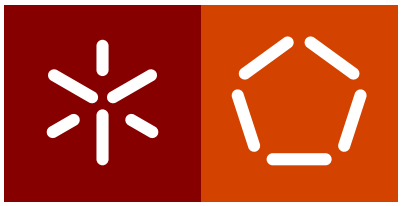
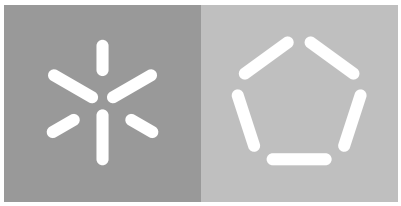


Universidade do Minho
Escola de Engenharia
Departamento de Informática

Joana Catarina Maciel Pereira

Construção Automática de CMDB

Maio 2021



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Joana Catarina Maciel Pereira

Construção Automática de CMDB

Master dissertation
Integrated Master's in Informatics Engineering

Dissertation supervised by
Professor António Luís Pinto Ferreira de Sousa

Maio 2021

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

LICENÇA CONCEDIDA AOS UTILIZADORES DESTE TRABALHO:

CC BY

https://creativecommons.org/licenses/by/4.0/

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

AGRADECIMENTOS

Em primeiro lugar tenho de agradecer aos meus pais, que sempre acreditaram em mim e me deram a oportunidade de chegar até aqui. Que fizeram todos os sacrifícios por mim para eu ter sempre o melhor. Que todos os dias são uma inspiração e me incentivaram sempre a dar o meu melhor.

Ao meu avô, a pessoa mais querida e a estrela mais bonita do céu. Que me faz querer ser uma pessoa melhor todos os dias.

À minha sobrinha, que me ensinou a olhar para a vida de outra forma. Que com o seu sorriso contagiante se tornou a minha maior motivação.

Aos amigos que fiz ao longo destes anos. Quero agradecer por todas as experiências que vivemos. Por me ensinarem tanto. Por estarem sempre lá.

A toda a gente, que me ensinou e acreditou em mim.

Finalmente, um obrigada ao Professor António Luís Sousa, cuja orientação e disponibilidade tornaram este projeto possível.

ABSTRACT

Computing infrastructure management is increasingly demanding and has to comply with regulatory requirements. To comply with these requirements, the existence of a *Configuration Management Database (CMDB)* is fundamental. One of the challenges that any team has when starting *IT Service Management (ITSM)* is to create the organization's CMDB.

CMDB is a database that stores information about the components, usually called *Configuration Items* (*CIs*), of the infrastructure and the relationships between them. Thus, the CMDB creation implies discovering information about the infrastructure, saving it in the CMDB chosen by the organization.

This dissertation presents a tool for the automatic creation of a CMDB that uses automatic discovery, mapping, and population mechanisms, to find information about the infrastructure components and store these results in the CMDB. It was also necessary to adapt the populate operation according to the database structure of the selected CMDB.

This tool uses several discovery mechanisms to explore different types of *Configuration Items (CIs)*, discovering information about them and their dependencies. It also uses an automatic mapping mechanism to adapt the types of discovered data with the CMDB structure where they will be stored. Finally, it populates the CMDB using its *Application Programming Interface (API)* to create the CIs and relationships.

Keywords: CMDB, CI, automatic discovery, mapping, API.

RESUMO

A gestão de infraestruturas computacionais é cada vez mais exigente e tem de cumprir cada vez mais com requisitos normativos. Para estar de acordo com estes requisitos, a existência de uma CMDB é fundamental. Um dos desafios que qualquer equipa tem ao iniciar a gestão de uma infraestrutura é a criação da sua CMDB.

Uma CMDB é uma base de dados que guarda informação acerca dos componentes, usualmente denominados de itens de configuração (CIs), de uma infraestrutura computacional e dos relacionamentos entre si. Assim, a criação de uma CMDB implica descobrir informação acerca dos componentes que fazem parte da infraestrutura e armazenar esta na CMDB escolhida pela organização.

Nesta dissertação é apresentada uma ferramenta de criação automática de uma CMDB que recorre a mecanismos automáticos de descoberta, mapeamento e povoamento, de forma a encontrar informação acerca da infraestrutura, e armazenar estes resultados na CMDB. Tendo em conta que existem produtos de software distintos que implementam CMDBs, foi necessário adaptar o povoamento de acordo com a estrutura da base de dados selecionada.

Esta ferramenta recorre a diversos mecanismos de descoberta de forma a explorar diferentes tipos de componentes, com a finalidade de descobrir informação acerca destes e das dependências entre estes. Recorre também a um mecanismo de mapeamento automático entre modelos de dados, de forma a adaptar os tipos de dados descobertos com a estrutura da CMDB onde estes vão ser armazenados. Finalmente, utiliza um mecanismo para efetuar o povoamento da CMDB que utiliza a API desta para efetuar a criação dos componentes e dos relacionamentos.

Palavras-chave: CMDB, CI, descoberta automática, mapeamento, API.

CONTEÚDO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE LISTAGENS

SIGLAS

ACI Application Centric Infrastructure.
AP Access Point.
API Application Programming Interface.
ARP Address Resolution Protocol.

BD Bases de Dados.
BGP Border Gateway Protocol.
BIOS Basic Input/Output System.

CDM Common Data Model.
CDP Cisco Discovery Protocol.
CI Configuration Item.
CIDR Classless Inter-Domain Routing.
CIM Common Information Model.
CLI Command Line Interface.
CMDB Configuration Management Database.
CMS Configuration Management System.
CPU Central Processing Unit.
CSV Comma-separated values.

DAS Direct Attached Storage.
DMTF Distributed Management Task Force.
DNS Domain Name System.

FDB Forwarding Database.
FDP Foundry Discovery Protocol.
FTP File Transfer Protocol.

GB Gigabyte.
GHz Gigahertz.
GPU Graphics Processing Unit.

HDD Hard Disk Drive.
HTTP HyperText Transfer Protocol.
HTTPS HyperText Transfer Protocol Secure.

ICMP Internet Control Message Protocol.
IEC International Electrotechnical Commission.
IP Internet Protocol.
IPMI Intelligent Platform Management Interface.
ISO International Organization for Standardization.
ITIL Information Technology Infrastructure Library.
ITSM IT Service Management.

JSON JavaScript Object Notation.

LAN Local Area Network.
LLDP Link Layer Discovery Protocol.

MAC Media Access Control.
MIB Management Information Base.
MID Management, Instrumentation, and Discovery.
MIEI Mestrado Integrado em Engenharia Informática.

NAS Network Attached Storage.
NDP Neighbor Discovery Protocol.

OSI Open System Interconnection.
OSPF Open Shortest Path First.

PHP Hypertext Preprocessor.

RAM Random-access memory.
RDF Resource Description Framework.
REST Representational State Transfer.
RPC Remote Procedure Call.

SAN Storage Area Network.
SCCM System Center Configuration Manager.
SGBD Sistemas de Gestão de Base de Dados.
SNMP Simple Network Management Protocol.
SPARQL SPARQL Protocol and RDF Query Language.
SQL Structured Query Language.
SSD Solid-State Drive.
SSH Secure Shell.
STP Spanning Tree Protocol.
SVS Service Value System.

TCP Transmission Control Protocol.

TI Tecnologias da Informação.
TTL Time to Live.

UCS Unified Computing System.
UDP User Datagram Protocol.
UM Universidade do Minho.
UML Unified Modeling Language.
UPS Uninterruptible Power Source.
URL Uniform Resource Locator.
UUID Universally Unique Identifier.

VLAN Virtual Local Area Network.
VPN Virtual Private Network.

WAN Wide Area Netwok.
WinRM Windows Remote Management.
WMI Windows Management Instrumentation.

XML eXtensible Markup Language.

1

INTRODUÇÃO

Este documento descreve o projeto de dissertação desenvolvido no contexto do *Mestrado Integrado em Engenharia Informática (MIEI)* da *Universidade do Minho (UM)*, baseando-se na criação, de forma automática, de uma *Configuration Management Database (CMDB)*.

Neste capítulo vão ser apresentados o contexto, motivação e principais objetivos do problema enunciado, assim como a organização do restante documento.

1.1 CONTEXTUALIZAÇÃO

Os departamentos de *Tecnologias da Informação (TI)* das organizações enfrentam vários desafios na entrega de serviços confiáveis que suportam os objetivos de negócio das organizações. Os desafios de gestão dos serviços de TI podem envolver, por exemplo, o desempenho da rede e dos sistemas, a gestão do ciclo de vida das aplicações, a gestão de ativos e mudanças, a otimização financeira e a segurança dos sistemas [66].

O processo de gestão de configurações procura, de uma forma compreensiva e sistemática, especificar, controlar e fazer o rastreio dos componentes da infraestrutura computacional e todas as alterações feitas a estes. De acordo com a *Information Technology Infrastructure Library (ITIL)* [83], a gestão de configurações cria um modelo da infraestrutura e dos serviços disponíveis, identificando, controlando, guardando e verificando toda a informação relativa aos componentes existentes nessa infraestrutura [92].

A ITIL é uma *framework* que descreve um conjunto de práticas, instruções e recomendações que explica processos, procedimentos e tarefas para gerir os serviços de uma organização [56]. Estes elementos não são específicos de nenhuma organização e podem ser aplicados por qualquer uma. A ITIL não é um *standard*, ou seja, não prescreve as práticas que descreve. As organizações não precisam de implementar cada processo ou seguir a *framework* de forma rigorosa. Cada empresa pode escolher as áreas do seu negócio mais apropriadas para implementar os processos. O foco da ITIL é garantir que os serviços consigam acompanhar as necessidades do negócio, oferecendo às organizações uma oportunidade de aumentar a eficiência, melhorar a produtividade e aumentar o valor comercial.

Enquanto que as práticas ITIL representam apenas recomendações, existe, desde 2005, o *standard ISO* 20000 [15], definido pela *International Organization for Standardization (ISO)*, que especifica os requisitos necessários para a gestão de serviços de uma organização. O ISO 20000 é uma evolução das práticas ITIL e tem a intenção de ser compatível com estas [92].

Entre outras coisas, este *standard* implica a utilização de uma CMDB que guarde todos os itens de configuração, assim como o seu histórico de alterações e problemas.

CMDB é, então, uma base de dados, que segue as normas definidas pela ITIL, usada por uma organização para guardar informação sobre a sua infraestrutura computacional.

Uma das maiores tarefas da gestão de configurações passa pela identificação dos itens de configuração a guardar na CMDB. Um dos seus principais objetivos passa por fornecer informação sobre os itens e configurações na infraestrutura da empresa e os seus serviços.

1.2 Motivação

Os ambientes empresariais possuem infraestruturas computacionais complexas que são compostas por diversos componentes, que podem incluir elementos de hardware, software, documentação ou pessoal, e que dependem entre si de várias formas. Tendo em conta a grande complexidade da infraestrutura de uma empresa, esta necessita de ferramentas de gestão mais sofisticadas, tal como uma CMDB [78]. Sendo uma ferramenta que permite a uma organização ter conhecimento de todos os seus ativos e dos relacionamentos entre estes, é descrita em [54] como "um dos recursos mais valiosos que uma organização tem à sua disposição, se for executada corretamente".

A CMDB torna-se uma ferramenta mais relevante à medida que são adicionados mais itens de configuração ao sistema, porque permite manter informação sobre estes e perceber como alterações aos mesmos podem afetar os restantes componentes da infraestrutura. Para além disto, à medida que a infraestrutura se torna mais complexa, aumenta, também, a importância de manter informação sobre os seus componentes, de forma a facilitar a sua gestão.

Existem diversas razões pelas quais é relevante ter informação sobre a infraestrutura da organização. Por exemplo, conhecer todos as máquinas existentes facilita a prevenção da entrada de intrusos na rede da empresa; conhecer as dependências entre os vários componentes permite saber quais destes possuem acesso a informação sensível; saber de todas as instâncias instaladas de um *software* permite fazer uma melhor gestão do licenciamento do mesmo; conhecer todos os ativos envolvidos numa tarefa e todos os seus relacionamentos, fornece um melhor planeamento de ações de recuperação e de análise relacionadas com a disponibilidade de serviços; o conhecimento detalhado dos componentes de *software* e de *hardware*, assim como as dependências entre si, permitem a virtualização e migração de serviços e máquinas; entre outros [78].

No entanto, obter toda esta informação sobre os componentes da infraestrutura implica custos, principalmente se tiver de ser efetuada manualmente. Considerando as empresas já existentes, que já possuem uma estrutura mais complexa e desenvolvida, este processo torna-se ainda mais difícil. Desta forma foi observado em [73] que das várias organizações que a cada ano tentam construir a sua CMDB, cerca de 85% falham nesta tarefa. Para além disto, é também constatado em [66] que mais de 75% de todas as iniciativas estratégicas, sendo estas a implementação de uma CMDB ou outras, falham, pelo menos, na resposta às expetativas iniciais das organizações. Uma das maiores razões pela qual isto acontece recai no facto de ser uma abordagem excessivamente manual e, sem as ferramentas apropriadas, as organizações consideram que este trabalho é demasiado minucioso e demorado [73]. Adicionalmente, a tentativa de modelar manualmente os diferentes relacionamentos entre os CIs, é uma atividade propensa a erros. Uma boa abordagem para resolver este problema passa por obter um método que recolha informação sobre os componentes.

As ferramentas de descoberta automática são essenciais para a automatização do processo de recolha de informação para uma CMDB. Porém, usualmente, estas ferramentas são difíceis de implementar e executar e não recolhem a quantidade e tipo de dados pretendidos [78]. Assim, quem precisa destas informações acaba por implementar as suas próprias ferramentas ou realizar a descoberta manualmente. Além disto, a descoberta das dependências relacionadas com pessoas e de informação relacionada com *software à medida* torna-se uma etapa crítica, devido à especificidade deste tipo de dados. A precisão da informação obtida é também um fator essencial, pois não devem existir ativos ou dependências não descobertas. A falta de informação pode levar a que, por exemplo, processos de recuperação ou de migração acabem por falhar [78].

Assim, o processo de descoberta deve ser o mais simples, abrangente e preciso possível, de forma a que a informação obtida seja o mais útil possível na tomada de decisões acerca da infraestrutura, uma vez que estas influenciam os resultados da organização.

1.3 OBJETIVOS

Com este projeto, é esperado que seja possível fazer a criação de uma CMDB automaticamente, ou seja, com o mínimo de trabalho desempenhado por parte do utilizador.

Desta forma, e de um modo geral, os principais objetivos deste projeto passam então por:

- Perceber como funcionam as tecnologias que implementam CMDBs e como é que os dados recolhidos podem ser armazenados;
- Desenvolver uma arquitetura para a automatização do processo de criação de uma CMDB;

- Implementar um processo de descoberta automática dos componentes computacionais de uma organização recorrendo a ferramentas de análise de redes e inventário de sistemas;
- Fazer o povoamento da CMDB, adaptando-o à ferramenta escolhida pelo utilizador;
- Minimizar o esforço do utilizador na tarefa de criação da CMDB.

1.4 ORGANIZAÇÃO DO DOCUMENTO

Além deste capítulo de introdução, este documento encontra-se estruturado em mais 5 capítulos:

- **Capítulo 2 - Estado da Arte:** são expostos os conceitos e implementações mais relevantes acerca do tema.
- **Capítulo 3 - Problema e Desafios:** é apresentada a solução proposta para o problema identificado, assim como os desafios que possam surgir.
- **Capítulo 4 - Desenvolvimento:** aborda o desenvolvimento e implementação da solução, expondo as decisões tomadas.
- **Capítulo 5 - Testes:** apresenta os testes efetuados à ferramenta desenvolvida, assim como os resultados obtidos nestes.
- **Capítulo 6 - Conclusões e Trabalho Futuro:** conclui a dissertação, apresentando um resumo dos resultados obtidos, assim como uma perspetiva de melhoramento do projeto a abordar num possível trabalho futuro.

2

ESTADO DA ARTE

Neste capítulo vão ser apresentados os conceitos e implementações existentes relevantes para a documentação e gestão de infraestruturas computacionais. Isto inclui a apresentação de conceitos, tais como ITIL, ISO 20000, infraestrutura computacional, CMDB, descoberta automática e mapeamento entre modelos de dados, assim como experiências e conhecimentos já fundamentados acerca destes. Finalmente, será apresentado algum do trabalho relacionado analisado.

2.1 ITIL

À medida que as infraestruturas se foram tornando mais complexas, durante os anos 80, o governo britânico examinou a sua infraestrutura de TI com o objetivo de encontrar melhores formas de usar os seus serviços e recursos. Surgiu então um conjunto de padrões e melhores práticas para melhorar o desempenho das TI que evoluiu para uma *framework* composta por um conjunto de livros [66] - a ITIL [83] - que se encontra, actualmente, na versão 4.

Um dos componentes chave da ITIL 4 é o *Service Value System (SVS)*. Neste, é representada a forma como componentes e atividades diferentes podem trabalhar em conjunto, em qualquer tipo de organização, com o objetivo de facilitar a criação de valor através de serviços de TI. Os componentes do SVS são:

- **Cadeia de valor de serviço:** Conjunto de atividades realizadas por uma organização para entregar produtos ou serviços com valor aos seus clientes;
- **Práticas:** Conjunto de recursos organizacionais desenhados para executar um trabalho ou atingir um objetivo;
- **Princípios orientadores:** Recomendações capazes de orientar uma organização em todas as circunstâncias, independentemente dos seus objetivos, estratégias, tipo de trabalho ou estrutura de gestão;
- **Governação:** Sistema que que orienta e controla a organização;

- **Melhoria contínua:** Atividade organizacional recorrente que garante que o desempenho da organização continua a responder às expectativas das partes interessadas.

As práticas de gestão definidas no SVS incluem práticas gerais de gestão, práticas de gestão de serviços e práticas de gestão técnica. De todas as práticas definidas, a relevante para este projeto é a prática de gestão de configurações de serviços, inserida nas práticas de gestão de serviços. A prática de gestão configurações de serviços garante que a informação sobre os serviços e os CIs que os suportam é exata, confiável e está disponível quando e onde for precisa. Isto inclui informação sobre como os CIs estão configurados e como se relacionam entre si. Recolhe e gere informação sobre uma grande variedade de CIs, que usualmente incluem hardware, software, redes, edifícios, pessoas, fornecedores, documentação e serviços. Esta informação pode ser guardada numa única fonte de dados, ou pode ser distribuída entre várias fontes. Inicialmente, a ITIL começou por descrever uma CMDB como uma base de dados única, mas a partir da última versão, começou a utilizar o conceito de *Configuration Management System (CMS)* para descrever uma ferramenta que recolhe, armazena, gere, atualiza, analisa e apresenta toda a informação sobre os CIs e os seus relacionamentos.

Ainda assim, ITIL define uma CMDB como "uma base de dados utilizada para guardar os detalhes dos CIs durante o seu ciclo de vida e os relacionamentos entre estes." Outros conceitos relevantes no contexto de uma CMDB são infraestrutura de TI, gestão de configurações, item de configuração (CI), atributo e tipo de CI. A infraestrutura de TI é descrita pela ITIL como "todo o hardware, software, redes e instalações necessárias para desenvolver, testar, entregar, monitorizar, gerir e suportar serviços". A gestão de configurações "fornece informação sobre os CIs (...) e os seus relacionamentos: como interagem, se relacionam, e dependem uns dos outros (...). Isto inclui informação sobre as dependências entre serviços". Um CI é "qualquer componente que necessita de ser gerido para fornecer serviços aos clientes" e pode abranger elementos de software, hardware, serviços, processos ou pessoas [83]. O atributo de um CI é essencialmente informação sobre este, tal como, preço, versão, nome ou número de série [66]. Já o tipo de um CI corresponde a "uma categoria usada para classificar CIs. Os tipos de CIs identificam os atributos e relacionamentos desse registo". Definir, gerir e otimizar os CIs e as suas dependências é o foco da CMDB.

2.2 ISO/IEC 20000

A *International Organization for Standardization (ISO)* [3] é uma organização internacional de definição de *standards* fundada em 1947 que já desenvolveu mais de 20000 *standards* internacionais.

O ISO/*IEC* 20000 [15] é um *standard* cujos processos estão alinhados com as práticas descritas na ITIL [83]. Define os requisitos de gestão dos serviços de TI necessários para que as organizações prestem serviços de qualidade aos seus clientes.

O ISO/*IEC* 20000 possui 5 grupos de processos que garantem a qualidade da prestação dos serviços por parte das organizações [88]:

- Conceção e transição de serviços novos ou alterados;
- Processos de entrega de serviços;
- Processos de relacionamento;
- Processos de resolução;
- Processos de controlo.

Incluído no grupo de processos de controlo, a gestão de configurações relata que, para cada serviço, devem ser identificados os CIs relevantes, os seus atributos e os seus relacionamentos e dependências [88]. Neste contexto, são relevantes os conceitos item de configuração e CMDB. CI é descrito pelo ISO/*IEC* 20000 como um "elemento que necessita de ser controlado para entregar serviços" e CMDB é especificada como uma base de dados que "guarda informação sobre os atributos dos itens de configuração e dos relacionamentos entre estes, controlando os componentes durante o seu ciclo de vida" [15].

Devido ao foco na qualidade e eficiência dos serviços prestados, as organizações que seguem os princípios e práticas de gestão de serviços do ISO/*IEC* 20000 tiram partido de benefícios tanto a nível interno como a nível competitivo, visto que uma certificação deste tipo não é algo simples e comum.

2.3 Infraestrutura de TI

O modelo de uma infraestrutura pode englobar vários conceitos dependendo do contexto da situação. No entanto, de uma forma generalizada, o modelo descrito em [82] parece adequado neste contexto em específico. Segundo este, a infraestrutura computacional de uma organização pode ser agrupada nas seguintes categorias:

- dispositivos dos utilizadores finais;
- sistemas operativos;
- armazenamento;
- computacional;
- rede;
- centros de dados.

Os centros de dados são os locais que albergam a maior parte dos componentes de hardware da infraestrutura. Para além disto, fornecem os recursos necessários para que estas máquinas funcionem corretamente, tais como, energia, arrefecimento, sistemas de deteção e prevenção de incêndios, entre outros [82]. Para além destes, os CIs da empresa podem também estar localizados em, por exemplo, escritórios.

A arquitetura de rede é baseada no modelo OSI e pode estar organizada de diversas formas e possuir vários componentes. Podem tratar-se, por exemplo, de redes *Wide Area Netwok (WAN)* [57, Capítulo 8.2.3.4], *Local Area Network (LAN)* [57, Capítulo 8.1] ou *Virtual Private Network (VPN)* [57, Capítulo 8.2.6.1]. Podem possuir diversos componentes que as conectam, tais como routers, switches, firewalls, hubs, modems, pontes ou *Access Points (APs)* [82].

Tabela 2.1: Camadas do modelo OSI.

Camada	OSI
7	Aplicação
6	Apresentação
5	Sessão
4	Transporte
3	Rede
2	Ligação de Dados
1	Física

O modelo OSI foi desenvolvido pela ISO [3] no final dos anos 70 e definido no *standard* ISO/IEC 7498 [8]. Na sua arquitetura estão definidas 7 camadas, que são apresentadas na Tabela 2.1. As duas camadas mais relevantes neste contexto são a camada de rede e a camada de ligação de dados [57].

A camada de rede é responsável por ter informação sobre a disposição da rede. Esta utiliza um endereço lógico - *Internet Protocol (IP)* - que é atribuído aos dispositivos, para perceber onde estes se encontram na rede [57]. Usualmente são denominados dispositivos *layer* 3, porque operam na camada de rede [57]. Para além de um dispositivo poder apresentar múltiplos endereços IP, estes podem também ser alterados, sendo necessário verificar se dois, ou mais, IPs não pertencem ao mesmo dispositivo [84]. Os protocolos IPv4 e IPv6 e o seu encaminhamento e endereçamento são implementações desta camada [82].

A camada de ligação de dados utiliza um endereço físico - *Media Access Control (MAC)* - para conduzir o tráfego na rede [85]. Este endereço, apesar de poder ser alterado na máquina, não é alterado tão frequentemente, nem automaticamente, como o endereço IP [84]. Usualmente os dispositivos que operam na camada de ligação de dados são denominados dispositivos *layer* 2 [57]. Alguns exemplos de implementações nesta camada são a Ethernet, Wi-Fi e WANs [82].

A nível computacional, são considerados os computadores e servidores localizados no centro de dados, que podem ser máquinas físicas ou virtuais. As máquinas físicas possuem fontes de energia, *Central Processing Units (CPUs)*, *Basic Input/Output System (BIOS)*, memória, portas, conectividade com a rede e, opcionalmente, periféricos. Usualmente estas máquinas possuem um papel específico dependendo das tecnologias que implementam e dos serviços que fornecem, podendo ser, por exemplo, servidores de ficheiros, web, aplicacionais, de bases

de dados, e-mail ou de virtualização. Para além das suas máquinas físicas, a infraestrutura pode também recorrer a plataformas *cloud* para executar os seus processos.

As instâncias dos produtos de software *container*, usualmente geridos por soluções de gestão de *containers*, são unidades de software que permitem a execução de aplicações.

Em relação ao armazenamento, as máquinas podem utilizar armazenamento interno e/ou armazenamento externo. São exemplos os discos *Hard Disk Drives (HDDs)* e *Solid-State Drives (SSDs)*, e os sistemas *Direct Attached Storage (DAS)*, *Network Attached Storage (NAS)* e *Storage Area Network (SAN)*. A documentação, que se encontra armazenada nos sistemas de armazenamento, pode incluir informação acerca de políticas, licenças ou manuais de utilização e de resolução de problemas.

Os sistemas operativos fornecem uma camada abstrata entre o hardware e o software [82]. São um conjunto de programas que gerem o funcionamento interno do computador - memória, processos, sistemas de ficheiros, periféricos - e controlam os programas deste [94].

Os utilizadores representam os ativos humanos de uma organização, que exercem funções, possuem habilidades e certificados e utilizam serviços. Tendo em conta que representam os recursos mais dispendiosos da organização, a informação acerca de quem são, o que fazem, como podem ser localizados e que serviços lhes estão relacionados torna-se importante [89]. Os utilizadores interagem com as aplicações através de dispositivos, tipicamente portáteis, computadores, *tablets*, telemóveis e impressoras.

Usualmente as máquinas da infraestrutura utilizam produtos de software, que englobam os programas e aplicações instalados nas máquinas. Neste caso em concreto é necessário perceber que nem todos os programas instalados numa máquina representam um CI, isto porque existem programas instalados automaticamente nos dispositivos, como por exemplo jogos, que se encontram automaticamente instalados em portáteis [92]. Para além disto, usualmente são também utilizados *Sistemas de Gestão de Base de Dadoss (SGBDs)*, que incluem todos os motores de bases de dados que estão a ser executados nas máquinas, assim como, todas as instâncias de *Bases de Dados (BD)* existentes.

2.4 Configuration Management Database

Nesta secção vão ser abordadas ideias e noções relativas ao conceito de *Configuration Management Database (CMDB)*.

2.4.1 *Conceitos*

Uma CMDB é uma base de dados utilizada por uma organização para guardar toda a informação relevante dos componentes da sua infraestrutura e relacionamentos entre estes.

Deve conter informação sobre todos os elementos importantes do negócio, fornecendo uma vista completa da interação entre estes [56].

Sugerido por Thomas Schaaf e Boran Gögetap [81] [106], está representado na Figura 2.1, um modelo concetual de uma CMDB, baseado no diagrama apresentado em [106]. Este mostra, visualmente, a definição de CMDB, e permite obter um melhor entendimento acerca dos conceitos básicos associados a esta, sendo estes o conceito de CI, relacionamento e tipo, a estes associado. Cada CI e cada relacionamento entre dois CIs tem um tipo associado. Este modelo não apresenta implicações a nível de implementação, funcionando como base para modelos mais complexos.

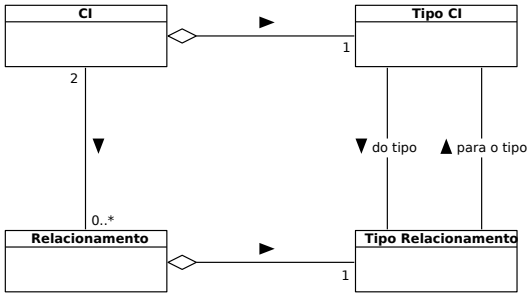


Figura 2.1: Modelo concetual da CMDB.

Este tipo de base de dados permite guardar informação sobre as alterações feitas na infraestrutura, percebendo qual o impacto dessas mudanças no sistema [67] [66]. Desta forma, a utilização de uma CMDB torna-se relevante porque permite uma melhor gestão dos itens de configuração da organização. No entanto, é importante que os dados guardados na CMDB sejam exatos, porque informação errada pode levar a más decisões e, consequentemente, à obtenção de maus resultados [66].

Uma CMDB deve, portanto, possuir diversas características, tais como:

- identificar unicamente cada CI;
- guardar o estado da configuração do sistema;
- guardar todos os detalhes relevantes dos CIs que fazem parte da infraestrutura da organização;
- descrever dependências e relacionamentos entre CIs;
- descrever os serviços oferecidos pelo sistema;
- servir de catálogo de serviços;
- servir de inventário dos CIs;
- guardar informação sobre as alterações feitas na infraestrutura.

2.4.2 *Vantagens*

Sendo uma base de dados que guarda informação sobre toda a infraestrutura, traz benefícios para quem a utiliza. Assim, a implementação de uma CMDB possui várias vantagens para uma organização, tais como [80] [66]:

- permitir o controlo da infraestrutura computacional e do ciclo de vida dos itens de configuração;
- permitir o controlo das alterações feitas no sistema;
- diminuir o tempo de resolução de problemas;
- visualizar a informação de cada componente individual da infraestrutura;
- prevenir erros de manutenção;
- diminuir custos, eliminando recursos redundantes ou desnecessários;
- melhorar a gestão da ocorrência de incidentes;
- avaliar antecipadamente potenciais riscos de mudanças;
- evidenciar vulnerabilidades representando as dependências entre os itens de configuração;
- ajudar na tomada correta de decisões acerca dos componentes da infraestrutura;
- gerir mais facilmente infraestruturas complexas.

2.4.3 *Dificuldades*

Como já foi referido na Secção 1.2, cerca de 75% de todas as iniciativas estratégicas, sendo estas a implementação de uma CMDB ou outras, falham, pelo menos, na resposta às expetativas iniciais [66], e cerca de 85% das iniciativas de implementação de uma CMDB tendem a falhar [73].

O processo de implementação de uma CMDB numa organização acaba por ser um processo complexo, que envolve a execução de várias tarefas. Pode incluir, por exemplo, a definição de objetivos que se espera atingir, a perceção da utilização e disponibilidade dos recursos da organização, a definição de requisitos e arquiteturas, a seleção de tecnologias e sua implementação, resolução de problemas, revisões e avaliações, entre outras [66]. Desta forma, acaba por ser um processo que possui muitos pontos críticos que podem levar a falhas, como por exemplo [66] [73]:

- existência de outras prioridades na organização leva a que o processo não receba a atenção necessária;

- recursos, de pessoal e financeiros, limitados;
- falta de comunicação que leva à perceção de expetativas irrealistas;
- não planeamento dos tipos, atributos e relacionamentos que se quer manter, leva à tentativa insustentável de manter demasiada informação;
- utilização de uma abordagem maioritariamente manual acaba por ser insustentável tendo em conta todo o processo de criação e manutenção da CMDB;
- informação desatualizada ou incorreta leva à incoerência da CMDB;
- falta de capacidade de integração com outras ferramentas;
- capacidades de reconciliação e normalização fracas, levando à incoerência da informação;
- falta de utilização de ferramentas de inventário e descoberta que são capazes de automatizar o processo.

2.4.4 *O que guardar na CMDB*

Tendo em conta que os CIs podem variar bastante em termos de complexidade, tamanho e tipo, no processo de construção da CMDB devem ser tomadas decisões relativamente ao nível de detalhe de cada CI. Estas decisões devem ter em conta os objetivos da organização e os serviços que esta deve entregar aos seus clientes. Esta definição do nível de detalhe a guardar é bastante relevante, porque manter um grande volume de dados sobre os componentes da infraestrutura e os seus relacionamentos pode ter custos bastante elevados [83]. Também é necessário perceber que não é viável encontrar e documentar cada um dos componentes da infraestrutura. Isto, porque o trabalho necessário para manter a informação atualizada seria bastante dispendioso [66].

Assim, os dados a guardar na CMDB podem ser definidos segundo três parâmetros: âmbito, alcance e granularidade [80].

O âmbito é definido pelos tipos de objetos e relacionamentos que vão ser incluídos. Basicamente, de entre os tipos existentes de CIs e relacionamentos, são escolhidos aqueles que são importantes para a organização. Esta escolha pode englobar um conjunto de componentes de alto nível, ou, caso seja relevante para a organização fazer o rastreio de determinados itens, pode abranger um conjunto de elementos mais específicos, ou seja, de mais baixo nível [94]. Por exemplo, pensando numa infraestrutura, pode ser fácil definir algumas categorias amplas tais como, hardware, software e documentos. Dentro da categoria hardware, de um modo mais específico, podemos incluir servidores e equipamentos de rede. Na categoria software podem ser definidos, por exemplo, sistemas operativos, aplicações e sistemas de gestão de bases de dados. Já os tipos de relacionamentos não são tão facilmente

identificáveis, sendo, na maior parte dos casos, definidos tendo em conta os tipos de CIs a ser identificados. Assim como existem alguns tipos que as organizações fazem questão de manter informação, podem existir outros que estas pretendam excluir, com o principal intuito de não incluir objetos desnecessários, de modo a diminuir os custos, não só da descoberta, como também de manutenção da CMDB [80].

O alcance indica, de entre os tipos de CIs a ser descobertos, os grupos específicos a ser incluídos para cada um. Isto porque as organizações podem não querer identificar todos os CIs de um determinado tipo, mas apenas alguns específicos desse tipo [80].

A granularidade determina o conjunto de atributos a descobrir sobre cada tipo de CI podendo esta ser fixa ou variável para cada tipo de CI. A granularidade fixa implica que todos os tipos de CIs possuam o mesmo conjunto de atributos. Este conjunto deve ser definido de forma a abranger todos os tipos. Outro conjunto de atributos deve ser definido para todos os tipos de relacionamentos. Estes tipos representam aqueles que foram definidos no âmbito. A vantagem de utilizar um conjunto fixo é a simplificação da recolha dos dados. No entanto, se os CIs não se adaptarem ao modelo, é necessário tomar uma decisão diferente. Uma solução possível passa por atribuir opcionalidade aos atributos. Para além disto, existem algumas variações que tentam atingir alguns benefícios da granularidade variável. Por exemplo, é possível definir atributos que podem ter valores diferentes dependendo da categoria do CI. No entanto, para analisar estes atributos, que usualmente possuem nomes genéricos, é necessário perceber o mapeamento destes campos para as diferentes categorias. A granularidade variável permite definir um conjunto de atributos para cada tipo de CI e de relacionamento, mas esta maior complexidade na definição dos atributos implica um maior custo na recolha dos dados [80].

Assim, de forma a definir o âmbito, alcance e granularidade da CMDB, é necessário perceber todos os tipos de CIs, atributos e relacionamentos que existem e que são passíveis de ser descobertos, assim como perceber a forma como estes se relacionam [81].

2.4.5 *Ferramentas*

Atualmente, existem vários fornecedores no mercado que possuem ferramentas que implementam CMDBs [6] [1] [38] [37] [47]. Neste capítulo, o objetivo é fazer a análise de um conjunto de características de algumas das soluções existentes e, no final, ser capaz de fazer uma comparação destas segundo alguns fatores.

i-doit

O i-doit [2] é uma ferramenta *open source* que segue as normas da ITIL, e permite fazer a gestão dos elementos computacionais da organização, sendo a informação acerca destes armazenada numa base de dados MySQL [30] ou MariaDB [29].

Relativamente ao modelo de dados do sistema, existe uma hierarquia definida entre diversos tipos de dados, que é possível observar na Figura 2.2. Tipos de grupos de objetos compreendem tipos de objetos; tipos de objetos herdam categorias; tipos de objetos instanciam objetos, que herdam as categorias do seu tipo de objeto; e categorias incluem atributos e podem ser caracterizadas como globais, específicas ou personalizadas [12].

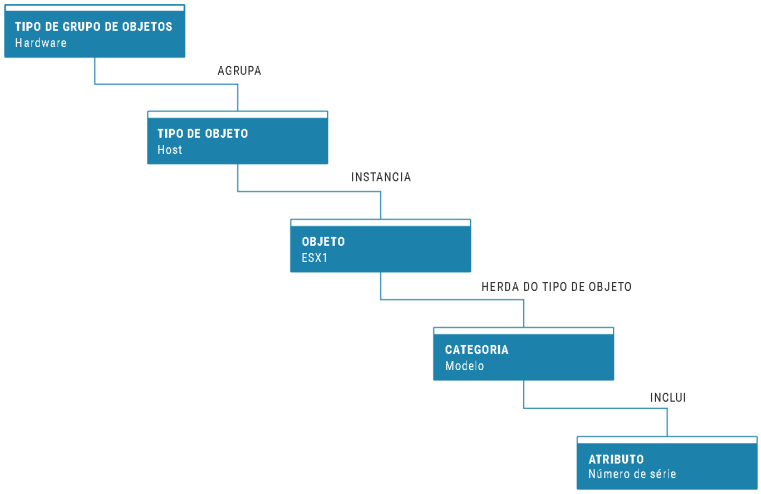


Figura 2.2: Hierarquia do modelo de dados do sistema do i-doit (inspirado no esquema presente em https://kb.i-doit.com/display/en/Structure+of+the+IT+Documentation).

Aqui, o termo objeto refere-se ao termo CI da ITIL, e o termo tipo de objeto refere-se a tipo de CI. Além disto, o i-doit possui também vários relacionamentos definidos, que podem ser ajustados e modificados. Um relacionamento estabelece um dos objetos como *master* e outro como *slave* sendo que o segundo está dependente do primeiro.

Em termos de base de dados relacional, estes tipos, categorias e relacionamentos representam tabelas. Os vários tipos de grupos de objetos encontram-se definidos na tabela "*isys_obj_type_group*", os tipos de objetos estão definidos na tabela "*isys_obj_type*", e cada objeto da CMDB é um registo na tabela "*isys_obj*". Um relacionamento é representado por um objeto do tipo "*relation*". Para além disto, cada categoria está refletida na base de dados como uma tabela. Em termos de nomenclatura, existem prefixos e sufixos que identificam categorias e atributos. Por exemplo, todas as tabelas possuem o prefixo "*isys_*". O estado de vida de um componente é documentado na CMDB e é guardado como um atributo de um objeto. Existem estados já definidos, mas é também possível definir outros estados [12].

O i-doit também permite acesso externo através de uma API, que oferece funcionalidades semelhantes às oferecidas pela interface gráfica, com a vantagem de poderem ser automatizadas. Através desta é possível fazer pedidos ao servidor de forma a executar determinados métodos seguindo uma abordagem *Remote Procedure Call (RPC)* [110], sendo utilizados pedidos *HyperText Transfer Protocol (HTTP)* [68] POST e JSON [5] como formato de troca de informação [12].

Dos vários métodos que podem ser utilizados através da API, os mais relevantes neste contexto são:

- cmdb.object.create: permite a criação de objetos;
- cmdb.object_types.read: interroga os tipos de objetos;
- idoit.constants: devolve os tipos de objetos e categorias, globais e específicas, existentes na base de dados;
- cmdb.category_info: devolve os atributos de uma categoria;
- cmdb.object_type_groups.read: devolve os tipos de objetos que estão associados aos tipos de grupos de objetos;
- cmdb.object_type_categories.read: devolve as categorias associadas aos tipos de objetos.

iTop

O iTop [4] é uma solução *open source* que permite que os CIs e os seus relacionamentos sejam geridos numa CMDB sendo a informação acerca destes armazenada numa base de dados MySQL ou MariaDB.

