INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

Modelo integrado de dados abertos para Smart Cities

Othmane El Arbaoui

Mestrado

Projecto Final para obtenção do Grau de Mestre
em Engenharia Informática e de Computadores

Orientadores : Prof. Doutor Nuno Miguel Soares Datia
Prof. Doutor José Manuel de Campos Lages Garcia Simão

Júri:

Presidente: Prof. Doutor Nuno Miguel Machado Cruz

Vogais: Prof. Doutor Paulo Manuel Trigo Cândido da Silva
Prof. Doutor Nuno Miguel Soares Datia

Setembro, 2023

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

Modelo integrado de dados abertos para Smart Cities

Othmane El Arbaoui

Mestrado

Projecto Final para obtenção do Grau de Mestre
em Engenharia Informática e de Computadores

Orientadores : Prof. Doutor Nuno Miguel Soares Datia
Prof. Doutor José Manuel de Campos Lages Garcia Simão

Júri:

Presidente: Prof. Doutor Nuno Miguel Machado Cruz

Vogais: Prof. Doutor Paulo Manuel Trigo Cândido da Silva
Prof. Doutor Nuno Miguel Soares Datia

Setembro, 2023

Dedico este TFM ao meu querido avô, Mohamed El Maataoui, que infelizmente nos deixou antes de ver a conclusão desta obra. Sua sabedoria, amor e apoio ao longo da minha vida serão sempre lembrados com carinho. Este trabalho é uma homenagem ao seu espírito inspirador e à sua influência duradoura no meu percurso académico. Que Allah lhe conceda o mais alto dos lugares no Paraíso e que sua alma descanse em paz, vovô.

Agradecimentos

Gostaria de expressar a minha profunda gratidão a todos os que tornaram este trabalho possível. Em primeiro lugar, agradeço a Allah, o Misericordioso, por me guiar e me dar forças para concluir este TFM.

Aos meus orientadores, Prof. Doutor Nuno Miguel Soares Datia e Prof. Doutor José Manuel de Campos Lages Garcia Simão, gostaria de expressar a minha mais sincera gratidão. Eles mostraram infinita paciência e orientação sábia ao longo deste caminho, o que me faz sentir ainda mais grato por tê-los como guias neste processo. Seus conselhos, orientações e revisões detalhadas foram fundamentais para a excelência deste trabalho, e sua dedicação ao meu crescimento académico é algo que nunca esquecerei.

À minha família, expresso o meu sincero agradecimento pelo seu apoio inabalável e amor incondicional. Suas palavras de encorajamento e compreensão em tempos desafiadores foram a força motriz por trás de mim.

Aos meus amigos e colegas, agradeço a colaboração, as discussões enriquecedoras e o apoio mútuo. As suas contribuições foram inestimáveis para o desenvolvimento das ideias apresentadas neste trabalho.

Por fim, dedico um agradecimento especial à memória do meu avô, cujo legado de sabedoria e inspiração permanecerá sempre comigo.

Que Allah abençoe a todos vocês e lhes conceda saúde e felicidade em suas vidas.

Com profunda gratidão,

Othmane EL Arbaoui

Resumo

Existem cada vez mais dados disponíveis para consulta, fornecidos tanto por entidades públicas quanto por entidades privadas, alguns dos quais são considerados dados abertos. O seu acesso pode variar, pela consulta uma simples página Web, pela consulta de uma API ou através de uma aplicação mais complexa de acesso e análise de dados. Embora o número e diversidade de dados seja maior, estes continuam pouco ricos em metainformação, sem recurso a standards nem a termos com semântica bem conhecida. Como resultado, os conjuntos de dados permanecem frequentemente "isolados", sem uma ligação clara entre eles, sem um mecanismo de integração de dados, tornando difícil para o utilizador identificar e inter-relacionar fontes de dados relevantes.

Neste trabalho final de mestrado, com base nos dados fornecidos pelo portal de dados abertos da Câmara Municipal de Lisboa (CML), pretendemos analisar as lacunas na metainformação e apresentar uma proposta de integração de dados em conformidade com a especificação DCAT-AP do W3C, usando o vocabulário do Catálogo de Dados (DCAT) e recorrendo a portais de dados abertos europeus.

O processo de integração recorre a modelos de dados baseados nos princípios da Web Semântica, associada a soluções de dados conectados e suas respetivas tecnologias. Esse processo tem como objetivo estabelecer conexões entre conjuntos de dados abertos orientados para cidades inteligentes, que são modelados como dados interligados e disponibilizados na plataforma de dados abertos da CML.

O resultado deste trabalho possibilita que o processo de gestão dos dados seja realizado de forma a torná-los mais compreensíveis, com melhor qualidade e maior potencial de integração. Isso, por sua vez, promove uma maior transparência no acesso à informação e, adicionalmente, abre portas para o desenvolvimento de novos modelos de negócio e serviços que se baseiam nos dados.

Palavras-chave: Web Semântica, DCAT-AP, Integração de Dados, Metadados, Dados Abertos

Abstract

The volume of available data for consultation is rapidly increasing, provided by both public and private entities, with some categorized as open data. Access to this data can range from simple web page queries to the use of APIs or more complex applications for data access and analysis. Despite the growing diversity and quantity of data, they often lack comprehensive metadata, standards, or well-established semantic terms.Consequently, datasets frequently exist in isolation, with no clear interconnection or data integration mechanism, making it challenging for users to identify and correlate relevant data sources.

In this master's thesis, we focus on addressing the existing gaps in metadata. Leveraging data from the open data portal of the Lisbon City Council (CML), our objective is to propose a data integration solution aligned with the W3C's DCAT-AP specification, utilizing the Data Catalog (DCAT) vocabulary and drawing insights from European open data portals.

The integration process relies on data models rooted in the principles of the Semantic Web, complemented by linked data solutions and associated technologies. The ultimate aim of this process is to establish connections among open datasets tailored for smart cities, representing them as interconnected data available on the CML's open data platform.

The outcomes of this work facilitate data management, enhancing comprehensibility, quality, and integration potential. This, in turn, fosters greater transparency in information access and opens avenues for the development of new business models and services centered around data.

Keywords: Semantic Web, DCAT-AP, Data Integration, Metadata, Open Data

Índice

Lista de Figuras

Lista de Tabelas

Lista de Listagens

Lista de Abreviaturas e Siglas

API Application Programming Interfac. 16

CKAN Comprehensive Knowledge Archive Network. 8
CML Câmara Municipal de Lisboa. ix, xi, 4

DCAT-AP *Application profile for data portals in Europe*. 4

EU European Union. 12

FOAF Friend of a Friend. 13

GLD Government Related Data. 13

HTTP HyperText Transfer Protocol. 10

ISEL Instituto Superior de Engenharia de Lisboa. 3

JSON JavaScript Object Notation. xvii, 21

LOD Linked Open Data. 10

ONG Non-Governmental Organisation. 13

PDF Portable Document Format. xvii, 21

RDF Resource Description Framework. 10

SDMX Statistical Data and Metadata eXchange. 17

UML Unified Modeling Language. 18
URI Uniform Resource Identifier. 10

W3C The World Wide Web Consortium. ix, xi
WEB World Wide Web. 9

XML Extensible Markup Language. 10

1

Introdução

O presente capítulo apresenta uma contextualização do trabalho a desenvolver neste projeto, as motivações que levaram ao seu desenvolvimento, os seus objetivos e, por fim, a organização geral deste documento.

1.1 Enquadramento e Motivação

Usando o motor de pesquisa Google com a seguinte interrogação "*How much data is created every day in 2021*" [1], obtemos o impressionante numero de 14190000000 entradas e verificamos que em 2021 o valor diário estimado de dados produzidos é cerca de 2000000 Terabytes [9]. Na Figura 1.1 percebemos que muitos desses dados são gerados em redes sociais, tendo esses um enorme valor para as empresas que gerem essas redes, tornando-se pouco valiosos para a maioria das pessoas por estarem em redes fechadas. Embora com um menor volume de dados disponibilizados, os portais de dados abertos são hoje uma realidade. Quer os portais sejam privados, e.g. Kaggle [25], quer públicos, e.g. *The official portal for European data* [17], é necessário ter em conta um conjunto de factores para que os dados produzidos se traduzam, de facto, em valor para quem os produz e para a comunidade. Em 2020, a União Europeia (EU) divulgou um estudo sobre o impacto económico dos dados abertos que teve como principais resultados, resumidamente [16]:

[1] Data de pesquisa: 22/07/2021.

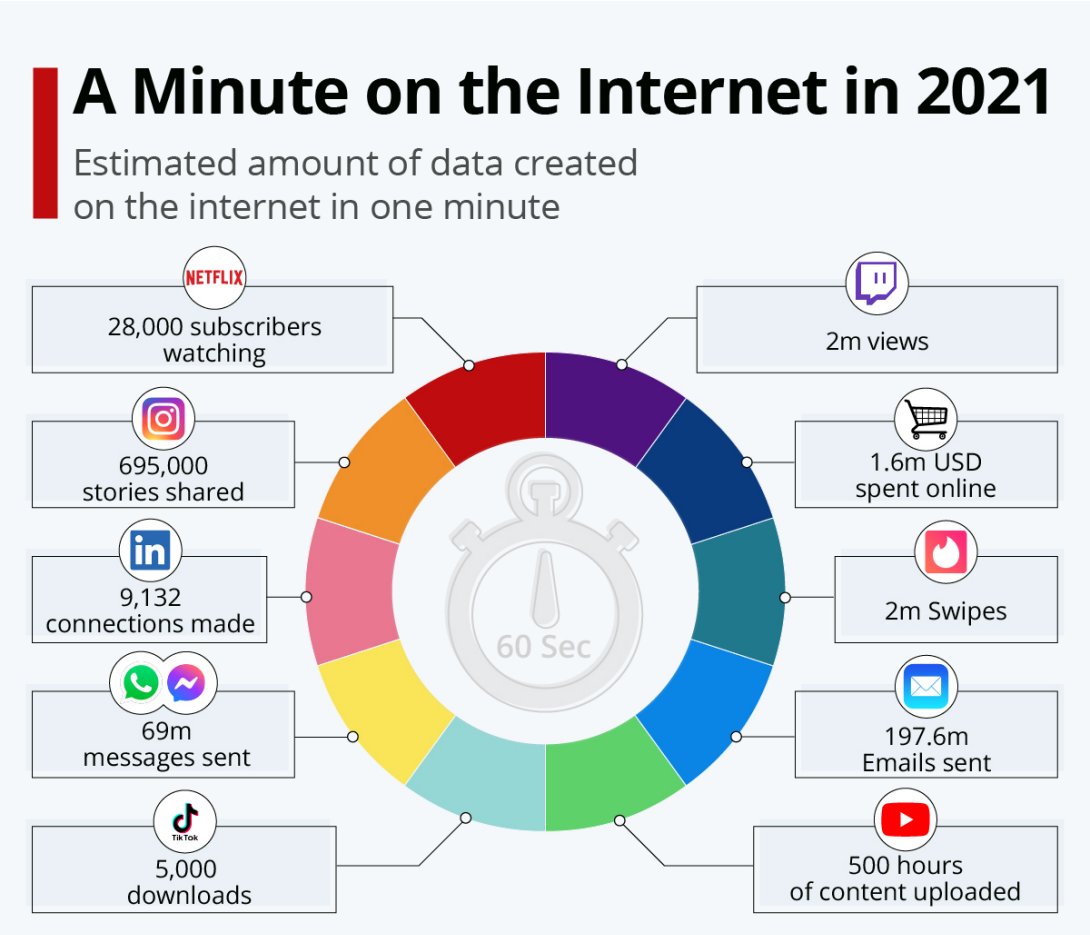

Figura 1.1: Valor estimado para o volume de dados produzidos por minuto num conjunto de actividades na internet. Imagem retirada de [31].

- A especificação e a implementação de conjuntos de dados de valor acrescentado [44] a como parte da nova diretiva relativa aos dados abertos representam uma oportunidade promissora de abordar as exigências de qualidade e quantidade dos dados abertos;

- É importante abordar as exigências de qualidade e quantidade dos dados abertos, embora tal não seja suficiente para concretizar todo o potencial dos dados abertos;

- Os reutilizadores de dados abertos que os usam para gerar nova informação devem estar informados e serem capazes de compreender e explorar todo o potencial;

- A criação de valor dos dados abertos faz parte do desafio mais vasto da integração de dados mais complicada. Isto é uma questão organizacional do que técnica, onde os proprietários de dados internos (departamentos e empresas municipais)

nem sempre compreendem plenamente a necessidade de abrir dados e o valor de o fazer de uma forma padronizada em portais de dados abertos;

- As iniciativas setoriais e a colaboração nos setores público e privado, quer isoladamente, quer transversalmente, fomentam a criação de valor;
- Combinar dados abertos com dados pessoais, partilhados ou colaborativos, é vital para a consecução de maior crescimento do mercado dos dados abertos;
- Para os diferentes desafios, devemos explorar e melhorar os diversos métodos de reutilização dos dados que sejam éticos, sustentáveis e adequados.

Embora a publicação de dados tenha evoluído, ainda existe o problema dos silos de informação — quando um sistema de dados é incompatível ou não integrado a outros sistemas de dados relacionados logicamente, o que dificulta o cruzamento e o reaproveitamento de informações.

Para que os dados abertos tenham impacto e valor, eles tê ser colocados em uso. Portanto, tornar os dados utilizáveis é uma das componente principais de uma iniciativa de dados abertos de sucesso [44].

Duas características distintivas de uma iniciativa de dados abertos de sucesso são quando os criadores de conteúdo aproveitam os dados abertos para produzir novos produtos e serviços ou quando integram os dados abertos a produtos e serviços existentes [44]. Num cenário ideal, isso leva aquilo que se chama de "ciclo virtuoso", onde novos produtos aumentam o pedido por dados abertos, o que catalisa o lançamento de mais conjuntos de dados, o desenvolvimento de novas aplicações e assim por diante [44].

Nesse sentido, a união europeia (EU) define os requisitos necessários a cumprir pelos portais de dados abertos, disponibilizados pelos milhares de entidades públicas. Neles inclui-se o *Data Catalogue Vocabulary Application profile for data portals in Europe* (DCAT-AP) [11] que é uma especificação baseada no Vocabulário de Catálogo de Dados (DCAT) desenvolvido pela W3C.

Em suma, neste mundo ligado pela Internet, onde cada vez mais dados são disponibilizados sem custos, é importante garantir que a semântica dos termos usados para descrever os diferentes conceitos usados sejam claros, para além do domínio de onde são produzidos. Só assim se consegue tirar o maior valor possível dos dados. Foi assim, então, que surgiu este projeto que resulta de uma parceria entre a Câmara Municipal de Lisboa (CML) e o Instituto Superior de Engenharia de Lisboa (ISEL), no âmbito da temática das `Smart Cities`. Mas para se terem cidades ditas "mais inteligentes" é

necessário que um dos seus ativos mais importantes, os dados, sejam adequadamente tratados.

1.2 Descrição dos Objetivos

Uma vez que muitos dos dados a disponibilizar são obtidos por API externas, cujo código fonte não pode ser alterado, e que não cumprem os requisitos para poderem ser mapeados ao formato *Application profile for data portals in Europe* (DCAT-AP) , dificultando a disponibilização desses dados e a sua integração entre entidades. Assim, o objetivo principal deste trabalho assenta na definição e implementação de uma solução que permita transformar uma API externa incompatível com DCAT-AP, numa API DCAT-AP conforme, fazendo o levantamento dos requisitos mínimos exigidos pelo DCAT-AP conjuntamente com os requisitos necessários para a CML.

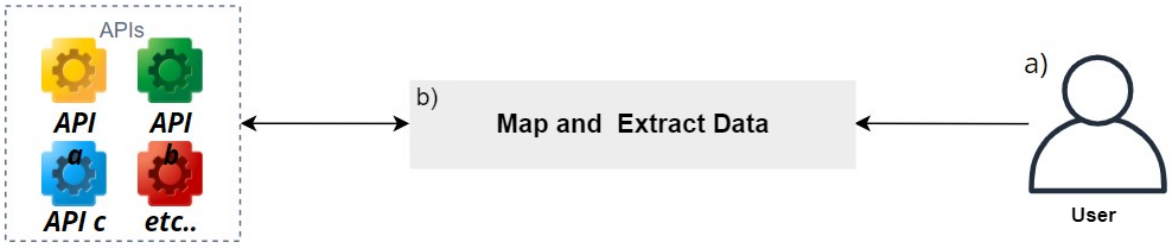

Figura 1.2: Ilustração da estrutura simplificada.

A Figura 1.2 representa de forma simplificada o cenário da solução, onde (o utilizador fornece um link de uma API qualquer onde a informação se encontra alojada e de seguida essa informação é b) extraída e mapeada para o formato DCAT-AP para gerar os seus ficheiros turtle.

1.3 Organização do documento

Este documento está estruturado em seis capítulos, sendo que um deles é o capítulo atual. Os restantes capítulos estão organizados da seguinte forma:

- **Conceitos e estado de arte -** No capítulo 2 é feita uma descrição dos principais conceitos abordados ao longo deste documento de forma a facilitar a contextualização do leitor e o estado de arte, nomeadamente sobre DCAT-AP, CKAN e RDF;

- **Abordagem -** No capítulo 3 será detalhada a abordagem desenvolvida para a conversão de dados;
- **Arquitetura e sua implementação -** No capítulo 4 será abordada a arquitetura proposta e tida em conta para o trabalho e de que forma foi concretizada a implementação do modelo de integração pensado;
- **Avaliação da Conversão de Dados para o DCAT-AP -** No capítulo 4 haverá detalhes sobre a avaliação da precisão e eficácia da conversão de dados. Métricas e resultados obtidos;
- **Conclusão -** No capítulo 6 será feita a Síntese dos principais resultados e contribuições do trabalho. Discussão das limitações e sugestões para futuras pesquisas.

2

Conceitos e Trabalho Relacionado

Neste capítulo apresenta-se uma descrição dos principais conceitos relacionados com o trabalho a desenvolver e referências aos trabalhos relacionados.

2.1 Metadados

Diferentes tipos de metadados são necessários para descrever, descobrir, recuperar, usar, apresentar e preservar recursos digitais, tal como um conjunto de dados [48]. Os metadados podem ser classificados em três categorias, nomeadamente, (i) descritivos, (ii) administrativos e (iii) estruturais, seguindo uma abordagem comum utilizada pela Ciência da Informação [30]. De forma sucinta, podemos descrever cada uma delas da seguinte forma:

Metadados descritivos: descrevem um recurso com o propósito, por exemplo, de descoberta ou identificação. Isso pode incluir elementos como título, resumo, autor e palavras-chave.

Metadados estruturais: indicam como objetos compostos são colocados juntos, por exemplo, como é recursos são dicionário para formar um conjunto de dados.

Metadados administrativos: fornecem informações para auxiliar na gestão de um recurso, como por exemplo, quando e como o mesmo foi criado, qual o tipo de ficheiros e outras informações técnicas, e sobre quem tem acesso a elas.

Nos metadados descritivos, as etiquetas geralmente fazem parte dos metadados associados aos conjuntos de dados de um portal de dados abertos. Em geral, uma etiqueta é um elemento de metadados, identificada por uma *string* que representa seu nome [46]. O responsável pela publicação dos conjuntos de dados em portais de dados abertos é livre para atribuir as etiquetas. Obviamente, estas podem indicar a semântica do conteúdo, mas não necessariamente. Frequentemente as etiquetas que representam a organização que publica o conjunto de dados ou referências aos nomes dos sistemas que o criaram, não representa necessariamente o conteúdo dos dados [46]. Em geral, os metadados são livremente atribuídos pelos responsáveis pela publicação dos dados. Alguns elementos podem ser criados e associados ao software de suporte à publicação de dados, como datas de criação e atualização. Além disso, os elementos de metadados podem ser associados a termos de um vocabulário, como o *Dublin Core*, que define quinze propriedades para uso na descrição de recursos [14] e ontologias, especificando assim as restrições desse vocabulário [33], formando metadados semânticos [1], sejam eles: (i) descritivos, indicando o assunto dos conjuntos de dados e etiquetas associadas; (ii) administrativo, que indica a origem dos dados ou Estrutural, que indica a forma como os dados são disponibilizados.

