



Universidade de Aveiro

Ano 2021/2022

**António Jorge
Ferreira Ramos**

**Validação da autenticidade de documentos
impressos**



Universidade de Aveiro
Ano 2021/2022

**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 pelo apoio e auxílio prestado.

palavras-chave

Marca de água, códigos de barras, textual watermark, documentos impressos, segurança, processamento de imagem, qrcodes, verificação de integridade

resumo

keywords

Watermark, barcodes, textual watermark, printed documents, security, image processing

abstract

Índice

Índice.....	ii
Lista de tabelas.....	iii
Lista de Figuras.....	iv
1. Introdução.....	1
1.1. Descrição do sistema existente	1
1.2. Objetivos.....	1
2. Estado da Arte	3
2.1. Código de barras	3
2.1.1. Análise de cada tipo de código de barras	3
2.2. Texto com watermark	5
2.2.1. Ataque zero com mudança de texto	5
2.2.1.1. Exemplo	6
2.2.1.1. Vantagens.....	6
2.2.1.2. Desvantagens.....	6
2.2.2. Ataque zero sem mudança no texto	6
2.2.2.1. Vantagens.....	7
2.2.2.2. Desvantagens.....	7
2.2.3. Documentos texto baseados em Eigenvalues.....	7
2.2.3.1. Testes.....	7
2.2.3.4. Vantagens.....	8
2.2.3.5. Desvantagens.....	8
2.2.4. Line-Shift Coding	8
2.2.4.1. Word-Shift Coding	9
2.2.4.2. Character Coding.....	9
2.2.4.3. Vantagens dos métodos	9
2.2.4.4. Desvantagens dos métodos.....	9
2.2.5. ECL zero-based watermarking	9
2.2.5.1. Vantagens.....	9
2.2.5.2. Desvantagens.....	9
3. Qrcode	11
4. Código de barras 128.....	13
5. Entropia documento com watermark.....	15
6. Métodos implementados.....	17
6.1. Primeira implementação	17
6.2. Análise de quantidade de qrcode para a verificação de integridade.....	21
6.3. Verificação de integridade com 9 qrcode com processamento de imagem.....	22
7. Proposta de desenvolvimento	26
8. Arquitetura da solução.....	28
8.1. Estrutura do documento	28
8.2. Base de dados	29
8.3. Criação do código de barras e Qrcode	29
8.4. Operações a desempenhar pelo utilizador.....	30
8.5. Processamento do documento.....	30
8.6. Retificar documento.....	30
8.7. Verificação de integridade	30
9. Testes.....	34
10. Conclusões e futuro trabalho.....	36
11. Referências.....	38
Apêndice A.....	40
OCR.....	40
Iron Software.....	41

Lista de tabelas

Tabela 1 - Estudo de código de barras lineares	3
Tabela 2 - Possíveis escolhas	4
Tabela 3 - Resultados Eigenvalues	8
Tabela 6 - Classificação de erros do Qrcode	11
Tabela 4 - Mudança de cor	18
Tabela 5 - Performance	20

Lista de Figuras

Figura 1 – Exemplo de um ataque zero	6
Figura 2 - Esquema de geração e extração da watermark	7
Figura 3 - Antes do Watermark	7
Figura 4 - Depois do Watermark	7
Figura 5 - Comparação Watermark	8
Figura 6 - Exemplo line-shift	8
Figura 7 - Exemplo word-shift	9
Figura 8 - Exemplo Character Coding.....	9
Figura 9 - Funcionamento do algoritmo	9
Figura 10 – Exemplo de um Qrcode.....	11
Figura 11 - Estrutura Qrcode.....	11
Figura 12 - Código de barra 128	13
Figura 13 - Tabela ASCII.....	13
Figura 14 - Zona livre.....	18
Figura 15 - Zonas críticas.....	18
Figura 16 - Ficheiro exemplar.....	19
Figura 17 - Ficheiro processado	19
Figura 18 - Resultado Qrcode	20
Figura 19 - Resultado consola.....	20
Figura 20 - Verificação de integridade pretendida	21
Figura 21 - Segmentos de reta no ficheiro PDF	22
Figura 22 - Qrcode com processamento de imagem	23
Figura 23 - Pós processamento.....	24
Figura 24 - Resultado processamento de imagem de 9 qrcodes	24
Figura 25 - Arquitetura solução proposta.....	28

1. Introdução

No âmbito da tese de Mestrado de Engenharia Informática da Universidade de Aveiro, pretende-se que seja desenvolvido um módulo aplicacional a integrar com uma solução de gestão de informação classificada desenvolvida pela empresa iCreate Consulting.

Esta solução prevê a possibilidade de impressão de documentos classificados, devendo ser implementados mecanismos de marcação dos mesmos no sentido de garantir a sua autenticidade.

1.1. Descrição do sistema existente

Estes documentos vão ser tratados em áreas de segurança física destinadas para o efeito, onde para aceder o utilizador têm que ser pessoas idóneas (de confiança) e acesso a documentos com informação classificada deverá ter um certificado periódico.

Cada utilizador tem acesso a um determinado Posto de Trabalho, e aos documentos que foram destinados a esse mesmo Posto de Trabalho.

Por outro lado, esse utilizador só tem permissão para ver documentos até um nível de classificação para o qual está autorizado: por exemplo se só pode ver documentos até ao nível Confidencial, não terá acesso aos documentos Secretos ou de nível de classificação superior.

Apesar de ter acesso a documentos localizados num determinado posto, deve respeitar o princípio da “Necessidade de Conhecer”, não devendo consultar indiscriminadamente os documentos a que tem acesso.

1.2. Objetivos

Para assegurar a sua autenticidade propõe-se dois métodos de segurança no documento.

O primeiro é criar um código de barras que permita validar de uma forma explícita se o documento teve origem no sistema, que possível terá mais informações que caracterize o documento, nomeadamente marcas de segurança. A ideia deste primeiro é passar uma pistola que leia o código de barras e averiguar se o código de barras é válido e se a informação do código de barras corresponde ao documento.

O segundo é criar uma watermark segura, única, por documento, gerado por um conjunto de informações do documento, que permitam auxiliar em aspetos mais forenses.

1.3. Estrutura do documento

2. Estado da Arte

Este capítulo tem como objetivo agregar informações relativas ao tema da tese, sendo dividido em dois subcapítulos, códigos de barras e *text watermarking*. O subcapítulo código de barras servirá para verificar se um documento é fidedigno, isto é comparar a assinatura do documento com a informação que estará no código de barras. O *text watermarking* verificará se a informação do documento foi modificada ou se é o original através de métodos de *watermarking*.

2.1. Código de barras

Código de barras é um modo de representar informação num estado visual, que possa ser visível sem de ter a necessidade de escrever texto. É classificado em duas categorias, linear (1D) e 2D. Um dos requisitos da empresa era que o código de barras não ocupar muito espaço no documento tendo espaço livre no cabeçalho e no rodapé como a categoria linear ocupa menos espaço, mas tem um comprimento variável, isto é o documento depende do conjunto de informação que estiver contido no código de barras, optou-se por abordar mais esta categoria. Como existem vários tipos de códigos de barras nesta categoria, criou-se uma tabela (1), para comparar os mesmos e escolher o que se vai usar no desenvolvimento para inserção no documento. Esta tabela é constituída por colunas nome – designação do código de barras, imagens de exemplo (retiradas do Wikipédia), se o comprimento é variável ou fixo, o seu uso, e se é necessário 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 bem gerada.

Existem duas maneiras de ler um código linear, com uma máquina própria (leitor de barras de barras) ou com um smartphone que tenha uma câmara e em alguns casos é necessário ter aplicações disponíveis na loja do sistema operativo (android e iOS) para determinados códigos de barras.







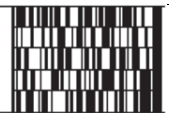

A empresa não permito uso de telemóveis pessoais dentro do estabelecimento, então descartou-se a hipótese de usar os smartphones e usar leitores de barras especiais, tendo um custo extra para a empresa. A solução foi criar uma maneira no algoritmo reconhecer o código de barras para a leitura e extrair dados.

