**INSTITUTO DE INFORMÁTICA**
Universidade Federal de Goiás

# Padrões, Ferramentas e Boas Práticas no desenvolvimento de software para Web Semântica

**Lucas Felipe Moreira Silva**
lucassilva@inf.ufg.br

2019

INSTITUTO DE INFORMÁTICA
UFG

UFG
UNIVERSIDADE FEDERAL DE GOIÁS

# Agenda

- Fundamentação Teórica
  a. Web Semântica, Arquitetura da Web Semântica, Padrões URI/IRI, Tríade XML, XML e Web Semântica, Padrão RDF, SPARQL, Linguagens de ontologia RDFs e OWL
- Boas Práticas
  a. Gerenciamento de IRIs, Especificação de unidades de medida, representação de relacionamento n-ários

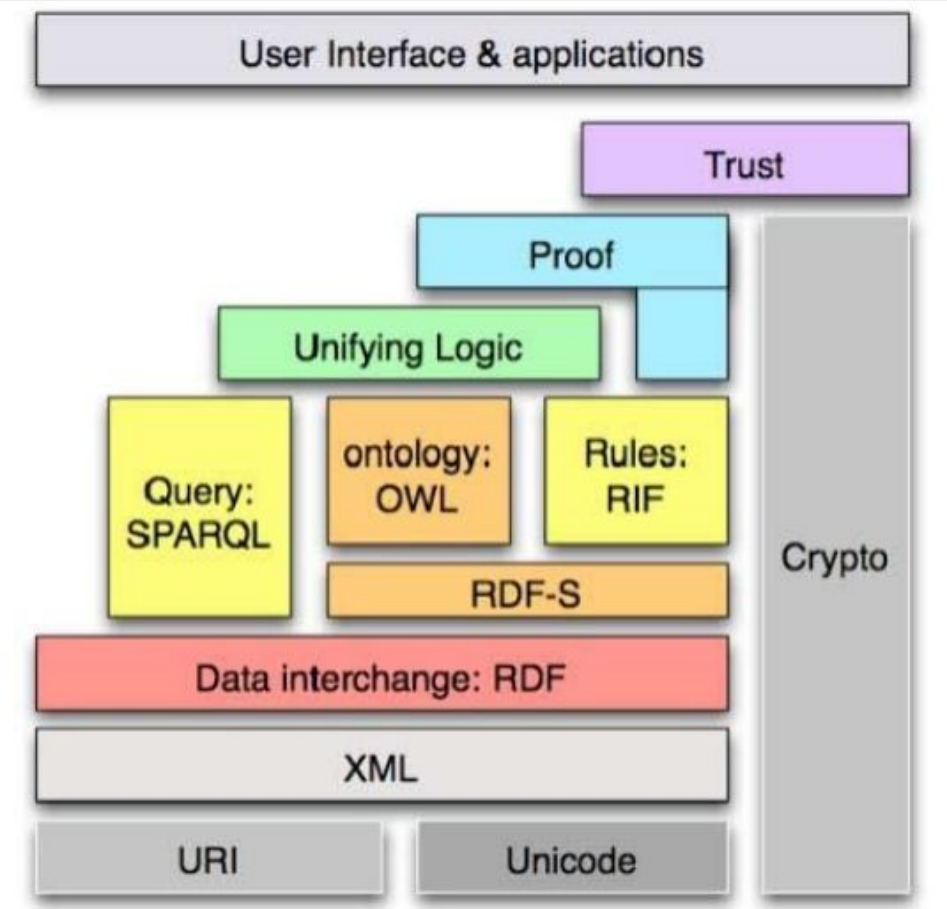
# Fundamentação Teórica

# Web Semântica

- Representação de dados em formatos adequados.
  a. Processamento, Integração e Raciocínio automatizado
  b. Especificações foram criadas para a padronização
- Atualmente as aplicações incluem serviços de web semântica
- Web de dados descritos e interligados de maneira a se estabelecer um contexto ou semântica que adere a uma linguagem e regras gramaticais bem definidas

# Arquitetura

Arquitetura em camadas da Web Semântica

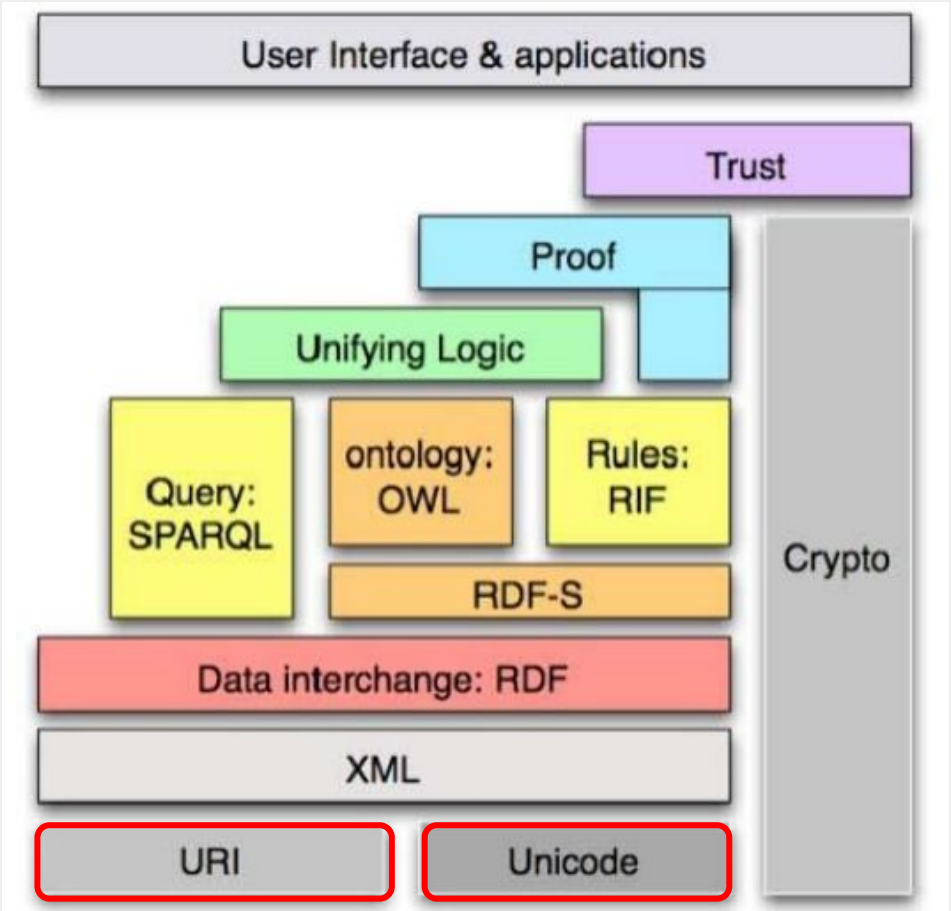


# Arquitetura da Web Semântica

- Representação em camadas.
  a. Serviços básicos nas camadas inferiores
  b. Cada camada só pode requisitar serviços da camada imediatamente abaixo

