Faculdade de Engenharia da Universidade do Porto

Design e desenvolvimento de uma plataforma gráfica para síntese de anúncios de publicidade online

Sara Capela Pereira de Sousa

Versão Final

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Dr. André Monteiro de Oliveira Restivo
Co-orientador: Dr. Luís Filipe Pinto de Almeida Teixeira
Co-orientador: Eng. Pedro Bandim Faustino

23 de Julho de 2016

Resumo

As empresas que fornecem serviços/soluções *web*, especialmente de gestão de publicidade *online*, muitas vezes necessitam de realizar uma gestão pormenorizada dos vários *websites*: quer da própria empresa, quer de potenciais clientes. Neste projeto em concreto, as implementações de integração das unidades publicitárias podem ser diferentes consoante os domínios e podem gerar até incompatibilidades. É portanto crucial desenvolver o código HTML/CSS/JS por forma a que o funcionamento e aparência não dependam do *browser* de Internet utilizado, do ambiente onde são desenvolvidos, entre outros fatores. O estado atual da distribuição da publicidade *online* envolve manter uma arquitetura que seja flexível a modificações como por exemplo a alteração de parâmetros. Essas empresas são então obrigadas a alterar o código-fonte sempre que querem introduzir/modificar um anúncio e alertar os seus clientes (que possuem *websites*) dessas mudanças, de forma a que a simbiose anúncio-website seja cumprida.

Apesar da maioria das vezes serem pequenas modificações, como redefinição das dimensões do componente ou até o volume do som, o trabalho é desnecessariamente repetitivo e pouco prático. Em certos casos, este processo ainda é mais dificultado, uma vez que requer a alteração de parâmetros apenas a partir do código minimizado (código submetido a uma remoção de caracteres desnecessários do código-fonte sem alteração da funcionalidade). Este problema surge quando a empresa, devido à falta de conhecimentos técnicos, não sabe como modificar o código-fonte e criar um ficheiro minimizado. Portanto a empresa terá de aceder ao ficheiro minimizado que está em produção e modificá-lo.

O *website* desenvolvido permite otimizar e facilitar todo o processo da gestão dos anúncios, uma vez que passa a ser realizado recorrendo a uma plataforma gráfica, sem existir a necessidade de programação. A plataforma denominada "Ad Builder"possibilita a criação, personalização e o *deployment* de unidades publicitárias *online*. Este projeto foi proposto pela empresa Omibee para um cliente específico: a companhia Adklip.

Abstract

Companies that provide web services/solutions, especially online advertising management, often need to carry out a detailed management of several websites: either from the company itself or from potential clients. In this particular project, implementing advertising unit integration can be different depending on the domain and can even generate incompatibilities. It is therefore crucial to develop HTML/CSS/JS code so that advertisement operation and design do not depend on the Web browser or the environment in which they are developed, among other factors. The current state of the online advertising distribution involves keeping an architecture that is flexible to changes such as parameter modifications. These companies are then forced to alter the source code on every ad creation/modification and alert their clients (who own websites) of these changes, so that the ad-website symbiosis is fulfilled.

Although most of the times these are minor changes, such as the redefinition of the component's dimensions or audio volume, it is unnecessarily repetitive and unpractical work. In some cases, this process is even more difficult, since it requires that parameter modification be done from the minified code (code subjected to the removal of unnecessary characters of the source code without changing its functionality). This problem arises when the company, due to the lack of technical knowledge, does not know how to modify the source code and create a minified file. In that circumstance, the company will have to access the minified file and change it.

The presented website allows to optimize and facilitate the entire ad management process by using a graphic platform, without requiring any programming knowledge. The "Ad Builder" platform allows the creation, customization and deployment of online advertising units. This project was proposed by Omibee for a specific client: Adklip.

Agradecimentos

Este meu percurso foi apoiado por várias pessoas a quem quero agradecer:

À minha mãe Glória Capela, ao meu pai Francisco Sousa e à minha irmã Joana Sousa que sempre me apoiaram em todas as etapas da minha vida, nas decisões que tomei e que sempre representaram um grande exemplo de trabalho, força e responsabilidade. E sobretudo por todo o amor e carinho.

Ao meu namorado João Meira, por todo o apoio, carinho, motivação, paciência e compreensão. E agradeço por ter tido a oportunidade de realizar este curso ao seu lado, partilhando assim todos os momentos pelo caminho.

À minha amiga Teresa Teixeira por toda a amizade, apoio e cumplicidade.

À restante família por me acompanharem ao longo deste percurso.

A todos os meus amigos de Espinho e os que criei durante estes cinco anos pelos momentos agradáveis e de boa disposição.

Ao meu orientador André Restivo pela ajuda durante a escrita da dissertação.

Ao meu co-orientador Luís Teixeira pelos conselhos e orientação.

Ao meu orientador da empresa Pedro Faustino pela sua visão de negócios, conselhos e por todos os conhecimentos que me transmitiu durante todo o acompanhamento do meu trabalho.

Ao Engenheiro Luís Mendes por toda a sua disponibilidade e ajuda nas dificuldades técnicas e práticas que enfrentei durante a realização do projeto.

À Omibee pela oportunidade que me foi concedida em fazer parte da equipa, ao longo destes 4 meses.

"Sometimes it is the people no one can imagine anything of who do the things no one can imagine."

Alan Turing

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

Amazon EC2	Amazon Elastic Compute Cloud
Amazon RDS	Amazon Relational Database Service
Amazon S3	Amazon Simple Storage Service
AOL	America Online
API	Application Programming Interface
AT&T	American Telephone and Telegraph
AWS	Amazon CloudFront
CDN	Content Delivery Network
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DNS	Domain Name System
EC2	Amazon Elastic Compute Cloud
GIF	Graphics Interchange Format
GNN	Global Network Navigator
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JS	Javascript
JSON	JavaScript Object Notation
MVC	Model-view-controller
ORM	Object-relational Mapping
OSI	Open Systems Interconnection
PHP	Hypertext Preprocessor, originalmente Personal Home Page
PPC	Pay-per-click
S3	Amazon Simple Storage Service
SQL	Structured Query Language
SVG	Scalable Vector Graphics
TCP/IP	Transmission Control Protocol/Internet Protocol
URL	Uniform Resource Locator
UTC	Universal Time Coordinated
VAST	Video Ad Serving Template
WWW	*World Wide Web*

Capítulo 1

Introdução

1.1 Contexto

Todos os dias milhões de pessoas utilizam a Internet quer seja para pesquisar informação, aceder às redes sociais ou para uso genérico [7]. O acesso à maioria dos *websites* implica a visualização de um anúncio cujo objetivo é apelar ao utilizador para comprar um determinado produto ou usar um dado serviço. Com o intuito de aumentar o lucro produzido pelos *websites*, muitos destes têm um ou mais espaços dedicados à publicidade [7]. O papel da publicidade *online* tem vindo a crescer nos últimos anos [8], uma vez que fornece informação sobre os produtos de uma forma fácil, acessível e barata a um vasto número de utilizadores. Dependendo do objetivo do anúncio, este pode aumentar o número de vendas, a lealdade do consumidor perante a marca, a familiarização de um determinado produto, entre outros [8].

Durante a última década as diversas mudanças na tecnologia – *big data*, *cloud computing* e novos métodos analíticos – permitiram a criação de aplicações que exploram mais profundamente as características das diferentes camadas dos modelos de arquitetura e de comunicações [9]. Ou seja, com o aumento da complexidade e do tamanho das redes foi necessário uma maior atenção na implementação e uma manutenção mais exaustiva [10]. Estas tecnologias surgem da otimização das camadas inferiores das *stacks* de arquitetura. Do ponto de vista da aplicação, o facto de existir uma infraestrutura altamente otimizada e fornecida através de serviços pagos, fiáveis e com disponibilidade elevada, possibilitou um aumento das oportunidades de negócios para pequenas empresas [10]. Estas oportunidades vão desde o produto final – servido ao utilizador – a camadas de *middleware* que integram serviços distintos em plataformas específicas a certos domínios. Ou seja, um individuo que não tenha conhecimentos aprofundados de programação em *Web* pode usufruir de ferramentas textuais e gráficas *online* que são dedicadas a um domínio particular, como por

exemplo *marketing*. Dessa forma, as empresas hoje em dia quando desenvolvem plataformas conseguem focar-se mais na esfera de influência onde atuam [11] (por exemplo moda, astronomia, publicidade, entre outros), adotando assim uma visão de mais alto nível e por isso mais abstrata. O pensamento do *developer* é no serviço ponto-a-ponto a fornecer e não nos componentes físicos [10].

Na área da publicidade *online* este novo paradigma é especialmente concorrido na medida em que apresenta às empresas a oportunidade de atacar o problema de forma distribuída. Isto é, cada empresa focar-se-ia num aspeto muito específico da área, fornecendo à camada superior um serviço.

1.2 Motivação

Os atores principais da publicidade, *online* ou *offline*, são o anunciante (*advertiser*) e o editor (*publisher*), os quais representam a oferta e a procura, respetivamente. O editor possui o espaço publicitário e a audiência, e o anunciante possui a publicidade (mensagem). No entanto o ecossistema da publicidade *online* é mais complexo e envolve muitos mais atores, entre eles as *video advertising networks* [12].

As *video ad networks* agregam a oferta de vários editores (normalmente pequenos e médios editores) e vendem esse inventário aos anunciantes. A gestão da oferta e procura é feita pela *video ad network*, através de um servidor de publicidade, onde o operador é responsável por inserir e configurar campanhas publicitárias [13].

As campanhas publicitárias consistem em vídeos publicitários que são oferecidos por dezenas de anunciantes através de centenas de servidores de publicidade. A *video ad network* é então responsável por publicar esses vídeos nos *websites* dos vários editores, através de unidades publicitárias que sejam inovadoras e que chamem a atenção do visitante [14].

A Figura 1.1 representa todo o percurso de distribuição das unidades publicitárias até ao utilizador final.

Todavia a integração das unidades publicitárias pode ser um problema, na medida em que podem surgir incompatibilidades relativamente ao *browser* de Internet utilizado, do ambiente onde são desenvolvidos, entre outros fatores. Para que estas incompatibilidades sejam resolvidas, a *video ad network* tem que reconfigurar as definições do produto (dimensões do componente, volume do som, entre outros).

Num ambiente empresarial, uma *video ad network* está sujeita a "stress de concorrência". Isto é, tem que constantemente expandir e melhorar os métodos de responder às necessidades dos clientes como reação aos padrões das outras empresas. É portanto extremamente

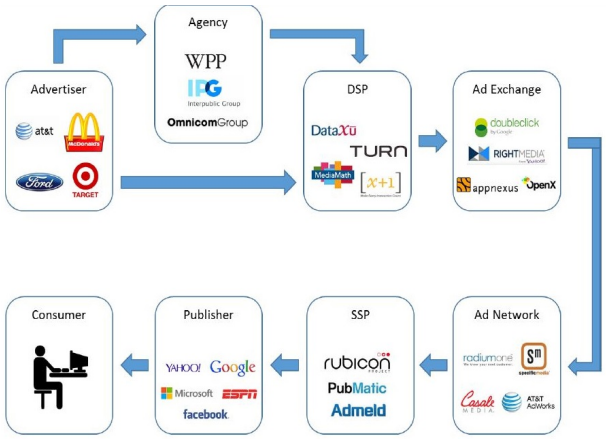

Figura 1.1: Processo de Distribuição de Publicidade [1]

valorizado por uma *video ad network* dispor de mecanismos que permitam uma adaptação mais rápida e um tempo de resposta mais competitivo face às nuances do próprio formato da publicidade *online* [15].

Cada unidade publicitária tem as suas próprias especificações e características e como tal é necessário modificar os seus parâmetros. Como estas alterações são efetuadas através de interação com o código-fonte, este torna-se um processo altamente especializado e moroso. Contudo este procedimento pode ser otimizado com o uso de uma plataforma que as implemente através de uma interface gráfica. Propõe-se então o desenvolvimento desta plataforma.

1.3 Enquadramento

Esta Dissertação surgiu através de uma proposta da empresa Omibee que se dedica ao design e desenvolvimento de serviços *web* e à criação de sistemas informáticos para manipulação e apresentação de informação. A empresa disponibiliza soluções para clientes do mundo inteiro e em múltiplas indústrias [16].

Este produto foi desenvolvido para um cliente específico - a Adklip. A Adklip é uma empresa composta por profissionais da área das tecnologias de publicação de anúncios, com mais de 10 anos de experiência de trabalho direto com *advertisers* e *publishers* no ecossistema digital. A plataforma da Adklip permite que *publishers* encontrem soluções inovadoras e programáticas para aumentar as suas receitas.

A Omibee colabora com a Adklip através do desenvolvimento de soluções que integram anúncios de vídeo, nativos e de impacto elevado. Uma dessas soluções inclui uma aplicação

web de filtragem de anúncios de alta disponibilidade, alta escalabilidade e baixa latência. A Omibee também fornece apoio técnico e consultoria à Adklip, implementado soluções em toda a cadeia de valor [17].

Neste contexto a Adklip representa uma *video ad network* e a Omibee uma empresa que lhe apresenta uma solução que passa pelo desenvolvimento de uma plataforma de gestão de anúncios publicitários. Esta solução advém da ideia de melhorar todo o processo, tornando-o automático, ao contrário da anterior que exigia conhecimentos de programação e em que o processo era moroso e requeria suporte técnico.

A Adklip fornece uma solução para a exibição de *video ads* em *websites* através de um *player* de anúncios publicitários. Este *player* está implementado a partir de um ficheiro Javascript que para além de incluir os mecanismos para a reprodução dos anúncios, também contém os seus parâmetros configuráveis, como a altura, a largura, o volume, entre outros.

Existem três *templates*[1] de reprodução de anúncios [18], tal como é ilustrado na Tabela 1.1.

Template	**Descrição**
Infeed	Um *player* nativo que fornece conteúdo de vídeo relevante sob a forma de *pre-roll* e é inserido no conteúdo geral do *website* ou no conteúdo de um artigo
Inpage	Um *player* que reside dentro do conteúdo editorial e reproduz o anúncio após ser ativado através de *scrolling*
Slider	Um *player* que desliza para o canto inferior do *website* sem interromper o fluxo do utilizador

Tabela 1.1: Tabela dos tipos de *players* fornecidos pela Adklip

1.4 Objetivos

A plataforma a desenvolver é uma ferramenta que permite a criação, a personalização e o *deployment* para produção de unidades publicitárias *online*. O excessivo trabalho de adicionar e alterar cada componente para adaptar às especificações do *website* em causa será quebrado e substituído por uma transparência na implementação. A *video ad network*, responsável pela gestão dos vários *websites*, poderá selecionar e alterar os parâmetros relativos à unidade publicitária através de uma interface gráfica, em vez de recorrer ao código-fonte. Nessa interface gráfica estarão presentes escolhas pré-definidas e limitadas das características dos anúncios como o tipo de visualização ou até as suas dimensões.

[1]Consultar Glossário para obter o significado dos termos estrangeiros

As configurações da *video ad network* serão guardadas para gerar um novo ficheiro Javascript e que será posteriormente minimizado. A última versão desse ficheiro é enviado para a Content Delivery Network (CDN). Esta rede fornece conteúdo aos utilizadores finais de forma transparente. Permite dessa forma aumentar a rapidez da distribuição do conteúdo *web*, uma vez que utiliza técnicas como a replicação da informação [19]. O CDN a ser utilizado será o fornecido pela infraestrutura Amazon Web Services - Amazon CloudFront.

Uma interface gráfica aumenta o nível de abstração do sistema, tornando-o de mais simples utilização e independente do nível de especialização do utilizador. O utilizador não necessita de ter conhecimentos profundos quanto à implementação da integração dos anúncios com o *website*, o que diminui o risco de provocar danos e o medo de experimentar novas configurações.

A alteração de parâmetros das unidades publicitárias através de uma interface gráfica também permite visualizar em tempo real o aspeto do anúncio no *website*. É portanto imediatamente percetível o resultado final e é desenvolvida pelo utilizador uma noção de causa-efeito entre os parâmetros modificados e as mudanças áudio-visuais do conteúdo contribuindo assim para a sua aprendizagem [20, 21, 22].

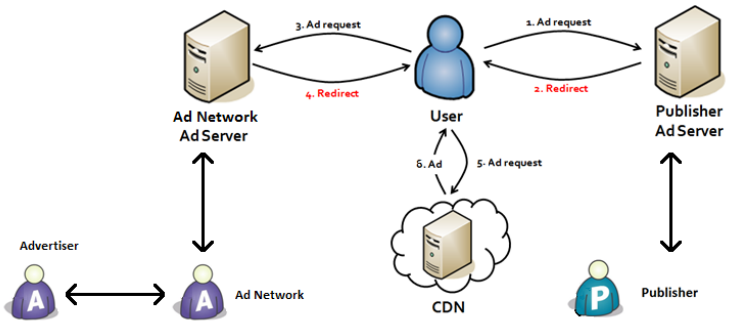

Figura 1.2: Arquitetura do Sistema de Publicidade [2]

A Figura 1.2 retrata as etapas efetuadas desde a oferta da publicidade até à sua visualização pelo utilizador final. Também é ilustrada a possibilidade do utilizador final obter a publicidade através do CDN, caso o conteúdo desejado esteja disponível neste meio. É exemplificado o caso em que existe apenas um elemento para cada ator.

Com esta solução pretende-se diminuir os tempos de *download* dos anúncios para o utilizador final. A entrega do conteúdo em redes CDN é realizada através de servidores estrategicamente colocados numa área próxima dos utilizadores finais. Assim a distância reduzida da viagem do conteúdo diminui a latência e a perda de pacotes. Da perspetiva dos utilizadores finais a experiência *online* será melhorada devido à redução do *jitter* durante a transmissão, à melhoria da qualidade global de *streaming* e ao aumento da disponibilidade do conteúdo [23]. Também será possível diminuir a redundância da perspetiva da

video ad network, pois o conteúdo passa a ser distribuído externamente, libertando a carga do servidor [24]. A natureza distribuída deste sistema diminui a probabilidade de erros, dado que caso um servidor falhe, outros terão uma cópia do conteúdo guardado na rede. Esta plataforma potencia também uma menor dependência com o suporte técnico devido à automatização do novo processo de *deploy* de unidades publicitárias.

1.5 Metodologia de Trabalho

Numa primeira instância ocorreu uma definição dos requisitos da plataforma através do uso de *User Stories* (disponíveis no Apêndice A). Também foram projetados *mockups* de todas as *interfaces* com o utilizador (ilustrados no Apêndice B). A aplicação desenvolvida recorre à *framework* de PHP Symfony (descrito na Secção 3.3.1) e emprega o Modelo Doctrine (descrito na Secção 3.2.2.1) para representação do modelo de dados. O servidor *web* escolhido foi o servidor HTTP Apache (descrito na Secção 3.4.2.1). Adicionalmente a plataforma comunica com um repositório privado no GitHub, onde foi recorrentemente arquivado para controlo de versões. Além disso este repositório permite a colaboração de uma equipa no mesmo projeto e faculta gestão de conflitos entre diferentes versões.

A implementação foi dividida em duas fases: a primeira consistiu no desenvolvimento local da aplicação e a segunda passou pela integração com o Amazon Web Services.

De modo a efetuar as configurações do ambiente de desenvolvimento local foi empregue uma máquina de virtual do Vagrant (descrito na Secção 3.6.1). Esta aplicação começou com a utilização de uma instância local de uma base de dados MariaDB (descrito na Secção 3.2.1.1). A gestão da base de dados ocorria através da ferramenta PhpMyAdmin (descrito na Secção 3.2.1.2).

Durante a primeira fase decorreu o desenvolvimento da interface gráfica da plataforma, empregando o Bootstrap (descrito na Secção 3.3.2). Após a sua conclusão, procedeu-se ao desenvolvimento da lógica de negócios. Esta componente não necessitou de medidas de escalabilidade, uma vez que foi projetada exclusivamente para um cliente: Adklip. Também foi projetada nesta fase a estrutura da base de dados e os objetos correspondentes no modelo de dados.

A segunda fase destinou-se ao *deployment* do código e à integração da aplicação com a infraestrutura Amazon Web Services (AWS, descrito na Secção 3.4.3). Para tal foram utilizados diversos serviços nomeadamente Amazon EC2, Amazon Elastic Beanstalk, Amazon S3, Amazon RDS, Amazon CodeDeploy e ainda Amazon CloudFront. O resultado desta migração é a passagem da execução da aplicação do servidor Apache (onde residia originalmente) para a sua utilização a partir de instâncias do Amazon EC2. Isto permite garantir

a disponibilidade do *website* e atribuir-lhe um domínio que é reconhecido por DNS (ad-builder.omibee.com). A base de dados MariaDB foi substituída por uma base de dados MySQL. Esta foi posteriormente migrada para o serviço Amazon RDS, tornando-se independente da aplicação em termos de fiabilidade. Estas duas componentes - Amazon EC2 e Amazon RDS - são unificadas e geridas pelo Amazon Elastic Beanstalk. O Amazon S3 é responsável por armazenar os ficheiros com as configurações dos anúncios publicitários e garantir espaço para as versões anteriores. Adicionalmente, de forma a garantir a disponibilidade dos ficheiros ao utilizador, o Amazon CloudFront estabelece uma abstração que protege a privacidade destes e mantém uma *cache*. O acesso aos ficheiros é realizado através de um URL do Amazon CloudFront. Por fim, o Amazon CodeDeploy permite automatizar o *deployment* da aplicação a partir de um repositório do GitHub.

Após a conclusão da integração com o AWS foram realizados testes unitários exaustivos para garantir o correto funcionamento da plataforma. Para tal recorreu-se a ferramentas como o CasperJS (descrito na Secção 3.5.2) e o Codeship (descrito na Secção 3.5.3).

1.6 Estrutura do Documento

Este documento encontra-se organizado em seis capítulos: Introdução, Publicidade Online, Tecnologias, Website Ad Builder, Validação do Website Ad Builder e ainda Conclusão.

No primeiro capítulo é realizada uma contextualização do mundo da publicidade *online* e são introduzidos alguns conceitos desse ambiente. É também introduzido o problema a discutir e é descrita a solução implementada.

O segundo capítulo apresenta uma breve introdução relativamente à história da publicidade *online*. De seguida são descritos em pormenor os intervenientes da publicidade *online* e os seus respetivos papéis e ainda são apresentados os diferentes tipos de publicidade *online*. Neste capítulo também se refere o mercado da publicidade, onde são indicados alguns produtos que permitem construir de raiz os anúncios e ainda personalizá-los.

No terceiro capítulo encontra-se uma descrição de cada tecnologia/ferramenta utilizada para a realização desta dissertação: desde linguagens *web*, serviços *web* como a Amazon Web Services até às plataformas de teste.

O quarto capítulo foca-se na apresentação pormenorizada da solução implementada onde são descritas todas as fases de desenvolvimento. Entre estas fases incluem-se o levantamento de requisitos, a implementação da interface gráfica, o desenvolvimento da lógica de negócios, a integração com o Amazon Web Services e ainda a implementação dos testes unitários. De forma a compreender todas estas etapas são apresentados diversos diagramas,

como o diagrama de Casos de Uso, o fluxograma das interfaces, o diagrama de componentes e o diagrama do Modelo Entidade Associação.

No quinto capítulo apresenta-se o resultado de dois questionários realizados com vista a validar o projeto. Um questionário foca-se na gestão dos anúncios e na interação entre as empresas Omibee-Adklip. O outro questionário foca-se na *user experience* da solução implementada.

Finalmente no sexto capítulo é realizada uma análise do projeto e são expostas considerações para trabalhos futuros.

Capítulo 2

Publicidade Online

2.1 História da Publicidade Online

A publicidade *online* nasceu em resposta ao aumento do número de compras *online* efetuadas pelos consumidores [25]. Desde então, tornou-se numa indústria de milhões de euros.

A primeira forma de publicidade foi o *Spamming* [26]. O termo original provém de uma marca de carne enlatada [27]. No entanto, este termo foi completamente revolucionado, após um *sketch* da comédia Britânica "Monty Python", onde um homem e a sua mulher encontram-se num restaurante e tudo o que pedem contém "*spam*". Mesmo quando o casal pedia algo sem *spam*, esse pedido era ignorado. Assim, as indústrias *online* utilizaram esta estratégia para inundar grupos da Internet e enviar *e-mails* não solicitados para propagar facilmente um determinado conteúdo. O primeiro *spam* comercial foi enviado pelos advogados Laurence Canter e Martha Siegel de forma a promover as suas práticas de advocacia de imigração. E desde então são enviados mais de 90 milhões de *e-mails spam* todos os dias. No entanto, este método em promover ou vender um produto é ineficaz [26].