2.1.1. Análise de cada tipo de código de barras

Este subcapítulo tem como objetivo expor a análise de vários códigos de barras lineares, cujos parâmetros de escolha foram comprimento (variável ou fixo), tamanho das barras, uso e se é necessário comprar um leitor próprio para a sua leitura. A análise recolhida está apresentada na tabela 1.

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 com a 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	Necessário Leitor (S/N)
Post Code Austrália		Fixo	4	Correios Austrália	S
CodaBar		Fixo	2	Librarias	S
Code 25		Variável	2	Librarias	S
Code 11		Fixo	2	Telefones, já não se usa muito devido à sua antiguidade	S
Code 32		Fixo	2	Farmácia	N
Code 39		Fixo	2	Vários	S
Code 49		Variável	Vários	Vários	S
Code 93		Variável	Vários	Vários	N









Code 128		Variável	Vários	Vários	N
EAN 2		Variável	Vários	Revistas	N
EAN 5		Variável	Vários	Livros	N
EAN-8		Variável	Vários	Retalho	N
GS1-128		Variável	Vários	Vários	S
GS1 Databar		Variável	Vários	Vários	N
Intelligent Mail barcode		Fixo	4	Correios USA	S
ITF-14		Variável	2	Encomendas	N
ITF-6		Variável	2	Vários	N
JAN		Variável	Vários	Usado no Japão	S
Planet		Variável	Grande/Pequeno	Correios USA	S
Plessey		Variável	2	Catálogos, revistas, inventários	S
PostBar		Fixo	4	Correios Canada	S
PostNET		Fixo	Grande/Pequeno	Correios USA	S
RM4SCC/KIX		Fixo	4	Correios	S
RM Mailmark C		Fixo	4	Correios	S
Universal Product Code		Variável	Vários	Retalho	N
Telepen		Variável	2	Librarias da Inglaterra	N

Tabela 2 - Possíveis escolhas

Nome	Imagem	Tamanho máximo de dados	Tipos de dados	Vantagens	Desvantagens
Code 39		43	Letras e números	Ocupa menos espaço.	Não processa caracteres especiais tal como ç.

Code 128		48	Letras, números e caracteres especiais	Faz encoding de strings muito eficiente. Tem <i>checksum</i>	Precisa de espaço reservado no início, fim e <i>check symbols</i> . Tem pouco limite de informação que pode guardar.
Code 93		30	Letras, números e caracteres especiais	Open Source	Não tem checksum

2.2. Texto com watermark

A *watermark* é um método que permite a salvaguarda de documentos originais e verificação da autenticidade de documentos ou imagens. É usada nos dias de hoje como por exemplo nas notas europeias, para impedir a impressão de notas não autorizadas. Existem muitos métodos e ataques de *watermarking*, neste documento vai ser apenas abordado o método *text watermark* e os métodos ataques zero devido às necessidades apresentadas pela empresa.

2.2.1. Ataque zero com mudança de texto

O objetivo dos ataques zero é aumentar a fragilidade da *watermark* do documento contra ataques de pessoas não autorizadas, normalmente denominados por hackers. Estes ataques tendem a alterar o texto do documento sem alterar a *watermark*. Estes ataques normalmente são de inserção, reordenação e remoção de texto.

Uma das soluções propostas pelos autores do artigo [1] foi ataque de substituição que explora a troca de ordem das palavras no texto. Esta solução é constituída por dois elementos. No primeiro é necessária uma lista de palavras que se pretende substituir no documento. O segundo é uma lista de palavras que vão estar depois da substituição do documento.

Em suma, escolhe-se um conjunto de palavras no documento que queremos substituir por novas palavras. 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.

Existem duas maneiras de escolher a lista de palavras para a substituição, normal e avançada. A normal consiste em selecionar palavras aleatórias do documento. A avançada consiste em selecionar as palavras com mesmo tamanho do que as palavras da lista de substituição.

Em conclusão, o *watermarking* textual proposto por [1] consiste em 3 passos, selecionar palavras do documento para substituição, selecionar as novas palavras e ocorrência de implementação do texto.

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

Equação 1

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

Equação 2

2.2.1.1.Exemplo

Os autores do artigo [1] forneceram um exemplo de tipos de substituição como demonstra a figura 1. Podemos verificar que se substitui o pronome mais comum em inglês “The” por “close” na substituição normal e “job” na avançada. Este método permite assim ter palavras-chave ou informação principal que não se pretende perder numa maneira que só quem vai substituir as palavras sabe as originais.

Original Text: The historic art and architecture of Venice is the key to the city's popularity as a tourist destination, but every two years, from May to October, it also becomes a place of pilgrimage for contemporary art lovers.
Normal Replacement Attack: Close historic art and architecture of Venice is close tech to close city's popularity as a building destination, but every two years, from May to October, it also becomes a defence of pilgrimage for contemporary art lovers.
Advanced Replacement Attack: Job historic art and architecture of Venice is job run to job city's popularity as a brokers destination, but every two years, from May to October, it also becomes a there of pilgrimage for contemporary art lovers.

Figura 1 – Exemplo de um ataque zero

2.2.1.1.Vantagens

As vantagens deste método são permitir alterar as palavras no texto através do utilizador e o algoritmo permitir saber a zona em que se alterou o documento.

2.2.1.2.Desvantagens

As desvantagens deste método são na leitura de texto, as mudanças podem não fazer sentido para a pessoa que não alterou o documento, e não conseguir captar métodos de cópia de texto, por exemplo criar um documento totalmente novo com o mesmo texto.

2.2.2. Ataque zero sem mudança no texto

A solução proposta pelos autores do artigo [2] consiste em usar as características do documento para gerar a *watermark* com o seguinte layout “autor:watermark:data:tempo” registada pela uma autoridade certificada (CA). O objetivo principal deste método é usar as palavras que tenham um tamanho maior do que quatro caracteres para proteger contra possíveis ataques.

A escolha de palavras maiores do que quatro caracteres é devido a generalidade dos ataques serem direccionados a palavras com tamanho menor do que quatro caracteres.

Para gerar a *watermark*, é necessário saber todas as frases de um documento, e assim percorrer cada palavra achando as palavras maiores do que quatro caracteres, quando acabar, guardar o primeiro caracter de cada dessas palavras e gerar a *watermark*.

Por exemplo na seguinte frase: “O José gosta muito de ler” a *watermark* ficava JGM (José Gosta Muito). Depois de percorrida todas as frases juntasse os *watermark* de cada frase, dando o *watermark* final.

Na figura 2 demonstra-se como é que a *watermark* é gerada e extraída. Deteta-se um ataque quando o *pattern* da *watermark* tem pelo menos ser 70% igual ao documento comparativo.

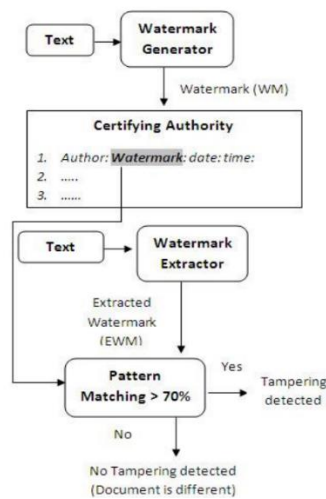


Figura 2 - Esquema de geração e extração da watermark

2.2.2.1.Vantagens

Tem uma entidade certificadora que vai possuir a *watermark* original, que vai ser a única pessoa a 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.

2.2.3. Documentos texto baseados em Eigenvalues

A solução proposta pelos autores do artigo [3] envolve um algoritmo que guarda as posições de todas as palavras do documento numa matriz e o peso das mesmas em ASCII. Um documento normalmente é constituído por palavras, espaçamento, números e pontuações. Os autores consideram cada contagem para calcular o peso ASCII para gerar o esquema da *watermark* com base numa chave privada, que um utilizador autenticado vai verificar o documento recebido.