Existem várias classes definidas para os CIs da CMDB que estão agrupadas por módulos, sendo que a utilização de mais módulos significa a existência de mais tipos de CIs na CMDB. Ao nível do modelo de dados, estas classes são representadas por tabelas e os atributos de cada CI são atribuídos dependendo da sua classe, ou seja, são representados pelas colunas da tabela. Os relacionamentos entre os CIs são também representados por tabelas que são identificadas pelo prefixo "*lnk*".

Possui uma interface *Representational State Transfer (REST)* [69, Capítulo 5] que permite uma interação remota com a CMDB. Esta interface utiliza o protocolo HTTP e o formato JSON para a troca de informação. Dos métodos existentes o mais relevante neste contexto é o método core/create que permite a criação de novos objetos na CMDB [27].

Device42

O Device42 [24] é uma solução de software para gestão de infraestruturas de TI. De entre os vários mecanismos que disponibiliza, o mais pertinente neste contexto é a sua CMDB. Esta fornece uma vista centralizada do ambiente computacional, mantendo informação acerca de todos os CIs. Possui um esquema dos CIs e relacionamentos já definido que pode ser extendido pelo utilizador. No topo da hierarquia encontram-se os edifícios, que alojam uma ou mais salas de computadores. Cada sala pode conter um ou mais *racks* ou outros objetos, que são ativos ou dispositivos. Estes dispositivos incluem dispositivos físicos, como servidores ou switches, máquinas virtuais, clusters, entre outros. Os ativos podem

incluir cabos, faxes, monitores, scanners, produtos de software, entre outros. Estão também definidos como tipos de CIs permissões, clientes, vendedores, palavras-passe, modelos de hardware, sistemas operativos, contratos e aplicações.

Possui uma API REST que permite a manipulação dos CIs na CMDB. No entanto, o *Uniform Resource Locator (URL)* [59] de cada pedido varia dependendo do tipo de CI envolvido [19].

BMC CMDB

A BMC CMDB [21] é um dos componentes do BMC Atrium Core e permite a gestão, armazenamento e monitorização da informação relacionada com os componentes da infraestrutura computacional.

A informação acerca dos CIs e dos seus relacionamentos está organizada em *datasets*, que passam por um processo de reconciliação. Quando os CIs e os seus relacionamentos são adicionados à CMDB, cada entrada é atribuída a um *dataset*, que identifica a origem da informação. O catálogo de produtos inclui os nomes e definições normalizados dos produtos de software, hardware e serviços disponíveis na organização. O motor de normalização garante a coerência da informação proveniente de diversas fontes, aplicando a nomenclatura especificada e eliminando CIs duplicados. O motor de reconciliação combina a informação dos vários *datasets* num único, que é depois utilizado pelos consumidores.

A informação está organizada de acordo com o *Common Data Model (CDM)*, que consiste no conjunto de classes de CIs e relacionamentos presentes na CMDB. Cada classe representa uma tabela na base de dados. Este modelo já está pré-definido na CMDB mas pode ser estendido, permitindo a criação de novas classes e atributos destas. Estas classes representam os diferentes tipos de CIs e relacionamentos. Uma instância de um relacionamento relaciona duas instâncias de CIs, podendo também ter atributos associados. Para além disto, um dos CIs é considerado a origem do relacionamento e o outro o destino.

Existem diferentes tipos de classes no CDM:

- **Regulares:** têm a sua própria tabela definida com os seus atributos específicos, podendo ser superclasses ou subclasses, sendo que estas últimas combinam os seus atributos específicos aos da sua superclasse;
- **Categorização:** não têm a sua própria tabela definida, sendo que os atributos específicos de uma classe de categorização são adicionados à estrutura da sua superclasse, sendo estes apenas preenchidos pelas instâncias da classe de categorização, tornando-se opcionais;
- **Abstratas:** não têm a sua própria tabela definida nem atributos, funcionando apenas para organizar subclasses;
- **Abstratas com replicação de dados:** novas tabelas que agregam toda a informação acerca das subclasses de uma classe abstrata.

De forma a manter a consistência com os *standards* definidos neste contexto, o CDM foi baseado no CIM [23] e no *Windows Management Instrumentation (WMI)* [18], tendo sido adotados os componentes básicos destes. O CIM é um modelo de dados orientado a objetos que contém informação sobre diferentes partes de uma empresa e foi definido pela *Distributed Management Task Force (DMTF)* [25]. O WMI [18] é a infraestrutura para a gestão de dados e operações em sistemas operativos baseados em Windows [53], que utiliza também o CIM para representar sistemas, aplicações, redes, dispositivos e outros componentes.

A BMC CMDB [21] possui uma API REST que utiliza o protocolo HTTP e o formato JSON para a troca informação. Dos vários pedidos que podem ser feitos através da API, os mais relevantes neste contexto são:

- GET /cmdb/v1.0/classes: retorna a lista das classes do CDM;
- GET /cmdb/v1.0/attributes/namespace/className: retorna os atributos de uma determinada classe;
- POST /cmdb/v1.0/instances: suporta a criação de múltiplas instâncias.

ServiceNow CMDB

A solução proposta pela ServiceNow [34] permite o armazenamento de informação acerca dos componentes da infraestrutura, construindo uma representação lógica dos ativos e relacionamentos entre estes.

O modelo de dados da CMDB consiste num conjunto de tabelas ligadas que contêm todos os ativos e serviços controlados pela organização, assim como as suas configurações [34].

Existem três tabelas chave na CMDB [34]:

- cmdb**:** tabela base para armazenar CIs não tecnológicos;
- cmdb_ci**:** tabela base para armazenar CIs;
- cmdb_rel_ci**:** tabela que define os relacionamentos entre CIs.

As tabelas podem ser estendidas para outras, atribuindo uma hierarquia de classificação aos vários CIs. É então possível ter tipos de CIs que descendem de outros tipos na hierarquia. Todos os CIs no sistema têm um tipo (classe), que se refere ao nome da tabela em que são armazenados. Também é possível definir novas classes de CIs e relacionamentos. Um relacionamento na CMDB consiste em dois CIs, o "pai" e o "filho", e o tipo do relacionamento. Além disto, o sistema mantém a tabela cmdb_rel_type_suggest que contém os tipos de relacionamentos apropriados para um tipo de CI, com base na sua classe. Os relacionamentos são definidos aos pares, ou seja, o relacionamento existe da perspetiva do CI "pai" e o relacionamento contrário existe da perspetiva do CI "filho" [34].

É possível também aceder aos dados presentes na CMDB através da sua API REST, que utiliza o protocolo HTTP e os formatos JSON ou *eXtensible Markup Language (XML)* [75] para troca de informação, e suporta os seguintes métodos [34]:

- POST /now/cmdb/instance/{classname}: criação de um CI, que pode ter relacionamentos associados, sendo que {classname} se refere ao nome da classe na CMDB;
- GET /now/cmdb/meta/{classname}: para uma determinada classe da CMDB, definida no parâmetro {classname}, retorna a classe que estende, as classes que a estendem, os seus atributos e os relacionamentos que pode integrar.

Comparação das ferramentas

É possível observar na Tabela 2.2 uma comparação entre as ferramentas descritas anteriormente. Esta teve como principais parâmetros de comparação, a forma como as CMDBs armazenam os CIs, relacionamentos e informação acerca destes, as capacidades fornecidas pelas suas APIs e os SGBDs suportados por estas.

Tabela 2.2: Comparação das CMDBs analisadas.

	Características	i-doit	iTop	Device42	BMC CMDB	ServiceNow CMDB
CIs	**Representação**	Registo na tabela isys_obj	Registo na tabela do seu tipo	?	Registo na tabela do seu tipo	Registo na tabela do seu tipo
	Tipo	Definidos na tabela isys_obj_type	Diferentes tabelas representam tipos de CIs diferentes	?	Diferentes tabelas representam tipos de CIs diferentes	Diferentes tabelas representam tipos de CIs diferentes
	Hierarquia	Tipos de grupos de CIs compreendem tipos de CIs, que têm categorias associadas	Não existe	?	Existem sub-tipos que herdam as características do seu super-tipo	Tabelas estendem outras tabelas e herdam as suas características
	Atributos	Categorias incluem atributos	Colunas da tabela	?	Colunas da tabela	Colunas da tabela
Relacionamentos	**Representação**	CI do tipo relation	Tabela com o prefixo lnk	?	Tabela do seu tipo	Tabela do seu tipo
API	**Protocolo**	HTTP	HTTP	HTTP	HTTP	HTTP
	Formato troca	JSON	JSON	JSON	JSON	JSON XML
	Criar CI	Permite	Permite	Permite	Permite	Permite
	Obter tipos de CIs	Permite	Não permite	Não permite	Permite	Não permite
	Obter atributos de um tipo	Permite	Não permite	Não permite	Permite	Permite
SGBD	**Sistemas suportados**	MySQL MariaDB	MySQL MariaDB	PostgreSQL	Microsoft SQL Server Oracle	MariaDB Microsoft SQL Server Oracle

? - significa que não foi possível obter informação concreta

Tendo em conta os parâmetros analisados, é possível verificar a existência de algumas semelhanças e diferenças entre as várias CMDBs.

As principais semelhanças detetadas foram:

- Base nas normas da ITIL;
- Acesso à CMDB através de uma API que utiliza o protocolo HTTP e JSON como formato de troca de informação;
- Possibilidade de criação de novos CIs através da API.

As principais diferenças detetadas foram:

- Existência de uma tabela com a definição dos vários tipos de CIs e relacionamentos em oposição à existência de uma tabela para cada tipo existente;
- Existência, ou não, de uma hierarquia entre os diversos tipos existentes;
- Colunas da tabela representam atributos em oposição à existência de categorias de atributos associadas a tipos;
- Obtenção da informação acerca do modelo de dados da CMDB através da API em oposição a acessos diretos à base de dados;
- Diferentes SGBDs suportados.

2.5 DESCOBERTA AUTOMÁTICA

Nesta secção vão ser abordadas ideias, conceitos e implementações relativos à descoberta automática de componentes de uma infraestrutura de TI. Este processo mostra-se particularmente relevante devido à capacidade de automatizar o processo de recolha de informação acerca da infraestrutura, visto que efetuar este processo manualmente implica um trabalho bastante minucioso, demorado e propenso a erros.

2.5.1 *Processo*

Atualmente, as empresas possuem infraestruturas computacionais complexas, que compreendem uma enorme variedade de componentes e dependências entre si [78], apresentando diferentes tipos de arquiteturas dependendo dos seus requisitos [109]. Os seus componentes estão conectados de forma a executar todas as tarefas que suportam os serviços da organização, estando habitualmente também ligados à rede, que permite a comunicação de dados entre os vários dispositivos [107]. Neste contexto, o conceito de "rede" pode ser definido como um conjunto de dispositivos ligados que trocam informação uns com os outros, podendo utilizar diversos protocolos de comunicação [109]. Estes dispositivos podem também estar localizados numa área comum, ou em localizações geográficas distintas, podendo estar várias redes e sub-redes conectadas umas com as outras [109].

Assim, de modo a fazer a criação automática da CMDB, é necessário descobrir os CIs que existem na infraestrutura e os relacionamentos entre estes. A descoberta da infraestrutura da rede envolve a recolha de dados dos vários dispositivos que existem nessa rede, sendo que cada dispositivo possui um conjunto de informação que o caracteriza [84].

Ao longo do processo de descoberta, e à medida que é obtida nova informação, esta pode ser necessária para obter dados mais específicos [78] [84], ou seja, pode ser necessário saber algo sobre uma máquina para a poder explorar mais detalhadamente. Para além disto, pode também ser necessário fornecer credenciais para aceder a mais informação acerca dos CIs, assim como iniciar as comunicações dentro da firewall da empresa [94] de forma a poder comunicar com os diversos componentes.

2.5.2 *Protocolos*

Atualmente, existem ferramentas no mercado que implementam diversos métodos relacionados com a descoberta de componentes numa infraestrutura. No entanto, a utilização de um único método de descoberta não é suficiente, porque várias ferramentas descobrem conjuntos diferentes de informação [78]. É então necessário perceber como funcionam, que protocolos e métodos utilizam e que tipo de dados é possível obter através destas ferramentas. Outra questão iminente neste processo passa por perceber como é que estes podem ser utilizados e se são suportados pelos dispositivos.

O protocolo *Internet Control Message Protocol (ICMP)* [99] permite descobrir os endereços IP das máquinas ativas na rede. Através do comando *ping* é possível testar a conetividade entre dispositivos. Já o comando *traceroute*, permite descobrir o caminho ente dois nodos [57]. Desta forma é possível perceber como estão conectados os vários dispositivos de rede da infraestrutura. Como necessita do endereço IP, é apenas suportado por dispositivos que implementem a camada 3 do modelo OSI. Em muitas redes, apenas uma pequena percentagem dos endereços IP estão ativos num determinado momento, sendo que este intervalo de endereços pode ser contínuo ou não [109].

As tabelas de *routing* são outro mecanismo que permite obter informação sobre a rede, porque guardam informação sobre a topologia desta. Estas estão guardadas em dispositivos de rede, como por exemplo, routers. Fornecem informação referente aos endereços associados ao dispositivo; as sub-redes a que estes se encontram diretamente conetados; o conjunto de endereços utilizado pelos dispositivos na sua vizinhança; e a explicação de como os pacotes devem ser encaminhados na rede [57].

Também o protocolo *Address Resolution Protocol (ARP)* [98] pode ser utilizado para obter informação acerca do endereçamento na rede. Este mapeia endereços IPv4 em endereços MAC, associando endereços de rede a endereços físicos. Esta informação, quando obtida, é armazenada na cache ARP dos dispositivos. Ao nível dos endereços IPv6 é utilizado o protocolo *Neighbor Discovery Protocol (NDP)* [93], que permite que sejam encontrados outros dispositivos na sua vizinhança. Cada dispositivo que implemente este protocolo possui a sua *neighbor* cache, que possui informação acerca da resolução de endereços IPv6 em endereços MAC e se estes correspondem a um *host* ou a um router [72].

Outro recurso que pode ser utilizado para obter informação sobre a rede é a tabela *Forwarding Database (FDB)*. Esta tabela é utilizada por dispositivos *layer* 2 [57] e mapeia os endereços MAC nas portas em que foram descobertos. Usualmente os *switches* são descobertos através da tabela FDB, pois podem não reconhecer IP.

Para além destes, existe também o protocolo *Link Layer Discovery Protocol (LLDP)* [63] que, ao nível da camada de ligação de dados, facilita a troca de informação entre dispositivos. É utilizado pelos dispositivos de rede para anunciarem a sua identidade, capacidades e os seus vizinhos mais próximos numa LAN [64]. A informação sobre os dispositivos é armazenada em *Management Information Bases (MIBs)* [63], que são conjuntos organizados

de informação que podem ser acedidos utilizando o protocolo *Simple Network Management Protocol (SNMP)* [60].

Também o protocolo *Spanning Tree Protocol (STP)* [97] permite obter informação sobre a topologia da rede [109]. Este protocolo constrói uma *spanning tree* que caracteriza o relacionamento entre os nodos da rede, conectada com pontes. Para obter a topologia da rede na camada de ligação de dados, é necessário analisar os dados da *spanning tree* de cada ponte, usando, por exemplo, o protocolo SNMP [57].

O protocolo SNMP utiliza pedidos *User Datagram Protocol (UDP)* para aceder aos dados armazenados nas MIBs [57] dos dispositivos, tais como routers, *switches*, servidores, impressoras, *workstations*, entre outros. A informação é tratada na forma de variáveis e descreve o estado do sistema e a sua configuração. Existem pedidos definidos exclusivamente para obter informação sobre os dispositivos:

- ***GetRequest*:** pedido feito para obter o valor de uma variável ou de uma lista de variáveis;
- ***GetNextRequest*:** tem como objetivo descobrir as variáveis existentes e os seus valores, sendo possível percorrer toda a MIB;
- ***GetBulkRequest*:** pedido de múltiplas iterações de *GetNextRequest*.

Existem três versões principais deste protocolo: SNMPv1 [61], SNMPv2c [101] e SNMPv3 [62]. Nas versões SNMPv1 e SNMPv2c a autenticação é feita recorrendo a uma *community string*, que representa uma palavra-passe necessária para a troca de informação entre os dispositivos. Na versão SNMPv3, os dispositivos são configurados com um nome de utilizador e com uma palavra-passe, sendo que, para os pedidos serem autenticados é necessário que os dispositivos tenham conhecimento das palavras-passe associadas aos nomes de utilizador. Para além disto, para a encriptação dos pedidos, é também necessário que os dispositivos tenham conhecimento da palavra-passe associada ao nome de utilizador.

No entanto, este protocolo também apresenta algumas desvantagens. Este implica a configuração de um agente SNMP na máquina; está suscetível à perda de informação devido à utilização do protocolo UDP; e implica a transferência de grandes quantidades de dados entre máquinas [60].

É apresentado em [95] uma proposta de descoberta da topologia da rede utilizando o protocolo SNMP. O termo topologia é definido como "a combinação de ligações e nodos numa rede e as interconexões entre os nodos". O processo de descoberta descrito está dividido em três módulos, sendo estes a descoberta de dispositivos, a descoberta do tipo de dispositivos e a descoberta de conetividade entre estes. Inicialmente, com base num conjunto de endereços IP e uma ou mais *community strings*, é utilizada a tabela de roteamento, a cache ARP e o protocolo ICMP para encontrar dispositivos na rede, tornando o mecanismo de descoberta num processo recursivo. Para cada dispositivo encontrado, é descoberto o seu tipo, que é obtido tendo em conta o valor guardado em *sysServices*[1]. O algoritmo de descoberta do tipo de dispositivo verifica que camadas do modelo OSI são suportadas pela máquina, determinando de que tipo de dispositivo se trata. Finalmente, dependendo do tipo, é recolhida informação específica de forma a encontrar conetividade entre os vários dispositivos.

O protocolo *Secure Shell (SSH)* [113] permite estabelecer uma conexão com um sistema. A partir deste tipo de conexão é possível ter acesso à máquina e obter informação mais

1 http://oid-info.com/get/1.3.6.1.2.1.1.7

específica sobre a mesma [94]. No caso de sistemas Microsoft Windows, pode ser utilizado o WMI, que permite a gestão e acesso de dados nestes sistemas. Permite também que este acesso seja feito de forma remota recorrendo ao *Windows Remote Management (WinRM)* [17]. Tanto o protocolo SSH como o WinRM necessitam das credenciais das máquinas para acederem a estas.

2.5.3 *Arquiteturas*

Alguns autores publicaram as suas implementações de processos de descoberta automática num ambiente de rede [109] [107] [94].

É disso exemplo o sistema apresentado em [109], que tem como propósito descobrir vários dispositivos num ambiente de rede e as ligações entre si. Este é composto por:

- **Mecanismo de descoberta:** descobre os dispositivos presentes na rede e as ligações entre estes, determinando a topologia da infraestrutura de rede (através do comando *ping*, da consulta da informação das MIBs, via protocolo SNMP, e do protocolo STP);
- **Monitor de rede:** monitoriza os dispositivos para detetar alterações na rede;
- **Interface de rede:** permite ao mecanismo de descoberta comunicar com os dispositivos dentro da rede;
- **Base de dados da rede:** armazena a informação recolhida pelo mecanismo de descoberta;
- **Dispositivo de entrada:** permite que o utilizador envie informação ao sistema;
- **Dispositivo de exibição:** permite ao utilizador uma visualização da topologia da rede.

Este sistema apresenta também uma interface com o utilizador que o permite configurar a descoberta, iniciá-la, apresentar resultados e informação sobre o estado da mesma e filtrar resultados indesejados.

Outro exemplo é o processo de descoberta apresentado em [107]. Este é composto por:

- **Unidade de descoberta:** responsável por descobrir os componentes do sistema e a sua informação de configuração (utiliza o comando *ping* para detetar que endereços IP estão ativos na rede);
- **Base de dados:** mantém um repositório central dos componentes do sistema descobertos;
- **Mapa da rede:** fornece uma vista hierárquica do sistema e das propriedades dos componentes que fazem parte deste.

2.5.4 *Ferramentas*

Atualmente, existem ferramentas de descoberta que utilizam alguns dos protocolos descritos anteriormente para fazer a descoberta e exploração dos componentes da rede. No entanto, estas podem não responder às necessidades dos utilizadores, sendo muitas vezes necessário utilizar várias ferramentas ou proceder à descoberta manual para obter os dados pretendidos [78].

Nmap

O Nmap [86] é uma ferramenta *open source* que permite a exploração e inventário da rede, sendo suportado pela maioria dos sistemas operativos. Esta não necessita das credenciais de acesso de cada máquina para efetuar a descoberta destas na rede [94]. Permite fazer o *scan* de uma rede inteira ou de apenas um *host* [86].

O *scan* efetuado começa por explorar os *hosts* especificados pelo utilizador, que podem ser uma combinação de nomes *Domain Name System (DNS)* [90], endereços IP ou notações de rede *Classless Inter-Domain Routing (CIDR)* [70], verificando se estes se encontram ativos na rede, recorrendo a pedidos ARP, *Transmission Control Protocol (TCP)* ou ICMP. Depois de saber que *hosts* se encontram ativos, é efetuada a resolução de nomes DNS. É também verificado o estado das portas do *host*, se estas se encontram abertas, fechadas, filtradas e não-filtradas e que software, e respetivas versões, está a ser executado nestas. Além disto, o *scan* é também capaz de descobrir o sistema operativo que está a ser executado no *host* e, através do comando `traceroute` permite também descobrir os caminhos na rede entre vários *hosts*. Adicionalmente, é possível recorrer a *scripts* para obter ainda mais informação sobre os *hosts* (deteção de versões avançada, notificação de vulnerabilidades de serviços, descoberta de *backdoors* e outro *malware*, entre outras coisas) [86].

Assim, o Nmap é capaz de [76]:

- detetar os *hosts* ativos na rede;
- detetar o nome e tipo do *host*;
- identificar os endereços IP e MAC;
- explorar as portas abertas, fechadas, filtradas e não-filtradas no *host*;
- detetar serviços (protocolos utilizados, nome da aplicação e versão) a correr nas portas abertas;
- detetar sistemas operativos e respetivas versões.

Angry IP Scanner

O Angry IP Scanner [79] é uma ferramenta *open-source* que funciona em sistemas operativos Windows, macOS [42] e Linux [41], e permite fazer o *scan* da rede.

Usualmente, o utilizador fornece o intervalo de endereços IP com o objetivo de o *scan* verificar se cada um destes se trata de um *host* ativo e obtém informação sobre os mesmos. A informação recolhida pode incluir:

- se o *host* está ativo;
- nome do *host* e do domínio;
- endereços IP e MAC;
- portas TCP e UDP abertas e filtradas;
- serviços a correr e respetivas versões;
- o valor médio do *roundtrip time*;

- o valor do *Time to Live (TTL)*, que permite descobrir a distância do *host* em termos de números de *routers* que o pacote atravessou.

Permite a exportação dos resultados em diversos formatos, nomeadamente texto, CSV ou XML.

LibreNMS

O LibreNMS [28] é uma ferramenta *open source* de monitorização da rede baseado em *Hypertext Preprocessor (PHP)* [43], MySQL e SNMP e tem a capacidade de descobrir os dispositivos na rede.

Para a descoberta automática dos dispositivos são necessárias as credenciais SNMP destes, assim como as redes e sub-redes em que estes se encontram. Esta descoberta é executada recorrendo a vários protocolos de encaminhamento e da camada de ligação de dados, nomeadamente ARP, *Foundry Discovery Protocol (FDP)* [40], *Cisco Discovery Protocol (CDP)* [11], LLDP, *Open Shortest Path First (OSPF)* [91], *Border Gateway Protocol (BGP)* [102] e SNMP.

Permite a exportação dos resultados em formato CSV, texto ou XML. Tem também suporte de uma API que permite a gestão dos dados presentes no sistema.

É suportado por vários sistemas operativos Linux, ou pode ser utilizado através de uma imagem Docker [39], ou de imagens VirtualBox [52], já construídas.

NetXMS

O NetXMS [16] é uma ferramenta *open-source* de monitorização e gestão de redes. Esta ferramenta é capaz de fazer a descoberta automática da rede, estando disponíveis dois modos de descoberta: passivo e ativo.

No modo passivo, são apenas utilizados métodos não-intrusivos para obter informação acerca dos dispositivos, nomeadamente as tabelas ARP e de *routing* dos nodos conhecidos. No caso de a ferramenta ter acesso a *routers* e *switches* via SNMP, é possível adicionar automaticamente todos os nodos da rede através do processo de descoberta.

No modo ativo, são definidos intervalos de endereços que são analisados utilizando pedidos ICMP. Quando encontra um novo dispositivo tenta obter mais informações acerca do mesmo, recorrendo aos agentes SNMP e NetXMS. Pelo menos um destes deve estar implementado nas máquinas a ser descobertas para poder ser utilizado.

O NetXMS é também capaz de criar o modelo da topologia da rede recorrendo a vários métodos, tais como, tabelas de *routing*, ARP e FDB e os protocolos CDP, LLDP e NDP.

Em termos de arquitetura, o sistema apresenta três elementos centrais: os agentes de monitorização que recolhem informação (agentes SNMP ou NetXMS); o servidor que processa e armazena a informação recolhida pelos agentes; e a interface com o utilizador, que pode ser uma página web ou uma consola. Em termos de instalação, o servidor é suportado por grande parte dos sistemas operativos existentes.

Estão também disponíveis subagentes que oferecem mais funcionalidades ao agente NetXMS, como por exemplo, monitorização de sistemas de gestão de bases de dados ou conexões via SSH. No entanto, isto requer que o agente NetXMS esteja instalado na máquina a explorar.

O NetXMS disponibiliza também uma API Java e um *web service*. O segundo permite o acesso ao sistema utilizando o protocolo HTTP e JSON como formato de troca de informação.

Spiceworks

O Spiceworks [35] é outro exemplo que se enquadra neste contexto. Esta aplicação possui diversas ferramentas de gestão de redes e software. Estas podem ser utilizadas online, ou podem ser instaladas em máquinas físicas, sendo suportadas em sistemas Linux, Windows e macOS.

A versão online, apesar de necessitar de pouca configuração, implica um inventário simples e rápido dos dispositivos da rede. A sua instalação numa máquina permite o seu total controlo e um inventário do sistema (dispositivos, software, serviços *cloud*, utilizadores, entre outros) mais robusto, automatizado e personalizável.

A ferramenta de inventário faz a análise do sistema e descobre todos os equipamentos conectados na rede, incluindo equipamentos das camadas de rede e de ligação de dados. Não só regista cada dispositivo, como também é capaz de obter informação detalhada sobre o mesmo. Para isto, necessita de comunicar com as máquinas utilizando os protocolos SSH para explorar sistemas Unix, WMI para explorar sistemas Microsoft Windows, SNMP para explorar dispositivos da camada de ligação de dados e HTTP para obter informação acerca de dispositivos NAS.

Esta ferramenta permite também descobrir todo o software e serviços disponíveis, assim como perceber que aplicações estão a sobrecarregar a rede.

Para além disto, esta descoberta pode ser realizada com a utilização de um agente que necessita de estar configurado na máquina alvo, permitindo, por exemplo, a descoberta de dispositivos *Uninterruptible Power Source (UPS)*, sistemas de armazenamento e antivírus.

2.5.5 *Processamento dos Dados Recolhidos*

A informação obtida no processo de descoberta dos CIs e dos seus relacionamentos pode ter origem em diversas fontes e os seus resultados podem apresentar diversos formatos, havendo a necessidade do processamento desta informação com o objetivo de a tornar útil [94].

Podem então ser considerados os processos de normalização e de reconciliação, que devem transformar os dados recolhidos em dados confiáveis para o utilizador.

O processo de normalização consiste na imposição de um formato geral da informação. Tendo em conta a existência de diferentes convenções de nomenclatura e a utilização de abreviações e omissões por parte das diferentes fontes de informação, é necessário transformar os dados recolhidos em informação coerente. Aqui, pode ser necessário executar um conjunto de procedimentos de normalização, tais como, a remoção de notações (por exemplo, *camel case*, *pascal case*, *snake case* ou *kebab case* [7]), a expansão de abreviaturas e acrónimos, a expansão de conceitos com os seus sinónimos e hiperónimos, e a remoção de artigos, preposições e conjunções [65]. Este processo pode ter como base uma lista que define os termos específicos a ser utilizados neste contexto [21]. Os objetivos do processo de normalização passam pela eliminação de informação redundante e desnecessária, estruturação dos dados no formato apropriado e eliminação de incoerências e erros nos dados [84].

O processo de reconciliação consiste na combinação dos dados das diversas fontes de informação. Para isto, é necessário identificar objetos diferentes que correspondam à mesma entidade e eliminar informação duplicada [21]. O processo de identificação pode ser baseado em regras, que devem ser aplicadas de acordo com o tipo de objeto a ser analisado [34]. Estas regras podem consistir num conjunto de atributos que devem ser organizados de acordo com a sua prioridade, sendo que uma maior prioridade significa uma maior confiabilidade para a caracterização única do objeto [65]. Tendo em conta que não devem existir dois, ou mais, objetos que representem a mesma entidade, quando estes são identificados, os seus dados devem ser combinados. Este processo deve ser capaz de eliminar informação duplicada, bem como, resolver situações em que o mesmo atributo apresente valores diferentes. Este problema pode ser resolvido atribuindo um fator de prioridade às fontes de informação, ou pode ser resolvido manualmente [34].

2.6 MODELOS DE DADOS

Nesta secção vão ser abordados conceitos e implementações relativos aos modelos de dados utilizados no contexto de infraestruturas de TI.

2.6.1 *Noções e conceitos*

Após efetuada a descoberta de todos os CIs e dos seus relacionamentos, é necessário armazenar toda esta informação de uma forma estruturada. As opções mais comuns passam por utilizar um esquema de dados personalizado, ou utilizar um já existente. A principal vantagem de utilizar um esquema personalizado recai na possibilidade de personalização dos tipos e dependências convenientes de acordo com o contexto [78]. Por outro lado, utilizar um esquema já existente permite tirar partido de algo que é atualmente adotado por diversas entidades, e que já foi validado por estas.

Neste contexto existe o CIM [23], que é um *standard* que foi definido pela DMTF [66] e define uma forma de representar os elementos de um ambiente computacional como um conjunto de objetos e relacionamentos entre si. O seu objetivo é, através da semântica definida, obter um modelo único para informações de gestão e serviços [9]. Este *standard* inclui um esquema concetual e a sua especificação. Este modelo está organizado seguindo uma abordagem orientada a objetos, sendo definido um conjunto detalhado de classes que descrevem os elementos do ambiente. Neste modelo os componentes são modelados como classes, o estado dos componentes como propriedades, o seu comportamento como métodos da classe e os relacionamentos entre componentes são representados por associações [23]. Sendo baseado em *Unified Modeling Language (UML)* [50], promove herança, abstração, encapsulamento e relacionamentos para melhorar a qualidade e coerência da informação. Uma das vantagens deste modelo é o facto de este poder ser estendido. Para além disto, foi projetado para ser independente de tecnologias [9].

2.6.2 *Mapeamento dos modelos de dados*

Depois de dispor da informação descoberta, esta deve ser transposta para a CMDB utilizada pela organização. Neste contexto, surge a necessidade de partilhar esta informação entre dois modelos de dados distintos: o modelo onde está armazenada a informação

descoberta e o modelo da CMDB. Esta necessidade pode ser resolvida através de uma metodologia de transformação de modelos.

Considerando que a transformação é feita entre modelos, é necessário perceber o que estes são e como são construídos. Os modelos de dados devem então descrever como a informação é representada e definir a sua estrutura e semântica. Neste caso em específico, a transformação ocorre entre dois tipos de modelos diferentes. Em vários casos, a nomenclatura dada ao mesmo tipo de componentes é diferente. Para além disso, os modelos variam na sua organização hierárquica, existindo casos em que, para o mesmo nodo lógico, a sua hierarquia é diferente em dois modelos distintos [111] [96].

Tendo em conta que o mapeamento manual não é opção, porque não é uma tarefa trivial nem é fácil de implementar devido à dimensão dos modelos envolvidos no processo, surge a necessidade de automatizar este processo [104] [96]. Este processo de transformação pode então ser descrito em duas fases: deteção dos mapeamentos entre os dois modelos; e definição das regras de transformação [111].

O mapeamento entre os dois modelos pode ser feito ao nível dos seus elementos. Este é efetuado com base nas representações semânticas e sintáticas destes elementos. Enquanto que as medidas de verificação sintáticas se baseiam na ocorrência de caracteres nas palavras comparadas, as medidas de verificação semântica focam-se no significado atribuído a essas palavras [96] [105].

É, portanto, necessário definir um mecanismo de deteção automática que, aplicando medidas de verificação semânticas e sintáticas, capta os significados e relações entre os elementos dos modelos. Isto pode ser obtido através da análise dos elementos e das suas propriedades. O objetivo passa por descobrir se dois elementos, um proveniente do modelo de origem e outro do modelo de destino, representam o mesmo conceito. A proposta apresentada em [111] baseia-se no teste das relações sintáticas e semânticas entre a nomenclatura de dois elementos e dos seus grupos de propriedades. Estas medidas de verificação apresentam um valor que varia entre zero e um, e quanto maior esse valor, maior semelhança apresentam os dois elementos comparados.

Verificação sintática

As medidas de verificação sintática são usadas para calcular a semelhança entre dois termos para perceber se estes representam o mesmo conceito [111]. Esta similaridade é especificada em termos de um coeficiente, tipicamente entre zero e um, e mede a distância ao nível linguístico entre os dois termos [71].

Verificação semântica

As medidas de verificação semânticas são utilizadas para calcular a semelhança entre o significado de dois termos [111].

Para perceber a relação semântica entre dois termos é necessário entender que relações podem existir entre os significados destes, podendo tratar-se de sinónimos, hiperónimos, hipónimos, merónimos ou holónimos. Neste contexto a relação mais relevante é a de sinonímia, visto que quando dois termos são sinónimos, apresentam o mesmo significado.

O cálculo da similaridade semântica pode ser baseado na utilização de ferramentas que exploram informação semântica, tais como bases de dados lexicais ou *thesaurus*. As bases de dados lexicais armazenam os possíveis significados de palavras ou expressões [96]. O WordNet [36] é exemplo de uma base de dados lexical. Os *thesaurus* são dicionários que

usualmente armazenam entradas com relações de sinónimos e hiperónimos e identificam relações entre termos [71].

Assim como a similaridade sintática, também a similaridade semântica pode ser medida em termos de um coeficiente entre zero e um, que indica a força do relacionamento entre os termos [87].

Regras de mapeamento

Depois de obter os coeficientes de similaridade entre todos os termos dos modelos, é necessário perceber quais os que vão efetivamente tornar-se regras de mapeamento. Para isto, pode ser definido um valor de limite que representa o valor mínimo de similaridade entre dois conceitos que podem ser considerados como compatíveis. Ou seja, dois termos só serão considerados correspondentes se o seu coeficiente de similaridade for superior ao valor de limite [105]. Maiores valores de limite garantem uma maior semelhança entre os conceitos selecionados [87].

Depois de serem identificados os potenciais mapeamentos, devem ser geradas as regras de transformação que permitem estabelecer correspondências entre os elementos dos dois modelos [111].

2.7 TRABALHO RELACIONADO

Atualmente, existem ferramentas que permitem fazer a descoberta automática dos componentes da infraestrutura e das dependências entre si para fazer o povoamento da CMDB.

2.7.1 *Ferramentas*

Device42

A ferramenta disponibilizada pela Device42 [24] possui um mecanismo de descoberta automática e de mapeamento de dependências de aplicações e serviços.

Relativamente ao processo de descoberta, o Device42 permite a definição de tarefas que devem ser configuradas de acordo com a informação que se quer descobrir. Desta forma, é necessário o preenchimento de formulários para definir as tarefas necessárias que podem depois ser agendadas para serem executadas novamente. Para além disto, algumas destas tarefas de descoberta implicam a instalação e configuração de ferramentas de software nas máquinas a descobrir. No entanto, no processo de descoberta em geral, é necessário ter em atenção as permissões associadas aos dispositivos a serem descobertos [19].

As tarefas disponibilizadas pelo Device42 utilizam diferentes métodos para a recolha de dados, sendo os mais utilizados:

- *Ping*;
- Nmap;
- Captura de pacotes;
- SNMP;
- Acesso SSH recorrendo às credenciais da máquina;

- *File Transfer Protocol (FTP)* [100];
- DNS;
- *Intelligent Platform Management Interface (IPMI)*;
- NetFlow Collector (ferramenta);
- *System Center Configuration Manager (SCCM)* (ferramenta);
- Instalação de um agente na máquina a ser explorada.

Estas tarefas permitem a descoberta de diferentes tipos de informação. Na Tabela 2.3 são apresentados os dispositivos possíveis de explorar pelas várias tarefas disponibilizadas pela ferramenta.

Tabela 2.3: Dispositivos explorados pela ferramenta de descoberta do Device42.

Sistemas Operativos	Bases de Dados	Rede	Dispositivos	Cloud	Virtualização	Armazenamento	Documentação
Microsoft Windows				Amazon Web Services	HyperV		
Linux		Switches	IBM I/AS400	Microsoft Azure	ESX		
macOS		APs	Servidores Blade HP	Linode	ESXi		
Unix	Microsoft SQL Server	*Virtual Local Area Networks (VLANs)*	Servidores Blade IBM	Digital Ocean	Citrix XenServer		
IBM z/OS	Oracle DB	Sub-redes	*Unified Computing System (UCS)*	Cloudstack	oVirt		
FreeBSD		*Application Centric Infrastructure (ACI)*	Cisco UCS Cluster	OpenStack	RedHat	SAN Netapp	Certificados
OpenBSD		ACI Fabric	Balanceador de Carga F5	Amazon API	KVM/libvirt	SAN EMC	Garantias
Solaris Sparc				Joyent	OpenV		
				Outscale	AIX HMC		
				Alibaba Cloud	Nutanix		
				Google Cloud	Docker		
				Oracle Cloud	LXC		
				Standalone Kubernetes	VMWare		

Tendo em conta que estes processos podem ser executados independentemente uns dos outros, o Device42 recomenda também uma ordem de descoberta com o objetivo de facilitar o processo de reconciliação. Sugere então que, inicialmente, seja criado o mapa da rede, fazendo o inventário dos dispositivos de rede, VLANs, sub-redes e endereços IP e MAC. De seguida, é sugerido que sejam descobertos *hosts*, máquinas virtuais e sistemas operativos. É depois proposto que estas máquinas sejam exploradas tendo em conta o seu sistema operativo. Seguidamente é recomendada a descoberta de informações acerca de instâncias de *cloud*. No passo seguinte é sugerida a identificação da localização dos servidores *blade* nos chassis, combinando os dados recolhidos nos passos anteriores. Finalmente, é aconselhada a utilização da IPMI para a gestão dos sistemas e monitorização das suas operações [19].

Para o processo de reconciliação dos dados recolhidos, o Device42 começa por verificar números de série, *Universally Unique Identifiers (UUIDs)* e nomes, com o objetivo de perceber se o dispositivo encontrado já existe ou não na CMDB. Para além disto, considera que, se for descoberta informação que já exista na CMDB mas seja diferente, a nova informação é considerada correta e a anterior é substituída por esta [19].

O Device42 possui também um módulo de mapeamento de dependências que verifica relações existentes em serviços e aplicações. Não só encontra conexões entre serviços que estejam a ser executados nas máquinas, como também agrupa serviços em aplicações e obtém a configuração destas [19].

Para o povoamento da CMDB, o Device42 utiliza a sua API para manipular toda a informação [19].

