|  |  |  |  |
| --- | --- | --- | --- |
|  | **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 e do Doutor Luís Amorim, Professor Auxiliar Convidado 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 | Quero agradecer desde já aos meus pais e irmã por me darem apoio e força para nunca desistir.  Não posso deixar de agradecer aos meus Orientadores André Zúquete e Luís Amorim pelo suporte e ideias para desenvolver a dissertação. |

|  |  |
| --- | --- |
|  |  |
|  |  |
| palavras-chave | Marca de Água, Códigos de Barras, Documentos impressos, Documentos Digitais, Documentos Eletrónicos, Segurança, Integridade, Autenticidade |
| resumo |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
| keywords | Watermark, Barcodes, Printed documents, Digital Documents, Electronic Documents, Security, Integrity |
| abstract |  |

# Índice

[Índice ii](#_Toc136555032)

[Lista de Tabelas iii](#_Toc136555033)

[Lista de Figuras iv](#_Toc136555034)

[1. Introdução 1](#_Toc136555035)

[2. Contexto 2](#_Toc136555036)

[2.1. Código de Barras 2](#_Toc136555037)

[2.1.1. Análise de Código de Barras 2](#_Toc136555038)

[2.1.1.1. QR Code 5](#_Toc136555039)

[2.1.1.2. Código de Barras 128 6](#_Toc136555040)

[2.2. Processo de criação da metainformação 7](#_Toc136555041)

[2.2.1. Criação de segmentos de reta e pontos 7](#_Toc136555042)

[2.2.2. Obtenção das letras e respetivas posições num documento PDF 7](#_Toc136555043)

[3. Trabalhos Relacionados 8](#_Toc136555044)

[3.1. Marca de Água 8](#_Toc136555045)

[3.2. Marcas de Água Visíveis 8](#_Toc136555046)

[3.2.1. Substituição de Palavras 8](#_Toc136555047)

[3.2.1.1. Exemplo 9](#_Toc136555048)

[3.2.1.2. Vantagens 9](#_Toc136555049)

[3.2.1.3. Desvantagens 9](#_Toc136555050)

[3.2.2. Marca de Água sem Mudança de Texto 10](#_Toc136555051)

[3.2.2.1. Vantagens 10](#_Toc136555052)

[3.2.2.2. Desvantagens 10](#_Toc136555053)

[3.2.3. Documentos baseados em Eigenvalues 10](#_Toc136555054)

[3.2.3.1. Testes 10](#_Toc136555055)

[3.2.3.2. Vantagens 12](#_Toc136555056)

[3.2.3.3. Desvantagens 12](#_Toc136555057)

[3.2.4. Marca de Água com Espaçamento 12](#_Toc136555058)

[3.2.4.1. Line-Shift Coding 12](#_Toc136555059)

[3.2.4.2. Word-Shift Coding 12](#_Toc136555060)

[3.2.4.3. Character Coding 13](#_Toc136555061)

[3.2.4.4. Vantagens dos Métodos 13](#_Toc136555062)

[3.2.4.5. Desvantagens dos Métodos 13](#_Toc136555063)

[4. Algoritmo de Marcação e Análise de Documentos Impressos 14](#_Toc136555064)

[4.1. Processamento de Documentos 14](#_Toc136555065)

[4.2. Verificação de Documentos Impressos 15](#_Toc136555066)

[5. Implementação 16](#_Toc136555067)

[5.1. Linguagens de Programação 16](#_Toc136555068)

[5.2. Arquitetura 16](#_Toc136555069)

[5.3. Documentos 17](#_Toc136555070)

[5.3.1. Conversão para Bitmap 17](#_Toc136555071)

[5.3.2. Obter Dimensões do PDF e da Imagem 17](#_Toc136555072)

[5.3.2. Criação dos Pontos para o Sistema de Verificação 18](#_Toc136555073)

[5.3.3. Criação do Documento que Verifica a Autenticidade e Adaptação dos Pontos 19](#_Toc136555074)

[5.4. Base de Dados 21](#_Toc136555075)

[5.5. Obtenção das letras e respetivas posições num documento PDF 22](#_Toc136555076)

[5.6. Código de Barras 24](#_Toc136555077)

[5.6.1. Criação e Leitura 24](#_Toc136555078)

[5.6.2. Posições 25](#_Toc136555079)

[5.7. Interface Gráfica (GUI) 29](#_Toc136555080)

[6. Testes 34](#_Toc136555081)

[6.1. Documentos Digitais com Rotações 34](#_Toc136555082)

[6.2. Mudança de Escala no Documento 38](#_Toc136555083)

[6.3. Modificações no Documento 44](#_Toc136555084)

[7. Conclusões 50](#_Toc136555085)

[7.1. Trabalho Futuro 50](#_Toc136555086)

[8. Referências 51](#_Toc136555087)

[Anexo 52](#_Toc136555088)

[A) Mudança de Escala do Documento 52](#_Toc136555089)

# Lista de Tabelas

[Tabela 1 - Código de Barras Lineares 3](#_Toc136554932)

[Tabela 2 - Lista de Códigos de Barras Ideias 4](#_Toc136554933)

[Tabela 3 - Classificação de Erros do QR Code 5](#_Toc136554934)

[Tabela 4 - Resultados Eigenvalues 12](#_Toc136554935)

# Lista de Figuras

[Figura 1 - Exemplo QR Code 5](#_Toc136554967)

[Figura 2 - Estrutura QR Code 5](#_Toc136554968)

[Figura 3 - Código de Barra 128 6](#_Toc136554969)

[Figura 4 - Tabela ASCII 6](#_Toc136554970)

[Figura 5 - Exemplo da Substituição de Texto 9](#_Toc136554971)

[Figura 6 - Antes da Marca de Água 11](#_Toc136554972)

[Figura 7 - Depois da Marca de Água 11](#_Toc136554973)

[Figura 8 - Comparação Marca de água 11](#_Toc136554974)

[Figura 9 - Exemplo Line-Shift 12](#_Toc136554975)

[Figura 10 - Exemplo Word-Shift 13](#_Toc136554976)

[Figura 11 - Exemplo Character Coding 13](#_Toc136554977)

[Figura 12 - Diagrama de Fluxo do Processamento de um Documento 15](#_Toc136554978)

[Figura 13 - Representação dos Valores dos Caracteres 15](#_Toc136554979)

[Figura 14 - Diagrama de Fluxo da Averiguação de um Documento 15](#_Toc136554980)

[Figura 15 - Arquitetura Solução 16](#_Toc136554981)

[Figura 16 - Código para criar uma Imagem da primeira Página do PDF 17](#_Toc136554982)

[Figura 17 - Código para obter as dimensões do documento PDF e da Imagem Convertida 17](#_Toc136554983)

[Figura 18 - Código para Demonstração dos Pontos da Marca de Água 18](#_Toc136554984)

[Figura 19 - Representação dos Pontos 19](#_Toc136554985)

[Figura 20 - Espaçamento dos Pontos 19](#_Toc136554986)

[Figura 21 - Criação da Imagem que Visualiza as Letras do Sistema de Verificação da Base de Dados no Documento e Adaptação dos Pontos das Letras 20](#_Toc136554987)

[Figura 22 - Diagrama da Base de Dados 21](#_Toc136554988)

[Figura 23 - Criação do utilizador para a base de dados 21](#_Toc136554989)

[Figura 24 - Código para Abrir o Documento e Extrair Informação 22](#_Toc136554990)

[Figura 25 - Escrita das Letras e Respetivas Posições num Ficheiro Temporário 23](#_Toc136554991)

[Figura 26 - Código para a Execução do Ficheiro .Jar 23](#_Toc136554992)

[Figura 27 - Exemplo de uma Letra e Respetiva posição no PDF com a Utilização da Ferramenta PDF-XChange Editor 23](#_Toc136554993)

[Figura 28 - Valor Lido pelo Algoritmo Representado na Base de Dados 23](#_Toc136554994)

[Figura 29 - Criação dos Códigos de Barras 128 e 39 24](#_Toc136554995)

[Figura 30 - Descodificação do Código de Barras 128 25](#_Toc136554996)

[Figura 31 - Obtenção de Posições do Código de Barras 26](#_Toc136554997)

[Figura 32 - Guardar Imagem Auxiliar 27](#_Toc136554998)

[Figura 33 - Representação dos Pontos do Código de Barras 28](#_Toc136554999)

[Figura 34 - Menu 30](#_Toc136555000)

[Figura 35 - Erro Extensão 30](#_Toc136555001)

[Figura 36 - Erro Ficheiro já Processado 30](#_Toc136555002)

[Figura 37 - Menu Processar 30](#_Toc136555003)

[Figura 38 - Mensagem de Aviso, caso o utilizador clique em aceitar ou rejeitar 31](#_Toc136555004)

[Figura 39 - Informação de que o processamento acabou e que é necessário aprovação ou rejeição do documento 31](#_Toc136555005)

[Figura 40 - Conclusão do Processamento 31](#_Toc136555006)