Apesar dos problemas, as etiquetas são frequentemente utilizadas. O conjunto de dados publicado no portal pode ter zero ou mais etiquetas a ele associadas. A etiqueta pode ser associada a mais de um conjunto de dados. Isso é útil para descobrir conjuntos de dados que tratam de problemas relacionados, por exemplo na figura 2.1, no CKAN se clicar na estiqueta "Lisbon parking" ia ser retornado uma lista de todos os conjuntos de dados que contem a mesma etiqueta.

Figura 2.1: Exemplo das etiquetas de um conjunto de dados da CML. [37]

[1] Qualquer metadados associados a elementos e/ou termos de vocabulários e ontologias.

2.2 *Linked Open Data* e Web Semântica

Nos últimos anos, vários elementos surgiram e expandiram o contexto da ideia original de Web Semântica de Berners-Lee [22]. O W3C [2] iniciou o processo de publicação, implementação e propagação de um conjunto de tecnologias que se foi agrupando à procura da Web Semântica perfeita. Muitos projetos [41] a volta do mundo também se desenvolveram no sentido de formar ambientes semânticos, tanto do ponto de vista da estrutura da informação quanto da recuperação da informação semântica.

Em 2010, foi atualizada e expandida a lista de *Sebastopol* e foram identificados dez princípios que fornecem perspetivas para avaliar até que ponto os dados governamentais são abertos e acessíveis ao público.

Os dados governamentais serão considerados abertos se forem divulgados de forma a atender aos seguintes princípios: Completos, Primários, Atuais, Facilidade de Acesso Físico e Eletrónico, Processáveis por máquina, Não discriminatório, Formatos de propriedade comum ou abertos, Licenças livres, Permanência e Custos de utilização [1].

Estruturar dados abertos de forma semântica não é apenas uma das formas de estabelecer a ligação entre o conceito de dados abertos e a web semântica, mas sim de criar um modelo de estrutura de dados que atende ao cumprimento do quinto princípio de dados abertos e também a previsão da lei de acesso à informação [2], que se refere à possibilidade de processamento de dados por máquina, além da associação entre informações de diferentes bases por meio de relações semânticas.

Essa característica torna os dados não apenas acessíveis e manipuláveis por máquinas, mas também sujeitos a processos de organização que podem facilitar a geração de novos dados, a apresentação de resultados, o cruzamento com outros conjuntos de dados, o aumento do conhecimento para a tomada de decisões e novos modelos de dados gerados a partir de cruzar dados de diversos domínios governamentais.

Para disponibilizar dados numa estrutura semântica é necessário pensar em partes do modelo descrito por Berners-Lee em 2001, *layercake*, estrutura de camadas que apresenta a Web Semântica. Nesse sentido, destaca-se a linguagem RDF, também referida para a representação de dados abertos, o uso de metadados e principalmente a construção e aplicação de ontologias de domínio. Um dos principais objetivos do RDF é especificamente criar uma rede de informação a partir de dados distribuídos.

Construir uma rede de informações onde os nós estão intrinsecamente ligados, formando um grande grafo global, com informações provenientes de diversas fontes diferentes sobre o planeta, é o conceito central do que se chama de *WEB* de dados. Um

[2] ver: https://www.w3.org/

grafo é um modelo matemático muito poderoso que pode ser aplicado para resolver uma série de problemas. Consiste num conjunto de vértices e arestas/arcos [40].

RDF [39] é uma tecnologia relacionada ao XML que é utilizada como base para o processamento de metadados na *web*. A padronização do mesmo cria um modelo de dados e sintaxe para codificação, representação e transmissão de metadados, com o objetivo de torná-los processáveis por máquina e promover a integração dos sistemas de informação disponíveis na *Web*. No modelo RDF temos diferentes formas de representação/serialização, como por exemplo *turtle* que foi criado para poder descrever prefixos relativos e IRIs na estrutura do documento [45]. O formato *turtle* é bastante simples e ainda mais fácil de ler. A representação mais simples é uma sequência de termos (sujeito, predicado, objeto), separados por espaço em branco e terminados por " ." após cada tripla.

```
1 <http://ex.pt/#spiderman> <http://ex2.pt/enemyOf>
2 <http://ex.pt/#green-goblin> .
```

Listagem 2.1: Exemplo de um formato *turtle*.

O RDF é um dos principais suportes tecnológicos na articulação no processo de estruturar a informação pela sua capacidade de associar sujeito, predicado e objeto.

O *Linked Open Data* (LOD) [28], que é apresentado atualmente como a melhor forma de integrar os conceitos e tecnologias da Web Semântica, é um projeto, com um conjunto de regras a seguir, que utiliza os mesmos princípios de ligação semântica de uma rede de Dados, porém tem características específicas, que indica um maior grau de requerimento na criação de sua rede de interligações.

O papel da Web Semântica não é apenas colocar dados na *web* [27]. Trata-se de fazer *links* para que qualquer pessoa ou máquina possa explorar esse conjunto de dados. Com o LOD, quando temos poucos dados, podemos encontrar outros relacionados. A construção do LOD é baseada em quatro princípios publicados por Berners-Lee [27]:

- Usar URIs como nomes para os objetos;
- Usar URIs HTTP para que as pessoas possam consultar esses nomes;
- Quando alguém consulta um URI, fornecer informação RDF útil;
- Incluir cláusulas RDF com links para outros URIs para permitir a descoberta de elementos relacionados.

O modelo de correlação LOD e *Data Web*, que se baseia especialmente no uso de conceitos e na aplicação da linguagem RDF, é muito semelhante ao conjunto de ligações que o cérebro humano realiza para formar e organizar a memória [4].

Apesar da configuração estrutural do modelo de organização de dados implementado por triplos RDF, para divulgar os dados e, essencialmente, disponibilizá-los para uma recuperação mais eficiente, é necessário que o esquema lógico esteja sob uma ontologia, preferencialmente usando uma representação de conhecimento e vocabulário uniformizados e reconhecidos internacionalmente. A utilização da ontologia é semelhante a construir um mapa organizado das palavras e conceitos que pertencem a um determinado campo ou área de conhecimento. Esta abordagem ajuda a dar estrutura aos dados, permitindo que sejam mais facilmente compreendidos por ferramentas de pesquisa e recuperação de informações.

Produzir uma ontologia nem sempre é a melhor ou a mais rápida maneira de disponibilizar os dados num formato semântico. Na maioria dos casos, o uso de uma ontologia pronta e universalmente aceite, reconhecida por uma determinada comunidade, agiliza o processo de disseminação de dados, mas principalmente favorece o processo de recuperação da informação em ambientes semânticos.

Não existe uma regra geral para o uso de uma ontologia normalizada, possivelmente esse não seja o ponto principal de uma organização global e ligação de dados, mas o processo de recuperação será mais fácil para os agentes computacionais de procura semântica, se houver uma sincronia lógica mínima entre os conjuntos de dados do mesmo tipo, por exemplo a propriedade `owl:sameAs` associa um indivíduo a outro [35].

```
1 <rdf:Description rdf:about="#William_Jefferson_Clinton">
2     <owl:sameAs rdf:resource="#BillClinton"/>
3 </rdf:Description>
```

Listagem 2.2: Exemplo da propriedade *owl:sameAs*.

Como no exemplo 2.2 na declaração desta propriedade é indicado que os dois URIs na verdade se referem à mesma coisa: os indivíduos têm a mesma "identidade".

No contexto de criação e uso de ontologias, é importante destacar o uso de vocabulário reconhecido internacionalmente, e este certamente é o núcleo da formação de modelos de dados semânticos, nos quais os significados dos recursos são mundialmente compreendidos.

Quando se refere ao uso de "vocabulário", é usado o termo que a própria equipa do

W3C e a equipa do LOD usam para nomear os elementos que compõem os esquemas de metadados.

Uma grande quantidade de dados ainda é publicada de forma aberta e não estruturada, utilizando recursos de ontologia e a linguagem RDF, porém organizada de acordo com modelos de metadados adaptados, conhecidos por *Application Profiles* [39].

2.3 Portal Europeu de Dados

O Portal Europeu (EU) de Dados recolhe dados e metadados, disponíveis em portais de dados abertos do setor público europeu. Também inclui informações sobre o fornecimento de dados e os benefícios de reutilizá-los.

O portal procura garantir a qualidade dos metadados para ajudar os fornecedores de dados e utilizadores a avaliarem seus registos de dados em relação a um conjunto padrão de critérios. Os critérios são os seguintes:

- Acessibilidade das distribuições;
- Possibilidade de ser lido por máquinas e humanos;
- Conformidade de metadados com a especificação de perfil de aplicativo DCAT-AP;
- Utilizar uma licença reconhecida pelo Portal Europeu de Dados.

Os conjuntos de dados estão disponíveis por categoria com dados de diferentes regiões e países da União Europeia. As categorias são: (i) agricultura, pesca, silvicultura e alimentação; (ii) energia; (iii) regiões e cidades; (iv) Transporte; (v) economia e finanças; (vi) questões internacionais; (vii) governo e setor público; (viii) justiça, sistema judiciário e segurança pública; (ix) ambiente; (x) educação, cultura e desporto; (xi) saúde; (xii) população e sociedade; e (xiii) ciência e Tecnologia.

Apesar de os portais governamentais de dados abertos estarem a tornar-se cada vez mais comuns, e sabe-se que os portais servem como o principal ponto de interação entre o fornecedor de dados e os utilizadores de dados, estes portais têm sido objeto de críticas. Tem sido revelado que os portais oferecem um baixo nível de usabilidade para utilizadores não técnicos e que a facilidade geral de utilização destes portais é muitas vezes inexistente [32].

Portais de fácil utilização são importantes, pois desempenham um papel crucial na reutilização dos dados governamentais abertos, o que logicamente implica que os portais

mais fáceis de utilizar e utilizáveis são capazes de conduzir a níveis mais elevados de criação de valor e aqueles que são menos utilizáveis conduzirão a níveis mais baixos de criação de valor. Para dizer isto de outra forma, é provável que um portal com um menor número de conjuntos de dados, mas que são altamente relevantes e fáceis de utilizar, criará mais valor público do que um portal com um maior número de conjuntos de dados que são menos localizáveis e utilizáveis.

Neste caso, os portais de dados governamentais abertos não devem ter apenas a ver com o fornecimento de grandes quantidades de dados, mas devem ajudar o utilizador a encontrar os dados de que necessita ou deseja. Embora muitas vezes os cidadãos sejam considerados como os principais utilizadores dos dados governamentais abertos, eles não são os únicos. Empresas, governos e Organização não governamental (ONG) utilizam. Cada um destes grupos de utilizadores é capaz de ter necessidades e desejos diferentes para a funcionalidade do portal de dados governamentais abertos, por exemplo, para uma empresa do sector privado, ter acesso fácil a um API pode ser de grande importância, enquanto um utilizador cidadão pode apenas desejar descarregar um *excel* para visualizar. Por esta razão, os portais de dados governamentais abertos devem assegurar um elevado nível de usabilidade entre audiências e grupos de interessados, para assegurar níveis mais elevados de reutilização de dados.

2.4 *Data Catalog Vocabulary*

O Data Catalog Vocabulary (DCAT) é um vocabulário RDF criado para facilitar a interoperabilidade entre catálogos de dados [10]. Criado no *Digital Enterprise Research Institute* (DERI), é uma unidade do Grupo de Trabalho W3C *Government Related Data* (GLD), organização que atualmente fornece recomendações e atualizações sobre este vocabulário. O DCAT é usado para descrever conjuntos de dados em catálogos de dados, com o objetivo de melhorar a descoberta de recursos, a disponibilidade descentralizada de catálogos e o processamento digital [10]. Também integra metadados de outros vocabulários já existentes, como *Dublin Core*[3] e FOAF[4].

[3]Ver: https://www.dublincore.org/
[4]ver: http://xmlns.com/foaf/spec/

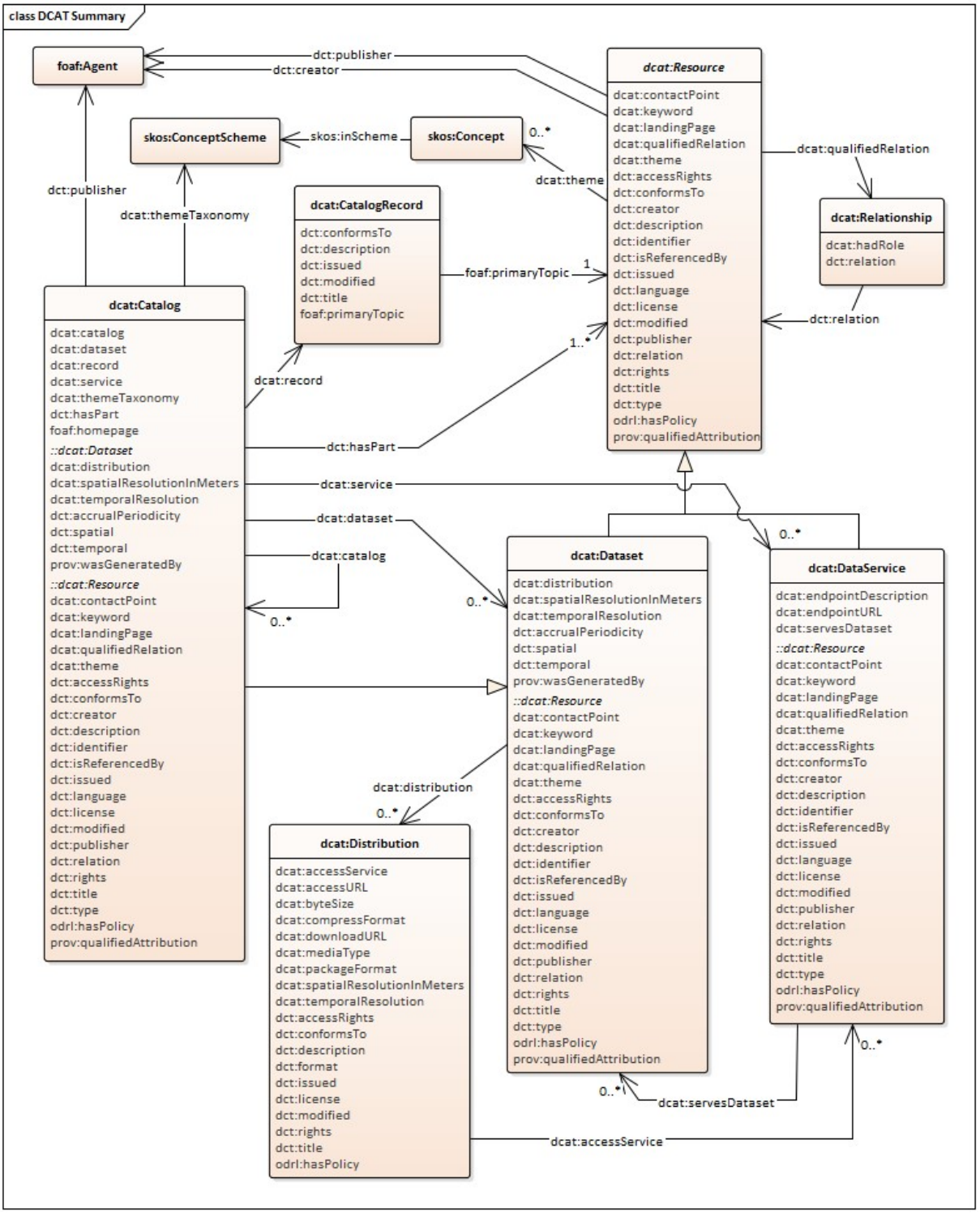

Figura 2.2: Visão geral do modelo DCAT, mostrando as classes de recursos que podem ser membros de um catálogo e os relacionamentos entre eles [10].

Percebe-se na Figura 2.2 que os elementos são inseridos a partir de outros esquemas, predominantemente do Dublin Core, mas também de outros como o SKOS[5] e FOAF.

[5]Ver: https://www.w3.org/2004/02/skos/

O DCAT é baseado em seis classes:

- `dcat:Catalog` representa um catálogo em que cada item individual é um registo de metadados que descreve um recurso; o âmbito do `dcat:Catalog` são coleções de metadados sobre conjuntos de dados ou serviços de dados. Na Listagem 2.3 é representado um catalogo que está em conformidade com o DCAT-AP e que aponta para seis conjuntos de dados;

```
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix e: <http://lisboaaberta.cm-lisboa.pt/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

e:dadosabertos a dcat:Catalog ;
dct:description "Conjunto de parâmetros ambientais inclui: Qualidade do Ar, Ruído, Estado do Tempo e Tráfego." ;
dct:language "None" ;
dct:license <http://www.opendefinition.org/licenses/cc-zero>;
dct:publisher "None" ;
dct:title "Monitorização de Parâmetros Ambientais da Cidade de Lisboa" ;
dcat:dataset e:dataset-001, e:dataset-002, e:dataset-003, e:dataset-004, e:dataset-005, e:dataset-006 ;
dcat:themeTaxonomy <>;
foaf:homepage "None" .
```

Listagem 2.3: Exemplo de um catálogo em DCAT-AP no formato *turtle*.

- `dcat:Resource` representa um conjunto de dados, um serviço de dados ou qualquer outro recurso que possa ser descrito por um registo de metadados num catálogo. Essa classe não deve ser usada diretamente, mas é a classe pai de `dcat:Dataset`, `dcat:DataService` e `dcat:Catalog`. Os itens de membro num catálogo devem ser membros de uma das subclasses, ou de uma subclasse destas, ou de uma subclasse `dcat:Resource` definida num perfil DCAT ou outro aplicação de DCAT. `dcat:Resource` é efetivamente um ponto de extensão para definir um catálogo de qualquer tipo de recurso. `dcat:Dataset` e `dcat:DataService` pode ser usado para conjuntos de dados e serviços que não estão documentados em nenhum catálogo;

- `dcat:Dataset` representa um conjunto de dados como na Listagem 2.4. Um conjunto de dados é uma coleção de dados, publicada ou recuperados por um único agente. Os dados vêm em muitas formas, incluindo números, palavras, pixels, imagens, som e outros multi-meios, e potencialmente outros tipos, qualquer

um dos quais pode ser recolhido num conjunto de dados;

```
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix e: <http://lisboaaberta.cm-lisboa.pt/> .

dct:description "Ultima medicao recebida de todos os parametros e de
todas as localizacoes (dados horarios)" ;
dct:title "Dados em tempo real - Ultima medicao recebida de ..." .
e:dataset-001 a dcat:Dataset .

```

Listagem 2.4: Exemplo de um *dataset* em DCAT-AP no formato *turtle*.

- `dcat:Distribution` representa uma forma acessível de um conjunto de dados, representado na listagem 2.5, como por exemplo um ficheiro para *download*;

```
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix e: <http://lisboaaberta.cm-lisboa.pt/> .
dct:accessURL <http://opendata-cml.qart.pt/lastmeasurements>.
e:dist-001 a dcat:Distribution .

```

Listagem 2.5: Exemplo de uma distribuição (recurso) em DCAT-AP no formato *turtle*.

- `dcat:DataService` representa um serviço de dados como está na listagem 2.6. Um serviço de dados é uma coleção de operações acessíveis por meio de uma interface (API) que fornece acesso a um ou mais conjuntos de dados ou funções de processamento de dados;

```
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix e: <http://lisboaaberta.cm-lisboa.pt/> .

e:dataServ a dcat:DataService.
dct:tite "titulo do servico".
dcat:endpointURL <http://opendata-cml.qart.pt/serviceEndpoint/>;
dcat:servesDataset e:dataset-001, e:dataset-002;

```

Listagem 2.6: Exemplo de um serviço de dados em DCAT-AP no formato *turtle*.