BMC Discovery

O BMC Discovery [22] é uma ferramenta capaz de automatizar o processo de descoberta de ativos e mapeamento de dependências entre aplicações, sendo constituída por vários componentes. O BMC Discovery Outpost é uma aplicação que obtém a informação acerca dos produtos de hardware e sofware da organização. É executada numa máquina com um sistema operativo Microsoft Windows e conecta-se aos dispositivos através do protocolo *HyperText Transfer Protocol Secure (HTTPS)* [103]. O BMC Discovery envia um pedido ao BMC Discovery Outpost para explorar os endereços IP fornecidos, e o BMC Discovery Outpost acede aos alvos recorrendo a credenciais especificadas previamente. Uma vez conectado à máquina alvo, executa comandos para aceder a informação acerca deste, cifrando os resultados obtidos e enviando-os para o BMC Discovery. Quando o BMC Discovery recebe os dados armazena-os no Datastore. Podem ser executados vários BMC Discovery Outpost em simultâneo para lidarem com vários segmentos de rede, sendo que todos comunicam com a mesma instância do BMC Discovery. O Discovery Service desempenha um trabalho semelhante ao Discovery Outpost, mas é executado na mesma máquina do BMC Discovery. Como é baseado em Linux, necessita de um ou mais proxies Windows para a descoberta de sistemas baseados em Microsoft Windows.

A Discovery Queue consiste numa fila de espera que possui tarefas a ser executadas. O Discovery service e o BMC Discovery Outpost interrogam a fila de espera por tarefas a executar. Quando estas são finalizadas, o seu resultado é enviado para a fila, sendo que depois o Reasoning interroga a fila de espera por estes resultados. O Reasoning utiliza padrões para inferir informação acerca de *hosts* e programas com base na informação descoberta. Cada padrão representa conhecimento particular acerca de um produto de hardware ou software, sendo este conhecimento utilizado para gerar informação mais detalhada acerca dos elementos descobertos. A informação é depois armazenada na Datastore. A Datastore é a base de dados onde a informação descoberta e inferida é armazenada. O Cluster consiste em duas ou mais máquinas BMC Discovery, em que uma delas coordena as outras, sendo que podem ser adicionadas mais máquinas ao Cluster em qualquer altura. Quando isto acontece a informação é separada em partes iguais pelas máquinas do Cluster, para que a sua utilização e desempenho sejam o mais otimizados possível [22].

A descoberta tem por base uma lista de endereços IP da rede a descobrir, assim como os endereços que não devem ser explorados.

Para efetuar a descoberta, o BMC Discovery utiliza diferentes métodos dependendo do tipo de alvos a explorar [22]:

- SNMP;
- SSH;
- WMI;
- HTTPS;
- proxies Windows;
- acesso recorrendo às credenciais;
- scripts;
- APIs REST.

Cada método permite a descoberta de determinados tipos de informação. Na Tabela 2.4 são apresentados os dispositivos possíveis de explorar pelos diferentes métodos disponibilizados pela ferramenta.

Tabela 2.4: Dispositivos explorados pela ferramenta de descoberta BMC Discovery.

Sistemas Operativos	Fontes de Dados	Rede	Dispositivos	Virtualização	Armazenamento
Microsoft Windows Linux Unix IBM z/OS	Bases de dados APIs REST	Switches Cisco Nexus	IBM I/AS400 Balanceador de Carga *Management Controllers* Oracle WebLogic Tomcat Mainframe WebSphere Clusters Dispositivos empilhados (Cisco) Plataformas Cloud	ESX ESXi EMC VPLEX Containers	Entidades de armazenamento

O BMC Discovery permite a integração da informação com a CMDB através do mecanismo de sincronização. Relativamente à BMC CMDB, este mecanismo permite que os dados presentes na CMDB sejam sincronizados com a informação descoberta pelo BMC Discovery. No entanto, o modelo de dados do BMC Discovery é diferente do CDM usado pela CMDB, sendo que o mecanismo de sincronização é responsável por transformar a informação de um modelo para o outro. O mapeamento padrão fornece a correspondência para o CDM, no entanto, quando são adicionados nodos personalizados, é necessário efetuar o mapeamento destes manualmente. Para além disto, é também possível integrar a informação descoberta com a ServiceNow CMDB, cujo mapeamento padrão também já se encontra definido [22].

ServiceNow Discovery

O ServiceNow Discovery [48] encontra aplicações e dispositivos na rede e atualiza a ServiceNow CMDB com a informação descoberta.

Podem ser efetuados dois tipos de descoberta distintos. A descoberta horizontal permite encontrar os dispositivos ativos na rede e recolhe informação acerca destes, podendo também criar relacionamentos entre os dispositivos e as aplicações que comunicam entre si. No entanto, não está ciente da existência de serviços. A descoberta de cima para baixo é utilizada para mapear as dependências entre as aplicações e dispositivos que compõem um serviço. Usualmente, são utilizadas em conjunto, sendo que é inicialmente executada a descoberta horizontal para encontrar as máquinas, e posteriormente é executada a descoberta de cima para baixo para estabelecer os relacionamentos entre serviços. O processo de descoberta utiliza *scripts* que recolhem e fazem o processamento da informação. Os *scripts* de descoberta são executados por um servidor *Management, Instrumentation, and Discovery (MID)*, que pode ser executado numa máquina da rede ou na cloud, que, após a execução, retorna os resultados para serem processados [48].

A descoberta horizontal é composta por um conjunto de etapas [94]:

- **Análise:** descobre portas abertas nas máquinas e alguma informação sobre estas;

- **Classificação:** tendo em conta o tipo do dispositivo, executa pedidos específicos para o explorar;
- **Identificação:** tenta obter mais informação sobre o dispositivo e determina se este já se encontra na CMDB, fazendo a reconciliação da informação, caso exista, ou criando o CI, caso contrário;
- **Exploração:** obtém mais informação sobre o dispositivo, como aplicações a correr, e outros atributos.

Os tipos de descoberta horizontal que podem ser utilizados pela aplicação são [48]:

- **Descoberta da rede:** deteta os endereços IP utilizados pela rede da organização;
- **Descoberta de CIs:** encontra dispositivos e aplicações na rede;
- **Descoberta de cloud:** encontra recursos cloud da organização;
- **Descoberta sem servidor:** descobre aplicações em *hosts*.

Para obter informação sobre as máquinas utiliza diferentes métodos dependendo do tipo de máquina a ser explorado. Recorre ao protocolo SSH para se conectar a uma máquina e executar comandos para recolher informação sobre esta; ao protocolo SNMP para recolher informação armazenada nas MIBs dos dispositivos; e ao WMI, PowerShell e WinRM para recolher informação sobre sistemas Microsoft Windows.

Esta ferramenta é então capaz de descobrir sistemas e *workstations* Windows, sistemas Unix e Linux, dispositivos de rede, servidores, bases de dados, impressoras, plataformas cloud, máquinas virtuais, clusters, relacionamentos entre processos a correr e servidores, entre outros.

2.7.2 *Sumário*

Após a análise das ferramentas é possível tirar algumas conclusões acerca dos métodos de descoberta utilizados, assim como da forma como a informação recolhida é integrada com as CMDBs.

É possível verificar que, em geral, os métodos de descoberta mais utilizados são:

- *Ping*;
- Nmap;
- Captura de pacotes;
- SNMP;
- Conexão com a máquina a ser explorada via WinRM e SSH, recorrendo às suas credenciais de acesso, executando comandos ou *scripts*;
- Instalação de ferramentas de software nas máquinas a ser exploradas.

Relativamente à integração com as CMDBs, é verificado que, usualmente, as ferramentas de descoberta de um determinado fornecedor permitem apenas a integração com a CMDB deste. Foi apenas possível perceber que o mecanismo BMC Discovery permite, não só a integração com a BMC CMDB, como também com a ServiceNow CMDB, recorrendo a mapeamentos já configurados. No entanto, geralmente, as CMDBs permitem a personalização dos tipos de CIs, relacionamentos e atributos que utilizam. Neste caso, o mapeamento destes nodos personalizados tem de ser feito manualmente.

3

PROBLEMA E DESAFIOS

Neste capítulo vai ser apresentada a solução proposta para o problema que o projeto tenta solucionar, assim como os principais desafios que podem surgir no desenvolvimento desta.

3.1 PROBLEMA

Analisando o resultado final em iniciativas de criação de uma CMDB, é estimado que cerca de 75% destas falham em comparação com as expetativas iniciais [66]. Assim, é possível afirmar que existe a necessidade de facilitar o processo de criação de uma CMDB.

No entanto, construir uma CMDB implica a utilização de mecanismos automáticos de descoberta e de integração com a CMDB, visto que a criação manual é um processo minucioso e propenso a erros.

Assim, existem alguns problemas derivados do facto de este processo ter de ser praticamente todo implementado pelas organizações. Um desses problemas deriva de inventários incompletos e falta de capacidades de descoberta. Esta informação incompleta pode levar a más decisões e, consequentemente, maus resultados.

Assim, o propósito deste projeto, passa pelo desenvolvimento de uma plataforma capaz de fazer a descoberta automática dos componentes da infraestrutura computacional da organização e o posterior povoamento da sua CMDB. Isto, tendo sempre em conta um processo automático com o mínimo esforço por parte do utilizador.

3.2 ABORDAGEM PROPOSTA

Existem quatro elementos necessários para o funcionamento do projeto: a infraestrutura computacional da organização; o administrador que vai interagir com a plataforma; a plataforma de criação automática a ser desenvolvida; e a CMDB utilizada pela organização. Estes quatro elementos necessitam de interagir entre si para o envio e receção de informação fundamental para a correta execução do processo.

As tarefas a serem realizadas pela plataforma podem ser divididas em três fases:

- fase de descoberta;
- fase de mapeamento;
- fase de povoamento.

3.3 ARQUITETURA DO SISTEMA

A Figura 3.1 apresenta a arquitetura proposta para o sistema, realçando a interação entre todos os elementos e as diversas fases do processo.

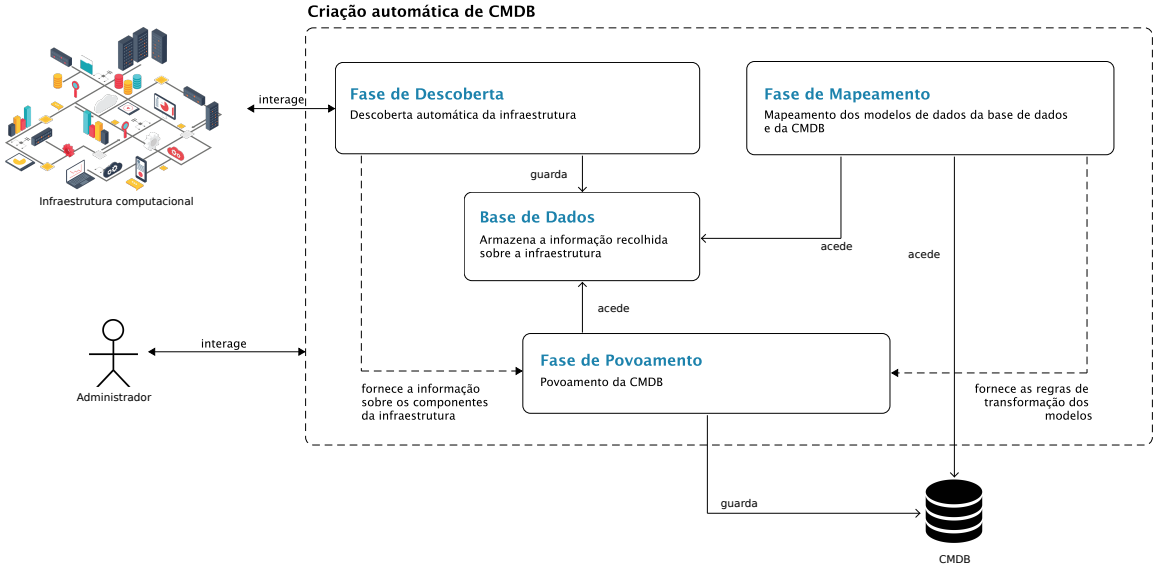


Figura 3.1: Arquitetura do sistema proposta.

3.3.1 *Fase de Descoberta*

A fase de descoberta deverá ser capaz de encontrar os componentes que fazem parte da infraestrutura a ser analisada e armazenar a informação recolhida acerca destes. Deve recorrer a mecanismos de descoberta distintos, que permitam recolher diferentes tipos de informação acerca de diversos dispositivos.

A fase de descoberta deve ter como base um conjunto definido de endereços que deve ser explorado. Deve ser capaz de efetuar uma descoberta básica nestes endereços, recolhendo informação acerca das máquinas ativas e informação essencial acerca destas, podendo ser composta por um ou mais mecanismos diferentes. Um mecanismo de descoberta consiste num método que é executado, recolhe informação acerca da infraestrutura, e envia essa informação para a ferramenta. De cada vez que um mecanismo é executado, deve ser realizado o processo de reconciliação com base na informação recolhida, porque é importante eliminar qualquer tipo de dados duplicados. Para além disto, estes mecanismos podem recolher o mesmo tipo de informação e obter valores diferentes. Desta forma, é necessário perceber qual a informação efetivamente válida. Uma das soluções passa pela execução destes mecanismos de acordo com a sua prioridade. Ou seja, devem ser executados primeiro os mecanismos de menor prioridade, porque desta forma, quando os mecanismos seguintes são executados e encontram um valor diferente para algum parâmetro já recolhido, como são considerados mais confiáveis, podem substituir esta informação, não sendo necessário um processo manual de seleção da informação mais viável. Um mecanismo de descoberta pode também precisar das credenciais de acesso da máquina para recolher de-

terminado tipo de informação, sendo necessário manter registadas estas credenciais de forma segura.

Posteriormente, deve ser efetuada uma descoberta mais detalhada, com base num conjunto de categorias de informação e nos tipos de máquinas descobertos na fase anterior, sendo executados os mecanismos de descoberta que se enquadram nestes parâmetros. Estes mecanismos devem estar associados a uma categoria e a um tipo de CI específico. Assim como na descoberta básica, estes mecanismos devem estar ordenados crescentemente de acordo com a sua prioridade de forma a facilitar o processo de reconciliação. Estes mecanismos específicos são então componentes independentes, que são utilizados de acordo com as necessidades do processo específico. É também relevante permitir a recolha de informação de outras ferramentas de descoberta já existentes. Desta forma, é possível tirar partido de ferramentas já utilizadas e que possuem informação relevante acerca da infraestrutura.

No final, toda a informação recolhida e processada deve ser armazenada numa base de dados.

É apresentado na Figura 3.2 o diagrama de atividades do processo de descoberta da infraestrutura.

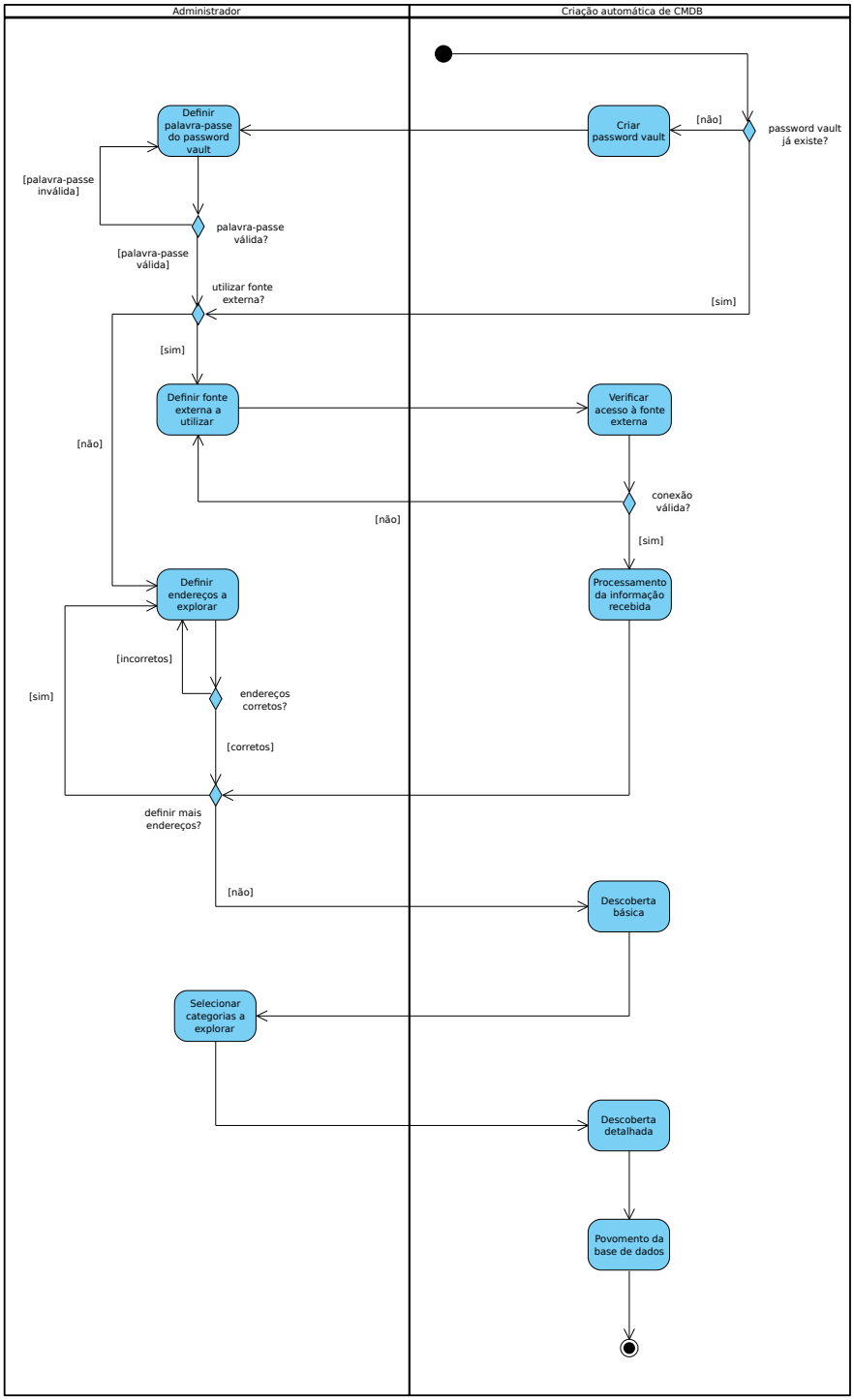


Figura 3.2: Diagrama de atividades da fase de descoberta.

3.3.2 *Fase de Mapeamento*

Esta fase consiste no mapeamento do modelo da base de dados do sistema de criação automática para o modelo da CMDB a ser utilizada pela organização. Devem ser definidas as regras de transformação entre os modelos que permitam associar os CIs da base de dados aos CIs da CMDB.

Inicialmente, é necessário obter informação acerca do modelo de dados da ferramenta, ou seja, da base de dados onde estão armazenados os dados obtidos na fase de descoberta. Posto isto, é necessário aceder à base de dados e perceber que tipos de CIs estão definidos, quais os seus atributos, que tipos de relacionamentos estão definidos entre estes componentes, e que atributos estão associados a estes relacionamentos. Pode ser então definido o modelo de dados da base de dados utilizada.

É apresentado na Figura 3.3 o diagrama de atividades que descreve o processamento do modelo de dados da BD da ferramenta.

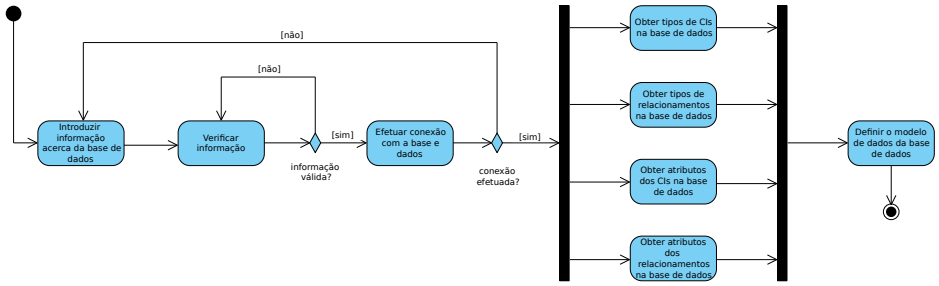


Figura 3.3: Diagrama de atividades do processamento do modelo de dados da base de dados.

De seguida, é necessário perceber como está estruturado o modelo de dados da CMDB. Tal como para a base de dados, é necessário perceber que tipos de CIs estão definidos, quais os seus atributos, que tipos de relacionamentos podem ser definidos entre estes componentes, e que atributos estão associados a estes relacionamentos. É necessário também perceber como são guardados os atributos na CMDB, nomeadamente que tipos de dados são utilizados.

Este processo pode ser executado de duas formas. A primeira, passa por analisar a estrutura da base de dados utilizada pela CMDB. A segunda passa por efetuar pedidos à API da CMDB para obter esta informação. Depois de obter toda a informação necessária da CMDB, é possível definir o modelo de dados da mesma.

Na Figura 3.4 é possível observar o diagrama de atividades que descreve o processamento do modelo de dados de uma CMDB.

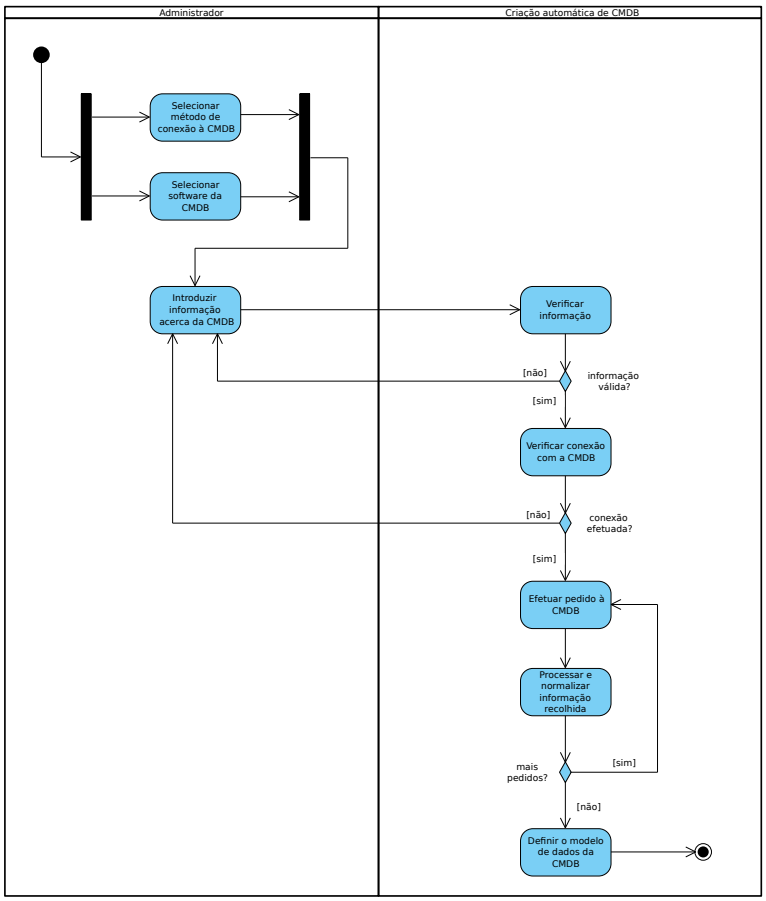


Figura 3.4: Diagrama de atividades do processamento do modelo de dados da CMDB.

Finalmente, é necessário fazer o mapeamento entre os modelos de dados da CMDB e da base de dados, ou seja, é necessário estabelecer correspondências entre os tipos de CIs, relacionamentos e atributos dos dois modelos.

Para isto, é necessário calcular a semelhança entre todos os elementos de cada modelo, e selecionar os mais semelhantes, atribuindo a um elemento do modelo da base de dados uma correspondência com um elemento do modelo da CMDB. Assim, no final, é obtido um conjunto de regras de transformação que associam os elementos mais semelhantes dos dois modelos.

Na Figura 3.5 é possível observar o processo de definição das regras de transformação entre os dois modelos de dados.

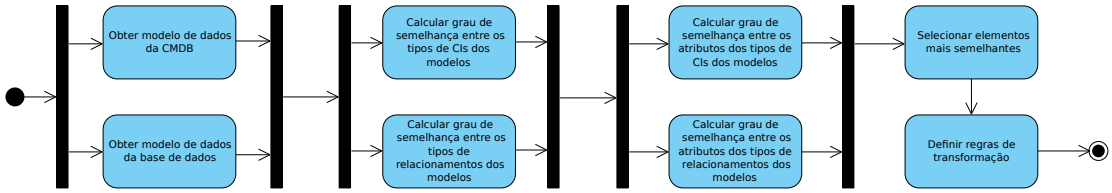


Figura 3.5: Diagrama de atividades da definição das regras de transformação entre os modelos de dados.

3.3.3 *Fase de Povoamento*

Nesta fase deverá ser efetuado o povoamento da CMDB com os dados recolhidos na fase de descoberta.

Nesta fase é necessário aceder aos dados guardados na base de dados e às regras de transformação definidas na fase de mapeamento.

Para cada objeto presente na base de dados, devem ser usadas as regras de transformação necessárias de forma a criar o mesmo objeto na CMDB. Para isto é necessário aceder à CMDB, sendo que a melhor solução é a utilização da sua API, visto que o acesso direto à sua base de dados pode levar a incoerências nos dados.

Na Figura 3.6 é apresentado o diagrama de atividades que descreve a fase de povoamento da CMDB.

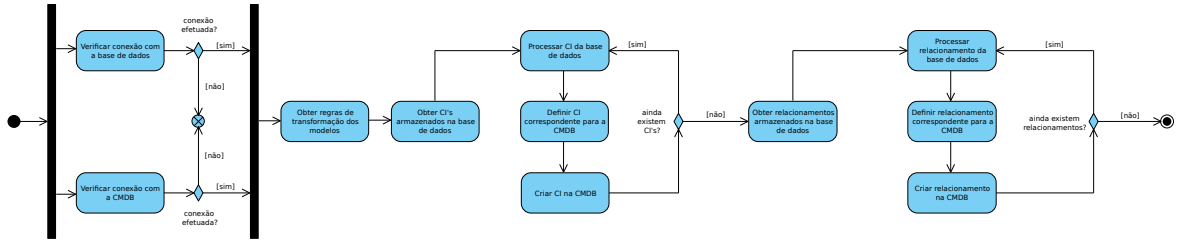


Figura 3.6: Diagrama de atividades da fase de povoamento.

Depois de todos os objetos terem sido criados na CMDB, temos o processo de criação automática da CMDB finalizado.

3.4 DESAFIOS

O maior desafio do processo de criação automática da CMDB é a automatização, ou seja, a redução ao máximo do esforço desempenhado pelo utilizador. Para além deste, também as diversas fases de execução apresentam alguns desafios.

Na fase de descoberta, surgem dificuldades no processo de implementação dos mecanismos de descoberta, assim como na descoberta de determinados tipos de dados. Para além disto, surgem problemas de segurança devido às permissões necessárias para a descoberta de informação, quer seja ao nível da firewall da empresa, como da necessidade de credenciais das máquinas a ser exploradas. Ainda no processo de descoberta, é necessária a deteção e reconciliação de dados duplicados, sendo um dos principais desafios a manutenção da coerência da informação recolhida. Além disso, é também necessário o tratamento de informação em diversos formatos devido à combinação de diferentes mecanismos de descoberta. Também a necessidade de um modelo genérico capaz de modelar qualquer CI, relacionamento ou atributo que possa estar definido em qualquer modelo de CMDB existente surge como um desafio nesta fase. Finalmente, o facto de a mesma informação poder ser guardada de forma diferente em CMDBs distintas, por exemplo, determinados dados podem ser guardados como atributos ou como um relacionamento, causa dificuldades ao nível do armazenamento dos dados recolhidos.

Na fase de mapeamento, os principais desafios estão relacionados com a complexidade dos modelos de dados envolvidos, visto que estes representam sistemas complexos [111]. No processo de mapeamento entre os elementos dos dois modelos não é possível utilizar a

hierarquia dos elementos como comparação, visto que, a hierarquia dos nodos raramente é a mesma em dois modelos de dados diferentes, mesmo para o mesmo nodo lógico [96]. Além disto, também não é possível recorrer aos tipos de dados dos atributos para perceber se correspondem ao mesmo conceito, visto que a mesma informação pode ser armazenada de formas diferentes [71]. Também a complexidade dos modelos, nomeadamente o número de elementos e de atributos de cada elemento, representam um desafio, visto ser necessário calcular um coeficiente de similaridade entre todos os elementos e entre todos os atributos dos dois modelos e, de entre todos os valores calculados, selecionar os potenciais mapeamentos. Ainda neste processo, é possível que existam situações em que a semelhança entre dois nodos de um modelo, e um nodo de outro, sejam iguais. Neste caso é necessário decidir qual dos nodos é o mais semelhante, mesmo que os valores de semelhança calculados sejam iguais.

Na fase de povoamento da CMDB surgem algumas dificuldades na criação dos CIs e relacionamentos entre estes. O primeiro desafio recai na necessidade de fazer a conversão dos tipos de dados da informação recolhida para aqueles utilizados pelo modelo da CMDB. No entanto, existem casos em que isto pode não ser possível. Para além disto, o modelo da CMDB pode possuir atributos em que os seus valores possíveis estejam previamente definidos. Ou seja, é necessário converter a informação recolhida, em informação que pode ser armazenada na CMDB. Finalmente, outro problema que pode ocorrer nesta fase é a necessidade de converter os valores recolhidos para as unidades utilizadas na CMDB. Consequentemente, outro desafio passa também por perceber que unidades são utilizadas na CMDB.

4

DESENVOLVIMENTO

Este capítulo aborda o desenvolvimento do sistema descrito no Capítulo 3. São apresentadas as decisões a nível tecnológico, assim como a metodologia implementada nos diversos componentes do sistema. O projeto encontra-se disponível em https://github.com/joanapereira115/cmdb-auto-creation.

4.1 DECISÕES

O propósito deste projeto passa pelo desenvolvimento de uma aplicação capaz de efetuar a criação automática de uma CMDB, utilizando mecanismos automáticos de descoberta de informação, mapeamento entre modelos e povoamento.

Para este efeito, a linguagem de programação utilizada foi Python [46]. Foi escolhida esta linguagem devido à sua facilidade de utilização, quer a nível de escrita como de execução do código, e à existência de diversas bibliotecas *open source* [108], que permitem a utilização de vários módulos essenciais no desenvolvimento das diversas funcionalidades do projeto. São disso exemplo, as seguintes bibliotecas:

- **requests [33]:** permite o envio de pedidos HTTP, sendo utilizada para efetuar pedidos às APIs das CMDBs no povoamento e obtenção do modelo de dados destas;
- **paramiko [32]:** permite a conexão via SSH com as máquinas, sendo utilizada para a recolha de informação na fase de descoberta;
- **pywinrm [17]:** permite a conexão via WinRM com sistemas Windows, sendo utilizada para a recolha de informação na fase de descoberta;
- **python-nmap [44]:** permite a utilização da ferramenta Nmap, sendo utilizada para a recolha de informação na fase de descoberta;
- **pyshark [45]:** utilizada na fase de descoberta para a captura e análise de pacotes na rede;
- **easysnmp [13]:** utilizada na fase de descoberta para a recolha de informação através do protocolo SNMP;
- **spacy [49]:** utilizada para o cálculo da similaridade semântica entre dois termos;
- **nltk [31]:** permite a utilização da base de dados léxica WordNet, sendo utilizada no cálculo da similaridade semântica entre dois termos;

- **mysql [10]:** permite o acesso a bases de dados MySQL, sendo utilizada para aceder às bases de dados das CMDBs;
- **wordninja [20]:** utilizado na normalização de texto, permite a separação de palavras concatenadas.

Para além disto, a familiaridade de utilização desta linguagem para processamento de texto foi também um fator a ter em consideração para a sua seleção.

Relativamente à base de dados, foi escolhido o GraphDB [26], que armazena os dados em estruturas em forma de grafo. Apesar das CMDBs estudadas recorrerem a bases de dados relacionais para armazenar a informação, foi escolhida uma base de dados em grafo, porque o modelo de dados desenvolvido é um modelo genérico, e deve ser capaz de representar qualquer tipo de componente, relacionamento ou atributo, que acabam por ser opcionais em diversos casos, enquanto que os modelos das CMDBs possuem toda a sua organização e caracterização especificada. Para além disto, é possível recorrer mais facilmente à visualização dos itens de configuração e dos seus relacionamentos na forma de grafo, porque as conexões são mais facilmente visualizadas no formato de grafo do que numa base de dados relacional.

4.2 IMPLEMENTAÇÃO

Com base na arquitetura apresentada na Secção 3.1, o projeto encontra-se dividido em três fases: descoberta, mapeamento e povoamento. Cada uma destas etapas utiliza diferentes componentes que trabalham em conjunto para executar todas as tarefas necessárias do processo de criação automática de uma CMDB, sendo que alguns destes são utilizados em mais do que uma fase. A arquitetura dos componentes do sistema é apresentada na Figura 4.1 e a descrição de cada um dos componentes é apresentada nesta secção.

4.2.1 *Esquema da base de dados*

Como já referido na Secção 2.6, foi necessário definir um esquema de dados para armazenar a informação descoberta. Tendo em conta que este modelo vai posteriormente ser utilizado para o mapeamento com os modelos de dados das várias CMDBs, este deve ser o mais genérico possível, de forma a conseguir representar qualquer CI, relacionamento ou atributo que possa estar definido nestes. Assim, o modelo de dados em que será baseada a base de dados a utilizar no processo, está representado na Figura 4.2. Este, teve como base essencial o modelo concetual apresentado por Thomas Schaaf e Boran Gögetap [81] [106] apresentado na Figura 2.1.

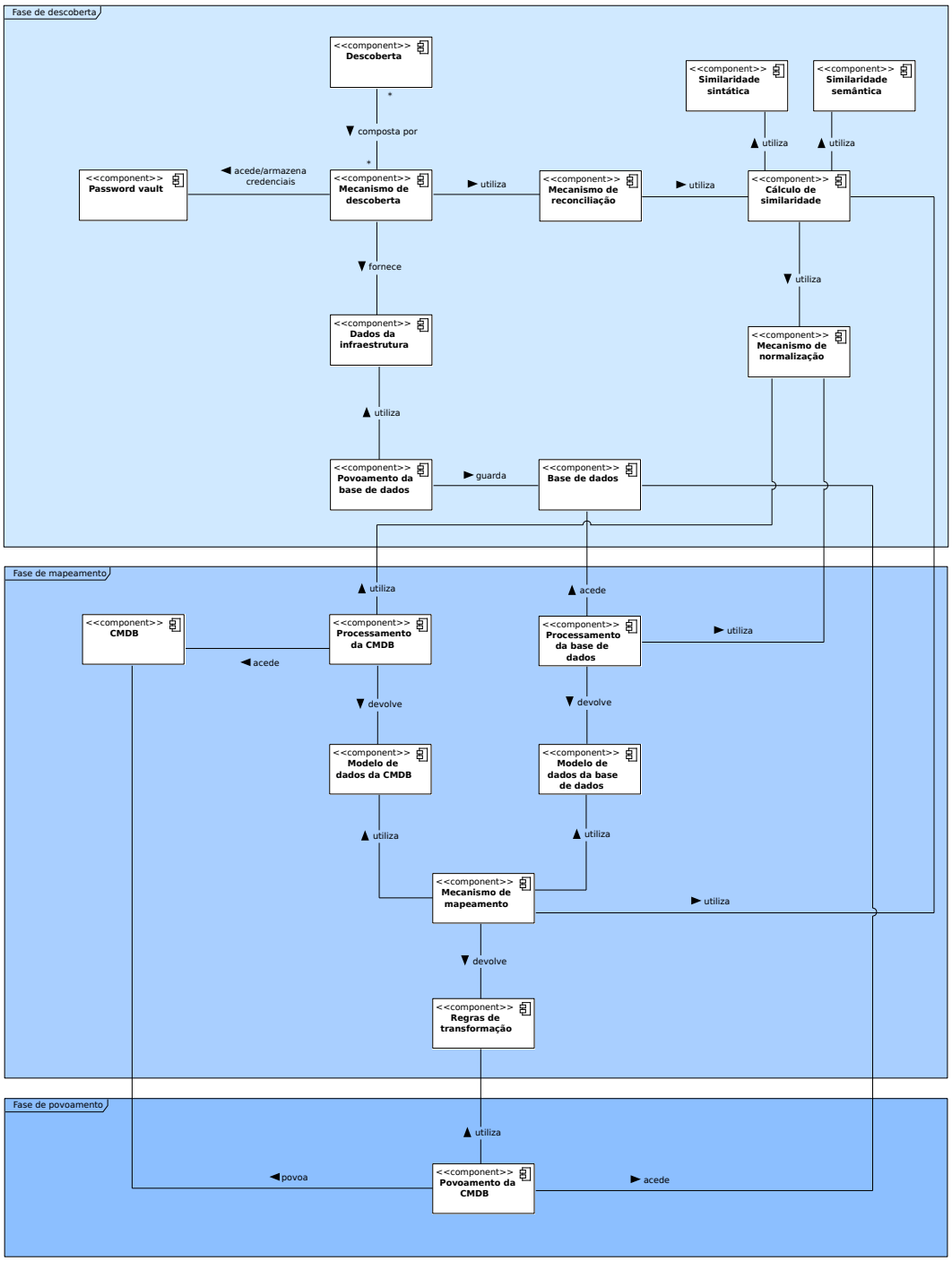


Figura 4.1: Arquitetura dos componentes do sistema.

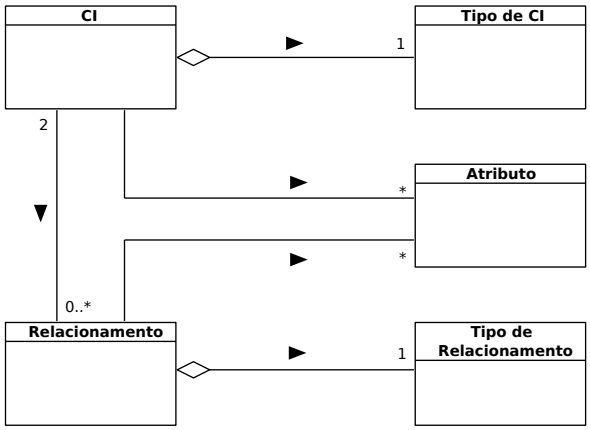


Figura 4.2: Modelo concetual da base de dados.

O modelo apresentado permite que sejam representados quaisquer tipos de CIs e relacionamentos entre si, assim como a designação de qualquer atributo a um CI ou relacionamento. Para além disto, não restringe o tipo de CIs envolvidos num relacionamento.

Apesar de os atributos serem definidos consoante a necessidade de caracterização de um CI ou relacionamento, foi definido um conjunto de atributos base a ser associado a estes. Estes atributos são importantes porque facilitam o processo de reconciliação e de seleção de mecanismos de descoberta a executar. Alguns dos atributos selecionados foram baseados nas classes mais genéricas do CIM [23], apresentadas na Figura 4.3.

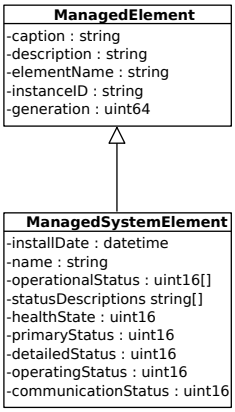


Figura 4.3: Classes *ManagedElement* e *ManagedSystemElement* do modelo CIM.

É então apresentado na Figura 4.4 o esquema do modelo de dados em que vai ser baseada a base de dados do sistema. Como é possível observar, os atributos referentes ao identificador, nome, descrição e estado tiveram por base as classes do CIM. Os atributos referentes ao UUID, número de série, endereço MAC e endereços IP foram definidos para facilitar o processo de reconciliação, visto que estes atributos ajudam na identificação única dos CIs. O atributo referente à família do sistema operativo foi utilizado para facilitar o processo da descoberta, que pode ser baseado neste atributo.

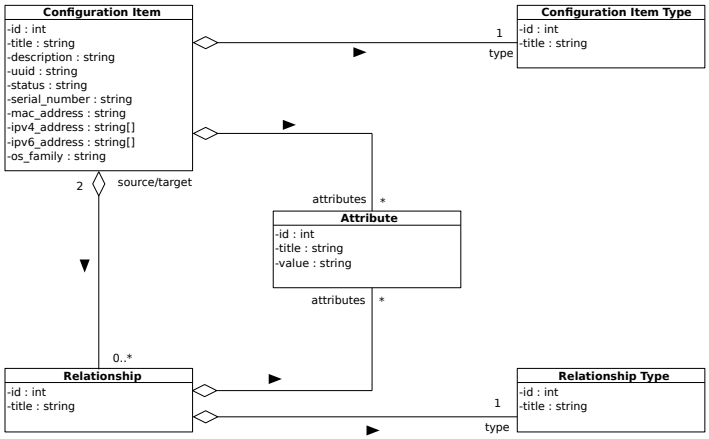


Figura 4.4: Diagrama de classes da base de dados.

