INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Plataforma para Recomendação de Artes Performativas

JOÃO PEDRO LOURENÇO DOMINGUES
Outubro de 2021

POLITÉCNICO
DO PORTO

Plataforma para Recomendação de Artes Performativas

João Domingues

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

Orientador: Prof. João Paulo Pereira

Júri:
Presidente:

Vogais:

Porto, 10 de outubro de 2021

Dedicatória

Pelas cantorias alegres e persistentes que emanavam
paixão e serviram de inspiração.
Pela característica e constante preocupação em cultivar
e regar a felicidade daqueles que o rodeavam.
Pela falta física que faz, compensada pela sempre
crescente presença criada pela lembrança.
Para ti, 'Zé Manel.

Resumo

A presente dissertação detalha o processo de recolha e análise de dados referentes ao desenvolvimento de uma plataforma *online*, no âmbito da unidade curricular Tese / Dissertação / Estágio do Mestrado em Engenharia de Informática, com área de especialização em Engenharia de Software, do Departamento de Engenharia Informática do Instituto Superior de Engenharia do Porto.

Ao longo do documento são descritos os diversos passos tomados no sentido do desenvolvimento de uma plataforma *online* de apoio à comunidade que constitui a indústria da música, nomeadamente, na sua vertente de retalho. Aqui, estão reunidas as fases de análise do problema, definição de objetivos, análise do mercado e identificação de lacunas nas soluções oferecidas, desenho e implementação de uma solução e respetiva avaliação.

Mais que uma simples rede social, a plataforma resultante deste projeto pretende oferecer ferramentas que permitam aos intervenientes da indústria da música aumentar a sua rede de contactos e tomar decisões informadas que permitam o avançar das suas carreiras. Num período que não discrimina profissionais de amadores no que toca a introduzir novas adversidades a esta indústria, o desenvolvimento deste projeto é um esforço no sentido de contribuir para que estas pessoas possam continuar a pintar o nosso mundo com a sua sonoridade.

Com a conclusão do projeto, é possível verificar que os objetivos traçados foram alcançados com sucesso e a plataforma resultante representa uma base de trabalho para um futuro sólido.

Palavras-chave: *Web Development*, Indústria da Música, *Networking*, Sistema de Recomendações, Sistema de Apoio à Decisão

Abstract

This dissertation details all the steps taken in the process of collecting and analyzing data related to the development of an online platform. This project is inherent to the Tese / Dissertação / Estágio course of the Masters in Computer Engineering, specialization area in Software Engineering, of the Instituto Superior de Engenharia do Porto.

This document describes the various steps taken towards the development of an online platform to support the community that constitutes the music industry. Here, the steps of problem analysis, definition of objectives, market analysis and identification of gaps in the solutions offered, design and implementation of a solution and it's evaluation are brought together.

More than a simple social network, the resulting platform aims at offering tools that allow music industry players to increase their networking abilities and make informed decisions that will allow them to advance their careers. In a period that does not discriminate professionals from amateurs when it comes to introducing new adversities to this industry, the development of this project is an effort to contribute so that these people can continue to paint our world with their sound.

With the completion of the project, it is possible to verify that the goals set were successfully achieved and the resulting platform represents a working basis for a solid future.

Agradecimentos

Agradeço à família pelo apoio incondicionalmente incondicional e que, nem sempre percebendo exatamente o que faço com um computador, davam o ânimo necessário para acreditar que tudo ia correr sempre pelo melhor.

Agradeço aos amigos - todos eles escolhidos a dedo - pela presença e paciência em todas as vertentes da vida, não só nesta aventura, como em todas as aventuras que lhe foram paralelas e perpendiculares. Às longas conversas, saídas à noite, noites sem dormir, loucuras sem fim, trabalhos em conjunto e histórias e estórias de vida que vamos construindo em conjunto.

Agradeço aos docentes com quem me cruzei durante o meu percurso académico, com especial ênfase ao professor João Paulo Pereira, pelo apoio prestado na execução deste projeto.

Agradeço ao Nuno Fontes, à Maby e ao Jorge Loura, pela paciência inabalável que permitiu que um simples estudante que era, até à data, um quase desconhecido, ter a oportunidade de os entrevistar.

Por fim, agradeço a todos os meus amigos musicais com os quais me cruzei durante os últimos 18 anos de vida e que, de uma forma ou de outra, serviram de grande inspiração para este projeto.

Conteúdo

Lista de Figuras **xv**

Lista de Tabelas **xvii**

Lista de Acrónimos **xix**

1 Introdução **1**
1.1 Enquadramento . . . 1
1.2 Descrição do Problema . . . 3
1.3 Objetivos . . . 4
1.4 Metodologia . . . 4
1.5 Contributos . . . 5
1.6 Planeamento de Trabalho . . . 6
1.7 Estrutura do relatório . . . 6

2 Contexto e Estado da Arte **9**
2.1 Contexto do Problema . . . 9
2.1.1 Testemunhos Profissionais . . . 9
2.1.2 Área de Negócio . . . 11
Agente - Promotor . . . 12
Agente - Artista . . . 12
Artista - Promotor . . . 13
Eventos . . . 14
A Competitividade do Mercado . . . 14
2.2 Trabalhos relacionados . . . 15
2.2.1 Sumário . . . 15
2.2.2 Contextualização de soluções . . . 16
Redes Sociais . . . 16
Plataformas de Streaming . . . 17
Serviços Online . . . 17
Plataformas Especializadas . . . 18
2.2.3 Comparação de Soluções . . . 18
Público Alvo . . . 19
Tipos Possíveis de Utilizador . . . 19
Funcionalidades . . . 20
2.2.4 Observações Finais . . . 23
2.3 Tecnologias existentes . . . 23
2.3.1 Frameworks . . . 23
Frontend . . . 24
Backend . . . 25
2.3.2 Media Storage Service . . . 26

2.3.3 Gestão do Projeto . . . 27
2.4 Conclusões . . . 27

3 Análise de Requisitos **29**
3.1 Modelo de Domínio . . . 29
3.2 Requisitos . . . 30
3.2.1 Requisitos Funcionais . . . 31
3.2.2 Requisitos Não Funcionais . . . 32
3.2.3 FURPS+ . . . 33
Funcionalidade . . . 33
Usabilidade . . . 34
Fiabilidade . . . 34
Performance . . . 34
Suportabilidade . . . 34
Restrições (+) . . . 34
3.3 Casos de Uso . . . 35
3.4 Conclusões . . . 36

4 Análise de Valor **39**
4.1 Processo de Inovação . . . 39
4.2 New Concept Development . . . 40
4.2.1 Identificação da Oportunidade . . . 40
4.2.2 Análise da Oportunidade . . . 41
Forças . . . 42
Fraquezas . . . 42
Oportunidades . . . 42
Ameaças . . . 42
4.2.3 Geração de Ideias . . . 43
4.2.4 Seleção de Ideias . . . 43
Divisão Hierárquica . . . 44
Definição de Prioridades . . . 45
Validação de Consistência . . . 50
4.3 Valor da Solução . . . 51
4.3.1 Valor . . . 52
4.3.2 Proposta de Valor . . . 52
Business Model Canvas . . . 52
4.3.3 Necessidades do Cliente . . . 54
4.3.4 Quality Function Deployment . . . 54
Necessidades do Cliente . . . 55
Características de Engenharia . . . 56
Escalas de Correlação . . . 56
Concorrentes . . . 57
4.4 Conclusões . . . 59

5 Desenho da Solução **61**
5.1 Arquitetura . . . 61
5.1.1 Frontend e Backend . . . 61
Alternativa A . . . 61
Alternativa B . . . 62

Seleção de Alternativas 62
5.1.2 Backend 62
Alternativa A 62
Alternativa B 63
Seleção de Alternativas 63
Alternativa C 64
5.1.3 Componentes do Backend 65
Alternativa A 65
Alternativa B 66
Seleção de Alternativas 66
5.1.4 Sistema de Mensagens 67
Alternativa A 67
Alternativa B 67
Alternativa C 68
Seleção de Alternativas 69
5.1.5 API Gateway 69
5.2 Casos de Uso 70
5.2.1 Diagrama de Classes 70
5.2.2 Diagrama de Sequência 72
5.3 Conclusões 77

6 Implementação da Solução **79**
6.1 Metodologia de Desenvolvimento 79
6.1.1 Modelo de Dados 79
6.1.2 Escolha de um Caso de Uso a desenvolver 81
6.1.3 Implementação do Backend 82
6.1.4 Implementação do Frontend 86
User Interface 87
Componentes 88
React Hooks 89
Redux 90
6.1.5 Realização de testes às funcionalidades desenvolvidas 92
6.2 Sistema de Recomendação 92
6.2.1 Recomendação por Semelhança 93
6.2.2 Recomendação por Interesse de Negócio 94
6.3 Conclusões 95

7 Avaliação da Solução **97**
7.1 Quantitative Evaluation Framework 97
7.2 Fontes de Informação 100
7.3 Metodologia de Avaliação 100
7.4 Resultados 101
7.4.1 Requisito UL02 102
7.4.2 Requisito UN01 103
7.4.3 Requisito UN02 104
7.4.4 Requisito UN03 105
7.4.5 Requisito UN04 106
7.4.6 Requisito UC01 107
7.4.7 Requisito UC02 107

7.4.8 Resultados Questionário 108
7.5 Conclusões 110

8 Conclusões **111**
8.1 Objetivos concretizados 111
8.2 Limitações e trabalho futuro 112
8.3 Apreciações Finais 113

Bibliografia **115**

A Diagramas de Sequência Completos **121**

B Métricas QEF **123**
B.0.1 Funcionalidade 123
B.0.2 Suportabilidade 125
B.0.3 Usabilidade 125

Lista de Figuras

1.1 Configuração do Setor Cultural e Criativo [1]. 1
1.2 Cadeia de valor de bens e serviços no setor cultural [1]. 2
1.3 Cadeia de *activities* da metodologia Design Science Research (DSR) [13]. . 5
1.4 Proposta de planeamento do Projeto. 6

3.1 Modelo de Domínio. 29
3.2 Diagrama de Casos de Uso. 35

4.1 Fases do processo de inovação - Modelo de Peter Koen [72]. 39
4.2 Análise SWOT da oportunidade identificada. 41
4.3 Árvore Hierárquica Analytic Hierarchy Process (AHP). 45
4.4 Tabela para obtenção do Índice Aleatório (IR). 51
4.5 *Business Model Canvas* associado ao produto a desenvolver. 53
4.6 Escala de relação entre Características de Engenharia do QFD. 56
4.7 Escala de correlação entre Características de Engenharia do QFD. 57
4.8 *Quality Function Deployment.* 58

5.1 Diagrama de Implantação da Alternativa A. 61
5.2 Diagrama de Implantação da Alternativa B. 62
5.3 Diagrama de Implantação da Alternativa B. 63
5.4 Diagrama de Implantação da Alternativa C. 65
5.5 Arquitetura MVC. 65
5.6 Arquitetura Multicamada [93]. 66
5.7 Polling vs. Websockets [96]. 68
5.8 Exemplo de um *MQTT Broker* utilizando arquitetura AMQP [98]. 68
5.9 Inclusão do *Message Broker* na arquitetura desenvolvida até agora. 69
5.10 Inclusão do Application Program Interface (API) *Gateway* na arquitetura desenvolvida. 70
5.11 Classes referentes ao monólito apresentado na imagem 5.10. 71
5.12 Classes referentes ao micro-serviço apresentado na imagem 5.10. 72
5.13 Fragmento do Diagrama de Sequência do UC18 - Criar Projeto. 73
5.14 Fragmento do Diagrama de Sequência do UC18 - Criar Projeto. 74
5.15 Diagrama de Sequência do UC06 - Enviar Mensagem. 76

6.1 Pedido GET que permite aceder às intâncias da classe Projeto armazenadas na base de dados. 81
6.2 Exemplo de pedido POST que permite criar um Projeto. 84
6.3 Home Page da aplicação. 87
6.4 Exemplo de uma das representações possíveis de um Cartão de Perfil. . . . 88
6.5 Lista de Recomendação por Semelhança num perfil de Projeto. 93
6.6 Lista de Recomendação por Interesse na *Dashboard* de um Artista. 95

7.1 Dimensão Usabilidade. 98
7.2 Dimensão Suportabilidade. 98
7.3 Dimensão Funcionalidade. 99
7.4 Respostas ao questionário. 101
7.5 Respostas à questão referente ao Requisito UL02. 103
7.6 Respostas à questão referente ao Requisito UN01. 103
7.7 Respostas à questão referente ao Requisito UN02. 104
7.8 Respostas à questão referente ao Requisito UN03. 105
7.9 Respostas à questão referente ao Requisito UN04. 106
7.10 Respostas à questão referente ao Requisito UC01. 107
7.11 Respostas à questão referente ao Requisito UC02. 108
7.12 Dimensão Usabilidade preenchida com os resultados dos utilizadores. 108
7.13 Dimensão Suportabilidade preenchida com a avaliação do autor. 109
7.14 Dimensão Funcionalidade preenchida com a avaliação do autor. 109

A.1 Diagrama de Sequência do UC18 - Criar Projeto. 122

B.1 Métricas de Avaliação do Fator *Networking*. 123
B.2 Métricas de Avaliação do Fator Utilizador. 123
B.3 Métricas de Avaliação do Fator *Feedback* do Sistema. 123
B.4 Métricas de Avaliação do Fator Artista. 124
B.5 Métricas de Avaliação do Fator Agente. 124
B.6 Métricas de Avaliação do Fator Promotor. 124
B.7 Métricas de Avaliação do Fator Recomendação. 124
B.8 Métricas de Avaliação do Fator Audiovisual. 125
B.9 Métricas de Avaliação do Fator Adaptabilidade. 125
B.10 Métricas de Avaliação do Fator Manutenibilidade. 125
B.11 Métricas de Avaliação do Fator Linguagem. 125
B.12 Métricas de Avaliação do Fator Navegação. 125
B.13 Métricas de Avaliação do Fator Consistência. 126

Lista de Tabelas

2.1 Público Alvo das plataformas alternativas. 19
2.2 Tipos Possíveis de Utilizador das plataformas alternativas. 19
2.3 Funcionalidades Base das plataformas alternativas. 21
2.4 Funcionalidades de *Networking* das plataformas alternativas. 22
2.5 Funcionalidades de Negócio das plataformas alternativas. 22
2.6 Resultados dos testes de *benchmark* de performance. 24
2.7 Classificação e pontuações das *frameworks* atribuídas pela comunidade. . . 25
2.8 Resultados dos testes de *benchmark* de performance. 26
2.9 Propriedades dos planos grátis associados a cada serviço. 27

4.1 Código de letras para Critérios e Soluções. 45
4.2 *The Fundamental Scale* de Saaty [76]. 45
4.3 Prioridades Relativas dos Critérios. 46
4.4 Prioridades Relativas das Soluções - Tempo. 47
4.5 Prioridades Relativas das Soluções - Manutenção. 47
4.6 Prioridades Relativas das Soluções - Dificuldade. 48
4.7 Prioridades Relativas das Soluções - Utilizadores. 49
4.8 Prioridades Relativas das Soluções. 50
4.9 Matriz de Consistência. 50
4.10 Necessidades do Cliente e respetivas Importâncias. 55

Lista de Acrónimos

AHP	Analytic Hierarchy Process.
APEFE	Associação de Promotores de Espetáculos, Festivais e Eventos.
API	Application Program Interface.
CRUD	Create, Read, Update e Delete.
DEI	Departamento de Engenharia Informática.
DSR	Design Science Research.
HTTP	Hypertext Transfer Protocol.
IC	Índice de Consistência.
IR	Índice Aleatório.
ISEP	Instituto Superior de Engenharia do Porto.
MQTT	Message Queueing Telemetry Transport.
MTV	Model, Template e View.
MVC	Model, View e Controller.
MVP	Minimum Viable Product.
OOP	Object Oriented Programming.
QEF	Quantitative Evaluation Framework.
QFD	Quality Function Deployment.
RC	Rácio de Consistência.
SaaS	Software as a Service.
SPA	Single-Page Application.
TCP	Transmition Control Protocol.
UI	User Interface.

Capítulo 1

Introdução

A importância da cultura na sociedade é ímpar, representando uma das maiores pegadas que deixamos como humanidade. Da cultura não nasce a sociedade, mas sim ao contrário e, como consequência, numa realidade em que o acesso a informação é possível com uma facilidade sem precedentes, podemos testemunhar a beleza intrínseca que nasce nas diferenças entre os povos.

Uma pequena parcela do leque cultural é representado pela música, que se insere numa indústria própria - a indústria da música.

1.1 Enquadramento

Em Portugal, o Setor Cultural e Criativo divide cada indústria artística em 3 vertentes: **Indústrias Culturais**, **Actividades Criativas** e **Setor Cultural**. O fator que distingue estas 3 vertentes são os *stakeholder* associados a cada, bem como a propriedade intelectual.

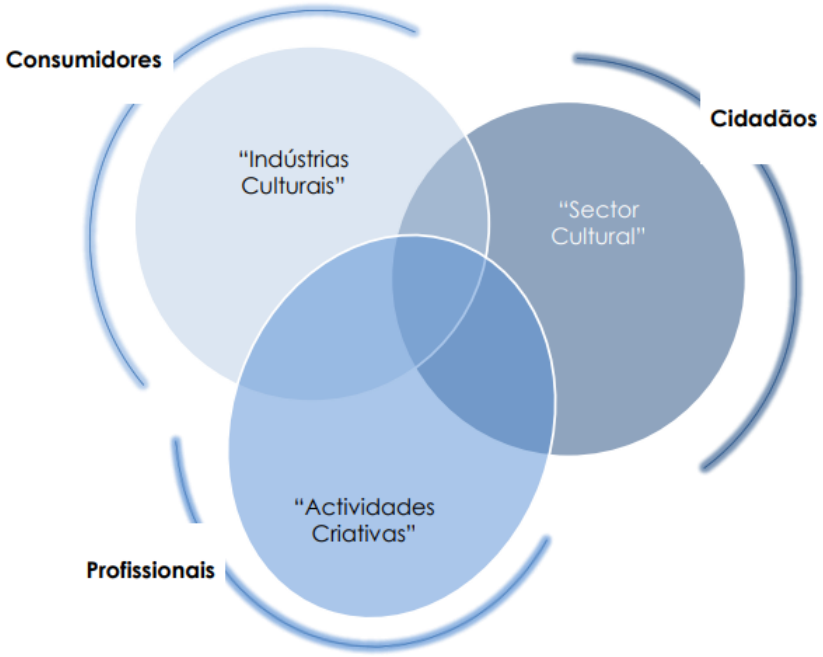


Figura 1.1: Configuração do Setor Cultural e Criativo [1].

Na figura 1.1 é possível perceber que vertente se liga a que *stakeholder*:

- **Actividades Criativas**, que têm como propriedade intelectual as artes de forma genérica [1], têm como *stakeholder* os profissionais, uma vez que se focam na parte de produção;
- **Setor Cultural**, que tem como propriedade intelectual o património artístico [1], tem como *stakeholder* os cidadãos, uma vez que benificiam culturalmente da sua existência;
- **Indústrias Culturais**, que têm como propriedade intelectual as redes de difusão e canais digitais [1], têm como *stakeholder* os consumidores, uma vez que se focam na parte de distribuição;

O processo que liga a génese do produto da indústria da música, por parte dos **profissionais**, ao seu consumo, por parte dos **consumidores**, trata-se de um processo multi-faseado, que pode ser representado por uma cadeia de valor, como representado na figura 1.2.

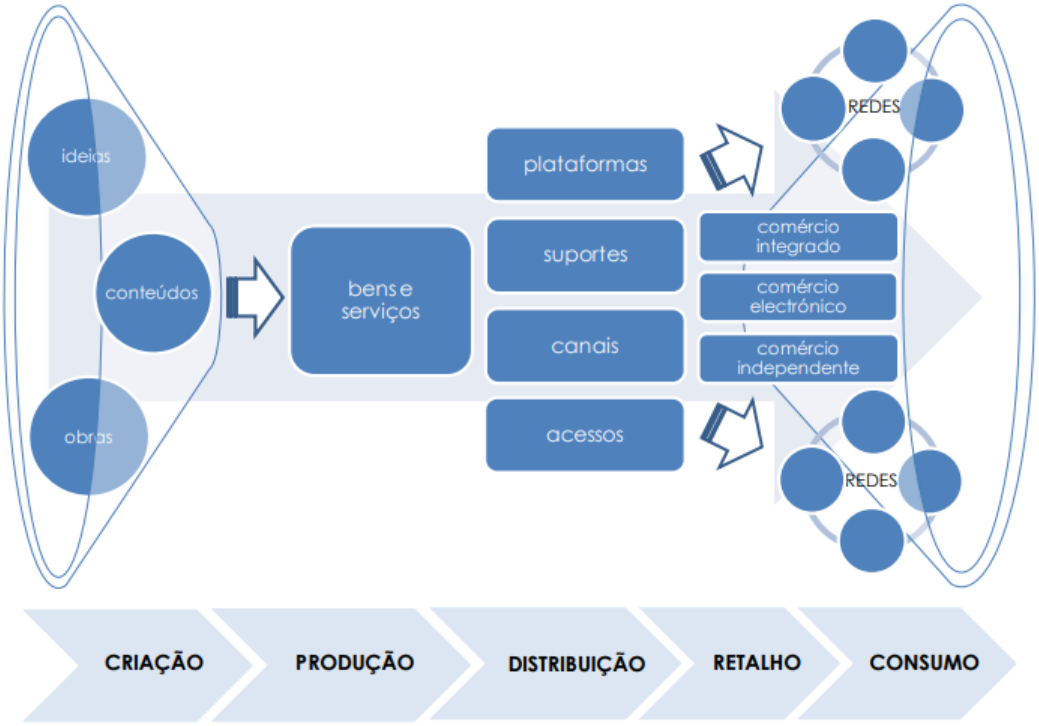


Figura 1.2: Cadeia de valor de bens e serviços no setor cultural [1].

Os mecanismos internos de cada fase funcionam à base de diferentes profissionais que, em conjunto, impulsionam a indústria no seu todo para a frente, satisfazendo as necessidades dos consumidores, respondendo às tendências do mercado musical.

No contexto da presente dissertação, o foco principal será a fase de **retalho** e 3 dos seus principais intervenientes - **artistas**, **agentes** e **promotores**.

- **Artistas** - Profissionais de música envolvidos nas artes performativas. A presente dissertação aborda os artistas olhando para a pessoa que está por de trás da personagem que veste em palco - um membro da sociedade que vive da sua função na indústria da música, dissociando-os de outros artistas ou profissionais. Dar-se-á especial atenção

aos artistas que não tenham um nome que, do ponto de vista da procura de trabalho, seja auto-sustentável (o próprio nome do artista dá-lhe acesso a oportunidades de trabalho, por ser quem é);

- **Agentes** - Profissionais da indústria da música que estabelecem um elo de ligação entre **artistas** e **promotores**. São intermediários que, como ferramenta de trabalho, servem-se da sua influência e rede de contactos;
- **Promotores** - Entidades, de uma forma genérica, que contratam **artistas** de forma a vender o seu produto, num formato de performances ao vivo.

A presente dissertação é o resultado do interesse de contribuir para o bem-estar dos profissionais de uma área de interesse para o autor - a indústria da música - através da idealização de uma plataforma online que forneça novas ferramentas a artistas que, por força dos fatores apresentados acima, estejam a passar por momentos de dificuldade.

Este é um projeto no âmbito da tese de mestrado em Engenharia Informática, no ramo de Engenharia de Software, ao abrigo do *Departamento de Engenharia Informática (DEI)* do *Instituto Superior de Engenharia do Porto (ISEP)*.

1.2 Descrição do Problema

A indústria da música fornece o seu produto - música - sob dois formatos: música ao vivo e música gravada [2].

Dentro da vertente da música gravada, o fenómeno da música digital introduzido na indústria no início do século [3], resultou num drástico declínio nas receitas associadas à venda de música gravada em formato físico. No início da década de 2010, registou-se uma nova tendência na indústria - a maior parcela da receita era gerada não pela vertente da música gravada, mas pela vertente da música ao vivo [4, 5].

Em 2020, como consequência do novo paradigma da *COVID-19*, a vertente de música ao vivo, que até então tinha verificado um aumento de receita estável [5], sofre um golpe sem precedentes [6]. As novas restrições sociais limitam não só o número de espetáculos, como a receita de cada espetáculo individual (derivado da restrição na quantidade de pessoas que compõe o público). Embora haja programas e iniciativas de apoio governamental [7, 8] que ajudem os profissionais do mundo artístico, tratam-se de ajudas que não têm capacidade de tapar a exposição destes profissionais à pobreza e que "[...] além de pequenas, não cobriram a imensidão de trabalhadores do sector"[6].

Num estudo levado a cabo pela *Associação de Promotores de Espetáculos, Festivais e Eventos (APEFE)* no dia 3 de abril de 2020, verificou-se que no período entre 8 de março e 31 de maio do mesmo ano, 24 mil eventos de artes performativas foram cancelados, demonstrando a escala do impacto pandémico [9].

A prosperidade na indústria da música, nomeadamente, no que toca a artistas independentes (que não têm uma equipa responsável pela sua promoção e exposição) de menores dimensões, baseia-se na sua própria rede de contactos. A ponte entre um determinado artista e um promotor é, muitas vezes, criada a partir ou de um contacto mútuo ou de contacto direto [10–12].

Um sistema desta natureza é sujeito às características de um *loop de feedback positivo*, no qual uma maior rede de contactos dá ao artista uma maior exposição no mercado, potenciando o crescimento da mesma a um maior ritmo [11].

Estes factos, quando em conjunto, representam dificuldades de dimensões sem precedentes para pequenos artistas, uma vez que estes se inserem no mesmo mercado que artistas com maior nome no mercado, bem como músicos amadores que também competem pelas mesmas oportunidades e eventos [10].

1.3 Objetivos

Este projeto tem como objetivo principal o desenvolvimento de uma plataforma *web* que crie um ecossistema que englobe três entidades distintas - artistas, agentes e promotores - fornecendo ferramentas e *workflows* desenhados em torno do modo de funcionamento da indústria da música, permitindo aos intervenientes a expansão da sua rede de contactos, potenciando assim o número de espetáculos e oportunidades de trabalho a que artistas tenham acesso.

De forma a alcançar este objetivo, foram traçadas *milestones* intermédias:

- **Análise do funcionamento e estado da indústria para recolha de requisitos** - Levantamento de requisitos junto dos *stakeholders*, de forma a concluir qual o tipo de funcionalidades a ser incluído na aplicação *web*;
- **Análise de soluções existentes no mercado** - Análise do funcionamento de soluções já existentes no mercado e do seu modo de funcionamento, identificando lacunas e/ou pontos de melhoria;
- **Design de uma arquitetura robusta e escalável e respetiva implementação** - Desenhar e implementar a solução, aplicando boas práticas de desenvolvimento de *software*.

1.4 Metodologia

A presente dissertação descreve o processo do desenvolvimento da plataforma *web*, desde a origem da ideia à sua conclusão. A fim de tornar a navegação por este processo mais eficaz e eficiente, foi delineada uma metodologia de trabalho - a metodologia *Design Science Research (DSR)*.

Esta metodologia guia-se por um processo de investigação e *design* de artefactos, orientado a um contexto definido, com a finalidade de abordar uma problemática associada a esse mesmo contexto, tentando criar/melhorar uma solução para atenuar ou resolver essa mesma problemática [13].

A metodologia *DSR* divide-se em 6 passos - *activities* [13]:

1. Identificação do problema e motivação;
2. Definição de objetivos para uma solução;
3. Design e desenvolvimento da solução;
4. Demonstração da solução;
5. Avaliação da solução;

6. Comunicação e exposição do processo e resultados.

A forma como as diferentes *activities* se relacionam está representado na figura 1.3:

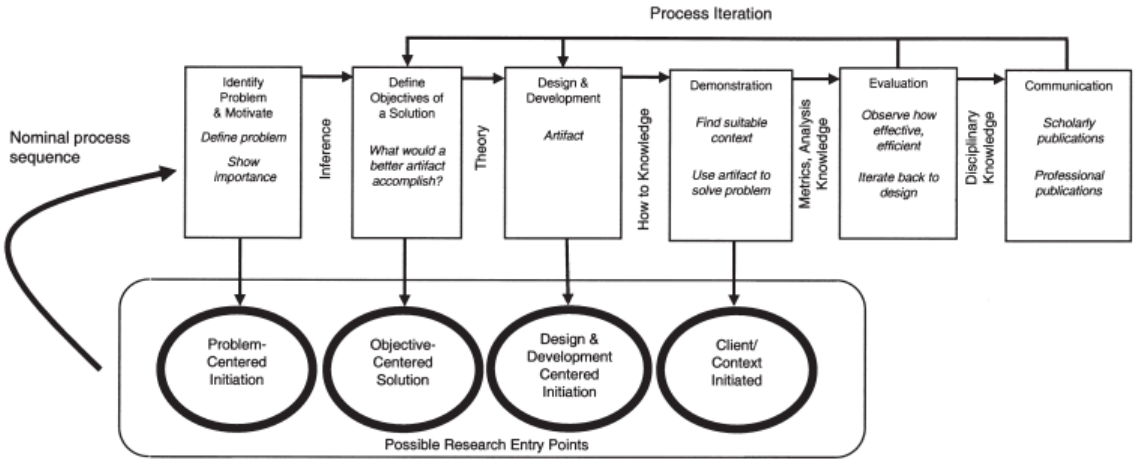


Figura 1.3: Cadeia de *activities* da metodologia *DSR* [13].

Numa primeira fase, proceder-se-á com a identificação do problema, apresentando ainda a motivação da execução do projeto, delineando-se objetivos e traçando um método de execução - **identificação do problema e motivação** e **definição de objetivos para uma solução**.

Segue-se uma descrição e análise da área de negócio em que o projeto se enquadra, bem como contextualização do estado da arte da indústria associada, seguindo-se uma análise dos requisitos levantados no processo anterior acompanhada de uma análise de valor de uma solução proposta.

Após isso, é apresentada uma proposta de *design* da solução selecionada a partir do método de apoio à decisão *Analytic Hierarchy Process (AHP)*, que precede a implementação da mesma - **design e desenvolvimento da solução**.

Com a finalidade de averiguar o cumprimento dos objetivos inicialmente definidos, termina-se com uma fase de avaliação da solução implementada - **avaliação da solução**.

Todo o processo acima descrito é acompanhado e documentado através da escrita da presente dissertação - **comunicação e exposição do processo e resultados**.

1.5 Contributos

O desenvolvimento desta plataforma traz benefícios de forma direta para três entidades distintas:

- **Artistas**, que beneficiarão de uma plataforma que disponibilizará novas ferramentas que potenciam o crescimento da sua rede de contactos, abrindo horizontes para novas oportunidades de trabalho;
- **Promotores**, que beneficiarão de uma plataforma que disponibiliza ferramentas que lhes permitirá encontrar novos artistas que melhor se encaixem com o seu plano cultural/artístico, expandindo a sua rede contactos e possibilitando um maior fluxo de clientes;

- **Agentes** que, servindo de interface entre **Promotores** e **Artistas**, tenham uma maior oportunidade de fornecer os seus serviços e/ou fazer crescer a sua rede de contactos, possibilitando um maior número de contratos.

1.6 Planeamento de Trabalho

O desenvolvimento deste projeto divide-se em 4 fases principais:

Numa fase inicial há um processo de **análise** - pesquisa e recolha de informação, que se baseia nos seguintes passos:

- Leitura de artigos que relatem o estado da cultura nacional, nomeadamente, quando confrontada com o plano da pandemia *COVID-19* e respetivos impactos;
- Entrevistas realizadas a alguns profissionais do mundo da música, com diferentes *backgrounds*;
- Análise de soluções existentes.

Segue-se uma fase de **design**, que se divide nos seguintes passos:

- Geração de diferentes soluções possíveis e acessão de qual a melhor;
- Análise de valor da solução selecionada;
- Produção de documentação que descreva a solução selecionada.

Após a fase de *design*, procede-se com a **implementação** da solução e respetivos **testes**, uma vez que esta esteja terminada.

Por fim, é feito um processo de **avaliação** da solução implementada.

A figura 1.4 representa de forma gráfica o planeamento delineado.

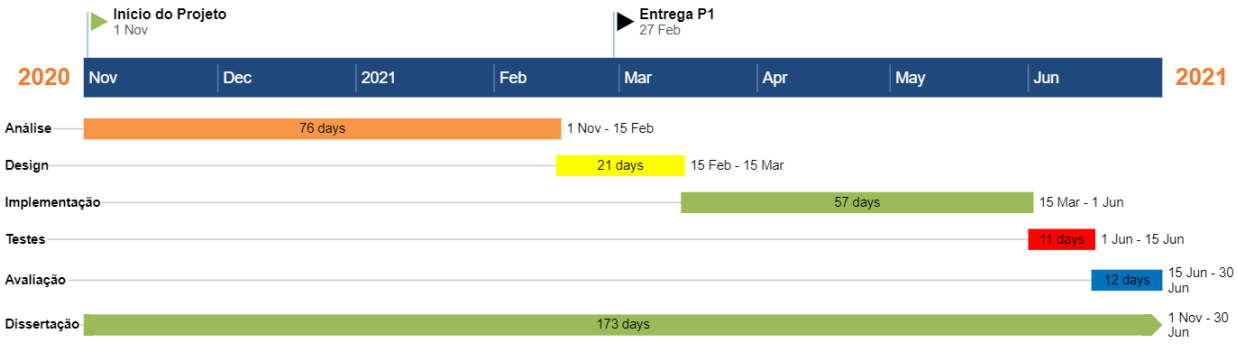


Figura 1.4: Proposta de planeamento do Projeto.

1.7 Estrutura do relatório

Em acréscimo ao presente capítulo, esta dissertação é composta por 7 capítulos adicionais.

No capítulo 2, **Contexto e Estado da Arte**, é feita uma descrição do atual estado da arte, sendo apresentados alguns trabalhos relacionados, bem como algumas tecnologias que permitam a execução de um trabalho semelhante.

No capítulo 3, **Análise de Requisitos**, é feita uma análise e levantamento dos requisitos indicados pelo consumidor final da plataforma.

No capítulo 4, **Análise de Valor**, é aferido o valor da solução, derivando-a das necessidades do cliente, fazendo-se uma análise à melhor abordagem a implementar.

No capítulo 5, **Desenho da Solução**, listam-se e detalham-se os artefactos produzidos que descrevem o *design* da solução resultante do capítulo anterior.

No capítulo 6, **Implementação da Solução**, descreve-se o processo de implementação, com especial foco em detalhes de nível técnico. Este capítulo acompanhar-se-á de alguns excertos de código, quando aplicável.

No capítulo 7, **Avaliação da Solução**, detalham-se as metodologias levadas a cabo no sentido de testar a solução implementada.

Por fim, no capítulo 8, **Conclusões**, apresentam-se observações finais do projeto, fazendo-se um levantamento dos objetivos concretizados e dando uma perspetiva do trabalho futuro a ser realizado.

Capítulo 2

Contexto e Estado da Arte

A fim de ter uma compreensão mais aprofundada da natureza do projeto, neste capítulo é feita uma abordagem ao estado da arte da temática em causa, de forma a enquadrar o leitor com o seu contexto.

Para tal, é feita uma exposição do tema sob dois formatos distintos - a opinião de um conjunto de profissionais da área sobre o estado da indústria e respetivas necessidades e a exploração e avaliação de produtos existentes;

Adicionalmente, é feita a uma abordagem às técnicas e recursos passíveis de serem utilizados no projeto em causa.

2.1 Contexto do Problema

O problema apresentado sugere que profissionais do mundo da música deparam-se com dificuldades em progredir na carreira devido ao modo de funcionamento da indústria. Este problema é potenciado tanto pela falta de ferramentas para auxiliar estes mesmos artistas na expansão da sua rede de contactos, como também, mais recentemente, pela introdução das novas restrições sociais impostas pela pandemia *COVID-19*.

2.1.1 Testemunhos Profissionais

Na fase inicial do desenvolvimento do projeto, foi realizado um conjunto de entrevistas a alguns profissionais da indústria da música. Estas entrevistas foram realizadas com 2 objetivos em mente:

- Perceber melhor o funcionamento da indústria da música, nomeadamente no que diz respeito aos seguintes pontos;
  - Expansão e manutenção da rede de contactos;
  - Processo de contratação de um artista por parte de um promotor;
  - O papel de um agente;
  - Exposição do trabalho em redes sociais.
- Perceber as necessidades da indústria, nomeadamente, no contexto apresentado no ponto anterior.

A seleção de candidatos procurou pessoas onde se identificassem as seguintes características:

- O candidato vive exclusivamente da sua atividade profissional na indústria da música;

- O candidato tem experiência considerável na indústria da música;
- No caso de a sua função ser uma de artista, tem que ser um profissional que não esteja associado a um agente - esta restrição é tida em consideração, uma vez que se pretendem artistas responsáveis pelo próprio processo de procura de promotores.

Adicionalmente e, se possível, esta seleção procurou candidatos que não partilhassem *backgrounds* com demasiado em comum. Deste processo resultaram três candidatos.

Candidato 1

- **Nome:** Jorge Loura
- **Profissão:** Músico Profissional
- **Experiência profissional:**
  - 25 anos de experiência como músico profissional;
  - Músico em diversos projetos ao longo dos anos, destacando-se, de momento, projetos como *Souq* [14], *47 de Fevereiro* [15] e *Troll's Toy* [16];
  - Diversos álbuns gravados e publicados;
  - Professor de guitarra elétrica;
  - Extenso histórico de performances ao vivo em vários contextos (festivais, bares, sub-contratação, salas de espetáculos, ...).
- **Outras observações:**
  - Considerado um dos 10 guitarristas portugueses mais preponderantes da última década pela magazine *A Arte Sonora* [17, 18].

