Instituto Militar de Engenharia
Programa de Pós-Graduação em Sistemas e Computação
Disciplina: Web Semântica Período: 2023.2
Aluna: Thaisa da Silva Pinto

DESENVOLVIMENTO DE UM MODELO DE DADOS EM RDF PARA UM DATASET DE DETECÇÃO DE INTRUSÃO: IMPLEMENTAÇÃO E PUBLICAÇÃO EM UM FAIR DATAPOINT

DEZ/2023

Sumário

1. INTRODUÇÃO

O RDF (Resource Description Framework) tem o papel de fornecer um modelo formal de dados e sintaxe para codificar metadados que podem ser processados por máquinas. O princípio por trás do RDF é fornecer interoperabilidade entre aplicativos que trocam informações na rede. Ele fornece as primitivas básicas para a criação de ontologias simples, incluindo relacionamentos de generalização para classes e propriedades (BREITMAN, 2000).

O RDF possui três princípios fundamentais: recursos, propriedades e declarações. Os recursos são objetos ou "coisas" das quais queremos falar. Cada recurso possui um identificar único, uma URI (Universal Resource Identificator) ou um literal. As propriedades descrevem relacionamentos entre os recursos, também são identificadas utilizando-se URI. As informações são representadas por declarações contendo: sujeito, predicado e objeto. As declarações em RDF podem ser representadas de três formas: triplas, grafo e XML (BREITMAN, 2000).

O RDF-Schema (RDFS) surgiu como extensão do RDF. Ele é uma linguagem de descrição de vocabulários que possibilita definir hierarquias de classes, hierarquias de propriedades e a definição de domínios e contradomínios para as propriedades, permitindo assim um primeiro conjunto de restrições sobre as triplas definidas, além de inferências que deduzem triplas não declaradas de forma explícita (LAUFER, 2015). O RDFS não fornece as classes e propriedades propriamente ditas, e sim um *framework* no qual é possível descrevê-las. As classes definidas no RDF-Schema permitem que os recursos (descritos no documento RDF) sejam definidos como instâncias ou subclasses das classes presentes no RDF-Schema. O RDF e o RDF-Schema são as fundações da Web Semântica. (BREITMAN, 2000).

Ontology Web Language (OWL) é uma linguagem que estende RDF e RDFS e oferece um conjunto muito mais amplo de tipos de restrições ao conjunto de triplas definidas. E por que definir restrições? De acordo com (LAUFER, 2015), uma das formas de se estender o que é semântica é pensar que o que faz com que diferentes pessoas possam entender o mesmo significado de algum conteúdo é restringir o número de diferentes interpretações possíveis sobre aquele conteúdo.

Diante do exposto, o objetivo deste trabalho é desenvolver um modelo de dados de um *dataset* de Sistemas de Deteção de Intrução (IDS) utilizando o RDF, implementar esse modelo utilizando o GraphDB e por fim, publicar seus metadados em um FAIR Data Point. Este trabalho está organizado da seguinte forma. A Seção 2 apresenta o *dataset* utilizado na modelagem; A Seção 3 apresenta a descrição do modelo de dados; A Seção 4 apresenta a implementação de modelo através do GraphDB; por fim, a Seção 5 apresenta a publicação dos metadados do *dataset* em um FAIR Data Point.

2. CONJUNTO DE DADOS CSE-CIC-IDS2018

O *dataset* CSE_CIC_IDS2018 ("IDS 2018 | UNB"), é um conjunto de dados de *benchmark* disponibilizado pela Universidade de New Brunswik (UNB). O mesmo é um *dataset* diversificado e abrangente para detecção de intrusões e contém representações abstratas, ou seja, simulações de eventos e comportamentos vistos na rede.

São incluídos seis cenários de ataques diferentes: força bruta, botnet, DoS, DDoS, ataques na Web e infiltração interna da rede. A infraestrutura de ataque inclui 50 máquinas e a organização vítima possui 5 departamentos e inclui 420 máquinas e 30 servidores. O conjunto de dados inclui capturas de tráfego de rede e logs do sistema de cada máquina, junto com 80 *features* (atributos) extraídos do tráfego capturado.

O dataset é composto pela coletânea de 10 arquivos, sendo 9 deles apresentando 79 atributos e o décimo arquivo contendo 83 atributos que representam 7 categorias de ocorrências: benign, brute Force, botnet, DoS, DDoS, web attacks e infiltration.

De acordo com Kenyon et al. (2020), o conjunto de dados UNB IDS está bem documentado, é bem mantido e inclui eventos brutos e metadados de fluxos significativos, os autores consideram que talvez é conjunto de dados mais útil disponível atualmente para investigação de IDS. Por esse motivo o mesmo foi escolhido para este trabalho.

3. DESCRIÇÃO DO MODELO DE DADOS

Para o desenvolvimento do modelo de dados do dataset CSE_CIC_IDS2018 foi utilizada a aplicação web WebVOWL, que permite a criação e visualização interativa de

ontologias e a geração de representações do modelo. As representações gráficas em formato de grafo podem ser convertidas em JSON, SVG, TeX, TTL e URL.

Ontologias fornecem meios para modelar as relações entre as entidades em um domínio de interesse (LAUFER, 2015). Em OWL existem três tipos de entidades:

- Instâncias: representam os recursos;
- Classes: definem conjuntos de instâncias; e
- Propriedades: representam relações binárias entre duas instâncias (object property) ou entre uma instância e um literal (datatype property).

Além das próprias definições criadas, este trabalho também utilizou a ontologia ToCo (Toucan Ontology), desenvolvida para sistemas de redes de telecomunicações. A ontologia ToCo utiliza o prefixo “net” (TOCO, [s.d.]).

A figura 1 apresenta uma visão geral do modelo de dados criado para este trabalho. A mesma imagem pode ser melhor visualizada no Anexo A deste documento.

