



Aplicação de Modelos de Atenção em Campanhas de Marketing Digital e Realidade Virtual

ISRC - Interdisciplinary Studies Research Center

2023 / 2024

1200690 Pedro Miguel Gaspar Alves Santos

ISEP INSTITUTO SUPERIOR
DE ENGENHARIA DO PORTO

Aplicação de Modelos de Atenção em Campanhas de Marketing Digital e Realidade Virtual

ISRC - Interdisciplinary Studies Research Center

2023 / 2024

1200690 Pedro Miguel Gaspar Alves Santos



Licenciatura em Engenharia Informática

Junho de 2024

Orientador ISEP: **Professora Doutora Ana Maria Madureira**

Supervisor: **Doutor Ivo Pereira**

*«Dedico este trabalho aos meus filhos pelo amor incondicional e à minha esposa
pelo amor e apoio constantes ao longo destes anos. »*

Agradecimentos

É com grande satisfação que chego ao final de mais uma Licenciatura em Engenharia, culminando esta com a conclusão do projeto final do curso, a etapa final da Licenciatura em Engenharia Informática.

Estou imensamente grato à minha esposa, Sara, por todo o apoio que me deu, não me deixando nunca desistir, mesmo quando a conclusão do curso teve impacto na vida pessoal. Igualmente estou muito grato aos filhos, Rita e Vasco, por todo o apoio que me deram, às vezes mesmo sem darem conta disso. A esta família mais próxima, sem o vosso apoio e sacrifício, nunca teria conseguido concluir este curso.

É importante também reconhecer a instituição onde o projeto foi desenvolvido, juntamente com os professores e alunos das turmas a que pertenci, uma vez que fomentaram e facilitaram a aprendizagem, mantendo sempre um elevado nível de partilha de aprendizagens e conhecimentos.

Um dos maiores agradecimentos vai para o meu principal grupo de trabalho que desempenhou um papel crucial sem o qual nada disto teria sido possível. Enfrentámos juntos diversos desafios, mas conseguimos ultrapassar todos os obstáculos. Assim, Leandro Pinto, António Sousa e Jorge Santos, aqui fica o meu mais sincero agradecimento.

Finalmente, tenho também de agradecer ao grupo de trabalho deste projeto: em primeiro lugar à minha orientadora, Professora Dra. Ana Madureira por me ter ajudado a concluir este projeto com sucesso e por me guiar no sentido de realizar um projeto que muito apreciei realizar, mas também que abre a porta futuramente a outros projetos na área de Inteligência Artificial; em segundo lugar ao Doutor Ivo Pereira, pelas ideias e críticas construtivas ao longo ao projeto, bem como por ter fornecido o conjunto de dados que irão ser usados para teste do sistema desenvolvido e, em terceiro lugar, à Inês Oliveira pela preciosa ajuda para colocar os Modelos de Atenção a funcionar corretamente e pela ajuda na escolha das melhores ferramentas a usar para a implementação deste projeto.

Resumo

O presente relatório descreve o projeto desenvolvido no âmbito da Unidade Curricular Projeto/Estágio (PESTI) da licenciatura em Engenharia Informática do Instituto Superior de Engenharia do Porto com o objetivo de desenvolver um sistema para a Aplicação de Modelos de Atenção em Campanhas de Marketing Digital e Realidade Virtual.

No contexto do Marketing Digital e da Realidade Virtual, é essencial capturar e manter a atenção do público-alvo para garantir o sucesso de campanhas e experiências publicitárias. No entanto, não existem ferramentas específicas onde se apliquem Modelos de Atenção o que dificulta a integração dessas técnicas nas estratégias de marketing e em experiências de realidade virtual.

Ao longo deste projeto serão demonstradas as várias etapas do desenvolvimento de um sistema que permite a aplicação simples e rápida de dois modelos de atenção (CASNet I e CASNet II) em imagens e vídeos, possibilitando a geração de mapas de saliência visual para orientar o design e a produção de conteúdo mais interessante para o utilizador. Esses mapas de saliência ficam disponíveis num sistema com interface web, visualmente simples e apelativo que produz resultados eficazes.

Palavras-chave (Tema): Aprendizagem Automática, Aprendizagem Profunda, Marketing Digital, Modelos de Atenção, Realidade Virtual

Palavras-chave (Tecnologias): Python, React, Django

Abstract

The following report describes the project developed in the Curricular Unit Project/Internship (PESTI) of Informatic Engineering degree, held in Instituto Superior de Engenharia do Porto with the goal of developing a system for Applying Attention Models in Digital Marketing and Virtual Reality campaigns.

In the context of Digital Marketing and Virtual Reality, it is essential to capture and maintain the target audience's attention to ensure the success of advertising campaigns and experiences. However, there are not specific tools where Attention Models are applied which makes it harder to integrate those techniques in marketing strategies and virtual reality experiences.

Along this project there will be demonstrated several steps of developing a system which allows the easy and fast application of two attention models (CASNet I and CASNet II) in images and videos, making possible the generation of visual saliency maps to guide the design and making the content production more interesting for the end user. These saliency maps are available in a web interface system, visually clean and appellant which produces effective results.

Key-words (Theme):

Machine Learning, Deep Learning, Digital Marketing, Attention Models, Virtual Reality

Key-words (Technologies):

Python, React, Django

Índice

1	<i>Introdução</i>	1
1.1	Enquadramento/Contexto	1
1.2	Descrição do Problema	2
1.2.1	Objetivos	2
1.2.2	Metodologia	3
1.2.3	Contributos	3
1.2.4	Planeamento do trabalho	4
1.3	Estrutura do relatório	4
2	<i>Estado da arte</i>	6
2.1	Marketing Digital	6
2.2	Machine Learning	7
2.2.1	Aprendizagem Supervisionada	8
2.2.2	Aprendizagem Não Supervisionada	9
2.2.3	Aprendizagem Por Reforço	9
2.3	Modelos de Atenção	10
2.4	Trabalhos relacionados	12
2.5	Tecnologias existentes	14
2.6	Sumário	16
3	<i>Análise e desenho da solução</i>	19
3.1	Domínio do problema	19
3.2	Requisitos funcionais e não funcionais	21
3.2.1	Requisitos Funcionais	21
3.2.2	Requisitos Não Funcionais	23
3.3	Desenho	26
3.3.1	Arquitetura do sistema	26
3.3.2	Solução proposta	29
3.3.3	Solução alternativa	29
3.4	Sumário	31
4	<i>Implementação e Avaliação da Solução</i>	33

4.1	Descrição da implementação.....	33
4.1.1.	Web Service	33
4.1.2.	Modelos de Atenção	36
4.1.3.	User Interface.....	40
4.1.4.	Base de dados	42
4.1.5.	Controlo de versões	44
4.1.6.	CI/CD	45
4.1.7.	Controlo de tarefas	46
4.2	Testes.....	47
4.3	Avaliação da solução.....	52
4.4	Sumário.....	58
5	Conclusões.....	59
5.1	Objetivos concretizados	59
5.2	Limitações e trabalho futuro	60
5.3	Apreciação final	61
	Referências.....	63

Índice de Figuras

Figura 1: Os 3 tipos de Aprendizagem Automática [12].....	8
Figura 2: Os 3 tipos de Aprendizagem Automática e Aprendizagem Profunda [15].....	10
Figura 3: Ranking do TIOBE [24]	15
Figura 4: Ranking do PYPL [25]	15
Figura 5: Modelo FURPS+ [41].....	19
Figura 6: Modelo de Domínio	20
Figura 7: Diagrama de Casos de Uso	21
Figura 8: Diagrama de Casos de Uso	22
Figura 9: Diagrama de componentes do sistema	27
Figura 10: Diagrama de implantação do sistema	28
Figura 11: Diagrama BPMN do sistema	29
Figura 12: Diagrama de componentes alternativo do sistema	30
Figura 13: Ficheiro url.py	34
Figura 14: Ficheiro views.py	35
Figura 15: Ficheiro models.py	35
Figura 16: Ficheiro settings.py.....	36
Figura 17: Função de processamento de imagens	37
Figura 18: Ficheiro split.py	38
Figura 19: Ficheiro create_video.py	39
Figura 20: Persistência na base de dados de imagens	39
Figura 21: Intercetador de pedidos web para inclusão de token de acesso	40
Figura 22: Página de login do sistema desenvolvido.....	41
Figura 23: Home page do sistema desenvolvido após login	42
Figura 24: Classe User Serializer	43
Figura 25: Classe Note Serializer	44

Figura 26: Commits realizados no repositório GitHub	45
Figura 27: Funcionalidades do software Choreo	46
Figura 28: Trello – controlo de tarefas	47
Figura 29: Testes unitários de domínio	48
Figura 30: Teste unitário de criação de utilizador	48
Figura 31: Teste unitário de criação de nota.....	49
Figura 32: Teste de validação de criação do mapa de saliências	50
Figura 33: Resultado de Testes e2e.....	50
Figura 34: Testes e2e relativo à criação de um mapa de saliências de uma imagem sem criação de nota associada	51
Figura 35: Imagem publicitária e respetivos mapas de saliência obtidos usando o Modelo de Atenção CASNet I e CASNet II.....	52
Figura 36: Imagem publicitária e respetivos mapas de saliência obtidos usando o Modelo de Atenção CASNet I e CASNet II.....	52
Figura 37: Imagem publicitária e respetivos mapas de saliência obtidos usando o Modelo de Atenção CASNet I e CASNet II.....	53
Figura 38: Imagem publicitária e respetivos mapas de saliência obtidos usando o Modelo de Atenção CASNet I e CASNet II.....	53
Figura 39: Imagem publicitária e respetivos mapas de saliência obtidos usando o Modelo de Atenção CASNet I e CASNet II.....	54
Figura 40: Imagem publicitária e respetivos mapas de saliência obtidos usando o Modelo de Atenção CASNet I e CASNet II.....	55
Figura 41: Imagem publicitária e respetivos mapas de saliência obtidos usando o Modelo de Atenção CASNet I e CASNet II.....	55
Figura 42: Imagem publicitária e respetivos mapas de saliência obtidos usando o Modelo de Atenção CASNet I e CASNet II.....	55
Figura 43: Vídeo do mundo de realidade virtual e respetivos mapas de saliência obtidos usando o Modelo de Atenção CASNet I e CASNet II	56

Índice de Tabelas

Tabela 1: Planeamento do projeto	4
Tabela 2: Requisitos de Usabilidade	23
Tabela 3: Requisitos de Confiabilidade	23
Tabela 4: Requisitos de Desempenho	24
Tabela 5: Requisitos de Suportabilidade	24
Tabela 6: Restrições de Desenho	25
Tabela 7: Restrições de Implementação	25
Tabela 8: Restrições de Interface	26
Tabela 9: Restrições de segurança	26
Tabela 10: Base de dados - Classe User Serializer	43
Tabela 11: Base de dados - Classe Note Serializer	44
Tabela 12 Comparação de tempo de processamento dos métodos CASNet I e CASNet II	57

Notação e Glossário

ASPP	Agrupamento de pirâmide espacial atrosa (do inglês <i>Atrous Spatial Pyramid Pooling</i>)
CNN	Redes Neurais Convolucionais (do inglês <i>Convolutional Neural Networks</i>)
CPU	Processador
CRM	Gestão da relação com o cliente (do inglês <i>Customer Relationship Management</i>)
CSS	Folhas de estilo em cascata (do inglês <i>Cascading Style Sheets</i>)
EMOd	Conjunto de dados de Atenção Emocional
IA	Inteligência Artificial
ISEP	Instituto Superior de Engenharia do Porto
ISRC	Centro de Desenvolvimento de Estudos Interdisciplinares (do inglês <i>Interdisciplinary Studies Research Center</i>)
NLP	Processamento de Linguagem Natural (do inglês <i>Natural Language Processing</i>)
PESTI	Unidade Curricular de Projeto/Estágio da Licenciatura de Engenharia Informática
RNN	Redes Neurais Recorrentes (do inglês <i>Recurrent Neural Networks</i>)
SO	Sistema operativo
WS	Serviço Web (do inglês <i>Web service</i>)

1 Introdução

O projeto final ou estágio curricular é um componente fundamental no processo da formação académica. De fato, no final de uma licenciatura, é crucial que seja dado ao estudante um problema que permita aplicar os conhecimentos adquiridos durante o percurso académico.

Este relatório descreve o projeto desenvolvido no âmbito da Unidade Curricular Projeto/Estágio (PESTI) com o objetivo de desenvolver um sistema para a Aplicação de Modelos de Atenção em Campanhas de Marketing Digital e Realidade Virtual. Este projeto foi desenvolvido no *Interdisciplinary Studies Research Center* (ISRC) no âmbito do projeto da Fundação para a Ciência e Tecnologia denominado *Enhancing Phygital Marketing through Multimodal Artificial Intelligence (PHYNHANCAI)* e será detalhado e contextualizado ao longo deste documento, incluindo os objetivos específicos, a abordagem, o planeamento e o desenvolvimento do projeto.

Este capítulo da introdução oferece assim uma visão geral do trabalho realizado no âmbito deste projeto.

1.1 Enquadramento/Contexto

Este relatório foi realizado no âmbito da Unidade Curricular Projeto/Estágio (PESTI) do 3º ano do curso de Licenciatura em Engenharia Informática do Instituto Superior de Engenharia do Porto (ISEP).

Devido à frequência em regime pós-laboral da licenciatura e, também, devido ao interesse em áreas de investigação e desenvolvimento, particularmente nas áreas de inteligência artificial, foi decidido realizar um projeto no *Interdisciplinary Studies Research Center* (ISRC).

O ISRC foca-se na integração de investigação científica com a área da educação em engenharia, preenchendo um vazio na investigação e desenvolvimento atual, tirando partido das sinergias interdisciplinares existentes na Instituição. Todos os avanços tecnológicos mundiais das últimas décadas implicam que todas as pessoas e, consequentemente, os seus países, estejam interligados numa sociedade cada vez mais global e multicultural. Este

fenómeno da multiculturalidade faz parte do dia-a-dia do estudante, uma vez que atualmente desempenha as suas atuais funções numa empresa americana multinacional.

1.2 Descrição do Problema

Este trabalho insere-se no âmbito do projeto de investigação PHYNNHANCAI (*Enhancing Phygital Marketing through Multimodal Artificial Intelligence*) e pretende o desenvolvimento de modelos de Inteligência Artificial Multimodal e a exploração de abordagens de Realidade Virtual e Realidade Aumentada para melhorar as estratégias de Marketing Omnicanal [1].

No contexto do marketing digital e da realidade virtual, é essencial capturar e manter a atenção do público-alvo para garantir o sucesso de campanhas e experiências publicitárias. No entanto, a análise manual de imagens e vídeos para identificar áreas de interesse pode ser morosa e suscetível a erros.

Para além destes problemas, não existem ferramentas específicas onde se apliquem modelos de atenção o que dificulta ainda mais a integração dessas técnicas nas estratégias de marketing e em experiências de realidade virtual. Assim, surge a necessidade de um sistema que permita a aplicação simples e rápida de modelos de atenção em imagens e vídeos publicitários, possibilitando a geração de mapas de saliência¹ visual para orientar o design e a produção de conteúdo publicitário mais interessante para o utilizador.

1.2.1 Objetivos

Este projeto tem como objetivo principal o desenvolvimento de um sistema para Aplicação de Modelos de Atenção que seja robusto e intuitivo em imagens de campanhas de email marketing e vídeos de realidade virtual. O sistema deverá ter uma utilização fácil e eficiente, proporcionando resultados precisos e informativos.

Subjacente ao desenvolvimento do sistema está a validação experimental da sua eficiência. Assim, irá realizar-se uma série de testes experimentais para avaliar a eficácia e precisão do sistema desenvolvido, que incluirá comparações entre os mapas de saliência gerados pelo sistema e avaliações humanas, garantindo a qualidade e relevância dos resultados para as estratégias de marketing e experiências de realidade virtual.

¹ Mapa de Saliência é uma imagem em que a luminosidade de um pixel representa quão saliente o pixel é, isto é, a luminosidade de um pixel é diretamente proporcional à sua saliência [59]

Um último objetivo é a implementação de funcionalidades de pré-processamento de dados para garantir a qualidade e consistência dos inputs fornecidos ao sistema. Isso incluirá técnicas de normalização, redimensionamento e tratamento de ruídos preparando os dados para uma análise mais precisa e eficaz pelos modelos de atenção.

1.2.2 Metodologia

Para o desenvolvimento de qualquer solução, a primeira ação a realizar é a leitura e compreensão de artigos publicados que tenham relação com o problema a resolver. Devido à exigência do cliente do uso da ferramenta Python, paralelamente a esta ação de recolha de informação publicada, foi aprendida esta tecnologia amplamente usada no mercado [2] [3].

Posteriormente, realizou-se a implementação do sistema propriamente dito, recorrendo a um processo ágil, iterativo e incremental. Em todas as reuniões com o cliente foi apresentada uma versão cada vez mais completa do *software*, de modo a obter *feedback* o mais cedo possível nas várias fases do desenvolvimento.

Para acompanhar todo o processo de desenvolvimento, após ser definido e acordado com o cliente e com a orientadora o planeamento geral do trabalho, foi usada a ferramenta *Trello* [4].

Na implementação do sistema, a metodologia seguida foi também incremental, isto é, começou-se por implementar a estrutura genérica do sistema, criando o *backend*, base de dados e *frontend*. Após ter essa estrutura implementada, adicionou-se componentes cada vez mais complexos a essa estrutura, como por exemplo o código dos Modelos de Atenção. Com esta metodologia foi possível fazer evoluir o sistema desenvolvido dando solução a cada um dos objetivos propostos.

1.2.3 Contributos

O desenvolvimento deste projeto e consequente documentação permite, tal como referido nos objetivos e através do uso de uma Aplicação de Modelos de Atenção, obter um *feedback* praticamente instantâneo acerca da qualidade de imagens ou vídeos de publicidade. Tendo em conta os diferentes objetivos do utilizador, existem também diversas opções implementadas que vão permitir dar solução a esses objetivos.

Deste modo, uma empresa criadora de conteúdos publicitários poderá facilmente, com o recurso a este sistema, determinar o sucesso ou não desses mesmos conteúdos antes mesmo de os lançar para o mercado. Isto fará com que exista um benefício claro da qualidade

e, conseqüentemente, do retorno financeiro dos conteúdos publicitários gerados pelos utilizadores do sistema desenvolvido.

1.2.4 Planeamento do trabalho

O planeamento do trabalho foi realizado tendo em conta as várias tarefas a desempenhar, priorizando inicialmente um conhecimento mais aprofundado sobre o tema e sobre as tecnologias a usar, passando posteriormente para a fase de desenvolvimento do sistema até à obtenção da solução final.

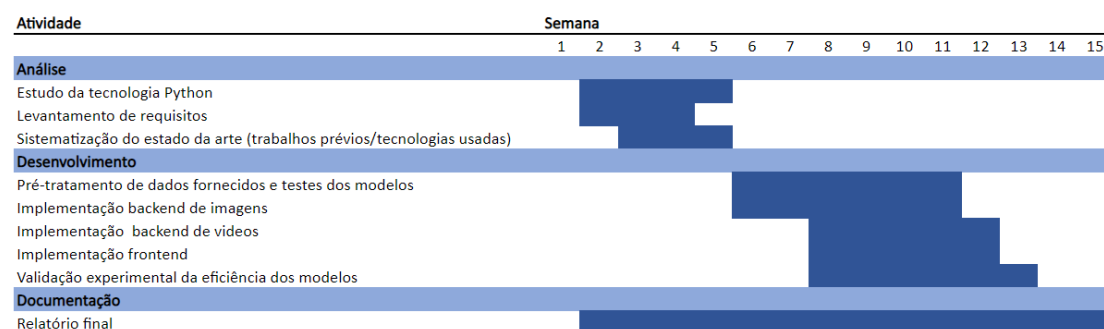


Tabela 1: Planeamento do projeto

Com este planeamento inicial foi possível gerir a ocupação de todo o tempo previsto para a frequência da unidade curricular na qual este projeto se insere, permitindo a conclusão do projeto dentro do tempo previsto para a conclusão da unidade curricular.