[Figura 41 - Informação Documento Aceite 31](#_Toc136555007)

[Figura 42 - Informação Documento Já Aceite 32](#_Toc136555008)

[Figura 43 - Informação do Erro na seleção de ficheiro 32](#_Toc136555009)

[Figura 44 - Erro Leitura Código de Barras 32](#_Toc136555010)

[Figura 45 - Verificação da Autenticidade de um Documento 32](#_Toc136555011)

[Figura 46 - Informação do Procedimento à Verificação da Integridade 32](#_Toc136555012)

[Figura 47 - Resultado da Verificação da Integridade 33](#_Toc136555013)

[Figura 48 - Documento Digital Torto 35](#_Toc136555014)

[Figura 49 - Documento Digital Torto Endireitado 36](#_Toc136555015)

[Figura 50 - Resultado da Averiguação da Integridade do Documento Torto 37](#_Toc136555016)

[Figura 51 - Resultado da Averiguação da Integridade do Documento (Formato Eletrónico) 38](#_Toc136555017)

[Figura 52 - Documento Modificado com 50% de Escala 39](#_Toc136555018)

[Figura 53 - Documento Original 40](#_Toc136555019)

[Figura 54 - Resultado da Averiguação Profunda da Integridade do Documento Original 41](#_Toc136555020)

[Figura 55 - Documento Modificado com 80% de Escala 42](#_Toc136555021)

[Figura 56 - Resultado da Verificação Profundo do Documento Modificado com 80% de Escala 43](#_Toc136555022)

[Figura 57 - Documento Exemplificativo 45](#_Toc136555023)

[Figura 58 - Código para a Inserção de Metadados 45](#_Toc136555024)

[Figura 59 - Resultado da Verificação da Integridade do Documento Exemplar 46](#_Toc136555025)

[Figura 60 - Resultado da Verificação da Integridade do Documento Exemplar com Substituição da Palavra 47](#_Toc136555026)

[Figura 61 - Resultado da Verificação da Integridade do Documento Exemplar com Eliminação de Texto 48](#_Toc136555027)

[Figura 62 - Resultado da Verificação da Integridade do Documento Exemplar com Eliminação de Espaçamento 49](#_Toc136555028)

[Figura 63 - Mudança Escala do Documento 52](#_Toc136555029)

[Figura 64 - Cálculo da Escala do Documento 52](#_Toc136555030)

[Figura 65 - Adaptação do Cálculo das Posições 53](#_Toc136555031)

# Introdução

Cada país tem os seus próprios segredos e informações valiosas tal como militar, político ou saúde, que não querem que sejam divulgadas a terceiros, ou a inimigos. Como diz Francis Bacon “*Conhecimento é poder*” [[1](#one)], isto demonstra que existem pessoas com saciedade para conhecer o desconhecido do inimigo, daí que haja roubo, destruição, acesso a matérias classificadas. Porventura, estas carecem de proteção especial para evitar fugas. Esta proteção pressupõe que a organização responsável por garantir a segurança das matérias classificadas do país, tenha conhecimento da possessão do material classificado, do lugar geográfico do material, em que lugar deveria se encontrar e quem tem ou teve acesso ao mesmo. Como referido anteriormente é necessário prevenir acesso indevido a terceiros, bem como possibilitar a investigação de incidentes ou falhas de segurança. Em Portugal existe a solução GESMAC (Gestão de Matéria Classificada) [[2](#two)], que cobre o ciclo de vida de uma matéria classificada, desde à sua criação à destruição da mesma, onde o objetivo principal é implementar mecanismos de segurança e controlo capaz de garantir a sua proteção.

A presente dissertação aborda o problema e possíveis soluções para documentos classificados, que após serem impressos num posto de controlo, podem ser apresentados a terceiros inapropriados ou terceiros credenciados que não têm maneira de validar a informação do documento.

Para resolver o problema do acesso a pessoas terceiras foram estudados métodos de marca de água visíveis e invisíveis que permitam validar a integridade do documento, e método de validação de documentos com base nos metadados e informação do documento, sendo este o escolhido para desenvolvimento e testes.

Este modelo assume que os documentos originais estão em formato PDF, contendo letras em determinadas posições e não *bitmaps*(formato imagem), cuja validação só será possível após scan sem OCR.

Os metadados ou metainformação do documento são um conjunto de letras obtidas na interseção de vários segmentos de reta aleatórios traçados invisivelmente na página inicial do PDF original. Esta informação é guardada em base de dados protegidas em edifícios com controlo de acesso, onde a visualização da verificação de documentos só é possível caso a aplicação tenha acesso à base de dados. Porventura caso este acesso à visualização por parte da aplicação seja sucedido, esta irá demonstrar as letras referidas nos metadados na posição do documento digitalizado, o que garante a validação da autenticidade do documento.

Adicionalmente os metadados podem conter a informação do utilizador que imprimiu o documento, a data da impressão do documento e o local do posto atual do documento. Reforçando assim a origem da fuga do documento, pois esta está sempre associada ao último local onde se encontra a matéria classificada.

# Contexto

Nesta secção irão abordar-se dois temas fundamentais para o desenvolvimento do sistema, sendo eles divididos em subsecções sendo elas códigos de barras e identificação de letras em posições específicas de um PDF.

* 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.

O código de barras e o QR Code foram escolhidos para serem adicionados ao documento para adicionar uma marca ao mesmo, para ser possível guardar dados externos através de identificadores correspondentes à base de dados, onde é possível verificar a metainformação do documento. Sendo o código de barras irá conter um conjunto de informações acerca do documento para a sua verificação rápida, tal como, data de impressão, utilizador que imprimiu ou assinou o documento, local onde o documento se encontra, caso este não corresponder o QR Code irá ter um conjunto de letras que são obtidas através de interseções de segmentos de reta do documento criadas invisivelmente no documento original.

Devido à enorme quantidade de códigos de barras, é necessário agregá-los e escolher os ideias para implementar na solução (adicionando no cabeçalho ou rodapé do documento, para garantir a sua descodificação). Esta informação é apresentada na Tabela 1 onde os critérios escolhidos para as comparações são as seguintes:

* Nome: designação do código de barras;
* Imagens exemplificativas, onde é possível ver como o código de barras é, ou seja, a sua imagem;
* Comprimento: variável ou fixo, essencial para que todos os documentos tenham o mesmo cumprimento do código de barras, sendo que este não é fator predominante;
* Uso: verificar onde se utiliza, para ver se é ideal para colocar num documento;

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

Existem várias maneiras de descodificar 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.
   * 1. Análise de Código de Barras

Através da análise da Tabela 1, verificou-se que o Post Code Australia, o Intelligent Mail Barcode, o Planet, o PostBar, o PostNet, o RM4SCC/KIX, o RM MailMark C e Code 49 não se enquadravam devido à sua tipologia, por terem tamanho de barras diferentes, o que impede o controlo da altura, o que poderá influenciar na descodificação do código de barras no documento, caso esta calhe em cima de texto. Os restantes enquadram na escolha, só que o barcode 39, 128 e 93 são os mais usados devido à sua capacidade de armazenamento, o que levou à criação da Tabela 2, onde se compara os mesmos, escolhendo o ideal.

Os critérios selecionados foram o tamanho máximo de caracteres que este pode guardar, tipos de dados que consegue codificar, vantagens e desvantagens do mesmo. Como o código de barras 128 permite codificar caracteres especiais, letras e números e a sua capacidade de dados é grande, optou-se por utilizar este, adicionando-o ao rodapé do documento, sendo que o código de barras 39 também seria uma possível escolha devido ao pouco espaço ocupado que iria ocupar no documento.

Tabela 1 - Código de Barras Lineares

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nome** | **Imagem** | **Fixo/**  **Variável** | **Tamanho das barras** | **Uso** |
| Post Code Austrália |  | Fixo | 4 | Correios Austrália |
| CodaBar |  | Fixo | 2 | Librarias |
| Code 25 |  | Variável | 2 | Librarias |
| Code 11 |  | Fixo | 2 | Telefones, já não se usa muito devido à sua antiguidade |
| Code 32 |  | Fixo | 2 | Farmácia |
| Code 39 |  | Fixo | 2 | Vários |
| Code 49 |  | Variável | Vários | Vários |
| Code 93 | Código de barras Código 93 - Simbologia nos códigos de barras | Cognex | Variável | Vários | Vários |
| Code 128 |  | Variável | Vários | Vários |
| EAN 2 |  | Variável | Vários | Revistas |
| EAN 5 |  | Variável | Vários | Livros |
| EAN-8 |  | Variável | Vários | Retalho |
| GS1-128 |  | Variável | Vários | Vários |
| GS1 Databar |  | Variável | Vários | Vários |
| Intelligent Mail barcode |  | Fixo | 4 | Correios USA |
| ITF-14 |  | Variável | 2 | Encomendas |
| ITF-6 |  | Variável | 2 | Vários |
| JAN |  | Variável | Vários | Usado no Japão |
| Planet |  | Variável | Grande/  Pequeno | Correios USA |
| Plessey | MSI Plessey | Barcode Scanner SDK | Manatee Works | Variável | 2 | Catálogos, revistas, inventários |
| PostBar |  | Fixo | 4 | Correios Canada |
| PostNET |  | Fixo | Grande/  Pequeno | Correios USA |
| RM4SCC/KIX |  | Fixo | 4 | Correios |
| RM Mailmark C |  | Fixo | 4 | Correios |
| Universal Product Code |  | Variável | Vários | Retalho |
| Telepen | Free Online Barcode Generator: Telepen Alpha | 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 |  | 43 | Letras e números | Ocupa menos espaço. | Não permite caracteres especiais tal como ç. |
| Code 128 |  | 48 | Letras, números e caracteres especiais | É mais eficiente a codificar texto.  Tem *checksum* | Precisa de espaço reservado no início, fim e *check symbols*. |
| Code 93 | Código de barras Código 93 - Simbologia nos códigos de barras | Cognex | 30 | Letras, números e caracteres especiais | Open Source | Não tem checksum |