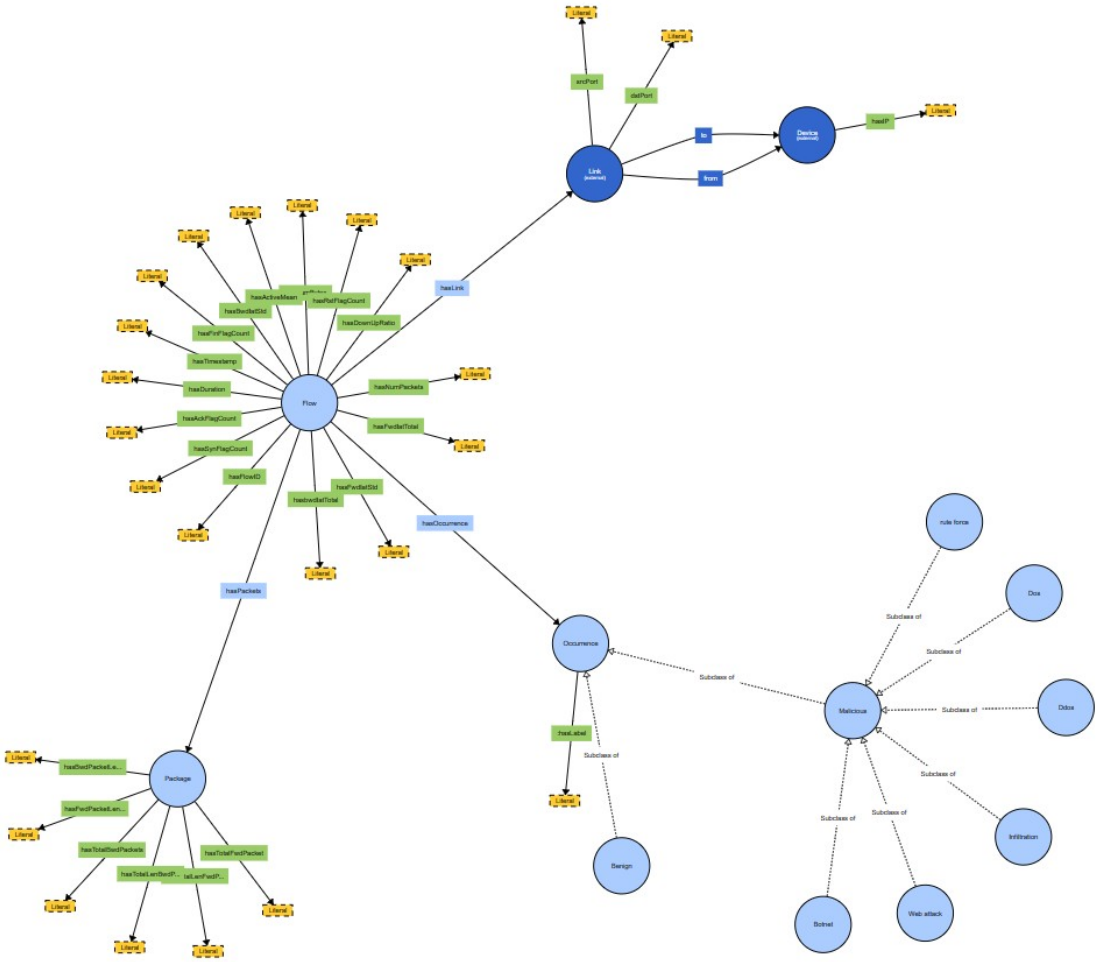

Figura 1: Visão geral do modelo de dados a partir da ferramenta WebVOWL.

3.1 Definição de classes

As classes e subclasses definidas para este trabalho estão definidas na Tabela 1.

Classe	IRI	Descrição	Subclasse
Flow	ex:Flow	Descreve o fluxo bidirecional presente em cada tupla do dataset. Esta classe possui atributos que representam recursos estatísticos como duração, número de pacotes, número de bytes, comprimento dos pacotes, etc.	
Occurrence	ex:Occurrence	Descreve as ocorrências descritas em cada tupla do dataset.	Malicious Benign
Malicious	ex:Malicious	Descreve as ocorrências rotuladas como malignas em cada tupla do dataset.	Brute-force Botnet Dos Ddos Infiltration Web_attack
Benign	ex:Benign	Descreve as ocorrências rotuladas como benignas em cada tupla do dataset.	
Package	ex:Package	Descreve os pacotes atribuídos aos fluxos. Esta classe possui atributos que representam recursos estatísticos referentes ao pacote.	
Link	net:Link	Descreve um meio (por exemplo, cabo trançado, fibra óptica, onda eletromagnética) usado para conectar dois dispositivos na rede de telecomunicações.	
Device	net:Device	Define os dispositivos da infraestrutura física do sistema de telecomunicações.	

Tabela 1: Definição das classes

3.2 Definição de propriedades

Em relação às propriedades, conforme apresentado no item 2, o *dataset* possui 10 arquivos, sendo 9 deles apresentando 79 atributos e o décimo arquivo contendo 83. Esses atributos foram categorizados em três tipos, conforme (Silva, 2022): atributos básicos de conexão, atributos de pacotes de rede, atributos de fluxo de rede. Cabe destacar que a quantidade de atributos, neste trabalho definidos como propriedades, foi reduzida por motivos didáticos.

Além das classes e propriedades, no nível de esquema também podem ser definidas restrições de domínio e de range. Uma restrição de domínio específica o domínio de uma

propriedade, ou seja, a classe de recursos que pode aparecer como sujeito da sentença. Uma restrição de range específica a que classe os valores de uma propriedade devem pertencer.

Para fins deste trabalho, as propriedades e suas respectivas restrições de domínio e de range estão definidos na Tabela 2.