- `dcat:CatalogRecord` representa um item de metadados no catálogo, como

está representado na listagem 2.7, principalmente em relação às informações de registo, como quem adicionou o item e quando.

```
1
2   @prefix dcat: <http://www.w3.org/ns/dcat#> .
3   @prefix dct: <http://purl.org/dc/terms/> .
4   @prefix e: <http://lisboaaberta.cm-lisboa.pt/> .
5   @prefix foaf: <http://xmlns.com/foaf/0.1/> .
6
7   e:catrec-001 a dcat:CataLogRecord .
8   dct:modified "Novembro 14, 2021, 15:28 (Europe/Lisbon)" ;
9   foaf:primaryTopic "dataset-001" .
10
```

Listagem 2.7: Exemplo de um *Catalog Record* em DCAT-AP no formato *turtle*.

O perfil de aplicações DCAT para portais de dados na Europa (DCAT-AP) é uma especificação baseada no Vocabulário de Catálogo de Dados (DCAT) desenvolvido pela W3C. DCAT-AP é uma extensão Linked Data do DCAT que adiciona campos de metadados e intervalos obrigatórios para propriedades específicas [12], como visto por exemplo na listagem 2.3.

Este perfil de aplicação é uma especificação para os registos de metadados para satisfazer as necessidades específicas de aplicações dos portais de dados na Europa, proporcionando simultaneamente a interoperabilidade semântica com outras aplicações com base na reutilização de vocabulários controlados estabelecidos e mapeamentos para os vocabulários de metadados existentes (por exemplo, Dublin Core, SDMX[6], metadados INSPIRE [29], etc.). Além disso, o DCAT-AP fornece uma especificação comum para descrever conjuntos de dados do setor público na Europa, a fim de permitir a troca de descrições de conjuntos de dados entre portais de dados. O DCAT-AP permite:

- Catálogos de dados para descrever as suas coleções de conjuntos de dados utilizando uma descrição normalizada, mantendo simultaneamente o seu próprio sistema de documentação e armazenamento;
- Agregadores de conteúdos, como o Portal Europeu de Dados, para agregar essas descrições num único ponto de acesso;
- Consumidores de dados para encontrar mais facilmente conjuntos de dados através de um único ponto de acesso.

[6]ver: https://sdmx.org/

DCAT-AP tem uma extensão GeoDCAT-AP[7] para descrever conjuntos de dados geoespaciais, séries de conjuntos de dados e serviços. Outra extensão, StatDCAT-AP[8], fornece especificações e ferramentas que aumentam a interoperabilidade entre descrições de conjuntos de dados estatísticos dentro do domínio estatístico e entre dados estatísticos e portais de dados abertos.

As classes de *Application Profile* na Figura 2.3 mostram um diagrama UML de todas as classes e propriedades incluídas no DCAT-AP.

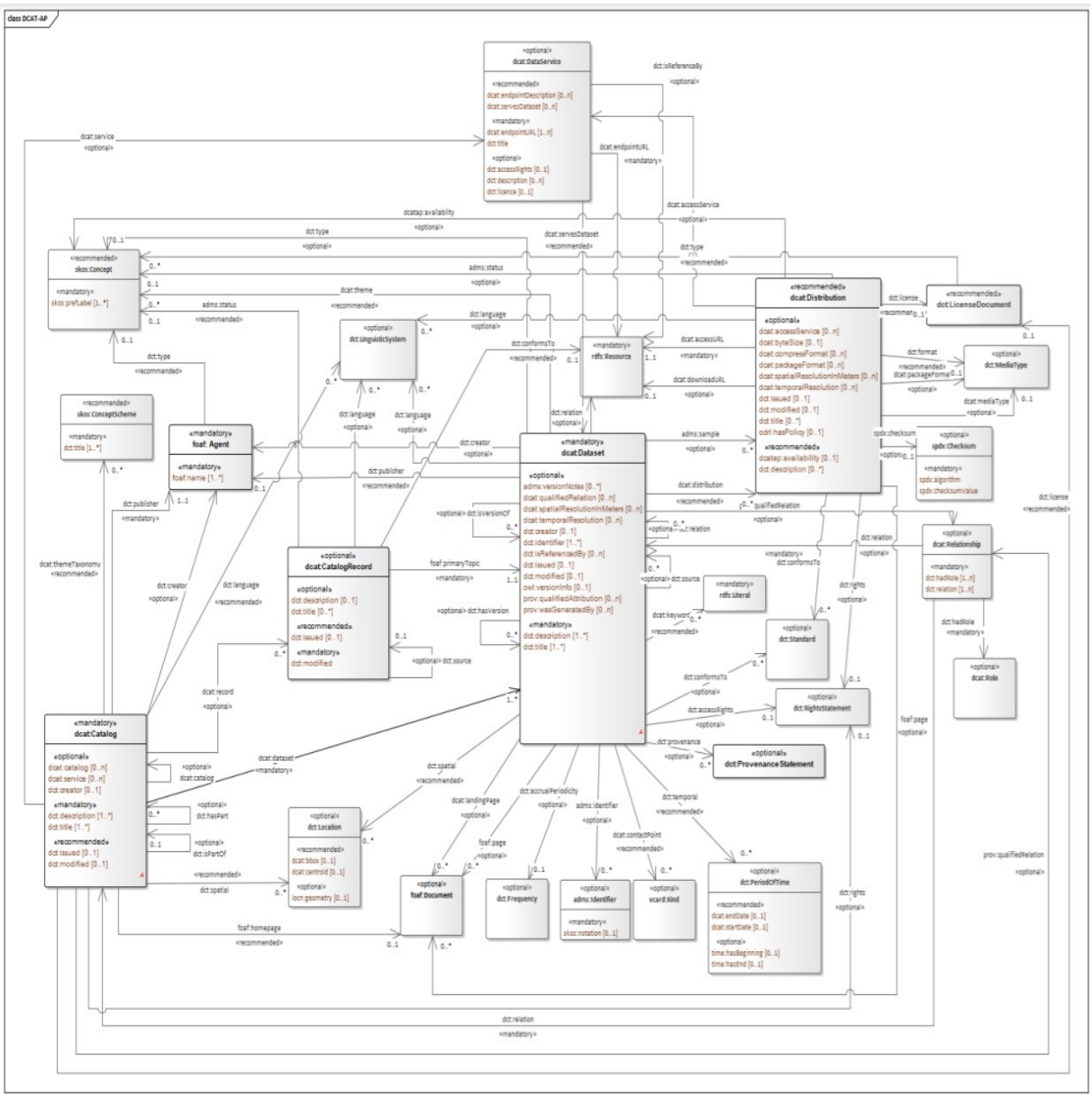

Figura 2.3: Diagrama de classe UML do DCAT-AP [12].

[7]Ver: https://inspire.ec.europa.eu/good-practice/geodcat-ap
[8]Ver: https://joinup.ec.europa.eu/release/statdcat-ap-v100

Para estar em conformidade com o DCAT-AP[9] , quem oferece os metadados deve [12]:

- Fornecer uma descrição do tipo Catálogo, incluindo cumulativamente pelo menos as propriedades obrigatórias (*dataset, description, publisher, title*);
- Fornecer informações para as propriedades obrigatórias (*primary topic, update/ modification date*). O fornecimento das descrições dos *Catalog Records* é opcional;
- Fornecer descrições dos *datasets* no catálogo, incluindo, pelo menos, as propriedades obrigatórias (*description, title*);
- Fornecer descrições de Distribuições(recursos), se houver, dos *Datasets* no Catálogo, incluindo pelo menos a propriedade obrigatória (*access URL*);
- Fornecer descrições de Data Services, se houver, dos Datasets no Catálogo, incluindo, pelo menos, as propriedades obrigatórias (*endpoint URL,title*);
- Fornecer descrições de todas as organizações envolvidas nas descrições do Catálogo e *Datasets*, incluindo pelo menos a propriedade obrigatória (*Name*);
- Fornecer descrições de todos os esquemas de categoria que contêm as categorias que são declaradas em qualquer uma das descrições dos *Datasets* no catálogo, incluindo pelo menos a propriedade obrigatória (*title*);
- Fornecer descrições de todas as categorias envolvidas nas descrições dos *Datasets* no Catálogo, incluindo, pelo menos, a propriedade obrigatória (*preferred label*).

Para uma lista completa das propriedades da especificação W3C DCAT 2.0 em que o DCAT-AP expressa restrições adicionais ou que o DCAT-AP deseja enfatizar seu uso, consulte o documento de origem [12].

2.5 *Comprehensive Knowledge Archive Network (CKAN)*

A maioria das iniciativas de abertura de dados realizadas por processo governamentais tem sido disponibilizados em portais de dados abertos (ODPs) estruturados usando a plataforma *Comprehensive Knowledge Archive Network*[10] (CKAN) [38]: uma ferramenta de código aberto e gratuita, com foco na criação de sites para disponibilização de dados abertos, desenvolvida e mantida pela *Open Knowledge Foundation* [7].

[9]**As classes recomendadas e opcionais podem ter propriedades obrigatórias, mas só se aplicam se e quando uma instância de tal classe estiver presente em uma descrição.

[10]ver: `https://ckan.org/`

A ideia do CKAN é tornar os dados facilmente acessíveis, localizáveis e apresentáveis, ao fornecer ferramentas para simplificar a publicação, partilha, pesquisa e a utilização de dados, onde os dados são organizados e disponibilizados ao público em geral em unidades chamados conjuntos de dados. Um conjunto de dados é uma coleção de dados indexado numa instância do portal. Os conjuntos de dados podem atender às especificações de vários padrões de organização de dados, que caracterizam os formatos de ficheiro de dados. Isso significa que o conjunto de dados pode ser disponibilizado ao público em diferentes formatos [6].

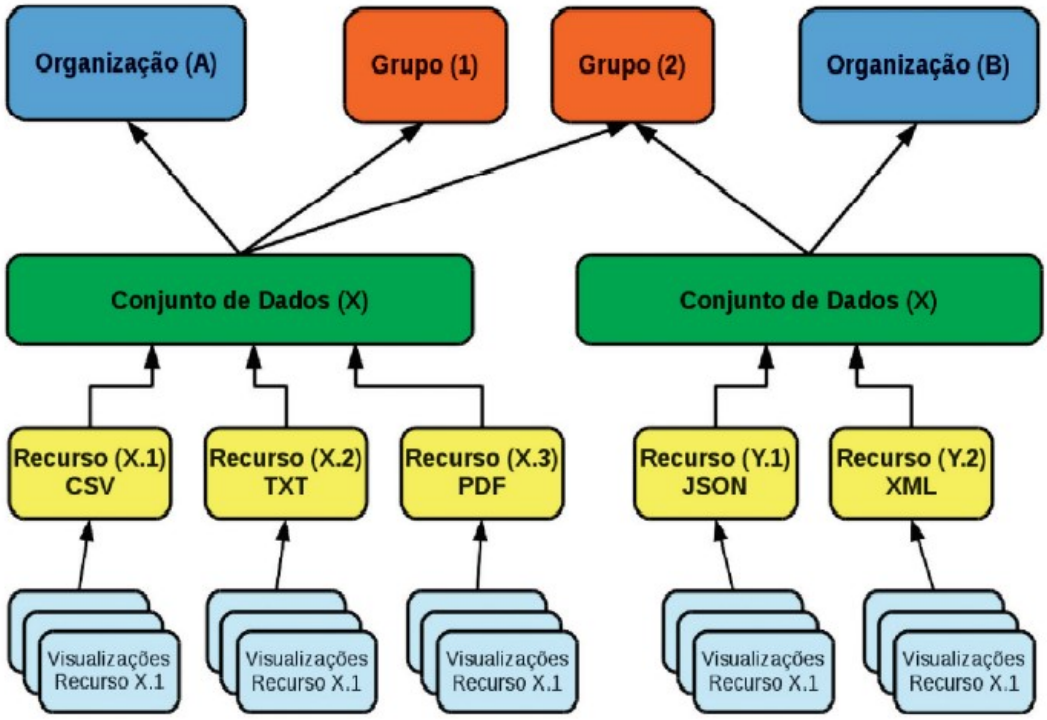

Figura 2.4: Conjuntos de dados agrupados [6].

A criação de grupos temáticos que reúnem conjuntos de dados independentes das organizações produtoras de dados (Fig. 2.4) facilita sua organização. Essas organizações lógicas facilitam a recuperação de dados correlacionados, independentemente da organização a que pertençam. Os recursos têm formatos nativos, podendo ser PDF, TXT, CSV, RDF, entre outros.

O CKAN não define regras sobre como devem os recursos de dados ser organizados. A única condição para o CKAN é que a publicação do conjunto de dados tenha pelo menos um recurso associado, como se vê na Figura 2.5.

Figura 2.5: Exemplo de um conjunto de dados do portal "Lisboa aberta"[11]com 6 recursos associados: cinco em JSON e um em PDF [37].

No processo de publicação dos conjuntos de dados no CKAN, pode registar-se a organização responsável por disponibilizar os dados. O registo da organização pode ou não ser obrigatório, dependendo de como o administrador da instância configurou o ambiente. Cada conjunto de dados de uma organização tem uma página com uma coleção de metadados. Para a visualização dos conjuntos de dados em formato tabular, imagens e até mesmo informações de localização geográficas, também é possível observá-los usando gráficos, tabelas e mapas na própria interface CKAN, se assim o publicador configurar [6]. Porém, para os ficheiros RDF, não há suporte disponível a não ser o *download* dos dados.

Originalmente, o CKAN fornece uma série de opções de descritor para cada conjunto de dados. Os metadados são especificados pelo publicador, por exemplo, título, identificador exclusivo via URL, descrição, licença, etiquetas, entre outros. Os metadados podem ser extraídos e explorados fora do CKAN usando a extensão "ckanext-dcat"[12], por exemplo a classe `ckanext.dcat.processors.RDFParse` permite ler serializações RDF em diferentes formatos e extrair dicionários do conjunto de dados CKAN.

Assim, o mesmo, procura por conjuntos de dados e distribuições (recursos) DCAT e cria conjuntos de dados e recursos CKAN, como dicionários que podem ser passados para as funções `package_create` ou `package_update` [13].

[11]Fonte: https://lisboaaberta.cmlisboa.pt/index.php/pt/
[12]ver: https://extensions.ckan.org/extension/dcat/

```
from ckanext.dcat.processors import RDFParser, RDFParserException

    parser = RDFParser()

    # Parsing a local RDF/XML file

    with open('datasets.rdf', 'r') as f:
        try:
            parser.parse(f.read())

            for dataset in parser.datasets():
                print('Got dataset with title {0}'.format(dataset['title'])

        except RDFParserException, e:
            print ('Error parsing the RDF file: {0}'.format(e))

    # Parsing a remote JSON-LD file

    import requests

    parser = RDFParser()

    content = requests.get('https://some.catalog.org/datasets.jsonld').
    content

    try:
        parser.parse(content, _format='json-ld')

        for dataset in parser.datasets():
            print('Got dataset with title {0}'.format(dataset['title'])

    except RDFParserException, e:
        print ('Error parsing the RDF file: {0}'.format(e))
```

Listagem 2.8: RDF DCAT Parser em Python [13].

Este é o *plug-in* oficial que permite estender o CKAN com uma interface DCAT-AP disponível [13]. No entanto, a sua principal funcionalidade é mapear as estruturas de dados preexistentes para uma serialização RDF, o que significa que traduz os dados existentes para um formato compatível com a RDF. No entanto, este *plug-in* não fornece funcionalidades nativas para dados ligados [26], ou seja, não oferece recursos para criar ou gerir dados ligados diretamente.

O CKAN é muito flexível quando se trata de personalização. Na interface web, o

[13] Ver: https://github.com/ckan/ckanext-dcat

CKAN *Action-API*[14] que está na interface web torna-o flexível, permitindo estender características e funcionalidades existentes ou incluir novos, adaptar extensões já desenvolvidas por terceiros ou através da opção de desenvolver uma nova extensão. Esse capacidade salienta a escolha do CKAN para criar portais para dados abertos governamentais. Cada instância do CKAN pode desenvolver, alterar ou utilizar as extensões de que necessita, sem qualquer tipo de hierarquia ou associação com outras instâncias [6].

Apesar de ser necessárias informações sobre o tipo de licença a ser usada para cada conjunto de dados, o CKAN não fornece nenhum tipo de controle de acesso aos próprios recursos. O único controle de acesso é para o editor de uma organização [6].

2.6 *Fiware*

Em 2011, a Comissão Europeia criou o programa *Future Internet Public Private Partnership* com o objetivo de aproveitar os benefícios da Internet do futuro e contribuir para o aumento da inteligência de diversos processos na Europa. [20]

As tecnologias do *FIWARE* visam à combinação da Internet das Coisas com a gestão de Informações de Contexto e serviços de *Big Data* na nuvem para facilitar o processamento, análise e visualização dessas informações.

Uma das formas de utilizar todas as capacidades disponibilizadas pelo FIWARE é através do *FIWARE Lab*. O *FIWARE Lab* é descrito como «um ambiente não-comercial para inovação e experimentação». Essa plataforma possibilita a criação de contas de utilizador em que máquinas virtuais e endereços IP flutuantes podem ser configurados para a utilização dos componentes do *FIWARE*.

Os componentes do *FIWARE* são conhecidos como *Generic Enablers* e possuem APIs bem definidas que implementam diversas funcionalidades definidas pela plataforma. As implementações de *GEs* mais bem estabelecidas incluem o *Orion Context Broker*, *Cygnus*, *IDAS*, *Wirecloud*, *Keyrock* e *CKAN extensions*. Além desses, também é possível construir novas implementações de GEs a partir da descrição de arquitetura do *FIWARE*. Para entender o *Orion Context Broker*, é necessário entender a *API NGSI* do *FIWARE*.

No contexto do DCAT, é importante observar que o DCAT concentra-se na descrição de datasets e metainformações associadas a eles, mas não impõe diretamente como

[14]Ver: https://docs.ckan.org/en/2.9/api/#api-versions

os datasets devem ser implementados. É aqui que o *FIWARE* desempenha um papel relevante.

O *FIWARE* oferece ferramentas e padrões que podem ser aplicados para impor consistência e estrutura na implementação de datasets, garantindo que estes cumpram os requisitos e padrões estabelecidos. Isso é fundamental para promover a interoperabilidade, qualidade e acessibilidade dos datasets, tornando-os mais eficazes para os utilizadores finais e para as iniciativas de dados abertos.

Portanto, ao unir o DCAT e o *FIWARE*, não apenas descrevemos os dados, mas também garantimos que estes sejam implementados de maneira consistente e estruturada, cumprindo os padrões estabelecidos no DCAT e aproveitando as capacidades oferecidas pelo *FIWARE*. Esta abordagem abrangente contribui para um ecossistema de dados mais robusto e eficiente.

3

Abordagem

Embora existam várias maneiras de disponibilizar dados e metadados [36], e.g. através de ficheiros CSV, cada uma apresenta problemas e desafios diferentes.

A abordagem seguida neste trabalho assume que dados e metadados são disponibilizados através da web API REST [19] que é a forma preferencial da CML na integração de dados. Usando os padrões existentes para documentação de API, nomeadamente o OpenAPI [34], onde os recursos se assumem bem descritos, é possível ter uma solução semi-automática para fazer o mapeamento para um modelo de metadados compatível com DCAT-AP, mas incluindo aspetos específicos para o domínio do problema, neste caso para a CML. Esta abordagem permite evoluir para cenários onde se podem usar ontologia [15], mas fora do âmbito do trabalho. Sendo assim, para cada API tem de haver um conjunto de dados específico e metadados de cada recurso ou conjunto de recursos, relativamente a cada uma das chamadas a API, como é representado na Figura 3.1, a qual vai ser descrita ao longo das secções seguintes. O suporte ao mapeamento será feito através de uma aplicação WEB, permitindo ao utilizador interagir com o sistema e alterar/melhorar o mapeamento, de uma forma iterativa.

Para chegar a essa abordagem, foram realizados vários ensaios, utilizando a técnica de *web scraping*. Esses ensaios levaram à solução apresentada na Figura 3.1. Durante o processo, ficou claro que os dados disponibilizados pela CML são suficientes para cumprir com a norma do DCAT-AP e que é possível gerar os metadados correspondentes.