* + - 1. QR Code

QR Code (Figura 1) 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 deteção de posição (4.1. Figura 2) que permite a descodificação em várias posições do scanner ou de um smartphone com câmara;
* padrão alinhamento (4.2. Figura 2) que corrige a distorção do QR Code em superfícies curvadas, o seu número varia consoante a informação contida;
* padrões de temporização (4.3. Figura 2) que permite obter o tamanho da matriz de dados;
* versão, que indica a versão do QR Code que está a ser utilizada (1. Figura 2);
* 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 3).

Qr code

Description automatically generated

Figura 1 - Exemplo QR Code

Graphical user interface, qr code

Description automatically generated

Figura 2 - Estrutura QR Code

Tabela 3 - 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 |

2.1.1.2. Código de Barras 128

O código de barras 128 (Figura 3) [3] 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 4).

Este é muito utilizado em aplicações de logística, como no controlo de stocks e na identificação de produtos em supermercados e lojas. Isso ocorre porque ele é capaz de codificar informações como o nome do produto, o 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. Este 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 a 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 averiguação 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 análise inserindo-o no rodapé ou cabeçalho do documento.



Figura 3 - Código de Barra 128

Shape

Description automatically generated with medium confidence

Figura 4 - Tabela ASCII

* 1. Processo de criação da metainformação

Um dos processos importantes é a criação do sistema de verificação da integridade de um documento que consiste em utilizar um conjunto de letras resultantes da interseção de segmentos de retas criados invisivelmente na primeira página do documento original. A grande pergunta está como isto funciona?, para explicar vai se abordar em dois métodos o primeiro na obtenção das letras, como se obtêm e qual as coordenadas da mesma, o segundo aborda a criação dos pontos para a criação dos segmentos de reta

* + 1. Criação de segmentos de reta e pontos

Os pontos são criados com base em nove pontos espalhados com distância igual em altura e largura, onde estes conseguem “preencher” o documento todo, estes são pré-definidos entre margens, ou seja, os pontos ficariam da seguinte maneira, sendo que superior significa 25% de altura, meio 50% e inferior 75% da altura do documento, e esquerda significa 25% do comprimento do documento, meio 50% e direita 75%:

* Margem superior esquerda/meio/direita;
* Margem meio esquerda/meio/direita,
* Margem inferior esquerda/meio/direita;

Tendo estes pontos pré-definidos, calcula-se os pontos da origem da reta com um conjunto de intervalo de espaçamento em altura e comprimento, onde este é escolhido aleatoriamente, cujo são denominados “r”: direita do ponto, “l”: esquerda do ponto e “b”: abaixo esquerda do ponto. Estes pontos referidos anteriormente irão dar origem a três pontos que irão dar origem a segmentos de retas que irão ligar-se entre si.

Em suma, tem-se nove pontos de segmentos de reta por cada ponto, totalizando oitenta e um pontos. Sendo que o objetivo da criação de segmentos é garantir que pelo menos uma letra se consiga obter através da interseção das mesmas com a letra.

* + 1. Obtenção das letras e respetivas posições num documento PDF

Num documento PDF com um conjunto de metainformação, consegue-se extrair ou procurar informação nesse documento, levando à possibilidade de extrair letras do mesmo. Sendo que através de leitores PDF não é possível a sua extração, levando à necessidade de criação ou utilização de código para a extração de texto e respetivas posições.

Um documento PDF de dimensão A4 tem as dimensões 595x842 points [4], podendo variar em ficheiros digitalizados ou outros, tendo a posição das letras aptada às dimensões do mesmo, sendo que as respetivas letras se encontram nesse intervalo (decimal). Sendo a sua obtenção de cima para baixo 0-842, e da esquerda para a direita 0-595. As letras ocupam espaço no documento, ou seja, nunca só ocupam um point, o que leva a criação de um intervalo por cada letra dividindo-se em começo e fim da palavra, onde estas se subdividem em x e y, perfazendo um intervalo de 4 valores. Como os valores são decimais, optou-se por arredondar para inteiros para a comparação dos valores de interseção, que também se arredondou para inteiro, ser mais fácil.

# Trabalhos Relacionados

Nesta secção irão ser abordados conjunto de trabalhos relacionados com a adição da marca de água num texto, sendo dividido em visíveis e invisíveis. Mas antes disso é necessário abordar o que é uma marca de água e porque a sua escolha para estudar métodos de validação em matérias classificadas.

* 1. Marca de Água

A marca de água é um método que permite dificultar a alteração do conteúdo de matérias classificados criando métodos que garantam a segurança das mesmas, dividindo-se em visíveis e invisíveis. Sendo que as visíveis contêm métodos capazes de verificar a informação das matérias classificado de uma forma implícita. Por outro lado as invisíveis, constituem métodos invisíveis a olho nu que garantem que estas não se alterem.

* 1. Marcas de Água Visíveis
     1. Substituição de Palavras

O documento quando está em circulação em formato papel ou eletrónico pode ser alvo de fuga de informação ou acesso ao documento por pessoas não autorizadas que 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 [5], 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 selecionar palavras aleatórias do documento. A avançada seleciona palavras com o mesmo tamanho que as palavras da lista de substituição.

Em suma, a marcação de água textual proposto por os autores do artigo [5] consiste em três passos, selecionar palavras do documento para substituição, selecionar as novas palavras e ocorrência de implementação no texto do documento.

Text

Description automatically generated

Equação 1 - Contagem de palavras

Text, letter

Description automatically generated

Equação 2 - Percentagem de ataque

* + - 1. Exemplo

Conforme mencionado no artigo [5] a Figura 5 demonstra um exemplo de categorias de substituição que podem ser usadas 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 5 - Exemplo da Substituição de Texto

* + - 1. 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 de 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 averiguação de alterações autorizadas.

* + - 1. 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.

* + 1. Marca de Água sem Mudança de Texto

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 [6] que consiste em 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 da generalidade da mudança textual por parte de pessoas não autorizadas em palavras menores do que quatro caracteres.

Para gerar amarca de água, é necessário ter acesso ao conteúdo do documento, e assim percorrer cada palavra encontrando as palavras superiores a quatro caracteres, guardando o primeiro caracter de cada palavra. No fim, juntam-se todas as marcas de água resultando na 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).

* + - 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.

* + - 1. 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 averiguação da integridade do documento.

* + 1. Documentos baseados em Eigenvalues

O artigo [7] apresenta um algoritmo que armazena as posições de todas as palavras de um documento numa 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.

* + - 1. Testes

No artigo [7] existem testes realizados a um documento com 47 linhas, não tendo disponibilizado o mesmo, apenas impressões de pequenas secções do mesmo.

A Figura 6 demonstra um excerto do texto antes de ter a marca de água, e na Figura 7 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. Dos resultados binários na Figura 8 (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 4, verifica-se que na coluna “*tamper* *detection*” está tudo a 100% que leva a concluir que o algoritmo detetou alterações em todos os casos.

Text

Description automatically generated

Figura 6 - Antes da Marca de Água

Text

Description automatically generated

Figura 7 - Depois da Marca de Água

Text

Description automatically generated

Figura 8 - Comparação Marca de água

Tabela 4 - Resultados Eigenvalues

Table

Description automatically generated

* + - 1. Vantagens

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

* + - 1. Desvantagens

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

* + 1. Marca de Água com Espaçamento

O artigo [8] 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 apuração da marca de água é necessário OCR (Optical Character Recognition), já que as modificações são difíceis de visualizar ao olho humano.

* + - 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 9, demonstra-se um exemplo. Neste exemplo a linha do meio começada por “Effects…” foi movida para baixo 1/300 inches, que equivale a 0.00846666667 cm.

A screenshot of a computer

Description automatically generated with medium confidence

Figura 9 - Exemplo Line-Shift

* + - 1. 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 10 é 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 10 - Exemplo Word-Shift

* + - 1. Character Coding

A letra escolhida é movida para cima ou baixo, enquanto as adjacentes não se alteram. A Figura 11 apresenta um exemplo. A primeira letra “e” da palavra “internet” foi movida para baixo 1/600 inches que equivale a 0.00423333333 cm.

