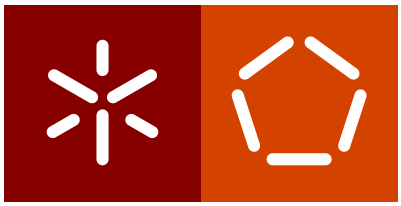


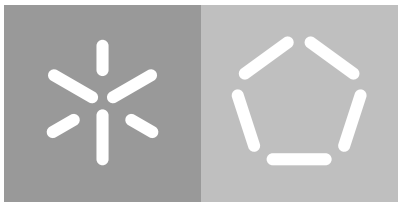
Universidade do Minho
Escola de Engenharia
Departamento de Informática

José Carlos Lima Martins

CLAV:
API de dados e Autenticação

Relatório de Pré-Dissertação

December 2019



Universidade do Minho
Escola de Engenharia
Departamento de Informática

José Carlos Lima Martins

CLAV:
API de dados e Autenticação

Relatório de Pré-Dissertação

Master dissertation
Master Degree in Computer Science

Dissertation supervised by
José Carlos Leite Ramalho

December 2019

ABSTRACT

Write abstract here (en)

RESUMO

Escrever aqui resumo (pt)

CONTEÚDO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE EXEMPLOS

GLOSSÁRIO

Application Programming Interface Interface ou protocolo de comunicação entre um cliente e um servidor i

ontologia Representação de conhecimento (conceitos e as relações entre estes) 2

Simplex Programa de Simplificação Administrativa e Legislativa 2

LISTA DE ACRÓNIMOS

AP Administração Pública 2
API Application Programming Interface 2–8, *Glossary:* Application Programming Interface

CC Cartão de Cidadão 5, 7
CLAV Classificação e Avaliação da Informação Pública iii, iv, 2–9
CSS Cascading Style Sheets 4
CSV Comma Separated Values 3

DGLAB Direção-Geral do Livro, dos Arquivos e das Bibliotecas 2, 6, 7

HMAC Hash-based Message Authentication Code 9, 10
HTML Hypertext Markup Language 4
HTTP Hypertext Transfer Protocol 8

JSON JavaScript Object Notation i, iii, 3, 9, 10
JWE JSON Web Encryption 9
JWS JSON Web Signature 9
JWT JSON Web Token iii, iv, vi, 7–11

LC Lista Consolidada 2, 3

NIC Número de Identificação Civil 6
NSA National Security Agency 10

PDF Portable Document Format 4

RDF Resource Description Framework 3
REST Representational State Transfer iii, 2, 9
RSA Rivest–Shamir–Adleman 9, 10

SHA-2 Secure Hash Algorithm 2 10

UM Universidade do Minho 2

XML Extensible Markup Language 3

1

INTRODUÇÃO

Vemos atualmente a mudança de paradigma em várias organizações e governos em relação a políticas e estratégias para a disponibilização de dados abertos nos domínios das ciências e da Administração Pública. Quanto à Administração Pública portuguesa têm sido promovidas políticas para a sua transformação digital com o objetivo de otimização de processos, a modernização de procedimentos administrativos e a redução de papel. De certa forma a agilização de procedimentos da Administração Pública portuguesa. Lourenço et al. (2019)

De forma a alcançar estes objetivos a Administração Pública (AP) tem desmaterializado processos e tem promovido a adoção de sistemas de gestão documental eletrónica bem como da digitalização de documentos destinados a serem arquivados. Lourenço et al. (2019)

Por forma a continuar esta transformação da AP a Direção-Geral do Livro, dos Arquivos e das Bibliotecas (DGLAB) apresentou a iniciativa da Lista Consolidada (LC) para a classificação e avaliação da informação pública. A LC serve de referencial para a construção normalizada dos planos de classificação e tabelas de seleção das entidades que executam funções do Estado. Lourenço et al. (2019)

