INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO

PEDRO MIGUEL BAPTISTA DA SILVA MARQUES VALENTE

ADS4SHOPS
PLATAFORMA DE GESTÃO DE ANÚNCIOS DE VIDEO

Mestrado em Engenharia de Software

Trabalho de Projeto efetuado sob a orientação de

Doutor Jorge Ribeiro

Mestre Pedro Carneiro

Julho de 2016

AGRADECIMENTOS

Primeiramente agradeço à família, pela força e encorajamento demonstrados ao longo do mestrado. Obrigado pela vossa compreensão nos longos períodos de ausência.

Aos meus orientadores Doutor Jorge Ribeiro e Mestre Pedro Carneiro, pela disponibilidade demonstrada e conselhos preciosos ao longo de todo o percurso.

Agradeço à comunidade *Laravel* e outras comunidades, pois sem o seu contributo, este projeto nunca seria possível.

Aos meus colegas de curso, pela ajuda prestada.

Muito obrigado a todos!!

RESUMO

Ao longo dos últimos anos, é notória a evolução da publicidade digital, quer pela quantidade de público que atinge, quer pela baixa de preço dos materiais envolvidos. Podemos afirmar que se trata de uma das formas de conquistar o consumidor pelo olhar, onde se pode receber mais de 85% das informações do mundo exterior, as tecnologias visuais modernas permitem induzir o consumo. Por outro lado, a sinalética digital presente em aeroportos, supermercados, lojas de moda e cafés, já faz parte de nossa vida de forma efetiva. Painéis eletrónicos com horários de voos substituíram seus pares eletromecânicos, muito mais complexos e caros, assim como os carrinhos de supermercado possuem ecrãs promovendo produtos, no momento da decisão da compra, orientando o consumidor e aumentando as vendas. Neste sentido, as lojas de moda veiculam seus desfiles e coleções, mostrando sapatos e bolsas, de uma maneira muito mais dinâmica e glamorosa. Cartazes e *banners*, que antes eram feitos em papel ou vinil, estão sendo gradualmente substituídos por ecrãs com animações dinâmicas, melhorando a visibilidade e a atratividade. São os *banners* digitais. Em termos de conteúdos digitais, estes são a essência deste novo ponto de venda, uma vez que permite a estes estabelecimentos enriquecerem-se com vídeos, fotos e textos rápidos e eficazes, para que o consumidor reaja positivamente. Todo um novo conceito de design está a ser criado no mundo visual eletrónico, aonde o que se busca não é apenas a beleza, mas impacto e conversão de vendas. O que está em jogo é a faturação e a eficiência da mensagem perante o consumidor. Contudo, o salto tecnológico não para aí, há muito mais disponível. Os conteúdos podem ser alimentados remotamente, pela internet, através de uma única central de gestão e distribuição. A grosso modo, qualquer empresa pode ter sua TV corporativa. Neste contexto, o presente projeto foi desenvolvido para ser uma plataforma *online* centralizada que permita a um conjunto de lojas aderentes, publicar e publicitar anúncios em vídeo que serão distribuídos e apresentados pelas restantes lojas aderentes. Em termos de desenvolvimento da aplicação foram seguidas as orientações da metodologia de investigação Design Research, e foram considerados importantes aspetos como: segurança, usabilidade e funcionalidade do sistema. Como resultado final foi desenvolvido um sistema que permitiu de uma forma centralizada fazer a gestão de

anúncios em vídeos, e que depois de uma forma aleatória são reproduzidos em lojas aderentes, todo o sistema desenvolvido é acedido via *web* e a reprodução é feita através de um *web browser*. Tendo em conta o período temporal de desenvolvimento do projeto conseguiu-se atingir os objetivos propostos. Como trabalho futuro, pretende-se desenvolver ainda mais a plataforma, criando um sistema de registos de passagem de vídeos, e um conjunto hardware/software que facilmente possa ser instalado nas lojas aderentes ao sistema, com o mínimo de intervenção possível.

Palavras-chave: publicidade digital, TV corporativa, sinalética digital

ABSTRACT

Over the past few years, developments of digital advertising is notoriously either by the amount of audience that reaches either the low price of the materials involved. We can win the consumer by the look, where we received more than 85% of information from the outside world, modern visual technologies allow induce consumption. On the other hand, digital signage presente in airports, supermarkets, fashion shops and cafes, is already part of our lives effectively. Electronic panels with departures, replaced their electromechanical pairs, much more complex and expensive, supermarket trolleys have screens promoting products at the time of purchase decision, guiding the consumer and increasing sales. In this sense, fashion shops convey his fashion shows and collections, showing shoes and bags, of a much more dynamic and glamorous way. Posters and banners, which were previously done on paper or vinyl, are gradually being replaced by screens with dynamic animations, improving the visibility and attractiveness. They are digital banners. In terms of digital content, these are the essence of this new point of sale, since it allows these establishments to enrich themselves with videos, photos and texts fast and effective, so that consumers react positively. A whole new design concept is being created in the electronic visual world, where what is sought is not only the beauty, but impact and sales conversion. What is at stake is the billing and the message to the consumer efficiency. However, the technological leap does not stop there, there is much more available. The contents can be powered remotely via the Internet, through a central management and distribution only. Roughly speaking, any company can have its corporate TV. In this context, this project is designed to be a centralized online platform that allows a set of participating stores, publish and advertise video ads that will be distributed and presented by the other participating stores.

In developing the application were followed the Design Research methodology guidelines, and were considered important aspects such as: security, usability and functionality of the system. As a final result, was developed a system that allowed a centralized way to the video ad management, and after a random play in participating stores, throughout the developed system is accessed via web and reproduction is done through a web browser.

Taking into account the time period of project development it managed to achieve the proposed goals. As a future work, we intend to develop the platform further, creating a logging system of playback videos, and a hardware/software set that can be easily installed in participating stores to the system with, the least possible intervention.

Keywords: digital advertising, corporate TV, digital signage

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

DEFINIÇÕES, ACRÓNIMOS E ABREVIATURAS

ACL	**A**ccess **C**ontrol **L**ist
API	**A**pplication **P**rogramming **I**nterface
Artisan	**Artisan** é uma interface em linha de comando, incluído no Laravel. Fornece um conjunto de comandos úteis, que podem ajudar na construção da aplicação.
BootStrap	Framework de front-end, free and open-source collection of tools for creating websites and web applications
CRUD	**C**reate, **R**ead, **U**pdate e **D**elete
CSS	**C**ascade **S**tyle **S**heets
HTML	**H**yper**T**ext **M**arkup **L**anguage
JavaScript	Linguagem de programação interpretada
JSON	**J**ava**S**cript **O**bject **N**otation
Laravel	Laravel é um *framework* PHP para desenvolvimento web que utiliza a arquitetura MVC.
LED	**L**ight **E**mitting **D**iode, díodo emissor de luz.
LCD	**L**iquid **C**rystal **D**isplay
MVC	**M**odel **V**iew **C**ontrol
OLED	**O**rganic **L**ight **E**mitting **D**iode**,** díodo emissor de luz orgânico
ORM	**O**bject-**r**elational **m**apping
PHP	Um acrónimo recursivo para *"**P**HP: **H**ypertext **P**reprocessor"* , linguagem de programação.
REST	**Re**presentational **S**tate **T**ransfer
SMTP	**S**imple **M**ail **T**ransfer **P**rotocol
URL	**U**niform **R**esource **L**ocator – Localizador Padrão de Recursos
XAMPP	**A**pache, **M**ySQL, **P**HP e **P**earl

1 INTRODUÇÃO

1.1 CONTEXTO

Em qualquer civilização, as formas simbólicas assumem um papel fundamental à medida que a linguagem se desenvolve. A comunicação torna-se então a relação básica para essas trocas simbólicas. Atualmente, com o avanço tecnológico, o seu papel passa a ser cada vez mais central e importante. Houve um tempo em que a produtividade estava ligada à terra, depois à indústria. Hoje, o desenvolvimento está diretamente ligado à informação. Logo, se o desenvolvimento depende da informação, a comunicação bem como a publicidade são essenciais para que as mudanças sociais aconteçam. A evolução do comportamento do homem contemporâneo com os meios de comunicação ocorre também com as transformações dos meios de comunicação, as quais ele está em contato, direto ou indiretamente (Alves, 2010).

Fazer publicidade digital até há muito pouco tempo, não era viável ou a relação custo-benefício não era boa, uma vez que os ecrãs eram muito caros e o retorno do investimento era muito baixo. Contudo, com a revolução dos LCD/Plasma mudou e está a mudar tudo isso. Os ecrãs são agora mais acessíveis, e podem rivalizar com os preços da impressão dos posters estáticos. Os ecrãs são finos, podem ser pendurados numa parede, podem comunicar com uma rede de computadores e apresentar novos conteúdos dinamicamente, eliminando os velhos dias em que os empregados tinham de ir de ecrã em ecrã carregados de cassetes de vídeo/dvds.

Devido à contínua descida dos preços dos ecrãs, estão a ser instalados ecrãs digitais em muitos espaços públicos ao ar livre e em espaços comerciais. Os sistemas existentes de publicidade em locais públicos, geralmente tem como característica um único ponto de controlo que é responsável pelo agendamento da apresentação dos conteúdos na rede e dispõem de sofisticados mecanismos para controlar a apresentação e a ordem dessa apresentação. A utilização de sistemas de difusão de anúncios em plataformas digitais, quer de gestão centralizadas ou simplesmente locais, existem já há alguns anos e são largamente utilizadas em centros comerciais, outdoors, estabelecimentos públicos, aeroportos, lojas, etc., e geralmente requerem grandes configurações, hardware específico e tem bastantes restrições no uso. Por outro lado, os sistemas de difusão de anúncios na sua maioria não são gratuitos.

Por outro lado, com a proliferação dos sistemas de exploração na web e a diminuição dos custos de alojamento de servidores web, é possível criar um sistema centralizado para gestão de anúncios em tempo real, com bastante segurança e com custos reduzidos.

1.2 OBJETIVOS

Tendo em consideração o contexto na seção anterior e neste sentido, a criação de plataformas de anúncios centralizada via *web browser*, torna-se uma ferramenta poderosa para as empresas, quer em termos de divulgação de anúncios, quer em termos de difusão massiva de divulgação em grande escala desses mesmos anúncios. Assim proposemos a criação de uma plataforma centralizada que permitisse a um conjunto de lojas aderentes, publicar e publicitar de forma distribuída, anúncios de vídeo, e visualizá-los em tempo real em várias lojas.

Este projeto tem como objetivo a criação de uma plataforma web, que permita gerir a visualização de anúncios em *streaming* através de um browser compatível com HTML 5 e *Bootstrap* [1] (para permitir a sua visualização ajustada em equipamentos móveis), explorando as potencialidades de uma framework, do padrão de arquitetura de software MVC e a programação na linguagem em PHP.

1.3 METODOLOGIA DE DESENVOLVIMENTO

Este trabalho não se centra num puro estudo de investigação científica, mas antes no desenvolvimento e exploração e um projeto de aplicabilidade específica, aproveitando os conhecimentos adquiridos ao longo do curso de Mestrado.

Em termos de metodologias de investigação seguimos a orientação da metodologia de investigação *Design Research* (Mason, 2006) (Järvinen, 1997) a qual pressupõe a ação do investigador numa determinada realidade, permitindo-lhe compreender um problema e mediante isso propor a construção de uma solução (artefacto) e proceder ao teste da mesma, passando o investigador de um mero observador para um indivíduo que age no contexto investigado/aplicado, procurando compreender uma determinada realidade, utilizando o seu potencial criativo para criar soluções para problemas ou necessidades reais.

Com base nesta orientação, a metodologia que seguimos foi a seguinte:

- Estudar e Analisar aplicações já desenvolvidas;
- Definir os requisitos da plataforma;
- Implementar a plataforma;
- Testar e analisar o desempenho;
- Apresentar uma discussão critica sobre a plataforma desenvolvida;
- Apresentar as conclusões e o trabalho futuro;

[1] http://getbootstrap.com/

1.4 ESTRUTURA DO DOCUMENTO

Este trabalho encontra-se dividido nos seguintes capítulos:

No presente capítulo é feito um enquadramento do tema escolhido e dos objetivos propostos para o projeto. No segundo capítulo apresentamos as definições e conceitos associados ao projeto assim como algumas aplicações similares já desenvolvidas. No terceiro capítulo apresentamos os requisitos da plataforma e no quarto o desenvolvimento do projeto. No quinto apresentamos a avaliação e discussão dos resultados do trabalho realizado e no capítulo seis as conclusões e trabalho futuro. No capítulo das referências apresentamos as ligações e documentos que sustentaram tecnicamente este trabalho.

2 TECNOLOGIAS, CONCEITOS E APLICAÇÕES DE REFERÊNCIA

2.1 INTRODUÇÃO

Este capítulo tem como objetivo apresentar os conceitos necessários à compreensão dos termos utilizados para apresentar as diferentes fases de desenvolvimento do projeto.

Existem alguns conceitos indispensáveis à compreensão deste projeto, tais como *digital signage (doranvante Sinalética Digital ou SD)*, arquitetura de *software* - *Model View Controller* (MVC), *streaming* de vídeo, reprodutor de vídeo HTML5, *fullscreen*.

Sinalética Digital, é um tipo de painel informativo tipicamente colocado em espaços públicos, usado normalmente para informar, publicitar ou simplesmente distrair. Sobre a sinalética clássica apresenta várias vantagens: pode mostrar animações e o conteúdo pode ser mudado com facilidade, adaptando-se ao contexto e audiência, até mesmo de modo interativo (Wikipédia, 2014).

A arquitetura da plataforma baseia-se no paradigma de computação cliente-servidor, mais especificamente na sua versão *web*, usando a arquitetura de *software* MVC como se ilustra na Figura 2.1, e explicado em 4.2.3 - MVC - Laravel (pág.35).

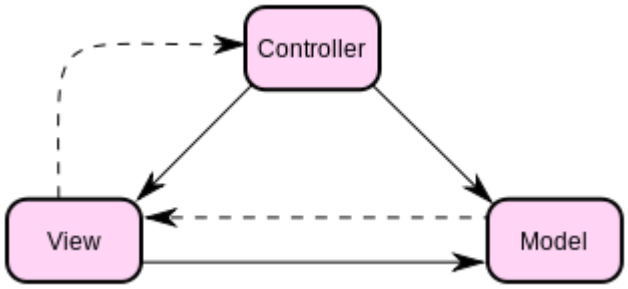

Figura 2.1 - Model View Controller

O termo *streaming* de vídeo é referido, pois a plataforma assenta na reprodução de vídeos em *streaming,* ou seja, os vídeos não são armazenados no cliente (a não ser uma pequena porção, denominada cache), mas sim à medida que vai chegando ao cliente e desde que a largura de banda o permita, são reproduzidos.

Foi usado HTML versão 5, porque permite criar aplicações quase universais, capazes de "rodar" numa grande variedade de dispositivos. "O HTML5 vai ajudar *smartphones, tablets*, PCs, televisores e veículos a convergir no futuro" (Mawston, 2011).

Neste sentido, estas tecnologias, conceitos foram usados na criação desta plataforma e serão detalhadas nas próximas secções.

2.2 TECNOLOGIAS E CONCEITOS

O *framework Laravel*[2] foi a principal tecnologia/ferramenta utilizada, pois é ela que liga todo este projeto.

Laravel é um *framework* PHP livre e *open-source,* criado por Taylor Otwell para o desenvolvimento de sistemas *web* que utilizam o padrão MVC *(model, view controller*). Algumas características proeminentes do *Laravel* são, a sintaxe simples e concisa, um sistema modular com gestor de dependências dedicado, várias formas de acesso a base de dados relacionais e vários utilitários indispensáveis na ajuda ao desenvolvimento e manutenção de sistemas (Wikipédia, 2016).

Apesar de existirem no mercado outros *frameworks* disponíveis, e após uma primeira experimentação de outros *frameworks*, como por exemplo *CodeIgnater* (Wikipédia, 2015) e *CakePHP* (Wikipédia, 2016), achamos que o *Laravel* era aquele que tinha melhor suporte na Web, encontrava-se mais desenvolvido e apresentava funcionalidades que eram necessárias para este projeto, tais como, autenticação, *RESTful Routing*[3], uma *template* agradável, entre outras. Também em fóruns da especialidade todos apontavam para que o *framework Laravel* fosse a mais atual, com uma curva de aprendizagem mais pequena, com mais suporte, e com uma comunidade de utilizadores mais ativa, fatores esses que nos levaram a optar pela *framewrok Laravel.*

Usamos a tecnologia de *streaming* de vídeo que permite que o cliente veja os vídeos sem ter de os descarregar na totalidade, o que não seria muito viável pois as listas de reprodução podem conter horas de vídeo.

O reprodutor de vídeo escolhido, é escrito em *Javascript* / HTML5 / CSS, para que possa correr em qualquer *web browser,* e para que seja facilmente configurável e adaptado às nossas necessidades.

O termo *fullscreen* é referido como a utilização de todo o ecrã do computador, televisão, ou outro dispositivo que permita visualizar vídeos, pois é nossa intenção que os vídeos sejam mostrados na totalidade do ecrã.

