IPG Politécnico da Guarda
Polytechnic of Guarda

# RELATÓRIO DE PROJETO

Licenciatura em Engenharia Informática

Nuno Miguel Rosa Pinto

dezembro | 2015

Instituto Politécnico da Guarda

Escola Superior de Tecnologia e Gestão

# Website sobre música

Nuno Miguel Rosa Pinto

n.º1010287

Projeto de Informática em contexto de estágio do curso

Engenharia Informática

01 de Dezembro de 2015


# Website sobre música

Nuno Miguel Rosa Pinto

n.º1010287

Projeto de Informática em contexto de estágio do curso Engenharia Informática

Supervisor: António Matias Gil, Gerente da empresa Dom Digital

Orientador: José Quitério Figueiredo

01 de Dezembro de 2015

# 1 Elementos Identificativos

**Aluno**

Nome: Nuno Miguel Rosa Pinto

Numero: 1010287

Curso: Engenharia Informática

**Estabelecimento de Ensino**

Escola Superior de Tecnologia e Gestão – Instituto Politécnico

Morada: Av. Dr. Francisco Sá Carneiro 50, 6300-559 Guarda

Telefone: 271220120 | Fax: 271220150

**Duração do Estágio**

**Inicio:** 15 de Junho de 2015

**Fim:** 10 de Setembro de 2015

**Orientador do Projeto**

**Nome:** José Quitério Figueiredo

**Grau académico:** Mestre

**Empresa onde realizei estágio**

**Nome:** Dom Digital

**Morada:** Av Rainha D Amelia, 142 cv, Guarda.

**Supervisor:** António Matias Gil

# 2 Agradecimentos

Gostaria de agradecer à Empresa Dom Digital por me propor e ter dado a oportunidade de poder fazer este projeto.

Ao Professor José Quitério Figueiredo por ter aceitado ser o meu orientador neste projeto, foi sem dúvida uma mais-valia para mim pois o seu conhecimento ajudou-me muito para tornar este projeto uma realidade.

Gostaria de agradecer a todos os colaboradores da Empresa Dom Digital pelo apoio e disponibilidade em ajudar na realização do projeto.

Gostaria, também, de agradecer a todos os docentes não referidos pelo apoio e disponibilidade em ajudar ao longo de todo o curso, que de alguma forma contribuíram para a realização do mesmo.

# 3 Resumo

No âmbito da Licenciatura em Engenharia Informática, foi realizado o estágio na Empresa Dom Digital, na cidade da Guarda.

O estágio decorreu no período compreendido entre 15 de Junho a 10 de Setembro de 2015.

Nesta Empresa, foi realizado um projeto de um website sobre música. Neste website é possível ouvir musica, ver notícias sobre cantores e ver videoclipes.

Neste relatório procura-se descrever as ferramentas e tecnologias utilizadas, como: *frameworks* para HTML/CSS, *frameworks* para JavaScript e PaaS.

Na conclusão é feita uma reflexão sobre esta experiência, destacando o que foi aprendido.

# 4 Abstract

Under the course of Computer Engineering, there was performed an internship in the Dom Digital company, in the city of Guarda.

This internship lasted from the 15th of June to the 10th of September of 2015, with a length of 280 hours.

On this company, there was realized a project about a music website. In this website it is possible to listen to music, read the news about singers and watch videoclips.

In this document there are described the tools and technologies used, like, HTML/CSS *frameworks* and JavaScript and PaaS *frameworks*.

Under the conclusion there are made some reflections about what this internship experience, highlighting what was learned.

# Índice geral

# Índice de figuras

# Índice de Tabelas

# Glossário

**API** – em Inglês Application Programming Interface é composta por uma série de funções por detrás de um website ou aplicação que conecta com outros sistemas ou até mesmo aplicações.

**Base de dados** – Conjunto de dados estruturados e relacionados entre si.

**HTML** – Hiper Text Markup Language, é uma linguagem de marcação para produção de páginas Web.

**CSS** – em Inglês Cascading Style Sheets é uma linguagem de estilo utilizada para definir a apresentação de documentos escritos por exemplo HTML. Tem como objetivo separar o formato do conteúdo.

**Javascript** – é uma das linguagens de programação de script mais usadas do lado do cliente em browsers. Muito útil em projetos para websites pois permite adicionar funcionalidades do lado do cliente.

