Mestrado em Informática e Sistemas

AudioSpot - Software para detecção de *audio watermarks*

Relatório apresentado para a obtenção do grau de Mestre em Informática e Sistemas

Especialização em Desenvolvimento de Software

Autor
Filipe Rafael Lopes Brandão

Orientador
Fernanda Maria dos Reis Brito e Rodrigues Correia
Professor do Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra

Supervisor
Nuno Emanuel Figueiredo Carvalho
WIT Software

Coimbra, Outubro, 2016

Agradecimentos

Expresso aqui os meus agradecimentos aos familiares mais próximos que de uma forma ou outra intervieram para a realização deste projecto.

Gostaria de agradecer ao Nuno Carvalho pela oportunidade de estagiar na WIT Software. A este e à Professora Fernanda Correia, agradeço a disponibilidade e apoio prestado na orientação de todo este trabalho.

Agradeço especialmente a todos os docentes que me transmitiram conhecimento ao longo do curso e a todos os amigos e colegas que me acompanharam e contribuíram para todo o sucesso que alcancei no decorrer do curso. Expresso ainda os meus agradecimentos aos elementos da "Team TV" que me apoiaram no decorrer do estágio e me proporcionaram o melhor ambiente de trabalho.

Para finalizar agradeço a toda a instituição ISEC pelas condições e qualidade de ensino que me ofereceram.

Coimbra, 6 de Outubro de 2016

Filipe Brandão

Resumo

Nos dias de hoje, vivemos num mundo poluído pela publicidade. De tal forma que não será exagerado dizer que muitas das pessoas filtram naturalmente a publicidade diminuindo a eficácia da mensagem que esta pretende transmitir. É do interesse das agências de publicidade e das marcas a si associadas garantir que a publicidade chega ao público-alvo com sucesso. Surge daí a importância em abordar tecnologias que tornem a publicidade mais cativante. Uma das maneiras de o fazer é tornando-a mais interactiva, por exemplo, através da tecnologia denominada "*audio watermarking"* que permite a injecção de sinais digitais adicionais ao áudio tornando possível a comunicação com dispositivos tais como *smatphones* ou *tablets*.

Visto que a empresa WIT Software possui um departamento dedicado ao desenvolvimento soluções de *software* para a área da televisão e que esta empresa pretende apostar em conceitos e tecnologias diferenciadoras, capazes destacar os seus produtos dos demais, surge o interesse em abordar a referida tecnologia que é pouco explorada não sendo comum encontrar uma aplicação que recorra essa mesma tecnologia.

O estágio aqui descrito teve como objectivos a análise da tecnologia denominada *audio watermarking*, o desenvolvimento de protótipos de aplicações móveis recorrendo a esta tecnologia e ainda o desenvolvimento de um módulo reutilizável para detectar *audio watermarks* facilmente em novas aplicações. O estágio foi dividido em várias tarefas que permitiram abranger o tema na sua generalidade bem como a especificação de um *backoffice* que suporta a utilização de *audio watermarks* numa aplicação móvel.

Durante o estágio foram analisadas as principais dificuldades, soluções e casos de uso relativas ao *audio watermarking* sendo ainda testadas algumas dessas soluções. A partir daí foram desenvolvidos vários protótipos de aplicações para o sistema Android para avaliação da tecnologia e demonstração a potenciais clientes. Paralelamente, iniciou-se o desenho de *mockups* para a componente de *frontend* de um *backoffice* e ainda a especificação funcional da componente de *backend* que foi, de seguida, implementada e alvo de diversos testes. No final do estágio desenvolveu-se um módulo reutilizável para permitir que os *audio watermarks* sejam facilmente integrados em novas aplicações móveis para o sistema Android. Os protótipos para o sistema Android, o *backend* e o referido módulo foram desenvolvidos recorrendo a uma das soluções analisadas, o SDK da Cifrasoft.

Os protótipos desenvolvidos permitiram testar conceitos que recorrem à tecnologia analisada e o restante *software* produzido, tanto o *backend* e como módulo reutilizável para aplicações móveis, será introduzido em novos produtos desenvolvidos pela empresa.

Abstract

Nowadays, we live in a world polluted by advertising. In such a way that it will not be exaggerated to say that many people naturally filter advertising thus reducing the effectiveness of the message it intends to convey. It is the best interest of the advertising agencies and associated brands to ensure that advertising reaches the target audience successfully. This gives rise to interest in addressing technologies that make advertising more captivating. One of the ways to do this is to make it more interactive, for example through technology called "audio watermarking" that allows the injection of additional digital signals to the audio making it possible to communicate with devices such as smatphones or tablets.

Being WIT Software a company that has a department dedicated to the development of software solutions for the television area and that it intends to invest on differentiating concepts and technologies, capable of highlighting its products from others, emerges the interest in approaching the said technology that is considered to be still little explored.

The internship described here aimed to analyze the audio watermarking technology, to develop mobile applications prototypes using this technology and to develop a reusable module to detect audio watermarks in new applications. The work was split into several tasks that covered the main topic and also the specification of a backoffice that supports the use of audio watermarks in mobile application.

During the internship the main difficulties, solutions and use cases for audio watermarking were analyzed and the solutions found were tested. From there, several prototypes of applications for the Android system were developed, in order to assess the technology and demonstrate it to potential customers. Simultaneously, user interface mockups for the frontend component of a backoffice were designed and the functional specification of the backend component was written. This backend was then implemented and targeted by several tests. Lastly, a reusable module was developed in order to allow easy integration of the audio watermarking technology into new mobile applications for the Android system. The mentioned prototypes, the backend and the reusable module were developed using one of the analyzed solutions: the Cifrasoft’s SDK.

It is considered that the achieved results will be relevant to the future activity of WIT Software because the audio watermarking technology will be introduced in new products developed by the WIT Software.

Índice

1. Introdução 1
1.1. Âmbito 2
1.2. Objectivos 2
1.3. Estrutura do Relatório 2
2. Contextualização do Estágio 5
2.1. Instituto Superior de Engenharia de Coimbra (ISEC) 5
2.2. Empresa WIT Software 5
3. Plano de trabalho 7
4. WP 100: Levantamento do Estado da Arte 9
4.1. Análise Sobre Audio Watermarks 9
4.1.1. Motivação 9
4.1.2. Diversas Aproximações 9
4.1.3. Soluções Existentes 12
4.1.4. Apreciação Geral das Frameworks 16
4.1.5. Watermarking e Fingerprinting 17
4.1.6. Casos de Uso e Implementações Existentes 17
4.1.7. Conclusão 20
4.2. Testes Realizados às Soluções Existentes 20
4.2.1. Ambiente de Testes 21
4.2.2. Resultados 23
4.2.3. Teste Adicional de Sobrevivência ao Transcoding 24
4.2.4. Conclusão 24
5. WP 200: Prototipagem rápida 27
5.1. Descrição Funcional 27
5.2. Arquitectura do Módulo de Detecção de Watermarks 27
5.3. Provas de Conceito Focadas na Interface 30
5.4. Ferramentas Usadas 34
5.5. Metodologia de Desenvolvimento 35

5.6. Conclusão 35
6. WP 300: Especificação do Software a Implementar 36
6.1. Especificação 36
6.1.1. Descrição do Sistema 36
6.1.2. Canais de Publicação 36
6.1.3. Definição de Stakeholders 39
6.1.4. Definição de User Roles e Stories 39
6.1.5. Desenho de Mockups 41
6.2. Arquitectura 42
6.2.1. Backoffice Frontend 42
6.2.2. Backoffice Backend 43
6.2.3. File Storage & Processing 43
6.3. Aprendizagem 43
6.4. Redefinição dos Objectivos 45
6.5. Plano de Desenvolvimento 48
6.6. Definição do Modelo de Dados 51
6.7. Conclusão 53
7. WP 400: Implementação do Software 55
7.1. Aplicação Servidor 55
7.1.1. Modelo de Dados 55
7.1.2. Web Services 56
7.1.3. Autenticação 57
7.1.4. Conclusão 58
7.2. Desenvolvimento de uma Biblioteca 59
7.2.1. Descrição da Biblioteca 59
7.2.2. Detalhes de Implementação e Reutilização 60
7.2.3. Conclusão 62
8. WP 500: Testes e Correcções ao Software 63
8.1. Testes Automatizados 63
8.1.1. Critério 63

8.1.2. Tecnologia 63
8.1.3. Conclusão 63
8.2. Testes de Carga 65
8.2.1. Caso de Teste 65
8.2.2. Tecnologia 66
8.2.3. Conclusão 66
9. Conclusões e Trabalho Futuro 69
Anexo A - Resultados dos Testes Realizados às Várias Soluções de Watermarking 73
Anexo B - Fluxo de Criação de Campanhas no Backoffice 77
Anexo C - Fluxo de Detecção de Campanhas Provisionadas no Backoffice 79
Anexo D - Tabela de Casos de Uso Mapeados em Canais de Publicação 81
Anexo E - Mockups Desenhados (versão final) 81
Anexo F – Calendarização das tarefas da primeira release do backend 92
Anexo G - Diagrama de Classes – Classes do Modelo de Dados do Backend 94
Anexo H – Resultados dos Testes de Integração Automatizados 96
Anexo I – Resultados dos Testes de Carga 98
Anexo J - Relatórios de Progresso 100
Anexo K - Proposta de Estágio 108
Bibliografia 112

Índice de Figuras

Figura 1- Evolução da quantidade de aplicações disponíveis na App Store da Apple 1

Figura 2 - Distribuição temporal das tarefas realizadas no estágio 8

Figura 3- Transmissão de informação digital através de áudio 9

Figura 4 - Espectrograma captado por um *smartphone* enquanto eram emitidos sinais sonoros em frequências inaudíveis 10

Figura 5 - Captura de ecrã da ferramenta de injecção de *watermarks* da Cifrasoft 14

Figura 6- Capturas de ecrã da aplicação desenvolvida para testes 21

Figura 7- Registo fotográfico do ambiente de testes 22

Figura 8 - Diagrama de classes do módulo de detecção de *watermarks* 29

Figura 9- Template gráfico utilizado nos ecrãs da aplicação de demonstração *DemoApp5* 32

Figura 10 - Capturas de ecrã da aplicação de demonstração *DemoApp5* para os casos de uso 4, 7, 9 e 10 33

Figura 11 - Iterações de desenvolvimento das aplicações de demonstração 35

Figura 12 - Árvore de hierarquia de ecrãs do *backoffice* 41

Figura 13 - Arquitectura inicial do backoffice 42

Figura 14 - Diagrama de arquitectura do *backend* revisto 47

Figura 15 - Diagrama físico do modelo de dados do *backend* 52

Figura 16 - Exemplo de código que define um *endpoint* de uma *API REST* através da *framework Spring Web MVC* 56

Figura 17- Exemplo de suite gerada pelo *Swagger* para um *web service* 57

Figura 18- Arquitectura simplificada da biblioteca Android desenvolvida 59

Figura 19 - Diagrama de sequência da comunicação entre os componentes da biblioteca 60

Figura 20- Diagrama detalhado que fluxo de comunicação entre os vários componentes da biblioteca 61

Figura 21- Sumário dos resultados de *test coverage* para as classes que definem os *web services* 65

Figura 22- Evolução do throughput real e do tempo de resposta médio 67

Figura 23 - Evolução da utilização do *CPU* 67

Figura 24- Evolução do mínimo de memória *RAM* livre....68
Figura 25 - Fluxo de criação de campanhas no backoffice....77
Figura 26 - Fluxo de detecção de campanhas provisionadas no backoffice....79
Figura 27 - Mockup - Ecrã de login....81
Figura 28 - Mockup - Ecrã de campanhas activas....81
Figura 29 - Mockup - Ecrã de campanhas activas (descrição da interacção)....82
Figura 30 - Mockup - Ecrã de criação de campanha....82
Figura 31 - Mockup - Ecrã de criação de campanha (2)....83
Figura 32 - Mockup - Ecrã de criação da campanha (detalhe de preenchimento de form)83
Figura 33 - Mockup - Ecrã de criação da campanha (associação de canais de publicação)....84
Figura 34 - Mockup - Ecrã de criação da campanha (restrição de localização)....84
Figura 35 - Mockup - Ecrã de criação da campanha (video element)....85
Figura 36 - Mockup - Ecrã de criação da campanha (parâmetros do video element)....85
Figura 37 - Mockup - Ecrã de criação da campanha (app element)....86
Figura 38 - Mockup - Ecrã de criação da campanha (parâmetros do app element)....86
Figura 39 - Mockup - Ecrã de criação da campanha (link element)....87
Figura 40 - Mockup - Ecrã de criação da campanha (parâmetros do link element)....87
Figura 41 - Mockup - Ecrã de criação da campanha (image element)....88
Figura 42 - Mockup - Ecrã de criação da campanha (parâmetros do image element)....88
Figura 43 – Mockup - Ecrã de reporting (lista de customers)....89
Figura 44 – Mockup - Ecrã de reporting (campanha)....89
Figura 45 - Mockup - Ecrã de customers....90
Figura 46 - Mockup - Ecrã de criação de customer....90
Figura 47 - Mockup- Ecrã de associação de um user a um customer....91
Figura 48 - Mockup - Parâmetros de diversos elementos das campanhas....91
Figura 49- Diagrama de *gantt* do plano para a primeira *release* do *backend*....92
Figura 50 – Diagrama de classes do modelo de dados do backend....94
Figura 51- Diagrama de classes mediadoras da interacção entre camada de negócio e modelo de dados....95
Figura 52 - Lista dos testes de integração automatizados e o seu tempo de execução....96

Figura 53 - Relatório de code coverage dos testes de integração 97

Índice de Tabelas

Tabela 1 - Apreciação geral das vantagens/desvantagens das diferentes aproximações ... 11

Tabela 2 - Comparação resumida das várias soluções analisadas 16

Tabela 3 - Medição de intensidade do ruído ambiente através das aplicações “SoundMeter PRO” e “Sound Meter for Android” 23

Tabela 4- Exemplo de código *JSON* e resultado produzido na aplicação de demonstração *DemoApp4* 31

Tabela 5 - Casos de uso explorados na aplicação de demonstração *DemoApp5* 32

Tabela 6 - Casos de uso mapeados em canais de publicação 38

Tabela 7 - User Roles identificadas para o backoffice 40

Tabela 8 - Funcionalidades do backoffice mapeadas nas User Roles 40

Tabela 9 - Comparação entre MEAN.js e Spring Boot 45

Tabela 10 - Tarefas desenvolvidas na *release* 1 49

Tabela 11 - Tarefas desenvolvidas na *release* 2 50

Tabela 12 - Listagem de testes implementados e o seu tempo de execução 64

Tabela 13 - Características da máquina de testes 66

Tabela 14 - Resultados dos testes às soluções de watermarking 73

Tabela 15 - Resultados dos testes às soluções de watermarking - Taxa de detecção 74

Tabela 16 - Resultados dos testes às soluções de watermarking - Taxa de detecção independente do smartphone 75

Tabela 17- Resultados dos testes de carga 98

Abreviaturas

API – Application Programming Interface

CLI – Command Line Interface

DEIS – Departamento de Engenharia Informática e de Sistemas

GUI – Graphical User Interface

HLS – HTTP Live Streaming

IDE - Integrated Development Environment

ISEC – Instituto Superior de Engenharia de Coimbra

JNI – Java Native Interface

JSON – JavaScript Object Notation

JWT – JSON Web Token

MWC – Mobile World Conference

REST - Representational State Transfer

SDK – Software Development Kit

WP – Work Package

WYSIWYG – What You See Is What You Get

1. Introdução

Com a disseminação dos *smartphones* e *tablets* para o público comum, vemos actualmente os vários mercados de aplicações completamente saturados com milhões de produtos. Os mais recentes dados estatísticos apontam para um crescimento, no último ano, que ultrapassa os 30% na quantidade de aplicações móveis disponíveis no mercado de aplicações da Apple (conhecido por ter requisitos de publicação mais restritos) chegando a um total de dois milhões de aplicações conforme ilustra a Figura 1 [1].

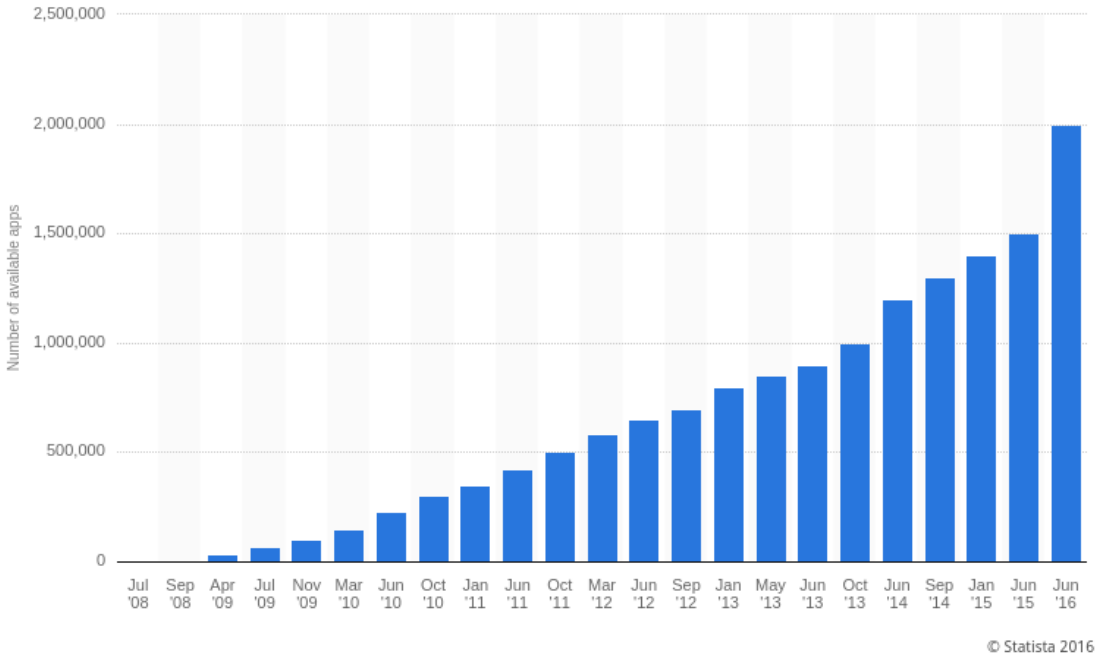


Figura 1- Evolução da quantidade de aplicações disponíveis na App Store da Apple

Na Figura 1, o eixo dos x representa a evolução temporal desde Julho de 2008 a Junho de 2016 e o eixo dos y representa a quantidade de aplicações disponíveis na AppStore num dado mês.

Os desenvolvimentos descritos neste relatório são maioritariamente direccionados para o sistema Android. Apresentam-se os dados relativos à plataforma de distribuição de aplicação do sistema concorrente apenas para demonstrar a tendência de crescimento do mercado de aplicações móveis que se encontra, há muito, saturado.

Surge assim a necessidade de apostar em conceitos e tecnologias diferenciadoras, capazes destacar um produto dos demais. Daí surgiu a curiosidade da empresa WIT Software em abordar a tecnologia denominada "*audio watermarking*". Não é comum encontrar uma aplicação que recorra a tal tecnologia e, teoricamente, será aplicável em aplicações móveis.

Pretendeu-se então focar este estágio na análise da referida tecnologia e testar a viabilidade da sua utilização em aplicações móveis através de diferentes protótipos que são utilizados para confirmar diversos casos de uso, culminando com o desenvolvimento de um módulo que permita reutilizar a tecnologia noutras aplicações. Adicionalmente pretendeu-se também desenvolver um *backoffice* capaz de dar suporte à utilização desta tecnologia num caso de uso relacionado com a publicidade.

1.1. Âmbito

Este relatório tem como objectivo descrever o trabalho desenvolvido na empresa WIT Software, no âmbito da unidade curricular de Estágio/Projecto do 2º ano do Mestrado em Informática e Sistemas (especialização em Desenvolvimento de Software) do Departamento de Engenharia Informática e de Sistemas do Instituto Superior de Engenharia de Coimbra (DEIS-ISEC).

1.2. Objectivos

O estágio tem como objectivo a análise da tecnologia de *audio watermarking* e o desenvolvimento de *software* que tire partido desta tecnologia. O *software* a desenvolver inclui vários protótipos para a plataforma Android e ainda um *backoffice* que integrarão a tecnologia analisada.

Pretende-se então obter um documento de levantamento e análise das soluções de *audio watermaking*, um documento com descrição de diversos casos de uso possíveis para esta tecnologia para aplicações móveis, o referido *software* e ainda um módulo reutilizável para a plataforma Android.

A proposta de estágio aprovada pela Comissão de Coordenação de Projectos e Estágios do DEIS pode ser consultada no “Anexo J - Proposta de Estágio”.

1.3. Estrutura do Relatório

Este relatório está organizado em capítulos que descrevem as diferentes tarefas realizadas, como foram realizadas, decisões tomadas e conclusões retiradas.

Após esta apresentação genérica do estágio realizado, no segundo capítulo, será apresentada a proposta de estágio bem como as entidades envolvidas no estágio: a WIT Software e o Instituto Superior de Engenharia de Coimbra.

No capítulo 3, é apresentado o plano de trabalho do estágio.

No capítulo 4, será descrito o processo de análise bem como os problemas principais e conclusões retiradas quanto às soluções e aproximações de *audio watermarking* analisadas bem como uma explicação dos testes adicionais que foram realizados sobre as diversas soluções analisadas.

O quinto capítulo descreve alguns detalhes acerca dos protótipos desenvolvidos para fins de demonstração.

De seguida, no sexto capítulo, é apresentada a especificação e arquitectura do *backoffice*, principalmente da componente de *backoffice* que foi realmente implementada.

O capítulo 7 é dedicado à implementação do referido *backend* bem como de uma biblioteca desenvolvida para Android e que permite reutilizar alguns conceitos aplicados nos protótipos desenvolvidos.

No capítulo 8, descrevem-se algumas preocupações tidas em conta para testar o *software*.

No último capítulo, serão apresentadas as principais conclusões de todo este trabalho.

2. Contextualização do Estágio

Neste capítulo são identificadas as entidades envolvidas na realização do estágio, bem como a proposta de estágio aprovada pela Comissão de Coordenação de Projectos e Estágios do DEIS.

2.1. Instituto Superior de Engenharia de Coimbra (ISEC)

O ISEC é uma unidade orgânica do Instituto Politécnico de Coimbra (IPC), fundada em 1974 com a conversão do Instituto Industrial e Comercial de Coimbra.

O Departamento de Engenharia Informática e de Sistemas (DEIS) dedica-se, desde 1989, à formação, investigação, desenvolvimento e prestação de serviços na área da Engenharia Informática. Prima pelo carácter prático dos cursos que ministra, traduzido pelo elevado número de aulas laboratoriais, trabalhos práticos, projectos e estágios o que, aliado à qualidade científica e pedagógica, resulta num elevado índice de empregabilidade [2].

O Mestrado em Informática e Sistemas (MIS) do Instituto Superior de Engenharia de Coimbra (ISEC) tem por objectivo formar Mestres em Informática e em Sistemas capazes de exercerem a sua actividade profissional com um elevado nível de competência técnica, científica e profissional em cada uma das áreas de especialização propostas.

A formação dada na especialização em Desenvolvimento de Software visa a preparação de profissionais competentes no exercício profissional no domínio de desenvolvimento de *software* em todas as suas etapas, incluindo os aspectos importantes de gestão de projectos, equipas e garantia de qualidade. Os graduados do mestrado nesta especialização serão engenheiros técnicos com pleno domínio científico dos temas relacionados com engenharia de *software*, possuindo capacidade de resposta a exigências constantes de inovação e modernização [3].

2.2. Empresa WIT Software

Sediada em Coimbra, neste momento a WIT Software possui escritórios em diversas partes do mundo tais como: Portugal - Porto, Leiria e Lisboa, Reino Unido – Reading, Alemanha – Dusseldorf e EUA – California. Actualmente detém mais de 200 empregados que trabalham com o objectivo de fornecer o melhor serviço aos seus clientes. Assim, ao longo dos últimos anos, a empresa ganhou reconhecimento e lançou um conjunto de produtos, de grande qualidade que representam a tecnologia de ponta, para o mercado [4].

A WIT Software é uma empresa especializada na criação de aplicações e serviços inovadores para os operadores de telecomunicações, operadores de televisão e internet móvel. Fundada em 2001 como *spin-off* da Universidade de Coimbra, estabeleceu desde logo uma parceria com a Vodafone Portugal, cliente com o qual continua a trabalhar nos

dias de hoje. Ao longo dos anos a empresa tem registado um rápido e sustentado crescimento no sector que lhe permitiu marcar uma posição no mercado mundial, contando com um portfólio de clientes onde se destacam nomes como: Grupo Vodafone, Telefónica, TeliaSonera, Deutsche Telekom, Orange, Unitel, CenturyLink.

O sucesso da empresa constata-se não só pelo portfólio de clientes que possui mas igualmente pelo reconhecimento que lhe é concebido sendo considerada desde 2009 uma empresa PME Líder distinguida sempre com excelência, resultado dos seus resultados financeiros assim como da estabilidade financeira alcançada. A empresa encontra-se igualmente certificada nas normas de Qualidade (ISO 9001), Ambiente (NP EN ISO 14001) e Inovação (NP EN 4457).

3. Plano de trabalho

O estágio foi dividido nas seguintes *work-packages* (WP):

- WP 100: Levantamento do estado da arte – Análise da tecnologia de *audio watermarking* e realização de testes a diferentes soluções.
- WP 200: Prototipagem rápida – Implementação de pequenas aplicações para fins de demonstração da tecnologia e de diversos casos de uso. As aplicações foram maioritariamente desenvolvidas para o sistema Android e houve ainda um pequeno protótipo inicial desenvolvido também para iOS.
- WP 300: Especificação do *software* a implementar – Documentação e desenho da arquitectura de um *web backoffice*.
- WP 400: Implementação da aplicação servidor e de uma biblioteca Android – Implementação da componente de *backend* do *backoffice* e módulo reutilizável para o sistema Android.
- WP 500: Testes e correções ao *software* – Descrição dos testes realizados ao software implementado.

A Figura 2 ilustra a distribuição temporal das *work-packages* . À esquerda, coluna *Task Name* indica o nome de cada tarefa/*work-package*. Da segunda à quarta coluna é apresentada a data de início (Start *Date*), data de término (*End Date) e a duração (Duration)* da tarefa em horas. À direita é apresentado um calendário, que inicia em Novembro de 2015 e prolonga-se até ao final de Junho de 2016, no qual é possível ver desenhadas as barras que correspondem à distribuição temporal das tarefas.

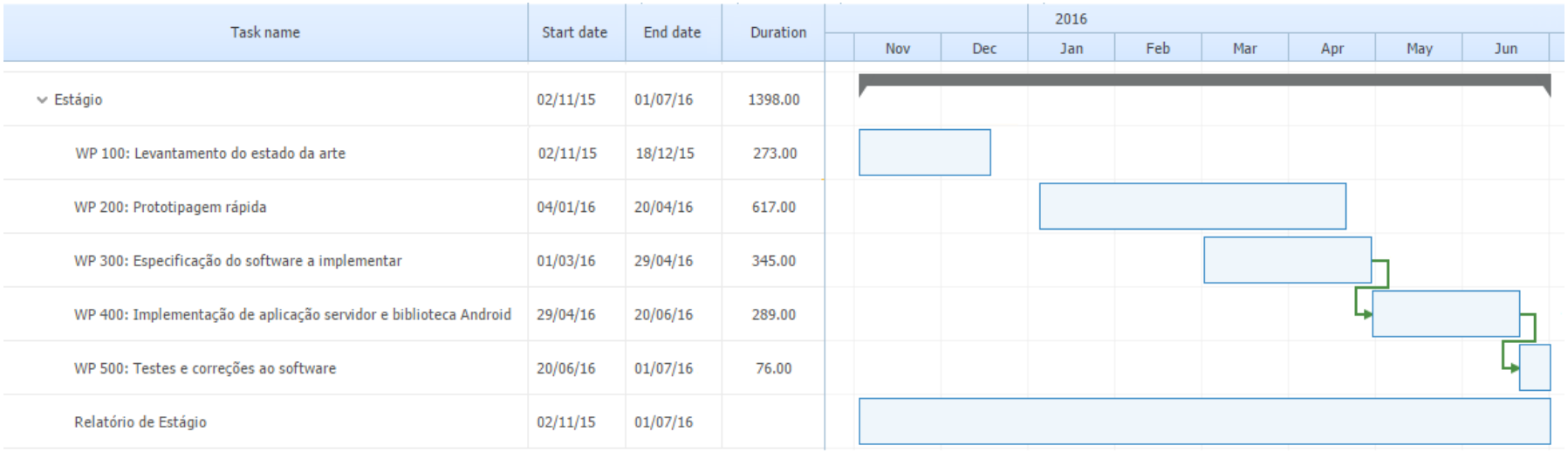


Figura 2 - Distribuição temporal das tarefas realizadas no estágio

4. WP 100: Levantamento do Estado da Arte

Neste capítulo é efectuada uma análise sobre a técnica de *watermarking* de áudio bem como das *frameworks* existentes e possíveis casos de uso para a sua aplicação. São ainda apresentados os testes realizados às várias *frameworks*.

4.1. Análise Sobre Audio Watermarks

O *audio watermarking* consiste em injectar informação digital adicional ao sinal áudio de um conteúdo que se apresente sobre esta forma de media [5]. É então possível transmitir essa informação, através de altifalantes e microfones que assumem o papel de emissores e receptores, e utilizá-la em diversos contextos que serão também abordados neste capítulo.

A Figura 3 ilustra o conceito de *audio watermaking* onde um dispositivo com um microfone (receptor) capta o som transmitido por um altifalante (emissor), podendo este som conter informação adicional que é obtida através do processamento do sinal.

Figura 3- Transmissão de informação digital através de áudio

4.1.1. Motivação

Esta tecnologia já havia sido identificada pela WIT Software com um potencial interesse futuro. No entanto, nenhum estudo tinha sido realizado até ao momento. Apenas era conhecida a oferta de uma outra empresa que acabou por fazer parte desta análise (*SoundPays*), conhecimento obtido através de uma feira de tecnologia. Foi com base no interesse existente da parte da WIT Software que surgiu a oportunidade de debruçar o estágio sobre a análise e utilização desta técnica, de forma a obter uma avaliação comprovada da tecnologia e também uma base de trabalho para projectos futuros que venham a utilizá-la.

4.1.2. Diversas Aproximações

Nesta secção são descritas as diferentes aproximações nas quais podem ser categorizadas as soluções que as diversas *frameworks* analisadas apresentam para a aplicação da técnica de *audio watermarking*.

Abordagem Inaudível

A aproximação inaudível é baseada na emissão das *watermarks* através de tons sonoros cuja frequência supera o limite do que Humano adulto consegue ouvir (tipicamente acima dos 18KhZ [6]). O objectivo é exactamente que os tons não sejam audíveis pelo Humano mas que o *hardware* consiga emitir e detectar sons nestas frequências. Teoricamente, existe pouco ruído ambiente nestas frequências o que se traduzirá numa melhor performance na detecção das *watermarks*. Apesar disso, é necessário ter em conta que nem todos os módulos de *hardware* para microfones (receptores) e altifalantes (emissores) são pensados para funcionar com estas frequências. Suspeita-se até que podem mesmo ser filtradas por motivos de optimização. Este facto coloca alguma dúvida na performance esperada para soluções que recorrem a esta aproximação inaudível.

A Figura 4 demonstra o sinal captado por um microfone de um *smartphone* no momento em que estavam a ser reproduzidos tons sonoros com frequências inaudíveis,

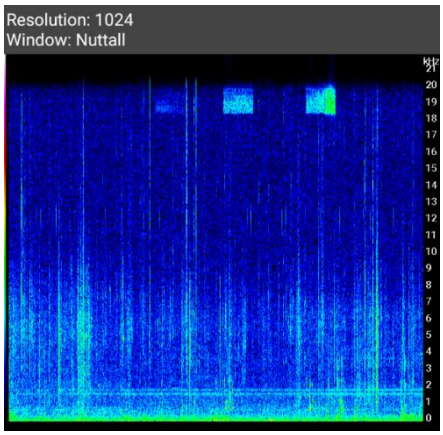


Figura 4 - Espectrograma captado por um *smartphone* enquanto eram emitidos sinais sonoros em frequências inaudíveis

Abordagem Audível mas Imperceptível

Ao contrário da aproximação inaudível, a audível e imperceptível utiliza frequências audíveis pelo ouvido humano para emitir os *watermarks*. No entanto, esta aproximação recorre a técnicas nas quais os *watermarks* são incluídos de forma imperceptível pelos humanos. Ou seja, ao ouvir o áudio original e o áudio com *watermarks*, um humano não será capaz de se aperceber da presença das *watermarks*. Dado que são usadas frequências comuns, a expectativa é que o vasto *hardware* seja capaz de suportar esta aproximação considerando que o ruído ambiente pode prejudicar a capacidade de detecção de *watermarks*.

Principais Valências

Se em alguns cenários esta tecnologia, de *audio watermarking*, não adiciona nada face a outras formas de comunicação tais como Bluetooth ou mesmo uma ligação à Internet, seja através de redes móveis 3G/4G ou por Wifi, existe um cenário particular no qual nenhuma outra tecnologia cumpre tão bem quanto esta. Esse cenário é o da sincronização exacta com transmissões televisivas. Sabendo que uma transmissão televisiva está sujeita a atrasos variáveis (dependendo por exemplo do meio de transmissão), imaginando que existe uma aplicação para *smartphone* que apresenta conteúdo acerca do que está a ser transmitido na televisão, é impossível sincronizar o conteúdo apresentado sem recorrer ao áudio da emissão televisiva. Assim, sabendo que o áudio da emissão está sempre sincronizado com a respectiva imagem, conseguimos fazer a sincronização através de *watermarks* que serão compatíveis com gravações e sistemas de visualização *on demand*.

Apreciação Geral

Apesar de ser possível indicar as principais valências de cada uma das aproximações apresentadas, certos aspectos podem variar dependendo das *frameworks* que implementam cada uma destas soluções dados os algoritmos específicos utilizados na implementação.

