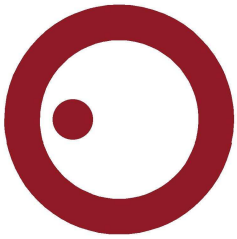


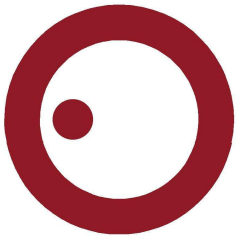
escola superior de tecnologia e gestão
instituto politécnico de leiria

Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Cibersegurança e Informática Forense

INTELLIGENCE BUSINESS CENTER

ESTUDANTE DANIEL MENDES PINTO

Leiria, dezembro de 2020



escola superior de tecnologia e gestão
instituto politécnico de leiria

Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Cibersegurança e Informática Forense

INTELLIGENCE BUSINESS CENTER

ESTUDANTE DANIEL MENDES PINTO
Número: 2170104

Estágio realizado sob orientação do Professor Doutor Filipe Neves (fneves@ipleiria.pt).

Leiria, dezembro de 2020

AGRADECIMENTOS

Concluir esta etapa não foi um processo fácil, no entanto existiram sempre pessoas capazes de me ajudar. Agradeço a oportunidade que tive ao longo da minha licenciatura e mestrado a várias pessoas que me serviram de exemplo, fizeram-me crescer a nível profissional e chegar hoje até aqui.

Agradeço ao meu orientador de mestrado, Professor Doutor Filipe Neves, que me ajudou a motivar para a escrita da tese e acreditou em mim, pela sua assertividade e pela sua paciência.

Ao Professor Mestre Vitor Carreira por ser sempre um exemplo durante a minha licenciatura e durante os anos que tive a oportunidade de trabalhar com ele. Inspirou-me a ser um bom programador, e foi graças a este incrível profissional que aprendi boas bases informáticas e ganhei motivação para querer ser sempre melhor na minha área.

Ao Professor Doutor José Magno Lopes, que sempre ensinou aos seus alunos que programar muito não é sinónimo de boa programação, mas que o mais importante é saber pensar. Um bom raciocínio prévio é o que destaca um bom programador.

Agradeço ainda aos seguintes professores que serviram para mim de exemplo: Patrício Domingues, Catarina Silva, Marisa Maximiano, Mário Antunes e Alexandra Nascimento.

Por fim, quero também agradecer aos meus pais a oportunidade de acreditarem em mim e aos meus amigos que sempre me incentivaram a continuar.

RESUMO

Pretende-se, com este estágio, implementar uma *framework* modular, com o objetivo de analisar, de uma forma geral e específica, a informação que circula dentro e fora de uma empresa, com o fim de compreender se a informação relativa à empresa está segura ou comprometida. Procura-se atingir este objetivo através do conhecimento de entidades e contactos, pela análise de dados de ficheiros e metadados, de e-mails, domínios, redes sociais, fugas de informação, tráfego de rede e registo de eventos internos e externos sobre a empresa. Tenciona-se também analisar as relações existentes entre a informação obtida, isto é, de que forma a informação se cruza e até que ponto se compromete ou cria possíveis pontos fracos de cibersegurança, afetando o grau de segurança da empresa.

ABSTRACT

It is intended, with this traineeship, to implement a modular framework, in order to analyze, in a general and specific way, the information that circulates inside and outside of a company, in order to understand if the information relative to the company is safe or compromised. It seeks to achieve this goal through the knowledge of entities and contacts, by analyzing data from files and metadata, from e-mails, domains, social networks, leaks, network traffic and by recording internal and external events about the company. It is also intended to analyze the existing relationships between the obtained information, that is, how the information connects itself and the extent to which it is compromised or possible creates weaknesses in cybersecurity, affecting the degree of security of the company.

ÍNDICE

Apêndices

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ACL	Access Control List.
ADSL	Assimetric Digital Subscriber Line.
API	Application Programming Interface.
ASCII	American Standard Code for Information Interchange.
BD	Base de dados.
BIOS	Basic Input/Output System.
bit	Digito binário.
Byte	Unidade de informação digital composta por oito bits.
CODEC	COmpression/DECompression.
CPU	Central Processing Unit.
CQL	Cypher Query Language.
CRUD	Create, Retrieve, Update and Delete.
CSS	Cascading Style Sheets.
CSV	Comma-Separated Values.
DB	Database.
DDoS	Denial-of-service attack.
DHCP	Dynamic Host Configuration Protocol.
DLL	Dynamic Link Library.
DNS	Domain Name System.
DOM	Document Object Model.
FTP	File Transfer Protocol.
HIDS	Host-based Intrusion Detection System.

HTML	Hypertext Markup Language.
HTTP	HyperText Transfer Protocol.
IBC	Intelligence Business Center.
IDS	Intrusion detection system.
IoT	Internet of Things.
IP	Internet Protocol.
IPS	Intrusion prevention system.
ISP	Internet Service Provider.
JSON	JavaScript Object Notation.
k-NN	K-Nearest Neighbors.
NIDS	Network-based Intrusion Detection System.
ORM	Object-relational mapping.
OSInt	Open-source intelligence.
PCA	Principal Component Analysis.
POO	Programação Orientada a Objetos.
PYPL	PopularitY of Programming Language.
REST	REpresentational State Transfer.
SIEM	Security Information and Event Management.
SO	Sistema Operativo.
SoC	Separation of Concerns.
SPAM	Sending and Posting Advertisement in Mass.
SQL	Structured Query Language.
SSH	Secure SHell.
TCP	Transmission Control Protocol.
TI	Tecnologias da Informação.

UI	User Interface.
URL	Uniform Resource Locator.
UUID	Universally Unique Identifier.

1 INTRODUÇÃO

No âmbito do Mestrado em Cibersegurança e Informática Forense, durante o período de estágio, o projeto descrito neste relatório pretendia desenvolver a *framework* Intelligence Business Center, capaz de analisar informação relacionada a uma empresa, por forma a permitir uma análise intuitiva que permitisse indicar se a Segurança da Informação da organização se encontra ou não comprometida.

O objetivo principal é fornecer um serviço personalizado às empresas que pretendam adquirir um software que proteja toda a estrutura informática da companhia e seus dados, à semelhança de um software de Gestão de Eventos e da Segurança da Informação [Security Information and Event Management (SIEM)].

1.1 MOTIVAÇÃO

Um cliente empresarial pretende uma solução que permita capturar, processar e mostrar, de forma clara e intuitiva, os dados informáticos da sua empresa. Além disso, quer contextualizar e categorizar esta informação criando classificações do tipo de dados e relações com outros. O cliente quer perceber como os dados e as suas relações entre si podem comprometer a segurança da informação. Após um pré-estudo do mercado sobre as soluções SIEM mais avançadas, concluiu-se que:

1. O cliente não pretende gastar tanto dinheiro nas alternativas de software existentes no mercado, pois estas têm um custo elevado.
2. Face ao objetivos do cliente, não existia um software ajustável às suas necessidades. Em vez disso, teria de obter várias soluções de *softwares* diferentes.
3. O cliente não tem de adquirir um serviço imediato. Pretendia uma solução flexível e adaptada ao seu contexto e está disposto a esperar, obtendo uma solução a longo prazo, de forma incremental.

Com efeito, surge a oportunidade, no âmbito do mestrado, de criar uma *framework* capaz de se assemelhar, em parte, a um SIEM, embora mais leve, e com serviços que

permitam ir além do que um SIEM oferece. Idealmente, pretende-se uma *framework* que:

1. Processe a informação da empresa de várias fontes de dados;
2. Permita visualizar, graficamente, os dados informáticos, para deles obter um significado semântico;
3. Tenha um módulo de alarmística de eventos ocorridos e gestão de relatórios periódicos;
4. Faça um reconhecimento da rede empresarial, obtendo, de forma geral, a estrutura informática;
5. Forneça um conjunto de ferramentas para obter informação que pode estar indiretamente ligada à empresa, mas que a pode comprometer, como por exemplo: Descoberta de *leaks* e traçamento de perfis de redes sociais;

Atualmente, no mercado, existem diversos produtos capazes de satisfazer estes requisitos. Nenhum objetivo proposto pelo presente documento visa a descobrir um serviço que ainda não exista ainda no mercado. Contudo, o fator diferenciador é não em reinventar a roda, mas, ao invés, "juntar várias rodas", criando um «veículo capaz de dirigir», de forma segura, a segurança da informação da organização, sem comprometer o financiamento da empresa, ao ser necessário adquirir várias soluções alternativas, muitas das quais funcionam por subscrições.

A inovação do projeto passa por agregar vários serviços que disponibilizam ferramentas para a Gestão da Segurança da Informação, como já acontece com as outras soluções no mercado, e ainda acrescentar tecnologias que permitam ao utilizador comum uma capacidade de fazer testes de penetração para avaliação da segurança da empresa e técnicas OSInt para exploração de terreno que possa ter associações com a empresa.

O projeto é ambicioso e irá ser desenvolvido ao longo dos anos. É esperado, com a conclusão do estágio, que o projeto continue e que possa entrar no mercado.

1.2 OBJETIVOS

No final do projeto, pretende-se alcançar os seguintes objetivos a curto prazo:

1. Criação de agentes de processamento de dados para os módulos: entidades, e-mails, *leaks* e domínios;

2. Serviço Web API para consulta e administração da informação recolhida;
3. Criação de relações lógicas entre os dados processados, para melhor compreensão do contexto da informação.

Ainda, a longo prazo, pretende-se estender este projeto com os seguintes objetivos:

1. Desenvolvimento de painel de análise de informação;
2. Módulo de análise de redes sociais;
3. Módulo de análise de rede;
4. Módulo de alarmística e gestão de relatórios;
5. Integração com várias plataformas da empresa, tais como: dispositivos IoT, *hardware*, *software* e outros serviços;
6. Implementação de técnicas para predição ataques e falhas de segurança, integrando a solução com IDS e outras tecnologias, nomeadamente Inteligência Artificial;

1.3 ESTRUTURA DO DOCUMENTO

O presente relatório segue a seguinte estrutura:

Inicia-se com o Trabalho Relacionado 2, onde menciona e explica as soluções no mercado comparativamente ao projeto que se quer desenvolver, bem como o estado da arte da Inteligência Artifical nesta área.

De seguida, segue-se o capítulo do Planeamento 3 onde é discutido, sumariamente, o plano traçado para alcançar os objetivos acima propostos 1.2, nomeadamente as metodologias de Engenharia de Software que serão utilizadas e as linguagens e bibliotecas mais importantes que serão utilizadas no trabalho.

Posteriormente, dá-se início ao capítulo de Desenvolvimento 4 onde explica minuciosamente o projeto e a arquitetura da *framework* desenvolvida, repleta de imagens retiradas da aplicação *web* (parte visual ou UI), para facilitar a compreensão.

Por fim, no capítulo da Conclusão 5 discutem-se as dificuldades sentidas ao longo do projeto, os resultados obtidos e o trabalho que fica para futuro.

Ainda existem, no final do documento, dois anexos A e B, que foram incluídos para que todas as dependências utilizadas pela aplicação sejam expostas ao público, não só em forma de agradecimento pelas ferramentas gratuitas disponibilizadas

por tantos indivíduos, como também para reforçar a ideia de que existe sempre, em parte, uma dependência externa destes recursos, que podem comprometer a eficácia, a robustez e até a segurança da aplicação, no contexto de um Mestrado em Cibersegurança e Informática Forense, onde é importante frisar que nunca há nada 100% seguro.

2

TRABALHO RELACIONADO

Atualmente existem inúmeras soluções para a gestão e segurança da informação das empresas. Muitas dessas soluções atuais conseguem inclusive prevenir ataques bem arquiteturados, antigamente muito difíceis de mitigar, como o caso do DDoS. Estas tecnologias possuem algoritmos complexos de pesquisa de informação e agentes poderosos que atuam para mitigar os ataques. Torna-se cada vez mais frequente (e até mesmo uma necessidade) ter uma solução SIEM que assegure a empresa. Até em empresas mais pequenas, cujo capital não permite ter uma solução tão eficaz, é frequente existir um agente controlador que pelo menos permita alertar eventos anormais, como as *firewalls* e os IDS.

A complexidade dos ataques evoluiu de tal forma que é um bem-essencial à empresa recorrer a uma solução de segurança. A solução mais complexa é um SIEM, porque atualmente é capaz de mastigar praticamente qualquer tipo de dados, vindos de qualquer fonte, e muitas destas soluções atuais até recorrem à Inteligência Artificial e ao *Data Mining*. Os SIEM permitem uma integração robusta com todo o sistema, tipicamente *firewalls*, antivírus, *routers*, *switchers*, IDS e infraestruturas do sistema.

Tendo em conta a evolução da complexidade acima mencionada, surgem constantemente algoritmos de Inteligência Artificial como meio de combate no auxílio à Segurança da Informação. Desta feita, dedicar-se-á uma secção do trabalho relacionado que se refere à Inteligência Artificial. Tendo em consideração que o projeto atual não inclui qualquer tipo de Inteligência Artificial, é, no entanto, importante salientar a sua importância e estudar os algoritmos mais comuns para uma possível aplicabilidade num módulo de alarmística.

2.1 SISTEMAS DE DETEÇÃO DE INTRUSÕES

A melhor forma de prevenção e mitigação contra ataques à segurança da informação era um uso de um IDS. Ainda hoje trata-se de um meio indispensável e eficaz. No entanto, conforme irá ser discutido abaixo em 2.2, existem soluções de prevenção e

mitigação ainda mais completas que integram estes sistemas e cruzam a informação de várias fontes, para uma solução ainda mais eficaz.

Um IDS, à semelhança de outras tecnologias de segurança, como as *firewalls*, filtragem de tráfego em *routers* e *switchers*, ACL's e antivirus, tem, como objetivo principal, a deteção (e possivelmente prevenção) de ataques, através da análise exaustiva do sistema ou conjunto de sistemas 2.1.3.

2.1.1 *Definição*

Considere-se as seguintes definições de um IDS:

> *"An intrusion-detection system dynamically monitors the actions taken in a given environment, and decides whether these actions are symptomatic of an attack or constitute a legitimate use of the environment."*
>
> *[15]*

> *"Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network."*
>
> *[5]*

Deste modo, é possível resumir a definição de um IDS como sendo uma tecnologia que, dado um conjunto de variáveis de ambiente, classifica eventos como sendo "normais"ou "anormais".

Um Intrusion prevention system (IPS) é semelhante em definição. No entanto, a diferença entre estes dois é que um IPS, para além de detetar ataques, também os previne, tornando-se este o seu objetivo principal, a prevenção ativa (pró-ativo) em vez de uma mitigação como resposta a um incidente (reativo). Um IPS toma uma ação face a uma ameaça, antes que o ataque prejudique o alvo, enquanto que o IDS apenas alerta, embora o fluxo continue normalmente. A resposta a desencadear face ao ataque não é da responsabilidade do IDS.

Com efeito, considera-se que os *ids* são agentes passivos na rede, que não tomam decisões, ao passo que os *ips* são agentes ativos, visto que podem alterar o estado da mesma.

Embora, à primeira vista, um *ips* pareça mais vantajoso, nem sempre se revela ser a melhor abordagem. Por exemplo, um *ips* pode desconfiar de um novo tráfego na rede, que está a efetuar uma exploração da rede. A ação a tomar é cortar esse tráfego. Porém, a ação pode ser benigna, se se tratar de uma exploração de rede por parte do departamento de segurança e redes, que pretende analisar portos ativos na infraestrutura digital da sua companhia.

2.1.2 *Objetivos*

O objetivo de um IDS está implícito na sua definição: monitorizar e detetar todo o comportamento que este considere nocivo ao ambiente em um ou vários *hosts* e/ou em uma ou mais redes, aplicando um método de análise (ou combinando vários métodos). O objetivo de um IPS é detetar e prevenir.

2.1.3 *Tipos de Sistemas de Deteção de Intrusões*

De forma simplificada, é possível categorizar os IDS de duas formas distintas [15]:

1. Classificação por localização:
   a) Baseados em *host*
   b) Baseados em rede

   Podem existir soluções de implementações com múltiplos agentes, por forma a combinar as vantagem de ambos os tipos de IDS identificados em 1. Se os agentes cruzarem a sua informação através de metodologias de sincronização, estes podem ser da seguinte forma [41, 65]:

   i. Centralizados
   ii. Distribuídos (*peer-to-peer*)
   iii. Híbridos

2. Classificação por modo de análise:
   a) Baseados em assinaturas
   b) Baseados em comportamento
   c) Híbridos

Embora estes sejam os tipos de IDS mais conhecidos, existem muitos outros tipos de categorias e subcategorias, cuja importância se desvia do assunto do tema do presente documento.

Exemplos de outras categorias:

1. Baseados em máquinas virtuais [21]
2. Baseados em perimetria [7, 9, 25]

Os IDS do pronto 1 utilizam as mais recentes inovações de tecnologia de virtualização e de *cloud computing*. Estes IDS validam máquinas virtuais. Os IDS do ponto 2 referem-se a agentes que têm instalados agentes ou sensores capazes de detetar a localização da intrusão. Um bom exemplo são os serviços de alarmes que as pessoas ou empresas subscrevem para proteger os seus imóveis.

2.1.3.1 *IDS baseados no host*

Também designados por Host-based Intrusion Detection System (HIDS), estes tipos de agentes são instalados localmente na máquina. O seu objetivo é monitorizar o sistema através da análise de pacotes de rede, aplicações, eventos e *logs* do sistema, políticas de acesso, estado do processador e da memória. São mais precisos na determinação de falsos-positivos, porque analisam várias informações, correlacionando-as. Uma desvantagem destes agentes é que ocupam processamento no *host*.

2.1.3.2 *IDS baseados na rede*

Denominados identicamente por Network-based Intrusion Detection System (NIDS), estes agentes são instalados em pontos estratégicos da rede. Podem servir para monitorizar toda ou apenas parte de uma rede. Este tipo de sistemas permite ter uma consciência do estado geral da rede, muito útil a nível empresarial. No entanto, por apenas poderem validar a rede, não são tão precisos em detecção de determinados ataques, em oposição dos HIDS.

2.1.3.3 *IDS baseados em assinaturas*

Este agentes são muito eficazes a determinar ataques conhecidos, e por essa razão também os mais utilizados no mercado. O mais conhecido é o *Snort*, da Cisco. O modo de operação é muito simples: Tipicamente, um ataque tem um vetor conhecido, também designado por padrão ou assinatura, isto é, uma "impressão

digital"(analogia), que neste caso pode identificar o ataque ou a categoria. Para que agente detete a assinatura, terá de conhecer as assinaturas malignas, assinaturas essas, que, no seu conjunto, se chama dicionário de ataques. No seu modo de funcionamento, o programa deteta o ataque ao ler o dicionário e encontrar uma assinatura que está presente no mesmo. Embora bastante eficazes, o seu calcanhar de Aquiles é a incapacidade de detetar ataques novos (*0-day*) ou cuja assinatura é desconhecida. Por isso, um agente deste tipo deve ter o seu dicionário constantemente atualizado com novas assinaturas.

2.1.3.4 *IDS baseados em anomalias*

Os IDS baseados em anomalias são agentes que classificam o tráfego ou o comportamento como "benigno"ou "maligno", alternativamente designado por "normal"ou "anormal". Estes sistemas são mais dinâmicos que os IDS baseados em assinaturas descritos em 2.1.3.3. Após a sua instalação, existe, tipicamente, um período de aprendizagem, no qual estes agentes "aprendem"o comportamento da rede e/ou sistema. Após a aprendizagem, a classificação benigna passa por eventos cujo seu padrão é conhecido, isto é típico ou comum na rede ou no sistema, enquanto que a classificação maligna ou anormal trata-se de eventos cujo seu padrão não é espectável. Este tipo de IDS podem ter um intervalo de tolerância na classificação, que os pode permitir ser mais flexíveis e inclusive adaptar-se a ambientes em constante mudança [22]. Este tipo de agentes são muito mais eficazes em classificar ataques novos ou cuja assinatura é desconhecida. No entanto, na sua maioria, este tipo de agentes criam muitos falsos-positivos, o que levanta bastantes desafios na sua utilização na vida real. Por este motivo, são menos utilizados no mercado.

2.1.3.5 *IDS híbridos*

São sistemas que tentam conciliar as vantagens de ambas as metodologias, por forma a apresentar um resultado mais preciso, no que toca à identificação se um comportamento que ocorre na rede é benigno ou maligno.

2.2 SECURITY INFORMATION AND EVENT MANAGEMENT

Um Gestor de Eventos e de Segurança da Informação é, de forma geral, um sistema que é constituído por um conjunto de tecnologias e que fornece vários serviços que permitem gerir a segurança da empresa.

2.2.1 *Definição*

Segundo [43], podemos definir um sistema SIEM pela seguinte proposição:

> *"The SIEM system is a complex collection of technologies designed to provide vision and clarity on the corporate IT system as a whole, benefitting security analysts and IT administrators as well."*
>
> *[43]*

2.2.2 *Objetivos*

Ainda segundo [43], um SIEM deve fornecer os seguintes serviços:

1. Gestão de *logs*
2. Conformidade regulatória das TI
3. Correlação de Eventos
4. Resposta Ativa
5. Segurança de *endpoints*

Pode-se afirmar que nos dias de hoje, os SIEM fazem mais do que isto. Existem soluções no mercado bastante mais elaboradas e que cobrem ainda mais conceitos, protegendo toda a infraestrutura informática da organização. O melhor exemplo figurativo de o que um SIEM faz, é imaginar um guarda-chuva que cobre toda a empresa de uma grande chuva.

2.3 APRENDIZAGEM COMPUTACIONAL

Atualmente a aprendizagem computacional é uma área muito investigada nas mais diversas áreas. Na área da Segurança, pode ser um aliado importante para a deteção de ameaças *zero-day*. Nesta secção, iremos abordar os diferentes tipos de aprendizagem computacional e os diferentes tipos de problemas.

2.3.1 *Tipos de Aprendizagem*

Dependendo do problema em análise, existem algoritmos que têm maior desempenho que outros, dependendo das circunstâncias do problema. Não há um tipo de aprendizagem melhor que outro. Depende, sobretudo, do tipo de problema, da qualidade e pré-processamento dos dados, do algoritmo escolhido, da sua correta parametrização e ainda dos resultados obtidos. Para um dado problema, considerem-se dois métodos distintos: Diz-se que um método é preferencial face ao outro quando ambos têm os dados bem processados e estão igualmente bem parametrizados, mas o desempenho do primeiro é superior ao do segundo. Diz-se então que a primeira solução encontrada adapta-se mais (é mais genérica) que a segunda, tendo em conta que não existe *overfitting*. Contudo, dados os mesmos algoritmos, para outro contexto, o segundo pode ser mais favorável. Em última consideração, pode-se afirmar que não existem algoritmos melhores ou piores, consoante o Teorema *No Free Lunch*, referido no livro "Aprendizagem Computacional"[61], que resumidamente diz, citando: "(...) todos os algoritmos de aprendizagem têm o mesmo desempenho quando se consideram todos os problemas e todos os dados possíveis".

Qualquer que seja o tipo de aprendizagem, o algoritmo deve encontrar uma solução que seja aceitável a partir de um limite mínimo definido, tal que resolva a maioria dos problemas (solução "satisfatória"). Por vezes, a solução "perfeita"que classifique bem todas as instâncias do modelo de treino pode não ser a mais conveniente. Geralmente, resulta em *overfitting*5 do modelo. Com os dados de teste, o algoritmo torna-se um mau classificador.

2.3.1.1 *Aprendizagem supervisionada*

Numa abordagem supervisionada [39, 61], o algoritmo conhece a solução esperada. O algoritmo constrói o modelo com base nos dados de entrada e no resultado esperado. Por outras palavras, o algoritmo classifica consoante a informação que lhe é fornecida e a sua capacidade de generalização. Este tipo de aprendizagem está associado a mecanismos de aprendizagem para problemas discretos e para problemas contínuos (também designados por problemas de classificação e problemas de regressão, respetivamente, como se pode verificar pela figura 1, retirada de [39]).

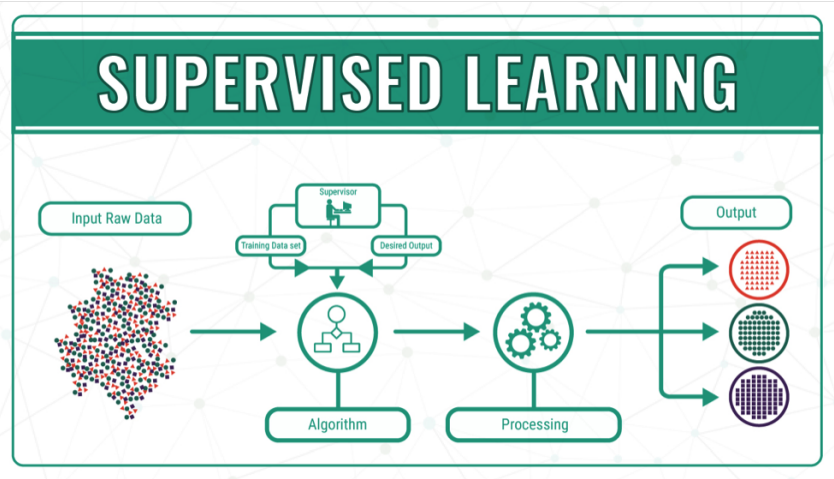


Figura 1: Aprendizagem supervisionada

2.3.1.2 *Aprendizagem não-supervisionada*

Numa abordagem não-supervisionada [39, 61] o agente irá aprender a classificar com base nas semelhanças e/ou diferenças que ele deteta nos dados. Portanto, o algoritmo não tem conhecimento antecipado da solução, como se pode verificar na figura 2, retirada de [39], que não possui um mecanismo de avaliação supervisionado, ao contrário da figura 1. Está associado aos algoritmos de *clustering*, isto é, agrupa as instâncias dos dados.

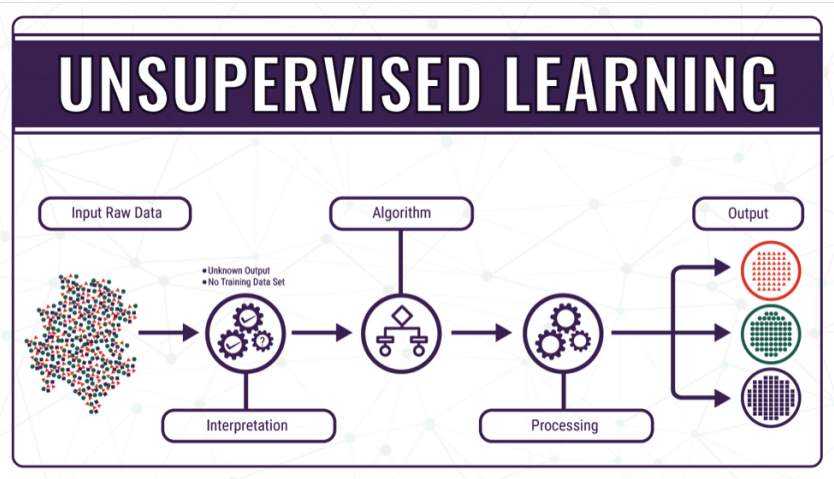


Figura 2: Aprendizagem não-supervisionada

2.3.1.3 *Aprendizagem por reforço*

Esta abordagem [39, 61] caracteriza-se pela aprendizagem do agente face ao ambiente em que se insere. O algoritmo analisa quais as ações que pode tomar e as respetivas

consequências que podem causar ao meio. Face às circunstâncias que consegue analisar e atendendo a um certo grau de incerteza do futuro, isto é, sem conhecimento da repercussão que as ações podem tomar, o algoritmo analisa qual a melhor abordagem a seguir. O objetivo é vencer o meio ambiente, adaptando-se. Deste modo, sempre que o agente atua sobre o meio, recebe um reforço (ou *feedback*), que pode ser positivo ou negativo (também designado por recompensa ou castigo). À medida que o ciclo de iterações incrementa, os efeitos que os reforços têm sobre o agente permitem que este aprenda as consequências das suas ações (ou seja, as consequências de errar - em analogia com a aprendizagem humana por tentativa/erro), aprendendo a "comportar-se"no meio ambiente. Este comportamento é exemplificado pela figura 3, retirada de [39].