Como gestor de base de dados usamos o MySQL, por ser gratuito, *open-source*, rápido de configurar e estar presente em quase todos os alojamentos web.

[2] https://laravel.com/

[3] https://laravel.com/docs/5.1/controllers#restful-resource-controllers

2.3 *DIGITAL PLACED BASED NETWORKS*

As pessoas têm vindo a assistir a programas e publicidade confortavelmente nas suas salas de estar, a *digital placed based networks,* oferecem conteúdo de vídeo em ecrãs, localizados em lugares onde as pessoas têm tempo de permanência, como por exemplo *shopping centers*, bares, hospitais, consultórios médicos, zonas de espera de escritórios, elevadores, restaurantes, táxis, aeroportos, aviões, hotéis, estações de abastecimento de combustível e outros locais de muito tráfego.

Esses ecrãs oferecem vídeo digitalmente entregue, alguns são inclusive interativos. Os conteúdos são programados, para a envolvente, são conteúdos que informam, entretêm ou frequentemente ambos. Dá hipótese aos comerciantes de atingirem o publico alvo mais preciso, bem além do publico padrão.

2.4 DOOH

Os meios de comunicação "*digital out of home*" (DOOH), são um tipo de publicidade exterior que faz essencialmente qualquer tipo de publicidade e que atinge o consumidor enquanto o público-alvo está fora de casa. Isto está em contraste com a transmissão, impressão e publicidade na *Internet*.

Meios de Comunicação "*Digital Out of Home*" de acordo com definições internacionais, pode ser dividida em três setores:

Alto Impacto: Enormes monitores de LCD disponíveis em diferentes locais ao ar livre e que atinge peões e pessoas em trânsito.

Ponto de venda: Monitores instalados em pontos de venda como supermercados, lojas, restaurantes e *Shopping Centers*.

Audiência Cativa: Comunicação exibida num local específico, com *targets* definidos, onde o consumidor está disponível, como autocarros, metro, comboio, elevador, aeroporto, maternidade, casas de banho públicas, etc.

(Outdoor Advertising Association of America, Inc., 2015)

2.5 *DIGITAL SIGNAGE*

2.5.1 HISTÓRIA

A origem do Digital Signage, doravante referido por Sinalética Digital (SD) é muito antiga. Ao reconhecer ameaças e oportunidades na vida, as pessoas instintivamente reagem a sons e movimento. Partindo desta dinâmica, apareceram as primeiras noções de pessoas tentando chamar a atenção para vender produtos e serviços. Já no século XX, primeiramente a rádio e depois a televisão, percorreram um longo caminho para fundar os princípios da SD.

Um dos primeiros exemplos de SD com o mesmo sentido proposto neste trabalho, aconteceu no final da década de 70 do seculo passado, quando lojas de vestuário de Nova Iorque gravaram numa cassete de fita, os seus desfiles e reproduziram os mesmos em aparelhos de televisão no interior das suas lojas. Em 1984, no Canadá, uma cadeia de supermercados disponibilizou televisores direcionados aos seus funcionários (usado primeiramente para instruir e aumentar o comprometimento com as regras e diretrizes da empresa) e outros direcionados para os clientes (usado para promover e alavancar as vendas e como auxiliar na publicidade).

Nos anos 90, com o surgimento de ecrãs cada vez maiores, com mais resolução, mais leves e mais finos, os japoneses criaram os primeiros monitores de plasma específicos para sinalética digital. Quando o foco da SD realmente evoluiu de apenas mostrar *shows* e jogos, a indústria de sinalética cresceu vertiginosamente. Como primeiro exemplo de sucesso da utilização em larga escala da SD, podemos citar o caso dos casinos de Las Vegas, que instalaram 100 monitores de plasma no final de 1999 (Schaeffler, 2008, p. 42).

A primeira "*Digital Signage Expo"*, aconteceu na cidade de São Francisco nos E.U.A. em 2004 e consolidou o nome de "*Digital Signage"* para esse setor (Digital Signage Connetion, 2013).

A partir dos anos 2000, a SD difundiu-se como forma de *marketing* tanto institucional como de promoção para clientes chegando aos dias de hoje, onde ecrãs veiculando algum tipo de informação já fazem parte da nossa rotina. Com a redução dos preços da internet banda larga e dos monitores *LCD* e *LED*, rapidamente grandes redes de sinalética digital foram instaladas e cresceu a procura por conteúdo de qualidade em tempo real.

2.5.2 APLICAÇÃO

A Sinalética Digital é utilizada em variados segmentos de *marketing*, como no comércio a retalho (comunicação social voltada para divulgação interna e promoção no ponto de venda), na restauração e noutros serviços como bancos, lavandarias, postos de combustíveis e em locais públicos (aeroportos, centrais rodoviárias, shoppings, elevadores, táxis, autocarros, metro, casas de banho).

Por exemplo, podemos citar o caso de retalhistas como a *Media Markt* que usam a Comunicação Social - Digital Indoor para coordenar o seu jornal de ofertas semanais e para divulgar produtos novos.

Os cinemas utilizam para exibir anúncios e exibição de *trailers* antes de iniciar a sessão de cinema. As empresas de alimentação do tipo *fast-food* utilizam como menus digitais para incentivar as vendas. Empresas e órgãos governamentais utilizam a SD para motivar e treinar os seus colaboradores no próprio local de trabalho. Diversas empresas utilizam a SD em feiras e eventos para divulgar os seus produtos e serviços.

Um conceito interessante, explorado pela ABDOH (Associação Brasileira de Mídia Out of Home, 2015), é que a sinalética digital pode possuir as características da Ubiquidade (capacidade de estar em todos os lugares ao mesmo tempo; qualidade do que está em toda a parte, do que é ubíquo), como é visto nas Figura 2.2 e Figura 2.3.

Figura 2.2- Ubiquidade -, tempo, espaço e contexto (Associação Brasileira de Mídia Out of Home, 2015)

Figura 2.3 - Ubiquidade por perfil (Associação Brasileira de Mídia Out of Home, 2015)

Um anunciante que passe os seus anúncios numa empresa de SD e que possua vários pontos instalados nos mais diversos tipos de locais (metro, bares, shoppings) terá a sua marca aparecendo praticamente todo o tempo, em algum lugar, atingindo os mais diferentes perfis de público alvo.

2.5.2.1 VANTAGENS

Como vantagens da SD, a INBOX MÍDIA (Inbox - Soluções Digitais, 2015), no seu sitio web afirma:

- 85% das decisões de compra são feitas no ponto-de-venda (PDV);
- Com a Sinalética Digital no PDV, o volume de vendas aumenta até 15%;
- Anúncio de fidelização de marca (*branding*) amplia até 10% das vendas;
- Preços promocionais anunciados incrementam até 22% nas vendas;
- Segmentação com interesse na região e padrões de compra do cliente;
- Exibição de media com qualidade superior à média convencional;
- Maior dinamismo, dado que o controle da ferramenta é 100% via internet;
- Todo tempo é horário nobre (dia inteiro) – Foco direto no público-alvo;
- Economia em suportes publicitárias e em distribuição.

2.5.2.2 DESVANTAGENS

Uma desvantagem é que a SD é um tipo de media predominantemente interna necessitando que o potencial espectador entre no estabelecimento para ter acesso à mesma.

Uma outra desvantagem que está a ser amplamente discutida nos países que já possuem essa tecnologia é o facto da SD ser muito invasiva, ocupando espaço e atenções de pessoas nem sempre interessadas naquele conteúdo.

A rápida evolução da tecnologia utilizada, torna oneroso a atualização dos meios utilizados na SD. Por fim, pode-se citar o fato de ser complexo instalar ecrãs em ambientes públicos, de grande circulação e em alguns particulares, por depender de autorizações, legislações e da aprovação dos respetivos donos.

2.5.3 TECNOLOGIAS

Os equipamentos de sinalética digital podem ser painéis rolantes, painéis LCD ou Plasmas, telas projetadas, painéis eletrónicos ou outros tipos emergentes como OLED que são controlados eletronicamente por um computador ou outro dispositivo equivalente, permitindo que os utilizadores alterem os seus conteúdos, tipicamente através da Internet ou rede local.

A calendarização e exibição dos conteúdos pode ser efetuada por vários métodos, desde os simples media *players* que reproduzem *loops* de imagens e filmes, a redes complexas e distribuídas que permitem o controlo de milhares de dispositivos através de um único ponto de acesso. O primeiro tipo é indicado para pequenos grupos de equipamentos, onde é possível a utilização de dispositivos de armazenamento móveis, enquanto o

segundo tipo permite que um único operador da rede de sinalética digital controle ou delegue conteúdos para equipamentos dispersos e com tecnologias de difusão variadas.

A recente introdução de aplicações gratuitas de sinalética digital expande ainda mais a potencial cobertura da tecnologia, passando a ser possível a pequenas empresas e a entidades sem fins lucrativos que de outro modo considerariam a tecnologia demasiado cara.

A difusão generalizada dos equipamentos móveis com capacidades multimédia, tem aberto novos caminhos para o desenvolvimento da área, ao introduzirem a capacidade de eles próprios se tornarem em equipamentos de sinalética digital ou de poderem interagir com os equipamentos existentes. Com recurso aos serviços de mensagens, leitura de códigos de barras, fotografia, filmes, o utilizador pode interagir com a sinalética digital ou clássica, permitindo-lhe obter instantaneamente informação.

Embora o termo SD esteja bastante difundido na maior parte da Europa e América do Norte, a mesma tecnologia é referida como *Narrowcasting* (literalmente difusão limitada, em oposição ao termo *broadcast* que traduz para difusão alargada); outros termos como *screen media*, *place-based media*, *digital merchandising* ou mesmo "*captive audience networks*", ou "*CANs*" são habitualmente utilizados para a sinalética digital. O grande número de designações levou a *POPAI* (*Point of Purchase Advertising International*) a formar um grupo para a criação de *standards* para terminologia e modelos de negócio da sinalética digital.

A SD já há algum tempo está presente em veículos equipados com os equipamentos necessários, como táxis e autocarros. Nesses ambientes, os sistemas costumam disponibilizar informações relevantes sobre os locais onde os veículos circulam. Em táxis, os ecrãs são instalados nos *headsets*, permitindo interatividade.

Uma novidade interessante é a possibilidade de entrega de serviços e conteúdos baseados em localização. O Brasil já possui esta tecnologia, que permite a localização do veículo por meio de um GPS ligado aos equipamentos de SD.

2.6 APLICAÇÕES DE REFERÊNCIA

Aquando da pesquisa, sobre aplicações existentes no mercado, para poder ter sugestões para o projeto, encontrou-se algumas aplicações existentes que se aproximam do tema, mas nenhuma satisfazia por completo as necessidades do projeto.

Das existentes salientamos, Xibo (Spring Signate, Ltd, 2015) e Intel Retail Client Manager (Intel, 2015).

2.6.1 XIBO

Xibo é uma solução de sinalética digital completa, composta por um sistema baseado em gestão de conteúdo web (CMS) e com reprodutores de sinalética em Windows ou Android.

- O CMS Xibo é uma aplicação web PHP/MySQL, que corre em Windows, Mac ou Linux. É a interface de administração central para a rede de ecrãs. Tem como requisitos:
- Apache, NGINX ou IIS
- PHP 5.3.3 ou superior
- MySQL com suporte PHP PDO

No Xibo os reprodutores de sinalética digital são referidos como Displays. Um Display é uma TV ou projetor que será usada para mostrar conteúdo. Xibo será executado num PC por trás do Display, que comunicará com o CMS Xibo, e exibirá o conteúdo dentro da programação estabelecida, conforme Figura 2.4.

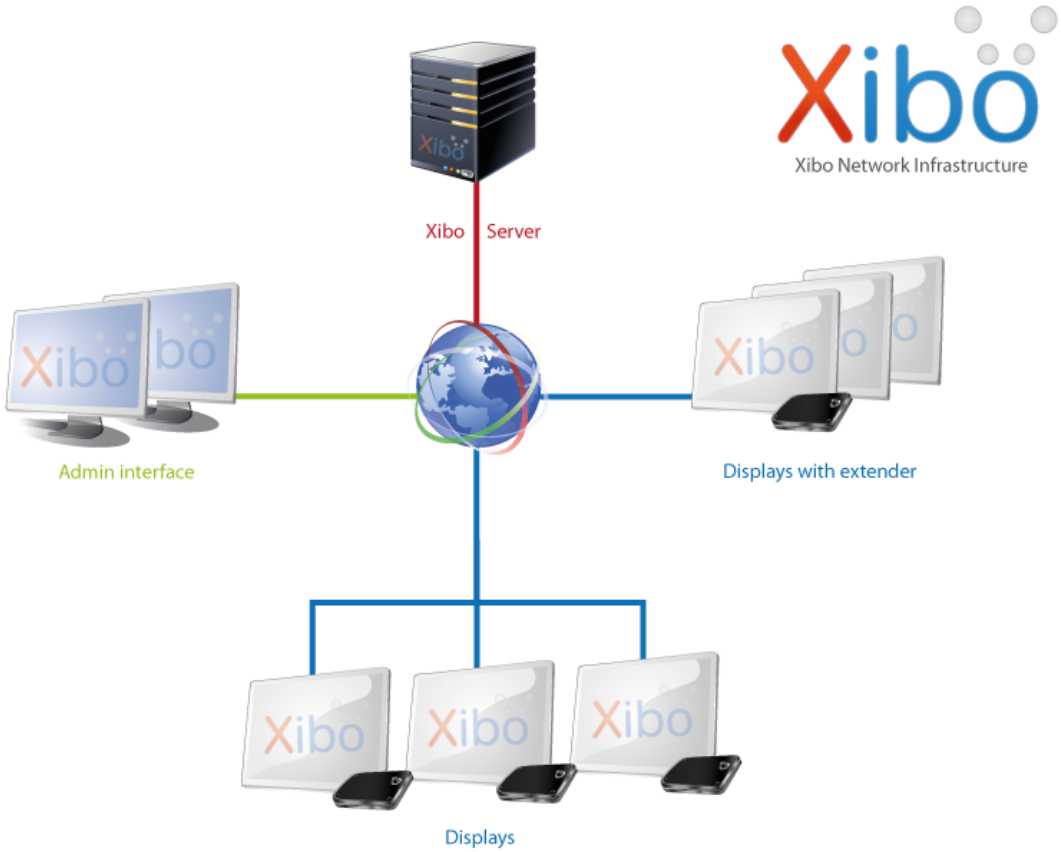

Figura 2.4 – Sistema Xibo (visão geral) (Spring Signate, Ltd, 2015)

Características importantes:

- Suporta vários tipos de media;

- Vários *Layouts*
- Calendarização
- Gestão dos *displays*
- Permissões

2.6.2

2.6.3 INTEL RETAIL CLIENT MANAGER

"Intel® Retail Client Manager (Intel® RCM) is the intelligent software solution that enables you to deliver digital content to every screen with greater impact, when and where it matters most. Reach your ideal audience segments with the right message at the right time. Improve the effectiveness of ad campaigns through audience analytics. Remotely manage your marketing campaign in near real-time. And launch customized content on the fly, to quickly and easily deliver a richer brand experience." (Intel, 2015).

INTEL RCM é um CMS para SD a partir do qual publicamos conteúdo específico sobre os nossos reprodutores e ecrãs que são implantados em diferentes localizações geográficas, desde que estejam ligados à Internet (pelo menos o tempo suficiente para poder renovar o conteúdo das campanhas). Mas, além de se comportar como simples sistema de SD tradicional, a plataforma também nos permite publicar conteúdo á medida, dependendo do perfil do público que está na frente de nossos ecrãs.

Características importantes:

- Plataforma intuitiva e fácil de usar
- Ferramentas inteligentes para controlar a distribuição de conteúdo quase em tempo real.
- Controlar remotamente pontos de contato digitais de retalho, a partir de qualquer lugar com capacidades de gestão, com base num navegador melhorado.
- Envolver segmentos de público ótimos, usando *Audience Analytics*
- Captura em tempo real de oportunidades de venda.
- Personalizar conteúdo "*on the fly*".
- Implementado globalmente com um leque de línguas (incluindo Inglês, Espanhol, Japonês, Russo, Chinês Simplificado, Alemão e Português).
- Receber relatórios de erros em tempo real.

- Controlar individualmente ponto digitais no consumidor, canais dedicados, ou redes de anúncios com características de gestão avançada tais como "*Power Off, Restart, Show Screen, and Direct Control".*
- Tirar partido das capacidades criticas de Wake Up e Reset to BIOS para a Intel® Active Management Technology (Intel® AMT).
- Criar conteúdo usando qualquer combinação de vídeo (desde a definição padrão até à alta-definição), imagens e som.
- Distribuir conteúdo de marketing com uma relação custo mais eficaz, sem os custos altos da distribuição e impressão tradicionais.

2.6.4 *COMPARATIVO*

Podemos assim, fazer um comparativo entre as duas aplicações Foram considerados importantes para comparação os seguintes aspetos, tecnologia usada, a facilidade de instalação, a capacidade de *fullscreen* automático, os formatos que pode reproduzir, se é código *open-source* ou não, ou se é capaz de reproduzir playlists.

