IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

Rui Pedro Pinto Borges

CEPAD – CLASSIFICAÇÃO E PROCESSAMENTO AUTOMATIZADO DE DOCUMENTO

Trabalho de Projeto submetido como requisito parcial para obtenção do grau de **Mestre em Engenharia de Software**

Júri

Presidente Professor Cláudio Miguel Garcia Loureiro dos Santos Sapateiro, ESTSetúbal/IPS

Arguente Professor João Miguel Jones Ventura, ESTSetúbal

Orientador Professor Nuno Miguel Vicente de Pina Gonçalves, ESTSetúbal/IPS

Novembro de 2022

Dedicatória de carácter pessoal.

…

(facultativo)

Agradecimentos

Quero agradecer em primeiro lugar aos meus pais que sempre me apoiaram e incentivaram a fazer mais e melhor, providenciando tudo para que pudesse sempre seguir o caminho que quis. Quero agradecer à minha namorada que sempre acreditou em mim e no meu trabalho, que me acompanhou ao longo de incontáveis horas de estudo e desenvolvimento. Quero ainda agradecer ao meu orientador, que sempre se mostrou disponível para me auxiliar e mentorar.

Quero agradecer ao Instituto Politécnico de Setúbal que me deu uma formação sólida e diferenciada, que me permite diariamente ter sucesso na minha vida profissional e à KCS IT que me permitiu aplicar os conhecimentos que adquiri ao longo do curso em projetos reais e motivadores, permitindo-me ter liberdade para inovar, explorar e desenvolver.

Resumo

A maior parte da faturação continua a ser feita com recurso a papel. A digitalização do processamento deste tipo de documentos promete apresentar vantagens financeiras e de qualidade. Pretendeu-se então estudar a possibilidade de desenvolver uma ferramenta que, recorrendo a *Machine Learning*, permita auxiliar na identificação e categorização dos campos presentes num documento. Como ponto de partida do desenvolvimento estudou-se o estado da arte atual de vários campos de estudo, como *Machine Learning*, *Optical Character Recognition* e tarefas de classificação, partindo-se em seguida para uma análise dos possíveis utilizadores, levantamento de requisitos e modelação do sistema a desenvolver. Recolhidos os dados que permitiram o treino de um modelo de *Machine Learning,* desenvolveu-se uma aplicação capaz de gerir documentos e processá-los, permitindo aos seus utilizadores validarem os dados inferidos, guardarem os resultados deste processamento e exportarem vários resultados simultaneamente. O modelo final apresenta uma percentagem de acerto de 69% quando tendo em conta resultados exata e parcialmente corretos, com uma Distância de Levenshtein média de 4, servindo assim como um auxílio ao processamento de faturas. Finalmente são apresentadas algumas propostas de trabalho futuro e áreas que poderão beneficiar de ferramentas que utilizem a mesma tecnologia da aplicação desenvolvida.

Palavras-chave: Aprendizagem de Máquina, Reconhecimento de Entidades Nomeadas, Faturação Eletrônica, Desenvolvimento Web

Abstract

Most invoicing today continues to use paper. The digitization of this type of document processing promises to present financial and quality advantages. It was then intended to study the possibility of developing a tool that, using Machine Learning, helped identify and categorize the fields present in an invoice. As a starting point for the development, the state of the art of various fields of study was studied, such as Machine Learning, Optical Character Recognition, and classification tasks, followed by an analysis of possible users, requirements gathering, and modeling of the system to be developed. Having collected the data that allowed the training of a Machine Learning model, an application was developed capable of managing documents and processing them, allowing its users to validate the inferred data, save the results of this processing and export several results simultaneously. The final model has an accuracy rate of 69% when considering exact and partially correct results, with an average Levenshtein Distance of 4, thus aiding invoice processing. Finally, some proposals for future work and areas that could benefit from tools that use the same technology as the developed application are presented.

Keywords: Machine Learning, Named Entity Recognition, E-Invoicing, Web Development

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Acrónimos

AS	Aprendizagem supervisionada
BN	Bayesian Network
DL	Distância de Levenshtein
IA	Inteligência artificial
IAAS	Infrastructure as a Service
IPS	Instituto Politécnico de Setúbal
JWT	JSON Web Token
ML	Machine Learning
NER	Named Entity Recognition (Reconhecimento de entidades nomeadas)
NIF	Número de Identificação Fiscal
OCR	Optical Character Recognition (Reconhecimento ótico de caracteres)
PAAS	Platform as a Service
RBAC	Role-Based Access Control
SAAS	Software as a Service
SDK	Software Development Kit

1. Introdução

Segundo previsões feitas no ano de 2019, seria de esperar a transação anual de 550 bilhões de faturas. Destas, somente 55 bilhões seriam enviadas não tendo como suporte papel (Koch, 2019). Apesar de se prever que esta percentagem aumente nos próximos anos, a ainda grande utilização do papel como meio de emissão de faturas apresenta um custo imenso para as empresas e para o ambiente. A utilização de faturas em papel leva a que os dados das mesmas tenham de ser introduzidos manualmente, num processo moroso e dispendioso. Segundo um caso de estudo, o custo de emissão de faturas em papel ao invés de em formato digital é até 59% superior (Koch, 2017), pelo que governos e empresas procuram formas de digitalizar as suas faturas. Cada vez mais governos estão a criar leis e diretivas que obriguem as empresas a efetuarem a transição para o digital. Em 2020 entrou em vigor uma diretiva da União Europeia que define um *standard* para faturação eletrónica (European Committee for Standardization, 2020) e declara que os seus estados-membros terão de alterar as suas normas internas (The European Parliament And The Council Of The European Union, 2014), querendo-se uma maior uniformização da faturação na Europa, uma redução de custos e uma maior sustentabilidade.

A escolha deste tema prende-se tanto com fatores pessoais como profissionais. A nível profissional, a automação tem vindo a ganhar cada vez mais destaque e muitas empresas procuram soluções que lhes permitam deixar o papel para trás e poupar dinheiro no processamento de documentação que posteriormente será arquivada. A nível pessoal, existe uma preocupação clara com a sustentabilidade e com o processo de digitalização em geral.

Numa primeira fase de desenvolvimento, é necessário conhecer qual a realidade atual dos trabalhadores que fazem o processamento de faturação, entender quais os seus maiores desafios no dia-a-dia e identificar oportunidades de melhoria nos processos atuais. De seguida será necessário entender de que forma estão estruturados os dados e definir de que maneira podem estes sofrer alterações, procedendo-se depois à análise, modelação e desenvolvimento de uma prova de conceito que permita, numa última fase, validar a solução a desenvolvida.

Este documento apresenta um enquadramento teórico às áreas de interesse para o desenvolvimento da solução, apresenta de seguida a metodologia que conduziu a investigação, os resultados obtidos bem como uma consideração sobre os mesmos. Termina com a conclusão do estudo, as limitações e melhorias do mesmo e propostas de trabalho futuro.

2. Estado da arte / Enquadramento teórico

Neste capítulo apresentam-se os conceitos relativos a faturação, ciência dos dados, machine learning e algumas das suas etapas, ocr (reconhecimento ótico de caracteres), melhoramento de imagem, aprendizagem supervisionada e não supervisionada e classificação. Será apresentada a evolução da faturação digital bem como algumas técnicas atuais que representam a automatização do processamento automático de documentos, sendo apresentadas as suas principais características, vantagens e desvantagens. O processamento de um documento obedece a três etapas gerais: 1) Captura do documento e reconhecimento de texto; 2) Identificação e classificação do documento; 3) Reconhecimento dos campos de negócio. É por base nestes passos que os conceitos presentes neste capítulo foram escolhidos.

2.1. Faturação tradicional e digital

A utilização de faturas, recibos e outros tipos de comprovativos de transações comerciais é algo quotidiano da maioria das pessoas. A comunicação destas transações a entidades governamentais é também uma prática que afeta todas as pessoas e empresas. Em 2009, cerca de 5% das faturas processadas mundialmente eram trocadas digitalmente (denominadas *e-invoices*) (Caluwaerts, 2010). As previsões para 2019, apontavam para uma duplicação desta percentagem, sendo expectável que mundialmente se ultrapasse 550 bilhões de faturas (Koch, 2019) Sendo que ainda 90% das faturas são em forma de papel, existe um custo muito grande associado com o processamento de todos estes documentos, como exibido na Tabela 1, uma análise de um caso de estudo que permite ter referência de quanto custa enviar e receber faturas, a nível de tempo e dinheiro, sendo os custos

monetários de processamento manual, em média, 3,4 vezes superiores aos de processamento automáticos.

Tabela 1 - Comparação de custo de envio e receção de faturas. Adaptado de (Caluwaerts, 2010; Penttinen, 2008)

Nível de automação	**Custo de envio de uma fatura**		**Custo de receção de uma fatura**	
	Custo (€)	Custo (min)	Custo (€)	Custo (min)
Processamento manual	18,55	10,5	28,8	14
Processamento semiautomático	11,1	6	18	10
Processamento automática	10,8	6	3,3	1

Para além do problema ambiental que existe inerente à impressão e envio de faturas, existe ainda um problema de precisão de inserção de dados. No processamento manual 20% a 30% de todas as faturas são tratadas como exceções, tendo de os dados ser posteriormente validados manualmente, resultando num custo elevado (Koch, 2019). Este tipo de exceções apresenta também uma dificuldade no processamento automático, algo que irá ser estudados nos próximos capítulos.

O processamento automático traz claras vantagens a nível de custo. O recurso a ML, técnica abordada de seguida, permitirá lidar com exceções enquanto é possível automatizar o processo.

Existem atualmente algumas soluções comerciais que permitem uma digitalização do processo de processamento de faturas. Estes *softwares* melhoram consideravelmente a experiência do operador que necessita de processar os documentos mas, contudo, apresentam também algumas limitações. O Abby FineReader, software especializado em soluções para documentos PDF, apesar de efetuar o processamento do texto, apresenta dificuldade no reconhecimento de

caracteres específicos da gramática portuguesa (como é o caso das cedilhas e acentuação), identificando-se ainda alguma dificuldade no ajuste automático do brilho da imagem. Outro dos *softwares* estudados foi o Ephesoft, uma solução já mais avançada, com reconhecimento de campos e técnicas de OCR avançadas, contando ainda com uma componente forte de integração com serviços externos. Esta ferramenta mostrou algumas limitações no que diz respeito à adequação à realidade portuguesa, como por exemplo, na classificação do Número de Identificação Fiscal. Esta adaptação ao contexto português apresentou-se como a maior limitação, já que a grande parte das ferramentas analisadas é elaborada com outros contextos em mente, nomeadamente os EUA.

Foi analisada ainda o programa DocDigitizer, que se apresenta como um *software* completo de gestão documental, apresentou resultados interessantes sem introdução de dados manual. Contudo, o processo de processamento não só é demorado como é a equipa do próprio *software* que assume controlo durante o processamento do documento, validando cada documento quanto à sua correção de processamento. Assume-se este processo como um ponto negativo do sistema, apesar de aumentar o grau de confiabilidade no seu processamento, já que os dados dos documentos são visualizados por terceiros, podendo daí advir algumas questões ao nível da transferência de dados sigilosos.

Por fim, foi analisada a solução Rossum, que apresenta uma elevada taxa de precisão em faturas em inglês não apresentando, contudo, resultados tão positivos quando aplicado a faturas em português, apresentando algumas limitações no reconhecimento de caracteres específicos do idioma.

2.2. Data Science

A ciência dos dados (*Data Science*, DS) é um campo multidisciplinar que constrói e sintetiza um número de disciplinas e conhecimentos, como por exemplo estatísticas, informática, computação, comunicação, gestão e sociologia (Cao, 2017). Mais concretamente, segundo (Kuznetsov, 2019), DS é a interceção da ciência da computação, matemática (em concreto estatística) e especificidades do domínio do problema. Aparecem também descritas como subproduto das

interceções das disciplinas anteriormente listadas: o *machine learning,* análise de dados e software tradicional, algo similar ao exibido na Figura 5.

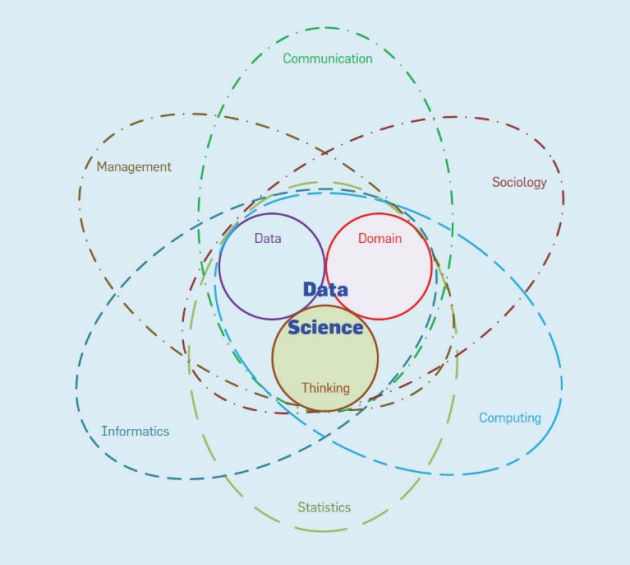

Figura 1 - Diagrama de abrangência da Data Science.
Adaptado de (Cao, 2017)

O software tradicional compreende todos os desenvolvimentos de software que visam responder a problemas de negócio típicos, como operações, contabilidade, recursos humanos, entre outros. Neste ramo, é essencial que se conheçam os requisitos de negócio, é necessário conhecer os *inputs* do problema, como os transformar e processar e quais os *outputs* esperados. Sempre que um destes componentes falha, não é possível recorrer somente ao software tradicional para a resolução total do problema.

Análise de dados é o processo sistemático de aplicar estatística e técnicas de lógica para descrever ilustrar sumariar e recapitular e avaliar dados (U.S. Department of Health & Human Services, 2022).

A análise de dados permite a tomada de decisão e a previsão com base na aplicação de modelos matemáticos a dados passados, sendo que neste processo é necessário que exista uma estruturação, limpeza e filtragem dos dados a utilizar.

Somente a análise de dados e o desenvolvimento tradicional de software não permitem o reconhecimento de documentos e a sua classificação. Contudo, são componentes usualmente integrantes de uma solução de *machine learning*, que

necessitará da componente de análise de dados para, por exemplo, tratamento dos dados de teste e de treino e de desenvolvimento tradicional de *software* para manipulação e construção de modelos.

Existe ainda outro conceito que, na sua raiz, está interconectado com termo de DS, a *Big Data*.

A *Big Data* é uma coleção de grandes quantidades de dados complexos, que são difíceis de processar com os métodos tradicionais ou recorrendo a estatística (Cielen et al., 2016), ao ponto que a DS permite o processamento de dados e consequente extração de informação. A evolução da DS permite o processamento das quantidades (e complexidades) de dados que a *Big Data* apresenta.

Um processo de DS geralmente conta com as seguintes fases, segundo (Cielen et al., 2016): definição de um objetivo de pesquisa; obtenção de dados; preparação dos dados; exploração dos dados; modelação dos dados (ou construção de modelo); apresentação e automação.

Foi sob este processo que foi desenvolvida a componente relativa aos dados no projeto realizado, sendo descrita posteriormente neste documento cada uma destas fases.

2.3. Machine Learning

A ML é uma subárea da Inteligência Artificial (IA) onde a máquina é responsável por determinada atividade de IA que sistematicamente aplica algoritmos para identificar as relações subjacentes entre os dados e informação (Awad & Khanna, 2015). Mais formalmente, a ML é um algoritmo computacional que aprende pela experiência *E* relacionada com alguma classe de tarefas *T* e uma medida de performance *P* se esta mesma performance em *T*, medida por *P*, melhora a experiência *E* (Mitchell, 1997a), ou seja, é assente na noção de que uma máquina, com base em dados históricos e métricas de desempenho, consegue identificar padrões e a maneira como os dados se relacionam, para que no futuro possa aplicar este conhecimento a conjuntos de dados novos.

Um exemplo de utilização de *ML*, entre muito outros casos de utilização onde o ML atualmente é aplicado, é o reconhecimento de fraude.

No caso de utilização anteriormente referido, são recolhidos dados históricos de, por exemplo, transações bancárias, algumas que constituem fraude e outras que são transações válidas. Este conjunto de dados deriva de análise elaborada por especialistas, que validam cada instância dos dados. Este conjunto de dados, usualmente chamado de *dataset*, contém variáveis de entrada como o remetente e destinatário da transação, a sua data e hora e valor. Juntamente com estas variáveis encontra-se uma variável de saída, um valor booleano que identifica a transação como válida ou fraude.

Este conjunto de dados é dividido, usualmente, em três conjuntos: treino, teste e validação. O subconjunto de treino é utilizado, com o nome indica, para treinar o modelo de ML, o subconjunto de teste é utilizado para comprovar e aferir a exatidão do modelo desenvolvido e o de validação é utilizado para comparação de diferentes modelos e de diferentes conjunções de hiperparâmetros.

Um hiperparâmetro, baseado na definição de (Claesen & De Moor, 2015) e normalmente referido na sua forma plural por λ, permite parametrizar um modelo de ML e deve ser definido apropriadamente para maximizar a utilidade da abordagem de aprendizagem, sendo utilizados para configurar vários aspetos do algoritmo de aprendizagem e podendo ter efeitos vastamente variáveis no modelo de ML resultante e na sua performance. Como exemplo de hiperparâmetro temos, numa rede neural, a quantidade de camas ocultas que a rede necessita de ter bem como quantos nós cada cama precisa de utilizar e o peso de cada nó, sendo estas variáveis que não estão diretamente relacionadas com os dados de entrada e as suas variáveis, mas são variáveis de configuração. Os hiperparâmetros são normalmente ajustados a cada execução de treino e o subconjunto de validação permite comparar a performance do modelo, sobre os mesmos dados, dadas diferentes parametrizações.

Após a subdivisão das transações históricas, desenvolve-se um algoritmo de aprendizagem e treina-se o modelo, iterativamente alterando os hiperparâmetros e ajustando a seleção de características. Do processo de treino obtém-se um modelo treinado que, com a utilização do subconjunto de teste, é avaliado quanto á sua

performance, utilizando métricas que serão descritas posteriormente neste documento. Todo este processo pode ser visualizado na Figura 2.

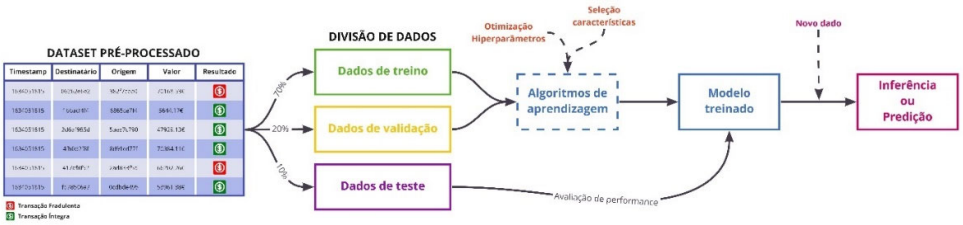

Figura 2 - Visualização do processo de Machine Learning.
Fonte: autoria própria

Aplicando a definição de (Mitchell, 1997b) ao exemplo da fraude, este algoritmo aprende pela experiência de analisar transações bancárias como legítimas ou fraudes (E), relacionada com alguma classe da tarefa de classificação dessas mesmas transações (T) e uma medida de performance denominada precisão (P) se esta mesma performance em T, medida por P, melhora a experiência E.

2.3.1. Seleção de características

Para uma classificação com um alto grau de confiança e com uma performance minimamente aceitável, é necessário que se proceda a uma seleção de características. Este processo consiste na eliminação de características irrelevantes, as que não tem qualquer informação útil e características redundantes, que não disponibilizam mais informação das características já selecionadas (Kumar & Minz, 2014).

2.3.2. Pré-processamento de dados

A ML pode ser divida por métodos de aprendizagem, dos quais se destacam a aprendizagem supervisionada, não-supervisionada, semi-supervisionada, aprendizagem por reforço e transferência de aprendizagem. Nos subcapítulos seguintes serão descritos alguns dos métodos que mais se destacam e quais os principais tipos de algoritmos que os compõem.

Um processo usual de *machine learning* é usualmente composto por quatro pontos: definir um problema, pré-processamento de dados, data mining e operações posteriores ao *data mining* (Zhang et al., 2003).

Destes passos, usualmente a limpeza e preparação dos dados exige uma grande parte do esforço, optando-se frequentemente por garantir que o processo é conduzido sobre dados com qualidade já conhecida *à priori.*

Pré-processamento de dados consiste num conjunto de passos para transformar dados não tratados (advindos da extração de dados) em dados limpos e organizados, para posterior análise (MIT Critical Data, 2016). Este processo pode muitas vezes até ser mais moroso que o de *data mining*, apresentando desafios iguais senão maiores (Yan et al., 2003). O pré-processamento é um processo muitas vezes *ad-hoc,* dependendo bastante do problema que se pretende resolver. Contudo, pode-se generalizar um processo habitual e tipos de passos que usualmente são utilizados, como por exemplo: seleção de instâncias e deteção de *outliers,* tratamento de valores em falta, discretização dos dados, normalização, seleção de características e construção de características (S. B. and K. D. and P. P. E. Kotsiantis, 2006).

A seleção de instâncias parte do princípio da redução de dados, permitindo filtrar os dados conforme certos parâmetros considerados aceitáveis. É possível eliminar valores acima ou abaixo de um determinado limite (reduzindo *outliers*), filtrar possíveis valores não aceitáveis para determinada característica, valores que estão, por exemplo escritos de forma incorreta ou que estão duplicados. Esta determinação de limites pode ainda ser definida com recurso a estatística, filtrando-se, por exemplo, valores acima ou abaixo da distribuição média ou com um desvio superior ao aceitável.

Não havendo uma ordenação definida dos passos do pré-processamento, já que é adaptado à realidade dos próprios dados, usualmente após a seleção executa-se o tratamento de valores em falta.

Tendo em consideração que usualmente os trabalhos de *data science* se baseiam em dados do mundo real, a probabilidade deste se encontrarem incompletos é bastante elevada. O cientista de dados que efetua o pré-processamento terá de se inquirir sobre a razão pela qual os dados estão incompletos, já que pode ser simplesmente um lapso ou determinada característica não se aplicar a determinada instância. Dependendo dos dados e da categoria de problema existem alguns métodos que podem ser aplicados para lidar com dados

incompletos (Lakshminarayan et al., 1999): ignorar as instâncias com características com valores em falta, substituir valores em falta pelo valor mais frequente dessa característica, substituição de valores pelo mais frequente, mas tendo em conta a classe da instância, substituição pelo valor médio, regressão ou classificação, identificar instâncias similares e posteriormente copiar valores e tratar os valores como especiais, a serem tratados à parte dos dados completos. No final deste passo os dados estão prontos a ser manipulados.

Segue-se a discretização, o processo de reduzir o número de valores distintos contínuos para uma variável discreta, dividindo o seu expectro de valores em um finito conjunto de intervalos disjuntos, relacionando esses intervalos com etiquetas significativas, reduzindo e simplificando os dados originais (Ghani, 2008). Este processo é comumente utilizado em captura de sinais físicos, já que é necessário transformar um valor analógico num valor discreto, com base na taxa de amostragem do dispositivo utilizado para captura.

Segundo (Liu, 2002) identificou-se um processo típico de discretização, que consiste em:

- Ordenação: onde os valores da característica a discretizar são ordenados ascendentemente ou descendentemente;
- Escolha de pontos de "corte": definição de pontos do intervalo que irão dividir ou agregar os valores contínuos em conjuntos;
- Avaliação de medida: avaliação de cada ponto com base nos "pontos de corte" anteriormente definidos;
- Dividir/agregar: Dividir ou agregar os valores consoante os grupos anteriormente criados com base nos pontos definidos anteriormente;
- Paragem: Existe uma condição de paragem que permite que o processo de discretização seja terminado. Esta condição de paragem pode ser, por exemplo, um número limite de classes.

O passo seguinte apresentado no pré-processamento de dados é a normalização destes. A normalização dos dados consiste no processo de modelar dados para um intervalo específico de valores, usualmente utilizado quando existe uma grande diferença na grandeza de valores das características (Ali & Faraj, 2014).

Como se pode ver na Figura 3, dado um intervalo inicial dos valores *input,* os valores são mapeados, com recurso a condições ou funções, de modo ao resultado obtido ser um valor dentro do intervalo esperado de *output*.

