



universidade
de aveiro

Web Semântica

Diogo Silva
NºMec: 107647

Bruno Páscoa
NºMec: 107418

Mestrado em Engenharia Informática, 2024/2025

Índice

Introdução ao Tema	2
Dados, fonte e transformações	2
Descrição do dataset	2
Preparação dos dados	3
Decisões tomadas	3
Operações sobre os dados	3
Pesquisa/Navegação por grafo	3
Listar sujeitos com base no tipo	4
Detalhes do sujeito	4
Inserir ou atualizar personagem	4
Remover sujeito	5
Funcionalidades da Aplicação	5
Página Inicial	5
Página de listagem de entidades com duas vistas	6
Página de Detalhes	8
Página de Search	10
Navbar	10
Grafo Interativo	10
Conclusões	11
Executar a Aplicação	11

Introdução ao Tema

O universo de Star Wars é um dos mais icônicos 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.

Dados, fonte e transformações

Descrição do dataset

Os dados necessários para a realização deste trabalho foram obtidos do Kaggle e seguem a estrutura a seguir apresentada. Para mais detalhes, recomenda-se consultar a fonte original.

- **battles.csv** (não utilizado): possui id, nome, localização, data, resultado (quem venceu) e participantes.
- **characters.csv**: possui id, nome, espécie, género, altura, peso, cor de cabelo, cor de olhos, cor da pele, ano de nascimento e ano de morte.
- **cities.csv**: possui id, nome, planeta onde se localiza e uma descrição
- **droids.csv**: possui id, nome, modelo, fabricante, altura, peso, cor do sensor, cor do corpo, função primário e os filmes em que apareceram.
- **events.csv** (não utilizado): possui id, nome, data, localização e descrição
- **films.csv**: possui id, título, data de lançamento, diretor(es), produtor(es), e o "texto de abertura"
- **music.csv**: possui id, título, compositor, data de lançamento e o filme em que aparece
- **organizations.csv**: possui id, nome, datas de fundação e dissolução, líder(es), membros, afiliação, descrição e os filmes em que apareceram.
- **planets.csv**: possui id, nome, diâmetro, período de rotação, período de órbita, gravidade (em comparação com Coruscant), população, clima, terreno, percentagem de água à superfície, residentes e os filmes em que apareceram.
- **quotes.csv**: possui id, o nome da personagem que a disse, a própria citação e o filme em que apareceu.
- **species.csv**: possui id, nome, classificação (mamífero, réptil,...), designação (racional ou irracional), altura e pesos médios, cores da pele, olhos e cabelos, esperança média de vida, língua e o planeta de origem.
- **starships.csv**: possui id, nome, modelo, fabricante, custo (em créditos), comprimento, velocidade máxima (com atmosfera), número de tripulantes e passageiros, capacidade de carga, duração das rações, classificação de "hyperdrive", MGLT (medida de velocidade deste universo), classe, pilotos e filmes em que apareceram.

- **timeline.csv** (não utilizado): possui id, uma descrição do evento e o ano em que decorreu
- **vehicles.csv**: possui id, nome, modelo, fabricante, custo (em créditos), comprimento, velocidade máxima (com atmosfera), número de tripulantes e passageiros, capacidade de carga, duração das rações, classe e filmes em que apareceram.
- **weapons.csv**: possui id, nome, modelo, fabricante, custo (em créditos), comprimento, tipo de arma e descrição.

Preparação dos dados

De modo a converter os vários ficheiros CSV num único ficheiro RDF, criámos um script (`graphdb/import/csv_to_rdf.py`) que lê cada um dos ficheiros, converte as "células" em triplos e depois adiciona-os a um grafo (da biblioteca `RDFLib`), que depois convertemos num ficheiro RDF/XML. Durante esta conversão, seguimos as seguintes "regras":

1. Os URIs são obtidos com base no título/nome ("slugified" para garantir a formatação correta), com exceção das citações (que utilizam ids devido ao tamanho do label)
2. Se um valor for inválido (`None`, "None" ou "Unknown", por exemplo) ele não é adicionado
3. Se uma "célula" possui múltiplos valores, cada valor é adicionado individualmente
4. Se um objeto for de um dos tipos presentes nos CSVs (personagem, cidade,...), este é referenciado pelo seu URI (juntamente com o seu label e tipo).