A picture containing text

Description automatically generated

Figura 11 - Exemplo Character Coding

* + - 1. Vantagens dos Métodos

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

* + - 1. Desvantagens dos Métodos

Uso de OCR para análise de um 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.

# Algoritmo de Marcação e Análise de Documentos Impressos

Esta secção visa a explicação do algoritmo da marcação do documento impresso bem como ocorre a sua análise.

* 1. Processamento de Documentos

O algoritmo de processamento de documentos para a criação de um sistema de verificação da integridade do mesmo segue o diagrama de fluxo presente na Figura 12, descrito abaixo:

1. Escolha do Documento a processar;
2. Verificação da escolha do documento. Caso tiver sido processado, é necessário escolher outro, existe a distinção de ficheiros processados e não processados com a extensão criada com base no nome do ficheiro original “nome\_ficheiro\_watermark\_dd\_mm\_yy\_hh\_m\_ss”. Porventura se não existir, vai se passar para a inserção dos metadados do ficheiro na base de dados;
3. Obtenção de caracteres e posições do Documento. 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 13;
4. Criação de pontos para os segmentos de retas e cálculo do ponto de interseção. Utilizando as 9 posições pré-definidas calcula-se as 3 novas posições (r, l, b), com um intervalo de números inteiros, onde cada valor do x e y para cada posição irá ser escolhido aleatoriamente;
5. Criação do código de barras, contendo um identificador do documento onde este está inserido numa base de dados interna ou externa;
6. Inserção do código de barras ao ficheiro duplicado do original, dando origem a um novo ficheiro, este tem de ser obrigatoriamente aceite ou rejeitado por um utilizador.

Inicialmente os QR Code, nomeadamente os quadrados de posição, eram utilizados para a criação dos pontos para os segmentos de reta, espalhados pelos 9 pontos pré-definidos, contudo estes em ficheiros digitais eram difíceis de compreender ou até mesmo desvaneciam a cor, levando à sua remoção e adotando uma nova estratégia.

A nova estratégia envolve utilizar dois códigos de barras distintos, um em cima e outro em baixo do documento, onde permitia calcular a distorção em x e y no ficheiro digitalizado, calculando assim a sua proporção e adaptação das posições ao ficheiro digitalizado. Este calculo é feito através das posições dos códigos de barras 128 e 39.

A picture containing diagram

Description automatically generated

Figura 12 - Diagrama de Fluxo do Processamento de um Documento

Shape

Description automatically generated

Figura 13 - Representação dos Valores dos Caracteres

* 1. Verificação de Documentos Impressos

O algoritmo de verificação de documentos impressos o diagrama de fluxo presente na Figura 14, descrito abaixo:

1. Escolha do ficheiro a examinar, pode ser formato PDF eletrónico ou digital, nos dois casos o documento é convertido para imagem para a colocação das letras presentes na base de dados;
2. Leitura do código de barras para descodificar a informação contida nele, se o identificador(id) do ficheiro estiver na base de dados e tiver sido aceite ou rejeitado a marca de água. Devolve-se alguns metadados ao utilizador e este faz uma verificação rápida de um conjunto de informações acerca do documento tal como utilizador que imprimiu e posto atual;
3. Verificação de integridade do documento caso a verificação dê errado, ou o utilizador queira processeder à mesma. Consiste na comparação das letras que estão no documento com as da base de dados, através da conversão do documento pdf para imagem e aquando a colocação das letras e respetivos pontos converte-se de novo para PDF. Nesta abordagem verificou-se que pode existir vários pontos de interseção inseridos na mesma letra, só que com valores diferentes, o que dificultava na visualização do documento, pelo que posteriormente eliminou-se os duplicados demonstra várias letras na mesma zona, o que à posteriori levou à eliminação dos duplicados;

A picture containing diagram, text, line, plan

Description automatically generated

Figura 14 - Diagrama de Fluxo da Averiguação de um Documento

# Implementação

Nesta secção irá abordar-se o código feito, as aplicações criadas, a base de dados, a arquitetura e bibliotecas utilizadas.

* 1. Linguagens de Programação

Para o desenvolvimento do programa utilizou-se:

* C#: criação de Windows Forms (aplicações windows);
* Java: obtenção das letras e posições nos ficheiros PDF;
* SQL: criação da base de dados - ferramenta Microsoft SQL Server Management.
  1. Arquitetura

A arquitetura criada é baseada em três camadas (Figura 15):

* Servidor: destinado à base de dados que irá conter dados armazenados acerca dos documentos, códigos de barras, pontos dos segmentos de retas, posições dos caracteres no ficheiro de input, a criação da marca de água e dimensões do ficheiro bem como as dimensões do código de barras.
* 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.

Diagram

Description automatically generated

Figura 15 - Arquitetura Solução

* 1. Documentos

O algoritmo lida com três tipos de documentos (formato PDF):

- Documento sem marca de água;

- Documento com marca de água;

- Documento digitalizado com marca de água;

5.3.1. Conversão para Bitmap

Para a conversão do ficheiro PDF para Bitmap, para ser possível a inserção das letras obtidas pela interseção de segmentos de reta, utilizou-se o package FreeWare.Pdf2Png [9]. Código para a conversão demonstra-se na Figura 16.

|  |
| --- |
| **public** string Convert\_pdf\_png**(**string file\_name\_png**)**  **{**  var dd **=** File**.**ReadAllBytes**(**file\_name\_png**);**  byte**[]** pngByte **=** Freeware**.**Pdf2Png**.**Convert**(**dd**,** 1**);**  string**[]** filename **=** file\_name\_png**.**Split**(new[]** **{** ".pdf" **},** StringSplitOptions**.**None**);**  **if(!**File**.**Exists**(**filename**[**0**]** **+** ".png"**))**  File**.**WriteAllBytes**(**filename**[**0**]+** ".png"**,** pngByte**);**  **return** filename**[**0**]** **+** ".png"**;**  **}** |

Figura 16 - Código para criar uma Imagem da primeira Página do PDF

5.3.2. Obter Dimensões do PDF e da Imagem

A necessidade de obter as dimensões do PDF e da Imagem adveio dos ficheiros digitalizados, já que estes quando criados, dependendo da impressora, tem dimensões diferentes do original o que causou, alguma discrepância nas posições. O package utilizado para obter as dimensões do documento PDF é o iTextSharp.text.pdf [10]. O código apresenta-se na Figura 17.

|  |
| --- |
| **public** void GetDimensionsImage**(**string img\_file**)**  **{**  Bitmap bmp **=** **new** Bitmap**(**img\_file**);**  width\_bmp **=** bmp**.**Width**;**  height\_bmp **=** bmp**.**Height**;**  bmp**.**Dispose**();**  **}**  **public** void GetDimensionsDocument**(**string file\_name**)**  **{**  **using** **(**Stream inputPdfStream **=** **new** FileStream**(**file\_name**,** FileMode**.**Open**,** FileAccess**.**Read**,** FileShare**.**Read**))**  **{**  var reader **=** **new** PdfReader**(**inputPdfStream**);**  PdfReader**.**unethicalreading **=** **true;**  width **=** **(**int**)**reader**.**GetPageSize**(**1**).**Width**;**  height **=** **(**int**)**reader**.**GetPageSize**(**1**).**Height**;**  reader**.**Close**();**  reader**.**Dispose**();**  **}**  **}** |

Figura 17 - Código para obter as dimensões do documento PDF e da Imagem Convertida

5.3.2. Criação dos Pontos para o Sistema de Verificação

Este subcapítulo tem como intuito demonstrar as posições que irão dar origem aos segmentos de reta. A Figura 18 apresenta um pedaço de código que irá escrever as posições dos pontos numa imagem auxiliar (Figura 19), 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 19 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 20 representa a quantidade de espaçamento entre pontos no eixo dos x e y.

|  |
| --- |
| 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 18 - Código para Demonstração dos Pontos da Marca de Água

Text

Description automatically generated

Figura 19 - Representação dos Pontos

Text

Description automatically generated

Figura 20 - Espaçamento dos Pontos

5.3.3. Criação do Documento que Verifica a Autenticidade e Adaptação dos Pontos

Como referido no ponto 5.3.1 é necessário a criação da imagem do documento PDF, para ser possível obter as suas dimensões, desenhando por cima da imagem, guardando uma nova e só depois converte-se outra vez para PDF para ser possível a visualização ao utilizador na verificação profunda. Neste caso é necessário também calcular a escala do documento para fazer ajustes bem como adaptar os pontos nos ficheiros digitalizados.