Candidato 2

- **Nome:** Mabília Martins (Maby)
- **Profissão:** Gerente de Projetos/Agente [19]
- **Experiência profissional:**
  - Sócia da editora *A Arte Sonora* [20] desde 2008, ocupando, também, o cargo de Comercial;
  - Trabalhou na área de *Marketing*;
  - Agente de variados artistas de diversos estilos e dimensões na indústria.
- **Outras observações:**
  - Agente do artista português Rui Veloso desde 2015, sendo responsável pela transição do mesmo para o mundo digital (gestão de redes sociais, páginas *web* pessoais, carreira e imagem digital, ...).

Candidato 3

- **Nome:** Nuno Fontes
- **Profissão:** Músico Profissional
- **Experiência profissional:**
  - 25 anos de experiência como músico profissional;
  - Músico em diversos projetos ao longo dos anos, trabalhando de momento em projetos pessoais;
  - Professor de guitarra elétrica;
  - Extenso histórico de performances ao vivo em vários contextos (festivais, bares, sub-contratação, salas de espetáculos, ...).
- **Outras observações:**
  - Experiência adicional como promotor.

As entrevistas foram levadas a cabo num ambiente informal, através de chamadas de vídeo (a partir das plataformas *Facebook* e *WhatsApp*) com gravação do áudio com consentimento informado dos entrevistados.

A estrutura de cada entrevista é a seguinte:

1. Pedido de permissão para a gravação do áudio da chamada, destacando neste pedido os seguintes pontos:
   - Para que será utilizada a gravação - consulta de respostas *a posteriori*;
   - Quem terá acesso à gravação - o autor da dissertação/condutor da entrevista;
   - Quem terá acesso às informações e opiniões dadas durante a entrevista - qualquer pessoa que tiver acesso à presente dissertação.
2. Apresentação pessoal por parte do condutor da entrevista/autor da dissertação;
3. Apresentação do tema da dissertação;
4. Apresentação pessoal por parte do entrevistado, com descrição do seu histórico e experiência;
5. Entrevista com questões de opinião adaptadas a cada contexto.

As entrevistas contabilizaram um total de 3 horas e 53 minutos, com um total de 23 perguntas.

Nas duas secções que se seguem, apresenta-se uma explicação do funcionamento da área de negócio em questão, bem como alguns conceitos associados à mesma. Todo o processo é construído a partir do relato e opinião dos três profissionais descritos acima.

2.1.2 Área de Negócio

De forma a entender com maior clareza o projeto descrito nesta dissertação, é importante explicar a área de negócio associada ao problema a ser resolvido - o processo que leva à obtenção de um acordo entre um promotor e um artista.

Trata-se de um processo difícil de definir, não pela sua complexidade intrínseca, mas pelo facto de não haver um processo *standard* com uma sequência de passos já definida. Cada **artista**, **promotor** e **agente** tem a sua própria forma de gerir o seu negócio e presença

na indústria. Por isso mesmo, tal como explica Jorge Loura na sua entrevista [11], é um processo em que "[...] as coisas surgem do acaso e do caos.".

Ainda que haja consenso entre todos os candidatos e da comunidade geral de que a ferramenta mais poderosa no arsenal de um membro da indústria da música seja a sua *network* e capacidade de a expandir/gerir, esse consenso não se verifica no que toca à forma de o fazer. No seu livro "*Artist Management for the Music Industry*", *Paul Allen* refere que

> *"Concluí que ser social no mundo do negócio assemelha-se muito a desempenhar um papel [...]. Olhamos pela sala e toda a gente está a fazer o mesmo, ainda que estejam a tentar transparecer que estão aqui por mera diversão.* **É** *trabalho e* **é** *parte da tarefa de networking, ponto final. Nem todos os artistas [...] gostam de contextos sociais, mas fazer parte deles [...] é um requisito deste trabalho."* [21] (tradução livre do autor)

A presente secção visa fazer uma análise do modo como três profissionais com *backgrounds* diferentes gerem a sua atividade na indústria, destacando pontos em comum e semelhanças, retirando conclusões.

Primeiro, apresentam-se os objetivos principais de cada interveniente:

- **Artista** - O objetivo do Artista é promover a sua arte num evento/espaço que ofereça boas condições de trabalho, bem como tentar obter o máximo de lucro possível. Como profissionais independentes, estão, financeiramente, dependentes das oportunidades que surgem;
- **Agente** - O objetivo de um Agente é arranjar eventos organizados por promotores onde possam meter os artistas que compõem a sua **carteira** (conjunto de artistas que contratam o agente [12]) [22];
- **Promotor** - O objetivo principal de um Promotor é conseguir obter o maior lucro possível com a contratação de um artista para um determinado evento. Este lucro pode ser obtido sob a forma da venda de bilhetes ou de outros produtos associados ao espaço do evento - qualquer um dos formatos é influenciado diretamente pelo número de pessoas que compõe o público [4].

De seguida, apresenta-se uma visão geral de como interage cada par de intervenientes.

Agente - Promotor

O contacto entre um agente e um promotor é "normalmente [um contacto em que] são pessoas que conheço pessoalmente", diz Mabília Martins. Adicionalmente, dependendo do prestígio do agente e artistas associados, poderá tratar-se de um processo em que a escolha do(s) artista(s) é exclusiva responsabilidade do próprio agente.

No caso da candidata, esta exemplifica, sublinhando a diferença entre promotores que a abordam com o interesse de contratar o seu maior cliente - Rui Veloso - em contraste com os seus restantes artistas, dando o exemplo de que "[...] o próprio agente pode juntar vários artistas num pacote que vende a um promotor".

Agente - Artista

A possibilidade de recorrer a um agente permite ao artista reduzir o seu número de responsabilidades e focar-se mais na sua arte.

Embora a agência de um artista possa ser requisitada a agências ou a *freelancers*, mais uma vez o consenso aponta para a importância ímpar que uma *network* sólida e desenvolvida tem no que toca a contratar alguém para o efeito.

No entanto, uma vez que se trata de um contrato, tem que haver peso e medida na relação entre as duas entidades.
Por um lado, Nuno Fontes refere que os agentes devem "[...] saber como trabalha uma banda - a carga de trabalho que cada músico tem.", sugerindo que o acompanhamento do trabalho artístico e sensibilização para o investimento financeiro (instrumentos, material, ...) são bons pontos de partida para um bom agente.
Por outro lado, Mabília Martins relembra que tem de haver capacidade de distinção entre o artista e a pessoa que o representa - no seu trabalho mais recente, refere que quando se lida com artistas que lhe são conhecidos pessoalmente, há sempre um "[...] medo de que não estejamos a ser realistas quanto ao trabalho do [nome do artista]".

No que toca à comparação de uma carreira independente com uma acompanhada da contratação de um agente, na opinião de ambos os artistas entrevistados, havendo a oportunidade de contrato, esta deve ser aproveitada. Jorge Loura refere que "chega a um certo ponto que não é só o teu trabalho artístico [que conta]"e que "por muito que te esforces [como artista independente], há sempre uma barreira em que vais bater".

Artista - Promotor

No caso de artistas que não têm acesso a um agente, fica ao cargo dos mesmos todo o processo de procura de oportunidades de trabalho junto dos promotores.

Após análise dos testemunhos dos candidatos apresentados, há alguns pontos de destaque em comum.

Para um determinado evento, a principal razão apontada para o **primeiro contacto** entre um artista e um promotor é o facto de uma das partes, de forma direta ou indireta (através de terceiros, por recomendação), conhecer a outra - fazem parte da mesma *network*. Ambos os artistas entrevistados explicam a diferença, relatando exemplos com os diferentes tipos de interação.

No caso do guitarrista Jorge Loura, este refere que no caso de *eventos planeados* (ver na secção 2.1.2 a subsecção Eventos), "[...] as coisas vêm ter connosco, mais ou menos..."e que são oportunidades que "[...] surgem do acaso e do caos". Tipicamente, o artista é conhecido por alguém que faça parte da organização do evento.

> *"Concertos em sítios maiores ou com cachês mais decentes vêm de convites de pessoas que nos conhecem - não necessariamente à banda, mas conhecem-me a mim ou ao [outros membros do projeto] [...]"* [11]

Em contraste, ambos os artistas referem que no caso de *eventos não planeados* ou *eventos planeados sem tema pré-definido*, que a iniciativa de contacto parte do artista.
Nuno Fontes relata a dificuldade e atenção que o artista tem que ter neste tipo de contexto, dizendo que se tratam de situações em que não se pode assumir à partida que o promotor "[...] esteja preparado para música ao vivo", nomeadamente nos *eventos não planeados*.
Jorge Loura reflete ainda - "O que é que parte da nossa parte? Bares e coisas assim, onde nós achamos que é importante tocar para as pessoas".

Na sua experiência, Nuno Fontes sublinha ainda que o processo de *networking* é contínuo e não acaba com o 'aperto de mão' inicial com o promotor - sempre que possível, deve-se tentar garantir a existência de **contactos sucessivos**. Esta fidelidade entre um promotor e um artista pode ser formal (sob a forma de um contrato, por exemplo) ou informal [4]. Adicionalmente, refere ainda que, para o promotor, o factor mais importante a ter em consideração para este tipo de fidelização é a **opinião do público**, uma vez que uma melhor opinião pública se traduz em mais pessoas, trazendo assim mais receita ao promotor.

Eventos

O tipo de evento no qual um artista participa tem, de forma geral, influência no modo como é abordado/tem que abordar o promotor. Dos diferentes tipos de evento discutidos ao longo das sessões, é possível agrupá-los nas seguintes categorias: **Eventos Não Planeados**, **Eventos Planeados com tema pré-definido** e **Eventos Planeados sem tema pré-definido**.

Entenda-se por **tema** aquilo que o produto artístico pretenda representar, seja um estilo de música específico (*rock, folk, fusion*, ...), uma ocasião (casamento, Natal, Dia dos Namorados, ...) ou qualquer outro tipo de temática (anos 80, *Disney*, Primavera, ...).

Eventos Não Planeados - Trata-se de um evento não premeditado por parte do promotor. Neste tipo de evento, o primeiro contacto é feito pelo artista/agente. Exemplos deste tipo de evento podem ser atuações em bares [10] ou um artista propor uma noite numa sala de espetáculos [12].

Eventos Planeados com tema pré-definido - Trata-se de um evento premeditado por parte do promotor. Neste tipo de evento, o primeiro contacto pode ser feito por qualquer um dos intervenientes. Aqui, o promotor pode dirigir-se diretamente a artistas do conhecimento da organização que encaixem na temática do evento, bem como aos agentes, em busca de algum artista da sua carteira que encaixe. Alternativamente, o promotor pode anunciar a concurso as vagas disponíveis do evento em causa, esperando pela candidatura dos artistas por parte dos próprios ou respetivos agentes. Um exemplo deste tipo de evento pode ser um festival [11].

Eventos Planeados sem tema pré-definido - Nesta categoria caem os eventos premeditados por parte do promotor, mas que não têm um tema pré-definido. De forma similar ao tipo anterior, há a possibilidade de o promotor se dirigir tanto diretamente ao artista como ao próprio agente, bem como abrir vagas a concurso. No entanto, tal como explica Mabília Martins na sua entrevista, devido ao facto de não haver uma linha temática que una os artistas neste tipo de evento, é mais comum os promotores recorrerem aos agentes, dando-lhes a liberdade de planearem o evento como pretenderem - "Quem está à frente das salas de espetáculos é extremamente influenciável pelos agentes [...] e normalmente contratam quem o agente quer que seja contratado."[12].

A Competitividade do Mercado

No que toca à competitividade do mercado, há um fator extra a ter em conta. Trata-se de um mercado extremamente saturado por artistas amadores (não profissionais) [10] que competem pelos mesmos lugares e promotores que os artistas profissionais.

Na sua entrevista, Nuno Fontes refere que artistas profissionais estão sujeitos a competição desleal, ainda que de forma não intencional por parte de quem a pratica - os artistas amadores.

Esta competição desleal surge nos preços associados à contratação de um artista para um determinado evento. Artistas amadores têm a possibilidade de pedir preços consideravelmente menores para fornecer o mesmo serviço (ainda que de qualidade inferior).

2.2 Trabalhos relacionados

Para esta secção foi feita a recolha de algumas soluções semelhantes ao projeto desenvolvido.

O critério de seleção de soluções alternativas passou por escolher plataformas/serviços que fornecem ferramentas ou apoios a artistas ou que permitem a procura (por parte de outros utilizadores) de artistas para potencial contratação.

Inicialmente, é feito um sumário de cada uma delas onde se descreve, de forma breve, no que consiste cada solução.

Segue-se uma contextualização de cada solução apresentada, onde são apresentados os respetivos objetivos, algumas características e funcionalidades, bem como uma listagem de vantagens e desvantagens associadas ao problema proposto.

Por fim, são apresentadas algumas considerações gerais.

2.2.1 Sumário

Abaixo estão listadas algumas das soluções existentes encontradas que vão ao encontro do tipo de plataforma descrita:

- **Portal do Artista** - Plataforma pertencente à Fundação GDA [23] (fundação pertencente à Gestão dos Direitos dos Artistas [24]) que dá a oportunidade a artistas de procurarem e candidatarem-se a programas de apoios e incentivos financeiros com uma finalidade pré-estabelecida. Disponível em: `https://www.portaldoartista.pt/` [25];
- **Artistas ao Sul** - Plataforma destinada a promotores e artistas, com o objetivo de servir de catálogo de artistas e promotores, facilitando a pesquisa através da disponibilização de diferentes filtros de pesquisa. Disponível em: `https://artistasaosul.pt/` [26];
- **Portal de Artistas** - Plataforma destinada a promotores e artistas, com o objetivo de servir de catálogo de artistas, facilitando a pesquisa através da disponibilização de diferentes filtros de pesquisa. Disponível em: `https://portaldeartistas.pt/` [27];
- **Central de Artistas** - Plataforma destinada a promotores e artistas, com o objetivo de servir de catálogo de artistas, facilitando a pesquisa através da disponibilização de diferentes filtros de pesquisa. Disponível em: `http://www.centraldeartistas.pt/` [28];
- **LANDR**- Serviço *online* para criadores de música, que disponibiliza várias funcionalidades que ajudam no processo de produção, masterização e distribuição de música. Adicionalmente, tem uma secção para procura e partilha de artistas. Disponível em: `https://www.landr.com/pt/` [29];
- **Spotify, Apple Music, Deezer** - Serviços de *streaming* de música. Disponíveis, respetivamente, em: `https://www.spotify.com/` [30], `https://www.apple.com/apple-music/` [31] e `https://www.deezer.com/` [32];

- **Facebook, Twitter, Instagram, LinkedIn** - Redes sociais. Disponíveis, respetivamente, em: `https://www.facebook.com/` [33], `https://www.twitter.com/` [34], `https://www.instagram.com/` [35] e `https://www.linkedin.com/` [36];

2.2.2 Contextualização de soluções

Diferentes soluções têm diferentes objetivos e/ou diferentes abordagens sobre um determinado problema.

Esta secção destina-se a detalhar os principais objetivos de cada uma das plataformas descritas, abordando ainda o papel e funcionalidades que cada uma oferece. Adicionalmente, são detalhados alguns pontos fortes e lacunas de cada solução.

São apresentados diferentes grupos, que englobam plataformas de teor ou funcionalidades semelhantes.

Redes Sociais

Neste grupo, inserem-se nomes como *Facebook*, *Instagram*, *Twitter* e *LinkedIn*.

Este tipo de plataformas visa providenciar aos utilizadores ferramentas que permitem um tipo de *networking* genérico, no sentido em que um utilizador não tem associado ao seu perfil um determinado papel na sociedade/na plataforma. Cada perfil de utilizador reflete um membro da sociedade, seja qual for o seu papel.

Utilizando a missão da *Facebook Inc's* [37] - "Dar às pessoas o poder de construir uma comunidade e unir o mundo."(tradução livre do autor) - é possível inferir que não há especificação do tipo de utilizador que interage com a plataforma.

No entanto, este tipo de plataforma não está totalmente desprovida de funcionalidades que ajudem um artista a promover o seu trabalho, destacando-se aqui o *Facebook*, onde é possível criar perfis do tipo **Página** (*Business Page*), que providenciam funcionalidades extra [38]. Alguns exemplos são:

- **Promoção da Página**, que permite efetuar ações como obter mais visitantes ao perfil ou site do artista, bem como promover uma determinada publicação;
- **Estatísticas**, que fornece alguns dados sobre o alcance da página em si e respetivas publicações, permitindo determinar o tipo de publicação que tem mais impacto no público;
- **Ferramentas de Publicação**, que são um facilitador do processo de publicação de conteúdo.

Uma determinada *Página* pode ter uma *tag* associada que descreve o tipo de entidade que representa, por exemplo, *Artista* ou *Autor(a)*.

O *Twitter* e o *Instagram* oferecem um leque muito mais reduzido de funcionalidades.

Vantagens deste tipo de solução:

- A quantidade de utilizadores deste tipo de plataformas é extremamente elevado, permitindo alcançar um maior número de pessoas, quer do ponto de vista de um artista em busca de público, quer do ponto de vista de um agente/promotor em busca de um artista [39];

- Fornece algumas funcionalidades para análise do público que interage com o conteúdo partilhado;
- Interação direta com a comunidade geral, através de comentários, mensagens, *likes* e partilha de conteúdo.

Desvantagens deste tipo de solução:

- A quantidade elevada de páginas existentes dificulta a tarefa de pesquisa por aquilo que se pretende [38];
- As funcionalidades disponibilizadas não são desenhadas para a indústria da música;
- O perfil de um artista/promotor/agente não se distingue do perfil de qualquer outro tipo;
- Não existem *workflows* especialmente desenhados para a indústria da música.

Plataformas de Streaming

Neste grupo, inserem-se nomes como *Spotify*, *Apple Music*, *Deezer*, entre outros.

Em contraste com as redes sociais, este tipo de plataforma já faz uma distinção entre dois tipos de utilizador - o utilizador normal e o artista. No entanto, tratam-se de plataformas exclusivamente desenhadas para o utilizador final, não tendo nenhum tipo de funcionalidade ou *workflow* próprio para a comunicação entre entidades.

A utilização deste tipo de plataforma por parte dos artistas está também sujeito a intermediários - serviços de distribuição.

Vantagens deste tipo de solução:

- Facilidade de acesso por parte do público geral ao produto do artista;

Desvantagens deste tipo de solução:

- Funcionalidades limitadas;
- Publicação de conteúdo não é feito diretamente pelos artistas [40];
- Não permite o contacto direto a partir dos próprios serviços.

Serviços Online

Nesta categoria insere-se apenas o *LANDR*. Embora o próprio *LANDR* tenha alternativas, é o único serviço do género que tem funcionalidades de *networking* integradas.

Tratam-se de serviços que fornecem ferramentas para ajudar artistas no processo criativo.

Uma vez que o foco principal deste tipo de serviços não se alinha com o contexto do projeto, as listagens de vantagens e desvantagens abaixo focar-se-ão apenas na funcionalidade de exceção do *LANDR*.

Vantagens deste tipo de solução:

- As pessoas que constituem a comunidade são, maioritariamente, profissionais com influência, havendo um maior valor associado ao potencial *networking* gerado na plataforma deste serviço;

Desvantagens deste tipo de solução:

- O objetivo dos utilizadores é ser contratados apenas para o processo criativo;
- Quantidade de utilizadores reduzida em Portugal.

Plataformas Especializadas

Nesta categoria inserem-se plataformas que, de uma forma ou de outra, foram concebidas com o objetivo de agir na indústria da música.

Embora se insira na mesma categoria que as restantes, o *Portal do Artista* tem objetivos muito distintos das restantes, tratando-se de um catálogo de concursos e apoios financeiros a artistas [41].

Em contraste, os restantes agem como um catálogo de artistas, permitindo-lhes criar um perfil onde definem o tipo de trabalho que desenvolvem e o tipo de eventos para os quais pretendem ser contratados. Permitem ainda a utilizadores externos requisitar um orçamento a um determinado artista.
Adicionalmente, no caso do *Portal de Artistas*, a plataforma está munida de um sistema de recomendação optativo para o utilizador. Aquando do pedido de orçamento a um artista, a plataforma pode enviar pedidos de orçamento a 5 outros artistas com características semelhantes.
Em contra-partida, nenhuma destas soluções oferece funcionalidades de *networking* entre os utilizadores.

Vantagens deste tipo de solução:

- O contexto de utilização alinha-se com a indústria da música;
- A informação referente aos artistas é relevante para os promotores;
- Contexto quase exclusivamente referente ao mercado português.

Desvantagens deste tipo de solução:

- A falta de funcionalidades relacionadas com *networking*;
- No caso do *Artistas ao Sul*, o facto de funcionar exclusivamente à base de um sistema de subscrição mensal, com vários tipos de subscrição, não havendo um plano de subscrição grátis. Adicionalmente, a criação de um evento só é possível com o nível mais elevado de subscrição [42].

2.2.3 Comparação de Soluções

Nesta secção faz-se uma comparação das soluções apresentadas na secção 2.2.1 segundo alguns parâmetros. Cada parâmetro procura responder a uma pergunta.

- **Público Alvo** - A quem se destina esta plataforma?
- **Tipos Possíveis de Utilizador** - Como utilizador desta plataforma, posso identificar-me como Artista, Agente ou Promotor?
- **Funcionalidades** - Que funcionalidades tem a plataforma?

Para efeitos de simplicidade, as plataformas que se inserem nas categorias **Redes Sociais** e **Serviços Streaming** serão representadas, em conjunto, pelo nome da respetiva categoria.

Público Alvo

Plataformas	Público Alvo			
	Genérico	Artistas	Promotores	Agentes
Portal de Artistas		X	X	
Artistas ao Sul		X	X	
Central de Artistas		X	X	
Portal do Artista		X		
LANDR		X		
Redes Sociais	X			
Serviços de *Streaming*	X			

Tabela 2.1: Público Alvo das plataformas alternativas.

A avaliação deste parâmetro visa, de uma forma genérica, servir como ponto de partida na filtragem de soluções. Embora seja possível retirar lições a partir de cada uma das soluções, aqui destacam-se as que foram desenhadas em torno de pelo menos um dos 3 intervenientes: **Artista**, **Agente** e/ou **Promotor**.

O preenchimento da tabela 2.1 foi feito a partir dos princípios associados a cada plataforma, junto da correspondente Missão, Visão e Valores, quando disponibilizados [37, 43–53].

A partir desta tabela destacam-se os seguintes pontos:

- Nenhuma plataforma engloba no seu público alvo os **agentes**;
- Existem 3 possibilidades no mercado onde se pode explorar funcionalidades e modelos de negócio virados para os **promotores**;
- Existem 5 possibilidades no mercado onde se pode explorar funcionalidades e modelos de negócio virados para os **artistas**.

Tipos Possíveis de Utilizador

Plataformas	Tipos Possíveis de Utilizador			
	Genérico	Artistas	Promotores	Agentes
Portal de Artistas		X		
Artistas ao Sul		X	X	
Central de Artistas		X		
Portal do Artista		X		
LANDR		X		
Redes Sociais	X	X	X	X
Serviços de *Streaming*	X	X		

Tabela 2.2: Tipos Possíveis de Utilizador das plataformas alternativas.

Em contraste com o parâmetro anterior, este visa expor o modo como um utilizador registado na plataforma se pode identificar dentro da mesma.

A identificação de um utilizador dentro da plataforma pode ser feita de várias maneiras - contexto da própria plataforma, uma *tag* associada ao utilizador, definição de um tipo de perfil ou uma mera descrição textual no perfil do utilizador.

A partir da tabela 2.2, destacam-se os seguintes pontos:

- As plataformas **Portal de Artistas** e **Central de Artistas**, embora incluam como público alvo os **promotores**, não oferecem a opção aos mesmos de criar uma conta de utilizador;
- As **Redes Sociais** possibilitam a distinção do tipo de utilizador de forma a ter qualquer tipo de utilizador;
- Os **Serviços de Streaming**, embora não tenham como público principal os próprios artistas (são desenhados como plataformas para dar acesso público ao trabalho artístico), possibilitam aos artistas de ter um perfil próprio.

Funcionalidades

A análise das funcionalidades disponibilizadas por cada plataforma será dividida em 3 partes:

- **Funcionalidades Base** - Funcionalidades mais genéricas, escolhendo para este propósito:
  - Existência de algum tipo de **Filtragem** de dados (perfis, utilizadores, propostas, contratos, ...);
  - Existência de **Workflows**, que pretende responder às questões - É uma plataforma meramente expositiva (um repositório/catálogo de dados)? Permite ao utilizador executar algum tipo de sequência de ações além da consulta de dados? Note-se que para responder a esta pergunta foram apenas contempladas ações de relevância para o tema (ver secção 2.2.4);
  - Existência de algum tipo de **Sistema de Recomendação** para o utilizador.
- **Funcionalidades de Networking** - Funcionalidades que permitam a interação entre utilizadores da plataforma, escolhendo para este propósito:
  - Possibilidade de comunicação entre utilizadores através de **mensagens internas**;
  - Possibilidade de estabelecer uma **ligação entre utilizadores** sob a forma de uma lista de contactos associada a um utilizador;
  - Possibilidade de partilhar com outros utilizadores conteúdo relacionado com o trabalho desenvolvido pelo próprio utilizador.
- **Funcionalidades de Negócio** - Funcionalidades que permitam ao utilizador (nomeadamente a **Artistas**, **Agentes** e/ou **Promotores**) expor um serviço a prestar ou procurar quem preste um serviço. Aqui, procurou-se responder às seguintes perguntas:
  - Eu, como [Artista/Agente/Promotor], disponho de ferramentas que me permitem **criar** a oferta de um serviço ou produto na plataforma?

– Eu, como [Artista/Agente/Promotor], disponho de ferramentas para **procurar** ofertas existentes na plataforma que vão ao encontro dos meus objetivos (ver secção 2.1.2) e critérios ?

Plataformas	**Funcionalidades**		
	Workflow	**Filtragem**	**Recomendação**
Portal de Artistas	X	X	X
Artistas ao Sul	X	X	
Central de Artistas	X	X	
Portal do Artista	X		
LANDR		X	
Redes Sociais		X	X
Serviços de *Streaming*			X

Tabela 2.3: Funcionalidades Base das plataformas alternativas.

Com a análise da tabela 2.3, é importante destacar o facto de o **Portal de Artistas** ser a única plataforma onde se verificam os três parâmetros - embora as redes sociais permitam a pesquisa e filtragem dos perfis existentes nas respetivas plataformas e tenham um sistema de recomendação orientado ao utilizador, não têm nenhum tipo de *workflow* orientado à vertente performativa da indústria da música.

Em contraste, o **Portal de Artistas** dá a opção ao utilizador de pedir um orçamento a um determinado Artista, através de um formulário que dará origem a uma mensagem [54]. As plataformas **Artistas do Sul** e **Central de Artistas** também disponibilizam esta funcionalidade.

Note-se ainda que, apesar da plataforma **Central de Artistas** disponibilizar a opção de pedir orçamento a um artista, esta aparenta não estar a funcionar.

Plataformas	Funcionalidades		
	Mensagens Internas	Ligação entre Utilizadores	Partilha de Conteúdo
Portal de Artistas	X		X
Artistas ao Sul	X		X
Central de Artistas	X		X
Portal do Artista	X		
LANDR	X		X
Redes Sociais	X	X	X
Serviços de *Streaming*		X	X

Tabela 2.4: Funcionalidades de *Networking* das plataformas alternativas.

Note-se que na avaliação do parâmetro **Mensagens Internas** foram incluídas soluções que contenham funcionalidades que estabelecem o contacto entre dois utilizadores, ignorando o facto de o contacto consequente ser feito dentro ou fora da plataforma. Um exemplo será a existência de um formulário na plataforma que será utilizado pela mesma para enviar um *e-mail* para o destinatário, por onde o contacto entre os intervenientes tem continuação.

Uma vez mais, no caso da **Central de Artistas**, embora a funcionalidade de **Mensagens Internas** existir, aparenta não estar a funcionar.

Plataformas	Funcionalidades					
	Criar Oferta			Procurar Oferta		
	Artistas	Promotores	Agentes	Artistas	Promotores	Agentes
Portal de Artistas	X				X	
Artistas ao Sul	X	X		X	X	
Central de Artistas	X				X	
Portal do Artista						
LANDR	X			X		
Redes Sociais		X				
Serviços de *Streaming*						

Tabela 2.5: Funcionalidades de Negócio das plataformas alternativas.

Ao analisar a tabela 2.5, destaca-se o facto de que nenhuma opção existente disponibiliza, a um **Agente**, ferramentas de criação e procura de ofertas que vão ao encontro dos seus interesses e objetivos.
É importante referir que, para este parâmetro, foram contempladas funcionalidades/filtros próprios para o efeito de procura e oferta.

No caso dos **Serviços de Streaming**, embora estes permitam a publicação do trabalho do artista, esta exposição é orientada ao utilizador comum, não indo (completamente) ao encontro dos objetivos e interesses dos **Promotores** e **Agentes**.

2.2.4 Observações Finais

Algumas das soluções, embora ofereçam algum tipo de funcionalidade ou conceito que vá ao encontro da temática pretendida, são concebidas com um objetivo completamente distinto. Os **Serviços de Streaming** são concebidos com o intuito de facilitar o acesso a música ao utilizador final e o **Portal do Artista** tem um foco na partilha de programas e concursos de incentivo financeiro/cultural nos quais os artistas podem fazer parte. Por esta razão, estas duas opções não são ideias.

No caso particular do **LANDR**, embora haja uma secção para contratação de artistas e de se tratar de uma plataforma especialmente desenhada para a indústria da música, é uma solução focada numa vertente da indústria diferente da parte performativa. Esta plataforma está focada exclusivamente no processo criativo e de produção musical - a secção de contratação, inclusive, tem por objetivo o contacto com artistas/técnicos para o processo criativo.

Restam duas opções - **Redes Sociais** e **Plataformas Especializadas**.
Por um lado, as **Redes Sociais** mostram estar muito mais aptas para suportar as necessidades de *networking* dos utilizadores mas, uma vez que não são desenhadas para o efeito, não estão munidas de funcionalidades com a indústria da música em mente. As ferramentas que efetivamente disponibilizam, são desenhadas para negócios genéricos.
Em contraste, as **Plataformas Especializadas** conseguem oferecer algumas funcionalidades mais adequadas que, embora de boa qualidade, pecam pela não abrangência de todos os atores, nomeadamente os **Agentes**. Adicionalmente, tratam-se de plataformas desprovidas de algumas funcionalidades referentes à parte de *networking*, mais prevalentes nas **Redes Sociais**.

2.3 Tecnologias existentes

A presente dissertação debruça-se sobre o desenvolvimento de uma aplicação *web*. Nesta secção expõem-se algumas tecnologias existentes que permitem o desenvolvimento de uma solução dessa natureza.

Neste sentido, serão analisadas tecnologias referentes a *frameworks* de *web development* e bases de dados. Dentro de cada vertente apresentam-se algumas alternativas, sendo feita uma comparação entre elas.

A comparação entre tecnologias da mesma categoria procura basear-se em parâmetros e contextos de relevância e semelhantes àqueles que se pretendem implementar no projeto em mãos. Como tal, os resultados apresentados ao longo das tabelas da presente secção, juntamente com as conclusões obtidas, são influenciadas pelo contexto do projeto da dissertação. Adicionalmente, a data de criação dos estudos apontados é tida como um critério de relevância, uma vez que se tratam de tecnologias em constante desenvolvimento.

2.3.1 Frameworks

As diferentes soluções apresentadas nesta secção foram escolhidas com base em três critérios iniciais:

- Soluções baseadas em linguagens de programação com as quais o autor está mais familiarizado. Esta restrição foi auto-proposta no sentido de reduzir o tempo necessário de aprendizagem das tecnologias a ser utilizadas, quando aplicável;
- Soluções baseadas na sua popularidade recente. Esta restrição visa procurar soluções que tenham uma comunidade ativa e que estejam munidas de boa documentação;
- Soluções *open-source*.

Serão analisadas *frameworks* para o *backend* e *frontend* da solução.

Frontend

Com o objetivo de desenvolver o *frontend* da aplicação, foram consideradas as seguintes opções para *framework*:

- **Vue.js** - *Framework open-source* desenvolvida por Evan You, em 2014. Trata-se de uma solução *lightweight* para desenvolvimento de *User Interface (UI)* e *Single-Page Application (SPA)*. Disponível em `https://vuejs.org/` [55];
- **React** - *Framework* desenvolvida pelo *Facebook*, em 2013, tendo sido convertida mais tarde para *open-source*. Uma das tecnologias mais promissoras a longo prazo, uma vez que engloba o conceito de *open-source* numa ferramenta pertencente a uma das maiores potências tecnológicas mundiais. Disponível em `https://reactjs.org/` [56];
- **Angular** - *Framework open-source* baseada em *TypeScript* desenvolvida pela *Google*, inicialmente lançada em 2016. Tal como **React**, trata-se de uma tecnologia com perspetivas de longevidade elevadas. Disponível em `https://angular.io/` [57].

De um ponto de vista de performance, a comparação entre as *frameworks* apresentadas será feita segundo os seguintes critérios [58]:

- **Tempo de Startup** - Tempo necessário para carregar, dar *parse* a código *Javascript* e renderizar a página;
- **Criação de Linhas** - Tempo necessário para apresentar uma tabela com 1.000 linhas após o período de *startup*;
- **Memória em runtime** - Memória utilizada para o teste **Criação de Linhas**.

Os detalhes da execução dos testes estão disponíveis em `https://github.com/krausest/js-framework-benchmark` [58]. Na tabela 2.6 apresentam-se os resultados:

	Startup	**Criação**	**Memória**
ReactJS	52.7 ms	181.2 ms	5.2 MB
VueJS	16.0 ms	148.9 ms	3.5 MB
Angular	79.8 ms	190.0 ms	5.7 MB

Tabela 2.6: Resultados dos testes de *benchmark* de performance.

Analisando a tabela 2.6, é possível determinar que, a nível de performance, *Vue.js* obteve os melhores resultados em todos os testes com uma margem considerável. Este resultado é esperado, uma vez que se trata de uma *framework* mais leve e que consome menos recursos no seu funcionamento. *React* destaca-se em segundo lugar, ainda que com um desfasamento menor para com *Angular*, quando comparado com os resultados obtidos por *VueJS*.

No que toca à opinião da comunidade de *Web Development*, um estudo conduzido pela *State of Javascript* em 2020 [59] demonstra a forma como estas e outras *frameworks* são encaradas pela comunidade.

A tabela 2.7 demonstra o *ranking* das 3 soluções selecionadas nas categorias de **Satisfaction**, **Interest**, **Usage** e **Awareness**.

	Satisfaction	**Interest**	**Usage**	**Awareness**
ReactJS	2º (88%)	3º (58%)	1º (80%)	1º (100%)
VueJS	3º (85%)	2º (63%)	3º (49%)	2º (99%)
Angular	8º (42%)	8º (21%)	2º (56%)	1º (100%)

Tabela 2.7: Classificação e pontuações das *frameworks* atribuídas pela comunidade.

Analisando a tabela 2.7 é possível averiguar que, de uma forma geral, o *React* é a *framework* de preferência. É também possível concluir que apesar da popularidade, *Angular* não se apresenta como uma solução satisfatória para os desenvolvedores que a adotam. A classificação geral de uma *framework* traduzir-se-á num maior suporte por parte da comunidade *online*.

Backend

Com o objetivo de desenvolver o *backend* da aplicação, foram consideradas as seguintes opções para *framework*:

- **Spring** - *Framework* baseada em *Java* desenvolvida por Rod Johnson, lançada em 2002. Embora as funcionalidades base disponibilizadas pela *framework* possam ser utilizadas por qualquer tipo de aplicação *Java*, a mesma disponibiliza extensões pensadas no desenvolvimento *web*. Disponível em `https://spring.io/` [60];
- **Django** - *Framework open-source* baseada em *Python* para desenvolvimento *web*. Desenvolvida no ano 2003 e em rápido crescimento de popularidade nos anos recentes, a par com o crescimento da popularidade da linguagem de programação associada. Disponível em `https://djangoproject.com/` [61];
- **Node.js** - *Runtime environment open-source* baseado em *JavaScript*. Desenvolvido por Ryan Dahl em 2009 de forma a permitir *server-side scripting* para criar páginas *web* dinâmicas. Disponível em `https://nodejs.org/` [62].

No sentido de fazer uma comparação de performance entre as 3 soluções apresentadas, são analisados os seguintes critérios [63]:

- **Número de pedidos por segundo** - Quantidade de pedidos *Hypertext Transfer Protocol (HTTP)* processados por segundo, de forma a avaliar a capacidade de carga do sistema;
- **Latência do 50º Percentil** - Tempo de latência para processar metade dos pedidos - representa o tempo necessário para processar todos os pedidos mais leves. Este tempo representa a utilização base da *framework*;
- **Latência do 90º Percentil** - Tempo de latência para processar 90% dos pedidos - representa o tempo necessário para processar os pedidos mais pesados, sem considerar

os *outliers* mais raros. Este tempo representa o comportamento perante pedidos mais pesados que a média.

Os detalhes da execução dos testes estão disponíveis em `https://github.com/the-benchmarker/web-frameworks` [63]. Na tabela 2.8 apresentam-se os resultados:

	N. Pedidos	50º Percentil	90º Percentil	Relação entre percentis
Spring	69.425,3	657,0 ms	4.784,7 ms	7.28
Django	9.596,7	6.973,7 ms	8.639,3 ms	1.23
Node.js	24.937	2.505 ms	3.853,7 ms	1.54

Tabela 2.8: Resultados dos testes de *benchmark* de performance.