Nasce assim o projeto Classificação e Avaliação da Informação Pública (CLAV) com um dos seus objetivos primordiais a operacionalização da utilização da LC, numa colaboração entre a DGLAB e a Universidade do Minho (UM) e financiado pelo Simplex. Lourenço et al. (2019)

A plataforma CLAV disponibiliza em formato aberto uma ontologia com as funções e processos de negócio das entidades que exercem funções públicas (ou seja a LC) associadas a um catálogo de legislação e de organismos. Desta forma, a CLAV viabiliza a desmaterialização dos procedimentos associados à elaboração de tabelas de seleção tendo como base a LC e ao controlo de eliminação e arquivamento da informação pública através da integração das tabelas de seleção nos sistemas de informação das entidades públicas alertando-as quando determinado documento deve ser arquivado ou eliminado. Esta integração promove também a interoperabilidade através da utilização de uma linguagem comum (a LC) usada no registo, na classificação e na avaliação da informação pública. Lourenço et al. (2019)

1.1 OBJETIVOS

A continuação do desenvolvimento da API de dados da CLAV nesta dissertação, seguindo uma metodologia REST, permite a processos ou aplicações aceder aos dados sem a intervenção

humana para além de suportar a plataforma CLAV. Um dos objetivos da API de dados é permitir futuramente a criação de novas aplicações através desta. Como tal, é extramemente essencial que a API de dados do CLAV possua uma boa documentação ajudando futuros programadores ou utilizadores a utilizar a API. Advém daí a necessidade de nesta dissertação realizar a documentação da API de dados em *Swagger*.

Apesar de o projeto ter em mente a disponibilização aberta de informação pública é necessário controlar a adição, edição e eliminação da informação presente na Lista Consolidada, bem como a informação de utilizadores, da legislação, das entidades, etc, mantendo-a consistente e correta. É, portanto, necessário controlar os acessos à API de dados com múltiplos níveis de acesso restringindo as operações que cada utilizador pode realizar consoante o seu nível. Desta forma garante-se que apenas pessoal autorizado pode realizar modificações aos dados.

Este controlo de acesso exige a existência de formas de autenticação. Como um cofre para o qual ninguém tem a chave não é útil pelo facto de que algo lá guardado ficará eternamente inacessível, também algo com controlo de acesso seria inútil caso não fosse possível ultrapassar esse controlo de alguma forma. Assim, uma das formas de autenticação usadas, Autenticação.gov, criada pelo Estado português, permite a autenticação dos cidadãos portugueses nos vários serviços públicos AMA (2019) entre os quais, a Segurança Social, o Serviço Nacional de Saúde e a Autoridade Tributária Aduaneira. Sendo este um projeto do Governo Português, a autenticação no CLAV através do Autenticação.gov é um requisito.

Por forma a contrariar o aumento da complexidade da API de dados com a adição do controlo de acesso e da autenticação pretende-se investigar se a criação de um API Gateway simplifica a comunicação entre interface/utilizadores e a API de dados.

Resumidamente, os objetivos desta dissertação são:

- Documentação em *Swagger* da API de dados da CLAV
- Adição de formatos de exportação à API de dados da CLAV (para além do já presente JSON, adicionar CSV, XML e RDF)
- (Continuação da) Integração do Autenticação.gov na CLAV
- Proteção da API de dados da CLAV com múltiplos níveis de acesso
- Estudo da crição de um API Gateway
- Integração do CLAV no iAP

2

ESTADO DA ARTE

2.1 ESTADO DA ARTE DO CLAV

Quando esta dissertação teve início o projeto CLAV já tinha cerca de 2 anos de desenvolvimento. Assim nesta secção será apresentado o estado da arte do CLAV quando esta dissertação iniciou aprofundando principalmente os pontos mais importantes sobre o tema desta dissertação.

2.1.1 *Estrutura*

O CLAV está dividido em duas partes:

- interface (*front-end*) presente em `http://clav.dglab.gov.pt`
- API de dados (*back-end* que inclui também duas bases de dados, *GraphDB* e *MongoDB*) presente em `http://clav-api.dglab.gov.pt`.