A Tabela 1 apresenta, na sua generalidade, as principais diferenças entre ambas as aproximações. Nas linhas da primeira coluna da tabela são apresentadas várias características vantajosas e nas colunas 2 e 3 é indicado se essas características se verificam ou não nas duas aproximações analisadas.

Tabela 1 - Apreciação geral das vantagens/desvantagens das diferentes aproximações

	Inaudível	Audível Imperceptível
Indistinguível pelo Humano	Sim	Sim
Compatibilidade com *hardware* existente	Não	Sim
Resistente ao ruído ambiente	Sim	Não

Ainda que uma avaliação tendo em conta valores do tipo “Sim” ou “Não” nem sempre seja a mais correcta, a Tabela 1 demonstra, de forma geral e resumida, as capacidades de cada uma das aproximações identificadas anteriormente. Resumindo, considera-se que ambas as aproximações implicam alterações no áudio que são indistinguíveis pelo Humano. No entanto, dadas frequências usadas na aproximação

inaudível, permite que teoricamente exista menos susceptibilidade ao ruído ambiente. Por outro lado, prevê-se que esta aproximação apresente eventuais problemas de incompatibilidade com o vasto *hardware* existente que poderá não ter sido optimizado para operar nas frequências inaudíveis.

Deve-se ainda referir que pela natureza de cada uma das aproximações, a "audível imperceptível" não funcionará em faixas de áudio silenciosas. Isto porque as técnicas utilizadas nesta aproximação baseiam-se na manipulação do sinal original que, no caso de uma faixa de áudio silenciosa, é inexistente [7].

4.1.3. Soluções Existentes

Nesta secção é efectuada uma descrição geral e condensada das plataformas analisadas. Esta análise foi feita essencialmente com base em documentação oficial existente, bem como em testes realizados através de um protótipo desenvolvido especialmente para o efeito recorrendo aos *SDKs* disponibilizados por cada uma das entidades que desenvolveram as soluções.

Soundpays

A *Soundpays* [8] é uma *startup* recente que oferece um produto que consiste numa carteira digital cujos pagamentos são realizados de forma segura, através de tons sonoros inaudíveis. Além desse produto, afirmam também no seu *website* que a sua tecnologia está preparada para suportar casos de uso de interacção com conteúdo televisivo bem como qualquer conteúdo de vídeo previamente preparado para incluir informação digital na sua faixa áudio. Houve desde logo interesse em contactar esta *startup* pois já tinha existido um contacto passado com a WIT que suscitou curiosidade sobre a tecnologia usada. Infelizmente, mesmo após contactar a entidade, não ofereceram nenhum *SDK* que permita a outros desenvolvedores utilizarem a sua tecnologia.

Digimarc

A *Digimarc* [9] é outra empresa com experiência na tecnologia de *audio watermarking* apresentando também conhecimentos em *mobile consumer engagement* na detecção de códigos de barras e *QR Codes*. Ao contrário da empresa anterior, a *Digimarc* oferece um *SDK* para as plataformas Android e iOS sem qualquer entrave, basta efectuar o registo na plataforma. Têm também uma aplicação *mobile* que serve de demonstração da tecnologia que oferecem.

Foi possível testar o *SDK* em Android e verificar que é integrado com um *backoffice* onde é possível adicionar *audio watermarks* a um vídeo arbitrário. É também possível definir a informação associada a esses *watermarks*, ainda que seja limitada apenas a um

pequeno valor de textual e um endereço URL a ser apresentado quando o *watermark* é detectado na aplicação *mobile*. Trata-se, portanto, de uma solução completa para *audio watermarking* que se verificou usar a aproximação audível imperceptível. No entanto, quando comparado o áudio original com o áudio processado pela plataforma da *Digimarc*, é possível escutar algum ruído adicional, semelhante a distorção, que compromete a qualidade do áudio e o conceito de ser realmente imperceptível.

Em conjunto com o *SDK*, é disponibilizada documentação e um projecto de aplicação que serve de *template* facilitando o entendimento de como funciona a integração do *SDK* e, por consequência, o desenvolvimento de aplicações recorrendo a esta tecnologia.

Cifrasoft Soundcode

A *Cifrasoft* [10] é uma empresa dedicada a *second screen apps* [11], análise de audiências e comunicação entre dispositivos, áreas nas quais aplica o *audio watermarking*. No caso da comunicação entre dispositivos, utilizam a sua tecnologia num produto denominado *SoundLogin* que permite facilitar o processo de autenticação em *websites* através do *smartphone*. Além de utilizarem a tecnologia em produtos próprios, disponibilizam também um *SDK* para que terceiros possam criar os seus produtos. No entanto, a obtenção deste *SDK* não é imediata e requer um contacto inicial que foi realizado com sucesso no nosso caso.

Tal como na solução anterior, este *SDK* suporta Android e iOS vem acompanhado de documentação e código de exemplo que facilita a sua utilização. Ao contrário da solução da *Digimarc*, aqui não é oferecido nenhum *backoffice* sendo que adição de *watermarks* a ficheiros de áudio ou vídeo deve ser realizada através de uma ferramenta com interface gráfica, apresentada na Figura 5, dedicada exclusivamente para esse efeito e disponibilizada juntamente com o SDK. Mais tarde, foi também possível obter a mesma ferramenta, mas sem GUI, para correr em servidores Linux.

A Figura 5 apresenta uma captura de ecrã da ferramenta com interface *GUI* que foi disponibilizada pela *Cifrasoft*, na qual é possível perceber que é possível indicar um ficheiro de áudio/vídeo para ser processado com um determinado "id" (código injectado na faixa de áudio).

Verificou-se, através de uma aplicação semelhante a uma espectrograma, que a aproximação utilizada é mais uma vez a audível imperceptível. Neste caso, não foi realmente perceptível qualquer diferença entre o áudio com *watermarks* e o áudio original.

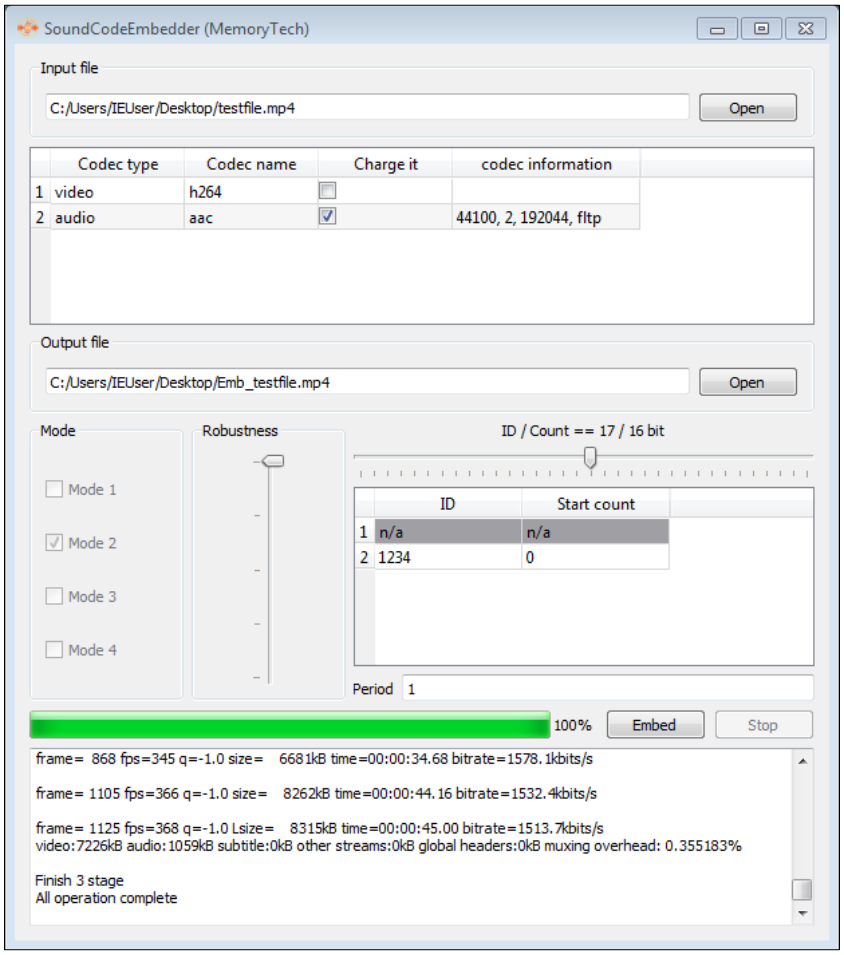


Figura 5 - Captura de ecrã da ferramenta de injecção de *watermarks* da Cifrasoft

Intrasonics

A *Intrasonics* [12] ,tal como a *Cifrasoft*, foca-se na utilização de *audio watermarking* para *second screen apps*, análise de alcance de publicidade e ainda aplicações interactivas para rádio e eventos ao vivo. Segundo a informação apresentada no *site* oficial da empresa, baseiam-se numa técnica denominada *artificial echo modulation* sendo portanto mais uma solução audível imperceptível. Dizem ainda que a sua tecnologia já está a ser usada por estações televisivas tais como a *BBC*, *Channel4*, entre outros, com múltiplas aplicações já disponíveis no mercado.

Para obter o *SDK* que disponibilizam, pedem um pagamento imediato de 99 libras. O *SDK* inclui uma ferramenta de *encoding*, exemplos de código, documentação e suporte adicional. Após um contacto directo foi possível obter o *SDK* de forma gratuita para fins de experimentação.

LISNR

A *LISNR* [13] é mais uma empresa que produz *software* recorrendo ao *audio watermarking* para emissões televisivas, eventos ao vivo (especialmente desportivos) e lojas. Pela quantidade de parceiros (*Warner Bros, T Mobile, AT&T, Heineken, Budweiser,* etc) e de implementações que apresentavam recorrendo a esta tecnologia, demonstravam ser um dos melhores parceiros possíveis. Foi também interessante que, após ter contactado com três entidades que usam a aproximação audível imperceptível, finalmente encontrávamos quem apostasse nas frequências inaudíveis.

Após um contacto inicial, foi possível obter acesso a um *SDK* multiplataforma (Android e iOS) bem como documentação e código de exemplo, tal com aconteceu com os concorrentes já apresentados. Neste caso, a plataforma *web* da *LISNR* permite gerar *watermarks* inaudíveis sendo necessário descarregar e fazer a mistura manualmente com um conteúdo multimédia arbitrário. Se por um lado, em relação aos outros *SDKs* analisados, tem o inconveniente de ser necessário que este passo final seja realizado manualmente, por outro lado demonstra que basta gerar os *watermarks* uma vez e reutilizar em quantos vídeos/músicas se queira porque na realidade, tratando-se da aproximação inaudível, não é necessário processar o áudio original para gerar *watermarks*.

Prontoly

A Prontoly [14] apresenta-se como uma empresa focada na utilização de ondas sonoras inaudíveis para comunicação *device-to-device* tal como no caso específico da autenticação por proximidade. Segundo a informação disponível no *website*, disponibilizariam o seu *SDK* mediante um pedido expresso. No entanto, após contacto com o *product manager* desta solução, apenas nos foi apresentada uma demo para efeitos de teste que consistia numa aplicação Android para detectar *watermarks* inseridos num vídeo específico.

SonicNet

Na tentativa de encontrar alguma alternativa *open-source*, descobriu-se a biblioteca *SonicNet* [15]. *SonicNet* é uma biblioteca *Javascript* que implementa a comunicação entre dispositivos através de tons sonoros nas frequências inaudíveis. Esta biblioteca é acompanhada por uma demonstração que foi testada de imediato e, empiricamente, foi possível desde logo perceber que era uma solução que ainda não apresentava a robustez necessária: por vezes os tons sonoros eram audíveis e nem todos os dispositivos testados (*smartphones* e computadores portáteis) conseguiam comunicar entre si. Por isto e também por se tratar de uma biblioteca *Javascript* (pretendia-se algo nativo) não foi aprofundada a análise desta biblioteca.

4.1.4. Apreciação Geral das Frameworks

A Tabela 2 lista o resumo da análise comparativa das diversas soluções analisadas, em que se exclui a *SonicNet* pelas razões já apresentadas.

Tabela 2 - Comparação resumida das várias soluções analisadas

	Aproximação	SDK	Disponibiliza demo	Ferramenta de *embedding*	Presença no mercado
Soundpays	(n.a)	✗ R	✗	(n.a.)	✗
Digimarc	Audível imperceptível	✓	✓	Web	✗
Cifrasoft	Audível imperceptível	✓ R	✓	GUI/CLI	✓
Intrasonics	Audível imperceptível	✓ R	✓	CLI	✓
LISNR	Inaudível	✓ R	✓ R	Manual	✓
Prontoly	Inaudível	✗ R	✓ R	(n.a.)	✗

"R" – Não está publicamente disponível, necessita de ser requisitado

Na primeira coluna da tabela são apresentadas as diversas soluções analisadas e nas colunas 2 a 6 são distinguidas essas soluções quanto a características relevantes tais como: a aproximação utilizada para injecção de *watermarks*, o facto de oferecerem um *SDK* (imediatamente ou após ser requisitado), se disponibilizam uma demonstração da solução, qual o tipo de interface da ferramenta de *embedding* de *watermarks* (interface *web*, ferramenta com interface gráfica, ferramenta com interface por linha de comandos ou o *embedding* deve ser feito manualmente) e se já têm alguma aplicação disponível no mercado que recorra à tecnologia de *audio watermarking*.

Como se pode verificar na tabela, a maior parte dos *SDKs* só estão disponíveis após pedido, implicando fosse necessário contactar todos. Durante alguns desses contactos foi possível perceber que algumas das entidades não tinham realmente nada pronto para oferecer a este nível. Percebeu-se que existia alguma diversidade no método de *embedding* e que a WIT tinha preferência por algo que fosse possível integrar directamente num servidor sem qualquer necessidade de interacção humana posterior. Portanto a melhor opção seria evitar ferramentas *GUI* e *Web* e preferir ferramentas *CLI*.

A nível de suporte para as várias plataformas móveis, todos os *SDKs* eram compatíveis com Android e iOS.

Pode-se também dizer que todos os *SDKs* são simples de utilizar dado que não existiram grandes dificuldades a implementar uma pequena aplicação Android recorrendo a todos os *SDKs* com a intenção de ser utilizada para testes futuros. Neste ponto, a existência de código de exemplo em todos os SDKs ajudou bastante.

Um factor importante para a análise foi também a existência de aplicações no mercado que recorressem a cada um dos *SDKs*, dado que seria sinal que a tecnologia apresenta qualidade suficiente para o público geral e ajudaria a perceber a aceitação da tecnologia através da análise dessas mesmas aplicações. Algumas dessas aplicações são apresentadas na secção "4.1.6 - Casos de Uso e Implementações Existentes".

Conseguimos então chegar a um conjunto de *SDKs* variado, que levou à necessidade de testar a performance de cada um deles. Esses testes são descritos no capítulo "4.2. Testes Realizados às Soluções Existentes".

4.1.5. Watermarking e Fingerprinting

Como alternativa à tecnologia de *watermarking* foi considerado o *fingerprinting*. Esta tecnologia difere do *watermarking* no sentido em que não é modificado o áudio original. É apenas calculado um identificador único com base em características específicas do áudio permitindo identificar unicamente uma faixa de áudio, tal como uma música [16]. Foi feita uma pequena análise comparando a utilidade de cada uma das tecnologias em casos de uso semelhantes e foi também testada a solução de *fingerprinting* fornecida pela *Cifrasoft*. Optou-se por não aprofundar a análise porque se preferiu, para já, utilizar apenas *watermarking*.

4.1.6. Casos de Uso e Implementações Existentes

Além da análise das plataformas disponíveis para o desenvolvimento de aplicações móveis recorrendo ao *audio watermarking*, foi também recolhido um conjunto de aplicações existentes no mercado que recorrem a esta tecnologia. Pretendeu-se essencialmente obter uma visão geral de como têm sido utilizados os *audio watermarks* e de como podem vir a ser utilizados nos mais diversos contextos tais como aplicações dedicadas para televisão/rádio, eventos ao vivo e comunicação entre dispositivos.

Aplicações para Televisão

Foram encontradas várias aplicações que usam o *audio watermarking* no contexto da televisão.

Um exemplo é a aplicação *Horse Tracker* [17] da estação televisiva *Channel4* que utiliza *watermarks* para sincronizar de forma precisa as corridas de cavalos emitidas nessa

mesma estação com uma aplicação móvel. Esta aplicação móvel, ao detectar *watermarks*, apresenta ao utilizador informações relevantes, em tempo real, acerca da corrida tal como a posição de um cavalo específico, a sua velocidade ou a tabela de classificação geral.

Outra aplicação que serve de acompanhante a um programa televisivo é a *QI Test* [18] da estação francesa *TF1*. Esta aplicação serviu para avaliar o QI (quociente de inteligência) dos telespectadores através de um *quizz* que decorria num programa televisivo, produzido especialmente para este efeito, com o qual a aplicação móvel estaria sincronizada e daria a hipótese de os telespectadores submeterem as suas respostas às questões que iam sendo colocadas. Trata-se, portanto, de mais um exemplo de uma aplicação que utiliza os *watermarks* para sincronizar o conteúdo televisivo com a informação que é apresentada na aplicação móvel.

Ainda no contexto televisivo, existiu também um protótipo do *eBay* denominado *Watch With eBay* [19] que esteve anteriormente no mercado e permitia que os telespectadores comprassem produtos relacionados com o programa televisivo ao qual estavam a assistir. Neste caso, os *watermarks* permitiam à aplicação móvel identificar o programa televisivo que estava a ser visualizado e, através disso, seriam apresentados produtos relacionados. Por exemplo, se o utilizador estivesse a ver um filme, a aplicação identificaria o filme e sugeria ao utilizador que comprasse as roupas que os actores estavam a vestir. Este é um caso de aplicação dos *watermarks* nas compras por impulso.

Num estilo ligeiramente diferente, a empresa norueguesa de produtos alimentares Solo colocou no mercado a aplicação *Solo Goodiebag* [20] que permitia ao utilizador acumular pontos e ganhar prémios por ver os anúncios publicitários dessa empresa. No fundo, os anúncios televisivos continham *audio watermarks* que ao serem detectados na aplicação móvel se traduziam em pontos que podiam ser acumulados e trocados por *merchandising* da marca. Este caso consiste, portanto, na utilização de conceitos de *gamification* [21] para motivar as pessoas a verem os anúncios da marca e criar *brand awareness*. É importante referir ainda que esta aplicação foi desenvolvida com recurso ao *SDK* da *Intrasonics* que também analisámos.

Por último, outra aplicação móvel que suscitou a nossa curiosidade foi a *APP* [22] dedicada a um filme holandês com o mesmo título. A aplicação permite desbloquear conteúdo extra (acerca do elenco, cenas cortadas, curiosidades, etc) ao longo do filme, à medida que os vários *audio watermarks* são detectados.

Mais exemplos de aplicações poderiam ser mencionados dentro do contexto televisivo, mas com este conjunto já é possível perceber que tipo de casos de uso são possíveis criar à volta da tecnologia de *audio watermarking*. Note-se que, como já foi referido anteriormente, os casos de uso que requerem sincronização precisa com uma emissão televisiva, dificilmente seriam tão bem cumpridos sem utilizar a sincronização

através do áudio, dado o atraso variável nos diferentes meios de transmissão (satélite, digital, etc).

Aplicações Para Eventos ao Vivo

No contexto de eventos ao vivo, foi encontrada uma aplicação móvel desenvolvida através pela própria *LISNR* que oferece um dos já referidos *SDKs* (e que usa *watermarks* inaudíveis). Trata-se de uma aplicação que foi testada no evento *Budweiser Made in America* [23] . Com esta aplicação, os visitantes deste evento poderiam obter descontos em produtos vendidos nas concessões de bebidas cada vez que se aproximavam fisicamente dos seus locais. De forma semelhante, recebiam descontos em viagens feitas através do *Uber* quando passavam na saída do recinto.

Aplicações Para Comunicação Entre Dispositivos

Quanto a aplicações que estão a usar esta tecnologia para comunicar entre dispositivos, foi possível analisar uma pequena aplicação para PC feita pela Google denominada *Google Tone* [24] que permite a partilha de endereços *web* com outros PCs que se encontram fisicamente próximos. Trata-se apenas de uma extensão para o *browser* Google Chrome e permite que os endereços visitados por um utilizador sejam partilhados com outras pessoas que estejam próximas de si apenas ao pressionando um botão.

Outro exemplo é a aplicação *Furby Boom* [25], uma aplicação para crianças, dedicada a um brinquedo com o mesmo nome. O objectivo desta aplicação é comunicar com o brinquedo desbloqueando jogos interactivos no *smartphone*/*tablet*.

Dada a preocupação recorrente em melhorar a segurança na Internet, existe também uma solução que recorre à tecnologia de áudio denominada *SoundLogin* [26]. Esta aplicação permite que o utilizador se autentique numa plataforma *web* utilizando o seu *smartphone* em vez de uma *password*. Isto é, ao aceder a um *site* que requer autenticação através do PC, o utilizador pode autenticar-se através de um canal de comunicação encriptado por áudio entre o seu PC e o seu *smartphone* pessoal. Basta para tal abrir a aplicação no *smartphone* e dar início à autenticação enquanto o PC se encontra à escuta de informação relevante através do microfone.

Ambas as aplicações comunicam através do som nas frequências inaudíveis.

Aplicações em Contextos Alternativos

Durante a pesquisa de aplicações existentes no mercado que recorressem ao *audio watermarking*, foram também encontradas aplicações que não se encaixam nos contextos apresentados anteriormente mas que, ainda assim, se consideram importantes mencionar para demonstrar efectivamente a diversidade de casos de uso em que esta tecnologia se pode inserir. Por exemplo, a aplicação *Shazam in-Store* [27] pretende atrair as pessoas às

lojas oferecendo ofertas especiais e conteúdo exclusivo enquanto uma pessoa visita uma loja. Neste caso a musica ambiente da loja contém *watermarks* que, ao serem detectados por uma aplicação *mobile*, indicam que o utilizador está na loja e pode receber as ofertas especiais.

A tecnologia de *audio watermarking* também já foi testada no cinema, tal como é o caso da aplicação móvel *Cinime* [28]. Esta aplicação foi pensada para entreter as pessoas que vão ao cinema e enquanto o filme não começa é apresentado um jogo tipo *quizz* no ecrã do *smartphone* através da utilização do sistema de som já existente nas salas de cinema para emitir os *watermarks* que despoletam a apresentação de perguntas específicas para o utilizador responder.

4.1.7. Conclusão

Toda a pesquisa e análise referidas neste capítulo foram úteis para perceber o quão evoluída está a tecnologia de *audio watermarking* e as suas possíveis utilizações nos mais diversos cenários. Percebeu-se que, tecnicamente, é relativamente fácil implementar esta tecnologia em aplicações móveis recorrendo às soluções existentes. No entanto, essas aplicações devem envolver a tecnologia em algo que o utilizador valorize porque, apesar de tecnicamente o *audio watermarking* ser interessante, a tecnologia por si só não conseguirá cativar os utilizadores a usarem uma aplicação.

Tendo realizado esta análise, interessava agora testar cada um dos *SDKs* para perceber qual o mais capaz em diferentes cenários.

4.2. Testes Realizados às Soluções Existentes

Esta secção descreve os testes realizados aos vários *SDKs* apresentados em "4.1.3 Soluções Existentes" e os resultados obtidos. Para este efeito, foi desenvolvida uma pequena aplicação que permite alternar entre cada um dos quatro *SDKs* que foram possíveis obter (*Digimarc*, *Cifrasoft*, *Intrasonics* e *LISNR*) e detectar *watermarks* num vídeo a partir do qual foram gerados quatro vídeos iguais mas onde cada um contém as *watermarks* específicas de cada *SDK*. Como a *Prontoly* não ofereceu um *SDK*, foi usada a aplicação de demonstração disponibilizada por eles para a realização destes testes. Deve-se ter então em conta na análise de resultados que este *SDK* não foi testado de igual para igual com as restantes soluções disponíveis.

A Figura 6 apresenta três capturas de ecrã da aplicação desenvolvida para testes: um ecrã com uma lista para seleccionar o *SDK* a utilizar, um ecrã com o *SDK* da *Cifrasoft* activo e, por fim, um ecrã após ter sido detectado um *watermark* através desse mesmo *SDK*.

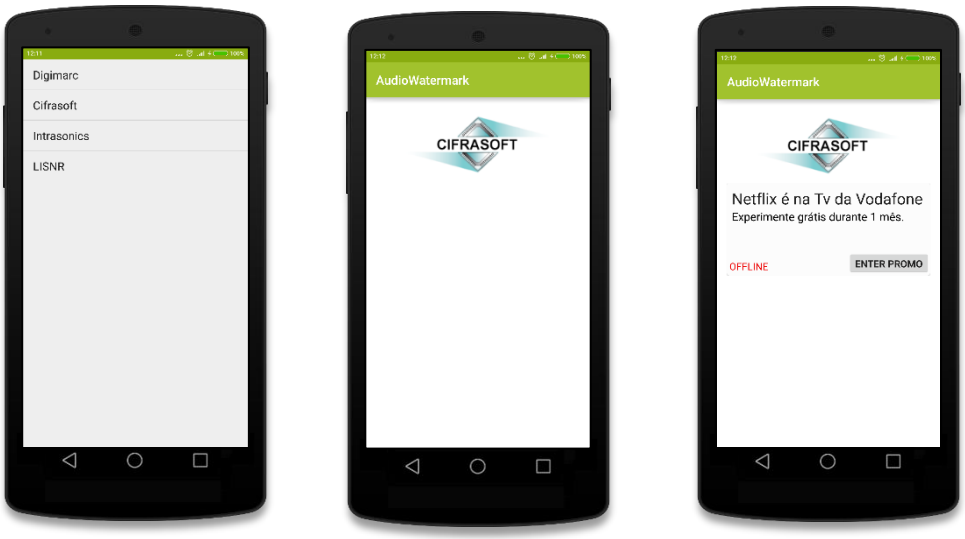


Figura 6- Capturas de ecrã da aplicação desenvolvida para testes

4.2.1. Ambiente de Testes

Cada um dos *SDKs* foi testado em dois ambientes simulados: ambiente silencioso e ambiente ruidoso. Dado que foi necessário testar tecnologia sem qualquer influência de ruído ambiente, foi utilizada uma sala completamente vazia para realizar os testes. Nesta sala foi colocada uma televisão onde foram emitidos os vídeos com *watermarks*, como demonstra o registo fotográfico na Figura 7. Para efeitos de eventual reprodução dos testes, o modelo da televisão era uma "Samsung UE40F6320AW" cujo volume foi colocado no nível 45. No canto oposto da sala, aproximadamente a 8 metros de distância da televisão, foram colocados três *smartphones* diferentes com a aplicação de teste instalada: um *Samsung Galaxy S3*, um *Samsung Galaxy Nexus* e um *Xiaomi Redmi Note 2*.

A Figura 7 apresenta algumas fotografias da sala usada para testes na qual é possível ver o televisor num canto e os vários *smartphones* no canto oposto da sala.

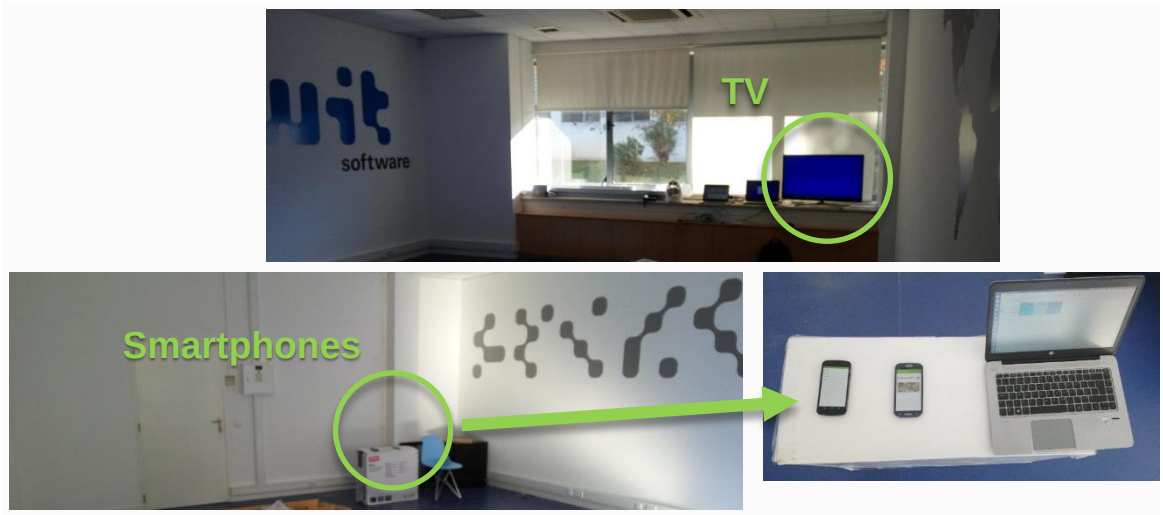
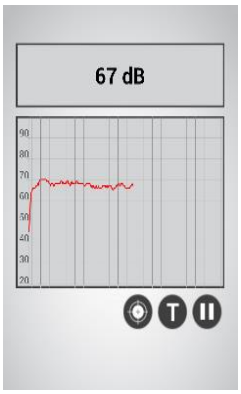
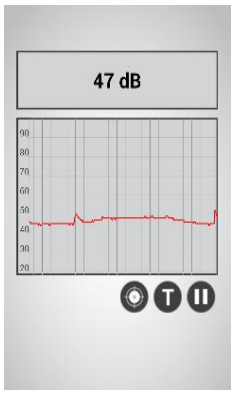


Figura 7- Registo fotográfico do ambiente de testes

Para simular um ambiente ruidoso, foi reproduzido continuamente um som semelhante a um local barulhento na mesma televisão onde foram reproduzidos os vídeos. Utilizando duas aplicações diferentes para medir a intensidade do som em ambos os contextos, obtiveram-se valores a rondar os 50 decibéis no ambiente silencioso e os 70 decibéis no ambiente ruidoso. Note-se que estes valores servem apenas para referência pois não é possível garantir exactidão dado que foi utilizado um *smartphone* (*Samsung Galaxy S3*) para efectuar esta medição

A Tabela 3 mostra os resultados de ruído ambiente obtidos através de duas aplicações diferentes. As linhas da tabela referem-se aos dois tipos de ambiente testados e as colunas dizem respeito às duas aplicações usadas para fazer a medição: *SoundMeter PRO* [29] e *Sound Meter for Android* [30].

Tabela 3 - Medição de intensidade do ruído ambiente através das aplicações “SoundMeter PRO” e “Sound Meter for Android”

	Aplicação “SoundMeter PRO”	Aplicação “Sound Meter for Android”
Ambiente ruidoso		
Ambiente silencioso		

Nos *smartphones* usados para detecção e nos SDKs, além da variação no ruído ambiente, introduziu-se também variação no volume em que os vídeos foram reproduzidos. Isto é, mantendo a televisão no nível 45 de volume, o volume de reprodução do vídeo variou entre os valores de 100% e 50%.

Foram então executados, no total, 60 testes (5 *SDKs*, em 3 *smartphones*, 2 ambientes e 2 níveis de volume).

4.2.2. Resultados

Os resultados completos podem ser consultados no “Anexo A - Resultados dos Testes Realizados às Várias Soluções de Watermarking”. A Tabela 14 apresentada no referido anexo, demonstra os resultados dos testes não tratados e organizados da seguinte forma: na primeira coluna (*SDK*) constam os diversos *SDKs* testados, na segunda coluna (*Watermark Volume*) estão os dois níveis de volume testados para cada *SDK*, na terceira coluna (*Expected*) é apresentado o valor esperado de detecção na utilização de um determinado *SDK* (os valores variam por *SDK* porque cada um deles injecta uma diferente

quantidade de *watermarks* por minuto), da coluna 4 a 6 (*Noisy Environment*) são apresentados os resultados das detecções obtidas no ambiente ruidoso para os três dispositivos mencionados anteriormente e as colunas 7 a 9 (*Silent Environment*) dizem respeito aos resultados obtidos no ambiente silencioso utilizando os mesmos três dispositivos.

A Tabela 15 no mesmo anexo apresenta os de detecção transformados em forma de percentagem para facilitar a interpretação.

A Tabela 16, que consta também no referido anexo, é uma simplificação da anterior onde se ignoram os resultados específicos de cada dispositivo. Assim, facilita a interpretação dos resultados de cada SDK tendo apenas em conta os dois ambientes testados de forma geral.

Analisando os resultados, é possível perceber que a solução da *LISNR* teve, no geral, as piores prestações. Além de ter demonstrado dificuldade em detectar os *watermarks*, revelou ainda variação dependente do *smartphone*. Por outro lado, a solução da *Cifrasoft* apresentou bons resultados com variação entre *smartphone*s praticamente nula. Apenas falhou no caso mais extremo (ambiente ruidoso com volume do vídeo a 50%). A demo da *Prontoly demonstrou* resultados ligeiramente inferiores à da *Cifrasoft* mas foi capaz de detectar *watermarks* em qualquer ambiente.

4.2.3. Teste Adicional de Sobrevivência ao Transcoding

Existiu alguma curiosidade em perceber se os *watermarks* da melhor solução encontrada, a da *Cifrasoft*, sobreviveriam ao *transcoding* para formatos com menos qualidade de áudio. Experimentou-se então fazer *upload* de um vídeo com *watermarks* para o *YouTube*, descarregando os vários formatos que o *YouTube* suporta (*MP4, FLV, WEBM, 3GPP*) e, de seguida, testando se era possível continuar a detectar os *watermarks*. Os resultados foram bastante positivos visto que foi possível detectar *watermarks* mesmo no vídeo com formato *3GPP* no qual o áudio perdeu imensa qualidade (o *bitrate* era 23*kbps*).

4.2.4. Conclusão

A realização destes testes permitiu perceber qual dos *SDKs* devíamos utilizar para desenvolver soluções futuras recorrendo a este tipo de tecnologia. Identificámos então a solução da *Cifrasoft* como a melhor opção .