Tabela 2.1 - Comparativo plataformas

	Xibo	Intel RCM
Tecnologia	.NET	proprietária
Facilidade de Instalação/configuração	média	alta
Full screen automático	sim	sim
Multi formato	sim	sim
Open source	sim	não
Playlist	sim	sim

Conforme se pode ver na Tabela 2.1 a plataforma Xibo é mais fácil de instalar e configurar e é *open-source*.

3 ANÁLISE DE REQUISITOS DA PLATAFORMA

3.1 INTRODUÇÃO

No que respeita aos requisitos colocados para a criação de uma plataforma centralizada que permita a um conjunto de lojas aderentes, publicar e publicitar anúncios vídeo que serão distribuídos e apresentados pelas restantes lojas em tempo real, devemos tem em consideração os seguintes: o acesso à plataforma ser feito através de autenticação; a possibilidade de acesso através de uma hiperligação com autenticação embutida, utilizando para isso um *token;* o número de anúncios permitidos por utilizador deverá ser definido; a duração dos anúncios; a não inclusão de anúncios de negócio concorrente na no ecrã do anunciante concorrente a esse mesmo negócio; a existência de uma solução alternativa para caso o acesso à plataforma ficar inacessível durante a exibição das listas de vídeos, e assim não ser exibida uma mensagem de erro.

Esta secção trata de responder aos problemas colocados, aos tipos de interações entre plataforma e utilizadores e às funcionalidades e conteúdos a incluir.

3.2 REQUISITOS

Num processo de engenharia de requisitos geralmente reconhecem-se dois grandes tipos de requisitos (Leite & Freeman, 1991) (Sommerville & Sawyer, 1997) (Sommerville & Kotonya, 1998), os funcionais e não funcionais, os quais passaremos a descrever com mais detalhe nas seções seguintes.

3.2.1 REQUISITOS FUNCIONAIS

É um requisito de sistema de software que especifica uma função que o sistema ou componente deve ser capaz de realizar. Estes são requisitos de software que definem o comportamento do sistema, ou seja, o processo ou transformação que componentes de software ou hardware efetuam sobre as entradas para gerar as saídas. Esses requisitos capturam as funcionalidades sob o ponto de vista do utilizador.

3.2.1.1 UTILIZADORES

Em termos de requisitos funcionais, os utilizadores são os principais atores, e em particular neste projeto, não pode existir nomes de utilizadores repetidos. Nesta análise, presume-se que, para um utilizador interagir na plataforma, necessita de estar autenticado. Por sua vez, todos os utilizadores terão associado um tipo de perfil e um tipo de negócio, e têm de ter um nome, nome de utilizador (*username*) e palavra-passe (*password*). O utilizador é criado pelo administrador ou pelo perfil que corresponde à criação de utilizadores. O utilizador necessita de ser ativado pelo administrador para poder fazer login. O utilizador tem de disponibilizar um caminho "*path*", que indica a

localização do ficheiro reproduzido em caso de erro (ou perda de ligação) na sessão cliente.

3.2.1.2 NEGÓCIOS

Este nível de requisitos diz respeito ao tipo de negócio associado ao utilizador, exemplo joalharia, restauração, etc. Serve para controlar a passagem dos anúncios nos negócios concorrentes. São definidos pelo administrador.

3.2.1.3 PERFIS

Corresponde ao conjunto de funções que o utilizador pode desempenhar na plataforma, são definidos pelo administrador. Deve existir a possibilidade de escolher quais as funções *Create*, *Read*, *Update* e *Delete* (CRUD) serão atribuídas aos perfis.

3.2.1.4 ANÚNCIOS

Neste tipo de requisito, os anúncios pertencem a um utilizador, não podendo existir anúncios com nomes repetidos, todos os anúncios têm uma data de início e de fim. Por outro lado, os anúncios podem ser desativados por vontade do anunciante independentemente da data de fim.

O carregamento de um anúncio enviará uma mensagem de correio eletrónico ao administrador, para conhecimento e posterior aprovação. Depois de carregados serão aprovados ou não aprovados pelo administrador, que dará *feedback,* utilizando um campo disponível para o efeito.

Uma outra particularidade é que o administrador terá acesso a uma pré-visualização em formato reduzido do anúncio na área de aprovação do mesmo, para não recorrer à página de visualização.

3.2.1.5 VISUALIZAÇÃO DOS ANÚNCIOS

Os anúncios serão visualizados num browser compatível com a tecnologia html5 e deve existir a hipótese de o utilizador criar um link para posterior acesso à plataforma sem preciso fazer *login*. Quando não existe ligação à Internet, e como os anúncios são exibidos em *streaming*[4], deve existir uma alternativa, para o ecrã não ficar a mostrar mensagens de erro, posteriormente quando a ligação voltar, a lista de vídeos é recarregada e os anúncios voltarão a ser exibidos normalmente em *streaming*. Deverão ser visualizados em *fullscreen*.

[4] https://pt.wikipedia.org/wiki/Streaming

3.2.3 Requisitos Não Funcionais

Os requisitos não funcionais são aqueles que não dizem especificamente respeito as funcionalidades do sistema. Incluem, entre outros, requisitos de fiabilidade, segurança, adaptabilidade, portabilidade e desempenho.

3.2.3.1 Escalabilidade

A aplicação terá de ser capaz de lidar com o aumento do número de conteúdos da base de dados bem como um esperado aumento do número de utilizadores.

3.2.3.2 Desempenho

Pretende-se que a plataforma tenha uma performance considerada boa. Esta performance mede-se pela capacidade de resposta do serviço, aos pedidos dos utilizadores. Uma vez que esta é sobretudo uma aplicação em que o uso de conteúdo multimédia é fundamental, irá sempre existir um processamento e informação enviada para o utilizador com o qual teremos sempre de contar.

3.2.3.3 Segurança

Todos os acessos à plataforma serão feitos após autenticação por nome de utilizador e palavra-passe guardada de forma encriptada no servidor. A inscrição dos utilizadores é feita pelo perfil administrador do sistema e todas as palavras passe são geradas pelo perfil administrador do sistema.

3.2.3.4 Usabilidade

Em termos de usabilidade pretende-se que a plataforma seja de utilização fácil, simples e intuitiva. A aplicação terá de ser acessível em qualquer tipo de sistema, design responsivo.

3.3 Modelo de Domínio

O Modelo de Domínio identifica os conceitos relacionados com requisitos do sistema e analisa o problema sob a perspetiva conceptual.

Assim, o Modelo de Domínio deve ser independente da solução física que virá a ser adotada e deve conter apenas elementos referentes ao domínio do problema em questão, ficando para a fase de projeto os elementos da solução, isto é, todos os conceitos que se referem à implementação propriamente dita, tais como: interfaces, formas de armazenamento (base de dados), segurança de acesso, comunicação, etc.

Para a solução do projeto Gestão Centralizada de Anúncios foi esboçado inicialmente o seguinte modelo:

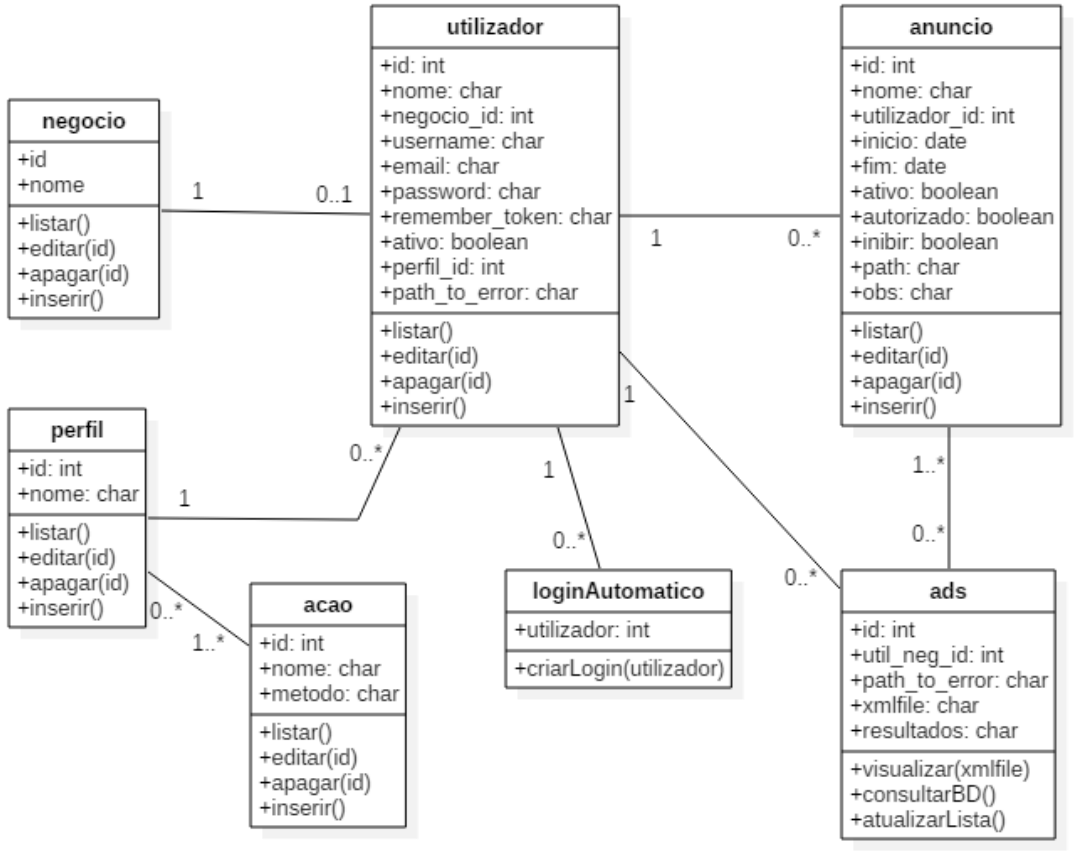

Figura 3.1 – Modelo de Domínio

Na Figura 3.1, estão descritas as classes do sistema. As classes são utilizador, anuncio, perfil, negocio, acao, ads e loginAutomatico. Nas classes utilizador, anuncio, negocio, perfil e acao, estão listados métodos que permitem listar, editar, apagar e inserir registos na base de dados, tal como se pretende, a classe loginAutomatico apenas tem um método que permite criar o login automático, por fim a classe ads tem três métodos, um que permite fazer a consulta à base de dados, outro que gera a lista xml e outro que atualiza a lista caso seja necessário. Os atributos das várias classes são os mínimos indispensáveis ao funcionamento do sistema.

3.5 CASOS DE USO

Um modelo de casos de uso é a descrição do conjunto de sequências de ações, incluindo variantes, que um sistema realiza e que deu um resultado observável de valor para um ator (Booch, et al., 1999).

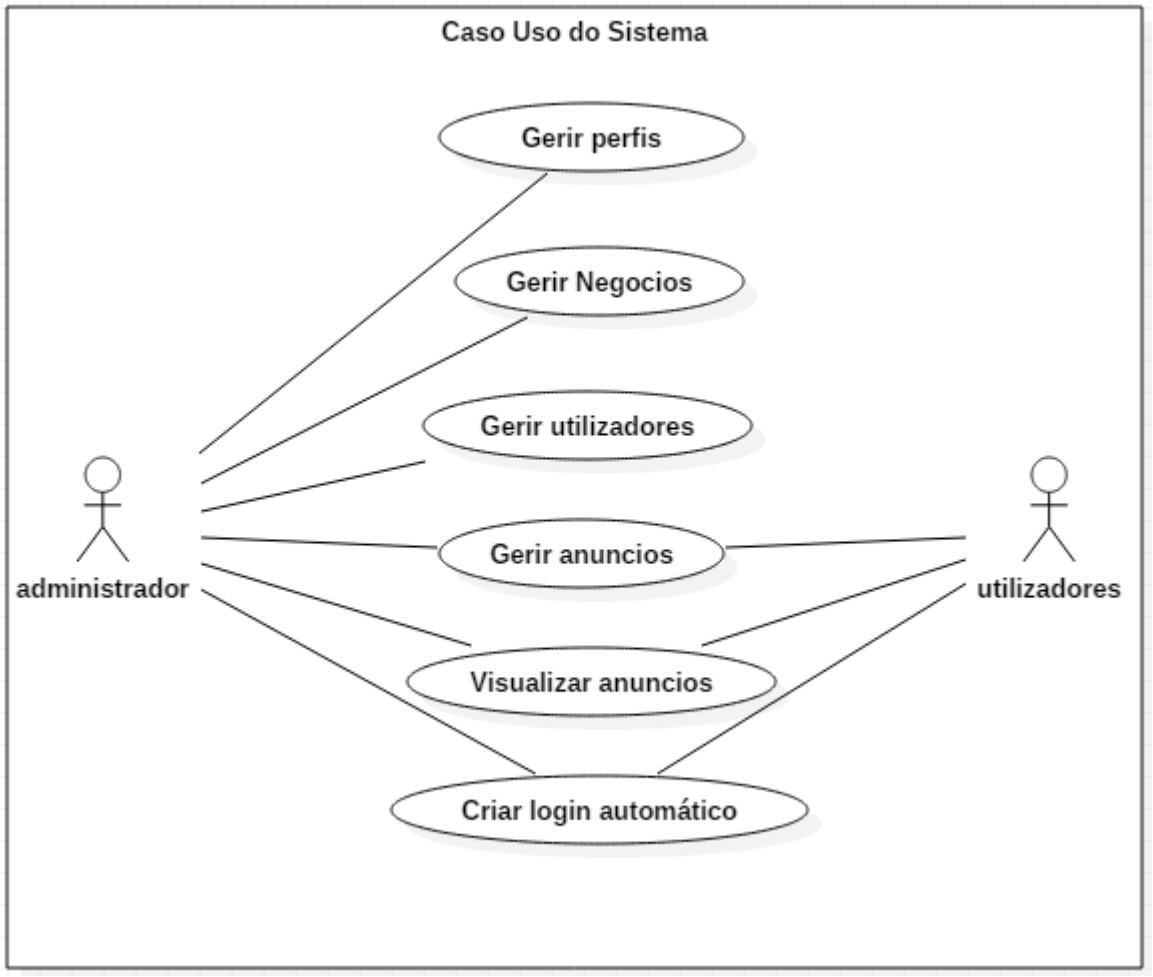

Figura 3.2 - Caso de uso do sistema

O diagrama da Figura 3.2, descreve as funcionalidades do sistema proposto. É ao administrador do sistema, que cabe a tarefa de gerir os perfis dos utilizadores, podendo atribuir a esses perfis, funções que podem realizar no sistema, gere também o tipo de negócios a incluir na plataforma, negócios esses que servem para agrupar o tipo de atividade dos utilizadores. O administrador também faz a gestão dos utilizadores, criando, apagando ou desativando utilizadores. É o responsável pela aprovação dos anúncios, podendo para isso visualizar todos os anúncios inseridos, para poder aferir de alguma inconformidade com os requisitos do negócio.

Ao utilizador do sistema para além das operações CRUD sobre os seus anúncios, pode ainda visualizar os anúncios que são gerados para cada um sob a forma de uma lista e criar *links* para *autologin*.

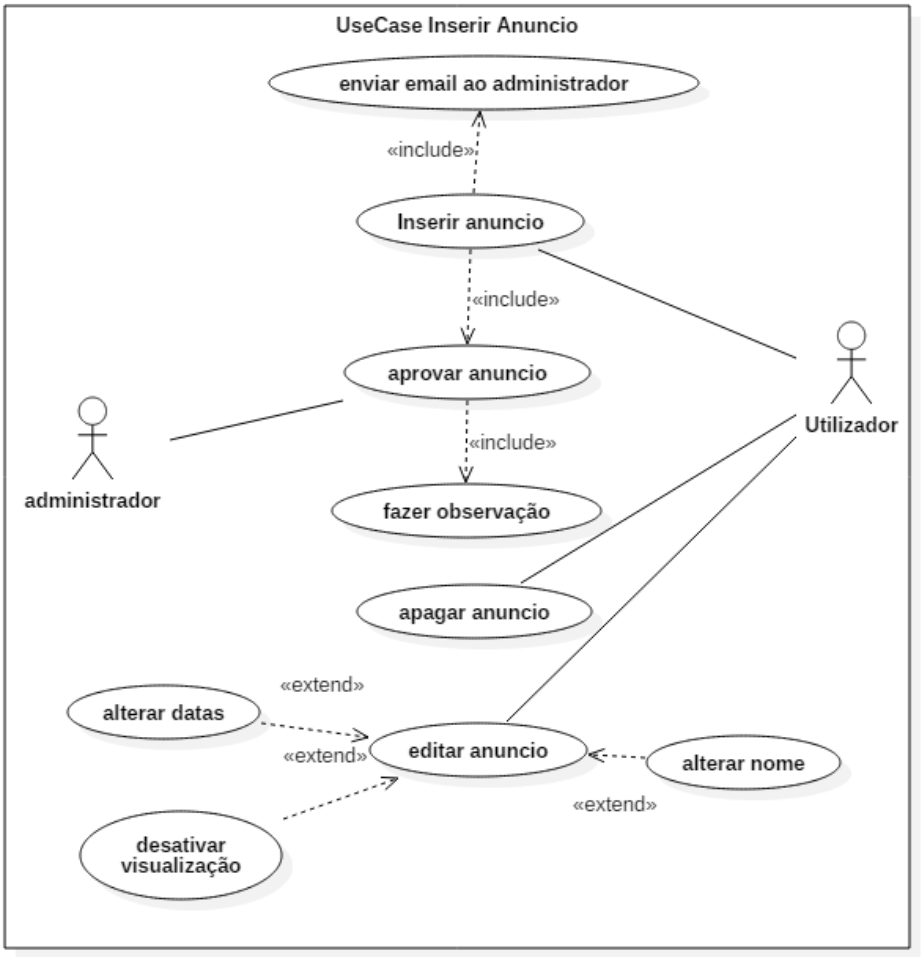

Figura 3.3 - Caso de Uso - Inserir Anúncio

O diagrama de caso de uso da Figura 3.3 descreve a inserção e ou edição de um anúncio por um utilizador. O utilizador insere o anúncio, após o qual o administrador receberá um email, notificando-o dessa mesma inserção. O administrador após visualização do mesmo para verificar se está tudo em conformidade com os requisitos do negócio, deve aprovar ou não, fazendo também uma observação em campo próprio para informação ao utilizador. O utilizador posteriormente poderá apagar o anúncio ou editá-lo no que diz respeito às datas de visualização ou proceder à sua desativação.