# URI
# e Unicode

Arquitetura em camadas da Web Semântica



# Unicode

- Padrão para representação de caracteres
- Fornece um código único para cada caractere, facilitando a compreensão por máquinas
- As cadeias de caracteres devem ter o mesmo padrão para que possam ser manipuladas de forma consistente pelas aplicações.

# Padrão URI / IRI

- Uniform Resource Identifier
- Cadeia de caracteres com sintaxe particular
- Identificador único para um recurso na Web
  a. Recurso: porção de conteúdo; uma página de texto, um clipe de vídeo ou de áudio, um programa, ou uma imagem.
- Atualmente, as URIs são definidas como IRIS
  a. Internationalized Resource Identifier
  b. Fornece codificação universal para nome e localização de recursos

# Sintaxe IRI

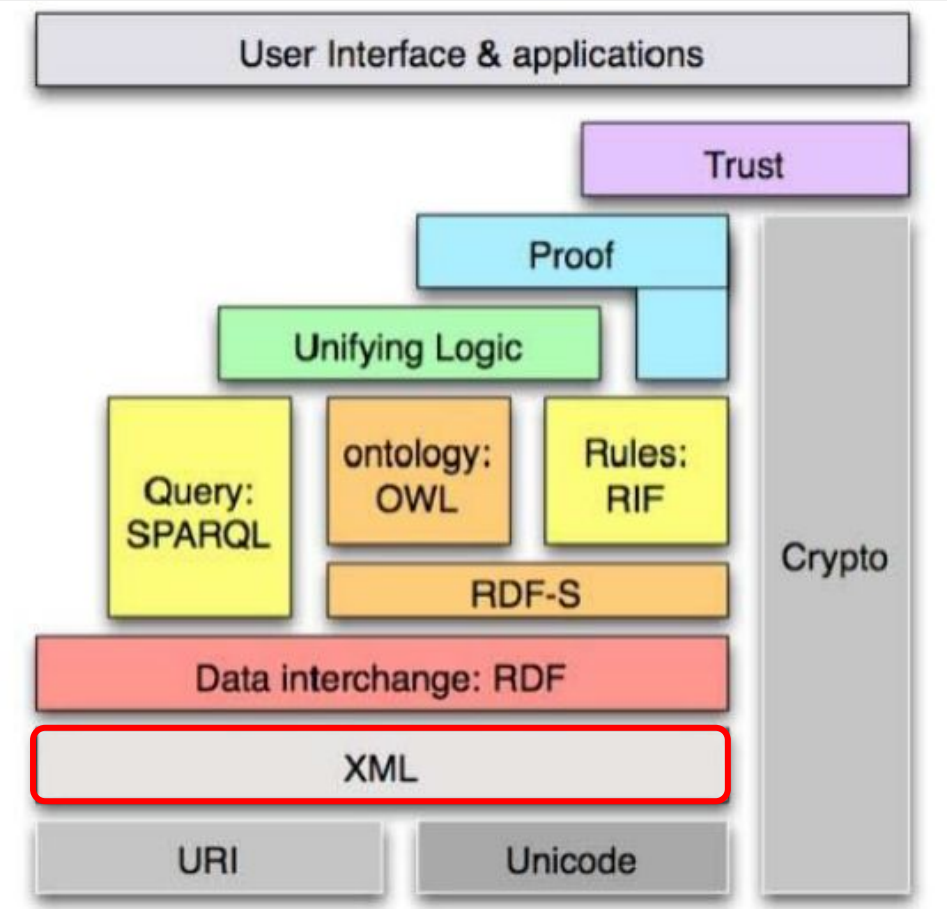
- Nome do protocolo ou mecanismo de acesso “://”
- Domínio ou autoridade sobre o recurso
- Caminho de acesso ao recurso
- Recurso em si (Pode haver fragmento do recurso)

# IRI Absoluta vs. Relativa

- Absolutas
  a. Identificações auto-contidas
  b. Independentes de contexto para resolução
- Relativas
  a. Identificações dependem do contexto em que estão inseridas

# URI
# e Unicode

Arquitetura em camadas da Web Semântica



# XML

- eXtensible Markup Language
- Linguagem de marcação extensível e flexível
- Documento representado como uma árvore de elementos
- Documento obedece regras de sintaxe

# Tríade XML

- XML - Linguagem de Marcação
- Namespace - Espaço de nomes
- XMLS - Especificação do Esquema

# XML - Regras da Sintaxe

1. Árvore - Possui raiz única; Sem Ciclos
2. Nós tem somente um único pai (a não ser a raiz)
3. Elementos devem ter uma *tag* de fechamento
4. *Tags* dos elementos são sensíveis à caixa
5. Ordem do aninhamento dos elementos importa
6. Ordem dos atributos não importa
7. Valores dos atributos devem estar entre aspas

# Namespace

- A análise do documento para se há qualquer erro na sintaxe
- *Parser* - Software que processa conteúdo XML
  a. Simples e rápido
- Para a Web semântica, XML fornece regras para construir outras especificações arquitetura
  a. RDF, RDFS, OWL, etc.
- Flexibilidade pode gerar conflito de nomes entre especificações
  a. Resolvido com o uso de namespaces