2.2.3.1. Testes

Os autores do artigo [3] fizeram testes com base num documento com 47 linhas. A figura 3 demonstra um excerto do texto antes de ter *watermark*. A figura 3 demonstra o texto depois da geração da *watermark*. Para simular um ataque foi alterada a palavra “OFF” para “ON” numa zona do texto e pelos resultados binários na figura 3, em primeiro está o original e de seguida o alterado, retira-se que os números são diferentes. Uma pequena alteração da palavra levou à mudança de 11 bits. Os autores fizeram testes na mudança de vogais, consoantes, caracteres especiais, palavras, números pontuações, e alterações aleatórias no texto, obtiveram os resultados da figura 5.

Na coluna “tamper detection” da tabela 3 verifica-se que em todo o tipo de ataques, o algoritmo deteta a sua alteração sendo assim confiável.

point in time receives a spe
rally speaking, the device 1
made it available. Unauthori
de the tampered version av
istance, a hacker operating
re image 111 as it is made a
nper with the image 111 as if
a tampered version of the
a variety of means.
140 extracts the verificatio

(a) Before watermarking

Figura 3 - Antes do Watermark

point in time receives a spe
rally speaking, the device 1
made it available. Unauthori
de the tampered version av
istance, a hacker operating
re image 111 as it is made a
nper with the image 111 as if
a tampered version of the
a variety of means.
140 extracts the verificatio

(b) After watermarking

Figura 4 - Depois do Watermark

Before Tampering Secret key generated is,
0000000000001001011100100000000000001000000101101
00000000000010111101100010100000000001111001111100
01000000000011100100010011111110110001101000110010
1110000000
After Tampering Secret key generated is,
00000000000010010111000110000000000001000000101101
00000000000010111101100010010000000001111001111100
01000000000011100100010100110110110001101000110010
1011000000

Figura 5 - Comparação Watermark

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%

2.2.3.4. Vantagens

Guarda a posição das palavras do texto numa matriz. Tem uma identidade certificadora que guarda o *watermark* e é a única pessoa que pode verificar se o documento é original.

2.2.3.5. Desvantagens

Difícil de compreensão, a execução do algoritmo leva muito tempo de execução e processamento para textos grandes.

2.2.4. Line-Shift Coding

Os autores do artigo [4] expuseram outra forma de fazer watermarking que se denomina, *Line-Shift Coding*, que consiste em mover 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 movida para baixo 1/300 inches, que equivale a 0.00846666667 cm. Esta mudança não é perceptível ao olho humano, só através de um OCR, Optical Character Recognition, é que se conseguiria comparar ficheiros para verificar se existe ou não mudança nas linhas.

the Internet aggregates traffic flows from many end systems. Understanding effects of the packet train phenomena on router and IP switch behavior will be essential to optimizing end-to-end efficiency. A range of interesting

the Internet aggregates traffic flows from many end systems. Understanding effects of the packet train phenomena on router and IP switch behavior will be essential to optimizing end-to-end efficiency. A range of interesting

Figura 6 - Exemplo line-shift

2.2.4.1. Word-Shift Coding

Os autores do artigo [3] deram outra forma de fazer watermarking que se denomina *Word-Shift Coding*, que consiste 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 inch, que equivale a 0.0169333333 cm, enquanto na primeira linha não se altera, a terceira é uma junção das duas anteriores. Como referido no ponto anterior esta mudança não é perceptível a olho humano, e para retificar a watermark recorrer a um OCR.

the Internet aggregates different sessions from many end systems. Understanding
the Internet aggregates different sessions from many end systems. Understanding
the Internet aggregates different sessions from many end systems. Understanding

Figura 7 - Exemplo word-shift

2.2.4.2. Character Coding

Os autores do artigo [3] deram outra forma similar, referidas anteriormente, mas desta vez consistia em mover o carácter para cima ou baixo, enquanto 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. Como foi referido anteriormente noutros capítulos é necessário um OCR, para retificar as mudanças feitas no texto, isto é dispendioso em termos de tempo, e difícil de percepção.

the Internet aggregates
Internet

Figura 8 - Exemplo Character Coding

2.2.4.3. Vantagens dos métodos

Tem watermark invisível, ou seja, não é perceptível para os olhos de um humano.

2.2.4.4. 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 mal classificação de um documento, por exemplo ser falso, quando se tem a certeza de que o documento é original e não foi modificado.

2.2.5. ECL zero-based watermarking

Os autores do artigo [5], propuseram um tipo de *watermarking*, que se denomina *ECL zero-based watermarking* que consiste num algoritmo que mantém o conteúdo original do texto do documento e constrói a *watermark* pela seleção de caracteres no documento. A *watermark* é guardada num sítio de confiança que se denomina *Certifying Authority*, para quando houver a necessidade de comparar o documento com a *watermark* para verificação da sua autenticidade, perceber que se o documento é original ou falsificado.

Os autores forneceram uma imagem (figura 9) em que se demonstra como funciona o algoritmo.

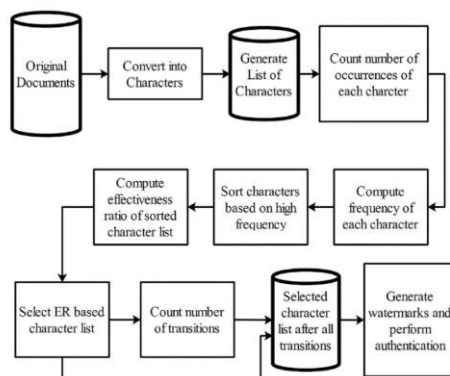


Figura 9 - Funcionamento do algoritmo

2.2.5.1. Vantagens

Tem uma identidade certificadora que garante que o documento original esteja seguro com uma pessoa confiável na empresa.

2.2.5.2. Desvantagens

Falta de testes em cópias de documentos. Pouco efetivo com documentos que tenham um *effectiveness ratio* (ER) menor que 0.1. Effectiveness Ratio determina quais os caracteres para a geração da watermark.

3. Qrcode

Atualmente vemos qrcode, nas faturas, em cartões de empresas e até mesmo na rua, maior parte das pessoas já viu, mas maior parte pode não saber o processo de leitura ou como surgiu o mesmo, para isso dediquei este capítulo ao qrcode.

O Qrcode (figura 10) surgiu em 1994 pela empresa Denso Wave, onde originalmente foi criado para categorizar peças de automóvel. Os Qrcode podem ter links para páginas web, texto, um endereço geográfico, uma imagem, um vídeo ou contacto telefónico.

Relativamente à estrutura do qrcode é constituído por 3 quadrados de deteção de posição (4.1. Figura 11) que permite a leitura em várias posições do scanner ou de um smartphone com câmara.

Por um padrão alinhamento (4.2. Figura 11) que corrige a distorção do qrcode em superfícies curvadas, o seu número varia consoante a informação contida.

Por padrões de temporização (4.3. Figura 11) que permite obter o tamanho da matriz de dados. Por informações da versão, que indica que versão do qrcode está a ser utilizada (1. Figura 11).

Por informações de formato, que contém informações sobre a tolerância de erros e o padrão da máscara de dados. Por códigos de dados e erros que podem ser do tipo L, M, Q, H, cujos estão apresentados na tabela 6.



Figura 10 – Exemplo de um Qrcode

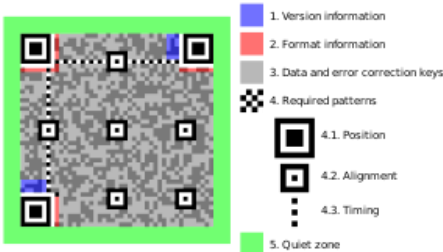


Figura 11 - Estrutura Qrcode

Tabela 4 - Classificação de erros do Qrcode

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

4. Código de barras 128

O código de barras 128 (figura 12 [6]) é um tipo de códigos de barras linear que permite ter 128 caracteres da tabela ASCII (figura 13).

É constituído por zonas quietas 1- figura 12, por símbolo de início 2- figura 12, por dados que são colocados no grcode 3- figura 12, por um símbolo de verificação que serve para verificar se o barcode foi bem gerado 4- figura 12, e por último um símbolo de stop, onde acaba o código de barras.

A largura do código de barras é dada pela quantidade de informação que este tem no ponto 3- figura 12.