Apesar do aparecimento do *spam*, a publicidade *online* apenas começou a ter impacto e a tornar-se a indústria que é hoje com a criação do primeiro anúncio *online* em 1993 [28]. Este anúncio foi vendido através do *website* GNN (Global Network Navigator) com a ação de um clique – estes tipo de anúncios foram posteriormente denominados de *banner ads*. O *website* GNN era um portal de informação *online* que consistia num centro de notícias e ao mesmo num catálogo de produtos [28]. Os clientes pediam *banner ads* com um formato fixo – 468x60px – para as suas companhias de publicidade. Num curto espaço de tempo, milhares de empresários e *advertisers* investiram neste método de alcançar o cliente. A empresa Yahoo! cobrava entre 30 e 100 dólares

para dispor estes *banners ads*. No entanto, os *banner ads* não eram eficientes, pois eram caros e o lucro era relativamente baixo. Eram portanto mais viáveis e eficazes os anúncios televisivos ou até os anúncios em papel [26].

Em 1994, a primeira companhia de revista *online* Hotwire idealizou o conceito de vender espaços de publicidade em grandes quantidades. O plano consistia em criar secções especiais nos *websites* para os *banners* serem disponibilizados. A empresa AT&T foi das primeiras companhias a comprar anúncios na Hotwire [28].

Numa altura onde existiam 16 milhões de utilizadores da Internet a nível mundial [29], a publicidade *online* surgiu como um método mais eficaz, barato e simples de atingir os consumidores. No entanto, no estado inicial os *advertisers* não entendiam quais as melhores abordagens a tomar. Apenas anos mais tarde, com estudos de mercado é que foram utilizados mecanismos mais eficazes.

Com a inflação, em meados de 2000, o interesse pela publicidade *online* decresceu. O mercado colapsou, o que resultou numa recessão. Inúmeras empresas faliram com a perda de procura. Esta época foi denominada de *dot-com bubble burst*.

Após o período *dot-com bubble burst*, apenas os motores de busca de conteúdo na Internet é que conseguiam obter lucros, tendo nomeadamente um aumento de crescimento do mercado em 2.3 mil milhões de dólares em 2003 [26].

A Google começou por ser apenas um motor de busca, que se focava em aperfeiçoar os algoritmos de pesquisa de forma a fornecer o melhores resultados possíveis. Porém, com a criação da Google AdWords em 2000, a empresa revolucionou a publicidade *online* – em vez de utilizarem *banner ads*, passaram a utilizar *text ads*. Quando um utilizador efetuava uma pesquisa, os *text ads* eram visualizáveis em cima ou ao lado dos resultados principais. Esta nova abordagem obteve sucesso, uma vez que a Google só exigia pagamento caso o utilizador carregasse no anúncio – reduzindo os custos – e porque era utilizado um algoritmo de atribuição da relevância do anúncio [28].

Entre 2001 e 2003, os anúncios *pop-up* foram introduzidos no mercado. No entanto, aborreciam os utilizadores dos *websites*. Para contrariar esse efeito, certos *browsers* como o Opera criaram *pop-up blockers*. Em 2005, a publicidade através de vídeos proporcionou aos mercados mostrar anúncios extremamente dinâmicos na Internet. Também nesse ano a DoubleClick desenvolveu métodos para mostrar anúncios consoante o tipo de público alvo. A empresa colecionava informação dos utilizadores através de *browser cookies* e com isso conseguiam direcionar os anúncios baseado no histórico de ações tomadas pelo utilizador [30].

Nos dias de hoje, com o grande poder das redes sociais como o Facebook, Twitter, Reddit,

entre outros, também começaram a existir anúncios nestas plataformas. Uma das grandes vantagens desta abordagem é que são mostrados anúncios focados nos gostos e interesses do público alvo [31]. Esta informação é conseguida através de métodos como *Data Mining*.

As redes sociais foram o maior impulsionador da publicidade *online* uma vez que os custos são inexistentes, tem um enorme número de consumidores e o conteúdo pode tornar-se facilmente viral. No entanto é difícil medir a eficiência da publicidade por este método, pois o número de "*likes*" ou "*follows*" pode não ser de facto convertido em vendas [28].

Em suma, desde 2004 a indústria da publicidade *online* conseguiu recuperar da recessão e para além disso veio a criar novos métodos de chamar a atenção das pessoas. Os anúncios começaram a utilizar ferramentas como o Flash e o Javascript para melhorar a interatividade e o aspeto visual. A publicidade *online* é portanto mais sofisticada e mais diversificada [26].

2.2 Intervenientes na Publicidade Online

Existem diversos intervenientes no mundo da publicidade *online*, tal como foi referido na Secção 1.2. É importante compreender a funcionalidade de cada um, uma vez que em determinados casos uma companhia ou uma empresa pode desempenhar diferentes papéis ao mesmo tempo, dependendo do tipo de publicidade online que realiza [12].

2.2.1 Advertiser

O anunciante ou *advertiser* é o cerne do ecossistema da publicidade. O seu papel é fornecer os anúncios publicitários e os parâmetros da campanha publicitária.

O grande objetivo de um *advertiser* é promover ou vender um determinado produto ou serviço, geralmente associado a uma empresa, a uma larga escala de possíveis compradores/clientes. Esta entidade também define os objetivos que pretende alcançar com cada anúncio.

O *advertiser* é o principal elemento que avalia o sucesso ou o insucesso de um anúncio publicitário, a partir do número real de vendas, em relação ao custo da produção da campanha publicitária. Dessa forma, o *advertiser* também examina relatórios com vista a verificar o cumprimento ou incumprimento dos objetivos iniciais de cada anúncio [12].

Os *advertisers* englobam desde pequenas e médias empresas até multi-nacionais como a Fiat, a NOS, a Super Bock ou a Coca-Cola.

2.2.2 Publisher

O editor ou *publisher* fornece o espaço - uma ou mais secções do seu *website* - onde a unidade publicitária irá ser disponibilizada e visualizável. Realizando um paralelismo com a publicidade *offline*, os *publishers* desempenham o mesmo papel que as estações televisivas, os jornais ou as revistas.

O grande objetivo do *publisher* é obter o máximo lucro possível para o seu *website* com a disponibilização dos espaços. Dessa forma, é do seu interesse mostrar anúncios apelativos para o público alvo do *website* e ainda otimizar a estratégia da entrega dos mesmos anúncios [7].

Em regra geral o *publisher* também é responsável por gerir a publicidade no seu *website*, na medida em que deve garantir que o tráfego acordado previamente com o *advertiser* seja cumprido. Muitos *publishers* trabalham com um elemento ou uma equipa dedicada à gestão dos anúncios - garantem que os requisitos dos *advertisers* sejam cumpridos e realocam os anúncios nos *websites* consoante o sucesso/insucesso da estratégica previamente definida.

Na circunstância em que um *publisher* não possui recursos suficientes ou tempo para vender os espaços publicitários eficazmente, este recorre frequentemente a contratos com *Ad Networks* [12].

Os *publishers* podem incluir jornais *online* como o Jornal de Notícias ou o Público, plataformas como o Youtube e o Sapo, blogues pessoais, entre outros.

2.2.3 Ad Networks

As *Ad networks* são a conexão entre a oferta e a procura de publicidade, uma vez que vendem os espaços publicitários dos *publishers* aos *advertisers*, tal como é possível visualizar na Figura 2.1. De forma geral uma *ad network* mantém relações de negócios com diversos *advertisers* e *publishers*. A gestão de todas as transações entre o conteúdo fornecido pelos *advertisers* e o local onde é disponibilizado é realizado recorrendo a um servidor de publicidade.

As *ad networks* têm experimentado um grande crescimento, pois fornecem a possibilidade de escalar o alcance que um *advertiser* deseja ao colmatar a falta de capacidade que cada *publisher* individual tem [25]. Assim a falta de tempo dos *advertisers* em procurar o *website* ideal entre milhões para mostrar o seu anúncio e a falta de recursos dos *publishers* em lidar com os requisitos das campanhas publicitárias é resolvido com o uso de *ad networks*. A distribuição da publicidade pelos *websites* é realizada de forma a obter a melhor compatibilidade possível entre produto-público, resultando num maior número de acessos ao

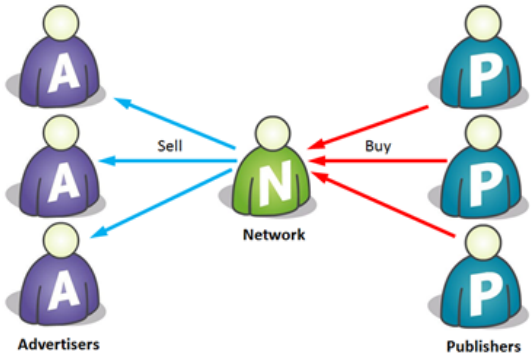

Figura 2.1: Relação entre os atores principais da publicidade [3]

anúncio e que por sua vez poderá traduzir-se em número de vendas. Na perspetiva dos *publishers*, o sistema permite lucros com o seu *website* que seriam impossíveis de outra forma.

A *ad network* recebe uma parcela da receita de cada anúncio publicitário. À medida que uma *ad network* cresce, esta conseguirá obter maior número de receitas e melhores taxas de sucesso de cada *advertiser* e *publisher* [12].

Entre as várias *ad networks* existentes incluem a Chitika, a Infolinks, a Clicksor, a Advertising.com, a Adblade, entre outras [32].

2.2.3.1 Video Ad Networks

Os *publishers* com vista a lucrarem mais com os espaços reservados à publicidade nos seus *websites*, começaram a experimentar novos formatos de transmissão das mensagens. Dessa forma, passaram a ganhar interesse pelos anúncios em formato vídeo, chamados de *video ads*. Esta forma inovadora de publicidade veio aumentar o lucro dos *publishers*, porque as comissões são superiores [13] e o formato é mais suscetível de se tornar viral.

Inúmeras *video ad networks* atingiram um grande sucesso e popularidade, nomeadamente o Youtube/Google, a AOL, a UnderTone, a Hulu, entre outros. Para alcançar esse patamar devem ser tidos em conta diversos fatores na abordagem das *video ad networks*. Conforme o alvo do *video ad* pode ser benéfico considerar uma *video ad network* que partilhe a mesma maneira de escolher o público alvo: com base no fator geográfico, comportamental ou mesmo contextual. Outro aspeto importante a considerar é garantir um suporte de visualização dos anúncios em diferentes plataformas. Os utilizadores hoje em dia utilizam a Internet a partir de diferentes dispositivos, por isso não basta garantir um bom funcionamento do *video ad* no computador mas também nos *smartphones* ou *tablets*. Para além disso também é valorizado o fornecimento de diferentes tipos de formatos de *video ads* (avi,

mp4, mkv...) e consequentemente diferentes tipos de visualização (*pop-up*, *pre-roll*...), uma vez que nem todos os formatos funcionam para todas as companhias de *video ads*. Um último aspeto e talvez o mais importante de todos é que uma *video ad network* deve fornecer às companhias de *video ads* análises estatísticas e ferramentas de medição necessárias para avaliar o desempenho dos seus *video ads* [15].

Existem diversas métricas para medir o desempenho dos *videos ads* e a escolha correta da métrica pode implicar um resultado positivo ou negativo. Ou seja, o mesmo *video ad* pode ser considerado um sucesso segundo uma determinada métrica e um fracasso noutra. Pode ser quantizada a percentagem da duração total do vídeo a que um utilizador efetivamente assistiu (se o utilizador visualizou 10 por cento do vídeo, 25 por cento ou se viu até a sua finalização). Pode ainda ser verificado por exemplo se o anúncio está a ser mostrado deliberadamente com som – ou se o utilizador decidiu desativar o áudio, ignorando a mensagem. Outras métricas incluem verificar se mais de metade dos pixeis do vídeo estão a ser apresentados na vista do *browser* (isto é, se o vídeo está de facto no campo de visão do utilizador ou se está obstruído por outros componentes) ou ainda no seu número de cliques e visitas de um determinado anúncio [33].

Em conclusão, um *video ad* que esteja sujeito à análise de um maior número de métricas fornece uma ideia mais precisa do seu desempenho. No entanto, nem sempre este desempenho na sua visualização se traduz no número de vendas do produto, pois é necessário garantir um público apropriado ao conteúdo do *video ad* [34]. Cabe então ao beneficiário destas estatísticas (o *advertiser*) decidir quais as métricas são mais importantes para o seu modelo. Para responder a isso de forma competitiva, as *video ad networks* devem disponibilizar diferentes escolhas (ou combinações destas).

2.3 Tipos de Publicidade Online

2.3.1 Affiliate Marketing

Neste tipo de publicidade *online* os *publishers* recebem uma comissão dos *advertisers* para cada venda efetuada do produto promovido no anúncio. O *publisher* publica os anúncios de um *advertiser* que lhe é associado(*affiliate*) com um determinado código que permite ao *advertiser* identificar que *website* foi visitado. Este sistema baseia-se na recompensa dada ao *publisher* pelo *advertiser* [12].

2.3.2 Banner

Método mais tradicional de publicidade *online* onde os *publishers* vendem uma secção especificada no *website*. O *advertiser* reserva ao *publisher* execuções do seu anúncio por um período pré-definido. É efetuado um acordo relativamente ao número de impressões (normalmente entre 100000 e 1000000) num espaço de tempo (normalmente num mês) entre as duas entidades. As impressões representam o número de vezes em que um determinado anúncio aparece no *website* independentemente do número de cliques que atinge [12]. Os *banners* podem abranger desde imagens estáticas e dinâmicas até conteúdo interativo como vídeos e áudio. Todavia hoje em dia este método tornou-se ineficaz, pois existem adições para os *browsers* que permitem "esconder" estes *banners* do utilizador (por serem considerados incómodos).

Existem três tipos de *banners* [35]:

- Estáticos – Contêm texto e grafismo simples. A imagem e o texto utilizados devem ser discretos. Este tipo de *banner* não deve ser perdido no meio do conteúdo do *website*, mas ao mesmo tempo deve conseguir integrar-se nele.
- Flash – São animados, tornando-os mais realistas e interativos e recorrem ao Flash. Podem utilizar efeitos especiais e som.
- Gifs Animados – *Banners* no formato GIF. Uma vez que têm um tamanho reduzido, não provocam atrasos no carregamento das páginas onde estão contidos, ao contrário dos Flash Banners.

2.3.3 Pay-per-clik (PPC)

O método *pay-per-click* (PPC) é associado à tecnologia de motores de busca, uma vez que a Google começou a utilizá-lo com a criação da Google Adwords. O *advertiser* paga ao *publisher* por cada clique efetuado no anúncio. Por norma, estes anúncios usam *text links* e são visualizados como parte dos resultados da pesquisa. O *advertiser* apresenta previamente a lista das palavras chave de cada anúncio para corresponder às possibilidades da pesquisa. [36]

2.3.4 Social Media

Esta forma de publicidade *online* encontra-se nas redes sociais como o Facebook, Twitter, Instagram, entre outros. Com este método é possível usufruir das capacidades inerentes

das redes sociais: têm um número elevado de utilizadores e o conteúdo pode tornar-se viral. Dessa forma, o alcance da mensagem pode ser muito significativo [37].

2.4 Mercado da Publicidade

2.4.1 Ad Builder

Com vista a facilitar o procedimento da criação, modificação e atualização dos anúncios publicitários foram criadas as Ad Builders. Estas plataformas oferecem ferramentas como a escolha do *template* e a personalização de cores, *fonts* e aspetos de cada anúncio. De um modo geral, utilizam uma interface gráfica para realizar o procedimento em detrimento de acesso direto ao código-fonte [20, 21, 22].

2.4.2 Produtos do Mercado

2.4.2.1 Celtra's Ad Creator

Tecnologia para visualização digital e para produção de vídeos de publicidade. O utilizador pode facilmente criar e personalizar o seu anúncio utilizando vídeos, URLs e *backgrounds* específicos para os textos. A Celtra's Ad Creator possibilita a criação de vídeos interativos com codificação de vídeo orientada à otimização para dispositivo alvo, velocidade e qualidade. Também poderá pré-visualizar o resultado nos diversos dispositivos como no computador ou no *smartphone* com tamanhos de ecrãs distintos. Com apenas um clique é enviado ao cliente um URL de pré-visualização da AdCreator. Este produto funciona numa vasta gama de dispositivos, ambientes, aplicações, *websites* e servidores.

Por fim, apesar do cliente não necessitar de aceder ao código-fonte para efetuar as alterações dos seus anúncios, a Celtra's Ad Creator fornece todos os componentes para adicionar ficheiros Javascript ou utilizar *frameworks*, eventos e interações através de uma API [22].

2.4.2.2 Ad Builder for HTML5

A Ad Builder para HTML5 fornece ferramentas que permitam construir anúncios de HTML5 diretamente na plataforma Sizmek MDX, cujo principal objetivo é a gestão de campanhas publicitárias.

A plataforma não exige conhecimento prévio de programação e oferece uma gestão intuitiva. É possível construir um anúncio de raiz ou ainda utilizar os *templates* disponibilizados com

os formatos de anúncios mais populares na indústria. O cliente pode ainda testar, pré-visualizar e efetuar *upload* do resultado.

A Ad Builder para HTML5 é compatível com os seguintes *browsers*: Google Chrome, Mozilla Firefox e Apple Safari [21].

2.4.2.3 Flashtalking

Com a importância do desempenho de uma campanha publicitária em mente, a Flashtalking oferece ferramentas para a criação e entrega de anúncios que focam nos interesses dos utilizadores. Ou seja, o grande objetivo desta plataforma é garantir a atenção do utilizador mostrando anúncios consoante o seu comportamento e perfil online.

A plataforma suporta anúncios dinâmicos Flash e HTML5, permitindo a sua visualização em computadores, *smartphones* e *tablets*. Caso seja utilizado HTML5, os *advertisers* podem disponibilizar os anúncios nesse formato ou ainda combinar Flash com HTML5.

A Flashtalking oferece soluções de escalonamento de versões de anúncios de acordo com eventos, reações à atividade competitiva e teste e ainda possibilita localizar campanhas multi-linguagem ou armazenar custos de produção e tempo de tráfego. Tudo isto é obtido editando os ficheiros Flash ou HTML5.

A edição de qualquer elemento da campanha em tempo real é possível com apenas alguns cliques, sem necessitar de conhecimentos aprofundados de programação. A interface permite atualizar, pré-visualizar e escalonar modificações sem necessitar especialistas ou longos tempos de execução de outras soluções [20].

Capítulo 3

Tecnologias

Para a realização deste projeto foi necessária a utilização de diversas ferramentas. Após uma fase de familiarização, decorreu a fase de implementação e por fim a fase de integração das várias plataformas.

3.1 Linguagens Web

3.1.1 HTML

HTML, sigla de Hypertext Markup Language, é a principal linguagem de estruturação/ marcação *web*. A codificação HTML foi projetada para exibir informações de forma simples, contendo texto, imagens, vídeos, som e animações em diferentes plataformas na Web.

A linguagem HTML é constituída por um conjunto de símbolos *markup* e cada um destes referencia um determinado elemento ou *tag*. A *markup* comunica ao *web browser* a estrutura do documento [38].

3.1.2 CSS

Cascading Style Sheets (CSS) é uma linguagem de folhas de estilo que permite definir a apresentação/aspeto das páginas *web* ou documentos escritos numa linguagem *markup* como HTML.

O código CSS fornece aos *web designers* um controlo completo sobre todos os elementos de uma página *web* e por conseguinte do aspeto/aparência da mesma. Para tal o *developer* utiliza *tags* específicas para incluir CSS no documento HTML, ou cria um *link* no

documento para um URL que contém o ficheiro CSS.

A sua principal vantagem é permitir a separação do aspeto e do conteúdo de um documento [39].

3.1.3 Javascript

Javascript (JS) é uma linguagem de programação interpretada ou de *scripting*, sendo assim executada no computador do utilizador e interpretada pelo *browser*.

Javascript juntamente com o HTML e o CSS é uma das principais linguagens Web. A sua utilização torna as páginas *web* mais interativas, uma vez que possibilita implementar animações, validar formulários, criar janelas modais, implementar *players* de música, entre outras funcionalidades [40].

As principais vantagens do Javascript vão desde a rapidez de execução do código na perspetiva do utilizador final, à poupança da largura de banda no servidor *web* até à facilidade de codificação. No entanto deteriora a segurança e a robustez do *website* [41].

3.1.4 PHP

PHP é uma linguagem de *scripting* do lado do servidor que foi criada com vista a melhorar as páginas *web*. O código PHP não é executado no computador do utilizador, mas no computador onde a página solicitada se encontra. O servidor *web* devolve os resultados ao utilizador e são exibidos no seu *browser*.

O código PHP pode ser embutido no documento HTML ou pode ser utilizado a partir da combinação de vários sistemas de *web templates* como o Smarty e *web frameworks* [42]. As principais *web frameworks* para PHP são CodeIgniter, CakePHP, Symfony, Zend e Laravel [43].

Entre os vários usos do PHP incluem-se o acesso a bases de dados, a implementação do *login* e *sign up* em *websites*, a criação e validação de formulários, a implementação de recuperação de *password*, entre outras.

3.1.4.1 Twig

Twig é um motor de *templates* utilizado pela linguagem PHP capaz de agilizar a lógica de visualização dos conteúdos recorrendo a uma sintaxe mais simples. Este tipo de componente serve para dinamizar a *view* (conceito introduzido na Secção 3.3.1.1).

É possível organizar hierarquicamente todos os ficheiros *twig*, uma vez que se pode aplicar herança de *layout* entre eles.

No entanto, os motores de *template* introduzem maior *overhead* a nível de volume de código e processamento, pois os *templates* têm que ser processados e só depois executados em PHP [44].

3.1.5 SQL

SQL, sigla de Structured Query Language, é uma linguagem de programação que possibilita obter e alterar informação de uma base de dados.

SQL utiliza *queries* que representam comandos que por sua vez permitem selecionar, inserir, atualizar, encontrar e localizar a informação pretendida [45].

3.2 Base de Dados

3.2.1 MySQL

MySQL é um sistema *open source* de gestão de base de dados relacional baseado na linguagem SQL.

MySQL executa virtualmente em diversos sistemas operativos nomeadamente Linux, UNIX e Windows. É suportada numa vasta gama de aplicações, no entanto, a sua utilização mais frequente é nas aplicações *web* e na publicidade *online* [46].

Para além da portabilidade e da compatibilidade entre várias plataformas, o MySQL oferece facilidade no manuseio, tem baixos custos e é um *standard* da indústria. Porém apresenta alguns problemas de estabilidade, devido à maneira como lida com determinadas funções. Além disso, quando o número de operações é demasiado elevado, o sistema tende a sofrer perdas de desempenho [47].

3.2.1.1 MariaDB

MariaDB é uma base de dados que surgiu como um *fork* do MySQL. Esta fornece um melhor desempenho, uma vez que executa as *queries* mais rapidamente, possui mais motores de armazenamento, e tem total compatibilidade até à penúltima versão do MySQL, tornando o processo de migração de uma para a outra simples. O próprio desenvolvimento da base dados apela mais à atenção de programadores, devido à sua natureza *open source* total:

todos os aspetos do seu desenvolvimento podem ser refeitos e debatidos num rastreador de *bugs* público [48].

3.2.1.2 PhpMyAdmin

PhpMyAdmin é uma ferramenta gratuita e *open source* escrita em PHP. Foi desenvolvida com o intuito de ajudar o *developer* a lidar com a administração de MySQL e MariaDB através do uso de um *browser*. A plataforma permite executar diversas tarefas entre as quais estão a criação, modificação e remoção de base de dados, tabelas, campos e linhas através de comandos SQL. Também faculta a gestão de utilizadores e das permissões [49].

3.2.2 ORM

ORM, sigla de Object-relational mapping, é um mecanismo que torna o processo de acesso e manipulação de objetos independente da fonte de informação destes.

Este mecanismo fornece aos *developers* um controlo e visão dos objetos ao longo do tempo que garante a abstração dos dados segundo diferentes perspetivas: das aplicações que os alteram, das fontes que os entregam e dos recetores que os requisitam. Esta visão deve ser consistente e resistente a mudanças por parte das camadas inferiores.

Um ORM fornece a gestão dos detalhes do mapeamento entre um conjunto de objetos e as correspondentes bases de dados relacionais, repositórios de informação, e outras fontes. Adicionalmente, encapsula o processo de interface objetos-dados de forma a permanecerem isolados dos *developers* e do seu código.

Soluções baseadas em ORMs beneficiam de uma flexibilidade em termos de manutenção de código, na medida em que permitem que equipas grandes utilizem o mesmo modelo de dados de forma consistente. O encapsulamento forte significa que apenas é necessário implementar as interfaces entre os objetos abstratos e os dados subjacentes. É portanto mais fácil gerir mudanças por parte da fonte de dados (incluindo API), bem como estender a aplicação em desenvolvimento para permitir novas tecnologias [50].