Analisando a tabela 2.8, é possível chegar às seguintes conclusões:

- **Spring** apresenta, com uma margem considerável, os melhores resultados. Contudo tem uma maior discrepância nos tempos entre percentis, o que significa que pedidos mais pesados no sistema são, em comparação com a média, bastante mais lentos (7,28 vezes mais lentos). É possível concluir que **Spring** é a melhor *framework*, de entre as selecionadas, para processar pedidos leves. Esta será a *framework* mais rápida das três;
- **Django**, sendo a *framework* que apresenta piores resultados médios, é também a que apresenta uma maior estabilidade, uma vez que o aumento de tempo entre percentis é o menos elevado (1,23 vezes superior). Esta será a *framework* mais estável das três;
- **Node.js** apresenta resultados satisfatórios, num equilíbrio entre performance e estabilidade. Uma vez que o valor do 90º percentil é o mais baixo em termos absolutos e o 2º mais baixo em termos relativos, pode-se concluir que esta *framework* está melhor preparada para lidar com qualquer tipo de tráfico. Esta será a *framework* mais versátil das três.

2.3.2 Media Storage Service

Com o objetivo de guardar os conteúdos audiovisuais referentes à aplicação, foram consideradas as seguintes opções:

- **Cloudinary** - Produto desenvolvido pela empresa *Cloudinary*, que é disponibilizado como um *Software as a Service (SaaS)*. Trata-se de um serviço de armazenamento e gestão de conteúdo audio-visual para a *web*, nomeadamente, vídeos e imagens. Além de funcionalidades de gestão, ainda fornece uma quantidade elevada de ferramentas de manipulação de imagem e vídeo, de forma a facilitar a sua incorporação em páginas e aplicações *web*. Disponível em `https://cloudinary.com/` [64].
- **Amazon S3** - Também conhecido como *Amazon Simple Storage Service*, trata-se de um serviço de *Cloud Storage* fornecido pela *Amazon Web Services*. Este serviço permite armazenar objetos de qualquer tipo numa infraestrutura escalável, que disponibiliza funcionalidades como *backup* e recuperação de dados. Disponível em `https://aws.amazon.com/pt/s3/` [65].

Uma vez que para o desenvolvimento deste projeto, no contexto académico em que se insere, procuram-se soluções sem custos associados, a tabela 2.9 analisa os planos oferecidos

por cada um dos serviços, a custo zero. Ambos os serviços têm opção de utilização via subscrição, que não será incluída para análise.

	Cloudinary	**Amazon S3**
Espaço de Armazenamento	25 GB	5 GB
Número de Operações	25000 Operações	20000 GET + 2000 PUT
Duração do Período Grátis	Plano sem prazo associado	12 meses

Tabela 2.9: Propriedades dos planos grátis associados a cada serviço.

Analisando a tabela 2.9 é possível averiguar que, considerando apenas os planos oferecidos a custo zero, a oferta dada pelo serviço da *Cloudinary* é preferível, uma vez que oferece maior espaço de armazenamento, não tem prazo associado e oferece mais operações. Adicionalmente, é importante referir que o *Cloudinary* oferece ainda funcionalidades próprias para transformação/manuseamento de ficheiros audio-visuais (imagem e vídeo), retirando essa responsabilidade ao produto desenvolvido. Por outro lado, a *Amazon S3* oferece armazenamento para objetos de qualquer tipo.

2.3.3 Gestão do Projeto

De forma a conseguir um processo de desenvolvimento do projeto mais organizado e eficiente, é importante selecionar tecnologias que facilitem a gestão das tarefas a realizar. Nesse sentido, seleciona-se uma ferramenta para cada uma das seguintes categorias: **Controlo de Versões**, **Repositório** e **Gestão do Projeto**.

A tecnologia selecionada para fazer o **Controlo de Versões** do projeto foi o **Git**. Desenvolvido por Linus Torvalds, o *Git* foi desenvolvido com o objetivo de criar um sistema que permite não só fazer as gestão de diferentes versões de um dado projeto, mas que também permita fazer *backup* ao projeto, quando necessário. O *Git* permite o desenvolvimento do projeto de forma concorrente, por várias pessoas em várias localizações.

O *Git* permite comunicação e sincronização da informação relativa a um projeto através da existência de um **Repositório** central (normalmente *online*) que armazena toda a informação. Para este projeto, foi selecionado o **Bitbucket** para repositório - um serviço de *hosting* de repositórios pertencente ao grupo *Atlasian*, que oferece funcionalidades de planeamento de tarefas, geração de relatórios estatísticos sobre o projeto a ser desenvolvido e de automação da execução de processos de teste e *build* do projeto hospedado.

Por fim, de forma a permitir uma organização das diferentes tarefas delineadas durante as diferentes fases do desenvolvimento do projeto, seleciona-se o **Trello** - uma aplicação *web* pertencente ao grupo *Atlasian* com o objetivo de fornecer um sistema de gestão de projetos com base numa interface simples e intuitiva.

2.4 Conclusões

Com a conclusão deste capítulo, o leitor tem agora uma melhor perceção do modo de funcionamento da indústria da música, que elementos a compõem e a forma como interagem entre si. É possível concluir ainda que se trata de uma vertente da indústria que carece de um modo de funcionamento padronizado, estando o mesmo aberto à interpretação de cada interveniente.

Adicionalmente, é feita uma abordagem às diferentes soluções oferecidas pelo mercado. No conjunto de soluções existente, é possível identificar 2 lacunas-chave:

- Não existem soluções que juntem componentes de *networking* e regras de negócio associadas ao domínio em questão;
- Os agentes não têm nenhuma solução disponível no mercado que vá ao encontro das suas necessidades.

Por fim, são analisadas diferentes tecnologias que permitem desenvolver uma solução que atinja os objetivos traçados. De forma a selecionar as tecnologias utilizadas no desenvolvimento deste projeto, numa análise final serão combinadas as conclusões obtidas ao longo de cada categoria com a opinião e preferência do autor.

Em primeiro, a escolha de uma *framework* para desenvolvimento do *frontend* da solução. A análise feita no capítulo apresenta *Vue.JS* como a melhor solução em termos de performance e *ReactJS* a solução mais proeminente no mercado. Uma vez que o autor pretende também investir tempo em adquirir conhecimento e experiência em tecnologias de relevância, a tecnologia selecionada nesta categoria é *ReactJS*.

De seguida, a escolha de uma *framework* para desenvolvimento do *backend* da solução. A análise feita no capítulo apresenta diferentes vantagens para cada uma das soluções consideradas. No contexto da presente dissertação, que pretende desenvolver um *Minimum Viable Product (MVP)*, a tecnologia *Django* apresenta uma vantagem adicional sobre as restantes alternativas - o tempo de desenvolvimento de novas funcionalidades é inferior. Por esta mesma razão, aliada à estabilidade associada à *framework*, esta é selecionada para o desenvolvimento da solução. Adicionalmente, para a gestão de conteúdo audiovisual associado à plataforma a desenvolver, a *framework* de eleição é *NodeJS*, pela versatilidade apresentada.

Por fim, no que toca à seleção de um *Media Storage Service*, opta-se pelo *Cloudinary*, uma vez que apresenta a melhor oferta no plano grátis.

Capítulo 3

Análise de Requisitos

A partir da análise dos relatos de profissionais (secção 2.1.1), é possível auferir alguns dos requisitos do cliente, bem como conceitos de negócio. No sentido de processar esses requisitos, a presente secção tem como objetivo a categorizá-los, bem como apresentar uma contextualização e descrição.

3.1 Modelo de Domínio

Numa primeira fase, é necessário mapear e perceber os conceitos de domínio associados ao problema. Com essa finalidade, foi desenvolvido um modelo de domínio, apresentado na figura 3.1.

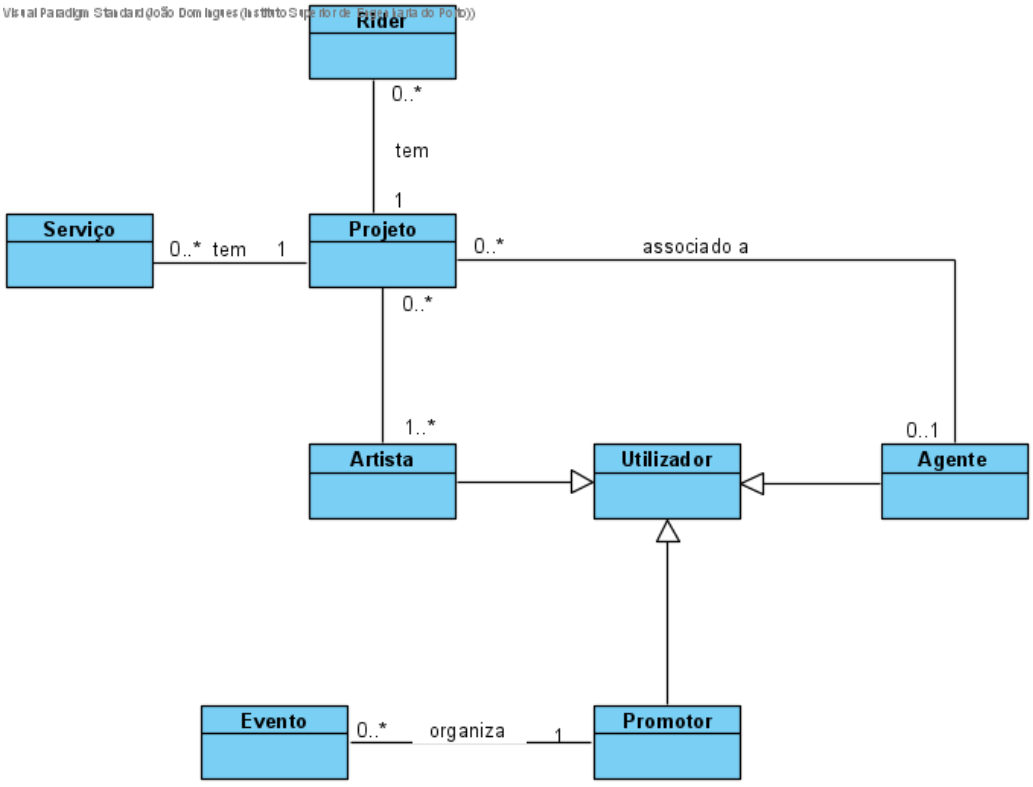


Figura 3.1: Modelo de Domínio.

É possível identificar os seguintes conceitos de domínio:

- **Utilizador** - Um utilizador genérico registado na plataforma;
- **Artista** - Um utilizador registado na plataforma, do tipo Artista;
- **Agente** - Um utilizador registado na plataforma, do tipo Agente;
- **Promotor** - Um utilizador registado na plataforma, do tipo Promotor;
- **Evento** - Ocasião organizada por um Promotor, associada a um local e uma data, aberta à participação de projetos, via candidaturas;
- **Projeto** - Projeto de um determinado género musical, composto por um conjunto de Artistas e gerido por um Agente, quando aplicável;
- **Rider** - Documento informativo destinado ao Promotor, produzido pelos Artistas de um Projeto;
- **Serviço** - Serviço disponibilizado a Promotores para contratualização.

3.2 Requisitos

Abaixo, listam-se todos os requisitos registados:

- **R01** - Os utilizadores devem poder registar-se na plataforma com uma conta de utilizador;
- **R02** - Os utilizadores devem ter acesso a um perfil pessoal editável, onde podem expor as suas informações bem como o seu material audiovisual;
- **R03** - Os utilizadores podem fazer gestão da sua rede de contactos, adicionando e removendo utilizadores da mesma;
- **R04** - Os utilizadores devem ter acesso a um sistema interno de mensagens, para poder comunicar com outros utilizadores dentro da plataforma;
- **R05** - Os utilizadores devem poder pesquisar por outros utilizadores que se encaixem nos filtros de pesquisa que aplicarem;
- **R06** - Os utilizadores devem poder ver a forma como a sua rede de contactos coincide com a de outros utilizadores;
- **R07** - Os utilizadores devem poder visualizar os conteúdos audiovisuais de outros utilizadores nas respetivas páginas;
- **R08** - Os agentes devem poder gerir a sua carteira;
- **R09** - Os agentes devem poder gerir os perfis dos seus clientes;
- **R10** - Os agentes devem poder informar os promotores da sua carteira;
- **R11** - Os agentes devem poder candidatar artistas que componham a sua carteira a eventos de promotores, através de um formulário;
- **R12** - Os agentes devem ser notificados de eventos que encaixem no perfil do trabalho dos artistas que compõem a sua carteira;
- **R13** - Os promotores devem poder gerir os seus eventos;

- **R14** - Os promotores devem ter acesso a artistas que encaixem nos seus eventos, sugeridos pelo sistema;
- **R15** - Os artistas devem poder candidatar-se a eventos de promotores, através de um formulário;
- **R16** - Os artistas devem poder gerar o seu *rider* técnico[1] para enviar aos promotores;
- **R17** - Os artistas devem poder associar-se a diferentes projetos, cada um com a sua própria página;
- **R18** - Os artistas devem poder gerar um serviço a ser comprado pelos promotores;
- **R19** - Os artistas devem ser notificados de eventos que encaixem no perfil do seu trabalho;
- **R20** - Os artistas devem poder gerir as suas próprias propostas a promotores;
- **R21** - A interface de utilizador deverá ser visualmente consistente ao longo da plataforma;
- **R22** - A plataforma deverá suportar a língua portuguesa e inglesa;
- **R23** - A plataforma deverá validar os dados inseridos pelo utilizador nos formulários;
- **R24** - O desenvolvimento da plataforma deverá seguir boas práticas de desenvolvimento de *software*;
- **R25** - A plataforma deverá estar disponível em diversos *browsers*;
- **R26** - A plataforma deverá permitir apresentar conteúdos audiovisuais provenientes de outras plataformas;
- **R27** - A plataforma deverá providenciar *feedback* visual em resposta às ações dos utilizadores;
- **R28** - A plataforma deverá permitir a notificação via *email* de eventos relevantes para os utilizadores;
- **R29** - A plataforma deverá utilizar terminologia proveniente da área de negócio.

Em primeira análise, dividir-se-ão estes requisitos em dois grupos - funcionais e não funcionais.

3.2.1 Requisitos Funcionais

Requisitos funcionais são aqueles que definem capacidades do sistema num contexto de funções executáveis pelo utilizador [67].

Dos requisitos listados acima, consideram-se requisitos funcionais os seguintes:

- **R01** - Os utilizadores devem poder registar-se na plataforma com uma conta de utilizador;

[1] Lista completa e detalhada do material que um artista vai usar numa atuação, com indicação do tipo de cabos associados a cada instrumento/material. Adicionalmente, contempla um Mapa de Palco, que demonstra a disposição dos membros em palco, numa representação gráfica simples. O objetivo de um *Rider* Técnico é dar a conhecer ao Promotor (ou equipa técnica associada) o tipo e quantidade de cablagem necessária, para efeitos de som, luz e alimentação elétrica dos instrumentos [66].

- **R02** - Os utilizadores devem ter acesso a um perfil pessoal editável, onde podem expor as suas informações bem como o seu material audiovisual;
- **R03** - Os utilizadores podem fazer gestão da sua rede de contactos, adicionando e removendo utilizadores da mesma;
- **R04** - Os utilizadores devem ter acesso a um sistema interno de mensagens, para poder comunicar com outros utilizadores dentro da plataforma;
- **R05** - Os utilizadores devem poder pesquisar por outros utilizadores que se encaixem nos filtros de pesquisa que aplicarem;
- **R07** - Os utilizadores devem poder visualizar os conteúdos audiovisuais de outros utilizadores nas respetivas páginas;
- **R08** - Os agentes devem poder gerir a sua carteira;
- **R09** - Os agentes devem poder gerir os perfis dos seus clientes;
- **R11** - Os agentes devem poder candidatar artistas que componham a sua carteira a eventos de promotores, através de um formulário;
- **R13** - Os promotores devem poder gerir os seus eventos;
- **R15** - Os artistas devem poder candidatar-se a eventos de promotores, através de um formulário;
- **R16** - Os artistas devem poder gerar o seu *rider* técnico para enviar aos promotores;
- **R17** - Os artistas devem poder associar-se a diferentes projetos, cada um com a sua própria página;
- **R18** - Os artistas devem poder gerar um serviço a ser comprado pelos promotores;
- **R20** - Os artistas devem poder gerir as suas próprias propostas a promotores.

3.2.2 Requisitos Não Funcionais

Requisitos não funcionais são aqueles que definem propriedades ou comportamentos do sistema, não estando diretamente ligados a funcionalidades específicas - "[...] um requisito de *software* que não descreve o que o *software* fará, mas sim como o fará"(tradução livre do autor) [67].

Dos requisitos listados acima, consideram-se requisitos não funcionais os seguintes:

- **R06** - Os utilizadores devem poder ver a forma como a sua rede de contactos coincide com a de outros utilizadores;
- **R10** - Os agentes devem poder informar os promotores da sua carteira;
- **R12** - Os agentes devem ser notificados de eventos que encaixem no perfil do trabalho dos artistas que compõem a sua carteira;
- **R14** - Os promotores devem ter acesso a artistas que encaixem nos seus eventos, sugeridos pelo sistema;
- **R19** - Os artistas devem ser notificados de eventos que encaixem no perfil do seu trabalho;

- **R21** - A interface de utilizador deverá ser visualmente consistente ao longo da plataforma;
- **R22** - A plataforma deverá suportar a língua portuguesa e inglesa;
- **R23** - A plataforma deverá validar os dados inseridos pelo utilizador nos formulários;
- **R24** - O desenvolvimento da plataforma deverá seguir boas práticas de desenvolvimento de *software*;
- **R25** - A plataforma deverá estar disponível em diversos *browsers*;
- **R26** - A plataforma deverá permitir apresentar conteúdos audiovisuais provenientes de outras plataformas;
- **R27** - A plataforma deverá providenciar *feedback* visual em resposta às ações dos utilizadores;
- **R28** - A plataforma deverá permitir a notificação via *email* de eventos relevantes para os utilizadores.

3.2.3 FURPS+

O FURPS+ trata-se de um modelo que permite, de forma mais extensa e detalhada, categorizar requisitos não funcionais. O FURPS+ divide-se em diferentes categorias, cada uma representada por uma das letras e o respetivo sinal "+". São elas a **F**uncionalidade, **U**sabilidade, Fiabilidade/Confiabilidade (**R***eliability*), **P**erformance e **S**uportabillidade. As características associadas ao "+"incluem restrições de design, de implementação, de interface, entre outros [68]. A análise dos requisitos não funcionais acima listados permite a seguinte divisão descrita ao longo da presente secção.

Funcionalidade

Aqui inserem-se os requisitos que representem capacidades funcionais do sistema que não estejam diretamente relacionadas com uma ação do utilizador.

Aqui inserem-se os seguintes requisitos:

- **R06** - Os utilizadores devem poder ver a forma como a sua rede de contactos coincide com a de outros utilizadores;
- **R10** - Os agentes devem poder informar os promotores da sua carteira;
- **R12** - Os agentes devem ser notificados de eventos que encaixem no perfil do trabalho dos artistas que compõem a sua carteira;
- **R14** - Os promotores devem ter acesso a artistas que encaixem nos seus eventos, sugeridos pelo sistema;
- **R19** - Os artistas devem ser notificados de eventos que encaixem no perfil do seu trabalho;
- **R28** - A plataforma deverá permitir a notificação via *email* de eventos relevantes para os utilizadores;
- **R29** - A plataforma deverá utilizar terminologia proveniente da área de negócio.

Usabilidade

Aqui inserem-se os requisitos que se baseiem na facilidade de manuseamento e de adaptação ao *software*, por parte do utilizador.

Aqui inserem-se os seguintes requisitos:

- **R21** - A interface de utilizador deverá ser visualmente consistente ao longo da plataforma;
- **R27** - A plataforma deverá providenciar *feedback* visual em resposta às ações dos utilizadores;
- **R29** - A plataforma deverá utilizar terminologia proveniente da área de negócio.

Fiabilidade

Aqui inserem-se os requisitos que se baseiem na capacidade do *software* de operar a um determinado nível performance desejado, quando exposto a um determinado conjunto de circunstâncias de utilização.

Aqui inserem-se os seguintes requisitos:

- **R23** - A plataforma deverá validar os dados inseridos pelo utilizador nos formulários.

Performance

Aqui inserem-se os requisitos que se baseiem na forma como o *software* gere os seus recursos, sejam eles tempo, energia, memória ou outros, visando estabelecer critérios de avaliação da eficiência dessa gestão. No âmbito do presente projeto, não foram identificados requisitos de performance.

Suportabilidade

Aqui inserem-se os requisitos que agrupam diversos parâmetros, entre eles a testabilidade, manutenibilidade e adaptabilidade do *software*, entre outros.

Aqui inserem-se os seguintes requisitos:

- **R24** - O desenvolvimento da plataforma deverá seguir boas práticas de desenvolvimento de *software*;
- **R25** - A plataforma deverá estar disponível em diversos *browsers*;
- **R26** - A plataforma deverá permitir apresentar conteúdos audiovisuais provenientes de outras plataformas.

Restrições (+)

Aqui são contempladas requisitos que apresentem restrições adicionais no âmbito do *design*, implementação, *interface*, entre outros.

Aqui inserem-se os seguintes requisitos:

Restrições de Interface

- **R22** - A plataforma deverá suportar a língua portuguesa e inglesa.

3.3 Casos de Uso

A partir dos requisitos listados na secção 3.2.1, formaram-se os seguintes Casos de Uso, representados na figura 3.2.

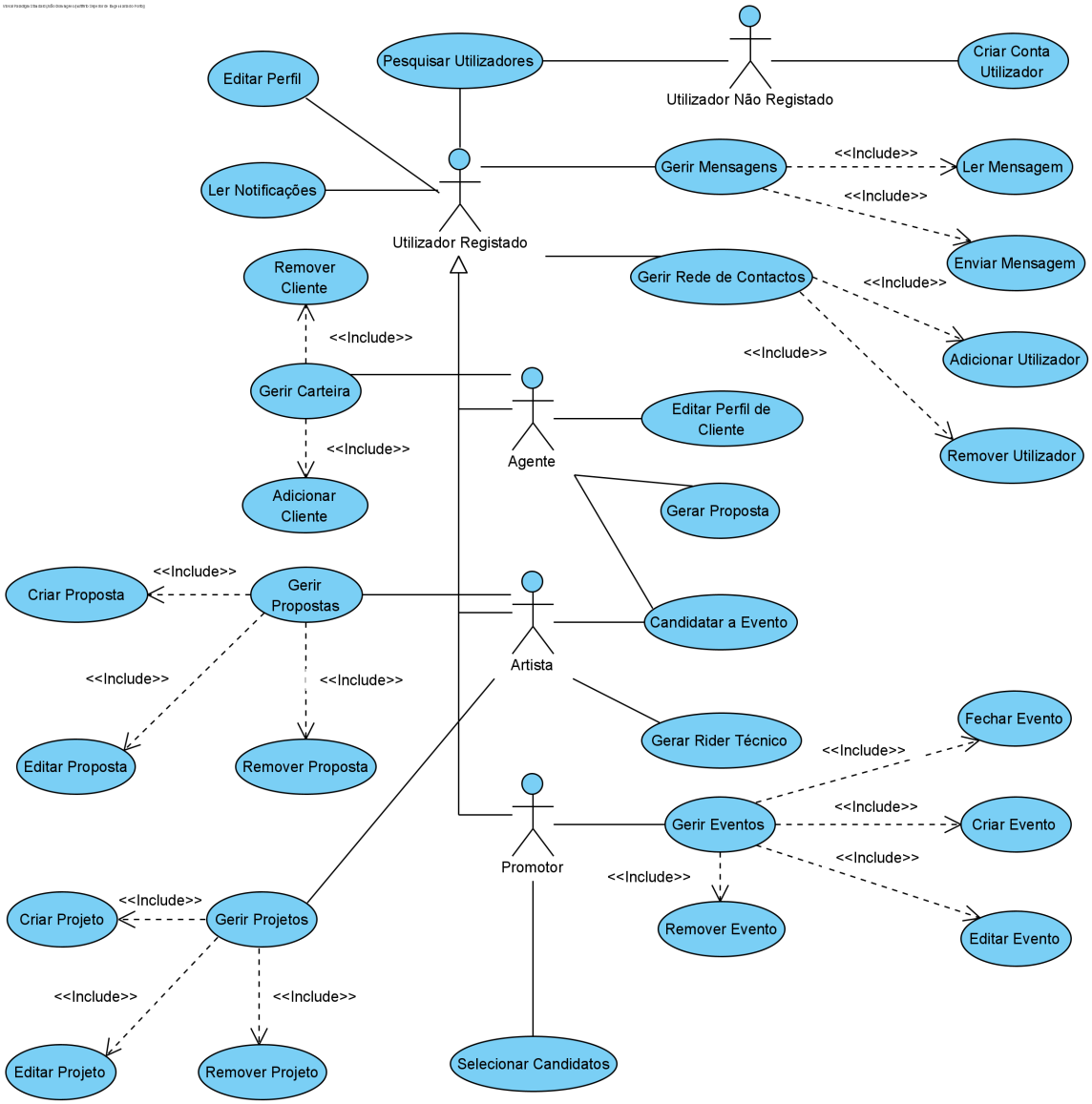


Figura 3.2: Diagrama de Casos de Uso.

- **UC01 - Criar Conta Utilizador** - Um utilizador não registado regista-se na plataforma, criando uma conta de utilizador;
- **UC02 - Pesquisar Utilizador** - Um utilizador da plataforma visualiza os utilizadores registados, aplicando filtros de pesquisa;
- **UC03 - Editar Perfil** - Um utilizador registado edita o seu perfil privado, atualizando as suas informações pessoais e/ou conteúdos audiovisuais;
- **UC04 - Ler Notificações** - Um utilizador registado lê as notificações do sistema;

- **UC05 - Ler Mensagem** - Um utilizador registado lê as mensagens de outros utilizadores;
- **UC06 - Enviar Mensagem** - Um utilizador registado envia mensagens para outros utilizadores;
- **UC07 - Adicionar Utilizador** - Um utilizador registado adiciona outro utilizador registado à sua rede de contactos;
- **UC08 - Remover Utilizador** - Um utilizador registado remove um utilizador registado da sua rede de contactos;
- **UC09 - Editar Perfil de Cliente** - Um agente edita o perfil privado de um cliente seu;
- **UC10 - Candidatar a Evento** - Um agente ou artista candidata-se ao evento de um promotor;
- **UC11 - Remover Cliente** - Um agente remove um cliente da sua carteira de artistas;
- **UC12 - Adicionar Cliente** - Um agente adiciona um cliente à sua carteira de artistas;
- **UC13 - Gerar Proposta** - Um agente gera uma proposta a partir da sua carteira para enviar a um promotor;
- **UC14 - Gerar Rider Técnico** - Um artista gera um rider técnico para associar a um projeto;
- **UC15 - Criar Proposta** - Um artista cria uma proposta de trabalho para ser contratada por um promotor;
- **UC16 - Editar Proposta** - Um artista edita uma das suas propostas criadas;
- **UC17 - Remover Proposta** - Um artista remove uma das suas propostas criadas;
- **UC18 - Criar Projeto** - Um artista cria a página de um projeto de que faça parte;
- **UC19 - Editar Projeto** - Um artista edita a página de um projeto de que faça parte;
- **UC20 - Remover Projeto** - Um artista remove um projeto que tenha criado;
- **UC21 - Criar Evento** - Um promotor cria um evento aberto aos artistas registados na plataforma;
- **UC22 - Editar Evento** - Um promotor edita um evento que tenha criado;
- **UC23 - Remover Evento** - Um promotor remove um evento que tenha criado;
- **UC24 - Fechar Evento** - Um promotor fecha um evento que tenha criado, não permitindo mais candidaturas;
- **UC25 - Selecionar Candidatos** - Um promotor seleciona de entre os candidatos a um evento.

3.4 Conclusões

Com a conclusão deste capítulo, o leitor tem agora uma melhor perceção dos requisitos em jogo e dos casos de uso a serem implementados.

Adicionalmente, tem-se agora uma noção dos conceitos de domínio a ser abordados pelo projeto e a forma como se relacionam entre si.

Por fim, estão definidas as funcionalidades a implementar e é feito o seu mapeamento com os respetivos atores.

Capítulo 4

Análise de Valor

De forma a inovar a partir da criação de um novo produto, serviço ou funcionalidade, é necessário fazer uma análise de valor. Esta análise tem em consideração fatores internos e externos da inovação e respetiva entidade que a desenvolve/idealiza, resultando daqui um valor - o valor da inovação.

A importância desta análise advém do facto de que o objetivo principal de uma iniciativa de inovação ser o acréscimo de valor para a entidade que a desenvolve ou para um potencial cliente. No entanto, a perceção do valor do resultado de uma inovação pode variar entre quem a desenvolve/idealiza e quem a consome. Então, a execução desta análise nivela potenciais discrepâncias na noção do valor da inovação entre estas duas entidades [69].

O valor da inovação resulta, internamente, da relação entre a *performance* e o custo do produto e, externamente, da necessidade e vontade do cliente, bem como a sua disposição a pagar por ele [70].

Este capítulo reflete a análise de valor à plataforma desenvolvida, bem como o respetivo processo de inovação.

4.1 Processo de Inovação

Os passos que unem a génese da ideia ao lançamento do respetivo produto no mercado formam um processo - o processo de inovação.

No seu livro *"Fuzzy front end: effective methods, tools, and techniques"*, Peter Koen [71] sugere a divisão deste processo em três fases sequenciais distintas, apresentadas na figura 4.1.

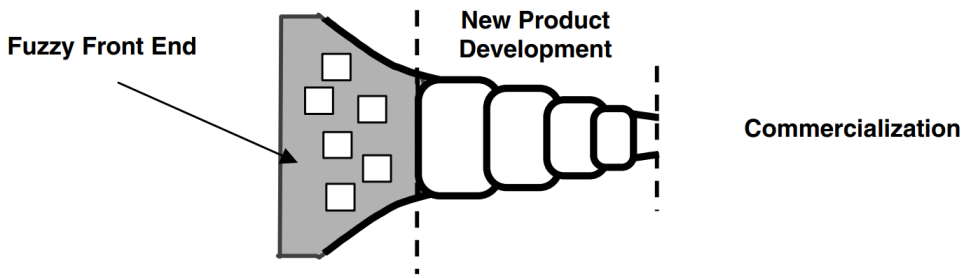


Figura 4.1: Fases do processo de inovação - Modelo de Peter Koen [72].

1. **Fuzzy Front End**, um conceito originalmente apresentado por Reinertsen e Smith em 1991 [73], que consiste na criação de um conceito com base na identificação e análise de uma oportunidade. Esta análise gera ideias que são avaliadas e selecionadas de forma a serem utilizadas na fase de desenvolvimento do produto. Em ambientes empresariais ou de equipas múltiplas, o modo de execução da fase de *Fuzzy Front End* tem implicações nas fases que o seguem. Estas implicações vêm em duas dimensões - temporal e pessoal - no sentido em que não conta apenas as decisões tomadas, como também a forma como são transmitidas para a equipa responsável pelo passo seguinte [74];
2. **New Product Development**, onde o conceito se materializa num produto, serviço ou funcionalidade e é testado, podendo, caso seja necessário, estar sujeito a alterações. Esta fase contempla ainda a análise do mercado em que se irá inserir;
3. **Comercialização**, que promove e vende o produto final ao consumidor alvo.

4.2 New Concept Development

Cada uma das fases de um processo de inovação pode ser dividida em sub-fases. Para o caso concreto do *Fuzzy Front End*, o *New Concept Development* surge com a intenção de as padronizar, tanto a nível de conceitos como de linguagem utilizada, tentando atenuar a natureza "experimental, muitas vezes caótica"(tradução livre do autor) do *Fuzzy Front End* [72].

É composto por 4 passos distintos: **identificação da oportunidade**, **análise da oportunidade**, **geração de ideias** e **seleção de ideias**.

4.2.1 Identificação da Oportunidade

Há várias variantes possíveis para a origem de uma oportunidade - uma lacuna nos produtos/serviços oferecidos no mercado, uma nova necessidade por parte do cliente ou até mesmo uma tendência geral do mercado.

No caso concreto da presente dissertação, a oportunidade identificada tem origem em três fatores principais, descritos ao longo do documento:

- O facto de não haver alternativas no mercado que reúnam um conjunto de funcionalidades que vá ao encontro de todas as necessidades dos artistas, promotores e agentes, tal como demonstrado na secção 2.2.3. Este facto é agravado no caso concreto dos agentes, que vêm uma ausência total de funcionalidades desenhadas em torno da sua função;
- Os impactos negativos causados pela pandemia *COVID-19* no mundo da arte, descritos em grande detalhe na publicação *Cadernos da Pandemia* emitida pelo Instituto de Sociologia da Universidade do Porto [6];
- No conjunto de entrevistas realizadas a profissionais da indústria da música, na secção 2.1.2, sobressair um tema em comum - a não existência de padronização nos processos de procura e oferta de eventos, baseados em *networks*.

4.2.2 Análise da Oportunidade

Identificada a oportunidade, é necessário analisá-la. Uma ferramenta passível de ser utilizada para este efeito é a **análise SWOT**.

A análise SWOT permite a análise da oportunidade, listando os seus pontos positivos/a favor e negativos/contra em dois contextos diferentes - **externo** e **interno** [75]. A combinação destas duas dimensões gera 4 parâmetros:

- **Strengths** - Definem as forças intrínsecas da ideia, fatores independentes que são a favor da sua implementação. Apresentadas na imagem 4.2 na categoria **Forças**;
- **Weaknesses** - Definem as fraquezas intrínsecas da ideia, fatores independentes que são contra da sua implementação. Apresentadas na imagem 4.2 na categoria **Fraquezas**;
- **Opportunities** - Definem as forças externas que incentivam a implementação da ideia, fatores relativos ao mercado/ambiente onde o produto resultante se vai inserir. Apresentadas na imagem 4.2 na categoria **Oportunidades**;
- **Threats** - Definem as forças externas que desincentivam a implementação da ideia, fatores relativos ao mercado/ambiente onde o produto resultante se vai inserir. Apresentadas na imagem 4.2 na categoria **Ameaças**;

A figura 4.2 representa as **Forças**, **Fraquezas**, **Oportunidades** e **Ameaças** associadas à oportunidade identificada na secção anterior.

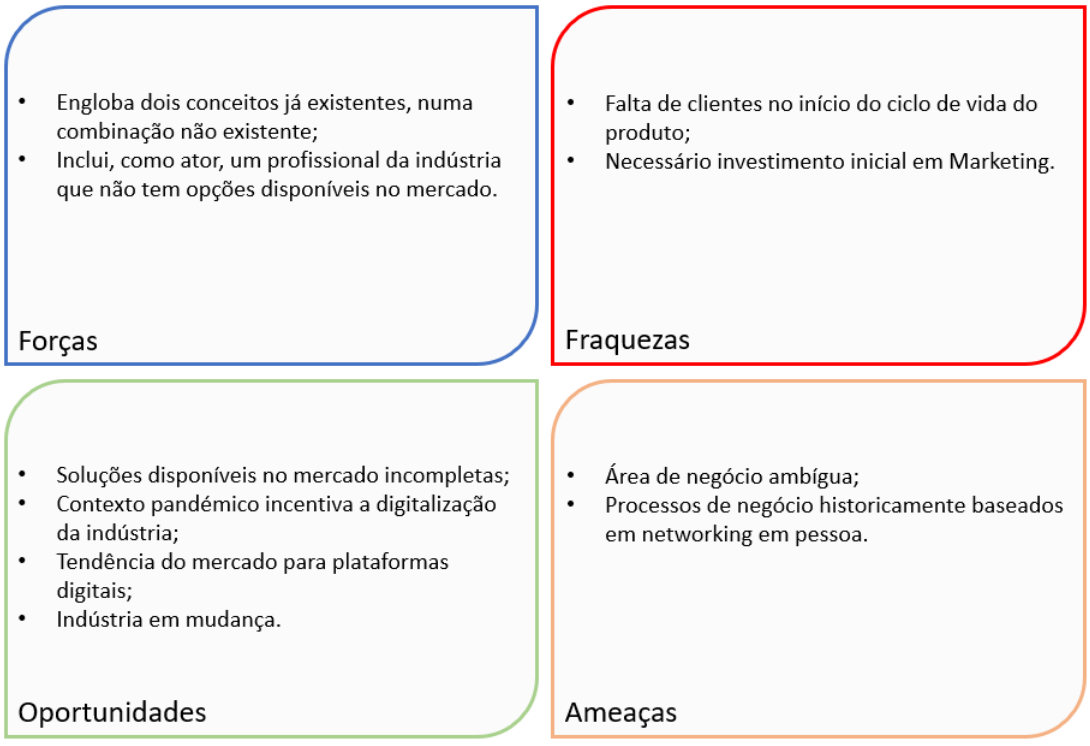


Figura 4.2: Análise SWOT da oportunidade identificada.

Forças

Engloba dois conceitos já existentes, numa combinação não existente - os dois conceitos referidos são o **networking** associado às redes sociais e as **regras de negócio** associadas às plataformas especializadas listadas na secção 2.2.1. Isto representa uma força uma vez que são duas dimensões de relevância para a indústria da música que não são partilhadas por nenhum produto existente no mercado. Tal como constatado na tabela 2.4, as plataformas especializadas existentes não têm funcionalidades relacionadas com *networking*. De igual forma, a tabela 2.5 mostra que as redes sociais não têm funcionalidades específicas para a indústria da música.

Inclui, como ator, um profissional da indústria que não tem opções disponíveis no mercado - tal como constatado nas tabelas 2.1 e 2.5, nenhuma das soluções disponíveis no mercado contempla como utilizador os agentes.

Fraquezas

Falta de clientes no início do ciclo de vida do produto - uma das vertentes da ideia debruça-se sobre a temática de *networking*, temática essa que beneficia amplamente de uma maior quantidade de utilizadores do produto. É necessário algum tempo após o seu lançamento para o mercado até se atingir uma quantidade de utilizadores aceitável.