# XMLS - Esquema XML

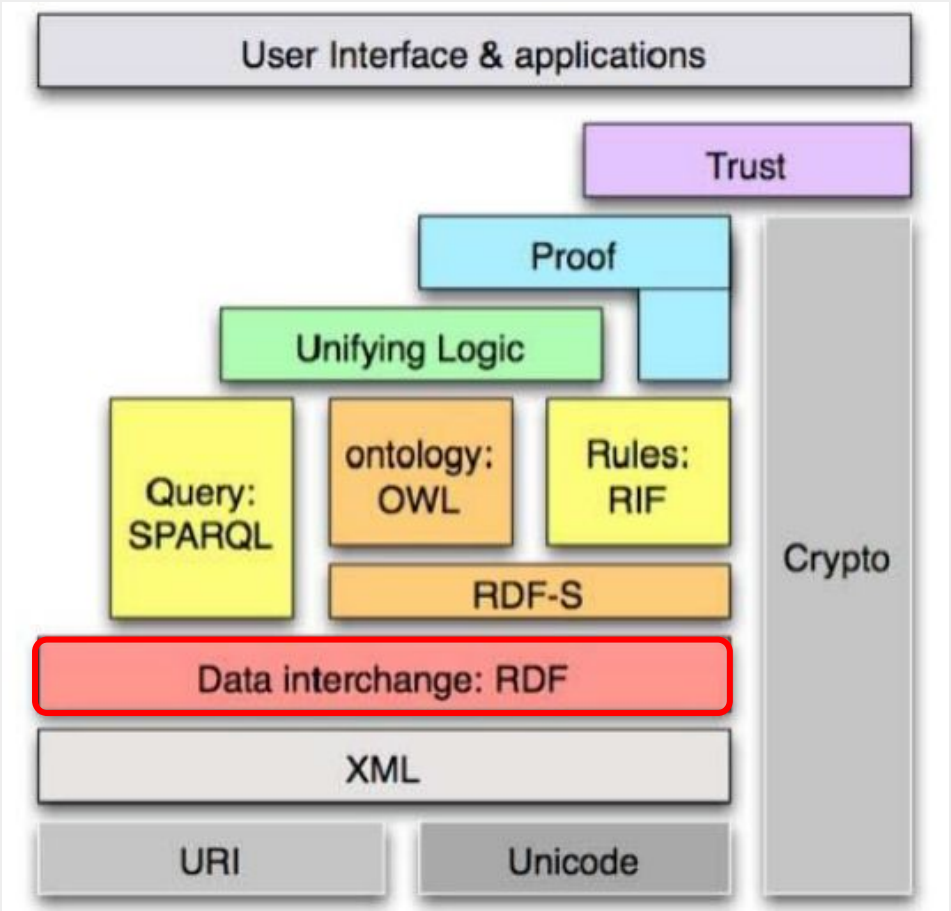
- Define a estrutura de documentos XML e de tipos de dados armazenados em cada elemento ou atributo
  a. Que elementos e atributos podem aparecer?
  b. Quantos filhos o elemento tem? Qual a ordem destes?
  c. Tipos de dados para elementos e atributos
- Regras gramaticais de validação de um ou mais documentos XML
- Não é obrigatória, mas pode ser interessante
- São extensíveis e reusáveis

# XML e Web Semântica

- XML não se adequa ao intercâmbio de dados na Web Semântica
- Dado pode ser representado de uma forma em um local e de outra no outro
- Web semântica demanda um modelo que represente qualquer informação de forma:
  a. Universal
  b. Legível por Máquinas
  c. Fácil de Integrar em diferentes fontes
- Isto justifica o nascimento do modelo RDF

# RDF

Arquitetura em camadas da Web Semântica



# RDF

- Resource Description Framework
- Padrão W3C para intercâmbio de dados na Web Semântica
- Feito para ser entendido por máquinas, não pessoas
- Escrito em XML (Linguagem: RDF/XML)
- Descreve recursos com propriedades e valores para essas propriedades

# Exemplos de uso RDF

- Pode ser usado para descrever:
  a. Propriedades de itens
  b. Informações de páginas web
     i. Conteúdo, autor, datas de criação e modificação, etc;
  c. Conteúdo em motores de pesquisa
  d. Bibliotecas Eletrônicas

# Recurso, Propriedade e Valor de Propriedade

- RDF identifica coisas através de IRIs
- Recurso: Qualquer coisa que tenha uma URI
  - Exemplo: "https://www.w3schools.com/rdf"
- Propriedade: Um recurso que possui um nome
  - Exemplo: “Autor”, “Homepage”
- Valor: Valor associado a uma Propriedade
  - Exemplo: “Jan Egil Refsnes”

# Declarações/Statements RDF

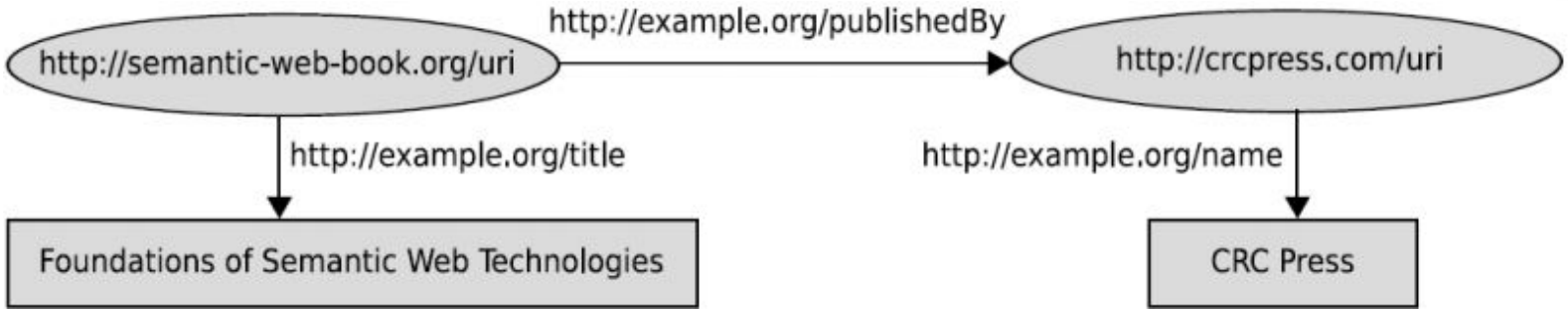
- A combinação de um Recurso, uma Propriedade e um Valor de Propriedade forma um Statement
- Esses elementos também são conhecidos como
  a. **Sujeito** - Do qual se declara algo
  b. **Predicado** - Descreve relacionamentos
  c. **Objeto** - Valor de uma propriedade ou recurso
- Uma Tripla RDF é um Statement nesta forma

