ações 104

7.4. Manipulando Informações 105

7.5. Geração de Executável 106

7.6. Considerações Finais do Capítulo 107

CONSIDERAÇÕES FINAIS 108

REFERÊNCIAS 112

APÊNDICE A – CASOS DE USO EXPANDIDOS 115

1. INTRODUÇÃO

Com o advento do Programa Saúde da Família (PSF), do Sistema Único de Saúde (SUS), os serviços oferecidos à população na área da saúde descentralizaram-se. O atendimento, que antes era oferecido apenas em hospitais e postos de saúde, passa a ser oferecido também na residência do paciente. Essa mudança aumentou a complexidade de gestão da informação, pois os atendimentos agora estão distribuídos em áreas muito maiores que as dos hospitais e envolvem um conjunto maior de informação.

Surge a necessidade da criação de novos métodos e ferramentas de gestão da saúde, apoiados por tecnologia da informação (TI), que tratem das especificidades do PSF, contribuam para a oferta de serviços de qualidade, guiem profissionais em suas atividades e auxiliem gestores a tomarem decisões e aplicarem recursos com eficiência.

Sistemas sensíveis ao contexto (SALBER; DEY; ABOWD, 1999) podem trazer bons resultados no apoio à gestão e à tomada de decisão na área de saúde. Esse tipo de sistema processa informações contextuais, ou seja, informações que caracterizam uma determinada situação em que se encontra um usuário e que servem de apoio a oferta de serviços inteligentes.

Na área de sistemas sensíveis ao contexto, o projeto Lariisa atua para dar suporte a aplicações que apoiem a tomada de decisão em governança de saúde. Utiliza informações contextuais, entre elas, do paciente em sua residência, para fornecer inteligência e agilidade para sistemas públicos de saúde em diversas instâncias (administrativa, epidemiológica, legal, entre outras).

Entretanto, o desenvolvimento de aplicação para o projeto Lariisa é de alta complexidade, por envolver a utilização de conceitos de representação de conhecimento, de inferências, entre outros. Além disso, utiliza fontes de dados (provedores de contexto) das mais diversas, como dispositivos móveis, *set-top-boxes*, sensores, entre outros, afetando consideravelmente a produtividade do desenvolvedor e aumentando custos com o desenvolvimento.

Esta dissertação propõe a Paola, uma Plataforma para o desenvolvimento de Aplicações baseadas em Ontologias para o projeto Lariisa. Apoiar o desenvolvedor na construção de aplicações para o projeto Lariisa é o foco principal desta dissertação. É proposta uma ferramenta, apoiada por uma metodologia simples de desenvolvimento, que facilita a criação de aplicações sensíveis ao contexto e que abstraia a complexidade do projeto Lariisa.

Esta dissertação considera as peculiaridades do projeto Lariisa e propõe a criação da plataforma Paola que integra características de bases de conhecimento, provedores de contexto, regras, ações, geração de *software* executável e simulação.

Para a concepção da plataforma Paola, foi realizado um levantamento das dificuldades encontradas no desenvolvimento de aplicações, como as vivenciadas por Antunes (2011) em sua aplicação na área de vigilância epidemiológica, utilizando o projeto Lariisa.

Como contribuição e relevância tecnológica está a aplicação de tecnologias da informação e comunicação (TIC) para apoiar o desenvolvimento de aplicações sensíveis ao contexto através de integração de provedores de contexto e de ferramentas de desenvolvimento. Com esse aparato sistêmico e tecnológico, desenvolvedores de sistemas na área da saúde serão auxiliados no processo de construção de *software*.

O trabalho tem relevância social, visto que pode ser utilizado (indiretamente) para melhoria dos serviços públicos de saúde, contribuindo para uma sociedade mais justa, onde a população tenha acesso a serviços de qualidade na área de saúde. Em momento oportuno, onde iniciativas do Governo Federal (Brasil) e do Governo do Estado do Ceará, como a TV Digital Brasileira, o Cinturão Digital do Ceará e o Sistema Cartão do Departamento de Informática do SUS (Datasus), são criadas condições favoráveis para que aplicações utilizando o projeto Lariisa tenham aplicabilidade e eficácia em suas áreas de atuação. Políticas públicas semelhantes, também em áreas correlatas, estão sendo desenvolvidas em diversos países.

Como resultado, espera-se que o processo de desenvolvimento de aplicações sensíveis ao contexto para o projeto Lariisa se torne mais simples, rápido e barato.

O objetivo geral desta dissertação é especificar uma plataforma para o desenvolvimento de aplicações inteligentes para tomada de decisão em governança de saúde para o projeto Lariisa, baseadas em ontologias.

São objetivos específicos:

- facilitar o desenvolvimento de aplicações no Projeto Lariisa;
- selecionar ferramentas adequadas aos objetivos da plataforma;
- criar um protótipo para de uma plataforma para o desenvolvimento de aplicações;
- descrever uma metodologia que suporta o mecanismo de desenvolvimento;
- realizar um estudo de caso com uma ontologia de domínio de representação da dengue.

Por exemplo, a Tabela 1 exibe o quantitativo de referências encontradas por resultado de pesquisa de palavra-chave na base de indexação do Google Acadêmico (os valores são aproximados).

Tabela 1 – Quantitativo de referências de pesquisa

Palavra-chave	**Quantidade de referências**
Context aware system	2.530.000
Context aware system AND health	110.000
Context aware system AND ontology	148.000
Health AND ontology	130.000
Context aware system AND development	1.510.000
Context aware system AND framework	1.840.000
Ontology	818.000

Fonte: Google Acadêmico (2011).

Para a concepção da plataforma de desenvolvimento proposta nesta dissertação, na etapa de análise e projeto, são criados diagramas arquiteturais, casos de uso (e seus diagramas), dentre outros artefatos da engenharia de *software*.

Por fim, uma reflexão sobre o trabalho é realizada identificando o porquê de cada decisão tomada. Também são apresentados os resultados obtidos e são sugeridos alguns trabalhos futuros.

O restante da dissertação está estruturado da seguinte forma:

- o capítulo 2 contém a fundamentação necessária para a compreensão desta dissertação. São abordados os temas que são a base do conteúdo da dissertação;

- o capítulo 3 contém uma discussão sobre os principais trabalhos relacionados com a dissertação, destacando os pontos relevantes, fortes e fracos de cada um;
- o capítulo 4 contém uma descrição do projeto Lariisa, que é um framework sensível ao contexto para desenvolvimento de aplicações inteligentes para governança em saúde;
- o capítulo 5 propões a Paola: Uma Plataforma Para O Desenvolvimento De Aplicações Baseadas Em Ontologias Para O Projeto Lariisa. É apresentada a arquitetura geral desta proposta, descrevendo o ambiente e os serviços oferecidos;
- no capítulo 6 são apresentados detalhes sobre a implementação dos artefatos do sistema;
- o capítulo 7 apresenta uma aplicação, o Sisa, e é relatado como ocorreu o desenvolvimento de uma aplicação simples para o projeto Lariisa utilizando a plataforma Paola;
- finalmente são apresentadas as considerações finais onde são destacadas as principais contribuições obtidas. São sugeridos alguns trabalhos que podem ser desenvolvidos no futuro e é realizada uma reflexão geral sobre o trabalho e a apresentação de resultados obtidos.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a fundamentação teórica, aborda conceitos e a terminologia básica utilizada no trabalho proposto nesta dissertação.

Os seguintes assuntos são explicitados neste capítulo: Contexto, Ontologias e Governança em Saúde.

2.1. Contexto

Com a popularização de dispositivos móveis, como *smartphones* e *tablets*, os usuários podem realizar tarefas nesses dispositivos enquanto se movimentam. Neste cenário, informações podem ser obtidas da situação na qual o usuário se encontra e podem servir para a oferta de serviços e informações personalizados. Esse tipo informação, que caracteriza uma situação e é utilizada para tomada de decisão, é chamada de contexto (SALBER; DEY; ABOWD, 1999). Já as aplicações que utilizam esse tipo de informação são denominadas aplicações sensíveis ao contexto.

As aplicações sensíveis ao contexto têm utilizado bastante princípios da computação ubíqua[1] para obter informações do contexto do usuário, objetivando auxiliá-los em suas tarefas cotidianas. Um exemplo simples é a utilização de sensores que detectam a presença de pessoas e, automaticamente, acionam a iluminação de um ambiente, de acordo com a localização das pessoas e o horário.

Sistemas sensíveis ao contexto, como o nome sugere, são cientes do estado de um contexto, percebendo suas mudanças. Além de detectar mudanças, eles reagem a essas, fornecendo serviços úteis. Schilit, Adams e Want (1994) definem três aspectos importantes de contexto: onde o usuário está, com quem está e quais recursos estão próximos. Esses aspectos podem sofrer alteração continuamente e uma grande quantidade de informação pode ser derivada deles, como, por exemplo, luminosidade, nível de ruído, conectividade de rede, custos de comunicação, largura de banda, situação social, entre outros.

[1] Computação ubíqua torna a interação homem-máquina imperceptível, integrando a informática a ações e comportamentos naturais das pessoas.

Contexto, segundo Schilit e Theimer (1994), pode ser classificado dentre as três formas a seguir:

- **Contexto computacional**. Está associado às informações do dispositivo que está sendo utilizado. Por exemplo: capacidade de armazenamento.
- **Contexto do usuário**. Está associado às informações intrínsecas do usuário. Por exemplo: batimento cardíaco.
- **Contexto físico**. Está associado às informações do ambiente físico em que o usuário está inserido. Por exemplo: umidade relativa do ar.

Para que seja possível levar em consideração o contexto, é necessário enfrentar vários desafios, dentre os quais Schimidt (2002) destacou:

- Entender como o contexto será utilizado.
- Adquirir informações de contexto.
- Conectar a aquisição de contexto na utilização do contexto.
- Entender a influência da interação entre o homem e o computador.
- Validar sistemas sensíveis ao contexto.

Para que um contexto possa ser representado é necessário que ele seja modelado por alguma técnica. Um modelo de contexto define tipos, nomes, propriedades e atributos das entidades envolvidas nas aplicações sensíveis ao contexto, como usuários, dispositivos móveis e outros. O modelo tenta prever a representação, busca, troca e interoperabilidade de informação contextual entre as aplicações.

Strang e Linnhoff-Popien (2004) afirmam que um modelo bem desenhado é a chave principal para qualquer sistema ciente de contexto. Abowd e Mynatt (2000) afirmam que uma boa representação de contexto deve permitir uma ampla faixa de possibilidades e uma real separação do sensoriamento do contexto e sua reação programável.

Existem diversos requisitos quando se trata de modelos contextuais. Dentre eles, Strang e Linnhoff-Popien (2004) definem alguns a seguir:

- Capacidade do modelo de ser descrito de maneira distribuída.
- Facilidade para detecção de conhecimento contextual inválido, de acordo com a descrição do modelo.
- Riqueza e qualidade da informação.
- Incompletude e ambiguidade da informação.
- Nível de formalidade.
- Aplicabilidade em ambientes existentes.

Existem várias técnicas para modelagem de contexto, algumas delas são: modelos chave-valor, modelos baseados em esquemas de marcação, modelos gráficos, modelos orientados a objetos, modelos baseados em lógica e modelos baseados em ontologias. Essa última utiliza ontologias para descrever as entidades do sistema e seus relacionamentos. Essa técnica é uma das mais promissoras, devido ao seu grande poder de descrição e expressão. Modelos baseados em ontologias é a técnica de modelagem de contexto utilizada nesta dissertação.

2.2. Ontologias

A quantidade de informações presentes na *web* cresce a cada dia. Cada vez mais pessoas estão se conectando à rede mundial de computadores, criando e compartilhando dados. Entretanto, as formas tradicionais de recuperação da informação não retratam a semântica dos dados, seus relacionamentos e o conhecimento que eles representam. Para que haja um crescimento sustentável da representatividade da informação, é preciso tratar essa grande massa de informação de maneira adequada. A *web* semântica ajuda as máquinas compreenderem o significado de informações na rede mundial de computadores. "A *Web* Semântica não é uma *web* separada, mas uma extensão da atual. Nela a informação é dada com um significado bem definido, permitindo melhor interação entre os computadores e as pessoas" (BERNERS-LEE; HENDLER; LASSILA, 2001).

O *World Wide Web Consortium* (2009) expôs os objetivos da web semântica:

> A proposta da Web Semântica é estender os princípios dos documentos da web para os dados. Os dados podem ser acessados usando a arquitetura

> Web (URI[2], por ex.), e estar relacionados uns com os outros da mesma forma que os documentos já são. Isso também significa criar uma plataforma comum que permita o compartilhamento e a reutilização dos dados por meio das fronteiras das aplicações, empresas e comunidades, podendo ser processados automaticamente tanto por ferramentas quanto manualmente, também revelando novos relacionamentos possíveis entre porções de dados.

Para se construir uma *web* semântica é necessária a criação e implantação de padrões tecnológicos que estabelecem a semântica para o compartilhamento de informações entre sistemas. É preciso criar mecanismos que descrevam dados e representem a codificação de significados compartilhados. Um desses mecanismos é definido por meio de ontologias.

A Ciência da Informação (CI) estuda os fenômenos relativos à informação nos mais variados aspectos, numa tentativa de entender e acompanhar os seus desdobramentos para que no futuro seja disponibilizado nos sistemas de informação. Na busca pelo entendimento, do ponto de vista do sujeito, as características sociais e técnicas estão contempladas na ação de produzir, sistematizar, organizar, difundir e recuperar informação. E essas informações são amparadas por ferramentas, objetos, processos e manifestações culturais, sociais e organizacionais.

> As ontologias oferecem um meio de lidar com a representação de recursos de informação: o modelo de domínio descrito por uma ontologia pode ser usado como uma estrutura unificadora para dar semântica e uma representação comum à informação *(GARCIA; ANTUNES; SANTOS, 2010).*

2.2.1. Classificação

Segundo Guarino (1998) as ontologias podem ser classificadas quanto ao seu nível de dependência em relação às tarefas assumidas como mostrado na Figura 1 e detalhadas a seguir.

[2] Identificador Uniforme de Recursos (URI) (*Uniform Resource Identifier* (em inglês)) é uma cadeia de caracteres compacta usada para identificar ou denominar um recurso na Internet. O principal propósito desta identificação é permitir a interação com representações do recurso através de uma rede.

- **Alto nível:** trata conceitos gerais como espaço, tempo, objeto, evento, ação e assunto, podem atuar de maneira independente a determinado problema ou domínio.
- **Domínio:** trata os vocabulários pertinentes ao domínio comum, como “medicina”, “automóvel”, etc., por meio da especificação de conceito inserido nas ontologias de alto nível.
- **Tarefa:** trata atividades ou tarefas comuns, como “diagnóstico”, “venda”, etc., por meio da especificação de conceitos inseridos nas ontologias de alto nível.
- **Aplicação:** delineia o conceito condicionado a uma ontologia de um domínio em especifico ou de uma tarefa. Consiste, em geral, nas definições dos dois tipos de ontologias no mesmo momento (domínio e tarefa).

As ontologias de alto nível unem as definições utilizadas por um grande grupo de usuários. Os outros tipos de ontologias são especificações daquelas, sendo as ontologias de aplicação as mais específicas.

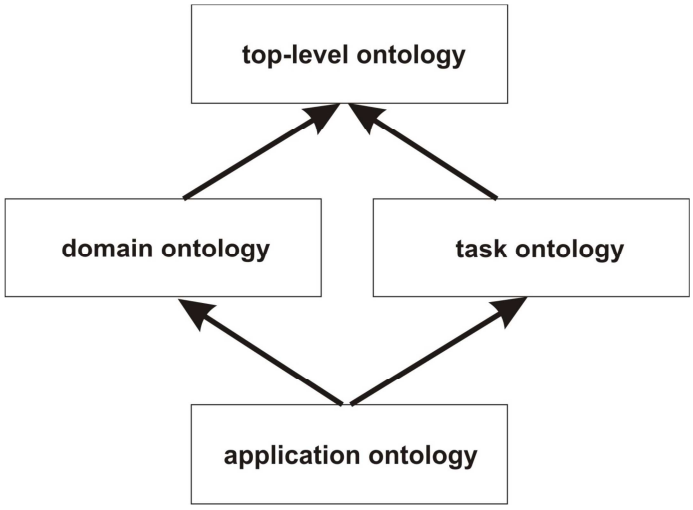

Figura 1 - Tipos de Ontologias (GUARINO, 1998)

2.2.2. Padrões

Conforme Breitman (2005) há necessidade de novas linguagens que sejam capazes de representar o conteúdo semântico na web, obtendo o uso da ontologia na *web* semântica.

Inicialmente, era utilizado o padrão XML, porém, esse padrão não possuía suporte à semântica. Depois surgiu o padrão RDF, que “possui semântica simples baseada em associações com identificadores” (GARCIA; ANTUNES; SANTOS, 2010).

As atuais linguagens fazem o uso de lógica descritiva, fornecendo semântica formal e suporte a inferência. Entre as principais linguagens utilizadas na Web semântica, destacam-se:

- **OIL** (*Ontology Inference Layer*): O OIL é uma linguagem que apresenta compatibilidade com RDFS, além de combinar com a modelagem ontológica fundamentada em frames com a capacidade de inferência e a semântica formal da lógica descritiva.
- **DAML** (DARPA *Agent Markup Language*): Com base no RDFS, a linguagem DAML ou DAML-ONT incluiu recursos suplementares como tipos de dados, enumerações e outras facilidades. Em 2001 a linguagem DAML-ONT foi combinada a OIL.
- **DAML + OIL**: é constituído por um grupo de tripla RDF, apresentando um vocábulo específico da linguagem. Além das triplas definições podem apresentar declarações RDF e combinações semânticas adicionais, mas o efeito dessa combinação é ignorado pela linguagem (Connolly et al. 2001). Sua estrutura é dividida em duas partes independentes. A primeira é responsável pelo domínio dos tipos de dados, onde todos são obtidos do XML. A segunda é responsável por constituir o domínio dos objetos, onde são componentes das classes definidas na ontologia.

O padrão atualmente indicado pelo W3C é o OWL – *Web Ontology Language* (WORLD WIDE WEB CONSORTIUM, 2009). A OWL descreve classes, propriedades e relações entre objetos conceituais que provêm facilidades a interpretação das máquinas de inferências sobre o conteúdo Web.

A OWL é definida como um vocabulário, da mesma maneira que o RDF e o RDFS, mas é enriquecida com semântica. Essencialmente, uma ontologia criada no padrão OWL é uma coleção de triplas RDF. A definição de OWL é organizada em três sub-linguagens:

- **OWL Lite** é a sub-linguagem sintaticamente simples. Destina-se a situações em que apenas são necessárias restrições e uma hierarquia de classe simples. Por exemplo, o OWL *Lite* pode fornecer uma forma de migração para tesauros[3] existentes, bem como de outras hierarquias simples.
- **OWL DL** é mais expressiva que OWL Lite e baseia-se em lógica descritiva, um fragmento de lógica de primeira ordem, passível, portanto, de raciocínio automático. É possível, assim, computar automaticamente a hierarquia de classes e verificar inconsistências na ontologia.
- **OWL *Full*** é a mais expressiva das três sub-linguagens. Destina-se a situações onde alta expressividade é mais importante do que garantir a decidibilidade ou completude da linguagem. Não é possível efetuar inferências em ontologias OWL *Full*.

Existem alguns requisitos do W3C definidos para linguagens de descrição de ontologias. Os mais importantes são: a estrutura da linguagem deve ser compatível com XML, seguir restrições lógicas e suportar a definição de vocabulários de ontologias.

2.2.3. Motivação para o uso de ontologias

Segundo Lopes (2012), o uso das ontologias em especial na área de Ciência da Computação permite ou simplifica a comunicação entre distintas pessoas e sistemas computacionais que participam do mesmo domínio de conhecimento, mas não necessariamente compartilham uma mesma forma de conceituação acerca dos componentes desse domínio.

A padronização da linguagem formal para representar o conhecimento possibilitou o reuso e o compartilhamento do conhecimento. Desse modo, simplificando a leitura e a interpretação da ontologia em outros domínios, possibilitaram-se ações como a inserção ou retirada de conceitos, axiomas e relacionamentos para, enfim, adequação em um novo domínio.

Para Gruninger (1996), outro motivo importante para o uso das ontologias é a importância da confiabilidade acerca dos conceitos do vocabulário ou da linguagem

[3] Tesauro é uma linguagem documentária caracterizada pela especificidade e pela complexidade existente no relacionamento entre os termos que comunicam o conhecimento especializado. É um vocábulo controlado formado por termos (descritores) semanticamente relacionados e atuam como instrumentos de controle terminológico.

que são usados em determinados ambientes. Dessa forma, com o uso da representação formal adquirida com essa aplicação, possibilita-se a automação da verificação de consistência, gerando ambientes mais confiáveis.

Dentre algumas vantagens da utilização das ontologias na área da ciência da computação, podem ser destacadas, segundo Lopes (2012):

- A representação do conhecimento utilizando vocabulário, uma vez que este evita interpretações equívocas.
- Compartilhamento e reuso da ontologia, para que seja um molde apropriado para determinado domínio.
- A precisão do conhecimento disponibilizado por uma ontologia, em relação a sua escrita em linguagem formal, a fim de evitar distorções semânticas como as existentes na linguagem natural, que podem ter semântica completamente diferente dependendo do contexto.
- A possibilidade de realizar o mapeamento da linguagem da ontologia, sem modificar a conceituação, podendo ser mencionada em diversas linguagens.
- Permitir o uso de ontologia genérica capaz de estender-se de forma adequada a um determinado domínio.

2.2.4. Ferramentas ontológicas

Segundo Moura (2009), o termo ferramentas ontológicas vem sendo usado para denominar os estudos conceituais específicos da área do conhecimento, partindo de um mapeamento das suas categorias gerais. Com base nesses mapeamentos conceituais, têm sido implementados programas computacionais especialistas tendo como objetivo oferecer um instrumento dinâmico de produção organizado e compartilhado de conhecimento.

São partes pertencentes à categoria de ferramentas ontológicas os seguintes elementos: as linguagens de indexação (verbais e simbólicas), as taxonomias, os mapas conceituais, as ontologias, Sistemas de Classificação Distribuída (*Distributed Classification Systems*-DCSs) e a mais atual, as chamadas *folksonomias* (MOURA, 2009).

Existem, também, algumas ferramentas e API’s que auxiliam a construção de ontologias. Entre elas o Protégé e o Jena.

O Protégé reúne um editor de ontologias e um *framework* de base de conhecimento gratuito e de código-fonte aberto (*open-source*).Ele é baseado em Java, é extensível, e provê um fácil acoplamento para uma rápida prototipagem e desenvolvimento de aplicações.

Jena é uma API, escrita na linguagem orientada a objetos Java, para manipulação dinâmica de modelos RDF e OWL. Foi desenvolvido por Brian MacBride, e caracteriza-se por possibilitar a criação e manipulação de grafos RDF, representadas pelos recursos (*resources)*, propriedades (*properties*) e literais (*literals*), formando tuplas que irão dar origem aos objetos criados pela linguagem Java.

A API Jena pode ser utilizada na área semântica, propondo codificações e busca de informações nos arquivos RDF e OWL. Esses arquivos, normalmente codificados como arquivos XML e inseridos em páginas XHTLM[4], possibilitarão aos motores de busca da *Web* localizar palavras por meio de métodos da API, objetivando uma maior precisão na obtenção dos resultados.

2.3. Governança em Saúde no Brasil

No Brasil, os serviços de saúde oferecidos à população pelo poder público são de responsabilidade do Ministério da Saúde e das Secretarias Estaduais e Municipais de Saúde. Essas três esferas formam a governança da saúde brasileira e regulam a área da saúde e os serviços ofertados por entidades públicas e privadas, através do SUS, um sistema que contribui para a gestão sistêmica da saúde brasileira.

Segue a descrição do SUS obtida do Portal da Saúde (2012) do Ministério da Saúde do Brasil.

> O Sistema Único de Saúde - SUS - foi criado pela Constituição Federal de 1988 e regulamentado pelas Leis n.º 8080/90 e nº 8.142/90, Leis Orgânicas da Saúde, com a finalidade de alterar a situação de desigualdade na assistência à Saúde da população, tornando obrigatório o atendimento

[4] XHTML, ou *Extensible Hypertext Markup Language*, é uma linguagem de marcação HTML baseada em XML. A linguagem combina as *tags* do HTML com as regras do XML. É utilizada principalmente para melhorar acessibilidade em diversos dispositivos (televisão, *smartphone*, celular, etc.).

> público a qualquer cidadão, sendo proibidas cobranças de dinheiro sob qualquer pretexto.
> Do Sistema Único de Saúde fazem parte os centros e postos de saúde, hospitais - incluindo os universitários, laboratórios, hemocentros, bancos de sangue, além de fundações e institutos de pesquisa, como a FIOCRUZ - Fundação Oswaldo Cruz e o Instituto Vital *Brazil*. Através do Sistema Único de Saúde, todos os cidadãos têm direito a consultas, exames, internações e tratamentos nas Unidades de Saúde vinculadas ao SUS da esfera municipal, estadual e federal, sejam públicas ou privadas, contratadas pelo gestor público de saúde.
> O SUS é destinado a todos os cidadãos e é financiado com recursos arrecadados através de impostos e contribuições sociais pagos pela população e compõem os recursos do governo federal, estadual e municipal.
> O Sistema Único de Saúde tem como meta tornar-se um importante mecanismo de promoção da eqüidade no atendimento das necessidades de saúde da população, ofertando serviços com qualidade adequados às necessidades, independente do poder aquisitivo do cidadão. O SUS se propõe a promover a saúde, priorizando as ações preventivas, democratizando as informações relevantes para que a população conheça seus direitos e os riscos à sua saúde. O controle da ocorrência de doenças, seu aumento e propagação - Vigilância Epidemiológica, são algumas das responsabilidades de atenção do SUS, assim como o controle da qualidade de remédios, de exames, de alimentos, higiene e adequação de instalações que atendem ao público, onde atua a Vigilância Sanitária.
> O setor privado participa do SUS de forma complementar, por meio de contratos e convênios de prestação de serviço ao Estado quando as unidades públicas de assistência à saúde não são suficientes para garantir o atendimento a toda a população de uma determinada região (PORTAL DA SAÚDE, 2012).

No setor privado de planos de saúde o governo atua regulando os serviços prestados, através da Agência Nacional de Saúde Suplementar (ANS), órgão vinculado ao Ministério da Saúde. Através dessa agência o governo define medidas e ações que envolvem a criação de normas e o controle e a fiscalização de segmentos de mercado explorados por empresas, para assegurar o interesse público.

A gestão da saúde brasileira envolve o SUS, os planos de saúde privados e, em um nível mais operacional, as entidades públicas, privadas e filantrópicas que atuam no território nacional.

Para a atenção básica[5] no SUS, existe o Programa Saúde da Família, definido a seguir:

[5] Atenção básica, ou atenção primária, é um conjunto de ações de saúde, no âmbito individual ou coletivo, que abrangem a promoção e a proteção da saúde, a prevenção de agravos, o diagnóstico, o tratamento, a reabilitação e a manutenção da saúde.

> Estratégia prioritária adotada pelo Ministério da Saúde para a organização da atenção básica, no âmbito do SUS, dispondo de recursos específicos para seu custeio. É responsável pela atenção básica em saúde de uma área determinada. Cada equipe (médico, enfermeiro e auxiliar de enfermagem) deve atender no mínimo 2.400 e no máximo 4.500 pessoas, podendo solucionar 80% dos casos em saúde das pessoas sob sua responsabilidade (MINISTÉRIO DA SAÚDE, 2012).

A Saúde da Família é operacionalizada mediante a implantação de equipes multiprofissionais em unidades básicas de saúde. Essas equipes são responsáveis pelo acompanhamento de famílias em uma área geográfica delimitada.

Quadro 1 – Atividades dos profissionais do PSF

Profissional	Atividades
Agente Comunitário de Saúde	Realizar mapeamento de sua área de atuação; Manter dados cadastrais das famílias; Identificar indivíduos em situação de risco; Identificar área de risco; Realizar visita domiciliar para acompanhamento de famílias; Orientar famílias para utilização adequada de serviços de saúde, encaminhando-os para os serviços necessários.
Médico	Realizar assistência integral; Consultas clínicas; Encaminhar pacientes para serviços de média e alta complexidade; Encaminhar para internação hospitalar.
Enfermeiro	Realizar assistência integral; Realizar consultas; Realizar procedimentos; Solicitar exames.
Cirurgião Dentista	Realizar diagnóstico; Realizar procedimentos clínicos; Realizar atenção integral à saúde bucal; Encaminhar pacientes para serviços de alta complexidade.
Gestor	Alocar profissionais para visita em domicílio; Aprovar encaminhamentos.

Dados do Panorama da Saúde do Brasil, publicação do Instituto Brasileiro de Geografia e Estatística - IBGE (2008), apontam que mais da metade dos lares brasileiros, cerca de 27,5 milhões de residências – de um total de 57,6 milhões, está cadastrada no PSF. São cerca de 96 milhões de pessoas que recebem, de alguma forma, os serviços de equipes do PSF. A maior parte das famílias atendidas vive na Região Nordeste, que concentra 9,7 milhões de famílias, em seguida o Sudeste (9,1 milhões), Sul (4,5 milhões) e Norte (2 milhões).

Segundo Vieira (2010), os profissionais do PSF visitam as casas mais pobres onde vivem pessoas com baixo nível de instrução.

2.4. Considerações Finais do Capítulo

Neste capítulo, foram abordados conceitos que o projeto Lariisa utiliza para a criação de aplicações sensíveis ao contexto: sensibilidade ao contexto e ontologias. Esses dois conceitos também são muito utilizados na especificação e na implementação da plataforma Paola (capítulos 5 e 6) assim como durante a construção e utilização da aplicação descrita no capítulo 7, um estudo de caso aplicando a plataforma Paola.

Foram tratados também aspectos de governança de saúde no Brasil, com o objetivo de compreender como a saúde pública é organizada no país.

3. TRABALHOS RELACIONADOS

Este capítulo apresenta um levantamento da literatura sobre as principais ferramentas para desenvolvimento de aplicações sensíveis ao contexto. Foram pesquisadas ferramentas em bases acadêmicas, em sistemas comerciais e em bases de *software* livre.

No final do capítulo é feita uma comparação entre as ferramentas pesquisadas.

3.1. The Context Toolkit

The Context Toolkit, proposto por Salber, Dey e Abowd (1999), é um conjunto de ferramentas para desenvolvimento de aplicações sensíveis ao contexto que trabalha as diferenças entre o contexto e a entrada de usuário. *The Context Toolkit* consiste em três principais abstrações, detalhadas a seguir.

- ***Widgets***. Os *context widgets* mediam a comunicação entre o usuário e o ambiente sensível ao contexto. Encapsulam informações sobre uma unidade de contexto, como localização por exemplo. Os *widgets* proveem uma *interface* uniforme para componentes e aplicações usarem o contexto, ocultando detalhes dos mecanismos de sensoriamento de contexto.
- ***Aggregators***. Os *context aggregators* são definidos como meta-widgets, possuindo as mesmas características dos *widgets* e, além disso, têm a capacidade de agregar informações de contexto de entidades do mundo real. É a porta de comunicação entre aplicações e os *widgets*, possuindo um nível ainda maior de abstração dos mecanismos de sensoriamento de contexto.
- ***Interpreters***. O *context interpreter* é utilizado para abstrair ou interpretar informação de contexto de baixo nível para um nível maior de abstração.