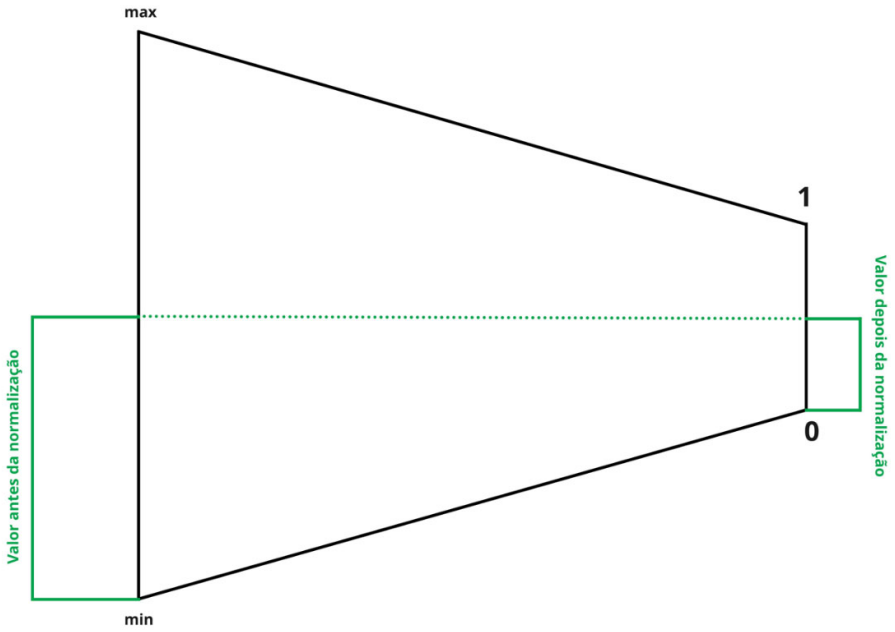

Figura 3 - Exemplificação do processo de normalização.
Fonte: Adaptado de (Ali & Faraj, 2014)

Posteriormente à normalização dos dados, e especialmente no caso dos dados servirem de *dataset* para uma aplicação de ML, é necessário que se proceda à seleção de características a considerar. Este processo, também chamado de seleção de variáveis, seleção de atributos ou seleção de subconjunto de variáveis, consiste na deteção as características dos dados que são relevantes e eliminação das que são ruídos, as irrelevantes e redundantes (Kumar & Minz, 2014). Segundo os mesmos autores, o processo de seleção de características é composto genericamente por quatro fases: geração do subconjunto, avaliação do subconjunto, avaliação do critério de paragem e validação de resultados. No processo de seleção de

características são utilizados métodos que podem ser classificados como filtros, *wrapper* e *embedded* (de Abreu et al., 2020).

O processo de filtro caracteriza-se por não ser necessário qualquer classificador, ou seja, avalia a relevância de cada característica olhando unicamente para as propriedades dos dados (Saeys et al., 2007). Não utilizando qualquer algoritmo, este método assenta numa métrica, como por exemplo entropia ou coeficiente de correlação, para decidir quais os atributos que mais têm impacto na classe da instância, e quais características podem ser truncadas do *dataset*. Quando comparadas duas características, deverá ser preferida a que trouxer um maior ganho de informação.

O método wrapper utiliza ele próprio um algoritmo de ML para avaliar os atributos a utilizar (ou subconjuntos), sendo que a decisão das características está intrinsecamente ligada com os resultados deste algoritmo. Esta abordagem não exige conhecimento do funcionamento do algoritmo, contudo, apresenta-se como uma solução de mais complexa implementação, de mais complexo desenvolvimento e computacionalmente exigente.

A alternativa aos dois processos descritos anteriormente é o *embedded*, onde a seleção ocorre naturalmente como parte do algoritmo de aprendizagem, sendo parte integrante da construção do classificador (Saeys et al., 2007), sendo assim, como o *wrapper*, específico para cada algoritmo de aprendizagem.

Tendo em conta que a abordagem baseada em filtros é mais versátil para o processo de investigação, dada a complexidade de implementação das abordagens concorrentes e o facto de existir conhecimento prévio do domínio ao qual os dados pertencem, optou-se por esta abordagem.

Posteriormente à seleção das características, existe a possibilidade de se realizar a construção de características, baseadas nas características originais, que permitam uma criação de classificadores mais concisos e precisos, ou seja, descobrir informações não presentes sobre relações entre características, inferindo ou criando características (Motoca & Liu, 2002). Um exemplo prático e quotidiano da construção de características pode ser a definição de uma nova característica IMC, derivada das características pré-existentes peso e altura. Consideradas

independentemente, as características pré-existentes poderão não apresentar um forte ganho de informação para o processo de inferência, mas que, ao ser "construída" a característica que junta estas duas, permita uma maior relação entre esta nova característica e a classe a que o dado pertence.

2.3.3. Aprendizagem supervisionada

Na aprendizagem supervisionada (*AS*) a máquina aprende por exemplo. Segundo (Chinnamgari, 2019) a AS é aplicada quando existe uma clara noção dos resultados que precisam de ser atingido para determinado problema, não sabendo, contudo, como os dados afetam os resultados. Neste tipo de aprendizagem o operador disponibiliza à máquina um conjunto de entradas (*inputs)* e quais as saídas (*output)* que são esperadas, tendo então o algoritmo de identificar os padrões que levam a um determinado resultado com base nos dados iniciais, aprender estas relações e conseguir fazer predições com base neste conhecimento.

A aprendizagem supervisionada divide-se em dois grandes tipos de problemas: classificação e regressão, cada um com os seus algoritmos específicos.

2.3.4. Classificação

Algoritmos de classificação permitem aferir o resultado de instâncias que são de valor somente discreto e não-ordenado (S. B. Kotsiantis et al., 2006). O treino começa com vários exemplos de entrada e as respetivas classes. Depois do treino, o algoritmo de classificação é capaz de identificar, dentro de um grau de precisão, a que classe qualquer dado *input* pertence, sendo este o processo de inferência. Este tipo de algoritmo é utilizado, por exemplo, na identificação de emails como *spam* ou não *spam*, onde dado um conjunto de características de cada instância do conjunto de dados (remetente, destinatário, conteúdo do email, entre outros…) o algoritmo é capaz de prever se determinado email é ou não uma mensagem não desejada. Formalmente, segundo (Lorena et al., 2007) seja f um classificador e F o conjunto de todos os classificadores que um determinado algoritmo de ML pode gerar, então esse algoritmo, durante o seu processo de aprendizagem, utiliza um conjunto de treino T, comporto de n pares (x_i, y_i), para gerar um classificador particular $\hat{f} \in F$.

Alguns dos algoritmos mais utilizados deste tipo são as Bayesian networks, Naïve Bayes, Logistic Regression, Decision Trees, Random Forests, Support Vector Machines, K Nearest Neighbor e discriminant Analysis (Singh et al., 2016).

As Bayesian Networks (BN, em português Redes Bayesianas) são, segundo (Marques & Dutra, 2002) são grafos acíclicos dirigidos que representam dependências entre variáveis num modelo probabilístico, isto é, uma representação da dependência condicional de várias variáveis aletórias. São especialmente vantajosas na modelação de situações onde a causalidade exerce um papel mas onde o nosso conhecimento do que se está a passar é incompleto, tendo então de se descrever a situação probabilisticamente (Charniak, 1991).

Como exemplo considere-se uma rede Bayesiana que envolve a variável aleatória de uma pessoa chegar ou não atrasado ao trabalho. Numa manhã normal, a pessoa acorda e faz a sua rotina normal. Habitualmente, esta pessoa desloca-se para o trabalho de barco. Existe a possibilidade de estar a chover. A existência de uma greve de transportes implicará, provavelmente, que não existam tantos transportes disponíveis, tendo a pessoa de optar por ir de carro para o trabalho. A chuva, caso seja uma chuva forte, pode também afetar a possibilidade de o transporte estar disponível, como no caso de um barco, por exemplo, optando assim a pessoa também por ir de carro. Também não é taxativo que se houver uma greve convocada a pessoa não possa ir de barco, pode ter a sorte de um dos horários em se utiliza barcos não ser afetado. Se existir trânsito para a entrada da ponte, quase de certeza que esta pessoa chegará atrasada ao trabalho. Somente se for de carro é que a pessoa se tem de preocupar com o trânsito, sendo que se for de barco, como normalmente em princípio irá chegar a horas ao trabalho.

Dentro desta descrição, que serve para modelar a variável de chegar ou não atrasado, existem outras variáveis que a condicionam, como a chuva, trânsito, que transporte utiliza ou o facto de haver greve. Este problema e as suas informações estão representadas na Figura 4.

Conseguimos ver que tanto a chuva como a greve afetam o meio de transporte, dado que se houver greve ou se estiver chuva forte a pessoa vai de carro. O trânsito só interessa se a pessoa for de carro e é afetado pela chuva. O meio de transporte e

o trânsito afetam a hora de chegada da pessoa ao trabalho. O BN apresenta como vantagem a fácil compreensão e implementação, facilidade de identificar interdependência entre variáveis e prever eventos, sendo as principais desvantagens a elevada exigência computacional face a outros algoritmos similares, a falta de dados probabilísticos e a assumpção que todas as variáveis são condicionalmente independente (Adebowale et al., 2013).

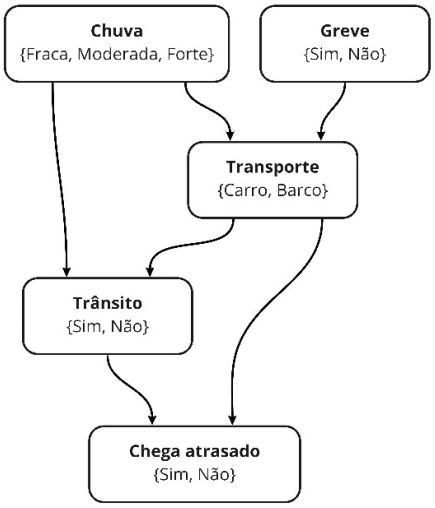

Figura 4 - Grafo de Rede Bayesiana.
Fonte: Autoria Própria.

De notar que, na Figura 4 as setas, chamadas num grafo de arestas, estão a apontar para uma variável em específico, representadas por retângulos, estes que num grafo se chamam nós, não podendo ser revertidas. Não é possível dizer que o facto da pessoa ir de carro afeta o facto de existir ou não chuva mas é possível afirmar o contrário. Formalmente, uma aresta (a, b), em que A e B são nós do grafo que representam variáveis aleatórias então $P(b|a)$. É possível, quando não se sabe o valor de $P(B|A)$, recorrer ao teorema de *Bayes*, através da seguinte fórmula:

$$P(b|a) = \frac{P(b) * P(a|b)}{P(a)}$$

Cada um dos eventos apresentados nos nós tem uma probabilidade associada, por exemplo a probabilidade de haver chuva é dada por meteorologistas e tem um valor, neste caso um valor discreto, como fraca, moderada ou forte. Isto é, a probabilidade de existir chuva $P(chuva) \in$

$\{0; 1\}$, chuva: $\{\text{fraca}, \text{moderada}, \text{forte}\}$. Isto que dizer que é possível calcularmos a probabilidade da pessoa ir de barco dado que existe chuva forte, utilizando o conceito de probabilidade condicionada, dado pela expressão $P(barco|chuva\ forte)$. Sabendo à partida algumas probabilidades de eventos e de arestas, é possível calcular a probabilidade de uma cadeia de eventos, como por exemplo, sabendo a probabilidade de existir trânsito caso chuva forte e a probabilidade de ir de carro dado a mesma chuva forte, conseguiremos calcular a probabilidade de chegar atrasado dados todos estes eventos.

A Naive Bayes (NB) é um caso específico de uma Rede Bayesiana e a sua utilização tornou-se popular para detecção de e-mails *spam*. Este algoritmo caracteriza-se pela desconsideração da correlação das variáveis, onde o nó de classificação é apresentado como pai dos restantes nós (Friedman et al., 1997). Formalmente, diz-se que tendo em conta instâncias de treino e o caso de teste E representados por n atributos $(a_1, a_2, \dots, a_n)$, o classificador Bayesiano, segundo (Al-Aidaroos et al., 2010), utiliza a equação E:

$$C_{NB}(E) = arg_{c\epsilon C} \max P(c) \prod_{i=1}^{n} p(a_i|c)$$

Como vantagem, o NB é simples e fácil implementar, não requer muitos dados de treinos, tem a capacidade de tratar de dados discretos e contínuos e é altamente escalável sendo as desvantagens do mesmo a dificuldade do algoritmo melhorar os resultados e, dado que assume que as variáveis são consideradas como independentes, é sensível a atributos reduntantes ou irrelevantes, existindo uma tendência para que atributos que sejam altamente correlacionados recebam muito peso na decisão final do algoritmo (Al-Aidaroos et al., 2010).

A Logistic Regression (LR, regressão logística) é um modelo matemático que permite a predição de relação entre observações e um conjunto classes discretas. A grande diferença entre LR e NB é o facto de LR ser um classificador discriminativo, enquanto NB é um classificador generativo (Daniel & Martin, 2021). Este classificador, por exemplo, tem a possibilidade de classificar observações em duas classes ou mais classes. A LR tem como base a função logística e, especialmente,

um caso particular desta, a função sigmoidal. Esta função em forma de "s", dada pela expressão:

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

A expressão permite o mapeamento entre probabilidade e predição, já que os seus valores se encontram entre zero e um. A função sigmoidal necessita de uma fronteira de decisão que permita o mapeamento do resultado da função para classes discretas, como por exemplo "sim/não" e "fraude/não fraude". A utilização da LR, segundo (Park, 2013), segue as seguintes suposições:

- A variável dependente deve ser de natureza discreta, preferencialmente dicotômica.
- Dado que a LR estima a probabilidade de um determinado evento ocorrer, então é necessário codificar as variáveis dependentes de acordo com esse facto;
- Cada observação deve ser independente;
- Não deve conter variáveis irrelevantes nem ignorar variáveis importantes;
- Deve haver pouca ou nenhuma multicolinearidade entre as variáveis independentes, isto é, as variáveis independentes não devem ser altamente correlacionadas entre si.
- A LR assume linearidade das variáveis face ao valor logarítmico da probabilidade de dado evento;
- É requerida uma amostra grande de dados.

Como exemplo de utilização da regressão logística, considere-se um stand automóvel que pretende entender a probabilidade de certa pessoa entrar na loja e comprar um carro. Como variáveis independentes, podem ser consideradas, por exemplo a idade e salário e a variável dependente, a classe a identificar, representa se a pessoa comprou ou não um carro. Neste caso, cada um dos casos históricos de um cliente comprar ou não um carro poderia ser dado por $f(idade, salário)$, onde i representa a idade do cliente, s o salário do cliente e c se comprou ou não o carro. O objetivo do algoritmo de ML é encontrar uma função $h(idade, salário)$ que se aproxime do comportamento de f.

Sendo a fronteira de decisão, por exemplo $h = 0{,}5$, então a previsão v é dada pela seguinte expressão:

$$v = \begin{cases} 0, se\ h < 0{,}5 \\ 1, se\ h \geq 0{,}5 \end{cases}$$

As Support Vector Machines (SVM) são modelos utilizado para problemas de regressão ou classificação que visa a obtenção de fronteiras lineares para a separação de dados pertencentes a duas classes (Lorena et al., 2007). Este algoritmo pretende encontrar um hiperplano que divida os dados conforme as suas classes. Na sua forma mais simples, SVM lineares, definem fronteiras lineares a partir de dados linearmente separáveis. Como se pode ver na Figura 5, existem dois tipos de observações, ou classes, que são separados por um hiperplano H definido pela função $w \cdot x + b = 0$. No exemplo, as duas classes são facilmente separáveis, são considerados os triângulos e os círculos. H é a linha que separa as duas classes e fica o mais distante possível das observações de treino mais próximas. É por isso criada uma margem que permite a tarefa de classificação. Se adicionadas novas instâncias fora da margem, o classificador terá de facilidade em categorizar estas.

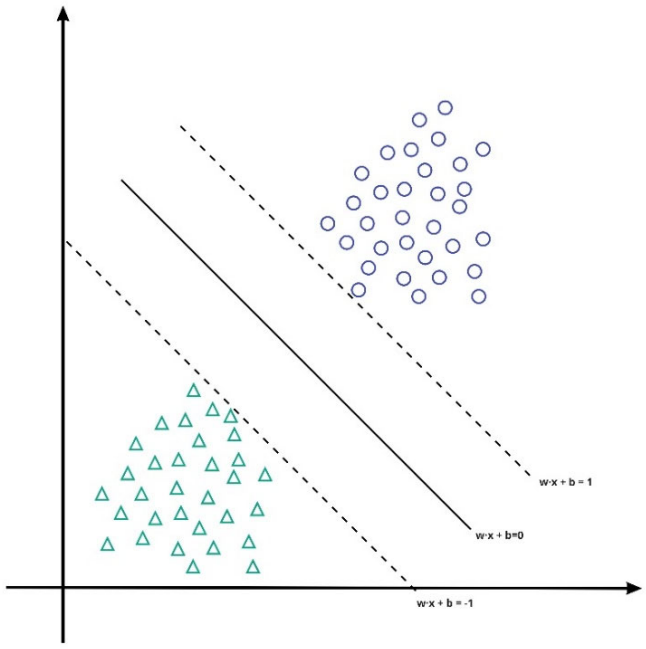

Figura 5 - Exibição de SVM.
Fonte: Autoria Própria.

2.3.5. Classificações de documentos

Como referido anteriormente, tem de se ter em conta dois tipos de classificação no processamento de documento: classificação de tipo de documento e classificação de dados de negócio.

A classificação de documentos pode ser feita através de algoritmos supervisionados ou não supervisionados. Havendo muitos modelos a utilizar nesta tarefa, destacam-se as SVM, que atingem uma taxa de precisão, nesta tarefa, que rondam os 99% (Basha et al., 2019). A testagem de modelos que permitem atingir estas percentagens de acerto serão descritas no ponto 4.1 do presente documento.

Já a classificação de dados de negócio, atividade que pertence a um subconjunto de problemas de ML chamado *Named Entity Recognition* (reconhecimento de entidades nomeadas, NER). Englobadas no NER, existem várias abordagens a adotar quanto à natureza dos modelos a utilizar, nomeadamente, modelos baseados em regras/heurísticas, modelos supervisionados, modelos semi-supervisionados, modelos distantemente supervisionados, modelos não supervisionados e modelos de modelação conjunta (Nasar et al., 2021).

Diferentes tipos de modelos têm grandes variações de precisão no que diz respeito aos resultados apresentados. De notar que os conjuntos de dados que se utilizam, nomeadamente no que diz respeito à sua qualidade e dimensão, influenciam bastante os resultados obtidos, sendo um desafio a obtenção/criação destes conjuntos de dados. Na classificação de dados de negócio e na identificação das relações entre si destacam-se os modelos de modelação conjunta e os classificadores SVM (Support Vector Machines), RNN (Recursive Neural Network), Bi-LSTM (Bi-Directional Long Short-Term Memory) e CNN (Recursive Neural Network) (Nasar et al., 2021).

Existem também alguns modelos que têm vindo a ser desenvolvidos (até no decorrer da escrita deste documento) para uma subárea de ML, a DocAI. Nesta área destaca-se o LayoutLM (Xu et al., 2020), um modelo que tira partido de uma modelo BERT (Devlin et al., 2018) adicionando-lhe uma componente de extração de informação visual. O BERT, modelo baseado em atenção que aprende as relações contextuais entre palavras, sendo treinado bidirecionalmente, isto é, permitindo uma aprendizagem contextual de uma palavra baseada nas restantes à sua volta, e não só da esquerda para a direita ou vice-versa. O LayoutLM tira partido desta técnica de extração de informação textual, adicionando mecanismos para extrair ainda dados relativos à disposição do documento, partindo da premissa que a posição relativa de uma palavra num documento contribui para a sua representação semântica e informação visual, a representação visual do texto e documento, como por exemplo negritos, sublinhados. O LayoutLM utiliza a arquitetura BERT como base de trabalho, adicionando-lhe dois *inputs embeddings*, sendo estes o posicionamento 2D de uma palavra (face ao restante documento e não somente à frase em que está inserido) e uma representação visual do texto, sendo utilizada uma Faster R-CNN para a geração destas características. Este modelo apresenta resultados bastante satisfatórios na tarefa de extração de dados de documentos, apresentando precisões bastante mais elevadas que modelos BERT tradicionais.

2.4. OCR

Existem várias ferramentas atualmente no mercado que permitem fazer o reconhecimento de caracteres. A Google, Microsoft, IBM e Amazon apresentam ferramentas de nível empresarial para esta tarefa e existem ainda algumas outras soluções que apresentam bibliotecas *opensource* para o reconhecimento. Foram identificadas as seguintes soluções comerciais de OCR a ser consideradas: Vision API (Google), Cloudmersive API (Cloudmersive), Azure API (Microsoft), IBM API (IBM), Textract (Amazon), Tesseract (*Opensource* - Patrocinado pela Google).

Grande parte das soluções comerciais disponíveis são analisadas como parte integrante dos serviços de Inteligência Artificial dos fornecedores dos mesmos, sendo que nesta categoria destacam-se a Microsoft, Google, IBM e Amazon (Gartner et al., 2020). Uma comparação destas soluções será descrita no ponto 4.1 do presente documento.

A tarefa de OCR nada mais é que uma atividade de classificação: dada uma entrada x, um algoritmo deverá calcular a probabilidade de essa mesmo entrada ser qualquer uma das classes, neste caso as letras e números, $[P(x) = \text{A}, P(x) = \text{B}, \dots, P(x) = n]$. Neste processo destaca-se uma arquitetura, um conjunto de passos e modelos a utilizar para esta tarefa, as LRCN (Long-term Recurrent Convolutional Networks). Esta abordagem permite que as imagens sejam pré-processadas numa primeira fase, passando depois por uma CNN e depois por uma LSTM (Long Short-Term Memory), gerando depois um *output* (Donahue et al., 2017). A utilização das CNN permite a extração de características da imagem e identificação de áreas de interesse, sendo os resultados desta passagem posteriormente alimentados para a LSTM que são responsáveis pelo *deep learning* criando aqui uma camada que relaciona os caracteres para uma melhor deteção.

2.5. Melhoramento de Imagem

No processo de reconhecimento existe a possibilidade de o *input* não estar nas melhores condições para ser processado. Existem várias metodologias para melhorar uma imagem e vários passos que se podem seguir para obter um melhor ponto de partida no processo de reconhecimento ótico. O pré-processamento de

imagem não consiste somente num passo, mas sim numa sequência de passos: binarização, normalização, amostragem, eliminação de ruído e desbaste (Poovizhi, 2014a).

A binarização é o primeiro passo no melhoramento da imagem. Esta técnica consiste na segmentação do plano principal e do plano de fundo de uma imagem (Jyotsna et al., 2016), passando uma imagem a cores para uma imagem a preto e branco e alterando-a para realçar tudo o que não é considerado o plano de fundo da imagem.

Figura 6 - Comparação de técnicas de binarização
Fonte: Autoria Própria

Esta técnica, como demonstrado na Figura 6, permite obter resultados variados no que toca à legibilidade do documento, mas também no que diz respeito à capacidade de um algoritmo conseguir entender os caracteres, devendo assim ser testadas várias abordagens para comparação de qual a que mais se adequa a cada documento.

Após terminada a binarização, segue-se a normalização. Este passo consiste no processo de transformar um conjunto de *inputs* numa forma normal, invariante em translação, rotação, escala e enviesamento (Pei & Lin, 1995).

Após o processo de normalização, tendo já dados de entrada uniformes, é possível efetuar a amostragem. O processo de amostragem, como o conceito matemático, consiste em escolher um subconjunto de um todo para analisar, estudar características que depois possam ser extrapoladas para o todo. A amostragem permite, por exemplo, o ajuste de resolução de uma determinada imagem de

entrada, já que é possível extrapolar o valor de um *pixel* com base num conjunto de outros.

É possível ainda a remoção de ruído da imagem. Numa qualquer imagem é provável que se encontre algum tipo de ruído. Este problema é principalmente recorrente em imagens capturadas por câmeras fotográficas que não sejam de última geração ou *scanners* que não sejam eficientemente mantidos.

Existem algumas técnicas matemáticas de remoção de ruído, como por exemplo filtro mediano 2D, desfoque gaussiano e filtro passagem baixa. Recentemente a utilização de CNN para remoção de ruído tem vindo a aumentar, nomeadamente WNNM, DnCNN-S, SCNN, FFDNet, NN3D, PDNN e PSNR (Thakur et al., 2019). A utilização de redes neurais para a tarefa de remoção de ruido permite uma maior performance na tarefa enquanto se conseguem atingir melhores resultados, tendo em conta que o modelo evolui com o treino.

O desbaste é uma fase opcional e consiste em encurtar a largura as linhas de caracteres escritos manualmente para que fiquem com um pixel de largura (Poovizhi, 2014b). Esta técnica permite desconsiderar, por exemplo, as diferenças de largura no traço dos caracteres, sendo assim um passo de padronização.

2.6. Avaliação de algoritmos de *Machine Learning*

Como descrito nos tópicos anteriores, existem alguns tipos de problemas, aprendizagens e modelos diferentes. Surge a necessidade de conseguir comparar os vários modelos que podem ser aplicados a determinado problema, de forma a avaliar de que forma atingem os objetivos esperados e quais as opções que apresentam melhores resultados. Dado que diferentes parâmetros originam diferentes modelos, e consequentemente diferentes resultados, é imperativo que existam mecanismos para comparar as diferentes abordagens possíveis. Pode ser tentador utilizar o resultado da inferência como única métrica de avaliação, partindo da premissa que um modelo tem uma qualidade superior a outro modelo quando apresenta predições corretas mais vezes, o que não é necessariamente verdade. A acurácia assenta no quociente das previsões certas pelo número total de previsões, ou seja, é a percentagem de previsões corretas que o modelo executa (Kubat et al., 1998). Esta métrica é representada pela equação:

$$Acurácia = \frac{Número\ de\ instâncias\ corretamente\ classificadas}{Total\ de\ instâncias}$$

Contudo, esta métrica apresenta algumas desvantagens quando utilizada como única métrica para a qualidade de um modelo, já que negligencia o tipo de erros que ocorrem e é dependente na distribuição das classes no *dataset*.

Na aplicação de modelos de ML ao mundo real, a distinção dos tipos de erro que um modelo pode gerar é imperativa. Um exemplo prático seria a deteção de fraude em transações bancárias. Um falso positivo, ou seja, uma transação fidedigna que é classificada como uma transação maliciosa apresenta menos risco que um falso negativo, que corresponde a uma classificação de uma transação fraudulenta como legítima. No caso do falso positivo, provavelmente algum operador iria rever a transação e classificá-la como legitima, ao passo que no caso do falso negativo, a transação iria passar despercebida e poderia trazer custos elevados para a instituição. Usualmente elabora-se uma matriz que apresenta a distribuição das previsões conforme as suas classes, denominada de matriz de confusão.