1.3 Estrutura do relatório

O presente relatório está dividido em cinco capítulos, os quais se descrevem seguidamente.

No primeiro capítulo intitulado Introdução realiza-se uma contextualização do problema e explicitam-se os objetivos definidos. Posteriormente, apresenta-se a abordagem seguida e também o planeamento do projeto, evidenciando as várias fases de desenvolvimento.

No segundo capítulo, Estado da Arte, é descrito o estado atual de conhecimento do problema, aborda trabalhos existentes e fornece uma visão geral das soluções existentes.

O terceiro capítulo, Análise e Desenho da solução, foca-se na análise do problema, explicando as diversas componentes e como o fluxo de informação é realizado. É também

esquemático o desenho da solução, dando resposta aos requisitos funcionais e não-funcionais.

No capítulo 4 é explicitada a Implementação da solução, seguindo o desenho da solução previsto e onde são avaliadas alternativas, vantagens e desvantagens recorrendo às métricas de validação.

Finalmente, no capítulo 5 são referidas as Conclusões, é combinando o conhecimento teórico com os resultados práticos, tendo em conta os objetivos propostos para o projeto. São também identificadas as limitações e futuras melhorias que podem ser realizadas.

2 Estado da arte

Neste capítulo, vai-se proceder à contextualização do tema e suas implicações na realidade do dia-a-dia, bem como proceder-se à análise de outros trabalhos relacionados. Por outro lado, ir-se-á detalhar as tecnologias usadas no desenvolvimento deste projeto. Este estudo teórico permitirá ajudar e corroborar as decisões efetuadas na formulação da solução final desenvolvida, detalhadas no capítulo seguinte.

2.1 Marketing Digital

O Marketing Digital² é hoje em dia uma ferramenta amplamente utilizada pelas empresas com a missão da continuidade do negócio empresarial. Através de um simples clique, as empresas são colocadas muito próximas do público-alvo, permitindo assim que as campanhas publicitárias sejam muito mais eficientes. Por outro lado, o Marketing Digital permite também um conhecimento mais aprofundado por parte das empresas do perfil de cada consumidor, pelo que as campanhas podem ser personalizadas e conseguir com isso maiores benefícios.

O conceito de Marketing de Atração, ou *Inbound Marketing*, foi inicialmente proposto por Brian Halligan em 2005, tendo sido usado como missão da sua empresa HubSpot, co-fundada por Dharmesh Shah [5].

Este conceito baseia-se fundamentalmente em realizar ações em 4 níveis distintos por forma a conseguir aumentar vendas influenciando as decisões de compra dos utilizadores [6]:

- Atrair – usando diferentes técnicas de marketing, sejam marketing de conteúdo, difusão nas redes sociais, etc. com o objetivo único de aumento de tráfego na página web da empresa
- Converter – usando técnicas como conteúdos *premium*, *webinars* e outros, permite conseguir os dados dos utilizadores de modo a ser possível uma posterior maior capacidade de personalização
- Vender – através de uma integração de uma equipa de vendas e uso de ferramentas de automatização de marketing ou de gestão de relação de clientes (CRM)

² Marketing Digital é o conjunto de atividades que uma empresa (ou pessoa) executa online com o objetivo de atrair novos negócios, criar relacionamentos e desenvolver uma identidade de marca [60].

- Agradar – conseguir que os compradores repitam a compra e aumentar assim o vínculo com a empresa.

Assim, num mundo cada vez mais digital e mais informado, os utilizadores pretendem realizar compras de forma mais consciente e, por isso, as empresas, através do Marketing de Atração³, adaptaram-se a essa nova realidade, aproveitando o maior conhecimento de cada cliente fornecendo campanhas de Marketing mais específicas e personalizadas.

2.2 Machine Learning

A Aprendizagem Automática (*Machine Learning*), é uma das áreas da Inteligência Artificial (IA), definida como um conjunto de “métodos computacionais que usam a experiência a fim de sofisticar a performance ou efetuar previsões confiáveis” [7].

Este campo de estudo tem o objetivo de imitar a capacidade da aprendizagem humana, permitindo que as máquinas melhorem ao longo do tempo, tornando-se cada vez mais precisas nas previsões ou classificações. Existem variadas aplicações usadas no nosso dia-a-dia e algumas que a maioria das pessoas nem reconhece que usa a Aprendizagem Automática. Como exemplo, pode-se explicitar algumas funcionalidades das redes sociais, permitindo uma personalização da experiência do utilizador; os assistentes virtuais, como a Siri da Apple [8], ou a Alexa da Amazon [9] recolhem dados de cada utilização e melhoram a resposta nas utilizações seguintes ou ainda motores de busca com recomendações que são usados em diversos sites de comércio online, uma vez que recolhem dados das pesquisas e compras efetuadas e depois recomendam produtos com base nessa recolha [10].

Existem 3 tipos de Aprendizagem Automática, a Aprendizagem Supervisionada (*Supervised Learning*), a Aprendizagem Não Supervisionada (*Unsupervised Learning*) e a Aprendizagem por Reforço (*Reinforcement Learning*) [11] [12].

³ Marketing de Atração é o conjunto de estratégias de Marketing que visam atrair e converter clientes usando conteúdo relevante [61].

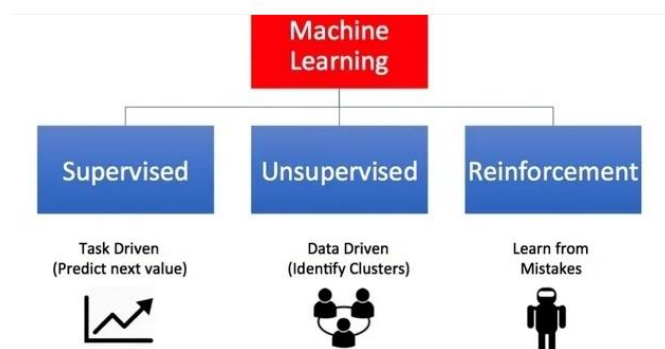


Figura 1: Os 3 tipos de Aprendizagem Automática [12]

2.2.1 Aprendizagem Supervisionada

A Aprendizagem Supervisionada é caracterizada pelo uso de dados rotulados para treinar algoritmos que classificam dados ou preveem resultados. De acordo com a maneira como os dados de entrada são inseridos no modelo, ele ajusta o peso de cada parâmetro até que o modelo seja ajustado corretamente, o que ocorre como parte do processo de validação cruzada. A Aprendizagem Supervisionada é a mais utilizada e ajuda organizações em todo o mundo a resolver uma variedade de problemas reais em grande escala e pode ser usada para criar modelos com elevada precisão [13].

Este tipo de Aprendizagem usa um *dataset* de treino para treinar o modelo, onde são providenciados dados de entrada e também as respostas previstas para esses dados de entrada. O algoritmo mede a sua eficácia e vai ajustando o erro até que este tenha sido suficientemente minimizado [13]. Após o treino do modelo, este pode ser usado para outros dados de entrada para obter a previsão da solução respetiva.

A Aprendizagem Supervisionada pode ser dividida em 2 tipos de problemas, problemas de classificação e problemas de regressão. Nos problemas de classificação é necessário assignar dados de teste a categorias específicas, sendo os modelos mais comuns para este processo as árvores de decisão, K-vizinhos mais próximos ou a *Random Forest*. Por outro lado, os problemas de regressão são usados para entender a relação entre variáveis dependentes e independentes, sendo os modelos de regressão mais comuns a regressão linear ou a regressão polinomial [13].

2.2.2 Aprendizagem Não Supervisionada

A Aprendizagem Não Supervisionada usa algoritmos para analisar e agrupar conjuntos de dados não rotulados. Estes algoritmos permitem descobrir padrões nos dados sem necessidade da interação com o ser humano, daí serem chamados de não supervisionados [14].

Estes modelos de Aprendizagem Não Supervisionada são usados para 3 principais funções: agrupar, associar ou reduzir a dimensão dos dados [14]:

- Agrupar é uma técnica para juntar dados não etiquetados baseando-se nas suas semelhanças ou diferenças. Por exemplo, K-vizinhos mais próximos agrupa pontos de dados semelhantes em grupos, em que K representa o tamanho do grupo ou granularidade.
- Associação é outra técnica que usa diferentes regras para encontrar relações entre variáveis de um grupo de dados. Estes métodos são muito usados para motores de busca de recomendações.
- Redução da dimensão é uma técnica usada quando o número de características (ou dimensões) de um grupo de dados é muito grande. Nestes casos, usa-se esta técnica reduzindo o número de dados de entrada para um valor aceitável preservando a integridade dos dados.

2.2.3 Aprendizagem Por Reforço

A Aprendizagem Por Reforço é uma técnica de Aprendizagem Automática que permite a um agente que aprenda com um ambiente interativo por tentativa e erro recebendo feedback das suas próprias ações e experiências [12].

Embora a Aprendizagem Supervisionada e a Aprendizagem por Reforço usem um mapeamento entre dados de entrada e resultados obtidos, elas não são iguais. Por um lado, na Aprendizagem Supervisionada, após o treino do modelo, é fornecido um dado de entrada para obter um resultado, no caso da Aprendizagem Por Reforço são fornecidos constantemente recompensas ou punições como sinais de comportamento positivo ou negativo [12]. Ao comparar a Aprendizagem Não Supervisionada e a Aprendizagem Por Reforço verifica-se que a grande diferença entre elas tem a ver com o objetivo. Enquanto que a Aprendizagem Não Supervisionada pretende descobrir semelhanças ou diferenças entre

pontos de dados, na Aprendizagem Por Reforço o objetivo é encontrar ações que maximizem o valor de recompensa [12].

2.3 Modelos de Atenção

A Aprendizagem Automática tem uma categorização transversal aos 3 tipos de Aprendizagem que é chamada de Aprendizagem Profunda (*Deep Learning*). De facto, a Aprendizagem Profunda é uma parte da Aprendizagem Automática que usa métodos baseados em redes neurais. As arquiteturas da Aprendizagem Profunda podem ser redes neurais profundas, redes neurais recorrentes, redes neurais convolucionais, entre outras [15].

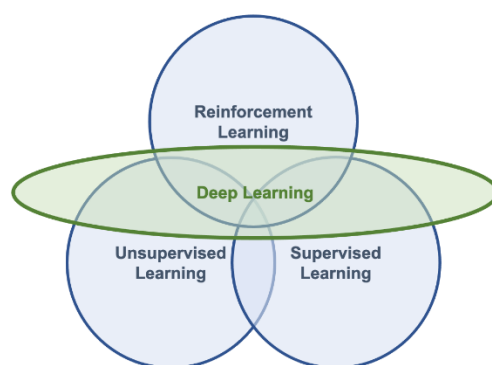


Figura 2: Os 3 tipos de Aprendizagem Automática e Aprendizagem Profunda [15]

Os Modelos de Atenção surgiram como uma técnica poderosa da Aprendizagem Profunda, particularmente nos campos do Processamento da Linguagem Natural (*Natural Language Processing* – NLP) e Visão por Computador. Estes modelos foram desenhados para melhorar o desempenho das redes neurais, usando a experiência adquirida da visão humana, permitindo um maior foco em partes específicas dos dados de entrada de modo a conseguirem obter um melhor resultado [16].

Um Modelo de Atenção é um componente de uma rede neuronal que atribui um nível de importância, ou atenção, a diferentes partes dos dados de entrada. Os Modelos de Atenção podem ser integrados em vários tipos de redes neurais, desde Redes Neurais Recorrentes (*Recurrent Neural Networks* RNN), Redes Neurais Convolucionais (*Convolutional Neural Networks* CNN) e também Modelos Transformacionais [16].

Existem diferentes tipos de Modelos de Atenção que criam diferentes mapeamentos entre entradas e saídas, incluindo diferentes fontes, codificadores, decodificadores e pesos [17]:

- **Modelo de Atenção Global (ou *Soft*)**

O Modelo de Atenção Global agrupa entradas de todos os estados do codificador e do decodificador antes de avaliar o estado atual para determinar a saída. Este modelo utiliza cada etapa do codificador e cada etapa preliminar do decodificador para calcular os pesos de atenção ou pesos de alinhamento. Também multiplica cada etapa do codificador pelos pesos de alinhamento globais para determinar o valor de contexto a ser alimentado para as redes neurais recorrentes (RNN). Isso permite que o modelo encontre a saída do decodificador.

- **Modelo de Atenção Local (ou *Hard*)**

O Modelo de Atenção Local é semelhante ao modelo de atenção global, mas usa apenas algumas posições do codificador para determinar os pesos de alinhamento. O modelo calcula os pesos de alinhamento e o vetor de contexto usando a primeira posição de alinhamento única e uma seleção de palavras da fonte do codificador. O Modelo de Atenção Local é semelhante ao Modelo de Atenção Rígida, mas o Modelo de Atenção Rígida não é diferenciável na maioria dos pontos.

- **Modelo de Própria Atenção**

O Modelo de Própria Atenção foca-se em diferentes posições da mesma sequência de entrada. É possível utilizar os *frameworks* dos Modelos de Atenção Global e Local para criar este modelo. No entanto, este modelo inclui a mesma sequência de entrada em vez da sequência de saída alvo. É também um componente chave nos Modelos Transformacionais.

A operação fundamental dum Modelo de Atenção envolve 3 componentes principais: consultas (*queries*), chaves (*keys*) e valores (*values*). Esses componentes são derivados dos dados de entrada e são usados para calcular resultados de atenção, que determinam quanto foco o modelo deve dar a cada parte dos dados [16].

Num mecanismo de atenção típico, cada elemento da sequência de entrada é associado a uma chave e a um valor. Uma consulta, frequentemente relacionada ao estado atual do modelo, é comparada com todas as chaves usando uma função de compatibilidade, que pode ser um produto escalar simples ou uma rede neuronal mais complexa. O resultado dessa comparação é um conjunto de resultados de atenção, que são então normalizados, para criar uma distribuição de probabilidade conhecida como pesos de atenção. Esses pesos de atenção são usados para criar uma soma ponderada dos valores, que representa a informação agregada na qual o modelo deve focar. Esta soma ponderada é então tipicamente passada por camadas adicionais da rede neuronal para produzir a saída final [16].

2.4 Trabalhos relacionados

O processamento de vídeo implica a necessidade de elevados níveis de processamento computacional, o que é um dos maiores problemas atualmente tendo em conta o aumento constante do peso e importância das redes sociais. Existem várias alternativas para tentar ultrapassar este problema, entre os quais melhorar o hardware (utilizando componentes com maior eficiência, mais rápidos, com menor latência de rede), otimização de algoritmos e a remoção de informação desnecessária. Gharahbagh et al. [18] propõem no seu artigo uma solução para este problema, identificando e analisando apenas as partes críticas de cada vídeo o que melhora a performance do sistema de uma maneira geral.

O algoritmo proposto por Gharahbagh et al. [18] usa um algoritmo de registo de imagens (*frames* do vídeo), o método inicialmente começa por remover o movimento da câmara e, posteriormente, cada *frame* é usada para criar um mapa de saliências. Ao combinar o mapa de saliências com a informação de movimento derivado do fluxo ótico e da segmentação de cores é possível criar um que contém informação do movimento e também informação espacial. Finalmente, é sugerida uma função não-linear para combinar os mapas de saliência temporais e espaciais, que foi otimizada usando um algoritmo genético de vários objetivos.

A principal diferença deste método com o método usado neste projeto tem a ver com a não inclusão de mapa de saliências de movimento, apenas é analisado cada *frame* do vídeo. Por outro lado, cada *frame* é analisado como se fosse uma imagem estática onde não é feita nenhuma segregação de partes críticas da imagem, não se perdendo nenhum dado. Este

método é mais custoso a nível de processamento computacional, mas entendeu-se que, tendo em conta a aplicabilidade deste projeto, o processamento computacional não terá necessariamente de fornecer resultados instantâneos.

Em relação aos Modelos de Atenção, Fan et al. [19][20] avaliaram como alguns detalhes criados com o objetivo de evidenciar emoções nas imagens se relacionam com a atenção do ser humano. Devido aos limites de capacidade do cérebro humano, nem toda a estimulação a que o ser humano está sujeito, pode ser processada em paralelo e avaliada. De facto, o fenómeno conhecido como atenção seletiva corresponde ao estado em que todos os estímulos visuais competem para se tornar o foco dos olhos e são codificados na memória visual de curto prazo. Existe muita pesquisa que indica que a relevância emocional de um estímulo influencia a atenção seletiva. Por exemplo, as pessoas tendem a prestar atenção preferencialmente a estímulos que evocam emoções, como bebés fofos ou cenas eróticas.

Fan et al. [20] criaram um conjunto de dados de Atenção Emocional (EMOd). Esse EMOd é um conjunto diverso de imagens que contém a real percepção ocular de 16 indivíduos e detalhes dos objetos na imagem, desde contorno, sentimento e categoria de objeto. Realizaram um estudo intensivo no EMOd para identificar como é que o sentimento da imagem se relaciona com a atenção humana. Descobriram que a atenção humana está mais focada em cenas de veículos e animais que provocam admiração no EMOd. Com o objetivo de modelar computacionalmente o comportamento de atenção humana, projetaram uma rede neuronal profunda (CASNet I) para previsão de saliências, que aprende através do contexto espacial e semântico do contexto da imagem. Este modelo foi comparado com outros 8 modelos de atenção com resultados iguais ou melhores em todas as métricas avaliadas.

No seguimento do desenvolvimento da rede neuronal profunda CASNet I⁴, Fan et al. [21] criaram o CASNet II⁵ [19], usando o mesmo EMOd que no modelo de atenção explicado anteriormente, mas estenderam a sua pesquisa. Por um lado, a análise deixa de ser apenas ao nível do objeto, mas também ao nível da cena em que o objeto se inclui. Por outro lado, o CASNet II inclui uma sub-rede de ponderação de canais que prioriza objetos que provocam emoção e uma estrutura de *Atrous Spatial Pyramid Pooling* (ASPP) customizada que aprende a importância relativa de regiões de imagem em múltiplas escalas. Os campos recetivos

⁴ CASNet I é uma rede neuronal profunda que codifica a importância relativa de múltiplas regiões salientes e considera a importância contextual para a atenção humana [21]

⁵ CASNet II é uma rede neuronal profunda que codifica a importância relativa de múltiplas regiões salientes usando fatores de imagem e cognitivos que influenciam a percepção de sentimento visual [20]

ampliados do ASPP permitem que a CASNet II aprenda a importância relativa em regiões da imagem maiores, estendendo a priorização baseada em objetos anteriores para uma área de imagem mais ampla. Por fim, substituíram a arquitetura de duplo fluxo no CASNet I por um único fluxo e reduziram a escala da imagem de entrada. Com essas mudanças, o CASNet II, modela melhor o efeito de priorização emocional humana e supera significativamente o CASNet I na previsão de saliência em todos os cinco conjuntos de dados de referência, melhorando também o tempo de processamento em quase 300% [20].