Foi possível perceber que, ao contrário do que teoricamente se previa, a aproximação inaudível não é garantidamente menos susceptível ao ruído ambiente tal como evidenciam os resultados do *SDK* da *LISNR*. Percebeu-se também que a teoria que previa dificuldades em garantir a capacidade de detecção de *watermarks* inaudíveis em diferente *hardware* provavelmente também se verifica na prática. Não o podemos garantir com toda a certeza, porque apenas foi possível testar o *SDK* da *LISNR* e apenas foram usados três

smartphones diferentes. Ainda assim os resultados deste *SDK*, contrastando com os resultados dos *SDKS* que usam a aproximação audível imperceptível, revelaram essa mesma tendência de variância na detecção de *watermarks* em dispositivos diferentes.

Apesar dos testes realizados terem permitido obter informação relevante seria interessante, por exemplo, testar o *SDK* da *Prontoly* em vez de usar a aplicação de demonstração oferecida por esta entidade, de forma a testar todos os concorrentes de igual para igual. Actualmente é difícil comparar a prestação da solução da *Prontoly* com as restantes pois nada nos garante que esta aplicação desenvolvida por eles não tenha sido optimizada especialmente para este tipo de testes ou para o vídeo específico que oferecem para testar com a aplicação de demonstração. Outras possíveis indicações para testes futuros seriam: adicionar mais variação nos dispositivos usados para detectar os *watermarks* (só foram usados 3 de um vasto universo) e também no dispositivo usado para emitir os vídeos que continham as *watermarks* (a televisão especifica utilizada pode ter influenciado os resultados obtidos).

5. WP 200: Prototipagem rápida

Após analisar e testar a tecnologia, surgiu interesse em desenvolver uma aplicação para fins demonstração. Para tal, numa primeira fase, foram desenvolvidas pequenas aplicações do mesmo tipo, mas com objectivos diferentes: uma para demonstração interna, outras para demonstração a clientes específicos e uma outra que acompanhou a WIT Software até Barcelona na *MWC 2016* (*Mobile World Congress*). Posteriormente foram também desenvolvidas outras duas aplicações, reutilizando o módulo de detecção de *watermarks*, que exploravam mais as questões da interface gráfica. Este capítulo descreve e detalha o desenvolvimento das referidas aplicações.

5.1. Descrição Funcional

Apesar dos diferentes objectivos, todas as aplicações partilhavam o conceito base, de detecção de em vídeos através de *smartphones*.

Na aplicação destinada a demonstrações internas (*DemoApp1*), a detecção de *watermarks* despoletava um ecrã de compra de produtos relacionados com os vídeos. Na aplicação destinada a apresentar a tecnologia aos possíveis clientes (*DemoApp2*), foram reutilizados os mesmos vídeos e respectivos *payoffs*, e foram também consideradas algumas alterações ao nível de interface da aplicação *mobile* e na forma como eram reproduzidos os vídeos para um cliente específico: foi integrado um *stream HLS* (*HTTP Live Streaming*) dos vídeos numa aplicação que estava a ser desenvolvida noutro projecto (para esse mesmo cliente) para *set top boxes*. Ou seja, os vídeos eram reproduzidos através dessa aplicação numa televisão.

Na aplicação usada na feira *MWC* (*DemoApp3*), foram injectados *watermarks* em quatro vídeos (relativos a produtos *white-label* da WIT Software que estavam a ser apresentados) que estiveram a ser reproduzidos em repetição no *stand* da empresa. Ao detectar esses *watermarks* através de uma aplicação *mobile*, era possível obter mais informação acerca dos respectivos produtos.

5.2. Arquitectura do Módulo de Detecção de Watermarks

Todas as aplicações reutilizaram o mesmo módulo que trata da obtenção e processamento áudio do microfone cuja arquitectura foi dividida em classes com responsabilidades bem definidas:

- *SoundCodeSettings* – Classe onde são definidos valores de configuração do comportamento do módulo.
- *SoundCode* – S*ingleton* com métodos para iniciar e interromper todo o processo de captação e processamento de áudio para identificação de *watermarks*.

- *SoundCodeReceiver* – Classe que cria um contexto em *background* e arranca a captação de áudio nesse mesmo contexto através da classe *AudioRecordingThread*. Redirecciona resultados de *watermarks* detectados para instâncias da interface *SoundCodeListener*.
- *AudioRecordingThread* – Classe responsável pela captação contínua de áudio através do microfone do dispositivo e pelo redireccionamento do *buffer* de áudio para a biblioteca nativa (é acedida através de uma *bridge JNI*) da Cifrasoft.
- *SoundCodeListener* – Interface a implementar por classes que queiram receber os resultados dos *watermarks* detectados.

Esta arquitectura foi também implementada para a plataforma iOS da Apple originando uma pequena aplicação que serve de prova de conceito para confirmar que o SDK utilizado funciona igualmente bem nessa mesma plataforma.

A Figura 8 apresenta o diagrama de classes do módulo desenvolvido em Android e iOS no qual é possível verificar as relações entre as classes referidas.

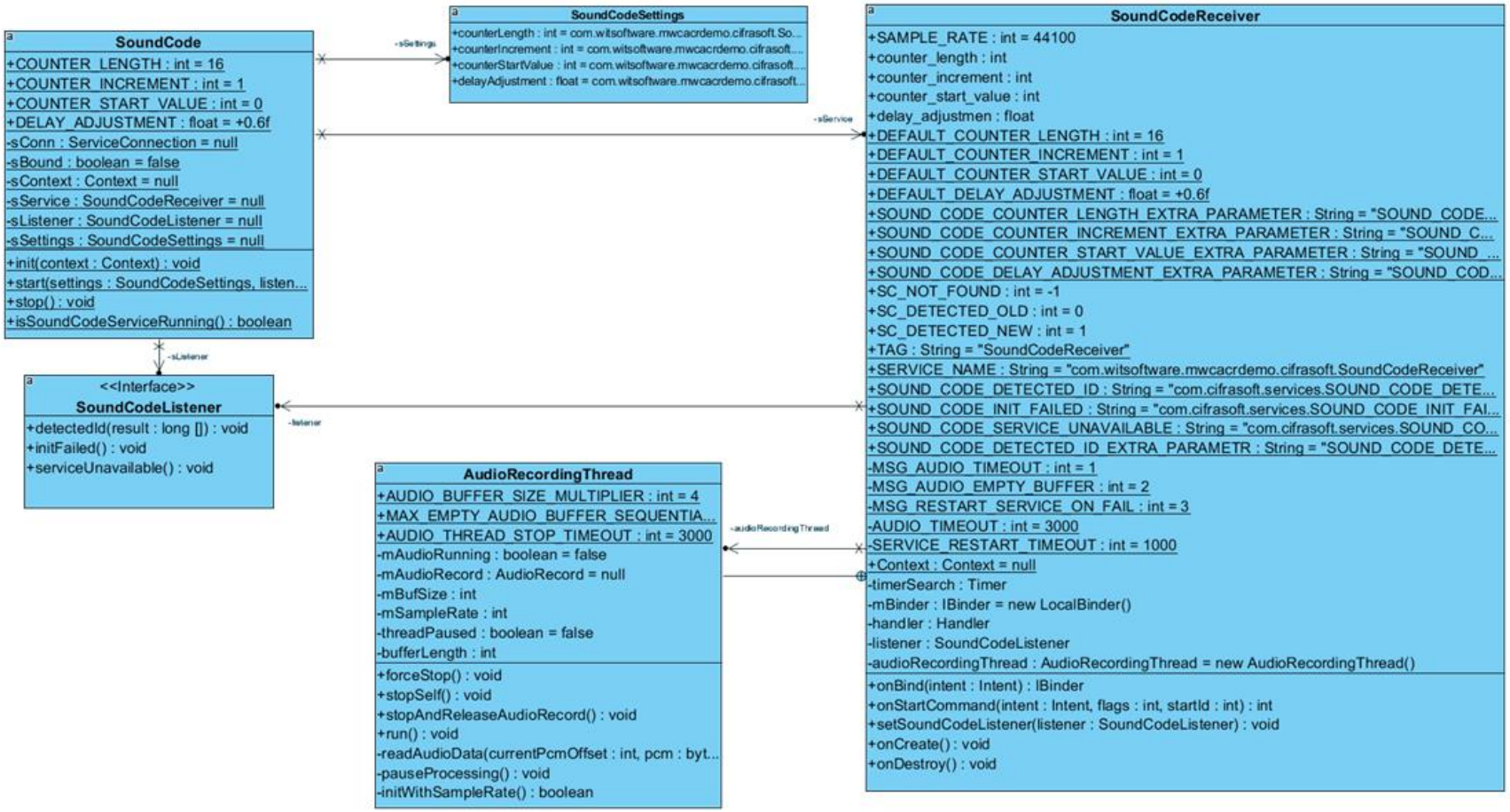


Figura 8 - Diagrama de classes do módulo de detecção de *watermarks*

5.3. Provas de Conceito Focadas na Interface

Tendo já explorado o desenvolvimento de aplicações para demonstração da tecnologia, surgiu também a necessidade de aliar essa tecnologia a conceitos mais exigentes. Por exemplo, seria interessante que qualquer pessoa pudesse criar o *layout* do ecrã despoletado num *smartphone* aquando da detecção de um *watermark* específico. Algo semelhante com o que a plataforma *Blippbuilder* [31] permite fazer com realidade aumentada. A plataforma *Blippbuilder* permite que qualquer pessoa possa criar o seu cenário de realidade aumentada que é apresentado cada vez que uma imagem específica seja detectada através da câmera do *smartphone*, através de um editor estilo *WYSIWYG* (*What You See Is What You Get*) com interacção *drag-and-drop*. Para tal, o estagiário desenvolveu uma pequena aplicação Android (*DemoApp4*) capaz de interpretar uma estrutura *JSON* (armazenada num servidor remoto) gerada de acordo com aquilo que o foi desenhado no editor *WYSWIG* e desenhar dinamicamente os ecrãs com base na informação dessa estrutura.

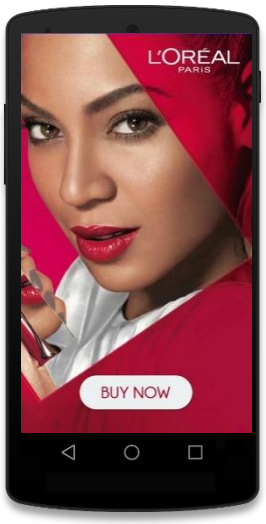
Foi suportado o desenho de imagens remotas, imagens remotas com ligação páginas web externas, vídeos remotos, galeria de imagens remotas, votações, botões com ligação a páginas web externas bem como a aplicações externas, realização de chamadas, envio de *e-mail*, registo de eventos no calendário e partilha de conteúdos nas redes sociais *Facebook* e *Twitter*.

Todos estes elementos podiam ser configurados dinamicamente quanto à sua posição no ecrã e quanto ao seu funcionamento. A Tabela 4 mostra um exemplo da estrutura *JSON*, na primeira coluna, e do resultado da interpretação e construção dinâmica do respectivo *layout* na segunda coluna.

Tabela 4- Exemplo de código *JSON* e resultado produzido na aplicação de demonstração *DemoApp4*

Código JSON	Resultado produzido

```
{
   "layout":[
      {
         "type":"image",
         "value":"http://i.imgur.com/93yt3a0.png",
         "scale_type":"center_crop",
         "position":{
            "x":0,
            "y":0,
            "width":100,
            "height":100
         }
      },
      {
         "type":"image_link",
         "value":"http://i.imgur.com/KniC9KG.png",
         "url":"http://www.lorealparis.pt/",
         "position":{
            "x":25,
            "y":85,
            "width":50,
            "height":10
         }
      }
   ]
}
```



Desenvolveu-se ainda outra aplicação de demonstração (*DemoApp5*) tendo como base este mesmo conceito de construção de ecrãs de forma dinâmica. No entanto, nesta aplicação não foi tido em conta a possibilidade de posicionar os elementos do *layout* dinamicamente, ainda que as acções disponíveis continuassem a ser apresentadas de forma dinâmica. Existia um *template* predefinido para o *layout* ao qual eram adicionados os componentes indicados na estrutura JSON em posições predefinidas. Este *template*, como exemplifica a Figura 9, considera uma imagem ou vídeo de fundo e um elemento gráfico que anima verticalmente e apresenta uma ou mais acções possíveis ao utilizador. É também possível reproduzir um ficheiro de som, também ele remoto, quando o ecrã é apresentado.

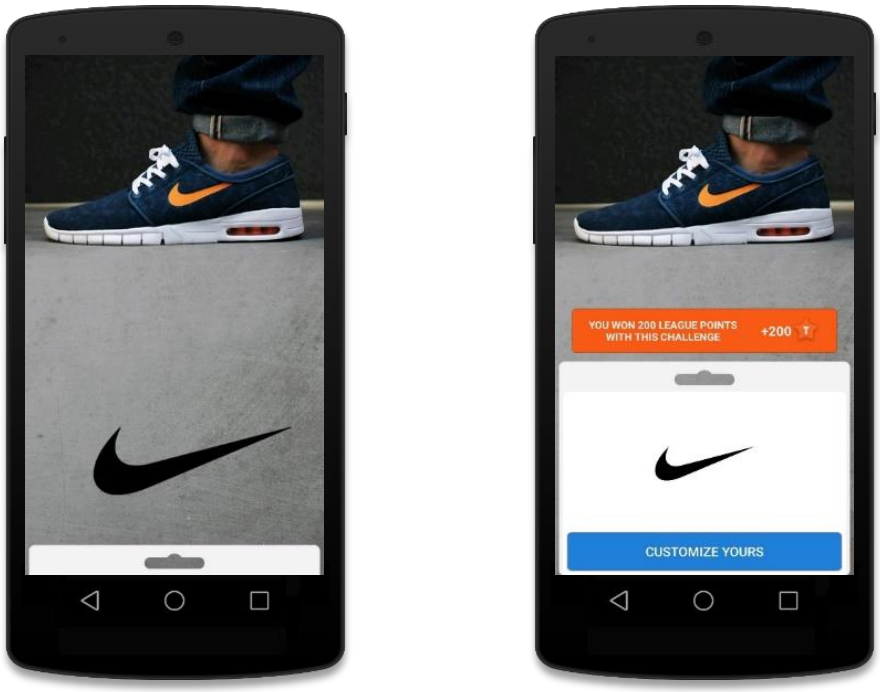


Figura 9- Template gráfico utilizado nos ecrãs da aplicação de demonstração *DemoApp5*

Esta aplicação de demonstração foi também integrada com a tecnologia de áudio *watermarking*. Ou seja, quando era detectado um *watermark* específico através do microfone do *smartphone*, era apresentado o ecrã correspondente. Esta aplicação foi mais focada em apresentar diversos casos-de-uso no contexto da publicidade, em vez do típico caso de compra por impulso, com o objectivo de ter uma demonstração mais apelativa para apresentar a potenciais clientes da WIT Software. Foram consideradas inicialmente catorze "campanhas" para diferentes cenários e marcas. Alguns dos exemplos são descritos na Tabela 5 onde, na segunda coluna, são apresentadas as marcas usadas para testar o caso de uso e, na terceira coluna, é descrito cada caso de uso.

Tabela 5 - Casos de uso explorados na aplicação de demonstração *DemoApp5*

	Marca	Caso de uso
1	**Zara**	Receber voucher de desconto na loja.
2	**Nike**	Personalizar sapatilhas e comprar imediatamente.
3	**Mini**	Pedir um *test-drive* de um veículo da marca.
4	**McDonalds**	Obter cupões de desconto.
5	**Sporting 1**	Comprar bilhete para um evento desportivo.
6	**Sporting 2**	Votar no jogador de futebol favorito do Sporting.

7	**Festival Paredes de Coura**	Ganhar um bilhete para o evento através de uma partilha no Facebook.
8	**Coca -cola**	Participar na campanha “share a coke”.
9	**BBCream**	Obter amostras de um produto.
10	**Blitz**	Assinar uma revista.
11	**BPI**	Ligar/enviar mail para saber mais sobre um produto financeiro.
12	**Calzedonia**	Acesso ao novo catálogo de produtos da marca.
13	**Billabong**	Comprar produtos da marca.
14	**Telepizza**	Encomendar uma pizza imediatamente

A Figura 10 ilustra os resultados de alguns dos vários casos de uso já referidos, nomeadamente para o caso da McDonalds, Festival Paredes de Coura, BB Cream e Blitz.

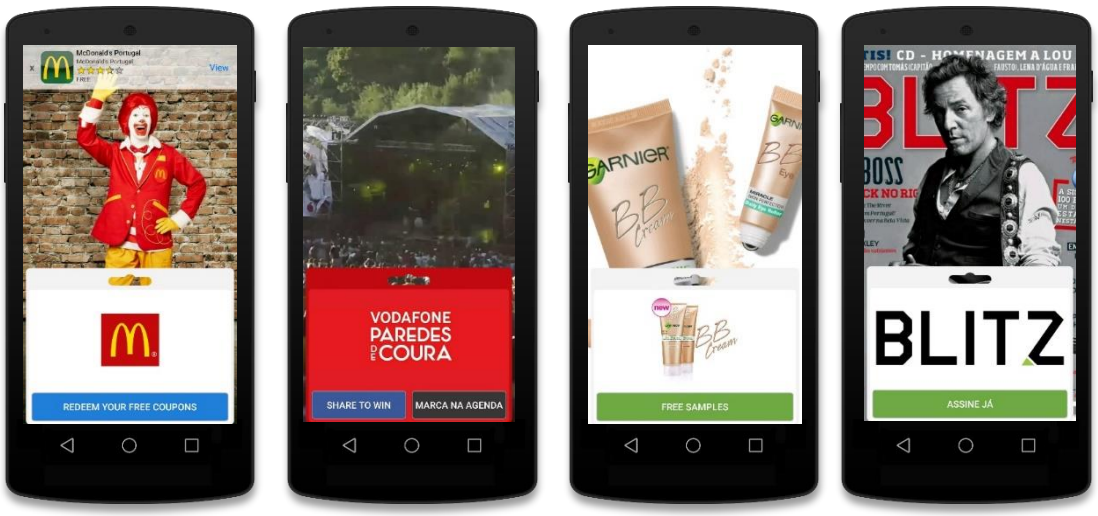


Figura 10 - Capturas de ecrã da aplicação de demonstração *DemoApp5* para os casos de uso 4, 7, 9 e 10

Para prevenir o caso de os utilizadores pressionarem um botão que os redirecciona para uma página *web* e terem que esperar demasiado tempo para que a página seja carregada, preferiu-se fazer o carregamento e renderização da página dentro da própria aplicação para de forma a ter mais controlo sobre esses aspectos. Assim foi possível iniciar previamente o carregamento das páginas *web* em *background* (de forma invisível) assim que as “campanhas” sejam detectadas, bastando mais tarde tornar a página *web* visível (já estando totalmente carregada), dando ao utilizador uma ilusão de rapidez que melhora a *user-experience*.

5.4. Ferramentas Usadas

Para desenvolver todas estas pequenas aplicações foram utilizadas, além daquela que é incluída no *SDK* da *Cifrasoft* dedicada à detecção dos *watermarks*, um conjunto de outras bibliotecas para Android.

Utilizou-se a biblioteca *Retrofit* [32] para a comunicação com *web services*. Esta biblioteca simplifica bastante o código necessário para fazer pedidos *HTTP* e tratar das respostas provenientes de *web services*.

Para ajudar na manipulação das estruturas *JSON*, foi utilizada a biblioteca *GSON* [33], aproveitando o facto de já ser usada pela biblioteca *Retrofit*. Esta biblioteca foi útil principalmente para fazer o *parsing* das estruturas referentes aos *layouts* dinâmicos.

Utilizou-se também a biblioteca *Picasso* [34] para carregamento de imagens remotas. Esta biblioteca permite também facilmente fazer *caching* das imagens, melhorando a *user-experience*.

Outra biblioteca bastante usada foi a *EventBus* [35] com o intuito de conseguir uma melhor separação de responsabilidades e remoção de dependências entre os componentes de interface gráfica e os de lógica. Como o próprio nome indica, permite criar uma *event queue* [36] que vários componentes podem subscrever e permitindo-os comunicar assincronamente. Ajudou também a manter as aplicações rápidas e responsivas graças a esta sua característica de assincronismo e também ao facto de facilitar a comunicação entre componentes que correm na *UI Thread* [37] e os que correm em *background threads*.

O ambiente de desenvolvimento utilizado foi o *Android Studio* [38] integrado com um repositório *Subversion* como implicam as boas práticas utilizadas na WIT Software que implicam que todos os projectos estejam integrados com um sistema de controlo de versões.

Todas estas ferramentas permitiram assim acelerar o desenvolvimento das aplicações.

5.5. Metodologia de Desenvolvimento

No desenvolvimento destas aplicações destinadas a demonstração foi seguida uma metodologia de prototipagem com *feedback* constante por parte do orientador, identificando aspectos a melhorar ou a explorar. Todas as aplicações desenvolvidas seguiram, portanto, a mesma linha de desenvolvimento sendo implementadas iterativamente e reutilizando aquilo que foi produzido das mais antigas para as mais recentes. A Figura 11 representa as várias iterações de desenvolvimento, nas iterações iniciais surgiram as aplicações de demonstração mais simples, sendo que a primeira iteração foi o desenvolvimento da aplicação utilizada nos testes das várias soluções de *audio watermarking* (referidos no capítulo "4.2 - Testes Realizados às Soluções Existentes"), tendo evoluído até às aplicações com uma interface gráfica mais cuidada e trabalhada.

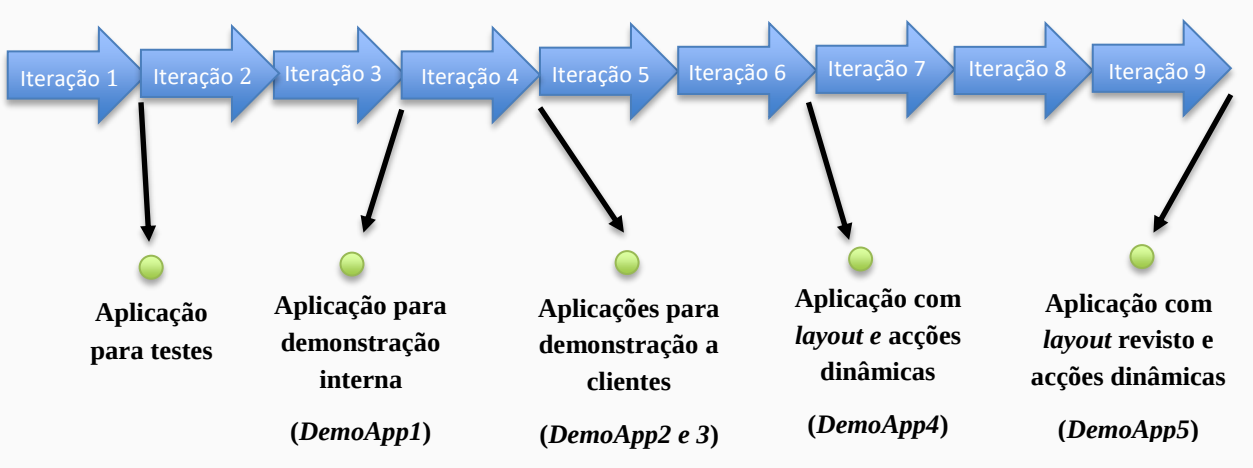


Figura 11 - Iterações de desenvolvimento das aplicações de demonstração

5.6. Conclusão

Através destas aplicações de demonstração foi possível apresentar a tecnologia internamente, averiguar o interesse e obter *feedback* dos clientes e ainda perceber como é que a detecção dos *watermarks* se comportava em ambientes menos favoráveis tal como foi o caso da feira *MWC*. Dado o ruído excessivo existente nessa feira, as aplicações de demonstração não funcionaram completamente bem. No entanto, fora dessa feira, houve *feedback* bastante positivo de potenciais clientes principalmente no que diz respeito à utilização desta tecnologia para fins de publicidade. Considera-se que o desenvolvimento de todas estas pequenas aplicações com intuitos diferentes permitem explorar e provar conceitos que ajudarão na utilização desta tecnologia no desenvolvimento de produtos direccionados para os mais diversos mercados. Principalmente porque permitiu desde cedo colocar soluções funcionais nas mãos de possíveis utilizadores/clientes, aspecto este que é de grande relevo na área de gestão de produtos visto que permite validar a visão do produto ou identificar a necessidade de pivotar [39] essa mesma visão noutra direcção.

6. WP 300: Especificação do Software a Implementar

Este capítulo descreve as tarefas de especificação e desenho da arquitectura de um *backoffice web* para criação de campanhas utilizando as *watermarks*.

Inicialmente foi previsto e especificado o *backoffice* tendo em vista o desenvolvimento de ambas as vertentes de *frontend* e *backend* ainda que, mais tarde, como explica este capítulo, decidiu-se desenvolver apenas no *backend*.

Pretendeu-se também considerar a integração deste *backoffice* num novo produto que estava a ser desenvolvido na WIT Software, tendo também em conta uma potencial evolução para se tornar um produto independente *business-to-business*.

6.1. Especificação

Apresentam-se aqui alguns detalhes em relação à especificação do *backoffice* começando pela descrição dos objectivos a que este sistema se propunha.

6.1.1. Descrição do Sistema

O que se pretendia com este *backoffice* era ter uma plataforma *web* onde os utilizadores pudessem colocar *watermarks* nos seus vídeos ou música e, quando esses *watermarks* fossem detectados por uma aplicação *mobile*, apareceria um ecrã criado de forma completamente personalizável através da mesma plataforma *web*. Por exemplo, um utilizador tal como um *Markeeter* poderia criar uma campanha submetendo um vídeo para publicitar o lançamento de um novo livro ao qual faria corresponder um ecrã com uma imagem da capa do livro e um botão que permitia comprá-lo imediatamente. Ou seja, quando o utilizador de um *smartphone* detectasse os *watermarks* no vídeo publicitário, seria redireccionado para um ecrã que apresentava a imagem e botão tal como havia sido definido aquando da criação dessa campanha. Ou seja, o objectivo principal do *backoffice* consistiu em enriquecer campanhas publicitárias, apresentando conteúdo relacionado, completamente personalizável, no *smartphone* dos utilizadores. Além disto eram pretendidos também requisitos mais específicos: as campanhas poderiam ter restrições de localização, de visualizações ou de data.

O fluxo de provisionamento de campanhas pode ser consultado no “Anexo B - Fluxo de Criação de Campanhas no Backoffice” e o fluxo de detecção de campanhas provisionadas é apresentado no “Anexo C - Fluxo de Detecção de Campanhas Provisionadas no Backoffice”.

6.1.2. Canais de Publicação

Com este *backoffice*, havia intenção de explorar vários canais de publicação diferentes. Por exemplo, os vídeos/músicas com *watermarks* poderiam ser publicados em

diversos canais tais como: na televisão, na rádio, no *YouTube*, no *Spotify*, em eventos ao vivo, em bares ou discotecas e até em lojas. Dentro desses canais, as motivações eram várias. Por exemplo, a motivação para publicar um vídeo na televisão poderia ser a de criar um vídeo publicitário que, ao ser detectado, permitisse aos utilizadores comprar imediatamente o produto apresentado. Por outro lado, ao colocar uma música ambiente (com *watermarks*) numa loja a motivação seria a de cativar a presença dos clientes nestes espaços físicos e oferecer descontos nos produtos da loja aos que detectassem os *watermarks*. Outros casos de uso foram pensados com diversas ofertas de valor para os utilizadores dentro dos vários canais onde poderiam ser publicados os conteúdos com *watermarks*. A Tabela 6 apresenta as várias acções/casos de uso (em cada linha) possíveis para os canais de publicação identificados (em cada coluna da tabela).

Tabela 6 - Casos de uso mapeados em canais de publicação

Acção / Canal de Publicação	TV	YouTube	Radio/ Spotify	Evento ao vivo	Loja	Bar Discoteca	Cinema (Pre-Movie)
Comprar produtos	Y	Y	Y				
Obter amostra de produto	Y	Y	Y		Y		
Obter promoção	Y	Y	Y	Y	Y	Y	
Ganhar pontos numa aplicação	Y	Y	Y	Y	Y	Y	Y
Jogar um quizz/concurso	Y	Y	Y	Y		Y	Y
Subscrever um serviço	Y						
Obter mais informações acerca de um produto	Y			Y	Y	Y	Y
Agendar um contacto	Y		Y				

6.1.3. Definição de Stakeholders

Os *stakeholders* identificados para este sistema foram:

- *Marketeers* e *Brand Managers*: serão os utilizadores do *backoffice* que pretenderão provisionar campanhas.
- *MediaSpot Manager*: o administrador do *backoffice*
- *End-users*: os utilizadores de *smartphones* para os quais as campanhas provisionadas são direccionadas.

6.1.4. Definição de User Roles e Stories

De forma a ter um conjunto de requisitos funcionais definidos para o *backoffice*, houve necessidade de identificar funcionalidades que, de seguida, foram mapeadas a *user roles* para segmentar o acesso a funcionalidades por parte de vários tipos de utilizador. A documentação da especificação do *software* foi feita através de identificação das tarefas, juntamente com diagramas desenhados e os *mockups*/*wireframes* de forma a ser mais expedita. A Tabela 7 apresenta as *user roles* identificadas (coluna 1) com uma breve descrição (coluna 2).

Tabela 7 - User Roles identificadas para o backoffice

User Role	Descrição
Customer | Viewer	Utilizador que apenas pode ver conteúdo associado ao seu *Customer*.
Customer | Creator	Utilizador que pode ver e criar conteúdo associado ao seu *Customer*.
Administrator	O administrador do *backoffice.*

A Tabela 8 apresenta as funcionalidades identificadas e organizadas em categorias designadas por Épico (coluna 1), descritas (na coluna 2) e mapeadas em cada uma das *user roles* (colunas 3 a 5), evidenciando as funcionalidades às quais tem acesso um certo tipo de utilizador.

Tabela 8 - Funcionalidades do backoffice mapeadas nas User Roles

Épico	Descrição	Customer | Creator	Customer | Viewer	Administrator
Autenticação	Autenticação de utilizadores / Login	●	●	●
Gestão de Campanhas	Criação/Edição de campanhas	●		
	Descarregar conteúdo *watermarked*	●	●	●
	Ver campanhas activas/arquivadas	●	●	●
	Arquivar campanhas	●		
	Editar campanhas	●		
	Suspender campanhas	●		
	Eliminar campanhas	●		
	Ver relatórios sobre campanhas	●	●	●
Gestão de Customers	Ver *Customers*			●
	Criar *Customer*			●
	Arquivar *Customer*			●

Ver utilizadores (associados a um *Customer*)	●		●
Criar utilizadores (associados a um *Customer*)	●		●
Arquivar utilizador	●		●

6.1.5. Desenho de Mockups

Para acompanhar a especificação do *software* foi necessário desenhar *mockups*, através *do* software *Balsamiq Mockups* [40], com o objectivo de dar uma base de trabalho para o designer que iria desenhar a interface do *backoffice* e também para servir de discussão na visão e âmbito deste sistema. Atendendo que a visão inicial sofreu iterações e houve necessidade de redesenhar mais de três versões diferentes. Por exemplo, inicialmente ainda não existia a ideia de integrar num outro produto da WIT Software então houve muitos detalhes que não foram tidos em conta e houve necessidade de voltar a desenhar os *mockups*. Levou algum tempo até ter a visão e âmbito final definidos, tendo o desenho dos *mockups* ajudado nessa definição. Aprendeu-se que deve-se ter sempre em conta a opinião de possíveis utilizadores deste *backoffice* para justificar as decisões tomadas no desenho dos *mockups*.

A versão final dos *mockups* pode ser consultada no anexo "Anexo D - Mockups Desenhados (versão final)" onde são apresentados todos os ecrãs desenhados (da Figura 25 à Figura 26) de acordo com a hierarquia de ecrãs da Figura 12.

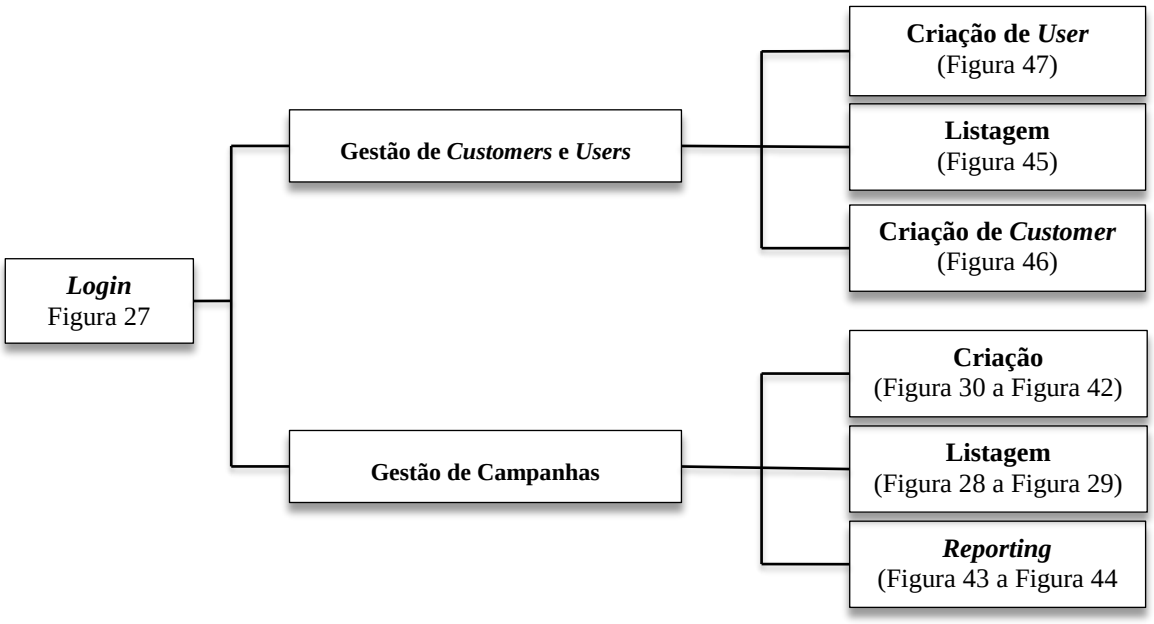


Figura 12 - Árvore de hierarquia de ecrãs do *backoffice*

6.2. Arquitectura

A arquitectura do *backoffice* foi compartimentada em três subsistemas principais, como demonstra a Figura 13. Esses subsistemas são o *frontend*, o *backend* e *file storage* & *processing* (este último para armazenamento e processamento de ficheiros áudio e vídeo).