3.6 NOTAS FINAIS

Neste capitulo foram apresentados os requisitos necessários à criação da PLATAFORMA DE GESTÃO DE ANÚNCIOS DE VIDEO - ADS4SHOPS, foram também projetados os modelos de suporte ao sistema, que serão desenvolvidos e implementados no próximo capitulo.

4 Desenvolvimento do Projeto

4.1 Introdução

O projeto foi desenvolvido por fases, tendo na primeira fase ocorrido a análise do futuro sistema e levantamento de requisitos, e produzida a interface de gestão dos anúncios. Na segunda fase concebida a interface de visualização dos anúncios, e na terceira a plataforma foi melhorada pois foram pedidas novas funcionalidades. Foi concretizada a aplicação e haverá lugar à implementação do projeto numa fase experimental.

4.2 Fase I - Análise e Levantamento de Requisitos

Após a análise do modelo de domínio e dos requisitos, partiu-se para a criação de um sistema acessível através de um *browser* (multiplataforma), onde as lojas/utilizadores aderentes depois de feita a autenticação poderão realizar operações CRUD sobre os seus anúncios.

Após a fase de análise, passou-se à modelação do problema.

4.2.1 Modelação de Dados

Um modelo de dados descreve os dados que suportam os processos de um sistema de informação organizacional.

4.2.1.1 Modelo Conceptual

O modelo conceptual é um diagrama em blocos que demonstra todas as relações entre as entidades.

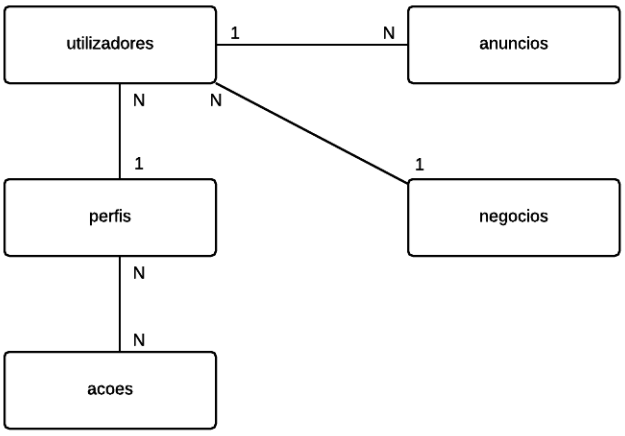

Figura 4.1 - Modelo conceptual de base de dados

No modelo representado na Figura 4.1, pode observar-se que um utilizador pode ter vários anúncios e que vários anúncios podem pertencer a um utilizador, que a um utilizador pertence um e só um negocio e também pertence um e só um perfil, podendo

negócios e perfis iguais pertencer a vários utilizadores diferentes. A um perfil corresponde um conjunto de ações e um conjunto de ações pode estar presente num ou em vários perfis.

4.2.1.2 MODELO RELACIONAL

O modelo relacional representa um modelo de dados utilizado em Sistemas Gestão de Bases de Dados Relacionais (SGBDRS). Baseiam-se no princípio de que todos os dados estão guardados em tabelas (ou, matematicamente falando, relações). Toda a sua definição é teórica e baseada na lógica de predicados e na teoria dos conjuntos.

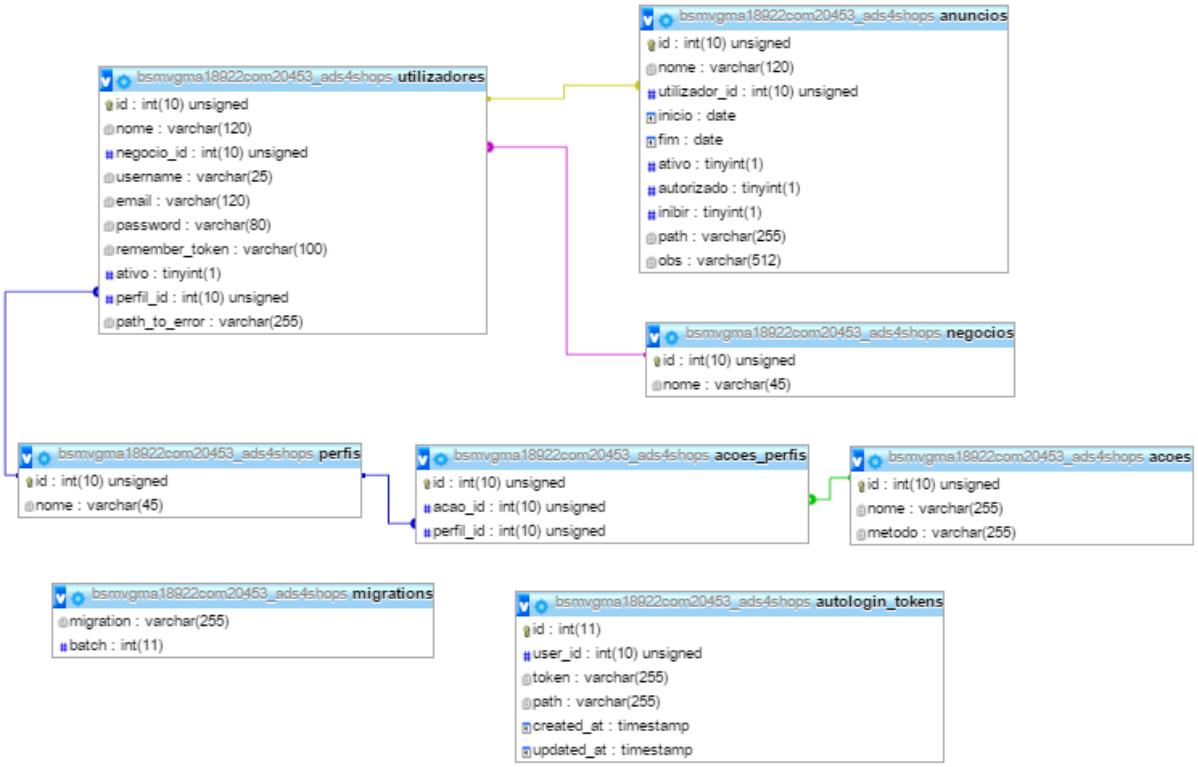

Figura 4.2 - Modelo relacional da base de dados

No modelo da Figura 4.2 os nomes das tabelas tiveram em conta a filosofia do *Laravel*, assim como o campo id. O tamanho do campo *password* também teve de ser do tipo *varchar* com um tamanho 80 para poder usar o *hash* do *Laravel*, nos demais campos, procuramos seguir a nomenclatura tradicional.

4.2.2 *LARAVEL*[5]

A instalação do *Laravel*, necessita de alguns pré-requisitos, nomeadamente o PHP, as extensões PHP *OpenSSL*, PHP PDO, PHP *Mbstring* e PHP *Tokenizer*.

O *Laravel* utiliza o *Composer*[6] para gerir as suas dependências pelo que deve estar instalado antes de procedermos à instalação do *Laravel*.

Para fazermos a instalação do *Laravel* recorremos à pagina do seu sítio web[7], que documenta muito bem a sua instalação.

4.2.3 *MVC - LARAVEL*

O *Laravel* assenta na arquitetura *MVC* (***M**odel* + ***V**iew* + ***C**ontroller*). Enquanto as *Views (vistas)* e os *Controllers (controladores)* têm presença na camada de apresentação, os componentes do tipo *Model (modelos)* implementam a camada de acesso a dados, refletindo o Modelo de Dados e, consequentemente, a BD.

O modelo *MVC* isola a lógica da aplicação da camada de interface do utilizador e suporta separação de conceitos. O Controlador recebe todas as solicitações para a aplicação e, em seguida, trabalha com o Modelo para preparar todos os dados necessários para a Vista. A Vista, depois, utiliza os dados elaborados pelo Controlador para gerar uma resposta apresentável final.

[5] https://laravel.com/

[6] https://getcomposer.org/doc/00-intro.md

[7] https://laravel.com/docs/4.2

Uma aplicação típica em *Laravel* consiste nos componentes *MVC* mencionados na Figura 4.3.

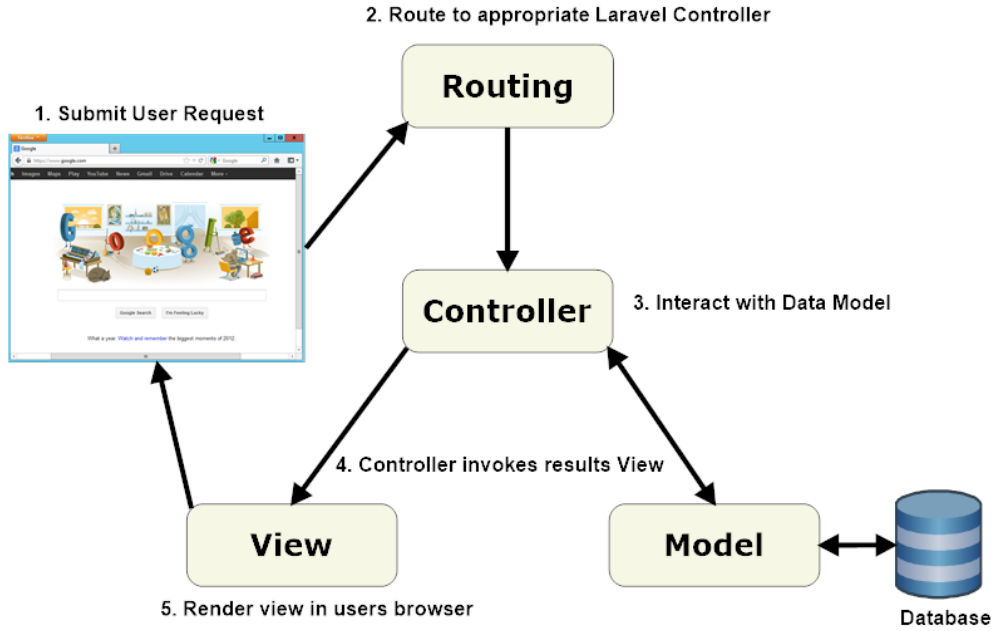

Figura 4.3 – Model View Controller em Laravel (Surguy, s.d.)

A Figura 4.3 – Model View Controller em LaravelFigura 4.3 descreve o modelo MVC anteriormente referido, mas em *Laravel* tem a particularidade de utilizar entre o pedido do utilizador, uma rota que encaminha o pedido para o respetivo Controlador.

4.2.3.1.1 LARAVEL ROUTING & FILTERS

Laravel Routes e *Laravel Route Filters* são termos do *Laravel* para lidar com endereçamento dentro do domínio gerido pelo *Laravel*. Sendo que se trata de uma plataforma *web*, todo o seu funcionamento responde a pedidos na forma de endereços URL geridos por estes elementos.

4.2.3.1.2 ROUTING

A programação dos endereços é realizada num ficheiro de nome “routes.php” que é colocado na pasta “app” na raiz da aplicação. Na Figura 4.4 podemos ver

Como por exemplo na Figura 4.4, quando alguém faz um pedido á raiz da aplicação (/), é

```
Route::get('/', function()
{
  if(Auth::check()){
    return View::make('utilizadores.welcome');
  }else {
    return View::make('utilizadores.login');
  }
});
Route::get('login', 'UtilizadoresController@login');
Route::post('login', 'UtilizadoresController@validate');
```

Figura 4.4 - Exemplo route

corrido a função, onde se testa se o utilizador está autenticado e nesse caso encaminha para a vista *welcome*, caso contrário para a vista *login*.

4.2.3.2 MODELS

Os modelos Laravel são classes que refletem e manipulam o estado da BD. Existe um modelo para cada tabela, escrito conforme a mesma, contendo as suas colunas, relações, índices e respetivas propriedades. Cada modelo é implementado na forma de classe PHP que deriva da classe *Eloquent* (O "*ORM Eloquent*" incluído no *Laravel* fornece uma implementação de *ActiveRecord* fácil para trabalhar com a BD) e mapeia diretamente a correspondente entidade, do Modelo de Dados, e tabela na BD. A título de exemplo, para a entidade "Utilizador", do Modelo de Dados, existe uma tabela "utilizadores" na BD e um modelo "Utilizador" na aplicação Laravel. A classe Eloquent fornece rotinas para implementação dos membros (atributos, métodos, construtores, ...) necessários para consulta e alteração do seu estado na BD (inserção, alteração e remoção) tanto em atributos, registos como relações. A parte que compete ao produtor dos modelos é a

```
class Utilizador extends BaseModel implements UserInterface,
RemindableInterface
{
  protected $table = 'utilizadores';
  public $timestamps = false;
  protected $fillable = array('nome', 'negocio_id', 'username', 'password',
'email', 'ativo', 'perfil_id');
  public static $rules = array(
   'nome' => 'required|min:3|max:120',
   'negocio_id' => 'required|integer',
   'username' => 'required|min:3|max:25|unique:utilizadores,username',
   'password' => 'required|min:3|max:80',
   'email' => 'required | email |min:3 |max:120 |unique : utilizadores,email',
   'ativo' => 'required|in:0,1',
   'perfil_id' => 'required|integer',
  );
}
```

Figura 4.5 – Exemplo classe de modelo *Laravel*

implementação de classes com estes membros, ver Figura 4.5.

4.2.3.3 *VIEWS*

Uma vista é uma área visível. A produção das Vistas é feita em *HTML*, *CSS* e *PHP* essencialmente produzidos pelo *Laravel* (*blades*[8]) e/ou pelo *framework Bootstrap*.

Refletem os contextos da plataforma, devem mudar drasticamente entre contextos e manterem-se coesas dentro de cada contexto, para que a participação dos seus utilizadores seja intuitiva.

4.2.3.3.1 *VISTA PÚBLICA*

A Figura 4.6 representa a vista que apresenta área de autenticação, e que é solicitada quando se aponta para o domínio http://www.ads4shops.com.

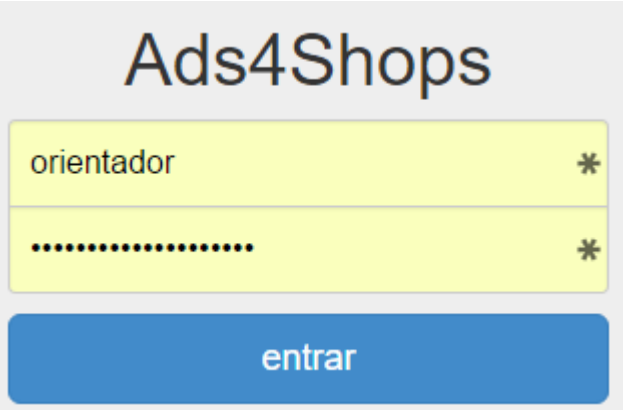

Figura 4.6 - Vista pública

4.2.3.3.2 *VISTA UTILIZADORES AUTENTICADOS*

A Figura 4.7 representa a vista que aparece depois do utilizador se autenticar com sucesso.