Propriedades	IRI	Domínio	Range	Descrição
hasTimestamp	ex:hasTimestamp	Flow	Literal	Define o data/hora do registro do fluxo.
hasDuration	ex:hasDuration	Flow	Literal	Duração do fluxo.
hasNumBytes	ex:hasByts	Flow	Literal	Nº de bytes no fluxo por segundo.
hasNumPackets	ex:hasPkts	Flow	Literal	Nº de pacotes no fluxo por segundo.
hasFwdIatTotal	ex:hasFwdIatTotal	Flow	Literal	Tempo total entre dois pacotes enviados no sentido do fluxo.
hasbwdIatTotal	ex:hasbwdIatTotal	Flow	Literal	Tempo total entre dois pacotes enviados no sentido inverso do fluxo.
hasFwdIatStd	ex:hasFwdIatStd	Flow	Literal	Desvio padrão no tempo entre dois pacotes enviados no fluxo.
hasBwdIatStd	ex:hasBwdIatStd	Flow	Literal	Desvio padrão no tempo entre dois pacotes enviados no sentido inverso do fluxo.
hasFinFlagCount	ex:hasFinFlagCount	Flow	Literal	Número de pacotes com FIN.
hasSynFlagCount	ex:hasSynFlagCount	Flow	Literal	Número de pacotes com SYN.
hasRstFlagCount	ex:hasRstFlagCount	Flow	Literal	Número de pacotes com RST.
hasAckFlagCount	ex:hasAckFlagCount	Flow	Literal	Número de pacotes com ACK.
hasDownUpRatio	ex:hasDownUpRatio	Flow	Literal	Taxa de download e upload.
hasActiveMean	ex:hasActiveMean	Flow	Literal	Tempo médio que um fluxo esteve inativo antes de se tornar ocioso.
hasOccorrence	ex:hasOccorrence	Flow	Occurrence	Define a ocorrência de um fluxo
hasPackets	ex:hasPackets	Flow	Package	Define os pacotes de um fluxo.
hasLink	ex:hasLink	Flow	Link	Define os links de um fluxo.
hasTotalFwdPacket	ex:hasTotalFwdPacket	Package	Literal	Total de pacotes no sentido do fluxo
hasTotalBwdPackets	ex:hasTotalBwdPackets	Package	Literal	Total de pacotes no sentido inverso do fluxo
hasTotalLenFwdPacket	ex:hasTotalLenFwdPacket	Package	Literal	Tamanho total do pacote no sentido do fluxo
hasTotalLenBwdPacket	ex:hasTotalLenBwdPacket	Package	Literal	Tamanho total do pacote no sentido inverso do fluxo
hasFwdPacketLengthStd	ex:hasFwdPacketLengthStd	Package	Literal	Desvio padrão do tamanho do pacote no sentido do fluxo
hasBwdPacketLenStd	ex:hasBwdPacketLenStd	Package	Literal	Desvio padrão do tamanho do pacote no sentido inverso do fluxo
from	net:from	Link	Device	Define o nó de origem do qual o link é transmitido.
to	net:to	Link	Device	Define o nó de destino do link.
srcPort	ex:srcPort	Link	Literal	Define a porta de origem
srcDest	ex:dstPort	Link	Literal	Define a porta de destino

hasIP	ex:hasIP	Device	Literal	Define o endereço IP da interface.
hasLabel	ex:hasLabel	Occurrence	Literal	Define o rotulo da ocorrência.

Tabela 2: Definição das propriedades

3.3 Instâncias

Para fins deste trabalho, foram utilizadas como exemplo oito tuplas, quatro com ocorrências rotuladas como benignas e quatro com ocorrências rotuladas como malignas do tipo DdoS, extraídas do arquivo *Thuesday 20-02-2018_TrafficForML_CICFlowMeter*.

3.4 Representação do modelo – RDF Turtle

RDF é um modelo de dados abstrato, não importando como ele seja representado, desde que a representação permaneça fiel às suas propriedades abstratas. As representações sintáticas para o modelo RDF, também denominadas de serializações são: RDF/XML, Turtle, N-Triples e JSON-LD (LAUFER, 2015). Para este trabalho, o modelo de dados foi representado em RDF Turtle (Terse RDF Triple Language). O Anexo B deste documento contém a representação RDF Turtle completa do modelo de dados desenvolvido. A seguir são apresentados alguns extratos da representação do modelo em Turtle para exemplificação.

As figuras 2, 3 e 4 representam respectivamente a definição das classes Malicious, Link e Ddos. Na classe Malicious destaca-se a definição da classe como sub-classe da classe Occurrence a partir da propriedade **rdfs:subClassOf**. Além disso, cabe destacar a definição da classe Malicious como disjunta da classe Benign, ou seja, as duas não possuem instâncias em comum. Na classe Link, destaca-se a utilização da ontologia TOCO, através do prefixo “net”, e na classe Ddos destaca-se a definição da propriedade **owl:equivalentClass** utilizada para indicar que uma classe ou propriedade específica numa ontologia é equivalente a uma classe ou propriedade numa segunda ontologia. Neste trabalho, a propriedade **owl:equivalentClass** foi utilizada para enriquecer o *dataset* fonte com a descrição dos tipos de ataques.

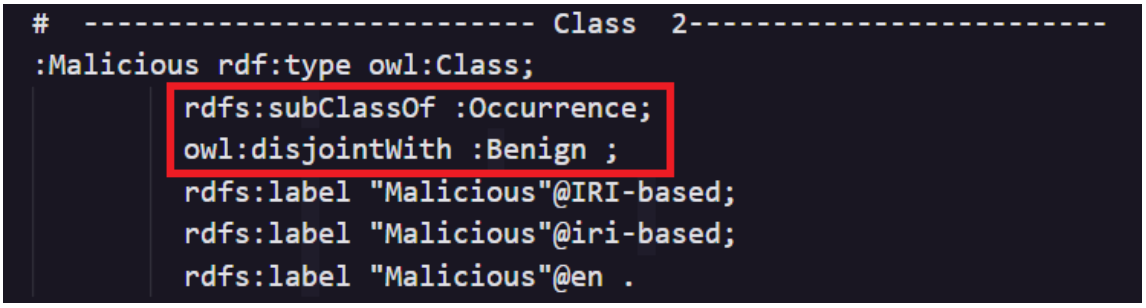

Figura 2: Definição da classe Malicious

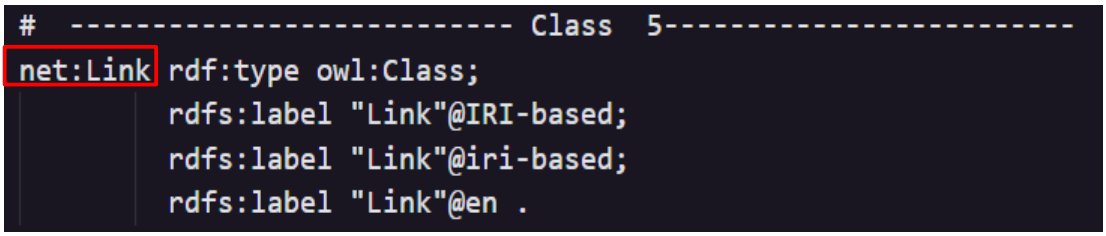

Figura 3: Definição da classe Link

Figura 4: Definição da classe Ddos.

As figuras 5 e 6 representam respectivamente as definições de uma propriedade do tipo **owl:DataTypeProperty** e **owl:ObjectProperty**.