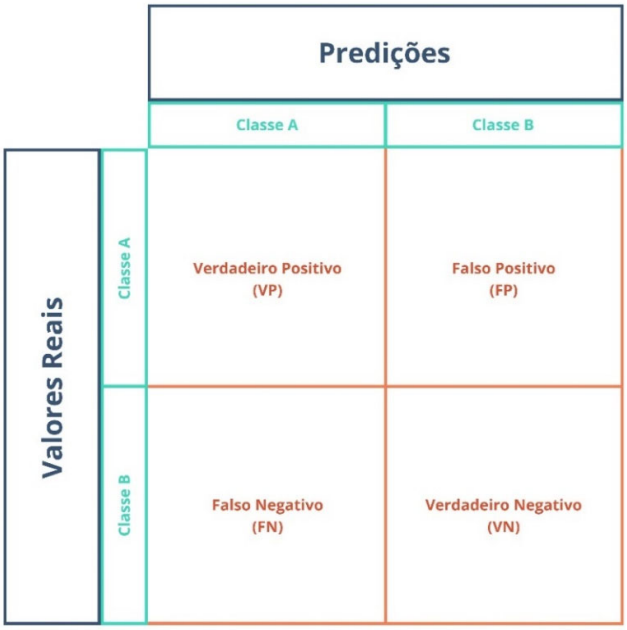

Figura 7 - Exemplo de matriz de confusão.
Fonte: Autoria Própria

Na Figura 7 é possível observar que existe uma classe que assume o valor de positiva, que no exemplo anteriormente descrito poderia representar uma transação

ser fraudulenta, e uma classe negativa. No caso de um problema que resulte em mais que duas classes, todas as classes que não são o objetivo da métrica, adquirem o valor negativo (Dj Novakovi et al., 2017).

Da matriz de confusão é ainda possível extrair outras métricas que são úteis na comparação de modelos, nomeadamente: Taxa de Falsos Positivos, Taxa de Verdadeiros Negativos, Taxa de Falsos Negativos, Taxa de Verdadeiros Positivos e Precisão, sendo estes últimos os mais relevantes para o estudo atual.

A Taxa de Verdadeiros Positivos, usualmente chamada de *Recall*, apresenta o quão eficaz é um modelo a prever casos positivos, ou seja, pode ser representada pela equação (Tatbul et al., 2018):

$$Recall = \frac{VP}{VP\ + FN}$$

A Precisão é uma métrica que permite aferir o quão bem o modelo é a identificar positivamente os exemplos e é representado pela seguinte equação (Dj Novakovi et al., 2017):

$$Precision = \frac{VP}{VP + FP}$$

Complementarmente a estas métricas derivadas da matriz de confusão, existem métricas como o *f-score* que exibem qual a relação entre a precisão e o *recall* do modelo, como dado pela expressão (Sasaki, 2007):

$$\text{f-score} = 2 * \frac{precision * recall}{precision + recall}$$

Sendo estas métricas essenciais para a avaliação dos modelos, existem algumas bibliotecas e pacotes de desenvolvimento que agilizam os processos de cálculo, como é o caso do SKLearn (Pedregosa et al., 2011), pacote utilizado para este efeito no desenvolvimento deste projeto.

2.7. Síntese do enquadramento

O presente capítulo apresenta uma comparação do mercado de faturação digital e faturação tradicional, mostrando a literatura que ainda existe uma margem significativa para progredir, sendo a faturação digital uma abordagem que apresenta vantagens a nível financeiro, processual e ambiental. Foi ainda introduzido o conceito de *data science*, uma área de estudo bastante abrangente e com capacidade para diferentes tipos de análises, dependendo da natureza de dados e objetivos, tendo sido aprofundada a explicação do campo de *machine learning*, dada a natureza do projeto a desenvolver. Englobados no ML, destacaram-se dois processos essenciais para o processo de aprendizagem, nomeadamente o pré-processamento e seleção de características. Havendo várias abordagens, exibiu-se a escolha pela aprendizagem supervisionada e descreveu-se a classificação, um tipo de problema frequente nesta área. Abordou-se ainda a classificação de documentos, o melhoramento de imagens e reconhecimento ótico de caracteres, áreas estas que estão intimamente ligadas com os componentes a desenvolver ao longo do projeto.

No próximo capítulo será enquadrada a metodologia utilizada neste projeto, definindo-se os métodos e instrumentos utilizados e relacionando o projeto com seus desafios e hipóteses.

3. Metodologia

3.1. Descrição de problema e hipóteses

Existe, como descrito anteriormente neste documento, ainda uma grande utilização do papel nos processos de faturação, onde as faturas neste meio são ainda as mais frequentes. O processamento atual é maioritariamente efetuado através de processos manuais e que consomem tempo, requerendo ainda profissionais especializados para os executarem.

Com base na crescente transformação digital e tecnológica a ser aplicada no setor empresarial, surge a seguinte questão: "O processamento atual de documentos financeiros, ineficaz e que leva a desperdício de recursos, aumentando a taxa de erros pode ser auxiliado por uma ferramenta que recorra a ML?".

Apesar de ser possível a continuação do processo atual ou a utilização de soluções RPA existentes, para responder a esta questão, são propostas as seguintes hipóteses para melhorar o processamento atual: o desenvolvimento de uma solução à medida, com recurso a ML, ajuda no processamento de faturas.

Posto isto, pretende-se desenvolver um estudo para entender se seria possível, com uma solução de *machine learning (ml),* melhorar e automatizar o processo de reconhecimento, processamento e catalogação de faturas, havendo vários objetivos a ser alcançados:

- Investigar a viabilidade do processamento de faturas ser auxiliado por formas automatizadas;
- Comparar diferentes algoritmos nas diferentes áreas do processamento;
- Investigar a aplicabilidade de métodos já estudados para a realidade portuguesa.

Este desenvolvimento assenta na elaboração de uma prova de conceito (POC), que tem por objetivo primário a construção de uma solução capaz de efetuar o processamento automatizado de documentos, extraindo o texto presente nos mesmos, classificando-os e inferindo os campos de negócio em si presentes, permitindo a avaliação qualitativa quando comparada com um conjunto de soluções atualmente presentes no mercado. A elaboração do referido POC permite ainda uma comparação quantitativa de soluções e modelos de OCR bem como a comparação quanlitativa de modelos de reconhecimento de tipos de documentos e de NER.

3.2. Descrição da metodologia

Para a resolução do desafio encontrado, a identificação e classificação de campos em documentos financeiros, foram identificados os processos de negócio usualmente relacionados com esta atividade e foi desenvolvida uma solução, em formato de POC, assente em tecnologias Microsoft, que permite oferecer interfaces para que o utilizador consiga processar um documento de forma automática. O desenvolvimento de uma prova de conceito permite a comparação qualitativa dos resultados obtidos no processamento de documentos, quando comparados com os resultados obtidos com tecnologias existentes no mercado.

O desenvolvimento de uma prova de conceito, que inclui todo o processo necessário para o reconhecimento de documentos e campos de interesse, tem de ter em conta as seguintes fases:

1. Recolha de dados de teste e treino;
2. Comparação de abordagens de OCR;
3. Comparação de modelo de classificação de entidades;
4. Construção de prova de conceito;
5. Elaboração de avaliação do desenvolvimento efetuado com base nos resultados;
6. Testes de usabilidade e/ou análise qualitativa.

A primeira fase apresenta um desafio, visto que não existem muitos dados específicos disponíveis para o treino e teste de modelos de ML.

Todo o processo de desenho e validação do projeto decorreu entre março de 2020 e agosto de 2021, estando envolvido no processo de definição de solução, desenvolvimento da mesma e análise de resultados o autor deste documento bem como a equipa de R&D da KCS IT S.A., consultora especialista na área das tecnologias de informação e gestão de projeto. Espera-se aferir, em termos quantitativos, quais as melhores soluções para cada um dos componentes da solução para que o POC final obtenha um resultado satisfatório na sua avaliação qualitativa.

Para avaliação da prova de conceito desenvolvida, terão de ser utilizados vários critérios qualitativos e quantitativos. Por estar em causa a automatização de um processo que usualmente é levado a cabo por um operador humano, onde existe um fator de interação pessoa e máquina, não basta simplesmente comparar o resultado do processo manual em comparação com o automatizado, já que, por exemplo, o operador pode, num processo sub-ótimo, adaptar-se para compensar uma falha de desenho do sistema/processo (Madhavan et al., 2009). Segundo (Pina et al., 2008), existem cinco classes generalizadas para o controlo de supervisão humana, nomeadamente métricas de eficácia de missão, eficiência comportamental da plataforma, eficiência do comportamento humano, precursores do comportamento humano e métricas colaborativas. As métricas utilizadas e as comparações serão

detalhadas do ponto 4.5.6 do presente documento, onde são apresentados os dados recolhidos para efetuar a comparação, bem como a pertinência das métricas.

3.3. Instrumentos e materiais utilizados

O desenvolvimento das comparações anteriormente mencionadas foram feitos com recurso aos ambientes de desenvolvimento Visual Studio 2019, Visual Studio 2022, Visual Studio Code e PyCharm 2020.3.2.

O desenvolvimento do POC tem como base tecnológica a *framework* .Net 6, sucessora do .Net *framework* e que apresenta como vantagens o facto de ser *cross-platform*, a sua performance, portabilidade e o facto do seu código ser aberto ao público em geral (Troelsen & Japikse, 2017), o que permite que a base de desenvolvimento seja estendida conforme necessário e auditada pela comunidade. A criação de interface gráfica para que o utilizador final possa interagir com a solução foi inicialmente desenhada para ser desenvolvida em React, sendo posteriormente implementada com recurso a *Bootstrap*, uma *framework* para desenvolvimento web que tem por base o HTML, CSS e Javascript. A escolha do *Bootstrap* prende-se com a sua performance, com a sua simplicidade de implementação e capacidade de adaptabilidade e usabilidade. Para o desenvolvimento de modelos de Machine Learning foi utilizado o Python 3.8, linguagem de programação amplamente utilizada para desenvolvimento de projetos de Inteligência Artificial.

Os dados de treino e teste dos modelos a utilizar nas classificações são provenientes de um conjunto de dados proprietário composto por diversos documentos, nomeadamente faturas, notas de crédito e guias de transporte. Foram ainda usados dados provenientes de desafios disponíveis na internet, que visam a solucionar problemas similares aos abordados pela solução a desenvolver. Foi utilizado o conjunto de dados ICDAR 2019 (Huang et al., 2019), um conjunto de dados composto por recibos que foram previamente anotados e classificados.

A obtenção de conjuntos de dados para treino, teste e validação dos modelos de ML demonstrou-se desafiante. Cada dado a utilizar é composto por dois elementos: 1) uma imagem que contem uma impressão do documento; 2) um ficheiro de texto que contém os dados espaciais, o texto e classificação referentes à

imagem. Os dados existentes surgem, muitas vezes, com ausência de alguns dos campos que se pretende identificar e em idiomas que dificultam o desenvolvimento de um modelo de ML consistente.

A criação de conjuntos de dados (*datasets)* com base em documentos próprios apresenta também os seus desafios. Para cada uma das imagens que representa, por exemplo, uma fatura, um operador terá de identificar manualmente a localização do texto na mesma e classificar esse mesmo texto conforme as classes a identificar, sendo este um processo moroso e dispendioso.

Na tarefa de comparação de soluções OCR, espera-se que as soluções comerciais obtenham quantitativamente melhores resultados que as opções *open-source*. A escolha da solução a utilizar para o POC deverá, contudo, ter em conta também as limitações, o custo e a fiabilidade de cada solução comparada.

Espera-se ainda que na tarefa de classificação de dados de negócio e tipos de documentos, a SVM obtenha melhores resultados, já que existe uma amostra reduzida de dados de treino de diferentes classes.

4. Implementação

De modo a validar que é possível colmatar a necessidade de um método automatizado de processamento, optou-se pela investigação da área de estudo e elaboração de uma prova de conceito que suprisse as principais necessidades dos *stakeholders*.

4.1. Comparação de OCR

Para decidir que ferramenta de base utilizar para o reconhecimento ótico de caracteres, primeiro é necessário desenvolver um comparador que coloque lado-a-lado as soluções existentes no mercado. Havendo já algum trabalho feito nesta área por grandes *players* como a Microsoft, Amazon e Google e soluções *opensource* prontas a utilizar, optou-se pela escolha de serviços e bibliotecas prontas a utilizar por desenvolvedores, ao invés da construção de um modelo de ML proprietário. Estas ferramentas recebem como *input* uma imagem e retornam, na sua maioria, o texto presente na imagem enviada e as caixas que delimitam o texto, havendo ainda

algumas soluções que retornam confiança com que o texto foi reconhecido e o conteúdo divido por palavra, linha e parágrafo. Juntando as informações enviadas com as recebidas podemos entender como estas soluções funcionam, como mostrado na Figura 8. Podemos ver que o texto no logótipo é reconhecido, com uma confiança perto dos 100%:

Figura 8 - Demostração dos passos de reconhecimento de caracteres.

Fonte: Autoria Própria

Na Figura 8 é possível ver em 1) a imagem enviada para o serviço OCR (Tesseract), em 2) a resposta do serviço com identificação do texto e geometria e em 3) a imagem sobreposta pela geometria retornada.

Existem, contudo, algumas limitações no texto reconhecido: A palavra Instituto foi agregada à sigla IPS e não ao restante nome da instituição; dada a natureza da imagem, existiu alguma dificuldade por parte do serviço a identificar o que eram as linhas do texto, visto existir texto de cobre duas linhas; os acentos são completamente ignorados no retorno. Estes aspetos serão encarados como limitações das soluções existentes e irão ser colmatados nos próximos capítulos deste documento.

O primeiro desafio do desenvolvimento do mecanismo de comparação é a integração de todos os serviços numa só aplicação. Como nenhum dos serviços testados retorna a mesma estrutura de resposta, foi necessário criar uma interface

que conseguisse padronizar os resultados recebidos de modo a estes serem comparados. A solução desenvolvida é apresentada na Figura 9.

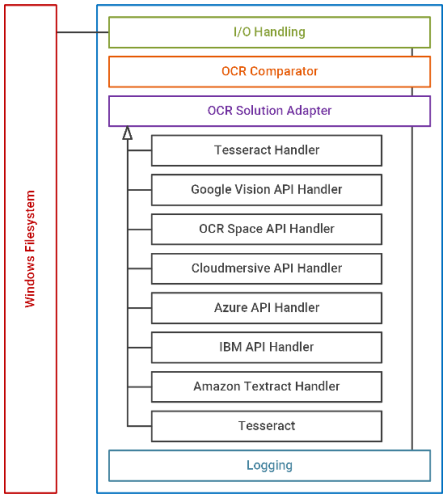

Figura 9 - Diagrama que representa a arquitetura da solução de comparação desenvolvida
Fonte: Autoria Própria

A solução desenvolvida, que assenta em tecnologias *Microsoft*, tira partido do padrão de desenvolvimento *Adapter* e das boas práticas de programação orientada a objetos, para uma comparação das várias tecnologias. Como exibido na Figura 9, a solução tem como principais componentes uma classe de *logging*, de gestão de entradas e saídas, uma classe de comparação dos algoritmos e uma implementação do padrão *Adapter* onde cada solução de OCR é representada por um adaptador. A classe de *logging* permite registar cada execução de cada um dos algoritmos, bem como erros e avisos decorrentes destas execuções. Esta classe foi implementada com recurso ao padrão *singleton* para garantir a unicidade de instanciação e melhor gestão de pedidos.

A classe responsável pelas entradas e saídas tem é composta por três objetos: um módulo que permite importar dados de entrada (*pdfs* ou ficheiros de imagem) e lidar com eventuais erros; um conversor de *pdfs*, que permite transformar entradas de ficheiros *pdf* em imagens; um módulo que permite a escrita de resultados em formato *xlxs* compatível com o Microsoft Excel, para posterior manipulação dos resultados da comparação.

A classe *OCR Comparator* representa o motor de comparação de cada uma das soluções de OCR, calculando cada um dos parâmetros e lidando com a fila de pedidos de processamento.

Cada uma das soluções implementam a interface *OCR Solution Adapter*. Esta interface permite que todas as soluções de OCR, independentemente de como são implementadas, tenham sempre o mesmo comportamento. O desenvolvimento de cada um dos componentes de OCR utilizou diferentes bibliotecas e *SDKs*, disponibilizados pelos desenvolvedores e utilizou na sua maioria pedidos *http* para obter os resultados do processamento de cada ficheiro.

De seguida, foi necessário identificar que dados utilizar na comparação das soluções. Era necessário encontrar imagens que tivessem sido previamente etiquetadas, para que fosse possível comparar com cada uma das respostas dos serviços. Este *dataset* de teste foi extraído de um desafio de reconhecimento de texto em imagens, proposto pela Universidade Autónoma de Barcelona. Este desafio visa detetar texto em imagens do quotidiano, como sinais de trânsito, placas sinalizadoras ou produtos. Este *dataset* é composto por 229 imagens e 229 ficheiros de texto com as coordenadas das caixas de delimitação de palavras e o texto que nelas está presente.

Depois de adquirido o *dataset* foi necessário definir uma métrica para comparar o texto que se sabe estar contido na imagem com a resposta dos diferentes serviços de OCR. Uma comparação que binariamente define se a resposta é igual aos dados etiquetados (ou dados de referência) não é suficiente pois cada serviço poderá, por exemplo, detetar mais uma linha que os dados de referência, mas o texto contido nestas linhas ser o mesmo.

Optou-se por utilizar algoritmos que comparam padrões no texto e não só se dois conjuntos de caracteres são iguais. Foram escolhidos os algoritmos da Distância de Levenshtein (Levenshtein, 1966) e o Reconhecimento de Padrões Ratcliff/Obershelp (Black, 2004) para extrair as métricas de qual serviço de OCR se aproxima mais dos dados de referência. A escolha destes dois algoritmos prende-se com o seu vasto uso na área de estudo, performance e simplicidade de implementação.

Esta análise não tem qualquer comparação de custo de utilização de serviços um fator, contudo, importante na escolha de uma solução base para um contexto empresarial. Esta é uma análise puramente baseada na diferença do texto de referência para o texto devolvido para cada serviço.

Na Tabela 2 são apresentados os resultados da comparação das diferentes soluções de OCR. Como métrica de comparação foram utilizados o tempo total de execução (TTE), tamanho dos pedidos ao serviço (TPS), distância de Levenshtein (DL) e precisão média (PM).

O TTE, expresso em segundos, é utilizado para medir o tempo total de execução de cada pedido, desde o início do pedido até ao cálculo de métricas e escrita de resultados. Esta métrica, assumindo que para qualquer uma das soluções OCR o mesmo documento irá demorar o mesmo tempo a ser carregado, permite aferir a rapidez com que o servidor ou serviço específicos da solução identificam o texto e retornam uma resposta. O TPS, em kilobytes, mede o tamanho de pedidos ao servidor de cada uma das alternativas (as soluções de são *offline* ignoram esta métrica). O TPS permite estimar possíveis custos com infraestrutura e conexão, dado que a maioria dos serviços *cloud* indexam os seus preços a estas variáveis. A precisão média, medida em percentagem, é definida pela expressão $PM = \frac{1}{n} * \sum_{i=1}^{n} Pc(x_i)$, $Pc(x_i) \geq 0$ onde n representa o número de blocos de texto identificados no documento e $Pc(x_i)$ representa a precisão de cada um dos blocos, sendo só contabilizadas todas as precisões superiores a 0, dado que blocos de texto a branco usualmente têm precisão de -1 ou 0.

A Distância de Levenshtein, expressa em percentagem, é a uma métrica de distância entre um texto de origem (s) e um de destino (t), sendo essa distância o número de eliminações, inserções e substituições que são necessárias para transformar s em t (Haldar & Mukhopadhyay, 2011).

A LD é, segundo os mesmos autores, calculada com base no seguinte algoritmo:

1) Inicialização
    a. Define-se n para ser o tamanho do texto fonte s e m para ser o tamanho do texto objetivo t;
    b. Contruir uma matriz contendo m linhas e n colunas
    c. Inicializar a primeira linha com os valores $[0,1, \dots, n]$
    d. Inicializar a primeira coluna com os valores $[0,1, \dots, m]$
2) Processamento
    a. Examinar s (i de 1 a n)
    b. Examinar t (j de 1 a m)
    c. Se $s[i] = t[j]$, o custo é 0
    d. Se $s[i] \neq t[j]$, o custo é 1
    e. Assignar à célula da matriz $d[i, j]$ igual ao mínimo entre:
        i. A célula imediatamente acima mais 1: $d[i-1\ j] + 1$
        ii. A célula imediatamente à esquerda mais 1: $d[i\ j-1] + 1$
        iii. A célula diagonalmente acima e à esquerda mais o custo: $d[i-1\ j-1] + custo$
3) Resultado
4) Repetir 2) até o valor de $d[n, m]$ ser encontrado

O cálculo da fórmula para comparação de cada uma das soluções de OCR tem com base todos os parâmetros anteriormente mencionados, sendo o TTE e a PM os dois parâmetros de maior importância. A pontuação de cada solução segue a seguinte expressão:

$$PF(s) = PM(s) * \frac{2}{5} + (\frac{60-TTE(s)}{60}) * \frac{3}{10} + \frac{DL(s)}{5} + (\frac{400-(TPS(s)/100)}{400}) * \frac{1}{10}.$$

Tabela 2 - Resultados da comparação de soluções OCR
Fonte: Autoria Própria

Solução	Criador	TTE (s)	Tamanho dos pedidos (KB)	Distância de Levenshtein (%)	Precisão Média (%)	
Vision API	Google	1.64	8667.2	34.05%	96.02%	0,82
OCR.Space API	A9t9 software GmbH	2.54	39033.8	11.61%	92.60%	0,68
Cloudmersive API	Cloudmersive, LLC	3.09	782.2	39.67%	95.90%	0,85
Azure Computer Vision API	Microsoft	3.13	530.2	40.44%	95.97%	0,85
IBM API	IBM	37.22	412.7	13.37%	95.83%	0,62
Textract	Amazon	26.65	978.0	40.82%	95.74%	0,73
Tesseract	Opensource	1.30	0	-	91.4%	

Ao analisar a tabela encontram-se alguns algoritmos que claramente estão mais avançados para a tarefa de identificação de texto. Uma solução com o Tesseract, o Azure Computer Vision API ou o Cloudmersive permitirá uma elevada precisão média no reconhecimento de caracteres, mantendo um tamanho dos pedidos baixo, métrica importante se tivermos em conta um conjunto vasto de utilizadores a fazer pedidos ao mesmo tempo.

4.2. Comparação de modelos e soluções de classificação de entidades

Como referido anteriormente, a tarefa de classificação de entidades implica o reconhecimento de determinados excertos de texto como um campo importante de um determinado documento. Para esta atividade, foram identificadas duas possibilidades: o desenvolvimento de um modelo de raiz, com recurso a SVM, uma abordagem de ML que funciona através da aprendizagem de um híper plano que separa os exemplos para a sua classificação, reconhecida pela sua boa performance, boa precisão e por não ser facilmente *over-fitted* (Ekbal & Bandyopadhyay, 2009) ou uma solução comercial em que, partindo já de uma modelo genérico pré-

treinado, permita a extração de campos de negócio, como é exemplo o *Azure Form Recognizer* (Microsoft, 2017), serviço da Microsoft que permite a extração de informações de formulários e imagens, recebendo ficheiros em *PDF* ou imagem e retornando os campos identificados com o respetivo grau de confiança desse reconhecimento. Uma vantagem de utilização do *Azure Form Recognizer* é a utilização adjacente do *Azure Computer Vision* API, solução de *OCR* que tinha sido demonstrada como uma das melhores na comparação anteriormente feita, permitindo um desenvolvimento mais simples e célere da prova de conceito.

Em ambos os casos, desenvolvendo um modelo de raiz ou utilizando uma solução comercial, é necessário ser efetuado treino no modelo. Este treino é essencial para que os campos de negócio sejam reconhecidos com sucesso, apesar de apresentar um dos maiores desafios da definição de um modelo de ML, sendo um trabalho maioritariamente manual e demorado. Para este treino é necessário recolher dados, fase descrita posteriormente neste documento, também um processo manual onde é necessário identificar documentos, dividir os dados em conjuntos de treino, teste e validação, extrair os textos dos documentos, classificar cada um dos campos presente no documento e executar o algoritmo de treino do modelo para melhorar a precisão da inferência do mesmo.

A classificação de cada um dos campos, processo comumente denominado de etiquetagem, com o objetivo de treinar o modelo, é feita através de algumas aplicações desenvolvidas especificamente para este efeito. Foram testadas várias soluções para etiquetagem, nomeadamente o *LabelBox* (Labelbox, 2021), *Computer Vision Annotation Tool* (CVAT) (OpenVinoToolKit, 2018) e *Form OCR Testing Tool* (Microsoft, 2021a). Despois de serem todas testadas, optou-se por utilizar na generalidade a solução da *Microsoft*, tirando partido do ecossistema em que está inserida, tendo uma fácil integração com o *Azure Form Recognizer*, tendo-se pontualmente também recorrido à ferramenta *LabelBox*.