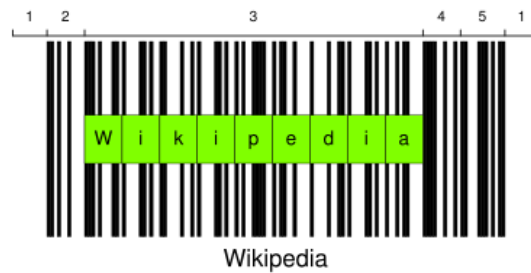


Figura 12 - Código de barra 128

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 13 - Tabela ASCII

5. Entropia documento com watermark

O presente projeto de tese apresenta estudos e desenvolvimento de documentos com watermark, já que estes documentos são confidenciais, ou seja, não se pretende que qualquer pessoa que não seja autorizada os veja. Contudo no mundo atual existem pessoas que pretendem descobrir os documentos mais confidenciais e os divulgar, ou até mesmo, alterar para outros efeitos. Um programa de segurança deve evitar ao máximo que o algoritmo seja descoberto, ou seja, deve garantir que a entropia do algoritmo seja menor possível.

Entropia é definida como uma forma de medir o grau de incerteza de um sistema, ou seja, quanto mais um sistema for incerto menos probabilidade tem se de descobrir como um sistema ou algoritmo funciona. Para é necessário que o algoritmo desenvolvido seja mais discreto e mais aleatório possível.

6. Métodos implementados

Antes de iniciar o desenvolvimento do programa e ter uma proposta final, foram colocadas em estudo várias técnicas.

Em primeiro o objetivo era usar o algoritmo de espaçamento de letras referido no estudo da arte no subcapítulo

2.2.4.1, contudo descartou-se porque era difícil de implementar e também a taxa de erros do espaçamento seria maior se o documento tivesse sido impresso torto ou com falta de letras, devido por exemplo à falta de tintores, ou a erros de impressão.

Em segundo pensou-se em alterar os caracteres do texto no documento, como foi colocado no subcapítulo 2.2.2., mas quando a empresa referiu que os documentos são imutáveis, ou seja, não permite a alteração de texto, descartou-se.

Em terceiro, pensou-se em ter um código de barras 128, que confirmaria o utilizador que assinou o documento, e um qrcode colocado no texto, que conteria as informações do documento, mas foi descartado devido ao espaço ocupado pelo código de barras no footer do documento, e porque a empresa queria que informação do documento estivesse toda agregada num só código de barras.

De seguida, implementou-se um qrcode, contendo como informação um índice de uma tabela da base de dados que contém as características do ficheiro, inserido numa posição aleatória do documento. Cujo irá ser abordado com mais detalhe no subcapítulo 6.1.

Entretanto surgiu a necessidade de criar funcionalidades de aprovação/rejeição de documentos gerados pelo programa, depois de se pensar na questão de o utilizador não querer que o qrcode esteja naquela posição documento, para isso adaptou-se o código e criou-se um programa novo que contém interface gráfica ao utilizador, levando a que o tempo de processamento demorasse mais devido à criação das instâncias visuais.

Como o processamento de imagem é lento surgiu a ideia de criar segmentos de retas para ligar posições de posicionamento do qrcode, abordado no subcapítulo 6.2.

6.1. Primeira implementação

Como referido anteriormente, a posição do qrcode vai ser aleatória no documento, podendo calhar em zonas que contém imagens, texto e zonas livres, que não tem texto nem imagens. Por exemplo na figura 14, que contém um documento de teste, o qrcode está inserido numa zona livre. Se o qrcode calhar numa imagem, é colocado noutra posição já que é impossível de ler qrcode com interferência de imagem. Se porventura calhar no texto, é possível ler, só que é necessário processar o documento, que obrigatoriamente tem de ser convertido para imagem, para permitir a alteração da cor dos pixels.

Um leitor de qrcode só consegue ler se não existir interferência dentro do mesmo, para perceber melhor criou-se a imagem 15, que demonstra as zonas críticas do qrcode num documento exemplar. Ou seja, quando aparece preto no cinza deve-se alterar para cinza, e quando aparece preto no branco, alterar para branco.

Para isso criou-se um algoritmo de processamento de imagem, que altera a cor do Qrcode no documento.

Este algoritmo consiste em processar o documento até encontrar a cor do qrcode, que é cinza-claro, se existir outro cinzento no documento a cor vai ser alterada também, que irá influenciar na leitura do qrcode se este sobrepor ou estar perto da cor cinza. Quando encontrar o qrcode, o algoritmo vai procurar os pixels de cor preta, que simboliza o texto, de seguida lê a cor dos vizinhos – cima, baixo, esquerda, direita – mudando a cor do pixel consoante a tabela 5 e a cor pretendida como referido em cima, sendo o certo para mudar e errado para não mudar.

Para demonstrar os resultados do algoritmo, retirou-se um screenshot de dois ficheiros auxiliares o ficheiro com o qrcode (imagem 16) e o ficheiro processado com o algoritmo (imagem 17) com o tamanho do Qrcode de 75x75 pixels que equivale a aproximadamente 2cm. Pela análise da imagem 17 retira-se que o algoritmo fez a substituição da cor do qrcode com alguns restos, e que se passar um leitor de qrcode consegue-se ler, dependendo da qualidade telemóvel, é de salientar que o conteúdo do qrcode é exemplificativa. Contudo testar num documento com muito texto vai falhar devido ao tamanho do pixel do qrcode ser inferior ao carácter do texto, concluindo que o tamanho adequado é de 100x100.

A tabela 6 analisa a performance do algoritmo, avaliando o tempo de execução desde a criação do qrcode, inserção no documento, processamento de imagem à verificação da leitura. Na figura 19 tem-se o output do programa desenvolvido em c# em consola para efeitos de testes do documento, a parte vermelha e os outros documentos não se pode demonstrar devido à confidencialidade. Os prints representam o nome do ficheiro a processar, o resultado do id do ficheiro associado na base de dados e o tempo de execução do tipo “hh:mm:ss”.