Necessário investimento inicial em Marketing - esta fraqueza é derivada da anterior e refere que de forma a reunir uma quantidade desejável de utilizadores para o produto, o mesmo estará sujeito a um processo de *marketing*, que tem custos associados.

Oportunidades

Soluções disponíveis no mercado incompletas - tal como referenciado ao longo da secção 2.2.3, há lacunas nas funcionalidades das soluções disponíveis no mercado. Estas lacunas são mais prevalentes nas funcionalidades relativas a **sistemas de recomendação**, **ligação entre utilizadores**, funcionalidades de **criar ofertas** para promotores e agentes e de **procurar ofertas** para artistas e agentes.

As três restantes oportunidades podem ser justificadas em cadeia.

Contexto pandémico incentiva a digitalização da indústria - os impactos negativos causados pela pandemia *COVID-19* descritos ao longo da publicação *Cadernos da Pandemia* [6] incentivam a indústria a abraçar o mundo digital que, embora não seja um par novo [2], verificará uma crescente ligação.

No caso concreto da indústria da música, haverá a *tendência do mercado para plataformas digitais*, uma vez que, no caso dos artistas, estas permitem comunicar com o público através de, por exemplo, *livestreams* dos seus espetáculos [2].

Ameaças

Área de negócio ambígua - ao longo da secção 2.1.2 estão descritas as semelhanças entre o modo de trabalho na indústria da música de 3 profissionais distintos. O ponto de maior destaque é que se trata de uma área baseada em **relações humanas** e na maneira como cada profissional gere os seus contactos. Esta gestão é um processo ambíguo e está dependente de muitos fatores sociais e pessoais.

Processos de negócio historicamente baseados em networking em pessoa - a situação descrita no ponto anterior é uma constante antiga da indústria. A rede de contactos de um profissional é madurada ao longo da sua permanência na indústria, crescendo não só em tamanho como em grau de confiança entre as pessoas, tratando-se de relações interpessoais que têm muito mais impacto quando criadas/mantidas em pessoa. Por isso mesmo, nomeadamente nas camadas etárias mais velhas da indústria, pode haver alguma resistência na transição para o mundo digital [12].

4.2.3 Geração de Ideias

Após identificação dos fatores positivos e negativos, intrínsecos e extrínsecos da oportunidade identificada, segue-se a génese de ideias de potenciais modos de implementação do produto.

Nesta fase do processo, registam-se todas as ideias que, de uma forma ou de outra, possibilitariam a concretização dos objetivos estabelecidos, independentemente da exequibilidade das mesmas no contexto de desenvolvimento em que se inserem.

Adicionalmente, para no processo de geração de ideias, foram consideradas as lacunas observadas nas soluções alternativas, na secção 2.2.3.

Foram geradas as seguintes ideias:

- Desenvolvimento de uma plataforma nova, de raiz, que englobasse as funcionalidades de *networking* e das regras de negócio associadas ao projeto;
- Desenvolver uma *Application Program Interface (API)* para ser integrada com uma das **plataformas especializadas** (ver secção 2.2.2), de forma a poder integrar as funcionalidades de *networking* em falta;
- Desenvolver uma *API* para ser integrada com uma das **redes sociais** (ver secção 2.2.2), de forma a poder integrar as regras de negócio em falta.

Entre si, as ideias variam na abordagem ao desenvolvimento da solução. Desta variação, surge a necessidade de analisar cada uma à luz de um conjunto de critérios-chave, de forma a averiguar qual a melhor solução.

4.2.4 Seleção de Ideias

A avaliação das ideias geradas na secção anterior e subsequente seleção da mais indicada é feita ao analisar cada solução à luz de um conjunto de critérios-chave, associando um peso ao modo como cada solução interage com cada solução e fazendo comparações com as restantes combinações critério-solução.

De forma a atingir este objetivo, no âmbito da presente dissertação utiliza-se o método de apoio à decisão *Analytic Hierarchy Process (AHP)*, inicialmente desenvolvido por Tomas Saaty no período de 1971-1975, aquando da sua estadia na *Universidade da Pensilvânia* [76].

Este método consiste em três fases:

- **Divisão Hierárquica**, onde se identifica o **objetivo principal** a atingir com a utilização do *AHP*, definindo-se ainda os **critérios de avaliação** que são utilizados para avaliar as **ideias geradas** previamente;

- **Definição de Prioridades**, onde se hierarquiza os critérios definidos, com base na sua importância e impacto no processo de seleção de uma ideia que atinja o objetivo definido;
- **Validação de Consistência**, que averigua se os pesos atribuídos a cada critério na **Definição de Prioridades** foram tidos em conta, de forma consistente, ao longo do processo de avaliação.

Divisão Hierárquica

A divisão hierárquica vê a definição de 3 pontos chave: o **objetivo**, os **critérios** e as **soluções**.

No contexto desta dissertação, o **objetivo** de utilização do *AHP* é **escolher a melhor abordagem** de entre as soluções apresentadas.

De forma a avaliar as soluções, foram escolhidos 4 **critérios**:

- **Tempo** - Este critério diz respeito ao tempo de desenvolvimento do projeto. Tratando-se de um projeto académico, tem um prazo de desenvolvimento intrínseco que deverá ser cumprido. Considera-se que uma solução é melhor que a outra se o seu tempo de desenvolvimento for menor;
- **Manutenção** - Este critério diz respeito à manutenção requerida pelo produto após o seu desenvolvimento. Considera-se que uma solução é melhor que a outra se o seu processo de manutenção for mais simples e/ou menos volumoso;
- **Dificuldade** - Este critério **não** diz respeito à dificuldade de implementação, mas sim à dificuldade burocrática associada. Uma determinada abordagem poderá implicar entidades externas (outras empresas, associações, desenvolvedores, entre outros) que podem ou não ser acessíveis ou estar abertos a negociação da inclusão de novos requisitos que irão ao encontro dos objetivos da presente dissertação. Considera-se que uma solução é melhor que a outra se implicar um menor/mais simples processo burocrático;
- **Utilizadores** - Este critério diz respeito ao número de utilizadores disponíveis na fase de lançamento do produto. Considera-se uma solução melhor que a outra se o número de utilizadores à partida for superior.

Em conjunto com as soluções idealizadas na secção anterior, é possível agora esquematizar uma árvore hierárquica.

Por motivos de simplicidade, reduzir-se-ão as três soluções propostas à seguinte terminologia:

- Projeto pessoal;
- API Plat. Especializada;
- API Rede Social.

Na figura 4.3, apresenta-se o **objetivo** na caixa amarela (no topo), juntamente com os **critérios** nas caixas azuis (linha do meio) e as **soluções** nas caixas vermelhas (no fundo).

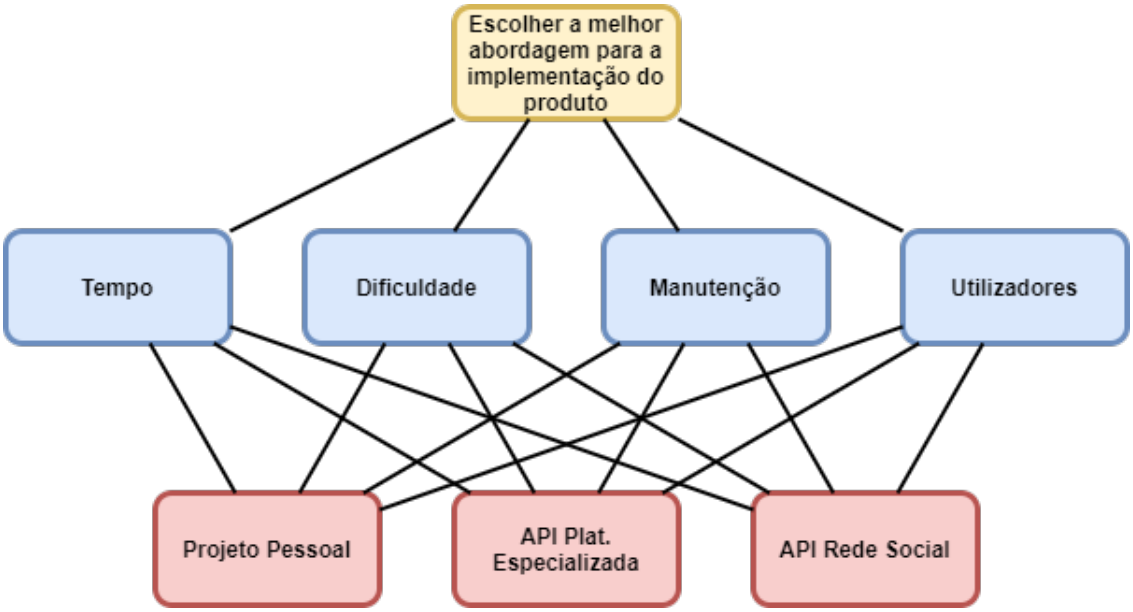


Figura 4.3: Árvore Hierárquica *AHP*.

Para efeitos de simplicidade, ao longo das secções seguintes, os critérios e soluções serão representados nas tabelas usando o esquema de associações apresentado na tabela 4.1:

Letra	Critério/Solução
A	Tempo
B	Manutenção
C	Dificuldade
D	Utilizadores
X	Projeto Pessoal
Y	API Plat. Especializada
Z	API Rede Social

Tabela 4.1: Código de letras para Critérios e Soluções.

Definição de Prioridades

A segunda fase do método *AHP* consiste em definir as prioridades de cada um dos **critérios** definidos, recorrendo, para tal, à escala referida por Saaty como *The Fundamental Scale*, apresentada na tabela 4.2 [76].

Índice de Importância	Definição
1	Mesma importância
3	Importância moderada de um sobre o outro
5	Essencial ou elevada importância
7	Importância muito elevada
9	Importância extrema
2,4,6 & 8	Valores intermédios

Tabela 4.2: *The Fundamental Scale* de Saaty [76].

Associando a cada par de critérios um **índice de importância** que defina o modo como ambos os critérios de cada par se relacionam, é possível calcular a **Prioridade Relativa** de cada critério.

	A	**B**	**C**	**D**	**Prioridade Relativa**
A	1	3	1/4	2	0.19
B	1/3	1	1/7	1/3	0.06
C	4	7	1	6	0.61
D	1/2	3	1/6	1	0.12

Tabela 4.3: Prioridades Relativas dos Critérios.

O preenchimento da tabela 4.3 evidencia que os critérios, por ordem decrescente de prioridade, são:

Dificuldade > **Tempo** > **Utilizadores** > **Manutenção**

e que:

- O critério **Utilizadores** é:
  - 3 vezes mais importante que o critério Manutenção.
- O critério **Tempo** é:
  - 2 vezes mais importante que o critério Utilizadores;
  - 3 vezes mais importante que o critério Manutenção.
- O critério **Dificuldade** é:
  - 4 vezes mais importante que o critério Tempo;
  - 6 vezes mais importante que o critério Utilizadores;
  - 7 vezes mais importante que o critério Manutenção.

Esta ordem corresponde à opinião do autor, que classifica os critérios conforme as possíveis consequências associadas:

1. **Dificuldade** é o critério mais importante, uma vez que se trata de um fator que engloba entidades externas e que, em casos extremos (ver secção 4.2.4), pode levar a que a execução do projeto não seja viável de todo;
2. **Tempo** é o segundo critério mais importante, uma vez que o tempo de desenvolvimento do projeto é importante no contexto académico da presente dissertação. As consequências de um tempo de desenvolvimento elevado não são tão graves como as possíveis consequências do critério **Dificuldade**;
3. **Utilizadores**, é o terceiro critério mais importante, derivando das necessidades de *networking* do produto. As consequências de uma solução com má pontuação no critério **Utilizadores** são sentidas diretamente pelo utilizador final, uma vez que terá em mãos um produto pouco utilizado pela comunidade;
4. **Manutenção**, é o quarto critério mais importante. Em contraste com o critério **Utilizadores**, as possíveis consequências que derivam deste critério só afetam o desenvolvedor do projeto.

Segue-se a priorização das soluções consoante cada um dos critérios. Para tal, apresenta-se uma tabela por cada critério, que contém as prioridades de cada par de soluções, bem como a prioridade relativa de cada solução.

	X	**Y**	**Z**	**Prioridade Relativa**
X	1	1/7	1/8	0.06
Y	7	1	3	0.61
Z	8	1/3	1	0.32

Tabela 4.4: Prioridades Relativas das Soluções - Tempo.

No caso do critério **Tempo**, o preenchimento da tabela 4.4 evidencia que as soluções, por ordem decrescente de prioridade, são:

API Plat. Especializada > **API Rede Social** > **Projeto Pessoal**

e que:

- A solução **API Rede Social** é:
  - 8 vezes mais importante que a solução Projeto Pessoal.
- A solução **API Plat. Especializada** é:
  - 3 vezes mais importante que a solução API Rede Social;
  - 8 vezes mais importante que a solução Projeto Pessoal.

Esta ordem corresponde à opinião do autor, que classifica as soluções da seguinte forma:

1. **API Plat. Especializada** é a solução mais importante, uma vez que a implementação das funcionalidades de *networking* poderá utilizar alguns conceitos já existentes nas próprias plataformas, tal como a existência de contas e perfis de utilizador;
2. **API Rede Social** é a segunda solução mais importante, uma vez que a implementação das regras de negócio seria feita de raiz, uma vez que as redes sociais não têm nenhuma funcionalidade relacionada, tal como demonstrado na secção 2.2.3, mas não implicaria a implementação das funcionalidades de *networking*;
3. **Projeto Pessoal** é a terceira solução mais importante, uma vez que se teria que implementar o produto na sua totalidade.

	X	**Y**	**Z**	**Prioridade Relativa**
X	1	3	5	0.65
Y	1/3	1	2	0.23
Z	1/5	1/2	1	0.12

Tabela 4.5: Prioridades Relativas das Soluções - Manutenção.

No caso do critério **Manutenção**, o preenchimento da tabela 4.5 evidencia que as soluções, por ordem decrescente de prioridade, são:

Projeto Pessoal > **API Plat. Especializada** > **API Rede Social**

e que:

- A solução **API Plat. Especializada** é:
  - 2 vezes mais importante que a solução API Rede Social.
- A solução **Projeto Pessoal** é:
  - 3 vezes mais importante que a solução API Plat. Especializada;
  - 5 vezes mais importante que a solução API Rede Social.

Esta ordem é expectável, uma vez que:

1. **Projeto Pessoal** é a solução mais importante, uma vez que, apesar de ser uma carga maior para o *developer*, este teria conhecimento profundo da *code base* na sua totalidade, bem como haveria menos dependência de serviços externos;
2. As outras duas soluções consistiriam em *code bases* mais pequenas, mas estariam dependentes das plataformas correspondentes.

	X	**Y**	**Z**	**Prioridade Relativa**
X	1	7	9	0.75
Y	1/7	1	5	0.19
Z	1/9	1/5	1	0.06

Tabela 4.6: Prioridades Relativas das Soluções - Dificuldade.

No caso do critério **Dificuldade**, o preenchimento da tabela 4.6 evidencia que as soluções, por ordem decrescente de prioridade, são:

Projeto Pessoal > **API Plat. Especializada** > **API Rede Social**

e que:

- A solução **API Plat. Especializada** é:
  - 5 vezes mais importante que a solução Rede Social.
- A solução **Projeto Pessoal** é:
  - 7 vezes mais importante que a solução API Plat. Especializada;
  - 9 vezes mais importante que a solução API Rede Social.

Esta ordem corresponde à opinião do autor, que considera que:

1. **Projeto Pessoal** é a solução mais importante, uma vez que não é necessária autorização de/negociação com entidades externas para utilização/alteração dos respetivos serviços/plataformas;
2. **API Plat. Especializada** é a segunda solução mais importante, uma vez que, na sua maioria, as **Plataformas Especializadas** na secção 2.2.2 são/foram desenvolvidas por *developers* ou pequenas equipas portuguesas. Não é certo que uma destas equipas aceite uma proposta do autor de forma a mudar a forma de funcionamento das suas plataformas;
3. **API Rede Social** é a terceira solução mais importante, estendendo a justificação dada no ponto anterior, mas aplicando a grandes empresas multinacionais com negócios estabelecidos, em contraste com as pequenas equipas portuguesas. Uma vez que as

plataformas listadas na secção 2.2.2 são grandes potências mundiais, é extremamente improvável a negociação de novas funcionalidades que vão ao encontro dos objetivos da presente dissertação.

	X	**Y**	**Z**	**Prioridade Relativa**
X	1	1/5	1/9	0.06
Y	5	1	1/5	0.22
Z	9	5	1	0.72

Tabela 4.7: Prioridades Relativas das Soluções - Utilizadores.

No caso do critério **Utilizadores**, o preenchimento da tabela 4.7 evidencia que as soluções, por ordem decrescente de prioridade, são:

API Rede Social > **API Plat. Especializada** > **Projeto Pessoal**

e que:

- A solução **API Plat. Especializada** é:
  - 5 vezes mais importante que a solução Projeto Pessoal.
- A solução **API Rede Social** é:
  - 5 vezes mais importante que a solução API Plat. Especializada;
  - 9 vezes mais importante que a solução Projeto Pessoal.

Esta ordem corresponde à opinião do autor, que considera que:

1. **API Rede Social** é a solução mais importante, uma vez que se tratam de plataformas com uma quantidade de utilizadores incontestavelmente maior do que as restantes soluções [39];
2. **API Plat. Especializada** é a segunda solução mais importante, uma vez que, embora não detenham uma quantidade de utilizadores tão grande como uma rede social, já têm alguns utilizadores registados;
3. **Projeto Pessoal** é a terceira solução mais importante, uma vez que a base de utilizadores começaria do zero, sem qualquer utilizador registado, tal como descrito na figura 4.2.

Uma vez concluído o processo de priorizar as soluções consoante cada um dos critérios, o próximo passo consiste em agregar toda a informação recolhida até então, de forma a determinar a melhor **solução**, considerando todos os **critérios**, de forma a atingir o **objetivo**.

Na tabela 4.8, encontram-se as prioridades relativas de cada solução, em relação a cada critério, bem como as prioridades relativas de cada critério. A matriz referente à prioridade relativa final de cada solução está apresentada na coluna da direita, tendo sido calculada a partir da multiplicação da matriz das prioridades das soluções, com a matriz de prioridades dos critérios, representada na última linha da tabela.

	A	B	C	D	Prioridade Relativa
X	0.06	0.65	0.75	0.06	**0.52**
Y	0.61	0.23	0.19	0.22	0.28
Z	0.32	0.12	0.06	0.72	0.20
Prioridade Relativa	0.19	0.06	0.62	0.13	

Tabela 4.8: Prioridades Relativas das Soluções.

Em primeira análise ao resultado da tabela 4.8, conclui-se que a melhor abordagem é a implementação de um produto de raiz. No entanto, é necessário averiguar a consistência das prioridades ao longo desta implementação do *AHP*.

Validação de Consistência

De forma a validar a consistência das prioridades ao longo da secção anterior, nesta secção será calculado o *Índice de Consistência (IC)* e o *Rácio de Consistência (RC)*. Esta consistência de prioridades traduz-se numa consistência do julgamento das soluções e critérios ao longo da **definição de prioridades**.

Caso exista consistência de julgamento, é possível verificar a Lei da Transitividade nas prioridades atribuídas entre critérios. Por exemplo, se o critério A é mais importante que o critério B e o critério B é mais importante que o critério C, significa que o critério A é mais importante que o critério C.

Assim, o cálculo do *IC* dá-se a partir da fórmula 4.1:

$$IC = \frac{\lambda max - n}{n - 1} \tag{4.1}$$

com n a corresponder ao número de critérios utilizado.

O valor de λmax recorre ao maior valor próprio da matriz resultante do cálculo de consistência (apresentada na tabela 4.9), bem como à matriz de prioridade relativa dos critérios, disponível na tabela 4.3.

A	0.19
B	0.06
C	0.62
D	0.13

Tabela 4.9: Matriz de Consistência.

O valor de λmax é então obtido a partir da formula 4.2:

$$\lambda max = \frac{\frac{0.79}{0.19} + \frac{0.26}{0.06} + \frac{2.60}{0.62} + \frac{0.52}{0.13}}{4} = 4.11 \tag{4.2}$$

Retornando à fórmula do *IC*, substituindo agora pelos valores, obtém-se o valor apresentado no cálculo 4.3:

$$IC = \frac{4.11 - 4}{4 - 1} = 0.04 \tag{4.3}$$

No que toca ao cálculo do *RC*, este recorre ao *IC* e ao *Índice Aleatório (IR)*, sendo feito segundo a fórmula 4.4:

$$RC = \frac{IC}{IR} \tag{4.4}$$

O *IR* trata-se de um valor tabelado [76]. A tabela proposta por Saaty apresenta-se na figura 4.4. O valor de *IR* a usar é o que corresponde ao *n* a ser utilizado nesta implementação do *AHP* - 4.

n	1	2	3	4	5	6	7	8	9	10
Random consistency index (R.I.)	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49

Figura 4.4: Tabela para obtenção do *IR*.

O valor de IR é, então, 0.90, tornando agora possível o cálculo do *RC*, como demonstrado no cálculo 4.5:

$$RC = \frac{0.04}{0.90} \approx 0.04 \tag{4.5}$$

Na sua proposta, Saaty sugere a inclusão de uma margem de inconsistência de 10% (0.10), uma vez que "[...] sem inconsistência, não se pode admitir novo conhecimento que altere a ordem de preferências."(tradução livre do autor) [76]. Para a implementação do *AHP* ser válida, o *RC* deverá ser menor que a margem de inconsistência.

Uma vez que esta condição se verifica, pode-se concluir a validade das conclusões obtidas com o *AHP*, fazendo com que a melhor abordagem seja a implementação de um produto de raiz.

4.3 Valor da Solução

Do processo descrito ao longo da secção 4.2, resulta uma possível solução para a realização do projeto em causa. No entanto, é ainda necessário averiguar se e que valor é que o cliente poderá retirar do produto a desenvolver, fazendo uma análise das **necessidades do cliente**.

Adicionalmente, a partir da definição das necessidades do cliente, estabelecer-se-á uma ligação entre estas necessidades e as características de qualidade do sistema, recorrendo para isso ao método *Quality Function Deployment (QFD)*.

Primeiro, e de modo a determinar o valor da solução para o cliente, definir-se-á a terminologia utilizada ao longo da secção seguinte.

4.3.1 Valor

A noção de **valor** deriva de mais fatores além do produto/serviço em causa, tratando-se do *trade-off* existente entre os custos e benefícios associados a toda a relação que se estabelece entre o cliente e o fornecedor/produtor neste processo. Da mesma forma que os benefícios englobam mais do que o próprio produto/serviço, os custos não se cingem ao valor monetário associado ao mesmo [77]. No caso do **valor**, os benefícios associados a esta relação são classificados como *satisfação do cliente* [78].

Esta satisfação é formada por fatores monetários e não-monetários, que podem contribuir, emocionalmente, quer de forma positiva como de forma negativa para o **valor para o cliente**. Fatores não-monetários positivos podem incluir a qualidade do produto, o serviço ao cliente ou até mesmo a posição do fornecedor/produtor no mercado, enquanto que os negativos podem incluir o esforço, tempo e energia necessários para obtenção do produto/serviço [77].

Em contraste com a ligação emocional que gera o **valor para o cliente**, a ligação cognitiva gera o **valor percecionado**. Na indústria de *software*, variáveis como, por exemplo, as necessidades técnicas para utilização do produto, influenciam o **valor percecionado** [77].

O **valor percecionado**, que deriva do **valor** do produto, tem impacto no **valor para o cliente** e ditará as intenções do cliente na relação criada com o produtor/fornecedor [78].

4.3.2 Proposta de Valor

De forma a comunicar o valor de um produto ao cliente, é necessário apresentar uma proposta de valor, onde se faz uma exposição convidativa, ao evidenciar os benefícios associados à compra (ou equivalente) do produto.

No contexto da presente dissertação e respetivo produto associado, é feita uma proposta de valor em dois formatos distintos - **Business Model Canvas** e **Value Proposition Canvas**.

Business Model Canvas

O *Business Model Canvas*, proposto por Alexander Osterwalder em 2004 [79], permite representar os aspetos relativos ao desenvolvimento do produto associado à presente dissertação, expondo, por um lado, os benefícios do negócio para os clientes e a equipa de desenvolvimento e, por outro, detalhando o modo de funcionamento do negócio.

<table>
<tr><td rowspan="2">Parceiros Chave:

---</td><td>Atividades Chave:

- Análise das necessidades dos utilizadores;

- Design, análise e implementação de uma solução;</td><td rowspan="2">Proposta de Valor:

- Exposição do trabalho artístico;

- Expansão e gestão da rede de contactos com várias entidades que desempenhem um de três papéis na indústria;

- Simplicidade de utilização;

- Recomendação de contactos para contrato;

- Variedade de funcionalidades disponíveis numa única plataforma;</td><td>Relação com Clientes:

- Autonomia por parte dos utilizadores;</td><td rowspan="2">Segmentos de Clientes

- Profissionais da indústria da música;

- Participantes na indústria por hobby;</td></tr>
<tr><td>Recursos Chave:

- Relato das necessidades dos utilizadores;

- Análise das soluções disponíveis no mercado;

- Conhecimento técnico de web development;</td><td>Canais:

- Disponível online;</td></tr>
<tr><td colspan="2">Custos:

- Marketing;</td><td colspan="3">Fontes de Retorno:

---</td></tr>
</table>

Figura 4.5: *Business Model Canvas* associado ao produto a desenvolver.

Na figura 4.5, apresenta-se o *Business Model Canvas* referente ao produto a ser desenvolvido ao longo da presente dissertação. Aqui, podem ser encontrados os seguintes parâmetros:

Parceiros Chave - Aqui são listados os parceiros externos ligados à entidade que desenvolve o produto. No contexto deste projeto, como é desenvolvida por um único *developer* independente, não há parceiros associados.

Atividades Chave - Aqui listam-se as atividades que levam à génese do produto. No contexto deste projeto, listaram-se a análise inicial às necessidades dos utilizadores, realizada na secção 2.1.1, bem como o design, análise e implementação do próprio produto.

Recursos Chave - Aqui listam-se os recursos necessários à execução do projeto. No contexto deste projeto, listaram-se os relatos das necessidades dos utilizadores disponibilizados na secção 2.1.1, a análise das soluções disponíveis no mercado feita na secção 2.2.3 e conhecimentos técnicos de *web development*, uma vez que se trata de uma plataforma *web*.

Proposta de Valor - Aqui faz-se a proposta de valor, onde se listam os motivos pelos quais o cliente quererá avançar com o negócio, apresentando as razões para escolher este produto em vez de outras alternativas do mercado. No contexto deste projeto, estão reunidas funcionalidades de diferentes naturezas (*networking* e *negócio da indústria da música*) numa só plataforma, providenciando ainda um meio de partilha do trabalho artístico do utilizador. Adicionalmente, o produto oferece simplicidade de utilização e a oportunidade de expandir e gerir a rede de contactos do utilizador.

Relação com Clientes - Aqui detalha-se o modo como a organização se relacionará com os clientes e que tipo de suporte lhes será dado. No contexto deste projeto, o cliente será autónomo na utilização da plataforma.

Canais - Aqui listam-se os canais que levarão o produto ao cliente. No contexto deste projeto, o cliente terá acesso *online* ao produto.

Segmentos de Clientes - Aqui descrevem-se os tipos de clientes que se espera que utilizem o produto. São eles os profissionais da indústria da música, nomeadamente, aqueles referidos na secção 2.1.2.

Custos - Aqui listam-se os custos inerentes à implementação da solução. Tal como apresentado na figura 4.2, o projeto terá custos associados a *marketing*, com o objetivo de angariar utilizadores para a plataforma.

Fontes de Retorno - Aqui listam-se as fontes de retorno do produto. No âmbito deste projeto não foram idealizadas fontes de retorno.

4.3.3 Necessidades do Cliente

As necessidades do cliente são um dos guias principais do rumo que um produto irá tomar - a satisfação destas é, de uma forma geral, o objetivo principal de qualquer produto, independentemente da sua natureza [80].

Por isso mesmo, a inclusão dessas necessidades nas fases iniciais do ciclo de vida de um produto é essencial, nomeadamente, nas fases de planeamento e *design*. A utilidade desta inclusão surge quando se definem os critérios de qualidade de um produto, uma vez que podem ser diretamente derivados das necessidades do cliente [81].

Ao utilizar metodologias como o *QFD*, é possível "[...] traduzir as necessidades do cliente para conceitos do produto. Os requisitos de *design* servem como *input* de forma a estabelecer a componente de características do produto"(tradução livre do autor) [81].

4.3.4 Quality Function Deployment

O *QFD* é uma metodologia multi-faseada com origem na indústria automóvel japonesa, mais concretamente, na *Mitsubishi Heavy Industries Ltd*, no final da década de 60, com Yoji Akao como autor [82]. Foi, mais tarde, implementada pela primeira vez nessa mesma empresa, em 1972, por Kobe Shipyard [83].

O *QFD* trata-se de uma metodologia que pode ser utilizada em diversas fases do ciclo de vida de um projeto, permitindo a equipas polivalentes fazer *trade-offs* entre aquilo que o cliente pretende e aquilo que o produto oferece(rá), permitindo também fazer um *benchmarking* aos produtos concorrentes e estabelecer correlação entre conceitos do produto [81].

O *QFD* conta com 4 fases distintas:

- **Strategy and Concept Definition** - transforma *Requisitos do Cliente* em *Características de Engenharia*;
- **Product Design** - transforma *Características de Engenharia* em *Parâmetro de Design do Produto*;
- **Process Design** - transforma *Parâmetros de Design do Produto* em *Parâmetros de Design do Processo*;
- **Manufacturing Operations** - transforma *Parâmetros de Design do Processo* em *Parâmetros do Processo de Desenvolvimento*;

No âmbito desta dissertação, a fase que mais apropriada para ser utilizada, é a primeira - **Strategy and Concept Definition**.

Para tal, tem que se definir os seguintes parâmetros:

- Necessidades do Cliente;
- Importância das necessidades do Cliente;
- Características de Engenharia;
- Escalas de Correlação;
- Concorrentes.

Necessidades do Cliente

No contexto da presente dissertação, as necessidades do cliente derivam de testemunhos de profissionais da indústria da música, nomeadamente, dos 3 candidatos entrevistados na secção 2.1.1 - abaixo estão listadas necessidades que os testemunhos destacaram ao longo dos processos de entrevistas.

- **Páginas pessoais** - O produto deverá providenciar a cada utilizador uma página pessoal na plataforma, editável;
- **Facilidade de utilização** - Uma vez que se tratam de utilizadores que não detém, obrigatoriamente, conhecimentos técnicos, o produto deverá ser acessível na sua utilização;
- **Comunicação interna com outros utilizadores** - Deverá ser possível estabelecer comunicação com outros utilizadores através de canais embutidos no produto;
- **Gestão da rede de contactos** - O produto deverá permitir gerir as ligações que têm para com outros utilizadores;
- **Aplicação de regras de negócio** - O produto deverá conter um conjunto de funcionalidades que facilitem alguns processos descritos ao longo da secção 2.1.2, como por exemplo, a gestão de utilizadores externos ou o contacto com finalidade de contratação;
- **Recomendação de utilizadores** - O produto deverá recomendar soluções que encaixem nos critérios de oferta e procura dos utilizadores.

A cada uma destas necessidades é atribuído um índice de importância, numa escala de 1 a 5, onde 5 corresponde ao grau mais importante. Adicionalmente, as importâncias foram normalizadas. Os resultados apresentam-se na tabela 4.10.

	Índice de Importância	**Importância Normalizada (%)**
Páginas Pessoais	5	23.7
Facilidade de Utilização	3	13.0
Comunicação Interna	4	17.4
Gestão da Rede de Contactos	4	17.4
Regras de Negócio	5	23.7
Recomendação de Utilizadores	2	8.7

Tabela 4.10: Necessidades do Cliente e respetivas Importâncias.

Características de Engenharia

Num processo semelhante ao da secção anterior, nesta secção listam-se as características de engenharia que visam responder às necessidades do cliente.

- **Interface Gráfica User-Friendly** - Característica associada à facilidade de utilização da plataforma e de interação com a mesma;
- **Conteúdo Audiovisual** - Característica associada ao suporte de diferentes tipos de conteúdos audiovisuais (vídeo, áudio, entre outros);
- **Login a partir de plataformas externas** - Característica associada à capacidade de registo na plataforma a partir de serviços externos;
- **Boa Usabilidade** - Característica associada à capacidade de o utilizador usar o produto de forma a atingir os seus objetivos de forma eficaz e eficiente;
- **Workflow Intuitivo** - Característica associada à facilidade de o utilizador compreender o funcionamento das regras de negocio associadas ao produto sem necessidade de formação prévia/adicional.

Escalas de Correlação

Com as duas dimensões **Necessidades do Cliente** e **Características de Engenharia** definidas, é preciso criar uma escala que possa ser utilizada para representar as relações entre os elementos de cada uma delas.

Adicionalmente, é necessário ainda estabelecer uma escala que possa ser utilizada para representar a correlação entre diferentes **Características de Engenharia**.

Nas tabelas representadas nas figuras 4.6 e 4.7, apresenta-se a simbologia que será utilizada no *QFD*.

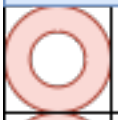
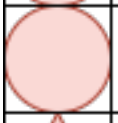
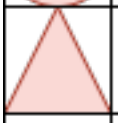
Código de Relação		
	Forte	9
	Médio	3
	Fraca	1
	Sem Relação	0

Figura 4.6: Escala de relação entre Características de Engenharia do QFD.

Código de Correlação	
++	Muito Forte
+	Forte
-	Fraca
--	Muito Fraca
	Sem Relação

Figura 4.7: Escala de correlação entre Características de Engenharia do QFD.

Concorrentes

Adicionalmente, as **necessidades do cliente** já definidas serão ainda avaliadas em produtos concorrentes, utilizando uma escala de 1 a 5, em que 5 é a melhor classificação. É importante frisar dois pontos desta avaliação e da escolha de concorrentes:

- Idealmente, para o *QFD*, os concorrentes tidos em causa são soluções que conseguem responder às necessidades do cliente definidas. No entanto, e como já descrito ao longo da secção 2.2.3, não existe uma solução que satisfaça todas necessidades do cliente. Como resultado disso, o autor decidiu escolher a melhor solução das categorias **Redes Sociais** e **Plataformas Especializadas**, bem como uma outra solução aleatória da categoria **Plataformas Especializadas**;
- A avaliação é feita pelo próprio autor, de acordo com a experiência de utilização de cada plataforma.

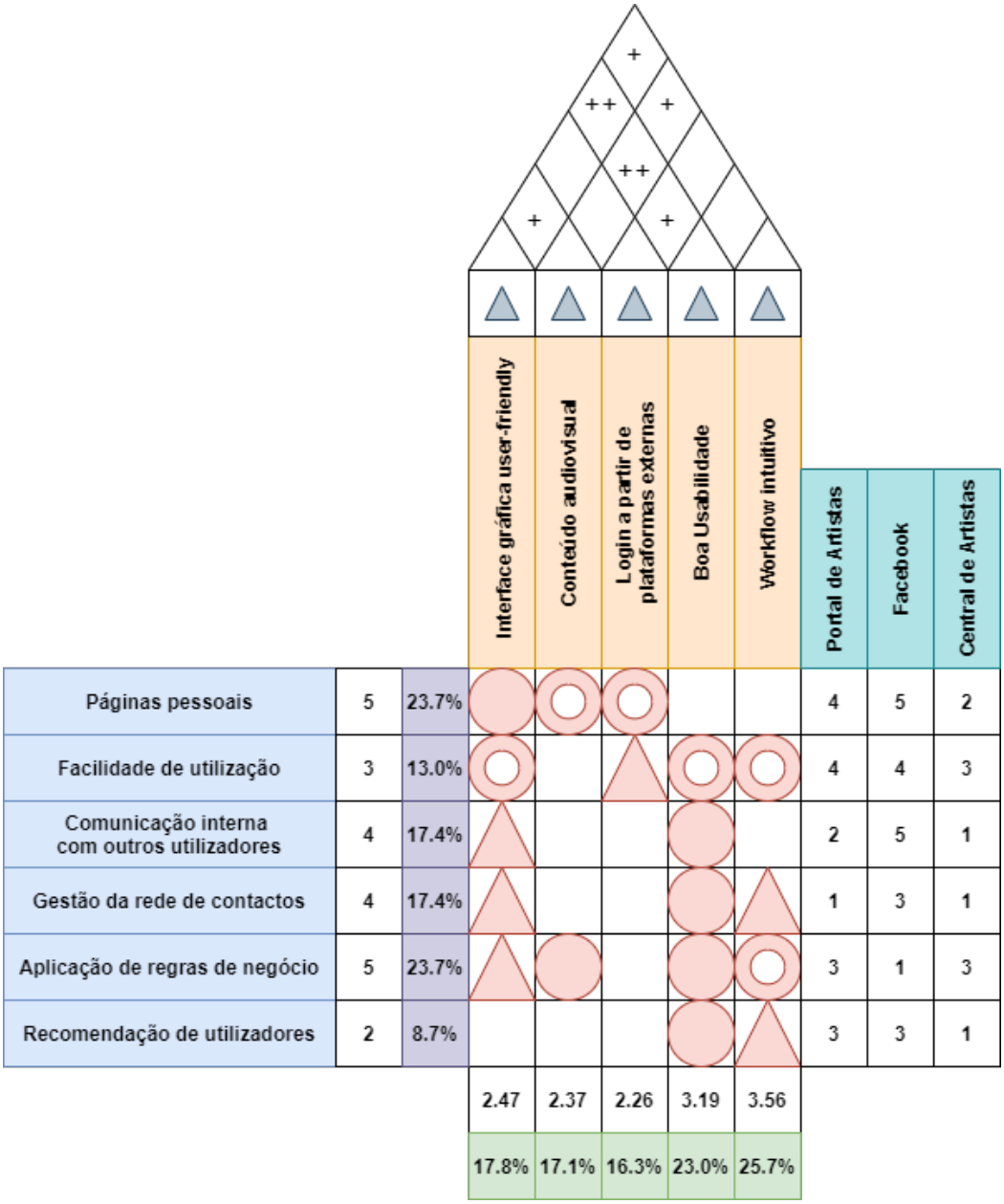


Figura 4.8: *Quality Function Deployment.*

Na figura 4.8 apresenta-se a forma final do *QFD*, desta podem retirar-se as seguintes conclusões:

- O **Workflow Intuitivo** é a característica de engenharia que causa mais impacto na procura pela satisfação das necessidades dos clientes, enquanto que o **Login a partir de plataformas externas** é o menos importante;
- Não há correlações negativas entre características de engenharia;
- As correlações mais fortes entre características de engenharia encontram-se nos pares **Interface gráfica user-friendly/Boa Usabilidade** e **Conteúdo Audiovisual/Boa Usabilidade**.