Tendo em conta que o GraphDB [26] permite a importação dos dados no formato *Resource Description Framework (RDF)* [14], ou seja, a expressão da informação e do esquema de dados é feita recorrendo a triplos, é necessário efetuar a definição deste esquema. Em RDF um triplo representa uma relação de recursos e propriedades. Existem diversas sintaxes RDF, mas a selecionada foi Turtle [58].

Assim, para definir o esquema de dados, é necessário definir as classes e propriedades da ontologia. Na Listagem 4.1 é possível observar a definição das classes em Turtle. Na Listagem 4.2 são apresentadas as propriedades dos dados, que correspondem aos atributos das classes no esquema da Figura 4.4. Por fim, na Listagem 4.3, estão definidas as propriedades dos objetos que correspondem às associações entre as classes.

Listagem 4.1: Classes da ontologia definidas em Turtle.

```
:ConfigurationItem rdf:type owl:Class ;
    rdfs:label "Configuration Item" ;
    rdfs:comment "Represents an organization's infrastructure component." .

:Relationship rdf:type owl:Class ;
    rdfs:label "Relationship" ;
    rdfs:comment "Represents the relationship between two organization's infrastructure
        components." .

:ConfigurationItemType rdf:type owl:Class ;
    rdfs:label "Configuration Item Type" ;
    rdfs:comment "Represents the type of a configuration item." .

:RelationshipType rdf:type owl:Class ;
    rdfs:label "Relationship Type" ;
    rdfs:comment "Represents the type of a relationship." .

:Attribute rdf:type owl:Class ;
    rdfs:label "Attribute" ;
```

```
    rdfs:comment "Represents an attribute of a configuration item or relationship." .
```

Listagem 4.2: Propriedades dos dados da ontologia definidas em Turtle.

```
:uuid rdf:type owl:DatatypeProperty ;
    rdfs:domain :ConfigurationItem ;
    rdfs:range xsd:string ;
    rdfs:comment "The universally unique identifier (128-bit number) assigned to the item.".

:serial_number rdf:type owl:DatatypeProperty ;
    rdfs:domain :ConfigurationItem ;
    rdfs:range xsd:string ;
    rdfs:comment "The manufacturer-allocated number used to identify the item.".

:title rdf:type owl:DatatypeProperty ;
    rdfs:domain :ConfigurationItem ,
                :Relationship ,
                :ConfigurationItemType ,
                :RelationshipType ,
                :Attribute ;
    rdfs:range xsd:string ;
    rdfs:comment "The label by which the item is known.".

:description rdf:type owl:DatatypeProperty ;
    rdfs:domain :ConfigurationItem ;
    rdfs:range xsd:string ;
    rdfs:comment "The textual description of the item.".

:status rdf:type owl:DatatypeProperty ;
    rdfs:domain :ConfigurationItem ;
    rdfs:range xsd:string ;
    rdfs:comment "The current status value for the operational condition of the item.".

:mac_address rdf:type owl:DatatypeProperty ;
    rdfs:domain :ConfigurationItem ;
    rdfs:range xsd:string ;
    rdfs:comment "The media access control address assigned to the item.".

:value rdf:type owl:DatatypeProperty ;
    rdfs:domain :Attribute ;
    rdfs:range xsd:string ;
    rdfs:comment "The value of the attribute.".

:has_ipv4 rdf:type owl:DatatypeProperty ;
    rdfs:domain :ConfigurationItem ;
    rdfs:range xsd:string ;
```

```
    rdfs:comment "The associated IPv4 address.".

:has_ipv6 rdf:type owl:DatatypeProperty ;
    rdfs:domain :ConfigurationItem ;
    rdfs:range xsd:string ;
    rdfs:comment "The associated IPv6 address.".

:os_family rdf:type owl:DatatypeProperty ;
    rdfs:domain :ConfigurationItem ;
    rdfs:range xsd:string ;
    rdfs:comment "The associated operating system family.".
```

Listagem 4.3: Propriedades dos objetos da ontologia definidas em Turtle.

```
:has_ci_type rdf:type owl:ObjectProperty ;
    rdfs:domain :ConfigurationItem ;
    rdfs:range :ConfigurationItemType .

:has_rel_type rdf:type owl:ObjectProperty ;
    rdfs:domain :Relationship ;
    rdfs:range :RelationshipType .

:has_source rdf:type owl:ObjectProperty ;
    rdfs:domain :Relationship ;
    rdfs:range [rdf:type owl:Restriction;
                owl:onProperty :has_source;
                owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger;
                owl:onClass :ConfigurationItem ] .

:has_target rdf:type owl:ObjectProperty ;
    rdfs:domain :Relationship ;
    rdfs:range [rdf:type owl:Restriction;
                owl:onProperty :has_target;
                owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger;
                owl:onClass :ConfigurationItem ] .

:has_attribute rdf:type owl:ObjectProperty ;
    rdfs:domain :Relationship ,
                :ConfigurationItem ;
    rdfs:range :Attribute .
```

4.2.2 *Gestão de palavras-passe*

Tendo em conta que alguns mecanismos de descoberta precisam das credenciais das máquinas para as explorar, surge a necessidade de utilizar uma ferramenta capaz de as

armazenar de forma segura. Desta forma, foi adotada a utilização de um *password vault*, capaz de armazenar e cifrar várias palavras-passe, sendo que o utilizador necessita apenas de memorizar a palavra-passe do *password vault* de forma a aceder a estas.

Considerando que já existem ferramentas deste género implementadas, recorreu-se à adaptação do projeto PasswordVault [55]. Para a utilização deste foi necessário implementar o processo de uma forma diferente, de forma a que este se enquadrasse no projeto descrito neste documento.

Essencialmente, este mecanismo permite ao programa:

- criar um novo *password vault* caso este ainda não exista;
- definir a sua palavra-passe;
- desbloquear ou bloquear o *password vault*;
- armazenar uma nova palavra-passe, associando-a a um nome de utilizador e a um domínio;
- aceder a uma palavra-passe previamente armazenada;
- eliminar o *password vault*.

Na Figura 4.5 são apresentados alguns exemplos de utilização das funcionalidades do *password vault* descrito.

```
>>> vault.initialize()
>>> Initializing the vault...
>>> Vault exists.
>>> Creating the database file...

>>> vault.define_master_key("test_password")
>>> Your vault has been created and encrypted with your master key.

>>> vault.unlock("test_password")
>>> Vault unlocked.

>>> vault.add_secret("10.10.10.1", "test", "password")
>>> Password added to the vault.

>>> vault.show_secret_by_username("test")
'password'

>>> vault.show_secret_by_name("10.10.10.1")
'password'

>>> vault.lock()
>>> Vault locked.

>>> vault.show_secret_by_name("10.10.10.1")
>>> Vault is locked.
```

Figura 4.5: Exemplo de utilização do *password vault*.

4.2.3 *Normalização*

Este mecanismo tem como objetivo a imposição de um formato global para a informação, de forma a tornar esta coerente.

Para atingir este objetivo, foi necessário efetuar determinadas transformações no texto a ser processado, nomeadamente:

- remover formatações do texto (*snake, kebab, pascal* ou *camel case*);
- separar palavras concatenadas;
- remover múltiplos caracteres especiais, como por exemplo, espaços, parágrafos ou *tabs*;
- ignorar maiúsculas e minúsculas;
- expandir siglas e acrónimos;
- remover palavras vazias, como por exemplo artigos, preposições e conjunções;
- remover sinais de pontuação.

O Algoritmo 1 apresenta o processo de normalização aplicado ao texto, com o objetivo de remover ao máximo o conteúdo desnecessário deste. Este é utilizado para o processamento de texto que vai ser comparado com outro, que deve também ser normalizado.

Algoritmo 1 Processamento de texto a ser comparado.

Require: $txt \neq None$

$res1 \leftarrow$ remover formatação *snake, kebab, pascal* ou *camel case* de txt

$res2 \leftarrow$ remover múltiplos caracteres especiais de $res1$

$res3 \leftarrow$ remover sinais de pontuação de $res2$

$res4 \leftarrow$ converter $res3$ para minúsculas

$res5 \leftarrow$ expandir siglas e acrónimos presentes em $res4$

$res6 \leftarrow$ separar palavras concatenadas de $res5$

$res7 \leftarrow$ remover palavras vazias de $res6$

return $res7$

A Figura 4.6 apresenta alguns exemplos de utilização do mecanismo de normalização. É possível perceber que o mecanismo é capaz de efetuar as transformações descritas anteriormente. Desta forma, e tendo em conta que a informação é proveniente de fontes diferentes, o processamento permite uma posterior comparação entre texto com o mesmo formato.

```
>>> normalization.parse_text_to_compare("Layer 3-net")
'layer 3 network'
>>> normalization.parse_text_to_compare("os_family")
'operating system family'
>>> normalization.parse_text_to_compare("hasIpv4")
'internet protocol version 4'
>>> normalization.parse_text_to_compare("CPU")
'central processing unit'
>>> normalization.parse_text_to_compare("applicationsolution")
'application solution'
```

Figura 4.6: Exemplo de utilização do mecanismo de normalização.

4.2.4 *Reconciliação*

O mecanismo de reconciliação é um componente fundamental deste processo, visto que assegura que não existem dados duplicados e que a informação referente à mesma entidade é combinada no mesmo objeto.

O primeiro passo deste processo passa pela identificação de objetos que correspondam à mesma entidade. Para isto é necessário verificar se um objeto já existe antes de criar um novo. Este processo apresenta diferenças entre itens de configuração e relacionamentos.

Para verificar se um CI já existe são inicialmente comparados alguns dos seus atributos, sendo que estes são verificados de forma ordenada de acordo com a sua prioridade. O primeiro a ser comparado é o atributo que guarda o valor do UUID. Se este valor existir e for igual nos dois objetos, é concluído que estes correspondem à mesma entidade. Caso isto não se verifique, é utilizada a mesma metodologia para o atributo que representa o número de série. Caso esta verificação também não corresponda, é verificado o atributo que guarda o valor do endereço MAC do objeto. Se nenhum destes valores estiver associado ao objeto que está a ser verificado, a comparação é direcionada para os seus endereços IP e para os seus atributos. Após efetuada a comparação entre todos os atributos e os seus valores, é concluído que os componentes correspondem à mesma entidade se mais de 80% dos seus atributos for semelhante.

Para verificar se um relacionamento já existe é necessário comparar este com todos os já existentes. Inicialmente são verificados os componentes fonte e alvo envolvidos no relacionamento. Se estes forem iguais, é comparado o tipo do relacionamento. Após efetuada a comparação entre todos os relacionamentos, é concluído que os relacionamentos correspondem à mesma entidade se envolverem os mesmos componentes e o coeficiente de similaridade calculado entre os seus tipos for o máximo valor encontrado superior a 0.90 (zero ponto noventa).

Caso seja encontrado um CI ou um relacionamento que já existe, é necessário seguir para o processo de reconciliação. Assim como para a identificação, também a reconciliação é diferente entre itens de configuração e relacionamentos.

Para a reconciliação de dois itens de configuração é inicialmente combinada a informação dos atributos. Para os atributos que não apresentem um valor no novo objeto, é adquirida a informação do objeto já existente, sendo que os atributos do novo objeto são mantidos, assumindo que a execução dos mecanismos de descoberta é feita de forma crescente de acordo com a sua fiabilidade. Apenas para os atributos relativos à descrição e aos endereços IP é combinada a informação dos dois componentes. Finalmente é necessário fazer a combinação dos relacionamentos. Para todos os relacionamentos em que o objeto já existente esteja envolvido, é necessário substituir pelo novo objeto.

Na reconciliação entre dois relacionamentos é necessário combinar os seus atributos. Para os atributos que não apresentem um valor no novo relacionamento, é adquirida a informação do relacionamento já existente, sendo que os atributos do novo relacionamento são mantidos.

Em ambos os processos de reconciliação, é utilizado o cálculo de similaridade, descrito na Secção 4.2.9, para perceber se um atributo do novo objeto possui algum atributo correspondente no objeto já existente. Este processo é também utilizado ao longo da descoberta, ou seja, antes de definir um atributo de um objeto, é verificado se este já existe. Além disto, seguindo a mesma metodologia, antes de criar um tipo de CI ou de relacionamento é também verificado se este já existe.

```
>>> ci1 = ConfigurationItem.ConfigurationItem()
>>> ci1.set_title("Macbook Pro")
>>> ci1.set_serial_number("C02S80K6G8WE")
>>> methods.define_attribute("cpu frequency unit", "MHz", ci1)
>>> methods.define_attribute("cpu frequency unit", "GHz", ci1)

>>> ci2 = ConfigurationItem.ConfigurationItem()
>>> ci2.set_serial_number("C02S80K6G8WE")
>>> ci2.add_ipv4_address("192.168.1.5")

>>> methods.add_ci(ci1)
>>> methods.ci_already_exists(ci2)
<models.ConfigurationItem.ConfigurationItem object at 0x1292bc760>
>>> ci1
<models.ConfigurationItem.ConfigurationItem object at 0x1292bc760>
>>> methods.add_ci(ci2)

>>> print_obs.print_objects()
SERIAL NUMBER: C02S80K6G8WE
TITLE: Macbook Pro
IPV4: 192.168.1.5
CPU FREQUENCY UNIT: GHz
```

Figura 4.7: Exemplo de utilização do mecanismo de reconciliação.

Como é possível observar na Figura 4.7, foram criados dois componentes com o mesmo número de série, um deles com um atributo relativo ao nome e outro à unidade de frequência do CPU, e o outro com um endereço IP associado. Quando o segundo elemento é guardado, como o valor do atributo referente ao número de série é o mesmo do primeiro, é combinada a informação dos dois objetos num único. No final é possível verificar a existência de um único objeto com atributos relativos ao nome, à unidade de frequência do CPU, ao número de série e ao endereço IP. É também possível observar que, quando é adicionado outro atributo referente à unidade de frequência do CPU, como este já existe, não é criado um novo, mas é atualizado o valor desse atributo.

4.2.5 *Descoberta*

A descoberta da infraestrutura computacional implica a utilização de diferentes mecanismos que permitam a recolha de informação variada. Tendo em conta que a descoberta tem como ponto inicial um conjunto de endereços, durante este processo é mantida informação acerca dos endereços descobertos, assim como das redes a que pertencem.

O Algoritmo 2 apresenta o processo de descoberta implementado, sendo que as várias fases deste são descritas de seguida.

Algoritmo 2 Processo de descoberta

```
if utilização de fonte externa then
  for objeto da fonte do
    guardar endereço
    processar informação do objeto
  end for
end if
if utilizador introduz endereços then
  guardar endereços
end if
basic_discovery ← descoberta básica nos endereços definidos
categories ← seleção das categorias a explorar
detailed_discovery ← descoberta detalhada com base nas categorias selecionadas
```

Fonte externa

O primeiro mecanismo, de uso opcional, que pode ser utilizado neste processo é a utilização de uma fonte externa. O uso de um mecanismo deste género permite que o utilizador execute processos de descoberta com que já está familiarizado. Neste caso em específico, como exemplo, foi implementado um mecanismo capaz de processar os resultados obtidos através da ferramenta Angry IP Scanner [79], já abordada na Secção 2.5.4. Como esta ferramenta permite a exportação dos resultados em formato CSV, o mecanismo necessita apenas de ler este ficheiro e, tendo em conta a nomenclatura dada às colunas, criar os objetos e atributos existentes e guardar a informação referente aos endereços, sendo que estes vão ser posteriormente utilizados no processo de descoberta.

É possível observar na figura 4.8, que a informação proveniente do ficheiro CSV apresentada na Tabela 4.1 foi processada, e foram criados os CIs com esses dados.

Tabela 4.1: Exemplo de um ficheiro CSV exportado da ferramenta Angry IP Scanner.

IP	192.168.1.69	192.168.1.254	192.168.1.64	192.168.1.71	192.168.1.83
Ping	1 ms	2 ms	3 ms	5 ms	19 ms
Hostname	macbookprojoana.local	[n/a]	DESKTOP-DPANURS	[n/a]	[n/a]
NetBIOS Info	[n/a]	[n/a]	WORKGROUP\DESKTOP-DPANURS [48-4D-7E-A7-D6-5B]	[n/a]	[n/a]
MAC Address	C4:B3:01:C0:1A:D4	CC:19:A8:62:4B:1G	48:4D:7E:A7:D6:5B	9E:DE:D0:7D:DB:E2	20:7C:8F:D8:26:9B
MAC Vendor	Apple	PT Inovação e Sistemas SA	Dell	[n/a]	Quanta Microsystems

```
[>>> angry_ip_scanner.parse_info()

 >>> Make sure that you have imported the Angry IP Scanner .csv file into the folder 'external_data'.
 ? Have you imported the file into the 'external_data' folder?
   Yes

[? Enter the Angry IP Scanner .csv filename.  angry_ip_scanner.csv
[>>>
[>>> print_objects()

 -----------------------------------
 TITLE: apple
 TYPE: vendor
 -----------------------------------
 TITLE: macbookprojoana.local
 MAC ADDRESS: c4:b3:01:c0:1a:d4
 IPV4: 192.168.1.69
 HOSTNAME: macbookprojoana.local
 VENDOR: apple
 -----------------------------------
 TITLE: pt inovação e sistemas sa
 TYPE: vendor
 -----------------------------------
 MAC ADDRESS: cc:19:a8:62:4b:1g
 IPV4: 192.168.1.254
 VENDOR: pt inovação e sistemas sa
 -----------------------------------
 TITLE: dell
 TYPE: vendor
 -----------------------------------
 TITLE: desktop dpanurs
 MAC ADDRESS: 48:4d:7e:a7:d6:5b
 IPV4: 192.168.1.64
 HOSTNAME: desktop dpanurs
 NETWORK BASIC INPUT/OUTPUT SYSTEM: workgroup\desktop dpanurs [48 4d 7e a7 d6 5b]
 VENDOR: dell
 -----------------------------------
 MAC ADDRESS: 9e:de:d0:7d:db:e2
 IPV4: 192.168.1.71
 -----------------------------------
 TITLE: quanta microsystems
 TYPE: vendor
 -----------------------------------
 MAC ADDRESS: 20:7c:8f:d8:26:9b
 IPV4: 192.168.1.83
 VENDOR: quanta microsystems
```

Figura 4.8: Exemplo de execução da descoberta recorrendo à ferramenta Angry IP Scanner.

Descoberta básica

A descoberta básica procura encontrar as máquinas de acordo com um conjunto de endereços, adquiridos através da ferramenta externa e/ou definidos pelo utilizador.

A primeira etapa passa pela análise de pacotes LLDP. Tendo em conta que este protocolo, executado na camada de ligação de dados, é usado pelos dispositivos para comunicar informação com outros na rede, são filtrados os pacotes LLDP na máquina que está a executar a descoberta, como forma de descobrir outras máquinas. Para que esta análise seja realizada, é necessário que a máquina que está a executar o processo de descoberta possua o protocolo LLDP configurado para a receção de pacotes. Desta forma, se existirem dispositivos na rede a enviar pacotes LLDP, estes vão ser captados.

De seguida, para cada endereço encontrado, são executados mecanismos baseados nos protocolos ICMP e SNMP, abordados na Secção 2.5.2, e na ferramenta Nmap, abordada na Secção 2.5.4.

Inicialmente, é efetuado um pedido do tipo *ping*, para verificar se existe uma máquina ativa nesse endereço. Neste caso, é necessário que a máquina consiga responder ao pedido, ou seja, esta pode estar ativa mas não ser capaz de responder, por exemplo devido a configurações da firewall. Em qualquer caso, se a máquina não responder, o endereço não é mais considerado para exploração, porque não é possível aceder a esta.

Depois é utilizada a ferramenta Nmap, que recolhe informação acerca das máquinas, nomeadamente:

- deteção do nome e tipo da máquina;
- identificação dos endereços IP e MAC associados a esta;
- exploração das portas abertas, fechadas, filtradas e não-filtradas da máquina;
- deteção de serviços (protocolos utilizados, nome da aplicação e versão) a correr nas portas abertas;
- deteção da família do sistema operativo associado à máquina.

Foi também utilizado o protocolo SNMP de forma a obter mais informação acerca das máquinas anteriormente exploradas, nomeadamente os endereços associados a estas e os seus tipos. Recorreu-se às tabelas de *routing* e ARP das máquinas para encontrar outras máquinas na rede e para fazer o mapeamento entre endereços IP e MAC. Aqui, é também descoberto o tipo de dispositivo que está a ser explorado tendo em conta o valor armazenado em *sysServices*. O diagrama apresentado na Figura 4.9 foi baseado no diagrama apresentado em [95], que descreve a forma de inferir o tipo do dispositivo tendo por base o valor guardado em *sysServices*.

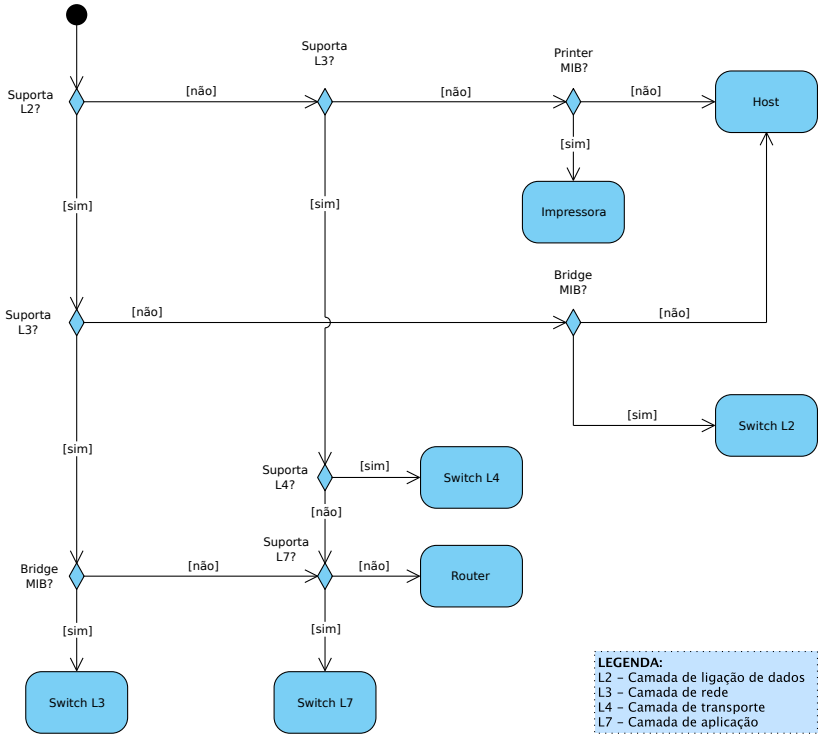


Figura 4.9: Algoritmo de descoberta do tipo de dispositivo.

Para obter esta informação é necessário que o utilizador forneça a *community string* que permite o acesso do dispositivo via SNMP, sendo que esta vai ser armazenada no *password vault*.

A informação recolhida utilizando o protocolo SNMP está dependente dos dados que as máquinas possam fornecer. Assim como para o protocolo LLDP, também neste caso é

necessário que a máquina que está a executar o processo de descoberta possua o protocolo SNMP configurado, de forma a poder tirar partido deste mecanismo de descoberta.

Finalmente, são criadas as redes que foram exploradas, tendo em conta os endereços ativos e as máscaras de rede obtidas pelos mecanismos anteriores, associando a estas as máquinas encontradas.

O Algoritmo 3 apresenta o processo de descoberta básica implementado.

Algoritmo 3 Processo de descoberta básica

```
explore_packets ← explorar os pacotes LLDP recebidos nas várias interfaces ativas
unlock_vault ← desbloquear o password vault
define_snmp_community ← definir as credencias SNMP
for ip de endereços a explorar do
    ping ← verificar se ip se encontra ativo
    if ip ativo then
        nmap ← explorar ip usando Nmap
        snmp ← explorar ip usando SNMP
    end if
end for
define_networks ← definir as redes e sub-redes exploradas
```

Categorias

Depois da descoberta básica da infraestrutura, é efetuada uma descoberta mais detalhada das máquinas, que é baseada num conjunto de categorias que o utilizador pretende explorar mais minuciosamente.

As categorias escolhidas para representar os vários pontos de exploração da infraestrutura foram as seguintes:

- **Rede:** conexões existentes e configurações de rede;
- **Dispositivos:** máquinas diretamente conetadas;
- **Sistemas Operativos:** sistema operativo detetado no dispositivo no instante da descoberta;
- **Processamento:** especificações dos diferentes processadores da máquina (por exemplo, CPU ou *Graphics Processing Unit (GPU)*);
- **Armazenamento:** sistemas de armazenamento e estado da memória dos mesmos;
- **Software:** produtos de sofware instalados na máquina;
- **Hardware:** especificações de hardware da máquina;
- **Máquinas Virtuais:** instâncias de máquinas virtuais a ser executadas e correspondente ambiente virtualizado;

- **Bases de dados:** sistemas de gestão de bases de dados e bases de dados existentes na máquina;
- **Serviços:** serviços detetados;
- **Containers:** instâncias de containers detetados;
- **Sistemas *Cloud*:** sistemas *Cloud* a ser utilizados;
- **Localização:** localização geográfica;
- **Pessoal:** informação acerca de utilizadores;
- **Documentos:** ficheiros guardados no dispositivo.

Descoberta detalhada

A descoberta detalhada tenta explorar as máquinas previamente descobertas, sendo que obtém informação de acordo com as categorias selecionadas pelo utilizador.

Para isto é necessário que o utilizador forneça as credenciais das várias máquinas, sendo que estas são também guardadas no *password vault*.

Neste caso foram implementados processos de descoberta baseados apenas no sistema operativo detetado na máquina a ser explorada. Desta forma, para obter informação acerca de outro tipo de dispositivos, poderiam ser implementados outros tipos de mecanismos de descoberta. Foram então implementados processos de descoberta em sistemas Microsoft Windows, macOS e Linux. Para estabelecer uma conexão com as máquinas Windows recorreu-se ao WinRM, e para as máquinas Linux e macOS foi utilizado SSH. Em todos os casos, são executados diferentes comandos, sendo depois a informação processada e guardada na base de dados. Para simplificar este processo não foram tidas em consideração as distribuições e versões dos sistemas operativos, sendo que não foram utilizados comandos diferentes, de acordo com estas, para descobrir a mesma informação. Assim, no caso em que não é possível executar o comando, a informação não é recolhida.

No caso da exploração em sistemas Microsoft Windows, foram utilizados os seguintes comandos de acordo com as categorias descritas anteriormente:

- **Sistema Operativo:** systeminfo;
- **Serviços:** wmic service.

Para a exploração sistemas macOS, recorreu-se à utilização dos seguintes comandos, que exploram diferentes categorias:

- **Sistema Operativo:** sw_vers e system_profiler SPSoftwareDataType;
- **Processamento:** system_profiler SPHardwareDataType e system_profiler SPDisplaysDataType;
- **Localização:** pedido HTTP a https://freegeoip.app/json/;
- **Armazenamento:** system_profiler SPMemoryDataType, system_profiler SPSerialATADataType e system_profiler SPStorageDataType;

- **Software:** system_profiler SPApplicationsDataType, system_profiler SPFrameworksDataType e system_profiler SPExtensionsDataType;
- **Hardware:** networksetup -listallhardwareports, system_profiler SPPowerDataType, system_profiler SPAudioDataType, system_profiler SPCameraDataType, system_profiler SPCardReaderDataType e system_profiler SPDisplaysDataType;
- **Rede:** system_profiler SPFirewallDataType e system_profiler SPNetworkDataType.

No caso dos sistemas Linux, foram utilizados os seguintes comandos, que são também associados a uma categoria:

- **Sistema Operativo:** cat /etc/*-release, hostnamectl e systemd-detect-virt;
- **Processamento:** lscpu;
- **Armazenamento:** blkid, df e lsblk;
- **Software:** top, dpkg-query -l.

A utilização e execução de mais comandos permite a descoberta de mais informação. Desta forma, para descobrir outro tipo de informação sobre as máquinas, seria apenas necessário processar os resultados da execução de outros comandos.

4.2.6 *Povoamento da base de dados*

Considerando que o GraphDB [26] permite a importação dos dados no formato RDF [14], e que o formato escolhido foi Turtle [58], para efetuar o povoamento da base de dados é necessário definir os itens de configuração e relacionamentos descobertos, assim como todos os atributos associados a estes. Neste processo é necessário criar primeiro os tipos de CIs e de relacionamentos, visto que estes são posteriormente atribuídos aos objetos em si. Para além disso, é também necessário começar por criar os CIs e só depois os relacionamentos. Este processo de povoamento da base de dados é apresentado no Algoritmo 4.

Algoritmo 4 Povoamento da base de dados

```
ci_types ← {}
for type em tipos de CIs descobertos do
    id ← type.get_id()
    db_id ← criar tipo de CI na BD
    ci_types[id] ← db_id
end for
rel_types ← {}
for type em tipos de relacionamentos descobertos do
    id ← type.get_id()
    db_id ← criar tipo de relacionamento na BD
    rel_types[id] ← db_id
end for
ci_ids ← {}
for ci em CIs descobertos do
    id ← ci.get_id()
    db_id ← criar CI na BD
    ci_ids[id] ← db_id
end for
for rel em relacionamentos descobertos do
    db_id ← criar relacionamento na BD
end for
```

No final, o ficheiro Turtle gerado é importado na base de dados através de um pedido HTTP POST onde é possível observar toda a informação descoberta. Para isto é necessário que o utilizador forneça informação do servidor onde a base de dados está a ser executada, da porta utilizada e do nome do repositório onde os dados vão ser armazenados.

Na Listagem 4.4 é possível observar um exemplo de um ficheiro Turtle com alguns CIs, relacionamentos, tipos e atributos descobertos.

Listagem 4.4: Exemplo em Turtle de CIs, relacionamentos, tipos e atributos a ser armazenados na base de dados.

```
:host16 rdf:type :ConfigurationItemType ;
    :title "host".

:layer_3_network101 rdf:type :ConfigurationItemType;
    :title "layer 3 network".

:part_of_network152 rdf:type :RelationshipType ;
   :title "part of network".
```

```
:11516 rdf:type :ConfigurationItem ;
   :status "up";
   :os_family "linux";
   :mac_address "52:54:00:f3:21:fc";
   :has_ipv4 "192.168.121.164";
   :has_attribute :221operating_system;
   :has_ci_type :host16.

:221operating_system rdf:type :Attribute ;
   :title "operating system";
   :value "cumulus linux".

:110101192_168_121_024 rdf:type :ConfigurationItem ;
   :title "192.168.121.024";
   :has_ci_type :layer_3_network101.

:169152part_of_network_192_168_121_024 rdf:type :Relationship ;
   :title "part of network 192.168.121.024";
   :has_source :11516;
   :has_target :110101192_168_121_024;
   :has_rel_type :part_of_network152.
```

4.2.7 *Processamento do modelo de dados da base de dados*

Após o povoamento da base de dados com a informação recolhida, é necessário fazer a conexão a esta para obter informação acerca do seu modelo de dados. O objetivo é obter os tipos de CIs e relacionamentos que existem, assim como os atributos associados a cada um destes, capturando a nomenclatura associada a cada elemento.

Tendo em consideração o esquema da base de dados definido na Secção 4.2.1 e que o motor utilizado foi o GraphDB, foi necessário efetuar interrogações *SPARQL Protocol and RDF Query Language (SPARQL)* para obter toda a informação necessária.

No caso dos tipos de CIs e relacionamentos, é apenas necessário capturar o nome associado a cada um destes. No caso dos atributos é necessário ter em consideração que existem atributos globais a todos os objetos e atributos que são criados de acordo com a informação descoberta. Desta forma, é necessário executar interrogações para estes dois conjuntos de atributos, quer para os CIs, quer para os relacionamentos. Os atributos devem ser diferenciados de acordo com o tipo de CI ou relacionamento a que estão associados.

Na Listagem 4.5 é apresentado o resultado do processamento do modelo de dados da BD, que tinha armazenado os dados presentes na Listagem 4.4.

Listagem 4.5: Exemplo do resultado do processamento do modelo de dados da base de dados.

```
{
   "ci_types":{
      "host":"host",
      "layer 3 network":"layer 3 network"
```

```
   },
   "rel_types":{
      "part of network":"part network"
   },
   "ci_attributes":{
      "host":{
         "operating system":"operating system",
         "mac_address":"media access control address",
         "has_ipv4":"internet protocol version 4",
         "os_family":"operating system family",
         "status":"status"
      },
      "layer 3 network":{
         "title":"title"
      }
   },
   "rel_attributes":{
      "part of network":{
         "title":"title"
      }
   }
}
```

4.2.8 *Processamento do modelo de dados da CMDB*

O objetivo desta etapa é obter a representação do modelo de dados da CMDB, capturando a nomenclatura associada a cada elemento. Para isso, é necessário efetuar algumas interrogações à CMDB de forma a obter informação acerca de CIs, relacionamentos e atributos. Isto pode ser efetuado através da API da mesma, nos casos em que esta forneça métodos para tal, ou acedendo diretamente à base de dados utilizada. Tendo em conta as diferenças identificadas na Secção 2.4.5 e o quão distintos podem ser os modelos de dados das CMDBs, foi desenvolvido um mecanismo específico para cada software. Desta forma, o custo de desenho da solução de processamento depende apenas da implementação do processo uma vez para cada tipo de software.

Nesta fase é necessário processar e analisar o modelo da CMDB, obtendo informação acerca:

- dos tipos de CIs existentes;
- dos tipos de relacionamentos existentes;
- dos atributos associados a cada tipo de CI;
- dos atributos associados a cada tipo de relacionamento;
- do tipo de dados de cada atributo;
- das restrições, caso existam, entre os tipos de componentes que podem estar envolvidos num relacionamento;

- dos valores possíveis para os atributos cujo valor está pré-definido.

i-doit

Para o caso específico do i-doit, foi definido um mecanismo de processamento que utiliza a sua API e a sua base de dados para obter informação sobre o seu modelo de dados. No caso da API, o utilizador necessita de fornecer informação acerca do servidor onde a CMDB está a ser executada, do nome de utilizador, da palavra-passe e da chave associada à API. Para a base de dados, é necessário que o utilizador indique o servidor onde está a ser executada, o nome da base de dados, o nome de utilizador e a palavra-passe.

Para obter maior parte da informação o mecanismo recorre à API, no entanto, necessita de aceder também a informação presente na base de dados no caso em que atributos referenciam outras instâncias de outras tabelas.

Para obter a informação necessária para capturar o modelo de dados, foram utilizados os seguintes métodos da API:

- idoit.constants: devolve os tipos de objetos e categorias, globais e específicas, existentes na CMDB;
- cmdb.category_info: devolve os atributos de uma categoria, assim como os seus tipos de dados;
- cmdb.dialog.read: devolve os valores possíveis para um determinado atributo de uma determinada categoria, cujos valores possíveis são pré-definidos na CMDB;
- cmdb.object_type_categories.read: devolve as categorias associadas a um tipo de objeto.

Na Listagem 4.6 é apresentado o código implementado para o envio de um pedido do tipo idoit.constants, sendo que as variáveis identificadas pelos nomes "server", "username", "password" e "apikey" correspondem, respetivamente, ao endereço do servidor onde a CMDB está a ser executada, ao nome de utilizador, à palavra-passe e à chave associada à API, fornecidos pelo utilizador.

Listagem 4.6: Exemplo de execução do método idoit.constants à API do i-doit.

```
import requests

api_url = "http://" + server + "/i-doit/src/jsonrpc.php"

headers = {}
headers["Content-Type"] = "application/json"
headers["X-RPC-Auth-Username"] = username
headers["X-RPC-Auth-Password"] = password

body = {}
body["version"] = "2.0"
body["method"] = "idoit.constants"
body["params"] = {}
```

```
body["params"]["apikey"] = apikey
body["params"]["language"] = "en"
body["id"] = 1

try:
    s = requests.Session()
    constants_request = s.post(api_url, json=body, headers=headers)
    constants = constants_request.json()
except requests.exceptions.RequestException:
    print("Unable to connect to the API. Please verify the connection information.")
```

Apesar de através da API ser possível obter os valores pré-definidos de alguns atributos, isto não é possível para todos os casos. Isto porque existem atributos que fazem referência a instâncias de outras tabelas. Desta forma, foi necessário recorrer a interrogações *Structured Query Language (SQL)* para obter este tipo de informação. Sabendo que a nomenclatura das colunas das tabelas segue o mesmo esquema em toda a base de dados, é possível inferir o nome das colunas das quais é necessário obter informação, tendo apenas o nome da tabela. É então apresentado na Listagem 4.7 o código implementado para recolher informação da base de dados do i-doit, onde as variáveis identificadas pelos nomes "server", "db_name", "username" e "password" correspondem, respetivamente, ao servidor onde está a ser executada a base de dados, ao nome da base de dados, ao nome de utilizador e à palavra-passe. A variável identificada pelo nome "table" corresponde ao nome da tabela de onde vai ser recolhida a informação, referente ao identificador e nome das instâncias nela presentes.

Listagem 4.7: Código implementado para recolha de informação de tabelas da base de dados do i-doit.

```
import mysql.connector
from mysql.connector import errorcode

try:
    connection = mysql.connector.connect(user=username, password=passwd, host=server, database=
        db_name)

    print(green + "\n>>> " + reset + "Successfully connected to the i-doit database.")

except mysql.connector.Error as err:
    if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
        print(red + "\n>>> " + reset + "Something is wrong with your username or password.")
    elif err.errno == errorcode.ER_BAD_DB_ERROR:
        print(red + "\n>>> " + reset + "Database does not exist.")
    else:
        print(red + "\n>>> " + reset + str(err))

if connection != None:
    cursor = connection.cursor()
```

```
values = {}
if table != None:
    name = str(table) + "__id"
    desc = str(table) + "__title"
    query = ("SELECT " + name + ", " + desc + " FROM " + db_name + "." + table + ";")

    cursor.execute(query)
    for t in cursor:
        name, value = t
        values[name] = value
```

O resultado parcial do processamento do modelo de dados da CMDB i-doit pode ser consultado na Listagem A.1 do Anexo A. Devido à enorme extensão do modelo, é apenas possível apresentar uma parte do resultado do processamento do mesmo.

iTop

Para o caso específico do iTop, foi definido um mecanismo de processamento que recorre a interrogações à sua base de dados para obter informação acerca do seu modelo de dados.

Tendo em conta que o SGBD utilizado pelo iTop é o MySQL, o utilizador necessita de fornecer informação acerca do servidor onde a base de dados está a ser executada, do nome de utilizador, da palavra-passe e do nome da base de dados, sendo que é também necessário que a máquina a executar o processo possa aceder a esta. Para além disto, o utilizador precisa também de fornecer informação acerca da API da CMDB, porque o posterior povoamento desta deve ser efetuado através da API e não da base de dados, para evitar possíveis erros e incoerências no povoamento. Para tal, o utilizador deve indicar o servidor onde a CMDB está a ser executada, assim como, o nome de utilizador e palavra-passe para aceder a esta.

Para obter a informação necessária para capturar o modelo de dados, foram utilizadas interrogações SQL à base de dados de forma a:

- obter as tabelas existentes na base de dados;
- obter os atributos associados a cada tabela;
- obter os tipos dos atributos associados a cada tabela.

Depois de obter as tabelas existentes na base de dados e, sabendo que as tabelas identificadas com o prefixo "*lnk*" correspondem a relacionamentos, é possível inferir que estas correspondem aos tipos de relacionamentos existentes no modelo de dados e que as restantes correspondem aos tipos de CIs. Além disto, os relacionamentos presentes no iTop apresentam restrições ao nível dos tipos de CIs envolvidos. Nas tabelas correspondentes aos relacionamentos é possível verificar que existem dois atributos que possuem o sufixo "_id", que identificam os CIs envolvidos no relacionamento. O restante nome destes atributos corresponde ao tipo do CI, sendo possível inferir as restrições dos tipos de CIs envolvidos em cada relacionamento.