Figura 14 - Zona livre

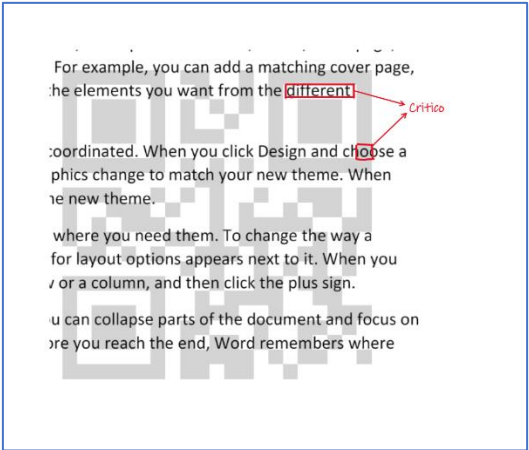


Figura 15 - Zonas críticas

Tabela 5 - Mudança de cor

	CIMA	BAIXO	ESQUERDA	DIREITA
CIMA	×	✓	✓	✓
BAIXO	×	×	✓	✓
ESQUERDA	✓	✓	×	×
DIREITA	✓	✓	✓	×

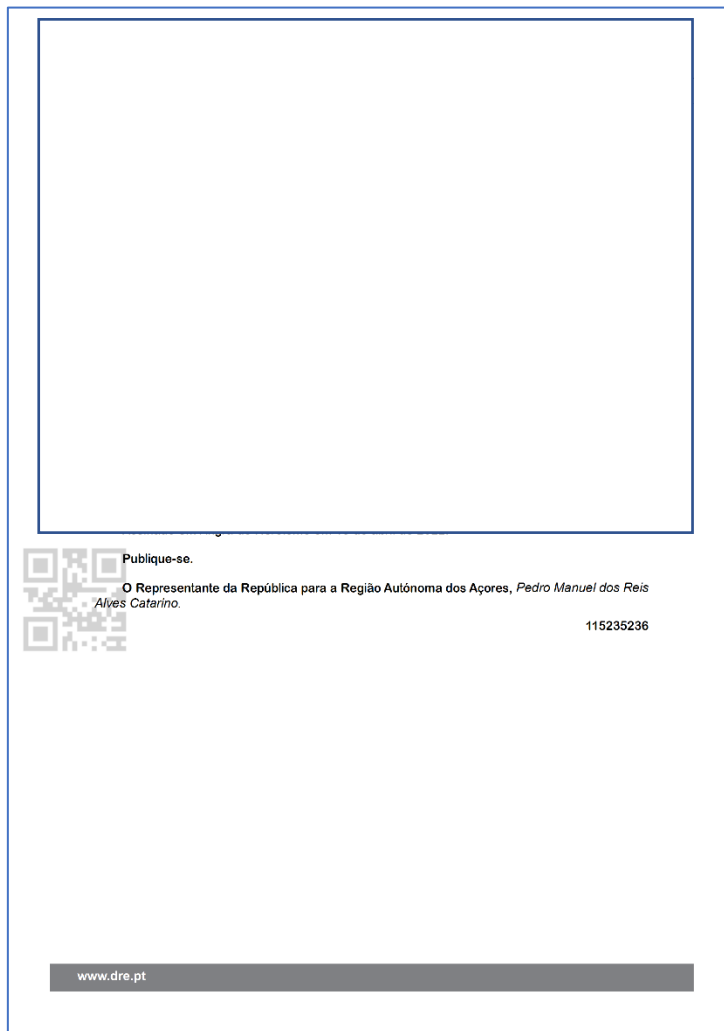


Figura 16 - Ficheiro exemplar

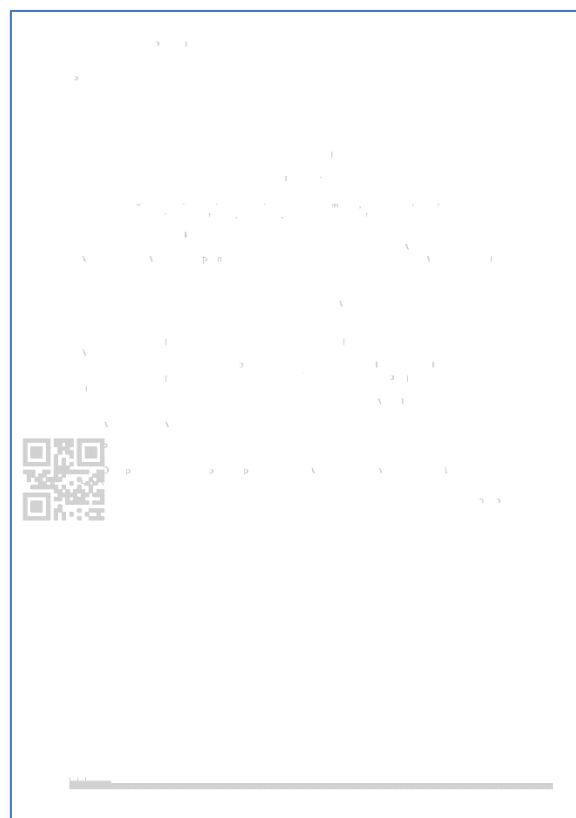


Figura 17 - Ficheiro processado

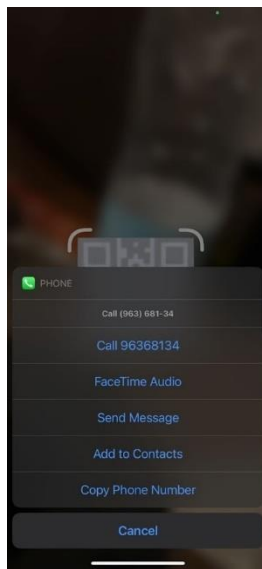


Figura 18 - Resultado Qrcode

Tabela 6 - Performance

Tamanho Qrcode(pixels)	Ficheiro	Tempo (horas:minutos:segundos)
200x200	NACIONAL_1_2022_01000	00:00:41
100x100	NACIONAL_1_2022_01000	00:00:39
100x100	NACIONAL_19208_2022_29000	00:01:11
100x100	NACIONAL_23_2022_05000	00:01:10
100x100	NATO_190_2021_13000	00:00:34
75x75	NACIONAL_19208_2022_29000	6 min – falhou com 10 tentativas

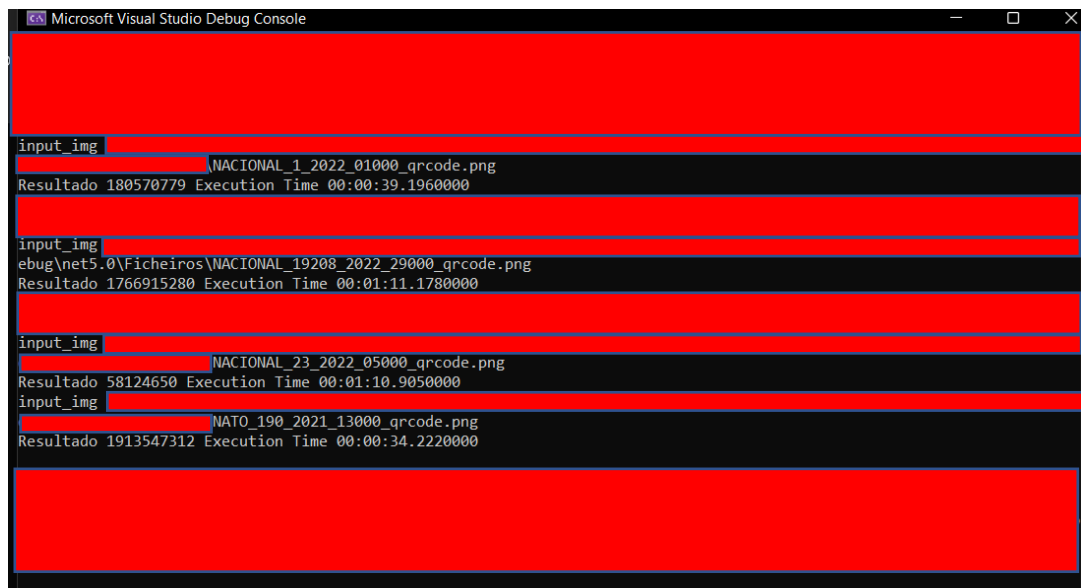


Figura 19 - Resultado consola

6.2. Análise de quantidade de qrcode para a verificação de integridade

O objetivo deste capítulo é estudar o número ideal de qrcode para inserir no documento para guardar as letras com base no segmento de reta desenhados a partir dos quadrados de posição referidos no capítulo 3 figura 11. A figura 20 representa o que se pretende desenvolver para a verificação de integridade.

Na tabela 7 está representado o número de estudo de Qrcode a usar no ficheiro, tem três colunas com as seguintes características “Nº de Qrcode”, “Nº Retas por quadrado de posição” e “Nº ligações”, as linhas representam os respetivos valores.

Em primeiro lugar pensou-se na possibilidade de haver só um segmento de reta a sair por quadrado de posição de um qrcode e retificar a interseção com a reta, mas como essa pode passar por várias letras, causa problemas de armazenamento e visualização difícil por utilizadores na verificação de integridade. Para resolver esse problema e preencher todo o documento resolveu-se colocar 9 Qrcode numerados num novo documento PDF com base no documento PDF de entrada, e por cada quadrado de posição tem-se 3 segmentos de reta (figura 21), como decerto algumas retas se cruzam (círculos amarelos na figura 21) resolveu-se calcular o ponto de interseção e guardar o caracter que esteja inserido nesse ponto.