# Exemplo de Tripla RDF

- Declaração: “O autor de https://www.w3schools.com/rdf é Jan Egil Refsnes”
- **Sujeito**: https://www.w3schools.com/rdf
- **Predicado**: autor
- **Objeto**: Jan Egil Refsnes

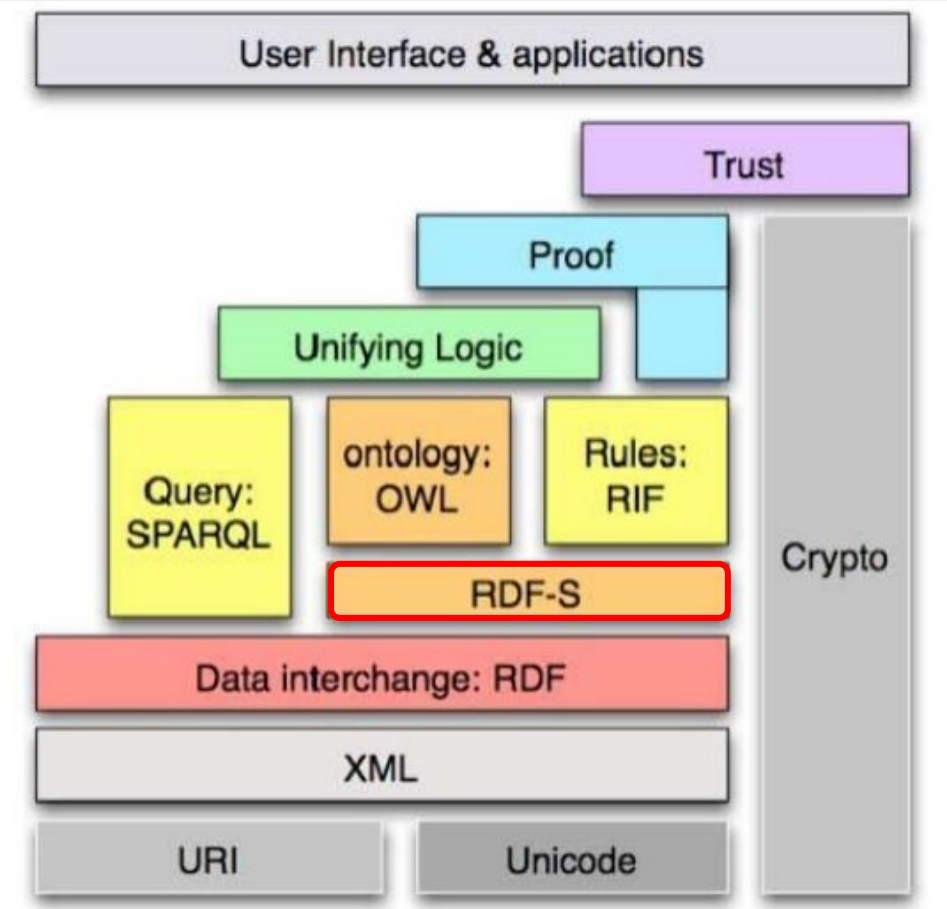
# RDF Mapeadas em Grafos Dirigidos

- Triplas podem ser mapeadas diretamente em grafos dirigidos (Do recurso para o valor da propriedade)
  a. Convencionalmente, as IRIs de sujeitos e objetos são ovais
  b. Se o objeto for literal, usam-se retângulos
  c. Predicados usam arestas são setas do predicado ao objeto (vértices)
- Não é usado para exibir dados, mas sim para estruturá-los



# RDF-S e OWL2

Arquitetura em camadas da Web Semântica



*Ontologia é uma especificação* ***explícita*** *e* ***formal*** *de uma* ***conceitualização compartilhada***.

Dieter **Fensel**

# Ontologia

- **Conceitualização** - modelo abstrato de algum fenômeno que identifica conceitos relevantes deste
- **Explícita** - conceitos e restrições são explicitamente definidos
- **Formal** - a representação da ontologia deve ser feita sob uma linguagem legível por sistemas de software
- **Compartilhada** - a ontologia representa o conhecimento consensual de um domínio

# Descrição de Ontologias

- A semântica do vocabulário de uma ontologia deve ser descrita formalmente
- Para isso faz-se uso de modelagem
- Uma ontologia pode ser representada por classes, propriedades e objetos
- As relações na ontologia são de variados tipos, como a transitividade, simetria, equivalência, etc.

# Construção de Ontologias

- O processo de construção de Ontologias é dividido em cinco fases:
  1. Especificação
  2. Conceitualização
  3. Formalização
  4. Implementação
  5. Manutenção

# Construção de Ontologias

- Especificação
  - Propósito e Escopo da Ontologia
  - Por quê? Para quê? Para quem?
- Conceitualização
  - Descrição em modelo conceitual
  - Consiste de conceitos do domínio e seus relacionamentos
  - Deve atender à especificação

# Construção de Ontologias

- Formalização

  - Transformação do modelo conceitual em um modelo formal
  - Organização de conceitos em hierarquias (generalização, especialização e composição)

- Implementação

  - Transformação do modelo formal em linguagem de representação de conhecimento
  - Escolha da linguagem baseada em expressividade

# Construção de Ontologias

- Manutenção

  - Atualização e correção da ontologia implementada
  - Ocasionada por novas demandas
  - Busca garantir consistência da ontologia

# Construção de Ontologias

- Em termos práticos, o desenvolvimento de uma Ontologia inclui:
  1. Definir as classes na Ontologia
  2. Arranjar as classes em uma hierarquia taxonomica (subclass - superclass)
  3. Definir atributos e valores aceitáveis para estes
  4. Preencher atributos com valores para as instâncias

# Linguagens de Ontologia

- A escolha de uma linguagem para a representação de uma ontologia é baseada na expressividade do modelo formal da ontologia
- RDF é limitado
  a. Expressa fatos, mas não modela domínios
  b. Não fornece uma linguagem para modelagem de recursos que descreve
  c. Não consegue representar a semântica desses recursos
- Para tratar essas limitações, criou-se a linguagem RDF Schema

# RDFS

- RDF SCHEMA
- Extensão semântica do RDF
- Possibilita construção simples Ontologias
- RDF descreve recursos através de classes, propriedades e valores
- RDFS descreve classes específicas para uma aplicação, e propriedades