3.2.2.1 Modelo Doctrine

O Modelo Doctrine é um mecanismo de ORM que representa o *schema* da base de dados, na medida em que descreve a configuração completa da base de dados [51].

O Doctrine permite manipular e aceder aos dados da base de dados através da criação de objetos que descrevem as tabelas e a relação entre elas. Toda esta descrição da base dados é conseguida através de um conjunto de classes PHP que definem o comportamento do modelo.

Uma vez que são utilizados objetos, não é necessário recorrer a comandos SQL para manipular os dados [52].

3.3 Frameworks

3.3.1 Symfony

O Symfony é uma *framework full-stack* orientada a objetos que permite criar *websites* e aplicações *web* a partir de um conjunto de componentes PHP [53]. Alguns desses componentes são representados na Tabela 3.1 [54].

Componente	**Descrição**
Asset	Gere a geração de URLs e o versionamento de alguns *web assets*, nomeadamente ficheiros de CSS, JS e imagens
ClassLoader	Carrega as classes do projeto automaticamente caso sigam as convenções do *standard* PHP
Form	Fornece ferramentas para facilitar a criação, processamento e reutilização de formulários de HTML
HttpFoundation	Define uma camada orientada a objetos para a especificação de HTTP
HttpKernel	Fornece blocos de construção para uma criação rápida e flexível de estruturas baseadas em HTTP
Routing	Mapeia um pedido HTTP a um conjunto de variáveis de configuração
Security	Fornece uma infraestrutura para sistemas de autorização sofisticados
Yaml	Carrega e exporta ficheiros YAML

Tabela 3.1: Tabela de alguns Componentes do Symfony

As informações da base de dados relacional devem ser mapeadas para um modelo de objeto [52]. Para tal deve-se utilizar uma ferramenta ORM (Secção 3.2.2) e no caso do Symfony a mais utilizada é o Doctrine, anteriormente explicado na Secção 3.2.2.1.

O Symfony segue o paradigma Model-view-controller [55] que irá ser descrito na Secção 3.3.1.1.

Parte da facilidade desta *framework* reside na convenção da distribuição das pastas e dos ficheiros. A sua arquitetura é apresentada na Figura 3.1.

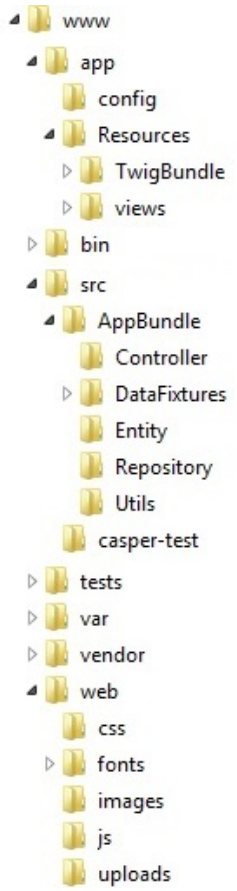

Figura 3.1: Arquitetura das pastas do Symfony

Na pasta *app* é possível encontrar os ficheiros com as configurações da base de dados, do ambiente de trabalho (produto, desenvolvimento ou teste) e dos *routings* das várias paginas *web*. Também contém todos os ficheiros *twig* dentro da pasta Resources. Estes ficheiros *twig* representam o conteúdo das páginas *web*.

A pasta src contém os Controllers que descrevem classes de PHP que lidam com a informação de um pedido de HTTP e devolvem uma resposta HTTP. A resposta HTTP pode abranger desde uma página HTML, um documento XML, uma imagem, um redirecionamento a um erro "404 - Página não encontrada"[56]. A pasta src também contém todas as entidades e repositórios que se relacionam com a base de dados. Esta estrutura de pastas é gerada automaticamente quando se utiliza um modelo de dados como o Doctrine (Secção 3.2.2.1). Na pasta Entity está presente um ficheiro para cada tabela da base de dados, no qual são apresentados todos os seus atributos e funções de acesso a esses mesmos atributos.

A pasta *web* inclui todos os *assets* do projeto - ficheiros Javascript, CSS, *fonts* e ainda imagens.

Por fim a pasta *bin* integra a consola, a pasta *var* inclui a *cache* e as informações detalhadas dos sucessos ou insucessos das execuções do código e a pasta *vendor* contém todos os elementos externos necessários para o projeto.

3.3.1.1 Paradigma Model-view-controller

O Symfony segue o paradigma Model-view-controller que é ilustrado na Figura 3.2.

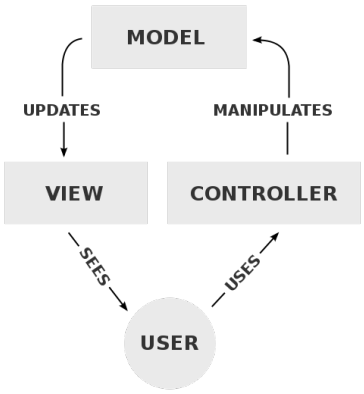

Figura 3.2: Diagrama das interações entre os três componentes do paradigma Model-view-controller [4]

Este paradigma consiste na divisão da aplicação em três componentes ou objetos separados conforme a sua relação com a informação. A componente do Model encapsula os dados e a lógica numa aplicação de *software*. O objeto Model não contém qualquer informação relativa à *interface* com o utilizador. O objeto View é a representação visual do Model, funcionando como canal cujo propósito é a reprodução da informação. Todos os elementos que representam a *interface* com o utilizador estão presentes nesta componente. Os vários elementos incluem *buttons*, *inputs* e *display boxes*. Por fim, o Controller é a ponte entre as outras duas componentes. Esta retrata a ação desejada pelo utilizador na medida em que processa e gere a interação com as *views* e o *model* [57].

Muitos *developers* adotaram o padrão model-view-controller com vista a reutilizar o código em projetos futuros e a reduzir significativamente o tempo de desenvolvimento de aplicações que utilizem *interfaces* com o utilizador [58].

No caso do Symfony a componente Model não é incluída por defeito. No entanto a Symfony Standard Edition, que é a distribuição mais utilizada a nível global [59], integra o Modelo Doctrine que foi apresentado na Secção 3.2.2.1.

3.3.2 Bootstrap

Originalmente criado por *developers* do Twitter, o Bootstrap tornou-se um dos *frameworks* de *front-end* mais populares do mundo. O Bootstrap utiliza HTML, CSS e JS e foi projetado para desenvolvimento de aplicações *web*. Uma vez que assenta no conceito de *design* responsivo, permite escalar facilmente e eficazmente os *websites* e aplicações para diversos dispositivos: *smartphones*, *tablets* e *desktops*.

O Bootstrap oferece um conjunto de componentes de CSS e HTML personalizados e *plug-ins* de jQuery, de modo a facilitar o desenvolvimento de *software* relativo à interação com o utilizador [60].

3.4 Serviços Web

3.4.1 Protocolo HTTP

O Protocolo HTTP, sigla de HyperText Transfer Protocol, é um protocolo utilizado na camada da aplicação do modelo TCP/IP e OSI para a comunicação entre sistemas distribuídos [61]. HTTP baseia-se numa comunicação cliente-servidor onde os clientes efetuam pedidos a um servidor HTTP e este responde no formato de uma mensagem, tal como é possível visualizar na Figura 3.3.

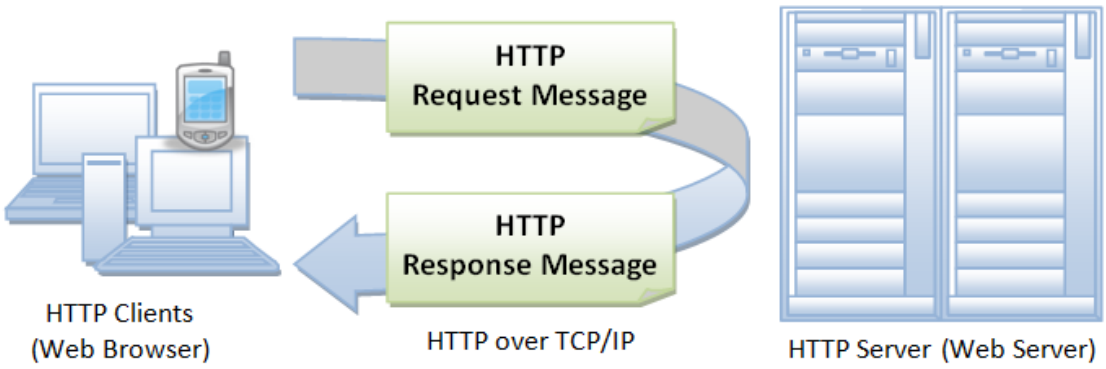

Figura 3.3: Intercâmbio de mensagens entre os clientes e o servidor utilizando o protocolo HTTP [5]

É um protocolo que não guarda informações sobre o estado, ou seja, um pedido não tem qualquer conhecimento sobre o conteúdo dos pedidos anteriores. Uma vez que a sua implementação permite flexibilidade a nível da representação e do tipo de dados, os sistemas poderão ser construídos independentemente dos dados a serem transferidos [5].

Este protocolo é a base da comunicação de informação na World Wide Web [61].

3.4.1.1 URL

URL, sigla de Uniform Resource Locator, é o endereço virtual de um determinado recurso informático disponível numa rede. Através deste caminho, o utilizador poderá ter acesso ao recurso, sendo este um ficheiro ou uma máquina ou até uma página *web*. No contexto desta Dissertação o URL é um *link* ou endereço de um *website*.

A estrutura de um URL é representado na Figura 3.4.

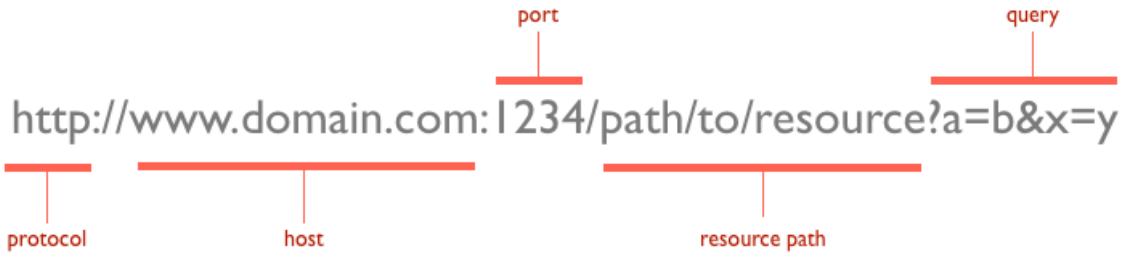

Figura 3.4: Estrutura de um URL [6]

Um URL é composto por um protocolo, que pode ser tanto HTTP, que é um protocolo de comunicação ou FTP, que é uma forma rápida de transferir ficheiros na Internet. Também contém um *host* (ou domínio), uma porta (que poderá ser omitida), o caminho para o recurso desejado e opcionalmente uma *query* [6].

3.4.2 Servidor Web

Um servidor *web* é responsável por receber e aceitar pedidos e por enviar as respostas a esses pedidos. De modo a realizar este processo, o sistema utiliza o protocolo HTTP [62]. Cada servidor *web* possui um endereço IP e na maioria das vezes um domínio. O acesso a um URL como `http://www.amazon.com/books` implica o envio de um pedido ao servidor *web* cujo domínio é amazon.com. O servidor solicita a página *books* e envia de volta para o *browser* do utilizador [63].

Todo o conteúdo que pode ser acedido através da Internet é entregue pelos servidores *web*. O conteúdo inclui desde documentos HTML, folhas de estilo CSS, multimédia (nomeadamente imagens, vídeos, música) e *scripts* de Javascript [64].

Existem dois tipos de servidores *web*: estáticos ou dinâmicos. Os primeiros consistem num computador (hardware) em junção com um servidor de HTTP (software), enquanto que os

segundos utilizam mais *software*, incluindo um servidor de aplicações e uma base de dados [65].

A designação "servidor *web*"pode aplicar-se tanto a computadores dedicados apenas a esta função como a aplicações que recebem e respondem a pedidos [62].

3.4.2.1 Servidor HTTP Apache

O servidor HTTP Apache é um servidor *open source* desenvolvido em 1995 pela Apache Software Foundation [66]. Originalmente criado com o intuito de executar apenas no sistema operativo Linux/Unix, no entanto foi posteriormente adaptado para funcionar noutros sistemas como Windows e Mac [62]. A tecnologia do Apache foi responsável pelo início do crescimento da World Wide Web [64].

Para além da sua robustez que permite lidar com volumes elevados de tráfego, também consegue servir diferentes tipos de conteúdo. Pode ser configurado de forma a responder às especificações de diversos ambientes, utilizando extensões e módulos para tal [64]. Outras características predominantes do Apache são a rapidez, a fiabilidade e a segurança [66].

O Apache é o servidor *web* mais utilizado a nível global, fornecendo o conteúdo a 50% dos *websites* ativos [67].

3.4.3 Amazon Web Services (AWS)

A Amazon Web Services é uma plataforma de computação em nuvem (*cloud*) onde são fornecidos serviços *online* para *websites* e para aplicações do cliente. Esta plataforma pertence à empresa americana Amazon [68].

A Amazon Web Services contém servidores em diversas regiões do mundo, nomeadamente nos Estados Unidos da América, Brasil, Irlanda, Singapura, Japão e Austrália [69].

Os serviços disponíveis pela AWS incluem armazenamento, base de dados, análise, segurança, aplicações empresariais e ainda serviços móveis [70].

De seguida será realizada uma descrição de cada um dos serviços utilizados nesta dissertação.

3.4.3.1 Amazon EC2

Amazon Elastic Compute Cloud ou Amazon EC2 ou apenas EC2 é um serviço que fornece capacidade de computação redimensionável na nuvem através da utilização de instâncias [71]. As instâncias de EC2 são servidores virtuais para desenvolver e executar aplicações na infraestrutura da Amazon Web Services. É possível executar até milhares de instâncias do servidor simultaneamente [72].

A interface do Amazon EC2 oferece um controlo completo dos seus recursos computacionais. Também disponibiliza vários tipos de instâncias com diferentes configurações do CPU, da memória, do armazenamento e da capacidade de rede necessária para a execução da aplicação. Para além disso, o *developer* poderá interagir com as instâncias, interrompê-las e reiniciá-las remotamente usando as APIs de serviços *web*.

O Amazon EC2 permite aumentar ou diminuir a sua capacidade à medida que os requisitos de computação são alterados, tudo numa questão de minutos, contribuindo para um escalonamento rápido.

Esta solução apresentada pela Amazon beneficia a componente de negócios, uma vez que o *developer* paga apenas a capacidade que utiliza. O Amazon EC2 também fornece aos *developers* ferramentas para construir aplicações resistentes a falhas [71].

3.4.3.2 Amazon Elastic Beanstalk

O AWS Elastic Beanstalk é um serviço de fácil utilização para *deployment* e escalabilidade de aplicações e serviços da *web* desenvolvidos com Java, .NET, PHP, Node.js, Python, Ruby, Go e Docker em servidores, como Apache, Nginx, Passenger e IIS.

O Elastic Beanstalk separa-se do Amazon EC2 através de uma camada de abstração. Este trata de estabelecer um ambiente onde é possível conter várias instâncias EC2 e opcionalmente uma base de dados ou ainda outras componentes da AWS como Elastic Load Balancer, Auto-Scaling Group e/ou Security Group. Elastic Beanstalk gere todos estes elementos, sempre que é atualizado o *software* presente na AWS.

Após o *upload* do código, o Elastic Beanstalk encarrega-se automaticamente da implementação, desde o provisionamento de capacidade, o balanceamento de carga e a escalabilidade automática até a monitorização do desempenho da aplicação. Ao mesmo tempo, o Elastic Beanstalk oferece um controlo total sobre os recursos da AWS que possibilitam a operação da aplicação.

Não há custos adicionais associados ao Elastic Beanstalk, uma vez que apenas se paga os recursos da AWS necessários para executar e armazenar aplicações [73].

3.4.3.3 Amazon S3

O Amazon Simple Storage Service também denominado de Amazon S3 é um sistema de armazenamento e de obtenção de dados *online*. O S3 consegue armazenar e aceder a qualquer volume de dados, a qualquer momento, de qualquer lugar na Web. Os dados estão organizados no Amazon S3 por *buckets*. É um sistema escalável que funciona a altas velocidades e a baixos custos [74]. A interface com esta componente da AWS realiza-se através da AWS Management Console.

3.4.3.4 Amazon RDS

O Amazon Relational Database Service (Amazon RDS) é um serviço de bases de dados relacional na nuvem que facilita a sua configuração, operação e escalabilidade. Este executa várias tarefas de rotina como provisionamento, *backup*, recuperação, deteção de falhas e reparação.

É possível replicar a base de dados do Amazon RDS para uma base de dados secundária como MySQL, PostgreSQL e Amazon Aurora de forma síncrona. A replicação torna-se útil para aumentar a disponibilidade e a fiabilidade e também agilizar, em termos de capacidade, nos casos em que uma base de dados tenha um uso intensivo de leitura.

O Amazon RDS disponibiliza seis mecanismos de bases de dados: Amazon Aurora, Oracle, Microsoft SQL Server, PostgreSQL, MySQL e MariaDB [75].

3.4.3.5 Amazon CodeDeploy

O AWS CodeDeploy faculta a automatização do *deployment* do código presente em qualquer instância, incluindo as do Amazon EC2 e as executadas localmente, a partir de repositórios do GitHub ou de *buckets* do Amazon S3. Uma vez que o processo é automatizado, diminui a ocorrência dos erros inerentes às operações manuais. Este serviço também facilita o lançamento rápido de novos recursos, ajuda a evitar tempo de inatividade durante o *deployment* de aplicações e lida com a complexidade de atualizar as aplicações.

O AWS CodeDeploy inclui outros serviços de configuração, gestão e integração como AWS Elastic Beanstalk, AWS CodePipeline, AWS CloudFormation e AWS OpsWorks [76].

3.4.3.6 Amazon CloudFront (CDN)

O Amazon CloudFront é um serviço de distribuição de conteúdo na *web* fornecido pela Amazon Web Services. Podem ser incluídos vários tipos de conteúdo: dinâmico, estático, *streaming* e conteúdo interativo [77]. A distribuição é realizada utilizando uma rede internacional de servidores [78], localizada perto dos utilizadores finais nos Estados Unidos, Europa, Ásia, América do Sul e Austrália. Quando um utilizador solicita um determinado conteúdo, esse pedido é automaticamente direcionado para o servidor mais próximo. Dessa forma garante-se uma distribuição com melhor desempenho, menor latência e ainda com elevadas taxas de transferência de dados. Este método de distribuição de conteúdo representa uma solução mais económica para as empresas, pois apenas a quantidade de conteúdo que é realmente distribuída representa um custo.

O Amazon CloudFront integra outros produtos da Amazon Web Services, nomeadamente a Amazon Simple Storage Service (S3) e a Amazon Elastic Compute Cloud (EC2). Porém, também pode ser utilizado sem os outros produtos através de uma API. O Amazon CloudFront também funciona com qualquer servidor que não seja da Amazon Web Services, desde que armazene as versões originais e definitivas de seus arquivos [77].

3.4.4 Git

O Git é um sistema de controlo de versionamento vastamente utilizado no desenvolvimento de *software*. Este assenta em garantias de um serviço distribuído com ênfase na integridade dos dados, na velocidade, e que suporta *workflows* distribuídos e/ou não lineares. Outra vantagem inerente ao versionamento é a possibilidade de restaurar versão anteriores do código, desfazendo efetivamente alterações nocivas ao projeto [79].

Existem três ações essenciais no Git para gestão de um repositório: *pull*, *push* e *commit*. Commit consiste em adicionar modificações do código no repositório local e eventualmente adicionar uma mensagem a descrever essas alterações. Push é a transferência do(s) último(s) *commit(s)* para o servidor remoto e por fim, a transferência das modificações do conteúdo no servidor remoto para o repositório local denomina-se Pull [80].

3.5 Ferramentas de Teste

3.5.1 PhantomJs

A definição real do PhantomJs é "*headless Webkit scriptable*". No entanto, traduzindo em termos mais simples, o PhantomJS é um *web browser* que não utiliza uma *interface* gráfica com o utilizador. Esta característica permite executar quaisquer interações de forma automática com uma página *web*, sem necessitar de abrir um *browser* convencional.

O PhantomJs faculta um rápido e vasto suporte nativo para vários *web standards*, nomeadamente seletores de CSS, JSON, Canvas, DOM handling e SVG [81].

3.5.2 CasperJS

CasperJs é uma ferramenta de *scripting* de navegação e um utilitário de testes desenvolvida para o PhantomJS, escrita em Javascript. Muitos *developers* recorrem a esta ferramenta com a finalidade de testarem a sua aplicação *web* de forma automática, uma vez que executa num nível acima do PhantomJs, estendendo as suas capacidades como *browser* [82].

3.5.3 Codeship

O Codeship possibilita a execução automática de testes e notifica o *developer* caso tenha ocorrido um insucesso. É uma plataforma que funciona de forma integrada com repositórios como os do GitHub. Sempre que é realizado um *push*, os testes são executados, segundo as configurações de teste previamente definidas. Os testes são realizados de forma paralela à codificação do projeto.

Para além da execução automática na plataforma *online*, o *developer* também poderá aceder por SSH ao ambiente de teste.

O Codeship suporta diversas linguagens (Ruby on Rails, Python, PHP, Java, Go e Node.js), bases de dados (PostgreSQL, MySQL, MongoDB, SQLite, etc) e ferramentas de *deployment* (Amazon Web Services e Heroku). Este último ponto é especialmente útil, pois permite que só as execuções do código bem sucedidas sejam enviadas para serviços de *deployment* como o EC2. Isto resulta numa redução de custos e impedindo que versões anómalas da aplicação cheguem a produtivo [83].

3.6 Outras Plataformas

3.6.1 Vagrant

O Vagrant é um *software* que permite criar e executar de forma simples máquinas virtuais, recorrendo a programas como o VirtualBox ou o VMware. Dessa forma um *developer* pode trabalhar num ambiente de desenvolvimento à sua escolha. O acesso às máquinas virtuais ocorre através de conexões SSH.

As máquinas virtuais são totalmente configuráveis e podem conter quaisquer programas instalados. Caso todo o ambiente de desenvolvimento seja baseado em máquinas virtuais personalizadas - com as próprias configurações - um *developer*, após instalar o Vagrant, apenas necessita executar um único comando para correr o ambiente. Estas configurações são guardadas em *scripts* que garantem, entre outras, que as instalações das ferramentas, linguagens e *frameworks* necessárias à aplicação são executadas e permanecem atualizadas. O Vagrant utiliza um ficheiro onde o *developer* pode configurar a chamada destes *scripts*. Estas características contribuem para uma facilidade de manuseio e para uma portabilidade total de código.

O Vagrant compartilha um diretório entre o *host* (a máquina real) e o *guest* (a máquina virtual), o que implica que qualquer alteração realizada numa máquina é imediatamente refletida na outra. Dessa forma o *developer* poderá programar no seu editor de texto em Windows ou Mac e executar o projeto em Linux, por exemplo [84].

Em suma, a grande vantagem do Vagrant é a possibilidade de replicar máquinas virtuais personalizadas que executem em qualquer ambiente.

3.6.2 Adobe Ilustrator

O Adobe Illustrator é uma ferramenta desenvolvida pela Adobe para a criação de imagens vetoriais nomeadamente gráficos, logótipos, ícones, diagramas e ilustrações complexas para impressão, Web, conteúdo interativo, vídeos e dispositivos móveis.

Uma imagem vetorial poderá ser infinitamente ampliada ou minimizada, sem sofrer quaisquer perdas de qualidade. [85].

Capítulo 4

Website - AdBuilder

O *website* desenvolvido, denominado "Ad Builder", surge como uma ferramenta que permite a criação, a personalização e o *deployment* de unidades publicitárias *online*. É importante definir o conceito de unidade publicitária e de criação destas mesmas. Neste contexto o anúncio não representa um objeto multimédia, mas sim, um ficheiro Javascript que agrega informação de estilo, bem como referências para a fonte do anúncio. Entende-se então que por "criar"uma unidade publicitária consiste em especificar parâmetros que permitem obter, formatar e exibir o conteúdo multimédia da publicidade.

Este projeto foi proposto pela empresa Omibee, que desenvolveu a especificação base que serve como *template* para unidades publicitárias. Esta especificação implementa o *player* da Adklip que inclui mecanismos de obtenção de conteúdo multimédia e uma discriminação dos parâmetros configuráveis. Os parâmetros principais são descritos na Tabela 4.1. Os outros parâmetros foram omitidos uma vez que não são utilizados no *player* atual, sendo mantidos por questões de compatibilidade. Estes definiam um conjunto de credenciais de acesso a servidores multimédia.

