
# Web Semântica

Diogo Silva
NºMec: 107647

Bruno Páscoa
NºMec: 107418

Mestrado em Engenharia Informática, 2024/2025

# Índice

# Introdução ao Tema

O universo de Star Wars é um dos mais icónicos e influentes da cultura popular. Criado por George Lucas em 1977, tornou-se uma das franquias de maior sucesso na história do entretenimento, abrangendo filmes, séries, livros, jogos e muitas outras formas de mídia. Com uma base de fãs extremamente ativa e apaixonada, Star Wars continua a marcar gerações e a expandir seu legado.
O nosso objetivo neste projeto é desenvolver um sistema de informação baseado na web que permita a exploração e gestão de informações sobre o universo de Star Wars.
A escolha deste tema justifica-se pelo imenso volume de informação presente no universo de Star Wars, que abrange personagens, planetas, naves e muito mais. Esse vasto conjunto de dados pode ser estruturado e modelado num sistema semântico, permitindo uma abordagem enriquecedora para a sua exploração.

# Ontologia

## Visão Geral da Ontologia

A ontologia desenvolvida modela de forma abrangente o universo Star Wars, capturando os principais conceitos, entidades, relações e características que definem esta vasta galáxia. O domínio de conhecimento engloba personagens, droids, planetas, cidades, espécies, organizações, veículos, armas, naves, filmes, músicas, citações e as complexas relações que os interligam.

A ontologia foi desenvolvida utilizando o Protégé, uma ferramenta de edição ontológica que proporcionou um ambiente integrado e intuitivo para a modelação do conhecimento. Esta plataforma permitiu a criação visual e estruturada da hierarquia de classes, facilitando a organização conceptual das entidades do universo Star Wars. Através do Protégé foi possível definir propriedades de objeto e de dados, bem como implementar restrições semânticas que garantem a consistência lógica da ontologia. A ferramenta incorpora o reasoner HermiT, que foi fundamental para a validação contínua da consistência ontológica e para o teste da classificação automática das entidades com base nos axiomas definidos.

### Classes Principais

A estrutura hierárquica da ontologia é organizada em torno de várias classes fundamentais, cada uma representando um aspecto central do universo Star Wars:

#### Hierarquia de Entidades

- **Character** - Representa personagens individuais
  - **Droid** - Droides e inteligências artificiais
- **Planet** - Planetas habitáveis ou significativos
- **Organizations** - Grupos e facções
- **Vehicle** - Meios de transporte e naves
  - **Starship** - Naves espaciais
- **Specie** - Espécies inteligentes da galáxia

- **Weapon** - Armas e equipamentos de combate
- **City** - Cidades importantes
- **Quote** - Citações das personagens
- **Film** - Filmes do universo Star Wars
- **City** - Cidades importantes

## Propriedades

### Object Properties

As relações entre entidades são modeladas através de propriedades de objeto que estabelecem conexões semânticas:

- **appears_in** - Relaciona algumas entidades com o filme onde aparecem
  - Domínio: Character or Organizations or Planet or Quote or Starship or Vehicle or Weapon
  - Range: Film
- **associated_with** - Liga as músicas aos filmes onde aparecem
  - Domínio: Music
  - Range: Film
- **has_quote** - Relaciona os personagens as suas citações
  - Domínio: Character
  - Range: Quote
  - Propriedade Inversa: said_by
- **homeworld** - Relaciona os personagens e espécies ao seu planeta
  - Domínio: Character or Specie
  - Range: Planet
  - Características: Functional
- **ispilotof** - Relaciona os personagens ao veiculo ou nave que pilotam
  - Domínio: Character
  - Range: Starship or Vehicle
  - Propriedade Inversa: pilot
- **leader** - Relaciona um persongem que é líder de uma organização
  - Domínio: Organizations
  - Range: Character
  - Cardinalidade: tem que ter pelo menos um líder
- **member** - Relaciona um personagem que é membro de uma organização
  - Domínio: Organizations
  - Range: Character
  - Cardinalidade: tem que ter pelo menos um membro
- **pilot** - Relaciona um veículo ou nave com o personagem que a pilota
  - Domínio: Starship or Vehicle
  - Range: Character
  - Propriedade Inversa: ispilotof
- **planet** - Relaciona a cidade com o planeta onde se localiza
  - Domínio: City
  - Range: Planet

- **resident** - Relaciona um planeta com os personagens que residem no planeta
  - Domínio: Planet
  - Range: Character
- **said_by** - Relaciona uma citação com a personagem que a disse
  - Domínio: Quote
  - Range: Character
  - Propriedade Inversa: has_quote
- **specie** - Relaciona uma personagem com a sua especie
  - Domínio: Character
  - Range: Specie
- **used_by** - Relaciona uma arma com a personagem que a usa
  - Domínio: Weapon
  - Range: Character
  - Propriedade Inversa: uses_weapon
- **uses_weapon** - Relaciona um personagem com a arma que usa
  - Domínio: Character
  - Range: Weapon
  - Propriedade Inversa: used_by

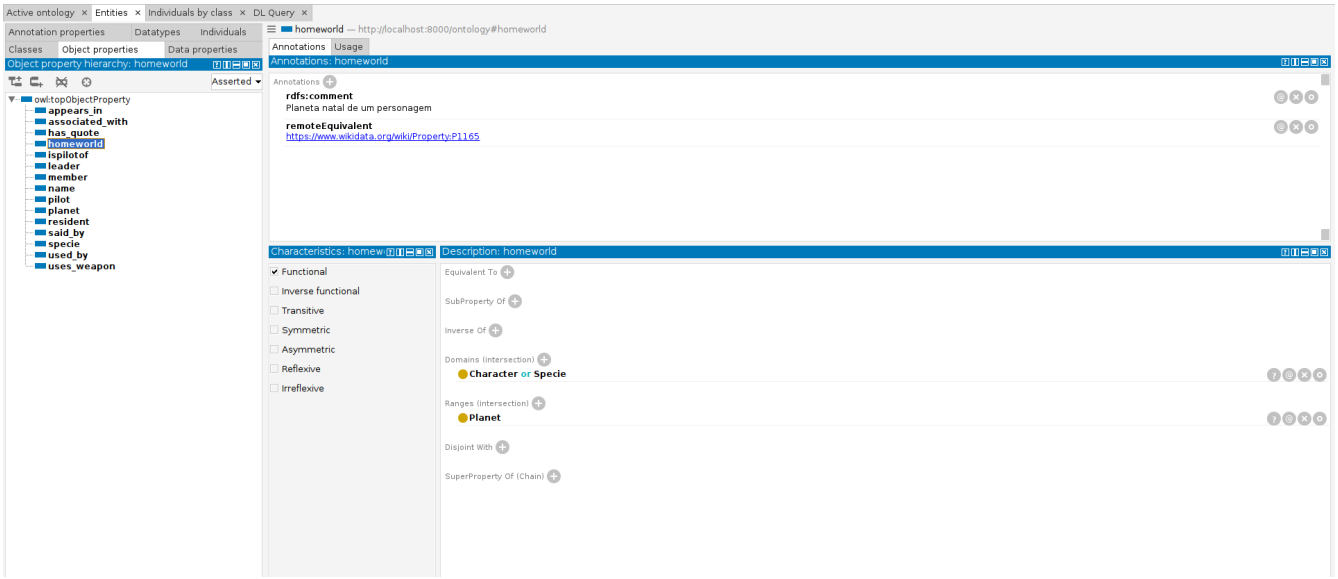


Figure 1: Object Properties no Protege

## Data Properties

As propriedades de dados (data properties) capturam características específicas e atributos literais das entidades, fornecendo informações detalhadas sobre cada conceito modelado na ontologia. Estas propriedades estabelecem relações entre as entidades e valores de tipos de dados primitivos, permitindo a representação de informações como nomes, idades, dimensões e outras características quantificáveis ou descritivas.

Como podemos observar nas figuras 2 e 3, a ontologia incorpora um conjunto abrangente de data properties que cobrem os diversos aspetos do domínio Star Wars. Estas propriedades foram organizadas de forma hierárquica e categorizada, garantindo uma representação semântica rica e estruturada dos dados.

As data properties implementadas incluem propriedades fundamentais como identificadores únicos, nomes e descrições, bem como propriedades mais específicas do domínio, tais como características físicas de personagens, especificações técnicas de veículos, dados geofísicos de planetas, entre outras. Esta diversidade de propriedades

permite não apenas o armazenamento estruturado de informação, mas também facilita consultas complexas e inferências baseadas em critérios específicos.