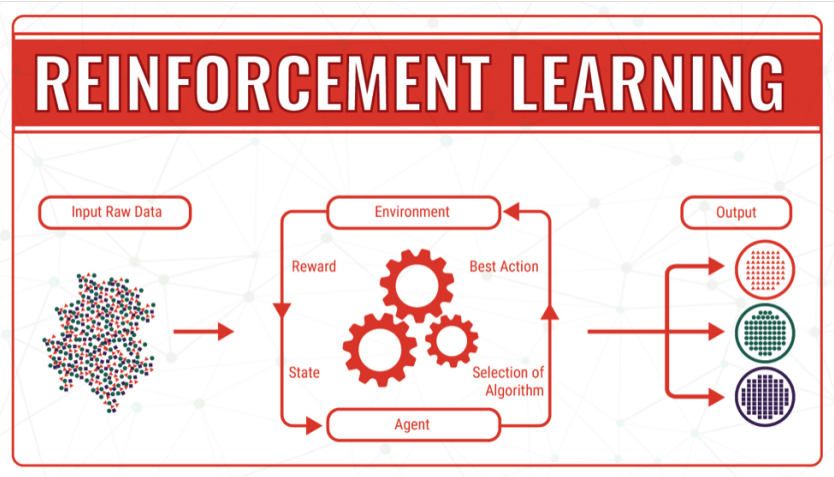


Figura 3: Aprendizagem por reforço

Pode-se comparar, na psicologia, com a teoria do comportamento operante ou comportamento instrumental, em que o sujeito recebe a recompensa ou o estímulo positivo, se executa a ação correta. A mesma teoria do comportamento operante pode funcionar de forma inversa, isto é, através de um castigo, ou seja, uma aprendizagem por estímulo negativo, sempre que o indivíduo age de maneira incorreta. Ver figura 4 retirada de [10].

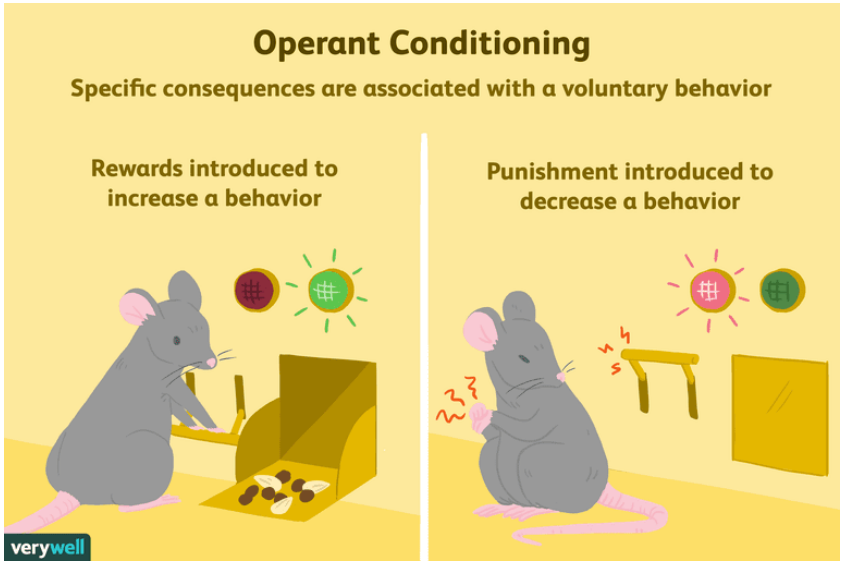


Figura 4: Teoria do Comportamento Operante

2.3.2 *Tipos de Problemas*

Existem várias formas de classificar os tipos de problemas. Contudo, pode-se generalizar a maioria os problemas em:

- Problemas de Classificação
  - Binária
  - Multi-classe
- Problemas de Regressão

Os **problemas de regressão** 2.3.2 têm como objetivo determinar um valor que, na maioria das vezes, é numérico. Exemplo: predizer o valor estimado das finanças trimestrais. Para dar este valor, teremos de fornecer um conjunto de variáveis de entrada e o histórico da situação atual da Economia. O objetivo é compreender se as ações irão diminuir, manter ou aumentar.

Os **problemas de classificação** 2.3.2 pretendem classificar os dados num tipo de classe. Um exemplo tipicamente utilizado é a classificação de e-mails normais e e-mails de SPAM. Neste caso, trata-se de um problema de classificação (mais concretamente, uma classificação binária: zero ou um, ou seja, ou é "normal"ou é "anormal").

2.4 IDS: PROBLEMA DE CLASSIFICAÇÃO

No caso de um Intrusion detection system (IDS), o objetivo é classificar se determinado pacote ou fluxo de dados pertence à classe "normal"ou "anormal". Por outras palavras, determinar se é um ataque. Com efeito, é excluída a hipótese de seguir uma abordagem para problemas de regressão.

Existem vários algoritmos de aprendizagem computacional para resolver problemas de classificação, como por exemplo:

1. Redes Neuronais
2. *k-Nearest Neighbors*
3. *k-Means Clustering*
4. Árvores de Decisão
    a) ID3
    b) J-48
    c) *Random Forest*
5. *Support Vector Machine*
6. Algoritmos Genéticos
7. Sistemas Difusos
8. Redes Naïve bayes
9. Mapas Auto-organizados

Existem algoritmos que não servem apenas para um tipo de problema. Por exemplo, os *Support Vector Machine* e os Sistemas Difusos identificados acima também são utilizados para problemas de regressão. Contudo, não é o objetivo do tema discutir as diferenças para ambos os tipos, visto que o problema a estudar é de classificação.

2.4.0.1 *Avaliação do desempenho*

As soluções de cada algoritmo precisam de ser avaliadas para determinar a qualidade da solução encontrada, tendo, deste modo, o cuidado de utilizar métricas de avaliação de performance, tais como: Taxa de Verdadeiros Positivos, Taxa de Falsos Positivos, Medida F1, Medida de Erro, Acurácia e Precisão. É também necessário garantir que o modelos não sofrem de *overfitting*. Deve-se, por isso, utilizar medidas corretas

para avaliar o modelo, que são, primeiramente, o *k-fold cross-validation* e ainda a técnica de *split* dos dados em duas partições: uma para construção de modelo e outra para validação.

A aprendizagem por reforço também pode ser estudada para o domínio do problema. Dada a necessidade de classificar um ataque, o agente pode, inicialmente, aprender a classificar se é ou não malicioso dado o ambiente de testes fornecido, mesmo não sabendo ainda bem as consequências das suas ações. Com o tempo, o agente aprende a classificar, por reforço, a rede. De seguida, ao ser fornecido um ambiente real, o estado da rede e o tendo conhecimento histórico das consequências das suas ações, o algoritmo aprende a classificar empiricamente os fluxos.

2.4.1 *Etapas da Aprendizagem Computacional*

Para uma aprendizagem computacional eficaz, é necessário seguir um conjunto de etapas que melhoram os dados [61]. Apresentam-se quatro passos importantes:

1. Pré-processamento de Dados
2. Algoritmos e Parametrização
3. Treino: Deteção e Classificação
4. Teste: Avaliação e Comparação de Resultados

2.4.1.1 *Pré-processamento de Dados*

O pré-processamento é uma etapa fundamental, obrigatória tanto para aprofundar o conhecimento do problema, mas sobretudo para uma resolução muito mais eficiente. Por exemplo, na maioria dos problemas, a cardinalidade de características é demasiado elevada para uma aprendizagem eficaz. Os dados terão de ser reduzidos. O problema que se coloca é qual a melhor estratégia a tomar para esta filtragem. É necessário garantir um corte que mantenha uma proporção semelhante de informação face à quantidade original. Senão, pode-se correr o risco de deturpar a aprendizagem.

Irão também ser abordadas técnicas para reduzir tanto o número de características e igualmente os valores possíveis que uma propriedade pode tomar. Uma das medidas mais importantes a tomar será o *PCA*. É um método de elevada importância, pois auxilia a descoberta de características correlacionadas, reduzindo o seu número

e derivando em outras. A sua definição pode ser encontrada no artigo *"Principal Componente Analysis"* [29] que, citando, descreve:

"The central idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of a large number of interrelated variables, while retaining as much as possible of the variation present in the data set. This is achieved by transforming to a new set of variables, the principal components (PCs), which are uncorrelated, and which are ordered so that the first few retain most of the variation present in all of the original variables."

Segundo o livro [61], esta fase contempla cinco vertentes:

1. Existência de dados inválidos
2. Existência de demasiados dados com muitas repetições ou não informativos
3. Necessidade de quantização e normalização
4. Filtragem, seleção de características
5. Extração de características

2.4.1.2 *Algoritmos e Parametrização*

Os algoritmos de Aprendizagem Computacional podem e devem ser parametrizados de forma que se ajustem para o melhor desempenho, sem que esta cause *overfitting*. Para compreender melhor o objetivo de uma boa parametrização que generalize melhor o modelo, segue-se uma imagem, retirada do sítio web *Apixio* [1].

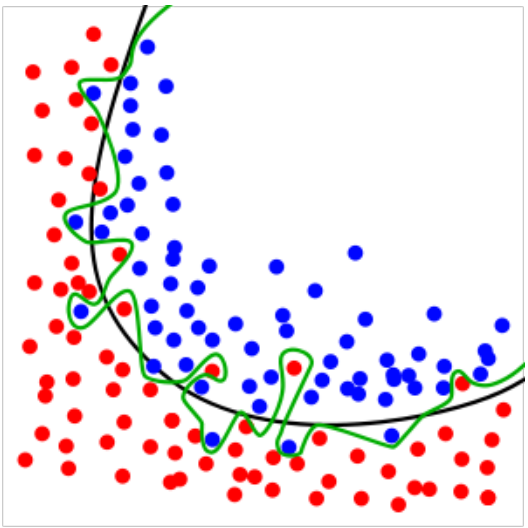


Figura 5: Classificação com *overfitting* (verde), *vs* a fronteira da melhor classificação (preto)

1 https://www.apixio.com/engineering/introduction-to-machine-learning/

Analisando a figura 5, pode-se tirar a conclusão que o *overfitting* é algo que deve ser tido em atenção e evitado, porque nem sempre os melhores resultados refletem a melhor solução.

2.4.1.3 *Treino: Deteção e Classificação*

A fase de treino recebe, como entrada, os dados previamente processados. O algoritmo constrói um modelo e classifica com base no modelo gerado. No caso dos algoritmos preguiçosos, designados em inglês como algoritmos *"lazy learning"* (como por exemplo o k-NN, ver 2.4.2.1), ao receber os valores de entrada, limita-se a classificar os dados sem a geração de um modelo.

De forma geral, podemos considerar que todos os algoritmos de aprendizagem computacional respeitam a seguinte lógica:

```
algorithm = 'SVM'  # for example

train_parameters = {
    '<parameter_1>': '<value>',
    '<parameter_2>': '<value>',
    '<...>': '<...>'
}

train_dateset = [
    [...],
    [...],
    [...]
]

model = learn(algorithm, train_parameters, train_dataset)
```

2.4.1.4 *Teste: Avaliação e Comparação de Resultados*

Por fim, a fase de teste tem como objetivo avaliar o desempenho do modelo/solução. Os dados de teste nunca devem ser os mesmos que os usados na fase anterior. No caso em que se utilizam os mesmos, não pode existir a garantia que o modelo é capaz de generalizar ou classificar bem novas instâncias. As abordagens mais comuns para avaliar o modelo são: *k-fold cross-validation* e o *split* dos dados.

Na técnica *k-fold cross-validation* o dataset é particionado em k partes distintas. Para cada valor *i* em *k*, utilizam-se *k-1* partições para avaliar os dados, excluindo a iteração *i* atual. De seguida utilizam-se as métricas de performance referidas anteriormente 2.4.0.1 para avaliar o modelo. No final, faz-se a média das avaliações para calcular a performance do modelo. Segue-se um exemplo genérico que ilustra este método de avaliação:

```
k = 10  # parameter of k-fold cross-validation
evaluations = []  # array to store the k evalutations

for i in range(k):
    test_dataset = dataset[,i] + dataset[i+1,]
    evaluation = evaluate(model, test_dataset)
    evaluations.append(evaluation)

final_evaluation = avg(evaluations)  # final evaluation
```

O método de avaliação por partição de dados (por norma, 70/30), contempla a seguinte estrutura lógica:

```
split_pct = 0.7  # percentage to split. range: ] 0, 1 [

dataset = [
    [...],
    [...],
    [...]
]

(train_dataset, test_dataset) = split(split_pct, dataset)

# train_dataset contains 70%
# test_dataset contains the remaining (30%)

"""  ******* TRAINING PHASE *******  """

train_parameters = {
    'learning_algorithm': 'k-NN',  # for example
    '<parameter_1>': '<value>',
    '<parameter_2>': '<value>',
```

```
    '<...>': '<...>'
}

model = learn(train_parameters, train_dataset)

""" ******* TESTING PHASE ******* """

final_evaluation = evaluate(model, test_dataset)
```

2.4.2 *Métodos mais usados na área dos IDS*

Atualmente a área da Inteligência Artificial é uma das áreas mais interessantes e com maior contributo científico. Por isso, é importante estar a par dos algoritmos mais importantes ou frequentemente mais utilizados. É típico encontrar artigos que misturam vários conceitos e métodos de aprendizagem, originando novas descobertas de metodologias de avaliação, com algoritmos derivados. Não obstante, existe um conjunto de métodos de aprendizagem elementares que são a base para explorar a Aprendizagem Computacional. Segue-se um conjunto de métodos interessantes para a investigação:

2.4.2.1 *k-NN*

O método k-NN calcula a distância entre diferentes pontos no espaço da amostra. De seguida, agrupa os elementos consoante uma medida de proximidade. Deste modo, os elementos são agrupados em k classes distintas, sendo k um número inteiro positivo definido pelo utilizador que representa o número de grupos para agrupamento. A deteção do valor correto para k é crucial para a performance do algoritmo:

- Se k for demasiado elevado, existirá um maior número de agrupamentos, podendo existir demasiada diversidade de classes que deveriam representar um só agrupamento. Por outras palavras, pode-se perder o poder de generalização. Em contrapartida, regra geral, quanto maior o valor de k maior será a robustez do algoritmo ao ruído dos vizinhos.
- Se k for demasiado pequeno, o classificador pode não conseguir separar bem as instâncias, agrupando casos diferentes como sendo apenas um agrupamento.

Por outro lado, poderá contribuir para uma resistência maior ao problema do *overfitting*.

O k-NN é um método muito simples e eficaz. É muito utilizado porque a sua compreensão é intuitiva e a perceção dos resultados gerados são compreendidos facilmente e igualmente demonstrados matematicamente. A desvantagem é que inicialmente os k valores pré-definidos são colocados aleatoriamente no espaço de procura. O ponto no espaço onde são inicializados determina a performance resultante. Como consequência, pode agrupar mal as instâncias.

2.4.2.2 *Árvores de Decisão*

Este tipo de aprendizagem assenta sobre regras *IF-THEN*. A partir de um valor base utilizado como raiz, são aplicadas consecutivas regras *IF-THEN*, utilizando uma estratégia de "dividir para reinar". No final, as folhas da árvore representam a classe a determinar. Este algoritmo segue uma abordagem supervisionada.

A determinação do atributo raiz e dos valores consecutivos para os nós, conforme uma dada regra decisão, utiliza fórmulas matemáticas que calcula a entropia e o ganho de informação que o atributo tem sobre as soluções (classes). Isto é dado um espaço S de possíveis caminhos de procura, pretende-se encontrar o subconjunto S tal que o subconjunto S' seja o menor caminho possível a percorrer (a menor árvore a encontrar). A influência ou peso que o valor do atributo tem para a determinação da classe é determinado pelo o ganho de informação. Deste modo, menor será a entropia dos nós subjacentes. O atributo com maior ganho de informação será a raiz da árvore. Com efeito, o espaço de procura para determinar a classe será reduzido num subconjunto menor de possibilidades para um sub-espaço de S. O algoritmo para até que todas as instâncias estejam bem classificadas ou até um limite máximo de nós (previamente definido pelo utilizador).

As árvores de decisão, embora sejam simples, apresentam um desafio no problema de generalização. Por norma, sem um limite máximo de nós, a árvore gerada pode sobre-ajustar os resultados. Por outras palavras, classifica bem para um dado conjunto de treino, mas a performance cai drasticamente no conjunto de testes. Contudo, podem ser aplicadas técnicas para podar a árvore, auxiliando a generalização do modelo. É importante destacar as *Random Forests*, um caso particular das árvores de decisão que recorre a um comité de árvores de decisão para tomar a melhor decisão. Trata-se de uma aplicação das árvores para dar melhor performance, minimizando as desvantagens mencionadas.

2.4.2.3 *Redes Neuronais*

As redes neuronais são inspirados no cérebro humano. A rede neuronal, tal como na biologia, é constituída por neurónios. Um neurónio contém as dentrites, o núcleo e o axónio. Um neurónio recebe estímulo de um ou mais neurónios através das dentrites. Assim que esta informação chega ao núcleo, o neurónio determina a ação a tomar, transmitindo o impulso elétrico pelo axónio, originando a sinapse, isto é, voltando novamente ao ciclo de retransmissão de informação pelas dentrites dos neurónios vizinhos.

Em analogia, na computação um neurónio é um percetrão que recebe vários estímulos sensoriais. Cada estímulo tem um peso associado (importância). Assim que a informação chega ao núcleo, determina-se, por uma função de ativação, se a informação deve ou não ser retransmitida para *output*. A função de ativação valida a força do estímulo em função do seu peso (importância), tendo ainda em conta uma constante *bias*, que representa a velocidade de aprendizagem. No final, conforme o resultado é ou não o pretendido, os pesos são atualizados para que permitam ajustar-se ao *output* desejado. Dependendo das implementações, por vezes o *bias* também é diminuído gradualmente. Esta escolha permite imitar ainda mais o comportamento humano. Quando o cérebro aprende um novo tema/área, inicialmente a aprendizagem é mais rápida. Com o tempo, aprofundar o conhecimento torna-se cada vez mais árduo.

Existem diversas implementações e derivações de algoritmos neste tópico. Há redes multi-camada: a primeira camada é a camada sensorial, seguem-se as camadas ocultas, por fim a última camada representa o output. Pode-se encontrar também implementações *feed-forward* e *back-propagation*. No primeiro caso, a rede só pode influenciar os pesos da próxima camada. A conexão é unilateral. No segundo caso, a rede pode influenciar os pesos das camadas anteriores. É tal a extensão sobre este tema que as redes neuronais são consideradas num subdomínio da Aprendizagem Computacional: o *Deep Learning*.

2.4.2.4 *Support Vector Machine*

Este é um algoritmo de aprendizagem supervisionada que pode ser utilizado tanto para problemas de classificação como de regressão. É conhecido por ser muito robusto e, regra geral, obtém bons resultados por ser mais imparcial ao ruído dos *outliers*. Resumidamente, o algoritmo calcula um hiperplano num espaço multi-dimensional. A solução que separa bem as instâncias, classificando-as corretamente, é

determinado pelo hiperplano que tem a maior distância relativa aos pontos fronteira que pertencem a classes distintas (em problemas de classificação). Os restantes valores são ignorados. Visto que delimitam o hiperplano, por essa razão são chamados pontos de suporte, originando o vetor de suporte da distância do hiperplano.

2.4.2.5 *Algoritmos Genéticos*

Este é mais um método baseado na biologia. A teoria deste algoritmo baseia-se na teoria darwinista da qual a sobrevivência das espécies, a sua evolução e adaptação ao meio-ambiente é determinada pela seleção natural dos indivíduos mais aptos.

O método combina vários conceitos como o gene, o genoma, o cromossoma, o genótipo, o fenótipo e a população. Ainda combina algoritmos de seleção, cruzamento e mutação. Dependendo dos autores e do nível de detalhe que se pretende, genericamente pode-se considerar: um gene representa uma característica do indivíduo. O genótipo representa todas as características (o genoma do indivíduo) e o fenótipo a representação do indivíduo no meio, isto é, a solução. O conjunto dos genótipos da população constitui o espaço de procura. O conjunto de fenótipos, o espaço de soluções.

A seleção é feita utilizando a uma função de *fitness* que avalia o desempenho do indivíduo ao meio. Os mais aptos terão maior probabilidade de seleção para cruzamento de espécies. No momento da combinação do material genético de dois indivíduos, surge um novo genótipo, isto é, um novo p' em P', em que p' é um valor do novo espaço de procura P', derivado do espaço de procura inicial, P. s' e S' representam, logicamente, uma solução no espaço de soluções derivadas de S. É no momento da troca de material que pode existir uma probabilidade K, por norma muito baixa, que influencia o material genético, alterando um ou mais genes de alguns indivíduos. Promove a diversidade das espécies, isto é, garante explorar outros espaços de procura. O número de derivações (número de gerações) pode ser determinada por um valor mínimo aceitável da função de *fitness* ou um número máximo de gerações permitidas.

2.4.2.6 *Redes Naive-Bayes*

As redes Naive-Bayes são classificadores probabilísticos, que se baseiam na probabilidade de um Evento A ocorrer, caso se verifique que o evento B ocorreu. Estes classificadores probabilísticos baseiam-se no Teorema de Bayes, que se expressa matematicamente pela expressão na figura 6, retirada de [46]:

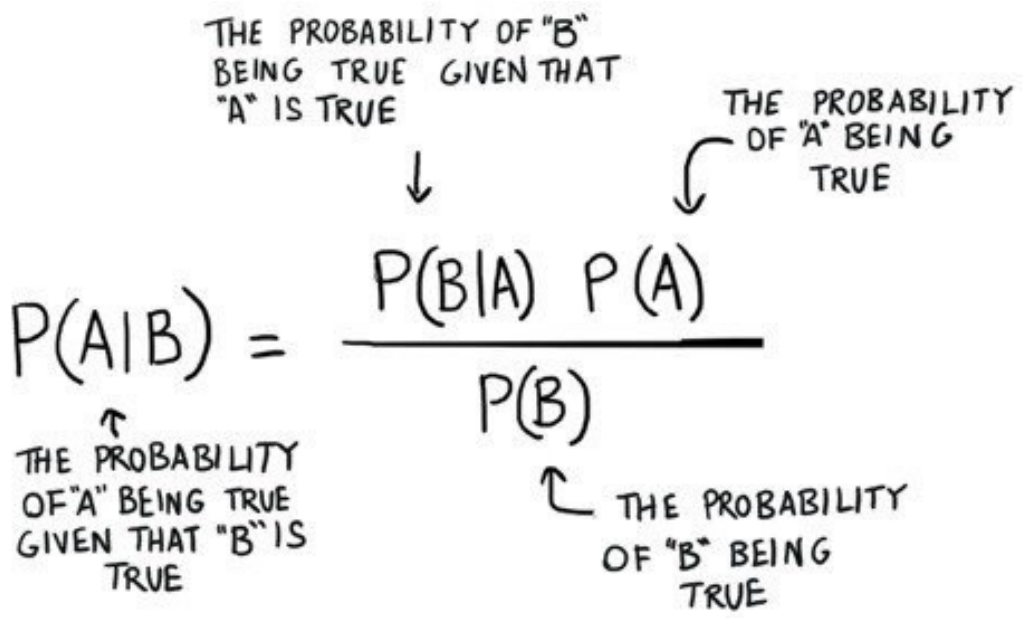


Figura 6: Fórmula matemática do Teorema de Bayes.

Ao utilizar o teorema de Bayes, é possível encontrar a probabilidade de A ocorrer, dado que B ocorreu. Neste contexto, B é a evidência e A é a hipótese. Assume-se que as condições são independentes entre si, ou seja, que a particularidade de uma condição não afeta a outra. Por isso é que o algoritmo chama-se «naive», do inglês: «ingénuo».

Para exemplificar melhor esta teoria, suponha-se o seguinte cenário, retirado e traduzido do sítio [46]:

1% da população desenvolverá a doença VIRUSMAU18. O João vai ao médico verificar se tem a doença VIRUSMAU18. Ao receber o resultado do exame, o João verifica que foi diagnosticado como positivo. O médico diz ao João que 95% das pessoas com a doença testam positivo. Apenas 5% das pessoas que testam positivo ao diagnóstico não têm a doença.

Ao analisar assim o contexto, parece que, infelizmente, o mais certo é o João ter a doença. Contudo, o João, insatisfeito com a situação, e querendo investigar melhor sobre o tema, faz a seguinte análise:

- Se 100 000 pessoas no mundo são testadas, apenas 1 000 devem ter a doença, pois apenas afeta 1% da população;
- Se, dessas 100 000 pessoas, 5% acusam falso-positivo, então ter-se-á um total de 4 950 pessoas com teste positivo sem terem a doença;

- Além disso, os restantes 95% das pessoas que testam positivo e que realmente têm a doença, representam apenas 950 casos, pois os restantes casos (50 casos, 5%) estão contidos no grupo dos falsos-positivos.

Com efeito, dos 100 000 testes, existe um total de 5 900 resultados positivos (4 950 + 950), e apenas 950 destes resultados têm de facto a doença, o que dá apenas cerca de 16% de probabilidade de ter a doença ao ser diagnosticado positivo.

Se se utilizar a fórmula de Bayes temos o seguinte:

P(A): representa a probabilidade de ter a doença (1%)

P(A|B): A probabilidade de ter a doença, dado que o teste foi positivo

P(B|A): A probabilidade de o teste ser positivo, dado que se tem a doença

P(B): representa a probabilidade de testar positivo (5.9%)

A probabilidade de B, isto é, P(B) é obtida através da soma das probabilidades de o teste ser positivo sabendo que se tem a doença mais a probabilidade de o teste ser positivo sabendo que não se tem a doença. Por outras palavras: P(B) = P(B|A) + P(B|$\overline{A}$), que é igual a 95% de 1% da população que testaram positivo e realmente tem a doença, que é 0,95, mais 5% de 95% da população, que representa o número de pessoas que testam positivo sem terem a doença, que é 4,95%.

Utilizando a formula descrita na imagem 6, obtemos os seguintes valores: (0,95) * (0,01) / (0,59) = 0,161... ou seja, cerca de 16%.

Depois do João verificar que é mais provável não ter a doença, mesmo assim decide fazer comprar um outro teste mais rápido que pode ser feito em casa. O João repara que, neste teste, em 85% dos casos das pessoas que têm teste positivo têm de facto a doença, e o teste gera 10% de falsos positivos.

Utilizando a probabilidade anterior de 16% que representa a probabilidade de ter a doença, pretende-se obter certeza se o teste agora voltará a ser positivo. É bom relembrar que ainda há 84% de probabilidade de não ter a doença. Utilizando os novos valores temos que:

P(A): A probabilidade de ter a doença, calculada anteriormente, que é 16%.

P(A|B): A probabilidade de ter a doença dado que o teste deu positivo. O valor que queremos obter.

P(B|A): A probabilidade de o teste dar positivo dado, sendo que tem a doença, o que é igual a 85%.

P(B): A probabilidade do teste ser positivo, que é 85% dos 16% de probabilidade de ter a doença, mais os 10% de falsos positivos dos 84% de chance de não ter a doença.

Aplicando a fórmula, temos:

P(A|B) = (.85 * .16) / ((.85 * .16) + (.10 * .84)) ≈ 0.62

Conclui-se, graças ao cálculo da probabilidade condicional utilizando o Teorema de Bayes, que, embora o segundo teste seja menos preciso, utilizando-o como complemento ao primeiro teste, há uma maior probabilidade de que o João tenha a doença, representando uma probabilidade de 62%.

2.4.2.7 *Lógica Difusa*

A lógica difusa, ao contrário da lógica booleana, é um mecanismo no qual a classificação de verdadeiro ou falso não é representada como sendo uma verdade absoluta. Ao contrário da lógica booleana, o valor um representa o valor verdadeiro e o valor zero representa o valor falso. Na lógica difusa, a representação de valores assume uma escala de valores reais entre zero e um. Neste domínio, um valor pode ser "verdadeiro", isto é, pode ser aceite, ainda que sendo uma verdade parcial, não correspondendo a um valor um, mas próximo deste. Esta vertente tem vindo a ser apreciada pela comunidade científica e tem sido aplicada em diversas áreas na vida real, pois aproxima-se de um conceito mais próximo do pensamento humano, no que respeita a tomada de decisões. Nem sempre as coisas são preto no branco, e por vezes, uma situação, não sendo a perfeita, pode ser a aceitável.

Um caso muito prático será a determinação de alturas das pessoas. Assumindo os conceitos BAIXO e ALTO, tem-se que, na lógica booleana, a pessoa é considerada BAIXO se altura $<$ 1.70m. O indivíduo é considerada ALTO se a sua altura $\geqslant$ 1.70m. Na vida real, muitas vezes estes valores não podem ser considerados apenas como SIM ou NÃO. Existem "meios-termos"que devem ser analisados. Por exemplo, um indivíduo com precisamente 1.70m pode ser considerada por uns como ALTO e por outras pessoas como BAIXO, dependendo da perspectiva de cada um. A definição de BAIXO e ALTO é relativa. No entanto, é do senso comum que uma pessoa com dois metros é alta, e uma pessoa com um metro e meio é baixa. Mas as alturas entre estes dois extremos são subjetivas, como a questão: "O copo está meio vazio ou meio cheio?".