O projeto Ad Builder pretende otimizar o processo de gestão dos anúncios realizado pela *video ad network* Adklip. É portanto uma ferramenta de *back office* para apoio à Adklip.

A Adklip, responsável pela gestão de vários *websites*, poderá criar anúncios a partir dos *templates* disponíveis, alterar e selecionar parâmetros dos respetivos anúncios e finalmente publicá-los. De forma a exibir os anúncios nos *websites* dos *publishers*, a Adklip fornece-lhes o URL de um ficheiro que contém as configurações do anúncio a publicar.

A fase inicial deste projeto passou pela definição dos requisitos recorrendo à criação de *User Stories* (presentes no Apêndice A) para modelação ágil. *User Stories* consistem na descrição das funcionalidades segundo a perspetiva do utilizador. De seguida foram desenhados os *mockups* de todas as *interfaces* (ilustrados no Apêndice B).

Parâmetro	**Descrição**
Ad Tag	URL do *Ad VAST* a ser exibido. Para que um *video ad* seja reproduzido num *video player*, este envia um pedido ao servidor de publicidade *VAST*. O servidor devolve o URL do *Ad VAST*
Demo	Permite substituir temporariamente o CSS de um anúncio por um ficheiro *custom*. Possibilita experimentação de alterações sem comprometer o CSS principal que é utilizado por todos os *websites*
Height	Altura em pixeis
Infeed Autoplay	Caso o *product* escolhido seja o *Infeed*, o utilizador poderá escolher se o anúncio executa automaticamente ou não após o carregamento do *website*
Orientation	Orientação do anúncio. Tem dois valores possíveis: esquerda ou direita
Product	Tipo de *template* do anúncio. A Adklip fornece três tipos de *templates*: *Infeed*, *Inpage* e *Slider*
Width	Largura em pixeis
Volume	Volume do áudio do anúncio quando o rato lhe é passado por cima

Tabela 4.1: Tabela dos parâmetros dos anúncios

4.1 Estrutura

A partir do URL `http://adbuilder.omibee.com` o utilizador poderá aceder à pagina de entrada do *website*. Nesta página são apresentadas três opções de conta ao utilizador: *sign up* onde poderá criar uma conta; recuperação de *password* onde é apresentada um processo que envia um e-mail com instruções para alteração da *password*; e finalmente *login* que autentica o utilizador. Uma vez autenticado o utilizador, será conduzido para a *Homepage* onde estão exibidos todos os seus anúncios, publicados ou não. Com a funcionalidade da pesquisa, presente nesta página, o número de anúncios mostrados ao utilizador é reduzido e assim torna-se mais fácil encontrar o anúncio desejado. Dependendo do estado de publicação do anúncio, o utilizador poderá publicá-lo ou desativá-lo do *website* do cliente. Aquando da publicação de um anúncio, é disponibilizado um URL que referencia sempre a versão selecionada. O *publisher* poderá então introduzir este URL no seu *website* para exibir o anúncio, não necessitando de o atualizar posteriormente, mesmo que o anúncio sofra alterações. Adicionalmente o utilizador poderá remover o anúncio ou aceder aos seus detalhes. Nesta última ação o utilizador é reencaminhado para uma página onde estão discriminados todas as suas configurações e lhe é permitido editar esses parâmetros ou consultar versões anteriores destes.

Caso o utilizador esteja autenticado, poderá aceder à *Homepage* a partir de qualquer página. O acesso às página de ajuda e de informação pode ser realizado a partir de qualquer página. A primeira contém esclarecimentos sobre a utilização da ferramenta e a segunda fornece alguma contextualização do *website*.

É apresentada, na Figura 4.1, a estrutura funcional do *website* a partir de um fluxograma.

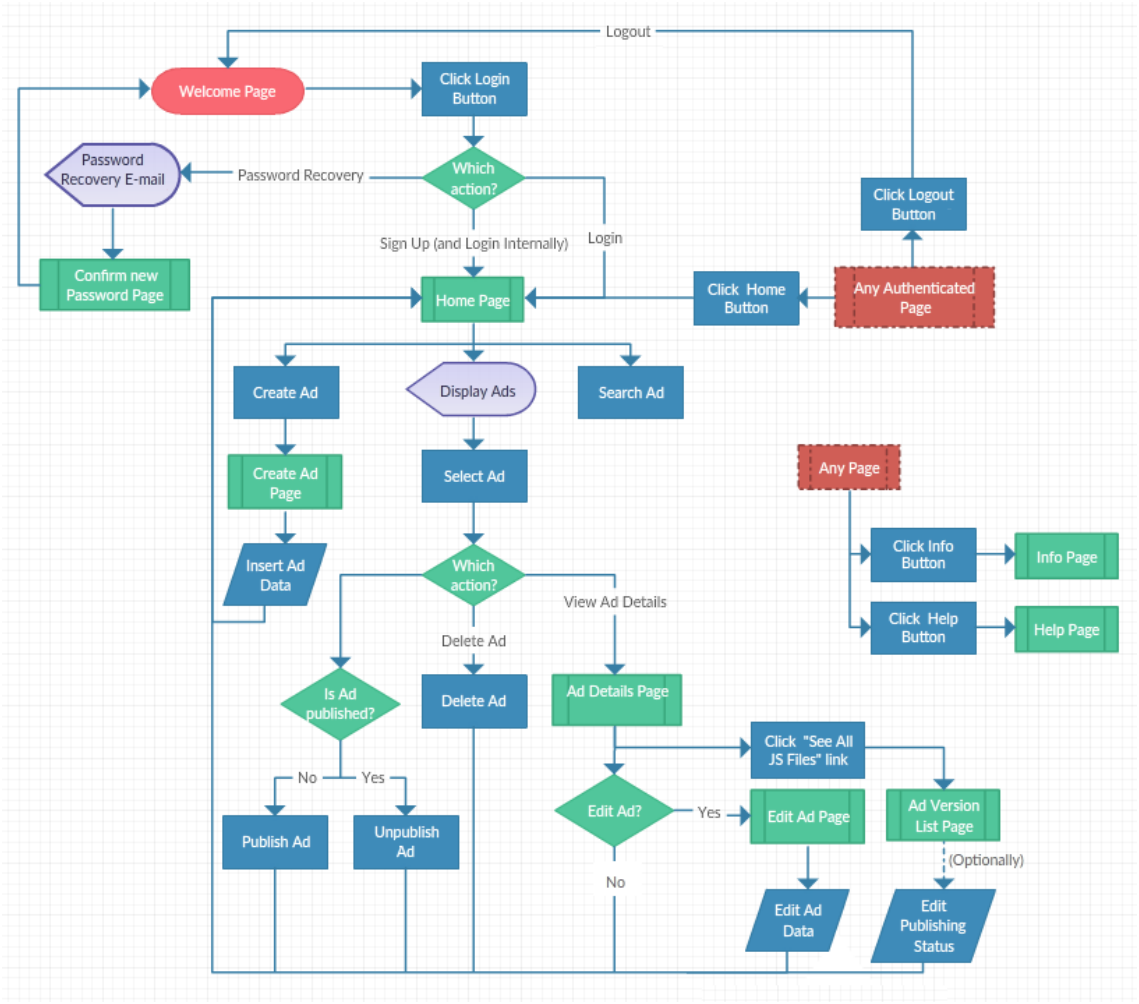

Figura 4.1: Fluxograma do *website* Ad Builder

4.1.1 Página de Entrada

A página de entrada é composta por três botões: Login, Info e Help. A imagem de fundo foi construída utilizando a plataforma Adobe Ilustrator.

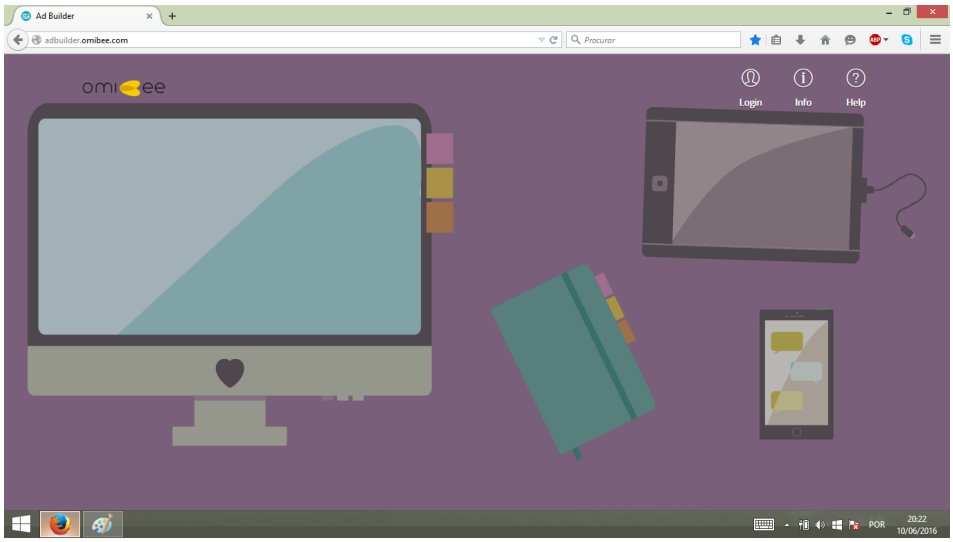

Figura 4.2: Página de Entrada (Resolução: 1366x768)

Caso o utilizador carregue no botão Login é-lhe apresentado o seguinte cenário:

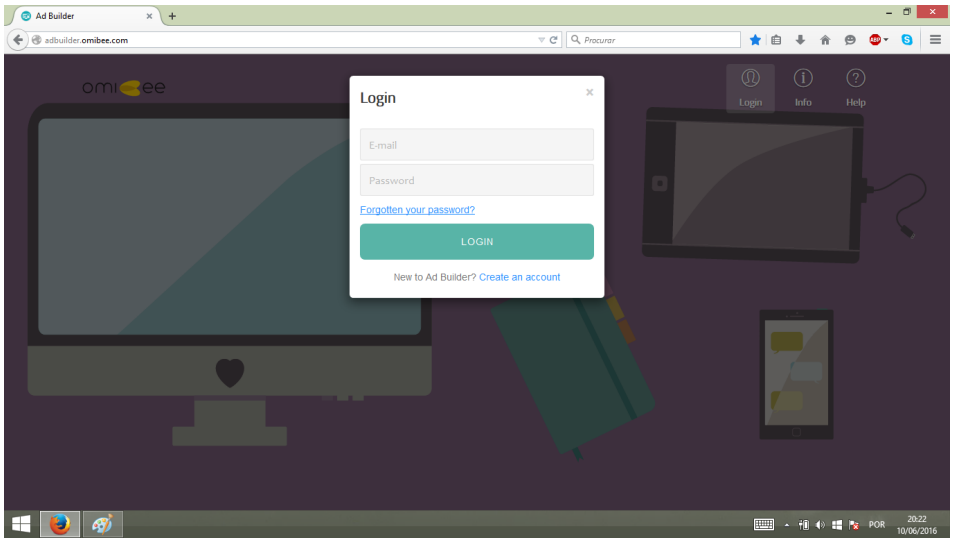

Figura 4.3: *Login* (Resolução: 1366x768)

Se o utilizador não estiver registado pode fazer *sign up* carregando no *link* "*Create an account*"e seguirá para o seguinte formulário:

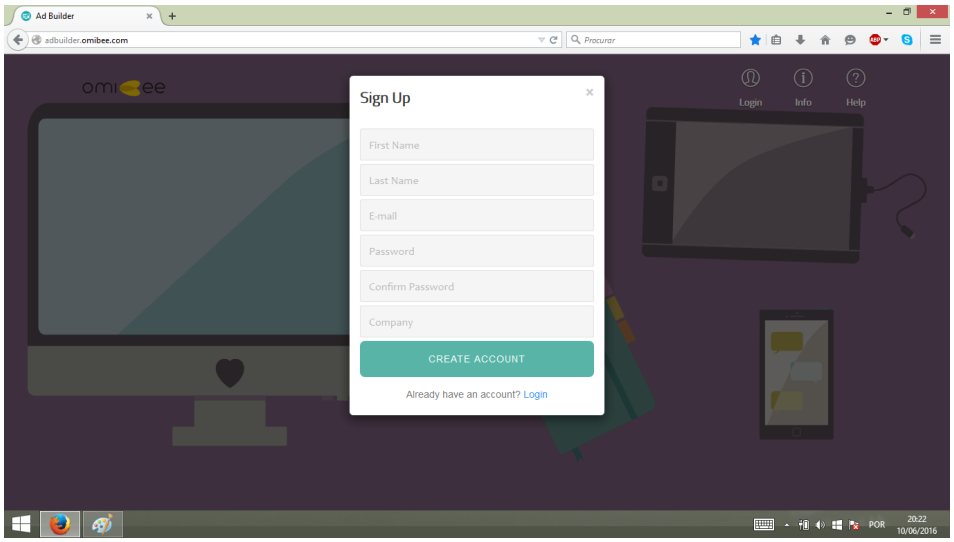

Figura 4.4: *Sign Up* (Resolução: 1366x768)

Também foi implementada a opção de recuperação de *password*, onde o utilizador deve fornecer o seu endereço de e-mail.

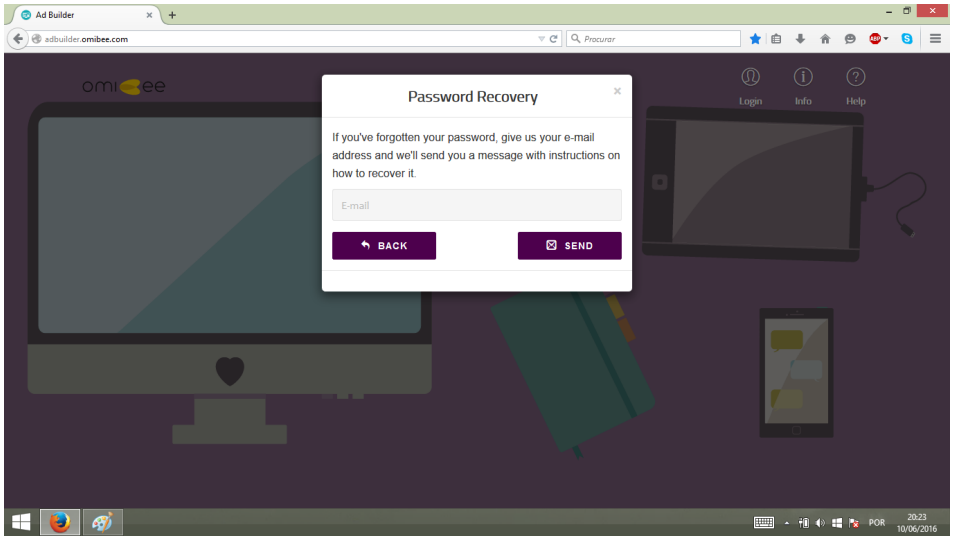

Figura 4.5: Recuperação de *Password* (Resolução: 1366x768)

De seguida o utilizador receberá um e-mail com um URL que lhe permitirá introduzir uma nova *password*.

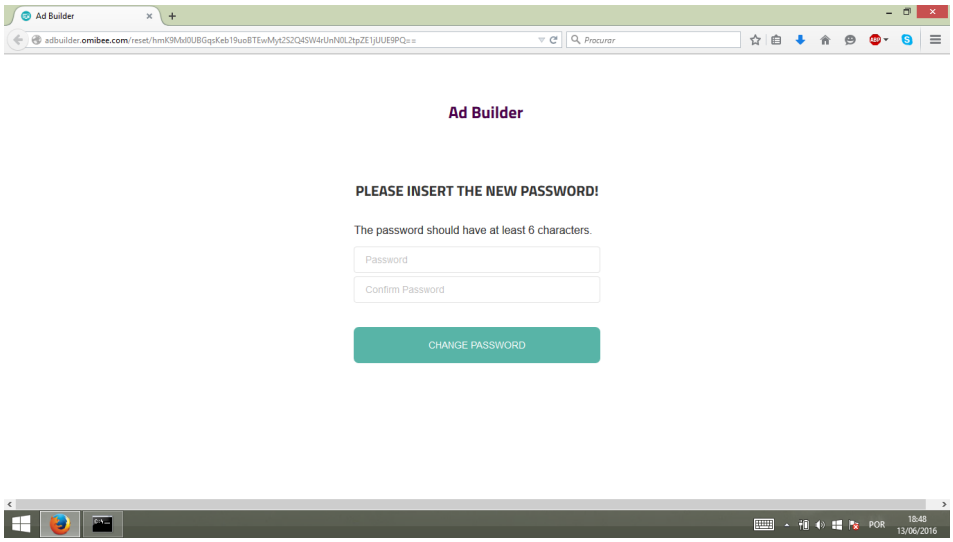

Figura 4.6: Confirmação de nova *Password* (Resolução: 1366x768)

4.1.2 Página de Informação

Esta página reúne informação técnica sobre o projeto. São abordados tópicos como o objetivo do Ad Builder, uma descrição dos tipos de *templates* suportados e ainda uma introdução do Amazon CloudFront que é o único serviço da AWS com o qual o utilizador tem contacto direto.

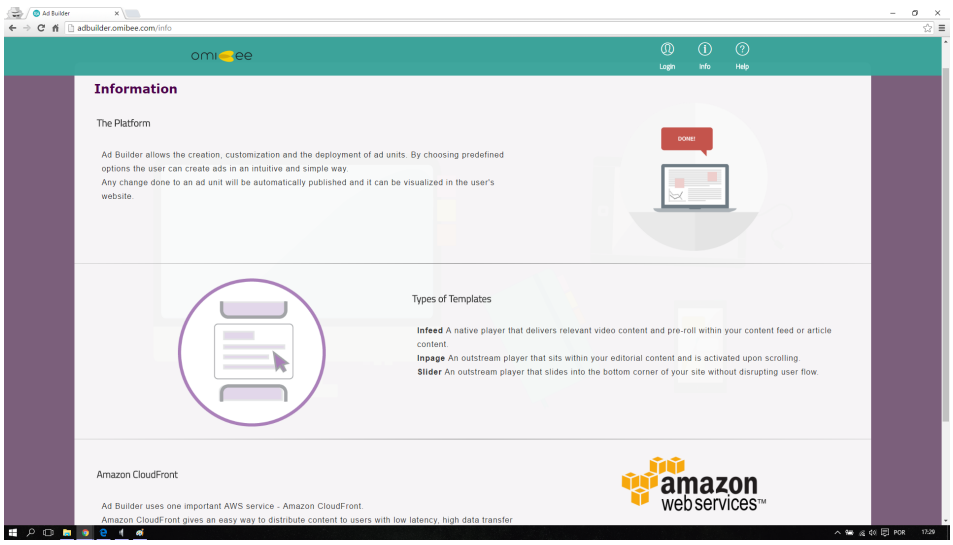

Figura 4.7: Informação (Resolução: 1920x1080)

4.1.3 Página de Ajuda

A página de Ajuda pretende expor uma compilação de casos de utilização mais relevantes da plataforma no formato pergunta-resposta.

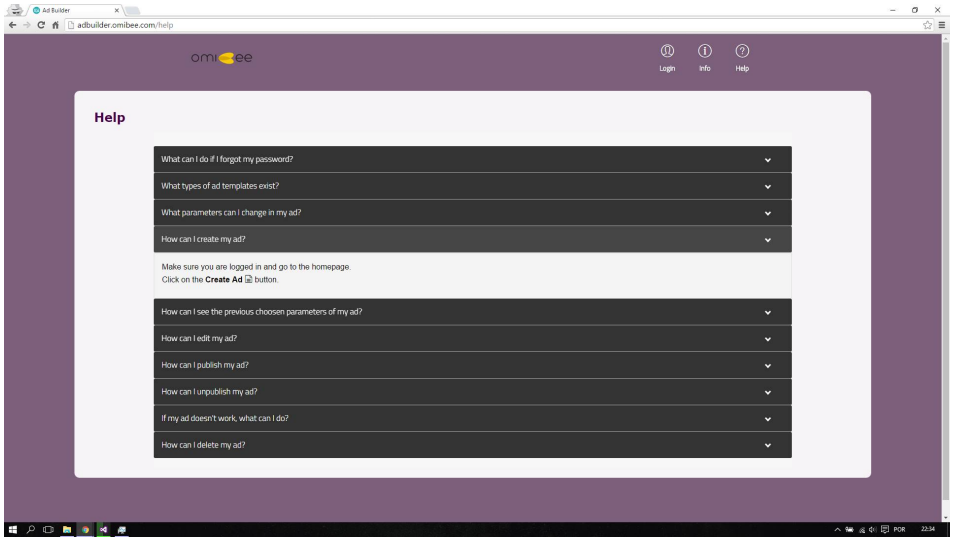

Figura 4.8: Ajuda (Resolução: 1920x1080)

4.1.4 Homepage

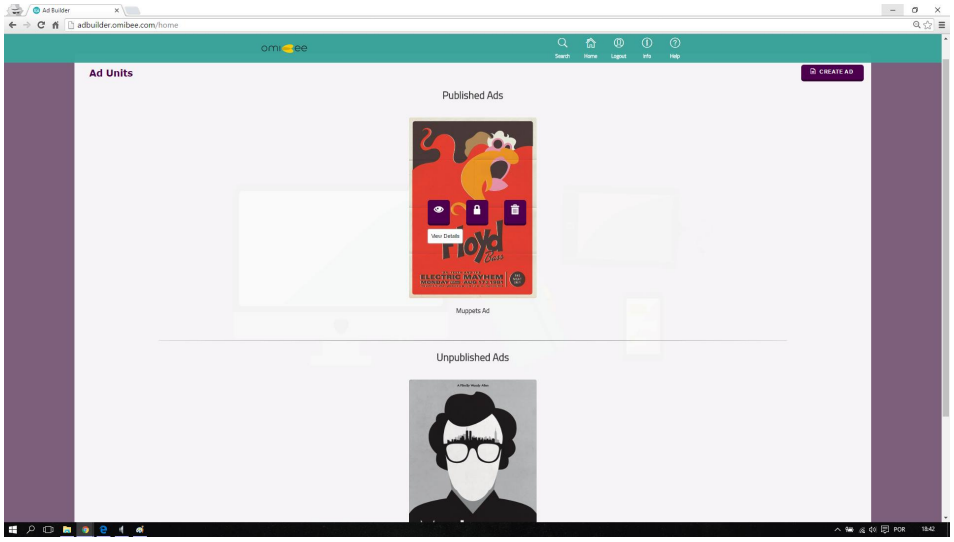

Figura 4.9: *Homepage* (Resolução: 1920x1080 e *Zoom Out* de forma a visualizar anúncios publicados e desativados)

Após um utilizador se autenticar, é conduzido até à página *Homepage*, onde são exibidos os seus anúncios. É efetuada uma divisão entre anúncios publicados e não publicados. É

importante salientar que as imagens apresentadas não representam os anúncios em concreto, mas sim um identificador arbitrário - *thumbnail* - uma vez que todos os anúncios são em formato vídeo.

Nesta página o utilizador poderá pesquisar um anúncio em particular, a partir do seu nome. Finalmente, também é fornecida a possibilidade de criar um anúncio.

Ao selecionar um determinado anúncio, o utilizador tem a possibilidade de escolher uma das três opções: removê-lo, publicá-lo ou desativá-lo (caso o anúncio já se encontre publicado poderá desativá-lo, em caso contrário poderá publicá-lo) ou ainda visualizar os seus parâmetros. Seguidamente irá ilustrar-se os dois primeiros casos.