# RDFS

- Permite a aplicações deduzir informações com base na semântica expressa pelo vocabulário da ontologia construída
- Especificação formal de modelo abstrato de um domínio de conhecimento
- Recursos podem ser definidos como instâncias de classes, e subclasses de classes
- Classes em RDFS são muito parecidas com as de OO

# Sintaxe RDFS

- **Recursos** podem ser organizados em classes, ou seja, classes são recursos
- **Membros** de classes são chamados **Instâncias** ou indivíduos
- Propriedade rdf:type permite declarar que um recurso é **instância** de uma classe
- Classes diferentes podem ter a mesma instância
    - Ex.: Lucas como aluno ou como morador da cidade Goiânia

# Sintaxe RDFS

- Construtor rdfs:Class cria **classes** de recursos
- Propriedade rdfs:subClassOf cria **especializações** entre classes
  a. Classes podem ser organizadas de forma hierarquizada
  b. Classe pode ter múltiplas superclasses
  c. Propriedade transitiva

# Exemplo de RDFS

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base="http://www.animals.fake/animals#">

<rdfs:Class rdf:ID="animal" />
<rdfs:Class rdf:ID="horse">
  <rdfs:subClassOf rdf:resource="#animal"/>
</rdfs:Class>

</rdf:RDF>
```

# Sintaxe RDFS

- Esses mecanismos sintáticos permitem que conhecimento implícito possa ser obtido
- Se no exemplo anterior “animal” fosse uma subclasse de ser vivo

```
<rdfs:Class rdf:ID="animal" />

<rdfs:Class rdf:ID="animal">

  <rdfs:subClassOf rdf:resource="#living being"/>

</rdfs:Class>
```

- Pela transitividade, “horse” é também “living being”, consequência lógica (conhecimento implícito)

# Vantagens do Vocabulário RDFS

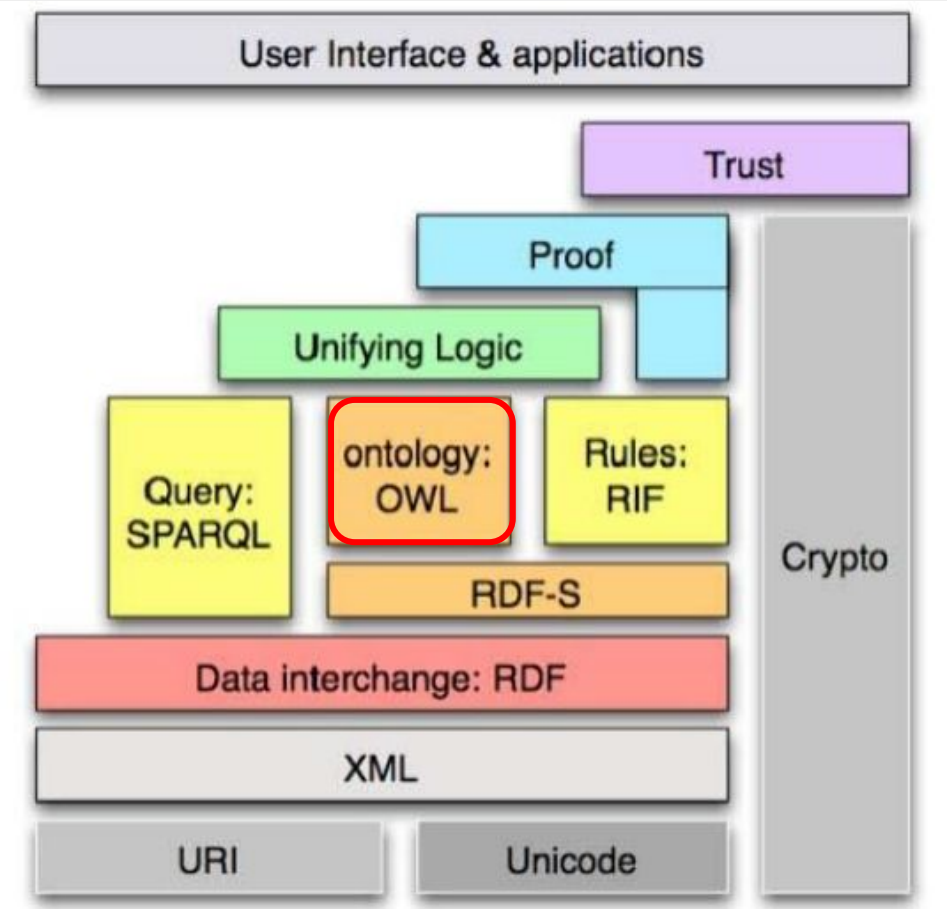
- Modelar recursos como classes ou membros de classes
- Especificar hierarquias de classes e de propriedades
- Permite declarar que uma classe de recursos pertence a uma dada propriedade
- Determinar quais valores (de outro recurso ou um literal) uma propriedade pode assumir

# Lacunas do RDFS

- RDFS é simples demais para construir ontologias com maior expressividade e complexidade lógica
  - Não oferece primitivas de simetria, unicidade, inversão, disjunção, reflexão, união, equivalência e intersecção
- Necessidade de linguagem de ontologia mais expressiva sobre dados e baseada nos padrões da Web Semântica para representação de informação

# RDF-S e OWL2

Arquitetura em camadas da Web Semântica



# OWL2

- Criada pelo W3C, OWL2 é a linguagem padrão para construir ontologias para a Web Semântica
- Web Ontology Language
- Estende o vocabulário do RDFS para modelar conhecimento de domínio
- Possui construtores mais ricos em questão de expressividade e inferência

# Características da OWL2

- Usa o modelo de dados padrão RDF, sintaxe XML e especificações do XML
- Pode representar relacionamentos de simetria, unicidade, inversão, transitividade, reflexão, disjunção, união, complemento, equivalência, etc.
- Suporte a restrições sobre valores e cardinalidades de propriedades
    a. Como uso de quantificações universais e existenciais
    b. Cardinalidade exata, mínimo e máxima

# Elementos da OWL2

- Para modelar as ontologias, OWL faz uso das seguintes abstrações:
  a. Classes
  b. Propriedades
  c. Instâncias (indivíduos)

# Elementos da OWL2

- Para declarar estas abstrações, os seguintes métodos são utilizados:
  a. **owl:Class** para Classes
  b. **owl:DatatypeProperty** para Propriedades que são atributos, ou seja, Tipos de Dados
  c. **owl:ObjectProperty** para Propriedades que são relacionamentos, ou seja, Objetos
  d. **owl:Individual** para Instâncias (indivíduos)

# Hierarquia de Classes na OWL2

- Para modelar a hierarquia de classes na owl2, faz-se uso do método rdfs:subClassOf; como o nome sugere, um método do RDFS
- Por exemplo, definir uma classe Coordenador que é subclasse de um recurso Professor