**Website** - é uma palavra que resulta da justaposição das palavras inglesas web (rede) e site (sítio, lugar). No contexto das comunicações eletrônicas, website e site possuem o mesmo significado e são utilizadas para fazer referência a uma página ou a um agrupamento de páginas relacionadas entre si, acessíveis na internet através de um determinado endereço. No Português Europeu é também comum utilizar o termo sítio da internet ou sítio eletrônico.

**Browser** – pode-se dizer também que web browser é responsável pela comunicação com os servidores, é ele que processa os dados recebidos pelos servidores da Internet e processa as respostas.

**PaaS** – em Inglês Platform as a service é um plataforma de computação em nuvem que fornece uma plataforma que permite aos clientes desenvolver, executar e gerenciar aplicativos da Web sem a complexidade de construção e manutenção da infraestrutura, tipicamente associada com o desenvolvimento e lançamento de uma aplicação.

***Framework*** - une vários códigos comuns entre vários projetos de software tornando-a numa funcionalidade genérica. Pode atingir uma funcionalidade específica, por configuração, consoante a necessidade. Ao contrário das bibliotecas, é o *framework* quem dita o fluxo de controlo da aplicação.

**Computação em nuvem** - ou em (em inglês, cloud computing) refere-se à utilização da memória, das capacidades de armazenamento, cálculo de computadores e servidores compartilhados e interligados por meio da Internet, seguindo o princípio da

computação em grade. O armazenamento de dados é feito em serviços que podem ser acedidos de qualquer lugar do mundo, a qualquer hora, não havendo necessidade de instalação de programas ou de armazenar dados e onde se pode ter acesso a informações, arquivos e programas num sistema único, independente de plataforma.

**URL** - é uma sigla (anglicismo da tecnologia da informação) para as palavras em inglês "Uniform Resource Locator", (traduzido para a língua portuguesa, Localizador Padrão de Recursos). Um URL refere-se ao endereço de rede no qual se encontra algum recurso informático, como por exemplo um arquivo de computador ou um dispositivo periférico (impressora, equipamento multifuncional, unidade de rede etc.), essa rede pode ser a Internet.

**Servidor FTP** – é um servidor que fornece, através de uma rede de computadores, um serviço de acesso a um disco rígido ou servidor de ficheiros para utilizadores através do protocolo de transferência de arquivos: File Transfer Protocol.

# 5 Introdução

A evolução da tecnologia permitiu que a informática fosse cada vez mais importante nos dias de hoje.

Hoje em dia existem diversos PaaS específicos com bases de dados já feitas onde as pessoas só têm o trabalho de introduzir os dados que desejam, sem terem que fazer todo o trabalho complexo que está por detrás disso.

Para ligação do PaaS ao meu website existe a solução de APIs que facilitam e agilizam o trabalho do programador.

A cada dia que passa há mais pessoas a criarem *frameworks*, que facilitam o trabalho, de quem deseja programar mais facilmente e ter resultados mas rápidos.

Neste projeto em contexto de estágio foram utilizadas todas essas novas tecnologias, e assim aprender novas técnicas e utilizar novas ferramentas, porque é algo em expansão nos dias de hoje.

## 5.1 Estrutura do documento

O documento compreende cinco capítulos, para além da introdução e de um capítulo de conclusões.

No segundo capítulo é apresentado o estado da arte, onde se fará referência a alguns *frameworks* já existentes no mercado e será apresentada uma comparação dos mesmos.

No terceiro capítulo é descrita a metodologia a seguir e a descrição das tarefas que foram seguidas e todo o processo de desenvolvimento do website.

No quarto capítulo é descrita a análise pormenorizada das tecnologias necessárias para o desenvolvimento do website.

No quinto capítulo descreve-se a implementação da tecnologia utilizada com algumas capturas de ecrã das ferramentas usadas.

No sexto capítulo, vai ser apresentada uma avaliação do website.

Finalmente, no sexto capítulo, são apresentadas as conclusões.

## 5.2 Motivação

A principal motivação para o desenvolvimento deste projeto em contexto de estágio é a possibilidade de conhecer o trabalho realizado numa empresa.