O código para o uso da escala e respetiva criação da imagem com as letras na imagem original, com as diferenças e proporções nas coordenadas x e y calculadas, está presente na Figura 21.

|  |
| --- |
| **public** string DrawImage**(**List**<**string**>** return\_list**,** string f**,** int diff\_x**,** int diff\_y**,** double prop\_x**,** double prop\_y**,** int diff\_width\_doc**,** int diff\_height\_doc**,** int diff\_width\_bmp**,** int diff\_height\_bmp**,** double scale\_doc**)**  **{**  int sizeLetter **=** 10**;**  Point intersection**;**  Commom commom **=** **new** Commom**();**  img\_file **=** commom**.**Convert\_pdf\_png**(**f**);**  **using** **(**Bitmap bmp **=** **new** Bitmap**(**img\_file**))**  **{**  **using** **(**Graphics g **=** Graphics**.**FromImage**(**bmp**))**  **{**  Pen red **=** **new** Pen**(**Color**.**Red**,** 7**);**  Pen yellow **=** **new** Pen**(**Color**.**Yellow**,** 5**);**  int size\_cross **=** 25**;**  int width\_circle **=** 1**;**  int height\_circle **=** 1**;**  int startAngle **=** 0**;**  int sweepAngle **=** 360**;**  Font drawFont **=** **new** Font**(**"Arial"**,** sizeLetter**);**  SolidBrush drawBrush **=** **new** SolidBrush**(**Color**.**Blue**);**  SolidBrush orBrush **=** **new** SolidBrush**(**Color**.**Red**);**  double diff\_width **=** **(**double**)**diff\_width\_doc **/** diff\_width\_bmp**;**  double diff\_height **=** **(**double**)**diff\_height\_doc **/** diff\_height\_bmp**;**  **for** **(**int i **=** 0**;** i **<** return\_list**.**Count**;** i**++)**  **{**  string**[]** values **=** return\_list**[**i**].**Split**(**'|'**);**  string**[]** inter\_point **=** values**[**0**].**Split**(**','**);**  string ch **=** values**[**1**];**  int res\_x **=** Convert**.**ToInt16**(**inter\_point**[**0**]);**  int res\_y **=** Convert**.**ToInt16**(**inter\_point**[**1**]);**  **if** **(!**img\_file**.**Contains**(**"scan"**))**  **{**  **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**);**  intersection **=** **new** Point**(**new\_x**,** new\_y**);** //adjust point barcode  **}**  **}**  **else**  **{**  int n\_x **=** Convert**.**ToInt16**(**res\_x **\*** w **/** bmp**.**Width**);**  int n\_y **=** Convert**.**ToInt16**(**res\_y **\*** h **/** bmp**.**Height**);**  int s\_x **=** Convert**.**ToInt32**((**n\_x **-** diff\_x**)** **\*** prop\_x **);**  int s\_y **=** Convert**.**ToInt32**((**n\_y **-** diff\_y**-(**diff\_height\_doc**\***2**)+**1**)** **\*** prop\_y **);**  int new\_x **=** Convert**.**ToInt16**(**s\_x **\*** bmp**.**Width **/** w**);**  int new\_y **=** Convert**.**ToInt16**(**s\_y **\*** bmp**.**Height **/** h**);**  intersection **=** **new** Point**(**new\_x**,** new\_y**);** //adjust point barcode  **}**  g**.**DrawString**(**ch**,** drawFont**,** drawBrush**,** intersection**);**  g**.**DrawLine**(**red**,** intersection**.**X **-** size\_cross**,** intersection**.**Y**,** intersection**.**X **+** size\_cross**,** intersection**.**Y**);**  g**.**DrawLine**(**red**,** intersection**.**X**,** intersection**.**Y **-** size\_cross**,** intersection**.**X**,** intersection**.**Y **+** size\_cross**);**  g**.**DrawArc**(**yellow**,** intersection**.**X**,** intersection**.**Y**,** width\_circle**,** height\_circle**,** startAngle**,** sweepAngle**);**  **}**  filename **=** img\_file**.**Split**(new[]** **{** ".png" **},** StringSplitOptions**.**None**);**    bmp**.**Save**(**filename**[**0**]** **+** integrity\_extension **+** ".png"**);**  bmp**.**SetResolution**(**300**,** 300**);**  g**.**Dispose**();**  bmp**.**Dispose**();**  **}**  **}**  **return** filename**[**0**]** **+** integrity\_extension **+** ".png"**;**  **}** |

Figura 21 - Criação da Imagem que Visualiza as Letras do Sistema de Verificação da Base de Dados no Documento e Adaptação dos Pontos das Letras

* 1. Base de Dados

A base de dados SQL foi criada localmente, tendo um utilizador Figura 23 e base de dados, que permite a utilização do mesmo no 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 22, está presente um diagrama da base de dados que contém as tabelas usadas e respetivas conexões, de relação um para muitos, onde o ícone da chave significa chave principal.

O diagrama é constituído pelas seguintes tabelas:

* “document”: guarda características do documento (metados);
* “barcode”: guarda informações relativas ao código de barras;
* “watermark”: guarda as confirmações do documento com marca de água, se foi aceite ou não, para efeitos de rastreamento, e as posições dos códigos de barras 128 e 39;
* “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;
* “dimensions\_document”: guarda as dimensões do ficheiro original em formato PDF e bitmap;

Para a utilização da base de dados na aplicação utilizou-se o package “System.Data.SqlClient”, que permite fazer todo o tipo de ações numa base de dados.

A screenshot of a computer

Description automatically generated with low confidence

Figura 22 - Diagrama da Base de Dados

|  |
| --- |
| **CREATE** LOGIN antonio  **WITH** PASSWORD **=** 'antonio'**;**  **GO**  -- Creates a database user for the login created above.  **CREATE** **USER** antonio **FOR** LOGIN antonio**;**  **GO** |

Figura 23 - Criação do utilizador para a base de dados

* 1. Obtenção das letras e respetivas posições num documento PDF

Apesar de existirem alguns packages em C# que 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 mesma para ser usado na verificação de integridade, optou-se por utilizar o package Apache PDFBox [11] 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 se lê a primeira página do documento, sendo possível depois alterar para todas as páginas, contudo o tempo de processamento aumenta também. Para os valores lidos serem acedidos na aplicação desenvolvida 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 25, por vezes o algoritmo não lê certos caracteres como “-” e “ ’ ”, daí que faça substituição respetiva pelos caracteres, no primeiro por “-” e no segundo vazio (“ ”);

Contudo é necessário compilar o código desenvolvido 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 26.

Para demonstrar que o algoritmo obtém as letras precisamente, utilizou-se um leitor PDF denominado “PDF-Xchange Editor” que permite observar as posições do rato no documento como demonstra a Figura 27, a seta aponta para onde está o rato (representado por um círculo), dando os valores (71, 38), na base de dados presente na Figura 28, tem se que os valores de “start\_x” a 71 e o “stop\_y” a 37, mostrando que existe um bocado de discrepância de valores devido a arredondamentos.

|  |
| --- |
| 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 24 - Código para Abrir o Documento e Extrair Informação

|  |
| --- |
| 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 25 - Escrita das Letras e Respetivas Posições num Ficheiro Temporário

|  |
| --- |
| 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 26 - Código para a Execução do Ficheiro .Jar

A picture containing graphical user interface

Description automatically generated

Figura 27 - Exemplo de uma Letra e Respetiva posição no PDF com a Utilização da Ferramenta PDF-XChange Editor

Graphical user interface, application

Description automatically generated

Figura 28 - Valor Lido pelo Algoritmo Representado na Base de Dados

* 1. Código de Barras

5.6.1. Criação e Leitura

Para a criação do código de barras 128 e 39 foi utilizado o package ZXing.net [12], *open-source* (grátis e mantido por uma comunidade).

Como referido anteriormente o código de barras irá ter um identificador referenciador para o documento e um identificador para o código de barras, ou seja, serve para saber que código de barras corresponde a qual ficheiro, pois este tem ids diferentes e *timestamp* no documento diferentes.

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.

Para gerar os códigos barras utiliza-se o código da Figura 29, e para a leitura optou-se por ler o código de barras 128 inserido no rodapé, utilizando o código da Figura 30.

|  |
| --- |
| 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 **+** commom**.**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 29 - Criação dos 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 30 - Descodificação do Código de Barras 128

5.6.2. Posições

Com o intuito de retirar as posições dos códigos de barras 128 e 39, para o calculo da distorção do ficheiro original para o digital, utilizam-se o package ZXing.Net [12] para o código de barras 128 e o Bytescout [13] para o 39, não se utilizou o ZXing.Net para o 39, devido a erro de leitura nos ficheiros com escala variável, contudo as posições do package Bytescount são necessárias o ajuste para o x2 e y2, que foram calculadas com base no comprimento e altura do código de barras 39, já que este package não fornece a segunda localização. O código utilizado para obtenção das posições dos códigos de barras, descrito anteriormente apresenta-se na Figura 31.