Cada parte encontra-se numa máquina diferente.

Através da figura 1 é possível ver o possível fluxo tanto de um utilizador a aceder à interface como a de um utilizador a aceder diretamente à API de dados. No primeiro caso, quando um utilizador acede o servidor da interface do CLAV é descarregado para o lado do utilizador o ficheiro HTML (*index*) e os vários ficheiros *JavaScript*, CSS e *assets* (como imagens, PDFs, etc) quando necessários. O servidor da interface é nada mais que um servidor *web* com recurso ao *Nginx* que hospeda estes ficheiros, os quais representam a interface construída com o *Vue* e o *Vuetify*. Como tal o código apresenta-se todo do lado do utilizador e os pedidos à API serão feitos do computador do utilizador para o servidor da API de dados e não do servidor da interface para o servidor da API de dados. Ou seja, o fluxo de cada um desses pedidos será igual ao fluxo no caso em que se acede diretamente a API sem uso de qualquer interface.

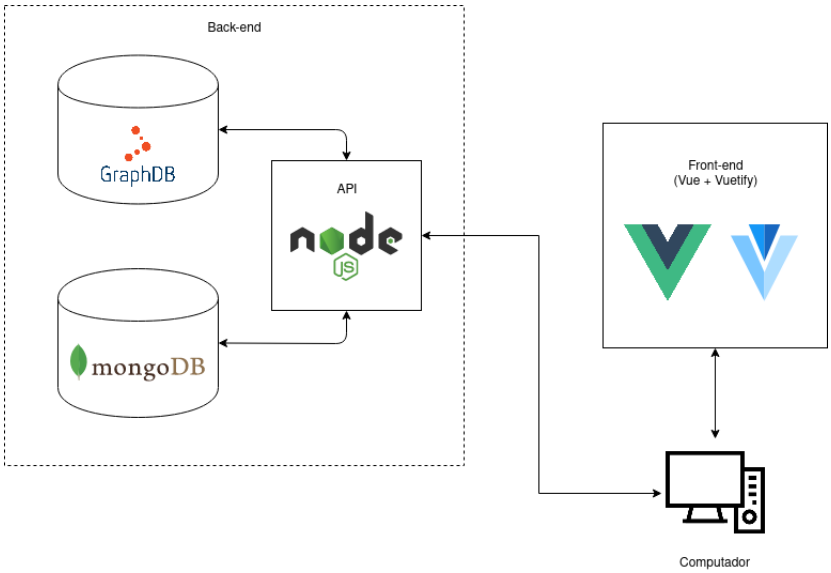


Figura 1: Estrutura do CLAV incluíndo a interação de um utilizador com a mesma

2.1.2 *Formas de autenticação*

A API de dados e a interface estavam inicialmente "juntas" (aplicação monolítica) onde as rotas eram protegidas contudo, com a separação da aplicação em duas partes, ambas partes deixaram de estar protegidas. Devido à plataforma já ter estado protegida esta já possui duas formas de autenticação, através de chaves API ou através de utilizadores registados. Ou seja, tanto o registo de utilizadores e de chaves API já se encontra implementado bem como o *login* de utilizadores.

As chaves API existem por forma a dar acesso a certas rotas da API a aplicações que interajam com a mesma (por exemplo sistemas de informação) sem a necessidade de interação humana.

Já os utilizadores possuem múltiplos níveis de acesso sendo que consoante o seu nível podem ou não aceder a uma rota da interface ou da API. Os utilizadores podem autenticarem-se através de *email* e *password* ou com recurso ao Cartão de Cidadão (CC) através do Autenticação.gov, este último apenas disponível através da interface do CLAV.