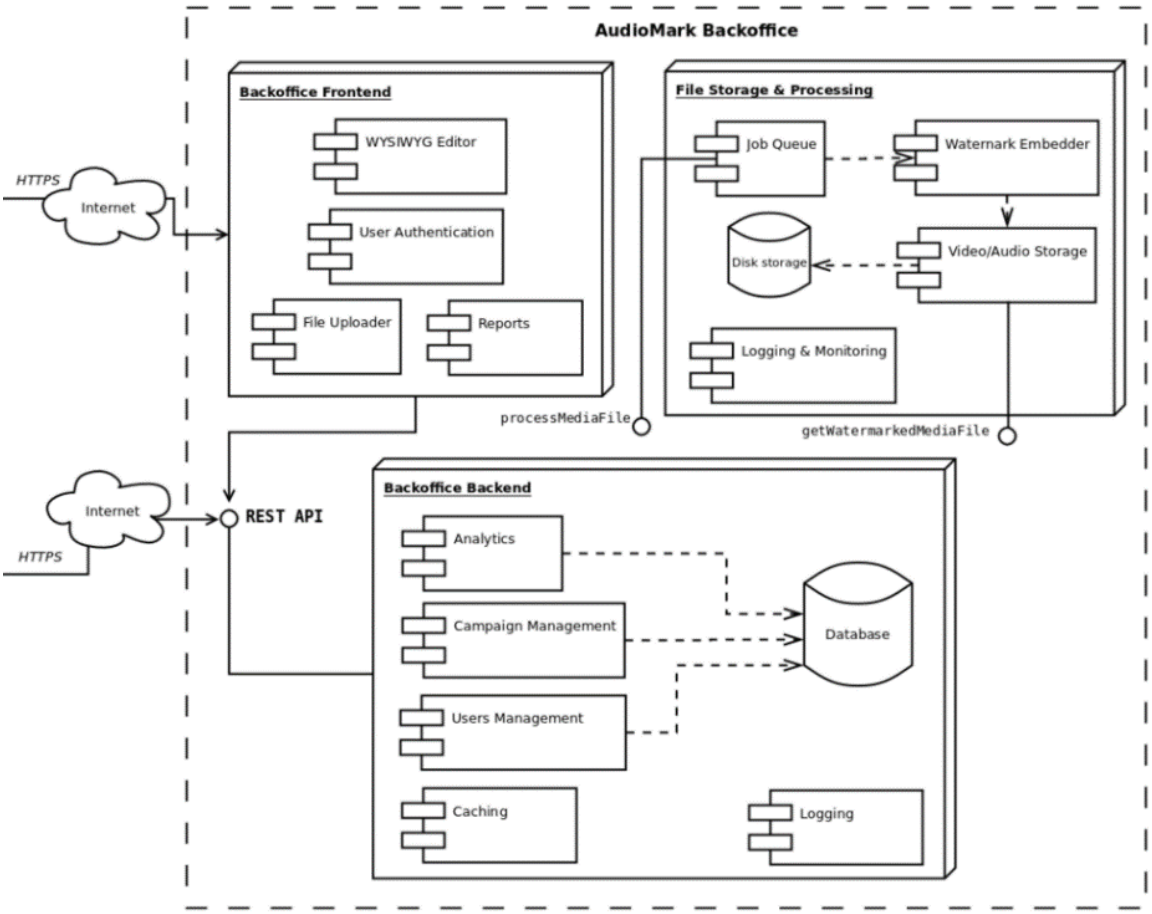


Figura 13 - Arquitectura inicial do backoffice

6.2.1. Backoffice Frontend

O *frontend*, como o nome indica é o módulo responsável pela *user interface*. Contém componentes tais como o *WYSIWYG Editor*, responsável por permitir o desenho dos *layouts* das campanhas, o *File Uploader* que permite fazer *upload* de ficheiros de áudio e vídeo (que serão processados por outro modulo), o componente *User Authentication* responsável pela autenticação dos utilizadores no *backoffice*, o componente de *Reports* que apresenta informação estatística da utilização do sistema, entre outros pequenos componentes que não foram especificados porque nesta fase não se pretendeu tal nível de detalhe.

6.2.2. Backoffice Backend

Quanto ao módulo de *backend* consiste num módulo onde constam os dados e lógica do sistema, de forma completamente separada da interface, expondo uma interface *REST* para ser consumida pelo *frontend* e também pelo cliente *mobile*. É composto por componentes de gestão de campanhas e utilizadores, bem como por componentes de obtenção de dados para gerar os *reports* no *frontend* e também componentes comuns a qualquer sistema *web* que permitem fazer *caching* dos dados ou até gerir *logs* para monitorização do próprio sistema.

6.2.3. File Storage & Processing

Este módulo é responsável apenas por processar as *watermarks* em ficheiros de vídeo e áudio e armazena-los, disponibilizando-os para o *backoffice*. Apresenta necessidades especiais de espaço em disco e capacidade de processamento sendo que está previsto a inclusão de um componente *Job Queue* para não permitir que sejam processados demasiados ficheiros em simultâneo (acabaria por saturar toda a capacidade de processamento). Inclui a ferramenta proprietária da *Cifrasoft* para injectar os *watermarks* nos ficheiros e expõe uma simples interface para que sejam submetidos ficheiros para processamento e que permite também obter os ficheiros de *output* já com os *watermarks* injectados.

6.3. Aprendizagem

Paralelamente ao desenho de *mockups*, foi antecipada a aprendizagem necessária para o desenvolvimento do *backoffice*. Já existia alguma experiência passada com soluções web em *PHP*,*ASP.Net* e *JSF* (*JavaServer Faces*). No entanto nenhuma destas tecnologias é utilizada na WIT Software e, pessoalmente, existia também curiosidade em adquirir conhecimentos em tecnologias concorrentes.

Foi analisada a famosa *framewok Angular* [41] para o desenvolvimento do *frontend* recorrendo a *Javascript*. Quanto ao *backend*, exploraram-se *frameworks* para as linguagens *Javascript* e *Java*. Inicialmente foi explorada a *stack MEAN.js* [42] na qual se prevê a utilização de *NodeJS* [43] (um *runtime Javascript*), *ExpressJS* [44] (*framework web*) e *MongoDB* [45] (base de dados não relacional) para o desenvolvimento do *backend*. De seguida foi também explorada uma *stack* baseada em Java que utiliza a plataforma *Spring Boot* [46] (desenvolvida pelos autores da famosa *framework Spring*). A aprendizagem resumiu-se a leitura, configuração de ambientes de desenvolvimento e experimentação de casos práticos simples.

O facto da *framework* Angular ser bastante usada actualmente, existindo bastante documentação e suporte da comunidade, permitiu que, aliando os conhecimentos já

existentes em *HTML*, *CSS* e *Javascript*, se percebesse rapidamente como funciona e quais as suas capacidades.

Quanto à *stack MEAN.js* foi possível perceber como é que os seus vários componentes se integram recorrendo a projectos *open-source* disponíveis na plataforma *GitHub* e à extensa documentação de cada um desses componentes. À excepção de alguns conhecimentos prévios da linguagem *Javascript*, todos estes componentes eram novos, não existindo qualquer experiencia prévia. Foi interessante experimentar a base de dados *MongoDB* que faz parte desta *framework*, visto que difere um pouco das tradicionais bases de dados relacionais já conhecidas. Considera-se que a aprendizagem de conceitos básicos de todos estes componentes que são cada vez mais usados para o desenvolvimento de plataformas *web* foi uma experiência interessante e que despertou a vontade de explorar melhor o desenvolvimento de *web* recorrendo a *Javascript*.

A aprendizagem da plataforma *Spring Boot* foi também ela interessante. Percebeu-se que esta plataforma é nada mais que uma simplificação da *framework* Spring (que não é apelativa para iniciantes devido ao excesso de configuração necessária). As vantagens da utilização do *Spring Boot* em relação ao *Spring* são principalmente o facto de o *Spring Boot* assumir uma configuração *default* para os componentes utilizados, enquanto que a utilização da *Spring Framework* (individualmente) necessita que seja despendido bastante tempo a configurar um projecto básico (algo que também requer conhecimento e experiência). Além de seguir este paradigma de "*convention over configuration"*, o *Spring Boot* também torna transparente todo o processo de gestão do servidor aplicacional: quando se executa um projecto, não é necessário ter o *Tomcat* instalado e configurado, automaticamente é arrancada uma versão *standalone* do *Tomcat*. Todas as actividades de aprendizagem relacionadas com esta plataforma revelaram exactamente que as promessas de facilidade de aprendizagem e desenvolvimento rápido são notoriamente cumpridas. Ao mesmo tempo que é escondida toda a necessidade de configuração, continua a ser possível escrutinar e alterar todos os parâmetros de configuração da *framework* Spring, caso seja necessário.

Resumem-se na Tabela 9 as principais vantagens e desvantagens de cada uma das *frameworks* experimentadas com vista ao desenvolvimento do *backend*, nomeadamente as *frameworks MEAN.js* (na coluna 2) e *Spring Boot* (na coluna 3).

Tabela 9 - Comparação entre MEAN.js e Spring Boot

Tecnologia	MEAN.js	Spring Boot
Vantagens	- Muito utilizada actualmente. - Usa tecnologias *open-source*. - Aparentemente simples de usar.	- Baseia-se na *Spring Framework* que é inegavelmente uma das mais utilizadas em desenvolvimento *web* há vários anos (tem uma imensa comunidade de utilizadores). - Bastante conhecimento *in-house* acerca da Spring Framework e há também já alguns projectos a usar especificamente o *Spring Boot*. - *Open-source*. - Aparentemente simples de usar.
Desvantagens	- Tecnologia nova para o estagiário.	- Tecnologia nova para o estagiário.

A escolha da tecnologia a usar no *backend* acabou por pender para o lado da plataforma *Spring Boot* porque utiliza tecnologias já conhecidas *in-house*. Se fosse necessário, outros colegas poderiam dar indicações, recomendações e ajuda na solução de possíveis problemas.

De forma a aprofundar o conhecimento e continuar a aprendizagem acerca do *Spring Boot*, decidiu-se iniciar o desenvolvimento do componente de *File storage & Processing* (que já foi identificado anteriormente) visto que, sendo um componente pequeno e com responsabilidades bem definidas, dificilmente ia sofrer alterações independentemente das mudanças no rumo do projecto. Aproveitou-se então para adiantar o desenvolvimento ao mesmo tempo que se aprendia, experimentava e aprofundava a tecnologia.

6.4. Redefinição dos Objectivos

Após a fase de desenho de *mockups* e arquitectura, percebendo que a visão e âmbito do *software* precisavam de ser ajustados tendo possivelmente um grande impacto no desenvolvimento do *frontend*, decidiu-se que iria ser apenas desenvolvido o *backend* do *backoffice*. Foi decidido também que o *backoffice* iria conciliar a detecção de campanhas

não só através de *audio watermarks* mas também através de realidade aumentada [47] sendo que este tecnologia estava a ser desenvolvida noutro estágio a decorrer paralelamente na WIT Software. No fundo pretendia-se também que, ao detectar uma imagem através da câmara de um *smartphone,* fosse apresentada informação adicional sobrepondo-se virtualmente ao ambiente real captado pela câmara.

O *backend* passou então a ser responsabilidade de mais um desenvolvedor que obrigou a um esforço adicional de colaboração e sincronização até porque o colega não trabalhava nos escritórios de Coimbra. Para acomodar a introdução deste novo componente de realidade aumentada e a remoção do *frontend,* foi revista a arquitectura do *software* a desenvolver, tendo sido também adicionado um pouco mais de detalhe para garantir que os pressupostos estavam bem assentes com o novo colega.

Além de ter sido adicionado mais detalhe no componente principal do *backend*, como já foi referido, foi destacado o *frontend* que passou a ser um sistema externo que, tal como a biblioteca *mobile,* há-de comunicar com este *backend*. Foi também adicionado o módulo de *Image Detection* da responsabilidade do outro colega. Foi isolado e partilhado um módulo de *Media File Storage* para ser utilizado por este módulo de *Image Detection* e pelo módulo de *Watermark Processing*.

Pretendeu-se ter um módulo principal denominado *Mediaspot* composto por *webservices* expostos através de uma interface *REST* e pela lógica principal, base de dados e ainda componentes de suporte à manutenção do sistema tais como monitorização, *logging* e configuração. Este módulo principal comunica com o módulo responsável unicamente pelo processamento de *watermarks* (*Watermark Processing*) que possui também *webservices* para que entidades externas possam interagir com as funcionalidades expostas numa *API*, gere as tarefas de processamento numa *job queue* e possui ainda componentes de monitorização. Da mesma forma, existe um módulo responsável unicamente pelo provisionamento de imagens (*Image Detection*) no serviço de detecção de imagens que suporta a utilização da realidade aumentada. Entre estes dois módulos existe ainda um quarto módulo (*Media File Storage*), utilizado por ambos, que permite guardar ficheiros de forma centralizada. A Figura 14 ilustra a arquitectura prevista para o *backend* após a revisão aqui descrita.

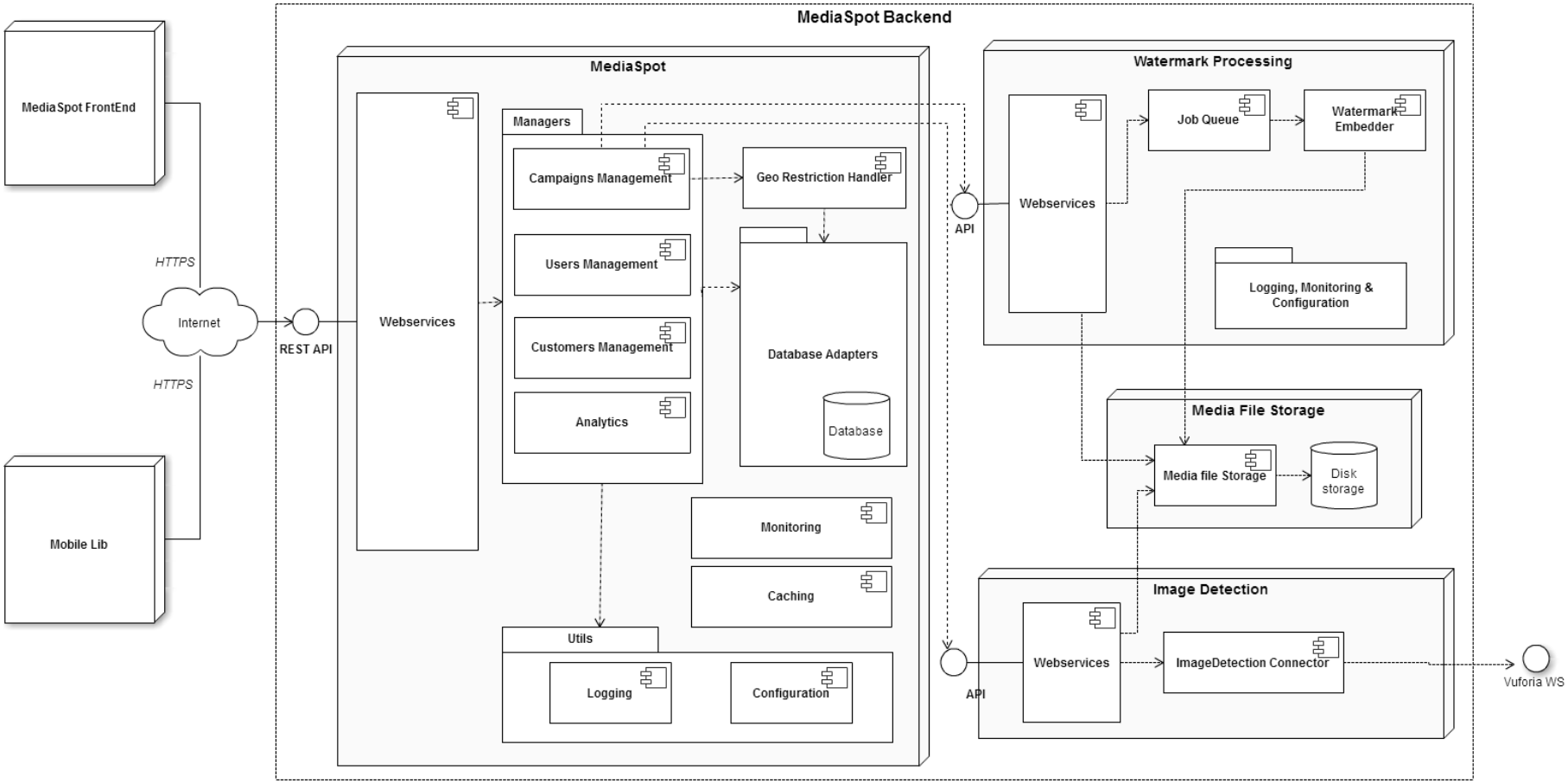


Figura 14 - Diagrama de arquitectura do *backend* revisto

6.5. Plano de Desenvolvimento

Dado que o *software* a desenvolver tinha agora um âmbito fechado, foi possível estimar e planear o seu desenvolvimento.

Inicialmente foram identificadas as tarefas, com o objectivo de serem tarefas suficientemente pequenas para facilitar a estimação da sua duração e também o acompanhamento do progresso. De seguida, foram priorizadas e divididas as duas fases: uma primeira fase com todo o *setup* do projecto e desenvolvimento de funcionalidades nucleares e uma segunda fase com funcionalidades menos prioritárias, mas que são também necessárias para completar o *software* pretendido. As tarefas identificadas e o resultado desta separação em duas *releases* diferentes podem ser consultados nas seguintes tabelas (Tabela 10 e Tabela 11) que apresentam, na primeira coluna um identificador da tarefa, na segunda coluna o "épico" que categoriza as tarefas e na terceira coluna uma pequena descrição da tarefa.

Tabela 10 - Tarefas desenvolvidas na *release* 1

Release 1		
Story ID	Epic	Story
1	Definição	
2		Definição do modelo de dados
3		Definição da api dos webservices
4	Setup do projecto	
5		setup logger
6		setup DB connection
7		setup base configs
8		setup swagger
9	Modelo de dados	
10		Implementação de entidades do modelo de dados
11		Implementação de adapters para DB
12	Web services	
13		Autenticação e Autorização
14		Retailers/Customers
15		Campaigns
16		Create
17		Retrieve
18		Update
19		Delete
20		Image/Video/Audio upload
21	Serviços 3rd Party	
22		Vuforia
23		Upload de imagem
24		Criação de targets
25		Edit target
26		Get database summary
27		Integrar na criação de campaigns
28		Watermark Embedder
29		Upload de media file
30		Integração do ffmpeg + lib da Cifrasoft
31		Processamento em job queue
32		Retrieve dos media files processados
33		Integrar na criação de campaigns
34	Actualizar Lib Android	
35		Integrar campaign retrieval com este backend
36	Testar	
37		Testar end-to-end e corrigir eventuais bugs

Tabela 11 - Tarefas desenvolvidas na *release* 2

Release 2		
Story ID	Epic	Story
1	Actualizar Web services	
2		Implementar User Management
3		Create
4		Retrieve
5		Update
6		Delete
7		Implementar Customer Management
8		Create
9		Retrieve
10		Update
11		Delete
12		Actualizar autenticação para considerar user reais
13		Actualizar criação de campaigns
14		Adicionar country restriction
15		Adicionar hit limit
16		Implementar endpoint para Analytics
17	Improvements	
18		Adicionar suporte para ETag caching
19		Implementar health check e métricas
20	Actualizar Lib Android	
21		Actualizar campaign retrieval
22		Integrar analytics
23	Testar	
24		Testar end-to-end e corrigir eventuais bugs

Tendo as tarefas identificadas, procedeu-se à estimativa do esforço necessário para a sua implementação. Para tal, foi usada principalmente a experiência obtida durante a aprendizagem das tecnologias a utilizar. A estimativa foi feita também com a colaboração do colega implementou a parte de realidade aumentada no deste *backend* e o procedimento usado foi:

1. Discussão prévia de cada uma das tarefas para garantir que havia alinhamento a nível de pressupostos;
2. Estimativa realizada individualmente, em dias e utilizando apenas números pertencentes à "Sequência de Fibonacci" [48];
3. Partilha dos valores obtidos e discussão para chegar a um consenso no caso de terem sido obtidos valores diferentes.

Foram então estimados 39 dias para a implementação da primeira *release*. Mais tarde, já durante a implementação da primeira *release* e contando com alguma experiência adicional obtida entretanto, foi seguido o mesmo procedimento que resultou em 36 dias para a segunda *release*.

Tendo a duração prevista para as tarefas, estas foram escalonadas num diagrama de *gantt* obtendo então um plano de desenvolvimento para a primeira *release*, com início a 19 de Abril e final a 25 de Maio, que foi acompanhado com reuniões semanais de forma verificar o progresso do projecto.

No anexo "Anexo E – " pode ser consultado um diagrama de *gantt* com a calendarização das tarefas para a primeira *release* (Figura 49).

6.6. Definição do Modelo de Dados

Tendo em conta os requisitos conhecidos através do desenho de *mockups*, identificação de *stories* e conhecimento técnico, foi desenhado um simples diagrama físico que demonstra o modelo de dados relacional previsto para o sistema (**Error! Reference source not found.**).

Algumas das tabelas apresentam um relacionamento de herança com uma tabela base e foi utilizado o sistema de gestão de base de dados PostgreSQL (versão 9.5.2) que suporta o conceito de herança.

Resumidamente, teve-se em conta uma tabela para Utilizadores (*user*) que são associados a *Customers* Estes Utilizadores podem criar Campanhas (*campaign*) com restrições de data de início e fim, de localização (relação com tabela *geozone*) e limite de detecções. Note-se que o *layout* da Campanha é guardado na respectiva tabela, num campo de texto com vista a armazenar a estrutura JSON semelhante à que foi definida para a aplicação de demonstração já discutida. É importante também notar que a tabela das campanhas de *audio watermarking* (*audio_watermarking_campaign*) possui um atributo para o código/*watermark* único a ser injectado nos ficheiros de áudio/vídeo e que permitirá identificar a campanha associada.

Existe ainda um conjunto de tabelas para armazenar eventos de *analytics*. O objectivo é poder extrair relatórios de visualização das campanhas sendo que a biblioteca *mobile* a desenvolver notificará o *backend* sempre que seja iniciada a detecção de *watermarks*, sempre que seja detectada/fechada uma Campanha específica e sempre que seja pressionada uma acção no *layout* de uma Campanha.

Todas as tabelas possuem atributos que indicam a data de criação e última actualização visto que a experiencia já adquirida no desenvolvimento deste tipo de soluções mostra que são campos úteis em situações de *debugging* e também para permitir *caching* dos dados no *frontend*.

A Figura 15**Error! Reference source not found.** ilustra o diagrama correspondente ao modelo de dados aqui descrito.

Figura 15 - Diagrama físico do modelo de dados do *backend*

6.7. Conclusão

Começando desde logo pela especificação do *software*, tal como descrito, houve alguns problemas a garantir que todas as partes interessadas estavam de acordo naquilo que ia ser desenvolvido. Por vezes foi complicado lidar com as mudanças de opinião constantes e sem grande fundamento mas reconhece-se que, até certo ponto, acaba por ser um comportamento natural. Fica a lição que é necessário interpretar estes sinais desde cedo para reagir e eliminar obstáculos na especificação do *software*. Depois de tanto avanço e retrocesso na especificação, preferiu-se reformular os objectivos de forma a obter mais *feedback* dos possíveis clientes antes de começar a implementar as mais importantes decisões.

7. WP 400: Implementação do Software

Após a especificação do *software*, procedeu-se então à sua implementação que é descrita neste capítulo tal como o desenvolvimento do módulo reutilizável que estava previsto no plano do estágio.

7.1. Aplicação Servidor

Descrevem-se agora alguns detalhes sobre a implementação do *backoffice*.

7.1.1. Modelo de Dados

Para implementar o modelo de dados foi utilizada *API JPA*, através da *framework Hibernate ORM* [49] (dependência esta que é automaticamente adicionada pelo *Spring Boot*). A implementação das classes referentes às entidades do modelo de dados foi simples. Recorrendo a vários exemplos de código é possível perceber quais anotações Java deviam ser usadas para implementar correctamente aquilo que havia sido desenhado. Ainda que o PostgreSQL suporte herança, percebeu-se mais tarde que o *JPA*, por ser genérico a qualquer sistema de gestão de base de dados, não suporta directamente esse conceito. Em vez disso, o *JPA* possui uma anotação para mapear a herança entre classes *Java* em três diferentes estratégias de herança: *SINGLE_TABLE*, *TABLE_PER_CLASS*, *JOINED* [50]. Evitou-se usar a estratégia *SINGLE_TABLE* dado que armazena os dados de todas as subclasses de uma entidade numa só tabela da base de dados, o que leva a que existam atributos que possam ser nulos e a uma base de dados que não estaria normalizada. Esta estratégia só seria usada caso, mais tarde, se verificassem problemas de *performance*. Posto isto, foi utilizada a estratégia *JOINED* para representar a herança relativa às *Campaigns*, permitindo que uma campanha esteja ao mesmo tempo associada à detecção por áudio e por realidade aumentada, e a estratégia *TABLE_PER_CLASS* para a herança das tabelas *Analytics* dado que se pretendeu exactamente uma tabela isolada por cada tabela que herdava da tabela principal. No “Anexo F - Diagrama de Classes – Classes do Modelo de Dados do Backend” pode ser consultado o diagrama de classes relativo às classes relacionadas com o modelo de dados.

Foram ainda criadas várias classes que espelham o padrão *Repository* [51] . No fundo servem de ponte entre a lógica de domínio e o modelo de dados, permitindo efectuar as típicas operações de inserção, edição, actualização e remoção de dados. Para isto utilizou-se a *framework Spring Data* pertencente ao ecossistema Spring que possui classes genéricas como a *JPARepository* que facilitam imenso a implementação deste padrão arquitectural e que permitem com pouco esforço funcionalidades extra, tais como *queries* paginadas que são úteis para suportar os *web services*.

A utilização do JPA permitiu uma grande vantagem a nível de rapidez de implementação.

7.1.2. Web Services

Os *web services* implementados seguiram ao máximo o padrão *REST* [52] que é praticamente o *standard* absoluto nos dias de hoje. Foram implementados com recurso à *framework Spring Web MVC* [53], facilmente integrável através do *Spring Boot*, que permite facilmente criar uma *API REST*: consistiu em anotar métodos Java com anotações específicas dessa *framework* que indicam o caminho onde um método de um *web service* pode ser acedido, tipo de pedido HTTP ao qual responde (GET, POST, etc) e parâmetros de *input* (que são mapeados nos parâmetros do método Java). Como exemplo veja-se a Figura 16 que mostra um excerto de código com as anotações necessárias para expor um *endpoint* na API:

```
@RequestMapping(value = "/pets/{petId}", method = RequestMethod.GET,
produces="application/json")
@ResponseBody
public Pet getPet(@PathVariable String petId, Model model) {
    // implementation omitted
}
```

Figura 16 - Exemplo de código que define um *endpoint* de uma *API REST* através da *framework Spring Web MVC*

A documentação dos *web services* foi gerada com auxílio da *framework Swagger* [54] que permite, também através de anotações, especificar no código a descrição dos métodos, seus parâmetros e possíveis códigos de retorno. Através dessas anotações, o *Swagger* gera uma página *web* com a especificação dos métodos, organizada e bem formatada. Essa página permite , de uma forma fácil, testar os métodos da *API* manualmente (tal como se pode confirmar no exemplo oficial [55]). As anotações inseridas passaram a servir então para três efeitos: comentários ao código, descrição dos métodos dos *web services* e *suite* de teste destes mesmos métodos. Na Figura 17 é possível ver um *screenshot* de uma parte da página *web* gerada para um dos *web services* desenvolvidos (nomeadamente o *web service* dedicado à entidade *Customer*).

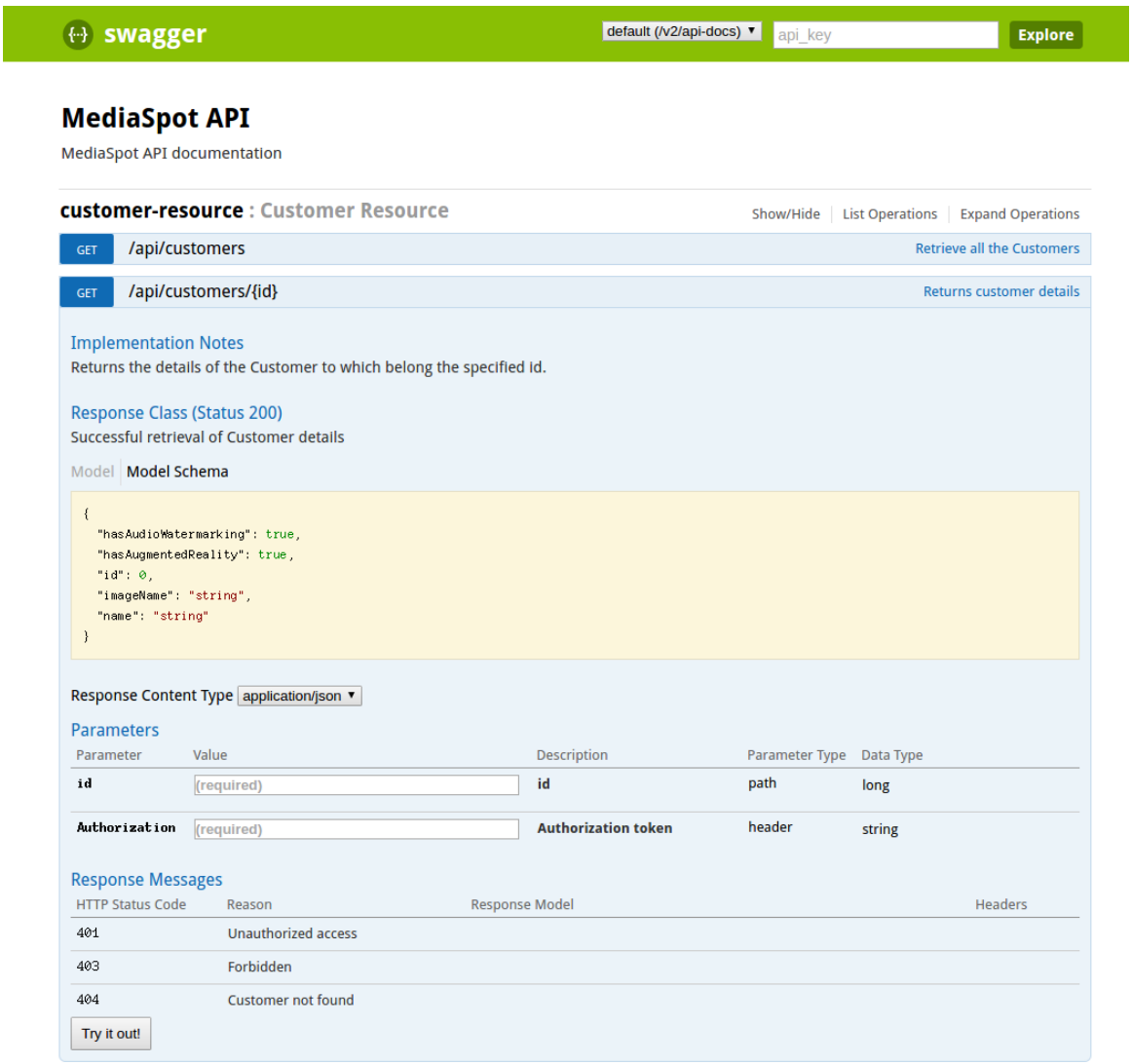


Figura 17- Exemplo de suite gerada pelo *Swagger* para um *web service*

Mais uma vez, a utilização destas *frameworks* requereu alguma aprendizagem e trouxe imensas vantagens e rapidez de desenvolvimento.

7.1.3. Autenticação

Como sempre no desenvolvimento de um *backend* é necessário ter em conta questões de segurança tais como a autenticação e autorização dos pedidos. Neste caso, a autenticação foi implementada através de um *web service* que responde a pedidos de autenticação compostos por um *username* e respectiva *password*. A resposta inclui um *token JWT* [56] gerado com auxílio da biblioteca *JJWT* [57]. Assim, quando é realizado um pedido a *web services* que necessitem de autenticação é necessário que este *token* seja indicado no *header* do pedido sendo validado e verificada a sua data de expiração, permitindo que seja identificado o utilizador que está a efectuar o pedido e validar se tem de facto autorização para aceder ao *web service*.

Foram incluídos alguns dados do utilizador codificados no próprio *token*, para evitar fazer *queries* à base de dados, a cada pedido recebido no *web service*, visto que esses dados eram usados em grande parte dos métodos do *web service*.

Mais tarde percebeu-se que, no caso em que fossem alterados esses dados na base de dados, seria necessário invalidar *tokens* de forma a obrigar a actualização dos dados neles contidos. Caso contrário corria-se o risco de ocorrerem casos de erro porque os dados contidos no *token* já não estavam coerentes com os dados que estavam realmente guardados na base de dados. Para tal, foi necessário implementar um mecanismo para rejeitar *tokens* com dados antigos. Com o objectivo de evitar, mais uma vez, a necessidade de fazer *queries* à base de dados para verificar se os dados contidos no *token* estavam actualizados, preferiu-se incluir também no próprio *token* o *timestamp* que indica o momento em que foi criado. Passou-se também a registar numa estrutura em memória o *timestamp* referente ao momento em que um determinado utilizador foi actualizado. Desta forma, todos os pedidos que necessitam de autorização, passam por um filtro que verifica, com um *overhead* reduzido, se houve alterações aos dados do utilizador desde que o *token* foi criado. Em caso afirmativo, o pedido é rejeitado obrigando a que o utilizador faça *login* novamente.

7.1.4. Conclusão

Como referido anteriormente, quanto à aplicação servidor foi apenas implementado um *backend* que ainda assim exigiu alguma análise e aprendizagem quanto às tecnologias a utilizar. Sabe-se que este *backend* não estará completamente fechado enquanto não houver uma decisão definitiva em relação à especificação do *frontend* que o integrará, ainda assim considera-se que foi implementada uma boa base para desenvolvimentos futuros.