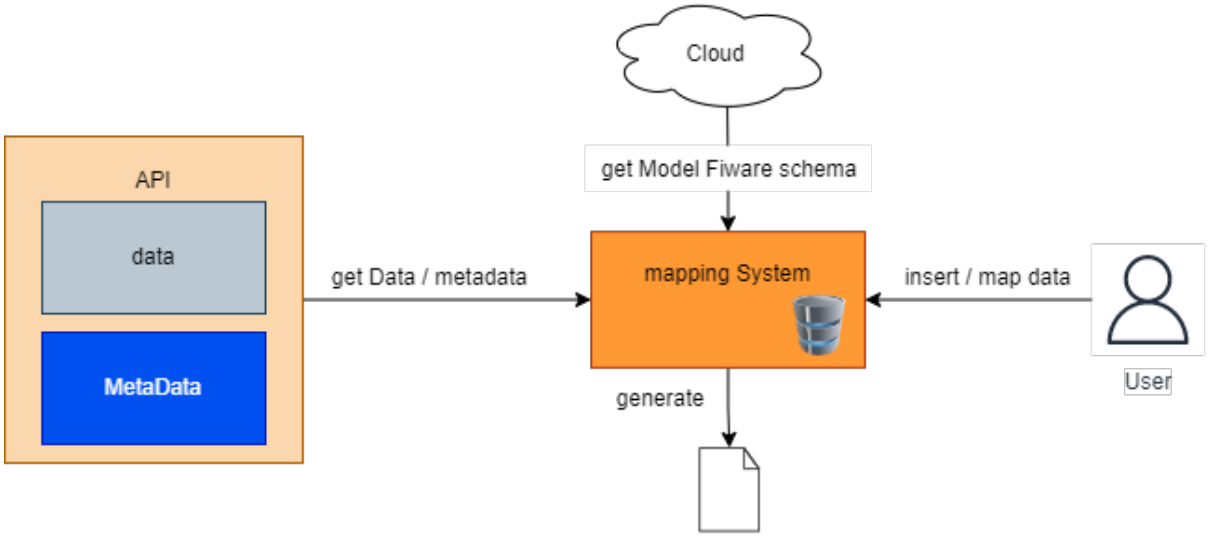

Figura 3.1: Ilustração da solução proposta para organização dos metadados.

3.1 Trabalho preliminar

Como trabalho preliminar foi realizado um protótipo onde, através de *web scraping*, foram extraídos os dados pretendidos do portal de dados abertos da CML [37] e gerados ficheiros no formato *turtle* com os dados estandardizados. Esta implementação teve dois propósitos:

- Mostrar que os dados existentes no portal de dados aberto da CML têm a informação suficiente para cumprir com a norma DCAT-AP;
- Fazer a geração de um ficheiro de metadados, num formato normalizado.

Embora a solução final não seja assente neste tipo de abordagem, o *web scraping* pode ser bastante útil quando a fonte da qual se pretende obter dados, não tem uma API ou, caso tenha, forneça apenas um acesso limitado aos dados. Mais tarde verificou-se que a CML tem a API CKAN que disponibiliza os dados necessários. Assim surgiu a ideia de generalizar o *output* de API que é devolvido em diferentes formatos precisando de fazer a sua integração, portanto, optou-se por usar API em vez do *web scraping* por não ser uma opção fidedigna uma vez que a estrutura da página web pode mudar e nem todos os sites tem a mesma estrutura para apresentar os dados, uma vez que o objetivo é estandardizar o *output* de qualquer API, mesmo não sendo da CML.

3.2 Modelo de metadados intermédio

O modelo de metadados intermédio é o modelo compreendido pelo *mapping system*, representando uma versão "traduzida" dos dados recebidos da API web. Para ilustrar este conceito, considere o exemplo de um ficheiro *.java*, que é legível por programadores, mas não é diretamente executável pela máquina. Para torná-lo executável, é necessário compilá-lo num ficheiro *.class*, que contém código binário compreensível pela máquina.

O modelo de metadados intermédio irá conter todos os campos necessários para mapear para o Modelo final de metadados que segue o DCAT-AP e o qual a ele será mapeado os campos de output da API. Como no exemplo anterior da Listagem 3.3, todos os campos serão descritos por três variáveis:

- ***Type*** — que representa um *link* correspondente a discrição da metadata do termo mantido, por exemplo, pela *Dublin Core*[1] ou *Schema.org*[2];
- ***Description*** — que representa uma pequena explicação o que campo representa;
- ***Parent*** — que indica a que ficheiro o campo vai pertencer sendo o campo de um Catalogo ou um dataste ou uma distribuição.

Os campos ***Type*** e ***Parent*** são muito importantes, pois, vão permitir o mapeamento automático entre os campos da API e os campos do modelo de metadados intermédio que são acessíveis no *mapping system* que tem ligação com a API Web na Figura 3.1. Este assunto será abordado em mais detalhe na Secção 3.4.

3.2.1 Campos de extensão

A infraestrutura da CML que integra a maioria dos dados existentes no município é *FIWARE enabled* [5]. Como a maioria das integrações são feitas via WEB API [43], e há a necessidade de convergir para um conjunto de modelos abertos e normalizados [42], a abordagem vai suportar alguns destes modelos (por exemplo, device[3], Weather[4] e Air Quality Observed[5]). Os seus campos obrigatórios serão adicionados ao modelo de

[1]ver: https://www.dublincore.org/specifications/ dublin-core/dces/
[2]ver: https://schema.org/
[3]Ver: https://smart-data-models.github.io/dataModel.Device/DeviceModel/schema.json
[4]Ver: https://smart-data-models.github.io/dataModel.Weather/WeatherObserved/schema.json
[5]Ver: https://smart-data-models.github.io/dataModel.Environment/AirQualityObserved/schema.json

metadados intermédio, sendo que estes já se encontram descritos no esquema de cada modelo, os quais serão consultados pelo sistema via pedido *http* como na Figura 3.1.

Por exemplo, se o utilizador indicar que os dados são compatíveis com o modelo *device*, obrigatoriamente, no modelo de metadados intermédio tem que conter os campos ***id, type, category, controlledProperty, manufacturerName, brandName e modelName*** que estão descritos no *link* deste *schema*[3] como obrigatórios.

3.2.2 Modelo entidade associação

O modelo de metadados intermédio vai ter algumas relações entre as partes representadas na Figura 3.2, exatamente o Fiware Model e o nome das classes do DCAT-AP, o ***Catalog***, ***Dataset*** e a ***Distribution***, ou seja, os valores que o campo ***Parent*** pode tomar no modelo de metadados intermédio.

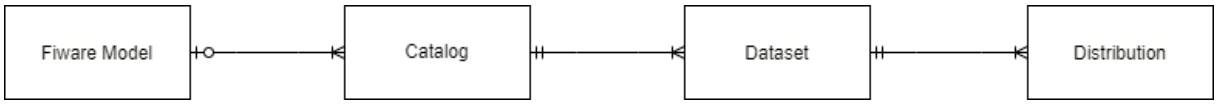

Figura 3.2: Diagrama de relações entre entidade.

Assim haverá obrigatoriamente um ***Catalog***, que pode seguir ou não um ***Fiware Model***, contendo um ***Dataset*** ou mais, que por sua vez cada ***Dataset*** terá que ter uma ou várias ***Distribution***.

A representação *flat* do modelo intermédio não fará perder as relações entre as partes, uma vez que para representar a relação entre o ***Catalog*** e o ***Dataset***, automaticamente se sabe que o ***Catalog*** contém um ou mais ***Dataset*** quando uma lista de um ou mais títulos é apresentada. Entretanto, o mesmo não se aplica ás ***Distribution*** do dataset. Nesse caso, foi adicionado um campo extra, ***partOfDataset***. Esse campo utiliza o índice da lista de um campo específico, cujo valor do campo ***Parent*** seja definido como *Dataset*. Essa abordagem permite que o índice da lista seja atribuído ao campo ***partOfDataset***. Por exemplo, se a lista do campo "título" do dataset contiver dois valores, a representação seria assim `[índice do primeiro dataset, índice do segundo dataset]`. Além disso, a representação da distribuição é simplesmente o próprio índice da lista no campo ***partOfDataset***, conforme ilustrado na Figura 3.3.

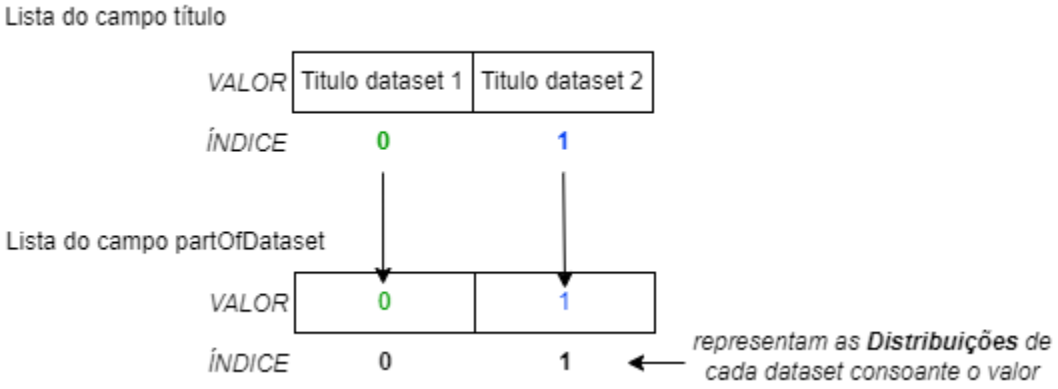

Figura 3.3: Relação entre os datasets e as suas distribuições usando Listas.

No exemplo da Listagem 3.1, os valores presentes na lista do campo ***partOfDataset***, [0, 0], possuem uma interpretação específica. O valor "0" corresponde ao índice zero da lista do campo **title**, que, por sua vez, contém o valor "titulo A". Já os índices "0" e "1" na lista do campo ***partOfDataset*** representam as duas distribuições associadas a esse dataset. A primeira distribuição é identificada pelo título "dist 1", enquanto a segunda é representada pelo título "dist 2". Dessa forma, o campo ***partOfDataset*** estabelece a relação entre as distribuições nos índices "0" e "1", as quais pertencem ao dataset de índice "0" na lista do campo **title**, cujo valor é "title A".

```
{
    "title": ["title A", "title B"],     //isto quer dizer que temos dois datasets
    "titleDist": ["dist 1", "dist 2"],   //isto quer dizer que temos duas distribuições
    "partOfDataset": [0,0]              //isto quer dizer que as duas distribuições pertencem ao primeiro dataset
}
```

Listagem 3.1: Exemplo relação entre as distribuições e o *dataset*.

3.3 Requisitos Web API

Na Figura 3.1 pode observar-se que se propõem como solução para o problema que a WEB API tenha dois recursos separados: (i) Um conjunto de *endpoints* que permitem aceder aos dados disponibilizados pela WEB API; (ii) Um *endpoint* que dá acesso a metadados da WEB API, cujos valores não podem ser obtidos da informação de contexto existentes nos outros *endpoints*. Assim, assume-se que no *schema* de metadata tem que descrever os campos da mesma forma do modelo de metadados intermédio, como ilustrado na Listagem 3.3, pelo que o nome dos campos é irrelevante. O *schema* da API de dados não é importante uma vez que são suportados objetos aninhados onde também o nome dos campos é irrelevante. No entanto, as API tem que cumprir os seguintes requisitos:

1. É requerido um *endpoint* que devolva um objeto, ou uma lista de objetos, em formato *JSON*, onde cada campo pode ser representado de uma forma das seguintes representações:

   - Title: "título do catálogo";
   - Title: ["título do catálogo"];
   - Title: [{ Title: "título do catálogo"}].

2. É requerido um segundo *endpoint* que devolve metadados da API de dados no formato *JSON*;

3. É requerido um *schema* de metadados como na Listagem 3.2.

Considerando o exemplo seguinte, que requer apenas dois campos, título e descrição, será utilizado o modelo de metadados intermédio do sistema de mapeamento representado na Listagem 3.3.

Sendo forma, a API de metadados deve devolver os metadados de acordo com o esquema apresentado na Listagem 3.2, o qual é compatível com o modelo intermédio da Listagem 3.3. Importa referir que a ordem dos campos, os respetivos nomes e a inclusão de campos adicionais não têm relevância, como ilustrado no exemplo da Listagem 3.4.

Tendo um *output* mais completo da API de metadados, conforme apresentado no Anexo C, juntamente com um modelo de metadados intermédio mais abrangente do que o exemplificado na Listagem 3.3 (ou seja, no Anexo A), e considerando também o output de dados da API descrito no Anexo B, podemos afirmar que essa combinação atenderá plenamente aos requisitos pretendidos.

3.4 Estratégias de mapeamento

Tendo uma API de metadata que devolve um objeto *JSON* de todos ou alguns campo da API de dados descritos, segundo o *schema* representado na Listagem 3.2 e o modelo de metadados intermédio que descreve o conjunto de campos necessários e que segue o mesmo *schema* da API de metadados, ou seja, o *schema* representado na Listagem 3.2.

Assim para fazer o mapeamento automático é utilizando um algoritmo simples, o qual compara o ***type*** e o ***parent*** dos metadados recebidos da *API* com o ***type*** e o ***parent*** do modelo de metadados intermédio.

```
{
  "$schema": "http://json-schema.org/schema#",
  "$id": "http://json-schema.org/schema#",
  "title": "Core metaData SCHEMA",
  "description": "metadata schema",
  "type": "object",
  "properties": {
    "patternProperties": {
           "^.*$": {
      "type": {
        "type": "string",
        "format": "uri",
        "enum": [   "https://schema.org/name",
                    "https://schema.org/description",
                    "https://schema.org/keywords",
                    "https://schema.org/publisher",
                    "http://purl.org/dc/terms/accessRights",
                    "https://www.w3.org/ns/adms#sample",
                    "https://schema.org/dateModified",
                    "https://schema.org/contactPoint",
                    "https://schema.org/language",
                    "http://purl.org/dc/terms/rights",
                    "http://purl.org/dc/terms/creator",
                    "https://schema.org/contentUrl",
                    "https://schema.org/encodingFormat",
                    "https://schema.org/license",
                    "https://schema.org/Number"
        ]
      },
      "description": {
        "type": "string"
      },
      "parent": {
        "type": "string",
        "enum": [   "Catalog",
                    "Dataset",
                    "Distribution"
        ]
      }
    }
    },
      "additionalProperties": false
  }
}
```

Listagem 3.2: Interface de metadados para a API de Dados e modelo intermédio.

```
{
  "titulo": {
    "type": "https://schema.org/name",
    "description": "Titulo do dataset",
    "parent": "Dataset"
  },
  "descricao": {
    "type": "https://schema.org/description",
    "description": "Descricao do dataset.",
    "parent": "Dataset"
  }
}
```

Listagem 3.3: Exemplo do modelo de metadados intermédio de dois campos, título e descrição.

```
{
  "desc": {
                "type": "https://schema.org/description",
                "description": "Descrição do dataset.",
                "parent": "Dataset"
  },
  "name": {
                "type": "https://schema.org/name",
                "description": "Titulo do dataset",
                "parent": "Dataset"
  },
  "tituloC": {
                "type": "https://schema.org/name",
                "description": "Titulo do Catalogo",
                "parent": "Catalog"
  }
}
```

Listagem 3.4: Exemplo de um output da API de metadados.

Se todos os campos estiverem descritos, não será necessário a intervenção humana, sendo o mapeamento resolvido automaticamente. Caso alguns campos que não estejam devidamente descritos, esses necessitam de ser mapeados manualmente através da interface web disponibilizada ao utilizador. Todos os mapeamentos, automáticos e manuais, serão persistidos numa base de dados do sistema de mapeamento (ver Figura 3.1). Assim, para futuros mapeamentos, por exemplo por alteração pontual de algum campo, não é necessário refazer todo o mapeamento, mas apenas efectuar as alterações pretendidas.

A Figura 3.4 ilustra todos os passos do algoritmo de mapeamento, onde o primeiro passo, o **utilizador** introduz o *endpoint* da API dos dados e depois o *endpoint* da API de metadados, de seguida, o **utilizador** tem que selecionar uma das várias opções, em que, se o modelo já assume uma interface de dados a receber segue algum modelo de FIWARE ou não. Concluindo esta primeira etapa o utilizador só terá que submeter o pedido. Uma vez submetido, o **sistema** (que corresponde ao *mapping system* na Figura 3.1) vai à base de dados e verifica se já existe algum mapeamento para a API em questão:

- se SIM, é apresentado na interface do utilizador o mapeamento existente entre os campos da API de dados e os campos do modelo intermédio e neste caso o **utilizador** pode alterar, remover ou adicionar algum mapeamento entre campos que não foram mapeados automaticamente;
- se NÃO, o **sistema** compara os atributos (***parent*** e ***type***) de cada campo com os atributos (***parent*** e ***type***) de cada campo do modelo de metadados intermédio:
  - se forem iguais, o **sistema** percebe que o campo na iteração corresponde ao campo na do modelo de metadados intermédio e precede ao seu mapeamento;
  - caso não sejam iguais, compara os restantes campos pelo nome:
    - se forem iguais, faz o mesmo que foi dito anteriormente.
    - se não forem iguais, faz o mesmo que foi dito quando o sistema verifica que existe na base de dados algum mapeamento para a API em questão.

3.5 Modelo final de metadados

O Modelo final de metadados vai seguir o modelo do DCAT-AP representando as suas classes obrigatórios, concretamente o *Catalogue*, *Dataset* e *Distribution* e para cada classe

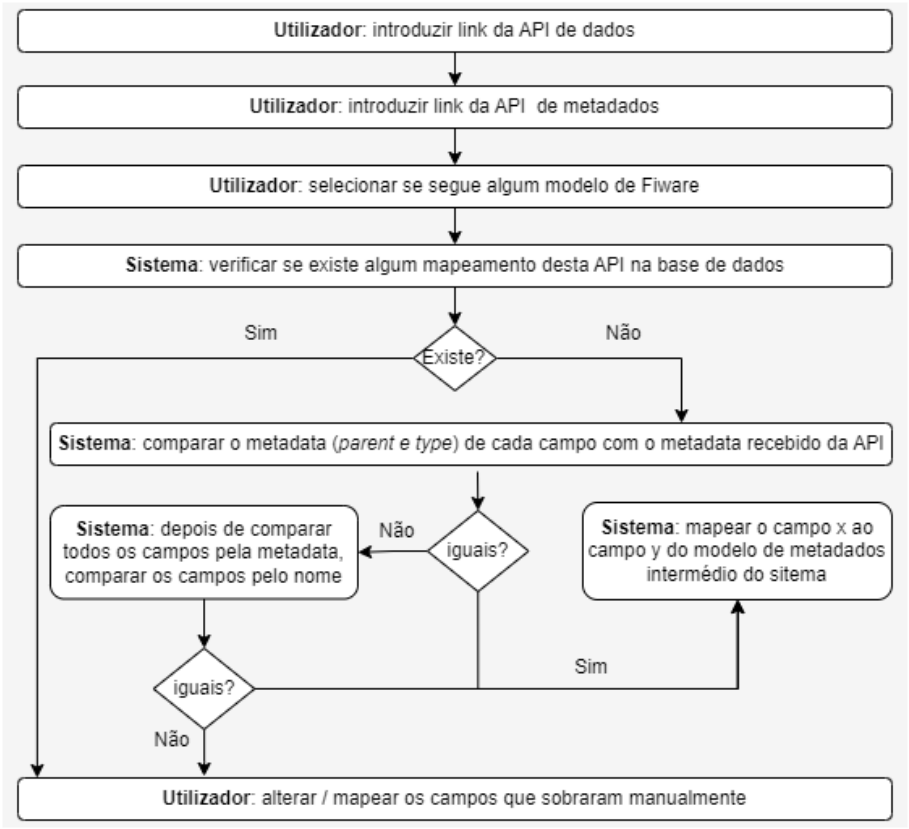

Figura 3.4: Passos do mapeamento.

os seus campos obrigatórios, como indicado na Tabela 3.1.