4.2.9 *Cálculo de similaridade*

O coeficiente de similaridade mede a semelhança entre dois termos baseando-se na comparação sintática e semântica entre estes. No Algoritmo 5 é apresentado o processo implementado para o cálculo do coeficiente de similaridade entre dois termos. No final é obtido um valor entre zero e um, sendo que quanto maior for esse valor, maior é a semelhança entre os termos comparados. Este cálculo não atribui um peso às medidas de similaridade sintática e semântica, sendo que apenas devolve o maior valor calculado entre estes. Assim, o valor do coeficiente de similaridade poderia apresentar valores diferentes no caso de serem atribuídos diferentes pesos a estas medidas.

Algoritmo 5 Cálculo da similaridade entre dois termos

Require: $txt1 \neq None \wedge txt2 \neq None$
Ensure: $res \geq 0 \wedge res \leq 1$
  $syn \leftarrow$ calcular similaridade sintática entre $txt1$ e $txt2$
  if $syn == 1$ **then**
    $res \leftarrow syn$
    return res
  else
    $sem \leftarrow$ calcular similaridade semântica entre $txt1$ e $txt2$
    if $sem > syn$ **then**
      $res \leftarrow sem$
      return res
    else
      $res \leftarrow syn$
      return res
    end if
  end if

Similaridade sintática

O cálculo da similaridade sintática é especificado num coeficiente entre zero e um e mede a distância, com base na ocorrência de caracteres, entre dois termos. Quanto maior este valor, mais semelhantes são os termos a ser comparados.

Inicialmente, para calcular este coeficiente, é feito o tratamento do texto, ou seja, a sua normalização, como descrito na Secção 4.2.3. Como a informação é recolhida de várias fontes, possui formatos diferentes, sendo necessário colocá-la toda no mesmo formato de forma a poder compará-la.

Depois, é necessário converter a informação num formato numérico, isto é, calcular os vetores dos dados a serem comparados. De seguida, é possível obter a semelhança entre os vetores calculados. A técnica utilizada para este efeito foi a similaridade por cosseno, visto ser uma técnica comummente utilizada [77]. Esta comparação retorna um valor entre zero e um, sendo que o valor um representa que os termos que estão a ser comparados são

exatamente iguais, e o valor zero implica que não apresentam nenhuma semelhança entre si.

Definição 4.2.1 *"A similaridade por cosseno mede a semelhança entre dois vetores de um espaço interno de produto. É medido pelo cosseno do ângulo entre dois vetores e determina se esses vetores apontam aproximadamente na mesma direção. É usualmente utilizado para medir a similaridade entre documentos em análise de texto [74]."*

O Algoritmo 6 apresenta o processo implementado para o cálculo da similaridade sintática entre dois termos.

Algoritmo 6 Cálculo da similaridade sintática

Require: $txt1 \neq None \wedge txt2 \neq None$

Ensure: $res \geq 0 \wedge res \leq 1$

$t1 \leftarrow$ normalizar $txt1$

$t2 \leftarrow$ normalizar $txt2$

$vetores \leftarrow$ calcular vetores de $t1$ e $t2$

$res \leftarrow$ calcular similaridade por cosseno de $vetores$

return res

Similaridade semântica

O cálculo da similaridade semântica é especificado num coeficiente entre zero e um e mede a distância, ao nível do significado, entre dois termos.

Tal como para o cálculo da similaridade sintática, também aqui é necessário fazer o tratamento do texto, ou seja, a sua normalização, como descrito na Secção 4.2.3. Para o cálculo da similaridade semântica foi utilizada a medida proposta por Zhibiao Wu e Martha Palmer [112], implementada na biblioteca nltk [31], e apresentada na Definição 4.2.2. Esta medida devolve um valor que representa o quão semelhantes são os sentidos das palavras comparadas. Poderia ser utilizada outra medida para o cálculo da similaridade semântica, sendo que isso levaria à obtenção resultados diferentes.

Definição 4.2.2 *"A semelhança conceitual entre C1 e C2 é:*

$$ConSim(C1, C2) = \frac{2 * N3}{N1 + N2 + 2 * N3}$$

C3 é o conceito comum mais específico de C1 e C2. N1 é o número de nós no caminho de C1 a C3. N2 é o número de nós no caminho de C2 a C3. N3 é o número de nós no caminho de C3 até a raiz [112]."

O Algoritmo 7 apresenta o processo implementado para o cálculo da similaridade semântica entre dois termos.

Algoritmo 7 Cálculo da similaridade semântica

Require: $txt1 \neq None \wedge txt2 \neq None$
Ensure: $res \geq 0 \wedge res \leq 1$
  $t1 \leftarrow$ normalizar $txt1$
  $t2 \leftarrow$ normalizar $txt2$
  $max \leftarrow 0$
  $significados1 \leftarrow$ calcular sentidos possíveis de $t1$
  $significados2 \leftarrow$ calcular sentidos possíveis de $t2$
  for *sig1* em *significados1* **do**
    for *sig2* em *significados2* **do**
      sem ← calcular similaridade semântica entre *sig*1 e *sig*2
      if *sem* > *max* **then**
        max ← *sem*
      end if
    end for
  end for
  res ← *max*
  return *res*

Na Figura 4.10 são apresentados alguns exemplos do cálculo do coeficiente de similaridade entre termos.

```
>>> similarity.calculate_similarity("Host", "Server")
1.0
>>> similarity.calculate_similarity("Host", "host")
1.0
>>> similarity.calculate_similarity("A Layer 3 Network", "layer-3 net")
0.9999999999999998
>>> similarity.calculate_similarity("A Layer 3 Network", "net layer-3")
0.9999999999999998
>>> similarity.calculate_similarity("media access control address", "MAC_Address")
1.0
>>> similarity.calculate_similarity("media access control address", "MAC")
0.8660254037844388
```

Figura 4.10: Exemplos de utilização do mecanismo que calcula a similaridade entre dois termos.

4.2.10 *Mapeamento*

Neste passo devem ser geradas as regras de transformação que descrevem o mapeamento do modelo da base de dados no modelo da CMDB a utilizar. Tendo em conta que o mapeamento dos modelos de forma manual é um processo complexo, devido à grande dimensão e complexidade destes, foi necessário desenvolver um mecanismo que executasse esta tarefa automaticamente. Este mecanismo recorre ao cálculo de similaridade descrito na Secção 4.2.9 e aos modelos de dados obtidos nas fases de processamento dos mesmos.

Inicialmente são calculados os coeficientes de similaridade entre todos os tipos de CIs e relacionamentos existentes nos dois modelos. Depois de obter todos os valores e de

os ordenar de forma decrescente, são selecionadas as combinações mais semelhantes, ou seja, as que apresentam um maior valor do coeficiente calculado. Para o caso em que são encontradas combinações com o mesmo valor de semelhança, foi dada a opção de escolha ao utilizador. As seleções efetuadas implicam o mapeamento único entre os elementos dos dois modelos, ou seja, um elemento de um modelo, mapeia apenas no elemento do outro modelo cujo valor do coeficiente de similaridade é o máximo calculado. A mesma metodologia é depois aplicada aos atributos de cada tipo de componente e relacionamento. No final é apresentado o mapeamento construído, assim como o valor de similaridade calculado. Nesta fase o utilizador deve definir um valor limite para seleção final das regras de transformação. Para todos os coeficientes com valor inferior ao limite definido pelo utilizador, os mapeamentos vão ser descartados. Assim, um maior valor do limite garante a seleção de elementos mais semelhantes.

Na Figura 4.11 é possível observar o processo de mapeamento realizado entre o modelo da base de dados presente na Listagem 4.5 e o modelo da CMDB presente na Listagem A.1 do Anexo A. Na Listagem 4.8 é possível observar as regras de transformação geradas neste mapeamento, considerando que o valor limite escolhido foi 0.80 (zero ponto oitenta).

```
>>> Executing the model mapper...

>>> Calculating configuration item types similarity...
? The similarities between 'host' with 'C__OBJTYPE__SERVER' and 'C__OBJTYPE__HOST' are equal. Choose the one to consider.  C__OBJTYPE__HOST
? The similarities between 'layer 3 network' with 'C__OBJTYPE__LAYER3_NET' and 'C__OBJTYPE__LAYER2_NET' are equal. Choose the one to consider.  C__OBJTYPE__LAYER3_NET

>>> Calculating relationship types similarity...

>>> Calculating configuration item attributes similarity...

>>> Calculating relationship attributes similarity...

================================================================================================================================================================
CONFIGURATION ITEMS MAPPING
================================================================================================================================================================

CI in CMDB              Description    CI in DB         Description       Similarity Coeficient
----------------------  -------------  ---------------  ----------------  -----------------------
C__OBJTYPE__HOST        Host           host             host                                  1
C__OBJTYPE__LAYER3_NET  Layer 3-net    layer 3 network  layer 3 network                       1

********************************************************************************************************
C__OBJTYPE__HOST Attributes Mapping
********************************************************************************************************

Attribute in CMDB    Description     Attribute in DB    Description                    Similarity Coeficient
-------------------  --------------  -----------------  -----------------------------  -----------------------
status               Condition       status             status                                             1
mac                  MAC             mac_address        media access control address                      0.866025
application          Application     operating system   operating system                                   0.842105
system_drive         System drive    os_family          operating system family                            0.766736
assigned_version     Version number  has_ipv4           internet protocol version 4                        0.672933

********************************************************************************************************
C__OBJTYPE__LAYER3_NET Attributes Mapping
********************************************************************************************************

Attribute in CMDB    Description    Attribute in DB    Description     Similarity Coeficient
-------------------  -------------  -----------------  --------------  -----------------------
title                Title          title              title                                 1

================================================================================================================================================================
RELATIONSHIPS MAPPING
================================================================================================================================================================

Relationship in CMDB                Description          Relationship in DB    Description      Similarity Coeficient
----------------------------------  -------------------  --------------------  -------------  -----------------------
C__RELATION_TYPE__NET_CONNECTIONS   Network connections   part of network       part network                  0.817846

********************************************************************************************************
C__RELATION_TYPE__NET_CONNECTIONS Attributes Mapping
********************************************************************************************************

Attribute in CMDB    Description    Attribute in DB    Description     Similarity Coeficient
-------------------  -------------  -----------------  --------------  -----------------------
title                Title          title              title                                 1

? Enter the threshold value that you want to consider (similarities below that value will not be considered):  0.8
```

Figura 4.11: Exemplo de execução do mecanismo de mapeamento entre dois modelos.

Listagem 4.8: Regras de transformação entre modelos.

```
{
   "ci_types":{
      "host":"C__OBJTYPE__HOST",
      "layer 3 network":"C__OBJTYPE__LAYER3_NET"
   },
   "rel_types":{
      "part of network":"C__RELATION_TYPE__NET_CONNECTIONS"
   },
   "ci_attributes":{
      "host":{
         "status":"status",
         "mac_address":"mac",
         "operating system":"application"
      },
      "layer 3 network":{
         "title":"title"
      }
   },
   "rel_attributes":{
      "part of network":{
         "title":"title"
      }
   }
}
```

4.2.11 *Povoamento da CMDB*

O objetivo desta etapa consiste em enviar à CMDB a informação recolhida acerca da infraestrutura durante o processo de descoberta. Para isto é necessário aceder à base de dados para obter esta informação e, recorrendo às regras de transformação calculadas no processo de mapeamento, povoar a CMDB. É preciso também ter em consideração que este povoamento deve ser efetuado através da API da CMDB para evitar erros e incoerências nos dados.

Tendo em conta as diferenças entre as várias CMDBs identificadas na Secção 2.4.5 e o quão distintas podem ser as APIs, foi desenvolvido um mecanismo específico para cada software. Desta forma, o custo de desenho da solução de povoamento depende apenas da implementação do processo uma vez para cada tipo de software.

Durante este processo é relevante ter em atenção que os CIs devem ser os primeiros a ser criados, visto que estes estão envolvidos nos relacionamentos. Ou seja, para criar um relacionamento é necessário que os CIs neste envolvidos sejam previamente criados na CMDB, assim como é preciso ter informação acerca do seu identificador na CMDB. Este processo é apresentado no Algoritmo 8.

Algoritmo 8 Povoamento da CMDB

```
cis_ids ← obter CIs existentes na base de dados
for ci em cis_ids do
    cis_types[ci] ← obter o tipo do ci
    cis_attributes[ci] ← obter os atributos do ci
end for
rels_ids ← obter relacionamentos existentes na base de dados
for rel em rels_ids do
    rels_types[rel] ← obter o tipo do rel
    rels_attributes[rel] ← obter os atributos do rel
    sources[rel] ← obter o CI fonte do rel
    targets[rel] ← obter o CI alvo do rel
end for
for ci em cis_types do
    ci_id ← criar ci na CMDB
end for
for rel em rels_types do
    rel_id ← criar rel na CMDB
end for
```

i-doit

No caso específico do i-doit, foi utilizado o método `cmdb.object.create` que permite a criação de componentes e relacionamentos através da sua API.

O povoamento é iniciado com a criação dos CIs existentes na base de dados. Para cada CI descoberto, é verificado o seu tipo. Caso não exista uma correspondência do seu tipo nas regras de mapeamento, este não é criado. Se existir, são definidos os seus atributos. Para cada um destes é verificado o seu tipo nas regras de transformação. Se estes corresponderem a um atributo que permite apenas determinados valores pré-definidos, é necessário fazer a correspondência entre o valor descoberto e os valores permitidos. Noutro caso, é apenas necessário converter o valor do atributo no tipo que é utilizado pela CMDB. No entanto, pode acontecer de não ser possível converter o valor do atributo para o tipo utilizado pela CMDB, não sendo possível enviar esta informação no pedido. Quando o CI é finalmente definido, segue-se o envio de um pedido para a sua criação. Para cada CI criado, é guardado o seu identificador na CMDB para utilizar como referência nos relacionamentos.

Depois de todos os CIs criados, segue-se a criação dos relacionamentos. Tal como para os CIs, para cada relacionamento é inicialmente verificada a existência do seu tipo nas regras de transformação. Caso exista uma correspondência são depois verificados os CIs envolvidos no relacionamento. É necessário que estes tenham sido previamente criados na CMDB. Novamente, para cada atributo é verificado o seu tipo nas regras de transformação. Se estes corresponderem a um atributo que permite apenas determinados valores pré-definidos, é necessário fazer a correspondência entre o valor descoberto e os valores permitidos. Caso contrário, é apenas necessário converter o valor do atributo no tipo que é utilizado pela

CMDB, sendo que nos casos em que isto não é possível, a informação não é enviada. Quando o relacionamento é definido, segue-se o envio de um pedido para a sua criação.

De salientar que o método cmdb.object.create não permite que o valor do parâmetro referente ao nome seja enviado a vazio. Por isso, tanto na criação dos CIs como dos relacionamentos, é utilizado o nome do tipo do objeto como nome padrão. No entanto, caso o objeto apresente um atributo na BD associado ao nome, este valor substitui o definido anteriormente. Desta forma, é garantido que o pedido é enviado. Para além disto, os atributos têm de ser enviados de acordo com a categoria a que pertencem, sendo no momento da definição do pedido necessário associar os atributos às categorias a que pertencem. Na Listagem 4.9 é apresentado um exemplo da estrutura, em JSON, de um pedido do tipo cmdb.object.create para a criação de um objeto.

Listagem 4.9: Exemplo da estrutura JSON de um método cmdb.object.create para a criação de um objeto no i-doit.

```
{
   "version": "2.0",
   "method": "cmdb.object.create",
   "params": {
        "type": "C__OBJTYPE__HOST",
        "title": "Host example",
        "apikey": "xxx",
        "language": "en",
        "categories": {
            "C__CATG__CPU": [{
                "frequency": "2.0",
                'cores': 4
            }],
            "C__CATG__NETWORK": [{
                "serial": "abcde"
            }],
            "C__CATG__GLOBAL": [{
                "description": "Description of host example.",
                'status': 7
            }]
        }
    },
    'id': 1
}
```

Recorrendo às regras de transformação presentes na Listagem 4.8, e aos dados presentes na Listagem 4.4, foi executado o povoamento da CMDB. É possível observar a execução deste na Figura 4.12 e o resultado na CMDB na Figura 4.13.

```
>>> Obtaining the existing CI's in the database...
>>> Obtaining the existing relationships in the database...
>>> Starting the population of the CMDB...
>>> Creating the configuration items...
>>> Error converting string 'cumulus linux' to int.
>>> Object of type C__OBJTYPE__HOST created successfully in the CMDB.
>>> Object of type C__OBJTYPE__LAYER3_NET created successfully in the CMDB.
>>> Creating the relationships...
>>> Object of type C__RELATION_TYPE__NET_CONNECTIONS created successfully in the CMDB.
>>> Successfully disconnected.
>>> CMDB population completed.
```

Figura 4.12: Exemplo de execução do povoamento da CMDB i-doit.

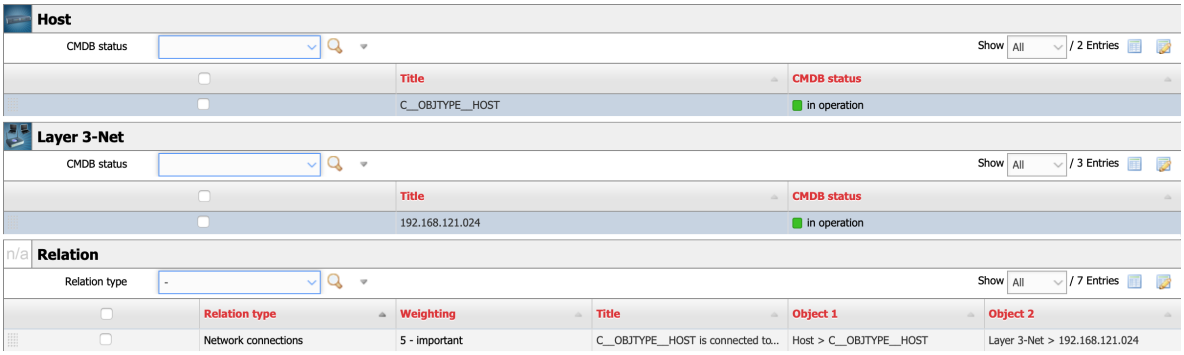


Figura 4.13: Exemplo de objetos criados no i-doit.

iTop

No caso do povoamento do iTop, foi utilizado o método core/create da sua API.

O povoamento segue o mesmo procedimento que foi descrito para o i-doit. No entanto, apresenta uma diferença no processo de criação dos relacionamentos. Neste caso, como os relacionamentos apresentam restrições ao nível do tipo de CIs envolvidos, é necessário fazer a verificação destas, para perceber se é possível criar o relacionamento com os CIs que lhe estão associados.

Assim como para o i-doit, também no iTop a informação dos pedidos é enviada no formato JSON. Na Listagem 4.10 é apresentado um exemplo da estrutura de um pedido do tipo core/create para a criação de um CI.

Listagem 4.10: Exemplo da estrutura JSON de um método core/create para a criação de um objeto no iTop.

```
{
   "operation":"core/create",
   "comment":"Synchronization from CMDB automatic creation...",
   "user":"itop",
```

```
   "password":"itop-WebServices1",
   "class":"Server",
   "fields":{
        "title": "Server example",
        "description": "Description of server example"
        "cpu":"server cpu",
        "ram":"16"
   }
}
```

4.3 Sumário

Neste capítulo foi apresentada a metodologia implementada no desenvolvimento da ferramenta capaz de fazer a criação automática de uma CMDB. A linguagem de programação utilizada para a implementação da ferramenta foi Python, e a base de dados escolhida foi GraphDB. É apresentada na Figura 4.14 a arquitetura do sistema implementado que apresenta a interação entre os diversos elementos do processo.

Como já referido na Secção 3.2, o processo de criação foi dividido em três fases distintas: descoberta, mapeamento e povoamento.

A fase de descoberta começa pela configuração do *password vault*, que vai armazenar as palavras-passes utilizadas ao longo de todo o processo. Para isto, é necessário que o utilizador defina a palavra-passe do *password vault*. De seguida, é necessário que o utilizador indique se a descoberta vai obter informação de uma ferramenta externa. Em caso positivo, o utilizador deve indicar a ferramenta a utilizar e a forma de aceder à informação recolhida por esta. Foi escolhida a ferramenta Angry IP Scanner, abordada na Secção 2.5.4, para provar este conceito. Tendo em conta que a descoberta se baseia num conjunto de endereços, é também necessário que o utilizador defina o(s) endereço(s) que pretende que sejam explorados na descoberta. É então iniciada a descoberta básica. Segue-se a captura de pacotes LLDP. De seguida, segue-se a exploração dos vários endereços recorrendo à execução de comandos *ping*, à ferramenta Nmap e ao protocolo SNMP, sendo que o utilizador pode definir as credenciais SNMP das máquinas da infraestrutura caso pretenda que seja obtido este tipo de informação. De seguida, segue-se a descoberta detalhada. Esta descoberta tem por base as categorias selecionadas pelo utilizador, e as credenciais das máquinas disponibilizadas por este. Para cada máquina e em cada categoria selecionada, é recolhida a informação, sendo que para isto são utilizados os protocolos WinRM e SSH para efetuar a conexão com a máquina. No final da descoberta, o utilizador deve indicar os dados de acesso à base de dados GraphDB a utilizar, e esta é povoada com a informação recolhida.

Na fase de mapeamento o utilizador deve indicar a CMDB que pretende utilizar e a informação necessária para aceder a esta. Segue-se então o processamento dos modelos de dados da base de dados e da CMDB, sendo que foram apenas implementados mecanismos capazes de processar os modelos de dados de duas CMDBs: i-doit e iTop. Depois do processamento, segue-se o mapeamento entre os modelos. Durante o mapeamento, caso sejam encontrados mapeamentos com o mesmo valor, o utilizador deve selecionar o(s) que considera mais adequado(s). No final são apresentados todos os mapeamentos calculados e o utilizador deve definir o valor limite para a seleção dos mapeamentos. Com base neste valor, são definidas as regras de transformação.

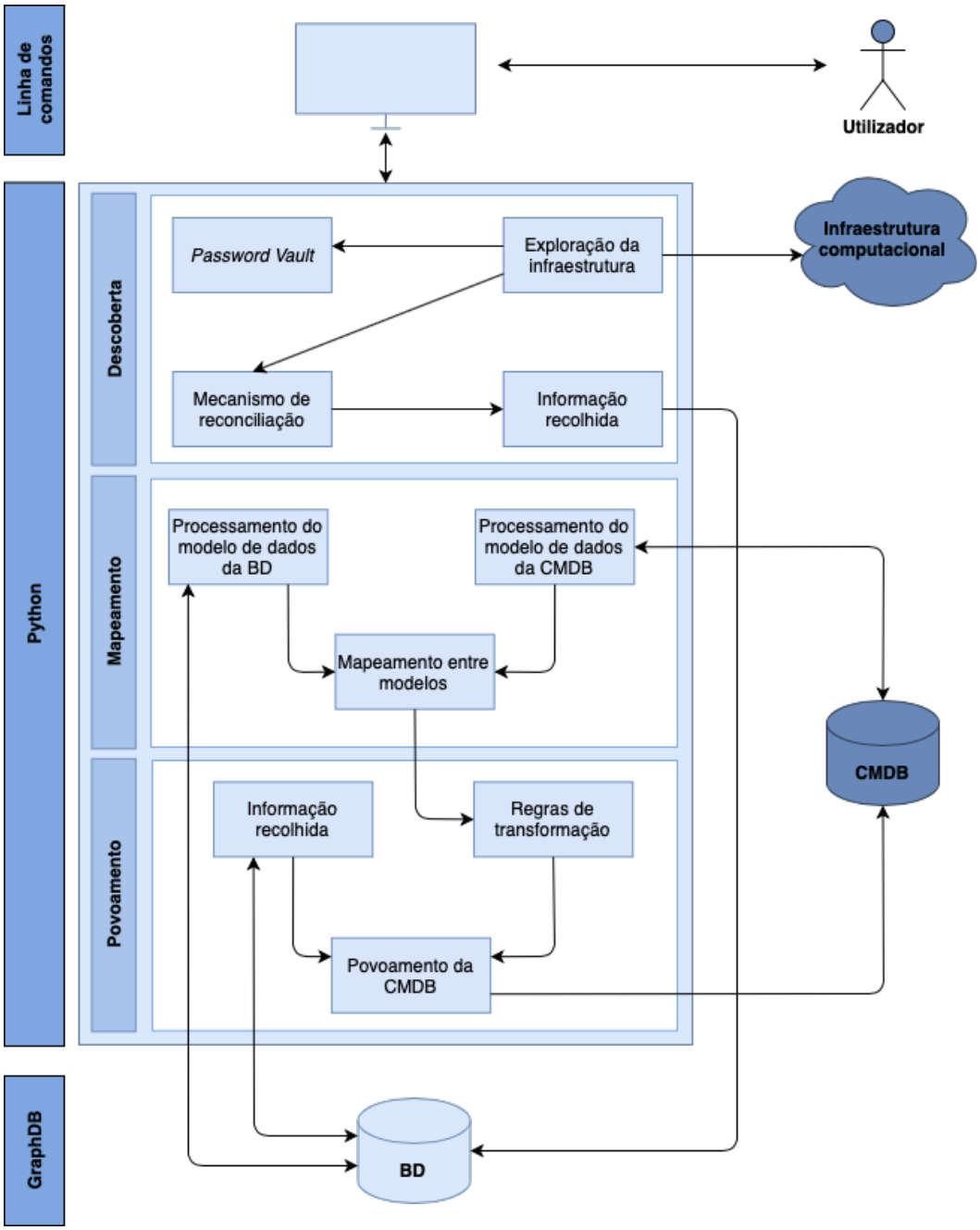


Figura 4.14: Arquitetura do sistema.

Finalmente, na fase de povoamento, é obtida toda a informação armazenada na base de dados e, recorrendo às regras de transformação entre os modelos, é efetuado o povoamento da CMDB.

No Anexo B é apresentado um exemplo de execução da ferramenta. Na Figura B.1 é apresentada a execução da fase de descoberta. Na Figura B.2 é apresentado o processamento e mapeamento dos modelos de dados da BD e da CMDB na fase de mapeamento. Ainda desta fase, são apresentados os mapeamentos dos CIs na Figura B.3 e dos relacionamentos na Figura B.4. Finalmente, na Figura B.5, é apresentada a execução da fase de povoamento.

Tendo em conta o desenvolvimento da plataforma, a próxima fase passa pela avaliação da mesma.

5

AVALIAÇÃO

Este capítulo apresenta os testes efetuados à ferramenta desenvolvida. Estes abordam as variações possíveis oferecidas pela ferramenta na execução do processo de criação automática de uma CMDB.

Com a execução desta avaliação é fundamental perceber se:

- O processo de descoberta é capaz de recolher a informação requerida pelo utilizador?
- Os resultados do mapeamento retratam a correspondência entre os elementos mais semelhantes dos dois modelos?
- A informação é enviada para a CMDB?

5.1 CASO DE TESTE 1

No primeiro caso de teste, a ferramenta vai ser executada de forma a recolher informação de apenas uma máquina física. As características da máquina a ser testada podem ser consultadas na Tabela 5.1. O processo vai ser executado nesta mesma máquina, sendo que a base de dados se encontra também a ser executada nesta. A CMDB utilizada, que neste caso será a i-doit 1.14 Open, é executada na máquina cujas características se encontram na Tabela 5.7. É relevante referir que, como as máquinas se encontram na mesma rede, são capazes de trocar informação entre si, permitindo que a primeira aceda à CMDB a ser executada na segunda.

Tabela 5.1: Características da máquina para o primeiro caso de teste.

Máquina	MacBook Pro
Sistema Operativo	macOS Catalina 10.15.7
Endereço IP	192.168.1.73
RAM	16 *GB*
CPU	2,2 *GHz* Intel Core i7 quad-core
Nmap	7.91
LLDP	Não instalado
SNMP	Não instalado
SSH	OpenSSH_8.1p1, LibreSSL 2.7.3
Python	3.9.1
GraphDB	9.3.1

5.1.1 *Descoberta*

A descoberta teve por base o endereço IP referente à máquina descrita na Tabela 5.1. Na descoberta básica, tendo em conta que a máquina não possui os protocolos LLDP e SNMP instalados, foi recolhida informação utilizando apenas a ferramenta Nmap, não tendo sido descobertas outras máquinas. Na descoberta detalhada foram selecionadas as categorias referentes à rede, sistema operativo, processamento, armazenamento, software, hardware e localização, sendo que as credenciais da máquina foram necessárias para a obtenção desta informação.

Devido à grande extensão dos dados recolhidos, vão ser apenas apresentados resultados parciais. Tendo em conta a máquina e as categorias selecionadas, é possível observar na Tabela 5.2 os itens que seria de esperar que fossem descobertos, e aqueles que foram efetivamente criados. Aqui são apenas apresentados os nomes associados às instâncias armazenadas na base de dados, sendo que os restantes atributos não estão representados. Também para o tipo de cada CI é apenas apresentado o nome que tem associado. Na Tabela C.1 do Anexo C é possível observar alguns atributos acerca de determinados itens descobertos.

Tabela 5.2: Alguns dos CIs encontrados na fase de descoberta no primeiro caso de teste.

Esperado	**Categoria**	**Encontrado**	
		Nome do tipo de CI	**Nome dos CIs**
Máquina descrita na Tabela 5.1	—	Host	macbookprojoana
Conexões de rede	Rede	Hardware Port	Thunderbolt 2 AirPort Ethernet
		Thunderbolt Bridge	Thunderbolt Bridge
		Thunderbolt Ethernet	Thunderbolt Ethernet
		VPN Uminho	VPN Uminho
		Wi-Fi	Wi-Fi
		Firewall	spfirewall_settings
		Bluetooth PAN	Bluetooth PAN
Sistema Operativo macOS	Sistema Operativo	Operating System	macOS 10.15.7 19H15
CPU e GPU	Processamento	CPU	Quad-Core Intel Core i7
		GPU	kHW_IntelIrisProItem
Disco e Memória *RAM*	Armazenamento	RAM	BANK 1DIMMo
		SSD Controller	APPLE SSD SM0256G
		Volume	Macintosh HD - dados
			Macintosh HD
Aplicações (apenas as utilizadas no último mês)	Software	Application	zoom Postman Adobe Acrobat Reader DC iMovie Firefox PyCharm Visual Studio Code Spotify Google Chrome FileZilla MRT Slack
Características de hardware	Hardware	Built-in Microphone	Built-in Microphone
		Built-in Output	Built-in Output
		Battery	bq20z451
		spcardreader	spcardreader
		Display	Color LCD
Localização da máquina	Localização	Location	PT

Na Tabela 5.3 é possível observar os atributos do CI que representa a máquina descrita na Tabela 5.1 que foram recolhidos na fase de descoberta.

Tabela 5.3: Informação recolhida sobre a máquina explorada na fase de descoberta no primeiro caso de teste.

Atributo	Valor
title	macbookprojoana
hostname	macbookprojoana.local
status	up
uuid	0B793129-B10E-5C77-89DD-4529FEF05AD6
serial number	C02S80K6G8WN
description	MacBook Pro MacBook ProMacBook Pro
mac address	c4:b3:01:c0:1a:d3
ipv4	192.168.1.73
ipv6	2001:8a0:f576:7000:1cac:32c1:d287:1804
ipv6	2001:8a0:f576:7000:19c8:36e6:6c25:e634
os family	macOS
os name	Mac OS X
os version	10.15.7
os	macOS 10.15.7 (19H15)
kernel version	Darwin 19.6.0
boot mode	normal_boot
time since boot	up 1:8:11:46
CPU	Quad-Core Intel Core i7
CPU cores	4
CPU speed	2,2 GHz
CPU frequency	2,2
CPU frequency unit	GHz
physical memory	16 GB
memory ram	16
physical memory unit	GB
GPU	kHW_IntelIrisProItem
battery	bq20z451
display	Color LCD

Também na fase de descoberta foi possível encontrar relacionamentos entre os CIs. Na Tabela 5.4 é possível observar alguns dos relacionamentos presentes na base de dados, sendo exibidos os nomes dos CIs envolvidos no relacionamento e o nome e tipo do relacionamento.

5.1.2 *Mapeamento*

Na fase de mapeamento foram analisados os modelos de dados da CMDB e da base de dados gerada no processo de descoberta, tendo sido produzidos potenciais mapeamentos entre estes modelos. Tendo em conta que neste caso foi utilizado o i-doit, é possível consultar na Listagem A.1 do Anexo A o resultado parcial do processamento do seu modelo de dados.

Considerando que o valor limite escolhido para a seleção das regras de transformação foi 0.85 (zero ponto oitenta e cinco), é possível observar na Tabela 5.5 as regras de transformação geradas.

Tabela 5.4: Alguns dos relacionamentos encontrados na fase de descoberta no primeiro caso de teste.

Nome do CI Fonte	Nome do CI Alvo	Tipo do Relacionamento	Nome do Relacionamento
macOS 10.15.7 19H15	macbookprojoana	running os	macOS 10.15.7 19H15 running os macbookprojoana
macbookprojoana	macOS 10.15.7 19H15	installed os	macbookprojoana installed os macOS 10.15.7 19H15
macbookprojoana	Quad-Core Intel Core i7	associated processor	macbookprojoana associated processor Quad-Core Intel Core i7
ssh	22	is running on port	ssh is running on port 22
APPLE SSD SM0256G	macbookprojoana	has storage	APPLE SSD SM0256G is storage of macbookprojoana
Visual Studio Code	macbookprojoana	has installed	Visual Studio Code installed on macbookprojoana
Color LCD	macbookprojoana	display of	Color LCD display of macbookprojoana
Built-in Microphone	Apple Inc.	has manufacturer	Built-in Microphone has manufacturer Apple Inc.
macbookprojoana	PT	located	macbookprojoana located PT
PT	macbookprojoana	location of	PT location of macbookprojoana

Tabela 5.5: Regras de transformação geradas no primeiro caso de teste.

		CMDB		Base de dados		Coeficiente de
		Nome	Descrição	Nome	Descrição	similaridade
CIs		C_OBJTYPE_APPLICATION	Application	Application	application	1
		C_OBJTYPE_HOST	Host	Host	host	1
		C_OBJTYPE_OPERATING_SYSTEM	Operating System	Operating System	operating system	1
		C_OBJTYPE_MONITOR	Monitor	Display	display	0.947368
		C_OBJTYPE_WORKSTATION	Workplace	Location	location	0.933333
		C_OBJTYPE_AMPLIFIER	Amplifier	CPU	central processing unit	0.888889
		C_OBJTYPE_CONVERTER	Converter	Charger	charger	0.875
Atributos dos CIs	C_OBJTYPE_APPLICATION	title	Title	title	title	1
		description	Description	description	description	1
	C_OBJTYPE_HOST	frequency	CPU frequency	CPU frequency	central processing unit frequency	1
		cores	CPU cores	CPU cores	central processing unit cores	1
		title	Title	title	title	1
		description	Description	description	description	1
		status	Condition	status	status	1
		frequency_unit	CPU frequency unit	CPU frequency unit	central processing unit frequency unit	1
		serial	Serial number	serial_number	serial number	1
		unit	Memory unit	physical memory unit	physical memory unit	0.928369
		memory	Memory	memory ram	memory random access memory	0.893177
		plug_type	Plug	battery	battery	0.888889
		speed_unit	Speed unit	CPU speed	central processing unit speed	0.877123
		device	On device	display	display	0.875
		mac	MAC	mac_address	media access control address	0.866025
	C_OBJTYPE_OPERATING_SYSTEM	assigned_version	Version number	version number	version number	1
		title	Title	title	title	1
	C_OBJTYPE_MONITOR	resolution	Resolution	resolution	resolution	1
		depth	Depth	depth	depth	1
		title	Title	title	title	1
		connection_type	Connection type	connection type	connection type	1
		date	Date	year	year	0.857143
		size	Display	display type	display type	0.851235
	C_OBJTYPE_WORKSTATION	latitude	Latitude	latitude	latitude	1
		longitude	Longitude	longitude	longitude	1
		title	Title	title	title	1
		parent	Location	region	region	0.888889
	C_OBJTYPE_AMPLIFIER	title	Title	title	title	1
		changes	Changes	Speed	speed	0.875
	C_OBJTYPE_CONVERTER	assigned_connector	Connected to	connected	connected	1
		image	File	charging	charging	1
Relacionamentos		C_RELATION_TYPE_NETWORK_PORT	Ports	port from	port	1
		C_RELATION_TYPE_LOCATION	Location	location of	location	1
		C_RELATION_TYPE_OPERATION_SYSTEM	Operating system	running os	running operating system	0.942343
Atributos dos Relacionamentos	C_RELATION_TYPE_NETWORK_PORT	title	Title	title	title	1
	C_RELATION_TYPE_LOCATION					
	C_RELATION_TYPE_OPERATION_SYSTEM					

5.1.3 *Povoamento*

Tendo em conta os itens descobertos e as regras de transformação geradas, foi possível comprovar a criação dos objetos que corresponderam a esses requisitos. É possível observar

na Tabela 5.6, as instâncias criadas na CMDB e os seus tipos associados. No caso dos relacionamentos, só são criados se os CIs neles envolvidos tenham também sido criados. Desta forma, mesmo que exista um mapeamento para um tipo de relacionamento, este não vai ser criado na CMDB se não houver um mapeamento possível para os tipos de CIs envolvidos neste. Para além disto, quando não é possível converter os tipos de dados dos atributos para os utilizados pela CMDB, também não é possível enviar essa informação. Por exemplo, no caso do CI que corresponde ao sistema operativo macOS 10.15.7, na base de dados este possui o atributo "version number" com o valor "10.15.7". O mapeamento deste atributo é feito com o atributo "assigned_version" na CMDB, que corresponde a um valor inteiro. Desta forma, esta informação não é enviada no pedido à API visto não ser possível converter este valor para um valor inteiro.

Tabela 5.6: Elementos criados na CMDB no primeiro caso de teste.

	Tipo na CMDB	Tipo na Base de Dados	Nome dos objetos
CIs	C_OBJTYPE_APPLICATION	Application	Firefox Postman Slack zoom Adobe Acrobat Reader DC iMovie PyCharm Visual Studio Code Spotify Google Chrome FileZilla MRT
	C_OBJTYPE_HOST	Host	macbookprojoana
	C_OBJTYPE_OPERATING_SYSTEM	Operating System	macOS 10.15.7 19H15
	C_OBJTYPE_MONITOR	Display	Color LCD
	C_OBJTYPE_WORKSTATION	Location	PT
	C_OBJTYPE_AMPLIFIER	CPU	Quad-Core Intel Core i7
	C_OBJTYPE_CONVERTER	Charger	C_OBJTYPE_CONVERTER
Relacionamentos	C_RELATION_TYPE_OPERATION_SYSTEM	running operating system	macOS 10.15.7 19H15 running os macbookprojoana
	C_RELATION_TYPE_LOCATION	location of	PT location of macbookprojoana

Na Listagem 5.1 é possível observar a informação em JSON do pedido gerado para a criação do CI correspondente à máquina descrita na Tabela 5.1. Este foi gerado tendo por base a informação acerca do CI presente na base de dados, que pode ser consultada na Tabela 5.3, e as regras de transformação presentes na Tabela 5.5. Na Figura 5.1 é possível verificar a criação deste CI na CMDB.