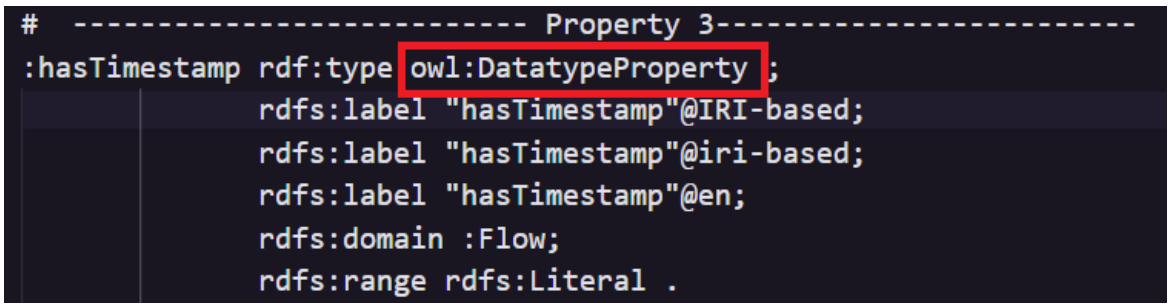

Figura 5: Definição de uma propriedade do tipo owl:DataTypeProperty

```
#  ------------------------- Property 20-------------------------
:hasOccurrence rdf:type owl:ObjectProperty ;
               rdfs:label "hasOccurrence"@IRI-based;
               rdfs:label "hasOccurrence"@iri-based;
               rdfs:label "hasOccurrence"@en;
               rdfs:domain :Flow;
               rdfs:range :Occurrence .
```

Figura 6: Definição de uma propriedade do tipo owl:DataTypeProperty

As figuras 7 e 8 apresentam respectivamente instâncias das classes Flow, Device e Link.

```
<http://example.com/CSE-CIC-IDS2018/flow_1> rdf:type :Flow;
                                            ex:hasFlowID "172.31.69.1-172.31.69.25-67-68-17" ;
                                            ex:hasTimestamp "20/02/2018 08:50:51" ;
                                            ex:hasDuration "716" ;
                                            ex:hasNumBytes "878491.6201" ;
                                            ex:hasNumPackets "2793.296089" ;
                                            ex:hasFwdIatTotal "0" ;
                                            ex:hasbwdIatTotal "0" ;
                                            ex:hasFwdIatStd "0" ;
                                            ex:hasBwdIatStd "0" ;
                                            ex:hasFinFlagCount "0" ;
                                            ex:hasSynFlagCount "0" ;
                                            ex:hasRstFlagCount "0" ;
                                            ex:hasAckFlagCount "0" ;
                                            ex:hasDownUpRatio "1" ;
                                            ex:hasActiveMean "0" ;
                                            ex:hasPackets <http://example.com/CSE-CIC-IDS2018/package_1> ;
                                            ex:hasLink <http://example.com/CSE-CIC-IDS2018/link_1> ;
                                            ex:hasOccurrence <http://example.com/CSE-CIC-IDS2018/occurrence_1> .
```

Figura 7: Instância da classe Flow.

```
<http://example.com/CSE-CIC-IDS2018/device_2> rdf:type net:Device ;
                                                  ex:hasIP "172.31.69.1" .

<http://example.com/CSE-CIC-IDS2018/link_1> rdf:type net:Link ;
                                            net:from <http://example.com/CSE-CIC-IDS2018/device_1> ;
                                            net:to <http://example.com/CSE-CIC-IDS2018/device_2> ;
                                            ex:srcPort "68" ;
                                            ex:dstPort "67" .
```

Figura 8: Instâncias das classes Link e Device.

4. IMPLEMENTAÇÃO DO MODELO DE DADOS NO GRAPHDB

Esta seção apresenta a implementação do modelo formal desenvolvido na seção anterior através do GraphDB. O GraphDB é um banco de dados gráfico compatível com especificações RDF e SPARQL. O Workbench é usado para pesquisar, explorar e gerenciar repositórios GraphDB. O banco de dados foi instalado localmente na máquina da autora, e acesso através da URL localhost:7200/. O modelo formal no formato TTL foi importado no GraphDB.

A seguir são apresentadas alguns exemplos de visualizações gráficas através do GraphDB.

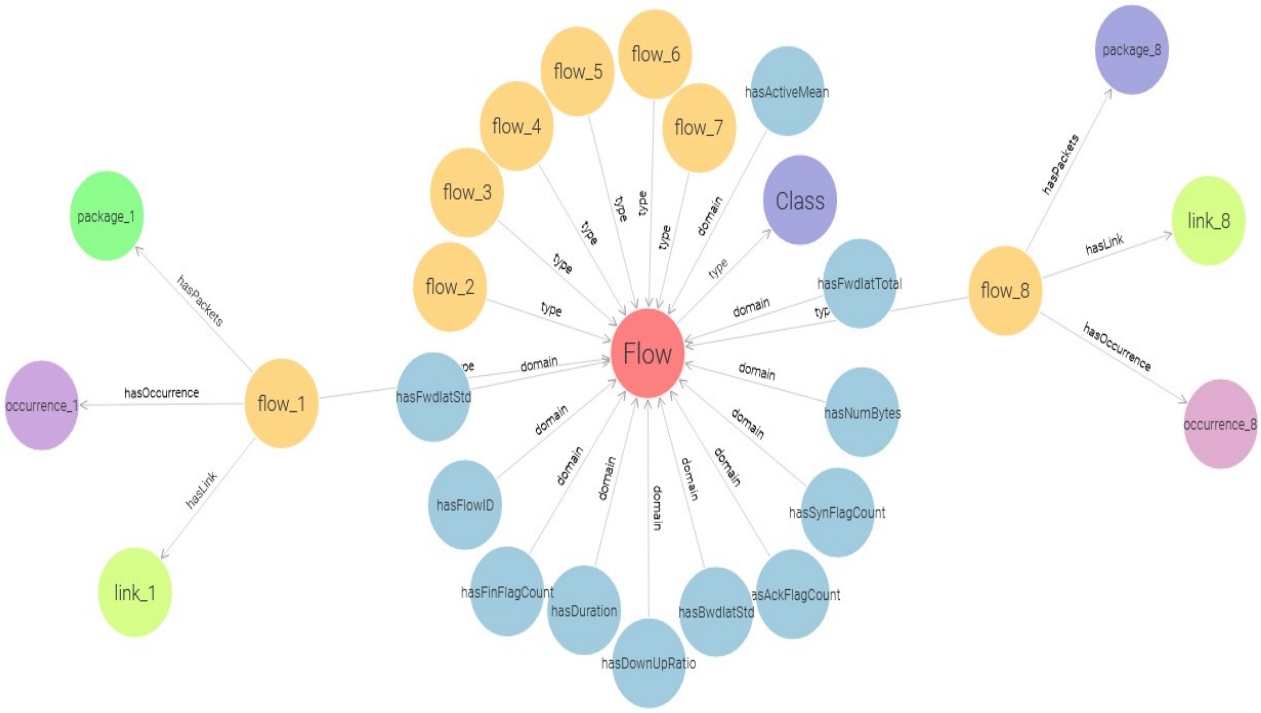

Figura 9: Extrato com a visualização do grafo da classe Flow através do GraphDB.