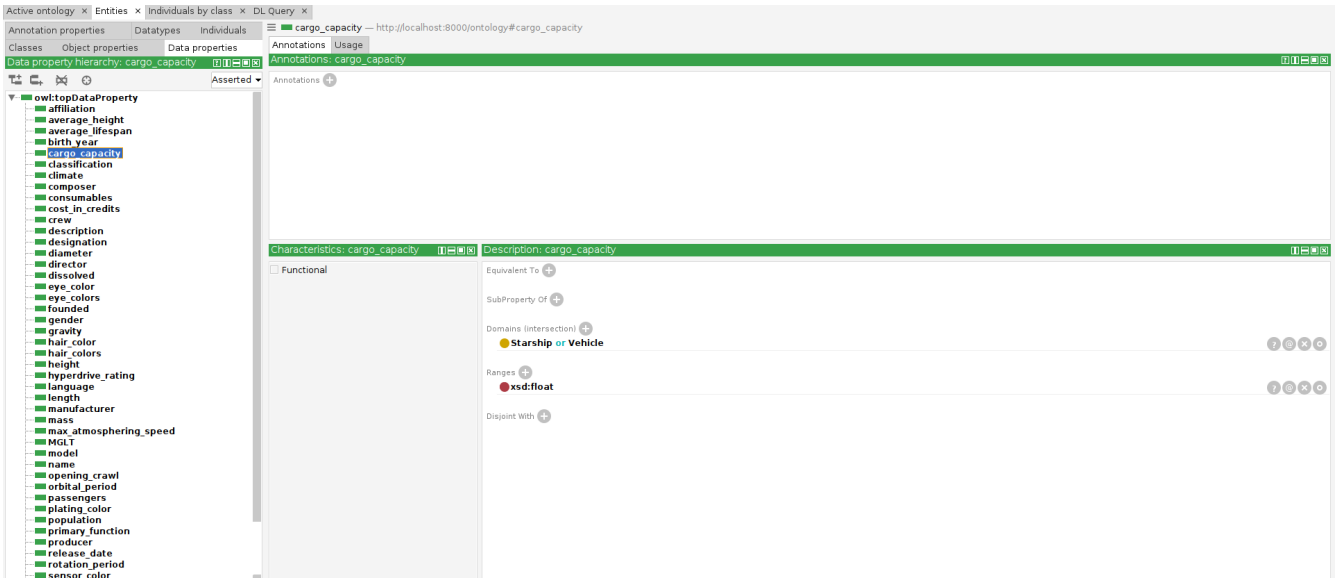


Figure 2: Data Properties no protege

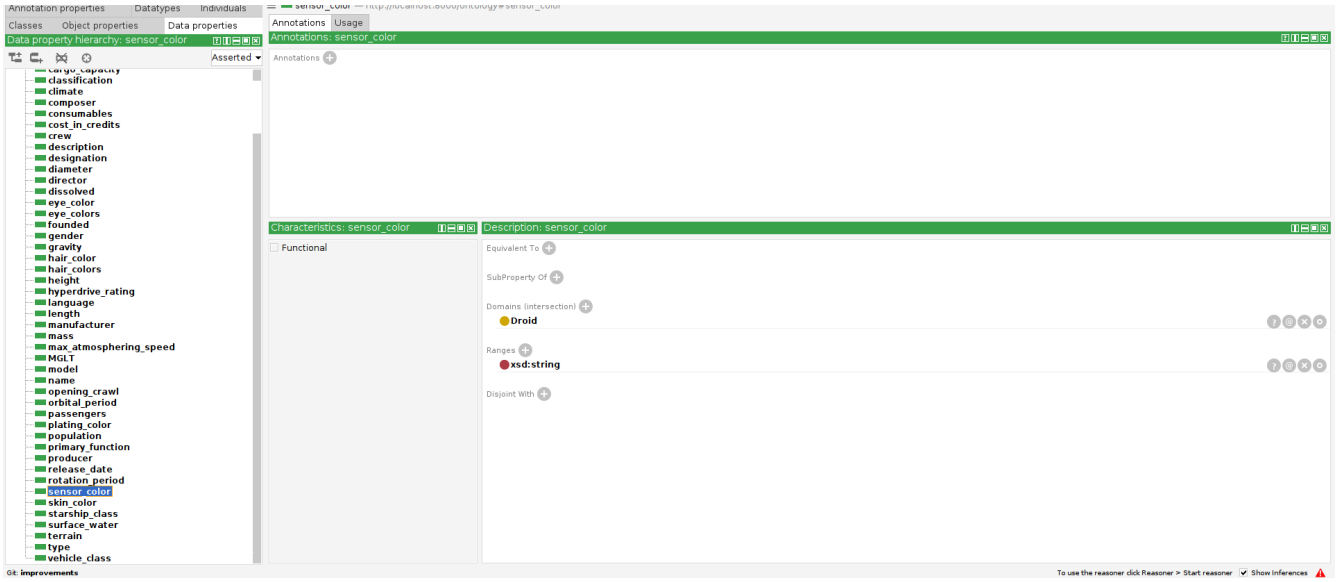


Figure 3: Data Properties no Protege

## Restrições e Axiomas

Para garantir a consistência semântica e a integridade conceptual da ontologia, foram definidas restrições específicas que estabelecem condições necessárias para a classificação automática de entidades em determinadas classes. Estas restrições estabelecem condições obrigatórias que devem ser satisfeitas para que uma entidade seja corretamente classificada numa determinada classe, garantindo assim a precisão e consistência da ontologia.

As principais restrições implementadas incluem:

- **City** – Uma entidade só é classificada como cidade se possuir obrigatoriamente um planeta associado através da propriedade `planet`. Esta restrição garante que todas as cidades estejam geograficamente localizadas num planeta específico do universo Star Wars.
- **Music** – Para uma entidade ser considerada uma composição musical, deve ter associada a propriedade `composer` com um valor de string correspondente ao nome do compositor.
- **Starship** – Uma nave espacial é automaticamente classificada como tal apenas se possuir a propriedade `starship_class` definida, indicando a sua categoria ou tipo específico dentro da classificação de naves do universo Star Wars.

Estas restrições são implementadas através de axiomas OWL que definem condições necessárias e suficientes para a pertença às respetivas classes, permitindo que o motor de inferência classifique automaticamente as entidades com base nas suas propriedades e relações estabelecidas.

## Classificação Automática

A ontologia desenvolvida permite a classificação automática da maioria das entidades através dos motores de inferência padrão. Contudo, identificou-se uma limitação específica relativamente à classe `Weapon`, para a qual o reasoner integrado no GraphDB não possui capacidade de inferência automática devido às limitações do motor de inferência utilizado.

Como pode ser visto em 1, foi criada uma regra SPIN que implementa a classificação automática de entidades na classe `Weapon`. Esta regra utiliza uma consulta SPARQL que examina a propriedade `ont:type` de cada entidade e aplica um filtro que verifica se o valor do tipo contém uma das categorias de armas predefinidas: "melee", "blaster", "explosive" ou "projectile". A verificação é realizada de forma case-insensitive através das funções `CONTAINS` e `LCASE`, garantindo que entidades com tipos correspondentes sejam automaticamente classificadas como armas.

```
PREFIX ont: <http://localhost:8000/ontology#>
INSERT {
    ?x a ont:Weapon .
}
WHERE {
    ?x ont:type ?type .
    FILTER(
        CONTAINS(LCASE(STR(?type)), "melee") ||
        CONTAINS(LCASE(STR(?type)), "blaster") ||
        CONTAINS(LCASE(STR(?type)), "explosive") ||
        CONTAINS(LCASE(STR(?type)), "projectile")
    )
}
```

1: Regra SPIN para classificação automática da classe Weapon

Para testar a inferência automática da ontologia, foi criado um ficheiro denominado all.rdf, que reúne tanto os dados quanto a ontologia. Esse ficheiro foi carregado no Protégé, onde se utilizou o raciocinador HermiT para validar as inferências. A Figura 4 demonstra que as inferências foram realizadas com sucesso.

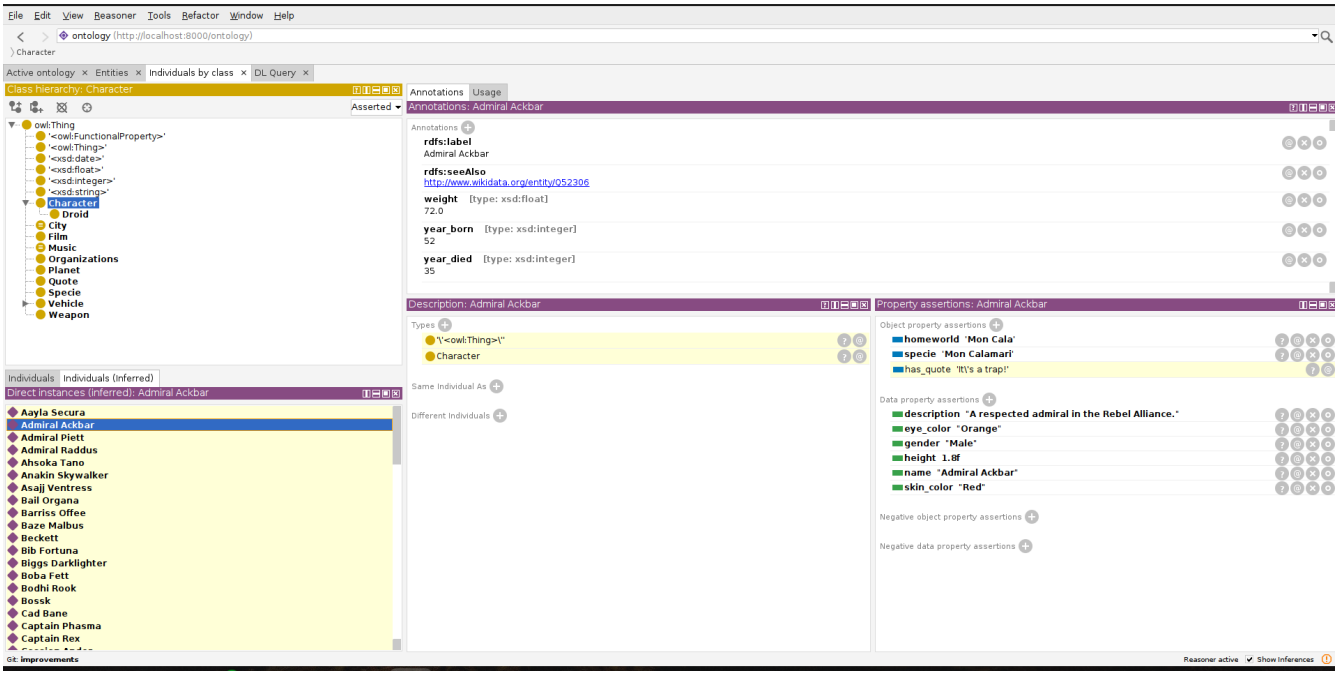


Figure 4: Enter Caption

# Inferências

As inferências constituem um mecanismo fundamental nos sistemas de conhecimento semântico, permitindo a derivação automática de novos factos a partir de informação existente através da aplicação de regras lógicas. No contexto desta ontologia do universo Star Wars, as inferências possibilitam o enriquecimento automático dos dados, estabelecendo novas relações entre entidades e classificando automaticamente elementos com base em propriedades e características específicas.

