



Universidade de Aveiro

Ano 2022/2023

**António Jorge
Ferreira Ramos**

**Validação da autenticidade de documentos
impressos**



Universidade de Aveiro
Ano 2022/2023

**António Jorge
Ferreira Ramos**

**Validação da autenticidade de documentos
impressos**

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática, realizada sob a orientação científica do Doutor André Zúquete, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

o júri

presidente

Prof. Doutor João Antunes da Silva
professor associado da Universidade de Aveiro

Prof. Doutor João Antunes da Silva
professor associado da Universidade de Aveiro

Prof. Doutor João Antunes da Silva
professor associado da Universidade de Aveiro

Prof. Doutor João Antunes da Silva
professor associado da Universidade de Aveiro

**agradecimentos /
acknowledgements**

Agradeço a ajuda dos orientadores, amigos e familiares no desenvolvimento da dissertação e constante apoio.

palavras-chave

Marca de água, códigos de barras, documentos impressos, documentos digitais, segurança, verificação de integridade

resumo

Cada vez mais, a segurança é um meio principal que permite proteger documentos de qualquer formato, levando à constante procura de novos métodos. Uma abordagem é a marca de água, que permite a proteção do documento ao adicionar algo visível ou invisível a este.

Nesta dissertação, serão abordados métodos e o desenvolvimento de um método de marca de água.

Neste caso servirá para verificação da integridade de um documento por meio de deteção visual de letras. Esse processo envolve a criação de retas e cálculo de interseções, identificando as letras do intervalo específico.

Além disso, para detetar a marca de água cria-se código de barras contendo informações sobre o documento, que será inserido no cabeçalho e rodapé numa cópia do documento original.

keywords

Watermark, barcodes, printed documents, digital documents, security, integrity

abstract

Increasingly, security is a key means of protecting documents in any format, leading to a constant search for new methods. One approach is watermarking, which enables document protection by adding something visible or invisible to it.

In this dissertation, methods and the development of a watermarking method will be addressed. In this case, it will serve to verify the integrity of a document through visual detection of letters. This process involves creating lines and calculating intersections, identifying the letters within a specific range.

Additionally, to detect the watermark, a barcode is created containing information about the document, which will be inserted in the header and footer of a copy of the original document

Índice

Índice.....	ii
Lista de tabelas.....	iii
Lista de Figuras.....	iv
1. Introdução.....	1
1.1. Objetivos.....	1
2. Estado da Arte.....	3
2.1. Código de barras.....	3
2.1.1. Análise de código de barras.....	3
2.2. Marca de água sobre texto.....	5
2.2.1. Substituição de palavras.....	5
2.2.1.1. Exemplo.....	6
2.2.1.2. Vantagens.....	6
2.2.1.3. Desvantagens.....	7
2.2.2. Marca de água sem mudança textual.....	7
2.2.2.1. Vantagens.....	7
2.2.2.2. Desvantagens.....	7
2.2.3. Documentos baseados em Eigenvalues.....	7
2.2.3.1. Testes.....	7
2.2.3.2. Vantagens.....	8
2.2.3.3. Desvantagens.....	8
2.2.4. Marca de água com espaçamento.....	9
2.2.4.1. Line-Shift Coding.....	9
2.2.4.2. Word-Shift Coding.....	9
2.2.4.3. Character Coding.....	9
2.2.4.4. Vantagens dos métodos.....	9
2.2.4.5. Desvantagens dos métodos.....	9
3. Tipos de códigos de barras.....	11
3.1. QR Code.....	11
3.2. Código de barras 128.....	12
4. Sistema de verificação de documentos impressos ou digitais.....	14
4.1. Arquitetura.....	14
4.2. Estrutura do documento.....	15
4.3. Base de dados.....	15
4.4. Criação do código de barras.....	16
4.5. Operações desempenhar pelo utilizador.....	16
4.6. Processamento do documento.....	17
4.7. Verificação do documento.....	19
4.8. Verificação de integridade.....	19
5. Descrição do funcionamento do algoritmo.....	22
6. Documentos em digital.....	28
7. Robustez do algoritmo.....	32
7.1. Mudança de escala no ficheiro.....	32
7.2. Alterações no ficheiro original.....	35
8. Conclusão.....	40
8.1. Futuro Trabalho.....	40
9. Referências.....	42
Anexo.....	43
A) Processamento de ficheiro PDF.....	43
B) ZXing.NET.....	45
C) Descodificação de posições dos códigos de barras.....	46
D) Posições aleatórias para marca de água.....	48
E) Mudança de escala do documento.....	50

Lista de tabelas

Tabela 1 - Estudo de código de barras lineares	4
Tabela 2 - Lista de códigos de barras ideias.....	5
Tabela 3 - Resultados Eigenvalues	8
Tabela 4 - Classificação de erros do QR Code	11

Lista de Figuras

Figura 1 - Nota 10€.....	5
Figura 2 - Exemplo ataque zero.....	6
Figura 3 - Antes da marca de água	8
Figura 4 - Depois da marca de água.....	8
Figura 5 - Comparação Marca de água	8
Figura 6 - Exemplo line-shift	9
Figura 7 - Exemplo word-shift	9
Figura 8 - Exemplo Character Coding	9
Figura 9 - Exemplo QR Code.....	11
Figura 10 - Estrutura QR Code.....	11
Figura 11 - Código de barra 128.....	12
Figura 12 - Tabela ASCII.....	12
Figura 13 - Arquitetura solução.....	15
Figura 14 - Diagrama da Base de dados.....	16
Figura 15 - Processamento exemplo	17
Figura 16 - Erro ao clicar em aceitar ou rejeitar sem antes de processar o ficheiro	17
Figura 17 - Reconhecimento caracteres	18
Figura 18 - Finalização do processamento	18
Figura 19 - Finalização da marca de água	18
Figura 20 - Mensagem Documento Aceite.....	18
Figura 21 - Mensagem documento já aceite.....	18
Figura 22 - Abrir Documento.....	18
Figura 23 - Verificação documento	19
Figura 24 - Algoritmo que conta número de retas que se interseitam	20
Figura 25 - Algoritmo interseção retas.....	20
Figura 26 - Exemplo verificação integridade	20
Figura 27 - Processamento diagrama de fluxo	23
Figura 28 - Documento exemplar sem marca de água	23
Figura 29 - Documento com marca de água.....	24
Figura 30 - Documento	24
Figura 31 - Código de barras.....	24
Figura 32 - Marca de água documento.....	24
Figura 33 - Obtenção de caracteres.....	24
Figura 34 - Representação dos valores dos caracteres	24
Figura 35 - Pontos para verificação de integridade.....	25
Figura 36 - Fluxograma Verificar.....	25
Figura 37 - Resultado Verificação.....	25
Figura 38 - Resultado Análise Forense	25
Figura 39 - Ficheiro digitalizado normal.....	26
Figura 40 - Resultado ficheiro digitalizado normal	26
Figura 41 - Ficheiro digitalizado torto	26
Figura 42 - Resultado scan torto.....	26
Figura 43 - Atualização da tabela watermark e criação da tabela “dimensions_document”.....	28
Figura 44 - Documento scan torto	29
Figura 45 - Documento scan composto.....	29
Figura 46 - Resultado verificação de integridade	30
Figura 47 - Resultado integridade documento normal	30
Figura 48 - Documento com 50% de escala	32
Figura 49 - Documento Original	33
Figura 50 - Resultado integridade do documento original	33
Figura 51 - Documento com 80% de escala	34
Figura 52 - Resultado integridade do documento com 80% de escala	34
Figura 53 - Documento Teste.....	35
Figura 54 - Código Inserção metados	36
Figura 55 - Resultado da verificação da integridade do documento normal.....	36
Figura 56 - Resultado da verificação da integridade do documento com substituição de palavra	37
Figura 57 - Resultado da verificação da integridade do documento com eliminação de texto	37
Figura 58 - Resultado da verificação da integridade do documento com eliminação de espaçamento	38
Figura 59 Código de extração dos caracteres num ficheiro PDF.....	43
Figura 60 - Obtenção dos valores.....	43
Figura 61 - Execução do ficheiro jar.....	44
Figura 62 - Exemplo posição de caracter no PDF	44
Figura 63 - Valor representativo da base de dados	44
Figura 64 - Gerar códigos de barras 128 e 39.....	45
Figura 65 - Leitura de código de barras 128.....	45
Figura 66 - Obtenção de posições do código de barras	46

Figura 67 - Guardar imagem auxiliar.....	46
Figura 68 - Guardar imagem auxiliar.....	47
Figura 69 - Representação dos pontos do código de barras.....	47
Figura 70 - Código para demonstra posições na figura.....	48
Figura 71 – Demonstração da criação dos pontos.....	49
Figura 72 – Espaçamento dos pontos.....	49
Figura 73 - Alterar escala do documento	50
Figura 74 – Cálculo da escala do documento	50
Figura 75 - Adaptar cálculo das posições.....	50

1. Introdução

A informação é um conhecimento necessário para o desenvolvimento e sobrevivência da sociedade. Classificando-se em diversas categorias, tais como, militar, saúde e política. Dentro destas, existem organizações responsáveis por entregar/receber/reencaminhar as informações entre diversas entidades, levando à criação de um caminho para a partilha da mesma, o que a torna vulnerável a acessos inéditos.

Estas podem ser privadas ou públicas, sendo que as privadas contêm informações confidenciais. Devido ao valor dado à informação, ou seja, à necessidade de conhecer algo que não sabemos, esta é alvo de ameaças - eliminação, alteração, divulgação e roubo. A partilha pode ser por via eletrónica (email), impressão e digitalização, sendo que maior parte das informações são guardadas online, levando à necessidade de pesquisa e inovação em métodos seguros capazes de proteger as informações.

Porém, apesar da implementação de soluções, estes documentos podem estar sujeitos ser impressos, o que leva a potenciais ameaças, nomeadamente, roubo e destruição.

O objetivo desta dissertação é encontrar e implementar métodos de marcação da informação de documentos, que permitam validar a sua autenticidade, como o utilizador, data/hora e o local em que o documento foi impresso.

1.1. Objetivos

A solução proposta tem de permitir uma validação rápida das informações do documento criar um mecanismo de marca de água capaz de validar as informações do documento mais profundamente, ou seja, determinar se o documento foi alterado e em que zona. Esta solução deve funcionar em documentos de formato eletrónico, digitalizado e impresso.

A solução deve ser dividida em dois processos diferentes, a criação da marca de água de um determinado ficheiro (sendo diferente para todos e para o mesmo) e a verificação do conteúdo da mesma.

Para validar rapidamente as informações do documento, optou-se por colocá-las num código de barras, onde só será possível decodificar a informação no local onde estará a base de dados, ou então um pedido à base de dados para obter os resultados na aplicação de verificação. Este código de barras irá ser colocado no cabeçalho para permitir a leitura do documento e garantir, maior parte das vezes a sua leitura.

Para a criação da marca de água optou-se por criar um QR Code, que contém informações mais detalhadas sobre o documento, como as características e algumas palavras do mesmo, escolhidas aleatoriamente para verificar a sua integridade.

2. Estado da Arte

Devido à enorme quantidade de código de barras e inúmeros métodos de marca de água, é necessário agregar e estudar os métodos para desenvolver a dissertação. Esta secção irá estar dividida em duas, onde em primeiro expõem-se os tipos de código de barras, escolhendo o ideal, e em segundo aborda-se tipo de marcas de água.

2.1. Código de barras

Código de barras é um modo de representar informação num estado visual, sem ter a necessidade de escrever texto. Classifica-se em duas categorias, linear (1D) e 2D.

Um dos requisitos da empresa era que o código de barras não ocupasse muito espaço no documento, tendo espaço livre no cabeçalho e no rodapé. Como a categoria linear ocupa menos espaço em altura, tendo alguns tipos comprimento variável influenciado pela informação contida no mesmo, é que se vai escolher para inserir no cabeçalho ou rodapé. Na categoria linear existem diversos códigos de barras, sendo que para analisar os mesmos optou-se por criar uma tabela (1), que compara os mesmos, escolhendo o ideal para inserir no documento. A tabela é constituída pelas colunas:

- Nome: designação do código de barras;
- Imagens exemplificativas;
- Comprimento - variável ou fixo;
- Uso;
- Necessidade de aparelho especial para ler.

Em alguns códigos de barras existe um espaço reservado que não se pode alterar que se chama *checksum* ou *check digit*, cujo objetivo é verificar se a informação do código de barras foi gerada corretamente.

Existem várias maneiras de decodificar códigos de barras, salientando-se as seguintes:







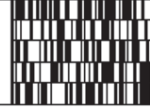















1. Máquina própria (leitor de barras de barras)
2. Smartphone (iOS ou android) com uma aplicação que permita a leitura.

2.1.1. Análise de código de barras

Esta subsecção abrange a análise de vários códigos de barras lineares, cujos parâmetros de escolha foi comprimento (variável ou fixo), tamanho das barras. A tabela 1 apresenta um conjunto de códigos de barras com as características previamente expostas.




Na tabela 2 encontram-se os possíveis códigos de barras que porventura se possam utilizar. Estes códigos foram selecionados com o critério tamanho máximo de caracteres que este pode guardar, podendo mudar no futuro de acordo conforme escolha da empresa. A informação que se pretende guardar nos códigos de barras é a data do documento assinado, o local do posto de trabalho e um identificador único do documento, podendo ser mudado no futuro.

Tabela 1 - Estudo de código de barras lineares

Nome	Imagem	Fixo/ Variável	Tamanho das barras	Uso
Post Code Austrália	 12229111A B A 9	Fixo	4	Correios Austrália
CodaBar	 3 1117 01320 6375	Fixo	2	Librarias
Code 25	 0 1 2 3 4 5 6 7 8 9	Variável	2	Librarias
Code 11	 0123452	Fixo	2	Telefones, já não se usa muito devido à sua antiguidade
Code 32	 A012345678	Fixo	2	Farmácia
Code 39	 CODE39	Fixo	2	Vários
Code 49		Variável	Vários	Vários
Code 93	 CODE93	Variável	Vários	Vários
Code 128	 CODE128	Variável	Vários	Vários
EAN 2	 1550 0317-8910 05	Variável	Vários	Revistas
EAN 5	 9 770317-847001 54495	Variável	Vários	Livros
EAN-8	 9 781565-924798	Variável	Vários	Retalho
GS1-128	 (05)95812345678901 (10)9000123	Variável	Vários	Vários
GS1 Databar	 (01)00075678164125	Variável	Vários	Vários
Intelligent Mail barcode	 Wikimedia Foundation Inc. PO BOX 78350 SAN FRANCISCO CA 94107-8350	Fixo	4	Correios USA
ITF-14	 3 47 28432 10321 3	Variável	2	Encomendas
ITF-6	 123457	Variável	2	Vários
JAN	 5 901234 123457 >	Variável	Vários	Usado no Japão
Planet		Variável	Grande/ Pequeno	Correios USA
Plessey	 4 3 2 1 6 Start 43216 Check Digit Stop	Variável	2	Catálogos, revistas, inventários
PostBar		Fixo	4	Correios Canada
PostNET		Fixo	Grande/ Pequeno	Correios USA

RM4SCC/K IX	 Lloyds TSB Bank plc CREDIT CARD SERVICES BOX 1 BX1 1LT	Fixo	4	Correios
RM Mailmark C		Fixo	4	Correios
Universal Product Code	 9 8 7 6 5 4 3 2 1 0 9 8	Variável	Vários	Retalho
Telepen	 ABC-abc-1234	Variável	2	Librarias da Inglaterra

Tabela 2 - Lista de códigos de barras ideias

Nome	Imagem	Tamanho máximo de dados	Tipos de dados	Vantagens	Desvantagens
Code 39	 CODE 39	43	Letras e números	Ocupa menos espaço.	Não permite caracteres especiais tal como ç.
Code 128	 Wikipedia Schematic of a barcode (Code 128B). 1:quiet zone, 2:start code, 3:data, 4:checksum, 5:stop code	48	Letras, números e caracteres especiais	É mais eficiente a codificar texto Tem <i>checksum</i>	Precisa de espaço reservado no início, fim e <i>check symbols</i> .
Code 93	 CODE93	30	Letras, números e caracteres especiais	Open Source	Não tem checksum

2.2. Marca de água sobre texto

A marca de água é um método que permite dificultar a alteração de conteúdo de informação do documento, e assim proteger a autenticidade. Atualmente existe em diversos setores, o mais comum é nas notas europeias, exemplo nota 10€ (figura 1), onde o objetivo é autenticar a nota e não permitir que haja falsificação das mesmas [1].



Figura 1 - Nota 10€

2.2.1. Substituição de palavras

O documento quando está em circulação em formato papel ou eletrónico pode ser alvo de fuga informação ou acesso ao documento onde pessoas não autorizadas conseguem alterar o texto (inserir, reordenar, substituir e remover), levando à mudança de conteúdo e da estrutura do texto do mesmo.