The Context Toolkit torna transparente a natureza distribuída de aplicações sensíveis ao contexto. As aplicações não necessitam conhecer se os componentes de contexto são executados localmente ou remotamente, facilitando o trabalho do desenvolvedor. Os componentes compartilham um mecanismo de comunicação

comum (XML sobre HTTP) e executam independentemente da aplicação e pode ser utilizada por múltiplas aplicações.

A Figura 2 ilustra a arquitetura de contexto do *The Context Toolkit* no suporte a uma aplicação sensível ao contexto, exemplificando uma configuração de componentes de contexto e os conceitos de *widgets*, *aggregators* e *interpreters*.

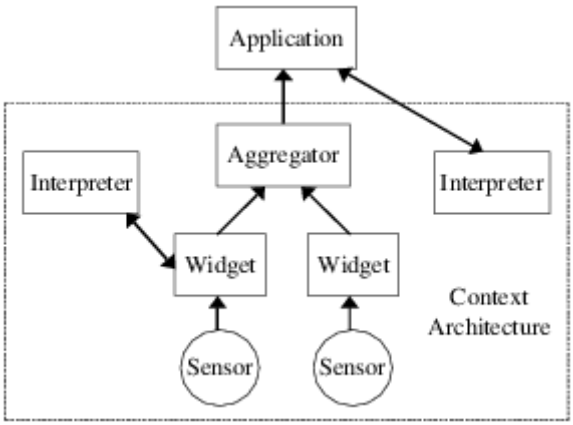

Figura 2 – Exemplo de configuração de componentes de contexto. Setas indicam fluxo. (SALBER; DEY; ABOWD, 1999)

Salber, Dey e Abowd (1999) propuseram algumas aplicações construídas utilizando o *The Context Toolkit* que serviram como prova de conceito do projeto:

- um quadro de entrada/saída que exibe quais membros do grupo de pesquisa estão no prédio;
- uma tela que exibe informações relevantes de uma pessoa que está situada em sua frente;
- uma lista de *e-mails* que envia mensagens apenas para os assinantes que estão localizados atualmente no prédio;
- um quadro-branco eletrônico que inicia a gravação de anotações e de áudio quando detecta a presença de usuários em sua frente;
- um assistente de conferências que informa sobre apresentações interessantes para um pesquisador, guardando anotações suas e de sua equipe.

The Context Toolkit também possui técnicas para tratar dados contextuais ambíguos e para tratar a privacidade dos dados, diferenciando dados públicos e

privados. No projeto também é estudado os aspectos de como a ferramenta pode ser utilizada para o reuso de software.

3.2. VadeMecum

Proposto por Figueiredo (2009), o VadeMecum tem como principal objetivo facilitar o desenvolvimento de aplicações sensíveis ao contexto. A infraestrutura criada é formada pelos componentes a seguir.

- **Servidor de contexto**. Responsável pela aquisição, armazenamento, inferência e monitoramento das informações contextuais. Opera baseado em regras contextuais, que indicam que ações devem ser executadas nas aplicações quando um determinado estado for atingido.
- **CARE**. Acrônimo de *Context-Aware Rule Editor*. Utilizado na edição de regras contextuais, pelo usuário final, no servidor de contexto do VadeMecum.
- **CARE *Emulator***. Emulador para validação das regras contextuais. O usuário pode selecionar estados contextuais possíveis e verificar se uma ação desejada foi executada quando um determinado estado for escolhido.

A Figura 3 ilustra a estrutura e o fluxo do sistema criado e nela estão contemplados os conceitos do Servidor de Contexto, do CARE, do CARE *Emulator*, das regras, do Modelo de Contexto, das ações e de uma aplicação.

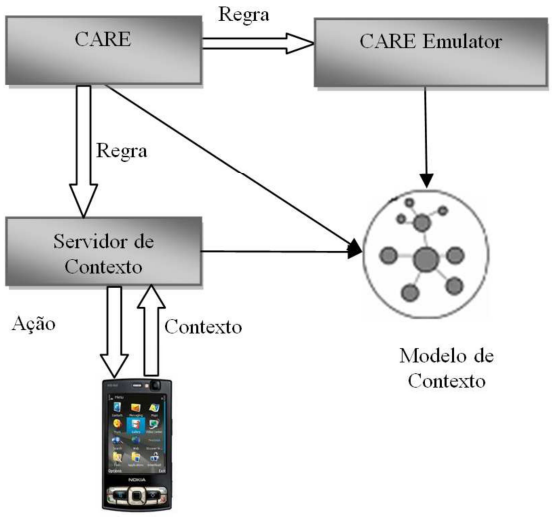

Figura 3 – Esquema de criação de regras do VadeMecum (FIGUEIREDO, 2009)

Juntamente com o Servidor de Contexto do VadeMecum está o Modelo de Contexto. Figueiredo (2009) adotou a utilização de ontologias na linguagem OWL para a modelagem de contexto, devido à sua alta expressividade. O modelo proposto, além de informações contextuais, descreve as entidades e relacionamentos envolvidos na criação de regras e na apresentação de ações para os usuários.

Como estudo de caso Figueiredo (2009) desenvolveu uma aplicação multimídia simples: *Outdoor* Virtual. Na aplicação, o usuário deseja que sejam exibidas propagandas multimídia para possíveis clientes de acordo com o contexto deles. Por exemplo, se o usuário estiver em uma determinada localização na época de acontecimento de um grande evento festivo, ele receberá arquivos multimídia sobre aquele evento. A Figura 4 ilustra a aplicação.

Figura 4 – Outdoor Virtual – aplicação móvel multimídia desenvolvida com o VadeMecum (FIGUEIREDO, 2009)

Para desenvolver o *Outdoor* Virtual, Figueiredo (2009) utilizou as ferramentas criadas e de provedor de contexto de localização geográfica, provedor de status do usuário e sua agenda e provedor das preferências do usuário. Com essas informações foi possível traçar o contexto e os interesses do usuário para poder oferecer o melhor serviço multimídia possível. Foram criadas regras com o auxílio do

CARE, que também possui suporte a geoprocessamento com mapas, e uma aplicação multimídia simples.

3.3. A Framework for Developing Mobile, Context-aware Applications

Biegel e Cahill (2004) propuseram um *framework* para o desenvolvimento de aplicações móveis sensíveis ao contexto. O *framework* (será chamado apenas assim nesta subseção) permite que o desenvolvedor reúna dados de diferentes sensores, represente o contexto da aplicação e processe semanticamente o contexto, sem necessitar escrever um código complexo. Utiliza um paradigma de comunicação baseado em eventos especialmente feito para redes sem fio *ad-hoc*.

O *framework* fornece ao desenvolvedor o aparato necessário para facilitar o projeto, o protótipo e o teste, sendo possível inserir em um ambiente com processo de desenvolvimento interativo e rápido. Além disso, Biegel e Cahill afirmam que *designers* e usuários finais podem construir suas próprias aplicações utilizando o *framework.*

O modelo proposto provê uma abordagem sistemática do desenvolvimento de aplicações sensíveis ao contexto no ambiente móvel *ad-hoc*. Para isso os seguintes pontos chaves estão contemplados no *framework*.

- Provê abstrações para sensores e atuadores, poupando o desenvolvedor da responsabilidade de tratar a interação em baixo nível com uma variedade de dispositivos de *hardware*.
- Provê um mecanismo probabilístico para reunir fragmentos de dados capturados de sensores e fornecer informação contextual de alto nível.
- Provê uma abordagem eficiente para o raciocínio inteligente baseado em hierarquia de contextos.
- Provê um mecanismo de comunicação baseado em eventos para a interação entre sensores, objetos e atuadores.
- Provê uma ferramenta para a programação visual de fácil utilização para o desenvolvimento de aplicações, reduzindo a necessidade de escrita de código-fonte.

A Figura 5 ilustra os objetos sensíveis ao contexto (*sentient object*) e seu relacionamento com outros componentes no *framework*. Esses objetos podem ser sensores ou atuadores. A atuação dos objetos é controlada com base na entrada dos sensores para uma lógica interna de controle, que consiste em filtros, fusão de informação, inferência, entre outros.

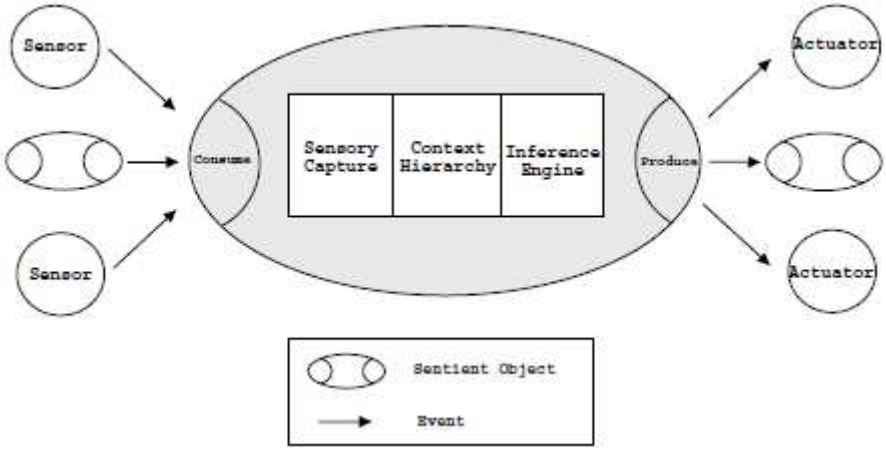

Figura 5 – *Sentient object model* proposto por Biegel e Cahill (2004)

Sensores são definidos como entidades que produzem eventos de software em uma reação a estímulos do mundo real. No modelo da Figura 5 é possível ver o *consume*, módulo do *framwork* responsável por receber estes eventos.

Para validar o *framework* Biegel e Cahill (2004) desenvolveram uma aplicação que usa um modelo sensível ao contexto de carros que trafegam e obedecem semáforos automaticamente.

3.4. Protégé

Protégé é editor de ontologias *open-source* e um *framework* para manipulação de bases de conhecimento. A ferramenta suporta a modelagem de ontologias via editores de Protégé-*Frames* e Protégé-OWL. As ontologias no Protégé podem ser exportadas para uma variedade de formatos, como RDF(S), OWL, XML *Schema*, entre outros.

A ferramenta é baseada em Java, é extensível e provê um ambiente *plug-and-play* que favorece, com flexibilidade, a rápida prototipagem e desenvolvimento de aplicações. O Protégé é utilizado por uma grande comunidade de

desenvolvedores, acadêmicos, setor público e usuários corporativos. É também aplicado para soluções de conhecimento em diversas áreas como biomedicina, captação de inteligência, modelagem de negócios, entre outros.

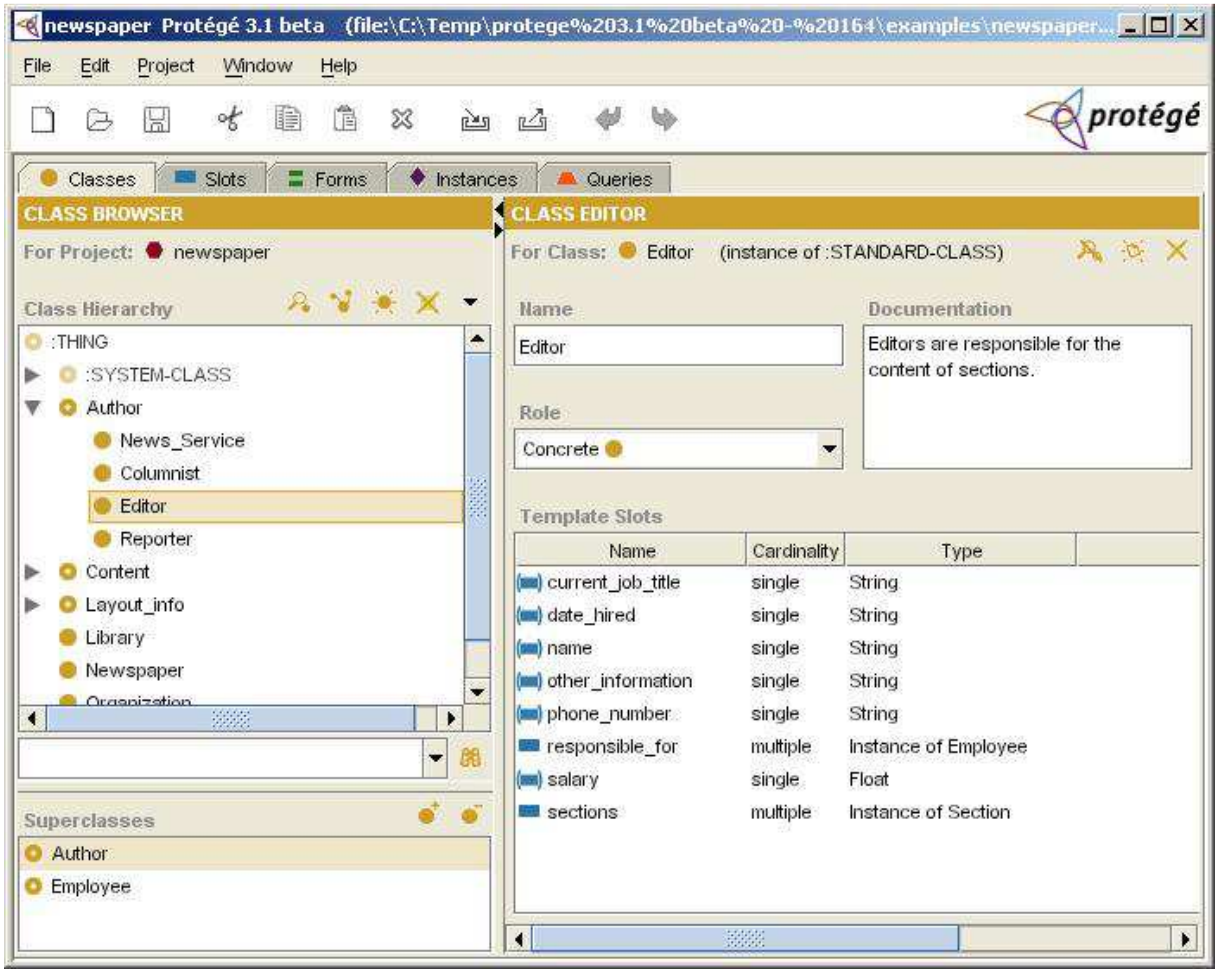

Figura 6 - Protégé

O Protégé é uma das ferramentas mais utilizadas para a criação de ontologias na *web*. Ele não é utilizado para desenvolvedor uma aplicação sensível ao contexto, mas pode ser utilizado para definir ontologias, que, por sua vez, são utilizadas nas aplicações *context-aware*.

A Figura 6 ilustra o editor do Protégé, onde é possível ver algumas de suas funções, como *classes, slots, forms, instances* e *queries.*

3.5. Comparação

Nesta seção é apresentada uma comparação entre os trabalhos relacionados descritos neste capítulo. Para efeitos de comparação as características a seguir foram consideradas relevantes para o propósito do trabalho do qual trata esta dissertação.

1. Auxilia o desenvolvimento de aplicações sensíveis ao contexto.
2. Auxilia a execução de aplicações sensíveis ao contexto.
3. Auxilia a simulação de aplicações sensíveis ao contexto.
4. Modelo de contexto baseado em ontologias.
5. Aquisição de dados de sensores físicos e virtuais (integração com provedor de contexto).
6. Definição de regras.
7. Definição de ações.
8. Agregador de contexto.
9. Raciocinar com base no contexto.
10. API para consulta a informação.

A Tabela 2 exibe um comparativo entre os trabalhos relacionados. Nela é realizado um cruzamento do trabalho relacionado com as características enumeradas anteriormente (verificar numeração). Os campos marcados com '+' indicam que a ferramenta em questão possui determinada característica, os marcados com '-' indicam que não a possuem.

Tabela 2 – Comparação entre os trabalhos relacionados:

Característica / Ferramenta	1	2	3	4	5	6	7	8	9	10
The Context Toolkit (SALBER; DEY; ABOWD, 1999)	+	-	-	-	+	-	-	-	-	-
VadeMecum (FIGUEIREDO, 2009)	+	+	+	+	-	+	+	-	-	-
A Framework for Developing Mobile, Context-aware Application (BIEGEL; CAHILL, 2004)	+	-	-	-	+	-	-	+	+	-
Protégé (STANFORD UNIVERSITY SCHOOL OF MEDICINE, 2012)	+	+	-	+	-	-	-	-	+	+

É possível ver na Tabela 2 que todos os trabalhos relacionados listados tratam-se de ferramentas que auxiliam o desenvolvimento de aplicações sensíveis

ao contexto. É possível ver também que há diferenças, entre as ferramentas analisadas, no que diz respeito às características elencadas nesta seção.

3.6. Considerações Finais do Capítulo

Neste capítulo foram apresentados quatro trabalhos e foi possível analisar as especificidades de cada um e as diferenças entre eles. Os três trabalhos abordam o desenvolvimento de aplicações sensíveis ao contexto, porém cada um aborda o tema de forma ligeiramente diferente e tem uma contribuição específica individual bem definida. O *The Context Toolkit* (SALBER; DEY; ABOWD, 1999) tem o foco principal nos *widgets* e como integrá-los de forma eficiente em uma aplicação sensível ao contexto. O VadeMecum (FIGUEIREDO, 2009) tem como principal funcionalidade a edição de regras contextuais. Já o trabalho desenvolvido por Biegel e Cahill (2004) contribui principalmente na agregação de dados contextuais e nas inferências em modelos contextuais. Já o Protégé foca principalmente na edição de ontologias.

Foi possível também constatar que é possível construir uma ferramenta para o desenvolvimento de aplicações sensíveis ao contexto utilizando diversas tecnologias e diferentes metodologias de desenvolvimento. As experiências vivenciadas pelos autores dos trabalhos mostrados foram bastante enriquecedoras e contribuem positivamente para a concepção da proposta desta dissertação.

4. LARIISA

"O Lariisa é um *framework* para tomada de decisão em governança para sistemas públicos de saúde" (OLIVEIRA et al., 2010). Engloba e integra dados sobre residências de famílias em um novo sistema inteligente de informação para cuidados de saúde. Para apoiar as interações com o usuário final, o *framework* foi construído sobre o *middleware* GINGA, desenvolvido para a TV Digital Brasileira, que terá acesso universal até o ano 2015. Com base em cinco domínios de governança: conhecimento, normativo, clínico-epidemiológico, administrativo e gerenciamento compartilhado, o *framework* conta uma infraestrutura de comunicação óptica e sem fio (WiMAX), o Cinturão Digital do Ceará, que segundo a Etice[6] (2011), é projeto para oferecer banda larga de alta velocidade, sendo a maior e mais veloz rede pública de dados do Brasil, cobrindo 82% da população urbana do estado. São mais de 2.500Km de fibra ótica de alta velocidade (1 a 2Gbps), conectando escolas, hospitais, postos de saúde, delegacias e demais órgãos públicos. Servirá para telemedicina, educação à distância, TV digital, videoconferência e outros serviços para o desenvolvimento do estado e a qualidade de vida da população.

O projeto Lariisa está centrado no conceito de informação de contexto de saúde, caracterizando situações de entidades em um sistema de saúde. Uma entidade é, por exemplo, um membro da família, um agente de saúde, gestor da saúde, entre outros, que são considerados relevantes para as interações entre um usuário e um sistema de saúde, a fim de fornecer sistemas inteligentes.

4.1. Modelo de Contexto de Saúde

É necessário definir um modelo de contexto formal em saúde, a fim de facilitar a representação do contexto, compartilhamento e interoperabilidade semântica no sistema de governança da saúde. Para este fim, o projeto Lariisa define duas ontologias OWL-DL (ver Figura 7 e Figura 8) para a modelagem de informações de contexto local e global de saúde. Contexto de saúde local descreve a situação de qualquer entidade interagindo com o sistema de governança, tais como usuários

[6] A Empresa de Tecnologia da Informação do Estado do Ceará (Etice) é uma empresa pública responsável pelos serviços de TICs no Governo do Estado do Ceará, entre eles o Cinturão Digital.

finais (pacientes), gestores de saúde, agentes de saúde, dentre outros. Essa informação é utilizada para a definição de regras de decisão locais de saúde e para construir o contexto de saúde global. O contexto global de saúde descreve informações de alto nível, derivado do contexto de saúde local, e é utilizado para tomada de decisão em governança de saúde. Por exemplo, ele descreve o número de casos de dengue confirmados em uma região (ex: bairro, cidade, comunidade), durante um determinado período de tempo (ex: um dia, uma semana). Portanto, essas informações podem ser vistas como indicadores globais utilizados para melhorar as decisões de governança.

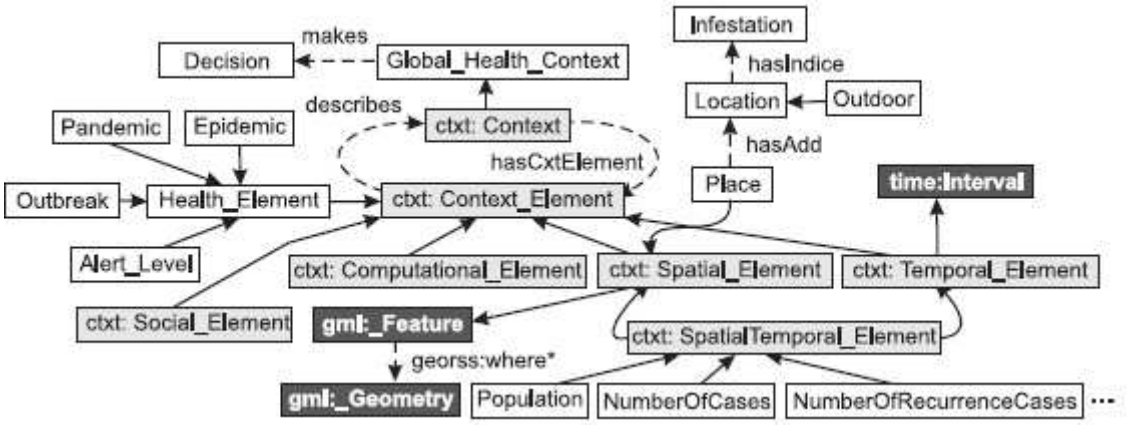

Figura 7 – Modelo Global de Saúde do projeto Lariisa (OLIVEIRA et al., 2010)

A classificação dos contextos local e global de saúde é realizada em seis dimensões:

- Espacial – quaisquer informações que caracterizem a situação da dimensão espacial (ex: localização, local, coordenadas GPS).
- Temporal – quaisquer informações que caracterizem a situação da dimensão do tempo (ex: instante, intervalo, período do dia, período do mês, período do ano, estação).
- Espaço-Temporal – quaisquer informações que caracterizam a situação que é dependente tanto da dimensão espacial quanto da dimensão temporal (ex: condições climáticas, temperatura, ruído, luminosidade).
- Social – quaisquer informações que caracterizem a situação dos relacionamentos sociais.

- Computacional – quaisquer informações que descrevem a situação das características computacionais (ex: configuração de dispositivos do usuário).
- Elemento de saúde – classifica o contexto da informação a partir do ponto de vista da saúde (ex: batimento cardíaco, pulso, pressão sanguínea).

O projeto Lariisa reutiliza conceitos do *Geographically Encoded Objects for Really Simple Syndication feeds* (GeoRSS), uma simples marcação com informação de localização para descrição de coordenadas e relações geo-espaciais, assim como o OWL-*Time* que é utilizado para representar conteúdo temporal.

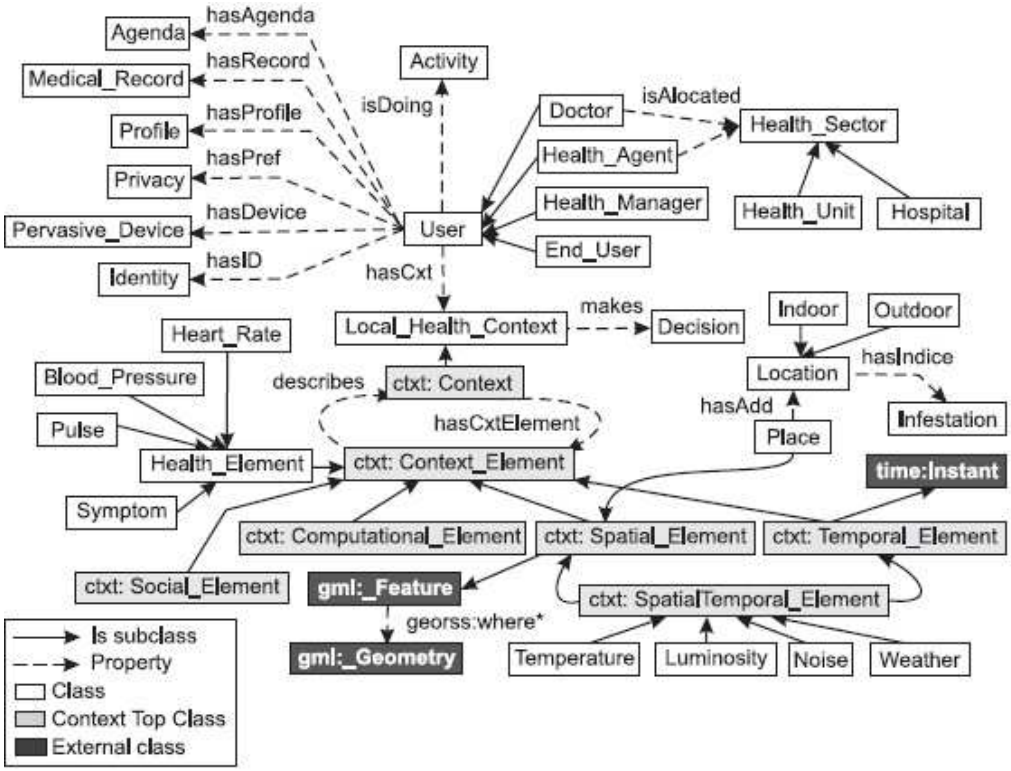

Figura 8 – Modelo Local de Saúde do projeto Lariisa (OLIVEIRA et al., 2010)

O projeto Lariisa define as classes *Global_Health_Context* (ver modelo da Figura 7) e *Local_Health_Context* (ver modelo Figura 8) para representar os conceitos do contexto. Esses conceitos capturam do contexto quaisquer informações para caracterizar uma situação que é relevante para contribuir com decisões em governança de saúde, isto é, que podem ser utilizadas para definir regras de decisão locais e globais. O *framework* utiliza a base do modelo *Event-Condition-Action* (ECA) para descrever regras de decisão local e global que são traduzidas dentro de regras

utilizando a *Semantic Web Rule Language* (SWRL), “uma linguagem com sintaxe de abstração de alto nível para regras da OWL” (HORROCKS, 2002). Um evento representa a identificação de mudanças no contexto. Uma condição descreve um conjunto válido de restrições de contexto, e uma ação descreve uma decisão.

4.2. Configuração de Governança do projeto Lariisa

Esta seção aborda as áreas de governança em que o projeto Lariisa atua, que são: Gerenciamento de Conhecimento, Normatização Sistêmica, Clínico-Epidemiológico, Administrativo e Gerenciamento Compartilhado. Alguns exemplos e trechos de códigos-fonte utilizados nesta seção foram descritos por Oliveira et al. (2010) em seu trabalho sobre o projeto Lariisa.

4.2.1. Governança de Gerenciamento de Conhecimento

É composto por estratégias e práticas utilizadas pelas organizações para identificar, criar e representar experiências de cuidados de saúde. Essas estratégias e práticas são utilizadas para manter e transferir essas experiências, utilizando a pesquisa formal e processos empíricos, além de outras formas de construção de conhecimento e melhorias.

Um exemplo de decisão de governança é criar uma Sala de Emergência (SE) para tratamento clínico de casos graves. O Quadro 2 mostra um exemplo de regra de decisão global utilizando pseudocódigo e o Quadro 3 mostra o código SWRL dessa regra. Este exemplo utiliza informações do número de casos de dengue em uma região em um período de tempo para tomar decisão de criar uma SE na região atingida.

Quadro 2 – Exemplo de regra de decisão global

```
IF ((numberOfDengueRecurrenceCases(region Y, period Z) > X)
THEN {Alert: to create an ER in the region Y}
```

(OLIVEIRA et al., 2010)

Quadro 3 – Exemplo de regra de decisão global utilizando SWRL

```
Global_Health_Context(?ghc) ^ Location(?Y) ^ time:Interval(?Z)
^ hasContextElement(?ghc, ?Y)
^ hasContextElement(?ghc,?Z)
```

```
^ NumberOfRecurrenceCases(?W) ^ hasContextElement(?ghc,?W)
^ swrlb:greaterThan(?W,X)
 MakingDecision(?ghc,"Alert: to create an ER in the region
Y")
```

(OLIVEIRA et al., 2010)

Um exemplo de decisão local de governança é encaminhar um paciente com sintomas de dengue que está em uma área com altos níveis de infestação para uma SE anteriormente criada. O Quadro 4 exibe essa regra de decisão local em pseudocódigo e o Quadro 5 mostra a regra em SWRL. Como resultado da execução dessas regras, espera-se a redução dos índices de mortalidade causados pela dengue.

Quadro 4 – Exemplo de regra de decisão local

```
IF ((the patient has contracted Dengue more than once) AND (he lives in an area of high infestation
indices) AND (he has symptoms A,B,C))
THEN {the patient must consult the ER-SC about this case}
```

(OLIVEIRA et al., 2010)

Quadro 5 - Exemplo de regra de decisão local utilizando SWRL

```
End_User(?patient) ^ Local_Health_Context(?lhc)^Location(?region)
^Infestation(?deng) ^ Symptom(?A) ^ Symptom(?B) ^ Symptom(?C)
^ Medical_Record(?dengue)
^ hasContext (?patient, ?lhc)
^ hasRecord(?patient,?dengue)
^ swrlb:greaterThan(?dengue,1)
^hasContextElement(?patient, ?region)
^hasIndice(?region,?infestation)
^hasContextElement(?patient,?A) ^hasContextElement(?patient,?B)
^hasContextElement(?patient,?C)
 MakingDecision(?lhc,"Alert: the patient must consult the
ER-SC about this case")
```

(OLIVEIRA et al., 2010)

4.2.2. Governança de Normatização Sistêmica

Refere-se à participação de agente públicos e gestores de saúde para a utilização e elaboração de leis, a fim de gerar padrões de consistência, concretude e certeza dos sistemas de saúde.

Um exemplo de decisão de governança em normatização sistêmica é avaliar o valor e as sanções previstas em uma determinada lei. Neste exemplo, quando um fiscal visita um depósito de resíduos de um estabelecimento e identifica

irregularidades, a partir de seu dispositivo móvel, ele é capaz de acessar o sistema e atualizá-lo com essa informação. Ao aplicar regras de decisão local o sistema está apto a identificar este evento e checar se é um caso reincidente. Se este for o caso, o inspetor vai receber um alerta para a aplicação da multa e fechar o estabelecimento. Essa regra de decisão é descrita no Quadro 6, onde um fiscal aplica sanções previstas em lei. Como resultado de aplicações de regras como essa, espera-se a melhoria na fiscalização e, indiretamente, mudanças das características observadas nos depósitos de resíduos.