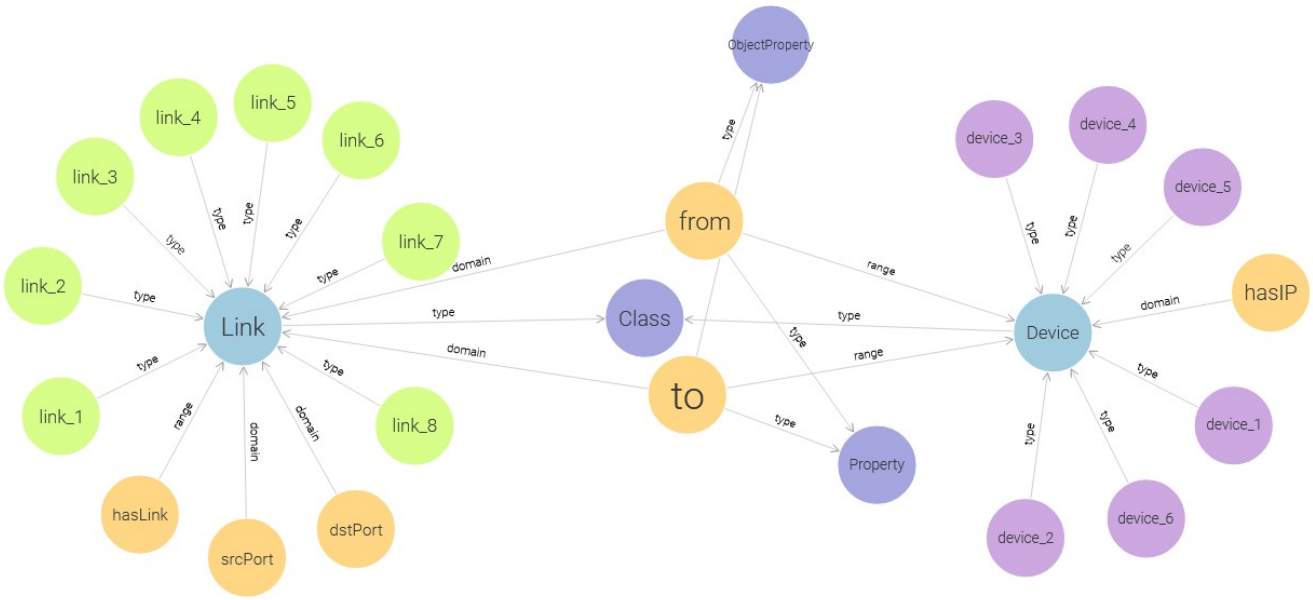

Figura 10: Extrato com a visualização do grafo das classes Link e Device através do GraphDB

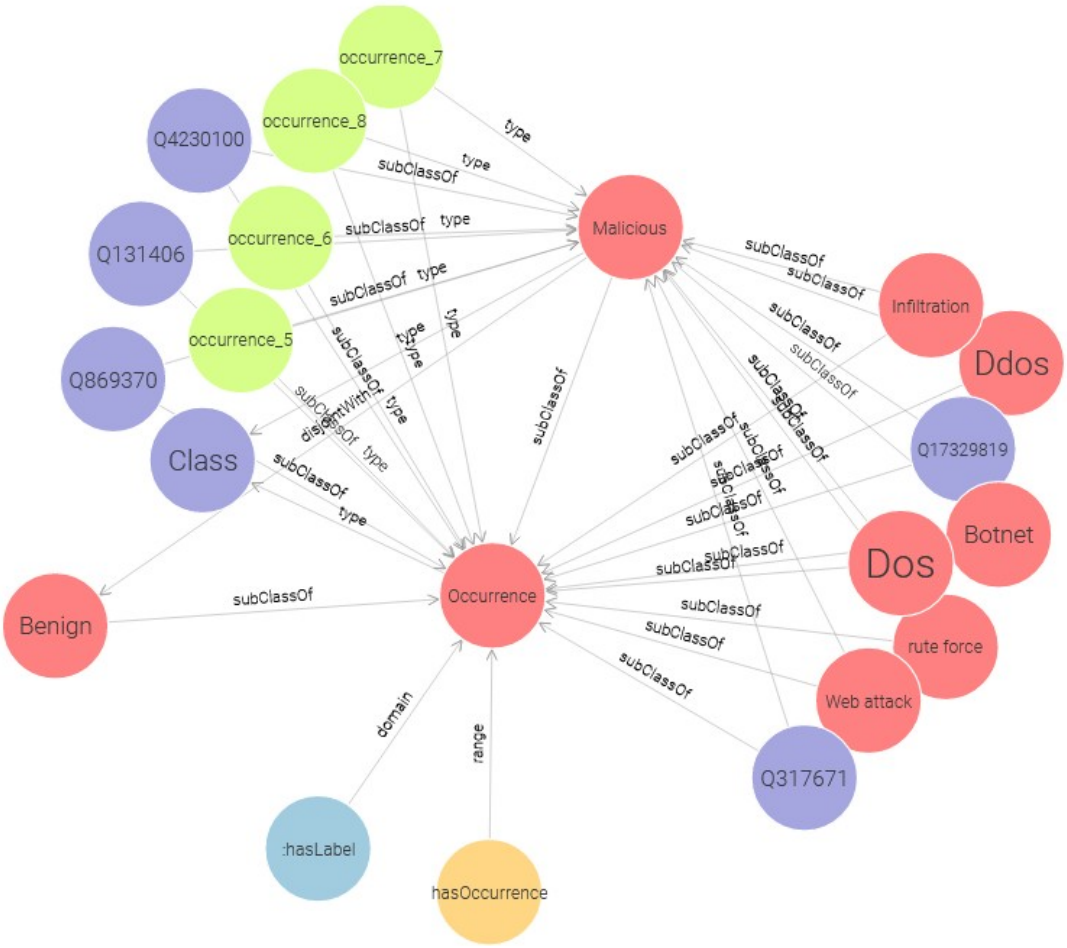

Figura 11: Extrato com a visualização do grafo da classe Occurrence e suas respectivas subclasses Benign e Malicious