O facto de ter existido a colaboração de outro estagiário foi interessante porque trouxe uma nova dimensão ao projecto. Existiu uma grande necessidade de comunicar frequentemente (e remotamente), de partilhar ideias e de rever o trabalho produzido. Isto traduziu-se na partilha de conhecimentos constante e numa motivação extra para trabalhar.

7.2. Desenvolvimento de uma Biblioteca

Fazia ainda parte do plano deste estágio, desenvolver uma biblioteca reutilizável em aplicações Android. Serve este capítulo para descrever o seu desenvolvimento.

7.2.1. Descrição da Biblioteca

O objectivo desta biblioteca é fazer a ligação ao *backend*, obter as campanhas provisionadas, e mostrar um ecrã de apresentação da campanha. Acontece que no desenvolvimento de aplicações de demonstração houve desde logo a preocupação de modularizar componentes permitindo que se tenha gerado uma biblioteca distribuível praticamente sem esforço nenhum. Pegando no trabalho realizado anteriormente, como referido na secção "5.3 - Provas de Conceito Focadas na Interface", isolou-se apenas o código que interessava reutilizar ficando apenas com quatro componentes principais, como mostra a Figura 18:

- *Campaign Retriever*: entidade responsável pela obtenção de campanhas comunicando com o servidor.
- *Campaign Manager*: entidade responsável por armazenar as campanhas.
- *Campaign Presenter*: entidade responsável por apresentar as campanhas visualmente.
- *Watermak Detector*: entidade responsável por detectar *watemarks* sonoros através do microfone.

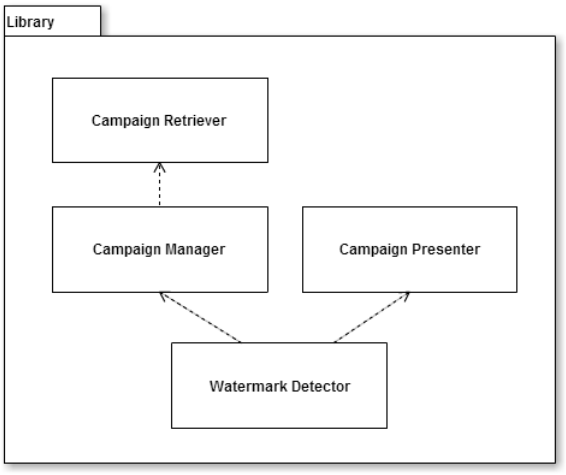


Figura 18- Arquitectura simplificada da biblioteca Android desenvolvida

Na Figura 19 é apresentado o diagrama de sequência que demonstra um exemplo da comunicação entre os vários componentes da biblioteca.

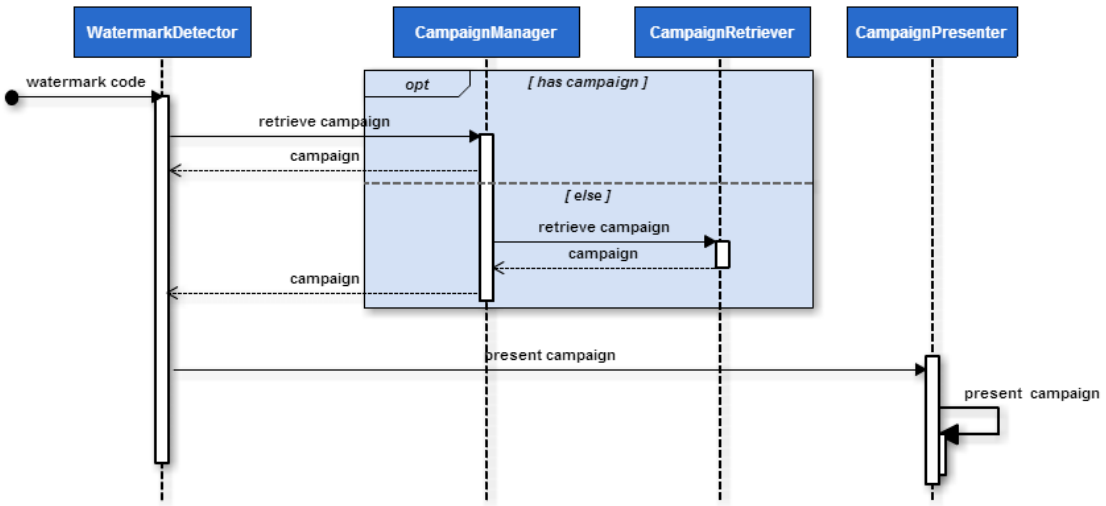


Figura 19 - Diagrama de sequência da comunicação entre os componentes da biblioteca

Esta biblioteca foi desenhada para que todos os componentes que fazem parte dela sejam reutilizados num futuro produto da WIT Software. O mesmo colega que acompanhou o desenvolvimento do *backend* pode desde logo reutilizar todos os componentes para integrar a tecnologia de reconhecimento de imagens para a detecção de campanhas provisionadas no *backend*.

7.2.2. Detalhes de Implementação e Reutilização

Sendo já conhecido o objectivo desta biblioteca e funcionamento pretendido, descrevem-se agora alguns detalhes de implementação e indicações para reutilização. O diagrama ilustrado na Figura 20 permite perceber melhor o funcionamento interno de cada componente que pertence à biblioteca. Os componentes desenvolvidos por terceiros estão coloridos a azul.

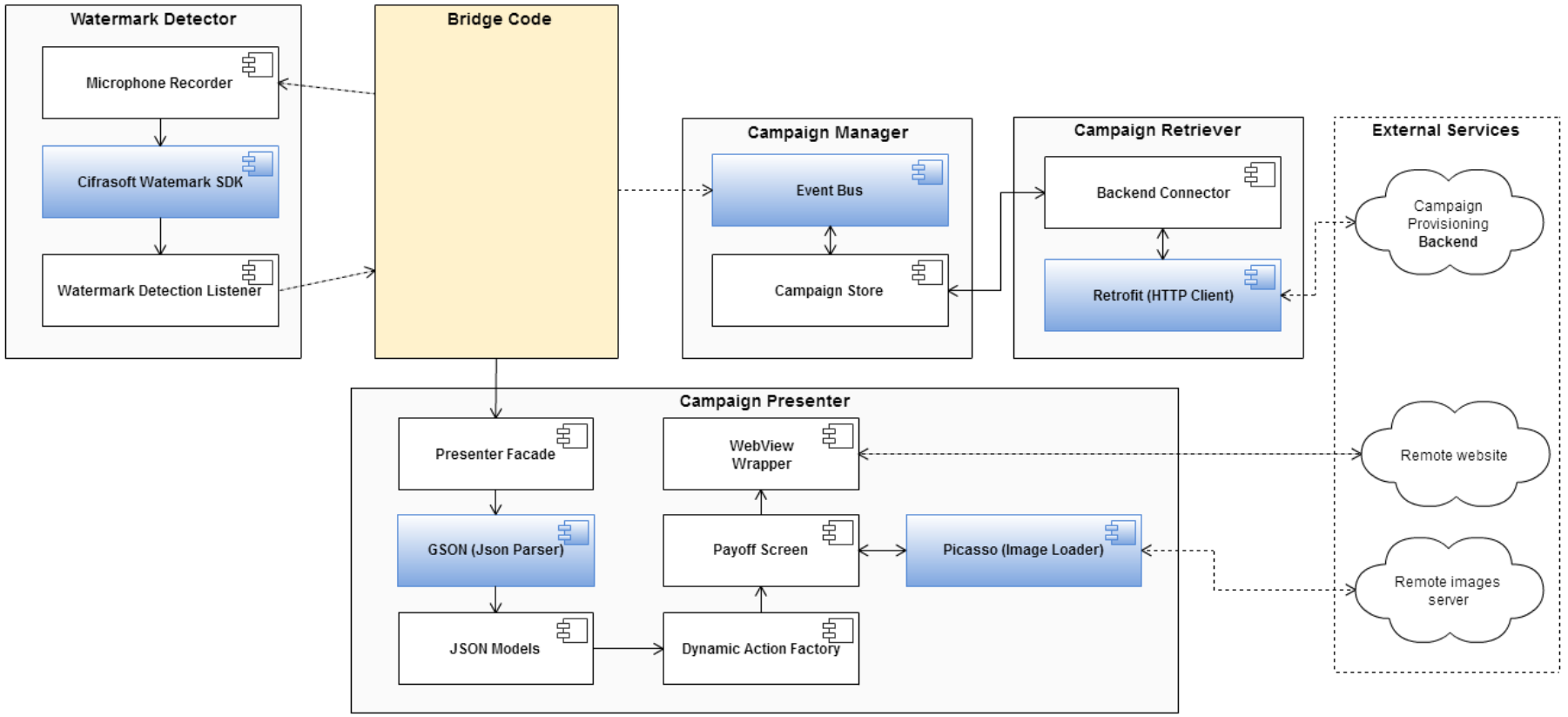


Figura 20- Diagrama detalhado que fluxo de comunicação entre os vários componentes da biblioteca

O componente *Watermark Detector* é no fundo o modulo descrito na secção "5.2 - Arquitectura do Módulo de Detecção de Watermarks", todos os outros componentes já foram descritos, de forma sucinta, na secção anterior,

Destaque-se o componente denominado *Bridge Code*. Este componente na realidade não existe na biblioteca e deve ser implementado pelo programador que a reutilize para: activar a detecção de *watermarks*, reagir à detecção de um *watermark* específico, despoletar o carregamento de campanhas do servidor remoto e apresentar o ecrã desenhado especificamente para a visualização de campanhas. Preferiu-se deixar este componente ao critério do programador porque, caso contrário, estaríamos a forçar as dependências entre módulos. Assim, a biblioteca pode ser adaptada flexivelmente a outros casos de uso. Por exemplo, mais tarde pode decidir-se reutilizar apenas o componente de detecção de *watermarks* noutro contexto ou pode ser substituído todo o componente de apresentação de campanhas sem qualquer problema. Desta forma, fica também ao critério de quem reutilizar a biblioteca decidir qual o melhor momento para obter as campanhas do *backend* sem estar obrigado a uma lógica predefinida e forçada para tal.

Esta aproximação permitiu também que o colega que acompanhou o desenvolvimento do *backend*, como já foi referido, tenha podido detectar campanhas através de reconhecimento de imagens não estando obrigado a que a detecção seja feita por *audio watermarks*.

No cenário a que se propõe este estágio, o *Bridge Code* é responsável por obter todas as campanhas no arranque de uma aplicação e activar a detecção por *watermarks*. Depois, reagindo às detecções, verifica se o código do *watermark* corresponde a uma das campanhas obtidas. Em caso afirmativo, inicia a criação do *layout* de forma dinâmica para a respectiva campanha e apresenta visualmente o ecrã com o layout que foi construído. Caso esse *layout* possua imagens, as mesmas são carregadas assincronamente. Caso possua botões que redireccionam o utilizador para uma página *web*, essa mesma página é apresentada sem que seja necessário sair da aplicação para um *browser* externo.

7.2.3. Conclusão

Como descrito neste capítulo, o esforço dedicado à produção da biblioteca acabou por ser reduzido dado que todo o trabalho realizado até então já previa a necessidade de reutilização e modularização. Está prevista a sua integração brevemente num produto novo que está actualmente a ser desenvolvido na WIT Software.

8. WP 500: Testes e Correcções ao Software

Visto que, hoje em dia, cada vez mais se vê o desenvolvimento de testes automatizados como uma vantagem na qualidade do *software* produzido, a aprendizagem e desenvolvimento do *backend* não estariam completos sem a experimentação das ferramentas de testes indicadas para as plataformas/*frameworks* utilizadas.

8.1. Testes Automatizados

Os testes automatizados foram considerados desde o início da implementação do *backend* e, por esse motivo, cada componente implementado foi acompanhado por um conjunto de testes automatizados.

8.1.1. Critério

Não se pretendeu que os testes fossem exaustivos, tendo-se seguido uma lógica de *"just enough"*. Os testes foram aplicados apenas à interface pública do *backend*, nomeadamente os *web services* desenvolvidos. Pretendeu-se aplicar um conjunto de testes correspondentes a *smoke tests*, que garantem o funcionamento do *software*, substituindo desta forma a necessidade de os executar manualmente. Para tal, garantiu-se que os testes cobriam as classes que definem os *web services* com um critério de cobertura próximo de 100% *node coverage* [58], garantindo assim que foram testados todos *result codes* esperados para vários *inputs* dos *web services*. Além disso, para todos os defeitos encontrados por experimentação manual, também foram implementados testes que reproduziam esses mesmos defeitos acompanhando a sua correcção.

8.1.2. Tecnologia

Foi utilizada a *framework Spring MVC Test* que pertence ao universo *Spring* sendo garantia de compatibilidade com o que estava a ser desenvolvido. A *framework* oferece vários *mocks* para todos os mecanismos que gerem os pedidos *HTTP* e permite correr testes que envolvem estes mesmo pedidos sem que seja necessário executar efectivamente um serviço *HTTP*, tudo isto integrado com o *jUnit*.

8.1.3. Conclusão

Ao todo foram desenvolvidos exactamente 99 testes cujo tempo de execução total é de apenas 1 minuto e 38 segundos. Estes testes resultam em 84% *node coverage* para as classes que definem os *web services* e que foram identificadas como sendo o principal alvo dos testes. Indirectamente, cobrem também 76% da totalidade das linhas de código de todas classes do *backend*. Alguns dos resultados são apresentados no anexo "Anexo G – Resultados dos Testes de Integração Automatizados". Nesse anexo, a Figura 52 apresenta

o tempo de execução de cada teste e a Figura 53 apresenta um relatório de cobertura por *package*.

A Tabela 12 resume as classes de teste desenvolvidas (coluna 1), o seu tempo de execução (coluna 2) e quantidade de testes nelas contidas (coluna 3). Na última linha são apresentados os valores totais.

Tabela 12 - Listagem de testes implementados e o seu tempo de execução

Classe	Tempo de Execução	Quantidade de testes
CSVImporterTests	33s 574ms	1
CampaignControllerTest	49ms	1
DatabaseTests	732ms	4
UserTest	58ms	1
AnalyticsActionPressedEventResourceIntTest	5s 751ms	10
AnalyticsCampaignEventResourceIntTest	1s 273ms	3
AnalyticsDetectionEventResourceIntTest	9s 853ms	11
AuthenticationResourceIntTest	1s 451ms	2
CampaignResourceActiveIntTest	2s 147ms	3
CampaignResourceIntTest	9s 533ms	12
CampaignResourceUpdateIntTest	4s 538ms	5
CampaignResourceUploadIntTest	2s 436ms	5
CustomerResourceCreateIntTest	404ms	1
CustomerResourceDeleteIntTest	934ms	2
CustomerResourceIntTest	2s 494ms	3
CustomerResourceUpdateIntTest	1s 11ms	3
CustomerResourceUploadIntTest	1s 56ms	3
UserResourceCreateIntTest	6s 616ms	6
UserResrouceDeactivateIntTest	4s 75ms	4
UserResourceRetrieveIntTest	7s 938ms	8
UserResourceUpdateIntTest	8s 614ms	11
Total	*1m 38s 537ms*	*99*

A Figura 21 apresenta os resultados de *code coverage* relativos apenas ao *package* onde constam as classes que definem os *web services* que, como já foi referido, foi o principal alvo dos testes implementados. São apresentados valores de cobertura por classe, por métodos e por linhas de código.

Coverage Summary for Package: com.witsoftware.mediaspotbackend.web.rest

Package	Class, %	Method, %	Line, %
com.witsoftware.mediaspotbackend.web.rest	88.9% (8/ 9)	94.9% (37/ 39)	84.3% (291/ 345)

Class	Class, %	Method, %	Line, %
AnalyticsActionPressedEventsResource	100% (1/ 1)	100% (3/ 3)	100% (14/ 14)
AnalyticsCampaignEventsResource	100% (1/ 1)	100% (3/ 3)	68.2% (15/ 22)
AnalyticsDetectionEventsResource	100% (1/ 1)	100% (3/ 3)	100% (12/ 12)
AuthenticationResource	100% (1/ 1)	100% (2/ 2)	100% (9/ 9)
CampaignResource	100% (1/ 1)	90% (9/ 10)	81.8% (112/ 137)
CountryResource	100% (1/ 1)	100% (2/ 2)	100% (7/ 7)
CustomerResource	100% (1/ 1)	100% (8/ 8)	84.9% (62/ 73)
Mappings	0% (0/ 1)	0% (0/ 1)	0% (0/ 1)
UserResource	100% (1/ 1)	100% (7/ 7)	85.7% (60/ 70)

Figura 21- Sumário dos resultados de *test coverage* para as classes que definem os *web services*

O conjunto de testes implementados serviu desde logo para garantir que não foram introduzidos defeitos durante as tarefas da segunda *release* afectando as funcionalidades que já faziam parte da primeira *release*. Não foi necessário perder tempo voltar a testar as funcionalidades manualmente como aconteceria caso não tivessem sido criados testes automatizados.

Pode-se então concluir que a automatização de testes foi uma mais-valia por ter contribuído para a qualidade do *software* e aumento da confiança no trabalho realizado. Conclui-se também que o tempo "perdido" a implementar estes mesmo testes foi amortizado por não ter sido necessário voltar a testar manualmente todas as funcionalidades em diversos momentos.

8.2. Testes de Carga

Os testes de carga também foram considerados para aferir a capacidade de resposta do *software* implementado.

8.2.1. Caso de Teste

Para realizar os testes de carga foram executados, de forma automatizada, os seguintes grupos de pedidos ao servidor onde o *software* estava disponível:

- Grupo 1:
  - Obtenção de campanhas activas;

- Obtenção de uma campanha através do seu “id”.

- Grupo 2:
  - Envio de um evento de início de detecção;
  - Envio de um evento de campanha detectada;
  - Envio de um evento de acção pressionada;
  - Envio de um evento de finalização da detecção.

Os testes consistiram em executar estes pedidos continuamente durante 2 minutos através de 100 *threads* em simultâneo. Cada teste foi configurado para fixar um valor específico de *throughput* (pedidos por segundo) que foi distribuído em 50% para cada grupo de pedidos.

A Tabela 13 apresenta as características mais relevantes da máquina onde foi instalado o *backend*.

Tabela 13 - Características da máquina de testes

Características da Máquina de Testes	
CPU	Intel Core i5 2.10 GHz (4 cores)
Memória	8 GB
Sistema Operativo	Ubuntu 14.04
Servidor Web	Tomcat v8.0.32
Base de Dados	Postgres v9.5.3

8.2.2. Tecnologia

Para automatizar estes testes foi utilizada a ferramenta *Apache JMeter* [59] que permite, entre outras coisas, fazer testes de carga através de pedidos *HTTP*. Esta ferramenta gera ainda pequenos relatórios com dados relevantes, tais como o tempo médio de resposta a um certo pedido.

8.2.3. Conclusão

Os resultados obtidos são apresentados no “Anexo H – Resultados dos Testes de Carga”. A Figura 22 resume os resultados de *throughput* real obtidos e o tempo de resposta médio dos pedidos realizados em cada teste.

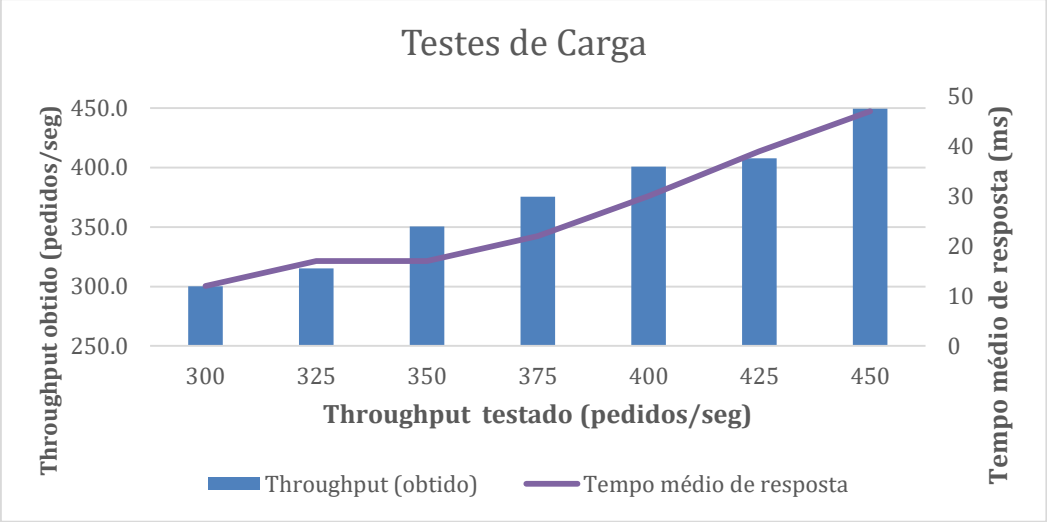


Figura 22- Evolução do throughput real e do tempo de resposta médio

No eixo dos x são apresentados os vários valores de *throughput* que foram testados. No eixo dos y, à esquerda, são apresentados os valores de *throughput* obtidos e, à direita, é mostrado o tempo de resposta médio em milissegundos. No gráfico é traçada uma linha que indica o tempo médio de resposta à medida que o *throughput* testado é aumentado e são apresentadas barras que indicam o valor real de *throughput* obtido.

Como era de esperar, assistiu-se a um aumento no tempo médio de resposta à medida que o *throughput* testado foi sendo aumentado. Ainda assim, no pior cenário testado, os valores mantiveram-se sempre baixos (abaixo dos 50 milissegundos).

A Figura 23 apresenta a utilização de capacidade de processamento em cada teste.

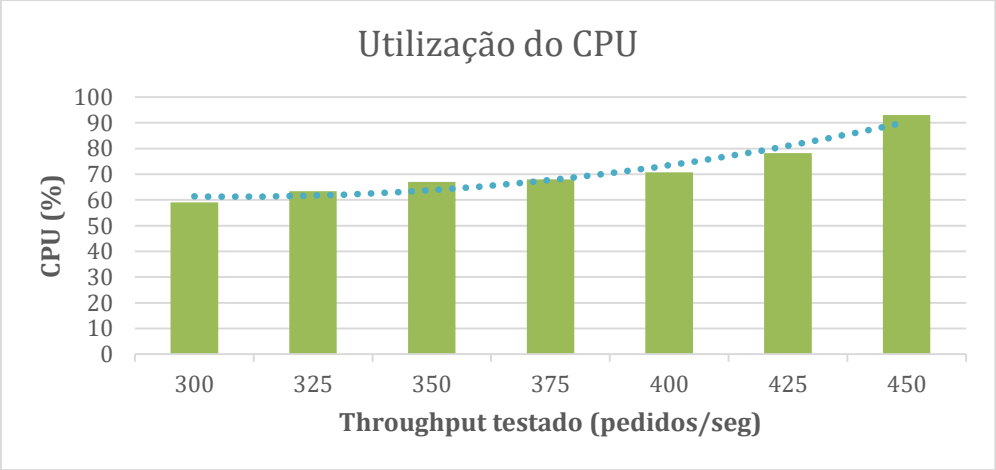


Figura 23 - Evolução da utilização do *CPU*

No eixo dos x é apresentado o *throughput* testado e no eixos do y é apresentada a utilização de *CPU*. Ou seja, as barras presentes no gráfico indicam a utilização média de

CPU para cada valor de *throughput* testado. A linha traçada é meramente indicatória da tendência da evolução da utilização de *CPU*.

É possível verificar que a partir dos 425 pedidos por segundo, a utilização de *CPU* aproxima-se dos 80 porcento. Pode-se concluir que este será o valor máximo aceitável, considerando que é prejudicial para o *CPU* manter-se com uma carga superior aos 80 porcento e tendo também em conta que será necessário reservar algum processamento para serviços de administração da máquina tal como acesso via *SSH*.

A Figura 24 mostra a evolução do mínimo de memória *RAM* livre em cada teste.

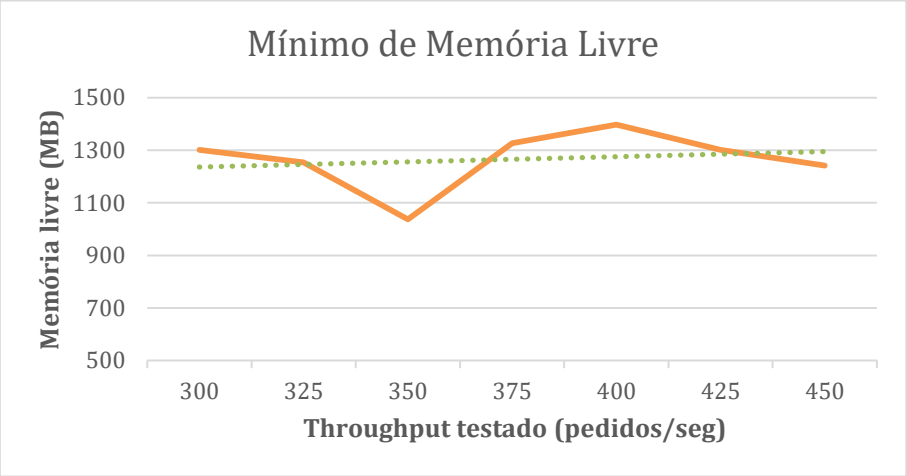


Figura 24- Evolução do mínimo de memória *RAM* livre

Dado que não se assistiu a um claro aumento no consumo de memória, calcula-se que não exista nenhum *memory leak* grave.

No geral, obtiveram-se bons indicadores de como o *backend* se comportará num cenário exigente.

9. Conclusões e Trabalho Futuro

No decorrer deste estágio realizado na empresa WIT Software, foi abordada a tecnologia de *audio watermarking* tendo sido analisadas duas aproximações diferentes: a aproximação inaudível e a aproximação audível imperceptível. Foi possível entender as principais motivações para o uso desta tecnologia e encontrar soluções para o seu uso.

Foram testadas diversas soluções que recorrem a estas aproximações, com o objectivo de perceber se eram suficientemente robustas para aplicar em novos produtos desenvolvidos pela WIT, tendo sido identificada a solução da empresa Cifrasoft como a mais robusta tendo em conta a sua *performance* em diferentes ambientes (silenciosos e ruidosos) e em diferentes *smartphones*.

De seguida, desenvolveram-se protótipos de aplicações *mobile* para efeitos de análise da tecnologia e para utilização na sua demonstração, o que permitiu apresentar a tecnologia internamente, averiguar o interesse e obter *feedback* dos clientes e ainda perceber melhor como é que a detecção dos *watermarks* se comportava em ambientes menos favoráveis tal como foi o caso da feira MWC. Considera-se que o desenvolvimento de todas estas pequenas aplicações com diferentes objectivos permitiu explorar, provar e validar conceitos que ajudarão na utilização desta tecnologia no desenvolvimento de novos produtos na WIT.

O desenvolvimento de um *backoffice*, para suportar a utilização da tecnologia de *audio watermarking* em aplicações *mobile*, fazia também parte do plano do estágio. Foi necessário especificar este *backoffice* e isso levou, à reformulação dos objectivos e dos requisitos para o desenvolvimento do *backoffice*, de forma a obter mais *feedback* dos possíveis clientes antes de começar a implementar os requisitos baseados nas mais importantes decisões. Por este motivo, deu-se prioridade à implementação da componente de *backend* dado que esta menos susceptível a alterações no âmbito do *backoffice*.

Após a especificação do *backoffice*, partiu-se para uma pequena análise sobre as *frameworks MEAN.js* e *Spring Boot*, de modo a ser feita uma escolha ponderada e fundamentada quanto à tecnologia a utilizar. Optou-se pela *framework Spring Boot* e uma das razões também foi porque já existia bastante conhecimento *in-house* sobre a mesma, apesar de não ser conhecida pelo estagiário, tendo sido, por isso, necessária a sua aprendizagem que se revelou bastante proveitosa na sua posterior utilização pelo estagiário. A *framework Spring Boot* mostrou imensas vantagens destacando-se principalmente o facto de acelerar o desenvolvimento de aplicações. comparando a qualquer outra *framework* já conhecida. A integração do *Swagger* para gerar facilmente documentação da interface pública do *backend* também mostrou ser vantajosa, e passou também a ser utilizado imediatamente noutros projectos da WIT, dado que existe uma relação bastante positiva entre o esforço de integração e as vantagens obtidas.

A implementação do *backend* correu de acordo com o plano elaborado e o escalonamento previsto foi cumprido. A automatização de testes foi uma mais-valia por ter contribuído para a qualidade do *software* e para o aumento da confiança no trabalho realizado. O tempo utilizado a implementar estes testes foi compensado por não ter sido necessário voltar a testar manualmente todas as funcionalidades em diversos momentos. Outro factor que contribuiu para o aumento de confiança no trabalho realizado deveu-se à realização de testes de carga onde se obtiveram bons indicadores de como o *backend* se comportará num cenário exigente. Pode-se concluir que as preocupações quanto à qualidade do trabalho produzido, através da realização de diversos testes em diferentes fases do estágio, revelaram-se uma mais-valia, porque não só contribuíram para a qualidade geral do *software* desenvolvido, indo ao encontro da cultura de qualidade que é fomentada na WIT, como também serviram para aumentar a confiança no trabalho realizado.

A destacar que esta tecnologia é viável, interessante e incomum, mas por si só, não será atractiva para os utilizadores. O *software* em que seja incluída deverá oferecer uma proposição de valor aos utilizadores que vá além da utilização forçada desta tecnologia.

Considera-se que todo o trabalho desenvolvido terá relevância para actividade futura da WIT Software, visto que vai ser incluída em produtos futuros. Em particular, os protótipos desenvolvidos foram desde logo utilizados para demonstrações a possíveis clientes e em feiras internacionais.

A realização de todas estas tarefas permitiu obter uma solução personalizada e facilmente reutilizável, não sendo conhecida nenhuma igual no mercado. Como sugestão de trabalho futuro, o estagiário sugere a implementação de um módulo reutilizável também para o sistema iOS semelhante ao que foi aqui desenvolvido para Android pelo estagiário. É igualmente importante testar a solução desenvolvida diversificando mais os *smartphones* utilizados na detecção dos *watermarks* e também os dispositivos utilizados na sua emissão, de forma a aferir ainda com maior detalhe a robustez da tecnologia.

Como conclusão final, considera-se que o estágio foi uma experiência excelente pela diversidade das tarefas realizadas em ambiente profissional e pelo rigor e empenho que foi necessário colocar na sua realização.