Listagem 5.1: Informação JSON do método cmdb.object.create gerado para a criação do CI correspondente à máquina descrita na Tabela 5.1.

```
{
    'version': '2.0',
    'method': 'cmdb.object.create',
    'params': {
        'apikey': 'cmdb-auto',
```

```
        'language': 'en',
        'type': 'C__OBJTYPE__HOST',
        'title': 'C__OBJTYPE__HOST',
        'categories': {
            'C__CATG__CONTROLLER_FC_PORT': [{
                'speed_unit': 1
            }],
            'C__CATG__CPU': [{
                'frequency': 2.0,
                'cores': 4,
                'frequency_unit': 3
            }],
            'C__CATG__GRAPHIC': [{
                'memory': 16.0
            }],
            'C__CATG__NETWORK_PORT': [{}],
            'C__CATG__DRIVE': [{}],
            'C__CATG__MEMORY': [{
                'unit': 3
            }],
            'C__CATG__NETWORK': [{
                'serial': 'C02S80K6G8WN'
            }],
            'C__CATG__GLOBAL': [{
                'title': 'macbookprojoana',
                'description': 'MacBook Pro MacBook ProMacBook Pro',
                'status': 7
            }],
            'C__CATG__NETWORK_LOG_PORT': [{
                'mac': 'c4:b3:01:c0:1a:d3'
            }]
        }
    },
    'id': 1
}
```

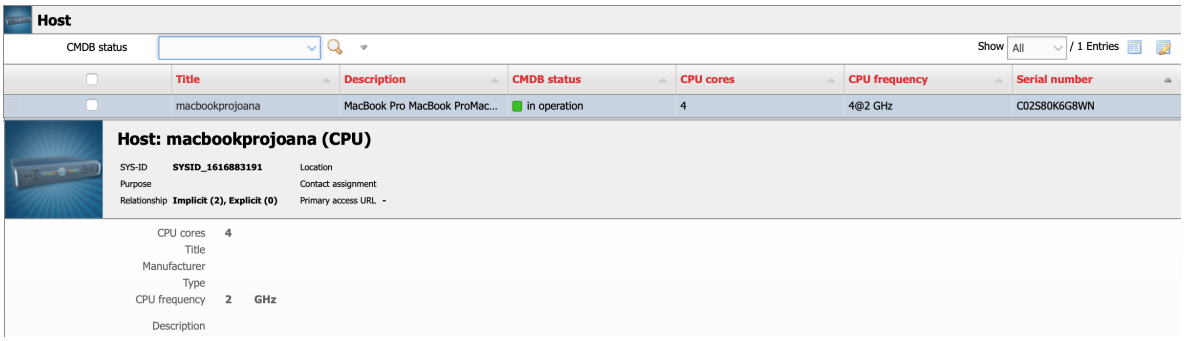


Figura 5.1: Objeto referente à máquina descrita na Tabela 5.1 criado na CMDB no primeiro caso de teste.

Na Figura 5.2 é também possível observar os restantes CIs criados na CMDB.

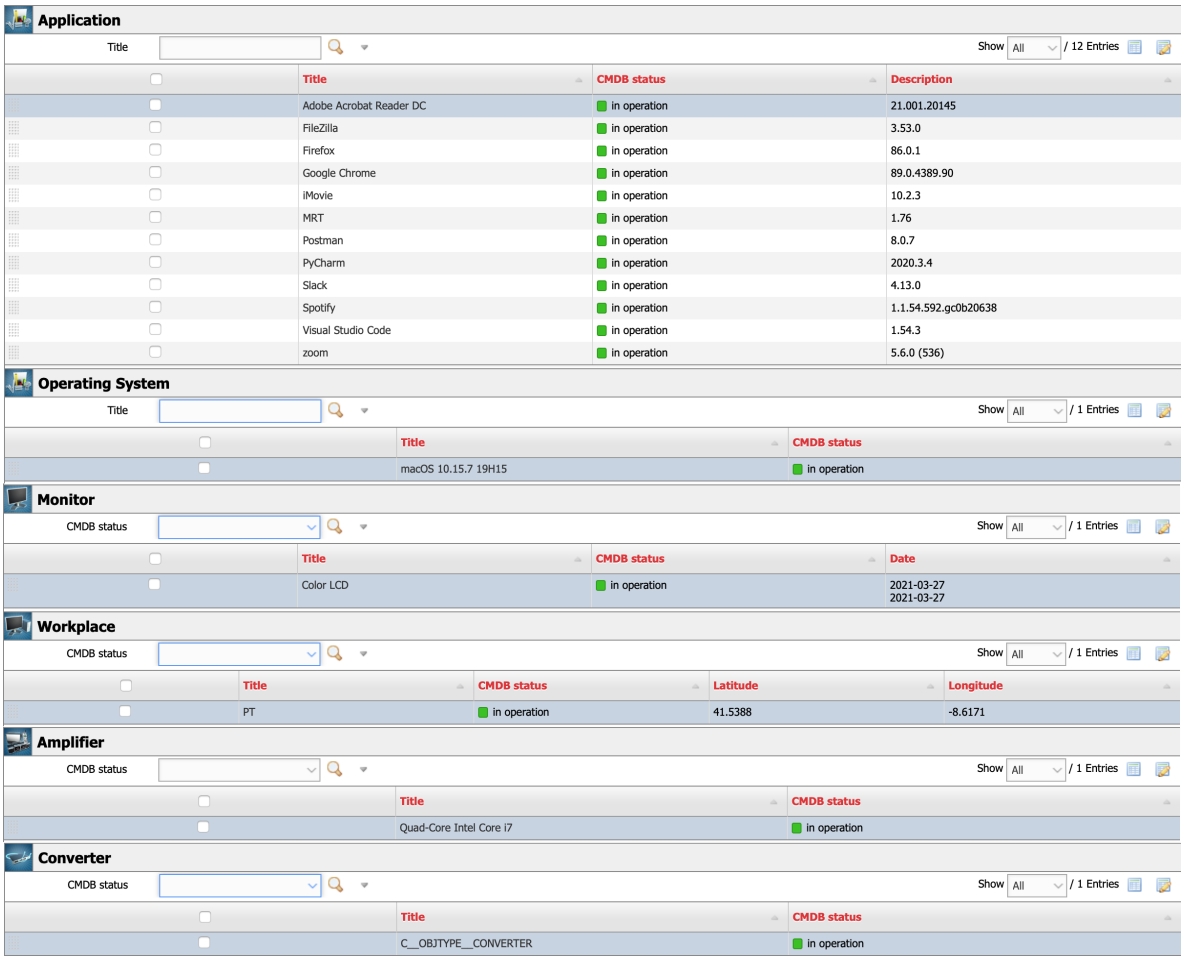


Figura 5.2: Objetos, e seus atributos, criados na CMDB no primeiro caso de teste.

Finalmente, na Figura 5.3, são apresentados os relacionamentos presentes na CMDB após o povoamento. É possível verificar a criação dos relacionamentos e a associação destes com os CIs envolvidos nos mesmos.

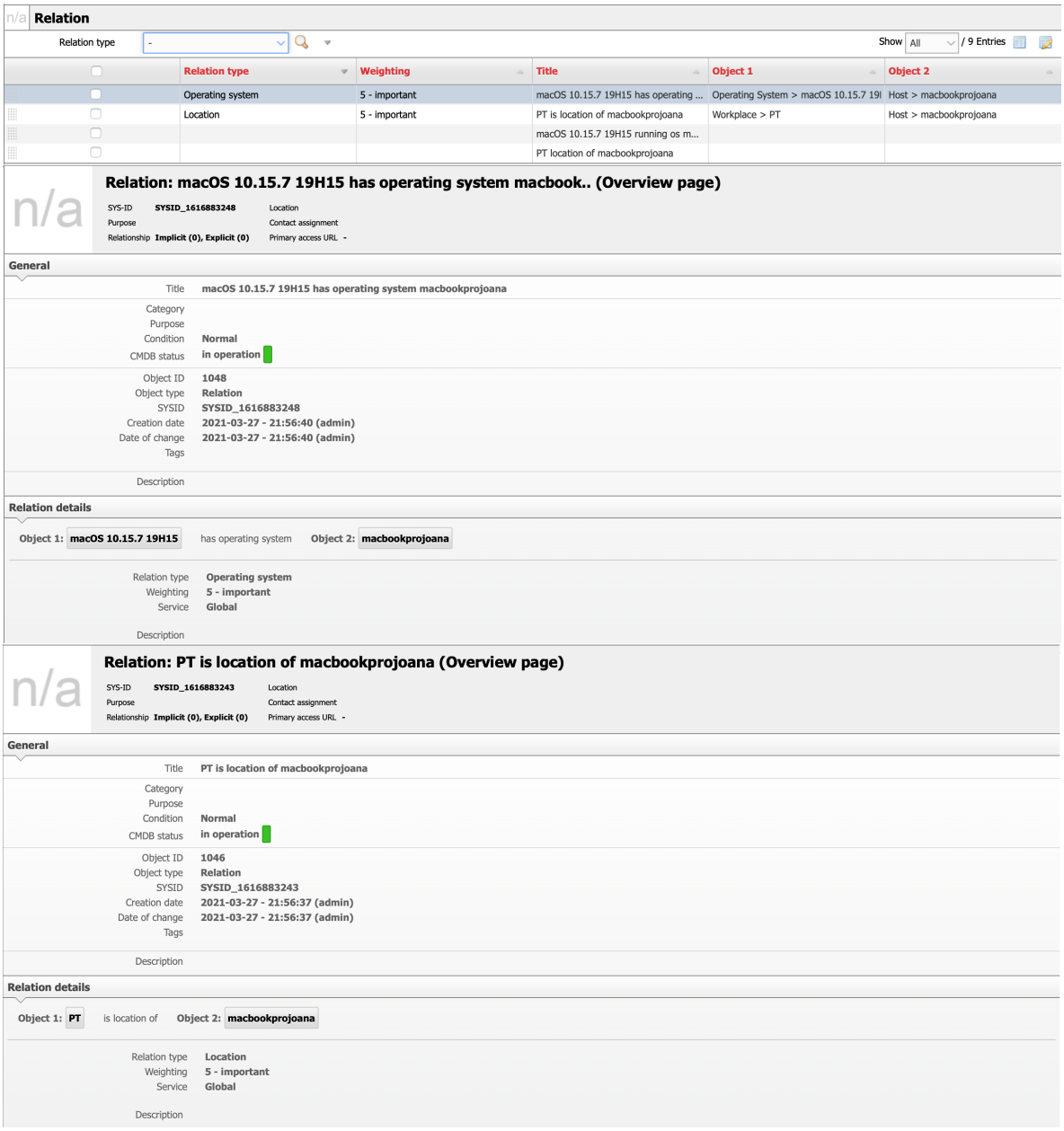


Figura 5.3: Relacionamentos criados na CMDB no primeiro caso de teste.

5.2 Caso de Teste 2

No segundo caso de teste, a ferramenta vai ser novamente executada de forma a recolher informação de apenas uma máquina física. As características da máquina a ser explorada podem ser consultadas na Tabela 5.7. A ferramenta é executada na máquina descrita na Tabela 5.1, mas a base de dados e a CMDB utilizada, que neste caso será o iTop, são executadas na máquina a ser explorada.

Tabela 5.7: Características da máquina para o segundo caso de teste.

Máquina	Dell OptiPlex 7010
Sistema Operativo	Ubuntu 18.04.5 LTS
Endereço IP	192.168.1.88
RAM	8 *GB*
CPU	2,9 *GHz* Intel Core i5-3470S
GraphDB	9.3.1
i-doit	1.14 Open
iTop	2.7.3-6624
Virtualbox	5.2.42_Ubuntur137960
Vagrant	2.2.14
SNMP	5.7.3

5.2.1 *Descoberta*

A descoberta teve por base o endereço IP referente à máquina descrita na Tabela 5.7. Na descoberta básica, foi possível obter informação da máquina através da ferramenta Nmap e do protocolo SNMP. Na descoberta detalhada foram selecionadas as categorias referentes ao sistema operativo, processamento, armazenamento e software, sendo que as credenciais da máquina foram necessárias para a obtenção deste tipo de informação.

Devido à grande extensão dos dados recolhidos, vão ser apenas apresentados resultados parciais. Tendo em conta a máquina explorada e as categorias selecionadas, é possível observar na Tabela 5.8 os CIs que seria de esperar que fossem descobertos, e aqueles que foram efetivamente criados. Aqui são apenas apresentados os nomes associados às instâncias armazenadas na base de dados, sendo que os seus atributos não estão representados. Também para o tipo de cada CI é apenas apresentado o seu nome. Na Tabela C.2 do Anexo C é possível observar alguns atributos de alguns dos CIs descobertos.

Tabela 5.8: Alguns dos CIs encontrados na fase de descoberta do segundo caso de teste.

Esperado	**Categoria**	**Encontrado**	
		Nome do tipo de CI	**Nome dos CIs**
Máquina descrita na Tabela 5.7	—	Host	ubuntu-pc
Sistema Operativo Ubuntu	Sistema Operativo	Operating System	Ubuntu 18.04.5 LTS
CPU	Processamento	CPU	IntelR CoreTM i5-3470S CPU 2.90GHz
Discos	Armazenamento	SSD	sda4
Aplicações (apenas as que estão a utilizar mais memória virtual)	Software	Application	graphdb-free gnome-shell firefox snapd evolution-data-server

Também na fase de descoberta foi possível encontrar relacionamentos entre os CIs. Na Tabela 5.9 é possível observar alguns dos relacionamentos encontrados nesta fase, sendo exibido o nome, o nome do tipo e o nome dos CIs envolvidos no relacionamento.

Tabela 5.9: Alguns dos relacionamentos encontrados na fase de descoberta do segundo caso de teste.

Nome do CI Fonte	Nome do CI Alvo	Tipo do Relacionamento	Nome do Relacionamento
ubuntu-pc	IntelR CoreTM i5-3470S CPU 2.90GHz	associated processor	ubuntu-pc associated processor IntelR CoreTM i5-3470S CPU 2.90GHz
graphdb-free	cumulus linux	has installed	ubuntu-pc has installed graphdb-free
ubuntu-pc	Dell	has vendor	has vendor Dell
ubuntu-pc	Ubuntu 18.04.5 LTS	installed os	ubuntu-pc installed os Ubuntu 18.04.5 LTS
Ubuntu 18.04.5 LTS	ubuntu-pc	running os	Ubuntu 18.04.5 LTS running os ubuntu-pc
mysql	3306	is running on port	mysql is running on port 3306

5.2.2 *Mapeamento*

Na fase de mapeamento foram analisados os modelos de dados da CMDB, que neste caso foi o iTop, e da base de dados gerada no processo de descoberta, tendo sido produzidos os potenciais mapeamentos entre estes modelos.

Considerando que o valor limite escolhido para a seleção das regras de transformação foi 0.80 (zero ponto oitenta), é possível observar na Tabela 5.10 as regras de transformação geradas. É relevante referir que, no caso dos atributos, podem existir, para um mesmo atributo, mapeamentos diferentes entre diferentes tipos de CIs visto que, diferentes tipos de CIs podem apresentar o mesmo atributo.

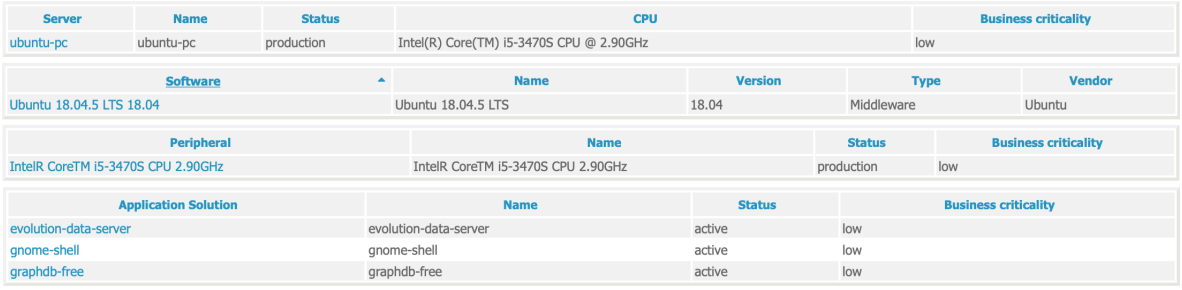
Tabela 5.10: Regras de transformação geradas no segundo caso de teste.

		CMDB		Base de Dados		Coeficiente de Similaridade
		Nome	**Descrição**	**Nome**	**Descrição**	
CIs		server	server	Host	host	1
		software	software	Operating System	operating system	0.941176
		peripheral	peripheral	CPU	central processing unit	0.888889
		applicationsolution	application solution	Application	application	0.871378
Atributos dos CIs	server	cpu	cpu	CPU	central processing unit	1
		id	id	Machine ID	machine id	0.880476
	software	id	id	ID	id	1
		vendor	vendor	vendor	vendor	1
		type	type	description	description	0.9
		name	name	title	title	0.923077
		version	version	version number	version number	0.819853
	peripheral	id	id	Vendor ID	vendor id	0.876197
	applicationsolution	status	status	title	title	0.833333

5.2.3 *Povoamento*

Tendo em conta os itens descobertos e as regras de transformação geradas, é possível verificar na Figura 5.4 a criação dos CIs que corresponderam a esses requisitos.

Na Listagem 5.2 é possível observar a informação em JSON do pedido gerado para a criação do CI correspondente ao sistema operativo Ubuntu da máquina descrita na Tabela 5.1. Este foi gerado tendo por base a informação armazenada na base de dados, e as regras de transformação presentes na Tabela 5.5.



Server	Name	Status	CPU	Business criticality
ubuntu-pc	ubuntu-pc	production	Intel(R) Core(TM) i5-3470S CPU @ 2.90GHz	low

Software	Name	Version	Type	Vendor
Ubuntu 18.04.5 LTS 18.04	Ubuntu 18.04.5 LTS	18.04	Middleware	Ubuntu

Peripheral	Name	Status	Business criticality
IntelR CoreTM i5-3470S CPU 2.90GHz	IntelR CoreTM i5-3470S CPU 2.90GHz	production	low

Application Solution	Name	Status	Business criticality
evolution-data-server	evolution-data-server	active	low
gnome-shell	gnome-shell	active	low
graphdb-free	graphdb-free	active	low

Figura 5.4: CIs criados no iTop no segundo caso de teste.

Listagem 5.2: Informação JSON da operação core/create gerado para a criação do CI correspondente ao sistema operativo da máquina descrita na Tabela 5.7.

```
{
    'operation': 'core/create',
    'comment': 'Synchronization from CMDB automatic creation...',
    'user': 'itop',
    'password': 'itop-WebServices1',
    'class': 'Software',
    'fields': {
        'vendor': 'Ubuntu',
        'version': '18.04',
        'name': 'Ubuntu 18.04.5 LTS',
        'type': 'Middleware'
    }
}
```

5.3 Caso de Teste 3

No terceiro caso de teste, a ferramenta vai ser executada de forma a recolher informação de uma infraestrutura computacional. Esta é constituída por 7 máquinas virtuais, tendo sido utilizado VirtualBox [52] e Vagrant [51] para a virtualização e administração destas. Estas máquinas são executadas na máquina física apresentada na Tabela 5.7. As características das máquinas virtuais que compõem a infraestrutura podem ser consultadas na Tabela 5.11. A ferramenta de criação automática de CMDB vai ser executada na máquina identificada pelo nome "ubuntu". A base de dados e a CMDB utilizada - i-doit 1.14 Open - são executadas na máquina descrita na Tabela 5.7.

Tabela 5.11: Características das máquinas que compõe a infraestrutura computacional para o terceiro caso de teste.

Máquina	Vagrant Box	Sistema Operativo	Endereço IP	LLDP	SNMP	SSH	Python	Nmap	Wireshark
router	CumulusCommunity/VX-3.0	Cumulus Linux 3.3.0	192.168.10.10	0.7.16-430-g9ed8a25	5.7.3	OpenSSH_6.7p1 Debian-5+deb8u3, OpenSSL 1.0.1t	—	—	—
sw1	CumulusCommunity/VX-3.0	Cumulus Linux 3.3.0	192.168.10.20	0.7.16-430-g9ed8a25	5.7.3	OpenSSH_6.7p1 Debian-5+deb8u3, OpenSSL 1.0.1t	—	—	—
sw2	CumulusCommunity/VX-3.0	Cumulus Linux 3.3.0	192.168.10.30	0.7.16-430-g9ed8a25	5.7.3	OpenSSH_6.7p1 Debian-5+deb8u3, OpenSSL 1.0.1t	—	—	—
ubuntu	generic/ubuntu1804	Ubuntu 18.04.5 LTS	192.168.10.22	0.9.9	5.7.3	OpenSSH_7.6p1 Ubuntu-4ubuntu0.3, OpenSSL 1.0.2n	3.9.1	7.60	2.6.10
windows	gusztavvargadr/windows-10	Microsoft Windows 10 Enterprise Evaluation 10.0.19042	192.168.10.21	Não instalado	Não instalado	—	—	—	—
dbs	ubuntu/xenial64	Ubuntu 16.04.7 LTS	192.168.10.31	Não instalado	5.7.3	OpenSSH_7.2p2 Ubuntu-4ubuntu2.10, OpenSSL 1.0.2g	—	—	—
wbs	ubuntu/xenial64	Ubuntu 16.04.7 LTS	192.168.10.32	Não instalado	5.7.3	OpenSSH_7.2p2 Ubuntu-4ubuntu2.10 OpenSSL 1.0.2g	—	—	—

5.3.1 *Descoberta*

A descoberta teve por base a informação exportada da ferramenta Angry IP Scanner, que contém informação acerca de três das máquinas que compõem a infraestrutura. Os dados exportados podem ser consultados na Tabela 5.12.

Tabela 5.12: Ficheiro CSV exportado da ferramenta Angry IP Scanner para o terceiro caso de teste.

IP	Ping	MAC Address	Hostname	Ports	NetBIOS Info	MAC Vendor
192.168.10.10	0 ms	08:00:27:12:F6:B5	[n/a]	[n/a]	[n/a]	PCS Systemtechnik
192.168.10.20	0 ms	08:00:27:C7:45:8C	[n/a]	[n/a]	[n/a]	PCS Systemtechnik
192.168.10.30	0 ms	08:00:27:9C:71:2B	[n/a]	[n/a]	[n/a]	PCS Systemtechnik

Na descoberta básica, foi possível recolher informação através dos protocolos LLDP e SNMP e da ferramenta Nmap, devido ao facto de estes protocolos estarem configurados nas máquinas que compõem a infraestrutura, e a ferramenta estar instalada na máquina a executar a descoberta. Desta forma, foi possível aceder às tabelas de roteamento e tabelas ARP dos dispositivos, o que permitiu, tendo apenas conhecimento de três das máquinas, descobrir as restantes. Na descoberta detalhada foram selecionadas as categorias referentes ao sistema operativo, processamento, armazenamento, software e serviços, sendo que as credenciais das máquinas foram necessárias para a obtenção deste tipo de informação.

Neste caso em específico, é possível verificar que apesar de pelo menos duas máquinas apresentarem o mesmo sistema operativo na mesma versão - Cumulus Linux 3.3.0, por exemplo - foi apenas criada uma instância deste sistema operativo na base de dados, provando as capacidades do mecanismo de reconciliação.

É também relevante referir que foram criados dois CIs referentes ao CPU da máquina descrita na Tabela 5.7. Isto aconteceu porque as configurações do CPU nas máquinas virtuais são diferentes.

Devido à grande extensão dos dados recolhidos, vão ser apenas apresentados resultados parciais, sendo possível observar na Tabela 5.13 os componentes que seria de esperar que fossem descobertos, e aqueles que foram efetivamente criados. Aqui são apenas apresentados os nomes associados às instâncias armazenadas na base de dados, sendo que

os restantes atributos não estão representados. Também para o tipo de cada CI é apenas apresentado o seu nome associado.

Tabela 5.13: Alguns dos CIs encontrados na fase de descoberta do terceiro caso de teste.

Esperado		**Categoria**	**Encontrado**	
			Nome do tipo de CI	**Nome dos CIs**
Máquinas virtuais descritas na Tabela 5.11	router	—	virtual host	router
	sw1			switch-of
	sw2			switch-dc
	ubuntu			ubuntu-pc
	windows			windows-pc
	dbs			db-server
	wbs			web-server
Sistemas Operativos	Cumulus Linux	Sistema Operativo	Operating System	Cumulus Linux
	Ubuntu 16.04			Ubuntu 16.04.7 LTS
	Ubuntu 18.04			Ubuntu 18.04.5 LTS
	Microsoft Windows			Microsoft Windows 10 Enterprise Evaluation
CPU		Processamento	CPU	IntelR CoreTM i5-3470S CPU 2.90GHz
				IntelR CoreTM i5-3470S CPU 2.90GHz
Discos		Armazenamento	SSD	sda3
				sdb
				sda1
				sda4
Aplicações		Software	Application	lldpd
				auditd
				python3
				irqbalance
				lxcfs
				systemd
				unattended-upgrades
				nginx
				snmpd
				python
				switchd
				portwd
Serviços		Serviços	Service	AudioEndpointBuilder
				Audiosrv
				BFE
				BrokerInfrastructure
				CDPSvc
				CertPropSvc
				CoreMessagingRegistrar 496
				CryptSvc
				DcomLaunch
				Dhcp

Na figura 5.5 é possível observar as máquinas e os sistemas operativos armazenados na base de dados GraphDB, sendo possível observar a relação entre estes.

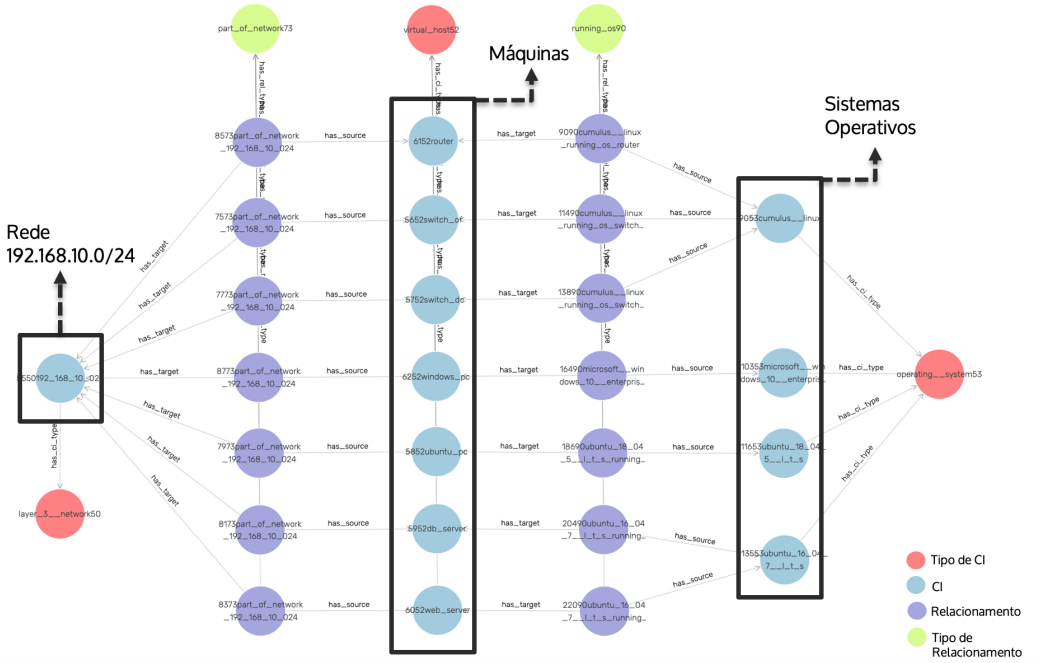


Figura 5.5: Visualização das máquinas descobertas na rede explorada na base de dados GraphDB no terceiro caso de teste.

Na Tabela 5.14 é possível observar os atributos do CI que representa a máquina identificada pelo nome "ubuntu" descrita na Tabela 5.11.

Também na fase de descoberta foi possível encontrar relacionamentos entre os CIs. Na Tabela 5.15 é possível observar alguns dos relacionamentos encontrados nesta fase, sendo exibido o nome, o nome do tipo e o nome dos CIs envolvidos no relacionamento.

5.3.2 *Mapeamento*

Na fase de mapeamento foram analisados os modelos de dados da CMDB - i-doit - e da base de dados gerada no processo de descoberta, sendo produzidos os potenciais mapeamentos entre estes modelos.

Considerando que o valor limite escolhido para a seleção das regras de transformação foi 0.86 (zero ponto oitenta e seis), é possível observar na Tabela 5.16 as regras de transfromação geradas.

Tabela 5.14: Informação recolhida sobre uma das máquinas virtuais explorada na fase de descoberta no terceiro caso de teste.

Atributo	**Valor**
title	ubuntu-pc
hostname	ubuntu-pc
has_ipv4	10.0.2.15
has_ipv4	192.168.10.22
mac_address	08:00:27:73:B2:EB
os_family	Linux
status	up
os family	Ubuntu
os name	Ubuntu 18.04.5 LTS
os version	18.04
CPU cores	2
CPU	Intel(R) Core(TM) i5-3470S CPU @ 2.90GHz
CPU frequency unit	MHz
CPU frequency	2893.420
CPU speed	2893.420 MHz
Icon name	computer-vm
Chassis	vm
Machine ID	2ee2cdc15b964490aa29b72b9808a579
Boot ID	657798635c9e40978de53e21a44a053d
Virtualization	oracle
Operating System	Ubuntu 18.04.5 LTS
Kernel	Linux 4.15.0-130-generic
Architecture	x86-64

Tabela 5.15: Alguns dos relacionamentos encontrados na fase de descoberta do terceiro caso de teste.

Nome do CI Fonte	**Nome do CI Alvo**	**Tipo do Relacionamento**	**Nome do Relacionamento**
ubuntu-pc	IntelR CoreTM i5-3470S CPU 2.90GHz	associated processor	ubuntu-pc associated processor IntelR CoreTM i5-3470S CPU 2.90GHz
ubuntu-pc	python3	has installed	ubuntu-pc has installed python3
nginx	web-server	has installed	nginx installed on web-server
sda1	web-server	has storage	sda1 is storage of web-server
BFE	windows-pc	running on	BFE running on windows-pc
Cumulus Linux	router	running os	Cumulus Linux running os router
db-server	Ubuntu 16.04.7 LTS	installed os	db-server installed os Ubuntu 16.04.7 LTS

Tabela 5.16: Regras de transformação geradas no terceiro caso de teste.

		CMDB		Base de dados		Coeficiente de similaridade
		Nome	**Descrição**	**Nome**	**Descrição**	
CIs		C_OBJTYPE_VIRTUAL_HOST	Virtual Host	virtual host	virtual host	1
		C_OBJTYPE_APPLICATION	Application	Application	application	1
		C_OBJTYPE_OPERATING_SYSTEM	Operating System	Operating System	operating system	1
		C_OBJTYPE_IT_SERVICE	Service	Service	service	1
		C_OBJTYPE_LAYER3_NET	Layer 3-net	Layer 3 Network	layer 3 network	1
		C_OBJTYPE_AMPLIFIER	Amplifier	CPU	central processing unit	0.888889
Atributos dos CIs	C_OBJTYPE_VIRTUAL_HOST	title	Title	title	title	1
		hostname	Hostname	hostname	hostname	1
		frequency	CPU frequency	CPU frequency	central processing unit frequency	1
		cores	CPU cores	CPU cores	central processing unit cores	1
		status	Condition	status	status	1
		frequency_unit	CPU frequency unit	CPU frequency unit	central processing unit frequency unit	1
		serial	Serial number	serial_number	serial number	1
		speed_unit	Speed unit	CPU speed	central processing unit speed	0.877123
		mac	MAC	mac_address	media access control address	0.866025
		ipv4_address	IPv4 address	has_ipv4	internet protocol version 4	0.866025
	C_OBJTYPE_APPLICATION	title	Title	title	title	1
		description	Description	description	description	1
	C_OBJTYPE_OPERATING_SYSTEM	assigned_version	Version number	version number	version number	1
		title	Title	title	title	1
		description	Description	description	description	1
		manufacturer	Manufacturer	manufacturer	manufacturer	1
		id	ID	ID	id	1
	C_OBJTYPE_IT_SERVICE	title	Title	title	title	1
		description	Description	description	description	1
		status	Condition	status	status	1
		service_alias	Aliases	Name	name	0.923077
	C_OBJTYPE_LAYER3_NET	title	Title	title	title	1
	C_OBJTYPE_AMPLIFIER	title	Title	title	title	1
		description	Description	description	description	1
Relacionamentos		C_RELATION_TYPE_NETWORK_PORT	Ports	port from	port	1
		C_RELATION_TYPE_OPERATION_SYSTEM	Operating system	running os	running operating system	0.942343
		C_RELATION_TYPE_ORGANIZATION	Organization	running on	running	0.9
Atributos dos Relacionamentos	C_RELATION_TYPE_NETWORK_PORT	title	Title	title	title	1
	C_RELATION_TYPE_OPERATION_SYSTEM					
	C_RELATION_TYPE_ORGANIZATION					

5.3.3 *Povoamento*

Tendo em conta os itens de configuração criados e as regras de transformação geradas, foi possível comprovar a criação dos objetos que corresponderam a esses requisitos. É possível observar na Tabela 5.17, as instâncias criadas na CMDB e os seus tipos associados.

Tabela 5.17: Elementos criados na CMDB no terceiro caso de teste.

	Tipo na CMDB	Tipo na Base de Dados	Nome dos objetos
CIs	C_OBJTYPE_VIRTUAL_HOST	virtual host	router switch-of switch-dc windows-pc ubuntu-pc db-server web-server
	C_OBJTYPE_APPLICATION	Application	auditd irqbalance lldpd lxcfs nginx portwd python python3 snmpd switchd systemd unattended-upgrades
	C_OBJTYPE_OPERATING_SYSTEM	Operating System	Cumulus Linux Microsoft Windows 10 Enterprise Evaluation Ubuntu 16.04.7 LTS Ubuntu 18.04.5 LTS
	C_OBJTYPE_IT_SERVICE	Service	AudioEndpointBuilder Audiosrv BFE BrokerInfrastructure CDPSvc CertPropSvc CoreMessagingRegistrar 496 CryptSvc DcomLaunch Dhcp
	C_OBJTYPE_LAYER3_NET	Layer 3 Network	10.0.2.024 192.168.10.024
	C_OBJTYPE_AMPLIFIER	CPU	IntelR CoreTM i5-3470S CPU 2.90GHz IntelR CoreTM i5-3470S CPU 2.90GHz
Relacionamentos	C_RELATION_TYPE_OPERATION_SYSTEM	running os	Cumulus Linux has operating system router Cumulus Linux has operating system switch-dc Cumulus Linux has operating system switch-of Microsoft Windows 10 Enterprise Evaluation has operating system windows-pc Ubuntu 16.04.7 LTS has operating system db-server Ubuntu 16.04.7 LTS has operating system web-server Ubuntu 18.04.5 LTS has operating system ubuntu-pc
	C_RELATION_TYPE_ORGANIZATION	running on	AudioEndpointBuilder has member windows-pc Audiosrv has member windows-pc BFE has member windows-pc BrokerInfrastructure has member windows-pc CDPSvc has member windows-pc CertPropSvc has member windows-pc CoreMessagingRegistrar 496 has member windows-pc CryptSvc has member windows-pc DcomLaunch has member windows-pcDhcp has member windows-pc

Na Listagem 5.3 é possível observar a informação em JSON do pedido gerado para a criação do CI correspondente à máquina identificada pelo nome "ubuntu" descrita na Tabela 5.11. Este foi gerado tendo por base a informação acerca do CI presente na base

de dados e as regras de transformação presentes na Tabela 5.16. Na Figura 5.6 é possível verificar a criação deste CI na CMDB.

Listagem 5.3: Informação JSON do pedido cmdb.object.create gerado para a criação do CI correspondente à máquina virtual da infraestrutura no terceiro caso de teste.

```
{
    'version': '2.0',
    'method': 'cmdb.object.create',
    'params': {
        'apikey': 'cmdb-auto',
        'language': 'en',
        'type': 'C__OBJTYPE__VIRTUAL_HOST',
        'title': 'C__OBJTYPE__VIRTUAL_HOST',
        'categories': {
            'C__CATG__IP': [{
                'hostname': 'ubuntu-pc',
                'ipv4_address': '192.168.10.22'
            }],
            'C__CATG__CPU': [{
                'cores': 2,
                'frequency_unit': 2,
                'frequency': 2893.42
            }],
            'C__CATG__CONTROLLER_FC_PORT': [{
                'speed_unit': 3
            }],
            'C__CATG__GLOBAL': [{
                'title': 'ubuntu-pc',
                'status': 7
            }],
            'C__CATG__NETWORK_LOG_PORT': [{
                'mac': '08:00:27:73:B2:EB'
            }]
        }
    },
    'id': 1
}
```

Observando a informação JSON gerada, é possível perceber que o mecanismo faz efetivamente a correspondência entre os valores descobertos e os valores permitidos na CMDB, no caso dos atributos que possuem valores pré-definidos. É disso exemplo o atributo *"frequency_unit"*. Este foi mapeado com o atributo *"CPU frequency unit"* da base de dados e como é possível verificar na Tabela 5.14, na base de dados o valor descoberto foi *"MHz"*. No entanto, o atributo na CMDB armazena um valor inteiro que corresponde a uma unidade. Na Listagem 5.4 é possível observar os valores possíveis para o atributo *"frequency_unit"* e as unidades a que correspondem, sendo possível verificar que a correspondência gerada é

a mais correta - o valor *"MHz"* da base de dados foi mapeado com o valor *"2"* na CMDB, que efetivamente corresponde à unidade *"MHz"*.

Listagem 5.4: Resultado da execução do método cmdb.dialog.read para o atributo *"frequency_unit"* do i-doit.


```
"result": [
    {
        "id": "1",
        "const": "C__FREQUENCY_UNIT__KHZ",
        "title": "KHz"
    },
    {
        "id": "2",
        "const": "C__FREQUENCY_UNIT__MHZ",
        "title": "MHz"
    },
    {
        "id": "3",
        "const": "C__FREQUENCY_UNIT__GHZ",
        "title": "GHz"
    },
    {
        "id": "4",
        "const": "C__FREQUENCY_UNIT__THZ",
        "title": "THz"
    }
]
```

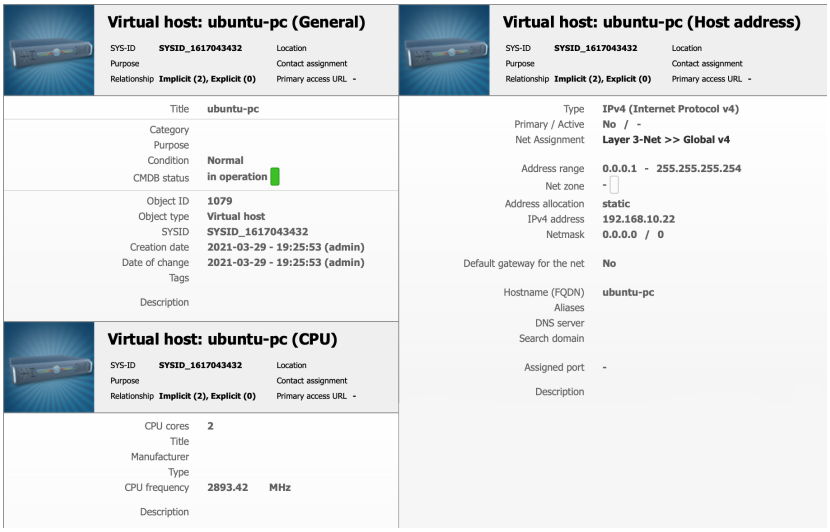
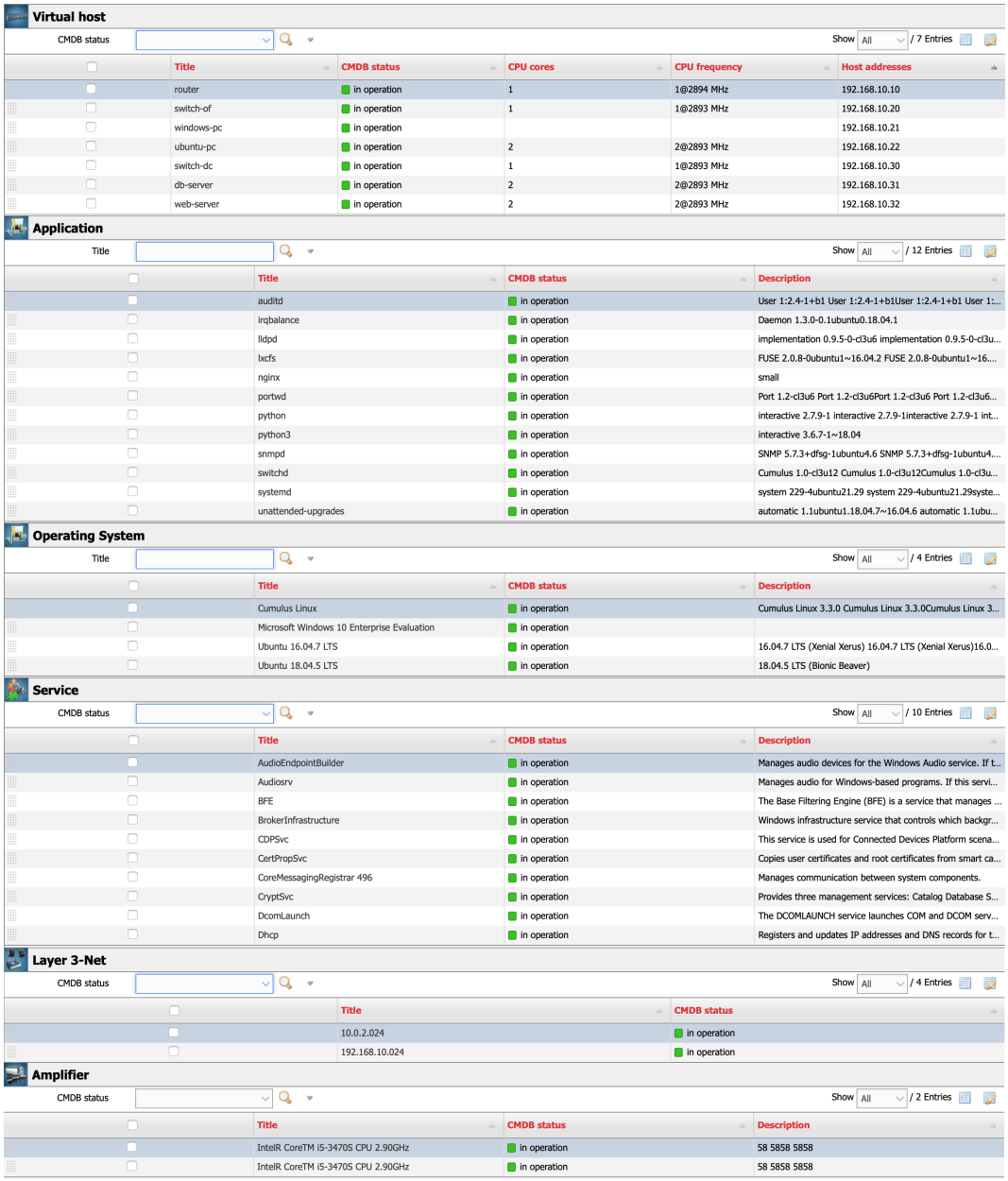



Figura 5.6: Objeto, e atributos correspondentes, referente à máquina virtual descoberta, criado na CMDB no terceiro caso de teste.

