Universidade de Aveiro
2018

Departamento de Eletrónica,
Telecomunicações e Informática

**André Filipe
da Silva Santos**

Segurança em Sistemas de Informação para Cidades Inteligentes

Security in Information Systems for Smart Cities

Universidade de Aveiro
2018

Departamento de Eletrónica,
Telecomunicações e Informática

André Filipe
da Silva Santos

Segurança em Sistemas de Informação para Cidades Inteligentes

Security in Information Systems for Smart Cities

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor João Paulo Barraca, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Diogo Gomes, Professor auxiliar do Departamento de Electrónica e Telecomunicações da Universidade de Aveiro.

Esta dissertação foi financiada pelo programa LIFE Ambiente da União Europeia através do projeto LIFE 15 ENV/PT/000609.

Dedico este trabalho aos meus pais, à minha irmã, à Rita, à família e amigos presentes, de uma forma ou de outra, durante este percurso académico.

o júri / the jury

presidente / president

Professor Doutor André Ventura da Cruz Marnoto Zúquete
Professor Auxiliar da Universidade de Aveiro

vogais / examiners committee

Professor Doutor Sérgio Armindo Lopes Crisóstomo
Professor Auxiliar da Universidade do Porto na Faculdade de Ciências

Professor Doutor João Paulo Silva Barraca
Professor Auxiliar da Universidade de Aveiro

agradecimentos /
acknowledgements

Quero agradecer a toda a gente que esteve comigo desde o primeiro ao último dia, e que contribuíram de uma maneira ou outra para a conclusão desta fase tão importante da minha vida.
Aos meus pais que sempre me apoiaram e permitiram que continuasse este percurso mesmo quando as coisas não corriam bem.
À Rita que esteve presente de forma incansável dando todo o apoio e carinho nos melhores e piores momentos, motivando-me a ser sempre melhor.
À restante família pela ajuda, motivação e apoio em todos os momentos.
A todos os meus amigos e conhecidos que encontrei durante estes anos, especialmente aos que tornaram esta etapa tão especial, o Filipe, o Fábio e o André.
Um agradecimento também ao Luís e ao Hélder, colegas durante o projeto. Este trabalho deve-se também ao trabalho que ambos realizaram e à ajuda que me deram durante a sua realização.
E por fim ao meu orientador, o Professor Doutor João Paulo Barraca, pela sua disponibilidade, conhecimento e pela oportunidade de realizar esta dissertação.

Palavras Chave

Segurança Informática, Cidades Inteligentes, Internet das Coisas, Privacidade, Proteção de Dados, Controlo de Acesso, Políticas de segurança, PAYT

Resumo

Com o crescimento das populações nos grandes centros urbanos, surgem cada vez mais problemas ao nível da gestão de recursos das cidades. Graças à evolução das tecnologias ligadas à internet é possível utilizar essas tecnologias para auxiliar na solução de problemas, criando assim as cidades inteligentes. Este trabalho analisa uma solução de cidade inteligente existente ao nível da segurança dos dados e aplicação e apresenta uma solução para colmatar as suas falhas e mostrar como desenvolver aplicações com uma camada de segurança adequada. É apresentada uma solução que tem em conta a criticalidade dos dados, tal como exigido no novo Regulamento de Proteção de Dados da União Europeia, e depois é executada a implementação. A implementação da solução é por fim validada por vários testes de desempenho e também funcionais o que permite avaliar o nível de sucesso do trabalho apresentado nesta dissertação.

Keywords

Security in Information Systems, Smart Cities, Internet of Things, Privacy, Data Protection, Access Control, Security Policies, PAYT

Abstract

With a significant population growth in large urban centers, the problems in the management of cities' resources are rapidly increasing. But due to the evolution of the technologies connected to the internet it is possible to use those to aid in the solution of problems, thus creating the so called Smart Cities. This dissertation examines an existing Smart City solution, and presents a solution to address its flaws and show how to develop applications with a proper security layer for the data and application layer. A solution is presented which takes into account the criticality of the data as required by the new EU Data Protection Regulation, and then implements using the appropriate technologies. The implementation of the solution is validated by several performance and functional tests, which allows to evaluate the level of success of the work presented in this dissertation.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

MAC	Controlo de acesso obrigatório
NFC	Near Field Communication
ICN	Information Centric Networking
ISO	Organização Internacional de Normalização
DAC	Controlo de acesso discricionário
RBAC	Controlo de acesso baseado em papéis
ABAC	Controlo de acesso baseado em atributos
XACML	eXtensible Access Control Markup Language
PAYT	Pay-As-You-Throw
IoT	Internet of Things
RGPD	Regulamento Geral de Proteção de Dados
EUA	Estados Unidos da América
UE	União Europeia
RFID	Radio-Frequency Identification
NDI	National Digital Identity
MOL	Moments of Life
OWASP	Open Web Application Security Project
XSS	Cross-Site Scripting
SQLi	SQL Injection
CSRF	Cross-Site Request Forgery
TLS	Transport Layer Security
HTTPS	Hypertext Transfer Protocol Secure
HTTP	Hypertext Transfer Protocol
DDoS	Distributed Denial of Service
AES	Advanced Encryption Standard
USB	Universal Serial Bus
NATO	North Atlantic Treaty Organization
ACL	Lista de Controlo de Acesso
NIST	National Institute of Standards and Technology
SHA-1	Secure Hash Algorithm 1
SHA-2	Secure Hash Algorithm 2
SHA-224	Secure Hash Algorithm 224
SHA-256	Secure Hash Algorithm 256
SHA-384	Secure Hash Algorithm 384
SHA-512	Secure Hash Algorithm 512
MD5	Message Digest 5
MD4	Message Digest 4
DES	Data Ecnryption Standard
XML	eXtensible Markup Language
MiTM	Man In The Middle
CKAN	Comprehensive Knowledge Archive Network
ELK	Elasticsearch + Logstash + Kibana
API	Application Programming Interface
ZFS	Zettabyte File System
SQL	Standardized Query Language
PBKDFv2	Password-Based Key Derivation Function 2
RSA	Rivest, Shamir, & Adleman
AES-256	Advanced Encryption Standard 256 Bits
JSON	JavaScript Object Notation

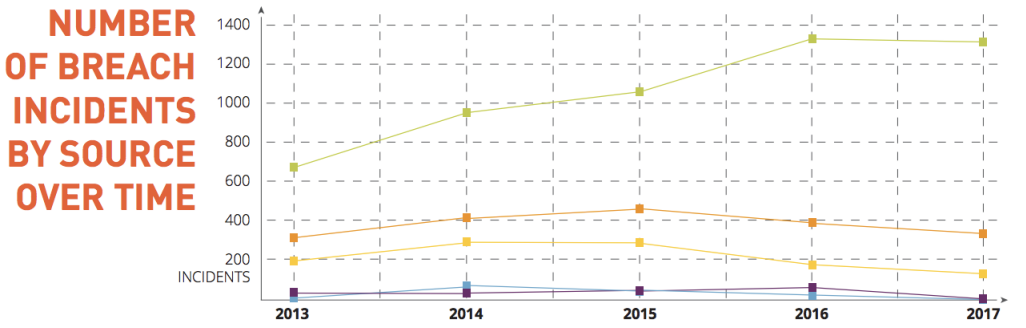
AES-128	Advanced Encryption Standard 128 Bits
AES-256	Advanced Encryption Standard 256 Bits
AES-512	Advanced Encryption Standard 512 Bits
IEC	International Electrotechnical Commission
PEP	Policy Enforcement Point
PDP	Policy Decision Point
PIP	Policy Information Point
PRP	Policy Retrieval Point
PAP	Policy Administration Point
CMS	Content Management System
PBKDFv2	Password-Based Key Derivation Function 2
URL	Universal Resource Locator

CAPÍTULO 1

INTRODUÇÃO

A proliferação de informação e o avanço tecnológico verificada nos dias de hoje, motivam o aparecimento do Internet of Things (IoT), que tem com o objetivo simplificar, agilizar e otimizar casos de uso aplicáveis, tais como gestão de infraestruturas (casas, escritórios, oficinas, etc.), de serviços e aplicações. Por outro lado, o crescimento das populações nas áreas urbanas tornou-se um desafio: surge então a necessidade de adotar políticas que permitam otimizar a gestão das mesmas. É neste contexto que aparece o conceito *Smart City* (cidade inteligente), um exemplo da aplicação do conceito de IoT.

Os sistemas de IoT captam, processam e guardam imensa informação sobre as populações o que traz grandes responsabilidades para quem os desenha e programa. Esta responsabilidade tem ainda mais valor com a introdução do novo Regulamento Geral de Proteção de Dados (RGPD) da União Europeia. O novo regulamento surge como uma medida que visa proteger a informação pessoal dos cidadãos que é cada vez mais utilizada pelos sistemas de informação garantindo que apenas cada pessoa tenha a propriedade dos seus dados. Com cada novo sistema a entrar em funcionamento aumenta também a probabilidade de incidentes de segurança com perda ou roubo de dados que podem prejudicar gravemente cidadãos e organizações. Na figura 1.1 estão ilustrados o número de incidentes relacionados com sistemas de informação de vários tipos. Um dado que devemos destacar são os incidentes com origem em ações maliciosas que, desde 2013 a 2017, duplicou e que poderá vir a ter um crescimento continuo à medida que os sistemas de informação proliferem. Além dos potenciais ataques, a questão da ética da utilização da informação pessoal por parte das grandes organizações, sem consentimento dos utilizadores, adquire um novo destaque com o RGPD cujo principal objetivo é proteger os interesses dos proprietários da informação, de forma a que possam ter controlo sobre a gestão da sua informação pessoal.



BREACH SOURCE	2013	2014	2015	2016	2017
Malicious Outsider	662	955	1,087	1,336	1,269
Accidental Loss	302	413	442	393	326
Malicious Insider	195	290	278	179	164
Hacktivist	27	20	36	49	4
State Sponsored	12	61	36	20	1
Unknown	19	4	4	4	1
TOTALS	1,217	1,743	1,883	1,981	1,765

Source: BREACHLEVELINDEX.COM

Figura 1.1: Número de incidentes de segurança nos últimos 5 anos [1].

1.1 MOTIVAÇÃO

Este trabalho está enquadrado no âmbito do projeto LIFE PAYT que tem como objetivo demonstrar a eficácia dos sistemas PAYT através de sistemas inteligentes integrados numa componente de software desenvolvida na Universidade de Aveiro. Os sistemas PAYT, tal como o próprio nome indica, permitem cobrar uma quantidade justa de taxas a cada cidadão pelos resíduos produzidos. Através da identificação dos resíduos e a associação a um cidadão ou conjunto de cidadãos é possível aplicar o "princípio do poluidor-pagador" o que promove a reciclagem, através da aplicação de um estímulo: uma recompensa monetária para essas ações, o mesmo acontecerá nas faturas da água e eletricidade.

Na Europa Central e do Norte, nomeadamente na Suíça, Áustria, Alemanha, Itália, Dinamarca e Holanda existe já uma grande adesão ao programa PAYT. Nos Estados Unidos da América (EUA) o programa começou a ser implementado no final dos anos 90. Atualmente, mais de 7000 municípios dos EUA implantaram esquemas PAYT representando quase um quarto do número total de municípios e população dos EUA [2]. Em relação à sua aplicação, o esquema PAYT pode ser feito de várias maneiras. Nos EUA, o pay-per-bin com sistemas de registo individuais predominam nos municípios de maior dimensão, enquanto que nos municípios mais pequenos a escolha incide sobre pay-per-bag ou pay-per-bin com sistemas de *tag*, que consiste na identificação digital com recurso à tecnologia Near Field Communication (NFC) ou autocolantes [2].

Este programa é uma nova abordagem à gestão de resíduos no sul da Europa e espera-se que com o projeto LIFE PAYT, mais municípios comecem a adotar o esquema PAYT para incentivar uma gestão consciente de resíduos, de forma a cumprir as metas definidas pela União Europeia. Os municípios portugueses envolvidos neste projeto piloto são Aveiro, Condeixa-a-Nova e Lisboa, estão também presentes outros países europeus, Grécia (Vrilissia) e Chipre (Larnaca).

1.2 OBJETIVOS

O projeto LIFE PAYT terá como objetivo a implementação do conceito de PAYT, nos diversos municípios e irá recorrer a tecnologias utilizadas em sistemas de *Smart City*, que envolverão armazenamento e processamento de informação pessoal por parte de quem utilizará o sistema. Com o novo RGPD torna-se extremamente importante ter em atenção este aspeto de forma a que o sistema cumpra todas as obrigações legais.

O objetivo principal desta dissertação é melhorar a solução de software existente disponível no projeto, com o foco na aplicação da mais recente regulamentação de privacidade de dados e as práticas recomendadas pela Organização Internacional de Normalização (ISO) no documento 27001 [3]. A nossa principal preocupação será a proteção da informação, através da utilização de boas-práticas de segurança informática, tais como: mecanismos de controlo de acesso, gestão de palavras-passe e de equipamentos e políticas de segurança. O trabalho desenvolvido deverá mitigar ataques ao sistema que possam comprometer a proteção da informação e cumprir o novo RGPD.

1.3 CONTRIBUTOS

O nosso trabalho, para além da dissertação, terá implicação prática na vida quotidiana dos cidadãos, na medida em que constituiu um contributo significativo para a segurança do software que suporta o sistema LIFE PAYT, que irá ser implementado nos locais supracitados.

Este trabalho teve em grande atenção o novo RGPD, e esteve também em linha com as normas da certificação ISO 27001, o que garante conformidade com a nova legislação e dá ao projeto a possibilidade de no futuro estar mais próximo de obter

uma certificação ISO 27001. O desenvolvimento do trabalho orientado à certificação mesmo sem a obter, é um passo importante para a obtenção de um software final com mais qualidade e segurança. Isto porque o ISO 27001 ajuda na adoção de políticas e mecanismos de trabalho para obter melhores resultados e alcançar os objetivos definidos pela certificação.

1.4 ESTRUTURA DA DISSERTAÇÃO

Esta dissertação está dividida em 6 capítulos, Introdução (atual), Estado de Arte, Solução, Implementação, Resultados e Testes, Conclusões e Trabalho Futuro. Os capítulos seguintes podem ser resumidos da seguinte forma:

- **Capítulo 2:** neste capítulo, inicialmente será feito o enquadramento teórico do tema da dissertação e serão abordados e definidos os conceitos-chave para a elaboração desta dissertação, tais como os conceitos de Smart City, Segurança, Privacidade, Modelos de controlo de acesso, proteção de informação e soluções relevantes.
- **Capítulo 3:** após uma apresentação do estudo de caso e dos requisitos em termos de segurança será apresentada neste capítulo uma visão sobre a solução a adotar e quais os casos de uso que ela pode resolver.
- **Capítulo 4:** após a solução apresentada no capítulo anterior serão apresentadas possíveis implementações e a implementação por fim adotada para cumprir os objetivos definidos.
- **Capítulo 5:** apresentação dos resultados dos testes realizados que visam compreender o impacto em termos de desempenho das alterações feitas ao sistema.
- **Capítulo 6:** conclusões sobre o trabalho efetuado e indicação de possíveis melhoramentos a fazer no futuro.

CAPÍTULO 2

Estado de Arte

2.1 Smart Cities

Decorre o ano 2018 e tudo o que é tecnologia utilizada no nosso dia a dia é considerado *smart*: os nossos telefones, os nossos relógios, carros, televisões e até máquinas de café e de lavar a louça. Tudo começa a ser *smart* e a estar ligado à internet, a transmitir e receber dados e até a tomar decisões por nós. A figura 2.1 ilustra alguns exemplos da aplicação de *Smart City*, como a gestão inteligente de recursos energéticos, mobilidade, serviços de saúde, indústria agrícola, serviços de venda e as próprias habitações.

Figura 2.1: Funcionalidades inerentes a uma Smart City [4].

Com a crescente evolução tecnológica, começa a ser possível, e até necessário, aplicar o conceito *smart* às cidades, uma vez que estas se vão tornando cada vez mais

"inteligentes". Esta inteligência define-se, por exemplo, pela possibilidade de existir constante ligação à internet, a utilização de sensores- capazes de medir temperaturas, nível de humidade, localizações, entre outras, que visam efetivar um processamento automático de informação interligada. Todos estes acontecimentos dão uma nova vida à cidade, tendo como objetivo torná-la mais inteligente, mais eficiente, mais agradável e mais segura para os seus cidadãos, acabando por gerar também uma nova fonte de rendimento económico para os municípios e país [5].

O termo *Smart City*, cuja definição não consensual [6], assenta, segundo vários autores em 3 pilares fundamentais: IoT, *Cloud* e *Big Data*. Embora existam vários estudos e literatura sobre estas 3 áreas analisadas em separado, o conceito *Smart City* vem criar um novo debate, pois combina a utilização dos 3 conceitos de uma forma complexa sobre a qual ainda não temos informação e conhecimento profundo como nas 3 áreas que o compõem [5].

Em relação à definição de Internet of Things (IoT), esta foi proposta pela primeira vez em 1998 e define IoT como um conjunto de objetos únicos e identificados com uma representação virtual numa estrutura semelhante à internet [7]. Como em todas as tecnologias houve, entretanto, uma evolução deste conceito até à forma como conhecemos hoje o IoT. Atzori et al. identificam 3 gerações de IoT: Radio-Frequency Identification (RFID) e sensores, serviços web e computação na internet, e por último a geração da *Cloud*, redes sociais e Information Centric Networking (ICN) [8].

A primeira geração está diretamente ligada à introdução da tecnologia de identificadores de radiofrequência e teve como principal objetivo a criação de uma norma para suportar a utilização de RFID e código eletrónico de produto [8]. Após a introdução da primeira geração de IoT começou-se a pensar em reduzir a utilização de RFID e dar aos objetos o acesso à internet como se de um computador tratasse. Estes objetos são dispositivos de baixo consumo com alguma capacidade de processamento e funcionalidades sensoriais [8].

Por fim temos a terceira geração de IoT que continua com os mesmos princípios básicos da segunda geração, mas com foco em objetos sociais e computação na *Cloud*. Aqui os dados são processados e armazenados na *Cloud* e os próprios sensores têm capacidades de processamento e de armazenamento temporário (*cache*) [8]. Esta é a geração na qual se inserem as mais recentes iniciativas de *Smart Cities* que procuram através da análise da informação solucionar problemas complexos nas áreas urbanas. Esta definição poderá estar na origem do que muitos autores consideram a definição de *Smart City*, que é uma mistura de IoT, *Cloud* e *Big Data*. Uma vez que esta geração de IoT é muito dependente da *Cloud*, e o armazenamento de grandes quantidades de

informação em grandes quantidades dá origem ao Big Data, pois diz a definição de De Mauro A. et al. que "Big Data são ativos de informação caracterizados por um grande volume, velocidade e variedade que exigem métodos tecnológicos e analíticos específicos para que possam ser transformados em algo com valor" [9].

O conceito de *Smart City* começa a ser cada vez mais popularizado e adotado por parte dos governantes para resolver problemas das áreas urbanas. Com o crescimento do número de pessoas a viver nas áreas urbanas [10] começam a surgir problemas na organização das cidades, seja a nível de transportes, utilização de serviços e bem-estar geral da população. Estima-se que em 2050 dois terços da população mundial viva em áreas urbanas e nessa altura o grande desafio será conseguir cobrir as necessidades básicas das populações como comida, água e energia o mais eficiente possível [11]. Terão de garantir ainda mais a segurança e proteção da população, facilidade de deslocamento com recurso a transportes públicos inteligentes e mais eficientes, e ainda sustentabilidade em termos económicos, sociais e ambientais [11]. Estas dificuldades levam os governos locais a apostar fortemente nas *Smart Cities* para resolver os seus problemas de forma mais eficaz, com vantagens económicas a longo prazo mesmo após um investimento considerável que uma *Smart City* representa tornando-se numa grande oportunidade de negócio para fornecedores de soluções *Smart City*. Segundo Dominique Bonte da ABIresearch [12] que analisou 75 cidades com mais de 5 milhões de habitantes, a estimativa de redução de custos pode ascender aos 3 triliões de dólares no conjunto das cidades em estudo. É por isso importante ter em conta cada vez mais a proteção dos dados, pois com o crescimento destes sistemas é fundamental haver uma consciência de quem os desenvolve perceber os riscos e de como os mitigar.

2.1.1 CASOS RELEVANTES

A cidade de Barcelona investiu fortemente no conceito de *Smart City* nos últimos anos. As grandes preocupações com redução de emissões, redução de utilização de energia e poupança de água levou a que se apostasse nas novas tecnologias para ajudar a resolver estes problemas. Por toda a cidade existem sensores que controlam a qualidade do ar, ruído, estacionamento inteligente, iluminação inteligente e muitos *hotspots* Wi-Fi para fácil acesso à internet [13]. A área dos transportes é uma das grandes apostas para reduzir emissões através de tecnologia avançada. Apostou-se em veículos elétricos e numa experiência interativa com o serviço de autocarros, permitindo os cidadãos consultar a localização em tempo real de cada autocarro, ligação Wi-Fi e portas Universal Serial Bus (USB) para carregar dispositivos. Os parques de estacionamento utilizam sensores no asfalto para localizar lugares livres e guiar condutores à procura de um

lugar até um lugar vazio [14]. Toda esta informação é gerida através do Sentilo [15], uma plataforma *Open-Source* disponível para todos aplicarem aos seus casos de uso. A criação de plataformas *Open-Source* e a publicação de dados abertos são formas de dar credibilidade e uma boa imagem aos projetos *Smart City* e também uma forma de envolver os cidadãos na participação e no apoio às iniciativas. Esta credibilidade aumenta ainda mais se analisarmos que dados são recolhidos neste caso da cidade de Barcelona.

O governo de Singapura, vai mais longe e pretende com o seu projeto *Smart Nation* tornar-se numa nação competitiva e agradável para habitar, com recurso à tecnologia, levando o conceito de *Smart City* para outro nível. As iniciativas estilo *Smart City* focam-se principalmente nas áreas da Saúde, Serviços, Mobilidade e Qualidade de vida [16]. Algumas funcionalidades implementadas são sensores que detetam se pessoas estão a fumar em zonas não autorizadas ou se alguém está a despejar lixo na rua sem utilizar um contentor adequado [13].

Os principais projetos desta iniciativa de *Smart Nation* são o National Digital Identity (NDI), um sistema que permite efetuar transações de forma segura e conveniente entre cidadãos e empresas um pouco à imagem das moedas virtuais. Similar ao projeto NDI é o *e-Payments* que permite efetuar transações *swift* com a mesma facilidade. Outros projetos igualmente importantes são o *Smart Nation Sensor Platform* e *Smart Urban Mobility* que vão facilitar o dia a dia em termos de mobilidade e utilização de serviços e assegurar a segurança da população. Por fim existe o Moments of Life (MOL) que permite facilitar a vida a todos as famílias com crianças abaixo dos 6 anos de idade. A iniciativa tem como objetivo evitar o contacto com múltiplas agências aquando do nascimento de uma criança. Com o MOL as famílias podem de uma só vez e tudo na mesma plataforma online registar o nascimento do filho, concorrer ao subsídio de nascimento, fazer uma escolha preferencial de instituições para a criança frequentar o ensino pré-escolar e escolar, aceder a uma lista de instituições que beneficiam do subsídio de nascimento fornecendo descontos a quem tem direito, aceder ao registo de vacinas e consultas médicas e por fim obter qualquer informação por parte do governo relativamente aos filhos [17].

Estes dois casos, Barcelona e Singapura, aplicaram a ideia de *Smart City* a larga escala revolucionando grande parte da vida dos cidadãos. Outros municípios e países de menor dimensão começam a apostar também nestas tecnologias para ajudar com tarefas simples tais como a recolha de resíduos urbanos. Uma referência para a aplicação da tecnologia nas infraestruturas de recolha e tratamento de resíduos urbanos é o município alemão Aschaffenburg. Em muitos países como a Suíça, a Alemanha e Estados Unidos

é aplicado o sistema PAYT, um sistema que cobra aos utilizadores uma tarifa como acontece no fornecimento de água ou eletricidade, fazendo com que cada utilizador pague aquilo que produz.

O conceito de PAYT segundo os autores Batllevell e Hanf [18] definem o PAYT como a conjugação de dois princípios de políticas ambientais, o princípio 'poluidor-pagador' e o conceito de responsabilidade partilhada.

O município alemão Aschaffenburg implementou o sistema PAYT há mais de 20 anos, em 1997 e tem hoje uma das maiores taxas de reciclagem registadas, 86%, e uma das taxas de produção de resíduos mais baixas fixadas em 55 kg per capita por ano [19]. Aschaffenburg não é caso único de sucesso na Alemanha, no entanto os números e o facto de estar implementado há tanto tempo com sucesso torna este município um caso de estudo. Atualmente o sistema PAYT no município de Aschaffenburg é implementado utilizando um sistema baseado no peso. Os contentores contêm todos um identificador RFID que é lido pelos sensores nos camiões de recolha e é feita a associação ao cidadão de forma a calcular a taxa a cobrar. À medida que os dados vão sendo recolhidos e processados, o município utiliza essa informação para otimizar os percursos dos camiões do lixo tendo em conta o padrão de utilização de cada contentor.

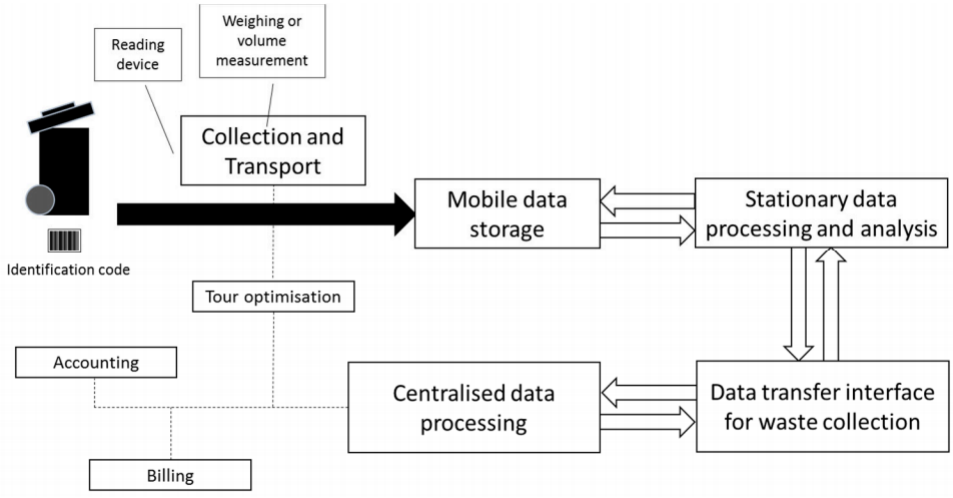


Figura 2.2: Processo de identificação e transferência de dados num sistema PAYT [19].

Na figura 2.2 temos um esquema adaptado da identificação dos resíduos e transferência de informação envolvida. Os camiões de recolha possuem um leitor de código de barras ou outra forma de identificador que lê o identificador do contentor e mede o peso ou volume de lixo do contentor. Depois essa informação é enviada através de uma ligação de internet móvel para os servidores, onde a mesma é processada, sendo associada ao cidadão ou conjunto de cidadãos para depois ser efetuada a faturação de

acordo com o lixo produzido num determinado espaço de tempo. Além disso o sistema neste caso processa informação para criar estatísticas de produção de lixo para otimizar as rotas de recolha de lixo de acordo com a produção típica da população, permitindo reduzir nos custos de recolha de resíduos.

Este município é uma referência a nível mundial, os resultados obtidos atingem uma taxa de reciclagem de 86%. Em contraste, a taxa média de reciclagem alcançada pelas implementações PAYT ronda os 70% noutros municípios e estes resultados devem-se aos seguintes fatores diferenciadores: utilização de um sistema baseado no peso, infraestruturas adequadas e uma grande consciencialização da população para a questão ambiental [19]. Na figura 2.3 podemos ver o decréscimo significativo da produção das 4 principais categorias de resíduos per capita (papel a laranja, vidro a azul-claro, plástico a azul-escuro e resíduos residuais a verde) desde a introdução do PAYT em 1997.

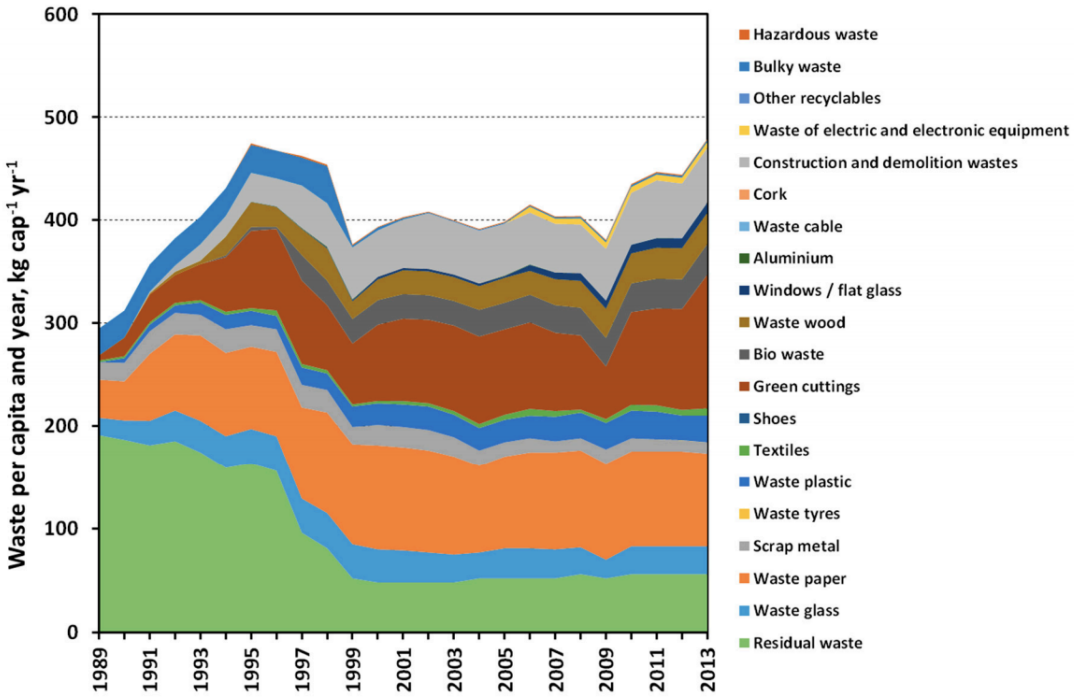


Figura 2.3: Evolução da reciclagem de diferentes tipos de resíduos ao longo dos anos na área com PAYT implementado [19].

O facto de haver uma clara estabilização da produção destes resíduos confirma o efeito positivo e duradouro da introdução do PAYT neste município, servindo de referência para todos os municípios que pretendam implementar o PAYT.

2.2 SEGURANÇA NOS DADOS DE SMART CITIES

Segurança contempla privacidade da informação e funcionamento consistente das aplicações. O foco desta dissertação é a segurança dos sistemas aplicados em *Smart Cities* e é importante ter a noção do impacto que as *Smart Cities* e a falta de segurança têm na privacidade dos cidadãos. Toda a atividade é registada, processada e armazenada pelos sistemas através de sensores, seja quando um cidadão passa em determinado sítio a determinada hora, por exemplo quando sai ou entra no local de trabalho ou na própria casa. Toda essa informação começa a ser tratada e analisada e se não for protegida devidamente e acedida por terceiros não autorizados podemos começar a ter problemas de segurança maiores do que aqueles que tínhamos e que tentamos solucionar com a implementação destes sistemas.

O problema torna-se ainda maior tendo em conta que a maioria dos sistemas de *Smart Cities* são implementados com recurso a parcerias público-privadas em que os governos locais recorrem a empresas especializadas para implementar e gerir os sistemas responsáveis pela gestão da infraestrutura inteligente da cidade. Nesses casos, os dados passam a estar na mão das empresas e isso pode levantar questões morais e políticas graves. Com tanta partilha de informação a probabilidade de acesso não autorizado ou de perda de dados para terceiros aumenta e pode trazer problemas de segurança e levar à desconfiança da população para com os sistemas de *Smart City* [5].

Como e quem é que pode utilizar a informação que é recolhida é uma questão fundamental para a segurança da informação. Tomamos como exemplo o recente caso do Facebook, que sem informar nem pedir autorização aos seus utilizadores partilhou dados pessoais de cerca de 87 milhões de utilizadores com a consultoria política Cambridge Analytica [20]. Embora os dados tenham sido dados como eliminados dos sistemas da consultoria, eles foram utilizados sem autorização e sem finalidade clarificada. Este é um exemplo perfeito para justificar a importância do novo RGPD da União Europeia, que tem o objetivo de garantir que os dados pessoais continuam a ser pessoais e que as pessoas tenham direitos sobre eles, como escolher se querem que os seus dados sejam partilhados com terceiros, utilizados para estatísticas de utilização ou para perfilamento de forma a fornecer um serviço mais personalizado. Após este acontecimento ter sido tornado público e com a entrada em vigor do novo RGPD e o Facebook atualizou a sua política de segurança e todos os utilizadores tiveram a opção de escolher as suas preferências de privacidade e de tomar conhecimento o que era processado e armazenado nos servidores do Facebook.

No entanto o Facebook não é uma *Smart City* e no caso dos sistemas *Smart City* os problemas de privacidade poderão ser mais complexos e difíceis de resolver. Uma vez

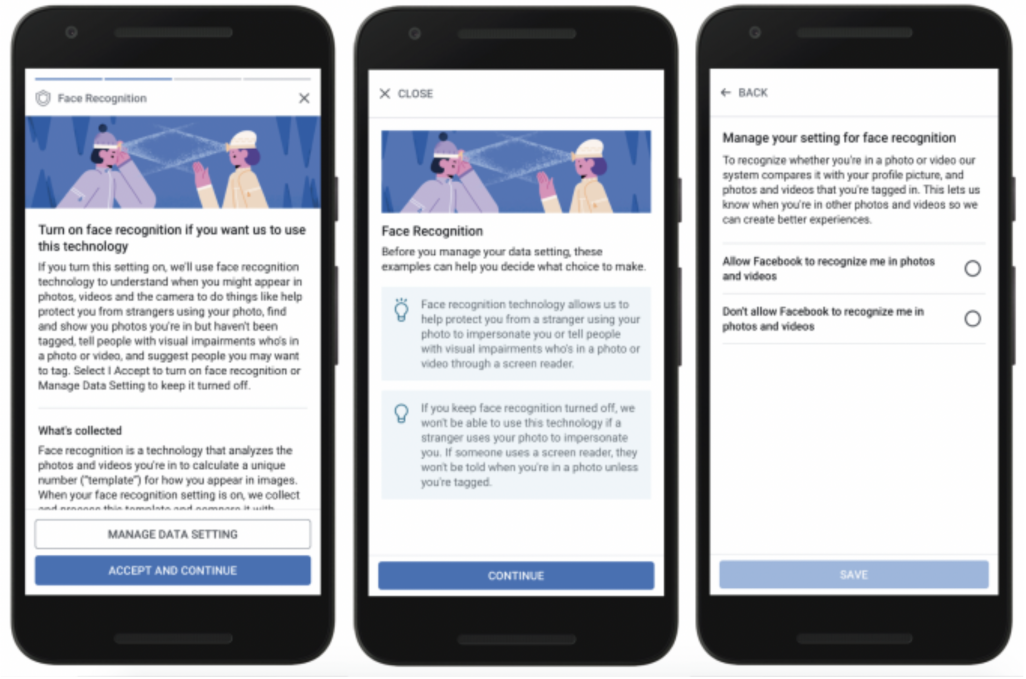


Figura 2.4: A mudança da política de segurança do Facebook [21].

que num sistema de *Smart City* existem vários dispositivos interligados, com várias bases de dados e entidades com acesso às mesmas, a privacidade dos dados pode facilmente ser comprometida se não for feito um controlo de acesso adequado e uniforme em todos os sistemas envolvidos. Tal como referido na secção anterior, o que define uma *Smart City* é o conjunto de 3 conceitos, IoT, *Big Data* e *Cloud*, em que cada um acarreta diferentes ameaças para a privacidade dos seus utilizadores o que torna as *Smart Cities* um caso de uso especial em termos de privacidade e proteção de dados [5].

2.2.1 PROBLEMAS DE SEGURANÇA E PRIVACIDADE NAS SMART CITIES

As *Smart Cities* definem-se como uma aplicação de IoT como vimos na secção 2.1 e a fundação Open Web Application Security Project (OWASP) que tem como objetivo apoiar o desenvolvimento de aplicações de confiança [22], definiu os 10 principais riscos de segurança e de privacidade que podem surgir no desenvolvimento deste tipo de aplicações. Os 10 principais riscos de segurança enumerados pela OWASP são interface web inseguras (1), autenticação insuficiente (2), serviços de rede inseguros (3), comunicação não encriptada (4), preocupações de privacidade (5), interface de *cloud* insegura (6), interface móvel insegura (7), configurações de segurança insuficientes (8), software e firmware inseguro (9) e segurança física insuficiente (10). Para cada uma delas a OWASP definiu diretrizes básicas e recomendações que têm o objetivo de ajudar a combater estes riscos de segurança [23].

Para os pontos 1, 6 e 7, autenticação insuficiente, interface de *cloud* e móvel inseguras respetivamente, a OWASP define as seguintes diretrizes e recomendações. As interfaces (*web*, *cloud* ou móvel) não devem aceitar palavras-passe fracas, ou seja, que tenham um número mínimo considerável de caracteres e a utilização de caracteres especiais e numéricos. Deve ser também permitido que a conta de acesso às interfaces possa ser bloqueada em caso de um incidente de segurança, permitir a alterações dos dados de acesso como nome de utilizador e palavra-passe, e implementar autenticação de dois fatores. Recomendações mais técnicas são garantir que a interface foi testada contra vulnerabilidades Cross-Site Scripting (XSS), SQL Injection (SQLi) e Cross-Site Request Forgery (CSRF) e o transporte de informação de forma encriptada (Transport Layer Security (TLS) ou Hypertext Transfer Protocol Secure (HTTPS) por exemplo).

Os riscos de segurança relacionados com autenticação insuficiente (ponto 2) podem ser minimizados com políticas simples que aumentam significativamente a segurança do sistema, tais como a obrigação de palavras-passe complexas, autenticação de dois fatores, sistemas de recuperação de palavras-passe, renovação obrigatória das palavras-passe e autenticação tendo em conta o papel do utilizador no sistema.

Relativamente aos serviços de rede (ponto 3) a recomendação é utilizar uma infraestrutura de rede previamente aprovada e testada que consiga lidar bem com exceções e ataques Distributed Denial of Service (DDoS). É também importante garantir que a informação seja transmitida de forma cifrada (ponto 4), seja utilizando protocolos de rede adequados ou a encriptação antes do envio da mesma.

O ponto 8, configurações de segurança insuficientes, é mais técnico e para evitar problemas de segurança relacionados é recomendado a adoção de medidas de segurança e de criptografia por defeito, ou seja, aquando o desenho da aplicação a segurança deve ser sempre tida em conta. Um exemplo disso pode ser a escolha do algoritmo de encriptação. Se o algoritmo base for o Advanced Encryption Standard (AES) pode haver opção de escolha entre o Advanced Encryption Standard 128 Bits (AES-128), Advanced Encryption Standard 256 Bits (AES-256) ou Advanced Encryption Standard 512 Bits (AES-512), sendo a decisão entre um ou outro conforme a quantidade de informação que se pretende anonimizar, com o AES-256 a dar maiores garantias em relação ao AES-128 de que não haverá textos cifrados iguais.

Um problema grande dos sistemas *Smart City* é o software dos dispositivos e os diferentes componentes de software que compõem a *Smart City* (ponto 9). Desde software de dispositivos inteligentes, sensores e servidores de processamento, há muito software que tem que ser mantido e atualizado constantemente. Essa manutenção em grande escala pode tornar-se problemática e é preciso garantir que todo o software pode

ser facilmente atualizado para evitar falhas de segurança por falta de atualização de software. Além do software também a segurança "física" tem que ser mantida (ponto 10), não para quedas ou danos ambientais, mas sim para acesso não autorizado. Servidores devem ter acesso restrito a pessoas autorizadas e sensores ou outros dispositivos inteligentes devem evitar ter por exemplo portas USB ou outras formas de interação expostas que possam comprometer a segurança dos mesmos.

Por fim, o ponto 5, que depende muito da implementação e execução de políticas de privacidade adequadas. A OWASP define algumas políticas básicas para diminuir os riscos de privacidade que os sistemas de informação acarretam consigo. Algumas destas políticas são, como poderemos ver na secção seguinte, também apresentadas pelo novo RGPD da União Europeia e pela certificação ISO 27001, uma referência Internacional para a gestão da Segurança da informação. Alguns exemplos dessas políticas são: recolher o mínimo possível de informação pessoal, garantir acesso apenas a pessoas autorizadas, anonimizar informação pessoal sempre que possível, dar a opção ao utilizador de escolher que dados são ou não recolhidos e processados.

Todos estes riscos de segurança e medidas para os diminuir atuam de maneira diferente e em pontos diferentes da arquitetura de um sistema *Smart City*. Wan J et al. definiram uma arquitetura de segurança de 4 camadas, composta pela camada de perceção, de rede, de suporte e de aplicação [7].

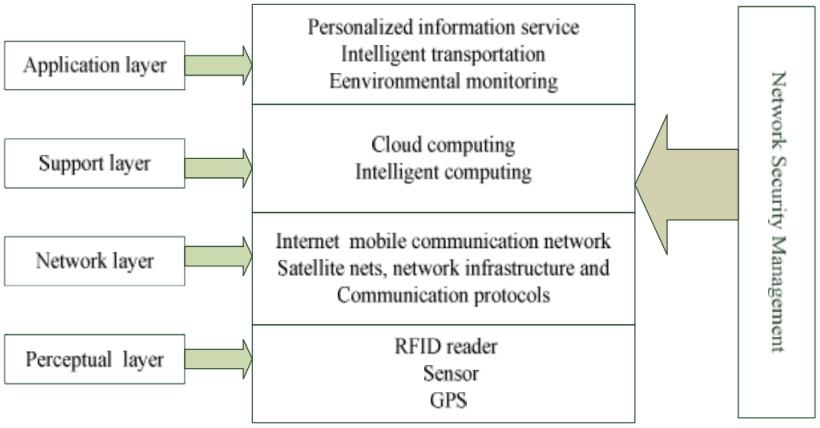


Figura 2.5: Camadas de segurança de sistemas de Internet das Coisas [7].

O trabalho desta dissertação não irá cobrir todos estes aspetos de segurança apresentados na figura 2.5, sendo o foco principal na camada de aplicação onde questões como as políticas de segurança, proteção da informação, controlo de acesso aos dados e à plataforma são as principais preocupações para ter um sistema seguro ao nível da camada da aplicação.

2.2.2 LEGISLAÇÃO E NORMAS

Nesta secção serão apresentados o novo Regulamento (União Europeia (UE)) 2016/679 do Parlamento Europeu sobre a proteção de dados (RGPD) e também o padrão internacional ISO 27001 que certifica a segurança de sistemas de gestão de informação. O RGPD tem especial importância porque entrou em vigor em 2018, ano de publicação desta dissertação e por isso traz novas responsabilidades e muitas mudanças em como a informação pessoal é gerida. O ISO 27001 é uma norma que cumpre o novo RGPD, sendo ainda mais extenso nos requisitos de segurança, e que pode ser usado como um guia de implementação para um sistema mais seguro mesmo não obtendo a certificação.

Regulamento Geral de Proteção de Dados

Com a introdução do novo Regulamento Geral de Proteção de Dados muitos sistemas existentes tiveram que alterar as suas políticas de segurança para que pudessem continuar em funcionamento. O objetivo do novo regulamento é garantir que os sistemas de informação e o tratamento de dados tenha princípios básicos de segurança, permitindo ao utilizador ter algum controlo sobre a informação pessoal que fornece na utilização destes sistemas. Esses princípios básicos definidos no artigo 5º do regulamento (UE) 2016/679 são a transparência, limitação de propósitos, minimização da recolha de dados, precisão dos dados, armazenamento limitado, integridade dos dados e responsabilidade sobre os dados. Estes princípios básicos refletem-se ao longo de todo o regulamento e vamos analisar alguns dos artigos mais relevantes do RGPD.

Um dos grandes objetivos é que os proprietários dos dados possam exercer os seus direitos sobre eles. Os artigos 13º a 22º estabelecem esses direitos, definindo regras básicas tais como a obrigatoriedade de dar a conhecer ao utilizador que informação, para que fim e durante quanto tempo é recolhida. Caso uma entidade tenha alguma informação pessoal relativa a um sujeito obtida através de terceiros deve sempre que pedido esclarecer quem, como e onde recolheu os dados, podendo o proprietário pedir até um documento com toda a informação pessoal recolhida e processada. A rede social Instagram por exemplo, desde a entrada em vigor do novo RGPD permite que seja enviado por e-mail toda a informação relativa a um utilizador que é armazenada e processada nos seus servidores.

São vários os direitos que o novo RGPD concede aos cidadão da União Europeia. O primeiro, apresentado nos artigos 13º e 14º, estabelece que todo o processamento de dados pessoais, quer sejam obtidos do proprietário ou de terceiros, deva ser o mais transparente possível. Ou seja, os utilizadores devem ter o conhecimento de informações

Figura 2.6: Opção de obter toda a informação de um utilizador da rede social Instagram. Retirado da aplicação móvel para Android.

de contacto do Encarregado de Proteção de Dados e da respetiva organização, e dos propósitos do processamento dos seus dados pessoais. Estes artigos estabelecem também que o proprietário dos dados deve tomar conhecimento do intervalo de tempo que os seus dados serão armazenados, e que os proprietários devem ter conhecimento do direito de ser esquecido (artigo 17º) e do direito de retificação dos seus dados (artigo 16º). No caso de uma destas ações ser executada, eliminação ou retificação dos dados pessoais, o utilizador tem o direito de ser notificado de qualquer uma destas ações (artigo 19º). Outro direito que contribui para a transparência do processamento de dados pessoais é o direito definido no artigo 15º, onde consta que cada utilizador tem o direito de a qualquer momento ter informação se os seus dados pessoais são processados, para que fim e durante quanto tempo.

O utilizador tem ainda o direito de escolher se os seus dados pessoais podem ser processados (artigo 18º), no entanto pode perder funcionalidades que muitos serviços prestam com recurso ao processamento automatizado de informação pessoal. No entanto este consentimento tem que ser dado pelo utilizador e o mesmo pode ser revogado a qualquer altura tendo ainda o direito de ser esquecido, desde que essa eliminação da informação não interfira com a utilização do sistema ou do modelo de negócio que depende do sistema de informação. Um exemplo disso são dados relativos a qualquer tipo de faturação que servem para justificar o valor cobrado aos clientes, mesmo quando estes já não forem clientes.

Por fim, o RGPD estabelece direitos no que diz respeito a processamento automatizado, com recurso a inteligência artificial por exemplo, em que são tomadas decisões por um sistema sobre os dados do utilizador. Estabelece também direitos em sistemas que utilizam dados pessoais para perfilamento dos utilizadores, criando padrões de

comportamento ou de identidade. Em ambos os casos, os direitos proveniente dos artigos 21º e 22º, salvaguardam os interesses dos utilizadores ao dar a opção de não serem sujeitos a estes tipos de processamento automatizados.

Estes são os direitos que o novo RGPD dá aos cidadãos da União Europeia e vão mudar a forma de desenhar aplicações e sistemas de informação no futuro. Além dos direitos, também são criadas novas responsabilidades nas empresas ou entidades que armazenam e processam informação pessoal. Segundo o artigo 37º do novo regulamento, a empresa ou entidade deve designar um encarregado de proteção de dados. As principais funções do encarregado de proteção de dados são informar trabalhadores que processam dados de como o devem fazer e quais as obrigações ao abrigo do RGPD, controla e é responsável pelo cumprimento com o RGPD e serve como ponto de contacto com as autoridades de controlo. A aplicação de normas certificadas na segurança de um sistema informático é uma recomendação definida no artigo 42º. A certificação internacional ISO/International Electrotechnical Commission (IEC) 27001 é um exemplo de uma certificação que pode ser obtida e que pode ajudar no cumprimento de muitos dos direitos e obrigações estabelecidas pelo RGPD.

Certificação ISO/IEC 27001

A certificação internacional ISO/IEC 27001 [3] ajuda a implementar e manter sistemas de informação e a sua segurança, mantendo a confidencialidade, integridade e disponibilidade da informação. O documento contém 10 capítulos curtos e explicativos sobre como obter a certificação e um anexo. O anexo A do documento do ISO/IEC 27001 contém uma tabela com os pontos de controlo que uma organização deve cumprir para obter esta certificação. Estes dividem-se em 14 categorias, com subcategorias em cada uma e vários pontos de controlo para cada subcategoria, totalizando 114 pontos de controlo (ver anexo C). Vamos analisar os pontos de controlo mais relevantes para o trabalho desta dissertação.

Uma das primeiras preocupações no desenho de um sistema de informação são os seus requisitos de segurança. Esta é uma questão a ter em conta para a obtenção da certificação (ver A.14.1), e os requisitos devem ser revistos sempre que houver alterações do sistema ou da informação que o mesmo processa e armazena. Estando estes requisitos definidos é depois mais fácil fazer o desenho e desenvolvimento aplicando a segurança por defeito (ver A.14.2) tornando os sistemas desde a sua raiz mais robustos e seguros. A certificação ISO 27001 define algumas obrigações que podem ser consideradas requisitos de segurança, como por exemplo a utilização de criptografia adequada (ver A.10.1), criar cópias de segurança (ver. A.12.3), fazer a classificação da informação (ver A.8.2), entre outros (ver anexo C).

Os pontos de controlo da categoria A.8.2 (classificação da informação) afirmam que a informação deve ser classificada de acordo com um esquema de rotulagem definido de acordo com a criticidade da informação que a organização possui. A North Atlantic Treaty Organization (NATO) definiu para classificar os seus documentos e informações os rótulos "NATO RESTRICTED", "NATO CONFIDENTIAL", "NATO SECRET", e "COSMIC TOP SECRET" [24]. Estas classificações da informação podem ajudar a definir a política de controlo de acesso (ver categoria A.9.1) que controla o acesso a funcionalidades e a informação (ver categoria A.9.4). A implementação do controlo de acesso à informação é feita de acordo com as políticas de segurança da organização tal como requerido pela certificação (ver A.5.1). Além do controlo de acesso à informação o acesso de utilizadores ao sistema também tem normas para cumprir de acordo com o ponto A.9.2, de forma a garantir uma gestão adequada dos utilizadores permitindo registar e eliminar utilizadores, e gerir os seus privilégios de acesso.

A certificação além destas questões mais técnicas define outros requisitos tais como as obrigações dos funcionários na manutenção da segurança da informação (ver A.7.1, A.7.2, A.7.3), segurança física dos dispositivos (ver A.11.2), gestão de recursos (ver A.8.1) entre outros. No entanto o foco principal do trabalho desenvolvido para esta dissertação, definir requisitos e políticas de segurança para gerir o acesso dos utilizadores ao sistema e aos dados. No próximo subcapítulo iremos analisar os modelos de controlos de acesso tradicionais e que poderão servir como base de uma solução de controlo de acesso que aja em função da classificação da informação.

2.3 MODELOS DE CONTROLO DE ACESSO

Neste subcapítulo serão apresentados modelos de controlo de acesso, desde os mais tradicionais aos mais modernos. Os modelos mais conhecidos que vamos analisar são o Controlo de acesso obrigatório (MAC), Controlo de acesso discricionário (DAC), Controlo de acesso baseado em papéis (RBAC), e Controlo de acesso baseado em atributos (ABAC). Por fim, serão apresentados algumas propostas de modelos que combinam alguns destes modelos mais clássicos originando modelos mais completos e versáteis.

Mandatory Access Control – MAC

Este modelo de controlo de acesso apresentado pela em 1985 pelo Departamento de Defesa do Estados Unidos no Trusted Computer System Evaluation Criteria, mais conhecido como o Orange Book (livro laranja), funciona com base em rótulos atribuídos

aos objetos ou dados [25]. Os rótulos atribuídos aos objetos são uma classificação hierárquica ou não hierárquica com pelo menos 2 níveis de classificação. O acesso ao objeto é autorizado ou não conforme os rótulos atribuídos ao sujeito que está a aceder. Numa classificação hierárquica se o sujeito tiver um rótulo igual ou superior ao do objeto pode lê-lo, e apenas pode escrever sobre o objeto se o seu nível hierárquico for menor ou igual ao do objeto. Numa classificação não hierárquica o sujeito apenas pode ler ou escrever o objeto se possuir todos os rótulos atribuídos ao objeto. Este modelo é mais simples de aplicar, mas pode não oferecer a mesma flexibilidade que do DAC.

Este tipo de controlo de acesso é aplicado por exemplo nos sistemas operativos Windows, denominado de 'Mandatory Integrity Control' que atribui diferentes níveis de classificação aos objetos, 'low', 'medium', 'high' e 'system' [26] [27]. Neste caso um processo com o rótulo 'low' não pode escrever um objeto cujo rótulo é 'medium' e vice-versa.

Discretionary Access Control – DAC

Segundo o já mencionado Orange Book, o DAC é definido pela separação entre sujeitos e objetos, em que o proprietário de um dado objeto define que utilizadores têm acesso ou não ao objeto. A atribuição dessas permissões pode também ser feita a um grupo de utilizadores, o que torna a gestão do controlo de acesso mais simples [25]. Existe por isso para cada objeto uma lista de utilizadores que tem permissão para executar ações sobre o mesmo, como ler, escrever ou executar no caso dos sistemas operativos.

Tomamos como exemplo de uma implementação do DAC o controlo de acesso do sistema de ficheiros do sistema operativo Linux. Quando há uma tentativa de acesso a um ficheiro, seja para escrever ou ler, é consultada a Lista de Controlo de Acesso (ACL) que é uma lista constituída com a lista de permissões do ficheiro em causa. Nessa lista está contemplado para todos os utilizadores do sistema operativo que permissões cada utilizador tem sobre o ficheiro a que pertence a ACL [28]. Podemos ver um exemplo disso mesmo no sistema operativo Linux Ubuntu na figura 2.7.

As permissões de acesso estão definidas na primeira coluna, por exemplo a pasta "Desktop" tem as permissões "drwxr-xr-x". A primeira posição define o tipo de ficheiro, neste caso "d" o que significa que é um diretório. Depois as restantes 9 posições dividem-se em 3 grupos de permissões, o primeiro para o proprietário do ficheiro ou pasta, o segundo para um grupo de utilizadores e o terceiro para os outros utilizadores. Portanto no exemplo anterior o proprietário tem as permissões "rwx", com "r" a ser leitura, "w" a ser escrita e "x" a ser execução. Para os utilizadores que pertencem

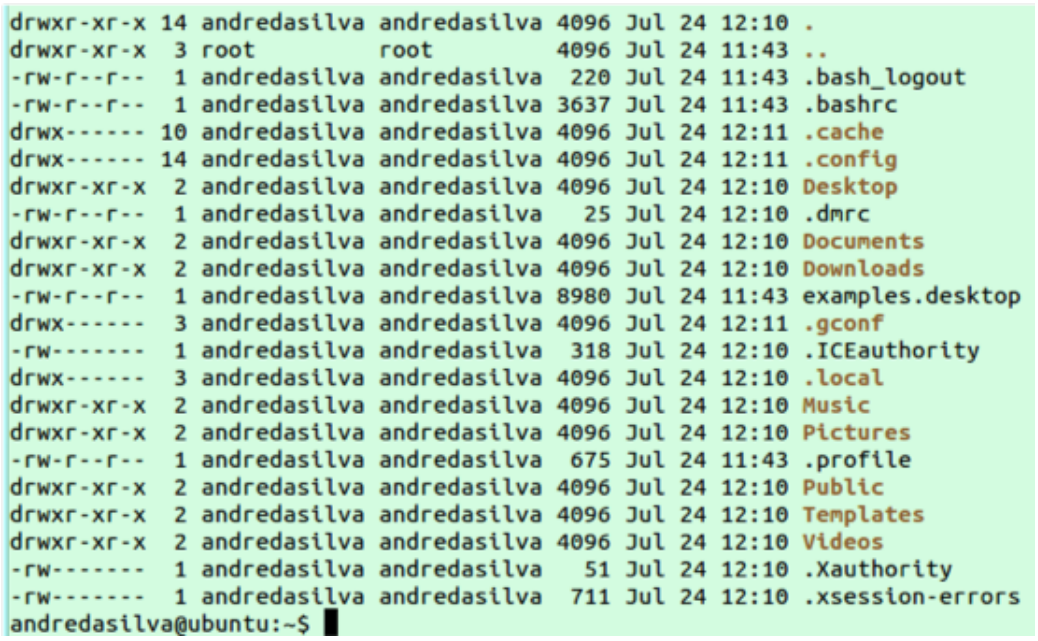


Figura 2.7: Exemplo de uma lista de permissões de acesso no Sistema Operativo Linux.

ao grupo do ficheiro as permissões são apenas de leitura e execução tal como para os restantes utilizadores, definido por "r-x".

Este modelo de controlo de acesso e o MAC são os modelos mais básicos e para algumas situações muito restritos por causa da sua simplicidade. Por isso começaram a surgir outras opções de controlo de acesso tais como o RBAC e o ABAC que vamos apresentar de seguida.

Controlo de acesso baseado em papéis – RBAC

O modelo de controlo de acesso RBAC definido por Sandhu et al. em 1996 [29], autoriza o acesso aos objetos com base no papel atribuído aos utilizadores. Esse papel que é atribuído tem associado a si um conjunto de privilégios, semelhante ao que acontece no DAC quando se atribui privilégios a um grupo de utilizadores.

Este modelo tem a vantagem de ser simples de gerir, mas pode tornar-se pouco flexível. Se o número de papéis que um administrador de sistema tem que criar para conseguir atribuir as permissões certas aumentar em demasia o modelo torna-se insustentável e muito mais difícil de gerir.

Controlo de acesso baseado em atributos – ABAC

Segundo Hu V et al. na publicação especial 800-162 do National Institute of Standards and Technology (NIST) [30] o ABAC é um modelo de controlo de acesso baseado nos atributos do sujeito. Na verdade, o RBAC pode ser visto como um caso

especial de ABAC em que o papel do utilizador é o atributo a ter em conta. A diferença entre ABAC e RBAC é que pode ser mais que um atributo a definir as políticas de acesso em vez de apenas o papel do utilizador. As políticas de acesso aos objetos no ABAC são definidas por expressões booleanas que avaliam os diferentes atributos do sujeito e decidem se há ou não autorização para acesso ao objeto.

O ABAC é um modelo de controlo de acesso recente e que é a base de muitos outros modelos que adotam o conceito de controlar o acesso com base nos atributos. De seguida iremos analisar várias propostas que assentam no modelo ABAC e ao qual juntam uma ou mais propriedades de outros modelos aqui apresentados.

Novos modelos de controlo de acesso

Os modelos de controlo de acesso que apresentados anteriormente, são modelos essenciais para implementar um mecanismo de controlo de acesso. No entanto, estes podem revelarem-se pouco flexíveis, no caso do RBAC e ABAC, ou serem de uma gestão difícil quando o número de utilizadores e regras é elevado, como acontece no MAC e DAC. Por esta razão, muitos autores começaram a estudar novos modelos de controlo de acesso, como por exemplo o modelo de controlo de acesso que utiliza uma combinação dos modelos ABAC e MAC proposto por Lawrence Kerr e Jim Alves-Foss [31]. O modelo combinado proposto é caracterizado da seguinte forma:

$$M = \{S,\ O,\ As,\ Ao,\ Ae,\ C,\ P,\ R\}$$

O modelo é composto por vários componentes. "S" é o conjunto de todos os sujeitos, "O" é o conjunto de objetos, "As", "Ao", "Ae" são os atributos dos sujeitos, objetos e ambiente respetivamente, "C" são as limitações impostas aos atributos, "P" são as operações que um determinado sujeito pode executar sobre um objeto e "R" são as regras de acesso baseadas nos atributos [31]. Portanto a operação "P" sobre um objeto "O" por parte de um sujeito "S" é ou não autorizada se os atributos "As", "Ao" e "Ae" constam na regra "R" que é aplicada ao objeto. É desta forma que as propriedades MAC deste modelo são aplicadas, havendo atributos obrigatórios para os pares de sujeitos e objetos autorizados a interagirem. Este modelo descrito por Kerr et al. é bastante generalizado e pode ser aplicado em vários tipos de aplicações onde até as condições do ambiente de execução podem ter influência na autorização do acesso aos objetos. O modelo permite até aplicar restrições aos valores que os atributos podem ter, permitindo que as regras sejam mais abrangentes e reduzir o seu número. Esta solução, no entanto, não tem em consideração o papel do utilizador que o modelo RBAC contempla. Poderia considerar-se que o papel fosse um atributo, mas não seria correto uma vez que o papel

por si só tem associado à partida um conjunto de privilégios pré-definidos [30]. Se o papel fosse visto como um atributo esses privilégios teriam que ser expressos com uma ou várias regras, o que num sistema com papéis que têm um conjunto de privilégios complexos levaria a que a gestão das permissões que se tornaria numa tarefa bastante difícil à medida que o número de atributos cresceria para descrever os papéis [31] [32]. No entanto um modelo puramente baseado em RBAC também não seria a solução pois isso levaria a uma quantidade enorme de papéis no sistema, o que torna a gestão das permissões complexa e pouco adaptável [32]. Por essa razão Kuhn et al. propõem uma outra combinação de modelos, um modelo baseado em papéis com a adição de funcionalidades dos modelos baseados em atributos, chamando-lhe o RBAC-A. Kuhn et al. definiram 3 abordagens do modelo RBAC-A para gerir as relações entre papéis e atributos: Dynamic roles, Attribute-centric e Role-centric. A abordagem Dynamic roles utiliza atributos dinâmicos como a hora para determinar o papel do utilizador de forma dinâmica enquanto que a abordagem Attribute-centric considera o papel do utilizador apenas como um atributo o que leva a perda da simplicidade que o RBAC original oferece. Por último temos a abordagem Role-centric que utiliza os atributos para restringir as permissões que um utilizador obteve através do papel que tem [32].

Para concluir este subcapítulo, importa referir que os principais modelos estarão sempre presentes de alguma forma, seja o MAC, DAC, RBAC ou ABAC. Mas como pudemos ver, existem outras formas de os utilizar e até combinar como Kerr et al. e Kuhn et al. propuseram [32], dando origem a modelos novos e mais completos. Na nossa solução, poderemos ter um ou vários modelos aplicados e, caso seja pertinente, propor uma variação destes modelos de forma a que a os requisitos sejam cumpridos na íntegra. Visto como podemos controlar o acesso à informação, no subcapítulo seguinte serão apresentadas formas de proteger e anonimizar informação que é transmitida e armazenada no sistema, não só para complementar possíveis falhas de segurança do controlo de acesso, mas também falhas que o controlo de acesso não consiga mitigar.

2.4 ANONIMIZAÇÃO E PROTEÇÃO DE DADOS

A anonimização dos dados é um dos primeiros passos para os proteger e existem várias técnicas para o fazer, nomeadamente funções de síntese e algoritmos criptográficos. As funções de síntese e os algoritmos criptográficos tornam ambas a informação ilegível, no entanto no caso das funções de síntese não é possível fazer o processo inverso e obter o texto ou informação original (figura 2.8), enquanto que os algoritmos criptográficos cifram o texto ou informação original numa cifra secreta que pode ser decifrada pelo

sujeito com acesso à chave de encriptação.

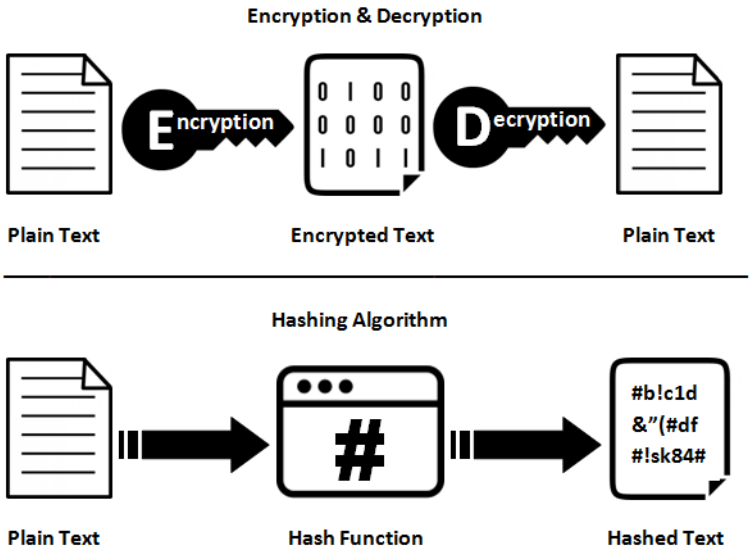


Figura 2.8: Funcionamento das funções de síntese vs. algoritmos de cifra [33].

Como podemos constatar na figura 2.8, a informação original é cifrada utilizando uma chave de encriptação, que depois é possível decifrar através dessa mesma chave obtendo assim a informação original, ou o respetivo par de chaves no caso das cifras assimétricas. O objetivo dos algoritmos criptográficos são também a proteção da informação, mas a possibilidade de dar acesso a quem o proprietário da informação achar adequado partilhando a chave com que foi cifrada a informação. Para isso existem as cifras simétricas e cifras assimétricas, em que as simétricas usam a mesma chave para cifrar e decifrar, enquanto que as assimétricas utilizam uma chave para a cifra e outra para a decifra [33]. Por outro lado, a função de síntese apenas faz a transformação num sentido sendo por isso uma função adequada para gerar e validar assinaturas digitais, calcular autenticadores de mensagens e armazenar informação como as palavras-passe dificultando a sua descoberta caso haja acesso ao armazenamento das mesmas [34]. Nas próximas secções, 2.4.1 e 2.4.2, iremos analisar algumas das funções e algoritmos mais relevantes atualmente.

2.4.1 FUNÇÕES DE SÍNTESE

Como referido, as funções de síntese podem ser úteis para assinaturas digitais, geração de autenticadores de mensagens e derivação de textos simples em conteúdo indecifrável. O funcionamento delas consiste em produzir valores de dimensão constante, para o mesmo algoritmo, a partir de conjuntos de textos variáveis produzindo para cada um desses conjuntos um valor único. Estas características derivam das três propriedades

que uma função síntese deve possuir para que seja uma função de qualidade e confiança [34]: resistência à descoberta de um texto original, resistência à descoberta de um segundo texto original e resistência à colisão. A primeira propriedade, resistência à descoberta de um texto original, significa que tendo um texto processado por uma função síntese é computacionalmente inviável descobrir o texto original. A segunda propriedade, resistência à descoberta de um segundo texto original, diz que dado um texto original e a sua síntese, não deve ser concretizável a obtenção de um outro texto original que produza a mesma síntese que o primeiro. Quando esta propriedade se verifica estamos perante um texto único e que pode ser utilizado como assinatura digital como se de uma impressão digital tratasse. E por fim, a última propriedade estabelece que uma função de síntese não deverá produzir o mesmo texto a partir de textos diferentes. Para que esta propriedade seja concretizada é importante a função de síntese ter uma saída com um número de bits mínimo, geralmente igual ou superior a 128 bits [34]

As funções de síntese mais conhecidas são o Message Digest 5 (MD5) e Secure Hash Algorithm 1 (SHA-1) [34], embora o sucessor do último, o Secure Hash Algorithm 2 (SHA-2) que é um conjunto de funções de síntese composto pelo Secure Hash Algorithm 224 (SHA-224), Secure Hash Algorithm 256 (SHA-256), Secure Hash Algorithm 384 (SHA-384) e Secure Hash Algorithm 512 (SHA-512), seja cada vez mais utilizado, tornando-se no padrão da indústria, pois em 2017 foi encontrada a primeira colisão de sínteses do SHA-1 [35].

Funções de síntese - Legado

Em 1992 R. Rivest publicou [36] a função de síntese MD5, função esta que vinha substituir o Message Digest 4 (MD4) por ser já pouco segura. O MD5 foi desenhado para ser mais seguro que o MD4 e por isso é um algoritmo um pouco mais lento que o seu antecessor. Esta função de síntese tem como entrada uma mensagem de tamanho variável e após o processamento da mensagem dá origem a uma mensagem de 128 bits. Hoje em dia a sua utilização já não é recomendada, uma vez que em 2005 foram encontradas colisões de sínteses [37] quando dois investigadores geraram sínteses idênticas para dois ficheiros de algoritmos substancialmente diferentes. Mais tarde no mesmo ano V. Klima apresentou um algoritmo que encontrava colisões de sínteses em 8 horas num processador Intel Pentium de 1.6 Ghz [38]. O MD5, no entanto, teve uma grande adoção por questões de performance na altura em que o desempenho nos processadores relevantes em 1996 chegava a ser 2 a 5 vezes mais rápido que o mais seguro e robusto SHA-1 [39]. Hoje em dia isso já não se verifica nos processadores modernos e o SHA-1 chega mesmo a ser mais rápido que o MD5 como

demonstrado nestes testes de desempenho executados num núcleo de um processador i7 a 2.60 Ghz [40]. Por isso nos últimos anos, principalmente desde a descoberta de colisões no MD5, o SHA-1 foi utilizado em larga escala, uma vez que garantia a segurança da informação e era de rápida execução. Esta função de síntese consegue produzir a partir de qualquer mensagem menor que 2^{64} bits uma síntese de 160 bits, sendo que na altura da publicação, 1995 [41], pensava-se ser impossível encontrar dois textos diferentes que pudessem produzir a mesma síntese. No entanto em 2017, uma investigação conjunta entre a Google e o Instituto CWI de Amsterdão, demonstrou ser possível gerar duas sínteses idênticas para dois documentos PDF diferentes. Esta descoberta veio demonstrar a necessidade de adotar como padrão outras funções de síntese, como o SHA-256 da família de sínteses SHA-2 [35].

Secure Hash Algorithm 2 - SHA-2

A família de funções de síntese SHA-2 foi apresentada pela primeira vez na publicação FIPS 180-2 que substituía a publicação referida anteriormente, a FIPS 180-1, em 2001. O documento válido atualmente é a publicação FIPS 180-4 onde tanto o SHA-2e o SHA-1 são apresentados [42]. O conjunto de funções que compõem o SHA-2 são as funções SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 e SHA-512/256. O que distingue essencialmente cada uma destas funções é o tamanho da mensagem a ser sintetizada e o tamanho da síntese produzida, com o SHA-1, SHA-224 e SHA-256 a aceitarem mensagens até 2^{64} bits, produzindo sínteses de 160, 224 e 256 bits respetivamente. As funções SHA-384 e SHA-512 produzem sínteses de 384 e 512 bits respetivamente a partir de mensagens até 2^{128} bits, e as funções SHA-512/224 e SHA-512/256 funcionam da mesma forma que o SHA-512, mas produzem sínteses mais pequenas, 224 e 256 bits respetivamente.

Algoritmo	**Tamanho da mensagem**	**Tamanho do bloco**	**Tamanho das palavras**	**Tamanho da síntese**
SHA-1	$<2^{64}$	512 bits	32 bits	160 bits
SHA-224	$<2^{64}$	512 bits	32 bits	224 bits
SHA-256	$<2^{64}$	512 bits	32 bits	256 bits
SHA-384	$<2^{128}$	1024 bits	64 bits	384 bits
SHA-512	$<2^{128}$	1024 bits	64 bits	512 bits
SHA-512/224	$<2^{128}$	1024 bits	64 bits	224 bits
SHA-512/256	$<2^{128}$	1024 bits	64 bits	256 bits

Tabela 2.1: Tabela de comparação das várias funções de síntese.

2.4.2 ALGORITMOS CRIPTOGRÁFICOS

Os algoritmos criptográficos, ou cifras modernas como também são conhecidos, podem ser caracterizados segundo o tipo de chave que utilizam. Os algoritmos que serão apresentados de seguida utilizam uma chave de cifra simétrica no Data Ecnryption Standard (DES) e AES, e uma chave assimétrica no caso do Rivest, Shamir, & Adleman (RSA). Nos algoritmos com uma chave simétrica a chave para cifrar e decifrar a informação é a mesma. Isso significa que a confidencialidade da informação é garantida apenas por uma entidade, ou por mais que 1 desde que a chave secreta seja partilhada entre todas. Por outro lado, os algoritmos que utilizam chaves assimétricas, usam um par de chaves: uma chave pública e uma chave privada, sendo impossível calcular a partir da chave pública a chave privada correspondente. Estes algoritmos cifram a informação com a chave pública do destinatário e decifram com a chave privada do mesmo, garantindo que apenas o destinatário pode aceder ao conteúdo. Também é possível fazer o processo inverso, cifrar com a chave privada e decifrar com a chave pública, mas nesse caso como a chave pública é conhecida serve apenas como forma de assinatura digital [34].

DES – Data Encryption Standard

Num concurso público lançado pelo National Bureau of Standards em 1974 para selecionar um algoritmo criptográfico com chave simétrica para ambientes comerciais, foi selecionada a proposta que viria a ser conhecida como DES [34]. O DES é um algoritmo com chave simétrica por blocos que utiliza blocos de 64 bits e chaves de 56 bits. Em termos de segurança o DES pode hoje em dia ser comprometido em poucos dias com recurso a um computador pessoal através de ataques de busca exaustiva de chaves, que consiste numa estratégia de tentativa e erro para todas as chaves possíveis até encontrar a chave certa. Isto é possível uma vez que as chaves que o DES utiliza são de 64 bits, ou seja 2^{64} possibilidades, um número relativamente baixo para os processadores disponíveis hoje em dia. Por essa razão são raras as aplicações que ainda utilizam este algoritmo uma vez que é fácil de comprometer [43].

AES – Advanced Encryption Standard

O Advanced Encryption Standard (AES), também conhecido como o algoritmo de Rijndael, foi apresentado em 2001 na publicação FIPS-197, é um algoritmo que cifra e decifra blocos de 128 bits com chaves que podem ter 128, 192 ou 256 bits de comprimento [44]. Este algoritmo é em termos de desempenho mais lento quando comparado com o AES, pois utiliza chaves pelo menos 2 vezes maiores e pode processar blocos com o dobro do tamanho [45]. Em termos de segurança o AES não foi até

à data comprometido e é por isso uma ótima escolha para manter os dados seguros, especialmente se a chave for de 256 bits, que demorará ainda mais a ser comprometida do que uma chave de 128 bits.

RSA

Este algoritmo é diferente dos dois anteriores pois trata-se de um algoritmo assimétrico, ou seja, em vez de uma chave única utiliza um par de chaves, uma pública e uma privada, para encriptar e decifrar. O RSA foi apresentado em 1977 pelos autores Rivest, Shamir e Adleman [34] e é hoje um padrão de segurança. É preciso ter, no entanto alguns cuidados para evitar ataques de descoberta das chaves públicas ou privadas e utilizar um número de bits suficientemente alto para não comprometer a segurança da informação. A NIST recomenda a utilização de chaves de 2048 bits, estimando que até 2030 este tamanho das chaves seja suficiente para garantir a segurança [44].

Estas funções de síntese e de cifra, oferecem uma vasta opção de proteger a informação para os mais variados casos de uso. Seja para armazenamento seguro, transporte seguro ou até mesmo assinatura digital e verificação de integridade dos dados.

2.5 SOLUÇÕES RELEVANTES

Neste subcapítulo serão apresentadas algumas soluções que podem ser relevantes para a solução que esta dissertação irá apresentar. Nas primeiras secções serão apresentadas formas de implementar e aplicar autorização e controlo de acesso em *Frameworks* utilizadas a grande escala, nomeadamente o Django, eXtensible Access Control Markup Language (XACML) e ASP.NET. Depois iremos fazer uma breve análise ao Wordpress de forma a perceber como é que são geridos os utilizadores e os seus privilégios num dos sistemas para gestão de aplicações web mais utilizados no mundo. Será depois analisado como poderá ser feita a gestão e armazenamento de palavras passe de forma a que as mesmas se mantenham seguras em ambientes profissionais. Por fim iremos fazer uma breve análise sobre boas práticas de cópias de segurança e que soluções existem para implementá-las.

2.5.1 DJANGO

O Django é uma *Framework Open-Source* para o desenvolvimento de aplicações web escrita em Python, que permite em poucos passos ter uma página web a funcionar com autenticação básica, administração de conteúdo ou mapa da página. Aplicações

web como o Instagram, Pinterest ou Open Stack, foram desenvolvidas com ajuda da Framework Django, o que demonstra todo o seu potencial [46]. Para a realização deste trabalho, vamos analisar como é que esta *Framework* utilizada a larga escala lida com a gestão de utilizadores e as respetivas permissões.

Para o nosso estudo, interessa saber como é que esta *Framework* gere os utilizadores e como é que lhe atribui permissões, para conceder ou bloquear acesso a determinado conteúdo. Em primeiro lugar, vamos analisar como é que os utilizadores são caracterizados pelo Django. A cada utilizador são associados vários campos de informação, tais como username, primeiro e último nome, E-Mail, palavra passe, grupos a que pertence, as permissões, data do último acesso, data de registo. Por fim, são ainda associados três campos booleanos, '`is_active`' que indica se a conta está ativa, '`is_staff`', que indica se o utilizador é parte do *staff* e com isso pode aceder à página de administração, e '`is_superuser`', que indica o utilizador com acesso total, possuindo todas as permissões de acesso [47]. Para consultar e editar estas informações existem várias funções pré-definidas, como por exemplo a função '`has_perm()`' que consulta uma determinada permissão para o utilizador. O sistema de permissões do Django permite atribuir permissões a utilizadores ou grupos de utilizadores. As permissões pré-definidas na Framework podem restringir o acesso a objetos a utilizador com a permissão '`view`' ou '`change`', a ação de adicionar utilizadores está limitada à permissão '`add`', editar utilizadores ou outras informações apenas é autorizado a utilizadores com a permissão '`change`' e '`delete`' é a permissão que permite eliminar objetos [48]. É possível adicionar mais permissões para novas funcionalidades que sejam implementadas através da Framework, e esta tem formas de fazer o controlo de acesso com base nas permissões que existem.

A partir destas permissões, é possível aplicar o controlo de acesso a objetos específicos dentro da aplicação Web. Este controlo de acesso pode ser executado através de decoradores Python [49]. Os decoradores Python são funções que envolvem as respetivas funções decoradas e podem assim alterar o seu comportamento sem que seja alterado qualquer linha de código [50]. Estes são uma ferramenta muito poderosa, pois permite adicionar ou alterar funcionalidades de um código já ultrapassado e mantê-lo simples e legível. No caso do Django, existem 3 decoradores diferentes para executar o controlo de acesso e autorização a conteúdos. O decorador mais comum será o '`@login_required`', que redireciona qualquer utilizador sem login efetuado para o Universal Resource Locator (URL) que contém a página de autenticação da aplicação Web (ver bloco de código 1).

Este decorador pode, no entanto, ser alterado e redirecionar o utilizador para uma

```
from django.contrib.auth.decorators import login_required

@login_required
def my\_view(request):
        ...
```

Bloco de código 1: Utilização do decorador login_required() [49].

outra página qualquer, passando o URL como argumento ao decorador como vemos no bloco de código seguinte 2.

```
from django.contrib.auth.decorators import login_required

@login_required(login_url='/accounts/login/')
def my_view(request):
        ...
```

Bloco de código 2: Utilização do decorador login_required() com passagem de um parâmetro [49].

Além deste existe um outro decorador, o '`@user_passes_test`' que executa uma função passada como argumento ao decorador que pode executar qualquer tipo de teste ou uma simples pergunta para confirmar a identidade do utilizador. No bloco de código 3 podemos ver um exemplo simples, em que o teste é verificar que o e-mail do utilizador que pretende aceder à função '`my_view`' deve terminar em '`@example.com`'.

```
from django.contrib.auth.decorators import user_passes_test

def email_check(user):
        return user.email.endswith('@example.com')

@user_passes_test(email_check)
def my_view(request):
        ...
```

Bloco de código 3: Utilização do decorador user_passes_test() [49].

Para além de um possível argumento, como demonstrado, é possível também indicar o URL que redirecione o utilizador para a página de login através do argumento posicional '`login_url`'.

Por fim, temos o decorador que define todo o controlo de acesso para os utilizadores com login efetuado, que é o decorador '`@permission_required`'. É importante referir que a mesma função pode ter mais que um decorador, pode ter estes 3 ou apenas 1, cabe ao programador decidir qual a forma mais segura de proteger o objeto. Este decorador recebe como argumento o nome da permissão que os utilizadores devem possuir para

ter autorização de acesso à função. No bloco de código 4, a função decorada só é acessível a utilizadores que possuem a permissão '`polls.can_vote`'. Caso contrário, são redirecionados para o URL indicado '`login_url`'.

```
from django.contrib.auth.decorators import permission_required

@permission_required('polls.can_vote', login_url='/loginpage/')
def my_view(request):
        ...
```

Bloco de código 4: Utilização do decorador permission_required() [49].

Este segundo argumento, '`login_url`', não é obrigatório e quando não declarado reencaminha o utilizador para a página de login pré-definida.

Esta forma de controlo de acesso, em termos programáticos é bastante poderosa pois qualquer programador sem o conhecimento do código de controlo de acesso pode aplicar. Além disso, permite que futuras alterações à política de controlo de acesso sejam possíveis sem que o código da função protegida seja alterado, e ajuda além disso a tornar o código mais legível o que num grupo de trabalho com 20 ou 30 programadores é bastante importante para tornar o trabalho mais eficiente. No entanto, em termos das políticas propriamente ditas, a sua gestão pode ser mais complexa uma vez que o modelo de controlo de acesso se baseia fundamentalmente no modelo DAC. O DAC foi desenhado de forma a que seja o proprietário dos objetos a definir as suas políticas e a atribuir as permissões de acesso aos utilizadores [26]. Esta estratégia é boa para os utilizadores controlarem o acesso ao seu conteúdo, é por isso que funciona bem em sistemas operativos, ou aplicações como redes sociais. No entanto, existem casos de uso em que a única entidade a atribuir permissões de acesso são os administradores de sistema. Nesse caso, a gestão poderá ser mais complexa pois 4 ou 5 permissões podem não ser suficientes para cobrir todas as situações e nem sequer se está a considerar os objetos individuais que cada função protegida está a aceder, apenas a funcionalidade da mesma. Se, por exemplo, um contexto de utilizador for constituído por 3 atributos, para o mesmo objeto poderá ser necessário criar até 6 permissões diferentes. Num sistema a larga escala isto pode ser problemático. Este caso é pouco provável, e se acontecesse, o Django permite alterar a forma como funciona o controlo de acesso uma vez que é uma *Framework* escalável e configurável. Para o nosso caso de estudo, é importante reter a forma como é aplicado o controlo de acesso. A utilização de decoradores que executam uma determinada função de controlo de acesso pode ser útil no nosso caso de uso uma vez que todo o sistema foi desenvolvido em Python. Aplicado ao sistema LIFE PAYT, poderemos ter um ou vários decoradores para as diferentes funções que estarão acessíveis aos utilizadores autenticados e não autenticados.

2.5.2 ASP.NET

O ASP.NET é uma *Framework* para o desenvolvimento de aplicações web, escrita em .NET e desenvolvida pela Microsoft. Com esta *Framework Open-Source* é possível criar aplicações web baseadas em HTML5, CSS e JavaScript [51]. Em termos de controlo de acesso, esta *Framework* permite executar a autorização ao acesso de objetos de 4 formas distintas: convenções de autorização de páginas *Razor*, autorização simples, autorização baseada em papéis, autorização baseada em alegações e autorização baseada em políticas [52].

As convenções de autorização de páginas *Razor*, executam o controlo de acesso a cada uma das páginas de forma individual. No excerto de código seguinte temos todos os exemplos possíveis de como fazer o controlo de acesso desta forma.

```
services.AddMvc()
.AddRazorPagesOptions(options =>
{
        options.Conventions.AuthorizePage("/Contact");
        options.Conventions.AuthorizeFolder("/Private");
        options.Conventions.AllowAnonymousToPage("/Private/PublicPage");
        options.Conventions.AllowAnonymousToFolder("/Private/PublicPages");
})
.SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
```

Bloco de código 5: Autorização no ASP.NET com recurso a convenções Razor [53].

Neste excerto de código 5 podemos encontrar 4 autorizações distintas para os utilizadores. Na linha 4, '`options.Conventions.AuthorizePage("/Contact")`', a autorização configurada é que a página da aplicação web '`/Contact`' requer autorização, sendo esta concedida a utilizadores que tenham login efetuado. Na linha seguinte, '`options.Conventions.AuthorizeFolder("/Private")`', aplica-se a mesma lógica mas neste caso para uma pasta específica, a pasta '`/Private`'. As duas linhas seguintes configuram a página e pasta respetivamente de forma a que sejam acessíveis a utilizadores anónimos [53].

A autorização simples é aplicada através de um atributo do '`AuthorizeAttribute`', que quando aplicado a uma determinada ação ou controlador limita o acesso a utilizadores com login efetuado. Existem dois atributos para controlar o acesso a controladores ou ações, '`Authorize`' que indica a necessidade de ser um utilizador com login, e '`AllowAnonymous`' que permite que qualquer utilizador, com ou sem registo, possa aceder. Nos excertos de código seguintes, 6 7 8 encontramos 3 possibilidades de aplicar esta forma de controlo de acesso [54].

Outra forma relativamente simples de aplicar o controlo de acesso nesta *Framework*,

```
[Authorize]
public class AccountController : Controller
{
        public ActionResult Login()
        {
        }

        public ActionResult Logout()
        {
        }
}
```

Bloco de código 6: Autorização simples no ASP.NET aplicada ao controlador [54].

```
public class AccountController : Controller
{
        public ActionResult Login()
        {
        }

        [Authorize]
        public ActionResult Logout()
        {
        }
}
```

Bloco de código 7: Autorização simples no ASP.NET aplicada a uma ação [54].

```
[Authorize]
public class AccountController : Controller
{
        [AllowAnonymous]
        public ActionResult Login()
        {
        }

        public ActionResult Logout()
        {
        }
}
```

Bloco de código 8: Autorização simples no ASP.NET aplicada ao controlador e ação [54].

é através do papel de utilizador. Cada utilizador tem associado a si pelo menos um papel de utilizador, sendo os mais básicos o papel de administrador e o de utilizador [55]. Para executar este controlo de acesso, existem várias formas de o aplicar. É possível atribuir a um controlador ou ação uma lista de papéis aos quais os utilizadores devem pertencer, sendo que podemos escolher se o utilizador deve pertencer a todos, ou a pelo menos um papel de utilizador da lista definida. No seguinte exemplo 9 temos o caso

em que o utilizador deve pertencer pelo menos a um dos papéis definidos, neste caso, 'HRManager' ou 'Finance', ou mesmo ambos.

```
[Authorize(Roles = "HRManager,Finance")]
public class SalaryController : Controller
{
}
```

Bloco de código 9: Autorização baseada em vários papéis no ASP.NET - Deve pertencer a pelo menos um [56].

Se, no entanto, o programador pretender que o utilizador pertença a todos os papéis de utilizador definidos para um controlador ou ação, deve atribuir vários atributos 10, ao invés de apenas um como no exemplo anterior.

```
[Authorize(Roles = "PowerUser")]
[Authorize(Roles = "ControlPanelUser")]
public class ControlPanelController : Controller
{
}
```

Bloco de código 10: Autorização baseada em vários papéis no ASP.NET - Pertence a todos obrigatoriamente [56].

Neste excerto de código 10, o utilizador tem que obrigatoriamente pertencer aos papéis de utilizador 'PowerUser' e 'ControlPanelUser', caso contrário o acesso é negado ao controlador ou ação a que foram atribuídos estes atributos. Esta forma de controlo de acesso, é também possível ser expressa através da definição de políticas. Embora o controlo de acesso por políticas seja uma forma distinta de executar o controlo de acesso, é também possível aplicar políticas em que são definidos quais os papéis de utilizador permitidos para um controlador ou ação [55].

Antes de fazer uma análise sobre o controlo de acesso baseado em políticas, vamos estudar o controlo de acesso baseado em alegações. As alegações são configuradas como políticas e por isso cabe ao programador configurar as políticas de acordo com as alegações que se pretendem verificar. Uma alegação é atribuída e reconhecida por uma entidade de confiança a um sujeito, sendo que essa alegação é uma característica do sujeito, como por exemplo a data de nascimento, ou número de identificação fiscal que no caso dos cidadãos é reconhecida e atribuída nos documentos de identificação por uma entidade ligada ao estado [57]. Para implementar o controlo de acesso por alegações, é necessário registar uma política e associar-lhe qual a alegação que o sujeito deve apresentar. No exemplo seguinte 11, a política chamada 'Founders' define que devem possuir o número de empregado, 'EmployeeNumber', entre 1 e 5.

```
public void ConfigureServices(IServiceCollection services)
{
services.AddMvc();

services.AddAuthorization(options =>
{
        options.AddPolicy("EmployeeOnly", policy =>
                policy.RequireClaim("EmployeeNumber"));
});
}
```

Bloco de código 11: Autorização baseada em alegações no ASP.NET - configuração da alegação [57].

Para definir os controladores ou ações que devem cumprir esta política, é utilizada a seguinte declaração 12, semelhante ao que acontecia na autorização por papéis de utilizador.

```
[Authorize(Policy = "EmployeeOnly")]
public IActionResult VacationBalance()
{
        return View();
}
```

Bloco de código 12: Autorização baseada em alegações no ASP.NET - aplicação da regra da alegação 'EmployeeOnly' [57].

Esta forma de autorização, apesar de aplicar políticas de segurança, é apresentada à parte da autorização por políticas pois é uma forma muito simplificada quando comparada com o potencial que a autorização por políticas oferece. Para registar uma política de autorização aplica-se o mesmo princípio que no caso anterior, registando a política na função '`ConfigureServices`' [56].

```
public void ConfigureServices(IServiceCollection services)
{
        services.AddMvc();

        services.AddAuthorization(options =>
        {
                options.AddPolicy("AtLeast21", policy =>
                policy.Requirements.Add(new MinimumAgeRequirement(21)));
        });
}
```

Bloco de código 13: Autorização baseada em políticas no ASP.NET - configuração da política [56].

Aqui começam as diferenças para a autorização por alegações, ao invés de aplicar a uma política uma alegação aplica-se um requisito. Esse requisito no caso

acima, é o '`MinimumAgeRequirement(21)`', que representa uma chamada à função '`MinimumAgeRequirement`' com o parâmetro '21'. Ou seja, qualquer controlador ou ação sujeito a esta política requer que o utilizador tenha pelo menos 21 anos. A função '`MinimumAgeRequirement`' está definida de forma a que receba qualquer parâmetro o que significa que é possível criar mais que uma política com a mesma função alterando apenas o parâmetro [56]. No excerto de código 14 podemos ver a definição de 'MinimumAgeRequirement'.

```
using Microsoft.AspNetCore.Authorization;

public class MinimumAgeRequirement : IAuthorizationRequirement
{
        public int MinimumAge { get; private set; }

        public MinimumAgeRequirement(int minimumAge)
        {
                MinimumAge = minimumAge;
        }
}
```

Bloco de código 14: Autorização baseada em políticas no ASP.NET - Função que implementa a política 'MinimumAgeRequirement' [56].

Tal como este requisito mais simples, é possível criar requisitos mais complexos e que verifiquem até mais que um parâmetro o que pode levar a criação de políticas de controlo de acesso que podem resolver os mais variados casos de uso.

O ASP.NET permite muitas formas de autorização e de aplicação das mesmas. Um pouco semelhante ao Django, tem a capacidade de restringir certos controladores e ações a utilizador registados ou anónimos através de um atributo para o respetivo elemento. Isso acontece de forma semelhante no Django, onde são utilizados decoradores para indicar a função sujeita à política associada ao decorador. Mesmo a aplicação de políticas e de autorização por papéis de utilizador é semelhante à aplicação de decoradores no caso do Python, o que torna o código mais legível e simples de alterar no futuro.

2.5.3 EXTENSIBLE ACCESS CONTROL MARKUP LANGUAGE – XACML

O XACML é uma framework de controlo de acesso que vai de encontro ao que o modelo base, o ABAC, define utilizando os atributos do sujeito, objetos, ações e ambiente de execução para aplicar as políticas e regras de controlo de acesso. O XACML é composto por 3 componentes. O primeiro é a linguagem de políticas XACML que especifica os requisitos de controlo de acesso utilizando regras e políticas. Estas

são expressas em função do sujeito, objeto, ação e variáveis de ambiente. O segundo componente é o protocolo de pedido e resposta XACML, que faz o pedido ao mecanismo de decisão que vai avaliar o pedido com base nas regras e políticas de controlo de acesso definidas. Por último temos a arquitetura de referência XACML, que define como aplicar módulos de software para aplicar as políticas e como aplicar o mecanismo de decisão [58].

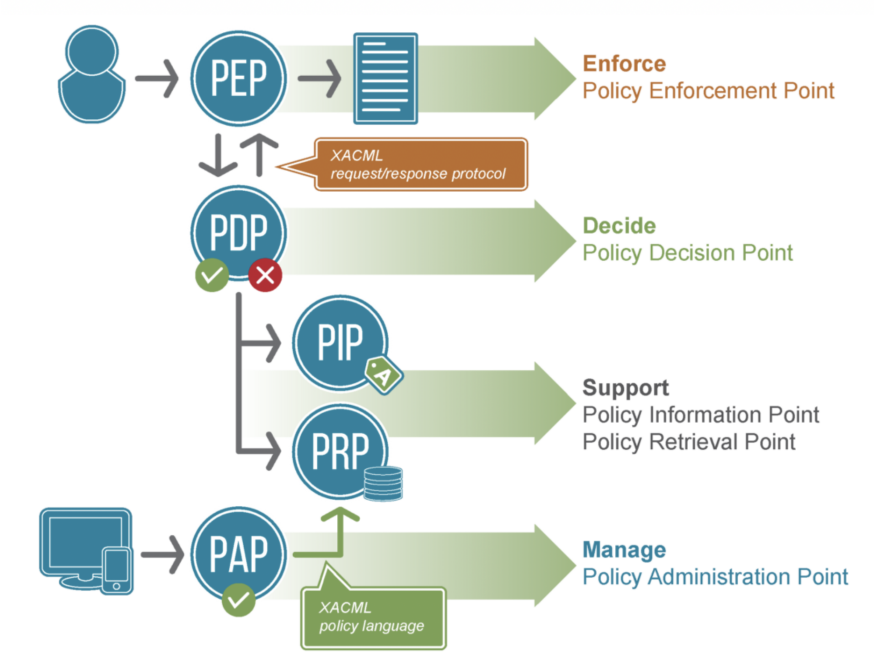


Figura 2.9: Arquitetura base da Framework XACML [59].

Esta figura representa essa arquitetura de referência, onde encontramos vários módulos de software como o Policy Decision Point (PDP), que faz a avaliação das políticas para os pedidos com origem no Policy Enforcement Point (PEP). O PDP toma a decisão de autorizar ou não com base na informação proveniente do Policy Information Point (PIP), e nas versões iniciais do Policy Retrieval Point (PRP) que nas versões mais recentes deixou de ser definido. O PRP era utilizado como um ponto de armazenamento de políticas que eram aplicadas no PEP e podia ser definido numa base de dados ou ficheiro. O mesmo se aplica ao PIP, que armazena os atributos que uma determinada política de controlo de acesso XACML deve avaliar para que a autorização de acesso seja concedida. Por fim temos o Policy Administration Point (PAP), que é o componente nesta arquitetura que é responsável pela gestão das políticas e das regras das mesmas [59]. Na figura 2.10 podemos ver o fluxo de informação que esta arquitetura oferece.

Neste fluxo de informação, podemos encontrar a seguinte interação: (1) um utilizador faz o pedido para visualizar o objeto '`# 123`', (2) o PEP que recebe o pedido reencaminha

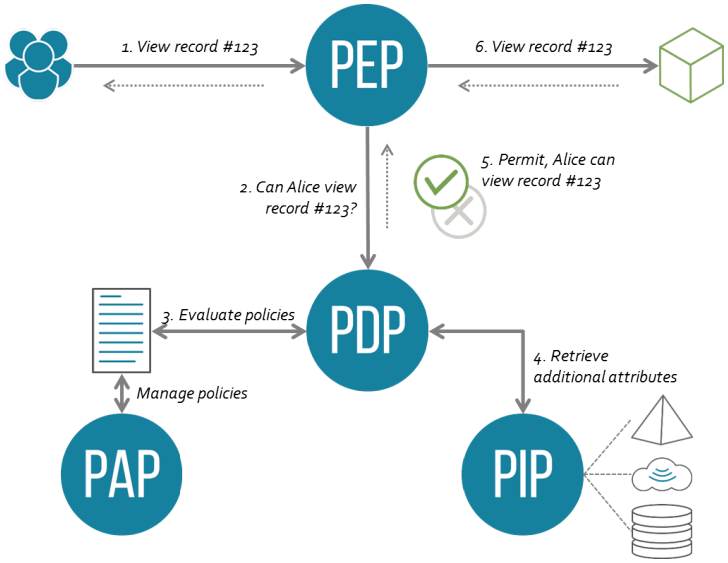


Figura 2.10: Fluxo de informação implementado pela Framework XACML [60].

o mesmo para o PDP que por sua vez irá consultar as políticas e avaliar o pedido (3); sempre que necessário o PDP irá recolher no PIP atributos que possam ser necessários para a tomada de decisão (4); o PDP acaba por autorizar o acesso ao objeto '`# 123`' (5) que o PEP irá apresentar (6) uma vez que recebeu do PDP a autorização.

A solução que a *Framework* XACML propõe, demonstra ser relevante acima de tudo devido à sua arquitetura e em como são concedidas as autorizações e geridas as políticas de acesso. Uma vez que toda a comunicação se baseia em eXtensible Markup Language (XML), a aplicação direta desta *Framework* por vezes não é possível. É, no entanto, importante ter um fluxo de informação, seja este apresentado pelo XACML ou outro, que defina claramente como se processa a gestão, aplicação e verificação de políticas de controlo de acesso.

2.5.4 WORDPRESS

O Wordpress é um Content Management System (CMS) bastante flexível, que permite em poucos passos a criação de blogues e páginas web simples. Este CMS é baseado em componentes PHP e suportada por bases de dados em MySQL [61]. É possível também para programadores mais experientes criar aplicações web e portais mais complexos disponibilizando *Frameworks* e *plugins* que permitem criar soluções à medida [62]. Atualmente mais de 31% das páginas web são desenvolvidas na plataforma Wordpress o que demonstra bem a sua flexibilidade e potencial. O nosso foco aqui será perceber como é que uma plataforma tão popular gere os utilizadores que se registam nos blogues e páginas criadas através do Wordpress.

A seguinte figura mostra o painel de gestão de utilizadores pré-definido do Wordpress, que permite alterar os papéis de utilizador num contexto de blogue.

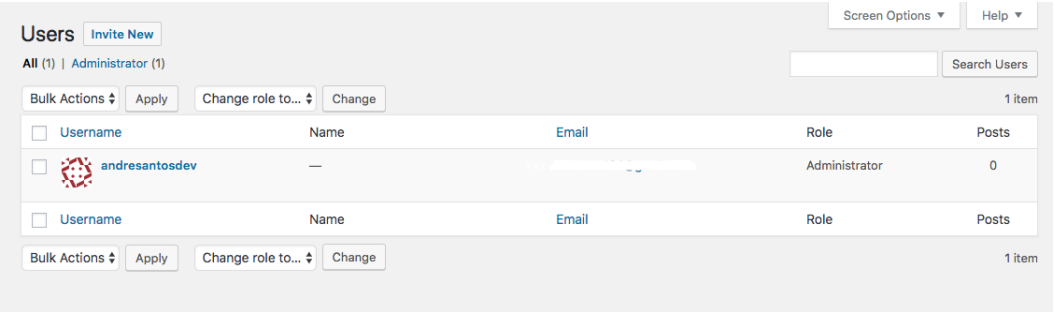


Figura 2.11: Interface de gestão de utilizadores Wordpress.

Aqui é possível adicionar, remover e editar utilizadores. A cada utilizador é possível atribuir os papéis de utilizador de 'Administrator', 'Contributor', 'Author' e 'Editor'. Para o contexto de utilização de blogue, estes papéis cobrem as permissões necessárias para o correto funcionamento da página. As permissões básicas encontram-se na figura abaixo nas 3 categorias pré-definidas, 'Editing Capabilities', 'Deletion Capabilities' e 'Reading'. No entanto, para aplicações e páginas mais complexas, é possível criar e gerir as permissões dos papéis de utilizadores. Para isso existe um plugin, o 'Capability Manager Enhanced' que é uma ferramenta que depois de ativa permite alterar as permissões associadas a cada papel de utilizador [63]. Como podemos constatar na figura, existem muitas outras permissões que podem ser criadas para que outros casos de uso possam ser salvaguardados. Além das permissões, também podemos criar assim novos papéis de utilizador e associar-lhe as permissões (figura 2.12).

Esta forma de gerir utilizadores e as respetivas permissões, assenta no modelo de controlo de acesso RBAC, onde cada utilizador atua segundo as permissões que o seu papel no sistema acarreta. Esta é uma opção que cobre muitos casos de uso onde há poucos papéis de utilizadores e bem definidos. Numa aplicação cujos os intervenientes possam ter papel A, B, C e D, bem definidos o controlo de acesso é fácil de configurar e de gerir. No entanto, se dentro do papel de utilizador A poderá haver diferenças entre eles, apesar de terem o mesmo papel no sistema, terão que ser criados mais papéis de utilizador para os poder separar o que num caso extremo, pode levar a criação de muitos papéis de utilizador, aproximando-se a gestão de permissões à gestão que o modelo DAC implementa.

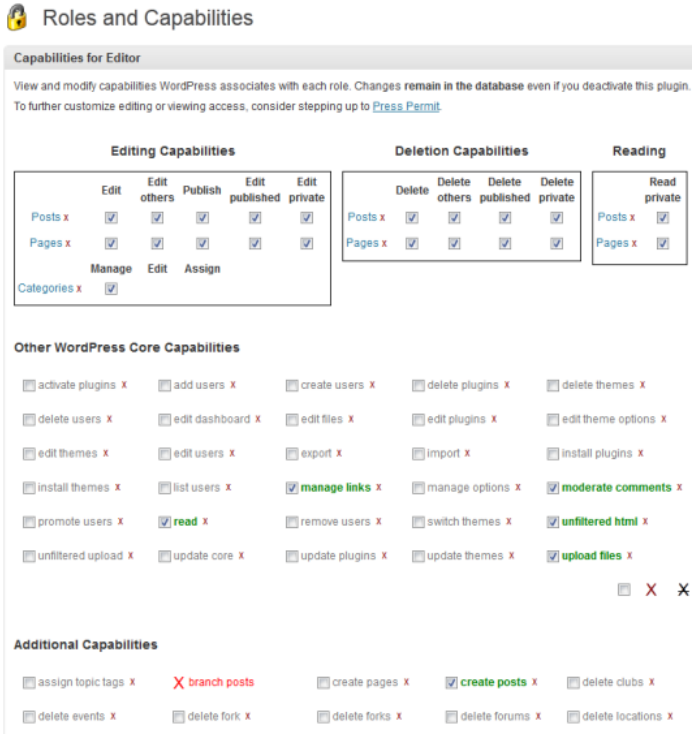


Figura 2.12: Interface de gestão de permissões do plugin Capability Manager Enhanced [63].

2.5.5 ARMAZENAMENTO SEGURO DE PALAVRAS PASSE

As palavras passe são uma informação essencial para a segurança das contas de utilizador do sistema e por isso é extremamente importante garantir o armazenamento e acesso seguro às mesmas. Inicialmente as distribuições Linux armazenavam a informação relativamente às palavras passe no ficheiro '/etc/passwd', ficheiro esse que era acessível a todos os utilizadores pois continha outras informações importantes que eram necessárias estarem acessíveis a todo o sistema e representava por isso uma falha de segurança [64]. Uma vez que o ficheiro '/etc/passwd' tinha que continuar acessível a todos os utilizadores para aceder à lista de utilizadores e respetivos diretórios pessoais e preferências da 'shell', as distribuições hoje instalam por defeito o pacote 'shadow-utils' que permite guardar de forma segura as palavras passe no ficheiro '/etc/shadow' que apenas está acessível ao utilizador 'root' [64]. Neste ficheiro podemos encontrar o conteúdo representado na figura 2.13, num sistema Linux Debian com o utilizador 'andre':

```
andre:$6$tbxM8snC$TWUcuWLl01QurYJoI3VvaOned8OFH3Y/5NqZMNDo1efcj6zJK2KVnIkWNu/T3aDaGuHjjf4pi86TCcCISMZ4l/:17792:0:99999:7:::
geoclue:*:17792:0:99999:7:::
andre@debian:~$
```

Figura 2.13: Exemplo de string armazenada para guardar palavras passe em sistemas operativos Linux.

Cada linha representa um utilizador e em cada uma há uma separação de campos por ':' totalizando 9 campos de informação. O primeiro campo representa o nome de utilizador e o segundo um conjunto da síntese da palavra passe, com a identificação da

função de síntese utilizada e o sal utilizado para gerar a síntese. Depois temos campos para especificar o número de dias de validade da palavra passe, neste caso 17792, depois o número de dias após os quais deve ser feita a alteração da palavra passe que neste caso é 0, o campo seguinte representa o número de dias antes da alteração que o utilizador deve ser avisado (99999), o campo seguinte representa o número de dias que demora até a conta ser desativada após expiração da palavra passe (7) e por fim o número de dias que a conta estará inativa (indefinido). O último campo não está definido pois ainda não é utilizado pelo 'shadow-utils' [64]. Analisando o campo mais relevante, que é o campo que contém a síntese da palavra passe, o sal e função de síntese de utilizada, temos neste caso o seguinte valor:

6tbxM8snC$TWUcuWLl01Q/ ... /jjf4pi86TCcCISMZ4l/

A primeira informação que aparece neste campo é a identificação da função de síntese utilizada, neste caso $6 o que corresponde à função SHA-512. Além do $6 esta porção pode tomar os valores $1 que significa MD5, $2 Blowfish, $2a eksblowfish e $5 SHA-256 [64]. A seguinte secção contém 'tbxM8snC' que corresponde ao sal utilizado pela função síntese e por fim, temos a palavra passe sintetizada pelo SHA-512 e o sal apresentado, que corresponde a '*TWUcuWLl01Q/ ... /jjf4pi86TCcCISMZ4l/*', 128 caracteres no total como podemos ver na 2.13. A síntese gerada é extremamente longa e um ataque de dicionário ou de força bruta torna-se ineficaz mesmo tendo acesso ao sal utilizado no caso da função de síntese utilizada for o SHA-256 ou SHA-512.

Esta é uma forma segura de armazenar e gerir palavras passe, e não é exclusiva ao Linux. A *Framework* Django permite configurar as suas palavras passe de forma a que sejam armazenadas de uma maneira semelhante [65]. Tipicamente, a palavra passe é armazenada segundo a estrutura da *string* seguinte:

<algorithm>$<iterations>$<salt>$<hash>

Por defeito, o algoritmo escolhido é o Password-Based Key Derivation Function 2 (PBKDFv2), utilizando a função de síntese SHA-256 como é recomendado pela NIST. Existe ainda a possibilidade de utilizar o BCrypt, Argon2 e PBKDFv2 com SHA-1. Com a existência destas funções de derivação de chave, é relativamente simples armazenar as palavras passe de forma segura, quer tenhamos uma aplicação web criada de raíz ou desenvolvida sobre alguma *Framework* como o Django.

2.5.6 CÓPIAS DE SEGURANÇA

As cópias de segurança são um mecanismo de defesa contra potenciais ataques ou falhas de sistema críticas, pois permitem restabelecer o estado do sistema aquando do ataque ou falha do sistema. Para executar as cópias de segurança existem vários métodos que podem diferir quer no tipo de cópia de segurança, quer na localização destino das mesmas. O processo de executar cópias de segurança está bastante consolidado atualmente, sendo o maior desafio a escolha da solução adequada para cada caso de uso.

Estratégias de cópias de segurança

Vamos analisar as 4 estratégias mais comuns para a execução de cópias de segurança, cópias de segurança totais, cópias de segurança incrementais, cópias de segurança diferenciais e cópias de segurança totais virtuais [66]. As cópias de segurança totais copiam para a localização de destino todos os ficheiros da pasta ou disco de armazenamento selecionado. Os ficheiros são comprimidos para poupar algum espaço, uma vez que todos os ficheiros sem exceção são copiados em todas as cópias de segurança, o que leva a uma grande ocupação de espaço num curto espaço temporal. Além do consumo elevado de espaço de armazenamento, poderão ser consumidos muitos recursos de rede e causar grande desgaste nos discos de destino devido ao tamanho de cada cópia de segurança [66]. Um disco com uma ocupação de 320 GB, ao fim de 10 cópias de segurança iria consumir na localização de destino das cópias de segurança 3200 GB. O facto de ocupar muito espaço e consumir recursos de forma significativa a nível de rede e/ou acessos ao disco de destino constituem as suas principais desvantagens. No entanto, a vantagem das cópias de segurança totais é a facilidade de restauração das mesmas. Basta saber qual o ficheiro e a data do qual se pretende restaurar e copiar o ficheiro para o destino. No caso de uma restauração total do sistema, o restauro é uma simples cópia para o disco a restaurar da cópia de segurança pretendida.

As cópias de segurança incrementais são uma solução que em relação às cópias de segurança totais poupam bastante espaço, resolvendo assim a sua principal desvantagem. Ao contrário do que acontece nas cópias de segurança totais, as incrementais ao copiar apenas os ficheiros novos ou alterados desde a última cópia de segurança total, que existe sempre pelo menos 1 vez [66]. Desta forma, cada nova cópia de segurança armazena apenas os novos ficheiros o que resulta em cópias de segurança que ocupam muito menos espaço pois é improvável que todos os ficheiros de um sistema de computação sejam alterados. Esta estratégia tem a desvantagem de requisitar de forma considerável recursos computacionais pois a cada cópia de segurança, o sistema tem que comparar cada ficheiro atual com o que foi armazenado na cópia de segurança inicial. Até o próprio processo de restauro total do sistema torna-se mais demorado pois é necessário

para cada ficheiro encontrar a cópia de segurança onde se encontra a versão mais recente, ou versão pretendida, do mesmo. Esta desvantagem é, no entanto, ultrapassada ao agendar as cópias de segurança para horas do dia ou alturas da semana em que os sistemas computacionais dispõem de mais recursos livres [66].

Em contraste com as cópias de segurança incrementais, existem as cópias de segurança diferenciais. Ambas as estratégias são muito semelhantes, mas ao invés de a cada cópia de segurança armazenar apenas os ficheiros novos ou alterados desde a última cópia de segurança parcial, armazena todos os ficheiros novos ou alterados desde a última cópia de segurança total [66]. Ou seja, um ficheiro alterado e guardado na cópia de segurança nº1 após a cópia total, estará também na cópia de segurança nº5. A vantagem em relação às cópias de segurança incrementais reflete-se no restauro da cópia de segurança, onde apenas é necessária a cópia de segurança inicial e a última cópia diferencial pois contém todas as alterações desde a cópia de segurança inicial. Isto torna o processo de restauro mais simples e como consequência, mais rápido. Na figura 2.14 podemos ver representados estas 3 estratégias de cópias de segurança representadas.

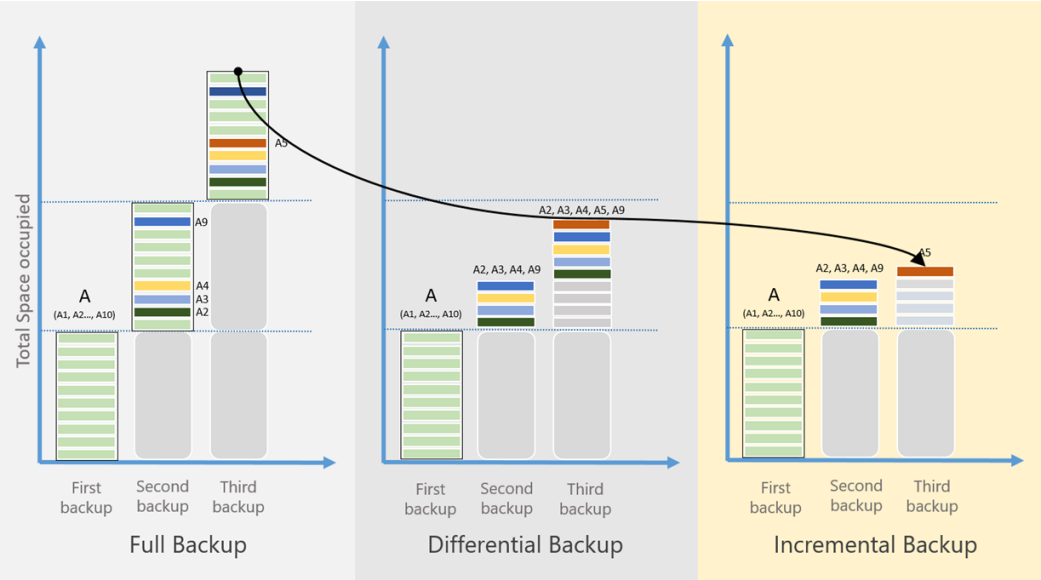


Figura 2.14: Comparação entre cópias de segurança totais, incrementais e diferenciais [67]

Por fim, temos as cópias de segurança totais virtuais. Neste caso a cópia total acontece apenas uma vez, sendo que nas seguintes cópias de segurança seguintes as alterações aos ficheiros são armazenadas e registadas numa base de dados. Cada uma das cópias é associada a uma cópia de segurança anterior de forma a manter uma linha temporal correta o que permite o restauro no futuro como se de uma cópia de segurança total se tratasse [68].

Estas são as principais estratégias de executar uma cópia de segurança, e para as

executar existem aplicações que fazem esse trabalho por nós. A seguir iremos analisar 2 soluções que executam cópias de segurança aplicando algumas destas estratégias de cópia de segurança.

Software para executar cópias de segurança

A aplicação da Apple, o Time Machine, faz cópias de segurança incrementais o que permite recuperar qualquer ponto das cópias de segurança que o utilizador desejar. É possível através do explorador de ficheiros recuperar o estado completo do computador a que pertence a cópia de segurança ou apenas recuperar ficheiros ou aplicações específicas. Por defeito, as cópias de segurança não estão cifradas e qualquer um pode aceder aos ficheiros desde que tenha acesso físico ao disco onde é feita a cópia de segurança. No entanto é possível aplicar a cifra das cópias, e até escolher como localização destino um disco de armazenamento ligado em rede. No entanto num ambiente profissional existem soluções que oferecem mais flexibilidade e configurações, e que podem ser executadas em outros ambientes além do MacOS, como por exemplo o Duplicati.

O Duplicati é, tal como o Time Machine, uma aplicação de cópias de segurança incrementais e cifradas por defeito. Em relação ao Time Machine, o Duplicati possui algumas vantagens. O facto de ser multiplataforma e gratuito, e de poder efetuar cópias de segurança para vários tipos de armazenamento, seja armazenamento externo, local, em rede, remoto ou para a *Cloud* (Google Drive por exemplo) tornam o Duplicati numa solução usável em ambientes profissionais. O Duplicati aplica essencialmente o conceito de cópias de segurança incrementais, após a cópia de segurança inicial apenas as partes alteradas dos ficheiros modificados são adicionadas à cópia de segurança. No entanto, na altura do restauro dos ficheiros a experiência de utilização assemelha-se à de cópias de segurança totais. Isto é possível pois o Duplicati gere as versões ao analisar não o ficheiro, mas sim os blocos que os constituem. Assim, sempre que um ficheiro é restaurado o software reconstrói o ficheiro ou ficheiros pretendidos utilizando apenas os blocos necessários [69].

Este exemplo representado na figura 2.15 apresenta uma possível configuração de cópias de segurança e é possível configurar periodicidade das cópias de segurança, origem e destino, e configurações de retenção de ficheiros permitindo a escolha da idade ou tamanho máximo de uma cópia de segurança. É também possível exportar uma configuração para um comando e criar comandos automatizados para as cópias de segurança e personalizar ainda mais como são executadas.

Figura 2.15: Interface de gestão de cópias de segurança do Duplicati.

2.6 CONCLUSÃO

As *Smart Cities* mostram ser um desafio quando se trata de segurança dos dados que recolhe e processa. É importante garantir a segurança do armazenamento dos dados e que o acesso aos mesmo seja feito de acordo com políticas bem definidas. Com a introdução do novo Regulamento Geral de Proteção de Dados (RGPD) é necessária atenção especial às questões de privacidade em qualquer sistema informático e o nosso caso de estudo, o projeto LIFE PAYT, surge numa altura em que esse regulamento entra em vigor o que permite desenhar um sistema de raíz com esse regulamento em consideração. Para ajudar a alcançar o desenvolvimento de sistemas mais seguros e que cumpram os respetivos regulamentos podemos também seguir boas práticas de segurança ou normas certificadas, que mesmo não sendo seguidas à risca, podem ser um bom ponto de partida para quem nunca teve realmente a preocupação de desenvolver um sistema que tem que cumprir não só requisitos técnicos, mas também legais.

As soluções apresentadas poderão ser úteis no desenvolvimento de uma solução para os requisitos que serão apresentados mais à frente nesta dissertação. Software como o Duplicati e a aplicação do controlo de acesso de *Frameworks* usadas a larga escala como o Django, ASP.NET e XACML poderão servir como um ponto de partida para desenhar uma solução que cumpra os objetivos deste trabalho. No caso do Django podemos retirar a utilização de decoradores para o controlo de acesso, um pouco como acontece também no ASP.NET. O armazenamento seguro de palavras passe poderá ser alcançado adotando a estratégia também aplicada na *Framework* Django caso a circunstância assim o permita. O XACML mostra como se consegue implementar uma estrutura de controlo de acesso completa onde há uma separação entre políticas, objetos

acedidos, entidades que acedem e mecanismo de decisão. Esta *framework* é no entanto bastante complexa e pode tornar o sistema lento, mas a ideia geral do XACML pode ser adaptada ou simplificada para o nosso caso de uso.

No capítulo seguinte será apresentada a solução, que irá cumprir uma série de requisitos a vários níveis, sejam estes técnicos ou funcionais. Analisando a arquitetura atual do sistema LIFE PAYT, poderemos depois definir os requisitos da solução que irá ser apresentada. Com base nos requisitos, será desenhada uma solução que poderá ter em consideração algumas das soluções aqui apresentadas.

capítulo 3

Life Payt - Solução

Este capítulo terá como objetivo propor uma solução para os requisitos de segurança em termos de ataques externos, desempenho de sistema, funcionalidades e questões legais. O caso de estudo que é o projeto LIFE PAYT, uma aplicação do conceito de cidade inteligente em Portugal (Aveiro, Lisboa e Condeixa), na Grécia (Vrilissia) e no Chipre (Larnaka).

Nas próximas secções será feita uma apresentação do projeto LIFE PAYT (secção 3.1), os requisitos da solução (secção 3.2 e qual o estado do sistema no que diz respeito ao controlo de acesso, à segurança em geral (secção 3.3)). A solução geral definida na secção 3.4 terá uma aplicação específica para o sistema LIFE PAYT e terá uma breve apresentação na secção 3.5 de como será implementada como estará explicado detalhadamente no capítulo Y sobre a implementação desta solução de controlo de acesso.

3.1 caso de estudo - projeto life payt

O projeto LIFE PAYT é um sistema de cidade inteligente que tem como principal objetivo ajudar na gestão de resíduos urbanos e reduzir a sua produção incentivando a reciclagem. O projeto está a ser implementado em países no Sul da Europa que não atingiram as metas definidas pela União Europeia no Programa de Ação Ambiental que promove uma hierarquia de resíduos, prevenção, reutilização e reciclagem. Como incentivo será aplicado nas áreas de intervenção a tarifa PAYT, que cobra periodicamente uma tarifa pelos resíduos produzidos de cada cliente, pagando apenas os resíduos que produz ao invés de uma taxa global ou variável que nem sempre corresponde à quantidade

de resíduos produzidos. Aveiro, Condeixa-a-Nova, Lisboa (Portugal), Vrilissa (Grécia) e Larnaka (Chipre) são as 5 áreas de intervenção definidas pela União Europeia. Cada município será responsável pela sua gestão PAYT podendo assim adaptar o sistema às necessidades específicas de cada município e demostrar que o sistema PAYT pode ser aplicado com sucesso em contextos diferentes.

O sistema LIFE PAYT tem uma componente de software e uma de hardware, com a última a ser responsabilidade de cada município e por isso este trabalho foca-se apenas na componente do software. O papel do software no projeto, independentemente das especificidades de cada município, é receber os dados relativamente à produção de resíduos associados aos clientes e apresenta-los aos várias partes envolvidas que irão interagir com o sistema. As funcionalidades podem, no entanto, variar muito de município para município pois dependem não só do contexto dos mesmos como também do hardware que envia a informação, e que informação, para o sistema. Haverá processamento de dados relativamente a consumos, faturação e dados pessoais e esse processamento terá que ser controlado e devidamente autorizado de forma a cumprir o novo Regulamento de Proteção de Dados da União Europeia.

A interação com o sistema LIFE PAYT será feita através do portal que está em desenvolvimento, em '`http://portal.life-payt.eu`', sendo esse o mesmo para todos os municípios que participam no projeto. O portal estará disponível para administradores de sistema, funcionários das câmaras e clientes das áreas de intervenção cuja tarifa mudou para PAYT, existindo também um espaço dedicado a dados abertos onde serão apresentados dados relativamente ao projeto e devidamente anonimizados.

3.1.1 UTILIZAÇÃO DO SISTEMA

As funcionalidades do portal para todos os municípios participantes dividem-se por 3 tipos de utilizadores: administradores de sistema, funcionários e utilizadores. Para cada um deles foi desenhado um portal à medida das necessidades e modular que permite atribuir novas funcionalidades no futuro. Na figura 3.1 podemos ver a página apresentada aos utilizadores após uma autenticação com sucesso. Mais sobre o desenho da *interface* do portal pode ser lido na dissertação de L. Pedro Martins de Almeida, "Plataforma de Software para Sistemas PAYT em Cidades Inteligentes"[70]. Cada utilizador recorre a um nome de utilizador para se autenticar, que no caso de Aveiro é o número do contrato, e uma palavra passe. No entanto, existe ainda uma chave mestra, uma chave que é atribuída aquando o registo do utilizador e serve apenas para validar a conta na primeira autenticação e para no futuro, caso necessário, redefinir

a palavra passe. Os administradores são os únicos tipos de utilizadores que não estão associados a qualquer município, por isso as funcionalidades para os funcionários e utilizadores poderão no futuro variar conforme as necessidades específicas de cada um. Atualmente a única área piloto com o sistema a funcionar é o município de Aveiro e por isso as funcionalidades mais específicas são ainda muito direcionadas para o contexto da aplicação do programa PAYT no município de Aveiro.

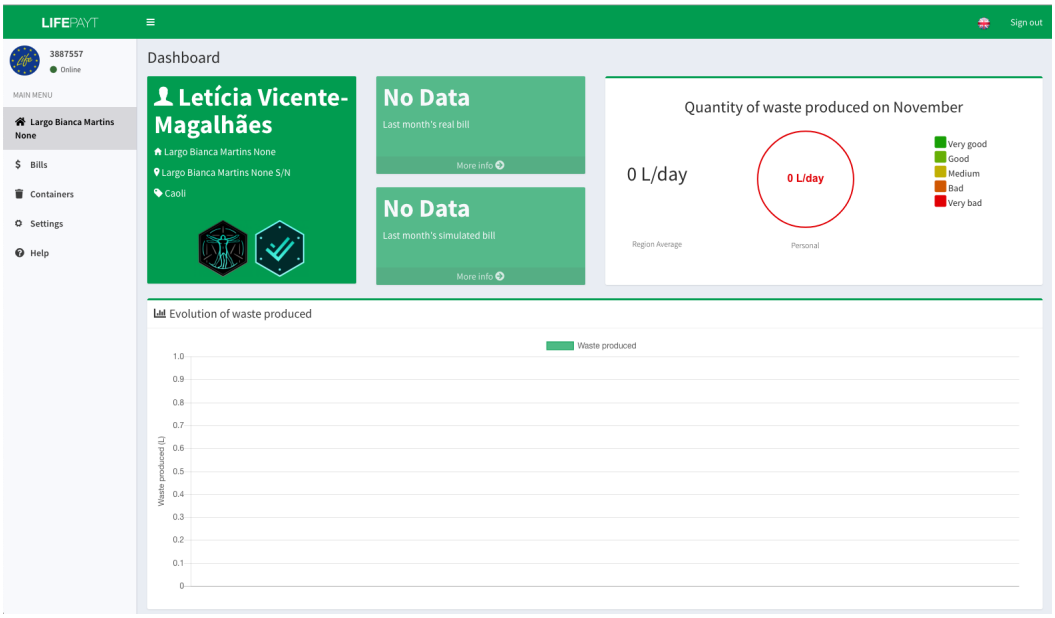


Figura 3.1: Página apresentada aos utilizadores autenticados. Todos os dados são fictícios.

Administradores

Os administradores têm o papel de gerir o sistema, com a gestão dos serviços e utilizadores de todos os municípios. A gestão de utilizadores consiste em adicionar e eliminar utilizadores, enquanto que a gestão de serviços passa por adicionar, remover ou parar serviços, como por exemplo as instâncias que servem cada município. Todos esses serviços têm que ter uma chave secreta e essa é introduzida pelos administradores na área reservada aos mesmos.

Funcionários

Estes tipos de utilizadores são funcionários dos municípios responsáveis pela gestão da plataforma no contexto em que estão inseridos. Na primeira área piloto com o portal em funcionamento, em Aveiro, os funcionários podem consultar estatísticas gerais de utilização dos contentores, consultar a data do último acesso ao portal por parte dos seus clientes, atribuir ou remover a associação de cartões aos clientes, consultar a utilização de contentores específicos e fazer o carregamento de ficheiros de faturação para associar valores de faturação aos clientes.

Utilizadores

Os utilizadores são os clientes dos municípios e podem através do portal consultar os seus gastos em termos de resíduos urbanos. No município de Aveiro os clientes podem ver o histórico de consumos e valores faturados, apresentando um comparativo entre a nova tarifa PAYT e a tarifa que é aplicada fora da área piloto. Além disso os utilizadores podem alterar as suas preferências relativamente à proteção de dados, como ser esquecidos e de ter a opção de receber e-mails informativos sobre o projeto. Na área pessoal podem também alterar o seu e-mail e alterar a palavra-passe, funcionalidades que são comuns a todos os tipos de utilizadores do sistema.

3.2 REQUISITOS DA SOLUÇÃO

O objetivo deste trabalho é proteger a informação e garantir o cumprimento do novo RGPD seguindo algumas das diretrizes da norma ISO 27001 [3]. No desenho inicial do sistema algumas indicações foram tidas em conta, como o cuidado de guardar e processar apenas informação estritamente necessária para o funcionamento do modelo de negócio do LIFE PAYT. Os papéis de utilizador permitem também criar uma separação de deveres e atribuição de diferentes responsabilidades a cada papel tal como previsto nos pontos de controlo subjacentes ao ponto A.6.1 do ISO 27001. No entanto poucos mecanismos de segurança existem ainda implementados, sendo que um deles bastante crítico para a proteção da informação, que é o mecanismo de controlo de acesso que tem falhas de segurança que podem comprometer o sistema. No entanto o controlo de acesso por si só não garante um sistema seguro e é por isso necessário definir os requisitos de segurança que o sistema deve cumprir. Este subcapítulo irá definir vários requisitos tais como requisitos de proteção contra ataques externos, desempenho e legalidade do sistema, e por fim os requisitos funcionais do controlo de acesso.

Proteção contra ataques

Sendo um sistema que irá armazenar informação pessoal e sensível a possibilidade de ser atacado irá sempre existir. Atualmente a quantidade de informação sensível ainda é reduzida, mas a obtenção do E-Mail, número de identificação fiscal, morada e interação de um cidadão com o sistema pode representar um perigo para o proprietário. Por exemplo, com o registo das vezes e a que horas um cidadão vai ao contentor do lixo pode-se a partir daí deduzir quando é que uma pessoa se encontra em casa ou não. Além disso, a plataforma está numa fase inicial e pronta para expandir e ter outras funcionalidades. Poderá no futuro ter que guardar informações de pagamento o

que representa um tipo de informação bastante sensível e ainda mais apetecível para potenciais atacantes. O ataque para o roubo deste tipo de informação é geralmente executado por pessoas com intenções de tirar algum partido de informações de terceiros, como por exemplo o assalto a uma propriedade ou a utilização de métodos de pagamento. Este é um modelo de um possível atacante e as técnicas podem ser simples como a descoberta fácil da palavra passe por tentativa e erro, ou outras mais complexas tais como Man In The Middle (MiTM), SQL Injection (SQLi), roubo de cópias de segurança ou ataque direto aos dados ultrapassando o controlo de acesso aos mesmos. Estas formas de atacar o sistema definem o primeiro modelo de atacante tem como principal objetivo roubar informação de terceiros e com isso obter outros tipos de benefícios através da mesma. É por isso importante implementar as medidas de proteção adequadas para estes ataques, evitar a interceção de informação e a sua legibilidade, filtrar todo o texto introduzido pelo utilizador e realizar e proteger as cópias de segurança e as bases de dados para recuperar o sistema em caso de um incidente fatal, seja causado por um ataque externo ou falha do próprio sistema. Deverá também ser implementado um mecanismo de controlo de acesso adequado de forma a que os utilizadores tenham acesso apenas à informação a que podem aceder. As cópias de segurança devem ser periódicas, cifradas e se possível devem ser armazenadas em uma ou várias unidades de armazenamento remotas para o caso de a falha de sistema ser fatal, e assim os dados não se perderem.

Temos ainda outro modelo de atacante, o atacante que pretende causar danos a uma organização ao tornar o serviço indisponível para utilização por parte dos utilizadores. Isto pode ser alcançado com recurso a um ataque DDoS ao serviço, esgotando os recursos das máquinas a disponibiliza-lo levando a que deixe de estar disponível para acesso dos seus utilizadores. É por isso importante ter esta forma de ataque em conta pois é fácil de executar e pode causar grandes danos quer a nível da imagem da entidade responsável pelo serviço como pelos seus lucros que possam ser diretamente dependentes do mesmo.

Desempenho do sistema

O controlo de acesso vai a cada leitura ou escrita verificar se o utilizador a fazer o acesso tem ou não permissão para aceder. Para isso o mecanismo de decisão vai avaliar um conjunto de regras a cada acesso, processo que certamente tornará o acesso mais demorado do que sem qualquer tipo verificação. Em termos de desempenho, o novo mecanismo de decisão não deverá comprometer a utilização por parte dos utilizadores, mesmo que simultaneamente. Espera-se que os testes de desempenho revelem uma resposta mais lenta, mas que não deverá comprometa a utilização normal do sistema.

Requisitos legais

O projeto LIFE PAYT entrou em funcionamento para o público em 2018, ano em que o novo regulamento de proteção de dados da União Europeia entrou em vigor. O novo regulamento estabelece novas obrigações no tratamento de dados, tornando se num assunto de grande importância. O regulamento estabelece que os proprietários da informação devem dar sempre o seu consentimento para a utilização dos seus dados e se possível até escolher que dados querem partilhar ou que sejam processados. É fundamental que todo o processamento seja legitimado e que o proprietário tenha conhecimento de todo o processamento que é feito e para que fins através da política de segurança da organização. A política de segurança definida pode ser encontrada no anexo A.

Controlo de acesso - Requisitos funcionais

Para a implementação do mecanismo de controlo de acesso vamos considerar algumas diretrizes do ISO 27001. A primeira diretriz a considerar é o ponto de controlo A.5.1.1. Esta estabelece que um conjunto de políticas de segurança devem ser definidas e transmitidas a todos os funcionários da organização. Na secção 3.4.4 teremos a política de privacidade definida para o projeto LIFE PAYT e que se irá refletir nas políticas de segurança e regras de controlo de acesso. Estas políticas são diretamente definidas por algumas diretrizes da certificação ISO 27001. Um exemplo disso é a diretriz proveniente do ponto de controlo A.6.1.2. Nas políticas terá que estar definido uma segregação de obrigações dos diferentes utilizadores, definindo o que cada um pode aceder. Os usos de caso descritos na secção anterior podem ser a definição dessa política de segurança por exemplo. Um aspeto fundamental da solução o controlo de acesso que será implementada é a classificação da informação segundo uma escala pré-definida com vários níveis de classificação, tal como proposto nos pontos de controlo A.8.2.1 e A.8.2.2 da certificação ISO 27001. Esta classificação é parte da base da solução para controlar o acesso à informação. As características relativamente à sensibilidade da informação farão parte do processo de decisão, tal como o papel que cada utilizador tem dentro do sistema.

O controlo de acesso poderia resumir-se apenas a estes pontos, mas há mais considerações importantes. No sistema LIFE PAYT irão participar vários municípios, sendo que os funcionários de cada um têm o mesmo papel dentro do sistema, o papel de funcionário. Um mecanismo de controlo de acesso que tivesse em conta apenas o papel do utilizador e a sensibilidade da informação que acede, significa que um funcionário de um município X poderia aceder a informações do município Y. Isto pode aplicar-se também aos utilizadores normais e este cenário não é de todo desejado, pois compromete a segurança geral do sistema e pode levar a conflitos entre municípios. Por isso

é necessário ter em conta um atributo do utilizador, o município a que pertence. O mecanismo de decisão irá, portanto, decidir com base não só no papel do utilizador, mas também nos atributos da informação que está a ser acedida.

No próximo subcapítulo será apresentada a arquitetura atual do sistema LIFE PAYT e os seus componentes, para depois analisar quais serão os possíveis impactos que os requisitos definidos neste subcapítulo poderão ter no sistema.

3.3 ARQUITETURA DO SISTEMA

Na sequência da apresentação dos requisitos da solução, este subcapítulo irá apresentar uma visão geral sobre a arquitetura do sistema LIFE PAYT. O sistema LIFE PAYT tem uma arquitetura de *multitenancy*, ou seja, a instância da aplicação LIFE PAYT é executada num ou mais servidores de forma isolada para um determinado grupo de utilizadores. No nosso caso, um *tenant* representa um município, e apenas utilizadores pertencentes a esse município irão aceder à sua instância, ou *tenant*. Isto permite que cada *tenant* possa ser adaptado a requisitos específicos de cada grupo de utilizadores. Nesta secção serão apresentados os módulos principais, o de autenticação e o *tenant* base, e as respetivas bases de dados, pois as alterações serão refletidas essencialmente nestes componentes do sistema.

3.3.1 ARQUITETURA DO SISTEMA LIFE PAYT

Na figura 3.2 temos uma vista geral sobre o sistema LIFE PAYT inicial e quais os seus componentes e respetivas bases de dados.

Todos estes componentes, com exceção às bases de dados, correm sobre a plataforma Docker tendo cada componente um contentor (*container*) dedicado, o que garante uma execução isolada de cada módulo, proporcionando o isolamento entre dados e implementações dos diferentes municípios. Destes componentes destacam-se o módulo de autenticação e respetiva base de dados, e os módulos *tenant*, que são o núcleo do sistema LIFE PAYT onde os dados são processados e armazenados respetivamene. Na figura apresentada podemos ver além dos principais componentes já mencionados, o servidor Hypertext Transfer Protocol (HTTP), o Comprehensive Knowledge Archive Network (CKAN) e o Elasticsearch + Logstash + Kibana (ELK). O servidor HTTP é o que serve a página web com o seu domínio, portal.life-payt.eu, a ser mantido e registado pelos serviços do Cloudflare [71]. O módulo denominado CKAN é o portal

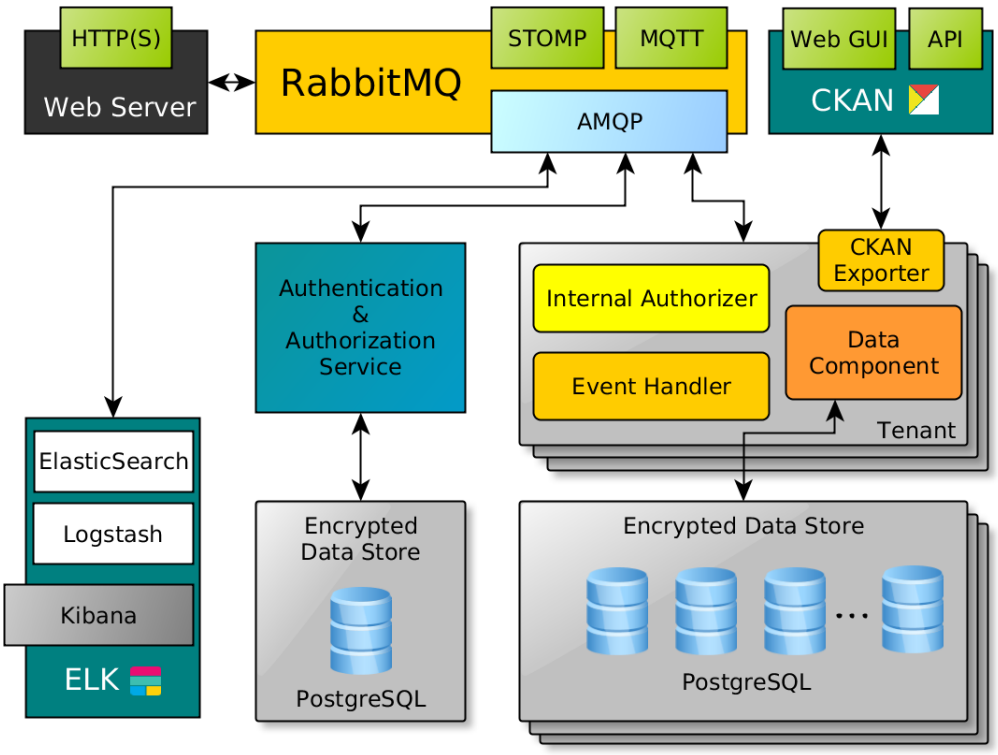


Figura 3.2: Arquitetura Geral do Sistema LIFE PAYT.

de dados abertos [72] do projeto, onde informação estatística sobre a implementação PAYT é apresentada, e o ELK é o sistema de monitorização que que analisa e apresenta os *logs* do sistema [73].

A comunicação entre componentes é executada através do RabbitMQ [74], com a gestão de mensagens a ser feita através da biblioteca Python 'asyncio' que permite envio e receção de mensagens de forma concorrente. Esta implementação do sistema de comunicação traz também outras funcionalidades, uma delas bastante importante para a segurança do sistema em geral, que é a identificação do remetente de cada mensagem. Uma vez que o acesso às funções de cada módulo, e consequentemente aos dados armazenados nas bases de dados, é feito através do envio e receção dessas mensagens, é muito importante identificar o remetente de cada uma. Com a identificação atribuída automaticamente, em função do utilizador estar autenticado ou não, é possível controlar o acesso, embora de forma simplista, aos dados. Caso seja um utilizador ou serviço reconhecido pelo módulo 'Auth', é-lhe atribuído um identificador único que identifica um utilizador ou serviço, sendo essa associação armazenada nas respetivas bases de dados.

Módulo de autenticação (*Auth*)

O módulo de autenticação é o módulo responsável pela autorização de acesso ao

sistema por parte dos utilizadores e por parte dos serviços (outros componentes), como por exemplo instâncias do *Tenant*, à rede interna. Além da autorização, é também responsável pela gestão de todos os utilizadores, das instâncias do módulo *Tenant* e da gestão dos papéis de utilizadores. Toda a informação processada por este módulo é gerida pelos próprios utilizadores se forem proprietários da mesma, ou pelos administradores do sistema. Este módulo é suportado pela base de dados 'Auth', que iremos analisar na próxima secção. Para a comunicação com a base de dados, existe um submódulo 'dbfuncs' que contém funções que executam comandos Standardized Query Language (SQL) e formatam as respostas para o formato adequado para apresentação ao utilizador. As funcionalidades inerentes a este módulo estão acima de tudo disponíveis aos utilizadores cujo papel é o administrador de sistema.

Módulos *Tenant*

Cada módulo destes corresponde a um município e as funcionalidades básicas que o módulo oferece são as mesmas para todos, desde gestão de clientes e faturas, consultas de consumos e estatísticas sobre os consumos, e gestão dos meios de identificação dos clientes (cartões no caso de Aveiro) nos contentores PAYT. Estas são as funcionalidades básicas que este módulo oferece, existindo a possibilidade de adicionar mais conforme o contexto em que é aplicado. Cada município tem assim a sua aplicação personalizada do sistema e evita a partilha de dados sensíveis e potencialmente danosos com outros municípios. Tal como no módulo de autenticação a comunicação é feita através do submódulo 'dbfuncs', e embora tenha o mesmo nome como no módulo de autenticação, nada tem a ver em termos de implementação. Além do submódulo 'dbfuncs', os *tenant* possuem também submódulos de exportação de dados para a plataforma de dados abertos, e outro para comunicação com a Application Programming Interface (API) dos fabricantes dos contentores e cartões de identificação para sincronizar dados de utilização com o sistema LIFE PAYT.

3.3.2 BASES DE DADOS

Existem duas bases de dados no sistema LIFE PAYT, a base de dados Auth e a base de dados *Tenant*. A base de dados *Tenant* é replicada pelo número de municípios com o sistema implementado a funcionar e podem ser diferentes entre si, dependendo das características de cada município. Na descrição que se segue para a base de dados *Tenant* estará representado o esquema base que é apresentado a todos os municípios e que cobre por exemplo as necessidades do município de Aveiro. As bases de dados estão instaladas através do PostgreSQL em servidores dedicados que comunicam com o

servidor principal através da rede interna. Para a proteção das bases de dados, os discos do servidor foram formatados com o sistema de ficheiros Zettabyte File System (ZFS), um sistema que garante a integridade dos dados ao aplicar a cada bloco do disco um *checksum* que verifica a cada leitura e escrita os dados de cada bloco [75]. Além da integridade dos dados, o sistema de ficheiros cifra todo o conteúdo nele armazenado sendo isso mais uma camada de segurança para os dados nele armazenados [76]. Além da segurança oferecida pelo próprio sistema de ficheiros, as permissões de acesso às bases de dados são atribuídas apenas aos módulos a que pertencem, ou seja, o módulo pertencente ao município X não consegue ter acesso, mesmo utilizando os seus métodos de acesso, à base de dados do município Y. Este facto cria a primeira camada de segurança no acesso às bases de dados. De seguida serão analisadas as duas principais bases de dados, primeiro a base de dados de autenticação e de seguida a base de dados do *Tenant*.

Base de dados *Auth*

A base de dados do módulo de autenticação, descrita na figura 3.3, guarda todos os utilizadores na tabela '`users`' guardando o e-mail, palavra-passe, papel de utilizador, nome de utilizador e um identificador único no campo '`user_id`'.

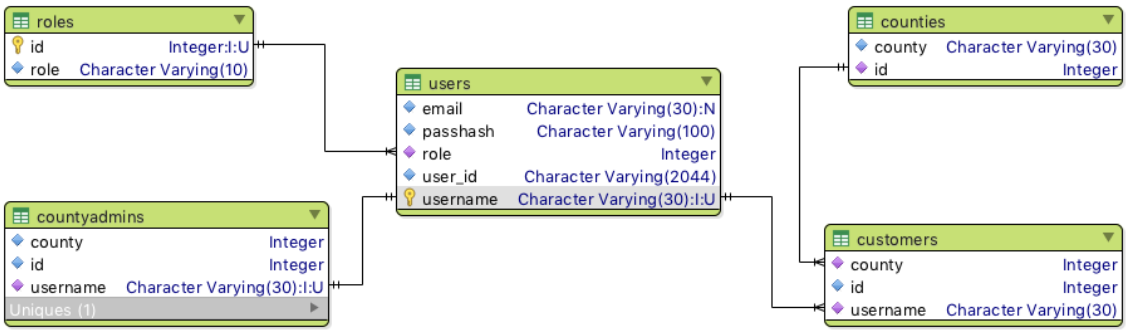


Figura 3.3: Diagrama da base de dados que suporta o módulo de autenticação.

A cada utilizador é associado um papel no sistema que pode ser administrador, utilizador ou funcionário das câmaras municipais, sendo estes armazenados na tabela 'roles' e só os papéis inseridos na tabela são possíveis atribuir. Existem ainda duas tabelas que servem de suporte para a identificação mais completa de utilizadores e funcionários das câmaras municipais, as tabelas '`customers`' e '`countyadmins`'. Além da listagem de cada tipo de utilizador, associam também cada um ao município a que pertencem que tal como no caso dos papéis de utilizador, se encontram listados numa tabela dedicada, a tabela '`counties`'.

Base de dados *Tenant*

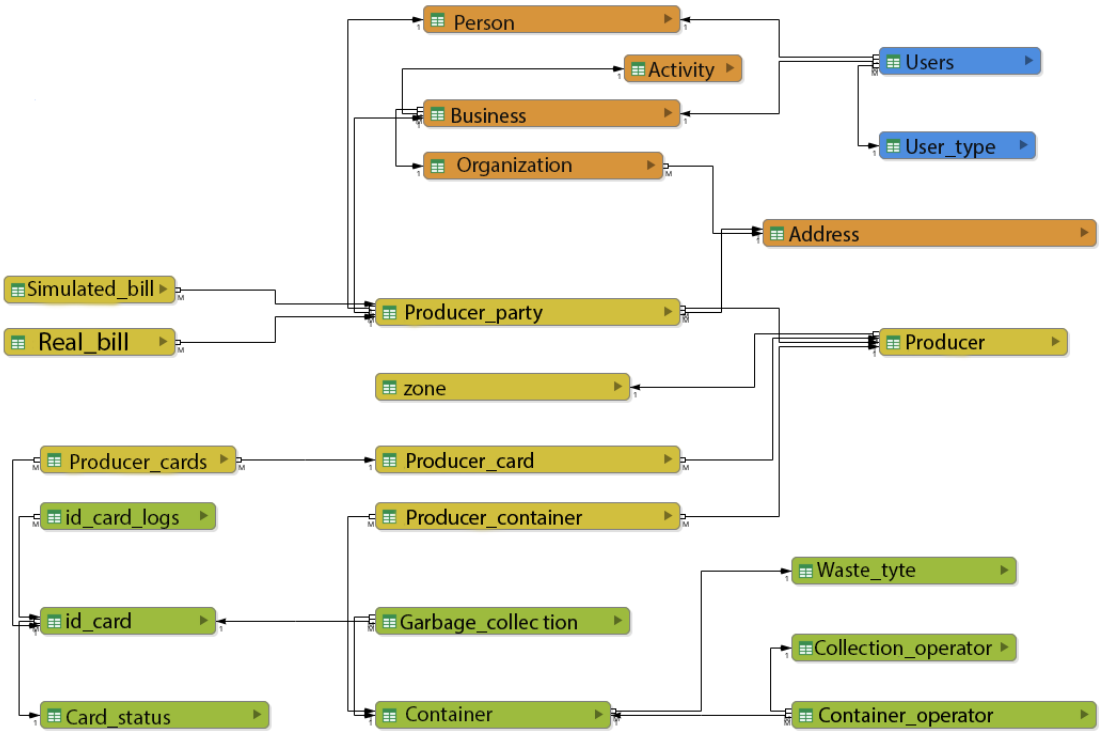


Figura 3.4: Diagrama da base de dados que suporta o módulo Tenant.

A figura 3.4 representa o diagrama geral da base de dados que serve o módulo *Tenant*. O identificador único de cada utilizador proveniente do módulo Auth é armazenado na tabela '`users`' (figura 3.5) de forma a que o sistema consiga ligar as contas de utilizador aos seus dados de utilização do LIFE PAYT. As contas de utilizador podem ser do tipo '`personal`' ou '`business`', com a primeira a ser uma conta de um utilizador doméstico e a segunda uma conta de um utilizador não doméstico como por exemplo uma loja ou empresa.

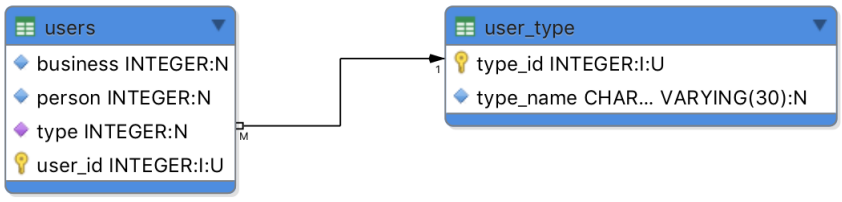


Figura 3.5: Tabelas que armazenam informação relativamente aos utilizadores da base de dados que suporta o módulo Tenant.

Associados a esses tipos de utilizadores está a informação pessoal de cada um, número de identificação fiscal, e caso seja um utilizador não doméstico o tipo de negócio

e qual o contrato de fornecimento de água com a câmara municipal e a descrição da organização, definida pelo número de identificação fiscal e a morada (figura 3.6).

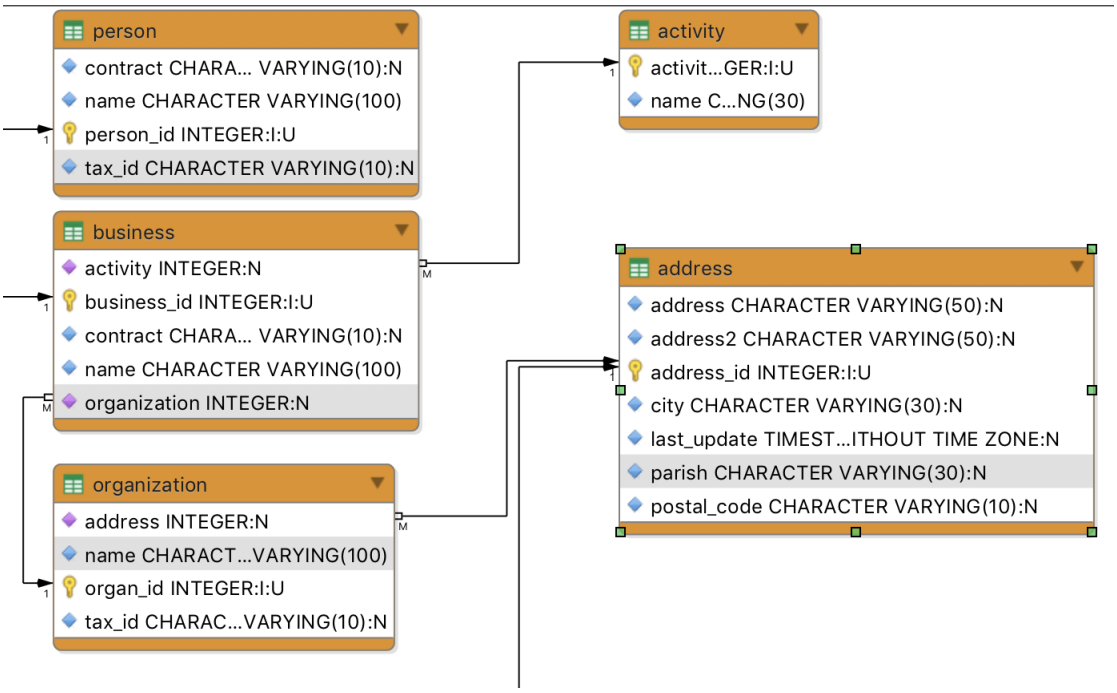


Figura 3.6: Tabelas que armazenam informação pessoal das pessoas e organizações da base de dados que suporta o módulo Tenant.

Este conjunto de informação é associado a um agregado de produtores (’`producer_party`’), que é caracterizado pela morada, utilizadores associados ao contrato (domésticos ou não domésticos), datas de início e fim de produção. Este agregado é identificado por um identificador na tabela ’`producer`’, sendo cada contrato um produtor mesmo que existam várias pessoas associadas ao contrato, ou seja, vários produtores de resíduos. Associado ao produtor, que identifica uma habitação ou negócio, está o identificador do cartão de acesso aos contentores. Esta informação é a que permite associar um consumo a um produtor e gerar faturas reais para o mesmo. Esta identificação serve tanto para utilizadores domésticos e não domésticos, enquanto que o aluguer de um contentor é apenas para utilizadores não domésticos, aí o consumo é associado ao contentor apenas. A tabela ’`producer_container`’ existe para registar esses contentores alugados (figura 3.7).

Os consumos são guardados na tabela ’`garbage_collection`’ onde é feita uma associação entre contentor e cartões de utilizador, sendo que este último é o que permite chegar ao cliente que teve o consumo associado (figura 3.8). Neste caso o consumo é uma abertura do contentor, pois no caso de Aveiro a tarifa é por aberturas dos contentores. No entanto a base de dados está preparada para outros tipos de tarifas, como por exemplo uma tarifa baseada no peso.

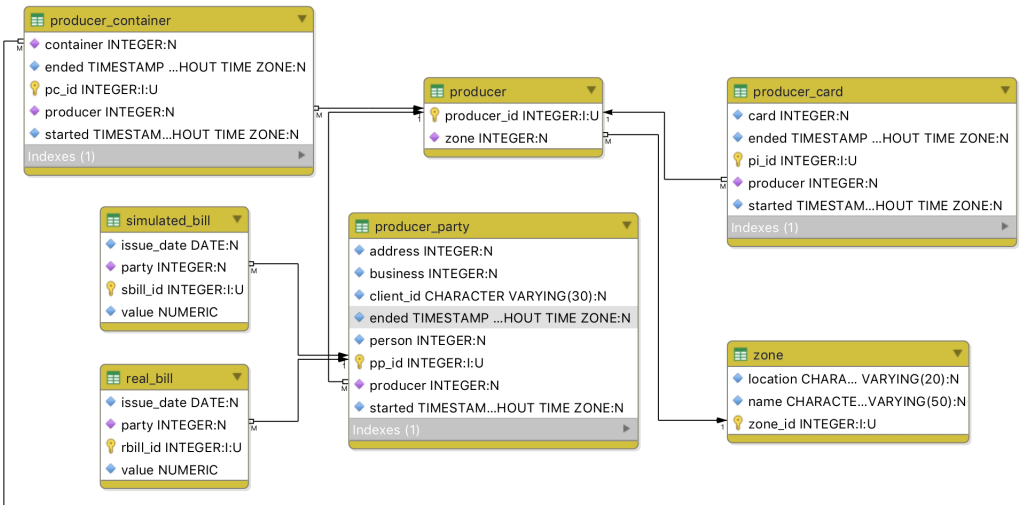


Figura 3.7: Tabelas que armazenam informação sobre os produtores e identificação dos mesmos da base de dados que suporta o módulo Tenant.

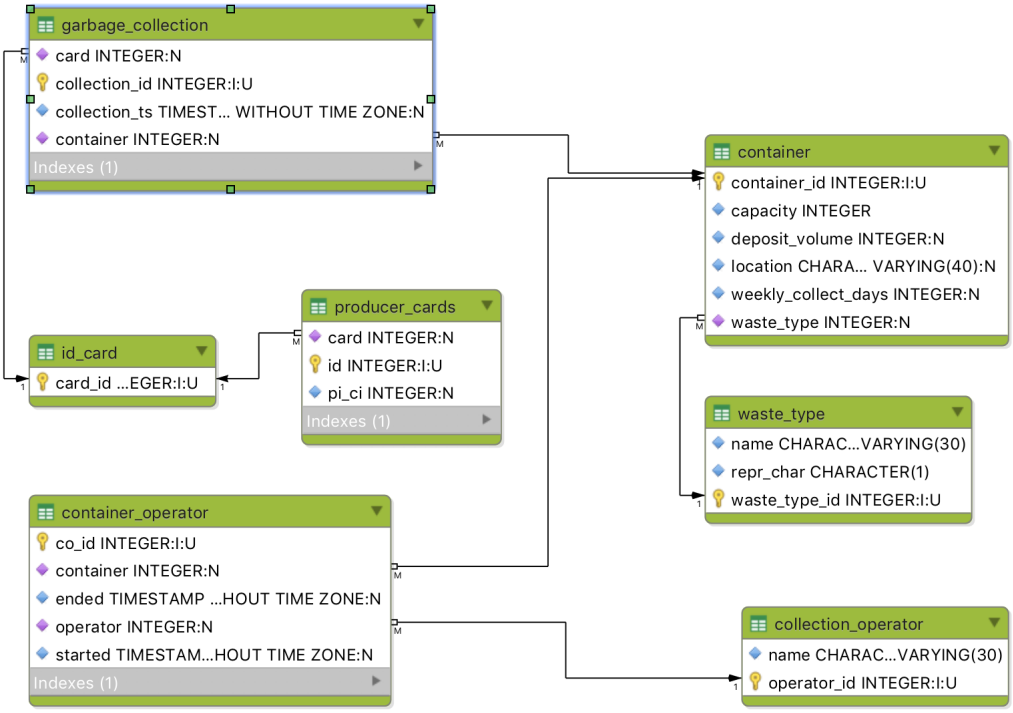


Figura 3.8: Tabelas que armazenam informação sobre consumos, cartões de acesso e contentores da base de dados que suporta o módulo Tenant.

Estas são as descrições das bases de dados que se encontram atualmente implementadas no sistema. Durante o desenho e implementação da nossa solução é importante ter esta estrutura de dados em conta, pois qualquer alteração indevida poderá comprometer o funcionamento do portal LIFE PAYT. A base de dados que suporta o módulo de autenticação irá estar mais sujeito a alterações, pois é lá que a grande parte das questões de segurança poderão ser resolvidas.

3.4 PROPOSTA DE SOLUÇÃO

Neste subcapítulo será apresentada a solução que irá ter em conta a arquitetura do sistema e requisitos expostos nos subcapítulos anteriores. Alguns requisitos já são cumpridos parcial ou totalmente e nesses casos será apresentado o que foi feito ou o que terá que ser feito para os cumprir. Para satisfazer todos os requisitos apresentados serão apresentadas soluções para a proteção contra ataques, gestão de cópias de segurança e controlo de acesso.

3.4.1 PROTEÇÃO CONTRA ATAQUES

Os ataques para os quais serão desenhadas soluções à medida da importância do projeto são a descoberta da palavra-passe por tentativa e erro, ataques MiTM, SQLi e ataques diretos ao servidor com o roubo de cópias de segurança e acesso direto às bases de dados.

Bloqueio de utilizadores

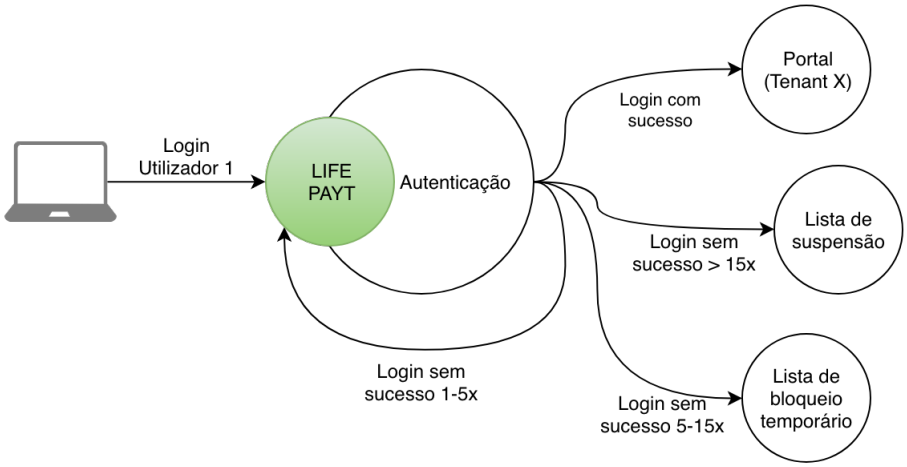


Figura 3.9: Diagrama de interação do mecanismo de bloqueio de utilizadores.

A descoberta da palavra-passe pode de uma só vez desbloquear o acesso a todos os dados relativos a um utilizador e deixá-lo sem acesso à sua conta caso a palavra-passe seja alterada. Para evitar a descoberta fácil a solução passa por exigir palavras-passe com um tamanho mínimo definido e alguma complexidade associada, como por exemplo a obrigação de utilizar pelo menos um dígito além dos caracteres alfabéticos. Esta prática é aliás uma das recomendações que o ISO 27001 apresenta no ponto de controlo A.9.4.3. As palavras-passe dos utilizadores LIFE PAYT devem ter pelo menos 8 caracteres, sendo no mínimo um deles um dígito. Além das regras para a definição da palavra-passe, é importante evitar que um mesmo endereço IP ou utilizador possa continuamente tentar

fazer logins sem sucesso. Por isso é necessário manter um registo de tentativas de login falhadas e com base nisso bloquear temporariamente o login a determinado utilizador. Se após 10 tentativas falhadas houver uma 11ª, o sistema bloqueia o acesso durante 5 minutos. Após esses 5 minutos se voltar a errar a palavra-passe, 12ª tentativa, volta-se a bloquear o acesso por 15 minutos, 30 minutos à 13ª tentativa, e se mesmo depois houver uma outra tentativa falhada a conta fica suspensa até que a palavra-passe seja reposta. A figura 3.9 apresenta o diagrama de interação deste mecanismo de bloqueio de utilizadores.

Armazenamento da palavra passe

Outros ataques que visam o roubo da palavra-passe podem acontecer seja por ataques MiTM, SQLi, acesso direto às bases de dados ou cópias de segurança. Para diminuir ou até mesmo eliminar o risco de que o roubo da palavra-passe é fulcral que a mesma nunca seja armazenada nem transmitida sem qualquer tipo de transformação, ou seja, deve ser cifrada. As palavras-passe nunca devem ser enviadas ser estarem cifradas, e a cifra armazenada deverá ter uma complexidade suficientemente alta de forma a que a descoberta do texto original que deu resultado ao texto cifrado seja computacionalmente inviável. O canal de comunicação é HTTPS, portanto a informação durante o transporte é salvaguardada. No entanto de forma a que se crie mais uma camada de segurança durante o transporte que será também útil para o armazenamento da palavra-passe, o lado do cliente produz uma síntese do texto introduzido recorrendo à função de síntese SHA-256. Este processo apenas acontece no caso da palavra-passe, uma vez que não precisa de ser decifrada no servidor, pois é utilizada a síntese como comparação. Para proteger as palavras-passe cifradas no armazenamento será utilizada a função de derivação de chaves PBKDFv2 [77] que recorre a uma função de síntese, um sal relativamente longo e um número de iterações pré-definido. Neste caso será utilizada a função de síntese SHA-512, um sal único de 64 bits gerado aleatoriamente e 10.000 iterações. Segundo a Cryptosense [78], uma palavra-passe de 8 caracteres aleatórios, um sal único de 64 bits e 10.000 iterações, demoraria 3 anos e 5 meses a ser descoberta com recurso a uma placa gráfica a efetuar 5 milhões de tentativas de adivinha. Este número de iterações pode a qualquer momento ser alterado caso assim se justifique para um número superior uma vez que o poder de processamento disponível é bastante grande. No nosso caso a palavra-passe quando chega ao servidor são 64 caracteres gerados a partir de uma palavra-passe original, portanto pode-se supor que o tempo que um possível atacante demoraria a descobrir a palavra-passe é grande o suficiente para não justificar sequer a tentativa de ataque. A figura 3.10 representa o fluxo de informação quando uma palavra passe é armazenada no servidor ou utilizada para autenticação.

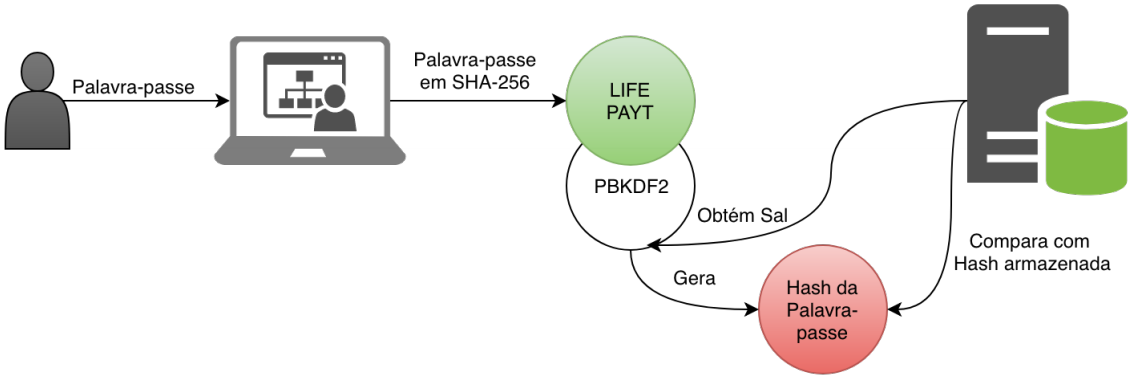


Figura 3.10: Diagrama de interação do armazenamento de palavras passe.

Além da palavra passe, é também armazenada a chave mestra. Esta chave é inalterável e serve para ativar a conta na primeira utilização da conta de utilizador e para recuperar a palavra passe no caso de esta ser esquecida. A chave é constituída por 16 caracteres gerados aleatóriamente aquando a criação da conta de utilizador. Uma vez que é necessário os administradores de sistema terem acesso a esta chave para auxiliar utilizadores que percam esta informação, a chave é armazenada em texto na base de dados.

Canais de comunicação seguros

Como já referido os canais de comunicação utilizam HTTPS, cifrando com a chave pública do destinatário todo o conteúdo das mensagens através de Transport Layer Security (TLS), tornando ataques MiTM ineficazes pois a informação é indecifrável. Adicionar outra camada de proteção como por exemplo cifrar o conteúdo com um algoritmo assimétrico como o RSA, seria de gestão difícil pois implicaria a gestão e troca de chaves, o que por si só pode ser um problema quando há milhares de utilizadores. Há casos em que se justifique como aplicações bancárias ou outras do género, mas não é este o caso. Uma outra camada de proteção a aplicar que não protege diretamente a informação é a anonimização dos dados. Os dados em vez de estarem associados a um nome de utilizador ou e-mail, são associados a um identificador único, no nosso caso um número inteiro. Assim mesmo que uma mensagem ou até um ficheiro de monitorização sejam intercetados e decifrados, a informação não é associada a nenhuma pessoa diretamente. Esta é, entre outras que vamos analisar de seguida, uma boa prática de segurança que ajuda também a dissuadir um atacante que consiga obter acesso ao servidor e à base de dados, tornando mais difícil a associação entre tabelas de informação, especialmente se os dados estiverem dispersos em bases de dados diferentes como é o nosso caso.

Boas práticas de segurança

Outra boa prática de segurança é fazer a sanitização ao conteúdo de texto enviado do navegador para o servidor. Os ataques SQLi podem constituir uma ameaça séria e se não houver tratamento do texto recebido o comando SQL pode executar um outro comando e devolver informação de forma não autorizada para o atacante. De forma a evitar estes tipos de ataques, os campos devem aceitar apenas o tipo de informação que pedem. No portal LIFE PAYT existem poucos campos de introdução de texto, no login é o nome de utilizador e palavra-passe, e depois de fazer o login existem campos de introdução de e-mail, nome, morada, palavra-passe e número de cartão de acesso. O campo da palavra-passe não necessita de qualquer controlo, uma vez que é aplicado a função de síntese SHA-256 a todo o texto introduzido aqui, por isso qualquer ataque SQLi não chegaria ao servidor como um comando legível. Os campos de introdução de e-mail e número de cartão de acesso apenas aceitam no lado do cliente os formatos corretos, ou seja, formato de e-mail e um número inteiro respetivamente. Os restantes campos são sempre sanitizados no lado do servidor. Todos os acessos à base de dados serão executados através da biblioteca '`aiopg`' da *Framework* '`asyncio`' que é baseado no *driver* de acesso a bases de dados PostgreSQL 'Psycopg' [79]. Este *driver* implementa por defeito a sanitização dos argumentos recebidos para a execução do comando de acesso à base de dados, evitando assim potenciais ataques SQLi [80].

Cópias de segurança

Por fim, um mecanismo de defesa que servirá não só para ataques externos como também para falhas de sistema ou incidentes ambientais que comprometam o sistema, são as cópias de segurança. O próprio ISO 27001 estabelece as cópias de segurança como um requisito para obter a certificação (A.12.3.1 do anexo 6). Existem ataques que podem corromper os dados do sistema e tendo cópias de segurança é possível restabelecer o estado anterior das bases de dados, e existem outros que podem procurar atacar as próprias cópias de segurança. Para colmatar isso, não só o acesso às bases de dados tem que ser protegido, também as próprias cópias de segurança o devem ser. Essa é uma funcionalidade que aplicações como o Duplicati, apresentado no capítulo 2, possuem e será algo obrigatório para proteger contra acessos não autorizados. No caso do Duplicati a proteção que oferece é a cifra de todo o conteúdo com o algoritmo AES-256 que só poderá ser decifrado com uma palavra passe definida aquando das configurações das cópias de segurança. Além da proteção das cópias de segurança as mesmas devem ser incrementais, o que permite poupar espaço de armazenamento, e ter uma periodicidade adequada. No caso do LIFE PAYT a periodicidade deverá ser apenas 1 vez por dia pois os consumos, no caso de Aveiro para já, são sincronizados apenas

1 vez por dia, portanto as alterações à informação mais crítica acontece exatamente 1 vez por dia. Um outro aspeto importante das cópias de segurança que advém do novo RGPD, é durante quanto tempo que o sistema LIFE PAYT mantém as cópias de segurança. Os utilizadores deverão ser informados durante quanto tempo após saírem do programa os seus dados continuarão a ser armazenados e possivelmente processados e neste caso, esse período não deverá ultrapassar os 3 ou 4 meses. Resumindo, as cópias de segurança são iterativas e contínuas, acontecem diariamente, são cifradas e não se devem manter por mais do que 3 a 4 meses. As cópias de segurança idealmente deverão ser alojadas num armazenamento diferente do sistema principal, salvaguardando os dados em casos de falhas críticas do servidor ou algum incidente ambiental, como um incêndio ou inundação.

Estas são as soluções que visam cumprir os requisitos definidos pela proteção contra ataques externos, com algumas delas a proteger também contra falhas de sistema em geral e incidentes ambientais. Na secção seguinte será apresentada a solução para o mecanismo de controlo de acesso adaptado às necessidades do sistema LIFE PAYT.

3.4.2 CONTROLO DE ACESSO

Tendo em conta a recomendação A.9.1.1 da certificação ISO 27001, vamos criar políticas de controlo de acesso que visam proteger cada objeto de forma individual de acordo com a sua criticalidade. A nossa solução de controlo de acesso requer que seja classificada toda a informação nas bases de dados através de um esquema de rotulagem, em que cada rótulo representa uma política de controlo de acesso. Assim, conseguimos executar o controlo de acesso à informação com base não só no papel do utilizador, mas também na classificação que irá definir o nível de acesso ao objeto. A criação destes rótulos e consecutiva atribuição aos objetos das bases de dados, cumpre diretamente dois dos pontos de controlo A.8.2 do ISO 27001 (ver anexo 6). Deste fazem parte a classificação da informação (A.8.2.1), criação de um esquema de rotulagem (A.8.2.2) e gestão de ativos (A.8.2.3), dos quais os dois primeiros são aqui aplicados diretamente. Esta solução pode ser vista como um modelo de controlo de acesso personalizado que utiliza o ABAC e RBAC para fazer um controlo de acesso ao estilo de MAC, que aplica as condições de acesso conforme o rótulo atribuído ao objeto. Com esta solução é possível aos administradores de sistema alterar a política de segurança associada aos objetos a qualquer momento. Na figura 3.11 está ilustrada uma arquitetura genérica de como se processa este controlo de acesso.

Recordando o fluxo de informação que a arquitetura da *Framework* XACML im-

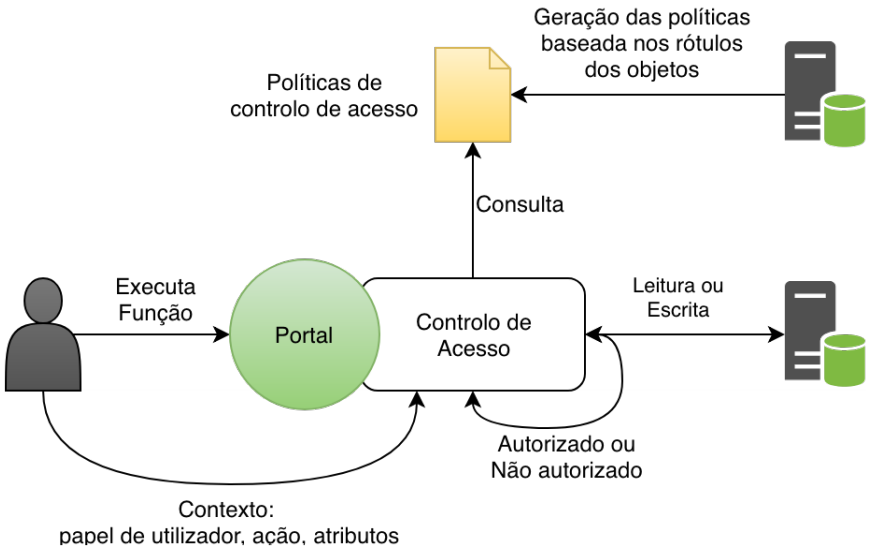


Figura 3.11: Diagrama de interação do mecanismo de controlo de acesso.

plementa, esta foi a melhor forma encontrada para implementar o controlo de acesso com base no contexto de utilizador e na sensibilidade dos objetos acedidos. Tal como na arquitetura do XACML, deverá existir um pedido por parte dos utilizadores, esse pedido enviado do portal para o controlo de acesso e avaliado conforme atributos e outros informações presentes em base de dados ou ficheiros com a informação necessária à decisão de autorizar ou não o acesso.

O mecanismo de controlo de acesso, terá em consideração para a decisão o contexto do utilizador que inclui o papel, ação e um atributo que é o município a que pertence. Atribuindo um identificador a cada objeto da base de dados relativamente à sua sensibilidade, e identificando o que cada função acedida retorna ou escreve, é possível a partir dos objetos acedidos atribuir uma política de controlo de acesso associada ao identificador. Com esta informação, é possível criar uma ACL, que irá conter a função e política associada. O mecanismo de controlo de acesso irá consultar essa ACL (objeto amarelo na figura anterior) para autorizar ou não um utilizador tendo em conta o seu contexto. Ou seja, se uma função Y acede aos objetos O1 e O2 cujos rótulos são R1 e R2 respetivamente, em que R2 é o rótulo no esquema de classificação mais alto (mais sensível), então a função Y deverá estar sujeita à política associada a R2. Com base na descrição do modelo de controlo de acesso de Kerr et al. [31], chegamos a esta fórmula simplificada e adaptada ao que aqui foi descrito o que permite expressar as políticas ou regras associadas às funções da seguinte forma:

$$P = \{subject,\ operation\}\ ou\ P = \{(role,\ A1,\ \ldots,\ An),\ (r,\ w,\ r\&w)\}$$

Em que o *subject* é definido pelo papel no sistema e um ou vários atributos-chave, dependendo do caso de uso e a *operation* é a operação feita sobre os objetos, escrever, ler ou ambos. Cada uma destas políticas corresponde a um nível de classificação que é atribuída aos objetos que são acedidos pelos utilizadores que por sua vez vão estabelecer a política que cada função deve cumprir.

A classificação deve ser feita de acordo com um esquema de rotulagem definido de acordo com a criticidade da informação que a organização possui armazena e processa. Para a organização LIFE PAYT o esquema de rotulagem para classificar a informação terá 4 níveis diferentes, cada com uma política de acesso associada. Os rótulos, ou níveis de classificação, podem tomar valores como: público, restrito, uso interno e proprietário.

Público

Informações rotuladas como públicas são objetos de informação não críticos, como os consumos de lixo de uma determinada área ou município, o número de utilizadores no sistema ou outros dados que se tornados públicos não causam qualquer dano à organização e aos seus utilizadores.

Restrito

Os objetos considerados restritos são informações que só podem ser acedidas pelo proprietário ou pela administração do município. O endereço de um utilizador, o seu nome ou e-mail devem ser rotulados como restritos, pois embora sejam informações pessoais, são necessárias para o funcionamento do modelo de negócio do LIFE PAYT e por isso devem estar acessíveis à administração dos municípios. Uma outra regra que estabelece é que os funcionários para poder aceder devem pertencer ao mesmo município do proprietário do objeto, evitando que um funcionário do município X consiga roubar dados de utilizadores pertencentes a um município Y. Em termos de leitura e escrita, os proprietários têm autorização para ler e escrever enquanto que os funcionários dos municípios apenas podem ler.

Uso interno

Este rótulo considera os objetos usados internamente para a operação do sistema. Objetos que encaixam neste rótulo são os identificadores únicos de utilizadores, contentores, os papéis de utilizadores ou a listagem de serviços com acesso ao sistema. Todos os objetos com este rótulo apenas podem ser acedidos por parte dos administradores ou pelo próprio sistema, com permissão para leitura e escrita em ambos os casos.

Proprietário

Objetos rotulados como confidenciais são informações que só podem ser acedidos pelo proprietário. No contexto da organização LIFE PAYT são poucos os objetos que podem ser acedidos exclusivamente pelo proprietário, nomeadamente o sal utilizado para proteger a palavra-passe e a palavra-passe encriptada. Isto são objetos que apenas o proprietário acede para efeitos de login e mais nenhum utilizador está autorizado a fazê-lo. Como os objetos apenas estão acessíveis ao proprietário, estes podem tanto ler como escrever sobre os mesmos.

Para sumarizar estas políticas de controlo de acesso baseado na rotulagem dos objetos temos a seguir a tabela 1. O contexto é definido pelo papel de utilizador, a ação, e se a informação é proprietária ou de terceiros, utilizando a seguinte denotação genérica: (*role*, *act*, *prop*), com 'role' a tomar os valores de 'user', 'county', 'admin' ou 'public', 'act' pode ser 'w' ou 'r', escrever e ler respetivamente, e 'prop' a poder tomar os valores 'own' ou 'other'. Para um utilizador com o papel 'user' que pretende ler um objeto que pertence a outro utilizador a denotação é: (*user*, *r*, *other*).

Contexto	**Público**	**Restrito**	**Uso Interno**	**Proprietário**
(user/county, r/w, own)	Sim	Sim	Não	Sim
(user, r, other)	Sim	Não	Não	Não
(user/county, w, other)	Não	Não	Não	Não
(county, r, other)	Sim	Sim	Não	Não
(admin, r/w, own)	Sim	Sim	Sim	Sim
(admin, r, other)	Sim	Sim	Sim	Não
(admin, w, other)	Não	Não	Sim	Não

Tabela 3.1: Tabela de comparação das várias funções de síntese.

Para implementar estas políticas será fundamental a identificação do contexto do utilizador, que é composto pelo papel e município a que pertence, pois, com essa identificação é logo possível aplicar por exemplo a política associada a 'Uso interno' que apenas é acessível por parte dos administradores. As políticas associadas a 'Confidencial' são também críticas e com base na arquitetura já implementada é possível utilizar uma funcionalidade muito importante que é o remetente da mensagem, o 'user_id'. Uma função marcada como 'Confidencial' apenas poderá ter como alvo a identificação única proveniente do RabbitMQ. Para os outros casos terá que ser analisada uma forma de identificar o alvo ou alvos sobre quais as funções irão obter dados. Por fim, temos outro aspeto que é se a funcionalidade lê ou escreve dados no sistema. Essa informação será armazenada junto da política que uma função obtém, ou poderá estar implícito no nome da mesma. De que forma isso irá acontecer irá ser estudado no capítulo 4, onde será discutida e apresentada a implementação para esta solução de controlo de acesso.

Esta solução irá trazer alterações à arquitetura do sistema, nomeadamente aos seus componentes basilares que são o módulo de autenticação, módulo *Tenant* e as respetivas bases de dados. Os impactos da solução serão estudados no próximo subcapítulo.

3.5 IMPACTO DA SOLUÇÃO NO SISTEMA

A implementação desta solução no sistema atual terá impactos não só na arquitetura do sistema e as suas bases de dados, como também no desempenho do mesmo devido ao novo mecanismo de controlo de acesso. Espera-se que o desempenho do sistema embora possa degradar, não tenha impacto na utilização normal do mesmo, questão que será confirmada ou não pelos testes a realizar (capítulo 5) para avaliar o cumprimento dos requisitos definidos neste capítulo. O impacto em termos de arquitetura de sistema poderá ser mais evidente nas estruturas de base de dados, que tanto no caso do módulo de autenticação e nos módulos *Tenant* irão crescer e sofrer também algumas alterações nas estruturas de dados existentes. No caso da base de dados de autenticação as alterações devem-se a 3 novas funcionalidades e alterações de funcionamento. A primeira alteração é a forma de armazenar as palavras passe, pois a tabela '`users`' terá que guardar no lugar da palavra passe uma síntese resultante da palavra passe introduzia pelo utilizador, e mais um campo adicional que é o sal utilizado para gerar a síntese correspondente. Para a nova funcionalidade de banir ou bloquear o acesso a utilizadores que ultrapassem um certo limite de tentativas de acesso falhadas, será necessário a adição de novas tabelas de dados para armazenar informação como número de tentativas erradas, uma listagem de utilizadores bloqueados ou banidos, e em caso de estarem bloqueados, por quanto tempo. Por fim, será necessário armazenar informação relativamente às políticas de controlo de acesso e para isso serão necessárias pelo menos 2 tabelas de dados, uma para associar políticas aos campos das bases de dados, e outra para associar a política às funções existentes. Estas tabelas irão existir não só no módulo de autenticação, como também em todos os *Tenants* em execução permitindo a aplicação de políticas diferentes entre municípios. Nas bases de dados do *Tenant* as alterações poderão ir além das já mencionadas, poderão ir mais longe dependendo de funcionalidades ou requisitos que possam ser impostos pelas partes interessadas. Estas são as principais alterações que podemos esperar a nível das estruturas de bases de dados, mas também existirão alterações a nível da lógica de execução de ambos os componentes, autenticação e *Tenant*.

Ao nível da camada lógica dos módulos da aplicação irão existir também algumas alterações estruturais, e claro também lógicas, que se devem acima de tudo ao novo

mecanismo de controlo de acesso. Na figura 3.12, estão representadas 3 formas gerais de executar o mecanismo de decisão.

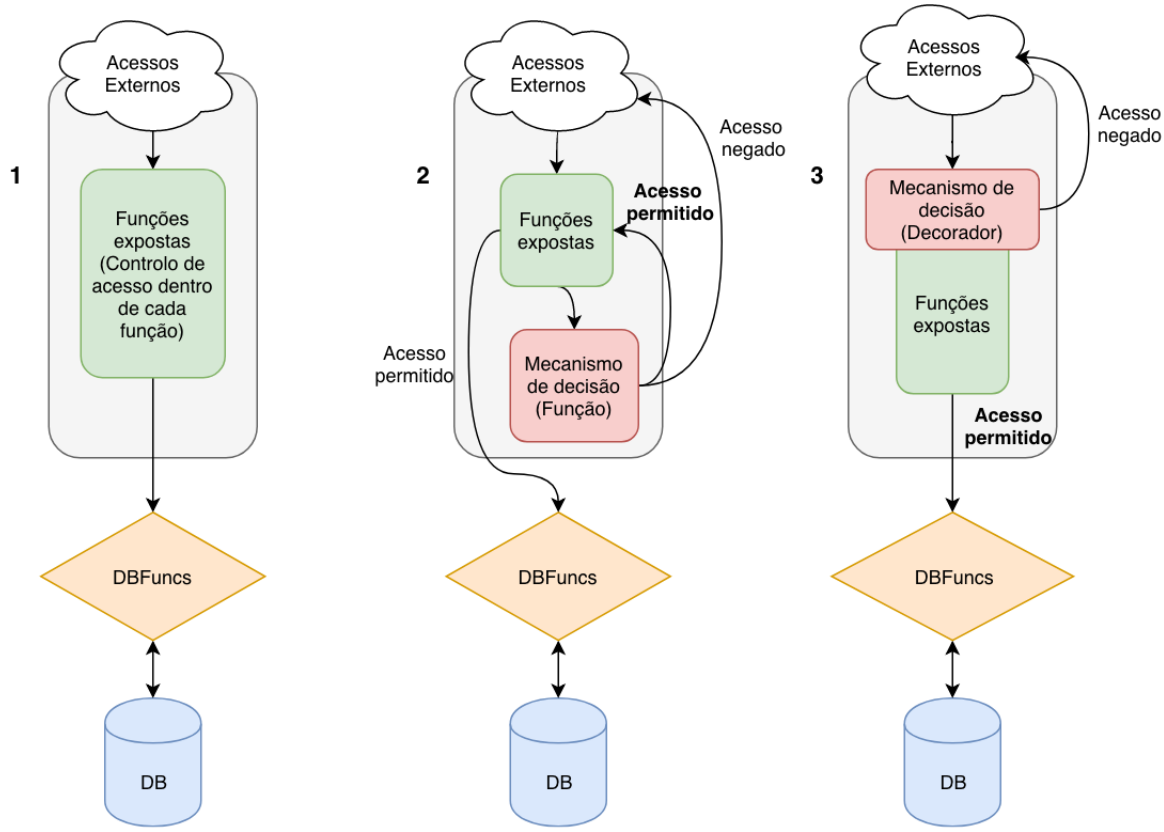


Figura 3.12: Possíveis aplicações do controlo de acesso no sistema.

No primeiro diagrama (1) está representada a que existe atualmente, onde dentro de cada função é programado um mecanismo de decisão personalizado e à medida das necessidades de controlo de acesso. Os outros dois diagramas representam 2 possibilidades de implementar a solução proposta. No diagrama do meio (2), o mecanismo de decisão é implementado numa função dedicada que é chamada sempre que uma função acedida externamente, e no último diagrama (3) a solução é implementada de forma semelhante, mas com recurso a decoradores Python. Os decoradores Python funcionam como um invólucro de uma função normal que altera o seu funcionamento sem que o código executado seja alterado [50]. Esta solução é uma prática adotada também na *Framework* Django, e de forma semelhante na *Framework* ASP.NET. Tem a vantagem de tornar o código mais legível e fácil de alterar no futuro. Este decorador, no nosso caso, iria decidir executar a função normalmente ou não a executar, dependendo da política associada a função executada.

Estes impactos nos vários componentes são apenas previsões que conforme as especificações técnicas existentes e as soluções que as tecnologias utilizadas permitem

adotar podem ou não acontecer. A preferência é adotar os decoradores, pois é uma solução testada e aprovada por *Frameworks* utilizadas a larga escala. No entanto, dificuldades técnicas e até lógicas da solução poderão alterar alguma solução proposta, e tema esse que será discutido no próximo capítulo, o capítulo 4.

capítulo 4

Life Payt - Implementação

Este capítulo irá apresentar a implementação das diferentes funcionalidades e alterações propostas no capítulo anterior. De seguida serão apresentados os objetivos da implementação tendo em conta os requisitos estabelecidos, e depois é apresentada a implementação das diferentes funcionalidades. A avaliação desta implementação será realizada no capítulo seguinte.

Os objetivos da implementação são cumprir os requisitos definidos no capítulo anterior que vão desde funcionalidades de segurança a boas práticas que cumpram os requisitos de desempenho e também legais. Um objetivo fundamental que não poderá falhar de forma alguma é o cumprimento dos requisitos legais, que com a introdução do novo RGPD se não forem respeitados pode levar a penalizações graves à organização responsável pelo LIFE PAYT.

4.1 implementação da proposta de solução

4.1.1 bloqueio de utilizadores

Para implementar a funcionalidade de bloquear ou banir utilizadores com base no número de tentativas falhadas de entradas de palavras passe, é necessário manter um registo dessas tentativas falhadas e de utilizadores bloqueados ou banidos. Esta informação deve ser armazenada em duas tabelas distintas, uma para as tentativas de acesso falhadas, em que é registado o identificar único do utilizador e data, e outra para fazer uma lista de utilizadores bloqueados ou banidos, em que é armazenado o identificador do utilizador e a data completo em que o bloqueio termina. Definiram-se, portanto, as seguintes tabelas na base de dados de autenticação (figura 4.1).

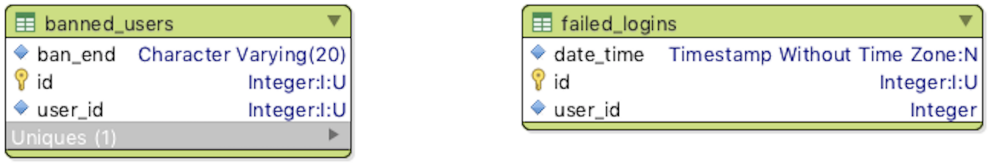


Figura 4.1: Estrutura de dados que suporta a funcionalidade de bloqueio de utilizadores.

Ambas as tabelas têm um identificador para cada entrada e o identificador único do utilizador banido e que falhou o acesso respetivamente. No caso da tabela '`failed_logins`' o '`user_id`' não é definido como único para que seja possível haver mais que uma entrada na tabela permitindo a contagem de entradas para um dado utilizador. Esta tabela poderia ter um campo de contagem de tentativas falhadas, mas isso não iria permitir ter um registo da data e hora de todas as tentativas falhadas, pelo menos sem que fosse necessário criar uma 3ª tabela. Assim, sempre que um utilizador falha o acesso ao portal, é introduzido na tabela um registo que contém o '`user_id`' e data completa. Sempre que o utilizador falhar o acesso, é adicionado um registo e após o registo é feita uma contagem de registos para o utilizador, caso o número de registos ultrapasse as 5 tentativas, o utilizador é adicionado à tabela '`banned_users`' onde é introduzido o '`user_id`' e a data de término do bloqueio, bloqueio este que tem uma duração de 5 minutos para 5 e 6 tentativas falhadas, 10 minutos entre a 7ª e a 11ª tentativa falhada, 15 minutos entre a 12ª e 15ª tentativa falhada, e a partir da 17ª tentativa falhada o utilizador é banido.

Os utilizadores banidos são identificados ao ter no campo '`ban_end`' o valor '-1' ao invés de uma data completa. De notar que, à 5ª tentativa falhada o utilizador é bloqueado durante 5 minutos, após os 5 minutos se o utilizador voltar a falhar, ou seja é a 6ª tentativa, volta a ser bloqueado por 5 minutos. Se após qualquer período de bloqueio o utilizador introduzir a palavra passe correta ou decidir repor a palavra passe, todos os registos para esse utilizador são eliminados da tabela '`failed_logins`'. O processo de *login* também muda, cada vez que a função de login é executada, ela verifica se o utilizador está banido ou bloqueado, e se for o caso apresenta durante quanto tempo a conta se encontra bloqueada (figura 4.2). Caso a conta seja bloqueada permanentemente, o utilizador deverá repor a sua palavra passe recorrendo à chave mestra com que ativou a conta, ou contactar os administradores de sistema caso a tenha perdido.

```
        async def set_banned(self, user_id):
                failed_logins = await self._count_failed_logins(user_id)

        if failed_logins[0][0] >= 5 and failed_logins[0][0] < 7:
                time = 5
                expire_dt = self.get_expire_dt(time)
                await self._set_ban(user_id, expire_dt)

                return 1, 5

        if failed_logins[0][0] >= 7 and failed_logins[0][0] < 12:
                time = 10
                expire_dt = self.get_expire_dt(time)
                await self._set_ban(user_id, expire_dt)

                return 1, 10

        if failed_logins[0][0] >= 12 and failed_logins[0][0] < 15:
                time = 15

                expire_dt = self.get_expire_dt(time)
                await self._set_ban(user_id, expire_dt)

                return 1, 15

        if failed_logins[0][0] > 17:
                time = '-1'

                expire_dt = self.get_expire_dt(time)
                await self._set_ban(user_id, expire_dt)

                return 1, -1

        return 0, 0
```

Bloco de código 15: Código de implementação (simplificado) que estabelece o período de bloqueio de um utilizador.

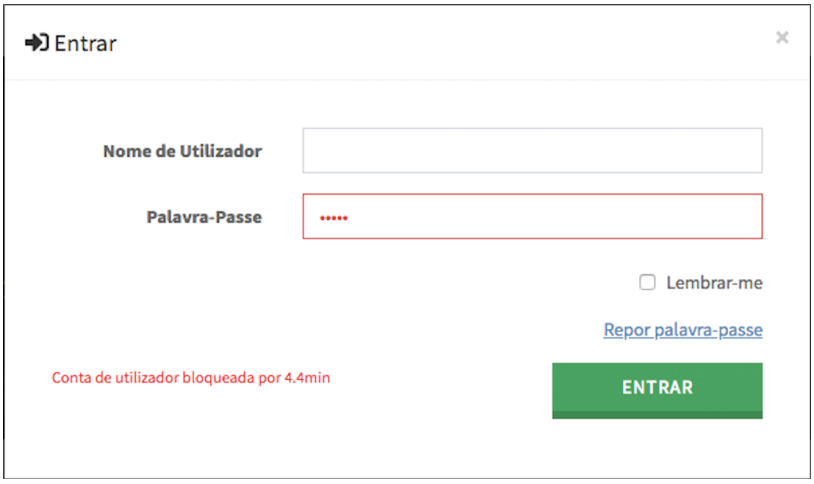


Figura 4.2: Mensagem apresentada ao utilizador que foi bloqueado.

4.1.2 PROTEÇÃO E ARMAZENAMENTO DAS PALAVRAS PASSE

Como explicado no capítulo 3, o armazenamento e a gestão das palavras passe terá que ser alterado de forma a que seja seguro o seu armazenamento e transporte no sistema e canais de comunicação. A solução a implementar passa por utilizar a função de derivação de chaves, a PBKDFv2, função que é possível configurar selecionando a função de síntese a utilizar, número de iterações e o sal associado. Destas três configurações possíveis é necessário armazenar apenas uma em base de dados, o sal, para que possamos ter um sal único para cada utilizador. Portanto a tabela de base de dados que armazena os utilizadores, '`users`', terá que sofrer algumas alterações, nomeadamente armazenar o sal e no campo da palavra passe armazenar a síntese gerada em vez da palavra passe. A figura 4.3 mostra a nova definição da tabela '`users`' tendo em conta o armazenamento seguro das palavras passe utilizando o PBKDFv2.

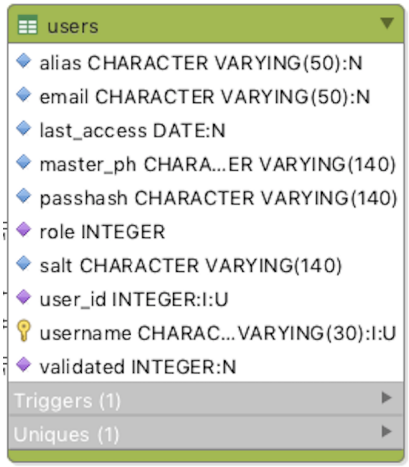


Figura 4.3: Definição da nova tabela que armazena os utilizador na base de dados de autenticação.

Portanto, a nova tabela passa a conter um campo adicional chamado '`salt`', que é o sal para a função de derivação de chave, com 140 caracteres. São 140 caracteres pois o sal é gerado pseudo-aleatóriamente através da função '`urandom`' biblioteca os.py [81], com o parâmetro de bytes a ser definido estaticamente em 64 bytes (512 bits) originando *strings* hexadecimais de tamanhos máximos de 128 caracteres. Este tamanho do sal é suficientemente alto para o tempo de vida do projeto de forma a que não hajam valores iguais do sal para utilizadores diferentes, pois com 128 caracteres hexadecimais temos 16^{128} combinações possíveis. No entanto optou-se por definir um tamanho maior para ter alguma flexibilidade e no futuro ser possível alterar a lógica do sal caso não se pretenda que seja gerado pseudo-aleatóriamente. O mesmo princípio aplicou-se para o campo da síntese da palavra passe, as sínteses geradas através do PBKDFv2 utilizando a função de síntese SHA-512, origina sínteses de 512 bits, ou seja, 64 bytes tal como no caso do sal o que significa um máximo de 128 caracteres hexadecimais. Isto origina em

```
class Cryptohash:
        async def doHash(self, pwd):
                salt_hex = urandom(64)
                hex_got = do_pbkdf2('sha512', pwd.encode(), salt_hex, 10000)
                pwdhash = hexlify(hex_got).decode()
                salt = hexlify(salt_hex).decode()

                return salt, pwdhash

        async def comparePW(self, salt, pwd, realpwd):
                hex_got = do_pbkdf2('sha512', pwd.encode(), unhexlify(salt), 10000)
                pwdhash = hexlify(hex_got).decode()

                if pwdhash == realpwd:
                        return 1
                else:
                        return 0
```

Bloco de código 16: Código de implementação (simplificado) do PBKDFv2 no sistema LIFE PAYT.

ambos os casos um espaço disponível de 12 caracteres para aumentar a complexidade de ambos os campos. O último parâmetro que já tinha sido definido na solução é o número de iterações que a função de derivação de chave executa, 10.000 iterações. Relembrando o diagrama de interação na figura 3.10, o armazenamento e validação da palavra passe processa-se da seguinte forma: o navegador envia uma síntese SHA-256 da palavra passe introduzida pelo utilizador, e no caso da definição da palavra passe é gerado um sal pseudo-aleatório, e é gerada a derivação da síntese da palavra passe com o sal gerado sendo ambos armazenados na base de dados. Para a verificação da palavra passe, o servidor obtém o sal associado ao utilizador a tentar fazer login, gera a derivação com o sal e palavra passe fornecida e compara com a derivação armazenada em base de dados. Se forem iguais, o utilizador tem acesso, caso contrário é guardada uma entrada de acesso sem sucesso e o utilizador vê o acesso negado.

No bloco de código 16 podemos ver as duas funções descritas anteriormente, a função '`doHash`' executa a função de derivação de chave PBKDFv2 com os parâmetros definidos, e a função '`comparePW`' verifica se a palavra passe introduzida é a mesma que está armazenada na base de dados. Uma vez que a função '`urandom`' da biblioteca Python 'os' gera valores hexadecimais, tal como a '`pbkdf2_hmac`' da biblioteca '`hashlib`' foi necessário descodificar o conteúdo hexadecimal para se tornar armazenável em base de dados.

4.1.3 CÓPIAS DE SEGURANÇA

O objetivo das cópias de segurança é salvaguardar os dados armazenados nas bases de dados, uma vez que se forem perdidos ou destruídos não poderão ser recuperados, ao contrário das aplicações que correm em contentores Docker. As aplicações que constituem o sistema LIFE PAYT, têm o seu código fonte público em repositórios de código públicos e privados. Para implementar as cópias de segurança, vamos recordar os requisitos definidos no capítulo anterior. Foi definido que as cópias de segurança devem ser protegidas recorrendo a uma cifra, ter uma periodicidade de 1 vez por dia, não ser armazenadas por mais do que 3 meses, devem ser incrementais, permitindo voltar a um ponto específico do passado e devem ser armazenadas num armazenamento remoto ou externo ao servidor.

Para cumprir com estes requisitos será utilizada a aplicação Duplicati, uma solução completa de cópias de segurança que permite configurar estas funcionalidades todas e até mais. No entanto a informação que queremos guardar em cópias de segurança encontra-se numa base de dados PostgreSQL e por isso é necessário guardar os dados em ficheiro. Para isso será necessário executar a funcionalidade '`pg_dump`' sempre que é executada a cópia de segurança. Esta é uma função que escreve para ficheiro a constituição da base de dados selecionada, permitindo guardar estrutura, dados ou ambos para recuperar posteriormente. Um exemplo básico da utilização desta funcionalidade é o seguinte comando, que escreve para ficheiro a base de dados '`db_test`' para o ficheiro '`db_test_dump.sql`'.

$$pg\_dump\ -d\ db\_test > db\_test\_dump.sql$$

O resultado deste comando é um ficheiro que contém código SQL para a criação da estrutura de dados e a inserção dos dados que se encontravam na base de dados aquando da execução do comando. Para bases de dados grandes os ficheiros podem tornar-se demasiado grandes, por isso existem algumas opções para o comando '`pg_dump`' que permitem comprimir o ficheiro. Através da opção '`-F`' (*format*), é possível escolher vários formatos de escrita do ficheiro gerado. A que mais nos interessa é o formato *custom* (c), é um formato que nos permite aquando da restauração selecionar a informação e tabelas a restaurar, ordenar os dados e, mais importante, comprime o ficheiro por defeito. Para uma base de dados que originava um ficheiro de 25 Megabytes, com a formatação *custom* o ficheiro passava a ter apenas 5 Megabytes, 5 vezes menor. Portanto o comando para escrever a base de dados para ficheiro é então:

$$pg\_dump\ -d\ db\_test\ -F\ c > db\_test\_dump.sql$$

Este comando será executado para cada uma das bases de dados, ou seja, para a base de dados de autenticação e para cada base de dados de *tenant* existentes. Os ficheiros que resultam desta operação serão depois o alvo das cópias de segurança, sendo após cada cópia de segurança eliminados. Este processo de escrever o conteúdo das bases de dados para ficheiro e de executar a cópia de segurança são dois processos distintos e que têm que ser sincronizados. O Duplicati permite o agendamento das cópias de segurança, ou a execução através da linha de comandos e uma vez que iremos recorrer ao agendamento da execução de ambos os processos através do *crontab* é mais conveniente e flexível executar as cópias de segurança através da linha de comandos. De forma a que os ficheiros alvos da cópia de segurança sejam eliminados após a sua execução, o processo de escrita dos dados para ficheiro começa às 2:55 da manhã, ficando depois 2 minutos em espera antes de eliminar os ficheiros que gerou. A execução da cópia de segurança está agendada para as 2:56, e o *script* responsável pela a sua execução tem apenas um comando exportado diretamente do Duplicati. Esta execução de ambos os processos têm a seguinte distribuição temporal (figura 4.4).

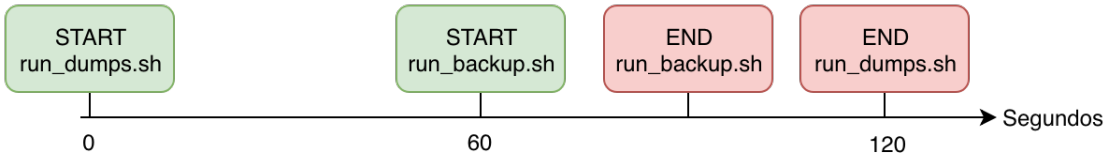


Figura 4.4: Distribuição temporal dos comandos de execução da cópia de segurança.

O processo da execução da cópia de segurança termina e pouco depois termina também o processo de escrita dos dados para os ficheiros que acaba a eliminar os ficheiros gerados através do comando '`shred`', que reescreve os ficheiros com informação ilegível e depois os elimina de forma a que os ficheiros desprotegidos continuem em disco (bloco de código 17).

Sempre que se adicionar uma base de dados nova ao sistema, basta alterar este *script* e adicionar mais uma linha com o comando '`pg_dump`' e respetivos argumentos. Estes comandos geralmente demoram pouco tempo a executar, no entanto deverá aumentar-se o tempo de espera à medida que o número de bases de dados a escrever aumenta também.

Relativamente à configuração das cópias de segurança, além das configurações básicas como a cifra das cópias, tempo de retenção das mesmas (3 meses), foram também configurados os alertas por e-mail. Todas as operações que tenham o resultado '`Error`', '`Warning`' ou '`Fatal`' irão notificar por e-mail as partes interessadas e responsáveis pelo servidor no projeto. Estas configurações adicionais originam o seguinte comando de execução da cópia de segurança. A localização de destino das cópias de segurança é

```
#!/bin/bash

AUTH='auth_dump'
AVEIRO='aveiro_dump'
CAOLI='caoli_dump'
DATE=`date +\%Y-\%m-\%d_\%H:\%M:\%S`
EXT='sqlc'

echo $DATE

cd tmp_dumps

pg_dump payt_caoli -F c > ${CAOLI}.${EXT}
pg_dump payt_aveiro -F c > ${AVEIRO}.${EXT}
pg_dump payt_auth -F c > ${AUTH}.${EXT}

sleep 120

shred -u *.sqlc
```

Bloco de código 17: Script que gera os ficheiros que contêm dados armazenados nas bases de dados do sistema.

como podemos ver um caminho local do servidor. Não foi possível em tempo útil ter um ou vários locais de armazenamento remotos para armazenar as cópias de segurança, por isso foi criado um diretório específico e apenas acessível ao utilizador 'paytbackups' para armazenar as cópias de segurança.

```
#!/bin/bash

mono /usr/lib/duplicati/Duplicati.CommandLine.exe backup \
file:///home/paytbackups/backups/home/paytbackups/tmp_dumps/ \
--backup-name=payt_storage --dbpath=/root/.config/Duplicati/EOUYSLXMJS.sqlite \
--encryption-module=aes --compression-module=zip --dblock-size=50mb \
--keep-time=3M --passphrase=******** --send-mail-level="\"Warning,Error,Fatal\""
--send-mail-body="%RESULT%" --send-mail-from=******** \
--send-mail-password=******** --send-mail-to=******** --send-mail-url=mail.ua.pt \
--send-mail-username=******** --send-mail-any-operation=false \
--disable-module=console-password-input
```

Bloco de código 18: Script que executa as cópias de segurança com o Duplicati a partir dos ficheiros gerados anteriormente.

Qualquer falha na execução deste comando 18 será notificada por e-mail com uma descrição detalhada do problema. Sempre que for necessário recuperar o sistema ou alterar o sítio onde está alojado, o processo de restauro é relativamente simples sendo possível executá-lo através da linha de comandos ou da *interface* da aplicação. O restauro do sistema numa eventual falha ou ataque, começaria com o restauro dos ficheiros que originam do '`pg_dump`', e estes teriam que depois serem lidos e processados

pelo PostgreSQL de forma a estabelecer as bases de dados conforme descrito nos ficheiros recuperados. Essa operação é executada através do seguinte comando:

pg_restore -C -F c -d basedados ficheiro.sqlc

Existem algumas opções para o comando, e vamos assumir que a base de dados que estamos a restaurar não existe, e por isso a opção '`-C`', que cria a base de dados '`basededados`', a partir do ficheiro '`ficheiro.sqlc`' com o formato *custom* tal como foi criado o ficheiro inicialmente. Existe ainda a possibilidade, uma vez que se trata de um ficheiro com a formatação *custom* de ordenar ou selecionar itens a serem restaurados. Mas de forma geral, este deverá ser o comando para restaurar a base de dados.

4.1.4 CONTROLO DE ACESSO

O controlo de acesso é fundamental à segurança do sistema, pois garante que apenas utilizadores autorizados acedam a dados sobre os quais têm permissão de leitura ou de escrita. Na arquitetura atual, o controlo de acesso do sistema era em termos práticos inexistente. No entanto alguma segurança é, e continuará a ser assegurada pelo sistema de mensagens que atribui um identificador único a cada utilizador e que é utilizado em algumas funções para identificar o alvo da ação que a função executa. Por exemplo, na função '`update_email`' (bloco de código 19) não é passado nenhum argumento para identificar o utilizador cujo e-mail será atualizado, pois o alvo da ação é o próprio chamador da função identificado pela variável '`user_id`'. Esse identificador é atribuído automaticamente para utilizadores que submetem as credenciais de acesso corretas e não é alterável.

No entanto em muitas outras funções as ações dizem respeito a dados de terceiros e por isso é importante garantir que apenas utilizadores autorizados as podem executar.

```
async def update_email(self, email, **kwargs):
        user_id = kwargs.get('userid')
        role = await self._get_role(user_id)
        ret = await self._.set_email(email, user_id)

        if not ret:
                return 0
        else:
                return 1
```

Bloco de código 19: Função '`update_email`' (simplificada) sem controlo de acesso.

```python
async def delete_user(self, target_usr, **kwargs):
        user_id = kwargs.get('user_id')
        role = await self._get_role(user_id)

        res = await self..delete_user(target_usr)
        if res:
                return 1

        return 0
```

Bloco de código 20: Função '`delete_user`' (simplificada) sem controlo de acesso.

A função '`delete_user`' por exemplo, é executada por um utilizador, mas a ação, tem como alvo um outro utilizador. No bloco de código 20, podemos ver nos argumentos da função a variável '`target_usr`' que é o '`user_id`' do utilizador a ser eliminado. Esta função, conforme está, pode ser executada em linha de comandos do navegador por qualquer utilizador com ou sem login feito, uma vez que não há qualquer verificação de quem está a executar a função.

Este exemplo é especialmente perigoso pois qualquer um pode a partir página inicial do LIFE PAYT executar a função e eliminar utilizadores introduzindo números aleatórios correspondentes a um '`user_id`'. Além de adivinhar o '`user_id`', é também necessário o atacante saber a assinatura da função e quais os seus argumentos, o que não é de todo difícil uma vez que se trata de um projeto *Open-Source* estando por isso o código fonte disponibilizado ao público. Nas figuras 4.5 e 4.6 podemos ver um exemplo deste ataque, em que a partir da página inicial sem qualquer login efetuado, é possível executar o comando relativo à função apresentada e assim eliminar o utilizador com o '`user_id`' igual a '22102'. De notar que estes ataques foram executados num ambiente de testes controlado e com dados falsos.

Figura 4.5: Execução da função '`delete_user`' sem qualquer autenticação.

Como esta, existem outras funções que podem afetar dados de utilização de terceiros, desde a sua consulta, alteração ou eliminação. É por isso importante implementar algum, controlo de acesso como o que foi proposto no capítulo anterior que toma decisões com base na informação acedida e no contexto de utilização.

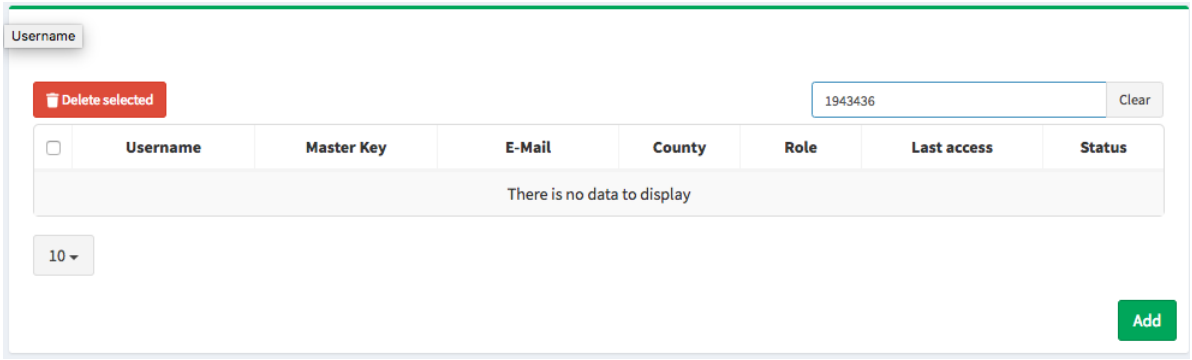


Figura 4.6: Resultado da execução da função 'delete_user' sem qualquer autenticação.

Em termos práticos, para conseguirmos ter uma implementação de acordo com o descrito no capítulo anterior, é necessário em primeiro lugar classificar cada objeto a que os utilizadores vão aceder. No anexo B podemos ver a classificação atribuída a cada objeto das bases de dados de autenticação e do *tenant*. Para fazer isso e para que possa ser alterado sempre que necessário, a nossa proposta é guardar essa informação na base de dados e a partir daí gerar as políticas para cada método registado e acessível, com base nessa informação. Cada módulo irá armazenar essa informação permitindo que diferentes municípios tenham políticas de acesso diferentes uma vez que os dados também podem ser diferentes. A estrutura de dados a adicionar em todos as bases de dados é a seguinte:

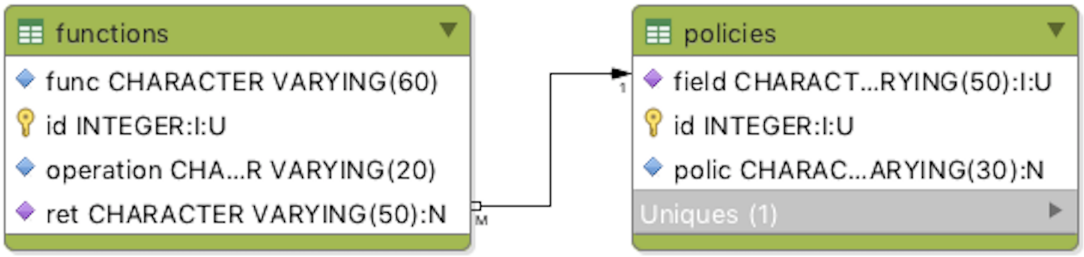


Figura 4.7: Estrutura de dados para armazenar informações sobre as políticas de controlo de acesso e permissões.

Na tabela '`policies`' será criada uma entrada para cada campo que representa um objeto e que existe na respetiva base de dados, identificado pelo campo '`field`'. A cada um será associado uma política que é identificada pelos 4 rótulos criados no desenho da solução: público, restrito, uso interno e proprietário. Estes campos são inseridos automaticamente na tabela através do seguinte comando aquando da criação da base de dados:

insert into authdb.policies (field) select concat(table_name::text, 'column_name::text) as field from information_schema.columns where table_schema = 'authdb';

Este comando é também executado sempre que o sistema reinicia, procurando novos campos que possam ter sido adicionados devido a implementação de novas funcionalidades. Na gestão das políticas que poderá acontecer no futuro, os administradores de sistema têm assim a possibilidade de alterar a política de acesso associada a um objeto específico e alterar o comportamento do controlo de acesso da função que lê ou escreve o respetivo campo.

A tabela '`functions`' irá estabelecer a ligação entre a política de controlo de acesso e a função. Cada entrada na tabela é constituída pelos campos '`func`' que representa o nome da função, '`operation`' que indica se a função lê, escreve ou faz ambos, e '`ret`' que indica qual o campo que retorna ('`r`' de *read*) ou escreve ('`w`' de *write*). Na *interface* de gestão de políticas é possível alterar a operação que cada função executa sobre o respetivo campo o que em alguns casos irá alterar o comportamento do controlo de acesso. Esta tabela tal como a '`policies`' tem que ser preenchida no arranque do sistema. Para tal vamos escrever um ficheiro de texto que irá conter a informação necessária para preencher a tabela no arranque do sistema e que deverá ser alterado sempre que uma nova funcionalidade for adicionada. O ficheiro contém em cada conjunto de 3 linhas a informação relativamente a cada função, sendo a primeira linha o nome da função, a segunda o campo que lê ou escreve e a terceira a operação sobre o campo. Esta solução para atualizar a tabela deve ser revista no futuro pois não é a forma ideal de o fazer, tal como a possibilidade de alterar o campo de leitura ou escrita que deverá ser implementada no futuro. A cada função é associado o campo que retorna ou escreve, no caso de serem mais que um, deve-se escolher o campo mais restritivo. Por exemplo, a função '`get_users`' retorna informação generalizada sobre os utilizadores como o e-mail, nome de utilizador e papel de utilizador. No caso destes campos mencionados, o mais restritivo é o campo do papel de utilizador, que é considerado de uso interno. Neste caso, a função tem permissão de uso interno e, portanto, apenas é executada pelo administrador de sistema. Na figura 4.8 temos a interface de gestão de políticas do módulo de autenticação, e podemos encontrar na tabela de permissões uma exceção a esta atribuição de permissões às funções.

A função '`payt_login`', é uma função de leitura, mas não retorna nenhuma informação, apenas se os dados introduzidos são validos ou não. Por isso nenhum campo é associado a esta função e quando assim é, a função é considerada pública.

Com estas duas tabelas preenchidas podemos então gerar a informação necessária ao mecanismo de controlo de acesso. Será gerado um objeto do tipo JavaScript Object Notation (JSON) que funciona como uma ACL que será consultada aquando da decisão de autorização de acesso à função. Esta informação é guardada num objeto JSON, pois

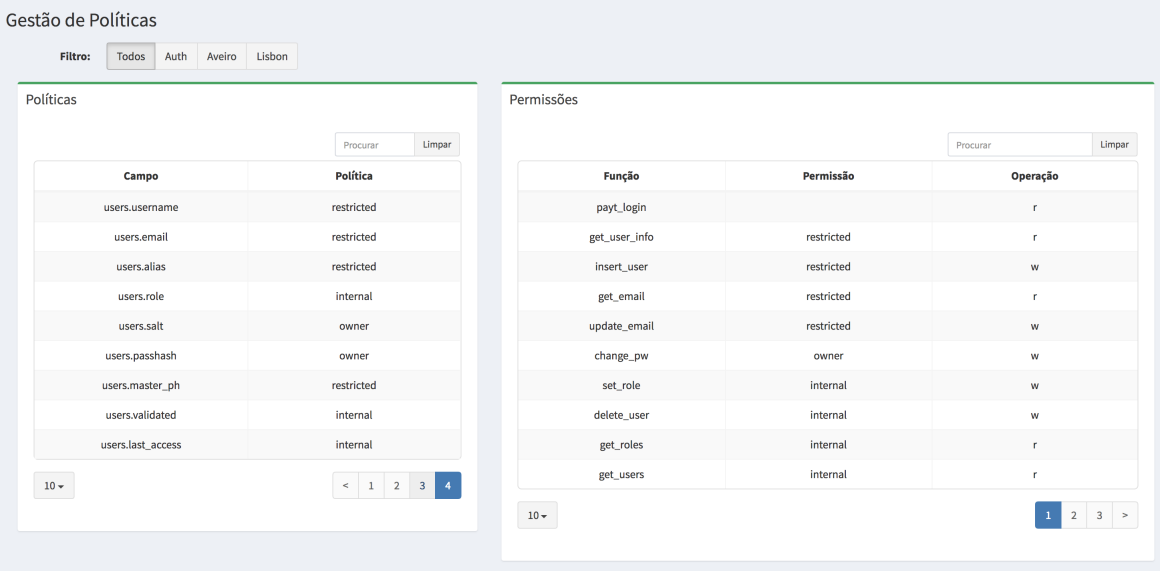


Figura 4.8: Interface de gestão de permissões e políticas do LIFE PAYT.

a consulta constante à base de dados dessa informação seria danosa para o desempenho do sistema e não é isso que se pretende. Na figura 4.9 podemos ver o gráfico com os resultados do teste de velocidade de acesso à base de dados, que utiliza a biblioteca '`asyncio`' tal como na implementação dos diversos módulos, para uma consulta com a junção de informação de duas tabelas. Para simular o acesso simultâneo como numa situação real, as chamadas à base de dados foram executadas a cada '`x`' segundos por 1, 2, 5, 7 e 10 utilizadores em simultâneo, em que '`x`' foi escolhido aleatoriamente a cada chamada e tomava valores entre 0.5 e 2. Foram executadas 10.000 chamadas à base de dados e para efeitos de comparação, foi escrito um objeto do tipo JSON que foi acedido nas mesma condições que a informação da base de dados.

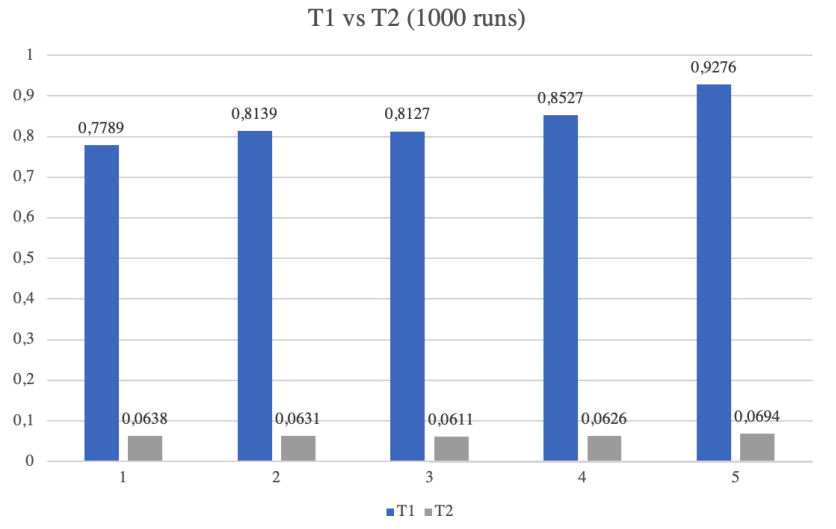


Figura 4.9: Gráfico dos resultados dos testes de tempo de acesso.

Como podemos ver o acesso à base de dados tem o potencial de tornar o sistema mais lento quando utilizado em massa. De 1 para 10 utilizadores em simultâneo, o tempo de acesso aumenta 23.35% contra um aumento de 11.02% no caso do acesso ao objeto. Além disso, o tempo de acesso médio por chamada ou acesso ao objeto é muito mais favorável ao segundo caso e garante que em caso de um crescimento acentuado de utilizadores, não seja comprometido o desempenho. Portanto, a partir da informação armazenada nas duas tabelas apresentadas será gerado um objeto que contém a informação de controlo de acesso para cada função que é possível executar, quer na interface quer na linha de comandos do navegador. O objeto JSON gerado a partir da base de dados terá a seguinte estrutura:

{nome_função: 'politica-operação'}

Portanto, para a função '`payt_login`', a entrada no objeto JSON irá corresponder ao seguinte:

{payt_login: 'public-r'}

A partir deste objeto podemos então começar o desenho da função de decisão de controlo de acesso. Como vimos no capítulo de solução, consideramos 2 soluções semelhantes para implementar o mecanismo de decisão, utilizar decoradores, ou uma função normal. Os decoradores permitem que após a execução do decorador seja devolvida a função decorada ou outra qualquer, o que encaixa perfeitamente no contexto de controlo de acesso. É devolvida a função chamada caso o controlo de acesso autorize a sua execução ou uma função que não faça nada caso não autorize e faça o registo do acesso negado. Esta solução é mais elegante e permite que as funções não necessitem de código adicional que seria necessário na implementação com a função dedicada, onde teria que ser avaliado em cada função o resultado do mecanismo de decisão. Recordando o esquema da figura 3.12 do capítulo anterior, o diagrama de funcionamento do mecanismo de controlo de acesso implementado com decoradores está representado na figura 4.10.

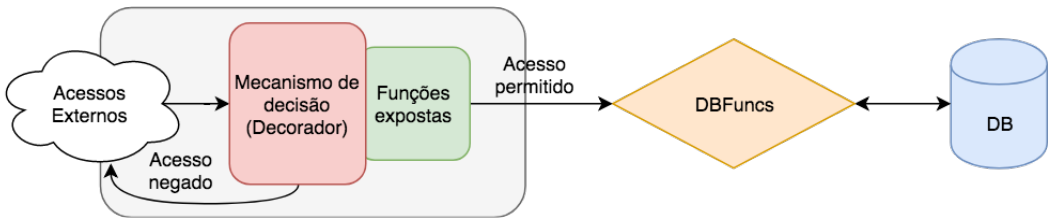


Figura 4.10: Diagrama de interação do controlo de acesso implementado com decoradores Python.

```
def access_control(function):
        async def wrapper(self, *args, **kwargs):
                user_id = kwargs.get('user_id')

                if isinstance(user_id, str):
                        if user_id == 'anonymous':
                                return await self.denied()

                if function.__name__ in self.perms:
                        p, op = self.get_perms(function.__name__)
                else:
                        return await function(self, *args, **kwargs)

                role = await self._get_role(user_id)
```

Bloco de código 21: Primeira parte da função '`access_control()`'.

Esta abordagem permite uma decisão rápida e eficiente mantendo o sistema a funcionar sem grandes atrasos. O código do mecanismo de decisão é igual para todos os módulos envolvidos, pois a decisão é focada nas políticas aplicadas aos objetos acedidos. Para outros casos de uso, este mecanismo de decisão terá que ser adaptado conforme as regras de cada política. Os seguintes blocos de código representam a função '`access_control`' que será o decorador de todas as funções sujeitas a controlo de acesso. Neste primeiro bloco 21 o decorador, cujo argumento é a função decorada, obtém todo o contexto de utilização da função.

Primeiro é obtido o '`user_id`' para identificar o chamador da função e é feito o registo nos *logs* do sistema. Com base no nome da função é obtida a permissão associada à mesma, separando a permissão e a operação em duas variáveis separadas 'p' e 'op'. Caso a função não tenha qualquer política associada a decisão é autorizar a sua execução. Decidiu-se optar por esta via pois a função decorada poderá estar ainda em testes, ou quando toda a manutenção das políticas for executada na interface de gestão, a função poderá ter que ser executável até que tenha a política correta associada sendo por isso algo temporário e por muito pouco tempo. Após obtida a política de controlo de acesso e tendo já registado o '`user_id`', é obtido o papel de utilizador do chamador da função. Com todo o contexto de utilização criado passamos então à decisão de autorização tendo em conta a política de acesso, representado no excerto de código 22.

Estas condições refletem a tabela criada no capítulo da solução, ou seja, a autorização é concedida com base no papel de utilizador, operação e se os dados são próprios ou de terceiros. Para identificar se a função atua sobre dados de terceiros, todas que aceitem um alvo da operação como argumento devem ter esse argumento com o nome '`target`' e a aparecer em primeiro lugar. Assim, sempre que a política associada a

```
def access_control(function):
...
        if p == 'public':
                target = inspect.getargspec(function)[1]
                if target == 'target' and op == 'w':
                        return await self.denied()

                return await function(self, *args, **kwargs)

        if p == 'restricted':
                if role == 'admin':
                        return await function(self, *args, **kwargs)

                target = inspect.getargspec(function)[1]
                if checkPolicy(p, op, role, target):
                        return await function(self, *args, **kwargs)

                return await self.denied()

        if p == 'internal':
                if role == 'admin' or isinstance(user_id, str):
                        return await function(self, *args, **kwargs)

                return await self.denied()

        if p == 'owner':
                target = inspect.getargspec(function)[1]
                if not target:
                        return await function(self, *args, **kwargs)

                return await self.denied()

        else:
                return await self.denied()

        return wrapper
```

Bloco de código 22: Segunda parte da função '`access_control()`'.

uma função necessite de verificar se a mesma atua sobre dados de terceiros, é obtido o primeiro argumento da função e verificado se esse argumento é '`target`'. De forma a minimizar o impacto no desempenho, esta operação é executada apenas quando necessário sempre em último caso. Ou seja, esta operação é executada apenas quando a política associada é '`owner`' ou '`restricted`', e no caso da última só após verificar se não é um administrador que está a chamar a função. Em todos os casos, se a decisão for autorizar o acesso, é devolvida a função decorada e os respetivos argumentos. Caso contrário, é devolvida a função '`denied`' que retorna '`None`' e faz o registo nos *logs* de uma tentativa de acesso negada. O portal do LIFE PAYT está desenhado de forma a que apenas funcionalidades adequadas sejam apresentadas aos utilizadores corretos.

No entanto os ataques eram possíveis e no futuro, quando novas funcionalidades forem implementadas e mais municípios entrarem em funcionamento com o sistema, não podemos prever todos os comportamentos e interações que serão desenhadas.

Existem algumas situações que não foram previstas, como o controlo de acesso à função '`payt_login`', mas aí o problema resolve-se ao não aplicar o decorador que executa o controlo de acesso, ou ao não incluir nenhum objeto alvo. Dadas as ações executadas pelo decorador, espera-se que o desempenho do sistema de forma global possa degradar, especialmente se for utilizado em massa pelos utilizadores. A cada chamada de uma função são adicionadas entre 5 linhas de código, no melhor dos casos, a 18 linhas de código no pior dos casos. Este fator, que é um dos requisitos, será avaliado tal como os restantes no capítulo 5.

capítulo 5

Resultados

Após a apresentação da solução e a respetiva implementação nos capítulos anteriores, vamos neste 5º capítulo fazer uma avaliação da solução tendo em conta os requisitos que foram definidos no capítulo 3. Para analisar o sucesso ou não da implementação vamos dividir os requisitos em requisitos técnicos e funcionais. Os requisitos técnicos são constituídos pelos requisitos de proteção contra ataques e desempenho do sistema, e os requisitos funcionais pelos requisitos que estabelecem cumprimento do novo RGPD e funcionais do controlo de acesso.

5.1 requisitos técnicos

Em termos técnicos, os requisitos apresentados no capítulo 3 definiam formas de proteger o sistema contra ataques que pudessem roubar informação ou comprometer o bom funcionamento do sistema. Os ataques para os quais foi necessário implementar medidas adicionais de segurança no sistema foram os ataques de descoberta de palavra passe, roubo das cópias de segurança e controlo de acesso. Os restantes ataques, SQLi, MiTM e DDoS já eram de uma maneira ou outra mitigados. No caso dos ataques SQLi a própria arquitetura e desenho do sistema, desde a página web ao *backend* que serve a página, evitam esses ataques. No caso da página web, os campos onde é possível definir uma estrutura clara do texto introduzido, como por exemplo o campo de e-mail ou número de telefone, só aceitam esse tipo de informação. Nos restantes campos, o *backend* sanitiza todo o texto introduzido de forma automática graças a utilização do *driver* de acesso à base de dados 'psycopg2' utilizado pela biblioteca 'aiopg'. Os ataques MiTM são desencorajados pela utilização de canais de comunicação seguros, nomeadamente HTTPS, e por boas práticas de segurança na definição das respostas do

servidor. Estas devem sempre conter o mínimo de informação possível, e se intercetadas e decifradas, não serem possíveis de associar diretamente a um utilizador ou pessoa. Por fim, o ataques DDoS são mitigados externamente pela infraestrutura responsável pela gestão dos servidores, sendo por isso desnecessário alguma alteração à aplicação nesse sentido. O roubo das cópias de segurança não é impossível, no entanto houve a preocupação de que estas fossem cifradas estando por isso mais protegidas. O próprio ficheiro que é escrito com toda a informação das bases de dados não é escrito em texto aberto, atuando como um fator de desencorajamento para um possível ataque. Como pudemos ver no capítulo da implementação, a cópia de segurança tem como destino um local no próprio servidor, ao invés do que foi escrito nos requisitos, num ou vários locais de armazenamento remotos e seguros. No entanto, por questões de limitação de tempo e de logística, as cópias de segurança estão para já armazenadas localmente. De seguida iremos apresentar o que acontece quando é feito uma tentativa de ataque de descoberta de palavra passe, SQLi e de acesso não autorizado a funções do sistema.

Descoberta da palavra passe por tentativa e erro

As medidas para combater ataques de descoberta de palavra passe, roubo das cópias de segurança e superar o controlo de acesso foram implementadas com sucesso. A descoberta de palavras passe é agora praticamente impossível, a não ser que seja descoberta em menos de 5 tentativas o que é muito improvável. Sempre que um utilizador erre a sua palavra passe mais do que 5 vezes, a sua conta será bloqueada durante 5 minutos. Após esse tempo o utilizador terá novas tentativas e se voltar a errar mais 2 vezes, volta a ser bloqueado. Isto até um máximo de 15 tentativas e 15 minutos de espera. Depois a conta é bloqueada permanentemente até que seja desbloqueada pela reposição da palavra passe ou administradores de sistema. Na figura 5.1 podemos

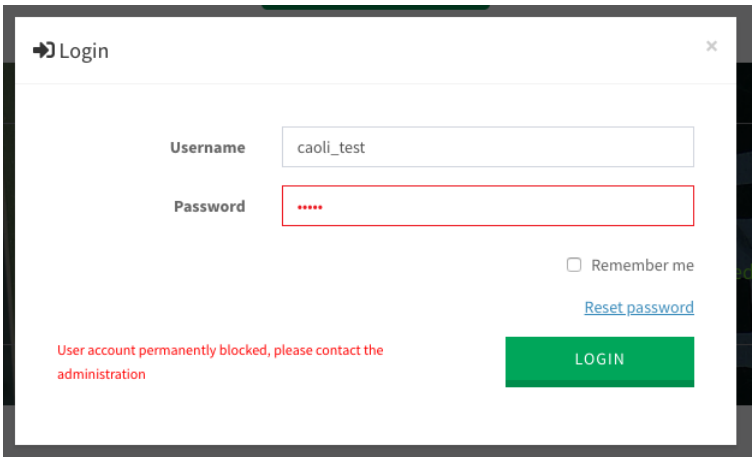


Figura 5.1: Mensagem apresentada a um utilizador bloqueado permanentemente.

ver a mensagem apresentada a um utilizador que é bloqueado após tentativas falhadas

excessivas de autenticação. Para reverter esta situação será necessário restabelecer a palavra passe com recurso à chave mestra fornecida no registo do utilizador no portal.

Ataques por SQL Injection (SQLi)

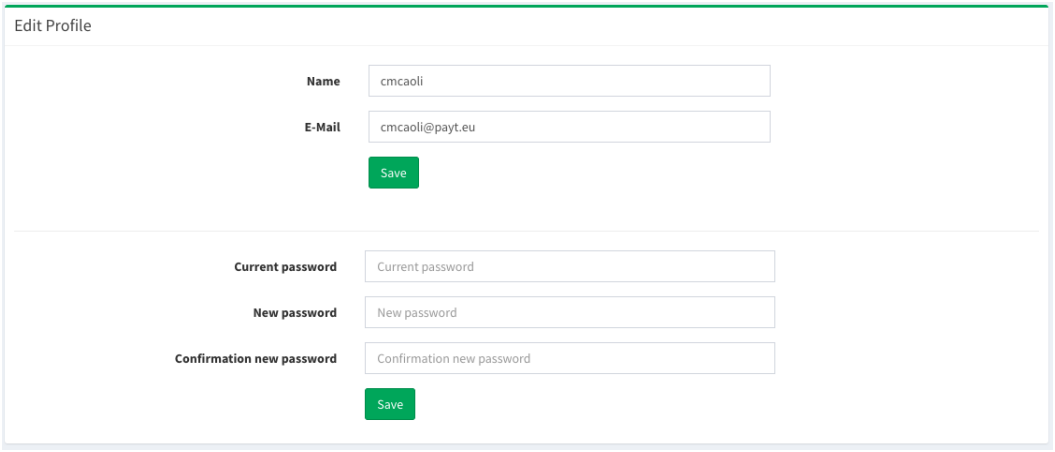


Figura 5.2: Interface de definições de um utilizador.

Para demonstrar a eficácia da proteção contra ataques SQLi oferecida pela biblioteca 'psycopg2' e verificação do conteúdo dos campos de texto, tentamos executar um ataque que tinha o objetivo de eliminar uma das tabelas na base de dados. Como podemos ver na figura 5.2 podemos alterar várias informações. No campo da palavra passe qualquer ataque é ineficaz, uma vez que o texto introduzido não chega ao servidor como texto legível. No campo de e-mail, apenas uma *string* com o formato correto é aceite pela *interface*, resta nos tentar no campo do nome. A função SQL que é executada através da alteração do nome é apresentada do bloco de código 23.

```
update authdb.users
set alias = %s
where user_id = %s
```

Bloco de código 23: Comando SQL executado aquando da edição do nome nas definições do utilizador.

Como podemos ver, o comando SQL é executado com dois parâmetros sendo o primeiro o texto introduzido pelo utilizador. A parametrização dos comandos SQL garante que os ataques SQLi sejam ineficazes. Nesta situação, para executar qualquer ataque SQLi, temos que introduzir vários comandos *sql* de uma vez só para que não ocorra qualquer erro ao nível da semântica. Se introduzirmos apenas "*'attack' where user_id=1; drop table users;*"a execução do comando iria falhar por um erro semântico, pois o comando terminaria com "*where user_id = x*"o que não é um comando completo. Por isso, a tentativa de ataque passa por introduzir a seguinte *string*: "*'attack' where*

user_id=1; drop table users; update table banned_users set ban_end=-1". A introdução desta *string* dará em teoria origem ao comando representado no bloco de código 24.

```
update users
set alias = 'attack'
where user_id=1;
drop table users;
update table banned_users
set ban_end=-1
where user_id = (uid)
```

Bloco de código 24: Comando SQL executado aquando da edição do nome nas definições do utilizador.

Mas como todos os comandos SQL são parametrizados é evitado como iremos ver qualquer ataque. Para efeitos de teste os *logs* registam o comando SQL e respetivos parâmetros. Ao introduzir a *string* de ataque mencionada, nada do que era previsto acontece, apenas é alterado o nome para o texto introduzido. Isto acontece porque o comando SQL é parametrizado, ou seja, executa apenas a função SQL já construída no servidor e insere como nome todo o texto introduzido pelo utilizador, mitigando qualquer comando malicioso nele contido. Uma análise aos *logs* de sistema mostra que o primeiro parâmetro é o seguinte: "*'attack' where user_id=1; drop table users; update table banned_users set ban_end=-1*". Ao transformar o texto completo numa *string* única e escapando caracteres especiais nele contido, qualquer comando deixa de ser um comando e é utilizado como o valor a ser inserido na base de dados.

Controlo de acesso - segurança e desempenho

Por último, temos o controlo de acesso que no início deste trabalho era praticamente inexistente. Após a implementação do mesmo podemos afirmar que este funciona e que é agora impossível por exemplo eliminar um utilizador sem sequer estar autenticado no portal. Se repetirmos o ataque apresentado no capítulo 4, desta vez a resposta é outra e o utilizador não é eliminado como podemos verificar na figura 5.3.

```
> window.payt_session.call('payt.auth.delete_user',{args: [1734777],
            callback: function (res){
                console.log(res)
            }.bind(this),
            exchange: 'auth'
        });
< undefined
  null
```

Figura 5.3: Tentativa falhada de executar a função 'delete_user' sem autenticação mínima necessária.

Como podemos verificar, a resposta é zero o que significa que a ação falhou. Uma consulta nos *logs* do sistema mostra que foi executada uma ação não permitida por um utilizador anónimo (figura 5.4).

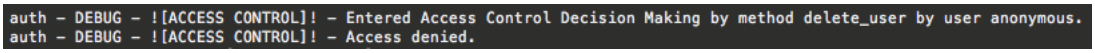


Figura 5.4: Mensagem registada aquando da tentativa de acesso sem autorização à função 'delete_user'.

Com a aplicação do controlo de acesso, é esperado que o desempenho global do sistema possa ser afetado. Um dos requisitos definidos no capítulo 3, foi o requisito do desempenho que não pode sofrer alterações significativas de forma a que o utilizador sinta lentidão ao utilizar o portal. Para verificar esse requisito foram executados alguns testes de desempenho que simulam uma utilização típica por parte dos 3 tipos de utilizadores, administradores, funcionários municipais e clientes. Cada teste foi executado 1000 vezes para obter uma média consistente, e cada execução consiste em ligar-se ao portal, executar a autenticação, e algumas das funções mais utilizadas por parte de cada tipo de utilizador. No caso do utilizador que representa a utilização por parte de administradores, foram executadas 7 funções ao todo. De todas, apenas 1 poderá ser mais exigente em termos de recursos que é a função de listagem de utilizadores. As funções chamadas pelo utilizador do tipo funcionário municipal, 10 no total, são as mais exigentes do teste, pois são as funções executadas logo na janela principal deste utilizador que apresentam dados estatísticos sobre consumos e faturas dos últimos meses. Além disso, o facto deste utilizador consultar acima de tudo dados de terceiros, requer uma análise por parte do mecanismo de controlo de acesso mais complexa. Por último, temos as ações executadas pelo utilizador que representa o cliente, que são ações mais simples e de consulta relativamente ao próprio utilizador, totalizando 9 funções executadas. Este será o uso de caso mais importante pois os clientes deverão ter sempre uma experiência de utilização mais agradável de forma a manter a boa imagem do portal perante a grande fatia dos seus utilizadores.

Executados os testes de desempenho, chegamos aos seguintes números apresentados na figura 5.5. Em média, cada execução apresentada por parte dos utilizadores do tipo administrador demorava 677,20 milissegundos, com o novo controlo de acesso demora 870,64 milissegundos o que representa um aumento de 28,56% do tempo de execução acumulado para todas as ações típicas para este utilizador. O número pode parecer grande, no entanto por cada ação individual este número representa apenas um aumento médio de 27,63 milissegundos de atraso por função 5.6. Os números apresentados pelo utilizador do tipo funcionário municipal apresenta também um aumento no tempo de execução, passou de 691,82 milissegundos para 912,98 milissegundos o que significa um aumento de 31,97%. Neste caso, por cada função executada a diferença é em média de 22,19 milissegundos de atraso por função. Por último, a utilização por parte de um utilizador do tipo cliente, demorava no total 633,71 milissegundos a terminar e passou a demorar 714,08 milissegundos. Aqui o aumento do tempo de execução é de apenas

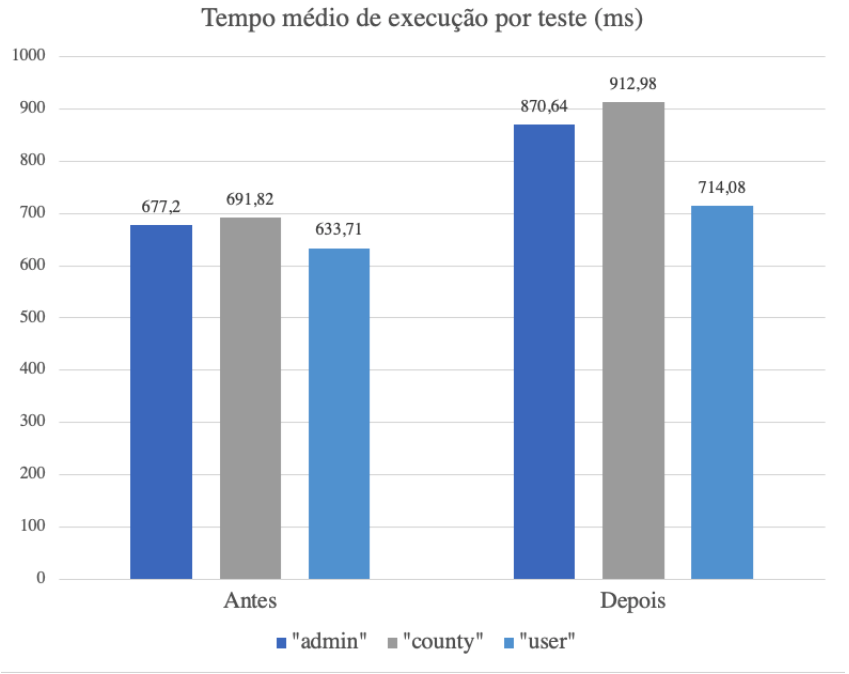


Figura 5.5: Tempo médio de execução de cada teste em milissegundos.

12,68%, ou 8,93 milissegundos de atraso em média por função o que é bastante residual.

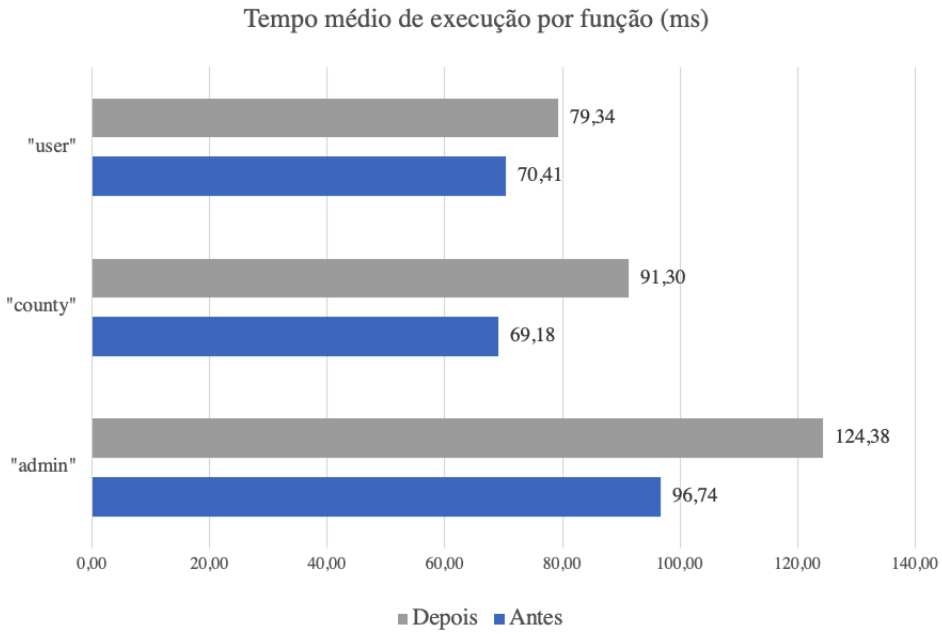


Figura 5.6: Tempo médio de execução por função em milissegundos.

Este impacto poderá ser maior se utilizado em massa de forma concorrente, mas visto que os números, principalmente para os utilizadores do tipo cliente são tão baixos, esse seria um problema que não teria origem propriamente no processamento resultante do mecanismo de decisão, mas sim de uma questão de recursos e desenho global da

arquitetura.

5.2 REQUISITOS FUNCIONAIS

Os requisitos funcionais englobam o cumprimento do novo RGPD, extremamente importante para o projeto. Além do cumprimento do RGPD, pelo menos a nível técnico tendo em conta os nossos interesses, será necessário avaliar o cumprimento dos requisitos do controlo de acesso. Em termos técnicos o controlo de acesso cumpre os requisitos pois não impõe um atraso significativa na resposta do sistema na sua utilização. A avaliação nesta secção é no sentido de perceber se com as políticas definidas e uma classificação dos objetos é possível ter uma solução funcional e que se consiga aplicar a todos os casos do sistema LIFE PAYT, e se é possível aplicar a outros sistema a mesma ideia.

Cumprimento do Regulamento Geral de Proteção de Dados (RGPD)

Com a entrada em vigor do RGPD as políticas de segurança tornam-se num dos principais focos ao nível do desenvolvimento de projetos que envolvam informação pessoal. O desenvolvimento do projeto LIFE PAYT coincide com a introdução e entrada em vigor do novo RGPD e por isso foi possível desenhar a arquitetura e funcionalidades já a pensar nesse mesmo regulamento. Relembrando o resumo sobre o RGPD no capítulo 2, os principais princípios do regulamento são: transparência, limitação de propósitos, minimização da recolha de dados, precisão dos dados, armazenamento limitado, integridade dos dados e responsabilidade sobre os dados.

O sistema LIFE PAYT recolhe apenas a informação estritamente necessária ao bom funcionamento do portal. Quando os funcionários municipais enviam os ficheiros referentes à faturação dos seus clientes, apenas os campos necessários são armazenados com os restantes a serem ignorados e o ficheiro enviado a ser eliminado. Além disso, a informação necessária aos utilizadores para obter acesso ao portal é também mínima, sendo possível aceder sem ter definido um endereço de e-mail para cada utilizador. Estas práticas vão de encontro ao princípio de minimização da recolha de dados, o que é um bom ponto de partida para a limitação de propósitos dos mesmos.

Os dados estritamente pessoais nos servidores que suportam a plataforma LIFE PAYT não serão utilizados para fins de estatísticas ou de perfilamento. Já os dados de consumo, poderão ser utilizados para esse efeito sendo disponibilizados no portal de dados abertos do projeto. Todos os dados de consumo podem ser utilizados para fins de estatísticas, uma vez que é possível obtê-los sem que haja uma associação direta a

um cidadão individual. Esta é uma política de privacidade estabelecida no anexo A, e que deverá ser transmitida a todos os utilizadores. Todo o processamento de dados de consumo que envolva a associação a um utilizador ou conjunto de utilizadores e que permita estabelecer padrões de comportamento não é executado na plataforma. Em relação aos dados pessoais, especialmente os que possam servir como contacto aos utilizadores (e-mail, telefone ou endereço), não poderão ser utilizados para fins de contacto sem a autorização explicita dos utilizadores. Cada utilizador poderá nas definições da sua conta optar por receber ou não notas informativas por e-mail, o que vai de encontro ao principio de limitação de propósitos e transparência da utilização de informação pessoal. Uma outra característica do sistema para sustentar estes princípios e que evita por exemplo o perfilamento dos utilizadores, é a anonimização dos utilizadores nas mensagens de monitorização do sistema. Os utilizadores são identificados por um identificador interno que não identifica diretamente um utilizador, são números sem qualquer significado relevante. Esta característica é importante, uma vez que as mensagens de monitorização não serão eliminadas, e se determinado utilizador pedir o esquecimento dos seus dados não será necessário eliminar as mensagens que poderão ser importantes no futuro para descobrir tentativas de ataques ou falhas gerais.

O sistema LIFE PAYT assume uma responsabilidade sobre os dados de forma a que todas as entidades envolvidas tenham apenas acesso à informação à qual podem ou devem ter. Ou seja, existe um controlo de acesso que não permite utilizadores consigam acesso a dados de outros utilizadores, ou dados referentes à administração do sistema por exemplo. Controlar o acesso à informação é essencial para para cumprir este principio. Este controlo de acesso é aplicado também na arquitetura do sistema. Uma vez que existem vários municípios envolvidos no projeto, é da nossa responsabilidade garantir que os dados de cada município não sejam misturados ou transmitidos entre si. Por essa razão, toda a parte lógica do sistema e de armazenamento de informação é separada quer em contentores Docker ao nível da aplicação, quer em bases de dados diferentes ao nível do armazenamento de informação. A aplicação destes princípios de segurança logo na concepção do sistema facilita a implementação de políticas de privacidade e segurança que garantem a transparência da utilização de dados pessoais.

Por fim, há também um cuidado especial com o armazenamento de informação pessoal nas cópias de segurança. Se um utilizador deixar de ser cliente, os seus dados pessoais devem ser eliminados assim que possível, tal como a associação do indivíduo aos seus consumos na plataforma. No entanto este processo ainda não está implementado, pois não houve ainda uma decisão por parte das entidades envolvidas de como processar esta eliminação de utilizadores da plataforma. Já no caso das bases de dados, a política de segurança deve constar que toda a informação pessoal e de consumos será

mantida nas cópias de segurança até um máximo de 3 meses após a sua eliminação. Isto acontece porque caso haja alguma falha crítica e seja necessário para efeitos de faturação retroceder 1 ou 2 meses, esses dados não sejam perdidos.

Em termos de legislação sobre a privacidade e proteção de dados, falta no entanto ainda a apresentação da política de privacidade ao utilizador e a opção de ser esquecido. A política de privacidade terá que ser aprovada por todas as partes envolvidas, e assim que o estiver poderá ser adicionada ao portal de forma a que seja apresentada aquando o registo do utilizador, e numa secção devidamente identificada de forma a que possa ser consultada sempre que necessário. O mesmo se aplica ao direito de esquecimento do utilizador. É necessário discutir com as autarquias como processar esse esquecimento, e que dados são necessários manter para que o modelo de negócio estabelecido funcione sem que nenhuma entidade seja prejudicada.

De uma forma geral, ao nível técnico e da implementação do sistema tudo foi feito de forma a que seja possível cumprir com o RGPD. É possível adicionar mais funcionalidades ao sistema que processem ou utilizem informação pessoal, mas tal como no caso da autorização para receber notas informativas por e-mail, isso é algo que deverá ser autorizado sempre pelo utilizador. A apresentação da política de privacidade e direito de esquecimento dos utilizadores é algo importante e que deverá ser resolvido num futuro próximo.

Controlo de acesso - funcionalidade

O controlo de acesso implementado no sistema LIFE PAYT globalmente funciona como era previsto no capítulo 3. A ideia de que determinado método é ou não autorizado a ser executado com base na classificação do objeto ou objetos que retorna ou escreve, e no contexto do utilizador foi aplicada com sucesso neste caso de uso. Existem no entanto algumas arestas por limar e questões que deverão ser revistas para outros casos de uso.

Em teoria, se um método X acede a um objeto Y com uma política de controlo de acesso Z, o método X estará sujeito a política Z. No entanto houve uma exceção no nosso caso, que é a função 'payt_login'. Esta função acede a alguns objetos cuja política associada é 'restrito' ou até mesmo 'proprietário', mas não retorna diretamente essas informações, apenas um resultado de uma comparação que os envolve. Mas esta função deve estar sempre acessível a qualquer utilizador não autenticado de forma a poder autenticar-se. Portanto, foi necessário criar uma exceção e não se atribuiu qualquer objeto de leitura ou escrita resultando na atribuição da política de controlo de acesso 'público'. Se esta função tivesse outro propósito qualquer para utilizadores

autenticados, a política atribuída seria 'proprietário'. Esta falta de controlo de acesso a esta função não é uma falha de segurança, e qualquer ataque à mesma é mitigado com a implementação do mecanismo de proteção contra a descoberta de palavras passe por tentativa e erro.

Outra questão com a atribuição dos objetos de leitura ou escrita para alguns métodos surge quando estes retornam mais que um objeto. E temos casos, como a função 'get_users' que retorna um conjunto de informação global sobre os utilizadores. Esta função foi criada apenas para acesso por parte dos administradores, e de forma a que o controlo de acesso aplique a política correta é preciso analisar os objetos que retorna e atribuir o objeto cuja política associada seja a correta. Neste caso, seria a política de 'uso interno' para que apenas os administradores de sistema pudessem aceder. Uma forma de contornar este problema seria criar uma estrutura de dados que suportasse vários objetos para cada função. Essa alteração no entanto iria tirar alguma simplicidade à solução, e teria efeito prático apenas em alguns casos isolados como este.

capítulo 6

Conclusões e Trabalho Futuro

Os projetos de *Smart Cities*, tal como o projeto LIFE PAYT, serão cada vez mais frequentes no futuro tanto em Portugal, como no resto do mundo. No caso do LIFE PAYT, é possível fazer uma gestão de resíduos urbanos mais cuidada e inteligente e com isso, permitir a poupança em termos económicos não só aos utilizadores como também às autarquias, resultando num impacto positivo no meio ambiente com a redução na produção de resíduos. O trabalho desta dissertação permitiu ao projeto a entrada em funcionamento com a aplicação de medidas de segurança adequadas, além de garantir o cumprimento com o novo Regulamento Geral de Proteção de Dados da União Europeia. Durante o trabalho foram tidos também em conta algumas diretrizes da certificação ISO 27001, um padrão importante na segurança informática e que ajuda a na concepção e manutenção de sistemas informáticos e respetivas organizações.

As alterações mais relevantes introduzidas no projeto com o trabalho desenvolvido nesta dissertação foram a alteração da gestão das palavras passe, medidas para evitar a descoberta da palavra passe por tentativa e erro, configuração de cópias de segurança e por fim, o desenvolvimento de um modelo de controlo de acesso. Este modelo de controlo de acesso foi algo criado de raíz, aplicando vários conceitos de modelos mais tradicionais tais como Controlo de acesso por papéis de utilizador, atributos, MAC e DAC. Esta solução de controlo de acesso pode em teoria ser aplicada a qualquer caso de uso, permitindo uma gestão diferente e potencialmente mais fácil da aplicação de políticas de controlo de acesso em sistemas informáticos. Como referido, foram tidos em consideração aspetos do ISO 27001, principalmente ao nível de controlo de acesso, que permitiu o desenvolvimento desta solução. O ponto de partida foi a classificação

da informação e a criação de um esquema de classificação. Foi também desenvolvida uma política de privacidade (anexo 6), que visa o cumprimento do novo RGPD e vai de encontro ao que o ISO 27001 recomenda. Com o desenvolvimento focado na segurança dos dados e respetiva criticalidade, conseguimos assim obter um sistema funcional que cumpre com o RGPD, como também cumpre alguns pontos do ISO 27001. Isto permite que no futuro seja possível de forma mais fácil tentar a obtenção da certificação pois os aspetos técnicos foram desenvolvidos tendo em consideração algumas diretrizes da certificação em mente.

Tendo terminado o trabalho desta dissertação com sucesso, é possível concluir que a aplicação de segurança logo desde a conceção dos sistemas pode tornar o processo muito mais simples. Neste caso, a arquitetura já estava pensada tendo em conta aspetos de segurança importantes, como a separação entre dados e lógica de aplicação, separação entre municípios quer na parte lógica quer nos dados armazenados, e também no sistema de mensagens que permite a identificação do utilizador, quer esteja autenticado ou não. A partir daí, a aplicação de boas práticas de segurança e de mecanismos como o armazenamento de palavras passe ou interdição de utilizadores abusivos são passos importantes para obter um sistema mais robusto. O controlo de acesso é fundamental, pois como demonstramos no capítulo 4 era em alguns casos inexistente e permitia ataques ao sistema. Com este controlo de acesso tivemos um impacto significativo na gestão e acesso à informação. Sempre que uma nova funcionalidade for adicionada, ela terá que ter em conta a classificação da informação e da política a ela associada para que possa utilizar o mecanismo de controlo de acesso implementado. Caso mais informação seja adicionada no futuro às bases de dados, também esta terá que ser classificada, quer seja com os rótulos existentes quer seja com a criação de novos rótulos. Não há sistemas infalíveis, mas para cada um existe um nível de segurança adequado e para o caso do LIFE PAYT, conseguimos encontrar uma nível de segurança bastante alto tendo em conta a informação que armazena e processa. Para isso será importante, principalmente em sistemas de *Smart City*, onde os dados são sobre as pessoas e são recolhidos em via pública, entender a criticalidade dos dados e definir políticas adequadas para os aceder e fazer uma implementação de acordo com essas políticas. Se executarmos estes dois passos, estaremos sempre mais próximos de obter um sistema mais seguro e robusto.

Como vimos no capítulo anterior 5, existem ao nível da legislação dois aspetos importantes por implementar, o direito ao esquecimento e a disponibilização da política de segurança aos utilizadores. Estes são os aspetos que deverão ser trabalhados e implementados num futuro próximo. Além disso, a solução de controlo de acesso também pode ser revista quando aplicado num outro caso de uso. A atribuição de apenas um campo de leitura ou escrita para uma função pode levar a que o processo de

atribuição de uma política de controlo de acesso não seja assim tão simples como era o nosso objetivo. Além disso, seria interessante validar esta ideia aplicando esta forma de controlo de acesso a outro caso de uso, de forma a perceber se é possível aplicar a mesma estrutura noutros exemplo ou se esta solução foi demasiadamente desenhada à medida deste projeto.

Referências

[1] Gemalto, «The Year of Internal Threats and Accidental Data Breaches 2017», Gemalto, rel. téc., 2017. [Online] Disponível: https://www.gemalto.com/press/pages/first-half-2017-breach-level-index-report-identity-theft-and-poor-internal-security-practices-take-a-toll.aspx.

[2] L. A. Skumatz, «Pay as you throw in the US: Implementation, impacts, and experience», *Waste Management*, vol. 28, pp. 2778–2785, 2008. DOI: 10.1016/j.wasman.2008.03.033.

[3] «ISO/IEC 27001:2013 - Information technology - Security techniques -Information security management systems - Requirements», International Organization for Standardization, Genébra, rel. téc., 2013. [Online] Disponível: https://www.iso.org/standard/54534.html.

[4] *Axis Communications şi Allied Telesis redefinesc conceptul de Smart City – ClubIT&C*, 2018. [Online] Disponível: http://www.clubitc.ro/2018/04/23/axis-communications-si-allied-telesis-redefinesc-conceptul-de-smart-city/ (acedido em 02/01/2018).

[5] L. Edwards, «Privacy, Security and Data Protection in Smart Cities: a Critical EU Law Perspective», 2015. DOI: 10.5281/zenodo.34501.

[6] V. Albino, U. Berardi e R. M. Dangelico, «Smart Cities: Definitions, Dimensions, Performance, and Initiatives», DOI: 10.1080/10630732.2014.942092.

[7] J. Wan, J. Liu, H. Suo e C. Zou, «Security in the Internet of Things: A Review», 2012. DOI: 10.1109/ICCSEE.2012.373.

[8] L. Atzori, A. Iera e G. Morabito, «Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm», *Ad Hoc Networks*, vol. 56, pp. 122–140, 2017. DOI: 10.1016/j.adhoc.2016.12.004.

[9] A. De Mauro, M. Greco e M. Grimaldi, «What is big data? A consensual definition and a review of key research topics», em *AIP Conference Proceedings*, vol. 1644, 2015, pp. 97–104, ISBN: 9780735412835. DOI: 10.1063/1.4907823. arXiv: arXiv:1804.02998v1.

[10] United Nations Publications, ed., *The World's Cities in 2016*. United Nations, Department of Economic e Social Affairs, Population Division, 2016, ISBN: 9211515491. [Online] Disponível: https://books.google.pt/books?id=CI5CvgAACAAJ.

[11] *Smart Cities - Why cities need to become smart now*. [Online] Disponível: https://www.iec.ch/smartcities/ (acedido em 30/11/2017).

[12] D. Bonte, «Smart Cities and Cost Savings», 2017. [Online] Disponível: https://www.chordant.io/white_papers/abi-research-smart-cities-and-cost-savings.

[13] *The Top 5 Smart Cities*. [Online] Disponível: https://www.iotworldtoday.com/2016/05/18/world-s-5-smartest-cities/ (acedido em 23/10/2017).

[14] L. Adler, *How Smart City Barcelona Brought the Internet of Things to Life | Data-Smart City Solutions*, 2016. [Online] Disponível: https://datasmart.ash.harvard.edu/news/article/how-smart-city-barcelona-brought-the-internet-of-things-to-life-789 (acedido em 02/12/2017).

[15] *Home Sentilo - Sentilo*. [Online] Disponível: http://www.sentilo.io/wordpress/ (acedido em 29/01/2018).

[16] *Smart Nation Initiatives | Singapore's Efforts to Improve Living*. [Online] Disponível: www.smartnation.sg/what-is-smart-nation/initiatives (acedido em 23/02/2018).

[17] *Moments of Life Initiative Begins with Supporting Every Young Child*. [Online] Disponível: https://www.smartnation.sg/initiatives/Services/moments-of-life-initiative-begins-with-supporting-every-young-child (acedido em 23/02/2018).

[18] M. Batllevell e K. Hanf, «The fairness of PAYT systems: Some guidelines for decision-makers», *Waste Management*, vol. 28, n.º 12, pp. 2793–2800, 2008. DOI: 10.1016/j.wasman.2008.02.031.

[19] J. Morlok, H. Schoenberger, D. Styles, J.-L. Galvez-Martos e B. Zeschmar-Lahl, «The Impact of Pay-As-You-Throw Schemes on Municipal Solid Waste Management: The Exemplar Case of the County of Aschaffenburg, Germany», *Resources*, 2017, ISSN: 2079-9276. DOI: 10.3390/resources6010008.

[20] *Facebook scandal 'hit 87 million users' - BBC News*. [Online] Disponível: https://www.bbc.com/news/technology-43649018 (acedido em 07/04/2018).

[21] *Facebook se conforme au RGPD et offre de nouvelles protections à tous, partout dans le monde | Facebook Newsroom France*, 2018. [Online] Disponível: https://fr.newsroom.fb.com/news/2018/04/facebook-se-conforme-aux-nouvelles-lois-sur-la-protection-de-la-vie-privee-et-offre-de-nouvelles-protections-a-tous-partout-dans-le-monde/ (acedido em 02/09/2018).

[22] *About The Open Web Application Security Project - OWASP*. [Online] Disponível: https://www.owasp.org/index.php/About_The_Open_Web_Application_Security_Project (acedido em 06/03/2018).

[23] *IoT Security Guidance - OWASP*. [Online] Disponível: https://www.owasp.org/index.php/IoT_Security_Guidance (acedido em 06/03/2018).

[24] *NATO - Declassified: For your eyes only*. [Online] Disponível: https://www.nato.int/cps/su/natohq/declassified_138449.htm (acedido em 12/01/2018).

[25] D. of Defense, «Trusted Computer System Evaluation Criteria - DoD 5200.28-STD», 1983. [Online] Disponível: https://csrc.nist.gov/csrc/media/publications/conference-paper/1998/10/08/proceedings-of-the-21st-nissc-1998/documents/early-cs-papers/dod85.pdf.

[26] «CIS/CSE 643: Computer Security (Syracuse University) Mandatory Access Control», 2009. [Online] Disponível: http://www.cis.syr.edu/~wedu/Teaching/cis643/LectureNotes_New/MAC.pdf.

[27] *Mandatory Integrity Control | Microsoft Docs*. [Online] Disponível: https://docs.microsoft.com/en-us/windows/desktop/secauthz/mandatory-integrity-control (acedido em 08/05/2018).

[28] *Discretionary Access Control*. [Online] Disponível: http://www.cs.cornell.edu/courses/cs5430/2015sp/notes/dac.php (acedido em 05/05/2018).

[29] R. S. Sandhu, E. J. Coyne, H. L. Feinstein e C. E. Youman, «Role-Based Access Control Models», *Computer*, vol. 29, n.º 2, pp. 38–47, fev. de 1996, ISSN: 0018-9162. DOI: 10.1109/2.485845.

[30] V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller e K. Scarfone, «Guide to Attribute Based Access Control (ABAC) Definition and Considerations», DOI: 10.6028/NIST.SP.800-162.

[31] L. Kerr e J. Alves-Foss, «Combining mandatory and attribute-based access control», em *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2016, ISBN: 9780769556703. DOI: 10.1109/HICSS.2016.328.

[32] D. R. Kuhn, E. J. Coyne e T. R. Weil, «Adding Attributes to Role-Based Access Control», *Computer*, vol. 43, n.º 6, pp. 79–81, jun. de 2010, ISSN: 0018-9162. DOI: 10.1109/MC.2010.155.

[33] *Difference Between Hashing and Encryption*. [Online] Disponível: https://www.ssl2buy.com/wiki/difference-between-hashing-and-encryption (acedido em 09/07/2018).

[34] A. Zúquete, *Segurança em Redes Informáticas*, 4ª Ed. Aum. FCA - Editora de Informática, 2013, ISBN: 978-972-722-767-9.

[35] *Google Online Security Blog: Announcing the first SHA1 collision*. [Online] Disponível: https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html (acedido em 23/07/2018).

[36] R. Rivest, *The MD5 Message-Digest Algorithm - RFC 1321*. (acedido em 03/08/2018).

[37] *More MD5 Collisions - Schneier on Security*. [Online] Disponível: https://www.schneier.com/blog/archives/2005/06/more_md5_collis.html (acedido em 14/08/2018).

[38] V. Klíma, «Finding MD5 Collisions-a Toy For a Notebook», rel. téc., 2005. [Online] Disponível: http://cryptography.hyperlink.cz.

[39] *Re: MD5 vs. SHA-1, Performance & Pedigree*. [Online] Disponível: http://www.sandelman.ca/ipsec/1996/05/msg00116.html (acedido em 13/08/2018).

[40] *MD5, SHA-1, SHA-256 and SHA-512 speed performance | Automation Rhapsody*. [Online] Disponível: https://automationrhapsody.com/md5-sha-1-sha-256-sha-512-speed-performance/ (acedido em 14/08/2018).

[41] *SHA-1/FIPS-180-1 - Rosetta Code*. [Online] Disponível: https://rosettacode.org/wiki/SHA-1/FIPS-180-1 (acedido em 30/09/2018).

[42] W. E. May, «Federal Information Processing Standards Publication 180-4 Secure Hash Standard (SHS)», 2012. DOI: 10.6028/NIST.FIPS.180-4.

[43] *DES Vulnerabilities*. [Online] Disponível: https://paginas.fe.up.pt/~ei10109/ca/des-vulnerabilities.html (acedido em 18/08/2018).

[44] B. Kaliski, *TWIRL and RSA Key Size*, 2003. [Online] Disponível: https://web.archive.org/web/20170417095741/https://www.emc.com/emc-plus/rsa-labs/historical/twirl-and-rsa-key-size.htm (acedido em 03/09/2018).

[45] J. Thakur e N. Kumar, «DES, AES and Blowfish: Symmetric Key Cryptography Algorithms Simulation Based Performance Analysis», rel. téc., jan. de 2011, pp. 6–12.

[46] *Django overview | Django*. [Online] Disponível: https://www.djangoproject.com/start/overview/ (acedido em 24/09/2018).

[47] *django.contrib.auth | Django documentation | Django*. [Online] Disponível: https://docs.djangoproject.com/en/2.1/ref/contrib/auth/#permission-model (acedido em 25/09/2018).

[48] *Using the Django authentication system - Permissions and authorization | Django documentation | Django*. [Online] Disponível: https://docs.djangoproject.com/en/2.1/topics/auth/default/#permissions-and-authorization (acedido em 25/09/2018).

[49] *Using the Django authentication system - Limiting access to logged in Users | Django documentation | Django*. [Online] Disponível: https://docs.djangoproject.com/en/2.1/topics/auth/default/#limiting-access-to-logged-in-users (acedido em 25/09/2018).

[50] *PEP 318 – Decorators for Functions and Methods | Python.org*. [Online] Disponível: https://www.python.org/dev/peps/pep-0318/ (acedido em 30/05/2018).

[51] *Open Source | The ASP.NET Site*. [Online] Disponível: https://www.asp.net/open-source (acedido em 26/05/2018).

[52] *Introduction to authorization in ASP.NET Core | Microsoft Docs*. [Online] Disponível: https://docs.microsoft.com/pt-pt/aspnet/core/security/authorization/introduction?view=aspnetcore-2.1 (acedido em 26/05/2018).

[53] *Razor Pages authorization conventions in ASP.NET Core | Microsoft Docs*. [Online] Disponível: https://docs.microsoft.com/en-us/aspnet/core/security/authorization/razor-pages-authorization?view=aspnetcore-2.1 (acedido em 26/10/2018).

[54] *Simple authorization in ASP.NET Core | Microsoft Docs*. [Online] Disponível: https://docs.microsoft.com/en-us/aspnet/core/security/authorization/simple?view=aspnetcore-2.1 (acedido em 26/09/2018).

[55] *Role-based authorization in ASP.NET Core | Microsoft Docs*. [Online] Disponível: https://docs.microsoft.com/en-us/aspnet/core/security/authorization/roles?view=aspnetcore-2.1 (acedido em 26/05/2018).

[56] *Policy-based authorization in ASP.NET Core | Microsoft Docs*. [Online] Disponível: https://docs.microsoft.com/en-us/aspnet/core/security/authorization/policies?view=aspnetcore-2.1 (acedido em 26/05/2018).

[57] *Claims-based authorization in ASP.NET Core | Microsoft Docs*. [Online] Disponível: https://docs.microsoft.com/en-us/aspnet/core/security/authorization/claims?view=aspnetcore-2.1 (acedido em 26/09/2018).

[58] D. Ferraiolo, R. Chandramouli, V. Hu e R. Kuhn, «A Comparison of Attribute Based Access Control (ABAC) Standards for Data Service Applications: Extensible Access Control Markup Language (XACML) and Next Generation Access Control (NGAC)», 2016. DOI: 10.6028/NIST.SP.800-178.

[59] *XACML Reference Architecture - Axiomatics*. [Online] Disponível: https://www.axiomatics.com/blog/xacml-reference-architecture/ (acedido em 24/02/2018).

[60] *File:XACML Architecture; Flow.png - Wikipedia*. [Online] Disponível: https://en.wikipedia.org/wiki/File:XACML_Architecture_%26_Flow.png (acedido em 24/10/2018).

[61] *WordPress Hosting Recommendations | WordPress.org*. [Online] Disponível: https://wordpress.org/hosting/ (acedido em 25/05/2018).

[62] *Features | WordPress.org*. [Online] Disponível: https://wordpress.org/about/features/ (acedido em 30/05/2018).

[63] *Capability Manager Enhanced | WordPress.org*. [Online] Disponível: https://wordpress.org/plugins/capability-manager-enhanced/ (acedido em 30/05/2018).

[64] *How are passwords stored in Linux (Understanding hashing with shadow utils)*. [Online] Disponível: https://www.slashroot.in/how-are-passwords-stored-linux-understanding-hashing-shadow-utils (acedido em 19/01/2018).

[65] *Password management in Django | Django documentation | Django*. [Online] Disponível: https://docs.djangoproject.com/en/2.1/topics/auth/passwords/ (acedido em 22/01/2018).

[66] *The Four Methods of Server Backup*. [Online] Disponível: https://www.r1soft.com/blog/the-four-methods-of-server-backup (acedido em 28/09/2018).

[67] *Incremental backups on Microsoft Azure Backup: Save on long term storage | Blog | Microsoft Azure*. [Online] Disponível: https://azure.microsoft.com/en-us/blog/microsoft-azure-backup-save-on-long-term-storage/ (acedido em 29/09/2018).

[68] *Backup Modes*. [Online] Disponível: https://support.unitrends.com/UnitrendsFree/s/article/000001743 (acedido em 29/09/2018).

[69] *Introduction - Duplicati 2 User's Manual*. [Online] Disponível: https://duplicati.readthedocs.io/en/latest/01-introduction/#the-backup-process-explained (acedido em 29/09/2018).

[70] L. P. M. de Almeida, «Plataforma de Software para Sistemas PAYT em Cidades Inteligentes», tese de mestrado, Universidade de Aveiro, 2018.

[71] *Cloudflare - The Web Performance & Security Platform | Cloudflare*. [Online] Disponível: https://www.cloudflare.com/#what-is-cloudflare (acedido em 21/10/2017).

[72] *ckan - The open source data portal software*. [Online] Disponível: https://ckan.org/ (acedido em 13/10/2017).

[73] *An Introduction to the ELK Stack for Logs and Metrics | ELK*. [Online] Disponível: https://www.elastic.co/webinars/introduction-elk-stack (acedido em 10/09/2017).

[74] *Messaging that just works - RabbitMQ*. [Online] Disponível: https://www.rabbitmq.com/ (acedido em 08/09/2017).

[75] *ZFS End-to-End Data Integrity | Oracle Jeff Bonwick's Blog*. [Online] Disponível: https://blogs.oracle.com/bonwick/zfs-end-to-end-data-integrity (acedido em 01/03/2018).

[76] Oracle, «Oracle Solaris 11 Express 2010.11 - What's New», 2010. [Online] Disponível: https://www.oracle.com/technetwork/server-storage/solaris11/documentation/solaris-express-whatsnew-201011-175308.pdf.

[77] B. Kaliski, «RFC 2898 - PKCS #5: Password-Based Cryptography Specification Version 2.0», rel. téc., 2000.

[78] *Parameter choice for PBKDF2 - Cryptosense*. [Online] Disponível: https://cryptosense.com/blog/parameter-choice-for-pbkdf2/ (acedido em 23/01/2018).

[79] *aiopg — aiopg 0.15.0 documentation*. [Online] Disponível: https://aiopg.readthedocs.io/en/stable/ (acedido em 30/11/2017).

[80] *Basic module usage — Psycopg 2.8.dev0 documentation*. [Online] Disponível: http://initd.org/psycopg/docs/usage.html#query-parameters (acedido em 30/11/2017).

[81] *os — Miscellaneous operating system interfaces — Python 3.7.1 documentation*. [Online] Disponível: https://docs.python.org/3/library/os.html (acedido em 23/10/2017).

Anexo A: Política de segurança e privacidade do LIFE PAYT

Política de Privacidade

Este aviso de privacidade informa de uma forma geral como nós, o consócio LIFE-PAYT, recolhemos e usaremos os seus dados pessoais, sendo que sempre que exista recolha dos mesmos, informaremos acerca da finalidade e do tratamento que será dado.

O tratamento que efetuamos da sua informação é diferenciado para o serviço fornecido e operamos sempre com o intuito de processar apenas a informação necessária para a tarefa. A interação que os utilizadores possuem com os sistemas do projeto LIFE-PAYT irá ditar como os seus dados são processados e os sistemas operam de forma independente, sem qualquer cruzamento de informação.

O que abrange esta política de proteção de dados?

Esta política de proteção de dados abrange apenas os dados da responsabilidade do consórcio LIFE-PAYT, no âmbito do fornecimento da sua página web no endereço www.portal.life-payt.eu, serviços de apoio aos municípios e serviços de dados abertos. Estes sistemas poderão ter ligações a outros serviços externos, que terão políticas de proteção de dados próprias e independentes da nossa. O consórcio LIFE-PAYT não assume qualquer responsabilidade sobre os dados fornecidos a essas plataformas.

O que são dados pessoais?

Os dados pessoais são qualquer informação, em qualquer suporte, relativa a uma pessoa singular, que permita a sua identificação direta ou mesmo indireta. Dados como o endereço IP, o nome, a morada, o número de telefone ou email são exemplos de dados pessoais, os quais tratamos com a maior seriedade.

Quem é o encarregado da proteção de dados?

O encarregado da proteção de dados é definido no âmbito do consórcio LIFE-PAYT, visando pela correta aplicação da regulamentação legal em vigor para os serviços prestados. Caso deseje contactar o encarregado da proteção de dados, poderá enviar um email para o seguinte contacto: dpo@life-payt.eu.

Os cookies recolhem informação pessoal?

As nossas páginas web utilizam cookies em determinadas áreas. Os cookies são ficheiros que armazenam informações no disco rígido ou browser do Utilizador. Os nossos cookies não recolhem informação pessoal e só poderão eventualmente recolher no futuro obtendo o seu consentimento. O Utilizador pode configurar o seu browser para recusar os cookies, porém nesse caso, o website ou partes do mesmo podem não funcionar corretamente. Os cookies são utilizados para monitorizar e analisar a utilização do website, assim como proporcionar a melhor experiência de utilização.

Além dos cookies gerados pelos nossos sistemas, utilizamos o serviço Google Analytics para a obtenção de estatísticas de acesso. Estes dados são utilizados por parte do consórcio apenas como uma forma de medir a adesão ao programa PAYT e mais nenhuma informação é retirada deles. Este serviço pode fazer uso de cookies próprios e possui uma política de privacidade específica, ver https://policies.google.com/privacy.

Como pode saber que dados possuímos sobre si?

Pode exercer o seu direito à informação enviando um email para o Encarregado da Proteção de Dados, através do endereço dpo@life-payt.eu, ou enviando uma carta postal para o endereço:

Encarregado da Proteção de Dados – Projeto LIFE-PAYT
Departamento de Eletrónica, Telecomunicações e Informática
Campus de Santiago
3810-193 Aveiro
Portugal

Os dados pessoais são partilhados com outras entidades?

Os dados individualizados não são partilhados com entidades externas ao consórcio. Para a criação de relatórios de atividade ou outras obrigações que nos são legalmente impostas, poderemos partilhar indicadores numéricos sobre o agregado de utilizadores.

Iremos partilhar todos os dados com qualquer entidade que possua o direito legal de os obter, sendo que este direito deverá ser expresso por uma ordem judicial ou outro documento legalmente obrigativo.

Quais as medidas adotadas para assegurar a segurança dos seus dados?

Os dados são armazenados num centro de dados, dotado de controlos de acesso e registos, que sempre que possível incluem o acesso ao edifício, à zona onde se encontram os dados, aos servidores e às aplicações informáticas utilizadas. Todas as ligações externas utilizam protocolos que garantem a autenticação e confidencialidade dos dados, nomeadamente através de Transport Layer Security (TLS), ou outro sistema de segurança idêntica. São também utilizadas cifras para proteger o conteúdo dos dados quer no transporte, quer no armazenamento dos mesmos.

Quais os tipos de dados pessoais tratados?

Ao aceder ao portal do LIFE PAYT dados sobre a utilização da página são recolhidos através do Google Analytics de forma anónima. É identificado o tempo de acesso total e individualizado por secção do portal. Além disso, dados de consumo e respetivos valores de faturação são processados para gerar estatísticas globais sobre um conjunto alargado de utilizadores, se não todos. O endereço de e-mail poderá ser, caso autorizado, utilizado para contacto por parte das entidades do consórcio LIFE PAYT. Todos os restantes dados pessoais servem apenas como identificação do indivíduo ou entidade para fins contratuais.

Como utilizamos a sua informação?

A informação pessoal é utilizada como identificação dos clientes que participam nos sistemas PAYT implementados pelas entidades que pertencem ao consórcio LIFE PAYT. Não são utilizados para gerar informação como padrões de comportamento ou identificação através do mesmo, nem partilhados com outras entidades legalmente proibidas. A única informação que é utilizada para gerar estatísticas e para previsão de dados futuros são os dados de consumo e faturação nos quais é descartada a ligação ao seu proprietário.

Porque recolhemos dados?

Recolhemos dados apenas para lhe apresentar informação do projeto de forma segura, considerando igualmente as necessidades de melhoria do fornecimento do mesmo. Sem os seus dados pessoais não é possível estabelecer uma ponte entre dados de consumo e cada indivíduo e apresentá-los de forma correta a cada tipo de utilizador.

Em que circunstâncias poderei ser contactado?

Podemos contactá-lo apenas nas situações que previamente autorizou ao utilizar o nosso sistema:

- Se adicionar um comentário ou outro conteúdo ao nosso sistema, poderemos contactá-lo no seguimento desse conteúdo e para poder potencialmente responder às dúvidas que coloque.
- Se enviar um pedido de contacto através dos formulários, para responder ao pedido de contacto.
- Se subscrever a nossa newsletter, iremos periodicamente enviar, um resumo das nossas atividades.
- Se enviar um correio eletrónico para o nosso contacto, iremos contactá-lo para responder a esse pedido.

Por quanto tempo serão conservados os seus dados?

Os dados serão conservados enquanto o utilizador possuir um registo no sistema, enquanto a informação que adicionou ao sistema for válida, ou enquanto mantiver a expressão de interesse em receber a nossa informação através das newsletters existentes. Além deste tempo, os sistemas possuem cópias de segurança, efetuadas de forma periódica, que poderão reter esta informação, num formato offline, durante um período superior. Este período nunca será superior a 4 meses e destina-se apenas à recuperação de informação em caso de perda catastrófica dos nossos sistemas.

Posso pedir a eliminação dos meus dados pessoais?

Todos os dados pessoais serão eliminados definitivamente da plataforma assim que um utilizador deixe de ser um utilizador ativo, e num prazo máximo de 4 meses das cópias de segurança. Se um utilizador ativo pretenda que os seus dados sejam eliminados, mesmo que continue a fazer parte do programa PAYT, este irá perder acesso à plataforma online e às vantagens oferecidas pelo programa PAYT.

Posso escolher os dados a serem processados e armazenados?

Nos sistemas LIFE PAYT aplicou-se o princípio de recolha mínima de informação sobre os seus utilizadores. Apenas dados estritamente necessários são recolhidos e utilizados apenas para o bom funcionamento do sistema e modelo de negócio. Dado que apenas a informação necessária é recolhida, a única opção de escolha que existe para os utilizadores é a autorização para utilizar o endereço de e-mail para receber notas informativas (newsletters). Os restantes dados são essenciais para o bom funcionamento da plataforma.

Quem vai aceder aos meus dados?

Os dados pessoais no sistema do consórcio LIFE PAYT apenas são acessíveis aos municípios participantes, e os dados são separados entre os municípios. Isto significa que os seus dados serão apenas acessíveis aos responsáveis do município onde reside e que tem a responsabilidade sobre os dados, contrato e faturação dos seus valores. Eles terão acesso ao nome completo, número de identificação fiscal, morada, contactos, consumos e valores de faturação. Os administradores de sistema terão acesso a uma listagem de utilizadores que será anonimizada, onde serão apenas apresentados o nome de utilizador, tipo de utilizador, data do último acesso, estado da conta e chave mestra. Os utilizadores terão acesso apenas aos seus próprios dados.

Obrigações legais e direitos

O sistema LIFE PAYT, o consórcio LIFE PAYT e os utilizadores da plataforma estão abrangidos pelo novo Regulamento Geral de Proteção de Dados da União Europeia 2016/679. Isto significa que esta política de privacidade tem que cumprir as regras estabelecidas pelo RGPD, e que os utilizadores têm os direitos descritos nesse mesmo RGPD.

Anexo B: Classificação dos objetos do sistema LIFE PAYT

Classificação da Base de Dados Tenant

Tabela	Campo	Tipo	Descrição	Rótulo
'zone'	'zone_id'	serial	Identificador único de cada objeto	**interno**
	'name'	varchar	Nome da zona	**público**
	'location'	varchar	Localização da zona	**público**
'producer'	'producer_id'	serial	Identificador único do objeto	**interno**
	'zone'	integer	Identificador da zona (zone.zone_id)	**interno**
'card_status'	'id'	serial	Identificador único de cada objeto	**interno**
	'status'	varchar	Estado do cartão	**restrito**
'id_card'	'card_id'	serial	Identificador único de cada objeto	**restrito**
	'status_id'	integer	Identificador do estado (card_status.id)	**interno**
'id_card_logs'	'id'	serial	Identificador único de cada objeto	**interno**
	'card'	varchar	Identificador do cartão (id_card.card_id)	**restrito**
	'producer'	varchar	Identificação do produtor	**restrito**
	'dt_ts'	timestamp	Identificação temporal da mensagem	**restrito**
	'log_msg'	varchar	Mensagem registada	**restrito**
'producer_card'	'pi_id'	serial	Identificador único de cada objeto	**interno**
	'started'	timestamp	Identificação temporal do início de atividade	**restrito**
	'ended'	timestamp	Identificação temporal do fim de atividade	**restrito**
	'card'	integer	Identificação do cartão associado	**restrito**
	'producer'	integer	Identificação do produtor (producer.producer_id)	**restrito**
'waste_type'	'id'	serial	Identificador único de cada objeto	**interno**
	'name'	varchar	Nome do tipo de lixo	**restrito**
	'repr_char'	char	Caracter identificador do tipo de lixo	**interno**
'container'	'container_id'	serial	Identificador único de cada objeto	**interno**
	'capacity'	integer	Capacidade do contentor	**restrito**
	'deposit_vol'	integer	Volume por saco depositado	**público**
	'lat'	double	Latitude	**público**
	'long'	double	Longitude	**público**
	'collect_days'	integer	Dias de recolha	**público**
	'waste_type'	integer	Tipo de lixo	**público**
	'cb_id'	integer	Identificador na API de Aveiro	**interno**
'prod_container'	'pc_id'	serial	Identificador único de cada objeto	**interno**
	'started'	timestamp	Identificação temporal do início de atividade	**restrito**
	'ended'	timestamp	Identificação temporal do fim de atividade	**restrito**
	'container'	integer	Identificação do produtor (container.container_id)	**restrito**
	'producer'	integer	Identificação do produtor (producer.producer_id)	**restrito**
'collection_op'	'operator_id'	serial	Identificador único de cada objeto	**interno**
	'name'	varchar	Nome do operador	**restrito**
'container_op'	'co_id'	serial	Identificador único de cada objeto	**interno**
	'started'	timestamp	Identificação temporal do início de atividade	**restrito**

	‘ended’	timestamp	Identificação temporal do fim de atividade	**restrito**
	‘container’	integer	Identificador único do contentor	**restrito**
	‘operator’	integer	Identificador único do operador	**restrito**
‘garbage_collect’	‘id’	serial	Identificador único do objeto	**interno**
	‘ts’	timestamp	Identificação temporal	**restrito**
	‘card’	varchar	Identificador do cartão	**restrito**
	‘container’	integer	Identificador do contentor	**restrito**
‘address’	‘id’	serial	Identificador único do objeto	**interno**
	‘address’	varchar	Primeiro campo do endereço	**restrito**
	‘address2’	varchar	Segundo campo do endereço	**restrito**
	‘parish’	varchar	Número	**restrito**
	‘city’	varchar	Cidade	**restrito**
	‘postal_code’	varchar	Código postal	**restrito**
	‘last_update’	timestamp	Data da última alteração	**restrito**
	‘alias’	varchar	Alias para a morada	**restrito**
‘activity’	‘activity_id’	serial	Identificador único do objeto	**interno**
	‘name’	varchar	Nome da atividade empresarial	**restrito**
‘organization’	‘organ_id’	serial	Identificador único do objeto	**interno**
	‘name’	varchar	Nome da Organização	**restrito**
	‘tax_id’	varchar	Número de identificação fiscal	**restrito**
	‘address’	integer	Identificação do endereço (address.id)	**restrito**
‘business’	‘b_id’	serial	Identificador único do objeto	**interno**
	‘name’	varchar	Nome do estabelecimento	**restrito**
	‘contract’	varchar	Tipo de contrato	**restrito**
	‘activity’	integer	Setor de atividade	**restrito**
	‘organization’	integer	Organização a que pertence	**restrito**
‘person’	‘id’	serial	Identificador único do objeto	**interno**
	‘name’	varchar	Nome	**restrito**
	‘tax_id’	varchar	Número de identificação fiscal	**restrito**
	‘contract’	varchar	Tipo de contrato	**restrito**
‘producer_party’	‘pp_id’	serial	Identificador único do objeto	**interno**
	‘client_id’	varchar	Identificação do contrato	**restrito**
	‘started’	timestamp	Identificação temporal do início de atividade	**restrito**
	‘ended’	timestamp	Identificação temporal do fim de atividade	**restrito**
	‘address’	integer	Identificação do endereço (address.id)	**restrito**
	‘person’	integer	Identificação da pessoa (person.id)	**restrito**
	‘business’	integer	Identificação do negócio (business.b_id)	**restrito**
	‘producer’	integer	Identificação do produtor (producer.id)	**restrito**
‘real_bill’	‘rbill_id’	serial	Identificador único do objeto	**interno**
	‘issue_date’	date	Data de emissão	**restrito**
	‘value’	numeric	Valor da fatura	**restrito**
	‘period_begin’	date	Inicio do período de faturação	**restrito**
	‘period_end’	date	Final do período de faturação	**restrito**
	‘party’	integer	Produtor associado	**restrito**
‘simulated_bill’	‘sbill_id’	serial	Identificador único do objeto	**interno**
	‘issue_date’	date	Data de emissão	**restrito**
	‘value’	numeric	Valor da fatura	**restrito**

	‘period_begin’	date	Inicio do período de faturação	**restrito**
	‘period_end’	date	Final do período de faturação	**restrito**
	‘party’	integer	Produtor associado	**restrito**
‘user_type’	‘type_id’	serial	Identificador único do objeto	**interno**
	‘type_name’	varchar	Nome do tipo	**interno**
‘users’	‘user_id’	integer	Identificador único do objeto	**interno**
	‘person’	integer	Identificação da pessoa (person.id)	**restrito**
	‘business’	integer	Identificação do negócio (business.id)	**restrito**
	‘type’	integer	Identificação do tipo (user_type.type_id)	**interno**
‘mailing’	‘user_id’	integer	Identificação do utilizador (users.user_id)	**restrito**
‘policies’	‘id’	serial	Identificador único do objeto	**interno**
	‘field’	varchar	Objeto da base de dados	**interno**
	‘polic’	varchar	Política associada ao objeto	**interno**
‘functions’	‘id’	serial	Identificador único do objeto	**interno**
	‘func’	varchar	Nome da função	**interno**
	‘ret’	varchar	Objeto que retorna	**interno**
	‘operation’	varchar	Operação (escrita ou leitura)	**interno**

Classificação da Base de Dados de Autenticação

Tabela	Campo	Tipo	Descrição	Rótulo
‘users’	‘user_id’	serial	Identificador único de cada objeto	**interno**
	‘username’	varchar	Nome de utilizador	**interno**
	‘email’	varchar	E-Mail do utilizador	**restrito**
	‘alias’	varchar	Alias da conta do utilizador	**proprietário**
	‘role’	integer	Identificação do papel no sistema (roles.id)	**interno**
	‘salt’	varchar	Sal da palavra passe	**proprietário**
	‘passhash’	varchar	Síntese da palavra passe	**proprietário**
	‘master_ph’	varchar	Chave mestra	**restrito**
	‘validated’	integer	Identificação do estado da conta (validation_status.id)	**interno**
	‘last_access’	date	Data do último acesso ao portal	**restrito**
‘customer’	‘id’	serial	Identificador único do objeto	**interno**
	‘username’	varchar	Nome de utilizador (users.username)	**interno**
	‘county’	integer	Identificação do município	**interno**
‘countyadmins’	‘id’	serial	Identificador único do objeto	**interno**
	‘username’	varchar	Nome de utilizador (users.username)	**interno**
	‘county’	integer	Identificação do município	**interno**
‘counties’	‘id’	serial	Identificador único de cada objeto	**interno**
	‘county’	varchar	Nome do município	**restrito**
‘validation_sts’	‘id’	serial	Identificador único de cada objeto	**interno**
	‘status’	varchar	Nome do estado	**interno**
‘policies’	‘id’	serial	Identificador único do objeto	**interno**
	‘field’	varchar	Objeto da base de dados	**interno**
	‘polic’	varchar	Política associada ao objeto	**interno**

‘functions’	‘id’	serial	Identificador único do objeto	**interno**
	‘func’	varchar	Nome da função	**interno**
	‘ret’	varchar	Objeto que retorna	**interno**
	‘operation’	varchar	Operação (escrita ou leitura)	**interno**
‘services’	‘id’	serial	Identificador único de cada objeto	**interno**
	‘name’	varchar	Nome do serviço	**interno**
	‘salt’	varchar	Volume por saco depositado	**proprietário**
	‘secret’	varchar	Latitude	**proprietário**
‘banned_users’	‘id’	serial	Identificador único do objeto	**interno**
	‘user_id’	integer	Identificador do utilizador	**interno**
	‘ban_end’	varchar	Final da interdição do utilizador	**interno**
‘failed_logins’	‘id’	serial	Identificador único do objeto	**interno**
	‘user_id’	integer	Identificador do utilizador	**interno**
	‘date_time’	timestamp	Identificador temporal do acesso falhado	**interno**

Anexo C: Lista de pontos de controlo do ISO 27001

Annex A
(normative)

Reference control objectives and controls

The control objectives and controls listed in Table A.1 are directly derived from and aligned with those listed in ISO/IEC 27002:2013[1], Clauses 5 to 18 and are to be used in context with Clause 6.1.3.

Table A.1 — Control objectives and controls

A.5 Information security policies		
A.5.1 Management direction for information security		
Objective: To provide management direction and support for information security in accordance with business requirements and relevant laws and regulations.		
A.5.1.1	Policies for information security	*Control* A set of policies for information security shall be defined, approved by management, published and communicated to employees and relevant external parties.
A.5.1.2	Review of the policies for information security	*Control* The policies for information security shall be reviewed at planned intervals or if significant changes occur to ensure their continuing suitability, adequacy and effectiveness.
A.6 Organization of information security		
A.6.1 Internal organization		
Objective: To establish a management framework to initiate and control the implementation and operation of information security within the organization.		
A.6.1.1	Information security roles and responsibilities	*Control* All information security responsibilities shall be defined and allocated.
A.6.1.2	Segregation of duties	*Control* Conflicting duties and areas of responsibility shall be segregated to reduce opportunities for unauthorized or unintentional modification or misuse of the organization's assets.
A.6.1.3	Contact with authorities	*Control* Appropriate contacts with relevant authorities shall be maintained.
A.6.1.4	Contact with special interest groups	*Control* Appropriate contacts with special interest groups or other specialist security forums and professional associations shall be maintained.
A.6.1.5	Information security in project management	*Control* Information security shall be addressed in project management, regardless of the type of the project.
A.6.2 Mobile devices and teleworking		
Objective: To ensure the security of teleworking and use of mobile devices.		

Table A.1 *(continued)*

A.6.2.1	Mobile device policy	*Control* A policy and supporting security measures shall be adopted to manage the risks introduced by using mobile devices.
A.6.2.2	Teleworking	*Control* A policy and supporting security measures shall be implemented to protect information accessed, processed or stored at teleworking sites.
A.7 Human resource security		
A.7.1 Prior to employment		
Objective: To ensure that employees and contractors understand their responsibilities and are suitable for the roles for which they are considered.		
A.7.1.1	Screening	*Control* Background verification checks on all candidates for employment shall be carried out in accordance with relevant laws, regulations and ethics and shall be proportional to the business requirements, the classification of the information to be accessed and the perceived risks.
A.7.1.2	Terms and conditions of employment	*Control* The contractual agreements with employees and contractors shall state their and the organization's responsibilities for information security.
A.7.2 During employment		
Objective: To ensure that employees and contractors are aware of and fulfil their information security responsibilities.		
A.7.2.1	Management responsibilities	*Control* Management shall require all employees and contractors to apply information security in accordance with the established policies and procedures of the organization.
A.7.2.2	Information security awareness, education and training	*Control* All employees of the organization and, where relevant, contractors shall receive appropriate awareness education and training and regular updates in organizational policies and procedures, as relevant for their job function.
A.7.2.3	Disciplinary process	*Control* There shall be a formal and communicated disciplinary process in place to take action against employees who have committed an information security breach.
A.7.3 Termination and change of employment		
Objective: To protect the organization's interests as part of the process of changing or terminating employment.		
A.7.3.1	Termination or change of employment responsibilities	*Control* Information security responsibilities and duties that remain valid after termination or change of employment shall be defined, communicated to the employee or contractor and enforced.
A.8 Asset management		
A.8.1 Responsibility for assets		

Table A.1 *(continued)*

Objective: To identify organizational assets and define appropriate protection responsibilities.		
A.8.1.1	Inventory of assets	*Control* Assets associated with information and information processing facilities shall be identified and an inventory of these assets shall be drawn up and maintained.
A.8.1.2	Ownership of assets	*Control* Assets maintained in the inventory shall be owned.
A.8.1.3	Acceptable use of assets	*Control* Rules for the acceptable use of information and of assets associated with information and information processing facilities shall be identified, documented and implemented.
A.8.1.4	Return of assets	*Control* All employees and external party users shall return all of the organizational assets in their possession upon termination of their employment, contract or agreement.
A.8.2 Information classification		
Objective: To ensure that information receives an appropriate level of protection in accordance with its importance to the organization.		
A.8.2.1	Classification of information	*Control* Information shall be classified in terms of legal requirements, value, criticality and sensitivity to unauthorised disclosure or modification.
A.8.2.2	Labelling of information	*Control* An appropriate set of procedures for information labelling shall be developed and implemented in accordance with the information classification scheme adopted by the organization.
A.8.2.3	Handling of assets	*Control* Procedures for handling assets shall be developed and implemented in accordance with the information classification scheme adopted by the organization.
A.8.3 Media handling		
Objective: To prevent unauthorized disclosure, modification, removal or destruction of information stored on media.		
A.8.3.1	Management of removable media	*Control* Procedures shall be implemented for the management of removable media in accordance with the classification scheme adopted by the organization.
A.8.3.2	Disposal of media	*Control* Media shall be disposed of securely when no longer required, using formal procedures.
A.8.3.3	Physical media transfer	*Control* Media containing information shall be protected against unauthorized access, misuse or corruption during transportation.
A.9 Access control		
A.9.1 Business requirements of access control		

Table A.1 *(continued)*

Objective: To limit access to information and information processing facilities.		
A.9.1.1	Access control policy	*Control* An access control policy shall be established, documented and reviewed based on business and information security requirements.
A.9.1.2	Access to networks and network services	*Control* Users shall only be provided with access to the network and network services that they have been specifically authorized to use.
A.9.2 User access management		
Objective: To ensure authorized user access and to prevent unauthorized access to systems and services.		
A.9.2.1	User registration and de-registration	*Control* A formal user registration and de-registration process shall be implemented to enable assignment of access rights.
A.9.2.2	User access provisioning	*Control* A formal user access provisioning process shall be implemented to assign or revoke access rights for all user types to all systems and services.
A.9.2.3	Management of privileged access rights	*Control* The allocation and use of privileged access rights shall be restricted and controlled.
A.9.2.4	Management of secret authentication information of users	*Control* The allocation of secret authentication information shall be controlled through a formal management process.
A.9.2.5	Review of user access rights	*Control* Asset owners shall review users' access rights at regular intervals.
A.9.2.6	Removal or adjustment of access rights	*Control* The access rights of all employees and external party users to information and information processing facilities shall be removed upon termination of their employment, contract or agreement, or adjusted upon change.
A.9.3 User responsibilities		
Objective: To make users accountable for safeguarding their authentication information.		
A.9.3.1	Use of secret authentication information	*Control* Users shall be required to follow the organization's practices in the use of secret authentication information.
A.9.4 System and application access control		
Objective: To prevent unauthorized access to systems and applications.		
A.9.4.1	Information access restriction	*Control* Access to information and application system functions shall be restricted in accordance with the access control policy.
A.9.4.2	Secure log-on procedures	*Control* Where required by the access control policy, access to systems and applications shall be controlled by a secure log-on procedure.

Table A.1 *(continued)*

A.9.4.3	Password management system	*Control* Password management systems shall be interactive and shall ensure quality passwords.
A.9.4.4	Use of privileged utility programs	*Control* The use of utility programs that might be capable of overriding system and application controls shall be restricted and tightly controlled.
A.9.4.5	Access control to program source code	*Control* Access to program source code shall be restricted.
A.10 Cryptography		
A.10.1 Cryptographic controls		
Objective: To ensure proper and effective use of cryptography to protect the confidentiality, authenticity and/or integrity of information.		
A.10.1.1	Policy on the use of cryptographic controls	*Control* A policy on the use of cryptographic controls for protection of information shall be developed and implemented.
A.10.1.2	Key management	*Control* A policy on the use, protection and lifetime of cryptographic keys shall be developed and implemented through their whole lifecycle.
A.11 Physical and environmental security		
A.11.1 Secure areas		
Objective: To prevent unauthorized physical access, damage and interference to the organization's information and information processing facilities.		
A.11.1.1	Physical security perimeter	*Control* Security perimeters shall be defined and used to protect areas that contain either sensitive or critical information and information processing facilities.
A.11.1.2	Physical entry controls	*Control* Secure areas shall be protected by appropriate entry controls to ensure that only authorized personnel are allowed access.
A.11.1.3	Securing offices, rooms and facilities	*Control* Physical security for offices, rooms and facilities shall be designed and applied.
A.11.1.4	Protecting against external and environmental threats	*Control* Physical protection against natural disasters, malicious attack or accidents shall be designed and applied.
A.11.1.5	Working in secure areas	*Control* Procedures for working in secure areas shall be designed and applied.
A.11.1.6	Delivery and loading areas	*Control* Access points such as delivery and loading areas and other points where unauthorized persons could enter the premises shall be controlled and, if possible, isolated from information processing facilities to avoid unauthorized access.

Table A.1 *(continued)*

A.11.2 Equipment		
Objective: To prevent loss, damage, theft or compromise of assets and interruption to the organization's operations.		
A.11.2.1	Equipment siting and protection	*Control* Equipment shall be sited and protected to reduce the risks from environmental threats and hazards, and opportunities for unauthorized access.
A.11.2.2	Supporting utilities	*Control* Equipment shall be protected from power failures and other disruptions caused by failures in supporting utilities.
A.11.2.3	Cabling security	*Control* Power and telecommunications cabling carrying data or supporting information services shall be protected from interception, interference or damage.
A.11.2.4	Equipment maintenance	*Control* Equipment shall be correctly maintained to ensure its continued availability and integrity.
A.11.2.5	Removal of assets	*Control* Equipment, information or software shall not be taken off-site without prior authorization.
A.11.2.6	Security of equipment and assets off-premises	*Control* Security shall be applied to off-site assets taking into account the different risks of working outside the organization's premises.
A.11.2.7	Secure disposal or re-use of equipment	*Control* All items of equipment containing storage media shall be verified to ensure that any sensitive data and licensed software has been removed or securely overwritten prior to disposal or re-use.
A.11.2.8	Unattended user equipment	*Control* Users shall ensure that unattended equipment has appropriate protection.
A.11.2.9	Clear desk and clear screen policy	*Control* A clear desk policy for papers and removable storage media and a clear screen policy for information processing facilities shall be adopted.
A.12 Operations security		
A.12.1 Operational procedures and responsibilities		
Objective: To ensure correct and secure operations of information processing facilities.		
A.12.1.1	Documented operating procedures	*Control* Operating procedures shall be documented and made available to all users who need them.
A.12.1.2	Change management	*Control* Changes to the organization, business processes, information processing facilities and systems that affect information security shall be controlled.

Table A.1 *(continued)*

A.12.1.3	Capacity management	*Control* The use of resources shall be monitored, tuned and projections made of future capacity requirements to ensure the required system performance.
A.12.1.4	Separation of development, testing and operational environments	*Control* Development, testing, and operational environments shall be separated to reduce the risks of unauthorized access or changes to the operational environment.
A.12.2 Protection from malware		
Objective: To ensure that information and information processing facilities are protected against malware.		
A.12.2.1	Controls against malware	*Control* Detection, prevention and recovery controls to protect against malware shall be implemented, combined with appropriate user awareness.
A.12.3 Backup		
Objective: To protect against loss of data.		
A.12.3.1	Information backup	*Control* Backup copies of information, software and system images shall be taken and tested regularly in accordance with an agreed backup policy.
A.12.4 Logging and monitoring		
Objective: To record events and generate evidence.		
A.12.4.1	Event logging	*Control* Event logs recording user activities, exceptions, faults and information security events shall be produced, kept and regularly reviewed.
A.12.4.2	Protection of log information	*Control* Logging facilities and log information shall be protected against tampering and unauthorized access.
A.12.4.3	Administrator and operator logs	*Control* System administrator and system operator activities shall be logged and the logs protected and regularly reviewed.
A.12.4.4	Clock synchronisation	*Control* The clocks of all relevant information processing systems within an organization or security domain shall be synchronised to a single reference time source.
A.12.5 Control of operational software		
Objective: To ensure the integrity of operational systems.		
A.12.5.1	Installation of software on operational systems	*Control* Procedures shall be implemented to control the installation of software on operational systems.
A.12.6 Technical vulnerability management		
Objective: To prevent exploitation of technical vulnerabilities.		

Table A.1 *(continued)*

A.12.6.1	Management of technical vulnerabilities	*Control* Information about technical vulnerabilities of information systems being used shall be obtained in a timely fashion, the organization's exposure to such vulnerabilities evaluated and appropriate measures taken to address the associated risk.
A.12.6.2	Restrictions on software installation	*Control* Rules governing the installation of software by users shall be established and implemented.
A.12.7 Information systems audit considerations		
Objective: To minimise the impact of audit activities on operational systems.		
A.12.7.1	Information systems audit controls	*Control* Audit requirements and activities involving verification of operational systems shall be carefully planned and agreed to minimise disruptions to business processes.
A.13 Communications security		
A.13.1 Network security management		
Objective: To ensure the protection of information in networks and its supporting information processing facilities.		
A.13.1.1	Network controls	*Control* Networks shall be managed and controlled to protect information in systems and applications.
A.13.1.2	Security of network services	*Control* Security mechanisms, service levels and management requirements of all network services shall be identified and included in network services agreements, whether these services are provided in-house or outsourced.
A.13.1.3	Segregation in networks	*Control* Groups of information services, users and information systems shall be segregated on networks.
A.13.2 Information transfer		
Objective: To maintain the security of information transferred within an organization and with any external entity.		
A.13.2.1	Information transfer policies and procedures	*Control* Formal transfer policies, procedures and controls shall be in place to protect the transfer of information through the use of all types of communication facilities.
A.13.2.2	Agreements on information transfer	*Control* Agreements shall address the secure transfer of business information between the organization and external parties.
A.13.2.3	Electronic messaging	*Control* Information involved in electronic messaging shall be appropriately protected.

Table A.1 *(continued)*

A.13.2.4	Confidentiality or non-disclosure agreements	*Control* Requirements for confidentiality or non-disclosure agreements reflecting the organization's needs for the protection of information shall be identified, regularly reviewed and documented.
A.14 System acquisition, development and maintenance		
A.14.1 Security requirements of information systems		
Objective: To ensure that information security is an integral part of information systems across the entire lifecycle. This also includes the requirements for information systems which provide services over public networks.		
A.14.1.1	Information security requirements analysis and specification	*Control* The information security related requirements shall be included in the requirements for new information systems or enhancements to existing information systems.
A.14.1.2	Securing application services on public networks	*Control* Information involved in application services passing over public networks shall be protected from fraudulent activity, contract dispute and unauthorized disclosure and modification.
A.14.1.3	Protecting application services transactions	*Control* Information involved in application service transactions shall be protected to prevent incomplete transmission, mis-routing, unauthorized message alteration, unauthorized disclosure, unauthorized message duplication or replay.
A.14.2 Security in development and support processes		
Objective: To ensure that information security is designed and implemented within the development lifecycle of information systems.		
A.14.2.1	Secure development policy	*Control* Rules for the development of software and systems shall be established and applied to developments within the organization.
A.14.2.2	System change control procedures	*Control* Changes to systems within the development lifecycle shall be controlled by the use of formal change control procedures.
A.14.2.3	Technical review of applications after operating platform changes	*Control* When operating platforms are changed, business critical applications shall be reviewed and tested to ensure there is no adverse impact on organizational operations or security.
A.14.2.4	Restrictions on changes to software packages	*Control* Modifications to software packages shall be discouraged, limited to necessary changes and all changes shall be strictly controlled.
A.14.2.5	Secure system engineering principles	*Control* Principles for engineering secure systems shall be established, documented, maintained and applied to any information system implementation efforts.

Table A.1 *(continued)*

A.14.2.6	Secure development environment	*Control* Organizations shall establish and appropriately protect secure development environments for system development and integration efforts that cover the entire system development lifecycle.
A.14.2.7	Outsourced development	*Control* The organization shall supervise and monitor the activity of outsourced system development.
A.14.2.8	System security testing	*Control* Testing of security functionality shall be carried out during development.
A.14.2.9	System acceptance testing	*Control* Acceptance testing programs and related criteria shall be established for new information systems, upgrades and new versions.
A.14.3 Test data		
Objective: To ensure the protection of data used for testing.		
A.14.3.1	Protection of test data	*Control* Test data shall be selected carefully, protected and controlled.
A.15 Supplier relationships		
A.15.1 Information security in supplier relationships		
Objective: To ensure protection of the organization's assets that is accessible by suppliers.		
A.15.1.1	Information security policy for supplier relationships	*Control* Information security requirements for mitigating the risks associated with supplier's access to the organization's assets shall be agreed with the supplier and documented.
A.15.1.2	Addressing security within supplier agreements	*Control* All relevant information security requirements shall be established and agreed with each supplier that may access, process, store, communicate, or provide IT infrastructure components for, the organization's information.
A.15.1.3	Information and communication technology supply chain	*Control* Agreements with suppliers shall include requirements to address the information security risks associated with information and communications technology services and product supply chain.
A.15.2 Supplier service delivery management		
Objective: To maintain an agreed level of information security and service delivery in line with supplier agreements.		
A.15.2.1	Monitoring and review of supplier services	*Control* Organizations shall regularly monitor, review and audit supplier service delivery.
A.15.2.2	Managing changes to supplier services	*Control* Changes to the provision of services by suppliers, including maintaining and improving existing information security policies, procedures and controls, shall be managed, taking account of the criticality of business information, systems and processes involved and re-assessment of risks.

Table A.1 *(continued)*

A.16 Information security incident management		
A.16.1 Management of information security incidents and improvements		
Objective: To ensure a consistent and effective approach to the management of information security incidents, including communication on security events and weaknesses.		
A.16.1.1	Responsibilities and procedures	*Control* Management responsibilities and procedures shall be established to ensure a quick, effective and orderly response to information security incidents.
A.16.1.2	Reporting information security events	*Control* Information security events shall be reported through appropriate management channels as quickly as possible.
A.16.1.3	Reporting information security weaknesses	*Control* Employees and contractors using the organization's information systems and services shall be required to note and report any observed or suspected information security weaknesses in systems or services.
A.16.1.4	Assessment of and decision on information security events	*Control* Information security events shall be assessed and it shall be decided if they are to be classified as information security incidents.
A.16.1.5	Response to information security incidents	*Control* Information security incidents shall be responded to in accordance with the documented procedures.
A.16.1.6	Learning from information security incidents	*Control* Knowledge gained from analysing and resolving information security incidents shall be used to reduce the likelihood or impact of future incidents.
A.16.1.7	Collection of evidence	*Control* The organization shall define and apply procedures for the identification, collection, acquisition and preservation of information, which can serve as evidence.
A.17 Information security aspects of business continuity management		
A.17.1 Information security continuity		
Objective: Information security continuity shall be embedded in the organization's business continuity management systems.		
A.17.1.1	Planning information security continuity	*Control* The organization shall determine its requirements for information security and the continuity of information security management in adverse situations, e.g. during a crisis or disaster.
A.17.1.2	Implementing information security continuity	*Control* The organization shall establish, document, implement and maintain processes, procedures and controls to ensure the required level of continuity for information security during an adverse situation.

Table A.1 *(continued)*

A.17.1.3	Verify, review and evaluate information security continuity	*Control* The organization shall verify the established and implemented information security continuity controls at regular intervals in order to ensure that they are valid and effective during adverse situations.
A.17.2 Redundancies		
Objective: To ensure availability of information processing facilities.		
A.17.2.1	Availability of information processing facilities	*Control* Information processing facilities shall be implemented with redundancy sufficient to meet availability requirements.
A.18 Compliance		
A.18.1 Compliance with legal and contractual requirements		
Objective: To avoid breaches of legal, statutory, regulatory or contractual obligations related to information security and of any security requirements.		
A.18.1.1	Identification of applicable legislation and contractual requirements	*Control* All relevant legislative statutory, regulatory, contractual requirements and the organization's approach to meet these requirements shall be explicitly identified, documented and kept up to date for each information system and the organization.
A.18.1.2	Intellectual property rights	*Control* Appropriate procedures shall be implemented to ensure compliance with legislative, regulatory and contractual requirements related to intellectual property rights and use of proprietary software products.
A.18.1.3	Protection of records	*Control* Records shall be protected from loss, destruction, falsification, unauthorized access and unauthorized release, in accordance with legislatory, regulatory, contractual and business requirements.
A.18.1.4	Privacy and protection of personally identifiable information	*Control* Privacy and protection of personally identifiable information shall be ensured as required in relevant legislation and regulation where applicable.
A.18.1.5	Regulation of cryptographic controls	*Control* Cryptographic controls shall be used in compliance with all relevant agreements, legislation and regulations.
A.18.2 Information security reviews		
Objective: To ensure that information security is implemented and operated in accordance with the organizational policies and procedures.		
A.18.2.1	Independent review of information security	*Control* The organization's approach to managing information security and its implementation (i.e. control objectives, controls, policies, processes and procedures for information security) shall be reviewed independently at planned intervals or when significant changes occur.

Table A.1 *(continued)*

A.18.2.2	Compliance with security policies and standards	*Control* Managers shall regularly review the compliance of information processing and procedures within their area of responsibility with the appropriate security policies, standards and any other security requirements.
A.18.2.3	Technical compliance review	*Control* Information systems shall be regularly reviewed for compliance with the organization's information security policies and standards.