Tanto no CASNet I como no CASNet II foram usadas métricas idênticas para avaliação dos resultados. Por um lado, a AUC (*Area Under ROC Curve*) que trata o mapa de saliências como um classificador binário. Foram usadas 3 variantes da AUC: AUC-Judd, AUC-Borji [22] e a mistura de AUC, sAUC (*shuffled AUC*) [23] que minimiza os efeitos de *bias*. Embora estes métodos sejam muito utilizados na comunidade científica, a AUC por natureza não consegue distinguir casos em que a previsão dos modelos inclui valores de importância relativa diferente para diferentes regiões duma imagem, o que era necessário no estudo. Foram usadas outras 5 métricas para medição a semelhança entre o mapa de saliências e o mapa de fixação, nomeadamente o *Normalized Scanpath Saliency* (NSS), *Linear Correlation Coefficient* (CC), *histogram intersection* (SIM), *Kullback-Leibler divergence* (KL) e *Information Gain* (IG) [22].

Neste projeto, foram usadas as redes neuronais profundas referidas (CASNet I e CASNet II) para análise do conjunto de dados fornecidos. Em vez de se procurar estabelecer o melhor método de análise de imagens e apreender a atenção do ser humano ao visualizar essas mesmas imagens, usaram-se ambas as redes CASNet para avaliar imagens de marketing digital e comparar com dados empíricos obtidos pela análise de um conjunto de seres humanos.

2.5 Tecnologias existentes

As técnicas realizadas com Aprendizagem Profunda estão ainda bastante dependentes das tecnologias implementadas, uma vez que exigem bastante recursos computacionais. Neste projeto utilizaram-se 2 modelos de redes neuronais profundas diferentes, em que já se conhecia à priori a sua grande diferença de desempenho. Com vista a suportar o desenvolvimento do sistema, foram usadas outras tecnologias que se entenderam ser mais eficientes para a concretização do projeto.

Python

É considerada pela TIOBE [24], uma organização especializada em métricas de qualidade de software e pela PYPL [25], um índice criado pela Google destinado a analisar a popularidade das linguagens de programação, a linguagem de programação mais usada há vários anos. De facto, a linguagem que adquiriu o nome devido a um grupo de comediantes chamado Monty Python [26], tem-se mantido no top 3 destes rankings desde maio de 2019 e destacadamente número 1 desde junho de 2023.

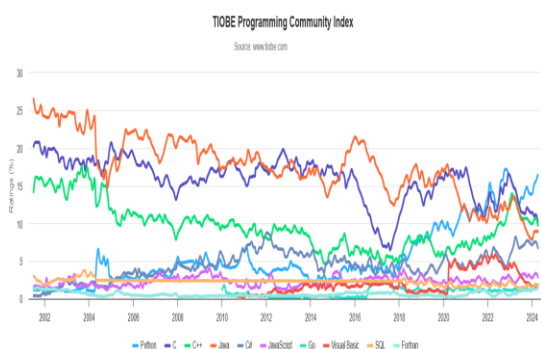


Figura 3: Ranking do TIOBE [24]

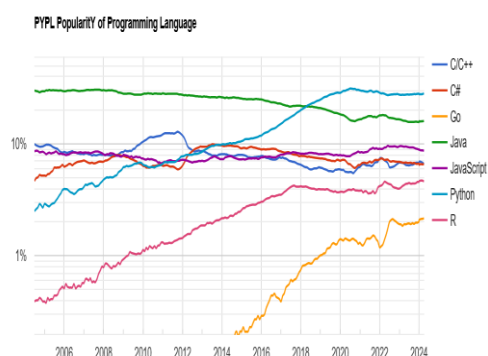


Figura 4: Ranking do PYPL [25]

Python [27] é uma linguagem de fácil manipulação e que, por ter um número significativo de utilizadores, existe bastante documentação. O facto de ser integralmente público (*open source*) facilita também a colaboração dos utilizadores desenvolvendo módulos e bibliotecas.

O *backend* foi desenvolvido em Django [28] que é uma framework web de alto nível para implementação de código Python. No Django, foi ainda necessário o uso de diferentes bibliotecas do Python, desde Pathlib [29], Datetime [30], Corsheaders [31], Rest_framework [32], para identificar algumas.

Anaconda

Conda é uma ferramenta de linha de comandos que serve para gerir diferentes ambientes e packages, usando diferentes versões de cada software [33]. No projeto em

questão o código dos Modelos de Atenção utilizados exigia versões específicas de diferentes packages, pelo que esta ferramenta foi essencial para a concretização do projeto.

React

React [34] é uma ferramenta muito utilizada para a criação de código *frontend*, ou seja, que permite a criação de interfaces para o utilizador, comunicando com o *backend* para obtenção de dados. Foi usado VITE [35] para contruir este projeto de forma mais rápida e leve e também vários componentes que foram necessários para concretizar o projeto: axios [36], jwt-decode [37], mysqlclient [38] são alguns dos componentes instalados.

MySQLFreeNetHosting

Em relação à base de dados para persistir os dados dos utilizadores, bem como outros dados que pudessem ser pertinentes, foi usada uma base de dados relacional do tipo MySQL, neste caso a base de dados foi alojada em servidores do MySQLFreeNetHosting.net [39].

HTML

Foi necessária também a criação de uma página web de modo a visualizar o resultado obtido nos mapas de saliência do conjunto de dados fornecidos. HTML – Hyper Text Markup Language é a linguagem em que a maior parte das páginas web são desenhadas, normalmente complementada design mais apelativo codificado em CSS [40].

2.6 Sumário

Neste capítulo foi detalhado o tema proposto, tendo o mesmo sido contextualizado com o recurso a conteúdos teóricos e científicos.

Foram explorados conteúdos teóricos relativos ao Marketing Digital de modo a entender os objetivos propostos e como seria possível dar resposta aos mesmos. Por outro lado, ao aprofundar o estudo da Aprendizagem Automática e as suas sub-divisões foi possível compreender a proveniência dos Modelos de Atenção e o seu enquadramento na Aprendizagem Profunda.

Sempre tendo em consideração os objetivos do projeto, realizou-se uma pesquisa aprofundada sobre os trabalhos relacionados com o tema proposto de modo a poder estruturar a solução que foi desenhada e posteriormente implementada. Deste modo, foi possível adicionar uma compreensão avançada das contribuições científicas para o tema proposto e ainda confirmar a inexistência de sistemas semelhantes ao que foi desenvolvido.

Finalmente, realizou-se um destaque às tecnologias e às suas capacidades no que diz respeito à implementação do sistema proposto.

3 Análise e desenho da solução

Tendo efetuado na secção anterior a contextualização do tema a abordar e quais os assuntos mais relevantes, torna-se agora importante começar a análise do problema que se pretende resolver.

A documentação do processo de análise e posterior desenho da solução será realizada através do levantamento de requisitos funcionais e não funcionais, de acordo com o modelo FURPS+ [41].

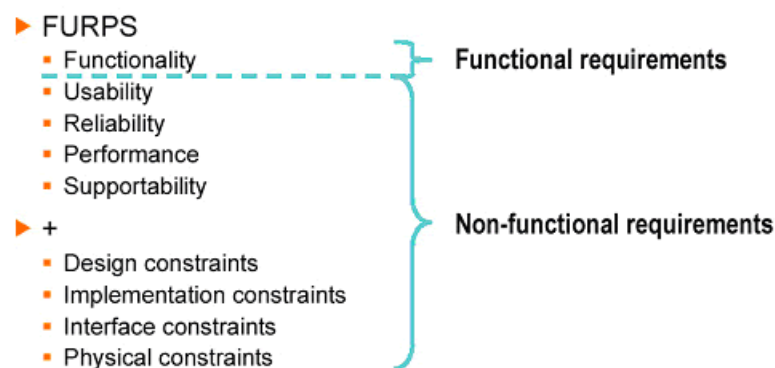


Figura 5: Modelo FURPS+ [41]

Tal como se pode verificar na Figura 5, o modelo FURPS divide-se em requisitos funcionais: **F**uncionalidade, **U**sabilidade, **R**eliabilidade (*Reliability*), **P**erformance (*Performance*), **S**uportabilidade e em requisitos não funcionais: restrições de design, restrições de implementação, restrições de interface e restrições de segurança. Cada um deste tipo de requisitos será detalhado e aprofundado, sendo devidamente enquadrado no problema em questão.

3.1 Domínio do problema

Antes de começar com a análise propriamente dita dos requisitos funcionais e não funcionais, é importante esclarecer o domínio em que o problema se vai centrar.

De facto, este projeto prende-se essencialmente com o desenvolvimento de uma ferramenta que irá ser útil na implementação da análise de imagens ou vídeos que, até hoje,

teriam de ser manipulados um a um, tendo em conta qual o modelo de análise escolhido. Isto significa que, para cada imagem ou vídeo que se pretendesse analisar, teríamos diferentes implementações para cada modelo de análise.

Tendo em conta o âmbito do Marketing Digital, essa ferramenta será muito importante para melhorar esta análise. Pretende-se assim que, numa ferramenta simples e intuitiva, qualquer utilizador possa escolher o modelo de análise numa página web e, ao clicar num botão, obter o mapa de saliências resultante desse método, sendo esse mesmo resultado mostrado ao utilizador na mesma página web, paralelamente à imagem ou vídeo original.

Estes resultados, obtidos através de vários métodos de Modelos de Atenção, permitirão ao utilizador adaptar as imagens e obter o mapa de saliências respetivo, repetindo facilmente o processo quantas vezes forem necessárias, até obter o resultado que pretende.

Por outro lado, na análise de vídeos, foram usados vídeos retirados de mundos virtuais com o intuito de analisar a potencialidade de colocar publicidade no mundo virtual e perceber qual a atenção que o utilizador irá ter. Deste modo, também se pode escolher a melhor localização do mundo virtual para essa campanha, bem como adaptar a própria imagem publicitária atuando da mesma maneira usada para a otimização de imagens estáticas.

Com o recurso a linguagens gráficas como o UML, serão criados artefactos que podem ajudar na compreensão do problema e que permitem visualizar as ligações entre os vários intervenientes no problema, bem como especificar as responsabilidades de cada uma delas.

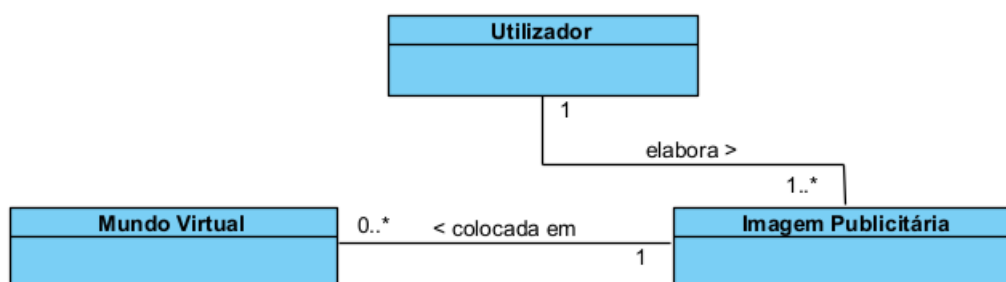


Figura 6: Modelo de Domínio

A Figura 6 pretende demonstrar o domínio do problema caracterizado pela existência de uma imagem publicitária elaborada por um utilizador. Essa imagem pode ou não ser colocada num mundo virtual. Pode-se facilmente entender as entidades que poderão usar o sistema desenvolvido, uma vez que o utilizador poderá fazer parte de uma empresa publicitária, com clientes que irão adquirir serviços nessa área, neste caso, a criação da

imagem com o melhor resultado de atenção dos seus clientes, bem como a possível colocação dessa imagem num mundo virtual, de modo a obter um melhor resultado da publicidade, apostando em vários canais de marketing.

3.2 Requisitos funcionais e não funcionais

A engenharia de requisitos é um processo de análise e levantamento de necessidades do problema e permite retirar ideias antes da formulação da solução propriamente dita. Esses requisitos podem ser divididos em 2 grandes categorias, requisitos funcionais e não funcionais.

3.2.1 Requisitos Funcionais

Estes requisitos definem as unidades funcionais, os limites de cada funcionalidade e os limites dos seus efeitos nas operações individuais e conjuntas do sistema [42]. O diagrama de casos de uso é o diagrama que melhor explicita as funcionalidades do sistema, tal como se pode verificar na Figura 7.

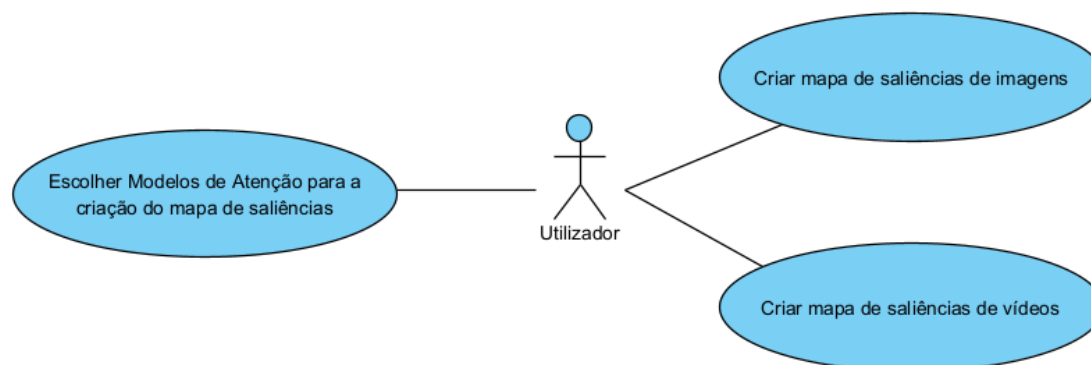


Figura 7: Diagrama de Casos de Uso

RF001: Escolher Modelos de Atenção para a criação do mapa de saliências

O primeiro requisito funcional do sistema desenvolvido é o facto do utilizador ter a opção de escolher qual o modelo de atenção que pretende usar na criação no mapa de saliências. Assim, o sistema deverá ter a possibilidade de escolha de mais do que um modelo de atenção de modo a ser possível comparar os mapas de saliências obtidos por cada um dos modelos.

RF002: Criar mapa de saliências de imagens

O requisito funcional de criar o mapa de saliências de imagens tem a ver com a implementação propriamente dita dos Modelos de Atenção, escolhidos no requisito anterior, e obter o resultado previsto pelo modelo. Com este resultado, pretende-se ter uma previsão de como o utilizador irá interagir, ou seja, como vai percecionar e focar a sua atenção em cada parte da imagem fornecida como entrada de dados.

RF003: Criar mapa de saliências de vídeos

O requisito funcional de criar um mapa de saliências de vídeos vai ser usado para receber como dado de entrada um vídeo de realidade virtual, dividir esse vídeo em *frames* que correspondem a imagens e, para cada imagem, converter essa imagem num mapa de saliências à semelhança do RF002. Posteriormente, essas imagens geradas pelo modelo de atenção escolhido, serão agrupadas num vídeo para que o utilizador possa comparar esse resultado com o vídeo original. Deste modo, será possível verificar quais são as zonas do vídeo em que o utilizador presta mais atenção, de modo a que se possa usar esse aumento de atenção para, por exemplo, criar uma publicidade e assim conseguir mais foco do utilizador.

Os requisitos funcionais deste sistema estão muito ligados entre eles, pelo que foi criado um diagrama de sequência genérico que engloba os 3 requisitos funcionais. O principal ponto de foco é que apenas utilizadores autenticados têm acesso às funcionalidades do sistema (Figura 8). A diferença de implementação para cada um dos requisitos funcionais será detalhado no capítulo seguinte.

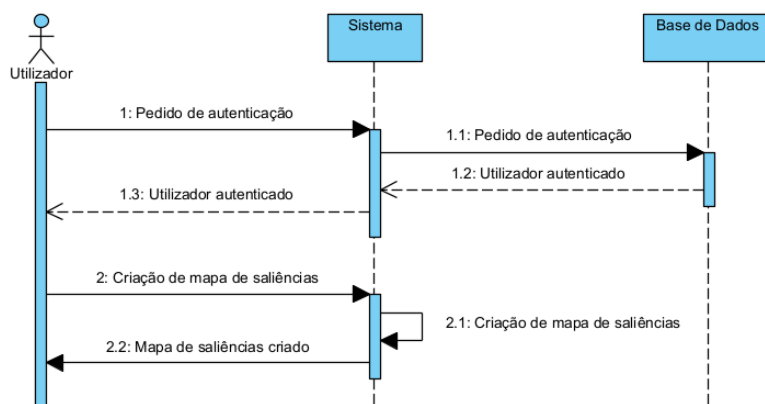


Figura 8: Diagrama de Casos de Uso

3.2.2 Requisitos Não Funcionais

Os requisitos não funcionais do sistema prendem-se com características do sistema que não têm a ver diretamente com a funcionalidade pretendida, mas sim com a forma como essas funcionalidades são mostradas ao cliente.

RNF001: Usabilidade

A usabilidade de um sistema tem a ver com a facilidade de seguir intuitivamente os passos necessários para que, no presente projeto, um determinado dado de entrada produza um determinado resultado. Essencialmente, tem como objetivo capturar e avaliar requisitos baseados na interface com o utilizador [43].

Tabela 2: Requisitos de Usabilidade

RNF001.1	O utilizador deverá ter uma curva de aprendizagem perante a solução desenvolvida reduzida
----------	---

No sistema a ser desenvolvido, pretende-se que a interface web com que o utilizador vai interagir seja simples, de forma a que o utilizador entenda facilmente quais os passos a dar para que forneça um dado de entrada e receba o resultado que pretende.

RNF002: Confiabilidade

A confiabilidade de um sistema tem a ver com a disponibilidade, precisão, recuperabilidade e tempo médio entre falhas [43].

Tabela 3: Requisitos de Confiabilidade

RNF002.1	O servidor deve-se manter operacional e funcional após falhas, devendo existir cópias de segurança para recuperar rapidamente o sistema.
RNF002.2	O servidor deverá ter alta disponibilidade, ou seja, disponível no mínimo em 99% do tempo.

Tal como é exposto na Tabela 3, no projeto a ser desenvolvido pretende-se que o servidor se mantenha operacional e funcional após falhas. Por outro lado, o servidor deverá ter alta disponibilidade, ou seja, disponível no mínimo em 99% do tempo.

RNF003: Desempenho

O desempenho de um sistema diz respeito a rendimento, tempo de resposta do sistema, tempo de recuperação e tempo de inicialização [43].

Tabela 4: Requisitos de Desempenho

RNF003.1	O sistema deverá produzir resultados de imagens, independentemente do Modelo de Atenção escolhido, no tempo máximo de 1 minuto.
RNF003.2	O sistema deverá disponibilizar um contador de tempo de processamento, para o utilizador acompanhar o progresso.

Neste projeto, o desempenho do sistema é algo que poderá ser bastante crítico. Isto deve-se não necessariamente à base do sistema, mas sim ao tempo necessário para o processamento e posterior visualização do resultado dos modelos de atenção. Como já foi referido, um dos problemas da Aprendizagem Profunda tem a ver com o tempo de processamento e neste projeto esse problema também será uma questão importante (Tabela 4).

RNF004: Suportabilidade

A suportabilidade de um sistema está relacionada com limitações de linguagem, testabilidade, adaptabilidade, manutenibilidade, compatibilidade e escalabilidade [43].

Tabela 5: Requisitos de Suportabilidade

RNF004.1	O sistema deverá permitir a inclusão simples de novas funcionalidades.
RNF004.2	O sistema deverá permitir a implementação de outros modelos de atenção de forma fácil.

O sistema deverá permitir a inclusão de novas funcionalidades, de maneira simples (Tabela 5). O sistema deverá ser desenvolvido de maneira a poder incluir outros modelos de

atenção mantendo o design da página web. Estes requisitos permitem que o sistema seja flexível, facilitando a manutenibilidade e melhorando a adaptabilidade.

RNF005: Restrições de desenho

Nas restrições de desenho é limitada a forma como a aplicação é desenhada [43].

Tabela 6: Restrições de Desenho

RNF005.1	O sistema deverá comunicar com a base de dados de forma assíncrona.
----------	---

No presente projeto, a comunicação com a base de dados deve ser assíncrona (Tabela 6), uma vez que essa comunicação não será contínua e assim evita-se a latência do sistema. A base de dados a implementar deverá conter, numa primeira fase, apenas os dados necessários para registo e autenticação no sistema.