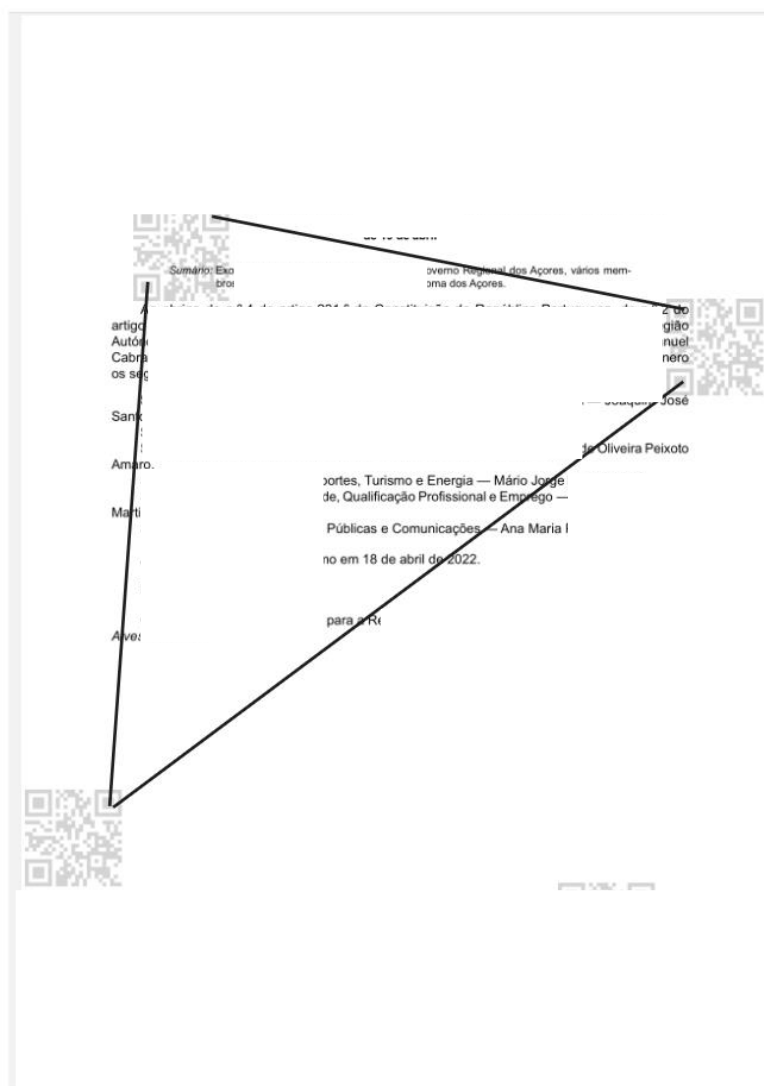


Figura 20 - Verificação de integridade pretendida

Tabela 7 – Estudo do número de qrcode a usar no ficheiro pdf

Nº de Qrcode	Nº Retas por quadrado de posição	Nº ligações (Nº Retas que saem por quadrado ligadas ao quadrado de posicionamento)
2	1	9
2	3	9
4	1	18
6	1	27
8	1	36
9	3	81

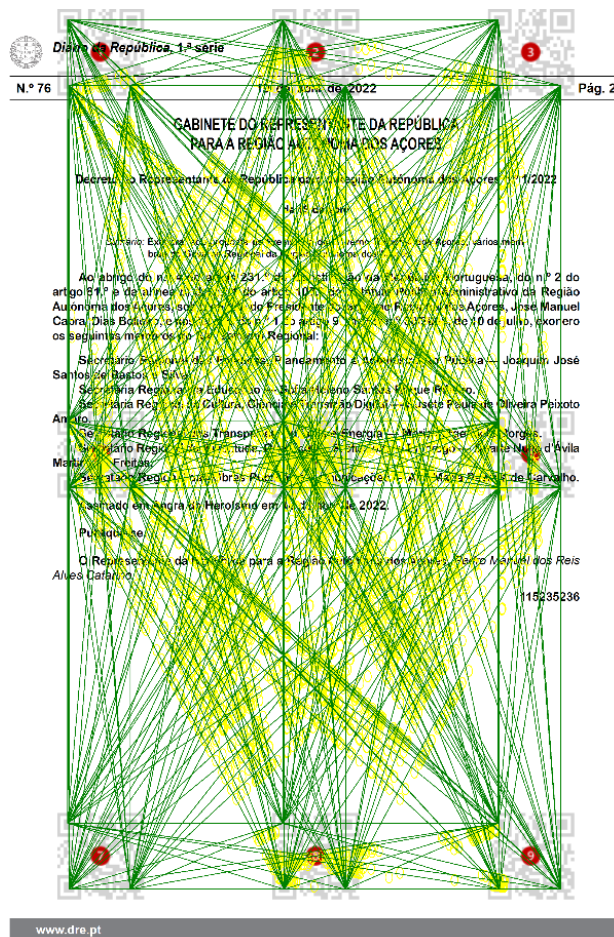


Figura 21 - Segmentos de reta no ficheiro PDF

6.3. Verificação de integridade com 9 qrcode com processamento de imagem

Antes do surgimento da ideia de usar os segmentos de reta no qrcode surgiu a proposta de realizar processamento de imagem no qrcode. O programa desenvolvido convertia o documento PDF numa imagem PNG (imagem 22), para alterar a cor dos pixéis, a conversão é feita com a utilização do package Freeware.Pdf2Png. De seguida calculava-se as posições dos qrcode que são as seguintes “2,25|8,25|12,25|2,32|8,32|12,32|2,41|8,41|12,41” cm, cuja ordem é do qrcode 1 para o 9.

Com base na posição de cada qrcode processava-se o documento na zona das posições (imagem 23), guardando a imagem do qrcode na posição para retificar se consegue ler a informação (conjunto de imagens com borda preta figura 24, cuja ordem da esquerda para direita e cima baixo é de qrcode 1 para 9). Através dos maus resultados e difícil perceção como também o tempo de execução da mesma descartou-se esta proposta.



**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 Boileiro, 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.

Secretaria 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



Figura 22 - Qrcode com processamento de imagem

7. Proposta de desenvolvimento

O desenvolvimento da solução para o problema consiste em duas partes, gerar a marca de água do ficheiro com base nos metadados, e verificar se o ficheiro recebido é original e não foi modificado, designando-se verificação de integridade.

Relativamente à verificação do ficheiro recebido como referido no capítulo 3.1 o tempo de execução do algoritmo de processamento de imagem aumenta devido à componente gráfica do *Windows Form* (C#) passa de 1 minuto para 3 minutos. Descartando completamente o processamento de imagem

A solução pensada envolve colocar:

- 9 qrcode fixos pelo documento, contendo um identificador de ficheiro por questão de backup e a versão do qrcode que é usada, para permitir melhorias ou inserção no código. Nestes irão ser traçados 3 segmentos de reta por cada quadrado de posição do qrcode, que neste caso contém 3, que dá um total de 9 segmentos de reta por qrcode, totalizando 81 segmentos de reta, sendo que estes não vão ser mostrados no documento nem ao utilizador.
- Código de barras para ter um conjunto de informações do documento e as posições dos qrcode inseridos no documento, isto para permitir o operador fazer uma análise rápida ao documento para retificar se o documento que recebeu é o mesmo que consta no código de barras, senão corresponder aí vai ser usado a verificação de integridade que consiste em saber que zonas do documento foram alteradas com ajuda da interseção dos segmentos de reta traçados nos quadrados de posição do qrcode.

Como uma medida de segurança o utilizador só tem acesso ao valor lido pelo leitor de barras, se estiver na base de dados do sistema, onde está a picar/ler o mesmo, isto descarta os mais curiosos, não tendo acesso ao conjunto de informações do documento, tal como a pessoa que o assinou.

8. Arquitetura da solução

Ao longo do desenvolvimento do programa de consola referido no capítulo 5.1 e necessidades da empresa, criou-se a seguinte arquitetura presente na figura 25.

O programa desenvolvido tem três camadas:

1. Servidor destinada à base de dados que irá ter dados armazenados acerca dos ficheiros, códigos de barras, segmentos de reta traçados entre qrcode, posição dos caracteres no ficheiro de input e a geração da marca de água.
2. Programa onde acontece o desenvolvimento da criação do código de barras, do qrcode, do processamento do ficheiro para obter as posições dos caracteres no documento, da retificação do documento, e verificação de integridade, esta também é responsável pelas conexões entre camadas, desencadeadas pelo utilizador.
3. Utilizador que desencadeia as ações que o utilizador pode fazer como por exemplo processar o documento ou retificar.