Quadro 6 – Exemplo de regra de decisão local em Normatização Sistêmica

IF ((the waste deposit of an establishment did not obey the law) AND (it is a recidivist)) THE< {Alert: to apply the fine and close the establishment}

(OLIVEIRA et al., 2010)

4.2.3. Governança Clínico-Epidemiológica

Assegura que os processos de conhecimento de doenças, a partir do conceito que saúde é determinada por fatores biológicos, sociais, econômicos, genéticos, de estilo de vida, influenciando os serviços dos sistemas de cuidados de saúde.

Um exemplo de regra de decisão nessa configuração de governança é a oferta de serviços de hidratação intravenosa em unidades de saúde, baseada em indicadores com informações de contexto, sobre as doenças registradas que requerem hidratação intravenosa em seu tratamento. Como resultado espera-se descentralizar os serviços de hidratação intravenosa. O Quadro 7 e o Quadro 8 mostram as regras em pseudocódigo e SWRL, respectivamente.

Quadro 7 – Exemplo de regra de decisão global (Clínico-Epidemiológico)

IF ((there are cases of disease re-infection in the districts) AND (there are indicators of epidemic disease)) THEN {Alert: to create in the health unit a new intravenous hydration procedure and classify that district in red alert}

(OLIVEIRA et al., 2010)

Quadro 8 – Exemplo de regra de decisão global (Clínico-Epidemiológica) utilizando SWRL

Global_Health_Context(?ghc) ^ Location(?district) ^ time:Interval(?inter) ^ Epidemic(?dengue) ^ hasContextElement(?ghc, ?district)

```
^ hasContextElement(?ghc,?inter)
^ NumberOfRecurrenceCases(?cases) ^
hasContextElement(?ghc,?cases) ^ swrlb:greaterThan(?cases,0)
^ hasContextElement(?ghc,?dengue)
 MakingDecision(?ghc," to create in the health unit a new
intravenous hydration procedure")
Alert_Level(?red) ^ hasContextElement(?ghc,?red)
```

(OLIVEIRA et al., 2010)

4.2.4. Governança Administrativa

Refere-se ao ato de gerir profissionais para a realização de um determinado objetivo e a responsabilidade de manter e supervisionar as entidades relacionadas ao objetivo.

Um exemplo de decisão de governança na área administrativa é a alocação de profissionais para treinamento. Para essa ação é necessária a verificação da quantidade de profissionais treinados em determinada área em unidades de saúde. Como resultado de decisões como essa espera-se otimizar o gerenciamento de treinamento de pessoal. O Quadro 9 e o Quadro 10 mostram a regra em pseudocódigo e em SWRL, respectivamente, deste exemplo.

Quadro 9 – Exemplo de regra de decisão em administração

```
IF (the global quantity X of trained professionals) < (amount of trained professional in the hospitals
and health units)
THEN {Alert: request M new employees and train 4
professionals in X, Y, Z skills}
```

(OLIVEIRA et al., 2010)

Quadro 10 – Exemplo de regra de decisão em administração utilizando SWRL

```
Local_Health_Context(?lhc) ^ User(?professional)
^ hasContext(?professional,?lhc)
^ (< X Health_Unit)(?professional)
^ (< X Hospital)(?professional)
 MakingDecision(?ghc,"Alert: request M new employees
and train 4 professionals in X, Y, Z skills")
```

(OLIVEIRA et al., 2010)

4.2.5. Governança de Gerenciamento Compartilhado

Refere-se à capacidade de compartilhamento de conhecimento nos sistemas de saúde, proporcionando visões de gestão global dos processos internos, competências do governo, experiências da sociedade e suas instituições

representativas, mantendo uma relação harmônica com outros estados federais e entidades internacionais.

Um exemplo de decisão nessa configuração de governança é a mobilização da sociedade e de organizações para criar um comitê especial. Nessa situação um fiscal visita uma região com surto de dengue, se ele encontrar um edifício em construção com mais de X meses, ele vai atualizar essa informação no sistema, utilizando seu dispositivo móvel. Este exemplo está ilustrado no Quadro 11. O sistema, aplicando regras de decisão local, vai incluir este edifício na lista vermelha para acompanhamento de uma comissão especial. Como resultado espera-se a diminuição de taxas de contaminação da dengue.

Quadro 11 – Exemplo de regra de decisão em Gerenciamento Compartilhado

IF ((the building is in construction for more than X months) AND (it is located in an area where the disease infestation > 50%)) THEN {this building is being included in the “red list” for monitoring by the Special Committee}

(OLIVEIRA et al., 2010)

4.3. Knowledge to Action Model

O *framework* Lariisa provê facilidades sensíveis ao contexto para cada conjunto de usuários envolvidos (por exemplo: usuários finais, gestores de saúde e agentes de saúde). Por um lado, o *framework* deve considerar os requisitos do processo de tomada de decisão em governança a fim de se tornar um sistema mais efetivo e integrado de cuidados de saúde. Por outro lado, geralmente há uma lacuna entre a criação do conhecimento, a detecção de contexto e processos de aplicação do conhecimento. Foram propostos dois ciclos para cada etapa: i) ciclo de criação de conhecimento; ii) ciclo de ação. O modelo KTA foi projetado para ajudar profissionais, pesquisadores, políticos, pacientes e o público em geral a entender como o conhecimento e prática nos sistemas de saúde interagem e se influenciam mutuamente.

O conhecimento chega de várias fontes e inclui tanto a experiência pessoal e de pesquisa. A criação de conhecimento é um processo de adaptação, onde questões de pesquisa são projetadas para resolver os problemas identificados pelos usuários, enquanto os resultados de pesquisa e sua divulgação são adaptados para atender as necessidades de públicos específicos. No ciclo de ação, é utilizada a

teoria de ação planejada (modelos) para descrever o que acontece nos ciclos. Esses modelos são utilizados para prever a probabilidade de mudanças.

O projeto Lariisa oferece mecanismos sensíveis ao contexto para reduzir a diferença no processo de transferência de conhecimento para o ciclo de ação. De forma semelhante à criação de conhecimento e processo de ação do modelo de Graham, há uma lacuna entre a detecção de contexto da saúde para adaptar o conhecimento para a situação local/global e como este contexto afeta os aplicativos relacionados com a saúde (ação). No entanto, o projeto Lariisa é capaz de reduzir essa lacuna, oferecendo mecanismos sensíveis ao contexto de adaptação para cada etapa do ciclo de ação.

O ciclo de criação de conhecimento do modelo KTA também pode adaptar seus processos tendo em conta informações de contexto global de saúde. Este ciclo é mais complexo que o ciclo de ação. Ele tem características específicas e dinâmicas, o que pode ser assistido por sistemas inteligentes, independentemente do ciclo de ação.

4.4. Lariisa Framework Core

A essência do framework Lariisa é descrita nesta seção. A Figura 9 mostra o Lariisa *Framework Core* e seus componentes: *Context Provider* (CP), *Context Aggregator* (CA), *Context Reasoner* (CR), QoC *Evaluator* (QoCE), *Service Adapter* (SA), *Context-aware service* (CAS), CAS *Container* (CasC) e *Query Adapter* (QA), que são explicados a seguir.

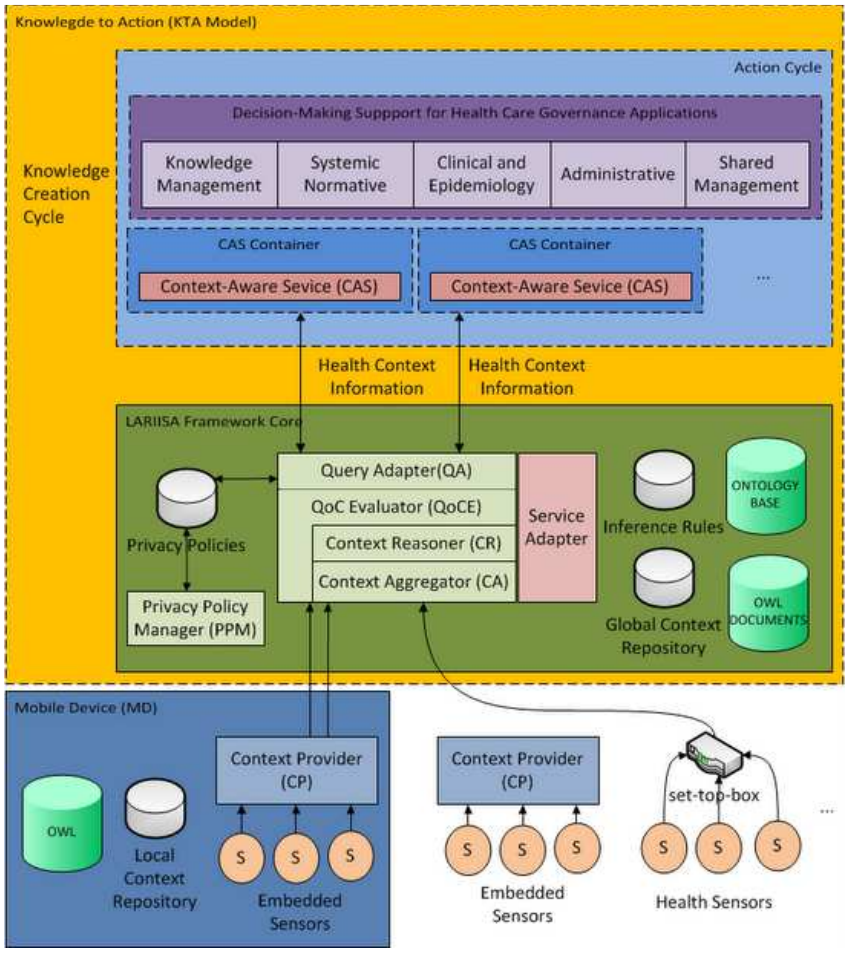

Figura 9 – Lariisa *Framework Core*

Context Provider

O CP, ou Provedor de Contexto, é o responsável pela coleta de dados brutos com contexto de um ambiente. Contexto de sensores móveis (ex: dispositivo móvel de agente de saúde) e residências de famílias (usando *set-top-box*), podem ser enviados ao CA (ver a seguir). Esses sensores podem estar fisicamente conectados ao *set-top-box* da TV Digital ou podem estabelecer uma conexão externa (ex: através de WiMAX, GSM/GPRS/3G ou similares) com o sistema para transmitir dados recuperados do contexto. O projeto Lariisa utiliza o Diga Saúde para recuperar sinais vitais do usuário final, através da integração de sensores com o *set-top-box*, como a temperatura do corpo, batimento cardíaco, pulso, taxa respiratória e pressão sanguínea.

Context Aggregator

O CA, ou Agregador de Contexto, é responsável por receber informação com contexto de saúde oriundas de vários Provedores de Contexto e por executar operações de agregação de contexto. O principal objetivo do CA é obter um contexto de alto nível representado pela ontologia de Contexto Local de Saúde.

Context Reasoner

O CR, ou Raciocinador de Contexto, executa processos de inferência e derivação nas informações de contexto de saúde descritas pelas instâncias do Contexto Local de Saúde com o objetivo de obter uma semântica de alto nível de informação contextual e gerar informações para o Contexto Global de Saúde. Por exemplo, o Raciocinador de Contexto é capaz de inferir uma situação de epidemia (ou seja, informação de Contexto Global de Saúde) a partir do Contexto Local de Saúde obtido a partir das residências das famílias e por agentes de saúde. A linguagem SRWL é utilizada para descrever as regras de inferência e derivação.

QoC Evaluator

O QoCE, ou Avaliador de Qualidade de Contexto, avalia informações de qualidade do contexto e gera indicadores de qualidade a cada conceito de contexto. É utilizado para melhorar as decisões em governança de saúde (ex: precisão e atualização da localização espacial).

Service Adapter

O SA, ou Adaptador de Serviço, é a camada principal do projeto Lariisa. Ela é responsável pela identificação de informação relevante de contexto de saúde para um dos três seguintes ciclos: i) processo de criação de conhecimento; ii) processo de tomada de decisão em governança de saúde; e iii) ações de cuidados de saúde. Além disso, lida com as seguintes funções: (i) adaptação sensível ao contexto de regras de decisão local de saúde, tendo em vista as decisões de governança (adaptação *top-down*); (ii) adaptação sensível ao contexto de regras de decisão local de saúde, tendo em vista do contexto de saúde local; (iii) oferecer indicadores de saúde sensíveis ao contexto que descrever o contexto global de saúde para as entidades de criação de conhecimento e aplicações de tomada de decisão em

governança (adaptação *bottom-up*). O SA também é responsável por executar automaticamente regras de decisão local e global, utilizando Pellet[7].

Context-aware service

O CAS, ou Serviço Sensível ao Contexto, utiliza informações dos contextos local e global de saúde obtidas do SA para adaptar suas funcionalidades. Serviços sensíveis ao contexto irão compor as aplicações de tomada de decisão em governança de saúde projetadas de acordo com o Ciclo de Ação do modelo KTA.

CAS Container

O CasC, ou serviço de contêiner sensível ao contexto, representa um grupo de CAS. Uma aplicação de tomada de decisão em governança é composta por um ou mais CasC.

Query Adapter

O QA, ou Adaptador de Consulta, manipula consultas contextuais dos Serviços Sensíveis ao Contexto e de entidades do ciclo de conhecimento, extraindo informações contextuais relevantes do *Context Global health Repository* (ver Figura 9). As políticas de privacidade protegem informações contextuais e são armazenadas e executadas pela *Privacy Policy Management* (ver Figura 9).

4.5. Diga Saúde

Diga Saúde (SANTOS, 2011) é um sistema de baixo custo que disponibiliza serviços de *home care*, tendo como principal *interface* com o usuário a TV Digital Interativa. O Diga Saúde aproveita-se da popularização prevista para TV Digital brasileira que possibilitará a existência de um mecanismo de processamento computacional (*set-top-box*) em todas as residências no país.

O Diga Saúde teve significativa participação no projeto GINGA *Code Development Network* (CDN), da RNP, em 2008, e concebido para explorar a pesquisa e o desenvolvimento de ferramentas de apoio à criação, gerenciamento e operação de uma rede de desenvolvedores de código para o *middleware* Ginga.

[7] Pellet é um raciocinador para Java que provê serviços para ontologias em OWL.

O Diga-Saúde também serviu de lastro conceitual na concepção do projeto Lariisa, um projeto orientado a contexto para tomada de decisão em saúde que está sendo implementado no município de Tauá, no Ceará, pelos grupos de pesquisa Rede Interdisciplinar de Pesquisa e Avaliação em Sistemas de Saúde (Ripass) e Laboratório de Redes (LAR) da UFC e IFCE, respectivamente.

As características de interatividade do Ginga agregam, naturalmente, mais funcionalidades ao Diga Saúde, além do envio de mensagens e avisos, de exibição de dicas de saúde e de monitoramento de sinais vitais já presentes em sistemas de *home care* e do acompanhamento e administração do uso de medicamento do paciente, inexistente em soluções dessa natureza. As funcionalidades do Diga Saúde são ilustradas a seguir na Figura 10.

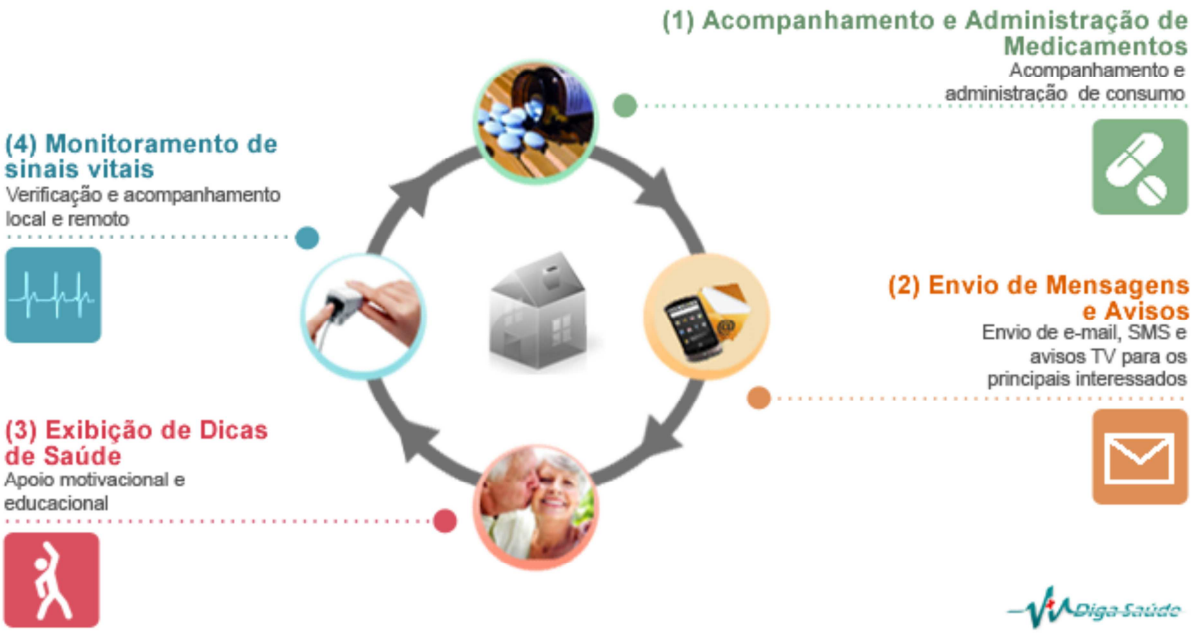

Figura 10 – Funcionalidades do Diga Saúde (SANTOS, 2011)

O Diga Saúde envolve sistemas e serviços *web*, servidores, aparelhos eletrônicos (como sensores, celulares, entre outros) e a TV Digital. Uma visão geral do cenário de organização é ilustrado na Figura 11.

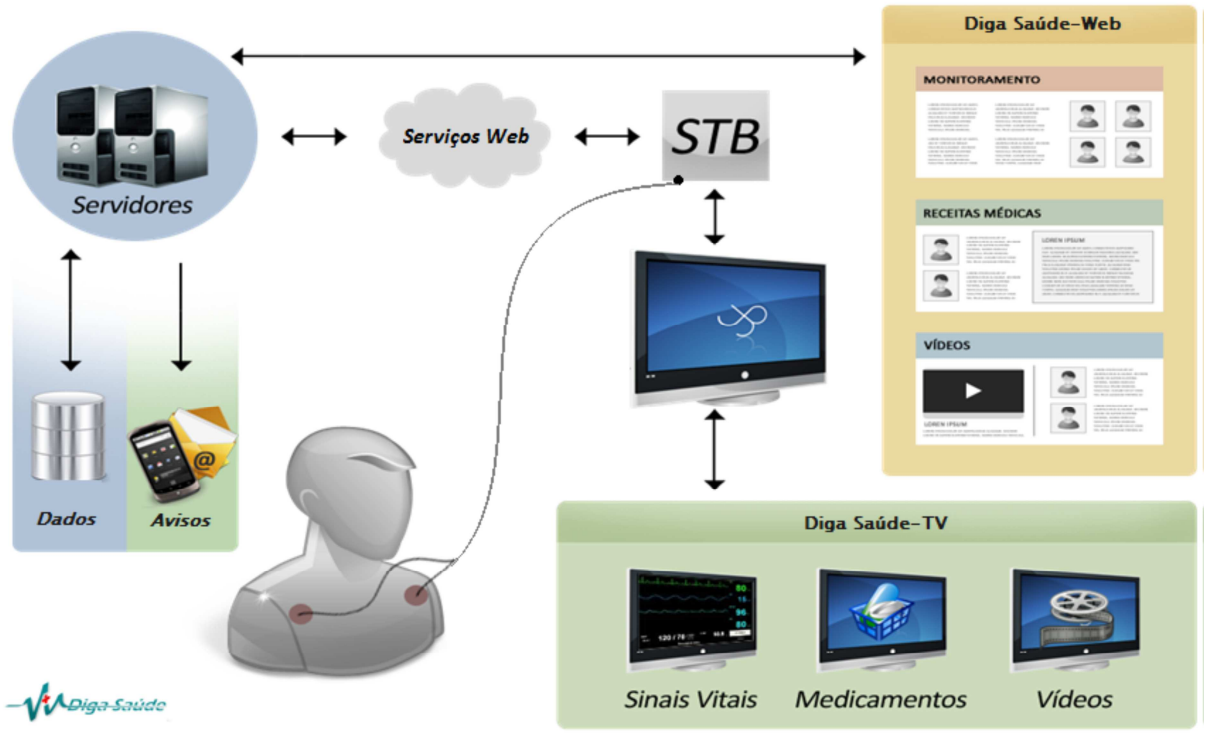

Figura 11 – Visão Geral do Diga Saúde (SANTOS, 2011)

Foi feita uma prova de conceito para verificar que o Diga Saúde atende todas as funcionalidades de um sistema de *home care* baseado em TV Digital Interativa de baixo custo, com o diferencial de utilizar o Ginga e auxiliar o acompanhamento e administração do consumo de medicamentos no dia-a-dia do paciente.

Vale dizer que o Diga Saúde é, até o momento, o único sistema de *home care* baseado no Ginga. Isso se torna mais relevante na medida em que o *middleware* brasileiro é uma recomendação mundial na área de TV Digital, sendo adotado em mais 10 países latino-americanos e em alguns países da África.

4.6. Lisa: *Lariisa Integration System Architecture*

A Lisa (*Lariisa Integration System Architecture*) é uma arquitetura de integração de provedores de contexto heterogêneos ao projeto Lariisa de forma a permitir sua expansibilidade, flexibilidade e facilidade de retirada/inclusão de provedores de contextos mesmo que não tenham sido especificamente preparados para interfaceamento ao modelo. A arquitetura Lisa foi proposta por Frota (2011).

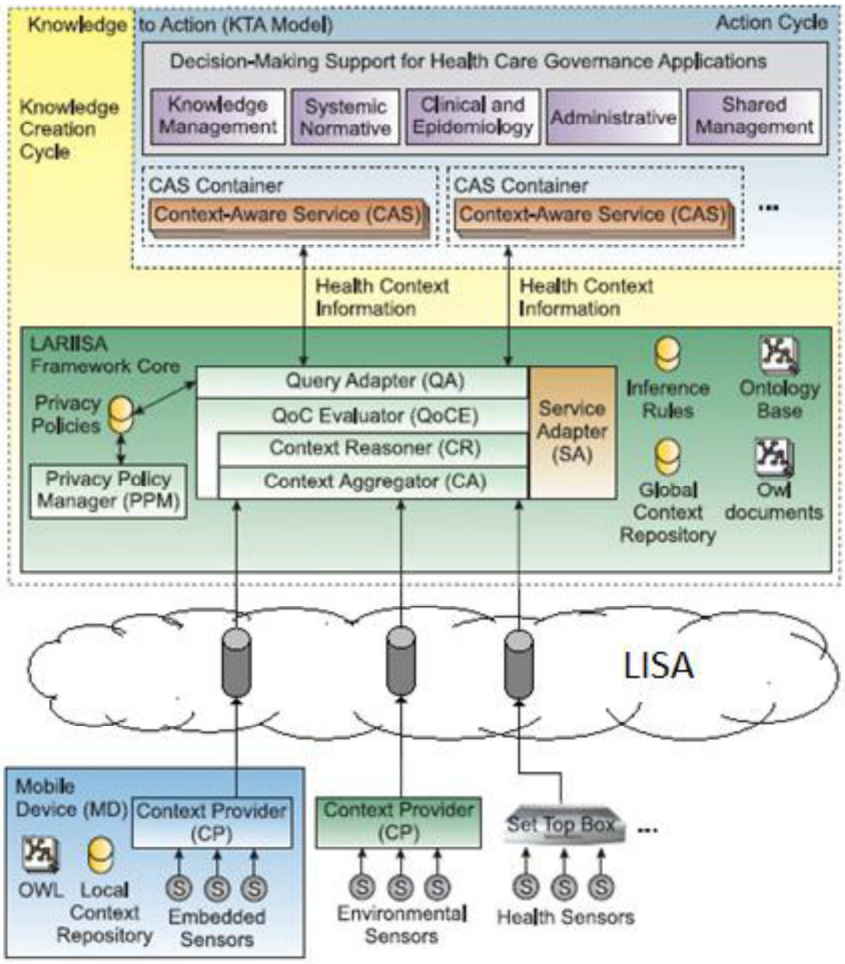

Figura 12 – Diagrama de Blocos do Projeto Lariisa acrescido da arquitetura Lisa (FROTA, 2011)

Como é possível ver na Figura 12, onde a arquitetura Lisa é representada por uma nuvem, que abstrai detalhes a cerca dos provedores de contexto. Ele realiza este serviço através de um *Enterprise Service Bus* (ESB). No ESB são implementadas as formas de comunicação para cada tecnologia de dispositivos provedores de contexto.

Com a arquitetura Lisa é possível integrar quaisquer dispositivos ao Lariisa, pois a arquitetura proposta por Frota (2011) permite a adaptação da informação contextual mediante tradução de mensagem e adaptação de canal (ver Figura 13).

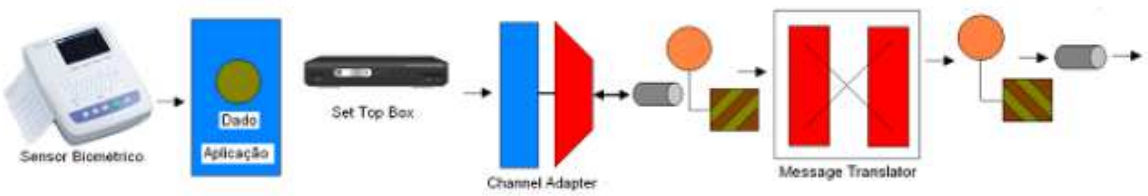

Figura 13 – Integração de aplicativos na arquitetura Lisa/Lariisa (FROTA, 2011)

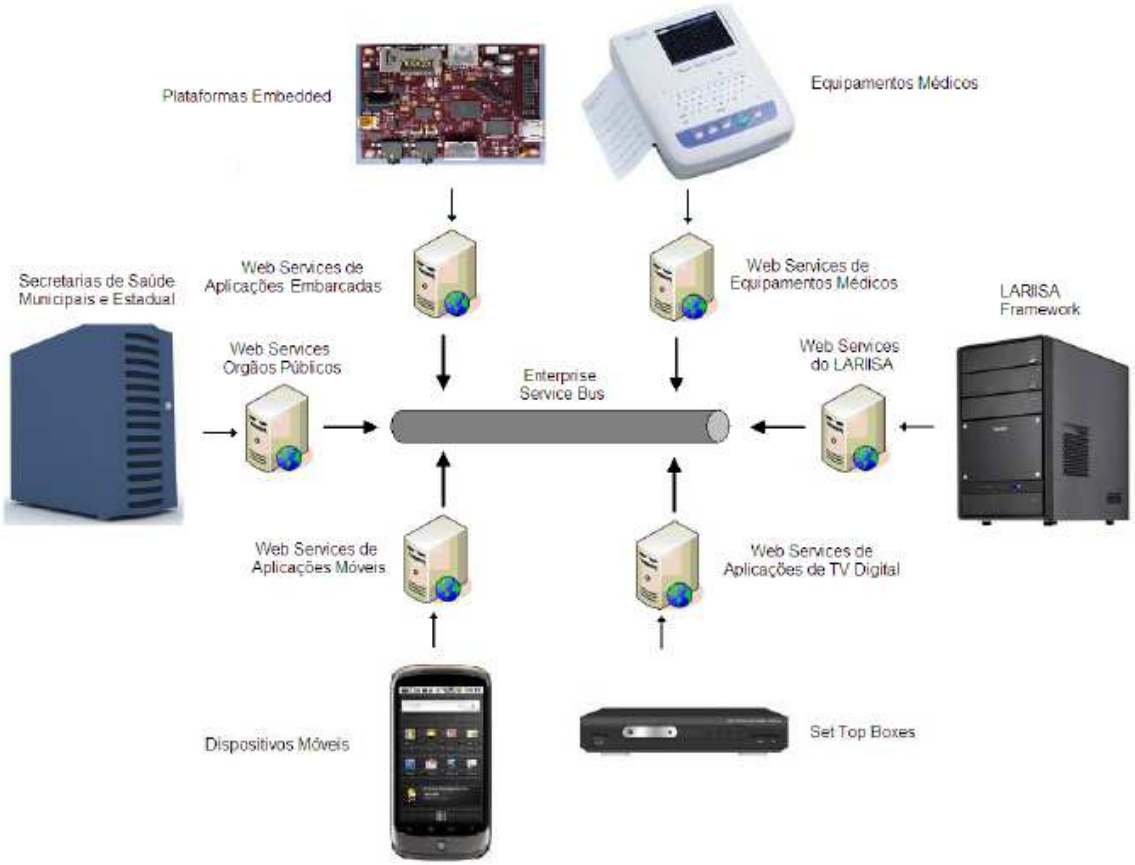

Figura 14 – Arquitetura da Lisa (FROTA, 2011)

A Figura 14 ilustra o possível cenário onde diversos dispositivos provedores de contexto fornecem informação contextual captada pelo ESB da arquitetura Lisa e integrados ao Larissa.

4.7. Sisa

Sisa (ANTUNES, 2011) é um sistema sensível ao contexto de gestão em saúde integrado ao domínio epidemiológico do projeto Lariisa. Propõe um serviço inteligente, baseado em regras de decisão, de suporte à tomada de decisão em gestão de saúde pública. Para obter os dados da família e do contexto externo, o SISA utiliza *WebServices*, o Ginga, *middleware* da TV Digital Brasileira e a tecnologia de computação sensível ao contexto, inexistentes em outras soluções dessa natureza

Um protótipo foi implementado utilizando as notificações de dengue no estado do Ceará. Trata-se, assim, de um sistema capaz de perceber os fatores externos ao processo e adaptar-se, em tempo real, ao novo contexto para auxiliar a tomada de decisão dos usuários finais aos gestores de saúde (Secretário de Saúde e/ou Governador de Estado, por exemplo). Seja alterando a agenda diária de visita dos agentes de saúde, sinalizando novos focos de dengue, ou gerando mapas epidemiológicos, que serão visualizados e analisados para, assim como acontece em outras soluções, auxiliar a tomada de decisões de forma ágil e eficiente. O Sisa tem o diferencial de analisar o contexto presente no dia-a-dia dos cidadãos e profissionais de saúde.