Para implementar estas capacidades de inferência, foram desenvolvidas regras SPIN (SPARQL Inferencing Notation) que complementam as funcionalidades de raciocínio automático da ontologia OWL. Estas regras foram inicialmente concebidas para serem aplicadas de forma global a todas as entidades em simultâneo, otimizando assim o processo de inferência. Contudo, durante a implementação e teste do sistema, foram identificadas limitações técnicas relacionadas com timeouts nos serviços da Wikidata, que impossibilitaram a execução de algumas das regras de forma massiva. Esta situação obrigou à reformulação da estratégia de implementação, sendo necessário aplicar algumas das regras SPIN individualmente a cada personagem.

## Regras SPIN

### Regras SPIN das Personagens

Como mencionado anteriormente, estas regras aplicam-se exclusivamente a uma entidade específica. As regras SPIN desenvolvidas para as personagens inferem automaticamente relações familiares complexas a partir das relações básicas já existentes no grafo de conhecimento, nomeadamente as propriedades fundamentais de Pai, Mãe e Filho/-Filha.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX : <http://localhost:8000/ontology#>

INSERT {{
    ?localX :hasSibling ?target .
    ?target rdfs:label ?siblingLabel .
}} WHERE {{
    ?localX rdfs:seeAlso wd:{wikidata_id} .

    SERVICE <https://query.wikidata.org/sparql> {{
        {{
            wd:{wikidata_id} wdt:P22 ?parent .
            ?sibling wdt:P22 ?parent .
            FILTER(?sibling != wd:{wikidata_id})
        }} UNION {{
            wd:{wikidata_id} wdt:P25 ?parent .
            ?sibling wdt:P25 ?parent .
            FILTER(?sibling != wd:{wikidata_id})
        }}
        OPTIONAL {{ ?sibling rdfs:label ?siblingLabel FILTER (lang(?siblingLabel) =
            "en") }}
    }}

    BIND(COALESCE(?localY, ?sibling) AS ?target)

    OPTIONAL {{ ?localY rdfs:seeAlso ?sibling }}

    FILTER NOT EXISTS {{ ?target rdfs:label ?siblingLabel }}
```

```
}}
```

2: Regra SPIN para a inferência de iramãos

Na regra 2 são inferidas automaticamente relações de parentesco entre irmãos, consultando a Wikidata para encontrar personagens que partilham o mesmo pai (P22) ou mãe (P25) e estabelecendo a propriedade :hasSibling entre a entidade local e os seus irmãos descobertos, fazendo a conexão com a entidade local, caso existam.

```
PREFIX : <http://localhost:8000/ontology#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>

INSERT {{
    ?localX :hasUncle ?target .
    ?target rdfs:label ?uncleLabel .
}} WHERE {{
    ?localX rdfs:seeAlso wd:{wikidata_id} .

    SERVICE <https://query.wikidata.org/sparql> {{
        wd:{wikidata_id} wdt:P22|wdt:P25 ?parent .
        ?uncle wdt:P22|wdt:P25 ?grandparent .
        ?parent wdt:P22|wdt:P25 ?grandparent .
        FILTER(?uncle != ?parent)
        ?uncle wdt:P21 wd:Q6581097 .
        OPTIONAL {{ ?uncle rdfs:label ?uncleLabel FILTER (lang(?uncleLabel) = "en")
            }}
    }}

    OPTIONAL {{ ?localUncle rdfs:seeAlso ?uncle }}
    BIND(COALESCE(?localUncle, ?uncle) AS ?target)

    FILTER NOT EXISTS {{ ?target rdfs:label ?uncleLabel }}
}}
```

3: Regra SPIN para a inferência de tios

Na regra 3 são inferidas relações de parentesco entre tios e sobrinhos, consultando a Wikidata para identificar irmãos masculinos (P21 = Q6581097) dos pais da entidade e criando a propriedade :hasUncle entre a entidade local e os tios descobertos.A regra navega através de duas gerações (pais e avós) para encontrar os tios do personagem.

```
PREFIX : <http://localhost:8000/ontology#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>

INSERT {{
    ?localX :hasAunt ?target .
    ?target rdfs:label ?auntLabel .
}} WHERE {{
    ?localX rdfs:seeAlso wd:{wikidata_id} .

    SERVICE <https://query.wikidata.org/sparql> {{
        wd:{wikidata_id} wdt:P22|wdt:P25 ?parent .
        ?aunt wdt:P22|wdt:P25 ?grandparent .
        ?parent wdt:P22|wdt:P25 ?grandparent .
        FILTER(?aunt != ?parent)
        ?aunt wdt:P21 wd:Q6581072 .
        OPTIONAL {{ ?aunt rdfs:label ?auntLabel FILTER (lang(?auntLabel) = "en") }}
    }}

    OPTIONAL {{ ?localAunt rdfs:seeAlso ?aunt }}
    BIND(COALESCE(?localAunt, ?aunt) AS ?target)

    FILTER NOT EXISTS {{ ?target rdfs:label ?auntLabel }}
}}
```

4: Regra SPIN para a inferência de tias

Na regra 4 são inferidas relações de parentesco entre tias e sobrinhos, consultando a Wikidata para identificar irmãs femininas (P21 = Q6581072) dos pais da entidade e criando a propriedade :hasAunt entre a entidade local e

as tias descobertas. A regra funciona de forma análoga à regra dos tios, navegando através de duas gerações para encontrar as tias.

```
PREFIX : <http://localhost:8000/ontology#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>

INSERT {{
    ?localX :hasNephew ?target .
    ?target rdfs:label ?nephewLabel .
}} WHERE {{
    ?localX rdfs:seeAlso wd:{wikidata_id} .

    SERVICE <https://query.wikidata.org/sparql> {{
        wd:{wikidata_id} wdt:P22|wdt:P25 ?parent .
        ?sibling wdt:P22|wdt:P25 ?parent .
        FILTER(?sibling != wd:{wikidata_id})
        ?sibling wdt:P40 ?nephew .
        ?nephew wdt:P21 wd:Q6581097 .
        OPTIONAL {{ ?nephew rdfs:label ?nephewLabel FILTER (lang(?nephewLabel) = "en
            ") }}
    }}

    OPTIONAL {{ ?localNephew rdfs:seeAlso ?nephew }}
    BIND(COALESCE(?localNephew, ?nephew) AS ?target)
}}
```

5: Regra SPIN para a inferência de sobrinhos

Na regra 5 são inferidas relações de parentesco entre tios/tias e sobrinhos, consultando a Wikidata para identificar filhos masculinos (P21 = Q6581097) dos irmãos da entidade e criando a propriedade :hasNephew entre a entidade local e os sobrinhos descobertos. A regra navega através dos irmãos (que partilham os mesmos pais) e depois identifica os seus filhos do sexo masculino para estabelecer a relação de sobrinho.

```
PREFIX : <http://localhost:8000/ontology#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>

INSERT {{
    ?localX :hasNiece ?target .
    ?target rdfs:label ?nieceLabel .
}} WHERE {{
    ?localX rdfs:seeAlso wd:{wikidata_id} .

    SERVICE <https://query.wikidata.org/sparql> {{
        wd:{wikidata_id} wdt:P22|wdt:P25 ?parent .
        ?sibling wdt:P22|wdt:P25 ?parent .
        FILTER(?sibling != wd:{wikidata_id})
        ?sibling wdt:P40 ?niece .
        ?niece wdt:P21 wd:Q6581072 .
        OPTIONAL {{ ?niece rdfs:label ?nieceLabel FILTER (lang(?nieceLabel) = "en")
            }}
    }}

    OPTIONAL {{ ?localNiece rdfs:seeAlso ?niece }}
    BIND(COALESCE(?localNiece, ?niece) AS ?target)
}}
```

6: Regra SPIN para a inferência de sobrinhas

Na regra 6 são inferidas relações de parentesco entre tios/tias e sobrinhas, consultando a Wikidata para identificar filhas femininas (P21 = Q6581072) dos irmãos da entidade e criando a propriedade :hasNiece entre a entidade local e as sobrinhas descobertas. A regra funciona de forma semelhante à regra dos sobrinhos, navegando através dos irmãos e identificando os seus filhos do sexo feminino para estabelecer a relação de sobrinha.

```
PREFIX : <http://localhost:8000/ontology#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
```

```
PREFIX wd: <http://www.wikidata.org/entity/>

INSERT {{
    ?localX :hasGrandfather ?target .
    ?target rdfs:label ?grandfatherLabel .
}} WHERE {{
    ?localX rdfs:seeAlso wd:{wikidata_id} .

    SERVICE <https://query.wikidata.org/sparql> {{
        wd:{wikidata_id} wdt:P22|wdt:P25 ?parent .
        ?parent wdt:P22 ?grandfather .
        OPTIONAL {{ ?grandfather rdfs:label ?grandfatherLabel FILTER (lang(?
            grandfatherLabel) = "en") }}
    }}

    OPTIONAL {{ ?localGrandfather rdfs:seeAlso ?grandfather }}
    BIND(COALESCE(?localGrandfather, ?grandfather) AS ?target)
}}
```

7: Regra SPIN para a inferência de avô

Na regra 7 são inferidas relações de parentesco entre netos e avôs, consultando a Wikidata para identificar os pais da entidade e depois os avôs paternos desses pais, criando a propriedade :hasGrandfather entre a entidade local e os avôs descobertos. A regra navega através da hierarquia familiar (filho → pai → avô) e estabelece automaticamente essas relações de avô no grafo local, dando preferência a entidades locais quando disponíveis.