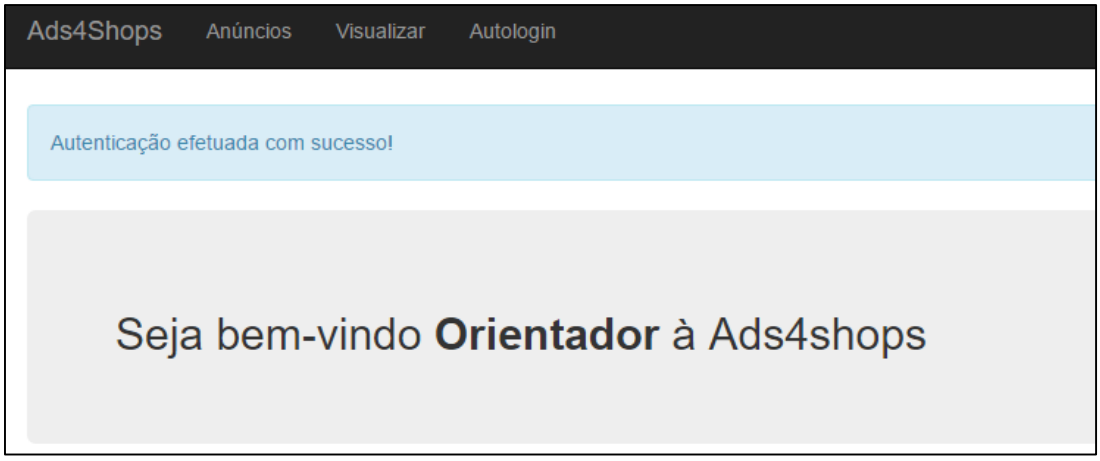

Figura 4.7 - Vista após autenticação - boas vindas

[8] Blade is a simple, yet powerful templating engine provided with Laravel - https://laravel.com/docs/4.2/templates#blade-templating

Os menus variam conforme o perfil do utilizador. Para isso usou-se uma macro:

Em macros.php (Figura 4.8):

```
HTML::macro('mountMenu', function() {
        $menu = null;
        $cache = Cache::get('actions' . Auth::user()->id
        if(is_null($cache{ Redirect::to('logout');}
        $index = preg_grep('/index|\*/', $cache);
        $links = array(
                'utilizador' => 'Utilizadores',
                'perfil' => 'Perfis',
                'negocio' => 'Negócios',
                'anuncio' => 'Anúncios',
                'ads' => 'Visualizar',
                'automaticlogin' => 'Autologin'
        );
        $parametros = array('id' => 'ver');
   foreach($links as $key => $value) {
        foreach ($index as $i) {
            if ($key == current(explode('.', $i)) || $i == '*') {
                if ($key == 'ads') {
                    $menu.= '<li>'.link_to($key,$value,$parametros)
'</li>';
                } else {$menu .= '<li>' . link_to($key, $value) .
'</li>';}
            }
        }
    }
return $menu;
});
```

Figura 4.8 - Macro para menu

Em views/partials/_navigation.blade.php:

```
<div class="collapse navbar-collapse">
        <ul class="nav navbar-nav">
        {{ HTML::mountMenu() }}
```

Assim temos menus distintos por exemplo para Administrador (Figura 4.9) e para o Anunciante (Figura 4.10).

Administrador:

Figura 4.9 - Menu administrador

Anunciante:

Figura 4.10 - Menu anunciante

4.2.3.4 CONTROLLERS

Controladores são elementos que reagem aos eventos de vistas e invocam comandos sobre vistas e modelos (i.e, como elementos afetados pelos comandos).

Neste projeto conforme se pode ver na Tabela 4.1 procurou-se manter uma relação direta entre controladores e vistas ou modelos.

Tabela 4.1 - Rotas *Laravel*

View	Controller	Model
anuncios/index.blade.php	AnunciosController.php	Anuncio.php
anuncios/create.blade.php		
anuncios/edit.blade.php		
anuncios/edit_admin.blade.php		

4.2.4 MySQL e Laravel

A base de dados (Figura 4.11) foi criada em MySQL, criaram-se as tabelas, mencionadas no modelo relacional, utilizando uma ferramenta do *Laravel,* as *migrations*.

Figura 4.11 - phpMyAdmin base de dados Ads4Shops

4.2.4.1 Migrations

Migrations são um tipo de controlo de versões para a base de dados. As *migrations* estão tipicamente ligadas com o *Schema Builder* para facilmente gerir o esquema da aplicação.

Para criar uma *migration*, temos de usar o comando *migrate:make* na linha de comandos *Artisan* como na Figura 4.12.

```
php artisan migrate:make create_users_table
```

Figura 4.12 - criar uma migration com o comando artisan

A *migration* será colocada na pasta *app/database/migrations*, e contêm um *timestamp* que permite ao *framework* determinar a ordem das *migrations*, conforme pode ser visto na Figura 4.13.

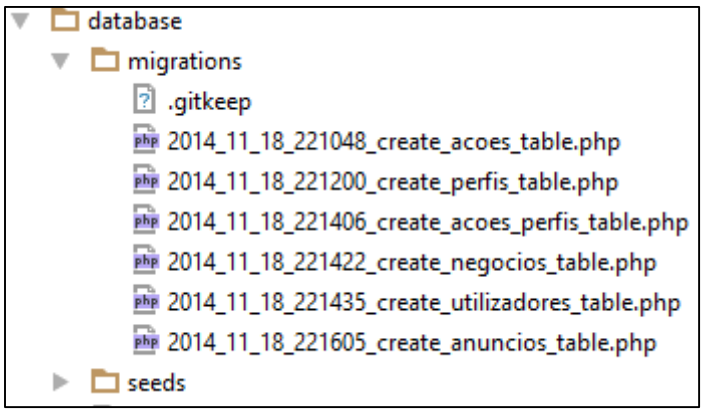

Figura 4.13 - *migrations* com *timestamp*

4.2.4.2 SCHEMA BUILDER

A classe *Schema* do *Laravel* fornece uma maneira agnóstica de manipular tabelas na base de dados. Funciona bem em todas as bases de dados suportadas pelo *Laravel*, e tem uma API[9] unificada através de todos esses sistemas.

Na Figura 4.14, por exemplo para criarmos a tabela utilizadores utilizamos o método *Schema::create.*

```
public function up()
{
    Schema::create('utilizadores', function(Blueprint $table)
    {
        $table->increments('id');
        $table->string('nome', 120);
        $table->integer('negocio_id')->unsigned();
        $table->string('username', 25);
        $table->string('email',120);
        $table->string('password', 80);
        $table->boolean('ativo')->default(0);
        $table->integer('perfil_id')->unsigned();

        $table->foreign('negocio_id')->references('id')->on('negocios')->on_delete('restrict');
        $table->foreign('perfil_id')->references('id')->on('perfis')->on_delete('restrict');
    });
}
```

Figura 4.14 - Método Schema::create

No qual definimos as colunas da tabela, os tipos de dados e outras configurações, até mesmo chaves estrangeiras. O tipo *increments* automaticamente cria a chave primária.

4.2.4.3 DATABASE SEEDING

Foram preenchidas algumas tabelas essenciais para o arranque do sistema, utilizando a ferramenta do *Laravel* as *seeds*.

[9] https://en.wikipedia.org/wiki/Application_programming_interface

O *Laravel* também incluí uma maneira simples de preencher a base de dados utilizando a classe *seed*. Todas as classes *seed* são guardadas na pasta *app/database/seeds*. As classes *seed* podem ter o nome que nós quisermos, mas devemos seguir um padrão. Por defeito o *Laravel* já fornece uma classe *DatabaseSeeder*, desta classe podemos chamar o método *run* para fazer correr as outras classes *seed* conforme Figura 4.15, permitindo deste modo controlar a ordem.

```
class DatabaseSeeder extends Seeder {

    public function run()
    {
        Eloquent::unguard();

        $this->call('NegocioTableSeeder');
        $this->call('AcaoTableSeeder');
        $this->call('PerfilTableSeeder');
        $this->call('UtilizadorTableSeeder');
    }
}
```

Figura 4.15 - Classe DatabaseSeeder

Na Figura 4.16 um exemplo de uma classe *seed* para a tabela perfis.

```
class PerfilTableSeeder extends Seeder {

    public function run()
    {
        DB::table('perfis')->delete();

        $perfil = Perfil::create(array(
            'nome' => 'Administrador',
        ));

        $perfil->acoes()->sync(array(1));
    }

}
```

Figura 4.16 - Classe Seeder personalizada

4.2.4.4 AUTENTICAÇÃO

O sistema de autenticação, tal como o *Laravel* anuncia, é feito de uma maneira muito simples, dado que está quase tudo configurado "out of the box". O ficheiro de configuração da autenticação está localizado em app/config/auth.php, o qual contem muitas opções documentadas para afinar o comportamento das instalações de autenticação.

Os utilizadores são criados pelo administrador, e as *passwords* são guardadas encriptadas na base de dados. O *Laravel* tem uma classe *Hash*[10] que o permite encriptar dados de forma fácil, utiliza o método *Bcrypt*[11] para fazer o *hash* das palavras passe, conforme se pode ver na Figura 4.17., ficando depois na base de dados as palavras passe guardadas de uma forma encriptada, ver Figura 4.18.

```
public function store()
{
    $input = Input::all();
    $validator = $this->utilizador->validate($input);

    if($validator->fails()) {
        return Redirect::back()
            ->withInput()
            ->withErrors($validator)
            ->with('danger', Util::message('MSG001'));
    } else {
        $input['password'] = Hash::make(Input::get('password'));
        $this->utilizador->create($input);

        return Redirect::to('utilizador')
            ->with('success', Util::message('MSG002'));
    }
}
```

Figura 4.17 - Função Hash

	id	nome	negoci	username	email	password	ativo	perfil_id
Apagar	1	Administrador	1	admin	admin@ads4shops.com	$2y$10$r4fBTw6Z6Ol1r5tzQhGZTOwemZqtm85Ff6bsX62wUc/...	1	1
Apagar	2	Pedro Valente	1	pmv	pedro@email.com	$2y$10$nwlENiSwodIr7N9YxacXXeozelebbwT5HkQrfWuhXqB...	1	1
Apagar	5	Paulo Lima	7	paulolima	paulolima@cabeleireiroLima.pt	$2y$10$UBIPTObkCk4cG5XJ9qfXhuRmcjsMqZ.g7tlX9/yu3IQ...	1	5
Apagar	6	João Sardinha	6	joaosardinha	joaosardinha@ipvc.pt	$2y$10$HPO6/ALmUTgQEWcpSn8iROS2GdvaZelQTh.IpWwLmS2...	1	5
Apagar	7	Ana Guedes	14	anaguedes	anaguedes@ipvc.pt	$2y$10$.0knEsRXpS9cVUNGPh.p3ebhByWjprrAnAe5EFndK30...	1	6
Apagar	8	Carlos Silva	14	carlossilva	carlossilva@cs.pt	$2y$10$/3tQ/GG.R1myTcupEmDoa.uixzuZd/LAcOulDesJa.I...	1	5

s seleccionados: Muda Apagar Exportar

Figura 4.18 - MySQL - password hash

Depois de autenticados, aparecerá aos utilizadores um menu com as funcionalidades que poderão utilizar, tal como referido anteriormente em 4.2.3.3.2

4.2.5 ACL (Access Control List)

Para resolver o problema de saber se o utilizador tem permissão de executar uma ação, e não ter de estar a verificar através de consultas à base de dados, isso tornaria o sistema muito pesado, supondo que podem estar muitos utilizadores a aceder ao sistema, foi

[10] https://en.wikipedia.org/wiki/Cryptographic_hash_function

[11] https://en.wikipedia.org/wiki/Bcrypt

implementada uma abordagem ACL. Primeiro alteramos a rota em app/routes.php (Figura 4.19).

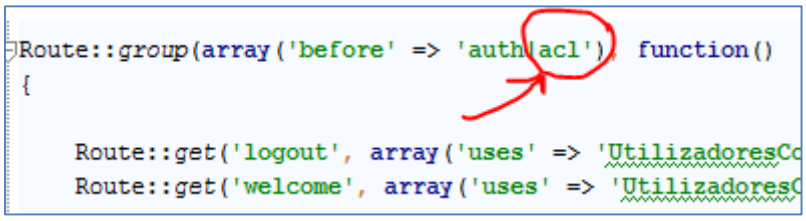

Figura 4.19 - ACL route

Depois criamos o filtro do ACL em app/filters.php (Figura 4.20), onde essencialmente verificamos se a rota condiz com as ações que o utilizador pode executar e com o que está em cache, que será explicado a seguir. E senão acontecer isso, redireciona para vista anterior.

```
Route::filter('acl', function()
{
    //estas ações vão estar disponiveis
    $permission = array('utilizador.welcome', 'utilizador.logout');
    //captura a rota que o utilizador está a usar
    $routeName =  Route::currentRouteName();
    //verificar o ficheiro de cache e concatenar com o id do utilizador
    $cache = Cache::get('actions' .Auth::user()->id);
    //se a rota está a ser exucutada é diferente desta permissão, se é diferente do cache e se temos * na ação que é acesso total
    if( !in_array($routeName, $permission) && !in_array($routeName, $cache) && !in_array('*', $cache)){
    return Redirect::back()
        ->with('danger', Util::message('MSG013'));//exibe mesagem de erro
    }
});
```

Figura 4.20 - ACL filter

4.2.6 CACHE

Depois de fazer a autenticação, o utilizador vai ser direcionado para a vista *welcome*. Verificamos no controller app/controllers/UtilizadoresController.php, no método *welcome* (Figura 4.21) se o utilizador já tem um ficheiro de cache gerado, se não tiver criamos um. Para isso vamos buscar o perfil e as ações associadas e guardamos no *array actions*[] todas as ações do utilizador.

Depois guardamos na cache todos os métodos que existem no *array actions*[].

```
//depois de logar vai ser direcionado para esta vista
public function welcome(){
    //verifica se já tem esse ficheiro gerado em cache concatenado com o id
    if(!Cache::has('actions' . Auth::user()->id)){
        foreach (Auth::user()->perfil->acoes as $acao){//se não tiver criamos
            $actions[] = $acao->metodo;
        }
        Cache::put('actions' . Auth::user()->id, $actions, 120);
    }
    return View::make('utilizadores.welcome');
}
```

Figura 4.21 - método *welcome*

No *logout* (Figura 4.22) é importante apagar o *cache* do respetivo utilizador.

```php
public function logout(){
    if(Cache::has('actions' . Auth::user()->id))
    {
        Cache::forget('actions' . Auth::user()->id);
    }
    Auth::logout();
    return Redirect::to('login')
        ->with('flash_error',Util::message('MSG010'));
}
```

Figura 4.22 - método *logout*

4.2.7 RESTful Resouce Controllers

Os controladores de recursos facilitam a criação de controladores RESTful, à volta dos recursos. Por exemplo, se quisermos criar um controlador que gere as "fotos" guardadas pela nossa aplicação. Usando o comando *controller:make* via *Artisan* (Figura 4.23) na linha de comandos, e do método *Route::resource*, podemos rapidamente criar esse controlador.

Para criar o controlador via linha de comandos, executa-se o seguinte comando:

```
php artisan controller:make FotoController
```

Figura 4.23 - Criar *controller* via comando *artisan*

Agora podemos registar nas rotas :

```
Route::resource('fotos', 'FotoController');
```

Esta rota simples, cria múltiplas rotas, para manusear uma variedade de ações RESTful num recurso 'fotos'.

Da mesma forma, o controlador gerado já terá embebido métodos para cada uma dessas ações com notas informando que *URIs* e verbos aceitam.

Ações manuseadas pelo "*Resource Controller*"

Tabela 4.2 - ações manuseadas pelo "Resource Controller"

Verb	Path	Action	Route Name
GET	/resource	Índex	resource.index
GET	/resource/create	Create	resource.create
POST	/resource	Store	resource.store
GET	/resource/{resource}	Show	resource.show
GET	/resource/{resource}/edit	Edit	resource.edit
PUT/PATCH	/resource/{resource}	Update	resource.update
DELETE	/resource/{resource}	Destroy	resource.destroy

No Ads4shops existem as seguintes:

Domain	URI	Name	Action	Before Filters
	GET|HEAD autologin/{token}	autologin	Watson\Autologin\AutologinController@autologin	
	GET|HEAD /		Closure	
	GET|HEAD login		UtilizadoresController@login	
	POST login		UtilizadoresController@validate	
	GET|HEAD logout	utilizador.logout	UtilizadoresController@logout	auth, acl
	GET|HEAD welcome	utilizador.welcome	UtilizadoresController@welcome	auth, acl
	GET|HEAD utilizador	utilizador.index	UtilizadoresController@index	auth, acl
	GET|HEAD utilizador/create	utilizador.create	UtilizadoresController@create	auth, acl
	POST utilizador	utilizador.store	UtilizadoresController@store	auth, acl
	GET|HEAD utilizador/{utilizador}/edit	utilizador.edit	UtilizadoresController@edit	auth, acl
	PUT utilizador/{utilizador}	utilizador.update	UtilizadoresController@update	auth, acl
	PATCH utilizador/{utilizador}		UtilizadoresController@update	auth, acl
	DELETE utilizador/{utilizador}	utilizador.destroy	UtilizadoresController@destroy	auth, acl
	GET|HEAD perfil	perfil.index	PerfisController@index	auth, acl
	GET|HEAD perfil/create	perfil.create	PerfisController@create	auth, acl
	POST perfil	perfil.store	PerfisController@store	auth, acl
	GET|HEAD perfil/{perfil}/edit	perfil.edit	PerfisController@edit	auth, acl
	PUT perfil/{perfil}	perfil.update	PerfisController@update	auth, acl
	PATCH perfil/{perfil}		PerfisController@update	auth, acl
	DELETE perfil/{perfil}	perfil.destroy	PerfisController@destroy	auth, acl
	GET|HEAD negocio	negocio.index	NegociosController@index	auth, acl
	GET|HEAD negocio/create	negocio.create	NegociosController@create	auth, acl
	POST negocio	negocio.store	NegociosController@store	auth, acl
	GET|HEAD negocio/{negocio}/edit	negocio.edit	NegociosController@edit	auth, acl
	PUT negocio/{negocio}	negocio.update	NegociosController@update	auth, acl
	PATCH negocio/{negocio}		NegociosController@update	auth, acl
	DELETE negocio/{negocio}	negocio.destroy	NegociosController@destroy	auth, acl
	GET|HEAD anuncio	anuncio.index	AnunciosController@index	auth, acl
	GET|HEAD anuncio/create	anuncio.create	AnunciosController@create	auth, acl
	POST anuncio	anuncio.store	AnunciosController@store	auth, acl
	GET|HEAD anuncio/{anuncio}/edit	anuncio.edit	AnunciosController@edit	auth, acl
	PUT anuncio/{anuncio}	anuncio.update	AnunciosController@update	auth, acl
	PATCH anuncio/{anuncio}		AnunciosController@update	auth, acl
	DELETE anuncio/{anuncio}	anuncio.destroy	AnunciosController@destroy	auth, acl
	GET|HEAD ads	ads.index	AdsController@index	auth, acl
	GET|HEAD ads/create	ads.create	AdsController@create	auth, acl
	POST ads	ads.store	AdsController@store	auth, acl
	GET|HEAD ads/{ads}	ads.show	AdsController@show	auth, acl
	GET|HEAD ads/{ads}/edit	ads.edit	AdsController@edit	auth, acl
	PUT ads/{ads}	ads.update	AdsController@update	auth, acl
	PATCH ads/{ads}		AdsController@update	auth, acl
	DELETE ads/{ads}	ads.destroy	AdsController@destroy	auth, acl
	POST adsupdate	ads.adsupdate	AdsController@adsupdate	auth, acl
	GET|HEAD automaticlogin	automaticlogin.index	AutomaticLoginController@index	auth, acl

Figura 4.24 - php artisan routes - Ads4shops

4.2.8 EMAIL

O *Laravel* fornece uma *API* simples da popular livraria *SwiftMailer*[12].