Primeiramente são definidas as classes em que cada um dos blocos de texto pode ser classificado. Segundo o Artigo 226º da Secção 4 do Capítulo 3 do Título XI da Diretiva 2006/112/CE da União Europeia (Oficial et al., 2006), uma fatura deverá obrigatoriamente conter:

- Data de emissão da fatura;
- Número sequencial, que identifica univocamente uma fatura;
- NIF da entidade vendedora ou prestadora de serviço;
- NIF do adquirente ou destinatário;
- Nome e endereço da entidade vendedora ou prestadora de serviços;
- Nome e endereço do adquirente ou destinatário;
- A quantidade e natureza dos bens entregues ou serviços prestados;
- A data de entrega de bens ou prestação de serviços
- O valor tributável para cada taxa ou isenção, o preço unitário líquido de IVA e eventuais abatimentos;
- A taxa de IVA aplicável;
- O montante de IVA a pagar;
- Outros campos específicos em caso de regimes especiais.

A lista de campos apresentada serve então de base para as classes, visto serem campos obrigatórios nas faturas emitidas, tendo sido adicionados ao *Azure Form Recognizer.*

Existem, contudo, algumas limitações decorrentes da aplicação desta diretiva nos softwares de faturação já que, por exemplo, um consumidor que não tenha requerido os seus dados na fatura poderá encontrar o nome "Consumidor Final" no documento ou simplesmente não encontrar qualquer nome.

Um subconjunto de dados foi adicionado ao serviço de armazenamento da *Microsoft*, conectado com a ferramenta de etiquetagem (exibida na Figura 10), juntando-se à lista de campos para o processo de treino.

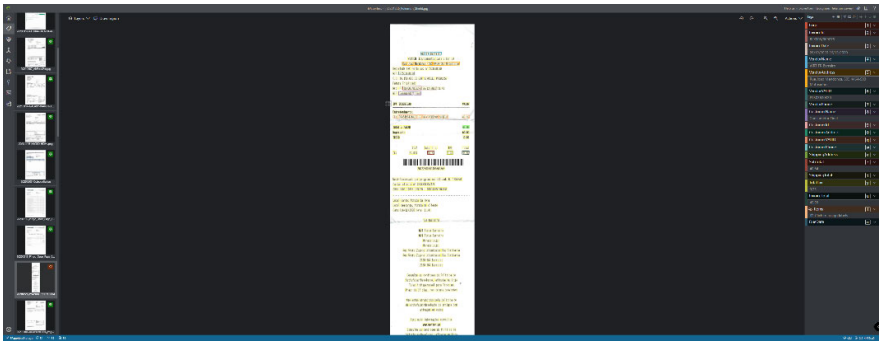

Figura 10 - Interface gráfica do Form OCR Testing Tool
Fonte: Autoria Própria

Numa primeira fase da prova de conceito, nem todos os campos terão a mesma importância, contudo, num treino inicial todos os campos são etiquetados, agilizando o trabalho para futuros desenvolvimentos.

A ferramenta começa por reconhecer todo o texto presente no documento, com recurso à própria solução de OCR. De seguida o operador que está a efetuar o reconhecimento carrega no texto que pretende classificar e escolhe a que classe pertence este texto. Após finalizar todas as classificações, a plataforma treina, de forma autónoma, um modelo que fica posteriormente disponível para inferência.

É disponibilizado um serviço, denominado *Custom Form*, que recebe o ID do modelo gerado do treino anteriormente feito, retorna os textos na página, as caixas de localização dos mesmos, a confiança de inferência dos textos e as classificações e as mesmas classificações. As operações deste serviço permitem a customização de modelos, treino dos mesmos, análise de um formulário e receber o resultado da análise. As interações entre o operador, a ferramenta de treino OCR, o *Form Recognizer* e o POC são descritas na Figura 11.

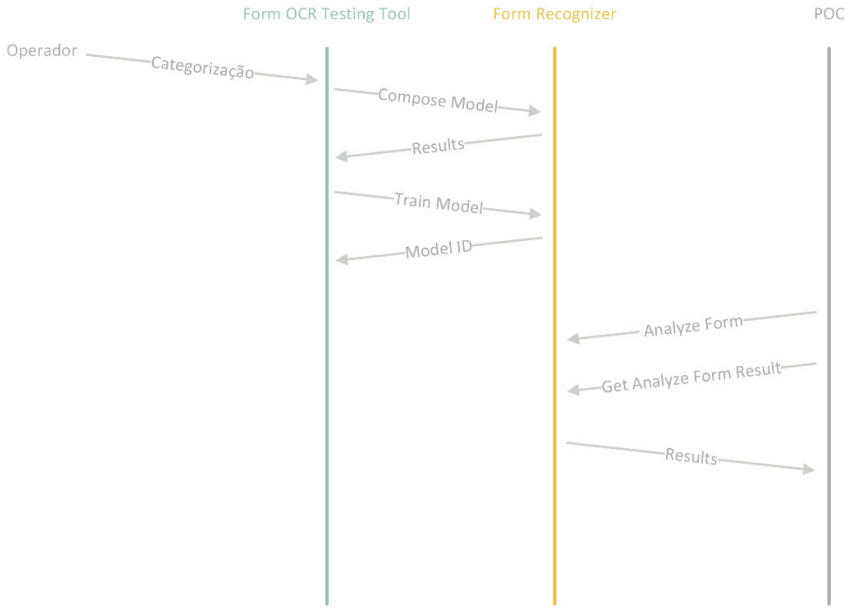

Figura 11 - Interações entre o operador de ML, o Form OCR Testing Tool, Form Recognizer e a prova de conceito
Fonte: Autoria Própria

Os resultados devolvidos nem sempre serão totalmente corretos, apresentando um desafio claro dado que o conteúdo de faturas tem de estar completamente correto para ser comunicado às entidades reguladoras. Existirá sempre uma validação manual do utilizador antes de um documento ter o seu processamento encerrado, aumentando o grau de confiança em todo o processo. Todas as alterações que o utilizador faz aos dados processados ficam guardadas para posteriormente serem utilizadas para treino de novas gerações do modelo de reconhecimento. O reconhecimento por *templates*, descrito em maior detalhe posteriormente, permite também a alimentação de novos treinos do modelo. Todo este processo, usualmente chamado de Feedback Loop, permite que a interação do utilizador do sistema tenha impacto no mesmo (Khritankov, 2021) e que o modelo possa aprender também pela interação do utilizador com os resultados das suas inferências. Este processo, que corresponde à utilização do resultado de determinado sistema de inteligência artificial em conjunção com o resultado que um utilizador atribui à mesma tarefa, de forma a re-treinar e melhorar um modelo de ML (Nord, 2013), garante que o modelo irá sempre evoluindo com os seus utilizadores.

4.3. Levantamento de Requisitos

O desenvolvimento de uma solução como a descrita neste documento partiu de uma necessidade identificado por uma empresa parceira da área da transformação digital aplicada ao setor contabilístico. Esta necessidade foi apresentada numa reunião onde foi levantada a ideia de uma aplicação que, com recurso a *data science,* processasse documentos como faturas e notas de crédito, de modo a agilizar o trabalho do dia a dia da empresa parceira.

É com base nesta descrição que se começou o processo de levantamento de requisitos e de desenho, sendo esta empresa parceira tida como *stakeholder* no decorrer do projeto, tomando o papel de especialista na área de negócio a modelar.

Juntamente com o departamento financeiro, este *quórum* foi essencial para a definição da solução a desenvolver, permitindo a captura de necessidades e elaboração de requisitos que respondessem a estes.

De forma a clarificar os eixos a abordar na análise, e tendo já em mente a necessidade desta solução se tornar num produto viável, a primeira tarefa consistiu no preenchimento de um *business canvas model (bcm)*. Esta ferramenta permite criar um modelo de negócio que descreve o raciocínio de como uma organização pretende criar, entregar e captar valor (Osterwalder & Pigneur, 2010). Esta ferramenta tem sido adotada no setor do empreendedorismo, já que permite uma visibilidade de vários aspetos essenciais quando se cria um produto ou serviço.

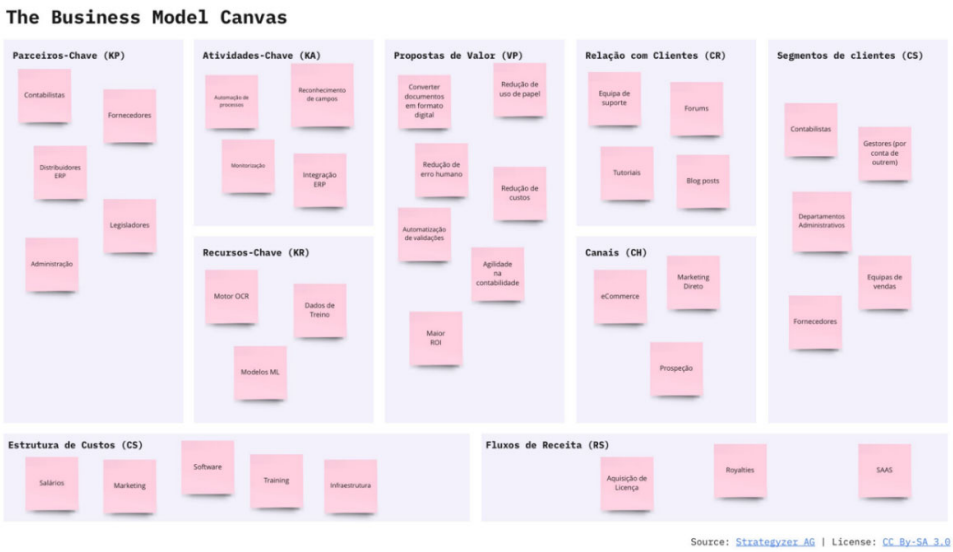

Figura 12 - Business Model Canvas
Fonte: Autoria Própria, adaptado de (Strategyzer, 2022)

Como de pode ver na Figura 12, este modelo de negócio é composto por 9 elementos, sendo que cada um destes caracteriza o negócio numa vertente diferente.

Este modelo pode ser aplicado na construção de produtos, serviços ou mesmo na definição de uma organização como entidade. No contexto desta solução, o *business model canvas* será aplicado à construção do produto que resultará do desenvolvimento da solução assente na prova de conceito descrita no ponto seguinte deste documento.

A secção das relações com o cliente, na imagem identificado como CS, diz respeito a que grupos a solução pretende alcançar ou servir. Todos os componentes do BCM estão interligados entre si, sendo o CS diretamente afetado pelo CH e CR. Este primeiro diz respeito a que canais serão utilizados para interagir com os clientes de modo a lhes entrega a proposta de valor VP, já o CR descreve que tipos de relações a solução irá estabelecer com os diferentes CS.

No centro deste modelo aparece o anteriormente falado VP, que engloba todas as características da solução que irão trazer valor para o cliente. O RS explicita de que maneira os diferentes CS irão gerar ganho financeiro para a solução e o CS representa todos os custos que a equipa irá ter de suportar para desenvolver e operar a solução.

São ainda descritos neste documento os KR, que listam os ativos mais importantes para implementar o modelo de negócio, as KA que representam as tarefas fundamentais que necessitam de ser feitas para que o modelo de negócio se implementado e os KP, que são uma descrição da rede de parceiros e fornecedores que serão essenciais para a execução do modelo.

Com base no BMC e nos seus componentes, surgiu a necessidade de reunir também com o departamento administrativo e financeiro (DAF) pois este tem uma visão mais alargadas dos processos de contabilidade e permitiu considerar mais casos de uso e potenciais clientes.

Foram inquiridos vários *stakeholders* que representam cada um dos grupos representamos na secção CS. Estas reuniões permitiram o levantamento dos casos de uso e requisitos funcionais da solução a ser desenvolvida.

4.3.1. Definição de perfis

Foram identificados três grupos de utilizadores:

- Utilizadores empresariais
- Utilizadores individuais
- Equipa de suporte/Administradores

Os utilizadores da equipa de suporte auxiliam outros integrantes da solução dando-lhes suporte aplicacional e, dado esse facto, necessitam de permissões especiais de gestão de licenciamento, definições e contas. Os utilizadores individuais representam qualquer utilizador sem conexão a outras entidades registadas que pretende que os seus dados sejam tratados como pessoais, enquanto os utilizadores empresariais têm sempre relações entre a sua conta e outras contas de empresa.

Dentro do próprio grupo de utilizadores empresariais, são compreendidas várias categorias de perfis:

- Direção (administrator)
- Gestores
- Administrativo

- Temporário

Estes perfis representam necessidades diferentes no que diz respeito a funcionalidades no sistema e, por isso, têm também permissões distintas uns dos outros.

Após as reuniões iniciais de definição de necessidade, foram definidos casos de uso para cada um dos *stakeholders* e de seguida foram definidos quatorze módulos funcionais aos quais os requisitos funcionais pertencem. Os requisitos funcionais são requisitos que especificam que funções o sistema (ou componentes do mesmo) necessitarão de executar (IEEE, 1990).

O agrupamento dos requisitos funcionais em módulos permite uma maior organização tanto na gestão como no planeamento do projeto.

Na Tabela 3 é possível consultar a lista de módulos nos quais os requisitos funcionais foram divididos.

Tabela 3 - Módulos de requisitos
Fonte: Autoria Própria

Identificador	Nome	Descrição
TM	Templates	Este módulo apresenta as funcionalidades relacionadas com a criação e gestão de modelos de documentos na aplicação.
AD	Administração	Área relacionada com a gestão da aplicação como um todo, principalmente no que diz respeito a gestão de acessos, bem como na definição de componentes transversais.
PE	Personalização	Este módulo engloba todos os requisitos relacionados com a personalização da experiência de utilizador de cada indivíduo ou empresa.
AU	Auditoria	Neste módulo encontram-se os requisitos que permitem a entidades externas acreditadas e a entidades internas consultar os registos de ações na plataforma.
MI	Melhoramento de imagem	Este módulo é responsável por assegurar o pré-processamento de

		imagens, de modo a ter melhores resultados no processo de ML.
IO	Importar e Exportar	Este módulo é referente aos requisitos que englobem carregamentos e descarregamentos de ficheiros e dados de e para a plataforma.
CA	Captura	Este módulo engloba os requisitos relacionados com a captura de documentos em tempo real, sendo assim uma especificidade do módulo de "Importar e Exportar".
AT	Autenticação	A Autenticação é o modulo que lida com a identificação e entrada do utilizador dentro do ambiente protegido da aplicação.
NO	Notificações	Este módulo é responsável pela gestão de comunicações com os vários utilizadores da plataforma.
PR	Processamento	Este é o principal módulo da solução, já que representa a capacidade de um documento ser reconhecido e tratado pela mesma. É neste módulo que assentam os requisitos relativos a ML e DS.
IN	Integração	Este módulo apresenta os requisitos relacionados com a integração com sistemas satélite.
GD	Gestão Documental	Este módulo engloba os requisitos de gestão de documentos, pesquisa dos mesmos e segragação por autorização.
UT	Utilizador	Este módulo lida com a informação pessoal do utilizador e a sua identidade dentro da aplica
AR	Armazenamento	Este módulo, fortemente ligado com o GD, contém os requisitos necessários para a persistência de dados, tanto do lado do cliente, como do lado do servidor.

Os requisitos foram por sua vez alvo de uma cuidada análise e consequente prioritização. A prioritização dos requisitos funcionais foi feita com base no método MoSCoW (Ahmad et al., 2017) que permite atribuir uma prioridade aos requisitos que é compreendida entre "*Won't Have This Time (WH)*", "*Could Have (CH)"*," Should *Have (SH)*", "*Must Have (MH)"* onde WH é a prioridade menos urgente e MH a mais urgente. Este tipo de prioridade, juntamente com o agrupamento em

módulos, permite agilidade e rapidez na construção de um plano de projeto que vá de encontro com as necessidades dos utilizadores. Conforme (Ahmad et al., 2017), os requisitos MW representam exigências que necessitam de ser incluídas na versão final da solução, os SH são requisitos de alta prioridade que devem ser incluídos se possível, considerando o tempo disponível, os CW são requisitos que podem ser adicionados, desde que não afetem em grande medida o custo ou esforço e os WH são requisitos que os *stakeholders* concordam que não estarão de todo na versão atual da solução.

No processo de levantamento de requisitos funcionais e no desenho da solução a implementar, foi necessário também definir Requisitos Não-Funcionais (*Non-Funcional Requirement, NFR*). Apesar de não existir um consenso no que diz respeito a uma definição única do que representa um NFR (Glinz, 2007) mas como base de trabalho foi adotada a definição de que um NFR é um atributo ou um constrangimento do sistema a desenvolver (Chung & do Prado Leite, 2009). Outras definições mencionam que um NFR representa uma Qualidade do Sistema. A definição adotada, juntamente com o facto anteriormente mencionado explicam a adoção do *standard* de qualidade de *software* ISO/IEC 25010:2011 (International organization for standardization, 2011), uma definição estandardizada de qualidade de um produto de software. Os requisitos desta norma assentam em oito eixos distintos: funcionalidade, desempenho, compatibilidade, usabilidade, confiabilidade, segurança, manutenibilidade e portabilidade. Estes eixos permitem a definição de restrições ao desenvolvimento bem como a definição de atributos de qualidade que, numa fase posterior do desenvolvimento irão servir de base para o desenho dos casos de teste a realizar, partindo assim os NFR dos requisitos funcionais (de que ações o sistema necessita de proporcionar) e restringindo as opções a tomar na fase de desenho e modelação, etapa esta descrita no ponto 4.4 deste documento.

	Administrador							Director							Manager							Administrativo							Individual						
	C	R	U	D	P	V	S	C	R	U	D	P	V	S	C	R	U	D	P	V	S	C	R	U	D	P	V	S	C	R	U	D	P	V	S
Documentos	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	
Tipos de documentos	X	X	X	X			X		X							X							X						X	X	X	X			
Templates	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X			X	X		X	X	X	X	X	X	
Regras de Documentos	X	X	X	X			X		X							X							X												
Grupos Regras	X	X	X	X			X		X							X							X												
Perfis	X	X	X	X				X	X	X	X				X	X	X	X													X	X	X			
Auditoria																																			
Mensagens	X	X	X	X				X	X	X	X				X	X	X	X				X	X	X	X				X	X	X	X			
Notificações	X	X	X	X				X	X	X	X				X	X	X	X				X	X	X	X				X	X	X	X			
Temas	X	X	X	X			X		X							X							X						X	X	X	X			X
Empresa		X	X	X					X							X							X												
Departamentos	X	X	X	X		X		X	X	X																									
Equipas	X	X	X	X		X		X	X	X	X		X		X	X	X																		

Figura 13 - Matriz de permissões
Fonte: Autoria Própria

Na

Figura 13 é demonstrada a Matriz de Permissões da aplicação, que permite uma visualização de que acessos cada tipo de utilizador possui e que operações pode executar. É possível ainda visualizar na figura que cada perfil apresenta permissões que são representadas pelas suas iniciais, nomeadamente: (C) Create, (R) Read, (U) Update, (D) Delete, (P) Process, (V) Validate e (S) Share. No caso das quatro primeiras permissões, estas são tradicionalmente utilizadas para denominar operações de escrita, leitura, atualização e eliminação em sistemas informáticos. As restantes permissões são específicas ao sistema a desenvolver, onde é necessário que os utilizadores possam processar os documentos e templates, validar os mesmos e partilhar alguns conteúdos dentro do sistema.

Após o levantamento inicial dos processos a responder durante o desenvolvimento da solução, de que funcionalidades é que seriam necessárias para satisfazer as necessidades dos *stakeholders* e que características é que a solução teria de possuir para garantir os seus requisitos e qualidades, deu-se início à fase de desenho e modelação, fase esta onde é descrita em detalhe a visão da equipa técnica para o desenvolvimento. Esta fase está descrita no próximo capítulo.

4.4. Desenho e modelação

Após o processo de analise inicial, existe necessidade de definir tecnicamente a abordagem a utilizar, de modo a permitir à equipa de desenvolvimento ter uma visão geral da abordagem a implementar.

Esta análise é executada numa etapa de desenho e modelação, onde se especificam abordagem e decisões no que diz respeito a arquitetura de soluções, componentes da mesma, abordagens de segurança e autenticação, infraestrutura, modelos de dados, recomendações de implementação, protótipos dos ecrãs a desenvolver, entre outros tópicos pertinentes.

4.4.1. Definição de arquitetura

Com base nos requisitos da solução, tanto os funcionais com os não-funcionais, define-se uma arquitetura física e lógica que responda a estes e que obedeça às boas práticas da programação.

Optou-se por uma arquitetura híbrida baseada em microserviços. Este microserviços são serviços pequenos e autónomos que trabalham em conjunto. (Newman, 2015a). Os microserviços seguem os mesmos princípios da Programação Orientada por Objetos, mas aplicada a cada serviço independente. São definidas as fronteiras e cada serviço é responsável pelas suas operações, dados e lógica. Segundo (Newman, 2015) as principais vantagens de uma arquitetura orientada a serviços passam: pela capacidade de ter várias linguagens de programação diferentes para diferentes módulos da aplicação, utilizando a linguagem que mais se aplica para cada função; resiliência, já que quando um componente falha não irá derrubar toda a solução, ao contrário de um sistema tradicional, monolítico, onde quando uma parte do sistema falha, todo o sistema falha; escalabilidade, podendo expandir-se só parte dos serviços e não a solução toda, tornando-se uma abordagem mais viável a nível financeiro e flexível a nível de *deployment,* já que quando existe uma grande procura por determinado serviço e este não está a dar resposta, pode clonar-se a instância do serviço singular e duplica-se a capacidade de resposta; facilidade de instalação, pois cada serviço pode ser instalado individualmente, ao contrário de um monólito, que necessita de ser todo construído e implantado, mesmo que tenha sido feita uma pequena alteração; adequação a desenvolvimento por várias equipas, sendo cada serviço visto como independente e pode ser construído por várias equipas paralelamente.

Optou-se pela abordagem híbrida tendo em conta as limitações de recursos disponíveis para o desenvolvimento e *deployment*, complexidade e pelo facto de

existir a possibilidade de no futuro efetuar as alterações necessárias para responder às características de uma arquitetura completa de microserviços.

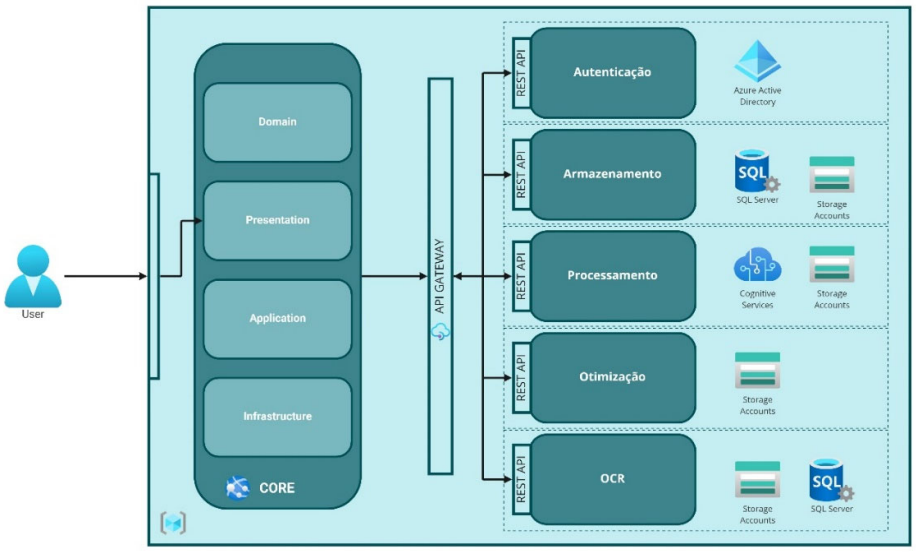

Figura 14 - Arquitetura da solução
Fonte: Autoria Própria

Esta arquitetura é composta por vários componentes que trabalham em conjunto para realizar as tarefas pedidas pelo utilizador, como pode ser visto na Figura 14. O utilizador acede diretamente ao *Frontend*, a componente de visualização da solução. Esta componente, por sua vez comunica com a *Core App*, componente responsável pela maior parte da lógica de negócio da aplicação. A Core App comunica diretamente com uma *Application Gateway,* uma opção para roteamento de pedidos conforme endereço para o serviço que é requerido. Este *Gateway* comunica com vários serviços, cada um deles implementado como um módulo coeso, que pode ser substituído por outro que disponha as mesmas interfaces.

O módulo de Autenticação trata das operações relacionadas com a identificação e autorizações dos utilizadores da plataforma. O Armazenamento ocupa-se das funções relacionadas com armazenamento de ficheiros, já que é utiliza uma *Storage Account*, serviço do *Azure*, para armazenamento de ficheiros na *cloud*. O módulo de Processamento engloba as operações relacionadas com o processamento de ficheiros, utilizando inteligência artificial ou o sistema de *templates*. O módulo de Otimização abrange as funcionalidades relacionadas com

a conversão de imagens e a sua otimização, tratando assim também do pré-processamento de ficheiros em imagem. Finalmente, o módulo de OCR diz respeito às operações que implicam a execução de reconhecimento de caracteres. A divisão destes módulos em unidades independentes e autossuficientes, permite que tenham sido desenvolvidos em várias tecnologias, nomeadamente, .Net 6, NodeJS e Python. A escolha das tecnologias prendeu-se com a adequação da mesma à tarefa a executar, tendo sido essas decisões tomadas no âmbito da fase de modelação e desenho. Sendo a autenticação e o armazenamento desenvolvidos utilizando a *framework* da *Microsoft*, os módulos de otimização e OCR utilizando NodeJS e Processamento utilizando Python. De notar que nem sempre é linear a divisão dos módulos aplicacionais em microserviços. Aliás, é de extrema importância um estudo cuidado limites de fronteira de cada serviço.