4.4 Conclusões

Com a conclusão deste capítulo, o leitor tem agora uma melhor perceção do valor associado à proposta apresentada. Aqui, são abordadas várias ideias tidas em consideração que têm potencial para abordar a oportunidade identificada e respetivos objetivos traçados. A partir das ideias geradas, foi feita uma seleção da melhor opção para o desenvolvimento de um projeto no contexto da presente dissertação, através da utilização do método *AHP*. A partir deste processo concluiu-se que a melhor opção é o desenvolvimento de uma plataforma de raiz.

Por fim, com a solução já selecionada, é feita uma análise do respetivo valor, através do método *QFD*, onde são definidas as característas de engenharia consideradas e é calculado a importância associada a cada uma. Deste processo conclui-se que:

- O **Workflow Intuitivo** é a característica de engenharia que causa mais impacto na procura pela satisfação das necessidades dos clientes, enquanto que o **Login a partir de plataformas externas** é o menos importante;
- Não há correlações negativas entre características de engenharia;
- As correlações mais fortes entre características de engenharia encontram-se nos pares **Interface gráfica user-friendly/Boa Usabilidade** e **Conteúdo Audiovisual/Boa Usabilidade**.

Capítulo 5

Desenho da Solução

De forma a que a fase de desenvolvimento do projeto decorra de forma eficiente e eficaz, procede-se anteriormente a um processo de planeamento, onde são tomadas decisões referentes à arquitetura do *software*. Daqui, resultam artefactos relevantes de um ponto de vista de engenharia de *software* que, em conjunto, representam ao detalhe o produto a ser desenvolvido.

É ainda importante referir e ter em consideração que as tecnologias selecionadas na secção 2.3 têm impacto nas decisões tomadas no presente capítulo.

5.1 Arquitetura

O desenvolvimento de *software* com base numa arquitetura visa ter como consequência, a partir de curto-médio prazo, uma diminuição no tempo necessário para implementar novas funcionalidades, bem como um processo de manutenção mais fácil e intuitivo [84]. Com o intuito de tomar a melhor decisão possível, é importante iterar por várias soluções de arquitetura possíveis.

Ao longo desta secção, é apresentada a evolução da arquitetura do projeto *ArPA*, onde cada passo e decisão tomada são justificados em conjunto com as alternativas tidas em consideração.

5.1.1 Frontend e Backend

Numa primeira fase, é tido em consideração a forma como interagem o *frontend* e o *backend* da aplicação e a forma como estarão dispostos arquiteturalmente. Consideram-se duas possíveis soluções.

Alternativa A

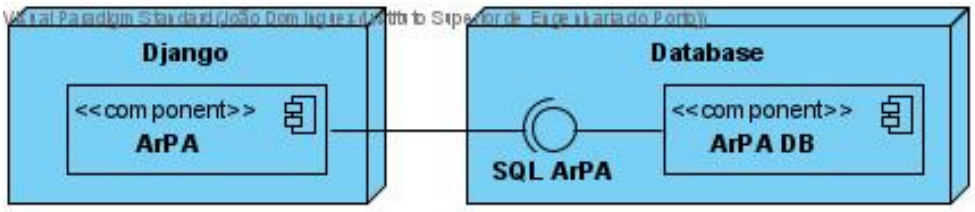


Figura 5.1: Diagrama de Implantação da Alternativa A.

A figura 5.1 representa a primeira alternativa tida em consideração, na qual o *frontend* e *backend* se encontram concentrados no mesmo monolito. Este tipo de abordagem é compatível com o conjunto de tecnologias selecionado, através, por exemplo, da utilização da linguagem de *templating* da *framework Django* [85].

Alternativa B

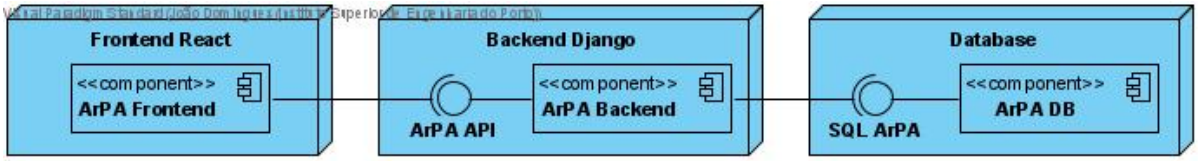


Figura 5.2: Diagrama de Implantação da Alternativa B.

A figura 5.2 representa a segunda alternativa tida em consideração. Aqui, há uma distribuição de responsabilidades dentro do sistema. Agora, o *frontend* separa-se do *backend* em dois componentes distintos, cada um deles dedicado exclusivamente a uma função.

Seleção de Alternativas

No seu livro [86] e blog [87], *Robert C. Martin* define *Single Responsibility Principle* como a prática de separação de responsabilidades por diferentes módulos ou componentes de *software*. Uma vez que há separação da aplicação em pelo menos dois componentes, onde um fica responsável pelo *frontend* e o outro pelo *backend*, está-se a promover este princípio. Por isso mesmo, foi selecionada a **Alternativa B**.

5.1.2 Backend

Com a separação da aplicação em dois componentes, o próximo passo é analisar cada um destes componentes e determinar a sua estrutura, começando pelo *backend*.

Alternativa A

A primeira solução possível para a implementação do *backend* é a implementação de um **monólito**, em semelhança ao que já se encontra representado na figura 5.2.

Um monólito engloba todas as funcionalidades da aplicação num único componente - uma única unidade. Esta singularidade traz algumas vantagens [88], nomeadamente:

- Um processo inicial de desenvolvimento mais simples;
- Processo de teste do *software* mais simples;
- A implantação de um monólito é um processo mais simples do que as restantes alternativas.

Alternativa B

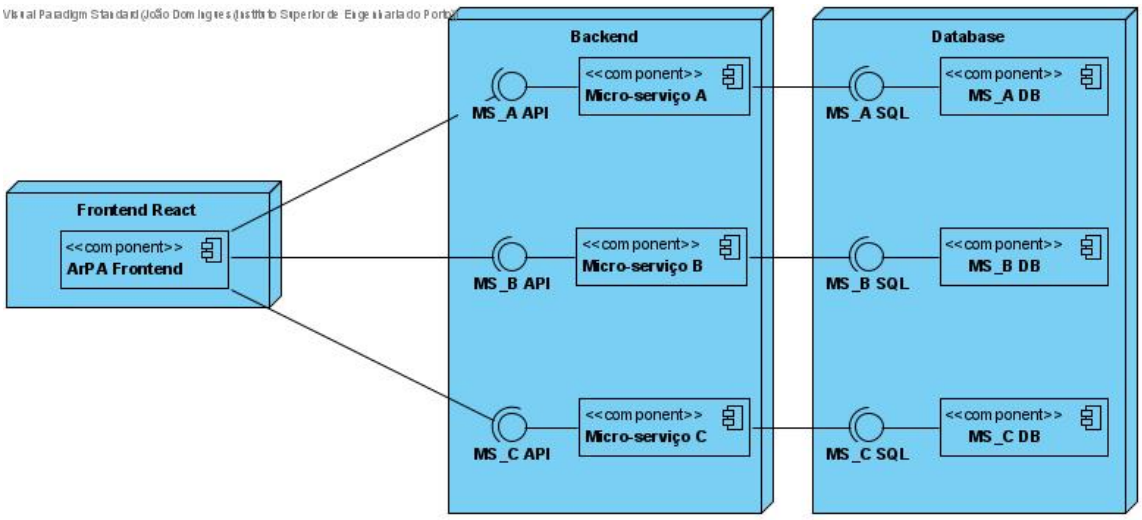


Figura 5.3: Diagrama de Implantação da Alternativa B.

A base da segunda alternativa são os **micro-serviços**, que separam um monólito em diversos componentes independentes e independentemente implantáveis e implementáveis [89].

No caso dos micro-serviços, esta separação do monólito em componentes mais pequenos, cada um com a sua responsabilidade, tem as seguintes vantagens [88, 89], nomeadamente:

- A separação de um monólito em diversos micro-serviços através do delinear de diferentes capacidades de negócio promove a reusabilidade de um determinado micro-serviço;
- Uma vez que cada micro-serviço pode ser testado de forma independente e tem responsabilidades reduzidas, o processo de testes ao *software* é mais eficaz (ainda que mais complexo), havendo um menor custo de garantia de qualidade;
- O facto de cada micro-serviço ser implantado de forma independente implica que uma alteração a um determinado micro-serviço não necessita da implantação total do sistema;

Seleção de Alternativas

A abordagem tomada nesta fase do processo de *design* da arquitetura baseou-se na ideia apresentada por *Martin Fowler* denominada de *Monolith First* [90]. Aqui, *Martin Fowler* argumenta que o sucesso de uma arquitetura em micro-serviços resulta de forma mais comum a partir de casos em que a implementação inicial foi feita num monólito que, posteriormente, conforme a análise das diferentes capacidades de negócio desse mesmo monólito, foi dividido em diversos micro-serviços.

Adicionalmente, e em contraste, *Fowler*, que defende a ideia da divisão de monólitos em micro-serviços, argumenta que normalmente casos que são desenhados de raiz com uma arquitetura de micro-serviços resultam em *software* confuso que, a médio-longo prazo, herda as desvantagens associadas a monólitos. *Fowler* atribui este fenómeno a fatores como erros na fase de *design* da arquitetura, onde se faz uma má avaliação daquelas que são as capacidades de negócio da aplicação.

Deste modo, procede-se à análise das capacidades de negócio do monólito apresentado inicialmente. Daqui identificam-se três capacidades de negócio distintas:

- **Sistema de Recomendações** - A aplicação inclui um Sistema de Recomendações que apoiará o poder de decisão dos utilizadores dentro da plataforma. Este sistema serve-se dos dados guardados pela aplicação, fazendo operações com base nesses mesmos dados;
- **Gestor de Dados Multimédia** - Existe, no entanto, um sub-conjunto de dados que têm duas características em comum - os dados multimédia. Este tipo de dados destaca-se porque não é contemplado pelo Sistema de Recomendações e implica uma carga no sistema muito superior a qualquer outro tipo de dados, uma vez que se lida com ficheiros multimédia;
- **Gestão de Informação** - A aplicação lida com informação dos utilizadores. Aqui aplicam-se as operações *Create, Read, Update e Delete (CRUD)* comuns, com as devidas regras de negócio aplicáveis.

Embora à primeira vista pareça que se formam 3 micro-serviços, é preciso analisar em maior detalhe cada um deles e a forma como eles interagem entre si.

No caso da capacidade de negócio **Gestão de Informação**, uma vez que agrega quase todos os conceitos de negócio e respetivas regras de negócio, não se classifica como micro-serviço, permanecendo um monólito.

O **Sistema de Recomendações** é um potencial candidato a formar um micro-serviço. No entanto, serve-se exclusiva e inteiramente dos conceitos de negócio guardados no sistema de gestão de informação. Isto traduz-se no facto de que sempre que há alguma alteração à estrutura de dados no monólito do gestor de informação, o sistema de recomendações terá que ser alterado também.

Um ponto a favor de converter o **Sistema de Recomendações** num micro-serviço é o facto de a operação de gerar recomendações poder ser custosa e criar uma carga desnecessária no sistema. No entanto, na opinião do autor, o facto de o Sistema de Recomendações estar dependente do Gestor de Informação, retira-lhe qualquer valor como micro-serviço.

Em contraste, o **Gestor de Dados Multimédia** é completamente independente das outras capacidades de negócio. A acrescentar a este facto, trata-se de uma capacidade de negócio com um custo de performance extremamente elevado. Por isso mesmo, neste caso concreto, é justificada a criação de um micro-serviço.

Uma vez que há pontos a favor da separação do monólito inicial em micro-serviços e pontos a favor de o manter como está representado na figura 5.2, o autor optou por criar uma terceira alternativa.

Alternativa C

A terceira alternativa consiste numa arquitetura híbrida, que agrega um monólito e um micro-serviço. Na opinião do autor, é a melhor opção, uma vez que junta a melhor parte de cada uma das alternativas **A** e **B**.

Na figura 5.4 representa-se a arquitetura nesta fase do processo, respondendo à **Alternativa C**.

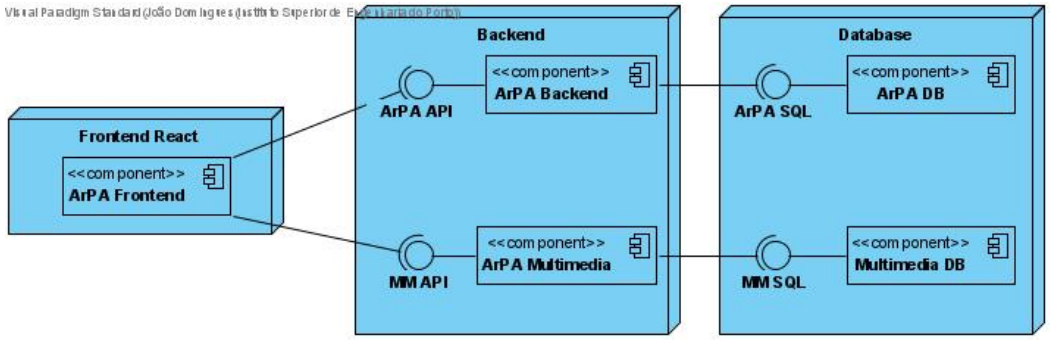


Figura 5.4: Diagrama de Implantação da Alternativa C.

5.1.3 Componentes do Backend

Uma vez que já estão definidos os componentes que fazem parte do *backend*, o próximo passo consiste em determinar a arquitetura de cada um dos componentes. Foram consideradas as seguintes alternativas:

Alternativa A

A primeira solução considerada é o padrão arquiterural *Model, View e Controller (MVC)*, representado na figura 5.5, que visa separar a interface do utilizador de uma aplicação, dos dados associados à mesma [91]. Aqui, os utilizadores não interagem com os dados de forma direta, mas sim com uma representação desses dados - uma **View** - gerida por um **Controller** que, por sua vez, atualiza os dados - o **Model** - em conformidade com as ações do utilizador.

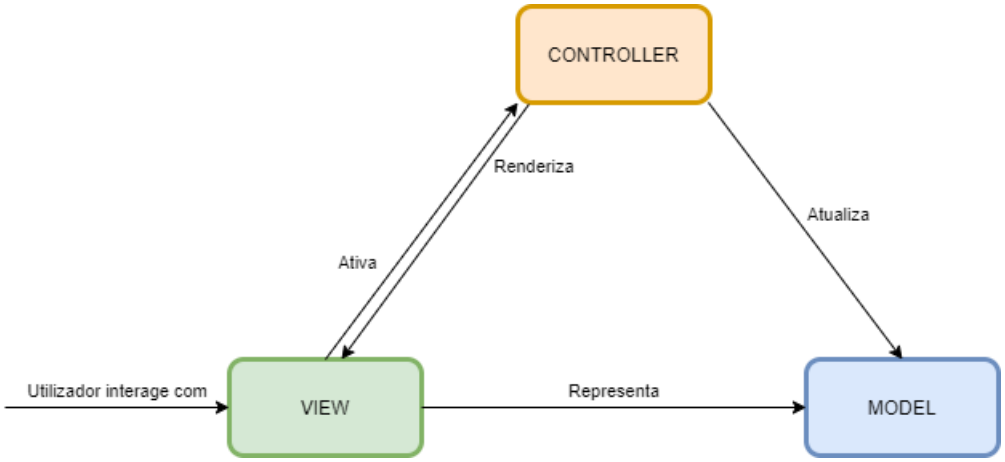


Figura 5.5: Arquitetura MVC.

A arquitetura *MVC* divide-se nos seguintes componentes:

- **Model** - Consiste no conjunto de conceitos associados ao negócio. Podem ser representados por classes, como é feito no caso do paradigma de *Object Oriented Programming (OOP)*. O **Model** não tem conhecimento de mais nada além de si mesmo [92];

- **View** - Consiste na representação do **Model** sob uma determinada forma. A **View** é utilizada para o mundo exterior poder observar o **Model**.

  As **Views** podem surgir sob variadas formas, como por exemplo, em formato de *GUI's*, *CLI's* ou *API's* [92];

- **Controller** - Em contraste com a **View**, que representa o **Model** mas não o altera, o **Controller** tem o poder de provocar alterações no **Model** e, por consequência, na **View** também. O **Controller** responde às ações do utilizador [92];

Alternativa B

Alternativamente, é considerada uma arquitetura **multicamada**, que separa o *software* em camadas horizontais, sendo atribuída, a cada uma, uma responsabilidade específica.

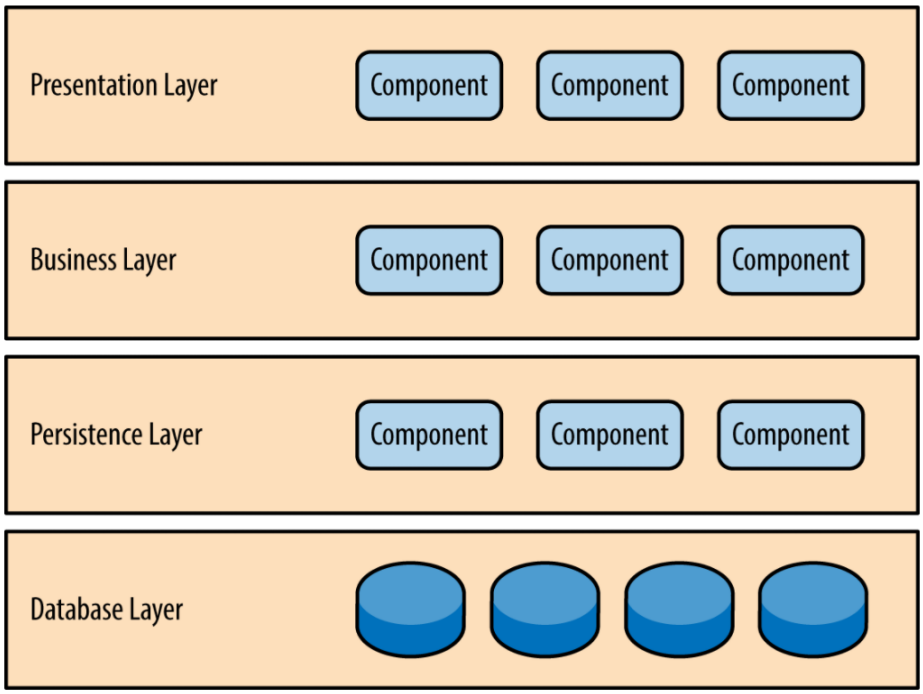


Figura 5.6: Arquitetura Multicamada [93].

Tal como representado na figura 5.6, as camadas são constituídas por **Componentes** que lidam exclusivamente com a lógica associada à camada a que pertencem.

Tipicamente, numa arquitetura multicamada podem verificar-se as camadas de **Apresentação**, **Negócio**, **Persistência** e de **Base de Dados**.

Seleção de Alternativas

De forma a selecionar uma alternativa para cada um dos componentes que constituem o *backend* do sistema representado em 5.4, é importante analisar o objetivo de cada um deles.

No caso do monólito, uma vez que se trata de um componente responsável pela gestão da informação referente ao negócio que é, na sua totalidade, relevante para o utilizador, é do interesse dos mesmos ter acesso a essa informação sob determinada forma. Este tipo de

necessidade é respondida pela **alternativa A** - o padrão *MVC*. Uma vez que o monólito em questão estará a servir o *frontend* da aplicação, as *Views* são disponibilizadas sob a forma de uma *Application Program Interface (API)*.

Por outro lado, o micro-serviço especificado na imagem 5.4 tem como responsabilidade a gestão de dados multimédia, que são armazenados em serviços externos, tal como referenciado na secção 2.3 - este micro-serviço apenas armazena referências para os conteúdos multimédia armazenados externamente. É, então, possível delinear duas funções distintas:

- Interagir com os serviços externos;
- Armazenar as referências.

Ao recorrer à arquitetura **multicamada** (**alternativa B**), é possível fazer esta separação de responsabilidades, ao atribuir cada uma delas a uma determinada camada. Assim:

- A interação com os serviços externos é da responsabilidade da camada de *Business*;
- O armazenamento de referências é da responsabilidade da camada de *Persistence*.

5.1.4 Sistema de Mensagens

Um sub-conjunto dos requisitos listados para a aplicação na secção 3.2.1 gira à volta da ideia que dois utilizadores podem trocar mensagens entre si, em tempo real, a partir de um *chat*.

Este tipo de comunicação em tempo real requer que haja uma forma de um cliente saber que há uma nova mensagem no servidor que foi emitida para ele.

De forma a resolver este problema, foram consideradas as seguintes soluções:

Alternativa A

Uma possível alternativa mais arcaica consiste em implementar um sistema de **espera ativa** no lado do cliente, também conhecida como **polling** [94].

Esta espera ativa consiste em fazer com que o cliente emitisse pedidos constantes (ou com uma alta frequência) ao servidor a requisitar as mensagens que lhe foram enviadas.

Alternativa B

A segunda alternativa tida em consideração recorre ao conceito de **WebSockets**, onde se estabelece um canal de informação bidirecional entre o cliente e o servidor a partir de uma única conexão *Transmition Control Protocol (TCP)* estabelecida entre os dois [95].

Em contraste com os pedidos normais de *Hypertext Transfer Protocol (HTTP)*, onde a única forma de o servidor comunicar com o cliente é em resposta a um pedido feito por parte do cliente, a utilização de **WebSockets** permite ao servidor tomar a iniciativa de comunicar com o cliente [95].

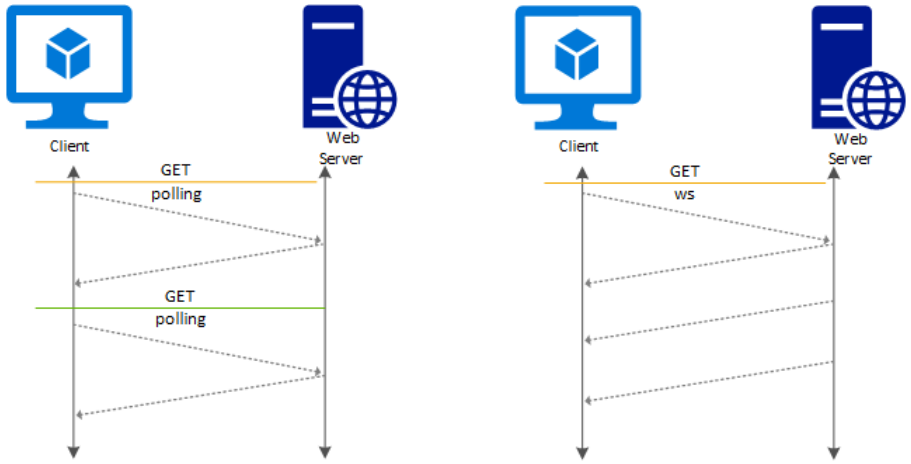


Figura 5.7: Polling vs. Websockets [96].

A utilização de *WebSockets* requer, no entanto, que haja um pedido *HTTP* inicial de forma a estabelecer a comunicação entre as duas entidades - essa comunicação é então convertida para um *WebSocket*, tal como demonstrado na imagem 5.7.

Alternativa C

A terceira alternativa consiste na utilização de um **Message Queueing Telemetry Transport (MQTT)**, representado na figura 5.8. Em semelhança aos *WebSockets*, um *MQTT* dá a possibilidade ao servidor de comunicar com o cliente por iniciativa própria, de forma assíncrona, recorrendo ao paradigma *Publish-Subscribe* [97].

Dentro do próprio conceito de *MQTT* é ainda possível explorar diferentes opções arquiteturais.

Esta solução recorre a um módulo externo que serve de *middleware* de forma a distribuir as mensagens que são emitidas pelo *Publisher*, entregando-as às entidades interessadas - os *Subscribers*.

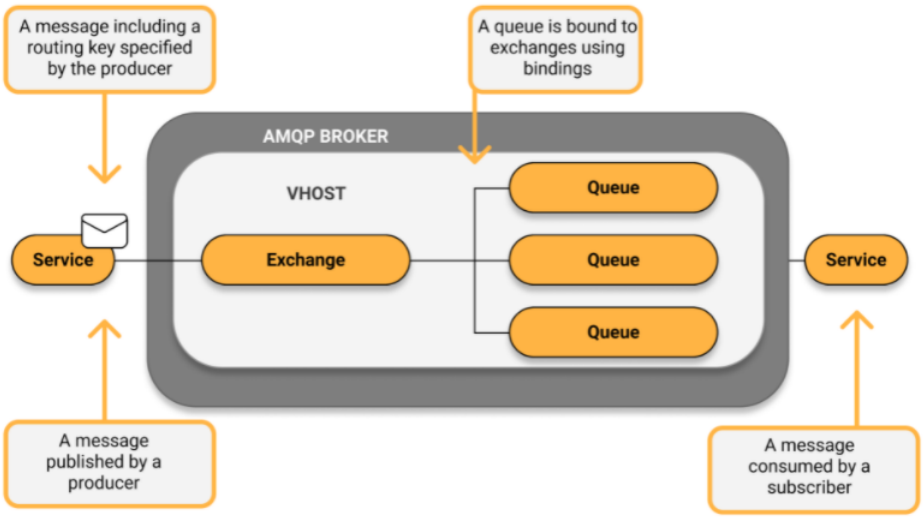


Figura 5.8: Exemplo de um *MQTT Broker* utilizando arquitetura AMQP [98].

Seleção de Alternativas

No sentido de implementar um sistema de troca de mensagens instantâneas entre utilizadores, é necessário haver uma ponte de comunicação aberta entre o servidor e o cliente.

A **alternativa A**, embora simples, não é de todo escalável, impondo uma carga enorme e desnecessária no sistema, o que não é desejado.

Ainda que se recorresse a uma técnica de *Long Polling*, na qual o cliente mantém a conexão estabelecida com um pedido por um longo período que terminasse quando surgisse uma nova mensagem (ou quando ocorresse um *timeout*), esta solução não resolve os problemas associados com o servidor não ter conhecimento de quando o cliente se desconecta [94].

Adicionalmente não é uma solução que se adapte a utilizadores que não utilizem a funcionalidade de *chat* com frequência, uma vez que se estariam a alocar recursos para tentar notificar um utilizador de algo que ele não usa.

Entre as duas alternativas que dão ao servidor a oportunidade de ter a iniciativa de comunicar com o cliente - **alternativa B** e **C** - tem que se analisar as vantagens e desvantagens de cada um à luz do contexto do problema.

Quanto à escalabilidade, o *MQTT Broker* apresenta uma ligeira vantagem sobre os *WebSockets* quando em situações de maior carga. Ambas as soluções apresentam taxas de falha na entrega das mensagens negligenciáveis [99].

No entanto, enquanto que os *WebSockets* permitem apenas a ligação entre dois pontos, o *MQTT* permite que as mensagens enviadas por um *Publisher* possam ser recebidas por diversos *Subscribers* a partir da mesma *Queue* - isto abre a possibilidade de expandir a funcionalidade de *chat* de forma a incluir comunicação entre grupos de pessoas.

Por isso mesmo, o autor opta pela **alternativa C**, como representado na figura 5.9.

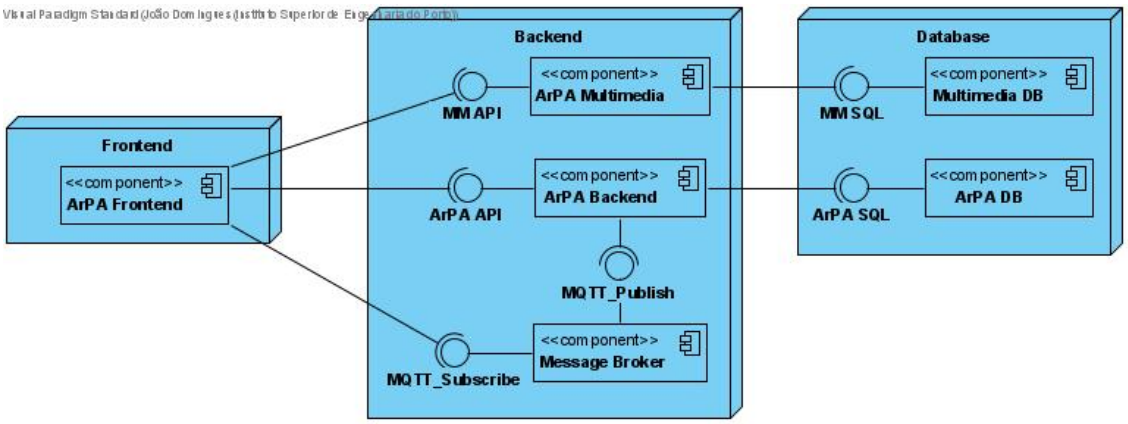


Figura 5.9: Inclusão do *Message Broker* na arquitetura desenvolvida até agora.

5.1.5 API Gateway

Analisando a figura 5.9, é possível concluir que o *frontend* da aplicação tem de ter conhecimento de duas *API* diferentes. A introdução de um **API Gateway** retira ao *frontend* a responsabilidade de ter conhecimento quantas *API* o *backend* disponibiliza e de como as aceder [100, 101].

A utilização de um **API Gateway** traz algumas vantagens:

- Encapsulamento de todo o *backend* num único ponto de acesso;
- Controlo de segurança e monitorização dos acessos aos serviços disponibilizados pelo *API Gateway*.

Deste modo, representa-se na figura 5.10 o sistema incluindo o *API Gateway*.

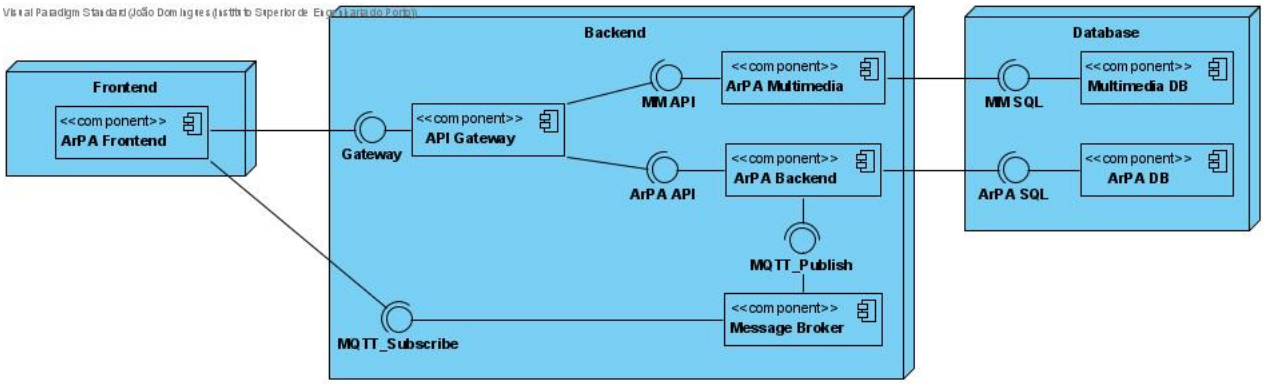


Figura 5.10: Inclusão do *API Gateway* na arquitetura desenvolvida.

5.2 Casos de Uso

A nível de implementação, os casos de uso listados na secção 3.3 são bastante semelhantes entre eles, havendo apenas três tipos distintos de implementação:

- Casos de uso que apenas acedem a informação providenciada pelo monólito *ArPA Backend* apresentado na figura 5.10 - estes casos de uso operam apenas com informação referente aos utilizadores da plataforma, informação esta que não engloba elementos multimédia;
- Casos de uso que consistem numa extensão do ponto anterior, acedendo também ao micro-serviço *ArPA Multimédia* apresentado na figura 5.10 - estes casos de uso interagem com elementos multimédia;
- Casos de uso referentes ao sistema de mensagens instantâneas que utilizam o *Message Broker*.

Estabelecendo-se esta distinção, apresenta-se o *design* referente ao segundo e terceiro tipo, recorrendo para tal a um caso de uso para cada um deles, para exemplificação.

5.2.1 Diagrama de Classes

A presente secção pretende listar as diferentes classes disponibilizadas pela aplicação, apresentando a sua utilidade e a forma como interagem entre si.

As classes apresentam-se agrupadas pelos diferentes serviços que compõem o sistema.

- **React Component** - Cada componente que constitui o *frontend* representa ele próprio uma classe. Uma vez que uma determinada página é constituída por dezenas de componentes, aqui será apenas representado o componente responsável pelo caso de uso

a ser representado. Esta separação da *User Interface (UI)* em diversos componentes promove a separação de responsabilidades, bem como a reutilização de componentes.

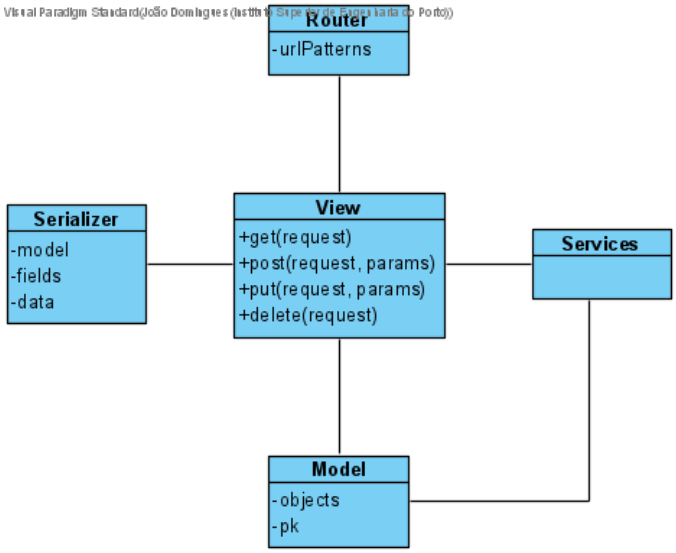


Figura 5.11: Classes referentes ao monólito apresentado na imagem 5.10.

- **Django Router** - Classe responsável por reencaminhar os pedidos recebidos dentro do serviço para serem processados;
- **Django View** - Contrariamente ao que o nome indica, as *Views* disponibilizadas pela *framework Django* correspondem a um *Controller* convencional, uma vez que *Django* utiliza um equivalente ao *MVC*, o *Model, Template e View (MTV)* [102]. Assim, esta classe é responsável por processar o pedido externo, após este ter sido reencaminhado pelo *Router*;
- **Django Model** - Classes de domínio, representam os conceitos de domínio expressos no Modelo de Domínio;
- **Django Serializer** - Classe responsável por definir como é que as classes de domínio são serializáveis, de forma a serem enviadas como resposta ao pedido recebido pelo serviço;
- **GestorGeneros**[1] - Classe que representa um serviço onde se reúnem as responsabilidades de gestão dos géneros de projetos que a aplicação vai gerir. Este serviço comunica com uma *API* externa;
- **GeradorNotificacoes**[2] - Classe que representa um serviço onde se reúnem as responsabilidades de gerar as notificações de sistema, de forma a poder haver algum tipo de *feedback* para o utilizador sobre a forma de notificações.

[1] Representado pela classe Services na figura 5.11.
[2] Representado pela classe Services na figura 5.11.

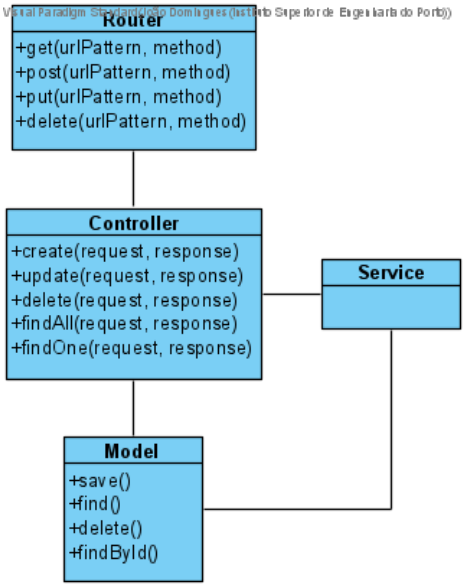


Figura 5.12: Classes referentes ao micro-serviço apresentado na imagem 5.10.

- **NodeJS Router** - Classe responsável por reencaminhar os pedidos recebidos dentro do serviço para serem processados;
- **NodeJS Controller** - Classe responsável por processar o pedido externo, após este ter sido reencaminhado pelo *Router*;
- **NodeJS Model** - Classes de domínio, representam os conceitos de domínio expressos no Modelo de Domínio;
- **PhotoService**[3] - Classe que representa um serviço onde se reúnem as responsabilidades de gestão de elementos multimédia da aplicação que sejam imagens;
- **VideoService**[4] - Classe que representa um serviço onde se reúnem as responsabilidades de gestão de elementos multimédia da aplicação que sejam vídeos.