O ficheiro de configuração do *email* está em app/config/mail.php, e contem opções que permitem mudar o *host* SMTP, portos e credenciais, assim como definir um endereço global "*from*" para todas as mensagens que serão entregues pela livraria.

Uma das particularidades desta *API* é que tem drivers para *SMTP, Mailgun, Mandrill, Amazon SES*, função mail do PHP e *sendmail*, sendo facilmente configurável e podendo enviar emails de um serviço local ou baseado na *web* (Figura 4.25).

[12] SensioLabs, http://swiftmailer.org/

```php
<?php

return array(

    'driver' => 'smtp',

    'host' => 'smtp.gmail.com',

    'port' => 587,

    'from' => array('address' => 'ads4shops@gmail.com', 'name' => 'Ads4Shops'),

    'encryption' => 'tls',

    'username' => 'ads4shops@gmail.com',

    'password' => '[redacted]',

    'sendmail' => '/usr/sbin/sendmail -bs',

    'pretend' => false,

);
```

Figura 4.25 - *Email setup*

4.2.8.1 *CONTROLLER*

No *controller* AnunciosController.php, no método *store()*, sempre que um anúncio é guardado, é enviado um email ao administrador com o anúncio que foi inserido e com os respetivos dados do utilizador, nome do anúncio e datas de inicio e fim (Figura 4.26).

```php
Mail::send('emails.newad',
    array( 'nomeanuncio' => Input::get('nome'),
        'datainicio' => Input::get('inicio'),
        'datafim' => Input::get('fim'),
        'user' => Auth::user()->nome,
    ),
    function($message){
        $message->to('ads4shops@gmail.com', 'Administrador')->subject('novo anuncio');
});
```

Figura 4.26 - Email *controller*

A Figura 4.27, mostra o aspeto do email recebido pelo administrador.

Figura 4.27 - Email

4.3 FASE II - VISUALIZAÇÃO DA INFORMAÇÃO

Nesta fase foi tratada a parte da visualização dos anúncios nas lojas aderentes.

Tentamos criar uma interface que permita o visionamento dos anúncios publicados pelos anunciantes (Figura 4.28).

Foi preciso obedecer a certas restrições:

O administrador deve validar os anúncios quanto ao conteúdo e duração, utilizando para isso um menu específico (Figura 4.29). Procuramos que o administrador através de um *preview* consiga visualizar o vídeo e ver a sua duração. Para isso tomamos o cuidado de na *tag* <vídeo> do HTML5 apenas fazer o *preload* dos metadados e não do vídeo todo, para não gerar muito trafego.

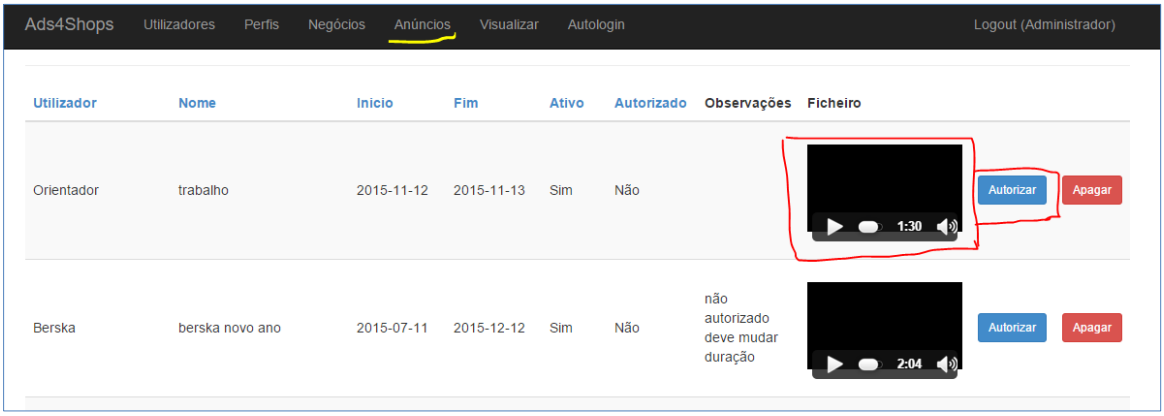

Figura 4.28 - Menu anúncio modo administrador

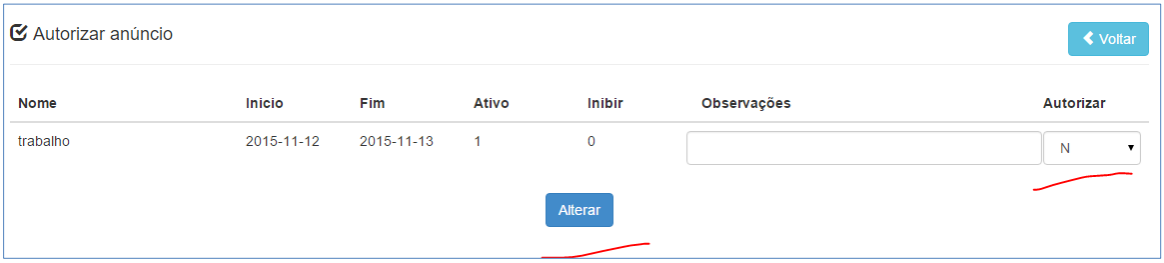

Figura 4.29 - Autorizar anúncio

Também foi preciso criar um algoritmo que permita apenas que sejam postos na lista de reprodução apenas X anúncios de cada utilizador, que não passem anúncios de negócios iguais ao do utilizador que está em sessão e que as listas de reprodução sejam aleatoriamente diferentes de utilizador para utilizador. Isso foi feito no *controller* específico, neste caso o *AdsController.php*, onde fazemos a consulta à base de dados (Figura 4.30).

```
private function adsEng($id)
{

    $user = Auth::user();

    //consulta à DB
    DB::statement(DB::raw('SET @prev=0, @rownum=0, @at=1, @au=1'));

    $resultados =
        DB::select(
            DB::raw("
      SELECT `utilizador_id`, `nome`, `path`
      FROM (
        SELECT *,
             IF( @prev <> `utilizador_id`,
                 @rownum := 1,
                 @rownum := @rownum + 1
             ) AS `rank`,
             @prev := utilizador_id,
             @rownum
        FROM (
          SELECT `ads`.* FROM `anuncios` ads
          INNER JOIN `utilizadores` `users` on `ads`.`utilizador_id` = `users`.`id`
          WHERE `ads`.`autorizado` = @au
          AND `ads`.`ativo` = @at
          AND `ads`.`fim` >= CURRENT_DATE
          AND `ads`.`inicio` <= CURRENT_DATE
          AND (`users`.negocio_id <> :negocio_id  OR `ads`.`utilizador_id` = :user_id)
          ORDER BY `ads`.`utilizador_id`, rand()
        ) AS `random_ads`
      ) AS `ads_ranked`
      WHERE `rank` <= 2;
    "), ['negocio_id' => $user->negocio_id, 'user_id' => $user->id]);
    //fim consulta à DB
```

Figura 4.30 - consulta à BD

Tendo em conta o objetivo da operação, procedemos à operação de "baralhar" (*shuffle*) o *array* resultante da consulta, para que a lista seja o mais aleatória possível e os vídeos não sejam reproduzidos sempre na mesma ordem (Figura 4.31),

```
$path = array();

foreach ($resultados as $r){
    $path[] = $r->path;
}
//baralha o array com os nome dos ficheiros para a playlist
shuffle($path);
```

Figura 4.31 - criação do array resultante da consulta à BD

e criamos a lista *xml* para enviar para a vista que tem o reprodutor de vídeo (Figura 4.32).

```
//escreve o conteudo XML RSS
$output="<rss version='2.0' xmlns:jwplayer='http://rss.jwpcdn.com/'>". PHP_EOL ."<channel>".
PHP_EOL;

foreach($path as $p){
    $output .= "" . PHP_EOL . "\t<item>" . "<jwplayer:source file='$p'/>" . "</item>" . PHP_EOL;
}

$output .= PHP_EOL ."</channel>". PHP_EOL . "</rss>";

//escreve no ficheiro xmlfile o conteudo RSS
file_put_contents($xmlfile,$output);

//caso o ficheiro da cache não existe cria uma nova
if(!Cache::has('actions' . Auth::user()->id)){
    foreach (Auth::user()->perfil->acoes as $acao){//se não tiver criamos
        $actions[] = $acao->metodo;
    }
    Cache::put('actions' . Auth::user()->id, $actions, 120);
}

return $xmlfile;
```

Figura 4.32 - criação da lista de reprodução XML

4.3.1 REPRODUTOR DE VÍDEO

O reprodutor de vídeo escolhido foi o JWPLAYER[13]. Escolhemos este reprodutor, pois satisfaz todos os requisitos necessários à elaboração do projeto. O JWPLAYER é escrito em *JavaScript*, é multiplataforma, permite *fullscreen*, suporta os *codecs* de áudio (AAC, MP3 e Vorbis) e vídeo (MP4, WebM e FLV), permite a deteção de erros na reprodução. Também permite a utilização de listas de reprodução, condição essencial para o projeto.

[13] https://www.jwplayer.com/

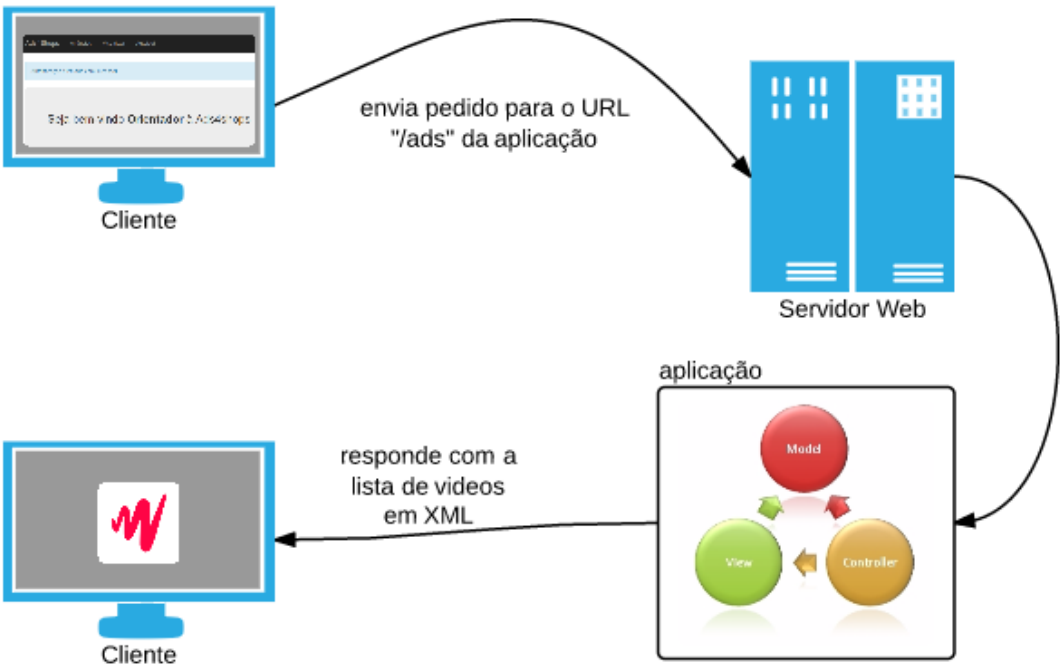

Figura 4.33 - Processo de visualização de anúncios

Na Figura 4.33, podemos ver o processo de visualização de anúncios da plataforma, que começa por um pedido do cliente que carrega no link Visualização do menu, o servidor Web processa o pedido e através da rota correspondente, chama o controlador e despoleta o processo MVC conforme descrito anteriormente em 4.2.3, que gera uma lista em XML conforme anteriormente descrito em 4.3, que depois vai ser reproduzida pelo reprodutor de vídeo, conforme descrito mais à frente em 4.3.2.

4.3.1.1 INSTALAÇÃO

A instalação do *JWPlayer* requer que se faça um registo no *site* da empresa proprietária, para fazer o *download* do *player* e da respetiva chave de ativação, e só se pode usar em utilizações não comerciais.

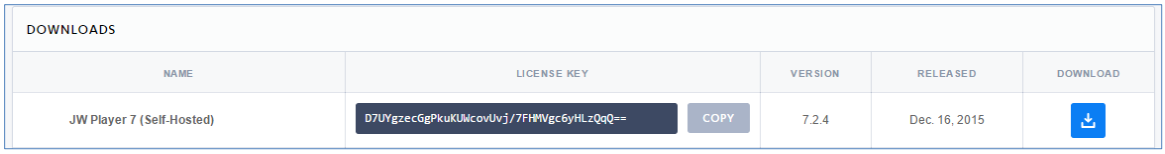

Figura 4.34 - *JWPlayer download*

Como escolhemos uma versão *self-hosted player*, devemos iniciar a biblioteca *Javascript* e depois fazer referência à chave da versão *self-hosted* (Figura 4.35).

```
<script src="assets/jwplayer/jwplayer.js" ></script>
<script> jwplayer.key="D7UYgzecGgPkuKUWcovUvj/7FHMVgc6yHLzQqQ==";</script>
```

Figura 4.35 - referencia à chave JWPlayer

4.3.2.1 CONFIGURAÇÃO

A configuração do *JWplayer* é feita em código na *view* onde os vídeos são reproduzidos, e baseando-nos na documentação *Javascript* API na página de suporte da empresa[14], nos fóruns da empresa e no fórum do *site* stackoverflow[15], primeiro configuramos o *setup* do *JWPlayer* (Figura 4.36), em que escolhemos as seguintes opções: localização da playlist, se desejamos o *autostart*, o modo de *render preferido,* neste caso foi o html5, o tipo de *streching, aqui escolhemos o "uniform"*, escolhemos ainda desligar o *visualplaylist (visualização das listas),* a configuração do logotipo, desligar a barra de controle do vídeo e também fizemos desligamos o som.

```
function inicio(){
    playerInstance.setup({
        debug: true,
        playlist: rss,
        autostart: true,
        controls: false,
        primary: "html5",
        streching: "exacfit",
        controls: true,
        visualplaylist: false,
        logo: {
            file: 'assets/img/acice_logo.png',
            position: 'bottom-right',
            margin: '0'
        },
        height: "100%",
        width: "100%",
        preload: "auto",
}).setMute(true);
}
```

Figura 4.36 - setup do jwplayer

Também foi necessário a falta de criar uma solução para acesso ao servidor da aplicação, evitando as respetivas e desagradáveis mensagens de erro, como por exemplo:

Error loading media: File not found

Neste caso, criamos e chamamos uma função local (Figura 4.37), que despoleta o processo de reproduzir um vídeo de apoio que estará disponível localmente na maquina do cliente sendo essa localização guardada no perfil do utilizador e lida da base de dados para uma variável erro, pois assim cada utilizador poderá guardar o ficheiro em sítios diferentes, e que entrara em ação evitando assim a que mensagem anteriormente mencionada, bloqueie todo o sistema.

