INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




NST - Certificação Elétrica

TIAGO ANDRÉ GOMES DE SOUSA
Outubro de 2022

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

NST – Certificação Elétrica

Tiago André Gomes de Sousa

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Engenharia de Software

Orientador: Maria Goreti Marreiros

Porto, 15 de Outubro, 2022

Dedicatória

À minha família, amigos, colegas e professores, pelo apoio e incentivo demonstrado durante o meu percurso académico.

Resumo

Atualmente a nearSea Technologies opera diretamente com a HMC, líder no fornecimento de serviços de conformidade elétrica na Inglaterra. A HMC é uma empresa inglesa que atua no setor elétrico, garantindo que a infraestrutura elétrica das casas e outros edifícios funciona de forma segura e eficiente a toda a hora.

De momento, grande parte dos serviços prestados pela HMC são efetuados através de inspeções por parte dos seus engenheiros. Das inspeções resulta um documento formal, o certificado de instalação elétrica, que contém a descrição da instalação elétrica, resultados de testes efetuados à instalação, observações decorrentes e finalmente a aprovação ou não da instalação. Dada a elevada frequência de inspeções e o trabalho repetitivo e exaustivo necessário em cada uma delas, surgiu a necessidade de transformar e otimizar todo este processo.

De modo a atender a esta necessidade, a nearSea Technologies comprometeu-se a desenvolver diversas aplicações que permitam aos profissionais no terreno, efetuar todas as suas tarefas de forma mais eficiente e eficaz.

Dessa forma, surgiu a oportunidade de desenvolver o projeto atual, sendo que, o principal objetivo passa não só pela criação de uma nova plataforma, como por dotar as plataformas existentes de funcionalidades que permitam aos profissionais efetuar as suas tarefas do dia-a-dia. Dentro destas funcionalidades, destaca-se o desenvolvimento de uma API que suporte a importação e análise de certificados elétricos através de tecnologias e ferramentas de OCR.

Terminado o desenvolvimento do presente projeto, foi disponibilizada uma nova plataforma aos gestores da HMC, onde os mesmos eram capazes de efetuar a importação e respetiva validação de certificados emitidos em inspeções prévias através de ferramentas de OCR. Efetuada a importação dos mesmos, os engenheiros tinham a possibilidade de consultar o histórico de certificados importados e duplicar os mesmos, caso necessário, de forma a reusar a sua informação e acelerar todo o processo de inspeções.

A plataforma desenvolvida e as funcionalidades adicionadas aos serviços existentes foram avaliadas através do modelo *Quantitative Evaluation Framework* (QEF) e consideradas bem-sucedidas, dado que apresentaram uma dimensão de 95% na avaliação da sua qualidade.

Palavras-chave: Inspeção elétrica, Certificação elétrica, API, OCR

Abstract

Currently, nearSea Technologies works directly with HMC, the leading provider of electrical compliance services in the UK. HMC is an UK organization that works in the electrical sector, ensuring that the electric infrastructure of homes and other buildings is always working safely and efficiently.

Currently, most of the services provided by HMC are carried out in the form of inspections by its engineers. These inspections, result in a formal document, the electrical certificate, which contains the description of the electrical installation, the results and observations of the tests carried out on the installation and finally its approval or rejection. Given the high frequency of inspections and the repetitive and exhaustive work required in each of them, the need to transform and optimize this process arose.

To meet this need, nearSea Technologies committed to developing multiple applications that allow professionals in the field to carry out all their tasks more efficiently and effectively.

Therefore, appeared the opportunity to develop the current project, and the main objective is not to only create a new platform, but to also provide the existing platforms with features that allow professionals to solve the problems described above. Among these features, we highlight the development of an API that supports the import and analysis of electrical certificates through OCR technologies and tools.

After finishing the development of the current project, a new platform was provided to the HMC managers, where they could import and validate certificates emitted in previous inspections through OCR tools. After the certificates were imported, the engineers had the possibility to consult an history of the imported certificates and duplicate each of them, if necessary, to reuse their information and speed up the inspections process.

The developed platform and the functionalities added to the existing services were evaluated through the Quantitative Evaluation Framework (QEF) model and considered successful, given that it presented a dimension of 95% in the evaluation of its quality.

Keywords: Eletrical inspection, Eletrical certification, API, OCR

Agradecimentos

Mais do que um mero formalismo, expresso os meus sinceros agradecimentos àqueles que me incentivaram e apoiaram direta ou indiretamente nesta caminhada. Deixo o meu agradecimento, de uma forma mais particular:

- À Professora Doutora Maria Goreti Marreiros, orientadora da faculdade, pela disponibilidade demonstrada ao longo do desenvolvimento do projeto e pelas opiniões e conselhos enriquecedores que contribuíram para a realização deste relatório.

- Ao Engenheiro Pedro Pinho, supervisor na organização, pela oportunidade de realizar o presente projeto e pela confiança depositada em mim desde o primeiro dia.

- À Susana, *Product Owner* da equipa integrada durante o desenvolvimento do projeto, pelo apoio e disponibilidade demonstrados, tanto no desenvolvimento da solução como do presente relatório.

- Aos restantes colaboradores da nearSea Technologies, pela forma como me acolheram desde o primeiro dia e pela ajuda e partilha de conhecimento ao longo do projeto.

- Aos docentes no ISEP, que ao longo do meu percurso académico me transmitiram o conhecimento e competências necessárias para o meu sucesso.

- A todos os meus amigos e família pela contribuição para o meu desenvolvimento enquanto pessoal e profissional.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Excertos de Código

Lista de acrónimos

API	*Application Programming Interface*
AWS	Amazon Web Services
CI / CD	*Continuous Integration* / *Continuous Deployment*
FAST	*Function Analysis System Technique*
GCP	Google Cloud Platform
HCL	*HashiCorp Configuration Language*
HTTP	*Hypertext Transfer Protocol*
JSON	JavaScript *Object Notation*
MNIST	*Modified National Institute of Standards and Technology database*
NPM	*Node Package Manager*
OCR	*Optical Character Recognition*
ODM	*Object Data Modeling*
QEF	*Quantitative Evaluation Framework*
REST	*Representational state transfer*
SDK	*Software Development Kit*
UI	*User Interface*

Glossário

Backend	Camada de acesso aos dados de um sistema de um sistema de *software*
Backlog	Conjunto de tarefas a realizar durante o desenvolvimento de um produto
Browser	Aplicação de *software* utilizada para aceder à *World Wide Web*
DevOps	Conjunto de práticas que visa automatizar os processos de desenvolvimento de *software*
Framework	Biblioteca de *software*
Frontend	Camada de apresentação dos dados de um sistema de *software*
Full Stack	Envolve todas as camadas de um sistema de software, tanto *frontend* como *backend*
Javascript	Linguagem de programação frequentemente utilizada em *browsers web*
Node.js	Interpretador de código Javascript
Open Source	Licenciamento livre
Pipeline	Fluxo de dados composto por diversos componentes, em que cada um deles possui uma tarefa específica
Python	Linguagem de programação de alto nível
Scrum	Metodologia ágil de *software*.
Software	Conjunto de dados, programas e instruções utilizados para a execução de tarefas de um computador.
Stakeholders	Pessoas ou organizações impactadas pelas ações de um determinado projeto ou empresa (partes interessadas).
Typescript	Linguagem de programação baseada em Javascript.

1 Introdução

No âmbito da unidade curricular Tese/Dissertação/Estágio (TMDEI) do Mestrado de Engenharia de Software no ramo de Engenharia Informática, foi elaborado o presente documento. Este pretende descrever as atividades desenvolvidas durante o projeto e transparecer o processo de aprendizagem no seu decurso, analisando de forma crítica as tarefas desenvolvidas e a solução implementada.

Este capítulo pretende de forma resumida descrever e contextualizar o problema que se pretende resolver com este projeto. Além disso serão evidenciados os objetivos, tal como a abordagem a seguir para alcançar os mesmos. Por último, será apresentada a estrutura do presente documento.

1.1 Apresentação da organização

O presente projeto foi desenvolvido na nearSea Technologies, organização fundada em 2020 e especializada no desenvolvimento de software Mobile (iOS e Android), Frontend e Backend.

A nearSea Technologies opera atualmente em múltiplas áreas como Ambiente, Educação e Saúde desenvolvendo soluções e projetos inovadores.

Dada a elevada qualidade dos serviços que fornece, a nearSea Technologies possui diversos clientes, tanto a nível nacional como internacional, em países como Reino Unido, Estados Unidos e Alemanha.

1.2 Contexto

A nearSea Technologies opera diretamente com a HMC Compliance, líder na prestação de serviços de conformidade elétrica na Inglaterra. A HMC Compliance é a atual líder na prestação de serviços de conformidade elétrica na Inglaterra, garantindo que a infraestrutura elétrica das casas e outros edifícios funciona de forma segura e eficiente a toda a hora. Os serviços prestados pelos seus engenheiros, são efetuados de acordo com os protocolos e

regulações estabelecidas, sendo emitido um certificado elétrico no final de cada inspeção, a comprovar que o edifício se encontra seguro.

Numa era digital, cada vez mais o uso de plataformas digitais faz parte do dia-a-dia, permitindo otimizar os processos de negócio ao eliminar a necessidade de gerir e colecionar dados através de papel (Hoos et al., 2015). De momento, os serviços prestados pelos engenheiros da HMC são efetuados no terreno através de inspeções elétricas. Dado que todos os certificados são atualmente emitidos em papel, torna-se complicado efetuar a sua gestão devido à elevada quantidade de informação necessária para emitir cada um deles.

De modo a atender a esta necessidade, a nearSea Technologies comprometeu-se a desenvolver diversas aplicações que permitam aos profissionais no terreno, efetuar todas as suas tarefas de forma mais eficiente e eficaz. A Figura 1 representa a arquitetura já existente e desenvolvida pela nearSea Technologies, de modo a digitalizar e acelerar todo o processo relativo às inspeções efetuadas pelos engenheiros.

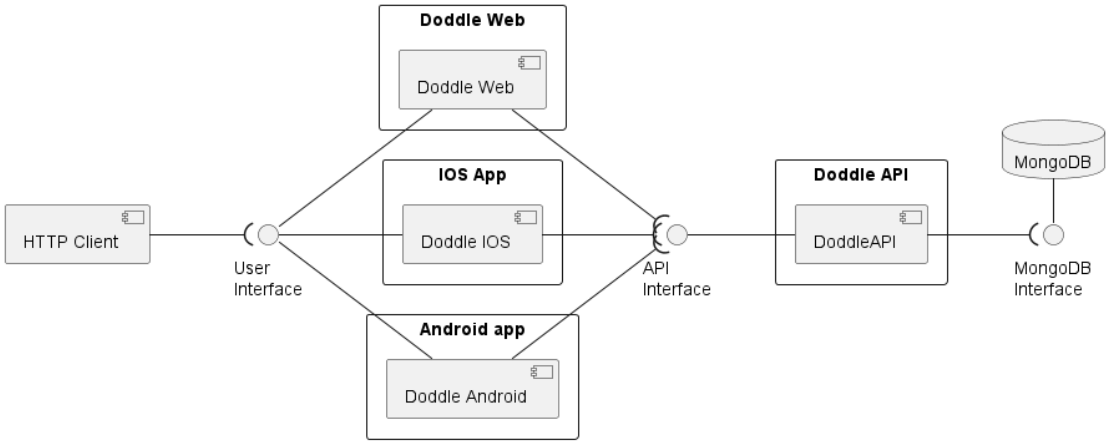


Figura 1 – Arquitetura já existente do projeto

As aplicações permitem aos engenheiros adicionar a empresa e clientes para os quais trabalham e posteriormente efetuar a criação e edição de certificados, através de formulários adaptados a cada tipo de certificado elétrico.

1.3 Problema

De modo a garantir a segurança de pessoas e bens, os edifícios novos e os já existentes são inspecionados sob várias vertentes, sendo uma delas a inspeção da instalação elétrica. Da inspeção decorre uma imagem da instalação elétrica que existe em todo o edifício, seja este uma casa - que comporta um número baixo de quadros elétricos e, portanto, a complexidade da instalação é baixa - ou uma fábrica onde o número de quadros é elevado e ramifica-se por vários outros quadros, gerando desta forma uma rede complexa. Esta inspeção é feita com uma periodicidade regular, que é determinada tanto pela idade do edifício como pelas observações decorrentes de inspeções anteriores. Da inspeção resulta um documento formal, o certificado de instalação elétrica (EICR) que contém a descrição da instalação elétrica, resultados de testes efetuados à instalação, observações decorrentes e finalmente a aprovação ou não da instalação.

Contudo, estas inspeções são efetuadas periodicamente de modo a renovar as certificações emitidas pelos engenheiros, fazendo com que o volume de informação seja bastante alto e a sua gestão seja difícil, nomeadamente a procura de histórico.

De forma a apoiar os profissionais no terreno, torna-se essencial obter a informação de inspeções anteriores, nomeadamente os certificados, em formato digital de modo a reutilizar essa informação, facilitando e melhorando o processo de inspeção. O acesso fácil e rápido a certificados anteriores, permite um melhor entendimento da instalação existente e o fácil acesso a observações de inspeções anteriores. Estas funcionalidades apoiam na deteção de situações potencialmente perigosas, algo que de momento se encontra limitado, dada a dificuldade ao acesso a estes dados de histórico.

1.4 Objetivos

De modo a solucionar o problema descrito anteriormente, surgiu a necessidade de melhorar o processo de inspeção dos engenheiros através do desenvolvimento de um sistema que permitisse importar certificados emitidos em inspeções prévias. Para isso serão utilizadas tecnologias de *Optical character recognition* para efetuar a importação e posteriormente disponibilizar os certificados automaticamente preenchidos aos engenheiros para consulta ou para operações como duplicação dos mesmos, de modo a reutilizar a informação preenchida previamente.

O principal objetivo do presente projeto, passa pelo desenvolvimento de uma nova plataforma que permita a importação de certificados previamente emitidos utilizando tecnologias de OCR. Além disso, serão adicionadas novas funcionalidades às aplicações atualmente utilizadas pelos engenheiros, com o intuito de permitir reutilizar a informação dos certificados importados nas suas inspeções.

De forma a implementar o sistema descrito anteriormente, foram identificados diversos objetivos a cumprir com o intuito de atingir o mesmo:

- Analisar abordagens existentes na aplicação de ferramentas e tecnologias de OCR.
- Redesenhar arquitetura existente, de modo a suportar o sistema e funcionalidades descritas anteriormente.
- Desenvolver uma API capaz de analisar e extrair informação de certificados elétricos utilizando tecnologias e ferramentas de OCR, assim como capaz de mapear a informação extraída para preencher novos certificados automaticamente.
- Desenvolvimento de um Portal Web que permita aos gestores da HMC efetuar a importação e gestão de certificados. Adicionar novas funcionalidades às aplicações existentes, que permitam acelerar o processo de inspeção dos engenheiros.
- Documentar e testar solução desenvolvida através de ferramentas apropriadas.
- Disponibilizar a solução desenvolvida ao cliente final num ambiente de produção.

1.5 Metodologia de trabalho

Na secção que se segue, é descrita a estrutura e funcionamento das equipas na nearSea Technologies, tal como as tecnologias e metodologias de desenvolvimento utilizadas na elaboração do projeto.

Ao longo da elaboração do projeto, o aluno teve a oportunidade de integrar uma das equipas de desenvolvimento da organização. Esta equipa encontrava-se dividida em *feature teams*, compostas por programadores e gestores de projeto, cuja responsabilidade, foi definida de acordo com as competências dos seus elementos. De modo a atuarem de forma eficiente e produtiva, o *software* desenvolvido pela equipa é realizado aplicando os conceitos e ideais da metodologia de trabalho ágil Scrum.

O aluno teve a oportunidade de participar em diferentes cerimónias Scrum em conjunto com a equipa integrada, das quais se destacam:

- **Reuniões diárias -** destinadas a apurar o que cada elemento da equipa fez no dia anterior para atingir o objetivo do *Sprint*, informando qualquer problema que possa ter sugerido durante o desenvolvimento.

- **Planeamento do *Sprint* –** reunião onde são definidas todas as tarefas a desenvolver durante o *Sprint*.

- **Refinamento do *Backlog* do produto –** reunião onde se revê o *backlog* e esclarecem dúvidas sobre o que se pretende atingir com cada tarefa e se descreve como vai ser executada pela equipa de desenvolvimento, as quais se encontram em espera para ser integradas em futuros *Sprints*.

- ***Sprint Review* –** destinada à apresentação do trabalho efetuado ao longo do *Sprint* aos *stakeholders* e restantes elementos da equipa.

1.6 Estrutura do Documento

De momento o documento encontra-se organizado em cinco capítulos com temas distintos e compartimentados.

Capítulo 1 – Introdução, enquadramento do projeto e respetivos objetivos, bem como, uma abordagem do problema em estudo e o seu contributo para ambos a organização e o aluno.

Capítulo 2 – Estado da arte, são apresentadas soluções que incluem funcionalidades de OCR e apresentam semelhanças à solução a implementar, bem como, as tecnologias a utilizar no desenvolvimento da mesma.

Capítulo 3 – Análise de Valor, é efetuada a análise de valor do projeto, identificando o cliente e detalhando o processo de inovação e oportunidade do projeto.

Capítulo 4 – Análise e Desenho, descrição do desenho geral da solução, tal como das propostas de arquitetura a adotar para o desenvolvimento da mesma.

Capítulo 5 – Implementação, é demonstrada a implementação da solução desenhada, descrevendo todos os serviços desenvolvidos e respetivas funcionalidades, assim como a comunicação efetuada entre eles.

Capítulo 6 – Experimentação e Avaliação, são identificados os indicadores de qualidade da solução desenvolvida e posteriormente utilizados para determinar o sucesso da solução desenvolvida, através do uso de ferramentas adequadas para a sua avaliação.

Capítulo 7 – Conclusões, apresenta os objetivos atingidos do presente projeto e menciona as suas limitações, assim como as possibilidades de trabalho futuro a desenvolver.

2 Estado de Arte

A principal área de conhecimento a destacar neste capítulo é o OCR, no que diz respeito à tecnologia a utilizar para a implementação da solução. Por outro lado, importa também encontrar ferramentas e *frameworks* existentes, que permitam utilizar esta tecnologia e converter o conteúdo dos certificados elétricos importados pelos engenheiros.

Neste capítulo são também referidas as linguagens e tecnologias utilizadas ao longo do projeto, tal como uma comparação das mesmas com tecnologias semelhantes que poderiam também ser adotadas.

2.1 Optical Character Recognition

Optical Character Recognition, ou OCR, é uma tecnologia que permite que a partir de uma imagem, se consiga reconhecer os carateres nela presentes. Isto possibilita a conversão de documentos, em dados que podem ser analisados e editados pelos utilizadores através de aplicações em diferentes plataformas. A imagem obtida é transformada em conteúdo legível, similar ao conteúdo no documento original (Indravadanbhai Patel et al., 2012).

2.1.1 A evolução

Ao longo dos anos, a extração de dados de documentos digitalizados, como PDFs e imagens sempre foi efetuada manualmente por parte das organizações. De modo a revolucionar este processo, foi inventada a OCR ou *Optical Character Recognition*, tecnologia que surgiu por volta de 1950 nas primeiras aplicações (Rufenacht, 2020).

Dado o grande sucesso desta tecnologia, surgiu a necessidade de aplicá-la em diversas situações. Dessa forma, surgiu o Optacon, criado em 1954, foi o primeiro dispositivo a ser utilizado numa empresa de jornalismo denominada de *American Magazine Reader's Digest* (Figura 2). O dispositivo permitia converter relatórios de vendas escritos manualmente em dados que podiam ser lidos e analisados pelos computadores. Contudo, não era ainda possível extrair os dados dos documentos (Rufenacht, 2020).

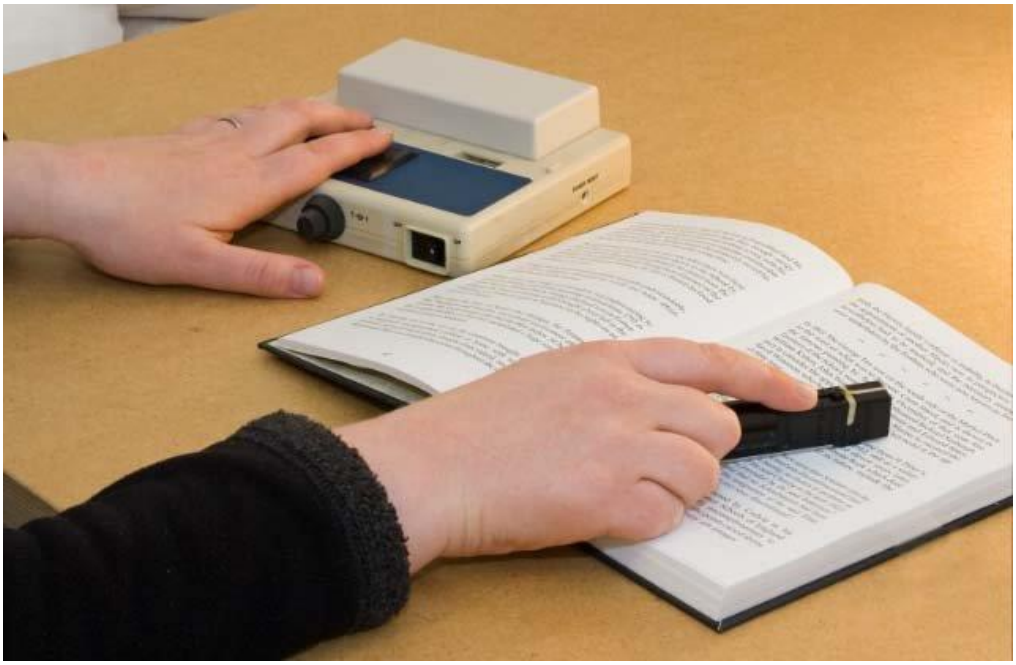


Figura 2 – Optacon, primeiro dispositivo OCR criado (Brewster, 2008)

Ao longo dos anos, começaram a surgir os primeiros sistemas que permitiam processar documentos e identificar diversos símbolos e fontes presentes nos mesmos. Contudo, essa identificação era baseada em *Template matching*, ou seja, os dispositivos eram configurados e treinados para ler secções predefinidas dos documentos. Isto significava que grande parte do trabalho necessitava de intervenção humana para efetuar o processamento com sucesso (Rufenacht, 2020).

Contudo, por volta de 2013 ocorreu uma revolução na aplicação desta tecnologia, sendo a mesma combinada com *machine learning.* Esta combinação foi possível, devido à criação da base de dados MNIST, que contem inúmeras letras e palavras escritas manualmente, permitindo treinar os algoritmos para a deteção das mesmas. Dessa forma, todo o processo de leitura e análise de documentos tornou-se completamente automatizado.

2.1.2 Machine Learning-based OCR

A análise OCR baseada em *machine learning*, é uma técnica de análise OCR, que permite efetuar o processamento e análise de documentos de modo completamente automatizado. Isto é feito treinando os algoritmos para ler inúmeros caracteres e fontes diferentes, permitindo detetar os mesmos em qualquer secção dos documentos. Este treino é efetuado através de bases de dados que possuem grandes quantidades de possíveis caracteres presentes nos documentos (Indravadanbhai Patel et al., 2012).

De forma a evoluir o algoritmo, é implementado um sistema de validação das análises, efetuado com base em intervenção humana. Este sistema, permite ao algoritmo distinguir as análises corretas das incorretas, evoluindo ao longo do processo. Esta técnica possui diversas vantagens na análise de documentos, face a outras técnicas de OCR como:

- Maior flexibilidade.
- Maior percentagem de sucesso.
- Maior desempenho.
- Menor número de recursos necessários.

2.1.3 Template-based OCR

Template Matching é uma técnica de OCR que requer intervenção humana para o processamento e análise de documentos. Inicialmente é feita uma configuração baseado no documento a ser processado, identificando cada um dos campos a ser analisado e extraído (Srihari et al., 2003).

Apesar de permitir efetuar a análise de documentos após ser configurado, esta técnica possui algumas desvantagens face ao OCR baseado em *machine learning:*

- Necessita de intervenção humana para cada documento analisado.
- Requere um elevado número de recursos para configurar cada documento.

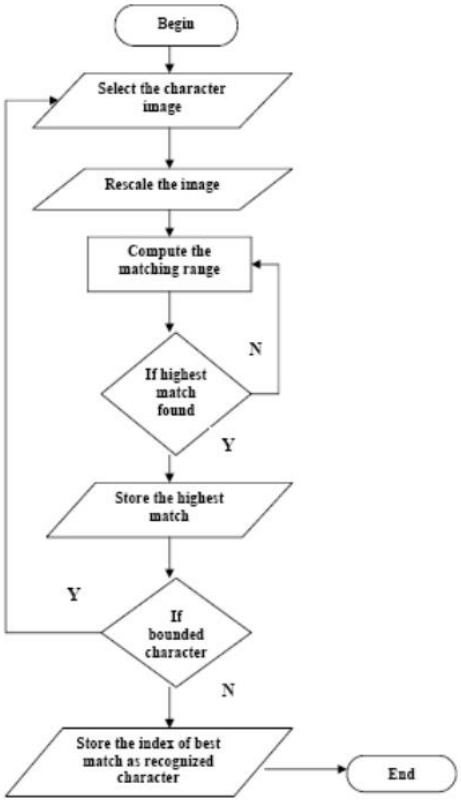


Figura 3 – Análise baseada em *Template Matching* (Md Anwar Hossain et al., 2019)

2.2 Ferramentas e Tecnologias de OCR

Nesta secção é feita uma análise às diferentes ferramentas e tecnologias de OCR, de modo a considerar a melhor opção para a implementação da solução. Inicialmente é feita uma descrição das mesmas, de modo a posteriormente comparar as vantagens provenientes do seu uso.

2.2.1 Tesseract

Originalmente desenvolvido pela HP (Hewlett Packard) entre 1985 e 1994 e mantido pela Google desde 2006, esta ferramenta é atualmente *open source* (sob licença Apache 2.0) e mantém-se uma das melhores soluções gratuitas de OCR, tanto em relação ao número de línguas que é capaz de reconhecer como relativamente à precisão dos seus resultados.

Esta *framework* mantém-se relevante até hoje e continua a ser uma das ferramentas de OCR mais utilizadas, tendo a sua última versão estável (5.0.1) sido lançada recentemente em 07/01/2022 (Tesseract, 2022).

Dado que é uma ferramenta de código aberto com grande precisão de resultados e suporta a maioria das linguagens de programação, o Tesseract foi uma das ferramentas consideradas para o desenvolvimento da solução proposta.

2.2.2 Amazon *Textract*

Desenvolvido e lançado pela Amazon em 2018, o Amazon *Textract* é um serviço de *machine learning* que extrai automaticamente texto, tabelas e formulários de documentos digitalizados. Além do simples uso de OCR para identificar, manipular e extrair dados de formulários e tabelas, o *Textract* usa *machine learning* para ler e processar qualquer tipo de documento, extraindo com precisão texto, tabelas e outros dados sem esforço manual.

Além disso, em conjunto com o *Amazon Augmented AI*, permite adicionar análises humanas para fornecer supervisão acerca dos modelos analisados e verificar dados confidenciais (Amazon Web Services, 2022c).

Apesar de ser uma excelente ferramenta de OCR, permitindo analisar qualquer tipo de documento com grande precisão, o Textract tem um custo associado ao seu uso, algo que influência a sua escolha para a implementação da solução proposta.

2.2.3 Apache PDFBox

A *Apache PDFBox* é uma ferramenta de código aberto desenvolvida por Ben Litchfield em 2002. Esta ferramenta é apenas destinada para uso com documentos PDFs, permitindo efetuar inúmeras ações nos mesmos, como, criação de novos documentos, manipulação de documentos existentes, extração de conteúdo e impressão dos mesmos (The Apache Software Foundation, 2021).

No entanto possui uma grande limitação quanto ao seu uso, dado que apenas suporta a linguagem de programação Java. Dessa forma, torna-se numa opção pouco viável para o contexto do projeto, dado que, Node.js é a plataforma de desenvolvimento a ser utilizada.

2.2.4 Cloud Vision API

A Cloud Vision API é um serviço criado pela Google e disponível na GCP. Este serviço possui diversas funcionalidades como, reconhecimento facial e de objetos, extração de texto de imagens utilizando OCR e até identificação de conteúdo explicito. Para isso, o serviço possui um *Software Development Kit,* capaz de ser integrado com diversas linguagens de programação como, Python, Java ou Typescript. Através do SDK é possível utilizar qualquer uma das funcionalidades descritas anteriormente (Google Cloud Platform, 2022a).

Visto que esta ferramenta possui uma grande capacidade de análise e extração de texto de documentos através de tecnologias de OCR e suporta a maioria das linguagens de programação, foi considerada como uma das ferramentas a adotar na implementação da solução.

2.2.5 Comparação das tecnologias

A nível de ferramentas e tecnologias de OCR, é possível concluir que existem diversas opções que podem ser adotadas, possuindo vantagens e desvantagens. A nível de implementação, a Apache PDFBox é a única que apresenta restrições quanto à linguagem de programação a utilizar, neste caso Java. Por outro lado, a Amazon Textract apesar da sua grande precisão de análise e suporte de uma grande variedade de documentos, necessita de uma subscrição para a sua utilização. Na Tabela 1 encontra-se uma comparação das diferentes tecnologias de OCR identificadas.

Tabela 1 - Comparação das tecnologias de OCR

	Tesseract	**Amazon Textract**	**Apache PDFBox**	**Cloud Vision API**
Análise rápida e eficaz	X	✓	X	✓
Análise demorada e eficaz	✓	X	✓	X
Mapeamento por coordenadas dos dados extraídos.	X	✓	X	✓
Open-source	✓	X	✓	X
Baixo custo de utilização	✓	X	✓	✓
Compatível com Typescript e Node.JS	✓	✓	X	✓

Dado que no contexto do projeto a desenvolver o serviço de computação em nuvem a utilizar é a Digital Ocean, o Amazon Textract acaba por ser descartado como opção para o desenvolvimento da solução.

Atualmente grande parte dos serviços desenvolvidos pela nearSea Technologies encontram-se implementados em Typescript e Node.js. Dessa forma, apenas o Tesseract e a Cloud Vision API são as únicas opções viáveis para o desenvolvimento do projeto. Após investigar ambas as tecnologias, concluiu-se que a Cloud Vision API era uma tecnologia bem mais completa e robusta para a implementação da solução, dado que permite o mapeamento da informação extraída por coordenadas e o seu tempo de análise é bastante rápido, como detalhado na Tabela 1. Dessa forma, a tecnologia escolhida para o desenvolvimento da solução foi a Cloud Vision API.

2.3 Linguagens e plataformas de desenvolvimento web

Dado o considerável aumento do uso da *Internet* nos últimos anos, a aposta em tecnologias que permitam agilizar o desenvolvimento *web* tem crescido imenso (Ella, 2019).