A hierarquia dos níveis de acesso, do nível que permite menor para o maior acesso, é a seguinte:

- Nível 0: Chaves API
- Nível 1: Representante Entidade
- Nível 2: Utilizador Simples
- Nível 3: Utilizador Avançado
- Nível 3.5: Utilizador Validador (AD)

- Nível 4: Utilizador Validador
- Nível 5: Utilizador Decisor
- Nível 6: Administrador de Perfil Funcional
- Nível 7: Administrador de Perfil Tecnológico

As chaves API poderão aceder a algumas rotas com método `GET`. Já os utilizadores poderão realizar todos os pedidos que as chaves API podem realizar mas quanto maior o seu nível de acesso mais rotas poderão aceder.

A proteção da API terá de ter esta hierarquia em conta.

Registo

Como já referido tanto o registo de chaves API como de utilizadores já se encontra implementado.

Para o registo de uma chave API é necessário providenciar um nome, um email e a entidade a que pertence. Após o registo da chave a informação desta chave API é mantida numa base de dados *MongoDB*.

Um utilizador pode se registar através de `email + password` ou através do Autenticação.gov. No primeiro caso, ao se registar necessita obviamente de indicar o seu email, a *password*, o seu nome, a entidade a que pertence e o nível de acesso que pretende. Já no caso do Autenticação.gov para o registo do utilizador é necessário todos os campos anteriores exceto a *password* (pode ser depois definida), sendo também necessário o campo Número de Identificação Civil (NIC) do utilizador. Caso o registo seja efetuado com recurso à interface do Autenticação.gov apenas será necessário indicar o email, a entidade a que pertence e o nível de acesso que pretende visto que os restantes campos são fornecidos pela Autenticação.gov quando o utilizador se autentica e autoriza nesta a partilha dessa informação com a plataforma do CLAV. A *password* é armazenada não na sua forma literal mas sim a sua *hash* ao aplicar a função criptográfica `bcrypt`. A utilização de funções de *hash* criptográficas ao armazenar *passwords* impede que as *passwords* originais se saibam caso a base de dados seja comprometida. Para além disso, como o `bcrypt` combina um valor aleatório (`salt`) com a *password* do utilizador, é impossível pré-computar a *password* que deu origem ao *hash* sem saber o `salt`[1].

Durante esta tese com a proteção da API ficará apenas possível o registo de utilizadores através de utilizadores que já estejam registados e possuam um nível de acesso suficiente para registar utilizadores. Estes utilizadores registados e autorizados pertencem à entidade DGLAB. Portanto por forma a utilizadores representantes de outras entidades se registarem na plataforma terão de: DGLAB (2019)

- Preencher o formulário disponibilizado para o efeito, para cada representante designado pela entidade;

1 Para mais informação veja *rainbow table attack*

- O formulário deverá ser assinado por um dirigente superior da Entidade e autenticado com assinatura digital, se o envio for feito por via eletrónica (NB: não serão aceites assinaturas do formulário por dirigentes intermédios). Esta autorização autenticada pelo dirigente superior é o equivalente a uma delegação de competências, uma vez que o representante da entidade passa a ter capacidade para, em nome da entidade, submeter autos de eliminação, propostas de tabelas de seleção e novas classes para a Lista Consolidada;
- O formulário deverá ser remetido à DGLAB por via postal ou eletrónica, respetivamente, para:
  - DGLAB, Edifício da Torre do Tombo, Alameda da Universidade, 1649–010 Lisboa (formulário assinado manualmente) ou
  - clav@dglab.gov.pt (formulário com assinatura digital).
- Após receção do formulário, a DGLAB efetuará o(s) respetivo(s) registo(s) até 48 horas úteis;
- Findo esse prazo, o utilizador poderá aceder à plataforma, selecionando a opção "Autenticação";
- A autenticação, no primeiro acesso, deve ser efetuada com o Cartão de Cidadão.