Decisões tomadas

Durante a transformação do dataset, foram tomadas as seguintes decisões (que foram depois mantidas durante a duração do trabalho):

- Todos os URIs presentes no grafo (quer eles sejam sujeitos ou objetos) têm de possuir um tipo e um label (de modo a permitir uma visualização correta). Isto aplica-se mesmo no ato de remoção.
- Os ficheiros "timeline.csv", "events.csv" e "battles.csv" foram descartados devido a estarem altamente incompletos (adicionalmente, algumas colunas de "battles.csv" possuem tipos ambíguos).
- Por uma questão de conveniência, a maioria dos predicados não foram alterados (com algumas exceções) e foi criado um namespace para cada tipo (para além do namespace "global" `sw`).
- De modo a facilitar a resolução de URIs, estes condizem com os URLs do servidor.

Operações sobre os dados

As operações sobre os dados são realizadas garantindo a segurança e a integridade das consultas. Para leitura, utilizam-se queries parametrizadas, protegendo contra RDF Injection. No entanto, devido a limitações da `SparqlUpdateStore` — um wrapper do `RdfLib` que permite interagir com um grafo remoto como se fosse local —, as operações de escrita recorrem à biblioteca `requests`, que não proporciona o mesmo nível de proteção.

Pesquisa/Navegação por grafo

Resolução direta de URI (caso o input seja o URI de um sujeito ou o seu nome)

```
SELECT DISTINCT ?s ?p ?o ?sName
WHERE {
    ?s ?p ?o .
    FILTER((?s = ?q && ?p = rdfs:label) || (?p = rdf:type && ?o = ?q))
    ?s rdfs:label ?sName .
}
```

Listar todos os resultados da pesquisa (caso o input seja parte de um objeto)

```
SELECT DISTINCT ?s ?sName ?p
  WHERE {
    ?s ?p ?o .
    FILTER (regex(?o,?q,"i"))
    ?s rdfs:label ?sName .
  }
```

Ambas as queries podem necessitar de utilizar:

```
OPTIONAL { ?s ?p ?o . }
```

Isto ocorre porque algumas entidades podem não possuir certos atributos que a maioria das entidades do mesmo tipo possuem.

Listar sujeitos com base no tipo

Vista de grafo:

```
CONSTRUCT {
  ?s rdf:type ?type ;
  ?p ?o .
}WHERE{
  ?s rdf:type ?type ;
  ?p ?o .
}
```

Vista de lista:

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

Detalhes do sujeito

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

Inserir ou atualizar personagem

Inserir label e tipo:

```
INSERT DATA {
  ?uri rdfs:label ?label ;
  rdf:type sw:Character .
}
```

Inserir objetos literais:

```
INSERT DATA {
  ?uri ?attribute ?value .
}
```

Inserir espécie:

```
INSERT DATA {
  ?character_uri sw:specie ?specie_uri .
  ?specie_uri rdf:type sw:Specie ;
  rdfs:label ?specie .
}
```

Inserir planeta de nascimento:

```

INSERT DATA {
    ?character_uri sw:homeworld ?homeworld_uri .
    ?homeworld_uri rdf:type sw:Planet ;
    rdfs:label ?homeworld .
}

```

Remover sujeito

Verificar URI é objeto de outro sujeito (caso seja, não podemos remover label ou tipo):

```
ASK { ?s ?p ?uri . }
```

Remoção parcial de sujeito:

```

DELETE WHERE {
    ?uri ?p ?o .
    FILTER(?p != rdf:type && ?p != rdfs:label)
}

```

Remoção total de sujeito (também usado em update):

```

DELETE {
    ?uri ?p ?o .
} WHERE {
    ?uri ?p ?o .
}

```

Embora apenas tenhamos implementado as operações de remoção, adição e edição para entidades do tipo personagem, os queries podem ser aplicados a qualquer entidade. No entanto, a operação de edição precisaria de algumas adaptações, dependendo dos atributos específicos de cada entidade.