Saber como é o realização dos projetos a distribuição de tarefas e a organização de uma empresa, para quando entrar no mundo do trabalho estar mais preparado.

Outro fator foi aprender uma nova linguagem que não lecionada durante o curso.

O curso ajudou a ter os conhecimentos básicos da programação, não uma unidade curricular mas sim todas porque a diversidade de linguagens aprendidas, permitiu que conseguisse aprender outra sem estar dependente de uma em específico.

Pelos motivos descritos, ficamos logo ligados à proposta e foi mais um incentivo em estudá-la. Gostar do que fazemos é a melhor forma de fazer o trabalho com qualidade e profissional.

Este projeto obrigou a pesquisar e a pôr em prática muitos dos conhecimentos adquiridos ao longo do curso e acima de tudo criar um website atrativo e acessível para qualquer utilizador.

Estes foram os fatores que influenciaram na escolha e desenvolvimento deste projeto.

## 5.3 Solução

A solução encontrada para a proposta que foi feita, de acordo com os requisitos definidos foi a criação de um website sobre música. Este projeto será desenvolvido em JavaScript com a *framework backbone.js* e HTML5. Para escrever o código foi utilizado um editor de texto e para controlo de versões vai ser utilizado o Bitbucket.

Também vai ser utilizo o website ardina.press para inserir conteúdo na base de dados e a ardina.api para fazer conexão do ardina.press à *framework backbone.js*.

O website vai ser alojado no servidor do IPG.

## 5.4 Contribuição

A contribuição principal deste projeto é o desenvolvimento, implementação e teste de um website, utilizando as tecnologias descritas, tornando acessível ao público em geral que queira ouvir musica, ver videoclips e visualizar noticias sobre os mais variados artistas.

## 5.5 Contextualizar o trabalho de projeto/estágio

Com a crescente popularidade da Internet, é quase inevitável que toda a empresa ou negócio tenha um website. Não se trata apenas de uma moda, mas sim uma consciência cada vez maior das vantagens de usar um website como método de publicidade e negócio.

E assim o objetivo deste projeto em contexto de estágio consiste essencialmente na elaboração de um website sobre música que permita aos seus utilizadores, em ambiente web, ouvir música, visualizar notícias, visualizar videoclips ou outra informação do interesse dos utilizadores em qualquer plataforma.

O desenvolvimento do website terá como base os serviços ardina.com e ardina.api e ardina.press.

Ardina.api e ardina.press faz parte da família ardina.com que é produzida com base force.com, a plataforma líder de desenvolvimento de aplicações empresariais na *cloud* que permite obter conteúdo de forma dinâmica sem que se tenha que criar uma base de dados e coloca-la num servidor.

## 5.6 Objetivos previstos

Os objetivos a alcançar com o desenvolvimento deste projeto são:

- Carregar conteúdos no ardina.press que serve como base de dados.
- Implementar e manter um website responsivo.
- Utilizar *framework Backbone.js* que permite a ligação aos programas da família ardina.com, sobretudo ardina.api.
- Gestão do *Backend*, através da ligação da ardina.api ao ardina.press onde a base de dados está guardada na Salesforce.

## 5.7 Mapa de Gantt

A seguir é apresentado mapa de gantt com as tarefas que vão ser realizadas.

O único imprevisto que ouve na atribuição de tarefas foi a mudança de tema.

Primeiro era para realizar um website para uma rádio, mas como não foi possível, chegou-se a acordo de se realizar um website sobre música.

| ID | Nome da Tarefa | Início | Conclusão | Duração |
|---|---|---|---|---|
| 1 | Tutorial HTML/CSS | 15-06-2015 | 22-06-2015 | 6d |
| 2 | Tutorial de JavaScript | 23-06-2015 | 06-07-2015 | 10d |
| 3 | Carregar conteúdo no Ardina.press | 07-07-2015 | 13-07-2015 | 5d |
| 4 | Exemplo de website Em Backbone.js | 14-07-2015 | 27-07-2015 | 10d |
| 5 | Divisão do HTML | 27-07-2015 | 03-08-2015 | 6d |
| 6 | Construção do website em Backbone.js | 03-08-2015 | 10-09-2015 | 29d |
| 7 | Realização do relatório | 10-09-2015 | 29-10-2015 | 36d |


*Figura 1 - Mapa de Gantt*

## 5.8 Metodologia a usar

A metodologia que vai ser utilizada no projeto vai ser a metodologia Scrum.