O artigo [30] exemplifica de forma clara a aplicação da lógica difusa na determinação significados de BAIXO, MÉDIO e ALTO consoante a altura de uma pessoa.

Determinadas as etiquetas BAIXO, MÉDIO e ALTO, a determinação de um valor entre zero e um para um determinado indivíduo é calculado através da função de pertença de cada etiqueta, isto é, calcula a pertença que um indivíduo tem a cada *cluster*, em função do distanciamento face ao centro do grupo. Por exemplo, o *cluster* BAIXO tem como centro do grupo o valor de altura inferior ou igual a um metro e cinquenta centímetros e considera um elemento completamente fora deste se a sua altura for superior a um metro e oitenta centímetros.

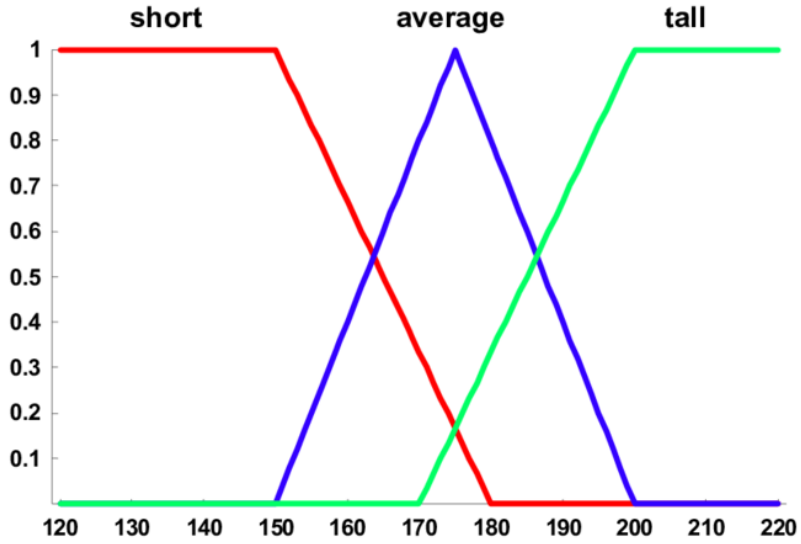


Figura 7: Determinação do significado de BAIXO, MÉDIO e ALTO consoante a altura de uma pessoa

Segundo a figura 7, retirada de [30], O valor "SHORT"representa valor um para pessoas abaixo de um metro e meio, e o valor zero para pessoas com mais de um metro e oitenta. o valor "AVERAGE"representa o valor uma altura de um metro e setenta e cinco e valor zero para a altura inferior a um metro e cinquenta ou superior a dois metros. Por fim, o valor "TALL"assume valor zero para pessoas com altura inferior a um metro e setenta, mas tem valor um para pessoas maiores que dois metros. Neste caso, é possível observar que uma pessoa A com altura = 1.65m tem um valor SHORT $\approx$ 0.5, um valor AVERAGE também $\approx$ 0.5, e um valor para TALL = 0. Na linguagem humana e complexa, é considerada uma pessoa "um pouco mais baixa que o normal, mas não necessariamente baixa, e nunca uma pessoa alta". Uma pessoa B, com uma altura = 1.73m podia perfeitamente (dependendo das funções de pertença para cada valor) obter os seguintes valores: SHORT $\approx$ 0.3,

AVERAGE ≈ 0.9 e TALL ≈ 0.1. Isto significa que a pessoa B é uma pessoa dentro da média da altura populacional, mas dentro dessa mesma média, é considerada mais baixa que o comum, por ter um valor SHORT > TALL.

2.4.2.8 *Outros*

O universo da Aprendizagem Computacional não termina aqui. Existem muitas outras abordagens e teorias. Neste contexto apenas foram evidenciadas as mais importantes e famosas até à data.

2.4.3 *Comités*

Comités, em inglês conhecidos como *ensembles* ou *mixture of experts*, é a combinação de vários classificadores. Esta metodologia tem uma importância especial para o tema. Um dos objetivos é pretender aumentar a performance da classificação. Utilizando comités, será possível alcançar este objetivo.

O conceito assenta na ideia de que dois ou mais indivíduos tomam a melhor decisão, ou a decisão correta, quando comparado com apenas uma opinião. Por exemplo, nos eventos de desporto é comum encontrar mais do que um juiz de prova, para garantir que são tomadas as decisões corretas. No momento de uma decisão, um comité é sempre preferível, pois ajuda a reforçar probabilisticamente a solução correta.

2.4.3.1 *Vantagens*

Utilizar um comité traz mais benefícios que a utilização de apenas um classificador, mesmo nos casos em que o classificador seja forte e os vários classificadores do comité sejam fracos. Através dos vários métodos que irão ser explicados adiante, pode-se fortalecer o desempenho dos classificadores fracos, por forma a obter um desempenho superior em relação ao classificador único.

Para além disto, ainda são identificadas as vantagens enumeradas no livro [61]:

1. Estatística
2. Grandes volumes de dados
3. Poucos dados
4. Dividir-para-reinar

5. Fusão de dados

O item 1 refere-se ao aumento do desempenho na capacidade de generalização. Classificadores diferentes dão origem a modelos distintos. Deste modo, no momento de teste, reduz-se a probabilidade de obter um mau desempenho.

O item 2 relaciona-se com os métodos para criar comités. Quando existe um grande volume de dados, é comum separar os dados em subconjuntos de testes. Cada subconjunto é extraído recorrendo a uma métrica discutida abaixo. Cada subconjunto pode ser treinado com diferentes classificadores.

No caso do item, 3, pode-se utilizar métricas para reamostragem de dados, originando vários subconjuntos. Cada subconjunto serve de treino a um classificador. Dada a dimensão pequena dos dados, o subconjunto pode ser criado a partir de dados repetidos ou derivados dos originais.

Quando se tem um problema muito grande, é possível particionar a complexidade do problema por vários classificadores. Existem técnicas para combinação de resultados. Assim, o item 4 é uma clara vantagem na utilização desta metodologia.

Por fim, os comités também auxiliam o caso do item 5. Existem contextos onde os dados podem ser provenientes de diferentes fontes, representar tipos distintos ou ter uma quantidade de dados díspar. Nestes casos, um classificador, por exemplo, pode tratar de uma fonte de dados. No final, os resultados são fundido num único *output*, recorrendo a técnicas de combinação.

2.4.4 *Tipos de comités*

Podemos dividir os comités em dois tipos:

- Homogénios
- Heterogénios

O primeiro 2.4.4 combina vários classificadores recorrendo a métodos com o mesmo tipo de aprendizagem. O segundo 2.4.4 combina vários classificadores de diferentes tipos de aprendizagem.

2.4.5 *Métodos para criar comités e metodologias de combinação*

Existem vários métodos para criar comités [16, 48, 61, 62, 71], e várias metodologias de combinação [28, 60], que irão ser apenas mencionados, de forma sucinta, para que o trabalho relacionado 2 não desvia ainda mais do escopo do tema do relatório. Os métodos para criar comités são o *bagging* , o *boosting*, os *random forests* e o *stacking*. As metodologias de combinação podem ser: Votação por maioria, Votação por maioria com peso (influência) e métodos que combinam resultados contínuos, como por exemplo, combinação por *ranking* de resultados.

3

PLANEAMENTO

Abordar-se-á, no presente capítulo, o planeamento que conduziu o estágio e as tecnologias utilizadas, bem como as decisões tomadas e o motivo das mesmas.

3.1 METODOLOGIAS UTILIZADAS

Por forma a agilizar o processo de desenvolvimento de software, a metodologia escolhida teve inspiração no Scrum [56]. Também utilizámos as *issues* da plataforma `https://gogs.io/`, para coordenar trabalho por fazer, deteção e correção de bugs.

3.1.1 *Scrum*

A presente secção irá abordar a definição e o objetivo do *Scrum* através do guia oficial do *Scrum* [57]. Deste modo, o texto aqui escrito está fortemente baseado e traduzido do texto original do documento anteriormente mencionado e referenciado.

3.1.1.1 *O que é e para que serve*

A definição de *Scrum*[59] é a seguinte:

> *"A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value."*
>
> *[57]*

Scrum é ainda considerado:

1. Leve
2. Simples de compreender
3. Difícil de dominar

A seguinte figura 8, retirada de [49], demonstra o quadro geral da *framework Scrum*:

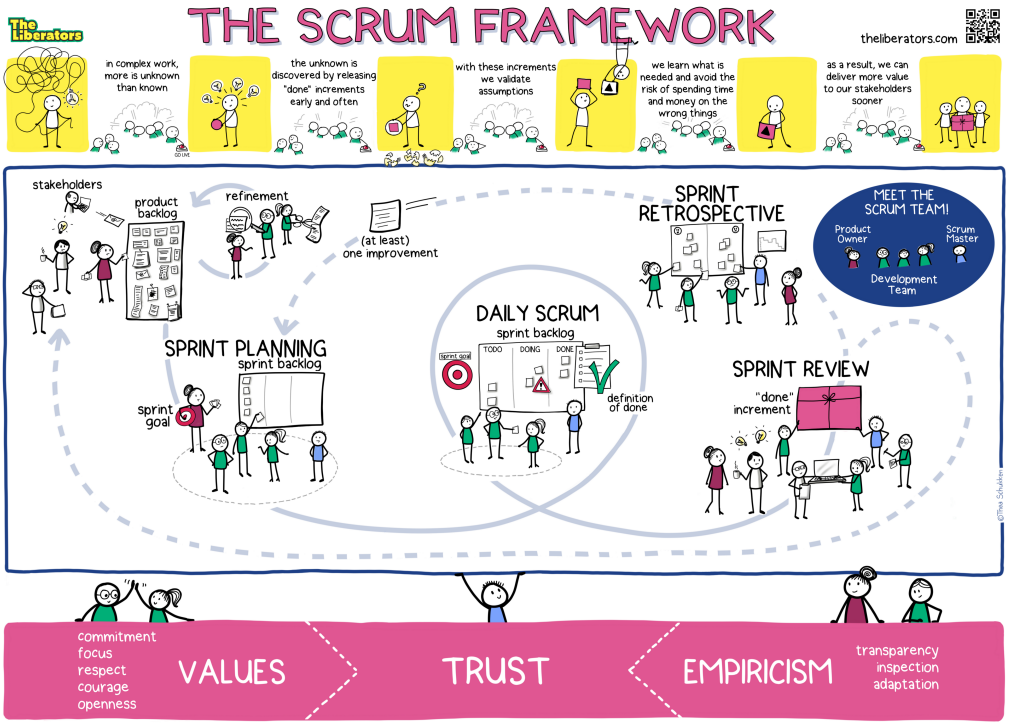


Figura 8: Framework Scrum

Scrum é uma *framework* que pretende facilitar processos complexos, focando-se nas pessoas e não no processo [6, 12, 20], com o propósito de, através da boa gestão e interação entre as pessoas e/ou partes interessadas, conseguir cumprir com os objetivos pretendidos. O *Scrum* não é um processo, uma técnica ou metodologia definida. Em vez disso, é uma *framework* que pretende utilizar vários processos, técnicas ou metodologias, para entregar o produto de forma mais eficiente.

3.1.1.2 *Pilares do Scrum*

O *Scrum* é fundada sobre a teoria empírica de controlo de processos ou empirismo e tem como pilares a transparência, a inspeção e a adaptação.

A transparência no *Scrum* define que deve existir uma linguagem comum referente ao processo, utilizada por todos os participantes. Os que executam o trabalho e aqueles que inspecionam o resultado incrementante devem partilhar a mesma definição comum de "feito".

Na framework *Scrum*, a inspeção é conduzida através do *Scrum Goal*, que pretende avaliar os artefactos, para detetar variâncias indesejadas. A inspeção não deve ser

frequente, por forma a não seja dada maior importância a esta do que ao trabalho, obstruíndo-o.

Por fim, a adaptação determina que qualquer aspeto que desvie o processo do objetivo desejável, deve ser adaptado, quer seja o produto final ou o processo *per se*. Este último ponto evidencia a agilidade do *Scrum*. Esta *framework* é considerada uma das metodologias ágeis de desenvolvimento de software [2, 56] e é provavelmente a mais conhecida e utilizada no mercado.

No *Scrum* existem perfis, eventos, artefactos e regras. Falar-se-á de cada um em específico.

A inspeção e adaptação acontecem em eventos específicos do ciclo *Scrum*, que são: *Sprint Planning*, *Daily Scrum*, *Sprint Review* e *Sprint Retrospective*. Estes eventos irão ser discutidos em pormenor mais adiante.

3.1.1.3 *Valores do Scrum*

Os valores do *Scrum* são o compromisso, a coragem, o foco, a abertura e o respeito, tanto entre a equipa como entre o inter-diálogo com as partes interessadas. A abertura e a transparência é fundamental para um negócio que dê sucesso. Quando os valores são respeitados por todos, é possível alcançar os pilares do *Scrum* 3.1.1.2, criando confiança entre todos.

3.1.1.4 *A Equipa Scrum*

A equipa *Scrum* é constituída por: *Product Owner* 3.1.1.4, a equipa de desenvolvimento 3.1.1.4 e o *Scrum Master* 3.1.1.4. As equipas devem ser auto-organizadas e funcionais entre si. As equipas devem escolher a melhor forma de conseguir os seus objetivos, em vez de serem dirigidas por outros fora da equipa. As equipas funcionais entre si devem conseguir alcançar o seu trabalho sem depender de outros que não pertençam à sua equipa.

PRODUCT OWNER É a pessoa responsável por gerir o *Product Backlog*. Deve: exprimir claramente os itens do *Product Backlog*, ordenar os itens por ordem de importância, optimizar o valor do trabalho da equipa de desenvolvimento 3.1.1.4, garantir que o *Product Backlog* é visível, transparente e claro para totós, mostrar à equipa 3.1.1.4 qual a próxima ordem de trabalhos e ainda garantir que a equipa de desenvolvimento 3.1.1.4 compreende todos os itens até ao nível de detalhe necessário para o seu trabalho.

O *Product Owner* é uma pessoa, nunca um comité, mas pode representar os desejos de um comité no *Product Backlog*. É responsável maximizar o valor do trabalho feito pela equipa de desenvolvimento 3.1.1.4.

Por fim, é fundamental que toda a organização respeite as suas decisões, por forma a que este possa suceder.

EQUIPA DE DESENVOLVIMENTO A equipa de desenvolvimento, como o nome indica, é a equipa responsável pelo desenvolvimento de todos os "incrementos"que foram pré-selecionados no início da *Sprint*, ou seja, os "incrementos"que estão no *Sprint Backlog*. Esta equipa produz constantemente o produto, num conceito de "Feito"no final de cada *Sprint*, de forma incremental. O incremento de "Feito"é mandatório na *Sprint Review*.

SCRUM MASTER O *Scrum Master* é o elemento responsável por implementar de forma eficiente a *framework* Scrum, tal como definida no *Scrum Guide*. O *Scrum Master* deve conseguir fazer com que a teoria seja praticada por todos os elementos, de forma a que todos a compreendam e aceitem de uma forma ativa e positiva.

3.1.1.5 *Eventos do Scrum*

Uma *Sprint* é composta por vários eventos. Cada evento é uma oportunidade para introspecção e adaptação dos artefactos do Scrum. Tais eventos foram desenhados para garantir a transparência de todo o processo.

SPRINT A *Sprint* é o «*heartbeat*» do Scrum, onde as ideias são transformadas em valor. Para criar consistência, uma *Sprint* tem sempre um tamanho fixo, tipicamente de um mês ou menos. Uma *Sprint* começa assim que a anterior *Sprint* acaba.

SPRINT PLANNING O *Sprint Planning* é o que dá início à *Sprint*, definindo o trabalho que irá ser desenvolvido durante a mesma. O plano resultante é criado em colaboração com toda a equipa.

O *Product Owner* é o responsável por garantir que todos os participantes compreendem claramente os itens no *Product Backlog* e a sua importância no objetivo final.

Tipicamente, o *Sprint Planning* é conduzido através dos seguintes tópicos:

- Porque é que esta *Sprint* é importante;
- O que é que pode ser feito nesta *Sprint*;
- Como é que os itens escolhidos para desenvolvimento irão ser feitos.

DAILY SCRUM O *Daily Scrum* é um evento, que deve demorar, no máximo, 15 minutos e permite sincronizar as atividades e definir um plano para as próximas 24 horas. Por isso, é um evento que ocorre todos os dias, normalmente sempre à mesma hora. Otimiza a colaboração e a performance da equipa, através da análise do *Daily Scrum* anterior e prevendo o trabalho do dia que se segue.

Pretende-se que cada membro da equipa participe e aborde as questões:

- O que é que eu fiz ontem?
- O que farei hoje?
- Que obstáculos existem para a realização do meus objetivos?

SPRINT REVIEW A *Sprint Review* é o penúltimo evento da *Sprint*, e serve para apresentar, pela equipa, às partes interessadas, os resultados do trabalho feito durante a *Sprint* e como estão mais próximos do objetivo final do produto. É cronometrado para durar, no máximo, 4 horas, para uma *Sprint* de um mês. Para *Sprints* mais pequenas, deve demorar proporcionalmente menos.

SPRINT RETROSPECTIVE Por último, a *Sprint Retrospective* tem como objetivo planear maneiras de melhorar a qualidade e a efetividade. A equipa deve inspecionar como correu a última *Sprint* e aprender com os pontos positivos e negativos que vão surgindo. Devem expor estes pontos a toda a equipa para que, em conjunto, a equipa defina passos concretos para combater os pontos negativos e aperfeiçoar os pontos positivos. Este evento tem um tempo máximo de 3 horas para *Sprints* mensais. Para *Sprints* de períodos temporais menores, deve durar menos tempo.

Uma boa *Sprint Retrospective* segue os seguintes tópicos:

- O que correu bem durante a *Sprint*?
- O que é que pode ser melhorado?
- O que é a equipa se compromete a melhorar na próxima *Sprint*?

3.1.1.6 *Artefactos do Scrum*

Os artefactos do *Scrum* são 3: O *Product Backlog*, o *Sprint Backlog* e o Incremento.

PRODUCT BACKLOG Trata-se de uma lista de itens de unidades de trabalho, conforme a figura 9, retirada de [68], de tudo o que há a desenvolver no produto, para cumprir com o objetivo final do produto. O *Product Backlog* é dinâmico, nunca está completo, sofre constantes mudanças, assim que o projeto avança e se vai refinando pormenores e descobrindo novas funcionalidades ou adicionando correções de bugs.

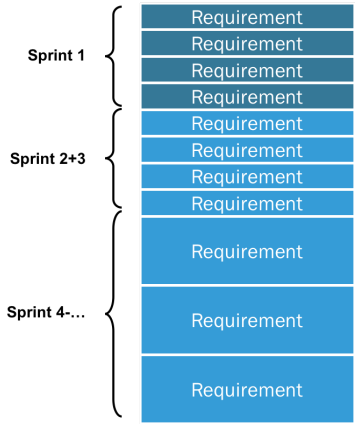


Figura 9: *Scrum* / *Product Backlog*

SPRINT BACKLOG A *Sprint Backlog* é um subconjunto selecionado a partir do *Product Backlog*. É o conjunto de itens a desenvolver ao longo de uma *Sprint*. Como exemplificado na figura 10, retirada de [69], a *Sprint Backlog* pode ser definida por um quadro kanban 3.1.2 com a definição da funcionalidade a desenvolver na primeira coluna, e os marcos ou etapas nas restantes colunas, onde estão os sub-itens que compõem a funcionalidade.

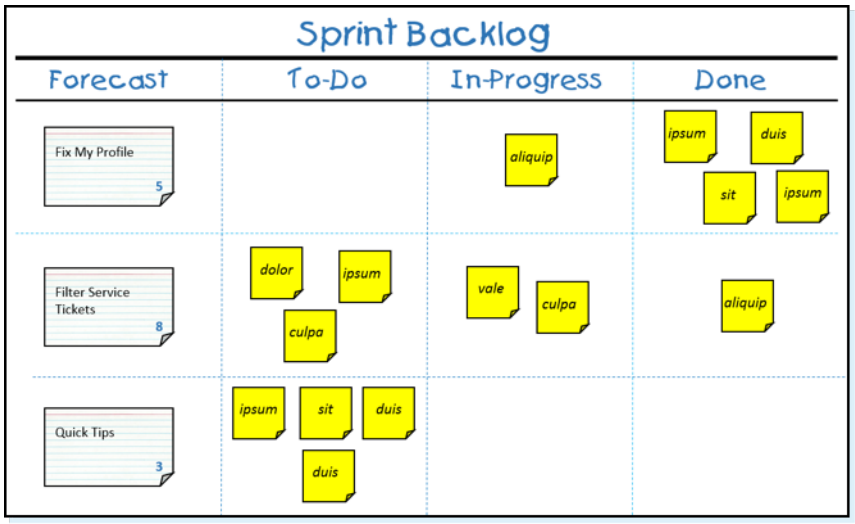


Figura 10: *Scrum* / *Sprint Backlog*

INCREMENTO Um incremento é um marco específico em direção ao objetivo do produto. Cada incremento é "adicionado"a todos os incrementos anteriores e verificado, garantindo que todos os incrementos funcionam juntos. Para que o incremento tenha valor, o incremento deve ser utilizável.

Vários incrementos podem ser criados numa só *Sprint*. A soma dos incrementos é apresentada na *Sprint Review*. No entanto, um incremento pode ser entregue às partes interessadas antes do final da *Sprint*.

3.1.1.7 *Aplicação da metodologia no trabalho*

Dada a dinâmica da empresa e do projeto, não foi sentida a necessidade de aplicar a metodologia *Scrum* na sua totalidade, porque o projeto contava, à data, com apenas dois colaboradores no desenvolvimento. O Estudante Daniel Mendes Pinto era o "responsável"do projeto, na medida em que era a este a quem se pediam esclarecimentos do estado de desenvolvimento e era quem esclarecia os próximos itens a serem desenvolvidos. Era o *Scrum Master*. Para além disso, tinha a seu encargo as tarefas da desenvolvimento relativas ao *backend*. Para o *frontend*, contou-se com o trabalho de Diogo Vinagre Simão, colaborador na mesma empresa. O *Product Owner* do projeto é o professor Carlos Antunes, empregador da empresa Icingslice, Lda.

Não se seguiu a *framework* Scrum na sua totalidade, visto que não é aconselhado que o *Product Owner* seja o empregador, embora não haja nada contra no guia oficial. Para além disso, o *Scrum Master* por vezes era quem especificava os itens do

Product Backlog, dividindo-os em subitens mais compreensíveis em termos técnicos, para cumprir a definição de "feito".

3.1.2 *Kanban*

O *Kanban*, termo do japonês que significa «cartão», é uma metodologia de gestão de trabalho, criado pela Toyota, no final da década de 1940. Esta metodologia foi originalmente criada para melhorar os fluxos de produção nas fábricas de produção. A ideia da empresa era introduzir a fabricação «*just in time*», ou seja, produzir apenas o que for necessário, conforme se vão puxando novas tarefas. Esta metodologia é um sistema que funciona por puxar as tarefas a realizar, em vez da abordagem tradicional de empurrar as tarefas para a produção, gerando sobrecarga. [19, 32].

3.1.2.1 *Como funciona*

O *Kanban* é representado por um quadro que contém as tarefas que a equipa deve completar, conforme se pode verificar pela figura 11, retirada de [32]. Este quadro torna o processo mais ágil porque é visualmente mais motivante (ao completar uma tarefa, arrastando-a para o lado direito, dá o sentido de «dever cumprido» e motiva toda a equipa, tornando-a mais produtiva) e ainda permite simplificar o trabalho, atribuindo tarefas específicas para produção. Numa empresa onde não haja esta gestão ou uma metodologia para gestão de produção, as tarefas vão sendo empurradas e empilhadas para cima da equipa de produção que, de repente, vê-se com imensas tarefas por realizar e em simultâneo. Para além de frustrante ter de lidar com tantas coisas em simultâneo, é mais propenso a esquecer-se de algumas tarefas, ou perder a meio uma tarefa já a ser realizada, ou simplesmente concluir uma tarefa sem ser da melhor maneira, quase num "desenrasca", porque o indivíduo que a realiza está focado em múltiplas coisas ao mesmo tempo, não conseguindo ser verdadeiramente bom na tarefa que devia ter concluído.

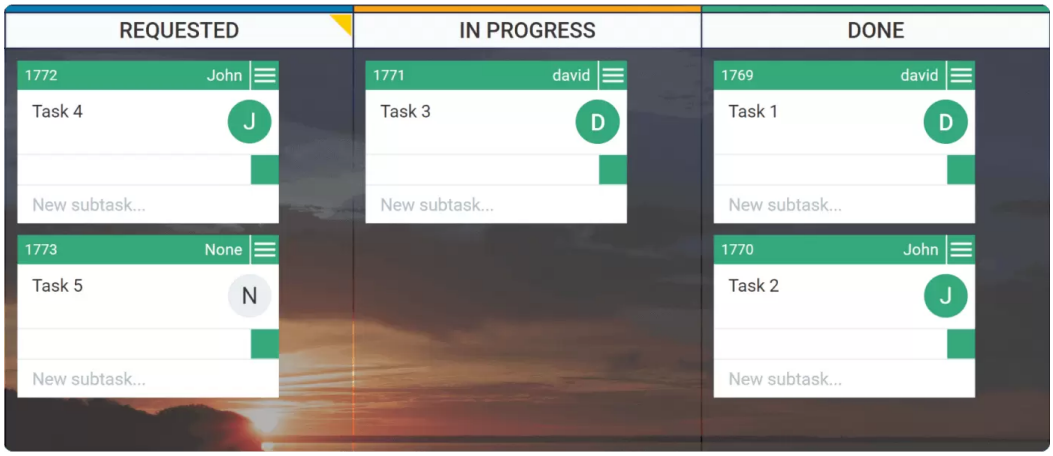


Figura 11: Quadro *Kanban* simples, para exemplificação

Na figura 11, podem-se verificar três etapas principais onde estão distribuídas várias tarefas: O que está por fazer («*requested*» ou «*todo*»), as tarefas que estão a ser desenvolvidas («*in progress*» ou «*doing*») e, por fim, as tarefas já realizadas («*completed*» ou «*done*»).

Aplicando o *Kanban* ao desenvolvimento de *Software*, e integrando com a metodologia ágil *Scrum*, o *Kanban* é aplicado no *Sprint Backlog*, que representa a lista de tarefas a realizar durante uma *Sprint* (ver figura 10). É comum existirem ainda mais marcos no *Kanban*, para além do «*todo*», «*doing*» e o «*done*», principalmente numa empresa mais complexa que envolve vários procedimentos e na Engenharia de Software. Por exemplo, pode ser importante criar uma tarefa entre o«*doing*» e o «*done*», chamada «*ready*» que é o momento em que a tarefa foi concluída pela equipa de desenvolvimento (e, com efeito, está "pronta"), mas ainda tem de ser testada pela equipa de testes, pelo que não é considerado ainda "feita", no conceito de «*Definition of Done*» da metodologia *Scrum*.

3.2 LINGUAGENS, TECNOLOGIAS E BIBLIOTECAS UTILIZADAS

Nesta secção falar-se-ão das linguagens de programação, tecnologias e bibliotecas utilizadas ao logo do desenvolvimento do produto IBC. Visto que a *framework* é dividida em duas componentes distintas, o servidor API e o painel administrativo em *Nuxt*, será este tema será dividido em dois, o *backend*3.2.1 e o *frontend*3.2.2.

3.2.1 *Backend*

Nesta secção será feita uma comparação das linguagens candidatas para utilização no projeto, da parte do *backend*, e as principais bibliotecas e tecnologias utilizadas.

3.2.1.1 *Linguagem de Programação*

No entanto, no início do projeto, decidiu-se fazer um estudo comparativo no mercado, antes de eleger a linguagem adequada ao projeto.