Cada um dos microserviços foi desenhado tendo em conta o princípio da Clean Architecture (Martin, 2017), separando as responsabilidades de infraestrutura, lógica de negócio, domínio do problema e visualização. Cada microserviços segue ainda o padrão de desenho CQRS, um padrão designado de "responsabilidade de consulta e comando", que permite a definição de uma interface comum para comunicação entre serviços e separação de responsabilidades de escrita e leitura, permitindo melhor performance nestas operações.

Todos os componentes da solução foram implementados e desenhados tendo em conta o seu *deployment* na nuvem. Esta escolha pela utilização de uma abordagem *off-premises*, que acompanha a tendência do mercado de *software* em computação na nuvem, prende-se pelo facto da utilização dos serviços do Azure, na modalidade de Infrastructure-As-A-Service (IaaS), Platform-As-A-Service (PaaS) e Software-As-A-Service (SaaS) apresentar vantagens claras quando comparadas com a utilização de alternativas ditas clássicas. A comparação entre estes níveis de serviço pode ser consultada na Figura 15. Segundo (Apostu et al., 2013), IaaS apresenta-se como a disponibilização virtual de poder computacional e memória, PaaS a disponibilização de ambientes de execução, como é o exemplo de servidores aplicacionais e de base de dados e SaaS disponibiliza um *software* pronto a utilizar, geralmente através de um *browser*.

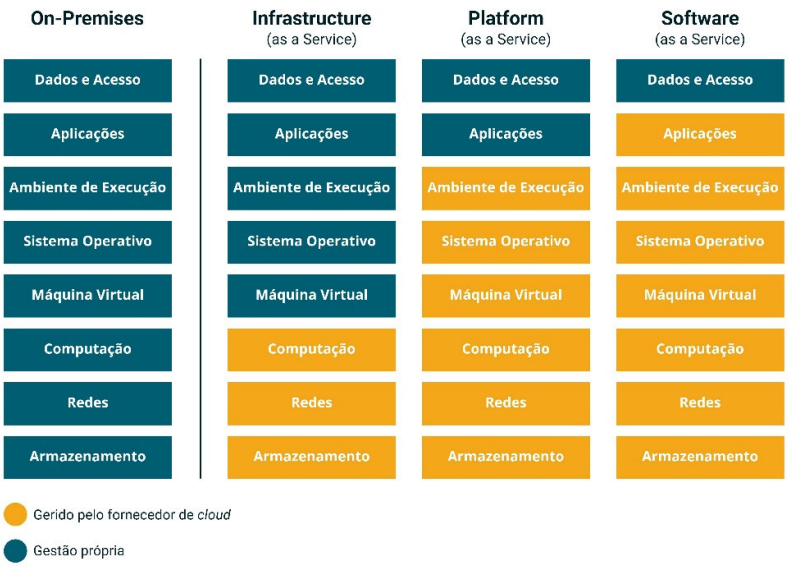

Figura 15 - Comparação dos diferentes níveis de serviços do IaaS, PaaS e SaaS.
Fonte: Adaptado de (Wang, 2017).

O PaaS, extensivamente utilizado na solução descrita neste documento (como por exemplo em cada um dos serviços da solução), permite um aumento de produtividade, permite uma melhor organização e reduz custos de *software*, eliminando a necessidade dos programadores terem de configurar os próprios servidores para executar os seus desenvolvimentos, escalar os mesmos ou implementar e integrar ferramentas de gestão (Lawton, 2008). Esta abordagem foi especialmente benéfica por permitir dar como garantida a configuração de servidores, execução de atualizações de segurança e gestão de incidentes, já que toda essa gestão é feita pela *Azure*.

Como descrito anteriormente, cada um dos módulos da aplicação foi implantado num *App Service*, uma ferramenta da *Azure* que permite desenvolver e hospedar aplicações *web*, *backends mobile* e APIs *RESTful*, na linguagem mais conveniente para o programador, sem este ter de se preocupar com a infraestrutura (Microsoft, 2022). Foi ainda utilizado, para hospedagem do serviço de ML os *Cognitive Services* da *Microsoft*, produto este abordado no capítulo 4.5.5 do presente documento.

4.4.2. Segurança

A solução a desenvolver lida com documentos de natureza sensível, tornando-se imperativo que seja definida uma estratégia de segurança que assegure a privacidade dos dados e documentos presentes na plataforma. No que diz respeito à segurança, autenticação, autorização e *logging* foram os pontos que foram definidos na fase de desenho. A autorização, referida no ponto 4.3.1 do presente documento, diz respeito a que ações determinados tipos de utilizador podem fazer sobre cada tipo de entidade dentro da plataforma, bem como a que páginas e documentos podem aceder.

A autenticação é a definição de que abordagem será utilizada para garantir a identidade do utilizador a aceder à solução. Optou-se por uma estratégia que permita aos utilizadores identificarem-se ou com o seu email e password ou através de um provedor externo de identidade, posteriormente mapeado também ao email do utilizador, permitindo ao utilizador utilizar várias maneiras de se identificar.

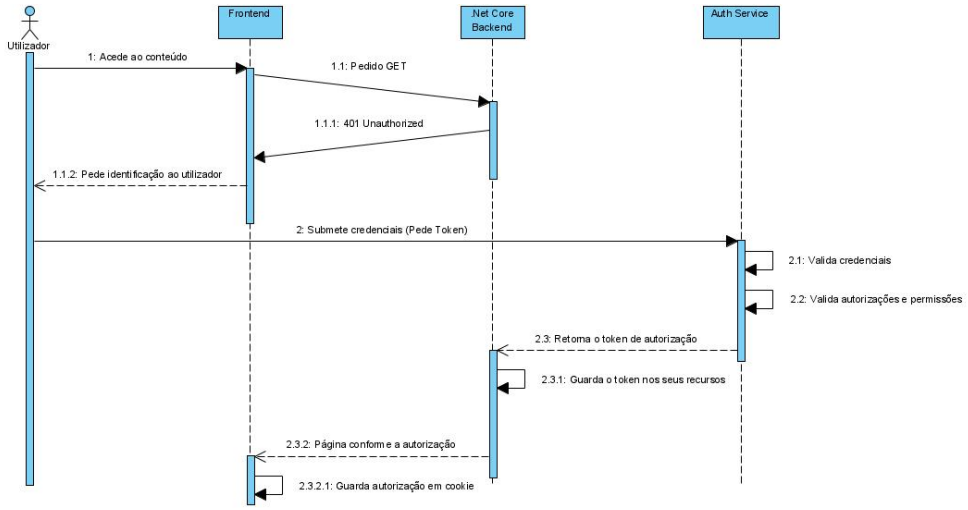

Figura 16 - Diagramas de sequência de autenticação
Fonte: Autoria Própria

O diagrama presente na Figura 16 permite uma visão da relação entre o utilizador e os diferentes componentes da solução que permitem a sua identificação. No caso de um fornecedor de identificação externo, como o caso do LinkedIn, Facebook ou Google, a interação é a representada na figura, sendo que quando são

utilizadas contas com *email* e *password*, o *backend* executa também o papel de serviço de autenticação.

Este esquema de autenticação é utilizado sempre que o utilizador pretende aceder a conteúdo que é declarado com um acesso "não-anónimo". Se o utilizador tentar aceder a um conteúdo a que não tem acesso por não estar identificado, este é redirecionado para a página que lhe permite efetuar essa autenticação. Caso o utilizador tente aceder a um conteúdo ao qual não tem acesso por questões de autorização, é retornada uma página de erro que lhe informa que não tem acesso aquele recurso, sendo retornado juntamente um código *HTTP* 401, que evidencia esse acesso não autorizado.

Cada utilizador, com base nos perfis descritos anteriormente neste documento, está limitado às permissões do seu perfil e, caso tente executar uma ação fora desse âmbito, recebe como resposta também o erro 401.

É assegurada a rastreabilidade das tarefas realizadas pelos utilizadores, para fins de auditoria, controlo de qualidade ou suporte, ficando estes registos registados em forma de *logs* aplicacionais.

Estes registos são guardados em ficheiros de texto no servidor aplicacional e contêm registos das operações, data e hora, utilizador que as executou e uma categorização da ação, como por exemplo *error* (registar erros)*, info* (registar operações ou informações) *e warn* (para registar avisos). Estes *logs* permitem, por exemplo, que fique registada uma ação que determinado utilizador pretenda fazer e que tenha resultado num código de erro, sendo que um operador de suporte poderá aceder a este ficheiro e aferir qual a razão para este erro. Optou-se ainda por padronizar as mensagens a serem escritas para permitirem um melhor contexto de cada operação.

4.4.3. Prototipagem

Na fase de modelação, é necessário elaborar propostas de visualização para os vários ecrãs da solução, permitindo uma maior consistência durante o desenvolvimento, retirando da equipa de desenvolvimento a tarefa de modelar a página que vai desenvolver. Nesta fase, as propostas de visualização têm um baixo

grau de fiabilidade, quando comparados com o futuro produto final, e são comumente chamados de *wireframes*. Estes protótipos são uma técnica de prototipagem ágil para rascunhar o "esqueleto" da estrutura de uma página *Web* (de Lange et al., 2020).

No caso da solução a desenvolver, foram desenvolvidos protótipos de alta-fidelidade, que se assemelham ao produto final, exibindo já as cores, tipografia e ícones que foram posteriormente utilizados no desenvolvimento.

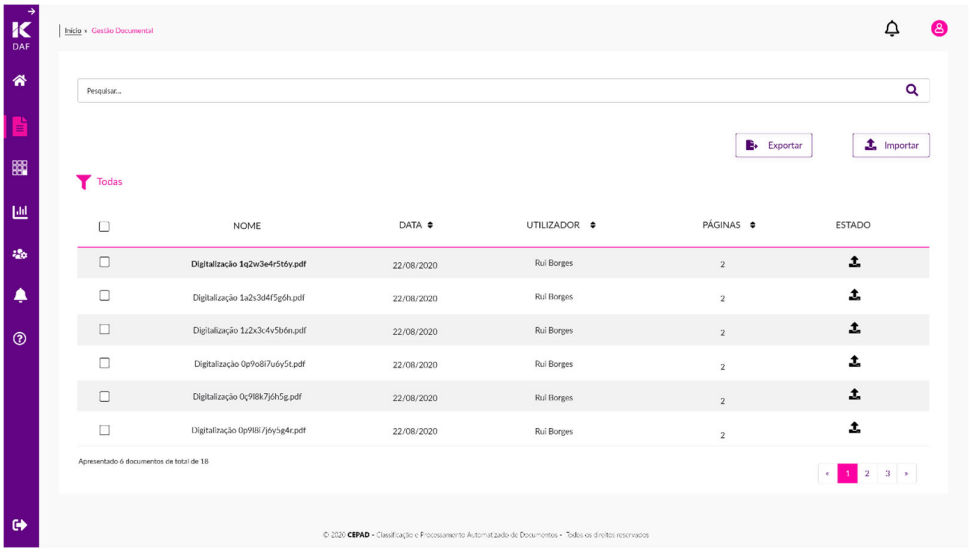

Figura 17 - Protótipo de alta-fidelidade da solução
Fonte: Autoria Própria

Na Figura 17 é possível ver um exemplo de protótipo desenvolvido aquando da fase de desenho. De notar que o processo de especificação de requisitos e desenho é de natureza iterativa, tendo em conta a metodologia utilizada no decorrer deste projeto, logo os artefactos resultantes da primeira fase de especificação foram sofrendo consecutivas alterações do decorrer do projeto.

Finda uma primeira fase de desenho e tendo os artefactos necessários para começar o desenvolvimento, começou-se a fase de desenvolvimento da prova de conceitos, fase esta descrita no ponto 4.5 deste relatório.

4.5. Desenvolvimento da prova de conceito

A solução desenvolvida, como descrito anteriormente consiste numa aplicação central (*core*), com o seu back-end em .NET , na sua versão 6, o seu front-end desenvolvido em HTML, CSS e JavaScript com utilização de bibliotecas como o *JQuery* e *Bootstrap* e base de dados SQL Server. Foram ainda desenvolvidos serviços em NodeJs e Python. Cada um dos componentes desta solução será abordado mais detalhadamente posteriormente neste documento. A solução utiliza ainda a infraestrutura *Azure*, como descrito no capítulo 4.4 - Desenho e modelação do presente documento, para as suas versões de desenvolvimento, testes e produção. A Figura 18 mostra uma representação da arquitetura implementada para a solução. Foram criados grupos de recursos, uma agrupação organizacional dentro do Azure, para cada um dos ambientes.

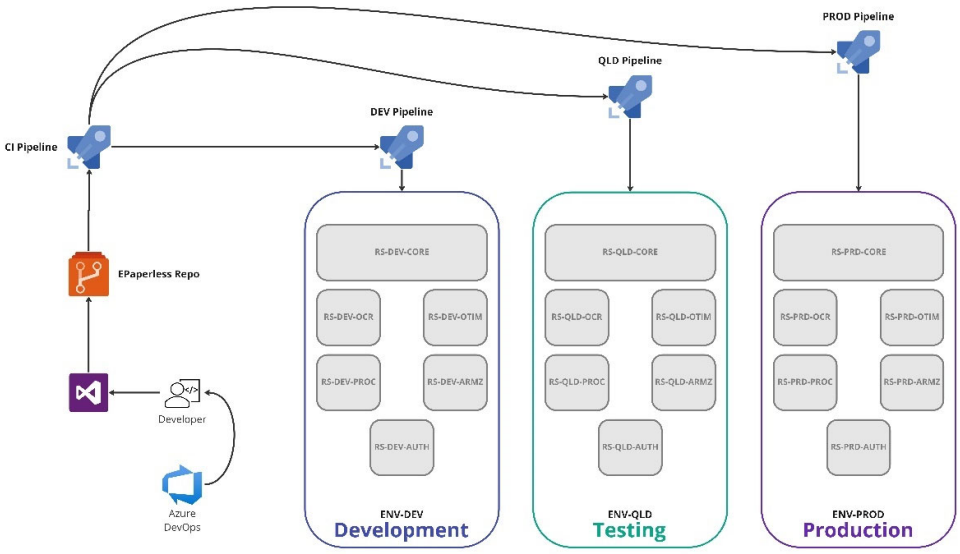

Figura 18 - Arquitetura geral da solução
Fonte: Autoria Própria

A utilização de três ambientes distintos permite aos programadores desenvolverem a solução sobre uma base estável mas que está sujeita a alterações constantes (desenvolvimento), permite a uma equipa de qualidade a validação de novas funcionalidades ou alterações num ambientes mais estável que o de desenvolvimento e que sofre menos alterações (testes ou qualidade) e permite um

ambiente onde o cliente final pode aceder para utilizar a aplicação, onde todo o conteúdo já foi testado e aprovado previamente (produção).

A base para a gestão de trabalho do desenvolvimento foi o Azure DevOps, plataforma essa que permite registar tarefas, histórias de utilizador, descrições de funcionalidades, entre outros.

No processo de desenvolvimento, o programador escolhe uma tarefa a completar, cria um *branch* num repositório Azure DevOps, referente à tarefa que vai executar e desenvolve-a. Quando termina o seu desenvolvimento, é feito um *Pull Request* para juntar as suas atualizações para um *branch* com uma versão estável de desenvolvimento (*development),* como exibido na Figura 19.

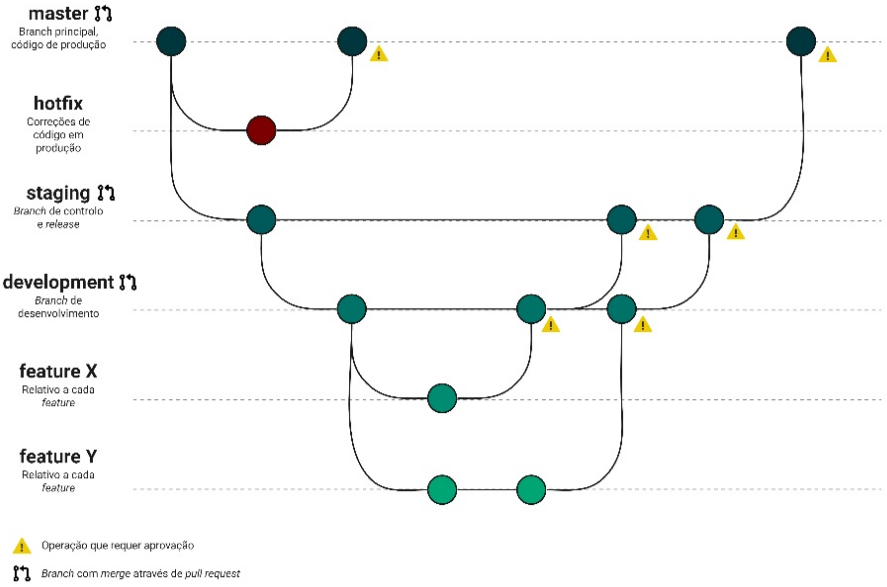

Figura 19 - Estratégia de controlo de código
Fonte: Autoria Própria

Cada programador conhece a estratégia a seguir com base num diagrama que foi elaborado aquando do desenho inicial da solução.

Depois da aprovação das alterações, é criada uma *build* que contém a versão mais atualizada do código. Cada versão do código é submetida a um controlo automático de qualidade de código, para garantir que todo o código que é passado para o ambiente de desenvolvimento segue as mesmas nomenclaturas e padrões de qualidade. Esta passagem de código regular para um repositório central em

passagens de porções pequenas permite que exista uma maior transparência da qualidade de código (Vadapalli & Safari, 2017), sendo um princípio base da integração contínua (*continuous integration*, CI). Algumas outras características da CI que foram utilizadas incluem a automatização de *builds*, implementações automáticas em ambientes de pré-produção, verificação regular de código e automatização de testes.

Quando existe uma versão sólida e validada do código, e depois do processo de *build*, esta versão é *deployed* automaticamente em ambiente de desenvolvimento, como referenciado anteriormente. No final de cada ciclo de desenvolvimento, a versão mais recente do ambiente de desenvolvimento foi transferida para o ambiente de *staging (*ou qualidade*).*

Cada entregável principal da aplicação é passado automaticamente para produção quando os testes do ambiente de qualidade são aprovados. Todo este processo chama-se entrega contínua e permite otimizar o tempo que se despende em tarefas operacionais. A automatização deste processo de CI e entrega contínua definem a Implementação contínua (continuous deployment, CD) (Vadapalli & Safari, 2017). A utilização do CI/CD neste projeto permitiu otimizar a gestão de versões, testes e entregas do mesmo. Faz ainda parte da estratégia de controlo de código um *branch* referente ao código mais estável, que deve ser uma cópia do código que está em produção, o *master* e um *branch* onde se corrigem os defeitos encontrados em produção, o *hotfix*.

Cada ambiente aplicacional tem uma base comum, os componentes presentes na Figura 14. As alterações presentes entre cada ambiente prendem-se com o poder computacional destes e os mecanismos de controlo e *logging* ativos. Se no caso de um ambiente de desenvolvimento faz sentido registar informações que possam ajudar os programadores a solucionar erros, num ambiente produtivo estes registos irão criar ruído e ocupar espaço desnecessário e que pode ser dispendioso.

Para reconhecimento de documentos é necessário seguir com conjunto passos que recebem como dado de entrada o dado de saída do passo anterior. O processo de reconhecimento de um documento começará sempre com o *input* do mesmo. Este documento poderá estar num formato de *pdf*, o mais usual, ou então estar em

formado de imagem. Para que se possa progredir para o passo seguinte, é necessário que exista uma uniformização de tipos de ficheiro. Existem alguns pontos a ter em conta nesta conversão de formatos: terá de se minimizar a perda de qualidade e informação aquando da conversão; os documentos pdf poderão ter várias páginas e terá de se agrupar as imagens de cada ficheiro para existir uma rastreabilidade. Depois de se ter uma imagem como input, é necessário tratar esta imagem para que se consigam obter melhores resultados na tarefa de OCR. O pré-processamento da imagem permite que se corrijam algumas características do *input* e consequentemente poderá ajudar a que se obtenham melhores resultados no reconhecimento de carateres, sendo este um processo extremamente importante, especialmente em imagens provenientes de câmeras digitais (Bieniecki et al., 2007). Existem vários métodos de proceder ao pré-processamento que serão abordados e comparados posteriormente neste documento, nomeadamente no ponto 4.5.4. Após o pré-processamento, o documento será então apresentado a uma solução de OCR que obterá uma representação do mesmo em texto, com o grau de confiança, blocos de texto, linhas, palavras, carateres e caixas de reconhecimento.

Obtido o texto da imagem, e com um grau de confiança aceitável, procede-se à identificação de entidades neste mesmo texto, através de NER, o processo que visa localizar, extrair e automaticamente classificar entidades nomeadas em classes ou tipos predefinidos em textos de domínio aberto e não estruturados, como por exemplo artigos de jornal. (Nadeau & Sekine, 2007; Shaalan, 2014)

Após o reconhecimento da informação contida no documento, é feita uma avaliação aos resultados. Se esta for negativa, poderá voltar-se a executar o processo alterando alguns parâmetros. Caso seja positiva, o documento é arquivado e o os dados nele contidos são exportados ou guardados numa base de dados. A Figura 20 exibe uma representação simplificada deste processo.

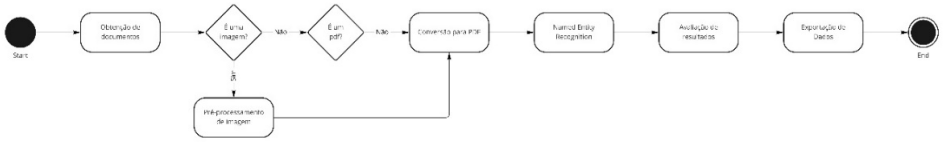

Figura 20 - Diagrama simplificado do processo de processamento de um documento.
Fonte: Autoria Própria

Nos subcapítulos seguintes são detalhadas as abordagens adotadas para cada um dos passos da anterior figura. Cada uma destas tarefas pode ser realizada por um serviço dedicado ou por um conjunto de serviços que trabalham em conjunto. Como referido no capítulo 4.4.1, referente ao desenho e divisão da solução por serviços independentes, esta abordagem permite aderir aos princípios da responsabilidade única e minimizar o acoplamento destes serviços. É importante um estudo cuidado, como referido anteriormente, dos limites dos serviços, mas também das comunicações que estes necessitam de estabelecer entre si, já que se duas funções de dois microserviços trocarem imensas mensagens entre si, provavelmente deveriam pertencer ao mesmo microserviços.

Como cada componente funciona como um módulo de software independente, é estruturado como tal, contando com controladores, responsáveis por processar os pedidos provenientes do utilizador ou de outros serviços, uma componente de lógica de negócios e de acesso a dados.

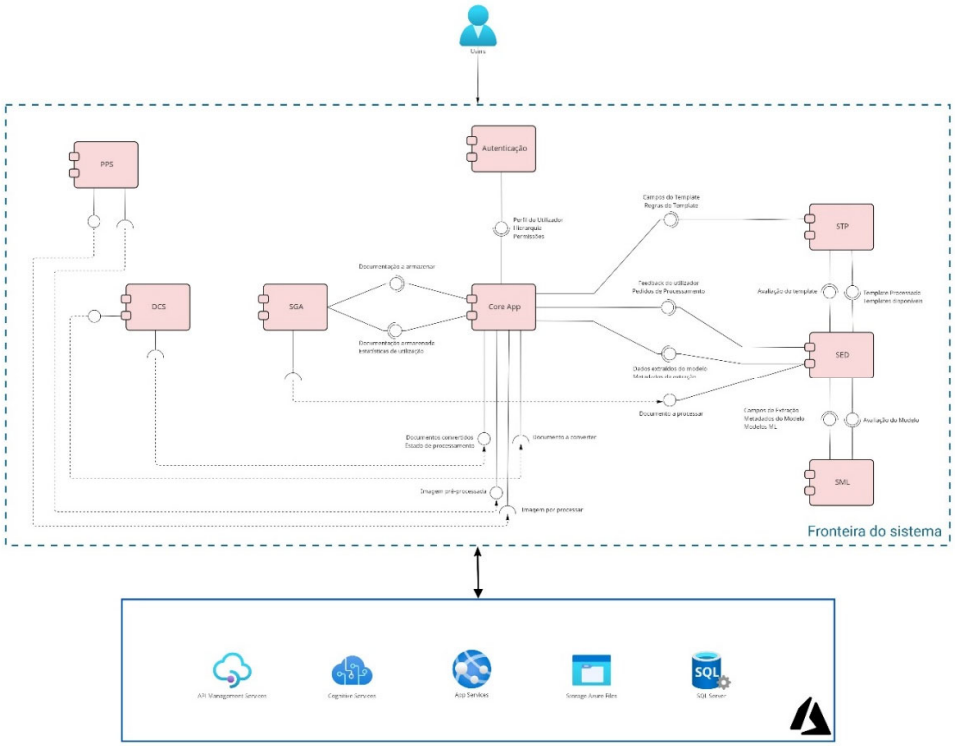

Figura 21 - Visão geral de componentes da solução
Fonte: Autoria Própria

Uma visão geral dos componentes criados estão podem ser consultados na Figura 21.

4.5.1. Autenticação

Dado que a solução a desenvolver foi concebida para ser utilizada por clientes do setor empresarial, a estratégia de autenticação e segurança de dados é essencial. É imperativo que se garanta o acesso dos dados a somente quem tiver a permissão explicita para os consultar. Como referido anteriormente no presente documento, a autenticação deste tipo de utilizador pode ser feita através de várias plataformas, contudo, a estratégia de autorização é independente da plataforma de autenticação, utilizando as permissões definidas anteriormente e descritas no ponto 4.3.1.