Anexos

Anexo A - Resultados dos Testes Realizados às Várias Soluções de Watermarking

Tabela 14 - Resultados dos testes às soluções de watermarking

			NOISY ENVIRONMENT			SILENT ENVIRONMENT		
SDK	Watermark Volume	EXPECTED	SGS3	SGNexus	Xiaomi	SGS3	SGNexus	Xiaomi
Digimarc	50,00%	25	0	0	0	20	19	14
	100,00%	25	16	16	16	25	25	25
Cifrasoft (robustness=max)	50,00%	6	0	0	0	6	6	6
	100,00%	6	5	5	4	6	6	6
Intrasonics	50,00%	3	0	0	0	3	3	2
	100,00%	3	2	1	1	3	3	3
LISNR (amplitude=0.8)	50,00%	12	1	3	0	3	5	0
	100,00%	12	1	9	0	4	10	0
Prontoly DEMO	50,00%	4	1	2	2	3	1	2
	100,00%	4	4	1	1	4	4	3

DETECTION RATE							
		NOISY ENVIRONMENT			SILENT ENVIRONMENT		
SDK	Watermark Volume	SGS3	SGNexus	Xiaomi	SGS3	SGNexus	Xiaomi
Digimarc	50,00%	0,0%	0,0%	0,0%	80,0%	76,0%	56,0%
	100,00%	64,0%	64,0%	64,0%	100,0%	100,0%	100,0%
Cifrasoft	50,00%	0,0%	0,0%	0,0%	100,0%	100,0%	100,0%
	100,00%	83,3%	83,3%	66,7%	100,0%	100,0%	100,0%
Intrasonics	50,00%	0,0%	0,0%	0,0%	100,0%	100,0%	66,7%
	100,00%	66,7%	33,3%	33,3%	100,0%	100,0%	100,0%
LISNR	50,00%	8,3%	25,0%	0,0%	25,0%	41,7%	0,0%
	100,00%	8,3%	75,0%	0,0%	33,3%	83,3%	0,0%
Prontoly DEMO	50,00%	25,0%	50,0%	50,0%	75,0%	25,0%	50,0%
	100,00%	100,0%	25,0%	25,0%	100,0%	100,0%	75,0%

Tabela 15 - Resultados dos testes às soluções de watermarking - Taxa de detecção

Tabela 16 - Resultados dos testes às soluções de watermarking - Taxa de detecção independente do smartphone

	DEVICE INDEPENDENT DETECTION RATE		
SDK	Watermark Volume	NOISY	SILENT
Digimarc	50,00%	0,0%	70,7%
	100,00%	64,0%	100,0%
Cifrasoft (robustness=max)	50,00%	0,0%	100,0%
	100,00%	77,8%	100,0%
Intrasonics	50,00%	0,0%	88,9%
	100,00%	44,4%	100,0%
LISNR (amplitude=0.8)	50,00%	11,1%	22,2%
	100,00%	27,8%	38,9%
Prontoly DEMO	50,00%	41,7%	50,0%
	100,00%	50,0%	91,7%

Anexo B - Fluxo de Criação de Campanhas no Backoffice

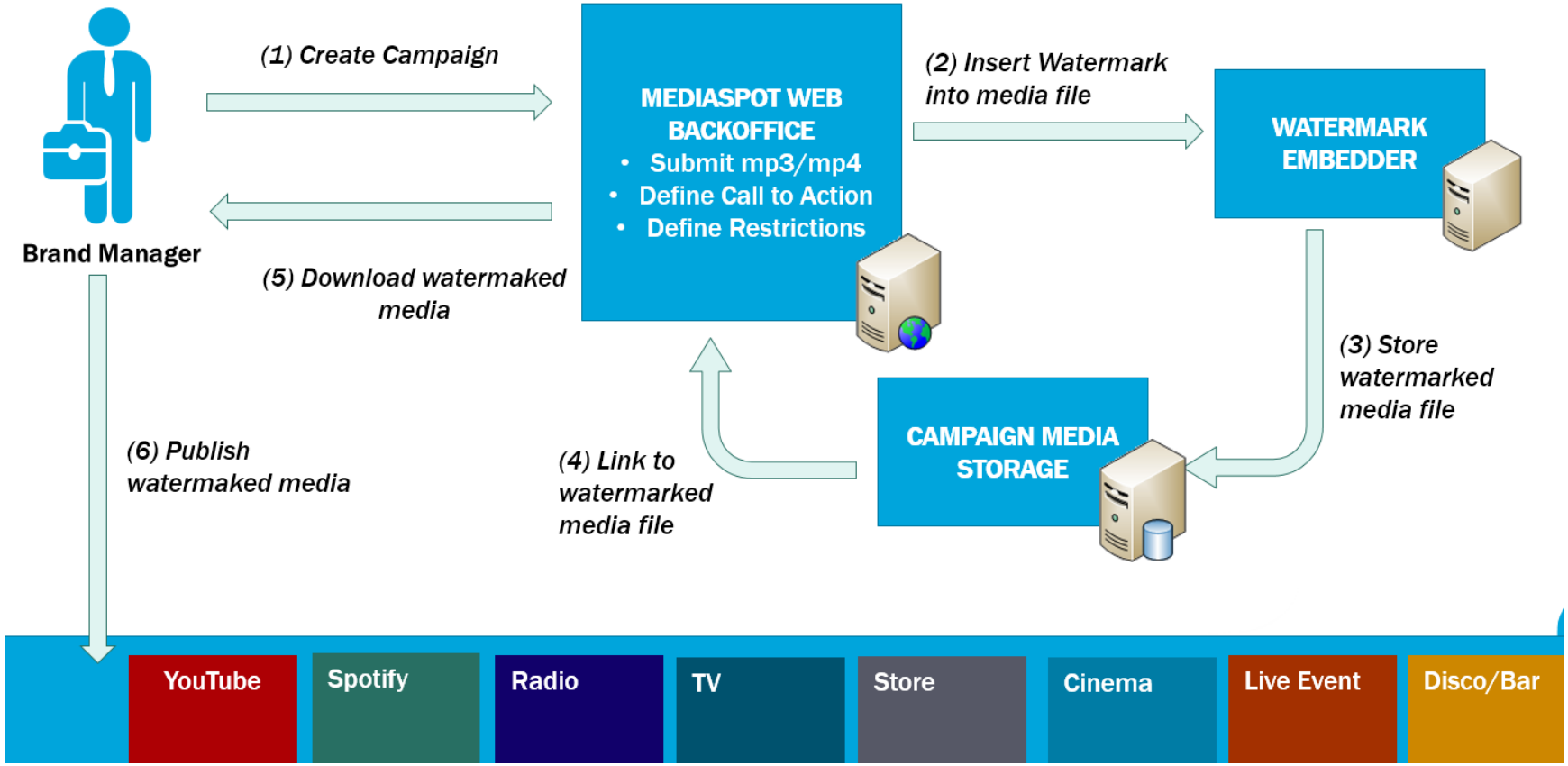


Figura 25 - Fluxo de criação de campanhas no backoffice

Anexo C - Fluxo de Detecção de Campanhas Provisionadas no Backoffice

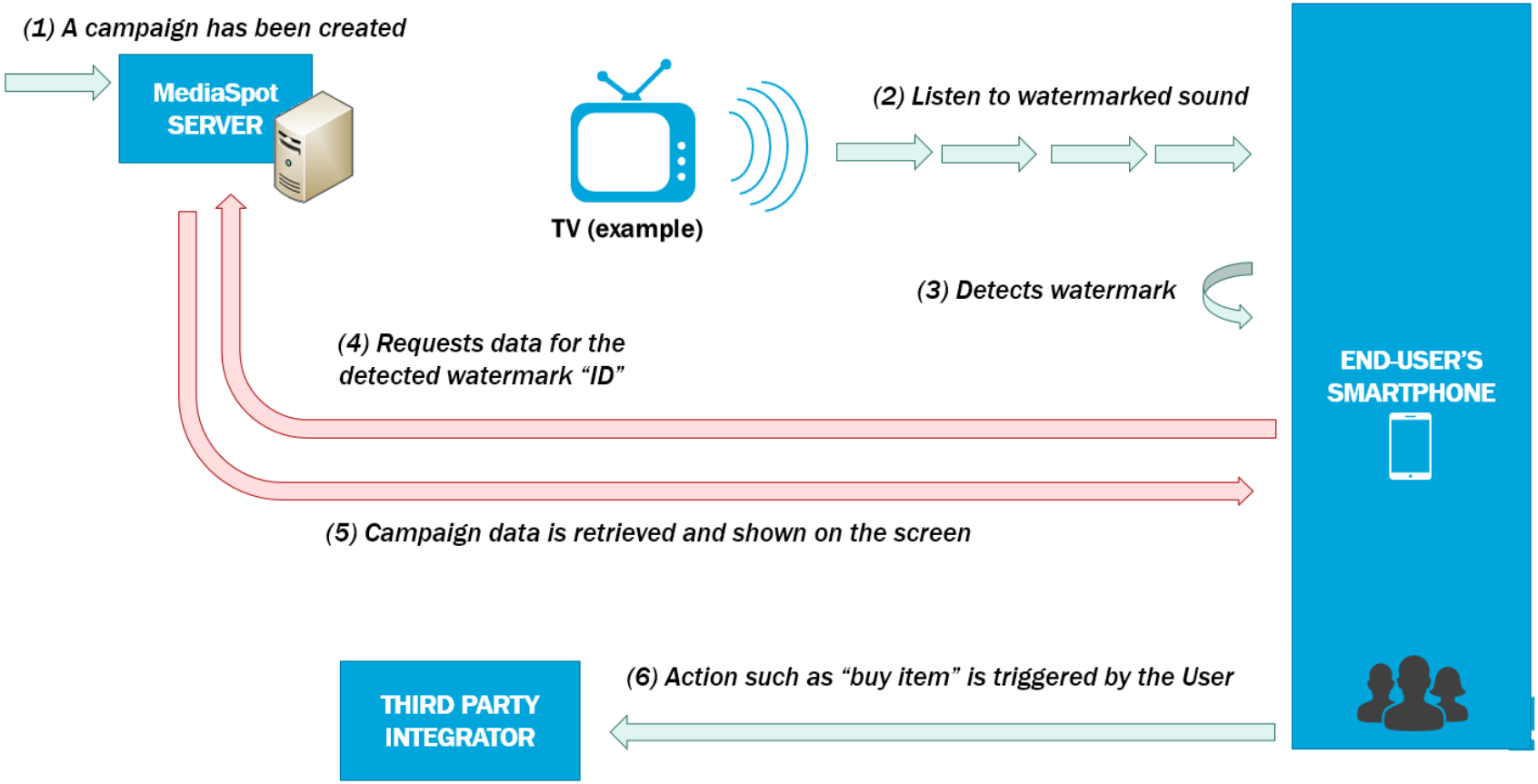


Figura 26 - Fluxo de detecção de campanhas provisionadas no backoffice

Anexo D - Mockups Desenhados (versão final)

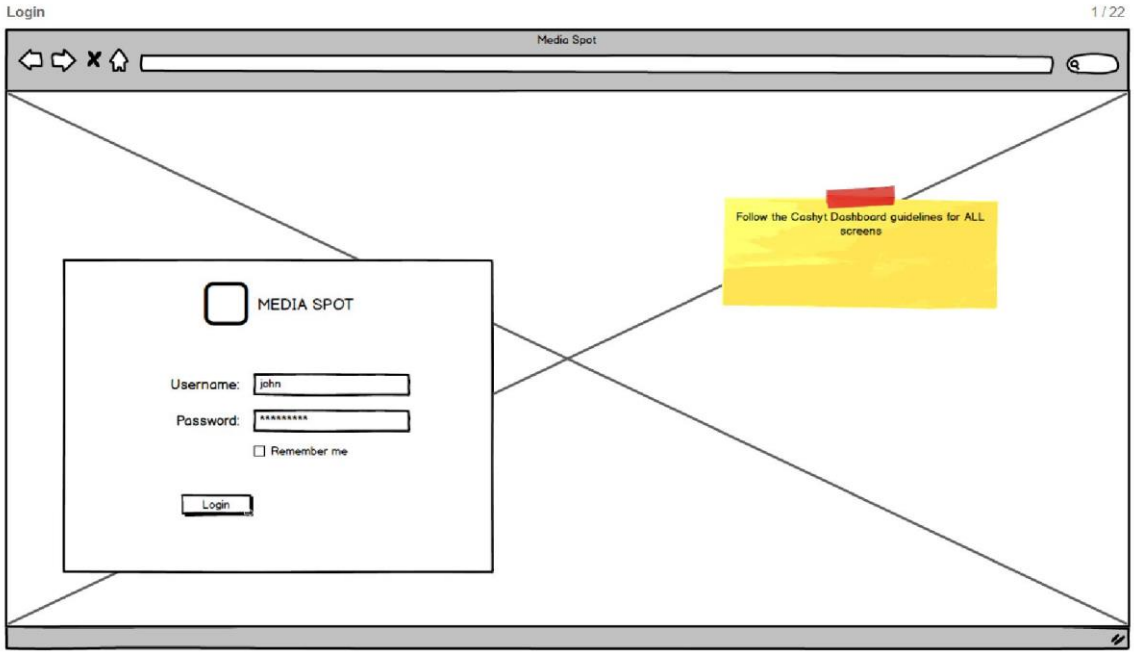


Figura 27 - Mockup - Ecrã de login

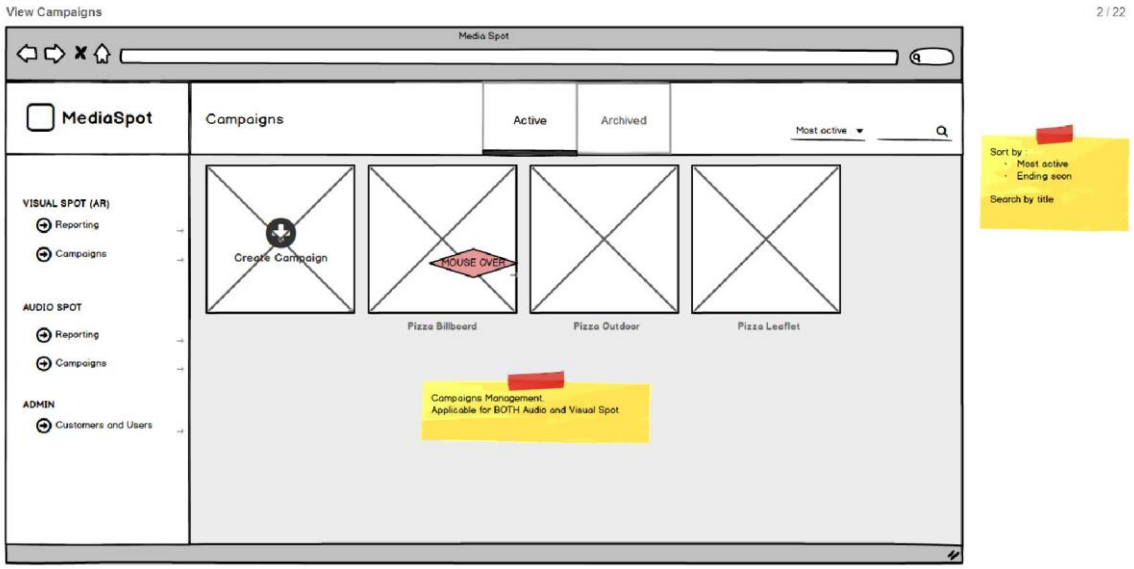


Figura 28 - Mockup - Ecrã de campanhas activas

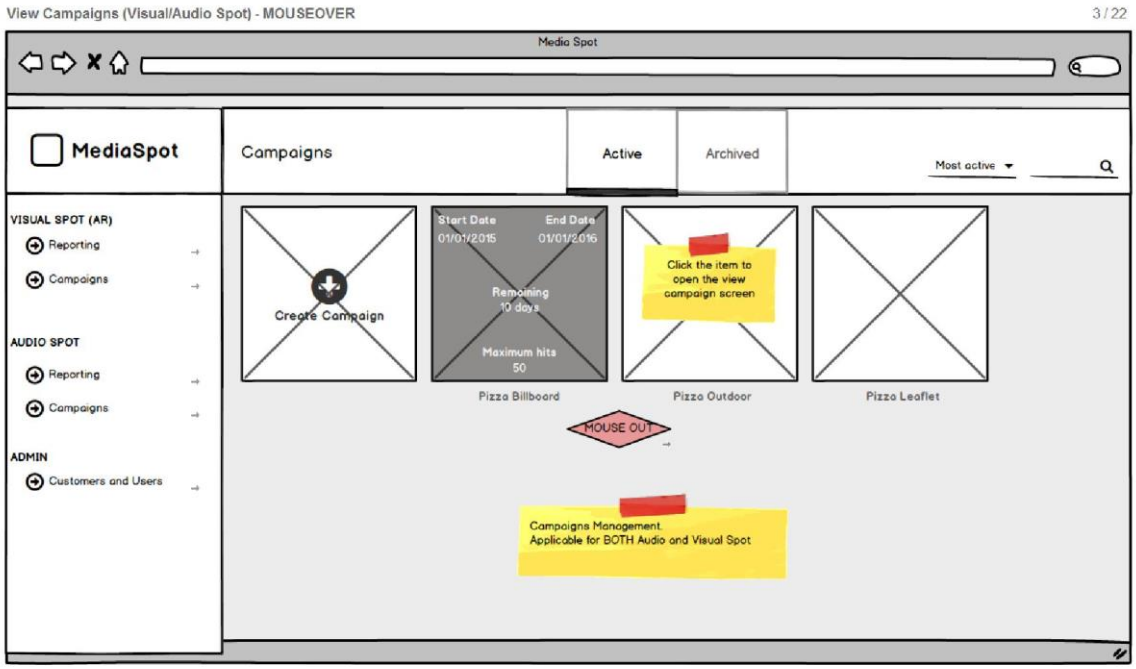


Figura 29 - Mockup - Ecrã de campanhas activas (descrição da interacção)

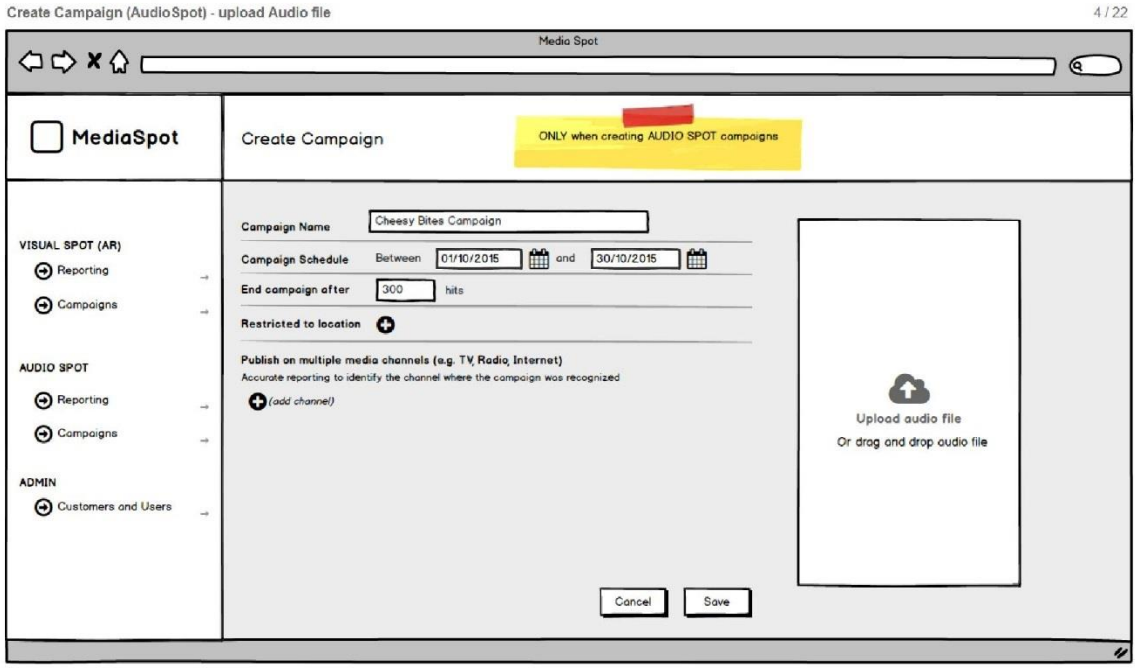


Figura 30 - Mockup - Ecrã de criação de campanha

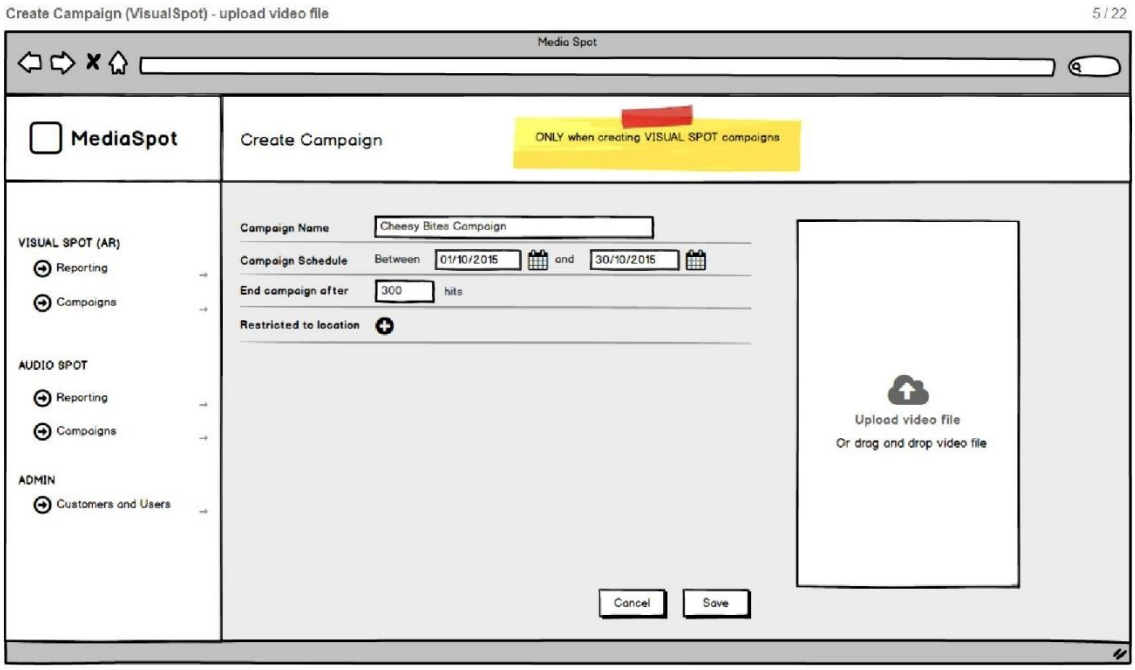


Figura 31 - Mockup - Ecrã de criação de campanha (2)

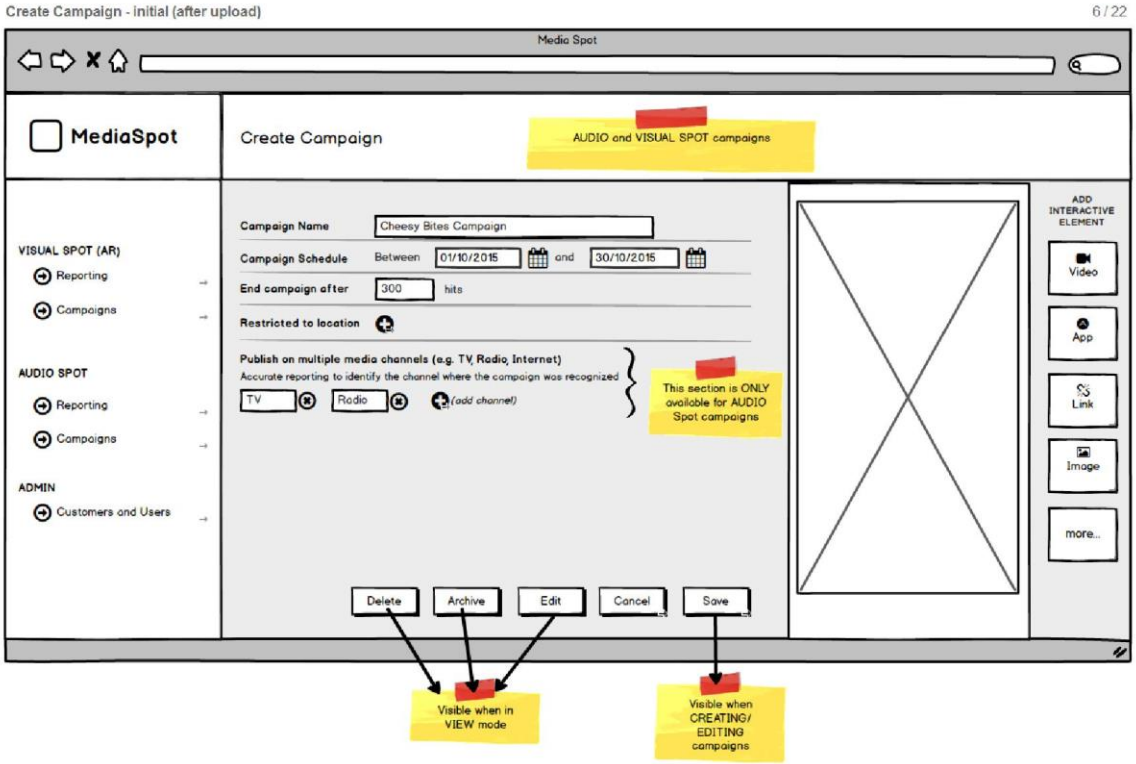


Figura 32 - Mockup - Ecrã de criação da campanha (detalhe de preenchimento de form)

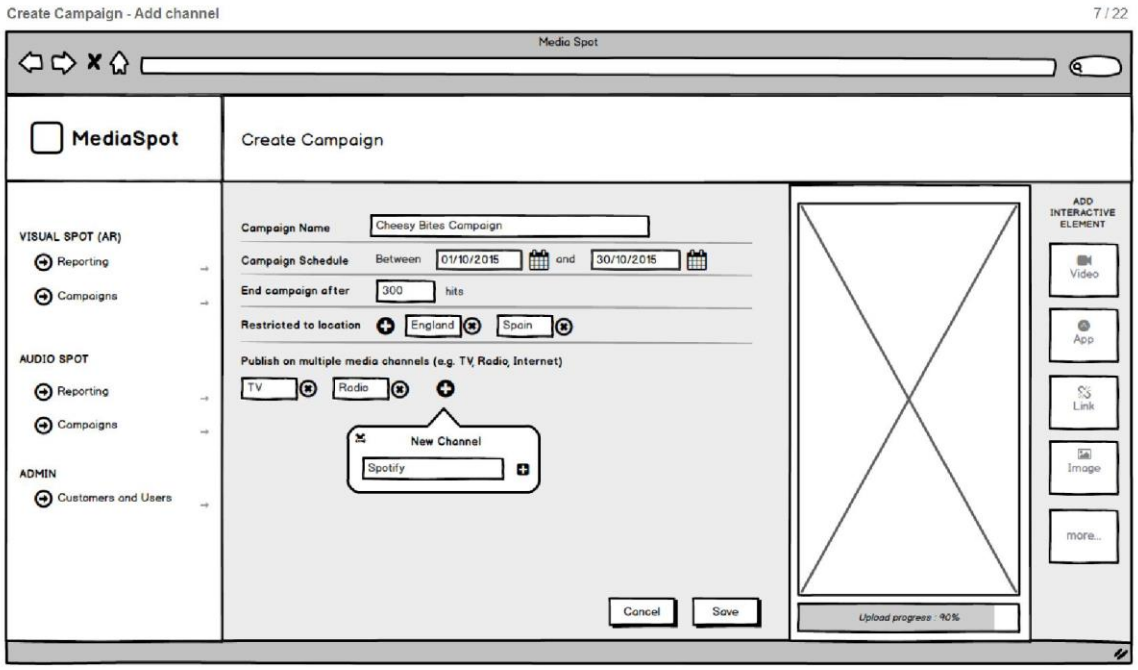


Figura 33 - Mockup - Ecrã de criação da campanha (associação de canais de publicação)

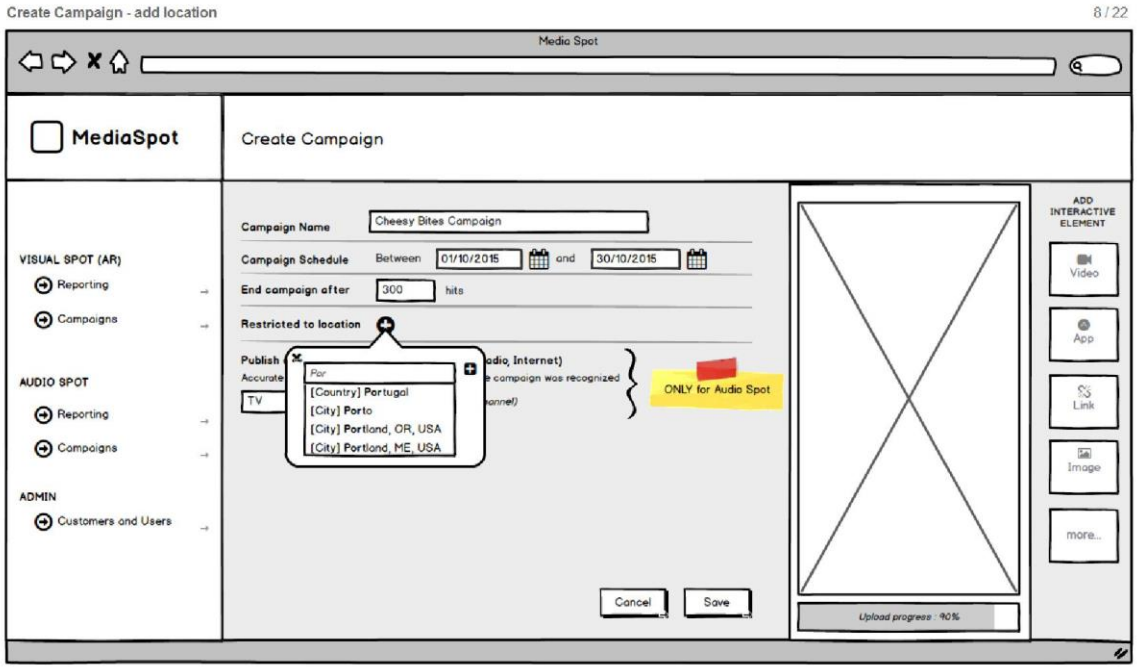


Figura 34 - Mockup - Ecrã de criação da campanha (restrição de localização)

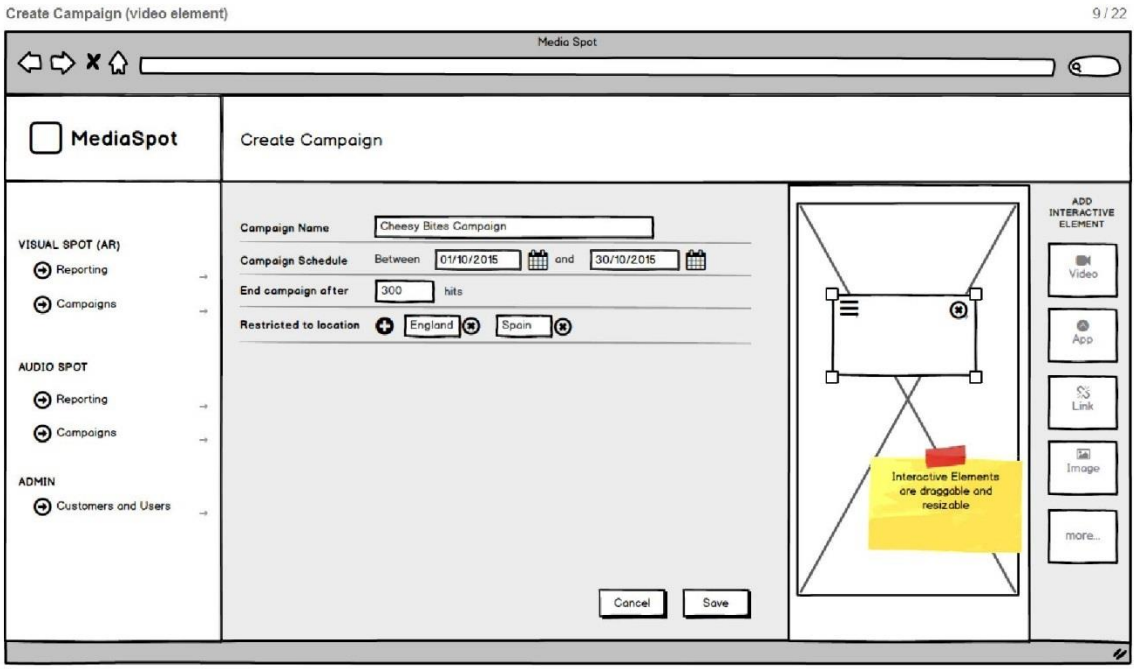


Figura 35 - Mockup - Ecrã de criação da campanha (video element)

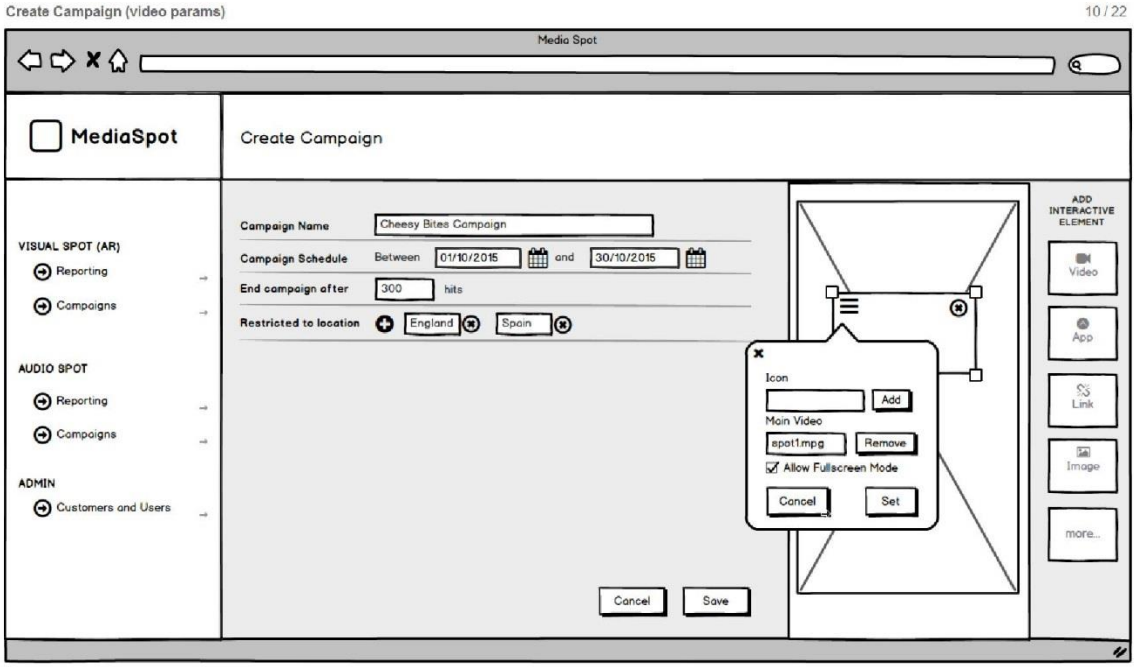


Figura 36 - Mockup - Ecrã de criação da campanha (parâmetros do video element)

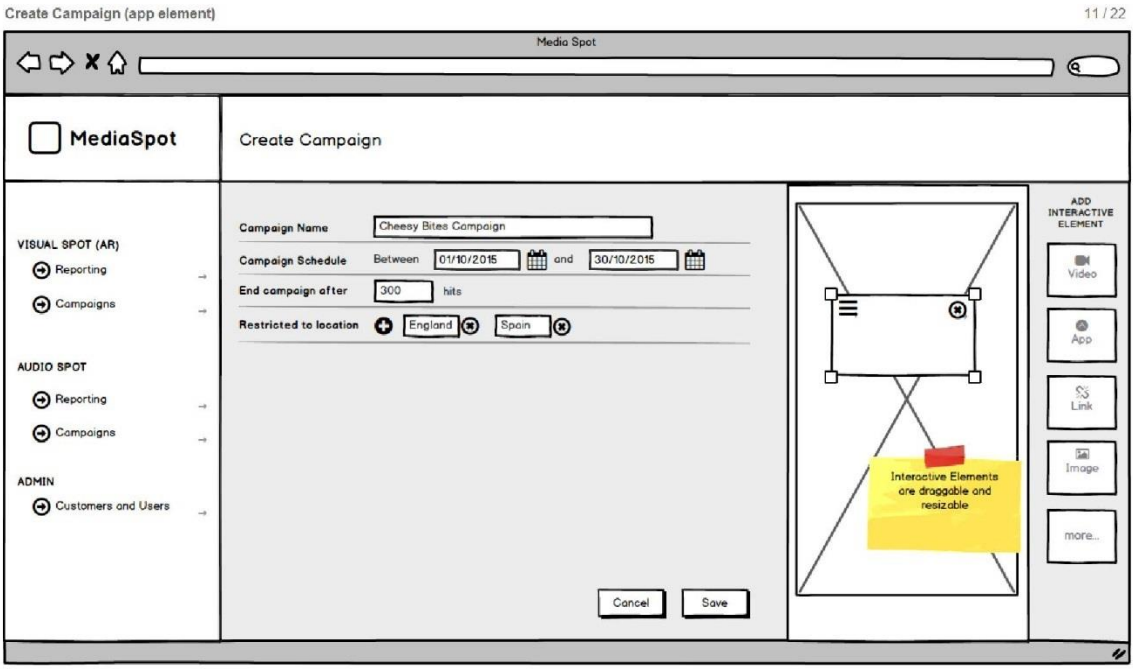


Figura 37 - Mockup - Ecrã de criação da campanha (app element)