RNF006: Restrições de implementação

Num sistema implementado em páginas web, é importante a compatibilidade entre diferentes navegadores de internet (Tabela 7). Este é um exemplo de uma restrição de implementação, uma vez que afeta a forma como o sistema é concebido, causando por vezes impacto no design da solução [43].

Tabela 7: Restrições de Implementação

RNF006.1	O sistema deverá poder ser utilizado em diferentes navegadores de internet.
----------	---

RNF007: Restrições de interface

No caso das restrições de interface, estas afetam a forma como certos componentes comunicam com outros componentes [43]. Na solução em questão, a ligação à internet

permanente é fundamental para que não haja interrupções no serviço a disponibilizar (Tabela 8).

Tabela 8: Restrições de Interface

RNF007.1	O sistema deverá manter a ligação à internet permanentemente.
----------	---

RNF008: Restrições de segurança

No caso das restrições de segurança, estas afetam a acessibilidade a certos componentes do sistema [43]. Na solução em questão, apenas utilizadores registados no sistema poderão usufruir de todas as funcionalidades implementadas (Tabela 9).

Tabela 9: Restrições de segurança

RNF007.1	Apenas utilizadores registados poderão aceder ao sistema.
----------	---

3.3 Desenho

O desenho da solução é o processo de identificar a melhor estrutura e funcionamento do programa desenvolvido de forma a responder com eficiência aos requisitos funcionais e não funcionais identificados na secção anterior.

3.3.1 Arquitetura do sistema

Tendo em conta as boas práticas de desenvolvimento de software e o domínio do problema apresentado na secção 3.1 deste relatório, foi feita a divisão do sistema por camadas. O principal objetivo desta divisão está relacionado com a responsabilidade de cada camada ser única e assim poder-se atuar facilmente aquando de processo de manutenção e escalabilidade, melhorando também a adaptabilidade e a segurança. Foram então definidos os seguintes componentes:

- *User Interface* codificado em React, onde vai estar desenhada a interface que o utilizador vai aceder. Esta interface irá receber os pedidos do utilizador, comunicar

com o *Web Service* para solicitar dados e apresentar os mesmos ao utilizador no *Browser*

- *Web Service*, codificado em Django é o componente responsável por interagir com a base de dados, para posterior envio desses dados à *User Interface*. Irá também possuir um serviço para pedir obter os dados dos mapas de saliências, usando uma ligação aos módulos *Python*.
- Modelos de Atenção, codificados em *Python*, serão os módulos de criação dos mapas de saliências, realizando a previsão dos modelos de atenção.
- Base de dados será um componente para registo dos dados de utilizador. Este componente deverá comunicar com o componente *Web Service*.
- *Browser* é o componente que irá ser usado pelo utilizador para realizar pedidos ao sistema e obter informação de forma simples e intuitiva para o utilizador, através de uma página HTML com CSS.

Deve-se referir que o *dataset*, ou seja, o conjunto de dados a ser analisado não faz parte do sistema, uma vez que será carregado dinamicamente a partir de qualquer dispositivo de armazenamento que o utilizador queira usar. Este desenho da solução é mais facilmente visível num diagrama de componentes, tal como se mostra na Figura 9.

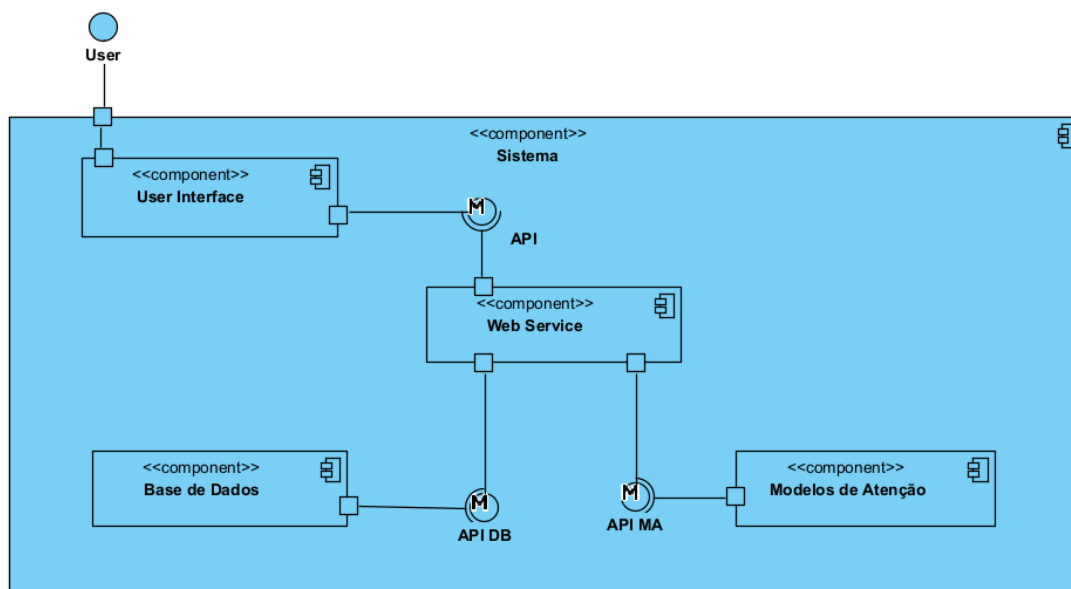


Figura 9: Diagrama de componentes do sistema

Na Figura 9 pode então confirmar-se visualmente a ligação entre os diversos componentes que compõem o sistema. O utilizador comunica com o sistema através de uma *browser*, identificado na Figura 9 pela porta de comunicação com o exterior. Esse componente, *browser*, comunica então com a *User Interface*, que corresponde ao componente onde está codificada e desenhada a interface que o *browser* irá renderizar. Este componente vai então receber e enviar pedidos para o *Web Service*, que corresponde ao componente de *backend*. O *backend* vai interpretar e redirecionar os pedidos para a base de dados ou para o componente onde estão implementados os modelos de atenção.

Tendo em conta os componentes acima explicitados é importante perceber também como eles estarão inseridos fisicamente nas estruturas disponíveis. De forma a validar esta implementação física, de seguida mostra-se na Figura 10 o diagrama de implantação do sistema.

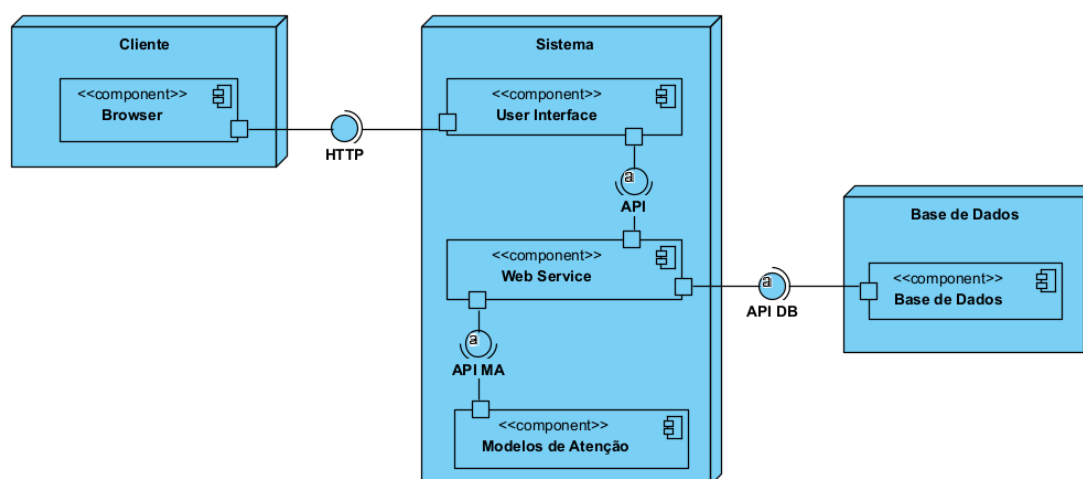


Figura 10: Diagrama de implantação do sistema

Finalmente, tendo em conta os vários componentes descritos anteriormente, procedeu-se ainda à elaboração do diagrama BPMN do sistema. O diagrama BPMN é um tipo de fluxograma que utiliza ícones padronizados para representar os diversos componentes e elementos do sistema e ainda o fluxo de informação entre eles [44].

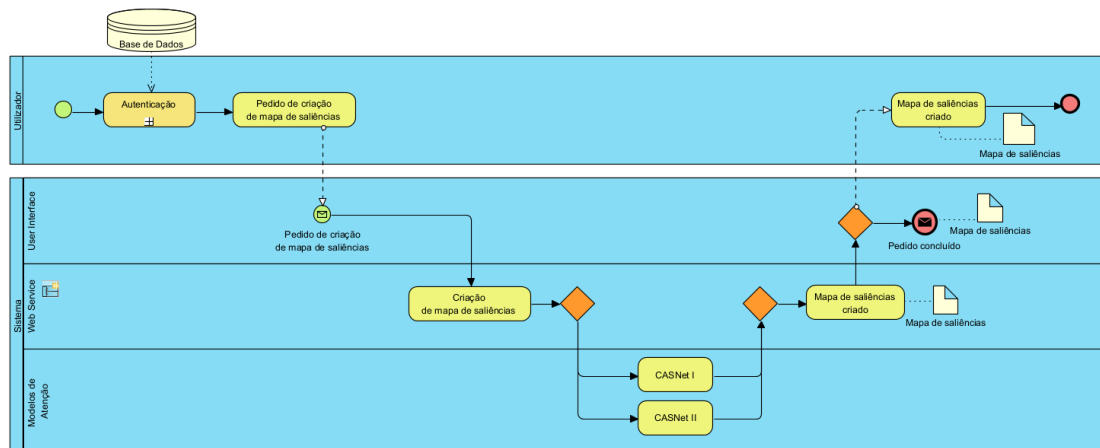


Figura 11: Diagrama BPMN do sistema

A Figura 11 representa o fluxo de informação entre os diversos componentes do sistema aquando de um pedido de criação de um mapa de saliências.

3.3.2 Solução proposta

Na solução proposta, temos os diferentes componentes com responsabilidades únicas e bem definidas, pelo que a solução consegue corresponder aos requisitos funcionais e não funcionais.

O facto de haver vários componentes permite uma melhor escalabilidade do sistema, bem como uma melhor adaptabilidade. De facto, se houver no futuro a necessidade de trocar a base de dados, por exemplo, apenas se terá de atualizar a ligação da API DB para que essa troca mantenha o sistema em funcionamento.

Nesta proposta de solução, existe também uma API MA (API Modelos de Atenção) que fará a ligação com o componente Modelos de Atenção que terá várias implementações, tantas quantas os Modelos de Atenção a serem implementados.

3.3.3 Solução alternativa

Como solução alternativa, poder-se-ia pensar na ligação da base de dados diretamente ao componente da *User Interface*, em vez da ligação ao componente *Web Service*. Pode-se analisar esta solução alternativa na Figura 12.

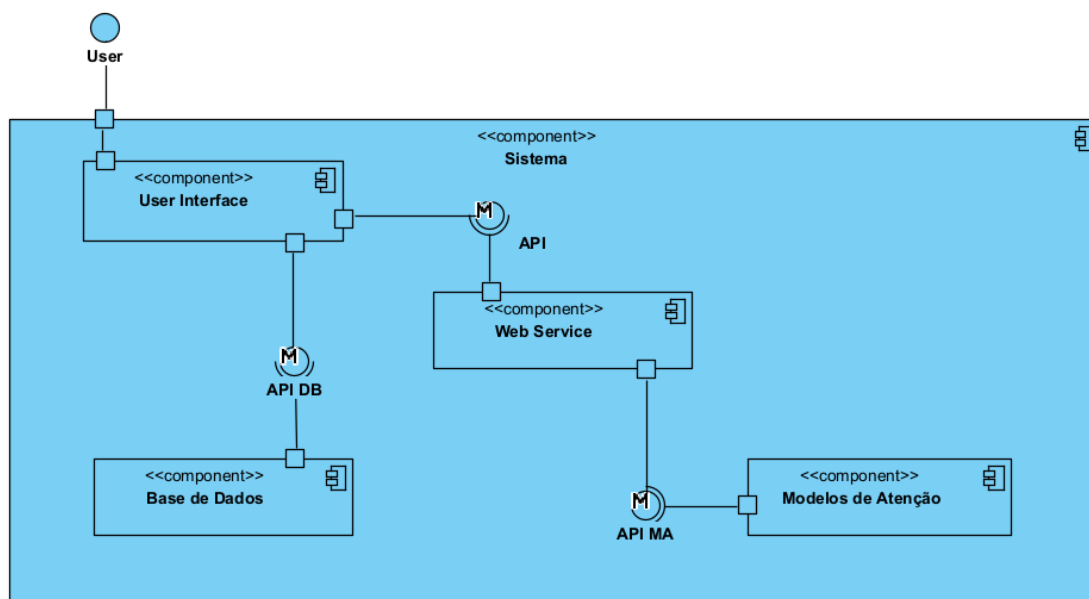


Figura 12: Diagrama de componentes alternativo do sistema

De facto, a base de dados tal como definido no âmbito deste projeto deveria apenas guardar os dados do utilizador. No entanto, no desenvolvimento do projeto achou-se complementar a base de dados com mais informação relevante, como por exemplo ter a possibilidade de escrever comentários sobre as imagens desenvolvidas.

Deste modo, esta solução mantém os padrões de desenvolvimento de software, mas tem um risco acrescido em segurança e também na usabilidade do sistema uma vez que não permite a adição de novas funcionalidades a serem persistidas na base de dados.

Em termos de segurança, o facto de os dados dos utilizadores passarem pelo *Web Service* permite uma camada adicional de segurança, pelo que esta solução é menos suscetível a ataques de pirataria e de roubo de dados confidenciais dos utilizadores. Por outro lado, se no futuro houver a necessidade de guardar outros dados na base de dados, ter-se-ia de criar outra ligação entre o *Web Service* e a base de dados, uma vez que esta solução não contempla essa opção.

Comparando as duas soluções, optou-se pela solução proposta, uma vez que a solução alternativa pode resultar em problemas de segurança futuros. Se por um lado, as possíveis fugas de informação (dados de login e notas) não são críticas nem afetariam os utilizadores envolvidos, deve-se desenhar um sistema que seja robusto e o menos susceptível a ataques tanto quanto possível. Este desenho permitirá a evolução do sistema para inclusão de novas funcionalidades, sem que haja necessidade de mudar o que já foi desenvolvido.

3.4 Sumário

Neste capítulo de análise e desenho da solução procedeu-se inicialmente à análise do problema detalhando o domínio do mesmo e definindo os diversos requisitos funcionais e não funcionais do sistema.

Procedeu-se posteriormente à estruturação da arquitetura do sistema, assegurando o cumprimento dos requisitos funcionais definidos. Por outro lado, a solução proposta deve também cumprir os requisitos não funcionais e, para isso, optou-se por uma solução em que cada componente tem uma responsabilidade única, pelo que facilitará o processo futuro de manutenção e escalabilidade, melhorando também a adaptabilidade e a segurança.

Foram definidos 4 componentes para este sistema:

- *User Interface* que irá receber os pedidos do utilizador, comunicar com o *Web Service* para solicitar dados e apresentar os mesmos ao utilizador no *Browser*
- *Web Service*, ou backend, que deverá interagir com a base de dados e um serviço para pedir obter os dados dos mapas de saliências, usando uma ligação ao componente dos Modelos de Atenção.
- Modelos de Atenção é o componente com a responsabilidade de criação dos mapas de saliências, realizando a previsão dos modelos de atenção.
- Base de dados será um componente para registo dos dados de utilizador.

São também explicitados neste capítulo os vários diagramas que ajudam a entender o modo como estes componentes serão implantados e como é realizado o fluxo de informação entre eles.

Finalmente é ainda discutida uma proposta de solução alternativa, em que a base de dados, por não conter informação que possa ser considerada sensível, estaria ligada diretamente ao *frontend*, solução que não foi adotada, uma vez que traria implicações futuras no caso de ser necessária a evolução do sistema para algo mais complexo ou com outro tipo de dados persistidos.

4 Implementação e Avaliação da Solução

Neste capítulo, irão ser apresentados os detalhes relacionados com o enquadramento e implementação das soluções explicitadas no capítulo anterior. Assim, o modo como os 4 principais componentes, *User Interface*, *Webservice*, Modelos de Atenção e a Base de Dados foram implementados serão detalhados neste capítulo.

4.1 Descrição da implementação

O primeiro passo num projeto deste género é criar, e posteriormente ativar, um ambiente virtual onde serão instalados os *packages* específicos para a execução desse mesmo projeto. Deste modo, nenhum outro projeto será afetado pela instalação ou não de certos *packages*, bem como se assegura que apenas os *packages* necessários neste projeto são os que estão instalados.

Como foi referido no capítulo anterior usou-se o Conda [33], na versão mini-conda para essa instalação de *packages* específicos necessários para o código dos Modelos de Atenção funcionar corretamente.

Tendo em conta a especificidade e complexidade de usar os Modelos de Atenção exigidos na proposta inicial deste projeto, foi necessário também o uso do *Jupyter Lab* [45]. O *Jupyter Lab* é um *software* de desenvolvimento gratuito, muito flexível e com base *web* que pode ser usado para vários fins, entre os quais *machine learning*.

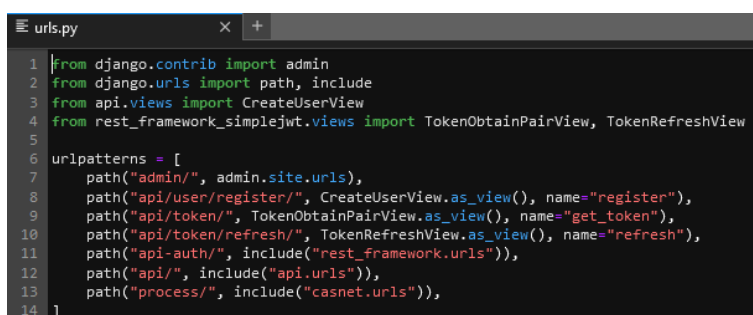
4.1.1. Web Service

O módulo *Web Service*, ou seja, o *backend* foi desenvolvido em Python através da *framework* Django. Para além de ser uma linguagem muito utilizada, o fato do código dos Modelos de Atenção também ser desenvolvido em Python ajudou à comunicação entre os vários componentes. Por outro lado, uma vez que era desconhecida esta linguagem à partida para este projeto, fez com que o contato com Python não tivesse sido apenas no código dos Modelos de Atenção, mas sim para todo o desenvolvimento do *backend*.

Após a ativação do ambiente virtual do projeto, é então necessária a instalação de todas as dependências para a execução do projeto. Já era sabido que seria necessária a instalação do Django [28], ou `djangorestframework` [32], ou ainda `PyJWT` [46] para nomear algumas. Depois dessa instalação, é necessária a criação de um novo projeto Django. Com esta criação é criada uma pasta com a informação genérica deste projeto, como por exemplo os *settings* ou *urls*.

De seguida, deve ser criada uma Django app onde se podem personalizar os diversos componentes e, assim, organizar o código logicamente. Deverá ser criada uma nova app para cada tipo de funcionalidade da aplicação, de modo a ser mais fácil a sua gestão e manutenibilidade.

