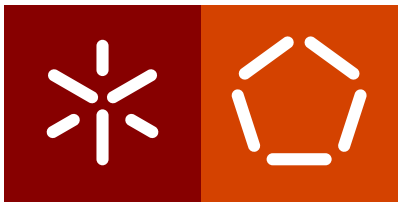
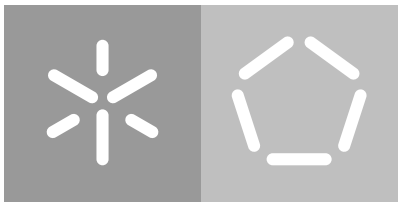


Universidade do Minho
Escola de Engenharia
Departamento de Informática

Sandra Clemente Baptista

Regulamento Geral de Protecção de Dados uma Plataforma de Apoio à Certificação

Julho 2021



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Sandra Clemente Baptista

Regulamento Geral de Protecção de Dados uma Plataforma de Apoio à Certificação

Master's dissertation
in Informatics Engineering

Dissertation supervised by
Pedro Rangel Henriques, Universidade do Minho

Julho 2021

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

Sandra Baptista _____

AGRADECIMENTOS

Em primeiro lugar, gostaria de agradecer ao professor Pedro Rangel Henriques pela paciência, apoio e por todo o acompanhamento e dedicação ao longo do desenvolvimento da mesma.

A toda a equipa da IdealMais, o meu agradecimento por me terem recebido e por todo o apoio ao longo destes meses na transmissão e discussão de ideias.

À minha família e amigos, a minha especial gratidão por toda a compreensão e apoio dado ao longo deste percurso. A vossa persistência permitiu que eu nunca desistisse dos meus sonhos e alcançasse os meus objetivos.

O meu enorme agradecimento a Sofia Caseiro por todas as ideias, passagem de conhecimentos, ajuda e disponibilidade prestada durante todo este processo.

Por último, o meu sincero agradecimento a Rute Painçal por ter caminhado ao meu lado durante esta incrível jornada.

ABSTRACT

The number of personal data circulating through computer applications and *web* today is quite large, leading the European Parliament to propose and approve a regulation aimed at protecting this data.

The General Data Protection Regulation (GDPR) is a new European legal framework that came into force on May 25, 2018 that focuses on the protection, collection and management of personal data, i.e. data about individuals. This regulation applies to all companies and organizations in the European Union

This Master's thesis in Computer Engineering focused on creating a solution that has as its main objective to facilitate the work of the people who are responsible for monitoring and ensuring that the regulation is being complied with.

This solution emerged in a work context from a sharing of ideas between me and the IdealMais entity, which proposed my integration in the architecture and development team of the solution.

The developed *backend* has as main features the management of measures, customers and compliance reports, i.e. reports intended to indicate the measures, their status and the actions that should be taken, serving as support for certification.

Keywords: certification, General Data Protection Regulation, *software*, programming, Data Protection Officer.

RESUMO

O número de dados pessoais que hoje em dia circula pelas aplicações informáticas e *web* é bastante grande, levando o Parlamento Europeu a propor e aprovar um regulamento destinado à proteção desses mesmos dados.

O Regulamento Geral de Proteção de Dados (RGPD) é um novo quadro jurídico europeu que entrou em vigor a 25 de maio de 2018 e que se centra na proteção, na recolha e na gestão de dados pessoais, ou seja, dados sobre indivíduos. Este regulamento aplica-se a todas as empresas e organizações da União Europeia

Esta tese de Mestrado em Engenharia Informática incidiu na criação de uma solução que tem como principal objetivo facilitar o trabalho das pessoas que são responsáveis por controlar e garantir que o regulamento está a ser cumprido.

Esta solução surgiu em contexto laboral a partir de uma partilha de ideias entre mim e a entidade IdealMais, que propôs a minha integração na equipa de arquitetura e desenvolvimento de *backend* da solução em causa.

O *backend* desenvolvido tem como principais características a gestão de medidas, clientes e relatórios de conformidade, isto é, relatórios destinados a indicar as medidas, o seu estado e as ações que devem ser tomadas, servindo de apoio à certificação.

Palavras-chave: certificação, Regulamento Geral de Proteção de Dados, *software*, programação, Responsável pela Proteção de Dados.

CONTEÚDO

LISTA DE FIGURAS

ACRÓNIMOS

DPO Responsável pela Protecção de Dados ou *Data Protection Officer*. 2–4, 6, 7, 11, 13, 20, 21, 24, 25, 34, 35

RGPD Regulamento Geral de Protecção de Dados. 2–8, 10–13, 20, 22, 25, 34

UE União Europeia. 11

1

INTRODUÇÃO

Este capítulo apresenta o enquadramento e motivação para esta dissertação e os objetivos que devem ser cumpridos. Além disso, também descreverá a estrutura do restante documento.

1.1 ENQUADRAMENTO

Hoje em dia, a *Internet* tornou-se um bem comum e indispensável, fazendo com que os utilizadores usufruam das suas múltiplas ferramentas. No entanto, esta facilidade de acesso e o banal fornecimento de dados levam ao surgimento de um problema: como proteger os dados de um indivíduo ou entidade.

Na União Europeia, as preocupações com a proteção de dados levaram à criação de um novo regulamento chamado Regulamento Geral de Protecção de Dados (RGPD) de 2016, que é dedicado à protecção de dados e à privacidade de todos os cidadãos individuais (1).

Este regulamento descontinua a Directiva 95/46/CE (2), dedicada à proteção de dados, que contém disposições e requisitos relacionados com o tratamento de dados pessoais de indivíduos.

Em 2018, na primavera, entrou como lei primária para regular as empresas e proteger os dados pessoais dos cidadãos da UE. No entanto, só entrou em vigor em 25 de maio de 2018.

Alguns dos principais requisitos de privacidade e protecção de dados do RGPD incluem:

- Requerimento do consentimento dos sujeitos para o processamento de dados;
- Anonimização dos dados recolhidos para proteção da privacidade;
- Fornecimento de notificações de violação de dados;
- Tratamento seguro da transferência de dados através das fronteiras;
- Exigência da nomeação, por parte das empresas, de um responsável pela proteção de dados para supervisionar o cumprimento do RGPD;

Para ajudar as empresas a regular as alterações necessárias no cumprimento do RGPD, foi criado o cargo de Responsável pela Protecção de Dados ou *Data Protection Officer* (DPO).

Esta função foi formalmente definida pela União Europeia como parte do seu Regulamento Geral de Proteção de Dados (RGPD).

O Responsável pela Proteção de Dados tem como funções a recolha e o tratamento dos dados pessoais dos cidadãos da UE, em conformidade com o artigo 37º do Tratado da UE. É responsável por fazer cumprir, dentro da empresa, as medidas aplicadas, formar o pessoal envolvido no processamento de dados e realizar auditorias regulares de segurança.

Tal como mencionado no artigo 39.º da RGPD, as responsabilidades do DPO incluem, entre outras, as seguintes:

- Informar a empresa e os funcionários sobre requisitos de conformidade importantes;
- Formar o pessoal envolvido no processamento de dados;
- Realizar auditorias para assegurar o cumprimento e resolver potenciais problemas;
- Ser o ponto de contacto entre a empresa e as autoridades de supervisão do RGPD;
- Monitorizar o desempenho e aconselhar sobre o impacto dos esforços de proteção de dados;
- Verificar os requisitos e atividades da empresa;
- Informar as pessoas sobre como os seus dados estão a ser utilizados, o seu direito a que os seus dados pessoais sejam apagados e que medidas a empresa implementou para proteger as suas informações pessoais.

1.2 MOTIVAÇÃO

O tema para esta dissertação de Mestrado surgiu no âmbito empresarial, do facto de ser programadora e da necessidade que todas as soluções atualmente criadas têm de respeitar o Regulamento Geral de Proteção de Dados (RGPD). Por este motivo, surgiu um espaço a explorar no mercado para o desenvolvimento de *software* especializado no controlo e gestão, com o fim de ajudar o DPO a cumprir as suas obrigações.

A função de um DPO deve ser desempenhada com base em competências profissionais, em especial, conhecimentos avançados em matéria de proteção de dados. Sendo detentor destes conhecimentos, o DPO deve ser capaz de desempenhar as funções que lhe são atribuídas no artigo 39.º, relativas à segurança e proteção de dados. Deve, nomeadamente, ter as seguintes funções:

- Sensibilizar e informar todos os que lidam com dados pessoais;
- Assegurar a duração das políticas de privacidade e proteção de dados;

- Controlar e regular o cumprimento do RGPD;
- Recolher informação para identificar atividades de processamento;
- Controlar e monitorizar a produção da Avaliação do Impacto da Proteção de Dados;
- Promover a privacidade por projeto e por abordagens padrão;
- Avaliar a exposição a riscos de violação de privacidade e mitigar com ações de melhoria;
- Recolher informações para identificar as atividades de tratamento;
- Manter os registos das atividades de processamento de dados atualizados;
- Monitorizar o cumprimento de contratos escritos de subcontratação;
- Promover a formação nas melhores práticas de proteção de dados;
- Ser o ponto de contacto com as pessoas em causa, a fim de esclarecer questões relacionadas com o tratamento de dados;
- Ser o ponto de contacto com as autoridades de supervisão.

A existência de uma função chamada DPO não é obrigatória, mas é recomendada em todas as organizações que lidam com dados pessoais ou sensíveis, devido ao que este profissional pode fazer.

Ao longo do processo de tese, tentar-se-á compreender o significado do Regulamento Geral de Proteção de Dados, do Responsável pela Proteção de Dados, bem como do processo de certificação e do processo de desenvolvimento de uma solução.

1.3 OBJETIVOS

Pensando no Regulamento Geral de Protecção de Dados e nas dificuldades que o DPO encontra ao desempenhar as suas funções pelo facto de utilizar ferramentas como papel e *Excel* na criação de documentação de apoio à certificação, pretende-se construir uma solução que permita melhorar e facilitar todo este processo, otimizando o seu trabalho.

O principal objetivo da tese aqui apresentada é construir uma ferramenta de *software* útil no contexto do RGPD, tendo em mente que pode ser adaptada a vários temas e modelos de negócios e que obedece a métricas reguladoras.

Deste modo, este trabalho tem os seguintes objetivos:

- Investigar sobre o Regulamento Geral de Proteção de Dados;
- Entender as funções e necessidades do responsável pela proteção de dados;

- Identificar o regulamento de métricas que precisa de ser respeitado;
- Entender a estrutura dos documentos que necessitam de ser criados e certificados;
- Analisar a melhor forma de encriptar e destruir dados.

1.4 ESTRUTURA DO DOCUMENTO

Esta dissertação de mestrado encontra-se organizada em 6 capítulos. Além deste primeiro capítulo introdutório, que contextualiza o leitor do assunto em estudo, as motivações e os objetivos deste trabalho, são ainda apresentados os restantes capítulos:

- **Capitulo 2** - Apresentação de um resumo da investigação realizada sobre os conceitos que estão diretamente relacionados com RGPD, tecnologias e certificação.
- **Capitulo 3** - Identificação do problema e dos desafios do desenvolvimento e aplicação das necessidades para o cumprimento do RGPD.
- **Capitulo 4** - Explicação do processo de desenvolvimento da solução e apresentação da mesma.
- **Capitulo 5** - Considerações finais relativas ao desenvolvimento da solução e sugestões para trabalhos futuros.

2

PROTEÇÃO DE DADOS E SEGURANÇA

Este capítulo é relativo ao estudo da Proteção de Dados e Segurança e à Revisão da Literatura, sendo descritos conceitos teóricos e científicos fundamentais relacionados com o tema desta dissertação como RGPD, DPO e outros conceitos importantes.

A figura 1 representa os pilares da implementação do RGPD.

Figura 1: Quatro pilares da implementação do RGPD

2.1 REGULAMENTO GERAL DE PROTEÇÃO DE DADOS

O Regulamento Geral de Proteção de Dados é um conjunto de novas regras a nível europeu, que visam garantir uma maior proteção dos dados pessoais dos cidadãos e entidades. Estes dados não são só de clientes e potenciais clientes, mas pertencem a qualquer pessoa que os disponibilize para que fiquem de alguma forma armazenados física ou informaticamente, como, por exemplo, colaboradores.

Para compreender o RGPD é necessário conhecer os princípios básicos sobre os quais está construído:

- "Legalidade, justiça e transparência- dados pessoais devem ser processados de forma legal, justa e transparente em relação à pessoa em causa;

- "Limitação de propósito- dados pessoais devem ser recolhidos para o fins específicos, explícitos e legítimos e não devem ser procurados de forma incompatível com esses fins;
- "Minimização de dados- dados pessoais devem ser adequados, relevantes e limitados ao necessário em relação aos fins para os quais são processados;
- "Precisão- dados pessoais devem ser precisos e, sempre que possível, mantidos atualizados;
- "Limitação de armazenamento- dados pessoais devem ser mantidos num formato que permita a identificação dos titulares de dados por não mais do que o necessário para as finalidades para as quais os dados pessoais são processados;
- "Integridade e confidencialidade- dados pessoais devem ser processados de uma maneira que possa garantir a segurança apropriada dos dados, incluindo proteção contra processamento não autorizado ou ilegal e contra perda, destruição ou dano acidental, usando medidas técnicas ou organizacionais apropriadas;
- "Imputabilidade- o DPO deve ser responsável e demonstrar a conformidade com o RGPD.

Durante o desfolhar do regulamento podemos encontrar uma série de regras divididas entre:

- “Informação aos titulares dos dados” - o que se deve fazer para informar as pessoas de como estão os seus dados a ser protegidos e quais os seus direitos;
- “Exercício dos direitos dos titulares dos dados” - o que é obrigatório cumprir no tratamento, edição e eliminação dos dados pessoais;
- “Consentimento dos titulares dos dados” - regras de como se deve obter a autorização dos titulares dos dados;
- “Natureza dos dados” - o que são dados sensíveis e como devem ser tratados;
- “Documentação e registo” - documentação atualizada que indique de que forma o regulamento está a ser cumprido, sendo necessário atualizações frequentes;
- “Subcontratação” - regras para a revisão dos contratos com todos os subcontratados que façam a gestão de dados;
- “Encarregado de Proteção de Dados” - DPO, explicado mais abaixo;

- "Processos de Segurança e Tratamento de Dados" - medidas técnicas que garantam a diminuição do risco de roubo dos dados;
- "Proteção de dados desde a conceção" - processos para planear novos projetos que envolvam tratamento de dados;
- "Notificação de violações de segurança" - comunicação de qualquer violação que ponha em risco a proteção dos dados dos titulares;
- "Coimas" - critérios que regulam a atribuição de coimas, que poderão ir até vinte milhões de euros ou quatro vezes o volume de negócios anual.

2.2 DADOS PESSOAIS

Toda a informação relativa a uma pessoa viva, identificada ou identificável, é considerada dado pessoal. Também constituem dados pessoais o conjunto de informações distintas que podem levar à identificação de uma determinada pessoa.

Dados pessoais que tenham sido descaracterizados, codificados ou colocados em pseudónimo, mas que possam ser utilizados para identificar uma pessoa, continuam a ser dados pessoais e são abrangidos pelo âmbito de aplicação do RGPD.

O RGPD protege os dados pessoais independentemente da tecnologia utilizada para o tratamento dos mesmos. É neutro em termos tecnológicos e aplica-se tanto ao tratamento automatizado como ao tratamento manual, desde que os dados sejam organizados de acordo com critérios pré-definidos. Também é irrelevante o modo como os dados são armazenados, num sistema informático, através de videovigilância, ou em papel. Em todos estes casos, os dados pessoais estão sujeitos aos requisitos de proteção previstos no RGPD.

De seguida, apresentam-se alguns exemplos de dados considerados pessoais:

- Nome
- Número de identificação
  - Bilhete de Identidade, Cartão do Cidadão
  - Carta de Condução
  - Passaporte
- Endereço de identificação e localização
  - Morada
  - Endereço de Email
  - Pagina *Web*

  - Redes Sociais
  - etc.
- Biométricos
  - Altura, peso, conotações físicas diversas
  - Genética
- Saúde
  - Síndromes, doenças
  - Desempenho físico ou mental
  - Dados de diagnósticos como pressão arterial ou ECG
- Económicos
- Culturais
- Sociais
- Políticos

2.3 DADOS SENSÍVEIS

O regulamento identifica uma série de dados pessoais que podem ser classificados como sensíveis, pois o seu tratamento pode implicar um risco significativo para com os direitos e liberdades fundamentais.

Com efeito, os seguintes dados pessoais são considerados «sensíveis» e estão sujeitos a condições de tratamento específicas:

- Dados pessoais que revelem a origem racial ou étnica, opiniões políticas e convicções religiosas ou filosóficas;
- Filiação sindical;
- Dados genéticos, dados biométricos tratados simplesmente para identificar um ser humano;
- Dados relacionados com a saúde;
- Dados relativos à vida sexual ou orientação sexual da pessoa.

2.4 TRATAMENTO

Para o RGPD, o tratamento constitui o conjunto de operações realizadas aos dados pessoais. As operações de tratamento incluem:

- Recolha;
- Registo;
- Estruturação;
- Conservação;
- Adaptação ou alteração;
- Recuperação;
- Consulta;
- Utilização;
- Divulgação por transmissão;
- Difusão ou qualquer outra forma de disponibilização;
- Comparação;
- Interconexão;
- Limitação;
- Apagamento ou destruição.

Todas estas formas de tratamento de dados podem ser processadas por meios automatizados ou por meios não automatizados, estando abrangidas pelos princípios do Regulamento Geral de Proteção de Dados. Contudo, o artigo 4º, nº 2 apenas enumera estas hipóteses para tratamento de dados, não as considerando únicas, isto é, podem existir outras operações que se podem enquadrar no conceito, mas que não estão contidas neste conjunto.

2.5 RESPONSÁVEL PELA PROTEÇÃO DE DADOS

O responsável de proteção de dados (DPO) assegura, de forma independente, que uma organização aplica as leis que protegem os dados pessoais dos indivíduos. A designação, o cargo e as funções de um DPO dentro de uma organização são descritos no artigo 37º do RGPD.

As principais responsabilidades do DPO incluem assegurar, através de orientações e formações, que as organizações a seu cargo estejam preparadas para o cumprimento de todas as obrigações relacionadas com o RGPD. Além disso, deve realizar auditorias para assegurar a execução das medidas e abordar questões potenciais de forma proativa, agindo como elo de ligação entre as suas organizações e o público em relação a todas as questões de privacidade de dados.

Dependendo do tamanho da organização, a existência de um DPO pode ou não ser de caráter obrigatório. Porém, é aconselhável a nomeação de um responsável de proteção de dados interno que tenha capacidades e conhecimentos para desempenhar as funções dentro da empresa ou organização. Nos casos de uma não nomeação, é obrigatória a subcontratação de um DPO externo que assegure a execução de auditorias.

Com efeito, as funções de um responsável de proteção de dados são:

- Informar e aconselhar em matéria de proteção de dados pessoais;
- Controlar a conformidade do tratamento de dados pessoais com o RGPD e com outras disposições de proteção de dados da União Europeia (UE);
- Prestar aconselhamento no que respeita à avaliação de impacto sobre proteção de dados e controlo da sua realização;
- Cooperar e funcionar como ponto de ligação com a autoridade de controlo.

Quando há necessidade de recorrer a um subcontratante para efetuar o tratamento de dados pessoais, há que ter em conta as seguintes regras:

- O recurso a serviços de tratamento de dados pessoais deve ser assegurado por garantias de execução de medidas técnicas e organizativas adequadas ao RGPD;
- O subcontratado não pode recorrer a outro subcontratante sem autorização prévia e específica por escrito do representante de proteção de dados;
- Um contrato de subcontratação terá de incluir o objeto e duração do tratamento de dados, natureza e finalidade, tipo de dados e categorias dos titulares, assim como as obrigações e os direitos do responsável.

2.6 CONSENTIMENTO

Esta nova legislação traz uma maior proteção aos titulares dos dados pessoais, garantindo ainda mais controlo sobre a informação pessoal. Agora, as comunicações de *marketing* vão exigir um consentimento livre, informado e explícito, aumentando assim a segurança no tratamento de dados. Desta forma, é necessário que se verifique sempre o consentimento antes de utilizar os dados dos clientes.

2.7 DIREITOS

Antes da aplicação do RGPD, já havia preceitos similares. No entanto, depois de este entrar em vigor, as regras tornaram-se um pouco mais claras e definidas. Por exemplo, o facto de um indivíduo dar consentimento não significa que este seja irrevogável ou vitalício.

Há, ainda, uma questão que naturalmente se coloca: quem é o titular dos dados? Segundo o RGPD, o titular dos dados pessoais é qualquer pessoa particular a quem os dados façam referência. Estes passam a assumir os seguintes direitos:

- Direito de acesso - o titular pode aceder em qualquer momento à sua informação;
- Direito a ser esquecido - o titular pode pedir para deixar de estar incluído numa base de dados, no que diz respeito a ações de comunicação. Contudo, note-se, no que diz respeito a faturas, o RGPD não se sobrepõe a uma série de outras leis, incluindo aquelas referentes a matéria fiscal;
- Direito à portabilidade dos dados - a pedido do titular ou de entidade que tenha obtido consentimento, terá de ser possível fornecer os dados pessoais que lhe digam respeito, num formato de uso corrente ou leitura automática;
- Direito de oposição - o titular de dados pode opor-se a um determinado tratamento de dados.

2.8 CERTIFICAÇÃO

Com a entrada em vigor do RGPD, levantou-se a necessidade de auditoria e de certificação de pequenas, médias e grandes organizações a fim de verificar o cumprimento da medidadas instauradas pelo regulamento. Para cumprir este propósito, foram criadas entidades reguladoras cujo objetivo é analizar toda a documentação entregue e verificar se esta cumpre os requisitos exigidos, procedendo-se, depois, à certificação.

Este processo é assegurado pelo DPO, que, ao auditar uma entidade, elabora um relatório de conformidade que é anexado como evidência nos restantes documentos exigidos pela entidade reguladora para a certificação. (Ver secção 2.5, funções de um DPO).

2.9 SEGURANÇA INFORMÁTICA

2.9.1 *Tokens*

Uma forma de segurança é a utilização de *token* de autenticação nos pedidos HTTP. Este é simplesmente uma *string* de caracteres que, caso cliente e servidor estejam sob HTTPS, permite que somente o servidor que conhece o *token* possa validar o conteúdo do mesmo e assim confirmar a autenticidade do pedido. O *token* não é criptografia, mas uma forma que permite, através de uma estrutura comprovada, impedir os acessos.

Um *token* é gerado com um data de expiração e, quando esta termina, é necessário gerar outro para que se possa continuar a poder fazer pedidos à API.

O processo de autenticação por meio do *token* é feito através do envio do mesmo no cabeçalho http, identificado como *Authorization*. Neste processo, ele é identificado como *Bearer token*.

A figura 2 mostra uma imagem do *software POSTMAN* que indica este sistema de *token*.

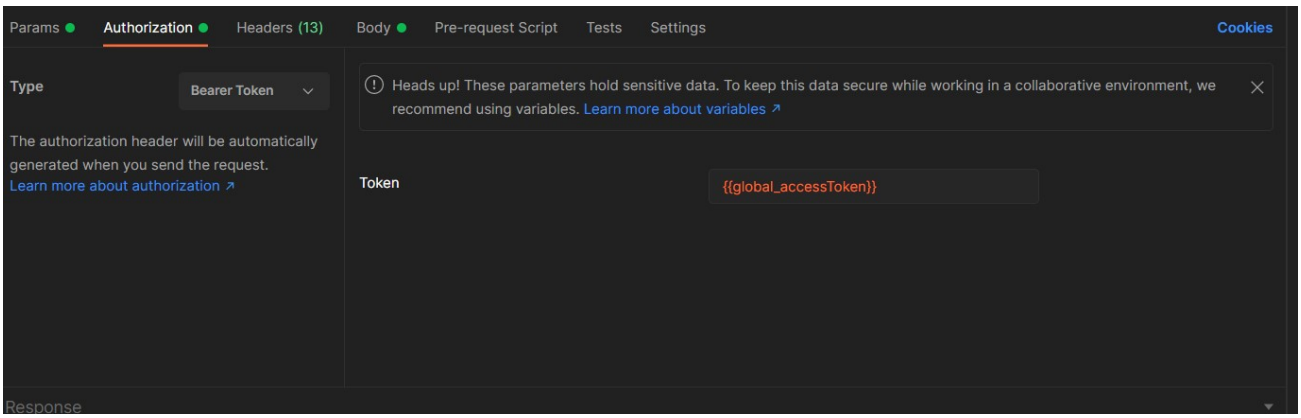


Figura 2: Representação do *Bearer Token*.

2.9.2 *Rivest-Shamir-Adleman (RSA)*

Desde há muitos anos que o Homem utiliza a criptografia para transmitir mensagens secretas. Esta ideia era bastante simples: uma chave que codificasse conteúdo da mensagem e só quem conhecesse a mesma é que conseguiria tornar a mensagem novamente legível.

O algoritmo *RSA* é um dos mais seguros que existe na atualidade. É considerado um algoritmo que, através da utilização de chaves públicas, privadas e certificados, possibilita a assinatura digital e encriptação de dados.

Num sistema de criptografia, a chave de encriptação é pública, sendo distinta da chave de desencriptação, que é mantida em segredo e chamada de chave privada. Porém, o grande problema desta abordagem é que tanto o emissor quanto o recetor têm de conhecer a chave de criptografia.

2.10 TECNOLOGIAS

2.10.1 *Application Programming Interface (API)*

Uma *API* é uma interface que permite que uma parte de um *software* comunique com outro. Podemos, então, considerar que uma *API* é como que um documento padrão que segue regras que, atualmente, todas as linguagens de programação são capazes de interpretar.

Dentro deste conceito de *API*, encontra-se outro, o de *REST API*. Uma *REST API (Representational State Transfer)*, ou seja, Transferência Representativa do Estado, é um estilo de arquitetura de *software* padronizado que permite comunicações no sentido cliente-servidor e vice-versa.

A figura 3 representa o pedido *REST* na forma de comunicação entre cliente e servidor.

Figura 3: Funcionamento de um pedido *REST API*

Portanto, quando se fala em *REST API*, significa utilizar uma *API* para efetuar pedidos a aplicações *backend*, de modo que essa comunicação seja feita com os padrões definidos pelo estilo de arquitetura *REST*.

Um serviço de *REST API* é composto por dois blocos:

- **Pedido**, que é enviado pelo cliente ao servidor.
- **Resposta**, que é enviada de volta ao cliente.

Uma *API* utiliza requisições HTTP, que são responsáveis pelas operações básicas necessárias para a manipulação dos dados. Este tipo de pedido é classificado de *CRUD*, que é o mesmo que "*Create, Read, Update, and Delete*"(Criar, Ler, Atualizar e Apagar), sendo estas algumas das principais restrições que permite a comunicação ao servidor.

O *CRUD* implica:

- *POST*: criar dados no servidor;
- *GET*: ler dados no *host (URL)*;
- *DELETE*: excluir as informações;
- *PUT*: atualizar os registos

A resposta aos pedidos efetuados utilizando o protocolo *HTTP* têm uma representação no formato *JSON*, que é capaz de ser interpretado em todas as linguagens de programação modernas.

Exemplificando com um caso real: imagine que trabalha numa loja de sapatos e necessita de construir uma aplicação *web* que mostre os vários tipos de sapatos que pretende vender. Esta aplicação vai fazer um pedido *REST API* como este "http://sapatariaA.com/api/sandalias"a um servidor. Esta operação é do tipo *"GET"* que devolve à aplicação *web* uma listagem em formato *JSON* das sandálias existentes em *stock*.

Em baixo encontra-se representada uma possível resposta para o pedido *"GET"*, tendo em consideração o exemplo anterior:

```
[
    {
        id: 1234,
        descricao: 'sandaliaA',
        amount: 39,
        image: http://sapatariaA.com/api/image?hsdhsahdoisahdoi
    },
    {
        id: 1235,
```

```
        descricao: 'sandaliaB',
        amount: 24,
        image: http://sapatariaA.com/api/image?yoiwqyriwqrrewr
    },
    {
        id: 1236,
        descricao: 'sandaliaC',
        amount: 15,
        image: http://sapatariaA.com/api/image?cnlsandclkdsmoe
    },
    {
        id: 1237,
        descricao: 'sandaliaD',
        amount: 49,
        image: http://sapatariaA.com/api/image?lpwrouoiecidds
    },
    {
        id: 1238,
        descricao: 'sandaliaF',
        amount: 39,
        image: http://sapatariaA.com/api/image?uoiwhdinineiofhoe
    }
]
```

2.10.2 *Nodejs*

Muitos consideram *Nodejs* uma linguagem de programação, todavia não é. Já há varias décadas que a linguagem *Javascript* é utilizada no desenvolvimento de aplicações *web*. Podemos então definir o *Nodejs* como um ambiente de execução *Javascript server-side*, o que significa que com o *Nodejs* é possível criar aplicações *Javascript* para serem executadas como uma aplicação *standalone* numa máquina, não sendo necessária a utilização de um *browser* para a executar.

Apesar de recente, o *Nodejs* já está presente num número grande empresas no mercado tecnológico, como *Netflix*, *Uber* e *LinkedIn*. O principal motivo para este facto é a alta capacidade de ser escalável. Além disso, é composto por uma arquitetura flexível e de baixo custo, caracteríscas que tornam o *Nodejs* uma boa escolha para implementação de microsserviços e componentes da arquitetura *Serverless*. Inclusive, os principais fornecedores de produtos e serviços *Cloud* já têm suporte para desenvolvimento de soluções escaláveis utilizando o *Nodejs*.

A principal característica que diferencia o *Nodejs* de outras tecnologias, como *PHP*, *Java* ou *C#*, é o facto de ser uma execução *single-thread*, o que significa que apenas uma *thread* é responsável por executar o código *Javascript* da aplicação, enquanto que nas outras linguagens a execução é *multi-thread*.

2.10.3 *Base de dados*

Relacional

Uma base de dados relacional é um repositório digital baseado no conceito de relacionamento de dados. Por outras palavras, cada linha de uma tabela é composta por atributos de dados e cada registo geralmente tem um valor atribuído, permitindo a criação de relações entre dados.

Este modelo traz vantagens. Por exemplo, se os dados estão inseridos em tabelas bem definidas, pode-se estabelecer um relação entre elas. Com efeito, existem três tipos de relações, abaixo explicadas: um para um, um para muitos e muitos para muitos.

- **Um para Um**: as tabelas podem ter apenas um registo em cada lado do relacionamento.
- **Um para Muitos**: a tabela principal contém apenas um registo que pode ser relacionado. Contudo, a outra já contém vários registos desse relacionamento.
- **Muitos para Muitos**: ambas a tabelas podem ter mais do que um registo a relacionar-se entre si.

Não Relacional

Uma base de dados não relacional não segue o modelo relacional fornecido pelos sistemas tradicionais de base de dados relacionais. Em vez disso, é composta por um sistema de registo *Key-Value*, ou seja, os dados são representados como uma coleção de pares. Estas bases de dados não relacionais são utilizadas sempre que os dados são consultados por parâmetros precisos e é necessário recuperar os dados rapidamente.

2.10.4 *Postman*

O *Postman* é uma aplicação que dá suporte à documentação das requisições feitas pela *API*. Ele possui ambiente para a documentação, execução de testes de *API* e requisições em geral, permitindo trabalhar com *API*. Além disso, permite guardar as mesmas para uso posterior e consegue analisar as respostas enviadas pela *API*.

A figura 4 mostra a interface do *Postman*.

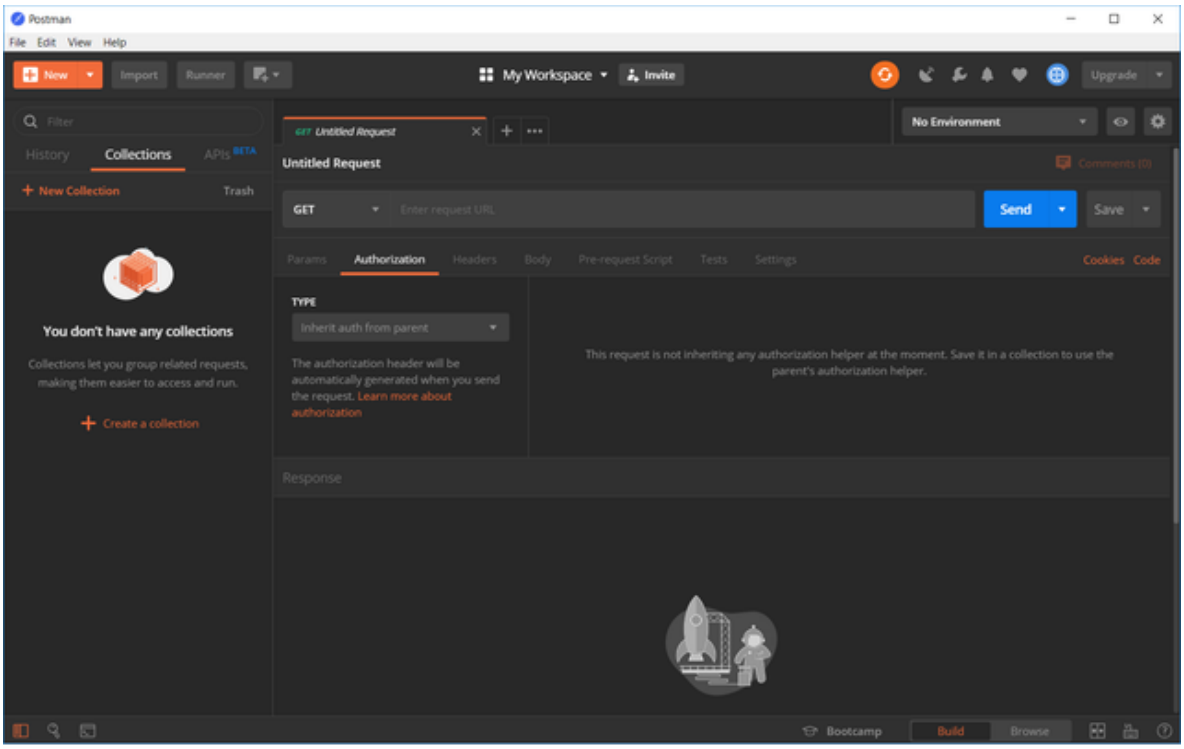


Figura 4: *Postman* interface

2.10.5 *Testes - JEST*

A utilização da computação tem vindo a aumentar de dia para dia, tornando-se cada vez mais difícil a verificação da qualidade dos*softwares* com a utilização de apenas testes manuais. Os *softwares* estão a ficar cada vez mais complexos e robustos, o que favorece a criação de erros e dificulta a sua deteção, de tal forma que é cada vez mais fácil que erros passem despercebidos para a versão de produção.

Para prevenir a formação de erros e a sua não deteção no processo de desenvolvimento de aplicações, começaram a ser elaborados testes unitários, testes *end-to-end* e testes automáticos.

Os testes unitários são pequenos excertos de código que permitem validar, através de *mocks*, a funcionalidade de um processo. Por sua vez, os testes *end-to-end* são mais complexos, permitindo testar um processo desde o pedido inicial (*API*) até à base de dados e respetiva resposta. Por último, os testes automáticos são *scripts* que executam os testes *end-to-end* de forma a simular um utilizador ou um sistema. Estes são executados no *deploy* da solução com o intuito de serem um camada de confirmação neste processo.

A utilização e implementação deste tipo de testes trouxe vantagens, nomeadamente:

- Menos erros: a elaboração de testes permite um reforço na consistência dos sistemas.
- Satisfação dos clientes: se o produto apresenta menos problemas, aumenta o grau de satisfação dos clientes.
- Documentação: os testes permitem obter uma série de evidências para relatório de qualidade e certificações.

Jest é um bom exemplo de ferramenta que permite desenvolver estes testes numa aplicação com base *Javascript*, trazendo todas as vantagem acima referidas no momento da criação de uma solução.

3

PROPOSTA

Apresentado o enquadramento e a motivação que impulsionaram o desenvolvimento deste projeto (capítulos 1 e 2) e depois de avaliar a viabilidade de desenvolvimento de uma solução de apoio à gestão do RGPD com o intuito de auxiliar o DPO a cumprir as suas tarefas, descritas no capitulo 2 (secções 2.5 e 2.8), identifica-se, então, o problema alvo de estudo deste trabalho.

Esta solução tem como objetivo principal o desenvolvimento centrado na execução da gestão de métricas, entidades, documentos e medidas, identificadas pelo DPO, e que acabaram por servir para auxiliar o cumprimento do RGPD.

A figura 5 representa o diagrama de contexto. Este demonstra o sistema a desenvolver e os processos envolventes. As características do sistema são:

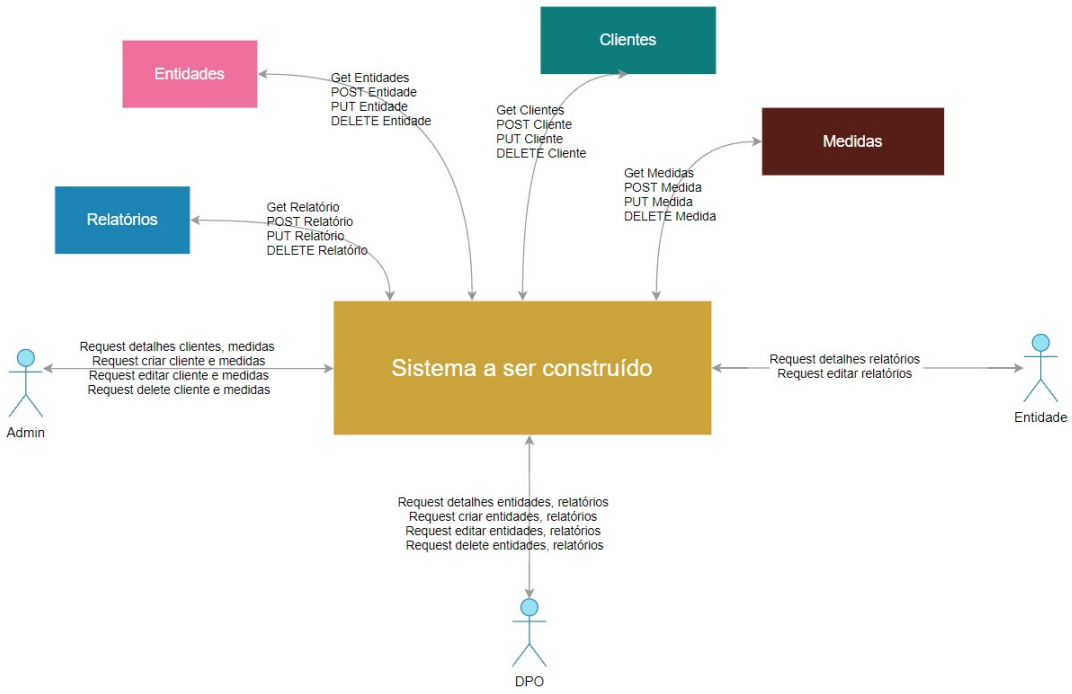


Figura 5: Diagrama de Contexto

A figura 6 esquematiza o processo por trás da ideia que levou ao desenvolvimento. Como se pode ver, o DPO pretende registar medidas e verificar se são aplicáveis. Nesse caso, é necessário verificar se foram implementadas e gerar o relatório final.

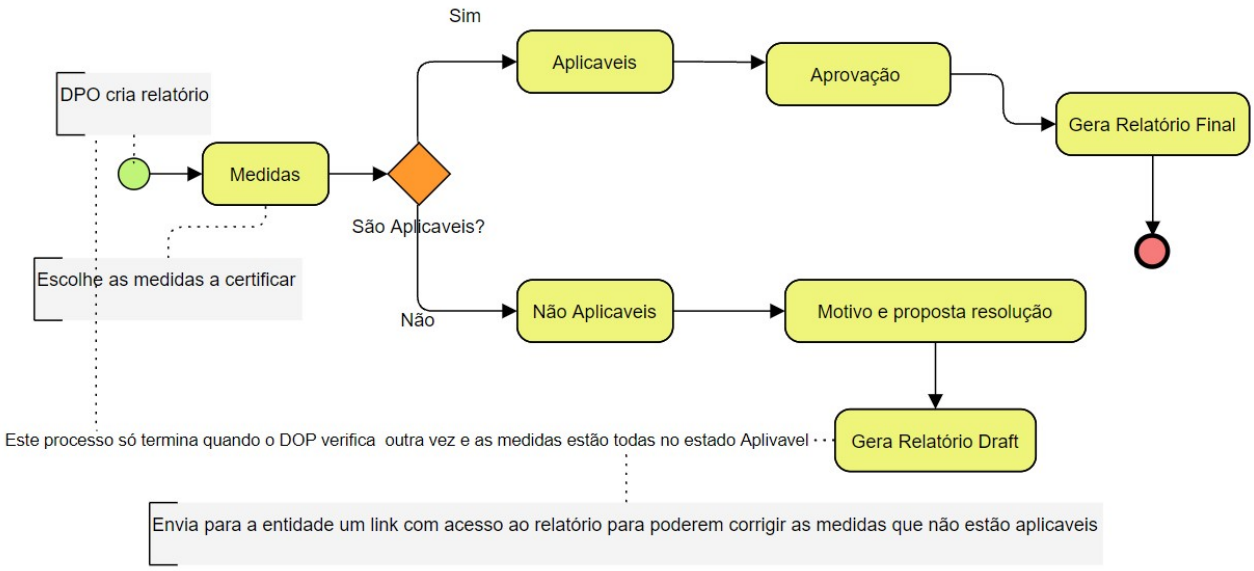


Figura 6: Diagrama de Processo - Criação de relatório de apoio à certificação

3.1 DESAFIOS / OBRIGAÇÕES

Antes de proceder ao desenvolvimento de uma solução tecnológica deste tipo, é necessário analisar o documento de medidas e regras, a fim de perceber os pontos que devem ser considerados e respeitados. De seguida, serão apresentados os principais pontos.

3.1.1 *Proteção de dados*

A proteção de dados indica que os *softwares* deverão conter as informações pessoais dos indivíduos com alguma forma de encriptação para que a informação esteja inacessível a utilizadores não autorizados.

A encriptação é, geralmente, aplicada de duas formas distintas:

- Armazenamento encriptado – é normalmente utilizada para encriptar um disco ou um dispositivo na sua totalidade.
- Conteúdo encriptado – também conhecido por encriptação granular – os ficheiros ou o texto são encriptados ao nível da aplicação.

Um dos maiores problemas ao nível da utilização de sistemas informáticos é a forma como os utilizadores pretendem partilhar as informações encriptadas. Existem dois métodos tradicionais:

- **Passwords partilhadas**, que podem ser simples de lembrar e inseguras ou impossíveis de lembrar e seguras. No entanto, estas últimas necessitam de ser escritas em algum lado;
- **Encriptação de chaves públicas**, que funcionam bem em grupos mais pequenos, mas que se tornam complexas ou problemáticas com equipas dinâmicas;

A figura 7 exemplifica a ideia de chave pública e chave privada, bem como a utilização da pública para encriptar e a privada para desencriptar. Ver capitulo 2, secção 2.9.2.

Figura 7: Chave Pública e Privada

3.1.2 *Política de esquecimento*

De acordo com o RGPD, todos os cidadãos têm o direito de pedir, junto dos responsáveis pela recolha e tratamento de dados, que apaguem as suas informações pessoais. Quer isto dizer que qualquer pessoa pode exigir a um empresa ou organização que os seus dados pessoais sejam apagados em definitivo, sendo que esta é obrigada a fazê-lo sem demoras injustificadas.

Este direito a ser esquecido deve garantir que os dados pessoais sejam apagados permanentemente de qualquer base de dados, garantindo também que desaparece a sua divulgação *online*. No entanto, este não é um direito absoluto e existem algumas condicionantes para que possa ser exercido e executado.

Se os dados pessoais que a empresa recolheu sobre o cidadão deixaram de ser necessários para a finalidade que motivou a sua recolha ou tratamento, eles devem ser, de imediato, apagados. Exemplificando, se já não tem ligações comerciais com uma empresa com quem teve contrato de fornecimento de energia, pode pedir-lhe que apague os seus dados pessoais que foram utilizados para manter a relação comercial agora terminada, com exceção dos dados que a empresa é obrigada a manter por imposição legal.

O direito a ser esquecido pode ser invocado por qualquer titular de dados pessoais. Contudo, a aplicação deste direito apenas pode "esquecer" dados pessoais que não sejam

necessários para cumprir com as conformidades legais da legislação portuguesa. Por exemplo, dados que ainda sejam necessários para efeitos de faturação, para o pagamento de dívidas ou para cumprir com as obrigações e contribuições ao Estado. Enfim, as empresas terão de cumprir os pedidos dos cidadãos e apagar os seu dados, exceto se esses dados forem legalmente essenciais para que a empresa cumpra a legislação nacional.

3.1.3 *Medidas*

Todas as entidades têm de cumprir uma série de medidas de forma a garantir o cumprimento do regulamento. Estas podem ser divididas em três grupos dentro do processo de conformidade:

- **Medidas organizacionais**
  - Requisitos da estrutura organizacional;
  - Processos, procedimentos e políticas;
  - Consciência para a Segurança dos Dados.
- **Medidas técnicas**
  - Responsabilidade;
  - Violação de dados;
  - Assegurar os direitos dos titulares de dados;
  - Comunicar informações de privacidade (consentimentos, avisos de processamento
  - Segurança dos dados (integridade e confidencialidade).
- **Medidas ao nível dos dados**
  - Documentação dos dados;
  - Base jurídica;
  - Auditoria.

3.2 REQUISITOS

Neste secção, será feita a apresentação dos principais requisitos funcionais que foram alvo de levantamento e análise.

3.2.1 *Principais Use Cases*

Na figura 8, são apresentadas as principais funcionalidades que um DPO pode realizar no sistema. Mais concretamente, o DPO pode efetuar a gestão de entidades (empresas que ele pretende certificar ao nível do RGP), bem como a gestão dos relatórios de conformidade que serviram como suporte para a certificação de entidade.

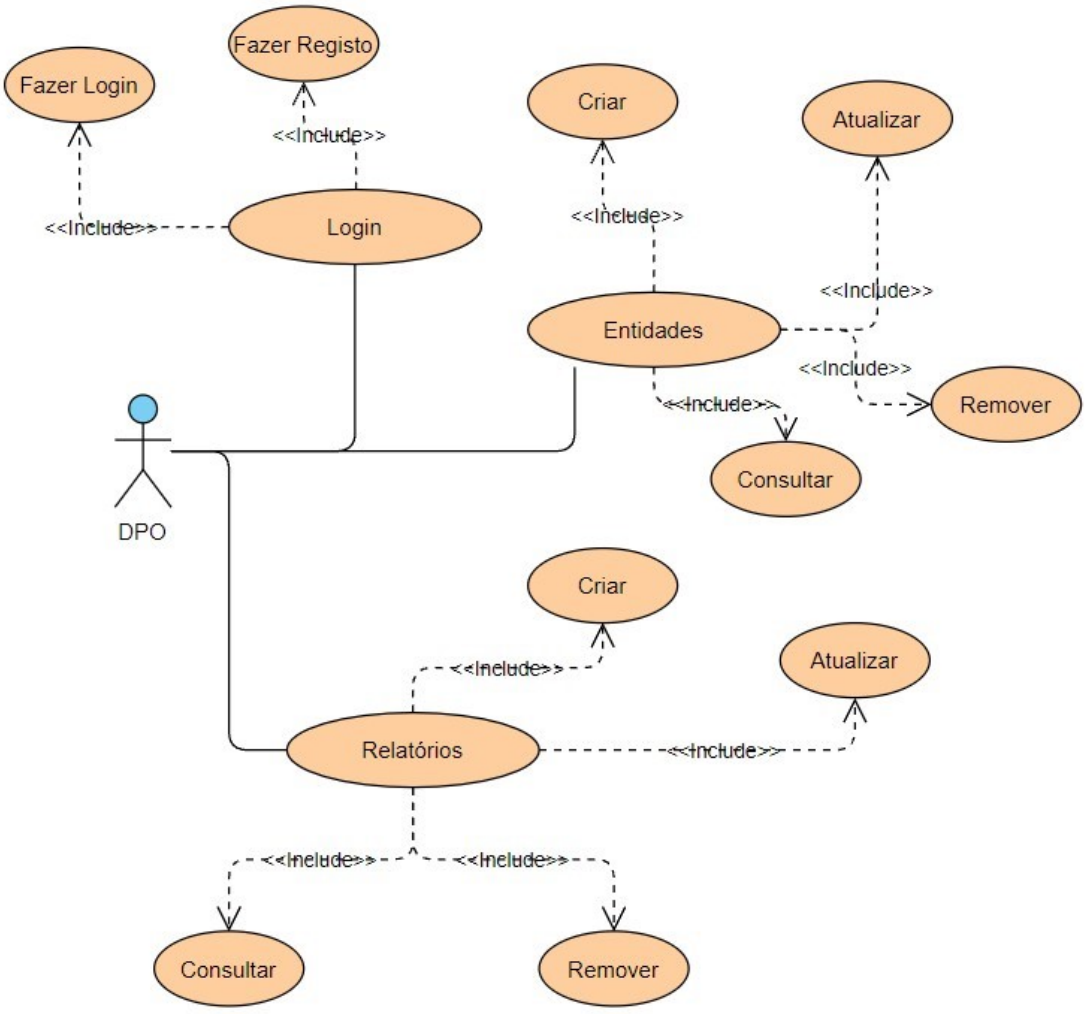


Figura 8: Caso de Uso do DPO

Por sua vez, a figura 9 representa a parte administrativa do sistema e as suas principais funcionalidades. Um administrador do sistema pode gerir as medidas, categorias e licenças que servirão como base dos relatórios criados pelo DPO . Além disso, o administrador tem ainda a possibilidade de gestão dos clientes, que tanto podem ser o DPO como qualquer responsável pela certificação do RGPD.

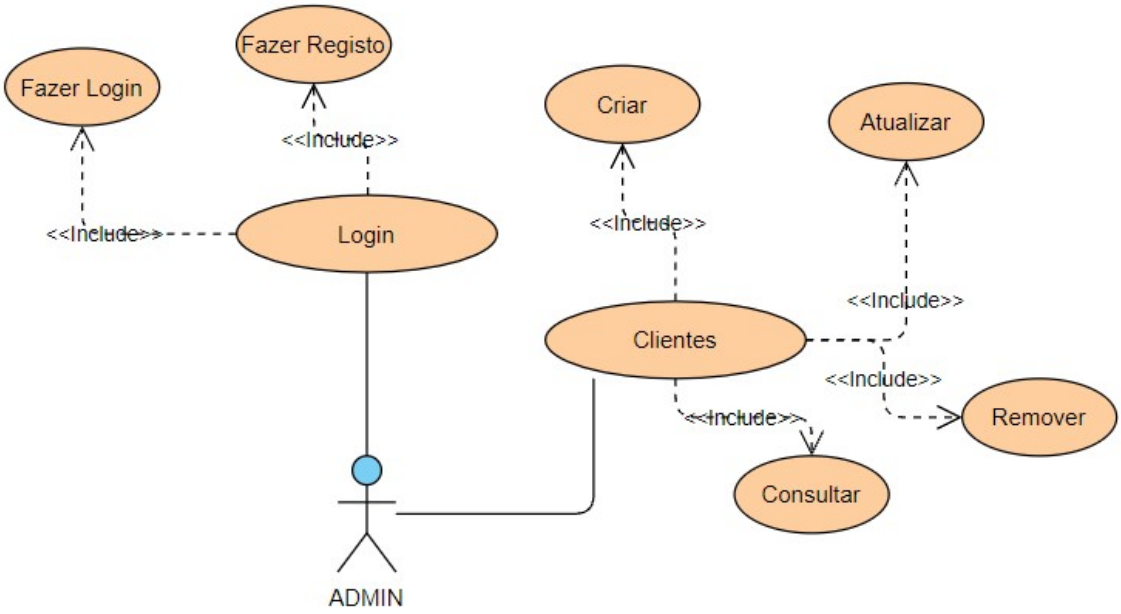


Figura 9: Caso de Uso do Admin

3.2.2 *Apresentação de Requisitos*

Abaixo, apresentam-se os principais requisitos que foram concebidos pelo negócio durante a passagem de conhecimento à equipa de desenvolvimento.

Registo na Solução

Como *DPO* **quero** fazer um registo na plataforma **para** poder iniciar sessão.

Fundamentação: Um *DPO* terá que fazer um registo com o seu nome, *email* e uma *password* para ter acesso a uma sessão de utilizador registado.

Criação de Entidades

Como *DPO* **quero** registar entidades.

Fundamentação: Um *DPO* poderei registar as entidades a meu cargo na solução.

Criação de Medidas

Como *DPO* **quero** registar medidas.

Fundamentação: Um *DPO* poderei registar as medidas a ser aplicadas pelas entidades na solução.

Alocação de Medidas

Como *DPO* **quero** poder associar medidas e entidades.

Fundamentação: Um *DPO* poderá escolher medidas de uma lista previamente ciada e associa-las a uma ou mais entidades.

Criação de Relatórios

Como *DPO* **quero** criar um relatório de medidas.

Fundamentação: Um *DPO* poderá gerar uma relatório final de comprimento ou não por parte das entidades em relação com as medidas proposta a cumprir.

Consulta de Relatórios

Como *DPO* **quero** consultar os relatórios.

Fundamentação: Um *DPO* poderá consultar um histórico de relativos das medidas executadas.

4

DESENVOLVIMENTO

Como já foi explicado anteriormente, o desenvolvimento deste projeto foi feito ao nível do *backend*. Nesta secção, serão expostas partes do processo de desenvolvimento e será feita a especificação da solução proposta. Serão apresentadas algumas das principais *API's*, modelos de base dados e preocupações de segurança, assim como as principais estruturas de código e os seus respetivos exemplos.

Na figura 10, é possível observar a estrutura de divisões de desenvolvimento do sistema, sendo este composto por *frontend*, *backend* e base de dados.

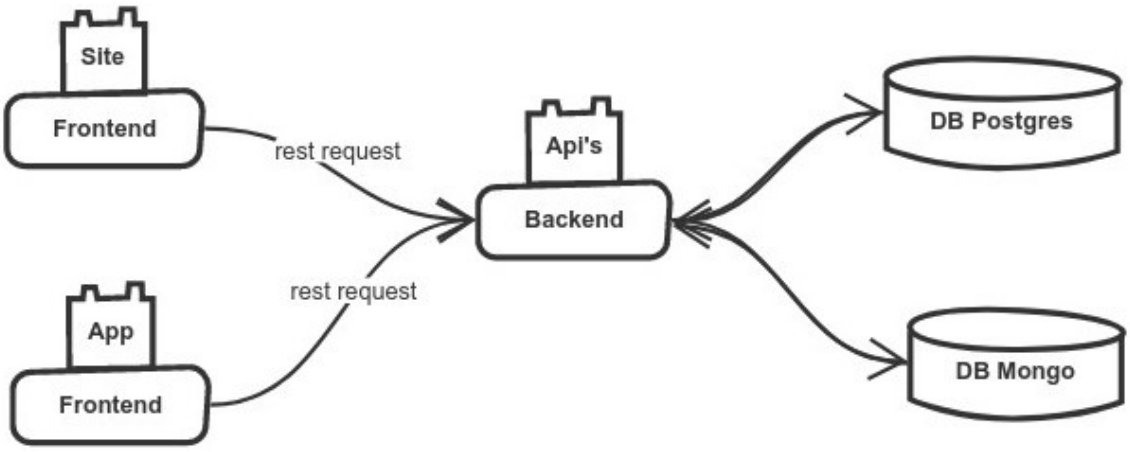


Figura 10: Representação da divisão da solução

4.1 ESPECIFICAÇÃO

Durante o processo de planeamento e especificação da solução, partiu-se dos requisitos propostos pelo negócio e dividiram-se em funcionalidades, que serão apresentadas abaixo como representações e diagramas. Estes foram desenvolvidos a baixo nível, procurando demonstrar uma visão mais geral do projeto.

4.1.1 *API's e Funcionalidades*

Com o apoio de ficheiros *YAML*, foram especificadas as *API's* que o sistema vai conter. A seguir, estão apresentadas algumas das principais.

API's de Customer

- Login

  A figura 11 mostra o fluxo de baixo nível da estrutura da funcionalidade de pedido de autenticação que será feito ao *backend*. Como se pode observar, foram pensadas e implementadas verificações que têm como objetivo verificar se os dados recebidos no pedido estão corretos e se são válidos.

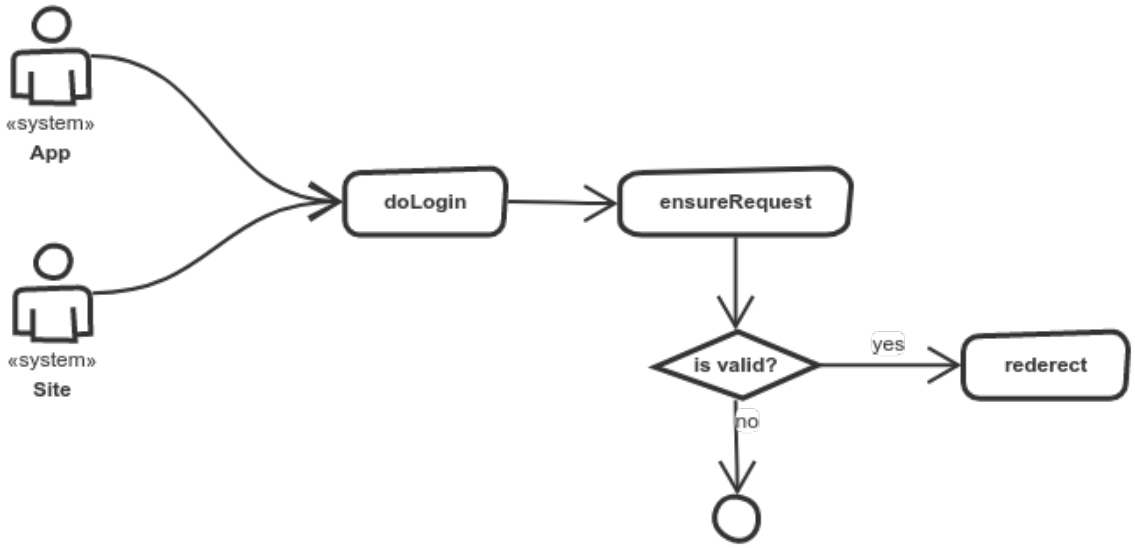


Figura 11: Estrutura da funcionalidade de *Login*

Também é possível verificar a estrutura da *API*, bem com o pedido e a resposta na figura 12. O pedido *POST* identificado permite efetuar a autenticação no sistema.

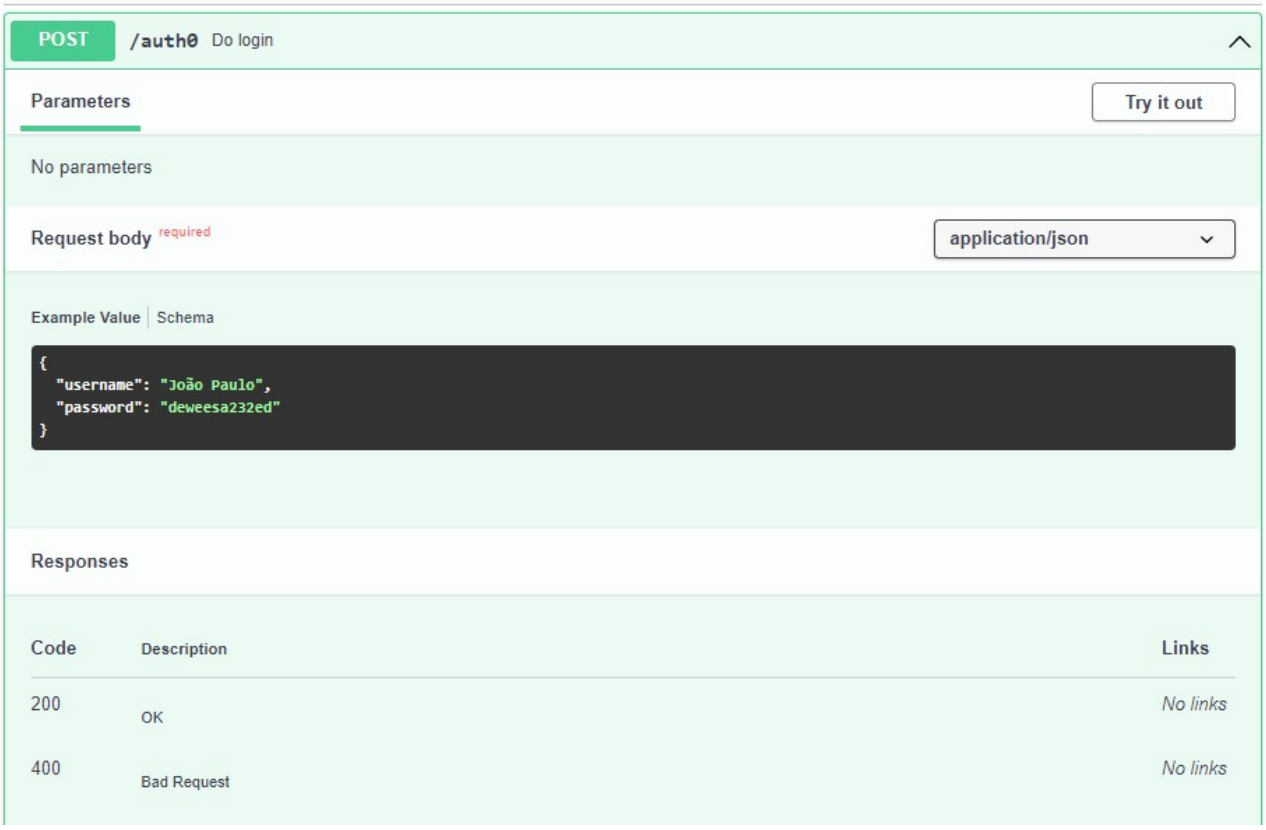


Figura 12: Estrutura de rotas *Login*

- *Customer*

  No que toca à gestão de clientes, o sistema vai permitir criar, atualizar, remover e consultar a informação. Os dados dos cliente serão encriptados antes de serem guardados em base de dados.

  A figura 13 esclarece os principais fluxos dos *customer* (pedido, verificações e finalização do processo).

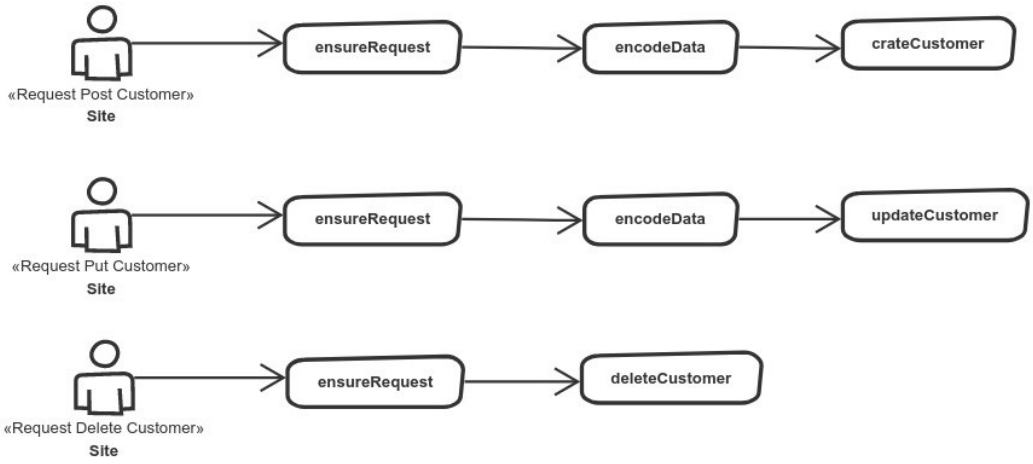


Figura 13: Processos *Customer*

Para além dos dados como nome, *email* e *password* que os clientes poderão armazenar, também terão associados dados como departamentos e grupo. Estes últimos serão *API's* que permitem alterações como *POST*, *GET*, *PUT* e *DELETE*, com o fim de se poder alimentar a base de dados com a informação necessária.

A figura 14 representa as principais *API's* dos *customer*, que são complementares às da figura 13. Duas destas são do tipo *GET*, uma para devolver todos os *customer* e outra apenas para pedir um *customer*. No que toca às restantes, uma é do tipo *POST* para criar um novo *customer* e, por último, a outra é um *PUT* para atualizar os dados.

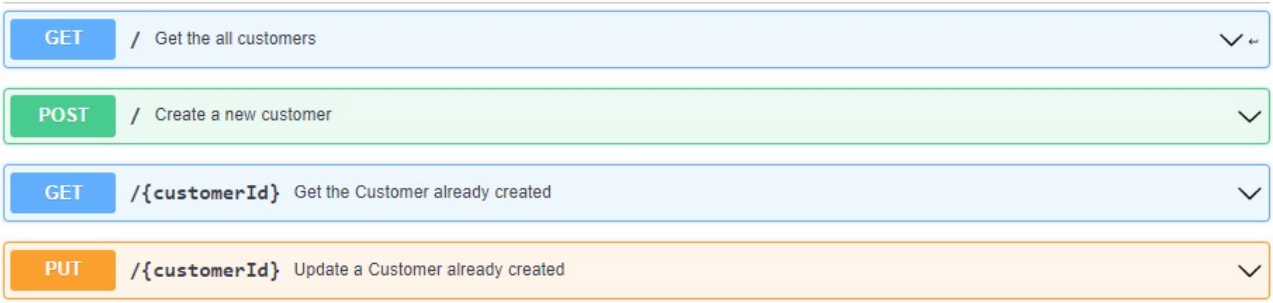


Figura 14: *API's* de *customer*

- Entidades

  Relativamente à gestão de entidades, o sistema vai permitir criar, atualizar, remover e consultar. Estas serão introduzidas pelos DPO's com a finalidade de ligar os relatórios gerados com uma instituição, o que pode ser observado na figura 15.

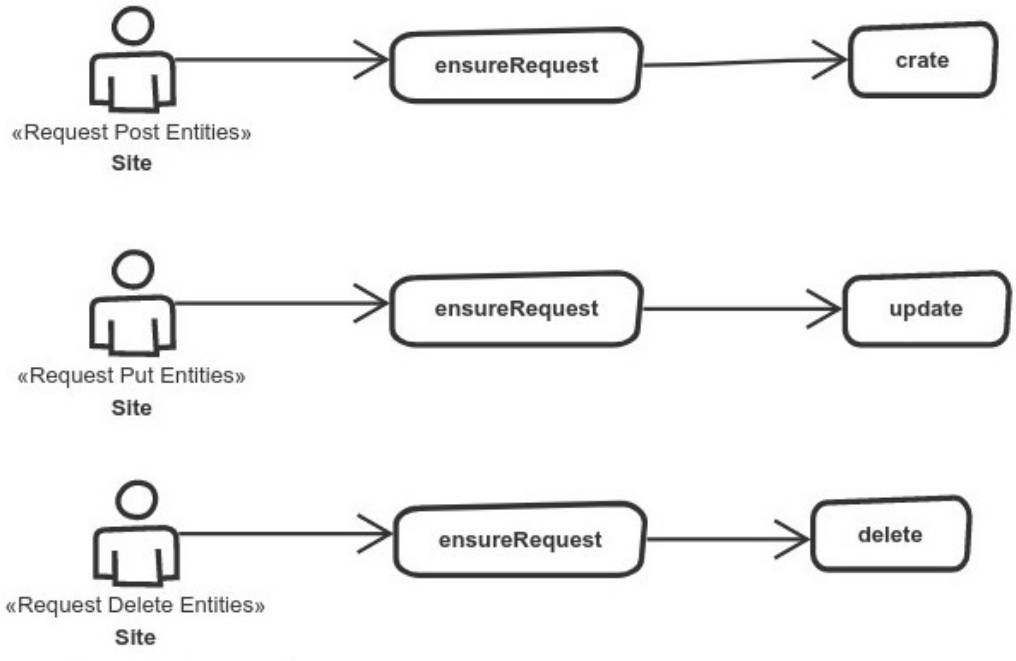


Figura 15: Estrutura das funcionalidades de Entidades

A figura 16 mostra as *API's* possíveis para as entidades, das quais duas são do tipo *GET*, uma para devolver todos as entidades e outra só para pedir uma entidade, outra é do tipo *POST* para criar e a última um *PUT* para atualizar os dados.

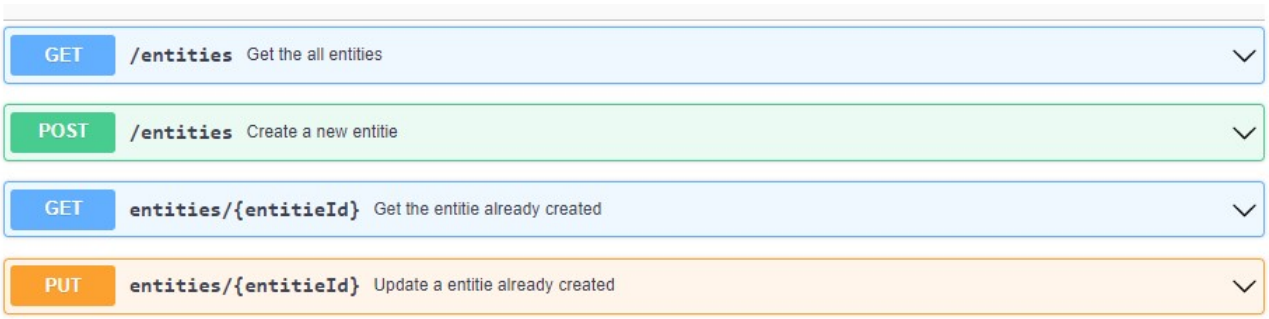


Figura 16: *API's* de entidades

Medidas

O diagrama representado na figura 17 mostra a estrutura da funcionalidade das medidas. Nestes fluxos estão os principais pedidos, verificações e fim dos processos.

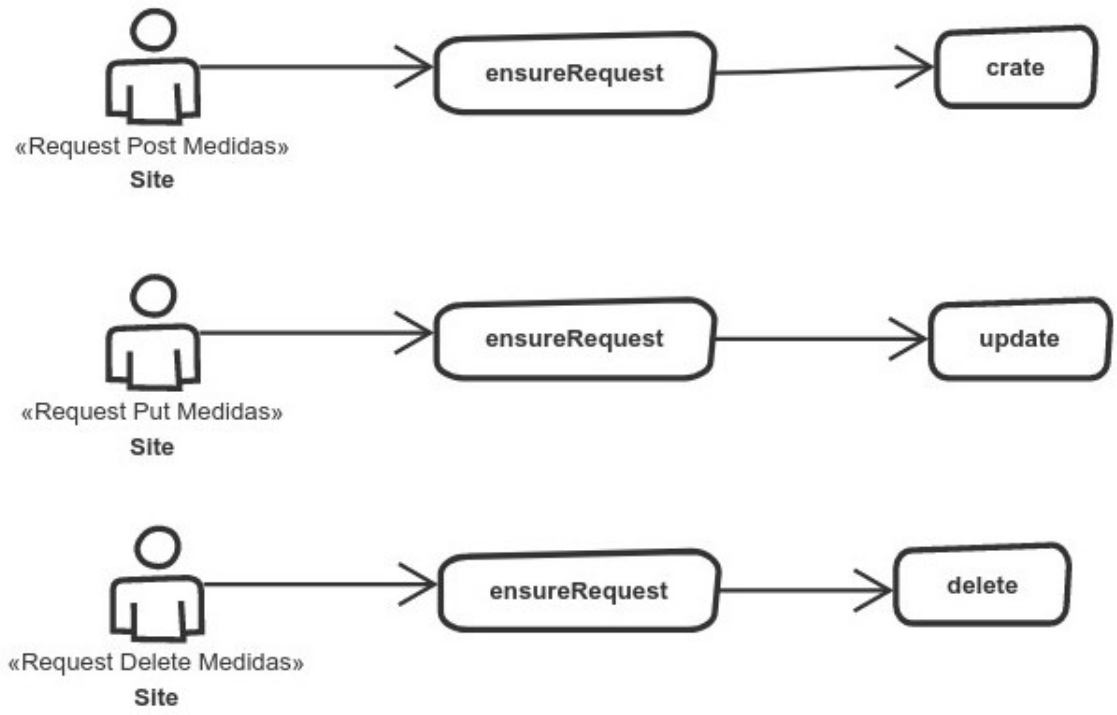


Figura 17: Estrutura das funcionalidades de Medidas

Durante o processo de estruturação, definiu-se que uma medida estava dividida em categorias/áreas e que estas estariam incluídas em planos de licenças, ou seja:

- Medidas: regras a serem seguidas;
- Categorias: áreas nas quais as medidas estão inseridas;
- Licenças: número de medidas e funcionalidades a que cada cliente terá acesso.

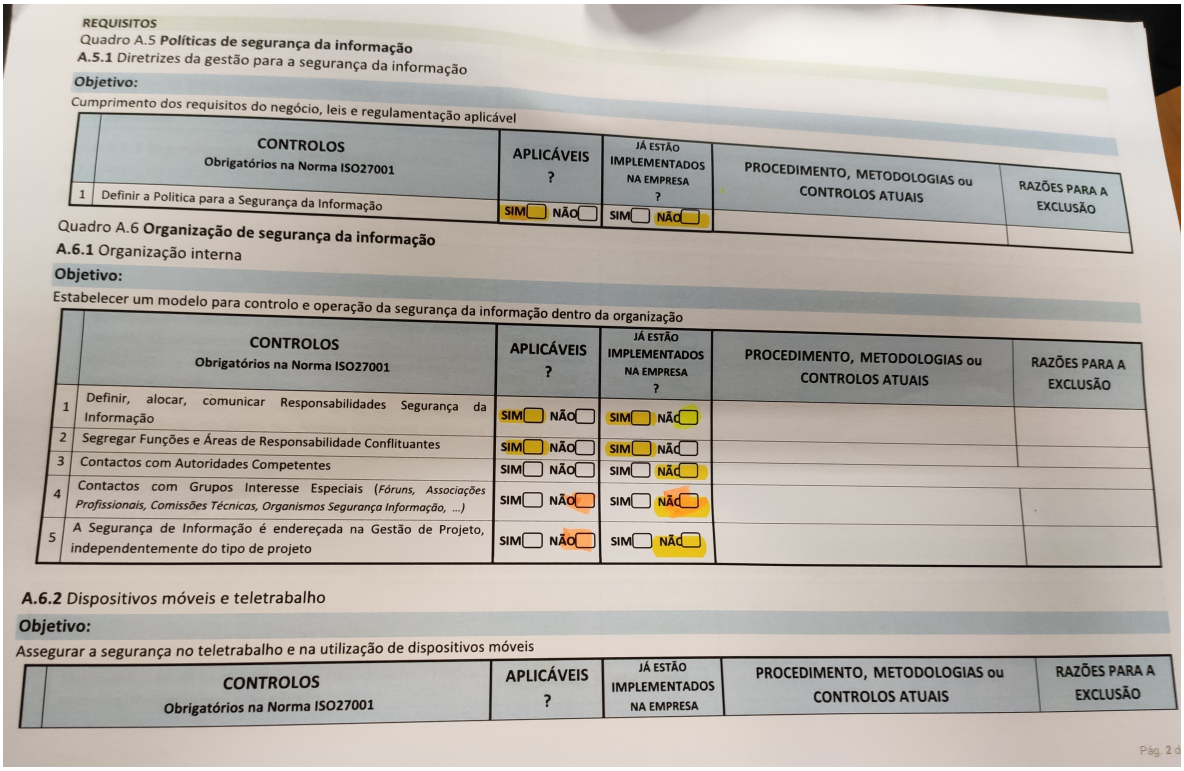
Os principais pedidos que são permitidos nas medidas são dois *GET's*, que facilitam o acesso a informação sobre as mesmas, um *POST* para criar e um *PUT* para atualizar. (Ver figura 18)

Figura 18: API's Medidas

Relatórios

Este setor é o mais importante porque junta as mediadas e apreciações num relatório de conformidade que servirá de apoio à certificação do RGPD, levando ao cumprimento do principal objetivo da solução. Ver capítulos 2 e 3 para recordar.

O ponto de referência para este planeamento foram as necessidades e documentos fornecidos pelo DPO, que ajudaram a perceber o que era necessário ser feito. A figura 19 mostra um exemplo real de um relatório de conformidade que é elaborado por um DPO. O facto de ser em papel levou à ideia para o desenvolvimento da solução.


REQUISITOS

Quadro A.5 **Políticas de segurança da informação**

A.5.1 Diretrizes da gestão para a segurança da informação

Objetivo:

Cumprimento dos requisitos do negócio, leis e regulamentação aplicável

	CONTROLOS Obrigatórios na Norma ISO27001	APLICÁVEIS ?	JÁ ESTÃO IMPLEMENTADOS NA EMPRESA ?	PROCEDIMENTO, METODOLOGIAS ou CONTROLOS ATUAIS	RAZÕES PARA A EXCLUSÃO
1	Definir a Politica para a Segurança da Informação	SIM NÃO	SIM NÃO		

Quadro A.6 **Organização de segurança da informação**

A.6.1 Organização interna

Objetivo:

Estabelecer um modelo para controlo e operação da segurança da informação dentro da organização

	CONTROLOS Obrigatórios na Norma ISO27001	APLICÁVEIS ?	JÁ ESTÃO IMPLEMENTADOS NA EMPRESA ?	PROCEDIMENTO, METODOLOGIAS ou CONTROLOS ATUAIS	RAZÕES PARA A EXCLUSÃO
1	Definir, alocar, comunicar Responsabilidades Segurança da Informação	SIM NÃO	SIM NÃO		
2	Segregar Funções e Áreas de Responsabilidade Conflituantes	SIM NÃO	SIM NÃO		
3	Contactos com Autoridades Competentes	SIM NÃO	SIM NÃO		
4	Contactos com Grupos Interesse Especiais (*Fóruns, Associações Profissionais, Comissões Técnicas, Organismos Segurança Informação, ...*)	SIM NÃO	SIM NÃO		
5	A Segurança de Informação é endereçada na Gestão de Projeto, independentemente do tipo de projeto	SIM NÃO	SIM NÃO		

A.6.2 Dispositivos móveis e teletrabalho

Objetivo:

Assegurar a segurança no teletrabalho e na utilização de dispositivos móveis

	CONTROLOS Obrigatórios na Norma ISO27001	APLICÁVEIS ?	JÁ ESTÃO IMPLEMENTADOS NA EMPRESA	PROCEDIMENTO, METODOLOGIAS ou CONTROLOS ATUAIS	RAZÕES PARA A EXCLUSÃO

Pág. 2 d

Figura 19: Exemplo de documento fornecido por DPO - IdealMais

A funcionalidade dos relatórios permite a um DPO criar um documento e guardá-lo antes de o finalizar, podendo ser alterado as vezes que forem necessárias até todas as medidas estarem no estado concluídas.

Igualmente nesta fase, os relatórios podem ser removidos. No entanto, quando passam para o estado de concluído, essa opção deixa de existir.

A figura 20 demonstra o que foi referido.

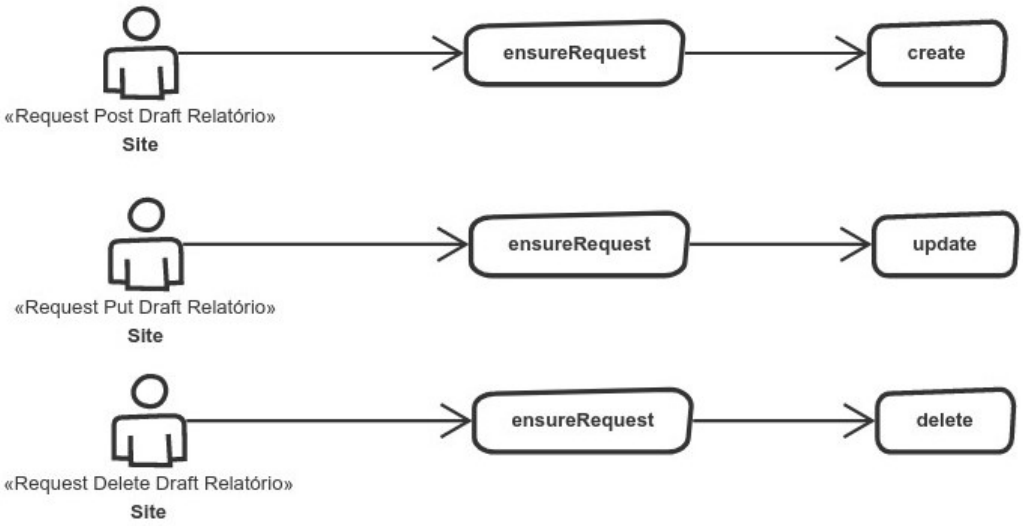


Figura 20: Estrutura da funcionalidade de Relatório

A figura 21 representa as *API's* que são possíveis nos relatórios. *GET's* que ajudam a obter os dados, *POST* que cria e *PUT* que atualiza.

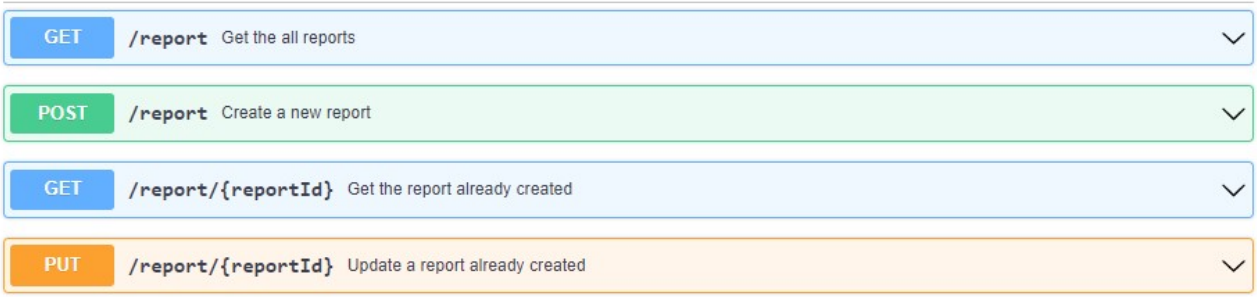


Figura 21: API's Relatórios

Para a compreensão da funcionalidade de relatórios, a figura 22 representa os passos a alto nível da mesma.

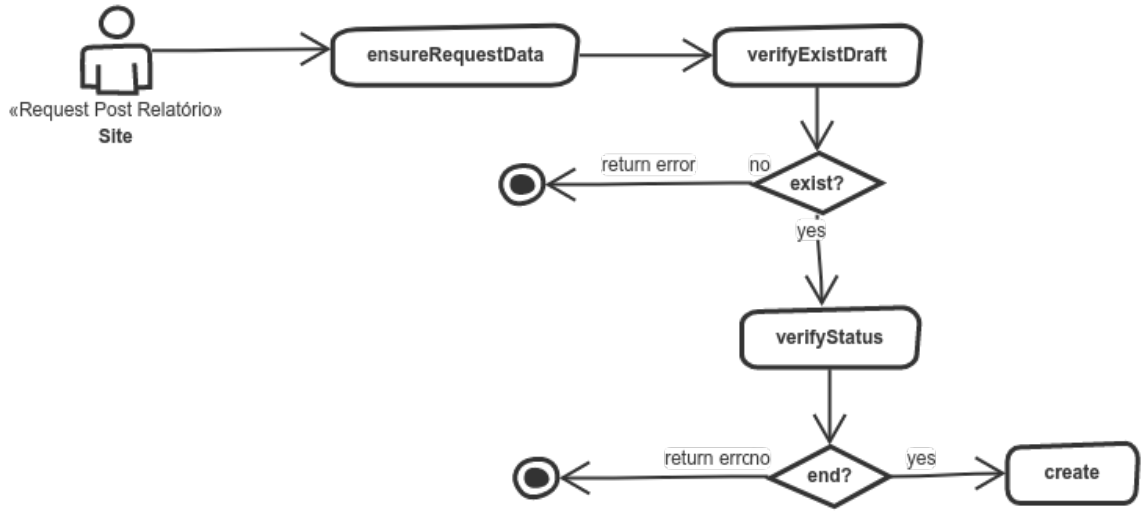


Figura 22: Estrutura de alto nível dos Relatórios

4.2 BASE DE DADOS

Durante o processo definiu-se que todas as necessidade de base de dados seriam implementadas em *Postgres* com a exceção dos relatórios que, devido à seu estrutura e pelo facto de já se encontrarem em *JSON*, seriam guardados em *Mongo*.

Postgres

Na base de dados encontram-se todas as tabelas relacionadas com clientes, entidades e configurações de sistemas. A figura 23 representa um diagrama que mostra as tabelas com os dados dos clientes.

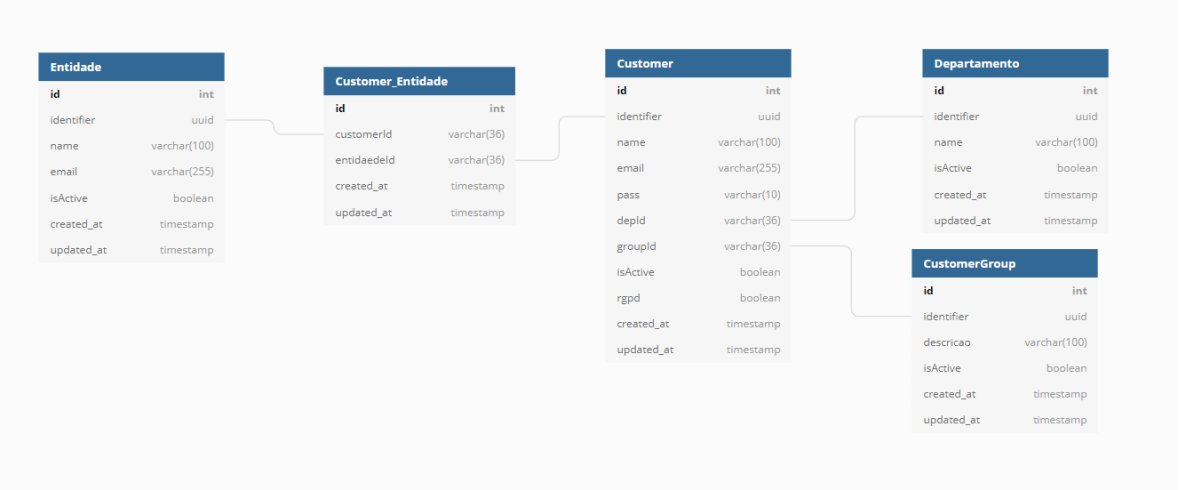


Figura 23: Estrutura da Base de Dados

Mongo

Os documentos de relatório são guardados com uma estrutura que contém:

- Key: identificação do documento.
- Operation: tipo de relatório.
- When: quando é que foi criado.
- Body: conteúdo do documento.

Um exemplo disso é o *JSON* representado em baixo.

```
{
    "_id" : ObjectId("60f8b0f77497ee64e956ff7b"),
    "key" : "uaehiuahefd-doipkaeoidk",
    "operation" : [
        "##NameTypeDoc##"
    ],
    "when" : [
        "##Occurence.when##"
    ],
    "body" : [
        "##Measures.body##"
    ]
}
```

Por sua vez, o *body* é estruturado da seguinte forma:

- Área: setor de atuação da medida.
- Responsável: pessoa que fica responsável pela correção da medida.
- Medida: regra que tem de ser confirmada.
- Estado da medida: ponto de situação da medida.
- Notas: informação adicional que pode ser relevante.
- Tarefas: indicação das ações que têm de ser executas.

O *JSON* a seguir apresentado exemplifica a estrutura do *body*.

```
{
    "area": "Operacoes",
    "responsavel": "as1d1f11dd",
    "medida": "Existe seguran a nos postos de trabalho?",
    "estadoMedida": "Conclu do",
    "notas": [
        {
            "descricao": "Existem senhas, mas sao muito intuitivas"
        },
        {
            "descricao": "Vai ser elaborada uma tarefa para definir senhas com os crit rios de
                seguran a definido pelo CNCS."
        }
    ],
    "tarefas": [
        {

        }
    ],
    "anexos": [
        {

        }
    ]
},
{
    "area": "",
    "responsavel": "22de2t2t",
    "medida": "",
    "estadoMedida": "",
    "notas": [
        {
```

```
            "descricao": ""
        }
    ],
    "tarefas": [
        {
        }
    ],
    "anexos": [
        {
        }
    ]
}
```

4.3 ENCRIPTAÇÃO

Relativamente às estratégias de segurança de dados, foram desenvolvidas duas partes distintas: uma com chaves que servem para encriptar e desencriptar dados e outra que, através de um *token*, permite atribuir um período de validade, o que obriga a gerar um novo *token* para se poder volta a fazer pedidos ao *backend*.

Chaves

As chaves do tipo "certificado"e "privada"foram geradas através de um sistema *unix* com a execução dos seguintes comandos:

- openssl req -newkey rsa:2048 -new -nodes -keyout key.pem -out csr.pem
- openssl x509 -req -days 365 -in csr.pem -signkey key.pem -out server.crt

Depois de geradas, foram introduzias na base de dados para serem consultadas pelo sistema. A sequência seguinte mostra um exemplo de chave privada e de certificado.

- Chave Privada

```
 -----BEGIN PRIVATE KEY-----
MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQCiihOmC5ZoE0vb
IrdnpHvzmRGKwHSKkpvB/bI+4qJWin4UI5w/kTgF97fbYv4kCL5tQ6jxxFfK7gd0
WHqXH1X364eY2iDHo8BrIAS8fX6FQLPOykahCm58LbRliAXqaHZLKdoY1gt3XgzS
7e/LQj+jfZN/xtXulFSyDHK859i0KVr0uyo/nb+hUM3VwL2uttYvR7h+cIV0+hiY
t9c4PlmLAqAx2njM5zjTKgjYMh/7Qd76TZ3CC4ddgWMGFiXuU5DozqnlHZfh5A63
pCks4HRAkyFNSo1VKD9TDLKVBWvS/sVCUYpRvs8Go/kstBicBaKMGtLBYfUzmWGi
FAn8ftdbAgMBAAECggEAVK0WO2TCAIo89HaLkaTxJiDSMLS+ggqKtm9SUsyCwg74
```

```
S8mZdsxSoVlCITrbJy6e9hFPjFQ7YG7M12HYy635fWToiUrp1NeMQMK3dPLz3Luu
cMaUMQVcNbkhx9srKckuB05QXqvz4UXP7Hskd4GV4bhkBSczAE2qKe+uPpeSH5iv
HT+GoC8W2++dA8rDenVHRER+P2XTRgHc4ciYSvzyVvXdmdk5NqYwJZqkqC2axQ/y
TbU8blbetpS/KZwjNAEuMaqiNpEaqrKUqsuZSV2g8MmsaNBXwDwaVpXHK28eaSlH
0z0yV/hJahg0ZioMlO6hWrJdGfq0zQqockhzul/dAQKBgQDUzENJixeur7t7zSpZ
0NnJ4mxpuykqCVcPm9wAVz1HlXN6et09ps6kBt+FRd9OsbmOLlSugBSsoAXEJ++w
bS+4czl6pkiu2sm9lSlaMs8v7tRfZ6U1LmdsVra/pMwHswN0Ng74OoWy18UWDQwl
I4Jn7ceBZmmYxM4wg22H/5sPwQKBgQDDibqRF7IWUdNHD4/2U0cpxWmxUY/jf0X6
IRBAIBV4UuyGP1fnsio5zXwekOlyee9QcH/DqrugaLp8jcagkow3sZs2w7fSyjV6
1CrofjyttZ18pBeFZ9KipdY6bl63wbAeSCmD4W1RAwZO+TerQbjxdQNYOJXYYNJC
SIcuPi+uGwKBgHaedlOkiUsDAot8TkNKvFxhLvWxMr0P/8vfob8+4hXpJTn9Cx62
wTGF80sjK9t+DmVHRkUPeHbRrOUvLdk2xAcZk8zerF1r8FERuL9i20No6lFUQS9n
8bvZnebanxlugbYDGbn2f+2Fwx32fqy/uL4nOWAdSGiwi8N8t+g2uKS/0JAq5X2c
UTG1ilzb7QPRPRJw+lBOC2vDRSkDw34pg/a99HPLEbwFcJxadMA8/hNG5Z4wKj+u
IVqimAXdkSS5Nmj/J0z+Wg==
-----END PRIVATE KEY-----
```

- Certificado

```
 -----BEGIN CERTIFICATE REQUEST-----
MIICijCCAXICAQAwRTELMAkGA1UEBhMCQVUxEzARBgNVBAgMClNvbWUtU3RhdGUx
ITAfBgNVBAoMGEludGVybmV0IFdpZGdpdHMgUHR5IEx0ZDCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBAKKKE6YLlmgTS9sit2eke/OZEYrAdIqSm8H9sj7i
olaKfhQjnD+ROAX3t9ti/iQIvm1DqPHEV8ruB3RYepcfVffrh5jaIMejwGsgBLx9
foVAs87KRqEKbnwttGWIBepodksp2hjWC3deDNLt78tCP6N9k3/G1e6UVLIMcrzn
2LQpWvS7Kj+dv6FQzdXAva621i9HuH5whXT6GJi31zg+WYsCoDHaeMznONMqCNgy
H/tB3vpNncILh12BYwYWJe5TkOjOqeUdl+HkDrekKSzgdECTIU1KjVUoP1MMspUF
a9L+xUJRilG+zwaj+Sy0GJwFoowa0sFh9TOZYaIUCfx+11sCAwEAAaAAMA0GCSqG
SIb3DQEBCwUAA4IBAQB1Wxn08Q6psyIqt25C+1OdAc2Dn3njwPX1al2RliJmxAEI
ibgh+ErnejK34lnDgSX35cXaQdok5YSFN9V4W3uGolmggjmNBQAROjFRfjAqYXhX
8q4aISI2vyfd0ZQ4oO9lxx7Ay4dyThtvZ93bfgPRrvyi0xz61V5XeFIghF4Fur5p
iy+6rS1PxNk/tMxR4hJoG6k4+OFW2pZZlVRXQFMO43txO7JFEGn/cEhqYiS6rE1Y
3LUMP8nR3lDf26eqwIJbun6RSetpQsiCYETZbAZ4uEpvfnalpRDYfw8cTRQfCpeE
DnJh/vcqQIwgvO8oIXqG9wlUf2GjVtVFjOyOQFmo
-----END CERTIFICATE REQUEST-----
```

- Exemplo de código utilizado para a encriptação de dados

```
encrypt(value) {
   var key = {
     key: this.keyData,
     passphrase: this.passphrase,
     padding: crypto.constants.RSA_PKCS1_PADDING
   };
   return crypto
     .publicEncrypt(key, Buffer.from(JSON.stringify(value)))
     .toString("base64");
 }
```

- Exemplo de código utilizado para a desencriptação de dados

```
function decrypt(data) {
 return crypto
   .privateDecrypt(
     {
       key: privateKey,
       passphrase: passphrase,
       padding: crypto.constants.RSA_PKCS1_PADDING
     },
     Buffer.from(data, 'base64')
   )
   .toString();
}

// Write Javascript code!
const valueToDecrypt =
  'YSVR000F5uVnWVLYjTe8XmJVFluE9pMX5onjF2gQs9C+hKNa2g3lgdZZR4+
      CyZ6QsnDIuycfGm1Ev5nokT9cNoD6bUwP9blmBBC/
      GptEnyQuCHzCuaoxEC0kp6tgiTbGodsh9SRikCcqcgfhwM7I/UqDWVWR+RmLzu/ZMk+n6lm74h/
      DVIMZtHiU0ggTVxFmdZQ3xNKU4ITJ0POJ9GnUBwZeBawCznRQgteQgNX3x91SSoVlc
  QbJHf5VsQiWIkJPxlfgvgznS0jTCkyQCeKNTGCUCo6QDiFokfVDdINt73RiH/
      v7PSrTzDkYGMM86nrJr16thNYLhkqEcoZCxxuj9RXNxQ0VhBAGRamK1ECyLSQ/
      XnNgyhc6ynsVXWIy3gamELJrR12mmiwLQDxH3deF/ZFI5UiepgG8VvnOwDB/
      upaCF3Ry6jNsLhdFU4Kx8Vo6ZsXGXkIEQ1r0bl0z6/9
      iLvharADbuQy2rcL6lnaxbjnpz9j9UbO8Q7BiHaOnD1BsiO5ffvZseSdYL8Cw+
      wsYnKrHG1kOYSxetUbFbwCcX1qLx9L5f95KsXhHSoLKrveaI9Ffz97j4ybq5C6o3GpsUz/86Y/
      mogVXvypCLyET6U+Gb9pcSH7/Xh7J9r0dpzbHFdRG9TrmoBv8/h6YOV3tvcAziOGV6axIPL0MN7IktrU=';

const obj = JSON.parse(decrypt(valueToDecrypt));
```

Bear Token

Sempre que é efetuado um pedido via API ao *backend*, este necessita de conter um *token*. Nesse sentido, foi criada uma *API* que gera um *token* e o devolve como campo de *autentication* no pedido que pretende efetuar.

No processo de gerar um *token*, é obtido como resposta um *JSON*, como se pode observar abaixo.


```
{
   "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6Im5PbzNaR
   HJPRFhFSzFqS1doWHNsSFJfS1hFZyIsImtpZCI6Im5PbzNaRHJPRFhFSzFqS1do
   WHNsSFJfS1hFZyJ9.eyJhdWQiOiJjMTc3MDdkNi0yYjI2LTRhZTktOGFmMC01ZD
   U1ZTJiYTJlZTYiLCJpc3MiOiJodHRwczovL3N0cy53aW5kb3dzLm5ldC81NzU3Y
   ThkYS0wNDc3LTQ2ZDQtYmM4YS1iMGZlYWJkY2M3MWEvIiwiaWF0IjoxNjI3Mzk2
   MTQ3LCJuYmYiOjE2MjczOTYxNDcsImV4cCI6MTYyNzQwMDA0NywiYWlvIjoiRTJ
   aZ1lGaHdablhvSG84elJvOHpQUlpNMU9VNENBQT0iLCJhcHBpZCI6IjhhYzg3OD
   diLWVlMWQtNDZiYi1iMWU3LTNlYWVmM2MwNzJhYiIsImFwcGlkYWNyIjoiMSIsI
   mlkcCI6Imh0dHBzOi8vc3RzLndpbmRvd3MubmV0LzU3NTdhOGRhLTA0NzctNDZk
   NC1iYzhhLWIwZmVhYmRjYzcxYS8iLCJvaWQiOiJjOGIxNzM4My1mNDE5LTQyYzg
   tOTY0Ny1lOTQ1ODRjOWE5MjAiLCJyaCI6IjAuQVFJQTJxaFhWM2NFMUVhOGlyRC
   1xOXpIR250NHlJb2Q3cnRHc2VjLXJ2UEFjcXNDQUFBLiIsInJvbGVzIjpbIlVzZ
   XJzUmVxdWVzdCJdLCJzdWIiOiJjOGIxNzM4My1mNDE5LTQyYzgtOTY0Ny1lOTQ1
   ODRjOWE5MjAiLCJ0aWQiOiI1NzU3YThkYS0wNDc3LTQ2ZDQtYmM4YS1iMGZlYWJ
   kY2M3MWEiLCJ1dGkiOiJ4SEZkcHE1Q1VVcUk5Wmcwc0pWckFBIiwidmVyIjoiMS
   4wIn0.F8nm7L7Y2qux__o7c5qu8VqKa2gXPtWOKaFfSTOSJkQQW4kt1lo9rtHJu
   pxWL4v1ZJyFd44f7YLdX-9tSZuqN_JyiflUcI2nGa9-CapTop48O4E4bN4c2Eb8
   tUjDaAoYLEQNML6hXAYDg13j2PN4bPgWhe_o1JnDcg1k3V9sisel8y8zt1eIuwo
   7nEnknTQTP9_xgUo_w3Z4u_EHrwGe0gwLrZoE6lZN73p4uSF8UDY8CK1YcAtyJD
   j1eBEgMJtoi76kpnlaaMQJZmkXckl4luKAFMIoFRACPwdDq7idNf_oP76uTo2RA
   bg2x0abIzkeVb9zjo9hDIgcHTu4NHZqCw"
}
```


4.4 BACKEND

Ao estruturar o sistema, tentou criar-se um esquema de pastas que fosse fácil de reconhecer e, para isso, muita das vezes foram consultados os *design-patterns* com o intuito de ajudar a obter um estrutura mais sólida.

Algumas dessas divisões são apresentadas na figura 24.

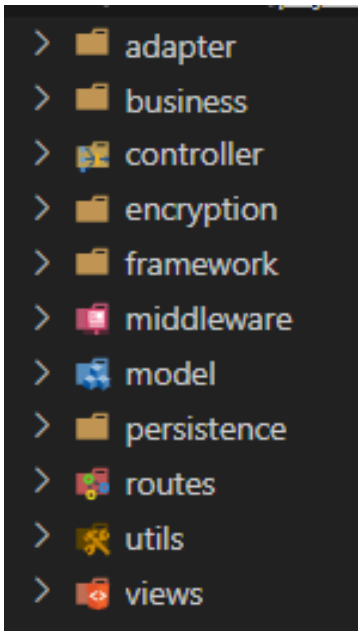


Figura 24: Estrutura de pastas do projeto

- Rotas

  Características significantes:

  - São os pedidos *REST* que chegam ao sistema;
  - Estão divididos em *POST*, *PUT*, *DELETE*;
  - Chamam um *middleware* de segurança que verifica se o pedido é valido;
  - Invocam os *controllers* consoante o pedido pretendido;

  A seguir, é possível visualizar um exemplo de código das rotas.

```
const router = Router({ mergeParams: true});

router.post("/", SecurityMiddleware.ensureKey, CustomerController.create);

router.put("/customerId", SecurityMiddleware.ensureKey ,CustomerController update) ;

router.delete("/customerId", SecurityMiddleware.ensurekey, CustomerController.delete)
     ;

export = router;
```

- *Model*

  Características significantes:

  - Contêm as interfaces;
  - Podem ser utilizados nos pedidos (*request*) para identificação, bem como nas respostas (*response*);

  O código seguinte mostra um exemplo de *model*.

```
export declare namespace Customer {

  interface Model{
      customerId: number;
      identifier: string;
      name: string;
      email: string;
      password: string;
      depId: string;
      groupId: string;
      isActive: boolean;
      rgpd: boolean;
  }

}
```

- *Controllers*

  Características significantes:

  - São invocados pelas rotas;
  - São responsáveis por efetuar uma primeira verificação do pedido;
  - Invocam os ficheiros de *business* consoante o tipo do pedido;
  - Retornam a resposta ao utilizador;

- *Business*

  Características significantes:

  - Responsáveis por executar o que foi pedido pelo negócio;
  - Comunicam com os repositórios, com o fim de obter dados relevantes.

- Repositórios

  Características significantes:

  - Responsáveis por comunicar com as base de dados;
  - Devolvem informação pedida à camada *business*.

- *Middleware*

  Características significantes:

  - Responsáveis por certificar informação antes dela chegar ao *controller*;
  - Servem como uma camada de segurança do sistema;
  - Podem ser utilizados por todo o sistema.

4.5 testes automáticos e unitários

Ao longo de todo o processo de desenvolvimento, foram sendo criados testes com o propósito de minimizar os problemas que o sistema pudesse ter. O testes foram divididos em dois tipos: por um lado, os testes unitários para a certificação de métodos e *models*, por outro lado, a utilização de um *plugin* do *Jest*, chamado *supertest*, que permitiu testar uma funcionalidade simulando um pedido *REST API*. (Ver capitulo 2 secção 2.10.5)

- Testes Unitários

  Com a utilização de teste unitários foi possível, por exemplo, confirmar se a verificação dos dados está a ser feita corretamente. Abaixo, encontra-se código relativo a estes testes.

```
describe('Customer Service', () => {
 describe('Ensure name is a string', () => {
   it('should accept a string', () => {
     const result = CustomerRules.ensureString('Jo o Paulo');
     expect(result).toBeUndefined();
   });
 });

 describe('Ensure mail is a string', () => {
   it('should accept a string', () => {
     const result = CustomerRules.ensureString('teste@gmail.com');
     expect(result).toBeUndefined();
   });
 });

 describe('Ensure isActive is a boolean', () => {
```

```
      it('should accept a boolean', () => {
        const result = CustomerRules.ensureBoolean(false);
        expect(result).toBeUndefined();
      });
    });
  });
```

- Testes *Supertest*

  O *supertest* funciona da mesma forma que o pedido *REST*. Os *headers* e o *body* podem ser declarados, permitindo efetuar um pedido *end-to-end*. É um tipo de teste bastante útil porque permite testar a funcionalidade do pedido desde o momento em que é feito até ao momento em que vai à base de dados. Além disso, permite ainda verificar se a resposta obtida é a pretendida.

  Para se poder utilizar o *supertest* é primeiro necessário declarar os tipos de pedidos que podem ser feitos (*POST, PUT, DELETE*, ...).

```
import supertest from "supertest";

const PORT = 8060;
const { post, get, put } = supertest('http://localhost:${PORT}/api/');
```

Depois de termos o *supertest* declarado, basta criar um pedido, como por exemplo *POST*, que contenha todos os dados pretendidos para o teste.

O código abaixo apresenta um exemplo de testes feitos com *supertest*.

```
      describe('Customer - Behavior Test', () => {
          it('should generate a Customer', async () => {
              const {statusCode, body} = await post('/')
                      .set("apiKey", requestMock.headers["apiKey"])
                      .set("Application-Name", requestMock.headers["Application-Name"])
                      .send(requestMock.body);
              console.log(body);
              expect(statusCode).toBe(201);
          });

          it('should update a Customer', async () => {
              const {statusCode, body} = await put('/1wertodir')
                      .set("apiKey", requestMock.headers["apiKey"])
```

```
                    .set("Application-Name", requestMock.headers["Application-Name"])
                    .send(requestMock.body);
            console.log(body);
            expect(statusCode).toBe(200);
        });

        it('should update a Customer', async () => {
            const {statusCode, body} = await delete('/1wertodir')
                    .set("apiKey", requestMock.headers["apiKey"])
                    .set("Application-Name", requestMock.headers["Application-Name"])
                    .send(requestMock.body);
            console.log(body);
            expect(statusCode).toBe(200);
        });
    });
```

5

CONCLUSÃO

O último capítulo desta dissertação expõe as considerações finais obtidas do desenvolvimento da solução.

5.1 CONSIDERAÇÕES FINAIS

Esta dissertação de mestrado teve como principal objetivo o desenvolvimento de uma ferramenta *web* que ajudasse na gestão do Regulamento Geral de Proteção de Dados Pessoais e que pudesse ser ser adaptada, ou sintonizada, a vários temas e modelos de negócio.

A ferramenta foi planeada e concebida para ser aplicada numa entidade empresarial. Porém, de maneira a compreender o problema em estudo, levou-se a cabo uma investigação relativamente aos principais conceitos teóricos e tecnológicos.

Durante o desenvolvimento desta dissertação de mestrado foram seguidas as etapas que a entidade para a qual este projeto foi proposto e desenvolvido exigiu, mais concretamente, levantamento de requisitos, arquitetura, *design* e desenvolvimento.

Inicialmente, o que estava proposto para o projeto era a criação de uma solução de raiz, ou seja, planeamento, *frontend* e *backend*. No entanto, foi decidido pela entidade que a participação seria apenas no desenvolvimento do *backend* e na arquitetura.

O facto de a participação ser apenas no desenvolvimento do *backend* trouxe bastantes desafios, sobretudo na área da programação, que são, de seguida, enumerados:

- Conhecimento de *TypeScript*;
- Conhecimento de *JEST*;
- Conhecimento de *RSA*;
- Questões de segurança.

Terminada a consciencialização do problema e o seu estudo, procedeu-se à elaboração de possíveis diagramas e ficheiros *"YAM"* que serviram para especificar as *"spec's"* das *API's* da solução.

De seguida, levou-se a cabo o desenvolvimento desta proposta, tendo sido explicado todo o processo e decisões tomadas. O processo foi de forma sequencial, ou seja, começou-se por preparar o ficheiro *express* do *nodejs* e, posteriormente, criaram-se pastas e ficheiros de código.

O código foi desenvolvido seguindo a ordem dos *designs patterns*:

- Rotas
- Models
- Controller
- Negócio
- Repositórios

O desenvolvimento do *backend* revelou-se ser o pretendido pela entidade, o que levantou a possibilidade de um trabalho futuro com a mesma.

5.2 TRABALHOS FUTUROS

Apesar da solução desenvolvida corresponder ao que foi inicialmente proposto, é concebível o planeamento de uma série de possíveis melhorias a serem aplicadas à solução.

Com o propósito de melhorar a ferramenta elaborada, destaca-se a possibilidade de aperfeiçoar as questões de segurança, utilizando, para isso, uma integração do sistema com uma plataforma de análise de erros como *Google Analytics, Piwik PRO* ou *New Relic*.

Além disso, a melhoria de testes automáticos pode ser outro ponto a ser trabalhado, não só para despiste de possíveis erros, mas também para facilitar os testes e a certificação da solução.

BIBLIOGRAFIA

[1] General Data Protection Regulation (GDPR) , Digital Guardian, *https : //digitalguardian.com/blog/what − gdpr − general − data − protection − regulation − understanding − and − complying − gdpr − data − protection*

[2] Directive 95/46/EC, Eur-Lex, *https : //eur − lex.europa.eu/*

[3] What is GDPR, zdnet, *https : //www.zdnet.com/article/gdpr − an − executive − guide − to − what − you − need − to − know/*

[4] DPO - Data Protection Officer, *https : //www.investopedia.com/terms/d/data − protection − officer − dpo.asp*

[5] Artigo 37, UE Regulamento Geral sobre a Proteção de Dados, Privacy-Regulation, *http : //www.privacy − regulation.eu/pt/37.htm*

[6] Artigo 39, UE Regulamento Geral sobre a Proteção de Dados, Privacy-Regulation, *http : //www.privacy − regulation.eu/pt/39.htm*

[7] Ariani Di Felippo and B. C. Dias-da-Silva, Uma introdução à Engenharia do Conhecimento Linguístico, *http : //www.nilc.icmc.usp.br/nilc/download/ariani/DiFelippo$_D$iasDaSilva$_R$evista$_d$e$_L$etras$_2$008.pdf*

[8] Manual de aplicação do RGPD, *Livro− SofiaCaseiro*

[9] Aplicação do RGPD, *https : //sofia − caseiro.pt/category/rgpd/ SofiaCaseiro*

[10] Artigo 9.o RGPD (GDPR). Tratamento de categorias especiais de dados pessoais *https : //gdpr − text.com/pt/read/article − 9/ GDPRTEX*

[11] Artigo 12.o RGPD (GDPR). Transparência das informações, das comunicações e das regras para exercício dos direitos dos titulares dos dados *https : //gdpr − text.com/pt/read/article − 12/ GDPRTEX*

[12] Artigo 13.o RGPD (GDPR). Informações a facultar quando os dados pessoais são recolhidos junto do titular *https : //gdpr − text.com/pt/read/article − 13/ GDPRTEX*

[13] Artigo 14.o RGPD (GDPR). Informações a facultar quando os dados pessoais não são recolhidos junto do titular
https : //gdpr − text.com/pt/read/article − 14/ GDPRTEX

[14] Artigo 15.o RGPD (GDPR). Direito de acesso do titular dos dados
https : //gdpr − text.com/pt/read/article − 15/ GDPRTEX

[15] Artigo 32.o RGPD (GDPR). Segurança do tratamento
https : //gdpr − text.com/pt/read/article − 32/ GDPRTEX

[16] Artigo 33.o RGPD (GDPR). Notificação de uma violação de dados pessoais à autoridade de controlo
https : //gdpr − text.com/pt/read/article − 33/ GDPRTEX

[17] Artigo 34.o RGPD (GDPR). Comunicação de uma violação de dados pessoais ao titular dos dados
https : //gdpr − text.com/pt/read/article − 34/ GDPRTEX

[18] Artigo 37.o RGPD (GDPR). Designação do encarregado da proteção de dados
https : //gdpr − text.com/pt/read/article − 37/ GDPRTEX

[19] Artigo 38.o RGPD (GDPR). Posição do encarregado da proteção de dados
https : //gdpr − text.com/pt/read/article − 38/ GDPRTEX

[20] Artigo 39.o RGPD (GDPR). Funções do encarregado da proteção de dados
https : //gdpr − text.com/pt/read/article − 39/ GDPRTEX

[21] A Look back on Cyber Security 2012
Livro− LuisCorrons

[22] A Study Of Cyber Security Challenges And Its Emerging Trends On Latest Technologies
*https : //www.researchgate.net/publication/*260126665$_{A}$*Study*$_{O}$*f*$_{C}$*yber*$_{S}$*ecurity*$_{C}$*hallenges*$_{A}$*nd*$_{I}$*ts*$_{E}$*mer*
NikhitaReddyGade, UganderGJReddy

[23] A Method for Obtaining Digital Signatures and Public-Key Cryptosystems
https : //people.csail.mit.edu/rivest/Rsapaper.pdf R.L.Rivest, A.Shamir, andL.Adleman

[24] Combination of Rivest-Shamir-Adleman Algorithm and End of File Method for Data Security
*https : //www.researchgate.net/publication/*323731078$_{C}$*ombination*$_{o}$*f*$_{R}$*ivest − Shamir − Adleman*$_{A}$*lgorithm*$_{a}$*nd*$_{E}$*nd*$_{o}$*f*$_{F}$*ile*$_{M}$*ethod*$_{f}$*or*$_{D}$*ata*$_{S}$*ecurity*
DianRachmawati, AmaliaAmalia, ElviwaniElviwani

[25] Application Programming Interface Documentation: What Do Software Developers Want?
https://www.researchgate.net/publication/318733467_Application_Programming_Interface_Documentat
Michael Meng, Stephanie Steinhardt, Andreas Schubert

[26] Web Application Programming Interfaces (APIs): general-purpose standards, terms and European Commission initiatives
https://publications.jrc.ec.europa.eu/repository/bitstream/JRC118082/jrc118082_api-landscape-standards.pdf Santoro, M., Vaccari, L., Mavridis, D., Smith, R.S., Posada, M., Gattwinkel, D

[27] Using Node.Js to Build High Speed and Scalable Backend Database Server
https://www.researchgate.net/publication/301788361_Using_NodeJs_to_Build_High_Speed_and_Scalable_Backe
Sunil L. Bangare

[28] A Comparative Analysis of Node.js (Server-Side JavaScript)
https://core.ac.uk/download/pdf/232792216.pdf Nimesh Chhetri

[29] Node Documentation
https://nodejs.org/en/docs/

[30] Understanding TypeScript
https://www.researchgate.net/publication/295731712_Understanding_TypeScript
Gavin Mark Bierman, Martn Abadi, Mads Torgersen

[31] Typescript Documentation
https://www.typescriptlang.org/docs/

[32] JEST Documentation
https://archive.jestjs.io/docs/en/22.x/getting-started.html

[33] Postman Documentation
https://www.postman.com/api-documentation-tool/

[34] Postgres Documentation
https://www.postgresql.org/docs/

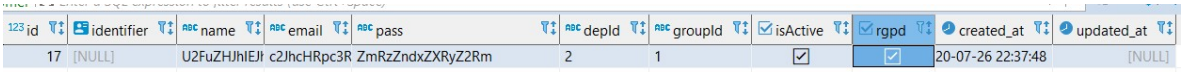
[35] Mongo Documentation
https://docs.mongodb.com/

A

MATERIAL DE SUPORTE

A.1 CLIENTES DADOS ENCRIPTADOS

Sempre que é criado um cliente, os dados pessoais são encriptados e guardados na base de dados, como mostra a figura 25.



id	identifier	name	email	pass	depId	groupId	isActive	rgpd	created_at	updated_at
17	[NULL]	U2FuZHJhIEJ	c2JhcHRpc3R	ZmRzZndxZXRyZ2Rm	2	1	☑	☑	20-07-26 22:37:48	[NULL]

Figura 25: Exemplo dos dados encriptados em Base de Dados

A.2 MONGO DB

Na base de dados mongo foi criada uma *collection* chamada *documents* que é responsável por conter os *JSON* dos relatórios de aceitação. A figura 26 mostra um exemplo disso.

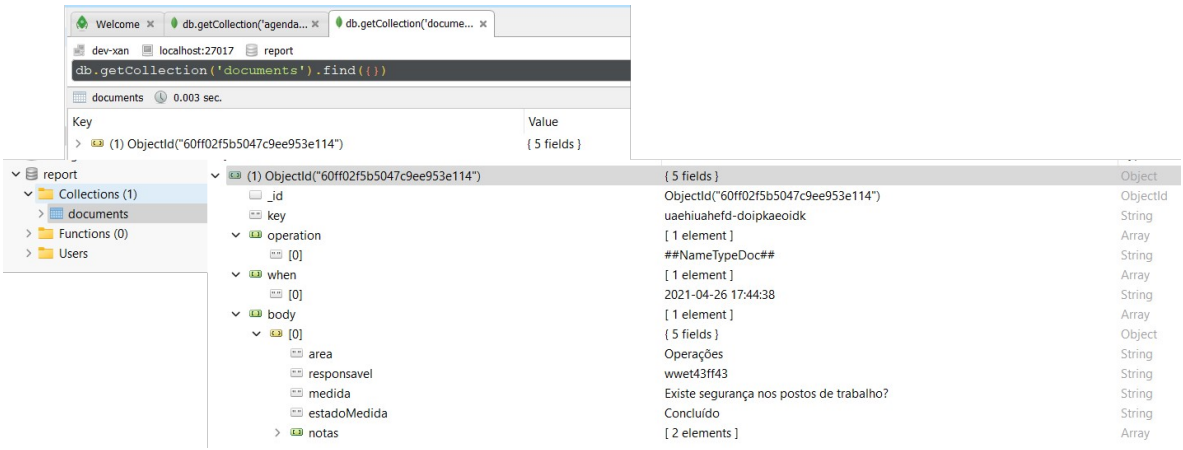


Figura 26: Estrutura e Exemplo de documento

A.3 RELATÓRIO *json* EXEMPLO

O *JSON* que o sistema guarda contém toda a informação preenchida pelo *frontend* e que é enviada para o *backed* que, por sua vez, a guarda em base de dados. Exemplo do *JSON* recebido :


```
{
    "_id" : ObjectId("60f8b0f77497ee64e956ff7b"),
    "key" : "ljdlashdi-lhikdsjfwq",
    "operation" : [
        " rea  1"
    ],
    "when" : [
        "2020-04-22 14:23:23"
    ],
    "body" : [
        {
            "area": "Opera  es",
            "responsavel": "hf29jk5idn",
            "medida": "Existe seguran a nos postos de trabalho?",
            "estadoMedida":"Em Curso",
            "notas": [
                {
                    "descricao": "Existem senhas, mas s o muito intuitivas"
                },
                {
                    "descricao": "Vai ser elaborada uma tarefa para definir senhas com os
                        crit rios de seguran a definido pelo CNCS."
                }
            ],
            "tarefas": [
                {
                    "descricao": "Espa o para adicionar tarefas a esta medida, tanto pode ser
                        vista pela medida como vai para uma tabela tarefas em que menciona a que
                         medida pertence e a que colaborador."
                }
            ],
            "anexos": [
                {
                    "descricao": "Relatorio anterior",
                    "url": "...."
                }
            ]
        },
        {
            "area": "Opera  es",
```

```
            "responsavel": "hf29jk5idn",
            "medida": "Possu  password nos computadores?",
            "estadoMedida":"Em Curso",
            "notas": [
            ],
            "tarefas": [
            ],
            "anexos": [
            ]
        },
        {
            "area": "Opera  es",
            "responsavel": "hf29jk5idn",
            "medida": "Obedece aos crit rios de seguran a da CNCS",
            "estadoMedida":"Em Curso",
            "notas": [
            ],
            "tarefas": [
            ],
            "anexos": [
            ]
        }
    ]
}
```

A.4 YAML

A.4.1 *API Login*

```
openapi: 3.0.1
info:
  title: Customer
  version: 0.8.0
externalDocs:
  description: Find out more about Swagger
  url: 'http://swagger.io'
servers:
  - url: /api/customers
tags:
  - name: Login
    description: ''
paths:
  /auth0:
```

```
    post:
      tags:
        - Login
      summary: Do login
      operationId: doLogin
      requestBody:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/LoginRequest'
        required: true
      responses:
        '200':
          description: OK
        '400':
          $ref: '#/components/responses/BadRequest'
        '404':
          $ref: '#/components/responses/NotFound'
        '500':
          $ref: '#/components/responses/InternalServer'
      x-codegen-request-body-name: body

components:
  schemas:
    LoginRequest:
      required:
        - username
        - password
      type: object
      properties:
        username:
          type: string
          description: Your customer name
          example: Jo o Paulo
        password:
          type: string
          description: Customer password
          example: deweesa232ed
    Error:
      type: object
      properties:
        errorCode:
          type: string
        message:
          type: string
  responses:
    BadRequest:
```

```
    description: Bad Request
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Error'
        example:
          errorCode: '400'
          message: 'The API request is invalid or improperly formed. Consequently, the API
              server could not understand the request.'
  NotFound:
    description: Not Found
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Error'
        example:
          errorCode: '404'
          message: The requested operation failed because a resource associated with the
              request could not be found.
  Forbidden:
    description: Forbidden
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Error'
        example:
          errorCode: '403'
          message: The requested operation is forbidden and cannot be completed.
  Gone:
    description: Gone
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Error'
        example:
          errorCode: '410'
          message: The request failed because the resource associated with the request has
              been deleted
  Duplicated:
    description: Duplicated
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Error'
        example:
          errorCode: '409'
```

```
            message: The requested operation failed because it tried to create a resource that
                already exists.
    InternalServer:
      description: Internal Server Error
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Error'
          example:
            errorCode: '500'
            message: The request failed due to an internal error.
    BadGateway:
      description: Bad Gateway
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Error'
          example:
            errorCode: '502'
            message: 'The server while working as a gateway to get a response needed to handle
                the request, got an invalid response.'
  securitySchemes:
    OAuth2:
      type: http
      bearerFormat: JWT
      scheme: bearer
```

A.4.2 *API's Relatórios*

```
openapi: 3.0.1
info:
  title: Report
  version: 0.8.0
externalDocs:
  description: Find out more about Swagger
  url: 'http://swagger.io'
servers:
  - url: /api
tags:
  - name: Report
    description: ''
paths:
  /report:
```

```
  post:
    tags:
      - Report
    summary: Create report
    operationId: createReport
    requestBody:
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Report'
      required: true
    responses:
      '200':
        description: OK
      '400':
        $ref: '#/components/responses/BadRequest'
      '404':
        $ref: '#/components/responses/NotFound'
      '500':
        $ref: '#/components/responses/InternalServer'
/report/{code}:
  get:
    tags:
      - Report
    operationId: getReport
    parameters:
      - name: code
        in: path
        required: true
        schema:
          type: string
    responses:
      '200':
        description: Founded
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Report'
      '400':
        $ref: '#/components/responses/BadRequest'
      '404':
        $ref: '#/components/responses/NotFound'
      '500':
        $ref: '#/components/responses/InternalServer'
  put:
    tags:
      - Report
```

```
      operationId: updateReport
      parameters:
        - name: code
          in: path
          required: true
          schema:
            type: string
      requestBody:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Report'
        required: true
      responses:
        '200':
          description: Updated
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Report'
        '400':
          $ref: '#/components/responses/BadRequest'
        '404':
          $ref: '#/components/responses/NotFound'
        '500':
          $ref: '#/components/responses/InternalServer'
      x-codegen-request-body-name: body
    delete:
      tags:
        - Report
      operationId: deleteReport
      parameters:
        - name: code
          in: path
          required: true
          schema:
            type: string
      responses:
        '204':
          description: Deleted
          content: {}
        '410':
          $ref: '#/components/responses/Gone'
        '500':
          $ref: '#/components/responses/InternalServer'

components:
```

```
schemas:
  Report:
    type: object
    properties:
      operation:
        type: string
        example: Area 1
      when:
        type: string
        example: '2020-04-22 14:23:23'
      body:
        type: object
        example: [{
          "area": "Opera es",
          "responsavel": "hf29jk5idn",
          "medida": "Existe seguran a nos postos de trabalho?",
          "estadoMedida":"Em Curso",
          "notas": [
              {
                  "descricao": "Existem senhas, mas s o muito intuitivas"
              },
              {
                  "descricao": "Vai ser elaborada uma tarefa para definir senhas com os
                      crit rios de seguran a definido pelo CNCS."
              }
          ],
          "tarefas": [
              {
                  "descricao": "Espa o para adicionar tarefas a esta medida, tanto pode ser
                      vista pela medida como vai para uma tabela tarefas em que menciona a que
                       medida pertence e a que colaborador."
              }
          ],
          "anexos": [
              {
                  "descricao": "Relatorio anterior",
                  "url": "...."
              }
          ]
      },
      {
          "area": "Opera es",
          "responsavel": "hf29jk5idn",
          "medida": "Possu password nos computadores?",
          "estadoMedida":"Em Curso",
          "notas": [
          ],
```

```
            "tarefas": [
            ],
            "anexos": [
            ]
        },
        {
            "area": "Opera es",
            "responsavel": "hf29jk5idn",
            "medida": "Obedece aos crit rios de seguran a da CNCS",
            "estadoMedida":"Em Curso",
            "notas": [
            ],
            "tarefas": [
            ],
            "anexos": [
            ]
        }]

    Error:
      type: object
      properties:
        errorCode:
          type: string
        message:
          type: string
  responses:
    BadRequest:
      description: Bad Request
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Error'
          example:
            errorCode: '4000'
            message: 'The API request is invalid or improperly formed. Consequently, the API
                server could not understand the request.'
    NotFound:
      description: Not Found
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Error'
          example:
            errorCode: '404'
            message: The requested operation failed because a resource associated with the
                request could not be found.
    Forbidden:
```

```
    description: Forbidden
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Error'
        example:
          errorCode: '403'
          message: The requested operation is forbidden and cannot be completed.
  Gone:
    description: Gone
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Error'
        example:
          errorCode: '410'
          message: The request failed because the resource associated with the request has
              been deleted
  Duplicated:
    description: Duplicated
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Error'
        example:
          errorCode: '409'
          message: The requested operation failed because it tried to create a resource that
              already exists.
  InternalServer:
    description: Internal Server Error
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Error'
        example:
          errorCode: '500'
          message: The request failed due to an internal error.
  BadGateway:
    description: Bad Gateway
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Error'
        example:
          errorCode: '502'
          message: 'The server while working as a gateway to get a response needed to handle
              the request, got an invalid response.'
```

```
securitySchemes:
  OAuth2:
    type: http
    bearerFormat: JWT
    scheme: bearer
```