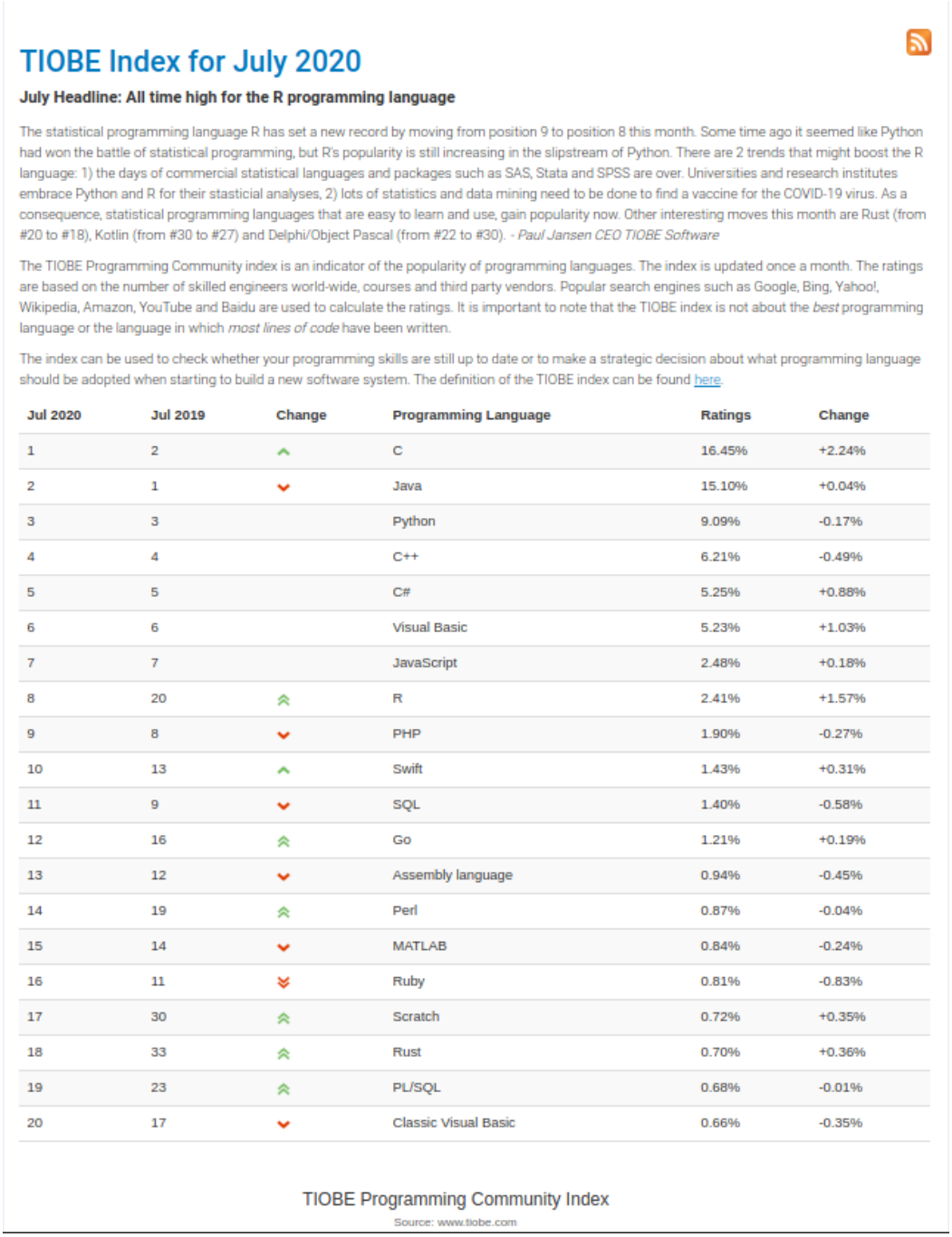
Primeiramente, tinha de ser uma linguagem: orientada à web, fácil compreensão, intuitiva e popular.

O PopularitY of Programming Language (PYPL) (*site*: `http://pypl.github.io/PYPL.html`) e a TIOBE (*site*: `https://www.tiobe.com/tiobe-index/`) são bons indicadores para escolher uma linguagem com base na popularidade, e são bastante conhecidos pela comunidade de programadores. À data do presente documento, as classificações são:

Worldwide, Jul 2020 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	31.73 %	+3.9 %
2		Java	17.13 %	-2.7 %
3		Javascript	7.98 %	-0.3 %
4		C#	6.67 %	-0.6 %
5	↑	C/C++	5.93 %	+0.1 %
6	↓	PHP	5.64 %	-1.1 %
7		R	4.14 %	+0.3 %
8		Objective-C	2.61 %	-0.1 %
9		Swift	2.29 %	-0.1 %
10	↑	TypeScript	1.91 %	+0.2 %
11	↓	Matlab	1.74 %	-0.1 %
12		Kotlin	1.62 %	+0.2 %
13	↑↑	Go	1.37 %	+0.2 %
14		VBA	1.27 %	-0.0 %
15	↓↓	Ruby	1.26 %	-0.1 %
16		Scala	0.99 %	-0.1 %
17		Visual Basic	0.83 %	-0.2 %
18	↑	Rust	0.81 %	+0.3 %
19	↑↑↑↑↑	Dart	0.52 %	+0.2 %
20	↑↑↑	Ada	0.47 %	+0.1 %
21	↑	Lua	0.46 %	+0.1 %
22	↓↓	Abap	0.46 %	-0.1 %
23	↓↓	Groovy	0.43 %	-0.1 %
24	↓↓↓↓↓↓	Perl	0.42 %	-0.2 %
25		Cobol	0.41 %	+0.1 %
26	↑↑	Julia	0.37 %	+0.1 %
27	↓	Haskell	0.29 %	+0.0 %
28	↓	Delphi	0.25 %	-0.0 %

Figura 12: Classificação das Linguagens de Programação segundo o índice de PYPL



TIOBE Index for July 2020

July Headline: All time high for the R programming language

The statistical programming language R has set a new record by moving from position 9 to position 8 this month. Some time ago it seemed like Python had won the battle of statistical programming, but R's popularity is still increasing in the slipstream of Python. There are 2 trends that might boost the R language: 1) the days of commercial statistical languages and packages such as SAS, Stata and SPSS are over. Universities and research institutes embrace Python and R for their stasticial analyses, 2) lots of statistics and data mining need to be done to find a vaccine for the COVID-19 virus. As a consequence, statistical programming languages that are easy to learn and use, gain popularity now. Other interesting moves this month are Rust (from #20 to #18), Kotlin (from #30 to #27) and Delphi/Object Pascal (from #22 to #30). - *Paul Jansen CEO TIOBE Software*

The TIOBE Programming Community index is an indicator of the popularity of programming languages. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings. It is important to note that the TIOBE index is not about the *best* programming language or the language in which *most lines of code* have been written.

The index can be used to check whether your programming skills are still up to date or to make a strategic decision about what programming language should be adopted when starting to build a new software system. The definition of the TIOBE index can be found here.

Jul 2020	Jul 2019	Change	Programming Language	Ratings	Change
1	2	↑	C	16.45%	+2.24%
2	1	↓	Java	15.10%	+0.04%
3	3		Python	9.09%	-0.17%
4	4		C++	6.21%	-0.49%
5	5		C#	5.25%	+0.88%
6	6		Visual Basic	5.23%	+1.03%
7	7		JavaScript	2.48%	+0.18%
8	20	⇈	R	2.41%	+1.57%
9	8	↓	PHP	1.90%	-0.27%
10	13	↑	Swift	1.43%	+0.31%
11	9	↓	SQL	1.40%	-0.58%
12	16	⇈	Go	1.21%	+0.19%
13	12	↓	Assembly language	0.94%	-0.45%
14	19	⇈	Perl	0.87%	-0.04%
15	14	↓	MATLAB	0.84%	-0.24%
16	11	⇊	Ruby	0.81%	-0.83%
17	30	⇈	Scratch	0.72%	+0.35%
18	33	⇈	Rust	0.70%	+0.36%
19	23	⇈	PL/SQL	0.68%	-0.01%
20	17	↓	Classic Visual Basic	0.66%	-0.35%

TIOBE Programming Community Index

Source: www.tiobe.com

Figura 13: Classificação das Linguagens de Programação segundo o índice da empresa TIOBE

É de notar que estes indicadores são bastante diferentes. A figura 12, retirada de `http://pypl.github.io/PYPL.html`, mostra uma realidade oposta à da figura 13, retirada de `https://www.tiobe.com/tiobe-index/`. Evidentemente, surge a questão: Como podem os dois indicadores mais comuns revelarem dados tão diferentes? A resposta está no seguinte:

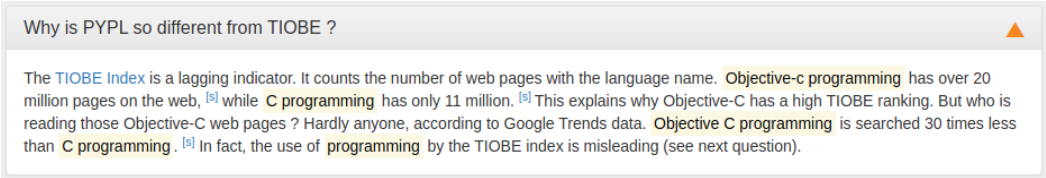


Figura 14: PYPL *vs.* TIOBE

Pela imagem 14, retirada de `http://pypl.github.io/PYPL.html`, na zona das questões frequentes, a resposta evidencia que o PYPL é, desta forma, bem mais fidedigno. É por esta classificação 12 que vão ser eleitas como candidatas as seguintes linguagens de programação (*Top* 5):

1. Python
2. Java
3. Javascript
4. C#
5. ~~C/C++~~ (não é uma linguagem vocacionada para a web)
6. PHP

Visto que C/C++ não é uma linguagem que seja famosa no desenvolvimento de aplicações web, foi automaticamente excluída. Assim, inclui-se a linguagem no 6º lugar do patamar, PHP.

Das quatro linguagens restantes, segue-se a tabela de prós e contras:

LINGUAGEM	PRÓS	CONTRAS
Python	Linguagem dinâmica	Lenta
	Linguagem interpretada	
	Linguagem de alto nível	
	Fácil de aprender	
	Multiplataforma	
	Funcional e orientada a objetos	
	Bastantes bibliotecas e recursos	
	Muito utilizada no ambiente académico	
	Muito utilizada em Inteligência Artificial e *Data Mining*	

Java	Linguagem de alto nível	Requer licença da Oracle (para lançamento da aplicação no mercado)
	Multiplataforma	Compila para *byte-code*
	Orientada a objetos	Estática
	Bastantes bibliotecas e recursos	
	Muito utilizada no ambiente académico	
	Muito utilizada em Inteligência Artificial e *Data Mining*	
Javascript	Linguagem dinâmica	Pode ser difícil de manter (tendência de fazer *spaghetti code*
	Multiplataforma	
	Linguagem de alto nível	
	Linguagem interpretada	
	Funcional e orientada a objetos	
	Bastantes bibliotecas e recursos	
C#	Linguagem de alto nível	Linguagem associada à Microsoft
	Multiplataforma	Compila para *byte-code*
	Orientada a objetos	Estática
	Bastantes bibliotecas e recursos	
PHP	Linguagem dinâmica	Lenta
	Linguagem de alto nível	
	Multiplataforma	
	Funcional e Orientada a objetos	
	Bastantes bibliotecas e recursos	

Tabela 1: Comparação entre as linguagens Python, Java, Javascript e C#

Nota

Dependendo do contexto ou do objetivo pretendido, algumas vantagens e desvantagens podem estar mal classificadas. Por exemplo: uma linguagem estática e compilada pode ser uma vantagem, por ser mais rápida.
No entanto, visto que se pretende uma linguagem para o desenvolvimento *web*, é mais importante a versatilidade da linguagem do que propriamente a velocidade de execução, ao contrário do que acontece no desenvolvimento jogos, por exemplo. Para esse caso, estaríamos a discutir, por exemplo, as vantagens da utilização da linguagem C/C++.
Com efeito, as vantagens e desvantagens descritas são por vezes subjetivas e dependem também, em parte, do gosto pessoal dos programadores e da empresa.

Tendo em conta as vantagens e desvantagens acima mencionadas, existem pelo menos duas linguagens que pretendemos excluir logo de início: Java e C#. O motivo é o seguinte: pretende-se, para o mestrado, uma linguagem *open-source*, independente de uma empresa, ou cuja utilização requer uma licença para lançar no mercado.

Das linguagens restantes, Python, Javascript e PHP são ótimas candidatas. No entanto, das três linguagens, Python é sem dúvida muito superior se for pretendido, no futuro, utilizar Inteligência Artificial e *Data Mining*. Já existem muitas bibliotecas em Python para o efeito, bastante populares. Javascript, por sua vez, embora, regra geral, tenha maior performance [37], não é a melhor candidata, dada a desvantagem indicada na tabela 1. PHP é sem dúvida a linguagem mais conhecida para o desenvolvimento *web*, mas não é uma linguagem indicada se a aplicação precisar de bastante processamento.

Data a popularidade de Python no ambiente académico, à facilidade de aprendizagem, à sua versatilidade e ao facto de possuir bibliotecas (3.2.1.3) bem implementadas para as tecnologias (3.2.1.2) descritas, optou-se por Python.

É importante referir que, no início do projeto, começou-se por desenvolver em paralelo uma pequena implementação do módulo de entidades 4.5.7 em Laravel, uma *framework web* na linguagem PHP. No entanto, devido às limitações da bibliotecas em PHP para integração dos modelo de dados com a base de dados Neo4j, o projeto foi abandonado rapidamente.

3.2.1.2 *Tecnologias*

No que toca à "melhor forma"de guardar os dados, o pensamento foi evoluindo consoante as necessidades e face ao tipo de dados. As primeiras bases de dados suficientemente robustas, surgiram com o conceito do Sistema R [67], o primeiro produto desenvolvido que utilizava a linguagem SQL. Este produto fora desenvolvido pela IBM e daqui nasceu o conceito das bases de dados relacionais, bastante sólido e seguro até aos dias de hoje. [8].

Contudo, com o advento do *Big Data*, com a crescente complexidade e sobretudo dada a dinâmica da forma que os dados podiam estar estruturados, as bases de dados deixam de conseguir responder as necessidades, visto que a estrutura de dados torna-se mais flexível, o que não acontece numa base de dados relacional, onde as tabelas e as colunas são previamente muito bem definidas [45]. Nascem, assim, as tecnologias de base de dados mais recentes, englobadas aqui genericamente como «*NoSQL*» [23], isto é, as bases de dados que fogem do padrão das bases de dados relacionais e, por isso, são mais flexíveis e não utilizam a linguagem SQL.

Após uma breve análise das necessidades a que o produto tinha de corresponder, e analisando os diferentes prós e contras de cada tecnologia [4, 23, 26, 45], foi decidido que a melhor abordagem passaria por uma base de dados gráfica.

Atualmente, existem muitas bases de dados gráficas no mercado [3, 31]. Optou-se pela Neo4j por ser o primeiro produto a implementar uma base de dados gráfica. É, deste modo, bastante madura e robusta, e é das bases de dados mais utilizadas atualmente no mercado. A comunidade Neo4j é numerosa e existem muitas bibliotecas implementadas que facilitam o uso desta ferramenta, pelo que é uma vantagem bastante favorável a ter em conta num projeto de grande dimensão: a estabilidade e o suporte das tecnologias utilizadas são dois pilares importantes. O Neo4j utiliza a Cypher Query Language (CQL)[47], a linguagem para fazer pesquisas em base de dados.

3.2.1.3 *Bibliotecas*

As principais bibliotecas utilizadas para o projeto são:

- django_neomodel [18].
- *Django Rest Framework*
- *libpff* [42]

- *mail-parser* [40]

A biblioteca *django_neomodel* é um *plugin* de integração em Django para a biblioteca *neomodel* [66]. Por sua vez, a biblioteca *neomodel* é uma biblioteca que permite integrar classes em Python como modelos que representam nós em Neo4j, utilizando o padrão de Engenharia de Software *Active Record*[44], um dos padrões Object-relational mapping (ORM)[34], um padrão que permite mapear os dados em base de dados para uma instância (objeto) de uma classe. Existem outras bibliotecas para além da *neomodel* para utilização do padrão ORM no projeto. No entanto, esta pareceu a mais favorável, tendo em conta a estabilidade, a comunidade que a desenvolve, a simplicidade e a boa integração com o Django através do *plugin* mencionado.

A biblioteca *Django Rest Framework* é uma biblioteca essencial para criar uma *api* seguindo as convenções REST [17, 35], para a *framework* Django. Não há bibliotecas tão bem implementadas para este efeito como esta.

As bibliotecas *libpff* e *mail-parser* são bibliotecas externas para poder processar e-mails. Estas bibliotecas são utilizadas no projeto porque os protocolos dos ficheiros de e-mail da Microsoft são desconhecidos, pelo que é necessário bibliotecas externas que forneçam este apoio.

Existem muitas outras bibliotecas utilizadas, como se pode verificar pelo apêndice A. No entanto, fugiria do âmbito do presente relatório mencionar cada uma delas em específico. Para além disso, mais de metade das dependências nos requisitos advém de bibliotecas que são adicionadas pelo programador, e estas dependências, ao serem adicionadas, importam outras dependências, sendo, portanto, dependências indiretas.

3.2.2 *Frontend*

Aqui discutir-se-á a razão da escolha do Javascript como linguagem de programação para o *frontend* e das tecnologias utilizadas.

3.2.2.1 *Linguagem de Programação*

Ao contrário das várias possibilidades de linguagens para o *backend* 3.2.1.1, no caso do *frontend*, a linguagem que impera ainda nos dias de hoje, e a qual poucas lhe fazem frente para o desenvolvimento *web*, é Javascript. Sem dúvida que se tornou a

linguagem mais utilizada para a *web* e só recentemente estão a surgir, aos poucos, novas linguagens (que inclusive prometem mais performance), mas que ainda hoje são pouco aceites ou utilizadas no mercado. Para além disso, não se pode comparar a enormidade que é a comunidade de programadores em Javascript e as imensas bibliotecas nesta linguagem, bem como muitas soluções já respondidas por tantos utilizadores na internet para os mais variados cenários. E como se trata de um projeto empresarial e não tão focado na investigação, não é conveniente desenvolver em linguagens recentes, onde não existe tanto suporte. Portanto, resta apenas uma opção: Javascript.

3.2.2.2 *Tecnologias*

A tecnologia, ou *framework*, escolhida foi o NUXT[11], que foi escrita com base no Vue.js[70] (ver: 3.2.2.3). Neste ponto, também não houve outra opção. Esta é a única *framework* suficientemente estável, robusta e famosa que integra com o Vue.js.

Para acelerar o processo de desenvolvimento da User Interface (UI), a empresa que desenvolve o produto IBC optou por comprar um *template* que já vem com uma base bastante sólida de componentes visuais prontos a reutilizar. O *template* foi desenvolvido pela *Creative Tim* (ver: `https://www.creative-tim.com/`) e chama-se *NUXT Argon Dashboard Pro* (ver: `https://www.creative-tim.com/product/nuxt-argon-dashboard-pro`).

3.2.2.3 *Bibliotecas*

Atualmente, as quatro bibliotecas mais utilizadas para facilitar o desenvolvimento do *frontend*, no que toca à manipulação do Document Object Model (DOM) são:

- jQuery
- Angular
- React
- Vue.js

jQuery foi uma biblioteca que há muito tempo era a melhor e mais conhecida ferramenta para manipular o DOM. No entanto, tem um grande defeito: é pesada. Com as novas bibliotecas que foram surgindo no mercado, que são bem mais leves, está, por este motivo, excluída para utilizar como biblioteca principal. Infelizmente, não é possível remover totalmente esta dependência, pois uma das *frameworks* de

CSS que o *template* utiliza é o Bootstrap da versão 4, que ainda tem dependência do jQuery.

Por fim, sobram as restantes *frameworks*: Angular, React e Vue. As três são bastante conhecidas, e estão em constante concorrência para serem líderes no mercado. Embora existam prós e contras em cada uma, regra geral, a qualidade e eficiência da utilização destas bibliotecas depende mais do programador do que da biblioteca. Por isso, trata-se mais de um gosto pessoal do que uma escolha assente numa tabela comparativa de prós e contras. Por este motivo, e porque o programador está familiarizado com a *framework* Vue.js, e por forma a acelerar o processo de desenvolvimento, diminuindo a curva de aprendizagem, optou-se por esta.

4

DESENVOLVIMENTO

4.1 ARQUITECTURA DA FRAMEWORK

A *framework* consiste num sistema capaz de criar um contexto da informação de uma empresa, através da representação de uma rede de nós e relações, recorrendo às tecnologias das bases de dados gráficas. As redes de nós e relações representam-se de forma semelhante a um *mind map*. Por serem intuitivas para a mente humana, promovem uma compreensão mais eficaz ao utilizador. Por analogia, pretende-se que o sistema represente a informação graficamente de tal forma intuitiva, que a percepção da informação seja clarividente ao utilizador que a interpreta.

O sistema a desenvolver divide-se nos seguintes componentes lógicos:

1. Bases de dados Relacional e Gráfica
2. API para para consulta, procura, filtragem, introdução e processamento de informação
3. Tarefas e subprocessos para tratamento de informação que consome tempo de processamento
4. Módulo para Alarmística e Relatórios de Segurança
5. Painel de administração e painel de consulta e análise

A figura 15 trata-se de um diagrama que representa, de forma abstracta, os componentes do sistema desenvolvido, as dependências e formas de comunicação entre componentes.

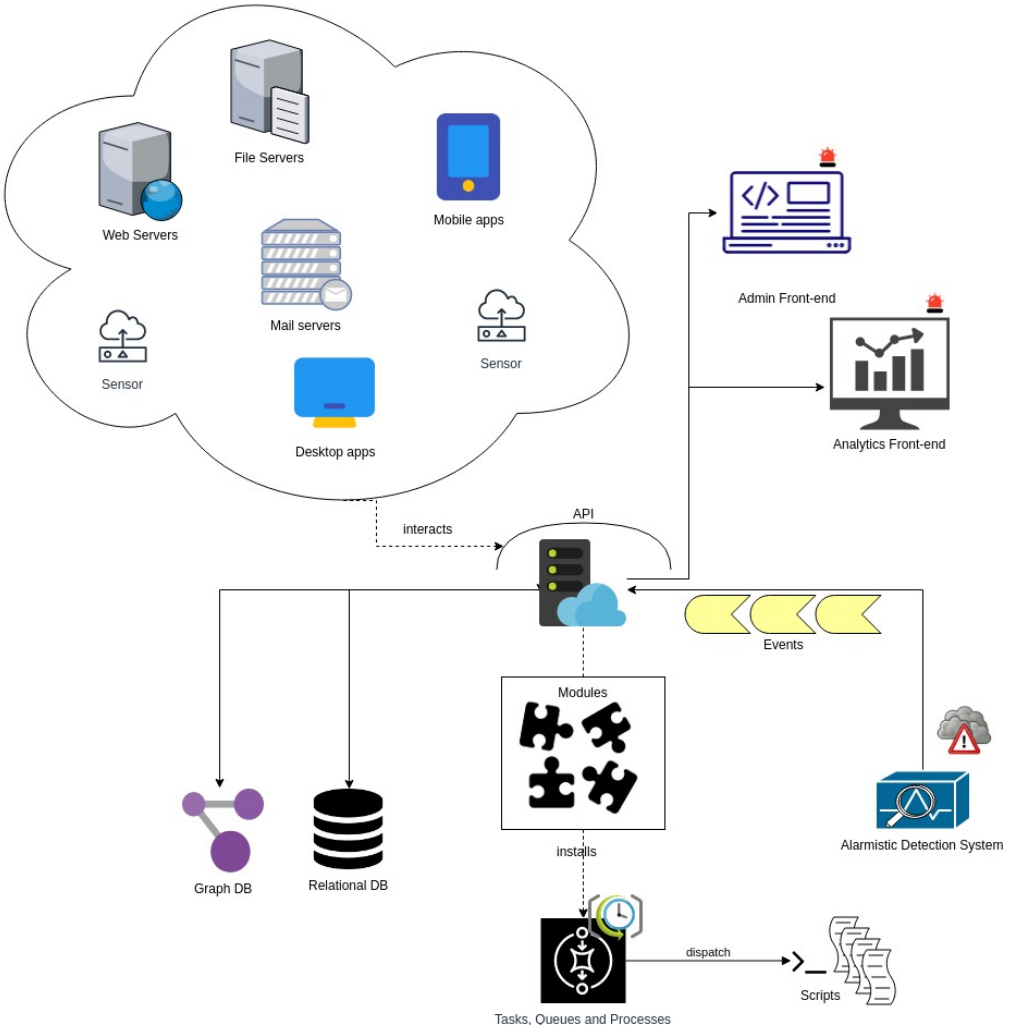


Figura 15: Arquitetura da Framework IBC

4.2 BASES DE DADOS

Existem, no mercado, diversas tecnologias de armazenamento de informação. As mais comuns são as bases de dados relacionais. É uma tecnologia que, apesar de antiga, é bastante estável, robusta e eficiente. Resolve a maioria dos problemas de representação dos modelos de domínio e a sua linguagem Structured Query Language (SQL) é um *standard* muito bem definido. No entanto, com o advento do *Big Data*, a complexidade dos dados aumentou substancialmente, dando origem a novos conceitos na forma como armazenamos os dados. Surgiram outras abordagens alternativas às bases de dados relacionais, como as bases de dados orientadas a objetos (ou documentos), chave-valor, à coluna e, por fim, as bases de dados gráficas.

4.2.1 *Base de dados relacional: Um auxílio administrativo*

Atualmente, a base de dados relacional guarda apenas informação que auxilia o bom desempenho da aplicação, ou informação secundária, cujo valor não é graficamente significante. Esta base de dados contém as tabelas de apoio à autenticação e gestão de permissões na plataforma, uma tabela de metadados de ficheiros guardados e ainda uma tabelas de histórico de pesquisa de informação sobre um determinado domínio (*WHOIS* [14]).

A seguinte figura demonstra o número atual de tabelas existentes na base de dados relacional:

```
ibc=# SELECT table_name FROM information_schema.tables WHERE table_schema = 'public';
                table_name
-------------------------------------------
 django_migrations
 django_content_type
 auth_permission
 auth_group
 auth_group_permissions
 auth_user
 auth_user_groups
 auth_user_user_permissions
 django_admin_log
 analysis_graphlinkconfig
 analysis_graphnodeconfig
 authtoken_token
 django_rest_passwordreset_resetpasswordtoken
 domains_whois
 files_metadata
 django_session
(16 rows)
```

Figura 16: Tabelas da base de dados relacional do projeto IBC

4.2.2 *Base de dados gráfica: Hierarquia de nós e relações*

Para construir uma base de dados gráfica de nós e relações, é importante ter em conta uma hierarquia base, que possa responder às necessidades do *software*.

À semelhança de um texto jornalístico que pretende responder às questões:

1. Quem?
2. O quê?
3. Quando?
4. Onde?
5. Porquê?

6. Como?

Também num sistema que pretende representar o contexto empresarial, precisa de representar informação e eventos de forma semelhante, para responder a este tipo de questões 4.2.2.

Com efeito, é necessário criar um modelo capaz de responder e englobar todas estas questões.

Considere-se o seguinte modelo de domínio:

1. Entidade
2. Objeto
3. Evento
4. Local

Uma Entidade ou Objeto representam respostas à pergunta 1. Os Eventos representam as respostas às perguntas 2 e 3. O Local representa a resposta à pergunta 4. Por fim, sobram as questões 5 e 6, que são respondidas analisando o contexto e as relações entre os nós.

Todos os nós são estruturados, isto é, definidos estaticamente.

4.2.2.1 *Entidades*

Um nó do tipo Entidade é um nó geral, que pode uma Pessoa Singular ou Colectiva no mundo real, isto é, Pessoa ou Organização/Empresa.

Toda a Entidade tem dois campos obrigatórios: nome e número de contribuinte. O número de contribuinte é único em todas as Entidades.

A origem de um evento ou a simples representação de uma informação pertinente é sempre uma Entidade ou um Objeto. Por exemplo: Pretende-se representar todos os colaboradores de uma Empresa. A empresa é uma Entidade do tipo Empresa. Para cada colaborador, cria-se um nó do tipo Pessoa e associa-se uma relação entre o colaborador e a empresa.

4.2.2.2 *Objetos*

Os Objetos são representações de coisas reais ou digitais. Elas representam também informação ou a origem de um evento. Por exemplo: Um computador foi abaixo. A origem do evento é um Objeto e não uma entidade. Ainda: Pretende-se representar

os ficheiros de uma dada diretoria. Cada ficheiro é um nó Objeto que pode ter informação como nome do ficheiro, extensão e caminho do ficheiro.

4.2.2.3 *Eventos*

Eventos representam acontecimentos ou "marcos históricos"que ocorreram no mundo digital ou físico, e que podem ajudar a compreender melhor a informação e o contexto. O estado atual do projeto ainda não implementa nós de eventos.

4.2.2.4 *Locais*

Os Locais representam o lugar ou sítio digital onde se encontra determinada informação ou ocorreu o Evento. O estado atual do projeto ainda não implementa nós de localização.

4.2.2.5 *Dados dinâmicos*

Para dados dinâmicos, poderiam ser criados nós cuja estrutura não é definida programaticamente. Embora a biblioteca *neomodel* permita criar nós dinâmicos, existem algumas limitações na mesma. Com efeito, os dados dinâmicos que existem até à data estão guardados na base de dados relacional, visto que este tipo de dados não têm importância gráfica significativa e, deste modo, não é necessário utilizar a biblioteca *neomodel*.