```
<owl:Class rdf:ID="Coordenador" />
  <rdfs:subClassOf rdf:resource="#Professor"/>
</owl:Class>
```

# Criação de indivíduos

- Outra sintaxe para a declaração de classes, mais simples, pode ser encontrada no exemplo a seguir:

Comparação na criação de um indivíduo:

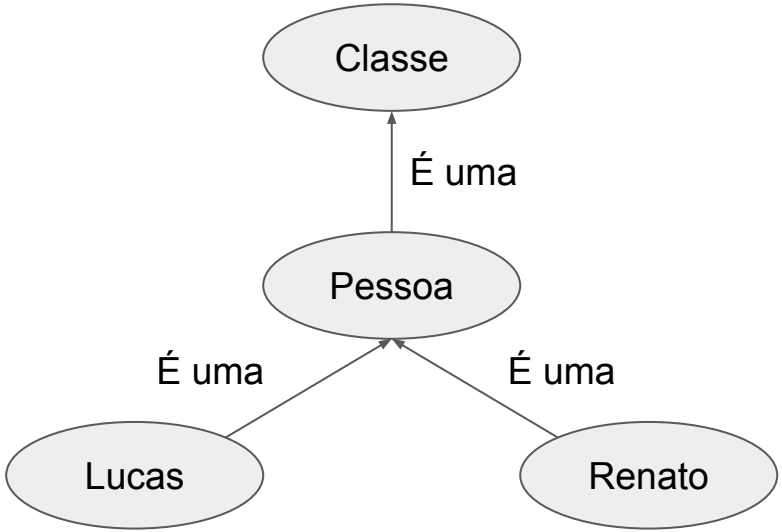
```
<owl:Class rdf:ID="Pessoa" />

<Pessoa rdf:ID="Lucas"/>
```

```
Pessoa a owl:Class

Lucas a Pessoa
```

# Criação de indivíduos



# Subclasses X Indivíduos

- **Subclasse** é um subconjunto de **membros** de uma classe **pai**
- **Indivíduo** (ou instância) representa um **membro** individual de uma classe
- No exemplo anterior, Lucas e Renato são indivíduos da classe pessoa, mas a classe pessoa pode ter uma subclasse Professor, por exemplo

# OWL - Propriedades

- As propriedades servem para estabelecer relacionamentos **entre indivíduos** ou **entre indivíduos e dados**
- 1) Propriedades **ObjectProperty**

  “Professor **orienta** Aluno”

- 2) Propriedades **DataTypeProperty**

  “Professor **nomeProfessor** Renato”

# OWL - ObjectProperty

```
<owl:objectProperty rdf:ID="Orienta">
    <rdfs:domain rdf:resource="#Professor"/>
    <rdfs:range rdf:resource="#Aluno"/>
</owl:objectProperty>
```

# OWL - DatatypeProperty