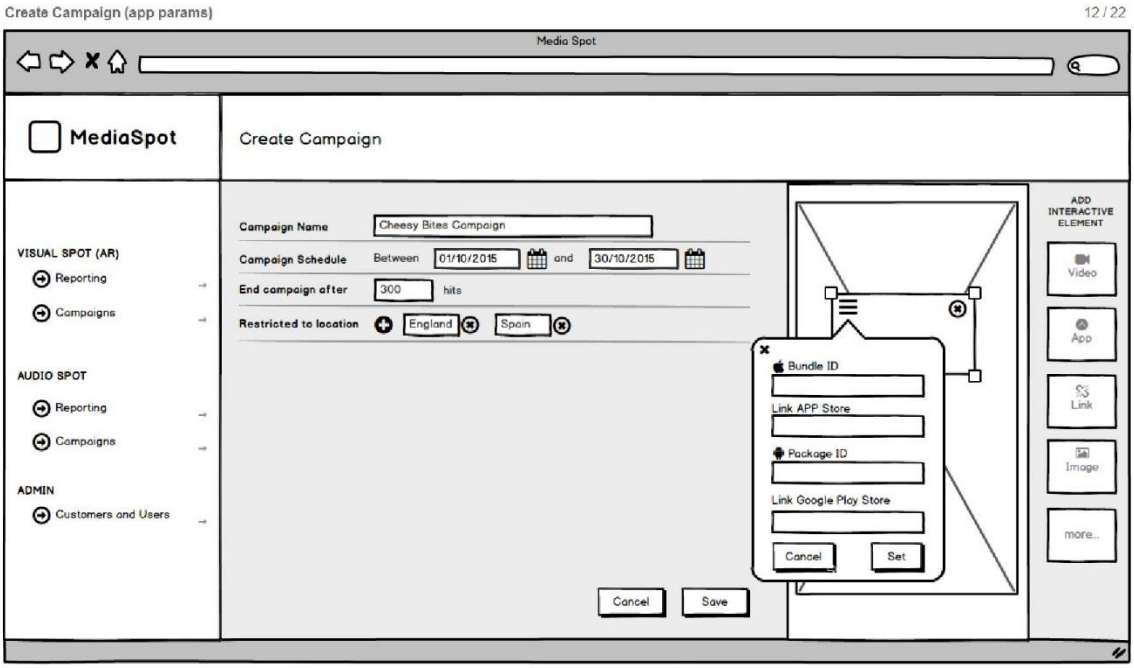


Figura 38 - Mockup - Ecrã de criação da campanha (parâmetros do app element)

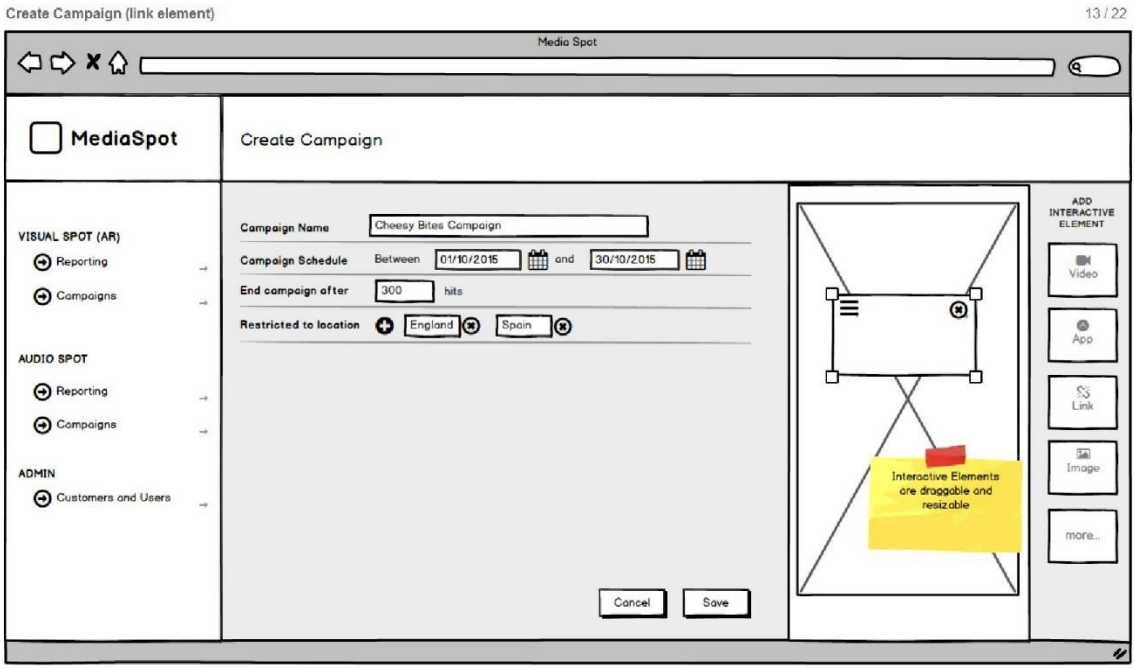


Figura 39 - Mockup - Ecrã de criação da campanha (link element)

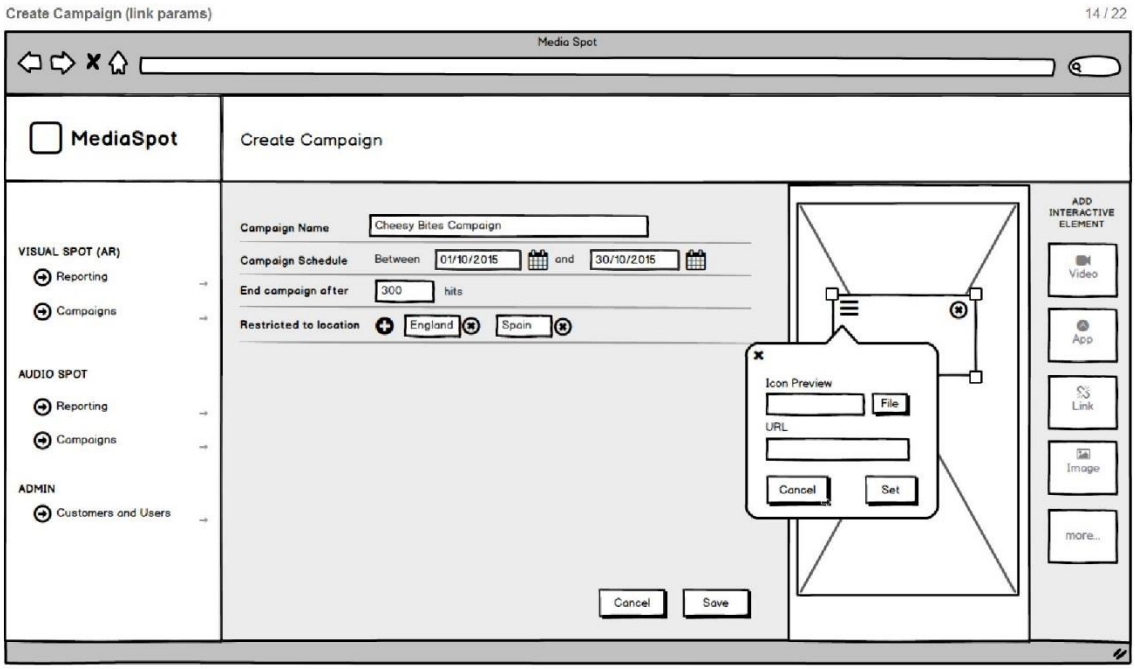


Figura 40 - Mockup - Ecrã de criação da campanha (parâmetros do link element)

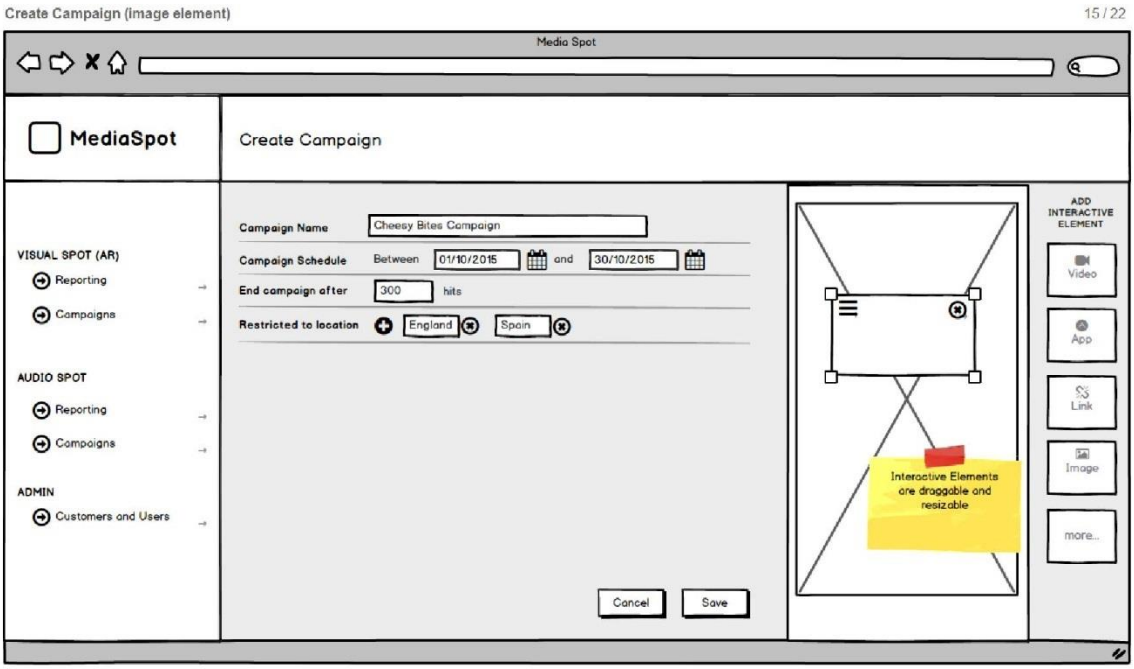


Figura 41 - Mockup - Ecrã de criação da campanha (image element)

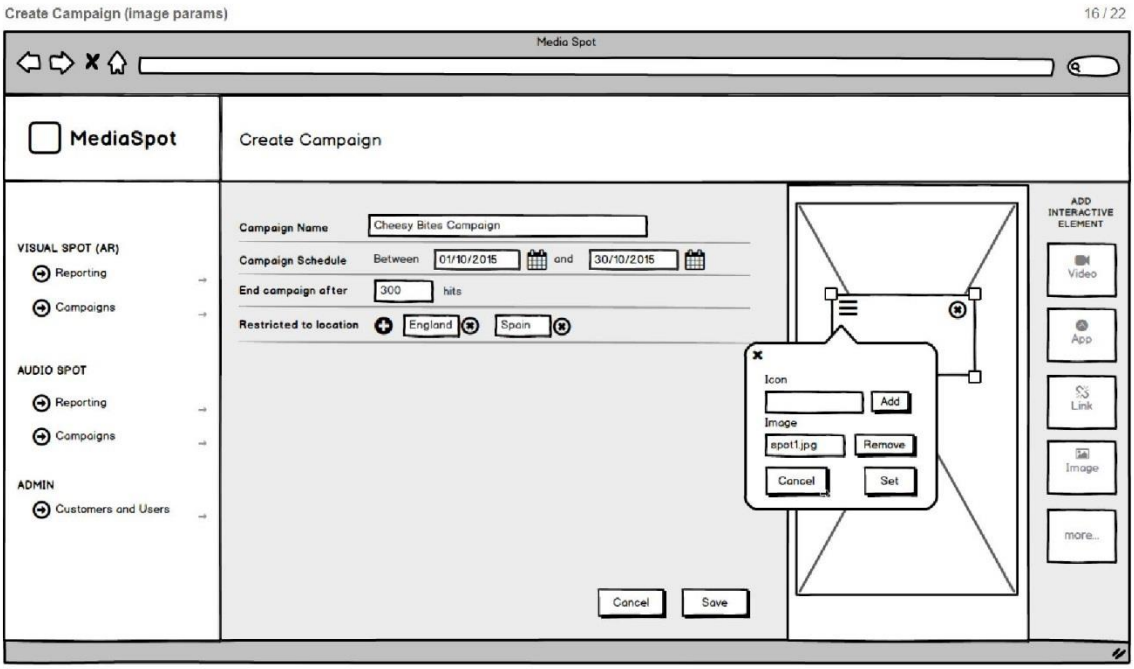


Figura 42 - Mockup - Ecrã de criação da campanha (parâmetros do image element)

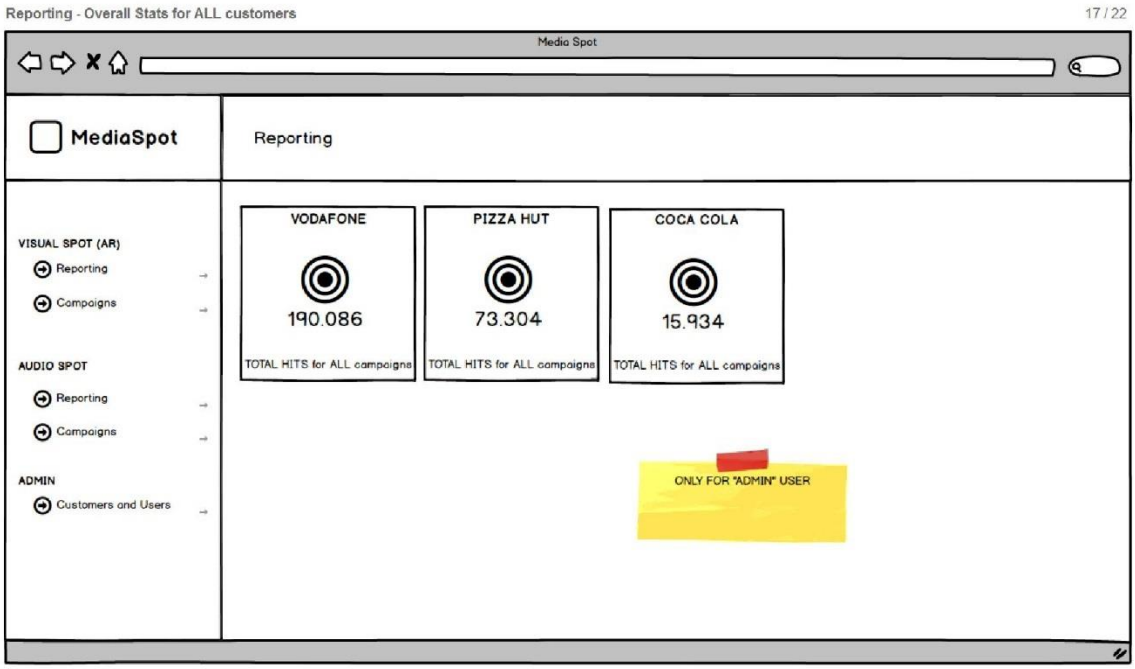


Figura 43 – Mockup - Ecrã de reporting (lista de customers)

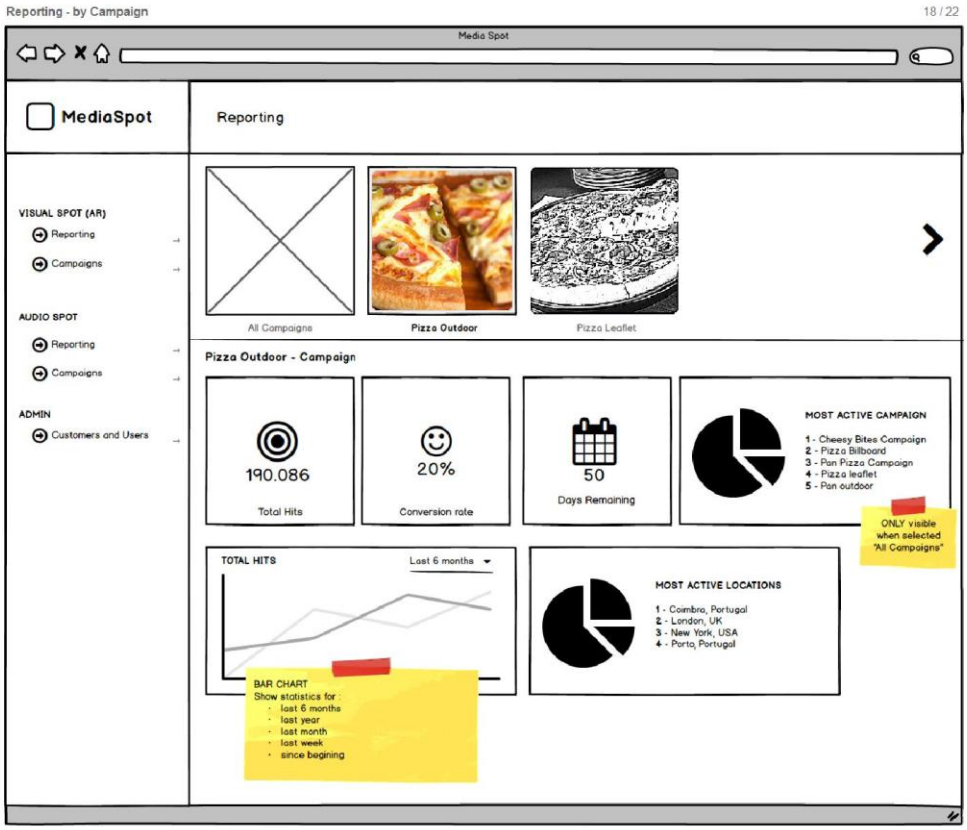


Figura 44 – Mockup - Ecrã de reporting (campanha)

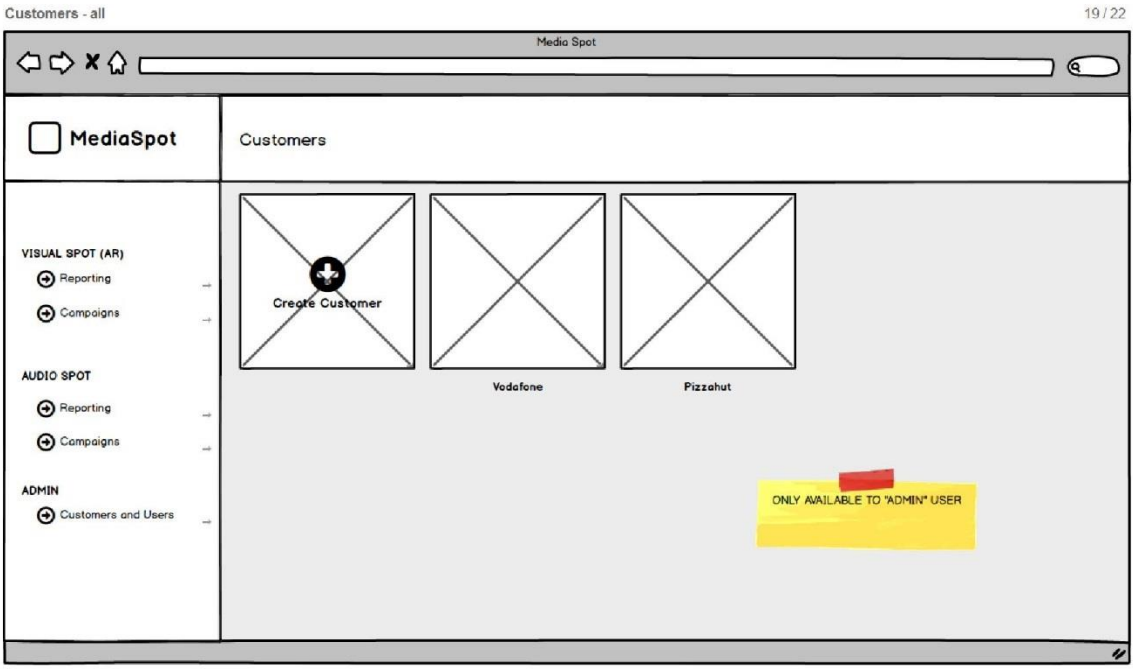


Figura 45 - Mockup - Ecrã de customers

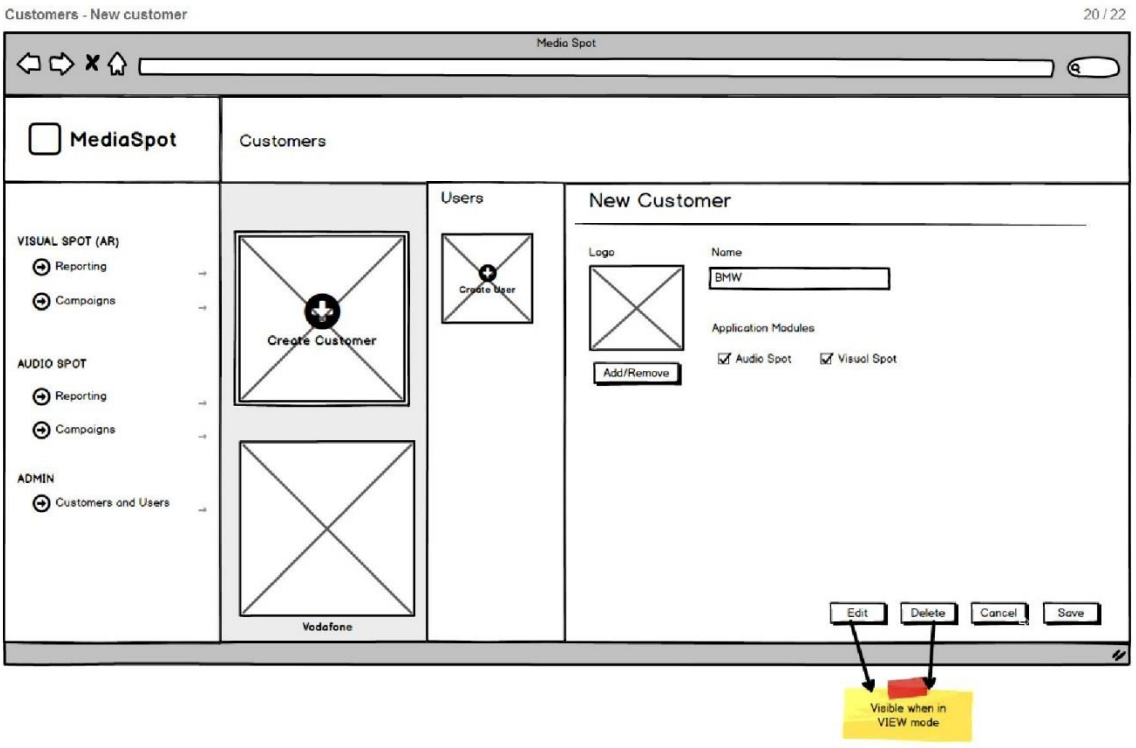


Figura 46 - Mockup - Ecrã de criação de customer

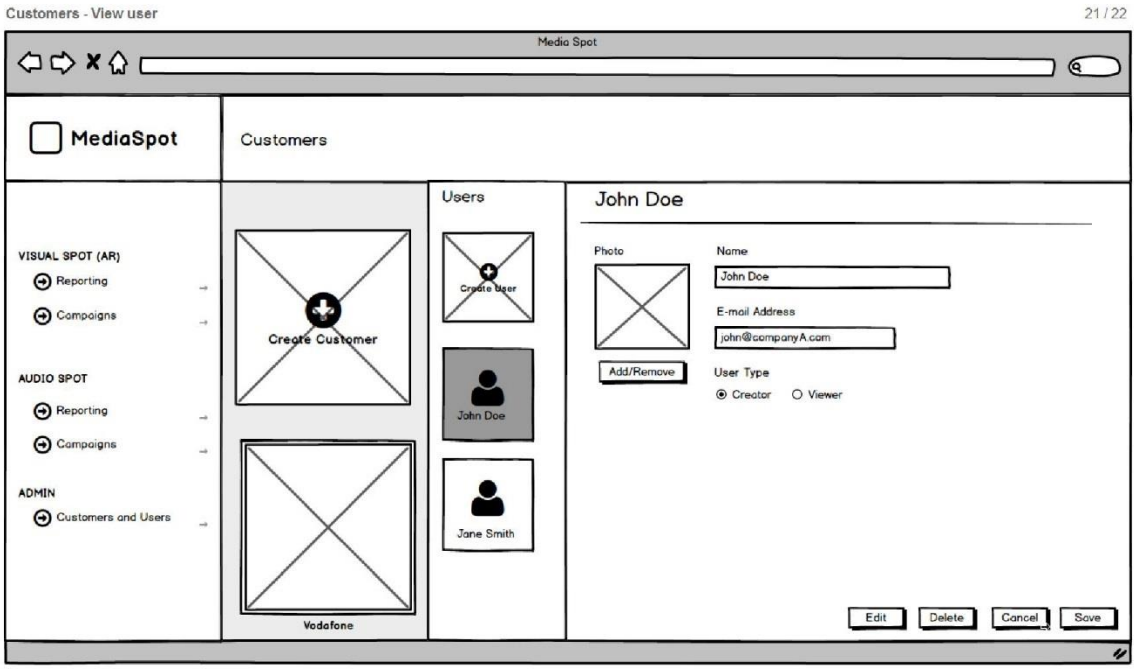


Figura 47 - Mockup- Ecrã de associação de um user a um customer

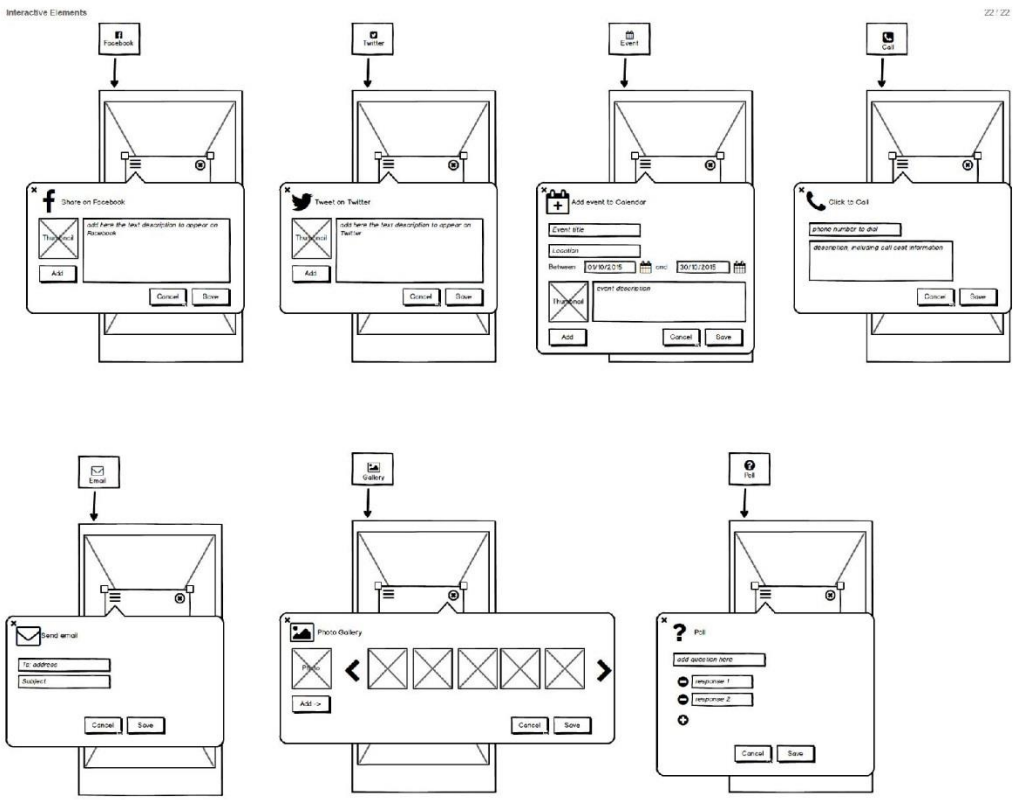
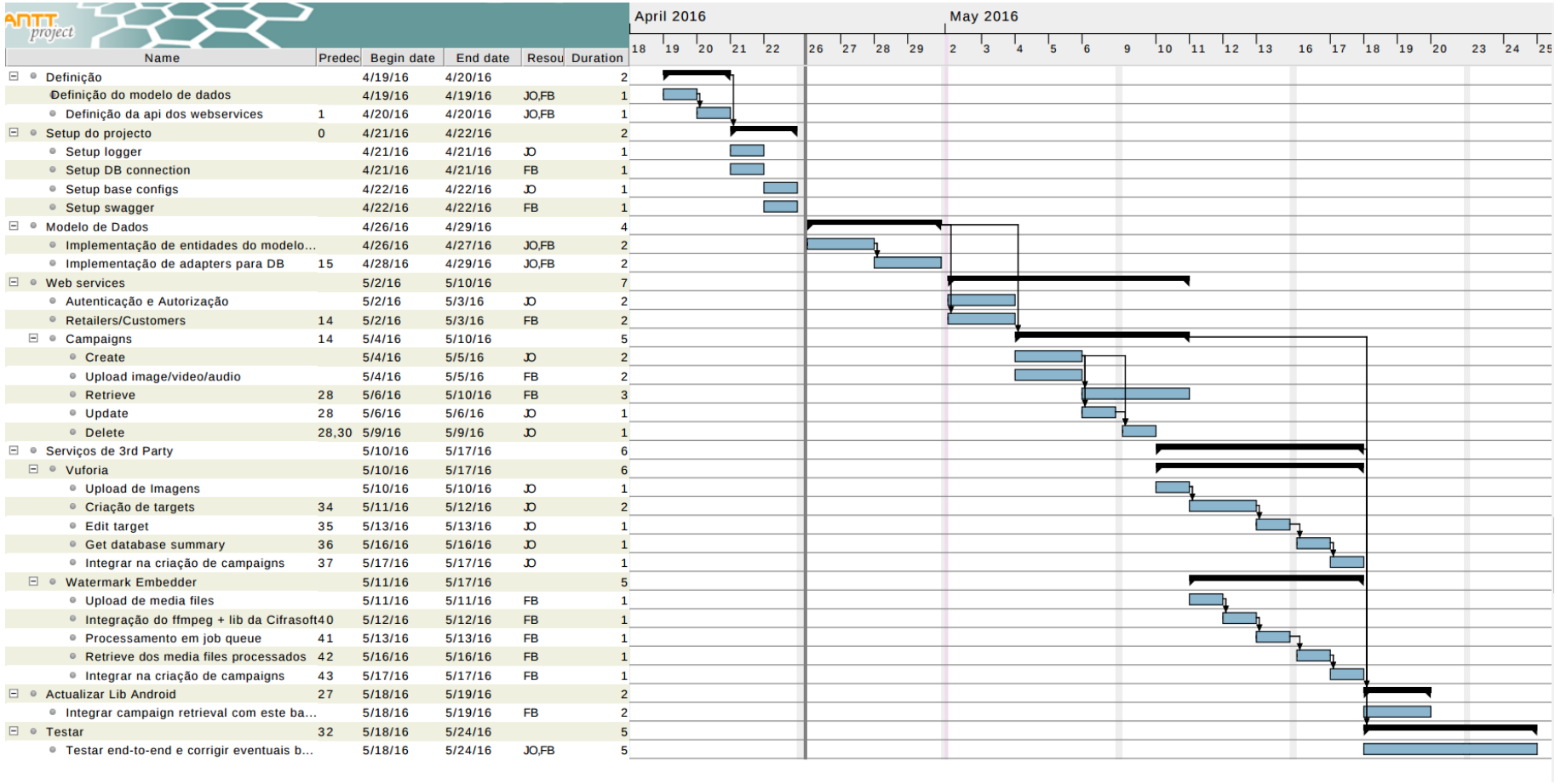


Figura 48 - Mockup - Parâmetros de diversos elementos das campanhas

Anexo E – Calendarização das tarefas da primeira release do backend



Name	Predec	Begin date	End date	Resou	Duration
Definição		4/19/16	4/20/16		2
Definição do modelo de dados		4/19/16	4/19/16	JO,FB	1
Definição da api dos webservices	1	4/20/16	4/20/16	JO,FB	1
Setup do projecto	0	4/21/16	4/22/16		2
Setup logger		4/21/16	4/21/16	JO	1
Setup DB connection		4/21/16	4/21/16	FB	1
Setup base configs		4/22/16	4/22/16	JO	1
Setup swagger		4/22/16	4/22/16	FB	1
Modelo de Dados		4/26/16	4/29/16		4
Implementação de entidades do modelo...		4/26/16	4/27/16	JO,FB	2
Implementação de adapters para DB	15	4/28/16	4/29/16	JO,FB	2
Web services		5/2/16	5/10/16		7
Autenticação e Autorização		5/2/16	5/3/16	JO	2
Retailers/Customers	14	5/2/16	5/3/16	FB	2
Campaigns	14	5/4/16	5/10/16		5
Create		5/4/16	5/5/16	JO	2
Upload image/video/audio		5/4/16	5/5/16	FB	2
Retrieve	28	5/6/16	5/10/16	FB	3
Update	28	5/6/16	5/6/16	JO	1
Delete	28,30	5/9/16	5/9/16	JO	1
Serviços de 3rd Party		5/10/16	5/17/16		6
Vuforia		5/10/16	5/17/16		6
Upload de Imagens		5/10/16	5/10/16	JO	1
Criação de targets	34	5/11/16	5/12/16	JO	2
Edit target	35	5/13/16	5/13/16	JO	1
Get database summary	36	5/16/16	5/16/16	JO	1
Integrar na criação de campaigns	37	5/17/16	5/17/16	JO	1
Watermark Embedder		5/11/16	5/17/16		5
Upload de media files		5/11/16	5/11/16	FB	1
Integração do ffmpeg + lib da Cifrasoft	40	5/12/16	5/12/16	FB	1
Processamento em job queue	41	5/13/16	5/13/16	FB	1
Retrieve dos media files processados	42	5/16/16	5/16/16	FB	1
Integrar na criação de campaigns	43	5/17/16	5/17/16	FB	1
Actualizar Lib Android	27	5/18/16	5/19/16		2
Integrar campaign retrieval com este ba...		5/18/16	5/19/16	FB	2
Testar	32	5/18/16	5/24/16		5
Testar end-to-end e corrigir eventuais b...		5/18/16	5/24/16	JO,FB	5