A partir das instâncias definidas na representação importada, foi possível realizar consultas em SPARQL. Abaixo exemplos de algumas consultas realizadas:

1) Selecionar o data/hora dos fluxos:

```
PREFIX ex: <http://example.com/CSE-CIC-IDS2018/>
select ?timestamp where {
    ?flow ex:hasTimestamp ?timestamp .
}
```

	timestamp
1	"20/02/2018 08:50:51"
2	"20/02/2018 08:32:55"
3	"20/02/2018 09:29:47"
4	"20/02/2018 09:48:07"
5	"20/02/2018 10:13:54"
6	"20/02/2018 10:13:54"
7	"20/02/2018 10:13:54"
8	"20/02/2018 10:13:54"

2) Selecionar os endereços IP de origem e o tipo de ataque dos fluxos com ocorrências maligna:

```
PREFIX ex: <http://example.com/CSE-CIC-IDS2018/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX net: <http://purl.org/toco/>
PREFIX : <http://example.com/CSE-CIC-IDS2018>
select ?flow ?src_ip ?type where {
    ?flow ex:hasLink ?link .
    ?link net:from ?src_device .
    ?src_device ex:hasIP ?src_ip .
    ?flow ex:hasOccurrence ?occurrence .
    ?occurrence rdf:type ?type .
    ?type rdfs:subClassOf ex:Malicious .
}
```

	flow	src_ip	label
1	ex:flow_6	"52.14.136.135"	"DDoS"
2	ex:flow_5	"52.14.136.135"	"DDoS"
3	ex:flow_7	"52.14.136.135"	"DDoS"
4	ex:flow_8	"52.14.136.135"	"DDoS"

3) Selecionar os endereços IP atacados e seus respectivos IP atacantes:

```
PREFIX ex: <http://example.com/CSE-CIC-IDS2018/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX net: <http://purl.org/toco/>
PREFIX : <http://example.com/CSE-CIC-IDS2018>
select ?src_ip ?dst_ip ?label where {
    ?flow ex:hasLink ?link .
    ?link net:from ?src_device .
    ?link net:to ?dst_device .
    ?src_device ex:hasIP ?src_ip .
    ?dst_device ex:hasIP ?dst_ip .
    ?flow ex:hasOccurrence ?occurrence .
    ?occurrence rdf:type ex:Malicious .
    ?occurrence ex:hasLabel ?label .
```

	src_ip	dst_ip	label
1	"52.14.136.135"	"172.31.69.25"	"DDoS"
2	"52.14.136.135"	"172.31.69.25"	"DDoS"
3	"52.14.136.135"	"172.31.69.25"	"DDoS"
4	"52.14.136.135"	"172.31.69.25"	"DDoS"

4) Informar a quantidade de ocorrências malignas.

```
PREFIX ex: <http://example.com/CSE-CIC-IDS2018/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX net: <http://purl.org/toco/>
select (COUNT(?occurrence)AS ?count)
where{
    ?flow ex:hasOccurrence ?occurrence .
    ?occurrence rdf:type ex:Malicious .
}
```

	count
1	"4"^^xsd:integer

5) Executar uma consulta federada que retorne a descrição sobre dos tipos de ataques ocorridos.

```
PREFIX ex: <http://example.com/CSE-CIC-IDS2018/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
select DISTINCT ?type ?description
where {
      ?flow ex:hasOccurrence ?occurrence .
      ?occurrence rdf:type ?type .
      ?type owl:equivalentClass ?x .
      SERVICE  <https://query.wikidata.org/sparql>
      {?x <http://schema.org/description> ?description .
       FILTER (langMatches(lang(?description),"en")) .}
}
```

label	description
"Ddos"@iri-based	"cyber attack"@en
"Ddos"@en	"cyber attack"@en

5. PRINCÍPIOS FAIR E FAIR DATA POINT

Os princípios FAIR (Findable, Accessible, Reusable and Interoperable), foram propostos em 2016 (WILKINSON et al., 2016), como um meio de tratar os aspectos computacionais envolvidos na disponibilização dos dados em diferentes contextos, dentre eles os dados de pesquisa (TORINO; VIDOTTI; CONEGLIAN, 2021). São um conjunto de práticas orientadoras para gestão e administração de dados, fornecendo diretrizes para melhorar a localização, acessibilidade, interoperabilidade e reutilização de ativos digitais. A figura 12 apresenta um resumo dos princípios FAIR.

Figura 12: Princípios FAIR. Fonte: Torino et al. (2021)

O FAIR Data Point é um servidor de aplicação que expõe metadados na internet seguindo os princípios FAIR. O conteúdo de metadados do FAIR Data Point pode ser acessado pela internet e é apresentado em formato RDF para facilitar o processamento computacional. O FAIR Data Point possui três objetivos principais (“Introdução ao FAIR Data Point”, 2020):

(i) Permitir que proprietários, criadores ou editores exponham os metadados de seus objetos digitais de uma forma que segue os princípios FAIR;

(ii) Permitir que os consumidores ou usuários descubram informações sobre objetos digitais de interesse; e

(iii) Forneça esses metadados de forma acionável por máquina.

Com base nesses objetivos, a Figura 13 retrata a arquitetura geral do FDP.