[14] https://support.jwplayer.com/customer/portal/articles/1413089-javascript-api-reference

[15] http://stackoverflow.com/

```
//faz correr video local. se no fim de o correr não existe internet corre outra vez
function local() {
    playerInstance.load({file: erro});
    playerInstance.play();
    playerInstance.on('complete', function () {
        if (doesConnectionExist()) {
            updateRss();
        } else {
            local();
        }
    });
}
```

Figura 4.37 - função local(), no reprodutor de video

Quando o vídeo de apoio chegar ao fim, é testado através de um método se existe acesso ao site, para isso foi usada a libraria Offline.js[16] que funciona fazendo um *request* XHR[17] para carregar um ficheiro muito pequeno, o *favicon.ico* (evitando assim gerar muito tráfego) e verificar a ligação. Temos de nos certificar de que a URL de destino tem a mesma origem que nossa página (o método de ligação, domínio e porta, devem ser todos o mesmo) (Figura 4.38), ou vamos ter problemas *Cross-Origin Resource Sharing* (CORS)[18]

Caso a ligação não exista o ficheiro local é carregado novamente, caso a ligação exista, é então gerada uma nova lista *rss* e depois carregada.

```
Offline.options = {
    checks: {
        xhr: {
            url: '/ads'
        }
    }
};
```

Figura 4.38 - Offline opções

Nesta fase deparamos com o problema do *fullscreen*, pois o carregamento da lista de vídeos obriga (por questões de segurança do *browser*) a sair do *fullscreen*, problema resolvido na fase seguinte.

4.3.3 AUTOLOGIN

Como era preciso criar uma solução de *autologin* autenticado, recorremos a uma solução de um membro da comunidade *Laravel*, mais propriamente de Dwight Watson (Watson, s.d.), que forneceu uma solução através de um pacote chamado *Autologin*, "*Autologin is a package for Laravel 4 which makes the process of generating links that will automate*

[16] Livraria open-source para testar ligação à rede, disponível em http://github.hubspot.com/offline/docs/welcome/

[17] XMLHttpRequest, é uma API disponível em linguagens de script para navegadores web tais como JavaScript.

[18] https://www.w3.org/TR/cors/

login to your application super simple. It works great for when sending out transactional emails and..." (Watson, s.d.).

O objetivo é a criação de uma hiperligação com caminho para /ads (abre o reprodutor de vídeo automaticamente).

O processo de instalação não foi fácil, pois não podendo usar o *composer*, teve de ser feito na máquina local e depois manualmente transferido para o sitio *web*, o que nem sempre corre bem.

No ficheiro /app/config/app.php foi acrescentado um novo "*Service Provider*" (Figura 4.39),

```
'Illuminate\Workbench\WorkbenchServiceProvider',
'Barryvdh\LaravelIdeHelper\IdeHelperServiceProvider',
'Watson\Autologin\AutologinServiceProvider',
```

Figura 4.39 - *Service Provider app.php*

e também para poder usar uma *Facade,* foi acrescentado no mesmo ficheiro /app/config/app.php (Figura 4.40).

```
'Validator'       => 'Illuminate\Support\Facades\Validator',
'View'            => 'Illuminate\Support\Facades\View',
'Autologin'       => 'Watson\Autologin\Facades\Autologin',
```

Figura 4.40 - *Facade Autologin app.php*

Foi também preciso analisar o pacote fornecido para descobrir qual a tabela que era preciso criar e quais os campos, pois isso não está documentado na página do autor.

Assim de acordo com as *migrations* da estrutura do autor, teve de ser criada uma tabela chamada *autologin_tokens* com os campos como na Figura 4.41.

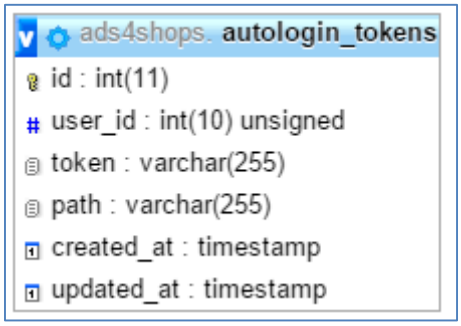

Figura 4.41 - Tabela autologin_tokens

O processo de *autologin* é então despoletado pelo utilizador, selecionando a opção apropriada no menu (Figura 4.42).

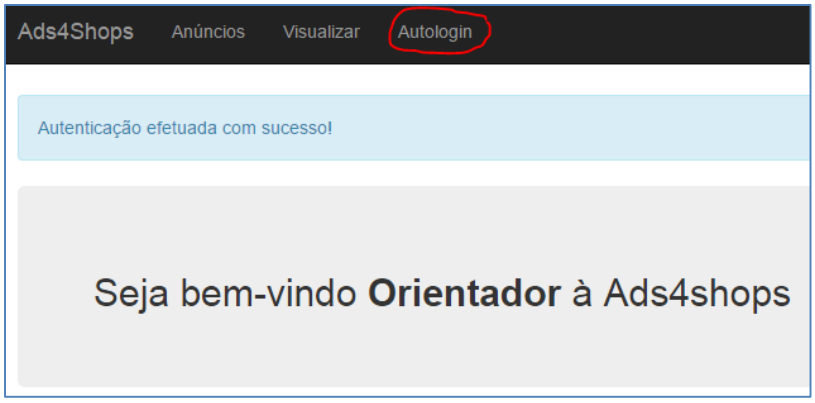

Figura 4.42 - Autologin menu

O controlador *AutomaticLoginController.php* (Figura 4.43), desencadeia o processo de criação da hiperligação,

```
public function index()
{
    $id = Auth::user()->id;
    $utilizador = Utilizador::find($id);
    $link = Autologin::to($utilizador, '/ads');
    return View::make('automaticlogin.index')
        ->with('link', $link);

}
```

Figura 4.43 – automaticlogincontroller

onde é passado o id do utilizador em sessão, e criada a hiperligação. É depois redirecionado para a vista automaticlogin.index (Figura 4.44), onde a hiperligação é visualizada no ecrã (Figura 4.45).

```
@extends('layouts.admin')


@section('content')

{{$link}}

@stop
```

Figura 4.44 - automaticlogin view

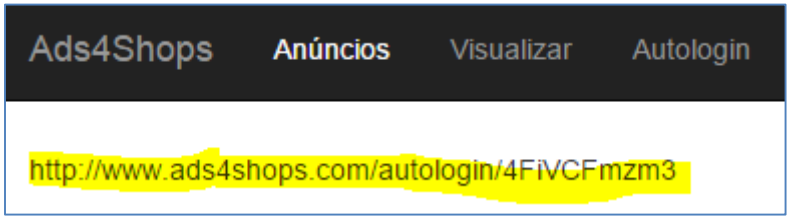

Figura 4.45 - Autologin

Alteramos ainda a duração em minutos do token para estar mais de acordo com os objetivos em /vendor/watson/autologin/src/config/config.php (Figura 4.46).

```
|-----------------------------------------------------------------------
| Token lifetime
|-----------------------------------------------------------------------
|
| Here you may specifiy the number of minutes you wish the autologin token
| to remain active.
|
*/

'lifetime' => 1440,
```

Figura 4.46 – Autologin lifetime token

A hiperligação depois de gerada poderá ser introduzida por exemplo num atalho para maior facilidade de acesso à plataforma, numa perspetiva de aceder logo à reprodução de anúncios.

4.4 FASE III - IMPLEMENTAÇÃO

Foi efetuada a implementação do projeto, os ajustes necessários a requisitos que o orientador sugeriu, os quais não foram considerados no início.

Para minimizar o problema do *fullscreen*, em que depois de muita pesquisa, optou-se por uma solução de uma extensão do *Google Chrome*, o *Kiosk*[19] (Figura 4.47).

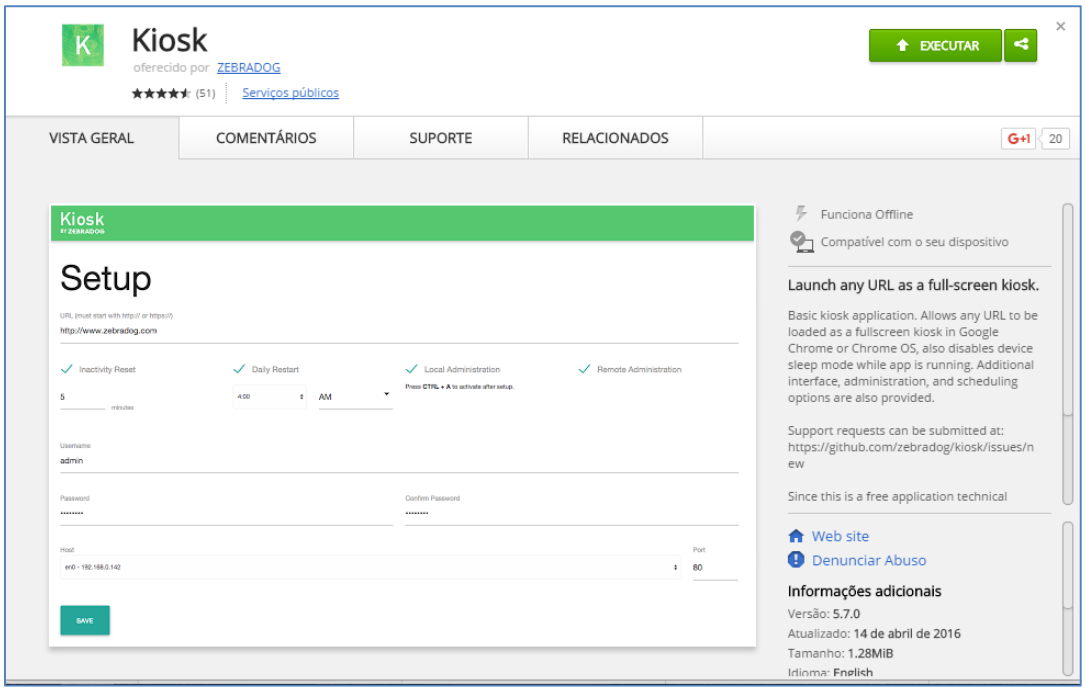

Figura 4.47 - Kiosk - chrome app

[19] https://chrome.google.com/webstore/detail/kiosk/afhcomalholahplbjhnmahkoekoijban

Ou uma solução em *Linux* que arranca diretamente para o *Chromium* em modo *kiosk*.

A migração da plataforma para um ambiente real de testes correu com alguma dificuldade, devido a condicionantes técnicas derivadas do alojamento não ser administrado por nós e conter outro site em simultâneo, tendo de ter sido aplicadas algumas técnicas de servidor Web mais complicadas, como por exemplo mexer no ficheiro "*.htaccess"* (quando um ficheiro "*.htaccess"* é colocado num diretório que por sua vez é "carregado via servidor Web Apache", o ficheiro "*.htaccess"* é detetado e executado pelo software Apache Web Server) ou alterar a estrutura tradicional do *Laravel*, para ir de encontro as configurações do alojamento.

.

4.5 NOTAS FINAIS

Neste capítulo foi descrito o desenvolvimento do projeto. Foi descrita a análise dos requisitos, a modelação do sistema, descreveu-se a criação da plataforma, utilizando para isso um *framework* PHP, um reprodutor de vídeo e um alojamento web para que fossem criadas as condições de funcionamento em produção.

A implementação deste projeto permitiu conhecer de perto e de uma forma bastante aprofundada, o processo de criação de uma solução que se aproxima bastante de uma solução comercial, com todas as dificuldades e entraves que se podem encontrar num projeto desta dimensão.

As principais dificuldades encontradas, foram primeiro a compreensão e adaptação a um *framework* PHP devido à pouca experiência em programação, a pouca experiência no levantamento de requisitos, que levou a que o projeto fosse alterado algumas vezes, o problema de os *browsers* atuais não permitirem o *fullscreen* automático, o problema dos dispositivos móveis não permitirem o *autoplay*, a passagem da plataforma de um *host* comercial para um outro com características diferentes.

Gostaríamos ainda de ter implementado um sistema de registos, que permita em tempo real saber quais os vídeos que passaram e em que cliente, a possibilidade de poder enviar email a avisar o utilizador que o seu vídeo foi aprovado pelo administrador.

5 AVALIAÇÃO OU DISCUSSÃO DOS RESULTADOS DO TRABALHO

5.1 INTRODUÇÃO

Tendo como objetivo a implementação deste sistema num ambiente real, era preciso testar o sistema, quer ao nível de funcionamento da sua interface e funcionalidade, quer ao nível de potenciais tipos de hardware a utilizar, para isso a colaboração do Dr. Pedro Carneiro revelou-se extremamente profícua, já que foi possível realizar testes num ambiente real.

5.2 TESTE DE FUNCIONALIDADE E USABILIDADE DO SISTEMA

De acordo com os objetivos, o sistema foi testado por alguns potenciais interessados quanto à sua funcionalidade e usabilidade. Foram criados para o efeito algumas contas a simular anunciantes de diferentes tipos de negócios, onde se testou por exemplo o carregamento de vídeos excedendo o limite de tamanho permitido, e o tipo de ficheiro, a passagem de vídeos em anunciantes concorrentes, a aleatoriedade das listas de vídeos, a facilidade de utilização da interface gráfica, a utilização do sistema em diversos tipos de ecrãs e dispositivos. Os testes efetuados foram todos realizados com sucesso e os resultados de acordo com as nossas espectativas.

5.3 *TESTE DE HARWDWARE*

Tendo como objetivo a implementação deste sistema em lojas, era preciso encontrar o hardware mais eficaz para o efeito, para isso teve-se em conta os seguintes aspetos, tamanho (é preciso que o dispositivo seja o mais silencioso e discreto possível), preço, ligação via *ethernet* e/ou *wi-fi*, poder de processamento gráfico, necessário ao bom funcionamento da visualização dos vídeos.

Para isso e após alguma pesquisa no mercado optou-se por testar uma unidade *Raspeberry PI 2B*, um *tablet android* e um ODROID C2.

5.3.1 RASPEBERRY PI 2 B

O Modelo B do *Raspberry Pi 2* é a segunda geração que veio substituir em fevereiro de 2015, o *Raspberry Pi* 1 Modelo B+.

O *Raspberry Pi* 2 B, tem as seguintes características:

- CPU quad-core ARM Cortex-A7 a 900MHz
- 1GB de RAM
- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port

- Micro SD card slot
- VideoCore IV 3D graphics core

Foram feitas várias tentativas com diversos sistemas e soluções (Tabela 5.1). Foi testado com *kernels* diferentes, e vídeos de diversas resoluções e tamanhos, mas encrava em todas as situações umas vezes mal arranca outras vezes ao fim de alguns minutos, a causa parece ser a baixa capacidade de processamento e memória do GPU e o modo *fullscreen* que exige recursos mais elevados.

Tabela 5.1 - Testes Hardware

S.Operativos	*Kernel*	*Browser*	Modo Normal	Modo *Kiosk*
Raspbian	4.4.13-v7+	*Chromium*	Encrava ao fim de poucos minutos	Encrava ao fim de poucos minutos
Raspbian	4.4.13-v7+	*Epiphany*	Encrava ao fim de poucos minutos	Encrava ao fim de poucos minutos
Raspbian	4.4.13-v7+	*Kweb*	Encrava ao fim de poucos minutos	Encrava ao fim de poucos minutos
FullPageOS	4.4.9-v7+	*Chromium*		Ao fim de alguns minutos começa a lagar e depois encrava
Ubuntu Mate 16.04	4.1.9-v7+	*Firefox*	Encrava passado alguns minutos	

5.3.2 TABLET COM SISTEMA OPERATIVO ANDROID

Procurou-se testar uma solução para dispositivos *Android* nomeadamente *tablets*. Aqui e na unidade testada um Samsung E 9.6 com as seguintes especificações:

- CPU quad-core 1,3 Mhz
- 1,5GB de RAM
- Sistema Operativo Android 4.4.4

Apesar de o hardware escolhido, reproduzir os vídeos, o problema é que no sistema Android não é possível, devido a restrições do sistema operativo, fazer reprodução automática, o que para o projeto é considerado como inviável.

5.3.3 ODROID C2

The ODROID-C2 is a 64-bit quad-core single board computer(SBC) that is one of the most cost-effective 64bit development boards available in the ARM world.It can function as a home theater set-top box, a general purpose computer for web browsing, gaming and socializing, a compact tool for college or office work, a prototyping device for hardware tinkering, a controller for home automation, a workstation for software development, and much more (HardKernel, 2016).

O modelo Odroid C2 tem as seguintes características:

- CPU quad-core Amlogic ARM Cortex-A53 (ARMv8) a 2GHz
- 2GB DDR3 de SRAM
- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Gigabit Ethernet port
- Micro SD card slot
- eMMC5.0 HS400 Flash Storage
- GPU ARM Mali-450
- VPU capaz de H.265 4K/60FPS e H.264 4K/30FPS

O sistema operativo testado foi o *Ubuntu Mate 16.04*.