Scrum é um modelo de desenvolvimento ágil para a gestão de projetos de *software*. Um dos principais objetivos é realizar um produto rapidamente com a maior qualidade possível. Este modelo fornece métodos de planeamento bem específicos do que cada pessoa na empresa tem para fazer.

Esta metodologia funciona da seguinte forma:

1. O produto é definido. Quais os requisitos? O que o cliente pretende?
2. O cliente especifica as funcionalidades do produto (Product Backlog).
3. Elege-se uma pessoa responsável pelo projeto que em conjunto com o cliente este definem os *sprints*.
4. O Product Backlog é dividido em vários *sprints*, que devem ser realizados conforme as prioridades que o cliente determina. É definido um tempo de trabalho para cada *sprint*. Ao fim de cada semana é mostrado o que foi feito no trabalho.
5. Os sprints vão sendo efetuados até o Product Backlog acabar e o cliente disser que foi obtido o produto final (1).

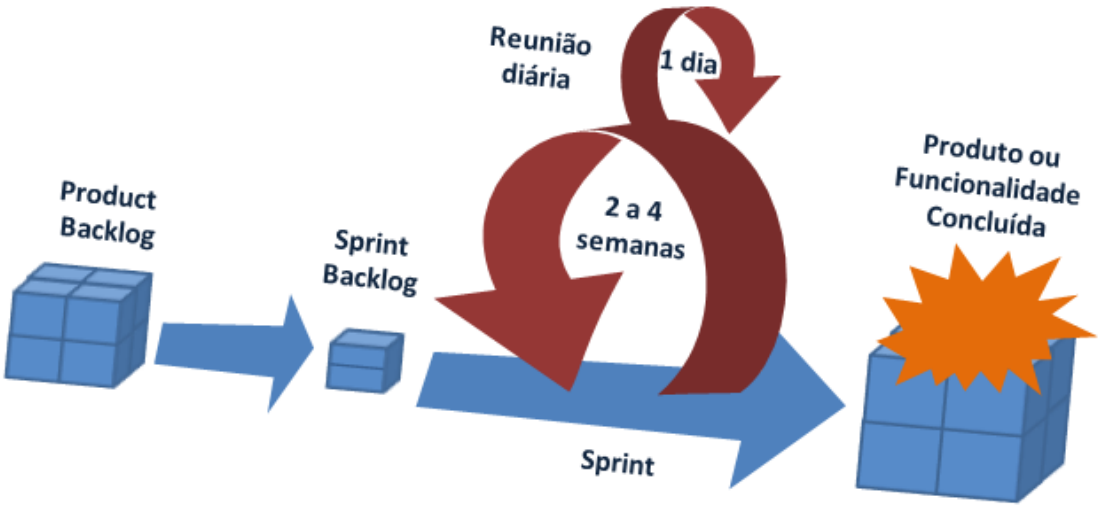


*Figura 2 - Diagrama da metodologia scrum (2).*

# 6 Estado da arte

Neste capítulo vão ser referidas as tecnologias necessárias para o projeto.

Vai ser feito um resumo da *framework* HTML/CSS, *framework* JavaScript e de PaaS que vão ser utilizadas e alguns exemplos de outras existentes no mercado.

Existem muitas *frameworks* para a criação de websites em JavaScript e muitas outras para se construir em HTML/CSS, todas com o mesmo princípio: o de facilitar o desenvolvimento, com o mínimo de programação possível.

Mas como na empresa onde foi realizado o estágio já utilizavam a *framework backbone.js* já há alguns anos, achou-se interessante e enriquecedor aprender uma nova ferramenta.

Para a base de dados foi proposto utilizar o ardina.press e a *framework* de HTML/CSS, ambas desenvolvidas pela empresa para lhes facilitar a criação de websites no futuro. O ardina.press permite a gestão de conteúdos multimédia de forma rápida e intui-ta.

A vantagem da *framework* HTML/CSS é que permite moldar o design do website conforme o cliente deseja, alterando as classes já existentes na *framework* e os seus parâmetros.