Os dados dinâmicos na aplicação são: os metadados; a resposta dos domínios ao comando *WHOIS*. As tabelas são *files_metadata* e *domains_whois*, respetivamente.

A tabela de *files_metadata* tem a seguinte estrutura:

COLUNA	TIPO DE DADOS	DESCRIÇÃO
id	inteiro	id (único) da linha
file_uuid	UUID [36]	UUID do nó ficheiro na BD gráfica
metadados	JSON [63] (binário) [51]	guarda os metadados do nó ficheiro em causa

Tabela 2: Estrutura da tabela de metadados.

```
ibc=# \d files_metadata;
                          Table "public.files_metadata"
  Column   |  Type   | Collation | Nullable |                  Default
-----------+---------+-----------+----------+-------------------------------------------
 id        | integer |           | not null | nextval('files_metadata_id_seq'::regclass)
 file_uuid | uuid    |           | not null |
 metadata  | jsonb   |           | not null |
Indexes:
    "files_metadata_pkey" PRIMARY KEY, btree (id)
    "files_metadata_file_uuid_cabf009f" btree (file_uuid)
```

Figura 17: Estrutura da tabela *files_metadata*

A tabela *domains_whois* é uma tabela que guarda o histórico das vezes que o utilizador executou o comando *WHOIS*, e tem a seguinte estrutura:

COLUNA	TIPO DE DADOS	DESCRIÇÃO
id	inteiro	id (único) da linha
domain_uuid	UUID [36]	uuid do nó *domain* na BD gráfica
queried_at	*timestamp*	data e hora do pedido
data	JSON [63] (binário) [51]	resultado *WHOIS* do nó em causa
ran_in_terminal	booleano	se falso, foi um script em Python
data_hash	*string*	hash dos dados, para verificar se já uma entrada igual (histórico)
hash_algorithm	*string*	indica o algoritmo de dados utilizado

Tabela 3: Estrutura da tabela da informação sobre os domínios.

```
ibc=# \d domains_whois;
                                    Table "public.domains_whois"
     Column      |           Type           | Collation | Nullable |                 Default
-----------------+--------------------------+-----------+----------+-----------------------------------------
 id              | integer                  |           | not null | nextval('domains_whois_id_seq'::regclass)
 domain_uuid     | uuid                     |           | not null |
 queried_at      | timestamp with time zone |           | not null |
 data            | jsonb                    |           | not null |
 ran_in_terminal | boolean                  |           | not null |
 data_hash       | character varying(255)   |           | not null |
 hash_algorithm  | character varying(255)   |           | not null |
Indexes:
    "domains_whois_pkey" PRIMARY KEY, btree (id)
    "domains_whois_data_hash_896d2473" btree (data_hash)
    "domains_whois_data_hash_896d2473_like" btree (data_hash varchar_pattern_ops)
    "domains_whois_hash_algorithm_7d6f4580" btree (hash_algorithm)
    "domains_whois_hash_algorithm_7d6f4580_like" btree (hash_algorithm varchar_pattern_ops)
```

Figura 18: Estrutura da tabela *domains_whois*

Cada linha na tabela *domains_whois* é um registo que contém a informação do domínio à data *queried_at*. Deve ser guardado um histórico dos pedidos porque

a informação sobre o domínio pode mudar ao longo do tempo. No entanto, para poupança de espaço e para não duplicar informação desnecessária, facilitando a filtragem e a comparação dos resultados ao utilizador, não se grava uma nova linha nesta tabela sempre que o utilizador executa o pedido para obter informações do domínio. Em vez disso, só é guardada informação se o novo resultado for diferente do anterior. Para tal, é calculado o *hash* do resultado e, se diferente, então cria-se uma nova entrada na tabela.

4.3 PAINEL DE ADMINISTRAÇÃO

O site da aplicação é um painel administrativo, que foi desenvolvido utilizando a framework NUXT, com um *template* pago (ver 3.2.2.2).

O painel foi desenhado para mapear internamente os módulos da API. No menu lateral esquerdo (figura 19), podemos verificar que existem dois grandes módulos, que servem como agrupadores, o módulo de entidades 4.5.7 e o módulo de objetos 4.5.12. Dentro do módulo de Objetos, verificam-se muitos outros submódulos, que irão ser explicados mais tarde 4.5. Os módulos a vermelho encontram-se inativos, ou seja, não existe ainda uma interface de utilizador para estes módulos, por não existirem ou encontrarem-se numa fase prematura de desenvolvimento.

O botão *Configuration* permite configurar algumas definições gerais e específicas a cada módulo no servidor. Por exmeplo, para o módulo *Leaks* 4.5.11, configura-se o e-mail e o token para aceder ao serviço da API DeHashed.

O botão *Analysis* permite aceder à página de análise, para efetuar pesquisas gráficas à aplicação. Irá ser descrito com mais detalhe em 4.5.1.

Por fim, o botão *NEO4J* é um link externo para aceder à página web da BD gráfica. Este *endpoint* encontra-se exposto porque a aplicação ainda se encontra em desenvolvimento. Numa fase mais madura da aplicação, remover-se-á este *endpoint* do acesso ao público, por forma a que os pedidos sejam consultados apenas pelo módulo *analysis* 4.5.1.

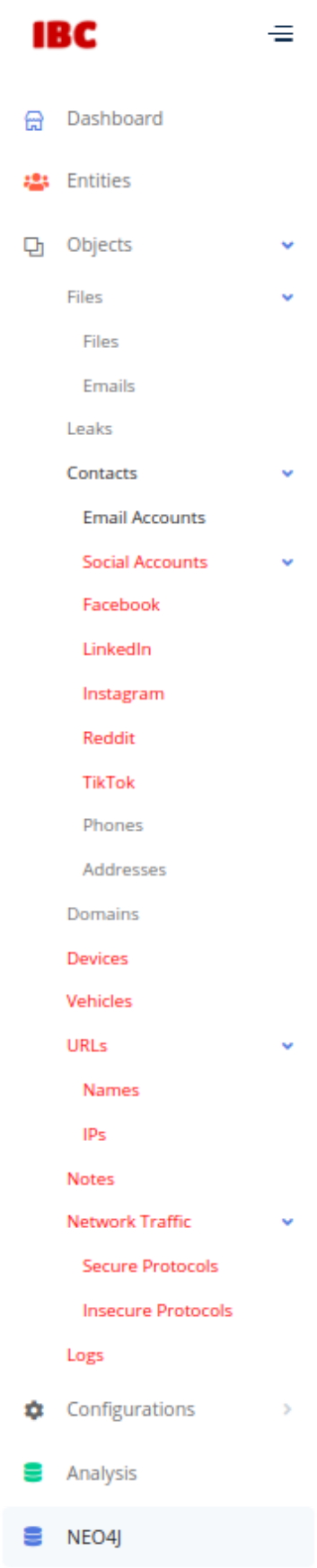


Figura 19: Menu lateral esquerdo da aplicação

4.4 PAINEL DE ANÁLISE DE INFORMAÇÃO

O painel de análise de dados, por se encontrar ainda em desenvolvimento, não está separado do painel administrativo. Segundo a arquitetura da *framework* idealizada 15, pretendem-se duas secções completamente distintas, isto é, duas aplicações *web* distintas, para melhor segurança e para separação de responsabilidades por perfis de utilizadores. No entanto, continua integrado no painel administrativo para simplificação do desenvolvimento da aplicação e porque as partes interessadas neste projeto preferem que assim se mantenha até que sejam determinadas pessoas específicas para cada perfil. Pessoas com acesso administrativo e pessoas com acesso para análise da informação e que possam gerar relatórios.

4.5 FRAMEWORK MODULAR

Pretende-se uma *framework* cujo código seja fácil de manter e melhorar. Como tal, decidiu-se seguir o princípio de Engenharia de Software sobre a Separação de Interesses, em inglês: Separation of Concerns (SoC) [27]. Por isso, modularizou-se a *framework* no seguinte:

MÓDULO	OBJETIVO
analysis	o módulo de análise de dados. É uma API que permite fazer *queries* à aplicação sobre os nós na base de dados gráfica.
authentication	gere a autenticação da framework
config	gere as configurações que o administrador pode alterar dinamicamente
core	conjunto de recursos essenciais à *framework*
domains	CRUD dos nós do tipo «Domain»
emails	processamento de e-mails, CRUD dos nós do tipo «Email»
entities	CRUD dos nós do tipo «Entity»
files	upload de ficheiros e CRUD de nós do tipo «File»
filesystem	permite o acesso a uma pasta no sistema de ficheiros do servidor
ibc	*bootstrap* aplicacional e onde se definem as configurações da aplicação
leaks	descoberta de *leaks* e CRUD de nós do tipo «Leak»
objects	CRUD de objetos
social	módulo para redes sociais. CRUD de nós do tipo «SocialAccount» (em desenvolvimento)

Tabela 4: Módulos da *Framework* IBC

Cada módulo define os seus próprios tipos de nós, através da herança, um dos quatro pilares da Programação Orientada a Objetos (POO). Na base de dados gráfica, o nome da classe é mapeado para a etiqueta do nó.

A classe base de onde estendem todos os nós da aplicação, está definido no módulo core e chama-se *Node*. Se existir uma *class* filha de *Node*, então o nome da *class* reflete a etiqueta na BD gráfica.

Por exemplo: Se existir a «A», que extende de *Node*, então, sempre que se criar um nó do tipo «A» no servidor, a base de dados grava esse nó com as duas etiquetas: «A» e «Node».

A hierarquia atual da aplicação é a seguinte:

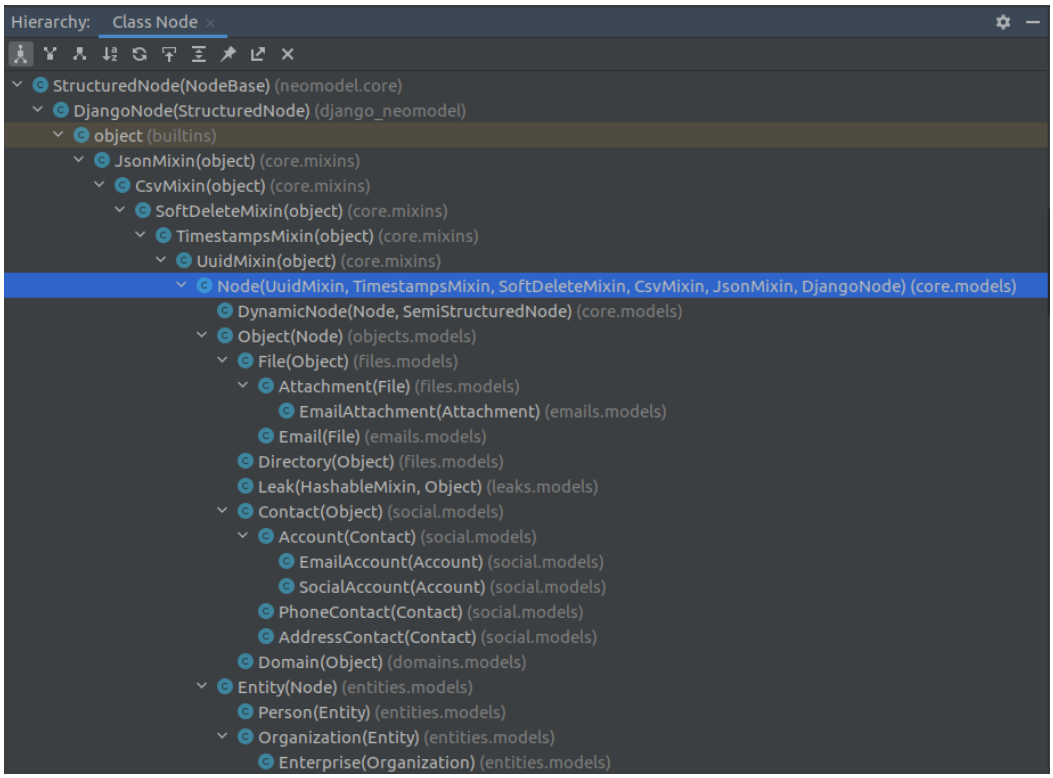


Figura 20: Hierarquia dos nós da aplicação IBC

Na figura 20, podemos constatar que uma instância de *Enterprise* terá como etiquetas em BD: «*Enterprise*», «*Organization*», «*Entity*» e, por fim, «*Node*» e uma instância de *SocialAccount* terá as etiquetas: «*SocialAccount*», «*Account*», «*Contact*», «*Object*» e «*Node*».

4.5.1 *Módulo «analysis»*

Este módulo devia chamar-se «*analytics*», mas devido aos conflitos com software de deteção de *adware*, os quais bloqueiam rotas URL que contenham um conjunto de palavras específicas, tal como a palavra «*analytics*» e «*ads*», chama-se «*analysis*».

Tem como objetivo fornecer ao cliente uma API para *query* de informação dos dados, de forma intuitiva ao analista, sem depender de conhecimentos de SQL ou *Cypher (Query Language)*. Ainda está em desenvolvimento.

O objetivo deste módulo é assemelhar-se à atual interface gráfica do *Neo4j Desktop*.

Figura 21: Módulo de análise gráfica dos dados

Na figura 21, o painel de análise de dados. À direita, em «*Database Information*» estão as opções possíveis de selecionar e configurar para efetuar o pedido.

Para executar uma *query*, é possível selecionar qual o tipo de nó que se quer obter como resultado. No caso da figura 21, está selecionada a etiqueta «*Domain*», portanto, obter-se-á nós desse tipo. A seguir, é possível escolher, em «*Relationship Types*» as relações e os nós dessas relações que estão relacionados com o nó algo, selecionado em «*Node label*». Por fim, nas etiquetas a amarelo, as «*Allowed labels in relationships*» representa a seleção do tipo nós que o cliente quer obter, que se relacionam com o tipo de nó pretendido, através do conjunto de relações que foram selecionadas nas etiquetas a vermelho.

Por defeito, se nenhuma relação for selecionada, assumem-se todas as relações do nó em pesquisa. Se nenhum tipo de nó relacionado for selecionado, o programa assume que se quer obter todos os nós nas relações com nó em pesquisa.

Em «*Properties*», é possível aplicar filtros de pesquisa pelas propriedades que o Domínio possui.

Por fim, para não sobrecarregar a aplicação, o utilizador pode escolher um limite máximo de nós compreendido entre 50 e 300 inclusive e uma densidade de pesquisa até 3 níveis. A densidade representa os níveis das relações.

Criando o seguinte cenário:

Para os nós «A», «B», «C» e «D», todos eles são nós com a *label* «*Person*» e relacionam-se entre si deste modo:

- (A)-[LIKES]->(B)
- (B)-[KNOWS]->(C)
- (C)-[IS_MARRIED_TO]->(D)

Para efeitos de demonstração, considere que o nome é um campo com uma restrição em que estes têm de ser únicos em base de dados. Internamente, as *queries* executadas são bem mais complexas, e utilizam UUIDs para garantir a coerência da pesquisa. A pessoa «A» chama-se Ana. A pessoa «B», Bernardo, a pessoa «C» chama-se Carlos e a pessoa «D» chama-se Diana.

Suponha-se a pesquisa de um nó do tipo Pessoa, com um filtro de pesquisa pelo campo nome igual a "Ana". Internamente, a aplicação irá executar a seguinte *query* em CQL[47]:

MATCH (p:Person) WHERE p.name = 'Ana' RETURN p;

Se a densidade de nós for igual a zero, a pesquisa termina, devolvendo apenas o nó «A» para a vista de análise.

Se a densidade de nós for igual a um, então, para além da *query* anterior, o servidor irá procurar por todas as relações que a Ana tem. Isto é,

MATCH (p1:Person)-[]-(p2:Personname: 'Ana') RETURN p1;

Se a densidade for igual a dois, o servidor irá, de novo, procurar as relações do nó com a densidade atual, isto é, o nó da pessoa «B», executando a *query*:

MATCH (p1:Person)-[]-(p2:Personname: 'Bernardo') RETURN p1;

Obtém-se também o nó C.

Assim será por diante até chegar ao limite da densidade. Portanto, como exemplificado acima, existe um algoritmo recursivo na aplicação que vai buscar os nós filhos, enquanto não atingir o valor máximo da densidade.

Nota

A *syntax* da linguagem CQL tem muitas diferenças em relação à linguagem SQL. Para um utilizador mais interessado sobre a tecnologia de *query* à base de dados gráfica em Neo4j, é recomendado que leia a documentação oficial disponível *online* em [47].

Ainda na mesma vista 21, a norte, encontra-se uma entrada de texto para escrever um termo de pesquisa que é aplicado globalmente na base de dados gráfica. Isto é, se o utilizador quiser procurar por qualquer nó, em qualquer propriedade do tipo

string, em todos os nós em base de dados, é só escrever esse termo na entrada de texto e a *query* será aplicada globalmente.

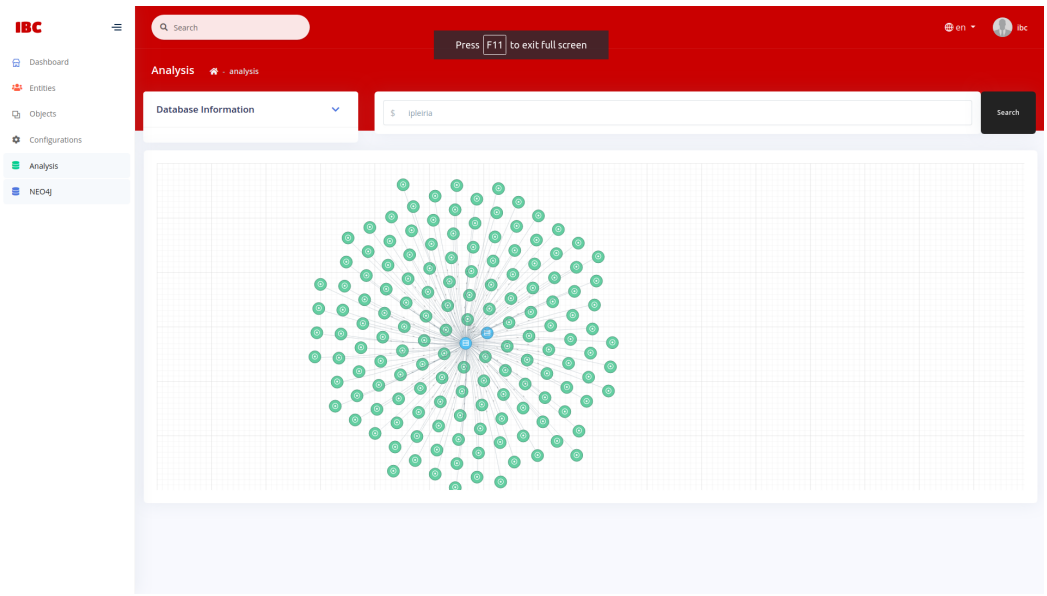


Figura 22: Exemplo de pesquisa no módulo *analysis*

A figura 22 é um exemplo de uma *query* executa, onde foi aplicado o filtro global "ipleiria"e com o objetivo de encontrar nós do tipo «Domínio». A azul, os domínios encontrados e a verde as contas de e-mail. Como foi aplicado com densidade igual a um, obteve-se também as contas de e-mail associadas a este domínio (previamente descobertas através do módulo de *leaks*) e ainda um subdomínio que é o «my.ipleiria.pt». Por exemplo, se fosse aplicado a densidade dois, devolveria os *leaks* relacionados a estas contas.

4.5.2 *Módulo «authentication»*

Este módulo permite autenticar um utilizador na plataforma. Atualmente, só existem utilizadores que são administradores. Com o desenvolvimento da plataforma de análise, pretende-se criar diferentes grupos e perfis de utilizador.

Neste módulo, utilizou-se a autenticação por *tokens*, recorrendo à *framework* . Não foi necessário criar um modelo de Utilizador para autenticação, visto que o que está implementado pela *framework* é suficiente. Apenas foi necessário criar o código respetivo para o registo e recuperação de password.

```python
# Account details
# Url: https://<your-domain>/auth/register/
# Headers:
#     Authorization: Token <token>
class RegistrationView(views.APIView):
    def post(self, request):
        serializer = RegistrationSerializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        serializer.save()

        return Response({'detail': 'User successfully registered.', })


# User details
# Url: https://<your-domain>/auth/account/
# Headers:
#     Authorization: Token <token>
class UserViewSet(viewsets.GenericViewSet):
    def list(self, request):
        serializer = UserSerializer(request.user)
        return Response(serializer.data)

    @action(methods=['post', 'put', ], detail=False, url_path='password/change', url_name='change_password')
    def change_password(self, request):
        user = self.request.user

        serializer = ChangePasswordSerializer(data=request.data)
        serializer.is_valid(raise_exception=True)

        # Check old password
        if not user.check_password(serializer.data.get("old_password")):
            return Response({"old_password": ["Wrong password."]}, status=status.HTTP_400_BAD_REQUEST)

        # set_password also hashes the password that the user will get
        user.set_password(serializer.data.get("new_password"))
        user.save()

        return Response({"detail": "Password successfully changed."})
```

Figura 23: Vistas de autenticação que foram necessárias desenvolver

Como é possível observar pela imagem 23, não é permitido a um convidado efetuar o registo na aplicação. Para registar um utilizador, é preciso estar autenticado. Esta funcionalidade bloqueia, deste modo, o registo indevido de utilizadores potencialmente indesejados.

```python
from rest_framework.authtoken.views import obtain_auth_token

app_name = 'auth'

router = routers.DefaultRouter()

router.register('account', UserViewSet, 'account')


urlpatterns = [
    path('login/', obtain_auth_token, name="login"),
    path('register/', RegistrationView.as_view(), name="register"),
    url(r'^password/reset/', include('django_rest_passwordreset.urls', namespace='password_reset')),
]

urlpatterns += router.urls
```

Figura 24: *Endpoints* de autenticação da *framework*

A interface gráfica desenvolvida para autenticação, é a seguinte:

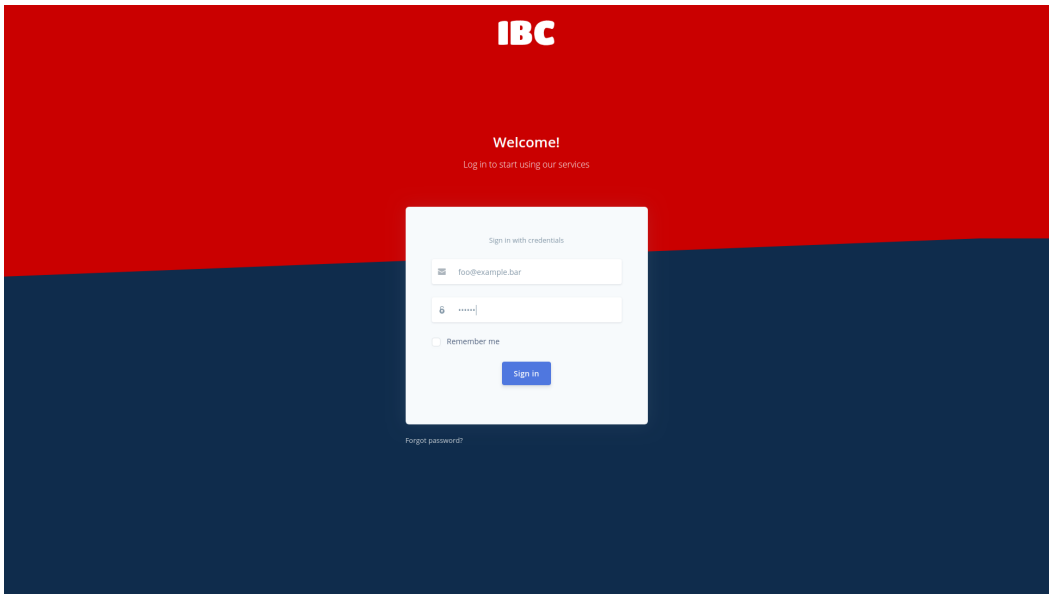


Figura 25: Página de autenticação

Por fim, a página para recuperação de password, que envia um e-mail com um link para recuperação da mesma, é:

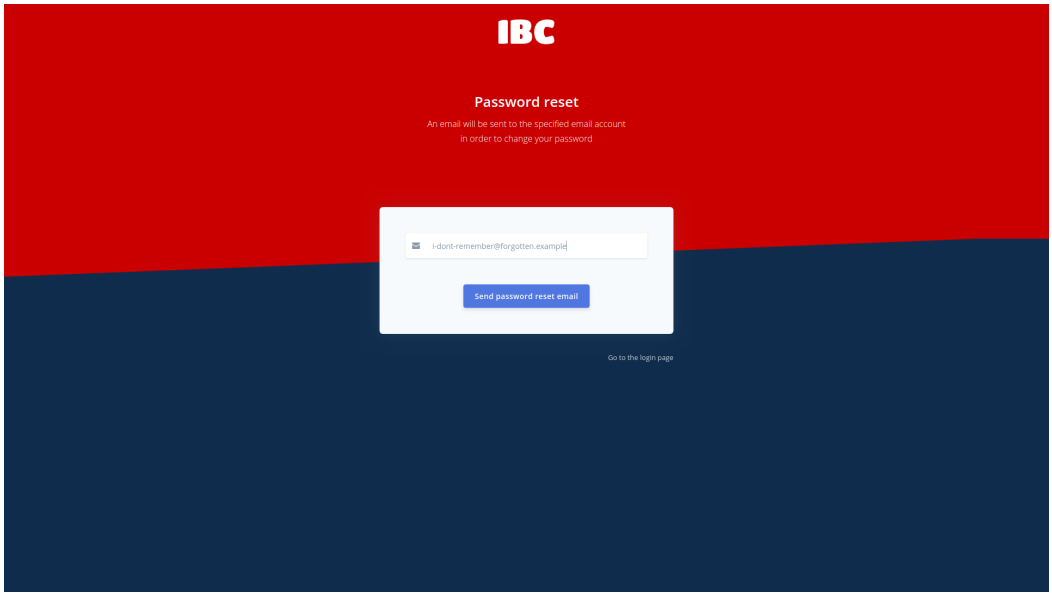


Figura 26: Página de autenticação

Não foi ainda desenvolvida a interface gráfica para registo de novos utilizadores, pois o cliente considera que essa funcionalidade não é prioritária.

Após a autenticação, o utilizador é redirecionado para a sua dashboard.

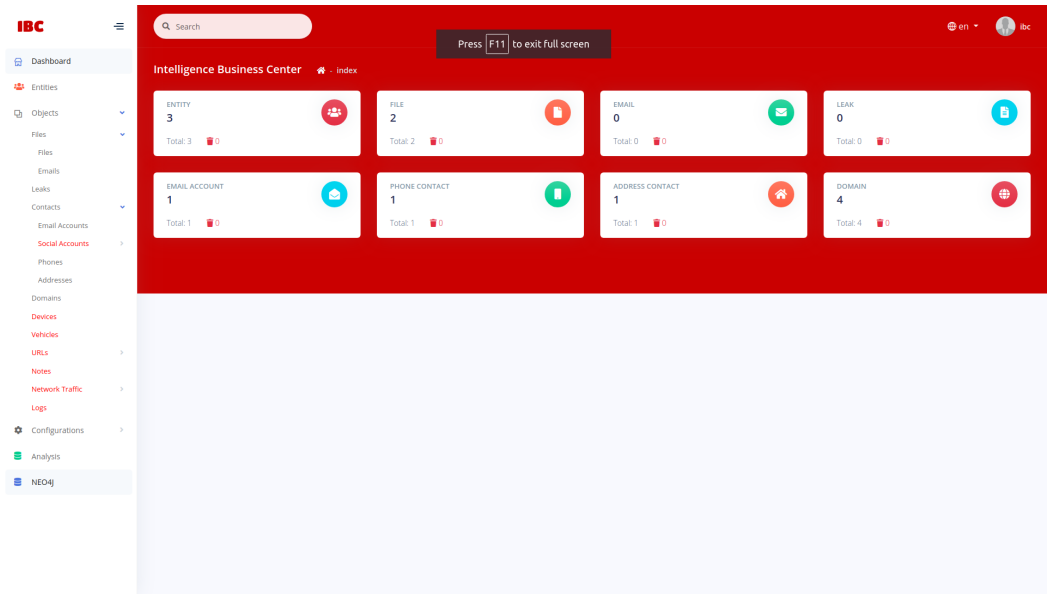


Figura 27: Dashboard

4.5.3 *Módulo «config»*

O módulo de configuração pretende ser um módulo que permite alterar algumas definições no servidor, em *runtime*. Por exemplo, as chaves para as APIs externas podem alterar com o tempo, dado a novas subscrições ou refrescamento de novos *tokens*. Por isso, é importante existir uma página para alteração destas configurações, sem a necessidade de ser o programador a alterar as configurações manualmente, obrigando o serviço web a ser reiniciado.