A estratégia de autoprização segue a abordagem de Role-Base Access Control (RBAC) que indica que as permissões são associadas a um determinado *Role* (Função) e que os utilizadores são inseridos nesta função, assim herdando as permissões destes (Sandhu, 1998). A esta abordagem de segurança é ainda

adicionado o contexto de empresa, assim permitindo a determinado utilizador, com determinada função, ter as ações correspondentes a esta função para operações que afetem somente o seu contexto empresarial.

Apesar da solução desenhada conter um ponto central, apresentou-se como opção mais lógica (principalmente tendo em conta o possível crescimento no futuro) a autenticação e validação dos dados de utilizador ser partilhada por todos os serviços, não contado somente com uma autenticação no ponto central.

Usualmente, quando se desenha um projeto com uma arquitetura orientada a microserviços, a autenticação começa com um pedido do utilizador para o servidor, que normalmente possui uma *API Gateway* que irá redirecionar as credenciais deste para um serviço de autenticação. Se as credenciais estiverem corretas, este serviço gera um *token* que é retornado ao cliente. A partir deste momento nos consequentes pedidos é só validada a validade do *token* e retornado o recurso se esta validação for efetuada com sucesso.

Para espelhar este comportamento, abordagem escolhida foi a utilização de JSON Web Token (JWT) para a autenticação entre serviços e com a componente *core,* recorrendo ao algoritmo de encriptação HS256. O JWT é um objeto *JSON* que é assinado digitalmente por uma entidade para este efeito e que representa reivindicações encriptadas a serem transferidas entre duas entidades (Jones et al., 2015).

O JWT contem o cabeçalho, o conteúdo e a assinatura. O cabeçalho contém usualmente metadados como a versão do protocolo e o algoritmo de encriptação utilizado no token, o conteúdo contém as reivindicações a ser partilhadas entre entidades, que podem ser por exemplo as informações do utilizador e as suas funções e a assinatura, que representa um valor gerado com base na encriptação dos valores contidos no cabeçalho e conteúdo. Ficou ainda definido que qualquer pedido onde não se conseguisse validar o *token* iria retornar o código HTTP 401, que representa uma ação não autorizada. Como é possível ver na Figura 21, existe um serviço de autenticação que é responsável por autenticar o utilizador também retorna os seus dados. Estes dados são retornados no conteúdo do *token* JWT.

4.5.2. Obtenção e armazenamento de documentos

Segundo a análise funcional, na solução desenvolvida, qualquer utilizador poderá carregar ficheiros. Se a conta de utilizador for individual somente ele poderá aceder aos ficheiros que carrega. Caso o utilizador tenha uma conta empresarial, todos os ficheiros são partilhados com base nas suas permissões. O utilizador acede à sua página de carregamento de ficheiros e pode arrastar ou selecionar um ficheiro em PDF ou um conjunto de imagens. O ficheiro é guardado em servidor, numa *storage account*, numa pasta isolada por utilizador e hierarquia. Optou-se por guardar o ficheiro em disco e não, por exemplo, como binário na base de dados, pela facilidade de leitura em disco e pela comparação de preço e performance.

Os ficheiros são guardados conforme o diagrama apresentado na Figura 22.

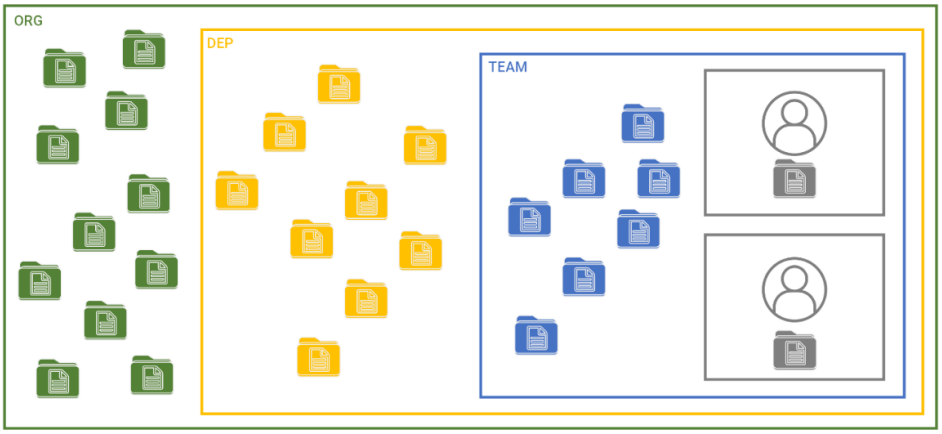

Figura 22 - Hierarquia de pastas do conteúdo dos utilizadores
Fonte: Autoria Própria

Este diagrama aplica-se aos Utilizadores Empresariais. Qualquer utilizador deste tipo está inserido dentro de uma estrutura hierárquica, onde o nível mais abrangente é o ORG, a empresa. Dentro de cada empresa existem vários departamentos (DEP) e cada departamento existem várias equipas (TEAM). Cada utilizador, por norma, estará assignado a uma equipa em específico, podendo, contudo, utilizadores ser adicionados a níveis hierárquicos superiores. Um CEO que queira utilizar a solução irá ser assignado á ORG, não a nenhum departamento ou equipa em específico. Cada carregamento de ficheiro, podendo vários ficheiros ser carregados ao mesmo tempo, gera um novo registo na base de dados correspondente ao carregamento, com as informações referentes a quem fez o upload, quando foi

efetuada a operação e o link para o ficheiro. O ficheiro é carregado para um serviço de *Blob* do *Azure* (Microsoft, 2021b), utilizando as capacidades deste serviço, nomeadamente a disponibilidade 24 horas por dia, 7 dias por semana e a utilização de *Content Delivery Network* (CDN) para distribuição dos ficheiros. A utilização de CDN, um sistema distribuído composto um largo número de nós espalhados pelo mundo, onde cada nó guarda uma réplica dos documentos mais acedidos (Cui et al., 2020), permite que os utilizadores tenham acesso mais rápido aos seus ficheiros, independentemente de onde estejam a cada momento. A estrutura de ficheiros no contentor correspondente à *blob storage* segue a mesma hierarquia do diagrama da Figura 22.

A gestão de carregamento e descarregamento de ficheiros é feita através de um serviço em *.net core 5*, utilizando o *SDK* disponibilizado pela *Microsoft* para este mesmo efeito.

O desenvolvimento deste módulo como uma REST API independente permite a separação de responsabilidades, a modularidade da API como um serviço, para replicação seletiva em cenários de alta procura e a reutilização deste módulo por projetos futuros. Este serviço será denominado de Serviço de Gestão de Armazenamento (SGA).

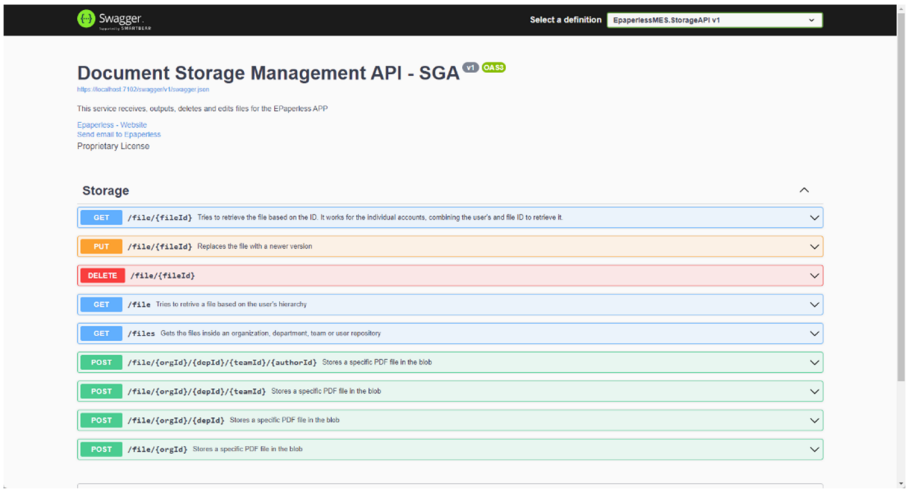

Figura 23 - Documentação do serviço de armazenamento
Fonte: Autoria Própria

A documentação deste serviço é feita através do *OpenAPI* (Swagger, 2020), uma especificação de documentação de *APIs*. A especificação de interfaces permite

que qualquer aplicação compreenda a forma como cada serviço comunica, tendo sido utilizada esta especificação para todos os serviços presentes na solução desenvolvida, permitindo a uniformização da comunicação com cada um dos serviços entre si. Esta arquitetura permite a adesão ao princípio de única responsabilidade, tornando-se uma solução mais fácil de manter e mais coesa.

Como é possível observar na Figura 23, o serviço de armazenamento dispõe de nove *endpoints*: quatro *endpoints* onde o serviço recebe um documento e guarda-o numa localização apropriada; três serviços que devolvem os ficheiros com base na sua localização ou ID; um serviço para substituir ficheiros e um serviço para eliminar ficheiros.

Para a utilização deste serviço, o *core* chama cada um dos *endpoints*, já tendo as informações do utilizador autenticado, e gere as suas respostas com base no recurso que o utilizador pediu. Este serviço é unicamente responsável por armazenar os ficheiros, então as suas ações são focadas neste aspeto. Por exemplo, o SGA não necessita de saber qual a hierarquia do utilizador nem se este tem permissões para aceder a determinado ficheiro ou pasta, já que esta validação é da responsabilidade doutro componete. Se o SGA mantiver a interface, com a mesma assinatura em cada um dos seus *endpoints,* para a solução *core* a sua implementação será sempre irrelevante, podendo esta ser modificada a qualquer momento, alterando-se lógica de negócio, armazenamento ou, por exemplo, fornecedor de *cloud.*

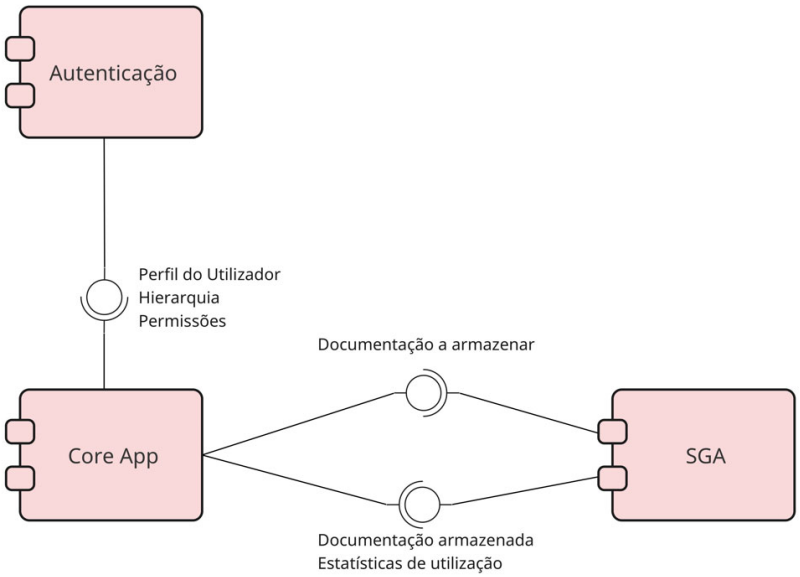

Figura 24 - Digrama de componentes do SGA
Fonte: Autoria Própria

Como é possível observar na Figura 24, a componente *core* comunica com a Autenticação para obter os dados referentes ao utilizador, sabendo assim a quais ficheiros tem acesso e que operações este pode fazer. Já em relação ao SGA, complementando o exibido na Figura 23, o *core* troca mensagens quando necessita de guardar ou obter ficheiros do SGA. É ainda possível à componente disponibilizar os seus dados estatísticos, como por exemplo, a percentagem de armazenamento em utilização por cada utilizador ou por hierarquia. Não existe uma interface que permita executar um pedido sobre essa informação, mas esta é armazenada em ficheiros de metadados que podem ser consultados sempre que necessário.

4.5.3. Conversão de *pdf* para imagem

De modo a podermos reconhecer um texto num determinado documento, algumas das abordagens necessitam requer uma ou mais imagens e não um ficheiro de *pdf*. Para esta conversão, foi desenvolvida um serviço em .Net 6. Este serviço é composto por dois componentes: uma REST API, que responde aos pedidos efetuados por outros serviços e pela aplicação *Core*; um módulo de manipulação de documentos que permite converter *PDFs* em imagens e imagens em *PDF*, converter ficheiros de texto em *PDF* e algumas validações necessárias para os ficheiros.

Para esta tarefa foi utilizado o Magick.NET (Lemstra, 2021), uma biblioteca *opensource* que representa uma abstração para .NET do ImageMagick. Esta biblioteca permite realizar as operações relacionadas com a manipulação de imagens e conversão de formatos.

Optou-se pela utilização de um serviço independente, e não pelo desenvolvimento de uma classe, pelo facto do repositório de dados onde se encontram os documentos ser externo e de poder este serviço ser utilizado por algumas aplicações que não a solução descrita. Como o serviço descrito em 0, este serviço de conversão também foi definido conforme o OpenAPI.

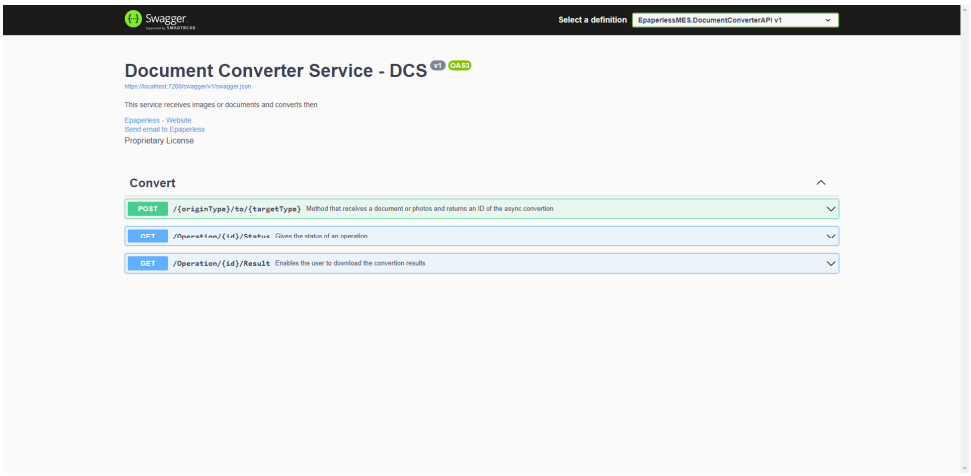

Figura 25 - Interface do serviço de conversão de ficheiros
Fonte: Autoria Própria

Como é possível observar na Figura 25, este serviço conta com três pontos de entrada: um método *Post* que permite ao módulo *core* executar o upload do documento que foi carregado pelo utilizador, retornando este um identificador de tarefa, para ser utilizado nos passos seguintes; um método *Get* que permite ao *core* inquirir sobre o estado de uma determinada tarefa, para isso recebendo o identificador da mesma como parâmetro de entrada; um segundo método *Get* que recebe um identificador de tarefa e retorna o resultado da conversão, em forma de um ficheiro ou de um conjunto de ficheiros, identificados pelo caminho para a pasta dos mesmos.

A determinação da natureza assíncrona do processamento da conversão, sendo esta feita a dois tempos (primeiro o upload e a seu tempo a obtenção dos resultados),

permite um processamento não bloqueante de pedidos, aderindo à natureza de não existir um estado definido de um pedido HTTP.

Quando é efetuado um pedido com sucesso ao DCS para converter um documento, este retorna simplesmente um identificador da tarefa, como referido anteriormente na descrição do método.

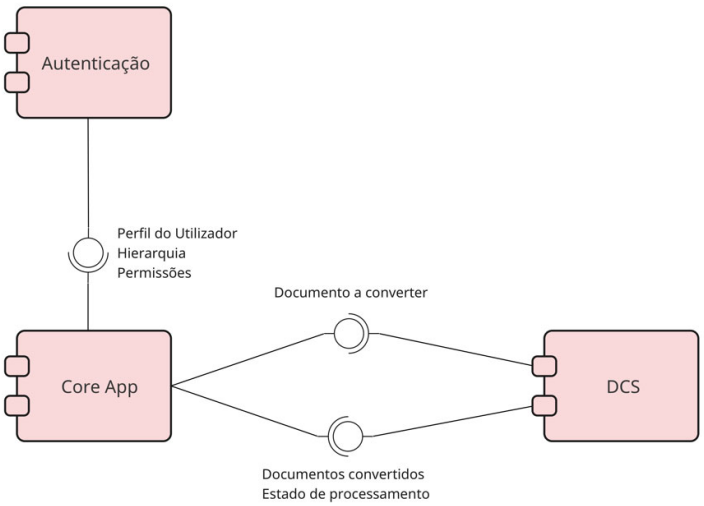

Figura 26 - Diagrama de componentes do módulo DCS
Fonte: Autoria Própria

O facto de ser retornado o ID (e de o *core* não ficar a aguardar uma resposta do ficheiro convertido) permite que não torne esta uma operação bloqueante, atribuindo o ónus da gestão de pedidos ao *core,* podendo este ser implementado de acordo com as regras especificas de negócio, que poderão ser de validar se a conversão está pronta num intervalo constante de tempo, por exemplo, ou só quando o utilizador realmente necessitar do documento convertido.

O método que permite este comportamento, como exemplificado na Figura 26, é o "Estado de processamento", que serve como uma aferição ao estado do pedido, otimizando assim os pedidos, já que um pedido por um conjunto de documentos é bastante maior no que diz respeito a tráfego de internet, do que um pedido pelo estado de um pedido.

Foi necessário ainda ter em conta os tipos de resultados esperados deste serviço, já que este pode retornar um único documento (no caso da conversão de um conjunto de imagens para um único ficheiro pdf) ou *n* imagens (no caso da

conversão contrária). Decretou-se que este serviço iria exportar os seus resultados em formato de lista de ficheiros com os caminhos absolutos dos mesmos, englobando assim todos os cenários de utilização.

4.5.4. Pré-processamento

No caso desta solução, a tarefa de pré-processamento é referente ao pré-processamento de imagens, já que o pré-processamento referente ao processo de Machine Learning foi efetuado aquando da construção do modelo. O pré-processamento de imagens, apresentou-se como não totalmente essencial, já que a qualidade das fotografias apresentadas é bastante satisfatória, normalmente proveniente de um documento capturado utilizando um *scanner* ou uma aplicação que já efetua algum pré-processamento. Posto isto, o processo é composto de:

- Rotação da imagem
- Correção de perspetiva
- Binarização

Para estas ações foi criado um serviço que compreende controladores, um para cada tipo de ação a executar. No que diz respeito à rotação da imagem, esta pode ser efetuada automaticamente, tentando o serviço rodar a imagem com base nas suas características e metadados e manualmente, havendo um *endpoint* que recebe a imagem e um ângulo de inclinação e roda-a para estar de acordo com esse ângulo. Já a correção de perspetiva é feita manualmente, pelo menos por este serviço, já que não foi possível identificar nenhuma biblioteca que ajudasse na tarefa de o fazer automaticamente. A correção é feita recebendo os pontos correspondentes aos vértices de uma caixa da imagem fonte (que representa um retângulo em formado de retrato e que contém todo o documento) e um retângulo que representa o mesmo retângulo, mas em perspetiva. O algoritmo presente no serviço efetua a correção e retorna a imagem rodada.

Por último, a binarização irá transformar uma imagem complexa numa mais simples, tornando-a por exemplo na mesma imagem em preto e branco, através de um método para o efeito.

4.5.5. Extração de dados

A extração de dados, de ficheiros PDF, é feita por um serviço desenvolvido em *python (posteriormente exposto utilizando interfaces em .Net 6)* para este âmbito. Este serviço, daqui em diante denominado de Serviço de Extração de Dados (SED), materializado numa solução desenhada por camadas, como exibido na Figura 27, permite ao utilizador executar pedidos de processamento de ficheiros, comunicando com outros serviços, mais concretos, que permitem o processamento recorrendo a ML ou ao processamento por *templates*.

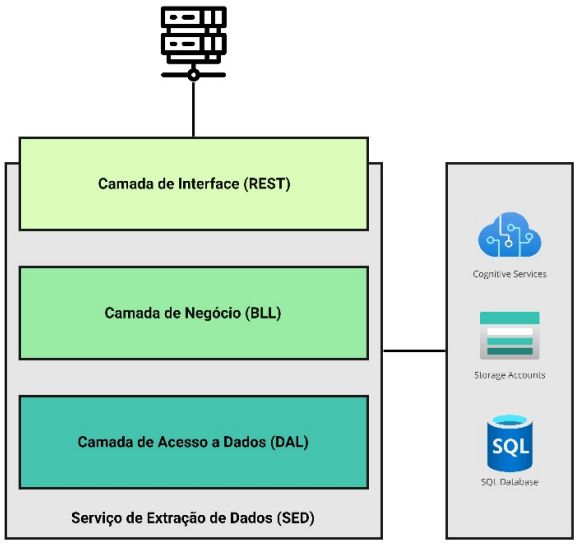

Figura 27 - Arquitetura lógica do serviço de Machine Learning
Fonte: Autoria Própria

Como cada serviço é desenhado como um componente independente, optou-se pela estruturação deste serviço com uma arquitetura lógica por três camas: uma camada responsável por aceder à persistência de dados, seja esta uma base de dados ou um serviço de armazenamento de ficheiros, como é o caso do utilizado, o Azure Storage Accounts (Pande, 2020); uma camada responsável por todo o processamento de dados e lógica de negócio; uma camada de abstração que expõe o serviço através de uma interface através de uma arquitetura REST.

A utilização de uma arquitetura em três camadas tem como vantagens o facto de se abstrair cada um dos componentes, diminuindo o acoplamento, aumentando

a coesão e criando código mais facilmente manutenível (Microsoft Corporation, 2009).

A utilização deste serviço começa quando um pedido é recebido pelo *Backend* e encaminhado para a camada de interface. Esta interface utiliza o OpenAPI para disponibilizar métodos aos restantes serviços. Para o reconhecimento, este serviço conta com dois métodos, um que recebe somente o identificador do documento a processar e um outro que para além desta mesma informação recebe ainda um identificador do *template* a utilizar para processar mais facilmente o documento.

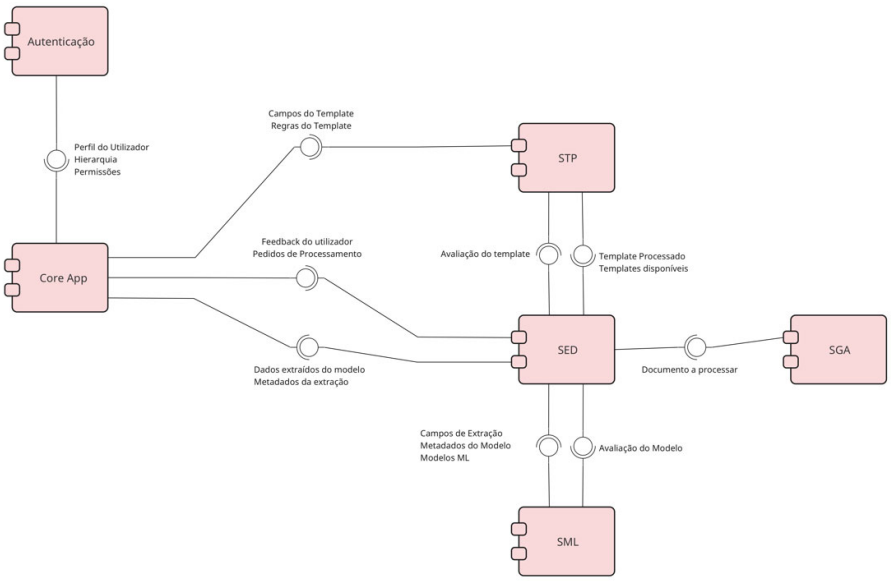

Figura 28 - Diagrama de componentes do SED e restantes serviços integráveis
Fonte: Autoria Própria

O SED comunica diretamente com a aplicação core e com outros serviços, como o de gestão de documentos para, por exemplo, obter os documentos com base no seu identificador (comunicando com o SGA), fazer o processamento todo e devolver a resposta ao utilizador através da aplicação core, como exibido na Figura 28.

A interação dos vários serviços neste caso de uso é essencial, tendo o desenho sido pensado com vista a minimizar a interdependência entre as partes. Prevê-se que o SED necessite de escalar bastante, já que é o ponto central das operações relativas com o processamento.

A implementação do reconhecimento por *template* tem como base a ideia de que faturas do mesmo comerciante tendem a seguir a mesma formatação, por exemplo, uma fatura do IPS deverá ter sempre o NIF do IPS na mesma localização, o mesmo se aplica ao valor total de fatura e a todos os outros elementos a extrair do documento. Esta abordagem é a mais simples e rudimentar, já que irá ser baseada no input do utilizador para identificar o que cada texto significa. É também a abordagem mais performante e que consome menos recursos computacionais, já que somente necessita de realizar o processo de OCR.

Quando o *endpoint* recebe o pedido com o documento e com o *template*, o primeiro passo é a normalização deste último à dimensão da imagem, já que poderá dar-se o caso da fatura em que o *template* foi feito não ser exatamente do mesmo tamanho que o documento a analisar. Normalizado o template, o documento passa por um processo de OCR, utilizando os *Cognitive Services* ou o *Tesseract* para este efeito, recebendo deste passo o texto reconhecido, o grau de confiança do reconhecimento e as *bounding boxes* do texto. Estas *bounding boxes* representam as coordenadas em que determinado texto está dentro do documento e são geralmente utilizadas em reconhecimentos de texto recorrendo a ML.

Contando já com o *template* normalizado, as *bounding boxes* e o texto reconhecido, o último passo é comparar estes dois primeiros para aferir que *bounding boxes* estão contidas dentro de cada campo do *template*. Se estiverem contidas, ou parcialmente contidas dada uma margem de erro, então o texto dentro da *bounding box*, é identificado com a etiqueta relativa ao campo do *template.*

Findado este passo, é devolvido pelo método, cada campo reconhecido, as suas respetivas coordenadas em forma de *bounding box* e o template normalizado para posteriormente ser exibido ao utilizador. É ainda devolvido o grau de confiança do processamento, compreendido pela média aritmética das confianças recebidas pelo OCR.

No reconhecimento através de ML, é somente recebido o documento a processar. Este documento é obtido do mesmo modo que no reconhecimento por *templates* e de seguida é enviado como dado de entrada para o modelo de ML que

retorna a inferência dos campos do documento e respetiva confiança e *bounding boxes.* São devolvidos os mesmos dados que no processamento por *templates*.

Ambas as abordagens se complementam. Um utilizador poderá optar, caso tenha um modelo de faturas frequente, criar um template e ter o seu processamento mais rapidamente, ou submeter o documento para processamento por ML, receber o resultado e criar um template com base nesse reconhecimento. O sistema foi ainda desenhado, como descrito anteriormente para que no futuro seja possível que o modelo aprenda com o utilizador, já que quando recebe os resultados do processamento de ML este poderá validar ou não a identificação de campos e retificar as *bounding boxes* identificadas pelo modelo. Existe um método no SED que recebe como entrada o documento, o reconhecimento que foi feito e a retificação que o utilizador fez. Estes dados são guardados e servem para posteriormente serem analisados e alimentados através do *feedback loop*. Optou-se pela não correção automática do modelo, já que é necessário a garantia de qualidade dos dados destas correções. Estas correções são validadas por um administrador da aplicação e posteriormente são utilizadas como dado de treino na melhoria do modelo.

Na secção de ML para além de métodos para processar documentos, existem ainda métodos complementares para manipular o modelo de ML, contando com um método que retorna uma lista com todos os modelos disponíveis para serem utilizados, um método para atualizar o modelo atualmente utilizado por defeito, um método para treinar o modelo atual e um método que retorna informações relevantes sobre o modelo atual. Em relação a este último é retornado, em formato *JSON*, os metadados relativos ao modelo (por exemplo a data e hora de início e fim do processo de treino) bem como os campos presentes no resultado do modelo e qual a sua precisão média de inferência.

Este retorno é essencial para a avaliação do modelo, já que apresenta os metadados do mesmo, como se pode verificar na Figura 29, onde é possível observar um identificador do modelo (*modelId*), o nome do modelo (*modelName*), e algumas outras propriedades do modelo e dos seus sub-modelos, cada um dos campos a inferir, como é o caso do *CommercialDiscounts*, campo que representa

os descontos numa fatura. É possível ver ainda a precisão média ao inferir este campo, 68%. No total, para o modelo mais recente à data de escrita do presente documento, tenta-se inferir 45 campos diferentes dentro de uma fatura.

```
{
  "value": {
    "modelId": "8b776129-034a-4ad0-87a2-818bf4cdc083",
    "modelName": "taloes-vs-a4",
    "properties": {
      "isComposedModel": true
    },
    "status": 1,
    "trainingStartedOn": "2021-10-15T10:14:45+00:00",
    "trainingCompletedOn": "2021-10-15T10:14:46+00:00",
    "submodels": [
      {
        "modelId": "180e7788-32b1-448b-a581-5082332f5cb8",
        "formType": "custom:180e7788-32b1-448b-a581-5082332f5cb8",
        "accuracy": 0.597,
        "fields": {
          "CommercialDiscounts": {
            "name": "CommercialDiscounts",
            "accuracy": 0.68,
            "label": null
          },
          "Currency": {
            "name": "Currency",
            "accuracy": 0.995,
            "label": null
          },
          "CustomerAddress": {
            "name": "CustomerAddress",
            "accuracy": 0.48,
            "label": null
```


Figura 29 - Exemplo de informações sobre o modelo
Fonte: Autoria Própria

Na Figura 30 é possível visualizar alguns dos campos a serem extraídos de uma única fatura. A imagem apresentada representa um caso ótimo, onde o modelo ignoraria todo o conteúdo que não é útil para a sua tarefa e identificaria por completo toda a informação necessária.

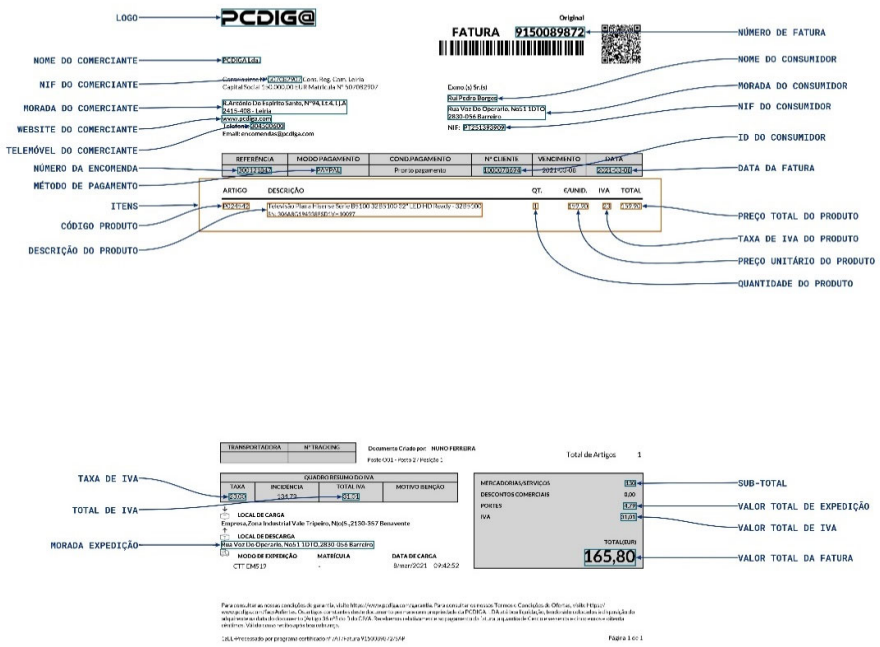

Figura 30 - Exemplo de informação extraída de uma fatura
Fonte: Autoria Própria

Em contraste com a Figura 30, a Figura 31 apresenta o reconhecimento do documento com recursos a ML. É possível observar que foram reconhecidos 23 campos do documento, tendo sido a confiança deste reconhecimento de 73%. Utilizou-se *NodeJS* para elaborar um módulo que permite exportar uma fatura com os respetivos campos identificados, não fazendo este módulo parte da solução pois foi usado maioritariamente em âmbito de desenvolvimento para validação do processo de reconhecimento. Esta ferramenta tira partido do módulo *Canvas* (Holowaychuk et al., 2022) que permite a manipulação de imagens por parte do NodeJS, criando uma área de desenho que pode ser programaticamente manipulada.

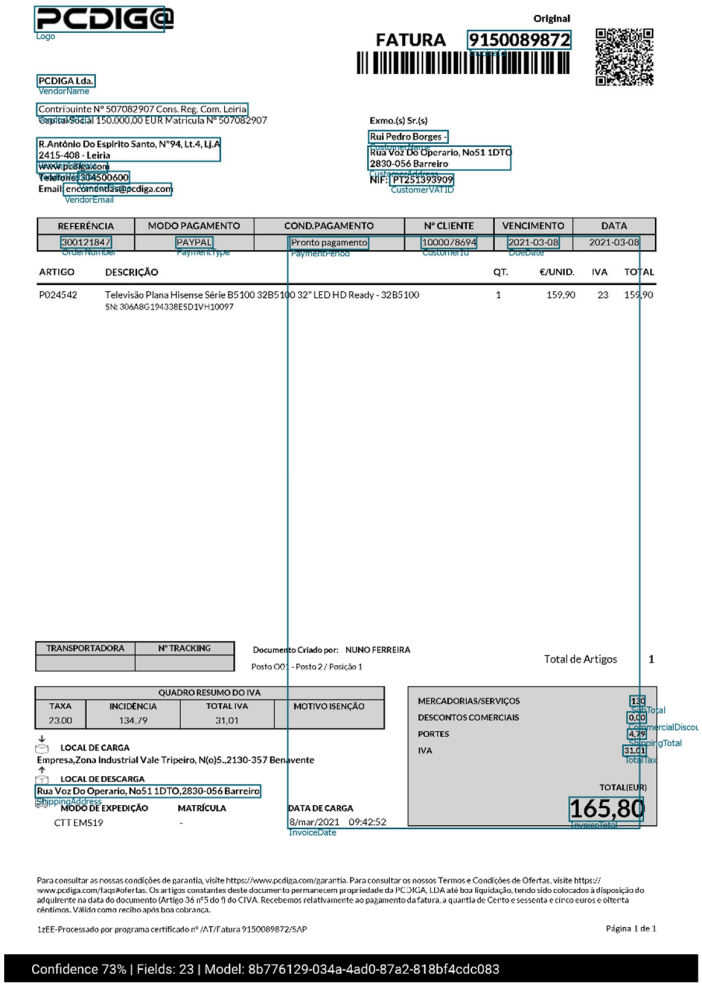

Figura 31 - Reconhecimento de uma fatura utilizando o modelo de ML
Fonte: Autoria Própria

Analisando a imagem da Figura 31, é claramente evidente que se afasta de um caso ideal. Contudo, é de notar que os dados principais de uma fatura são corretamente identificados. É ainda interessante de evidenciar que nenhum dos campos do produto foi reconhecido no exemplo. Esta questão, apesar de não ser impeditiva do propósito da solução a desenvolver, já que esta informação é considerada acessória, exibe claramente uma limitação da mesma. Este tipo de dados, não estruturados, não delimitados e muitas vezes com simbologia ausente, apresenta o maior desafio para uma solução de ML como a que se pretendia desenvolver.

A base da lógica de negócio do serviço desenvolvido assenta, como referido anteriormente, no *form recognizer* da Microsoft. Atualmente, a Microsoft

disponibiliza vários modelos pré-treinados para vários tipos de documentos e a capacidade de treinar modelos proprietários utilizando exemplos próprios. Destes modelos disponíveis, à data de início do desenvolvimento do módulo e para o caso de uso que é pretendido, o *Invoice Model* seria o que mais responderia às necessidades de processamento de uma fatura, já que foi exatamente pré-treinado para este efeito. Contudo, existiam algumas limitações que fizeram optar por um modelo customizado (*custom model* na nomenclatura do Azure) como ponto de partida. Esta abordagem representou um peso considerável de desenvolvimento, já que ainda não existir um modelo pré-treinado para o idioma português.

Contudo, no decorrer do desenvolvimento do módulo de extração de entidades, foi publicado pela Microsoft um modelo adaptado a faturas em português, estando esse modelo até à data de escrita deste documento publicado como ainda em testes, em versão *beta.* Foi este segundo o modelo mais utilizado na POC, podendo a qualquer altura efetuar-se a alteração de qual modelo a solução utiliza. Tanto o *Custom Model* como o *Invoice Model* são disponibilizados como modelos prontos a utilizar, sendo que o *Invoice Model* foi pré-treinados em documentos específicos (faturas) e que o *Custom Model* pode ser treinado com recurso a documentos e etiquetagem próprios.

Neste ponto é importante mencionar que nem todas as informações requerem a utilização de ML para a sua extração. Aliás, num ambiente corporativo quer-se minimizar a utilização de modelos *cloud* de ML, já que esta utilização significa custos. Um exemplo desta extração são campos altamente estruturados, como é o caso do código postal, NIF, email, datas e números de telemóvel. O NIF, por exemplo, é um exemplo de um tipo de dados que, apesar de ter muitas variações (cada indivíduo e empresa possui um diferente), segue sempre a mesma regra de validação, o que facilita o processo de extração após o OCR do documento. Esta validação pode ser feita através da chamada a serviços que são disponibilizadas para este efeito por entidades externas, bibliotecas ou através de um algoritmo para este efeito.

A opção pela manutenção do princípio da responsabilidade única é evidente quando se analisa a Figura 28, já que, apesar de existir um microserviço central de

extração de dados, não é este que executa o processamento propriamente dito, mas sim dois outros componentes: STP, responsável pelo processamento de documentos utilizando *templates*; SML, responsável pelo processamento de documentos recorrendo ao modelo de Machine Learning.

4.5.6. Avaliação de resultados de processamento

Depois de executar o pedido de processamento de um ficheiro, é exibido ao utilizador o resultado deste processamento, como exibido na Figura 32, onde este pode validar os dados presentes no documento, efetuar algumas alterações necessárias aos valores em cada um dos campos e salvar estes campos.

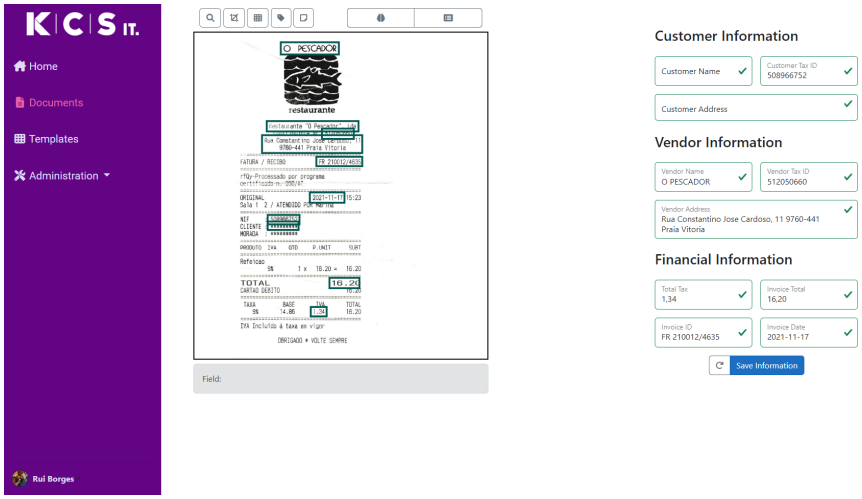

Figura 32 - Ecrã de validação de processamento
Fonte: Autoria Própria

Tanto o processamento original quanto os dados atualizados pelo utilizador são guardados em base de dados, incluindo os metadados de cada uma das ações, para posterior melhoramento do modelo.

4.5.7. Exportação de dados

Os dados contidos na solução, depois do processamento, são guardados em base de dados de forma estruturada, para que possam ser consultados e exportados. Na fase de desenho, foi ponderada a utilização de vários recursos de base de dados, incluindo bases de dados não relacionais, orientadas ao armazenamento de documentos. Existem concessões no que diz respeito à escolha a adotar, já que ao

escolher uma abordagem não relacional beneficia-se com a ausência de necessidade de existir à partida uma formulação de dados rígida tornando, contudo, a solução mais complexa.

Os dados são guardados em formato *JSON*, que pode ser guardado em formato de texto na base de dados relacional, mas também em ficheiro no sistema. Dado que ambas as abordagens apresentam vantagens, optou-se por uma conjunção das mesmas, guardando-se uma cópia do processamento mais recente em disco, para fácil exportação neste formato e processamento mais célere e armazenamento em base de dados, guardando os campos chave de cada documento nesta estrutura, permitindo a sua rápida visualização correção e categorização. A escolha do armazenamento também em ficheiro prendeu-se ainda com o facto de ser necessário guardar metadados, como *bounding boxes* e alguns dados relativos à performance do reconhecimento.

A exportação de dados retorna os valores em:

- JSON, sendo retornado um dicionário com cada campo e o respetivo valor e os valores das *bounding boxes* associadas aos mesmos;
- CSV, sendo retornado o conteúdo do processamento onde cada campo é separado do seguinte por vírgula;
- TSV, sendo retornados os campos do documento e os seus conteúdos, separados pelo caracter *TAB*;
- XLSX, sendo retornado um ficheiro compatível com Excel com os dados de processamento.

O utilizador tem disponível a funcionalidade de exportar os dados relativos a um processamento individual ou exportar vários processamentos ao mesmo tempo.

5. Apresentação e Discussão de Resultados

5.1. Apresentação de resultados

A investigação descrita neste documento resultou numa aplicação Web capaz de processar documentos financeiros e exportar os seus dados para diversos

formatos e, especificamente, num módulo de ML que permite a extração de entidades de documentos.

O sistema desenvolvido tinha como objetivos principais a capacidade de receber documentos financeiros, nomeadamente faturas, extrair destes dados essenciais para a sua catalogação e exportar os dados para formatos que seriam utilizados para outros fins e por outros *softwares*.

A solução de ML desenvolvida assenta em duas abordagens distintas: um modelo customizado para a realidade do problema e um modelo pré-definido e pré-treinado. O modelo de ML customizado, treinado com base em 133 faturas, apresenta uma acurácia média de 0,63 ou 63% tendo em conta todos os 45 campos em si presentes. Quando se tem em conta somente os campos essenciais para o processamento do documento, nomeadamente o nome, morada e número de identificação fiscal do cliente e da empresa vendedora e data, identificador, valor total e valor total de impostos da fatura, a confiança média sobe para 0,66 ou 66%.

Já o modelo providenciado pela Azure apresenta, no mesmo *dataset* utilizado para testar o modelo customizado uma confiança média de 0,75 ou 75% quando se tem em conta todos os campos que este modelo consegue extrair, subindo para 0,76 ou 76% quando tidos em conta os campos essenciais.

Optou-se por defeito em utilizar-se o modelo mais recente disponibilizado pela empresa, já que apresentou melhores resultados durante a fase de desenvolvimento.

Para medição da performance do modelo aplicado não é possível simplesmente elaborar uma matriz de confusão a partir do processamento dos dados, já que esta métrica toma em conta somente o resultado da predição da classe de determinado ponto de dados. Para aferição da performance foi utilizado um *subset* das faturas atuais, contendo a proporção de 10% dos pontos de dados do universo, ou seja, 19 faturas, não tendo sido estes documentos utilizados no processo de treino dos modelos que testam.

Para aferição da qualidade do modelo teve-se em conta os campos essenciais presentes no documento e procedeu-se à extração manual dos valores para cada um dos campos. Posteriormente, as mesmas faturas foram processa-das utilizando a

solução desenvolvida e os resultados analisados. A análise dos dados apresenta um desafio acrescido, quando comparando por exemplo com um problema de classificação tradicional, já que não somente se tem de avaliar a classificação da entidade, mas também do texto que é extraído. Mesmo na avaliação da extração do texto, é limitativo executar uma avaliação binária dos resultados (correto ou incorreto), já que a avaliação deve penalizar predições que são completamente dispares com o valor real e valorizar predições que, por exemplo, têm somente um caracter errado. (Segura-Bedmar et al., 2013) propõe uma abordagem baseada em vários padrões de contagens:

- Correspondência estrita: Correspondência exata de tipo e limites;
- Correspondência exata de limite: Correspondência de limite desconsiderando o tipo;
- Correspondência parcial de limite: Correspondência de limite parcial desconsiderando o tipo;
- Correspondência de tipo: Correspondência de tipo, desde que exista alguma sobreposição da predição e do valor real.

Cada um destes padrões pode ser avaliado independente dos outros ou em conjunção. Para cada categoria de contagem, a predição foi avaliada também tendo em conta o acerto relativo, como proposto por (Chinchor & Sundheim, 1993) em que cada correspondência é classificada como:

- Correta (COR): A predição e texto real são considerados iguais;
- Parcial (PAR): A predição e texto real são consideradas uma correspondência aproximada;
- Incorreta (INC): A predição e texto real são consideradas como incorretas, não existe correspondência;
- Falta (MIS): O texto real existe, mas não foi feita qualquer predição;
- Hipotético (SPU): Existe uma predição, mas não existe correspondência no texto real;
- Não Existente (NON): Ambos o texto real e predição são vazios.

Na Tabela 4 é possível visualizar uma amostra da classificação, para a cada um dos documentos para a entidade *Vendor Address* (Morada do Vendedor). De modo

a assegurar a privacidade dos dados contidos nos documentos, foram truncadas as colunas relativas ao texto real e predição. Na tabela é possível ver o identificador do documento, entidade, a distância de Levenshtein, e uma coluna para cada padrão de contagem. Em cada uma destas últimas, as células apresentam as classificações segundo (Chinchor & Sundheim, 1993) em conjunção com a Distância de Levenshtein (LD).

Tabela 4 - Comparação de resultados da entidade "Vendor Address" para cada um dos documentos
Fonte: Autoria Própria

Document ID	Entidades	LD	Type	Partial	Exact	Strict
16d82208-1864-4d8d-bfee-86591fe27e41	Vendor Address	0	COR	COR	COR	COR
fecfcf04-8218-44a5-991b-5a67e29fbcbf	Vendor Address	1	COR	PAR	INC	INC
c7643673-6fcf-4463-87f6-86ea5b74379c	Vendor Address	0	COR	COR	COR	COR
04f59d21-cafa-4aa2-a42f-34188af99574	Vendor Address	0	COR	COR	COR	COR
148971f5-34dc-4bbf-94f0-0d507764f2f10	Vendor Address	0	COR	COR	COR	COR
59b05197-553a-4c60-90b2-8319d66d5530	Vendor Address	0	COR	COR	COR	COR
ab8e7d6b-efca-49ae-aac6-0e36321a3f40	Vendor Address	21	COR	INC	INC	INC
97b14476-8bad-46ea-972b-94a3a484e540	Vendor Address	0	COR	COR	COR	COR
0b75c616-ce71-490e-a092-66d13c838508	Vendor Address	0	COR	COR	COR	COR
968e85e1-7d62-415e-b4f5-d654823e6673	Vendor Address	58	COR	INC	INC	INC
ca45aea5-70df-4777-9339-b4dc7e9f7da14	Vendor Address	0	COR	COR	COR	COR

dacffe3f-e345-4ce8-bc02-b351fec141d11	Vendor Address	1	COR	COR	COR	COR
373dafe7-dba6-49bd-a0c4-1ae2c480ed50	Vendor Address	11	COR	PAR	INC	INC
aef0de6d-879e-4eff-b25a-822851bd7a6b	Vendor Address	0	COR	COR	COR	COR
9a0b457b-ec8f-4c4a-99c0-bfd5f46a8731	Vendor Address	0	COR	COR	COR	COR
12aff662-cfba-4cc3-a67d-4d133faca49e	Vendor Address	0	COR	COR	COR	COR
364551b2-0611-4a71-a014-e98412a2f60e	Vendor Address	0	NON	NON	NON	NON
1f686f9a-f663-4447-afbf-0232615c976e	Vendor Address	0	COR	COR	COR	COR
bfe331d5-a0f3-4ff9-9458-3580b625022f	Vendor Address	0	COR	COR	COR	COR

Após a comparação do total dos tuplos de dados, é elaborada a contagem de cada possível classificação. Obtém-se a Tabela 5, onde é possível visualizar a contagem de cada classificação por cada padrão de contagem.

Tabela 5 - Contagem dos tipos de correspondência por padrão de classificação
Fonte: Autoria Própria

	Type	Partial	Exact	Strict
Correct	137	96	96	96
Incorrect	5	5	46	46
Partial	0	41	0	0
Missing	20	20	20	20
Spurius	1	1	1	1
Noncommittal	27	27	27	27
Possible	162	162	162	162
Actual	143	143	143	143
Precision	0,958042	0,671329	0,671329	0,671329
Recall	0,845679	0,592593	0,592593	0,592593

F1	0,898361	0,629508	0,629508	0,629508

Estão ainda presentes na tabela as métricas referentes à Precisão, Recall, F1 e os cálculos referentes aos casos possíveis e reais. Estas métricas e cálculos são dados seguindo as seguintes formulas:

$$possible\ (POS) = COR + PAR + INC + MIS$$

$$actual\ (ACT) = COR + PAR + INC + SPU$$

$$recall = \frac{COR}{POS}$$

$$precision = \frac{COR}{ACT}$$

Estes dados apresentam uma precisão de ≈0,96 na seleção do tipo correto da entidade. Apresentam ainda uma precisão de ≈0,67 para um reconhecimento parcial da entidade ou texto.

Não interessando somente nem o tipo ou nem o reconhecimento do texto, os autores anteriormente citados sugerem uma métrica que tenha em conta todos os dados e que tenha em conta as duas realidades, sendo o cálculo da precisão e recall o seguinte:

$$recall = \frac{COR + (PAR * 0{,}5)}{POS}$$

$$precision = \frac{COR + (PAR * 0{,}5)}{ACT}$$

Os resultados são apresentados na Tabela 6, onde é possível ver num cálculo de *precision, recall e f1* ponderando não só as instâncias completamente corretas mas também as parcialmente corretas, utilizando as fórmulas acima.

Tabela 6 - Tabela de métricas gerais
Fonte: Autoria Própria

COR	425
INC	102
PAR	41
MIS	80
SPU	4
Possible	648
Actual	572
Precission	0,6875
Recall	0,778846
F1	0,730328
$\bar{x}(dl)$	3,978947

Na tabela acima é possível consultar que a precisão média com base na ponderação é de ≈0,69, o *recall* ≈0,78, o F1 ≈0,73 e a média de DL de ≈4.

O F1-Score serve primariamente para a comparação de vários classificadores e poderá ser utilizado para comparar este modelo com futuros modelos. A distância de Levenstein foi calculada com base no algoritmo tendo sido usado como dados de entrada a normalização para maiúsculas do texto de origem e texto de objetivo.

5.2. Sumário, apreciação crítica, ou interpretação dos resultados

Previa-se que seria a tarefa de reconhecimento de campo neste tipo de documentos seria difícil. Esta tarefa foi adereçada com a componente de ML da solução.

O modelo utilizado na solução final apresenta uma precisão média ponderada de ≈0,69, tendo em conta as correspondências exatas e parciais. São valores que se apresentam como positivos no reconhecimento dos campos, mas que implicam alguma limitação na utilização destes modelos. No que diz respeito à distância de Levenshtein, a média é de 4 operações, isto é, para cada previsão que o modelo realiza, o operador terá de fazer em média 4 operações (substituições, eliminações ou inserções). Significa então que, mesmo quando a previsão do modelo não é

perfeita, poucas operações são necessárias para que o utilizador retifique alguma incorreção que o modelo possa fazer.

Dado o parco número de documentos disponíveis para teste da solução, não é possível retirar ilações estatísticas dos métodos de análise escolhido, carecendo o de mais dados para uma validação mais cuidada. É possível, contudo, efetuar uma análise qualitativa dos resultados, que se apresentam como positivos.

As faturas são documentos que, apesar de necessitarem de ter praticamente sempre as mesmas entidades em si presentes, pelo menos no que diz respeito à realidade portuguesa, apresentam uma grande heterogeneidade de formatos. Ainda hoje é possível um comerciante passar uma fatura de forma manual, escrevendo num formulário os dados necessários para que esta seja válida, facto que exemplifica a dificuldade de elaborar um modelo que consiga responder a todas as diferentes implementações de um documento deste género. Acresce ao problema anteriormente descrito o facto de existirem vários documentos com entidades similares, como notas de crédito e notas de encomenda, dificultando ainda mais o processo de reconhecimento. Este foi o maior desafio, já que não os dados nem sempre estão presentes e, quando estão, nem sempre estão na mesma localização. Pode ser esta uma das razões para ainda grande parte da faturação utilizar alguma forma de papel.

Apresentou-se ainda como desafio o facto de a tarefa em questão ser direcionada para o idioma português. A maior parte dos modelos pré-treinados estão preparados para receber como input documentos em inglês, não contando com acentuação ou cedilhas. Esta limitação é importante e foi identificada *à priori* como um risco no desenvolvimento da solução, exigindo assim um modelo customizado para lidar com este desafio, à data de início do desenvolvimento, tendo-se optado por essa decisão. A construção deste modelo customizado foi auxiliada posteriormente pela introdução do modelo pré-treinado para português do *Azure Form Recognizer*. No seguimento do desafio anterior, existe ainda o desafio do *cold start* numa tarefa de ML deste género, a necessidade de contar com faturas para *labeling*, treino e teste do modelo. Usualmente são necessárias grandes quantidades de dados para o treino de modelos e este tipo de documentos não existe em bases

de dados públicas ou facilmente acessíveis, já que são normalmente de natureza confidencial.

Como limitação do desenvolvimento face ao planeado inicialmente é apresentada a falta de implementação da funcionalidade que permite executar o *feedback loop* para alimentar o modelo com as predições corrigidas por parte do utilizador. Esta funcionalidade demonstrou-se desafiante de desenvolver e não foi terminado o seu desenvolvimento, já que seria necessária nova modelação do comportamento do utilizador e do funcionamento do modelo.

Considera-se que a solução encontrada cumpre o que se propôs a executar. É uma solução que não ambiciona tornar automático todo o processo de processamento de faturas, mas sim representar uma ajuda valiosa na diminuição de erros e rapidez no processamento de várias faturas em simultâneo, já que é possível processar múltiplos documentos, tendo do utilizador somente validar ou corrigir os resultados do processamento. A construção de *templates* permite ainda a extração mais eficaz de dados dos documentos, sendo que o modelo, neste caso de OCR, não necessita de inferir a classe, mas somente o conteúdo presente nas fronteiras de cada campo do template.

6. Conclusão

Em suma, a investigação desenvolvida tem dois resultados concretos: uma aplicação web que permite que o utilizador carregue documentos, os categorize e processe assincronamente e em massa; um modelo de *machine learning* capaz de inferir a localização, tipo e conteúdo de entidades de negócio, recendo como dado de entrada um documento em formato *pdf* ou como imagem.

Após uma revisão dos conceitos teóricos base relacionados com ML, classificação e OCR, onde foram expostas as várias fases de um modelo e algumas das suas tarefas usuais, foram estudas as opções disponíveis para o desenvolvimento de uma prova de conceito, que engloba os dois resultados anteriormente mencionados. A solução desenvolvida permite ao utilizador carregar ficheiros, incluindo de forma massiva, criar *templates* para extração dos dados, extrair os dados com recurso aos templates ou ao modelo de ML, exportar esses

dados para formatos *standard* e fazer a gestão de utilizadores com base nos seus papeis. O sistema desenvolvido funciona assim como um *document management system*, com a particularidade de permitir o processamento dos dados auxiliado por inteligência artificial.

O serviço de ML capaz de extrair as entidades conta com duas fases de desenvolvimento que representam abordagens diferentes de reconhecimento: um modelo customizado e um modelo adaptado do modelo pré-treinado disponibilizado pela Microsoft. Na solução final optou-se por utilizar este último modelo, tendo obtido a maior confiança, apresentando uma precisão de cerca de 69%, com DL médio de 4. O modelo pelo qual não se optou para a versão final do desenvolvimento serviu o propósito de aprendizagem tecnológica, modelação do problema em concreto utilizando ML e como base para a construção de um futuro modelo adaptado à realidade dos utilizadores, utilizando as suas interações com os resultados que disponibiliza para aprender.

A aplicação Web foi desenhada tendo como princípio a facilidade de utilização do utilizador final, mas tendo por base que os utilizadores que a aproveitaram têm conhecimentos prévios de faturação. Esta não foi desenhada para substituir o Humano na tarefa de extrair os dados dos documentos, mas sim para o auxiliar nesta tarefa, permitindo uma otimização de recursos. Cada um dos serviços desenvolvidos foi desenhado simultaneamente para ser utilizados pela aplicação web, mas também para serem completamente independentes, podendo ser expostos no futuro a outras aplicações. Esta estratégia de desenho permite maximizar a reutilização de código, garantindo ainda que cada serviço é autónomo e coeso.

6.1. Recomendações e propostas para trabalho futuro

Construindo sobre o trabalho realizado à data, seria interessante a reformulação arquitetónica da solução desenvolvida, dado que a adoção de uma componente de *API Gateway* permitiria a adesão da solução a uma arquitetura típica de microserviços, beneficiando das suas características de escalabilidade. Pode utilizar-se, por exemplo, o pacote de desenvolvimento Ocelot (*NuGet Gallery | Ocelot 18.0.0*, n.d.) que foi exatamente concebido para este efeito. Este pacote permitiria a criação de um ponto de entrada para os pedidos, podendo eliminar-se a

componente core. A utilização desta abordagem permite diminuir o acoplamento geral da solução, não existindo qualquer elemento central de lógica de negócio, poderá representar ganho em performance e diminuição de latência, já que o *gateway* permite alguma agregação de pedidos num único, permite uma estratégia centralizada de segurança, evitando expor todos os serviços individualmente, diminuindo a "superfície de ataque" e permite a centralização de preocupações transversais, simplificando a implementação das funcionalidades desta categoria.

No que diz respeito ao desenvolvimento da componente de ML, recomenda-se o desenvolvimento do conjunto de funcionalidades que permita ao utilizador participar no *feedback loop,* bem como a integração dessas funcionalidades no próprio modelo. Um dos maiores investimentos que se realiza num projeto de ML é na Engenharia de Dados, onde é necessário garantir a extração dos dados, aferir a qualidade dos mesmos, filtrando-os e criando as interfaces necessárias para estes serem consumidos e manipulados. Existe um esforço considerado da equipa de um projeto deste gênero na tarefa de etiquetagem (*labeling*) e obtenção de dados. Com a inclusão do utilizador (e seus dados, tendo claro sempre em conta o seu anonimato) neste "ciclo de retorno" é possível o modelo aprender com a sua consecutiva utilização por parte dos seus utilizadores. Ao adicionar esta funcionalidade acredita-se que o modelo pode ser bastante melhorado, principalmente no que diz respeito a alguns tipos de documentos em específico, já que a qualidade do modelo para cada formato de documento é claramente afetada pela quantidade de exemplos deste tipo com os quais o modelo pode aprender.

A utilização de ferramentas similares com a desenvolvida é, na opinião do autor, uma mais-valia para várias áreas de negócio, nomeadamente as áreas da banca, jurídico, recursos humanos e imobiliário. Atualmente os processos das áreas anteriormente enumeradas ainda assentam fortemente no papel como meio de comunicação, podendo estes processos ser otimizados e digitalizados. A utilização de ML poderá auxiliar no processo de transformação digital, reconhecendo campos em documentos ou até mesmo reconhecendo entidades e sentimentos em documentos, recorrendo a NER e análise sentimental.

7.Referências

Adebowale, A., Idowu, S. a, & a, A. A. (2013). Comparative Study of Selected Data Mining Algorithms Used For Intrusion Detection. *International Journal of Soft Computing and Engineering (IJSCE)*, *3*(3), 237–241.

Ahmad, K. S., Ahmad, N., Tahir, H., & Khan, S. (2017). Fuzzy_MoSCoW: A fuzzy based MoSCoW method for the prioritization of software requirements. *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, 433–437. https://doi.org/10.1109/ICICICT1.2017.8342602

Al-Aidaroos, K. M., Bakar, A. A., & Othman, Z. (2010). Naïve bayes variants in classification learning. *2010 International Conference on Information Retrieval Knowledge Management (CAMP)*, 276–281. https://doi.org/10.1109/INFRKM.2010.5466902

Ali, P. J. M., & Faraj, R. H. (2014). Data Normalization and Standardization: A Technical Report. *Machine Learning Technical Reports*, 1–6.

Apostu, A., Puican, F., Ularu, G., Suciu, G., Todoran, G., & others. (2013). Study on advantages and disadvantages of Cloud Computing–the advantages of Telemetry Applications in the Cloud. *Recent Advances in Applied Computer Science and Digital Services*, *2103*.

Awad, M., & Khanna, R. (2015). Machine Learning. In *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers* (pp. 1–18). Apress. https://doi.org/10.1007/978-1-4302-5990-9_1

Basha, M., Bagyalakshmi, K., Ramesh, C., Rahim, R., MANIKANDAN, R., & Kumar, A. (2019). Comparative Study on Performance of Document Classification Using Supervised Machine Learning Algorithms: KNIME. *Australian Journal of Emerging Technologies and Society*, *10*, 148–153.

Bieniecki, W., Grabowski, S., & Rozenberg, W. (2007). Image preprocessing for improving OCR accuracy. *Proceeding of the 3rd International Conference of Young Scientists "Perspective Technologies and Methods in MEMS Design",*

MEMSTECH 2007, 75–80. https://doi.org/10.1109/MEMSTECH.2007.4283429

Black, P. E. (2004). *Ratcliff/Obershelp pattern recognition*. Dictionary of Algorithms and Data Structures. https://xlinux.nist.gov/dads/HTML/ratcliffObershelp.html

Caluwaerts, P. (2010). Towards a European electronic invoicing framework: Why businesses, service providers and consumers should switch to e-invoicing. *Journal of Payments Strategy & Systems*, *4*(3), 231–241.

Cao, L. (2017). Data science: Challenges and directions. *Communications of the ACM*, *60*(8), 59–68. https://doi.org/10.1145/3015456

Charniak, E. (1991). Bayesian Networks without Tears. *AI Mag.*, *12*, 50–63.

Chinchor, N., & Sundheim, B. (1993). MUC-5 Evaluation Metrics. *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993*. https://aclanthology.org/M93-1007

Chinnamgari, S. K. (2019). R Machine Learning Projects. In Packt Publishing (Ed.), *Journal of Chemical Information and Modeling*.

Chung, L., & do Prado Leite, J. C. S. (2009). On Non-Functional Requirements in Software Engineering. In A. T. Borgida, V. K. Chaudhri, P. Giorgini, & E. S. Yu (Eds.), *Conceptual Modeling: Foundations and Applications: Essays in Honor of John Mylopoulos* (pp. 363–379). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-02463-4_19

Cielen, D., Meysman, A. D. B., & Ali, M. (2016). *Introducing Data Science*. Manning Publications.

Claesen, M., & De Moor, B. (2015). *Hyperparameter Search in Machine Learning*. 10–14.

Cui, S., Asghar, M. R., & Russello, G. (2020). Multi-CDN: Towards Privacy in Content Delivery Networks. *IEEE Transactions on Dependable and Secure Computing*, *17*(5), 984–999. https://doi.org/10.1109/TDSC.2018.2833110

Daniel, J., & Martin, J. H. (2021). *Speech and Language Processing*.

de Abreu, D. M., Carvalho, I. F., Abelém, A. J. G., Menasché, D. S., Leão, R. M. M., & Silva, E. S. (2020). Seleção de Características por Clusterização para Melhorar a Detecção de Ataques de Rede. *Anais XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2020)*, 295–308. https://doi.org/10.5753/sbrc.2020.12290

de Lange, P., Nicolaescu, P., Neumann, A. T., & Klamma, R. (2020). Integrating Web-Based Collaborative Live Editing and Wireframing into a Model-Driven Web Engineering Process. *Data Science and Engineering*, *5*(3), 240–260. https://doi.org/10.1007/s41019-020-00131-3

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv Preprint ArXiv:1810.04805*.

Dj Novakovi, J., Veljovi, A., Ili, S. S., Zeljko Papi, ˇ, & Tomovi, M. (2017). Evaluation of Classification Models in Machine Learning. In *Theory and Applications of Mathematics & Computer Science* (Vol. 7, Issue 1).

Donahue, J., Hendricks, L. A., Rohrbach, M., Venugopalan, S., Guadarrama, S., Saenko, K., & Darrell, T. (2017). Long-Term Recurrent Convolutional Networks for Visual Recognition and Description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(4), 677–691. https://doi.org/10.1109/TPAMI.2016.2599174

Ekbal, A., & Bandyopadhyay, S. (2009). Named entity recognition using support vector machine: A Language independent approach. *World Academy of Science, Engineering and Technology*, *39*(June 2014), 548–563. https://doi.org/10.5281/zenodo.1057979

European Committee for Standardization. (2020). *CEN/TC 434 - Electronic Invoicing*.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian Network Classifiers. *Machine Learning*, *29*(2–3). https://doi.org/10.1023/a:1007465528199

Gartner, Krensky, P., den Hamer, P., Brethenoux, E., Hare, J., Idoine, C., Linden, A., Sicular, S., & Choudhary, F. (2020). Magic Quadrant for Data Science and

Machine Learning Platforms. In *Gartner* (Issue March). https://doi.org/G00467320

Ghani, Ts. Dr. M. F. M. (2008). Discretization of Continuous Valued Dimensions in OLAP Data Cubes. *JCSNS International Journal of Computer Science and Network Security*, *8*(11).

Glinz, M. (2007). On Non-Functional Requirements. *15th IEEE International Requirements Engineering Conference (RE 2007)*, 21–26. https://doi.org/10.1109/RE.2007.45

Haldar, R., & Mukhopadhyay, D. (2011). Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach. *ArXiv*, *abs/1101.1*(Ld).

Holowaychuk, T., Rajlich, N., Vagg, R., & Zaytsev, J. (2022). *canvas - npm*. https://www.npmjs.com/package/canvas

Huang, D. Z., Chen, D. K., He, D. J., Bai, D. X., Karatzas, D. D., Lu, D. S., & Jawahar, D. C. V. (2019). *ICDAR 2019 Robust Reading Challenge on Scanned Receipts OCR and Information Extraction*.

IEEE. (1990). IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*, 1–84. https://doi.org/10.1109/IEEESTD.1990.101064

International organization for standardization. (2011). *ISO - ISO/IEC 25010:2011 - Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. ISO/IEC 25010. https://www.iso.org/standard/35733.html

Jones, M., Bradley, J., & Sakimura, N. (2015). *Json web token (jwt)*.

Jyotsna, Chauhan, S., Sharma, E., & Doegar, A. (2016). Binarization techniques for degraded document images-A review. *2016 5th International Conference on Reliability, Infocom Technologies and Optimization, ICRITO 2016: Trends and Future Directions*, 163–166. https://doi.org/10.1109/ICRITO.2016.7784945

Khritankov, A. (2021). *Analysis of hidden feedback loops in continuous machine learning systems*. 1–7. https://doi.org/10.1007/978-3-030-65854-0_5

Koch, B. (2017). *Business Case E-Invoicing / E-Billing*.

Koch, B. (2019). *The e-invoicing journey 2019-2025* (Issue September).

Kotsiantis, S. B. and K. D. and P. P. E. (2006). Data preprocessing for supervised leaning. *International Journal of Computer Science*, *1*(2), 111–117.

Kotsiantis, S. B., Zaharakis, I. D., & Pintelas, P. E. (2006). Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review*, *26*(3), 159–190. https://doi.org/10.1007/s10462-007-9052-3

Kubat, M., Holte, R. C., Matwin, S., Kohavi, R., & Provost, F. (1998). *Machine Learning for the Detection of Oil Spills in Satellite Radar Images* (Vol. 30).

Kumar, V., & Minz, S. (2014). Feature Selection: A literature Review. *Smart Computing Review*, *4*(3).

Kuznetsov, V. (2019). *Data Science: state of the art*.

Labelbox. (2021). *Labelbox*. https://labelbox.com

Lakshminarayan, K., Harp, S. A., & Samad, T. (1999). Imputation of Missing Data in Industrial Databases. *Applied Intelligence*, *11*(3), 259–275. https://doi.org/10.1023/A:1008334909089

Lawton, G. (2008). Developing Software Online With Platform-as-a-Service Technology. *Computer*, *41*(6), 13–15. https://doi.org/10.1109/MC.2008.185

Lemstra, D. (2021). Magick.NET. *GitHub Repository*.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 707–710.

Ligeiro Marques, R., & Dutra, I. (n.d.). *Redes Bayesianas: o que são, para que servem, algoritmos e exemplos de aplicações*.

Liu, H. (2002). Discretization: An Enabling Technique. In *Data Mining and Knowledge Discovery* (Vol. 6).

Lorena, A. C., Lorena, A. C., & Carvalho, A. C. P. L. F. de. (2007). Uma Introdução às Support Vector Machines. *Revista de Informática Teórica e Aplicada*, *14*(2), 43–67. https://doi.org/10.22456/2175-2745.5690

Madhavan, R., Tunstel, E., & Messina, E. (2009). Performance evaluation and benchmarking of intelligent systems. In *Performance Evaluation and Benchmarking of Intelligent Systems*. https://doi.org/10.1007/978-1-4419-0492-8

Martin, R. C. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Pearson.

Microsoft. (2017). *Azure Form Recognizer* (V2.1.). Microsoft.

Microsoft. (2021a). Form OCR Testing Tool. *GitHub Repository*.

Microsoft. (2021b). *Introdução ao Armazenamento de blobs do Azure*. https://docs.microsoft.com/pt-pt/azure/storage/blobs/storage-blobs-introduction

Microsoft. (2022, April 13). *Overview - Azure App Service | Microsoft Docs*. Azure Product Documentation. https://docs.microsoft.com/en-us/azure/app-service/overview

Microsoft Corporation. (2009). *Microsoft Application Architecture Guide*.

MIT Critical Data. (2016). Secondary Analysis of Electronic Health Records. In *Secondary Analysis of Electronic Health Records*. Springer International Publishing. https://doi.org/10.1007/978-3-319-43742-2

Mitchell, T. (1997a). *Machine learning*.

Mitchell, T. (1997b). *Machine learning*.

Motoca, H., & Liu, H. (2002). *Feature Selection, Extraction and Construction*.

Nadeau, D., & Sekine, S. (2007). A Survey of Named Entity Recognition and Classification. *Lingvisticae Investigationes*, *30*(2). https://doi.org/10.1075/li.30.1.03nad

Nasar, Z., Jaffry, S. W., & Malik, M. K. (2021). Named Entity Recognition and Relation Extraction: State-of-The-Art. *ACM Computing Surveys*, *54*(1).

Newman, S. (2015a). Building Microservices. In *O'Reilly*.

Newman, S. (2015b). Building Microservices. In *O'Reilly*.

Nord, T. (2013). *What is a Feedback Loop?* 27 Juni 2013. https://blog.returnpath.com/what-is-a-feedback-loop/

NuGet Gallery | Ocelot 18.0.0. (n.d.). Retrieved August 3, 2022, from https://www.nuget.org/packages/Ocelot/

Oficial, P. T. J., Conselho, C. E. D. O., Europeia, C., Europeu, P., & Europeu, S. (2006). DIRECTIVA 2006/112/CE de 28 de Novembro de 2006. *Jornal Oficial Da União Europeia*, *OJ L 347*.

OpenVinoToolKit. (2018). Computer Vision Annotation Tool. *GitHub Repository*.

Osterwalder, A., & Pigneur, Y. (2010). *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers* (John Wiley & Sons, Ed.).

Pande, S. (2020). Understanding azure storage and azure storage explorer. *Journal Homepage: Http://Www. Ijmra. Us*, *10*(02).

Park, H. A. (2013). An introduction to logistic regression: From basic concepts to interpretation with particular attention to nursing domain. *Journal of Korean Academy of Nursing*, *43*(2), 154–164. https://doi.org/10.4040/jkan.2013.43.2.154

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel V. and Thirion, B., Grisel, O., Blondel, M., Prettenhofer P. and Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pei, S. C., & Lin, C. N. (1995). Image normalization for pattern recognition. *Image and Vision Computing*, *13*(10), 711–723. https://doi.org/10.1016/0262-8856(95)98753-G

Penttinen, E. (2008). *Electronic Invoicing Initiatives in Finland and in the European Union - Taking the steps towards the real-time economy*.

Pina, P. E., Donmez, B., & Cummings, M. L. (2008). Selecting metrics to evaluate human supervisory control applications. *HAL Lab: MIT Department of Aeronautics and Astronautics*, *May*, 1–94.

Poovizhi, P. (2014a). A Study on Preprocessing Techniques for the Character Recognition. *International Journal of Open Information Technologies*, *2*(12), 21–24.

Poovizhi, P. (2014b). A Study on Preprocessing Techniques for the Character Recognition. *International Journal of Open Information Technologies*, *2*(12), 21–24.

Saeys, Y., Inza, I., & Larranaga, P. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics*, *23*(19), 2507–2517. https://doi.org/10.1093/bioinformatics/btm344

Sandhu, R. S. (1998). *Role-based Access Control* (M. v Zelkowitz, Ed.; Vol. 46, pp. 237–286). Elsevier. https://doi.org/https://doi.org/10.1016/S0065-2458(08)60206-5

Sasaki, Y. (2007). *The truth of the F-measure*. https://www.cs.odu.edu/~mukka/cs795sum09dm/Lecturenotes/Day3/F-measure-YS-26Oct07.pdf

Segura-Bedmar, I. P., Herrero-Zazo, M., & Martínez, P. (2013). SemEval-2013 Task 9 : Extraction of Drug-Drug Interactions from Biomedical Texts (DDIExtraction 2013). *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, 341–350. https://aclanthology.org/S13-2056

Shaalan, K. (2014). A survey of arabic named entity recognition and classification. *Computational Linguistics*, *40*(2), 469–510.

Singh, A., Thakur, N., & Sharma, A. (2016). A review of supervised machine learning algorithms. *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, 1310–1315.

Strategyzer. (2022). *Business Model Canvas – Download the Official Template*. https://www.strategyzer.com/canvas/business-model-canvas

Swagger. (2020). *OpenAPI Specification*.

Tatbul, N., Lee, T. J., Zdonik, S., Alam, M., & Gottschlich, J. (2018). Precision and Recall for Time Series. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 31). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/8f468c873a32bb0619eaeb2050ba45d1-Paper.pdf

Thakur, R. S., Yadav, R. N., & Gupta, L. (2019). State-of-art analysis of image denoising methods using convolutional neural networks. *IET Image Processing*, *13*(13), 2367–2380. https://doi.org/10.1049/iet-ipr.2019.0157

The European Parliament And The Council Of The European Union. (2014). Official Journal of the European Union. *Official Journal of the European Union*, *53*(L 133).

Troelsen, A., & Japikse, P. (2017). *Pro C# 7, with .NET and .NET Core, Eight Edition*.

U.S. Department of Health & Human Services. (2022). *Data Analysis*. https://ori.hhs.gov/education/products/n_illinois_u/datamanagement/datopic.html

Vadapalli, S., & Safari, an O. M. Company. (2017). *Hands-on DevOps* (1st ed.). Packt.

Wang, L. (2017). An overview of internet-enabled cloud-based cyber manufacturing. In *Transactions of the Institute of Measurement and Control* (Vol. 39, Issue 4, pp. 388–397). SAGE Publications Ltd. https://doi.org/10.1177/0142331216687817

Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., & Zhou, M. (2020, August). LayoutLM: Pre-training of Text and Layout for Document Image Understanding. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery &amp Data Mining*. https://doi.org/10.1145/3394486.3403172

Yan, X., Zhang, C., & Zhang, S. (2003). Toward databases mining: pre-processing collected data. *Applied Artificial Intelligence*, *17*(5–6), 545–561. https://doi.org/10.1080/713827171

Zhang, S., Zhang, C., & Yang, Q. (2003). Data preparation for data mining. *Applied Artificial Intelligence*, *17*, 375–381. https://doi.org/10.1080/08839510390219264