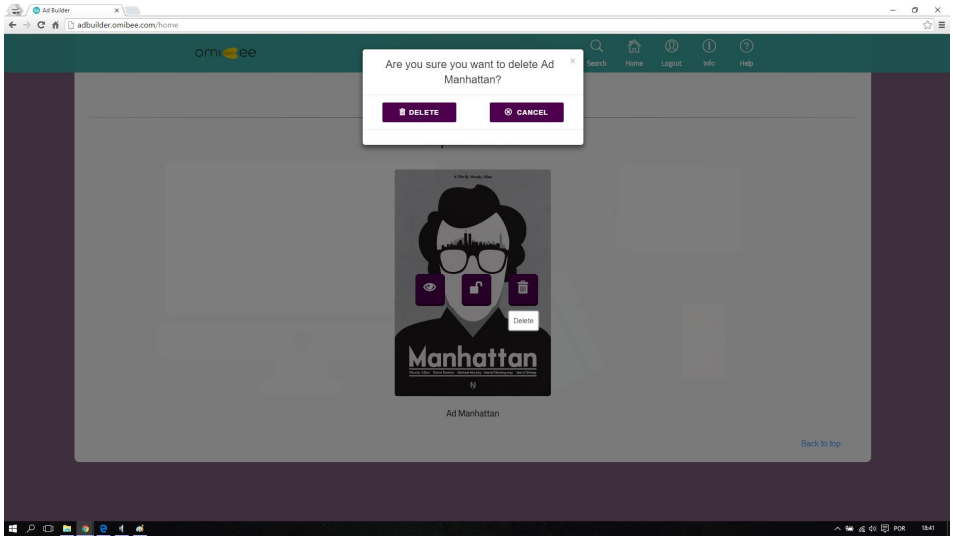

Figura 4.10: Remoção de Anúncio (Resolução: 1920x1080)

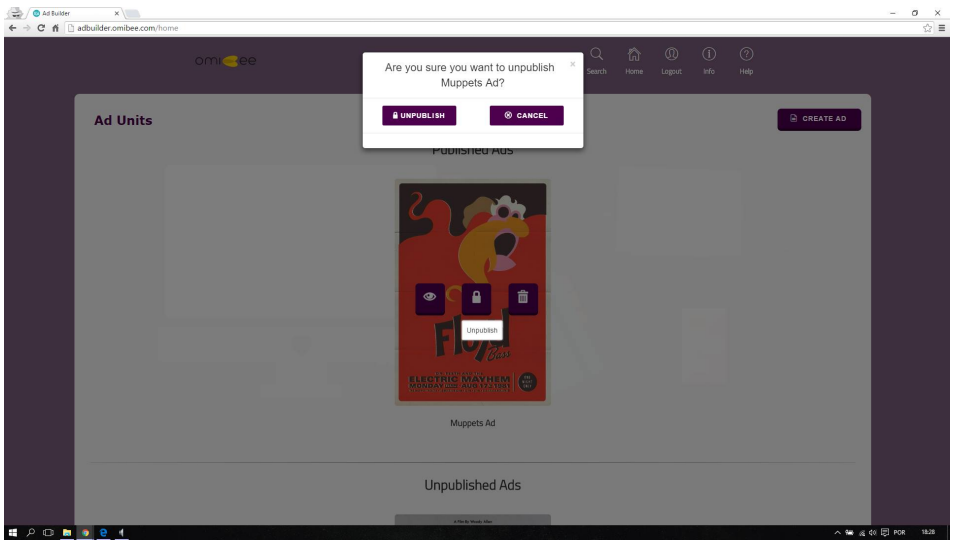

Figura 4.11: Desativação de Anúncio (Resolução: 1920x1080)

No caso da publicação de um anúncio, será mostrada uma mensagem ao utilizador para que introduza o seguinte código no seu *website*:

<script type='text/javascript' src='URL OF YOUR AD'></script>

Um exemplo concreto seria:

<script type='text/javascript' src='http://d312nv6tcxevti.cloudfront.net/22.js'></script>

O utilizador só necessitará de introduzir este código uma vez, após a primeira publicação, pois nas ações seguintes (como edição, omissão ou remoção do anúncio) a plataforma trata de atualizar os dados para corresponder à situação. Portanto, o mesmo código passa a referenciar a versão desejada.

4.1.5 Página da Visualização dos Parâmetros de um Anúncio

Nesta página são apresentados todos os parâmetros da última versão do anúncio. O utilizador poderá editá-lo ou ainda visualizar todas as configurações anteriores, carregando no *link* "*See All JS Files*".

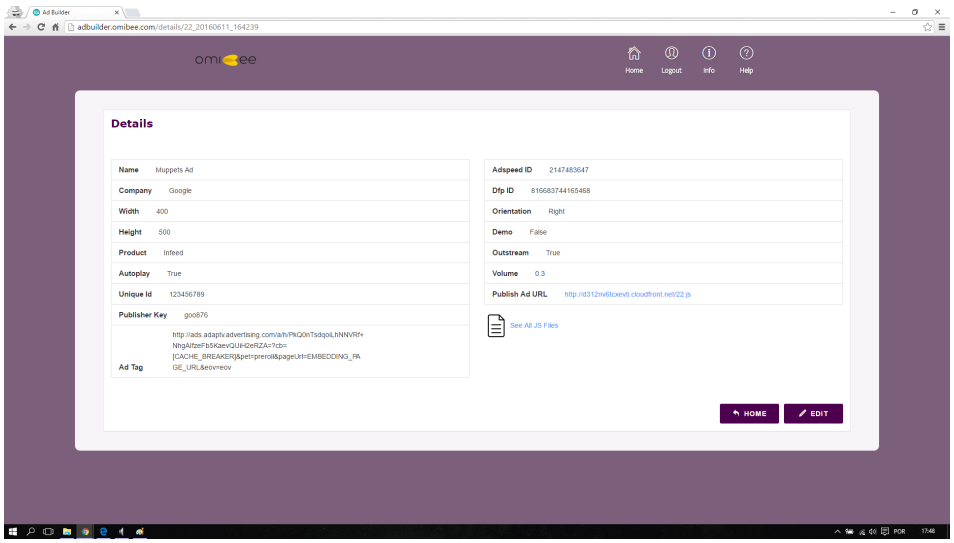

Figura 4.12: Visualização dos Parâmetros de um Anúncio (Resolução: 1920x1080)

4.1.6 Página de Criação de Anúncios

Esta página destina-se à criação de anúncios a partir de um de quatro *templates* possíveis. Os *templates* surgem como combinações de três parâmetros dos anúncios: o produto, a altura e a largura. Os produtos disponíveis são Slider, Inpage e Infeed.

Um dos parâmetros dos anúncios é o volume do áudio. O controlo deste elemento é realizado através de um *slider*. Este *slider* foi implementado em Javascript e desenhado no Adobe Ilustrator.

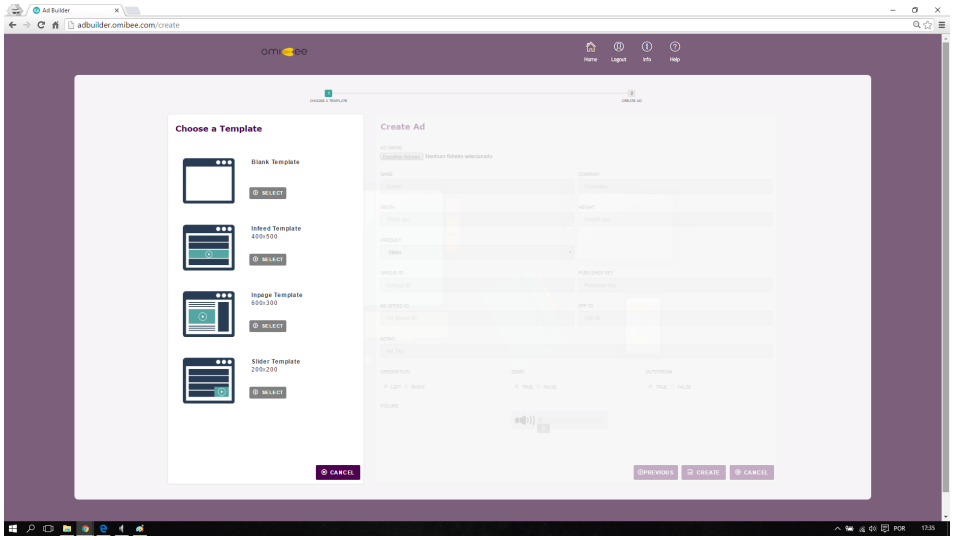

Figura 4.13: Seleção de *Template* (Resolução: 1920x1080 e *Zoom Out* de forma a visualizar todo o formulário)

Após a seleção de um *template*, o utilizador poderá introduzir os dados em falta, configurando assim o anúncio.

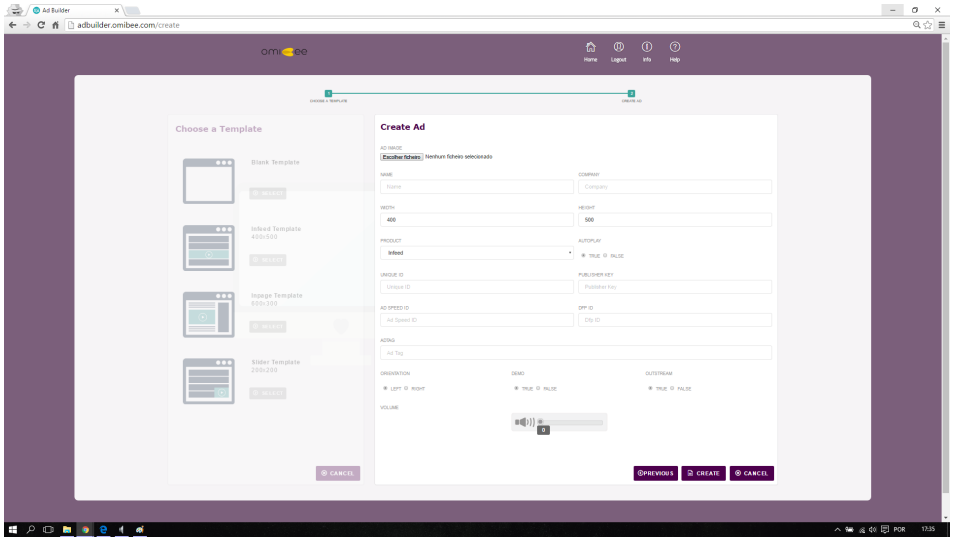

Figura 4.14: Criação de Anúncio (Resolução: 1920x1080 e *Zoom Out* de forma a visualizar todo o formulário)

4.1.7 Página de Visualização das Versões Anteriores de um Anúncio

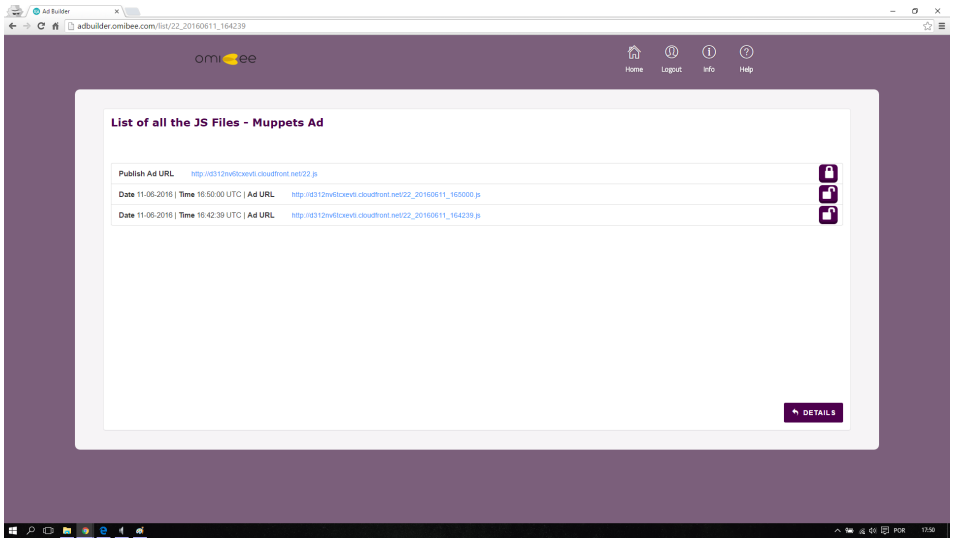

Figura 4.15: Lista dos URLs dos ficheiros das versões anteriores (Resolução: 1920x1080)

Nesta página é apresentada uma lista com todos os URLs dos ficheiros Javascript com as configurações anteriores. A cada URL é associado a data e o tempo da criação do ficheiro usando o fuso horário UTC. Se assim o desejar, o utilizador pode ativar uma versão diferente ou ainda desativar a atual.

4.1.8 Página de Edição de Anúncios

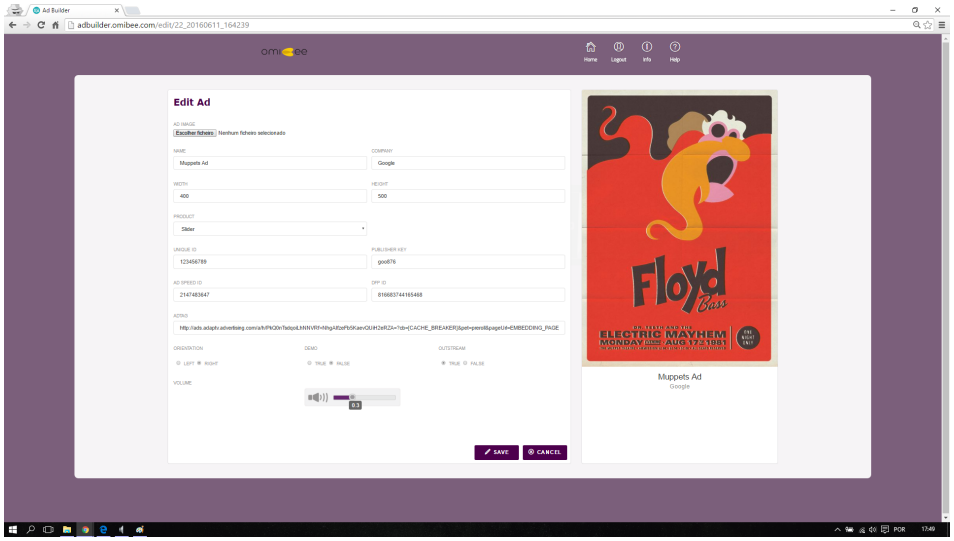

Figura 4.16: Edição de Anúncio (Resolução: 1920x1080 e *Zoom Out* de forma a visualizar todo o formulário)

A página de edição é semelhante à página de criação, na medida em que expõe os mesmos parâmetros e a sua alteração tem o mesmo efeito. Após uma edição é criada uma nova versão do ficheiro Javascript para arquivo.

4.1.9 Página de Exceção - Erro 404

Quando uma página não é encontrada é exibida a seguinte imagem com a opção de regressar à Página de Entrada através de um *link*.

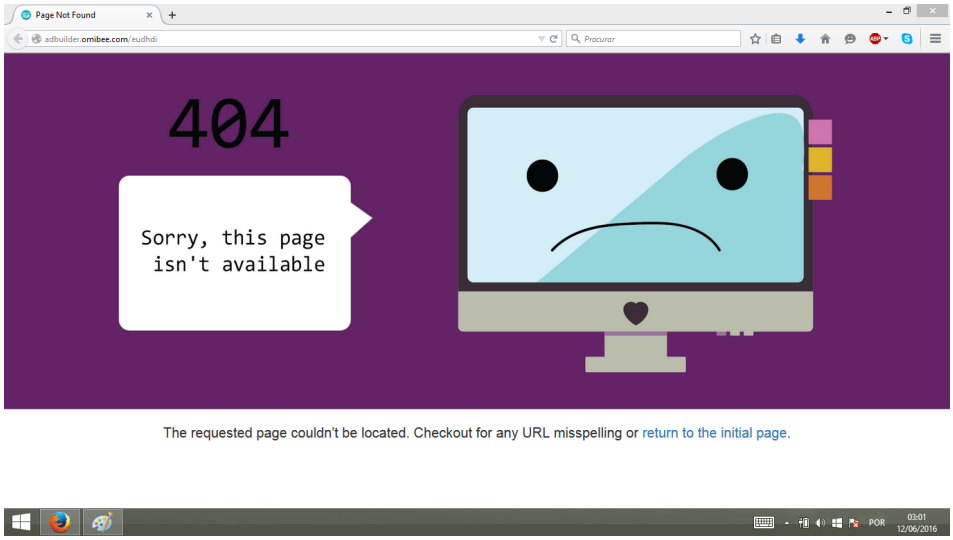

Figura 4.17: Erro 404 (Resolução: 1366x768)

4.2 Concepção e desenvolvimento

Há duas componentes sem as quais o desenvolvimento do projeto não pode começar: a *framework* Symfony e a ferramenta Vagrant.

4.2.1 Configuração do ambiente de desenvolvimento

O Vagrant desempenha o papel de solidificar a especificação do sistema operativo alvo por meio de uma máquina virtual de Linux. Recorreu-se à utilização de um *script* (apresentado no Apêndice C) que na primeira execução da máquina trata de instalar o servidor HTTP Apache, a base de dados MariaDB (posteriormente substituída por MySQL), a plataforma PhyMyAdmin (somente utilizada durante a fase de desenvolvimento), a linguagem PHP e a ferramenta CasperJS (e o *browser* PhantomJS necessário para a sua execução). É também

instalada a ferramenta Composer que será posteriormente necessária para configurar o Symfony.

O Symfony, no entanto, não é um *software* que possa ser instalado. É sim um conjunto de ficheiros e de convenções que são seguidas pelo projeto. Como tal, o projeto deve ser inicializado para conformar com este padrão - e para isso é usado o Composer. Depois desta operação, o projeto considera-se criado e a sua hierarquia de pastas estará definida. Pode então dar-se início ao desenvolvimento.

4.2.2 Diagrama de Casos de Uso

No entanto, antes de começar a programar deve estruturar-se uma lógica de utilização. Para este efeito, traduziram-se as *User Stories* para um diagrama de Casos de Uso, que é uma representação mais compacta e direta da cadeia de ações com as quais o utilizador influencia a componente Controller do paradigma MVC.

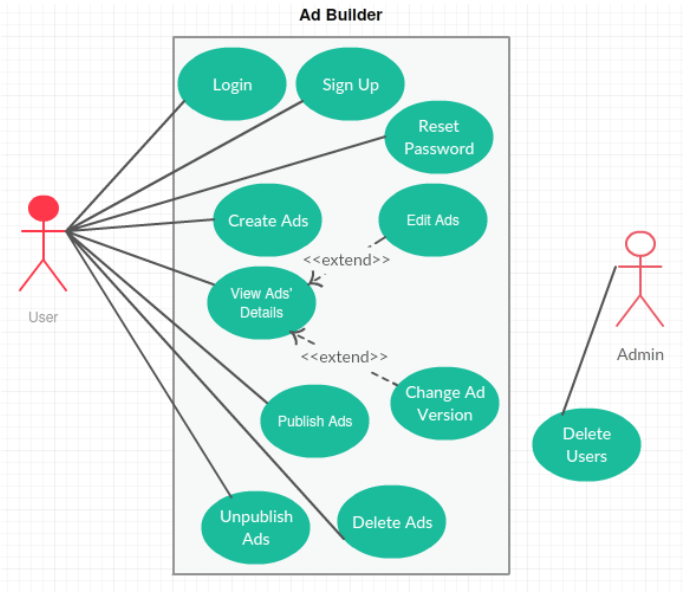

Figura 4.18: Diagrama de Casos de Uso

Note-se que existe um ator denominado Admin desconexo do plataforma Ad Builder. Este desempenha o papel administrativo de remover utilizadores da base de dados. Essa funcionalidade não foi implementada na camada gráfica da plataforma uma vez não se pretende delegar funções administrativas ao utilizador regular. Se assim o desejar, um administrador pode registar-se como utilizador e manter as permissões de gestão de utilizadores fora da plataforma.

4.2.3 Implementação

O diagrama da secção anterior é particularmente útil quando é usado como fundamento para um Controller. Isto porque advém, da definição de Controller, o seu princípio de operação baseado em ações. Cada uma destas ações está isolada das outras através de um caminho único, denominado *route*. Adicionalmente, uma ação pode ser renderizada para uma *view* em particular, na maior parte dos casos passando por um documento *twig*.

A associação de uma ação com o seu respetivo *route* permite a simplificação do processo de endereçamento, resultando em URLs curtos e compreensíveis. A título de exemplo, imagine-se a ação de acesso à pagina da Informação à qual se chama "info"; o URL para esta ação será então "adbuilder.omibee.com/info", sendo o "adbuilder.omibee.com"o domínio e "info"o *route*.

Esta lógica está distribuída por dois Controllers: o ControllerBase e o AdController. O primeiro trata de gerir as sessões de PHP e criar o objeto que representa um utilizador autenticado. O segundo trata de implementar todas as funcionalidades representadas no diagrama de Casos de Uso. Cada caso de Uso é processado dentro de uma ação correspondente.

É de notar que qualquer tipo de submissão de um formulário, seja este o da criação ou edição de anúncios ou ainda um relacionado com a conta do utilizador, é sempre sujeita à validação dos campos através de funções Javascript. Um mecanismo utilizado foi a utilização de expressões regulares.

Noutra nota, distribuídos pela arquitetura das pastas estão outros ficheiros de igual importância para o funcionamento da aplicação: os *templates twig* na pasta *views*; as extensões externas, na pasta *vendor*; e os *web assets* na pasta *web*.

Os *templates twig* subdividem-se em três categorias:

- O *base*, que contém a estrutura padrão do *website*, incluindo as referências aos *web assets* e a barra de navegação, presentes em todas as páginas.

- As páginas individuais, que são extensões do *base*, e estão associadas às ações correspondentes do AdController: *edit*, *create*, *delete*, etc.

- A *view* correspondente à exceção que é lançada quando ocorre o erro 404. Isto permite fazer *overwrite* à interface original que seria mostrada quando uma página não fosse encontrada.

A pasta *web* contém todos os ficheiros que a aplicação utiliza para efeitos de estilo (CSS e *fonts*), interatividade (Javascript) e imagens internas à plataforma (imagem de fundo, ícone da plataforma, *slider* de áudio, entre outros, não contendo as imagens dos anúncios).

A pasta *vendor* contém todos os componentes de terceiros, que são utilizados pelo projeto, nomeadamente o *Doctrine* (utilizado para a base de dados), a biblioteca Minify (utilizado para minimizar ficheiros), a biblioteca Swift Mailer (usado para a recuperação da *password*), entre outros.

A biblioteca Swift Mailer é particularmente útil para o envio do e-mail da recuperação da *password*. O mecanismo implementado foi a criação de uma conta do Gmail: adbuilderproject@gmail.com que é utilizado como o emissor do e-mail. Aquando da receção do e-mail, o utilizador pode aceder ao URL disponível para introduzir uma nova *password*. Para garantir a segurança do procedimento, a identificação do utilizador é encriptada utilizando uma cifra simétrica (AES256 no modo CBC) e incluída neste URL. Posteriormente o sistema obtém a identificação através da desencriptação desta componente do URL. Este método permite especificar o utilizador sem expor a sua informação.

4.2.4 Integração com o Amazon Web Services

4.2.4.1 Configuração

A etapa da configuração da integração com o Amazon Web Services foi realizada em colaboração com a Omibee, por questões de segurança e de permissões. Foram utilizados diversos serviços do Amazon Web Services nomeadamente Amazon EC2, Amazon Elastic Beanstalk, Amazon S3, Amazon RDS, Amazon CodeDeploy e ainda Amazon CloudFront.

A plataforma Ad Builder utiliza as instâncias do EC2 juntamente com o servidor Apache para servir as páginas da aplicação aos utilizadores. Esta migração para o EC2 é feita de forma automática depois de configurado o Amazon CodeDeploy. Incluídas nesta versão da aplicação vão credenciais de acesso à base de dados que reside atualmente no Amazon RDS. A base de dados anterior foi substituída por uma de MySQL. Estes três últimos serviços são geridos pelo Amazon Elastic Beanstalk, para o qual não são necessárias alterações por parte da aplicação. O Amazon S3 e o Amazon CloudFront são utilizados como pontos fulcrais da componente inovadora do projeto.

Como resultado desta configuração foi associado ao *website* Ad Builder um domínio:
http://adbuilder.omibee.com

Para uma melhor compreensão da interação entre os vários serviços foi desenhado um diagrama de componentes, representado na Figura 4.19.

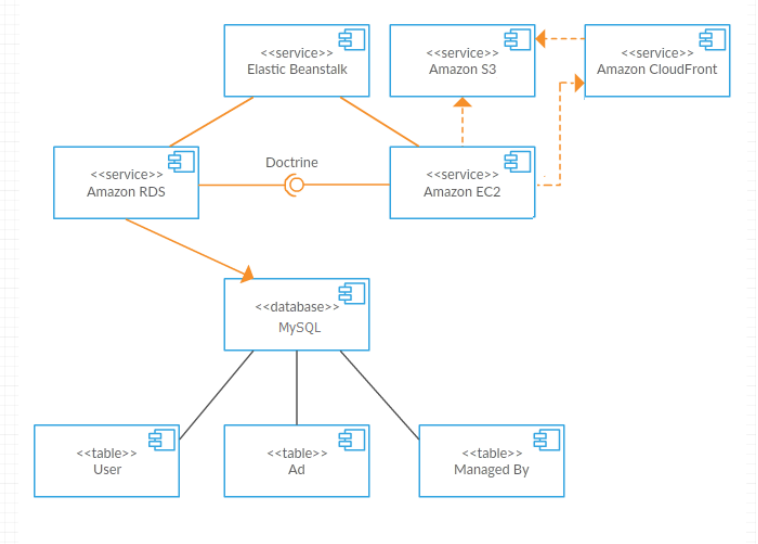

Figura 4.19: Diagrama de Componentes

4.2.4.2 Implementação

O Amazon S3 é responsável por armazenar todos os ficheiros dos utilizadores. Entre estes incluem as imagens que servem como *thumbnails* dos anúncios e ainda os ficheiros Javascript com as configurações dos mesmos anúncios. Para tal foi criado um *bucket* no S3 denominado adbuilder.omibee.com. Todo o conteúdo disponível no S3, no âmbito desta aplicação, também pode ser acedido através do CloudFront, utilizando o mesmo sufixo do URL. Por exemplo:

O URL de um anúncio no S3:

https://s3.eu-central-1.amazonaws.com/adbuilder.omibee.com/22.js

O URL do mesmo anúncio no CloudFront:

http://d312nv6tcxevti.cloudfront.net/22.js

O utilizador só acede a ficheiros a partir do CloudFront e nunca do S3.