Na secção que segue, é feita uma análise de algumas das mais populares linguagens e plataformas de desenvolvimento web. Todas as opções consideradas possuem vantagens, dependendo do caso em que são aplicadas. Dessa forma, realizou-se uma comparação entre as mesmas, de modo a decidir qual a linguagem ou plataforma mais adequada para a solução a implementar.

2.3.1 Node.js

Node.js é uma plataforma de desenvolvimento *open source* que permite construir aplicações *web* escaláveis de alta performance utilizando Typescript como sintaxe. Esta plataforma foi construída sobre o motor V8, o qual interpreta JavaScript e foi criado pela Google com o propósito de ser usado no seu navegador, o Google Chrome. Node.js foi uma grande melhoria para este motor, pois permitiu que o Typescript fosse utilizado não só no *browser,* mas também nos servidores (Sufiyan, 2021)

Através desta plataforma, é possível desenvolver *websites* ou aplicações, permitindo, manipular milhares de conexões simultâneas, numa única máquina física, tornando-o na plataforma ideal para quem necessite de suportar grandes cargas e imensos usuários simultâneos (Kaneriya, 2020).

Um belo exemplo disso, foi o Walmart que na *black-friday* em 2013, resolveu tratar todo o tráfego móvel através de Node.js. Os servidores não passaram de 1% de utilização de CPU, mesmo com mais de 200 milhões de utilizadores *online* (Glozic, 2014).

Além disso, o Node.js utiliza o NPM como gestor de pacotes e bibliotecas, que por sua vez é o maior ecossistema de bibliotecas *open source* do mundo. No entanto, possui algumas vulnerabilidades como as dependências existentes entre os diferentes pacotes. Caso algum pacote frequentemente utilizado deixe de ser suportado e seja eliminado, ou apresente algum tipo de vulnerabilidade, todos os pacotes dependentes do mesmo são afetados.

2.3.2 Python

Python é uma linguagem de programação criada por Guido van Rossum, com o objetivo de ser produtiva e legível. Python é considerada uma das linguagens mais fáceis de aprender. Tal como o Node.js, Python possui uma livraria de módulos, prontos para realizar qualquer tipo de tarefa, tornando-a uma linguagem bastante útil para usar em qualquer tipo de projetos, maioritariamente em *data*, *machine learning* e áreas da educação e ciência (Madeira, 2020)

Em termos de desenvolvimento *web*, Python possui diversas *frameworks*, sendo que umas das mais utilizadas é Django, a qual é *open source* e segue o padrão MVC*, Model-View-Controller.*

2.3.3 Java

Java é uma linguagem de programação baseada em classes e orientada a objetos desenvolvida pela Sun Microsystems em 1995 e posteriormente adquirida pela Oracle em 2009. Java é uma das linguagens mais utilizadas mundialmente por diversos programadores e organizações, dadas as suas capacidades de segurança e diversas ferramentas e bibliotecas disponíveis para o desenvolvimento de aplicações (Hartman, 2022).

Além disso, é uma linguagem de fácil aprendizagem sendo frequentemente utilizada para o desenvolvimento de aplicações Mobile e APIs. Dessa forma, foi uma das linguagens consideradas para o desenvolvimento do projeto, visto que é compatível com as ferramentas e tecnologias de OCR identificadas na secção 2.2.

2.3.4 Comparação das tecnologias

Comparando estas tecnologias para desenvolvimento *web*, conclui-se que Python é uma boa opção para projetos de pequena a média dimensão, devido à sua capacidade de desenvolver soluções bastante rápidas e efetuar um elevado número de mudanças nas mesmas.

Contudo, em termos de performance, Node.js é uma boa opção, pois permite criar soluções onde existem elevados números de conexões, devido principalmente a ser uma plataforma assíncrona. Além disso, Node.js permite usar Typescript como linguagem *Fullstack*, ou seja, todos os serviços desenvolvidos são efetuados com a mesma linguagem. Este é um dos principais motivos pelo qual o Node.js e Typescript têm vindo a ser favoritos entre os programadores *web* (Luchaninov, 2021).

Por outro lado, Java permite desenvolver APIs seguras e complexas através do elevado número de ferramentas e bibliotecas que suporta durante o desenvolvimento de aplicações.

Na Tabela 2 é efetuada uma comparação das possíveis tecnologias a utilizar na implementação da solução.

Tabela 2 - Comparação das linguagens de programação

	Node.js + Typescript	**Python**	**Java**
Compatível com tecnologias de OCR	✓	✓	✓
Possui *frameworks* para desenvolvimento de APIs	✓	✓	✓
Possui ferramentas de desenvolvimento de alta qualidade	✓	✓	✓
Open-source	✓	✓	✓
Linguagem *Full Stack*	✓	✓	X
Tecnologia existente na organização.	✓	X	X

No contexto do projeto a desenvolver grande parte das equipas e plataformas existentes na nearSea Technologies, utilizam Node.js e Typescript. Além disso, o objetivo deste projeto passa não só por desenvolver uma nova API, mas também por dotar os serviços existentes de novas funcionalidades. Dessa forma, Node.js acaba por ser a tecnologia ideal para a solução a desenvolver, visto que também possui compatibilidade com as tecnologias de OCR mencionadas anteriormente, como o Tesseract e a Cloud Vision API.

2.4 Serviços de Computação em Nuvem

Os serviços de computação em nuvem revolucionaram o mundo da tecnologia, oferecendo o acesso a software e serviços através da *Internet.* Esta revolução foi especialmente útil para a implantação de serviços por parte das organizações, permitindo poupar imenso em recursos de *hardware* e manutenção do mesmo, para as suas infraestruturas (Fryer, 2022).

Na secção que segue, é feita uma análise a diversos serviços em nuvem que permitem a implantação de aplicações. Todas as opções consideradas possuem vantagens, dependendo do caso em que são aplicadas. Dessa forma, realizou-se uma comparação entre as mesmas, de modo a decidir qual a mais adequada para a solução a implementar.

2.4.1 Amazon Web Services

Amazon Web Services (AWS), é uma das plataformas em nuvem mais reconhecida e utilizada a nível mundial, fornecendo uma enorme variedade de serviços de computação em nuvem aos seus utilizadores.

A nível geográfico, a AWS suporta diversas regiões e continentes, entre eles, Europa, América, Ásia e Austrália. Isto permite que grande parte dos seus clientes consiga utilizar os serviços de forma eficiente e sem grande latência associada, aumentando a *performance* das aplicações implantadas pelos clientes (Amazon Web Services, 2022a).

Além dos serviços de implantação de aplicações, a AWS permite também a criação e manutenção de bases de dados relacionais e não relacionais em nuvem, aliviando os custos associados às mesmas para as organizações (SpinupWP, 2020)

Dado que no contexto do projeto a desenvolver, o principal objetivo é desenvolver um serviço que possua funcionalidades de OCR e disponibilizá-lo na Internet a AWS é uma escolha a considerar, visto possuir serviços como o *Amazon EC2* e o Amazon *Textract*

O *Amazon EC2* permite implantar as aplicações desenvolvidas, sem qualquer tipo de manutenção ou preocupação a nível de escalabilidade, sendo uma mais-valia para a solução a apresentar (Amazon Web Services, 2022b).

2.4.2 Google Cloud Platform

Google Cloud Platform (GCP), em semelhança à AWS, é uma plataforma oferecida pela Google, que fornece serviços em nuvem como armazenamento de dados ou hospedagem de aplicações e bases de dados.

Em termos geográficos, a GCP abrange diversas regiões da Europa, América, Ásia e Austrália, sendo agora um aspeto padrão para as grandes plataformas prestadores de serviços em nuvem. (Google Cloud Platform, 2022b)

A nível de hospedagem de aplicações, a GCP possui integração com Kubernetes e Docker, algo que facilita bastante a gestão e manutenção das mesmas. A fácil integração com estes dois serviços, acaba por ser uma grande vantagem para programadores que não possuam um forte conhecimento de DevOps e infraestrutura. (Google Cloud Platform, 2022c)

Contudo, uma das desvantagens da GCP, é o preço dos serviços que fornece em comparação com outras plataformas (Google Cloud Platform, 2022d). Apesar da maior qualidade que certos serviços oferecem, acaba por ser um ponto negativo para certas organizações e projetos de menor dimensão, que não possuam a capacidade para suportar os mesmos.

2.4.3 Digital Ocean

Fundada em 2011, a DigitalOcean é uma plataforma prestadora de serviços em nuvem. No entanto o seu foco, difere das grandes plataformas que atualmente existem no mercado, como AWS e GCP (SpinupWP, 2020). O seu principal objetivo é atender às necessidades dos programadores, facilitando a implantação dos seus serviços, através de interfaces simples e minimalistas, como uma extensa documentação acerca da sua utilização (DigitalOcean, 2022).

A nível geográfico, a DigitalOcean possui uma capacidade inferior às plataformas como AWS e GCP, dado que suportar um menor número de regiões em cada continente (DigitalOcean, 2021).

Relativamente a outras plataformas, a DigitalOcean possui preços bastantes acessíveis em grande parte dos serviços disponibilizados. No entanto, estes apenas se destinam a aplicações de pequena dimensão, dada a sua capacidade de escalabilidade (DigitalOcean, 2022). Além disso, as máquinas disponibilizadas, não possuem GPUs, algo essencial para as áreas de *machine learning* e *data science* (Taylor, 2022).

2.4.4 Comparação das plataformas

Em termos de plataformas baseadas em nuvem, é possível concluir que existem diversas soluções que podem ser adotadas pelas organizações. Cada uma delas possui as suas vantagens e desvantagens e a sua escolha deve ser baseada nas mesmas. Por um lado, a AWS e a GCP possuem uma enorme quantidade de serviços capazes de suportar um elevado número de utilizadores. Por outro lado, a DigitalOcean permite configurar e integrar os seus serviços de forma rápida e eficaz, quando se trata de aplicações de pequena dimensão. A Tabela 3, apresenta uma comparação dos diferentes serviços de computação em nuvem identificados paro o desenvolvimento da solução.

Tabela 3 - Comparação dos serviços de computação em nuvem

	AWS	**GCP**	**Digital Ocean**
Grande diversidade de serviços	✓	✓	✓
Grande complexidade na configuração da infraestrutura	✓	✓	X
Custo elevado	✓	✓	X
Utilizado nos serviços da organização	X	✓	✓

No contexto da solução a desenvolver, a escolha mais adequada foi a Digital Ocean, dado que, possui os serviços necessários à implementação da mesma e o seu custo é bastante reduzido comparado com os restantes serviços. Além disso, grande parte da infraestrutura da organização já se encontra hospedada na mesma.

2.5 Trabalhos Relacionados

Neste subcapítulo são abordadas diversas soluções que partilham semelhanças a nível de tecnologias utilizadas e funcionalidades a implementar, com o projeto proposto.

2.5.1 Adobe Acrobat Pro

O Adobe Acrobat Pro é um serviço *online* que permite converter arquivos PDF em documentos editáveis do Word ou Excel. Este serviço, oferece suporte para OCR ao efetuar as conversões dos documentos, permitindo aos utilizadores analisar e editar o texto dos mesmos (Adobe, 2021).

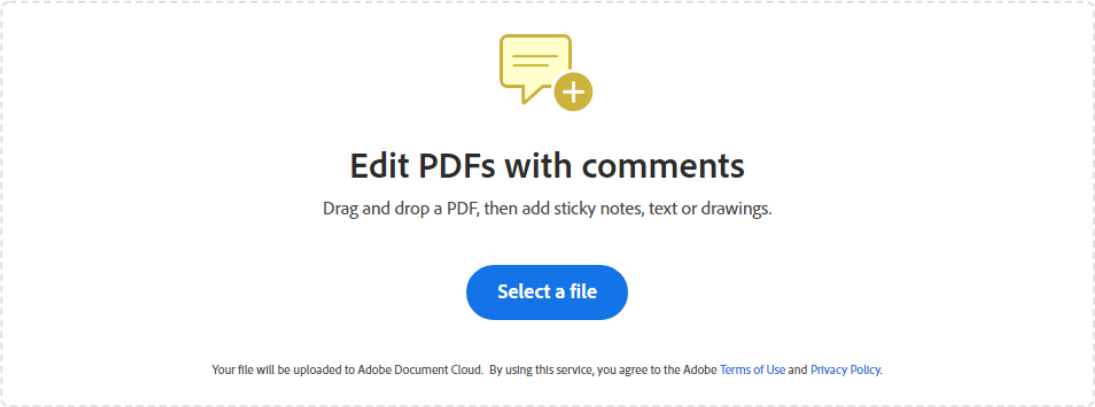


Figura 4 - Adobe Acrobat Pro, selecionar PDF a editar (Adobe, 2021)

Através da Figura 4, é possível observar a interface apresentada para introdução dos documentos. Esta ferramenta permite editar os mesmos de forma simples e rápida. Após introdução dos mesmos, é aberta uma *Web App* onde é possível efetuar a sua edição, como demonstrado na Figura 5.

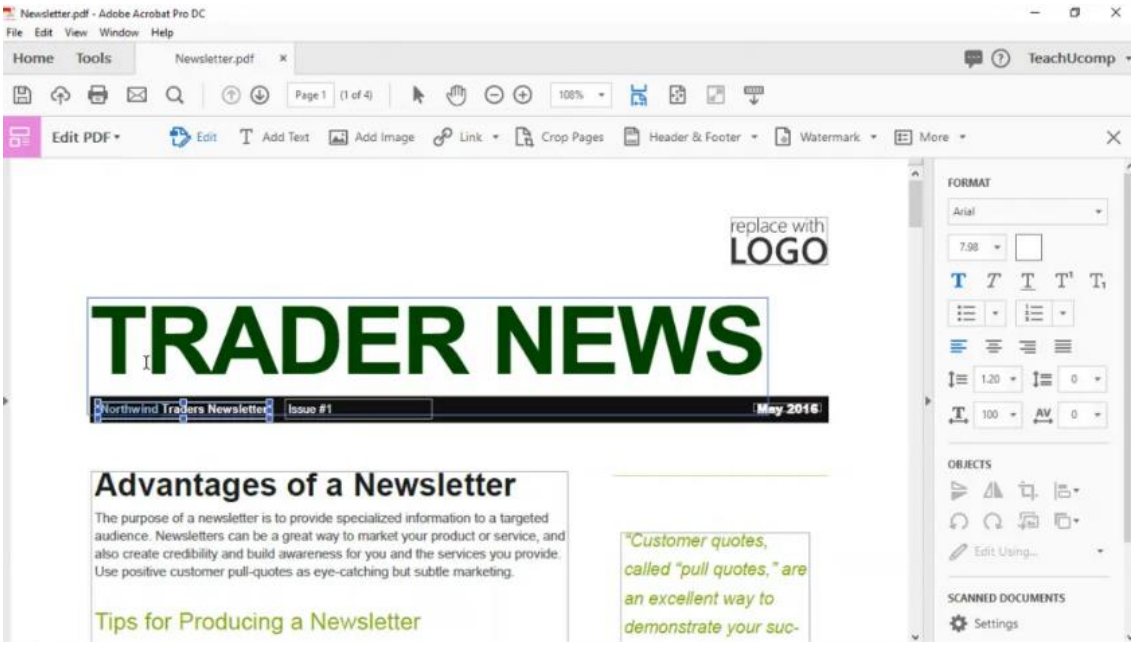


Figura 5 – Interface de edição de PDFs do Adobe Acrobat Pro (Brownell, 2018)

Caso o utilizador não pretenda utilizar a *Web App*, esta ferramenta encontra-se também disponível como aplicação *Desktop*, podendo ser transferida e utilizada sem acesso à Internet.

Apesar do seu fácil uso, esta ferramenta necessita de licença para utilizar grande parte das suas funcionalidades, sendo que a edição dos PDFs se encontra incluída nas mesmas.

2.5.2 PDFelement

PDFelement é uma das ferramentas mais populares para manipulação de documentos PDF, através do uso de OCR. Esta ferramenta, permite criar e editar PDFs através de uma aplicação que pode ser transferida pelos utilizadores. Além disso, permite adicionar dados aos documentos sem alterar a formatação dos mesmos, complementando as fontes e estilos presentes nos mesmos.

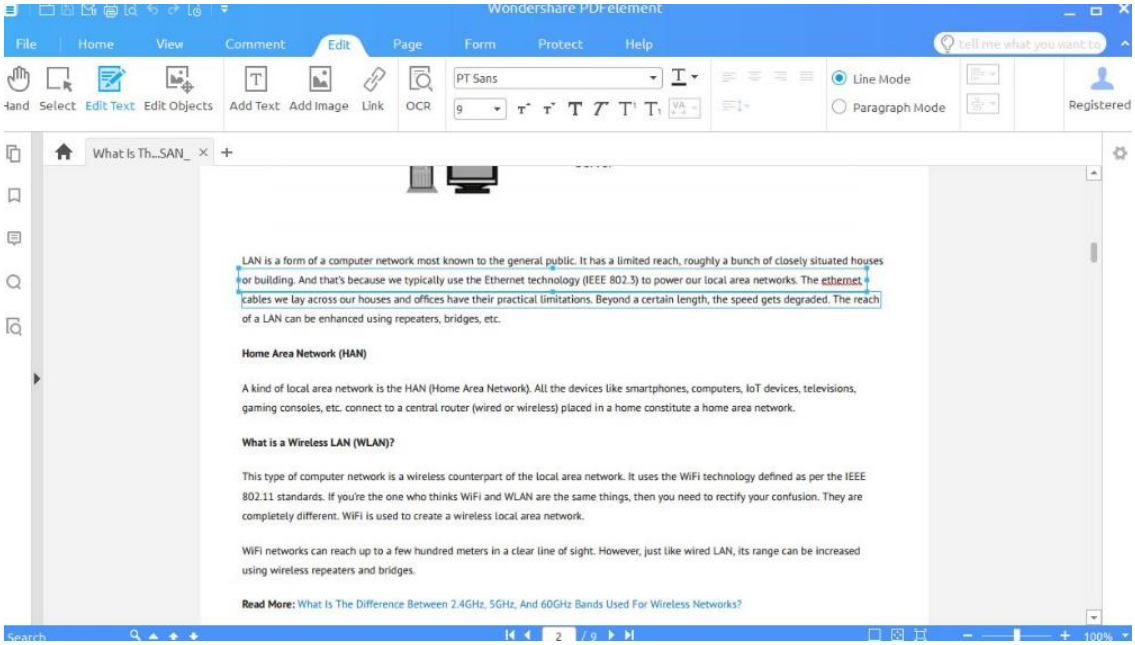


Figura 6 – Interface de edição de PDFs do PDFelement (Verma, 2017)

Como demonstrado na Figura 6, a sua interface é bastante intuitiva e permite efetuar qualquer tipo de ação de forma simples e eficiente. Contudo, esta ferramenta necessita de licença para usufruir de grande parte das suas funcionalidades.

2.5.3 DocDigitizer

DocDigitizer é um serviço de gestão de documentos que permite aos seus utilizadores, extrair e manipular dados de qualquer tipo de documentos, utilizando tecnologias de OCR e *machine learning*. Este serviço é disponibilizado através de uma API, permitindo aos seus clientes integrar a plataforma nas suas aplicações.

Dado o uso de *machine learning*, este serviço possui uma precisão de quase 100% na análise e extração de dados dos documentos. Todo este processo é efetuado com intervenção humana, permitindo que os utilizadores validem os dados extraídos, e como resultado, melhorar o processo(DocDigitizer, 2022).

Contudo, é uma ferramenta que envolve custos para o seu uso, sendo os seus planos normalmente apropriados para organizações que pretendam efetuar este tipo de integração.

2.5.4 Comparação das soluções

De forma a comprovar a necessidade da solução existente, foi efetuada uma comparação das soluções de OCR já existentes no mercado com a solução desenvolvida (Tabela 4).

Tabela 4 - Comparação de soluções de OCR existentes

	Doddle OCR	**Adobe Acrobat Pro**	**PDFelement**	**DocDigitizer**
Open-Source	✓	X	X	X
Suporta extração de informação de imagens	✓	X	X	✓
Capaz de extrair informação com grande eficácia	✓	✓	✓	✓
Permite edição de PDFs através de UI	X	✓	✓	X
Permite receber *input* do utilizador para melhorar análises futuras	X	X	X	✓
Permite mapeamento da informação extraída para criação de certificados	✓	X	X	X

Como demonstrado anteriormente, todas as soluções apresentadas anteriormente possuem funcionalidades não existentes no sistema a desenvolver (Doddle OCR). No entanto, nenhuma resolve o problema apresentado na secção 1.3, visto que é necessário extrair a informação de forma rápida e mapear a mesma de modo a preencher automaticamente novos certificados.

Dessa forma, tornou-se necessário desenvolver um novo sistema que utilizasse algumas das tecnologias e funcionalidades já existentes no mercado, assim como novas funcionalidades que permitissem resolver o problema do presente projeto. Estas funcionalidades incluem a capacidade de extrair informação de fotos tiradas aos certificados previamente emitidos e o mapeamento da informação extraída para a criação automática de novos certificados elétricos.

Por fim, o desenvolvimento de um novo serviço, apenas implica um custo bastante reduzido associado à sua infraestrutura e permite a adição de novas funcionalidades de acordo com as necessidades do cliente.

3 Análise de Valor

Neste capítulo é efetuada a análise de valor do presente projeto, ilustrando e esclarecendo o valor do projeto para o cliente. Dessa forma, é inicialmente identificado o cliente do presente projeto, seguido do enquadramento do processo de inovação, no qual é identificada e analisada a oportunidade. De modo a analisar a oportunidade, são identificadas e abordadas as tendências tecnológicas a nível de OCR e enquadrado o seu valor para o cliente. Por fim, é realizada uma análise *Function Analysis System Technique* (FAST) ao principal serviço a desenvolver na solução proposta.

3.1 Identificação do Cliente

O cliente do presente projeto é a HMC, atual líder na prestação de serviços de conformidade elétrica na Inglaterra. A HMC é uma empresa inglesa que atua no setor elétrico, garantindo que a infraestrutura elétrica das casas e outros edifícios funciona de forma segura e eficiente a toda a hora. Todos os serviços prestados pelos mesmos, são efetuados de acordo com as regulações estabelecidas, simplificando todo esse processo.

Dado a necessidade otimizar e melhorar todo o processo de inspeção dos seus engenheiros, a HMC, decidiu transformar o seu processo de negócio, através do uso de plataformas digitais. Dessa forma, a aposta no desenvolvimento de novas aplicações tem sido o seu principal objetivo nos últimos anos. Esta aposta foi efetuada na nearSea Technologies, tendo sido contratada para desenvolver diversas aplicações que permitissem aos engenheiros efetuar as suas tarefas do dia-a-dia, preenchendo e emitindo os certificados resultantes das suas inspeções. O presente projeto surgiu da necessidade desta digitalização, tendo sido identificados diversos requisitos a desenvolver, descritos na secção 4.1.1.

3.2 Processo de inovação

Dada a rápida e crescente evolução tecnológica, emergem constantemente novos produtos e serviços no mercado, de forma a atender às necessidades dos seus clientes. Contudo, esta constante inovação pode apresentar diversos riscos, tornando-se necessário a existência de um processo de inovação.

Como demonstrado na Figura 7, este processo é dividido em três fases distintas, *Fuzzy Front End*, *New Product Development* e *Commercialization*.

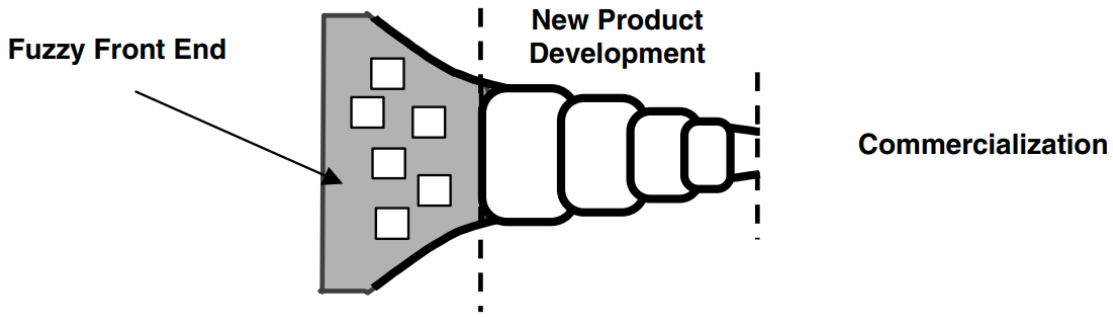


Figura 7 – Processo de inovação (P. A. Koen et al., 2002)

A primeira fase do processo de inovação é denominada de *Fuzzy Front End*. Nesta, é identificado o problema e geradas ideias e oportunidades. Contudo, as atividades desenvolvidas são ainda imprevisíveis e pouco estruturadas.

De seguida, segue o *New Product Development*, fase caracterizada por atividades mais previsíveis e com maior grau de certeza. Além disso é mais rigorosa em termos temporais, sendo orientada a objetivos previamente definidos (P. A. Koen et al., 2002).

A fase final do processo é a fase de *Commercialization*, na qual o produto desenvolvido na anterior fase já se encontra ser lançado no mercado. É nesta fase que os requisitos definidos em conjunto com o cliente são satisfeitos.

No presente projeto, apesar de já se encontrar identificado o problema e as respetivas soluções a implementar, ainda existe alguma incerteza face às tecnologias e estratégias a adotar para o desenvolvimento da mesma. Dessa forma, o projeto insere-se na fase inicial do processo do processo de inovação, *Fuzzy Front End*.

New Concept Development

Com o propósito de criar uma linguagem comum, definindo vários processos constituintes do *Fuzzy Front End*, Koen propôs o modelo *New Concept Development* (P. Koen et al., 2001). Como ilustrado na Figura 8, este é dividido em três partes, o motor, as cinco atividades chave e os fatores de influência.

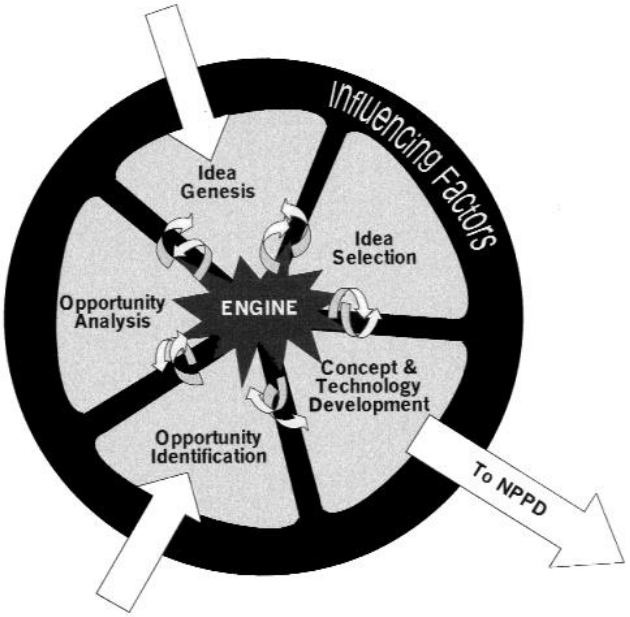


Figura 8 – Modelo *New Concept Development* (P. Koen et al., 2001)

- **Motor** – representa a liderança, a cultura e estratégia empresarial da organização;
- **Atividades Chave** – principais atividades capazes de ser controladas pela organização, sendo elas a identificação e análise da oportunidade, geração, enriquecimento e seleção de ideias, e definição do conceito;
- **Fatores influenciadores** – são os fatores externos não controláveis pela organização que afetam o seu processo de inovação.

3.2.1 Identificação e Análise da Oportunidade

Como mencionado na secção 1.1, cada vez mais o uso de plataformas digitais faz parte do dia-a-dia, permitindo otimizar os processos de negócio das organizações. Dessa forma, o cliente para o qual se destina o presente projeto, decidiu adotar o uso da digitalização nas suas tarefas do dia-a-dia, de modo a aumentar a produtividade e eficiência dos seus engenheiros.

Assim sendo, surgiu a oportunidade de desenvolver o projeto atual em colaboração com a nearSea Technologies, fornecendo diversas funcionalidades que permitem otimizar o trabalho efetuado pelos engenheiros.

Na perspetiva do cliente final, apesar de existirem diversas ferramentas de OCR e soluções dedicadas à importação e extração de informação de PDFs, nenhuma delas permite uma integração com as funcionalidades já existentes, a nível de preenchimento automático de certificados. Daí ser necessário desenvolver diversos serviços que forneçam as funcionalidades de OCR necessários e sejam capazes de ser integrados com a arquitetura já existente.

3.2.2 Geração e seleção da ideia

A ideia surgiu através da investigação e comparação de diversas abordagens existentes na aplicação de tecnologias e ferramentas de OCR, face as necessidades especificadas pelo cliente. De modo a atender às mesmas, decidiu-se que seria desenvolvido um ou mais serviços capazes de permitir a importação e respetivo preenchimento de certificados elétricos de forma automática.

3.3 Valor

"O objetivo principal da análise de valor é avaliar como aumentar o valor de um item ou serviço com o menor custo, sem sacrificar a qualidade." (Nicola, 2020). Nesta secção são apresentados e detalhados os termos valor para o cliente e valor percebido pelo mesmo.

3.3.1 Valor para o cliente

O valor para o cliente é baseado na perceção do mesmo acerca dos benefícios que obtém do produto ou serviço desejado, face aos custos associados aos mesmos (CallMiner, 2020).

No contexto do projeto a desenvolver, o valor para o cliente é maioritariamente a oportunidade de transformar todo o seu processo de negócio através da digitalização das tarefas efetuadas pelos seus engenheiros no dia-a-dia.

3.3.2 Valor percebido

Nesta secção serão abordados os benefícios e os sacrifícios que o estudo e implementação do presente projeto poderão trazer para o cliente final.

Na vertente de produto, um dos principais benefícios para o cliente, é o facto de o presente projeto ser realizado dentro de uma equipa interna da nearSea Technologies, no âmbito da presente dissertação de mestrado. Dessa forma, a solução apresentada será mantida posteriormente pela mesma, existindo sempre espaço para melhoria da mesma, tanto a nível de performance como funcionalidade, garantindo a qualidade da mesma.

Além disso, o presente projeto será desenvolvido utilizando a metodologia Scrum, permitindo obter *feedback* continuo por parte do cliente face à solução implementada. Isto apresenta um enorme benefício para o cliente, visto que garante uma maior flexibilidade da solução e permite ao cliente customizar a mesma.

No entanto, o cliente terá um consumo de tempo e esforço ao longo do desenvolvimento do projeto, de modo a garantir a qualidade da mesma, sendo este o principal sacrifício do mesmo. Este consumo, será efetuado através de sessões de levantamento de requisitos e de avaliação da solução implementada durante o seu período de desenvolvimento.