5.2.2 Diagrama de Sequência

Uma vez que estão identificadas as responsabilidades atribuídas aos diferentes tipos de classes que compõem os diferentes componentes do sistema e a forma como interagem entre si, nesta secção apresentam-se dois exemplos de Casos de Uso que demonstram a implementação do sistema.

Assim, nas figuras 5.13 e 5.14, apresenta-se o Diagrama de Sequência do **UC18 - Criar Projeto** fragmentado em duas partes distintas - primeiro, a que diz respeito ao funcionamento do monólito, seguido do diagrama referente ao micro-serviço.

[3] Representado pela classe Services na figura 5.12.
[4] Representado pela classe Services na figura 5.12.

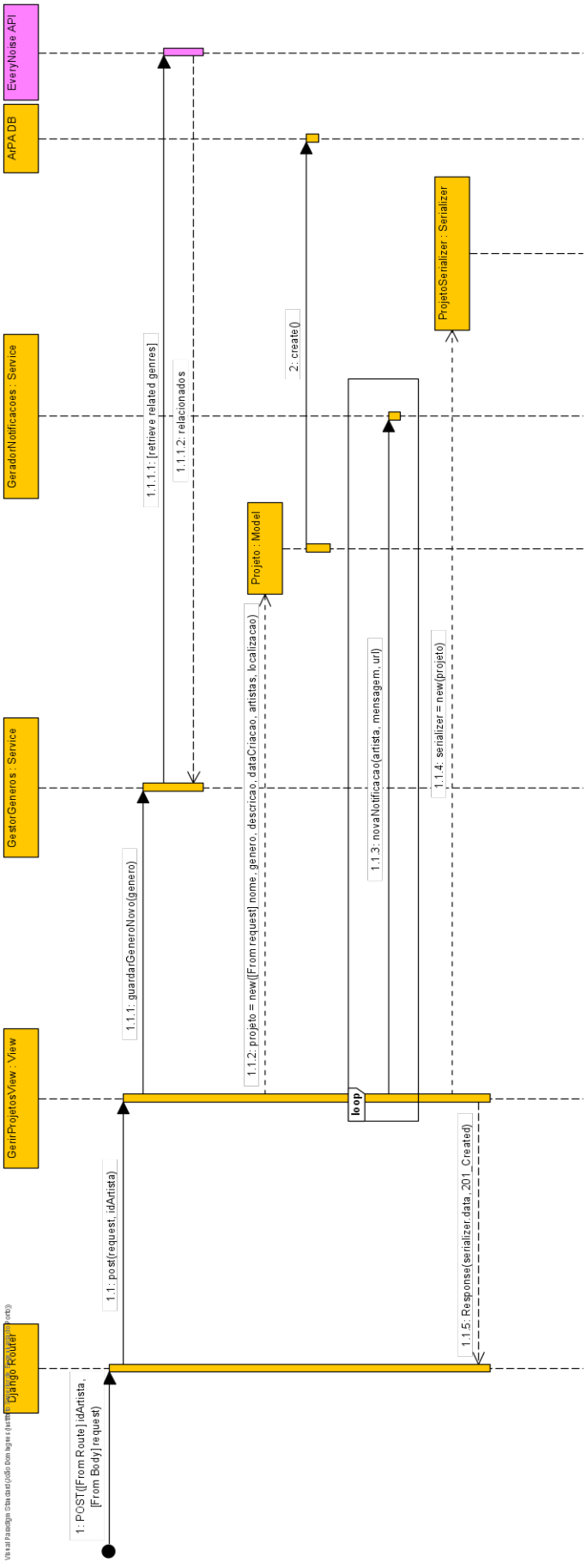


Figura 5.13: Fragmento do Diagrama de Sequência do UC18 - Criar Projeto.

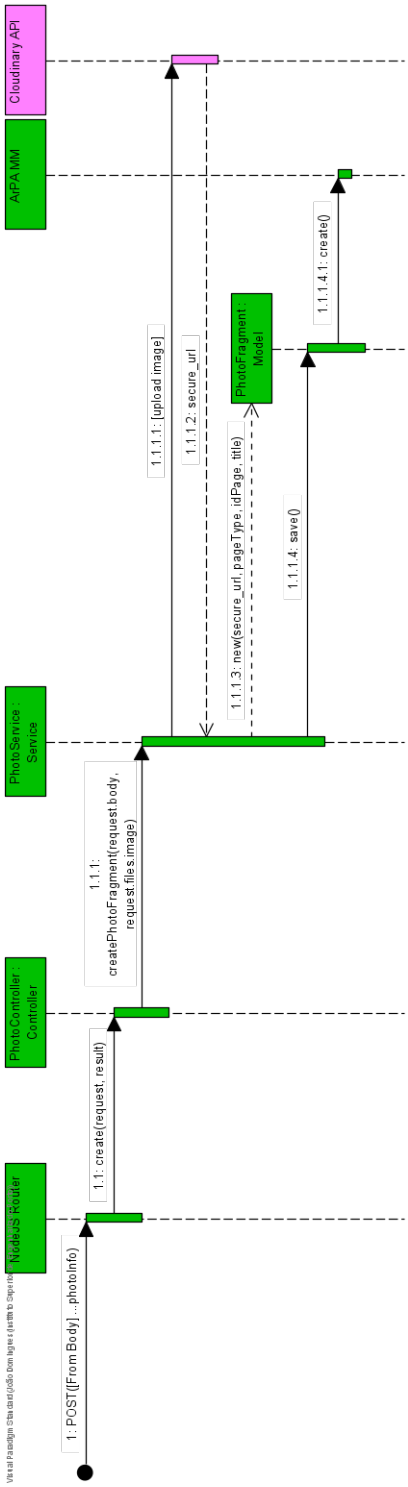


Figura 5.14: Fragmento do Diagrama de Sequência do UC18 - Criar Projeto.

É possível ver uma versão completa do diagrama de sequência do **UC18 - Criar Projeto** no Apêndice A.

Nos dois diagramas apresentados nas figuras 5.13 e 5.14 é possível verificar uma concordância para com os diagramas de classes representados nas figuras 5.11 e 5.12. Os pedidos são

recebidos pelo *Router*, que os reencaminha para o *Controller* correto, consoante a tecnologia em uso, onde se dá o processamento do pedido.

No caso do monólito, as regras de negócio são aplicadas pelos *Services* ou pela própria *View* (no caso de o processamento de informação ser único e reduzido, não precisando de um *Service*). A criação de instâncias do *Model* é feita a partir de qualquer um dos dois, mas a instanciação de um *Serializer* é da responsabilidade da *View*, que o utilizará para gerar uma resposta ao pedido recebido com informação referente ao *Model*.

No caso do micro-serviço, as regras de negócio são também aplicadas pelos *Services* ou pelos próprios *Controllers*.

A comunicação com *API*'s externas é feita exclusivamente a partir dos *Services* em qualquer um dos casos.

Em contraste, na figura 5.15, apresenta-se o Diagrama de Sequência do **UC06 - Enviar Mensagem**.

Aqui, está representada a assincronia inerente da utilização do *Message Broker* de forma a sinalizar a transmissão de mensagens entre diferentes clientes.

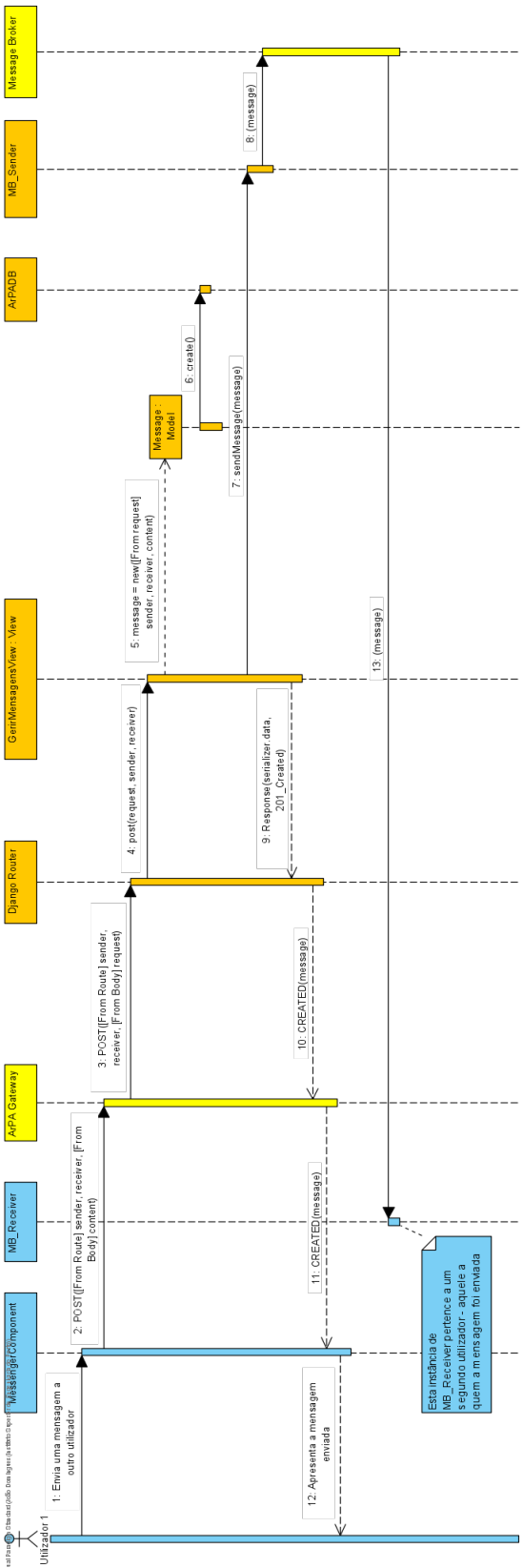


Figura 5.15: Diagrama de Sequência do UC06 - Enviar Mensagem.

5.3 Conclusões

Com a conclusão deste capítulo, o leitor tem agora uma melhor perceção do processo tomado para o desenho da arquitetura que suportará o desenvolvimento deste projeto.

Esta arquitetura consiste num *backend* composto por um monólito e um micro-serviço, responsáveis pela gestão da informação referente aos utilizadores da plataforma e pela gestão de conteúdos audiovisuais, respetivamente. Estes dois componentes irão servir um único *frontend*. Dentro destes componentes, as arquiteturas utilizadas são o *MTV* e multicamada.

Adicionalmente, foi definido o modo de funcionamento do sistema de mensagens, que é baseado na utilização de um *message broker*, de forma a criar canais bi-direcionais de informação entre servidor e cliente para providenciar comunicação em tempo real entre utilizadores.

Finalmente, estabeleceu-se uma barreira de segurança e abstração entre o *frontend* e o *backend* ao incluir-se uma *API Gateway*.

Após a definição da arquitetura do sistema, passou-se à definição do funcionamento dos casos de uso, onde foram destacados os diferentes tipos de caso de uso oferecidos pela plataforma.

Aqui, para cada um dos tipos, apresentaram-se os respetivos diagramas de classes e de sequência de forma a poder visualizar o modo como as classes que constituem os diversos componentes do sistema interagem entre si.

Capítulo 6

Implementação da Solução

Uma vez concluída a fase de *design* da aplicação a desenvolver, procede-se com a implementação da mesma.

Neste capítulo é apresentada a metodologia de desenvolvimento utilizada ao longo da implementação do projeto, seguida de uma descrição de cada uma das fases que compõem a metodologia, apresentando-se alguns exemplos práticos.

6.1 Metodologia de Desenvolvimento

O desenvolvimento deste projeto consiste em 5 fases principais:

1. Criação do modelo de dados no *backend*;
2. Escolha de um Caso de Uso a desenvolver;
3. Desenvolvimento do *backend* para o Caso de Uso selecionado;
4. Desenvolvimento do *frontend* para o Caso de Uso selecionado;
5. Realização de testes às funcionalidades desenvolvidas.

O passo 1 decorreu uma única vez, enquanto que os restantes passos decorrem de forma cíclica. Nesta secção descreve-se cada um dos passos.

6.1.1 Modelo de Dados

A criação do modelo de dados consiste na implementação de todos os conceitos de negócio que constituem o modelo de domínio, juntamente com as suas relações e operações *Create, Read, Update e Delete (CRUD)* básicas. Este passo não contempla quaisquer regras de negócio.

O objetivo principal consiste em ter um modelo de dados sólido e acessível para o desenvolvimento posterior da aplicação - o desenvolvimento de um novo Caso de Uso não deve implicar a alteração do modelo de dados que impacte outros Casos de Uso.

No excerto de código 6.1, apresentam-se alguns excertos de código que demonstram o trabalho desenvolvido nesta fase.

O primeiro excerto representa a definição de uma classe de domínio, onde são listados os atributos da classe, com os respetivos parâmetros necessários para ser persistida na base de dados.

```
1  from django.db import models
2
3  class Projeto(models.Model):
4      nome = models.CharField(max_length=100)
5      genero = models.CharField(max_length=100)
6      descricao = models.CharField(max_length=500)
7      dataCriacao = models.DateField()
8      localizacao = models.ForeignKey(Localizacao, on_delete=models.
       SET_DEFAULT, default=0)
9      artistas = models.ManyToManyField("Utilizador", related_name="
       artistas")
10     agente = models.ForeignKey(Utilizador, on_delete=models.SET_NULL,
       null=True, related_name="agente")
```

Listagem 6.1: Definição de uma classe de domínio no models.py.

No excerto de código 6.1, que representa a classe **Projeto**, estão definidos campos de texto, datas e de classes terceiras. Estas classes terceiras são referênciadas pelo seu **id** (*ForeignKey*) e é atribuída uma ação *default* quando a instância referênciada é apagada.

Uma vez criada uma classe de domínio, é-lhe associada um *serializer*, de forma a definir quais os parâmetros que serão expostos a partir da *Application Program Interface (API)*. Nesta fase inicial, são expostos todos os parâmetros de todas as classes, de forma a ter acesso a toda a informação através da *API*.

```
1  from rest_framework import serializers
2  from arpa.models import Projeto
3
4  class ProjetoSerializer(serializers.ModelSerializer):
5
6      # Redefine como sao apresentados estes campos
7      artistas = UtilizadorSerializer(many=True)
8      localizacao = LocalizacaoSerializer(read_only=True)
9
10     class Meta:
11         model = Projeto
12         fields = '__all__'
```

Listagem 6.2: Definição de um *Serializer* no serializers.py.

No excerto de código 6.2, ainda é possível ver, nas linhas 7 e 8, a forma como campos que representam instâncias de outras classes são serializados. Estes campos podem ser representados ou pelo seu **id** ou pelo seu próprio *serializer*.

É ainda necessário criar um *ModelViewSet* de forma a permitir aceder e criar/editar instâncias da classe de domínio, bem como adicionar um padrão *URL* ao *Router*, tal como representado nas listagens 6.4 e 6.3, respetivamente.

```
1  from rest_framework import routers
2  from .api import ProjetoViewSet
3
4  router = routers.DefaultRouter()
5  router.register('api/arpa/projetos', ProjetoViewSet, 'arpa')
6
7  urlpatterns = router.urls
```

Listagem 6.3: Inclusão de um novo padrão de *url* no urls.py.

```python
from rest_framework import viewsets, permissions
from arpa.models import Projeto
from .serializers import ProjetoSerializer

class ProjetoViewSet(viewsets.ModelViewSet):
    queryset = Projeto.objects.all()
    permission_classes = [
        permissions.AllowAny
    ]
    serializer_class = ProjetoSerializer
```

Listagem 6.4: Definição de um *ViewSet* no api.py.

O resultado deste passo consiste numa *API* que expõe um conjunto de classes de domínio com uma estrutura lógica em conformidade com o modelo de domínio. Estas classes são passíveis de serem persistidas numa base de dados e é possível efetuar operações *CRUD* básicas recorrendo a pedidos *HTTP*. Estas operações básicas iniciais foram efetuadas recorrendo a uma ferramenta de utilização de *API's* - neste caso concreto, o *Postman*, como representado na figura 6.1.

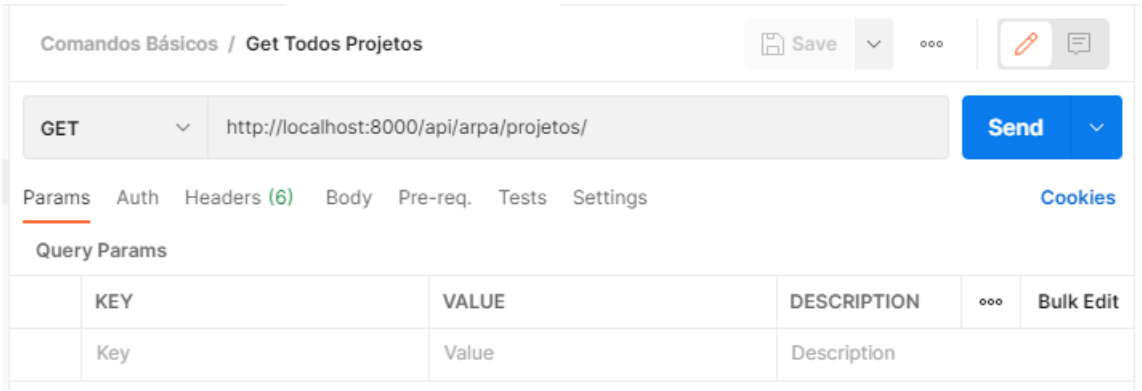


Figura 6.1: Pedido GET que permite aceder às intâncias da classe Projeto armazenadas na base de dados.

6.1.2 Escolha de um Caso de Uso a desenvolver

Uma vez criada a base da estrutura de dados para criar a aplicação, procede-se com a seleção de Casos de Uso para desenvolver.

A seleção de Casos de Uso baseou-se em dois critérios - tipo de operação *CRUD* e relevância/grau de utilização do conceito de negócio.

Os Casos de Uso foram agrupados em tipo de operação, onde foi dada prioridade às operações de leitura, seguidas de escrita, edição e, finalmente, remoção de informação. Para cada grupo, foram implementados os Casos de Uso referentes a conceitos de negócio por ordem de relevância.

Na opinião do autor, ainda que fosse mais eficiente implementar todos os Casos de Uso referentes a um determinado conceito de negócio de uma só vez, a metodologia adotada permitiu ao protótipo adquirir de forma muito mais rápida o estatuto de *Minimum Viable Product (MVP)*, uma vez que Casos de Uso mais frequentes referentes a conceitos mais importantes eram implementados primeiro.

No entanto, é importante referir que a lista de requisitos foi adaptada ao *feedback* recebido e aos testes realizados - embora a metodologia se tenha mantido ao longo de todo o processo de desenvolvimento, conceitos considerados prioritários variaram ao longo do tempo.

6.1.3 Implementação do Backend

Esta secção serve-se do **UC18 - Criar Projeto** para exemplificar a implementação do *backend* da aplicação tanto no monólito como no micro-serviço. Ao longo desta secção são apresentados excertos de código referentes a este Caso de Uso. A utilização deste caso de uso como exemplo deve-se ao facto de ser o caso de uso mais abrangente em termos de conceitos de implementação do *backend*.

Começando pelo monólito, o primeiro passo é criar um padrão *url* para aceder à nova funcionalidade a ser implementada - este padrão é recebido pelo *router*, que o usa para reencaminhar o pedido *Hypertext Transfer Protocol (HTTP)* recebido para uma *View* que o processa e trata de enviar uma resposta. O resultado deste processo é demonstrado na listagem 6.5.

```
1 from django.urls import path, include
2 from arpa.views import GerirProjetosView
3
4 urlpatterns = [
5     path('artistas/<int:idArtista>/projetos/', GerirProjetosView.as_view
      ()),
6 ]
```

Listagem 6.5: Inclusão de um novo padrão de *url* no urls.py.

Após isso, define-se a *View*. Aqui, a criação de um novo projeto será feita a partir de um pedido **POST**, que recebe a informação referente ao projeto a ser criado no *body* do pedido, tal como representado na listagem 6.6.

```
1 from rest_framework.views import APIView
2
3 class GerirProjetosView(APIView):
4
5     # artistas/<int:idArtista>/projetos/
6     def post(self, request, format=None, idArtista=None):
7         pass
```

Listagem 6.6: Definição de uma *View* no views.py.

Uma vez recebido o pedido, o primeiro passo será extrair a sua informação, validando-a. As validações efetuadas no *backend* são referentes a regras de negócio e não de formato da informação recebida. No caso do **UC18 - Criar Projeto**, valida-se, entre outros, os artistas associados ao projeto que se pretende criar, verificando tratarem-se de utilizadores que existem e que são do tipo certo. No caso de haver alguma incoerência nos dados recebidos, é imediatamente enviada uma mensagem de erro. O resultado deste processo está representado na listagem 6.7.

```
def post(self, request, format=None, idArtista=None):
    # extracao dos campos no body do pedido
    [...]
    artistas = request.data.get("artistas")

    # validacao dos artistas propostos
    [...]
    for a in artistas:
        artista = Validator.validaUtilizador(a, TipoUtilizador.ARTISTA)

        if artista is None:
            return Response({"error": "Um dos utilizadores associados ao
     projeto nao e elegivel"}, status=status.HTTP_406_NOT_ACCEPTABLE)

    [...]
```

Listagem 6.7: Implementação do método POST no *GerirProjetosView*.

Uma vez que diferentes casos de uso irão partilhar o mesmo tipo de validações, foi criado um serviço de validação de forma a reduzir a duplicação de código e isolar a lógica de validação. No excerto de código 6.8 apresenta-se a implementação da validação de utilizadores, que verifica a sua existência no sistema e, quando necessário, procede à validação do seu tipo.

```
class Validator():

    def validaUtilizador(idUtilizador, tipoUtilizador=None):

        utilizador = Utilizador.objects.filter(pk=idUtilizador)

        if not utilizador.values():
            return None

        utilizador = utilizador[0]

        if tipoUtilizador is None:
            return utilizador

        if utilizador.tipoUtilizador != tipoUtilizador:
            return None

        return utilizador
```

Listagem 6.8: Validação do utilizador no serviço de validação.

Após validação dos dados recebidos, é então possível criar e armazenar uma instância do objeto a ser criado.

O processamento do pedido termina com o envio de uma resposta ao utilizador que o efetuou, de forma a indicar o sucesso da operação. Nesta resposta está incluída a informação referente ao projeto criado.

O **UC18 - Criar Projeto** insere-se num conjunto de Casos de Uso cujo resultado pode ser do interesse de mais utilizadores além daquele que o efetuou. Por esta razão, e de forma a providenciar um sistema que fornece *feedback* aos utilizadores, foi ainda criado um serviço adicional que gera notificações.

Estas notificações são armazenadas na base de dados para poderem ser acedidas pelo utilizador mesmo depois de serem vistas. Casos de Uso recorrem a este serviço quando há a

necessidade de notificar os utilizadores de algo que lhes é relevante ocorreu. No excerto de código 6.9, apresenta-se a criação de notificações para os artistas que compõem o projeto a ser criado no **UC18 - Criar Projeto**.

```
def post(self, request, format=None, idArtista=None):

    [...]

    mensagem = "%s criou um projeto (%s) do qual faz parte." %(artista1.
    nome, projeto.nome)
    url = "/project/%s" %(projeto.pk)

    for artista in projeto.artistas.all():
        GeradorNotificacoes.novaNotificacao(artista, mensagem, url)

    [...]
```

Listagem 6.9: Criação de Notificações.

As notificações referentes a um utilizador contêm uma mensagem a explicar o sucedido e estão associadas a um *url* que os reencaminhará dentro da aplicação para visualização do conteúdo referido pela notificação.

Como resultado do processo apresentado no excerto de código 6.9, é agora possível efetuar um pedido POST de forma a criar um projeto novo, tal como representado na figura 6.2.

Figura 6.2: Exemplo de pedido POST que permite criar um Projeto.

No entanto, o **UC18 - Criar Projeto** insere-se num grupo de Casos de Uso que dão utilidade ao micro-serviço responsável pela gestão de elementos multimédia. Segue-se a implementação feita no micro-serviço.

A função do micro-serviço é gerir o conteúdo multimédia da aplicação, independentemente da origem deste conteúdo - *uploaded* para a aplicação pelo utilizador ou proveniente de outra aplicação/plataforma. Os seguintes excertos de código descrevem a implementação do micro-serviço à data da escrita da presente dissertação.

Tal como representado na figura 5.12, existem 4 partes principais no funcionamento do micro-serviço - *router*, *controller*, *model* e *service*.

O primeiro passo trata de definir os *models* a serem usados pelo micro-serviço que, à data da escrita da presente dissertação, são apresentados nos excertos de código 6.10 e 6.11.

```
const PhotoFragmentSchema = mongoose.Schema({
    urlPhoto: String,
    title: String,
    isProfilePic: Boolean,
    idPage: Number,
    pageType: {
        type: String,
        enum: ['Perfil', 'Projeto', 'Evento'],
        default: 'Perfil'
    }
});
```

Listagem 6.10: Model associado a imagens.

```
const VideoFragmentSchema = mongoose.Schema({
    urlVideo: String,
    title: String,
    description: String,
    sourceType: {
        type: String,
        enum: ['Youtube', 'Facebook'],
        default: 'Youtube'
    },
    idProject: Number
});
```

Listagem 6.11: Model associado a vídeos.

Em seguida, implementam-se os *routers* que reencaminharão os pedidos recebidos pelo micro-serviço para os *controllers* corretos. Este processo está representado no excerto de código 6.12.

```
module.exports = (app) => {
    const PhotoFragments = require('../controllers/photoFragment.
    controller.js');

    // Create a new PhotoFragment
    app.post('/photoFragments', PhotoFragments.create);

    // Put a PhotoFragment by id
    app.put('/photoFragments/:id', PhotoFragments.delete);

    // Retrieve all PhotoFragments
    app.get('/photoFragments', PhotoFragments.findAll);

    // Retrieve a single PhotoFragment by id
    app.get('/photoFragments/:type/:id', PhotoFragments.findOne);
}
```

Listagem 6.12: Inclusão dos padrões de *url* no photoFragment.routes.js.

Um vez que o *controller* receba o pedido, a primeira fase do seu processamento é a validação dos dados recebidos. Esta validação varia conforme o tipo de operação a efetuar. O excerto de código 6.13 representa a validação efetuada para o **UC18 - Criar Projeto**, que verifica

a receção de todos os dados necessários para o armazenamento da imagem carregada pelo utilizador.

```
exports.create = (req, res) => {
    // Validate request
    if(!req.body.title || !req.body.pageType || !req.body.idPage || !req
    .files.image) {
        return res.status(400).send({
            message: "PhotoFragment is not valid"
        });
    }

    [...]

}
```

Listagem 6.13: Validação de dados no photoFragment.controller.js.

A responsabilidade de comunicação com o serviço que armazenará as imagens carregadas é atribuída a um *service* próprio para o efeito. Assim, retira-se esta responsabilidade ao *controller*, promovendo a redução na duplicação de código. Apresenta-se o *service* criado na listagem 6.14.

```
var cloudinary = require('cloudinary')
let streamifier = require('streamifier');

cloudinary.config({
    cloud_name: '????????',
    api_key: '????????',
    api_secret: '?????????'
});

module.exports = {
    createPhotoFragment: async function (body, image) {

        let cld_up_stream = cloudinary.v2.uploader.upload_stream(
            { folder: "ArPA" },
            // Armazenamento da referencia gerada pelo Cloudinary
            [...]
        );

        streamifier.createReadStream(image.data).pipe(cld_up_stream);
    }
}
```

Listagem 6.14: Comunicação com o Cloudinary Service no photoFragment.service.js.

6.1.4 Implementação do Frontend

A implementação do *frontend* da aplicação é feito inteiramente em *ReactJS*, recorrendo à *framework Material-UI* [103] para desenvolvimento dos elementos visuais.

Ao longo desta secção são apresentados alguns pormenores de implementação sob a forma de excertos de código e/ou *screenshots* da *User Interface (UI)* da aplicação.

User Interface

Figura 6.3: Home Page da aplicação.

Tal como demonstra a figura 6.3, a *UI* encontra-se segmentada em 4 partes fundamentais, estando duas delas estão apenas reservadas a utilizadores que têm o *login* efetuado. São elas:

1. **Header** - Cabeçalho da página permite acesso a um menu lateral, a uma barra de pesquisa e, dependendo se o utilizador tem o *login* feito ou não, dá acesso a:
    - Notificações do sistema;
    - Acesso às funcionalidades de *login* e registo de uma nova conta de utilizador.
2. **Body** - A aplicação em si, divida em várias páginas principais, com diferentes políticas de acesso. Em certos casos, a mesma página oferece diferentes opções a diferentes tipos de utilizadores:
    - *Páginas para todos os utilizadores* - *Homepage*, Sobre, Pesquisa, Perfis;
    - *Páginas para utilizadores sem login efetuado* - *Login*, Registar novo utilizador;
    - *Páginas para todos os utilizadores com login efetuado* - *Dashboard*;
    - *Páginas para determinado tipo de utilizador* - Formulários de interação com conceitos referentes a cada tipo de utilizador (eventos, projetos, candidaturas, ...).
3. **Footer** - Barra de acesso rápido a 3 páginas essenciais - perfil do utilizador com *login* efetuado, página de pesquisa e *dashboard* de utilizador. Apenas visível para utilizadores com *login* efetuado;
4. **Messenger** - Pequena janela colapsável que dá acesso ao *chat* com outros utilizadores.

De forma a reduzir a dificuldade e tempo de adaptação de um novo utilizador à aplicação, a *UI* foi desenhada de forma a que elementos herdados de outros produtos mais difundidos sejam apresentados de forma semelhante ou reconhecível, ou funcionem de forma semelhante. Exemplos de alguns destes elementos são as notificações e a janela do *messenger*, que têm

posicionamento e funcionamento bastante semelhantes a produtos como o *Facebook* e o *LinkedIn*. Este esforço contribui para uma melhor **Usabilidade** da aplicação.

De forma semelhante, alguns elementos que são exclusivos da aplicação e com os quais um novo utilizador ainda não está familiarizado, possuem aspetos subtis que, de forma indireta, indicam ao utilizador como interagir com os mesmos. Um exemplo para o caso supramencionado são os Cartões de Perfil, que permitem visitar um perfil ao serem clicados - quando um utilizador passa com o cursor por cima do Cartão, este muda de cor. Na figura 6.4 apresenta-se um Cartão de Perfil.

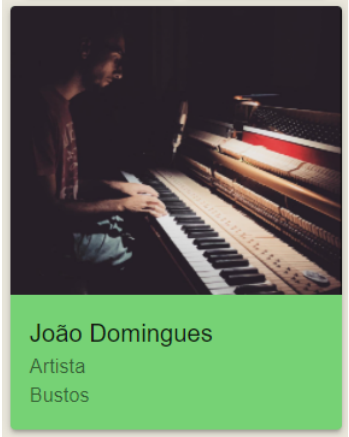


Figura 6.4: Exemplo de uma das representações possíveis de um Cartão de Perfil.

Componentes

A representação de componentes em *React* pode dar-se sob duas formas distintas - uma função ou uma classe.

Para a realização deste projeto, recorreu-se à implementação dos componentes sob a forma de função. De uma forma genérica, um componente representar-se-á sob a seguinte forma:

```
export default function ComponentName(props) {
    return (
        <>
            Conteudo deste componente
        </>
    );
}
```

Listagem 6.15: Implementação básica de um componente sob a forma de uma função.

O componente apresentado no excerto de código 6.15 está pronto para ser apresentado na *UI*, bem como a ser incorporado noutros componentes.

O desenvolvimento de componentes baseou-se em torno da separação de responsabilidades - cada componente pretende ter o mínimo de responsabilidades possíveis de forma a promover a reutilização dos mesmos. Estas responsabilidades não são apenas referentes a regras de negócio, mas também a simples elementos de *UI*.

A reusabilidade de componentes promove ainda a consistência da *UI* ao longo da aplicação - uma alteração visual a um determinado componente que é utilizado em várias páginas resulta na atualização de todas as páginas onde o componente é utilizado.

React Hooks

Embora o resultado da implementação de componentes a partir de classes ou funções seja o mesmo, há algumas diferenças a nível de implementação. Uma das diferenças mais cruciais baseia-se no facto de algumas funcionalidades mais recentes do *React* (versões $\leq$16.8) serem exclusivas às classes, sendo necessários **React Hooks** para aceder aos mesmos. Estes *Hooks* permitem acesso às funcionalidades de estados e de *lifecycle* presentes nas classes.

Desta forma, foram utilizados dois *Hooks*, com dois objetivos distintos - *useState* e *useEffect*.

O *useState* foi utilizado com a finalidade de atualizar o estado de um componente. Se um componente utiliza informação dinâmica, após haver uma atualização o componente deve também atualizar a *UI* e, quando aplicável, os componentes-filho. Quando um componente é montado (carregado por um componente-pai), é feita uma rotina inicial, com o objetivo de nos casos em que seja necessário um pedido ao *backend* ou qualquer tipo de processamento inicial, este seja feito no momento inicial do ciclo de vida de um componente. No excerto de código 6.16 apresenta-se um excerto de código que representa a página de perfil de um utilizador do tipo Artista.

```
1  export default function ProfileArtist(props) {
2
3      const [state, setState] = useState({ lastId: 0, projects: [] });
4
5      const getProjects = async () => {
6          var res = await axios.get(`${urlBackend}/artistas/${user.id}/
       projetos/`);
7          setState({ lastId: user.id, projects: res.data });
8      }
9
10     if (lastId !== user.id) getProjects();
11
12     [...]
13
14 };
```

Listagem 6.16: Componente com *useState* responsável pelo perfil de um artista.

O componente representado no excerto de código 6.16 está dependente de dois estados - o identificador do utilizador a quem pertence o perfil visitado e os projetos a que este faz parte. Tal pode ser verificado na linha 3. Na linha 10, verifica-se se o identificador do artista a ser visitado no momento é diferente do anterior, fazendo um pedido ao *backend*, requesitando os projetos do novo artista. Na linha 7, quando essa informação é recebida e atualizada, todo o componente é atualizado com a nova informação referente ao artista de quem se visita o perfil.

O *useEffect* foi usado em situações onde se pretendia efetuar uma operação periodicamente. O excerto de código 6.17 representa a utilização do *useEffect* para, de 15 em 15 segundos, fazer um pedido ao *backend* a requisitar novas notificações do utilizador com *login* efetuado.

```
export default function AutoUpdateNotifications() {

    [...]

    useEffect(() => {
        const interval = setInterval(() => {
            update();
        }, 15000); // 15 segundos

        return () => clearInterval(interval);
    }, [update]);

    [...]
}
```

Listagem 6.17: Componente com *useState* responsável pelo perfil de um artista.

O *useEffect* foi utilizado em componentes invisíveis que permaneciam ativos/montados enquanto há um utilizador com *login* efetuado.

Redux

Em *React*, o fluxo de informação é uni-direcional, no sentido em que componentes-pai podem transmitir informação aos seus componentes-filho (através da utilização de parâmetros), mas o contrário não é possível.

No entanto, há informações ou estados da aplicação que devem estar disponíveis em vários locais e níveis de componentes da aplicação - componentes esses que, muitas vezes, não têm qualquer tipo de ligação entre eles. Dois exemplos deste tipo de informação são: que utilizador está com o *login* feito e que definições da aplicação é que este utilizador tem selecionadas.

A utilização do *React Redux* disponibiliza um repositório de informação acessível por qualquer componente.

A utilização do *Redux* divide-se em 3 partes essenciais. Os excertos de código 6.18 representam estas 3 partes usando a funcionalidade de *login* de um utilizador como exemplo.

O *Redux* disponibiliza estas informações sob a forma de estados, armazenados em **Reducers**, que expõem o respetivo estado. Cada *reducer* pode receber **Actions**, que efetuam operações sobre o estado do *reducer*.

```
const loggedReducer = (state = null, action) => {
    switch(action.type) {
        case 'LOG_IN':
            return action.payload;
        case 'LOG_OUT':
            return null;
        default:
            return state;
    }
}

export default loggedReducer;
```

Listagem 6.18: Implementação do *Reducer* referente ao estado de *login* de um utilizador.

Nos *reducers* como os representado na listagem 6.18, as **Actions** tratam-se de uma função que retornam um *type* e um *payload*. O *type* indica ao *Reducer* que tipo de ação deve fazer com o seu estado, enquanto que o *payload* serve para transportar informação para o *Reducer*, caso necessário.

O excerto de código 6.19 apresenta duas *Actions* utilizadas pelo *loggedReducer* apresentado no excerto de código anterior. Uma delas é responsável por fazer o *login* de um utilizador, enquanto que a outra faz o *logout*.

```
export const login = (user) => {
    return {
        type: 'LOG_IN',
        payload: user
    }
}

export const logout = () => {
    return {
        type: 'LOG_OUT'
    }
}
```

Listagem 6.19: Implementação das *Actions* de *login* e *logout* de um utilizador.

É agora possível a qualquer componente aceder ao utilizador que está com o *login* feito na aplicação. O excerto de código 6.20 demonstra um exemplo de uma implementação que usa o estado do *loggedReducer*.

```
import { useSelector } from 'react-redux';

export default function ExemploComponent(props) {

    [...]

    const user = useSelector(state => state.logged);

    [...]

    if (user === null) return <></>;

    return (
        <>
            [...]
            user.nome
        </>
    );
}
```

Listagem 6.20: Utilização do *logged state* por um componente.

Adicionalmente, o excerto de código 6.20 demonstra ainda um exemplo de validação do *login* do utilizador para a utilização de um certo componente. A linha `if (user === null) return <></>;` retorna um componente vazio a um utilizador sem *login* feito quando este acede a, por exemplo, uma página que não tem permissão para ver (através do *link* direto à página, por exemplo).

6.1.5 Realização de testes às funcionalidades desenvolvidas

A realização de testes às funcionalidades desenvolvidas dá-se sob a forma de Testes de Aceitação, nos quais se realiza o Caso de Uso em vários cenários que representem diversos tipos de utilizador. Desta forma, no final de cada iteração, verifica-se a conformidade da aplicação para com os requisitos do utilizador.