Quando um utilizador cria um anúncio é gerado um ficheiro Javascript com o nome no formato id_date_time.js, sendo o *id* o identificador do anúncio, o *date* a data no formato YYYYMMDD e o *time* a hora segundo o fuso horário UTC, no formato HHMMSS. Este ficheiro representa uma nova versão deste anúncio e é guardado de forma persistente no Amazon S3 - mas apenas depois de ser minimizado recorrendo à biblioteca Minify. Neste processo é pedida ao utilizador uma imagem que representa a *thumbnail* do anúncio. Esta é também armazenada no S3 com um nome no formato semelhante: id_date_time.png.

Os formatos de imagens suportados são: jpeg, png, tiff e gif. Sempre que é criado um novo anúncio, este fica desativado por omissão. O utilizador tem depois a possibilidade de o publicar.

Caso o utilizador publique o anúncio, é sintetizado um novo ficheiro Javascript com o nome no formato id.js - que é uma cópia da versão desejada - sendo imediatamente minimizado e posteriormente armazenado no S3. Esta mudança de nome permite que se reutilize o URL para aceder à versão correta. Adicionalmente, é fornecido ao utilizador um *link* na forma:

<script type='text/javascript' src= 'http://d312nv6tcxevti.cloudfront.net/id.js'></script>

cuja fonte é o URL do anúncio no Amazon CloudFront. É de notar a presença do id no *link* - as versões anteriores são mantidas intactas no S3.

Reciprocamente, este ficheiro é apagado quando o utilizador desativa o anúncio, não eliminando portanto as versões anteriores arquivadas no S3. Como passo final desta operação é feita uma invalidação da *cache* de modo a garantir que o anúncio deixa de ser servido a partir do CloudFront, e consequentemente exibido no *website* do *publisher*.

No caso de se pretender editar um anúncio, existe uma etapa que é sempre realizada: é gerada uma nova versão do ficheiro, de forma semelhante à criação de um anúncio novo (id_date_time.js) - incluindo o processo de minimização e armazenamento no S3. Adicionalmente, se o utilizador decidir atualizar a *thumbnail* desse anúncio, pode fornecer uma nova imagem, que é armazenada no S3 com um nome concordante com a nova versão do anúncio. Neste caso, porém, a imagem antiga é apagada e a *cache* invalidada pois não faz sentido mantê-la do ponto de vista da aplicação. O caso especial da edição é quando o anúncio já se encontra publicado. Nesta circunstância o antigo ficheiro com o nome id.js é removido e a sua *cache* é invalidada. Isto impede que o CloudFront sirva a versão anterior do anúncio. De seguida é criado um novo ficheiro id.js com as novas configurações. Na perspetiva do *publisher*, nenhuma alteração terá de ser realizada, pois a plataforma Ad Builder substituiu efetivamente o ficheiro que é servido.

Quando o utilizador pretende remover um anúncio, são apagados todos os ficheiros - de imagem e Javascript - cujo id coincida com o do anúncio apagado. Adicionalmente, caso o anúncio esteja publicado, é necessário invalidar também a *cache*. É de notar que não é necessário invalidar a *cache* da imagem, uma vez que não estará acessível fora do Ad Builder.

4.2.5 Base de Dados

Com a integração da Amazon Web Services, a base de dados migrou para o Amazon RDS. A substituição da base de dados MariaDB para MySQL apenas implicou a alteração das credenciais num ficheiro de configuração do Symfony. No entanto, isto só foi possível porque já se tinha adotado o modelo Doctrine. Este modelo permitiu separar as entidades e as suas relações em objetos de PHP. Na perspetiva do Symfony estas classes têm que obedecer a certas regras. No entanto, a criação do código-fonte é feita de forma procedural através da consola, onde são definidos o nome e os atributos de cada entidade. É posteriormente gerado, também através da consola, um conjunto de ficheiros denominados Repositories.

As entidades intervenientes neste projeto são o User, a Ad e o ManagedBy que são apresentadas na Figura 4.20 segundo o modelo Entidade Associação (juntamente com os atributos de cada um).

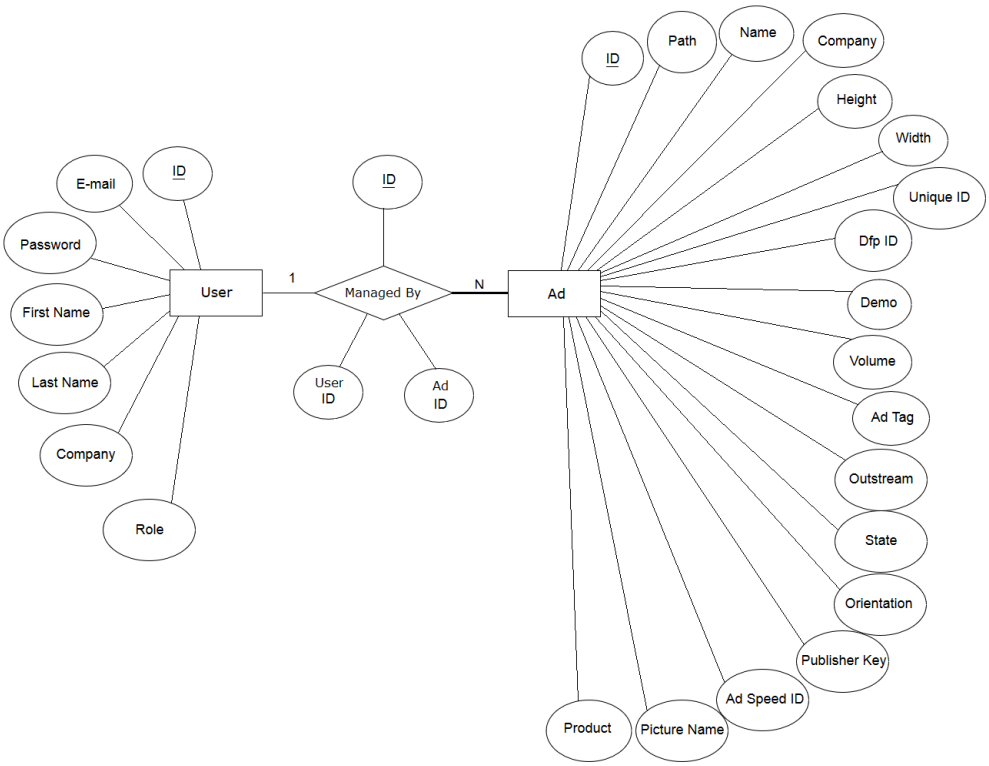

Figura 4.20: Diagrama do Modelo Entidade Associação

É de salientar que nas ações que envolvem a criação de uma *password* - como no *sign up* e na recuperação da *password* - esta é devidamente encriptada usando o algoritmo SHA256.

4.2.6 Testes Unitários

Para verificar o funcionamento e o cumprimento de requisitos da plataforma, foi estabelecido um *pipeline* de testes unitários que integra CasperJS, PhantomJS e Codeship.

Primeiramente foi criado um ambiente para o Symfony onde também residia uma base de dados especificamente populada para a realização de testes. Foi também desenvolvido um conjunto de *scripts* em Javascript chamado de *test suite*. Cada script desta *suite* testa uma funcionalidade específica de acordo com a lista de Casos de Uso. Para tal, cada teste percorre um número de passos - que avalia em sucesso ou fracasso - até chegar a uma conclusão. A API do CasperJS permite escrever testes que seguem uma abordagem da perspetiva do utilizador, e que são executados com a ajuda do PhantomJS. A título de exemplo, caso o utilizador quisesse testar o acesso à página de Informação, o teste começaria na página de entrada, passaria por clicar no botão "Info"e esperar que a página carregasse. Caso o teste fosse bem sucedido, a página carregada seria a página de Informação - algo verificável através de pistas textuais. Conclui-se que, para qualquer teste, é importante que as ações sejam encadeadas sequencialmente, e que a sua ordem seja respeitada.

O Codeship torna-se então uma ferramenta útil para automatizar este processo. Isto é, depois de definidos as configurações de teste num script (mais precisamente, instalação de ferramentas em ambiente de teste), o processo de realizar testes unitários é aliado ao processo de efetuar *git push* e em caso de sucesso, automaticamente efetua o *deployment* para o Amazon EC2 através do Amazon Elastic Beanstalk e Amazon CodeDeploy.

No Apêndice D apresenta-se como exemplo o teste unitário relativo à criação de um anúncio.

Capítulo 5

Validação do Website Ad Builder

De forma a validar a solução implementada nesta dissertação, foram realizados dois questionários. O primeiro questionário, presente no Apêndice E, foca-se na gestão dos anúncios publicitários e na interação contínua entre a Omibee e a Adklip. O segundo questionário, exibido no Apêndice F, pretende avaliar a experiência de um utilizador ao navegar no *website* Ad Builder. A Adklip foi inquirida através dos dois questionários, enquanto que a Omibee apenas foi inquirida pelo segundo.

O número reduzido de inquiridos resulta numa baixa relevância estatística dos questionários. Porém, representam uma porção dos utilizadores que integram o possível quadro do suporte técnico e os utilizadores da aplicação.

5.1 Questionário relativo à gestão dos anúncios e Interação Omibee-Adklip

A Adklip foi questionada relativamente ao processo anterior da gestão dos anúncios. Este inquérito foi respondido por um representante da empresa - um arquiteto de sistemas - que reflete a opinião da empresa. A partir do gráfico da questão 2 ("Choose the level of complexity of ad deployment (from creation to publishing)") e da questão 3 ("Describe the amount of time spent in each ad deployment") conclui-se que apesar do sistema atual não apresentar dificuldades alarmantes, também não representa um sistema ideal. A complexidade do sistema anterior e a quantidade de tempo gasto em cada *deployment* de um anúncio são expressos a partir um nível intermédio.

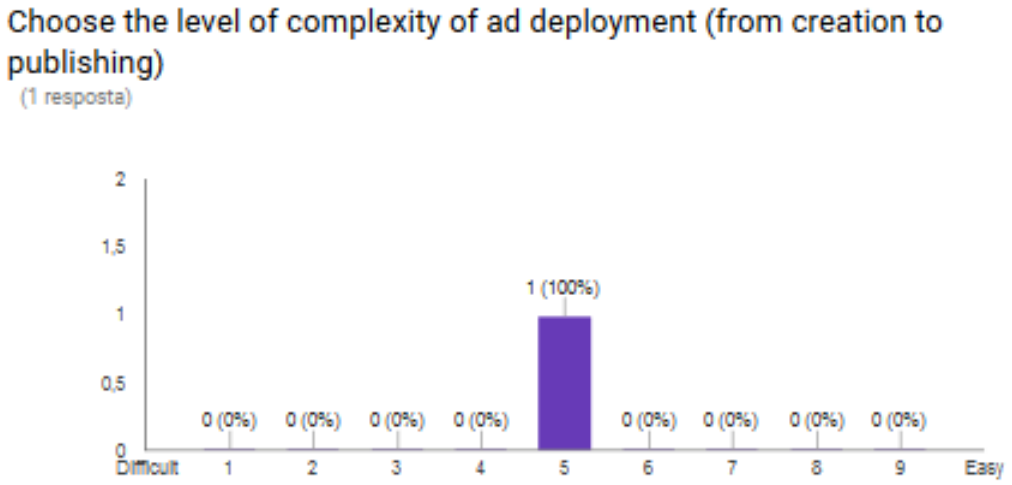

Figura 5.1: Gráfico de barras da segunda questão

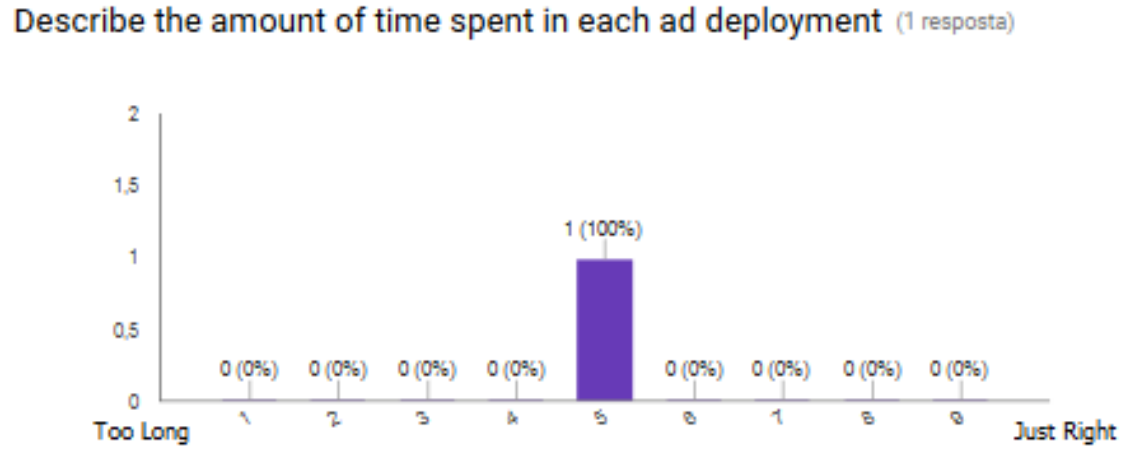

Figura 5.2: Gráfico de barras da terceira questão

Os problemas do sistema anterior são evidenciados quando os utilizadores do processo - Adklip - não têm elevados níveis de conhecimento de programação em Javascript, como é possível constatar no gráfico da questão 9. É importante notar que o *player* de anúncios foi implementado em Javascript. Caso surja o problema de efetuar alterações ou ocorra algum erro no código, a Adklip terá dificuldade em resolvê-los.

Figura 5.3: Gráfico circular da nona questão

Através da questão 6 ("Which parts of the deployment pipeline do you feel performs the worst?") e da questão 11 ("Which parts of the deployment pipeline requires more of Omibee's attention?"), a Adklip salientou duas principais dificuldades no processo de gestão de anúncios: a edição e o versionamento de ficheiros. Essa dificuldade resulta numa maior atenção da Omibee para garantir o seu correto funcionamento.

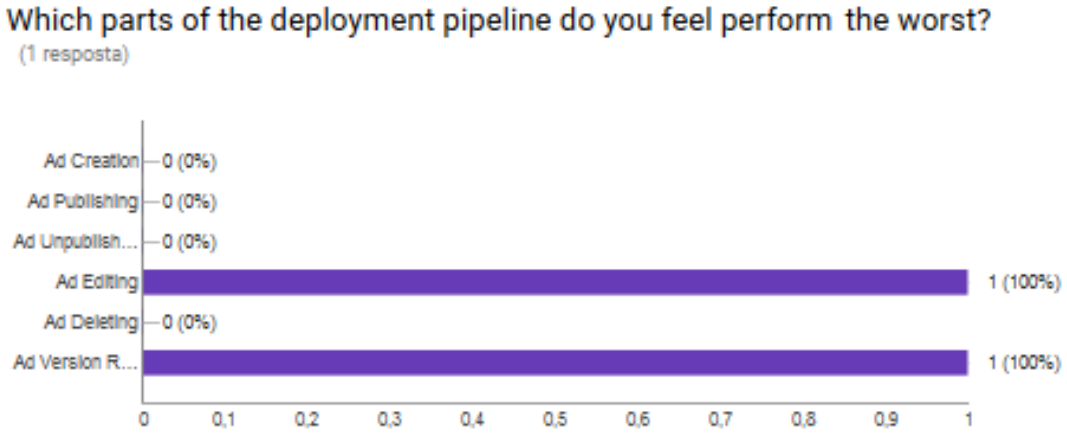

Figura 5.4: Gráfico de barras da sexta questão

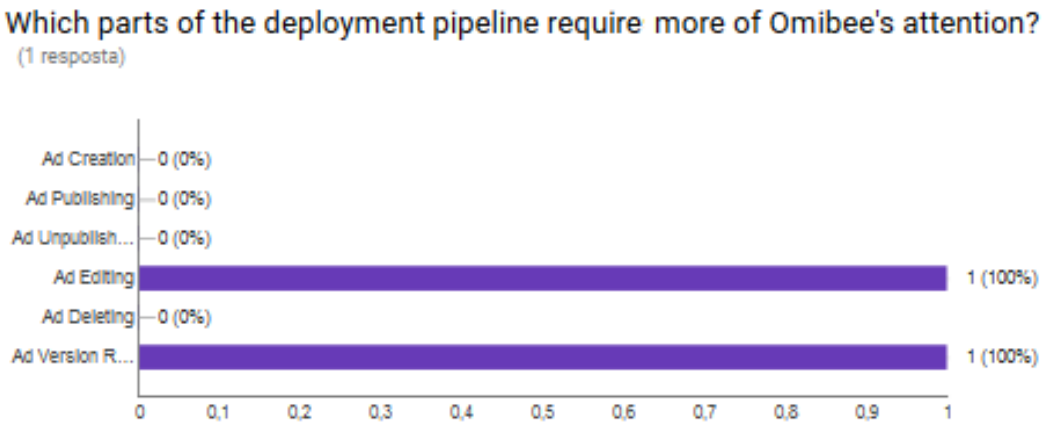

Figura 5.5: Gráfico de barras da décima primeira questão

Para assegurar uma escalabilidade saudável do sistema é necessário estabelecer uma estratégia robusta, a nível de passagem de conhecimento. Tal como foi mencionado pelo representante da empresa, a gestão de anúncios - com recurso a edição direta de ficheiros Javascript - não deverá ser realizada por utilizadores que não sejam desenvolvedores de código. Dessa forma, o problema poderia ser resolvido por uma de duas opções: implementar uma ferramenta para desenvolvedores ou uma ferramenta para não-desenvolvedores.

No negócio da publicidade *online*, a maior parte do conhecimento centra-se nos não-desenvolvedores. A solução escolhida não exige qualquer conhecimento de programação e torna-se então a solução mais simples e escalável. Com este método garante-se a união dos não-desenvolvedores ao código de forma segura.

Esta ferramenta também permite explicar conceitos e operações de forma mais intuitiva através da prática e do uso de indicadores visuais por oposição à alteração direta do código-fonte.

O comentário do representante da Adklip relativo à gestão dos anúncios e Interação Omibee-Adklip é apresentado de seguida.

"The current system is adequate and not too time consuming. However, manually editing JS files by a non-developer is probably not the best way to scale our business. Lag time from edits is also a problem."

5.2 Questionário relativo à *user experience* do *website* AdBuilder

5.2.1 Perspetiva da Omibee

A empresa Omibee é constituída atualmente por seis elementos: dois sócios-gerentes, uma *web designer* e três *software developers*. Um dos sócios-gerente é o co-orientador desta dissertação e uma vez que acompanhou todo o processo do desenvolvimento do projeto, não respondeu ao questionário. É importante salientar que apenas um membro da empresa - um dos *software developers* - colaborou com a Adklip e assim apenas esse membro tem conhecimentos do processo de gestão de anúncios e da especificação inicial do *player* de anúncios. Dado o contexto especializado do produto final, é de notar que a usabilidade sofre quando este é empregue por um utilizador sem formação. Por esta razão, alguns resultados não foram incluídos por não terem tanto significado na perspetiva da Omibee, estando dependentes das respostas da Adklip. Adverte-se então para o significado dos seguintes dados, a serem analisados posteriormente.

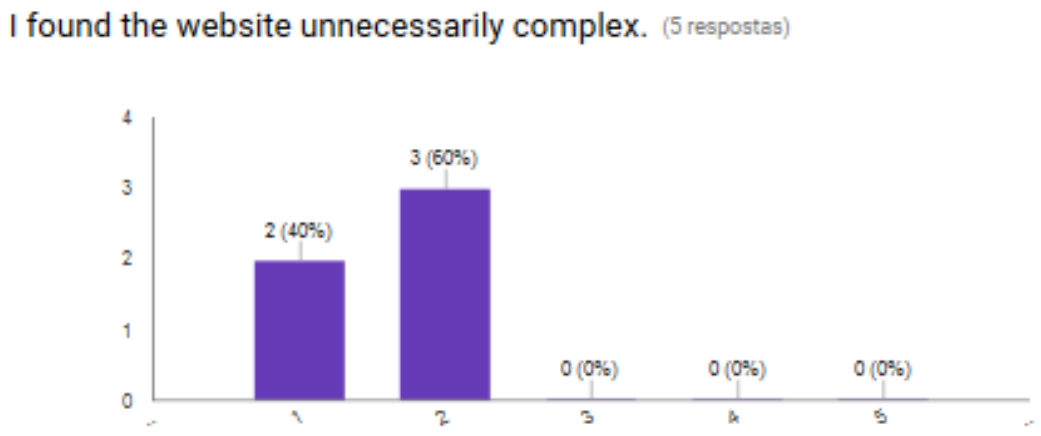

Figura 5.6: Gráfico de barras da quarta questão

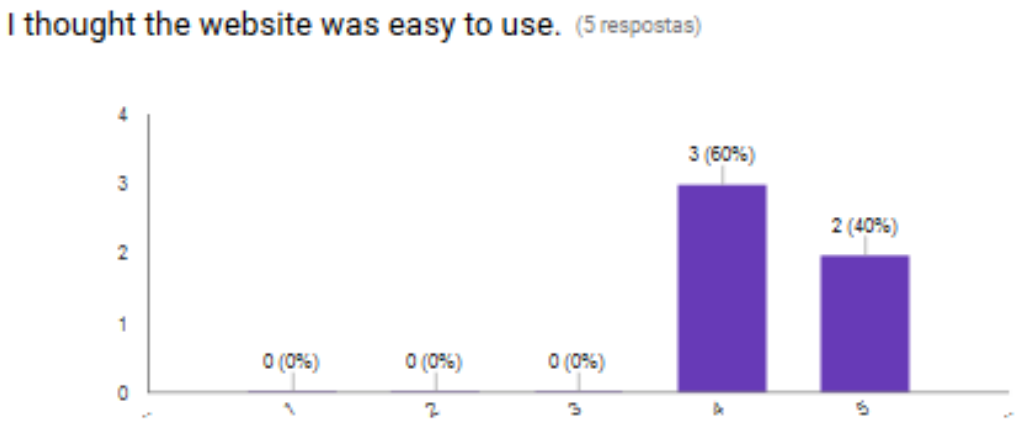

Figura 5.7: Gráfico de barras da quinta questão

A partir dos gráficos relativos das questões 4 ("I found the website unnecessarily complex") e 5 ("I thought the website was easy to use") é possível concluir que o *website* desenvolvido fornece aos utilizadores - com ou sem formação - uma experiência agradável de usabilidade e navegação.

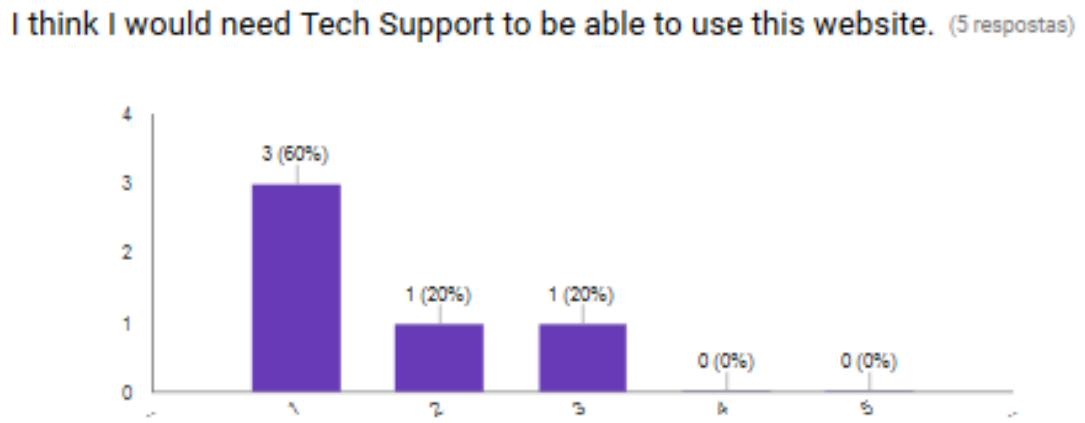

Figura 5.8: Gráfico de barras da sexta questão

No entanto a partir do resultado da questão 6 ("I think I would need Tech Support to be able to use this website"), um dos inquiridos manifestou-se quanto à necessidade de suporte técnico. Esta necessidade advém do pressuposto conhecimento prévio dos parâmetros dos anúncios publicitários. No entanto qualquer dificuldade pode ser colmatada com a informação presente na página de Ajuda do *website* Ad Builder.