3.4 Proposta de Valor

A proposta de valor deve ser simples, objetiva e flexível, definindo o valor que uma empresa pretende entregar aos clientes e tenta responder, a questões tais como, qual é o produto, qual ou quais são os clientes alvo do mesmo, qual o valor do produto e o que é que o torna único (Osterwalder et al., 2014).

De forma a efetuar a proposta de valor, é utilizado um modelo de negócio denominado de *Value Proposition Canvas*, demonstrado na Figura 9. Este modelo foi desenvolvido por Alexander Osterwalde, de forma a garantir que os produtos ou serviços desenvolvidos se encontrassem posicionado em torno dos valores do cliente (Osterwalder et al., 2014).

Figura 9 – *Value Proposition Canvas* (Pereira, 2021)

No contexto do presente projeto, foram identificados todos os componentes pertencentes ao modelo representado na Figura 9, sendo eles os ganhos, dores e trabalhos do cliente e os aliviadores de dor e geradores de ganho fornecidos pelo produto.

3.4.1 Perfil do cliente

Trabalhos do cliente – Inclui todas as tarefas que o cliente pretende realizar, os problemas que pretende solucionar e as necessidades que deseja satisfazer (Pereira, 2021).

- Importação e preenchimento automático de certificados elétricos.
- Histórico de inspeções efetuadas anteriormente.
- Gestão de organizações e engenheiros associados às mesmas.

Dores – Engloba todos os problemas e experiências negativas que o cliente encontra no seu dia-a-dia (Pereira, 2021).

- Trabalho repetitivo e exaustivo na emissão de certificados elétricos.
- Não possui um histórico de inspeções passadas, sendo necessário começar de raiz cada uma das inspeções.

Ganhos – Representa todos os benefícios que o cliente espera receber e que podem facilitar a sua vida (Pereira, 2021).

- Digitalização do processo atual.
- Aumento da produtividade e eficiência das inspeções efetuadas.
- Maior competitividade face à concorrência.

3.4.2 Mapa de valor

Produtos e serviços – Inclui todos os produtos e serviços a ser entregues ao cliente (Pereira, 2021).

- Portal para importação de certificados e gestão de organizações.

Geradores de ganhos – Especificam como o produto ou serviço adiciona valor ao cliente e quais os benefícios do mesmo (Pereira, 2021).

- Aumento da produtividade e eficiência das inspeções efetuadas.

Aliviadores de dor – Especificam como o produto ou serviço alivia as dores e dificuldades do cliente (Pereira, 2021).

- Remove o trabalho repetitivo e dispendioso associado à emissão de certificados.
- Diminui os recursos necessários em cada inspeção efetuada.

3.5 Function Analysis System Technique (FAST)

Em 1964, foi introduzida por Charles Bytheway, uma abordagem para a análise de produtos e processos através de uma representação gráfica da análise funcional dos mesmos. Esta representação denomina-se de diagrama FAST e organiza as funções de um determinado produto, processo ou serviço em estudo sobre a forma de questões “Como?” e “Porquê?” (Borza, 2011).

De modo a aplicar esta abordagem ao presente projeto, elaborou-se o diagrama ilustrado na Figura 10. O principal objetivo do projeto passa por desenvolver um serviço que permita analisar e extrair dados de documentos, como certificados elétricos, utilizando tecnologias e ferramentas de OCR. Assim sendo, as funcionalidades deste serviço encontram-se descritas através das tarefas e etapas que as constituem.

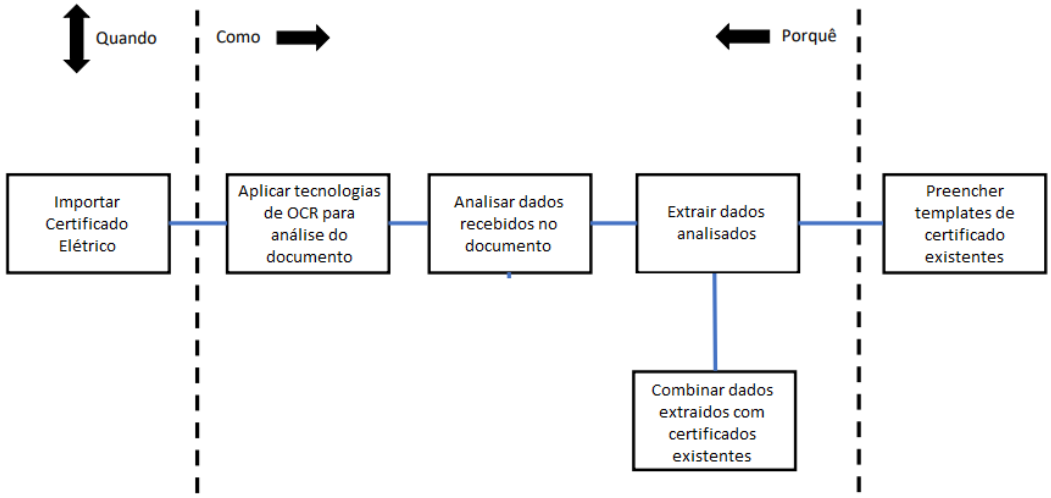


Figura 10 – Diagrama FAST

4 Análise e Desenho

O presente capítulo pretende detalhar a solução para o problema que o projeto se compromete a solucionar.

Inicialmente, é detalhada a Engenharia de Requisitos, onde são apresentados os requisitos não funcionais inerentes ao projeto e os requisitos funcionais dos principais casos de uso desenvolvidos. Por fim, é descrito o modelo de domínio da presente solução, o qual recai sobre os principais conceitos de negócio.

4.1 Análise

Engenharia de requisitos é um processo de recolha e definição dos objetivos que o sistema deve cumprir, baseados nas necessidades do cliente. Este processo é extremamente importante para assegurar que o problema apresentado é claro e que a respetiva solução se encontra implementada corretamente (Jin, 2018).

Nesta secção serão descritos ambos os requisitos funcionais e não funcionais definidos para a implementação da solução. O levantamento dos requisitos foi efetuado em conjunto com a *product owner* do projeto, tanto previamente, como paralelamente ao desenvolvimento do projeto.

4.1.1 Requisitos Funcionais

Os requisitos funcionais especificam os comportamentos e processos operacionais que o sistema deve disponibilizar aos seus utilizadores (Jin, 2018).

Na secção que se segue, são apresentadas as funcionalidades que o sistema deve disponibilizar ao utilizador, para que este realize as atividades/funções de negócio pretendidas.

Elaborou-se um diagrama de casos de uso de forma a demonstrar de forma simples e rápida os requisitos do sistema. A partir das Figura 11, é possível concluir que todos os requisitos do sistema estão distribuídos por dois atores, o Gestor do Portal e o Engenheiro.

De forma a complementar a Figura 11, é feita uma breve descrição sobre cada um dos casos de ambos os atores.

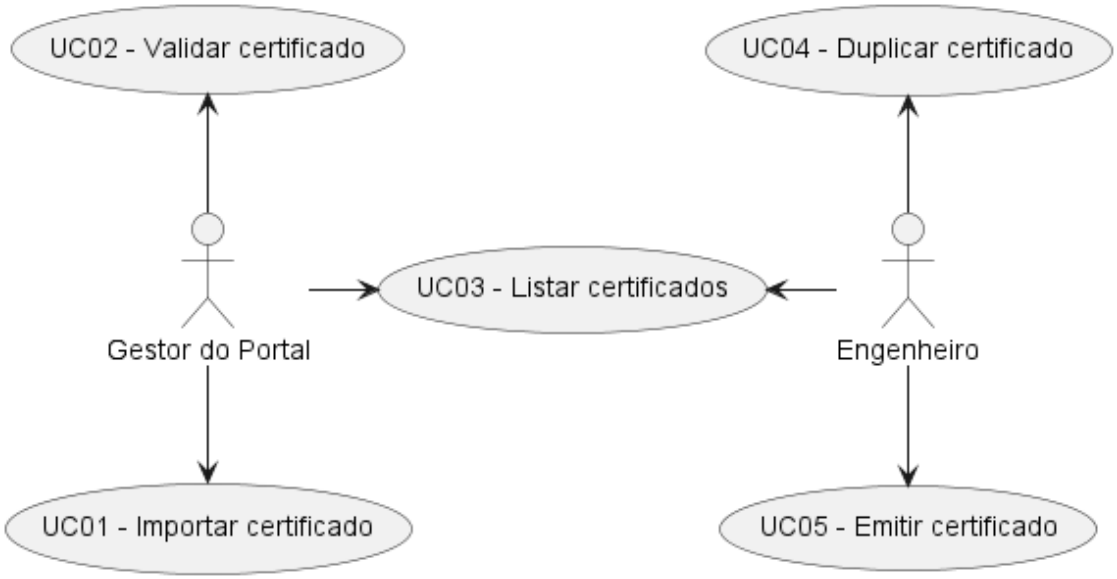


Figura 11 – Diagrama de Casos de Uso (Gestor do Portal e Engenheiro)

UC01 Importar Certificado

Pretende-se que o Gestor do Portal consiga importar certificados elétricos através do portal a desenvolver. Estes certificados devem ser analisados e preenchidos automaticamente através de tecnologias de OCR.

A Figura 12 representa um diagrama de sequência de sistema que demonstra o processo de importação de certificado efetuado pelo Gestor. Este deve previamente tirar ou selecionar as fotos associadas com cada página do certificado que pretende importar. De seguida, seleciona o certificado associado às mesmas e efetua o upload das mesmas para que sejam analisadas. Posteriormente, o sistema extrai a informação das imagens e cria um certificado baseado na mesma.

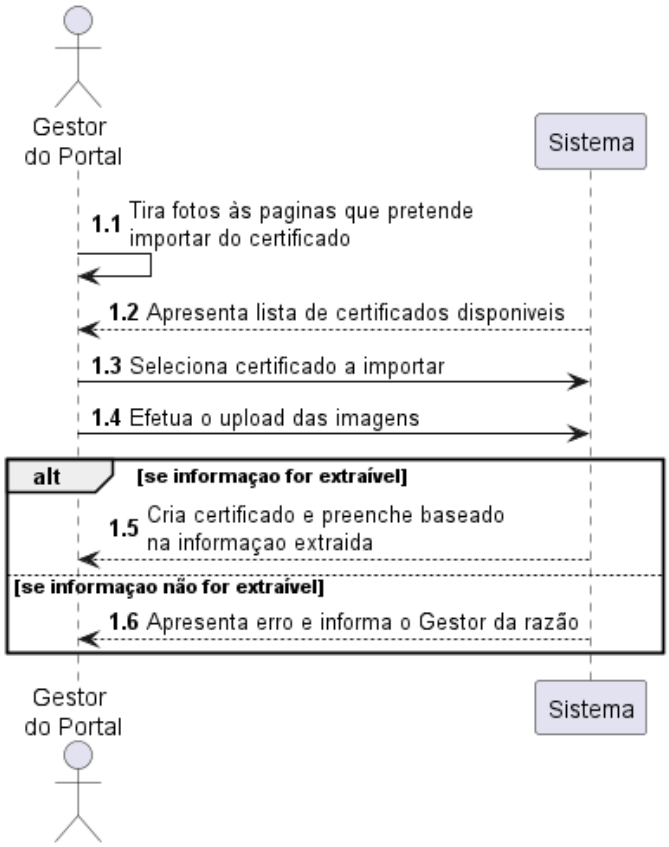


Figura 12 - Diagrama de Sequência de Sistema (UC01 – Importar Certificado)

UC02 Validar certificado

Pretende-se que o Gestor do Portal tenha a possibilidade de validar os certificados importados e preenchidos automaticamente, corrigindo ou complementando qualquer dado incorreto ou em falta.

A Figura 13 representa um diagrama de sequência de sistema que demonstra o processo de validação de certificado efetuado pelo Gestor. Inicialmente, o Gestor abre o certificado após importação do mesmo e altera ou preenche qualquer dado incorreto ou em falta, guardando os mesmos no fim. O sistema valida os dados introduzidos e guarda as alterações associadas ao certificado.

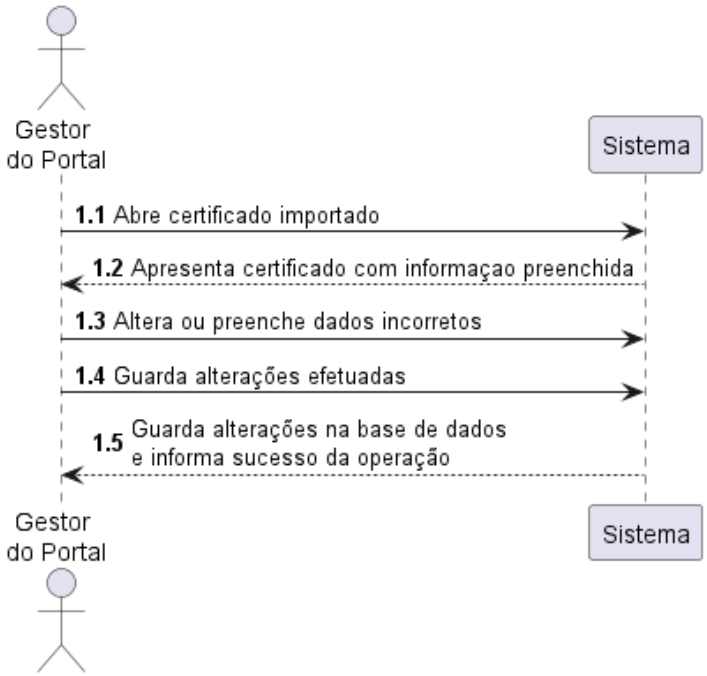


Figura 13 - Diagrama de Sequência de Sistema (UC02 – Validar Certificado)

UC03 Listar certificados

Pretende-se que ambos o Gestor do Portal e os Engenheiros possuam acesso a um histórico de todos os certificados importados previamente, de modo a consultar ou até duplicar e utilizar a informação dos mesmos.

A Figura 14 representa um diagrama de sequência de sistema que demonstra o processo de listagem de certificados demonstrado a ambos os atores. Após entrar no portal, o ator consegue ver todos os certificados importados previamente, permitindo ainda selecionar os mesmos para efetuar a ação desejada.

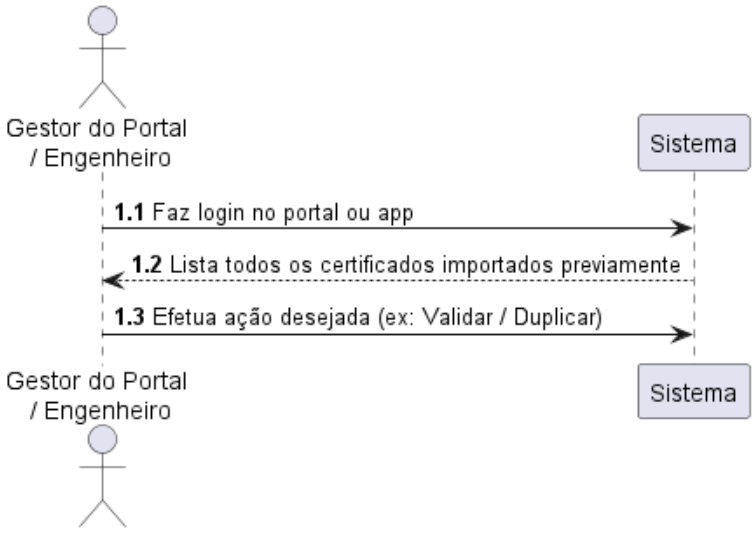


Figura 14 - Diagrama de Sequência de Sistema (UC03 – Listar Certificados)

UC04 – Duplicar Certificado

Pretende-se que o Engenheiro tenha a possibilidade de duplicar os certificados importados, de forma a utilizar a sua informação para emitir novos certificados em localizações onde já ocorreram inspeções previamente. A Figura 15, demonstra o processo mencionado anteriormente.

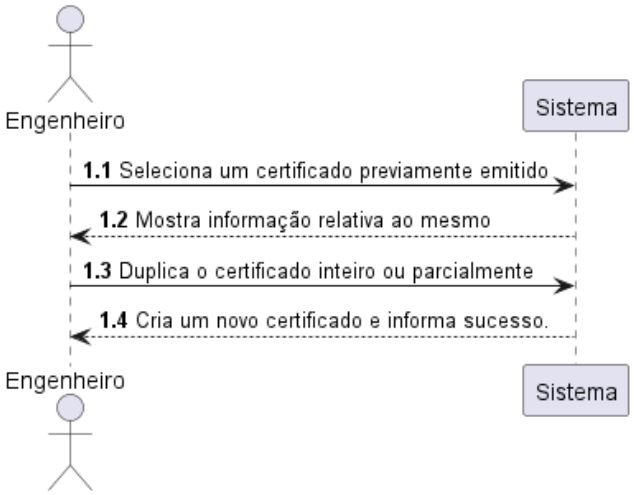


Figura 15 - Diagrama de Sequência de Sistema (UC04 – Duplicar Certificado)

UC05 – Emitir certificado

Pretende-se que o Engenheiro tenha a possibilidade de emitir certificados utilizando a informação de certificados já existentes para pré-preencher os mesmos e acelerar o processo de inspeção no terreno.

A Figura 16, representa um diagrama de sequência de sistema que demonstra o processo de emissão de certificados efetuado pelos Engenheiros. Após duplicar um certificado, o Engenheiro pode entrar no mesmo e editar ou preencher qualquer tipo de informação em falta. Por fim, o sistema valida toda a informação introduzida e permite ao Engenheiro emitir o certificado, informando posteriormente do sucesso da operação.

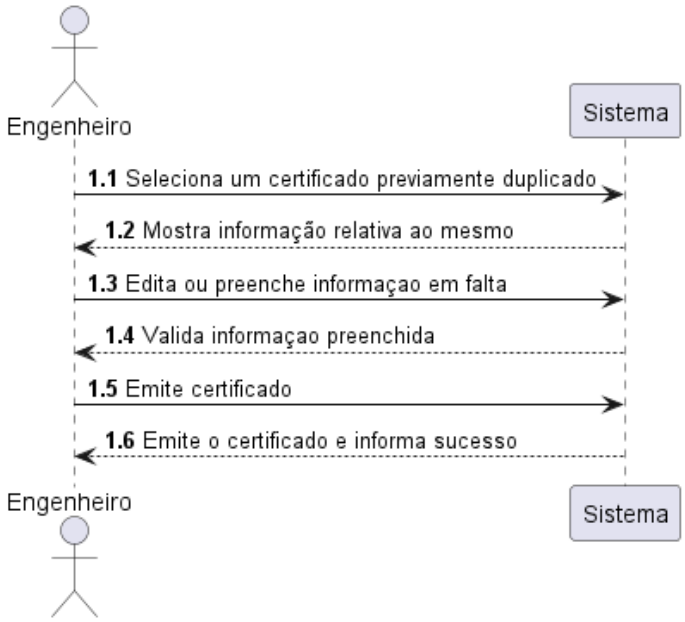


Figura 16 - Diagrama de Sequência de Sistema (UC05 – Emitir Certificado)

4.1.2 Requisitos Não Funcionais

Os requisitos não funcionais especificam os critérios de operação do sistema, ao invés da sua implementação. Assim sendo, os requisitos não funcionais definem as propriedades e restrições do sistema, de forma a existir um critério mais rigoroso na escolha da arquitetura a implementar. De modo a facilitar a identificação destes requisitos é comum utilizar o modelo de classificação de atributos de qualidade FURPS+ (Martin, 2022).

FURPS+ é um acrónimo para as palavras funcionalidade, usabilidade, confiabilidade, desempenho e suportabilidade que representa o modelo proposto pela Hewlett-Packard para classificar os atributos de qualidade de *software*. O '+' inclui requisitos de design, requisitos de implementação, requisitos de interface e requisitos físicos.

Na secção que se segue são apresentados os requisitos que se prendem principalmente com questões subjetivas relacionadas com a qualidade do *software*.

Funcionalidade

- Utilização do idioma inglês tanto na documentação, como na implementação;
- Certificados importados são preenchidos automaticamente com sucesso.
- Análise efetuada pelo SonarQube nas APIs deve ser classificada com a letra A para as métricas de *Vulnerabilities* e *Security Hotspots* da aplicação.

Usabilidade

- Interface gráfica simples e intuitiva, de forma a permitir uma fácil gestão de certificados;
- *Design* responsivo, ou seja, adaptar-se a vários formatos de ecrã mantendo uma boa legibilidade;
- Não obrigar a navegação excessiva (aceder a funcionalidades distintas rapidamente);
- Mensagens de erros objetivas;
- Documentação de todos os pedidos disponibilizados pela API.

Confiabilidade

- Disponibilidade dos servidores deve ser garantida em situações de falha;
- Uma falha deve ser o mais isolada possível, não comprometendo o funcionamento do resto do sistema;
- A comunicação entre serviços deve ser efetuada através de HTTPS.
- Análise efetuada pelo SonarQube nas APIs deve ser classificada com a letra A para a métrica de *Bugs* da aplicação.

Desempenho

- Preservar bons níveis de fiabilidade e performance;
- A API deve ser responsiva e garantir tempos de resposta aceitáveis para uma utilização fluída;
- A navegação na interface a disponibilizar aos utilizadores, deve ser rápida e permitir a execução de inúmeros pedidos simultaneamente.

Suportabilidade

- Toda a implementação deve ser feita de uma forma sustentável, de modo a facilitar a sua manutenção e adição de novas funcionalidades;
- Análise efetuada pelo SonarQube nas APIs deve ser classificada com a letra A para a métrica de *Code Smells* da aplicação.
- Os diferentes componentes do sistema devem poder ser testados de forma isolada;
- Deve ser possível aceder ao sistema através de qualquer *browser* (ex: Google Chrome, Firefox, Edge, *Internet* Explorer);

Restrições de Design

- Deverão ser adotadas boas práticas de *design*, como por exemplo, baixo acoplamento e alta coesão.
- O desenvolvimento do sistema deverá ser feito de forma iterativa e incremental utilizando controlo de versões (GIT).
- O processo de persistência deverá utilizar uma base de dados não relacional de pares chave/valor;
- Comunicação assíncrona entre a interface, a API e a base de dados.

Restrições de Implementação

- Deverão ser implementados testes unitários, de integração e aceitação nos serviços a desenvolver.
- O desenvolvimento do sistema deverá ser feito utilizando controlo de versões (GIT).

Restrições Físicas

- Desenvolvimento em *Windows*.
- Criação de Docker *containers* para todos os serviços desenvolvidos.
- Implantação de todos os serviços desenvolvidos para serviços Google Cloud Platform.

Restrições Físicas

- A *RESTful* API a desenvolver deve assegurar o envio e receção de informação rapidamente.
- Deve ser criada uma *Single-Page Application* para a importação e gestão de certificados elétricos.

4.1.3 Modelo de Domínio

Esta subsecção demonstra o modelo de domínio do presente projeto, através de um diagrama UML constituído pelas entidades e respetivas relações do sistema como apresentado na Figura 17.

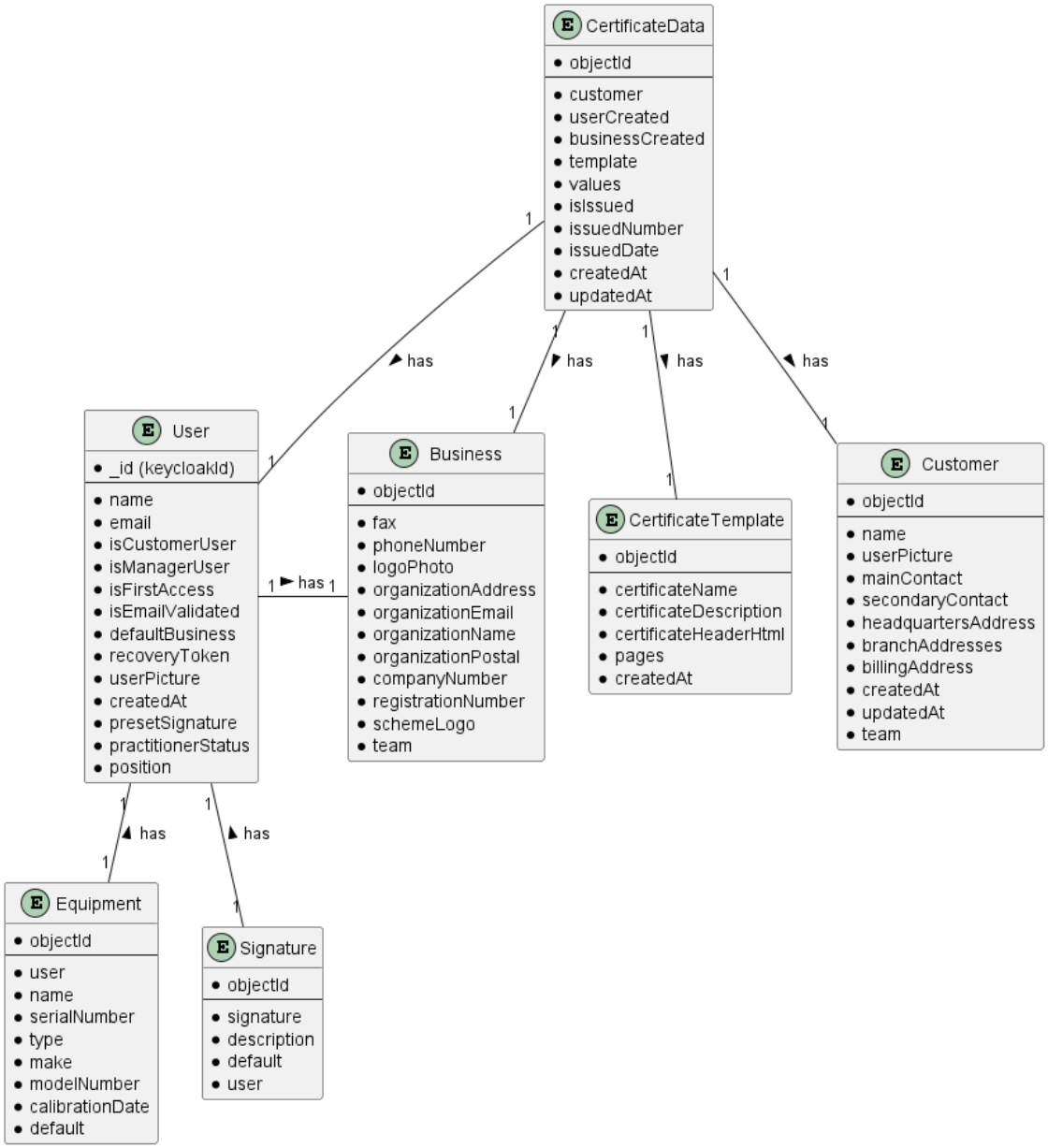


Figura 17 - Modelo de domínio do projeto

A entidade principal do modelo de domínio é a **CertificateData**, representando um certificado e respetiva informação do mesmo. Dado que existem diversos tipos de certificados, cada **CertificateData** possui um **CertificateTemplate** associado, o qual contém o tipo e estrutura do certificado ao qual pertencem os dados. Outra entidade essencial no sistema é o **User**, visto que corresponde a todos os utilizadores do sistema, sejam eles Gestores do Portal ou Engenheiros.

Por fim, as restantes entidades, permitem gerir a organização e respetivos clientes aos quais os utilizadores e certificados criados / importados pelos mesmos pertencem.

4.2 Desenho

A secção que se segue, é dedicada à apresentação da arquitetura do sistema a desenvolver, de forma a expor a estrutura geral do mesmo e a identificar os seus principais componentes para uma melhor compreensão da comunicação e dependências entre os mesmos. Visto que esta fase constitui o elo entre a engenharia de requisitos e o *design*, é uma das fases mais importantes do ciclo de desenvolvimento.

Inicialmente, é detalhada a arquitetura atual, de modo a contextualizar as decisões tomadas a nível arquitetural para a solução a desenvolver. A contextualização é seguida de uma comparação entre duas arquiteturas distintas, sendo elas a arquitetura monolítica e a arquitetura de micro serviços, adotando o modelo Cliente-Servidor. Por fim é demonstrada a arquitetura implementada através do modelo de arquitetura 4 + 1.

4.2.1 Contexto

De forma a satisfazer alguns dos requisitos do cliente, a nearSea Technologies desenvolveu diversas aplicações e serviços como demonstrado na Figura 18. Cada uma dessas aplicações diz respeito a uma plataforma especifica. Esta divisão provém da necessidade da HMC personalizar as suas aplicações de acordo com os diferentes dispositivos que podem ser utilizados no terreno pelos engenheiros.

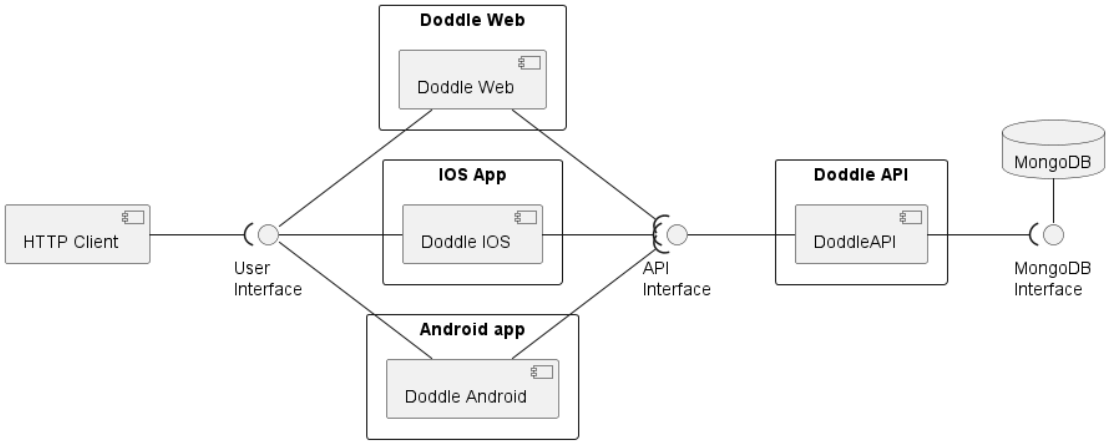


Figura 18 – Arquitetura existente do Doddle

Todas as aplicações possuem regras de negócio iguais, consumindo uma API em Node.js, denominada DoddleAPI. No contexto do projeto a desenvolver, será desenvolvida uma nova API que possuirá toda a lógica de negócio relativa a importação e *scan* de certificados, através de tecnologias de OCR. Esta API será consumida tanto pela API existente, de forma a suportar os requisitos definidos na secção 4.1.1.

4.2.2 Arquitetura Monolítica

Uma solução alternativa para o desenvolvimento do projeto, seria a implementação de uma arquitetura monolítica, ou seja, ambos o Portal e a lógica de negócio baseada em tecnologias OCR seriam desenvolvidos no mesmo serviço (Richardson, 2022). Este tipo de arquitetura possui vantagens como:

- Rapidez de desenvolvimento.
- Rapidez e simplicidade de implantação.
- Redução dos recursos necessários ao seu desenvolvimento e implantação.