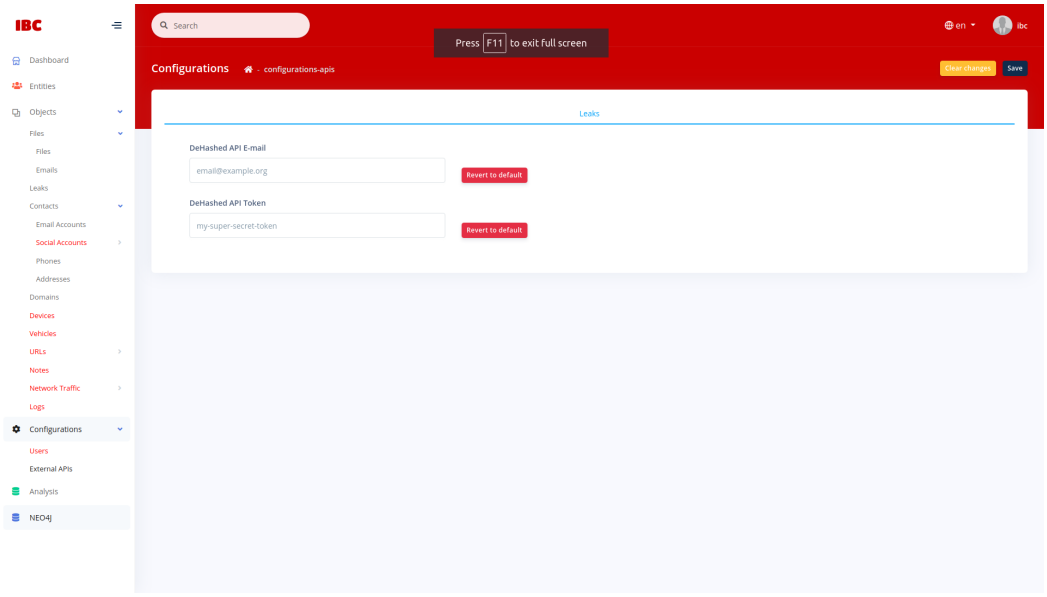


Figura 28: Página de configurações. Atualmente, permite apenas configurar a conta de e-mail e token associados à API da DeHashed.

4.5.4 *Módulo «core»*

O módulo *core* não disponibiliza nenhuma página em concreto para o utilizador. Antes, é um conjunto de funcionalidades, código, funções e objetos que são reaproveitados e importantes para o bom funcionamento do sistema.

4.5.5 *Módulo «domains»*

Neste módulo pretende-se a gestão CRUD de domínios registados na plataforma. Contém as vistas e o código respetivo para a gestão dos mesmos, bem como serviços à escuta para descoberta de subdomínios.

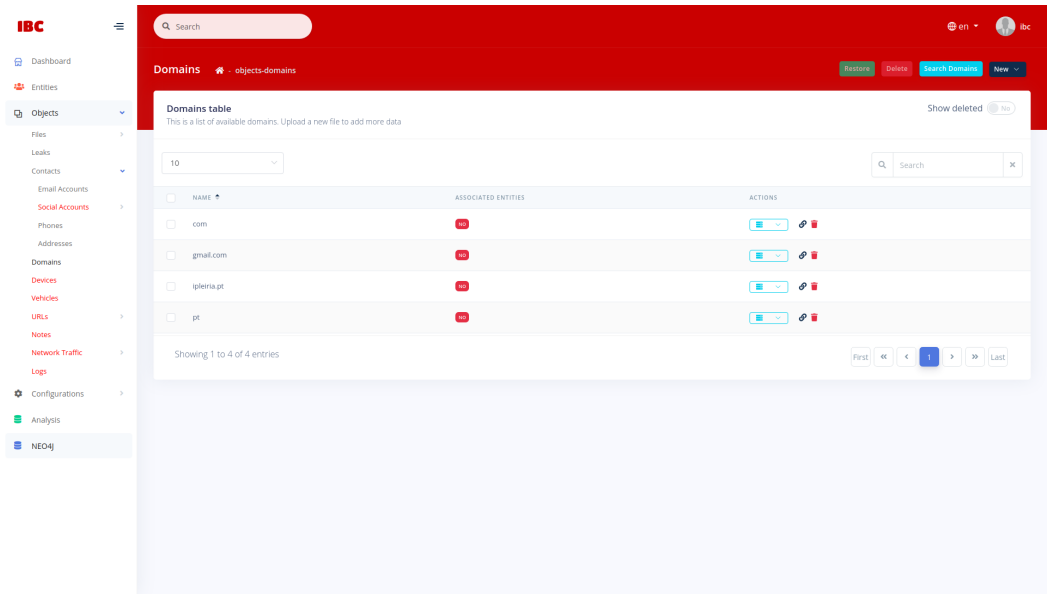


Figura 29: Página de listagem de domínios.

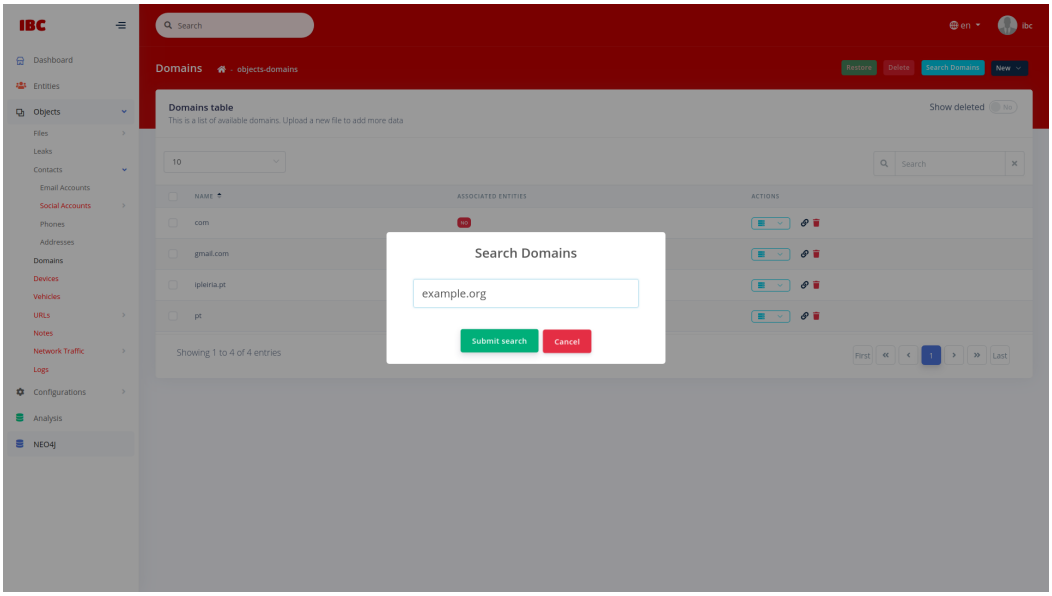


Figura 30: Dado um domínio, procura os seus subdomínios

Um domínio, por vezes, é constituído por subdomínios. A figura 30, ilustra que o utilizador pode descobrir subdomínios, apenas terá de carregar no botão «*Dig for Subdomains*». O que internamente este comando faz é procurar subdomínios utilizando os utilitários «Amass»[50] e «Sublist3r»[1]. No entanto, pretende-se integrar mais ferramentas no servidor [33, 53], tais como o «Subfinder»[52] e o «Sudmoy»[58].

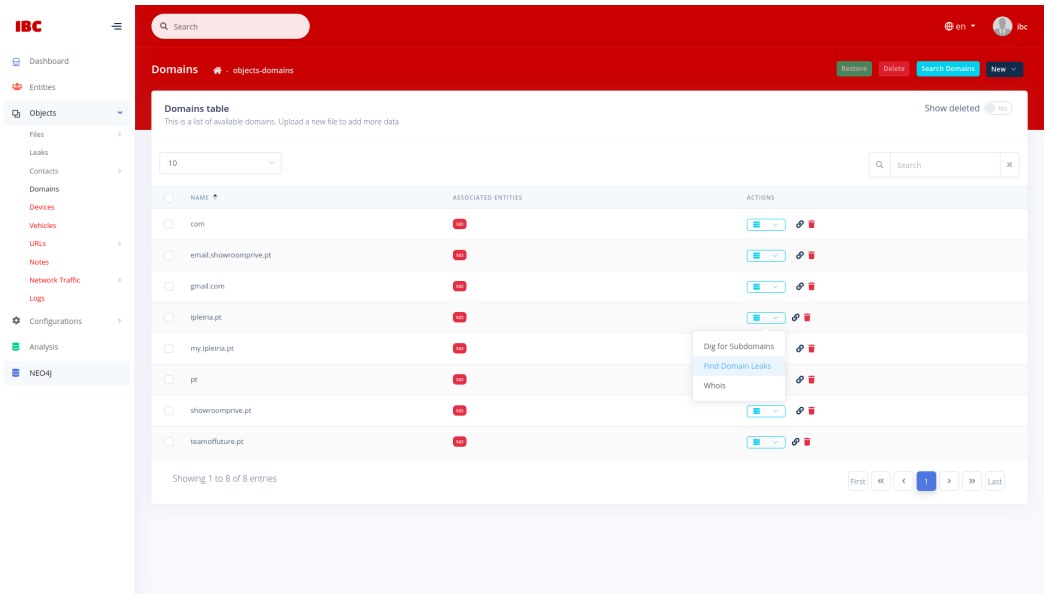


Figura 31: Dado um domínio, procura *leaks* desse domínio

É possível procurar *leaks* de relativos a um domínio, clicando apenas no botão «*Find Domain Leaks*», conforme se pode verificar pela figura 31.

Por fim, neste módulo é ainda possível consultar informação acerca um domínio através do protocolo WHOIS[14]. O resultado desta informação pode ser aproveitada para descobrir mais informação importante, como por exemplo e-mails, moradas e o nome da empresa que possui este domínio.

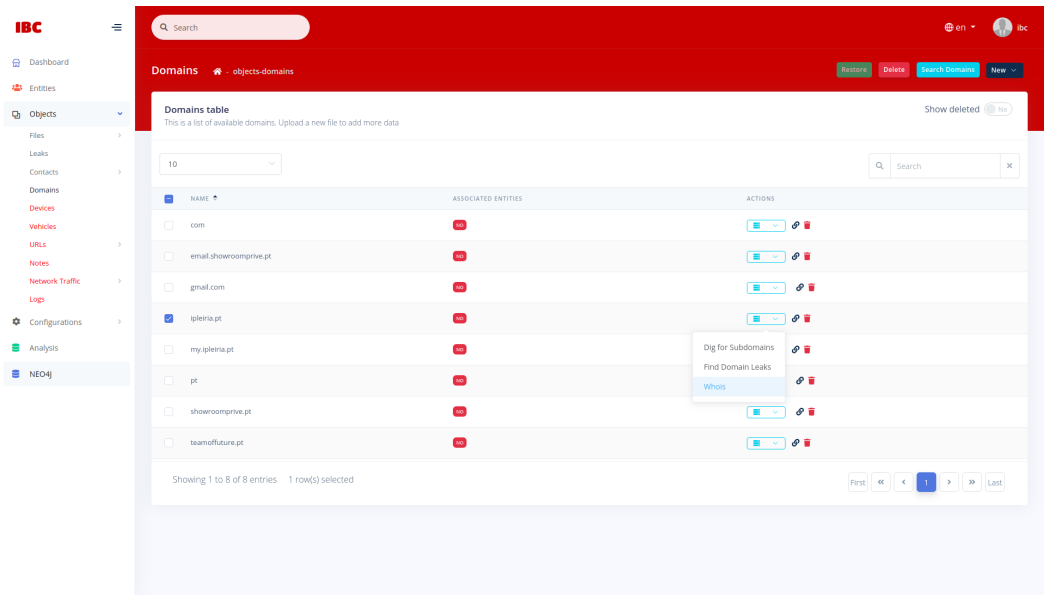


Figura 32: Menu de opções do domínio, onde existe opção de consultar informação de um domínio pelo protocolo WHOIS

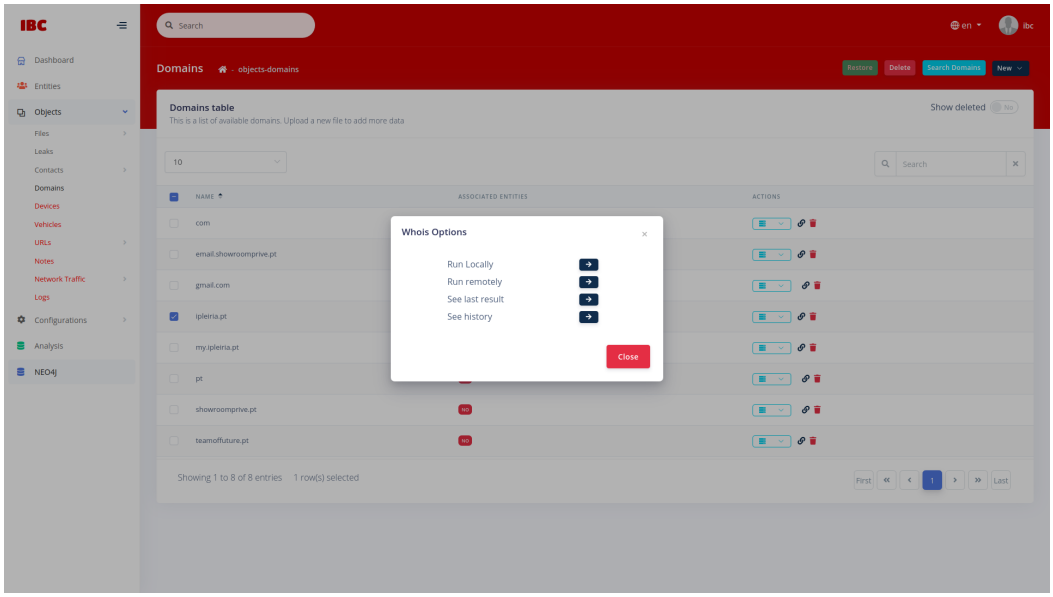


Figura 33: Menu de opções para consulta da informação WHOIS

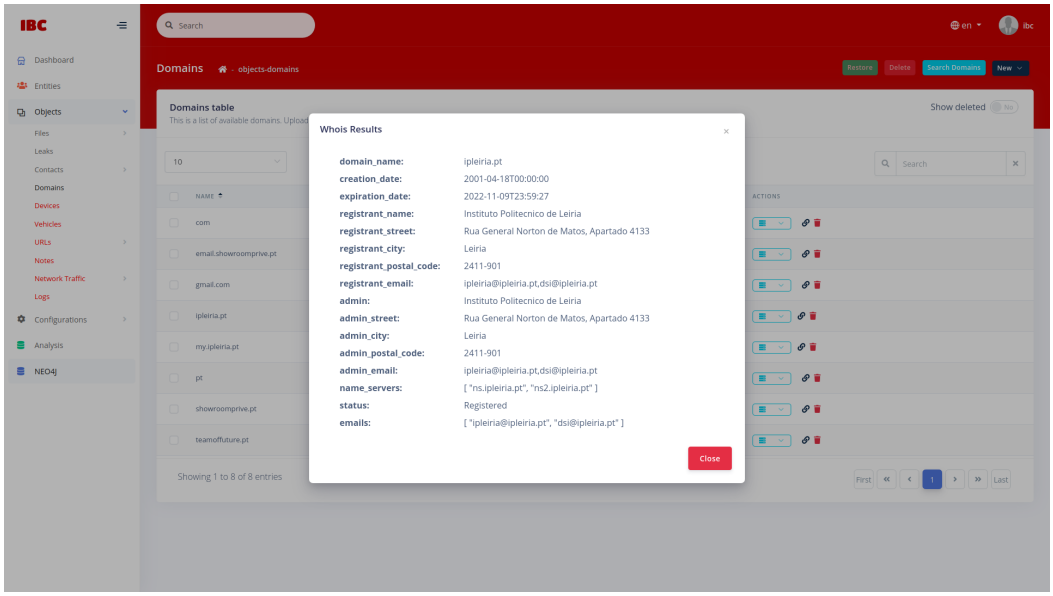


Figura 34: Exemplo de resultados de um pedido ao protocolo WHOIS

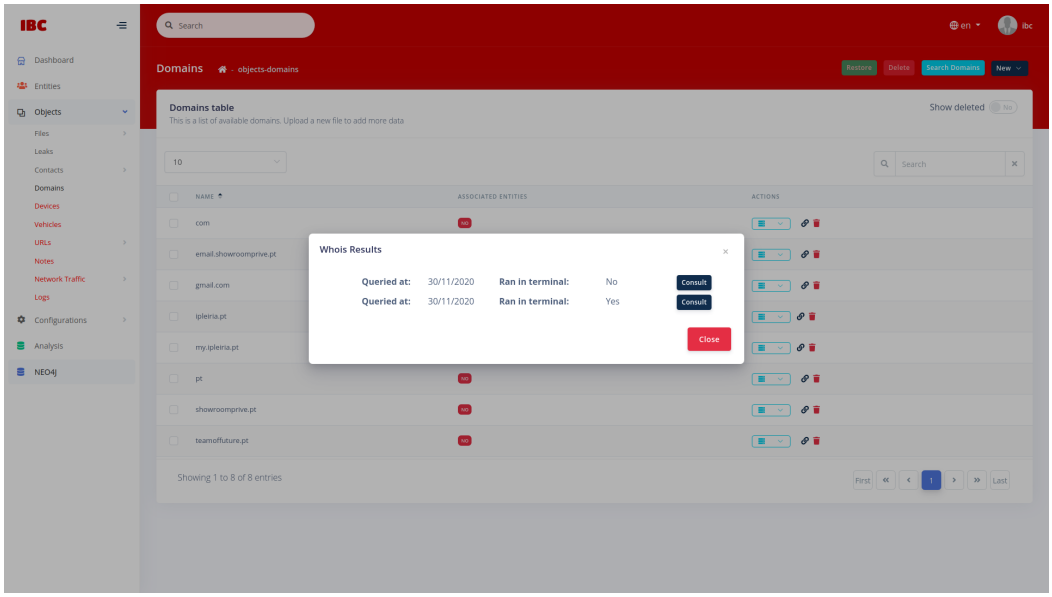


Figura 35: Histórico de pesquisas feitas ao protocolo WHOIS

A figura 32 mostra como se acede às opções do menu WHOIS. Ao clicar nessa opção, a aplicação mostra um menu de opções 33 para consultar informação a partir do protocolo WHOIS. As primeiras duas opções representam o pedido de informação ao protocolo WHOIS, mas de duas maneiras distintas. A primeira corre localmente, isto é, utiliza o utilitário «WHOIS» que já vem instalado no servidor. Para correr esta ferramenta, o servidor cria um processo à parte, executa o comando, e o resultado é obtido e formatado. A opção remota recorre a uma biblioteca em Python, que faz um pedido a um conjunto de servidores que respondem a este protocolo. Os resultados são similares 34, porém, mantém-se ambas as formas porque existem pequenas diferenças que podem ser interessantes para comparar e/ou obter mais informações. A aplicação guarda um histórico das pesquisas que o utilizador submeteu, visto que a informação varia ao longo do tempo, e pode ser interessante manter o rastreamento da informação 35.

Atualmente a informação obtida é guardada "como vem". Isto é, não é efetuado nenhum tipo de tratamento de dados para gerar novos nós e relações com este domínio. Este trabalho ainda não foi feito porque a análise dos campos do resultado da pesquisa WHOIS varia, e torna o processo complexo. Ver [38]. No entanto, está previsto para o futuro melhorar este módulo para descobrir novas informações e/ou falhas de segurança, através da análise semântica [38].

4.5.6 *Módulo «emails»*

Com este módulo, pretende-se ler e processar e-mails. Até à data são suportados os tipos de ficheiros *.eml* [13, 54, 55], *.pst* [42] e *msg* [40]. Estes são maioritariamente os tipos mais utilizados. No futuro, adicionar-se-ão mais tipos suportados, nomeadamente os ficheiros de e-mail gerados pela Apple.

Para um e-mail ser processado, deve-se fazer primeiramente o upload do ficheiro pelo módulo *files*. Após o upload, o servidor ficará encarregue de processar o email, através de um serviço que corre periodicamente e vai obter todos os ficheiros ainda não processados que tenham as extensões válidas acima indicadas.

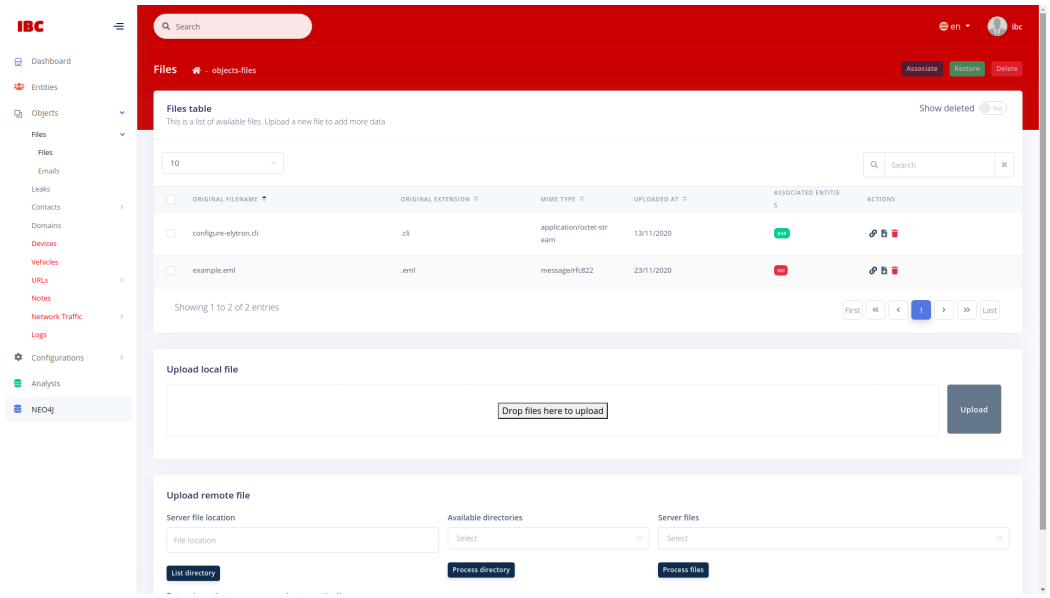


Figura 36: Exemplo de upload de um ficheiro do tipo e-mail. A extensão indica «.eml» e o *mime-type* é *message/rfc822*.

Depois de processado o e-mail, são criados um conjunto de nós e relações que refletem o que foi processado. É criado o nó «Email» e as contas que comunicaram entre si: «Account», isto é, as contas de e-mail que enviaram (campo *From*) e receberam (campos *To*, *Cc* e *Bcc*).

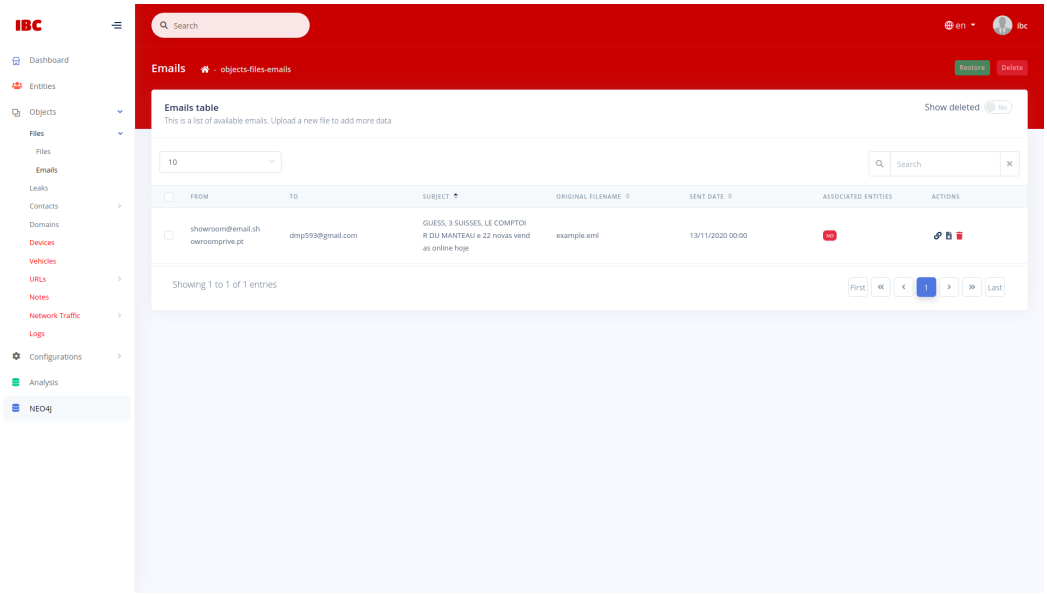


Figura 37: Apresentação da listagem de e-mails já processados pela plataforma

Após o upload, e assim que o ficheiro é processado, passa a ser listado no módulo de e-mail, conforme se pode verificar pela figura 37.

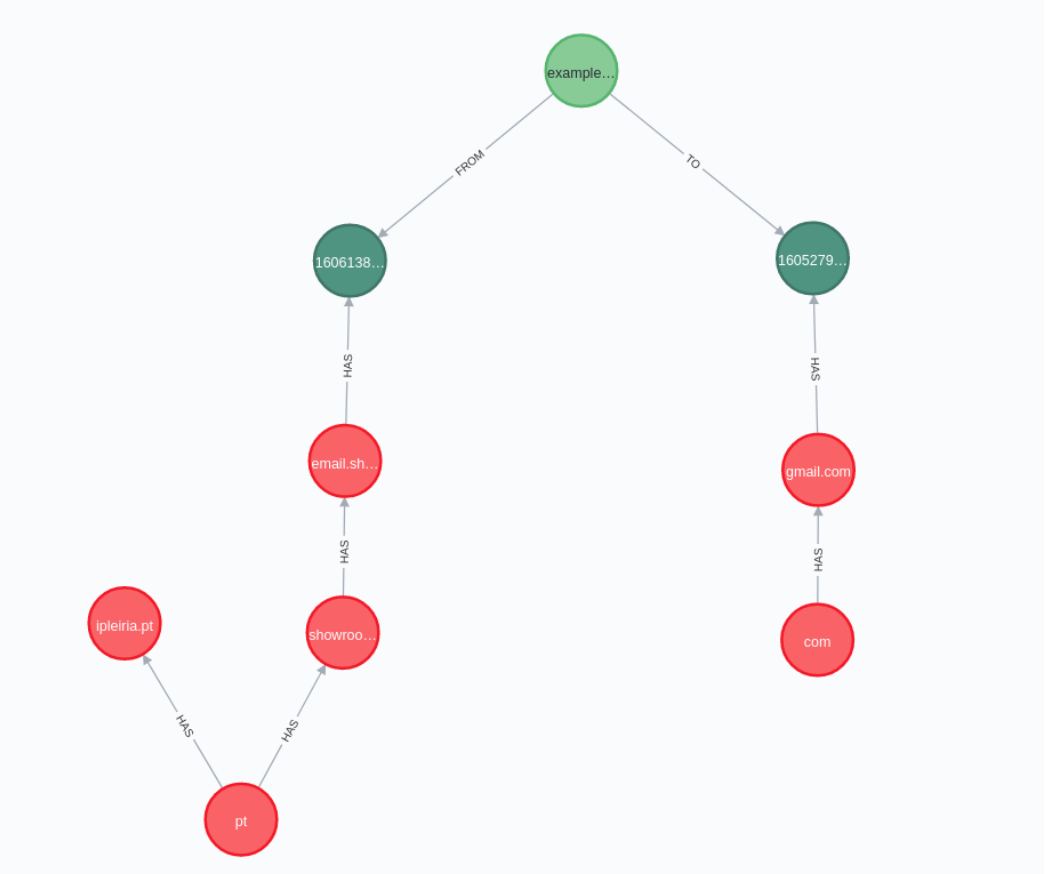


Figura 38: Resultado do e-mail processado na base de dados gŕafica.

.

Na figura 38, o primeiro nó, com verde mais claro, representa o ficheiro processado, agora do tipo «E-mail», que contém informações como o assunto e a mensagem em texto e/ou em HTML. No segundo nível, podemos verificar que o e-mail gerou dois nós do tipo «Account», isto é, as contas de e-mail obtidas dos cabeçalhos «From» e «To», como indicado nas ligações entre os nós. Assim que as contas são criadas, também são criados ou atualizados os domínios dessas contas de e-mail. Por exemplo, ao ser processado a conta de e-mail à esquerda, foi criado o domínio «gmail.com» e o seu pai, o domínico «.com», cujos registos ainda não existiam na plataforma.

4.5.7 *Módulo «entities»*

O módulo de entidades serve para registar de 3 tipos de Entidades: Pessoa, Organização e Empresa. Tem como objetivo guardar informações relevantes sobre as entidades, cruzando-as com outros nós, isto é, definindo relações que existem entre outras entidades ou objetos, para compreender o contexto.