Depois de serem personalizados alguns dos *settings* genéricos do projeto, como por exemplo, os dados de acesso à base de dados, o próximo passo é a criação dos *end-points* que teremos no projeto. Os *end-points* podem ser criados genericamente na app que contém a informação genérica do projeto, ou individualmente em cada app criada. No presente caso, foram criados *end-points* de autenticação no ficheiro genérico do programa e *end-points* específicos em cada app que, posteriormente, são chamados do ficheiro genérico `url.py`.



```
1 from django.contrib import admin
2 from django.urls import path, include
3 from api.views import CreateUserView
4 from rest_framework_simplejwt.views import TokenObtainPairView, TokenRefreshView
5
6 urlpatterns = [
7     path("admin/", admin.site.urls),
8     path("api/user/register/", CreateUserView.as_view(), name="register"),
9     path("api/token/", TokenObtainPairView.as_view(), name="get_token"),
10    path("api/token/refresh/", TokenRefreshView.as_view(), name="refresh"),
11    path("api-auth/", include("rest_framework.urls")),
12    path("api/", include("api.urls")),
13    path("process/", include("casnet.urls")),
14 ]
```

Figura 13: Ficheiro `url.py`

Ao serem criados os *end-points*, isto é, os endereços da internet específicos para cada caso de uso, definem-se as *views* a serem executadas. Em Django, as *views* são funções que recebem um pedido *web* e retornam uma resposta *web*. No projeto em questão, foram criadas várias funções na app “api”: uma para criar utilizadores, uma para criar e listar notas, uma para apagar notas. Na imagem abaixo, é possível verificar o caso específico da criação e listagem de notas. Inicialmente especifica-se a classe usada para a serialização, define-se quais as permissões (neste caso, apenas para utilizadores autenticados) e depois existem 2 sub-funções: uma para retornar da base de dados as notas filtradas pelo utilizador autenticado e

outra para persistir na base de dados os parâmetros que vêm da APIView, juntamente com o utilizador autenticado.

```

8 class NotelistCreate(generics.ListCreateAPIView):
9     serializer_class = NoteSerializer
10    permission_classes = [IsAuthenticated]
11
12    def get_queryset(self):
13        user = self.request.user
14        return Note.objects.filter(author=user)
15
16    def perform_create(self, serializer):
17        if serializer.is_valid():
18            serializer.save(author=self.request.user)
19        else:
20            print(serializer.errors)

```

Figura 14: Ficheiro views.py

No caso da app “casnet”, que deu suporte ao cálculo dos mapas de saliências obtidos, as funções criadas prendem-se com o processamento de imagens e outra com o processamento de vídeos.

O Django já possui um módulo de autenticação pelo que se fez uso desse módulo para tratar todo o acesso à aplicação, não sendo necessária nenhuma implementação específica. Por outro lado, foi necessária a criação de um modelo de Nota, na app “api”. Os modelos representam as classes a serem persistidas. Cada atributo do modelo representa um campo na base de dados e cada modelo mapeia uma tabela na base de dados.

```

1 from django.db import models
2 from django.contrib.auth.models import User
3
4
5 class Note(models.Model):
6     content = models.TextField(max_length=500)
7     imageURL = models.CharField(max_length=80)
8     model = models.CharField(max_length=50)
9     created_at = models.DateTimeField(auto_now_add=True)
10    author = models.ForeignKey(User, on_delete=models.CASCADE, related_name="notes")
11
12    def __str__(self):
13        return self

```

Figura 15: Ficheiro models.py

Podemos assim verificar os diferentes atributos da classe Note. Para estes atributos poderem ser persistidos na base de dados, é necessária a criação do ficheiro serializers.py onde se detalham os atributos e outras definições do que ficará persistido na base de dados. Os detalhes deste ficheiro serão apresentados na seção dedicada à base de dados.

O módulo de autenticação do Django faz uso do JWT (*Json Web Token*). Sempre que uma página *web* é acessada, o JWT vai atuar como as permissões ou uma autenticação a essa página *web*. Uma vez que, sempre que fazemos um pedido *web*, o *backend* tem de saber quem fez o pedido e que permissões tem, é incluído no pedido um *token* JWT que será decodificado de modo a perceber quais são as permissões que esse pedido tem. Inicialmente, um utilizador inicia o login e, ao colocar os seus dados de acesso, o *frontend* envia essas credenciais ao *backend*. O *backend* confirma a veracidade dessas credenciais e concede 2 *tokens*, um *token* de acesso e um *token* de atualização que serão guardados temporariamente no *frontend* [47]. O *token* de acesso será o usado em todos os pedidos e, para aumentar a segurança, este *token* tem uma validade curta, no caso foi definido 30 minutos. Assim, é definido um *token* de atualização, que vai atualizar o *token* de acesso frequentemente, mas que também tem um tempo de vida (24 horas) após o qual as credenciais de acesso terão de ser fornecidas novamente. Este processo e tipo de acesso aumenta consideravelmente a segurança do sistema desenvolvido. É, assim, necessária a configuração dos tempos limites dos *tokens* de acesso e de atualização, que foram executados no ficheiro genérico de configurações do projeto *settings.py*.

```
34 REST_FRAMEWORK = {
35     "DEFAULT_AUTHENTICATION_CLASSES": (
36         "rest_framework_simplejwt.authentication.JWTAuthentication",
37     ),
38     "DEFAULT_PERMISSION_CLASSES": [
39         "rest_framework.permissions.IsAuthenticated",
40     ],
41 }
42
43 SIMPLE_JWT = {
44     "ACCESS_TOKEN_LIFETIME": timedelta(minutes=30),
45     "REFRESH_TOKEN_LIFETIME": timedelta(days=1),
46 }
```

Figura 16: Ficheiro *settings.py*

4.1.2. Modelos de Atenção

A implementação dos Modelos de Atenção CASNet I [21] e CASNet II [20] foi sem dúvida a parte mais complexa deste projeto. Apesar do código fonte destes modelos estar disponível online, o seu uso é bastante complexo devido a vários fatores. O que é provavelmente mais difícil prende-se com as versões específicas de *packages* necessários à execução de cada modelo. Foi necessário obter essa informação, instalar cada *package* na versão correta, sendo que a maior parte das vezes não era a versão mais atual, de modo a poder executar o código, de outro modo era impossível a obtenção de qualquer resultado.

Convém relembrar que esta instalação foi realizada no ambiente virtual criado especificamente para todo o projeto.

Após ser ultrapassado esse problema, foram criados 2 *end-points* no *backend*, mais concretamente na app criada especificamente para o cálculo dos Modelos de Atenção “casnet”, sendo um específico para o processamento de imagens e outro para o processamento de vídeos.

```
@csrf_exempt
def process_image(request):
    if request.method == 'POST':
        image = request.FILES['image']
        option = request.POST.get('option', '')

        # Save the uploaded image to a temporary location
        image_path = "C:/Pedro/ISEP/PESTI/backend/casnet/00000.jpg"
        with default_storage.open(image_path, 'wb+') as destination:
            for chunk in image.chunks():
                destination.write(chunk)

        # Define the path to the Python script based on the selected option
        if option == 'casnet1':
            script_path = 'C:/Pedro/ISEP/PESTI/backend/casnet/casnet1code.py'
        elif option == 'casnet2':
            script_path = 'C:/Pedro/ISEP/PESTI/backend/casnet/casnet2code.py'
        else:
            return JsonResponse({'error': 'Invalid option'}, status=400)

        # Verify the paths
        if not os.path.exists(image_path):
            return JsonResponse({'error': f'Image path does not exist: {image_path}'}, status=500)
        if not os.path.exists(script_path):
            return JsonResponse({'error': f'Script path does not exist: {script_path}'}, status=500)

        # Execute the Python script
        try:
            result = subprocess.run(
                ['python', script_path, image_path],
                check=True,
                capture_output=True,
                text=True # Capture stdout and stderr as strings
            )

            # copy output image to final folder
            output_image = 'C:/Pedro/ISEP/PESTI/backend/casnet/temp.jpg'
            output_path = 'C:/Pedro/ISEP/PESTI/frontend/src/images/created_image.jpg'
            shutil.copy(output_image, output_path)

            # Delete the temporary files
            os.remove(image_path)
            os.remove(output_image)

            return JsonResponse({'success': True})

        except subprocess.CalledProcessError as e:
            return JsonResponse({'error': str(e)}, status=500)

    return JsonResponse({'error': 'Invalid request method'}, status=400)
```

Figura 17: Função de processamento de imagens

No processamento das imagens, tal como se pode verificar na Figura 17, criou-se uma função no ficheiro *views.py* que realizar os seguintes passos: inicialmente vai receber o pedido

do *frontend*, copiar a imagem para uma localização temporária, verificar qual a opção recebida do *frontend* (ou seja, qual o método a utilizar), executar o script do modelo escolhido e finalmente copiar o mapa de saliências criado pelo modelo para a localização que o *frontend* vai renderizar. Existem ainda nesta a função a verificação dos caminhos para os scripts dos modelos e no fim da função a remoção das imagens temporárias.

Para o processamento de vídeos, para além do código descrito necessário para o processamento de imagens, foi necessária a criação de 2 *scripts* auxiliares, um para dividir o vídeo em *frames* e outro para agrupar os mapas de saliências de cada *frame* e construir um vídeo desses mapas.

No caso do script para dividir o vídeo em frames, foi necessário o uso da biblioteca moviePy [48]. Neste script criado no ficheiro *split.py*, tal como é possível verificar na figura abaixo, foi escolhido retirar 2 *frames* por segundo de vídeo e cada *frame* é gravado temporariamente numa pasta a designar pela função que vai executar esse script.

```

1 from moviepy.editor import VideoFileClip
2 import os
3 import sys
4
5 def extract_frames(clip, times, frames_dir):
6     if not os.path.exists(frames_dir):
7         os.makedirs(frames_dir)
8
9     for t in times:
10        video_path = os.path.join(frames_dir, '{:05d}.jpg'.format(int(t*clip.fps)))
11        clip.save_frame(video_path, t)
12
13 if __name__ == "__main__":
14     video_path = sys.argv[1]
15     frames_dir = sys.argv[2]
16
17     clip = VideoFileClip(video_path)
18     times = [i for i in range(0, int(clip.duration * 2))] # 2 frames per second
19     times = [t / 2.0 for t in times] # Convert frame numbers to seconds
20
21     extract_frames(clip, times, frames_dir)

```

Figura 18: Ficheiro *split.py*

Após a divisão do vídeo em *frames* a função de processamento de vídeos vai iterar em cada *frame* e convertê-lo num mapa de saliências, tal como foi descrito na função de processamento de imagens. Após todos os *frames* terem sido convertidos em mapas de saliências, é necessária a execução de um outro *script* para criação do vídeo com todos os mapas de saliências criados. Esse *script* vai juntar todos os mapas e convertê-los num vídeo. De referir que esta função de criação de vídeo (*VideoWriter*) está incluída na biblioteca cv2 [49] do *Python*, que é uma biblioteca gratuita (*open-source*) de visão computarizada.


```

1 import os
2 from PIL import Image
3 import cv2
4 import sys
5
6 # Get the current working directory
7 current_path = os.getcwd()
8
9 # Folder containing images
10 img_dir = './processed_frames'
11
12 #Generate a video from all images in the specified folder.
13 def create_video_from_images(folder):
14     video_filename = 'created_video.mp4'
15     valid_images = [i for i in os.listdir(folder) if i.endswith((".jpg", ".jpeg", ".png"))]
16
17     first_image = cv2.imread(os.path.join(folder, valid_images[0]))
18     h, w, _ = first_image.shape
19
20     codec = cv2.VideoWriter_fourcc(*'avc1')
21     vid_writer = cv2.VideoWriter(video_filename, codec, 30, (w, h))
22
23     for img in valid_images:
24         loaded_img = cv2.imread(os.path.join(folder, img))
25         for _ in range(20):
26             vid_writer.write(loaded_img)
27
28     vid_writer.release()
29
30 if __name__ == "__main__":
31     img_dir = sys.argv[1]
32     create_video_from_images(img_dir)

```

Figura 19: Ficheiro create_video.py

Uma das dificuldades com a criação do vídeo tem a ver com o codec utilizado. Inicialmente foi usado o mp4v que seria o codec para criação de vídeos em mp4, mas verificou-se que nem todos os browsers permitiam a visualização do vídeo, isto sem adicionar extensões ao browser. Assim, foi decidido a utilização do codec avc1 que é amplamente utilizado e onde é conseguida uma melhor flexibilidade, tendo em conta que os utilizadores do sistema poderão usar diferentes browsers.

Para além da dificuldade acerca da criação do vídeo, outra questão prende-se com o armazenamento na base de dados, devido à quantidade necessária de espaço. Deste modo, implementou-se a persistência na base de dados apenas das imagens, através de um *Binary Field*, enquanto que no caso dos vídeos a persistência dos mesmos foi apenas local de modo a manter a base de dados com menor quantidade de informação.

Na Figura 20 é possível verificar um print da tabela da base de dados correspondente onde se comprova a serialização das imagens tal como referido acima.







Opções				id	name	image	author_id
<input type="checkbox"/>	 Edita	 Copiar	 Apagar	15	rita.jpg	[BLOB - 14.4 KB]	2
<input type="checkbox"/>	 Edita	 Copiar	 Apagar	16	pingo2.jpg	[BLOB - 15.9 KB]	2

Figura 20: Persistência na base de dados de imagens

4.1.3. User Interface

A *User Interface* foi desenvolvida em *React*. Tal como referido anteriormente usou-se a *framework* *Vite.js* em vez de outras, como por exemplo *Next.js*. Apesar da *framework* *Next* ter algumas características interessantes como o ISR, que permite a atualização de páginas estáticas sem ser necessária a compilação da página completa ou ainda uma componente de otimização de imagens, como mudança de tamanho ou recorte das mesmas que poderia ser interessante no projeto em desenvolvimento, optou-se por usar a *framework* *Vite.js* [35].

De fato, esta *framework* permite uma divisão do código que faz com que diversas partes da página apenas vão carregadas à medida que são necessárias, bem como aceita diversos plugins como pré-processadores CSS ou outros que melhoram a rapidez do arranque e atualizações da página [50]. Sendo o tempo um fator crucial, tal como referido anteriormente, em projetos de Aprendizagem Profunda, a escolha da *framework* recaiu na que mais rápido desempenho teria.

Assim que foi criada a pasta com o *React* instalado, foi necessária a instalação de algumas dependências, tais como *Axios* [36], *Router-dom* [51] e *jwt-decode* [37]. O projeto template do *React* não tem divisões lógicas, pelo que foram criadas diversas pastas para organizar o código, tais como as páginas, estilos e componentes.

Inicialmente, foi criado um intercetador de pedidos *web*. Este código permite, para cada pedido, verificar se existe um *token* de acesso válido e, se houver, anexá-lo no cabeçalho do pedido. Deste modo, em todo o código a ser escrito posteriormente não haverá a necessidade de continuamente verificar se existe esse *token*, este código permitirá verificar isso em cada pedido.

```
1 import axios from "axios";
2 import { ACCESS_TOKEN } from "../constants";
3
4 const apiURL = "/choreo-apis/newpesti/backend-qyo/v1"
5
6 const api = axios.create({
7   baseURL: "http://127.0.0.1:8000/" ? "http://127.0.0.1:8000/" : apiURL
8 });
9
10 api.interceptors.request.use(
11   (config) => {
12     const token = localStorage.getItem(ACCESS_TOKEN);
13     if (token) {
14       config.headers.Authorization = `Bearer ${token}`;
15     }
16     return config;
17   },
18   (error) => {
19     return Promise.reject(error);
20   }
21 );
22
23 export default api;
```

Figura 21: Intercetador de pedidos *web* para inclusão de *token* de acesso

Para além desta configuração, foram criados componentes *React* que são funções *javascript* que podem ser chamadas em diversas partes do código *frontend* para compor o que finalmente é mostrado ao utilizador no browser. Para além da criação do componente *Form*, que serviu de base às páginas de registo e de login do utilizador, sendo ambas muito semelhantes, talvez o componente mais importante de realçar é o componente de *ProtectedRoute*. A função deste componente é atualizar automaticamente o *token* de acesso, para que o utilizador possa continuar a ter acesso à aplicação, sem ter que se preocupar com a atualização do login. No caso de ter havido algum problema, ou ainda no caso do *token* de atualização ter expirado, então esta função vai pedir um novo login para que seja gerado um novo *token* de acesso.

A seguir à criação dos componentes, passou-se então à criação das páginas codificadas em *javascript*, para que depois sejam interpretadas no *browser*. Foram então criadas 4 páginas, uma para registo do utilizador, outra para login do utilizador, uma página genérica de erro e a página principal do projeto, *Home.jsx*. A página de registo de utilizador e de login foram criadas a partir do mesmo componente, apenas com redireccionamentos diferentes para as páginas seguintes. Na Figura 22 pode-se ver a página de login criada.

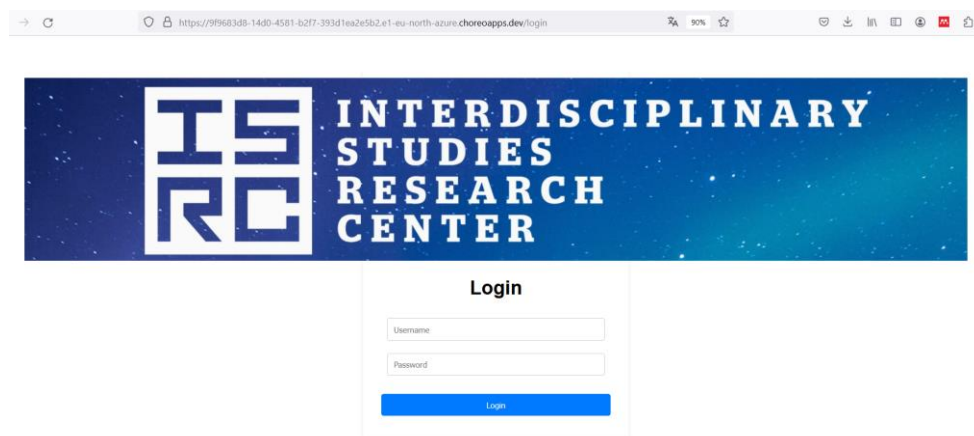


Figura 22: Página de login do sistema desenvolvido

Após o login bem sucedido, esta página reencaminha o utilizador para a *home page* onde todas as funcionalidades estão implementadas. Nesta página, foram criados botões específicos para a renderização de imagens e vídeos, um outro botão para escolha dos Modelos de Atenção, dois espaços para serem mostradas as imagens/vídeos escolhidos e o output dos modelos para melhor comparação. Finalmente, cada utilizador tem também a sua

própria área de comentários/notas que ficam ligados na base de dados à imagem e ao modelo utilizado no cálculo do respetivo mapa de saliências.

Figura 23: Home page do sistema desenvolvido após login

Na Figura 23 é possível verificar com detalhe as diferentes áreas da interface que o utilizador vai usar. Na parte superior tem a área da escolha da imagem / vídeo e escolha do Modelo de Atenção que pretende utilizar, na região do centro tem a visualização da escolha do utilizador, quer seja imagem, quer vídeo e também a visualização do resultado do Modelo de Atenção e ainda o botão para geração do mapa de saliências. Finalmente, numa região inferior da página, existe a área para criação e visualização de notas desse utilizador.

4.1.4. Base de dados

Neste projeto a base de dados foi desenhada de uma maneira simplista, uma vez que os dados a serem persistidos são dados simples, ou seja, apenas dados de login para acesso à ferramenta e, por outro lado, dados de texto simples para ser possível ao utilizador fazer corresponder um comentário ao mapa de saliências obtido. Assim, a pesquisa destes dados será também algo simples, pelo que a implementação de bases de dados mais modernas, do

tipo NoSQL [52], como Redis ou do tipo documentais, como MongoDB [53] não trariam vantagens para o projeto em questão. Estas bases de dados mais modernas são muito eficientes para dados não estruturados ou para dados que estão em constante mudança, mas no projeto em questão em que temos apenas dados estruturados e poucos tipos de dados, as bases de dados relacionais são mais eficientes.