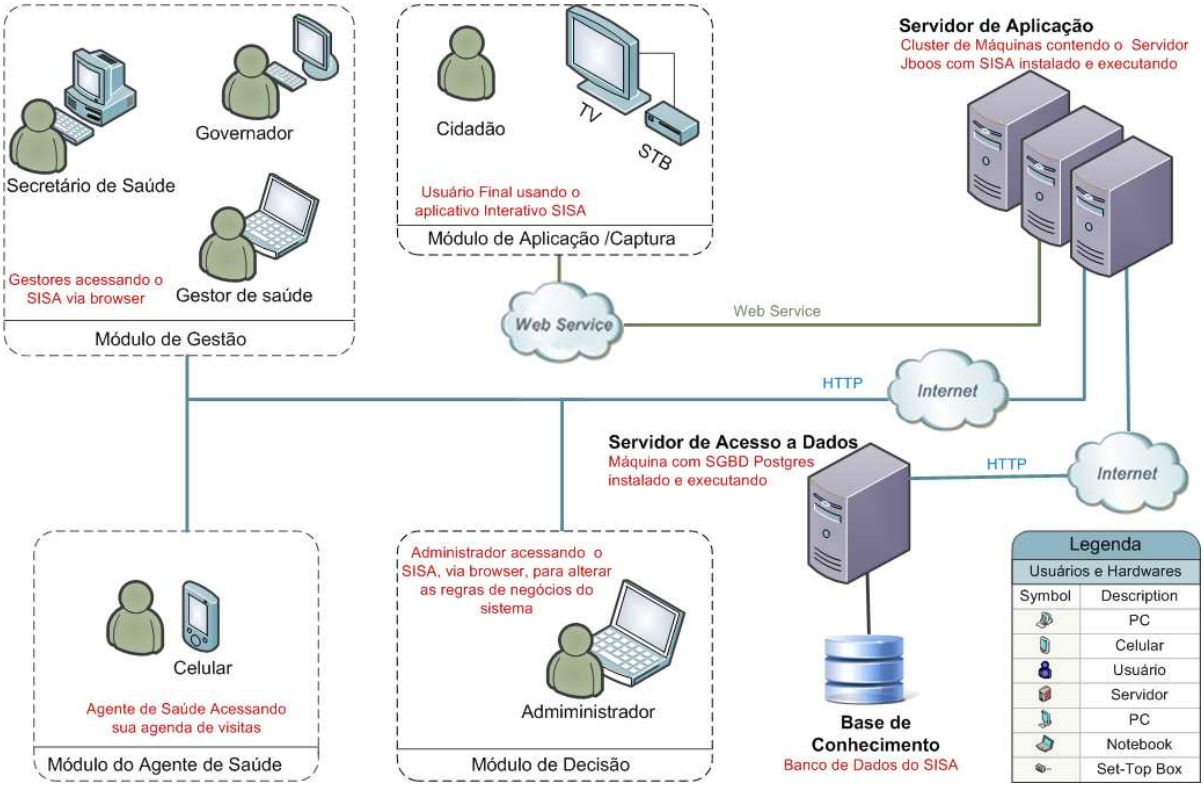

Figura 15 – Visão geral da aplicação Sisa (ANTUNES, 2011)

A Figura 15 ilustra alguns atores do sistema Sisa e seus relacionamentos com os módulos da aplicação. Pacientes utilizam aplicações instaladas nos *set-top-boxes* de suas residências para responder um questionário sobre sintomas da dengue, agentes de saúde reportam casos detectados através de celulares e gestores de saúde analisam informações contextuais e inferências realizadas pela aplicação. Já

a Figura 16 ilustra o Diagrama de Atividades do Sisa com um nível avançado de detalhes, que exibe o fluxo de informações e processos do sistema.

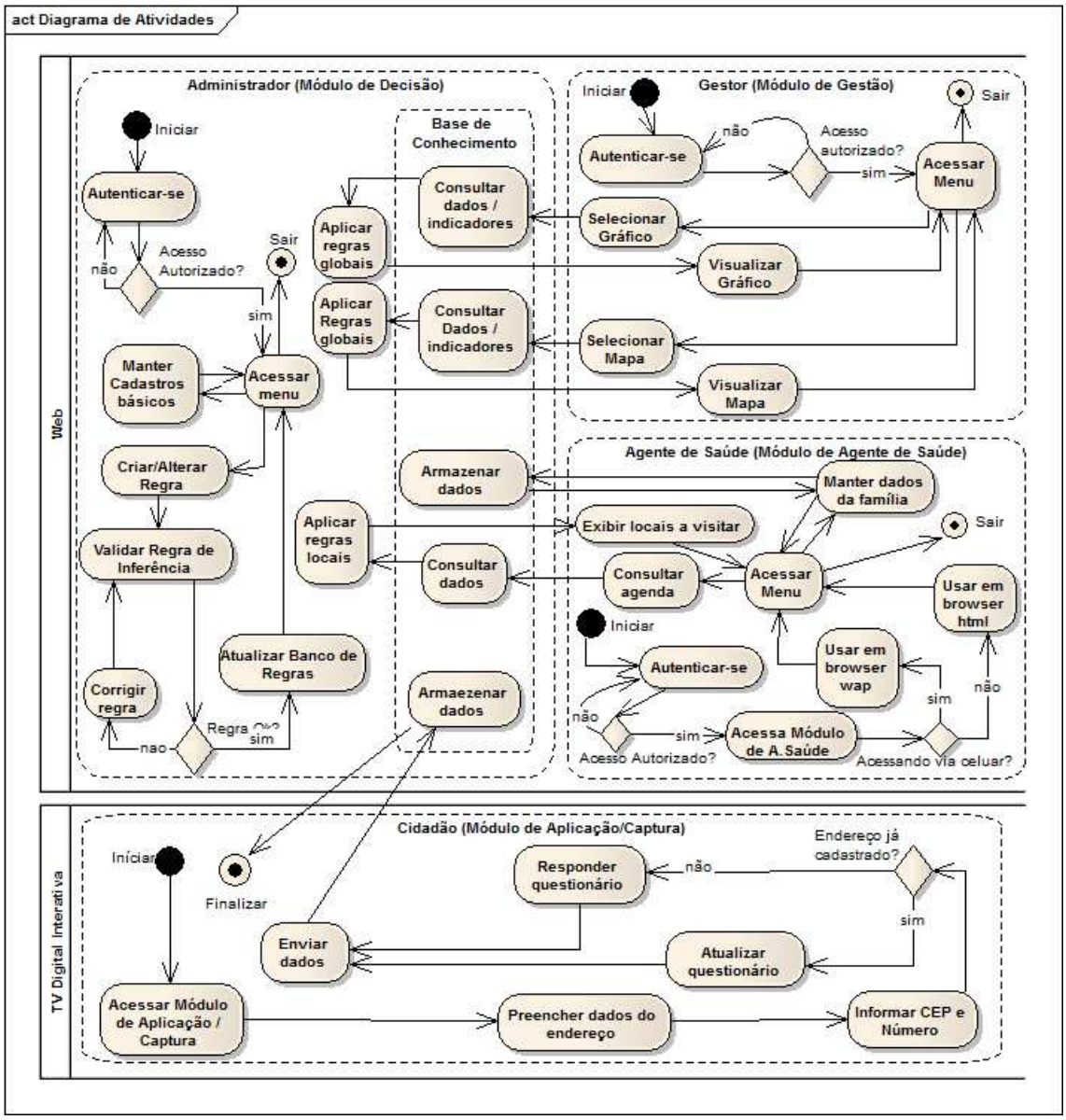

Figura 16 – Diagrama de Atividades do Sisa (ANTUNES, 2011)

O Sisa contribui diretamente com o projeto Lariisa como a primeira prova de conceito, pragmática sob o domínio seu domínio epidemiológico. A concepção do Sisa possibilita sua integração de maneira pratica e eficiente, promovendo a obtenção de resultados positivos, ocasionando expectativas reais de grande relevância e aplicabilidade na área da saúde.

4.8. Caminhos do Conhecimento

Bastos (2012) analisou o processo de adaptação do conhecimento no projeto Lariisa. A adaptação do conhecimento objetiva diminuir o abismo temporal e conceitual que há entre a produção do conhecimento e a prática do usuário final. Ela é essencial para se obter melhores resultados em cuidados de saúde, além de contribuir para a diminuição de custos na utilização do conhecimento, disponibilizando o conhecimento certo, para a pessoa certa, em tempo certo.

O estudo tomou como base o modelo KTA, desenvolvido por pesquisadores da Universidade de Toronto e do *Canadian Institute of Health Research* e que tem servido de referência ao sistema de saúde canadense. Como prova de conceito do estudo analítico realizado foi utilizado, como cenário de aplicação, o projeto Lariisa, que utiliza na especificação de sua arquitetura o modelo KTA.

Foram propostos os Caminhos do Conhecimento para a Tomada de Decisão na instância de Gerenciamento do Conhecimento do projeto Lariisa, ilustrados na Figura 17. Além dos caminhos do projeto Lariisa, foram definidos também os caminhos do profissional de saúde e dos gestores de saúde.

Figura 17 – Os caminhos do conhecimento dentro do projeto Lariisa (BASTOS, 2012)

Bastos (2012) também propôs alterações no projeto Lariisa no tocante à agregação de contexto. Além de agregar contexto dos provedores, o projeto Lariisa, com a alteração proposta, também irá adaptar a informação contextual.

O trabalho deu importante contribuição para a plataforma Lariisa por ser o primeiro a abordar a instância de configuração de governança de Gestão do Conhecimento, podendo servir de base para trabalhos futuros.

Os mapas conceituais permitiram ver os caminhos que o conhecimento percorre, em um sistema integrado de saúde, entre os tomadores de decisão e os diversos domínios de inteligência: Administrativo, Gestão do Conhecimento, Normativo, Clínico Epidemiológico e Gestão Compartilhada. Os mecanismos de inteligência se dão desde a entrada do conhecimento no sistema até a sua aplicação final aos tomadores de decisão.

4.9. Considerações Finais do Capítulo

Este capítulo abordou o projeto Lariisa, destacando o propósito do sistema e suas principais funcionalidades. Foram discorridos os modelos de contexto local e global de saúde que auxiliam na construção de aplicações inteligentes na área de governança da saúde.

Também foi visto a configuração de governança nas diversas instâncias do sistema: Governança de Gerenciamento de conhecimento, Governança de Normatização Sistêmica, Governança Clínico-Epidemiológica, Governança Administrativa e Governança de Gerenciamento Compartilhado.

O modelo KTA define estratégias de transferir o conhecimento para o domínio da ação, fazendo com que o sistema atue ativamente baseado em sua base de conhecimento.

Foi exposto também o funcionamento do Lariisa *Framework Core*, que é a essência do projeto Lariisa. Foi mostrado como funciona os componentes CP, CA, CR, QoCE, SA, CAS, CasC e QA.

Em seguida foram discorridos os trabalhos baseados no projeto Lariisa: Diga Saúde, arquitetura Lisa, Sisa e Caminhos do Conhecimento. O projeto Diga Saúde foi um projeto de *home-care* baseado no SBTVD que serviu de base para o projeto Lariisa. Já a Lisa é uma arquitetura para o projeto Lariisa de integração de sistemas.

O projeto Sisa é uma aplicação construída com base na configuração de Governança Clínico-Epidemiológica do projeto Lariisa que trata o domínio da dengue. Por fim, o projeto Caminhos do Conhecimento define estratégias de configuração da Governança de Gerenciamento de Conhecimento através da utilização de mapas conceituais e de fluxos de informação no projeto Lariisa.

5. PAOLA: UMA PLATAFORMA PARA O DESENVOLVIMENTO DE APLICAÇÕES BASEADAS EM ONTOLOGIAS PARA O PROJETO LARIISA

Este capítulo aborda a proposta desta dissertação, onde é especificada a Paola: Uma Plataforma para o Desenvolvimento de Aplicações baseada em Ontologias para o projeto Lariisa e são explanados seus serviços.

A plataforma envolve os seguintes temas: contexto, ontologias, informática na saúde, governança em saúde, desenvolvimento de sistemas e o projeto Lariisa. Um pouco de cada uma dessas áreas formam o sistema proposto.

Esta plataforma apoia o desenvolvimento de aplicações sensíveis ao contexto, ou seja, que se adaptam automaticamente às mudanças no ambiente e às necessidades correntes dos usuários, sem que haja intervenção direta de pessoas. O contexto da aplicação é modelado com ontologias, que fornecem abstração de estruturas de dados e de implementação, promovendo a interoperabilidade entre sistemas.

Nos moldes do projeto Lariisa, esta plataforma utiliza provedores de contexto, cria bases de conhecimento e define regras e inferências, através de uma interface de desenvolvimento que auxilia o desenvolvedor na construção do sua aplicação sensível ao contexto.

São estudadas formas de como a plataforma Paola pode auxiliar o desenvolvedor a criar aplicações com mais facilidade e com menores custos e tempo dispendidos. Como resultados desses estudos são definidos os requisitos funcionais e não funcionais do sistema, conforme a seguir.

Requisitos Funcionais:

- o sistema deve ser capaz de representar conhecimento com ontologias;
- o sistema deve ser capaz de manipular dados contextuais;
- o sistema deve ser capaz de criar e manipular regras;
- o sistema deve ser capaz de criar e manipular ações;

- o sistema deve ser capaz de selecionar provedores de contexto e suas informações utilizando a arquitetura Lisa;
- o sistema de ser capaz de realizar simulação de aplicação sensível ao contexto;
- o sistema deve ser capaz gerar artefato executável.

Requisitos Não Funcionais:

- o sistema deve ser materializado em uma aplicação *web*;
- o sistema deve disponibilizar documentação de fácil aprendizado;
- o sistema deve permitir que o desenvolvedor crie aplicações sem necessariamente saber detalhes do projeto Lariisa e a arquitetura Lisa;
- o sistema deve permitir que o desenvolvedor salve e exporte um projeto.

A partir do levantamento de requisitos são definidos os módulos da plataforma Paola, ilustrados na Figura 18, que são:

- **Gerenciamento de Bases de Conhecimento**: contém mecanismos para manipular bases de conhecimento.
- **Gerenciamento de Provedores de Contexto**: contém mecanismos para manipular provedores de contexto e suas informações.
- **Gerenciamento de Regras e Ações**: contém mecanismos para manipular regras e ações com base nos dados representados pelas bases de conhecimento.
- **Gerenciamento de Informação**: contém mecanismos para consulta e alteração de dados armazenados no sistema.
- **Gerador de artefato executável e simulação**: une todas as características definidas nos outros módulos e gera um artefato executável para ser acoplado aos sistemas sensíveis ao contexto. Também é possível simular a execução dessa aplicação.
- **Tutoriais**: contém documentação do sistema.

Figura 18 – Diagrama de blocos dos módulos da plataforma Paola

A seção a seguir descreve detalhadamente as funcionalidades de cada módulo.

5.1. Funcionalidades do Sistema

Esta seção detalha as funcionalidades que a plataforma Paola oferece para seus usuários, onde é mostrada uma visão geral dos serviços oferecidos aos desenvolvedores do projeto Lariisa. Neste capítulo não são focados os aspectos de implementação, este são vistos no capítulo 6.

Como se percebe na Figura 19, o desenvolvedor do Lariisa interage diretamente com a plataforma Paola, que encapsula detalhes do projeto Lariisa, da arquitetura Lisa e de suas tecnologias específicas. A plataforma Paola fornece uma interface para o desenvolvimento de aplicações onde o desenvolvedor foca seu esforço principalmente nos serviços que deseja construir, sem se preocupar diretamente com os detalhes de configuração do Lariisa. Com isso, a construção de aplicações sensíveis ao contexto para o projeto Lariisa deve ter seu nível de complexidade reduzido e possuir uma maior agilidade.

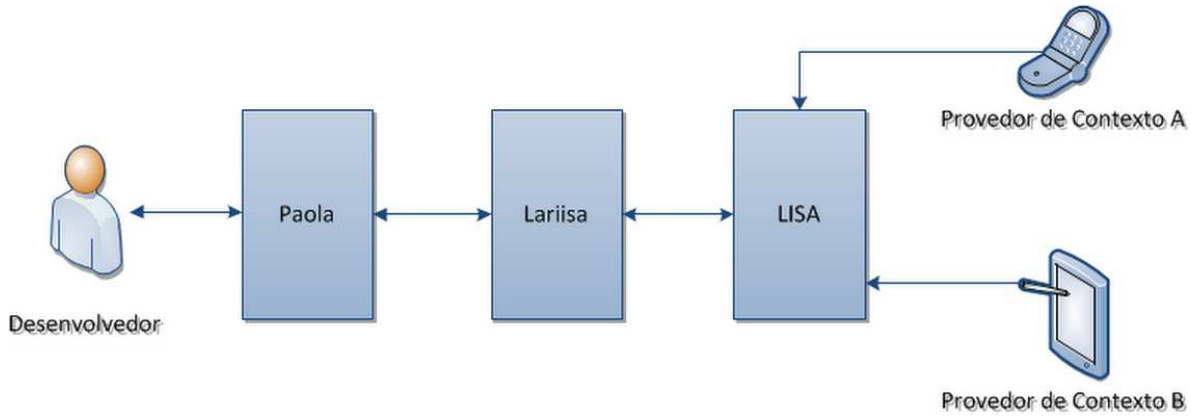

Figura 19 – Abstração de detalhes do projeto Lariisa através da plataforma Paola

Para descrever as funcionalidades da plataforma Paola do ponto de vista do desenvolvedor são utilizados casos de uso. Caso de uso é uma técnica de modelagem de sistemas usada para descrever funcionalidades/serviços oferecidos aos atores do sistema (entidades que interagem com o sistema).

Adiante são exibidos diagramas de casos de uso UML e as telas que foram desenvolvidas baseadas nos casos. Os casos de uso expandidos estão no Apêndice A. Juntamente com cada tela são feitas explanações sobre as funcionalidades do sistema com exemplos onde forem necessários.

Os diagramas de caso de uso desta dissertação são desenvolvidos utilizando o *software* Bouml, uma ferramenta livre para criação de diagramas na linguagem UML. Já os protótipos das telas foram desenvolvidos utilizando o *software* Balsamiq *Mockups*, uma ferramenta para a prototipagem de interfaces gráficas.

A Figura 20 ilustra o diagrama de caso de uso das funcionalidades gerais da plataforma Paola. Na figura é possível observar que o desenvolvedor interage diretamente com cada um dos módulos do sistema.

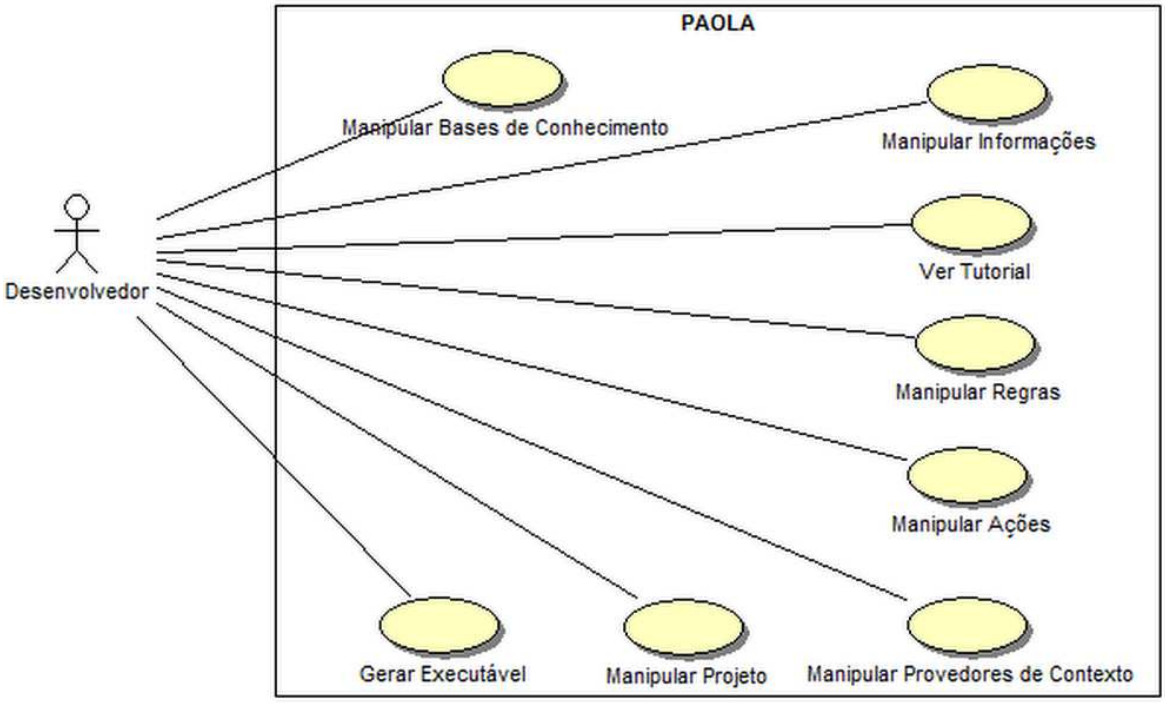

Figura 20 – Diagrama de Caso de Uso das funcionalidades da plataforma Paola

Baseado no caso de uso a tela principal, ilustrada na Figura 21, foi projetada. Na tela é possível visualizar todas as ferramentas disponíveis para criação e edição de bases de conhecimento, de regras/ações, de provedores de contexto, de dados, gerador de executável e simulação. Também existe uma seção de Tutorial e, nesta, há uma documentação que orienta o desenvolvedor que está utilizando a plataforma Paola pela primeira vez. Na tela principal é possível o usuário iniciar o desenvolvimento de um novo projeto para uma aplicação sensível ao contexto do projeto Lariisa e abrir cada funcionalidade da plataforma.

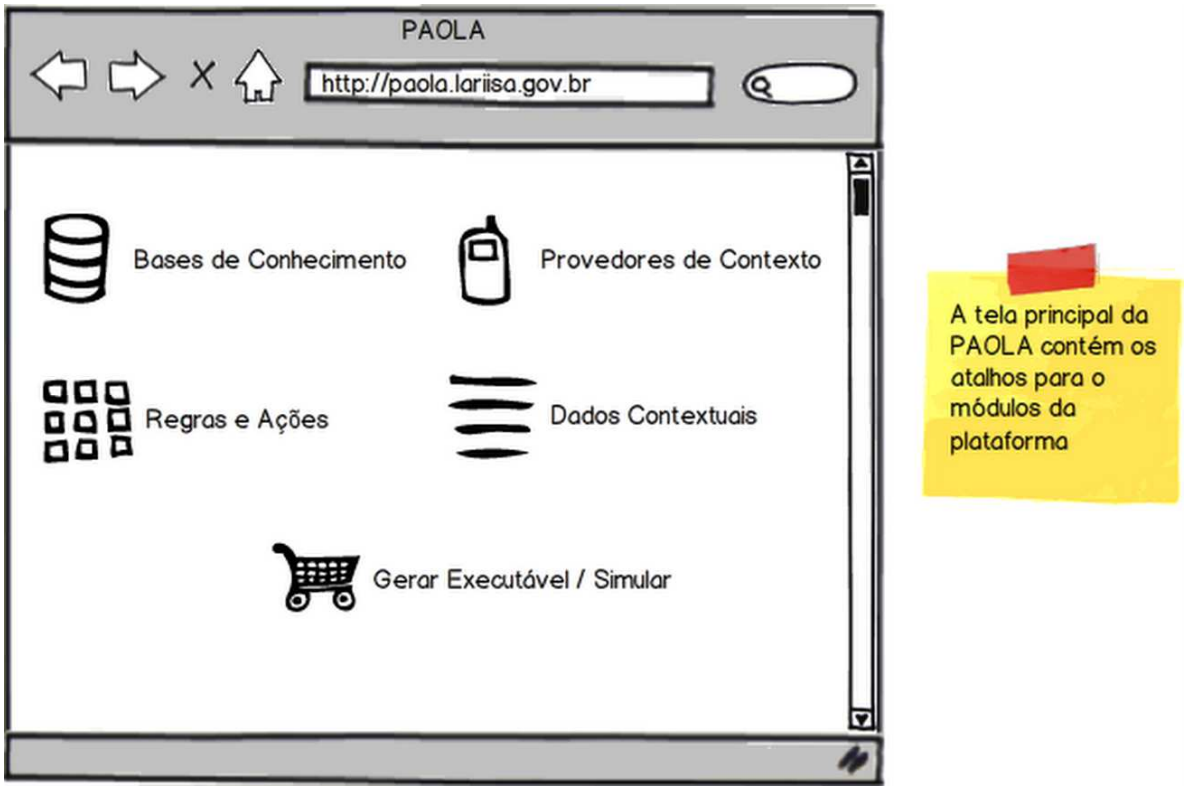

Figura 21 – Tela principal da interface gráfica da plataforma Paola

A tela principal serve como ligação de todos os módulos. A plataforma Paola trabalha com o conceito de projeto, ou seja, todas as funcionalidades e todos os módulos são configurados para o projeto que está sendo construído. É possível criar um novo projeto ou editar um projeto que esteja armazenado em disco. Também é possível salvar o projeto a qualquer momento.

A plataforma Paola também tem a funcionalidade de gerar o executável de um projeto. Através do executável o desenvolvedor pode disponibilizar o artefato gerado para integração com outros sistemas. Sistemas podem se integrar com o artefato gerado através de consultas aos dados, de consultas às bases ontológicas ou no recebimento de notificações quando houver mudança de contexto. Detalhes sobre o artefato executável fruto do desenvolvimento na plataforma Paola são vistos na seção 5.1.5.

A seguir é feito um discorrimento sobre cada módulo da plataforma Paola e são mostradas suas funcionalidades.

5.1.1. Funcionalidades do módulo de Manipulação de Bases de Conhecimento

Bases de conhecimento, nesta dissertação, são utilizadas na modelagem de representação de contexto. Um modelo bem definido é importante para que se tenha fidelidade ao representar uma realidade.

O Módulo de Manipulação de Bases de Conhecimento utiliza ferramentas para representar bases de conhecimento que permitam a leitura de ontologias e sejam capazes de realizar inferências. O conhecimento representado na base de conhecimento é importado para que dados possam ser armazenados, seguindo a semântica de domínio. Neste módulo é definido como acontece a comunicação com a base de conhecimento e a ligação com outros módulos: Gerenciamento de Informação e Gerenciamento de Regras e Ações.

Para a criação e edição de bases de conhecimento a plataforma Paola fornece um módulo que possui algumas funcionalidades para o trabalho com ontologias, listadas a seguir:

- Pesquisar ontologias.
- Importar ontologias.
- Criar ontologias.
- Editar ontologias.
- Remover ontologias.
- Adicionar base de ontologias.
- Remover base de ontologias.

O módulo de edição de bases de conhecimento facilita o reuso de representação do conhecimento. A Figura 22 ilustra o diagrama de caso de uso da edição de bases de conhecimento na plataforma Paola.

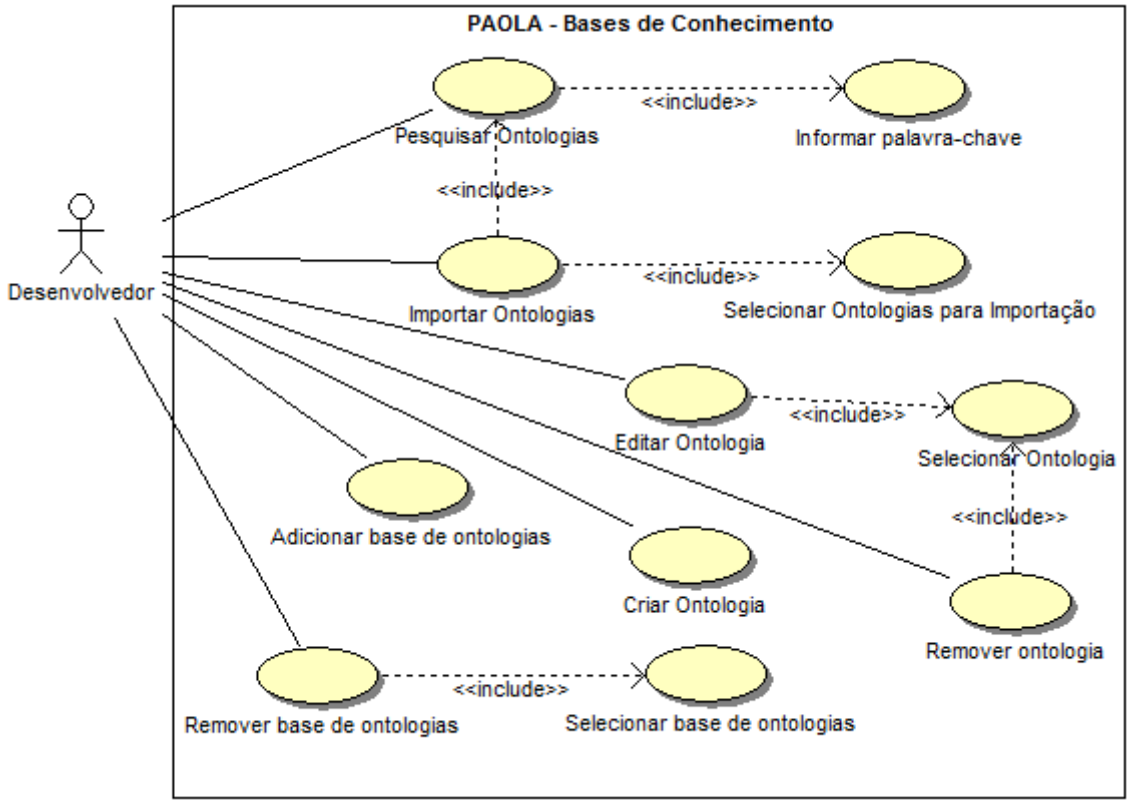

Figura 22 – Diagrama de Caso de Uso de edição de Bases de Conhecimento da plataforma Paola

No módulo é possível encontrar ontologias através de uma ferramenta de busca integrada à plataforma Paola, informando uma palavra-chave. A plataforma Paola possui integração com diversas bases de ontologias disponíveis na internet, que disponibilizam gratuitamente o código-fonte para reutilização. O desenvolvedor tem a opção de escolher uma ou mais ontologias na lista de resultados da busca e efetuar a importação para a sua aplicação.

A ferramenta de busca se integra com bases de ontologias na web que proveem códigos-fonte para reutilização. A plataforma Paola já possui algumas bases integradas e possibilidade do desenvolvedor adicionar e remover novas bases.

O desenvolvedor pode utilizar várias ontologias em sua aplicação e pode editar ou criar novas ontologias. A plataforma Paola oferece um editor simples de OWL para que se possa realizar a edição sem sair da ferramenta. Se o desenvolvedor desejar um editor mais completo, pode utilizar *softwares* de terceiros, como o Protegé, um editor gráfico de ontologias e, em seguida, exportar as ontologias e posteriormente salvá-las no editor da plataforma Paola.