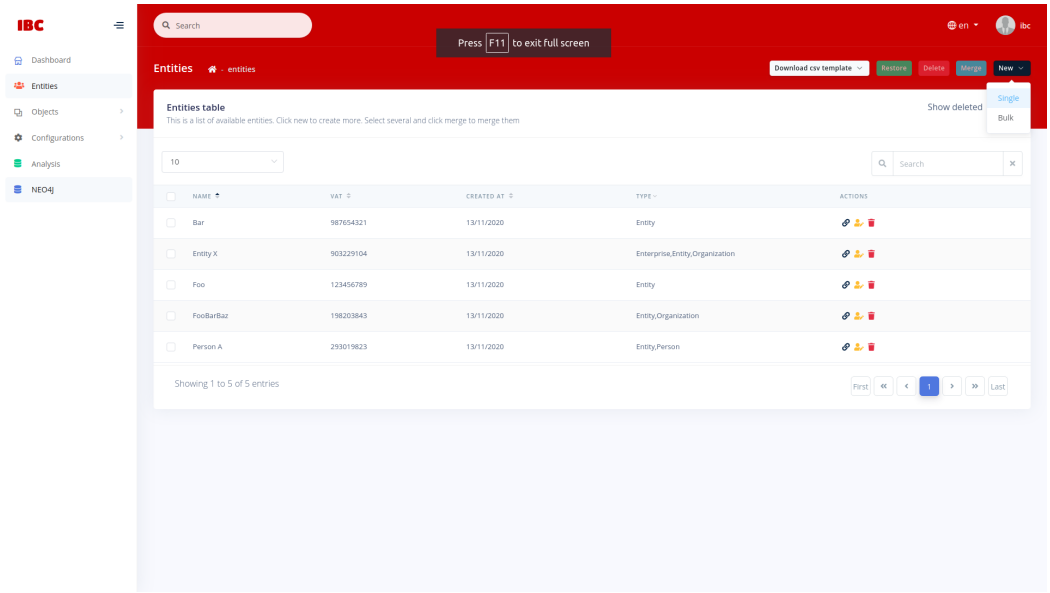


Figura 39: Listagem de entidades

Quase todos os módulos permitem a inserção de novos dados através de dois modos: *single* ou *bulk*, como se pode verificar na figura 39. O modo *single* é o modo normal, que apresenta um formulário para inserção dos dados. O modo *bulk* permite a inserção de novos registos através de um CSV.

Ao escolher o tipo de entidade a criar, o formulário renderiza diferentes parâmetros para o novo registo a ser inserido. Ver figuras 40, 41, 42 e 43.

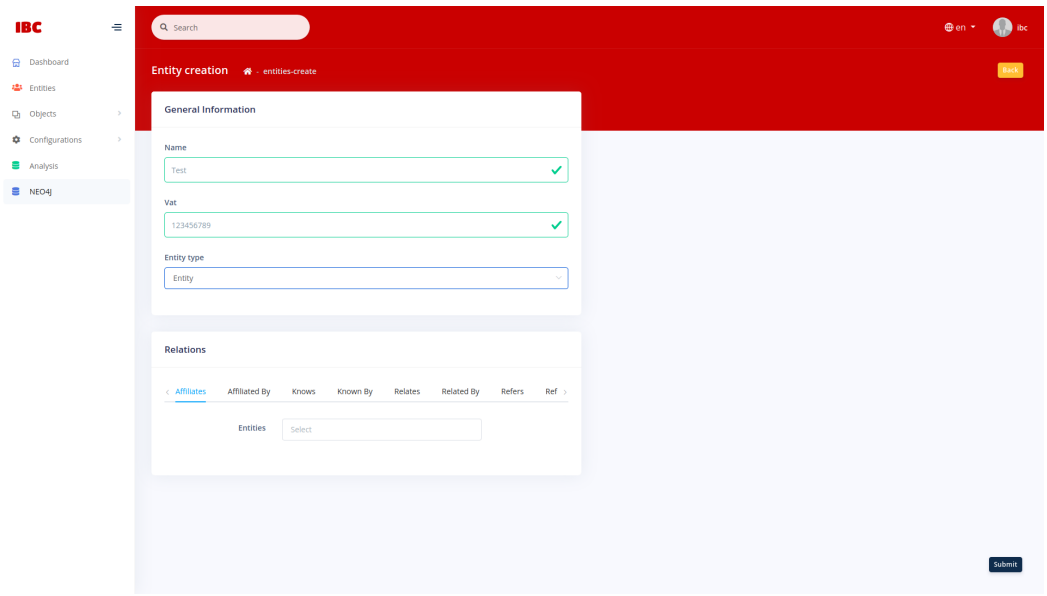


Figura 40: Criação de entidade genérica

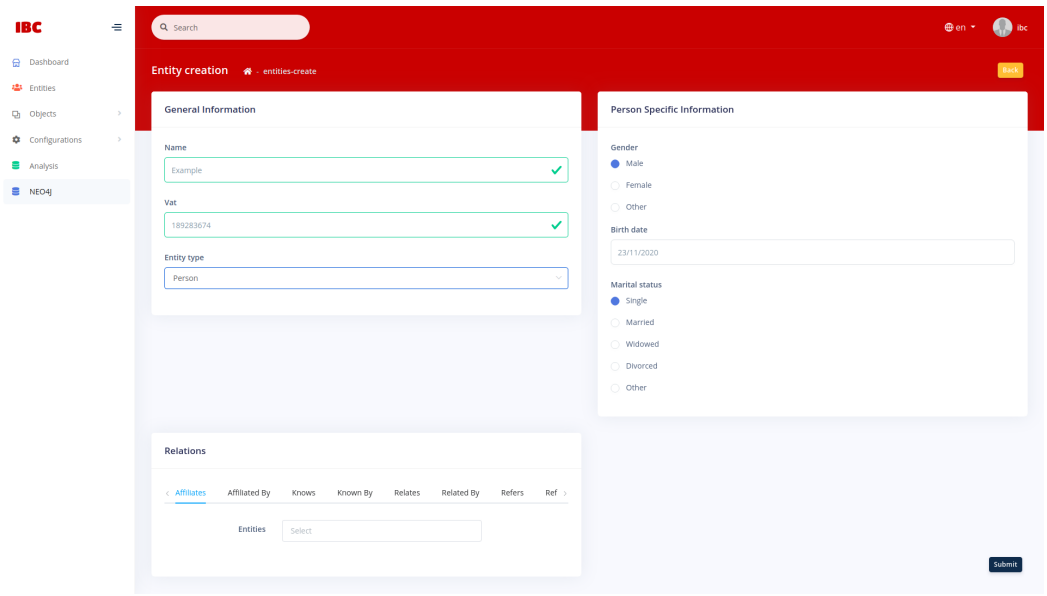


Figura 41: Criação de entidade do tipo Pessoa

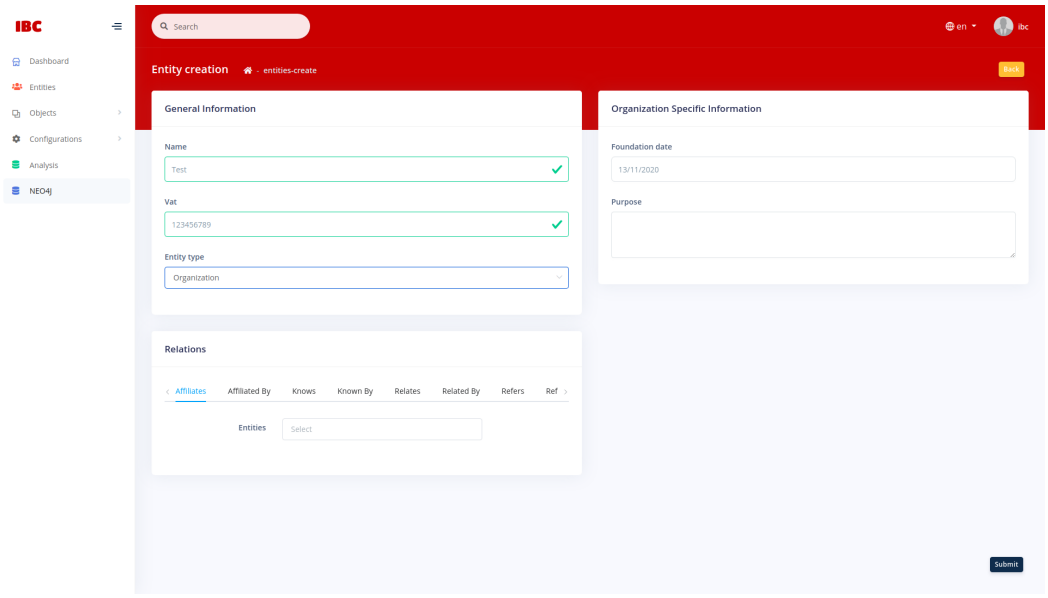


Figura 42: Criação de entidade do tipo Organização

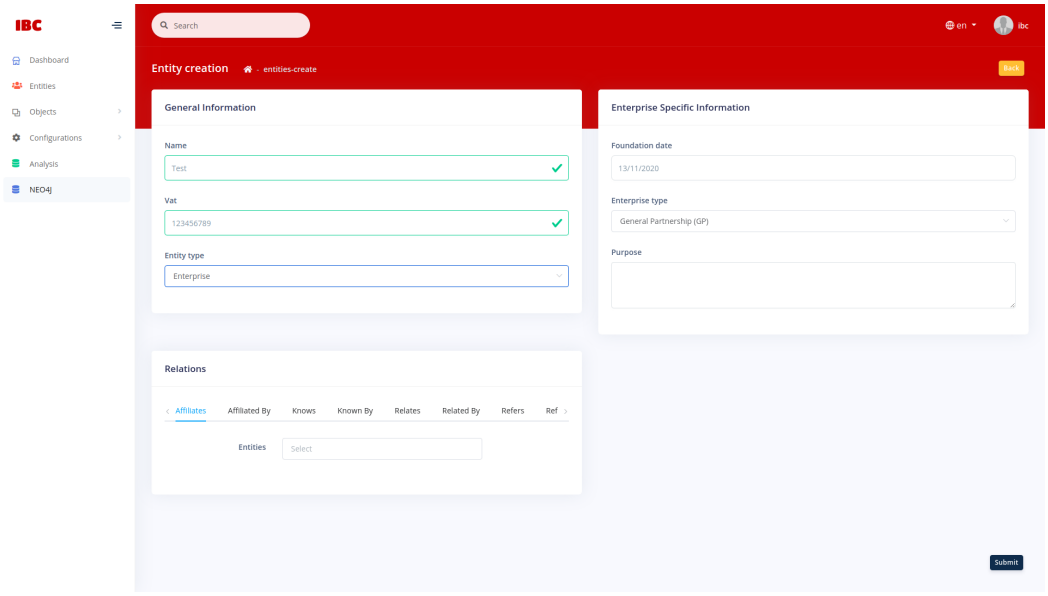


Figura 43: Criação de entidade do tipo Empresa

A diferença entre uma Organização e uma Empresa é pouca. Quando não se conhece toda a informação acerca de uma Empresa, ou esta é uma Organização sem fins lucrativos, deve-se inserir o registo como sendo do tipo "Organização".

Neste módulo é possível converter uma Entidade em qualquer outro tipo: Pessoa, Organização e Empresa. Também é possível converter uma Organização numa Empresa.

4.5.8 *Módulo «files»*

Este módulo permite fazer upload de ficheiros para o servidor e guardar a informação e os metadados dos ficheiros.

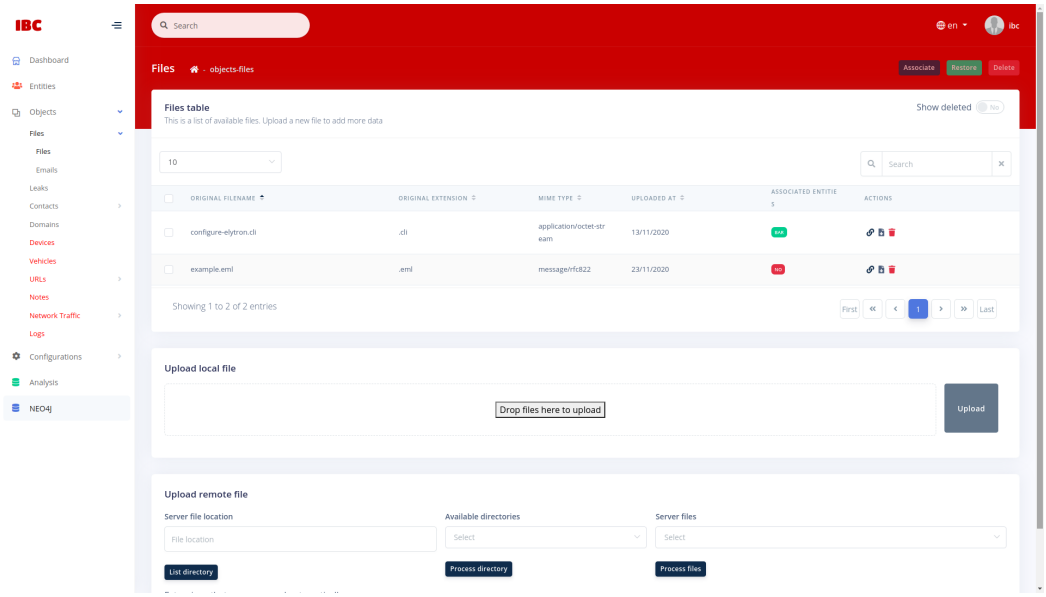


Figura 44: Listagem de ficheiros no servidor, com zona de upload para inserção de novos ficheiros.

Na figura 44, podemos verificar três zonas distintas. A primeira, que mostra a listagem de ficheiros registados no sistema e as duas abaixo, que servem para inserir novos ficheiros na aplicação. Existem duas formas de registar um ficheiro no servidor. A primeira forma, que pode ser efetuada pela segunda zona a branco na imagem, onde é feito o *upload* do ficheiro utilizando o protocolo HTTP. A segunda forma de registar um ficheiro no servidor, é feita através de dois passos.

Quando os ficheiros são relativamente pequenos ou em quantidade baixa, o utilizador pode recorrer ao *upload* pelo site, utilizando o protocolo HTTP. No entanto, este protocolo não foi desenhado para permitir o *upload* de grandes quantidades de dados. O *upload* de um ficheiro grande (por exemplo, acima de 16MB), irá sobrecarregar o servidor aplicacional. Por esta razão, para não sobrecarregarmos o servidor, desenvolveu-se uma alternativa para estes casos. Criou-se uma pasta no servidor, a qual o utilizador tem permissões de escrita, e por onde pode aceder, utilizando o protocolo SSH. O utilizador deve colocar nessa pasta os ficheiros e/ou diretorias que pretende importar para a aplicação. Depois, basta ir ao site escolher os ficheiros a inserir. No site, é possível enumerar os ficheiros e/ou diretorias que

o utilizador coloca nessa pasta partilhada via SSH e selecioná-los para inserir na aplicação.

4.5.9 *Módulo «filesystem»*

O módulo *filesystem* permite ao utilizador ver a estrutura de uma diretoria partilhada, que o cliente tem acesso via SSH. Surgiu esta necessidade, em vez do upload direto, porque a transferência através do protocolo SSH é mais rápida que o protocolo HTTP, para além das limitações de tamanho de ficheiro.

```
daniel@g14 ~ http :8000/filesystem/ $auth
HTTP/1.1 200 OK
Allow: GET, HEAD, OPTIONS
Content-Length: 897
Content-Type: application/json
Date: Sun, 29 Nov 2020 17:14:54 GMT
Referrer-Policy: same-origin
Server: WSGIServer/0.2 CPython/3.8.6
Vary: Accept
X-Content-Type-Options: nosniff
X-Frame-Options: DENY

{
    "dirs": [
        ".org.chromium.Chromium.G7ObwK",
        ".Test-unix",
        "tracker-extract-files.126",
        "systemd-private-eb4f8b9d83ff4df68f5416bc403dfa08-colord.service-PdbWXi",
        ".org.chromium.Chromium.oS9pYJ",
        ".font-unix",
        "systemd-private-eb4f8b9d83ff4df68f5416bc403dfa08-systemd-logind.service-kYVprj",
        ".com.google.Chrome.NCE4mY",
        ".XIM-unix",
        ".ICE-unix",
        "snap.snap-store",
        "systemd-private-eb4f8b9d83ff4df68f5416bc403dfa08-upower.service-Op24cg",
        ".X11-unix",
        "systemd-private-eb4f8b9d83ff4df68f5416bc403dfa08-systemd-resolved.service-ghJMuj",
        "systemd-private-eb4f8b9d83ff4df68f5416bc403dfa08-ModemManager.service-iAx5Ue",
        "systemd-private-eb4f8b9d83ff4df68f5416bc403dfa08-systemd-timesyncd.service-0OobAg",
        "systemd-private-eb4f8b9d83ff4df68f5416bc403dfa08-fwupd.service-5I80zi",
        "systemd-private-eb4f8b9d83ff4df68f5416bc403dfa08-switcheroo-control.service-cT6lZe"
    ],
    "files": [
        ".X1024-lock",
        ".X1025-lock",
        "Cl5GtwfS1RAxx0da"
    ]
}
```

Figura 45: Listagem de ficheiros e diretorias na pasta partilhada, cujo acesso é feito via SSH

Na figura 45, no servidor a pasta partilhada aponta para a pasta temporária «/tmp». Por isso, é possível visualizar as pastas temporárias típicas que se encontram no Sistema Operativo Ubuntu.

Assim que um ficheiro é registado no servidor, o programa vai obter também os metadados respetivos, para obter mais informação pertinente, como por exemplo: a data da última alteração e o nome do utilizador que o criou. Para tal, o programa recorre ao utilitário «ExifTool» [24] e guarda esta informação, atualmente, numa base de dados relacional.

4.5.10 *Módulo «ibc»*

Este módulo é a parte central da aplicação, onde contém as configurações iniciais, e pelo qual a aplicação é iniciada. Não existe um *endpoint* para este módulo. Pelo contrário, é por este módulo que a *framework* Django é iniciada. Através deste módulo são definidas as rotas da aplicação e, deste modo, os pedidos são resolvidos nos respetivos módulos, melhorando a lógica de negócio e mantendo um código sustentável.

4.5.11 *Módulo «leaks»*

Neste módulo permite-se o registo *leaks*. Um *leak* representa uma fuga de informação. Neste caso, quando o utilizador pretende, por exemplo, saber se o seu e-mail e *password* já foram comprometidos, basta inserir na aplicação o seu e-mail, e o site irá procurar por esse e-mail numa API com mais de doze milhões de registos.

A aplicação delega a pesquisa à API da DeHashed. Os resultados obtidos são processados pela aplicação, criando nós do tipo «*Leak*», e criando ou relacionado com entidades, através da criação de contactos.

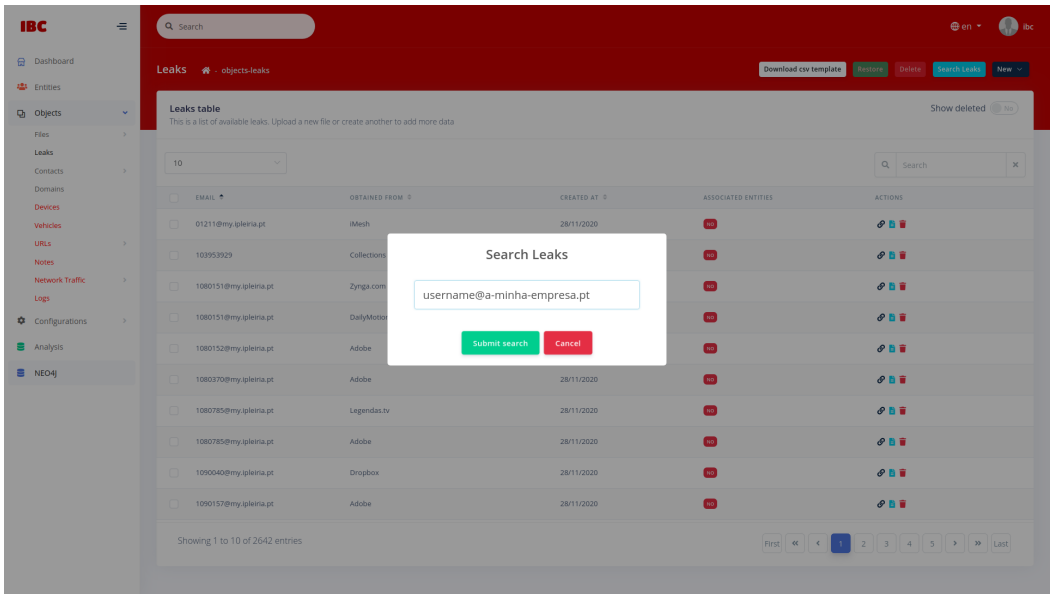


Figura 46: Exemplo de pesquisa de um leak

Na figura 46, está exemplificado a pesquisa de *leaks* pelo termo «username@a-minha-empresa.pt». No entanto, o termo de pesquisa pode ser mais geral. Neste caso, iria procurar especificamente por *leaks* com aquele endereço de e-mail.

A aplicação também permite, com apenas um botão, no módulo «*domains*», procurar por *leaks* de um determinado domínio 47. Supondo o domínio: «o-meu-domínio.pt», o que, internamente, a aplicação faz é a aplicação do filtro de pesquisa: «@o-meu-domínio.pt». Isto é, a aplicação procura por todos os endereços de e-mail relativos ao domínio em pesquisa.

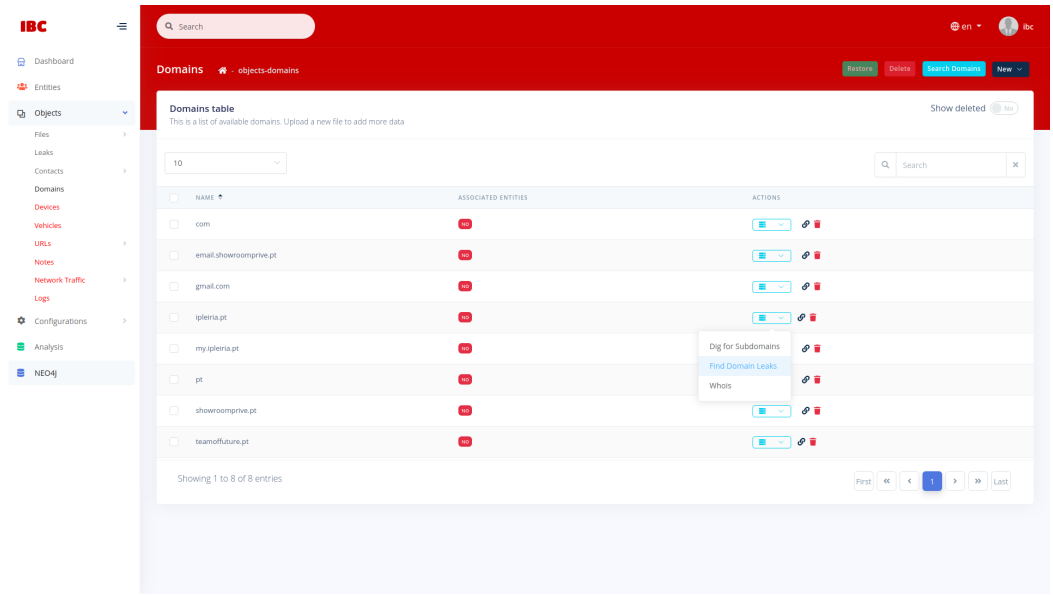


Figura 47: Opção para procurar *leaks* de um domínio.

Este módulo tem um bom potencial para detetar falhas de segurança e verificar se os utilizadores da empresa estão ou não comprometidos, tentando validar as credenciais das contas dos utilizadores das empresas com as passwords que vieram ao público na Internet. As passwords descobertas nos *leaks* podem vir em formato *cleartext* ou cifradas. Em ambos os casos, deve ser recomendado a mudança da palavra-passe, mesmo nos casos em que só é descoberta a palavra-passe cifrada, visto que também existem técnicas de força-bruta para obter a palavra-passe através do *hashing* da palavra-passe.

Poderão aparecer mais campos nos *leaks*, para além do utilizador e da password. Os campos que podem vir são:

- id: id único do leak na base de dados DeHashed;
- email: e-mail do utilizador;
- ip_address: endereço IP do utilizador;
- username: username do utilizador;
- password: password do utilizador (em *cleartext*);
- hashed_password: password cifrada;
- hash_type: tipo de cifra utilizada para cifragem;
- name: nome do utilizador;
- vin: matrícula do veículo do utilizador;
- address: morada do utilizador;
- phone: número de telemóvel do utilizador;
- database_name: sítio onde foi obtido o *leak*;

A título de exemplo, só para os domínios «ipleiria.pt» e «my.ipleiria.pt» foram descobertos mais de dois mil e seiscentos registos. Este assunto é preocupante e serve para alertar a comunidade IPLeiria que é possível obter acesso às contas de muitos docentes e estudantes.

É importante também prestar atenção ao campo *Obtained from* que nos indica de onde surgiu o *leak*. Por exemplo, se o campo *Obtained from* surge da Dropbox, significa que existiu um ataque à companhia no qual foram comprometidas as credenciais que o utilizador tinha para aceder ao site da Dropbox.

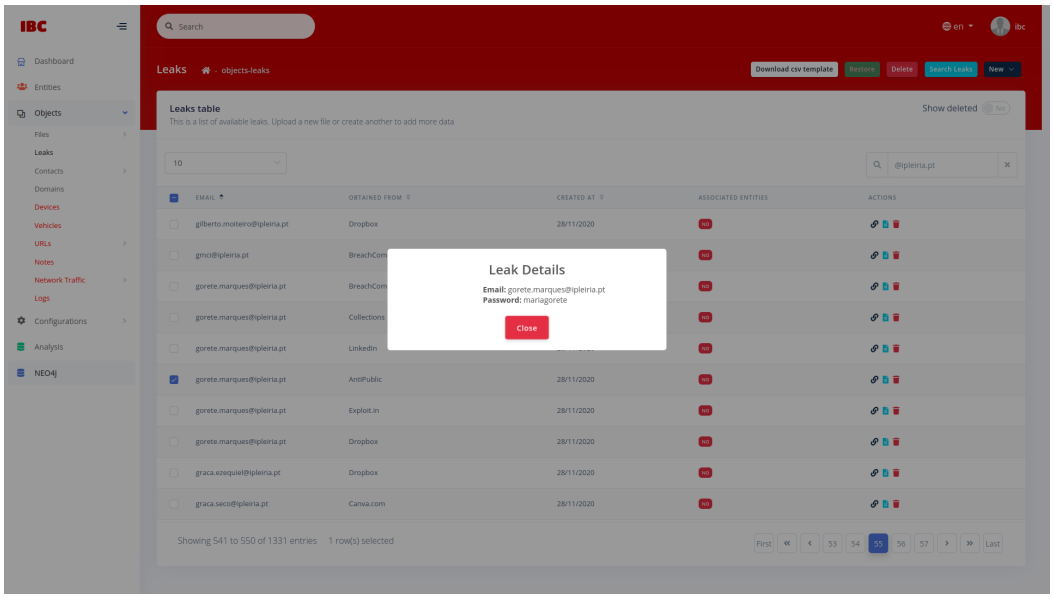


Figura 48: Exemplo de um leak onde a password vem em *cleartext*

Na figura 48 podemos constatar que o utilizador deve utilizar passwords fracas e que são fáceis de adivinhar. Bastando traçar apenas um perfil básico do utilizador, ao saber detalhes como o nome completo, cidade onde nasceu e/ou onde vive, aniversário, nome do animal de estimação, entre outros, podemos comprometer a segurança da informação e obter dados de forma ilícita. Para além do mais, foi possível obter um outro nome desta pessoa através da password descoberta: "Maria". Agora podemos procurar, por exemplo, por um perfil do Facebook que respeite estas condições e encontramos o utilizador nas redes sociais, onde iremos obter alguns dos campos básicos acima mencionados e a rede de amigos mais próxima.

É importante mencionar e consciencializar que os utilizadores com *passwords* fracas tendem a utilizar a mesma password para mais contas noutros sítios, quer por preguiça de criar novas *passwords*, por pressa, por desleixo ou simplesmente por receio de criar *passwords* complicadas que tendem a esquecer. Por isso, ao conhecer agora melhor o utilizador, podemos utilizar técnicas de força-bruta para combinar *passwords* com a informação básica que foi obtida das redes sociais e, conhecendo um e-mail pessoal deste utilizador, é possível efetuar um ataque que roube a identidade ou os dados do indivíduo.