Login

O *login* apenas está presente para o caso dos utilizadores visto que assim que uma chave API é registada é enviado por email um JWT com a duração de 30 dias a ser usado nos pedidos a realizar à API. O utilizador poderá ao fim dos 30 dias renovar a sua chave API, onde é gerado um novo JWT.

Portanto do lado dos utilizadores é possível como já referido realizar o *login* de duas formas através de uma estratégia local ou através do Autenticação.gov.

A estratégia local (`email + password`) é conseguida através do uso do *middleware Passport*. O *Passport* é um middleware de autenticação para *Node.js* que tem como objetivo autenticar pedidos. Passport.js (2019) Tem como única preocupação a autenticação delegando qualquer outra funcionalidade para a aplicação que a usa. Este *middleware* possui muitas estratégias de autenticação entre as quais a local (`email/username + password`), JWT, *OAuth*[2], *Facebook* ou *Twitter*. Cada estratégia está num módulo independente. Assim as aplicações que usam o *Passport* não terão um peso adicional devido a estratégias que nem sequer usam.

No caso do *login* através do Autenticação.gov, o utilizador tem de se autenticar na interface do Autenticação.gov (a partir do botão disponível na área de autenticação da interface do CLAV). O fluxo do *login* neste caso é:

2 Protocolo *open-source* com o objetivo de permitir a autenticação simples, segura e padrão entre aplicações móveis, *web* e *desktop*

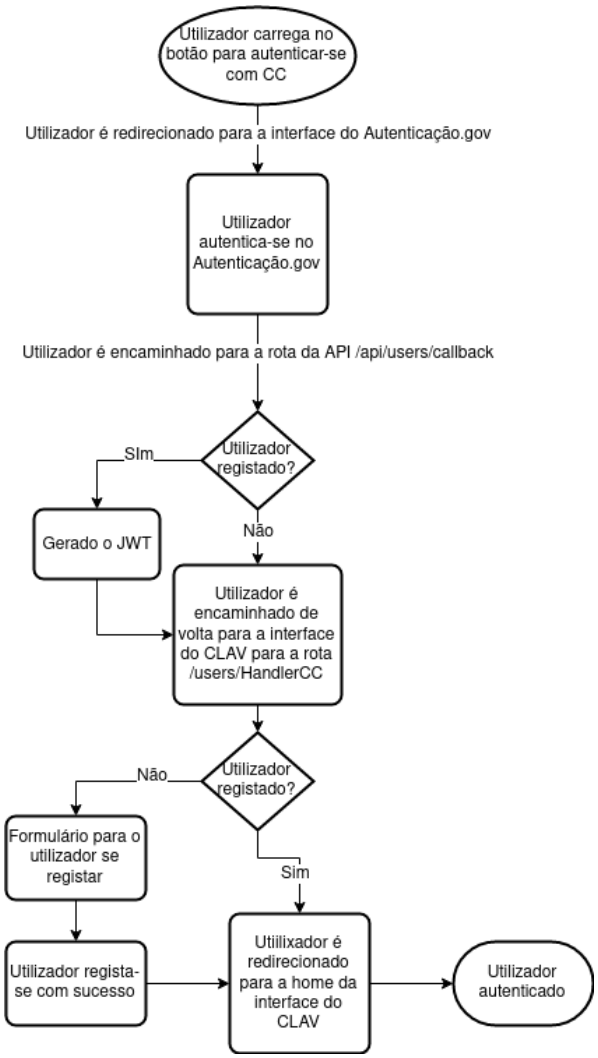


Figura 2: Fluxo do *login* de um utilizador através do Autenticação.gov

No *login* do utilizador é gerado um JWT com a duração de 8 horas que deve ser usado nos pedidos a realizar à API. No fim das 8 horas o utilizador necessita de se autenticar de novo.

A melhorar

As duas formas de autenticação necessitam de melhorias. Uma delas é a possibilidade de o JWT do utilizador puder ser enviado no `body` algo que nem todos os métodos HTTP permitem bem como o *Swagger* que iremos falar mais à frente não suporta a documentação desta forma de autenticação com JWT's. Logo deve ser removido este tipo de autenticação.

2.1.3 *Lista Consolidada*

2.1.4 *Tabelas de Seleção*

2.1.5 *Cache e Fecho Transitivo*

2.2 REST

Richardson and Ruby (2007)

2.3 JSON Web Token (JWT)

O JWT é um *open standard*[3] que define uma forma compacta e independente de transmitir com segurança informação entre partes com um objeto JSON. Autho (2019) O JWT pode ser assinado digitalmente (JWS), encriptado (JWE), assinado e depois encriptado (JWS encriptado gerando um JWE, ordem recomendada[4]) ou encriptado e depois assinado (JWE assinado gerando um JWS).

Caso seja assinado digitalmente é possível verificar a integridade da informação mas não é garantida a sua privacidade contudo podemos confiar na informação do JWT. A assinatura pode ser efetuada através de um segredo usando por exemplo o algoritmo HMAC ou através de pares de chaves pública/privada usando por exemplo o algoritmo RSA. No caso de se usar pares de chaves pública/privada a assinatura também garante que a parte envolvida que tem a chave privada é aquela que assinou o JWT.

Por outro lado, os JWTs podem ser encriptados garantindo a privacidade destes, escondendo a informação das partes não envolvidas.

Sendo assim em que casos é útil o uso de JWTs? Dois dos casos são os seguintes:

- Autorização: Este será o caso para o qual o JWT será usado na CLAV. Quando o utilizador realiza o *login* gera-se um JWT por forma a que os restantes pedidos desse utilizador sejam realizados com esse JWT (*Single Sign On*). O uso de JWTs para estes casos permitem um *overhead* pequeno e a flexibilidade de serem usados em diferentes domínios.

- Troca de informação: No caso de troca de informação entre duas partes os JWTs assinados são de bastante utilidade visto que permitem verificar se o conteúdo não foi violado e, no caso de se usar pares de chaves pública/privada para assinar, permitem ter a certeza que o remetente é quem diz ser.

3 Mais informação em https://tools.ietf.org/html/rfc7519
4 Mais informação em https://tools.ietf.org/html/rfc7519#section-11.2

2.3.1 *Estrutura do JWT*

Os JWTs são construídos a partir de três elementos, o *header* (objeto JSON), o *payload* (objeto JSON) e os dados de assinatura/encriptação (depende do algoritmo usado). Estes elementos são depois codificados em representações compactas (`Base64 URL-safe`[5]). As codificações `Base64 URL-safe` de cada elemento são depois concatenadas através de pontos dando origem a uma representação final compacta do JWT (*JWS/JWE Compact Serialization*).

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJuYW1lIjoiSm9zw6kgTWFydGlucyIsIm51bSI6ImE3ODgyMSJ9.
tRPSYVsFI-nziRPuAjdGZLN2tUez5MtLML_aAnPplgM

Figura 3: Exemplo de representação compacta de JWT, quebra de linhas por forma a melhorar leitura

De seguida vamos aprofundar cada elemento referido:

***Header*:** O cabeçalho consiste em duas partes (dois atributos):

- O atributo opcional `typ` (tipo do *token*) em que o seu valor é "JWT"
- O atributo obrigatório (único campo obrigatório para o caso de um JWT não encriptado) `alg` (algorítmo) onde é indicado que algoritmo é usado para assinar e/ou desencriptar. O seu valor pode ser por exemplo HS256 (HMAC com o auxilio do SHA-256[6]) ou RSA.

O cabeçalho é de grande importância visto que permite saber se o JWT é assinado ou encriptado e de que forma o resto do JWT deve ser interpretado.

Exemplo de cabeçalho:

```
{
    "alg": "HS256",
    "typ": "JWT"
}
```

Listing 2.1: *Header* do JWT usado na figura 3

Este JSON é depois codificado em `Base64URL` gerando a primeira parte do JWT

***Payload*:** O *payload* contém a informação/dados que pretendemos transmitir com o JWT.

Exemplo de *payload*:

```
{
    "name": "José Martins",
```

5 Variante da codificação `Base64` onde a codificação gerada é segura para ser usada em *URL*s. Basicamente para a codificação `Base64` gerada substitui os caracteres '+' e '/' pelos caracteres '-' e '_' respetivamente. Além disso, remove o caracter de *padding* e proibe separadores de linha

6 Função pertencente ao conjunto de funções *hash* criptográficas Secure Hash Algorithm 2 (SHA-2) desenhadas pela NSA

```
    "num": "a78821"
}
```

Listing 2.2: *Payload* do JWT usado na figura 3

***Signature*:** Exemplo de cabeçalho:

```
HMACSHA256(
    base64UrlEncode(header) + "." +
    base64UrlEncode(payload),
    segredo1.-uminho!clav
)
```

Listing 2.3: *Signature* do JWT usado na figura 3

2.4 CORS

2.5 HTTP STATUS

2.6 HEADERS DO HTTP

2.7 AUTENTICAÇÃO.GOV

AMA (2018)

2.8 MONGODB

Satheesh et al. (2015)

2.9 WEB SEMÂNTICA

DuCharme (2011)

2.9.1 *RDF*

DuCharme (2011)

2.9.2 *SPARQL*

DuCharme (2011)

2.10 GRAPHDB

2.11 SWAGGER

2.12 SWAGGER-UI

2.13 NGINX

DeJonghe (2018)

2.14 ONTOLOGIA

Arp et al. (2015)

2.15 DOCKER

Mouat (2015)

2.16 DOCKER COMPOSE

Mouat (2015)

3

O PROBLEMA E OS SEUS DESAFIOS

4

CONCLUSÃO

BIBLIOGRAFIA

AMA. *Autenticação.gov - Fornecedor de autenticação da Administração Pública Portuguesa*, 1.5.1 edition, 12 2018.

AMA. Autenticação.gov, 2019. URL https://autenticacao.gov.pt/fa/Default.aspx. Acedido a 2019-11-20.

Robert Arp, Barry Smith, and Andrew Spear. *Building Ontologies with Basic Formal Ontology*. MIT Press, 1st edition, 7 2015. ISBN 978-0-262-52781-1.

Autho. Introduction to json web tokens, 2019. URL https://jwt.io/introduction/. Acedido a 2019-12-19.

Derek DeJonghe. *NGINX Cookbook*. O'Reilly, 1st edition, 11 2018. ISBN 978-1-491-96893-2. Second Release.

DGLAB. CLAV - Classificação e Avaliação da Informação Pública, 2019. URL http://clav.dglab.gov.pt. Acedido a 2019-12-15.

Bob DuCharme. *Learning SPARQL*. O'Reilly, 1st edition, 7 2011. ISBN 978-1-449-30659-5.

Alexandra Lourenço, José Carlos Ramalho, Maria Rita Gago, and Pedro Penteado. Plataforma CLAV: contributo para a disponibilização de dados abertos da Administração Pública em Portugal. Acedido a 2019-11-20, 7 2019. URL http://hdl.handle.net/10760/38643.

Adrian Mouat. *Using Docker*. O'Reilly, 1st edition, 12 2015. ISBN 978-1-491-91576-9.

Passport.js. Overview, 2019. URL http://www.passportjs.org/docs/. Acedido a 2019-12-17.

Leonard Richardson and Sam Ruby. *RESTful Web Services*. O'Reilly, 1st edition, 5 2007. ISBN 978-0-596-52926-0.

Mithun Satheesh, Mithun Satheesh, and Jason Krol. *Web Development with MongoDB and NodeJS*. Packt Publishing, 2nd edition, 10 2015. ISBN 978-1-78528-752-7.

NB: place here information about funding, FCT project, etc in which the work is framed. Leave empty otherwise.