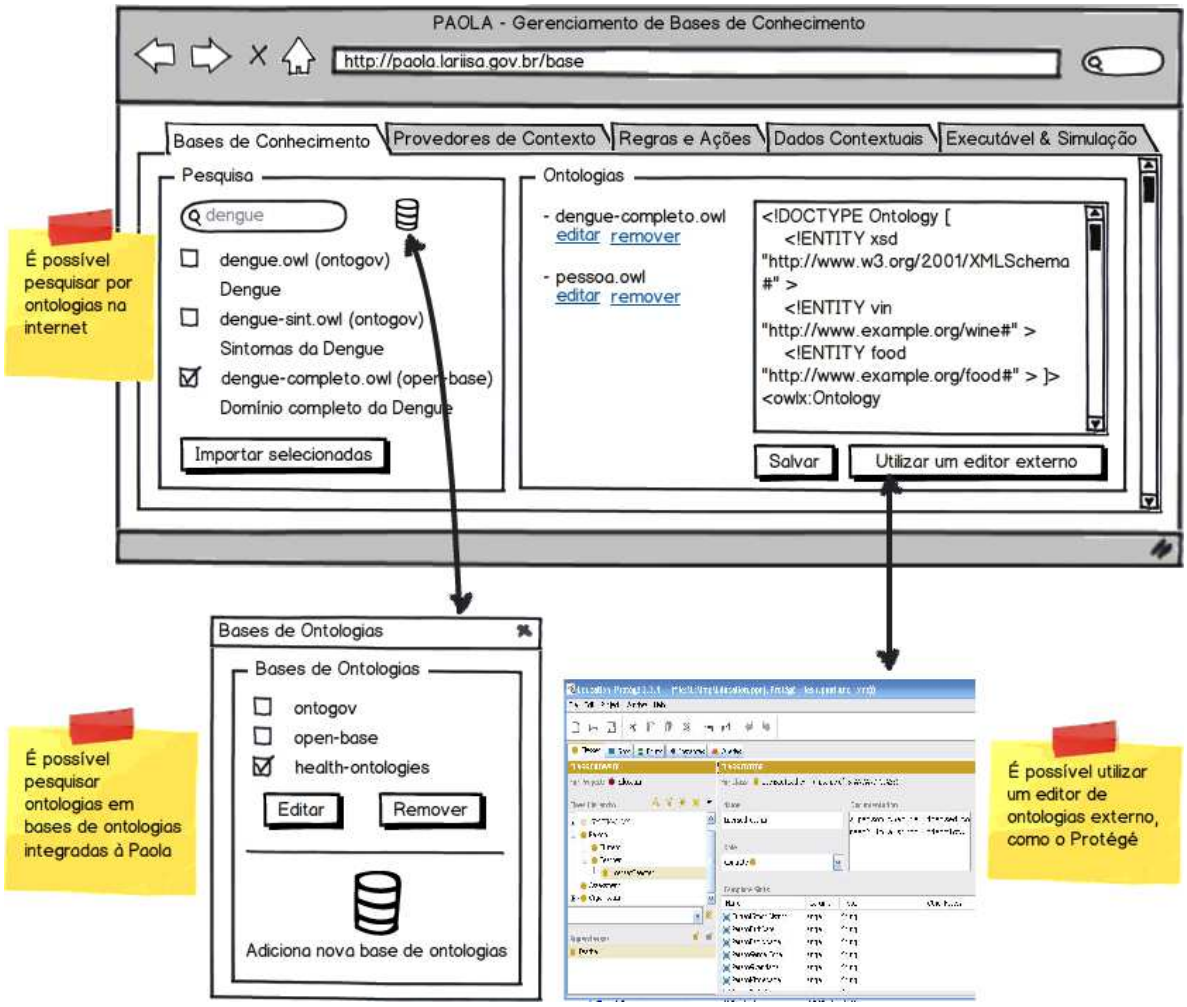

Figura 23 – Tela de edição de Bases de Conhecimento da plataforma Paola

A Figura 23 ilustra a tela projetada do módulo de edição de bases de conhecimento da plataforma Paola. Nela é possível ver as funcionalidades descritas nesta seção.

5.1.2. Funcionalidades do módulo de Manipulação de Provedores de Contexto

Provedores de contexto são agentes capazes de captar informação contextual e enviar para um sistema ciente de contexto ou um servidor de contexto. Representam dados brutos que serão analisados para representar informações contextuais.

A plataforma Paola oferece uma interface para o desenvolvedor manipular os provedores de contexto utilizando a arquitetura Lisa, que é uma arquitetura do

projeto Lariisa que facilita a integração de provedores de contexto e é utilizada nesta dissertação.

A Figura 24 ilustra o diagrama de caso de uso da edição de provedores de contexto da plataforma Paola. Neste módulo o desenvolvedor pode atuar da seguinte forma:

- Adicionar provedor de contexto.
- Adicionar diretório de provedores de contexto.
- Selecionar informações de provedor de contexto.
- Selecionar informações de diretório de provedores de contexto.
- Remover provedor de contexto.
- Remover diretório de provedores de contexto.

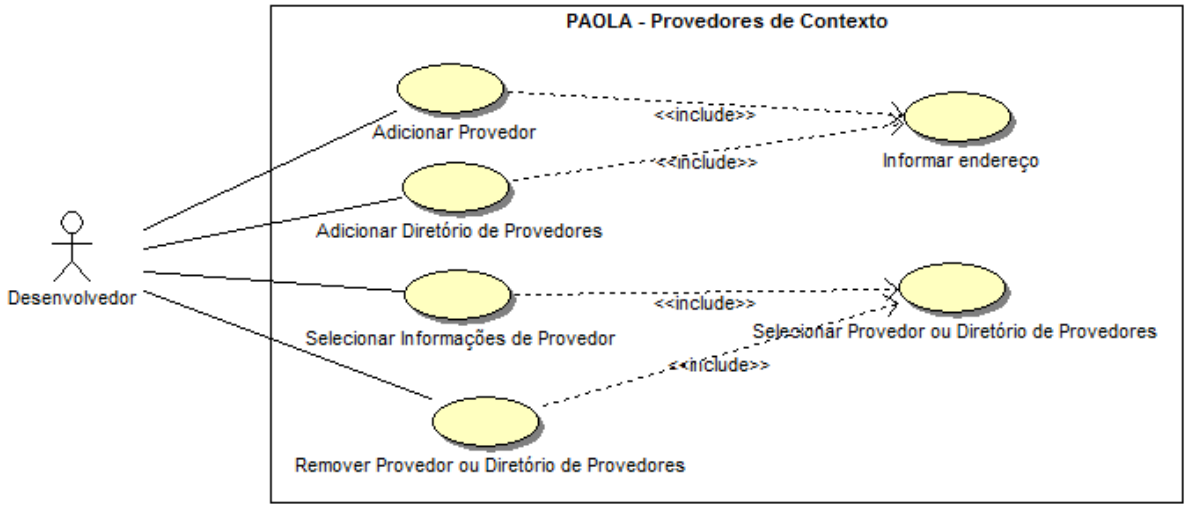

Figura 24 – Diagrama de Caso de Uso de edição de Provedores de Contexto da plataforma Paola

A Figura 25 ilustra a tela do módulo de seleção de provedores de contexto e suas informações providas. Nesta tela é possível adicionar um provedor de contexto específico mediante a digitação de um endereço IP ou de um domínio. A plataforma Paola abstrai detalhes sobre a implementação desses provedores, como sua arquitetura ou conexão, uma vez que a arquitetura da Lisa foi incorporada e serviu como infraestrutura para a plataforma Paola. Os provedores são localizados pelo endereço informado e são identificadas as informações que eles proveem. O diagrama de caso de uso associado a esta tela está ilustrado na Figura 24.

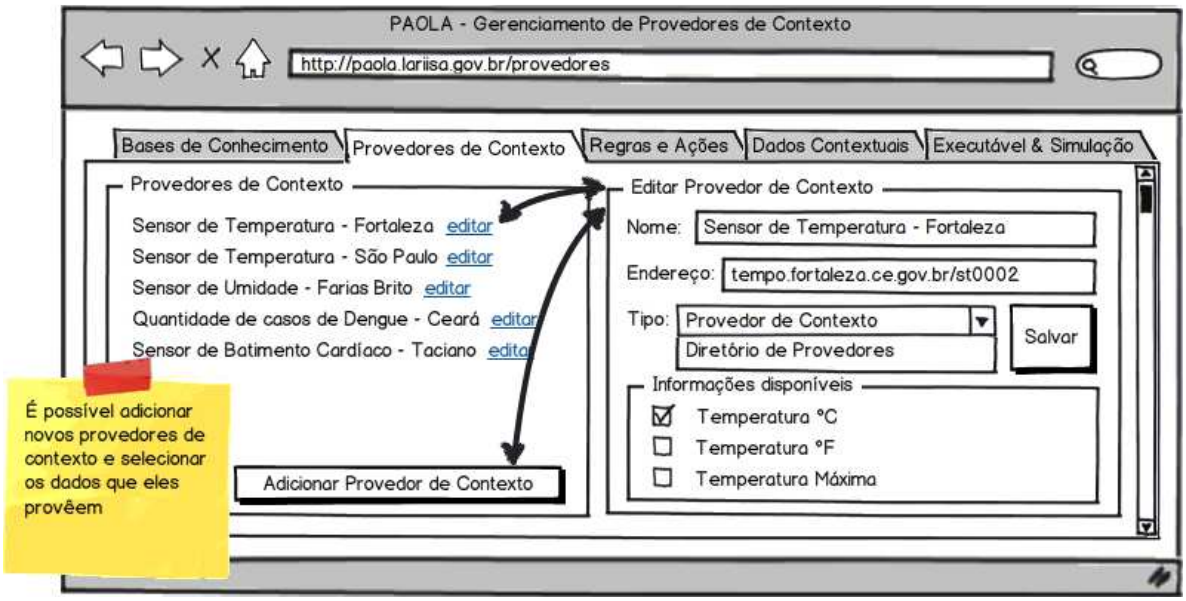

Figura 25 – Tela de edição de Provedores de Contexto da plataforma Paola

Um provedor de contexto específico é um dispositivo capaz de fornecer alguma informação contextual. Por exemplo, um celular equipado com GPS é capaz de fornecer a localização e o número do celular de um usuário.

Também é possível adicionar um conjunto de provedores de contexto, mediante a digitação do endereço de um diretório de provedores, que contém um número N de dispositivos semelhantes capazes de fornecer os mesmos tipos de informações. Por exemplo: sensores de temperatura em todos os bairros de uma cidade. Então, utilizando um diretório que representa todos esses sensores é possível saber a temperatura em cada área da cidade, pois todos eles fornecem o mesmo tipo de informação: temperatura.

5.1.3. Funcionalidades do módulo de Manipulação de Regras e Ações

O gerenciamento de regras verifica a cada alteração no contexto se uma regra foi atendida. Se sim, ele invoca o gerenciamento de ações. Neste módulo, é definido como será a verificação de regras e seu relacionamento com o Gerenciamento de Ações.

Para a edição de regras e ações, a plataforma Paola oferece uma interface para a definição de inferência e inteligência de uma aplicação sensível ao contexto. É possível criar regras e informar suas características: nome, descrição, pré-condições e ações.

Uma regra representa uma característica que o sistema infere mediante determinada configuração das informações contextuais. Um exemplo de regra na área da saúde é o de diagnóstico de dengue: o sistema detecta uma possível infeção de um paciente se ele possuir pelo menos três sintomas de dengue e morar em uma área de foco da dengue. As pré-condições são premissas para que a regra seja satisfeita e, sendo satisfeita, ela acarretará em uma ação.

A plataforma Paola mantém as regras em uma base de regras, que é um repositório de regras de decisão que representa determinados estados de configuração de dados na base de conhecimento. As regras definem as características obrigatórias para se chegar a resultados, através de inferências ou não. Neste componente, são definidas linguagens e formas de armazenamento de regras.

A Figura 26 ilustra o diagrama de caso de uso de edição de regras na plataforma Paola. Como é possível ver no diagrama, o desenvolvedor atua neste módulo da seguinte forma:

- Criar regra.
- Editar regra.
- Remover regra.
- Definir pré-condições para satisfação das regras.
- Definir ações.

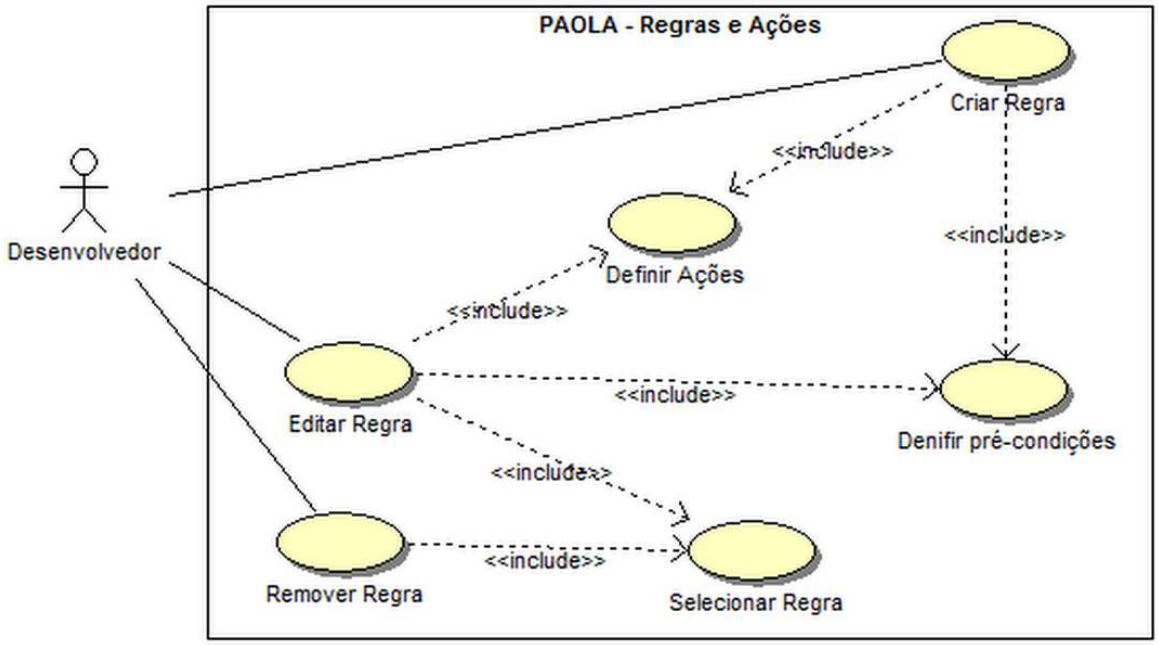

Figura 26 – Diagrama de Caso de Uso de edição de Regras na plataforma Paola

Nas pré-condições são utilizadas informações com contexto que podem ser oriundas das mais diversas fontes, obtidas através de provedores de contexto. Essas informações podem ser cruzadas com outras ou comparadas com parâmetros para a definição de regras.

Ação é uma das principais características de sistemas sensíveis ao contexto, onde um sistema reage (através das ações) a mudanças no contexto, sem necessariamente haver intervenção humana. Ação é um termo que pode representar muitas coisas em um sistema como: diagnóstico de doença, envio de sinais de alerta, investimento em bolsa de valores, entre outros.

O Gerenciamento de Ações é invocado pelo Gerenciamento de Regras (ambos na plataforma Paola em um mesmo módulo), quando alguma regra é atendida, e age para notificar uma aplicação ciente de contexto sobre a alteração de contexto. Neste módulo são definidos os relacionamentos com o Gerenciamento de Ações e como ocorre a notificação à aplicação.

A plataforma Paola tem, neste módulo, mecanismos de ações que integram sistemas através do envio de notificações via padrão *Observer*, mensagem de e-mail ou gravação em arquivo texto. Detalhes dessas técnicas são vistos no capítulo 6.

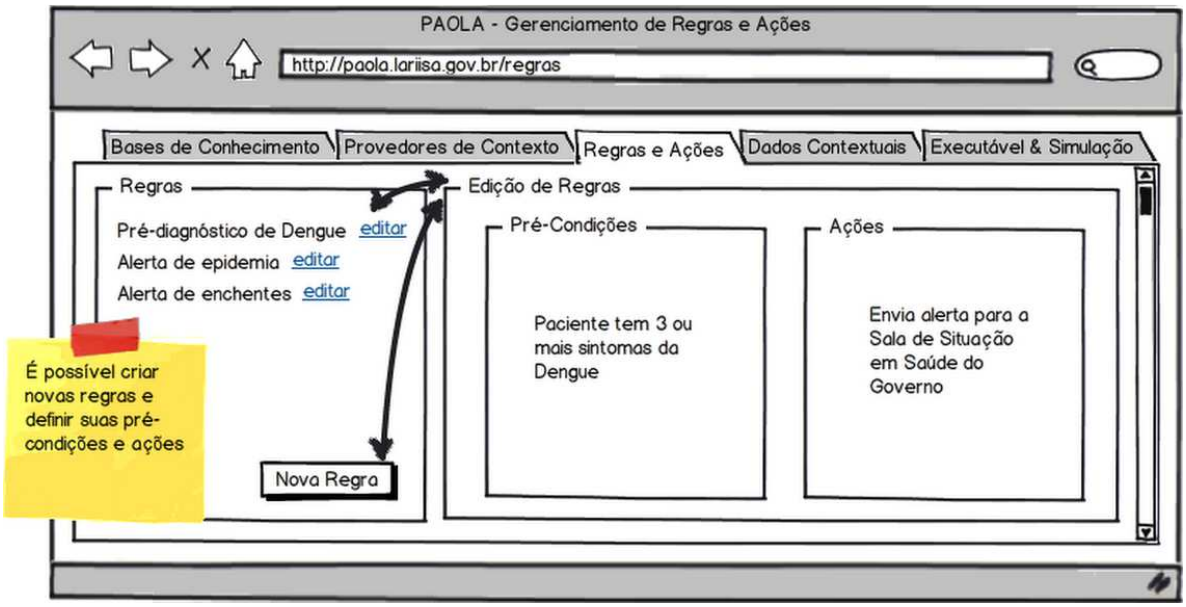

Figura 27 – Tela de edição de Regras da plataforma Paola

A Figura 27 ilustra o projeto da tela de edição de regras da plataforma Paola. Nela o desenvolvedor cria regras através da definição de pré-condições e ações e definir como as ações agirão sobre sistemas de terceiros.

5.1.4. Funcionalidades do módulo de Gerenciamento de Informação

Este módulo utiliza técnicas para realizar consultas em ontologias e inferências. Aqui é definido como o provimento de dados contextuais oriundos de provedores de contexto é refletido no sistema.

Após selecionados os provedores de contexto, estes estarão aptos a enviar informações contextuais. O armazenamento e a recuperação da informação contextual é responsabilidade do módulo de Gerenciamento da Informação. Os detalhes de implementação dos mecanismos que realizam essas tarefas são vistos no capítulo 7.

Do ponto de vista do usuário (desenvolvedor) tem-se as seguintes funcionalidades:

- Consultar dados armazenados na base de dados contextuais;
- Alterar dados armazenados na base de dados contextuais;
- Remover dados armazenados na base de dados contextuais.

Do ponto de vista dos provedores de contexto, sistemas sensíveis ao contexto e outros atores tem-se as seguintes funcionalidades:

- Armazenar dados oriundos de provedor de contexto;
- API de consulta a dados.

A Figura 28 ilustra o diagrama de caso de uso relacionado ao módulo de Gerenciamento da Informação. Contempla a visão do desenvolvedor, da aplicação sensível ao contexto, do provedor de contexto e da própria plataforma Paola.

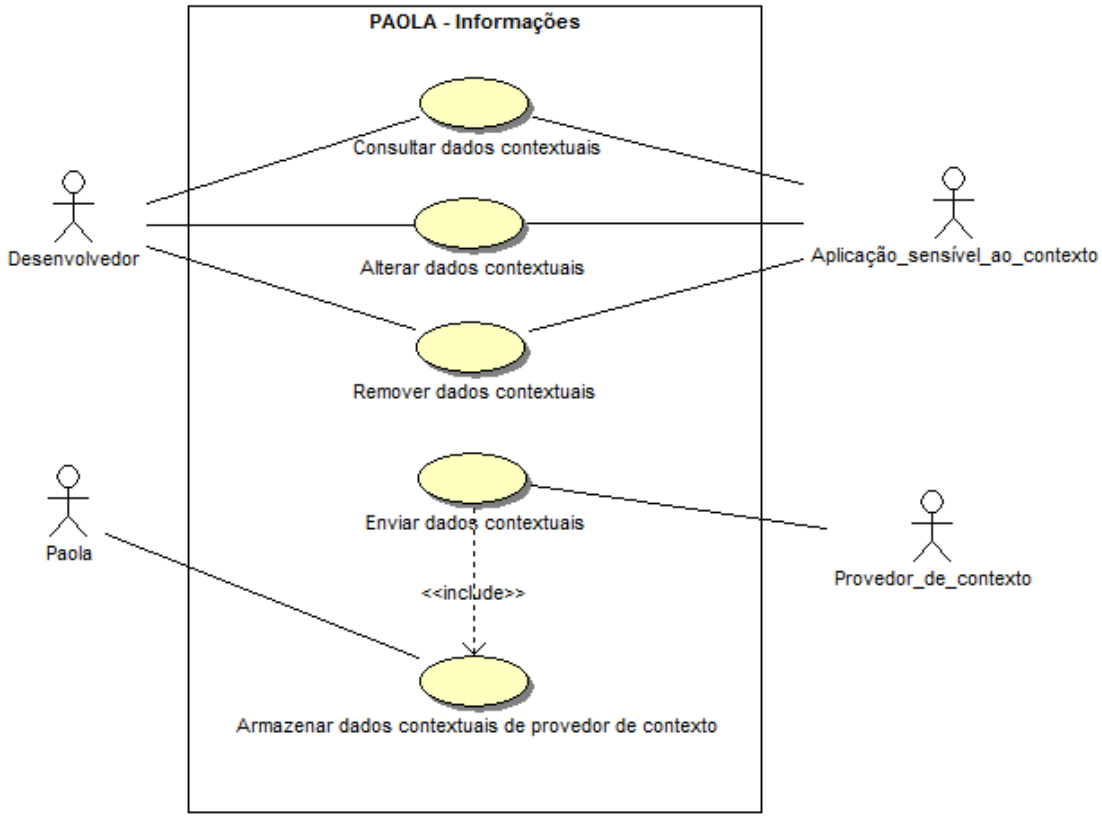

Figura 28 – Diagrama de caso de uso de Gerenciamento da Informação da plataforma Paola

Em todos os relacionamentos dos atores com os casos de uso é necessária a definição de um protocolo (API) para o envio/recebimento de dados contextuais. O desenvolvedor deve conhecer a sintaxe da linguagem de consulta para poder realizar busca às informações contextuais dos provedores de contexto e das bases de conhecimento.

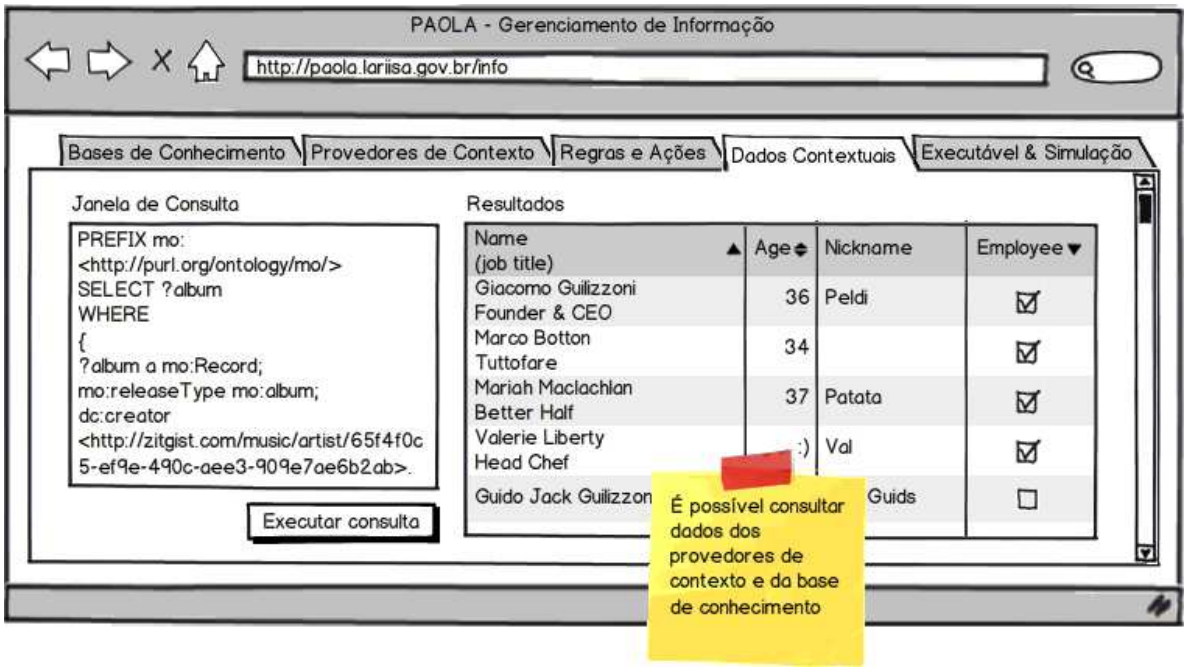

Figura 29 – Tela de Gerenciamento de Informação (consulta a dados contextuais)

A tela ilustrada pela Figura 29 permite que o desenvolvedor informe e execute uma consulta a dados. Ao lado o resultado da consulta é exibido após a sua execução.

5.1.5. Funcionalidades do módulo Gerador de Executável & Simulação

Após configurar bases de conhecimento, regras, ações, provedores de contexto e a forma de acesso às informações, o desenvolvedor pode gerar um artefato executável do projeto. Este artefato é acoplado a uma aplicação sensível ao contexto e é responsável por capturar informações contextuais, processá-las e acionar notificações para a aplicação que está inscrita para recebê-las.

O desenvolvedor pode também utilizar o executável para simular a execução na própria plataforma Paola. A plataforma oferece mecanismos para emular um ambiente sensível ao contexto com provedores de contexto e aplicação sensível ao contexto fictícios.

A Figura 30 ilustra o diagrama de caso de uso que representa a interação do desenvolvedor e da aplicação sensível ao contexto com o módulo gerador de executável & simulação. O desenvolvedor pode gerar um artefato executável e

simulá-lo, ou então, utilizar em sua aplicação: a aplicação sensível ao contexto utiliza o executável para perceber e reagir ao contexto representado.

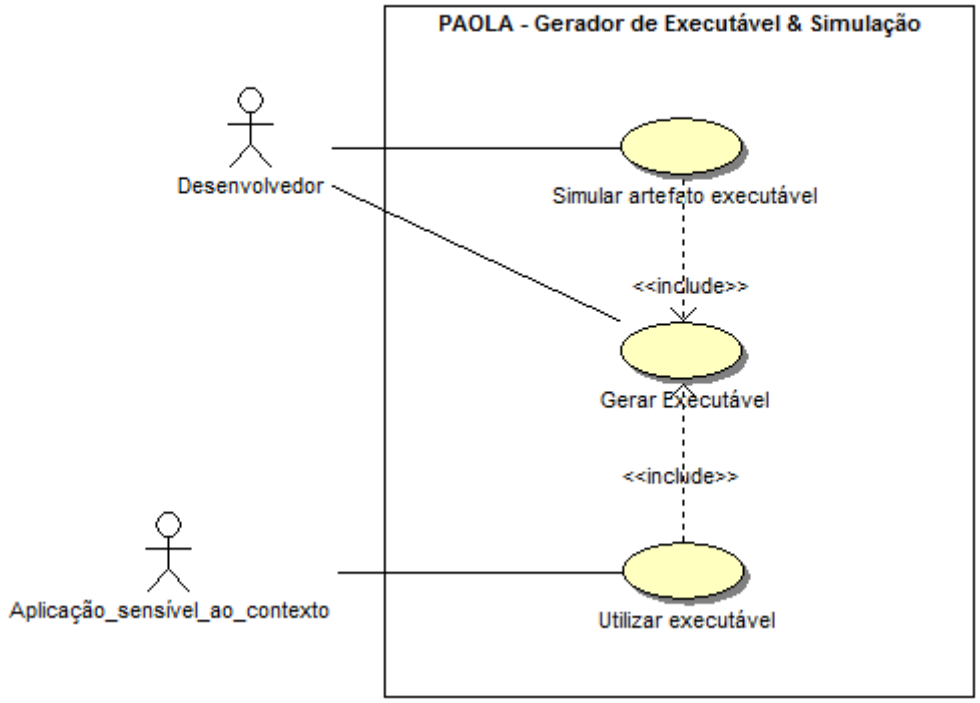

Figura 30 – Diagrama de caso de uso do módulo Gerador de artefato executável & Simulação

O artefato executável gerado por este módulo é o produto gerado pela plataforma Paola. Através dele uma aplicação pode trabalhar com a característica de contexto, interagindo diretamente com o executável para estabelecer comunicação com sensores, bases de conhecimento, notificações, entre outros.

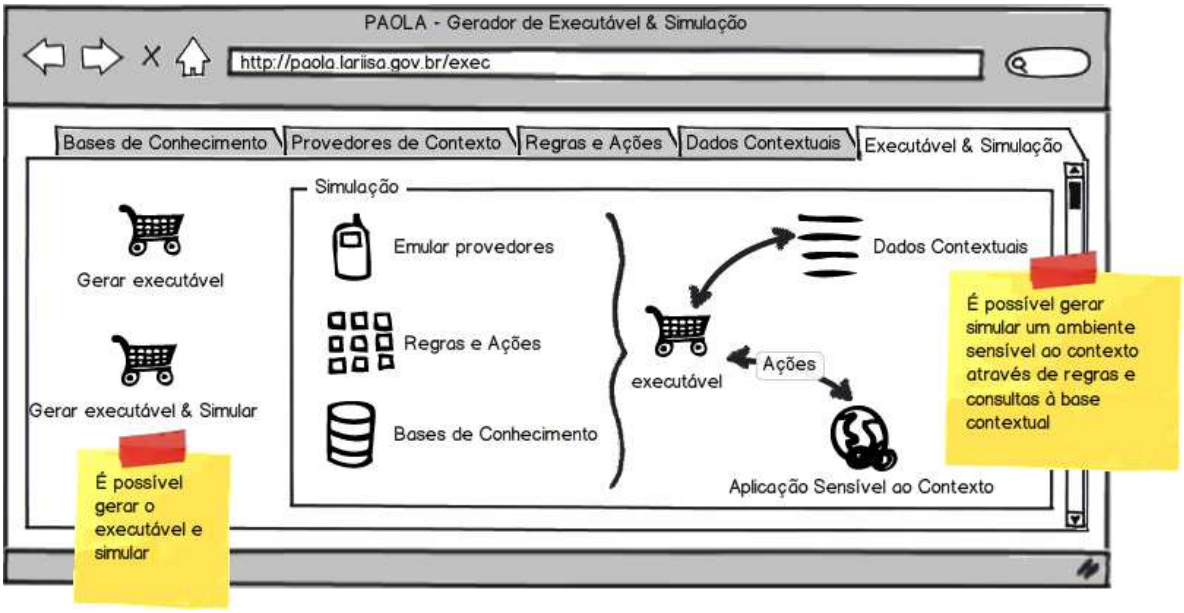

Figura 31 – Tela do Gerador de Artefato Executável & Simulação de Aplicação Sensível ao Contexto

É importante que o desenvolvedor saiba como o artefato gerado se comunica com a aplicação sensível ao contexto e com os provedores de contexto. Ele deve saber exatamente como integrar o executável ao seu sistema e conhecer também como é feita a simulação. A Figura 31 ilustra a tela do módulo Gerador de Executável e Simulação e, nela, é possível ter uma ideia de como acontece a interação do módulo executável gerado com outros componentes de um ambiente sensível ao contexto.

5.2. Considerações Finais do Capítulo

Neste capítulo, foram discorridos assuntos relativos às funcionalidades da plataforma Paola. Foram tratados aspectos de levantamento de requisitos mediante casos de uso e técnicas de prototipação de sistema através de telas. Casos de uso expandidos estão listados no apêndice A.

Foram detalhadas as funcionalidades de cada módulo: bases de conhecimento, provedores de contexto, regras e ações, informação e da geração de artefato executável & simulação do projeto.