A necessidade deste tipo de metadata, DCAT-AP, fornece uma especificação comum para descrever conjuntos de dados do setor público na Europa para permitir a troca de descrições de conjuntos de dados entre portais de dados, ao mesmo tempo em que fornece interoperabilidade semântica com outros aplicativos com base na reutilização de mapeamentos para vocabulários de metadados existentes como *Dublin Core* e *Schema*.

Embora compatível com o modelo DCAT-AP, terá a mesma linha de campos obrigatórios, tendo algumas modificações específicas para a CML, nos quais alguns campos recomendados e opcionais do DCAT-AP, consequentemente, passaram a ser obrigatórios no Modelo final de metadados.

Especificamente, neste modelo, os campos que eram considerados como opcionais no modelo *DCAT-AP*, conforme indicado na Tabela 3.1, passaram a ser de caráter obrigatório. Portanto, todos os campos listados na Tabela 3.1 são agora de preenchimento obrigatório no Modelo final de metadados.

Sendo o objetivo da CML necessitar de identificar, integrar, analisar e eventualmente disponibilizar num repositório de dados abertos, grande parte dos dados presentes na CML e entidades externas garantindo dados corretos, não ambíguos, consistentes e

Tabela 3.1: Todos os campos do modelo final de metadados.
Prefixos:

- `dc` - Dublin Core (http://purl.org/dc/dcmitype/);
- `so` - Schema.org (https://schema.org/);
- `adms` - Asset Description Metadata Schema (https://www.w3.org/ns/adms#);
- `dcat` - Data Catalog Vocabulary (http://www.w3.org/ns/dcat#).

Classe	Atributo	Obrigatório DCAT-AP	Definição
Catalogue	`update_modification`	Não	`so:dateModified`
	`license`	Não	`so:license`
	`rights`	Não	`dc:accessRights`
	`creator`	Não	`dc:creator`
	`dataset`	Sim	`dcat:dataset`
	`description`	Sim	`dc:description`
	`publisher`	Sim	`dc:publisher`
	`Title`	Sim	`dc:title`
Dataset	`contact`	Não	`so:contactPoint`
	`keyword_tag`	Não	`so:keywords`
	`publisher`	Não	`dc:publisher`
	`access`	Não	`dc:accessRights`
	`sample`	Não	`adms:sample`
	`update_modification`	Não	`so:dateModified`
	`description`	Sim	`dc:description`
	`Title`	Sim	`dc:title`
Distribution	`description`	Não	`so:description`
	`format`	Não	`so:encodingFormat`
	`license`	Não	`so:license`
	`update_modification`	Não	`so:dateModified`
	`access_URL`	Sim	`dcat:accessURL`

completos, elevando assim a qualidade destes e ainda apoiando na solução dos desafios que esta poderá enfrentar. Entende-se por desafios, descrever e identificar o ciclo de vida dos dados, caracterizando-os e detetando problemas à priori [8].

Todos os campos do Modelo final de metadados seguem uma semântica bem definida, pois estão em conformidade com o modelo DCAP-AP. É importante destacar que o modelo DCAP-AP já adere aos princípios e à semântica do *linked data*. Portanto, ao utilizar o modelo DCAP-AP, garantimos que os campos possuem uma estrutura coerente e estão em conformidade com os padrões do *linked data*.

4

Arquitetura e a sua implementação

Neste capítulo, será abordada pormenorizadamente a arquitetura geral do sistema e as suas características fundamentais. Após uma análise cuidadosa da abordagem apresentada no capítulo anterior, este capítulo oferece uma visão aprofundada da proposta de integração e implementação do sistema.

4.1 Arquitetura da solução

A arquitetura da solução, representada na Figura 4.1, ilustra de um modo geral o que acontece dentro do sistema de mapemaneto e as partes com quem interage. Assim, a mesma é composta por:

- um utilizador (A) que interage com o *front-end* (B) do sistema que é apresentado pelo *browser*, isto é qualquer pessoa que forneça um *endpoint* da API que deseja uniformizar e que saiba realizar a sua integração ao modelo de metadados intermédio;
- "map and extract data" (C) que acede à API (F) e realiza as seguintes operações antes de mapear para o modelo de metadados intermédios:
  - verifica se esse mapeamento já existe na base de dados (D);
  - consultar o modelo do Fiware do repositório (E), caso o utilizador selecionar que a API segue algum modelo.

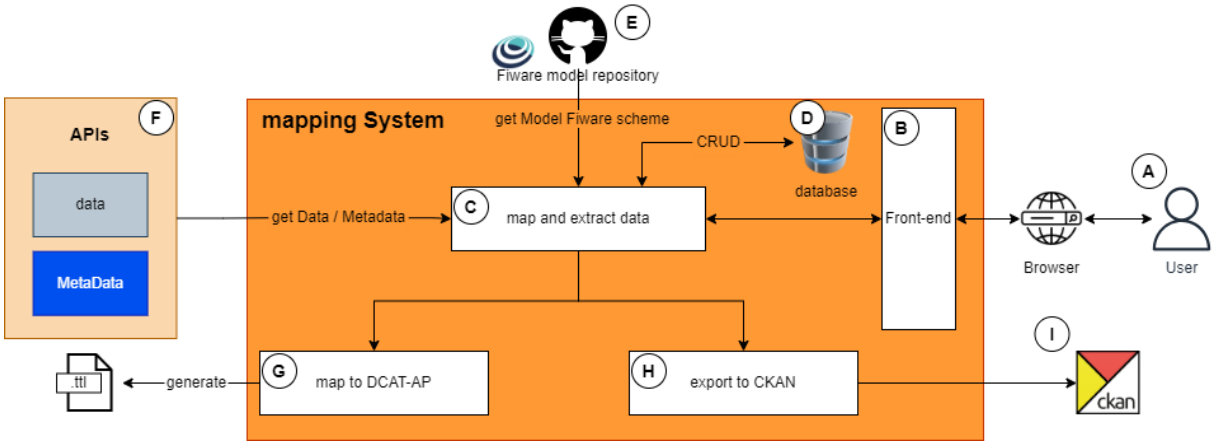

Figura 4.1: Arquitetura.

Na fase de "map and extract" o sistema apresenta na *Front-end* (B) para o utilizador (A) poder fazer alterações caso pretenda mapear manualmente campos que não foram possíveis de mapear automaticamente e por fim inserir ou atualizar o mapeamanto realizado na base de dados (D).

- "map to DCAT-AP" (G) mapeia o modelo de metadados intermédio para o modelo DCAT-AP e criar os ficheiros de *output* no formato *turtle* com a informação estandardizada, segundo o DCAT-AP para serem disponibilizados e consumidos por um utilizador final interessado. O motivo pelo qual a solução usa o formato *turtle* é por ser uma das formas standard e que tem suporte no ckan, além disso é muito fácil de ler por pessoas, pois é representado em forma de triplos;

- "export to CKAN" (H) mapeia os dados de forma a conseguir inserir os dados no CKAN (I), usando a sua API.

4.2 Implementação

No âmbito do desenvolvimento da aplicação, recorreu-se a diversas tecnologias que se revelaram cruciais para o sucesso do projeto e que serão detalhadamente apresentadas nesta secção.

Neste projeto, optou-se por utilizar Django[1] e Python[2], uma vez que o Django é um framework que oferece ferramentas para o desenvolvimento ágil de aplicações web

[1]ver: https://www.djangoproject.com/
[2]ver: https://www.python.org/

complexas. Por sua vez, a linguagem Python é amplamente reconhecida pela sua versatilidade e conta com uma vasta gama de bibliotecas e frameworks à disposição. A combinação de Python e Django proporciona diversas vantagens, incluindo facilidade de desenvolvimento, desempenho satisfatório, escalabilidade e segurança. Além disso, a escolha destas tecnologias reflete também a vontade de explorar algo novo, uma vez que nunca antes se havia desenvolvido aplicações web em Python.

A opção pelo SQLite[3] deveu-se à sua notável leveza e eficiência, considerando que o projeto não lidará com volumes substanciais de dados nem suportará um elevado número de utilizadores em simultâneo.

Quanto ao formato JSON[4], a sua adoção fundamentou-se na facilidade de leitura e escrita, bem como na sua capacidade de integração com diversas linguagens de programação e sistemas de gestão de bases de dados.

No que respeita às restantes tecnologias, nomeadamente JavaScript[5], AJAX[6] e jQuery[7], a sua incorporação no desenvolvimento web visou a criação de interatividade e dinamismo nas páginas web, contribuindo para uma experiência do utilizador mais enriquecedora.

4.2.1 Python

Python é uma linguagem de programação de alto nível, interpretada e de propósito geral, com uma ampla variedade de características que a tornam uma escolha sólida para o desenvolvimento deste projeto. A seguir, são apresentados os principais motivos que justificam a escolha do Python:

1. **Simplicidade e Clareza**: A sintaxe de Python é conhecida pela sua simplicidade e clareza. Essa característica torna o código fácil de ler e escrever, facilitando a compreensão e manutenção do projeto. A natureza legível do Python é especialmente benéfica quando várias pessoas estão envolvidas no desenvolvimento;

2. **Versatilidade**: Python é uma linguagem versátil que abrange uma ampla gama de domínios, desde desenvolvimento web até análise de dados e inteligência artificial. Isso proporciona flexibilidade para adaptar a linguagem às diferentes necessidades do projeto;

[3]ver: https://www.sqlite.org/
[4]ver: https://www.json.org/
[5]ver: https://developer.mozilla.org/pt-PT/docs/Web/JavaScript
[6]ver: https://developer.mozilla.org/pt-PT/docs/Web/Guide/AJAX
[7]ver: https://jquery.com/

3. **Eficácia no Desenvolvimento Web**: A escolha do framework Django, que é baseado em Python, facilita o desenvolvimento de aplicações web complexas. A robustez do Django oferece soluções prontas para desafios comuns no desenvolvimento web, permitindo uma implementação eficaz;

4. **Ecossistema Rico**: Python possui um ecossistema de bibliotecas e frameworks em constante crescimento, o que simplifica o acesso a recursos avançados. Além disso, a linguagem é suportada por uma vasta comunidade de desenvolvedores, garantindo atualizações regulares e suporte técnico;

5. **Interpretação e Facilidade de Depuração**: Python é uma linguagem interpretada, o que significa que o código é executado diretamente, sem a necessidade de compilação. Isso acelera o desenvolvimento e a depuração, uma vez que os erros podem ser identificados e corrigidos mais rapidamente;

6. **Tipagem Dinâmica**: Python possui tipagem dinâmica, o que significa que o tipo de variável é determinado em tempo de execução. Essa flexibilidade facilita a escrita de código e a manipulação de dados sem a necessidade de declarações rígidas de tipo;

7. **Programação Orientada a Objetos**: Python suporta programação orientada a objetos, o que permite a criação de código modular e reutilizável. Isso promove uma estrutura organizada e facilita a manutenção do projeto;

8. **Grande Biblioteca Padrão**: Python é acompanhado por uma ampla biblioteca padrão que abrange diversas áreas, desde manipulação de ficheiros até acesso a bases de dados. Isso reduz a necessidade de desenvolver funcionalidades a partir do zero, economizando tempo e recursos.

Em resumo, a escolha do Python para este projeto foi baseada em sua simplicidade, versatilidade, eficácia no desenvolvimento web, ecossistema robusto, facilidade de depuração, tipagem dinâmica, suporte à programação orientada a objetos e uma vasta biblioteca padrão. Essas características combinadas proporcionaram uma base sólida para o desenvolvimento bem-sucedido da aplicação.

4.2.2 Framework Django

O Django é um framework web de alto nível e código aberto, escrito em Python. Foi criado para tornar a criação de aplicações web mais rápida e fácil, com um foco em segurança e escalabilidade.

Uma das principais características do Django é o seu padrão Model-View-Controller (MVC) para desenvolvimento web. No Django, esse padrão é chamado de Model-View-Template (MVT) [47], onde:

- Model: representa a camada de dados da aplicação, que define como os dados serão armazenados e manipulados;
- View: é responsável por processar as requisições HTTP e retornar as respostas adequadas;
- Template: define a aparência visual da página, utilizando HTML, CSS e JavaScript.

O Django também possui outras características importantes, que o tornam uma escolha adequada para este projeto:

- Administração: O Django vem com um sistema de administração que permite gerenciar facilmente os modelos de base de dados através de uma interface web;
- Sistema de URLs: o Django usa um sistema de URLs para mapear as requisições HTTP aos recursos corretos;
- ORM (Object-Relational Mapping): O Django vem com um ORM que permite interagir com a base de dados usando objetos Python. Isso torna a escrita de código mais produtiva e elimina a necessidade de escrever consultas SQL diretamente;
- Segurança: o Django possui diversas camadas de segurança para proteger a aplicação contra vulnerabilidades, como proteção contra injeção de SQL, cross-site scripting (XSS) e falsificação de solicitação entre sites (CSRF);
- Escalabilidade: O Django é altamente escalável e pode lidar com um grande volume de tráfego e solicitações simultâneas;
- Adequação à Solução do Projeto:

  O Django é amplamente considerado adequado para a solução do problema deste projeto por diversas razões. Primeiramente, sua estrutura de desenvolvimento baseada no padrão Model-View-Template (MVT) oferece uma separação clara de responsabilidades, o que facilita o desenvolvimento de aplicações web complexas. O componente "Model"permite a definição e manipulação eficiente dos dados, enquanto o "View"lida com o processamento das requisições e respostas

HTTP. O "Template"permite a criação de interfaces de usuário dinâmicas e atraentes, contribuindo para a experiência do utilizador.

Além disso, o Django oferece um sistema de administração incorporado, tornando a gestão de modelos de base de dados mais eficiente através de uma interface web amigável. Isso é particularmente útil para a criação e manutenção de metadados de APIs e facilita o controlo e a gestão dos dados do aplicativo.

O sistema de URLs do Django ajuda a mapear as requisições HTTP para as funcionalidades apropriadas, simplificando a navegação e o acesso aos recursos da aplicação. O uso do Object-Relational Mapping (ORM) permite interagir com a base de dados de maneira mais produtiva, eliminando a necessidade de escrever consultas SQL complexas.

Em termos de segurança, o Django é conhecido por suas camadas de proteção robustas, o que é crucial para proteger a aplicação contra vulnerabilidades comuns na web. A sua natureza altamente escalável garante que a aplicação possa crescer e lidar com um grande volume de tráfego e múltiplas requisições simultâneas, atendendo às necessidades do projeto.

O Django é amplamente utilizado por muitas empresas, como Instagram, Pinterest e Mozilla. Com a crescente popularidade da linguagem Python, o Django se tornou uma escolha popular para o desenvolvimento de aplicações web, oferecendo facilidade de desenvolvimento, segurança e uma ampla gama de recursos para atender às necessidades deste projeto.

4.2.3 SQLite

SQLite é uma biblioteca de software livre que oferece um mecanismo de base de dados SQL embutido. Foi projetado para ser rápido, leve e de fácil utilização, tornando-se uma escolha popular para aplicações que necessitam de armazenamento de dados em dispositivos com recursos limitados ou que exigem uma base de dados local.

De acordo com Ferguson e Humphries [18], o SQLite é uma excelente escolha para aplicações com baixo tráfego ou que requerem acesso rápido a dados armazenados localmente. Pode ser facilmente integrado em aplicações móveis, bem como em aplicações desktop e web.

Uma das principais vantagens do SQLite é que se trata de uma base de dados autónoma, o que significa que não requer um servidor separado para funcionar. Além disso, suporta funcionalidades avançadas, como transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade) e chaves estrangeiras.

4.2.4 JavaScript

JavaScript é uma linguagem de programação utilizada principalmente para criação de aplicações web. Criada em 1995 por Brendan Eich, é uma linguagem de script que permite adicionar interatividade e dinamismo a páginas web, através da manipulação do Document Object Model (DOM) e do Cascading Style Sheets (CSS). O JavaScript é uma linguagem de programação orientada a objetos, com sintaxe similar a linguagens como C e Java, e é executada diretamente nos navegadores web. A linguagem é projetada para ser simples e fácil de aprender, mas é poderosa o suficiente para lidar com aplicações complexas [21]. Além da manipulação do DOM e CSS, o JavaScript também pode ser utilizado para fazer requisições assíncronas, gerenciamento de cookies e armazenamento de dados em bancos de dados locais, por exemplo. Com a evolução do JavaScript, surgiram diversas bibliotecas e frameworks, como o React, Angular e Vue, que facilitam o desenvolvimento de aplicações web mais complexas.

4.2.5 JQuery

A tecnologia JQuery é uma biblioteca de JavaScript que simplifica a manipulação de elementos HTML, a gestão de eventos e a animação na web. Foi criada em 2006 por John Resig e é utilizada em diversos projetos web, desde sites institucionais até aplicações mais complexas [3].

JQuery é uma tecnologia popular devido à sua simplicidade e facilidade de uso, bem como sua compatibilidade com diversos navegadores web.

Segundo a documentação oficial do JQuery, suas principais características são:

- Seleção de elementos simplificada;
- Manipulação de elementos HTML e CSS;
- gestão de eventos de forma intuitiva;
- Animação com suporte a efeitos personalizados;
- Requisições AJAX simplificadas.

4.2.6 AJAX

A tecnologia AJAX (Asynchronous JavaScript and XML) é uma técnica de desenvolvimento web que permite atualizações assíncronas de conteúdo, ou seja, sem a necessidade de recarregar a página inteira. Essa técnica é utilizada para melhorar a experiência do utilizador, tornando as aplicações mais rápidas e dinâmicas.

AJAX é uma combinação de diversas tecnologias, como JavaScript, XML, DOM e CSS, que juntas permitem o desenvolvimento de aplicações web mais interativas e responsivas. Através do uso de requisições assíncronas ao servidor, o conteúdo é atualizado sem que a página inteira precise ser recarregada, o que resulta em uma experiência mais fluida e rápida para o utilizador [23].

4.2.7 JSON

JSON (JavaScript Object Notation) é um formato de troca de dados que se tornou muito popular em aplicações web e APIs RESTful. Foi criado por Douglas Crockford em 2001 e é baseado em um subconjunto da linguagem JavaScript.

JSON é um formato leve, fácil de ler e escrever, e é amplamente utilizado em aplicações web para transferir dados entre o servidor e o cliente. É representado por uma estrutura de dados simples composta por pares chave-valor, que pode incluir outros objetos ou arrays [24].

4.3 Desenvolvimento da aplicação

Nesta secção serão detalhados os aspetos mais relevantes relacionados com as funcionalidades da aplicarão implementadas.

4.3.1 Estrutura do Projeto

Na criação de um projeto, o Django cria, por defeito, um conjunto de pastas e ficheiros cuja estrutura está apresentada na Figura 4.2

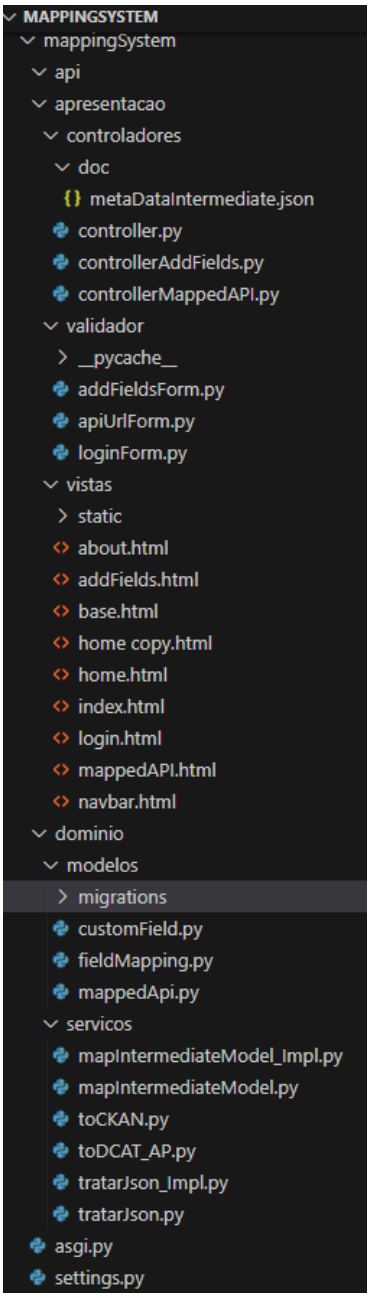

Figura 4.2: Estrutura de pastas e ficheiro.