```
PREFIX : <http://localhost:8000/ontology#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>

INSERT {{
    ?localX :hasGrandmother ?target .
    ?target rdfs:label ?grandmotherLabel .
}} WHERE {{
    ?localX rdfs:seeAlso wd:{wikidata_id} .

    SERVICE <https://query.wikidata.org/sparql> {{
        wd:{wikidata_id} wdt:P22|wdt:P25 ?parent .
        ?parent wdt:P25 ?grandmother .
        OPTIONAL {{ ?grandmother rdfs:label ?grandmotherLabel FILTER (lang(?
            grandmotherLabel) = "en") }}
    }}

    OPTIONAL {{ ?localGrandmother rdfs:seeAlso ?grandmother }}
    BIND(COALESCE(?localGrandmother, ?grandmother) AS ?target)
}}
```

8: Regra SPIN para a inferência de avó

Na regra 8 são inferidas relações de parentesco entre netos e avós, consultando a Wikidata para identificar os pais da entidade e depois as avós maternas desses pais (através da propriedade P25), criando a propriedade :hasGrandmother entre a entidade local e as avós descobertas. A regra navega através da hierarquia familiar (filho → pai/mãe → avó materna) e estabelece automaticamente essas relações de avó no grafo local, dando preferência a entidades locais quando disponíveis.

```
PREFIX : <http://localhost:8000/ontology#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>

INSERT {{
    ?localX :hasFatherInLaw ?target .
    ?target rdfs:label ?fatherInLawLabel .
}} WHERE {{
    ?localX rdfs:seeAlso wd:{wikidata_id} .

    SERVICE <https://query.wikidata.org/sparql> {{
        wd:{wikidata_id} wdt:P26 ?spouse .
        ?spouse wdt:P22 ?fatherInLaw .
```

```
        OPTIONAL {{ ?fatherInLaw rdfs:label ?fatherInLawLabel FILTER (lang(?
            fatherInLawLabel) = "en") }}
    }}

    OPTIONAL {{ ?localFatherInLaw rdfs:seeAlso ?fatherInLaw }}
    BIND(COALESCE(?localFatherInLaw, ?fatherInLaw) AS ?target)
}}
```

9: Regra SPIN para a inferência de genro

Na regra 9 são inferidas relações de parentesco entre genros/noras e sogros, consultando a Wikidata para identificar o cônjuge da entidade (através da propriedade P26) e depois o pai desse cônjuge (P22), criando a propriedade :hasFatherInLaw entre a entidade local e os sogros descobertos. A regra navega através da relação matrimonial e familiar (pessoa → cônjuge → pai do cônjuge) e estabelece automaticamente essas relações de sogro no grafo local, dando preferência a entidades locais quando disponíveis.

```
PREFIX : <http://localhost:8000/ontology#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>

INSERT {{
    ?localX :hasMotherInLaw ?target .
    ?target rdfs:label ?motherInLawLabel .
}} WHERE {{
    ?localX rdfs:seeAlso wd:{wikidata_id} .

    SERVICE <https://query.wikidata.org/sparql> {{
        wd:{wikidata_id} wdt:P26 ?spouse .
        ?spouse wdt:P25 ?motherInLaw .
        OPTIONAL {{ ?motherInLaw rdfs:label ?motherInLawLabel FILTER (lang(?
            motherInLawLabel) = "en") }}
    }}

    OPTIONAL {{ ?localMotherInLaw rdfs:seeAlso ?motherInLaw }}
    BIND(COALESCE(?localMotherInLaw, ?motherInLaw) AS ?target)
}}
```

10: Regra SPIN para a inferência de nora

Na regra 10 são inferidas relações de parentesco entre genros/noras e sogras, consultando a Wikidata para identificar o cônjuge da entidade (através da propriedade P26) e depois a mãe desse cônjuge (P25), criando a propriedade :hasMotherInLaw entre a entidade local e as sogras descobertas. A regra navega através da relação matrimonial e familiar (pessoa → cônjuge → mãe do cônjuge) e estabelece automaticamente essas relações de sogra no grafo local, dando preferência a entidades locais quando disponíveis.

```
PREFIX : <http://localhost:8000/ontology#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>

INSERT {{
    ?localX :hasGrandson ?target .
    ?target rdfs:label ?grandchildLabel .
}}
WHERE {{
    ?localX rdfs:seeAlso wd:{wikidata_id} .

    SERVICE <https://query.wikidata.org/sparql> {{
        wd:{wikidata_id} wdt:P40 ?child .
        ?child wdt:P40 ?grandchild .
        ?grandchild wdt:P21 wd:Q6581097 .  # male
        OPTIONAL {{ ?grandchild rdfs:label ?grandchildLabel FILTER(LANG(?
            grandchildLabel) = "en") }}
    }}

    OPTIONAL {{ ?localGrandchild rdfs:seeAlso ?grandchild }}
    BIND(COALESCE(?localGrandchild, ?grandchild) AS ?target)
}}
```

11: Regra SPIN para a inferência de neto

Na regra 11 são inferidas relações de parentesco entre avós e netos masculinos, consultando a Wikidata para identificar os filhos da entidade (através da propriedade P40) e depois os filhos desses filhos que sejam do sexo masculino (P21 = Q6581097), criando a propriedade :hasGrandson entre a entidade local e os netos descobertos. A regra navega através da hierarquia familiar descendente (avô → filho → neto masculino) e estabelece automaticamente essas relações de neto no grafo local, dando preferência a entidades locais quando disponíveis.

```
PREFIX : <http://localhost:8000/ontology#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>

INSERT {{
    ?localX :hasGranddaughter ?target .
    ?target rdfs:label ?grandchildLabel .
}}
WHERE {{
    ?localX rdfs:seeAlso wd:{wikidata_id} .

    SERVICE <https://query.wikidata.org/sparql> {{
        wd:{wikidata_id} wdt:P40 ?child .
        ?child wdt:P40 ?grandchild .
        ?grandchild wdt:P21 wd:Q6581072 .  # female
        OPTIONAL {{ ?grandchild rdfs:label ?grandchildLabel FILTER(LANG(?
            grandchildLabel) = "en") }}
    }}