Funcionalidades da Aplicação

A nossa aplicação disponibiliza várias páginas, cada uma projetada para oferecer funcionalidades específicas, garantindo uma experiência que permite aos utilizadores explorar e modificar dados com facilidade. A seguir, apresentamos uma visão geral das páginas e das suas principais funcionalidades.

- **Página Inicial:** Apresenta um grafo geral interativo.
- **Página de listagem de entidades com duas vistas:**
 - Lista com as entidades de um determinado tipo.
 - Conjunto das entidades do mesmo tipo, representado por um grafo interativo.
- **Página de Detalhes:** Exibe informações detalhadas sobre cada entidade.
- **Página de Pesquisa:** Exibe os resultados de uma pesquisa por um termo específico em todos os dados.

Página Inicial

Na página inicial, há uma janela que exibe um grafo geral dos dados, permitindo interação através de consultas e filtragem. Todos os nós do grafo são clicáveis, possibilitando a navegação para a página de detalhes da entidade ou para a página de pesquisa sobre a entidade.

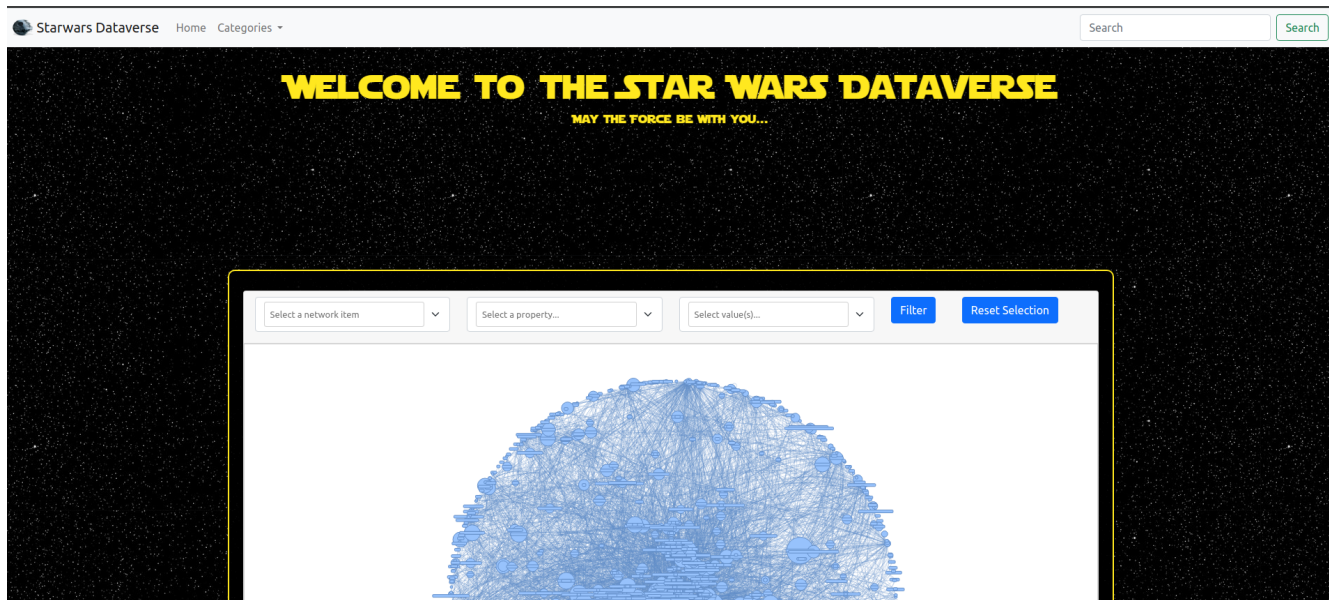


Figure 1: Página Inicial

Página de listagem de entidades com duas vistas

Nesta página, os utilizadores podem alternar entre duas vistas. A primeira apresenta uma lista das entidades de um determinado tipo, enquanto a segunda exibe essas mesmas entidades em um grafo interativo.