Na Figura 5.7 é também possível observar os restantes CIs, e seus atributos, criados na CMDB.



Virtual host

CMDB status — Show All / 7 Entries

Title	CMDB status	CPU cores	CPU frequency	Host addresses
router	in operation	1	1@2894 MHz	192.168.10.10
switch-of	in operation	1	1@2893 MHz	192.168.10.20
windows-pc	in operation			192.168.10.21
ubuntu-pc	in operation	2	2@2893 MHz	192.168.10.22
switch-dc	in operation	1	1@2893 MHz	192.168.10.30
db-server	in operation	2	2@2893 MHz	192.168.10.31
web-server	in operation	2	2@2893 MHz	192.168.10.32

Application

Title — Show All / 12 Entries

Title	CMDB status	Description
auditd	in operation	User 1:2.4-1+b1 User 1:2.4-1+b1User 1:2.4-1+b1 User 1:...
irqbalance	in operation	Daemon 1.3.0-0.1ubuntu0.18.04.1
lldpd	in operation	implementation 0.9.5-0-cl3u6 implementation 0.9.5-0-cl3u...
lxcfs	in operation	FUSE 2.0.8-0ubuntu1~16.04.2 FUSE 2.0.8-0ubuntu1~16....
nginx	in operation	small
portwd	in operation	Port 1.2-cl3u6 Port 1.2-cl3u6Port 1.2-cl3u6 Port 1.2-cl3u6...
python	in operation	interactive 2.7.9-1 interactive 2.7.9-1interactive 2.7.9-1 int...
python3	in operation	interactive 3.6.7-1~18.04
snmpd	in operation	SNMP 5.7.3+dfsg-1ubuntu4.6 SNMP 5.7.3+dfsg-1ubuntu4....
switchd	in operation	Cumulus 1.0-cl3u12 Cumulus 1.0-cl3u12Cumulus 1.0-cl3u...
systemd	in operation	system 229-4ubuntu21.29 system 229-4ubuntu21.29syste...
unattended-upgrades	in operation	automatic 1.1ubuntu1.18.04.7~16.04.6 automatic 1.1ubu...

Operating System

Title — Show All / 4 Entries

Title	CMDB status	Description
Cumulus Linux	in operation	Cumulus Linux 3.3.0 Cumulus Linux 3.3.0Cumulus Linux 3...
Microsoft Windows 10 Enterprise Evaluation	in operation	
Ubuntu 16.04.7 LTS	in operation	16.04.7 LTS (Xenial Xerus) 16.04.7 LTS (Xenial Xerus)16.0...
Ubuntu 18.04.5 LTS	in operation	18.04.5 LTS (Bionic Beaver)

Service

CMDB status — Show All / 10 Entries

Title	CMDB status	Description
AudioEndpointBuilder	in operation	Manages audio devices for the Windows Audio service. If t...
Audiosrv	in operation	Manages audio for Windows-based programs. If this servi...
BFE	in operation	The Base Filtering Engine (BFE) is a service that manages ...
BrokerInfrastructure	in operation	Windows infrastructure service that controls which backgr...
CDPSvc	in operation	This service is used for Connected Devices Platform scena...
CertPropSvc	in operation	Copies user certificates and root certificates from smart ca...
CoreMessagingRegistrar 496	in operation	Manages communication between system components.
CryptSvc	in operation	Provides three management services: Catalog Database S...
DcomLaunch	in operation	The DCOMLAUNCH service launches COM and DCOM serv...
Dhcp	in operation	Registers and updates IP addresses and DNS records for t...

Layer 3-Net

CMDB status — Show All / 4 Entries

Title	CMDB status
10.0.2.024	in operation
192.168.10.024	in operation

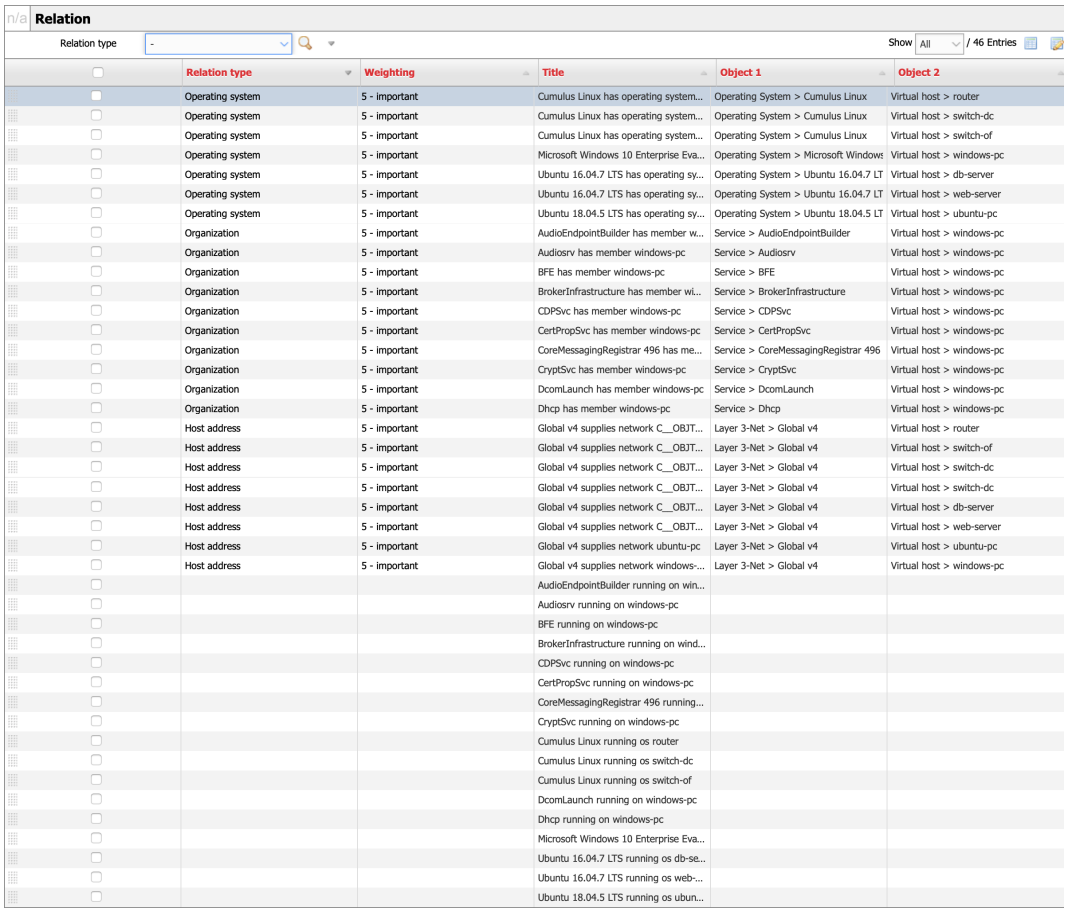
Amplifier

CMDB status — Show All / 2 Entries

Title	CMDB status	Description
IntelR CoreTM i5-3470S CPU 2.90GHz	in operation	58 5858 5858
IntelR CoreTM i5-3470S CPU 2.90GHz	in operation	58 5858 5858

Figura 5.7: Objetos, e seus atributos, criados na CMDB no terceiro caso de teste.

Finalmente, na Figura 5.8, são apresentados os relacionamentos presentes na CMDB após o povoamento. Os relacionamentos do tipo "Host address" foram gerados automaticamente no momento da criação das máquinas virtuais, porque estas foram associadas a um endereço IPv4.



n/a **Relation**

Relation type: -

Show All / 46 Entries

	Relation type	Weighting	Title	Object 1	Object 2
	Operating system	5 - important	Cumulus Linux has operating system...	Operating System > Cumulus Linux	Virtual host > router
	Operating system	5 - important	Cumulus Linux has operating system...	Operating System > Cumulus Linux	Virtual host > switch-dc
	Operating system	5 - important	Cumulus Linux has operating system...	Operating System > Cumulus Linux	Virtual host > switch-of
	Operating system	5 - important	Microsoft Windows 10 Enterprise Eva...	Operating System > Microsoft Windows	Virtual host > windows-pc
	Operating system	5 - important	Ubuntu 16.04.7 LTS has operating sy...	Operating System > Ubuntu 16.04.7 LT	Virtual host > db-server
	Operating system	5 - important	Ubuntu 16.04.7 LTS has operating sy...	Operating System > Ubuntu 16.04.7 LT	Virtual host > web-server
	Operating system	5 - important	Ubuntu 18.04.5 LTS has operating sy...	Operating System > Ubuntu 18.04.5 LT	Virtual host > ubuntu-pc
	Organization	5 - important	AudioEndpointBuilder has member w...	Service > AudioEndpointBuilder	Virtual host > windows-pc
	Organization	5 - important	Audiosrv has member windows-pc	Service > Audiosrv	Virtual host > windows-pc
	Organization	5 - important	BFE has member windows-pc	Service > BFE	Virtual host > windows-pc
	Organization	5 - important	BrokerInfrastructure has member wi...	Service > BrokerInfrastructure	Virtual host > windows-pc
	Organization	5 - important	CDPSvc has member windows-pc	Service > CDPSvc	Virtual host > windows-pc
	Organization	5 - important	CertPropSvc has member windows-pc	Service > CertPropSvc	Virtual host > windows-pc
	Organization	5 - important	CoreMessagingRegistrar 496 has me...	Service > CoreMessagingRegistrar 496	Virtual host > windows-pc
	Organization	5 - important	CryptSvc has member windows-pc	Service > CryptSvc	Virtual host > windows-pc
	Organization	5 - important	DcomLaunch has member windows-pc	Service > DcomLaunch	Virtual host > windows-pc
	Organization	5 - important	Dhcp has member windows-pc	Service > Dhcp	Virtual host > windows-pc
	Host address	5 - important	Global v4 supplies network C__OBJT...	Layer 3-Net > Global v4	Virtual host > router
	Host address	5 - important	Global v4 supplies network C__OBJT...	Layer 3-Net > Global v4	Virtual host > switch-of
	Host address	5 - important	Global v4 supplies network C__OBJT...	Layer 3-Net > Global v4	Virtual host > switch-dc
	Host address	5 - important	Global v4 supplies network C__OBJT...	Layer 3-Net > Global v4	Virtual host > switch-dc
	Host address	5 - important	Global v4 supplies network C__OBJT...	Layer 3-Net > Global v4	Virtual host > db-server
	Host address	5 - important	Global v4 supplies network C__OBJT...	Layer 3-Net > Global v4	Virtual host > web-server
	Host address	5 - important	Global v4 supplies network ubuntu-pc	Layer 3-Net > Global v4	Virtual host > ubuntu-pc
	Host address	5 - important	Global v4 supplies network windows-...	Layer 3-Net > Global v4	Virtual host > windows-pc
			AudioEndpointBuilder running on win...		
			Audiosrv running on windows-pc		
			BFE running on windows-pc		
			BrokerInfrastructure running on wind...		
			CDPSvc running on windows-pc		
			CertPropSvc running on windows-pc		
			CoreMessagingRegistrar 496 running...		
			CryptSvc running on windows-pc		
			Cumulus Linux running os router		
			Cumulus Linux running os switch-dc		
			Cumulus Linux running os switch-of		
			DcomLaunch running on windows-pc		
			Dhcp running on windows-pc		
			Microsoft Windows 10 Enterprise Eva...		
			Ubuntu 16.04.7 LTS running os db-se...		
			Ubuntu 16.04.7 LTS running os web-...		
			Ubuntu 18.04.5 LTS running os ubun...		

Figura 5.8: Relacionamentos criados na CMDB no terceiro caso de teste.

5.4 Caso de Teste 4

No quarto caso de teste foi avaliado o mecanismo de mapeamento implementado. Este, descrito na Secção 4.2.10, evita a repetição entre conceitos mapeados, ou seja, um elemento de um modelo, mapeia apenas num elemento do outro modelo cujo valor do coeficiente de similaridade é o máximo calculado. Desta forma, não há dois elementos de um modelo a mapear no mesmo elemento do outro modelo.

Neste caso de teste, foi executado o mapeamento entre modelos que permite a repetição entre elementos mapeados selecionando apenas o maior valor do coeficiente de similaridade calculado, de forma a ser possível comparar com o mecanismo implementado. Esta comparação foi efetuada quer para o mapeamento da base de dados com o i-doit, quer com o iTop. Os testes efetuados tiveram em consideração a base de dados obtida da combinação dos resultados das descobertas dos casos de teste 1, 2 e 3.

O primeiro caso apresentado é referente ao mapeamento com o i-doit. A Tabela 5.18 apresentada o número de mapeamentos, corretos e errados, obtidos ao executar os dois mecanismos de mapeamento para diferentes valores de limite. É possível observar que à medida que o valor limite aumenta, aumenta também a percentagem de mapeamentos corretos. Na Figura 5.9 é possível observar que para valores limite superiores, a percentagem de mapeamentos corretos aumenta. Isto acontece porque ao limitar a repetição, o

Tabela 5.18: Número de mapeamentos, corretos e errados, selecionados para o i-doit considerando diferentes valores limite.

		Com repetição					Sem repetição				
Valor limite		**0.5**	**0.7**	**0.8**	**0.9**	**1**	**0.5**	**0.7**	**0.8**	**0.9**	**1**
CIs	**Corretos**	6	6	6	6	5	6	6	6	6	5
	Errados	23	13	6	2	1	22	12	6	2	1
	Percentagem	20,68965517	31,57894737	50	75	83,3333333	21,4285714	33,3333333	50	75	83,3333333
Relacionamentos	**Corretos**	8	7	5	4	2	3	3	3	3	2
	Errados	27	16	5	1	0	26	9	5	1	0
	Percentagem	22,85714286	30,43478261	50	80	100	10,3448276	25	37,5	75	100
Total	**Corretos**	14	13	11	10	7	9	9	9	9	7
	Errados	50	29	11	3	1	48	21	11	3	1
	Percentagem	21,875	30,95238095	50	76,9230769	87,5	15,7894737	30	45	75	87,5

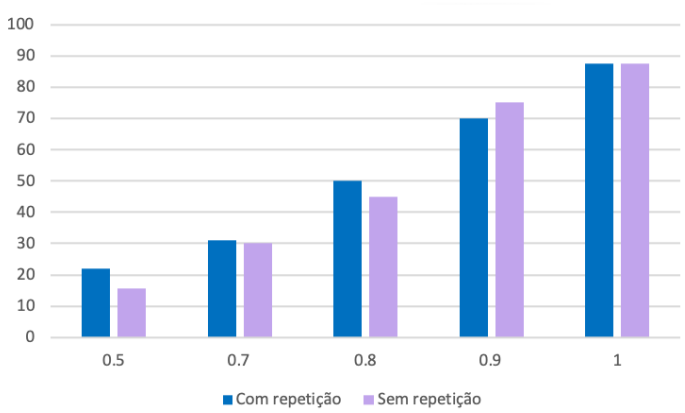


Figura 5.9: Comparação de mapeamentos corretos entre os dois mecanismos de mapeamento para diferentes valores limite no caso do i-doit.

valor geral do coeficiente de similaridade selecionado vai diminuir, porque não vai ser o máximo calculado entre todos os mapeamentos nos casos em que o elemento já tenha sido mapeado.

O segundo caso apresentado é referente ao mapeamento com o iTop. A Tabela 5.19 apresentada o número de mapeamentos, corretos e errados, obtidos ao executar os dois mecanismos de mapeamento para diferentes valores limite. É possível observar que à medida que o valor limite aumenta, aumenta também a percentagem de mapeamentos corretos. Na Figura 5.10 é possível observar que, no caso do iTop, a percentagem de mapeamentos corretos para o mecanismo que evita repetições, é sempre superior. Isto acontece porque ao limitar a repetição, o valor dos coeficientes de similaridade selecionados vai diminuir, porque não vão corresponder ao valor máximo calculado nos mapeamentos em que os elementos tenham sido já mapeados.

Tabela 5.19: Número de mapeamentos, corretos e errados, selecionados para o iTop considerando diferentes valores limite..

		Com repetição					Sem repetição				
Valor limite		**0.5**	**0.7**	**0.8**	**0.9**	**1**	**0.5**	**0.7**	**0.8**	**0.9**	**1**
CIs	**Corretos**	5	4	3	3	3	6	5	4	3	3
	Errados	21	14	8	2	0	20	13	6	2	0
	Percentagem	19,23076923	22,22222222	27,27272727	60	100	23,07692308	27,7777778	40	60	100
Relacionamentos	**Corretos**	3	1	0	0	0	3	1	0	0	0
	Errados	22	5	0	0	0	12	5	0	0	0
	Percentagem	12	16,66666667	0	0	0	20	16,6666667	0	0	0
Total	**Corretos**	8	5	3	3	3	9	6	4	3	3
	Errados	43	19	8	2	0	32	18	6	2	0
	Percentagem	15,68627451	20,83333333	27,27272727	60	100	21,95121951	25	40	60	100

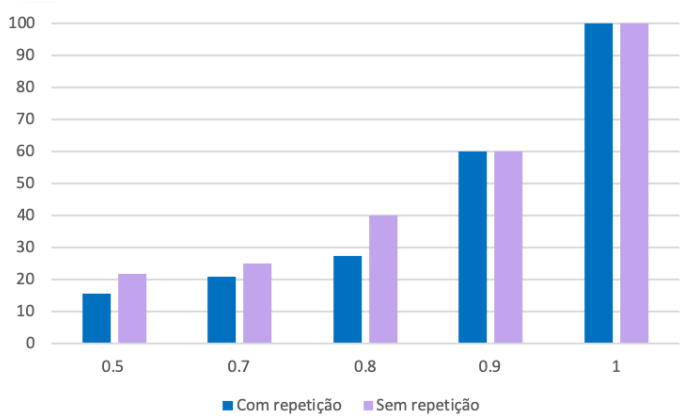


Figura 5.10: Comparação de mapeamentos corretos entre os dois mecanismos de mapeamento para diferentes valores limite no caso do iTop.

É então possível perceber, que, tendo em conta os valores de limite selecionados nos três casos de teste apresentados anteriormente, o mecanismo que evita repetições permite a seleção de um maior número de mapeamentos corretos.

5.5 Sumário

Depois de executados os testes, é possível responder às questões levantadas no início deste Capítulo.

O processo de descoberta é capaz de recolher a informação requerida pelo utilizador?
Observando os resultados dos testes, é possível afirmar que a ferramenta é capaz de recolher informação de acordo com o solicitado, sendo que os dados que se esperam recolher são efetivamente descobertos. No entanto, é possível melhorar os mecanismos de descoberta implementados, de forma a que estes recolham informação mais detalhada. Para além disto, tendo em conta que a ferramenta permite que sejam integrados quaisquer outros mecanismos de descoberta, é também possível desenvolver outros mecanismos e utilizá-los de forma a descobrir outros tipos de informação. Adicionalmente, seria também

relevante inferir mais informação tendo em conta os dados descobertos.

Os resultados do mapeamento retratam a correspondência entre os elementos mais semelhantes dos dois modelos?

O mapeamento entre os modelos está sempre dependente do valor limite escolhido, ou seja, se o valor escolhido, que pode variar entre zero e um, for um, é verificado que os mapeamentos estão efetivamente corretos. Nos testes realizados foram escolhidos valores de limite que variaram entre 0.8 (zero ponto oitenta) e 0.86 (zero ponto oitenta e seis), e é possível constatar que a maior parte dos mapeamentos representam efetivamente a correspondência entre os elementos mais semelhantes dos dois modelos. No entanto, é possível observar também alguns mapeamentos menos corretos. É disso exemplo, no segundo caso de teste, o mapeamento entre o tipo de CI que representa um periférico com um tipo que representa um CPU. Desta forma, é possível concluir que o mecanismo de mapeamento pode ser melhorado, de forma a fornecer regras de transformação mais exatas. Isto pode ser conseguido com a utilização de, por exemplo, uma comparação entre termos mais específica e baseada apenas no domínio da linguagem a ser utilizado. Outro aspeto que pode melhorar o mapeamento entre os modelos, passa pelo aperfeiçoamento da terminologia utilizada no momento da descoberta. Para além disto, este mecanismo considerou apenas o mapeamento único de elementos, isto é, foi apenas permitido que um elemento de um modelo fosse mapeado com apenas um elemento do outro modelo. Desta forma, não foi tido em consideração que dois ou mais elementos de um modelo podem mapear num mesmo elemento do outro modelo. Estes casos podem acontecer quando existem modelos de dados que possuem elementos mais abrangentes, e é sem dúvida um caso que pode ajudar na geração de regras de transformação mais corretas. Apesar de o quarto caso de teste apresentar melhores resultados para o mecanismo que evita a repetição, é também possível observar que o mecanismo que permite a repetição, apesar de apresentar mais mapeamentos errados, também apresenta mais mapeamentos corretos.

A informação é enviada para a CMDB?

Foi possível constatar que os CIs e relacionamentos descobertos e que se enquadram nas regras de transformação geradas são efetivamente criados na CMDB.

6

CONCLUSÕES E TRABALHO FUTURO

Neste último Capítulo é feita uma apreciação global do trabalho desenvolvido ao longo desta dissertação, assim como uma abordagem do trabalho futuro.

6.1 CONCLUSÃO

Nesta dissertação foi apresentada uma ferramenta de criação automática de uma CMDB, que executa a descoberta e exploração de uma infraestrutura, armazenando a informação recolhida numa base de dados, e faz o povoamento de uma CMDB.

Nesta fase, é possível fazer uma comparação entre os resultados obtidos e os objetivos levantados na fase inicial do projeto.

Perceber como funcionam as tecnologias que implementam CMDBs e como é que os dados recolhidos podem ser armazenados.
Foram analisadas cinco ferramentas de software, tendo sido possível perceber como é que a ferramenta desenvolvida poderia ser integrada com estas. Foi necessário, essencialmente, perceber como era possível obter o modelo de dados das CMDBs, assim como o povoamento destas poderia ser efetuado. Foi verificado que é possível obter o modelo de dados através das APIs das CMDBs ou através da análise das suas bases de dados. Relativamente ao povoamento, este pode ser efetuado através das APIs destas.

Desenvolver uma arquitetura para a automatização do processo de criação de uma CMDB.
Durante o desenvolvimento de todas as fases, procurou-se implementar o maior número possível de mecanismos automáticos, de forma a automatizar todo o processo.
Na fase de descoberta, foi implementada a descoberta automática da infraestrutura, que é capaz de explorar as máquinas encontradas de acordo com o seu tipo.
Na fase de mapeamento, foram implementados processos que captassem os modelos de dados da base de dados gerada pelos dados recolhidos e da CMDB a utilizar, definindo automaticamente correspondências entre os elementos dos dois modelos.
Na fase de povoamento foram definidos processos capazes de transformar a informação descoberta, recorrendo às regras de transformação calculadas na fase de mapeamento, em pedidos às APIs das CMDBs.

Implementar um processo de descoberta automática dos elementos computacionais de uma organização recorrendo a ferramentas de análise de redes e inventário de sistemas.

Com este objetivo em mente, a fase de descoberta da infraestrutura foi desenvolvida de modo a que:

- seja possível personalizar o detalhe da descoberta;
- seja possível integrar qualquer mecanismo de descoberta;
- os mecanismos de descoberta sejam capazes de ser executados de acordo com os dispositivos encontrados.

Fazer o povoamento da CMDB, adaptando-o à ferramenta escolhida pelo utilizador.
Inicialmente, foi necessário desenvolver um mecanismo capaz de processar o modelo de dados da CMDB. Este recorreu à API da CMDB ou a acessos diretos à sua base de dados para obter esta informação. Foi também necessário desenvolver um mecanismo capaz de adaptar o povoamento de acordo com a API da CMDB escolhida.
Ter um mecanismo de processamento e outro de povoamento para cada software não é o mesmo que definir um mapeamento padrão para cada um. Como os modelos das CMDBs podem ser personalizados, não é possível ter um mapeamento padrão para cada solução, porque, para o mesmo software, os modelos de dados podem ser diferentes em diferentes organizações. Desta forma, os mecanismos desenvolvidos indicam apenas como os modelos de dados podem ser obtidos e como a CMDB pode ser povoada, ou seja, apesar de serem dependentes do software, são independentes do modelo de dados da CMDB.

Minimizar o esforço do utilizador na tarefa de criação da CMDB.
Ao longo de todo o processo as interações com o utilizador foram reduzidas apenas ao necessário. Este só interage com a ferramenta para:

- indicar a utilização, ou não, de ferramentas externas;
- definir os endereços a explorar;
- especificar as credenciais dos dispositivos;
- especificar a base de dados a utilizar;
- especificar a CMDB a ser utilizada;
- selecionar os mapeamentos a adotar quando estes apresentam o mesmo valor entre diferentes elementos;
- escolher o valor limite das regras de transformação a considerar.

Além disto, é também possível verificar como a ferramenta desenvolvida responde a alguns dos problemas encontrados no processo de criação de uma CMDB, enunciados na Secção 2.4.3.

Recursos humanos e financeiros limitados.
Tendo isto em conta, a implementação da ferramenta foi simplificada, resultando no desenvolvimento de uma aplicação *Command Line Interface (CLI)*. Como a interface com o utilizador não era tão relevante neste processo, foi dada uma maior importância à

implementação do processo em si. Foi também utilizada a linguagem Python, assim como código *open-source*, que é facilmente executado.

Não planeamento dos tipos, atributos e relacionamentos que se quer manter, leva à tentativa insustentável de manter demasiada informação.
Neste caso, o facto de ser possível personalizar o nível do detalhe da descoberta, garante que seja apenas recolhida informação de acordo com os objetivos do utilizador.

Utilização de uma abordagem maioritariamente manual acaba por ser insustentável tendo em conta todo o processo de criação e manutenção da CMDB.
O processo desenvolvido é automático e tenta diminuir ao máximo o esforço desempenhado pelo utilizador.

Informação desatualizada ou incorreta leva à inconsistência da CMDB.
A combinação de vários mecanismos de descoberta para a recolha de informação garante que os dados recolhidos são o mais atualizados ou exatos possíveis.

Falta de capacidade de integração com outras ferramentas.
Possibilidade de integração de qualquer mecanismo de descoberta e com qualquer CMDB.

Capacidades de reconciliação e normalização fracas, que levam à inconsistência da informação.
Foram implementados mecanismos de normalização e reconciliação capazes de minimizar a ocorrência de incoerências nos dados.

Finalmente, é possível também comparar a ferramenta desenvolvida com as ferramentas abordadas na Secção 2.7.
A ferramenta desenvolvida é capaz de oferecer capacidades de descoberta semelhantes, com a vantagem de não necessitar da definição de várias tarefas - apenas uma com definição do nível de detalhe e exploração automática de acordo com os dispositivos descobertos. Para além disso, é uma solução *open-source* em oposição ao software proprietário, e usualmente pago, destas ferramentas. Finalmente, oferece a capacidade de integração com qualquer CMDB em oposição à integração com apenas a CMDB do mesmo fornecedor de software da ferramenta de descoberta.

6.2 TRABALHO FUTURO

O trabalho apresentado nesta dissertação implementa os processos essenciais da criação de uma CMDB. No entanto, este processo pode ser melhorado e até desenvolvido de forma a executar outras funções.
Em vez de fazer apenas a criação da CMDB, a ferramenta poderia também ser adaptada para fazer a atualização desta. Isto poderia ser conseguido começando por processar os dados já armazenados na CMDB. Depois, a descoberta poderia ser baseada nesta informação, e o mecanismo de reconciliação poderia ser utilizado de forma a conciliar a nova informação descoberta com aquela que já se encontrava na CMDB. Em vez enviar pedidos de criação à API, seria necessário adaptar estes para fazer a atualização dos CIs e relacionamentos. Neste caso, também a forma como o utilizador interage com a ferra-

menta pode ser modificada. No caso da criação da CMDB, é esperado que o processo seja realizado apenas uma vez. Assim, a introdução dos dados ao longo da execução do processo não apresenta qualquer desvantagem. No entanto, no caso da atualização da CMDB, é esperado que o processo seja executado várias vezes. Assim, a utilização de um ficheiro de configuração seria mais conveniente, uma vez que a informação não iria variar, ou seja, o processo em princípio iria ser executado sempre com base no mesmo ficheiro de configuração, fazendo com que o utilizador não tenha de introduzir manualmente a mesma informação.

Além disto, a ferramenta poderia também ser aperfeiçoada, tendo também já sido enunciadas algumas destas melhorias na Secção 5.5:

- ser posível o utilizador não só selecionar as categorias, como também os mecanismos de descoberta a serem efetivamente executados;
- implementar outros mecanismos de descoberta de modo a explorar outros tipos de dispositivos e recolher outros tipos de dados;
- ser capaz de inferir informação com base nos dados recolhidos;
- fazer a integração com outras ferramentas externas de descoberta;
- basear o cálculo de similaridade apenas no domínio de linguagem das TIs;
- ser possível efetuar o mapeamento entre múltiplos elementos;
- fazer a integração com outras CMDBs.

BIBLIOGRAFIA

[1] Combodo - CMDB and ITSM open source software iTop. https://www.combodo.com/?lang=en. Acedido: 06-02-2021.

[2] i-doit CMDB & IT Documentation: Open Source Solution for yout IT. https://www.i-doit.org. Acedido: 30-11-2019.

[3] ISO - International Organization for Standardization. https://www.iso.org/home.html. Acedido: 26-11-2019.

[4] iTop: open source ITIL ITSM CMDB Software. https://www.combodo.com/itop-193. Acedido: 28-12-2019.

[5] JSON. https://www.json.org/json-en.html. Acedido: 14-02-2021.

[6] Synetics GmbH - We are i-doit - About us | i-doit. https://www.i-doit.com/en/about-us/. Acedido: 06-02-2021.

[7] Text.casing. https://hackage.haskell.org/package/casing-0.1.4.1/docs/Text-Casing.html. Acedido: 16-04-2021.

[8] ISO/IEC 7498-1:1994(en), Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model — Part 1. https://www.iso.org/obp/ui/#iso:std:iso-iec:7498:-1:ed-1:v2:en, 1994. Acedido: 26-11-2019.

[9] CIM Overview Document. https://www.dmtf.org/sites/default/files/Why%20CIM%20Overview%20Document_2010.pdf, jun 2003.

[10] mysql · PyPI. https://pypi.org/project/mysql/, 2015. Acedido: 26-03-2021.

[11] Cisco Discovery Protocol Configuration Guide, Cisco IOS Release 15M&T. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/cdp/configuration/15-mt/cdp-15-mt-book/nm-cdp-discover.html, 2016. Acedido: 15-01-2021.

[12] Knowledge Base - English - Knowledge Base. https://kb.i-doit.com/display/en, 2016. Acedido: 30-11-2019.

[13] easysnmp · PyPI. https://pypi.org/project/easysnmp/, 2017. Acedido: 26-03-2021.

[14] Resource Description Framework (RDF) Model and Syntax Specification. https://www.w3.org/TR/PR-rdf-syntax/, 2017. Acedido: 30-12-2020.

[15] ISO/IEC 20000-1:2018(en) Information technology — Service management — Part 1: Service management system requirements. https://www.iso.org/obp/ui/#iso:std:iso-iec:20000:-1:ed-3:v1:en, 2018. Acedido: 26-11-2019.

[16] NetXMS - Open-source network and infrastructure monitoring and management system. https://www.netxms.org, 2018. Acedido: 01-02-2020.

[17] Windows Remote Management - Win32 apps | Microsoft Docs. https://docs.microsoft.com/en-us/windows/win32/winrm/portal, 2018. Acedido: 30-12-2020.

[18] Windows Management Instrumentation - Win32 apps | Microsoft Docs. https://docs.microsoft.com/pt-pt/windows/win32/wmisdk/wmi-start-page?redirectedfrom=MSDN, 2018. Acedido: 23-12-2020.

[19] The Hitchhiker's Guide to Device42 - Device42 Documentation | Device42 Documentation. https://docs.device42.com, 2019. Acedido: 20-05-2020.

[20] wordninja · PyPI. https://pypi.org/project/wordninja/, 2019. Acedido: 02-04-2021.

[21] Home - Documentation for BMC CMDB 20.02 - BMC Documentation. https://docs.bmc.com/docs/ac2002, 2020. Acedido: 27-12-2019.

[22] Getting started - Documentation for BMC Discovery 20.02 - BMC Documentation. https://docs.bmc.com/docs/discovery/120/getting-started-911455884.html, 2020. Acedido: 30-05-2020.

[23] CIM | DMTF. https://www.dmtf.org/standards/cim, 2020. Acedido: 07-02-2020.

[24] Device42 | Discovery & Assessment for IT Asset Management & Migration. https://www.device42.com, 2020. Acedido: 20-05-2020.

[25] Home | DMTF. https://www.dmtf.org, 2020. Acedido: 23-12-2020.

[26] GraphDB Downloads and Resources. https://graphdb.ontotext.com, 2020. Acedido: 30-12-2020.

[27] iTop Community Wiki [iTop Documentation]. https://www.itophub.io/wiki/page?id=2_7_0%3Astart, 2020. Acedido: 13-02-2021.

[28] LibreNMS. https://www.librenms.org, 2020. Acedido: 30-04-2020.

[29] MariaDB Foundation - MariaDB.org. https://mariadb.org, 2020. Acedido: 23-11-2020.

[30] MySQL. https://www.mysql.com, 2020. Acedido: 23-12-2020.

[31] nltk · PyPI. https://pypi.org/project/nltk/, 2020. Acedido: 26-03-2021.

[32] paramiko · PyPI. https://pypi.org/project/paramiko/, 2020. Acedido: 26-03-2021.

[33] requests · PyPI. https://pypi.org/project/requests/, 2020. Acedido: 26-03-2021.

[34] Configuration Management Database — ServiceNow Docs. https://docs.servicenow.com/bundle/newyork-servicenow-platform/page/product/configuration-management/concept/c_ITILConfigurationManagement.html, 2020. Acedido: 02-01-2020.

[35] #1 IT Network Inventory - Track your network. 100% free. https://www.spiceworks.com/free-pc-network-inventory-software/?source=navbar-drawer&utm_campaign=inventory&utm_medium=platform_redirect&utm_source=community, 2020. Acedido: 12-01-2020.

[36] WordNet — A Lexical Database for English. https://wordnet.princeton.edu, 2020. Acedido: 14-10-2020.

[37] BMC Software – Run and Reinvent. https://www.bmcsoftware.pt, 2021. Acedido: 06-02-2021.

[38] Device42 | Discovery & Assessment for IT Asset Management & Migration. https://www.device42.com, 2021. Acedido: 06-02-2021.

[39] Empowering App Development for Developers | Docker. https://www.docker.com, 2021. Acedido: 15-01-2021.

[40] Broadcom Inc. — Connecting Everything. https://www.broadcom.com, 2021. Acedido: 17-03-2021.

[41] Linux.org. https://www.linux.org, 2021. Acedido: 06-02-2021.

[42] macOS Big Sur - Apple. https://www.apple.com/macos/big-sur/, 2021. Acedido: 06-02-2021.

[43] PHP: Hypertext Preprocessor. https://www.php.net, 2021. Acedido: 17-03-2021.

[44] python-nmap · PyPI. https://pypi.org/project/python-nmap/, 2021. Acedido: 26-03-2021.

[45] pyshark · PyPI. https://pypi.org/project/pyshark/, 2021. Acedido: 26-03-2021.

[46] Welcome to Python.org. https://www.python.org, 2021. Acedido: 17-01-2021.

[47] ServiceNow – The smarter way to workflow. https://www.servicenow.com, 2021. Acedido: 06-02-2021.

[48] Discovery basics. https://docs.servicenow.com/bundle/quebec-it-operations-management/page/product/discovery/concept/c_GetStartedWithDiscovery.html, 2021. Acedido: 16-04-2021.

[49] spacy · PyPI. https://pypi.org/project/spacy/, 2021. Acedido: 26-03-2021.

[50] Welcome To UML Web Site! https://www.uml.org, 2021. Acedido: 15-01-2021.

[51] Vagrant by HashiCorp. https://www.vagrantup.com, 2021. Acedido: 06-02-2021.

[52] Oracle VM VirtualBox. https://www.virtualbox.org, 2021. Acedido: 15-01-2021.

[53] Explorar o SO Windows 10, Computadores, Aplicações e Muito Mais | Microsoft. https://www.microsoft.com/pt-pt/windows/, 2021. Acedido: 06-02-2021.

[54] C. Araujo. *The Quantum Age Of IT*. It Governance Publishing, 2012.

[55] J. Azevedo. Jason-Azevedo/PasswordVault: Command line app that stores encrypted passwords in an integrated database. https://github.com/Jason-Azevedo/PasswordVault, 2020. Acedido: 30-12-2020.

[56] A. Baron, A. Vinberg, A. T. Hopper, A. J. Sanghvi, D. R. Kumar, G. Zizys, N. G. Cain, and V. Rajarajan. Configuration management database state model, July 2010. US Patent 7,756,828.

[57] M. J. Bearden, L. Denby, B. Karacali, J. Meloche, and D. T. Stott. Network topology discovery systems and methods, Sept. 2013. US Patent 8,543,681.

[58] D. Beckett, T. Berners-Lee, E. Prud'hommeaux, and G. Carothers. RDF 1.1 Turtle. https://www.w3.org/TR/turtle/, 2014. Acedido: 30-12-2020.