    OPTIONAL {{ ?localGrandchild rdfs:seeAlso ?grandchild }}
    BIND(COALESCE(?localGrandchild, ?grandchild) AS ?target)
}}
```

12: Regra SPIN para a inferência de neta

Na regra 12 são inferidas relações de parentesco entre avós e netas femininas, consultando a Wikidata para identificar os filhos da entidade (através da propriedade P40) e depois os filhos desses filhos que sejam do sexo feminino (P21 = Q6581072), criando a propriedade :hasGranddaughter entre a entidade local e as netas descobertas. A regra navega através da hierarquia familiar descendente (avô → filho → neta feminina) e estabelece automaticamente essas relações de neta no grafo local, dando preferência a entidades locais quando disponíveis.

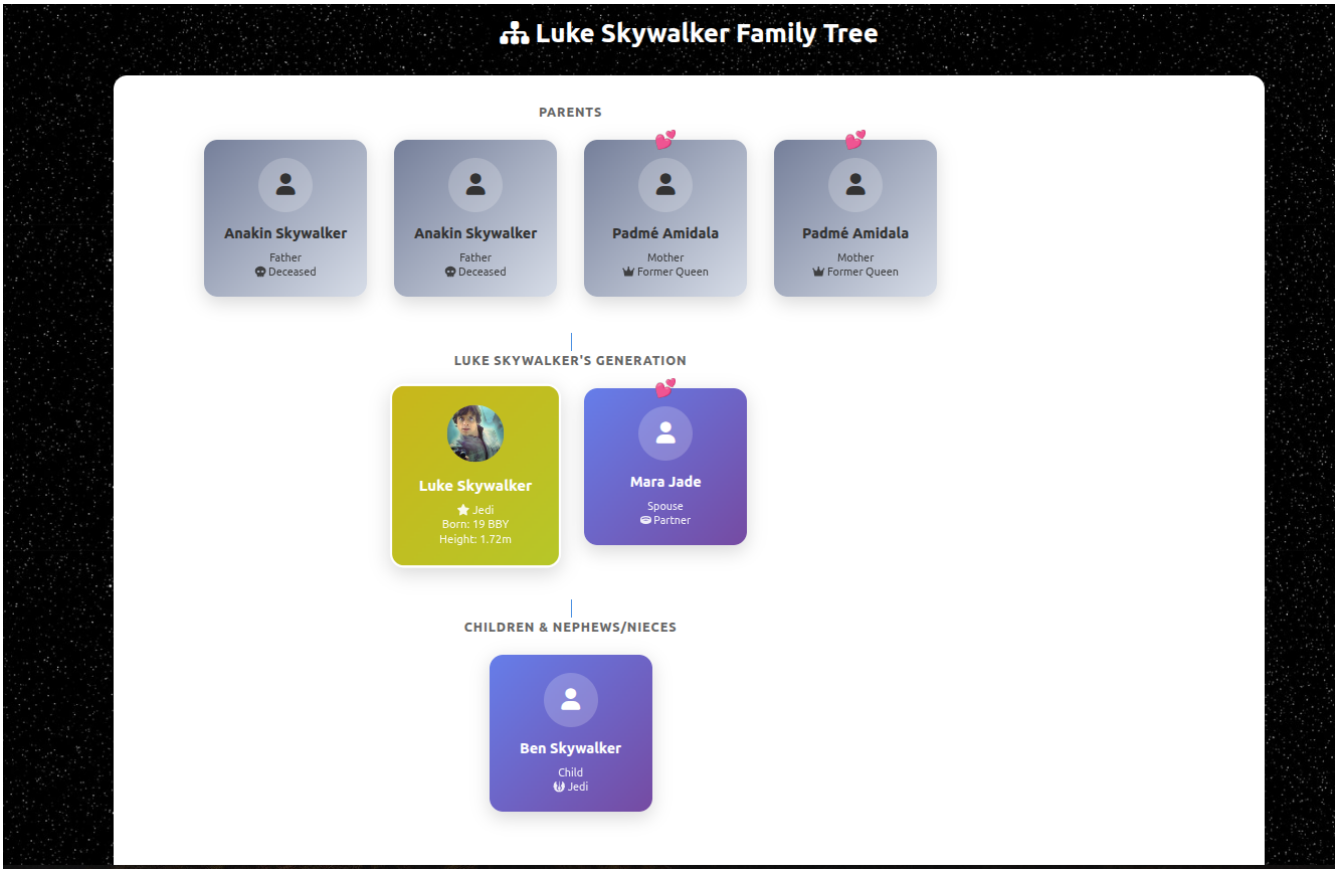


Figure 5: Árvore de família de um personagem antes das inferências

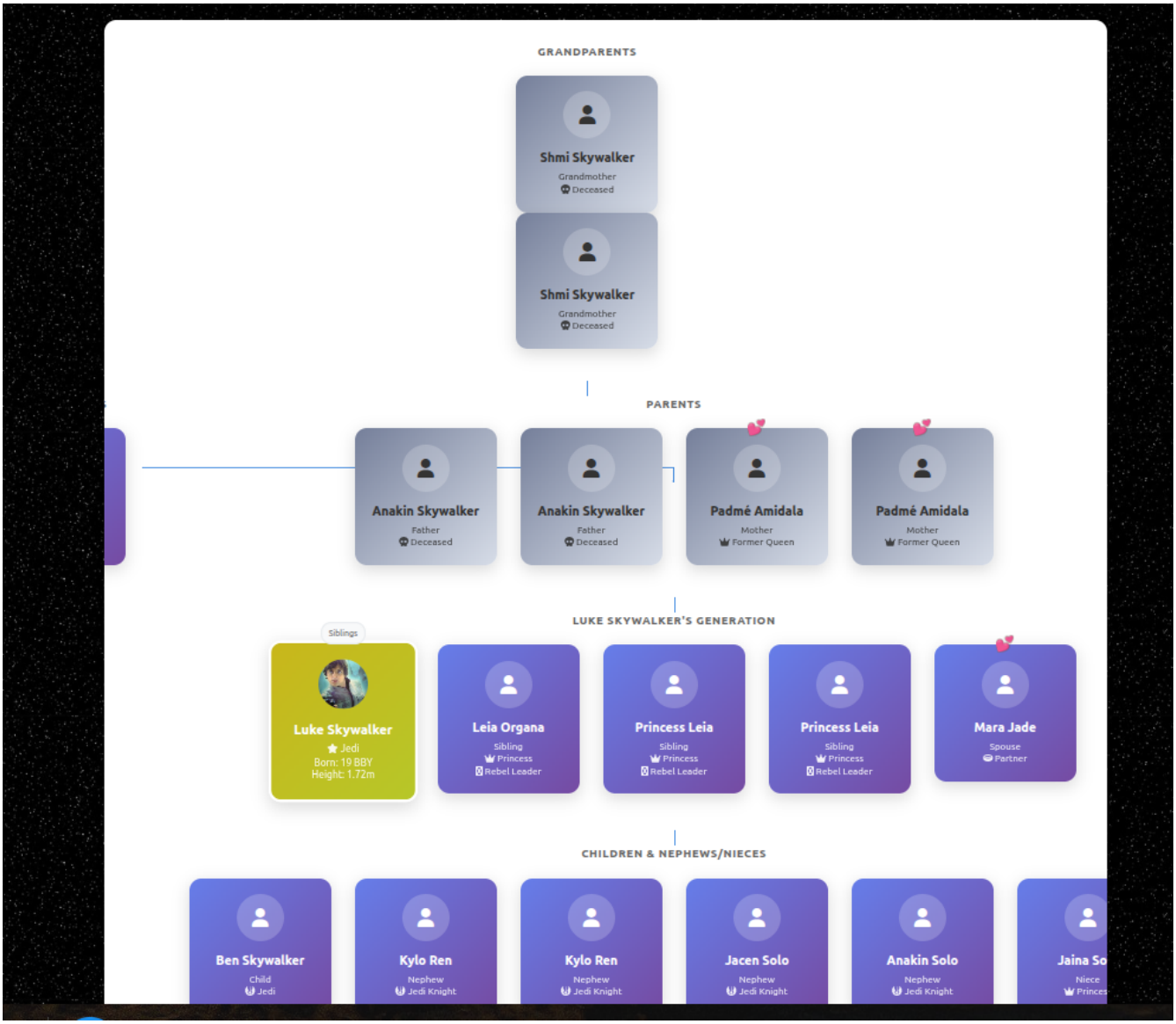


Figure 6: Árvore de família de um personagem depois das inferências

As figuras 5 e 6 ilustram o incremento dos dados obtido através da aplicação das regras SPIN.

## Regras SPIN Gerais

```
PREFIX : <http://localhost:8000/ontology#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

INSERT {{
    ?city :UrbanCenter true .
}}
WHERE {{
    ?city a :City ;
          :population ?pop .
    FILTER(xsd:integer(?pop) > 10000)
}}
```

13: Regra SPIN UrbanCenter true

Na regra 13 é inferido que se uma cidade tiver mais de 10000 população, então será adicionado um atributo :UrbanCenter com o valor true.

```
PREFIX : <http://localhost:8000/ontology#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

INSERT {{
```

```
?city :UrbanCenter false .
}}
WHERE {{
?city a :City ;
        :population ?pop .
FILTER(xsd:integer(?pop) <= 10000)
```

14: Regra SPIN UrbanCenter false

Na regra 14 é inferido que se uma cidade tiver menos de 10000 população, então será adicionado um atributo :UrbanCenter com o valor false.

```
PREFIX :     <http://localhost:8000/ontology#>
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>

INSERT {{
    ?planet :Habitable false .
}}
WHERE {{
    ?planet a :Planet .
    FILTER NOT EXISTS {{
        ?planet :population ?pop .
        FILTER(str(?pop) != "")
    }}
}}
```

15: Regra SPIN planeta não habitábel

Na regra 15 é inferido que se um planeta não tiver população, então é considerado como não habitável e será criado um atributo :Habitable com o valor false.

```
PREFIX :     <http://localhost:8000/ontology#>
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>

INSERT {{
    ?planet :Habitable true .
}}
WHERE {{
    ?planet a :Planet ;
            :population ?pop .
    FILTER(xsd:integer(?pop) > 0)
}}
```

16: Regra SPIN planeta habitábel

Na regra 15 é inferido que se um planeta tiver população, então é considerado como habitável e será criado um atributo :Habitable com o valor true.

## Importações de dados da wikidata

A importação de dados da Wikidata teve como objetivo enriquecer e completar as informações das entidades que se encontravam incompletas no dataset original. Estas entidades incompletas resultaram de terem aparecido exclusivamente como objetos em triplas RDF, nunca tendo sido descritas como sujeitos com as suas propriedades próprias. Este processo de enriquecimento foi aplicado a três tipos principais de entidades: personagens, droids e filmes, através da resolução dos respetivos identificadores Wikidata (QIDs) para obter informações detalhadas e estruturadas sobre cada uma destas entidades.

Relativamente às restantes entidades presentes no dataset, não se considerou necessário proceder ao seu enriquecimento, uma vez que os dados disponíveis na Wikidata ou coincidiam com a informação já existente, ou consistiam em propriedades irrelevantes que não acrescentavam valor significativo ao contexto e objetivos do projeto, não justificando assim a sua incorporação.

Esta inferência segue o seguinte processo:

1. Obter o URI remoto da entidade através da relação rdfs:seeAlso
2. Através da relação ont:remoteEquivalent, obter todas as propriedades locais e os seu equivalentes remotos
3. Num bloco SERVICE:
   (a) Obter o valor para cada propriedade remota

(b) Filtrar literais para manter apenas os que estejam em inglês (ou não tenham língua)

4. Se o valor remoto for um URI e existir um equivalente local, dar prioridade ao URI local (caso contrário usar o remoto)

5. Para cada uma dessas relações (que possui um valor para o dado sujeito), inserir o triplo (sujeito,propriedade local,valor local ou remoto) no grafo

```
INSERT{
    ?S ?PLocal ?V
}
WHERE {
  BIND(<http://localhost:8000/resource/general-grievous> AS ?S) #only exists because I
      had to do case-by-case (else it reached timeout)
  ?S rdfs:seeAlso ?QID .
  ?PLocal ont:remoteEquivalent ?PRemote .
  SERVICE <https://query.wikidata.org/sparql> {
    ?QID ?PRemote ?VRemote .
    FILTER(LANG(?VRemote) = "en" || !isLiteral(?VRemote)) || LANG(?VRemote) = ""
  }
  OPTIONAL {
    ?VLocal rdfs:seeAlso ?VRemote .
    FILTER(isURI(?VRemote))
  }
  BIND(COALESCE(?VLocal, ?VRemote) AS ?V)
}
```

17: Importar dados

# Funcionalidades Novas

## Novas Operações

### Detalhes da entidade (atualização)

A query para obter todos os detalhes de uma entidade (quer ela seja personagem, droid, planeta,...), foi atualizada e possui agora a seguinte funcionalidade:

1. Selecionar todos os dados locais (já presente na versão anterior)

2. Obter dados da wikidata:

   (a) Obter o uri da wikidata (se existe) através da relação rdfs:seeAlso

   (b) Dentro de um bloco SERVICE:

   - obter todos os predicados e valores remotos
   - filtrar os objetos literais para incluir os que estão em inglês (ou não possuem língua, como dados numéricos)
   - opcionalmente obter o label (em inglês) de objetos remotos

   (c) Opcionalmente obter o equivalente local aos objetos remotos (que sejam URIs) através da relação rdfs:seeAlso

   (d) Usar Bind(Coalesce()) para retornar objetos (URIs) remotos apenas se um equivalente local não existir