Os valores dos ajustes das posições foram obtidos através do auxílio da visualização da Figura 32,

onde as bolas amarelas (representadas pelas setas pretas) e a cor-de-rosa representam as posições do código de barras 128 e 39 respetivamente.

|  |
| --- |
| 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**;**  bmp**.**Dispose**();**  **return** $"{x\_barcode\_pos}:{y\_barcode\_pos}:{x2\_barcode\_pos}:{y2\_barcode\_pos}:{x\_39}:{y\_39}:{x2\_39}:{y2\_39}"**;** |

Figura 31 - 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 32 - Guardar Imagem Auxiliar

Text

Description automatically generated

Figura 33 - Representação dos Pontos do Código de Barras

* 1. Interface Gráfica (GUI)

O objetivo deste subcapítulo é demonstrar a GUI do programa desenvolvido.

Começando por o menu principal (Figura 34), que é constituído por dois submenus sendo eles “processar” e “retificar”. “Processar” é a opção que cria um sistema de validação de autenticação de documentos, com a obtenção de letras de um conjunto de interseções. “Retificar” é a opção destinada para a verificação da autenticidade e integridade de um documento digital ou eletrónico ou impresso que porventura é digitalizado.

Para adicionar a pré-visualização do documento utilizou-se o Adobe Reader.dll [14], a janela é adaptada às dimensões do ficheiro, fazendo com que a interface gráfica deste *add-on* mude de posição. Também é possível fazer algumas coisas do *adobe acrobat*, tal como zoom in/out e imprimir.

“Processar”, escolhe-se o ficheiro, averiguando os seguintes critérios:

* Extensão “.pdf”: erro (Figura 35);
* Ficheiro processado?: erro (Figura 36);

Caso tiver sucesso é reencaminhado para o formulário da Figura 37, onde este contem três botões para exercer funções:

* Processar: começa a ação da criação de um sistema de validação de autenticação e integridade do documento;
* Aceitar: depois de finalizado o processamento do documento, informado através da Figura 39, exerce a ação de aceitar o documento (informando ao utilizador da conclusão da ação - Figura 41) para colocar na base de dados para efeitos de *tracking*. Contudo caso aborde o botão “aceitar” sem o ficheiro ter finalizado é informado que não o pode fazer através da mensagem da Figura 38. Caso aceite de novo o documento é informado que o mesmo já foi aceite (Figura 42);
* Rejeitar: Exerce a ação de rejeitar um documento processado para colocar na base de dados e gera um novo sem necessidade de clicar em “Processar”;

Por outro lado o menu “Retificar” é apresentado na Figura 45, onde escolhe o ficheiro a verificar averiguando os seguintes critérios:

* Extensão “nome\_ficheiro\_watermark\_dd\_mm\_yy\_hh\_m\_ss”;
* Leitura do código de barras 128 com sucesso;
* Extensão “.pdf”: erro (Figura 35);
* Documento aceite ou rejeitado na base de dados;

Caso tiver sucesso nestes critérios o utilizador é redirecionado para o formulário presente na Figura 45, onde cabe ao utilizador averiguar os metadados do documento, caso este pretender passar para averiguação da integridade pode clicar em “Análise Forense”, onde é demonstrado num ficheiro PDF as letras inseridas e os pontos de interseção dos segmentos de retas presentes na base dados (Figura 47)

A screenshot of a computer

Description automatically generated with medium confidence

Figura 34 - Menu

A screenshot of a computer

Description automatically generated with medium confidence

Figura 35 - Erro Extensão

A screenshot of a computer

Description automatically generated with low confidence

Figura 36 - Erro Ficheiro já Processado

Graphical user interface, text

Description automatically generated

Figura 37 - Menu Processar

Graphical user interface, application, Word

Description automatically generated

Figura 38 - Mensagem de Aviso, caso o utilizador clique em aceitar ou rejeitar

Graphical user interface, text, application

Description automatically generated

Figura 39 - Informação de que o processamento acabou e que é necessário aprovação ou rejeição do documento

A screenshot of a computer

Description automatically generated with medium confidence

Figura 40 - Conclusão do Processamento

Graphical user interface, application, chat or text message

Description automatically generated

Figura 41 - Informação Documento Aceite

Graphical user interface, text, application, chat or text message

Description automatically generated

Figura 42 - Informação Documento Já Aceite

A screenshot of a computer

Description automatically generated with low confidence

Figura 43 - Informação do Erro na seleção de ficheiro

Graphical user interface, text, application

Description automatically generated

Figura 44 - Erro Leitura Código de Barras

A close-up of a document

Description automatically generated with medium confidence

Figura 45 - Verificação da Autenticidade de um Documento

A screenshot of a message

Description automatically generated with low confidence

Figura 46 - Informação do Procedimento à Verificação da Integridade

A picture containing text, screenshot, font, software

Description automatically generated

Figura 47 - Resultado da Verificação da Integridade

# Testes

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 poder ser colocada torta no digitalizador, sendo que esta é abordada na secção anterior. Nesta secção aborda-se o comportamento do algoritmo em situações de mudança de escala e alteração de letras no ficheiro original.

6.1. Documentos Digitais com Rotações

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

A Figura 48, demonstra um documento digitalizado relativamente torto, cujo deu entrada no sistema, depois de algum processamento de imagem o resultado apresenta-se na Figura 49, demostrando que o algoritmo conseguiu endireitar o documento, contudo podem existir casos em 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 50 , do resultado da verificação de integridade com o original (Figura 51), pode-se afirmar que a discrepância entre pontos é muito pouca 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.

Text, letter

Description automatically generated

Figura 48 - Documento Digital Torto

Text, letter

Description automatically generated

Figura 49 - Documento Digital Torto Endireitado

Chart, scatter chart

Description automatically generated

Figura 50 - Resultado da Averiguação da Integridade do Documento Torto

Scatter chart

Description automatically generated with low confidence

Figura 51 - Resultado da Averiguação da Integridade do Documento (Formato Eletrónico)

6.2. Mudança de Escala no Documento

A escala ideal para testar a robustez do algoritmo é entre 80-95%, sendo que abaixo de 50%, o documento torna-se ilegível a olho nu (Figura 52), sendo necessário o uso de ferramenta de leitura de PDF para realizar zoom.

O ficheiro utilizado para alteração da escala está apresentado na Figura 53, tendo o resultado da verificação na Figura 54.

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

Na Figura 55 apresenta-se uma captura de ecrã do ficheiro da Figura 53, com uma escala de 80%, sendo que o resultado obtido da verificação se demonstra na Figura 56. Através da comparação de pontos entre a verificação da integridade da Figura 56 e Figura 54, conclui-se que o algoritmo conseguiu adaptar as posições no ficheiro com escala variável.



Figura 52 - Documento Modificado com 50% de Escala

Text

Description automatically generated

Figura 53 - Documento Original

Scatter chart

Description automatically generated with low confidence

Figura 54 - Resultado da Averiguação Profunda da Integridade do Documento Original

Graphical user interface, text

Description automatically generated

Figura 55 - Documento Modificado com 80% de Escala

Graphical user interface, application

Description automatically generated

Figura 56 - Resultado da Verificação Profundo do Documento Modificado com 80% de Escala

6.3. Modificações no Documento

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 57 e os metadados da mesma na Figura 58. 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 59.

Criaram-se três ficheiros idênticos ao da Figura 57, com alterações, sendo eles:

1. Substituição de palavra: para a substituição resolveu-se substituir a palavra “importante” por “necessário” presente no retângulo da Figura 60. Pode-se conferir que a letra “n” está por cima da letra “r”, o que leva aferir que o documento naquela zona foi alterado;
2. Eliminação de palavras: eliminação do último paragrafo do documento da Figura 57 presente no retângulo da Figura 61. Averigua-se através da letra “D” inserida numa zona branca do documento, que houve eliminação de pelo menos uma palavra.
3. Eliminação de espaçamento: removeu-se um espaçamento do documento da Figura 57, representado pelo retângulo na Figura 62. Através da análise do documento, observa-se que existiu uma translação no eixo do y (de cima para baixo) - representada pela seta na Figura 62, inferindo que ocorreram 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 existam adições ou eliminações de palavra consegue detetar como é caso do ponto 2 e 3, referidos anteriormente.