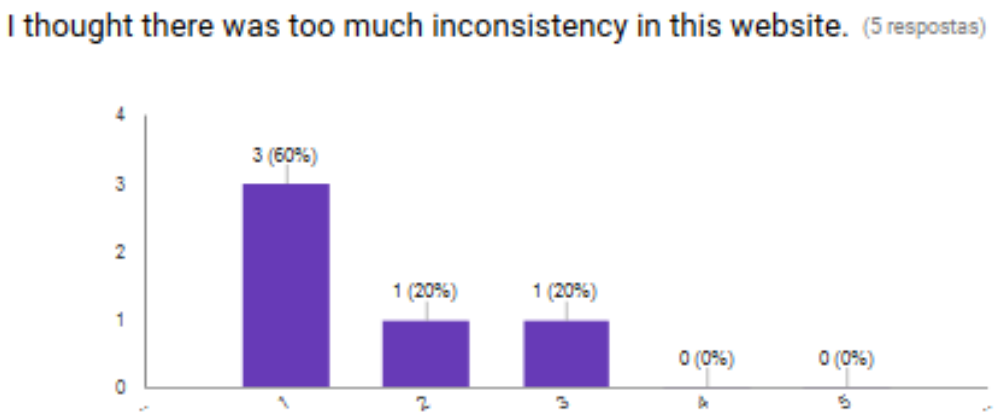

Figura 5.9: Gráfico de barras da oitava questão

Apesar da falta de conhecimentos relativos à gestão de anúncios por parte de diversos membros da Omibee, não foram encontradas inconsistências alarmantes no *website*, como é possível constatar na oitava questão ("I thought there was too much inconsistency in this website"). Consequentemente os inquiridos demonstraram-se confiantes na navegação e no uso do *website*.

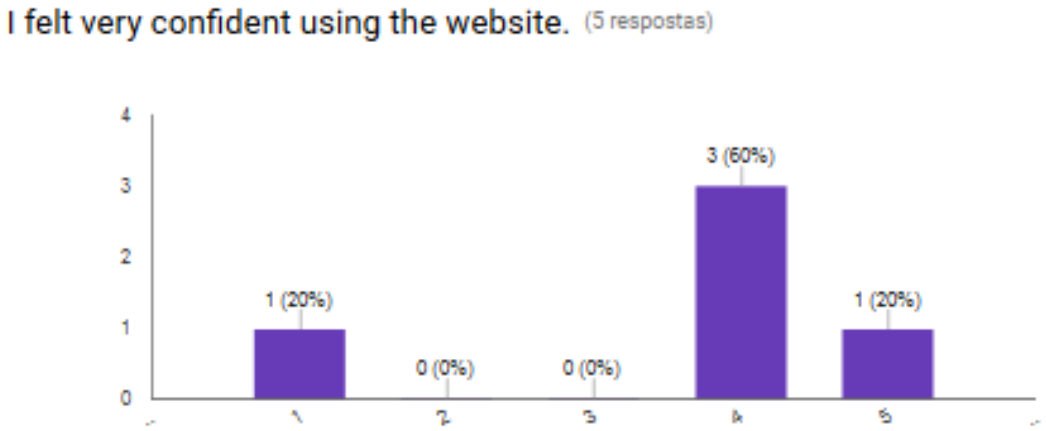

Figura 5.10: Gráfico de barras da décima questão

Os comentários dos inquiridos relativamente ao *website* são apresentados de seguida.

"*This tool will allow our client to save time and reduce manual human errors.*"

"*Nice job! There are required information like keys, unique id, adsped ID that I don't know what I should fill. Maybe some could be automatically filled and some tooltips could help to*

understand where to get it."

"*The concept seems fairly straightforward. I think there should be some more clarity regarding a few of the required form fields as it isn't clear what purpose they serve (maybe show a hint on mouseover). Overall, though, it was a pleasant experience.*"

"*Great work Sara!*"

"*I think it would provide a great service. Features are well developed and the overall website usage is great. Great features, good environment.*"

Visto que a amostra do questionário representa maioritariamente uma população não especializada, adverte-se para a significância dos dados no contexto da usabilidade. Isto é, considera-se que não refletem as impressões do cliente e dos intervenientes da gestão dos anúncios publicitários. É importante realçar que o *website* cumpre os requisitos exigidos e que nunca foi projetado para ser acessível fora da empresa Adklip - assumindo por isso uma base de utilizadores que tem formação na área. Apesar disto, o inquérito foi direcionado também para membros da Omibee, onde alguns dos quais poderão vir a fornecer suporte técnico no futuro.

Segundo os resultados do questionário, os inquiridos demonstram alguma dificuldade no uso da plataforma. Este resultado é expectável porque embora os parâmetros dos anúncios estejam explicitados na página da Ajuda, os utilizadores comentaram que não sabiam o que fazer com eles. Isto mostra que um utilizador sem conhecimentos prévios da área não consegue alcançar uma navegação tão fluída como é esperado que um utilizador especializado consiga. No entanto houve impressões positivas que referiam especificamente o cliente na medida em que anteviam o objetivo central desta plataforma: eficiência temporal e redução de erros humanos.

5.2.2 Perspetiva da Adklip

Tal como ocorreu no questionário presente na Secção 5.1, a opinião da Adklip foi ilustrada através de apenas um representante. A partir da Figura 5.11 é possível constatar que a empresa se mostra recetiva quanto à plataforma Ad Builder, uma vez que valoriza as vantagens desta face à alternativa atual. A agradável usabilidade e o grau reduzido de complexidade da ferramenta são pontos que prevalecem, visto que dada a especialização dos elementos da Adklip, o suporte técnico deixa de ser uma preocupação. O representante considera, sem hesitação, que o *website* implementado pode apresentar uma melhoria da qualidade de vida para os gestores do processo.

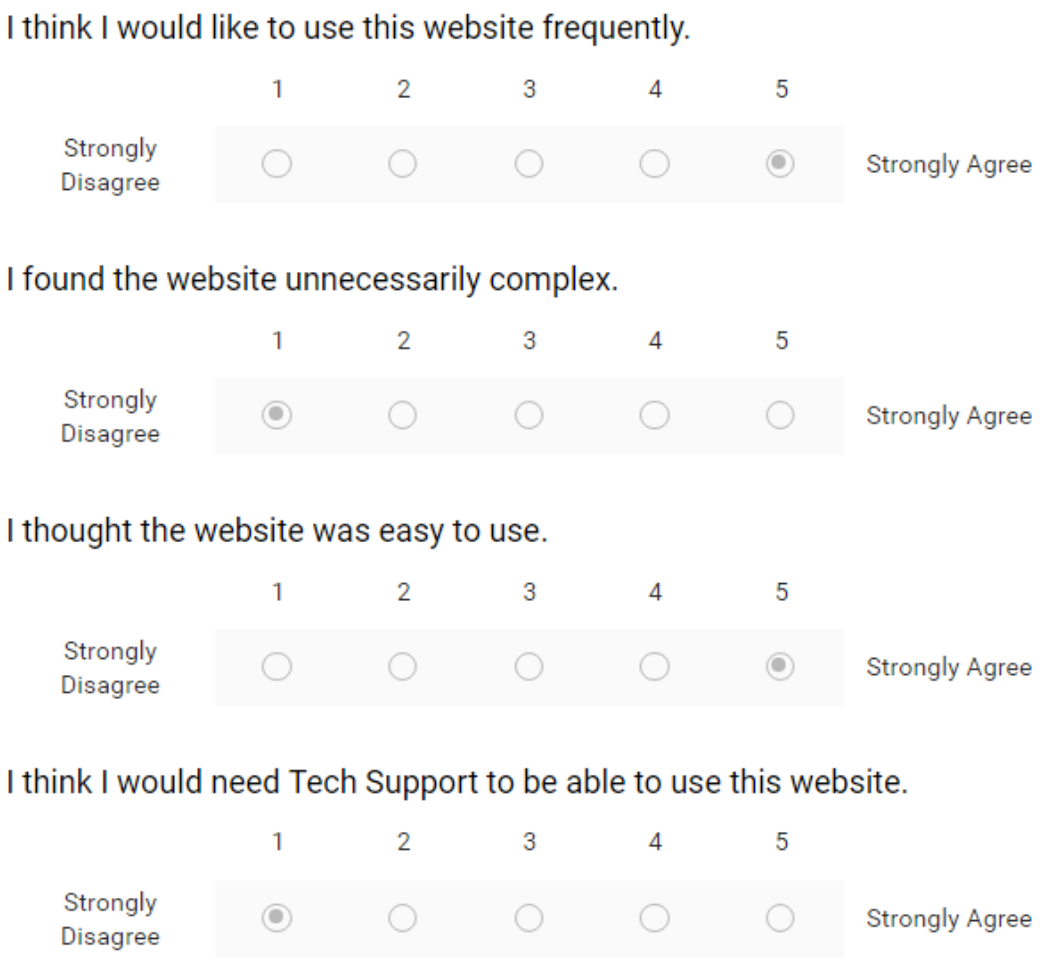

Figura 5.11: Respostas do representante da Adklip

A Figura 5.12 evidencia, ao contrário do que foi ilustrado na Secção 5.2.1, que o representante da empresa mostrou confiança na navegação do *website*, sem revelar qualquer percalço na sua utilização. Também é possível verificar que a utilização eficiente do *website* Ad Builder não obriga a obter mais conhecimentos além daqueles que já são adquiridos pelos elementos da Adklip.

O representante da empresa efetuou um comentário relativamente à plataforma Ad Builder que é apresentado de seguida.

"*This makes generating the JS files much easier and errors less common since we wouldn't be directly editing code.*"

A partir deste questionário conclui-se que os objetivos deste projeto foram alcançados, uma vez que se obteve uma validação direta do cliente. O problema que preocupava a Adklip

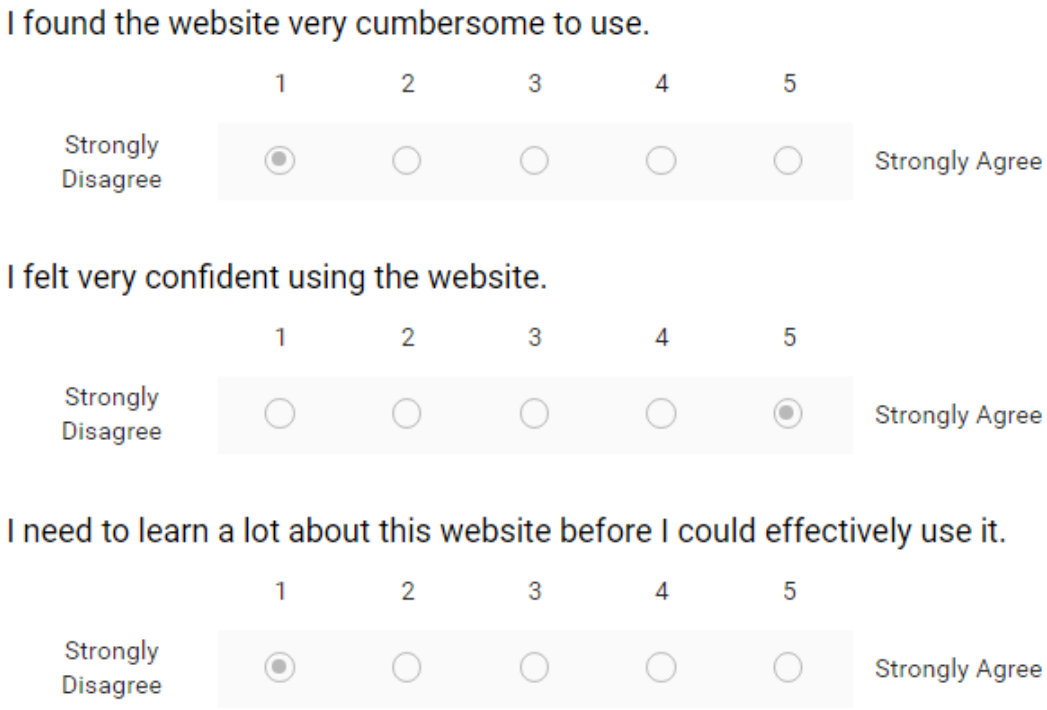

Figura 5.12: Respostas do representante da Adklip

em relação ao contacto direto com o código-fonte foi eliminado com segurança e o processo de modificação e de minimização de ficheiros foi totalmente automatizado.

Outra questão relevante para a Adklip é o suporte técnico atual necessário que é fornecido pela Omibee nas operações de edição de anúncios e no versionamento de ficheiros. Esta dificuldade é ultrapassada, visto que estas operações estão totalmente integradas na ferramenta Ad Builder.

Capítulo 6

Conclusão

6.1 Considerações Finais

A era atual é uma era na qual a Internet representa o principal meio para transportar conteúdo multimédia e como tal surgiu como um veiculo ideal para a publicidade. A transmissão de qualquer mensagem tornou-se dessa forma num processo simples, barato e com alcance ilimitado. A publicidade nesta plataforma torna-se numa estratégia de negócios imprescindível dada a tendência crescente da Internet.

O acesso à maioria dos *websites* implica a visualização de um anúncio cujo objetivo é apelar ao utilizador para comprar um determinado produto ou usar um dado serviço. A publicidade *online* representa uma parte significativa do mercado da *publicidade*, onde estima-se que ultrapassa a publicidade televisiva. A gestão da publicidade *online* é realizada em parte por empresas que fornecem serviços/soluções *web*. Estas empresas são responsáveis pela agregação da oferta de vários *advertisers* e pela venda desse inventário aos *publishers*.

A integração de unidades publicitárias nos diversos *websites* é altamente dependente do domínio e pode até gerar incompatibilidades. É portanto crucial desenvolver o código HTML/CSS/JS por forma a que o funcionamento e aparência não dependam do *browser* de Internet utilizado, do ambiente onde são desenvolvidos, entre outros fatores. O estado atual da distribuição da publicidade *online* envolve manter uma arquitetura que seja flexível a modificações como por exemplo a alteração de parâmetros.

Esta Dissertação surgiu através de uma proposta da empresa Omibee para um cliente específico: a Adklip. A Adklip fornece uma solução para a exibição de *video ads* em *websites* através de um *player* de anúncios publicitários. Este *player* está implementado a partir de um ficheiro Javascript que para além de incluir os mecanismos para a reprodução dos anúncios, também contém os seus parâmetros configuráveis, como a altura, a largura,

o volume, entre outros.

A motivação da Omibee advém da relação de dependência que a Adklip tem do suporte técnico, uma vez que foi a Omibee que desenvolveu a implementação inicial do *player*. Para tentar impedir também a introdução de erros humanos devido ao contacto direto com o código-fonte e para facilitar a gestão dos anúncios, optou-se por implementar uma aplicação gráfica com opções de interação limitadas.

A solução proposta nesta Dissertação consiste no desenvolvimento de uma plataforma que permite a criação, a personalização e o *deployment* de unidades publicitárias *online*. Cada anúncio é descrito em função de um conjunto de parâmetros, os quais o utilizador terá a liberdade de alterar. Seguidamente as configurações são guardadas num ficheiro Javascript. A exibição do anúncio num determinado *website* apenas requer o conhecimento do URL para esse mesmo ficheiro.

O *website* desenvolvido permite otimizar e facilitar todo o processo da gestão dos anúncios, uma vez que passa a ser realizado recorrendo a uma plataforma gráfica, sem existir a necessidade de programação. Tal como foi demonstrado nos resultados, a plataforma Ad Builder também fornece um aumento na eficiência temporal e uma redução dos erros humanos. A interação entre as duas empresas - Omibee e Adklip - passa a ser simbiótica e de suporte, em vez de dependência técnica.

6.2 Sugestões de trabalho futuro

Tal como foi referido anteriormente, este projeto tem em vista exclusivamente um cliente, onde é utilizado um formato específico de um *player* de anúncios em vídeo. Seria interessante ampliar o alcance do processo de geração de unidades publicitárias. Ou seja, expandir a oferta a outros tipos de *players*. No entanto devido à arquitetura implementada atualmente seriam necessárias alterações fundamentais para suportar outros tipos de ficheiros e respetivos parâmetros.

Outros tópicos possíveis de investigação incluem: melhoramento da segurança, nomeadamente a nível de restrições de acesso (como por exemplo, ter vários níveis de utilizador) e proteção da informação; e adição de funcionalidades administrativas (como por exemplo, ter a possibilidade de gerir os utilizadores, ter o mecanismo de atribuição dinâmica de unidades publicitárias a utilizadores específicos dentro das empresas).

Glossário

Termo	Descrição
Ad	Anúncio publicitário
Affiliate	Associação ou associado
Asset	Elementos como ficheiros Javascript, CSS e imagens que contribuem para a componente gráfica dos *websites*
Back-end	Sistema responsável pela lógica de negócio, serviços *web* e APIs de uma aplicação
Back Office	Uma aplicação *back office* inclui o *software* que uma organização utiliza para operações administrativas e as suas *interfaces* não são disponibilizadas aos utilizadores comuns
Big Data	Recolha e armazenamento de grandes quantidades de informações para eventual análise de dados
Browser	Programa que possibilita a interação dos utilizadores com documentos virtuais da Internet, também conhecidos como páginas *web*
Browser Cookie	Dados relativos ao utilizador que são geridos por um *browser*. As *Cookies* são trocadas constantemente entre o *browser* e a página *web* que se visita
Bucket	Unidade lógica de armazenamento utilizada pelo Amazon S3
Bug	Um erro ou defeito no *software* ou no *hardware* que causa o mau funcionamento de um programa
Cache	Área de armazenamento onde os dados ou processos frequentemente utilizados são guardados para um acesso futuro mais rápido, poupando tempo e uso desnecessário do *hardware*
Casos de Uso	Conjunto de possíveis sequências de interações entre sistemas e utilizadores num ambiente particular e relacionado com um objetivo particular
Cloud Computing	Utilização da rede e de servidores remotos na Internet para o armazenamento, gestão e processamento dos dados em detrimento da execução local
Controller	Retrata a ação desejada pelo utilizador na medida em que processa e gere a interação entre o *Model* e as *Views*
Data Mining	Processo de exploração de grandes quantidades de dados à procura de padrões consistentes
Deadline	Prazo de entrega
Deployment	Conjunto de atividades para garantir que um determinado sistema de *software* esteja disponível

Termo	**Descrição**
Developer	Desenvolvedor de *software*
Flash	*Software* utilizado geralmente para a criação de animações interativas que funcionam embutidas num *browser*
Follow	Botão que permite subscrever a atualizações de um certo conteúdo ou agente
Fork	Desenvolvimento de um programa a partir do código-fonte de outro programa, sendo este último geralmente um programa *open source*
Framework	Abstração que une códigos comuns entre vários projetos de *software* providenciando uma funcionalidade genérica
Front-end	Sistema responsável pela componente gráfica de uma aplicação, ou seja, pelas *interfaces* com o utilizador
Full-stack framework	*Frameworks* que permitem o desenvolvimento de aplicações desde a *interface* até ao armazenamento de dados
Gráfico de Gantt	Gráfico utilizado para ilustrar o progresso de diferentes tarefas de um projeto
GIF	Graphics Interchange Format - Formato de imagem de mapa de *bits* usado quer para imagens fixas, quer para animações
Git Commit	Adição de modificações do código de um projeto no repositório local e eventualmente a adição de uma mensagem a descrever essas alterações
Git Pull	Transferência das modificações do conteúdo no servidor remoto para o repositório local
Git Push	Transferência do(s) último(s) *git commit(s)* para o servidor remoto
HTML Tag	Código que descreve como uma determinada página *web* está formatada. As *tags* de HTML são definidas pelos caracteres < e >
Jitter	Variação estatística do atraso na entrega de dados numa rede
Layout	Disposição do texto, imagens e outros objetos numa página *web*
Like	Botão que permite demonstrar gosto por um certo conteúdo
Marketing	Definição da estratégia que será utilizada nas vendas, comunicações e no desenvolvimento do negócio
Markup	Sequência de caracteres ou símbolos que se inserem num documento para indicar ao *web browser* como processá-los e exibi-los
Middleware	Camada de *software* que fornece serviços às aplicações, para além daqueles disponíveis pelo sistema operativo
Mockups	Conjunto de imagens estáticas que mostram ao utilizador uma representação das *interfaces* que uma determinada aplicação terá
Model	Representação da estrutura lógica de dados de uma aplicação de *software*
Modelo Entidade Associação	Modelo de representação gráfica das relações lógicas entre as entidades (ou objetos) de modo a criar uma base de dados
Open Source	Programa cujo código-fonte está disponível para o público geral permitindo a sua utilização e/ou modificação
Overhead	Processamento ou armazenamento em excesso, seja de tempo de computação, de memória ou qualquer outro recurso necessário para executar uma determinada tarefa
Overwrite	Processo de gravar ou copiar novos dados sobre os dados existentes

Termo	**Descrição**
Pipeline	Sistema de processos executados em série para um determinado conjunto de dados
Player	Programa responsável por reproduzir conteúdo multimédia
Plug-in	Componente de *software* que adiciona uma funcionalidade específica a um programa existente
Pop-up	Janela de visualização que abre no *browser* ao visitar uma página *web* ou ao aceder uma hiperligação específica
Pop-up Blocker	Programa que bloqueia o aparecimento indesejado de *pop-ups* na Internet
Pre-roll	Vídeo que executa antes que o conteúdo que o utilizador selecionou apareça
Query	Pedido de informação que reside numa base de dados
Schema	Define as tabelas, os campos e as relações entre as tabelas e os campos de uma base de dados
Script	Sequência de instruções que são executadas sem a interação do utilizador
Sketch	Vídeo de comédia de curta duração
Spam	*E-mails* não solicitados pelos utilizadores que são enviados em massa
Stack	Pilha de componentes arquiteturais e suas interligações
Streaming	Tecnologia de transferência de dados multimédia em tempo real, utilizando redes de computadores e a Internet
Template	Documento de conteúdo, com apenas a apresentação visual (nomeadamente cabeçalhos) e instruções sobre onde e qual tipo de conteúdo deve entrar a cada parcela da apresentação
Test Suite	Conjunto de casos de teste que pretendem testar um programa de *software* de modo a a demonstrar que tem um conjunto específico de comportamentos
Text Ad	Anúncio publicitário em formato texto
Text Link	Hiperligação utilizada dentro de um documento que referencia outras partes desse documento ou outro documento
Thumbnail	Versão reduzida de uma imagem, usada para facilitar a sua procura e o seu reconhecimento
User Experience	Reflete a experiência total de uma pessoa ao utilizar um determinado produto, sistema ou serviço
User Stories	Descrição das funcionalidades de uma aplicação segundo a perspetiva do utilizador
VAST	Sigla para Video Ad Serving Template. É um padrão de requisitos de comunicação entre os servidores de anúncios e os *video players*
Video Ad	Anúncio publicitário em formato vídeo
View	Representação visual do Modelo de Dados
Web Designer	Desenvolver da estrutura, da apresentação gráfica e do conteúdo de *websites*
Website	Conjunto de páginas *web* acessíveis geralmente pelo protocolo HTTP na Internet
World Wide Web	Representa um sistema de documentos em hipermédia que são interligados e executados na Internet. Também designado por Web ou WWW

Anexo A

User Stories

Story 1

As a user, I want to login to the app, so I can work.

Acceptance criteria: login with username and password

Story 2

As a user, I want to see a list of all active ad units after logging in (the dashboard), so that I can see very clearly which ad units are active.

Story 3

As a user, I want to be able to create a new ad unit right from the dashboard, so that I can create it right after logging in.

Story 4

As a user, I want to be able to deactivate an ad unit, so that I can make sure it's not delivered anymore.

Story 5

As a user, I want to create a new ad unit from a list of ad unit templates, so that I can then configure the necessary parameters for that ad unit.

Story 6

As a user, I want to publish a new ad unit (a published ad unit is an active ad unit).

Story 7

As a user, I want to be able to delete an active or inactive ad unit.