```
SELECT ?p ?o ?oName
WHERE{
    {
        ?uri ?p ?o .
        OPTIONAL { ?o rdfs:label ?oName . }
```

```
    }
    UNION
    {
        ?uri rdfs:seeAlso ?wikidataURI .
        SERVICE <https://query.wikidata.org/sparql> {
            ?wikidataURI ?p ?VRemote .
        FILTER(!STRSTARTS(STR(?VRemote), "http://www.wikidata.org/entity/statement/")) .
            #exclude statements
            FILTER(
                isIRI(?VRemote) ||
                LANG(?VRemote) = "en" ||
                LANG(?VRemote) = ""
            )
            OPTIONAL {
                ?VRemote rdfs:label ?oName .
                FILTER(LANG(?oName) = "en" || LANG(?oName) = "")
            }
        }
        OPTIONAL {
            ?VLocal rdfs:seeAlso ?VRemote .
            FILTER(isIRI(?VRemote))
        }
        BIND(COALESCE(?VLocal, ?VRemote) AS ?o)
    }
}
```

18: Obter detalhes da personagem

### Redirecionar para a página correta

Como o URL da página já não condiz com o URI, foi criado um novo endpoint com uma query para obter o tipo da personagem, dentre os valores desta lista:

- ont:Character
- ont:City
- ont:Droid
- ont:Film
- ont:Music
- ont:Organization
- ont:Planet
- ont:Quote
- ont:Specie
- ont:Vehicle
- ont:Starship
- ont:Weapon

Com estes valores nesta ordem, nós sabemos sempre que o último tipo (já que droids, por exemplo, também são personagens), contém a página para a qual temos de redirecionar o utilizador

```
PREFIX ont: <http://localhost:8000/ontology#>
SELECT DISTINCT ?t
WHERE{
    ?uri rdf:type ?t .
    VALUES ?t { ont:Character ont:City ont:Droid ont:Film ont:Music ont:Organization ont
        :Planet ont:Quote ont:Specie ont:Vehicle ont:Starship ont:Weapon } .
}
```

19: Obter o tipo da entidade

As queries utilizadas nas operações de adicionar, editar e apagar um personagem foram adaptadas para serem compatíveis com a ontologia.

# Integração com dados remotos

## Wikidata

De modo a utilizar dados da wikidata, foi necessário primeiro fazer a reconciliação (obter os URIs da wikidata a partir de dados locais). Contudo, devido ao tamanho do dataset (um total de 500 entidades e pelo menos 100 personagens), fomos forçados a utilizar a biblioteca reconciler para fazer a reconciliação automática de entidades e conseguimos reconciliar personagens, droids e filmes.

Da wikidata, nós utilizámos principalmente as seguintes propriedades:

- Personagem
  - Imagem
  - Local de morte
  - Maneira como morreu
  - Responsável pela morto (morto por)
  - Últimas palavras
  - Inimigos
  - Ocupações
  - Graus de parentesco (pai, filho,...) (também usado para inferências)
  - Local de nascimento
  - Students(personagens dos quais ele foi professor)
  - Jedi Masters( mestres/professores dos presonagens)
- Droids
  - Imagem
- Filme
  - Género
  - Data de publicação
  - País de Origem
  - Linguagem orignal do filme
  - Roteirista
  - Membros do elenco(atores)
  - Dubladores
  - Editores do filme
  - Desginer de produção
  - Figurinista(pessoa que escolhe as roupas das personagens)
  - Compositor
  - Produtora
  - Empresas que distribuiram o filme
  - Locais fictícios onde se desenrola a ação do filme
  - Localizações reais onde o filme foi gravado
  - Duração do filme
  - Aspect Ratio
  - Cores(se o filme é a cores ou não)
  - Reviews
  - Prémios recebidos
  - Nomiações para prémios
  - Budget usado para fazer o filme
  - Personagens presentes no filme
  - Rating
  - Tópicos do filme

### DBPedia

Apesar de termos considerado incluir dados da dbpédia para complementar os da wikidata, não só o serviço de "search" deles (com exceção dos holandeses) estava em baixo durante o desenvolvimento do projeto (o que nos forçaria a reconciliar manualmente ou via construção de uri), mas também não possui nada que a wikidata já não tinha (e o que tinha estava altamente destruturado), levando à nossa decisão de excluir a DBPedia do nosso projeto.

### Star wars Ontology

Como menção honrosa, também destaco uma ontologia criada por um fã (link), que foi considerada para integração (mas decidimos não o fazer por causa da difilculdade em reconciliar)

## Publicação de dados semânticos

### RDFa

Os dados semânticos das páginas de detalhes de cada tipo de sujeito foram publicados com RDFa (incluindo dados externos da wikipédia), sendo incluídos não só os valores de cada propriedade e relação mas também, no caso das relações, o label de cada URI.

Contudo, apesar de inicialmente termos querido usar um schema standard como o de schema.org para facilitar o processamento dos nossos dados, o facto de muitos dos nossos dados não possuirem equivalentes "standard" levou-nos a usar os nomes internos dos predicados (incluindo os vindos da wikidata), de modo a manter a consistência.

Estes dados foram depois testados usando um addon de firefox chamado Open Link Structured Data Snifer.

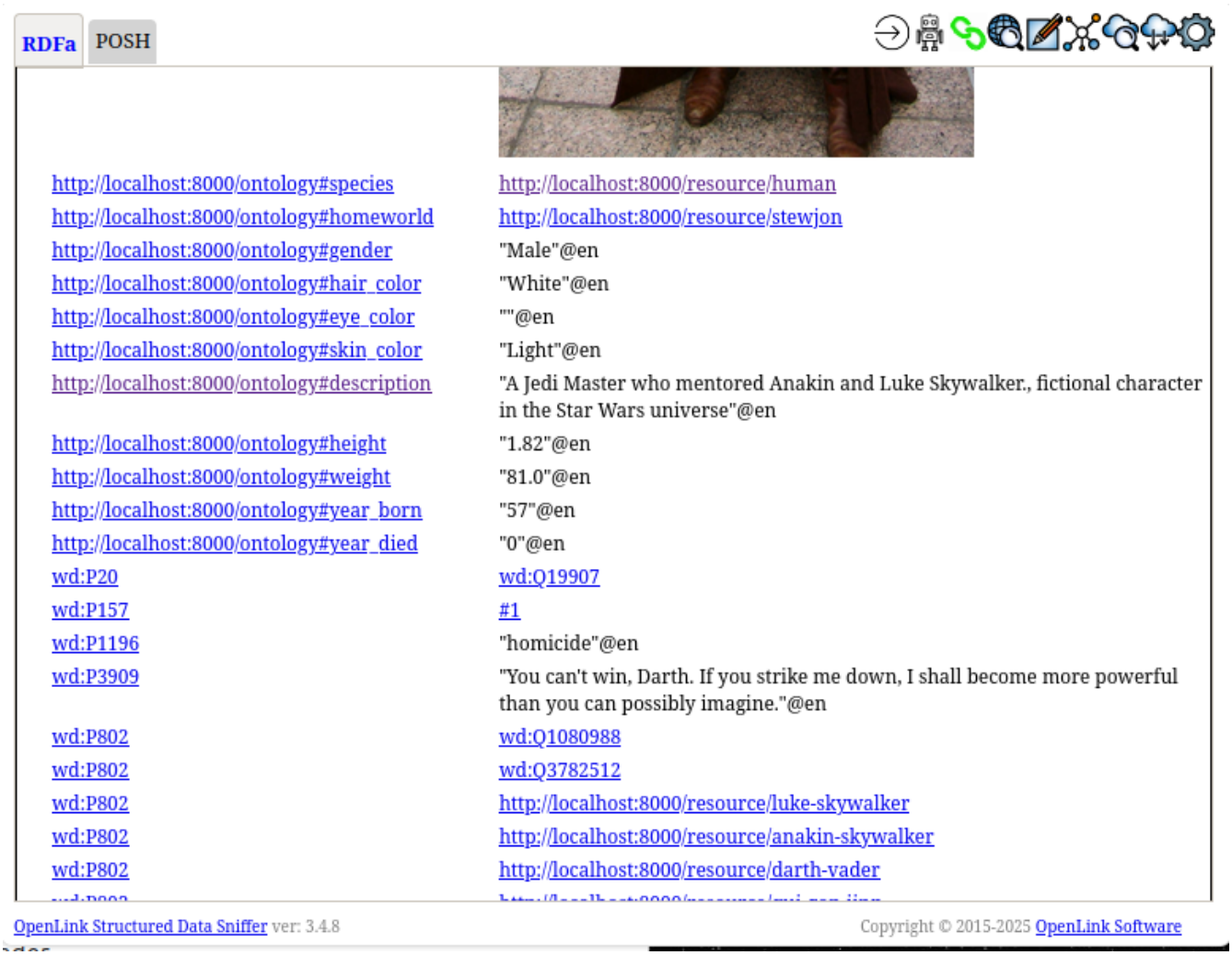


Figure 7: Exemplo de parsing por RDFa

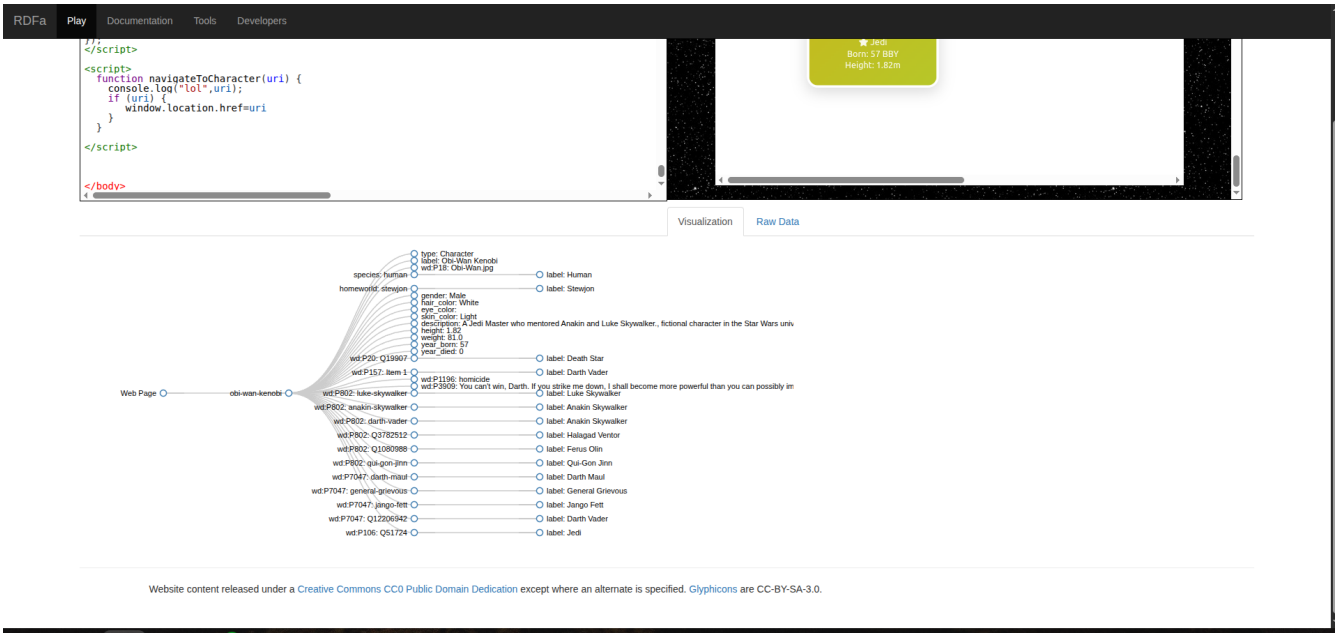


Figure 8: Teste rdfa

## Microformatos

Em termos de microformatos, devido à natureza ficcional dos dados (e à estrutura rígida dos microformatos) só conseguimos publicar um h-card com o nome e foto de cada personagem (todas as outras entidades e/ou propriedades não tinham um equivalente adequado em microformatos)

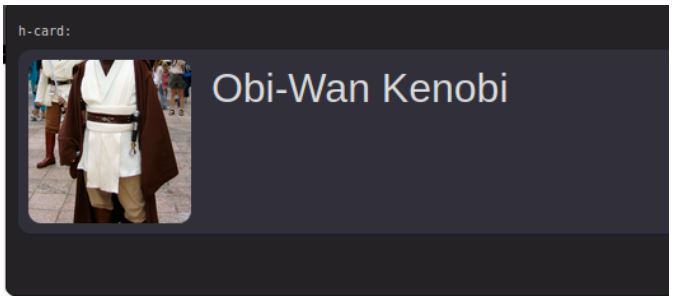


Figure 9: Exemplo de parsing de microformatos

Usando o parser de microformatos https://pin13.net/mf2/, obtivemos o seguinte json para a página de detalhes de um personagem:


```
{
    "items": [
        {
            "type": [
                "h-card"
            ],
            "properties": {
                "name": [
                    "Obi-Wan␣Kenobi"
                ],
                "photo": [
                    {
                        "value": "http://commons.wikimedia.org/wiki/Special:FilePath/Obi
                            -Wan.jpg",
                        "alt": "Obi-Wan␣Kenobi"
                    }
                ]
            },
            "lang": "en"
        }
```

```
],
"rels": {
    "stylesheet": [
        "https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css",
        "https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css",
        "https://www.fontlibrary.org/face/star-jedi"
    ],
    "icon": [
        "/static/favicon.ico"
    ],
    "species": [
        "http://localhost:8000/resource/human"
    ],
    "homeworld": [
        "http://localhost:8000/resource/stewjon"
    ],
    "wd:P20": [
        "http://www.wikidata.org/entity/Q19907"
    ],
    "wd:P802": [
        "http://localhost:8000/resource/luke-skywalker",
        "http://localhost:8000/resource/anakin-skywalker",
        "http://localhost:8000/resource/darth-vader",
        "http://www.wikidata.org/entity/Q3782512",
        "http://www.wikidata.org/entity/Q1080988",
        "http://localhost:8000/resource/qui-gon-jinn"
    ],
    "wd:P7047": [
        "http://localhost:8000/resource/darth-maul",
        "http://localhost:8000/resource/general-grievous",
        "http://localhost:8000/resource/jango-fett",
        "http://www.wikidata.org/entity/Q12206942"
    ],
    "wd:P106": [
        "http://www.wikidata.org/entity/Q51724"
    ]
},
"rel-urls": {
    "https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css": {
        "rels": [
            "stylesheet"
        ]
    },
    "/static/favicon.ico": {
        "type": "image/x-icon",
        "rels": [
            "icon"
        ]
    },
    "https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css": {
        "rels": [
            "stylesheet"
        ]
    },
    "https://www.fontlibrary.org/face/star-jedi": {
        "rels": [
            "stylesheet"
        ]
    },
    "http://localhost:8000/resource/human": {
        "text": "Human",
        "rels": [
            "species"
        ]
    },
    "http://localhost:8000/resource/stewjon": {
        "text": "Stewjon",
```

```
        "rels": [
            "homeworld"
        ]
    },
    "http://www.wikidata.org/entity/Q19907": {
        "text": "Death␣Star",
        "rels": [
            "wd:P20"
        ]
    },
    "http://localhost:8000/resource/luke-skywalker": {
        "text": "Luke␣Skywalker",
        "rels": [
            "wd:P802"
        ]
    },
    "http://localhost:8000/resource/anakin-skywalker": {
        "text": "Anakin␣Skywalker",
        "rels": [
            "wd:P802"
        ]
    },
    "http://localhost:8000/resource/darth-vader": {
        "text": "Anakin␣Skywalker",
        "rels": [
            "wd:P802"
        ]
    },
    "http://www.wikidata.org/entity/Q3782512": {
        "text": "Halagad␣Ventor",
        "rels": [
            "wd:P802"
        ]
    },
    "http://www.wikidata.org/entity/Q1080988": {
        "text": "Ferus␣Olin",
        "rels": [
            "wd:P802"
        ]
    },
    "http://localhost:8000/resource/qui-gon-jinn": {
        "text": "Qui-Gon␣Jinn",
        "rels": [
            "wd:P802"
        ]
    },
    "http://localhost:8000/resource/darth-maul": {
        "text": "Darth␣Maul",
        "rels": [
            "wd:P7047"
        ]
    },
    "http://localhost:8000/resource/general-grievous": {
        "text": "General␣Grievous",
        "rels": [
            "wd:P7047"
        ]
    },
    "http://localhost:8000/resource/jango-fett": {
        "text": "Jango␣Fett",
        "rels": [
            "wd:P7047"
        ]
    },
    "http://www.wikidata.org/entity/Q12206942": {
        "text": "Darth␣Vader",
        "rels": [
```