Como mencionado anteriormente, esta arquitetura possui vantagens em termos dos recursos necessários à sua implementação, dado que todo o sistema se encontra centralizado num só serviço (Figura 19).

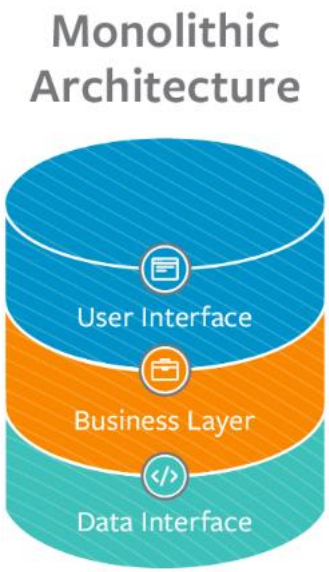


Figura 19 - Representação da arquitetura monolítica (Shirsath, 2020)

No entanto, o facto de toda o sistema se encontrar no mesmo serviço, implica que a sua manutenção implique maior uso de recursos, dificultando ainda a escalabilidade do mesmo. Este tipo de arquitetura dificulta ainda a adição de novas funcionalidades, visto que qualquer alteração possui impacto em todas as camadas do sistema.

Na Figura 20 é demonstrada a possível arquitetura da solução a desenvolver caso o estilo de arquitetura monolítica seja adotado.

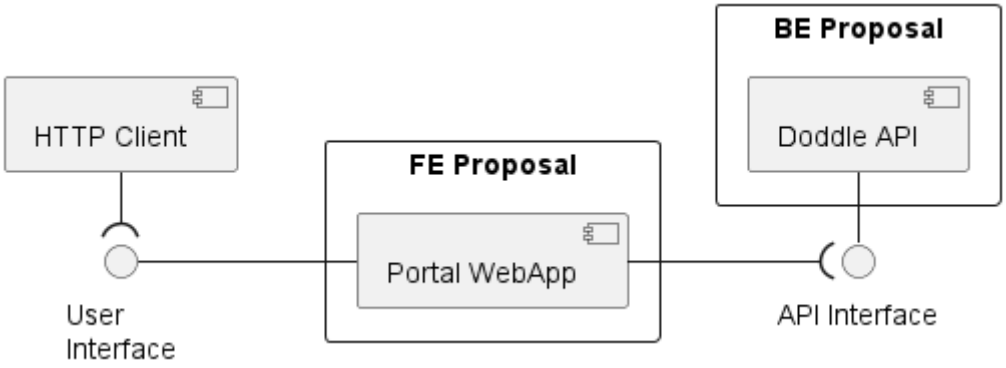


Figura 20 – Proposta arquitetural monolítica

4.2.3 Arquitetura de Micro Serviços

O modelo Cliente-Servidor tornou-se bastante popular nos últimos anos, sendo utilizado para diversos tipos de aplicações. É um modelo bastante utilizadas em arquiteturas de Micro Serviços, sendo constituído por pelo menos um cliente e um servidor, onde o cliente envia pedidos e o servidor responde aos mesmos após processá-los (Figura 21). A sua grande vantagem é permitir a comunicação de múltiplos clientes com o servidor, através de pedidos HTTP (Shakirat Oluwatosin, 2014).

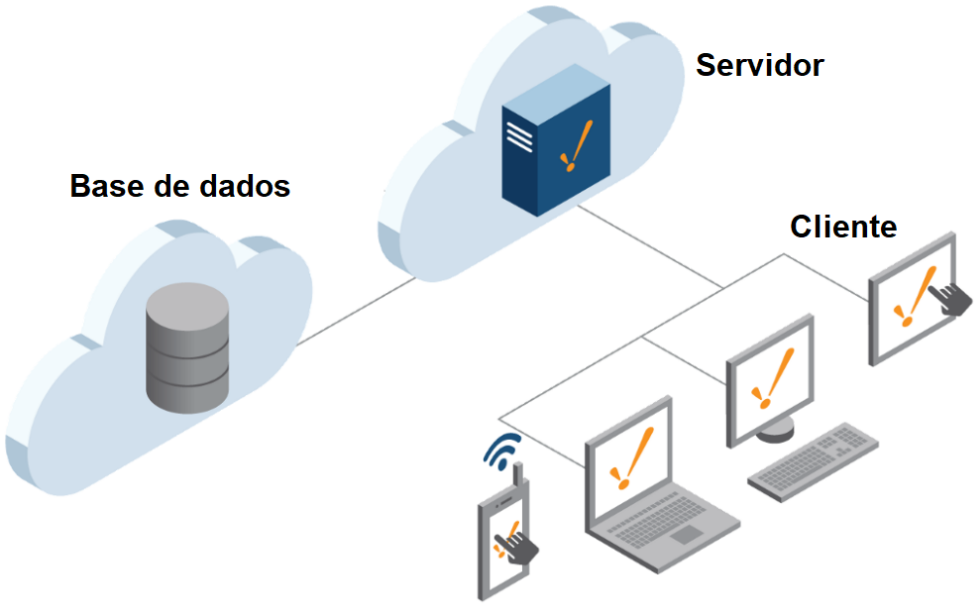


Figura 21 - Representação da arquitetura Cliente-Servidor (Leung, 2019)

Como mencionado anteriormente, cada vez mais este modelo tem vindo a tornar-se popular, visto que possui imensas vantagens:

- Divide o processamento da aplicação por diferentes máquinas;
- Permite maior escalabilidade dos serviços, sem qualquer tipo de interrupção no sistema;
- Reduz o custo de manutenção dos serviços;
- Permite a partilha dos recursos entre cliente e servidor de um modo rápido;
- Minimiza a redundância de informação, sendo que cada serviço tem a sua função.

Além das vantagens mencionadas, o modelo Cliente-Servidor tem vindo a crescer imenso devido à adoção da computação em nuvem por grande parte das empresas. A computação em nuvem consiste na utilização de servidores remotos hospedados na *Internet*, que fornecem serviços e recursos aos utilizadores, de forma a permitir o armazenamento e gestão de dados (Johnson & Shiff, 2021).

Na Figura 22 é demonstrada a possível arquitetura da solução a desenvolver caso o estilo de arquitetura de micro serviços seja adotado.

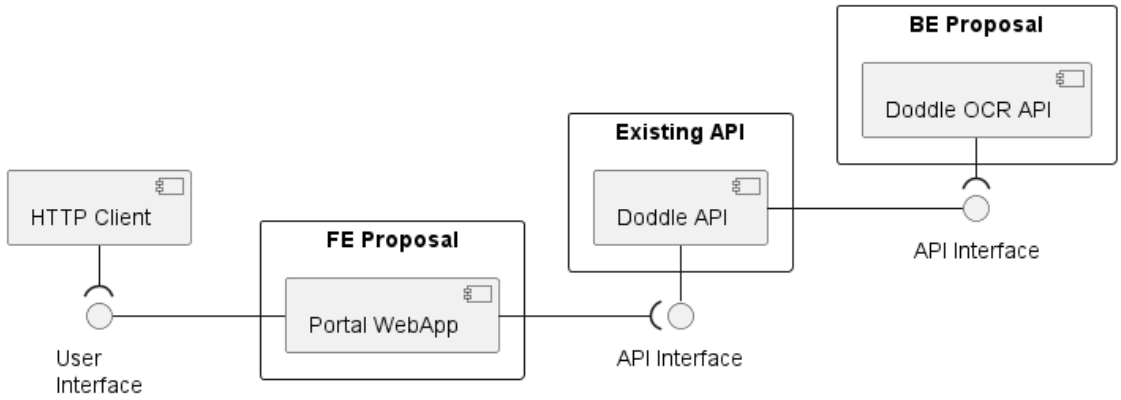


Figura 22 - Proposta arquitetural de micro serviços (Modelo Cliente-Servidor)

4.2.4 Justificação Arquitetural

Contrariamente a uma arquitetura monolítica, onde todos os componentes se concentram na mesma aplicação, a arquitetura Cliente-Servidor permite a separação dos requisitos funcionais em diferentes serviços, garantindo o isolamento e independência das unidades que o compõem, bem como abertura à possibilidade de expansão das mesmas.

A divisão do sistema em diferentes serviços permite dar resposta a diversos requisitos não funcionais definidos na secção 4.1.2, como a alta performance e fiabilidade dos servidores. Dessa forma, permite o isolamento das falhas, impedindo a propagação entre componentes, sendo bastante benéfico para os níveis de disponibilidade e suportabilidade do sistema.

A arquitetura em Micro Serviços é ainda mais vantajosa, quando as preocupações são a manutenção e escalabilidade do sistema. Dado que numa arquitetura monolítica todo o sistema está concentrado na mesma aplicação, torna-se bastante complexo expandi-lo, tal como fazer alterações ao mesmo. Por outro lado, no modelo Cliente-Servidor é possível efetuar a sua manutenção, tal como adicionar novas funcionalidades facilmente, devido à divisão do sistema em diferentes serviços, como mencionado previamente.

De um modo geral, a organização estrutural da aplicação em Cliente-Servidor permite flexibilizar a integração de novas funcionalidades, promovendo o potencial crescimento tecnológico e garantindo simultaneamente elevados níveis de escalabilidade e desempenho.

Tendo em conta os requisitos definidos, a arquitetura em Micro Serviços possui diversas vantagens face à arquitetura Monolítica, sendo a arquitetura em Micro Serviços a adotada para a solução a desenvolver.

5 Implementação

O capítulo que se segue, tem como objetivo descrever e exemplificar o processo de implementação realizado, procurando uma constante correlação do mesmo com a análise e o desenho descritos nos capítulos 4.

Dessa forma, este encontra-se divido sob a forma de secções, sendo que estas correspondem aos diferentes componentes do projeto e à implementação dos mesmos.

5.1 Doddle Backend

A secção que se segue, detalha toda a implementação relativa ao *Backend* da solução desenvolvida. Inicialmente é apresentada toda a configuração e estrutura utilizada para o desenvolvimento das APIs do projeto. De seguida, é efetuada uma descrição dos *endpoints* implementados de acordo com os requisitos funcionais definidos na secção 4.1.1. Por fim, é demonstrada e descrita a lógica de negócio implementada em cada uma das APIs.

5.1.1 Estrutura e configuração e das APIs

Nesta subsecção são apresentados e justificados todos os aspetos tidos em consideração durante a implementação do projeto, quer a nível de estrutura, como de configuração do mesmo. Dado que tanto a Doddle API, como a Doddle OCR API foram desenvolvidas utilizando NestJS (*framework* de Node.js), ambas apresentam estruturas e configurações idênticas.

A estrutura dos projetos em NestJS varia de projeto para projeto, no entanto existem diversos padrões definidos pela sua comunidade para organizar os mesmos. Dessa forma, foi adotada uma arquitetura de três camadas, sendo elas controladores, serviços e acesso aos dados (Eseme, 2022).

Na Figura 23, encontra-se a estrutura adotada para o desenvolvimento de ambas as APIs.

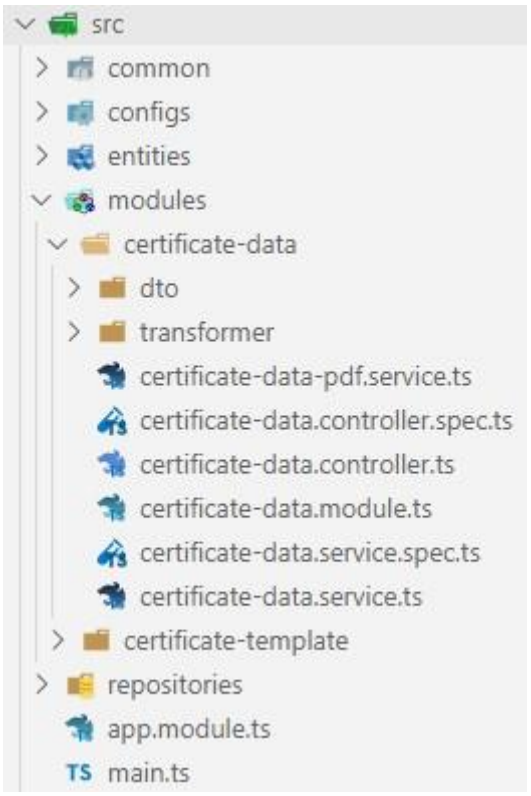


Figura 23 - Estrutura das APIs do projeto

Como demonstrado na Figura 23, toda a lógica de negócio encontra-se organizada dentro da pasta **src/**. Na listagem abaixo é descrito o propósito de cada uma das pastas ou ficheiros que se encontram dentro da mesma:

- **common** – Pasta onde se encontra todo o código genérico ou reutilizável dentro da aplicação, como interfaces, filtros, exceções ou métodos utilitários.

- **configs –** Pasta onde se encontram as configurações dos diversos serviços integrados com a API, como bases de dados e email.

- **entities –** Pasta onde se encontram definidas todas as entidades a criar na base de dados.

- **modules –** Pasta de onde se encontra grande parte da lógica de negócio. Cada módulo corresponde a uma entidade da aplicação e é sempre constituído por um controlador e um serviço.

    - **controlador –** local onde são definidos os *endpoints* da API.

    - **serviço –** local onde se encontra a lógica de negócio associada a cada endpoint.

- **repositories –** Pasta onde se encontram definidos todos os repositórios, responsáveis por permitir o acesso à base de dados.

Além da estrutura, um aspeto bastante importante ao longo do processo de desenvolvimento das APIs é a utilização de diversas ferramentas que permitam aumentar a qualidade e produtividade do mesmo.

Na Figura 24, encontram-se todos os ficheiros de configuração utilizados para o desenvolvimento da API.

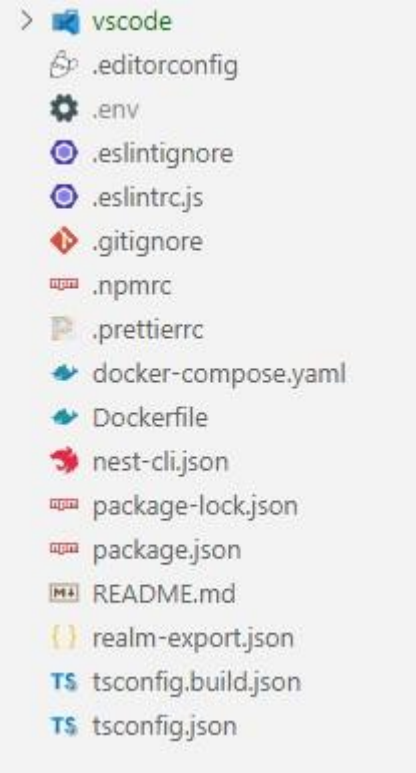


Figura 24 - Ficheiros de configuração das APIs

De entre os ficheiros apresentados na Figura 24, é possível destacar diversas ferramentas utilizadas como:

- **eslint –** ferramenta de análise de código, que identifica problemas em código Javascript ou Typescript de acordo com as regras configuradas previamente.

- **prettier –** ferramenta de formatação de código, que deteta qualquer alteração no código e formata o mesmo de acordo com padrões configurados previamente.

- **docker –** serviço utilizado para correr localmente os micro serviços e bases de dados, permitindo simular os ambientes de teste e produção.

- **.env –** ficheiro utilizado para definir e popular as variáveis de ambiente da aplicação, de modo a permitir o desenvolvimento local.

5.1.2 API Specification

Esta subsecção apresenta e descreve os *endpoints* que foram implementados na Doddle API de acordo com os requisitos funcionais definidos na secção 4.1.1. De forma a especificar os mesmos, foi utilizada o Swagger, ferramenta que permite documentar e servir de interface para testar REST APIs. Por fim, é apresentado todo o processo e interação dos diferentes componentes do projeto em cada um dos requisitos implementados.

Importar certificado

Este *endpoint* permite importar certificados elétricos através das imagens enviadas pelo utilizador. Na Figura 25, é demonstrada a especificação do mesmo, sendo necessário enviar um pedido utilizando o método **HTTP POST** para a rota **/ocr** em conjunto com os parâmetros e *body* definidos.

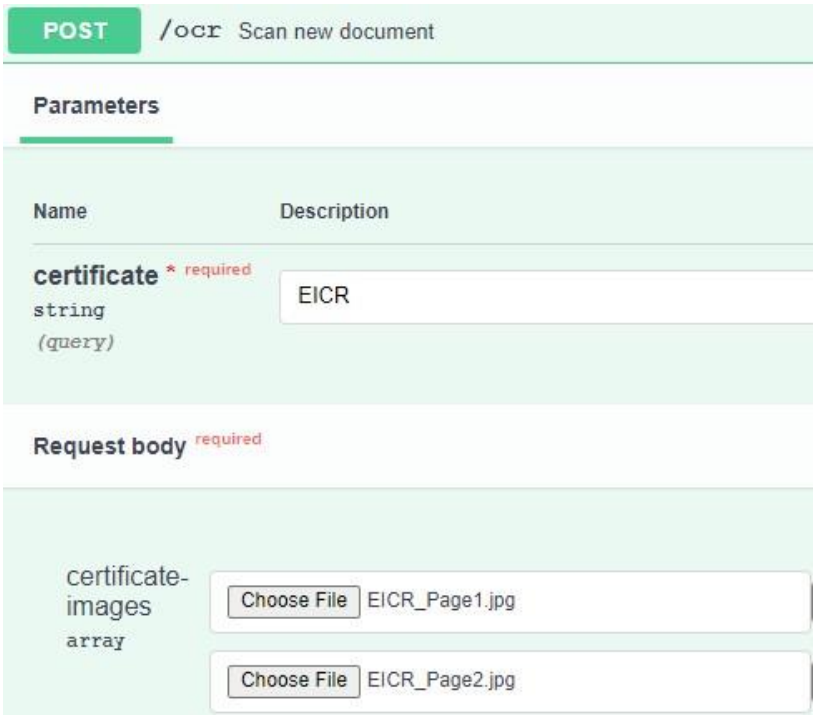


Figura 25 – Especificação Swagger Doddle API (Importar Certificado)

Os parâmetros e body necessários para o pedido descrito anteriormente encontram-se descritos na listagem abaixo:

- **certificate** – parâmetro obrigatório enviado no URL do pedido, especificando o tipo de certificado a importar. Deve corresponder com as imagens enviadas.

- **certificates-images –** imagens do certificado utilizadas para o *scan* e criação do certificado na aplicação.

Com o intuito de descrever todo o processo de importação do certificado e a interação dos diferentes serviços do sistema do certificado, criou-se o diagrama de sequência demonstrado na Figura 26.

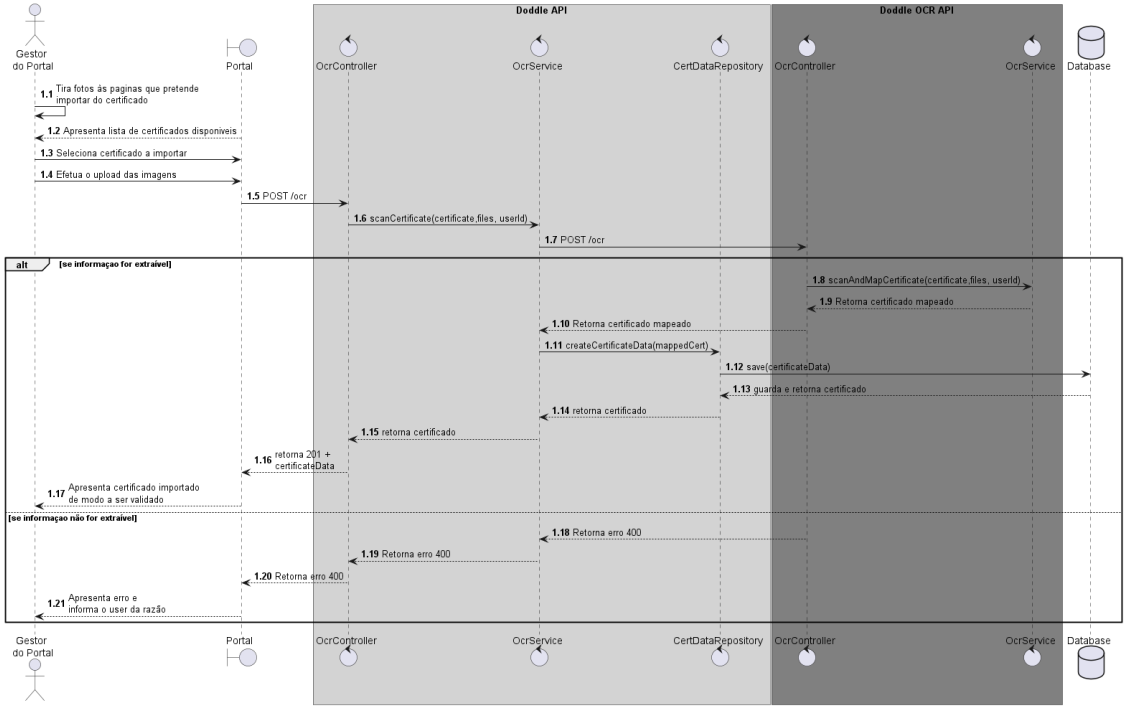


Figura 26 - Diagrama de Sequência (Importar Certificado)

Assim que o utilizador efetua a submissão das imagens, o Portal efetua um pedido à Doddle API, sendo este posteriormente recebido pelo **OcrController** e enviado para o **OcrService**, serviço responsável por comunicar com a Doddle OCR API e criar o certificado após receber todos os dados mapeados do mesmo. Após enviar o pedido à Doddle OCR API, este chega ao **OcrController** da mesma, sendo depois enviado para o **OcrService**, onde é efetuada toda a lógica de OCR, sendo analisadas todas a imagens e extraída a informação das mesmas. Por fim, a informação extraída é mapeada e retornada à Doddle API. Após obter os dados mapeados, o **OcrService** da Doddle API, cria um certificado através do **CertDataRepository** e retorna o mesmo ao utilizador.

Validar certificado

O *endpoint* que se segue, permite ao gestor do portal validar e ajustar os dados automaticamente preenchidos do certificado após importar o mesmo. Na Figura 27, é demonstrada a especificação do mesmo, sendo necessário enviar um pedido utilizando o método **HTTP PATCH** para a rota **/certificate-data/{id}** em conjunto com os parâmetros e body definidos.

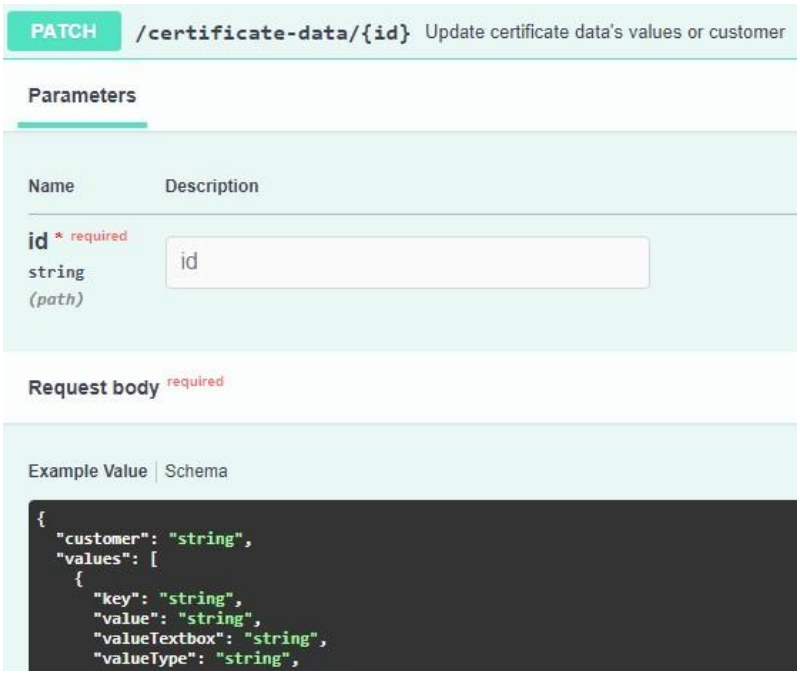


Figura 27 - Especificação Swagger Doddle API (Validar Certificado)

Os parâmetros e *body* necessários para o pedido descrito anteriormente encontram-se descritos na listagem abaixo:

- **id** – parâmetro obrigatório enviado no URL do pedido, especificando o id do certificado a editar.

- **values** - valores do certificado que o utilizador pretende corrigir ou preencher.

De modo a de descrever todo o processo de validação do certificado, criou-se o diagrama de sequência demonstrado na Figura 28.

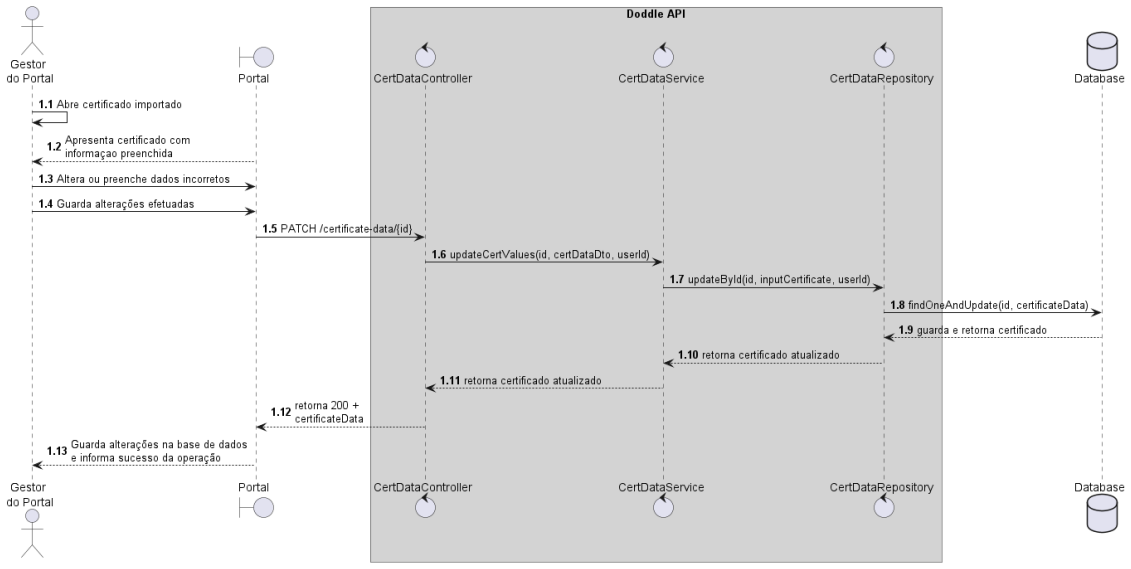


Figura 28 - Diagrama de Sequência (Validar Certificado)

Após guardar as informações validadas, o Portal efetua um pedido à Doddle API, sendo este posteriormente recebido pelo **CertDataController** e enviado para o **CertDataService**, serviço responsável pela lógica de negócio associada à edição de certificados. O serviço deteta todas os valores que foram preenchidos ou alterados e atualiza o certificado com os mesmos, invocando o **CertDataRepository**. Por fim retorna o certificado, já atualizado e informa o portal do sucesso da operação.

Listar certificados

Para que ambos o gestor do portal e os engenheiros consigam consultar os certificados previamente importados ou emitidos, foi criado o *endpoint* descrito abaixo. Na Figura 29, é demonstrada a especificação do mesmo, sendo necessário enviar um pedido utilizando o método HTTP **GET** para a rota **/certificate-data** em conjunto com os parâmetros definidos.

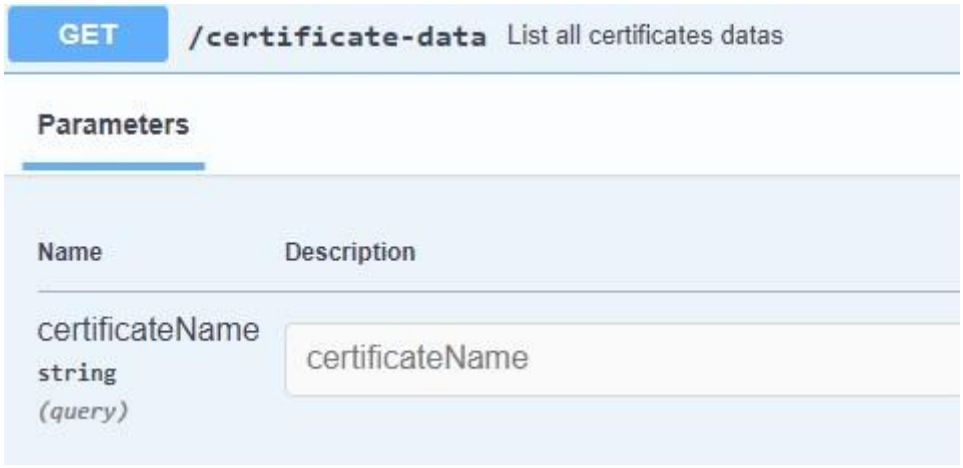


Figura 29 - Especificação Swagger Doddle API (Listar Certificados)

Os parâmetros necessários para o pedido descrito anteriormente encontram-se descritos abaixo:

- **certificateName** – parâmetro opcional que serve para filtrar os certificados por nome.

De modo a de descrever todo o processo de listagem dos certificados, criou-se o diagrama de sequência demonstrado na Figura 30.

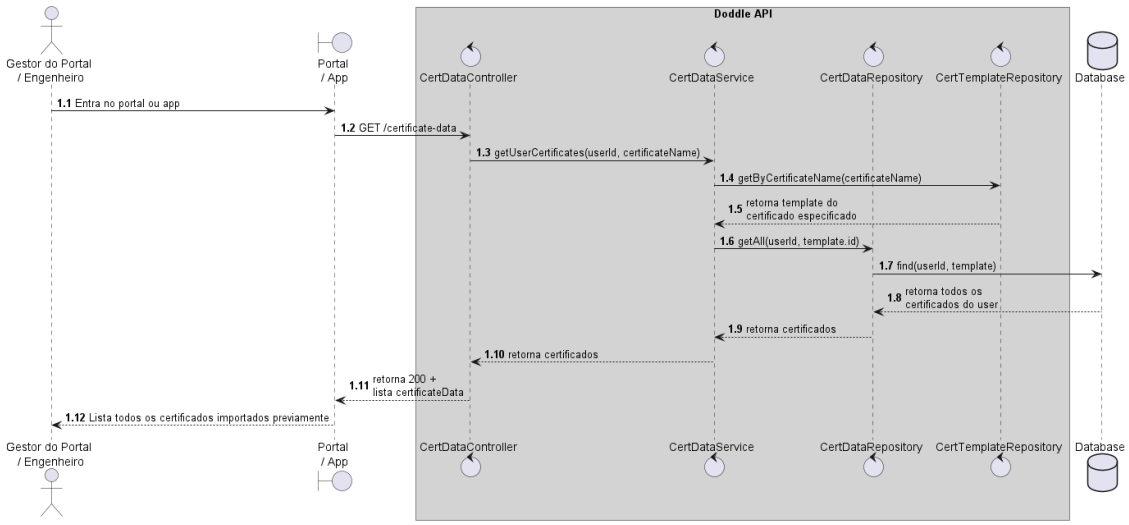


Figura 30 - Diagrama de Sequência (Listar certificados)

Assim que o utilizador entra no portal ou na *app*, é efetuado um pedido à Doddle API, sendo este posteriormente recebido pelo **CertDataController** e enviado para o **CertDataService**. Inicialmente, o serviço invoca o **CertTemplateRepository**, de modo a obter o *template* associado do certificado enviado por parâmetro. Assim que obtém o mesmo, efetua um pedido ao **CertDataRepository** para obter todos os certificados criados previamente pelo utilizador, filtrando-os pelo *template* obtido anteriormente. Por fim retorna uma lista de todos os certificados ao utilizador.