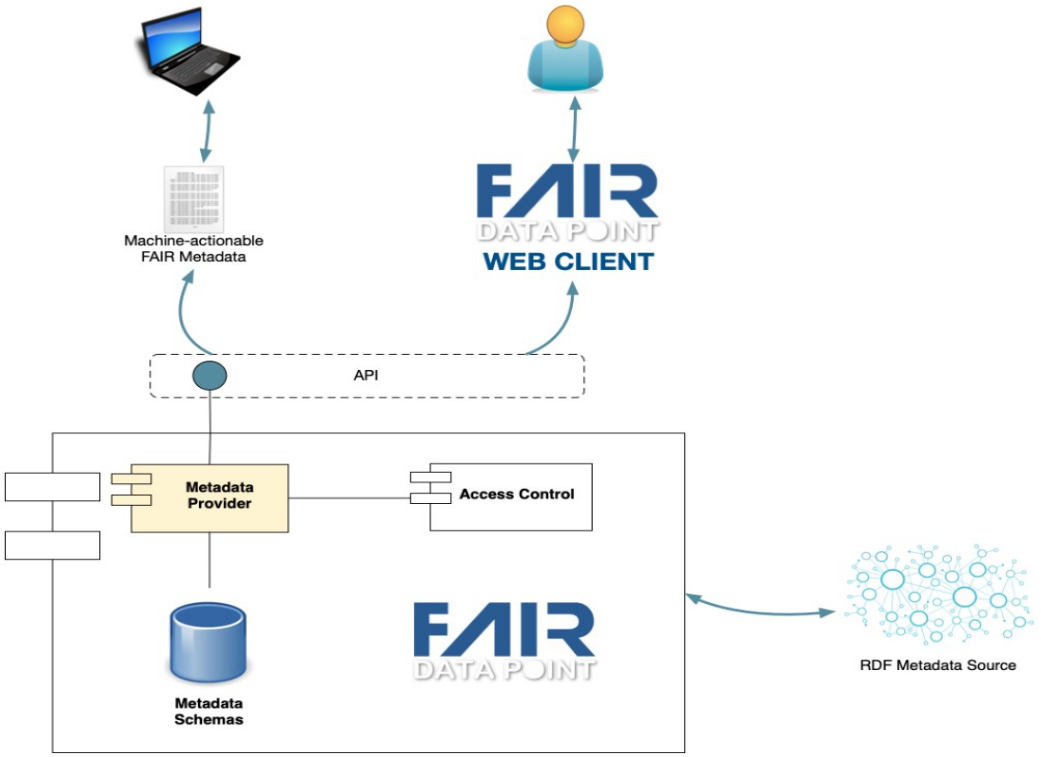

Figura 13: Arquitetura geral do FAIR DataPoint. Fonte: /

- **API:** é a forma como os aplicativos clientes interagem com o FDP
- **Metadata Provider:** provê os registros dos metadados armazenados por meio da API FDP.
- **Access Control:** Provê o controle de acesso sobre os metadados disponíveis.
- **Metadata Schemas:** Define a estrutura e a semântica dos registros de metadados relacionados.

Metadados

Os princípios FAIR dão especial atenção aos metadados. Os metadados podem ser definidos como dados que fornecem informações sobre outros dados. Esta informação pode incluir uma variedade de descrições diferentes, como origem, estrutura, proveniência, direitos e obrigações ou outras características dos objetos digitais (SANTOS et al., 2023).

O FDP usa o modelo Data Catalog Vocabulaire (DCAT) versão 2 como base para seus metadados:

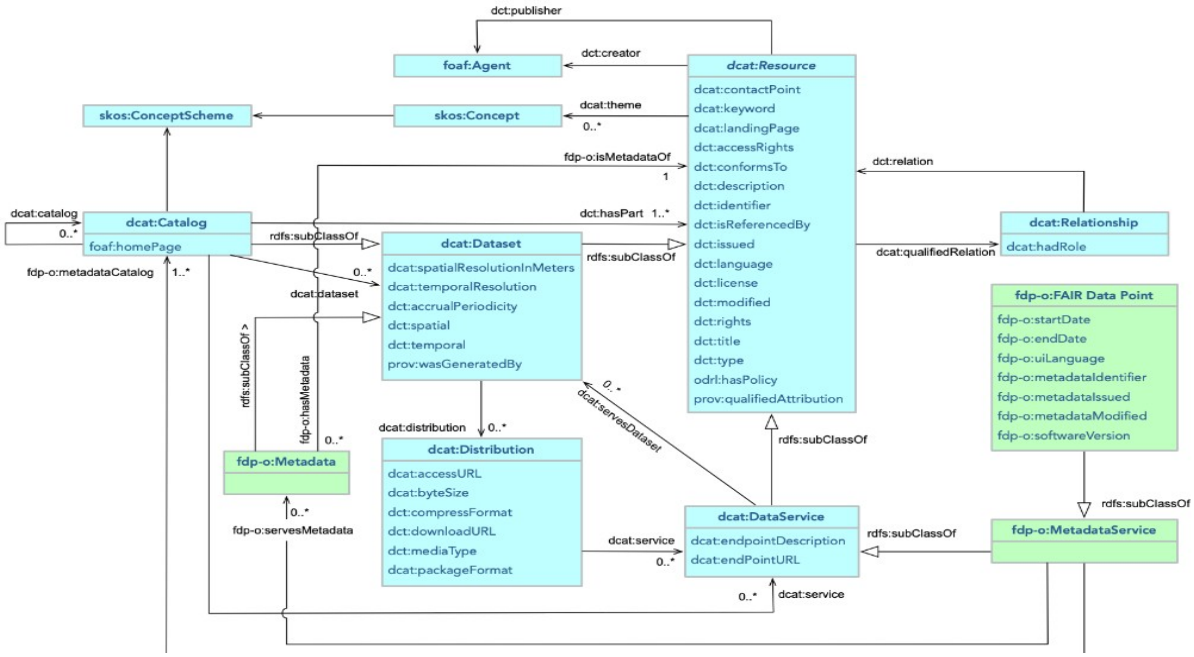

Figura 14: Extensão do modelo DCAT versão 2 para FDP. Fonte: https://specs.fairdatapoint.org/fdp-specs-v1.2.html

Conforme definido pelo DCAT, os catálogos são coleções arbitrárias de registros de metadados sobre diferentes tipos de objetos digitais (chamados de recursos nas especificações DCAT). Assim, no FDP, catálogos são usados para agrupar e organizar registros de metadados. Por exemplo, pode-se definir um catálogo de conjuntos de dados para agrupar os metadados sobre diferentes conjuntos de dados. DCAT já fornece as definições para conjuntos de dados.

- *dcat:Resource*: Resource representa entidades que podem ser descritas por um registro de metadados. Como foi definida como uma classe abstrata, não se destina a ser usada diretamente.
- *dcat:Dataset*: representa uma coleção de dados.
- *dcat:DataService*: Data Service representa serviços acessíveis por meio de uma interface (API) que atende conjuntos de dados.

- *dact:Catalog*: uma subclasse de Dataset, representa agregações de registros de metadados sobre objetos digitais. Por exemplo, um Catálogo pode conter referências aos registros de metadados de Conjuntos de Dados.

O FDP estende o modelo DCAT adicionando o conceito de FAIRDataPoint como um tipo específico de serviço de dados que serve catálogos e registros de metadados. As extensões DCAT e outros conceitos e relações específicos do FDP são definidos na ontologia do FDP (usando o prefixo de namespace fdp-o).