Na pasta *mappingSystem* encontram-se os ficheiros de configuração:

- *settings.py* ficheiro de configuração de instalação do Django;
- *urls.py* ficheiro de configuração de URLs, ficheiro responsável pelo mapeamento de cada URL para a *view* correspondente;
- *wsgi.py* ficheiro de configuração do *Web Server Gateway Interface* (WSGI), padrão utilizado para desenvolver aplicações Web em linguagem Python.

Na pasta *mappingSystem* encontram-se os ficheiros associados à aplicação desenvolvida que serão explicados ao longo deste capítulo. A pasta vistas contém os templates do projeto e os ficheiros de css e javascript. Os templates são normalmente ficheiros em *HTML* que têm como objetivo separar a apresentação dos dados.

O projeto seguem uma abordagem semelhante ao MVC. Esta organização, Figura 4.3, facilita a separação de responsabilidades, a reutilização de código, os testes e a escalabilidade do projeto. A pasta "controladores' ajuda a manter a lógica de processamento de requisições HTTP isolada, tornando o código mais organizado e fácil de entender. Além disso, essa estrutura padronizada melhora a colaboração com outros desenvolvedores e simplifica a manutenção do software.

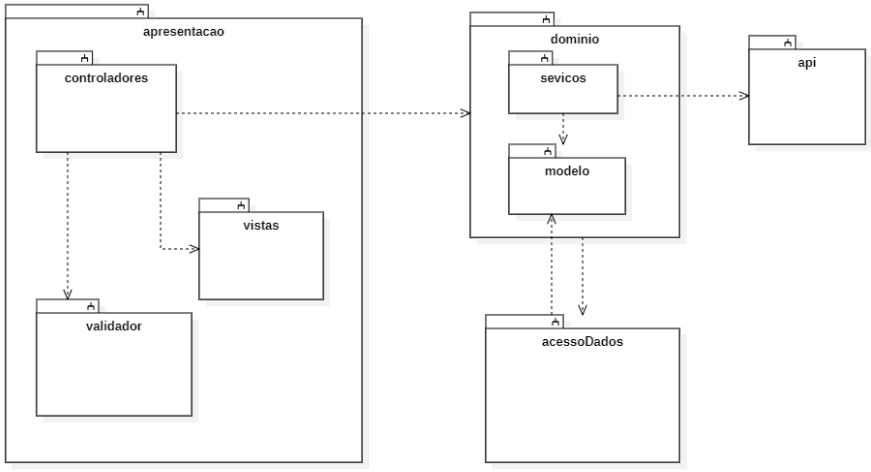

Figura 4.3: Estrutura global da aplicacção (organização geral).

4.3.2 Base dados

Com base nos requisitos do sistema, tornou-se necessário utilizar uma base de dados para armazenar todos os dados necessários. Nesse sentido, optou-se por utilizar a base de dados padrão do Django, que já possui as tabelas essenciais para a gestão de utilizadores, autenticação e segurança.

Adicionalmente, um novo modelo foi criado e associado a uma nova tabela, em que está relacionado com a tabela de utilizadores do Django. Esta abordagem permitiu aproveitar as funcionalidades avançadas já existentes na*framework*. Para a implementação da base de dados, foi utilizada a tecnologia SQLite3.

4.3.2.1 Ligação do Django com a base de dados SQLite3

Django permite a ligação à base de dados SQLite3 através da seguinte configuração no ficheiro settings.py, representado na Listagem 4.1.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3')
    }
}
```

Listagem 4.1: Exemplo de configuração que permite a ligação à base de dados SQLite3.

4.3.2.2 Migração da Base de Dados

A migração da base de dados é uma etapa crucial na manutenção da aplicação, pois além de manter a mesma atualizada, também garante a segurança dos dados, evitando perda de informações em caso de falhas humanas ou no sistema.

A migração consiste em aplicar alterações no esquema da base de dados, como a criação de novas tabelas ou alteração de campos existentes. Ao aplicar as migrações, o Django gera um histórico das alterações, permitindo a reversão caso necessário. Portanto, é fundamental realizar as migrações regularmente para garantir o bom funcionamento da aplicação e a integridade dos dados armazenados.

O sistema de migração do Django permite trabalhar com um grande número de migrações. Os comandos utilizados que permitiram fazer as migrações à base de dados foram:

```
    python manage.py makemigrations
    python manage.py migrate
```

Listagem 4.2: Comandos que permitem realizar as migrações à base de dados.

4.3.3 *Model*

Com base no requisitos do sistema, foi necessário criar um modelo de dados que permite o mapeamento de vários campos de recursos. Para isso, foi criado um modelo com três tabelas *FieldMapping*, *MappedApi*, *CustomField*, que é definido no ficheiro *fieldMapping.py*, *mappedApi.py* e *customField.py*.

```
1
2 class FieldMapping(models.Model):
3     user = models.ForeignKey(User, on_delete=models.CASCADE)
4     my_json_object = models.TextField()
5     ...
6     data_api_link = models.URLField()
7     metadata_api_link = models.URLField()
8     custom_fields = models.ManyToManyField(CustomField)
```

Listagem 4.3: Trecho de código do modelo da tabela FieldMapping.

Este Modelo contém a informação sobre os campos e os seus comportamentos, permitindo a definição de regras de validação, restrições de integridade e outros comportamentos específicos.

Cada campo do *fieldMapping* representa um campo da tabela do modelo, e cada tabela é representada por uma classe *Python* que estende a subclasse *django.db.models.Model*. Essa abordagem torna o sistema mais flexível e escalável, possibilitando a inclusão de novos campos e modelos sem a necessidade de alterações significativas na estrutura da base de dados. A mesma Tabela tem relação de vários para vários com a tabela *CustomField* o que vai resultar numa tabela, gerida pelo *django*, para poder guardar os campos adicionei criados pelo utilizador para serem mapeados. Tem também uma relação de um para vários com a tabela *MappedApi*, para poder guardar as varias alterações nos resultados do mapemaento de cada API.

O trecho de código mostrado na Listagem 4.3 mostra como as tabelas de base de dados são mapeadas. Nesse caso, o mapeamento é para a tabela *FieldMapping*, tabela destinada a armazenar informações sobre o mapeamento de campos das APIs.

Os campos **user** e *my_json_object* são exemplos de campos de propriedade do modelo. O campo **user** é uma chave estrangeira relativa à tabela de utilizadores do Django.

Cada campo é especificado como um atributo de classe e cada atributo é mapeado para uma coluna na tabela *FieldMapping*.

4.3.4 Funcionalidades Implementadas

Nesta secção será apresento todas as funcionalidades implementadas no sistema desenvolvido. Cada funcionalidade será descrita em detalhes. Também serão apresentados telas de amostra e trechos de código que demonstram cada funcionalidade.

4.3.4.1 Autenticação

Foi utilizado o *Admin* do *Django*, para gerar uma conta de administrador, com essa conta pode-se autenticar na aplicação e também para registar outros utilizadores. Com a finalidade de aceder a aplicação, o utilizador deve informar o seu nome de utilizador e senha. Caso algum dos campos referidos for introduzido incorretamente, o acesso á aplicação será negado e apresentará uma mensagem de erro ao utilizador. Na Figura 4.4, pode ser vista a interface de autenticação utilizada pelo utilizador.

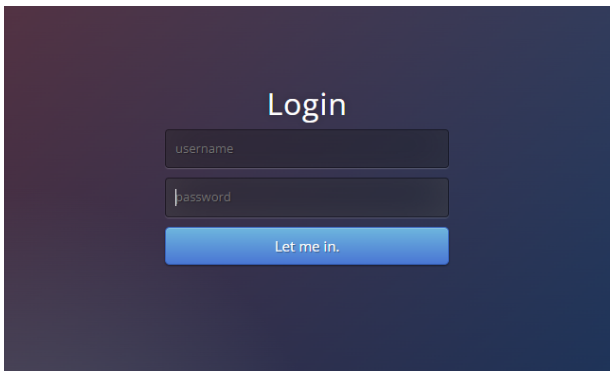

Figura 4.4: Interface de autenticação.

O código responsável por usar o autenticador do *django* é o apresentado na na Listagem 4.4 que tem como formulário que estende **AuthenticationForm** do django e adiciona dois campos personalizados com *placeholder*.

```
1 from django import forms
2 from django.contrib.auth.forms import AuthenticationForm, UsernameField
3
4 class UserLoginForm(AuthenticationForm):
5     def __init__(self, *args, **kwargs):
6         super(UserLoginForm, self).__init__(*args, **kwargs)
7
8     username = UsernameField(widget=forms.TextInput(attrs={"placeholder":"
    username"}))
9     password = forms.CharField(widget=forms.PasswordInput(attrs={"
    placeholder":"password"}))
```

Listagem 4.4: Formulário de autenticação do django, (ficheiro loginForm.py).

Uma vez o utilizador autenticado com sucesso será redirecionado para a página de mapeamento representada na Figura 4.5, onde deve introduzir o url da API e o seu url de metadados que deseja estandardizar e selecionar se segue algum modelo de *Fiware*, se for introduzido em algum campo dados inválidos será despoletada uma mensagem de erro, esses campos são validados pelo seguinte código na Listagem 4.5.

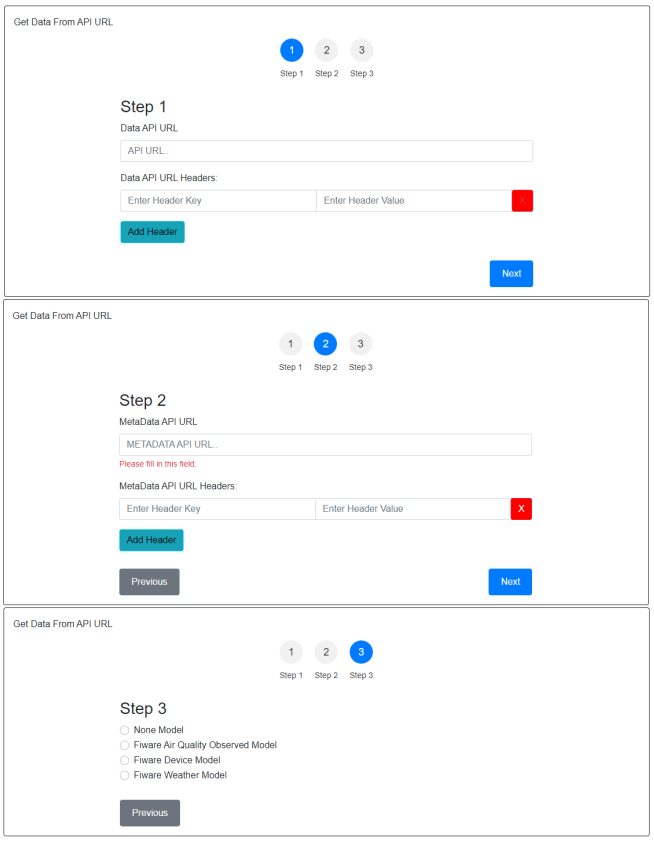

Figura 4.5: UI de mapeamento de APIs.

```
class apiUrlForm(forms.Form):
    url = forms.CharField(required=True, label="url", widget=forms.TextInput(attrs={"placeholder":"API URL..", "class":"form-control"}))
    urlMetaData = forms.CharField(required=True, label="urlMetaData", widget=forms.TextInput(attrs={"placeholder":"METADATA API URL..", "class":"form-control"}))

    def clean(self):
        cleaned_data=super().clean()
        url=cleaned_data.get('url')
        urlMetaData=cleaned_data.get('urlMetaData')

        regex=r'https?://(?:[-\w.]|(?:%[\da-fA-F]{2}))+'
        match=re.search(regex, url)
        if not match:
            raise forms.ValidationError({"url": "Enter a valid url starting with http:// or https://"})

...

        if response_metaData.status_code != 200:
            raise forms.ValidationError({"urlMetaData":"Code Error: "+str(response_metaData.status_code)+" » "+response_metaData.reason })

        if not self.is_valid():
            return cleaned_data

        return {"responseAPI": response_API.json(), "response_metaData": response_metaData.json()}
```

Listagem 4.5: Validação do Formulário da API a normalizar (ficheiro apiUrlFrom.py).

4.3.4.2 Campos a mapear da API

Para obter todos os campos e dados do formulário, o protocolo HTTP é usado para acessar os URLs fornecidos após o envio do formulário, conforme ilustrado na Figura 4.5. Em seguida, um algoritmo é executado para extrair os nomes de todos os campos cujos valores foram obtidos, no código da Listagem 4.6 encontra-se a função responsável pelo armazenamento do nome dos campos. Após isso, esses campos são apresentados na interface da API de dados e estão preparados para serem mapeados aos campos DCAT-AP previamente configurados.

Posteriormente, um segundo algoritmo é acionado para realizar o mapeamento automático entre os campos da API e os campos predefinidos do DCAT-AP, o trecho de código na Listagem 4.7 prepara uma lista de objetos de origem e destino de campos já associados uns aos outros para representa-lhos na interface como mapeados. O resultado desse mapeamento é mostrado na interface, como ilustrado na Figura 4.6. O utilizador pode editar esse mapeamento caso deseje.

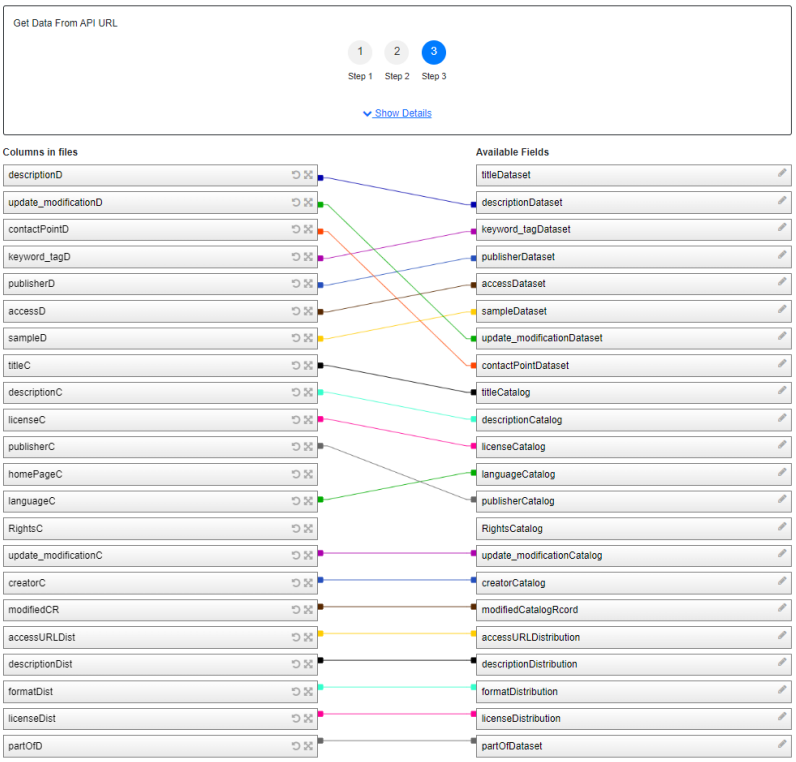

Figura 4.6: Interface de mapeamento de APIs.

Por fim ao submeter o mapeamento e antes de guardar na base de dados é executado o ultimo algoritmo presente no trecho de código da Listagem 4.8 que vais buscar os valores de cada campo mapeado e converter para o formato DCAT-AP e gera um output

```
def makeIntermediateModelFields(self, schemeFiwareModel, schemeMetaDataIntermediate):

        intermediateModelFields = list(schemeMetaDataIntermediate.keys())
        properties = self.findProperties(schemeFiwareModel)
        if properties is not None:
            intermediateModelFields = list(
                properties.keys()) + intermediateModelFields

        return intermediateModelFields
```

Listagem 4.6: Extrair os campos que vao ser mapeados e apresentados na interface, (ficheiro mapIntermediateModel_Impl).

```
def getMappedLinks(self, intermediateModelFields, schemeMetaDataIntermediate, APImetaData, data): # mapTo
        existingLinks = []
        element = {}

        for i in intermediateModelFields:
            if i in schemeMetaDataIntermediate:
                element = schemeMetaDataIntermediate[i]
                if isinstance(element, list):  # []
                    element = element[0]
                if not element:  # empty
                    continue
            for x in data:
                if x not in APImetaData:  # empty
                    if x == i:
                        existingLinks.append({"from": x, "to": i})
                    continue
                if i not in schemeMetaDataIntermediate:  # nao existe
                    continue
                metadata = APImetaData[x]

                if isinstance(metadata, list):  # []
                    metadata = metadata[0]
                if metadata['type'] == element['type'] and metadata['parent'] == element['parent']:
                    existingLinks.append({"from": x, "to": i})
        return existingLinks
```

Listagem 4.7: Uma das funções do algoritmo que faz o mapeamento automático (ficheiro mapIntermediateModel_Impl.py).

no formato Json-LD e Turtle, só depois então é que é guardado na base de dados todos os mapeamentos e os seus resultados para depois apresentar noutra interface.

4.3.4.3 Campos adicionais

Se o utilizador desejar adicionar campos adicionais aos campos predefinidos do DCAT-AP, há uma opção na interface de mapeamento ilustrada na Figura 4.6. Nessa opção, o utilizador pode adicionar campos personalizados, como ilustrado na Figura 4.7.

Primeiro, o utilizador precisa adicioná-los separadamente na secção *Add Fields*. Nessa área, é possível inserir vários campos e as respetivas informações, como metadados. Ao selecionar o *namespace* para o campo, é feito um acesso via protocolo HTTP para carregar todos os dados desse *namespace*. Em seguida, apenas os termos são extraídos e apresentados em uma lista para que o utilizador possa selecioná-los.

Por fim, o utilizador pode escolher se o campo será parte de um conjunto de dados, um catálogo ou uma distribuição do DCAT-AP, como ilustrado na Figura 4.8.