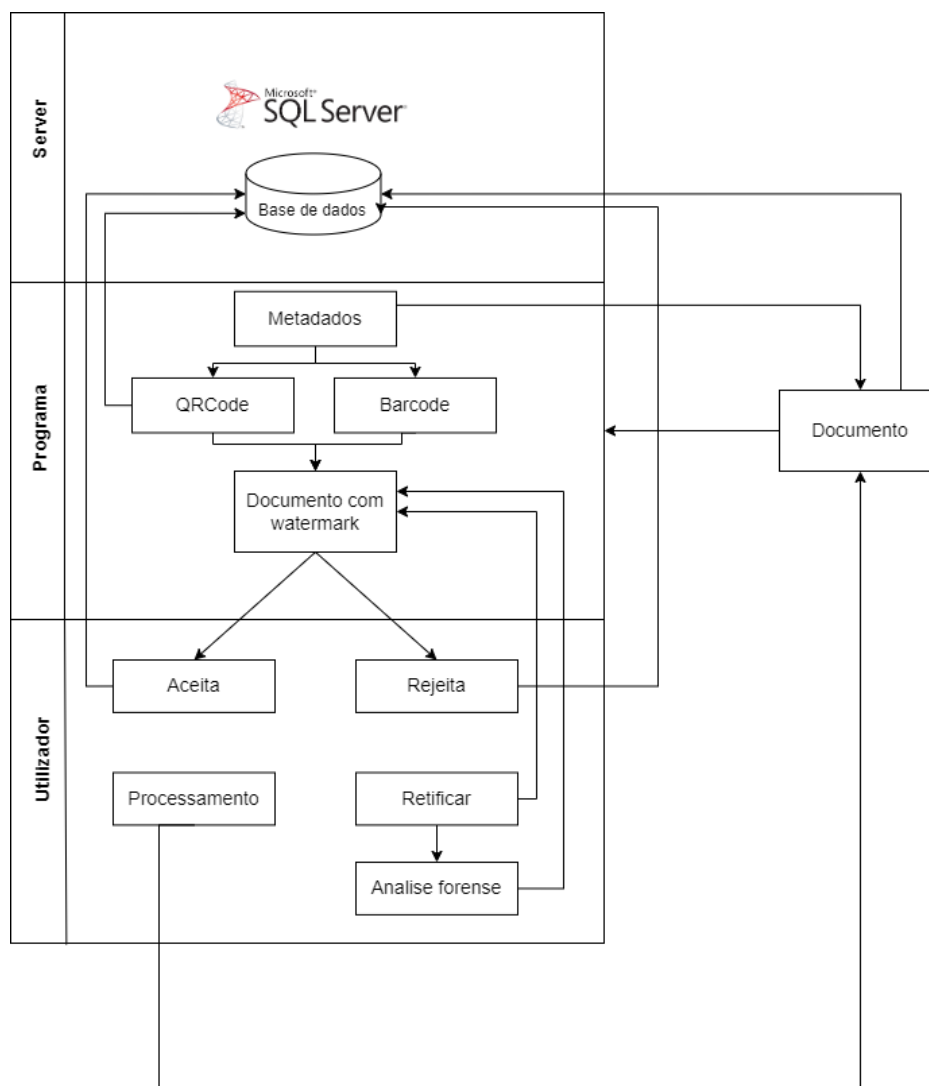


Figura 25 - Arquitetura solução proposta

8.1. Estrutura do documento

O documento que vai dar entrada no sistema, é um documento PDF, que tem metadados - informações sobre o documento, como por exemplo, a classificação de segurança – inseridos pela base de dados da empresa que pela uma função desenvolvida com os parâmetros necessários vai adaptar a estrutura para o do programa. Para realizar o processamento dos documentos estes têm de preencher dois requisitos o primeiro existirem na pasta predefinida, e o segundo existir metadados no programa.

O programa trata dois estados de documentos.

- Documentos com watermarking para retificar a origem deles e se for necessário proceder a uma verificação de integridade se o código de barras tiver informação diferente daquela que o documento apresenta, guardando os metadados do documento numa tabela da base de dados.
- Documentos sem watermarking que são utilizados para gerar a watermark, para que se consiga validar a autenticidade do documento. As watermark são colocadas na primeira página do documento, para a realização da tese.

8.2. Base de dados

O intuito de usar base de dados no sistema é garantir segurança sobre as informações do código de barras e qrcode, como se referiu anteriormente, o código de barras vai ter um id aleatório representante das características do documento cujo vai ser 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 chamado de 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 cria grupos em se 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, devido a ter muita experiência na licenciatura neste programa.

A base de dados SQL foi criada localmente, tendo um utilizador e base de dados dedicado apenas à consulta de informações pelo programa. As informações que se guardam são as características do documento, o código de barras, segmentos de reta traçados entre Qrcode, posições dos caracteres no ficheiro de entrada, e a criação da marca de água.

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

O diagrama é constituído por:

- “document” - guarda características do documento (metadados);
- “barcode” - guarda informações relativas ao documento através do id para se aceder na leitura do código de barras;
- “watermark_qrcode” - 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 qrcode, 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.

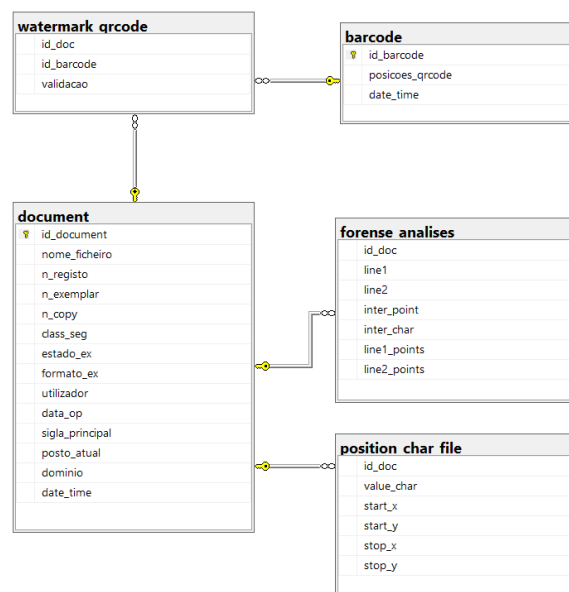


Figura 26 - Diagrama da Base de dados

8.3. Criação do código de barras e Qrcode

Para a criação do código de barras 128 e do qrcode foi utilizado um package para C# denominado Iron Barcode, que no anexo irá ser abordado em detalhe.

Como referido anteriormente o código de barras irá ter um id referenciador para o documento e para as posições dos Qrcode no documento. Em relação ao qrcode, irá ter a informação da versão do qrcode, caso se quiser mudar a estrutura futuramente e um identificador do documento para evitar perdas de informação caso o código de barras não seja visível.

As posições quer dos 9 Qrcode quer do código de barras são fixas, permitindo uma diminuição significativa do tempo de execução caso os qrcode fossem inseridos aleatoriamente.

8.4. Operações a desempenhar pelo utilizador

Um utilizador exerce 4 tipos de funções.

- Escolher entre
 - Processamento: que processa o documento sem a watermark para originar a watermark;
 - Retificar: que retifica o documento de input com a watermark para verificar se é o documento que se apresenta.
- Aceitar ou rejeitar o documento com watermark;
- Comparar a informação do documento com a watermark no submenu retificar, 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.

8.5. Processamento do documento

Com base no documento escolhido para processamento, gera-se um id aleatório identificador para a base de dados onde se vai inserir os metadados do documento inseridos previamente no programa, este programa só vai ter acesso um utilizador que seja fidedigno, ou seja, apenas pessoas que a empresa confie pode usar o código chave, fornecendo o programa, para inserir os metadados. Como a certo ponto pode existir colisões de criação do id, o programa verifica se já criou aquele id, se tiver gerado o mesmo número gera outro. De seguida cria-se os qrcode e o código de barras com a informação, referida no ponto 8.3. Seguidamente, coloca-se os qrcode numa posição fixa do documento e o código de barras no rodapé do mesmo.

Previamente lê-se as posições dos caracteres do ficheiro para inserir na base de dados e traça-se retas invisíveis para calcular o ponto de interseção, caso o ponto calhe em alguma letra, a informação é guardada na base de dados para a verificação de integridade.