Figura 49- Diagrama de *gantt* do plano para a primeira *release* do *backend*

Anexo F - Diagrama de Classes – Classes do Modelo de Dados do Backend

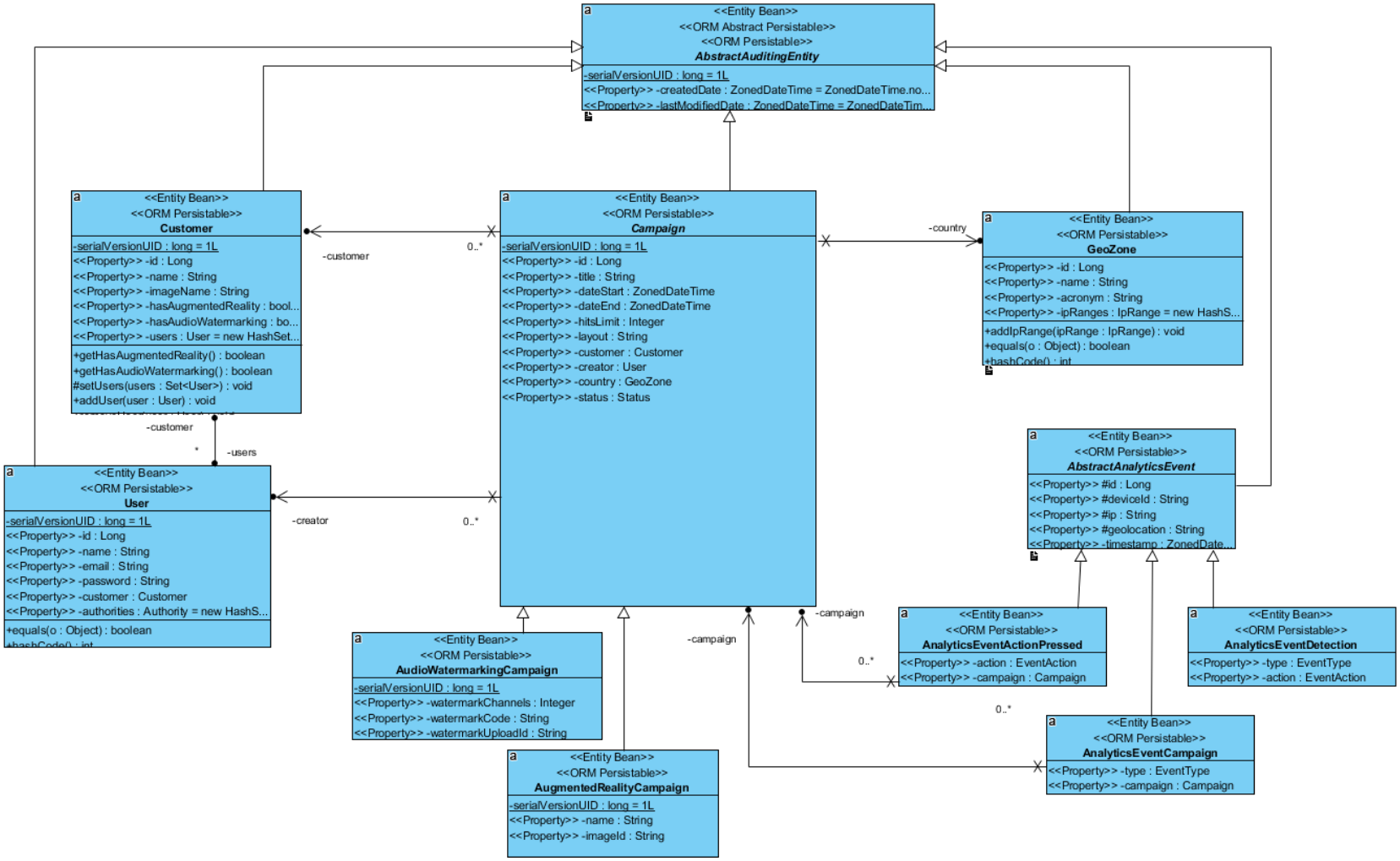


Figura 50 – Diagrama de classes do modelo de dados do backend

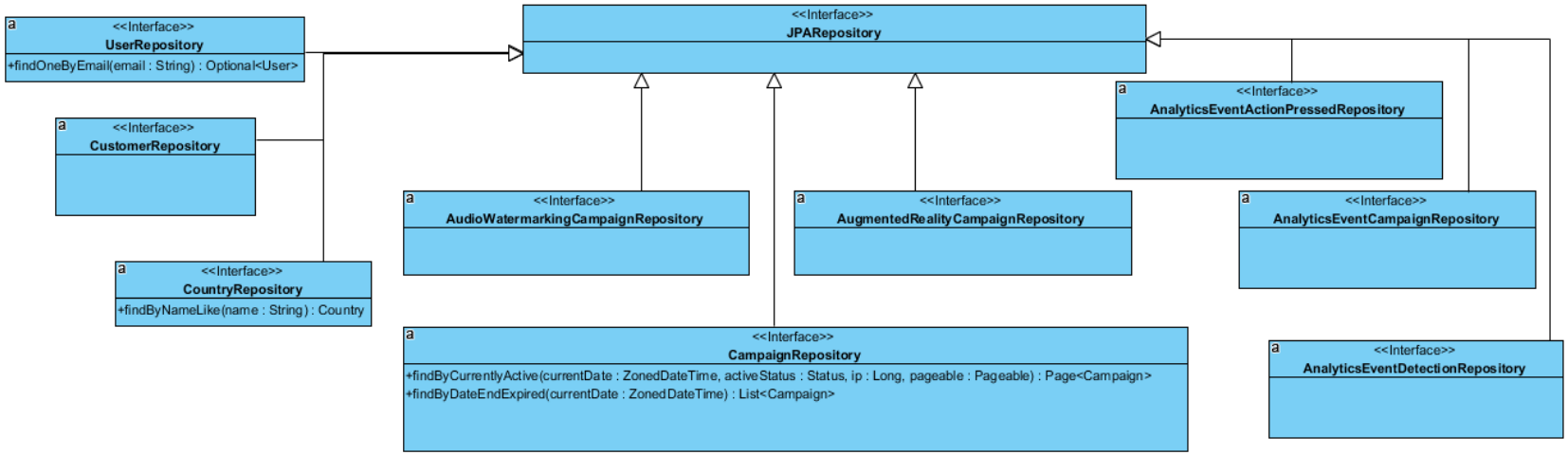


Figura 51- Diagrama de classes mediadoras da interacção entre camada de negócio e modelo de dados

Anexo G – Resultados dos Testes de Integração Automatizados

Figura 52 - Lista dos testes de integração automatizados e o seu tempo de execução

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	89% (113/ 127)	83.1% (552/ 664)	75.6% (1934/ 2557)

Coverage Breakdown

Package ▲	Class, %	Method, %	Line, %
com.witsoftware.mediaspotbackend	100% (2/ 2)	75% (9/ 12)	76.7% (69/ 90)
com.witsoftware.mediaspotbackend.aop.logging	100% (1/ 1)	50% (2/ 4)	40.9% (9/ 22)
com.witsoftware.mediaspotbackend.config	90.9% (10/ 11)	73.1% (38/ 52)	80.9% (114/ 141)
com.witsoftware.mediaspotbackend.domain	94.7% (18/ 19)	85.9% (116/ 135)	80.9% (195/ 241)
com.witsoftware.mediaspotbackend.domain.builder	100% (4/ 4)	73.3% (11/ 15)	77.9% (67/ 86)
com.witsoftware.mediaspotbackend.domain.specifications	100% (4/ 4)	87.5% (14/ 16)	63.4% (111/ 175)
com.witsoftware.mediaspotbackend.domain.utils	45.5% (5/ 11)	37.9% (11/ 29)	41.9% (13/ 31)
com.witsoftware.mediaspotbackend.security	80% (4/ 5)	64.3% (9/ 14)	61.6% (45/ 73)
com.witsoftware.mediaspotbackend.security.jwt	100% (3/ 3)	54.5% (6/ 11)	39.5% (32/ 81)
com.witsoftware.mediaspotbackend.service.exceptions	73.3% (11/ 15)	68.8% (11/ 16)	75.9% (22/ 29)
com.witsoftware.mediaspotbackend.service.impl	100% (9/ 9)	85.5% (59/ 69)	79.7% (326/ 409)
com.witsoftware.mediaspotbackend.task	100% (1/ 1)	50% (1/ 2)	16.7% (2/ 12)
com.witsoftware.mediaspotbackend.utils	100% (5/ 5)	80% (20/ 25)	82.1% (138/ 168)
com.witsoftware.mediaspotbackend.web.rest	88.9% (8/ 9)	94.9% (37/ 39)	84.3% (291/ 345)
com.witsoftware.mediaspotbackend.web.rest.constraints.validators	100% (3/ 3)	100% (12/ 12)	83.6% (46/ 55)
com.witsoftware.mediaspotbackend.web.rest.dto	100% (19/ 19)	95.6% (173/ 181)	91.3% (274/ 300)
com.witsoftware.mediaspotbackend.web.rest.mapper	100% (5/ 5)	73.3% (22/ 30)	58.7% (166/ 283)
com.witsoftware.mediaspotbackend.web.rest.utils	100% (1/ 1)	50% (1/ 2)	87.5% (14/ 16)

Figura 53 - Relatório de code coverage dos testes de integração

Anexo H – Resultados dos Testes de Carga

Tabela 17- Resultados dos testes de carga

		1	2	3	4	5	6	7
Configuration Values	Fixed Thread Number	100	100	100	100	100	100	100
	Overall Test Duration (seconds)	120	120	120	120	120	120	120
	Ramp-up (seconds)	10	10	10	10	10	10	10
	Target Throughput	**300**	**325**	**350**	**375**	**400**	**425**	**450**
Results	Throughput (obtained)	300.2	315.1	350.5	375.5	400.7	407.9	449.5
	#Samples (obtained/expected)	36025 / 36000	37816 / 39000	42061 / 42000	45056 / 45000	48085 / 48000	48942 / 51000	53945 / 54000
	Average response time	12	17	17	22	30	39	47
System Monitoring	**CPU**							
	Average (%)	59.1	63.4	67	68	70.8	78.2	93.1
	Peak (%)	96	94	86	88	88	94	93
	Min (%)	44	41	31	44	59	56	59
	Memory							
	Min Free (MB)	1301	1254	1037	1326	1397	1301	1241

Anexo I - Relatórios de Progresso

Período	8 de Novembro de 2015 a 13 de Novembro de 2015
Tarefas Realizadas	• Trabalho de pesquisa sobre *audio watermarking*. • Produção de um *powerpoint* com estado da arte sobre a matéria.
Principais Dificuldades	• Escassez de material *hands-on* que permite testar e comparar devidamente cada uma das soluções. Necessidade de entrar em contacto com cada uma das entidades para obter Demos e *SDKs* (se é que realmente existem). • Inexistência de soluções *open-source*.

Período	15 de Novembro de 2015 a 20 de Novembro de 2015
Tarefas Realizadas	• Desenvolvimento de uma demo com os *SDKs* da *Digimarc*, *Cifrasoft* e *Intrasonics* • Testes com ruido ambiente • Produção de um breve *powerpoint* com uma análise dos *SDKs* e resultados dos testes
Principais Dificuldades	• Ainda não foi possível obter *SDKs* que usem frequências inaudíveis

Período	22 de Novembro de 2015 a 27 de Novembro de 2015
Tarefas Realizadas	• Adicionado *SDK* da *LISNR* à demo desenvolvida • Testes isolados com e sem ruído ambiente • Produção de um *powerpoint* onde são abordados possíveis casos de uso da tecnologia
Principais Dificuldades	• Foi pedido para pensar num *use case* "fora do tradicional". Houve alguma dificuldade no pensamento inventivo dado que já existem imensos casos de uso distintos no mercado.

Período	30 de Novembro de 2015 a 4 de Dezembro de 2015
Tarefas Realizadas	• Produzido *powerpoint* com descrição dos testes realizados na semana anterior • Pesquisa e leitura sobre *Product Management* • Iniciado o desenho de *wireframes* para o *backend* da solução
Principais Dificuldades	• Não foi possível obter *feedback* em relação ao trabalho da semana passada

Período	6 de Dezembro de 2015 a 11 de Dezembro de 2015
Tarefas Realizadas	• Terminado o desenho dos *wireframes* • Análise de *frameworks* modernas para desenvolvimento web • Desenho de diagrama de componentes/*deploy* do *backoffice* • Preparação dos ambientes de desenvolvimento no *Docker*
Principais Dificuldades	• Nada a apontar

Período	14 de Dezembro de 2015 a 18 de Dezembro de 2015
Tarefas Realizadas	• Análise de *frameworks* e *libs* para desenvolvimento web (*JavaScript*, *nodejs*), configuração de *containers* de desenvolvimento. • Reunião interna para apresentação da tecnologia (demo técnica) • Desenvolvimento de demos rápidas aplicadas à publicidade
Principais Dificuldades	• Nesta fase os clientes ainda não se mostram convencidos acerca da tecnologia

Período	4 de Janeiro de 2016 a 8 de Janeiro de 2016
Tarefas Realizadas	• Reunião interna para apresentação de possíveis casos de uso desta tecnologia • Testes com *webview* no *payoff* da detecção de *watermarks* • Optimização de código *web* • Pesquisa de *frameworks web*
Principais Dificuldades	• Nada a apontar

Período	11 de Janeiro de 2016 a 15 de Janeiro de 2016
Tarefas Realizadas	• Preparação de demo para apresentar a um cliente: • Criação de nova *demo app* de detecção de *watermarks* • Experimentada a integração da tecnologia numa app que estava a ser desenvolvida na empresa para dispositivos *Nexus Player*
Principais Dificuldades	• Nada a apontar

Período	18 de Janeiro de 2016 a 22 de Janeiro de 2016
Tarefas Realizadas	• Desenvolvimento de demo em iOS
Principais Dificuldades	• Nada a apontar

Período	25 de Janeiro de 2016 a 29 de Janeiro de 2016
Tarefas Realizadas	• Testes com *transcoding* no *youtube* • Desenho diagramas *high level* para o *backoffice* • *Crash course* sobre *Angular* e bibliotecas para testes unitários
Principais Dificuldades	• Alguma dificuldade em conseguir *feedback* para validar o trabalho de realizado e continuar com outras tarefas

Período	1 de Fevereiro de 2016 a 5 de Fevereiro de 2016
Tarefas Realizadas	• Especificação de casos de uso adicionais • *Brainstorming* para um *use case* mais completo • Leitura sobre *gamification* e *steganography* • Reunião para conhecer um novo produto interno no qual se pretendia inserir a tecnologia de *audio watermarking*
Principais Dificuldades	• Progresso irrelevante

Período	8 de Fevereiro de 2016 a 11 de Fevereiro de 2016
Tarefas Realizadas	• Criação de novos *wireframes* • Desenvolvimento de demo com recurso a *fingerprinting*
Principais Dificuldades	• Alguma dificuldade em conseguir *feedback* para validar o trabalho de realizado e continuar com outras tarefas

Período	15 de Fevereiro de 2016 a 19 de Fevereiro de 2016
Tarefas Realizadas	• Produzido *powerpoint* com especificação do *backoffice* • Desenvolvimento de *demo app* para a feira *Mobile World Congress* • Análise sobre *Spring Boot*
Principais Dificuldades	• Alguma dificuldade em conseguir feedback para validar o trabalho de realizado e continuar com outras tarefas

Período	22 de Fevereiro de 2016 a 26 de Fevereiro de 2016
Tarefas Realizadas	• Revisão dos *wireframes* que foram finalmente validados • Testes com *framework Spring Boot*
Principais Dificuldades	• Nada a apontar

Período	29 de Fevereiro de 2016 a 4 de Março de 2016
Tarefas Realizadas	• Revisão dos *wireframes* que foram finalmente validados • Testes com *framework Spring Boot*
Principais Dificuldades	• Nada a apontar

Período	7 de Março de 2016 a 11 de Março de 2016
Tarefas Realizadas	• Após reunião, foi decidido refazer os *wireframes* Continuação de aprendizagem sobre *Spring Boot*
Principais Dificuldades	• Retrocesso em relação à definição do *backoffice*

Período	14 de Março de 2016 a 18 de Março de 2016
Tarefas Realizadas	• Continuação do desenvolvimento de *wireframes* • Continuação de aprendizagem sobre *Spring Boot*
Principais Dificuldades	• Não foi possível obter feedback (de todas as partes interessada) sobre o trabalho já realizado nos *wireframes*

Período	4 de Abril de 2016 a 8 de Abril de 2016
Tarefas Realizadas	• Actualização dos *wireframes* de acordo com *feedback* do orientador • Implementação de PoC em Android para a construção de ecrãs através de *JSON* definido remotamente
Principais Dificuldades	• Começou-se a perceber que as opiniões mudam constantemente acerca da visão e âmbito do *backoffice* (influenciando os *wireframes*)

Período	11 de Abril de 2016 a 15 de Abril de 2016
Tarefas Realizadas	• Desenvolvimento de demo com interface mais trabalhada
Principais Dificuldades	• Nada a apontar

Período	18 de Abril de 2016 a 22 de Abril de 2016
Tarefas Realizadas	• Continuação da tarefa da semana anterior
Principais Dificuldades	• Nada a apontar

Período	25 de Abril de 2016 a 29 de Abril de 2016
Tarefas Realizadas	• Terminado desenvolvimento da demo da semana anterior • Planeamento do desenvolvimento do *backend* • Início do desenvolvimento do *backend* (modelo de dados)
Principais Dificuldades	• Nada a apontar

Período	2 de Maio de 2016 a 6 de Maio de 2016
Tarefas Realizadas	• Continuação do desenvolvimento do *backend* (*web services*)
Principais Dificuldades	• Nada a apontar

Período	9 de Maio de 2016 a 13 de Maio de 2016
Tarefas Realizadas	• Continuação do desenvolvimento do *backend* (*web services*)
Principais Dificuldades	• Nada a apontar

Período	16 de Maio de 2016 a 20 de Maio de 2016
Tarefas Realizadas	• Continuação do desenvolvimento do *backend* (módulo de *watermarking*) • Alterações na demo Android e inserção de novas campanhas
Principais Dificuldades	• Nada a apontar

Período	23 de Maio de 2016 a 27 de Maio de 2016
Tarefas Realizadas	• Continuação do desenvolvimento do *backend* (módulo de *watermarking*) • Alterações na demo Android (melhorias na gestão da ligação à Internet e mensagens de erro)
Principais Dificuldades	• Necessidade de andar constantemente a mudar de contexto entre desenvolvimento do *backend* e realização de alterações na demo Android

Período	30 de Maio de 2016 a 3 de Junho de 2016
Tarefas Realizadas	• Continuação do trabalho realizado na semana anterior
Principais Dificuldades	• Nada a apontar

Período	6 de Junho de 2016 a 10 de Junho de 2016
Tarefas Realizadas	• Continuação do trabalho realizado na semana anterior
Principais Dificuldades	• Nada a apontar

Período	13 de Junho de 2016 a 17 de Junho de 2016
Tarefas Realizadas	• Continuação do trabalho realizado na semana anterior
Principais Dificuldades	• Nada a apontar

Período	19 de Junho de 2016 a 24 de Junho de 2016
Tarefas Realizadas	• Continuação do trabalho realizado na semana anterior
Principais Dificuldades	• Nada a apontar

Período	27 de Junho de 2016 a 1 de Julho de 2016
Tarefas Realizadas	• Testes adicionais ao *backend* e correcção de defeitos detectados
Principais Dificuldades	• Nada a apontar

Anexo J - Proposta de Estágio

Tema

AudioSpot – Software de detecção de audio watermarks

Sumário

No âmbito deste estágio, o estagiário terá a oportunidade de adquirir e desenvolver conhecimentos na área de engenharia de *software* desenvolvendo *software* que integra com tecnologias existentes de *audio watermarking*, existindo várias fases de prototipagem de uma aplicação em Android para *smartphone* e *tablet*. A aplicação móvel é fundamentalmente orientada à deteção de áudio watermarks permitindo despoletar conteúdo específico (ex: campanhas publicitárias), no dispositivo do utilizador. Nesse sentido, o estagiário deverá investigar e aplicar várias técnicas do estado da arte que permitam à aplicação disponibilizar a melhor User Experience possível. Adicionalmente, será implementado o Back-End, com interface web, para gestão da lógica do negócio (ex: criação das campanhas e aplicação das áudio watermarks). Entre as várias versões do protótipo, deverá ainda ser levado a cabo testes com vista à identificação de problemas e alterações a realizar no protótipo.

Âmbito

A WIT desenvolve *software* para Operadores de Telecomunicações, tais como o Grupo Vodafone, Telefónica, Orange, Deutsche Telekom, Singtel, Oi Brasil, Centurylink (USA), TeliaSonera, Unitel. No âmbito da estratégia de criação de produto da WIT, pretende-se analisar a tecnologia de *audio watermarking* com vista à criação de um produto inovador.

Este estágio implica, numa fase inicial, trabalho de pesquisa e análise das soluções existentes no mercado, assim como a implementação de protótipos que permitam explorar e validar as potencialidades e aplicabilidades da tecnologia de *audio watermarking*. Estes protótipos serão aplicações Android que possam ser testadas em dispositivos smartphone. Findo o processo de avaliação da tecnologia de áudio watermarking, serão exploradas as diversas aplicabilidades da tecnologia sob o ponto de vista de negócio. Será implementado um *backoffice* web que permita processar ficheiros multimédia de forma a incluir *watermarks* na faixa de áudio e dar suporte à lógica de negócio (ex: criação de campanhas de marketing).

Por último, será desenvolvido um SDK que permita a integração por outras equipas da WIT Software.

Pretende-se que o estagiário desenvolva uma solução holística seguindo as boas práticas de engenharia de *software* bem como a elaboração de documentação de especificação e arquitetura do *software desenvolvido*.

É expectativa que este estágio possa contribuir para a experimentação de novas técnicas e funcionalidades a desenvolver posteriormente como produto final.

Objectivos

O presente projeto pretende atingir os seguintes objetivos:

- Análise relativa à técnica de *audio watermarking*
- Prototipagem rápida de uma aplicação móvel (Android)
- Especificação e desenvolvimento iterativo de uma plataforma web (*backoffice*)
- Especificação e desenvolvimento de um SDK para aplicações Android
- Realização de testes ao software desenvolvido
- Elaboração do relatório final de estágio

Programa de trabalhos

O estágio consistirá nas seguintes atividades e respetivas tarefas:

- ***T1*** – *Levantamento e análise* – Estudo e análise relativo ao *audio watermarking*, casos de uso, possíveis parceiros e soluções competidoras.
- ***T2*** – *Prototipagem* – implementação de várias versões de protótipos de software para dispositivos Android incluindo testes a várias soluções encontradas.
- ***T3***– Desenvolvimento de *backoffice* web – especificação, implementação e testes de um backoffice web-based.
- ***T4*** – Desenvolvimento de SDK para plataformas móveis – especificação, implementação e testes de um módulo de deteção de *audio watermarks* para aplicações desenvolvidas para o sistema Android.
- ***T5*** –*Relatório de Estágio* – elaboração do relatório final de estágio.

Calendarização das tarefas

O plano de escalonamento dos trabalhos é apresentado em seguida.

Tarefas	NOV		DEZ		JAN		FEV		MAR		ABR		MAI		JUN	
T1																
T2																
T3																
T4																
T5																
Metas		M1	M2			M3							M4			M5 M6

Resultados

Os resultados do estágio serão consubstanciados num conjunto de documentos a elaborar pelo estagiário de acordo com o seguinte plano:

M1: Documento de levantamento e análise das soluções encontradas
M2: Documento de casos de uso possíveis e competidores para o *audio watermarking*
M3: Prótotipos para Android e iOS
M4: Plataforma web
M5: Módulo para Android
M6: Documento com relatório final

Local de trabalho

O local de trabalho será no escritório da WIT Software em Taveiro, Coimbra. O estágio será remunerado. O estagiário terá ao seu dispor os equipamentos necessários para desempenhar as suas tarefas. Além da remuneração, o estagiário poderá ser convidado para receber formação na WIT Academy.

Metodologia

O processo de desenvolvimento seguirá o modelo implementado na WIT, baseado num modelo iterativo e incremental.

A documentação e protótipos serão também sujeitos a uma validação e aperfeiçoamento constante, ao longo do tempo em que serão produzidos, dependendo do feedback e conclusões retiradas pelo estagiário.

Orientação

ISEC:

Fernanda Correia (fernanda@isec.pt)
Cargo: Professora Adjunta

Entidade de Acolhimento:

Nuno Carvalho (nuno.carvalho@wit-software.com)

Cargo: Head of Competence Center

Caracterização e Remuneração

- Data de início – Início de Novembro
- Data de fim – a definir
- Horário – a definir
- Tipo de regalias oferecidas – remunerado
- Tipo de formação oferecida aos estagiários – a identificar no âmbito do WIT Academy

Bibliografia

[1] "Number of available apps in the Apple App Store from July 2008 to June 2016," Statista, [Online]. Available: http://www.statista.com/statistics/263795/number-of-available-apps-in-the-apple-app-store/. [Acedido em 24 Junho 2016].

[2] "DEIS - Página Ofiial - Departamento de Engenharida de Informática e de Sistemas," [Online]. Available: http://www.deis.isec.pt/DEIS.aspx. [Acedido em 6 Junho 2016].

[3] "DEIS - Página Oficial - Mestrado em Informática e Sistemas," [Online]. Available: http://www.deis.isec.pt/curso_mei.aspx?view=0. [Acedido em 6 Junho 2016].

[4] "WIT Software – Press Kit – Company Profile," [Online]. Available: https://www.wit-software.com/wp-content/uploads/2016/02/company_profile_EN.pdf. [Acedido em 6 Junho 2016].

[5] "Audio Watermarking," Novembro 2015. [Online]. Available: http://www.musictrace.de/technologies/watermarking.en.htm.

[6] G. Elert, "Frequency Range of Human Hearing," 2004. [Online]. Available: http://hypertextbook.com/facts/2003/ChrisDAmbrose.shtml. [Acedido em Novembro 2015].

[7] K. V. Goenka e P. K. Patil2, "Overview of Audio Watermarking Techniques," *International Journal of Emerging Technology and Advanced Engineering,* vol. 2, 2012.

[8] "SoundPays - Site oficial," [Online]. Available: http://www.soundpays.com/. [Acedido em 1 Dezembro 2015].

[9] "Digimarc - Site oficial," [Online]. Available: https://www.digimarc.com. [Acedido em 1 Dezembro 2015].

[10] "Cifrasoft - Site oficial," [Online]. Available: http://www.cifrasoft.com. [Acedido em 1 Dezembro 2015].

[11] R. Holly, "Second Screen apps — what are they and why would you want to use one?," Android Central, 2015 Fevereiro 2015. [Online].

Available: http://www.androidcentral.com/second-screen-apps-what-are-they-and-why-would-you-want-use-one. [Acedido em 15 Junho 2016].

[12] "Intrasonics - Site oficial," [Online]. Available: http://www.intrasonics.com/. [Acedido em 1 Dezembro 2015].

[13] "LISNR - Site oficial," [Online]. Available: http://lisnr.com. [Acedido em 1 Dezembro 2015].

[14] "Prontoly - Site oficial," [Online]. Available: http://www.prontoly.com/. [Acedido em 1 Dezembro 2015].

[15] "GitHub - SonicNet Repository," [Online]. Available: https://github.com/borismus/sonicnet.js. [Acedido em 1 Dezembro 2015].

[16] "Fingerprinting," MusicBrainz, [Online]. Available: https://wiki.musicbrainz.org/Fingerprinting. [Acedido em 27 Junho 2016].

[17] "Monterosa - Case Studies - Channel4 Horse Tracker," [Online]. Available: http://www.monterosa.co.uk/cases/horse-tracker. [Acedido em 15 Dezembro 2015].

[18] "Monterosa - Case Studies - TF1 QI Test," [Online]. Available: http://www.monterosa.co.uk/cases/qi. [Acedido em 15 Dezembro 2015].

[19] "eBay Announcements - Watch With eBay," [Online]. Available: http://announcements.ebay.com/2011/11/new-%E2%80%9Cwatch-with-ebay%E2%80%9D-feature-for-the-ipad-lets-you-browse-and-buy-stuff-from-your-favorite-tv-shows%E2%80%94while-you-watch-tv/. [Acedido em 15 Dezembro 2015].

[20] "Intrasonics – Markets & Uses - Solo Goodiebag," [Online]. Available: http://www.intrasonics.com/solo-goodiebag/. [Acedido em 15 Dezembro 2015].

[21] S. Deterding, R. Khaled e L. E. N. Dixon, "Gamification: Toward a Definition," em *CHI Conference on Human Factors in Computing Systems*, Vancouver, 2011.

[22] "The Second Screen Comes To The Movies With App-Enhanced Film, "App"," [Online]. Available: http://www.fastcocreate.com/1682579/the-second-screen-comes-to-the-movies-with-app-enhanced-film-app. [Acedido em 15 Dezembro 2015].

[23] "LISNR – Use Cases - Budweiser Made in America," [Online]. Available: http://lisnr.com/use-cases/live-events. [Acedido em 15 Dezembro 2015].

[24] "Chrome WebStore - Google Tone," [Online]. Available: https://chrome.google.com/webstore/detail/. [Acedido em 15 Dezembro 2015].

[25] "Hasbro - Furby Boom," [Online]. Available: http://www.hasbro.com/pt-pt/brands/furby/toys-games. [Acedido em 15 Dezembro 2015].

[26] "SoundLogin - Site oficial," [Online]. Available: https://www.soundlogin.com. [Acedido em 15 Dezembro 2015].

[27] "Mood Media - Shazam In-Store," [Online]. Available: http://us.moodmedia.com/shazam/. [Acedido em 15 Dezembro 2015].

[28] "Cinime - Site oficial," [Online]. Available: http://www.cinime.com/. [Acedido em 15 Dezembro 2015].

[29] "Google Play Store - SoundMeter PRO," [Online]. Available: https://play.google.com/store/apps/details?id=com.soundmeter.app. [Acedido em 18 Dezembro 2016].

[30] "Google Play Store - Sound Meter for Android," [Online]. Available: https://play.google.com/store/apps/details?id=com.splendapps.decibel. [Acedido em 18 Dezembro 2015].

[31] "BlippBuilder - Site oficial," [Online]. Available: https://blippar.com/en/solutions/self-service-solutions. [Acedido em Dezembro 2016].

[32] "Retrofit - Site oficial," [Online]. Available: http://square.github.io/retrofit/. [Acedido em 20 Abril 2016].

[33] "GSON," [Online]. Available: https://github.com/google/gson. [Acedido em 20 Abril 2016].

[34] "Picasso - Site oficial," [Online]. Available: http://square.github.io/picasso. [Acedido em 20 Abril 2016].

[35] "EventBus - Repositório de código," [Online]. Available: https://github.com/greenrobot/EventBus. [Acedido em 20 Abril 2016].

[36] R. Nystrom, “Decoupling Patterns - Event Queue,” Game Programming Patterns, [Online]. Available: http://gameprogrammingpatterns.com/event-queue.html. [Acedido em 30 Abril 2016].

[37] “Android Developers - Processes and Threads,” [Online]. Available: http://developer.android.com/guide/components/processes-and-threads.html. [Acedido em 30 Abril 2016].

[38] “Android Studio : The official IDE for Android,” Google, [Online]. Available: https://developer.android.com/studio/index.html. [Acedido em 20 Abril 2016].

[39] B. Linders, “Pivoting when Using Lean Startup for Product Development,” 4 Julho 2014. [Online]. Available: http://www.infoq.com/news/2013/07/pivoting-product-development. [Acedido em 25 Abril 2016].

[40] “Balsamiq Mockups - Site oficial,” [Online]. Available: https://balsamiq.com/. [Acedido em 25 Abril 2016].

[41] “Angular JS - Site oficial,” [Online]. Available: https://angularjs.org/. [Acedido em 5 Maio 2016].

[42] “MEANJS - Site oficial,” [Online]. Available: http://meanjs.org. [Acedido em 10 Maio 2016].

[43] “NodeJS - Site oficial,” [Online]. Available: https://nodejs.org. [Acedido em 10 Maio 2016].

[44] “ExpressJS - Site oficial,” [Online]. Available: http://expressjs.com. [Acedido em 10 Maio 2016].

[45] “MongoDB - Site oficial,” [Online]. Available: https://www.mongodb.org. [Acedido em 10 Maio 2016].

[46] “Spring Boot - Site oficial,” [Online]. Available: http://projects.spring.io/spring-boot/. [Acedido em 10 Maio 2016].

[47] R. Azuma, “A Survey of Augmented Reality,” *Teleoperators and Virtual Environments,* vol. 6, Agosto 1997.

[48] “Scrum Institute - Scrum Effort Estimations,” [Online]. Available: http://www.scrum-institute.org/Effort_Estimations_Planning_Poker.php. [Acedido em 50 Maio 2016].

[49] "Hibernate - Site oficial," [Online]. Available: http://hibernate.org. [Acedido em 15 Maio 2016].

[50] "Java Persistence/Inheritance," [Online]. Available: https://en.wikibooks.org/wiki/Java_Persistence/Inheritance. [Acedido em 8 Maio 2016].

[51] E. Hieatt e R. Mee, "Repository Pattern," [Online]. Available: http://martinfowler.com/eaaCatalog/repository.html. [Acedido em 11 Maio 2016].

[52] S. Nunes e G. David, "Uma Arquitectura Web para Serviços Web," [Online]. Available: https://repositorio-aberto.up.pt/bitstream/10216/281/2/51002.pdf. [Acedido em 8 Maio 2016].

[53] "Introduction to Spring Web MVC framework," [Online]. Available: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html. [Acedido em 20 Maio 2016].

[54] "Swagger - Site oficial," [Online]. Available: http://swagger.io. [Acedido em 20 Maio 2016].

[55] "Swagger UI - Petstore (exemplo oficial do Swagger)," [Online]. Available: http://petstore.swagger.io/. [Acedido em 20 Maio 2016].

[56] "JSON Web Tokens," [Online]. Available: https://jwt.io/. [Acedido em 24 Maio 2016].

[57] "JJWT - Site oficial," [Online]. Available: https://github.com/jwtk/jjwt. [Acedido em 20 Maio 2016].

[58] D. Boeker, "Graph Coverage in Software Testing," [Online]. Available: http://www.tesseractapps.com/2015/graph-coverage/. [Acedido em 15 Junho 2016].

[59] "Apache JMeter - Site oficial," [Online]. Available: http://jmeter.apache.org/. [Acedido em 29 Maio 2016].