```
1
2  import json
3  from rdflib import Graph, Namespace, Literal, URIRef
4  from rdflib.namespace import RDF, XSD
5
6  def convert\_to\_dcat\_ap(json_data):
7      # Criacao do grafo RDF
8      dcat = Namespace("http://www.w3.org/ns/dcat#")
9      dcterms = Namespace("http://purl.org/dc/terms/")
10     foaf = Namespace("http://xmlns.com/foaf/0.1/")
11
12     g = Graph()
13     g.bind("dcat", dcat)
14     g.bind("dcterms", dcterms)
15     g.bind("foaf", foaf)
16
17     # Conversao para DCAT-AP
18     catalog\_uri = URIRef("http://example.org/catalog")
19
20     dataset\_uris = []
21 # tentar fazer esta parte automaticamente
22     # Adicionando informacoes do catalogo
23     catalo\g_node = URIRef(catalog\_uri)
24     g.add((catalog\_node, RDF.type, dcat.Catalog))
25     g.add((catalog\_node, dcterms.title, Literal(
26         json_data["catalog"]["title"])))
27     g.add((catalog\_node, dcterms.description, Literal(
28         json\_data["catalog"]["description"])))
29 ...
```

Listagem 4.8: Uma das funções do algoritmo que faz o mapeamento automático (ficheiro ToDCAT_AP.py).

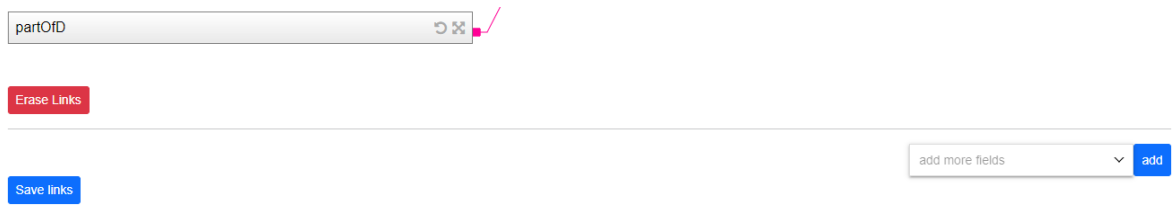

Figura 4.7: Adicionar campo novo.

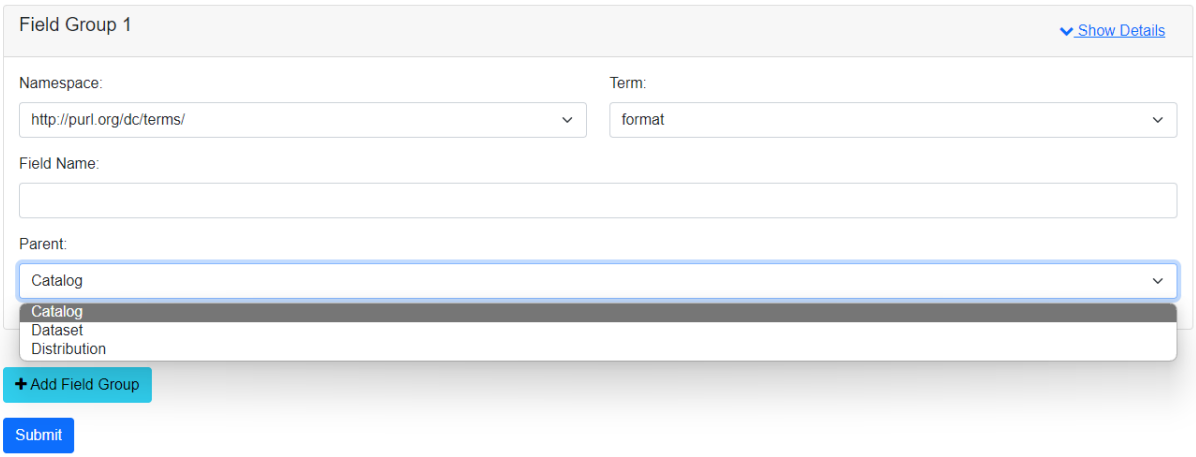

Figura 4.8: Criar campo novo.

4.3.4.4 Gestão de Resultados de Mapeamentos

Nesta subseção, vai ser explorado as diferentes funcionalidades que permitem ao utilizador aceder, editar, copiar, descarregar, importar dados ao CKAN e eliminar os resultados de mapeamentos, na Figura 4.9. Estas funcionalidades foram desenvolvidas para dar ao utilizador flexibilidade e controlo , facilitando o uso de mapeamentos anteriores nas suas atividades.

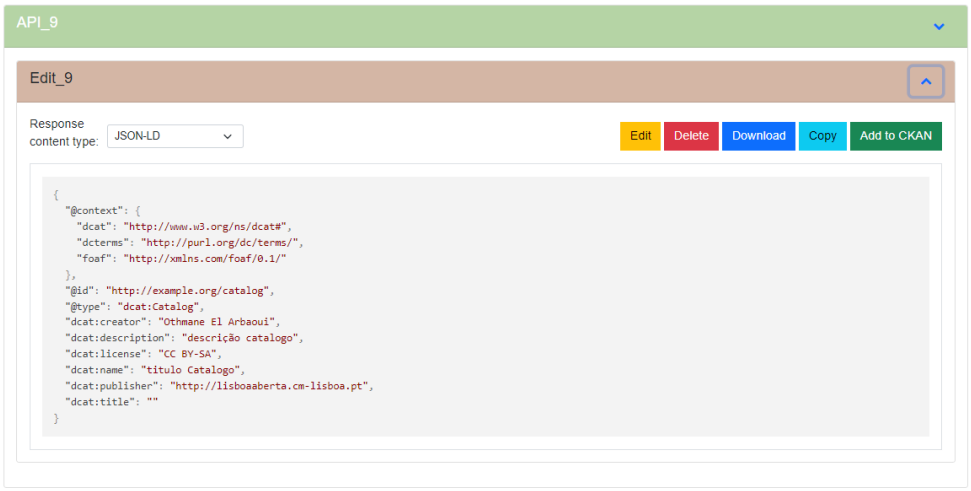

Figura 4.9: Gerir o resultado.

- **Consulta de Resultado:** Um dos pilares da aplicação é a capacidade de aceder facilmente a dados históricos. Com a funcionalidade de consulta de resultados, os utilizadores podem: Visualizar todos os mapeamentos anteriores numa interface organizada e intuitiva; Aceder a informações detalhadas sobre cada mapeamento, incluindo metadados relevante tendo uma opção de poder escolher o formato dos dados JSON-LD ou *turtle*;

- **Edição de Mapeamentos:** É disponibilizado ao utilizador a capacidade de fazer ajustes em mapeamentos existentes o que vai sempre gerar uma resultado diferente. Com esta funcionalidade: O utilizador pode consultar todas as edições por API, ou seja, todos os diferentes resultados, atualizar informações e melhorar o conteúdo de mapeamentos anteriores;

- **Cópiar e descarregar resultado:** É disponibilizado ao utilizador um botão para copiar o resultado e outro botão para descarregar o resultado no formato JSON-LD ou *turtle*. Isso oferece flexibilidade para usar os dados em outras ferramentas ou partilhá-los com colegas;

- **Eliminar Resultados:** Para garantir a organização e a limpeza da aplicação, é permitido ao utilizador eliminar resultados de mapeamentos antigos que já não são necessários;

- **Adicionar ao CKAN:** É disponibilizado ao utilizador um botão que permite aos utilizador adicionar automaticamente os dados ao CKAN. Ao clicar no botão '*add to CKAN*, surgirá uma pequena caixa de diálogo no ecrã, como na Figura 4.10. Este diálogo solicitará ao utilizador o URL do CKAN e o *token* de autenticação. Após fornecer essas informações, basta clicar em *ADD* e os dados serão importados automaticamente para o portal CKAN.

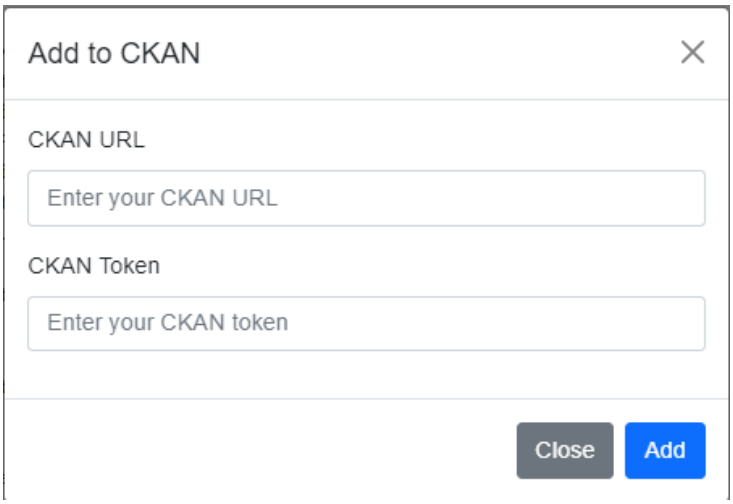

Figura 4.10: Caixa de diálogo para inserir os dados de acesso ao CKAN.

5

Avaliação da Conversão de Metadados para o DCAT-AP

Neste capítulo, vai ser abordada a avaliação da conversão de dados para o DCAT-AP. A conformidade com o DCAT-AP é crucial para garantir a interoperabilidade e a consistência dos dados convertidos. Para avaliar essa conformidade, vão ser exploradas as métricas que foram utilizadas para medir essa conformidade, analisando tanto a validação DCAT-AP como o tempo de processamento. Estas métricas desempenham um papel fundamental na garantia de que os dados estão em conformidade com os padrões estabelecidos e que a aplicação de conversão opera de forma eficaz e eficiente. Vai ser agora aprofundado o estudo dessas métricas e os resultados da avaliação.

5.1 Métrica de Validação DCAT-AP

A validação da conformidade da API com o perfil DCAT-AP é uma etapa crítica para garantir que os metadados gerados estejam em conformidade com os padrões estabelecidos. Para essa validação, é utilizada a ferramenta DCAT-AP *Validator*[1], que verifica se os metadados gerados seguem as *Controlled Vocabularies* da versão 2.1.1 do DCAT-AP.

A validação é realizada a partir da saída JSON-LD gerada pela aplicação, que é posteriormente submetida ao DCAT-AP Validator. A ferramenta DCAT-AP *Validator* analisa

[1]ver: https://www.itb.ec.europa.eu/shacl/dcat-ap/upload, acedido em 11/10/2023

os metadados e verifica se os termos e valores utilizados estão em conformidade com os padrões estabelecidos pelo DCAT-AP.

Para demonstrar o resultado da validação, a Figura 5.1 apresenta uma captura de ecrã do DCAT-AP *Validator*. Como pode ser observado, o resultado foi positivo, indicando que a API atende aos padrões DCAT-AP 2.1.1 *Controlled Vocabularies*.

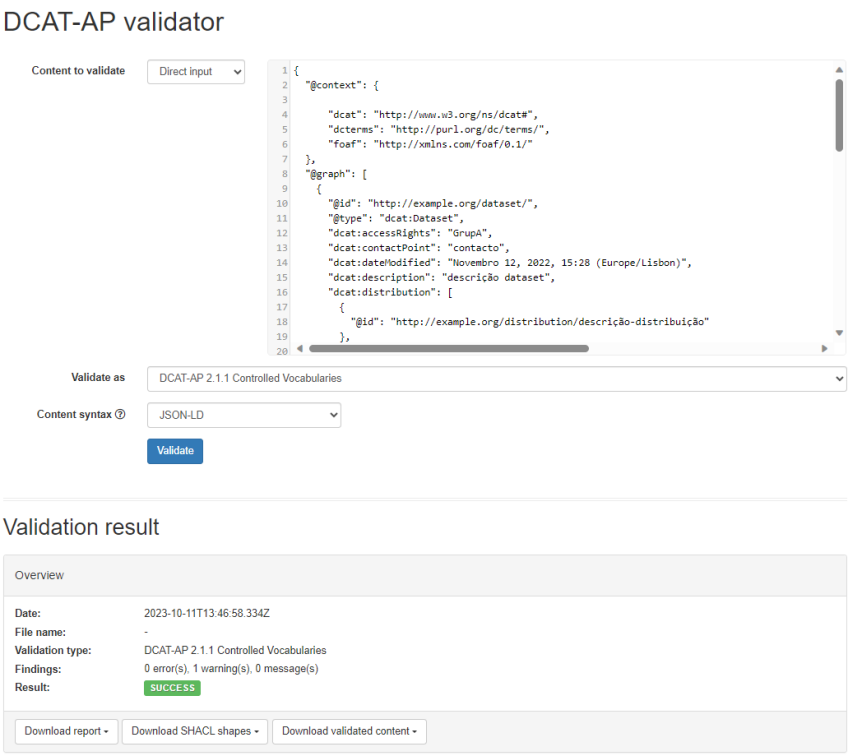

Figura 5.1: Resultado da validação da API pelo DCAT-AP Validator.

Esta métrica é essencial para garantir a interoperabilidade da API com outros sistemas e para cumprir as diretrizes de dados abertos estabelecidas pelo DCAT-AP.

5.1.1 Resultados da Validação

A validação do resultado da aplicação pelo DCAT-AP *Validator* resultou em 0 erro(s), 1 aviso(s) e 0 mensagem(s), indicando que os metadados estão em conformidade com as *Controlled Vocabularies* do DCAT-AP. Este resultado comprova que a API segue os padrões estabelecidos e está pronta para integrar com outros sistemas e portais de dados abertos.

5.2 Tempo de Processamento

A métrica de tempo de processamento é um fator crítico a ser considerado na avaliação da conversão de dados para o padrão DCAT-AP. O tempo necessário para a aplicação converter repetidamente uma API não uniformizada em um formato DCAT-AP é um indicador importante do desempenho da aplicação e pode afetar a sua usabilidade em cenários reais.

5.2.1 Metodologia de Medição

Para medir o tempo de processamento, foram realizados testes controlados nos quais a mesma API de dados não uniformizados foi convertida repetidamente para o padrão DCAT-AP. O tempo necessário para concluir cada conversão foi registado e analisado.

5.2.2 Resultados

Os resultados dos testes repetidos para a mesma API estão apresentados na tabela abaixo:

Número de Conversões	**Tempo Médio de Processamento (segundos)**
1	3,39
2	3,42
3	3,41

Tabela 5.1: Tempo de processamento repetido para a mesma API.

Os resultados indicam que o tempo de processamento é consistente para a mesma API, com pequenas variações. Em média, a aplicação demonstrou um desempenho eficiente ao converter a API para o formato DCAT-AP.

Além disso, é importante entender as etapas envolvidas no processo de conversão, que podem contribuir para o tempo de processamento. As principais etapas incluem:

- **Solicitação de Dados das APIs:** O processo começa com a solicitação de dados das APIs de dados e metadados. Esta fase requer a recuperação de informações pertinentes de ambas as fontes para posterior conversão;
- **Mapeamento Automático:** Uma vez que os dados são coletados, a aplicação executa um mapeamento automático para conectar os metadados apropriados aos

campos de dados apropriados. Isso é feito automaticamente e em segundo plano, garantindo que os dados sejam complementados com metadados pertinentes;

- **Conversão para DCAT-AP:** A aplicação usa os metadados mapeados com sucesso para converter os dados para o formato DCAT-AP, usando JSON-LD e Turtle. É fundamental realizar esse processo de conversão para garantir que os dados estejam em conformidade com o padrão DCAT-AP.

5.3 Justificação da Escolha das Métricas

A seleção das métricas usadas para avaliar a conversão de dados para o DCAT-AP desempenha um papel essencial na nossa investigação. Nesta secção, iremos explicar a razão pela qual optámos pelas métricas de validação DCAT-AP e tempo de processamento.

5.3.1 Métrica de Validação DCAT-AP

A validação da conformidade com o perfil DCAT-AP é fundamental para garantir que os metadados gerados estejam em estrita conformidade com os padrões estabelecidos. A escolha desta métrica baseia-se em vários motivos:

- **Padrões e Interoperabilidade:** O DCAT-AP é um padrão amplamente reconhecido para metadados de dados abertos na Europa. Validar a conformidade com esse padrão assegura que os dados gerados sejam interoperáveis com outros sistemas e estejam alinhados com as diretrizes de dados abertos;
- **Qualidade e Precisão:** A métrica de validação DCAT-AP permite avaliar a qualidade e a precisão dos metadados gerados. Isso é essencial para garantir que os metadados cumpram os requisitos rigorosos estabelecidos pelo DCAT-AP;
- **Transparência e Confiança:** A validação proporciona transparência e confiança aos utilizadores finais, uma vez que podem ter a certeza de que os metadados atendem a padrões reconhecidos e respeitados.

5.3.2 Métrica de Tempo de Processamento

A métrica de tempo de processamento é vital para a avaliação do desempenho da aplicação de conversão de dados para o DCAT-AP. A escolha dessa métrica é justificada pelos seguintes motivos:

- **Usabilidade e Eficiência:** O tempo de processamento afeta diretamente a usabilidade da aplicação. Um tempo de processamento mais rápido melhora a eficiência da aplicação, tornando-a mais atrativa para os utilizadores;
- **Dimensionamento e Escalabilidade:** Compreender o tempo necessário para converter dados é fundamental para dimensionar e dimensionar a infraestrutura, especialmente em cenários de uso intensivo;
- **Identificação de Ineficiências:** A medição do tempo de processamento permite identificar ineficiências na aplicação e áreas que podem ser otimizadas. Isso é crucial para o aprimoramento contínuo da aplicação.

Em resumo, as métricas de validação DCAT-AP e tempo de processamento foram escolhidas para garantir a conformidade, qualidade e eficiência na conversão de dados para o DCAT-AP. Essas métricas permitem uma avaliação abrangente do desempenho e da qualidade dos dados gerados pela aplicação.

6

Conclusão

A conversão de dados para o padrão DCAT-AP é uma tarefa crucial para garantir a interoperabilidade e a consistência dos dados abertos na União Europeia. Neste trabalho, desenvolvemos uma aplicação que automatiza o processo de conversão de APIs de dados não uniformizados para o formato DCAT-AP. Esta aplicação foi submetida a uma avaliação rigorosa para verificar a sua conformidade com o DCAT-AP e o seu desempenho em termos de tempo de processamento.

A métrica de validação DCAT-AP revelou que a aplicação cumpre os padrões do DCAT-AP, apresentando 0 erros, 1 aviso e 0 mensagens. Isto demonstra a conformidade da aplicação com as Controlled Vocabularies do DCAT-AP e a sua capacidade para gerar metadados de elevada qualidade.

A métrica de tempo de processamento revelou que a aplicação é eficiente na conversão de APIs para o DCAT-AP, com um tempo médio de processamento de aproximadamente 3,41 segundos. Isto evidencia que a aplicação é capaz de lidar com conversões repetidas de forma consistente e eficaz.

Em resumo, a aplicação desenvolvida neste trabalho demonstrou ser uma ferramenta eficaz para a conversão de dados para o padrão DCAT-AP. A sua conformidade com o DCAT-AP e o seu desempenho eficiente em termos de tempo de processamento fazem dela uma escolha sólida para organizações que desejam disponibilizar dados abertos de alta qualidade e interoperáveis.

Este trabalho contribui para a promoção de dados abertos na União Europeia e para a melhoria da interoperabilidade de dados entre diferentes sistemas e organizações. No

futuro, podem ser consideradas melhorias adicionais na aplicação.

O potencial impacto positivo desta aplicação no ecossistema de dados abertos é significativo, e espera-se que continue a desempenhar um papel importante na promoção da transparência e na disponibilização de dados de alta qualidade para o benefício de todos.

6.1 Trabalho Futuro

Para futuros desenvolvimentos, podem ser consideradas as seguintes melhorias:

- **Base de Dados SPARQL Dinâmico:** Uma futura implementação pode incluir um banco de dados SPARQL dinâmico que permita o armazenamento de namespaces e termos de forma escalável e flexível;
- **Adição de Termos Personalizados:** Pode ser considerada a funcionalidade que permite aos utilizadores adicionar termos personalizados a namespaces existentes ou criar os seus próprios namespaces;
- **Upload de Termos em XML:** A aplicação pode ser estendida com a capacidade de carregar termos via upload de arquivos XML, permitindo que os utilizadores processem conjuntos de termos personalizados;
- **Validação de Dados:** Para garantir a qualidade e a consistência dos dados, verificações e validações podem ser implementadas para assegurar a conformidade com diretrizes específicas;
- **Flexibilidade de Conversão:** A aplicação, embora tenha sido desenvolvida para converter para o padrão DCAT-AP, pode ser facilmente adaptada para converter para outros padrões sem a necessidade de mudanças significativas no código. Um bloco de código adicional pode ser implementado para suportar a conversão para o padrão desejado, mantendo a estrutura principal da aplicação inalterada.

Estas melhorias podem tornar a aplicação ainda mais valiosa para a comunidade de dados abertos e continuar a promover a interoperabilidade e a disponibilidade de dados de alta qualidade.

Referências

[1] Sunlight Foundation, *Ten Principles for Opening Up Government Information*, [Online; accessed 26-JANUARY-2022], 2010. URL: https://sunlightfoundation.com/wp-content/uploads/sites/2/2016/11/Ten-Principles-for-Opening-Up-Government-Data.pdf.

[2] AMA | dados.gov, *GUIA DADOS ABERTOS*, [Online; accessed 02-JANUARY-2022], 2016. URL: https://www.ama.gov.pt/documents/24077/24804/guia_dados_abertos_ama.pdf/aa97d8e8-c5fe-47ab-9500-734948c02b19.

[3] Earle Castledine & Craig Sharkie, *jQuery: Novice to Ninja*. SitePoint, 2012.

[4] Vitor Gonçalves, *A Web Semântica no Contexto Educativo*, [Online; accessed 27-JANUARY-2022], 2007. URL: https://repositorio-aberto.up.pt/bitstream/10216/11104/2/Texto\%20integral.pdf.

[5] Flavio Cirillo, Gürkan Solmaz, Everton Luís Berz, Martin Bauer, Bin Cheng & Ernö Kovacs, "A Standard-Based Open Source IoT Platform: FIWARE", *IEEE Internet of Things Magazine*, vol. 2, n.º 3, páginas 12–18, 2019. DOI: 10.1109/IOTM.0001.1800022.

[6] Lucas Ângelo Silveira, Diego José Macêdo, Ramón Martins Sodoma da Fonseca, Milton Shintaku & Lucas Rodrigues Costa, "Guia do usuário CKAN", 2017.

[7] European Commission, *The official portal for European data*, [Online; accessed 06-JANUARY-2022], 2021. URL: https://en.wikipedia.org/wiki/CKAN.

[8] "Dados ao serviço de Lisboa: Caracterização da situação atual e identificação de necessidades e investigação de soluções", 2022.

[9] Jacquelyn Bulao, *How Much Data Is Created Every Day in 2021?*, [Online; accessed 30-December-2021], 2021. URL: https://techjury.net/blog/how-much-data-is-created-every-day/.

[10] *Data Catalog Vocabulary (DCAT) - Version 2*, [Online; accessed 19-JANUARY-2022], 2020. URL: https://www.w3.org/TR/vocab-dcat/.

[11] "DCAT-AP @ONLINE". (nov. de 2021), URL: https://github.com/SEMICeu/DCAT-AP.

[12] A. Dragan, *DCAT Application Profile for data portals in Europe Version 2.0.1*, [Online; accessed 20-JANUARY-2022], 2019. URL: https://joinup.ec.europa.eu/sites/default/files/distribution/access_url/2020-06/e4823478-4458-4546-9a85-3609867ad089/DCAT_AP_2.0.1.pdf.

[13] *CKAN + DCAT*, [Online; accessed 28-JANUARY-2022]. URL: https://extensions.ckan.org/extension/dcat/.

[14] Dublin Core Metadata Initiative et al., "Dublin core metadata element set, version 1.1", 2012. URL: https://www.dublincore.org/specifications/dublin-core/dces/.

[15] Paola Espinoza-Arias, Daniel Garijo & Oscar Corcho, "Mapping the web ontology language to the openapi specification", em *Advances in Conceptual Modeling: ER 2020 Workshops CMAI, CMLS, CMOMM4FAIR, CoMoNoS, EmpER, Vienna, Austria, November 3–6, 2020, Proceedings 39*, Springer, 2020, páginas 117–127.

[16] Esther Huyer and Laura van Knippenberg, *The Economic Impact of Open Data: Opportunities for value creation in Europe*, [Online; accessed 30-December-2021], 2021. URL: https://data.europa.eu/sites/default/files/the-economic-impact-of-open-data.pdf.

[17] European Commission, *The official portal for European data*, [Online; accessed 30-December-2021], 2021. URL: https://data.europa.eu/en.

[18] Anthony Ferguson & Michael J. Humphries, *Practical SQL: The Sequel*. O'Reilly, 2010.

[19] Roy Thomas Fielding, "REST: architectural styles and the design of network-based software architectures", *Doctoral dissertation, University of California*, 2000.

[20] Felipe Silva, *Estudo de Componentes de FIWARE Para IoT e Cidades Inteligentes*, [Online; accessed 25-FEBRUARY-2023], 2019. URL: https://ri.ufs.br/bitstream/riufs/11236/2/Felipe_Matheus_Conceicao_Silva.pdf.

[21] David Flanagan, *JavaScript: The Definitive Guide*. O'Reilly Media, Inc., 2011.

[22] James Hendler, Ora Lassila & Tim Berners-Lee, "The semantic web", *Scientific American*, vol. 284, n.º 5, páginas 34–43, 2001.

[23] Steven Holzner, *AJAX: A Beginner's Guide*. McGraw-Hill Education, 2008.

[24] Wallace Jackson, *JSON Quick Syntax Reference*. Apress, 2016.

[25] Kaggle Inc., *Kaggle.com*, [Online; accessed 30-December-2021], 2021. URL: https://www.kaggle.com.

[26] Fabian Kirstein, Benjamin Dittwald, Simon Dutkowski, Yury Glikman, Sonja Schimmler & Manfred Hauswirth, "Linked data in the european data portal: A comprehensive platform for applying dcat-ap", em *International Conference on Electronic Government*, Springer, 2019, páginas 192–204.

[27] T. Berners-Lee, *Linked Data*, [Online; accessed 15-JANUARY-2022], 2006. URL: https://www.w3.org/DesignIssues/LinkedData.html.

[28] LinkingOpenDATA WC3 Sweo Community Project, *SweoIG/TaskForces/CommunityProjects/LinkingOpenData*, [Online; accessed 26-JANUARY-2022], 2017. URL: https://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData.

[29] H. Silva, A. Serronha, *METADADOS INSPIRE*, [Online; accessed 29-JANUARY-2022], 2015. URL: https://snig.dgterritorio.gov.pt/sites/default/files/documentos/307/SNIGontheroadMetadados_Silva_Serronha.pdf.

[30] Steven J Miller, *Metadata for digital collections: a how-to-do-it manual*. Neal-Schuman Publishers New York, 2011.

[31] Claire Jenik, *A Minute on the Internet in 2021*, [Online; accessed 30-December-2021], 2021. URL: https://www.statista.com/chart/25443/estimated-amount-of-data-created-on-the-internet-in-one-minute/.

[32] Anastasija Nikiforova & Keegan McBride, "Open government data portal usability: A user-centred usability analysis of 41 open government data portals", *Telematics and Informatics*, vol. 58, pág. 101 539, 2021.

[33] Ian Niles & Adam Pease, "Towards a standard upper ontology", em *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*, 2001, páginas 2–9.

[34] *OpenAPI Initiative*, [Online; accessed 12-MARCH-2022], 2021. URL: https://www.openapis.org.

[35] S.Bechhofer, F. Harmelen, J. Hendler,I. Horrocks,D. McGuinness,P. Patel-Schneider, L. Stein, *OWL Web Ontology Language Reference*, [Online; accessed 27-JANUARY-2022], 2004. URL: https://www.w3.org/TR/owl-ref.

[36] Jung-ran Park & Andrew Brenza, "Evaluation of semi-automatic metadata generation tools: A survey of the current state of the art", *Information technology and libraries*, vol. 34, n.º 3, páginas 22–42, 2015.

[37] CML, *CONJUNTOS DE DADOS*, [Online; accessed 20-January-2022], 2018. URL: https://lisboaaberta.cm-lisboa.pt/index.php/pt/dados/conjuntos-de-dados.

[38] Alfonso Quarati, "Open Government Data: Usage trends and metadata quality", *Journal of Information Science*, pág. 01 655 515 211 027 775, 2021.

[39] O. Lassila, *Resource Description Framework (RDF) Model and Syntax Specification*, [Online; accessed 15-JANUARY-2022], 1999. URL: https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[40] José Eduardo Santarem Segundo, "Representação iterativa: um modelo para repositórios digitais", 2010.

[41] José Eduardo SANTARÉM SEGUNDO, "Web Semântica: Introdução a recuperação de dados usando SPARQL", 2017.

[42] *Smart Data Models*, [Online; accessed 10-FEB-2023], 2022. URL: https://smartdatamodels.org.

[43] António Serrador, João Tremoceiro, Nuno Cota, Nuno Cruz & Nuno Datia, "iLX - A Success Case in Public Tender Methodology", em *CENTERIS 2018 - Conference on ENTERprise Information Systems / ProjMAN 2018 - International Conference on Project MANagement / HCIST 2018 - International Conference on Health and Social Care Information Systems and Technologies*, 2018.

[44] The World Bank Group, *Open Data Essentials*, [Online; accessed 01-January-2022], 2019. URL: http://opendatatoolkit.worldbank.org/en/essentials.html.

[45] D. Beckett, T. Berners-Lee, E. Prud'hommeaux, G. Carothers, L. Machina, *RDF 1.1 Turtle: Terse RDF Triple Language*, [Online; accessed 29-JANUARY-2022], 2014. URL: https://www.w3.org/TR/turtle/.

[46] Alan Freihof Tygel, "Semantic Tags for Open Data Portals: Metadata Enhancements for Searchable Open Data",

[47] William S. Vincent, *Django 3 - Web Development*. Packt Publishing, 2020.

[48] Iris Xie & Krystyna Matusiak, *Discover digital libraries: Theory and practice*. Elsevier, 2016, cap. 5, páginas 129–170.

Anexo do modelo de metadata intermediário

Os campos e a repetiva metadata (Modelo de metadados intermédio) que a aplicação espera mapear para poder uniformizá-los para DCAT-AP.