Por fim, este módulo poderá ser também integrado no futuro com um módulo de alarmística que deteta estes casos e notifica de imediato o utilizador que as suas credenciais foram comprometidas e, portanto, deve alterar o quanto antes. Poderá também suspender o acesso desse utilizador até que a password seja alterada pelo

mesmo, num computador fidedigno (por exemplo, um computador dentro da rede da empresa).

4.5.12 *Módulo «objects»*

Este módulo define a hierarquia base de todos os nós do tipo «*Object*». Trata-se de um módulo que permite organizar, como um container, todos os nós do tipo Objeto. Assim sendo, um pedido com o verbo GET à rota «/objects» irá pedir todos os nós com a etiqueta «*Object*», o que é muito útil quando se pretende generalizar um pedido que encontre um ou mais objetos em BD, consoante determinados filtros de pesquisa.

4.5.13 *Módulo «social»*

Este módulo ainda não foi desenvolvido. Pretende, à semelhança do módulo «*leaks*» ter um conjunto de APIs das quais obtém informação sobre redes sociais e contactos. Neste módulo pretender-se-á descobrir mais informações sobre pessoas e eventos, relacionando-os entre si. Ao obter os detalhes básicos e a rede de relacionamentos da pessoa, bem como os seus gostos, é possível recolher uma informação mais precisa e traçar um perfil, que poderá ser utilizado para extrair mais conhecimento sobre o indivíduo e possíveis ameaças à cibersegurança.

5

CONCLUSÕES

Neste relatório foi apresentado o trabalho de Mestrado em Cibersegurança e Informática Forense, desenvolvido no decurso da realização do projecto Intelligence Business Center. Nele foi realizada uma aplicação web e um painel de administração. No seu desenvolvimento surgiram algumas dificuldades, apresentadas na secção 5.1. A secção 5.2 apresenta de forma detalhada o trabalho desenvolvido e a secção 5.3, os trabalhos a desenvolver no futuro.

5.1 DIFICULDADES SENTIDAS

No decorrer do projeto, foram encontradas diversas dificuldades no seu desenvolvimento, mas que, ainda assim, não impediram o sucesso do produto final. Destacar-se-ão, de seguida, as maiores dificuldades que atrasaram o projeto.

A primeira dificuldade sentida foi perceber concretamente o que o cliente quer. É bastante comum no mundo da informática os clientes não saberem em concreto o que pretendem, ou a melhor forma de expressar o que gostariam de adquirir. Por vezes, ao fim de várias reuniões de planeamento, e iniciando-se o desenvolvimento, percebe-se que o cliente quis dizer com «A» significa «B», e recomeça-se de novo. No caso da empresa que adquiriu o software e ainda hoje financia o projeto, não chegou a este extremo. No entanto, houve fases que foi preciso descobrir o caminho meio "às cegas", numa tentativa de perceber se o que o cliente quis dizer era de facto o que foi implementado. A alegria da equipa ao ter apostado no Neo4j, por exemplo, deu-se devido ao cliente, quando ele viu um gráfico de nós e relações e propriedades em ambos, porque exclamou: «É isto mesmo que eu queria!». Para a empresa foi um passo bastante positivo ao compreender que foi feita uma boa seleção de tecnologias a utilizar, e que o tempo "perdido"na análise e comparação de base de dados foi produtiva.

A segunda dificuldade encontrada foi escolher as ferramentas certas para o desenvolvimento da *framework*. Para cada desafio existem imensas opções de tecnologias que prometem ou resolvem o problema. No entanto, é preciso fazer muito estudo

comparativo de bibliotecas e *frameworks* para compreender a que melhor se aplica a cada circunstância, como foi o caso da eleição das linguagens de programação 1, 3.2.1.1, 3.2.2. Aconteceu por diversas vezes desenvolver a aplicação utilizando uma tecnologia e, mais tarde, perceber-se que a tecnologia escolhida não foi a melhor opção porque encontrava-se com algumas limitações. Nesses casos, era necessário recomeçar do zero, utilizando uma nova abordagem.

Também surgiram dificuldades em implementar um módulo de redes sociais, pelo que, ao iniciar-se o desenvolvimento e verificar os obstáculos, decidiu-se passar este módulo para longo-prazo.

O módulo de e-mails não faz um processamento ainda totalmente eficaz, visto que um e-mail pode ser guardado em vários formatos e, dentro do mesmo formato, existem várias formas de guardar a mesma informação. Por vezes ainda surgem *bugs* em alguns e-mails específicos. O protocolo de e-mail mais bem implementado pela aplicação é o «eml». Este formato «eml» é um *standard* para guardar e ler e-mails, pelo que o Python já possui uma biblioteca nativa que permite-os processar de forma eficaz. Os restantes formatos de e-mail suportados, «msg» e «pst» encontram-se com algumas limitações. São utilizadas bibliotecas externas que tentam extrair e processar estes tipos de e-mail, mas existirão sempre limitações enquanto forem protocolos fechados pela Microsoft. Ir-se-á analisar esta situação novamente no futuro, procurando por bibliotecas pagas com um suporte melhor.

Por fim, o projeto foi idealizado para ser realizado por dois colaboradores, um especializado na aplicação em si, a parte da API, ou seja, responsável pelo *backend* e outro elemento responsável pela UI, isto é, pelo *frontend*. No entanto, mais para a parte final do projeto, o colaborador responsável pela parte do *frontend* foi-se embora, o que causou um atraso no desenvolvimento da aplicação, pois toda a aplicação passou a ser desenvolvida apenas por um colaborador, que teve de dedicar algum tempo a compreender o código deixado pelo antigo colaborador.

5.2 RESULTADOS OBTIDOS

Para avaliar o projeto e os resultados obtidos, a melhor métrica que podemos ter reflete-se na satisfação do nosso cliente. Até à data o cliente sempre ficou satisfeito com o projeto. Foi-se desenvolvendo módulo a módulo, até atingir o resultado atual. Sempre que um módulo se encontrava minimamente desenvolvido e estável, era feita uma reunião para apresentar o novo módulo ao cliente.

Atualmente a aplicação já tem um conjunto básico de módulos interessantes e prontos a utilizar em produção, que o cliente já pode usufruir, como é o caso do registo de entidades, domínios, e-mails e *leaks*. No momento presente, a aplicação também já conta com um módulo de análise de dados gráfica, que, embora ainda esteja um pouco cru, já foi apresentado e aprovado pelo utilizador, sendo esse o foco atualmente a curto-prazo, para além da resolução contínua de bugs.

Os resultados obtidos foram positivos: O cliente já utiliza a aplicação para registar informação sobre a sua empresa e de empresas concorrentes, onde analisa a informação que se relaciona. Também já utiliza o módulo de domínios para pesquisar subdomínios na sua rede que podem cair no esquecimento e inclusive domínios de empresas que pretende explorar. No que toca ao processamento de e-mails, alguns e-mails têm sido processados mas ainda pouco foi explorado neste campo, principalmente porque o cliente utiliza contas da Microsoft, e ainda existem algumas limitações na biblioteca utilizada pelo servidor, visto que a implementação do formato de e-mail «.msg» é um protocolo desconhecido e fechado. No módulo de *leaks*, o cliente procura por vazamentos de informação relativos às contas dos seus colaboradores, principalmente com o comprometimento da password. Embora todos os dados sejam anónimos e não é permitido revelar a identidade do cliente, é possível comprovar a eficácia da utilização em produção e a utilidade da aplicação, mostrando apenas as estatísticas gerais de nós que o cliente já criou:

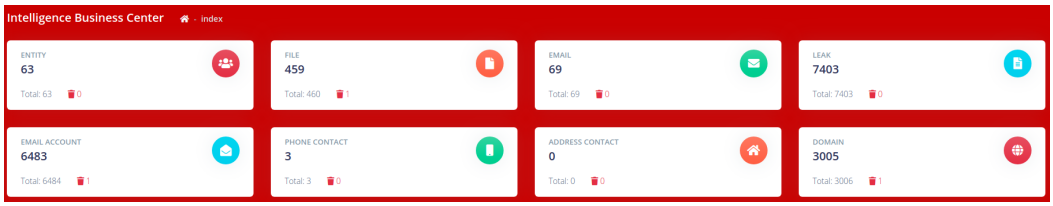


Figura 49: Estatística básica de número de nós por tipo de nó, no servidor em produção

Como podemos verificar pela figura 49, já existem milhares de nós criados na aplicação pelo utilizador, que utiliza para analisar e obter informação sobre a sua empresa e outras entidades concorrentes.

5.3 TRABALHO FUTURO

A seguir ao módulo de análise, a curto-prazo espera-se implementar um módulo para análise do tráfego de rede, se o cliente concordar que esta deveria ser a próxima ordem de trabalhos. Tudo poderá variar, dependendo das reuniões onde serão

discutidas as ideias com o cliente. O *Product Owner* 3.1.1.4 é que deve definir as prioridades na lista de funcionalidades do 3.1.1.6.

Por fim, a longo-prazo pretende-se cumprir com os objetivos todos propostos no início do documento 1.2. Neste sentido, é desejável que o cliente aceite ceder um tempo de investigação para a aprendizagem prática de técnicas de Aprendizagem Computacional, exploradas teoricamente no presente relatório 2.3, para as aplicar na aplicação.

BIBLIOGRAFIA

[1] aboul3la. *Sublist3r*. Nov. de 2020. URL: https://github.com/aboul3la/Sublist3r (acedido em 07/11/2020).

[2] Pekka Abrahamsson et al. «Agile software development methods: Review and analysis». Em: *arXiv preprint arXiv:1709.08439* (2017).

[3] Renzo Angles. «A comparison of current graph database models». Em: *2012 IEEE 28th International Conference on Data Engineering Workshops*. IEEE. 2012, pp. 171–177.

[4] Renzo Angles e Claudio Gutierrez. «Survey of graph database models». Em: *ACM Computing Surveys (CSUR)* 40.1 (2008), pp. 1–39.

[5] Rebecca Gurley Bace e Peter Mell. *Intrusion detection systems*. 2001.

[6] Kent Beck et al. *Manifesto for Agile Software Development*. 2001. URL: http://www.agilemanifesto.org/.

[7] Guilford Cantave. *Wireless perimeter intrusion detection system*. US Patent 9,330,548. Mai. de 2016.

[8] Donald D Chamberlin et al. «A history and evaluation of System R». Em: *Communications of the ACM* 24.10 (1981), pp. 632–646.

[9] Peichao Chen, Citian You e Panfeng Ding. «Event classification using improved salp swarm algorithm based probabilistic neural network in fiber-optic perimeter intrusion detection system». Em: *Optical Fiber Technology* 56 (2020), p. 102182.

[10] Kendra Cherry. *What Is Operant Conditioning and How Does It Work?* URL: https://www.verywellmind.com/operant-conditioning-a2-2794863 (acedido em 14/12/2019).

[11] Alexandre Chopin, Sebastien Chopin e Pooya Parsa. *Nuxt.js - The Intuitive Vue Framework*. URL: https://nuxtjs.org/ (acedido em 02/08/2019).

[12] Alistair Cockburn e Jim Highsmith. «Agile software development, the people factor». Em: *Computer* 34.11 (2001), pp. 131–133.

[13] D. Crocker. *Internet Message Format*. RFC 822. Ago. de 1982. URL: https://rfc-editor.org/rfc/rfc822.txt.

[14] Leslie Daigle. «WHOIS protocol specification». Em: *RFC 3912 (Draft Standard)* (set. de 2004). [accessed at July 13, 2020]. URL: https://tools.ietf.org/html/rfc3912.

[15] Hervé Debar, Marc Dacier e Andreas Wespi. «Towards a taxonomy of intrusion-detection systems». Em: *Computer networks* 31.8 (1999), pp. 805–822.

[16] Thomas G Dietterich. «Ensemble methods in machine learning». Em: *International workshop on multiple classifier systems*. Springer. 2000, pp. 1–15.

[17] Schahram Dustdar e Wolfgang Schreiner. «A survey on web services composition». Em: *International journal of web and grid services* 1.1 (2005), pp. 1–30.

[18] Robin Edwards e Devin Barry. *django_neomodel*. URL: https://github.com/neo4j-contrib/django-neomodel/ (acedido em 05/06/2019).

[19] Roberto Gil Espinha. *O que é o Kanban: Guia completo*. URL: https://artia.com/kanban/ (acedido em 29/11/2020).

[20] Martin Fowler, Jim Highsmith et al. «The agile manifesto». Em: *Software Development* 9.8 (2001), pp. 28–35.

[21] Tal Garfinkel, Mendel Rosenblum et al. «A virtual machine introspection based architecture for intrusion detection.» Em: *Ndss*. Vol. 3. 2003. Citeseer. 2003, pp. 191–206.

[22] Amin Ghafouri et al. «Optimal thresholds for anomaly-based intrusion detection in dynamical environments». Em: *International Conference on Decision and Game Theory for Security*. Springer. 2016, pp. 415–434.

[23] Jing Han et al. «Survey on NoSQL database». Em: *2011 6th international conference on pervasive computing and applications*. IEEE. 2011, pp. 363–366.

[24] Phil Harvey. *ExifTool*. Nov. de 2003. URL: https://exiftool.org/ (acedido em 10/11/2020).

[25] Robert A Hedin e Alfiero F Balzano. *Laser perimeter intrusion detection system*. US Patent 3,623,057. Nov. de 1971.

[26] Richard Hull e Roger King. «Semantic database modeling: Survey, applications, and research issues». Em: *ACM Computing Surveys (CSUR)* 19.3 (1987), pp. 201–260.

[27] Walter L Hürsch e Cristina Videira Lopes. «Separation of concerns». Em: (1995).

[28] Christine Johnson e Richard Swinbank. «Medium-range multimodel ensemble combination and calibration». Em: *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography* 135.640 (2009), pp. 777–794.

[29] Ian Jolliffe. «Principal component analysis». Em: *International encyclopedia of statistical science*. Springer, 2011, pp. 1094–1096.

[30] Wolfgang Kainz. «Fuzzy logic and GIS». Em: *Vienna: University of Vienna* (2007).

[31] Rohit kumar Kaliyar. «Graph databases: A survey». Em: *International Conference on Computing, Communication & Automation*. IEEE. 2015, pp. 785–790.

[32] Kanbanize. *Primeiros Passos com Kanban*. URL: https://kanbanize.com/pt/recursos-kanban/primeiros-passos (acedido em 29/11/2020).

[33] G Jaspher Kathrine, Ronnie T Baby e V Ebenzer. «COMPARATIVE ANALYSIS OF SUBDOMAIN ENUMERATION TOOLS AND STATIC CODE ANALYSIS». Em: ().

[34] Mike Keith e Merrick Schnicariol. «Object-relational mapping». Em: *Pro JPA 2*. Springer, 2009, pp. 69–106.

[35] Muhammad Waqas Khan e Eram Abbasi. «Differentiating Parameters for Selecting Simple Object Access Protocol (SOAP) vs. Representational State Transfer (REST) Based Architecture». Em: *Journal of Advances in Computer Networks* 3.1 (2015), pp. 63–6.

[36] P. Leach, M. Mealling e R. Salz. «RFC 4122: A Universally Unique IDentifier (UUID) URN Namespace». Em: (jul. de 2005). [accessed at July 14, 2020]. URL: https://tools.ietf.org/html/rfc4122.

[37] Kai Lei, Yining Ma e Zhi Tan. «Performance comparison and evaluation of web development technologies in php, python, and node. js». Em: *2014 IEEE 17th international conference on computational science and engineering*. IEEE. 2014, pp. 661–668.

[38] Suqi Liu et al. «Who is. com? Learning to parse WHOIS records». Em: *Proceedings of the 2015 Internet Measurement Conference*. 2015, pp. 369–380.

[39] Ronald van Loon. *Machine learning explained: Understanding supervised, unsupervised, and reinforcement learning*. Fev. de 2018. URL: https://bigdata-madesimple.com/machine-learning-explained-understanding-supervised-unsupervised-and-reinforcement-learning/ (acedido em 27/07/2019).

[40] Fedele Mantuano. *Mail Parser*. URL: https://github.com/SpamScope/mail-parser (acedido em 03/07/2020).

[41] Weizhi Meng et al. «When intrusion detection meets blockchain technology: a review». Em: *Ieee Access* 6 (2018), pp. 10179–10188.

[42] Joachim Metz. *libpff*. URL: https://github.com/libyal/libpff (acedido em 03/07/2020).

[43] David Miller et al. *Security information and event management (SIEM) implementation*. McGraw-Hill, 2011.

[44] Frederic P Miller, Agnes F Vandome e John McBrewster. *Active record pattern*. Alpha Press, 2010.

[45] Mohamed A Mohamed, Obay G Altrafi e Mohammed O Ismail. «Relational vs. nosql databases: A survey». Em: *International Journal of Computer and Information Technology* 3.03 (2014), pp. 598–601.

[46] Harris Nathel. *Introduction to Bayes' Theorem*. [accessed at November 21, 2019]. Ago. de 2019. URL: https://medium.com/@hmnathel/intro-to-bayes-theorem-8378648337fa.

[47] Neo4j. *Cypher Query Language*. URL: https://neo4j.com/developer/cypher/ (acedido em 15/09/2019).

[48] David Opitz e Richard Maclin. «Popular ensemble methods: An empirical study». Em: *Journal of artificial intelligence research* 11 (1999), pp. 169–198.

[49] Barry Overeem. *Refresh the purpose of the Scrum Framework*. [accessed July 9, 2020]. Jun. de 2019. URL: https://medium.com/the-liberators/refresh-the-purpose-of-the-scrum-framework-9e4bceb25499.

[50] OWASP. *Amass*. Nov. de 2020. URL: https://github.com/OWASP/Amass (acedido em 07/11/2020).

[51] PostgreSQL. *JSON Types (Version 12)*. [accessed at July 14, 2020]. URL: https://www.postgresql.org/docs/12/datatype-json.html.

[52] projectdiscovery. *Subfinder*. Nov. de 2020. URL: https://github.com/projectdiscovery/subfinder (acedido em 07/11/2020).

[53] Rizdqi Akbar Ramadhan, Redho Maland Aresta e Dedy Hariyadi. «Sudomy: Information Gathering Tools for Subdomain Enumeration and Analysis». Em: *MS&E* 771.1 (2020), p. 012019.

[54] Pete Resnick. *Internet Message Format*. RFC 2822. Abr. de 2001. URL: https://rfc-editor.org/rfc/rfc2822.txt.

[55] Pete Resnick. *Internet Message Format*. RFC 5322. Out. de 2008. URL: https://rfc-editor.org/rfc/rfc5322.txt.

[56] Ken Schwaber e Mike Beedle. *Agile software development with Scrum*. Vol. 1. Prentice Hall Upper Saddle River, 2002.

[57] Ken Schwaber e Jeff Sutherland. «The Scrum Guide™». Em: (nov. de 2017). [accessed at July 13, 2020]. URL: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf.

[58] Screetsec. *Sudomy*. Nov. de 2020. URL: https://github.com/Screetsec/Sudomy (acedido em 07/11/2020).

[59] *Scrum*. [accessed at July 30, 2019]. URL: https://www.scrum.org/.

[60] Amanda JC Sharkey et al. «The "test and select" approach to ensemble combination». Em: *International Workshop on Multiple Classifier Systems*. Springer. 2000, pp. 30–44.

[61] Catarina Silva e Bernardete Ribeiro. *Aprendizagem Computacional*. Imprensa da Universidade de Coimbra, jan. de 2018, p. 288. ISBN: 978-989-26-1507-3.

[62] Wilamis Kleiton Nunes da Silva e Araken de Medeiros Santos. «Estratégias de construções de comitês de classificadores multirrótulos no aprendizado semissupervisionado multidescrição». Em: *Revista de Informática Teórica e Aplicada* 24.2 (2017), pp. 71–100.

[63] Ed. T. Bray. *The JavaScript Object Notation (JSON) Data Interchange Format*. [accessed at July 14, 2020]. Dez. de 2017. URL: https://tools.ietf.org/html/std90.

[64] Ed. T. Bray. «The JavaScript Object Notation (JSON) Data Interchange Format». Em: (dez. de 2017). [accessed at July 14, 2020]. URL: https://tools.ietf.org/html/rfc8259.

[65] Emmanouil Vasilomanolakis et al. «Taxonomy and survey of collaborative intrusion detection». Em: *ACM Computing Surveys (CSUR)* 47.4 (2015), pp. 1–33.

[66] Dave Vernon e Laura Willis. *neomodel*. URL: https://github.com/neo4j-contrib/neomodel/ (acedido em 05/06/2019).

[67] Bradford W Wade e Donald D Chamberlin. «IBM relational database systems: The early years». Em: *IEEE Annals of the History of Computing* 34.4 (2012), pp. 38–48.

[68] *What is a Product Backlog?* [accessed at July 30, 2019]. URL: https://www.scrum.org/resources/what-is-a-product-backlog.

[69] *What is a Sprint Backlog?* [accessed at July 30, 2019]. URL: https://www.scrum.org/resources/what-is-a-sprint-backlog.

[70] Evan You. *Vue.js*. Fev. de 2014. URL: https://vuejs.org/.

[71] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

APÊNDICES

A

APÊNDICE A

A.1 LISTA DE REQUISITOS DO *backend*

A lista que se segue é copiada no ficheiro «*requirements.txt*», um ficheiro onde indica todas as dependências, indicando a versão, utilizadas pela nossa aplicação.

aiohttp==3.6.2

alabaster==0.7.12

amqp==5.0.1

asgiref==3.2.10

astroid==2.4.2

async-timeout==3.0.1

attrs==20.2.0

Babel==2.8.0

beautifulsoup4==4.9.1

billiard==3.6.3.0

blessed==1.17.10

blis==0.7.1

cached-property==1.5.2

catalogue==2.0.1

celery==5.0.2

certifi==2020.6.20

cffi==1.14.3

chardet==3.0.4

click==7.1.2

click-didyoumean==0.0.3

click-log==0.3.2

click-repl==0.1.6

colorama==0.4.3

cryptography==3.1.1

cymem==2.0.3

defusedxml==0.6.0

diff-match-patch==20200713

Django==3.1.2

django-appconf==1.0.4

django-constance==2.7.0

django-cors-headers==3.5.0

django-environ==0.4.5

django-extensions==3.0.9

django-flat-responsive==2.0

django-flat-theme==1.1.4

django-neomodel==0.0.4

djangorestframework==3.11.1

djangorestframework-recursive==0.1.2

djangorestframework-simplejwt==4.3.0

django-rest-passwordreset==1.1.0

django-seed==0.2.2

dnspython==2.0.0

docutils==0.16

enlighten==1.6.2

enum-compat==0.0.3

et-xmlfile==1.0.1

Faker==4.1.3

filetype==1.0.7

flower==0.9.5

future==0.18.2

humanfriendly==8.2

humanize==2.6.0

idna==2.10

imagesize==1.2.0

importlib-metadata==2.0.0

inflection==0.5.1

iniconfig==1.0.1

ipaddress==1.0.23

isort==5.5.3

jdcal==1.4.1

Jinja2==2.11.2

jsonschema==3.2.0

kombu==5.0.2

lazy-object-proxy==1.5.1

libpff-python-ratom==20200808

libratom==0.4.3

mail-parser==3.12.0

MarkupPy==1.14

MarkupSafe==1.1.1

mccabe==0.6.1

more-itertools==8.5.0

multidict==4.7.6

munch==2.5.0

murmurhash==1.0.2

neo4j==4.1.1

neo4j-driver==4.1.1

neobolt==1.7.17

neomodel==4.0.0

neotime==1.7.4

numpy==1.19.2

oauthlib==3.1.0

odfpy==1.4.1

olefile==0.46

openpyxl==3.0.5

packaging==20.4

pbr==5.5.0

pkg-resources==0.0.0

plac==1.2.0

pluggy==0.13.1

preshed==3.0.2

prometheus-client==0.8.0

prompt-toolkit==3.0.7

psutil==5.7.2

psycopg2-binary==2.8.6

py==1.9.0

pycparser==2.20

PyExifTool==0.1.1

Pygments==2.7.1

PyJWT==1.7.1

pylint==2.6.0

pyparsing==2.4.7

pyrsistent==0.17.3

pytest==6.0.1

python3-nmap==1.4.8

python3-openid==3.2.0

python-dateutil==2.8.1

python-magic==0.4.18

python-whois==0.7.3

pytz==2020.1

PyYAML==5.3.1

rarfile==4.0

redis==3.5.3

requests==2.24.0

requests-oauthlib==1.3.0

rope==0.17.0

Shapely==1.7.1

simplejson==3.17.2

six==1.15.0

snowballstemmer==2.0.0

socialscan==1.3.0

soupsieve==2.0.1

spacy==2.3.2

Sphinx==3.2.1

sphinxcontrib-applehelp==1.0.2

sphinxcontrib-devhelp==1.0.2

sphinxcontrib-htmlhelp==1.0.3

sphinxcontrib-jsmath==1.0.1

sphinxcontrib-qthelp==1.0.3

sphinxcontrib-serializinghtml==1.1.4

sphinx-rtd-theme==0.5.0

SQLAlchemy==1.3.19

sqlparse==0.3.1

srsly==2.2.0

striprtf==0.0.10

tablib==2.0.0

tabulate==0.8.7

text-unidecode==1.3

thinc==7.4.1

toml==0.10.1

tornado==6.0.4

tqdm==4.49.0

treelib==1.6.1

urllib3==1.25.10

vine==5.0.0

wasabi==0.8.0

wcwidth==0.2.5

wrapt==1.12.1

xlrd==1.2.0

xlwt==1.3.0

yarl==1.6.0

zipp==3.2.0

B

APÊNDICE B

B.1 LISTA DE REQUISITOS DO *frontend*

O código JSON [63, 64] que se apresenta de seguida, é a cópia do ficheiro «*package.json*», que contém as dependências da aplicação (ver: *dependencies*) e as dependências para o desenvolvimento (ver: *devDependencies*).

```
{
  "name": "ibc",
  "version": "1.0.0",
  "description": "IBC platform develop in nuxt",
  "author": "Icingslice, Lda",
  "private": true,
  "scripts": {
    "dev": "nuxt",
    "build": "nuxt build",
    "start": "nuxt start",
    "generate": "nuxt generate"
  },
  "dependencies": {
    "@fullcalendar/core": "^5.3.1",
    "@fullcalendar/daygrid": "^5.3.2",
    "@fullcalendar/interaction": "^5.3.1",
    "@fullcalendar/timegrid": "^5.3.1",
    "@fullcalendar/vue": "^5.3.1",
    "@nuxtjs/axios": "^5.12.2",
    "@nuxtjs/pwa": "^3.0.2",
    "@nuxtjs/toast": "^3.3.1",
    "@syncfusion/ej2-vue-filemanager": "^18.2.57",
    "axios-auth-refresh": "^3.0.0",
    "bootstrap": "^4.5.2",
    "chart.js": "^2.9.3",
```

```
    "cookieparser": "^0.1.0",
    "d3": "^6.1.1",
    "d3-force": "^2.1.1",
    "datamaps": "^0.5.9",
    "date-fns": "^2.16.1",
    "dotenv": "^8.2.0",
    "dropzone": "5.7.2",
    "element-ui": "^2.13.2",
    "es6-promise": "^4.2.8",
    "flatpickr": "^4.6.6",
    "fuse.js": "^6.4.1",
    "google-maps": "^4.3.2",
    "js-cookie": "^2.2.1",
    "moment": "^2.28.0",
    "neovis.js": "^1.5.0",
    "nouislider": "^14.6.2",
    "nuxt": "^2.14.5",
    "perfect-scrollbar": "^1.5.0",
    "quill": "^1.3.7",
    "sweetalert2": "^10.1.0",
    "v-tooltip": "^2.0.3",
    "vee-validate": "^3.3.11",
    "vue": "^2.6.12",
    "vue-chartjs": "^3.5.1",
    "vue-clipboard2": "^0.3.1",
    "vue-flatpickr-component": "^8.1.6",
    "vue-i18n": "^8.21.1",
    "vue2-transitions": "^0.3.0"
  },
  "resolutions": {
    "@babel/preset-env": "^7.8.7"
  },
  "devDependencies": {
    "@babel/core": "^7.11.6",
    "@nuxt/types": "^2.14.5",
    "babel-plugin-component": "^1.1.1",
    "cross-env": "^7.0.2",
```

```
63      "node-sass": "^4.14.1",
64      "nodemon": "^2.0.4",
65      "sass-loader": "^10.0.2"
66    }
67  }
```

D E C L A R A Ç Ã O

Declaro, sob compromisso de honra, que o trabalho apresentado neste relatório, com o título *"Intelligence Business Center"*, é original e foi realizado por Estudante Daniel Mendes Pinto (2170104) sob orientação de Professor Doutor Filipe Neves (fneves@ipleiria.pt).

Leiria, dezembro de 2020

Estudante Daniel Mendes Pinto