Na primeira visualização, o utilizador pode gerir as entidades existentes. Para remover uma entidade, basta clicar no ícone vermelho presente em cada linha da tabela. Além disso, é possível adicionar uma nova entidade através do botão localizado no canto superior direito, acima da tabela. Ao clicar nesse botão, um modal será exibido, contendo um formulário onde o utilizador poderá preencher os dados necessários para a criação da nova entidade.

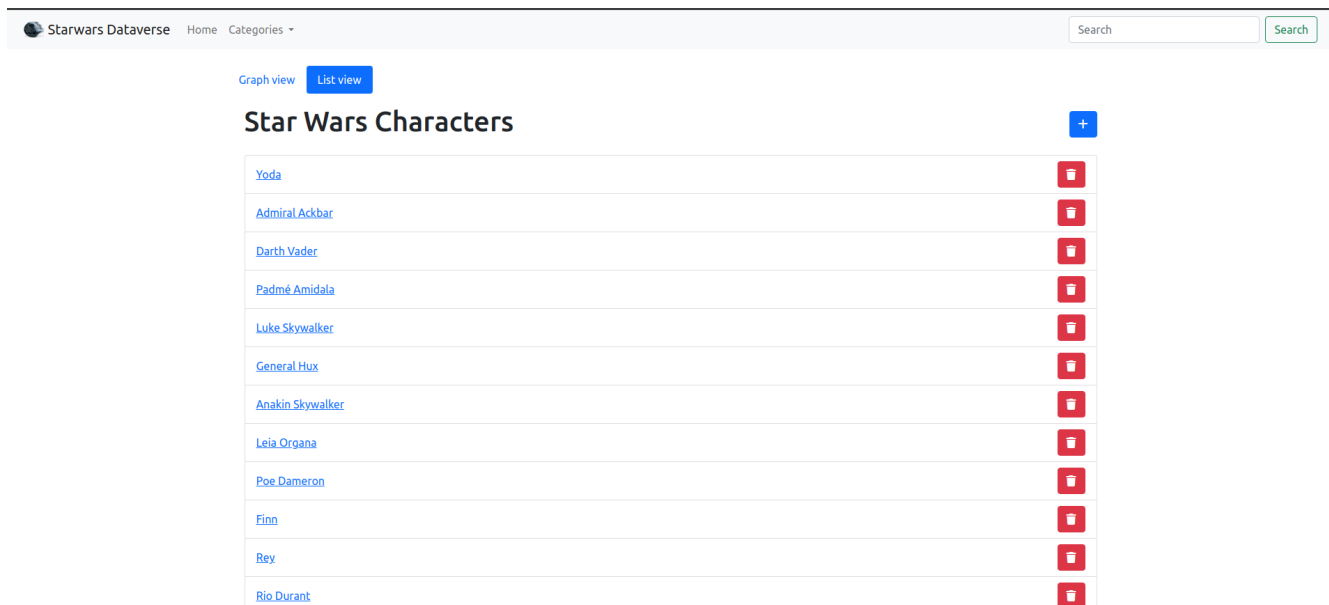
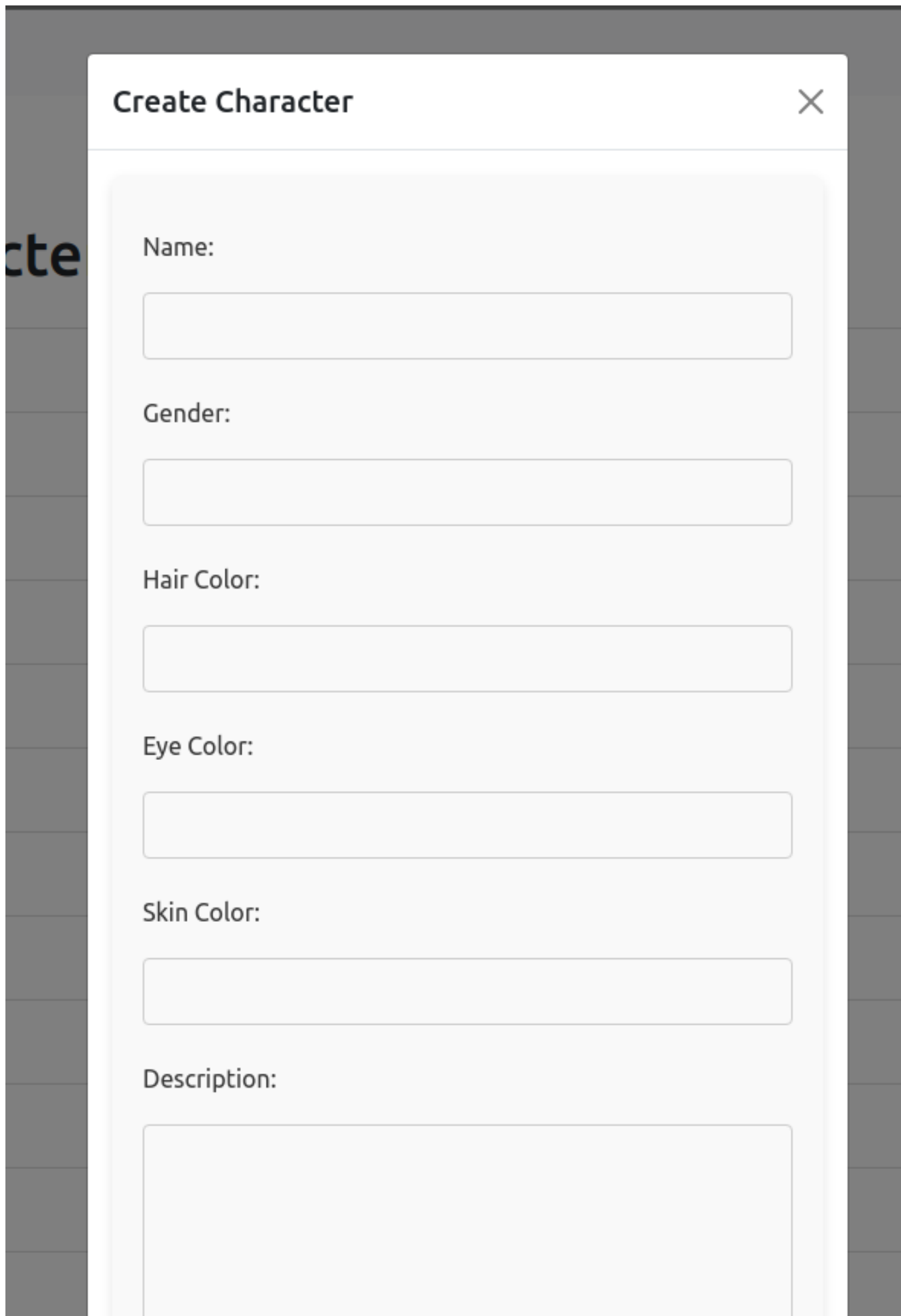


Figure 2: Visualização em forma de lista

A screenshot of a 'Create Character' form. The form is a white rectangular box with a dark gray border. At the top left, the title 'Create Character' is written in a bold, dark gray font. At the top right, there is a dark gray 'X' icon for closing the form. The form contains six input fields, each preceded by a label in a bold, dark gray font. The labels are 'Name:', 'Gender:', 'Hair Color:', 'Eye Color:', 'Skin Color:', and 'Description:'. The first five labels are followed by single-line text input fields, while the 'Description:' label is followed by a larger, multi-line text area. The background of the form is a light gray color.

Create Character ✕

Name:

Gender:

Hair Color:

Eye Color:

Skin Color:

Description:

Figure 3: Formulário de criação de personagem

Starwars Datasverse Home Categories ▾

Search

Graph view List view

Select a network item ▾ Select a property... ▾ Select value(s)... ▾ Filter Reset Selection

General Grievous
Luthe Rael
Admiral Ackbar
A Jedi Knight during the Clone Wars.
A Jedi Master and member of the Jedi High Council.
A member of the Crimson Dawn syndicate.
Dash Rendar
Nute Gunray
Din Djarin
kamino
1.7
22
50.0
55.0
White
3
15

Página de Detalhes

Starwars Daterverse

HomeCategories

Search

Search

Luke Skywalker

Species: [Human](#)

Homeworld: [Tatooine](#)

Gender: Male

Hair Color: Blond

Eye Color:

Skin Color: Light

Description: The main protagonist of the original trilogy.

Height: 1.72 cm

Weight: 77.0 kg

Year Born: 19

Year Died: 34

Edit Attributes

8

Edit Character



Name:

Luke Skywalker

Gender:

Male

Hair Color:

Blond

Eye Color:

Blue

Skin Color:

Light

Description:

The main protagonist of the original trilogy.

Figure 6: Formulário de Edição de um personagem

Página de Search

Após realizar uma pesquisa por um determinado termo, esta página é carregada, exibindo todos os resultados encontrados nos dados. Para cada pesquisa, os resultados são apresentados numa lista, permitindo ao utilizador clicar para aceder à entidade ou atributo onde o termo foi referenciado.

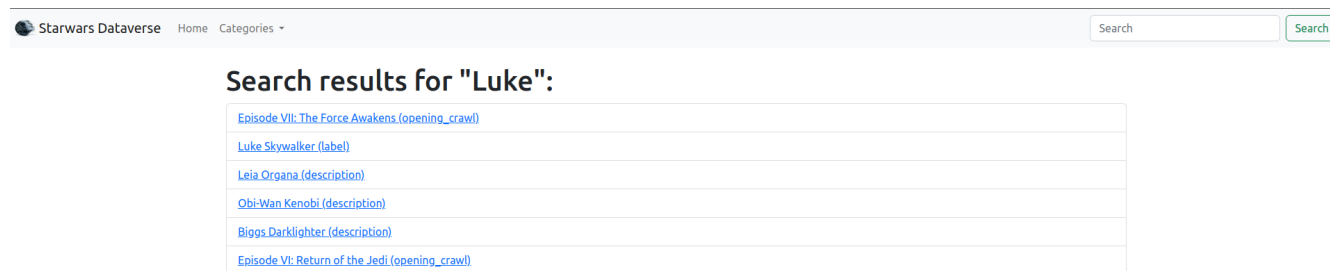


Figure 7: Página de Pesquisa

Navbar

A navegação entre as páginas é feita por meio de uma navbar, que permite acessar à página inicial, à página com as 2 visualizações para cada tipo de entidade e utilizar uma barra de pesquisa para procurar por um termo específico.



Figure 8: Navbar

Grafos Interativos

Como referido anteriormente, os utilizadores podem interagir com os grafos disponíveis. Neles, é possível filtrar por atributos e relações, além de clicar nas entidades (nós) para visualizar detalhes específicos de cada uma.

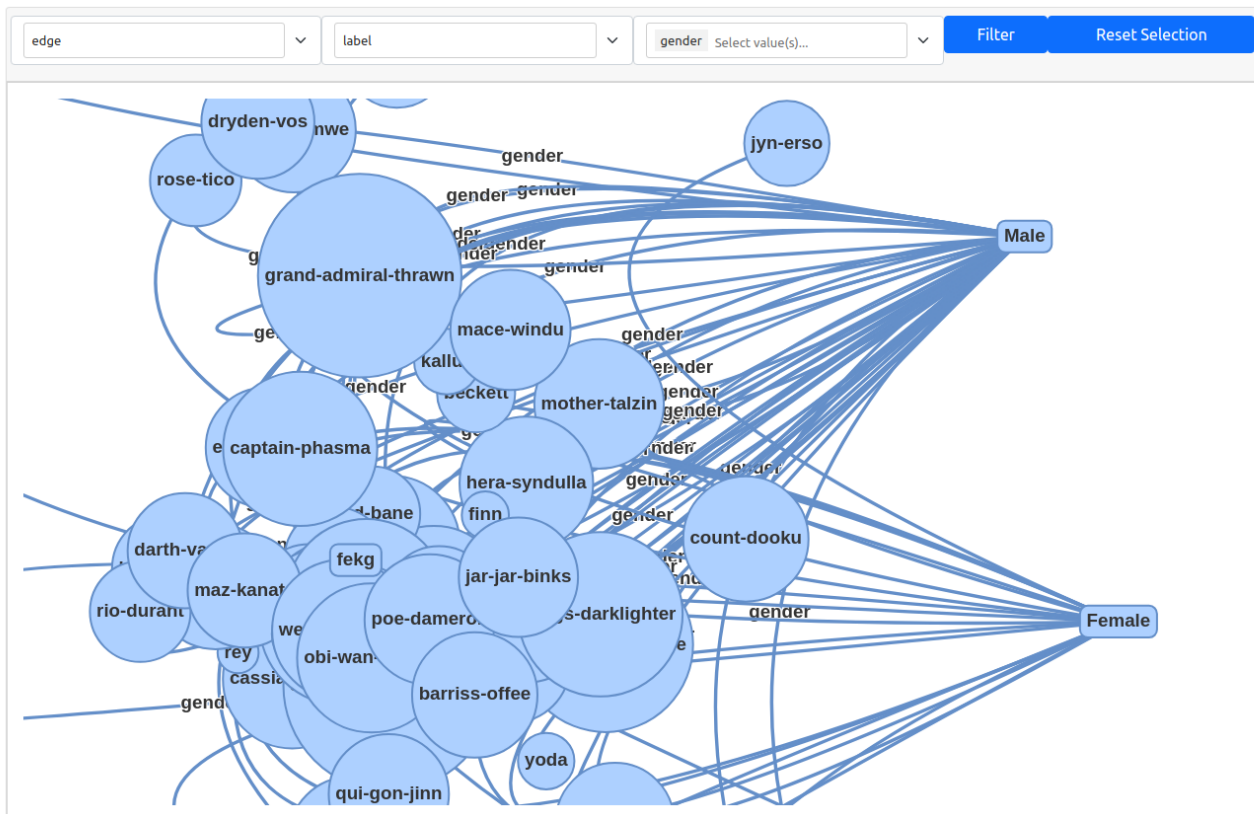


Figure 9: Pesquisa pelo grafo

Conclusões

Este projeto possibilitou a exploração da integração entre Django, GraphDB e dados no formato RDF, proporcionando uma forma mais eficiente de estruturar e interligar dados. A utilização do SPARQL facilitou a realização de consultas e modificações, destacando o potencial das redes de dados na gestão e análise da informação.

Executar a Aplicação

Para facilitar a execução do projeto, foi criado um arquivo ‘docker-compose.yaml’, que configura dois containers:

- Um container para a aplicação web.
- Um container para o GraphDB.

Para executar a aplicação, basta ter o Docker instalado no computador e executar o seguinte comando no terminal:

```
docker compose up
```

Para importar os dados, acesse o GraphDB Workbench, crie um repositório chamado ‘starwars’ (caso ainda não exista) e importe o ficheiro XML (./graphdb/import/starwars_rdf.xml).

Alguns avisos e considerações:

- Para evitar conflitos, não execute o GraphDB simultaneamente no ambiente local e no contêiner, pois ambos utilizam a mesma porta, o que pode causar erros de conexão.
- A importação dos dados está sujeita a um limite de tempo de 25 minutos, após o qual o contêiner com o Django interrompe a tentativa de conexão. Se a importação não for concluída dentro desse período, é necessário reiniciar o contêiner executando `docker compose down` seguido de `docker compose up`.

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 (`python3 csv_to_rdf.py` ou `python3 csv_to_rdf.py`, dependendo da instalação)