## 6.1 PaaS

**PaaS** (Platform as a Service) é um modelo de computação em nuvem que oferece um conjunto de ferramentas através da internet. Nesse conjunto de ferramentas do qual inclui sistema operativo, ambiente de desenvolvimento para desenvolvimento de programação, base de dados, servidores web, etc.

**O conceito de computação em *cloud*** refere-se à utilização da memória e das capacidades de armazenamento e cálculo de computadores e servidores compartilhados e interligados por meio da Internet, seguindo o princípio da computação distribuída o qual se chama *Grid Computing*.

O armazenamento de dados é feito em serviços que poderão ser acedidos de qualquer lugar do mundo, a qualquer hora, não havendo necessidade de instalação de programas ou de armazenar dados. O acesso a programas, serviços e arquivos é remoto, através da Internet. O uso desse modelo (ambiente) é mais viável do que o uso de unidades físicas, independente de plataforma (3).


*Figura 3 - Computação em nuvem (3).*

## 6.2 MVC

Model-View-Controller (MVC), é um padrão de desenvolvimento de *software*, que separa a informação da interface com a qual o utilizador interage com a interação do utilizador.

As Views são responsáveis pela interface que mostra todo o conteúdo que o utilizador visualiza como por exemplo o HTML.

Os Models fazem o controlo da informação, isto quer dizer que é tudo o que diz respeito à escrita, validação e leitura do conteúdo.

Os Controllers fazem a junção das Views com os Models como por exemplo mostrar uma View e ocultar outra.

Um diagrama simples exemplificando a relação entre Model, View e Controller, é apresentado na figura 3, as linhas sólidas indicam associação direta e as tracejadas indicam associação indireta (4).

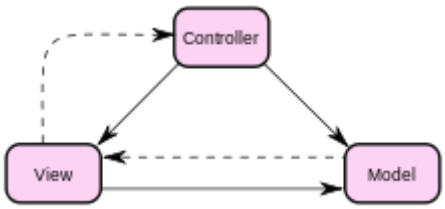


*Figura 4 - Diagrama do MVC (4).*

## 6.3 Salesforce

A plataforma *Salesforce* é um sistema *Customer Relationship Management* (CRM) o que significa Gestão de Relacionamento com o Cliente. Este tipo de *software* é utilizado para facilitar o relacionamento de uma empresa com os seus clientes, gerindo os processos de vendas e pós-venda de uma forma prática e simples.

O *software* de CRM tradicional cliente/servidor é uma espécie em extinção. O mercado de *software* de CRM mudou de um modelo de propriedade para aluguer, libertando as empresas das dificuldades e despesas de aquisição, implementação e manutenção do software. Graças a essa mudança foi possível a criação de computação em *cloud* é uma forma de desenvolver *software* mais atrativo de negócios pela Internet onde podemos reduzir custos porque só se paga para usar o software, em vez de comprá-lo e assim permite utilizar uma plataforma para vários clientes com uma infraestrutura única e comum para todos os clientes.

O CRM permite otimizar e automatizar processos de negócios, oferecer a todos os trabalhadores da empresa uma visão completa do cliente, fornece análises e perceções mais profundas das vendas críticas e da métrica de clientes e permite ainda manter todos concentrados na aquisição de novos clientes enquanto mantêm os atuais satisfeitos.

**A plataforma Salesforce é constituída pelos seguintes módulos:**

- *Sales* Cloud - plataforma de vendas
- Serviço na cloud (*Service Cloud*) - O futuro do serviço ao cliente
- ExactTarget Marketing Cloud - correio eletrónico, dispositivos móveis, redes sociais e produtos
- Salesforce1 Platform - Novas APIs e ferramentas tornam a criação de aplicações mais rápida
- Salesforce Communities - Conecta-se com clientes, parceiros e funcionários
- Chatter - Colabore em tempo real com toda sua empresa
- Desk.com - Suporte ao cliente integrado para pequenas empresas (5)

**A Force.com** é uma plataforma para criar e implementar uma aplicação em *cloud* de última geração. Não sendo necessário gerir ou comprar servidores ou *software* o utilizador pode concentrar-se em desenvolver aplicações num serviço seguro e eficiente.

Uma empresa e os seus utilizadores podem ter acesso ao Salesforce através de duas interfaces: o website completo do Salesforce, conectando-se através de um computador ou por um aplicação móvel.