Foi preciso criar alguns *scripts* e "*works arrounds*", para que a unidade testada funcionasse como pretendido, a saber:

- Arranque em modo *kiosk* diretamente para o *login* do cliente e reproduzindo a lista de vídeos. Foi criado para isso um script em *bash*[20] (***kiosk.sh***), que permite arrancar o *Chromium*[21] em modo *kiosk* diretamente para o reprodutor de vídeos do cliente (ver Figura 5.1)
- Utilização de uma versão *Chromium* modificada, pois a que está nos repositórios oficiais não corre vídeos H.264. foi escolhida a ***ppa:saiarcot895/chromium-dev***
- Script (***ler_temp.sh***) para monitorização da temperatura, feito em *bash* (ver Figura 5.2), que permite ler a temperatura do *cpu* e a frequência de cinco em cinco segundos e escreve para o ecrã. Correndo o seguinte comando ***#./ler_temp .sh | tee c2_c_caixa.txt*** produz um ficheiro ***c2_c_caixa.txt*** com os valores das leituras (ver Figura 5.3).
- Script (***c2plot.gp***) para impressão dos valores para análise (ver Figura 5.4), que é executado pelo comando seguinte ***#gnuplot -c2 c2plot.gp c2_c_caixa.txt c2_c_caixa.png*** para produzir os gráficos das Figura 5.5,Figura 5.6 e Figura 5.7.

```
#!/bin/bash
export GOOGLE_API_KEY="no"
export GOOGLE_DEFAULT_CLIENT_ID="no"
export GOOGLE_DEFAULT_CLIENT_SECRET="no"
chromium-browser --kiosk http://ads4shops.com/ads/autologin/6wt7Lsr0ju
```

Figura 5.1 - script modo Kiosk

[20] https://www.gnu.org/software/bash/

[21] https://www.chromium.org/

```
#!/bin/bash
t=0
while true :
do
 a=`cat /sys/devices/virtual/thermal/thermal_zone0/temp`
 b=`cat
 /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq`
 c=`cat
 /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_cur_freq`
 (( t += 5 ))
 echo $t,$((a/1000)),$b,$c
 sleep 5
done
```

Figura 5.2 - ficheiro ler_temp.sh

```
5,47000,2016000,
10,48000,2016000,
15,49000,2016000,
20,51000,2016000,
25,52000,2016000,
30,52000,2016000,
35,52000,2016000,
40,52000,2016000,
45,52000,2016000,
50,53000,2016000,
55,52000,2016000,
60,53000,2016000,
```

Figura 5.3 - valores temperatura e frequência Odroid C2

```
#!/usr/bin/gnuplot --persist
print "script name          : ", ARG0
print "data to plot         : ", ARG1
print "PNG output file      : ", ARG2
set term pngcairo size 700,400 enhanced font 'Verdana,9'
set output ARG2
set datafile separator ","

set title 'Ads4Shops - Odroid C2 com cooler e caixa'
set xlabel 'Tempo (segundos)'
set ylabel 'Frequencia'
set y2label 'Temperatura'
set yrange [0:2500000]
set ytics border nomirror
set y2tics border nomirror
set key left top
set style line 11 lc rgb '#808080' lt 1
set border 3 back ls 11
set tics nomirror
set style line 12 lc rgb '#808080' lt 0 lw 1
set grid back ls 12
set grid y2tics lt 0 lw 1 lc rgb "#008800"
# color definitions
set style line 2 lc rgb '#8b1a0e'  lw 2
set style line 1 lc rgb '#5e9c36'  lw 2
plot ARG1 using 1:3 w lines ls 1 title 'Frequencia', ARG1 using 1:2 w
lines ls 2 axes x1y2 title 'Temperatura'
```

Figura 5.4 - script c2plot.gp

A unidade testada conseguiu correr a plataforma Ads4shops sem percalços nos três testes efetuados, o primeiro sem caixa, o segundo com caixa e o terceiro com caixa e com *cooler*,

todos os testes foram executados com uma temperatura ambiente de 22 graus Celcius, e correram vídeos da plataforma Ads4shops durante aproximadamente 50 min.

No primeiro teste, sem caixa, a temperatura da unidade testada limitou-se entre os 58 e os 62 graus Celcius (ver Figura 5.5), no segundo teste com caixa a temperatura limitou-se entre os 70 e os 73 graus Celcius (ver Figura 5.6), no terceiro teste a temperatura limitou-se entre os 44 e os 48 graus Celcius.

Tabela 5.2 - teste temperatura Odroid C2

Caixa	Cooler	Temperatura média em graus Celcius
Não	Não	60
Sim	Não	71,5
Sim	Sim	46

Como se pode ver pela tabela anterior (Tabela 5.2), a versão ensaiada com cooler é a que melhor desempenho em termos de temperatura tem. O que permite utilizar o hardware mesmo em condições de temperatura exteriores mais elevadas.

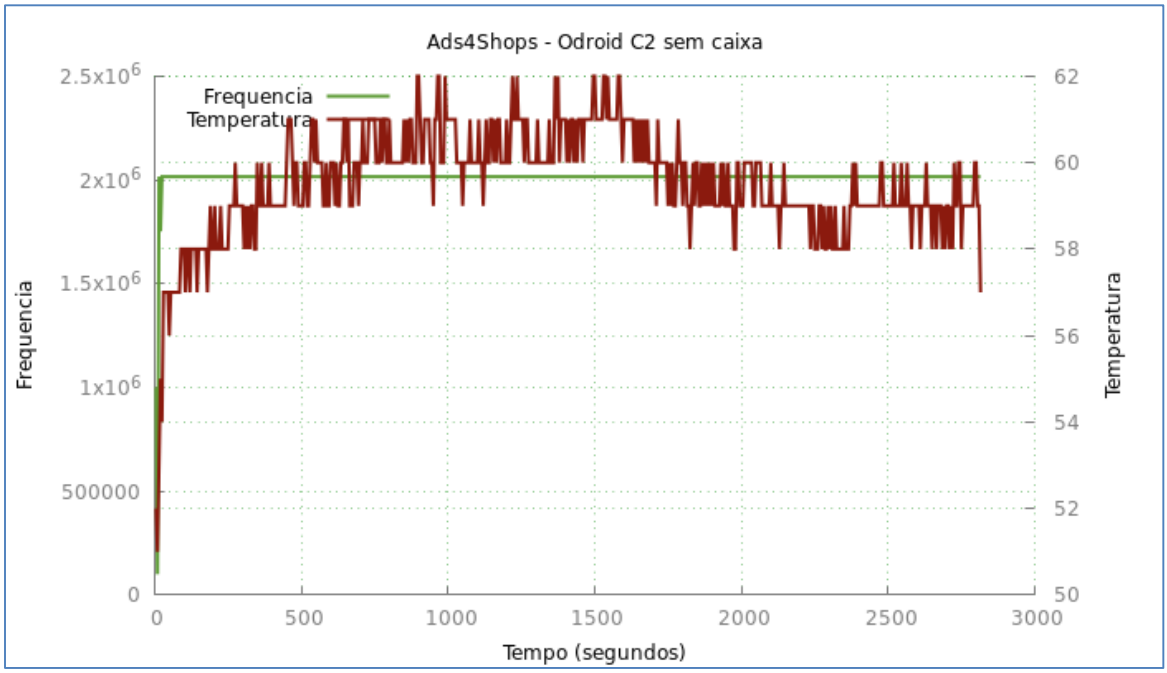

Figura 5.5 - teste Odroid C2 sem caixa

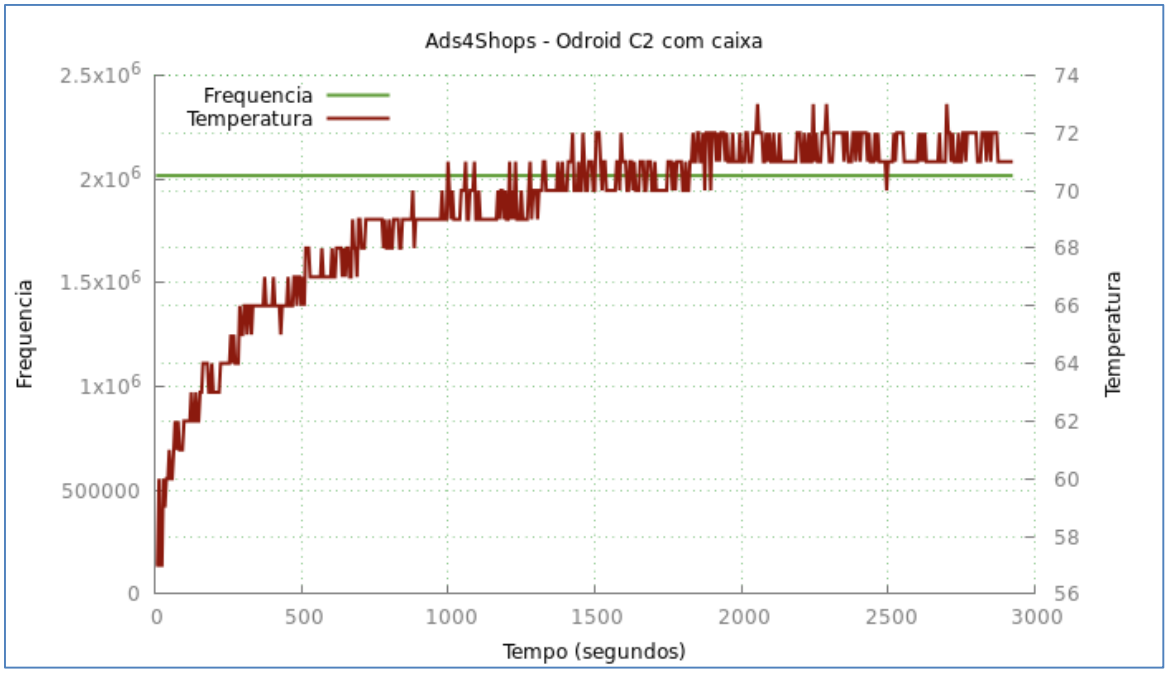

Figura 5.6 - teste Odroid C2 com caixa

Figura 5.7 - teste Odroid C2com caixa e cooler

5.4 NOTAS FINAIS

Ao nível da interface gráfica e da funcionalidade da plataforma, depois dos testes efetuados, não encontramos qualquer problema, a plataforma portou-se de acordo com os objetivos inicialmente delineados.

Quanto as unidades de *hardware* e avaliando as várias unidades estudadas, podemos concluir que, a que mais interessa a este projeto é a unidade Odroid C2 com caixa e *cooler*, da empresa sul coreana *HardKernel*, pois foi a única que obteve um bom desempenho na

reprodução dos vídeos disponibilizados pelos anunciantes, é relativamente acessível quanto ao preço (aproximadamente 100 euros) e permite a utilização de um sistema operativo gratuito como por exemplo o *Ubuntu Mate* 16.04., e sendo ao mesmo tempo uma solução elegante pois é uma caixa muito pequena, que se vai tornar muito discreta em qualquer lugar que se utilize.

Este conjunto de testes foi muito importante, pois podemos perceber o que funcionava menos bem e corrigi-lo. Também nos permitiu ter uma ideia mais precisa do tipo de hardware necessário para instalar o projeto em ambiente real.

6 CONCLUSÕES E TRABALHO FUTURO

A utilização de sistemas web para gestão de anúncios vídeo em *streaming* é extremamente vantajosa, pois permite a quem administra o sistema aceder de qualquer parte, precisando para isso apenas de um acesso à Internet, também a mesma vantagem é visível para os clientes, o que lhes traz bastante comodidade na manutenção dos seus objetivos ao poder rapidamente controlar os seus anúncios.

O principal objetivo deste trabalho, era criar um sistema de gestão de anúncios em vídeo, acessível via Browser/Internet. Existia ainda um outro, este relacionado com a minha profissão, professor, que era aprender novas tecnologias e ferramentas. Ambos estes objetivos foram cumpridos.

Conseguimos criar um sistema de gestão de anúncios, acessível através da Internet, utilizando para isso um web browser. Os objetivos iniciais foram cumpridos na totalidade e ainda foram implementadas novas funcionalidades ao longo do desenvolvimento do projeto que não foram previstas de inicio, funcionalidades essas que deram à aplicação um nível muito bom. Por fim o projeto foi implementado num ambiente real para realização de testes, onde foram encontrados alguns problemas. Uns foram resolvidos, outros ainda carecem de mais pesquisa para que se possa ultrapassá-los.

Como conclusão final, pode-se dizer que o projeto atingiu um nível de complexidade grande e que este trabalho foi apenas um passo num caminho a percorrer.

Em termos de trabalho futuro, existe alguns melhoramentos que podem ser conseguidos, o problema do *fullscreen,* que pensamos que poderá ser resolvido recorrendo à alteração do código de um *browser open-source*, também poderia ser implementado um sistema de parametrização das consultas a ser gerido pelo administrador, podendo este alterar por exemplo o número de vídeos a visualizar de cada utilizador, a instalação de um sistema de registos dos vídeos que vão sendo reproduzidos, quando e em que cliente, o aviso ao anunciante através de um email, que o seu anuncio foi aprovado ou não.

Também pensamos que seria possível e até desejável criar um conjunto de hardware/software pronto a usar, onde se pudesse simplesmente ligar o aparelho à corrente elétrica e o sistema começar a funcionar sozinho, conjunto esse que já se encontra em estudo.

Referências

Alves, E. R. e. L. M., 2010. *Publicidade Online: O poder das mídias e redes sociais.* Goiás: s.n.

Anon., s.d. *Swiftmailer.* [Online]
Available at: http://swiftmailer.org/

Associação Brasileira de Mídia Out of Home, 2015. *UBIQUIDADE.* [Online]
Available at: http://www.abmooh.com.br/midia-digital-out-of-home/

Booch, G., Jacobson, I. & Rumbaugh,, J., 1999. *Unified Modeling.* s.l.:Addison Wesley Longman, Inc.

Digital Signage Connetion, 2013. *Ten Years Strong: The History of Digital Signage Expo.* [Online]
Available at: http://www.digitalsignageconnection.com/ten-years-strong-history-digital-signage-expo-dse-992
[Acedido em 6 2015].

HardKernel, 2016. *HardKernel - Produtcs - Odroid C2.* [Online]
Available at: http://www.hardkernel.com/main/products/prdt_info.php
[Acedido em 28 6 2016].

Inbox - Soluções Digitais, 2015. *Sinalização Digital.* [Online]
Available at: http://www.inboxmidia.com.br/sinalizacao_vantagens.php
[Acedido em 5 2015].

Intel, 2015. *Intel® Retail Client Manager (Intel® RCM) Software.* [Online]
Available at: http://www.intel.com/content/www/us/en/retail/rcm.html

Järvinen, P., 1997. *Action Research is Similar to Design Science.* s.l.:s.n.

Leite, J. & Freeman, P., 1991. Requirements validation through viewpoint resolution. 17(12).

Mason, N., 2006. *Is operations research really research?.* s.l.:s.n.

Mawston, N., 2011. *One Billion HTML5 Phones to be Sold Worldwide in 2013.* [Online]
Available at: https://www.strategyanalytics.com/strategy-analytics/news/strategy-analytics-press-releases/strategy-analytics-press-release/2011/12/07/one-billion-html5-phones-to-be-sold-worldwide-in-2013#.VpNydOiLTIU
[Acedido em 15 12 2015].

Outdoor Advertising Association of America, Inc., 2015. *Out of Home Advertising.* [Online]
Available at: http://www.oaaa.org/OutofHomeAdvertising/OutofHomeAdvertising.aspx

Schaeffler, J., 2008. *Digital Signage: Software, Networks, Advertising, and Displays: A Primer for Understanding the Business.* s.l.:Focal Press.

Sommerville, I. & Kotonya, G., 1998. *Requirements Engineering: Processes and Techniques.* s.l.:John Wiley & Sons, Inc..

Sommerville, I. & Sawyer, P., 1997. *Requirements Engineering: A good pratice guide.* s.l.:John Wiley & Sons, Inc..

Spring Signate, Ltd, 2015. *Xibo.* [Online] Available at: http://xibo.org.uk/

Surguy, M., s.d. *Laravel: my first framework. Chapter 4 – Views.* [Online] Available at: http://maxoffsky.com/code-blog/laravel-first-framework-chapter-4-views/ [Acedido em 18 5 2105].

Watson, D. C., s.d. *About.* [Online] Available at: http://www.neontsunami.com/about

Watson, D. C., s.d. *autologin.* [Online] Available at: http://www.neontsunami.com/projects/autologin [Acedido em 2015].

Wikipédia, 2014. *Digital Signage.* [Online] Available at: https://pt.wikipedia.org/wiki/Digital_Signage [Acedido em 2015].

Wikipédia, 2015. *CodeIgnater.* [Online] Available at: https://pt.wikipedia.org/wiki/CodeIgniter [Acedido em 19 7 2016].

Wikipédia, 2015. *Digital Place-based Advertising Association.* [Online] Available at: http://en.wikipedia.org/wiki/Digital_Place-based_Advertising_Association [Acedido em 1 6 2015].

Wikipédia, 2016. *CakePHP.* [Online] Available at: https://pt.wikipedia.org/wiki/CakePHP [Acedido em 19 7 2016].

Wikipédia, 2016. *Laravel.* [Online] Available at: https://pt.wikipedia.org/wiki/Laravel [Acedido em 19 7 2016].