```
<owl:DataTypeProperty rdf:ID="nomeProfessor">
    <rdfs:domain rdf:resource="#Professor"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DataTypeProperty>
```

# Representando relacionamentos

- **ObjectProperty** possui subclasses para representar relacionamento de indivíduos, algumas delas são:

  **owl:TransitiveProperty**

  **owl:SymmetricProperty**

  **owl:FunctionalProperty**

  **owl:inverseOf**

  **owl:inverseFunctionalProperty**

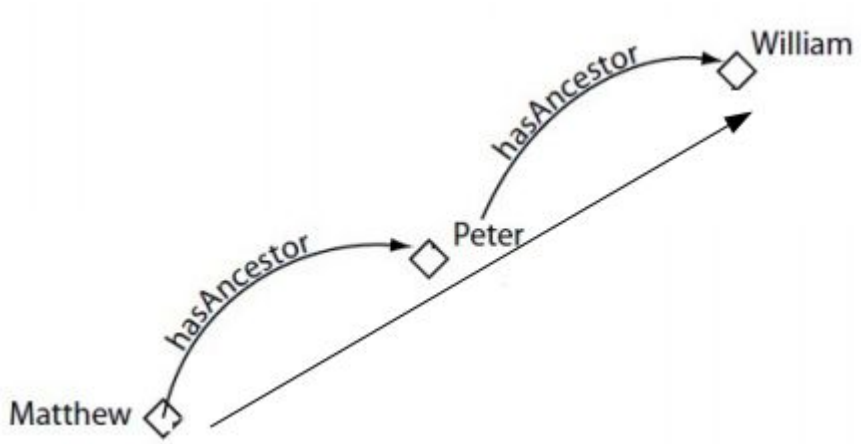
  ...

# owl:TransitiveProperty

- Descreve relacionamentos de **ascendência, hierarquia, parte-todo, estar contido em, …**
- hasAncestor a **owl:TransitiveProperty;**
  **rdfs:domain** Person;
  **rdfs:range** Person.

  Matthew hasAncestor Peter
  Peter hasAncestor William

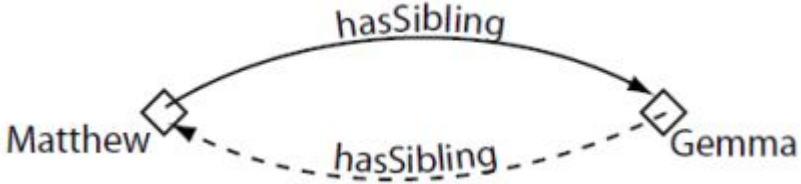
**Figura 1:** Exemplo de transitividade



# owl:SymmetricProperty

- Descreve relacionamentos de **namoro, amizade, casamento, igualdade, adjacência, proximidade ...**
- hasSibling a **owl:SymmetricProperty;**
  **rdfs:domain** Person;
  **rdfs:range** Person;
  Matthew hasAncestor Gemma

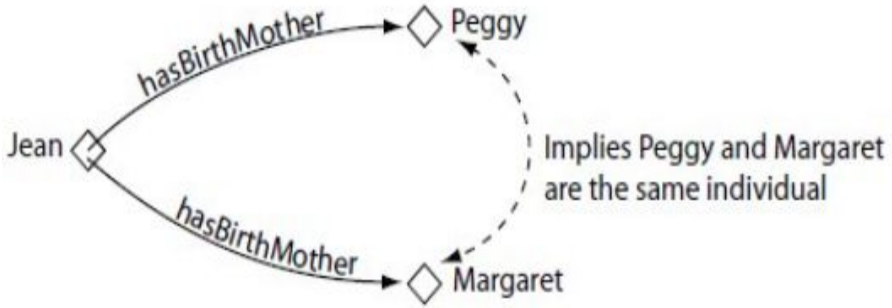
**Figura 2:** Exemplo de simetria



# owl:FunctionalProperty

- Se a propriedade é funcional para um **indivíduo A**, então pode haver **no máximo um indivíduo** relacionado a A através dessa propriedade
- hasBirthMother a **owl:FunctionalProperty;**
  **rdfs:domain** Person;
  **rdfs:range** Woman;
  Jean hasBirthMother Peggy
  Jean hasBirthMother Margaret

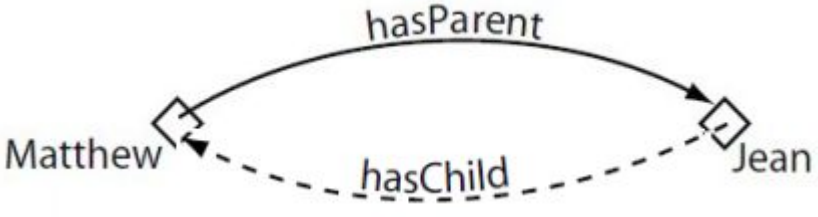
**Figura 3:** Exemplo de propriedade funcional



# owl:inverseOf

- Se há um relacionamento em uma **direção** implica que há outro na **direção inversa**
- hasParent **owl:inverseOf** hasChild
  **rdfs:domain** Person;
  **rdfs:range** Person;
  Matthew hasParent Jean

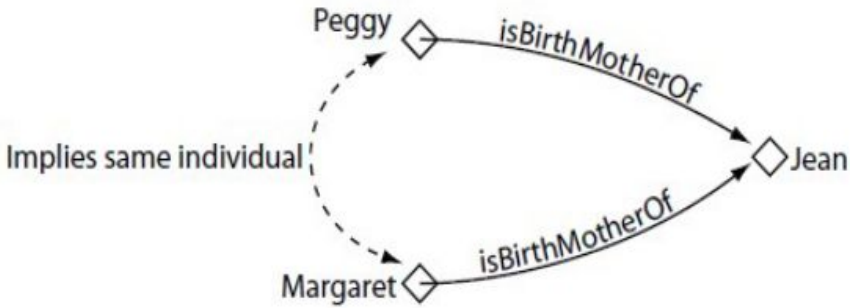
**Figura 4:** Exemplo de relacionamento inverso



# owl:InverseFunctionalProperty

- Se a propriedade é funcional inversa, para todo indivíduo A relativo ao **rdfs:range** existe apenas um indivíduo relativo ao **rdfs:domain** (conjunto domínio) relacionado através dessa propriedade
- isBirthMotherOf a **owl:InverseFunctionalProperty;**
  **rdfs:domain** Woman;
  **rdfs:range** Person;
  Peggy isBirthMotherOf Jean
  Margaret isBirthMotherOf Jean

**Figura 5:** Exemplo de relacionamento inverso para função



# Restrições em Propriedades

- Além de diversos métodos para modelar relacionamentos, OWL também pode criar restrições aos **valores** e à **cardinalidade** das propriedades
- Os métodos para restrição de **valores** são:

  owl:allValuesFrom; owl:someValuesFrom; owl:hasValue

- Os métodos para restrição de **cardinalidade** são:

  owl:cardinality; owl:maxCardinality; owl:minCardinality

# owl:allValuesFrom

```
<owl:Class rdf:ID="Mae">
    <owl:restriction>
        <owl:onProperty rdf:resource="#temFilho"/>
        <owl:allValuesFrom rdf:resource="#Pessoa"/>
    </owl:restriction>
</owl:Class>
```

- **Todos** os valores associados à propriedade **temFilho,** de indivíduos da classe **Mãe**, são instâncias da classe **Pessoa**

# owl:someValuesFrom

```
<owl:Class rdf:ID="Chefe">
    <owl:restriction>
        <owl:onProperty rdf:resource="#mandaEm"/>
        <owl:someValuesFrom rdf:resource="#Funcionario"/>
    </owl:restriction>
</owl:Class>
```

- **Pelo menos um** dos valores associados à propriedade **mandaEm,** de indivíduos da classe **Chefe**, é instância da classe **Funcionário**

# owl:hasValue

```
<owl:Class rdf:ID="FilhosDoAurimar">
    <owl:restriction>
        <owl:onProperty rdf:resource="#temPai"/>
        <owl:hasValue rdf:resource="#Aurimar"/>
    </owl:restriction>
</owl:Class>
```

- O valor associado à propriedade **temFilho**, de indivíduos da classe **FilhosDoAurimar**, é sempre a instância **Aurimar**

# owl:cardinality

```
<owl:Class rdf:ID="Pessoa">
     <owl:restriction>
          <owl:onProperty rdf:resource="#sobrenome"/>
          <owl:cardinality rdf:datatype=
               "&xsd;nonNegativeInteger">1
          </owl:cardinality>
</owl:Class>
```

- Indivíduos da classe **Pessoa** possuem um único sobrenome

# owl:minCardinality

```
<owl:Class rdf:ID="Pessoa">
     <owl:restriction>
          <owl:onProperty rdf:resource="#apelido"/>
          <owl:minCardinality rdf:datatype=
               "&xsd;nonNegativeInteger">0
          </owl:minCardinality>
</owl:Class>
```

- Indivíduos da classe **Pessoa** podem **não ter** apelido

# owl:maxCardinality