Foi então necessário criar 2 classes para persistir os dados, uma para os utilizadores, a classe `UserSerializer` e a classe `NoteSerializer`.

```
class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ["id", "username", "password"]
        extra_kwargs = {"password": {"write_only": True}}

    def create(self, validated_data):
        user = User.objects.create_user(**validated_data)
        return user
```

Figura 24: Classe `User Serializer`

Tabela 10: Base de dados - Classe `User Serializer`

#	Nome	Tipo	Agrupamento (Collation)	Atributos	Nulo	Predefinido	Comentários	Extra
<input type="checkbox"/> 1	id	int(11)			Não	None		AUTO_INCREMENT
<input type="checkbox"/> 2	password	varchar(128)	latin1_swedish_ci		Não	None		
<input type="checkbox"/> 3	last_login	datetime			Sim	NULL		
<input type="checkbox"/> 4	is_superuser	tinyint(1)			Não	None		
<input type="checkbox"/> 5	username	varchar(150)	latin1_swedish_ci		Não	None		
<input type="checkbox"/> 6	first_name	varchar(150)	latin1_swedish_ci		Não	None		
<input type="checkbox"/> 7	last_name	varchar(150)	latin1_swedish_ci		Não	None		
<input type="checkbox"/> 8	email	varchar(254)	latin1_swedish_ci		Não	None		
<input type="checkbox"/> 9	is_staff	tinyint(1)			Não	None		
<input type="checkbox"/> 10	is_active	tinyint(1)			Não	None		
<input type="checkbox"/> 11	date_joined	datetime			Não	None		

Como é possível ver nas duas figuras anteriores, os campos codificados para serem persistidos na base de dados não coincidem com os campos da tabela na base de dados. De fato, esta diferença é devida ao fato do Django possuir um sistema de autenticação incorporado que contém todos os campos que ficam guardados na base de dados. Uma vez que a classe `User Serializer` apenas tem como campos a persistir o `id`, `username` e a `password`, apenas estes campos têm dados nas tabelas da base de dados. Uma possível melhoria futura poderia ser a introdução de mais campos no registo de cada utilizador, se tal fosse necessário por algum motivo. Caso contrário, poder-se-ia apagar os campos não usados das tabelas para melhorar o desempenho da base de dados.

A classe *Note Serializer*, para além do comentário do utilizador, tem também um campo relativo ao URL da imagem escolhida, um campo para o método de atenção e também uma chave estrangeira para associar estes dados ao utilizador que os executou, para além da data/hora da criação. De notar que no campo de utilizador vemos uma chave cinzenta, que tem o significado de chave estrangeira, ou seja, é o campo que liga esta tabela à tabela de utilizadores. As chaves douradas significam chaves primária, ou seja, o campo único da tabela que permite a identificação unívoca do registo.

```
class NoteSerializer(serializers.ModelSerializer):
    class Meta:
        model = Note
        fields = ["id", "content", "imageURL", "model", "created_at", "author"]
        extra_kwargs = {"author": {"read_only": True}}
```

Figura 25: Classe Note Serializer

Tabela 11: Base de dados - Classe Note Serializer



#	Nome	Tipo	Agrupamento (Collation)	Atributos	Nulo	Predefinido	Com
1	id	bigint(20)			Não	None	
2	content	longtext	latin1_swedish_ci		Não	None	
3	created_at	datetime			Não	None	
4	author_id	int(11)			Não	None	
5	imageURL	varchar(80)	latin1_swedish_ci		Não	None	
6	model	varchar(50)	latin1_swedish_ci		Não	None	

Posteriormente, foi necessário configurar os dados de acesso à base de dados nos settings do projeto e, assim, os dados passaram a ser persistidos na base de dados.

Salienta-se que o provedor da base de dados funciona bem, mas por vezes tem alguns problemas na latência de rede, o que foi problemático principalmente após o *deployment* do *frontend* e *backend*.

4.1.5. Controlo de versões

O controlo de versões é a prática de rastrear e gerir alterações no código desenvolvido. O *GitHub* [54] e o *Bitbucket* [55] são ambos *softwares* de controlo de versões, sendo que o *Bitbucket* é mais focado em repositórios privados e apenas permite 5 colaboradores nesses repositórios gratuitamente, ao passo que o *GitHub* é focado em repositórios públicos.

Comparando os dois *softwares* em termos de usabilidade, ambos são simples e permitem a integração com outras plataformas também bastante usadas em desenvolvimento de *software*.

De referir que no início do projeto estava a ser usado um repositório gerido no *Bitbucket*. Quando foi necessária a integração do código dos Modelos de Atenção no projeto Django entretanto desenvolvido, ocorreram diversos erros de conflito de versões de packages, pelo que foi necessário criar um ambiente novo, adicionar os modelos e posteriormente integrá-los noutra projeto Django. Na sequência destas alterações, foi necessário criar um novo repositório e, sabendo da boa integração do *Choreo* [56], *software* que irá permitir o *deployment* do sistema com o *GitHub*, optou-se por criar um novo repositório no *GitHub*.

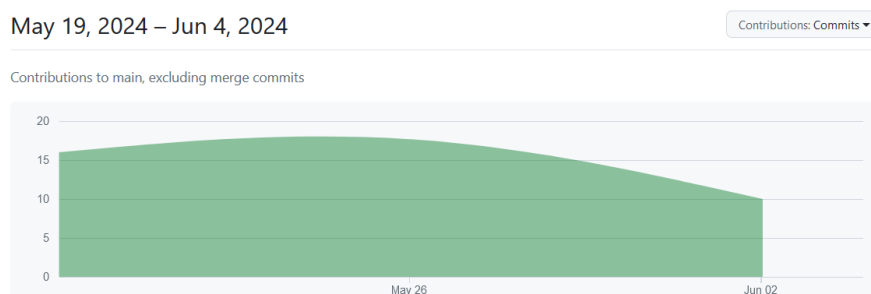


Figura 26: Commits realizados no repositório GitHub

Tal como se pode verificar na Figura 26, existiram sempre *commits* regulares, apesar da Figura acima do repositório *GitHub* não demonstrar a totalidade do tempo de desenvolvimento do projeto.

4.1.6. CI/CD

Uma *pipeline* é um processo que direciona o processo de desenvolvimento de *software* através de um caminho de compilação, realização de testes e implantação, tudo de forma automática. Este processo também é chamado de CI/CD que significa *Continuous Integration / Continuous Deployment*. Este processo pode ser realizado manualmente passo a passo, mas ao automatizar o processo, o objetivo é minimizar o erro humano e manter um processo consistente de passagem do *software* a ambiente de produção [57].

Numa *pipeline* podem ser incluídos diversas opções desde a compilação do código, testes unitários, análise de código ou ainda questões de segurança, por exemplo.

Para a criação da *pipeline* deste projeto usou-se a ferramenta *Choreo* [56]. Nesta ferramenta é possível realizar a gestão de todo o fluxo do processo de desenvolvimento e também monitorização de ambientes produtivos, tal como é possível verificar na Figura 27.

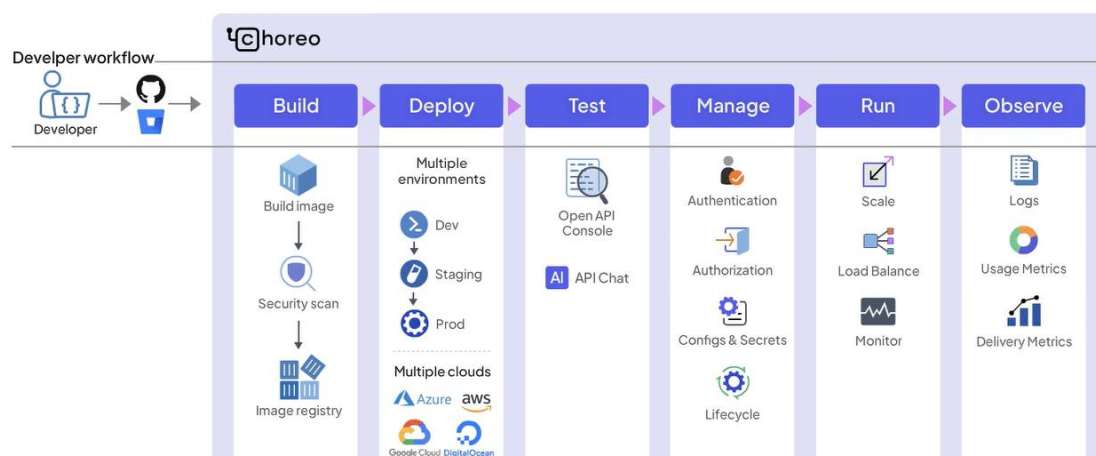


Figura 27: Funcionalidades do software *Choreo*

No *Choreo* é possível a integração com repositórios externos (*GitHub* ou *Bitbucket*), com bases de dados externas ou até a criação de bases de dados (gratuitas por um tempo limitado).

Uma vez que a base de dados já tinha sido criada e, para além disso, o facto de ter apenas um tempo limitado gratuito, fez com que se optasse por incluir no *Choreo* a base de dados externa, criando uma configuração específica para essa comunicação. Deste modo, como recursos específicos do *Choreo*, usou-se apenas a compilação e implantação de código.

4.1.7. Controlo de tarefas

No início do projeto tinha sido pensada a criação de um projeto na ferramenta *Jira* [58] para ajudar no controlo de tarefas a realizar. Apesar desta ferramenta ser uma ferramenta muito poderosa no sentido em que permite, para além do controlo das tarefas e o estado em que cada uma está no processo de desenvolvimento, permite ainda a ligação de cada uma dessas tarefas a *commits* no repositório e permite a ligação com outras ferramentas de documentação. Para além disso, permite a extração de diversos tipos de métricas que são particularmente importantes no desenvolvimento de *software* em equipas.

Neste projeto em específico, entendeu-se que uma ferramenta tão vasta e completa não seria necessária, uma vez que se trata apenas de um elemento a realizar todo o projeto.

Assim, optou-se por uma ferramenta mais simples, *Trello* [4], que funciona praticamente da mesma maneira, embora não permita a extração de métricas de uma maneira simples como o *Jira*, como por exemplo, o *burndown chart*.

Assim, foi possível usar esta ferramenta de modo a poder planejar as várias tarefas do projeto e ir acompanhando o seu estado.

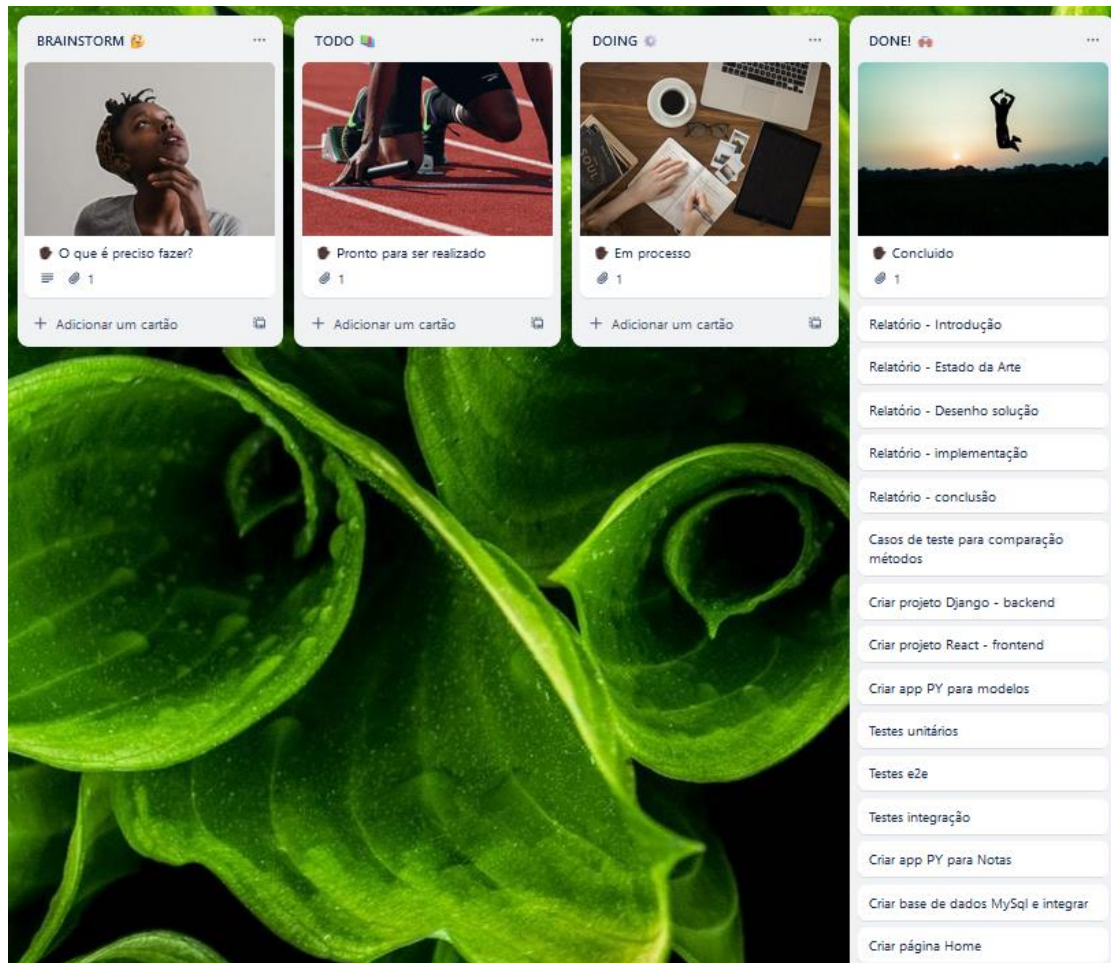


Figura 28: Trello – controlo de tarefas

4.2 Testes

No desenvolvimento de uma aplicação são sempre necessários diferentes tipos de testes unitários: os testes unitários para verificação e validação dos cálculos e registos realizados no *backend*, os testes verificação e validação dos pedidos de *frontend* realizados ao *backend* e ainda os testes end 2 end, ou seja, a verificação do uso da página no browser que gera os pedidos corretos no *frontend*.

Os testes de *backend* prendem-se com a verificação da criação do utilizador (com utilizador e respetiva password) e com a criação da nota (com conteúdo, URL da imagem, método escolhido e utilizador que a criou). Ao correr o comando de testes de Python no *backend*, obteve-se então a seguinte resposta:

```
System check identified no issues (0 silenced).
CreateUserViewTest_CreatedUserNOK: PASS
.CreateUserViewTest_CreatedUserOK: PASS
.CreateUserViewTest_PassNOK: PASS
.CreateUserViewTest_PassOK: PASS
.CreateUserViewTest_UserOK: PASS
.NoteModelTest_AuthorNOK: PASS
.NoteModelTest_AuthorOK: PASS
.NoteModelTest_ContentNOK: PASS
.NoteModelTest_ContentOK: PASS
.NoteModelTest_ModelNOK: PASS
.NoteModelTest_ModelOK: PASS
.NoteModelTest_UrINOK: PASS
.
-----
Ran 12 tests in 3.441s

OK
```

Figura 29: Testes unitários de domínio

Convém referir que foram realizados testes em que o resultado é positivo e também em que o resultado é negativo, como a criação de uma nota sem ter utilizador associado, por exemplo. Ir-se-á agora detalhar um dos testes de cada tabela da base de dados, um relativo aos utilizadores e um teste relativo às notas.

Em relação ao utilizador, como se pode ver na figura abaixo, primeiro define-se o nome de utilizador e a password, de seguida realiza-se um pedido POST à base de dados onde esses dados são registados e finalmente faz-se uma verificação através do comando `self.assertEqual` que a criação na base de dados foi bem sucedida, ou seja que o código HTTP da resposta ao pedido POST é 201.

```
class CreateUserViewTest_CreatedUserOK(APITestCase):
    def test_create_user(self):
        url = reverse('register')
        data = {
            'username': 'testuser',
            'password': 'testpassword123',
        }

        # Make a POST request to the CreateUserView
        response = self.client.post(url, data, format='json')

        # Assert that the response status is 201 Created
        try:
            self.assertEqual(response.status_code, status.HTTP_201_CREATED)
            self.test_result = 'PASS'
        except AssertionError as e:
            self.test_result = 'FAIL'
            raise e

        print(f"{self.__class__.__name__}: {self.test_result}")
```

Figura 30: Teste unitário de criação de utilizador

Em relação à nota, como se pode ver na figura abaixo, primeiro define-se um autor. Depois, ao executar o comando `Note.objects.create` e detalhando os vários campos necessários, o conteúdo, o URL da imagem, o modelo escolhido e o autor escolhido, a nota é gravada na base de dados. De seguida realiza-se um pedido GET à base de dados pelo id da nota criada e finalmente faz-se uma verificação através do comando `self.assertEqual` que a conteúdo da nota é o mesmo que foi indicado originalmente. Se assim for, o resultado do teste unitário é PASS.

```
class NoteModelTest_ContentOK(TestCase):
    def setUp(self):
        self.user = User.objects.create_user(username='testuser', password='testpassword')

    def test_create_note(self):
        note = Note.objects.create(
            content="This is a test note",
            imageURL="http://example.com/image.jpg",
            model="TestModel",
            author=self.user
        )

        # Retrieve the note from the database
        retrieved_note = Note.objects.get(id=note.id)

        # Assert that the note's fields are correct
        try:
            self.assertEqual(retrieved_note.content, "This is a test note")
            self.test_result = 'PASS'
        except AssertionError as e:
            self.test_result = 'FAIL'
            raise e

        print(f"{self.__class__.__name__}: {self.test_result}")
```

Figura 31: Teste unitário de criação de nota

Os testes de *frontend* necessários neste projeto têm a ver com o registo/login do utilizador e também com o pedido ao *backend* de criação do mapa de saliências de imagens ou vídeos.

Para tal, a função que se vai usar para validação do pedido ao *backend* é o `cy.intercept` que é uma função que vai verificar se o pedido POST foi feito e se o status do resultado é 200, ou seja, que o pedido POST foi concretizado.

```
describe('Home Component with CustomDialog', () => {
  it('shows dialog when "Criar mapa de saliências" button is clicked', () => {
    // Stub the necessary functions
    const handleCloseDialog = cy.stub().as('handleCloseDialogStub')
    const handleShowImage = cy.stub().as('handleShowImageStub')

    // Mount the Home component with the CustomDialog inside
    mount(
      <Home
        notes={[]}
        maps={[]}
        content=""
        title=""
        selectedImage={{ name: 'ambulancia.jpg' }}
        selectedOption="casnet1"
        showDialog={true} // Start with showDialog as true for testing
        onCloseDialog={handleCloseDialog}
        onShowImage={handleShowImage}
        imageName=""
        outputImage={null}
        showEmptyContainer={false}
        videoUrl={null}
        resultImage={null}
      />
    )

    // Simulate loading image
    cy.get('input[type="file"]').attachFile('ambulancia.jpg')

    // Label and button to choose method
    cy.get('#casnet').select('CASNET 2')

    // Check button to create map
    cy.get('[style="text-align: center; margin-top: 20px;"] > button').contains("Criar mapa de saliências").click()

    // Check if the dialog is visible
    cy.get('.custom-dialog').should('be.visible')

    // Click the "Criar mapa de saliências" button
    cy.get('.button-container > :nth-child(1)').contains("Criar mapa de saliências").click()

    // Intercept the POST request
    cy.intercept('POST', 'http://localhost:8000', (req) => {
      req.reply((res) => {
        expect(res.statusCode).to.equal(200)
      })
    }).as('postRequest')
  })
})
```

Figura 32: Teste de validação de criação do mapa de saliências

Finalmente, é necessário a verificação proveniente dos testes e2e, que são os testes que validam que as opções escolhidas pelo utilizador são corretamente enviadas para o *frontend* e o devido tratamento é realizado, seja apenas um roteamento para uma nova página ou o envio de dados para o *backend*.