Algumas funcionalidades-chave são apresentadas e discutidas com clientes interessados no desenvolvimento da aplicação.

Deste processo pode resultar um ajuste dos requisitos da aplicação, que são incluídos na iteração seguinte.

6.2 Sistema de Recomendação

Um dos principais objetivos da aplicação a ser desenvolvida é fornecer ferramentas que permitam facilitar a expansão da rede de contactos dos seus utilizadores, bem como a tomar decisões referentes às suas carreiras profissionais ou amadoras de forma mais informada.

Nesse sentido, criou-se um Sistema de Recomendações que assume dois formatos distintos, com objetivos distintos - Recomendação por Semelhança e Recomendação por Interesse de Negócio.

Ambos os sistemas de recomendação funcionam à base de um algoritmo totalmente modular que analisa utilizadores, eventos e projetos com base num conjunto de critérios pré-definido, gerando, para cada um, uma pontuação arbitrária e oculta ao utilizador final.

A falta de transparência do algoritmo para o utilizador final tem como objetivo a promoção da criação/gestão de perfis, eventos e projetos de forma honesta e verdadeira, sem haver adaptações irrealistas de forma a subir a pontuação.

6.2.1 Recomendação por Semelhança

O perfil de um determinado utilizador, projeto ou evento inclui uma lista limitada composta pelos *N* utilizadores/projetos/eventos mais semelhantes àquele cujo perfil se visita.

Cada perfil tem uma lista própria (como demonstra a figura 6.5) que é atualizada sempre que se faz uma visita ao mesmo. Isto significa que o sistema de pontuação de um determinado perfil nunca é absoluto, mas sim relativo - quando se compara o **perfil A** com o **perfil B**, obtém-se uma pontuação diferente de uma feita com uma comparação com o **perfil C**.

O objetivo deste tipo de recomendação destina-se a facilitar a expansão da rede de contactos dos utilizadores. Artistas podem encontrar artistas semelhantes de modo a formar novos projetos. Agentes e promotores podem encontrar projetos semelhantes àqueles com que trabalham. Projetos podem encontrar agentes e promotores que sejam do seu interesse.

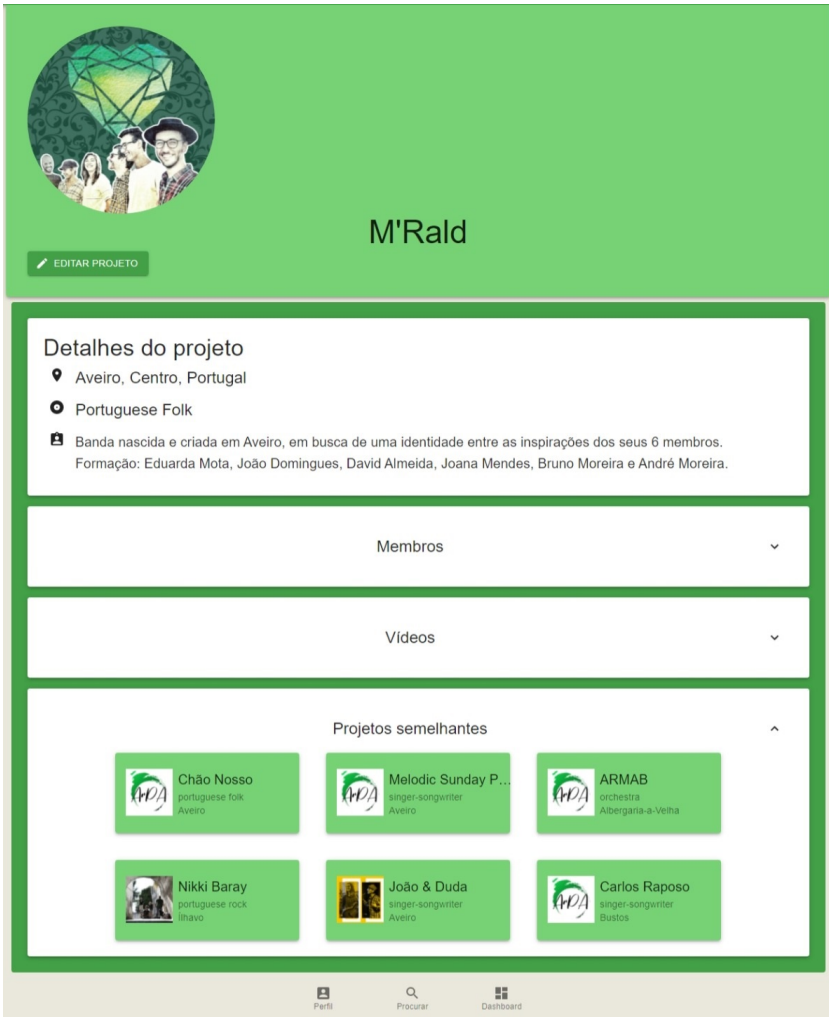


Figura 6.5: Lista de Recomendação por Semelhança num perfil de Projeto.

Embora o algoritmo para calcular a pontuação de cada perfil seja idêntico, cada tipo de perfil utiliza parâmetros de cálculo diferentes. Estes parâmetros, bem como o peso associado são arbitrários.

Por exemplo, os critérios utilizados para comparação entre projetos são:

- **Localização** - Proximidade geográfica entre projetos: quanto mais próximos, mais semelhantes;
- **Visitas de Perfil** - Sempre que um perfil é visitado, é registada uma nova visita. No caso de um perfil de um projeto, são apenas contabilizadas visitas de utilizadores não associados ao projeto. A similaridade entre o número de visitas contribui para a pontuação, utilizando uma fórmula que tem em conta a diferença absoluta e relativa entre os dois valores;
- **Género** - Quando é registado um projeto na aplicação com um género que ainda não existe, é feito um pedido a uma *API* [104]. Deste pedido resulta um *ranking* de géneros por ordem de similaridade ao novo género. A similaridade entre dois géneros é baseada neste *ranking*;
- **Elementos em Comum** - O número de elementos em comum contribui para a similaridade entre dois projetos;
- **Número de Artistas** - A proximidade entre o número de elementos dos projetos contribui para a pontuação de semelhança.

6.2.2 Recomendação por Interesse de Negócio

O segundo tipo de recomendação baseia-se em apresentar perfis relevantes para a tomada de uma determinada decisão para um determinado utilizador.

Este tipo de recomendação funciona de forma semelhante ao anterior, onde diferentes tipos de decisão recorrem a diferentes versões do mesmo algoritmo base, calculando uma pontuação final que resulta numa lista de apoio à decisão.

Um exemplo deste sistema está presente na *Dashboard* de artistas, tal como é visível na figura 6.6.

A figura 6.6 apresenta a *tab* "Projetos" presente na *Dashboard* de um Artista, que contém os projetos em que o artista faz parte. Os projetos que ainda não têm um Agente associado, contém uma *tab* de **Agentes Recomendados**. Aqui, está contida uma lista de *N* agentes que podem ser do interesse do projeto.

Este tipo de recomendação serve um propósito triplo:

- Tal como o tipo de recomendação anterior, pretende expandir a rede de contactos do utilizador (neste caso, um artista), ao apresentar soluções que podiam não ser do conhecimento do mesmo;
- Incentivar um dos objetivos da aplicação: dinamizar a indústria, promovendo interações entre artistas, promotores e agentes - neste caso, ao serem apresentados agentes para os seus projetos, um artista é relembrado que esta conexão é uma possibilidade de negócio;
- Apoiar a tomada de decisão de utilizadores menos experientes.

Figura 6.6: Lista de Recomendação por Interesse na *Dashboard* de um Artista.

6.3 Conclusões

Com a conclusão deste capítulo, o leitor tem agora uma melhor perceção da implementação do projeto ArPA.

Ao longo do capítulo foram narrados diferentes pormenores referentes ao desenvolvimento do projeto, numa demonstração que junta excertos de código e *screenshots* da *user interface*.

A implementação deste projeto focou-se em 3 pontos-chave:

- Promover boas práticas de programação;
- A Usabilidade do produto final;
- A implementação das funcionalidades apenas termina quando estiverem numa fase que apresente um elevado grau de qualidade.

Através da utilização de boas práticas de programação, a implementação das funcionalidades ficou progressivamente mais rápida e intuitiva, uma vez que a facilidade de manutenção, implementação de novas funcionalidades e de reutilização de código foram tidas em conta durante todo o processo.

Capítulo 7

Avaliação da Solução

Enquanto que no capítulo 4, a partir do contexto do problema, é feita a conceção, seleção e avaliação de ideias que levam a uma solução, no presente capítulo é avaliada a solução implementada e o modo como se encaixa nesse mesmo contexto e resolve o problema apresentado.

Aqui, delinear-se-á um modelo de avaliação sob o qual se analisará a solução desenvolvida. A partir desta avaliação averiguar-se-á a compatibilidade da solução com as necessidades do mercado e do cliente final.

Nesse sentido, será utilizada a metodologia *Quantitative Evaluation Framework (QEF)*.

7.1 Quantitative Evaluation Framework

Desenvolvido por Paula Escudeiro e José Bidarra [105], o modelo de avaliação *QEF* contextualiza a qualidade da solução implementada - **Solução Real** - confrontando-a com uma solução fictícia perfeita (à luz dos parâmetros de avaliação considerados) - **Solução Ideal**.

Este confronto é feito ao dispor ambas as soluções num espaço composto por três **Dimensões**, calculando-se a distância entre as duas soluções - quanto maior for esta distância, mais longe a **Solução Real** se encontra de ser comparável à **Solução Ideal**.

A utilização do *QEF* envolve três conceitos: **Dimensão**, **Fator** e **Requisito**. Estes três conceitos estão relacionados da seguinte forma: uma **Dimensão** é composta por **Fatores**, enquanto que um **Fator** é composto por **Requisitos**.

Um determinado **Requisito** é composto por dois elementos:

- *Avaliação* - Um valor percentual que representa a taxa de cumprimento do **Requisito**. A escala percentual divide-se em incrementos de 25 [1], estando atribuído a cada *step* uma descrição daquilo que é necessário verificar-se na solução para obtenção do mesmo;
- *Relevância* - O grau de relevância é um valor contido numa escala de 2 a 10 em incrementos de 2 [2], que representa a relevância do **Requisito** no contexto do **Fator** em que se insere.

Um determinado **Fator** é composto por dois elementos:

[1] [0, 25, 50, 75, 100].
[2] [2, 4, 6, 8, 10].

- *Importância* - O peso relativo de importância é calculado a partir de dois componentes: a soma dos graus de relevância dos **Requisitos** que compõem o **Fator** e o total, dos que compõem a **Dimensão**. Dividindo-se estes dois valores, obtém-se uma percentagem correspondente ao peso relativo de importância do **Fator** no contexto da **Dimensão**;
- *Cumprimento* - O peso relativo de cumprimento é calculado a partir da percentagem de *Avaliação* total obtida no conjunto de **Requisitos** que compõem o **Fator**.

O valor atribuído a uma determinada **Dimensão** é calculado a partir dos valores de *Importância* e *Cumprimento* dos **Fatores** que a compõem.

No âmbito do presente projeto, foram consideradas as seguintes dimensões:

- **Funcionalidade** - Composta pelos requisitos funcionais do projeto, apresentada na imagem 7.3;
- **Usabilidade** - Composta pelos requisitos de usabilidade do projeto, apresentada na imagem 7.1;
- **Suportabilidade** - Composta pelos requisitos de suportabilidade do projeto, apresentada na imagem 7.2;

Usabilidade	0,00	0,25	Linguagem	10,00	UL01 - Suporta as línguas portuguesa e inglesa como linguagem de utilização da plataforma
				10,00	UL02 - Utiliza terminologia existente e relevante no contexto da área de negócio
	0,00	0,50	Navegação	10,00	UN01 - A navegação entre diferentes menus deverá ser simples e ter um número reduzido de passos
				10,00	UN02 - Apresentação das páginas deverá ser feita rapidamente
				10,00	UN03 - Disposição clara das diferentes opções do utilizador
				10,00	UN04 - As ações do utilizador resultam num feedback visual
	0,00	0,25	Consistência	10,00	UC01 - Plataforma utiliza uma interface visualmente consistente ao longo das diferentes páginas
				10,00	UC02 - Os perfis dos utilizadores devem ter uma estrutura modular mas editável

Figura 7.1: Dimensão Usabilidade.

Suportabilidade	0,00	0,50	Adaptabilidade	10,00	SA01 - A plataforma está disponível em diferentes browsers
				10,00	SA02 - A plataforma deverá supor vários formatos de ficheiro audiovisual
	0,00	0,50	Manutenibilidade	10,00	SM01 - A implementação deverá estar acompanhada de documentação completa e detalhada
				10,00	SM02 - A implementação da plataforma deverá permitir a inclusão de novas funcionalidades de forma rápida e fácil

Figura 7.2: Dimensão Suportabilidade.

Funcionalidade	77,59	0,16	Networking	10,00	FNE01 - Permite adicionar utilizador à rede de contactos
				10,00	FNE02 - Permite remover utilizador da rede de contactos
				10,00	FNE03 - Permite ler mensagens de outros utilizadores
				10,00	FNE04 - Permite enviar mensagens para outros utilizadores
				10,00	FNE05 - Permite pesquisar por outros utilizadores
				8,00	FNE06 - Permite ver como as redes de contactos entre dois utilizadores se relacionam
	75,00	0,05	Utilizador	10,00	FUT01 - Permite criar uma conta de utilizador
				10,00	FUT02 - Permite editar o perfil de um utilizador
	78,95	0,11	Feedback do Sistema	10,00	FFS01 - Permite ler notificações do sistema
				10,00	FFS02 - O sistema notifica os utilizadores quando uma ação que lhes é relevante acontece
				10,00	FFS03 - O sistema redireciona o utilizador dentro da página de forma a demonstrar o resultado do evento de relevância que ocorreu
				8,00	FFS04 - Permite a notificação via email de eventos relevantes para os utilizadores
	37,18	0,22	Artista	10,00	FAR01 - Permite criar um projeto
				10,00	FAR02 - Permite editar um projeto
				10,00	FAR03 - Permite remover um projeto
				10,00	FAR04 - Permite gerar um rider técnico
				10,00	FAR05 - Permite candidatar a um evento
				10,00	FAR06 - Permite criar uma proposta
				10,00	FAR07 - Permite editar uma proposta
				8,00	FAR08 - Permite remover uma proposta
	20,00	0,14	Agente	10,00	FAG01 - Permite editar perfil de um cliente
				10,00	FAG02 - Permite adicionar um artista à carteira de artistas
				10,00	FAG03 - Permite remover um artista da carteira de artistas
				10,00	FAG04 - Permite gerar uma proposta a partir da sua carteira de artistas
				10,00	FAG05 - Permite apresentar a candidatura de um artista da sua carteira de artistas
	50,00	0,14	Promotor	10,00	FPR01 - Permite criar um evento
				10,00	FPR02 - Permite editar um evento
				8,00	FPR03 - Permite remover um evento
				10,00	FPR04 - Permite fechar um evento
				10,00	FPR05 - Permite selecionar candidatos para um evento
	75,00	0,11	Recomendação	10,00	FRE01 - Recomenda perfis semelhantes àquele que é visitado
				10,00	FRE02 - Recomenda perfis relevantes para apoio à decisão do Artista
				10,00	FRE03 - Recomenda perfis relevantes para apoio à decisão do Promotor
				10,00	FRE04 - Recomenda perfis relevantes para apoio à decisão do Agente
	33,33	0,08	Audiovisual	10,00	FAU01 - Permite a inclusão de ficheiros de som no perfil de um utilizador/projeto
				10,00	FAU02 - Permite a inclusão de ficheiros de vídeo no perfil de um utilizador/projeto
				10,00	FAU03 - Permite a inclusão de conteúdo audiovisual proveniente de outras plataformas

Figura 7.3: Dimensão Funcionalidade.

7.2 Fontes de Informação

Uma vez definidas as três dimensões a ser utilizadas no *QEF*, é necessário estabelecer que partes interessadas avaliam cada uma delas, definindo-se uma metodologia para cada uma.

Neste sentido, dividem-se as dimensões em dois conjuntos. O primeiro conjunto formado pelas dimensões **Funcionalidade** e **Suportabilidade**, que serão avaliadas pelo autor, enquanto que a dimensão **Usabilidade** será avaliada por um conjunto de utilizadores, com recurso a um questionário. Ambos os formatos de avaliação recorrem às métricas criadas para cada um dos Requisitos que compõem os Fatores das Dimensões referidas, disponíveis no Apêndice B.

7.3 Metodologia de Avaliação

Tal como referido na secção anterior, a avaliação da solução tem por base duas fontes distintas de avaliação, com metodologias distintas.

A avaliação das dimensões de **Funcionalidade** e **Suportabilidade** é feita recorrendo às métricas apresentadas no Apêndice B de forma direta. A tabela apresentada nas figuras 7.2 e 7.3 são preenchidas de acordo com a métrica que melhor representa o estado do projeto.

Por outro lado, a dimensão **Usabilidade** é avaliada por um conjunto de utilizadores, através da realização de um questionário que apresenta os *Requisitos* sob a forma de questões de escolha múltipla e as *Métricas* sob a forma de respostas. Para cada uma das questões é atribuída uma pontuação a cada resposta que representa a taxa de *fullfilment* do requisito. A pontuação final atribuída ao requisito é calculada a partir da média dos valores das respostas.

Como tal, a lista abaixo apresenta a lista de perguntas apresentada aos utilizadores:

- **UL02** - A linguagem e terminologia utilizadas na plataforma são reconhecíveis, consistentes com a utilização na indústria e estão bem aplicadas?
- **UN01** - Classifico a navegação dentro da plataforma como sendo:
- **UN02** - A plataforma é responsiva e as páginas carregam todo o seu conteúdo rapidamente?
- **UN03** - Como utilizador, consigo, a qualquer altura, saber quais são as opções que a plataforma me disponibiliza?
- **UN04** - A plataforma dá-me indicações visuais de que as minhas ações foram registadas e posso verificá-las imediatamente?
- **UC01** - A interface da plataforma é consistente ao longo das diferentes páginas e consigo reconhecer elementos entre diferentes páginas?
- **UC02** - Os perfis da plataforma apresentam a informação de forma organizada, modular e editável pelo utilizador do próprio perfil?

Note-se que o requisito **UL01** não foi incluído, uma vez que se trata de um requisito com uma métrica objetiva e, por isso, passível de ser respondida pelo autor, recorrendo à mesma metodologia das outras duas dimensões.

Este processo foi realizado ao longo do mês de Agosto e Setembro, de forma presencial, juntamente com os participantes. A lista de candidatos consiste em 11 participantes, que inclui os 3 entrevistados no capítulo 2.1.1.

A escolha de candidatos foi arbitrária, mas consiste exclusivamente de membros da indústria da música (profissionais e amadores).

O único dado recolhido sobre os participantes foi o tipo de utilizador que representaria dentro da plataforma.

A distribuição do tipo de utilizadores consiste na seguinte:

- 1 Agente;
- 2 Promotores;
- 8 Artistas.

O processo de demonstração do produto consistiu em 5 partes:

- Apresentação do produto;
- Demonstração do produto;
- Parte livre - aqui o utilizador estava livre de interagir com a plataforma. Era pedido para efetuar algumas ações (interagir com outros utilizadores, criar conta de utilizador, explorar outros perfis, ...);
- Resposta a possíveis questões;
- Responder ao questionário.

Embora as respostas ao questionário fossem feitas de forma presencial, as respostas foram dadas de forma privada (sem conhecimento do autor).

7.4 Resultados

Nesta secção é feita uma apresentação dos resultados obtidos e consequente análise. Para tal, apresentam-se na figura 7.4 os resultados obtidos no questionário realizado aos utilizadores selecionados.

	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9	User 10	User 11
UL02	100	100	100	100	100	100	100	100	100	100	100
UN01	100	100	50	50	100	100	50	100	100	100	100
UN02	100	50	100	100	100	50	100	100	50	100	100
UN03	100	100	50	50	100	100	50	100	100	100	100
UN04	50	100	50	100	100	50	50	50	100	50	100
UC01	100	100	50	100	100	100	100	100	100	100	100
UC02	50	50	50	50	50	50	50	50	50	50	100

Figura 7.4: Respostas ao questionário.

Tal como referido anteriormente, as respostas apresentadas aos utilizadores derivaram diretamente das métricas utilizadas nos requisitos do *QEF* (ver Apêndice B). Como tal, a

tabela apresentada na figura 7.4 apresenta já as respostas convertidas para o valor da métrica associada.

De forma a gerar um valor final para preenchimento do *QEF* a partir do conjunto de respostas obtidas para cada uma das questões, é feita uma análise para cada uma das questões. Para cada questão, é apresentada a distribuição de respostas, a média e mediana dos resultados, e é feito um cálculo do desvio padrão.

O valor final do requisito do *QEF* será extraído a partir da média dos resultados obtidos.

No entanto, antes da análise dos resultados, é importante referir dois aspetos importantes:

- Condições de teste a nível de *hardware* - Tal como referido anteriormente, as respostas aos questionários seguiram a utilização da plataforma, por parte dos candidatos, no computador do autor. Isto poderá impactar os resultados de duas formas:
  - Uma vez que os testes foram realizados em *hardware* com o qual os utilizadores não estavam familiarizados, em alguns casos poderá haver uma barreira adicional de utilização do produto;
  - Uma vez que as condições de utilização eram sempre as mesmas, os tempos de processamento da aplicação (relevante para a questão **UN02** do questionário) eram praticamente constantes e não apresentaram uma variação realista que seria verificável no caso de as demonstrações terem sido feitas no *hardware* dos candidatos.
- A execução dos testes, cronologicamente - Idealmente, a execução dos testes de usabilidade e respetiva resposta ao questionário seria feita no final de cada iteração. No entanto, uma vez que se trata de um projeto desenvolvido a solo, o autor não pode delegar este processo (que requer a participação de diversos candidatos) numa pessoa responsável. Como tal, a análise dos dados apresentada nas secções que se seguem representa o processo que teria lugar no final de cada iteração. Daqui, a equipa responsável pelo desenvolvimento do projeto concluiria se a iteração seria refeita (de um ponto de vista de Usabilidade), com base na comparação dos resultados de performance dos utilizadores com diferentes valores de *threshold*. Adicionalmente incluiria uma análise da satisfação geral da utilização da plataforma [106].

O autor está ciente de que os fatores acima podem ter algum impacto na interpretação dos resultados obtidos ao longo das seguintes secções.

7.4.1 Requisito UL02

Questão: A linguagem e terminologia utilizadas na plataforma são reconhecíveis, consistentes com a utilização na indústria e estão bem aplicadas?

Respostas possíveis e respetiva pontuação associada:

- **100%** - Concordo
- **0%** - Não concordo

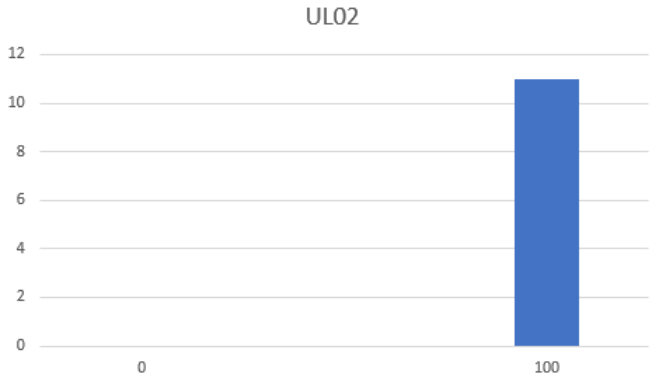


Figura 7.5: Respostas à questão referente ao Requisito UL02.

As respostas a esta questão (apresentadas na figura 7.5) foram unânimes - todos os utilizadores concordam que a terminologia e linguagem utilizada na plataforma é reconhecível a nível da indústria da música.

Por isso mesmo, o valor atribuído a este requisito será **100%**.

7.4.2 Requisito UN01

Questão: Classifico a navegação dentro da plataforma como sendo:

Respostas possíveis e respetiva pontuação associada:

- **100%** - Navegação simples, intuitiva e consistente com o modo de operação normal
- **50%** - Navegação simples, com algumas inconsistências para com o modo de operação normal
- **0%** - Navegação complexa ou pouco intuitiva e não consistente com o modo de operação normal

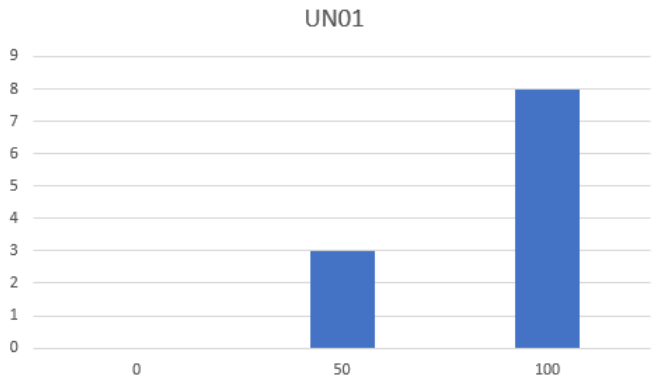


Figura 7.6: Respostas à questão referente ao Requisito UN01.

Nesta questão, cujas respostas se apresentam na figura 7.6, houve um conjunto de 3 utilizadores que apontaram algumas inconsistências entre o modo de funcionamento da plataforma e o modo como operam na indústria. Uma vez que, tal como apontado na secção 2.1.2, se trata de uma vertente da indústria em que não estão estabelecidas convenções de negócio,

a discordância por parte de alguns participantes é um resultado esperado e que valida esta mesma afirmação.

Destes dados retiram-se as seguintes estatísticas:

- **Valor mais elevado:** 100%
- **Valor mais baixo:** 50%
- **Média:** 86.36%
- **Mediana:** 100%
- **Desvio Padrão:** 22.27%

7.4.3 Requisito UN02

Questão: A plataforma é responsiva e as páginas carregam todo o seu conteúdo rapidamente?

Respostas possíveis e respetiva pontuação associada:

- **100%** - A plataforma comporta-se conforme esperado, de forma rápida.
- **50%** - A plataforma comporta-se conforme esperado, com alguns elementos com tempo de carregamento algo demorado.
- **0%** - A plataforma não responde conforme esperado.

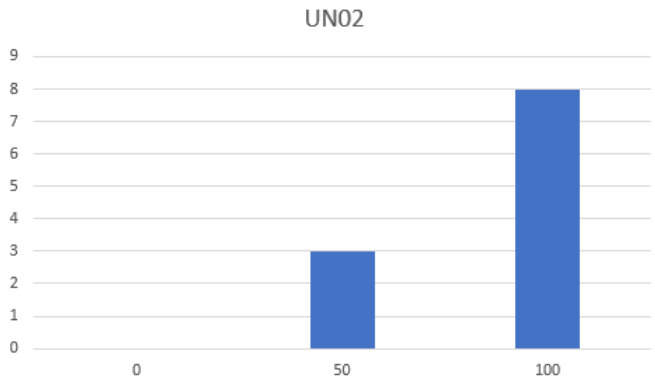


Figura 7.7: Respostas à questão referente ao Requisito UN02.

Nesta questão, cujas respostas se apresentam na figura 7.7, houve também um conjunto de utilizadores que apontaram um tempo de carregamento de alguns elementos da plataforma algo demorado, nomeadamente, as fotos de perfil e algumas mudanças no esquema de cores da plataforma. Ainda assim, a maioria dos utilizadores consideraram que a plataforma responde às suas ações de forma espectável e rápida.

Destes dados retiram-se as seguintes estatísticas:

- **Valor mais elevado:** 100%
- **Valor mais baixo:** 50%
- **Média:** 86.36%

- **Mediana:** 100%
- **Desvio Padrão:** 22.27%

7.4.4 Requisito UN03

Questão: Como utilizador, consigo, a qualquer altura, saber quais são as opções que a plataforma me disponibiliza?

Respostas possíveis e respetiva pontuação associada:

- **100%** - A plataforma apresenta todas as opções de utilização de forma clara.
- **50%** - A plataforma apresenta grande parte das opções de utilização de forma clara.
- **0%** - A plataforma apresenta uma pequena parte das opções de utilização de forma clara.

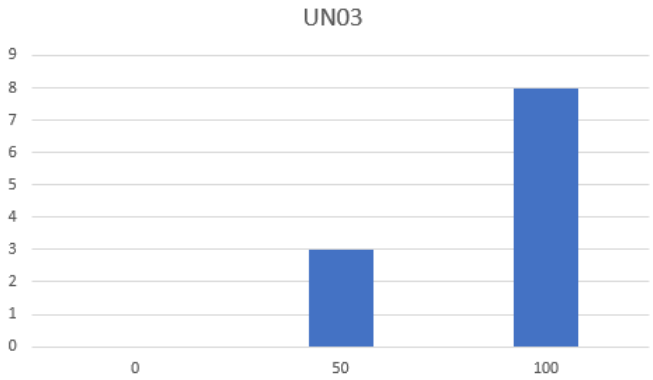


Figura 7.8: Respostas à questão referente ao Requisito UN03.

Nesta questão, cujas respostas se apresentam na figura 7.8, houve também um conjunto de utilizadores que não considera que todas as opções de utilização disponibilizadas pela plataforma são apresentadas de forma clara. Um detalhe importante a apontar neste conjunto de respostas surge quando se analisa a tabela total de respostas, na figura 7.4. Aqui, pode-se constatar que este conjunto de utilizadores é o mesmo que, na questão *UN01*, considerou que a plataforma apresenta algumas inconsistências para com o seu modo de operação. Estes resultados demonstram, mais uma vez, que se trata de uma vertente da indústria aberta a interpretação.

Destes dados retiram-se as seguintes estatísticas:

- **Valor mais elevado:** 100%
- **Valor mais baixo:** 50%
- **Média:** 86.36%
- **Mediana:** 100%
- **Desvio Padrão:** 22.27%

7.4.5 Requisito UN04

Questão: A plataforma dá-me indicações visuais de que as minhas ações foram registadas e posso verificá-las imediatamente?

Respostas possíveis e respetiva pontuação associada:

- **100%** - A plataforma informa-me sempre que concluí qualquer ação e mostra-me onde posso ver o resultado.
- **50%** - A plataforma, em algumas circunstâncias, informa-me que concluí uma ação e mostra-me onde posso ver o resultado.
- **0%** - A plataforma não me informa que concluí uma ação.

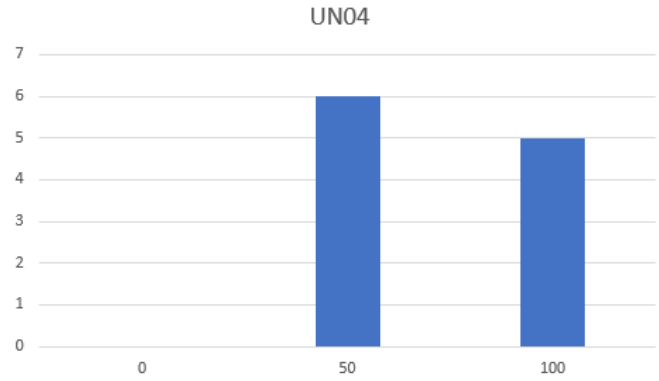


Figura 7.9: Respostas à questão referente ao Requisito UN04.

Esta questão, cujas respostas se apresentam na figura 7.9, apresenta os resultados mais divididos. Aqui, o autor conclui que esta polarização poderá depender do grau de intuição que cada utilizador apresenta para com a utilização geral de plataformas *online* - utilizadores mais adaptados/habituados a lidar com plataformas *online* poderão ser mais sensíveis à ideia de que uma operação já terminou e que foi feita com sucesso, enquanto que outros utilizadores poderão estar dependentes de *queues* mais destacadas.

Independentemente deste facto, os dados indicam que a plataforma carece de *feedback* para com o utilizador.

Destes dados retiram-se as seguintes estatísticas:

- **Valor mais elevado:** 100%
- **Valor mais baixo:** 50%
- **Média:** 72.73%
- **Mediana:** 50%
- **Desvio Padrão:** 24.90%

7.4.6 Requisito UC01

Questão: A interface da plataforma é consistente ao longo das diferentes páginas e consigo reconhecer elementos entre diferentes páginas?

Respostas possíveis e respetiva pontuação associada:

- **100%** - A plataforma é totalmente consistente ao longo das diferentes páginas.
- **50%** - A plataforma apresenta alguma variância em algumas páginas.
- **0%** - A plataforma não é consistente ao longo das diferentes páginas.

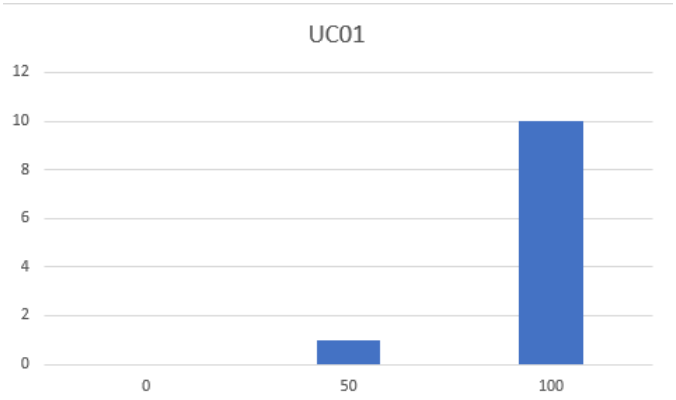


Figura 7.10: Respostas à questão referente ao Requisito UC01.

As respostas a esta questão foram praticamente unânimes, tal como demonstrado na figura 7.10, com apenas um utilizador a indicar que a plataforma não é totalmente consistente a nível de interface.

Destes dados retiram-se as seguintes estatísticas:

- **Valor mais elevado:** 100%
- **Valor mais baixo:** 50%
- **Média:** 95.45%
- **Mediana:** 100%
- **Desvio Padrão:** 14.37%

7.4.7 Requisito UC02

Questão: Os perfis da plataforma apresentam a informação de forma organizada, modular e editável pelo utilizador do próprio perfil?

Respostas possíveis e respetiva pontuação associada:

- **100%** - Os perfis estão bem organizados e as opções de edição e ordenação apresentadas são satisfatórias.
- **50%** - Os perfis estão bem organizados, mas as opções de edição e ordenação apresentadas não são satisfatórias.
- **0%** - Os perfis não estão bem organizados.

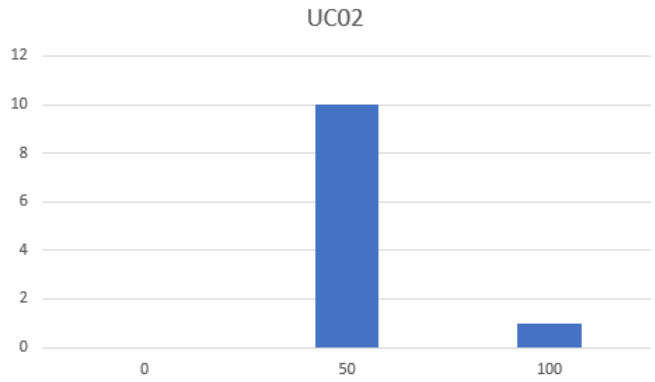


Figura 7.11: Respostas à questão referente ao Requisito UC02.

As respostas a esta questão, apresentadas na figura 7.11, indicam que o requisito *UC02* representa a maior lacuna a nível de Usabilidade da plataforma - a falta de opções de edição do perfil de um utilizador. Este conjunto de respostas é ainda suportado pela pontuação atribuída pelo autor no requisito *FUT02 - Permite editar o perfil de um utilizador* (ver figura 7.14), que evidencia que a funcionalidade ainda não está implementada na íntegra.

Destes dados retiram-se as seguintes estatísticas:

- **Valor mais elevado:** 100%
- **Valor mais baixo:** 50%
- **Média:** 54.55%
- **Mediana:** 50%
- **Desvio Padrão:** 14.37%

7.4.8 Resultados Questionário

A conversão dos resultados obtidos nas respostas às questões apresentadas aos utilizadores é feita utilizando a média dos valores associados às respostas. Como tal, o preenchimento do *QEF* resulta tabela representada na figura 7.12.

80,539	**Usabilidade**	81,25	0,25	Linguagem	6,00	UL01 - Suporta as línguas portuguesa e inglesa como linguagem de utilização da plataforma	50
					10,00	UL02 - Utiliza terminologia existente e relevante no contexto da área de negócio	100
		82,95	0,50	Navegação	10,00	UN01 - A navegação entre diferentes menus deverá ser simples e ter um número reduzido de passos	86,36
					10,00	UN02 - Apresentação das páginas deverá ser feita rapidamente	86,36
					10,00	UN03 - Disposição clara das diferentes opções do utilizador	86,36
					10,00	UN04 - As ações do utilizador resultam num feedback visual	72,73
		75,00	0,25	Consistência	10,00	UC01 - Plataforma utiliza uma interface visualmente consistente ao longo das diferentes páginas	95,45
					10,00	UC02 - Os perfis dos utilizadores devem ter uma estrutura modular mas editável	54,55

Figura 7.12: Dimensão Usabilidade preenchida com os resultados dos utilizadores.

Por fim, apresenta-se, na figura 7.13, a lista de *Requisitos* do *QEF* preenchida com a avaliação feita pelo autor. Primeiro, a dimensão da **Suportabilidade**:

76,39	**Suportabilidade**	75,00	0,50	Adaptabilidade	10,00	SA01 - A plataforma está disponível em diferentes browsers	100
					10,00	SA02 - A plataforma deverá supor vários formatos de ficheiro audiovisual	50
		77,78	0,50	Manutenibilidade	8,00	SM01 - A implementação deverá estar acompanhada de documentação completa e detalhada	50
					10,00	SM02 - A implementação da plataforma deverá permitir a inclusão de novas funcionalidades de forma rápida e fácil	100

Figura 7.13: Dimensão Suportabilidade preenchida com a avaliação do autor.