```
                "wd:P7047"
            ]
        },
        "http://www.wikidata.org/entity/Q51724": {
            "text": "Jedi",
            "rels": [
                "wd:P106"
            ]
        }
    },
    "debug": {
        "package": "https://packagist.org/packages/mf2/mf2",
        "source": "https://github.com/indieweb/php-mf2",
        "version": "v0.5.0",
        "note": [
            "This output was generated from the php-mf2 library available at https://
               github.com/indieweb/php-mf2",
            "Please file any issues with the parser at https://github.com/indieweb/php-
               mf2/issues",
            "Using the Masterminds HTML5 parser"
        ]
    }
}
```


## Mudanças de UI

Em termos de UI, uma das mudanças principais causadas pela ontologia foi a criação do endpoint "/resource/...". Este endpoint serve para resolução de URIs, obtendo a classe "principal" (Character,Droid,...) e redirecionando para o endpoint correto. Adicionalmente foi criada uma página para as entidades sem uma classe atribuída para importar dados da "wikidata" e uma lista dos atributos encontrados (para efeitos de debugging).

Além de uma melhoria visual abrangente em todas as páginas, algumas páginas de detalhes específicas foram substancialmente reformuladas para incorporar dados externos provenientes do Wikidata.

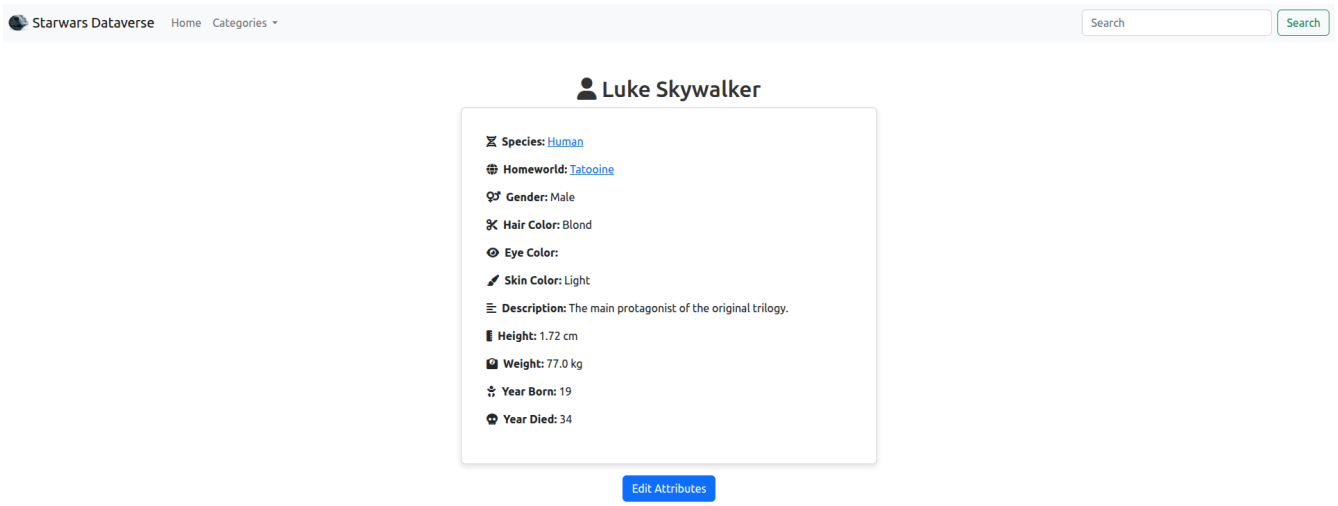


Figure 10: Página detalhes de um personagem do primeiro trabalho


Figure 11: Página detalhes de um filme do primeiro trabalho

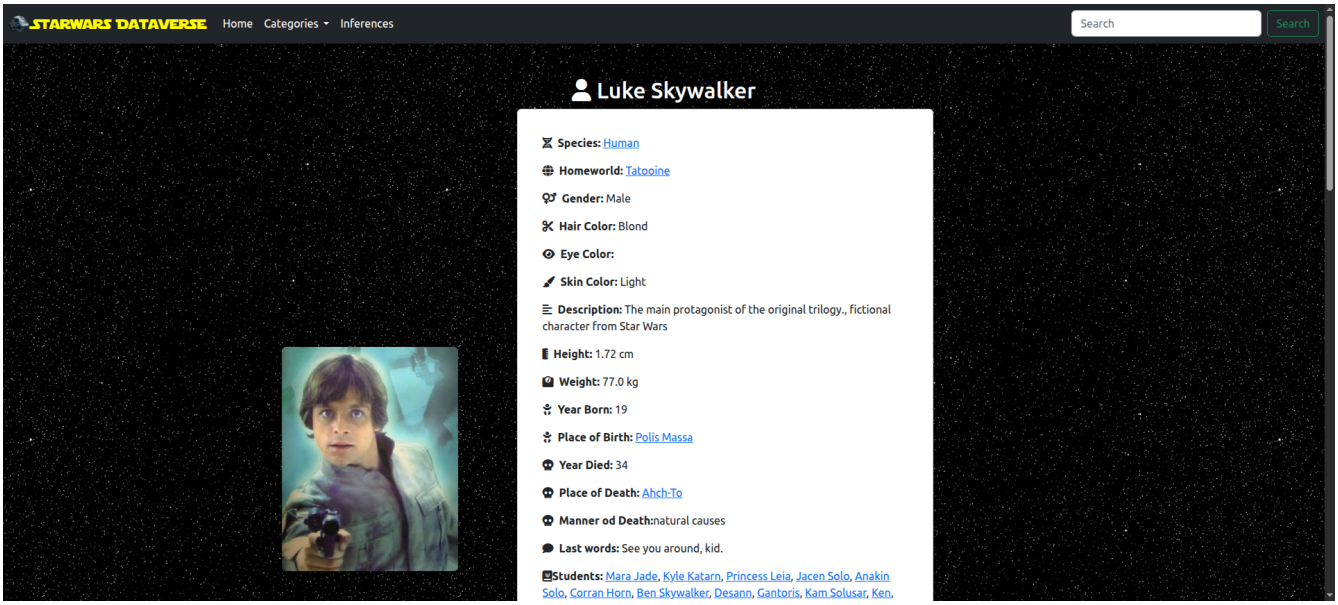


Figure 12: Página de detalhes de um personagem neste trabalho

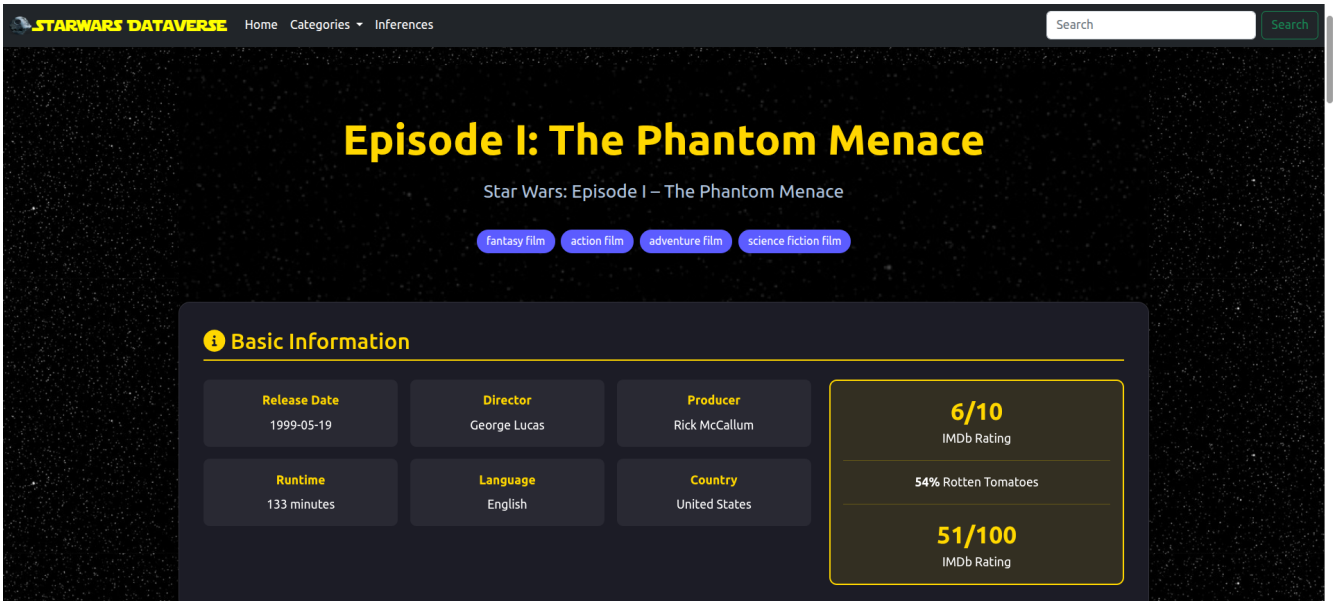


Figure 13: Página detalhes de um filme neste trabalho

Comparando a figura 10 e a figura 12, podemos observar as diferenças entre a página de detalhes de um personagem no primeiro trabalho e neste trabalho.

Comparando a figura 11 e a figura 12, podemos observar as diferenças entre a página de detalhes de um filme no primeiro trabalho e neste trabalho.

Aconselha-se a análise de todas as páginas da aplicação, particularmente as duas páginas referidas a cima, de forma a avaliar adequadamente as modificações efetuadas

# Conclusões

Este projeto demonstrou com sucesso a aplicação prática dos conceitos fundamentais da Web Semântica no desenvolvimento de um sistema de informação abrangente sobre o universo Star Wars. A ontologia desenvolvida em Protégé revelou-se robusta e expressiva, capturando eficazmente a complexidade deste domínio através de uma hierarquia bem estruturada de classes, propriedades e restrições semânticas. As regras SPIN implementadas permitiram a derivação automática de relações familiares complexas e a inferência de características contextuais, enquanto a integração bem-sucedida com a Wikidata enriqueceu substancialmente o conteúdo do sistema.

O sistema transcende uma simples base de dados, oferecendo uma experiência semântica rica que permite descobrir relações não explícitas entre entidades. As funcionalidades de inferência automática revelam conexões familiares complexas que não estavam originalmente presentes nos dados base, demonstrando o valor acrescentado das tecnologias semânticas. A publicação de dados através de RDFa e microformatos torna o sistema interoperável com outras aplicações da Web Semântica, contribuindo para o ecossistema mais amplo de dados ligados.

Embora tenham sido identificados alguns desafios técnicos relacionados com a performance de consultas e limitações dos motores de inferência, estas foram endereçadas através de soluções alternativas eficazes. O projeto estabelece uma base sólida para desenvolvimentos futuros e demonstra o potencial das tecnologias da Web Semântica na criação de sistemas de conhecimento ricos e inteligentes, cumprindo integralmente os seus objetivos

ao resultar numa aplicação funcional e semanticamente rica que exemplifica as melhores práticas da área.

# Executar a Aplicação

Como executar a aplicação:

- Ter uma instância de graphdb (local ou num container) aberta no port 7200
- Criar um repositório com o nome "starwars" em owl-horst
- Importar os ficheiros de ontologia e dados
- Iniciar o servidor de django

Como executar o ficheiro de conversão (./graphdb/import/csv_to_rdf.py):

- Criar um environment
- Instalar requirements (pip install -r requirements.txt) (Nota: o ficheiro encontra-se na root do projeto)
- Ir à pasta contendo o script (cd ./graphdb/import) e correr (*python*3 *csv_to_rdf.py* ou *python*3 *csv_to_rdf.py*, dependendo da instalação)

Link Github:https://github.com/DiogoSilva1904/WS_P2