Por fim o utilizador pré-visualiza o documento e pode aceitar ou rejeitar o documento com watermarking gerado, se aceitar o documento é guardado em formato PDF com o nome do documento inserido com adição “_qrcode_dd_mm_yy_hh_m_ss”, sendo d:dia, mm:mês, yy:ano, hh:hora, m: min, ss:segundos, se rejeitar todo o processo é descartado, guardando na base de dados e de seguida o documento é processado novamente.

8.6. Retificar documento

Para retificar um documento é necessário que o documento escolhido para retificação contenha a marca de água e cuja informação esteja na base de dados. Caso esteja, o programa, lê o código de barras 128 e mostra as informações contidas para o utilizador numa janela onde é possível ter a visualização do documento com a watermark escolhido e algumas informações sobre as características do documento (metadados). 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 que foram alteradas no documento.

8.7. Verificação de integridade

A verificação de integridade serve para retificar onde o documento foi alterado, caso a retificação de um documento falhar, para isso resolveu-se criar 3 segmentos de retas que saem dos quadrados de posição do qrcode, como a versão utilizada do Qrcode é a 2 tem 3 quadrados de posição tendo $3 \times 3 = 9$ segmentos de reta, totalizando $9 \times 9 = 81$ retas.

Como existem 81 retas e cada qrcode tem 3 quadrados e seleciono sempre duas retas a expressão fica $81 \times 3 \times 2$, que dá 486 retas escolhidas para calcular o ponto de interseção, mas como existem algumas duplicadas, isto é que já calculadas por exemplo Qrcode1_1:Qrcode8_b é mesma coisa que Qrcode8_b:Qrcode1_1, dando um total de 324 retas possíveis, como se pode retificar no resultado sublinhado a preto no algoritmo implementado na figura 27.

Aquando das retas traçadas vai se calcular o ponto de interseção da mesma, com a expressão da figura 28, de seguida 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. Figura 29, sendo o circulo amarelo o ponto de interseção e a azul o caracter que está na base de dados. Como se pode retificar existem letras repetidas e a apresentação é um bocado não apropriada, por isso mudou-se para linhas e só ter uma letra caso existisse, para não repetir.

A solução pensada para resolver o problema foi desenvolver uma query em sql que removesse as letras que tivessem a mesma origem ou mesmo destino. Na figura 30 pode-se retificar pelo output da query na base de dados que existem 1544 possíveis letras para uso de retificação de integridade. Através da query feita em sql o valor novo de letras diminui drasticamente para 557 (figura 31) contendo ainda letras repetidas. É de salientar que esta remoção diminui o tempo de processamento para inserção da base de dados e removeu possíveis interseções únicas.

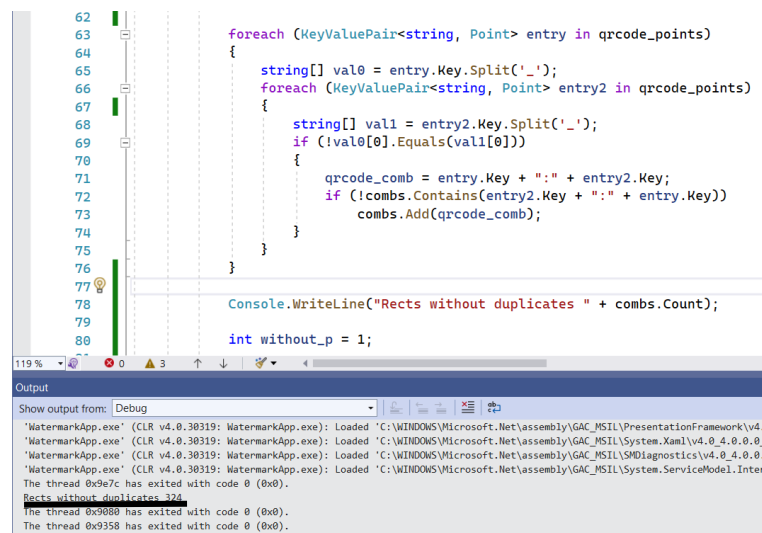


Figura 27 - Algoritmo que conta número de retas que se interseam

```

private Point Intersection(Point A, Point B, Point C, Point D)
{
    int x1 = A.X;
    int x2 = B.X;
    int x3 = C.X;
    int x4 = D.X;

    int y1 = A.Y;
    int y2 = B.Y;
    int y3 = C.Y;
    int y4 = D.Y;
    double t = ((double)((x1 - x3) * (y3 - y4) - (y1 - y3) * (x3 - x4)) / ((x1 - x2) * (y3 - y4) - (y1 - y2) * (x3 - x4)));
    double u = ((double)((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))
    {
        double x = x3 + u * (x4 - x3);
        double y = y3 + u * (y4 - y3);

        inter = new Point((int)x, (int)y);
    }

    return inter;
}

```

Figura 28 - Algoritmo interseção retas

GABINETE DO REPI
PARA A REGIÃO

presentante da República

Figura 29 - Exemplo verificação integridade

```

select Count(*) as NumeroTotalLetras, (select date_time from document where id_document = 342717422 ) as DataMarcaDeAguaGerada
from forense_analises where id_doc = 342717422 group by id_doc

```

Results

NumeroTotalLetras	DataMarcaDeAguaGerada
1544	23_1_2023_16_10_8

Figura 30 - Número Total Letras com letras repetidas

```

select Count(*) as NumeroTotalLetras, (select date_time from document where id_document = 342717422 ) as DataMarcaDeAguaGerada
from forense_analises where id_doc = 342717422 group by id_doc

```

Results

NumeroTotalLetras	DataMarcaDeAguaGerada
957	23_1_2023_16_10_8

Figura 31 - Número total de letras com algumas repetidas

GABINETE DO REPR PARA A REGIÃO A

representante da República

9. Testes

10. Conclusões e futuro trabalho

11.Referências

- [1] 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.
- [2] 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.
- [3] 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.
- [4] 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.
- [5] T. Saba, M. Bashardoost, H. Kolivand, M. S. M. Rahim, A. Rehman, and M. A. Khan, “Enhancing fragility of zero-based text watermarking utilizing effective characters list,” *Multimed. Tools Appl.*, vol. 79, no. 1–2, pp. 341–354, Jan. 2020, doi: 10.1007/S11042-019-08084-0/TABLES/5.
- [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] “About Us | IronSoftware.com.” <https://ironsoftware.com/about-us/> (accessed Jul. 26, 2022).
- [12] “C# .NET Barcode Quickstart Guide, Code Examples | Iron Barcode.” <https://ironsoftware.com/csharp/barcode/examples/barcode-quickstart/> (accessed Jul. 26, 2022).
- [13] “C# Tesseract OCR In 1 Line of Code | Iron OCR.” <https://ironsoftware.com/csharp/ocr/examples/simple-csharp-ocr-tesseract/> (accessed Jul. 26, 2022).

Apêndice A

OCR

Iron Software

Para desenvolver o programa foi necessário usar tecnologias de criação de Qrcode e leitura de Qrcode. Para isso usou-se um software open-Source, que tem a opção de comprar a licença para efeito de segurança e apoio, denominado Iron Software

[11]. O Iron software é usado por mais de 6.9 milhões de utilizadores tendo vários produtos como o Iron PDF, Iron OCR, Iron Barcode, Iron XL, Iron Updates e Iron suite, listados na imagem abaixo. Sendo que neste projeto se usou o Iron OCR para leitura do qrcode e Iron barcode para gerar o qrcode.

Optou-se por usar o Iron Barcode [12] para gerar o qrcode devido a fornecer documentação e pelas propriedades de geração do qrcode, permitindo mudar o estilo do qrcode, como por exemplo a cor do qrcode, também permite colocar o qrcode no ficheiro PDF, mas eu optei por usar outro método, já que alguns documentos PDF são protegidos e neste pacote não tem opção de escrever neles sem saber a password.

Para ler/detetar o qrcode no ficheiro PDF optou-se por usar o Iron OCR [13], cujo procura num documento PDF o qrcode, e devolve o resultado.