Force.com é uma plataforma com serviço (PaaS), que permite aos desenvolvedores criar aplicações que integram a principal aplicação do Salesforce.com.

Force.com estão hospedados na infraestrutura do Salesforce.com e as aplicações do Force.com são construídos usando Apex.

**Salesforce1**

A Plataforma Salesforce1 reúne Salesforce.com, a Force.com, Heroku, e ExactTarget em uma família de serviços em *cloud*.

**Work.com**

É uma plataforma de gestão de desempenho social que permite melhorar o desempenho do trabalho através de treinamento contínuo, permite ter acesso ao feedback em tempo real (6).

**Aplicações em base de dados**

A plataforma é baseada numa base de dados com informações estruturadas e consistentes.

**Aplicações colaborativas**

A plataforma permite ser acedida por vários utilizadores ao mesmo tempo podendo ser criadas assim aplicações colaborativas com dados e serviços compartilhados por vários utilizadores em diferentes locais. Estas aplicações podem ser acedidas a partir de qualquer lugar do mundo usando somente um navegador da Web, facilitando atividades e trabalho em equipa.

## As tecnologias por trás de uma aplicação da plataforma Force.com

### Arquitetura de multilocatário

Esta arquitetura permite que todos os utilizadores partilhem a mesma infraestrutura e a mesma versão da plataforma Force.com. Nas versões de locatários únicos, cliente-servidor ou servidores de correio eletrónico, disponibiliza as atualizações automaticamente e simultaneamente para todos os utilizadores. Não sendo necessário comprar equipamentos e preocupar-se em instalar as últimas atualizações.

### Modelo de Desenvolvimento orientado por objetos

A plataforma Force.com usa um modelo de desenvolvimento orientado por objetos que ajuda numa maior produtividade e na criação das aplicações. Guias, formulários e *links* são definidos como objetos numa base de dados, em vez de possuírem códigos embutidos em forma de uma linguagem de programação.

Nesta plataforma os programadores trabalham num nível de abstração muito mais alto do que trabalhariam se usassem Java ou C# graças ao desenvolvimento orientado por objetos, podendo assim aproveitar recursos avançados que a plataforma fornece como padrão (5).

## 6.4 Exemplos de outras PaaS já existentes

**AWS Elastic Beanstalk** utiliza-se para a implementação e dimensionamento de aplicações e serviços Web desenvolvidos em Java.NET, PHP, Node.js, Python, Ruby, e Docker Go em servidores como: o Apache, o Nginx, o Passageiro e o IIS na AWS *cloud* (7). A Elastic Beanstalk, permite implementar e gerir aplicações mais rapidamente através da AWS Cloud sem ter-mos que nos preocupar com a infraestrutura que executa as aplicações.

Com a AWS Elastic Beanstalk permite-nos reduzir a complexidade de gerenciamento. Basta carregar a aplicação e Elastic Beanstalk faz automaticamente o dimensionamento e monitorização da aplicação, mantendo assim o controlo sobre os recursos da AWS, permitindo desta forma ver os recursos subjacentes a qualquer momento (8).

**Heroku** é uma plataforma de serviço em *cloud* (PaaS) suportando várias linguagens de programação. Pertence à Salesforce.com que foi a pioneira na prestação de serviços no modelo PaaS. Estando em desenvolvimento desde junho de 2007, na altura só suportava a linguagem de programação Ruby, entretanto adicionou suporte para Java, Node.js, Scala, Clojure, Python, Go e PHP. Passando o sistema operativo de base Debian para Ubuntu.

O Heroku é um Paas (Platform as a Service) que corre sobre o Amazon EC2 (IaaS) permitindo aos utilizadores desenvolver as suas aplicações numa plataforma *cloud*, sem necessidade de instalar ou gerir o *hardware* e *software* inerente a estas (9).

**Google App Engine** é uma plataforma de oferta de serviço (PaaS) que permite criar e executar aplicações na infraestrutura do Google. Sendo fácil de construir, de manter, de configurar o tráfego e armazenamento de dados, têm integração com outros serviços em *cloud* do Google e APIs e armazenamento inteligente com consultas, classificações e transações.