Duplicar certificado

Este *endpoint* permite aos engenheiros duplicar certificados previamente importados e emitidos, de modo a reutilizar informação já preenchida em prévias inspeções. Na Figura 31, é demonstrada a especificação do mesmo, sendo necessário enviar um pedido utilizando o método HTTP **POST** para a rota **/certificate-data/duplicate** em conjunto com o *body* definido.

Figura 31 - Especificação Swagger Doddle API (Duplicar Certificado)

O *body* necessário para o pedido descrito anteriormente encontra-se descrito na listagem abaixo:

- **certificateId** – id do certificado a duplicar.
- **partial** – valor que determina se o certificado deve ser duplicado por completo ou apenas parcialmente.

De modo a descrever todo o processo de duplicação do certificado, criou-se o diagrama de sequência demonstrado na Figura 32.

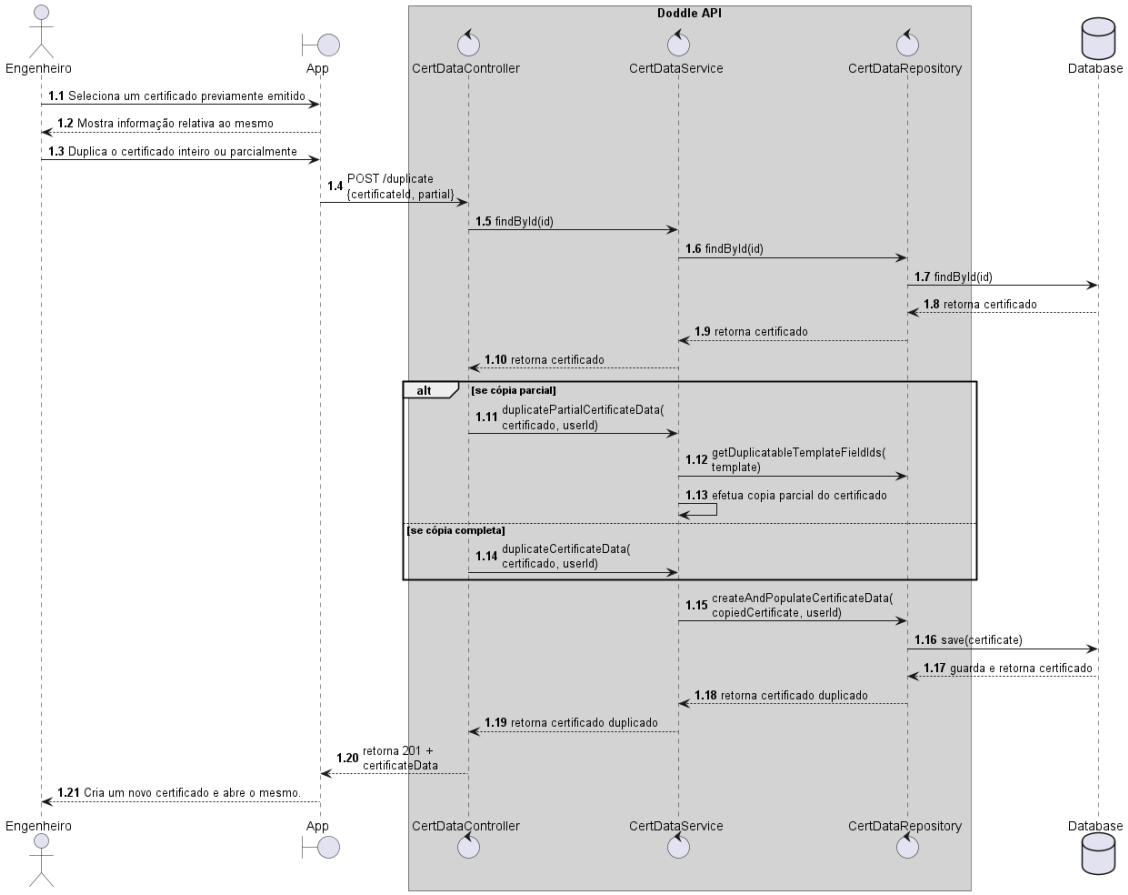


Figura 32 - Diagrama de sequência (Duplicar certificado)

Assim que o utilizador duplica um certificado previamente importado, é efetuado um pedido à Doddle API, sendo este posteriormente recebido pelo **CertDataController**. Dentro do controlador, é inicialmente obtido o certificado a duplicar e posteriormente enviado para o **CertDataService**. Dentro do serviço é verificado o campo **partial** para determinar se a duplicação deve ser efetuada por completo ou apenas parcialmente. De seguida, o serviço efetua a cópia dos valores do certificado e cria um novo invocando o **CertDataRepository**. Por fim é retornado o novo certificado criado ao portal.

Emitir certificado

O *endpoint* que se segue, permite ao engenheiro emitir certificados previamente duplicados, de forma a acelerar o processo de inspeção no terreno. Na Figura 32, é demonstrada a especificação do mesmo, sendo necessário enviar um pedido utilizando o método **HTTP PUT** para a rota **/certificate-data/{id}/issue** em conjunto com os parâmetros definidos.

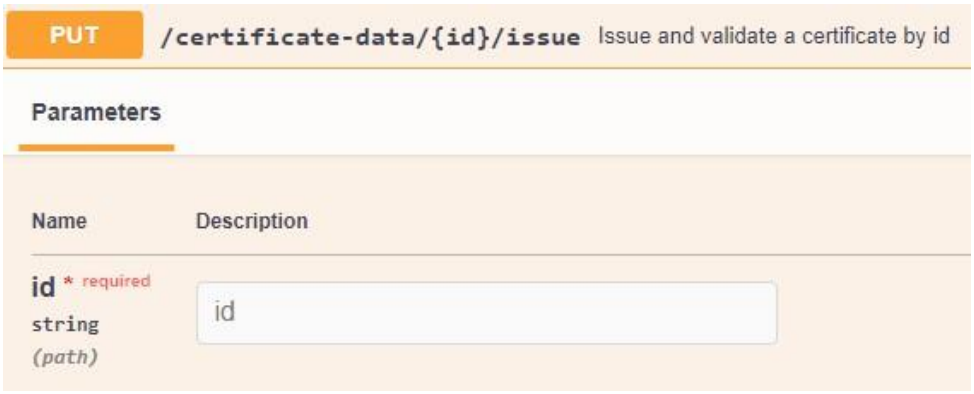


Figura 33 - Especificação Swagger Doddle API (Emitir Certificado)

O parâmetro necessário para o pedido descrito anteriormente encontra-se descrito na listagem abaixo:

- **id –** id do certificado a emitir.

De forma a descrever todo o processo de emissão do certificado, criou-se o diagrama de sequência demonstrado na Figura 32.

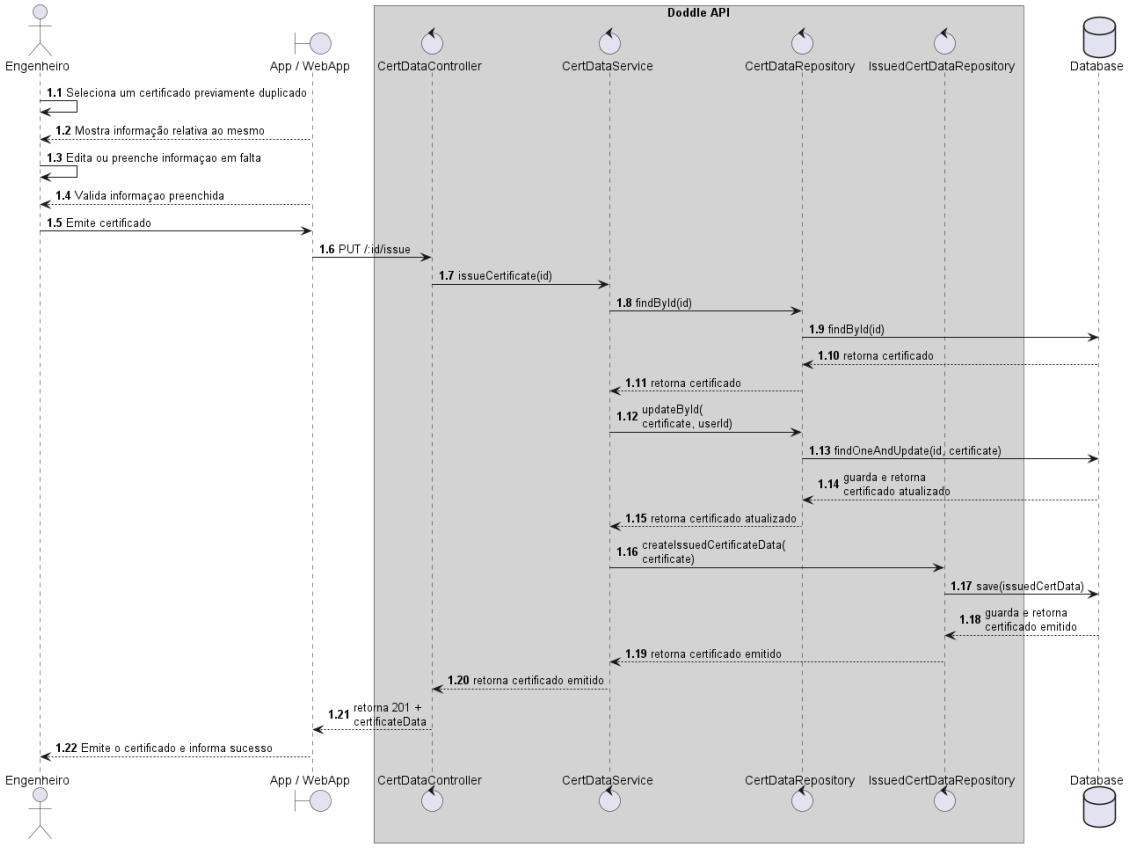


Figura 34 - Diagrama de Sequência (Emitir certificado)

Após selecionar um certificado duplicado e validado, os engenheiros têm a oportunidade de emitir o mesmo, efetuando um pedido à Doddle API. O pedido é recebido pelo **CertDataController** e é redirecionado para o **CertDataService**, onde é efetuada toda a lógica de emissão do certificado. Inicialmente, é obtido o certificado a emitir e atualizado o seu estado, sendo posteriormente invocado o **IssuedCertDataRepository** para criar uma cópia estática do certificado emitido e guarda a mesma na base de dados. Por fim é retornado o certificado emitido ao portal.

5.1.3 Doddle API

Esta subsecção pretende descrever e apresentar a lógica de negócio adicionada à API já existente no Doddle. Para o contexto do presente projeto, apenas serão descritas as novas funcionalidades à mesma, de forma a possibilitar os requisitos funcionais definidos na secção 4.1.1.

A Doddle API foi desenvolvida utilizando Typescript e NestJS, *framework* de Node.js que permite desenvolver aplicações de *backend*. Dada a necessidade de persistir os dados relativos aos certificados elétrico, foi criada uma base de dados MongoDB e integrada com a Doddle API, de forma a efetuar a gestão dos mesmos.

Como descrito na secção 5.1.1, a API encontra-se divida em três camadas a nível arquitetural. De maneira a apresentar as mesmas, será utilizado o processo de importação de certificado detalhado na Figura 26, dado que é a funcionalidade mais relevante do projeto.

Controlador

O controlador é a camada onde são definidos os *endpoints* da API, sendo responsável por receber os pedidos efetuados à API e responder aos mesmos após reencaminhá-los para a camada de serviço.

No Excerto de Código 1 é demonstrado o controlador implementado para a importação dos certificados.

```
1  export const CERTIFICATE_IMAGES_FIELD = 'certificate-images';
3  @Controller('ocr')
4  export class OCRController {
5    constructor(private readonly ocrService: OCRService) {}
6
7    @Post()
8    @UseInterceptors(FilesInterceptor(CERTIFICATE_IMAGES_FIELD, 20,
{ fileFilter: imageFileFilter }))
9    async scanCertificate(
10     @Req() { user }: JwtRequest,
11     @Query('certificate', new ParseEnumPipe(Certificate))
certificate: Certificate,
12     @UploadedFiles() files: Express.Multer.File[],
13   ) {
14     return this.ocrService.scanCertificate(files, certificate,
user.sub);
```

Excerto de Código 1 – Camada de Controlador (Importação de Certificado)

Uma das grandes vantagens da *framework* NestJS, é o facto de possuir imensos *decorators* que permitem definir não só os *endpoints*, mas também os parâmetros, *headers* e *body* enviados para os mesmos. Além disso, permite ainda definir *middlewares* e *interceptors* com apenas algumas linhas de código, com o intuito de implementar lógica extra sobre os pedidos efetuados.

Na listagem abaixo é efetuada uma descrição dos *decorators* (identificados pelo @) utilizados no Excerto de Código 1.

- **@Post –** define o *path* do *endpoint* e o método HTTP utilizada para o mesmo.
- **@UseInterceptors –** utilizado para intercetar os pedidos efetuados e filtrar as imagens enviadas nos mesmos.
- **@Req() { user } –** extrai a informação do utilizador enviada na autenticação, sendo posteriormente utilizada para identificar o mesmo.
- **@Query("certificate”...) –** define e valida o parâmetro onde é enviado o tipo de certificado a importar.

- **@UploadedFiles()** – define o tipo de informação que espera receber no *body* do pedido, que neste caso, é uma lista de imagens.

Serviço

O serviço é a camada onde se encontra implementada grande parte da lógica de negócio da API. Quando necessário, é a camada responsável por comunicar tanto com a camada de acesso aos dados, como com qualquer outro serviço externo, nomeadamente a Doddle OCR API.

No Excerto de Código 2, é demonstrado o controlador implementado para a importação dos certificados.

```
1   async scanCertificate(files: Express.Multer.File[], certificate:
Certificate, userId: string) {
2     const url = `ocr?certificate=${certificate}`;
3
4     const formData = new FormData();
5     files.forEach((file) =>
6       formData.append('cert-images', file.buffer,
file.originalname),
7     );
8
9     const headers = {
10      API_KEY: this.configService.get('OCR_API_KEY'),
11      ...formData.getHeaders(),
12    };
13
14    const response = await fetch(url, {
15      method: 'POST',
16      body: formData,
17      headers,
18    });
19
20    if (!response.ok) {
21      const data = await response.json();
22      throw new HttpRequestException(response)
23    }
24
25    const values: CertificateDataValuesDto[] = await
response.json();
26
27    const template = await
this.certTemplateRepo.get(Certificate.EICR);
28
29    return this.certificateDataService.createCertificateData(
30      { template: template.id, values },
31      userId,
32    );
33  }
```

Excerto de Código 2 - Camada de Serviço (Importar Certificado)

Este serviço é responsável por efetuar a comunicação com a Doddle OCR API, a qual foi desenvolvida com o intuito de extrair e mapear a informação das imagens relativas aos certificados elétricos. Na linha 14, é utilizada a libraria *fetch*, para efetuar a comunicação com a Doddle OCR API através de um pedido HTTP, sendo redirecionada a informação recebida no controlador para a mesma. Caso a análise OCR seja efetuada com sucesso, o pedido retorna os valores do certificado todos mapeados, sendo estes enviados para a camada de acesso aos dados com o intuito de criar um certificado com os mesmos.

Acesso aos Dados

A camada de acesso aos dados é responsável por estabelecer a comunicação com a base de dados e gerir toda a informação dentro da mesma. No contexto da solução desenvolvida, esta camada é identificada através de Repositórios, como demonstrado no Excerto de Código 3.

```
1 export class CertificateDataRepository {
2   constructor(
3     @InjectModel(CertificateData.name)
4     private readonly certificateDataModel:
Model<CertificateData>,
5   ) {}
6
7   async createCertificateData(
8     createCertificateDataDto: CreateCertificateDataDto,
9     userId: string,
10    businessId: Types.ObjectId,
11  ): Promise<CertificateData> {
12    const newCertificateData = new this.certificateDataModel({
13      ...createCertificateDataDto,
14    });
15    newCertificateData.userCreated = userId;
16    newCertificateData.businessCreated = businessId;
17
18    return newCertificateData.save();
19  }
20 }
```

Excerto de Código 3 - Camada de Acesso aos dados (Importar certificado)

Cada repositório pertence a uma entidade definida na base de dados, sendo responsável por todas as chamadas efetuadas à mesma. No Excerto de Código 3 encontra-se definido o repositório utilizado para gerir a entidade **CertificateData**, como podemos ver na linha 3 através do *decorator* *@InjectModel(CertificateData.name)*. Esta entidade foi criada com o intuito de guardar todos os valores associados aos certificados elétricos criados.

Base de dados

De maneira a efetuar a persistência dos utilizadores, certificados, e a restante informação relativa aos mesmos, foi utilizada uma base de dados MongoDB na Doddle API. No presente tópico, é demonstrado um exemplo da especificação das entidades da aplicação, as ferramentas utilizadas para a conexão à base de dados e a configuração das mesmas.

De forma a efetuar a conexão à base de dados, foi utilizada a ferramenta de modelação de objetos *mongoose*. O *mongoose* é um *Object Document Mapper* (ODM), que permite mapear os documentos definidos na API para entidades na base de dados. Esta ferramenta disponibiliza ainda uma interface que fornece todas as funcionalidades necessárias para gerir a informação na base de dados.

No Excerto de Código 4, é demonstrada a configuração utilizada para a conexão à base de dados.

```
1     MongooseModule.forRootAsync({
2       imports: [ConfigModule],
3       inject: [ConfigService],
4       useFactory: async (configService: ConfigService) => ({
5         uri:
6           configService.get('MONGODB_PROTOCOL', 'mongodb://') +
7           configService.get('MONGODB_USER') +
8           ':' +
9           configService.get('MONGODB_PASSWORD') +
10          '@' +
11          configService.get('MONGODB_HOST') +
12          '/' +
13          configService.get('MONGODB_DBNAME') +
14          configService.get('MONGODB_OPTIONS', ''),
15        sslCA: configService.get('MONGODB_CA_CERTIFICATE'),
16      }),
17    }),
18
```

Excerto de Código 4 - Configuração do ODM mongoose

A configuração demonstrada anteriormente permite à aplicação conectar-se à base de dados MongoDB assim que inicia, com o intuito de posteriormente efetuar as operações necessárias e gerir a informação dentro da mesma.

Através da utilização do *mongoose*, é possível definir todas as entidades da base de dados na API sem ter de as criar diretamente na base de dados. Dado que a principal entidade da aplicação é a **CertificateData**, como mencionado na secção 4.1.3, será descrita a implementação da mesma através do Excerto de Código 5.

```
@Schema()
export class CertificateData extends Document<Types.ObjectId> {
  @Prop({
    type: MongooseSchema.Types.ObjectId,
    required: false,
    ref: CertificateTemplate.name,
  })
  template?: Types.ObjectId;

  @Prop({ type: CertificateDataValuesDto, required: false })
  values?: Array<CertificateDataValuesDto>;

  @Prop({ type: Boolean, required: true, default: false })
  isIssued: boolean;

  @Prop({ type: Number, required: false })
  issuedNumber: number;

  @Prop({ type: Date, required: false })
  issuedDate: Date;
```

Excerto de Código 5 - Entidade CertificateData

Utilizando as funcionalidades disponibilizadas pelo *mongoose*, é possível definir a entidade através do código, sendo posteriormente criada automaticamente assim que a conexão à base de dados é efetuada.

Na listagem abaixo é efetuada uma descrição de cada um dos *decorators* utilizados na implementação da entidade:

- **@Schema –** especifica que a *class* é um documento da base de dados, ou seja, é criado na mesma assim que a API é iniciada.

- **@Prop –** utilizado para definir as propriedades do documento e configurar as mesmas.

5.1.4 Doddle OCR API

Na subsecção que se segue é descrita e apresentada a implementação do principal componente do presente projeto, a Doddle OCR API. Esta API contém toda a lógica de negócio relativa ao OCR e à extração e mapeamento da informação dos certificados elétricos importados.

Tal como a Doddle API, a Doddle OCR API foi desenvolvida utilizando Typescript e NestJS. Como mencionado na secção 5.1.1, a nível arquitetural, a Doddle OCR API apresenta uma estrutura idêntica à Doddle API. Visto já ter sido efetuada uma descrição da mesma na secção 5.1.3, os tópicos que se seguem serão apenas focados na implementação da tecnologia de OCR e não na arquitetura da API.

GCloud Vision API

Com o intuito de suportar as funcionalidades de OCR, foi utilizada a Cloud Vision API. API desenvolvida pela Google, capaz de analisar e extrair informação de imagens através de funcionalidades como reconhecimento facial, deteção de conteúdo explícito e reconhecimento ótico de caracteres (OCR).

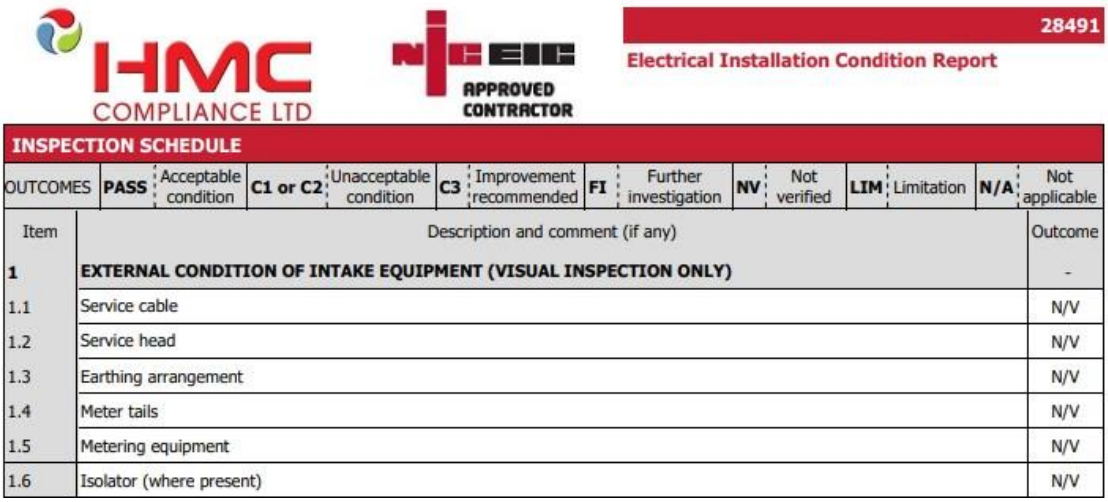
No Excerto de Código 6, é demonstrada a utilização e implementação desta ferramenta.

```
1  import { ImageAnnotatorClient } from '@google-cloud/vision';
2
3  private readonly client: ImageAnnotatorClient
4
5  async scanDocument(files: Express.Multer.File[]){
6    return Promise.all(files.map(async (file) =>
this.scan(file)));
7  }
8
9  protected async scan(file: Express.Multer.File) {
10   return this.client.documentTextDetection(file.buffer);
11 }
```

Excerto de Código 6 - Implementação da Cloud Vision API

Dado que a Cloud Vision API apenas permite analisar uma página de cada vez, é inicialmente efetuada uma separação das mesmas, sendo posteriormente agregado todo o texto extraído das mesmas.

Na linha 10, é utilizado o cliente instanciado da Cloud Vision API, para invocar o método **documentTextDetection**. Este método é responsável por efetuar uma análise OCR no documento e extrair todo o texto presente no mesmo, identificando-o por coordenadas e segmentando o mesmo, ou seja, agrupando o texto por linhas.



HMC COMPLIANCE LTD

APPROVED CONTRACTOR

28491

Electrical Installation Condition Report

INSPECTION SCHEDULE

OUTCOMES	**PASS** Acceptable condition	**C1 or C2** Unacceptable condition	**C3** Improvement recommended	**FI** Further investigation	**NV** Not verified	**LIM** Limitation	**N/A** Not applicable

Item	Description and comment (if any)	Outcome
1	**EXTERNAL CONDITION OF INTAKE EQUIPMENT (VISUAL INSPECTION ONLY)**	-
1.1	Service cable	N/V
1.2	Service head	N/V
1.3	Earthing arrangement	N/V
1.4	Meter tails	N/V
1.5	Metering equipment	N/V
1.6	Isolator (where present)	N/V

Figura 35 – Primeira secção da Página 5 do certificado elétrico EICR

No Excerto de Código 7, é demonstrado um exemplo do texto extraído e segmentado da página do certificado apresentada na Figura 35.

```
{
    "description": [
      "HHMC",
      "COMPLIANCE LTD",
      "INSPECTION SCHEDULE",
      "OUTCOMES PASS Acceptable C1 or C2",
      "condition",
      "Item", "1", "1.1", "1.2", "1.3", "1.4","1.5", "1.6",
      "Unacceptable", "condition", "Metering equipment", "Isolator
      (where present)","CEIC", "APPROVED", "CONTRACTOR",
      "Electrical Installation Condition Report", "Improvement",
      "Further", "recommended investigation", "FI", "Description
      and comment (if any)", "EXTERNAL CONDITION OF INTAKE
      EQUIPMENT (VISUAL INSPECTION ONLY)", "Service cable",
      "Service head", "Earthing arrangement", "Meter tails", "NV",
      "Not", "verified", "LIM; Limitation N/A;", "28491", "Not",
      "applicable", "Outcome", "N/V", "N/V", "N/V", "N/V", "N/V",
      "N/V"
    ]
}
```

Excerto de Código 7 – Exemplo de texto extraido pela Cloud Vision API

Embora a ferramenta seja capaz de extrair todo o texto presente na página, o resultado da segmentação automática efetuada pela mesma é impreciso. Como demonstrado no Excerto de Código 7, as linhas da tabela não se encontram agrupadas no resultado obtidos, sendo praticamente impossível identificar o ***Outcome*** de cada ***Item***.

Dessa forma, surgiu a necessidade de implementar um algoritmo que fosse capaz de segmentar todo o texto extraído do documento, agrupando e ordenando o mesmo em linhas, com o intuito de tornar a sua extração bem mais fácil.

Além da segmentação automática demonstrada no Excerto de Código 7, a análise OCR efetuada pela Cloud Vision API contém ainda uma lista de todas as palavras detetadas com as suas respetivas coordenadas, como demonstrado no Excerto de Código 8.

```
{
	"locations": [],
	"properties": [],
	"mid": "",
	"locale": "",
	"description": "Item",
	"score": 0,
	"confidence": 0,
	"topicality": 0,
	"boundingPoly": {
		"vertices": [
			{
				"x": 7,
				"y": 109
			},
			{
				"x": 60,
				"y": 109
			},
			{
				"x": 60,
				"y": 118
			},
			{
				"x": 7,
				"y": 118
			}
		],
		"normalizedVertices": []
	}
}
```


Excerto de Código 8 - Palavra extraída do certificado e identificada por coordenadas

A identificação das palavras e respetivas coordenadas é efetuada através de pequenos retângulos nas imagens analisadas, representados pelo objeto *boundingPoly* e respetivos vértices. Utilizando as coordenadas de cada palavra, é possível identificar a sua posição e consequentemente a linha a que cada uma delas pertence. Com o intuito de segmentar o resultado obtido pela Cloud Vision API e utilizar o mesmo para o mapeamento dos certificados, foi desenvolvido o algoritmo apresentado no tópico seguinte.

Algoritmo de Segmentação

Com o intuito de tornar o texto extraído mapeável, foi desenvolvido um algoritmo de segmentação, capaz de ordenar e agrupar todas as palavras no certificado em linhas. Utilizando as coordenadas de cada palavra, o algoritmo é capaz de identificar a posição das mesmas e agrupá-las em linhas.

No Excerto de Código 9, é apresentado o método responsável por agrupar as palavras em linhas.

```
1 export function mergeBoundingPolygons(segmentedLines){
2   Object.keys(segmentedLines).forEach((_, index) => {
3     let arr: Vertex[] = [];
4
5     // calculate line height
6     const h1 =
7       segmentedLines[index].boundingPoly.vertices[3].y -
8       segmentedLines[index].boundingPoly.vertices[0].y;
9     const h2 =
10      segmentedLines[index].boundingPoly.vertices[2].y -
11      segmentedLines[index].boundingPoly.vertices[1].y;
12    const avgHeight = (h1 + h2) / 2;
13
14    arr.push(segmentedLines[index].boundingPoly.vertices[0]);
15    arr.push(segmentedLines[index].boundingPoly.vertices[1]);
16    const line1 = getLineRectangle(deepcopy(arr), avgHeight);
17
18    arr = [];
19    arr.push(segmentedLines[index].boundingPoly.vertices[2]);
20    arr.push(segmentedLines[index].boundingPoly.vertices[3]);
21    const line2 = getLineRectangle(deepcopy(arr), avgHeight);
22
23    segmentedLines[index]['verticesMatrix'] =
createRectCoordinates(line1, line2);
24    segmentedLines[index]['lineNum'] = index;
25    segmentedLines[index]['match'] = [];
26    segmentedLines[index]['matched'] = false;
27  });
28 }
```

Excerto de Código 9 - Função responsável por agrupar as palavras extraídas em linhas

Utilizando os vértices y, é possível determinar a linha a que cada palavra pertence (Linhas 6 e 9), permitindo posteriormente identificar e agrupar todas as palavras que pertencem a determinada linha.

Após efetuar o agrupamento das palavras em linhas, é efetuada a segmentação das mesmas utilizando as coordenadas dos retângulos de cada palavra, como demonstrado no Excerto de Código 10.

```
1  function segmentLines(mergedWords: MergedEntityAnnotation[]) {
2    const segmentedLines: SegmentedLineCoordinate[] = [];
3
4    // Only lines that aren't matched should be at the start.
5    Object.keys(mergedWords).forEach((_, index) => {
6      if (mergedWords[index]['matched']) return;
7
8      let line: string;
9      if (mergedWords[index]['match'].length) {
10       line = sortWords(mergedWords, index);
11     } else {
12       line = mergedWords[index].description as string;
13     }
14
15     segmentedLines.push({
16       line,
17       coordinate: mergedWords[index].boundingPoly.vertices[0].y,
18     });
19   });
20
21   return segmentedLines;
22 }
```

Excerto de Código 10 - Função responsável pela segmentação das linhas extraídas

Após agrupar todas as palavras em linhas e identificar as mesmas através de retângulos representado as suas coordenadas, é efetuada uma análise sobre as mesmas, com o intuito de identificar as linhas que se encontram adjacentes através do objeto match na linha 9. Esta análise permite agrupar as palavras que se encontram em linhas adjacentes sendo o caso das moradas introduzidas pelos utilizadores. Por fim, são adicionadas as linhas e respetivas coordenadas a uma lista, a ser ordenada, que contém o resultado da segmentação.

O Excerto de Código 11, demonstra o resultado obtido pelo algoritmo de segmentação aplicado no certificado demonstrado na Figura 35.

```
{
	"segLinesCoords": [
		{
			"line": "OUTCOMES PASS Acceptable C1 or C2
condition Unacceptable condition recommended investigation
Improvement FI Further NV verified Not LIM; Limitation N/A;
applicable Not",
			"coordinate": 122
		},
		{
			"line": "Item Description and comment (if any)
Outcome",
			"coordinate": 152
		},
		{
			"line": "1 EXTERNAL CONDITION OF INTAKE EQUIPMENT
(VISUAL INSPECTION ONLY)",
			"coordinate": 178
		},
		{
			"line": "1.1 Service cable N/V",
			"coordinate": 200
		},
		{
			"line": "1.2 Service head N/V",
			"coordinate": 222
		},
		{
			"line": "1.3 Earthing arrangement N/V",
			"coordinate": 244
		},
		{
			"line": "1.4 Meter tails N/V",
			"coordinate": 266
		},
		{
			"line": "1.5 Metering equipment N/V",
			"coordinate": 288
		},
		{
			"line": "1.6 Isolator (where present) N/V",
			"coordinate": 310
		}
	]
}
```


Excerto de Código 11 - Exemplo de texto segmentado pelo algoritmo de segmentação

O resultado apresentado anteriormente, contém todas as linhas do certificado ordenadas por coordenadas, ou seja, pela altura em que se encontram na página. Comparando este resultado ao obtido no Excerto de Código 7, é possível concluir que a segmentação efetuada através do algoritmo desenvolvido é bem mais eficiente, dado que é capaz de detetar e agrupar as palavras por linhas como apresentadas na Figura 35. Além disso, o algoritmo adiciona a coordenada y de cada linha, permitindo utilizar a mesma para o mapeamento da informação, que se encontra descrito no tópico seguinte.

Mapeamento dos Certificados

De forma a criar e preencher automaticamente novos certificados, surge a necessidade de mapear toda a informação extraída e segmentada em valores dos certificados. Para tal, foram criadas duas classes com responsabilidades diferentes, sendo elas a **CertificateParser** e a **CertificateMapper**.

Com o intuito de suportar diversos certificados e tornar este serviço escalável, ambas as classes são genéricas, ou seja, a sua implementação não se encontra efetuada apenas para um certificado, mas sim para múltiplos caso necessário. A implementação genérica deste mapeamento, teve como base a utilização do padrão de *design Factory*, como demonstrado no Excerto de Código 12.

```
2    constructor(
3      protected parsers: {
4        [template: string]: CertificateParser;
5      },
6    ) {}
7
8    public get(template: string) {
9      if (!this.parsers[template]) {
10       throw new ApiException(`Parser for ${template} not implemented.`,
11         error: `${CertificateParserFactory.name}Exception`,
12         status: HttpStatus.NOT_IMPLEMENTED,
13       });
14     }
15
16     return this.parsers[template];
17   }
18 }
19
20 export abstract class CertificateParser {
21   public abstract parseSections(
22     lineSeg: string[],
23     segmentationResult: SegmentedLineCoordinate[],
24   ): CertificateSections;
25
26   public abstract parseTable(
27     lineSeg: string[],
28     segmentationResult: SegmentedLineCoordinate[],
29   ): CertificateSections;
30
```

Excerto de Código 12 – Padrão de *Design Factory* utilizado na Doddle OCR API

O padrão de *design* mencionado anteriormente, permitiu criar uma implementação genérica do mapeamento, utilizando apenas um único *endpoint* para a extração e mapeamento da informação dos diferentes certificados, como apresentado na Figura 25.

Todos os certificados elétricos importados, encontram-se divididos em diversas secções. De modo a identificar as mesmas foi criada a classe **CertificateParser** exibida no Excerto de Código 12. Esta classe é responsável por analisar o resultado retornado pelo algoritmo de segmentação e identificar as secções presentes no mesmo, dividindo o certificado pelas mesmas, como demonstrado no Excerto de Código 13.


```
{
	"SECTION A: DETAILS OF THE CLIENT": [
		{
			"line": " Address London 14-18 Finsbury Square",
			"coordinate": 115
		},
		{
			"line": "EC2A 1AH",
			"coordinate": 138
		}
	],
	"SECTION B: REASON FOR PRODUCING THIS REPORT": [
		{
			"line": "Safety assessment in line with GN3
recommended frequency for environment",
			"coordinate": 174
		},
		{
			"line": "Dates on which the inspection and testing
was carried out 27 Aug 2021",
			"coordinate": 196
		}
	]
}
```


Excerto de Código 13 – Resultado do CertificateParser após analisar informação extraída

Cada certificado possui um *template* associado onde se encontram definidos todos os campos do mesmo através de ids. De modo a conseguir identificar cada um dos campos, foi efetuada a divisão em secções demonstrada anteriormente e criado um objeto JSON com o mapeamento de cada campo baseado na sua secção e coordenada.

No Excerto de Código 14 é demonstrado um exemplo do mapeamento implementado para as secções A e B do certificado EICR.

```
1   'SECTION A': {
2     Name: {
3       key: 'field1618600632768',
4       valueType: 'customer-data',
5       coordinates: [77],
6     },
7   },
8   'SECTION B': {
9     'Reason for this report': {
10      key: 'field1618600650973',
11      valueType: 'combo-modal',
12      coordinates: [77],
13    },
14    'Inspection date': {
15      key: 'field1618600700974',
16      valueType: 'large-datepicker',
17      coordinates: [98],
18      extract: (value: string) => value.split('out ')[1],
19    },
20  }
```

Excerto de Código 14 - Exemplo do mapeamento dos campos do certificado EICR

Como demonstrado anteriormente, o mapeamento dos valores do certificado deve ser feito um a um, identificando os campos através das coordenadas mapeadas no algoritmo de segmentação. Utilizando um objeto JSON com todas as secções e campos a extrair das mesmas é possível mapear todos os campos do certificado a criar e preencher automaticamente o mesmo. Na listagem que se segue, são detalhados os diferentes campos utilizados e a sua utilidade:

- **Section** – Inicialmente é definida a secção e os campos dentro da mesma, como demonstrado nas linhas 1 e 2.

- **key** – identifica o id do campo, necessário para associar os valores do certificado aos campos definidos no *template*.

- **valueType** – tipo de valor que este campo aceita. Este campo é apenas necessário para o mapeamento dos valores com o *template* do certificado.

- **coordinates** – lista de coordenadas de onde a informação deve ser extraída, ou seja, as linhas que devem ser agrupadas neste valor. O resultado demonstrado no Excerto de Código 13, permite identificar as coordenadas de cada linha e consequentemente obter o valor das mesmas.

- **extract –** lógica a aplicar na extração da informação da linha associada à coordenada definida. Esta função é necessária pois certos campos apenas necessitam de uma ou mais palavras de uma linha completa para mapear o valor do campo. Um exemplo desta extração é o campo definido na linha 15 do Excerto de Código 14, dado que apenas queremos a data presente no fim da linha.

  Ex: "Dates on which the inspection and testing was carried out - 27 Aug 2021"

5.2 Doddle Frontend

A secção que se segue, detalha e demonstra as interfaces implementadas no Portal e na *Mobile App*, de modo a cumprir os requisitos funcionais e não funcionais definidos nas secções 4.1.1 e 4.1.2. Inicialmente são apresentadas as funcionalidades implementadas no Portal, ou seja, as tarefas efetuadas pelos gestores do portal. Por fim, são demonstradas as funcionalidades da *Mobile App*, isto é, as tarefas efetuadas pelos engenheiros nas suas inspeções.

5.2.1 Portal

Esta subsecção pretende descrever e apresentar as interfaces disponibilizadas aos gestores do portal através da utilização do mesmo. Para o contexto do presente projeto, apenas serão descritas as novas funcionalidades à mesma, de forma a possibilitar os requisitos funcionais definidos na secção 4.1.1.

O portal foi desenvolvido utilizando Typescript e React, biblioteca de Javascript que permite o desenvolvimento de aplicações *web*. Os tópicos que se seguem, demonstram as funcionalidades e respetivas interfaces utilizadas pelo gestor do portal nas tarefas do seu dia-a-dia.

Importar certificado

Após aceder ao menu de importação de certificado, o gestor do portal escolhe o tipo de certificado que pretende importar e efetua o *upload* das imagens relativas às páginas do mesmo.

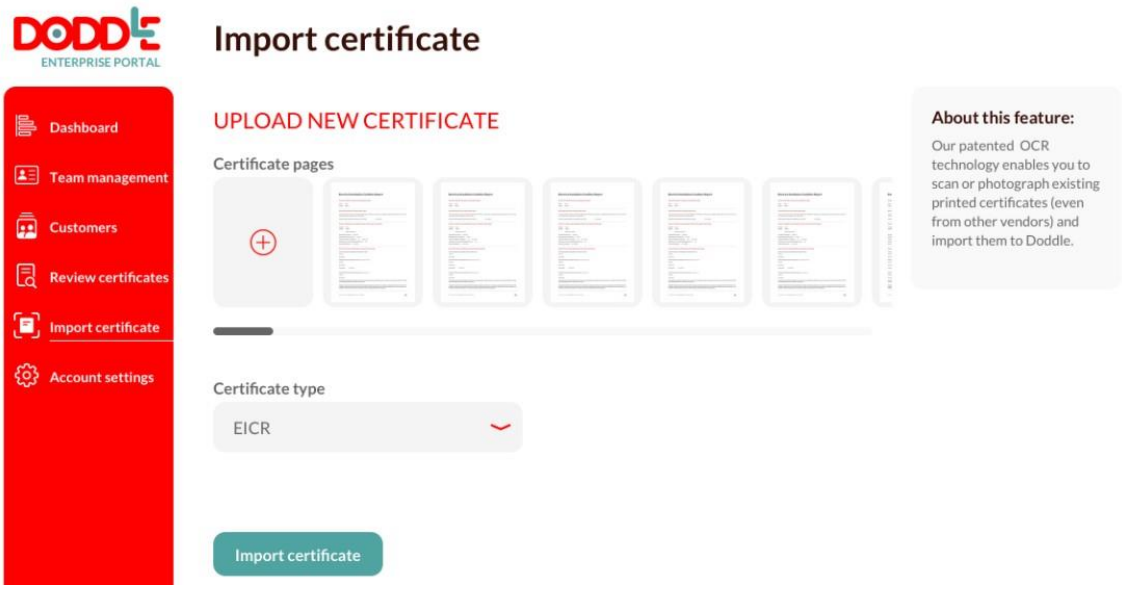


Figura 36 - Interface da importação de certificado

Validar certificado

Após importar um certificado, é apresentado um *preview* em formato PDF do certificado criado baseado na informação extraída e preenchida automaticamente. O gestor do portal pode validar toda a informação e editar a mesma caso necessário.

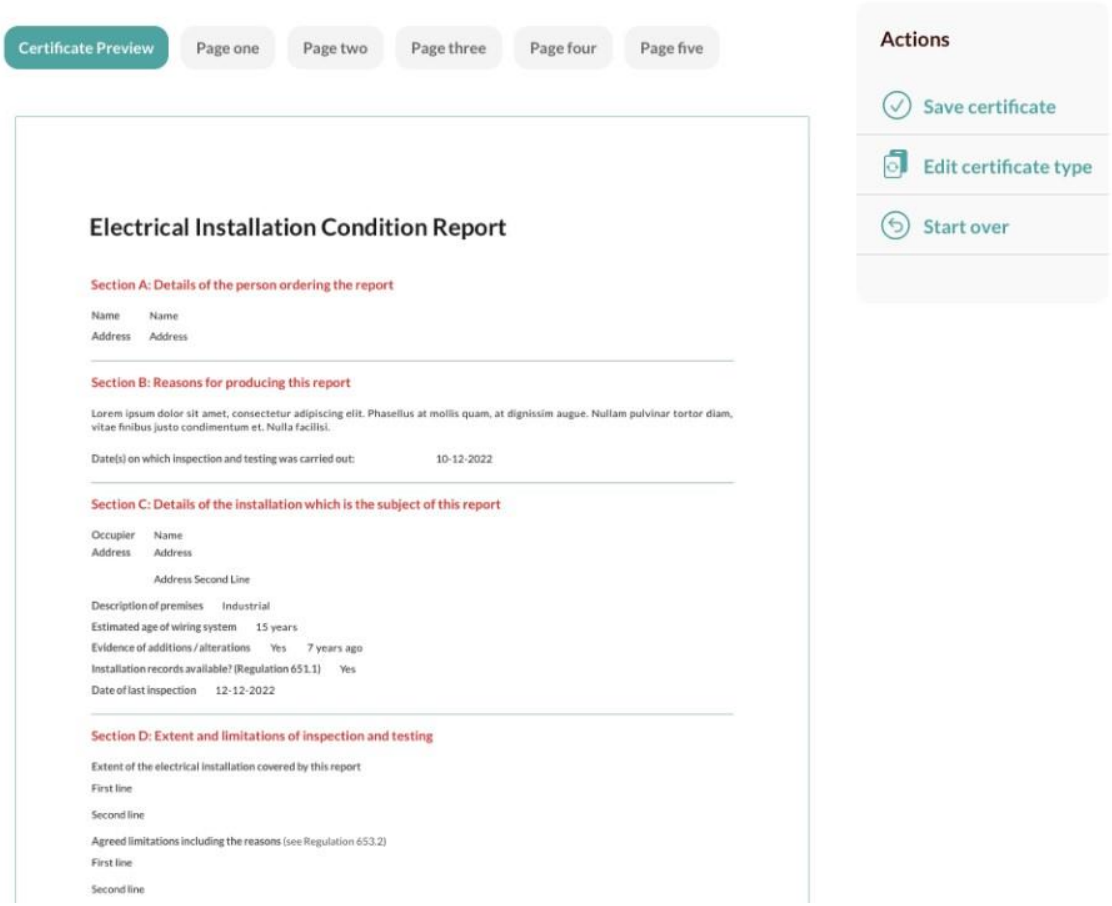


Figura 37 - Interface da validação de certificado

Listar certificados

O gestor tem a possibilidade de consultar todos os certificados importados e validados previamente, de modo a consultar a informação dos mesmos.

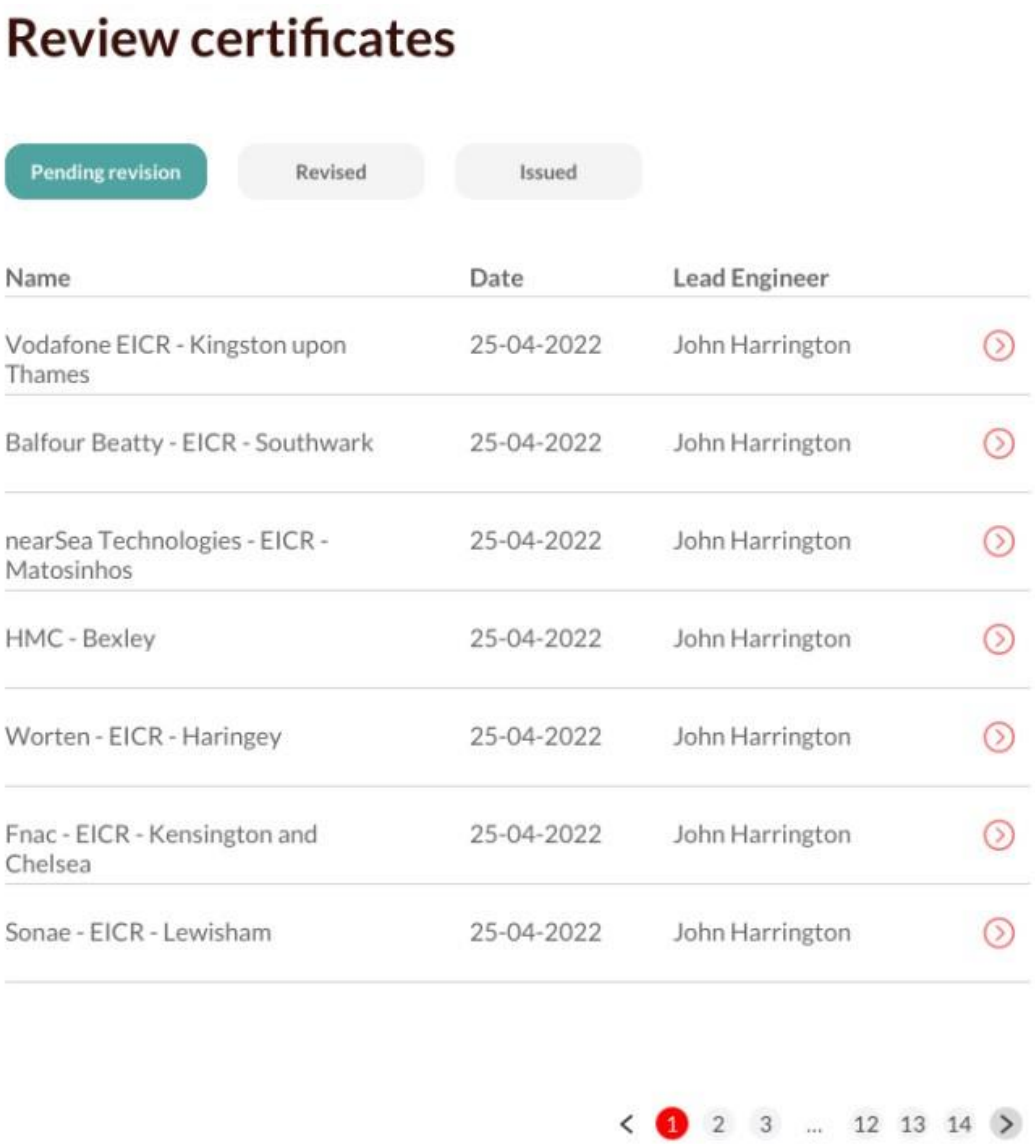


Figura 38 - Interface da listagem de certificados no portal

5.2.2 Mobile App / WebApp

Na subsecção que se segue, são descritas e apresentadas as interfaces disponibilizadas aos engenheiros. Parte das interfaces que se seguem, já se encontravam implementadas, no entanto são funcionalidades importantes após o processo de importação dos certificados, fazendo parte do dia-a-dia dos engenheiros.

Listar certificados

Assim que os engenheiros entram na aplicação é apresentada uma *dashboard* com uma lista de certificados previamente criados pelos mesmos. Esta listagem permite aos engenheiros consultar os certificados importados e emitidos previamente, de forma a efetuar operações como duplicação dos mesmos.

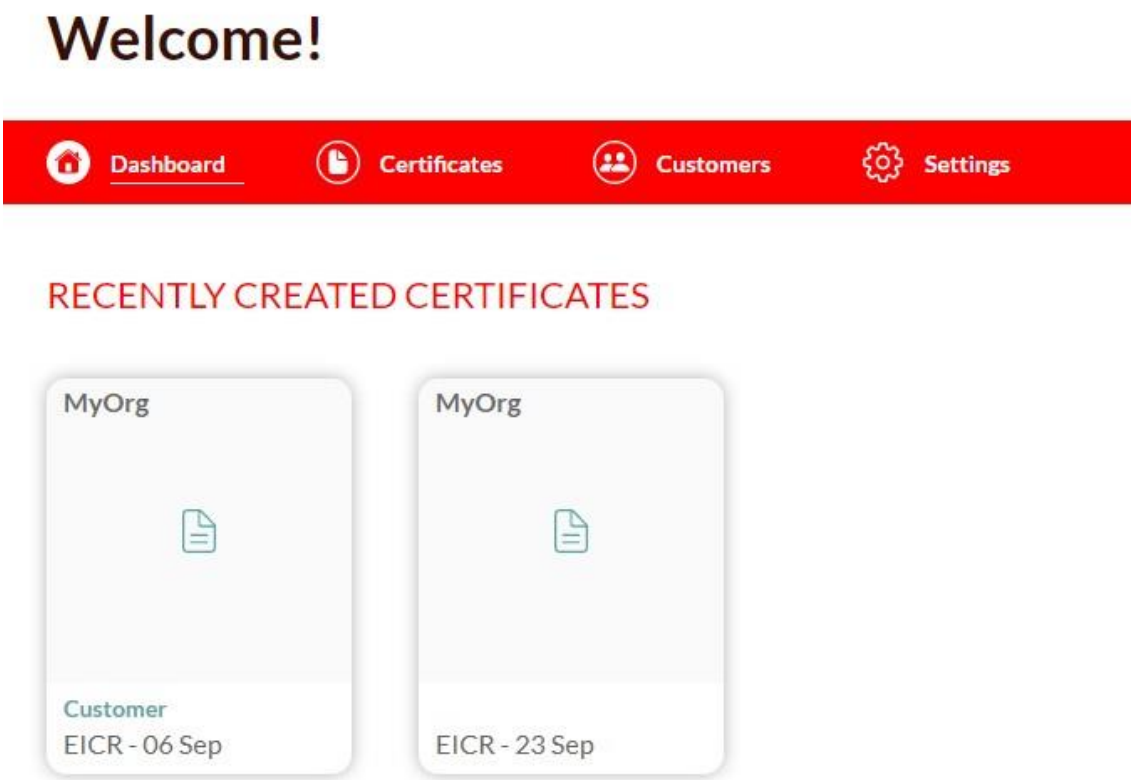


Figura 39 - Interface da listagem de certificados nas apps

Duplicar certificado

Assim que um engenheiro abre um certificado emitido previamente, tem a possibilidade de duplicar o mesmo de modo a reutilizar a informação já preenchida em inspeções anteriores, aumentando consideravelmente a eficiência das suas inspeções.

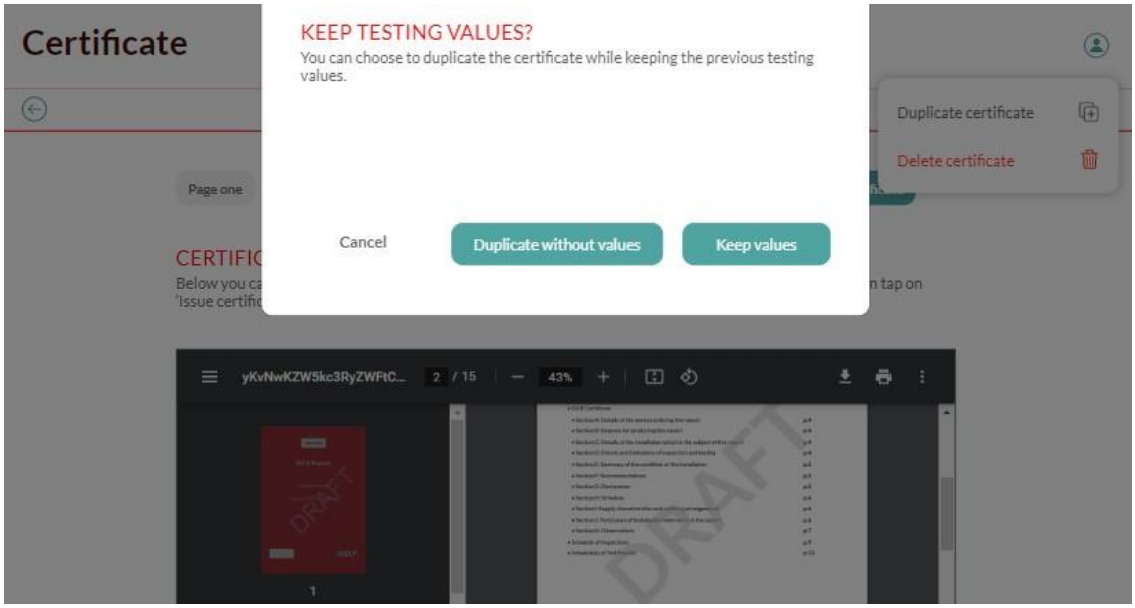


Figura 40 - Interface para duplicação de certificados

Emitir certificado

Após duplicar um certificado e preencher o mesmo por completo, os engenheiros têm a possibilidade de emitir os certificados no fim das suas inspeções, de forma a comprovar que as localizações se encontram inspecionadas e de acordo com a legislação.

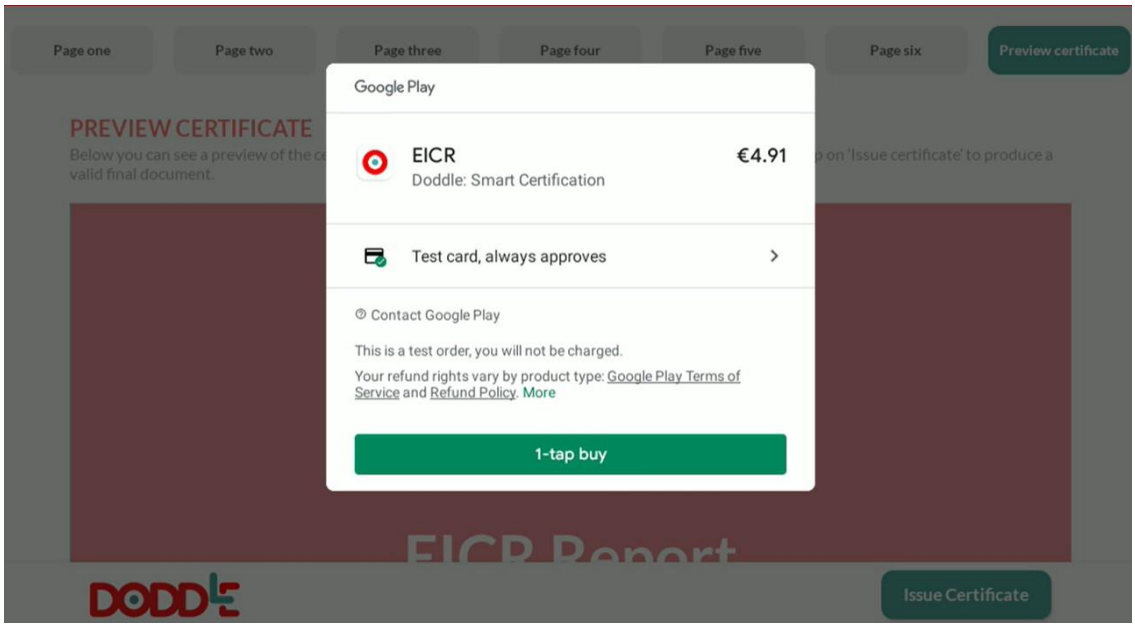


Figura 41 - Interface para emissão de certificados

5.3 DevOps

Ao longo do processo de desenvolvimento, foram utilizadas diversas ferramentas, responsáveis por automatizar e monitorizar todas as fases de desenvolvimento de *software*, de modo a contribuir para a produtividade e qualidade do projeto desenvolvido.

A secção que se segue, apresenta e descreve todas as ferramentas e serviços utilizados durante o processo de desenvolvimento e implantação da presente solução. Inicialmente, é feita uma análise ao processo de CI / CD, isto é, aos passos efetuados sempre que ocorre uma alteração no código de cada repositório. Por fim, é detalhada a infraestrutura atual da solução desenvolvida e os passos necessários para a implantação da mesma.

5.3.1 CI / CD

Esta subsecção diz respeito às pipelines implementadas em cada um dos repositórios, de modo a automatizar todas as fases de desenvolvimento do projeto.

Todos os projetos e respetivos repositórios desenvolvidos pela nearSea Technologies, encontram-se alojados no Github. O Github possui ferramentas de CI / CD que permitem automatizar o processo de testes e implantação dos serviços desenvolvidos, como a plataforma Github Actions. Através desta plataforma é possível especificar os *workflows* que devem ser executados sempre que é efetuada uma alteração em qualquer um dos serviços.

Embora cada repositório necessite de ter a sua própria pipeline definida através de ficheiros yaml, a implementação é idêntica em todas elas, dado que partilham a mesma infraestrutura e todas as aplicações usam Node.js e Typescript como tecnologias de desenvolvimento. Dessa forma, surgiu a necessidade de criar *workflows* genéricos que fossem partilhados por todas as pipelines, de modo a evitar a duplicação do código na implementação das mesmas.

O Excerto de Código 15 representa a pipeline implementada na Doddle OCR API utilizando os *workflows* responsáveis por servir de base para todos os repositórios.

```yaml
jobs:
  docker:
    uses: NearSeaTechnologies/workflows/.github/workflows/docker-
github.yaml@1.7.5
    secrets:
          username: ${{ github.actor }}
      password: ${{ secrets.GITHUB_TOKEN }}
      ALL_SECRETS: ${{ toJSON(secrets) }}
      PERSONAL_GITHUB_TOKEN: ${{ secrets.PERSONAL_GITHUB_TOKEN }}
      OTHER_ENVS: |
        {}

  node:
    uses:
NearSeaTechnologies/workflows/.github/workflows/node.yaml@1.7.5
    with:
      run_tests: true
      run_prettier: true

  terraform-validation:
    uses: NearSeaTechnologies/workflows/.github/workflows/terraform-
validation.yaml@1.7.5

  terraform-plan:
    uses: NearSeaTechnologies/workflows/.github/workflows/terraform-
plan.yaml@1.7.5
    with:
      inject_other_envs: true
    needs: docker
    secrets:
      ALL_SECRETS: ${{ toJSON(secrets) }}
      OTHER_ENVS: |
        {
          "TF_VAR_DOCKER_IMAGE":
"${{ needs.docker.outputs.docker_tag }}",
          "TF_VAR_REGISTRY_USERNAME": "${{ github.actor }}",
          "TF_VAR_REGISTRY_PASSWORD":
"${{ secrets.PERSONAL_GITHUB_TOKEN }}"
        }
      terraform_cloud_tokens: ${{ secrets.TF_CLOUD_TOKEN }}
```

Excerto de Código 15 – Ficheiro yaml de configuração da pipeline na Doddle OCR API

Como demonstrado anteriormente, a *pipeline* de cada repositório é composta por quatro passos essenciais. Na listagem abaixo é detalhada a responsabilidade de cada um deles:

- **docker –** este passo é responsável por compilar todas alterações e guardar as mesmas dentro de uma imagem. Esta imagem é posteriormente utilizada para levantar um *container* com a aplicação e disponibilizar a mesma aos utilizadores.

- **node –** passo onde correm ambos os testes unitários da aplicação e as ferramentas de controlo de qualidade como prettier e eslint.

- **terraform-validation –** responsável por validar todas as alterações a nível de infraestrutura através do Terraform.

- **terraform-plan –** passo onde ocorre a implantação das novas alterações para a infraestrutura existente utilizando o Terraform.

Na Figura 42, é demonstrado um exemplo da *pipeline* na Doddle OCR API, após efetuar alterações na *branch* de *development*.

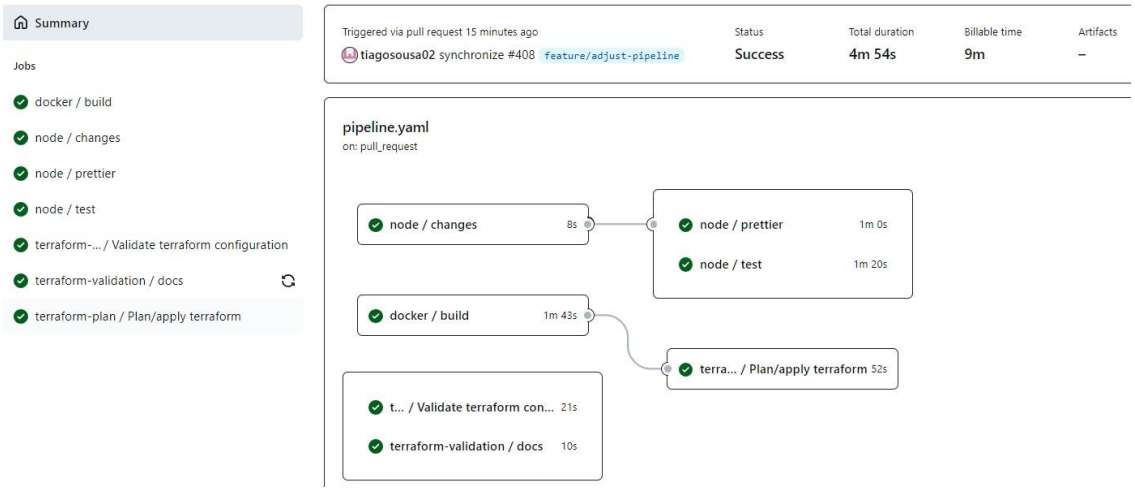


Figura 42 – Sumário da pipeline na Doddle OCR API

Assim que a *pipeline* termina com sucesso, é gerado um relatório no *pull request* associado com os resultados dos testes e das análises efetuadas pelas diferentes ferramentas, como demonstrado na Figura 43.

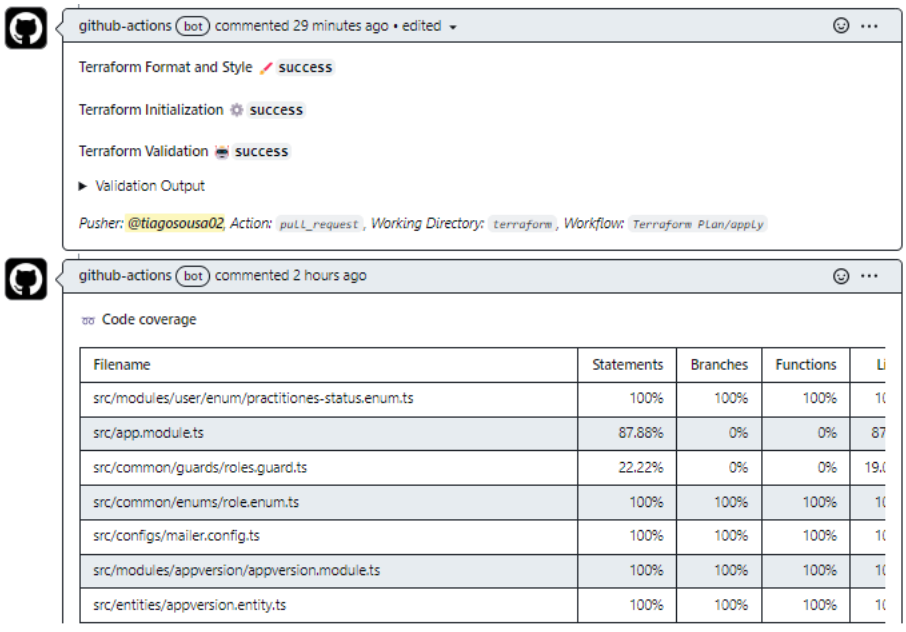


Figura 43 - Resultados da pipeline da Doddle OCR API

5.3.2 Infrastructure

A subsecção que se segue apresenta e descreve os serviços e respetiva implementação, utilizados para implantar as aplicações desenvolvidas no presente projeto.

Docker

De forma a implantar cada um dos serviços, foi utilizado o Docker, dado que esta ferramenta permitiu criar imagens com versões funcionais das aplicações, destinadas a correr dentro de *containers* através do Kubernetes.

Com o intuito de efetuar o versionamento das imagens, isto é, registar todas as alterações efetuadas às aplicações, foi utilizado o Docker Registry para guardar as mesmas e utilizá-las na implantação dos serviços. Na Figura 44 encontra-se o *package* associado à Doddle OCR API, serviço do GitHub onde são armazenadas todas as imagens criadas na mesma.

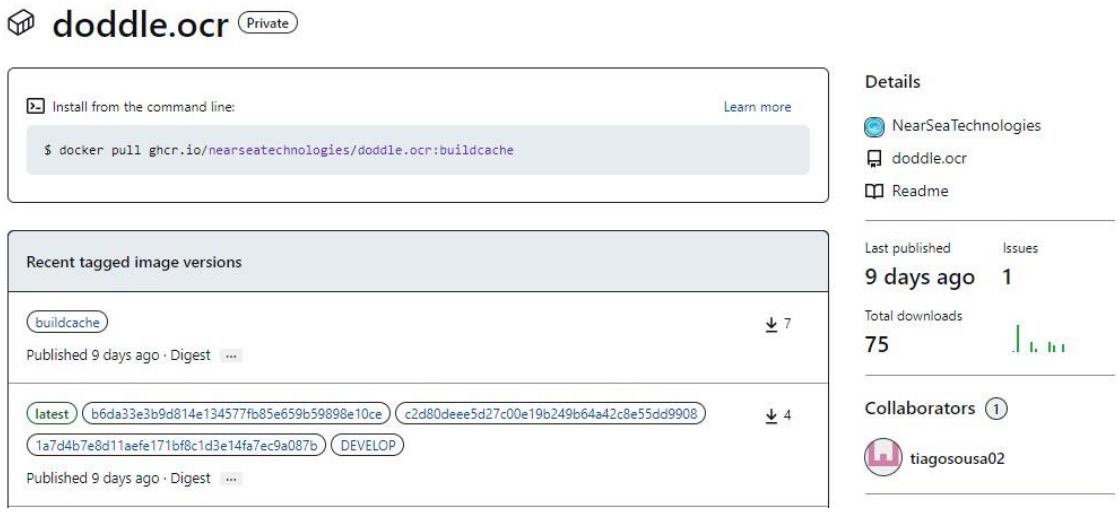


Figura 44 – Package da Doddle OCR API no Docker Registry

O Docker Registry é uma das ferramentas do Github que permite armazenar e versionar todas as imagens criadas em cada repositório. Posteriormente estas imagens podem ser transferidas e utilizadas localmente, ou implantadas em *containers* de Kubernetes para disponibilizar aos utilizadores. Para criar as imagens em cada uma das aplicações é necessário um Dockerfile, onde são especificadas as ações para compilar a aplicação e criar uma versão funcional da mesma. No Excerto de Código 16, encontra-se representado o Dockerfile criado para a Doddle OCR API.

```
1  FROM node:16.15.1-alpine AS base
2
3  WORKDIR /app
4
5  COPY package*.json tsconfig*.json ./
6
7  FROM base as build
8
9  WORKDIR /app
10
11 RUN npm ci
12
13 COPY ./src/ ./src/
14
15 RUN npm run build
16
17 FROM build as dev
18
19 ENV NODE_ENV=development
20
21 CMD ["npm", "run", "start:debug"]
22
23 FROM base as release
24
25 WORKDIR /app
26
27 ENV NODE_ENV=production
28
29 COPY --from=build /app/dist /app/dist
30 COPY --from=build /app/node_modules /app/node_modules
31 COPY --from=build /app/arma-connect*.json /app/arma-connect*.json
32
33 RUN npm prune --production
34
35 CMD ["npm", "run", "start:prod"]
```

Excerto de Código 16 – Dockerfile da Doddle OCR API

Como demonstrado anteriormente, o Dockerfile especifica as ações a efetuar durante a criação da imagem de cada aplicação. Inicialmente, são copiados todos os ficheiros para a imagem e posteriormente são instaladas todas as dependências da aplicação. Por fim, é exposta a porta na qual o servidor deve correr e corrido o comando que levanta a aplicação e a disponibiliza aos utilizadores.

Terraform

Outra ferramenta utilizada no processo de implantação da infraestrutura foi o Terraform. O Terraform é uma ferramenta de código aberto que permite especificar a infraestrutura através de código, utilizando uma linguagem declarativa denominada de *HashiCorp Configuration Language* (HCL).

Esta ferramenta foi utilizada com o intuito de manter um versionamento de toda a infraestrutura e facilitar a sua manutenção através de código. Cada repositório possui uma pasta **terraform**, onde se encontra toda a configuração da infraestrutura da aplicação, especificando bases de dados, camadas de redes, variáveis de ambiente e os *deployments* de Kubernetes. A Figura 45 demonstra a pasta criada na Doddle OCR API, de forma a especificar toda a sua infraestrutura.

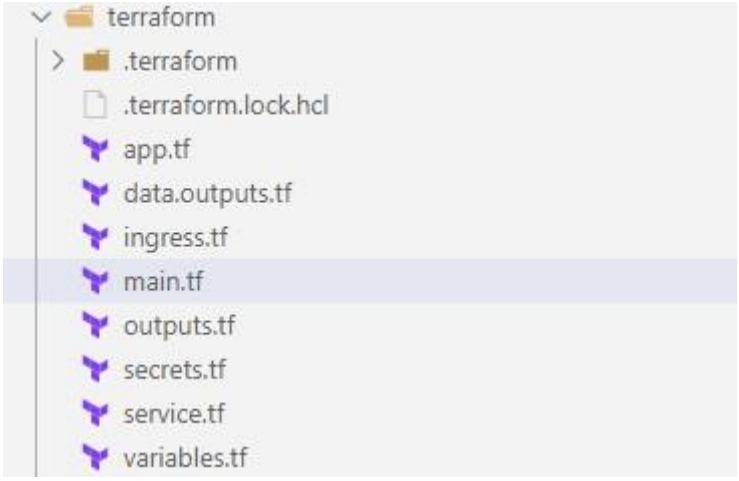


Figura 45 – Pasta de configuração de Terraform da Doddle OCR API

Além da configuração efetuada no repositório, cada aplicação possui ainda uma *workspace* na Terraform Cloud, onde são definidas as variáveis de ambiente a ser utilizadas durante o processo de implantação da infraestrutura. A Figura 46 apresenta as variáveis definidas para o ambiente de DEV da Doddle OCR API.

Workspace variables (4)

Variables defined within a workspace always overwrite variables from variable sets that have the same type and the same key. Learn more about variable set precedence.

Key	Value
SERVICE_ACCOUNT_DATA SENSITIVE	*Sensitive - write only*
API_KEY SENSITIVE	*Sensitive - write only*
BASE_URL	ocr-
INFRASTRUCTURE_WORKSPACE	infrastructure-dev

Figura 46 – Variáveis de Terraform para a Doddle OCR API

Assim que toda a infraestrutura da aplicação é definida utilizando a configuração do Terraform, é executada nas *pipelines* demonstradas na secção 5.3.1, de forma a criar ou atualizar, caso já exista, todos os serviços necessários para implantação das aplicações.

DigitalOcean

De forma a disponibilizar todas as aplicações aos utilizadores, foi necessário fazer a sua implantação. Para isso, utilizou-se o DigitalOcean, serviço de nuvem que permitiu hospedar todos os serviços numa máquina utilizando as imagens de Docker mencionadas anteriormente.

Através da configuração de Terraform demonstrada anteriormente, toda a infraestrutura dos serviços foi criada automaticamente no DigitalOcean sempre que uma alteração fosse efetuada nas aplicações. Na Figura 47 são demonstrados os clusters criados para o presente projeto.

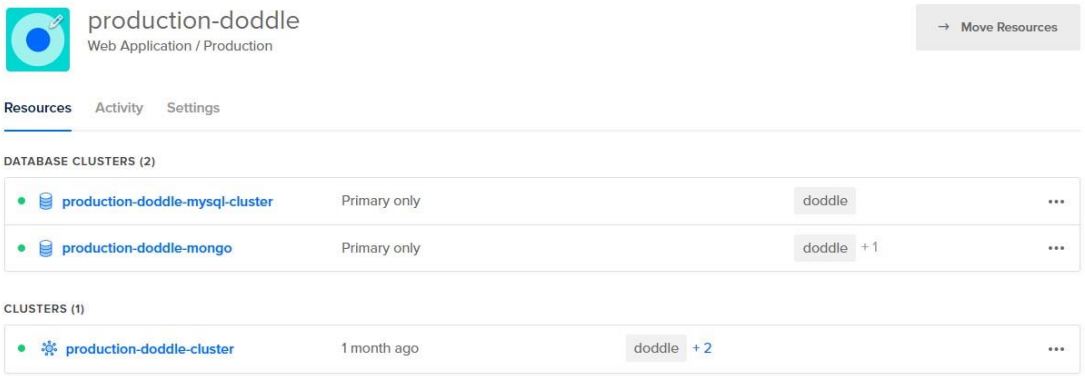


Figura 47 – *Clusters* do projeto no DigitalOcean

De forma a hospedar todas as aplicações desenvolvidas, foi criado um cluster de Kubernetes, onde se encontram diversas máquinas responsáveis por executar as imagens das aplicações e expor as mesmas. Dada a necessidade de persistência dos dados foram também criados clusters para cada uma das bases de dados necessárias, de modo a possuir funcionalidades como cópias de segurança da informação.

Por razões de confidencialidade não é possível detalhar todos os serviços criados dentro cluster do projeto, no entanto, este é composto por diversos *deployments*, responsáveis por levantar as imagens de cada aplicação. Estes *deployments* são posteriormente mapeados através de um Kubernetes Ingress, responsável por expor as aplicações através dos domínios definidos.

5.4 Testes

Esta secção aborda todo o contexto de testes que foram implementados e acompanharam todo o processo de desenvolvimento, de modo testar a qualidade e estabilidade da solução desenvolvida. Cada tipo de teste possuiu um papel importante para o sucesso do projeto. Dessa forma, foram desenvolvidos testes unitários, testes de integração e testes ponta-a-ponta.

5.4.1 Testes Unitários

Em particular, o desenvolvimento de testes ajuda a identificar falhas na lógica implementada, contribuindo para a qualidade e estabilidade do código que constitui o sistema. Desta prática, advém também a vantagem de induzir o programador a escrever código fácil de testar, ou seja, separando as responsabilidades de cada classe resultando num baixo acoplamento do mesmo.

Optou-se por recorrer ao uso de dados fictícios (*mocks*), que representassem os objetos utilizados em produção. Dessa forma, garantiu-se que dados sensíveis não fossem comprometidos e partilhadas com os utilizadores.

Backend

No contexto das APIs desenvolvidas, foram testadas as funcionalidades utilizando a libraria de testes ts-mockito. Esta, fornece várias funções que permitem testar todo o código implementado.

No Excerto de Código 17, é demonstrado um teste unitário correspondente à funcionalidade de importação de ficheiros.

```
1   import { MockitoTest } from 'nestjs-auto-tsmockito';
2
3
4   beforeEach(async () => {
5     const app = await MockitoTest.createMockedModule(
6       {
7         controllers: [OCRController],
8       },
9       {
10        imports: [AppModule],
11      },
12    ).compileMocked();
13
14    ocrControllerMock = app.get<OCRController>(OCRController);
15    ocrServiceMock = app.getMock(OCRService);
16  });
17
18  describe('Scan Document', () => {
19    it('should create OCR', async () => {
20      const inputOCR = [ocrDocumentFixture()];
21      const certificate = Certificate.EICR;
22
23      when(ocrServiceMock.scanAndMapCertificate(certificate,
inputOCR)).thenResolve();
24
25      await ocrControllerMock.scanAndMapCertificate(certificate,
inputOCR);
```

Excerto de Código 17 – Testes unitários desenvolvidos nas APIs

Inicialmente são criados objetos *mock*, de forma a controlar o comportamento das funções invocadas. Posteriormente são comparados os resultados obtidos com os *mocks* criados, de modo a testar o funcionamento da lógica implementada

Portal

No portal desenvolvido, foram também implementados testes unitários utilizando a libraria Jest, sendo esta destinada à realização de diferentes tipos de testes em aplicações React. A partir do Jest, é possível criar uma cópia da interface desenvolvida, de modo a testar o comportamento dos componentes que constituem a aplicação.

No Excerto de Código 18, encontra-se um exemplo de um teste unitário realizado para testar o componente ***ImportCertificate*** da aplicação.

```
1  import ImportCertificate from './components/ImportCertificate';
2
3  describe('Render Import Certificate', () => {
4    it('renders without crashing', () => {
5      shallow(<ImportCertificate />);
6    });
7
8    it('renders ImportCert header', () => {
9      const wrapper = shallow(<ImportCertificate />);
10
11     const mainHeader = <h1>Import Certificate</h1>;
12     expect(wrapper.contains(mainHeader)).toEqual(true);
13   });
14 });
```

Excerto de Código 18 - Testes unitários desenvolvidos no Portal

O componente apresentado anteriormente, é responsável por apresentar a página de importação de certificados. De modo a validar o mesmo, foi utilizada a função "*shallow*" (linha 9, Excerto de Código 18), permitindo criar uma cópia do componente e validar a sua renderização, de forma a testar todas as suas funcionalidades.

Na Figura 48 é demonstrado o resultado da execução dos testes unitários desenvolvidos. Todos as funcionalidades e respetiva lógica foram validadas, de modo a identificar possíveis falhas e a contribuir para a qualidade do código desenvolvido.

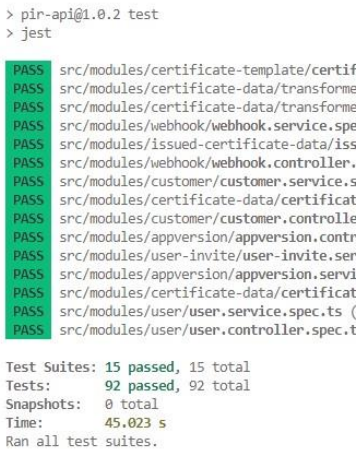


Figura 48 - Resultado dos testes unitários desenvolvidos

5.4.2 Testes de Integração

Os testes de integração, consistem em assegurar que a integração dos diferentes componentes do sistema funciona corretamente, ou seja, testam a comunicação entre os mesmos.

De modo a implementar os testes de integração nas APIs desenvolvidas, foram utilizadas as librarias de testes do NestJS e o Supertest. Foi também utilizada a ferramenta Postman, de modo a testar e documentar todos os pedidos disponibilizados pelas APIs. No Excerto de Código 19, encontra-se um dos testes de integração implementados na Doddle API.

```
1  import request from 'supertest';
2
3  describe('OCR Controller (e2e)', () => {
4    let app: INestApplication;
5
6    beforeEach(async () => {
7      const moduleFixture: TestingModule = await
Test.createTestingModule({
8        imports: [AppModule],
9      }).compile();
10
11     app = moduleFixture.createNestApplication();
12     await app.init();
13   });
14
15   it('/ocr (POST)', async () => {
16     const images = ocrDocumentFixture();
17     const responseData = certificateDataFixture();
18
19     return request(app.getHttpServer())
20       .post('/ocr')
21       .send(images)
22       .set('Accept', 'application/json')
23       .expect('Content-Type', /json/)
24       .expect(200)
25       .then((response) => {
26         assert(response.body, responseData);
```

Excerto de Código 19 - Teste de integração implementado na Doddle OCR API

Inicialmente é efetuado um pedido à API (linha 19, Excerto de Código 19), de modo a testar a resposta retornada pela mesma. Para que o pedido efetuado fosse o mais semelhante possível aos pedidos efetuados pelos utilizadores no ambiente de produção, foram criadas *fixtures*. As *fixtures* são objetos JSON com a informação a ser enviada e esperada na resposta da API, como imagens a analisar e certificados mapeados através da informação extraída.

Paralelamente aos testes implementados na API, criou-se uma coleção no Postman com o intuito de automatizar o processo de testes e documentar todos os pedidos disponibilizados pelas APIs, como demonstrado na Figura 49.

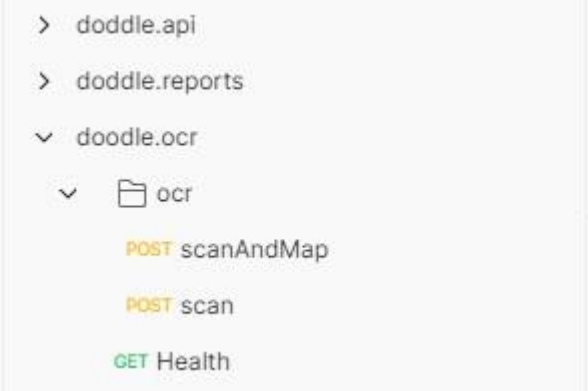


Figura 49 - Coleções de Postman das APIs desenvolvidas

Tal como na API, os testes desenvolvidos através do Postman, visam testar as respostas obtidas pelos diferentes pedidos, automatizando o processo de testes de integração.

A partir do Excerto de Código 20, é possível observar um dos testes criados na coleção mencionada anteriormente.

```
1  const responseJson = pm.response.json();
2
3  pm.test("Successful POST request", () => {
4     pm.response.to.have.status(201);
5     pm.response.to.have.header("Content-Type");
6  });
7
8  pm.test("Correct response body", () => {
9   pm.expect(responseJson).to.not.be.empty;
10  pm.expect(responseJson).to.include(response);
11 });
12
```

Excerto de Código 20 - Teste de integração para a importação de certificados

O teste demonstrado anteriormente pretende testar a importação de certificados, efetuando a submissão de várias imagens e comparando a sua resposta com os dados definidos.

A Figura 50 apresenta os resultados obtidos na execução dos testes de integração definidos na coleção mencionada anteriormente.

Figura 50 – Resultado dos testes de integração no Postman

5.4.3 Testes Não Funcionais

Os testes não funcionais visam testar a aplicação e validar os requisitos não funcionais definidos na secção 4.1.2.

Desempenho e Confiabilidade

De modo a testar o desempenho e confiabilidade da solução implementada, utilizou-se uma libraria de testes de carga, denominada Artillery. Esta libraria permitiu simular pedidos de diversos utilizadores em simultâneo à API, de forma a testar o seu desempenho. Os testes foram conduzidos em modo *benchmark*, ou seja, sem intervalo de tempo entre os pedidos. Esta decisão foi tomada por forma a testar o sistema perante cargas intensivas.

No Excerto de Código 21, encontra-se o ficheiro de configuração criado para a execução dos testes de carga. Definiu-se o URL para o qual os pedidos seriam efetuados, sendo que o URL definido corresponde ao ambiente de testes do projeto. Além do URL, definiu-se a duração do teste e o número de pedidos por segundo, sendo estes os valores aconselhados pela libraria utilizada. Por fim, foi adicionado um *processor,* responsável por mapear o pedido de importação de certificado enviado à API com as imagens do certificado EICR.

```
config:
  target: 'https://test.doddle-app.co.uk/api/v1'
  processor: 'cert-images-processor.js'
  phases:
    - duration: 60
      arrivalRate: 3
      name: 3 simultaneous users per second
```

Excerto de Código 21- Configuração para a execução dos testes de carga à Doddle API

Feita a configuração do teste, procedeu-se à especificação do cenário de testes a executar como demonstrado no Excerto de Código 22.

```
scenarios:
  - name: 'Import Certificate'
    flow:
      - post:
          url: '/auth'
          json:
            username: 'tiago.sousa@nearseatech.io'
            password: 'test123!'
          capture:
            - json: '$.access_token'
              as: 'acesss_token'
      - post:
          url: '/ocr?certificate=EICR'
          beforeRequest: addCertificateImages
          headers:
            authorization: 'Bearer {{ acesss_token }}'
```

Excerto de Código 22 - Cenário de teste de carga criados para a Doddle API

O cenário configurado, é composto por dois *endpoints* distintos, sendo o primeiro respetivo à autenticação na API, de forma a obter o *token* a enviar para os restantes pedidos. De seguida foi efetuado o pedido de importação do certificado utilizando o *token* obtido previamente.

O Excerto de Código 23, demonstra o resultado obtidos após executar os testes configurados anteriormente.

```
--------------------------------
Summary report @ 19:31:23(+0100)
--------------------------------

http.codes.201: ............................ 360
http.request_rate: ......................... 6/sec
http.requests: ............................. 360
http.response_time:
  min: ...................................... 215
  max: ...................................... 9532
  median: ................................... 1249.1
  p95: ...................................... 7407.5
  p99: ...................................... 9047.6
http.responses: ............................ 360
vusers.completed: .......................... 180
vusers.created: ............................ 180
vusers.created_by_name.Import Certificate: .. 180
vusers.session_length:
  min: ...................................... 1637.9
  max: ...................................... 10467.5
  median: ................................... 3395.5
  p95: ...................................... 9999.2
  p99: ...................................... 10201.2
```

Excerto de Código 23 - Resultado dos testes de carga efetuados à Doddle API

Foram efetuados cerca de 360 pedidos dentro de um minuto, ou seja, 6 pedidos por segundo efetuados por 3 utilizadores diferentes.

Tendo em conta que o pedido de importação de certificados envolve um elevado nível de processamento por parte da API e a mesma manteve uma disponibilidade de 100%, é possível concluir que a execução deste teste foi efetuada com sucesso. Além disso, o tempo de resposta também apresentou valores bastante positivos, sendo que média do mesmo foi de aproximadamente 3.4 segundos, face a 360 pedidos recebidos.

Sendo assim, os requisitos não funcionais de desempenho e confiabilidade definidos foram cumpridos sem qualquer problema.

Funcionalidade e usabilidade

Em termos de usabilidade e funcionalidade, também foram cumpridos certos requisitos não funcionais definidos, como a documentação de todos os pedidos disponibilizados pela API. Além disso a documentação encontra-se disponibilizada utilizando o idioma inglês, como demonstrado na Figura 51.

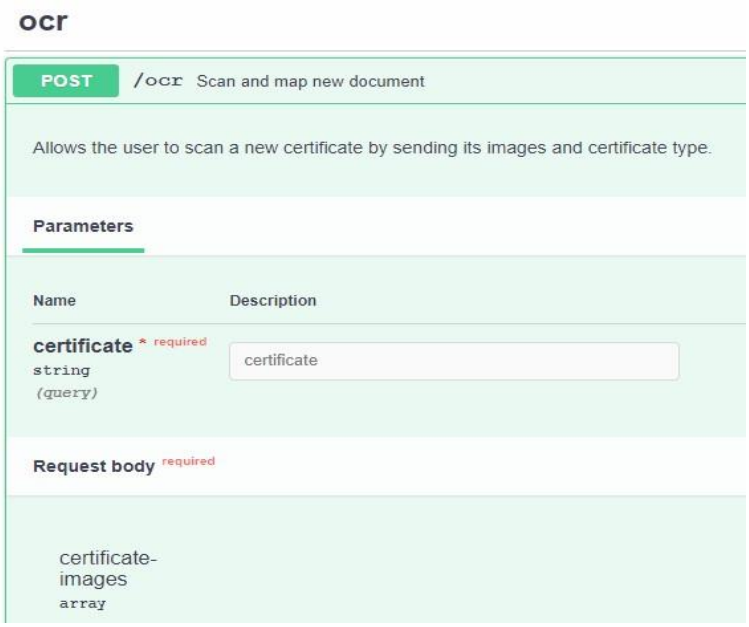


Figura 51 – Documentação do *endpoint* de importação de certificados

6 Experimentação e Avaliação

De forma a garantir a qualidade da solução desenvolvida, foram adotadas diversas metodologias e estratégias, com o intuito que a mesma se encontrasse de acordo com os padrões de qualidade definidos pela organização.

No capítulo que se segue, são inicialmente identificados os indicadores que serviram de base para a avaliação da solução e posteriormente descritas as metodologias adotadas, quer a nível técnico como funcional, para avaliar os mesmos. Por fim, é apresentada a qualidade final do projeto, tal como as ferramentas utilizadas para a determinar.

6.1 Identificação dos indicadores

Todo o *software* desenvolvido na nearSea Technologies deve respeitar altos padrões de qualidade em diferentes aspetos, nomeadamente, adaptabilidade, manutenibilidade, desempenho, segurança e funcionalidade. De modo, a assegurar que a solução desenvolvida se encontrava de acordo com os mesmos, foram identificados diversos indicadores de qualidade.

Nesse sentido, os indicadores foram divididos em duas vertentes a serem analisadas através de ferramentas apropriadas:

- Numa vertente mais técnica, será avaliada a qualidade da solução desenvolvida, a nível de código e boas práticas de *design*.

- Numa vertente a nível funcional, será avaliada de acordo com os requisitos funcionais e não funcionais definidos nas secções 4.1.1 e 4.1.2.

6.2 Metodologia de Avaliação

A avaliação da solução a desenvolver será dividida em duas fases. Numa fase inicial, efetuada durante o processo de desenvolvimento, serão utilizadas diversas ferramentas e metodologias que permitam garantir de qualidade de *software*. Entre estas ferramentas destacam-se o SonarQube, o Eslint e o Prettier.

O Eslint e o Prettier são ferramentas que permitem que permitem validar e condicionar a implementação do código, através da definição de regras na sua configuração. Estas ferramentas são frequentemente utilizadas em paralelo com o SonarQube, permitindo garantir a qualidade do *software* desenvolvido.

O SonarQube é uma ferramenta que efetua análise da qualidade do código implementado de forma automatizada. Após configurada, esta ferramenta permite detetar *bugs*, vulnerabilidades e potenciais problemas de performance no código das aplicações desenvolvidas. Feita a análise, é gerado um relatório que pode ser visualizado a partir da interface demonstrada na Figura 52. Esta, fornece um painel onde é possível visualizar toda a informação detalhada acerca de cada análise efetuadas.

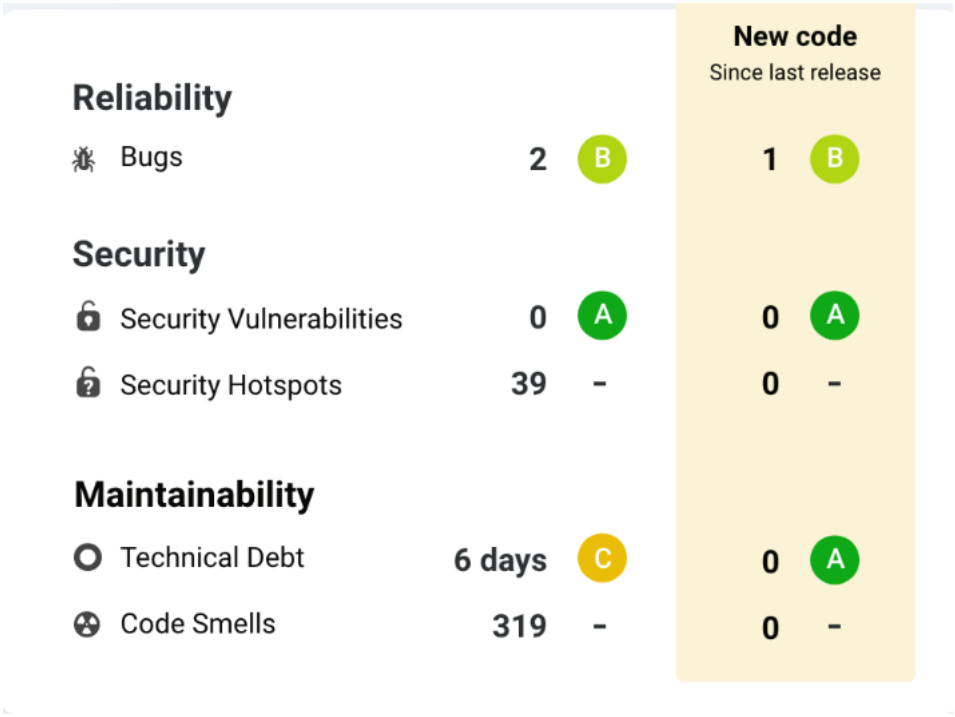


Figura 52 - Painel de análise do código, do SonarQube

Nesse sentido, o objetivo inicial será implementar uma *pipeline* que permita validar todas as alterações efetuadas no código da solução, integrando as ferramentas mencionadas previamente. No entanto, para efeitos de avaliação final apenas o SonarQube permitirá obter um relatório a analisar, de modo a avaliar a qualidade da solução implementada a nível técnico.

Numa fase final, será efetuada uma avaliação a nível funcional da solução, de modo a validar que todos os requisitos definidos nas secções 4.1.1 e 4.1.2 se encontram implementados corretamente. Esta avaliação será realizada através da ferramenta *Quantitative Evaluation Framework* (QEF).

6.3 Avaliação da qualidade

A secção que segue pretende avaliar a qualidade final do projeto desenvolvido utilizando as ferramentas de qualidade identificadas, sendo elas o Sonarqube e a *Quantitative Evaluation Framework* (QEF).

6.3.1 SonarQube

A utilização de ferramentas de controlo de qualidade na implementação da presente solução, permitiu garantir a qualidade da mesma em todos os processos de desenvolvimento e garantir que parte dos requisitos não funcionais definidos na 4.1.2.

Na Figura 53, é apresentado o relatório gerado para a Doddle OCR API, visto que foi a única aplicação desenvolvida de raiz e apresenta todas as funcionalidades de OCR desenvolvidas no presente projeto.

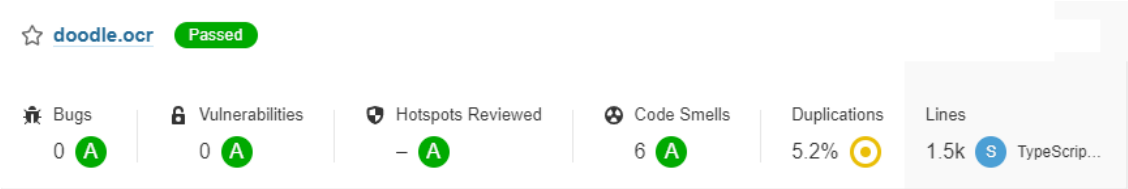


Figura 53 - Análise do SonarQube na Doddle OCR API

Como demonstrado anteriormente, todas as métricas foram classificadas com a nota **A**, indicando que o código implementado se encontro de acordo com os padrões de qualidade definidos pelo SonarQube.

Na listagem que se segue, é descrito o significado de cada umas das métricas apresentadas anteriormente:

- **Bugs (Confiabilidade) –** representa o número de possíveis falhas encontradas no código implementando.
- **Vulnerabilities (Funcionalidade) –** diz respeito ao número de vulnerabilidades a nível de segurança que se encontram implementadas no código da aplicação.
- **Hotspots Reviewed (Funcionalidade) –** possíveis falhas de segurança que necessitam de análise por parte dos desenvolvedores, de forma a reconhecer as vulnerabilidades da mesma.

- **Code Smells (Suportabilidade) –** representa a dívida técnica implementada no código da aplicação. Esta, é identificada através de más práticas de código que podem causar bugs ou vulnerabilidades no futuro.

- **Duplications –** retrata todo o código duplicado existente na aplicação que deve ser reestruturado, de modo a reutilizar o já existente.

Considerando as métricas onde foram identificados problemas na aplicação, procedeu-se a uma análise dos mesmos, com o intuito de determinar a sua causa e possível solução, caso necessário.

Code Smells

Foram identificados cerca de seis *Code Smells* na API, no entanto após análise dos mesmos constatou-se que a severidade dos mesmos era do tipo *Info* e diziam respeito a comentários deixados no código.

A Figura 54 apresenta a descrição dos problemas encontrados tal como a sua severidade.

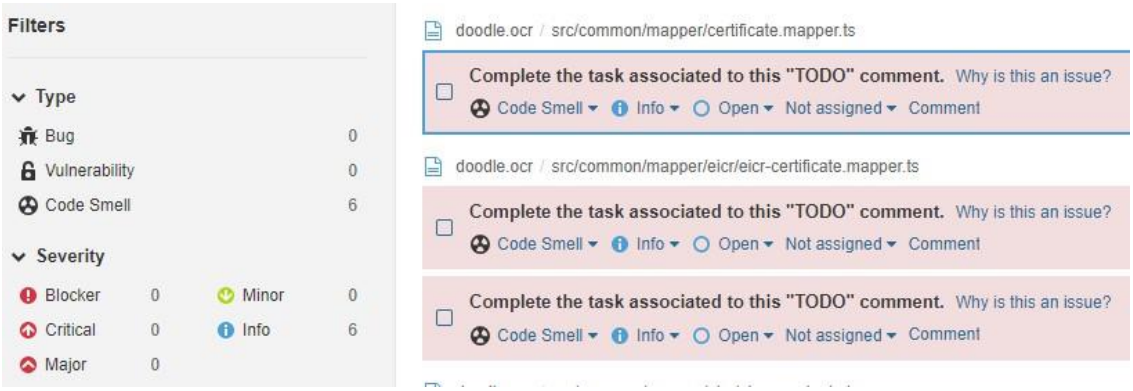


Figura 54 - *Code Smells* identificados na Doddle OCR API

Como demonstrado anteriormente, a severidade dos *Code Smells* identificados era do tipo *Info,* ou seja, apenas servem para informar o programador que as tarefas associadas aos mesmos devem ser implementadas. Dado que os comentários foram escritos de modo a identificar o trabalho futuro na aplicação, optou-se por deixar os mesmos.

Duplications

Além dos *Code Smells* descritos previamente, foi também detetado código duplicado na Doddle OCR API. Efetuada a análise ao mesmo, identificou-se que o código dizia respeito ao objeto JSON utilizado para efetuar o mapeamento dos certificados através das coordenadas dos campos. Visto que vários campos no certificado possuíam informação semelhante, mas ids diferentes, foi necessário manter os mesmos de modo a identificar as suas coordenadas em cada uma das páginas.

Dessa forma, é possível concluir que a nível de qualidade técnica, a solução implementada cumpriu com todos os requisitos não funcionais definidos na secção 4.1.2, comprovando a qualidade do código desenvolvido nas aplicações.

6.3.2 QEF

A subsecção que se segue, apresenta a avaliação da qualidade do projeto dentro de uma vertente mais funcional. De forma a efetuar esta avaliação, foi utilizada a ferramenta *Quantitative Evaluation Framework* (QEF) para desenhar um modelo e avaliar a solução a partir das métricas definidas, como demonstrado nas figuras que se seguem.

q	**D**	Q_i	**Dimensão**	Q_j	W_{ij} (Peso do fator *j* na Dim *i*) [0,1]	**Fator**	rw_{jk} (Peso do requisito *k* no Fator *j*) {2, 4, 6, 8, 10}	**Requisito**	wf_k % realização do requisito *k*) [0,100]
95%	0.13	97.5	Funcionalidade	100	0.33	Funcional (Referente aos casos de uso)	10	FF01 - Importar Certificado	100
							8	FF02 - Validar certificado	100
							6	FF03 - Listar certificados	100
							8	FF04 - Duplicar Certificado	100
							8	FF05 - Emitir certificado	100
				94.6429	0.47	Interação do utilizador	8	FIUI01 - Aplicações são intuitivas	100
							6	FIUI02 - Aplicações apresentam a mesma experiência de navegação	100
							10	FIUI03 - Permissões e ações baseadas em tipo de utilizador são garantidas	100
							8	FIUI04 - Aplicações têm rapido acesso às principais funcionalidades	100
							8	FIUI05 - Aplicações suportam ações de navegação	100
							10	FIUI06 - Aplicações mobile possuem modo offline.	100
							6	FIUI07 - Aplicações suportam ações de pesquisa	50
				100	0.20	Qualidade do conteúdo	6	FCQ01 - Toda a informação de certificados encontra-se bem organizada.	100
							10	FCQ02 - Toda o texto encontra-se bem escrito e as frases fazem sentido.	100
							8	FCQ03 - Todas as mensagens são fáceis de perceber.	100
		93.87755102	Suportabilidade	100	0.43	Manutenibilidade	10	SM01 - Implementação das aplicação efetuada de forma sustentável, facilitando a sua manutenção.	100
							8	SM02 - Aplicações apresentam conjuntos de interfaces, adicionando a possibilidade de introduzir novas funcionalidades.	100
							8	SM03 - Análise efetuada pelo SonarQube deve ser classifica com letra A para a métrica de Code Smells nas aplicações.	100
				78.5714	0.29	Testabilidade	6	ST01 - Aplicações possuem boa cobertura de testes	50
							8	ST02 - Diferentes componentes do sistema podem ser testados de forma isolada	100
				100	0.14	Compatibilidade	10	SC01 - Todas as aplicações suportam multiplos dispositivos e browsers	100
				100	0.14	Escalabilidade	10	SE01 - Sistema desenhado e implementado com capacidade de escalar.	100
		88.88888889	Confiabilidade	77.7778	0.50	Disponibilidade	10	CD01 - Disponibilidade das aplicações deve ser garantida em caso de erros	100
							8	CD02- Falhas devem ser o mais isoladas possivel, não comprometendo o funcionamento do resto do sistema.	50
				100	0.50	Computação	10	CC01 - Toda a comunicação entre serviços deve ser efetuada através de HTTPS	100
							6	CC02 - Análise efetuada pelo SonarQube deve ser classifica com letra A para a métrica de Bugs nas aplicações.	100

Figura 55 - Modelo QEF desenhado para a solução desenvolvida

Dimensão	Funcionalidade
Fator	Funcional

		Wfk - Realização (%)		
Requisito	**Avaliação da métrica**	**0**	**50**	**100**
FF01 - Importar Certificado	Utilizador consegue importar certificados submentendo uma ou mais imagens.	Sem acesso à funcionalidade	Acesso parcial à funcionalidade (não consegue submeter mais que uma imagem de cada vez.)	Acesso completo à funcionalidade
FF02 - Validar certificado	Utilizador consegue validar certificado importado e editar informação preenchida automaticamente.	Sem acesso à funcionalidade	Acesso parcial à funcionalidade (não consegue editar o certificado após importar o mesmo)	Acesso completo à funcionalidade
FF03 - Listar certificados	Utilizador consegue ver uma listagem de certificados importados previamente ao entrar na aplicação.	Sem acesso à funcionalidade	Acesso parcial à funcionalidade (não mostra apenas os certificados importados pelo utilizador)	Acesso completo à funcionalidade
FF04 - Duplicar Certificado	Utilizador consegue duplicar parcialmente ou completamente certificados importados previamente.	Sem acesso à funcionalidade	Acesso parcial à funcionalidade (não consegue especificar tipo de duplicação a efetuar)	Acesso completo à funcionalidade
FF05 - Emitir certificado	Utilizador consegue emitir certificados duplicados previamente.	Sem acesso à funcionalidade	-	Acesso completo à funcionalidade

Figura 56 - Métricas e percentagens de realização para o fator Funcional

Dimensão	Funcionalidade
Fator	Interação do utilizador

		Wfk - Realização (%)		
Requisito	Avaliação da métrica	0	50	100
FIU01 - Aplicações são intuitivas	A experiência do utilizador deve ser intuitiva	Não	Parcialmente	Sim
FIUI02 - Aplicações apresentam a mesma experiência de navegação	Todos os ecrãs e páginas possuem butoes e menus que permitem navegar pela aplicação	Não	Parcialmente	Sim
FIUI03 - Permissões e ações baseadas em tipo de utilizador são garantidas	Funcionalidades estão disponiveis apenas a certo tipo de utilizadores (autenticação)	Não	-	Sim
FIU04 - Aplicações têm rapido acesso às principais funcionalidades	Quando o utilizador entra na aplicação consegue ver grande parte das funcionalidades disponiveis	Não	-	Sim
FIU05 - Aplicações suportam ações de navegação	Aplicação possui botões de e menus que permitem navegar pela mesma.	Não	-	Sim
FIUI06 - Aplicações mobile possuem modo offline.	Aplicação permite efetuar todas ou grande parte das funcionalidades em modo offline.	Não	-	Sim
FIUI07 - Aplicações suportam ações de pesquisa	Aplicação possui barras de pesquisa e formas de filtrar os certificados	Não	Filtros pré definidos através de butões	Sim

Figura 57 - Métricas e percentagens de realização para o fator Interação do utilizador

Dimensão	Funcionalidade
Fator	Qualidade do Conteúdo

		Wfk - Realização (%)		
Requisito	Avaliação da métrica	0	50	100
FCQ01 - Toda a informação de certificados encontra-se bem organizada.	Toda a informação relativa aos certificados deve estar bem organizada e ter um visual apelativo e intuitivo	Não	-	Sim
FCQ02 - Toda o texto encontra-se bem escrito e as frases fazem sentido.	As mensagens são pequenas e encontram-se escritas corretamente.	Não	-	Sim
FCQ03 - Todas as mensagens são fáceis de perceber.	Todas as mensagens returnadas ao utilizador e não contêm códigos nas mesmas.	Não	-	Sim

Figura 58 - Métricas e percentagens de realização para o fator Qualidade do conteúdo

Dimensão	Suportabilidade
Fator	Manutenibilidade, Testabilidade, Compatibilidade, Escalabilidade

		Wfk - Realização (%)		
Requisito	Avaliação da métrica	0	50	100
SM01 - Implementação das aplicação efetuada de forma sustentável, facilitando a sua manutenção.	Sistema segue padrões de design e boas práticas de desenvolvimento na implementação do mesmo.	Não	-	Sim
SM02 - Aplicações apresentam conjuntos de interfaces, adicionando a possibilidade de introduzir novas funcionalidades.	Implementação das funcionalidades é efetuada de forma genérica, de modo a facilitar a adição de novas funcionalides.	Não	-	Sim
SM03 - Análise efetuada pelo SonarQube deve ser classificada com letra A para a métrica de Code Smells nas aplicações.	Relatório gerado pelo SonarQube encontra-se dentro dos padrões de qualidade a nivel de Code Smells.	Letra C ou pior	Letra B	Letra A
ST01 - Aplicações possuem boa cobertura de testes	Todas as aplicações desenvolvidas apresentam um bom nivel de cobertura de testes quer unitarios como integração.	Má cobertura de testes ou inexistente.	Boa cobertura em alguns serviços e funcionalidades.	Todos os serviços e funcionalidades encontram-se testadas
ST02 - Diferentes componentes do sistema podem ser testados de forma isolada	Todos os componentes do sistema podem ser testados de forma isolada.	Não	-	Sim
SC01 - Todas as aplicações suportam multiplos dispositivos e browsers	Aplicações desenvolvidas suportam diversos dispositivos e browsers a nivel de UI, sendo responsivo e a nivel de funcionalidades apresentadas	Não	-	Sim
SE01 - Sistema desenhado e implementado com capacidade de escalar.	Arquitetura do sistema adotada e implementada permite escalar o mesmo facilmente.	Não	-	Sim

Figura 59 - Métricas e percentagens de realização para o fator Adaptabilidade

Dimensão	Confiabilidade			
Fator	Disponibilidade, Computação			
		Wfk - Realização (%)		
Requisito	Avaliação da métrica	0	50	100
CD01 - Disponibilidade das aplicações deve ser garantida em caso de erros	Aplicações possuem formas de lidar com erros e continuar operacionais após os mesmos.	Não	-	Sim
CD02- Falhas devem ser o mais isoladas possivel, não comprometendo o funcionamento do resto do sistema.	Em caso de falhas de qualquer um dos serviços o resto do sistema deve continuar operacional.	Todo o sistema fica em baixo.	Apenas serviços que apresentam dependências com a falha ficam em baixo	Todos os serviços do sistema continuam operacionais em qualquer falha
CC01 - Toda a comunicação entre serviços deve ser efetuada através de HTTPS	Comunicação efetuada entre todos os serviços do sistema é efetuada através de HTTPS	Não	-	Sim
CC02 - Análise efetuada pelo SonarQube deve ser classifica com letra A para a métrica de Bugs nas aplicações.	Relatório gerado pelo SonarQube encontra-se dentro dos padrões de qualidade a nivel de Bugs.	Letra C ou pior	Letra B	Letra A

Figura 60 - Métricas e percentagens de realização para o fator Confiabilidade

Os requisitos apresentados e definidos previamente cobrem os aspetos mais importantes relacionados com o desenvolvimento e resultado da solução implementada. O modelo apresentado na Figura 55, avalia não só a qualidade das funcionalidades apresentadas, mas também de todo o sistema e processo de desenvolvimento do mesmo.

A nível de resultados, foi obtida uma dimensão de 95% na avaliação da qualidade da solução, demonstrando que o sistema desenvolvido é considerado viável, dado que a percentagem se encontra perto dos 100%.

7 Conclusões

Neste capítulo são descritos os principais objetivos atingidos, assim como o possível trabalho futuro a realizar.

7.1 Objetivos atingidos

O principal objetivo do presente projeto passou não só pela criação de uma nova plataforma, como por dotar as plataformas existentes de funcionalidades que permitem aos profissionais efetuar as inspeções elétricas de forma mais produtiva e eficiente.

A Tabela 5, apresenta todos os objetivos definidos no início do presente projeto, assim como o estado dos mesmos face à solução desenvolvida.

Tabela 5 – Concretização dos objetivos definidos

Objetivo	**Estado**
Analisar abordagens existentes na aplicação de ferramentas e tecnologias de OCR.	Atingido
Desenho da arquitetura do sistema a desenvolver.	Atingido
Desenvolver uma API capaz de analisar e extrair informação de certificados elétricos.	Atingido
Desenvolvimento de um Portal Web que permita ao cliente final efetuar a importação e gestão de certificados.	Atingido
Documentar e testar solução desenvolvida	Atingido
Disponibilizar ambos os serviços ao cliente final num ambiente de produção.	Atingido

Numa primeira fase foram analisadas as abordagens existentes na aplicação de ferramentas e tecnologias de OCR, de modo a selecionar a mais adequada para a solução a desenvolver.

Efetuada a análise, concluiu-se que a Google Cloud Vision API seria a escolha ideal, dada a sua capacidade de análise e extração de informação de documentos, assim como os custos associados à sua utilização.

A etapa seguinte consistiu em redesenhar a arquitetura existente, de modo a suportar as novas funcionalidades de OCR a desenvolver. Foi adotada uma arquitetura de micro serviços, de modo a tornar a API responsável pelas funcionalidades de OCR facilmente escalável.

De seguida, foi desenvolvida uma nova API (Doddle OCR API), responsável por efetuar a análise e extração de informação dos certificados elétricos, assim como mapear a informação extraída em valores dos certificados existentes nas aplicações, de forma a preenchê-los automaticamente.

Após desenvolver toda a lógica de OCR, foi implementada a interface a disponibilizar aos utilizadores que permitisse aos mesmos importar novos certificados elétricos submetendo as imagens dos certificados de inspeções anteriores. Para isso, foi desenvolvido um portal web, onde os utilizadores podem importar e validar os dados automaticamente preenchidos dos certificados elétricos.

Com vista a testar e garantir a qualidade da solução desenvolvida, realizaram-se testes unitários e de integração. Os testes unitários, foram desenvolvidos tanto nas APIs como no portal web, de forma a validar os blocos de código mais importantes no sistema. Quanto aos testes de integração, estes foram desenvolvidos utilizando o Postman, de modo a automatizar todo o processo de validação das funcionalidades desenvolvidas. Por fim realizaram-se os testes não funcionais do sistema, tendo sido utilizadas ferramentas como o Artillery, para efetuar testes de carga à API desenvolvida. Posteriormente foi criada toda a documentação da API utilizando o Swagger.

Por fim, foi efetuada a implantação dos serviços implementados com o intuito de disponibilizar os mesmos aos engenheiros. Para isso, foi utilizado o serviço de Cloud DigitalOcean, onde foram criados três ambientes: desenvolvimento, teste e produção.

Além das metas definidas foram ainda adotadas algumas práticas de desenvolvimento que facilitassem o desenvolvimento do mesmo, como, controlo de versões e divisão de tarefas em *sprints*. Desta forma, todas as metas foram cumpridas, sendo atingido o principal objetivo do projeto a desenvolver.

7.2 Limitações e trabalho futuro

Apesar do desenvolvimento do projeto ter sido efetuado com sucesso, existem alguns aspetos que poderiam ser melhorados ou incrementados, de forma a aumentar a qualidade da solução desenvolvida.

Uma das principais limitações é o suporte de diferentes tipos de certificados. Dada a elevada complexidade da lógica de extração e mapeamento de informação dos certificados, a API desenvolvida apenas suporta o certificado elétrico EICR. Dessa forma, o suporte de novos tipos de certificados é uma das principais funcionalidades a desenvolver no futuro, dado que, o cliente possui diversos formatos utilizados nas suas inspeções.

Referências

(Adobe, 2021) Adobe. (2021). Como usar o OCR no Adobe Acrobat Export PDF, na Document Cloud e no Reader. https://helpx.adobe.com/pt/document-cloud/help/using-ocr-exportpdf.html

(Amazon Web Services, 2022a) Amazon Web Services. (2022a). AWS global infrastructure - Amazon Web Services. https://aws.amazon.com/pt/about-aws/global-infrastructure/

(Amazon Web Services, 2022b) Amazon Web Services. (2022b). Elastic Compute Cloud - Amazon EC2 - AWS. https://aws.amazon.com/pt/ec2/

(Amazon Web Services, 2022c) Amazon Web Services. (2022c). Intelligently Extract Text & Data with OCR - Amazon Textract - Amazon Web Services. https://aws.amazon.com/textract/

(Borza, 2011) Borza, J. (2011). FAST Diagrams: The Foundation for Creating Effective Function Models General Dynamics Land Systems.

(Brewster, 2008) Brewster, S. A. (2008). Using the Optacon device to read text. The image on the right shows a... | Download Scientific Diagram. https://www.researchgate.net/figure/Using-the-Optacon-device-to-read-text-The-image-on-the-right-shows-a-close-up_fig3_227981053

(Brownell, 2018) Brownell, J. (2018). Edit Text in Acrobat Pro DC - Instructions and Video Lesson. https://www.teachucomp.com/edit-text-in-acrobat-pro-dc-instructions/

(CallMiner, 2020) CallMiner. (2020). What is Customer Value? | CallMiner. https://callminer.com/blog/what-is-customer-value

(DigitalOcean, 2021) DigitalOcean. (2021). Regional Availability Matrix :: DigitalOcean Documentation. https://docs.digitalocean.com/products/platform/availability-matrix/

(DigitalOcean, 2022) DigitalOcean. (2022). DigitalOcean. https://www.digitalocean.com/pricing

(DocDigitizer, 2022) DocDigitizer. (2022). Home Page - Docdigitizer. https://www.docdigitizer.com/

(Eseme, 2022) Eseme, S. (2022). NestJS Typescript: The Ultimate Guide - Mastering Backend Development. https://masteringbackend.com/posts/nestjs-typescrpt-ultimate-guide

(Fryer, 2022) Fryer, V. (2022). The History of SaaS and the Revolution of Businesses. https://www.bigcommerce.com/blog/history-of-saas/#ecommerce-saas-companies

(Glozic, 2014) Glozic, D. (2014). NodeDay 2014 – Dejan Glozic. https://dejanglozic.com/2014/03/04/nodeday-2014/

(Google Cloud Platform, 2022a) Google Cloud Platform. (2022a). Vision AI | Cloud Vision API | Google Cloud. https://cloud.google.com/vision

(Google Cloud Platform, 2022b) Google Cloud Platform. (2022b). Global Locations - Regions & Zones | Google Cloud. https://cloud.google.com/about/locations/

(Google Cloud Platform, 2022c) Google Cloud Platform. (2022c). Kubernetes - Google Kubernetes Engine (GKE) | Google Cloud. https://cloud.google.com/kubernetes-engine

(Google Cloud Platform, 2022d) Google Cloud Platform. (2022d). Pricing Overview | Google Cloud. https://cloud.google.com/pricing/

(Hartman, 2022) Hartman, J. (2022). What is Java? Definition, Meaning & Features of Java Platforms. https://www.guru99.com/java-platform.html#3

(Hoos et al., 2015) Hoos, E., Gröger, C., & Mitschang, B. (2015). Mobile apps in engineering: A process-driven analysis of business potentials and technical challenges. Procedia CIRP, 33, 17–22. https://doi.org/10.1016/J.PROCIR.2015.06.005

(Indravadanbhai Patel et al., 2012) Indravadanbhai Patel, C., Patel, D., Patel Smt Chandaben Mohanbhai, C., Patel, A., Chandaben Mohanbhai, S., & Patel Smt Chandaben Mohanbhai, D. (2012). Optical Character Recognition by Open source OCR Tool Tesseract: A Case Study. Article in International Journal of Computer Applications, 55(10), 975–8887. https://doi.org/10.5120/8794-2784

(Jin, 2018) Jin, Z. (2018). Requirements Engineering Methodologies. Environment Modeling-Based Requirements Engineering for Software Intensive Systems, 13–27. https://doi.org/10.1016/B978-0-12-801954-2.00002-9

(Johnson & Shiff, 2021) Johnson, J., & Shiff, L. (2021). What Is Microservice Architecture? Microservices Explained – BMC Software | Blogs. https://www.bmc.com/blogs/microservices-architecture/

(Kaneriya, 2020) Kaneriya, T. (2020). Advantages & Disadvantages of Node.js : Why to Use Node.js? https://www.simform.com/blog/nodejs-advantages-disadvantages/

(Leung, 2019) Leung, Y. L. (2019). A Broad Overview of Application Architecture | by Yung L. Leung | FAUN Publication. https://faun.pub/a-broad-overview-of-application-architecture-5acf98829140

(Luchaninov, 2021) Luchaninov, Y. (2021, December 8). Why, When and How to Use Node.js for Backend Development in 2022. https://mobidev.biz/blog/node-js-for-backend-development

(Madeira, 2020) Madeira, T. (2020). Why Use Python for Web Development? https://www.imaginarycloud.com/blog/why-use-python-for-web-development/

(Martin, 2022) Martin, M. (2022). What is Non-Functional Requirement in Software Engineering? Types and Examples. https://www.guru99.com/non-functional-requirement-type-example.html

(Md Anwar Hossain et al., 2019) Md Anwar Hossain, B., Afrin, S., Anwar Hossain α, M., & Afrin σ, S. (2019). Optical Character Recognition based on Template Matching.

(Nicola, 2020) Nicola, S. (2020). ANÁLISE DE VALOR.

(Osterwalder et al., 2014) Osterwalder, A., Pigneur, Y., Bernarda, G., Smith, A., & Papadakos, T. (2014). Value Proposition Design.

(P. A. Koen et al., 2002) Koen, P. A., Ajamian, G. M., Boyce, S., Clamen, A., Fisher, E., Fountoulakis, S., Johnson, A., Puri, P., & Seibert, R. (2002). FuzzyFrontEnd: Effective Methods, Tools, and Techniques LITERATURE REVIEW AND RATIONALE FOR DEVELOPING THE NCD MODEL.

(P. Koen et al., 2001) Koen, P., Ajamian, G., Burkart, R., Clamen, A., Davidson, J., D'amore, R., Elkins, C., Herald, K., Incorvia, M., Johnson, A.,

	Karol, R., Seibert, R., Slavejkov, A., & Wagner, K. (2001). PROVIDING CLARITY AND A COMMON LANGUAGE TO THE "FUZZY FRONT END."
(Pereira, 2021)	Pereira, D. (2021). What is the Value Proposition Canvas? https://businessmodelanalyst.com/value-proposition-canvas/
(Richardson, 2022)	Richardson, C. (2022). Monolithic Architecture pattern. https://microservices.io/patterns/monolithic.html
(Rufenacht, 2020)	Rufenacht, M. (2020). The evolution of document capture from the first OCR to machine learning. https://parashift.io/en/the-evolution-of-document-capture/
(Scrum, 2022)	Scrum. (2022). What is Scrum? https://www.scrum.org/resources/what-is-scrum
(Shakirat Oluwatosin, 2014)	Shakirat Oluwatosin, H. (2014). Client-Server Model. 1, 67–71. www.iosrjournals.orgwww.iosrjournals.org
(Shirsath, 2020)	Shirsath, R. (2020). Summarized: Microservices Vs Monolithic Architecture | by Reemi Shirsath | Medium. https://reemishirsath.medium.com/microservices-vs-monolithic-architecture-578266c8ab6e
(SpinupWP, 2020)	SpinupWP. (2020). DigitalOcean vs AWS vs Google Cloud vs Linode vs Vultr: Which is Best for Hosting WordPress? - SpinupWP. https://spinupwp.com/digitalocean-vs-google-cloud-vs-aws/
(Srihari et al., 2003)	Srihari, S. N., Shekhawat, A., & Lam, S. W. (2003). Optical character recognition (OCR) | Encyclopedia of Computer Science. https://dl.acm.org/doi/10.5555/1074100.1074664
(Sufiyan, 2021)	Sufiyan, T. (2021). What is Node.js: A Comprehensive Guide. https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs
(Taylor, 2022)	Taylor, D. (2022). Digitalocean vs AWS: 10 Most Important Differences You Must Know! https://www.guru99.com/digitalocean-vs-aws.html
(Tesseract, 2022)	Tesseract. (2022). tesseract-ocr/tesseract: Tesseract Open Source OCR Engine (main repository). https://github.com/tesseract-ocr/tesseract
(The Apache Software Foundation, 2021)	The Apache Software Foundation. (2021). Apache PDFBox | A Java PDF Library. https://pdfbox.apache.org/
(Verma, 2017)	Verma, A. (2017). How To Edit PDF Files Easily Using Wondershare PDFelement? https://fossbytes.com/how-to-edit-pdf-files-pdfelement/