A plataforma Paola contém uma documentação que auxilia o desenvolvedor no desenvolvimento de sua primeira aplicação sensível ao contexto para o projeto Lariisa. Essa documentação faz parte da plataforma e é acessível através do menu da plataforma Paola. Não é mostrada aqui a documentação, porém o capítulo 7 traz

um estudo de caso onde é possível ver como é o processo de desenvolvimento de um aplicação sensível ao contexto utilizando a plataforma Paola.

Na Figura 32 está ilustrado um diagrama de atividades UML que representa o fluxo de ações no sistema que um desenvolvedor deve seguir para construir um artefato executável para ser utilizado em uma aplicação sensível ao contexto. As atividades no diagrama representam ações do usuário e foram explicadas nas seções anteriores deste capítulo.

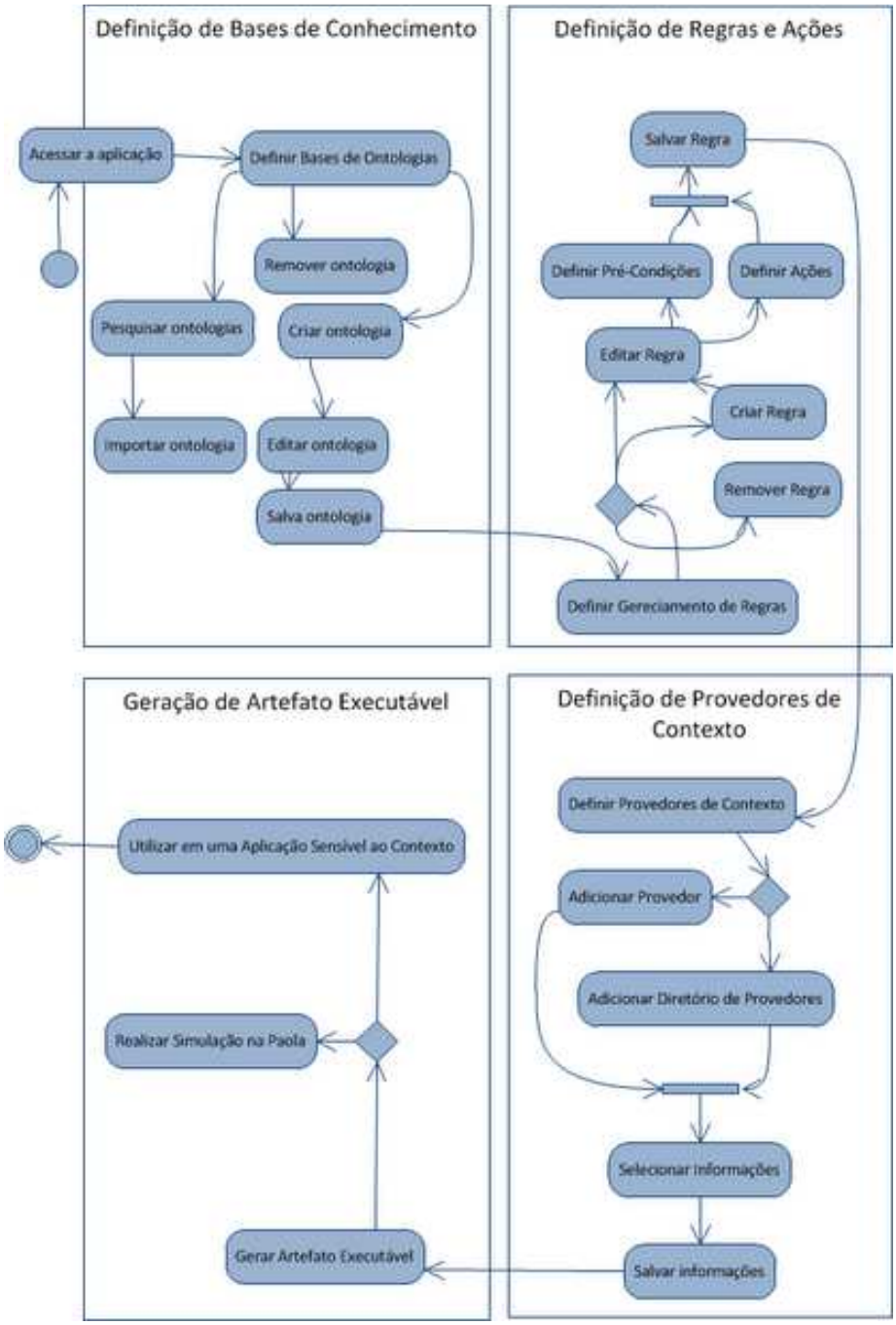

Figura 32 – Diagrama de Atividades do fluxo principal de execução da plataforma Paola

O desenvolvedor, seguindo o fluxo comum da plataforma Paola, dever passar pelas seguintes tarefas para construir seu projeto.

- Definir bases de conhecimento.
- Definir regras e ações.
- Definir provedores de contexto.
- Gerar artefato executável, simular e utilizar o artefato em sua aplicação sensível ao contexto.

Além dessas tarefas elencadas o desenvolvedor deve conhecer a API de acesso às informações contextuais.

O capítulo a seguir (6) trata dos aspectos de implementação das funcionalidades do sistema que foram descritas neste capítulo. Um processo completo de desenvolvimento é visto com detalhes no capítulo 7, que ilustra uma aplicação desenvolvida utilizando a plataforma Paola.

6. ASPECTOS DE IMPLEMENTAÇÃO

Este capítulo aborda os aspectos de implementação da plataforma Paola. São descritas as estratégicas e tecnologias utilizadas na definição e no desenvolvimento da plataforma. Apesar de não ter sido implementado completamente, são definidos vários aspectos que provam conceito e que servem como início de projeto de desenvolvimento.

As seções a seguir descrevem a arquitetura do sistema e as técnicas de implementação utilizadas em cada módulo da plataforma Paola. Ao final do capítulo é feita uma reflexão do que foi visto.

6.1. Arquitetura do Sistema

Esta seção discorre sobre a arquitetura da plataforma Paola. Ela é baseada na arquitetura do projeto Lariisa, obedecendo aos conceitos de provedores de contexto, bases de ontologias e aplicações. A Figura 33 ilustra a arquitetura da plataforma Paola. As setas indicam o sentido do fluxo de informações.

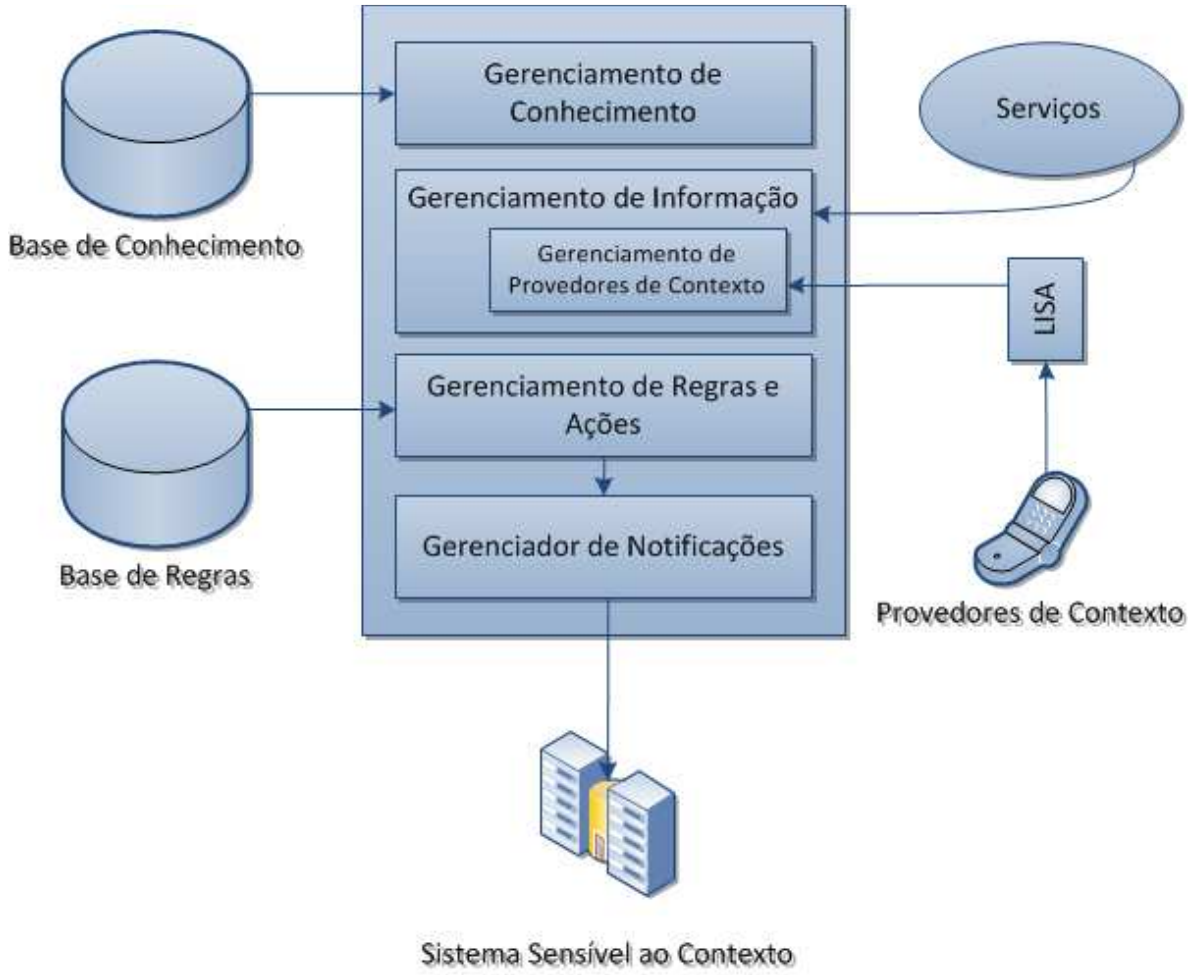

Figura 33 – Arquitetura da plataforma Paola

A seguir são descritos brevemente cada conceito presente na arquitetura mostrada na Figura 33.

- **Base de Conhecimento**. Local onde são armazenadas as ontologias que representam domínios de conhecimento.
- **Gerenciamento de Conhecimento**. Contém ferramentas que auxiliam a manipulação de ontologias.
- **Base de Regras**. Local onde são armazenadas as regras e suas pré-condições e ações associadas.
- **Gerenciamento de Regras e Ações**. Contém ferramentas que auxiliam a manipulação de regras incluindo as suas pré-condições e ações.
- **Provedores de Contexto**. Dispositivos capazes de capturar informação contextual de um ambiente.

- **LISA**. Infraestrutura para a integração de provedores de contexto do projeto Lariisa.
- **Gerenciamento de Provedores de Contexto**. Contém ferramentas que auxiliam a manipulação de provedores de contexto.
- **Gerenciamento de Informação**. Contém ferramentas que auxiliam a manipulação de dados contextuais.
- **Serviços**. Serviços dos mais diversos podem se beneficiar da base de conhecimento da plataforma Paola, através de consultas das informações do sistema. Este módulo é responsável por fornecer um API para que essa consulta seja realizada.
- **Sistema Sensível ao Contexto.** Aplicação capaz de perceber o contexto de um ambiente. Ela é capaz de receber notificações e atualizar a base de informação.

É fundamental conhecer cada conceito da arquitetura para entender como a plataforma Paola funciona. Nas seções a seguir são vistos os aspectos de implementação dos módulos principais da plataforma. Os conceitos da arquitetura serão bastante citados neste capítulo.

6.2. Aspectos de Implementação do módulo de Gerenciamento de Base de Conhecimento

Esta seção aborda os aspectos de implementação referentes ao módulo de Gerenciamento de Bases de Conhecimento da plataforma Paola. São mostradas as estratégias utilizadas para a construção do módulo e as tecnologias utilizadas. O funcionamento das bases de conhecimento foi visto com detalhes na seção 5.1.1.

As subseções a seguir detalham cada parte do módulo.

6.2.1. Manipulação de Bases de Ontologias

Algumas regras regem como acontece a manipulação de bases de ontologias, contemplando as operações de adicionar, alterar e remover bases. Também são definidas regras para a consulta a essas bases e importação de ontologias.

Para a manipulação de bases de ontologias, a plataforma Paola define uma base de dados em XML que contém as informações das bases de ontologias

associadas à plataforma. Para a criação da base de dados foi definido um XML *Schema* (XSD), conforme Quadro 12. O esquema contém uma estrutura bem simples capaz de representar o nome e a URL de uma ou mais bases de ontologias no Repositório de bases de Ontologias.

```
<?xml version=”1.0” encoding=”ISO-8859-1”?>
<xs:schema xmlns:xs=”http://paola.larissa.gov.br/2012/BaseDeOntologiasXMLSchema”>
        <xs:element name=”base”>
                <xs:complexType>
                        <xs:sequence>
                                <xs:attribute name=”baseId” type=”xs:integer” use=”required”/>
                                <xs:element name=”nome” type=”xs:string”/>
                                <xs:element name=”url” type=”xs:anyURI” use=”required”/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
</xs:schema>
```

Quadro 12 – XML *Schema* do Repositório de Bases de Ontologias

A partir do XSD é possível criar um arquivo XML contendo as informações de várias bases de ontologias. As operações de inclusão, alteração e remoção de bases de ontologias são refletidas diretamente no arquivo XML.

6.2.2. Manipulação de Bases de Conhecimento

As ontologias podem ser obtidas das bases de ontologias descritas na seção anterior (6.2.1) ou criadas pelo próprio desenvolvedor (utilizando o editor da plataforma Paola ou um editor externo, como o Protégé – ilustrado na Figura 34). A partir das ontologias são formadas as bases de conhecimento que a plataforma Paola manipula.

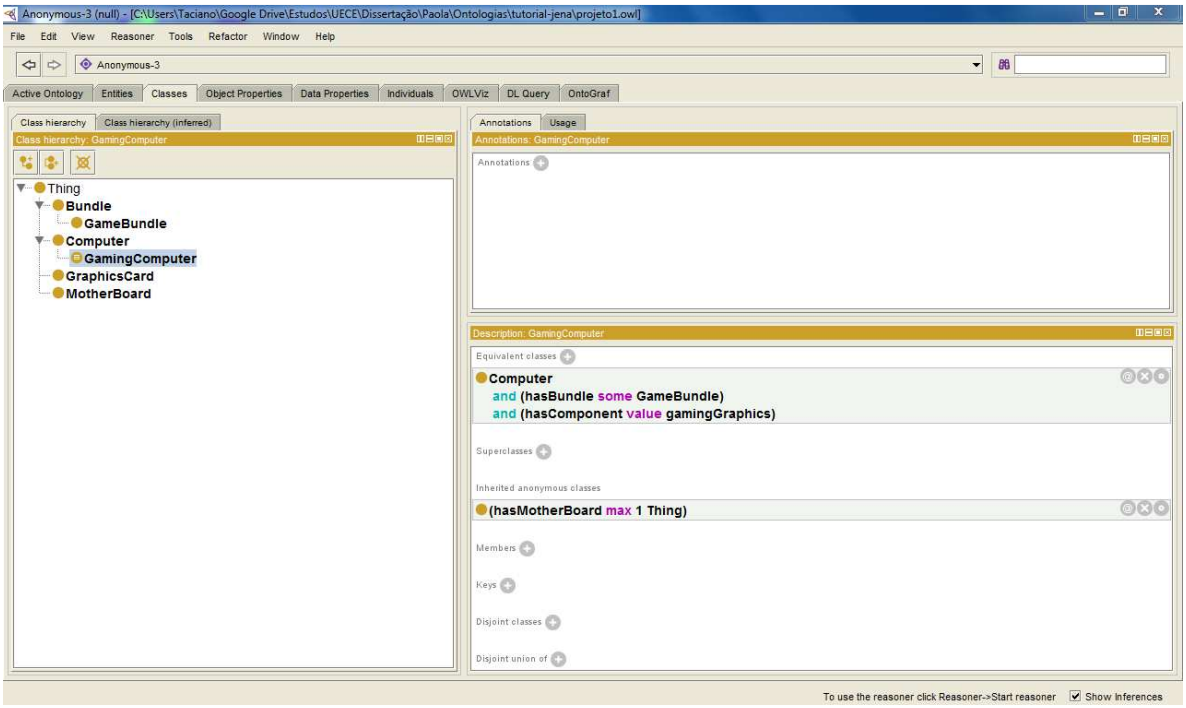

Figura 34 – Tela de edição de ontologia do Protégé

As ontologias, na plataforma Paola, são representadas através da linguagem OWL e são manipuladas utilizando o Jena, um *framework* para *web* semântica construído com a linguagem de programação Java. Com esse *framework* a plataforma Paola representa conhecimento e manipula ontologias, através da definição de classes, entidades, propriedades, entre outros. O Jena provê uma coleção de ferramentas e bibliotecas Java para ajudar o desenvolvedor (que constrói a plataforma Paola) a construir aplicações para *web* semântica e seus dados, ferramentas e servidores. A Figura 35 ilustra como o Jena está disposto na comunicação da plataforma Paola com a base de conhecimento.

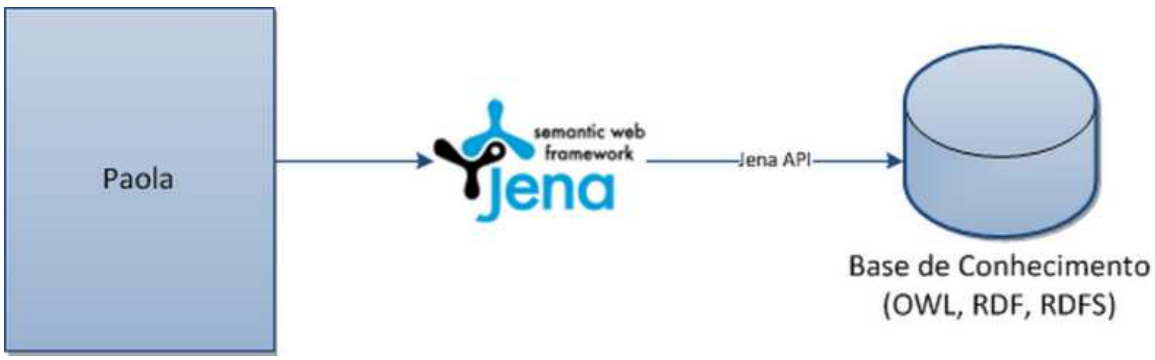

Figura 35 – Aspectos de implementação do Módulo de Gerenciamento de Bases de Conhecimento

Para codificar com o Jena foi utilizada a IDE Eclipse e a linguagem de programação Java. Diversas operações são feitas utilizando o Jena sobre uma ontologia OWL, como a carga e a manipulação das ontologias, dados e inferências. A Figura 36 ilustra uma tela do ambiente de desenvolvimento do módulo de manipulação de bases de conhecimento.

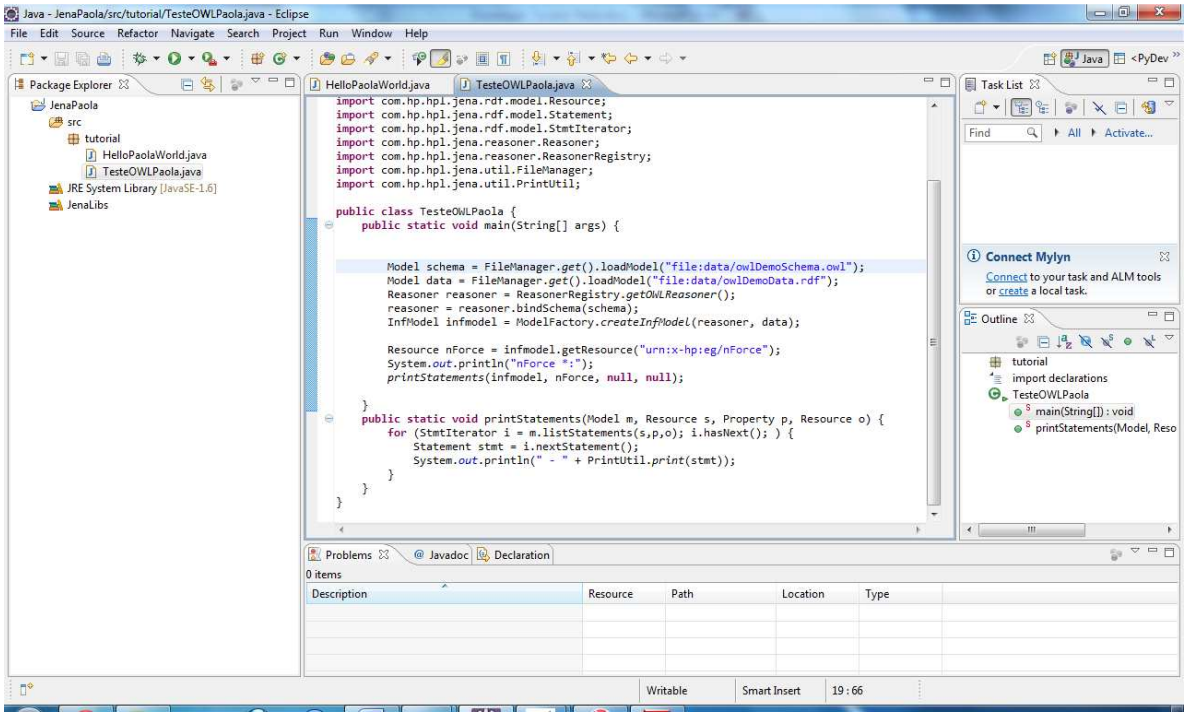

Figura 36 – Tela de desenvolvimento da manipulação de bases de conhecimento

A utilização do Jena é importante para fornecer à aplicação sensível ao contexto uma maneira mais fácil de representar o contexto, disponibilizar e manipular as informações contextuais. Outros módulos da plataforma Paola fazem a utilização de serviços fornecidos pelo Jena ou utilizando o Jena.

6.3. Aspectos de Implementação do módulo de Gerenciamento de Provedores de Contexto

O módulo de Gerenciamento de Provedores de Contexto é responsável por manter um repositório de provedores de contexto conectados e por captar os dados contextuais fornecidos por eles. Detalhes do funcionamento deste módulo foram vistos na seção 5.1.2.

Os provedores de contexto são conectados à plataforma Paola através da infraestrutura fornecida pela arquitetura Lisa (ver seção 6.3.2). Os provedores enviam informações contextuais que são armazenadas nas bases de conhecimento. Esta seção aborda como isso ocorre na plataforma.

As subseções a seguir detalham aspectos de implementação da manipulação do repositório de provedores de contexto, da integração com a arquitetura Lisa e da captação de dados contextuais.

6.3.1. Manipulação de Provedores de Contexto

A plataforma Paola tem a funcionalidade de integrar os provedores de contexto à aplicação sensível ao contexto. O desenvolvedor informa o nome do provedor, seu endereço (URL) e seu tipo (provedor simples ou diretório de provedores) e pode integrar o provedor ao seu projeto. Também é possível alterar ou remover essas informações.

Para representar e armazenar as informações dos provedores de contexto foi definido um XSD com elementos e atributos necessários para conectar um provedor, que pode ser visto no Quadro 13.

```
<?xml version=”1.0” encoding=”ISO-8859-1”?>
<xs:schema xmlns:xs=”http://paola.larissa.gov.br/2012/ProvedorDeContextoXMLSchema”>
	<xs:element name=”provedor”>
		<xs:complexType>
			<xs:sequence>
				<xs:attribute name=”provedorId” type=”xs:integer” use=”required”/>
				<xs:attribute name=”simples” type=”xs:boolean” use=”required”/>
				<xs:element name=”nome” type=”xs:string”/>
				<xs:element name=”url” type=”xs:anyURI” use=”required”/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>

	## criar elemento para representar dados contextuais ##

</xs:schema>
```

Quadro 13 – XML Schema do repositório de bases de ontologias

A partir do XSD é possível criar um arquivo XML contendo as informações de vários provedores de contexto. As operações de inclusão, alteração e remoção de bases de ontologias são refletidas diretamente no arquivo XML.

6.3.2. Integração com a arquitetura Lisa

A arquitetura Lisa disponibiliza o *Enterprise Service Bus* (ESB), um serviço que agrega serviços de diferentes tecnologias de provedores. A Figura 37 ilustra, de maneira geral, como ocorre a comunicação dos provedores de contexto com o projeto Lariisa, utilizando *WebServices* via ESB.

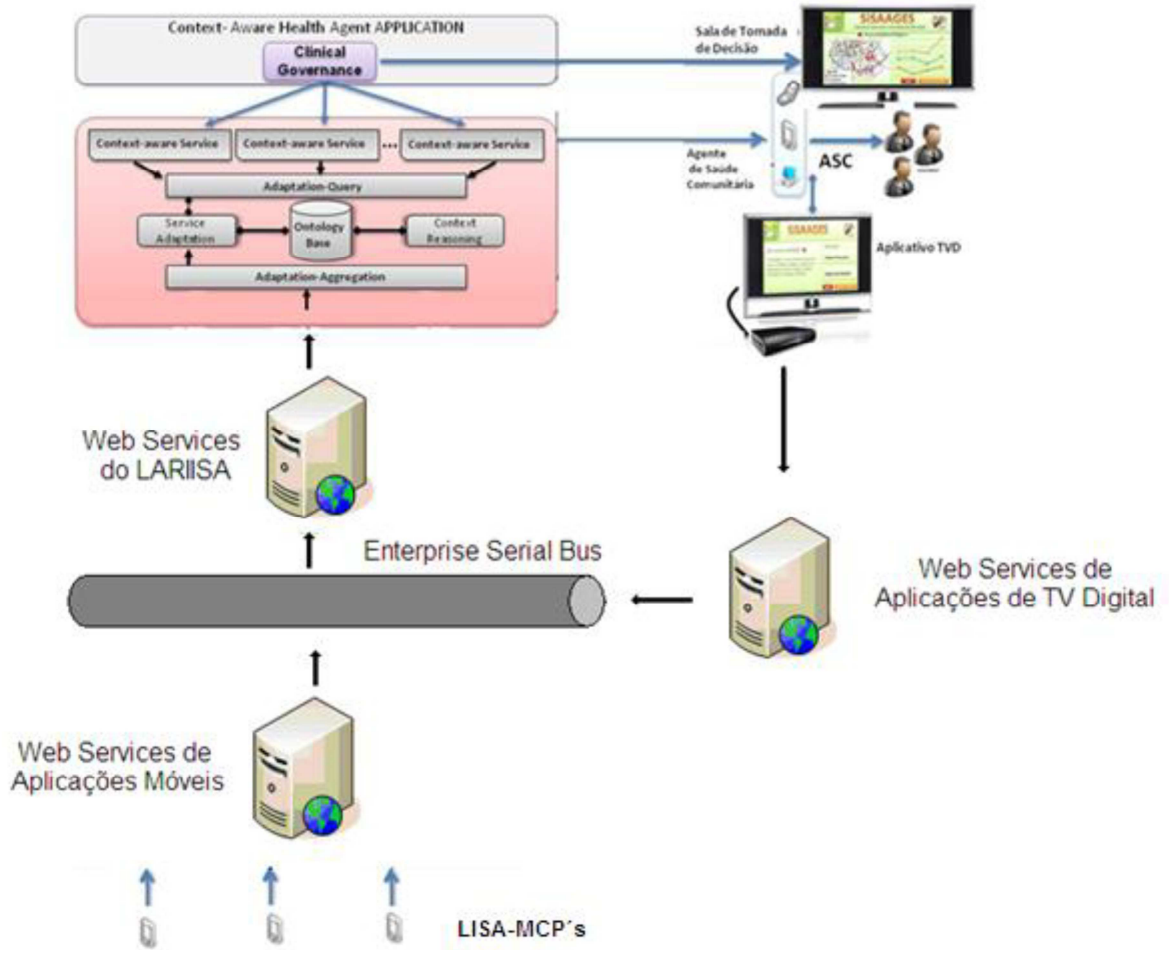

Figura 37 – Integração de provedores de contexto ao projeto Larissa com o Lisa

Integrado ao projeto Lariisa, a plataforma Paola possui mecanismos de ler dados dos *WebServices* e fornecer ao desenvolvedor informações de um nível maior de abstração. Tais informações representam a carga útil transportada pela arquitetura Lisa, que são os dados contextuais e suas representações.

6.3.3. Captura de Dados

A plataforma Paola utiliza um QA para realizar inserções de dados na base de dados contextual. Juntamente com o QA, a ferramenta Joseki, através da Sparql,

recebe as requisições HTTP *Get* e *Post* do QA e as executa nas bases de conhecimento. Uma visão geral deste cenário é ilustrada na Figura 38. Resumindo, as técnicas apresentadas realizam a comunicação dos provedores de contexto com as bases de conhecimento, utilizando a arquitetura Lisa, o QA, requisições HTTP e o Joseki.

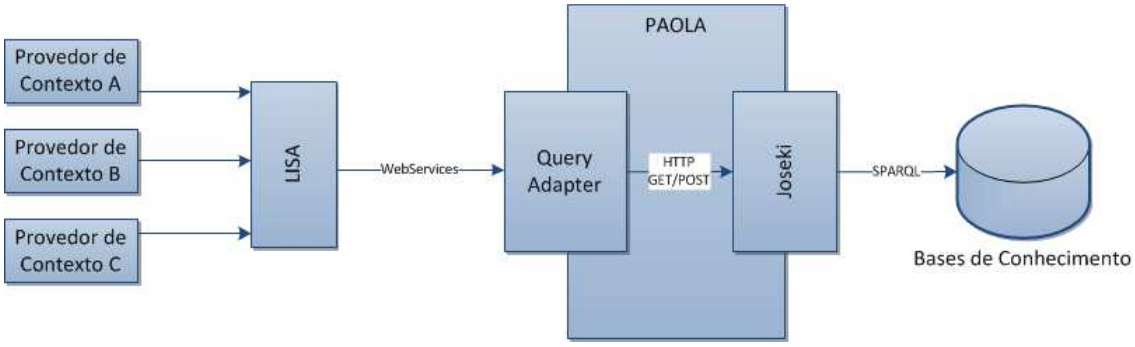

Figura 38 – Exemplo de integração de Provedores de Contexto à plataforma Paola utilizando a arquitetura Lisa

Para inserir dados contextuais na base de conhecimento é utilizada a linguagem SPARQL/*Update*, que é uma sublinguagem da SPARQL responsável por inserir, atualizar ou remover tripas (dados) contextuais.

Para inserir dados provenientes de provedores de contexto, o desenvolvedor deve criar consultas simples (semelhantes as da linguagem SQL) de acordo com a configuração dos seus dados. O Quadro 14 mostra um exemplo de consulta SPARQL para inserção de tripla.

```
PERFIX dc: <http://purl.com/dc/elements/1.1/>
INSERT DATA
{ <http://example/book3> dc:title “A new book” ;
                         dc:creator “A.N.Other” .
}
```

Quadro 14 – Exemplo de inserção da base de dados contextual

O exemplo do Quadro 14 utiliza a ontologia ‘dc:*creator’*, utilizada para representar uma entidade primária responsável por criar ou fazer algo. No exemplo é apresentado um autor de um livro, porém pode ser utilizado em qualquer cenário em que o autor produz alguma coisa.