Com o Google App Engine, basta simplesmente carregar a aplicação para a *cloud* e ela permite gerir diferentes versões da nossa aplicação sem ter-mos servidores para gerir.

O SDK gere a nossa aplicação localmente, enquanto o Developers Console Google gere a aplicação na produção. A Developers Console utiliza uma interface

baseada na Web que permite criar novas aplicações, configurar nomes de domínio, alterar a versão da aplicação e examinar o acesso e mensagens de erro.

É num ambiente seguro, numa área restrita que a App Engine, que é possível distribuir a aplicação por vários servidores, sendo independente do *hardware*, do sistema operativo, ou da localização física do servidor.

Google App Engine suporta aplicações escritas em: Java Runtime Environment, Python, PHP e Go (10).

### 6.4.1 Comparação

| | Amazon Beanstalk | Google App Engine | Heroku for Java | Force.com |
|---|---|---|---|---|
| Tomcat | Sim | Não | Não | Não |
| Java SE | Sim | Não | Sim | Sim |
| Java EE | Não | Não | Não | Sim |
| Suporte de bibliotecas Java | Sim | Não | Sim | Sim |
| Acesso ao sistema de ficheiros | Sim | Não | Sim | Sim |
| Acesso a threads | Sim | Não | Sim | Sim |
| Conexões a redes externas | Sim | Limitado | Sim | Sim |
| MySQL | RDS | Plano pago | Sim | Sim |
| Base de dados | RDS | Não | Externo | Externo |
| Suporte a Big Data | SimpleDB | BigTable | Externo | Externo |
| Dependências extras | Sim | Não | Sim | Sim |
| Compatiblidade com outras aplicações | Sim | Não | Não | Sim |
| Portabilidade | Alta | Baixa | Baixa | Baixa |
| Pronto para produção? | Sim | Sim | Beta | Sim |

*Tabela 1 - Comparação das várias Paas (11).*

## 6.5 JavaScript

JavaScript pode ser utilizada com vários modelos de programação, deixando ao cargo do programador a forma como programa. Como a estrutura em HTML da página pode ser facilmente manipulada permite que a estrutura da página interfira diretamente conforme o funcionamento do código.

A arquitetura de programas não-estruturados em JavaScript envolve a página em HTML, contendo eventos que acionam a lógica da aplicação, respondendo a eventos e alterando os dados da estrutura em HTML.

JavaScript não é mais do que uma simples linguagem de script, onde as suas implementações podem envolver aplicações inteiras com diversas funcionalidades, ou até mesmo projetos inteiros.

## 6.6 *Frameworks* já existentes para JavaScript

**Backbone.js**

Backbone.JS cria uma solução para o problema de falta de estrutura das aplicações JavaScript, separando características comuns de qualquer aplicação, como por exemplo: resposta a eventos, armazenamento de dados, e mais uma gama de outras funcionalidades.

É uma *Framework* JavaScript inspirada em uma interface JSON RESTful que oferece uma arquitetura MV* ou MVC: Model–View-Controller, sendo famosa pela sua leveza e robustez, a sua única dependência é o Underscore.js, que por sua vez é dependente da biblioteca jQuery para interações com o DOM.

**Knockout.js**

Knockout é uma biblioteca de JavaScript que o ajuda a criar websites mais acessíveis, mais limpos em termos de visualização e que permite tornar o website responsivo.

Permite ainda que o utilizador tenha atualizações dinâmicas do conteúdo, podendo ser implementadas de uma maneira mais simples e fácil com o Knockout (12).

**Angular**

AngularJS é um *framework* JavaScript open-source, mantido pelo Google, que auxilia na execução de single-page applications. O seu objetivo é aumentar aplicações que podem ser acedidas por um navegador web, sob o padrão Model–View–Controller (MVC), facilitando o desenvolvimento e o teste das aplicações.

O HTML é lido pela biblioteca, que contém tags especiais que executa a diretiva na qual esta tag pertence, e faz a ligação entre a apresentação e o seu modelo, representado por variáveis JavaScript comuns. O valor dessas variáveis JavaScript podem ser acedidas manualmente, ou via um recurso JSON estático ou dinâmico (13).