```
<owl:Class rdf:ID="Pessoa">
     <owl:restriction>
          <owl:onProperty rdf:resource="#prefixo"/>
          <owl:maxCardinality rdf:datatype=
               "&xsd;nonNegativeInteger">1
          </owl:maxCardinality>
</owl:Class>
```

- Indivíduos da classe **Pessoa** têm, **no máximo**, um prefixo

# OWL - Disjunção entre Classes

- Classes disjuntas são aquelas que **NÃO** possuem **indivíduos em comum**
- Um professor de Dedicação Exclusiva, **NÃO** pode fazer parte dos professores que só dão 20 horas aula semanais

```
ProfessorDE        a                owl:Class .
Professor20h       a                owl:Class .
ProfessorDE        owl:disjointWith Professor20h .
```

# OWL - Equivalência entre Classes

- Classes equivalentes são aquelas que possuem **os mesmos indivíduos**, sem considerar hierarquia entre estas
- **Todo** professor **também é** um docente

```
Docente      a                  owl:Class .
Professor    a                  owl:Class .
Docente      owl:equivalentClass  Professor .
```

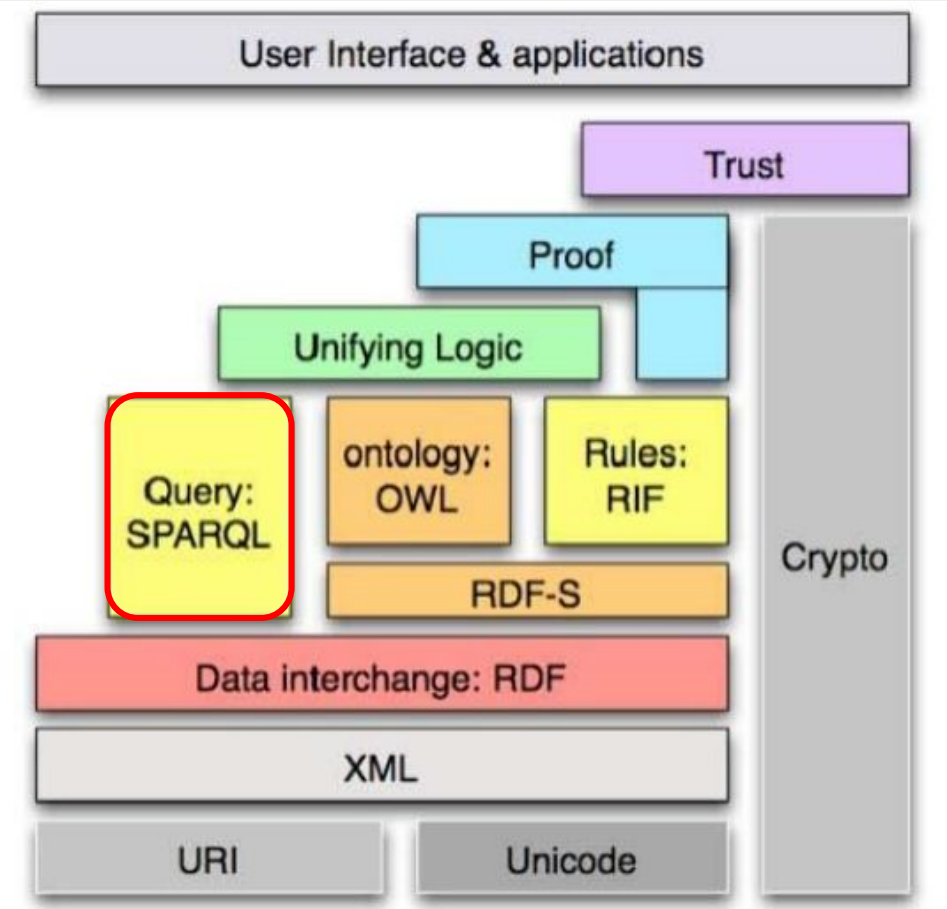
# OWL - Equivalência entre Indivíduos

- **IRIs diferentes** podem representar semanticamente **um mesmo indivíduo** no mundo real
- **LucasFelipe** é o mesmo indivíduo que LucasFelipeMS

| | | |
|---|---|---|
| LucasFelipe | owl:sameAs | LucasFMS . |
| LucasFMS | owl:sameAs | LucasFelipeMS . |
| LucasFelipe | owl:differentFrom | LucasMorais . |

# SPARQL

Arquitetura em camadas da Web Semântica



# Linguagem de consulta SPARQL

- Linguagem padrão para a escrita de consultas sobre triplas; segundo o modelo RDF
- Permite quatro formas a consulta a triplas RDF
  a. Cada uma tem sua finalidade
  b. Enfoque do artigo é na SELECT
- Assim como em SQL, há cláusulas da SPARQL que alteram o resultado de uma consulta

# As outras três formas

- ASK
  a. Para saber se um subgrafo existe na fonte de dados
  b. O resultado é booleano
- CONSTRUCT
  a. A principal diferença para o SELECT é que o resultado é um novo grafo
  b. Não é uma ligação de valores a variáveis de consulta
- Describe
  a. Quando se quer saber tudo sobre um recurso ou fonte de dados

# Boas Práticas

# Gerenciamento de IRIs

- A geração de IRIs deve garantir que sejam únicas, consistentes e resolúveis
  a. Unicidade: Pode-se usar uma única IRI para definir o espaço de nomes de cada ontologia criada, ou para servir de base para a geração de IRIs por meio de uma aplicação que produz dados RDF
  b. Consistência: Operações sobre recursos não devem modificar suas IRIs
  c. Resolubilidade: A IRI do espaço de nomes de uma ontologia ou de um documento RDF deve ser acessível na Web Semântica

# Especificação de Unidades de Medida

- RDF e OWL não dão suporte direto para unidades de medida de valores literais
- Para modelar literais uma técnica é usar propriedades e tipos de dados com unidades específicas para criar uma versão única de cada propriedade ou tipo de dado para a qual se tem uma unidade de medida
    a. Comprimento: lenght-feet ou lenght-meter
    b. Idade: age-years
    c. Ponto flutuante: float-feet
    d. Anos: int-years

# Representação de relacionamentos n-ários

1. Em RDF e OWL os predicados são binários - Um predicado liga um único sujeito a um único objeto
2. Às vezes é preciso representar relacionamentos n-ários
    a. Um objeto intermediário é usado e age como um container para os valores
    b. Esse modelo simplifica consultas e não usar o objeto intermediário poderia fazer com que valores fossem atribuídos a outro sujeito

# Obrigado

lucassilva@inf.ufg.br

Dúvidas ou sugestões?