Consultas de inserção semelhantes a exibida no Quadro 14 são utilizadas pela plataforma Paola para obter dados dos provedores de contexto. A linguagem

Sparql também é utilizada na recuperação de dados contextuais, conforme descrito na seção 6.5.

Para receber as requisições de consultas SPARQL em ambientes *web*, como o da plataforma Paola, foi utilizado o servidor de SPARQL Joseki, que recebe e atende requisições web. Dos *WebServices* do projeto Lariisa para a integração com a arquitetura Lisa, através do QA, são realizadas requisições HTTP *Post* contendo uma consulta SPARQL/Update. O QA é responsável por mediar a comunicação entre os *WebServices* e o Joseki via protocolo HTTP. O Joseki, por sua vez, executa as consultas em formato SPARQL diretamente nas bases de conhecimento, utilizando o Jena.

6.4. Aspectos de Implementação do módulo de Gerenciamento de Regras e Ações

Esta seção aborda os aspectos de implementação do módulo de Gerenciamento de Regres e Ações na plataforma Paola. Detalhes do funcionamento deste módulo foram vistos na seção 5.1.3.

6.4.1. Regras

Para manipular regras, o desenvolvedor utiliza a interface gráfica da plataforma Paola. Internamente as regras são representadas através de um módulo do *framework* Jena: *general purpose rule engine*. Com isso é possível realizar inferências nas bases de conhecimento ou detectar mudanças de contexto para o acionamento de ações.

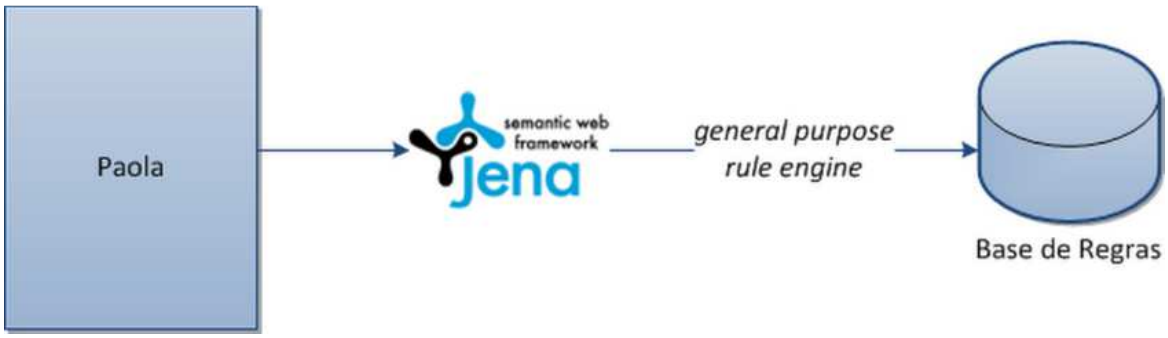

Figura 39 – Aspectos de implementação das Regras do Módulo de Gerenciamento de Regras e Ações

As regras são armazenadas em arquivo seguindo a sintaxe determinada pelo Jena e são manipuladas através de uma API. A Figura 39 ilustra como a plataforma Paola trata regras na base de regras, através do Jena e utilizando o *general purpose rule engine*.

Regras contêm pré-condições, que são as condições necessárias para que regras sejam satisfeitas. Regras podem estar associadas a ações, que são vistas na subseção a seguir (6.4.2).

As regras são descritas através do módulo de gerenciamento de regras e ações utilizando a interface gráfica, como visto no capítulo anterior na seção 5.1.3. Internamente a plataforma Paola trata as regras utilizando o Jena e o *general purpose rule engine*, seguindo uma linguagem pré-definida pelo Jena para a representação de regras. Através da linguagem é possível implementar um raciocinador (*reasoner*) baseado em regras para ontologias OWL. O Quadro 15 ilustra um exemplo simples de uma regra que pode ser criada utilizando o Jena. A regra verifica se uma pessoa é irmã do seu pai, se sim, então essa pessoa é seu tio.

```
@prefix pre: <http://jena.hpl.hp.com/prefix#>
@include <RDFS>

[rule1: (?f pre:pai ?a) (?u pre:irmao ?f) -> (?u pre:tio ?a)]
```

Quadro 15 – Exemplo de regra na Jena/GPRE

As regras são lidas pelas bibliotecas do Jena e processadas. O gerenciamento de regras aciona o gerenciamento de ações quando as pré-condições forem satisfeitas e se assim forem definidas nas regras.

6.4.2. Ações

Para manipular as ações a plataforma Paola utiliza o padrão *Observer,* que permite que sistemas interessados nas notificações sejam avisados da mudança de estado ou da satisfação de alguma regra definida pelo desenvolvedor.

A plataforma Paola implementa o lado *publisher* do padrão *Observer*, enquanto a aplicação sensível ao contexto implementa o lado *subscriber*. Ao integrar o módulo executável gerado pela plataforma Paola (ver seção 5.1.5) a aplicação sensível ao contexto já pode perceber o contexto e ser notificada sobre suas mudanças e eventos.

Pelo nome da ação (ver seção 5.1.3) a aplicação pode se cadastrar para receber as notificações da ação, que por sua vez verifica automaticamente as pré-condições de sua regra associada.

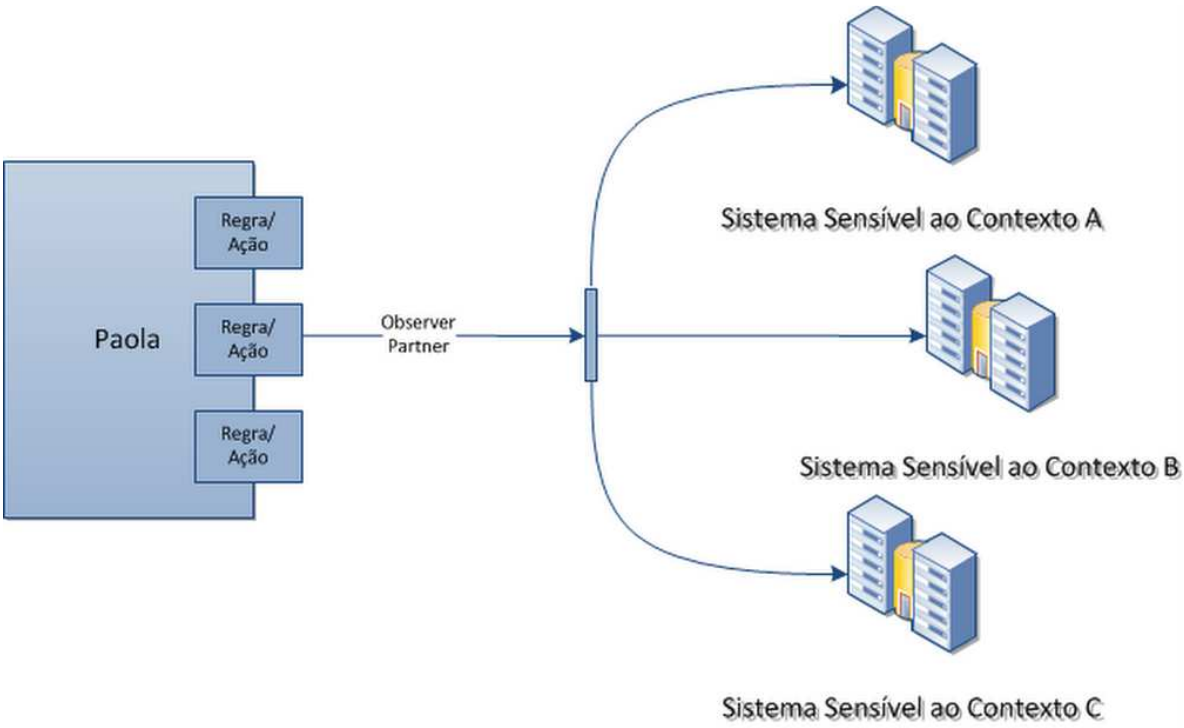

Figura 40 – Aspectos de implementação de Ações do Módulo de Gerenciamento de Regras e Ações

A Figura 40 ilustra um exemplo de situação onde três sistemas sensíveis ao contexto (*listeners*) estão “observando” a plataforma Paola e prontos para receberem notificações, conforma as configurações de suas regras e ações. A plataforma Paola fornece uma API, independente de linguagem de programação, para que os sistemas interessados possam implementar o padrão *Observer* seguindo as estruturas de notificação sensível ao contexto.

O módulo de gerenciamento de regras e ações é integrado às classes Java que implementam o padrão *Observer* e disponibilizam uma API para que aplicações interessadas possam se cadastrar para receber notificações, através do artefato executável.

6.5. Aspectos de Implementação do módulo de Gerenciamento de Informação

Esta seção aborda os aspectos de implementação do gerenciamento e manipulação de informação. Detalhes do funcionamento deste módulo foram vistos na seção 5.1.4.

É neste módulo que são definidas as formas de acesso aos dados e informações processadas, assim como a integração para a mediação da comunicação com provedores de contexto (visto na seção 6.3.3). A Figura 41 ilustra os aspectos de implementação do módulo.

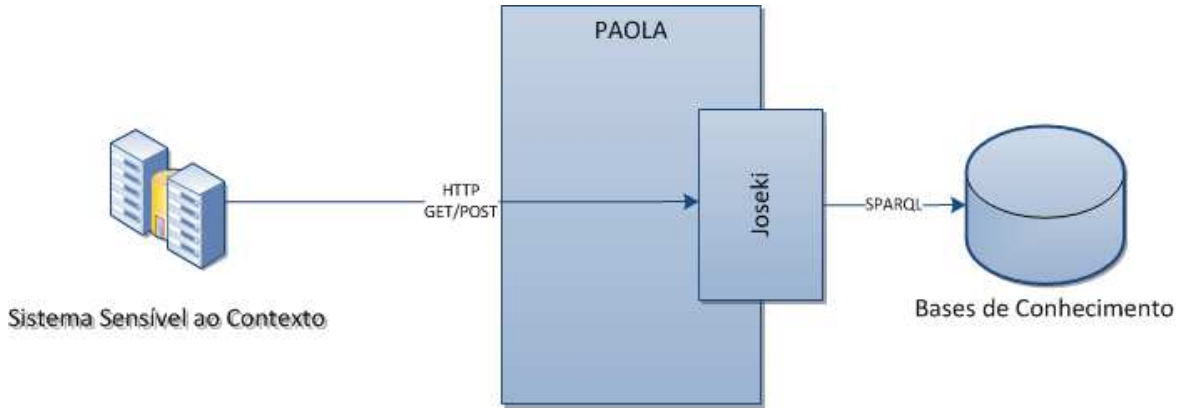

Figura 41 – Aspectos de implementação do Módulo de Gerenciamento de Informação

Assim como são processados os dados de provedores de contexto, a plataforma Paola utiliza requisições HTTP para o Joseki, que por sua vez executa as consultas via SPARQL na base de conhecimento. Um exemplo de consulta SPARQL é mostrado no Quadro 16. A consulta recupera os nomes e endereços (URLs) de provedores de contexto na base.

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?nome ?url
WHERE
        { ?x foaf:name ?url .
          ?x foaf:url ?url }
```

Quadro 16 – Exemplo de consulta a dados em base de conhecimento

É possível realizar consultas de busca (*select)* ou alteração (*update)* de dados. A aplicação sensível ao contexto do cliente deve enviar as requisições HTTP à plataforma Paola (Joseki) que se encarrega de executá-las.

Como resultado de uma consulta o Joseki retorna um arquivo XML com os dados encontrados na base de conhecimento. Um exemplo de resultado de uma consulta pode ser visto no Quadro 17. A consulta recupera dados de um usuário e seu contexto, no caso os sintomas da Dengue.

```
<?xml version=”1.0”?>
<sparql xmlns =”http://www.w3.org/2005/sparql-results#”>
        <head>
                <variable name=”user”/>
                <variable name=”context”/>
        </head>
        <results>
                <result>
                        <binding name=”user”>
                                <uri>http://paola.lariisa.gov.br/sisa.owl#ResidenciaPaciente1</uri>
                        </binding>
                         <binding name=”context”>
                                <uri> http://paola.lariisa.gov.br/sisa.owl#3SintomasDengue</uri>
                        </binding>
                </result>
         ...
        </results>
 </sparql>
```

Quadro 17 – Exemplo de resultado de consulta a dados em base de conhecimento

A aplicação sensível ao contexto deve interpretar o arquivo XML recebido como resultado da consulta para utilizar seu conteúdo (dados). Tal interpretação fica a cargo da aplicação e de seu domínio de atuação.

6.6. Considerações Finais do Capítulo

Este capítulo abordou os aspectos de implementação da plataforma Paola. Foi explanada sua arquitetura e as técnicas e tecnologias aplicadas em cada um dos módulos da plataforma. Foram citados apenas os principais aspectos que mostraram relevância para a discussão e alguns detalhes de implementação foram omitidos. A plataforma Paola não foi integrada totalmente, seus módulos foram implementados isoladamente. A implementação integrada de todos os módulos da plataforma Paola é sugerida como trabalho futuro.

No início do capítulo foi mostrada a arquitetura da plataforma. O conhecimento da arquitetura é fundamental para o entendimento dos aspectos de implementação dos módulos. A definição da arquitetura serve de base para todo o desenvolvimento da plataforma Paola.

As demais seções discorreram sobre os aspectos de implementação. Em cada módulo foi mostrado como se deu o processo de desenvolvimento e as estratégias e tecnologias utilizadas para a sua construção.

A seção 6.3.2 abordou aspectos de integração com a arquitetura Lisa utilizando *WebServices*. A plataforma Paola aproveitou várias funcionalidades e implementações realizadas por Frota (2011).

Cada detalhe mostrado neste capítulo foi analisado com base na experiência do autor e orientador em processos de desenvolvimento e objetivando obter bons resultados para o projeto Lariisa e sua integração com outros componentes.

7. ESTUDO DE CASO: APLICAÇÃO SISA

Para validar uma prova de conceito da plataforma Paola, este capítulo mostra um estudo de caso que aborda a criação da parte sensível ao contexto de uma aplicação utilizando a plataforma. O artefato executável a ser gerado pela plataforma proposta nesta dissertação é um sistema na área de controle epidemiológico da Dengue, utilizando informações dos pacientes em suas residências, de agentes de saúde, de sistemas de terceiros e de informações registradas por gestores de saúde.

A parte sensível ao contexto do Sisa descrita nesta dissertação é um subconjunto de suas funcionalidades. É mostrado, nas seções seguintes, um contexto simples de diagnóstico de Dengue baseado em sintomas. O contexto é modelado com ontologias e seus dados oriundos de provedores de contexto utilizando a arquitetura Lisa. Regras e ações também são definidas pela plataforma Paola para atuar com notificações para a aplicação Sisa.

As seções a seguir abordam o desenvolvimento de um projeto na plataforma Paola em cada um dos seus módulos.

7.1. Criação de Ontologias

Para criar as ontologias é utilizado o editor de ontologias Protégé. São definidas várias classes, subclasses e propriedades dos objetos. Também são definidas, em cada entidade/objeto criado, suas cardinalidades, restrições universais, classes primitivas e derivadas, condições necessárias/suficientes e outros aspectos relacionados à criação de uma ontologia. A Figura 42 é gerada pela ferramenta OntoGraf e representa a ontologia criada, mostrando os relacionamentos de suas entidades.

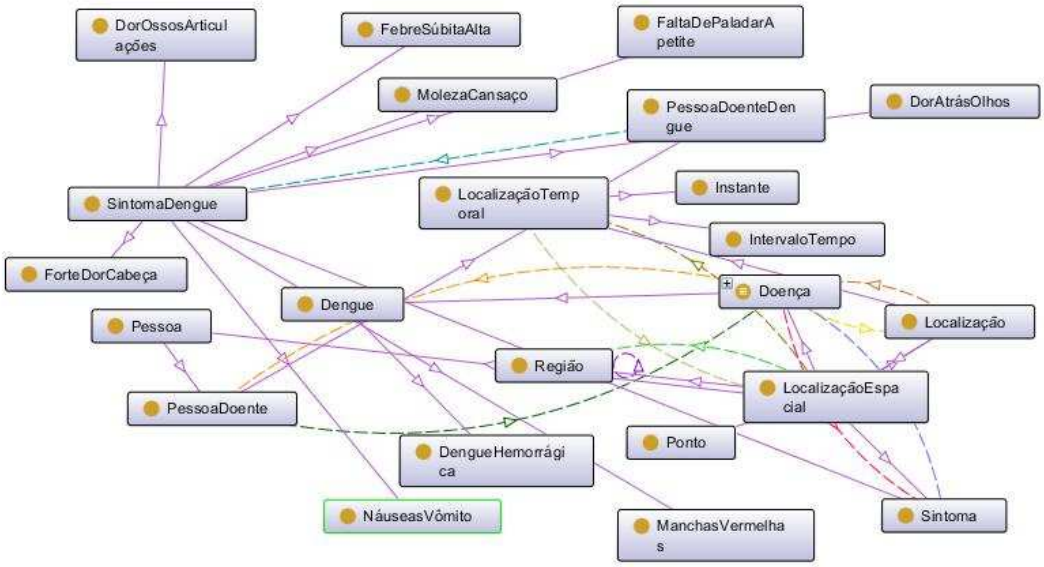

Figura 42 – Gráfico da ontologia

A ontologia é muito simples e apenas realiza um diagnóstico prévio de uma infecção de um paciente com suspeita de Dengue. Nas regras da ontologia são verificados se uma pessoa possui três ou mais sintomas e, se possuir, realiza algumas ações para realizar o diagnóstico prévio de dengue. A Figura 43 ilustra o ambiente de desenvolvimento da ontologia ‘PaolaSisa.owl’ criada para este estudo de caso.

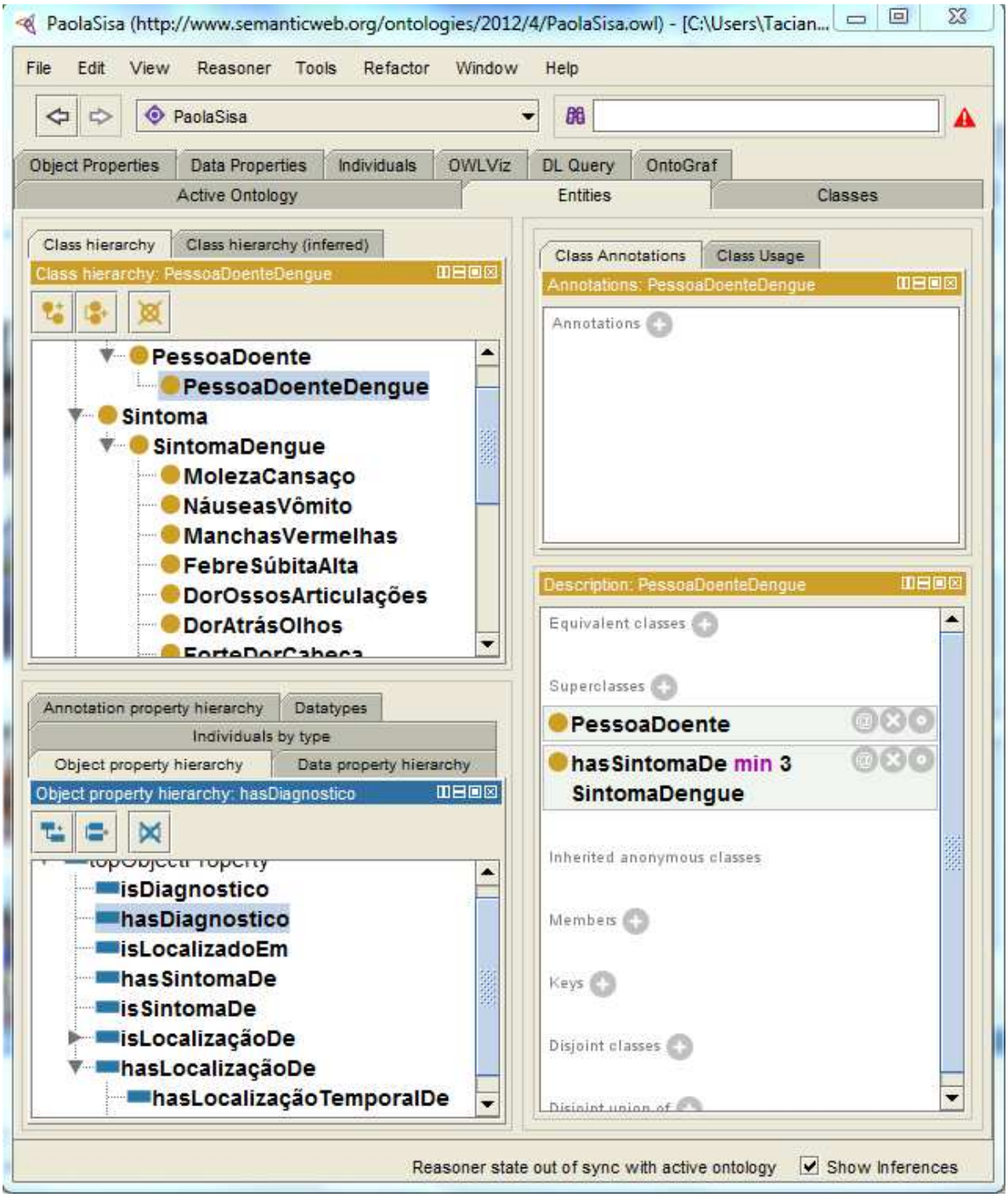

Figura 43 – Desenvolvimento de ontologias utilizando o Protégé

Após edição da ontologia, ela é salva no formato OWL e, em seguida, seu código-fonte é importado pela plataforma Paola, como definido na seção 5.1.1. A partir da importação pela plataforma Paola é possível utilizar os outros módulos para trabalhar com o contexto da aplicação de Dengue.

7.2. Definição de Provedores de Contexto

Trabalhamos, no Sisa desenvolvido com a plataforma Paola, é utilizado apenas um tipo de provedor de contexto: o *set-top-box* da TV Digital Interativa instalada nos lares da população.

O Sisa disponibiliza um enquete, através de uma aplicação que executa na TV Digital, para colher dados sobre o estado de saúde dos membros de uma família que se encontra em área de risco de contaminação de Dengue. Esta aplicação não é detalhada nesta dissertação, porém o leitor pode consultar Antunes (2011) para mais informações. A aplicação é capaz de coletar os sintomas que cada pessoa informa pelo controle remoto.

A plataforma Paola fica responsável pela tarefa de conectar o provedor de contexto e receber os dados fornecidos por ele. A aplicação Sisa é isentada de preocupar-se com esses detalhes.

Como a mesma aplicação rodará na casa de várias pessoas, a plataforma Paola pode tratar os provedores de contexto de forma idêntica. Para isso é necessário adicionar um diretório de provedores de contexto. O diretório contém o catálogo de todos os *set-top-boxes* que rodam a aplicação Sisa.

Utilizando o módulo de Gerenciamento de Provedores de Contexto é possível adicionar o diretório (informando uma URL) e selecionar os dados que eles fornecem, através da tela ilustrada na Figura 25. Por exemplo, a informação que o Sisa necessita é a identificação da pessoa e quantidade de sintomas que ela tem. Para fins de demonstração, o diretório de provedores de contexto está localizado no endereço: ‘http://paola.lariisa.gov.br/diretorios/sisa-set-top-boxes-ceara’ e eles fornecem dados contextuais denominados de ‘numeroCartaoNacionalSaude’, que identifica a pessoa no sistema e de ‘quantidadeSintomasDengueDeUmaPessoa’, de valor inteiro positivo que representa a quantidade de sintomas de Dengue que a pessoa possui.

7.3. Criação de Regras e Ações

Com base nos dados recebidos nos provedores de contexto é possível criar regras. Regras são formadas de pré-condições e ações. Neste estudo de caso é criada uma regra bem simples para realizar um diagnóstico prévio de Dengue.

As pré-condições da regra verificam se uma pessoa possui três ou mais sintomas de Dengue e se encontra em área de risco. O Quadro 18 mostra como é a sintaxe de uma regra na plataformaPaola, obedecendo a base de conhecimento criada. No exemplo é assumido que a cidade de Fortaleza é uma área de risco de contaminação de Dengue.

```
(hasSintoma min 3 SintomaDengue) and (hasLocalizaçãoEspacial in Fortaleza)
```

Quadro 18 - Pré-condições da regra do Sisa

Baseada nas pré-condições criadas, a regra pode ser satisfeita ou não, de acordo com os dados recebidos. A regra pode acionar uma ação para notificar a aplicação sensível ao contexto do desenvolvedor.

Foi criada uma ação utilizando o padrão *Observer*. Dentro do executável a ser gerado está uma classe denominada 'PaolaObservable', que herda da classe 'java.util.Observable'. A aplicação sensível ao contexto do usuário, no caso o Sisa, deve utilizar esta classe para observar seu comportamento. Baseado no nome da regra criada, a aplicação receberá as notificações.

Neste exemplo a notificação ocorrerá via Java e a aplicação Sisa, quando receber a mensagem, armazenará em banco de dados a localização da pessoa, sua identificação e o diagnóstico prévio de Dengue. Caso surjam muitos diagnósticos prévios em áreas próximas, o Sisa enviará alertas para os gestores de saúde tomarem decisões: como alocar mais médicos e disponibilizar mais medicamentos, por exemplo.

7.4. Manipulando Informações

O Joseki, servidor *web* para consultas SPARQL, é incorporado ao artefato gerado pela plataforma Paola. Portanto, está apto a receber requisições HTTP o processar as consultas na base de dados.

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT count(*)
WHERE
        { ?x integer:quantidadeSintomasDengue > 3
        }
```

Quadro 19 – Exemplo de consulta a dados em base de conhecimento

O Sisa pode consultar a base de conhecimento para leitura de dados. Por exemplo: é interessante a verificação da quantidade de diagnósticos prévios de casos de Dengue. Para isso uma consulta como a mostrada no Quadro 19 pode ser encapsulada em uma requisição HTTP. O desenvolvedor pode criar suas próprias consultas de acordo com sua base de conhecimento.

O resultado da consulta é também via *web* e em formato XML, semelhante ao mostrado no Quadro 17. O Sisa interpreta o arquivo recebido e integra os dados ao sistema.

7.5. Geração de Executável

Ao final do desenvolvimento do projeto, com as bases de conhecimentos definidas, provedores de contexto configurados, regras e suas pré-condições e ações criadas e a API de consulta definida é possível criar o executável do projeto para que ele possa ser incorporado ao Sisa.

Para utilizar os serviços oferecidos pelo pacote gerado a aplicação Sisa necessita adicionar o arquivo ‘sisa-ca.jar’ (gerado pela plataforma Paola) ao seu *classpath* e importar as bibliotecas Java em seu código-fonte, como mostrado no Quadro 20.

```
package sisa;

// importação do artefato executável gerado pela plataforma Paola
import br.gov.larissa.paola.sisa-ca;

public class Sisa {
        ...
}
```

Quadro 20 – Utilização do artefato executável no Sisa

A aplicação Sisa, com o pacote gerado pela plataforma Paola incorporado, torna-se sensível ao contexto e integrado com provedores e com regras e ações.

7.6. Considerações Finais do Capítulo

Este capítulo abordou como se dá o desenvolvimento de uma aplicação sensível ao contexto. Foi mostrado um exemplo simples da aplicação Sisa, proposta por Antunes (2011).

O ciclo completo de desenvolvimento começa na definição das ontologias, onde é modelado o contexto da aplicação. Foi utilizada, no exemplo, a ferramenta Protégé que auxiliou na edição de ontologias OWL.

Em seguida foram selecionados os provedores de contexto. Como a plataforma Paola utiliza a arquitetura Lisa como integrador de provedores, esta tarefa se torna muito simples e com um nível alto de abstração, pois a arquitetura Lisa é quem é responsável por tratar detalhes de configuração de integração de ambientes.

Com as ontologias criadas e os provedores selecionados, foi definida uma regra simples para ilustrar o funcionamento das pré-condições e ações. Uma vez que as pré-condições sejam satisfeitas, as ações são iniciadas e a aplicação sensível ao contexto notificada.

Também foi exemplificada como é realizada a consulta aos dados contextuais, através de uma linguagem de consulta simples que é executada na *web*, utilizando requisições HTTP.

Ao final foi mostrado como o projeto executável é gerado e como ele é incorporado à aplicação sensível ao contexto do desenvolvedor.

Através do estudo de caso mostrado é possível compreender, na prática, como a plataforma Paola funciona. É percebido que a utilização dela no Sisa traz grande agilidade no seu desenvolvimento, quando comparado ao trabalho de Antunes (2011).

CONSIDERAÇÕES FINAIS

No Brasil, assim como em diversos países em desenvolvimento, os recursos para a área da saúde são insuficientes para atender toda a demanda da população com qualidade. Também há necessidades na área de gestão da saúde, pois os recursos precisam ser aplicados visando aumentar a eficiência e reduzir os custos operacionais.

Neste cenário, iniciativas como as do projeto Lariisa são louváveis por ter relevâncias tecnológica e social, pois se utilizam dos recursos da informática aplicada à saúde para promover a melhoria de um serviço essencial à sociedade.

Desenvolver aplicações para o projeto Lariisa é uma tarefa muito complexa. Envolve conceitos de computação sensível ao contexto, detalhes da configuração de governança do projeto, integração de provedores de contexto, gerenciamento de conhecimento, entre outros. Tais dificuldades foram vivenciadas por Antunes (2011) em sua aplicação de controle epidemiológico da dengue.

A plataforma Paola cria mecanismos para que o desenvolvedor de aplicações do projeto Lariisa possa construir aplicações com mais facilidade. A integração dos módulos Base de Conhecimento, Provedores de Contexto, Regras & Ações, Gerenciamento de Informação e Gerador de Executável & Simulação cria uma ferramenta específica, projetada com base nas peculiaridades do projeto Lariisa, que atende as necessidades do desenvolvedor do Lariisa.

Nesta dissertação, foram aplicados conceitos de computação sensível ao contexto, ontologias e governança de saúde. No capítulo 2 foram vistos esses conceitos que serviram como base para a proposta.

Quatro ferramentas foram escolhidas e comparadas em relação às suas vantagens e desvantagens: *The Context Toolkit*, *VadeMecum*, *A Framework for Developing Mobile, Context-aware Applications* e Protégé.

A plataforma Paola se diferencia dos trabalhos mostrados no que diz respeito a ser aplicado ao projeto Lariisa e suas especificidades. Também, diferentemente dos trabalhos citados que criam aplicações sensíveis ao contexto, a plataforma Paola cria um *software* executável que é incorporado por uma aplicação para torná-la sensível ao contexto e estender os conceitos de bases de conhecimento,

provedores de contexto, regras, ações e gerenciamento de informação (consultas e manipulação de dados).

O módulo de manipulação Gerenciamento de Bases de Conhecimento auxilia o desenvolvedor a gerenciar e editar suas bases de conhecimento com mais facilidade. A plataforma Paola possui bibliotecas de *software* incorporadas que permitem a manipulação e fornecem uma API de fácil utilização.

O módulo de manipulação de Provedores de Contexto realiza, de forma simples, a integração de provedores de contexto à aplicação sensível ao contexto. O desenvolvedor é poupado de saber detalhes de configuração e arquitetura de dispositivos, pois a plataforma Paola abstrai esses detalhes com o Lisa/Lariisa.