De seguida, a dimensão da **Funcionalidade**.

53,48	**Funcionalidade**	77,59	0,16	Networking	10,00	FNE01 - Permite adicionar utilizador à rede de contactos	100
					10,00	FNE02 - Permite remover utilizador da rede de contactos	50
					10,00	FNE03 - Permite ler mensagens de outros utilizadores	100
					10,00	FNE04 - Permite enviar mensagens para outros utilizadores	100
					10,00	FNE05 - Permite pesquisar por outros utilizadores	100
					8,00	FNE06 - Permite ver como as redes de contactos entre dois utilizadores se relacionam	0
		75,00	0,05	Utilizador	10,00	FUT01 - Permite criar uma conta de utilizador	100
					10,00	FUT02 - Permite editar o perfil de um utilizador	50
		78,95	0,11	Feedback do Sistema	10,00	FFS01 - Permite ler notificações do sistema	100
					10,00	FFS02 - O sistema notifica os utilizadores quando uma ação que lhes é relevante acontece	100
					10,00	FFS03 - O sistema redireciona o utilizador dentro da página de forma a demonstrar o resultado do evento de relevância que ocorreu	100
					8,00	FFS04 - Permite a notificação via email de eventos relevantes para os utilizadores	0
		37,18	0,22	Artista	10,00	FAR01 - Permite criar um projeto	100
					10,00	FAR02 - Permite editar um projeto	50
					10,00	FAR03 - Permite remover um projeto	50
					10,00	FAR04 - Permite gerar um rider técnico	0
					10,00	FAR05 - Permite candidatar a um evento	50
					10,00	FAR06 - Permite criar uma proposta	0
					10,00	FAR07 - Permite editar uma proposta	0
					8,00	FAR08 - Permite remover uma proposta	50
		20,00	0,14	Agente	10,00	FAG01 - Permite editar perfil de um cliente	0
					10,00	FAG02 - Permite adicionar um artista à carteira de artistas	50
					10,00	FAG03 - Permite remover um artista da carteira de artistas	50
					10,00	FAG04 - Permite gerar uma proposta a partir da sua carteira de artistas	0
					10,00	FAG05 - Permite apresentar a candidatura de um artista da sua carteira de artistas	0
		50,00	0,14	Promotor	10,00	FPR01 - Permite criar um evento	100
					10,00	FPR02 - Permite editar um evento	50
					8,00	FPR03 - Permite remover um evento	50
					10,00	FPR04 - Permite fechar um evento	50
					10,00	FPR05 - Permite selecionar candidatos para um evento	0
		75,00	0,11	Recomendação	10,00	FRE01 - Recomenda perfis semelhantes àquele que é visitado	100
					10,00	FRE02 - Recomenda perfis relevantes para apoio à decisão do Artista	100
					10,00	FRE03 - Recomenda perfis relevantes para apoio à decisão do Promotor	50
					10,00	FRE04 - Recomenda perfis relevantes para apoio à decisão do Agente	50
		33,33	0,08	Audiovisual	10,00	FAU01 - Permite a inclusão de ficheiros de som no perfil de um utilizador/projeto	0
					10,00	FAU02 - Permite a inclusão de ficheiros de vídeo no perfil de um utilizador/projeto	50
					10,00	FAU03 - Permite a inclusão de conteúdo audiovisual proveniente de outras plataformas	50

Figura 7.14: Dimensão Funcionalidade preenchida com a avaliação do autor.

A partir dos resultados obtidos para cada uma das dimensões - **Funcionalidade** (53,48%), **Suportabilidade** (76,39%) e **Usabilidade** (80,54%) - é possível calcular o índice que representa a qualidade associada com a **Solução Real** a partir da fórmula 7.1:

$$q = \frac{\log(\frac{1+53,48}{100}) + \log(\frac{1+76,39}{100}) + \log(\frac{1+80,54}{100})}{3 * \log(2)} = 76\% \quad (7.1)$$

Então, a **Solução Real** encontra-se implementada a **76%** da **Solução Ideal**.

7.5 Conclusões

Com a conclusão deste capítulo, o leitor tem agora uma melhor perceção do processo utilizado para averiguar a qualidade da solução implementada, sendo apresentados os resultados no final.

A partir destes resultados, é possível concluir que:

As dimensões **Funcionalidade** e **Suportabilidade** têm os valores de avaliação mais baixos. Este número reduzido de funcionalidades totalmente desenvolvidas deriva do facto de se tratar de um projeto com uma elevada carga de trabalho associada. Adicionalmente, o *overhead* do projeto é, também ele, elevado - isto significa que há uma grande quantidade de trabalho a ser desenvolvido antes de se poder começar a desenvolver as funcionalidades de relevância para a área de negócio em questão.

A dimensão de **Usabilidade** apresenta o valor mais alto de avaliação. Este valor é justificado pela metodologia de implementação da solução, que visava focar-se principalmente na qualidade, em detrimento da quantidade (diminuindo assim o número de funcionalidades implementadas). Como tal, embora os utilizadores não tenham acesso a um vasto leque de funcionalidades, aquelas a que já têm acesso estão com um nível elevado de usabilidade.

Capítulo 8

Conclusões

Uma vez terminado o projeto, é importante olhar para o trabalho desenvolvido e analisar os resultados produzidos nas diferentes fases do processo. Neste capítulo é feita uma análise dos objetivos concretizados, com a finalidade de averiguar o sucesso do desenvolvimento do projeto. Adicionalmente, são apresentadas algumas limitações verificadas nos resultados produzidos e são sugeridos alguns pontos de melhoria e trabalho futuro. Por fim, é feita uma apreciação final do projeto no seu todo, com base na opinião exclusiva do autor da presente dissertação.

8.1 Objetivos concretizados

De forma a averiguar o sucesso do projeto em mãos, é essencial avaliar se os objetivos inicialmente traçados foram alcançados com sucesso. Nesse sentido, serão analisados os objetivos traçados na secção 2.1.2.

- **Análise do funcionamento e estado da indústria para recolha de requisitos** - Levantamento de requisitos junto dos *stakeholders*, de forma a concluir qual o tipo de funcionalidades a ser incluído na aplicação *web*;
- **Análise de soluções existentes no mercado** - Análise do funcionamento de soluções já existentes no mercado e do seu modo de funcionamento, identificando lacunas e/ou pontos de melhoria;
- **Design de uma arquitetura robusta e escalável e respetiva implementação** - Desenhar e implementar a solução, aplicando boas práticas de desenvolvimento de software.

O sucesso do primeiro objetivo é visível ao longo dos capítulos 2.1.2 e 3 onde, a partir dos testemunhos recolhidos de profissionais da indústria da música selecionados para o efeito, foi possível construir uma imagem fidedigna do modo de funcionamento da indústria, bem como definir requisitos para a plataforma claros e concisos. O resultado do cumprimento deste objetivo serviu de base para as fases seguintes de todo o processo.

O segundo objetivo vê o seu sucesso descrito ao longo da secção 2.2, onde se exploram soluções sob dois pontos de vista diferentes - *networking* e funcionalidades próprias para a indústria. Nesta secção analisa-se o que cada solução traz para o mercado, identificando-se os pontos positivos e lacunas existentes de forma a criar uma ideia do tipo de plataforma a desenvolver. Como resultado do cumprimento deste objetivo, foi possível averiguar que dois pontos-chave:

- Não existem plataformas que explorem, simultaneamente, as duas ideias acima apresentadas - *networking* e funcionalidades próprias para a indústria;

- As soluções disponíveis no mercado, na sua maioria ou até totalidade, não disponibilizam funcionalidades para os *stakeholders* da plataforma, nomeadamente, os agentes e promotores.

Ao longo do capítulo 5 foram exploradas várias opções arquiteturais de forma a alcançar uma arquitetura robusta e escalável que permita responder ao enquadramento da plataforma em mãos. De igual forma, no capítulo 6 está detalhado o processo de implementação, que se foca, primariamente, em 3 fatores essenciais:

- Aplicação de boas práticas de desenvolvimento de *software*;
- Manutenibilidade da plataforma;
- Possibilidade de desenvolvimento futuro fácil e fluído.

Uma vez que todos os objetivos foram alcançados, o autor considera que a execução do projeto em mãos foi bem sucedida.

8.2 Limitações e trabalho futuro

Ainda que os objetivos descritos na secção anterior tenham sido atingidos com sucesso, não significa que não haja pontos de melhoria e espaços para desenvolvimento futuro a explorar.

Em primeiro, abordam-se as limitações identificadas pelo autor.

A única limitação apontada pelo autor reflete-se no número reduzido de Casos de Uso implementados. Esta limitação deve-se a dois fatores:

- **Overhead do projeto** - Este é, na opinião do autor, o fator de maior contribuição para a limitação apresentada. O projeto ArPA é possível ser descrito como uma "rede social talhada para o mundo da indústria musical" - como tal, antes de ser possível desenvolver as funcionalidades de relevância para a indústria da música, foi necessário desenvolver um conjunto elevado de funcionalidades básicas associadas a uma rede social genérica;
- **Requisitos Não-Funcionais com elevado custo de implementação** - Alguns Requisitos Não-Funcionais listados na secção 3.2 apresentaram um elevado custo temporal de implementação. Um exemplo deste fator é a implementação do Sistema de Recomendações detalhado ao longo da secção 6.2;
- **Qualidade primeiro, quantidade depois** - Ao longo do desenvolvimento do projeto, foi dado especial ênfase à criação de funcionalidades que apresentassem um elevado nível de qualidade. Em muitos casos, a distribuição do tempo planeado focou-se no aprimoramento de casos de uso já desenvolvidos em vez da adição de novas funcionalidades;

Como tal, a limitação apresentada é considerada como um ponto a desenvolver como trabalho futuro, pelo autor. A nível de trabalho futuro, o autor considera os seguintes pontos:

- **Implementação de mais Casos de Uso** - Tal como referido acima, o número de Casos de Uso implementados foi reduzido. De forma a criar um produto que vá ao encontro das expetativas dos *stakeholders*, na sua totalidade, é necessário implementar mais funcionalidades;

- **Implementação de mais Requisitos Não-Funcionais** - De igual modo, é importante ainda incluir Requisitos Não-Funcionais que ainda não foram implementados;
- **Autorização e Autenticação** - Uma vez que o produto desenvolvido no contexto da presente dissertação se tratar de um *Minimum Viable Product (MVP)*, não foram abordados conceitos de Autenticação nem de Autorização na implementação do mesmo;
- **Marketing** - Uma vez desenvolvidos os pontos acima, tal como referido na secção 4.2.4, o desenvolvimento de uma plataforma de raiz implica uma reduzida quantidade e utilizadores no início do ciclo de vida do projeto. Como tal, a plataforma desenvolvida necessitará de um processo de *Marketing* que atraia utilizadores.

8.3 Apreciações Finais

Por fim, uma avaliação pessoal da experiência do desenvolvimento do projeto no seu todo, por parte do autor.

É possível destacar vários fatores para o sucesso e balanço positivo da experiência, desde as capacidades técnicas desenvolvidas, passando pelo resultado obtido e, pelo caminho, os contactos criados.

A experiência adquirida ao trabalhar com tecnologias proeminentes no mercado do *web development* será crucial no futuro, bem como o desenvolvimento da sensibilidade de abordar pessoas e tentar extrair delas uma necessidade que possa ser traduzida num produto que, de uma forma ou outra, consegue melhorar um pouco mais o mundo (por muito pequena que seja a escala). Destaca-se também a experiência auto-didata adquirida ao longo de todo o projeto, que foi executado a solo.

A satisfação com o produto desenvolvido é imensa - trata-se de um produto que surge de necessidades reais, de pessoais reais, num contexto exclusivo e sem precedentes. Ainda que não se encontre num estado de disponibilização para o público, é uma excelente base para progresso futuro.

Por fim, é ainda importante referir o *networking* efetuado ao longo de todo este processo. O processo de criação de um plataforma que, em grande parte, pretende potenciar as capacidades de *networking*, permitiu ao autor conhecer pessoas novas na indústria e, de forma registada ou não, estabelecer contactos para potenciais oportunidades futuras.

Bibliografia

[1] Augusto Mateus et al. «O sector cultural e criativo em Portugal». Em: *AMAS de Consultores, Ed.) O Sector Cultural e Criativo em Portugal* (2010), pp. 1–24.

[2] Mark Daman Thomas. «Digital performances Live-streaming music and the documentation of the creative process». Em: *The Future of Live Music* (2020).

[3] Jan Baetens. «Selling Digital Music, Formatting Culture». Em: *Leonardo* 50.1 (2017), pp. 108–109. doi: 10.1162/leon_r_01369.

[4] Paul Rutter. *The music industry handbook*. Routledge, 2016.

[5] Kashif Naveed, Chihiro Watanabe e Pekka Neittaanmäki. «Co-evolution between streaming and live music leads a way to the sustainable growth of music industry – Lessons from the US experiences». Em: *Technology in Society* 50 (2017). issn: 0160-791X. doi: https://doi.org/10.1016/j.techsoc.2017.03.005. url: https://www.sciencedirect.com/science/article/pii/S0160791X17300118.

[6] Tânia Leão et al. «Em suspenso: reflexões sobre o trabalho artístico, cultural e criativo na era Covid-19». Em: (2020).

[7] *Apoios à Cultura*. Jan. de 2021. url: https://covid19estamoson.gov.pt/cultura/.

[8] *Apresentação Apoios Novo Confinamento - Ministério da Cultura*. Jan. de 2021.

[9] Daniel Morgado Sampaio. «Compasso de espera. A indústria musical em 2020». Em: ().

[10] João Domingues. *Entrevista com Nuno Fontes*. Nov. de 2020.

[11] João Domingues. *Entrevista com Jorge Loura*. Out. de 2020.

[12] João Domingues. *Entrevista com Mabília Martins*. Out. de 2020.

[13] Ken Peffers et al. «A design science research methodology for information systems research». Em: *Journal of management information systems* 24.3 (2007), pp. 45–77.

[14] *Souq*. url: https://www.facebook.com/Souqmusic.

[15] *47 de Fevereiro*. url: https://www.facebook.com/47defevereiro.

[16] *Troll's Toy*. url: https://www.facebook.com/TrollsToy.

[17] Nero. «Jorge Loura, Guitarristas Preferidos & Influências». Em: *A Arte Sonora* (mai. de 2020). url: https://artesonora.pt/featured/jorge-loura-guitarristas-preferidos-influencias/.

[18] 2018. «REVISTA:10 Anos de Arte Sonora, 10 Grandes Guitarristas Portugueses». Em: *A Arte Sonora* (). url: https://artesonora.pt/featured/10-anos-arte-sonora-novo-coleccionavel-ja-disponivel/.

[19] *Mabília Martins*. url: https://www.linkedin.com/in/maby-martins-199724157/.

[20] *A Arte Sonora - Sobre*. Jan. de 2021. url: https://artesonora.pt/a-arte-sonora/.

[21] Paul Allen. *Artist management for the music business*. Routledge, 2018.

[22] *Music Agent*. url: https://www.berklee.edu/careers/roles/agent.

[23] *Fundação GDA - Sobre*. Jan. de 2021. url: https://www.fundacaogda.pt/fundacao-gda/.

[24] *GDA - Sobre*. Jan. de 2021. url: https://www.gda.pt/gda/.

[25] *Portal do Artista Homepage*. url: https://www.portaldoartista.pt/.
[26] *Artistas ao Sul Homepage*. url: https://artistasaosul.pt/.
[27] *Portal de Artistas Homepage*. url: https://portaldeartistas.pt/.
[28] *Central de Artistas Homepage*. url: http://www.centraldeartistas.pt/.
[29] *LANDR Homepage*. url: https://www.landr.com/pt/.
[30] *Spotify Homepage*. url: https://www.spotify.com/.
[31] *Apple Music Homepage*. url: https://www.apple.com/apple-music/.
[32] *Deezer Homepage*. url: https://www.deezer.com/.
[33] *Facebook Homepage*. url: https://www.facebook.com/.
[34] *Twitter Homepage*. url: https://www.twitter.com/.
[35] *Instagram Homepage*. url: https://www.instagram.com/.
[36] *LinkedIn Homepage*. url: https://www.linkedin.com/.
[37] *About Facebook*. Jan. de 2021. url: https://about.fb.com/company-info/.
[38] Yu-Ping Chiu. «Social Recommendations for Facebook Brand Pages». Em: *Journal of theoretical and applied electronic commerce research* 16.1 (2021). doi: 10.4067/s0718-18762021000100106.
[39] Published by H. Tankovska e Feb 2. *Facebook MAU worldwide 2020*. Fev. de 2021. url: https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/.
[40] Nils Wlömert e Dominik Papies. «On-demand streaming services and music industry revenues — Insights from Spotifys market entry». Em: *International Journal of Research in Marketing* 33.2 (2016), pp. 314–327. doi: 10.1016/j.ijresmar.2015.11.002.
[41] *Portal do Artista Concursos*. Jan. de 2021. url: https://www.portaldoartista.pt/FGDA_Artist/AllContestEntry.aspx.
[42] *Artistas ao Sul Registo*. url: https://artistasaosul.pt/registo/.
[43] *About Twitter*. Jan. de 2021. url: https://about.twitter.com/en/who-we-are/our-company.
[44] *About Instagram*. Jan. de 2021. url: https://about.instagram.com/.
[45] *About LinkedIn*. Jan. de 2021. url: https://about.linkedin.com/.
[46] *About Spotify*. Jan. de 2021. url: https://newsroom.spotify.com/company-info/.
[47] *About Apple Music*. Jan. de 2021. url: https://www.apple.com/apple-music/.
[48] *About Deezer*. Jan. de 2021. url: https://www.deezer.com/pt/company/about.
[49] *About LANDR*. Jan. de 2021. url: https://support.landr.com/hc/pt/articles/360004988813-O-que-%C3%A9-LANDR-.
[50] *Sobre Portal do Artista (Fundação GDA)*. Jan. de 2021. url: https://www.fundacaogda.pt/fundacao-gda/.
[51] *Sobre Portal de Artistas*. Jan. de 2021. url: https://portaldeartistas.pt/FAQ/sobre-nos.
[52] *Artistas ao Sul Sobre*. Jan. de 2021. url: https://artistasaosul.pt/sobre/.
[53] *Central de Artistas Sobre*. Jan. de 2021. url: http://www.centraldeartistas.pt/index.php?option=com_content&view=article&id=1&Itemid=70.
[54] *Portal de Artistas Contacto*. url: https://portaldeartistas.pt/contacto.
[55] *Vue JS Homepage*. Fev. de 2021. url: https://vuejs.org/.
[56] *React Homepage*. Fev. de 2021. url: https://reactjs.org/.
[57] *Angular Homepage*. Fev. de 2021. url: https://angular.io/.

[58] Krausest. *krausest/js-framework-benchmark: A comparison of the perfomance of a few popular javascript frameworks*. url: https://github.com/krausest/js-framework-benchmark.
[59] *State of JS 2020*. url: https://2020.stateofjs.com/en-US/technologies/front-end-frameworks/.
[60] *Spring Homepage*. Fev. de 2021. url: https://spring.io/.
[61] *Django Homepage*. Fev. de 2021. url: https://www.djangoproject.com/.
[62] *NodeJS Homepage*. Fev. de 2021. url: https://nodejs.org/en/.
[63] url: https://web-frameworks-benchmark.netlify.app/compare?f=django,express,spring.
[64] *Image and Video Upload, Storage, Optimization and CDN*. Set. de 2021. url: https://cloudinary.com/.
[65] A. J. Russo. *Amazon Simple Storage Service*. 2003. url: https://aws.amazon.com/pt/s3/.
[66] Ryan Brown et al. «TECHNICAL RIDER». Em: (1930).
[67] Lawrence Chung et al. *Non-functional requirements in software engineering*. Vol. 5. Springer Science & Business Media, 2012.
[68] Peter Eeles. «Capturing architectural requirements». Em: *IBM Rational developer works* (2005).
[69] Mervi Hasu et al. «Critical transition from developers to users: Activity-theoretical studies of interaction and learning in the innovation process». Em: (2001).
[70] Lawrence D Miles. *Techniques of value analysis and engineering*. Miles Value Foundation, 2015.
[71] Peter A Koen et al. «Fuzzy front end: effective methods, tools, and techniques». Em: *The PDMA toolbook* 1 (2002), pp. 5–35.
[72] Peter A Koen et al. «Fuzzy front end: effective methods, tools, and techniques». Em: *The PDMA toolbook* 1 (2002), pp. 5–35.
[73] Donald G Reinertsen e Preston Smith. *Developing products in half the time*. Van Nostrand Reinhold New York, 1991.
[74] Jongbae Kim e David Wilemon. «Focusing the fuzzy front–end in new product development». Em: *R&D Management* 32.4 (2002), pp. 269–279. doi: https://doi.org/10.1111/1467-9310.00259. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-9310.00259. url: https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-9310.00259.
[75] Emet Gürel e Merba Tat. «SWOT analysis: a theoretical review.» Em: *Journal of International Social Research* 10.51 (2017).
[76] R.W. Saaty. «The analytic hierarchy process—what it is and how it is used». Em: *Mathematical Modelling* 9.3 (1987), pp. 161–176. issn: 0270-0255. doi: https://doi.org/10.1016/0270-0255(87)90473-8. url: https://www.sciencedirect.com/science/article/pii/0270025587904738.
[77] Nina Helander e Pauliina Ulkuniemi. «Customer perceived value in the software business». Em: *The Journal of high technology management research* 23.1 (2012), pp. 26–35.
[78] Andreas Eggert e Wolfgang Ulaga. «Customer perceived value: a substitute for satisfaction in business markets?» Em: *Journal of Business & industrial marketing* (2002).
[79] Alexander Osterwalder. «The business model ontology a proposition in a design science approach». Tese de doutoramento. Université de Lausanne, Faculté des hautes études commerciales, 2004.

[80] Karin Bergquist e John Abeysekera. «Quality function deployment (QFD) — A means for developing usable products». Em: *International Journal of Industrial Ergonomics* 18.4 (1996), pp. 269–275. issn: 0169-8141. doi: https://doi.org/10.1016/0169-8141(95)00051-8. url: https://www.sciencedirect.com/science/article/pii/0169814195000518.

[81] C.P.M. Govers. «What and how about quality function deployment (QFD)». Em: *International Journal of Production Economics* 46-47 (1996). Proceedings of the 8th International Working Seminar on Production Economics, pp. 575–585. issn: 0925-5273. doi: https://doi.org/10.1016/0925-5273(95)00113-1. url: https://www.sciencedirect.com/science/article/pii/0925527395001131.

[82] *Quality Function Deployment*. Jan. de 2021. url: https://quality-one.com/qfd/.

[83] Mohamed Zairi e Mohamed A Youssef. «Quality function deployment». Em: *International Journal of Quality & Reliability Management* (1995).

[84] Martin Fowler. *Software Architecture Guide*. url: https://martinfowler.com/architecture/.

[85] *Django Documentation - Templates*. url: https://docs.djangoproject.com/en/3.2/ref/templates/language/.

[86] Robert C Martin, James Newkirk e Robert S Koss. *Agile software development: principles, patterns, and practices*. Vol. 2. Prentice Hall Upper Saddle River, NJ, 2003.

[87] Robert C Martin. *The Clean Code Blog*. Mai. de 2014. url: https://blog.cleancoder.com/uncle-bob/2014/05/08/SingleReponsibilityPrinciple.html.

[88] Jean Philippe. *From Monolith to Microservices: Lessons Learned on an Industrial Migration to a Web Oriented Architecture*. 2017. url: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7958457.

[89] Martin Fowler e James Lewis. *Microservices*. Mar. de 2014. url: https://martinfowler.com/articles/microservices.html.

[90] Martin Fowler. *Monolith First*. Jun. de 2015. url: https://martinfowler.com/bliki/MonolithFirst.html.

[91] A. Leff e J.T. Rayfield. «Web-application development using the Model/View/Controller design pattern». Em: (2001), pp. 118–127. doi: 10.1109/EDOC.2001.950428.

[92] John Deacon. «Model-view-controller (mvc) architecture». Em: *Online][Citado em: 10 de março de 2006.] http://www. jdl. co. uk/briefings/MVC. pdf* (2009).

[93] Mark Richards. *Software Architecture Patterns*. url: https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html.

[94] Alex. *DESIGN A CHAT SYSTEM*. url: https://systeminterview.com/design-a-chat-system.php.

[95] Ian Fette e Alexey Melnikov. *The websocket protocol*. 2011.

[96] Vhorne. *Suporte WebSocket em Gateway de aplicação Azure*. url: https://docs.microsoft.com/pt-pt/azure/application-gateway/application-gateway-websocket.

[97] Vineet John e Xia Liu. «A survey of distributed message broker queues». Em: *arXiv preprint arXiv:1704.00411* (2017).

[98] *What is AMQP and why is it used in RabbitMQ?* url: https://www.cloudamqp.com/blog/what-is-amqp-and-why-is-it-used-in-rabbitmq.html.

[99] Marcus Werlinder. *Comparing the scalability of MQTT and WebSocket communication protocols using Amazon Web Services*. 2020.

[100] JT Zhao, SY Jing e LZ Jiang. *Management of API gateway based on micro-service architecture*. IOP Publishing, 2018.

[101] Fabrizio Montesi e Janine Weber. «Circuit breakers, discovery, and API gateways in microservices». Em: *arXiv preprint arXiv:1609.05830* (2016).

[102] Tom Christie. *Class-based Views*. url: `https://www.django-rest-framework.org/api-guide/views/`.

[103] *UI: A popular React UI framework*. url: `https://material-ui.com/`.

[104] url: `https://everynoise.com/`.

[105] Paula Escudeiro e José Bidarra. «Quantitative evaluation framework (QEF)». Em: *Conselho Editorial/Consejo Editorial* 16 (2008).

[106] Deborah Hix e H Rex Hartson. *Developing user interfaces: ensuring usability through product & process*. John Wiley & Sons, Inc., 1993.

Apêndice A

Diagramas de Sequência Completos

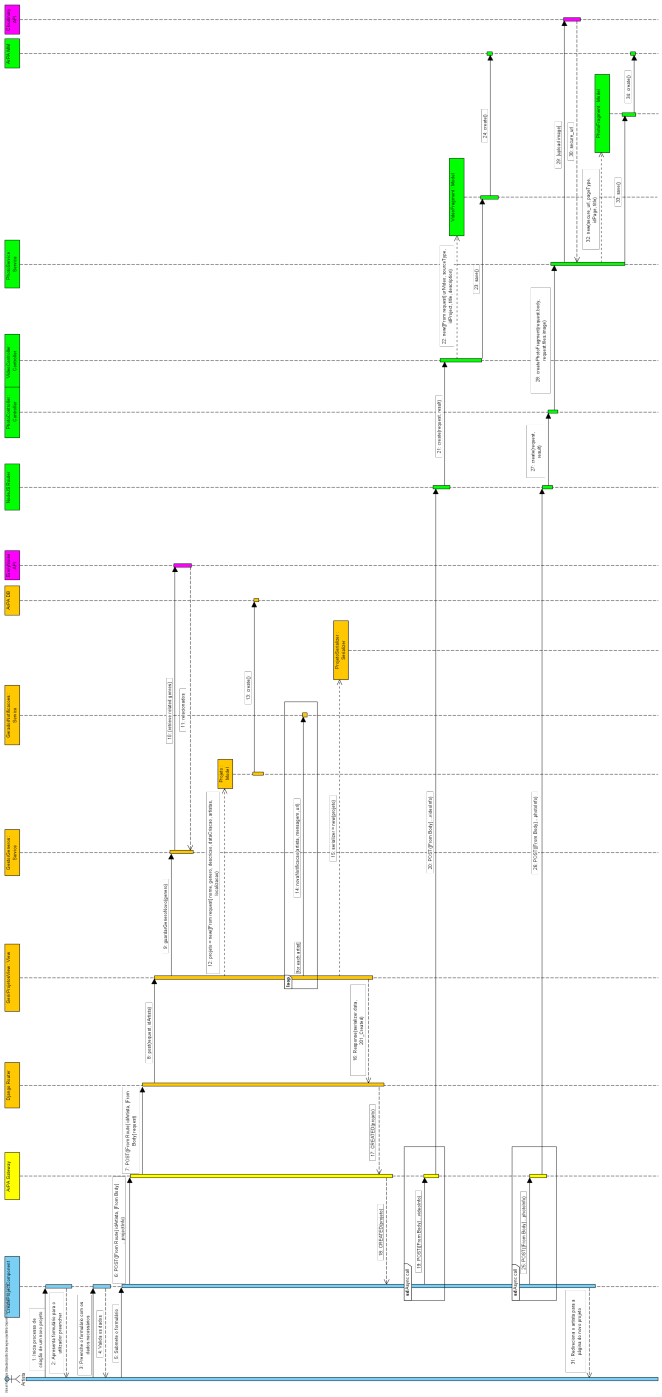


Figura A.1: Diagrama de Sequência do UC18 - Criar Projeto.

Apêndice B

Métricas QEF

B.0.1 Funcionalidade

	Wfk - Fullfilment (%)				
Requirement	**0**	**25**	**50**	**75**	**100**
FNE01 - Permite adicionar utilizador à rede de contactos	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado
FNE02 - Permite remover utilizador da rede de contactos	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado
FNE03 - Permite ler mensagens de outros utilizadores	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado
FNE04 - Permite enviar mensagens para outros utilizadores	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado
FNE05 - Permite pesquisar por outros utilizadores	Não implementado	-	Implementado, sem a presença de filtros	-	Complementamente implementado
FNE06 - Permite ver como as redes de contactos entre dois utilizadores se relacionam	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado

Figura B.1: Métricas de Avaliação do Fator *Networking*.

	Wfk - Fullfilment (%)				
Requirement	**0**	**25**	**50**	**75**	**100**
FUT01 - Permite criar uma conta de utilizador	Não implementado	-	Implementado, com a presença de bugs	-	Complementamente implementado
FUT02 - Permite editar o perfil de um utilizador	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado

Figura B.2: Métricas de Avaliação do Fator Utilizador.

	Wfk - Fullfilment (%)				
Requirement	**0**	**25**	**50**	**75**	**100**
FFS01 - Permite ler notificações do sistema	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado
FFS02 - O sistema notifica os utilizadores quando uma ação que lhes é relevante acontece	Não implementado	-	Parcialmente implementado	-	Complementamente implementado
FFS03 - O sistema redireciona o utilizador dentro da página de forma a demonstrar o resultado do evento de relevância que ocorreu	Não implementado	-	Parcialmente implementado	-	Complementamente implementado
FFS04 - Permite a notificação via email de eventos relevantes para os utilizadores	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado

Figura B.3: Métricas de Avaliação do Fator *Feedback* do Sistema.

	Wfk - Fullfilment (%)				
Requirement	**0**	**25**	**50**	**75**	**100**
FAR01 - Permite criar um projeto	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado
FAR02 - Permite editar um projeto	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado
FAR03 - Permite remover um projeto	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado
FAR04 - Permite gerar um rider técnico	Não implementado	-	Apenas permite incluir mapa de palco ou material	-	Complementamente implementado
FAR05 - Permite candidatar a um evento	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado
FAR06 - Permite criar uma proposta	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado
FAR07 - Permite editar uma proposta	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado
FAR08 - Permite remover uma proposta	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado

Figura B.4: Métricas de Avaliação do Fator Artista.

	Wfk - Fullfilment (%)				
Requirement	**0**	**25**	**50**	**75**	**100**
FAG01 - Permite editar perfil de um cliente	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado
FAG02 - Permite adicionar um artista à carteira de artistas	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado
FAG03 - Permite remover um artista da carteira de artistas	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado
FAG04 - Permite gerar uma proposta a partir da sua carteira de artistas	Não implementado	-	Permite apenas usar um artista	-	Permite usar qualquer combinação de artistas
FAG05 - Permite apresentar a candidatura de um artista da sua carteira de artistas	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado

Figura B.5: Métricas de Avaliação do Fator Agente.

	Wfk - Fullfilment (%)				
Requirement	**0**	**25**	**50**	**75**	**100**
FPR01 - Permite criar um evento	Não implementado	-	Permite criar eventos de raiz	-	Permite criar eventos a partir de propostas vindas de agentes ou artistas
FPR02 - Permite editar um evento	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado
FPR03 - Permite remover um evento	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado
FPR04 - Permite fechar um evento	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado
FPR05 - Permite selecionar candidatos para um evento	Não implementado	Não funcional	Parcialmente implementado	Implementado, com a presença de bugs	Complementamente implementado

Figura B.6: Métricas de Avaliação do Fator Promotor.

	Wfk - Fullfilment (%)				
Requirement	**0**	**25**	**50**	**75**	**100**
FAU01 - Permite a inclusão de ficheiros de som no perfil de um utilizador/projeto	Não implementado	-	Permite menos de 3 formatos de ficheiro	-	Permite 3 ou mais formatos de ficheiro
FAU01 - Permite a inclusão de ficheiros de vídeo no perfil de um utilizador/projeto	Não implementado	-	Permite menos de 3 formatos de ficheiro	-	Permite 3 ou mais formatos de ficheiro
FAU03 - Permite a inclusão de conteúdo audiovisual proveniente de outras plataformas	Não implementado	-	Permite menos de 3 fontes diferentes	-	Permite 3 ou mais fontes

Figura B.7: Métricas de Avaliação do Fator Recomendação.

	Wfk - Fullfilment (%)				
Requirement	**0**	**25**	**50**	**75**	**100**
FRE01 - Recomenda perfis semelhantes àquele que é visitado	Não implementado	-	Recomenda perfis com base num algoritmo que recorre a variáveis fixas	-	Recomenda perfis com base num algoritmo ajustável pelo utilizador
FRE02 - Recomenda perfis relevantes para apoio à decisão do Artista	Não implementado	-	Recomenda perfis com base num algoritmo que recorre a variáveis fixas	-	Recomenda perfis com base num algoritmo ajustável pelo utilizador
FRE03 - Recomenda perfis relevantes para apoio à decisão do Promotor	Não implementado	-	Recomenda perfis com base num algoritmo que recorre a variáveis fixas	-	Recomenda perfis com base num algoritmo ajustável pelo utilizador
FRE04 - Recomenda perfis relevantes para apoio à decisão do Agente	Não implementado	-	Recomenda perfis com base num algoritmo que recorre a variáveis fixas	-	Recomenda perfis com base num algoritmo ajustável pelo utilizador

Figura B.8: Métricas de Avaliação do Fator Audiovisual.

B.0.2 Suportabilidade

	Wfk - Fullfilment (%)				
Requirement	**0**	**25**	**50**	**75**	**100**
SA01 - A plataforma está disponível em diferentes browsers	Apenas acessível num browser	-	Acessível em 3 ou menos browsers	-	Acessível em mais de 3 browsers
SA02 - A plataforma deverá supor vários formatos de ficheiro audiovisual	Não suporta conteúdo audiovisual	-	Apenas suporta vídeo, imagem ou áudio	-	Suporta qualquer tipo de conteúdo audiovisual

Figura B.9: Métricas de Avaliação do Fator Adaptabilidade.

	Wfk - Fullfilment (%)				
Requirement	**0**	**25**	**50**	**75**	**100**
SM01 - A implementação deverá estar acompanhada de documentação completa e detalhada	Nenhuma documentação disponibilizada	-	Documentação incompleta	-	Documentação completa e detalhada
SM02 - A implementação da plataforma deverá permitir a inclusão de novas funcionalidades de forma rápida e fácil	Requerer conhecimento profundo da base de código e demora mais de dois dias	-	Requer conhecimento profundo da base de código e demora menos de dois dias	-	Não requer conhecimento profundo da base de código e demora menos de um dia

Figura B.10: Métricas de Avaliação do Fator Manutenibilidade.

B.0.3 Usabilidade

	Wfk - Fullfilment (%)				
Requirement	**0**	**25**	**50**	**75**	**100**
UL01 - Suporta as línguas portuguesa e inglesa como linguagem de utilização da plataforma	-	-	Apenas suporta português	-	Suporta português e inglês
UL02 - Utiliza terminologia existente e relevante no contexto da área de negócio	Terminologia não compatível	-	-	-	Terminologia compatível

Figura B.11: Métricas de Avaliação do Fator Linguagem.

	Wfk - Fullfilment (%)				
Requirement	**0**	**25**	**50**	**75**	**100**
UN01 - A navegação entre diferentes menus deverá ser simples e ter um número reduzido de passos	Navegação confusa	-	Navegação fácil	-	Navegação fácil e rápida e consistente com a lógica de negócio
UN02 - Apresentação das páginas deverá ser feita rapidamente	A plataforma não é responsiva e o tempo de carregamento é elevado	-	A plataforma tem um tempo de carregamento acima do normal	-	A plataforma é responsiva e tem um tempo curto de carregamento
UN03 - Disposição clara das diferentes opções do utilizador	Interface confusa e sem lógica aparente	-	Interface com algumas lacunas na facilidade de utilização	-	Interface intuitiva
UN04 - As ações do utilizador resultam num feedback visual	Não há feedback visual	-	Há feedback visual para algumas ações	-	Há feedback visual para todas as ações

Figura B.12: Métricas de Avaliação do Fator Navegação.

	Wfk - Fullfilment (%)				
Requirement	**0**	**25**	**50**	**75**	**100**
UC01 - Plataforma utiliza uma interface visualmente consistente ao longo das diferentes páginas	Não há consistência	-	Plataforma com algumas inconsistências	-	Plataforma totalmente consistente
UC02 - Os perfis dos utilizadores devem ter uma estrutura modular mas editável	Perfil não modular	-	Perfil modular, com uma ordem específica	-	Perfil modular, o utilizador pode escolher quais módulos apresentar e ordenar conforme desejar

Figura B.13: Métricas de Avaliação do Fator Consistência.