```
{
        "titleDataset":{
            "type": "https://schema.org/name",
            "description": "Property. Model:'https://schema.org/name'.
    Titulo do dataset",
            "parent": "Dataset"
        },
        "descriptionDataset":{
            "type": "https://schema.org/description",
            "description": "Property. Model:'http://purl.org/dc/terms/
    description'. descricao do dataset.",
            "parent": "Dataset"
        },
        "keyword_tagDataset":{
            "type": "https://schema.org/keywords",
            "description": "Property. Model:'https://schema.org/keywords'.
    etiquetatas do dataset.",
            "parent": "Dataset"
        },
        "publisherDataset":{
            "type": "https://schema.org/publisher",
```

```
19            "description": "Property. Model:'https://schema.org/publisher'.
      editor do dataset.",
20            "parent": "Dataset"
21        },
22         "accessDataset":{
23            "type": "http://purl.org/dc/terms/accessRights",
24            "description": "Property. Model:'http://purl.org/dc/terms/
    accessRights'. acesso ou restricoes do dataset.",
25            "parent": "Dataset"
26        },
27         "sampleDataset":{
28            "type": "https://www.w3.org/ns/adms#sample",
29            "description": "Property. Model:'https://www.w3.org/ns/adms#
    sample'. amostra do dataset.",
30            "parent": "Dataset"
31        },
32        "update_modificationDataset":{
33            "type": "https://schema.org/dateModified",
34            "description": "Property. Model:'https://schema.org/
    dateModified'.",
35            "parent": "Dataset"
36        },
37        "contactPointDataset":{
38            "type": "https://schema.org/contactPoint",
39            "description": "Property. Model:'https://schema.org/
    contactPoint'. data de modificacao do dataset.",
40            "parent": "Dataset"
41        },
42
43        "titleCatalog": {
44            "type": "https://schema.org/name",
45            "description": "Property. Model:'https://schema.org/name'.
    Titulo do Catalogo",
46            "parent": "Catalog"
47        },
48        "descriptionCatalog":{
49            "type": "https://schema.org/description",
50            "description": "Property. Model:'https://schema.org/description
     '. descricao do catalogo.",
51            "parent": "Catalog"
52        },
53        "licenseCatalog":{
54            "type": "https://schema.org/license",
55            "description": "Property. Model:'https://schema.org/license '.
    licença usada pela organizacao.",
```

```
56              "parent": "Catalog"
57          },
58          "languageCatalog":{
59              "type": "https://schema.org/language",
60              "description": "Property. Model:'https://schema.org/language '.
     ",
61              "parent": "Catalog"
62          },
63          "publisherCatalog":{
64              "type": "https://schema.org/publisher",
65              "description": "Property. Model:'https://schema.org/publisher'.
    ",
66              "parent": "Catalog"
67          },
68          "RightsCatalog":{
69              "type": "http://purl.org/dc/terms/rights",
70              "description": "Property. Model:'http://purl.org/dc/terms/
    rights'. permissoes do dataset.",
71              "parent": "Dataset"
72          },
73          "update_modificationCatalog":{
74              "type": "https://schema.org/dateModified",
75              "description": "Property. Model:'https://schema.org/
    dateModified'. data de modificacao do catalogo.",
76              "parent": "Catalog"
77          },
78          "creatorCatalog":{
79              "type": "http://purl.org/dc/terms/creator",
80              "description": "Property. Model:'http://purl.org/dc/terms/
    creator'. autor do catalogo.",
81              "parent": "Catalog"
82          },
83
84          "modifiedCatalogRcord":{
85              "type": "https://schema.org/dateModified",
86              "description": "Property. Model:'https://schema.org/
    dateModified '.",
87              "parent": "CatalogRecord"
88          },
89
90        "titleDistribution": {
91              "type": "https://schema.org/name",
92              "description": "Property. Model:'https://schema.org/name'.",
93              "parent": "Distribution"
94          },
```

```
95          "accessURLDistribution":{
96              "type": "https://schema.org/contentUrl",
97              "description": "Property. Model:'https://schema.org/contentUrl
        '.",
98              "parent": "Distribution"
99          },
100         "descriptionDistribution":{
101             "type": "https://schema.org/description",
102             "description": "Property. Model:'https://schema.org/description
        '.",
103             "parent": "Distribution"
104         },
105         "formatDistribution":{
106             "type": "https://schema.org/encodingFormat",
107             "description": "Property. Model:'https://schema.org/
        encodingFormat '.",
108             "parent": "Distribution"
109         },
110         "licenseDistribution":{
111             "type": "https://schema.org/license",
112             "description": "Property. Model:'https://schema.org/license '",
113             "parent": "Distribution"
114
115         },
116
117         "partOfDataset":{
118             "type": "https://schema.org/Number",
119             "description": "the number (index in the list) means which
        dataset this distribution belongs to",
120             "parent": "Distribution"
121
122         }
123
124     }
```

B

Anexo dados da API

Exemplo de dados que se espera do retorno da API.


```
[
    {
        "id": "myDevice-wastecontainer-sensor-345",
        "type": "DeviceModel",
        "name": "myDevice Sensor for Containers 345",
        "brandName": "myDevice",
        "modelName": "S4Container 345",
        "manufacturerName": "myDevice Inc.",
        "category": [
            "sensor"
        ],
        "function": [
            "sensing"
        ],
        "controlledProperty": [
            "fillingLevel",
            "temperature"
        ],
        "titleD": "titulo dataset",
        "descriptionD": "descrição dataset",
        "update_modificationD": "Novembro 12, 2022, 15:28 (Europe/Lisbon)",
        "contactPointD": "contacto",
        "keyword_tagD": [
            "tagA",
            "tagB",
```

```
            "tagC"
        ],
        "publisherD": "http://lisboaaberta.cm-lisboa.pt",
        "accessD": "GrupA",
        "sampleD": "https://geodados-cml.hub.arcgis.com/",
        "titleC": "titulo Catalogo",
        "descriptionC": "descrição catalogo",
        "licenseC": "CC BY-SA",
        "publisherC": "http://lisboaaberta.cm-lisboa.pt",
        "languageC": "https://publications.europa.eu/resource/authority/
    language/POR",
        "RightsC": [
            "RightA",
            "RightB"
        ],
        "update_modificationC": "Novembro 12, 2022, 15:28 (Europe/Lisbon)",
        "creatorC": "Othmane El Arbaoui",
        "modifiedCR": "Novembro 14, 2022, 15:28 (Europe/Lisbon)",
        "accessURLDist": [
            "https://services.arcgis.com/1dSrzEWVQn5kHHyK/ArcGIS/rest/
    services/Administracao_Publica/FeatureServer/0/query?where=1%3D1&
    outFields=*&f=pgeojson",
            "https://www.teste.com"
        ],
        "descriptionDist": [
            "descrição distribuição",
            "descriçao22"
        ],
        "formatDist": [
            "GeoJSON",
            "json"
        ],
        "licenseDist": [
            "CC BY-NC",
            "CC BY-NC-ND"
        ],
        "partOfD": [
            0,
            0
        ]
    }
]
```

Anexo metadados da API

Exemplo de metadata que se espera do retorno da API.

```
{
        "titleD": {
            "type": "https://schema.org/name",
            "description": "Property. Model:'https://schema.org/name'.
    Titulo do dataset",
            "parent": "Dataset"
        },
        "descriptionD": {
            "type": "https://schema.org/description",
            "description": "Property. Model:'http://purl.org/dc/terms/
    description'. descrição do dataset.",
            "parent": "Dataset"
        },
        "keyword_tagD": {
            "type": "https://schema.org/keywords",
            "description": "Property. Model:'https://schema.org/keywords'.
    etiquetatas do dataset.",
            "parent": "Dataset"
        },
        "publisherD": {
            "type": "https://schema.org/publisher",
            "description": "Property. Model:'https://schema.org/publisher'.
     editor do dataset.",
            "parent": "Dataset"
        },
```

```
22          "accessD": {
23              "type": "http://purl.org/dc/terms/accessRights",
24              "description": "Property. Model:'http://purl.org/dc/terms/
    accessRights'. acesso ou restri\ces do dataset.",
25              "parent": "Dataset"
26          },
27          "sampleD": {
28              "type": "https://www.w3.org/ns/adms#sample",
29              "description": "Property. Model:'https://www.w3.org/ns/adms#
    sample'. amostra do dataset.",
30              "parent": "Dataset"
31          },
32          "update_modificationD": {
33              "type": "https://schema.org/dateModified",
34              "description": "Property. Model:'https://schema.org/
    dateModified'.",
35              "parent": "Dataset"
36          },
37          "contactPointD": {
38              "type": "https://schema.org/contactPoint",
39              "description": "Property. Model:'https://schema.org/
    contactPoint'. data de modificação do dataset.",
40              "parent": "Dataset"
41          },
42          "titleC": {
43              "type": "https://schema.org/name",
44              "description": "Property. Model:'https://schema.org/name'.
    Titulo do Catalogo",
45              "parent": "Catalog"
46          },
47          "descriptionC": {
48              "type": "https://schema.org/description",
49              "description": "Property. Model:'https://schema.org/description
     '. descrição do catalogo.",
50              "parent": "Catalog"
51          },
52          "licenseC": {
53              "type": "https://schema.org/license",
54              "description": "Property. Model:'https://schema.org/license '.
    licença usada pela organização.",
55              "parent": "Catalog"
56          },
57          "languageC": {
58              "type": "https://schema.org/language",
```

```
59              "description": "Property. Model:'https://schema.org/language '.
    ",
60              "parent": "Catalog"
61          },
62          "publisherC": {
63              "type": "https://schema.org/publisher",
64              "description": "Property. Model:'https://schema.org/publisher'.
    ",
65              "parent": "Catalog"
66          },
67          "RightsC": {
68              "type": "http://purl.org/dc/terms/rights",
69              "description": "Property. Model:'http://purl.org/dc/terms/
    rights'. permissões do dataset.",
70              "parent": "Dataset"
71          },
72          "update_modificationC": {
73              "type": "https://schema.org/dateModified",
74              "description": "Property. Model:'https://schema.org/
    dateModified'. data de modificação do catalogo.",
75              "parent": "Catalog"
76          },
77          "creatorC": {
78              "type": "http://purl.org/dc/terms/creator",
79              "description": "Property. Model:'http://purl.org/dc/terms/
    creator'. autor do catalogo.",
80              "parent": "Catalog"
81          },
82          "modifiedCR": {
83              "type": "https://schema.org/dateModified",
84              "description": "Property. Model:'https://schema.org/
    dateModified '.",
85              "parent": "CatalogRecord"
86          },
87          "titleDist": {
88              "type": "https://schema.org/name",
89              "description": "Property. Model:'https://schema.org/name'.",
90              "parent": "Distribution"
91          },
92          "accessURLDist": {
93              "type": "https://schema.org/contentUrl",
94              "description": "Property. Model:'https://schema.org/contentUrl
    '.",
95              "parent": "Distribution"
96          },
```

```
97          "descriptionDist": {
98              "type": "https://schema.org/description",
99              "description": "Property. Model:'https://schema.org/description
        '.",
100             "parent": "Distribution"
101         },
102         "formatDist": {
103             "type": "https://schema.org/encodingFormat",
104             "description": "Property. Model:'https://schema.org/
    encodingFormat '.",
105             "parent": "Distribution"
106         },
107         "licenseDist": {
108             "type": "https://schema.org/license",
109             "description": "Property. Model:'https://schema.org/license '",
110             "parent": "Distribution"
111         },
112         "partOfD": {
113             "type": "https://schema.org/Number",
114             "description": "the number (index in the list) means which
    dataset this distribution belongs to",
115             "parent": "Distribution"
116         }
117 }
```