O módulo de manipulação de Regras & Ações fornece ao desenvolvedor uma forma simples de consultar o contexto, detectar mudanças e reagir a estas. Através de uma interface simples, intuitiva e utilizando uma linguagem fácil, a tarefa de criar uma regra e uma ação se torna bastante rápida.

O módulo de gerenciamento de Informação fornece ao desenvolvedor e aplicações uma facilidade para que estes possam acessar e manipular os dados contextuais da aplicação.

O módulo Gerador de Executável & Simulação é responsável por gerar um *software* executável e pode ser incorporado à aplicação sensível ao contexto ou pode ser simulado.

Com todas essas características, a plataforma Paola tem grande potencial de ajudar o desenvolvedor de aplicações para o projeto Lariisa. Foram desenvolvidos alguns módulos de forma isolada e os resultados foram satisfatórios em relação ao potencial da agilidade que o desenvolvedor terá. É um desafio implementar a plataforma Paola completamente e integrar todos os seus módulos. Porém o foco principal desta dissertação foi especificar e implementar algumas partes dos módulos como prova de conceito.

Como estudo de caso, parte da aplicação Sisa (ANTUNES, 2011) foi desenvolvida utilizando os conceitos da plataforma Paola. Esta experiência foi narrada no capítulo 7. Foi notado que para desenvolver uma aplicação bem simples com a plataforma Paola a complexidade foi diminuída em relação às enfrentadas por Antunes (2011).

A plataforma Paola trata de aspectos computacionais do desenvolvimento de aplicações para o projeto Lariisa. Trata também de semântica e para isso utiliza

ontologias para representação do conhecimento. Porém, o nível de abstração que se tem hoje com a plataforma Paola e o projeto Lariisa ainda não contemplam a criação de aplicação por leigos da informática que, no caso da área de governança de saúde, podem ser os gestores, dirigentes, administradores, prefeitos, governadores, entre outros.

O projeto Lariisa está evoluindo no sentido de aumentar o nível de abstração na criação e execução de suas aplicações e a plataforma Paola contribuiu muito nessa meta.

Conforme visto na seção 4.8, Bastos (2012) propôs, na área de gestão de conhecimento em saúde, a adaptação de conhecimento no projeto Lariisa. Essa proposta definiu caminhos de conhecimento que mostram como fluem os dados entre as áreas de governança (Gerenciamento de Conhecimento, Normatização Sistêmica, Clínico-Epidemiológica, Administrativo e Gerenciamento Compartilhado).

Em um nível de abstração das informações do projeto Lariisa mais elevado, Bastos (2012) especificou um trabalho que complementa a visão da plataforma Paola, à medida que empresta uma visão do modelo de Graham aplicada ao projeto Lariisa, ou seja, como o conhecimento empírico ou tácito pode efetivamente chegar ao usuário segundo o modelo KTA.

A Figura 44, a seguir, ilustra que os Mapas Conceituais (BASTOS, 2012) e a plataforma Paola trabalham de forma independente para definir o funcionamento das aplicações. As duas soluções são distintas, mas podem se complementar mutuamente.

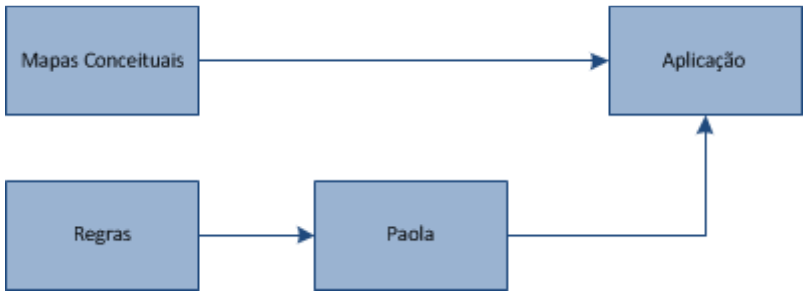

Figura 44 – Plataforma Paola e Mapas Conceituais (BASTOS, 2012) independentes

Com o objetivo de aumentar o nível de abstração para fornecer um sistema para ser utilizado por um número maior de usuários, independente de conhecimentos técnicos, a plataforma Paola, os Caminhos do Conhecimento e os

Mapas Conceituais (BASTOS, 2012) podem se unir para a definição de regras de negócio. A Figura 45 ilustra como seria o fluxo com esta integração, que é proposta como trabalho futuro.

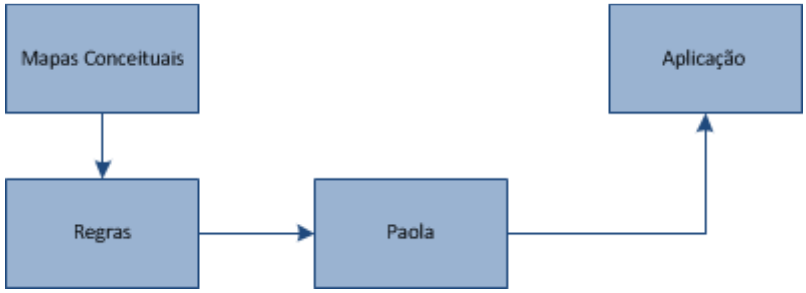

Figura 45 – Mapas Conceituais e integrados às regras e à plataforma Paola

Portanto, a plataforma Paola contribui para o projeto Lariisa no tocante ao desenvolvimento e adaptação de aplicações sensíveis ao contexto, facilitando o processo de construção de aplicações para a área de governança de saúde. Contribui também para o aumento no nível de abstração do domínio do projeto Lariisa e da integração de seus diversos projetos.

Ficam propostos como trabalhos futuros:

- implementar completamente o *software* da Plataforma Paola;
- integrar todos o módulos da plataforma Paola;
- oferecer funções para a edição de ontologias na plataforma Paola;
- desenvolver outras aplicações utilizando a plataforma Paola;
- incrementar o nível de abstração do projeto Lariisa utilizando os conceitos de mapas conceituais e caminhos do conhecimento.

REFERÊNCIAS

ABOWD, G. D.; MYNATT, E. D. Charting Past, Present, and Future Research in Ubiquitous Computing. *ACM Transactions on Computer-Human Interaction*, New York, v. 7, p. 29-58, 2000.

ANTUNES, F. *SISAGE: Um Componente do Lariisa de Gestão e Vigilância Epidemiológica em instâncias de agravo de Degue no Estado do Ceará*. 2011. Dissertação (Mestrado em Computação Aplicada) – Programa de Pós-Graduação em Ciências da Computação, Universidade Estadual do Ceará, Fortaleza, 2011.

BASTOS, A. V. M. C. Análise do Processo de Apadtação do Conhecimento em Saúde: *Cenários de Aplicação para a plataforma Lariisa*. 2012. Dissertação (Mestrado em Saúde da Família) – Faculdade de Medicina, Universidade Federal do Ceará, Sobral, 2012.

BERNERS-LEE T; HENDLER J; LASSILA O. The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 2001.

BIEGEL G.; CAHILL V. A framework for development mobile, context-aware applications. In: Second IEEE Annual Conference on Pervasive Computing and Communications, 2004. Anais… Dublin, p. 361-365, 2004.

BREITMAN, K. K. *Web Semântica*: a internet do futuro, Rio de Janeiro: LTC, v. 1, 2005. 212 p.

CONNOLLY, D.; HARMELEN F. V.; HORROCKS I.; MCGUINNESS D. L.; PATEL-SCHNEIDER P. F.; STEIN L. A. DAML+OIL (March 2001). *Reference Description, World Wide Web Consortium,* 2001.

FROTA, J. B. B. *Proposta de Solução de Integração de Provedores de Contexto ao Sistema Lariisa*, 2011. Dissertação (Mestrado em Computação Aplicada) – Programa de Pós-Graduação em Ciências da Computação, Universidade Estadual do Ceará, Fortaleza, 2011.

FIGUEIREDO, H. F. *Uma Infraestrutura de Suporte a Aplicações Cientes de Contexto com o Enfoque no Usuário Final*, 2009. Dissertação (Mestrado em Ciência da Computação), Universidade Federal de Campina Grande, Campina Grande, 2009.

GARCIA, L. M. L. S.; ANTUNES, F.; SANTOS, M. E. S. Utilização de recursos da Web Semântica na Construção de um Ambiente Web para Publicação Científica Indexada e Recuperada por Ontologias. In: INFOBRASIL TI & TELECOM 2010, 2010. *Anais*... Fortaleza, 2010.

GOOGLE ACADÊMICO. Disponível em: http://scholar.google.com.br/. Acesso em 19 de jan. 2012.

GRAHAN, I. D.; LOGAN, J.; HARRISON, M. B.; STRAUS, S. E.; TETROE, J.; CASWELL, W.; ROBINSON, N. Lost in Knowledge Translation: Time for a Map? *Jornal of Cantinuing Education in the Heath Professions*, X, v. 26, p. 13-24, 2006.

GRUNINGER, M. Designing and Evaluating Generic Ontologies. In: 12TH EUROPEAN CONFERENCE OF ARTIFICIAL INTELLIGENCE, 1996. *Anais*… Budapest, 1996.

GUARINO, N. *Formal Ontology in Information Systems*. Amsterdan: IOS Press, 1998. 341 p.

HORROCKS, I. DAML+OIL: a Description Logic for the Semantic Web. *Bulletin of the Technical Committee on Data Engineering*, v. 25, n. 1, p. 6-9, 2002.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. Um Panorama da Saúde no Brasil: Acesso e Utilização dos Serviços, Condições, de Saúde e Fatores de Risco e Proteção à Saúde, 2008. Disponível em: http://biblioteca.ibge.gov.br/visualizacao/monografias/GEBIS%20-%20RJ/panorama.pdf. Acesso em 19 de jan. 2012.

LOPES, J. L. B. Sensibilidade ao Contexto na Comutação Pervasiva. Disponível em: http://ppginf.ucpel.tche.br/TI-arquivos/2006/JoaoLopes/PPGINF-UCPel-TI-2006-2-06.pdf. Acesso em 19 de jan. 2012.

MINISTÉRIO DA SAÚDE DO BRASIL. *Entendendo o SUS. Brasília*, 2007. Disponível em: http://portal.saude.gov.br/portal/arquivos/pdf/cartilha_entendendo_o_sus_2007.pdf. Acesso em 19 de jan. 2012.

MOURA, M. A. Informação, Ferramentas Ontológicas e Redes Sociais Ad Hoc: A Interoperabilidade na construção de tesauros e ontologias. *Inf. & Soc.*, João Pessoa, v. 19, p. 59-73, 2009.

OLIVEIRA, M; HAIRON, C.; ANDRADE, O.; MOURA, R.; SICOTTE, C.; DENIS, J-L.; FERNANDES, S.; GENSEL, J.; BRINGEL, J.; MARTIN, H. A context-aware framework for health care governance decision-making systems: A model based on the Brazilian Digital TV. In: 2010 IEEE INTERNATIONAL SYMPOSIUM ON A WORLD OF WIRELESS MOBILE AND MULTIMEDIA NETWORKS, 2010, Montreal. *Anais*… Montreal, 2010, p. 1-6.

PORTAL DA SAÚDE. Sistema Único de Saúde. Disponível em http://portal.saude.gov.br/portal/saude/visualizar_texto.cfm?idtxt=24627. Acesso em 19 de jan. 2012.

SANTOS, M. E. S. Diga Saúde: *Uma Proposta de Sistema de Apoio a Serviços de Atendimento Domiciliar de Saúde Baseado no Modelo de TV Digital Interativa*. 2011. Dissertação (Mestrado em Computação Aplicada) – Programa de Pós-Graduação em Ciências da Computação, Universidade Estadual do Ceará, Fortaleza, 2011.

SALBER D.; DEY A. K.; ABOWD G. D. The Context Toolkit: aiding the development of context-enabled applications. *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit,* New York, ACM, p. 434-441, 1999.

SCHILIT, B.; ADAMS, N.; WANT, R. Context-Aware Computing Applications. In: FIRST WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS, 1994, Santa Cruz. *Anais*… Santa Cruz, 1994. p. 85-90.

SCHILIT, B.; THEIMER, M. Disseminating active map information to mobile hosts. *IEEE Network*, v. 8, p. 22-32, 1994.

SCHIMIDT, A. *Ubiquitous Computing - Computing in Context.* 2002. Tese (Ph.D. in Computer Science) – Computer Department, Lancaster University, Lancaster, 2002.

STANFORD UNIVERSITY SCHOOL OF MEDICINE. *Protégé project,* 2012. Disponível em: http://protege.stanford.edu/. Acesso em 19 de jan. 2012.

STRANG, T.; LINNHOFF-POPIEN, C. A Context Modeling Survey. In: WORKSHOP O GRAPHICAL MODELS, 2004. *Anais*… 2004. p. 1-8.

VIEIRA I. Programa Saúde da Família chega à metade das casas brasileiras, constata IBGE. *Agência Brasil*, 31 de mar. 2010. Disponível em: http://oglobo.globo.com/politica/programa-saude-da-familia-chega-metade-das-casas-brasileiras-constata-ibge-3030828. Acesso em 19 jan. 2012.

WORLD WIDE WEB CONSORTIUM. *W3C Semantic Web Frequentily Asked Questions.* Cambridge, 2009.

APÊNDICE A – Casos de Uso Expandidos

Código	UC1-BASE
Nome do Caso de Uso	Pesquisar Ontologias
Descrição	Permite que o ator pesquise ontologias em bases de ontologias acopladas ao sistema.
Atores	Desenvolvedor
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo Base de Conhecimento.
Fluxo de Eventos	-
Fluxo Principal	1. Ator digita uma palavra-chave; 2. Ator clica no botão pesquisar;
Fluxos Alternativos	-
Pós-Condições	1. O sistema lista o resultado da pesquisa.
Pontos de Extensão	-
Casos de Uso Incluídos	-

Código	UC2-BASE
Nome do Caso de Uso	Importar Ontologias
Descrição	Permite que o ator importe uma ontologia a partir dos resultados de uma pequisa realizada.
Atores	Desenvolvedor
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo Base de Conhecimento; 3. Ator realizou busca por ontologias (UC1-BASE) 4. Ator selecionou ontologias do resultado (UC3-BASE)
Fluxo de Eventos	-
Fluxo Principal	1. Ator clica no botão importar ontologia.
Fluxos Alternativos	-
Pós-Condições	1. A ontologia é importada pela Paola.
Pontos de Extensão	-
Casos de Uso Incluídos	UC1-BASE, UC3-BASE

Código	UC3-BASE
Nome do Caso de Uso	Selecionar Ontologias para a Importação
Descrição	O ator, após pesquiser por ontologias, seleciona um número X de resultados.
Atores	Desenvolvedor
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo Base de Conhecimento; 3. Ator realizou busca por ontologias (UC1-BASE)
Fluxo de Eventos	-
Fluxo Principal	1. Ator seleciona ontologias do resultado da busca;
Fluxos Alternativos	-
Pós-Condições	1. A(s) ontologia(s) selecionadas são destacadas no resultado de busca.
Pontos de Extensão	-
Casos de Uso Incluídos	UC1-BASE

Código	UC4-BASE
Nome do Caso de Uso	Selecionar Ontologia
Descrição	O ator seleciona uma ontologia na lista de Ontologias Utilizadas.
Atores	Desenvolvedor
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo Base de Conhecimento; 3. O ator possui alguma ontologia na lista de Ontologia Utilizadas - através da importação (UC2-BASE) ou da criação (UC7-BASE)
Fluxo de Eventos	-
Fluxo Principal	1. Ator seleciona uma ontologia na lista de Ontologias Utilizadas.
Fluxos Alternativos	-
Pós-Condições	1. A ontologia selecionada é destacada entre as demais.
Pontos de Extensão	-
Casos de Uso Incluídos	UC2BASE, UC7-BASE

Código	UC5-BASE
Nome do Caso de Uso	Editar Ontologia
Descrição	O ator edita o código-fonte de uma ontologia.
Atores	Desenvolvedor
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo Base de Conhecimento; 3. Ator seleciona uma ontologia (UC4-BASE)
Fluxo de Eventos	-
Fluxo Principal	1. Ator clica no botão editar.
Fluxos Alternativos	-
Pós-Condições	1. O código fonte da ontologia é exibido na ferramenta de edição.
Pontos de Extensão	-
Casos de Uso Incluídos	UC4-BASE

Código	UC6-BASE
Nome do Caso de Uso	Remover Ontologia
Descrição	O ator remove uma ontologia da lista de ontologias utilizadas
Atores	Desenvolvedor
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo Base de Conhecimento; 3. O ator possui alguma ontologia na lista de Ontologia Utilizadas - através da importação (UC2-BASE) ou da criação (UC7-BASE) 4. Ator seleciona uma ontologia (UC4-BASE)
Fluxo de Eventos	-
Fluxo Principal	1. Ator clica no botão excluir
Fluxos Alternativos	-
Pós-Condições	1. A ontologia é removida do sistema.
Pontos de Extensão	-
Casos de Uso Incluídos	UC2-BASE, UC7-BASE, UC4-BASE

Código	UC7-BASE
Nome do Caso de Uso	Criar Ontologia
Descrição	O ator cria uma nova ontologia no sistema.
Atores	Desenvolvedor
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo Base de Conhecimento;

Fluxo de Eventos	
Fluxo Principal	1. Ator clica no botão criar ontologia; 2. Ator nomeia a nova ontologia; 3. Ator ciar o código fonte da ontologia.
Fluxos Alternativos	-
Pós-Condições	1. A ontologia é criada no sistema.
Pontos de Extensão	-
Casos de Uso Incluídos	-

Código	UC8-BASE
Nome do Caso de Uso	Adicionar base de ontologias
Descrição	O ator adiciona uma base de ontologias ao sistema.
Atores	Desenvolvedor
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo Base de Conhecimento.
Fluxo de Eventos	-
Fluxo Principal	1. Ator clica no botão adicionar nova base de ontologias; 2. Ator informa o nome da base de ontologias e seu endereço na internet; 3. Ator salva a nova base de ontologias na plataforma.
Fluxos Alternativos	-
Pós-Condições	1. A base de dados de ontologias é adicionada ao sistema.
Pontos de Extensão	-
Casos de Uso Incluídos	-

Código	UC9-BASE
Nome do Caso de Uso	Selecionar base de ontologias
Descrição	O ator seleciona uma base de ontologias no sistema.
Atores	Desenvolvedor
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo Base de Conhecimento.
Fluxo de Eventos	-
Fluxo Principal	1. Ator abre a seção de bases de ontologias; 2. Ator seleciona uma base de ontologias.
Fluxos Alternativos	-
Pós-Condições	1. Uma base de ontologias é selecionada.
Pontos de Extensão	-
Casos de Uso Incluídos	-

Código	UC10-BASE
Nome do Caso de Uso	Remover base de ontologias
Descrição	O ator remova uma base de ontologias do sistema.
Atores	Desenvolvedor
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo Base de Conhecimento.
Fluxo de Eventos	-
Fluxo Principal	1. Ator seleciona uma base de ontologias (UC9-BASE); 2. Ator remove a base de ontologias.
Fluxos Alternativos	-
Pós-Condições	1. A base de ontologias é removida da plataforma.
Pontos de Extensão	-
Casos de Uso Incluídos	UC9-BASE

Código	UC10-REGRA-AÇÃO

Nome do Caso de Uso	Selecionar regra
Descrição	O ator seleciona uma regra no sistema.
Atores	Desenvolvedor
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo de Regras e Ações.
Fluxo de Eventos	-
Fluxo Principal	1. Ator acessa a lista de regras criadas; 2. Ator seleciona uma regra da lista de regras.
Fluxos Alternativos	-
Pós-Condições	Uma regra é selecionada na plataforma.
Pontos de Extensão	-
Casos de Uso Incluídos	-

Código	UC11-REGRA-AÇÃO
Nome do Caso de Uso	Definir pré-condições associadas a uma regra.
Descrição	O ator define pré-condições e as associa a uma regra recém-criada.
Atores	Desenvolvedor.
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo de Regras e Ações.
Fluxo de Eventos	-
Fluxo Principal	1. Ator seleciona uma regra no sistema (UC10-REGRA-AÇÃO) 2. Ator clica no botão adiciona pré-condição à regra; 3. Ator informa detalhes da pré-condição; 4. Ator salva a pré-condição.
Fluxos Alternativos	-
Pós-Condições	Uma pré-condição é adicionada a uma regra.
Pontos de Extensão	-
Casos de Uso Incluídos	UC10-REGRA-AÇÃO

Código	UC12-REGRA-AÇÃO
Nome do Caso de Uso	Definir ações associadas a uma regra.
Descrição	O ator define ações e as associa a uma regra recém-criada.
Atores	Desenvolvedor.
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo de Regras e Ações.
Fluxo de Eventos	-
Fluxo Principal	1. Ator seleciona uma regra no sistema (UC10-REGRA-AÇÃO); 2. Ator clica no botão adiciona ação à regra; 3. Ator informa como a ação se comportará; 4. Ator salva a ação.
Fluxos Alternativos	-
Pós-Condições	Uma ação é adicionada a uma regra.
Pontos de Extensão	-
Casos de Uso Incluídos	UC10-REGRA-AÇÃO

Código	UC13-REGRA-AÇÃO
Nome do Caso de Uso	Criar Regra
Descrição	O ator cria uma regra na plataforma.
Atores	Desenvolvedor
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo de Regras e Ações.
Fluxo de Eventos	-
Fluxo Principal	1. Ator clica no botão criar regra;

	2. Ator preenche dados da regra.
Fluxos Alternativos	1. Ator define pré-condições da regra (UC11-REGRA-AÇÃO); 2. Ator define ações da regra (UC12-REGRA-AÇÃO).
Pós-Condições	Uma regra é criada na plataforma.
Pontos de Extensão	-
Casos de Uso Incluídos	UC11-REGRA-AÇÃO, UC12-REGRA-AÇÃO

Código	UC14-REGRA-AÇÃO
Nome do Caso de Uso	Editar Regra
Descrição	O ator edita uma regra na plataforma.
Atores	Desenvolvedor
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo Regras e Ações.
Fluxo de Eventos	-
Fluxo Principal	1. Ator seleciona uma regra (UC10-REGRA-AÇÃO) 2. Ator edita dados da regra.
Fluxos Alternativos	1. Ator define pré-condições da regra (UC11-REGRA-AÇÃO); 2. Ator define ações da regra (UC12-REGRA-AÇÃO).
Pós-Condições	Uma regra é alterada na plataforma.
Pontos de Extensão	-
Casos de Uso Incluídos	UC10-REGRA-AÇÃO, UC11-REGRA-AÇÃO, UC12-REGRA-AÇÃO

Código	UC15-REGRA-AÇÃO
Nome do Caso de Uso	Remover Regra
Descrição	O ator remove uma regra da plataforma.
Atores	Desenvolvedor
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo Regras e Ações.
Fluxo de Eventos	-
Fluxo Principal	1. Ator seleciona uma regra (UC10-REGRA-AÇÃO); 2. Ator clica no botão remover regra.
Fluxos Alternativos	-
Pós-Condições	Uma regra, assim como as pré-condições e ações associadas, é excluída da plataforma.
Pontos de Extensão	-
Casos de Uso Incluídos	UC10-REGRA-AÇÃO

Código	UC16-PROVEDOR
Nome do Caso de Uso	Informar endereço
Descrição	O ator informa o endereço de um provedor de contexto acessível por rede.
Atores	Desenvolvedor
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo Provedores de Contexto.
Fluxo de Eventos	-
Fluxo Principal	1. Ator informa um endereço IP ou um nome de domínio de rede de um provedor de contexto acessível por rede.
Fluxos Alternativos	-
Pós-Condições	O endereço de um provedor de contexto é informado.
Pontos de Extensão	-
Casos de Uso Incluídos	-

Código	UC17-PROVEDOR
Nome do Caso de Uso	Adicionar Provedor de Contexto

Descrição	O ator adiciona um provedor de contexto à plataforma.
Atores	Desenvolvedor
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo Provedores de Contexto.
Fluxo de Eventos	-
Fluxo Principal	1. Ator clica no botão adicionar provedor de contexto; 2. Ator informa o endereço do provedor de contexto (UC16-PROVEDOR); 3. Ator salva o provedor de contexto.
Fluxos Alternativos	-
Pós-Condições	O provedor de contexto é adicionado à plataforma.
Pontos de Extensão	-
Casos de Uso Incluídos	UC16-PROVEDOR

Código	UC18-PROVEDOR
Nome do Caso de Uso	Adicionar Diretório Provedor de Contexto
Descrição	O ator adiciona um diretório de provedores de contexto à plataforma.
Atores	Desenvolvedor
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo Provedores de Contexto.
Fluxo de Eventos	-
Fluxo Principal	1. Ator clica no botão adicionar diretório de provedores de contexto; 2. Ator informa o endereço do diretório de provedores de contexto (UC16-PROVEDOR); 3. Ator salva o diretório de provedores de contexto.
Fluxos Alternativos	-
Pós-Condições	O diretório de provedores de contexto é adicionado à plataforma.
Pontos de Extensão	-
Casos de Uso Incluídos	UC16-PROVEDOR

Código	UC19-PROVEDOR
Nome do Caso de Uso	Selecionar provedor de contexto ou diretório de provedores de contexto.
Descrição	O ator seleciona um provedor de contexto ou um diretório de provedores de contexto.
Atores	Desenvolvedor
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo Provedores de Contexto.
Fluxo de Eventos	-
Fluxo Principal	1. Ator acessa a lista de provedores de contexto e de diretório de provedores de contexto; 2. Ator seleciona um provedor de contexto ou um diretório de provedores de contexto.
Fluxos Alternativos	-
Pós-Condições	Um provedor de contexto ou diretório de provedores de contexto é selecionado.
Pontos de Extensão	-
Casos de Uso Incluídos	-

Código	UC20-PROVEDOR
Nome do Caso de Uso	Remover provedor de contexto ou diretório de provedores de contexto.
Descrição	O ator remove da plataforma um provedor de contexto ou um diretório de provedores de contexto.
Atores	Desenvolvedor

Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo Provedores de Contexto.
Fluxo de Eventos	-
Fluxo Principal	1. Ator seleciona um provedor de contexto ou um diretório de provedores de contexto (UC19-PROVEDOR) 2. Ator clica no botão remover provedor de contexto ou diretório de provedores de contexto
Fluxos Alternativos	-
Pós-Condições	Um provedor de contexto ou diretório de provedor de contexto é removido da plataforma.
Pontos de Extensão	-
Casos de Uso Incluídos	UC19-PROVEDOR

Código	UC21-INFO
Nome do Caso de Uso	Consultar dados contextuas
Descrição	O ator consulta dados contextuais na base de conhecimentos.
Atores	Desenvolvedor, Aplicação sensível ao contexto
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo de Gerenciamento da Informação.
Fluxo de Eventos	-
Fluxo Principal	1. Ator cria uma consulta utilizando uma linguagem pré-definida pela plataforma; 2. Ator executa a consulta; 3. Ator recebe o resultado da consulta.
Fluxos Alternativos	-
Pós-Condições	Dados são consultados da base de dados contextual.
Pontos de Extensão	-
Casos de Uso Incluídos	-

Código	UC22-INFO
Nome do Caso de Uso	Alterar dados contextuas
Descrição	O ator altera dados contextuais na base de conhecimentos.
Atores	Desenvolvedor, Aplicação sensível ao contexto
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo de Gerenciamento da Informação.
Fluxo de Eventos	-
Fluxo Principal	1. Ator cria uma consulta de alteração utilizando uma linguagem pré-definida pela plataforma; 2. Ator executa a consulta; 3. Ator recebe o resultado da consulta.
Fluxos Alternativos	-
Pós-Condições	Dados são alterados da base de dados contextual.
Pontos de Extensão	-
Casos de Uso Incluídos	-

Código	UC23-INFO
Nome do Caso de Uso	Remover dados contextuas
Descrição	O ator remove dados contextuais na base de conhecimentos.
Atores	Desenvolvedor, Aplicação sensível ao contexto
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo de Gerenciamento da Informação.
Fluxo de Eventos	-
Fluxo Principal	1. Ator cria uma consulta utilizando uma linguagem pré-definida

	pela plataforma; 2. Ator executa a consulta; 3. Ator recebe o resultado da consulta.
Fluxos Alternativos	-
Pós-Condições	Dados são removidos da base de dados contextual.
Pontos de Extensão	-
Casos de Uso Incluídos	-

Código	UC24-INFO
Nome do Caso de Uso	Paola armazena dados contextuais.
Descrição	O ator recebe dados contextuais para serem armazenados na base de dados contextual.
Atores	Paola
Prioridade	Normal
Pré-Condições	-
Fluxo de Eventos	-
Fluxo Principal	1. O ator recebe informações contextuais provindas de provedores de contexto; 2. O ator armazena informações contextuais.
Fluxos Alternativos	-
Pós-Condições	Dados são armazenados na base de dados contextuais.
Pontos de Extensão	-
Casos de Uso Incluídos	-

Código	UC25-INFO
Nome do Caso de Uso	Provedor de contexto envia dados contextuais.
Descrição	O ator captura e envia dados contextuais para a plataforma armazenar na base de dados contextual.
Atores	Provedor de contexto
Prioridade	Normal
Pré-Condições	-
Fluxo de Eventos	-
Fluxo Principal	1. Ator envia dados contextuais para a plataforma de gerenciamento de informação; 2. A Paola armazena os dados contextuais fornecidos pelo provedor de contexto (UC24-INFO).
Fluxos Alternativos	-
Pós-Condições	Dados são enviados pelo provedor de contexto.
Pontos de Extensão	-
Casos de Uso Incluídos	UC24-INFO

Código	UC26-EXE
Nome do Caso de Uso	Gerar artefato executável.
Descrição	O ator gera um artefato executável do projeto que foi desenvolvido.
Atores	Desenvolvedor
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa o módulo de Geração de Artefato Executável & Simulação.
Fluxo de Eventos	-
Fluxo Principal	1. Ator clica no botão gerar artefato executável.
Fluxos Alternativos	-
Pós-Condições	O artefato executável é gerado pela plataforma.
Pontos de Extensão	-
Casos de Uso Incluídos	-

Código	UC27-EXE
Nome do Caso de Uso	Simular artefato executável.
Descrição	O ator simula um artefato executável na plataforma.
Atores	Desenvolvedor
Prioridade	Normal
Pré-Condições	1. Ator está com a aplicação aberta; 2. Ator acessa a módulo de Geração de Artefato Executável & Simulação.
Fluxo de Eventos	-
Fluxo Principal	1. Ator gera um artefato executável de um projeto desenvolvido (UC26-EXE) 2. Ator simula o artefato gerado na plataforma.
Fluxos Alternativos	-
Pós-Condições	A simulação do artefato executável é realizada.
Pontos de Extensão	-
Casos de Uso Incluídos	UC26-INFO

Código	UC28-INFO
Nome do Caso de Uso	Utilizar artefato executável.
Descrição	Uma aplicação sensível ao contexto utiliza o artefato executável.
Atores	Aplicação sensível ao contexto.
Prioridade	Normal.
Pré-Condições	-
Fluxo de Eventos	-
Fluxo Principal	1. Um artefato executável é gerado (UC26-EXE) 2. Aplicação utiliza o artefato executável.
Fluxos Alternativos	-
Pós-Condições	A aplicação utiliza o artefato executável.
Pontos de Extensão	-
Casos de Uso Incluídos	UC26-EXE