A solução proposta pelos autores do artigo [2], envolve focar a marca de água na localização das palavras no texto. Contudo a reordenação do texto é o maior desafio para esta marca de água sobre o texto. A marca de água sobre o texto proposta envolve selecionar certas palavras no documento e substituir por novas, mantendo o resto da informação intacta no final da criação da mesma. Esta implementação é constituída por dois elementos:

1. Lista de palavras que se pretende substituir no documento.
2. Lista de palavras contidas no documento depois da substituição.

Em suma, escolhe-se um conjunto de palavras no documento que se quer substituir por novas. Quando uma palavra é selecionada para se substituir todas as ocorrências da mesma são substituídas pela palavra nova. A percentagem do ataque determina a quantidade de modificação do documento. Se a palavra escolhida tiver várias ocorrências no documento, a percentagem vai aumentar.

A percentagem da contagem de palavras no documento é dada pela equação 1. A percentagem do ataque de substituição é dada pela expressão da equação 2.

A lista de substituição pode ser escolhida através de uma lista de palavras normal e avançada. A normal consiste em seleccionar palavras aleatórias do documento. A avançada selecciona palavras com mesmo tamanho do que as palavras da lista de substituição.

Em suma, a marcação de água textual proposto por [2] consiste em 3 passos, seleccionar palavras do documento para substituição, seleccionar as novas palavras e ocorrência de implementação no texto do documento.

$$WOR(x) = \frac{\text{Number of Occurrence of } x \text{ in Document}}{\text{Total Number of Words in Document}}$$

Equação 1

$$WSR = \frac{\sum WOR(x)}{\text{Total Number of Words in Document}}$$

Equação 2

2.2.1.1. Exemplo

Conforme mencionado no artigo [2] a figura 2, demonstra um exemplo de categorias de substituição que podem ser usados para mascarar informações importantes num texto. O exemplo demonstra a substituição da palavra "The", sendo o pronome mais comum em inglês, por "close" na substituição normal e por "job" na substituição avançada.

Este método pode ser útil em várias situações, como na proteção de informações sensíveis em documentos, comunicações confidenciais, entre outros. No entanto, é importante notar que este método não é infalível.

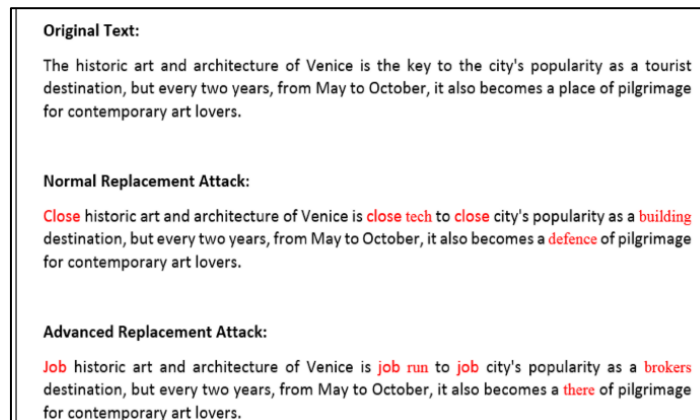


Figura 2 - Exemplo ataque zero

2.2.1.2. Vantagens

Uma das vantagens do método de substituição descrito é permitir a alteração de palavras num texto por um utilizador fidedigno e, em simultâneo, permitir que o algoritmo saiba em qual zona do documento as alterações foram feitas.

Isso pode ser útil em situações em que uma equipa de revisão precisa fazer alterações num documento, como num processo de edição de textos. Ao usar esse método, a equipa de revisão pode fazer as alterações necessárias sem perder informações importantes ou prejudicar a integridade do documento. Além disso, o algoritmo pode identificar facilmente as alterações realizadas e rastreá-las para fins de auditoria e verificação de alterações autorizadas.

2.2.1.3. Desvantagens

Uma das principais desvantagens é que as mudanças no texto podem não fazer sentido para pessoas que não estão familiarizadas com a substituição de palavras usada. Isso pode tornar a leitura do texto difícil e confusa, especialmente se muitas palavras forem substituídas. Se a substituição for realizada de forma excessiva, o texto pode-se tornar ininteligível.

Além disso, o método de substituição pode não ser capaz de detetar métodos de cópia de texto, como copiar e colar o texto num novo documento. Se o novo documento for criado com base no texto original antes da substituição, as informações confidenciais podem ser utilizadas sem que a substituição tenha qualquer efeito.

Outra limitação é que esse método de substituição não garante a segurança completa dos dados, pois é possível que alguém com conhecimento suficiente possa decifrar a substituição e utilizar as informações originais.

2.2.2. Marca de água sem mudança textual

A necessidade de comparar o método de marca de água com e sem mudança textual, surgiu para escolher o melhor, levando à implementação dos autores do artigo [3] que consiste usar as características do documento para gerar a marca de água, seguindo o layout “*autor:marcadeágua:data:tempo*”, onde esta, é registada por uma autoridade certificada (CA). O principal objetivo é usar palavras que tenham um comprimento superior a quatro caracteres. Esta escolha adveio de a generalidade da mudança textual por parte de pessoas não autorizadas em palavras menores do que quatro caracteres.

Para gerar a marca de água, é necessário ter acesso ao conteúdo do documento, e assim percorrer cada palavra, encontrado as palavras superiores a quatro caracteres, guardando o primeiro carácter de cada palavra dando. No fim, junta-se todas as marcas de água dando a final.

Por exemplo, na seguinte frase: “O José gosta muito de ler. A Ana não.” a marca de água final seria JGM (José Gosta Muito).

2.2.2.1. Vantagens

Tem uma entidade certificadora que possui a marca de água original, cuja é a única pessoa que pode comparar os documentos, permitindo assim que o documento esteja só na posse de uma pessoa, diminuindo os ataques ou distribuição do documento. A composição da entidade certificadora é similar ao pretendido no código de barras.

2.2.2.2. Desvantagens

Tem testes realizados com ataques aleatórios, como inserir, apagar, reordenar e alterar, que para textos reais os resultados poderão ser diferenciados, levando com que o comportamento do algoritmo seja diferenciado entre documentos, dificultando a sua implementação para a Verificação da integridade do documento.

2.2.3. Documentos baseados em Eigenvalues

O artigo [4] apresenta um algoritmo que armazena as posições de todas as palavras de um documento em uma matriz, juntamente com os seus pesos em ASCII. Um documento é normalmente constituído por palavras, espaçamento, números e pontuações. Os autores consideram cada ocorrência para calcular o peso ASCII e gerar um esquema de marca de água com base numa chave privada, cujo um utilizador fidedigno verifica o documento recebido.

2.2.3.1 Testes

No artigo [4] existe testes realizados a um documento com 47 linhas, não tendo disponibilizado o mesmo, apenas impressões de pequenas secções do mesmo.

A figura 3 demonstra um excerto do texto antes de ter a marca de água, e na figura 4 demonstra o texto depois da criação da marca de água. Para testar a eficiência da mesma, foi alterada a palavra “*OFF*” para “*ON*” numa zona do texto. Os resultados binários na figura 5, sendo que em primeiro está o documento original e a seguir o alterado, retira-se que os números são diferentes. Uma pequena alteração da palavra levou à mudança de 11 bits.

Os autores realizaram testes na mudança de vogais, consoantes, caracteres especiais, palavras, números, pontuações, e alterações aleatórias no texto, obtendo os resultados da tabela 3, podendo Verificar que na coluna “*tamper detection*” está tudo a 100% que leva a Verificar que o algoritmo detetou alterações em todos os casos.

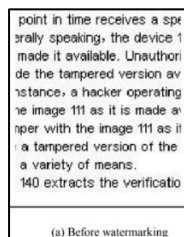


Figura 3 - Antes da marca de água

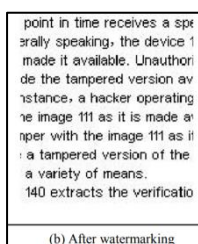


Figura 4 - Depois da marca de água

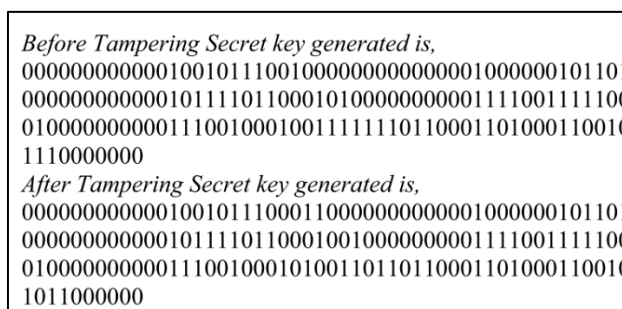


Figura 5 - Comparação Marca de água

Tabela 3 - Resultados Eigenvalues

Type of alteration (a single character)	Average eigen value shift	% bit change in secret key	Tamper Detection
Vowels	68.89	12.29	100%
Consonants	66.83	11.67	100%
Words	95.22	14.38	100%
Numerals	169.28	13.96	100%
Punctuations	53.94	10	100%
Random Alterations	127.95	15.63	100%

2.2.3.2 Vantagens

Guarda a posição das palavras do texto numa matriz. Tem uma identidade certificadora que guarda a marca de água e é a única pessoa que pode verificar se o documento é original ou falsificado.

2.2.3.3 Desvantagens

Difícil de compreensão, a execução do algoritmo demora bastante tempo.

2.2.4 Marca de água com espaçamento

O artigo [5] propôs 3 métodos de espaçamento para a criação de marca de água, sendo eles o espaçamento de linhas (line-shift), espaçamento de palavra (word-shift) e espaçamento de letras (character). Para a realização da verificação da marca de água é necessário OCR (Optical Character Recognition), já que as modificações são difíceis de visualizar com o olho humano.

2.2.4.1 Line-Shift Coding

Consiste em deslocar as linhas de texto de um documento para cima ou para baixo, enquanto as linhas adjacentes não são movidas. Na figura 6, demonstra-se um exemplo. Neste exemplo a linha do meio começada por “Effects...”, foi transferida para baixo 1/300 inches, que equivale a 0.00846666667 cm.

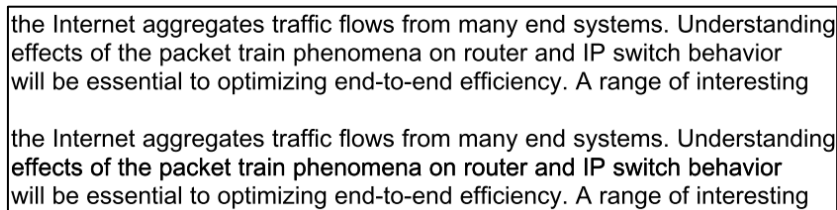


Figura 6 - Exemplo line-shift

2.2.4.2 Word-Shift Coding

Trata-se em mover as palavras para a esquerda ou para a direita, enquanto as palavras adjacentes não são alteradas. A figura 7, tem-se um exemplo. A segunda linha, contém quatro palavras movidas com espaçamento de 1/150 polegadas, que equivale a 0.0169333333 cm, enquanto na primeira linha não se altera, a terceira é uma junção das duas anteriores.

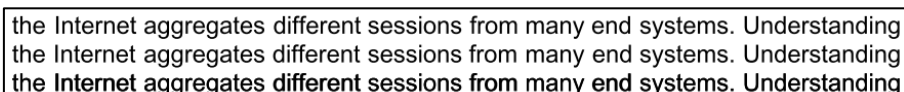


Figura 7 - Exemplo word-shift

2.2.4.3 Character Coding

A letra escolhida é movida para cima ou para baixo, enquanto as adjacentes não se alteram. A figura 8, apresenta um exemplo. A primeira letra “e” da palavra “internet” foi movida para baixo 1/600 inches que equivale a 0.00423333333 cm.



Figura 8 - Exemplo Character Coding

2.2.4.4 Vantagens dos métodos

Tem marca de água invisível, ou seja, não é perceptível para o olho humano.

2.2.4.5 Desvantagens dos métodos

Uso de OCR para verificar documento, e ter acesso ao texto para inserir o espaçamento. Erros de impressão, por exemplo faltar letras no documento, pode levar à má classificação da integridade de um documento.

3. Tipos de códigos de barras

3.1. QR Code

QR Code (figura 9) surgiu em 1994 pela empresa Denso Wave, originalmente com o propósito de categorizar peças de automóveis. Os QR Code podem ter links para páginas web, texto, um endereço geográfico, uma imagem, um vídeo ou contacto telefónico.

É formado por:

- 3 quadrados de detecção de posição (4.1. Figura 10) que permite a descodificação em várias posições do scanner ou de um smartphone com câmara;
- padrão alinhamento (4.2. Figura 10) que corrige a distorção do QR Code em superfícies curvadas, o seu número varia consoante a informação contida;
- padrões de temporização (4.3. Figura 10) que permite obter o tamanho da matriz de dados;
 - versão, que indica a versão do QR Code que está a ser utilizada (1. Figura 10);
- formato, onde agrega informações sobre a tolerância de erros e o padrão da máscara de dados;
- códigos de dados e erros que podem ser do tipo L, M, Q, H (apresentados na tabela 4).



Figura 9 - Exemplo QR Code

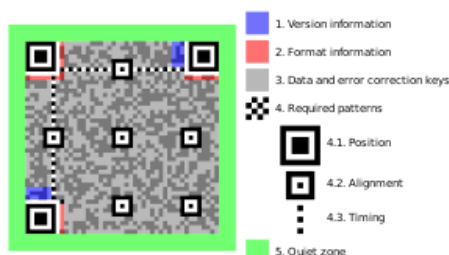


Figura 10 - Estrutura QR Code

Tabela 4 - Classificação de erros do QR Code

Nível de correção de erros	Percentagem de área danificada (%)
L (Low) – Baixo	7
M (Medium) – Médio	15
Q (Quartile) - Quartil	25
H (High) – Alto	30

3.2. Código de barras 128

O código de barras 128 (figura 11) [6] é um tipo de código de barras linear que pode ser usado para codificar uma grande variedade de dados alfanuméricos, incluindo letras, números e caracteres especiais. É capaz de codificar 128 caracteres ASCII (figura 12).

Este é muito utilizado em aplicações de logística, como no controlo de estoques e na identificação de produtos em supermercados e lojas. Isso ocorre porque ele é capaz de codificar informações como o nome do produto, seu número de série, o código de barras do fabricante e outras informações relevantes num único código.

É constituído por barras largas e compactas que representam cada caracter, com barras de início e fim que indicam onde começa e termina a sequência de caracteres. Ele é capaz de codificar uma abundância de informação num espaço relativamente pequeno, o que o torna uma ferramenta eficiente e económica para a gestão de inventário e outras aplicações similares.

É um código de barras universalmente reconhecido e amplamente utilizado em todo o mundo, o que leva ser uma opção confiável para empresas que necessitam de um sistema de identificação e rastreamento de produtos rápido e preciso.

Como referido inicialmente, o intuito da dissertação é validação de documentos impressos, para isso é necessário guardar informações acerca do documento em algum lado, para a verificação do mesmo. Como o código de barras consegue agregar informação, como, por exemplo, um identificador único de um documento onde, por fora esteja uma base de dados que contenha informação do documento, optou-se por utilizar como medida de Verificação inserindo-o no rodapé ou cabeçalho do documento.

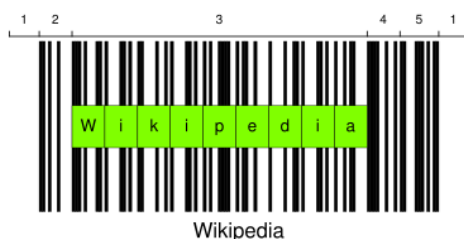


Figura 11 - Código de barra 128

ASCII Table															
Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(72	48	110	H	104	68	150	h
9	9	11		41	29	51)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	
29	1D	35		61	3D	75	=	93	5D	135]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Figura 12 - Tabela ASCII

4. Sistema de verificação de documentos impressos ou digitais

A primeira intenção para a criação da solução era usar QR Code, contudo em ficheiros digitalizados estes podiam desvanecer, ou seja, perder a cor, o que influencia a visualização do documento.

Para ultrapassar este problema, realizaram-se tentativas, tais como:

- processamento de imagem, que reconhece os QR Code no texto (mediante de substituição de cores, por exemplo, a cor preta presente dentro da cor branca do QR Code é uma zona critica, para corrigir é necessário alterar o preto para branco), descartada devido ao elevado tempo de processamento da mudança de cores;
- posições aleatórias no documento, descartada devido a porventura o QR Code ser colocado num sítio não permitido, como, por exemplo, debaixo de uma imagem, contudo sempre há oportunidade de gerar de novo a marca de água do ficheiro;
- quantidade de QR Code a colocar no documento;
- dimensão do QR Code.