**JavaScriptMVC**

O JavaScriptMVC é uma *framework* para o lado do cliente(cliente-side) no desenvolvimento em JavaScript, sendo uma das melhores maneiras de se criar aplicações com qualidade e fáceis de manusear num curto espaço de tempo (12).

### 6.6.1 Comparação

| | **Backbone** | **Knockout** | **JavaScriptMVC** | **Angular** |
|---|---|---|---|---|
| **Tamanho** | 6.8KB | 15.7KB | 7.14KB | 33KB |
| **Popularidade** | 23,000+ stars no GitHub | 6,000+ stars no GitHub | 0 stars no GitHub | 43,000+ stars no GitHub |
| **UI ligações** | Não | Sim | Não | Sim |
| **Visualizações compostas** | Não | Não | Sim | Não |
| **Documentação** | Sim | Sim | Sim | Sim |
| **Web Camada de Apresentação** | Sim | Sim | Sim | Sim |
| **Funciona com outras** | Sim | Sim | Sim | Sim |

*Tabela 2 - Comparação das várias frameworks JavaScript* (14) (15)

## 6.7 Exemplo de *framework* já existentes para HTML/CSS

**Bootstrap**

Bootstrap é uma coleção de vários elementos e funções personalizáveis para projetos web, acondicionados previamente numa única ferramenta. Os desenvolvedores podem escolher que elementos usar para projetar um website com o Bootstrap, podendo ter a certeza de que os elementos escolhidos não entram em conflitos entre si, cada elemento encaixa-se perfeitamente com os outros. A estrutura do Bootstrap pode ser facilmente personalizável, utilizando uma combinação de HTML, CSS e JavaScript. Bootstrap tem diversas funções, com 100% de capacidade de resposta móvel e várias opções de plug-in jQuery com muitos recursos (16).

***Framework* feita pela empresa**

Tem vários elementos e funções personalizáveis, que permite que os desenvolvedores possam escolher os vários elementos que desejam utilizar.

Disponibiliza várias funções que permite que o website se ajuste as diversas plataformas.

A estrutura pode ser facilmente personalizável, utilizando uma combinação de HTML, CSS e JavaScript.

**Foundation**

Foundation é uma *framework* CSS construída com Sass, um pré-processador CSS, que nos permite desenvolver muito mais rápido o nosso HTML/CSS e disponibiliza ferramentas para personalizar e construir em cima de estilos iniciais.

Com o Foundation permite escrever e organizar o código CSS para manter facilmente o website ao longo do tempo sem as dores de cabeça típicas. Também a funcionalidade de plug-ins JavaScript que fazem interações úteis e mais fáceis de implementar em diferentes resoluções do ecrã (17).

**HTML5 Boilerplate**

O Boilerplate é um *template* em html5 rápido, completo e adaptável. É a ferramenta ideal para o início de um website ou webapp, permitindo poupar tempo na construção da estrutura básica do código e na organização dos ficheiros.

Entre as funcionalidades do Boilerpalce destacam-se: os Normalizer, os Media queries e Print styles, os Jquery e Modernizr, o Google Analitycs e os Htaccess, icons, robots.

**Normalizer** é um CSS usado para manter a consistência de todos os elementos dos navegadores, não tendo que alterar os elementos com estilos diferentes para cada navegador.

**Media queries e Print styles** ajudam na visualização e impressão dos elementos em dispositivos diferentes.

**Jquery e Modernizr** são dois scripts do template essenciais para projetos web. O já bem conhecido Jquery que facilita a manipulação do JavaScript do projeto. O Modernizr é essencial para a deteção dos recursos dos browsers.

**Google Analitycs** já vem adicionado ao template, sendo necessário apenas trocar a ID da nossa conta do analitycs para que o GA funcione.

**Htaccess, icons, robot** e outros recursos prontos podem ser adaptados às necessidades do projeto (18).

### 6.7.1 Comparação

| | Foundation | HTML5 Boilerplate | Bootstrap | Feita pela empresa |
|---|---|---|---|---|
| Tamanho | 326KB | 160KB | 145KB | 20KB |
| Popularidade | 18,000+ stars no GitHub | 31,000+ stars no GitHub | 75,000+ stars no GitHub | - |
| Responsivo | Sim | Sim | Sim | Sim |
| Modular | Sim | Sim | Sim | Sim |
| Documentação | Sim | Não | Sim | