UNIVERSIDADE AUTÓNOMA DE LISBOA
LUÍS DE CAMÕES

DEPARTAMENTO DE CIÊNCIAS E TECNOLOGIAS

Gerador de Dados Sintéticos para Testes de Qualidade de Dados

Engenharia Informática - Relatório Projeto Final Curso

Autor/a: Fábio Miguel Rodrigues Matos de Brito
Sandra Marisa Afonso de Matos
Duarte Manuel Nunos Patrício Pessanha Santos
Orientadora: Valéria Magalhães Pequeno
Número dos candidatos: 20160803, 30001366, 20160234

Julho de 2019
Lisboa

1 Resumo

Na era dos data lakes, ter grandes quantidades de dados apresenta um enorme valor a médio e longo prazo tanto para novas oportunidades de negócio como para captar investimento.

A web semântica com os seus altos e baixos desde o início do século ganha novo fôlego com qualidade desses dados para a organização de dados estruturados (Linked Data) na criação de vocabulários de dados ligados para utilização em algoritmos de inteligência artificial.

A qualidade dos dados, quando não auferida irá resultar com certeza, em interpretações errôneas e distantes da realidade e por consequência, decisões erradas que podem pôr em causa objetivos traçados.

Assim, apresentamos uma ferramenta robusta, capaz de gerar dados de forma controlada para que nos seja possível aferir a qualidade dos mesmos: O Generatron.

Palavras-chave: Web Semântica; Gerador de Dados Sintéticos; Linked Data; Representação da Informação.

2 Abstract

In the era of datalakes, having huge amounts of data is of great value in the middle and long term for new business opportunities and capture investment where revenue is foreseen.

Web semantics with its ups and downs since the beginning of the century, gets a new push to guarantee data quality for the structuring of data (Linked Data) in the creation of vocabularies os connected data for its usage in artificial intelligence.

Critical decisions are taken everyday based in private data history and locked from the general audience.

Therefore, there is a need of a software tool that is able to produce controlled generated data which will then be crucial to predicting the quality of it: The Generatron.

Keywords: Semantic Web; Synthetic Data Generator; Linked Data; Data Representation.

3 Introdução

A WEB 2.0, ou Web Social, com o seu dinamismo, veio criar não só ligações interpessoais, como um vasto conteúdo de informação disponível aos vários setores da sociedade, transformando esta Web numa plataforma de partilha de informação.

Está assim criado o maior repositório de informação disponível à escala mundial, em que grande parte do seu conteúdo é criado e partilhado pelos utilizadores desta Web.

Existem mecanismos de pesquisa que exploram este oceano de informação, mas estes não se revelaram suficientes, tornando-se difícil encontrar aquilo que se procura, pois, o número de resultados obtidos é bastante grande e muitas vezes impreciso.

Existe uma grande variedade de sítios na internet com informação útil que muito dificilmente se consegue localizar. Por um lado, devido a grande parte dessa informação estar algures perdida num oceano de informação, por outro porque pode residir em bases de dados cujos mecanismos de pesquisa não conseguem alcançar.

Nos últimos anos a Web evoluiu de um espaço de informação global de documentos ligados para um onde ambos os documentos e dados estão ligados. Evoluímos então para a Web Semântica.

A Web Semântica ou do conhecimento, é a evolução da WEB 2.0 e veio introduzir uma perspetiva diferente do modo como utilizamos a internet. É a mudança de paradigma em que o conteúdo produzido passa a ter significado.

Esta Web de dados, permite a organização, hierarquização, integração, partilha e compreensão semântica dos dados transformando a então teia gigantesca de informação numa base de conhecimento partilhada tanto por humanos como por máquinas.

Passa a existir uma aproximação ao mundo da inteligência artificial onde a máquina aprende com o utilizador, em que este começa a obter respostas personalizadas, poupando o seu tempo. São utilizados mecanismos de inferência para a dedução de novos factos a partir dos existentes. Através das suas tecnologias, a Web Semântica permite a integração de dados de

várias fontes heterogéneas, trazendo interoperabilidade em aplicações, serviços, dispositivos, etc.

A Web semântica já existe nossa realidade em aplicações como a Siri da Apple, a Cortana da Microsoft ou a Alexa da Amazon.

Rapidamente, podemos logo pensar na aplicabilidade deste paradigma relativamente recente, ao mundo do Big Data, ou ao mundo da Internet das coisas (IoT) suportada em redes 5G. Big Data, são os milhares de migalhas que cada um de nós deixa na sua pegada digital, deixando um vasto rasto de informação para ser lida e interpretada. Trazendo tecnologias capazes de capturar, calcular, comunicar e colaborar em torno de um sistema de informação, a IoT torna-se numa espécie de acesso à rede capaz de ligar o mundo físico ao mundo da informação.

Ao escolher este tema para projeto final de curso, pensámos imediatamente em inovação e futuro da tecnologia Web. Deixámo-nos levar num mundo completamente novo que foi despertando bastante curiosidade e vontade de descobrir mais sobre o tema. Contudo, não se apresentou um trabalho fácil, houve muita análise e descoberta de novos conceitos, por detrás do que é aqui apresentado.

Não se pretende com este documento explorar e analisar todo o universo que é a Web semântica. Pretende-se sim, contar um pouco da história da Web Semântica, desde a evolução, passando por algumas das suas características, arquitetura e algumas tecnologias mais focadas no nosso trabalho para dar um contexto à ferramenta desenvolvida – Generatron.

A capacidade de ligar dados de diferentes fontes através de análise semântica significa criar datasets que após tratados podem definir o próximo passo para as organizações. É aqui que a qualidade desses dados se torna chave.

O Generatron, o nosso projeto que aqui apresentamos, é uma ferramenta desenvolvida na plataforma Java, permite gerar dados com erro induzido com o objetivo final de efetuar comparação entre dados reais de Web Semântica, e os dados similares gerados com os erros induzidos ao critério do utilizador de modo a aferir a qualidade dos mesmos.

Trazemos assim um contributo ao mundo da Web Semântica, pois o Generatron permite a produção de dados estruturados avulso de tipos especificados, isto é, permite a escolha de um

input RDF ou uma inicialização virgem, em que em ambos os casos os dados são gerados de uma forma pseudoaleatória, deixando ao critério do utilizador a escolha do tipo de output: RDF, SQL ou CSV. Traz também a possibilidade de o utilizador poder escolher que tipo de erro quer induzir assim como qual a percentagem de dados que serão impactados.

Devido a estas características, e a ausência de ferramentas que realizem as especificações do Generatron, consideramos ter encontrado um nicho de mercado para o qual trazemos mais valia.

Com vista a evidenciar o contexto da Web Semântica e explorar a ferramenta desenvolvida, este documento encontra-se dividido em vários capítulos, com destaque:

No quinto capítulo, é descrita uma pequena evolução desde a Web 1.0 à Web 3.0, indicando algumas das suas características. Falamos também das limitações da Web 2.0 demonstrando a importância da Web Semântica e o que muda no mundo das ligações da informação residente na Web.

No sexto capítulo, são ilustradas o que considerámos como principais características da Web Semântica.

No sétimo capítulo, é abordada a arquitetura da Web Semântica, com explicação de alguns conceitos subjacentes ao tema, bem como algumas tecnologias usadas. Não são exploradas em pormenor todas as particularidades da arquitetura, mas é dado enfoque às tecnologias base de desenvolvimento da ferramenta Generatron.

No oitavo capítulo, exploramos a ferramenta Generatron – Gerador de dados sintéticos para testes de qualidade dos dados. São detalhadas as vantagens desta ferramenta, explicando ao pormenor a sua definição, objetivos, respetivas funcionalidades bem como as características de implementação mais importantes.

No nono capítulo é apresentada uma análise global do trabalho realizado, principais dificuldades e possíveis melhoramentos no Generatron.

4 Índice

5 A evolução da World Wide Web

A primeira versão da internet ARPANET, surgiu na década de 1960 servindo objetivos militares dos Estados Unidos da América proporcionando a partilha de informação.

Criada na década de 1990 por Tim Berners Lee, a WEB veio permitir o acesso a dados online sob a forma de sites e hiperlinks, popularizando a internet entre o público sendo a base do desenvolvimento de uma grande quantidade de informação a que todos acedemos atualmente.

A WEB pode ser vista como uma rede de servidores que disponibiliza o acesso a um conjunto de documentos ou páginas interligadas que podem conter imagens, sons, vídeos, animações, etc.

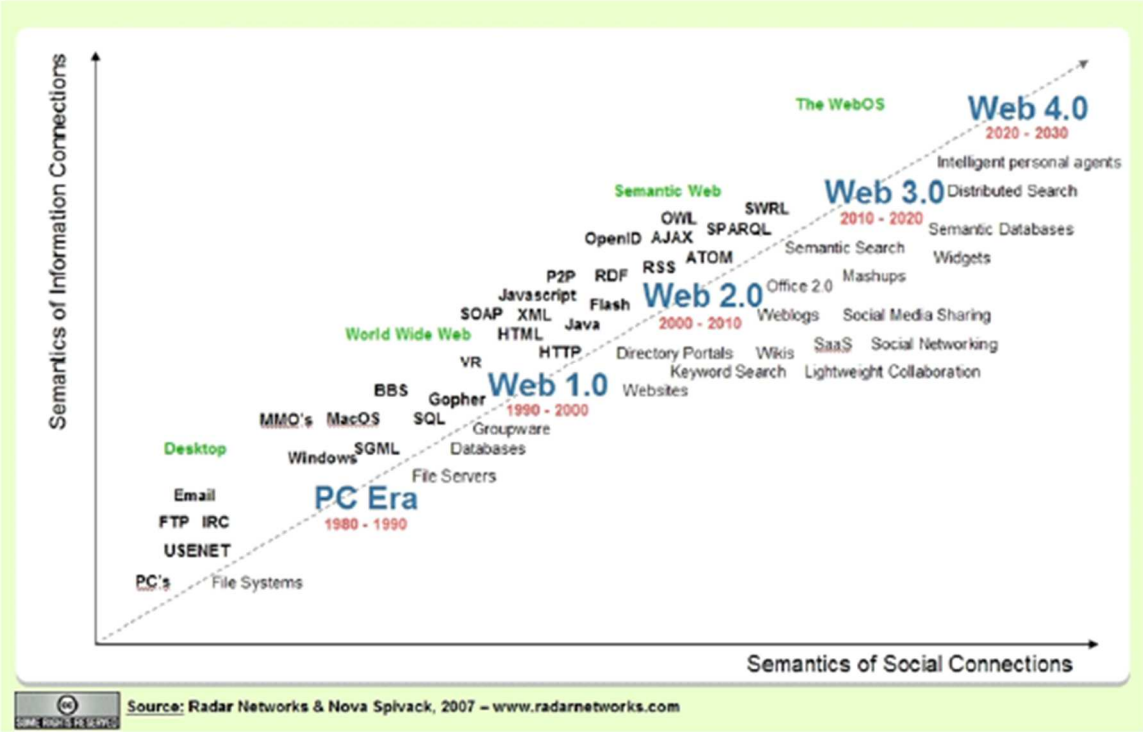

Figura 1 - Evolução da WEB[1]

1 https://books.google.pt/books?id=TetZYFPh_Y4C&pg=PA4&lpg=PA4&dq=radarnetworks.com&source=bl&ots=asAltY9yGa&sig=ACfU3U37Peihm-cC24mRad8sdJ5A2ayMpg&hl=en&sa=X&ved=2ahUKEwi53sDm2JTjAhU3QUEAHZdNCVQQ6AEwDnoECAgQAQ#v=onepage&q=radarnetworks.com&f=false

5.1 WEB 1.0

Lançou um mundo novo. Caracterizada como estática, pois a interatividade dos utilizadores era bastante baixa, limitando-se a disponibilizar grandes quantidades de informação apenas para leitura.

Esta WEB tinha poucas atualizações, e as páginas continham poucos elementos como imagens, ícones de navegação, textos, menus. A estrutura destas páginas era bastante minimalista e estática com um mínimo de interação entre sites.

5.2 WEB 2.0

Representa a segunda geração da internet - WEB Social. Esta WEB já é participativa, permitindo a leitura e escrita por parte dos utilizadores o que a tornou interativa, tornando-se na revolução dos blogs, redes sociais, Wikis,etc.

Conectiva e dinâmica, a WEB 2.0 passou a criar ligações entre pessoas que passaram a ter um papel importante no seu desenvolvimento levando à criação de uma enorme quantidade de informações disponíveis.

Desta forma, os motores de pesquisa aumentaram e tornaram-se mais avançados, ao usar algoritmos que permitam retornar os resultados mais relevantes.

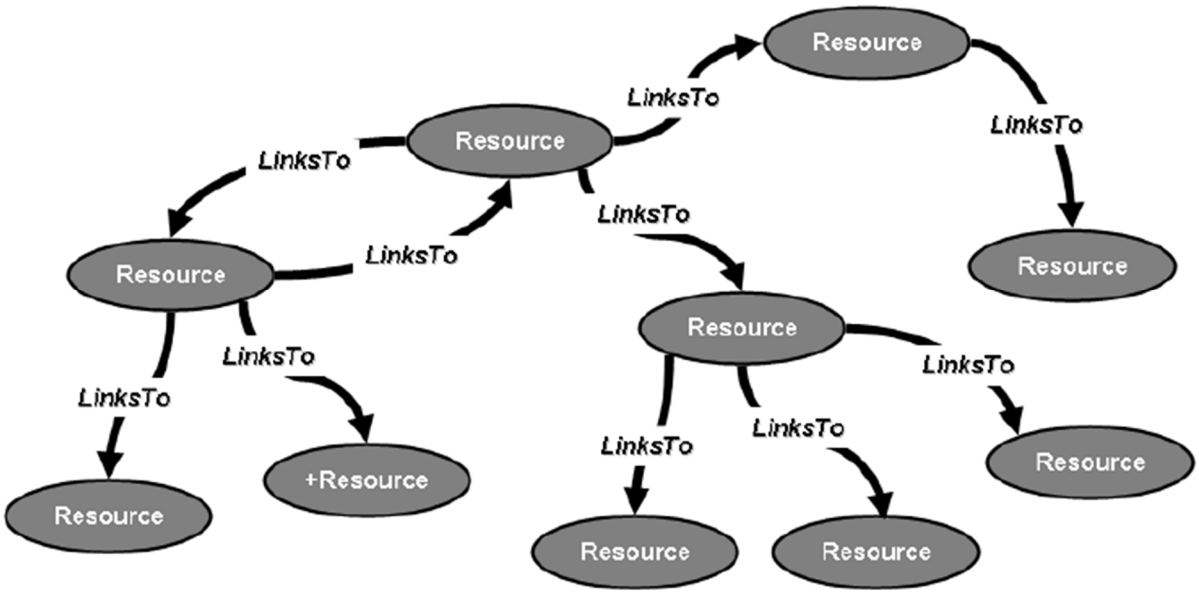

Figura 2 - Esquema da Web 2.0

5.3 WEB 3.0

É considerada uma evolução da WEB 2.0, veio dar estrutura, contexto e relacionamento aos dados disponíveis na internet de modo a que sejam compreendidos não só por humanos, mas também por computadores, constituindo um novo paradigma – WEB Semântica.

A WEB Semântica ou do conhecimento, já existe na nossa atualidade como a Siri da Apple, Cortana da Microsoft, Alexa da Amazon ou o WolframAlpha, estes sistemas usam linguagem natural e inteligência artificial para fornecer informações aos utilizadores. Os motores de pesquisa já não se limitam a recolher e apresentar dados sendo capazes de efetuar o uso mais inteligente desses dados e produzir respostas mais concretas, poupando tempo aos utilizadores ao evitar a navegação na imensidão de respostas obtidas na WEB anterior até à obtenção do resultado pretendido.

Para perceber melhor a diferença entre a WEB 2.0 e WEB 3.0, podemos efetuar uma pesquisa pelo jogo de futebol da copa américa Chile vs Peru:

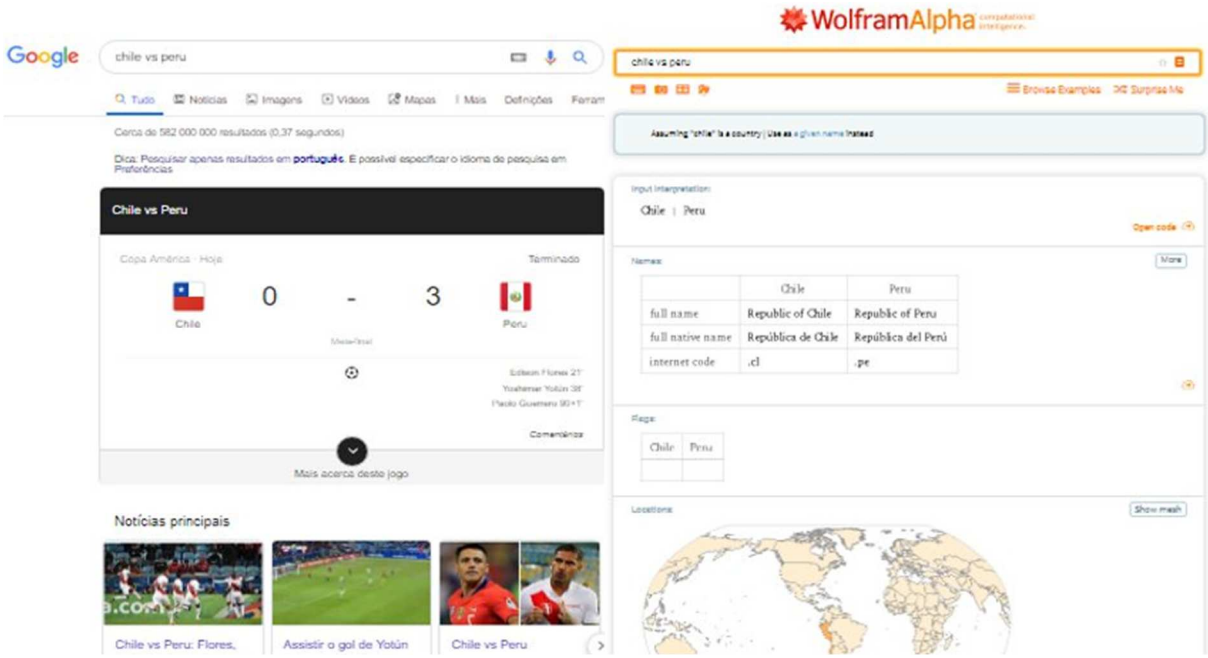

Figura 3 - Comparativo entre Google e WolframAlpha como exemplo da Web 3.0

No caso do Google, a pesquisa apresenta os conteúdos mais frequentes, dando maior importância ao jogo Chile e Peru. No caso do Wolfram Alpha, a pesquisa entende que se trata de uma comparação entre os dois países e retorna dados estatísticos, entre outras informações, dos mesmos.

5.4 Limitações da WEB 2.0

A WEB 2.0 passou a permitir que os utilizadores se transformassem em consumidores e produtores de informação.

Desde um simples blog em que alguém publica um conteúdo, permitindo que outros utilizadores comentem e expressem as suas opiniões a grandes redes sociais que permitem a conectividade e recolha de informação dos seus utilizadores. A produção de informação não se esgota aqui, empresas, órgãos do estado, escolas são mais alguns exemplos de produtores e consumidores de informação.

Grande parte do conteúdo é criado e partilhado pelos seus utilizadores, a WEB 2.0 transformou-se essencialmente numa plataforma de partilha de informação.

Está criado o maior repositório de informação à escala mundial que disponibiliza conteúdos e oferece serviços a todos os setores da sociedade.

Face ao crescimento de informação, surgiram mecanismos de pesquisa que facilitam a exploração e recuperação da informação. Contudo, estes não são suficientes face à teia de informação de dimensões gigantescas, tornando-se difícil encontrar aquilo que se procura, pois o número de resultados obtidos é bastante grande e muitas vezes impreciso.

Existem uma grande variedade de sítios na internet com informação útil que muito dificilmente se conseguem localizar. Por um lado, devido a que grande parte dessa informação estar algures perdida num oceano de informação, por outro porque pode residir em bases de dados cujos mecanismos de pesquisa não conseguem alcançar.

Parte deste problema reside por um lado, que grande parte da informação disponível só é entendida por humanos, por outro lado os motores de pesquisa que pesquisam a informação por palavras chave e rankings, não dão contexto e significado aos dados.

6 Web Semântica

A Web Semântica ou do conhecimento, é a evolução da WEB 2.0, veio introduzir uma perspetiva diferente do modo como utilizamos a internet. É a mudança de paradigma em que o conteúdo produzido passa a ter significado.

Passa a existir uma aproximação ao mundo da inteligência artificial onde a máquina aprende com o utilizador, que ao invés de efetuar pesquisas por palavras chaves nos motores de busca e obter centenas de resultados passa a obter apenas uma resposta personalizada e concreta.

Esta Web aposta na organização, estruturação de recursos de informação na descrição dos mesmos através de metadados e suportados em ontologias que dão significado e contexto à informação. Desta forma, as máquinas têm acesso a toda a informação disponível na internet.

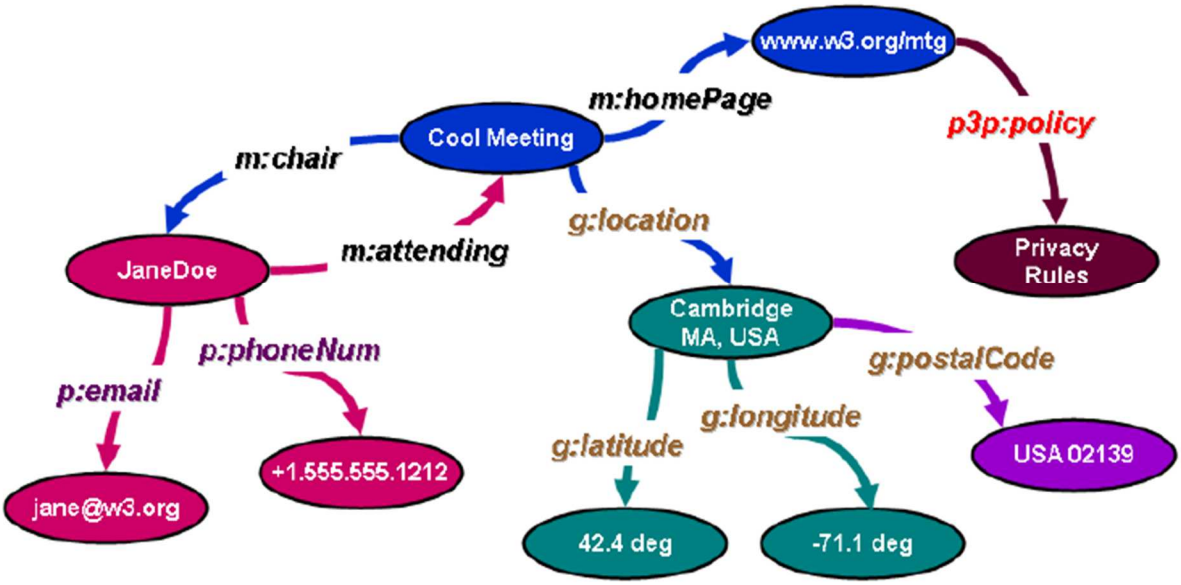

Figura 4 - Esquema da Web Semântica

As aplicações para esta Web, podem processar, utilizar e partilhar informação de forma inteligente, transformando a Web numa enorme base de conhecimento partilhado tanto por humanos como por aplicações e/ou máquinas.

Esta Web inteligente, é uma iniciativa do W3C (World Wide Web Consortium) cujo propósito principal é a integração, intercâmbio e compreensão semântica da informação, tanto na ótica dos humanos como na ótica das máquinas através de metadados (dados sobre dados), de ontologias (formas de representar explicitamente a semântica dos dados de um determinado domínio) e agentes de software (componentes de software capazes de atuar de forma a resolver tarefas pelo utilizador).

Em 1999, Tim Berners Lee expressou a sua visão sobre a Web Semântica:

> *Eu tenho um sonho para a Web [no qual os computadores] serão capazes de analisar todos os dados presentes na Web - os conteúdos, links e transações entre pessoas e computadores A 'Web Semântica", que deve tornar isso possível, ainda a surgir, mas quando isso acontecer os mecanismos do dia-a-dia da burocracia, do comércio e das nossas vidas diárias serão tratados por máquinas conversando entre si.*[2]

6.1 Algumas características da Web Semântica

- **Inteligência** - Um dos recursos mais promissores da Web 3.0 é a web com inteligência, ou seja, uma teia inteligente. As aplicações trabalharão de maneira inteligente com o uso da interação humano-computador e inteligência. Diferentes ferramentas e técnicas baseadas em Inteligência Artificial (IA) (tais como conjuntos fuzzy, redes neurais, machine learning, etc.) serão incorporadas às aplicações para funcionar inteligentemente. Documentos em diferentes idiomas podem ser traduzidos de uma forma inteligente para outras linguagens na Web 3.0. Deve permitir trabalhar através de linguagem natural. Portanto, os utilizadores podem usar seu idioma nativo para comunicação com os outros ao redor do mundo.

[2] I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A "Semantic Web", which makes this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. From https://en.wikipedia.org/wiki/Semantic_Web

- **Personalização** - Outra característica da Web 3.0 é personalização. Preferências pessoais ou individuais são considerados durante diferentes atividades, como processamento de informações, pesquisa, etc. A Web Semântica é a principal tecnologia de Personalização na Web 3.0.

- **Raciocínio** - Web Semântica permite a pesquisa, integração, respondendo a consultas complexas, ligações e análises, descoberta de padrões, mineração, validação de hipóteses, descoberta, visualização, etc.

- **Interoperabilidade** - A interoperabilidade refere-se a aspetos como a integração contínua de dados de fontes heterogêneas, composição dinâmica, interoperação de serviços da Web e os mecanismos de pesquisa. Aplicações Web 3.0, devem ser de fácil customização e podem funcionar independentemente em diferentes tipos de dispositivos. Uma aplicação baseada em Web 3.0 pode ser executada em vários tipos de computadores, dispositivos portáteis, telefones, TVs, automóveis e muitos outros.

- **Usabilidade** - Engloba novos paradigmas de recuperação de informação, interfaces de utilizador, interação e técnicas de visualização, que por sua vez exigem métodos para lidar com dependência de contexto, personalização, confiança entre outros, enquanto oculta os problemas computacionais subjacentes ao utilizador.

7 Arquitetura Web Semântica

O termo Web semântica, refere-se algumas vezes a um conjunto de tecnologias que permitem o relacionamento de dados e tornar possível a Web de dados.

Para ilustrar o relacionamento dessas tecnologias, existe a ilustração mais recente da W3C, conhecida por Layer Cake:

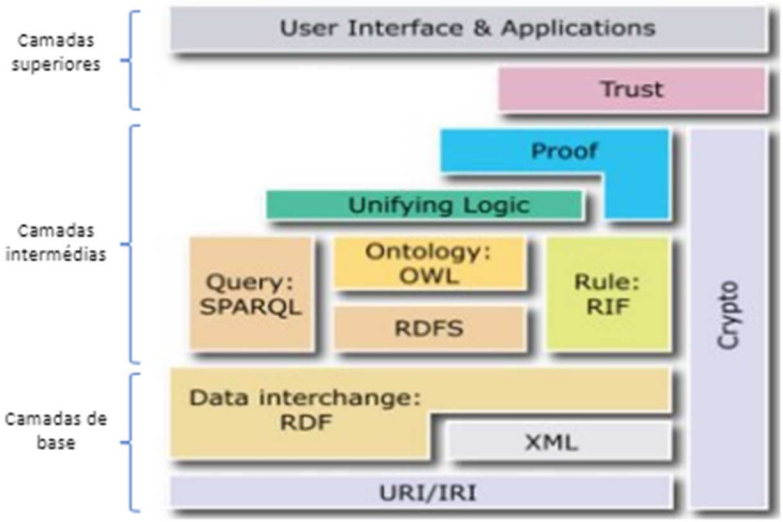

Figura 5 - Arquitetura Web Semântica, Fonte https://www.w3.org/2007/03/layerCake.png

Cada camada explora e utiliza as capacidades das camadas abaixo.

- As camadas de base, permitem estruturar e padronizar os dados, identificar recursos de forma única, e fornecer os meios para a representação, armazenamento e intercâmbio de informação. (Iremos detalhar um pouco URI, RDF, XML e Linked Data)

- As camadas intermédias, possibilitam a estruturação dos dados associados a um recurso e a verificação da consistência desse recurso de acordo com regras sintáticas formalmente descritas e previamente definidas e validadas.

Estas camadas, são também responsáveis pela criação de vocabulários para a descrição semântica dos recursos e para a definição das relações existentes entre estes, de acordo com especificações formais, explícitas e compartilhadas de conceitos.

Permitem também declarar regras lógicas que podem ser executadas pelos agentes de software para realização de inferências automáticas e verificação do nível de coerência lógica entre os recursos.

A camada lógica unificadora é necessária para garantir o intercâmbio confiável de dados entre as três camadas: consultas, ontologias e regras.

A Camada de criptografia fornece as técnicas de criptografia para verificar as origens da fonte de dados para entradas confiáveis, enquanto as camadas de prova e confiança garantem que todas as semânticas e regras sejam cumpridas e executadas com eficácia antes de passar os resultados para uma camada de interface de usuário e aplicativos. (Iremos detalhar um pouco Ontologias e Regras de Inferência)

- As camadas superiores, tratam da interface entre humanos e computadores e do desenvolvimento de aplicações e interfaces que facilitam a busca e a apresentação da informação com diferentes graus de qualidade e confiança.

7.1 URI/IRI

No núcleo tecnológico da Web Semântica está a camada URI/IRI, Uniform Resource Identifier e International Resource Identifier respetivamente.

Servem para identificar recursos na Web. Os URL (Uniform Resource Locator) fazem parte do padrão URI, todos os URL são URI, mas nem todos os URI são URL. O URL é um endereço onde se encontra um recurso na Web.

Esta camada em conjunto com a camada XML, fornece a base para o intercâmbio de dados na Web usando a camada RDF - formato gráfico para codificação e descrição de recursos na Web.

7.2 Recursos

Para publicar dados na internet, necessitamos de identificar os recursos.

Esses recursos podem ser de dois tipos: recursos de informação e recursos de não-informação.

Recursos de informação podem ser imagens, textos, documentos, vídeos, etc. Recursos de não-informação são os recursos sobre os quais queremos partilhar informações, pessoas, lugares, conceitos, ou seja, é qualquer recurso real que exista fora da Web.

Esta distinção é importante para o contexto de linked data, que falaremos a seguir, porque podemos indicar às máquinas qual o tipo de dados com que estão a trabalhar e como é que podem usar esses recursos.

As informações que são relacionadas com cada tipo de recurso são diferentes. Cada recurso é definido de forma a evitar ambiguidades e sua identificação também serve para dar algum contexto sobre dados relacionados. Estes recursos são identificados por URIs baseados no protocolo HTTP.

7.3 Linked Data - Dados interligados

Nos últimos anos a Web evoluiu de um espaço de informação global de documentos ligados para um onde ambos os documentos e dados estão ligados.

Sustentando esta evolução, está um conjunto de boas práticas recomendadas para publicar e conectar dados estruturados na Web conhecidos como Linked data (dados interligados). A adoção destas práticas levou à extensão da Web com um espaço de dados global que liga dados de diversos domínios: empresas, pessoas, publicações científicas, filmes, músicas, drogas e ensaios clínicos, etc. (Christian Bizer, 2009)

Esta Web de dados veio permitir um novo tipo de aplicativos. Existem navegadores de dados que permitem aos utilizadores começar a pesquisar numa determinada fonte de dados e, em seguida, navegar ao longo de ligações de fontes de dados relacionados.

Existem também mecanismos de pesquisa que efetuam o “crawling” na Web de dados, seguindo as ligações entre dados fornecendo consultas sobre os dados agregados semelhantes a consultas a bases de dados locais como conhecemos.

Linked data refere-se ao uso da Web para criação de ligações tipificadas entre dados de diferentes origens, refere-se a dados publicados na Web tornando-os legíveis para máquinas.

Para se ter uma ideia do estado deste conceito, existe um termo chamado de LOD Cloud - Linked Open Data Cloud (http://lod-cloud.net). O diagrama seguinte, mostra o estado das ligações de dados estruturados em 2007.

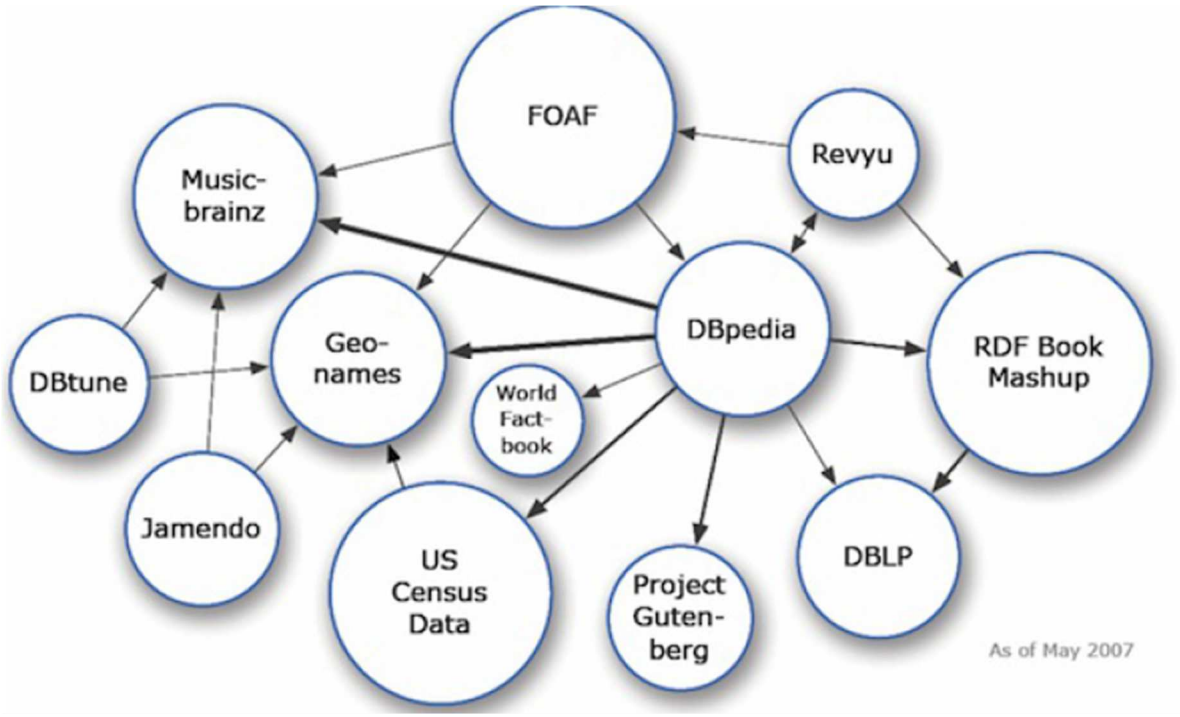

Figura 6: Fonte: http://lod-cloud.net/versions/2007-05-01/lod-cloud.png

O diagrama seguinte, mostra o estado das ligações de dados estruturados em 2019.

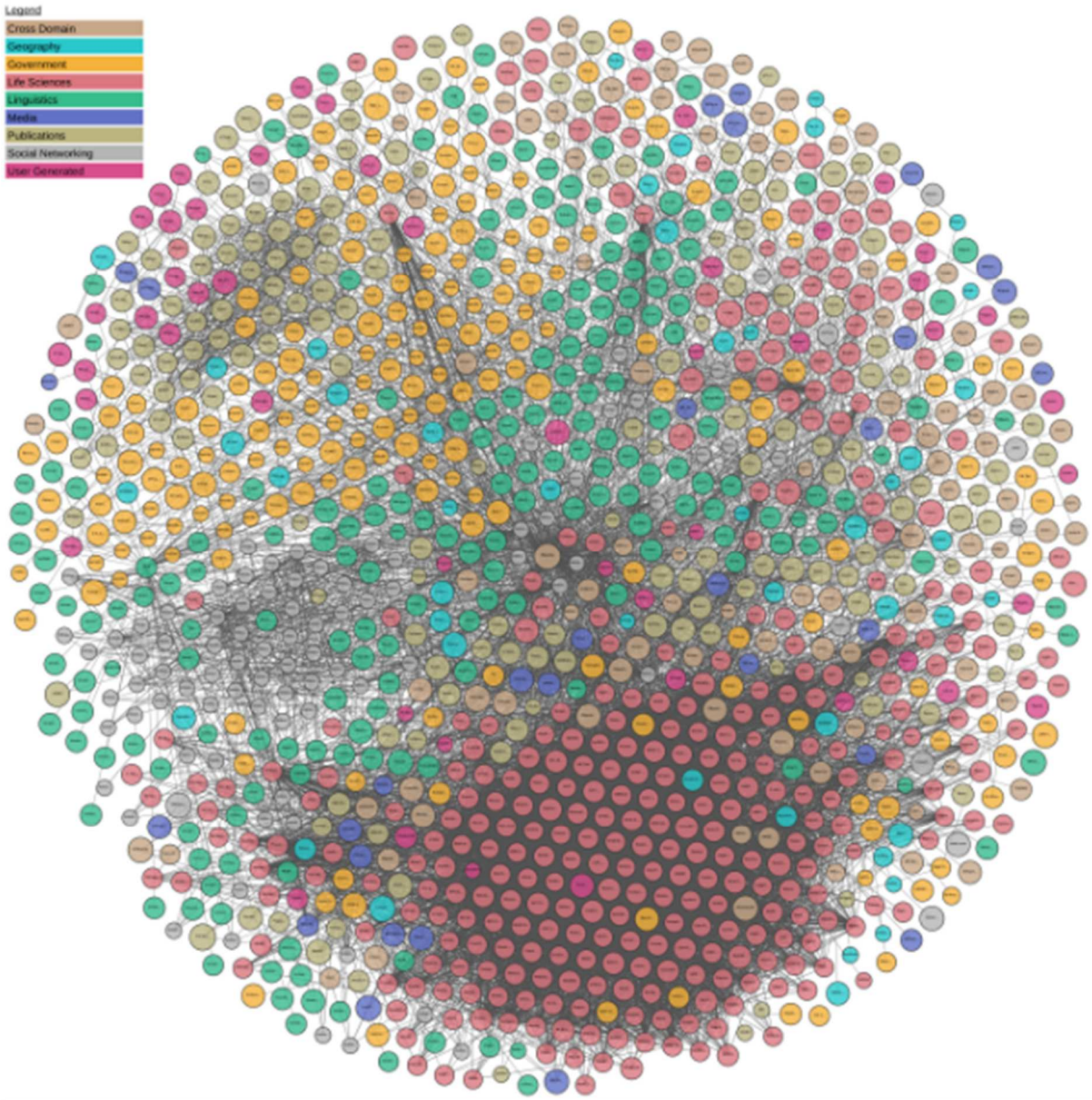

Figura 7: Fonte:https://lod-cloud.net/versions/2019-03-29/lod-cloud.png

Pela evolução, dos diagramas, consegue-se facilmente perceber que houve um esforço crescente para agrupar, organizar os dados existentes na Web num formato padrão. Os nós representam um conjunto de dados abertos - datasets. (Eis, 2017)

Enquanto os documentos com ligações não tipificadas da Web 2.0, assentam em HTML (Hypertext Markup Language), linked data assenta em documentos com dados no formato RDF (Resource Description Framework), assim em vez de simplesmente ligar esses documentos, linked data usa o RDF para fazer declarações tipificadas que ligam “coisas”, recursos, arbitrárias no mundo.

Existem na Web, sítios com datasets disponíveis, alguns exemplos:

- WikiData — https://www.wikidata.org/wiki/Wikidata:Main_Page
- DBPedia — http://dbpedia.org
- AWS Public Datasets — https://aws.amazon.com/datasets/
- data.gov — http://catalog.data.gov/dataset
- Unicef Data — https://data.unicef.org

(Berners-LEE, 2006), delineou um conjunto de regras para publicação de dados na Web de forma a que estes se tornem parte de um único espaço global de dados:

1 - Usar URIs (Uniform Resource Identifier) para nomear as "coisas".
2 - Usar HTTP URI para que as pessoas possam encontrar essas "coisas".
3 - Quando alguém procurar um URI, a informação útil deverá ser disponibilizada através dos standards RDF e SPARQL.
4 - Incluir Links para outros URIs para que se obtenha relacionamentos entre informação.

Estas regras assentam em duas tecnologias base da Web: URIs (Uniform Resource Identifier), para identificação de recursos e HTTP para descrever esses recursos de informação.

O HTML fornece estrutura e ligações para documentos na Web, o RDF é uma framework para modulação de dados de forma genérica relacionando-os através de triplas: sujeito, predicado e objeto.

O sujeito e objeto são ambos URIs (Uniform Resource Identifier) que identificam um recurso ou um URI ou uma string. O predicado específica como o sujeito e objeto estão relacionados e é também representado por um URI.

Figura 8 - Estrutura de uma tripla

A relação é feita do sujeito para o predicado que no RDF é chamado de propriedade.

Podemos pensar no RDF como uma linguagem para expressar declarações sobre recursos, facilitando a comunicação entre máquinas. (Eis, 2017) Exemplo:

1 - Diego > é uma > pessoa
2 - Diego > é casado com > Marcela
3 - Diego > nasceu em > 3 de dezembro de 1983
4 - Diego > é interessado na > Amazon
5 - Amazon > foi criada por > Jeff Bezos
6 - O livro A loja de tudo > é sobre a > Amazon

Neste exemplo, Diego é sujeito em quatro triplas (1 a 4), a Amazon é sujeito numa tripla (5) e objeto em duas triplas (4 e 6). Recursos ligados desta forma podem ser extraídos de diferentes datasets na Web.

Esta possibilidade de ter a mesma fonte como sujeito e objeto em várias triplas é o que possibilita encontrar ligações entre os dados.

Representação de triplas em forma de grafos:

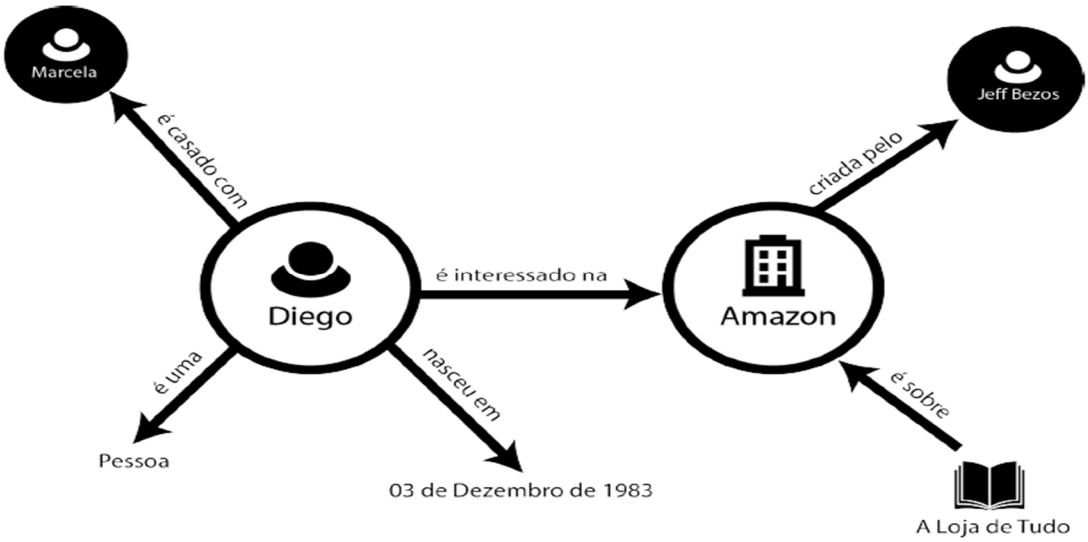

Figura 9 - Representação de triplas em grafos

Exemplo de ligações RDF:

```
Subject: http://dig.csail.mit.edu/data#DIG
Predicate: http://xmlns.com/foaf/0.1/member
Object: http://www.w3.org/People/Berners-Lee/card#i

Subject: http://data.linkedmdb.org/resource/film/77
Predicate: http://www.w3.org/2002/07/owl#sameAs
Object: http://dbpedia.org/resource/Pulp_Fiction_%28film%29
```

Figura 10 - Exemplo de ligações RDF

Na figura 8, existem dois exemplos de links RDF, o primeiro link é um recurso identificado pela URI http://www.w3.org/People/Berners-Lee/card#i que é membro de outro recurso identificado por outro URI http://dig.csail.mit.edu/data#DIG .

Quando o sujeito é solicitado via HTTP, o servidor dig.csail.mit.edu responde com a descrição do recurso identificado no formato RDF. Quando o objeto é solicitado via HTTP, o servidor W3C responde com um grafo em RDF descrevendo Berners Lee. No caso do predicado é produzida uma definição da ligação do tipo type member descrita em RDF.

O segundo link liga a descrição do filme Pulp Fiction do link Movie Database com a descrição do filme fornecida pela DBpedia, afirmando que o URI http://data.linkedmdb.org/resource/film/77 e o URI

http://dbpedia.org/resource/Pulp_Fiction_%28film%29 referem-se à mesma entidade no mundo - filme Pulp Fiction.

7.4 XML

Camada composta pela linguagem XML, pelo uso de namespaces (espaço de nomes) e pelo XML schema.

É responsável pelo estabelecimento correto da sintaxe da descrição dos dados. O XML proporciona uma melhor estruturação e padronização dos recursos de informação e dos dados e metadados que representam o recurso.

Esta linguagem torna-se fundamental, pois dá bastante importância ao conteúdo dos dados e não somente à sua forma de apresentação, possibilitando aos agentes de software uma melhor visualização dos dados.

O XML schema permite definir o formato válido de um documento XML, incluindo a validade dos elementos e atributos, quais as suas localizações, número de ocorrências de cada elemento e outras características, ou seja, proporciona mecanismos para a definição de gramáticas de correção de documentos XML.

Os namespaces ou espaço de nomes, considerados como um método para qualificar nomes de elementos e atributos em documentos XML, através da associação de referências URI. Através deste mecanismo é possível a combinação de documentos com a utilização de vocabulário partilhado. É possível também partilhar e reutilizar a definição de outros schemas XML sem existir colisão de nomes. Por possuírem estas características, os espaços de nomes são utilizados em RDF e ontologias.

7.5 RDF

O RDF, Resource Description Framework, é uma linguagem padrão do W3C para representar informações sobre recursos na Web.

Destina-se particularmente a representar metadados sobre recursos na Web, como título, autor e data de modificação de uma página da Web, informações sobre direitos de autor, etc. No entanto, ao generalizar o conceito de “recurso na Web”, o RDF também pode ser usado para

representar informações sobre coisas que podem ser identificadas na Web, como exemplo, informações sobre itens disponíveis em compras online (preços, disponibilidades, informações sobre especificações). (W3C, RDF Primer, 2004)

O RDF é destinado a situações em que as informações necessitam de ser processadas por máquinas/aplicativos. Fornece uma estrutura comum para expressar essa informação para que possa ser trocada entre aplicações sem perda de significado.

O RDF baseia-se na ideia de identificar as coisas usando URIs, descrevendo recursos em termos de propriedades simples e valores de propriedade. Esta informação sobre recursos propriedades e valores, é representada em grafos de nós e arcos, como já falado anteriormente, representado o sujeito, predicado e objeto.

Tomando como exemplo, que a pessoa John Smith criou uma página Web em inglês em 16 de Agosto de 1999, em termos declarativos ficaria:

http://www.example.org/index.html has a creator whose value is John Smith
http://www.example.org/index.html has a creation-date whose value is August 16, 1999
http://www.example.org/index.html has a language whose value is English

Em terminologia RDF, o sujeito é a página Web, a parte que identifica a propriedade ou característica do sujeito da declaração (creator, creator-date, language) são os predicados, a parte que identifica os valores das propriedades (John Smith, August 16, 1999, English) são os objetos.

Utilizando referências URI (URIrefs) o RDF pode descrever tudo sobre alguma coisa, continuando com o exemplo acima, uma possível declaração RDF:

Sujeito: http://www.example.org/index.html
Predicado: http://purl.org/dc/elements/1.1/creator , namespace usado para definir o vocabulário Dublin Core
Objeto: http://www.example.org/staffid/85740

Neste exemplo são usados URIrefs para identificar o sujeito, predicado e objeto. Podemos representar a declaração em forma de grafo:

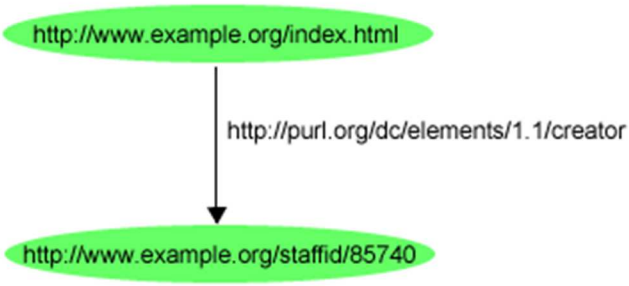

Figura 10 - Representação de uma declaração RDF em forma de grafo, Fonte: https://www.w3.org/TR/rdf-primer/

Na figura acima, o sujeito e objeto são representados por nós e o predicado por arco.

Os objetos podem ser URIrefs ou constantes (literais), como a data acima exemplificada. Os literais, usam caracteres unicode que permitem a representação de informação em várias línguas.

A estrutura de dados do RDF retorna uma coleção de declarações conhecidas como triplas, que significam uma unidade básica de informação que foi construída. Continuando com as declarações acima como exemplo, obtemos as seguintes triplas:

<http://www.example.org/index.html> <http://purl.org/dc/elements/1.1/creator> <http://www.example.org/staffid/85740> .

<http://www.example.org/index.html> <http://www.example.org/terms/creation-date> "August 16, 1999" .

<http://www.example.org/index.html> <http://purl.org/dc/elements/1.1/language> "en" .

A notação tripla requer que os URIrefs sejam escritos entre brackets, que como o exemplo ilustra, pode resultar em linhas muito longas.

Por conveniência, é usada uma abreviação que substitui um nome qualificado XML (QNAME) como abreviação de um URIref. Um QNAME contém um prefixo que está atribuído a um namespace URI, seguido de dois pontos e um nome local. Exemplos:

prefix exterms:, namespace URI: http://www.example.org/terms/ ,
prefix exstaff:, namespace URI: http://www.example.org/staffid/ ,
prefix ex:, namespace URI: http://www.example.org/ ,
prefix dc:, namespace URI: http://purl.org/dc/elements/1.1/ .

Ao usar esta abreviação, as triplas resultam em:

```
ex:index.html  dc:creator                exstaff:85740 .
ex:index.html  exterms:creation-date  "August 16, 1999" .
ex:index.html  dc:language               "en" .
```

Para processar instruções de forma a sejam entendidas por máquinas, o RDF utiliza o XML, conhecido como RDF/XML. Esta linguagem é a sintaxe para escrever RDF.

Pode-se usar as triplas acima declaradas e escrevê-las em RDF/XML:

```
1.  <?xml version="1.0"?>
2.  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.           xmlns:dc="http://purl.org/dc/elements/1.1/"
4.           xmlns:exterms="http://www.example.org/terms/">
5.    <rdf:Description rdf:about="http://www.example.org/index.html">
6.       <exterms:creation-date>August 16, 1999</exterms:creation-date>
7.       <dc:language>en</dc:language>
8.       <dc:creator rdf:resource="http://www.example.org/staffid/85740"/>
9.    </rdf:Description>
10. </rdf:RDF>
```

Linha 1, indica que o conteúdo do documento é XML e a sua versão.

Linha 2, começa com o elemento rdf:RDF e termina na linha 10, indica que o conteúdo em XML se destina a representar RDF. De seguida, está a declaração do namespace que indica

que todas as tags com o prefixo rdf: são parte desse namespace identificado pelo URI http://www.w3.org/1999/02/22-rdf-syntax-ns#.

Linhas 3 e 4, declaram mais dois namespaces identificados pelos URI http://purl.org/dc/elements/1.1/ e http://www.example.org/terms/ para os prefixos dc: e exterms: respetivamente.

Linhas 5 a 9, descrevem o recurso. Na linha 5, o recurso é identificado com o atributo rdf:about que especifica o sujeito com o URIref http://www.example.org/index.html .

As linhas 6, 7 e 8 descrevem as propriedades do recurso. O QNAME exterms:creation-date e o valor da tag representam o predicado e o objeto, bem com o QNAME dc:language.

O elemento dc:creator, representa uma propriedade cujo valor é outro recurso identificado pelo URI http://www.example.org/staffid/85740 .

7.6 RDFS

O RDFS, ou RDF schema, tem funções semelhantes ao schema XML. É uma linguagem padrão do W3C, que define a estrutura válida de um documento RDF.

O RDF schema não fornece um vocabulário de classes específico como exterms:creation-date ou dc:language, em vez disso, fornece os recursos necessários para descrever essas classes e propriedades e indica quais classes e propriedades devem ser usadas juntas, por exemplo, a propriedade ex3:jobTitle é usada para descrever a ex3:Person.

O RDF schema fornece um sistema de tipos ao RDF. Este sistema em alguns aspetos é similar ao sistema de tipos de linguagens de programação orientadas a objetos como o Java. O RDF schema permite que os recursos sejam definidos como instâncias de uma ou mais classes, permitindo também que as classes sejam organizadas de forma hierárquica.

Por exemplo, a classe ex:Cão pode ser definida como uma subclasse de ex:Mamífero que é uma subclasse de ex:Animal, o que significa que qualquer recurso da classe ex:Cão também, está implicitamente na classe ex:Animal.

Noutros aspetos o RDF schema é bastante diferente das linguagens de programação orientadas a objetos. Uma diferença importante, é que em vez de descrever uma classe como tendo uma coleção de propriedades específicas, o RDF schema descreve as propriedades como aplicáveis a classes específicas de recursos, usando domínios e intervalos.

Por exemplo, numa linguagem de programação orientada a objetos pode definir-se a classe Livro com o atributo autor com valores do tipo Pessoa. Num RDF schema, essa informação corresponde a uma descrição de uma classe ex:Livro, numa descrição separada uma propriedade ex:autor com um domínio ex:Livro e um intervalo ex:Pessoa.

Em suma, as descrições de propriedades, são por defeito, independentes das definições das classes e têm normalmente âmbito global, embora possam ser declaradas como aplicáveis apenas a determinadas classes, usando especificações de domínio.

7.6.1 Vantagens RDF

A linguagem RDF trouxe vantagens em alguns cenários, destacando:

- Catalogar os recursos de informação e as suas relações num sistema (Website, página Web ou biblioteca digital, etc.) ou entre vários sistemas.
- Classificar o conteúdo e descrever os recursos de informação.
- Pesquisar informação de forma mais precisa, permitindo que os motores de busca ofereçam melhores resultados.
- Filtrar com maior precisão dados para obter sistemas de avaliação de conteúdo mais viáveis.
- Representar conjuntos de documentos como um único e grande documento lógico, quando apropriado.
- Facilitar o intercâmbio e a partilha de conhecimento através de agentes de software inteligentes.
- Estabelecer relações seguras entre documentos e computadores para facilitar a troca de ideias e de recursos.
- Descrever os direitos de propriedade intelectual nas páginas Web.

- Expressar as preferências de privacidade de um utilizador ou as políticas de privacidade de um Website. (Gonçalves, 2009)

7.6.2 Tecnologias RDF

Algumas aplicações da tecnologia RDF:

- Tecnologias para a distribuição de conteúdos de notícias on-line: RSS (RDF Site Summary); PRISM (Publishing Requirements for Industry Standard Metadata); e XMLNews-Meta (XML and News packaging and RDF metadata format).
- Tecnologias para redes sociais: FOAF (Friend Of A Friend) que permite criar páginas legíveis pelas máquinas (machine-readable) para descrever as pessoas, as ligações entre elas e as coisas que criam e fazem; FOAFCorp (Corporate Friends of Friends) que é uma extensão de FOAF para descrever com maior detalhe a estrutura de redes sociais e as relações entre entidades organizacionais; e Client – Haystack que permite a gestão de informação de uma forma personalizada através de um modelo RDF.
- Tecnologias para descrever coleções pessoais de música, vídeo, fotografias e agendas, tais como: MusicBrains Metadata Initiative (intercâmbio de áudio e vídeo através de metadados expressos em RDF); RDFPics (descrever e recuperar imagens usando RDF e http); e vCard (representação do formato vCard em RDF).
- Tecnologias para descrever documentos, ficheiros ou outros recursos de informação, tal como o XMP (eXtensible Metadata Platform) que nos permite adicionar metadados a ficheiros.Web ou biblioteca digital, etc) ou entre vários sistemas.

De forma a facilitar a validação do código RDF, o W3C disponibiliza um serviço online denominado RDF Validation Service http://www.w3.org/RDF/Validator/ .

7.7 Turtle

O Turtle é uma serialização do RDF para a representação de triplas de fácil leitura. Existem mais como o N-TRIPLES, Notation 3, etc. Nesta secção vamos explorar um pouco mais o Turtle pois é a base do nosso projeto.

O Turtle é uma sintaxe que expõe de uma forma simplificada e estruturada os sujeitos, predicados e objetos da norma RDF dentro dos padrões que se encontram definidos na W3C.

A base da estruturação pode ser segmentada em quatro passos distintos:

1) **Prefixos - "os domínios"**

Os prefixos, normalmente definidos no início de um ficheiro Turtle, servem o propósito de referenciar URIs de toda e qualquer fonte, atribuindo um prefixo simplista dessa referência para ser usado ao longo do documento. Por exemplo:

@prefix foaf: <http://xmlns.com/foaf/0.1/> .

Iniciando com "@prefix", temos de seguida as letras que serão usadas nas referências de todos os dados seguintes (no caso "foaf:") que vão na realidade simbolizar o endereço URI que é descrito logo de seguida entre < ... > .

2) **Sujeitos - "O quê"**

Os sujeitos, apontam um determinado recurso que à partida, vai ser afetado. Pode ser um URI, um nome prefixado, ou qualquer outro indivíduo para o qual queiramos realizar uma operação de associação seja de propriedade, ou até mesmo de classe.
No exemplo seguinte, temos o sujeito de uma declaração destacado a amarelo:

mo:Michael_Jackson a mo:MusicArtist ;
dc:description "single, pop" ;
foaf:name "Michael Jackson" ;
foaf:homepage "http://www.michaeljackson.com/" .

O fim de uma declaração que afetamos ao sujeito, finaliza-se com um ".".

3) **Predicados - "Como"**

Os predicados são extremamente importantes, na medida em que, é aqui que vamos especificar como vamos categorizar a ligação que vai ser efetuada nos grafos RDF. É a segunda parte de cada tripla e onde se especifica se um sujeito / objeto vai ser classe, subclasse, propriedade, ou qualquer outra categorização de um qualquer domínio.

Os predicados podem também à semelhança dos restantes ser tanto URIs, como nomes prefixados que irão dar sentido à declaração que se encontra a ser afirmada.

No seguinte exemplo, temos destacado a verde todas os predicados da declaração:

mo:Michael_Jackson a mo:MusicArtist ;
dc:description "single, pop" ;
foaf:name "Michael Jackson" ;
foaf:homepage "http://www.michaeljackson.com/" .

De notar que, em Turtle, desde que a declaração esteja devidamente separada, é possível fazer várias afetações de predicados e respetivos objetos quando separando por ";" e assim realizar subsequentes afetações ao sujeito, conforme exemplo.
Aqui, temos 4 triplas distintas, nos quais todos partilham um sujeito comum: "mo:Michael_Jackson". Simplificamos assim o texto, poupando tamanho ao reduzir o número de caracteres, e ganhando na facilidade de leitura imediata.

4) **Objetos - "Com o quê"**
Os objetos, são o final de uma tripla e representam qual é a afetação que estamos a declarar. É onde acaba a ligação que acabamos de criar no grafo, refletindo-se assim como o nó final da ligação declarada.
Pode ser um URI ou um nome prefixado, ou então um valor literal de fim de cadeia, contendo um valor absoluto de fim de linha entre aspas, o qual fica inscrito na propriedade: uma string, um link para uma imagem, uma data, etc.

No exemplo já referido, temos agora destacados a azul os objetos da declaração:

mo:Michael_Jackson a mo:MusicArtist ;
dc:description "single, pop" ;
foaf:name "Michael Jackson" ;
foaf:homepage "http://www.michaeljackson.com/" .

Os objetos são assim, o objetivo final de uma declaração. É trazer o valor com o qual queremos afetar o sujeito, sendo elementares para dar significado a uma declaração.

À semelhança dos predicados, os objetos podem ser assignados em série a uma determinada propriedade e subsequente sujeito, bastando para isso separá-los por “,”.

Assim, temos então com algumas das possibilidades de “triplagens” em RDF, o seguinte exemplo:

```
mo:Sinal_Fechado a mo:Record ;
dc:date "1974" ;
mo:imagem
"http://pt.wikipedia.org/wiki/Sinal_Fechado#mediaviewer/File:Chico_Buarque-
Sinal_fechado.jpg" ;
 mo:title "Sinal Fechado" ;
 mo:track mo:FI , mo:SF ;
 mo:release mo:9010 ;
 foaf:maker mo:Chico_Buarque .
```

Em que:
A amarelo o sujeito e respetivo caractere de finalização de declaração.
A verde os predicados e respetivos caracteres de finalização de afirmação.
A azul os objetos e respetivos caracteres de separação.

Neste caso, estamos a dizer que um determinado recurso existente no domínio “mo” com o nome “Sinal_Fechado” (domínio esse que se encontra registado na secção de prefixos no início do ficheiro), pertence à classe mo:Record, tem uma propriedade dc:date com o valor “1974”, uma propriedade de imagem com um link para a mesma, um título com o nome “Sinal Fechado”, tem as faixas que são referenciados “mo:FI” e “mo:SF”, e por aí adiante.

7.8 Classes

Conforme falado anteriormente, um dos primeiros passos de descrição é a identificação dos vários “tipos de coisas” que se quer descrever. Esse “tipo de coisas”, em RDF são as classes.

Uma classe em RDF, corresponde ao conceito de um Tipo ou Categoria, similar ao conceito de classe de linguagens de programação orientadas por objeto. As classes são descritas usando recursos do RDF schema, rdfs:Class e rdfs:Resource e as propriedades rdfs:type e rdfs:subClassOf.

Supondo que uma empresa example.org, quer usar RDF para fornecer informações sobre veículos com motor. Em primeiro lugar, é necessária uma classe para representar a categoria de veículos com motor. Os recursos que pertencem a uma classe são as instâncias. Neste caso, as instâncias da classe são os recursos veículos com motor. Uma classe é qualquer recurso que tenha uma propriedade rdf:type cujo valor seja o recurso rdfs:Class.

Assim, a classe de veículos com motor seria descrita pela atribuição da classe a um URIref ex:VeiculosComMotor (usando ex: para representar o URIref http://www.example.org/schemas/vehicles , que é usado como prefixo de URIref do vocabulário example.org) e descrevendo esse recurso como uma propriedade rdf:type cujo valor é um recurso rdfs:Class:

ex:VeiculosComMotor rdf:type rdfs:Class .

7.9 Dublin Core Metadata

Metadados são dados sobre dados. São dados usados para identificar, descrever e localizar recursos de informação.

As normas de metadados NISO Z39.85-2001 e ISO 15836-2003 dizem respeito à especificação formal do Dublin Core Metadata Element Set (DCMES) e correspondem a um conjunto de elementos, propriedades, para descrever documentos (e gravar metadados) de um qualquer domínio de conhecimento.

O conjunto de metadados Dublin Core destina-se a ser adequado para uso de ferramentas de descoberta de recursos na Web, Web Crawlers, utilizados por mecanismos de pesquisa populares na Web. O DCMES usa quinze elementos para descrever qualquer recurso de informação:

- Title (título ou nome atribuído ao recurso).

- Creator (entidade responsável pela criação ou existência do recurso).
- Subject (assunto e palavras-chave que caracterizam o conteúdo do recurso).
- Description (descrição ou resumo do conteúdo do recurso).
- Publisher (entidade responsável por editar, publicar ou manter acessível o recurso).
- Contributor (entidade responsável por qualquer contribuição para o conteúdo do recurso).
- Date (data inerente à criação ou publicação do recurso).
- Type (tipo, função, natureza ou gênero do recurso).
- Format (formato do recurso: físico ou digital, tamanho ou duração).
- Identifier (referência para identificar o recurso num determinado contexto: URI, ISBN, etc.).
- Source (referência a um recurso de onde o recurso actual deriva).
- Language (língua do conteúdo intelectual do recurso: pt, en, fr).
- Relation (referência a um recurso relacionado).
- Coverage (extensão, alcance ou âmbito do recurso).
- Rights (gestão dos direitos inerentes ao recurso: direitos de propriedade intelectual, direitos de autor, entre outros).

Estes elementos são facultativos e podem ser repetidos sempre que necessário.

O exemplo abaixo, representa a descrição simples de um conjunto de recursos em RDF usando o vocabulário Dublin Core:

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:dc="http://purl.org/dc/elements/1.1/">
   <rdf:Description rdf:about="http://www.dlib.org">
    <dc:title>D-Lib Program - Research in Digital Libraries</dc:title>
    <dc:description>The D-Lib program supports the community of people
     with research interests in digital libraries and electronic
     publishing.</dc:description>
    <dc:publisher>Corporation For National Research Initiatives</dc:publisher>
    <dc:date>1995-01-07</dc:date>
    <dc:subject>
```

```
        <rdf:Bag>
          <rdf:li>Research; statistical methods</rdf:li>
          <rdf:li>Education, research, related topics</rdf:li>
          <rdf:li>Library use Studies</rdf:li>
        </rdf:Bag>
      </dc:subject>
      <dc:type>World Wide Web Home Page</dc:type>
      <dc:format>text/html</dc:format>
      <dc:language>en</dc:language>
    </rdf:Description>
</rdf:RDF>
```

7.10 Ontologias

Embora o RDF forneça um modelo para expressar metadados associando propriedades a recursos, o mesmo não é suficiente para entender o que o recurso representa, ou seja é necessário dar um contexto e significado de forma a evitar ambiguidades. (W3C, Standards, Semantic Web, Ontologies, s.d.)

Uma ontologia ou vocabulário, OWL (Web Ontology Language), é um grupo de classificação de conceitos e relações (também chamados de termos).

O papel das ontologias na Web Semântica, passa por ajudar na integração dos dados, quando por exemplo, existe ambiguidade dos termos usados nos datasets (linked data), ou quando um pouco de conhecimento extra (inferência) pode levar à descoberta de novos relacionamentos.

As ontologias são usadas para classificar os termos de uma aplicação específica, caracterizar possíveis relacionamentos e definir possíveis restrições ao uso desses termos. São a base da construção das técnicas de inferência da Web Semântica.

Também podemos dizer que uma ontologia é uma forma de representar o conhecimento através de hierarquias elementares, pode ser vista como uma taxonomia formada por classes e

subclasses de objetos, relacionadas entre si, à qual se junta um conjunto de propriedades e regras de inferência. A declaração das propriedades de cada classe, permite especificar o conhecimento inerente aos conceitos de um determinado domínio. A especificação das regras através de ontologias permite representar explicitamente a semântica dos dados.

Esta taxonomia permite, formar uma estrutura onde propriedades são atribuídas a determinadas classes e os objetos que pertencem a esta classe herdam as suas características.

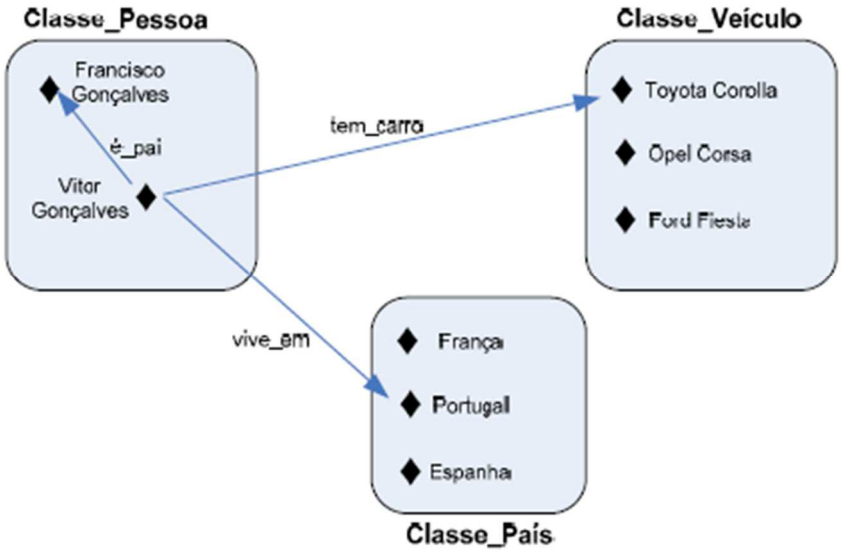

Figura 11 - Componentes básicos de uma ontologia

De uma forma genérica, os componentes básicos de uma ontologia:

- **Classes** - Correspondem a grupos, conjuntos ou coleções de objetos normalmente organizados numa taxonomia. Exemplo: Pessoa, Veículo, País.
- **Relações** - Representam os tipos de interações entre os conceitos de um domínio. Exemplo: a relação "é_pai".
- **Atributos** - São propriedades, características ou parâmetros que os objetos podem ter e partilhar. Exemplo: o objeto "Toyota Corolla" da classe Veículo poderia ter como atributos: Nome - Toyota, Versão - Corolla, Numero_Portas - 3, etc.
- **Instâncias** - Correspondem às concretizações dos conceitos e relações estabelecidas na ontologia, ou seja, são utilizadas para representar elementos específicos ao nível mais baixo, ou seja, os próprios objetos. Exemplo: "Toyota Corolla" é membro da classe Veículo e Portugal é membro da classe País.

A OWL possui três sub-linguagens:

- **OWL Lite** – sub-linguagem que fornece um menor grau de expressividade. Fornece suporte para aqueles que necessitam primeiramente de uma hierarquia de classificação com restrições simples. Por exemplo, suporta restrições de cardinalidade, porém só permite valores de cardinalidade 0 ou 1.
- **OWL DL ou OWL Description Logic** – sub-linguagem que inclui todos os construtores OWL para fornecer um maior grau de expressividade onde todas as conclusões são completa e garantidamente computáveis (computational completeness) e todos os processamentos terminam num tempo finito (decidability).
- **OWL Full** – sub-linguagem que fornece o máximo de expressividade e liberdade sintática, permitindo que o vocabulário predefinido de um RDF(S) ou OWL seja alargado ou combinado, mas sem garantia computacional. Por exemplo, uma classe pode ser tratada simultaneamente como uma coleção de indivíduos ou como um único indivíduo. Assim, os mecanismos de inferência poderão não derivar conclusões ou poderão não processar num determinado tempo finito.

Cada uma destas sub-linguagens é uma extensão da sua predecessora. Inicialmente poderemos optar por OWL Lite, mas à medida que necessitamos de um maior nível de expressividade, teremos que passar para OWL DL ou, eventualmente, para OWL Full.

Os axiomas (proposições lógicas usadas para modelar declarações sempre verdadeiras sobre os conceitos e relações), e as regras de inferência (representando a semântica que favorece a partilha de conhecimento entre os agentes) permitem definir as restrições.

As regras assumem um papel importante, pois permitem definir mecanismos para gerar novos relacionamentos com base nos existentes um pouco à imagem da linguagem de programação Prolog, permitem também analisar o conteúdo dos dados e descobrir possíveis inconsistências. (RIF - Rule Interchange Format, que especifica o formato para codificar e trocar as regras lógicas). (W3C, Standards, Semantic Web, Inference, s.d.)

Tomando como exemplo um dataset que inclui o relacionamento (Flipper isA Dolphin), uma ontologia pode declarar que “todo o golfinho é também um mamífero” (Mammal). Isto

significa que um programa de Web semântica entende a relação a noção de “X é também Y” e pode adicionar a declaração ((Flipper isA Mammal) ao conjunto de relações. Podemos afirmar que foi descoberta uma nova relação que não existia nos dados originais.

Outro exemplo, é expressar o facto “se duas pessoas têm o mesmo nome, homepage e endereço de email, então são idênticas”. Neste caso, a “identidade” de dois recursos pode ser descoberta via inferência.

O uso de ontologias permite por um lado pesquisas mais rápidas, uma vez que os agentes apenas pesquisam documentos que referem a informação desejada com base nas relações, conceitos e significados, em vez de todas as páginas Web que contêm as palavras chave ambíguas. Por outro lado, auxilia a comunicação e colaboração efetiva entre pessoas e aplicações computacionais.

Em suma, uma ontologia ou vocabulário, é uma descrição ou especificação formal dos conceitos e dos relacionamentos que podem existir e que podem ser usados por agentes, cuja finalidade principal é permitir a descoberta, a partilha e a reutilização do conhecimento.

Os vocabulários são indicados por um URI. Neste URI são encontradas referências às propriedades que cada vocabulário utiliza para formar um contexto. Para encontrar uma propriedade num vocabulário específico, geralmente, basta juntar o nome da classe ou propriedade na URL do vocabulário. Como exemplo, o vocabulário Schema.org. Para visualizar o type Person, basta juntá-la à URI do vocabulário: http://schema.org/Person .

Alguns vocabulários disponíveis:

- Schema.org (http://schema.org/) é uma iniciativa da comunidade, iniciada e patrocinada pelo Google, Yahoo!, Microsoft e Yandex, para criar vocabulários.
- FOAF, ou Fried-of-a-Fried (http://www.foaf-project.org/), é um conjunto de termos que descrevem a relação entre pessoas.
- SIOC (http://sioc-project.org/) é uma ontologia de termos que podem ser usados para descrever comunidades online.
- DOAP (https://github.com/edumbill/doap/wiki) é o projeto de um vocabulário para descrever projetos de software, principalmente projetos open source.

- Music Ontology (http://musicontology.com/) é um vocabulário para descrever coisas do meio musical como artistas, álbuns, músicas, arranjos etc.

O W3C oferece variedade de técnicas para descrever e definir diferentes formas de vocabulários num formato padrão: RDF e RDF schemas, Simple Knowledge Organization System (SKOS), Web Ontology Language (OWL) e Rule Interchange Format (RIF).

A escolha entre estas tecnologias depende da complexidade e do rigor requeridos por uma aplicação.

7.11 SPARQL

SPARQL, Protocol and RDF Query Language, é uma linguagem de consulta a dados estruturados - datasets produzida pelo W3C RDF Data Access Working Group (DAWG) .

Uma query SPARQL pode consistir numa estrutura simples de duas cláusulas: Select e Where.

O Select identifica as variáveis que vão aparecer no resultado da query, a cláusula Where mostra o padrão básico do grafo que fazem "match" com os dados.

A maioria dos datasets disponíveis, oferecem acesso via SPARQL EndPoint, que são interfaces online que permitem a realização de consultas diretamente por um endereço na Web. Exemplo: http://dbpedia.org/sparql , http://linkedgeodata.org/sparql.

Podemos ilustrar uma consulta usando a base de dados de triplas RDF online da DBPedia http://dbpedia.org/isparql/ :

Selecionar Países que contenham "Republic" no nome e tenham sido estabelecidos antes de 1920:

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX type: <http://dbpedia.org/class/yago/>
PREFIX prop: <http://dbpedia.org/property/>
SELECT ?lbl ?est
```

```
WHERE {
  ?country rdfs:label ?lbl .
  FILTER(bif:contains(?lbl, "Republic")) .
  ?country a type:Country108544813 ;
     prop:establishedDate ?est .
  FILTER(?est < "1920-01-01"^^xsd:date) .
}
```

Resultado:

lbl	est
Republic of Macedonia	1919
Republic of Macedonia	1913
Republic of Saugeais	1947
Czech Republic	1918-10-28
Dominican Republic	1863-08-16
Republic of Macedonia	1929

Resultado em XML:

```
<sparql xmlns="http://www.w3.org/2005/sparql-results#" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/sw/DataAccess/rf1/result2.xsd">
 <head>
  <variable name="lbl"/>
  <variable name="est"/>
 </head>
 <results distinct="false" ordered="true">
  <result>
   <binding name="lbl"><literal xml:lang="en">Republic of Macedonia</literal></binding>
   <binding name="est"><literal datatype="http://www.w3.org/2001/XMLSchema#integer">1919</literal></binding>
  </result>
  <result>
   <binding name="lbl"><literal xml:lang="en">Republic of Macedonia</literal></binding>
   <binding name="est"><literal datatype="http://www.w3.org/2001/XMLSchema#integer">1913</literal></binding>
  </result>
  <result>
   <binding name="lbl"><literal xml:lang="en">Republic of Saugeais</literal></binding>
   <binding name="est"><literal datatype="http://www.w3.org/2001/XMLSchema#integer">1947</literal></binding>
  </result>
  <result>
   <binding name="lbl"><literal xml:lang="en">Czech Republic</literal></binding>
   <binding name="est"><literal datatype="http://www.w3.org/2001/XMLSchema#date">1918-10-28</literal></binding>
  </result>
```

```
  <result>
   <binding name="lbl"><literal xml:lang="en">Dominican Republic</literal></binding>
   <binding name="est"><literal datatype="http://www.w3.org/2001/XMLSchema#date">1863-08-
16</literal></binding>
  </result>
  <result>
   <binding name="lbl"><literal xml:lang="en">Republic of Macedonia</literal></binding>
   <binding name="est"><literal
datatype="http://www.w3.org/2001/XMLSchema#integer">1929</literal></binding>
  </result>
 </results>
</sparql>
```

8 Generatron - Gerador de Dados Sintéticos para Testes de Qualidade de Dados

8.1 Motivação

Este projeto, consiste no desenvolvimento de uma ferramenta de geração de dados exemplo, formatados de acordo com critérios específicos da estrutura do esquema desejado.

A oferta existente de ferramentas no mercado que produzam dados estruturados avulso de tipos especificados, não existe no mundo da web semântica. Nas bases de dados relacionais ou dados livres, já existe, embora, sem erro induzido e respetivo controlo de tipo e percentagem pseudoaleatória, deixando uma importante oportunidade de contributo (exemplo de gerador de BD relacional: https://www.generatedata.com/).

Adicionalmente, não existem ferramentas que de um modo simples importem um ficheiro em formato de RDF, permitam a sua análise, visualização, reestruturação e/ou criação pelo utilizador num cockpit centralizado, e exportem tanto para RDF, como convertendo o que foi importado para relacional em SQL, ou até mesmo em CSV, num formato mais livre de dados agregados gerados aleatoriamente, independentemente do seu conteúdo ser RDF.

Finalmente, a possibilidade de induzir erros em dados que sejam gerados a pedido, controlados pelo utilizador, permite um novo tipo de análise na web semântica, já que por via da comparação, as similaridades nos padrões das bases de dados gerados com erros controlados versus bases de dados com dados produtivos, permitem uma inferência expectável de erro em produção. Uma vez que a web semântica é totalmente mutável, e consequentemente, suscetível a erros de categorização e/ou valores inscritos erroneamente, torna-se elementar existirem métodos de geração de dados com erro induzido controlados pelo utilizador.

Criámos assim o Generatron, programa que gera dados para RDF, SQL e CSV com parâmetros customizados e erro induzido em dados conforme especificação do utilizador.

Não só tem a capacidade de produzir este tipo de outputs, como também a partir de inputs, criar estruturas de dados inferidas vindas de um ficheiro à escolha do utilizador,

proveniente de uma base de dados de web semântica (RDF) com a sintaxe Turtle. Pode-se também criar uma nova estrutura de raiz, dando total controlo ao utilizador no seu domínio.

8.2 Objetivos

- Permitir a importação e leitura de um ficheiro RDF, e inicialização livre vazia.
- Dar ao utilizador a escolha do esquema de dados de output: RDF, relacional e CSV.
- Permitir a criação, remoção e associação de classes, propriedades e prefixos.
- Permitir a identificação do tipo de dados a ser gerado em cada coluna / propriedade.
- De acordo com o tipo de dados, mostrar um exemplo do formato de dados esperado.
- Permitir a escolha dos tipos de dados mais comuns tais como: dados pessoais, dados geográficos, dados numéricos, dados relacionados com publicações de artigos e meio musical.
- Permitir a inclusão de novos tipos de dados.
- Permitir ao utilizador determinar a percentagem de dados com problemas de qualidade que devem ser inseridos nos dados gerados e especificar qual o problema de qualidade de dados a ser inserido.
- Informar o total de registos a serem gerados pela aplicação.

Decidimos fazer este algoritmo na plataforma Java, via netbeans, com recurso a fxml e JavaFX. Esta decisão justifica-se:

- Pelo facto de que, cada janela fxml e JavaFX têm o layout num ficheiro de texto fxml, que é depois controlado por código java com recurso a JavaFX que corresponde ao seu controlador.
- Pela facilidade que o fxml fornece nas alterações e dinamismo dos layouts.
- Por ser um formato baseado em XML organizado em texto para dinamismo gráfico, que tem de forma organizada uma outra camada com código java com a função de ser seu controlador.

8.3 Importação

É no arranque do algoritmo, que pode ser realizada a importação de um ficheiro RDF:

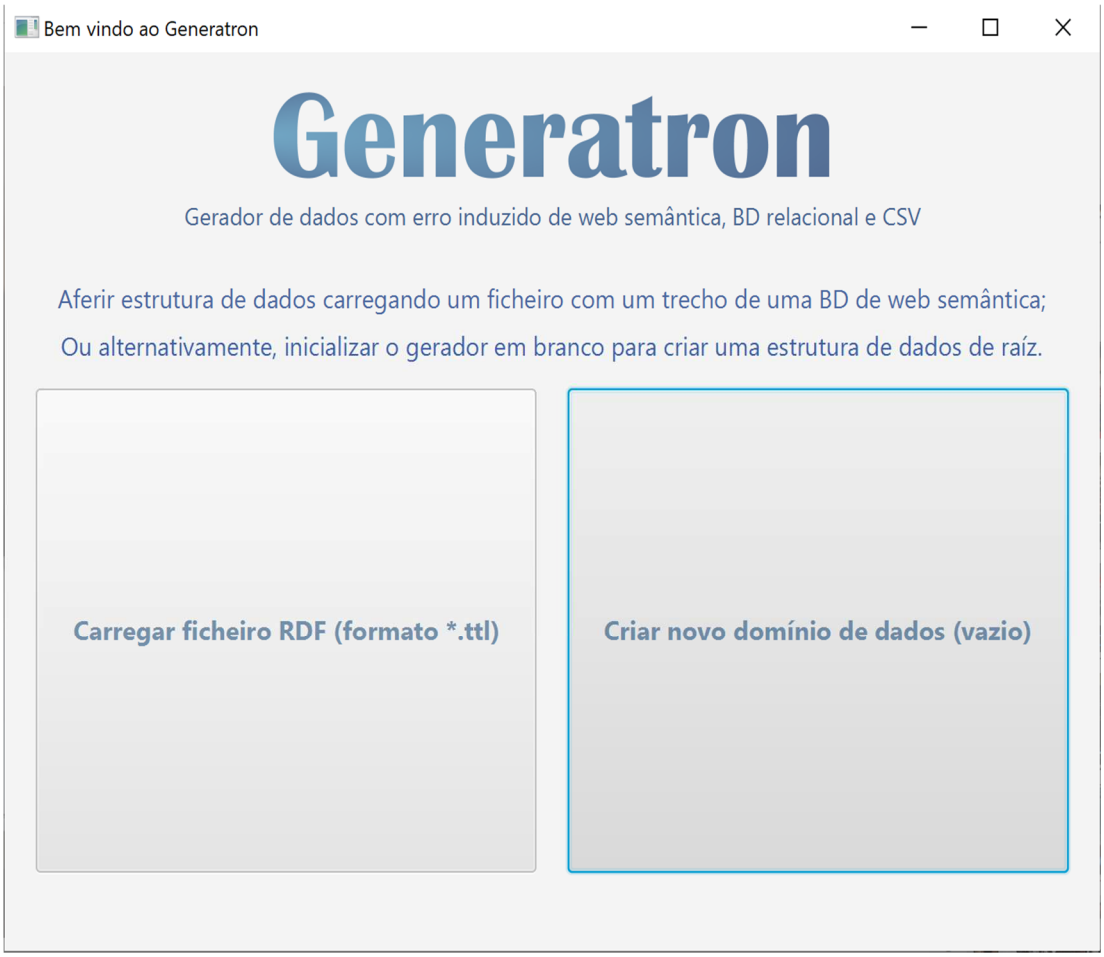

Figura 12 -Ecrã inicial do Generatron

Para importação apresentamos o exemplo que se segue. É proveniente de um dataset de uma base de dados de web semântica de música, e corresponde a um excerto de uma base de dados em RDF, a qual demonstra uma correta formatação de sintaxe em Turtle. Como tal pode ser importada, pelo mecanismo nativo do Generatron quando selecionado para importação com fiabilidade e sem erros.

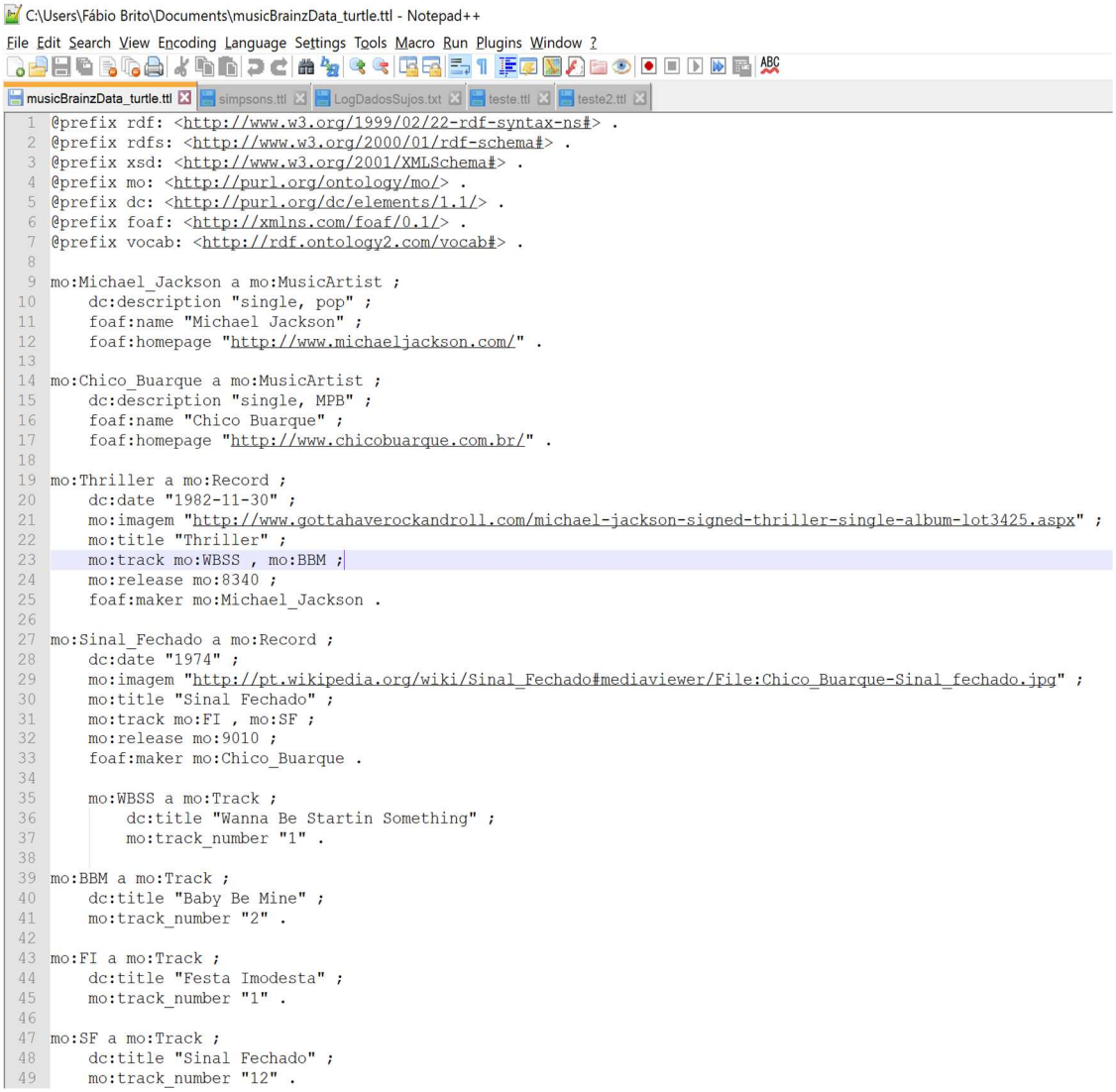

Figura 13 - Exemplo de ficheiro Turtle para input do algoritmo

8.4 Deteção de erros de importação

Para os casos em que existem erros de sintaxe no ficheiro que se encontra a importado, o Generatron possui metodologias de deteção de erros e escrita em log de todas as imparidades lógicas que encontre. No ficheiro de log consta o erro, a operação que se encontrava a ser realizada e a causa do problema em questão. Seguem alguns exemplos de tipos detetados:

No caso de erros de sintaxe:

- Não encontrar um predicado numa tripla.
- Erros de sintaxe na definição de uma classe.
- Tentativa de criação de um prefixo que já existe com URI.
- Erro de sintaxe na propriedade.

Em caso de erros lógicos:

- Erro na associação de uma propriedade.
- Não encontrar uma classe ou indivíduo para associação de uma propriedade.
- Não encontrar uma classe para associar um indivíduo

Em baixo, um exemplo de um erro lógico escrito em log quando encontrada imparidade:

Figura 14 – Exemplo de erro lógico escrito no log.

Não obstante da existência de erros, o Generatron prossegue com a análise do ficheiro até ao seu final. Nessa altura, é apresentado um breve sumário sob a forma de um popup simplista, com as analíticas do que foi inferido a partir do ficheiro, sendo que, em caso de ocorrência de erros nesse processo, é indicado ao utilizador para ver o log para mais detalhes.

O mecanismo de deteção de erros é importante para nos certificarmos que o erro que não controlamos, externo, é mitigado, e termos confiança nos dados que são apresentados.

No entanto, dentro do universo do que é uma sintaxe ordeira e imaculada, cabe ao Generatron ter a capacidade de inferência correta para que exista uma classificação estrutural assertiva.

8.5 Processo de importação

O processo de importação de dados RDF decompõe-se em várias fases. Dada a complexidade e variabilidade dos dados passíveis de ser importados, têm de ser encontrados padrões para minimizar o risco de erros lógicos de importação, assim como uma metodologia.

Aqui reside a primeira e maior dificuldade na implementação deste projeto, na medida em que, através de várias tentativas e erros de importação de várias bases de dados, nos deparámos com diversos problemas que careceram de intervenção intensiva. Como exemplo:

- Um espaço divide um sujeito de um predicado e de um objeto. No entanto, é perfeitamente admissível existirem espaços em strings com conteúdos literais.
- As vírgulas, ponto e vírgula e ponto são delimitadores de sintaxes de triplos. No entanto, em strings literais, estes existem de um modo bastante recorrente.
- Um sujeito, predicado ou objeto, pode ser escrito com um prefixo separado do seu nome dividido por ":". No entanto, no caso de ser um URI, o oposto é verdade.
- Um sujeito tanto se pode afirmar como uma classe, como um indivíduo de uma classe, ou até simplesmente um recurso, sem qualquer classe envolvida pelo predicado.
- Os valores dos objetos tanto podem ser literais, como referências para outros recursos.
- Nem todos os caracteres são admitidos na sintaxe Turtle. Nas URIs, existem exceções.
- Os comentários não devem ser considerados em circunstância alguma na importação.
- As pontuações segregadoras, podem estar no fim da palavra ou início da subsequente.

Para mitigar estas condicionantes, entre outras, e dado que valores de propriedades que sejam apontadores para recursos não literais não nos interessam uma vez que não findam o objetivo final, que é gerar dados aleatórios finais avulsos, adotaram-se as seguintes medidas iniciais:

- O Generatron verifica linha a linha se não está vazia e se não é um comentário até que o caractere final da linha seja um ".", anexando as linhas num buffer temporário.
- Quando encontra o denotador final, o algoritmo entende a concatenação do buffer como uma declaração final, e nela realiza as seguintes operações antes de prosseguir.
- \" -> Aspas dentro de strings literais são removidas, uma vez que confundem o parse realizado posteriormente no programa, onde tratamos objetos literais.
- "" -> Aspas sem texto são substituídas por "SemTexto".

- Textos entre aspas são substituídos por “”, entendidos posteriormente.
- Espaços entre < … > são substituidos por “_Espacos_Apagados_”.
- Textos entre (…) são substituídos por apenas “(“.
- A notação de linguagens (ex.: @en) que sucedem strings literais são removidas.

8.6 Conversão do texto para objetos

Tendo realizado a limpeza primordial ao texto, já se reúnem as condições necessárias para que o generatron possa fazer: parsing, segregação dos elementos das afirmações nos espaços, e posterior análise de sujeitos, predicados e objetos, com consequente classificação conforme categorização nos predicados de forma ordeira e correta.

Nesta fase, o Generatron gera tantas triplas sob forma de objeto quantos existam na declaração, e encapsula-os num conjunto de triplas que nomeámos “Declaracao”. São estas declarações, que à posteriori, são analisadas e sob sua influência é gerado todo o conteúdo da estrutura final que no ecrã como Prefixos, Classes, Propriedades e suas associações.

8.7 Conversão de paradigma RDF para esquema relacional / CSV

Uma vez que numa base de dados relacional, o paradigma de interligação de dados é totalmente diferente dos grafos através de triplos, a conversão que decidimos dar ao programa que achámos fazer sentido foi: olhar para os dados que vão ser gerados como as linhas de informação, dentro das propriedades que se categorizam como colunas, e classes como tabelas. No caso do CSV, a situação é idêntica, e como cada ficheiro é de certa forma uma representação de uma tabela onde colunas contém os dados que serão gerados, atribui-se essa mesma estrutura como metodologia de conversão, mas em ficheiros ao invés de tabelas.

Finalmente, em ambos os casos, não faz sentido existirem prefixos pois de uma forma genérica não é comum a sua utilidade nas Bases de Dados relacionais, ou nos ficheiros de Comma Separated Values. Como tal, extingue-se a secção de prefixos assim como a sua seleção nas várias janelas de criação de tabelas e colunas, quando escolhemos estes dois tipos de output, e em vez termos classes e propriedades, tabelas e colunas, respetivamente. De resto, as

funcionalidades estruturais, mantêm-se idênticas ao RDF, embora o layout e mecanismos internos de motor do Generatron mudem drasticamente em algumas das funções.

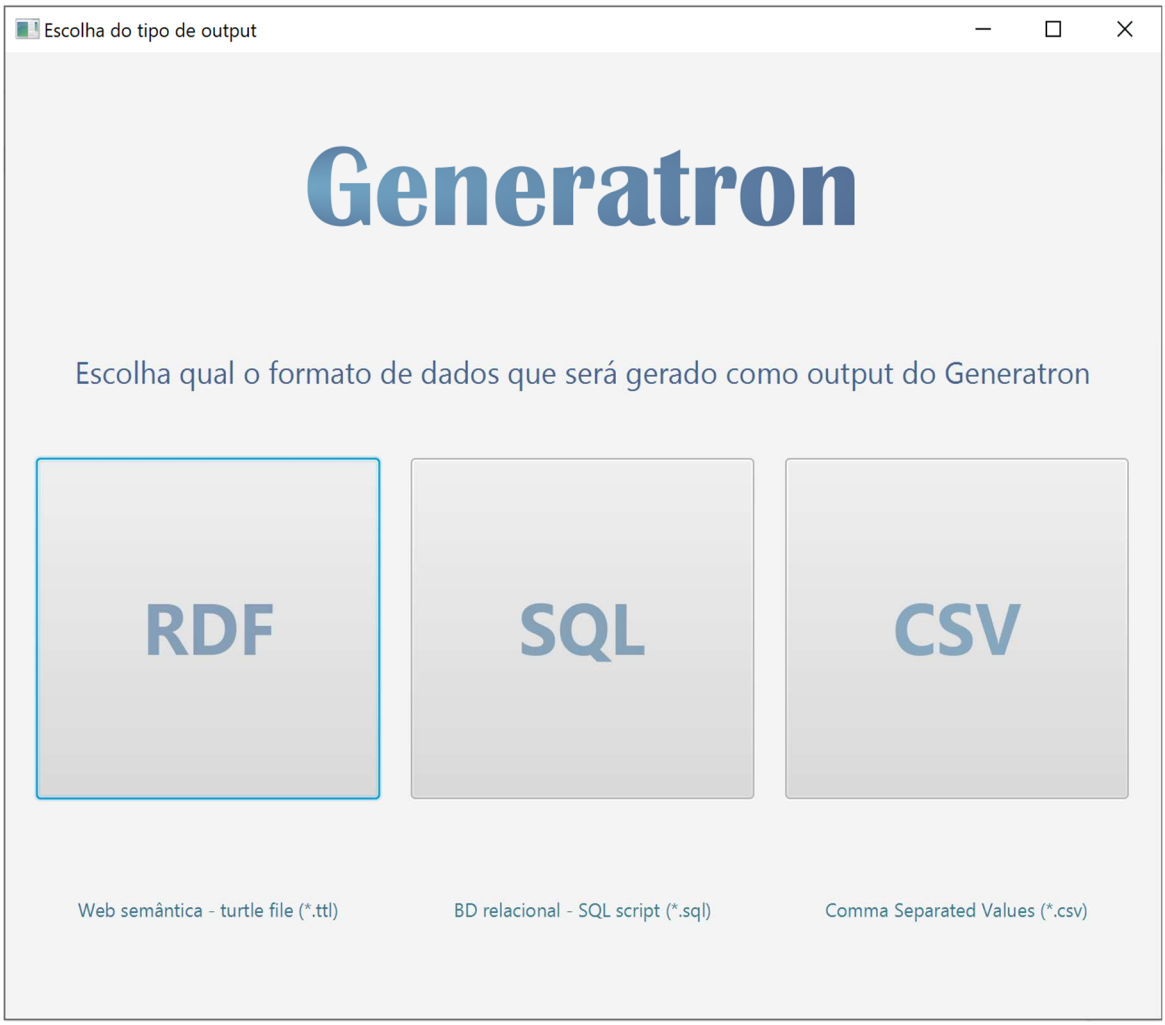

Figura 15 -Ecrã de escolha do tipo de output

8.8 Abordagem

Após análise de requisitos e análise estrutural dos ficheiros a serem importados, o algoritmo é dividido em quatro pacotes principais, cada um deles com funções dedicadas.

Nesta secção, todas as funcionalidades do algoritmo serão expostas de uma forma muito resumida, para não sobrecarregar o leitor, salientando pontos chave e de forma leve.

Para mais detalhes de nível de utilizador, será aconselhável consultar o manual de utilizador.

Já para detalhes mais profundos deverá ser consultado o código que está devidamente comentado para o efeito, que complementado com a seguinte introdução deverá ser facilmente geralmente navegável:

1) **BD**
    a) PropriedadesGlobais
    b) Bd
2) **Hierarquia**
    a) PrefixoURI
    b) Classe
    c) Propriedade
    d) Triplo
    e) Declaracao
    f) TipoDados
    g) TipoErro
    h) TipoOutput
    i) ErroPropriedadeClasse
3) **Processos**
    a) ConversaoTriplos
    c) Gerador
    d) SujadorDados
    e) LeitorFicheiro
    f) EscritaFicheiro
4) **UI_Generatron**
    a) Todas as janelas de UI.

8.9 Detalhe dos pacotes

1) BD

Neste pacote reside a base da informação que será usada em todo o programa.

Na classe Bd, tal como o nome indica, temos as listas de todos os Prefixos, Classes e Propriedades, assim como os métodos de inserção, modificação, consulta ou remoção.

Já na classe PropriedadesGlobais, temos as opções que são passíveis de ser alteradas, e que fazem "fine tuning" a todo o programa.

Definições como: quais os denotadores usados para segregar prefixos, classes, propriedades e objetos, qual o threshold a partir do qual existe um aviso para o utilizador de tamanho excessivo, número de caracteres a serem gerados, etc,

2) Hierarquia

Nesta classe, temos os "moldes" com os quais o programa cria objetos, que de forma estruturada e hierárquica contém os seus métodos e listas de subobjectos.

Segue uma imagem ilustrativa das interdependências de algumas das classes que existem e que considerámos chave no relatório para entendimento base do algoritmo.

Como referido inicialmente, para consulta das restantes, deverá ser consultado o código:

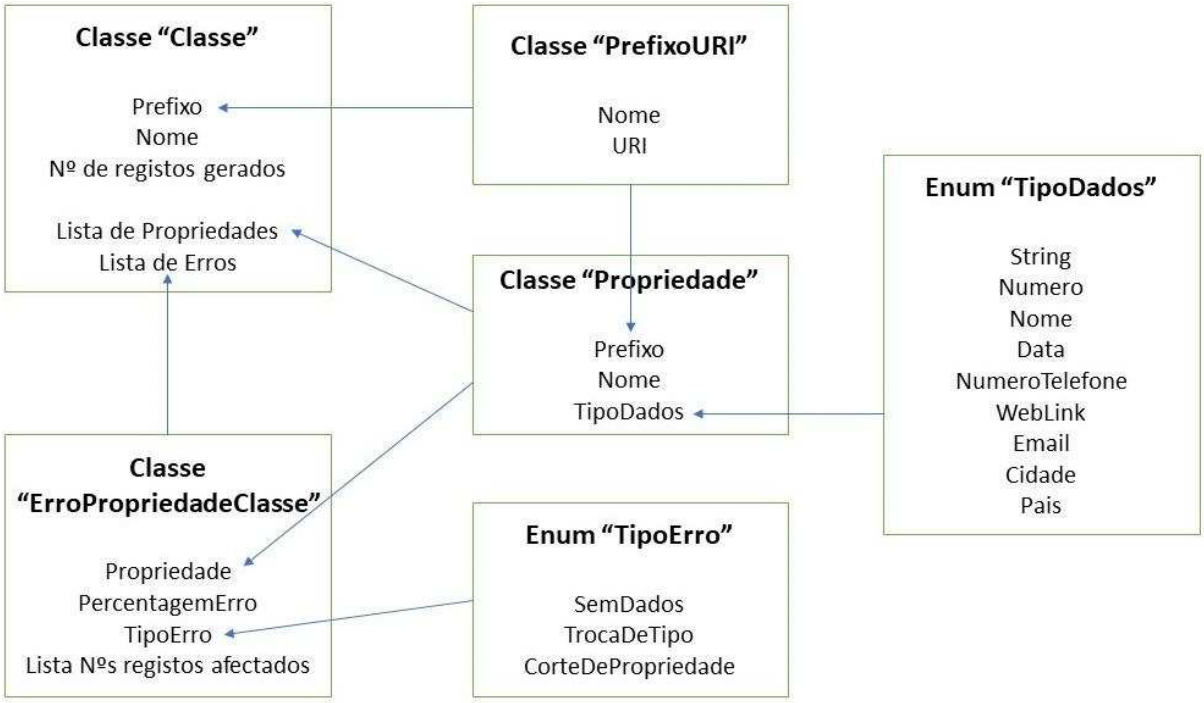

Figura 16 - Imagem ilustrativa das interdependências das classes

Um objeto da classe "Classe", tem na sua génese uma lista de propriedades, que são, por sua vez, também objetos compostos. O PrefixoURI está presente nestas duas classes, que juntas são a fundação da estrutura de dados que vai ser gerada pelo algoritmo.

Para registar os erros, existe uma classe dedicada para o efeito que acomoda tanto a propriedade em que os mesmos serão induzidos, como a percentagem, o tipo de erro que será induzido, e ainda os números de registos que serão "marcados" aleatoriamente para serem "sujos".

Através desta estrutura hierárquica, é fácil reorganizar a estrutura, e projetá-la de forma limpa e rápida para a UI, sendo que no motor os dados organizados como "matrioskas".

3) Processos

Este pacote contém maioritariamente, a base das funções que vão ser chamadas e são comuns a qualquer tipo de dado a ser exportado, ou na importação dentro do algoritmo.

Este pacote encontra-se subdividido em cinco classes:

3.a) ConversaoTriplos

Esta classe tem as funções que são responsáveis por a partir de plain text, realizar a conversão para a estrutura hierárquica montada no programa.

Além disso, tem também a responsabilidade de aquando conversão, verificar se o registo é válido, se já temos presente dados relevantes no programa, e devolve erro em caso de falha que é posteriormente guardado num log, na pasta onde se encontra o runnable do algoritmo.

É aqui que residem as funções que analisam e registam prefixos, classes e propriedades aquando processo de importação, chamado pelo LeitorFicheiro.

3.b) EscritaFicheiro

É uma classe simples, que apenas contém métodos para que a exportação dos dados gerados seja efetuada de modo segregado das restantes classes.

Nele consta também a função que escreve o log num ficheiro .txt na pasta do algoritmo.

3.c) Gerador

Classe crucial responsável pela geração dos dados.

Podemos dizer que tem os métodos que servem o objetivo principal da aplicação, na medida em que, é aqui que os dados aleatórios finais são todos produzidos.

Existe uma função chamada gerarDado, que serve de "recepção" para todos os pedidos e encaminha para a função que vai gerar o tipo de dado que lhe é passado correspondente.

Cada um dos tipos de dados passíveis de serem gerados, têm o seu próprio universo definido de possibilidades aleatórias de génese. Existem, por exemplo, para a função de gerar cidades uma lista que contém todas as capitais do mundo, assim como no caso da geração de países, nomes, números de telefone, datas, weblinks, emails, e strings e números totalmente aleatórios.

Finalmente, contem três métodos que correspondem à exportação do que se encontra estruturado na BD para os 3 tipos de output disponíveis: RDF, SQL e CSV, conforme pedido pelo utilizador. Dependendo do output, o ciclo de exportação é totalmente díspar.

3.d) LeitorFicheiro

Esta classe contém os métodos que fazem, como o nome indica, a importação e subsequente parsing de todo e qualquer tipo de ficheiro RDF Turtle, e sistematicamente alimenta a BD com a estrutura que vai sendo inferida à medida que o ficheiro é lido.

Para ler um ficheiro Turtle, esta classe recorre a três processos aqui descritos de uma resumida:

3.d.a). Ler o ficheiro, linha a linha até encontrar um fim de uma declaração (“.” no fim)

3.d.b) Envio da declaração para a função que analisa a mesma e converte o que está condensado em triplos de sujeito, predicado e objeto, independentemente de como foi exposto.

3.d.c) Envio de cada um dos triplos contidos em cada uma das declarações para registo na classe ConversaoTriplos, que é depois responsável por converter o texto e escrever na BD.

Em todos estes métodos, caso existam erros, os mesmos são escritos para um ficheiro de log.

3.e) SujadorDeDados

Esta é uma classe simples que tem como objetivo, como o nome indica, sujar os dados conforme indicações instruídas pelo utilizador.

A função predominante é a de sujarDado, que recebe como inputs uma determinada classe, propriedade e número de registo, e afere se esta afirmação em específico deve ser induzida em erro ou não, e que, em caso afirmativo fá-lo escrever o log.

Existe uma outra função que serve o propósito de, dada a percentagem instruída pelo utilizador, marcar os registos correspondentes como targets para indução de erro, tendo depois assim maneira de identificar que registos vão ser "sujos" posteriormente.

4) UI_Generatron

Aqui estão todos os ficheiros fxml, que contém a parte de arranjo gráfico, assim como as suas respetivas controladoras, com o código que "mexe" depois com o motor do algoritmo.

No início, o utilizador pode escolher se deseja importar um ficheiro Turtle para aferir a estrutura de dados do mesmo, ou se prefere criar uma nova estrutura de raiz, em branco.

Seguidamente, e após importação dos dados se for o caso, o utilizador escolhe qual é o tipo de output que deseja produzir: RDF, SQL ou CSV.

No caso RDF, a transição dos dados de importação para a estrutura é relativamente fácil. Basta uma correta associação do que foi convertido para objetos para as listas:

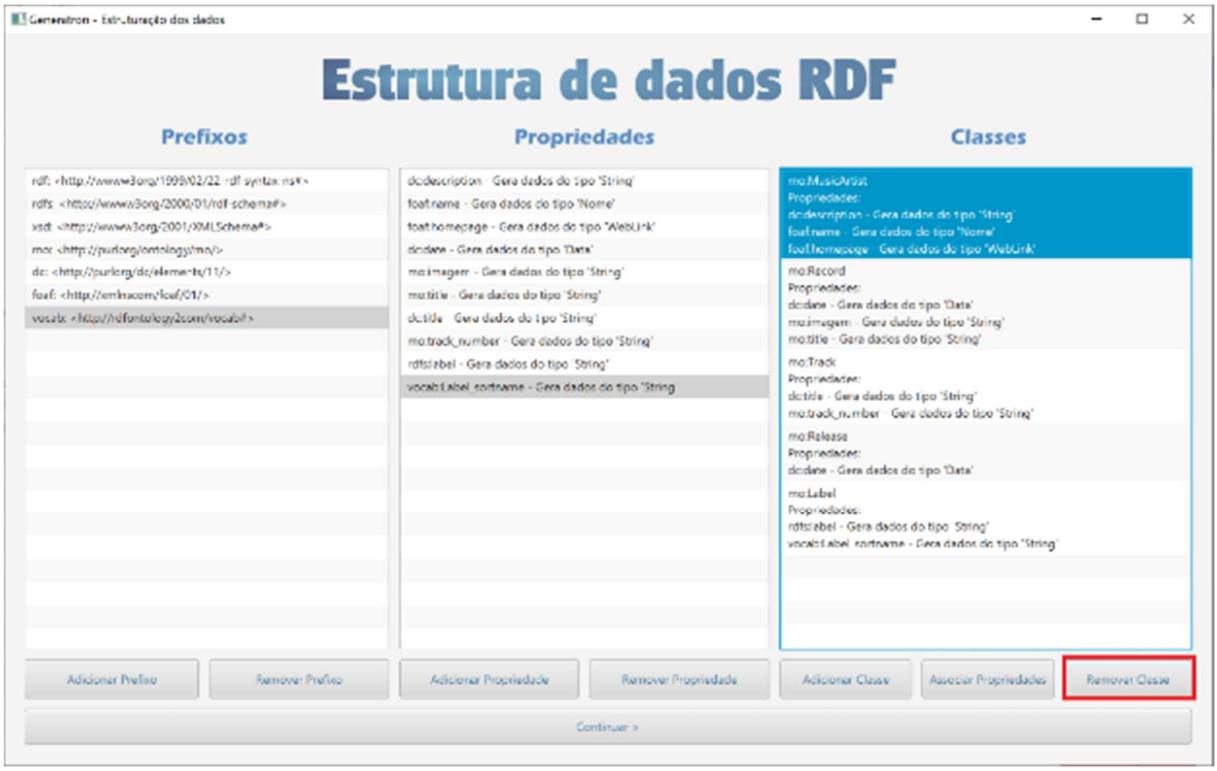

Figura 17 - Imagem ilustrativa da transição de dados de importação

Já no caso das bases de dados relacionais, SQL ou CSV (que na prática trata de colunas e linhas o que vai dar à mesma coisa), os prefixos devem ser ignorados, pois não trazem valor agregado para a estrutura relacional, enquanto que, as classes vão corresponder a tabelas e as propriedades a colunas dessas mesmas tabelas.

Neste caso, todos os caracteres que não sejam alfanuméricos são convertidos para underscores, enquanto que não são considerados os prefixos na equação de geração final:

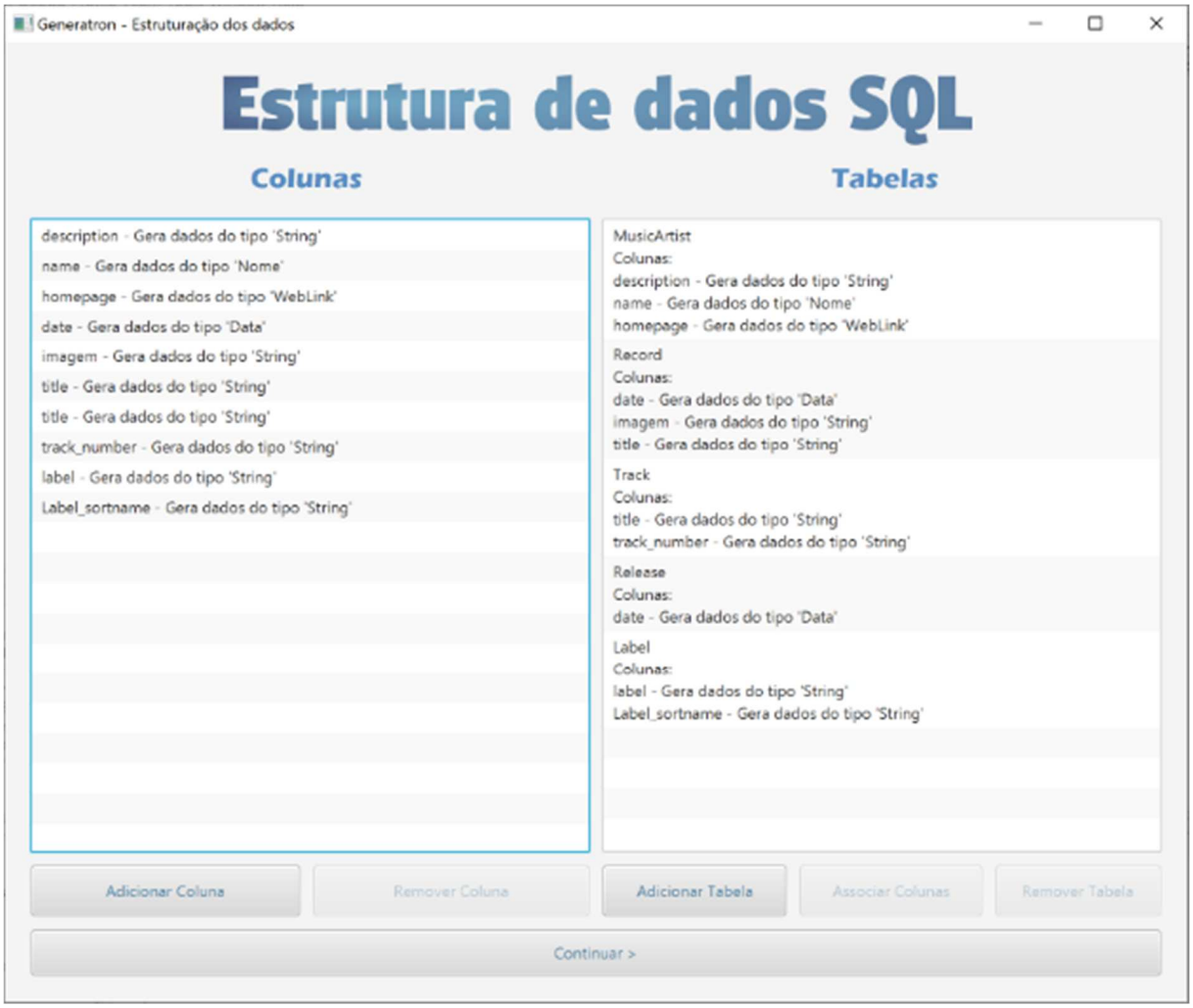

Figura 18 - Imagem ilustrativa da conversão de carateres

Independentemente do tipo de output escolhido, é possível alterar a estrutura conforme o utilizador deseje, tendo como opções a inclusão, remoção e associação das propriedades às classes e respetivos prefixos no caso do RDF, ou associar colunas a tabelas nos restantes.

8.10 Indução de erros

Finalmente, após definida a estrutura, o utilizador pode escolher atomicamente quais as Classes / tabelas que vão gerar dados, atribuindo às mesmas um número de registos a ser gerado. Nas propriedades / colunas dessas Classes / tabelas, caso queira, pode inscrever uma percentagem de erro, sendo que o tipo de erro é também livre para o utilizador escolher.

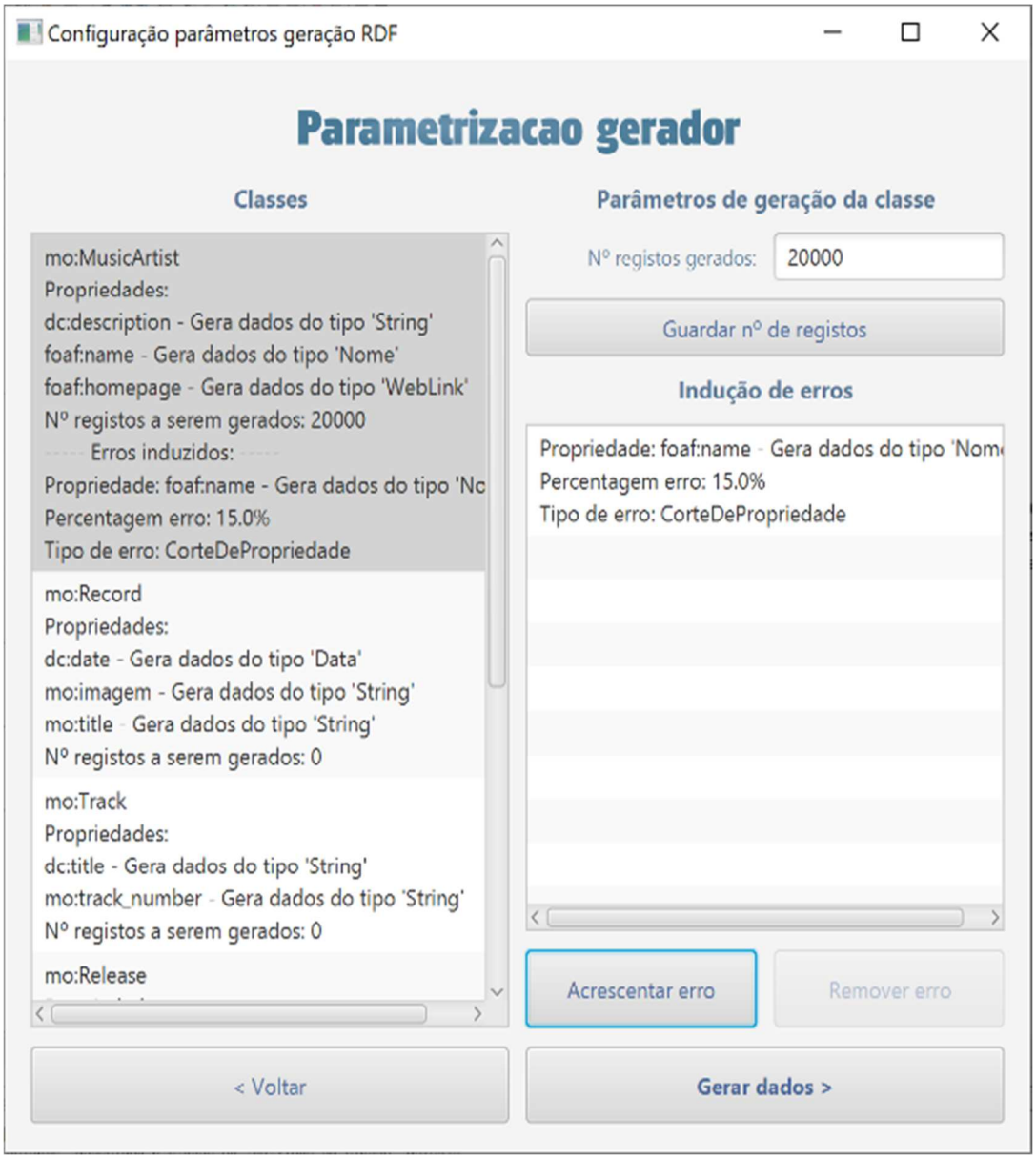

Figura 19 - Imagem ilustrativa da parametrização da estrutura.

De salientar que para cada registo que é alterado a pedido do utilizador, é escrito no ficheiro de log “LogDadosSujos.txt” na pasta da aplicação os registos impactados.

Exemplo do ficheiro de log gerado:

Figura 20 – Exemplo de ficheiro de log

Quando finalizada a parametrização, o utilizador gera os dados com o tipo de output que escolheu.

Em todas as interfaces gráficas, existe o cuidado de validar as informações que o utilizador escolhe, tentando mitigar ao máximo o “human error” que é indesejado.

Exemplo de má inserção de acrescento de erro numa propriedade:

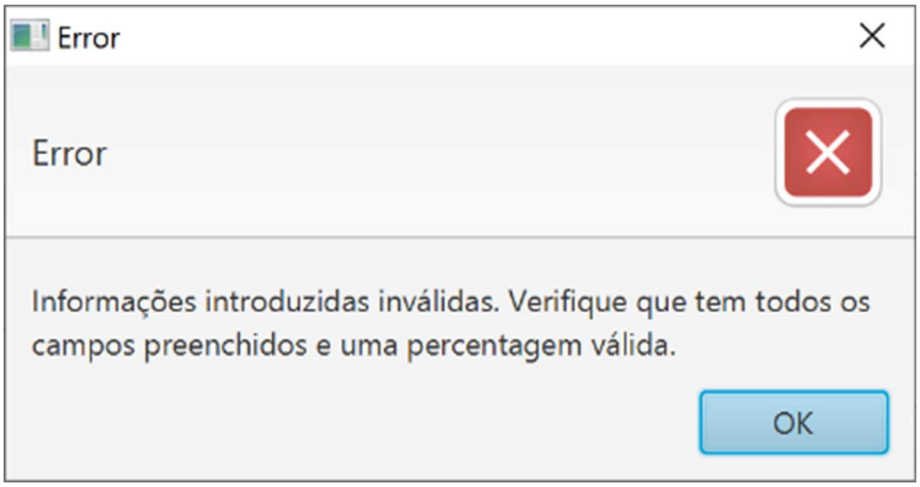

Figura 21 – Exemplo de mensagem de erro

8.11 Exportação

Na exportação dos dados, o comportamento do algoritmo é totalmente díspar em cada output. No caso RDF faz-se um loop por classe, seguidamente loop de conjunto de propriedade e valor. Exemplo de dados gerados pelo Generatron em RDF:

```
@prefix xsd: <http://wwww3org/2001/XMLSchema#> .
@prefix mo: <http://purlorg/ontology/mo/> .
@prefix dc: <http://purlorg/dc/elements/11/> .
@prefix foaf: <http://xmlnscom/foaf/01/> .
@prefix vocab: <http://rdfontology2com/vocab#> .

Individuo1 a mo:MusicArtist ;
    dc:description "pjM1a98CCJOv" ;
    foaf:name "Joséfa Mercedes Reis Furtado" ;
    foaf:homepage "http://www.2uXTB0OCfcRT.com/" .

Individuo2 a mo:MusicArtist ;
    dc:description "nc5t4xqtzNP1" ;
    foaf:name "Sandro do Paraíso Maria de São José Salgueiro Caetano" ;
    foaf:homepage "http://www.aOTqCM9yk1dq.com/" .

Individuo3 a mo:MusicArtist ;
    dc:description "9imZxU9jWxE2" ;
    foaf:name "Aldo Dino Infante Cardoso Mourato" ;
    foaf:homepage "http://www.Vd5ufIvQ9uOX.com/" .

Individuo4 a mo:MusicArtist ;
    dc:description "j6PcZDB6yRse" ;
    foaf:name "Mélanie Porciana Campos Caetano" ;
    foaf:homepage "http://www.R78rV7D6gMbU.com/" .

Individuo5 a mo:MusicArtist ;
    dc:description "O4rfNe1gPxx6" ;
    foaf:name "Romeia Stela Patrício Piedade" ;
    foaf:homepage "http://www.ho7X0jtoD2va.com/" .

Individuo6 a mo:MusicArtist ;
    dc:description "KEyrPWOaP7PX" ;
    foaf:name "Denisa Ginestal Santos Branco" ;
    foaf:homepage "http://www.taVe19Yp9ARp.com/" .

Individuo7 a mo:MusicArtist ;
    dc:description "3Q51DK3fRFfC" ;
    foaf:name "Chantal Eloisa Sampaio Brandão" ;
    foaf:homepage "http://www.yBqZhB09OxQe.com/" .

Individuo8 a mo:MusicArtist ;
    dc:description "7gCQv7ADBA1F" ;
    foaf:name "Zubaida Lúcio do Rosário Morais Roque" ;
    foaf:homepage "http://www.DfjdSGM4krEQ.com/" .
```

Figura 22 – Exemplo de dados RDF gerados pelo Generatron

No caso SQL, dentro de um loop por classe que corresponde à criação de uma tabela, é depois um loop em cada uma das propriedades expondo as colunas, e finalmente expondo os dados. Exemplo de dados gerados no Geratron em SQL:

```
CREATE TABLE MusicArtist(
        id INT NOT NULL,
        description VARCHAR(255),
        name VARCHAR(255),
        homepage VARCHAR(255),
        CONSTRAINT PK_MusicArtist PRIMARY KEY (id)
);

INSERT INTO MusicArtist VALUES(1,'89PgqgRKyxD5','Deótila Livramento Rosa Mota','http://www.JuM9vo1Ct9WD.com/');

INSERT INTO MusicArtist VALUES(2,'8dJPEUVcdLLL','Cristiano Delmar Geraldes Neves','http://www.IoecrES7s9z3.com/');

INSERT INTO MusicArtist VALUES(3,'CChqGTZfestK','Décia Sância Cardoso Neves','http://www.lH6XPEHBNHb6.com/');

INSERT INTO MusicArtist VALUES(4,'bLE5n7pH1lx4','Suria Bárbora Brito Bacelar','http://www.OOSAA4hIhCfm.com/');

INSERT INTO MusicArtist VALUES(5,'OKK7SkPII31S','Nilson Samaritano Silva Sousa','http://www.UWvmyWxdpriV.com/');

INSERT INTO MusicArtist VALUES(6,'cU6LIxEPVLuD','Dener Olindo Rola Simões','http://www.CMma0rYx9eOu.com/');

INSERT INTO MusicArtist VALUES(7,'qcOdDy8rn136','Gonzaga Sisínio Coelho Brito','http://www.90n537Eer5sd.com/');

INSERT INTO MusicArtist VALUES(8,'s12Qfp9TQvx7','Jorja Donzília Correia Vidigueira','http://www.8iQdZglP07KU.com/');

INSERT INTO MusicArtist VALUES(9,'Lc1fdjzdpRk8','Niceia Pia Jesus Alves','http://www.1QUjr6p0f53b.com/');

INSERT INTO MusicArtist VALUES(10,'ianKdJSx6NDt','Alexa Giovana Sobral Oliveira','http://www.UiFm41RhN0Gk.com/');

INSERT INTO MusicArtist VALUES(11,'pJaEbcoHX84z','Santa Maria Ticiana Espinho Neves','http://www.kTCNDMe6MEgf.com/');

INSERT INTO MusicArtist VALUES(12,'xOUWE7nhxpYk','Josselina Mirandolina Rodrigues Cardoso','http://www.V5yZbMzkMsIc.com/');

INSERT INTO MusicArtist VALUES(13,'H6TxGUAyfkTc','Fafe Romeu Vieira Brito','http://www.IWtJDyiVfBBc.com/');

INSERT INTO MusicArtist VALUES(14,'fBY5h5er3Y0F','Nessa Radija Gomes Garrido','http://www.Ba12ybdeNOuU.com/');

INSERT INTO MusicArtist VALUES(15,'eg0d3FHleQCH','Clívia Jaquelina Ferreira Chagas','http://www.Kma2virmsHRU.com/');

INSERT INTO MusicArtist VALUES(16,'tAFrkWRAyXzA','América Marília Pereira Sousa','http://www.98MPYIk69fkg.com/');

INSERT INTO MusicArtist VALUES(17,'7pXbkWeaYuMF','Gueir Marília Coelho Brandão','http://www.tVP7v7HJzVJ0.com/');

INSERT INTO MusicArtist VALUES(18,'jD5HTJxom1Ui','Adamantino Leanor Caldeira Reis','http://www.xPmr8hGItCRM.com/');

INSERT INTO MusicArtist VALUES(19,'ht8TThH2y2RV','Tude Gui Jesus Soares','http://www.hKY6LqaMo9Z0.com/');

INSERT INTO MusicArtist VALUES(20,'Un0bm73iUgH3','Antonieta Etelca Branco Sousa','http://www.5xF4c9MidmS6.com/');
```

Figura 23 – Exemplo de dados SQL gerados pelo Generatron

No caso do CSV, é feito um loop por classe que corresponde a um ficheiro específico, depois um loop por propriedade que corresponde a cada coluna, e finalmente, dado por coluna.

Exemplo de dados gerados no Generatron de CSV no Excel:

	A	B	C
1	description	name	homepage
2	YNvmz5GQDmPS	Galileu Ilundi Real Sereno	http://www.8LS5drRzMkzm.com/
3	shfd4uegf3EL	Selene Fara Guterres Lopes	http://www.WYsejtUUFFhd.com/
4	DizlNi9OBCI8	Alano Darci Veiga Guerreiro	http://www.0WCC3aByY7ON.com/
5	j5lJW9O7aP0l	Guido Leonor de Jesus Rato Dias	http://www.yhLet9pj6n9M.com/
6	n6X9Zq6rl4Tm	Alvarim Odeberto Roque Gralha	http://www.x5lpi9NgOiSP.com/
7	iogGeqoSbscC	Jabes Eliezer Coelho Saraiva	http://www.jaaIWdqkEF7N.com/
8	9SjiqBcvo4qR	Soledade Anael Roque Matos	http://www.krJWXVzpjrsU.com/
9	yaOQ3gczJ5xW	Zobaida Bernardim Caetano Rola	http://www.Qexuap8p1CkF.com/
10	TMhQGkiUchZ9	Boavida Viviane Sousa Pinheiro	http://www.JbhSUgl01lhx.com/
11	h7itTlsLPxdo	Delfino Miraldina Matos Coelho	http://www.7VtCnV93hQcS.com/
12	VSQiZy0jJa8A	Lua Gonzaga Gomes Costa	http://www.8ri5KBmQsVL6.com/
13	br0p8Zt41777	Dircea Calila Pinheiro Cardoso	http://www.99MQS15ss31D.com/
14	Wna9bon9FTIn	Murilo Sesira Carvalho Rodrigues	http://www.J0KayaDx0Yxz.com/
15	H7FouqRmXPrm	Joel Magna Geraldes Teixeira	http://www.5ldTersYV4Uj.com/
16	AmePv50rFXAR	Celisa Rolim Domingos Cardoso	http://www.sGmpLvvicrOF.com/
17	DUkOJpDiDQ8Y	Odeberto Sarah Borges Vieira	http://www.8RcQE8vZuCZU.com/
18	xoC1zD9FJkbm	Dulcelina Martins Furtado Dias	http://www.O8Kjeor5KMZG.com/
19	7zsCiRpnq73f	Aldo Dino Marco Almeida Rocha	http://www.m1j9WdKG118B.com/
20	bq2AJTGPTmD8	Darque Vianei Tavares Geraldes	http://www.keuP5agKClJy.com/
21	bxZLzhmSiIEn	Nilson Ãtila Sobral Dias	http://www.cfOWrAL0iGWI.com/
22	Opa3HJys4ptd	Vasco Ianesis Furtado Teixeira	http://www.hFZcrtvEXXYn.com/
23	K8igEEERmHhP	Ornela HerÃ©dia Rato Garrido	http://www.e4vjynAQXa6R.com/
24	ON8gYrQtzJlx	Jezabel Marcus Pimenta Nunes	http://www.3WGAtZTsvVEy.com/
25	R0tPnXViSlMC	Sandra Juvenal Ferreira Piedade	http://www.GWG59IILEH38.com/
26	TNWY7xbnOq4t	Mauro Pia Barroso Saraiva	http://www.lt4PkJ2o4VEu.com/
27	N4lz1f3zRP6G	Enide Hulda Mota Pimenta	http://www.PQnd2vQY7rrx.com/
28	G9GHj9Bb8HuB	Dorinda Nara Vidigueira Branco	http://www.8SEghPV2hfsi.com/
29	qPWHIlzcXb40	Soraia Eleine Portas Real	http://www.IKYa7cksNHh3.com/

Figura 24 – Exemplo de dados CSV gerados pelo Generatron

9 Conclusão

Este desafio, como projeto de final de curso simboliza o epítome da entrega destes futuros engenheiros, sendo um marco definitivo na história académica de cada um de nós.

A contribuição que nos propusemos a dar e entregarmos, levou de nós muita entrega, dedicação, interesse e satisfação no resultado final que acabou por ser produzido.

O facto da inexistência de algoritmos que preencham o vazio onde o Generatron se situa, de: permitir gerar dados aleatórios controlados a pedido do utilizador com a possibilidade de erro induzido, realizar conversão de dados de web semântica para bases de dados relacionais, inferir através de excertos importados qual a estrutura de uma base de dados de web semântica, permitir ao utilizador a visualização e adulteração dos dados e sua estrutura de forma gráfica limpa e simples com o intuito primordial de potenciar o conhecimento, até então, não trilhado, de erros existentes em bases de dados produtivas de web semântica de uma maneira tão intuitiva, foi uma abordagem empírica e recompensadora para nós.

A realização do entendimento de todo o conceito de web semântica e de tudo o que daí está para vir, das suas sintaxes estruturadas, das analíticas dos dados e dos erros induzidos, foi um desafio que ao ser implementado numa ferramenta que consegue limpar, analisar, deduzir a estrutura de grafos de ficheiros que vê à medida que lê conteúdo do que o utilizador escolhe importar, e que permite criar, alterar ou remover todo e qualquer parâmetro enquanto personaliza toda a geração de dados e induz até erros exportando finalmente em formato RDF, SQL e CSV conforme escolha, concluiu-se num resultado gratificante, com o qual aprendemos algo que nunca tínhamos tido contato anteriormente, ao mesmo tempo que superou todas as nossas expectativas contempladas inicialmente e que levamos connosco.

Felizmente, não só os objetivos inicialmente propostos foram totalmente cumpridos, como acabaram por ser superados, ao serem incluídas novas funcionalidades além daquilo que foi por todos acordado e era expectável, tal como a administração e gestão de erros de importação que são externos e estão fora do controle do nosso algoritmo, como a conversão excecionalmente limpa de grafos RDF com triplas para relacional de linhas e colunas.

Assim, podemos afirmar que a nossa contribuição acaba por ter um sabor especial, na medida em, que a ferramenta foi inovadora em vários sentidos.

Foi um esforço adicional que quisemos ter, pelo facto de ser um assunto com o qual não lidamos no dia a dia, e bastante proveitoso.

Em termos de trabalho futuro no algoritmo, onde poderiam ser consideradas melhorias, acreditamos que uma gestão isolada das propriedades globais seria algo positivo, pois permitirá a alteração de algumas variáveis que fazem sentido, sem necessidade de alterar código *hardcoded*. Como exemplo, temos thresholds, denotadores ou o número de caracteres.

Assim, ficamos felizes por podermos deixar o nosso pequeno contributo ao mundo, com apenas mais uma ferramenta que poderá trazer mais valias que eram inexistentes até então, e que tivemos a audácia de tentar engenhar, as quais esperamos ser do agrado do leitor.

10 Bibliografia

Berners-LEE. (2006). *Data, Information, Knowledge and Wisdom.* Obtido de http://www.w3.org/DesignIssues/LinkedData.html

Christian Bizer, T. H.-L. (2009). *Linked Data - The story so far.* Obtido de https://www.semanticscholar.org/paper/Linked-Data-The-Story-So-Far-Bizer-Heath/55c9ac390fee7d6b7df6096dcdb2c0ba09c93c6e

Eis, D. (2017). Introdução à Web Semântica.

Gonçalves, V. M. (2009). *A Web Semântica no contexto educativo*. Obtido de https://repositorio-aberto.up.pt/bitstream/10216/11104/2/Texto%20integral.pdf

W3C. (2004). *RDF Primer.* Obtido de https://www.w3.org/TR/rdf-primer/

W3C. (s.d.). *Standards, Semantic Web, Inference*. Obtido de https://www.w3.org/standards/semanticweb/inference

W3C. (s.d.). *Standards, Semantic Web, Ontologies*. Obtido de https://www.w3.org/standards/semanticweb/ontology

11 Manual de Utilizador

O Generatron, programa que gera dados para RDF, SQL e CSV com parâmetros customizados e erro induzido em dados conforme especificado. Tem a capacidade de produzir este tipo de outputs, como também a partir de inputs, criar estruturas de dados que são inferidas de um ficheiro à escolha do utilizador, ficheiro esse proveniente de uma base de dados de web semântica (RDF) com a sintaxe Turtle.

11.1 Ecrã Inicial

Abrindo o executável de java Generatron.jar, é apresentado um ecrã que permite 2 opções para inicialização do algoritmo.

Podemos assim:

1) Importar um ficheiro de uma BD já existente de RDF em formato Turtle.
2) Inicializar o programa sem dados, possibilitando ao utilizador criar uma estrutura de raiz.

Figura 1 – Ecrã de entrada do Generatron

Se o utilizador desejar carregar um ficheiro RDF, surge um popup para escolha do mesmo. Dado que o algoritmo está programado para trabalhar informações em formato Turtle, este só permite escolhas deste tipo de formato de ficheiro:

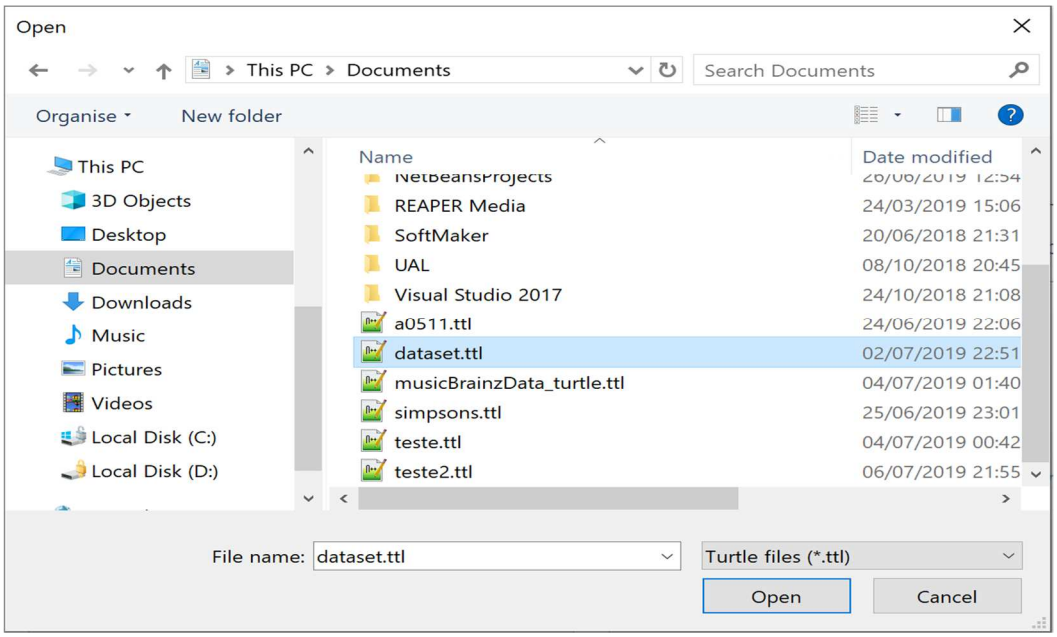

Figura 2 – Ecrã de escolha ficheiro de input Turtle

Depois de selecionar a opção "Open", o Generatron procede à sua importação, parsing, análise e conversão do formato textual de Turtle para o paradigma orientado a objetos. Este processo é necessário, para que mais tarde no ecrã principal, seja possível visualizar de forma clara e hierárquica a estrutura inferida, assim como possibilitar alterações, criações, remoções e configurações que o utilizador deseje realizar para posterior geração parametrizável de dados avulsa.

Caso o ficheiro seja maior do que 25 MB, surge um aviso com opção para "cortar" o número de linhas caso o utilizador assim deseje:

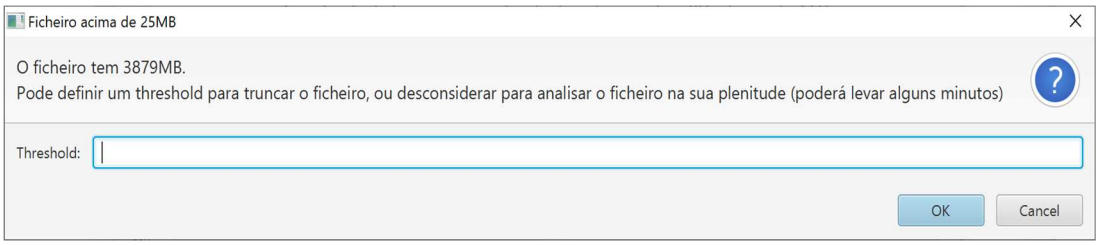

Figura 3 – Ecrã de aviso do tamanho do ficheiro de input

Enquanto o programa processa, é pedido ao utilizador para aguardar na parte inferior da janela:

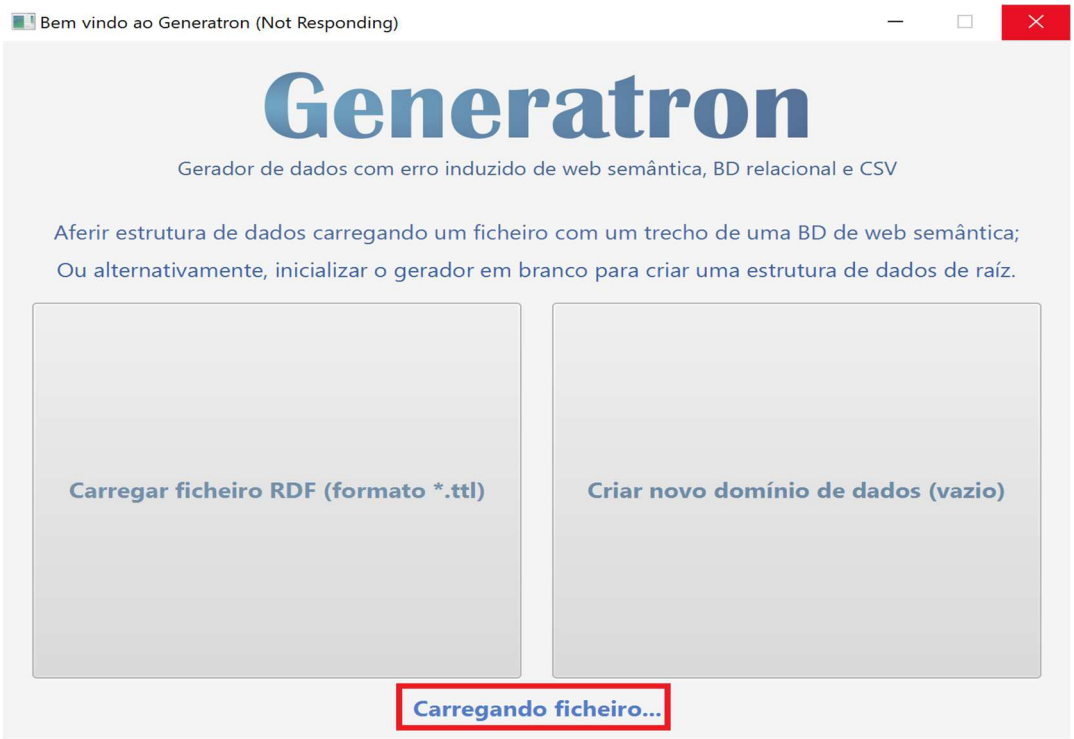

Figura 4 – Ecrã informativo

De seguida, são apresentadas as estatísticas de importação do ficheiro num popup informativo:

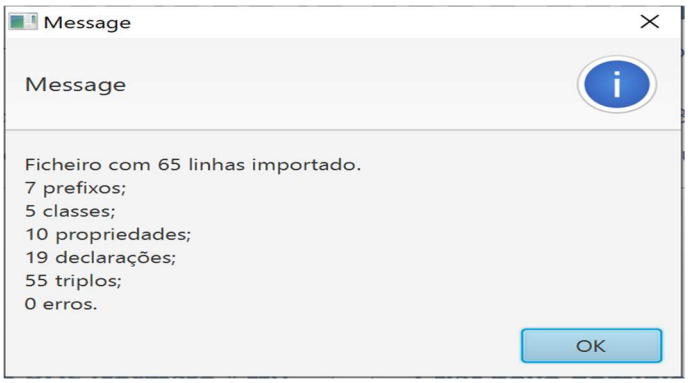

Figura 5 – Ecrã informativo das estatísticas de importação de ficheiro

No caso de existirem erros lógicos na importação, é criado um ficheiro de log na diretoria onde reside o executável do Generatron com o nome “LogErrosLogicosLeitura.txt”.

Possíveis erros lógicos que sejam obtidos no processo de conversão do ficheiro em formato Turtle para paradigma orientado por objetos, com o qual o algoritmo funciona, são escritos neste ficheiro, que contém a linha, fase do processo, e dados em que existiu a imparidade:

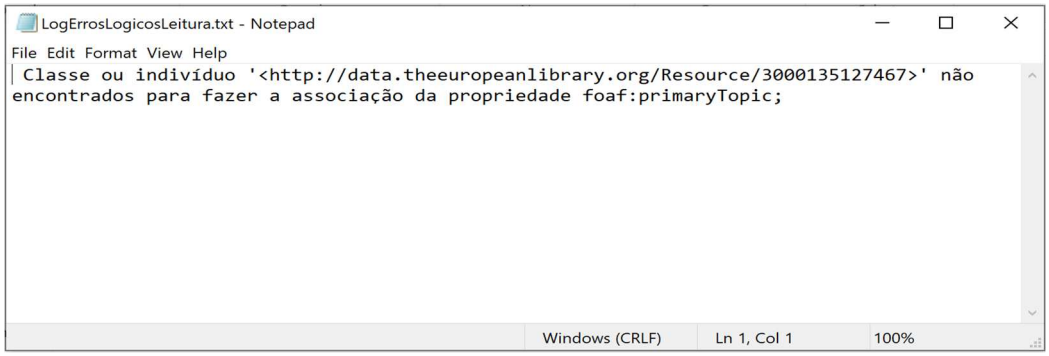

Figura 6 – Ecrã ilustrativo de um ficheiro de log

11.2 Ecrã de escolha de Output

Neste ecrã, o utilizador escolhe qual é o tipo de dados que deseja produzir como resultado final.

Figura 7 – Ecrã de escolha de output

11.3 Estruturação dos dados

A informação a seguir, mostra em detalhe a estrutura dos dados de output consoante a escolha do utilizador.

11.3.1 RDF

Ao escolher como output final a opção de RDF, o ecrã da estruturação dos dados permite configurar três tipos de objetos: Prefixos, Propriedades e Classes. Servem de estrutura base para dar suporte à organização dos dados que vamos querer gerar como output final do algoritmo.

Figura 8 – Ecrã de parametrização do output RDF

Apesar de estes 3 tipos de objetos serem adicionados e removidos de forma singular e independente, têm as seguintes interligações e condicionantes na sua criação:

- Um prefixo criado manualmente, tem sempre de ter uma nomenclatura e URI.
- Uma propriedade tem sempre de ter um prefixo, exceto se URI na importação.
- Uma classe tem sempre de ter prefixo, exceto quando for URI na importação.
- Uma propriedade tem sempre de ter um tipo de dados a ser gerado.
- Pode associar-se várias propriedades a uma classe para gerar os dados.

11.3.1.1 Adicionar Prefixos

Para adicionar um prefixo, é obrigatório escrever um nome e uma URI.

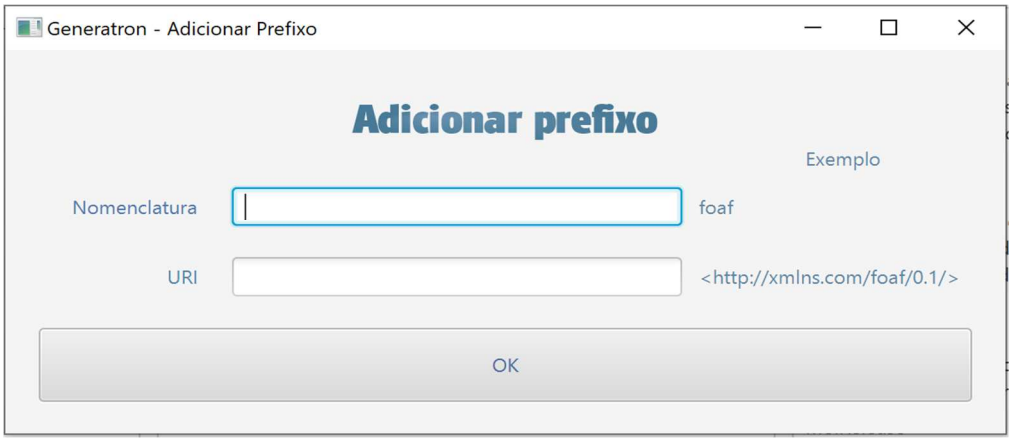

Figura 9 – Ecrã de definição de URI

Caso tal não aconteça, ou o input do utilizador não for válido, o Generatron avisa e rejeita.

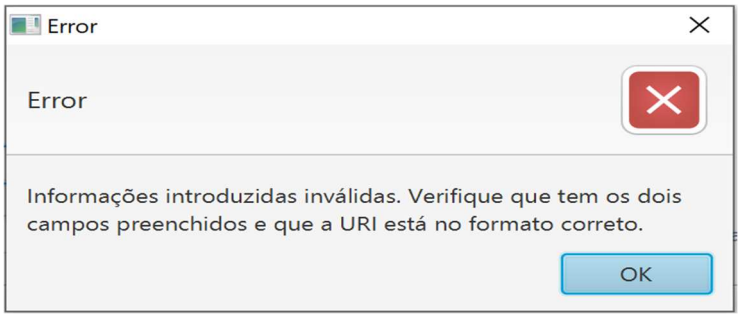

Figura 10 – Ecrã de informação de erro

11.3.1.2 Remover Prefixos

Para remover um prefixo, basta selecionar o Prefixo a ser apagado e carregar no botão com este nome. Este Botão só fica enabled quando existe um prefixo selecionado na lista:

Figura 11 – Ecrã de remoção de prefixos

11.3.1.3 Adicionar Propriedades

Para adicionar uma propriedade, escolhe-se um prefixo que já foi criado previamente (interligando-se assim os objetos), escolhe-se um nome, e escolhe-se que tipo de dados esta propriedade vai gerar, correspondente ao seu output normal gerado na fase final do algoritmo.

Os tipos de dados passíveis de ser gerados são:

- String - Um conjunto de caracteres alfanumérico aleatório.
- Número - Um conjunto de inteiros aleatórios.
- Nome - Um conjunto de 2 primeiros nomes e 2 apelidos aleatórios.
- Data - Uma data aleatória.
- NumeroTelefone - Um número de telefone aleatório com prefixos reais.
- WebLink - Um URL de site gerado aleatoriamente.
- Email - Um email gerado com nomes de pessoas e domínios reais.
- Cidade - Uma capital qualquer do mundo aleatória.
- País - Um país qualquer do mundo aleatório.

Após a seleção do tipo de dados, é mostrado um exemplo gerado aleatoriamente em tempo real à direita, de forma a que, o utilizador visualize um resultado realista e assim de forma imediata, entender o expectável do tipo de dados escolhido.

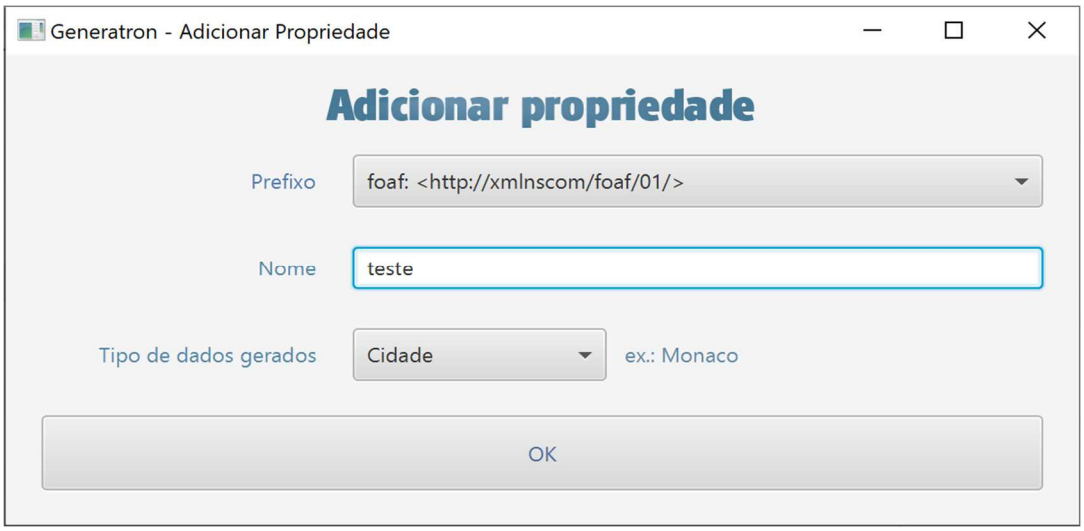

Figura 12 – Ecrã de adição de propriedades

É obrigatória a escolha de um prefixo, inscrever um nome com uma sintaxe válida e escolher um tipo de dados. Caso tal não aconteça, surge o popup de erro que rejeita por ser inválido:

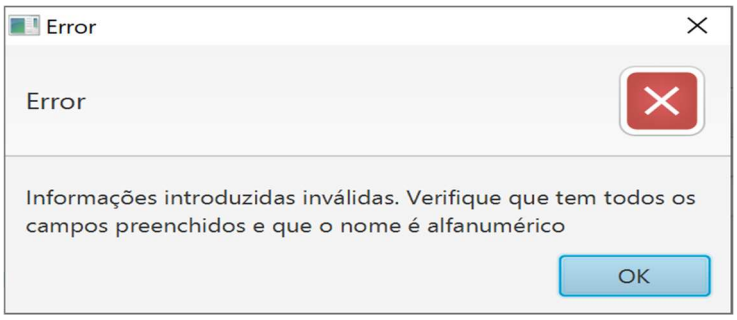

Figura 13 – Ecrã informativo de erro

11.3.1.4 Remover Propriedades

Para remover uma propriedade basta selecionar a que se deseja apagar, carregando no botão respetivo. Só fica enabled quando existe uma seleção na lista:

Figura 14 – Ecrã de remoção de propriedades

11.3.1.5 Adicionar Classes

Ao adicionar uma classe, é feita a escolha de um prefixo e de um nome.

Figura 15 – Ecrã informativo de erro

Caso o nome não seja alfanumérico ou não exista seleção no prefixo, temos um erro:

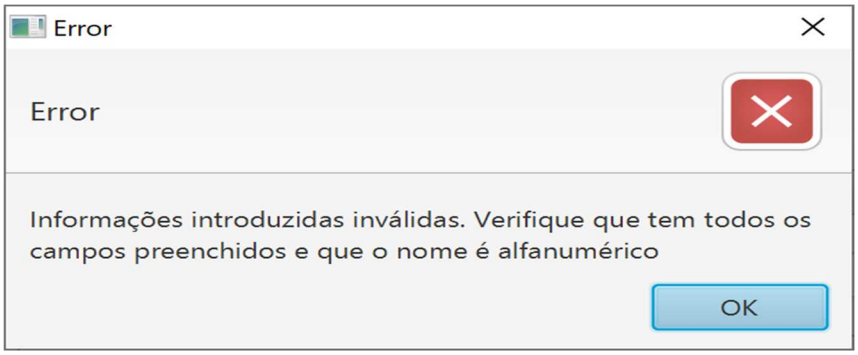

Figura 16 – Ecrã informativo de erro

11.3.1.6 Associar Propriedades

Quando temos uma classe selecionada, conseguimos escolher que propriedades queremos ter presentes na classe clicando neste botão. Através de uma janela com 2 listas, vemos quais são as propriedades que se encontram presentes na classe, e todas as que existem criadas no nosso domínio. Podemos associar as propriedades que queiramos, ou desassociar as que já façam parte:

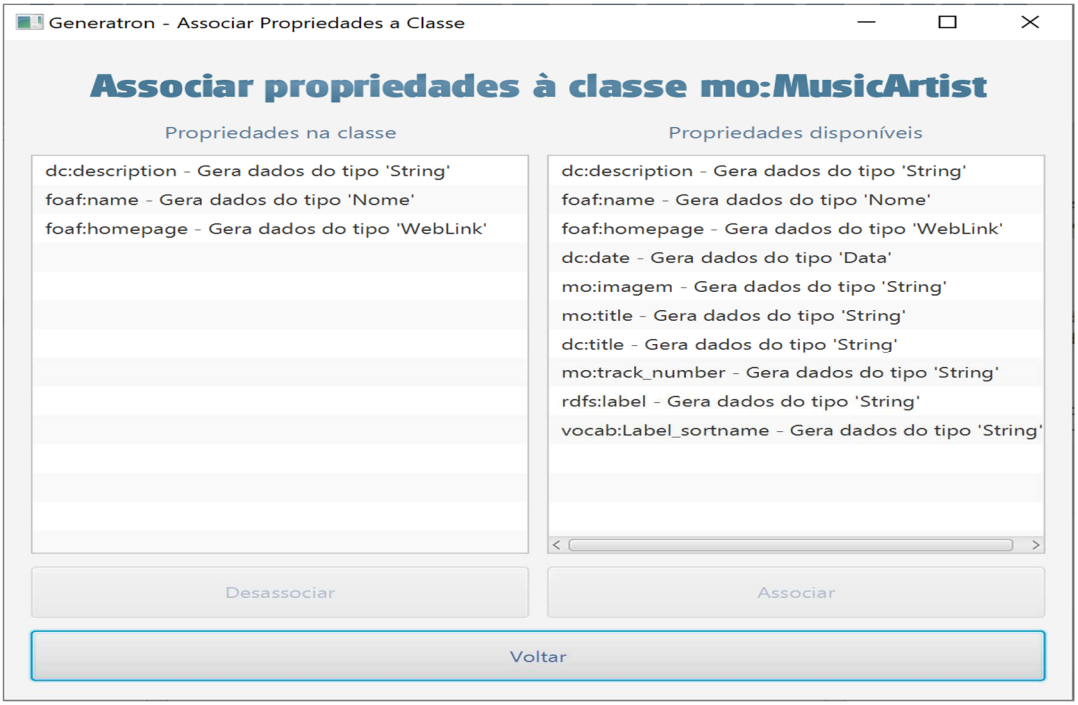

Figura 17 – Ecrã de associação de propriedades a classes

Existem restrições impostas com o fim de mitigar erro humano:

- O botão de Associar propriedades só está enabled com classe selecionada.
- O botão de Desassociar só está disponível quando há seleção na lista correspondente.
- O botão de Associar também só está enabled se existir seleção na sua lista.
- Só é possível associar propriedades que não existam já na classe.

11.3.1.7 Remover Classes

Para remover uma classe basta selecionar a que se deseja apagar, carregando no botão respetivo. Só fica enabled quando existe uma seleção na lista:

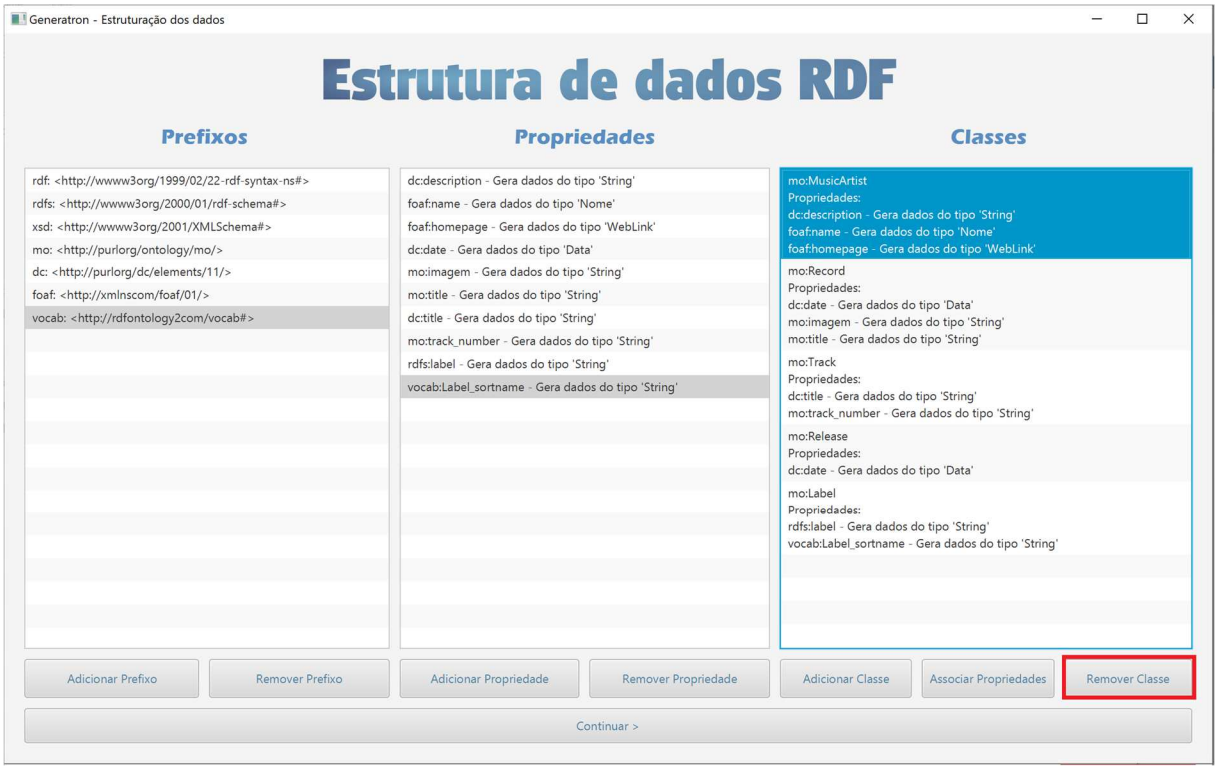

Figura 18 – Ecrã de remoção de classes

11.4 SQL e CSV

Na estruturação de dados do SQL e CSV, o layout e opções diferem ligeiramente. No caso do SQL, não existem classes, mas sim tabelas. No caso de CSV, ficheiros.

Por outro lado, em vez de propriedades temos colunas onde os dados serão gerados.

Finalmente, não faz sentido existirem prefixos, pois de uma forma genérica não é comum a sua utilidade nas bases de dados relacionais, ou nos ficheiros de Comma Separated Values.

Como tal, extingue-se a secção de prefixos assim como a sua seleção nas várias janelas de criação de tabelas e colunas. Em substituição das classes e propriedades, aparecem as tabelas e colunas, respetivamente. As funcionalidades estruturais mantêm-se idênticas ao RDF:

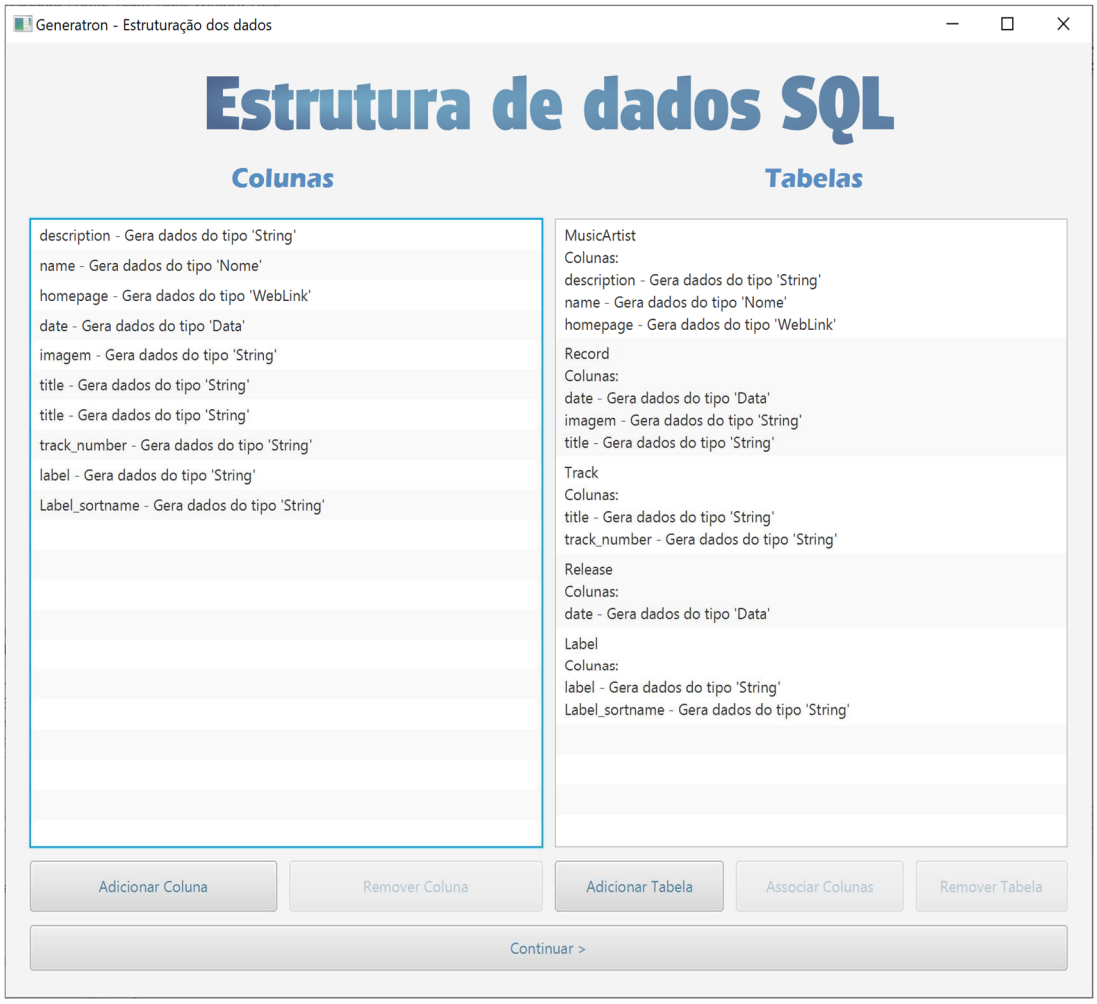

Figura 19 – Ecrã de definição da estrutura de dados SQL

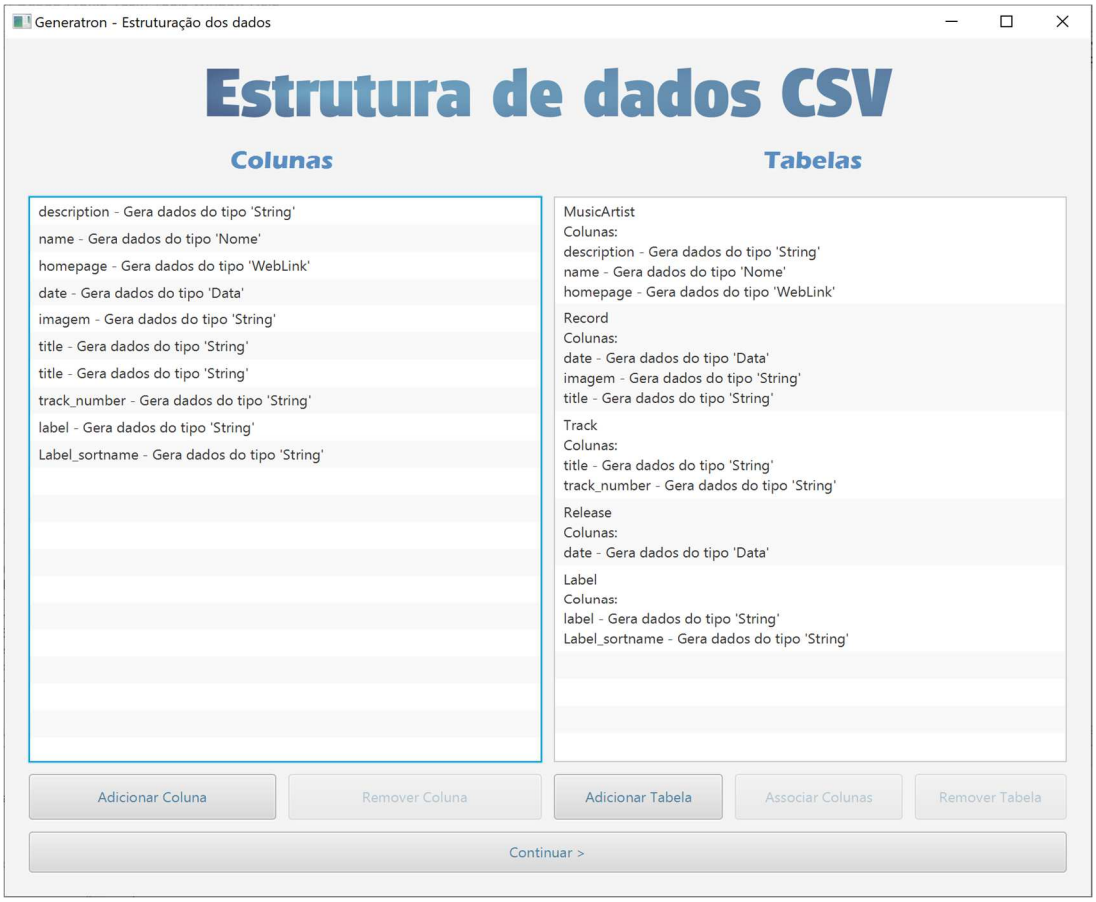

Figura 20 – Ecrã de definição da estrutura de dados CSV

11.5 Parametrização do Gerador

Após a definição minuciosa da estrutura dos dados, segue-se a parametrização do gerador de dados. A janela final seguinte, possibilita a indicação do número de registos a gerar de cada uma das classes ou tabelas, e opcionalmente, qual a percentagem de erro induzido a ocorrer em cada uma das propriedades ou colunas, incluindo o tipo de erro:

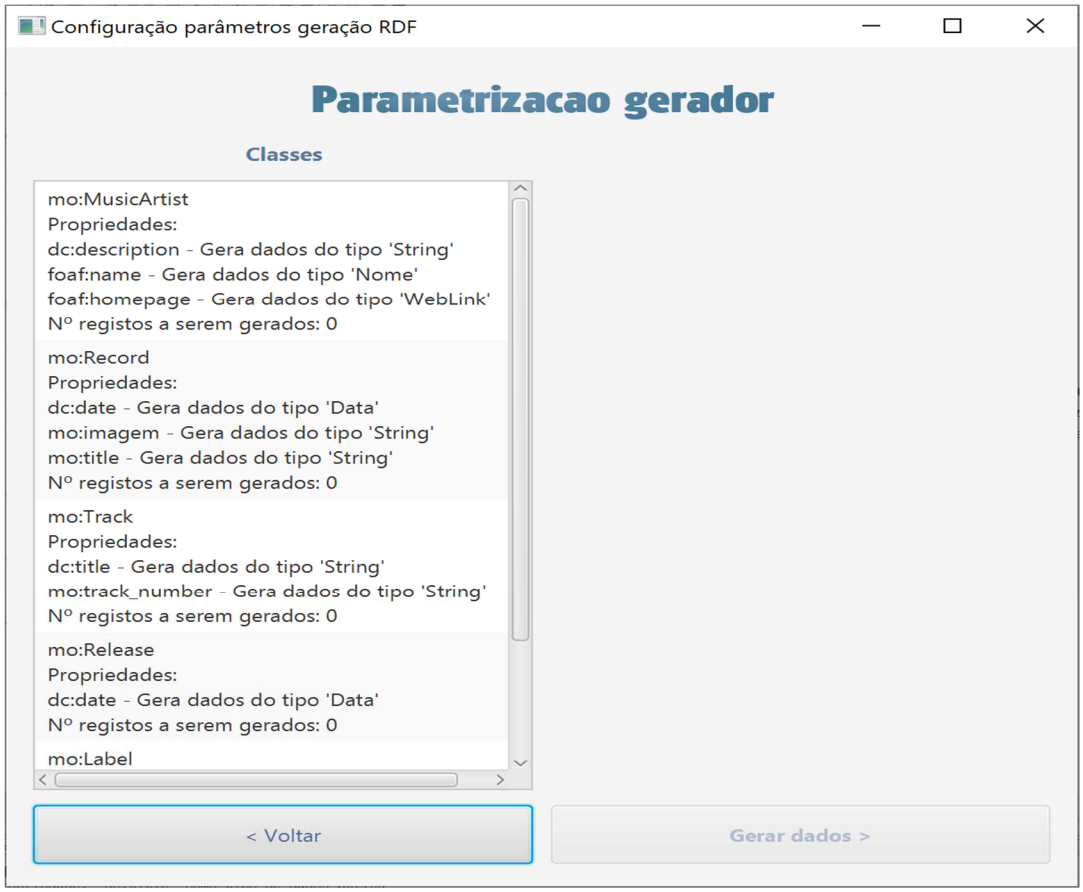

Figura 21 – Ecrã de parametrização do gerador de dados

11.5.1 Número de Registos a Gerar

Após a seleção de uma classe / tabela, existe uma opção no lado direito da janela, que permite escolher quantos registos vão ser gerados daquele bloco de estrutura previamente definida. Após essa indicação, o botão Guardar nº de registos deve ser acionado.

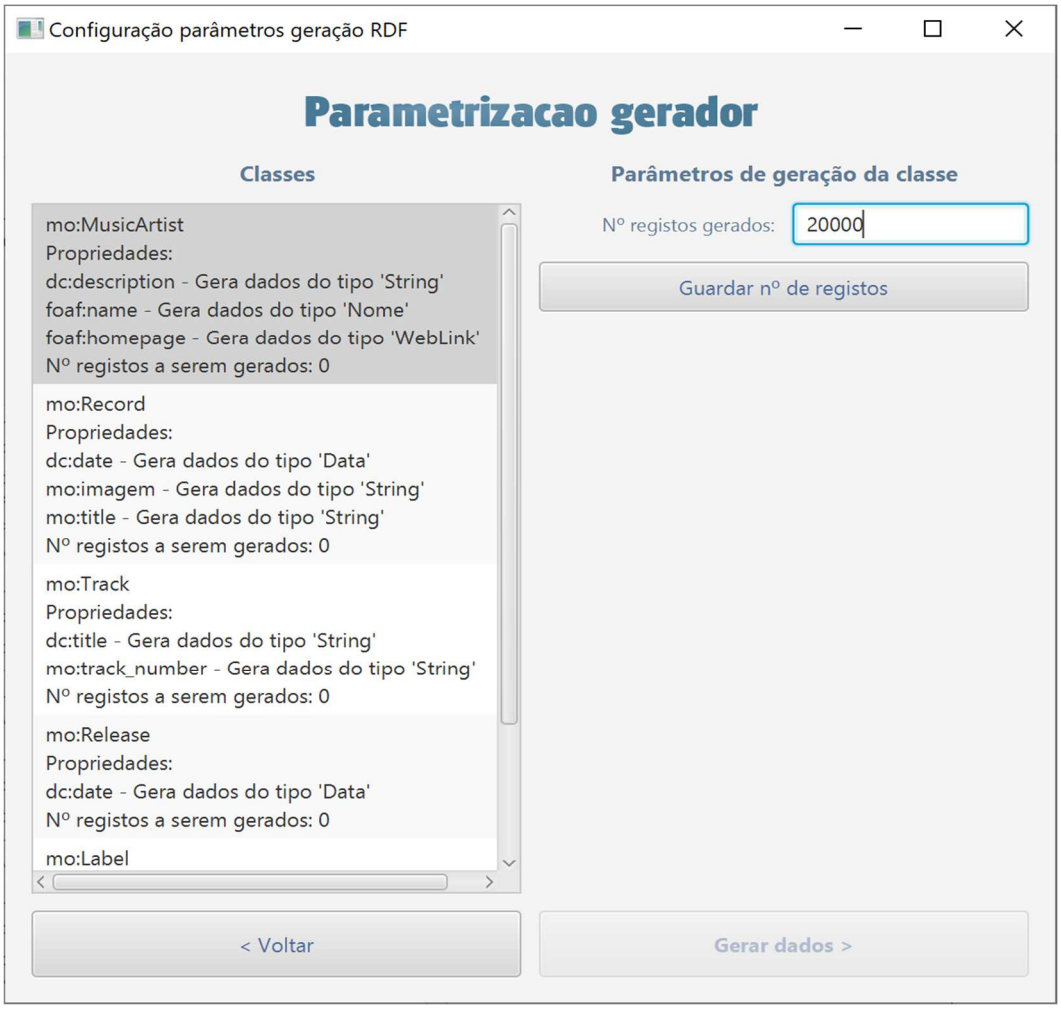

Figura 22 – Ecrã de parametrização do gerador de dados, escolha do número de registos

Caso o utilizador se engane e ponha caracteres que não números, é devolvido um erro:

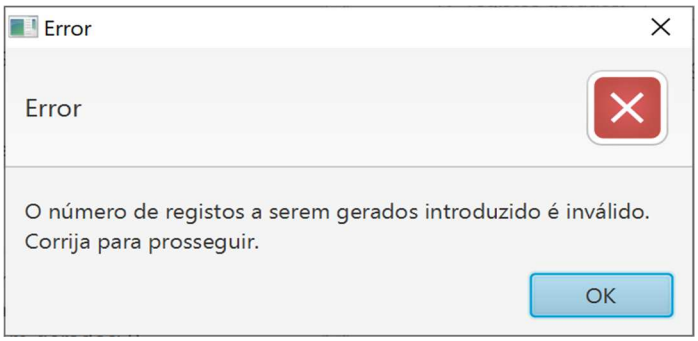

Figura 23 – Ecrã informativo de erros.

11.5.2 Indução de erros

Após registrar o número de registos a gerar numa determinada classe ou tabela, surgem novas opções na secção direita inferior:

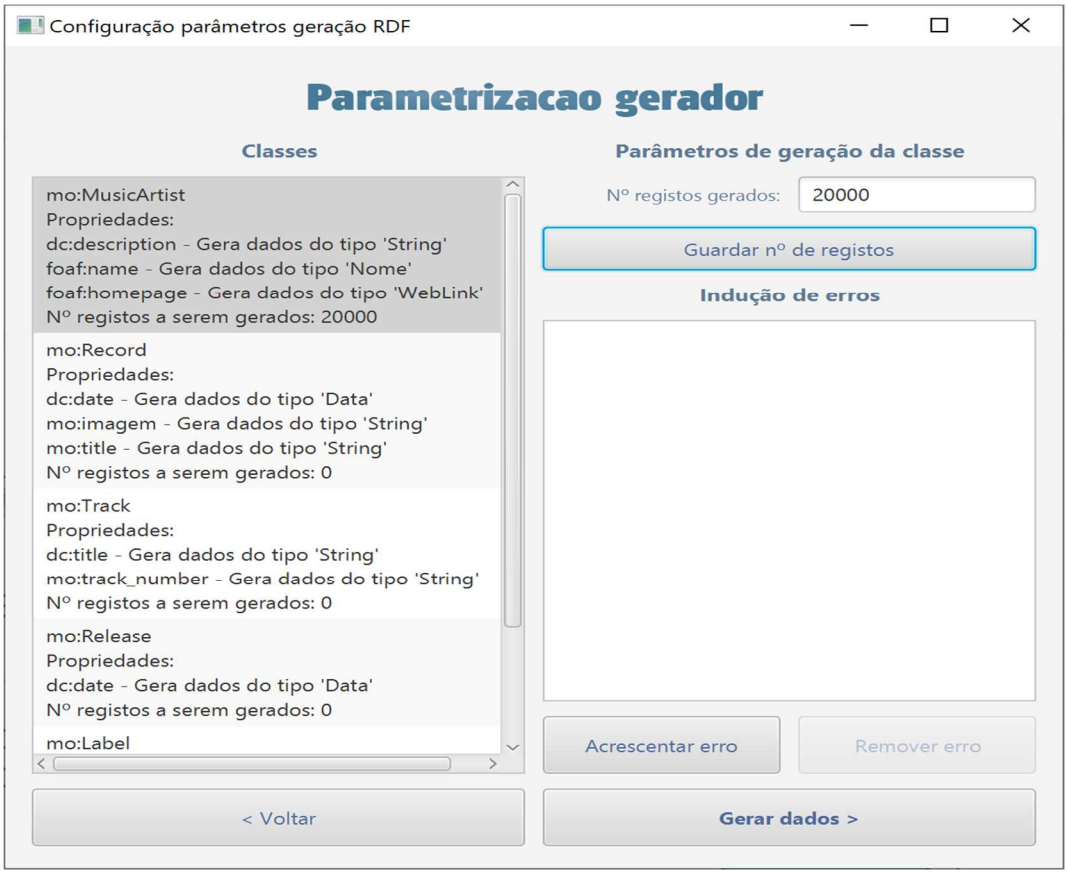

Figura 24 – Ecrã de opções da indução de erros

11.5.3 Acrescentar Erro

Para acrescentar um erro, clica-se em Acrescentar erro, de onde surge a janela seguinte:

Figura 25 – Ecrã de entrada do erro

Nesta janela, o utilizador deve escolher: qual a propriedade ou coluna em que vai induzir o erro, qual a percentagem de erro (do lado direito, é indicado o número estimado de registos que serão afetados), e finalmente que tipo de erro quer induzir.

Os tipos de erro existentes são:

- SemDados -> É posta uma String vazia em vez do dado original.
- TrocaDeTipo -> O tipo de dados é trocado (ex.: Nome passa a ser Data).
- CorteDePropriedade -> No caso RDF, a propriedade deixa de existir.

Caso exista informação em falta ou percentagem de erro que não seja entre 0 e 100, ou caracteres que não sejam números, é devolvida uma mensagem de erro:

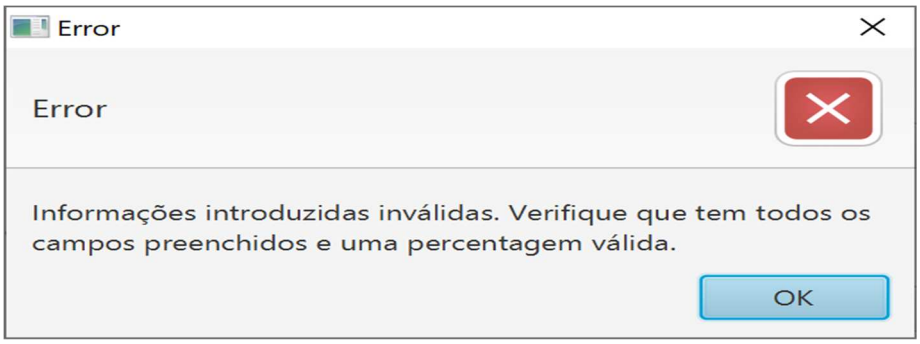

Figura 26 – Ecrã informativo de erros

Ao clicar OK na janela de Indução de erro, visualiza-se o que foi criado:

Figura 27 – Ecrã ilustrativo dos erros parametrizados

11.5.4 Remover Erro

O utilizador pode remover um erro que tenha criado, selecionando na lista o erro para enable.

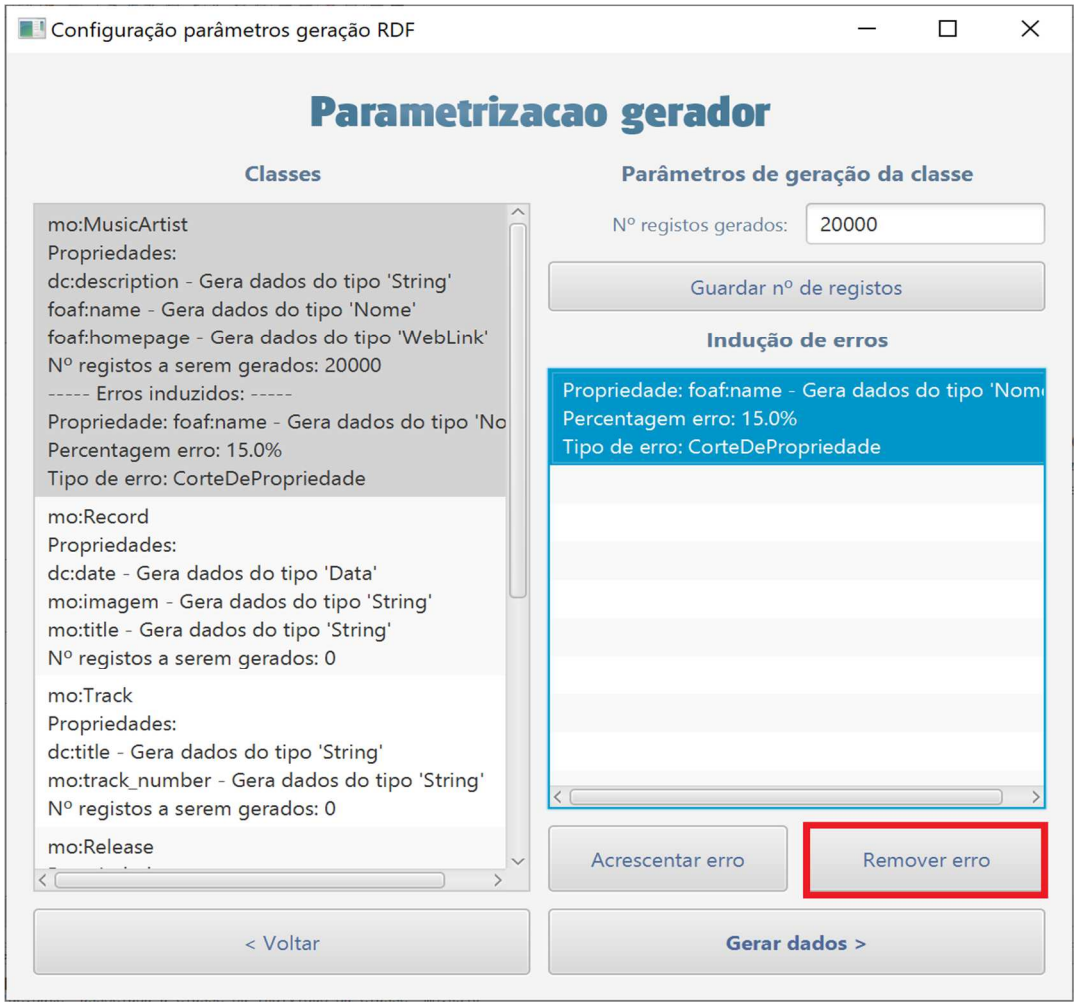

Figura 28 – Ecrã ilustrativo da remoção de erro

11.5.5 Gerar Dados

Após a indicação de: dados a gerar em cada classe / tabela, indicação de erros a induzir em cada uma das propriedades / colunas, aparece a opção de escolha de localização de guarda do ficheiro de output.

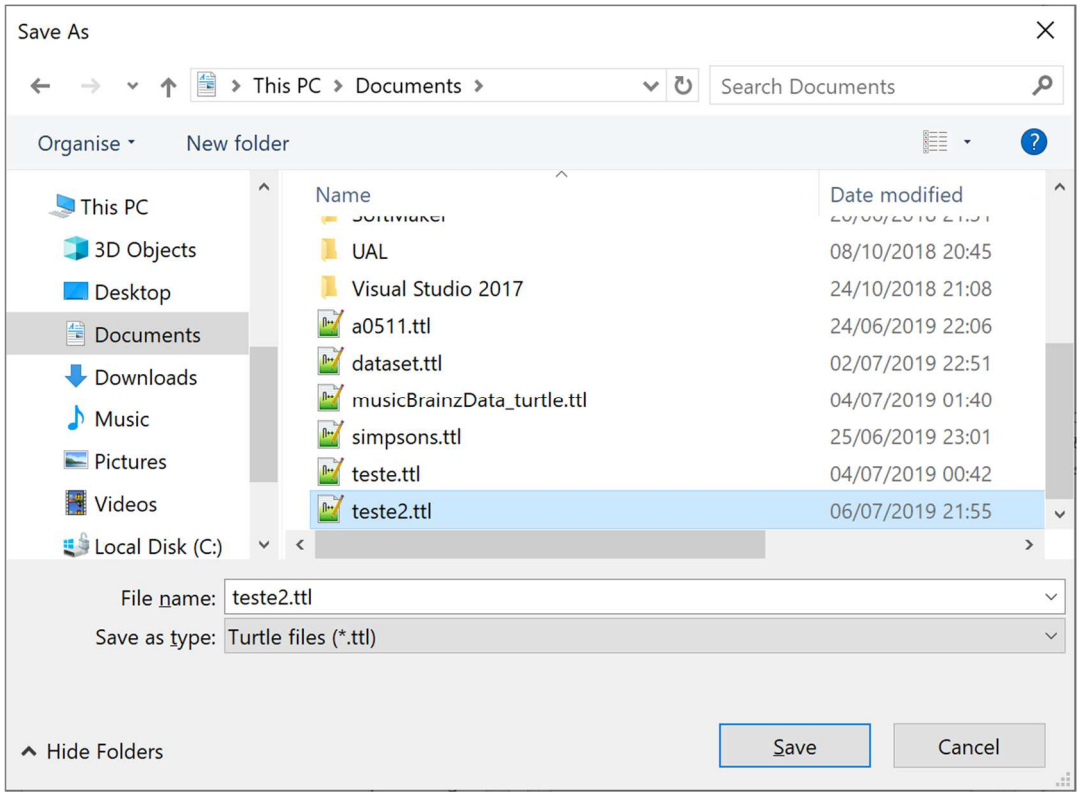

Figura 29 – Ecrã ilustrativo de localização para o ficheiro de output

No caso de RDF, será um ficheiro Turtle (*.ttl). Em SQL, um script sql (*.sql).

Nos CSVs, como cada tabela é na verdade um ficheiro de texto separado por “;”, geram-se tantos ficheiros quantas tabelas existam com os dados correspondentes na pasta do programa.

Após seleção, enquanto os dados são gerados, surge um popup informativo, que pede para o utilizador aguardar enquanto o programa processa os pedidos.

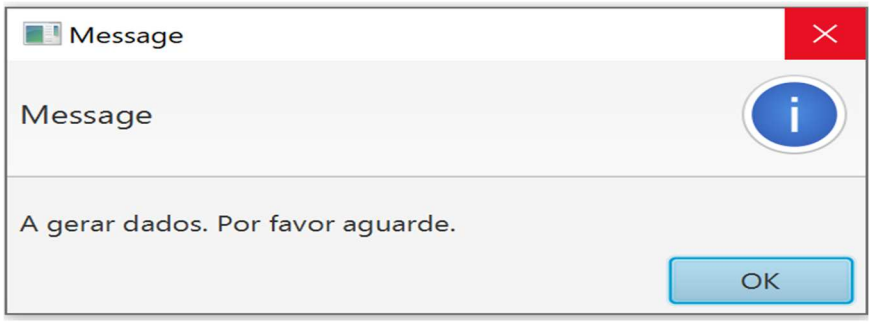

Figura 30 – Ecrã informativo de espera

Caso existam erros no processamento, surge um popup de erro.

Quando o processo acaba com sucesso, surge a seguinte mensagem:

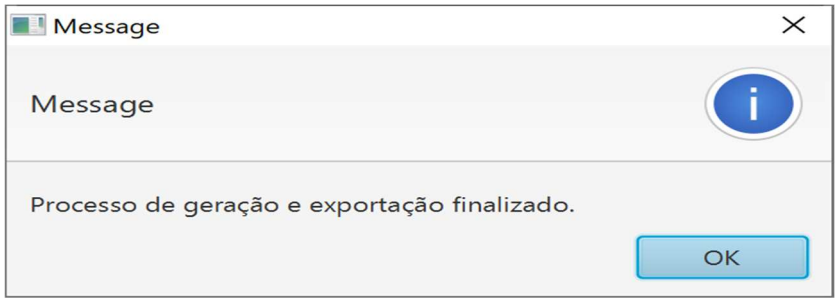

Figura 31 – Ecrã informativo de erros

11.6 Dados Exportados

11.6.1 Log de Registos com Erro Induzido

Casos o utilizador tenha escolhido realizar induções de erro, é importante perceber de forma clara em que registos existiram esses impactos. Para tal, o Generatron guarda todos os registos em que houve esse tipo ocorrências num ficheiro de texto específico, nomeado "LogDadosSujos.txt", na pasta em que o seu executável se encontre.

Exemplo de conteúdo do ficheiro:

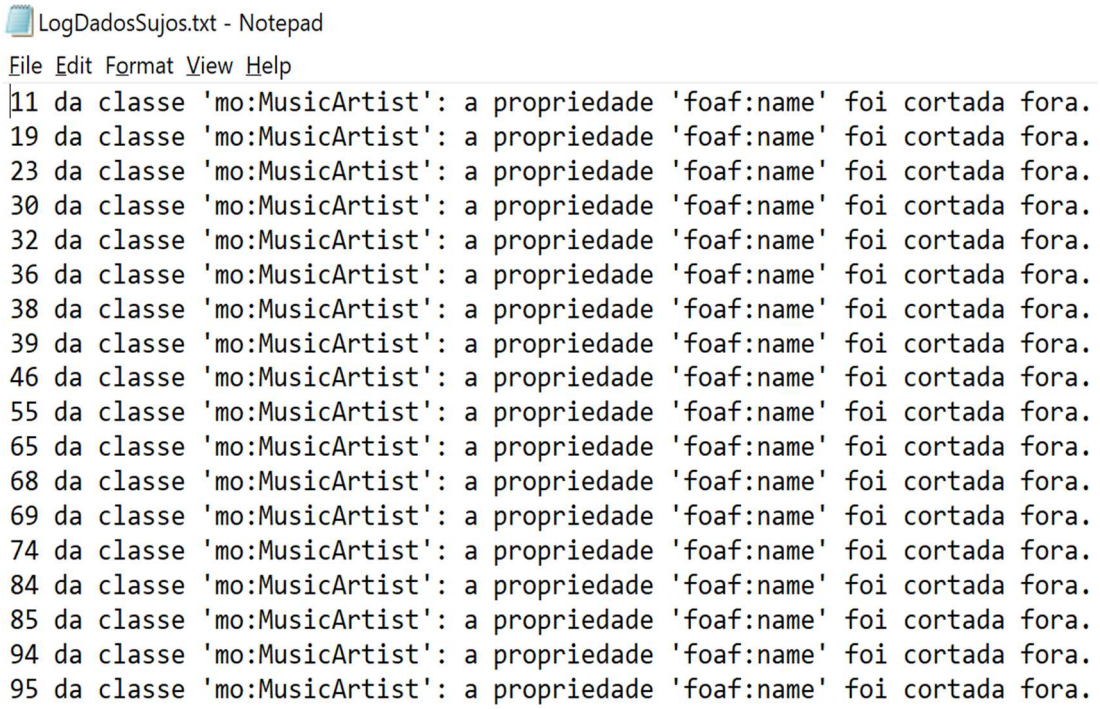

Figura 32 – Ecrã de log de erros induzidos

11.6.2 Exemplo de output RDF

```
@prefix xsd: <http://wwww3org/2001/XMLSchema#> .
@prefix mo: <http://purlorg/ontology/mo/> .
@prefix dc: <http://purlorg/dc/elements/11/> .
@prefix foaf: <http://xmlnscom/foaf/01/> .
@prefix vocab: <http://rdfontology2com/vocab#> .

Individuo1 a mo:MusicArtist ;
    dc:description "pjM1a98CCJOv" ;
    foaf:name "Joséfa Mercedes Reis Furtado" ;
    foaf:homepage "http://www.2uXTB0OCfcRT.com/" .

Individuo2 a mo:MusicArtist ;
    dc:description "nc5t4xqtzNP1" ;
    foaf:name "Sandro do Paraíso Maria de São José Salgueiro Caetano" ;
    foaf:homepage "http://www.aOTqCM9yk1dq.com/" .

Individuo3 a mo:MusicArtist ;
    dc:description "9imZxU9jWxE2" ;
    foaf:name "Aldo Dino Infante Cardoso Mourato" ;
    foaf:homepage "http://www.Vd5ufIvQ9uOX.com/" .

Individuo4 a mo:MusicArtist ;
    dc:description "j6PcZDB6yRse" ;
    foaf:name "Mélanie Porciana Campos Caetano" ;
    foaf:homepage "http://www.R78rV7D6gMbU.com/" .

Individuo5 a mo:MusicArtist ;
    dc:description "O4rfNe1gPxx6" ;
    foaf:name "Romeia Stela Patrício Piedade" ;
    foaf:homepage "http://www.ho7X0jtoD2va.com/" .

Individuo6 a mo:MusicArtist ;
    dc:description "KEyrPWOaP7PX" ;
    foaf:name "Denisa Ginestal Santos Branco" ;
    foaf:homepage "http://www.taVel9Yp9ARp.com/" .

Individuo7 a mo:MusicArtist ;
    dc:description "3Q51DK3fRFfC" ;
    foaf:name "Chantal Eloisa Sampaio Brandão" ;
    foaf:homepage "http://www.yBqZhB09OxQe.com/" .

Individuo8 a mo:MusicArtist ;
    dc:description "7gCQv7ADBA1F" ;
    foaf:name "Zubaida Lúcio do Rosário Morais Roque" ;
    foaf:homepage "http://www.DfjdSGM4krEQ.com/" .
```

Figura 33 – Ecrã ilustrativo de ficheiro de output em RDF

11.6.3 Exemplo de output SQL

```
CREATE TABLE MusicArtist(
        id INT NOT NULL,
        description VARCHAR(255),
        name VARCHAR(255),
        homepage VARCHAR(255),
        CONSTRAINT PK_MusicArtist PRIMARY KEY (id)
);

INSERT INTO MusicArtist VALUES(1,'89PgqgRKyxD5','Deótila Livramento Rosa Mota','http://www.JuM9vo1Ct9WD.com/');

INSERT INTO MusicArtist VALUES(2,'8dJPEUVcdLLL','Cristiano Delmar Geraldes Neves','http://www.IoecrES7s9z3.com/');

INSERT INTO MusicArtist VALUES(3,'CChqGTZfestK','Décia Sância Cardoso Neves','http://www.lH6XPEHBNHb6.com/');

INSERT INTO MusicArtist VALUES(4,'bLE5n7pH1lx4','Suria Bárbora Brito Bacelar','http://www.OOSAA4hIhCfm.com/');

INSERT INTO MusicArtist VALUES(5,'OKK7SkPII31S','Nilson Samaritano Silva Sousa','http://www.UWvmyWxdpriV.com/');

INSERT INTO MusicArtist VALUES(6,'cU6LIxEPVLuD','Dener Olindo Rola Simões','http://www.CMma0rYx9eOu.com/');

INSERT INTO MusicArtist VALUES(7,'qcOdDy8rn136','Gonzaga Sisínio Coelho Brito','http://www.90n537Eer5sd.com/');

INSERT INTO MusicArtist VALUES(8,'s12Qfp9TQvx7','Jorja Donzília Correia Vidigueira','http://www.BiQdZglP07KU.com/');

INSERT INTO MusicArtist VALUES(9,'Lc1fdjzdpRk8','Niceia Pia Jesus Alves','http://www.1QUjr6p0f53b.com/');

INSERT INTO MusicArtist VALUES(10,'ianKdJSx6NDt','Alexa Giovana Sobral Oliveira','http://www.UiFm41RhN0Gk.com/');

INSERT INTO MusicArtist VALUES(11,'pJaEbcoHX84z','Santa Maria Ticiana Espinho Neves','http://www.kTCNDMe6MEgf.com/');

INSERT INTO MusicArtist VALUES(12,'xOUWE7nhxpYk','Josselina Mirandolina Rodrigues Cardoso','http://www.V5yZbMzkMsIc.com/');

INSERT INTO MusicArtist VALUES(13,'H6TxGUAyfkTc','Fafe Romeu Vieira Brito','http://www.IWtJDyiVfBBc.com/');

INSERT INTO MusicArtist VALUES(14,'fBY5h5er3Y0F','Nessa Radija Gomes Garrido','http://www.Ba12ybdeNOuU.com/');

INSERT INTO MusicArtist VALUES(15,'eg0d3FHleQCH','Clívia Jaquelina Ferreira Chagas','http://www.Kma2virmsHRU.com/');

INSERT INTO MusicArtist VALUES(16,'tAFrkWRAyXzA','América Marília Pereira Sousa','http://www.98MPYIk69fkg.com/');

INSERT INTO MusicArtist VALUES(17,'7pXbkWeaYuMF','Gueir Marília Coelho Brandão','http://www.tVP7v7HJzVJ0.com/');

INSERT INTO MusicArtist VALUES(18,'jD5HTJxom1Ui','Adamantino Leanor Caldeira Reis','http://www.xPmr8hGItCRM.com/');

INSERT INTO MusicArtist VALUES(19,'ht8TThH2y2RV','Tude Gui Jesus Soares','http://www.hKY6LqaMo9Z0.com/');

INSERT INTO MusicArtist VALUES(20,'Un0bm73iUgH3','Antonieta Etelca Branco Sousa','http://www.5xF4c9MidmS6.com/');
```

Figura 34 – Ecrã ilustrativo de ficheiro de output em SQL

11.6.4 Exemplo de output CSV

	A	B	C
1	description	name	homepage
2	YNvmz5GQDmPS	Galileu Ilundi Real Sereno	http://www.8LS5drRzMkzm.com/
3	shfd4uegf3EL	Selene Fara Guterres Lopes	http://www.WYsejtUUFFhd.com/
4	DizlNi9OBCl8	Alano Darci Veiga Guerreiro	http://www.0WCC3aByY7ON.com/
5	j5lJW9O7aP0l	Guido Leonor de Jesus Rato Dias	http://www.yhLet9pj6n9M.com/
6	n6X9Zq6rI4Tm	Alvarim Odeberto Roque Gralha	http://www.x5lpi9NgOiSP.com/
7	iogGeqoSbscC	Jabes Eliezer Coelho Saraiva	http://www.jaaIWdqkEF7N.com/
8	9SjiqBcvo4qR	Soledade Anael Roque Matos	http://www.krJWXVzpjrsU.com/
9	yaOQ3gczJ5xW	Zobaida Bernardim Caetano Rola	http://www.Qexuap8p1CkF.com/
10	TMhQGkiUchZ9	Boavida Viviane Sousa Pinheiro	http://www.JbhSUgl01lhx.com/
11	h7itTlsLPxdo	Delfino Miraldina Matos Coelho	http://www.7VtCnV93hQcS.com/
12	VSQiZy0jJa8A	Lua Gonzaga Gomes Costa	http://www.8ri5KBmQsVL6.com/
13	br0p8Zt41777	Dircea Calila Pinheiro Cardoso	http://www.99MQS15ss31D.com/
14	Wna9bon9FTIn	Murilo Sesira Carvalho Rodrigues	http://www.J0KayaDx0Yxz.com/
15	H7FouqRmXPrm	Joel Magna Geraldes Teixeira	http://www.5ldTersYV4Uj.com/
16	AmePv50rFXAR	Celisa Rolim Domingos Cardoso	http://www.sGmpLvvicrOF.com/
17	DUkOJpDiDQ8Y	Odeberto Sarah Borges Vieira	http://www.8RcQE8vZuCZU.com/
18	xoC1zD9FJkbm	Dulcelina Martins Furtado Dias	http://www.O8Kjeor5KMZG.com/
19	7zsCiRpnq73f	Aldo Dino Marco Almeida Rocha	http://www.m1j9WdKG118B.com/
20	bq2AJTGPTmD8	Darque Vianei Tavares Geraldes	http://www.keuP5agKClJy.com/
21	bxZLzhmSiIEn	Nilson Ãtila Sobral Dias	http://www.cfOWrAL0iGWI.com/
22	Opa3HJys4ptd	Vasco Ianesis Furtado Teixeira	http://www.hFZcrtvEXXYn.com/
23	K8igEEERmHhP	Ornela HerÃ©dia Rato Garrido	http://www.e4vjynAQXa6R.com/
24	ON8gYrQtzJlx	Jezabel Marcus Pimenta Nunes	http://www.3WGAtZTsvVEy.com/
25	R0tPnXViSlMC	Sandra Juvenal Ferreira Piedade	http://www.GWG59IlLEH38.com/
26	TNWY7xbnOq4t	Mauro Pia Barroso Saraiva	http://www.lt4PkJ2o4VEu.com/
27	N4lz1f3zRP6G	Enide Hulda Mota Pimenta	http://www.PQnd2vQY7rrx.com/
28	G9GHj9Bb8HuB	Dorinda Nara Vidigueira Branco	http://www.8SEghPV2hfsi.com/
29	qPWHIlzcXb40	Soraia Eleine Portas Real	http://www.lKYa7cksNHh3.com/

Figura 35 – Ecrã ilustrativo de ficheiro de output em CSV