Já que a leitura do QR Code no texto era impossível, optou-se por criar pontos nos quadrados de posição do mesmo para a criação de retas, com o intuito de calcular o ponto de interseção e se este tivesse dentro de uma letra guardar como marca de água. Porventura, a proposta só dava uso aos quadrados de posição do QR Code, sendo que removeu os mesmos e continuou-se a traçar os pontos, aumentando a integridade e a entropia do documento.

Em suma, a proposta do sistema final para verificação de documentos impressos é feita pelos seguintes passos:

- 1) Identificação de letras e suas respetivas posições no documento;
- 2) Criação dos pontos para traçamento das retas;
- 3) Calcular o ponto de interseção;
- 4) Verificar se o ponto de interseção pertence a algum intervalo de letra;
- 5) Guardar os pontos na base de dados;

4.1. Arquitetura

Conforme descrito anteriormente é criada a arquitetura da figura 13, onde surge 3 camadas.

- Servidor: destinado à base de dados que irá conter dados armazenados acerca dos ficheiros, códigos de barras, segmentos de reta traçados, posição dos caracteres no ficheiro de input e a geração da marca de água.
- Programa: onde acontece o desenvolvimento da criação do código de barras, do processamento do ficheiro para obtenção das posições dos caracteres no documento, da verificação do documento, e verificação de integridade, esta também é responsável pelas conexões entre camadas, desencadeadas pelo utilizador.
- Utilizador: responsável por agregar as ações que o utilizador pode fazer como, processar o documento ou verificar.

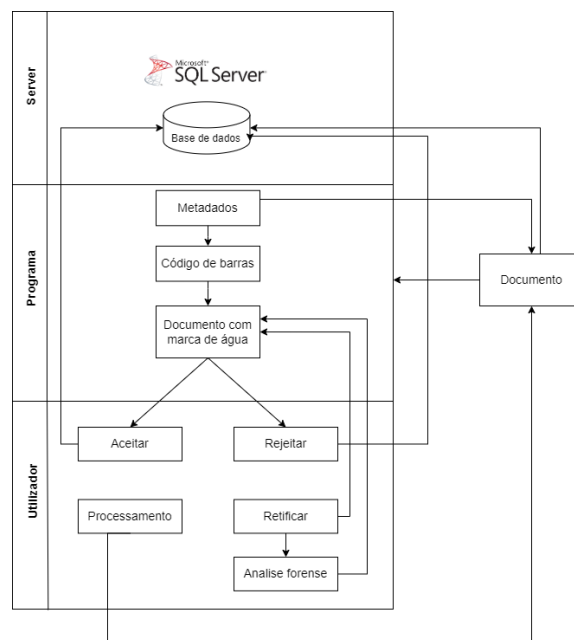


Figura 13 - Arquitetura solução

4.2. Estrutura do documento

Para a utilização de testes do programa foi disponibilizado pela empresa 4 documentos diferentes em formato PDF eletrónico. Cabe à empresa inserir manualmente os ficheiros na diretoria correspondente na inicialização do programa e os metadados no programa, ou adaptar o método para receber os metadados da base de dados da empresa, para tratar os mesmos. É de salientar que os documentos disponibilizados têm uma classificação de segurança mínima pelo que se pode partilhar os resultados do mesmo.

O algoritmo lida com três tipos de documentos:

- Documento sem marca de água;
- Documento com marca de água;
- Documento digitalizado com marca de água;

4.3. Base de dados

O intuito de usar base de dados no sistema é garantir segurança sobre as informações do documento através do uso de código de barras, como se referiu anteriormente, o código de barras tem um id aleatório representante das características do documento cujo é inserido na base de dados aquando do seu processamento do documento para a criação da marca de água.

Existem dois tipos de base de dados as relacionais e não relacionais (NoSQL). [7]

As bases de dados relacionais guardam os dados nas tabelas, tendo algumas delas partilha de informação, causando uma relação entre tabelas.

Cada tabela contém colunas que definem a informação que se pode guardar, e linhas que contém a informação.

Normalmente a tabela contém um identificador único que referencia cada linha denominada chave primária (primary key), caso se queira referenciar os valores a outra tabela utiliza-se a chave estrangeira (foreign key), que obrigatoriamente tem de existir previamente.

A linguagem que se usa para tratar base de dados relacionais é SQL, existem vários programas que permitem correr SQL, como, por exemplo, MySQL [8], Oracle SQL Developer [9] e Microsoft SQL Server Management Studio 2018 [10].

As bases de dados não relacionais, não usam tabelas relacionais, em vez disso faz criar grupos em que guarda a informação em informações diferentes.

Como o objetivo do trabalho é sempre perceber que documento é qual, é necessário haver relações entre a marca de água e o documento, então optou-se por usar uma base de dados relacional com o auxílio da ferramenta Microsoft SQL Server Management Studio 2018.

A base de dados SQL foi criada localmente, tendo um utilizador e base de dados, que permite a utilização do mesmo do algoritmo. As informações que se guardam são as características do documento, o código de barras, segmentos de reta traçados entre pontos, posições dos caracteres no documento de entrada (processamento apenas), e a criação da marca de água (aceitação ou rejeição).

Na figura abaixo, está presente um diagrama da base de dados que contém as tabelas usadas e respetivas conexões.

O diagrama é constituído pelas seguintes tabelas:

- “document”: guarda características do documento (metadados);
- “barcode”: guarda informações relativas ao documento;
- “watermark”: guarda as confirmações do documento com marca de água, se foi aceite ou não, para efeitos de rastreamento;
- “forense_analises”: guarda os segmentos de reta traçados entre dois pontos, o ponto de interseção e a letra que aparece no ponto de interseção para efeitos de verificação de integridade do documento;
- “position_char_file”: guarda as posições dos caracteres no documento.

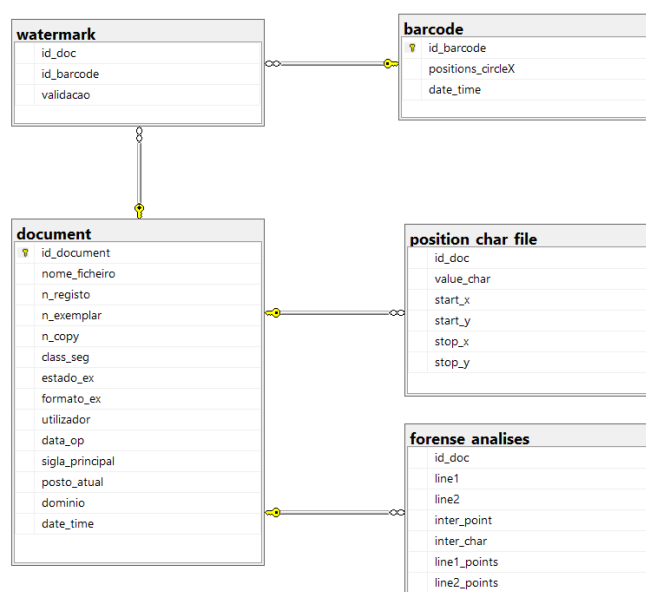


Figura 14 - Diagrama da Base de dados

4.4. Criação do código de barras

Para a criação do código de barras 128 foi utilizado um package C# denominado ZXing.net, abordado mais detalhadamente em Anexo.

Como referido anteriormente o código de barras irá ter um id referenciador para o documento e para as posições utilizadas para traçar os segmentos de reta no documento.

O código de barras 128, podendo o seu tamanho ou tipo ser alterado futuramente, irá ser colocado sempre no cabeçalho da primeira página por ser uma zona livre do documento, e permitir maior parte das vezes a sua leitura, caso falhar o utilizador é informado do insucesso da descodificação.

4.5. Operações desempenhar pelo utilizador

Um utilizador exerce 4 tipos de funções.

- Escolher entre
 - Processamento: que processa o documento sem a marca de água para originar a marca de água;
 - Verificação: que retifica o documento de input com a marca de água para verificar se é o documento que se apresenta ser.
- Aceitar ou rejeitar o documento com marca de água;

- Comparar a informação do documento com a marca de água no submenu verificar, para determinar a sua autenticidade;
- Verificação de integridade, para determinar a zona da alteração do documento, se por venturar o documento não ser autêntico, através da comparação de informações que aparecem na aplicação versus o documento.

4.6. Processamento do documento

Conforme mencionado anteriormente o utilizador escolhe o ficheiro a processar, o algoritmo só aceita caso o ficheiro não tiver na base de dados e se o nome do mesmo não conter “watermark” que se adiciona ao nome do ficheiro já existente para diferenciar os ficheiros sem e com marca de água, também se adicionou o dia e a hora que se executou o processamento do ficheiro levando ao nome final do ficheiro ser “nome_ficheiro_watermark_dd_mm_yy_hh_m_ss”, sendo dd:dia, mm:mês, yy:ano, hh:hora, m:min, ss:segundos. Antes de processar, caso o utilizador queira visualizar se o ficheiro selecionado é pretendido, o algoritmo mostra o ficheiro numa janela nova com os botões processar, aceitar e rejeitar (figura 15). Caso o utilizador clique em aceitar ou rejeitar o ficheiro sem antes de clicar “processar” aparece o erro da figura 16.

Consoante o clique “processar” o programa vai abrir uma consola que vai executar um ficheiro jar, fechando automaticamente depois da execução, que vai retirar os caracteres do ficheiro e as respetivas posições no documento. Para retirar as posições o algoritmo foi realizado em Java porque em C# não foi encontrado packages que retirassem a posição certas dos caracteres no documento (figura 17).

O processamento demora à volta de 2 min, sendo que é variável segundo o tipo de documento, ou seja, a informação que tiver nele, colocando a janela do processamento bloqueada até que as ações de calcular os pontos, obtenção dos pontos de interseção e letras, inserções na base de dados sejam concluídas.

Quando o processamento acaba o utilizador é abordado com uma mensagem da figura 18 para aceitar ou rejeitar o documento que se pode pré-visualizar presente na figura 20. Caso o documento seja aceite o utilizador recebe a mensagem “Documento Aceite!” figura 20, porventura se tentar aceitar de novo o documento, é informado de que o documento já foi aceite (figura 21). Caso rejeite, o documento é guardado, e o processamento é feito novamente. Caso feche a aplicação sempre poderá consultar o documento gerado para a diretoria que abriu o documento na extensão previamente exposta. A escolha é feita no menu principal quando se escolhe a opção pretendida. É aberto a diretoria predefinida da execução (figura 22) e pode-se mudar consoante o necessário, tendo a noção que ficheiros fora da pasta do algoritmo podem dar origem a erros.



Figura 15 - Processamento exemplo

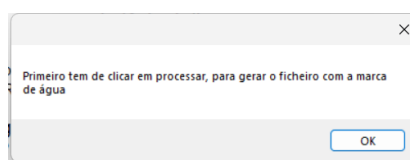


Figura 16 - Erro ao clicar em aceitar ou rejeitar sem antes de processar o ficheiro

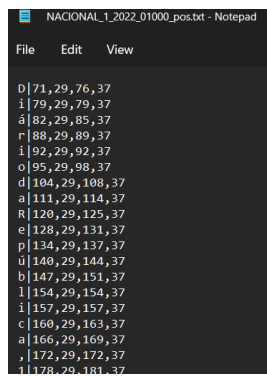


Figura 17 - Reconhecimento caracteres

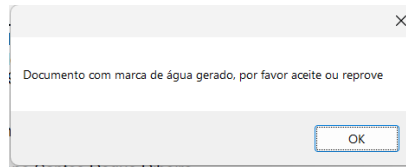


Figura 18 - Finalização do processamento

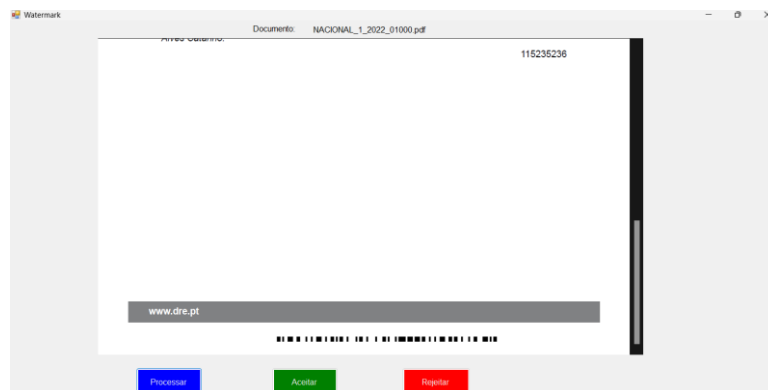


Figura 19 - Finalização da marca de água

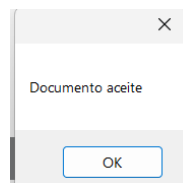


Figura 20 - Mensagem Documento Aceite

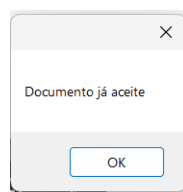


Figura 21 - Mensagem documento já aceite

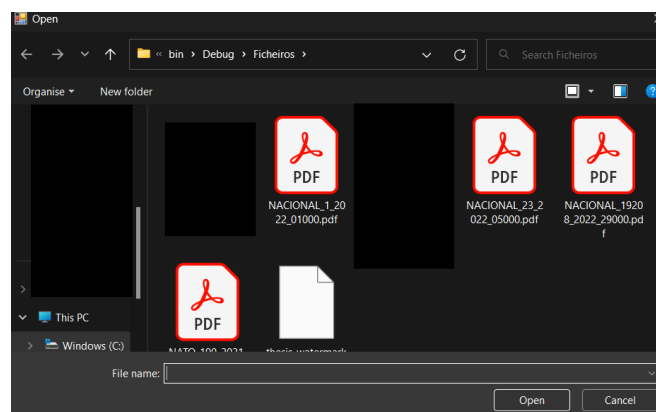


Figura 22 - Abrir Documento

4.7. Verificação do documento

Para Verificar um documento é necessário que o documento escolhido para Verificação contenha a marca de água e cuja informação esteja na base de dados. Caso esteja, o programa, tenta ler o código de barras 128, se ler com sucesso, mostra as informações contidas para o utilizador numa janela onde é possível ter a visualização do documento com a marca de água escolhido e algumas informações sobre as características do documento (metadados) (figura 23). Cabe ao utilizador averiguar se o documento e as características demonstradas são diferentes ou iguais, se determinar que são diferentes o utilizador pode proceder à verificação de integridade do documento, referido no ponto 6, determinando as zonas alteradas no documento.



Figura 23 - Verificação documento

4.8. Verificação de integridade

Antes de explicar a metodologia por detrás da verificação de integridade é necessário explicar um conceito fundamental denominada entropia, em que se baseia o algoritmo.

A entropia de um documento é uma medida de incerteza contida nele, ou seja, serve para avaliar a segurança de um algoritmo ou sistema criptográfico.

Quanto maior for a entropia maior é dificuldade para a pessoa/hacker descobrir como um algoritmo funciona.

Para atingir esse objetivo é necessário ocultar informações sobre a verificação da integridade e o modo como se gera a marca de água para um documento e desenvolver métodos aleatórios.

A verificação de integridade ou análise forense, serve para Verificar se o documento foi alterado, e em que zonas.

A solução para Verificar um documento, é criar 3 segmentos de retas calculados através dos pontos criados nas extremidades da folha 9 (3 × 3) pontos, totalizando 81 (9 × 9) pontos.

Para saber o número total de retas usou-se a expressão somatória do Gauss [11] (equação 1). O resultado da equação dá igual ao valor no código através do sublinhado a preto na figura 24.

$$n \times \frac{n(n-1)}{2}, n = 9 \leftrightarrow 9 \times \frac{9(9-1)}{2} = 9 \times \frac{9 \times 8}{2} = 9 \times \frac{72}{2} = 9 \times 36 = 324$$

Equação 1 - Gauss

Concluído o traçamento das retas calcula-se o ponto de interseção da mesma [12], com a expressão da figura 25, a seguir com base nas coordenadas dos caracteres obtidos no processamento do ficheiro verifica-se se o ponto pertence ao subconjunto das coordenadas, dando como output a letra correspondida.

Na figura 26 mostra um possível output da verificação da integridade, sendo o círculo amarelo o ponto de interseção e a azul o carácter que está na base de dados, extraído no processamento, sendo que estas apresentações podem ser mudadas no código futuramente, foi escolhida um círculo para ser mais fácil de visualização do que só ter um pixel.

```

62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
foreach (KeyValuePair<string, Point> entry in qrcode_points)
{
    string[] val0 = entry.Key.Split('.');
    foreach (KeyValuePair<string, Point> entry2 in qrcode_points)
    {
        string[] val1 = entry2.Key.Split('.');
        if (!val0[0].Equals(val1[0]))
        {
            qrcode_comb = entry.Key + ":" + entry2.Key;
            if (!combs.Contains(entry2.Key + ":" + entry.Key))
            {
                combs.Add(qrcode_comb);
            }
        }
    }
}

Console.WriteLine("Rects without duplicates " + combs.Count);

int without_p = 1;

```

Output

Show output from: Debug

'WatermarkApp.exe' (CLR v4.0.30319: WatermarkApp.exe): Loaded 'C:\WINDOWS\Microsoft.Net\assembly\GAC_MSIL\PresentationFramework\v4.0.0.0\PresentationFramework.dll'.

'WatermarkApp.exe' (CLR v4.0.30319: WatermarkApp.exe): Loaded 'C:\WINDOWS\Microsoft.Net\assembly\GAC_MSIL\System.Xaml\v4.0.0.0\System.Xaml.dll'.

'WatermarkApp.exe' (CLR v4.0.30319: WatermarkApp.exe): Loaded 'C:\WINDOWS\Microsoft.Net\assembly\GAC_MSIL\SMDiagnostics\v4.0.0.0\SMDiagnostics.dll'.

'WatermarkApp.exe' (CLR v4.0.30319: WatermarkApp.exe): Loaded 'C:\WINDOWS\Microsoft.Net\assembly\GAC_MSIL\System.ServiceModel.Internet\v4.0.0.0\System.ServiceModel.Internet.dll'.

The thread 0x9e7c has exited with code 0 (0x0).

Rects without duplicates 324

The thread 0x9e80 has exited with code 0 (0x0).

The thread 0x9358 has exited with code 0 (0x0).

Figura 24 - Algoritmo que conta número de retas que se interseitam

```

float x1 = A.X;
float x2 = B.X;
float x3 = C.X;
float x4 = D.X;

float y1 = A.Y;
float y2 = B.Y;
float y3 = C.Y;
float y4 = D.Y;
float t = (float)((x1 - x3) * (y3 - y4) - (y1 - y3) * (x3 - x4)) / ((x1 - x2) * (y3 - y4) - (y1 - y2) * (x3 - x4));
float u = (float)((x1 - x3) * (y1 - y2) - (y1 - y3) * (x1 - x2)) / ((x1 - x2) * (y3 - y4) - (y1 - y2) * (x3 - x4));

Point inter = new Point();
if ((t >= 0 && t <= 1) && (u >= 0 && u <= 1))
{
    int x = Convert.ToInt16(x3 + u * (x4 - x3));
    int y = Convert.ToInt16(y3 + u * (y4 - y3));
    inter = new Point(x, y);
}

```

Figura 25 - Algoritmo interseção retas

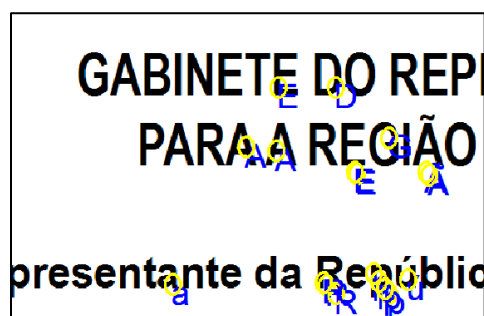


Figura 26 - Exemplo verificação integridade

5. Descrição do funcionamento do algoritmo

Esta secção visa a demonstração do funcionamento do algoritmo criado num documento exemplar fornecido pela empresa.

Como referido anteriormente existem dois tipos de ações que se pretende desenvolver no algoritmo, processamento de um documento sem marca de água para originar a marca de água e Verificação de um documento com marca de água, sendo ele em formato eletrónico ou pelo ficheiro digitalizado.

O algoritmo para o processamento segue a lógica do fluxograma da figura 27, descrito abaixo:

1. Escolha do ficheiro PDF a processar, no exemplo optei por escolher o documento da figura 28 denominado “NACIONAL_1_2022_0100”;
2. Verificação da escolha do documento. Caso tiver sido processado o utilizador irá ser abordado por uma mensagem a informar o sucedido e vai permitir uma nova escolha do documento. Porventura se não existir o algoritmo vai passar a inserção dos metadados do ficheiro na base de dados (figura 30) para o processamento do ficheiro, onde irão ser retirados os caracteres e as posições do mesmo;
3. Obtenção de caracteres e posições do ficheiro. O algoritmo irá abrir uma consola para executar um comando java para ler o ficheiro PDF e retirar as posições e os caracteres do mesmo. As posições são constituídas por, começo e o fim da letra em X e Y, seguindo a lógica da figura 34, colocando os valores na base de dados (figura 33);
4. Criação de pontos para traçamento de retas e cálculo do ponto de interseção (abordado detalhadamente em anexo). Utilizando as 9 posições pré-definidas calcula-se as 3 posições, com espaçamento fixo, por cada ponto para a criação da reta. Para finalizar calculam-se os pontos de interseção das retas e retifica-se se estes calhem dentro do intervalo de valores dos caracteres, guardando os valores na base de dados (figura 35);
5. Criação do código de barras. Dependendo do número do id do documento que irá ser colocado na base de dados (figura 30) e das posições dos pontos das retas (figura 31 - coluna posições), o código de barras irá ser diferente (figura 29 - código de barras);
6. Adição do código de barras no ficheiro original e averiguação do ficheiro pelo utilizador, que é feita através da visualização do documento criado com a marca de água, sendo necessária sua aprovação ou rejeição. Caso o utilizador rejeite o documento, este é colocado como rejeitado na base de dados e é gerado um novo processamento. Caso aceite o documento é guardado na base de dados (figura 32 - onde 1 significa aceite) e é dado a finalização do processamento.

Por outro lado, a Verificação segue a lógica do fluxograma da figura 36, descrito abaixo:

1. Escolha do ficheiro a Verificar, neste caso optei pelo documento de saída do passo anterior, presente na figura 29;
2. Leitura do código de barras para descodificar a informação contida nele, se o id do ficheiro tiver na base de dados e tiver sido aceite ou rejeitado a marca de água o algoritmo vai devolver alguns metadados do ficheiro (figura 37);
3. Averiguação de informações, isto é, comparar informações obtidas pelo código de barras com o documento (figura 37);
4. Verificação de integridade do documento caso a verificação der errado. Consiste na comparação das letras que estão no documento com as da base de dados (figura 38). Através da análise da figura 38, verificou-se que o algoritmo demonstra várias letras na mesma zona, o que levou à posteriori, remover duplicados;
5. Finalização da Verificação do ficheiro selecionado.

Existindo a possibilidade de os documentos estarem em formato digital, criou-se um algoritmo que vai tratar o documento em formato digitalizado do tipo direito de uma forma e torto doutra, tendo este de estar na base de dados. Para exemplificar digitalizou-se o mesmo documento com orientações diferentes um com a folha direita (figura 39) e outro com a folha torta (figura 41), sendo que o algoritmo consegue detetar o código de barras num ficheiro digitalizado direito (figura 40) e realizar a

verificação de integridade (figura 40), cujos pontos estão desfasados do original, sendo necessária uma correção no algoritmo, contudo para o ficheiro digitalizado torto não consegue, dando erro de insucesso de leitura (figura 42).

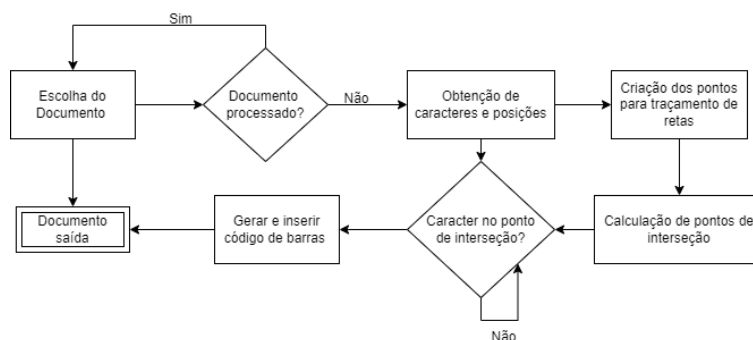


Figura 27 - Processamento diagrama de fluxo

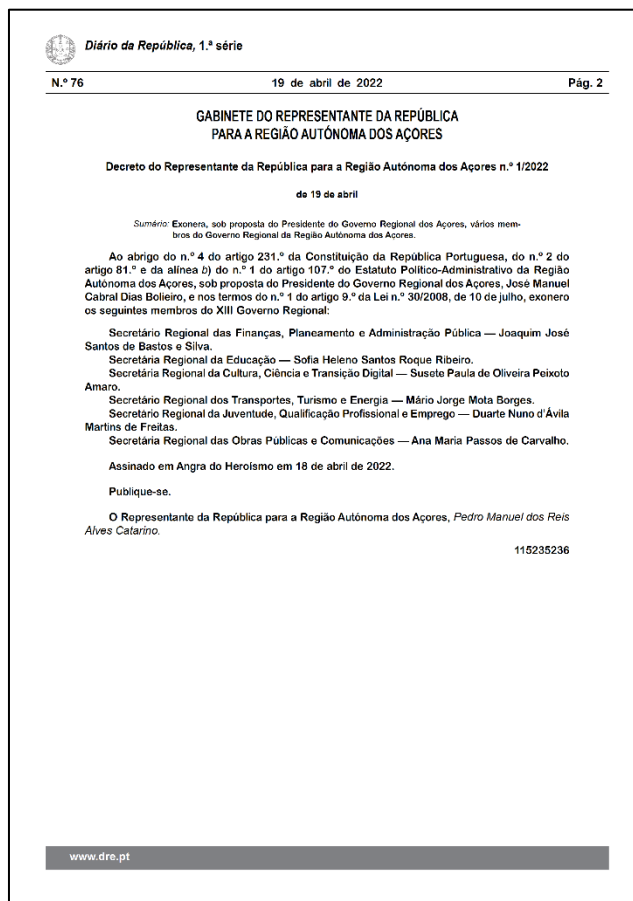


Figura 28 - Documento exemplar sem marca de água

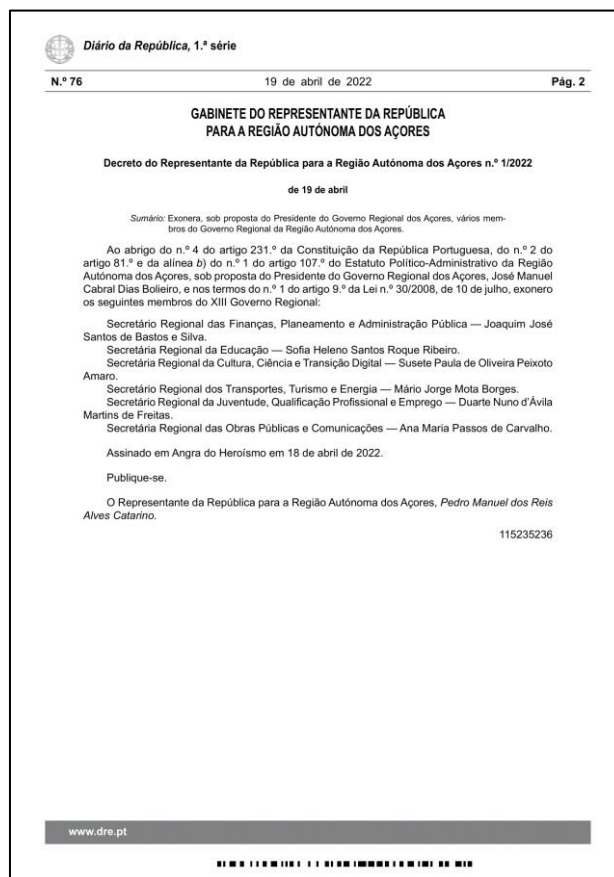


Figura 29 - Documento com marca de água

id_documento	nome_ficheiro	n_registo	n_exemplar	n_copys	class_seg	estado_ex	formato_ex	utilizador	data_op	sigla_principal	posto_atual	dominio
1770408244	NACIONAL_1_2022_01000	1/2022/01000	1	0	S	Arquivado	Eletronico	João Francisco	31/01/2022 15:01:35	Decreto do Representante da República para os Aç...	Registo Central	NACIONAL

Figura 30 - Documento

id_barcode	positions_circleX
1	50,50 297,50 525,50 50,421 297,421 525,421 50,817 297,817 525,817

Figura 31 - Código de barras no documento

id_doc	id_barcode	validacao
1770408244	1	1

Figura 32 - Marca de água do documento

id_doc	value_char	start_x	start_y	stop_x	stop_y
1	1770408244	71	29	76	37
2	1770408244	79	29	79	37
3	1770408244	82	29	85	37
4	1770408244	88	29	89	37
5	1770408244	92	29	92	37
6	1770408244	95	29	98	37
7	1770408244	104	29	108	37
8	1770408244	111	29	114	37
9	1770408244	120	29	125	37
10	1770408244	128	29	131	37
11	1770408244	134	29	137	37
12	1770408244	140	29	144	37
13	1770408244	147	29	151	37
14	1770408244	154	29	154	37
15	1770408244	157	29	157	37
16	1770408244	160	29	163	37

Figura 33 - Obtenção de caracteres do documento

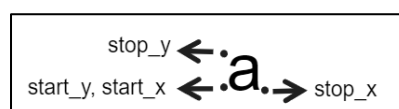


Figura 34 - Representação dos valores dos caracteres

id_doc	line1	line2	inter_point	inter_char	line1_points	line2_points
1770408244	point1_l.point5_l	point2_l.point4_b	767,874	G	268,123:1297,1669	1297,123:148,1754
1770408244	point1_l.point5_l	point2_l.point7_r	936,1126	s	268,123:1297,1669	1297,123:148,3318
1770408244	point1_l.point5_l	point2_l.point8_r	1242,1586	ã	268,123:1297,1669	1297,123:1177,3318
1770408244	point1_l.point5_l	point2_l.point8_b	1243,1588	ã	268,123:1297,1669	1297,123:1177,3403
1770408244	point1_l.point5_l	point2_b.point7_r	893,1063	p	268,123:1297,1669	1177,208:148,3318
1770408244	point1_l.point5_l	point2_b.point8_l	1229,1567	ç	268,123:1297,1669	1177,208:1297,3318
1770408244	point1_l.point5_l	point3_r.point4_l	930,1118	s	268,123:1297,1669	2127,123:268,1669
1770408244	point1_l.point5_l	point3_r.point7_b	1243,1587	ã	268,123:1297,1669	2127,123:148,3403
1770408244	point1_l.point5_r	point2_b.point7_l	891,1183	G	268,123:1177,1669	1177,208:268,3318
1770408244	point1_l.point5_r	point3_l.point4_l	890,1182	G	268,123:1177,1669	2247,123:268,1669
1770408244	point1_l.point5_r	point3_l.point4_b	888,1178	G	268,123:1177,1669	2247,123:148,1754
1770408244	point1_l.point5_r	point3_r.point4_r	853,1118	r	268,123:1177,1669	2127,123:148,1669
1770408244	point1_l.point5_r	point3_b.point4_l	889,1180	G	268,123:1177,1669	2127,208:268,1669
1770408244	point1_l.point5_r	point3_b.point4_b	887,1176	G	268,123:1177,1669	2127,208:148,1754
1770408244	point1_l.point5_b	point2_l.point4_l	736,964	4	268,123:1177,1754	1297,123:268,1669
1770408244	point1_l.point5_b	point2_l.point7_r	893,1245	F	268,123:1177,1754	1297,123:148,3318
1770408244	point1_l.point5_b	point3_r.point7_b	840,1166	L	268,123:1177,1754	1177,123:148,3403

Figura 35 - Pontos para verificação de integridade

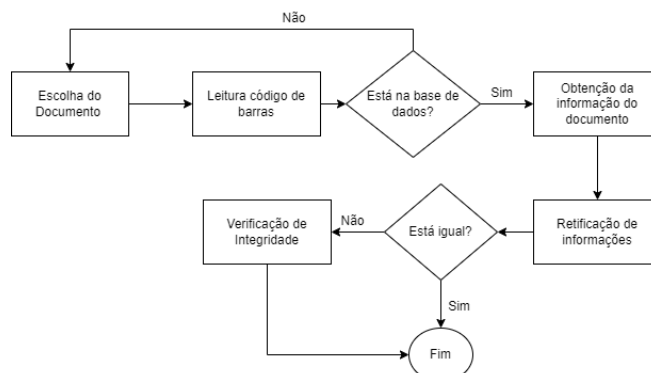


Figura 36 - Fluxograma Verificar

Retificar

Diário da República, 1.ª série

N.º 76 19 de abril de 2022 Pág. 2

**GABINETE DO REPRESENTANTE DA REPÚBLICA
PARA A REGIÃO AUTÓNOMA DOS AÇORES**

Decreto do Representante da República para a Região Autónoma dos Açores n.º 1/2022

de 19 de abril

Sumário: Exonera, sob proposta do Presidente do Governo Regional dos Açores, vários membros do Governo Regional da Região Autónoma dos Açores.

Ao abrigo do n.º 4 do artigo 231.º da Constituição da República Portuguesa, do n.º 2 do artigo 81.º e da alínea b) do n.º 1 do artigo 107.º do Estatuto Político-Administrativo da Região Autónoma dos Açores, sob proposta do Presidente do Governo Regional dos Açores, José Manuel Cabral Dias Rolaim, e nos termos do n.º 1 do artigo 9.º da Lei n.º 30/2008, de 10 de julho, exonero

Documento: NACIONAL_1_2022_01000

Utilizador: João Francisco

Sigla: Decreto do Representante da República para os Açores

Posto Atual: Registo Central

Análise Forense

Figura 37 - Resultado Verificação

AnáliseForenseForm

Diário da República, 1.ª série

N.º 76 19 de abril de 2022 Pág. 2

**GABINETE DO REPRESENTANTE DA REPÚBLICA
PARA A REGIÃO AUTÓNOMA DOS AÇORES**

Decreto do Representante da República para a Região Autónoma dos Açores n.º 1/2022

de 19 de abril

Sumário: Exonera, sob proposta do Presidente do Governo Regional dos Açores, vários membros do Governo Regional da Região Autónoma dos Açores.

Ao abrigo do n.º 4 do artigo 231.º da Constituição da República Portuguesa, do n.º 2 do artigo 81.º e da alínea b) do n.º 1 do artigo 107.º do Estatuto Político-Administrativo da Região Autónoma dos Açores, sob proposta do Presidente do Governo Regional dos Açores, José Manuel Cabral Dias Rolaim, e nos termos do n.º 1 do artigo 9.º da Lei n.º 30/2008, de 10 de julho, exonero os seguintes membros do Governo Regional:

Secretário Regional das Finanças, Planeamento e Administração Pública — Joaquim José Santos de Santos e Silva.

Secretária Regional da Educação — Sofia Heleno Santos Roque Ribeiro.

Secretária Regional da Cultura, Ciência e Transição Digital — Susete Paula de Oliveira Peixoto Amaro.

Secretário Regional dos Transportes, Turismo e Energia — Mário Jorge Mota Borges.

Secretário Regional da Juventude, Qualificação Profissional e Emprego — Duarte Nuno d'Ávila Martins de Freitas.

Figura 38 - Resultado Análise Forense



Figura 39 - Ficheiro digitalizado normal

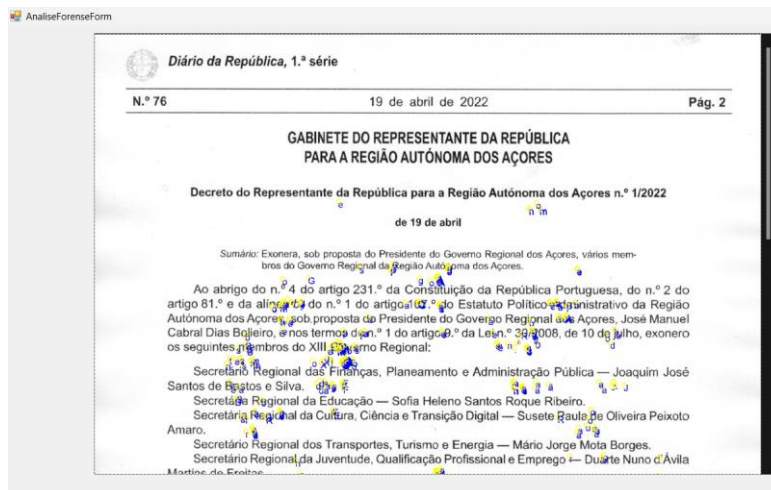


Figura 40 - Resultado ficheiro digitalizado normal

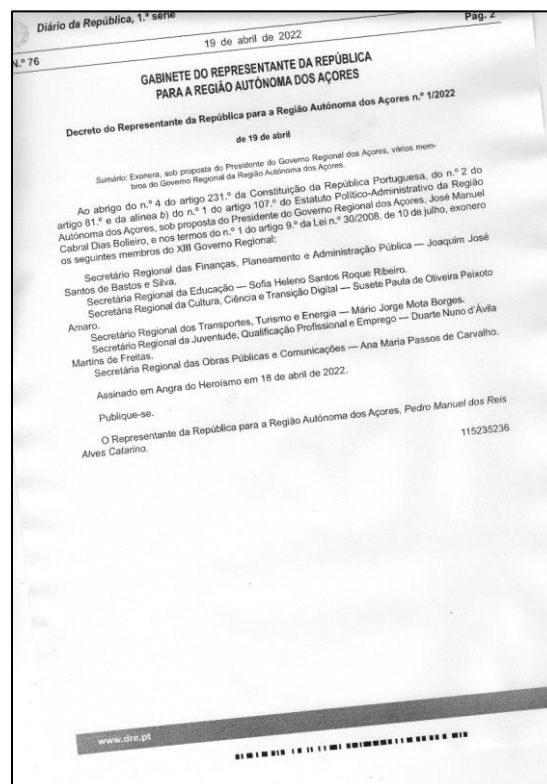


Figura 41 – Ficheiro digitalizado torto

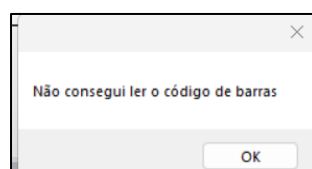


Figura 42 - Resultado scan torto

6. Documentos em digital

Na secção anterior, abordou-se um teste sobre um documento digitalizado que falhou. Caso a digitalização tivesse sido bem-sucedida, as letras estariam no sítio certo. Isto levou a pensar numa estratégia alternativa para compor o documento digitalizado que desse entrada na Verificação para garantir a sua Verificação.

A estratégia envolve alterar a origem dos pontos pré-definidos para serem calculados com base na posição do código de barras no documento e guardar as posições na base de dados do documento original. A tabela responsável por guardar esses valores é “*watermark*” que tem informações se o documento com marca de água foi ou não aceite (figura 43), bem como as posições dos códigos de barras.

Numa digitalização é possível que um documento seja digitalizado torto, levando à necessidade de adaptar o algoritmo.

A metodologia usada envolve calcular o ângulo da imagem e rodar o angulo contrário para endireitar o documento, substituindo o documento torto que deu entrada no sistema.

Para garantir o cálculo das novas posições do ficheiro digitalizado estejam corretas no eixo do x e y, criou-se outro código barras do tipo 39 cujo tem a mesma informação do 128, só que o intuito deste é calcular a proporção do y e quanto o documento andou no eixo do y, para calcular o x usa-se o código de barras 128. Porventura verificou-se que certas impressoras quando realizam digitalizações o ficheiro tem dimensões diferentes do original(595x842) e é necessário calcular a diferença para ajustar as posições.

A figura 44, demonstra um documento digitalizado relativamente torto, cujo dá entrada no sistema, depois de algum processamento de imagem o resultado apresenta-se na figura 45, mostrando que o algoritmo conseguiu endireitar o documento, contudo pode existir casos que a folha esteja muito torta e o algoritmo não funciona, caso isto aconteça cabe ao utilizador fazer uma digitalização nova.

Através da comparação da figura 46 do resultado da verificação de integridade com o original (figura 47), pode-se afirmar que a discrepância entre pontos é muito pouco sendo a percentagem de eficácia alta, é de salientar que devido à falta de impressoras, só se realizou os testes na impressora Brother MFC7460DN, sendo que os resultados poderão ser diferentes para outras impressoras.

A visualização de integridade foi melhorada relativamente a inicial, retirando pontos repetidos, e a adição de uma cruz para a detenção melhor do ponto de interseção.

```
CREATE TABLE watermark(  
  id_doc INT FOREIGN KEY REFERENCES document(id_document),  
  id_barcode INT FOREIGN KEY REFERENCES barcode(id_barcode),  
  validacao INT, -- 0 reject, 1 accept  
  x INT, -- start x position barcode  
  y INT, -- start y position barcode  
  x2 INT, -- end x position barcode  
  y2 INT, -- end y position barcode  
  x_39 INT,  
  y_39 INT,  
  x2_39 INT,  
  y2_39 INT  
);  
  
CREATE TABLE dimensions_document(  
  id_doc INT FOREIGN KEY REFERENCES document(id_document),  
  width int,  
  height int,  
  width_bmp int,  
  height_bmp int  
);
```

Figura 43 - Atualização da tabela watermark e criação da tabela “dimensions_document”

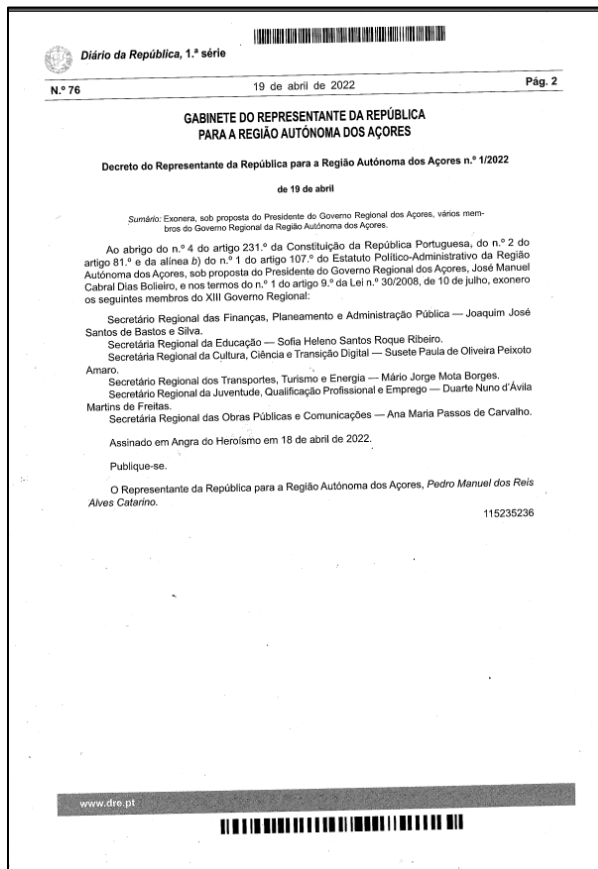


Figura 44 - Documento scan torto

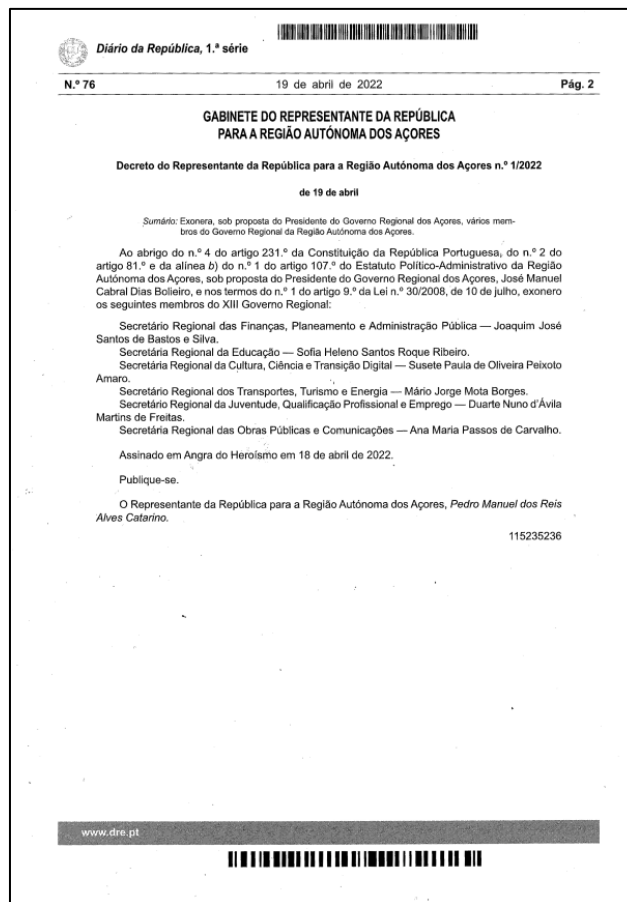


Figura 45 - Documento scan composto

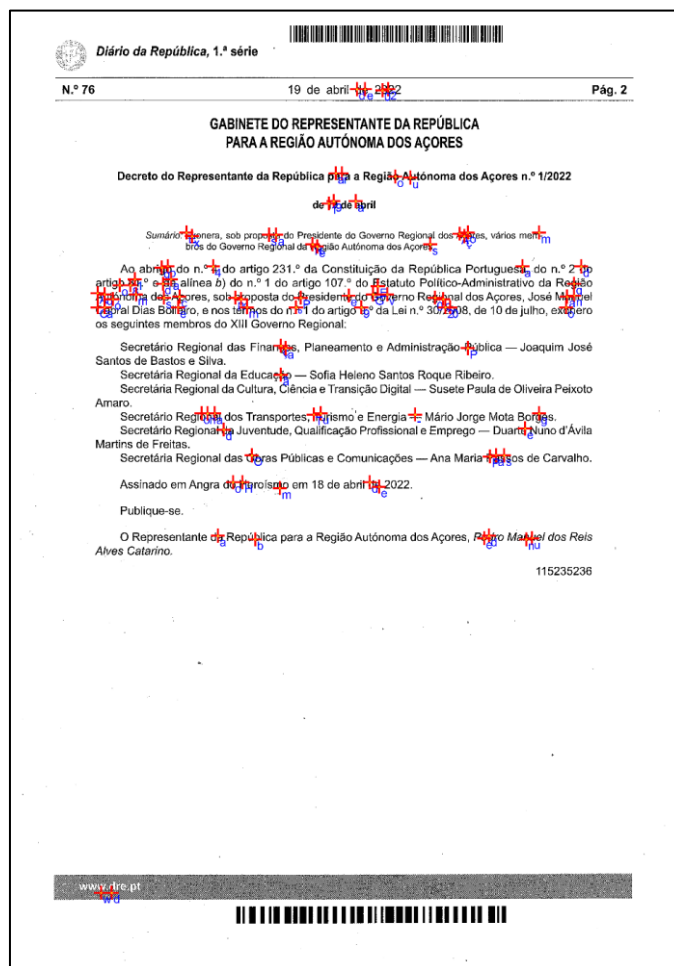


Figura 46 - Resultado verificação de integridade

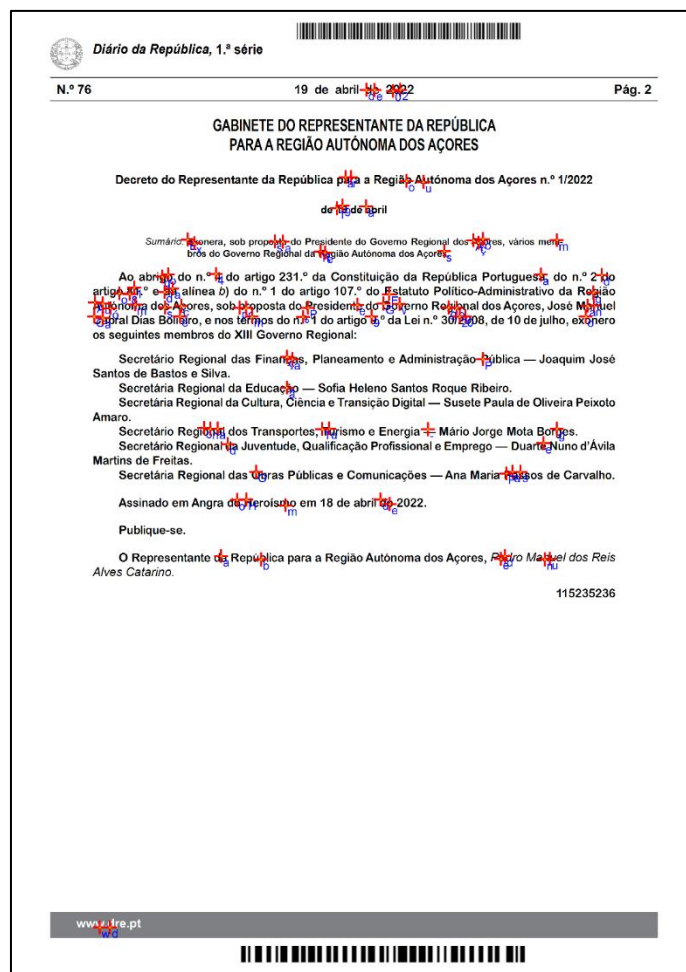


Figura 47 - Resultado integridade documento normal

7. Robustez do algoritmo

Para desenvolver um algoritmo é necessário garantir que ele funcione em múltiplos casos, ou seja, que seja robusto. O algoritmo desenvolvido envolve operações em ficheiros que podem ser alvo de alterações como, por exemplo, mudança de escala do ficheiro para impressão e a folha pode ser colocada torta no digitalizador, sendo que esta é abordada na secção anterior. Nesta secção abordará-se o comportamento do algoritmo em situações de mudança de escala e alteração de letras no ficheiro original.

7.1. Mudança de escala no ficheiro

A escala ideal para testar o algoritmo é entre 80-95%, sendo que menor de 50% o ficheiro torna-se ilegível a olho nu (figura 50), sendo necessário o uso de ferramenta de leitura de PDF para realizar zoom.

O ficheiro utilizado para alteração da escala está apresentado na figura 51, tendo o resultado da verificação na figura .

Para alterar a escala do ficheiro utilizou-se o package iTextSharp do C# (abordado mais detalhadamente em anexo).

NA figura 53 apresenta-se uma captura de ecrã do ficheiro da figura 51, com uma escala de 80%, sendo que o resultado obtido da verificação se apresenta na figura 52. Através da comparação de pontos entre a verificação da integridade da figura 52 e 50, conclui-se que o algoritmo conseguiu adaptar as posições no ficheiro com escala variável.

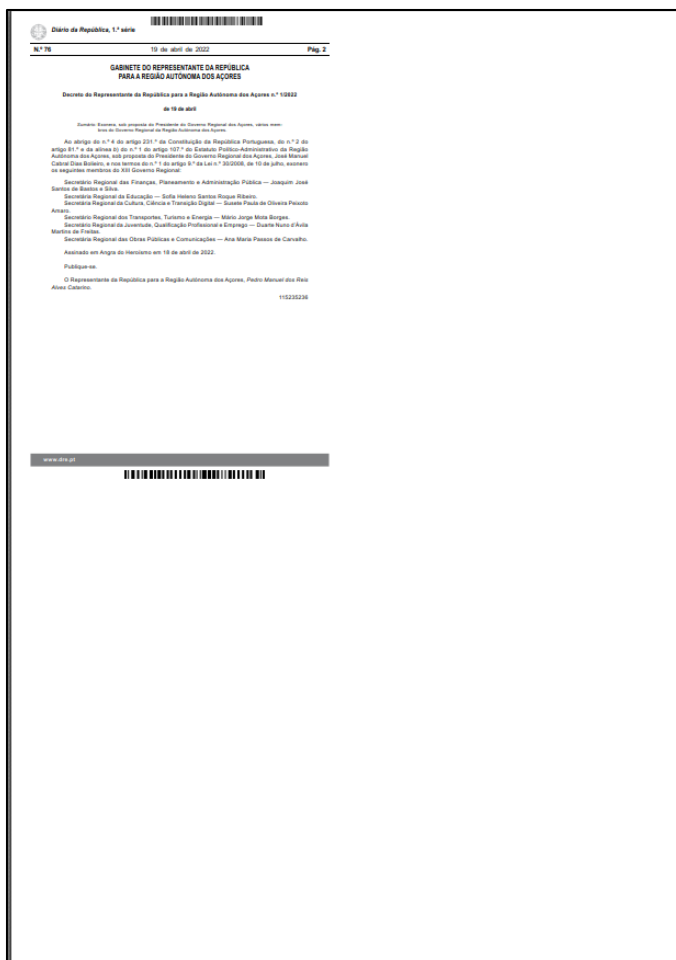


Figura 48 - Documento com 50% de escala

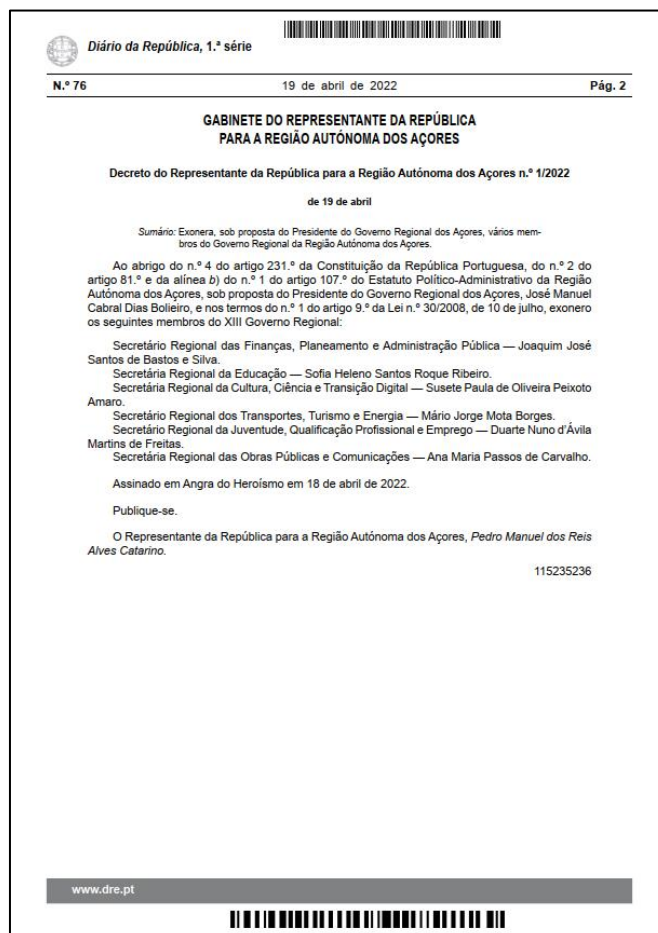


Figura 49 - Documento Original

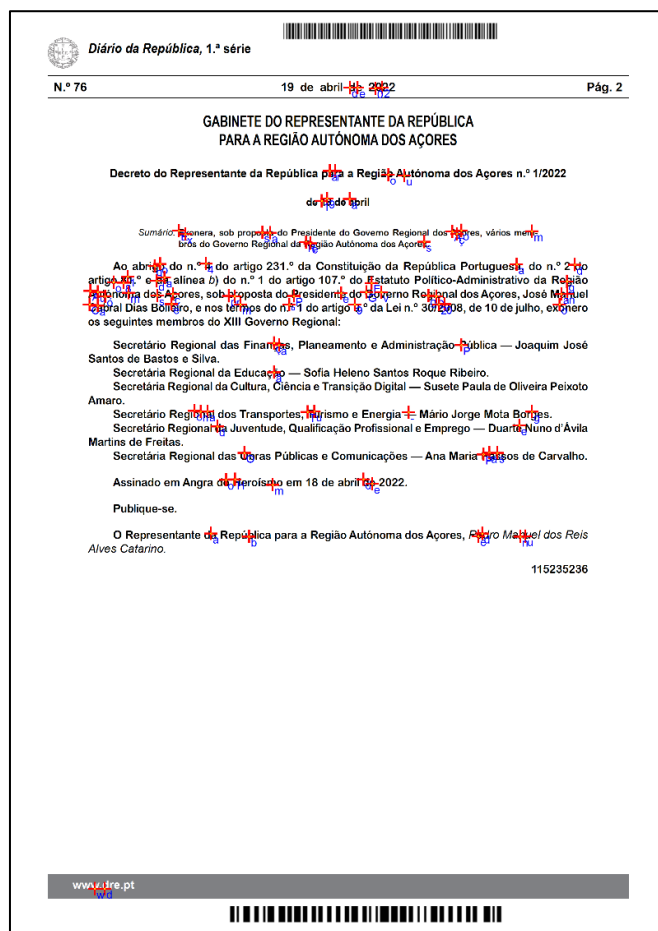


Figura 50 - Resultado integridade do documento original

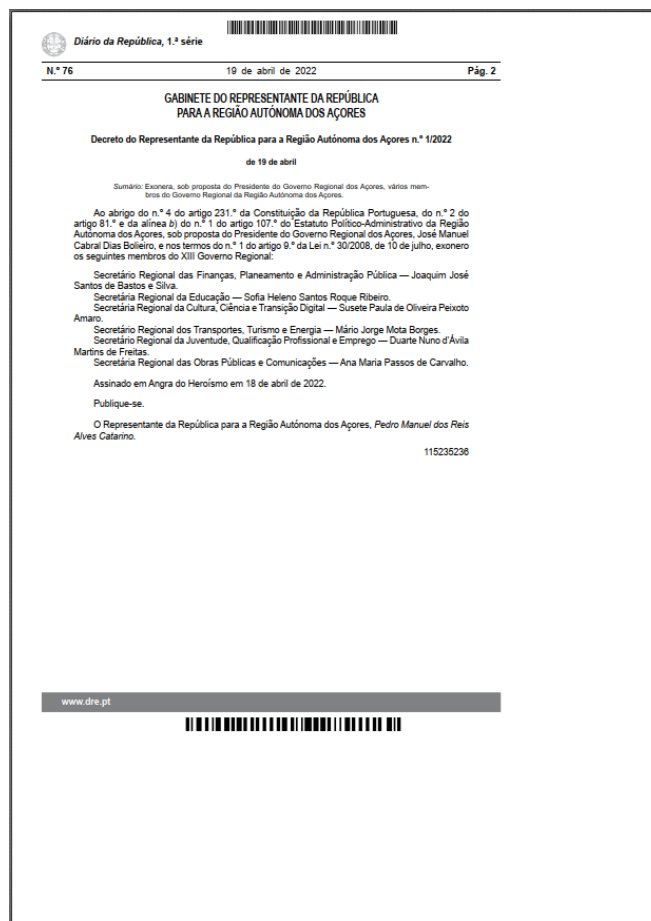


Figura 51 - Documento com 80% de escala

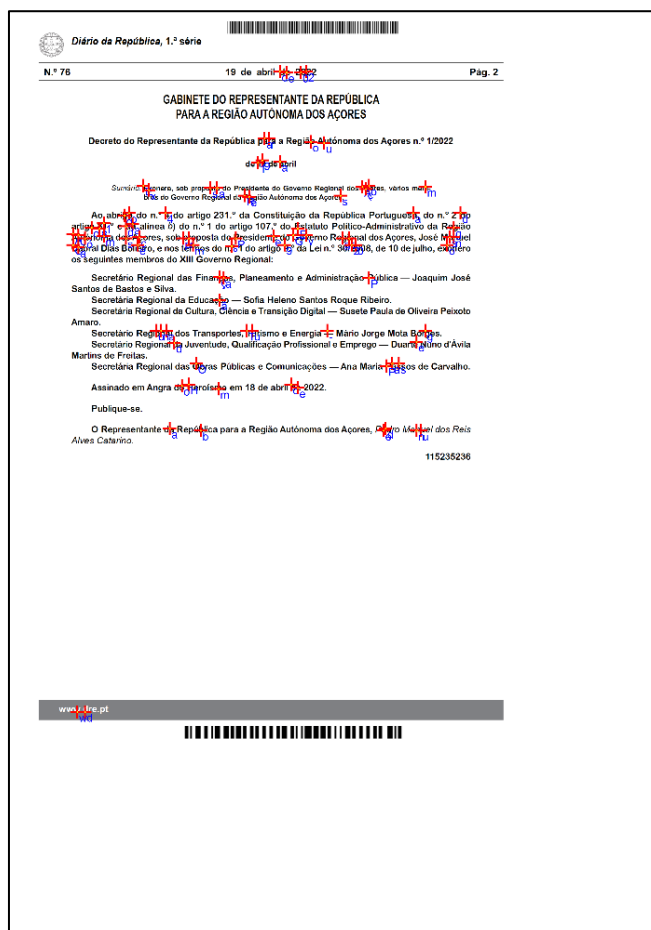


Figura 52 – Resultado integridade do documento com 80% de escala

7.2. Alterações no ficheiro original

A importância de testar o algoritmo para diversas situações é fundamental para que este seja robusto. A funcionalidade principal do algoritmo é a verificação de letras no documento, para isso é necessário visualizar o comportamento do algoritmo em situações de eliminação/adição de informação no ficheiro, alterações de palavras, e eliminação /adição de espaçamento entre linhas.

Para verificar se o algoritmo processa novos ficheiros, criou-se um com informações aleatórias presente na figura 53 e os metadados da mesma na figura 54. Com a realização do teste do processamento do ficheiro e verificação da integridade do mesmo, retificou-se que a solução é escalável, pelo output da figura 55.

Criaram-se 3 ficheiros idênticos ao da figura 53, com alterações, sendo eles:

1. Substituição de palavra: para a substituição resolveu-se substituir a palavra “importante” por “necessário” presente no retângulo da figura 56. Pode-se verificar que a letra “n” está por cima da letra “r”, o que leva a inferir que o documento naquela zona foi alterado;
2. Eliminação de palavras: eliminação do último parágrafo do documento da figura 53 presente no retângulo da figura 57. Verifica-se através da letra “D” inserida numa zona branca do documento, que houve remoção de pelo menos uma palavra.
3. Eliminação de espaçamento: removeu-se um espaçamento do documento da figura 53, representado pelo retângulo na figura 58. Através da análise do documento, observa-se que existiu uma translação no eixo do y (de cima para baixo) - demonstrada pela seta na figura 58, inferindo que ocorreu mudanças na estrutura do documento.

Em suma, se o documento que der entrada no sistema tiver os códigos de barras consegue-se inferir as zonas de alteração do documento, caso esta esteja incluída na marca de água. Dando um exemplo, imaginando que a palavra “Nacional” era alterada, o algoritmo não deteta, mas caso exista adições ou eliminações de palavra consegue detetar como é caso do ponto 2 e 3, referidos anteriormente.

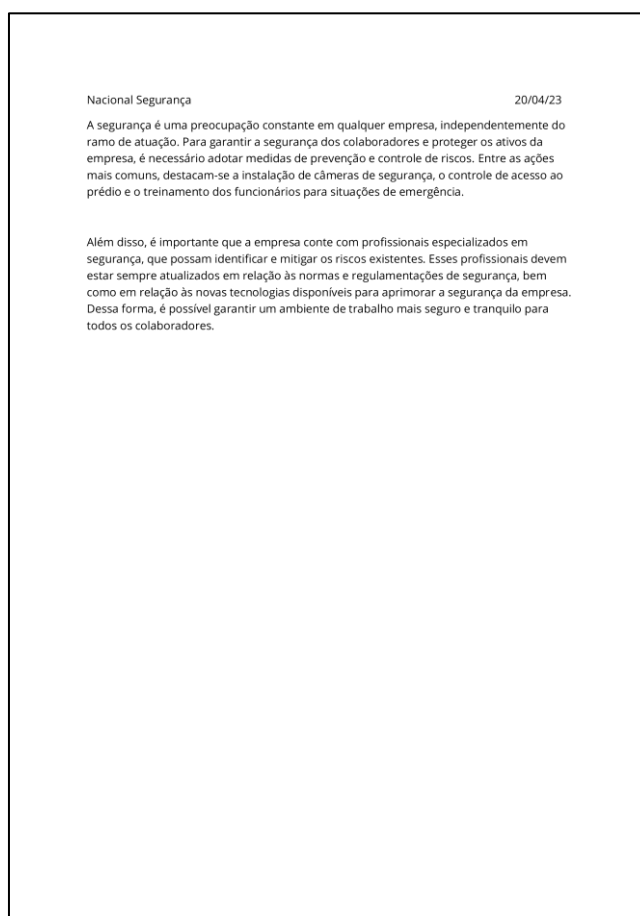


Figura 53 - Documento Teste

```

string ficheiroSeg = Path.Combine(partialPath, @"Ficheiros\Nacional
Segurança-test.pdf");
Metadata metadataRegistoSeg = new Metadata();
metadataRegistoSeg.NumeroRegisto = @"4/2022/1000";
metadataRegistoSeg.NumeroExemplar = 1;
metadataRegistoSeg.NumeroCopia = 0;
metadataRegistoSeg.ClassificacaoSeguranca = "S";
metadataRegistoSeg.EstadoExemplar = Estado.Ativo;
metadataRegistoSeg.FormatoExemplar = Formato.Eletronico;
metadataRegistoSeg.Utilizador = "Daniela Sequeira";
metadataRegistoSeg.DataOperacao = new DateTime(2022, 04, 20, 17, 00, 35);
metadataRegistoSeg.SiglaPrincipal = @"Nacional Segurança";
metadataRegistoSeg.PostoAtual = "Registo Central";
metadataRegistoSeg.Dominio = "NACIONAL";
conteudos.Add(metadataRegistoSeg, ficheiroSeg);

```

Figura 54 - Código Inserção metados

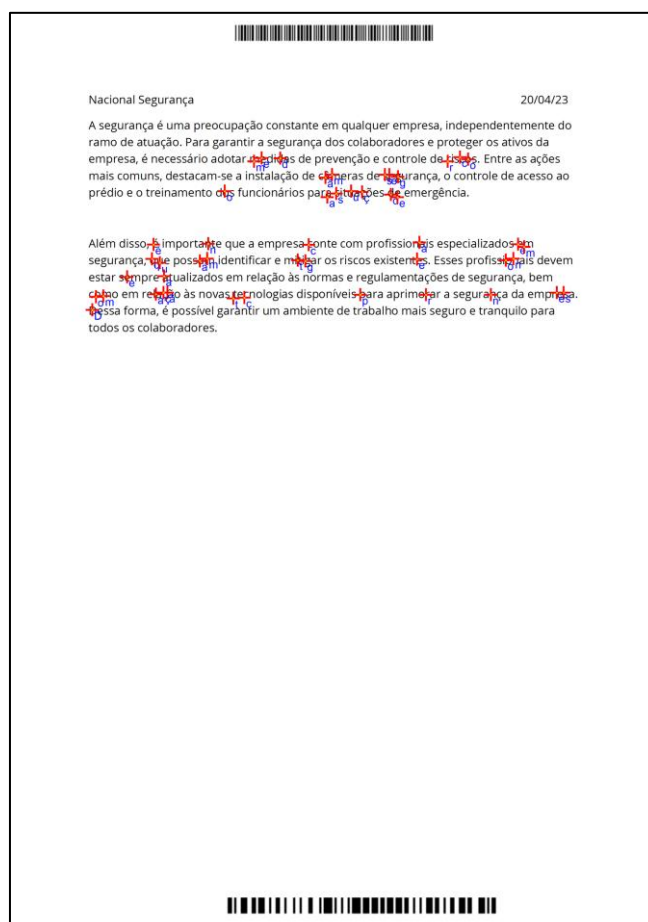


Figura 55 - Resultado da verificação da integridade do documento normal

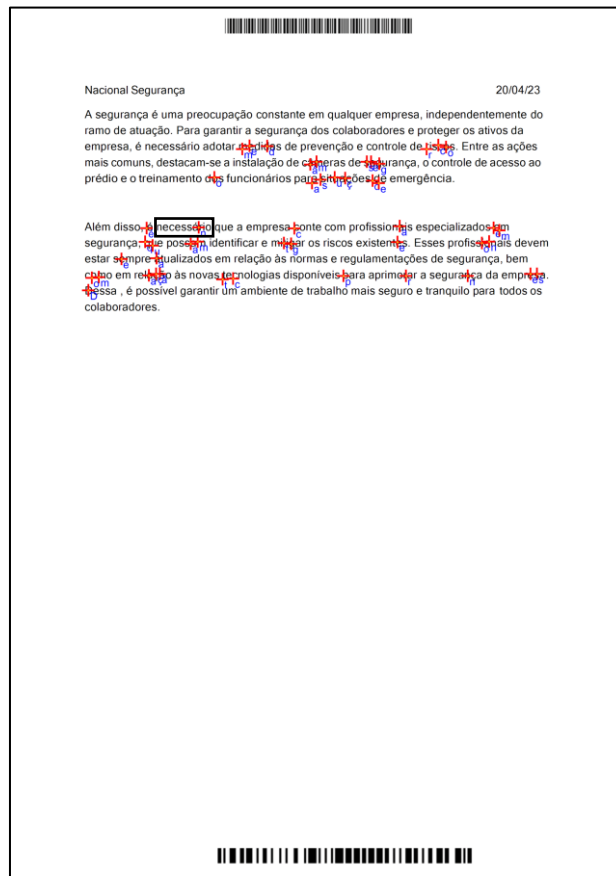


Figura 56 - Resultado da verificação da integridade do documento com substituição de palavra

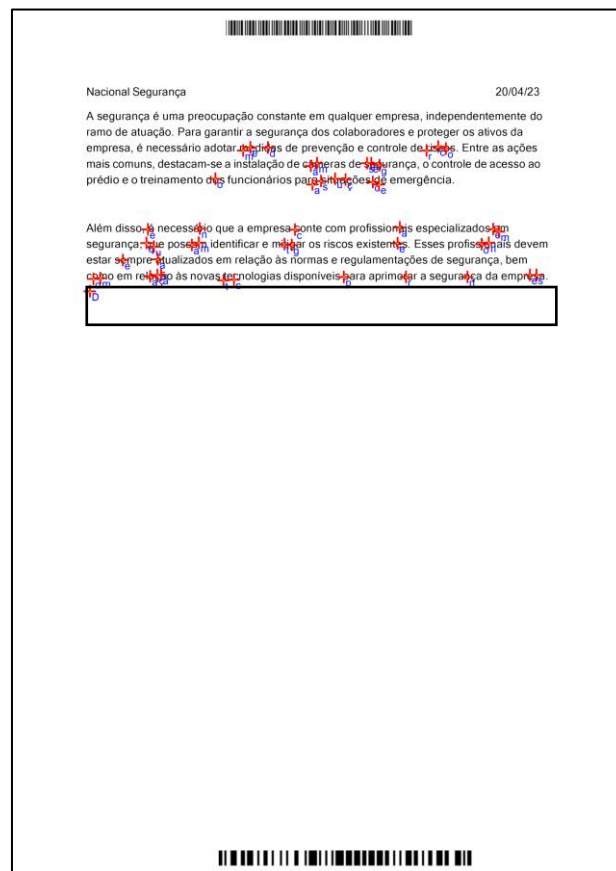


Figura 57 - Resultado da verificação da integridade do documento com eliminação de texto

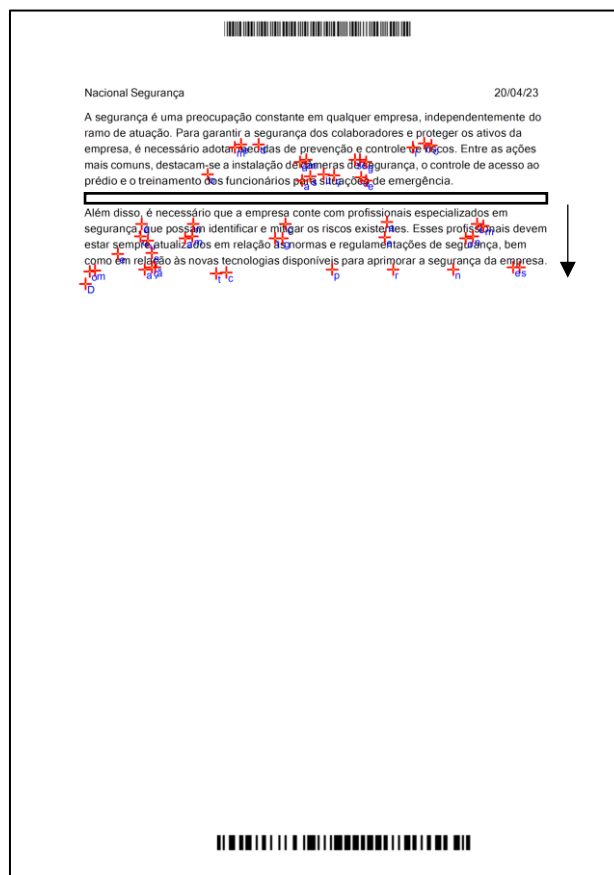


Figura 58 - Resultado da verificação da integridade do documento com eliminação de espaçamento

8. Conclusão

O intuito da dissertação é criar um método de autenticidade para documentos, quer digitais quer eletrónicos. Para isso pensou-se em marcas de água, mas a grande questão está como a usar?

Inicialmente pensou-se em usar marcas de água sobre texto, contudo a informação de documentos pode ser confidencial, o que leva a informação contida não ser possível modificada. A solução para resolver o problema foi usar código de barras para controlo de integridade e QR Code para verificação de integridade, usando os quadrados de posição como pontos de referência para traçar retas. Como os QR Code posteriormente não continham nenhuma informação e estavam a ocupar espaço visual no documento optou-se por removê-los, diminuindo assim a possibilidade de os hackers conseguirem perceber a origem dos pontos.

O sistema desenvolvido final permite assim concluir os objetivos iniciais de criação de um código de barras para validação rápida de um documento e criação de marca de água segura capaz de garantir que no documento contenha certas “*fingerprints*”, que permitam auxiliar os utilizadores em aspetos mais forenses para determinar as zonas do documento que foram alteradas. Apesar de o algoritmo desenvolvido funcionar apenas para a primeira página do documento, para fins demonstrativos, ele pode ser modificado no futuro, levando ao aumento de processamento e reconhecimento.

8.1. Futuro Trabalho

Para futuro trabalho, poderá se aumentar o espectro do algoritmo para mais folhas do documento e/ou mais pontos de letras. Verificar se possível a correção automática de códigos de barras deformados em ficheiros digitalizados.

Realizar um estudo de comparação de número ideal de palavras a guardar para garantir a proteção de um ficheiro confidencial.

Desenvolver um método de visualização das letras em 3D ou melhor para a comparação entre caracteres tornar-se mais fácil.

9. Referências

- [1] “Marca d’água – Wikipédia, a enciclopédia livre,” Oct. 22, 2022. https://pt.wikipedia.org/wiki/Marca_d%27água (accessed May 18, 2023).
- [2] M. B. Mohd, S. Mohd, R. Tanzila, and S. A. Rehman, “Replacement Attack: A New Zero Text Watermarking Attack,” *3D Res.*, vol. 8, 2017, doi: 10.1007/s13319-017-0118-y.
- [3] Z. Jalil, A. M. Mirza, and H. Jabeen, “Word length based zero-watermarking algorithm for tamper detection in text documents,” *ICCET 2010 - 2010 Int. Conf. Comput. Eng. Technol. Proc.*, vol. 6, no. May, 2010, doi: 10.1109/ICCET.2010.5486185.
- [4] T. Rethika, I. Prathap, R. Anitha, and S. V. Raghavan, “A novel approach to watermark text documents based on eigen values,” *2009 Int. Conf. Netw. Serv. Secur. N2S 2009*, no. c, pp. 1–5, 2009.
- [5] J. T. Brassil, S. Low, and N. F. Maxemchuk, “Copyright protection for the electronic distribution of text documents,” *Proc. IEEE*, vol. 87, no. 7, pp. 1181–1196, 1999, doi: 10.1109/5.771071.
- [6] “Code 128 - Wikipedia.” https://it.wikipedia.org/wiki/Code_128 (accessed Jan. 19, 2023).
- [7] “Relational Vs. Non-Relational Databases | MongoDB | MongoDB.” <https://www.mongodb.com/compare/relational-vs-non-relational-databases> (accessed Jan. 19, 2023).
- [8] “MySQL.” <https://www.mysql.com/> (accessed Jan. 19, 2023).
- [9] “Oracle SQL Developer Downloads.” <https://www.oracle.com/database/sqldeveloper/technologies/download/> (accessed Jan. 19, 2023).
- [10] “SQL Server Management Studio (SSMS) - SQL Server Management Studio (SSMS) | Microsoft Learn.” <https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16> (accessed Jan. 19, 2023).
- [11] “The Story of Gauss - National Council of Teachers of Mathematics.” <https://www.nctm.org/Publications/TCM-blog/Blog/The-Story-of-Gauss/> (accessed Feb. 08, 2023).
- [12] “Line–line intersection - Wikipedia.” https://en.wikipedia.org/wiki/Line–line_intersection (accessed Feb. 16, 2023).
- [13] “Apache PDFBox | A Java PDF Library.” <https://pdfbox.apache.org/> (accessed Feb. 17, 2023).

Anexo

A) Processamento de ficheiro PDF

Apesar de existirem alguns packages em c# retirassem as posições dos caracteres bem como os mesmos, não era suficiente, já que se precisava de um intervalo de valores onde começa e acaba a letra para determinar se o ponto de interseção pertence ou não à letra para ser usado na verificação de integridade, optou-se por utilizar o package Apache PDFBox [13] desenvolvido em java que conseguia, dar as posições do começo e fim de cada letra bem como a letra respetiva.

Para diminuir o tempo de processamento e para efeitos de teste apenas lê-se a primeira página do documento (figura 59), sendo possível depois alterar para todas as páginas, contudo o tempo de processamento aumenta também. Para os valores lidos serem acedidos em C# cria-se um ficheiro temporário que vai guardar as seguintes características “character|start_x,start_y,stop_x,stop_y” como se representa na figura 60.

Contudo é necessário compilar o código em java e criar um ficheiro Jar com os respetivos packages dependentes que permita a execução em C#, para isso abriu-se uma consola e executou-se um comando presente na figura 61.

Para demonstrar que o algoritmo obtém as letras precisamente, utilizou-se um leitor PDF denominado “PDF-Xchange Editor” que permite ver as posições do rato no documento como demonstra na figura 62, a seta aponta para onde está o rato, dando os valores (71, 38), na base de dados presente na figura 63, tem se que os valores de “start_x” a 71 e o “stop_y” a 37.

```
File file = new File(f[0]+"_pos.txt");
if(file.exists())
{
    file.delete();
}
else {
    file.createNewFile();
}

String fileName = args[0];
try {
    document = PDDocument.load( new File(fileName) );
    PDFTextStripper stripper = new PositionCharacter();
    stripper.setSortByPosition( true );
    stripper.setStartPage( 0 );
    stripper.setEndPage( 1 );
    Writer dummy = new OutputStreamWriter(new ByteArrayOutputStream());
    stripper.writeText(document, dummy);
}
```

Figura 59 Código de extração dos caracteres num ficheiro PDF

```
String[] f = file_name.split(".pdf");
FileWriter file = new FileWriter(f[0]+"_pos.txt", true);

String ch;

for (TextPosition text : textPositions)
{
    ch = text.getUnicode();

    //remove ?
    if (ch.equals("-"))
        ch = "-";
    else if (ch.equals("'"))
        ch = "";

    if(!ch.isBlank() && !ch.isEmpty()) // remove spaces
    {
        file.write(ch + "|" + Math.round(text.getX()) + "," +
Math.round(Math.abs(text.getHeight() - text.getY()))
+ "|" + Math.round(text.getEndX() - text.getWidthOfSpace()) + "|" +
Math.round(text.getY()) + "\n");
    }
}
```

Figura 60 - Obtenção dos valores

```

System.Diagnostics.Process process_file = new System.Diagnostics.Process();
process_file.StartInfo.UseShellExecute = false;
process_file.StartInfo.RedirectStandardOutput = true;
process_file.StartInfo.FileName = "java";
process_file.StartInfo.Arguments = "-jar " + '"' + jar_file + '"' + " " + '"' +
file_name + '"';
process_file.Start();
process_file.WaitForExit();

```

Figura 61 - Execução do ficheiro jar

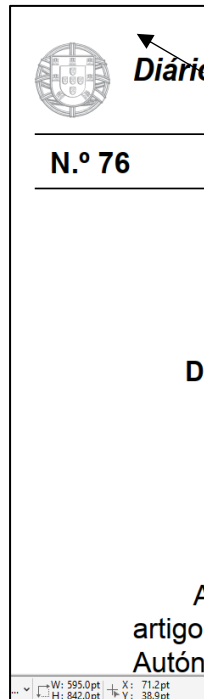


Figura 62 - Exemplo posição de caracter no PDF

Results		Messages				
	id_doc	value_char	start_x	start_y	stop_x	stop_y
1	308082904	D	71	29	76	37

Figura 63 - Valor representativo da base de dados

B) ZXing.NET

ZXing.NET é uma biblioteca “open-source” desenvolvida em Java, que permite decodificar e gerar códigos de barras (lineares e 2 dimensões). Esta permite ler e codificar os códigos de barras utilizados na dissertação que são eles o 128 e 39, também permite obter as posições dos códigos de barras na imagem.

O código para gerar o código de barras apresenta-se na figura 64 e a decodificação na figura 65, cujo tem só a leitura do código de barras 128, já que este tem informações necessárias que permite retirar dados da base de dados.

```
public void Generate_barcode(int id_barcode)
{
    string data_barcode = id_doc.ToString() + ";" + id_barcode.ToString();
    var writer = new BarcodeWriter
    {
        Format = BarcodeFormat.CODE_128,
        Options = new EncodingOptions
        {
            Height = resizedBarcode,
            Width = 250,
            Margin = 0
        }
    };
    var barcodeBitmap = writer.Write(data_barcode);
    barcodeBitmap.Save(filename + common.extension_barcode);
}

public void Generate_barcode_39(int id_barcode)
{
    string data_barcode = id_doc.ToString() + ";" + id_barcode.ToString();
    var writer = new BarcodeWriter
    {
        Format = BarcodeFormat.CODE_39,
        Options = new EncodingOptions
        {
            Height = resizedBarcode,
            Width = 200,
            Margin = 0
        }
    };
    var barcodeBitmap = writer.Write(data_barcode);
    barcodeBitmap.Save(filename + " code39.png");
}
```

Figura 64 - Gerar códigos de barras 128 e 39

```
Bitmap bmp = new Bitmap(img_file);
var reader = new BarcodeReader
{
    Options = new DecodingOptions
    {
        PossibleFormats = new List<BarcodeFormat> { BarcodeFormat.CODE_128 },
        TryHarder = true
    }
};

Result result = reader.Decode(bmp);

if (result != null)
{
    return result.Text;
}
else
    trackerServices.WriteFile("erro na leitura do código de barras");
```

Figura 65 - Leitura de código de barras 128

C) Decodificação de posições dos códigos de barras

Para confirmar se o package ZXing.Net retirava bem as posições do código de barras (figura 66), criou-se uma imagem auxiliar onde se escreverá as posições retornadas (figura 67). É de salientar que os valores foram ajustados para coincidir com as pontas do código de barras, sendo que para os documentos fornecidos, determinou bem as posições. A figura 68 representa um exemplo de um documento com os códigos de barras 128 e 39 onde as bolas amarelas representam as posições do código de barras 128 e 39.

```
Bitmap bmp = new Bitmap(img_file);
var reader128 = new BarcodeReader
{
    Options = new DecodingOptions
    {
        PossibleFormats = new List<BarcodeFormat> { BarcodeFormat.CODE_128 },
        TryHarder = true
    }
};

var reader39 = new Bytescout.BarCodeReader.Reader();
reader39.BarcodeTypesToFind.Code39 = true;
reader39.MaxNumberOfBarcodesPerPage = 1;
var result2 = reader39.ReadFrom(bmp);

var barcodeResult128 = reader128.Decode(bmp);
var result = barcodeResult128?.ResultPoints;

List<Point> list128 = new List<Point>();
List<Point> list39 = new List<Point>();
Point p1_barcode128 = new Point();
Point p2_barcode128 = new Point();
Point p1_barcode39 = new Point();

if (result != null && result2 != null)
{
    foreach (var point in result)
    {
        int x = (int)point.X * width / bmp.Width;
        int y = (int)point.Y * height / bmp.Height;
        list128.Add(new Point(x, y));
    }
    foreach (var point2 in result2)
    {
        int x = point2.Rect.X * width / bmp.Width;
        int y = point2.Rect.Y * height / bmp.Height;
        list39.Add(new Point(x, y));
    }

    for (int i = 0; i < list128.Count; i++)
    {
        p1_barcode128 = list128[0];
        p2_barcode128 = list128[1];
    }
    for (int i = 0; i < list39.Count; i++)
    {
        p1_barcode39 = list39[0];
    }
}
else
{
    trackerServices.WriteFile("erro ao obter as posições do código de barras");
    return "";
}

x_barcode_pos = p1_barcode128.X - 10;
y_barcode_pos = p1_barcode128.Y - 2;
x2_barcode_pos = p2_barcode128.X + 16;
y2_barcode_pos = p2_barcode128.Y - 2 + 15;

x_39 = p1_barcode39.X + 1;
y_39 = p1_barcode39.Y;
x2_39 = p1_barcode39.X + 195;
y2_39 = y_39 + 15;

if (file_name.Contains("scan"))
{
    x_barcode_pos = p1_barcode128.X - 10;
    y_barcode_pos = p1_barcode128.Y;
    x2_barcode_pos = p2_barcode128.X + 16;
    y2_barcode_pos = p2_barcode128.Y + 15;

    x_39 = p1_barcode39.X + 1;
    y_39 = p1_barcode39.Y + 1;
    x2_39 = p1_barcode39.X + 195;
    y2_39 = y_39 + 15;
}
bmp.Dispose();
return $"{x_barcode_pos}:{y_barcode_pos}:{x2_barcode_pos}:{y2_barcode_pos}:{x_39}:{y_39}:{x2_39}:{y2_39}";
```

Figura 66 - Obtenção de posições do código de barras

```

int p_x = p1_dig.X * bmp.Width / commom.width;
int p_y = p1_dig.Y * bmp.Height / commom.height;
int p2_x = p2_dig.X * bmp.Width / commom.width;
int p2_y = p2_dig.Y * bmp.Height / commom.height;
Point p1_l_u_r = new Point(p_x, p_y);
Point p1_r_u_r = new Point(p2_x, p_y);
Point p1_r_b_r = new Point(p2_x, p2_y);
Point p1_l_b_r = new Point(p_x, p2_y);


g.DrawArc(yellow, p1_l_u_r.X, p1_l_u_r.Y, w_arc, h_arc, startAngle, sweepAngle);
g.DrawArc(yellow, p1_r_u_r.X, p1_r_u_r.Y, w_arc, h_arc, startAngle, sweepAngle);
g.DrawArc(yellow, p1_l_b_r.X, p1_l_b_r.Y, w_arc, h_arc, startAngle, sweepAngle);
g.DrawArc(yellow, p1_r_b_r.X, p1_r_b_r.Y, w_arc, h_arc, startAngle, sweepAngle);


int p_x_39_dig = p1_39_dig.X * bmp.Width / commom.width;
int p_y_39_dig = p1_39_dig.Y * bmp.Height / commom.height;
int p2_x_39_dig = p2_39_dig.X * bmp.Width / commom.width;
int p2_y_39_dig = p2_39_dig.Y * bmp.Height / commom.height;
Point p1_l_u_39_dig = new Point(p_x_39_dig, p_y_39_dig);
Point p1_r_u_39_dig = new Point(p2_x_39_dig, p_y_39_dig);
Point p1_r_b_39_dig = new Point(p2_x_39_dig, p2_y_39_dig);
Point p1_l_b_39_dig = new Point(p_x_39_dig, p2_y_39_dig);

g.DrawArc(pink, p1_l_u_39_dig.X, p1_l_u_39_dig.Y, w_arc, h_arc, startAngle, sweepAngle);
g.DrawArc(pink, p1_r_u_39_dig.X, p1_r_u_39_dig.Y, w_arc, h_arc, startAngle, sweepAngle);
g.DrawArc(pink, p1_l_b_39_dig.X, p1_l_b_39_dig.Y, w_arc, h_arc, startAngle, sweepAngle);
g.DrawArc(pink, p1_r_b_39_dig.X, p1_r_b_39_dig.Y, w_arc, h_arc, startAngle, sweepAngle);

```

Figura 68 - Guardar imagem auxiliar


Diário da República, 1.ª série



N.º 76
19 de abril de 2022
Pág. 2

**GABINETE DO REPRESENTANTE DA REPÚBLICA
PARA A REGIÃO AUTÓNOMA DOS AÇORES**

Decreto do Representante da República para a Região Autónoma dos Açores n.º 1/2022

de 19 de abril

Sumário: Exonera, sob proposta do Presidente do Governo Regional dos Açores, vários membros do Governo Regional da Região Autónoma dos Açores.

Ao abrigo do n.º 4 do artigo 231.º da Constituição da República Portuguesa, do n.º 2 do artigo 81.º e da alínea b) do n.º 1 do artigo 107.º do Estatuto Político-Administrativo da Região Autónoma dos Açores, sob proposta do Presidente do Governo Regional dos Açores, José Manuel Cabral Dias Bolieiro, e nos termos do n.º 1 do artigo 9.º da Lei n.º 30/2008, de 10 de julho, exonero os seguintes membros do XIII Governo Regional:


Secretário Regional das Finanças, Planeamento e Administração Pública — Joaquim José Santos de Bastos e Silva.
 Secretária Regional da Educação — Sofia Heleno Santos Roque Ribeiro.
 Secretária Regional da Cultura, Ciência e Transição Digital — Susete Paula de Oliveira Peixoto Amaro.
 Secretário Regional dos Transportes, Turismo e Energia — Mário Jorge Mota Borges.
 Secretário Regional da Juventude, Qualificação Profissional e Emprego — Duarte Nuno d'Ávila Martins de Freitas.
 Secretária Regional das Obras Públicas e Comunicações — Ana Maria Passos de Carvalho.

Assinado em Angra do Heroísmo em 18 de abril de 2022.

Publique-se.

O Representante da República para a Região Autónoma dos Açores, *Pedro Manuel dos Reis Alves Catarino*.

115235236



www.dre.pt

Figura 69 - Representação dos pontos do código de barras

D) Posições aleatórias para marca de água

Este anexo tem como intuito demonstrar as posições que irão dar origem aos segmentos de reta. A figura 68 apresenta um pedaço de código que irá escrever as posições dos pontos numa imagem auxiliar (figura 70), sendo que vai buscar todos os pontos que se predefine, neste caso 9 (numberPoints), gerando um valor aleatório de espaçamento (x, y) para criar os pontos aleatórios para dar a origem aos pontos das retas (l - esquerda, r - direita, b - baixo). A figura 71 demonstra um exemplo dos pontos, sendo que para ponto tem as origens dos pontos das retas aleatórios, podem existir pontos com as mesmas coordenadas, já que estes são aleatórios. A figura 72 demonstra, mais detalhadamente o espaçamento entre pontos.

```
Graphics g = Graphics.FromImage(bmp);
Font drawFont = new Font("Arial", 8);
SolidBrush drawBrush = new SolidBrush(Color.Blue);

Dictionary<string, Point> circle_points = new Dictionary<string, Point>();
for(int i = 0; i < numberPoint; i++)
{
    string[] pos_circles = positions.Split('|');
    string[] circles = pos_circles[i].Split(',');
    int x_circle = int.Parse(circles[0]) * bmp.Width / w;
    int y_circle = int.Parse(circles[1]) * bmp.Height / h;

    Random random = new Random();
    int randomX = random.Next(min_random, max_random);
    int randomY = random.Next(min_random, max_random);

    Point origin_point = new Point(x_circle, y_circle);
    Point circles_l = new Point(x_circle + randomX, y_circle - randomY);
    Point circles_r = new Point(x_circle - randomX, y_circle - randomY);
    Point circles_b = new Point(x_circle - randomX, y_circle);

    g.DrawString("p", drawFont, drawBrush, origin_point);
    g.DrawString("l", drawFont, drawBrush, circles_l);
    g.DrawString("r", drawFont, drawBrush, circles_r);
    g.DrawString("b", drawFont, drawBrush, circles_b);

    circle_points.Add("point" + (i + 1) + "_l", circles_l);
    circle_points.Add("point" + (i + 1) + "_r", circles_r);
    circle_points.Add("point" + (i + 1) + "_b", circles_b);
}

string path = Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location) +
@"\Ficheiros\spefications.png";
bmp.Save(path);

return circle_points;
```

Figura 70 - Código para demonstra posições na figura

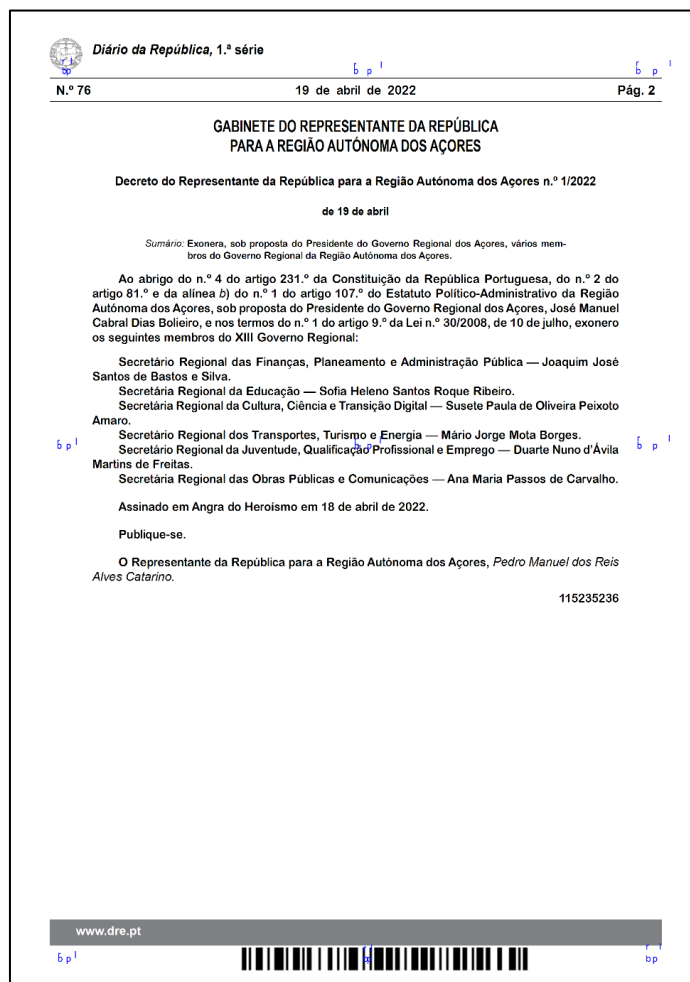


Figura 71 – Demonstração da criação dos pontos

Ponto 1	com	desfazamento	no x de 15	e no y de 34
Ponto 2	com	desfazamento	no x de 48	e no y de 21
Ponto 3	com	desfazamento	no x de 60	e no y de 27
Ponto 4	com	desfazamento	no x de 33	e no y de 15
Ponto 5	com	desfazamento	no x de 45	e no y de 20
Ponto 6	com	desfazamento	no x de 56	e no y de 26
Ponto 7	com	desfazamento	no x de 30	e no y de 14
Ponto 8	com	desfazamento	no x de 13	e no y de 37
Ponto 9	com	desfazamento	no x de 24	e no y de 43

Figura 72 – Espaçamento dos pontos

E) Mudança de escala do documento

A escala do documento é mudada através do uso do package ITextSharp que permite alterações em ficheiros PDF.

O código apresentado na figura 73, acede ao ficheiro que se quer alterar, permitindo a mudança de escala em ficheiros seguros através da definição unethicalreading. A seguir cria um ficheiro novo com a extensão da escola desejada. O novo ficheiro com a escala criada vai ser colocada na margem esquerda em cima do ficheiro com as dimensões do ficheiro original, ou seja, o ficheiro vai ter as dimensões do original, mas não vai preencher a página toda. Na secção 8.1 apresenta-se um documento exemplo com 80% de escala.

O cálculo da escala do ficheiro é calculado com base na posição do ponto y do código de barras 128 sobre a dimensão do ficheiro original que é sempre 842, somando 0.03 para ajustar a escala. Devido aos comprimentos e pontos sofrerem proporção da escala é necessário adaptar os pontos (figura 74). Para finalizar adaptam-se as posições de interseção com base na percentagem de escala como demonstra a figura 75.

```
public void Scale(decimal scalef)
{
    int s = Convert.ToInt16(scalef * 100);
    PdfReader reader = new PdfReader(name);
    PdfReader.unethicalreading = true; // aceder a documentos confidenciais
    Document doc = new Document();
    PdfWriter writer = PdfWriter.GetInstance(doc, new FileStream(name_without_ex + "_scale_" + s + ".pdf",
    FileMode.Create));
    doc.Open();
    PdfImportedPage page = writer.GetImportedPage(reader, 1); //page #1
    PdfDictionary pageDict = reader.GetPageN(1);
    pageDict.Put(PdfName.PRINTSCALING, new PdfNumber((float)scalef));

    float yPos = reader.GetPageSize(1).Height - (reader.GetPageSize(1).Height * (float)scalef);
    writer.DirectContent.AddTemplate(page, (float)scalef, 0, 0, (float)scalef, 0, yPos);

    doc.NewPage();
    doc.Close();

    reader.Close();
}
```

Figura 73 - Alterar escala do documento

```
scale_doc = Math.Round(Convert.ToDecimal(((double)p1_dig.Y / 842)), 2);
scale_doc += 0.03m;

if (scale_doc == 0.99m)
    scale_doc = 1.00m;
if (p1_39_dig.Y == 10)
    p1_39_dig.Y = 11;

if (scale_doc != 1.00m)
{
    int x_scale = Convert.ToInt16(x_diff_or * scale_doc);
    int x_39_scale = Convert.ToInt16(x_39_diff_or * scale_doc);
    int y_scale = Convert.ToInt16(y_diff_or * scale_doc);

    p1_dig.X += 2;
    p2_dig.X = Convert.ToInt16(p1_dig.X + x_scale);
    p2_dig.Y = Convert.ToInt16(p2_dig.Y + y_scale - y_diff_or);
    p2_39_dig.X = Convert.ToInt16(p1_39_dig.X + x_39_scale);
    p2_39_dig.Y = Convert.ToInt16(p2_39_dig.Y + y_scale - y_diff_or + 1);
}
```

Figura 74 – Cálculo da escala do documento

```
if (scale_doc >= 1.00f)
    intersection = new Point(res_x, res_y);
else
{
    int n_x = Convert.ToInt16(res_x * (double)w/ bmp.Width);
    int n_y = Convert.ToInt16(res_y * (double)h/ bmp.Height);
    int s_x = Convert.ToInt16(n_x * scale_doc);
    int s_y = Convert.ToInt16(n_y * scale_doc);
    int new_x = Convert.ToInt16(s_x * (double)bmp.Width/w);
    int new_y = Convert.ToInt16(s_y * (double)bmp.Height/h);
}
```

Figura 75 - Adaptar cálculo das posições