A picture containing text, font, screenshot, letter

Description automatically generated

Figura 57 - Documento Exemplificativo

|  |
| --- |
| 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 58 - Código para a Inserção de Metadados

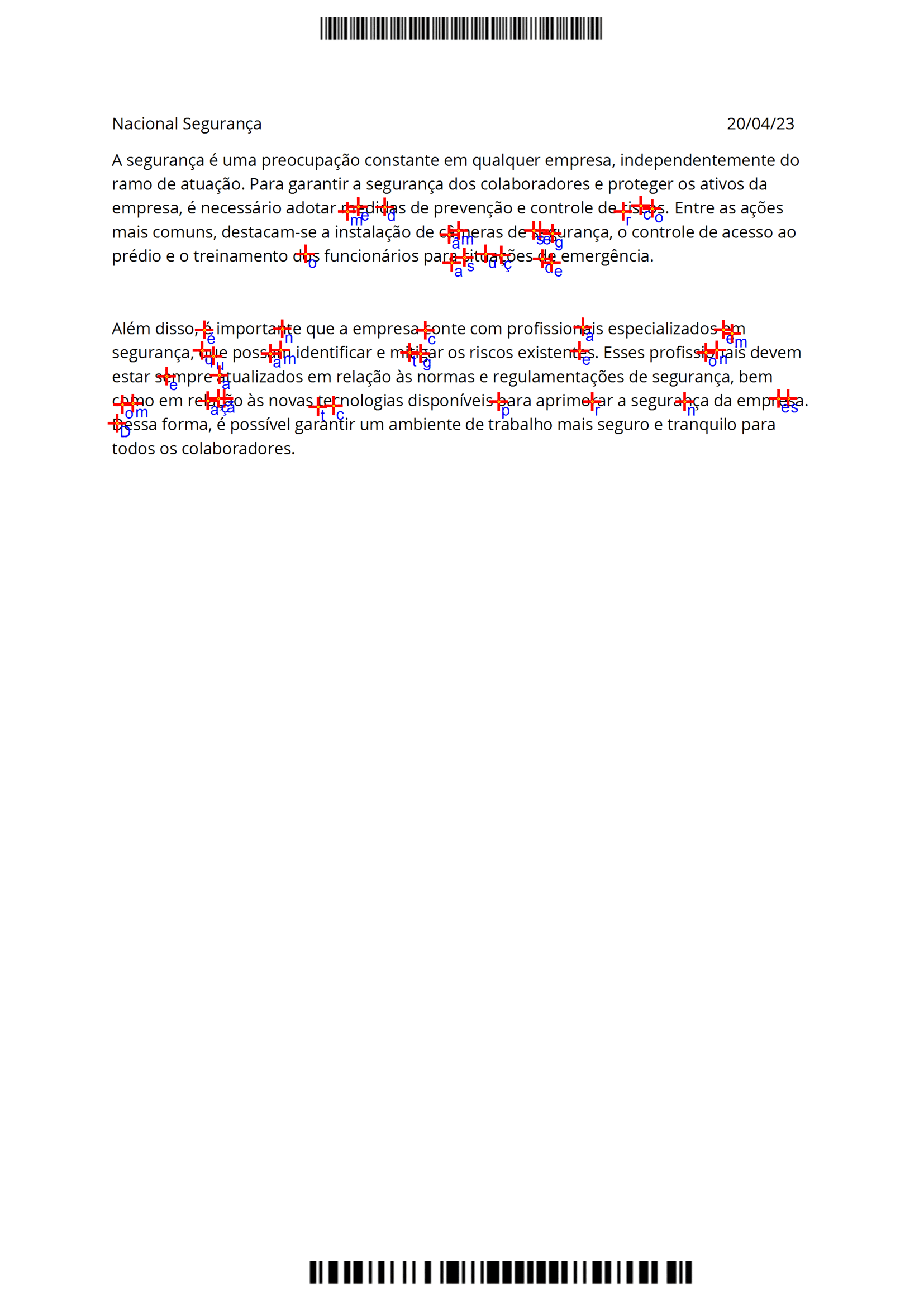


Figura 59 - Resultado da Verificação da Integridade do Documento Exemplar

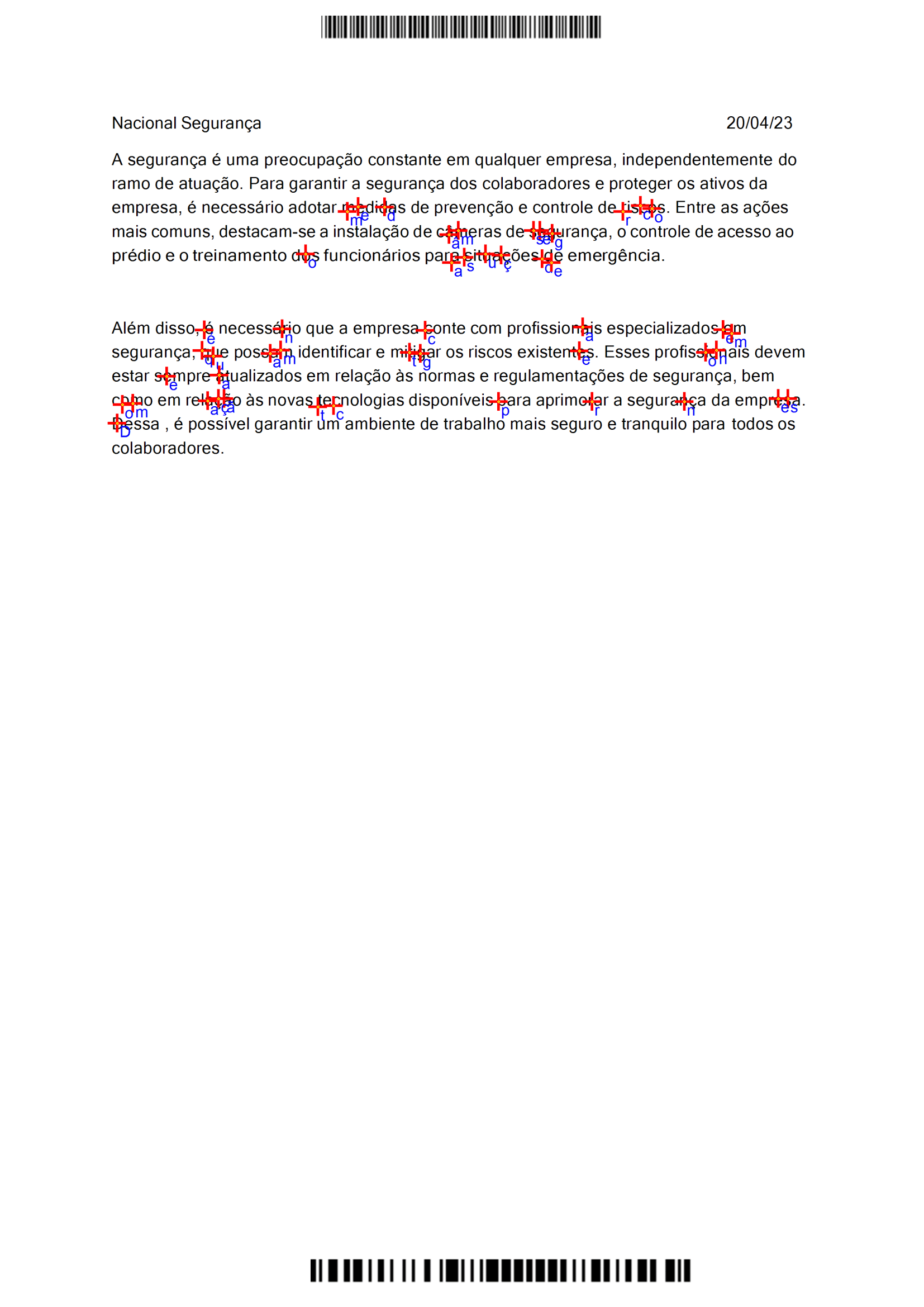


Figura 60 - Resultado da Verificação da Integridade do Documento Exemplar com Substituição da Palavra