Anexo B

Mockups da Interface Gráfica

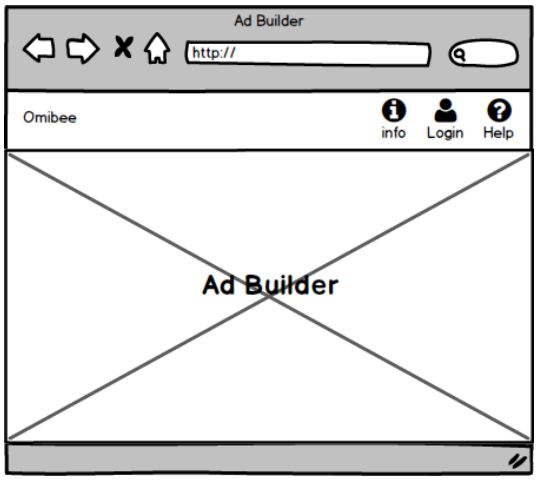

Figura B.1: Mockup da Página de Entrada

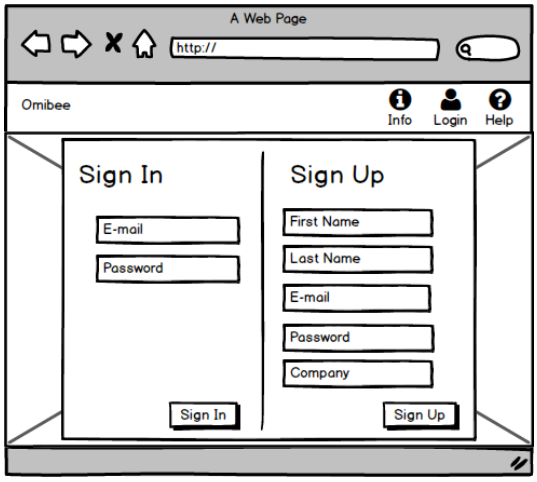

Figura B.2: Mockup dos Modais do *Login* e do *Sign Up*

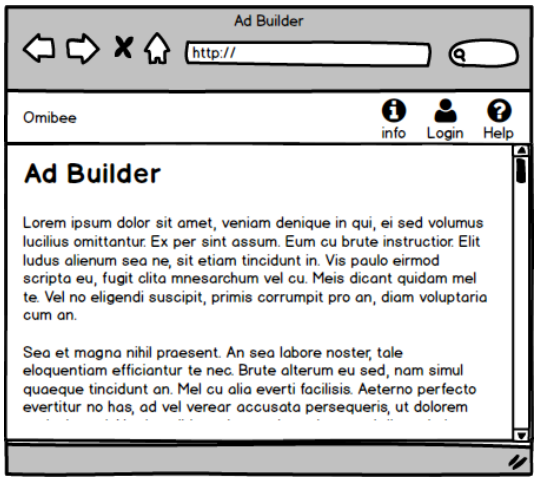

Figura B.3: Mockup da Página de Informação

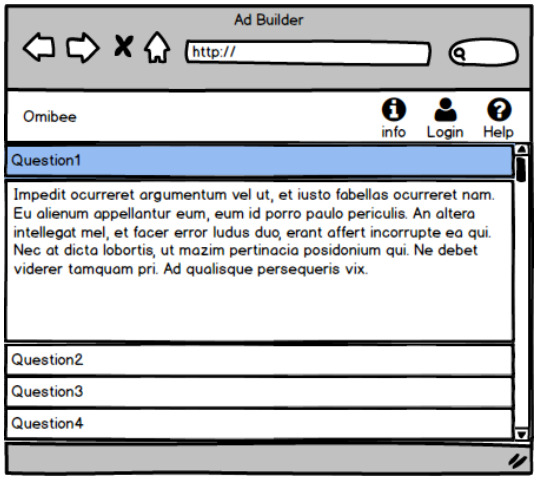

Figura B.4: Mockup da Página de Ajuda

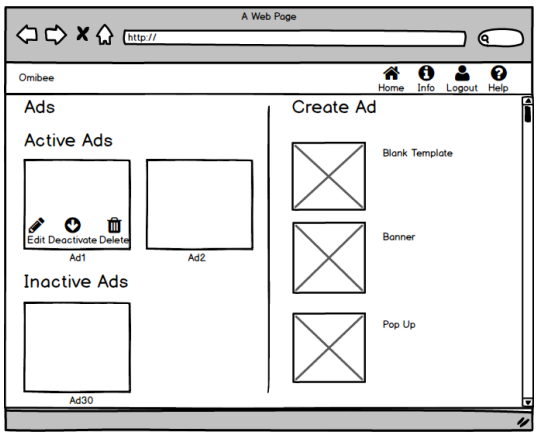

Figura B.5: Mockup da *Homepage*

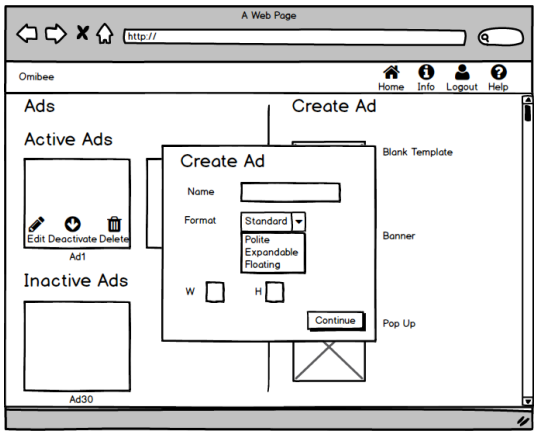

Figura B.6: Mockup do Modal da Criação de um Anúncio

Anexo C

Script de Instalação e Configuração

```
#!/usr/bin/env bash

export LANGUAGE=en_US.UTF-8

export LANG=en_US.UTF-8

export LC_ALL=en_US.UTF-8

export DEBIAN_FRONTEND=noninteractive

locale-gen en_US.UTF-8

dpkg-reconfigure locales

apt-get update


# Install apache

apt-get -y install apache2


# Install mariadb

apt-get -y install mariadb-server


# Install php

apt-get -y install php5 php5-cli libapache2-mod-php5 php5-mcrypt
```

```
php5-gd php5-curl

# Configure mysql root user
mysql -u root mysql -e 'update user set password=PASSWORD("root") where User="root";'
service mysql restart
mysql -u root –password=root -e 'create database adbuilder;'

# Install phpmyadmin
apt-get -y install phpmyadmin

# Install NodeJS
curl -sL https://deb.nodesource.com/setup | sudo bash -
apt-get install -y nodejs
apt-get install -y build-essential

# Install Phantom and CasperJS
npm install -g phantomjs-prebuilt
npm install -g casperjs

# Config apache
echo 'Listen 8000' >> /etc/apache2/ports.conf
cp /vagrant/config/default.conf
/etc/apache2/sites-available/000-default.conf
cp /vagrant/config/phpmyadmin.conf
/etc/apache2/sites-available/001-phpmyadmin.conf
a2ensite 001-phpmyadmin
a2enmod rewrite
```

```
service apache2 restart

# Install and run composer

curl -sS https://getcomposer.org/installer | php

mv composer.phar /usr/local/bin/composer

ln -s /usr/local/bin/composer /usr/bin/composer

cd /vagrant/www

php bin/console doctrine:schema:update –force

php bin/console doctrine:fixtures:load

php bin/console cache:clear –env=dev

php bin/console cache:warmup –env=dev

composer -n install

sed -i 's/database_port: null/database_port: 3306/' /vagrant/www/app/config/
parameters.yml

sed -i 's/database_password: null/database_password: root/' /vagrant/www/app/
config/parameters.yml

sed -i 's/database_name: symfony/database_name: adbuilder/' /vagrant/www/app/
config/parameters.yml
```

Anexo D

Teste unitário da Criação de um Anúncio

```
casper.test.tearDown(function(){
    casper.page.setCookies('');
});

casper.test.begin('Create ad test', 10, function suite(test){
    casper.start('http://127.0.0.1:3000/', function() {
         test.assertExists('#a_login', "login link is found");
    });

    casper.then(function(){
        this.clickLabel('Login', 'a');

        casper.waitFor(function check() {
            return this.evaluate(function() {
                return document.querySelectorAll('#form_login_form').length == 1;
            });
        }, function then() {
                test.assertExists('#form_login_form #form_email');
                test.assertExists('#form_login_form #form_password');
                this.fill('#form_login_form', {
                   'form[email]': 'sarasousa@gmail.com',
                   'form[password]': '123456'
                }, true);
            }, function(){
                this.echo('Failed to reach the login modal', 'ERROR');
            }, 5000);
    });


    casper.waitForUrl('/home', function(){
        test.assertTextExists('Ad Units', "Ad Units is found");
        test.assertExists('#create_ad');
        this.click('#create_ad');
    }, function(){
        this.echo('Failed to get to create ad page', 'ERROR');
    }, 30000);
```

```
   casper.waitForUrl('/create', function(){
      test.assertTextExists('Choose a Template', "Choose a Template is found");
      test.assertExists('#blank_template');
      this.click('#blank_template');
  }, function(){
      this.echo('Failed to select the template', 'ERROR');
  }, 5000);

   casper.waitFor(function check() {
          return this.evaluate(function() {
              return document.querySelectorAll('#form_create_form').length == 1;
          });
      }, function then() {
              test.assertTextExists('Create Ad', "Create Ad is found");
                this.fill('#form_create_form', {
                   'form[file]': 'src/casper-test/birds.png',
                   'form[name]': 'Ad Twitter',
                   'form[company]': 'Twitter',
                   'form[height]': '500',
                   'form[width]': '700',
                   'form[product]': 'infeed',
                   'form[autoplay]': '1',
                   'form[outstream]': '0',
                   'form[unique_id]': '55555555',
                   'form[publisher_key]': '444444444',
                   'form[adspeedID]': '333333',
                   'form[dfpID]': '111111111',
                   'form[adtag]': 'pijjiojrejrehjrjrfjefjejdjvcuVBWIBECEWDBQWBIUCBEBWCI',
                   'form[orientation]': 'left',
                   'form[demo]': '0',
                   'form[volume]': '0.8'
                }, true);
            }, function(){
                this.echo('Failed to create ad', 'ERROR');
            }, 5000);

    casper.waitForUrl('/home', function(){
        test.assertTextExists('Ad Units', "Ad Units is found");
        test.assertExists('.alert-success');
    }, function(){
        this.echo('Failed to get to home page', 'ERROR');
    }, 50000);

    casper.run(function(){
        test.done();
    });
});
```

Anexo E

Questionário relativo à gestão dos anúncios e Interação Omibee-Adklip

Ad Management and Omibee Interaction

This survey focuses on the ongoing interaction between two companies - Adklip and Omibee - in ad management.

*Obrigatório

Select your Department: *

- ○ System Architect
- ○ Consultant
- ○ Project Manager
- ○ Programer / Developer

Choose the level of complexity of ad deployment (from creation to publishing) *

	1	2	3	4	5	6	7	8	9	
Difficult	○	○	○	○	○	○	○	○	○	Easy

Describe the amount of time spent in each ad deployment *

	1	2	3	4	5	6	7	8	9	
Too Long	○	○	○	○	○	○	○	○	○	Just Right

How do you feel about the current ad management workflow? *

	1	2	3	4	5	6	7	8	9	
Frustrating	○	○	○	○	○	○	○	○	○	Satisfying

How well does the current system handle multiple clients? *

	1	2	3	4	5	6	7	8	9	
Inadequately	○	○	○	○	○	○	○	○	○	Adequately

Which parts of the deployment pipeline do you feel perform the worst? *

- ☐ Ad Creation
- ☐ Ad Publishing
- ☐ Ad Unpublishing
- ☐ Ad Editing
- ☐ Ad Deleting
- ☐ Ad Version Roll-Back

Do publishers often report ad malfunctioning? *

	1	2	3	4	5	6	7	8	9	
Always	○	○	○	○	○	○	○	○	○	Never

How fast can you respond to clients' requests? *

	1	2	3	4	5	6	7	8	9	
Very untimely	○	○	○	○	○	○	○	○	○	Timely enough

Classify your Javascript programming skills *

- ○ Non-existant
- ○ Minimal (can understand code)
- ○ Elementary (can edit code)
- ○ Average (can develop from scratch)
- ○ Advanced (can develop professional applications)

How dependent do you feel on Omibee's technical support? *

	1	2	3	4	5	6	7	8	9	
Very	○	○	○	○	○	○	○	○	○	Not at all

Which parts of the deployment pipeline require more of Omibee's attention? *

- ☐ Ad Creation
- ☐ Ad Publishing
- ☐ Ad Unpublishing
- ☐ Ad Editing
- ☐ Ad Deleting
- ☐ Ad Version Roll-Back

Describe your impressions of the current system

A sua resposta

Anexo F

Questionário relativo à *user experience* do *website*

AdBuilder

This survey focuses on the user experience of the AdBuilder website.

*Obrigatório

Select your Company: *

- ○ Adklip
- ○ Omibee

Select your Department: *

- ○ Programer/Developer
- ○ Manager
- ○ Consultant
- ○ Web Designer
- ○ Marketing

I think I would like to use this website frequently. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

I found the website unnecessarily complex. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

I thought the website was easy to use. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

I think I would need Tech Support to be able to use this website. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

I found the various functions in this website were well integrated. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

I thought there was too much inconsistency in this website. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

I found the website very cumbersome to use. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

I felt very confident using the website. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

I need to learn a lot about this website before I could effectively use it. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

Describe your opinion of the AdBuilder website. *

A sua resposta

Referências

[1] "When mad men meets flash boys." http://bit.ly/294sTql. [Online; acedido a 07-Fevereiro-2016].

[2] "The rise of the chief data officer." http://www.liesdamnedlies.com/adserving/. [Online; acedido a 05-Fevereiro-2016].

[3] "Programmatic advertising: Deconstructed." http://bit.ly/28Uam1A. [Online; acedido a 07-Fevereiro-2016].

[4] "Model–view–controller." http://eimschools.com/cms/documents/MVC_wiki.pdf. [Online; acedido a 24-Maio-2016].

[5] "Http (hypertext transfer protocol)." http://bit.ly/1QXdkx7. [Online; acedido a 06-Junho-2016].

[6] "Code http: The protocol every web developer must know - part 1." http://bit.ly/1dgu8kA. [Online; acedido a 06-Junho-2016].

[7] B. Baccot, V. Charvillat, and R. Grigoras, "A bandit's perspective on website adaptation." http://ceur-ws.org/Vol-539/paper_24.pdf. [Online; acedido a 26-Outubro-2015].

[8] M. Belc, "Online media." http://www.serviceplan.com/en/online-media.html. [Online; acedido a 25-Outubro-2015].

[9] W. Nichols, "Advertising analytics 2.0." https://hbr.org/2013/03/advertising-analytics-20. [Online; acedido a 18-Novembro-2015].

[10] "Virtualization is not enough, says enterprise software executive." bit.ly/1O5dBv6. [Online; acedido a 19-Novembro-2015].

[11] "How does third-party ad serving work?." http://www.allaboutcookies.org/ad-serving/. [Online; acedido a 18-Novembro-2015].

[12] "Guide to online advertising." http://bit.ly/1O5rHg2. [Online; acedido a 07-Fevereiro-2016].

[13] "The top 16 best video ad networks to improve your earnings." http://bit.ly/1QYPS4n. [Online; acedido a 05-Fevereiro-2016].

[14] "Advertising campaigns - meaning and its process." http://bit.ly/1Tf8CQv. [Online; acedido a 26-Novembro-2015].

[15] "Video ad networks." http://www.wordstream.com/video-ad-networks. [Online; acedido a 05-Fevereiro-2016].

[16] https://omibee.com/.

[17] "adklip." https://omibee.com/pt/work/adklip. [Online; acedido a 07-Junho-2016].

[18] "Our products." http://adklip.com/our-products/. [Online; acedido a 06-Junho-2016].

[19] J. Barr, "New - cross-region replication for amazon s3." http://amzn.to/1KIJcbo. [Online; acedido a 19-Novembro-2015].

[20] "Dynamic ads." http://www.flashtalking.com/uk/dynamic-ads/. [Online; acedido a 10-Fevereiro-2016].

[21] "Overview build an html5 ad in ad builder for html5." bit.ly/1RtCYxl. [Online; acedido a 10-Fevereiro-2016].

[22] "Dive into html5 with celtra's adcreator." https://www.celtra.com/adcreator. [Online; acedido a 10-Fevereiro-2016].

[23] T. Hopkins, "What are the benefits of using a cdn?." http://bit.ly/20OPFJ8. [Online; acedido a 26-Novembro-2015].

[24] "Advantages and disadvantages of using a content delivery network." http://www.cdnreviews.com/cdn-advantages-disadvantages/. [Online; acedido a 26-Novembro-2015].

[25] A. Karbasfrooshan, "The rise of video ad networks." http://tcrn.ch/1LjGALW. [Online; acedido a 05-Fevereiro-2016].

[26] "Rise and fall of online advertising." http://1stwebdesigner.com/online-advertising-history/. [Online; acedido a 04-Fevereiro-2016].

[27] "Spam brand history." http://www.spam.com/about. [Online; acedido a 07-Fevereiro-2016].

[28] A. Oberoi, "The history of online advertising." http://bit.ly/12D7jCP. [Online; acedido a 04-Fevereiro-2016].

[29] "Infographic: the history of online advertising." http://bit.ly/1QYPQJR. [Online; acedido a 04-Fevereiro-2016].

[30] "A brief history of online advertising." bit.ly/1nYIRXy, Janeiro 2013. [Online; acedido a 04-Fevereiro-2016].

[31] T. Donnelly, "How to manage your online advertising." http://bit.ly/1V7OUFN. [Online; acedido a 26-Outubro-2015].

[32] "The best ad networks: Display advertising & more." http://bit.ly/28TzHEg. [Online; acedido a 05-Fevereiro-2016].

[33] "Video viewability: Measuring the measures." bit.ly/1Q9cBOd. [Online; acedido a 05-Fevereiro-2016].

[34] "Audience measurement on trial: Online video metrics." bit.ly/1msTlNP. [Online; acedido a 05-Fevereiro-2016].

[35] Andrea, "The three types of banner ads you should know about." bit.ly/1TSGSk1. [Online; acedido a 07-Fevereiro-2016].

[36] "Different types of online advertising." http://bit.ly/1SKNnGF. [Online; acedido a 07-Fevereiro-2016].

[37] "Types of internet advertising." bit.ly/1eKNgZD. [Online; acedido a 07-Fevereiro-2016].

[38] "Html (hypertext markup language)." http://searchsoa.techtarget.com/definition/HTML. [Online; acedido a 14-Maio-2016].

[39] "cascading style sheet (css)." http://searchsoa.techtarget.com/definition/cascading-style-sheet-CSS. [Online; acedido a 14-Maio-2016].

[40] "Javascript." http://bit.ly/1XT2QqO. [Online; acedido a 14-Maio-2016].

[41] "Javascript: advantages and disadvantages." http://www.jscripters.com/javascript-advantages-and-disadvantages/. [Online; acedido a 14-Maio-2016].

[42] "What is php?." http://www.homeandlearn.co.uk/php/php1p1.html. [Online; acedido a 14-Maio-2016].

[43] "13 php frameworks to help build agile applications." http://on.mash.to/28zXiog. [Online; acedido a 14-Maio-2016].

[44] "Twig, smarty (template engines for php)." http://nileshgpangul.blogspot.pt/2016/03/twig-template-engine-for-php.html. [Online; acedido a 14-Maio-2016].

[45] "Sql (structured query language)." http://bit.ly/1XkVbAB. [Online; acedido a 14-Maio-2016].

[46] "Mysql." http://bit.ly/1XT3Yui. [Online; acedido a 14-Maio-2016].

[47] "Five advantages & disadvantages of mysql." https://www.datarealm.com/blog/five-advantages-disadvantages-of-mysql/. [Online; acedido a 14-Maio-2016].

[48] "10 reasons to migrate to mariadb (if still using mysql)." http://bit.ly/20pclvI. [Online; acedido a 24-Maio-2016].

[49] "Bringing mysql to the web." https://www.phpmyadmin.net/. [Online; acedido a 24-Maio-2016].

[50] "Definition object-relational mapping (orm)." http://bit.ly/1thNJdz. [Online; acedido a 24-Maio-2016].

[51] "Introduction to models." http://bit.ly/1UAFnDn. [Online; acedido a 20-Maio-2016].

[52] "O modelo de dados." http://bit.ly/1RXeRCx. [Online; acedido a 24-Maio-2016].

[53] "What is symfony." http://symfony.com/what-is-symfony. [Online; acedido a 24-Maio-2016].

[54] "Symfony components." http://symfony.com/components. [Online; acedido a 24-Maio-2016].

[55] "Symfony versus flat php." http://bit.ly/1kSKW01. [Online; acedido a 24-Maio-2016].

[56] "Controller." http://symfony.com/doc/current/book/controller.html. [Online; acedido a 06-Junho-2016].

[57] "Understanding model-view-controller." http://bit.ly/1rdm8rY. [Online; acedido a 24-Maio-2016].

[58] "model-view-controller (mvc)." http://whatis.techtarget.com/definition/model-view-controller-MVC. [Online; acedido a 24-Maio-2016].

[59] "Databases and doctrine." http://symfony.com/doc/current/book/doctrine.html. [Online; acedido a 24-Maio-2016].

[60] "About." http://getbootstrap.com/about/. [Online; acedido a 06-Junho-2016].

[61] "Http: The protocol every web developer must know - part 1." http://bit.ly/1dgu8kA. [Online; acedido a 06-Junho-2016].

[62] "An introduction to apache." http://code.tutsplus.com/tutorials/an-introduction-to-apache--net-25786. [Online; acedido a 06-Junho-2016].

[63] "Web server." http://www.webopedia.com/TERM/W/Web_server.html. [Online; acedido a 06-Junho-2016].

[64] "What is apache http server and what is it used for?." http://bit.ly/22W2puP. [Online; acedido a 06-Junho-2016].

[65] "What is a web server?." http://mzl.la/24D8Emr. [Online; acedido a 06-Junho-2016].

[66] "What is: Apache." http://www.wpbeginner.com/glossary/apache/. [Online; acedido a 06-Junho-2016].

[67] "November 2015 web server survey." http://news.netcraft.com/archives/2015/11/16/november-2015-web-server-survey.html. [Online; acedido a 06-Junho-2016].

[68] M. Rouse, "Amazon web services (aws)." http://bit.ly/20OQ66f. [Online; acedido a 05-Fevereiro-2016].

[69] R. Miller, "Estimate: Amazon cloud backed by 450,000 servers." bit.ly/1RtGzLR. [Online; acedido a 18-Novembro-2015].

[70] "Computação em nuvem com a amazon web services." https://aws.amazon.com/pt/what-is-aws/. [Online; acedido a 05-Fevereiro-2016].

[71] "Amazon ec2 – hospedagem de servidor virtual." https://aws.amazon.com/pt/ec2/. [Online; acedido a 06-Junho-2016].

[72] "Amazon ec2 instance." http://bit.ly/1XT44lR. [Online; acedido a 06-Junho-2016].

[73] "Aws elastic beanstalk." http://amzn.to/1Ur8Ns3. [Online; acedido a 06-Junho-2016].

[74] "Amazon simple storage service (amazon s3)." http://bit.ly/21blckn. [Online; acedido a 06-Junho-2016].

[75] "Detalhes do produto amazon rds." https://aws.amazon.com/pt/rds/details/. [Online; acedido a 06-Junho-2016].

[76] "Aws codedeploy." https://aws.amazon.com/pt/codedeploy/. [Online; acedido a 06-Junho-2016].

[77] "Amazon cloudfront." https://aws.amazon.com/pt/cloudfront/. [Online; acedido a 25-Outubro-2015].

[78] J. McIlwain, "How content delivery networks work." http://bit.ly/1TVpTO9. [Online; acedido a 17-Novembro-2015].

[79] "Getting started - a short history of git." http://bit.ly/25N22I6. [Online; acedido a 30-Maio-2016].

[80] "Using github to share with sparkfun." http://bit.ly/25NtCS0. [Online; acedido a 30-Maio-2016].

[81] "What is phantomjs and how is it used?." http://bit.ly/1taXiu5. [Online; acedido a 30-Maio-2016].

[82] "Casperjs." http://casperjs.org/. [Online; acedido a 30-Maio-2016].

[83] "Our features." https://codeship.com/features. [Online; acedido a 30-Maio-2016].

[84] "Usando o vagrant como ambiente de desenvolvimento no windows." http://bit.ly/1Y9Gu2h. [Online; acedido a 24-Maio-2016].

[85] "What is adobe illustrator used for? understanding vector images." https://blog.udemy.com/what-is-adobe-illustrator-used-for/. [Online; acedido a 30-Maio-2016].