Seguindo o mesmo raciocínio adotado anteriormente, foram criados diversos casos correspondentes a diferentes opções que o utilizador pode escolher, sendo que o resultado final desses testes foi o que está demonstrado na Figura 33.

(Run Finished)

Spec		Tests	Passing	Failing	Pending	Skipped
✓ home_page.cy.js	00:19	1	1	-	-	-
✓ home_page_full.cy.js	00:05	1	1	-	-	-
✓ home_page_full_note.cy.js	00:07	1	1	-	-	-
✓ home_page_image.cy.js	00:09	1	1	-	-	-
✓ home_page_part_full.cy.js	00:06	1	1	-	-	-
✓ login_user.cy.js	00:04	1	1	-	-	-
✓ logout.cy.js	00:05	1	1	-	-	-
✓ register_user.cy.js	00:05	1	1	-	-	-
✓ All specs passed!	01:04	8	8	-	-	-

Figura 33: Resultado de Testes e2e

Será agora apresentado um teste e2e que representa o caso em que o utilizador decide criar o mapa de saliências a partir de uma imagem, mas que decide não escrever nenhuma nota acerca deste processo.

```
describe('Criação de mapa de saliências sem nota', () => {
  it('passes', () => {
    cy.visit('https://38d24dda-c024-4004-aa5a-497ae648c80a.e1-eu-north-azure.choreoapps.dev/login')

    cy.get('img')

    // Fill in the username and password fields
    cy.get('input[placeholder="Username"]').type('testuser')
    cy.get('input[placeholder="Password"]').type('testpassword123')

    // Submit the form
    cy.get('form').submit()

    // Logout button
    cy.get('a').contains("Logout")

    // Label and button to choose image
    cy.get('[style="text-align: center;"] > label').contains("Escolha uma imagem ou um video para a criação do mapa de saliências:")
    cy.get('[style="text-align: center;"] > button').contains("Escolher Imagem")

    // Simulate Loading image
    cy.get('input[type="file"]').attachFile('ambulancia.jpg')

    // Label and button to choose method
    cy.get('.centered-container > label').contains("Escolha um modelo para a criação do mapa de saliências:")
    cy.get('#casnet').select('CASNET 1')

    // Check containers
    cy.get('#imageContainer')
    cy.get('#emptyContainer')

    // Check button to create map
    cy.get('[style="text-align: center; margin-top: 20px;"] > button').contains("Criar mapa de saliências")
    cy.get('[style="text-align: center; margin-top: 20px;"] > button').click()

    // Check pop-up for map creation
    cy.get('.custom-dialog')
    cy.get('.custom-dialog > h2').contains("Aplicação para uso de Modelos de Atenção")
    cy.get('p').contains("Tem a certeza que pretende criar o mapa de saliências?")
    cy.get('.button-container > :nth-child(1)').contains("Criar mapa de saliências")
    cy.get('.button-container > :nth-child(2)').contains("Cancelar")

    cy.get('.button-container > :nth-child(1)').contains("Criar mapa de saliências").click()

    // Simulate Loading image
    cy.get('input[type="file"]').attachFile('ambulancia.jpg')

    // Check if Loaded image is visible
    cy.get('#emptyContainer img').should('be.visible')
  })
})
```

Figura 34: Testes e2e relativo à criação de um mapa de saliências de uma imagem sem criação de nota associada

Como é possível verificar na Figura 34, inicialmente o teste começa por visitar a página de login e colocar dados de utilizador de teste, caso contrário a criação de mapas de saliências não é permitida. Depois é adicionada uma imagem de teste e escolhido um Modelo de Atenção. Depois é feita a confirmação do pedido a ser enviado ao *backend* e finalmente é verificada a existência de uma imagem no espaço em que o output tem de ser renderizado. Fica assim verificado que o pedido foi feito ao *backend*, que o mesmo foi processado e renderizado no browser a mostrar ao utilizador.

4.3 Avaliação da solução

Depois de implementada a solução, pode-se então utilizar o sistema para obter diferentes mapas de saliências de imagens publicitárias ou outros tipos de imagens e vídeos.

Obtiveram-se diversas imagens publicitárias, fez-se o pedido no sistema para a obtenção dos mapas de saliências de cada um dos 2 Modelos de Atenção implementados e de seguida apresentar-se-ão os resultados obtidos.

Para avaliar e comparar o resultado dos modelos, foi realizada uma comparação visual dos dois mapas de saliência resultantes de cada imagem do *dataset* original. Para além disso, foi ainda realizada uma comparação visual entre a imagem original e os mapas de saliências obtidos por cada um dos modelos.

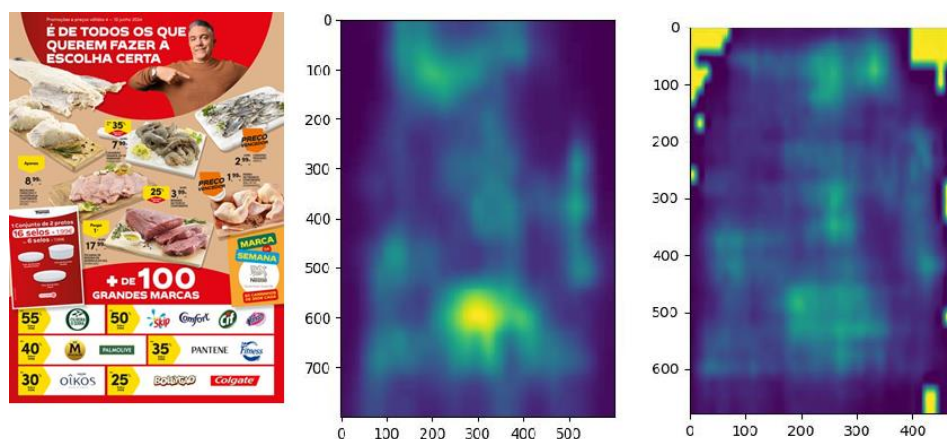


Figura 35: Imagem publicitária e respetivos mapas de saliência obtidos usando o Modelo de Atenção CASNet I e CASNet II

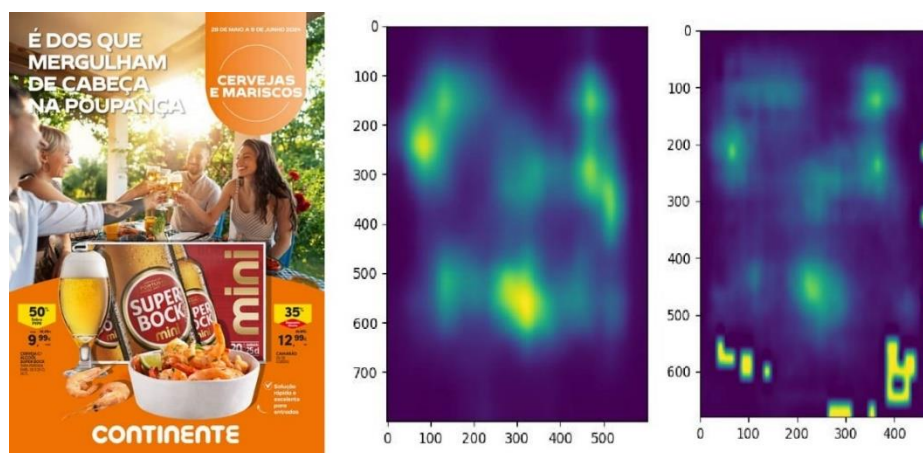


Figura 36: Imagem publicitária e respetivos mapas de saliência obtidos usando o Modelo de Atenção CASNet I e CASNet II

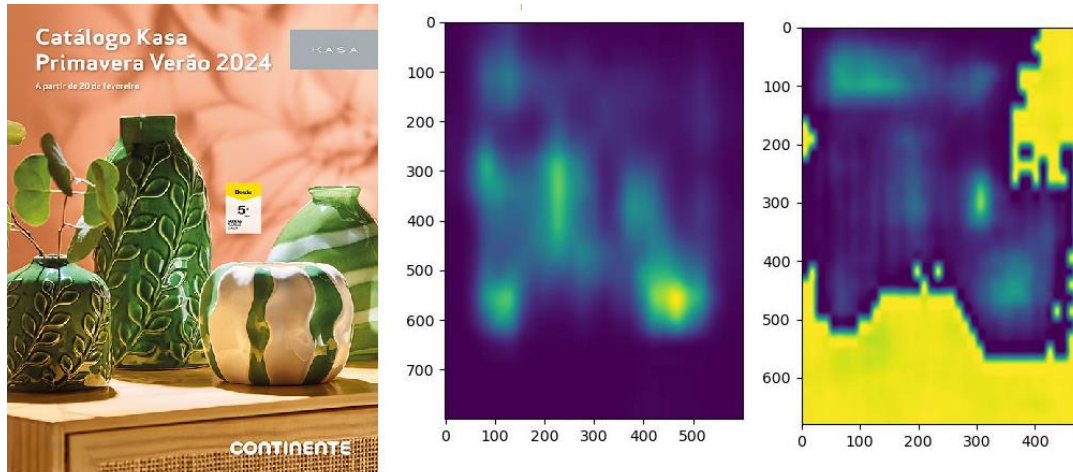


Figura 37: Imagem publicitária e respetivos mapas de saliência obtidos usando o Modelo de Atenção CASNet I e CASNet II

Da análise das 3 imagens acima (Figuras 35, Figura 36 e Figura 37), sendo folhetos da mesma marca, pode-se verificar que os mapas de saliências obtidos pelos 2 Modelos de Atenção são bastante diferentes. Tal comportamento era esperado, uma vez que os algoritmos são diferentes e o treino desses modelos foi também diferente. Optou-se por escolher 3 imagens publicitárias distintas, uma com bastante informação – muitos artigos e respetivos preços –, outra com alguma informação, mas bastante mais simples que a primeira e outra praticamente só com imagens e quase sem informação de preços. A conclusão empírica que se pode tirar da análise destes 3 folhetos é que o Modelo de Atenção CASNet I é o que produz resultados empiricamente mais consistentes e que a imagem da Figura 36 é a que tem resultados mais semelhantes entre os 2 Modelos de Atenção.

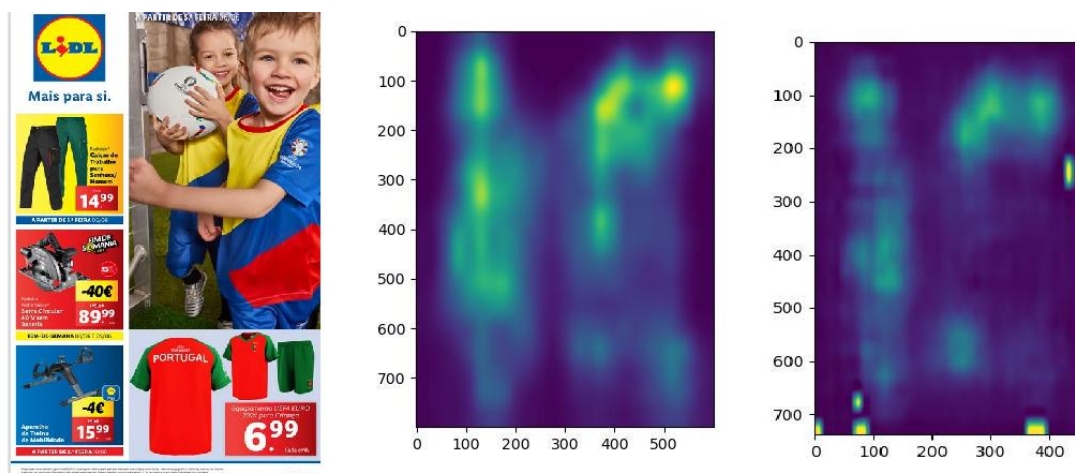


Figura 38: Imagem publicitária e respetivos mapas de saliência obtidos usando o Modelo de Atenção CASNet I e CASNet II

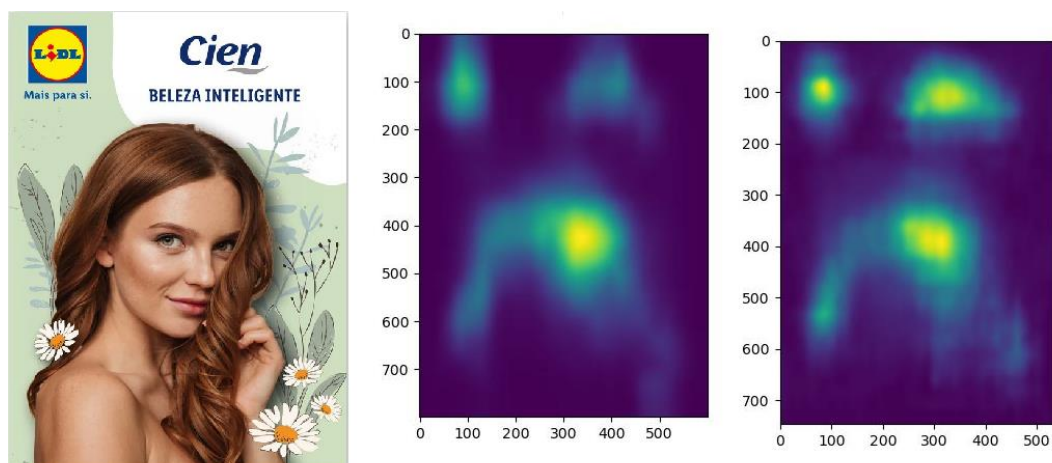


Figura 39: Imagem publicitária e respectivos mapas de saliência obtidos usando o Modelo de Atenção CASNet I e CASNet II

Realizou-se outro estudo comparativo dos mapas de saliência obtidos para imagens publicitárias de outra marca. No presente caso, voltou-se a seguir a metodologia de comparar um folheto publicitário com diversa informação, artigos e preços com outro folheto sem nenhuma informação, onde apenas tem a imagem de uma pessoa e a marca. Em relação aos folhetos desta marca, pode-se concluir que os mapas de saliências obtidos são muito similares entre eles para cada folheto. Isto indicia que são imagens que não são influenciadas pelo componente extra de sentimento visual presente no Modelo de Atenção CASNet II.

Por outro lado, se for analisada com detalhe a Figura 38, podemos ver que o objetivo da marca ao publicitar 4 produtos diferentes não tem o resultado pretendido, uma vez que os mapas de saliências não geram pixels salientes nessas regiões da imagem. Este é um caso óbvio em que o sistema desenvolvido neste projeto poderia trazer vantagens à empresa em questão elaborando o folheto publicitário de outra forma e assim conseguindo obter a atenção do utilizador na zona pretendida.

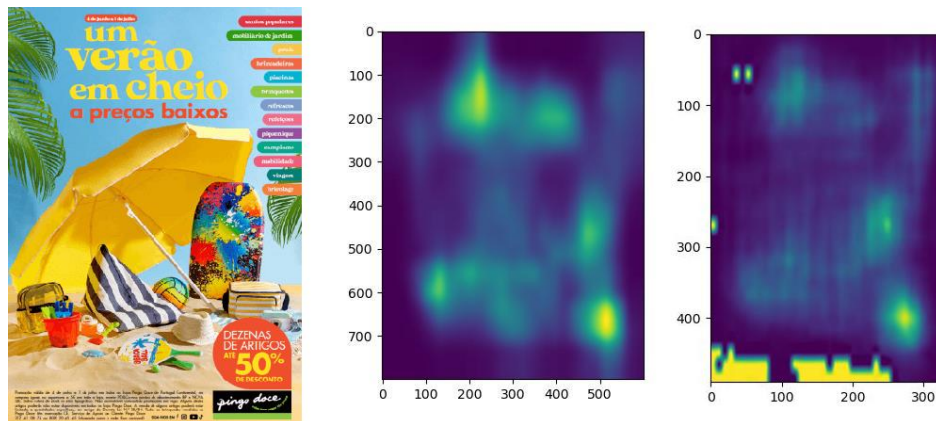


Figura 40: Imagem publicitária e respectivos mapas de saliência obtidos usando o Modelo de Atenção CASNet I e CASNet II

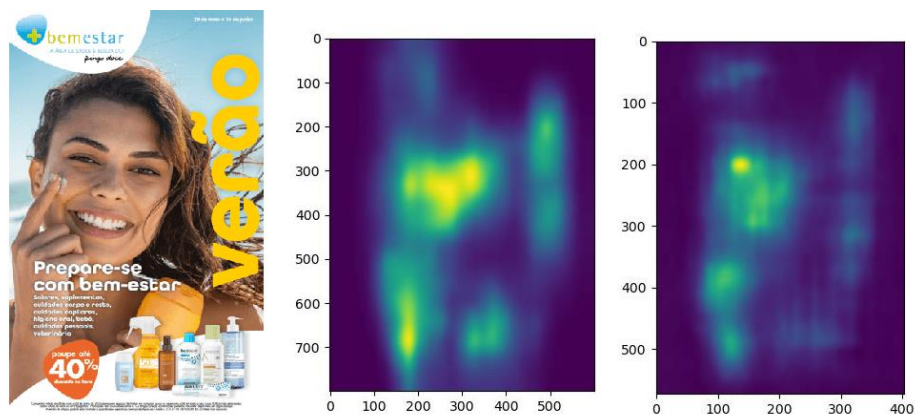


Figura 41: Imagem publicitária e respectivos mapas de saliência obtidos usando o Modelo de Atenção CASNet I e CASNet II

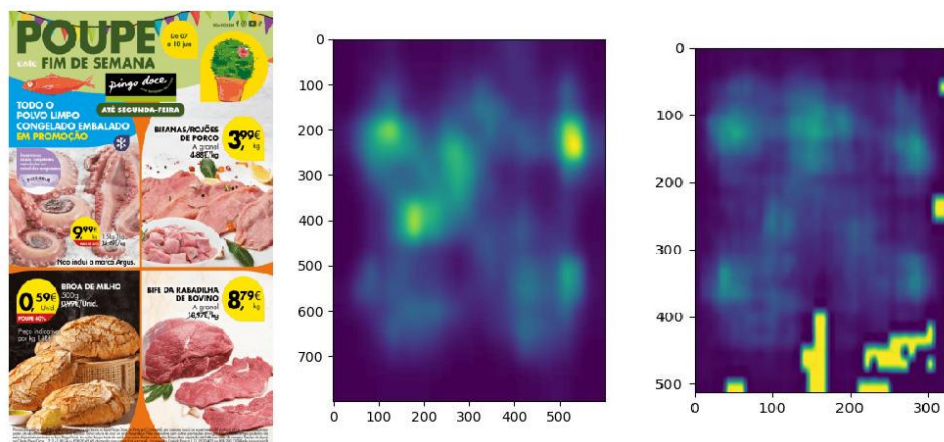


Figura 42: Imagem publicitária e respectivos mapas de saliência obtidos usando o Modelo de Atenção CASNet I e CASNet II

Analizou-se ainda uma terceira marca, igualmente usando 3 folhetos com diferente quantidade de informação em cada um deles. Neste caso em específico o folheto da Figura 41 foi o que obteve resultados mais similares entre os Modelos de Atenção CASNet I e CASNet II.

Nos outros 2 folhetos desta marca e também noutros folhetos publicitários das outras marcas analisadas, existem áreas que o Modelo de Atenção CASNet II reconhece diferenciadamente em relação ao Modelo CASNet I. Essas áreas parecem corresponder a áreas onde existe texto com letras de tamanho reduzido.

Em relação à análise de vídeos, esta comparação é mais complicada do que a comparação de imagens. No entanto, para ser possível essa comparação de modo expedito, decidiu-se escolher um vídeo de tempo reduzido para poder comparar o resultado de 2 Modelos de Atenção.