[59] T. Berners-Lee, L. Masinter, and M. McCahill. RFC 1738 - Uniform Resource Locators (URL). https://tools.ietf.org/html/rfc1738, 1994. Acedido: 23-12-2020.

[60] A. Boyko, V. Varkentin, and T. Polyakova. Advantages and Disadvantages of the Data Collection's Method Using SNMP. In *2019 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)*, pages 1–5, Oct 2019. doi: 10.1109/FarEastCon.2019.8934069.

[61] J. Case, M. Fedor, M. Schoffstall, and J. Davin. RFC 1157 - Simple Network Management Protocol (SNMP). https://tools.ietf.org/html/rfc1157, 1990. Acedido: 30-12-2020.

[62] J. Case, R. Mundy, D. Partain, and B. Stewart. RFC 3410 - Introduction and Applicability Statements for Internet-Standard Management Framework. https://tools.ietf.org/html/rfc3410, 2002. Acedido: 30-12-2020.

[63] P. Congdon. Link Layer Discovery Protocol and MIB. https://www.ieee802.org/1/files/public/docs2002/lldp-protocol-00.pdf, 2002. Acedido: 06-02-2021.

[64] N. Das, M. Q. Llacuna, and R. B. Ryali. Link layer discovery protocol (LLDP) on multiple nodes of a distributed fabric, Aug. 2017. US Patent 9,743,367.

[65] A. Doan, A. Halevy, and Z. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.

[66] D. N. Drogseth, R. Sturm, and D. Twing. *CMDB Systems*. Morgan Kaufmann, 2015.

[67] Y. Faihe, A. Boulmakoul, and M. Salle. Configuration management database and system, Apr. 2011. US Patent 7,926,031.

[68] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1. https://tools.ietf.org/html/rfc2616, 1999. Acedido: 23-12-2020.

[69] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[70] V. Fuller and T. Li. RFC 4632 - Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan. https://tools.ietf.org/html/rfc4632, 2006. Acedido: 06-02-2021.

[71] F. Giunchiglia and P. Shvaiko. Semantic matching. *The Knowledge Engineering Review*, 18, 09 2003. doi: 10.1017/S0269888904000074.

[72] W. Goralski. *The Illustrated Network: How TCP/IP Works in a Modern Network*. Morgan Kaufmann, 2017.

[73] L. Guddemi. Why 85% Of Companies Fail At Creating A Configuration Management Database (CMDB). https://www.forbes.com/sites/sungardas/2014/10/22/why-85-of-companies-fail-at-creating-a-cmdb-and-how-to-escape-their-fate/, 2014. Acedido: 26-12-2019.

[74] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2011.

[75] S. Hollenbeck, M. Rose, and L. Masinter. RFC 3470 - Guidelines for the Use of Extensible Markup Language (XML) within IETF Protocols. https://tools.ietf.org/html/rfc3470, 2003. Acedido: 23-12-2020.

[76] J. Hwang and M. Kim. Effective Detecting Method of Nmap Idle Scan. *Journal of Advanced Information Technology and Convergence*, 9(1):1–10, 2019.

[77] Intellica.AI. Comparison of different Word Embeddings on Text Similarity — A use case in NLP | by Intellica.AI | Medium. https://medium.com/@Intellica.AI/comparison-of-different-word-embeddings-on-text-similarity-a-use-case-in-nlp-e83e08469c1c, 2019. Acedido: 16-10-2020.

[78] N. Joukov, M. V. Devarakonda, K. Magoutis, and N. Vogl. Built-to-Order Service Engineering for Enterprise IT Discovery. In *2008 IEEE International Conference on Services Computing*, volume 2, pages 91–98, 2008. doi: 10.1109/SCC.2008.128.

[79] A. Keks. Angry IP Scanner - the original IP scanner for Windows, Mac and Linux. https://angryip.org, 2019. Acedido: 30-04-2020.

[80] L. Klosterboer. *Implementing ITIL Configuration Management*. IBM Press, 2008. ISBN-10: 0-13-242593-9.

[81] H. Kyurkchiev and K. Kaloyanova. Logical Design for Configuration Management Based on ITIL. *Information Systems & Grid Technologies*, 2012.

[82] S. Laan. *IT Infrastructure Architecture - Infrastructure Building Blocks and Concepts Third Edition*. Lulu.com, 2017.

[83] A. Limited. *ITIL Foundation, ITIL 4 Edition*. The Stationery Office, 2019. ISBN: 9780113316076.

[84] S. C. H. Lin. Multi-stage network discovery, June 2019. US Patent App. 10/326,656.

[85] D. Liu, M. Caceres, A. E. Earle, D. Ganger, W. Jayawickrama, J. Kanlirz Jr., T. Robichaux, E. S. Seangren, B. Smith, and C. Stokes. *Next Generation SSH2 Implementation*. Syngress, 2008.

[86] G. F. Lyon. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security*. Nmap Project, 2009.

[87] J. Madhavan, P. Bernstein, and E. Rahm. Generic Schema Matching with Cupid. *Proc 27th VLDB Conference*, 07 2001.

[88] A. D. C. Marcelino. *Visão crítica da gestão e controlo de projetos com ISO 20000*. PhD thesis, NOVA Information Management School, 2018.

[89] T. McAlpin. Can people be configuration items? *Viewpoint: Focus on CMDB Leadership*, pages 156–159, 05 2006.

[90] P. Mockapetris. RFC 1035 - Domain names - implementation and specification. https://tools.ietf.org/html/rfc1035, 1987. Acedido: 06-02-2021.

[91] J. Moy. RFC 2328 - OSPF Version 2. https://tools.ietf.org/html/rfc2328, 1998. Acedido: 23-12-2020.

[92] T. Müller. A project guideline for implementing a configuration management database. https://www.iet-solutions.com/files/6114/6418/0878/eng_CMDB_5_Steps_2016.pdf, 2016.

[93] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. RFC 4861 - Neighbor Discovery for IP version 6 (IPv6). https://tools.ietf.org/html/rfc4861, 2007. Acedido: 23-12-2020.

[94] B. Owen, J. Gerbasi, A. C. Dhuleshia, A. Arom-Zohar, G. Grisco, C. Nguyen, and O. Subayi. Operation of device and application discovery for a managed network, July 2 2019. US Patent 10,341,841.

[95] S. Pandey, M.-J. Choi, S.-J. Lee, and J. W. Hong. IP network topology discovery using SNMP. In *2009 International Conference on Information Networking*, pages 1–5, 2009.

[96] A. Pawar and V. Mago. Calculating the similarity between words and sentences using a lexical database and corpus statistics. *CoRR*, 02 2018.

[97] R. Perlman. An algorithm for distributed computation of a spanning tree in an extended lan. In *SIGCOMM 1985*, 1985.

[98] D. C. Plummer. RFC 826 - An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. https://tools.ietf.org/html/rfc826, 1982. Acedido: 23-12-2020.

[99] J. Postel. RFC 792 - Internet Control Message Protocol. https://tools.ietf.org/html/rfc792, 1981. Acedido: 23-12-2020.

[100] J. Postel and J. Reynolds. RFC 959 - File Transfer Protocol. https://tools.ietf.org/html/rfc959, 1985. Acedido: 16-04-2021.

[101] R. Presuhn. RFC 3416 - Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP). https://tools.ietf.org/html/rfc3416, 2002. Acedido: 30-12-2020.

[102] E. Rekhter, Y., E. Li, T., and E. S. Hares. A Border Gateway Protocol 4 (BGP-4). https://www.rfc-editor.org/info/rfc4271, 2006.

[103] E. Rescorla. RFC 2818 - HTTP Over TLS. https://tools.ietf.org/html/rfc2818, 2000. Acedido: 17-03-2021.

[104] S. Salam, S. Aravamudan, S. M. Krishnamurhty, V. R. Evuri, V. Shenoy, and A. T. N. Murthy. Method and apparatus for mapping network data models, Jan. 2018. US Patent App. 15/201,781.

[105] G. O. Santos and J. C. Reis. Identificação de mapeamentos entre ontologias em diferentes línguas. Technical report, Universidade Estadual de Campinas - Instituto de Computação, 2018.

[106] T. Schaaf and B. Gögetap. Requirements and Recommendations for the Realization of a Configuration Management Database. *Service Management*, 2011.

[107] T. A. Singer, S. L. Christenson, and B. Beheshti. System and method for network auto-discovery and configuration, Dec. 2004. US Patent 6,834,298.

[108] B. Srinivasa-Desikan. *Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras*. Packt Publishing Ltd, 2018.

[109] B. Tabbara. Method and apparatus for discovering network devices, July 2014. US Patent 8,775,584.

[110] R. Thurlow. RFC 5531 - RPC: Remote Procedure Call Protocol Specification Version 2. https://tools.ietf.org/html/rfc5531, 2009. Acedido: 23-12-2020.

[111] T. Wang, S. Truptil, and F. Bénaben. An automatic model-to-model mapping and transformation methodology to serve model-based systems engineering. *Information Systems and e-Business Management*, 15, 05 2017.

[112] Z. Wu and M. Palmer. Verbs semantics and lexical selection. Association for Computational Linguistics, 1994. doi: 10.3115/981732.981751.

[113] T. Ylonen and C. Lonvick. RFC 4252 - The Secure Shell (SSH) Authentication Protocol. https://tools.ietf.org/html/rfc4252, 2006. Acedido: 30-12-2020.

A

RESULTADOS DO PROCESSAMENTO DE MODELOS DE DADOS

Listagem A.1: Resultado (parcial) do processamento do modelo de dados do i-doit.

```
{
    "ci_types": {
        "C__OBJTYPE__APPLICATION": "Application",
        "C__OBJTYPE__SERVER": "Server",
        "C__OBJTYPE__SWITCH": "Switch",
        "C__OBJTYPE__ROUTER": "Router",
        "C__OBJTYPE__ACCESS_POINT": "Wireless Access Point",
        "C__OBJTYPE__LAYER3_NET": "Layer 3-net",
        "C__OBJTYPE__CONTAINER": "Container",
        "C__OBJTYPE__OPERATING_SYSTEM": "Operating System",
        "C__OBJTYPE__HOST": "Host",
        "C__OBJTYPE__IT_SERVICE": "Service",
        "C__OBJTYPE__DBMS": "Dbms",
        [...]
    },
    "rel_types": {
        "C__RELATION_TYPE__SOFTWARE": "Software assignment",
        "C__RELATION_TYPE__NETWORK_PORT": "Ports",
        "C__RELATION_TYPE__LOCATION": "Location",
        "C__RELATION_TYPE__IP_ADDRESS": "Host address",
        "C__RELATION_TYPE__DEFAULT": "Dependency",
        "C__RELATION_TYPE__DATABASE_INSTANCE": "Database instance",
        "C__RELATION_TYPE__NET_CONNECTIONS": "Network connections",
        "C__RELATION_TYPE__OPERATION_SYSTEM": "Operating system",
        [...]
    },
    "ci_attributes": {
        "C__OBJTYPE__APPLICATION": {
            "description": "Description",
            "price": "Investment costs",
            "order_date": "Order date",
            "guarantee_date": "Guarantee date",
            "title": "Title",
            "id": "ID",
```

```
            "created": "Creation date",
            "changed": "Last change",
            "cmdb_status": "CMDB status",
            "type": "Type",
            "application_type": "Type",
            "assigned_license": "Assigned license",
            [...]
        },
        "C__OBJTYPE__SERVER": {
            "mount_point": "Drive letter",
            "title": "Title",
            "filesystem": "Filesystem",
            "serial": "Serial number",
            "free_space": "Free Diskspace",
            "used_space": "Used Diskspace",
            "description": "Description",
            "manufacturer": "Manufacturer",
            "price": "Investment costs",
            "ip_address": "(Source) ip address",
            "id": "ID",
            "status": "Condition",
            [...]
        },
        "C__OBJTYPE__SWITCH": {
            "title": "Title",
            "description": "Description",
            "acquirementdate": "Date of invoice",
            "price": "Investment costs",
            "id": "ID",
            "status": "Condition",
            "created": "Creation date",
            "created_by": "Created by",
            "changed": "Last change",
            "changed_by": "Last change by",
            "latitude": "Latitude",
            "longitude": "Longitude",
            "manufacturer": "Manufacturer",
            "serial": "Serial number",
            "width": "Width",
            "height": "Height",
            "depth": "Depth",
            "weight": "Weight",
            "ipv6_address": "IPv6 address",
            [...]
        },
        "C__OBJTYPE__ROUTER": {
            "title": "Title",
```

```
            "description": "Description",
            "id": "ID",
            "status": "Condition",
            "manufacturer": "Manufacturer",
            "mac": "MAC",
            "layer2_assignment": "Layer 2 Net",
            "ipv4_address": "IPv4 address",
            "ipv6_address": "IPv6 address",
            "hostname": "Hostname",
            "domain": "Domain",
            "routing_protocol": "Routing protocol",
            "gateway_address": "Gateway address"
            [...]
        },
        "C__OBJTYPE__ACCESS_POINT": {
            "description": "Description",
            "inventory_no": "Inventory number",
            "title": "Title",
            "status": "Condition",
            "manufacturer": "Manufacturer",
            "serial": "Serial number",
            "firmware": "Firmware",
            "interface": "Connected interface",
            "ssid": "SSID",
            [...]
        },
        "C__OBJTYPE__LAYER3_NET": {
            "description": "Description",
            "title": "Title",
            "status": "Condition",
            "netmask": "Netmask",
            "gateway": "Default Gateway",
            "range_from": "DHCP from",
            "range_to": "DHCP to",
            "dns_server": "DNS server",
            "dns_domain": "DNS domain",
            "cidr_suffix": "CIDR-Suffix",
            "reverse_dns": "Reverse DNS",
            "address_range": "Address range",
            [...]
        },
        "C__OBJTYPE__CONTAINER": {
            "description": "Description",
            "id": "ID",
            "title": "Title",
            "status": "Condition",
            "created": "Creation date",
```

```
            "created_by": "Created by",
            [...]
        },
        "C__OBJTYPE__OPERATING_SYSTEM": {
            "title": "Title",
            "kernel": "Kernel",
            "ram": "RAM",
            "cpu": "CPU frequency",
            "disc_space": "Disc space",
            "status": "Condition",
            "common_name": "Common Name",
            "assigned_version": "Version number",
            "specification": "Specification",
            "manufacturer": "Manufacturer",
            "install_path": "Install path",
            [...]
        },
        "C__OBJTYPE__HOST": {
            "mount_point": "Drive letter",
            "title": "Title",
            "filesystem": "Filesystem",
            "serial": "Serial number",
            "free_space": "Free Diskspace",
            "used_space": "Used Diskspace",
            "description": "Description",
            "manufacturer": "Manufacturer",
            "model": "Model",
            "firmware": "Firmware",
            "price": "Investment costs",
            "storage_type": "Storage type",
            "network_type": "Network type",
            "cores": "CPU cores",
            [...]
        },
        "C__OBJTYPE__IT_SERVICE": {
            "service_id": "Service-ID",
            "service_level": "SLA service level",
            "reaction_time": "Reaction time",
            "recovery_time": "Recovery time",
            "description": "Description",
            "alert_level": "Alarm level",
            "comment": "Comment",
            "service_number": "Service number",
            "service_description_intern": "Service description internal",
            "service_description_extern": "Service description external",
            "service_alias": "Aliases",
            "title": "Title",
```

```
            "status": "Condition",
            [...]
        },
        "C__OBJTYPE__DBMS": {
            "title": "Title",
            "status": "Condition",
            "assigned_database_schema": "Database schema",
            "dbms": "DBMS",
            "variant": "Variant",
            "application_type": "Type",
            "application_priority": "Priority",
            "assigned_license": "Assigned license",
            "assigned_license_key": "Assigned license key",
            "assigned_version": "Version number",
            [...]
        },
        [...]
    },
    "rel_attributes": {
        "C__RELATION_TYPE__SOFTWARE": {
            "object1": "Object1",
            "object2": "Object2",
            "relation_type": "Relation type",
            "weighting": "Weighting",
            "description": "Description",
            "title": "Title",
            "status": "Condition",
            [...]
        },
        "C__RELATION_TYPE__NETWORK_PORT": {
            "object1": "Object1",
            "object2": "Object2",
            "relation_type": "Relation type",
            "weighting": "Weighting",
            "description": "Description",
            "title": "Title",
            "status": "Condition",
            "created": "Creation date",
            "created_by": "Created by",
            "net_type": "Type",
            "net": "Net",
            "zone": "Net zone",
            [...]
        },
        "C__RELATION_TYPE__LOCATION": {
            "object1": "Object1",
            "object2": "Object2",
```

```
            "relation_type": "Relation type",
            "weighting": "Weighting",
            "description": "Description",
            [...]
        },
        "C__RELATION_TYPE__IP_ADDRESS": {
            "object1": "Object1",
            "object2": "Object2",
            "relation_type": "Relation type",
            "weighting": "Weighting",
            "description": "Description",
            "application": "Application",
            "title": "Title",
            "status": "Condition",
            "primary_layer3_net": "Primary layer3 net",
            "primary_hostaddress": "Primary hostaddress",
            "primary_hostname": "Primary hostname",
            "net_type": "Type",
            "ipv4_address": "IPv4 address",
            "dns_server": "DNS Server",
            "dns_server_address": "DNS Server address",
            [...]
        },
        "C__RELATION_TYPE__DEFAULT": {
            "object1": "Object1",
            "object2": "Object2",
            "relation_type": "Relation type",
            "weighting": "Weighting",
            "description": "Description",
            "title": "Title",
            "status": "Condition",
            "created": "Creation date",
            "created_by": "Created by",
            [...]
        },
        "C__RELATION_TYPE__DATABASE_INSTANCE": {
            "object1": "Object1",
            "object2": "Object2",
            "relation_type": "Relation type",
            "weighting": "Weighting",
            "description": "Description",
            "assigned_databases": "Used databases",
            "id": "ID",
            "title": "Title",
            "status": "Condition",
            [...]
        },
```

```
        "C__RELATION_TYPE__NET_CONNECTIONS": {
            "object1": "Object1",
            "object2": "Object2",
            "relation_type": "Relation type",
            "weighting": "Weighting",
            "description": "Description",
            "id": "ID",
            "title": "Title",
            "status": "Condition",
            "net_type": "Type",
            "zone": "Net zone",
            "assigned_logical_port": "Assigned port",
            [...]
        },
        "C__RELATION_TYPE__OPERATION_SYSTEM": {
            "object1": "Object1",
            "object2": "Object2",
            "relation_type": "Relation type",
            "weighting": "Weighting",
            "description": "Description",
            "title": "Title",
            "status": "Condition",
            "user_name_static": "User",
            [...]
        },
        [...]
    },
    "ci_attributes_data_types": {
        "C__OBJTYPE__SERVER": {
            "mount_point": "text",
            "title": "text",
            "filesystem": "int",
            "serial": "text",
            "free_space": "float",
            "used_space": "float",
            "description": "text_area",
            "manufacturer": "int",
            "price": "double",
            "ip_address": "int",
            "id": "int",
            "status": "int",
            [...]
        },
        [...]
    },
    "rel_attributes_data_types": {
        "C__RELATION_TYPE__NETWORK_PORT": {
```

```
            "object1": "int",
            "object2": "int",
            "relation_type": "int",
            "weighting": "int",
            "description": "text_area",
            "title": "text",
            "status": "int",
            "created": "text",
            "created_by": "text",
            "net_type": "int",
            "net": "int",
            "zone": "int",
            [...]
        },
        [...]
    "rel_restrictions": {},
    "ci_dialog_attributes": {
        "C__OBJTYPE__SERVER": {
            "status": {
                "3": "Archived",
                "1": "Unready",
                "4": "Deleted",
                "2": "Normal",
                "6": "Template",
                "7": "Mass change template"
            },
            "cmdb_status": {
                "1": "planned",
                "2": "ordered",
                "3": "delivered",
                "4": "assembled",
                "5": "tested",
                "6": "in operation",
                "7": "defect",
                "8": "under repair",
                "9": "delivered from repair",
                "10": "inoperative",
                "11": "stored",
                "12": "scrapped"
            },
            "net_type": {
                "1": "IPv4 (Internet Protocol v4)",
                "2": "IPX (Internet Packet Exchange)",
                "3": "AT (AppleTalk)",
                "1000": "IPv6 (Internet Protocol v6)"
            },
            [...]
```

```
        },
        [...]
    },
    "rel_dialog_attributes": {}
}
```

B

EXECUÇÃO DA FERRAMENTA

```
================================================================================
 ____ __  __ ____ ____      _          _                          _ _
 / ___|  \/  |  _ \| __ )    / \   _   _| |_ ___  _ __ ___   __ _| |_(_) ___
| |   | |\/| | | | |  _ \   / _ \ | | | | __/ _ \| '_ ` _ \ / _` | __| |/ __|
| |___| |  | | |_| | |_) | / ___ \| |_| | || (_) | | | | | | (_| | |_| | (__
 \____|_|  |_|____/|____/ /_/   \_\\__,_|\__\___/|_| |_| |_|\__,_|\__|_|\___|

  ____                 _
 / ___|_ __ ___  __ _| |_(_) ___  _ __
| |   | '__/ _ \/ _` | __| |/ _ \| '_ \
| |___| | |  __/ (_| | |_| | (_) | | | |
 \____|_|  \___|\__,_|\__|_|\___/|_| |_|


================================================================================
>>> Initializing the vault...
>>> Vault exists.
>>> Creating the database file...
>>> Defining vault password...

? Enter your vault password:  *********
? Repeat your vault password:  *********

>>> Your vault has been created and encrypted with your master key.

**************************************************************************
 ___ _                                  ___ _
|   \(_)___ __ _____ _____ _ _ _  _    | _ \ |_  __ _ ___ ___
| |) | (_-</ _/ _ \ V / -_) '_| || |   |  _/ ' \/ _` (_-</ -_)
|___/|_/__/\__\___/\_/\___|_|  \_, |   |_| |_||_\__,_/__/\___|
                               |__/

**************************************************************************

? Do you want to import information from an external application/source?  No
? Enter the IPv4 address (yyx.yyx.yyx.yyx, where 'yy' is optional), IPv4 range (yyx.yyx.yyx.yyx-zzz, where 'yy' and '-zzz' are
optional) or CIDR (yyx.yyx.yyx.yyx/yx, where 'y' is optional) you want to discover.  192.168.1.73
? Do you want to specify another IPv4 address or range?  No

>>> Exploring packets in the interface lo0.
>>> Exploring packets in the interface en0.

? Enter your vault password:  *********
>>> Vault unlocked.

? Do you want to specify another SNMP community string? ("public" is the default)  No

>>> Checking the availability of the IP address 192.168.1.73...

>>> NMAP discovery in the address 192.168.1.73...

>>> Unable to retrieve SNMP information.

>>> Basic discovery ended.

? In which IT infrastructure categories would you want to make a more detailed discovery?  [operating systems]

? Enter your vault password:  *********

>>> Vault unlocked.

? Enter the username of the machine with yhe IP address 192.168.1.73:  Joana
? Enter the password of the machine with the IP address 192.168.1.73:  ******************
>>> Password added to the vault.
>>> Discovery in the OS X machine with the address 192.168.1.73...

? Enter the IP address of the GraphDB server (use format yyx.yyx.yyx.yyx where 'y' is optional):  192.168.1.88
? Enter the port number where GraphDB is running:  7200
? Enter the name of the GraphDB repository:  cmdb

>>> Successfully connected.

>>> Data imported into GraphDB.
```

Figura B.1: Exemplo de execução da fase de descoberta.

```
**********************************************************************
 __  __                   _           ___ _
|  \/  |__ _ _ __ _ __ __(_)_ _  __ _ | _ \ |_  __ _ ___ ___
| |\/| / _` | '_ \ '_ \ | ' \/ _` | |  _/ ' \/ _` (_-</ -_)
|_|  |_\__,_| .__/ .__/_|_||_\__, | |_| |_||_\__,_/__/\___|
            |_|  |_|         |___/

**********************************************************************
>>> Processing GraphDB data model...

>>> The database model was successfully processed.
? What is the software of the CMDB?  i-doit

>>> Make sure that i-doit is running.
? Enter the IP address of your CMDB server (use format yyx.yyx.yyx.yyx where 'y' is optional):  192.168.1.88
? Enter your CMDB username:  admin
? Enter your CMDB password:  *****
? Enter your API key:  cmdb-auto

>>> Checking API connection...
>>> Successfully connected.

? Enter the IP address of your database server (use format yyx.yyx.yyx.yyx where 'y' is optional):  192.168.1.88
? Enter your database name:  idoit_data
? Enter your database username:  root
? Enter your database password:  ********

>>> Checking i-doit database connection...
>>> Successfully connected to the i-doit database.
>>> Processing i-doit CMDB data model...
>>> Executing the model mapper...

>>> Calculating configuration item types similarity...
? The similarities between 'Host' with 'C__OBJTYPE__SERVER' and 'C__OBJTYPE__HOST' are equal. Choose the one to
consider.  C__OBJTYPE__HOST
? The similarities between 'Product' with 'C__OBJTYPE__SERVER' and 'C__OBJTYPE__IT_SERVICE' are equal. Choose the one to
consider.  C__OBJTYPE__IT_SERVICE
? The similarities between 'Product' with 'C__OBJTYPE__IT_SERVICE' and 'C__OBJTYPE__PRINTER' are equal. Choose the one to
consider.  C__OBJTYPE__IT_SERVICE

>>> Calculating relationship types similarity...
? The similarities between 'port from' with 'C__RELATION_TYPE__NETWORK_PORT' and 'C__RELATION_TYPE__UNIVERSAL_INTERFACE' are
equal. Choose the one to consider.  C__RELATION_TYPE__NETWORK_PORT

>>> Calculating configuration item attributes similarity...
? The similarities between 'protocol' with 'description', 'plug_type' and 'comment' are equal. Choose the one to
consider.  comment

>>> Calculating relationship attributes similarity...
```

Figura B.2: Exemplo de execução do processamento e mapeamento dos modelos de dados da base de dados e da CMDB.

```
================================================================================================================================================
CONFIGURATION ITEMS MAPPING
================================================================================================================================================
CI in CMDB                    Description                    CI in DB          Description         Similarity Coeficient
----------------------------  -----------------------------  ----------------  ----------------  ------------------------
C__OBJTYPE__HOST              Host                           Host              host                                    1
C__OBJTYPE__OPERATING_SYSTEM  Operating System               Operating System  operating system                        1
C__OBJTYPE__APPLIANCE         Appliance                      Product           product                                 0.75
C__OBJTYPE__APPLICATION       Application                    Protocol          protocol                                0.615385
C__OBJTYPE__RM_CONTROLLER     Remote Management Controller   Logical Port      logical port                            0.588253
************************************************************************************************
C__OBJTYPE__HOST Attributes Mapping
************************************************************************************************
Attribute in CMDB    Description                          Attribute in DB    Description                    Similarity Coeficient
-------------------  -----------------------------------  -----------------  -----------------------------  ------------------------
title                Title                                title              title                                                1
status               Condition                            status             status                                               1
serial               Serial number                        serial_number      serial number                                        1
application          Application                          os                 operating system                                     0.842105
port_mode            Mode                                 boot mode          boot mode                                            0.795953
system_drive         System drive                         os_family          operating system family                              0.766736
system               Virtualization system                os version         operating system version                             0.693314
ldev                 On device Logical devices (Client)   os name            operating system name                                0.669005
time_needed          Time need                            time since boot    time since boot                                      0.72518
assigned_version     Version number                       has_ipv4           internet protocol version 4                          0.672933
firmware             Firmware                             kernel version     kernel version                                       0.581027
wwn                  Node WWN                             hostname           hostname                                             0.450317
************************************************************************************************
C__OBJTYPE__OPERATING_SYSTEM Attributes Mapping
************************************************************************************************
Attribute in CMDB    Description         Attribute in DB    Description                Similarity Coeficient
-------------------  ------------------  -----------------  -------------------------  ------------------------
assigned_version     Version number      version number     version number                                    1
title                Title               title              title                                             1
install_path         Install path        build version      build version                                     0.568523
operation_expense    Operation expense   os family          operating system family                           0.700423
************************************************************************************************
C__OBJTYPE__APPLIANCE Attributes Mapping
************************************************************************************************
Attribute in CMDB    Description    Attribute in DB    Description      Similarity Coeficient
-------------------  -------------  -----------------  -------------  ------------------------
title                Title          title              title                                 1
description          Description    description        description                           1
************************************************************************************************
C__OBJTYPE__APPLICATION Attributes Mapping
************************************************************************************************
Attribute in CMDB    Description    Attribute in DB    Description      Similarity Coeficient
-------------------  -------------  -----------------  -------------  ------------------------
title                Title          title              title                                 1
************************************************************************************************
C__OBJTYPE__RM_CONTROLLER Attributes Mapping
************************************************************************************************

Attribute in CMDB    Description    Attribute in DB    Description                  Similarity Coeficient
-------------------  -------------  -----------------  ---------------------------  ------------------------
title                Title          title              title                                               1
status               Condition      status             status                                              1
event_static         Event          product            product                                             0.909091
itservice            Service        cpe                customer premises equipment                      0.652566
comment              Comment        protocol           protocol                                            0.615385
```

Figura B.3: Exemplo dos mapeamentos calculados entre os CIs da base de dados e da CMDB e respetivos atributos.

```
==================================================================================================================================================
RELATIONSHIPS MAPPING
==================================================================================================================================================

Relationship in CMDB                    Description                 Relationship in DB   Description                Similarity Coeficient
--------------------------------------- --------------------------- -------------------- -------------------------- ----------------------
C__RELATION_TYPE__NETWORK_PORT          Ports                       port from            port                                            1
C__RELATION_TYPE__ORGANIZATION          Organization                has protocol         protocol                                 0.571429
C__RELATION_TYPE__OPERATION_SYSTEM      Operating system            running os           running operating system                 0.942343
C__RELATION_TYPE__CONTROLLER_FC_PORT    FC port                     is running on port   running port                             0.672716
C__RELATION_TYPE__RM_CONTROLLER         Remote Management Controller installed os        installed operating system               0.670873
C__RELATION_TYPE__IT_SERVICE_COMPONENT  Service component           running product      running product                           0.55349

**************************************************************************************************
C__RELATION_TYPE__NETWORK_PORT Attributes Mapping
**************************************************************************************************
Attribute in CMDB    Description    Attribute in DB    Description    Similarity Coeficient
-------------------  -------------  -----------------  -------------  -----------------------
title                Title          title              title                                1
**************************************************************************************************
C__RELATION_TYPE__ORGANIZATION Attributes Mapping
**************************************************************************************************
Attribute in CMDB    Description    Attribute in DB    Description    Similarity Coeficient
-------------------  -------------  -----------------  -------------  -----------------------
title                Title          title              title                                1
**************************************************************************************************
C__RELATION_TYPE__OPERATION_SYSTEM Attributes Mapping
**************************************************************************************************
Attribute in CMDB    Description    Attribute in DB    Description    Similarity Coeficient
-------------------  -------------  -----------------  -------------  -----------------------
title                Title          title              title                                1
**************************************************************************************************
C__RELATION_TYPE__CONTROLLER_FC_PORT Attributes Mapping
**************************************************************************************************

Attribute in CMDB    Description    Attribute in DB    Description    Similarity Coeficient
-------------------  -------------  -----------------  -------------  -----------------------
title                Title          title              title                                1
**************************************************************************************************
C__RELATION_TYPE__RM_CONTROLLER Attributes Mapping
**************************************************************************************************
Attribute in CMDB    Description    Attribute in DB    Description    Similarity Coeficient
-------------------  -------------  -----------------  -------------  -----------------------
title                Title          title              title                                1
**************************************************************************************************
C__RELATION_TYPE__IT_SERVICE_COMPONENT Attributes Mapping
**************************************************************************************************
Attribute in CMDB    Description    Attribute in DB    Description    Similarity Coeficient
-------------------  -------------  -----------------  -------------  -----------------------
title                Title          title              title                                1
[? Enter the threshold value that you want to consider (similarities below that value will not be considered):  0.85
```

Figura B.4: Exemplo dos mapeamentos calculados entre os relacionamentos da base de dados e da CMDB e respetivos atributos.

```
**********************************************************************
 ___              _      _   _            ___ _
| _ \___ _ __ _  _| |__ _| |_(_)___ _ _   | _ \ |_  __ _ ___ ___
|  _/ _ \ '_ \ || | / _` |  _| / _ \ ' \  |  _/ ' \/ _` (_-</ -_)
|_| \___/ .__/\_,_|_\__,_|\__|_\___/_||_| |_| |_||_\__,_/__/\___|
        |_|

**********************************************************************
>>> Obtaining the existing CI's in the database...

>>> Obtaining the existing relationships in the database...

>>> Starting the population of the CMDB...

>>> Creating the configuration items...

>>> Error converting string '10.15.7' to int.
>>> Object of type C__OBJTYPE__OPERATING_SYSTEM created successfully in the CMDB.

>>> Object of type C__OBJTYPE__HOST created successfully in the CMDB.

>>> Creating the relationships...

>>> Object of type C__RELATION_TYPE__OPERATION_SYSTEM created successfully in the CMDB.

>>> Successfully disconnected.

>>> CMDB population completed.

? Do you want to delete the password vault?  Yes
```

Figura B.5: Exemplo de execução da fase de povoamento.

C

DETALHES DA EXECUÇÃO DOS TESTES

Tabela C.1: Exemplo de alguns atributos de CIs descobertos no primeiro caso de teste.

Identificador	Nome	Tipo	Atributo	Valor
110macbookprojoana	macbookprojoana	Host	os version	10.15.7
			kernel version	Darwin 19.6.0
			battery	bq20z451
			os	macOS 10.15.7 (19H15)
			boot mode	normal_boot
			time since boot	up 1:8:11:46
			CPU	Quad-Core Intel Core i7
			CPU speed	2;2 GHz
			display	Color LCD
			CPU frequency	2;2
			CPU frequency unit	GHz
			CPU cores	4
			physical memory	16 GB
			memory ram	16
			physical memory unit	GB
			GPU	kHW_IntelIrisProItem
			hostname	macbookprojoana.local
			os name	Mac OS X
			description	MacBook Pro MacBook ProMacBook Pro
			has_ipv4	192.168.1.73
			has_ipv6	2001:8a0:f576:7000:19c8:36e6:6c25:e634
			has_ipv6	2001:8a0:f576:7000:1cac:32c1:d287:1804
			mac_address	c4:b3:01:c0:1a:d3
			os_family	macOS
			serial_number	C02S80K6G8WN
			status	up
			uuid	0B793129-B10E-5C77-89DD-4529FEF05AD6
1211mac_0_s_10_15_7_19_h15	macOS 10.15.7 19H15	Operating System	os family	Mac OS X
			version number	10.15.7
			build version	19H15
3021py_charm	PyCharm	Application	arch kind	arch_i32_i64
			info	PyCharm 2020.3.4; build PY-203.7717.65. Copyright JetBrains s.r.o.; (c) 2000-2021
			last modified	2021-03-16T10:52:36Z
			path	/Applications/PyCharm.app
			obtained from	identified_developer
			description	2020.3.4
1312quad__core__intel__core_i7	Quad-Core Intel Core i7	CPU	Speed	2;2 GHz
			Number of Cores	4
			L2 Cache	256 KB
			L3 Cache	6 MB
			SMC Version	2.29f24
1615p_t	PT	Location	country	Portugal
			region	Braga
			city	Barcelos
			zip_code	4750-106
			time_zone	Europe/Lisbon
			latitude	41.5388
			longitude	-8.6171
1816b_a_n_k_1_d_i_m_m0	BANK 1DIMM0	RAM	manufacturer	0x802C
			part number	0x31364B544631473634485A2D314736453120
			size	8 GB
			speed	1600 MHz
			architecture	DDR3
			serial_number	-
			status	ok
1917a_p_p_l_e__s_s_d__s_m0256_g	APPLE SSD SM0256G	SSD Controller	bsd name	disko
			detachable drive	no
			device model	APPLE SSD SM0256G
			device revision	BXZ13A0Q
			partition map type	guid_partition_map_type
			removable media	no
			size	251 GB
			medium type	Solid State
			ncq	Yes
			ncq depth	32
			link speed	8.0 GT/s
			link width	x4
			physical interconnect	PCI
			port description	AHCI Version 1.30 Supported
			product	SSD Controller
			vendor	Apple
			serial_number	S2Z5NY0H811261

Tabela C.2: Exemplo de alguns atributos de CIs descobertos no segundo caso de teste.

Identificador	Nome	Tipo	Atributo	Valor
1817ubuntu_pc	ubuntu-pc	Host	Icon name	computer-desktop
			Chassis	desktop
			Machine ID	93b6da13546247df9c2c4e89abccfea3
			Boot ID	d17d71c7440943af87c6abc7a6db7822
			Operating System	Ubuntu 18.04.5 LTS
			vendor	Dell
			Kernel	Linux 5.4.0-70-generic
			Architecture	x86-64
			os family	Ubuntu
			os name	Ubuntu 18.04.5 LTS
			os version	18.04
			CPU cores	4
			CPU	Intel(R) Core(TM) i5-3470S CPU @ 2.90GHz
			CPU frequency unit	MHz
			CPU frequency	1596.396
			CPU speed	1596.396 MHz
			has_ipv4	192.168.1.88
			mac_address	F8:B1:56:AD:16:2F
			os_family	Linux
			status	up
2019ubuntu_18_04_5__l_t_s	Ubuntu 18.04.5 LTS	Operating System	vendor	Ubuntu
			DISTRIB_RELEASE	18.04
			DISTRIB_CODENAME	bionic
			DISTRIB_DESCRIPTION	Ubuntu 18.04.5 LTS
			ID	ubuntu
			ID_LIKE	debian
			version number	18.04
			HOME_URL	https://www.ubuntu.com/
			SUPPORT_URL	https://help.ubuntu.com/
			BUG_REPORT_URL	https://bugs.launchpad.net/ubuntu/
			PRIVACY_POLICY_URL	https://www.ubuntu.com/legal/terms-and-policies/privacy-policy
			VERSION_CODENAME	bionic
			UBUNTU_CODENAME	bionic
			description	18.04.5 LTS (Bionic Beaver)
2120intel_r__core_t_m_i5_3470_s__c_p_u__2_90_g_hz	IntelR CoreTM i5-3470S CPU 2.90GHz	CPU	Architecture	x86_64
			CPU op-modes	32-bit; 64-bit
			Byte Order	Little Endian
			CPUs	1
			On-line CPUs list	0-3
			Threads per core	1
			Cores per socket	4
			NUMA nodes	0-3
			Vendor ID	GenuineIntel
			CPU family	6
			Stepping	9
			CPU frequency unit	MHz
			CPU frequency	1596.396
			CPU speed	1596.396 MHz
			CPU max MHz	3600;0000
			CPU min MHz	1600;0000
			BogoMIPS	5787.05
			Virtualization	VT-x
			L1d cache	6144K
			Flags	fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca...
			description	58
2521sda4	sda4	SSD	PARTUUID	0780dcb0-04
			Filesystem	/dev/sda4
			1K-blocks	177521492
			Used	96%
			Available	7430508
			Mounted on	/
			NAME	sda4
			MAJ:MIN	08:04
			RM	0
			SIZE	173G
			RO	0
			MOUNTPOINT	/
			description	part partpart partpart partpart
			uuid	57170c8d-8978-48f5-8c8e-ed4e1f32f182
2625graphdb_free	graphdb-free	Application	Architecture	amd64
			description	GraphDB 9.3.1
2725gnome_shell	gnome-shell		Architecture	amd64
			description	graphical 3.28.4-0ubuntu18.04.3
2825firefox	firefox		Architecture	amd64
			description	Safe 87.0+build3-0ubuntu0.18.04.2
2925snapd	snapd		Architecture	amd64
			description	Daemon 2.48.3+18.04
3025evolution_data_server	evolution-data-server		Architecture	amd64
			description	evolution 3.28.5-0ubuntu0.18.04.3

This work is financed by National Funds through the Portuguese funding agency, Fundação para a Ciência e a Tecnologia (FCT), within project UIDB/50014/2020.