Instalação do FAIR Data Point

Para fins deste estudo, o objetivo da instalação do FAIR Data Point é expor os metadados do dataset CSE_CIC_IDS2018. Para tornar esse objetivo possível, as seguintes etapas foram realizadas:

(i) Instalação do FAIR DataPoint com a estrutura de metadados: FAIR Data Point → Catálogo → Conjunto de dados → Distribuição;
(ii) Criação da página de catálogo Instituto Militar de Engenharia com o objetivo de agrupar os registros de metadados dos *datasets*;
(iii) Criação de página de *dataset* com o objetivo de descrever os metadados do dataset;
(iv) Criação das páginas de distribuição do *dataset*.

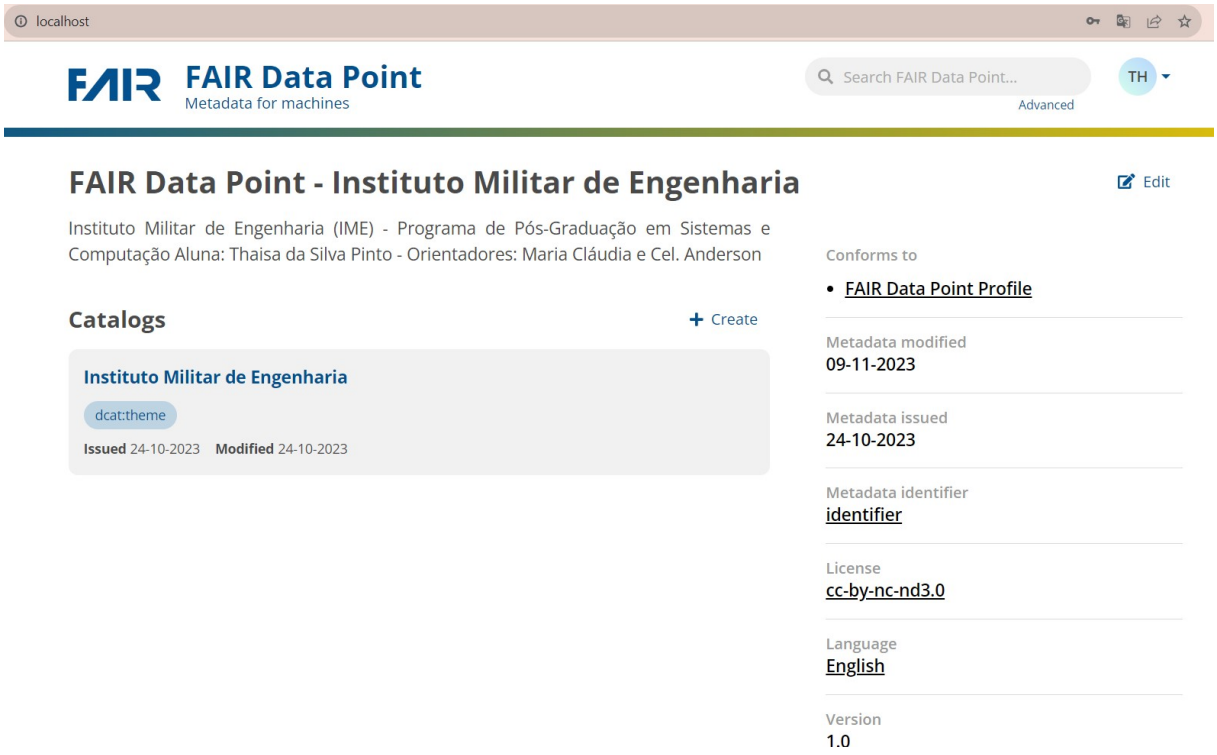

Figura 15: Página inicial com a descrição do FAIR Data Point.

Figura 16: Página com a descrição do catálogo.

Figura 17: Página com a descrição do dataset e suas distribuições.

6. REFERÊNCIAS

BREITMAN, K. K. **Web semântica: a internet do futuro.** [s.l.] Grupo Gen-LTC, 2000.

Data Catalog Vocabulary. Disponível em: <https://www.w3.org/TR/vocab-dcat-2/#classifying-datasets>. Acesso em: 13 nov. 2023.

DBPedia. Disponível em: <https://dbpedia.org/page/>. Acesso em: 20 nov. 2023.

IDS 2018 | | UNB. Disponível em: <https://www.unb.ca/cic/datasets/ids-2018.html>. Acesso em: 6 out. 2023.

Introdução ao FAIR Data Point. , 8 jun. 2020. Disponível em: <https://www.youtube.com/watch?v=PtS_ek7BXSA>. Acesso em: 20 out. 2023

KENYON, A.; DEKA, L.; ELIZONDO, D. Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets. **Computers & Security**, v. 99, p. 102022, 1 dez. 2020.

LAUFER, C. **Guia da Web Semântica**. , 2015.

OWL Web Ontology Language. Disponível em: <https://www.w3.org/TR/2004/REC-owl-ref-20040210/>. Acesso em: 20 nov. 2023.

SILVA, M. L. SEC4ML: **Anonimização de Dados de Incidentes de Segurança da Informação para tarefas de Aprendizado de Máquina**. 2022. Dissertação (Mestrado). Programa de Pós Graduação em Ciências em Sistemas e Computação. Instituto Militar de Engenharia, Rio de Janeiro, 2022.

SANTOS, L. O. B. DA S. et al. FAIR Data Point: A FAIR-Oriented Approach for Metadata Publication. **Data Intelligence**, v. 5, n. 1, p. 163–183, mar. 2023.

specs.fairdatapoint.org. Disponível em: <https://specs.fairdatapoint.org/fdp-specs-v1.2.html>. Acesso em: 13 nov. 2023.

TOCO. Disponível em: <https://qianruzhou333.github.io/toco_ontology/#d4e1810>. Acesso em: 31 out. 2023.

TORINO, E.; VIDOTTI, S. A. B. G.; CONEGLIAN, C. S. **#SejaJUSTOeCUIDADOSO: princípios FAIR e CARE na gestão de dados de pesquisa**. [s.l.] IBICT, 2021.

WILKINSON, M. D. et al. The FAIR Guiding Principles for scientific data management and stewardship. **Scientific Data**, v. 3, p. 160018, 1 mar. 2016.

W3C. Disponível em: <https://www.w3.org/TR/rdf11-primer/>. Acesso em 30 nov. 2023.