Na Figura 43, podem se visualizar 8 *frames* consecutivos do vídeo escolhido de um mundo virtual, bem como os mapas de saliências desses frames, usando os Modelos de Atenção CASNet I e CASNet II. Na comparação dos *frames* originais com o resultado da previsão dos 2 métodos, verifica-se que os resultados obtidos são semelhantes. Deste modo, pode-se concluir que neste excerto de vídeo não existem áreas nos *frames* que influenciadas pelo componente extra de sentimento visual presente no Modelo de Atenção CASNet II.

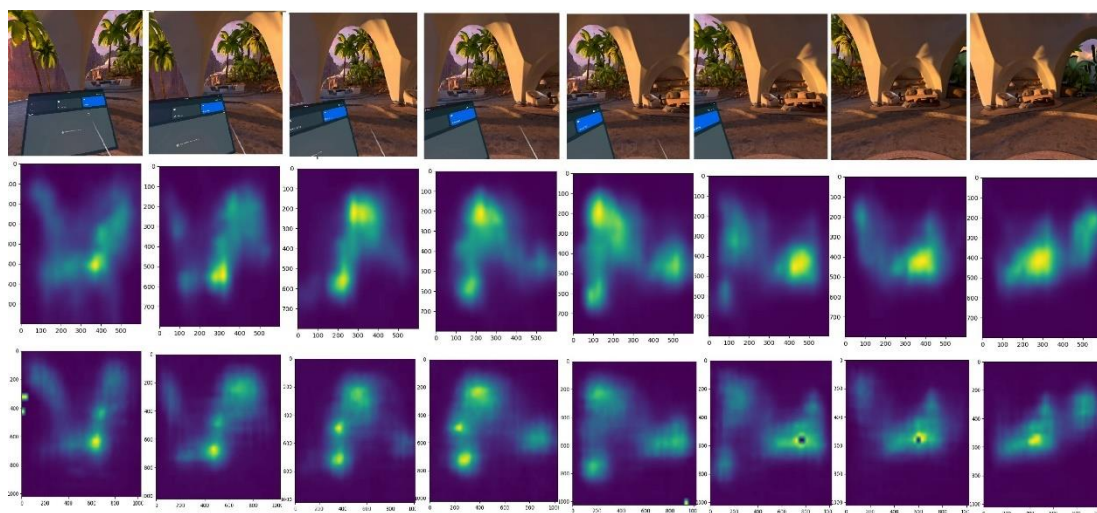


Figura 43: Vídeo do mundo de realidade virtual e respetivos mapas de saliência obtidos usando o Modelo de Atenção CASNet I e CASNet II

Globalmente, os resultados obtidos pelos 2 Modelos de Atenção são bastante interessantes e vão de encontro aos resultados esperados. De facto, com a criação de um sistema de simples utilização para obtenção de mapas de saliências, pretendia-se que a análise comparativa de diferentes métodos fosse simplificada e, de uma maneira geral, mais expedita. Esse objetivo foi amplamente conseguido, a obtenção de mapas de saliências é simples com o uso do sistema desenvolvido.

Em relação à comparação dos métodos propriamente dita, esta comparação seria um pouco genérica e talvez até dúbia se fosse questionada a veracidade dos mapas de saliência obtidos. Para eliminar essa questão, usou-se uma comparação direta dos resultados dos 2 métodos e não de cada método versus a imagem original. Nessa comparação foi possível verificar alguns resultados consistentes, como por exemplo a verificação que zonas com letras de tamanho reduzido são facilmente distinguíveis nos mapas de saliências, independentemente do método escolhido.

Para além dos resultados obtidos pelos dois Modelos de Atenção, foi ainda realizado um estudo relativo ao tempo de processamento de cada um dos Modelos de Atenção, uma vez que era conhecido que este era um fator crucial para o bom desempenho do sistema. Assim, a nível de desempenho, sendo este um requisito não funcional especialmente importante, fez-se também uma análise do tempo de resposta do sistema na obtenção de cada mapa de saliências de cada método. Na tabela abaixo é possível verificar os resultados individuais de cada processamento, bem como se pode aferir que o método CASNet I é, em média 2% mais rápido que o método CASNet II. É possível ainda verificar que o processamento de cada imagem demora cerca de 24seg.

Tabela 12 Comparação de tempo de processamento dos métodos CASNet I e CASNet II

	Tempo de processamento (s)	
	CasNET1	CasNET2
Continente 1	23,07	24,46
Continente 2	24,33	24,96
Continente 3	23,73	25,36
Lidl 1	24,42	25,44
Lidl 2	24,00	24,21
Pingo Doce 1	24,90	23,83
Pingo Doce 2	23,72	23,79
Pingo Doce 3	25,05	24,29
Média	24,15	24,54

Por um lado, os resultados obtidos em termos de desempenho não são um problema, tendo em conta que se trata de uma análise efetuada no âmbito de métodos de aprendizagem profunda, pelo que os utilizadores do sistema estarão sensibilizados para o elevado tempo de processamento nestas análises.

No entanto, se realizarmos a mesma análise a vídeos, vídeos estes que podem ter facilmente dezenas ou centenas de *frames*, passaremos a ter um tempo de processamento de horas ou até dias. Nessa altura, este desempenho passaria a ser uma variável crítica, tal como originalmente era expectável.

4.4 Sumário

Neste capítulo é descrita a implementação da solução, tendo em conta o desenho previsto no capítulo anterior. Foram assim descritos em detalhe o modo como cada um dos 4 componentes *User Interface*, *Webservice*, Modelos de Atenção e a Base de Dados foram implementados fazendo recurso de *print screens* dos métodos mais importantes.

Em seguida foi ainda concretizada o uso de ferramentas auxiliares ao projeto, seja para controlo de versões, criação de *pipeline* ou ainda controlo de tarefas a executar.

Posteriormente, foram descritos os diversos testes unitários realizados, sejam os testes de verificação e validação dos cálculos e registos realizados no *backend*, os testes verificação e validação dos pedidos de *frontend* realizados ao *backend* e ainda os testes end 2 end, ou seja, a verificação do uso da página no browser que gera os pedidos corretos no *frontend*.

Finalmente, é detalhada a avaliação da solução, fazendo recurso de imagens publicitárias e vídeos de realidade virtual fornecidos para concluir sobre as diferenças entre os 2 Modelos de Atenção implementados, o seu desempenho e ainda sobre a comparação entre a imagem original e o resultado de cada um dos modelos.

5 Conclusões

Este projeto tem como principal objetivo o desenvolvimento de um sistema para aplicação de Modelos de Atenção em campanhas de Marketing e realidade virtual. Para a concretização desse objetivo, foi necessária a definição inicial de objetivos, metodologia a implementar no desenvolvimento e a criação do planeamento que cumprisse com o tempo previsto da execução do projeto. De seguida, iniciou-se o estudo aprofundado do tema, comparando o projeto com trabalhos relacionados e ainda se fez um levantamento das tecnologias existentes. Posteriormente, após a definição dos requisitos funcionais e não funcionais, realizou-se o desenho da solução e a implementação da arquitetura prevista.

5.1 Objetivos concretizados

Este projeto teve 3 objetivos, sendo um deles o objetivo principal e dois objetivos secundários.

Tal como referido no capítulo da Introdução o objetivo principal é o desenvolvimento de um sistema para Aplicação de Modelos de Atenção que seja robusto e intuitivo em imagens de campanhas de email marketing e vídeos de realidade virtual. Esse objetivo foi amplamente concretizado. O sistema possui uma utilização fácil, proporcionando resultados precisos.

Para além da concretização do objetivo principal, é importante referir que foi criado também um espaço adicional para cada utilizador escrever comentários acerca de cada imagem processada. Esta implementação não era um objetivo do projeto, mas ao longo do desenvolvimento foi possível perceber que esta funcionalidade era algo que os utilizadores frequentes do sistema iriam rapidamente precisar, daí que esta primeira versão do sistema já inclui esta possibilidade.

Subjacente ao desenvolvimento do sistema está uma validação experimental da sua eficiência e eficácia. Neste objetivo secundário, foram realizados diversos testes experimentais para avaliar a eficácia e precisão do sistema desenvolvido, que incluíram comparações entre os mapas de saliência gerados pelo sistema e avaliações humanas. Foram ainda realizados testes de desempenho do sistema que permitiram aferir e comprovar a eficiência do sistema, uma vez que qualquer imagem é processada em menos de metade do tempo previsto no requisito não funcional.

Um último objetivo é a implementação de funcionalidades de pré-processamento de dados para garantir a qualidade e consistência dos inputs fornecidos ao sistema. Este objetivo não foi executado, uma vez que os Modelos de Atenção implementados já têm incorporados dados de treino, pelo que o pré-processamento de dados não foi necessário implementar.

Além disso, é importante destacar que o planeamento inicial do projeto foi cumprido. As fases foram planeadas e executadas dentro dos prazos estipulados, permitindo uma gestão adequada do tempo.

5.2 Limitações e trabalho futuro

Este projeto foi bastante complexo de implementar, essencialmente no que diz respeito às especificidades de implementação dos Modelos de Atenção. Uma limitação futura prende-se com a evolução do sistema. Sendo verdade que o sistema desenvolvido está preparado para uma evolução simples, por exemplo com a inclusão de um novo Modelo de Atenção, essa evolução traduzir-se-á num sistema mais complexo e, conseqüentemente, com menor desempenho. Assim, para ultrapassar essa limitação, o sistema deveria ser incorporado numa rede de servidores pagos, mas com desempenho muito superior ao implementado no projeto.

Por outro lado, no projeto em questão não foram realizadas todas as melhorias de desempenho para vídeos, uma vez que não era esse o âmbito do projeto. O código dos modelos praticamente não foi editado e existem melhorias que poderiam ser implementadas para um processamento mais rápido.

Em relação às tabelas da base de dados criadas automaticamente com o projeto Django, também poderiam ser melhoradas, uma vez que existem diversos campos que não fazem sentido para este projeto e que poderiam ser eliminadas o que permitiria uma melhor escalabilidade.

Finalmente, em relação à funcionalidade extra das notas relativas ao processamento de cada imagem que o utilizador tem a opção de criar, para utilizadores frequentes o número de notas poderá crescer bastante, pelo que poderia ser criada uma página específica para visualização e gestão dessas notas.

5.3 Apreciação final

O projeto desenvolvido foi realizado com sucesso, tendo sido cumpridos todos os objetivos propostos, sendo ainda incluídas funcionalidades extras ao longo do desenvolvimento.

Considero que o facto de ter sido exigido o uso de uma tecnologia desconhecida à partida, Python, foi extremamente importante. Tendo essa exigência em mente, decidi aproveitar este projeto individual para ganhar competências noutras tecnologias desconhecidas. Assim, para além do Python, aprendi a trabalhar com Conda, Jupyter Lab, React. Usando estas tecnologias, foi necessário também obter conhecimento acerca de testes realizados a componentes Python e React.

Considerando a experiência obtida neste projeto, evidencio ainda a aquisição de competências numa área de Aprendizagem Automática que me desperta muito interesse e na qual pretendo continuar a trabalhar futuramente.

Referências

- [1] PHYNHANCAI, “No Title,” *ISEP*.
<https://www2.isep.ipp.pt/isrc/index.php?page=phynhancai>
- [2] W. McKinney, *Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter*, 3rd editio. 2022. [Online]. Available: <https://wesmckinney.com/book/>
- [3] A. Subasi, *Practical Machine Learning for Data Analysis Using Python*. 2020.
- [4] “Trello.” <https://trello.com/> (accessed Jun. 08, 2024).
- [5] D. Halligan, B. & Shah, *Inbound marketing: get found using Google, social media and blogs*. 2010.
- [6] ComunicaWeb, “Inbound Marketing.” <https://comunica-web.com/blog/marketing-digital/inbound-marketing-que-es-que-significa/>
- [7] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*, 2nd ed. MIT Press, 2018.
- [8] “Siri,” *Apple.com*, 2024. <https://www.apple.com/siri/>
- [9] “Alexa,” *Amazon*. <https://alexa.com/> (accessed Jun. 08, 2024).
- [10] C. Staff, “Machine Learning,” 2024. <https://www.coursera.org/articles/types-of-machine-learning>
- [11] M. Rimol, “Understand 3 Key Types of Machine Learning,” 2020. <https://www.gartner.com/smarterwithgartner/understand-3-key-types-of-machine-learning>
- [12] S. Bhatt, “Reinforcement Learning 101,” *Towards Data Science*. <https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292>
- [13] IBM, “What is Supervised Learning.” <https://www.ibm.com/topics/supervised-learning>
- [14] J. Delua, “What is unsupervised learning?,” *IBM*.

- <https://www.ibm.com/blog/supervised-vs-unsupervised-learning/>
- [15] J. Torres, “A Gentle Introduction to Deep Reinforcement Learning,” *Towards Data Science*. <https://towardsdatascience.com/drl-01-a-gentle-introduction-to-deep-reinforcement-learning-405b79866bf4>
 - [16] DeepAi, “Understanding Attention Models in Deep Learning.” <https://deepai.org/machine-learning-glossary-and-terms/attention-models>
 - [17] I. E. Team, “Attention Model: Definition and When to use One.” <https://www.indeed.com/career-advice/career-development/attention-model>
 - [18] A. Gharahbagh, V. Hajihashemi, M. Ferreira, J. Machado, and J. Tavares, “Hybrid time-spatial video saliency detection method to enhance human action recognition systems,” *Multimed. Tools Appl.*, vol. 83, no. 14, 2024, [Online]. Available: <https://link.springer.com/article/10.1007/s11042-024-18126-x>
 - [19] R. Marois and J. Ivanoff, “Capacity limits of information processing in the brain,” *Trends Cogn. Sci.*, vol. 9, no. 6, pp. 296–305, 2005.
 - [20] S. Fan, Z. Shen, M. Jiang, B. Koenig, and M. Kankanhalli, “Emotional Attention: From Eye Tracking to Computational Modeling,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 1682–1699, 2023, [Online]. Available: https://ncrypt.comp.nus.edu.sg/site/ncrypt-top/sentiment/pdf/Emotional_attention_From_eye_tracking_to_computational_modeling.pdf
 - [21] S. Fan *et al.*, “Emotional Attention: A Study of Image Sentiment and Visual Attention,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
 - [22] Z. Bylinskii, T. Judd, A. Oliva, A. Torralba, and F. Durand, “What do different evaluation metrics tell us about saliency models?,” *arXiv*, 2016.
 - [23] B. W. Tatler, R. J. Baddeley, and I. D. Gilchrist, “Visual correlates of fixation selection: Effects of scale and time,” *Vis. Res.*, vol. 45, no. 5, pp. 643–659, 2005.
 - [24] “TIOBE Index.” <https://www.tiobe.com/tiobe-index/> (accessed Apr. 14, 2024).
 - [25] “PYPL Index.” <https://pypl.github.io/PYPL.html> (accessed Apr. 14, 2024).
 - [26] H. Bhasin, *Python Basics - A Self-Teaching Introduction*. 2019.

- [27] “Python.org.” <https://www.python.org/>
- [28] “Django - the web framework for perfectionists with deadlines.” <https://www.djangoproject.com/start/overview> (accessed Jun. 08, 2024).
- [29] “pathlib — Object-oriented filesystem paths,” *Python.org*. <https://docs.python.org/3/library/pathlib.html> (accessed Jun. 08, 2024).
- [30] “datetime — Basic date and time types,” *Python.org*. <https://docs.python.org/3/library/datetime.html> (accessed Jun. 08, 2024).
- [31] “django-cors-headers 4.3.1,” *pypi.org*. <https://pypi.org/project/django-cors-headers/> (accessed Jun. 08, 2024).
- [32] “Django Rest Framework.” <https://www.django-rest-framework.org/> (accessed Jun. 08, 2024).
- [33] “Conda,” *Conda.io*. <https://conda.io/projects/conda/en/latest/user-guide/getting-started.html> (accessed May 01, 2024).
- [34] “React - Quick start.” <https://react.dev/learn>
- [35] “Vite - getting started.” <https://vitejs.dev/guide/> (accessed Apr. 08, 2024).
- [36] “react - axios,” *NPM*. <https://www.npmjs.com/package/react-axios> (accessed Jun. 08, 2024).
- [37] “React JWT-decode,” *NPM*. <https://www.npmjs.com/package/jwt-decode> (accessed Jun. 08, 2024).
- [38] “MySQLClient,” *Anaconda.org*. <https://anaconda.org/main/mysqlclient> (accessed Jun. 08, 2024).
- [39] “<http://mysqlfreenethosting.net/>.” <http://mysqlfreenethosting.net/> (accessed Apr. 08, 2024).
- [40] “O que é o HTML?,” *OVH Cloud*. <https://www.ovhcloud.com/pt/learn/what-is-html/> (accessed May 08, 2024).
- [41] “FURPS+.” <https://qualidadebr.wordpress.com/2008/07/10/furps/>
- [42] A. Lamsweerde, *Requirements Engineering: From System Goals to UML Models to*

- Software Specifications*. Wiley, 2009.
- [43] R. Pressman and MaximB., *Engenharia de Software*. Mc Graw Hill, 2011.
 - [44] “BPMN - o que é,” *Miro.com*. <https://miro.com/pt/diagrama/o-que-e-bpmn/> (accessed Jun. 08, 2024).
 - [45] “Jupyter lab,” *jupyter.org*. <https://jupyter.org/> (accessed Jun. 08, 2024).
 - [46] “PyJWT.” <https://pyjwt.readthedocs.io/en/stable/> (accessed Jun. 08, 2024).
 - [47] “JWT.” <https://www.youtube.com/watch?v=c-QsfbznSXI> (accessed Apr. 08, 2024).
 - [48] “MoviePy,” *pypi.org*. <https://pypi.org/project/moviepy/> (accessed Jun. 08, 2024).
 - [49] “MovieWriter,” *OpenCV*. https://docs.opencv.org/4.x/dd/d9e/classcv_1_1VideoWriter.html (accessed Jun. 08, 2024).
 - [50] “Next.js or Vite.js: Which Framework is Better, and When?,” *Rollbar*, 2023. <https://rollbar.com/blog/nextjs-vs-vitejs/#> (accessed May 08, 2024).
 - [51] “React Router DOM,” *NPM*. <https://www.npmjs.com/package/react-router-dom> (accessed Jun. 08, 2024).
 - [52] “Base de Dados NoSQL – O que é NoSQL?,” *Azure*. <https://azure.microsoft.com/pt-pt/resources/cloud-computing-dictionary/what-is-nosql-database> (accessed May 08, 2024).
 - [53] “MongoDB.” <https://www.mongodb.com/> (accessed May 08, 2024).
 - [54] “GitHub.” <https://github.com/>
 - [55] “BitBucket.” <https://bitbucket.org/> (accessed Jun. 08, 2024).
 - [56] “Choreo.” <https://choreo.dev/> (accessed Jun. 08, 2024).
 - [57] “What’s a CI/CD pipeline?,” 2022. <https://www.redhat.com/en/topics/devops/what-cicd-pipeline> (accessed Jun. 05, 2024).
 - [58] “Jira.” <https://www.atlassian.com/software/jira> (accessed Jun. 08, 2024).
 - [59] “What is Saliency Map?” <https://www.geeksforgeeks.org/what-is-saliency-map/>

(accessed Jun. 08, 2024).

- [60] V. Peçanha, “O que é Marketing Digital?,” *rockcontent*. <https://rockcontent.com/br/blog/marketing-digital/> (accessed Jun. 15, 2024).
- [61] A. Siqueira, “O que é Inbound Marketing e como criar as melhores estratégias na prática,” *Resultados Digitais*. <https://www.rdstation.com/blog/marketing/o-que-e-inbound-marketing/> (accessed Jun. 08, 2024).