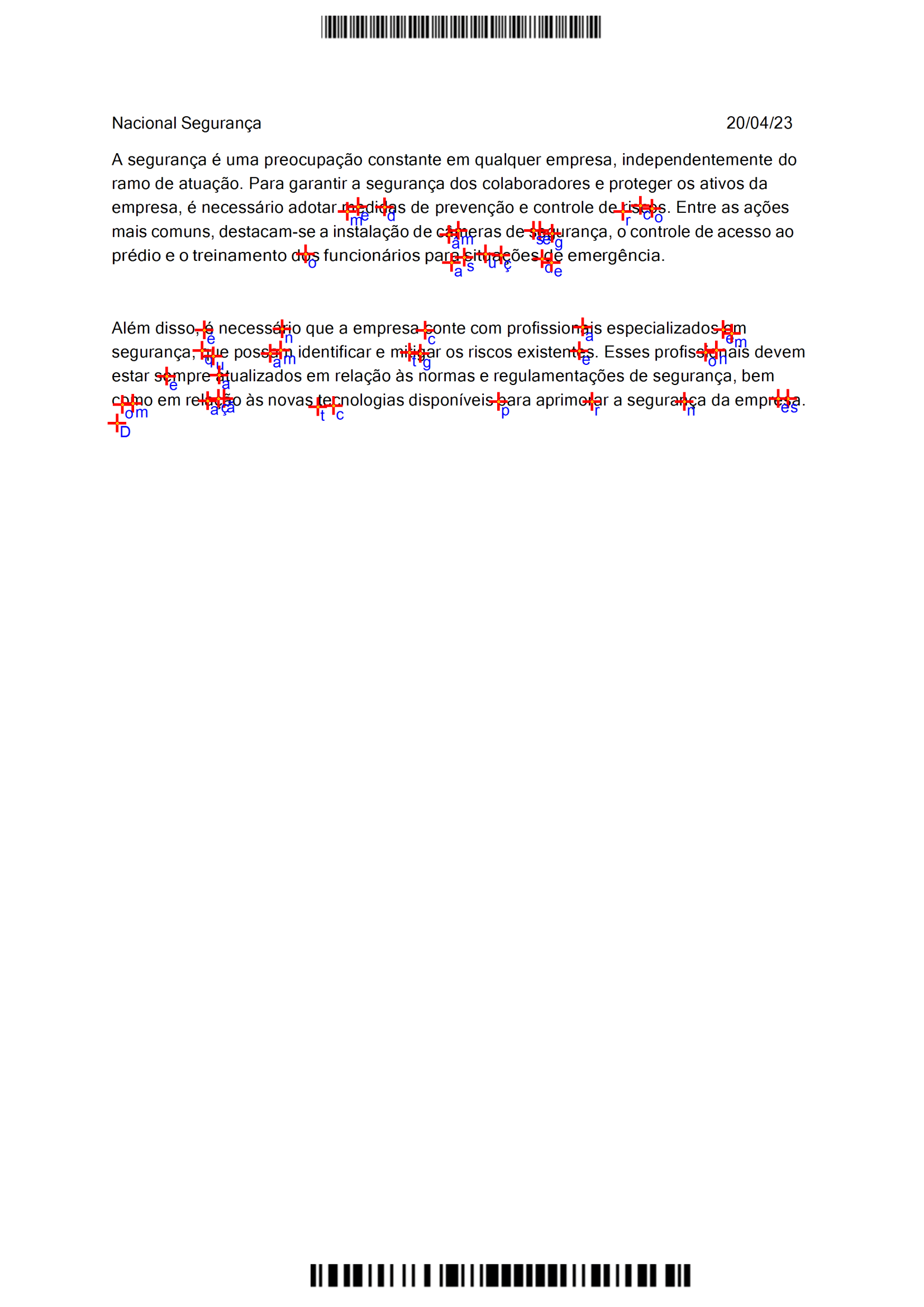


Figura 61 - Resultado da Verificação da Integridade do Documento Exemplar com Eliminação de Texto

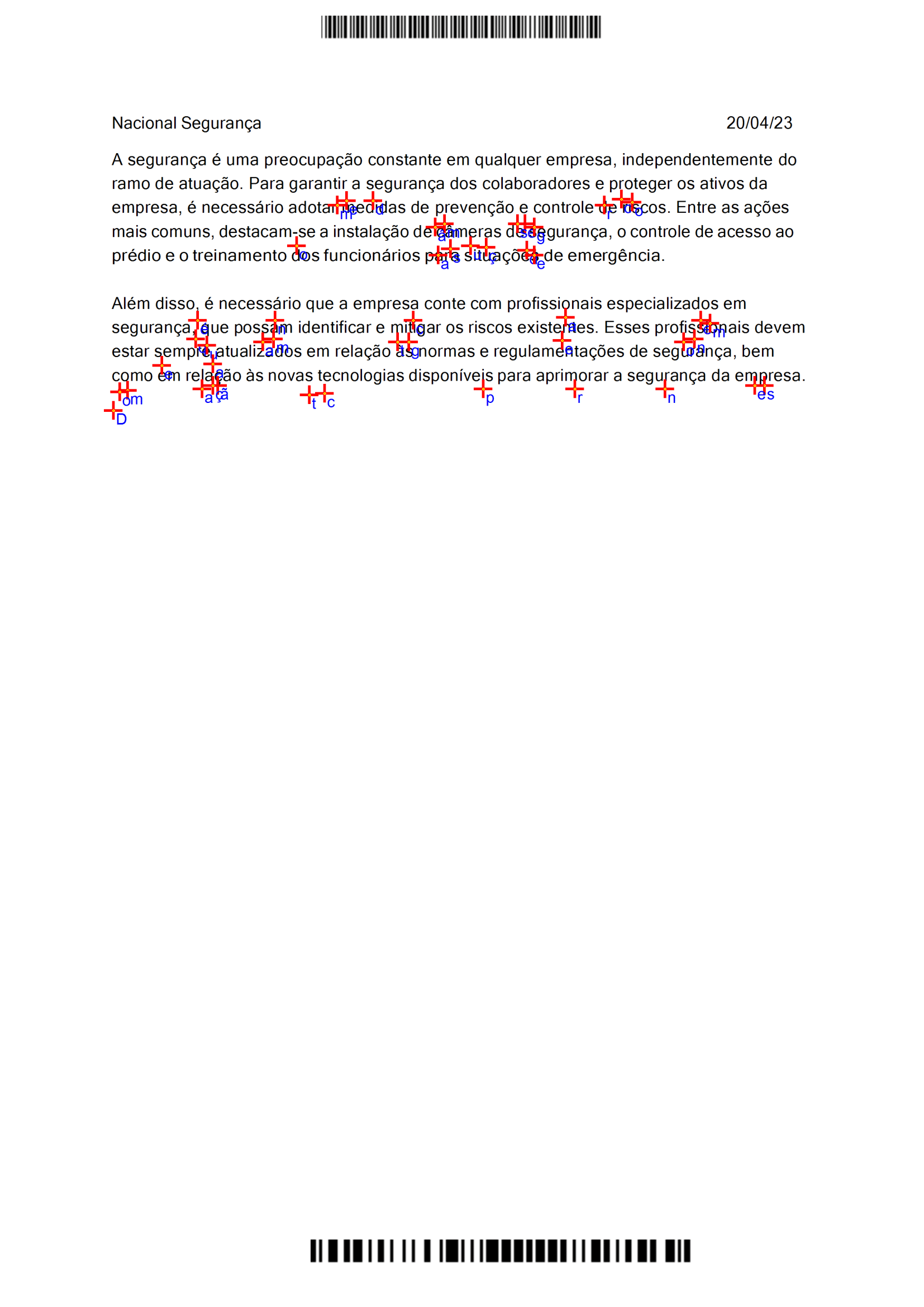


Figura 62 - Resultado da Verificação da Integridade do Documento Exemplar com Eliminação de Espaçamento

# Conclusões

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 poder ser 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 final desenvolvido permite assim concluir os objetivos inicias de criação de um código de barras para validação rápida de um documento e criação de marca de água segura capaz de garantir que o 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 para a primeira página do documento, para fins demonstrativos, ele pode ser modificado no futuro, para todas as páginas, levando ao aumento de processamento e reconhecimento.

7.1. Trabalho Futuro

Para futuro trabalho, poderá aumentar-se o espectro do algoritmo para mais folhas do documento e/ou mais pontos de letras. Verificar, se é 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 se tornar mais facilitada.

Adicionar um mecanismo que saliente as letras e zonas alteradas no documento, para que este seja mais percetível e intuitiva, já que é necessário ler o texto todo para encontrar a zona alterada.

# Referências

[1] “Conhecimento é Poder - Significado, origem e autores.” https://conceitosdomundo.pt/conhecimento-e-poder/ (accessed Apr. 31, 2023).

[2] “IDENTIDADE E BIOMETRIA - Gesmac: Gestão de Matéria Classificada.” http://www.sinfic.pt/ib/displayconteudo.do2?numero=34595 (accessed Apr 31, 2023).

[3] “Code 128 - Wikipedia.” https://it.wikipedia.org/wiki/Code\_128 (accessed Jan. 19, 2023).

[4] “A4 size in point. Read here what the A4 size is in po.” https://www.a4-size.com/a4-size-in-point/ (accessed Jun. 01, 2023).

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] “Freeware.Pdf2Png.” https://www.nuget.org/packages/Freeware.Pdf2Png/ (accessed Jun. 01, 2023).

[10] “iTextSharp | iText PDF.” https://itextpdf.com/products/itextsharp (accessed Jun. 01, 2023).

[11] “Apache PDFBox | A Java PDF Library.” https://pdfbox.apache.org/ (accessed Feb. 17, 2023).

[12] “micjahn/ZXing.Net: .Net port of the original java-based barcode reader and generator library zxing.” https://github.com/micjahn/ZXing.Net (accessed Jun. 01, 2023).

[13] “Bytescout BarCode Reader is a Desktop App to Read QR Code and Other Barcodes from Image Files (JPG, PNG, TIFF, GIF), PDF Documents and Webcam - ByteScout.” https://bytescout.com/products/enduser/misc/barcodereader.html (accessed May. 01, 2023).

[14] “(2) Making PDF Viewer in C#.net - YouTube.” https://www.youtube.com/watch?v=pQjskYF9yNY (accessed May. 01, 2023).

Anexo

1. Mudança de Escala do Documento

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

O código apresentado na Figura 63, 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 escolha desejada. O novo documento com a escala criada vai ser colocada na margem esquerda, ou seja, o documento vai ter as dimensões do original, mas não vai preencher a página toda. Na secção 6.1 apresenta-se um documento exemplo com 80% de escala (Figura 55).

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 64). Para finalizar adaptam-se as posições de interseção com base na percentagem de escala como demonstra a Figura 65.

|  |
| --- |
| 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**();**  writer**.**Close**();** |

Figura 63 - Mudança 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 64 - 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 65 - Adaptação do Cálculo das Posições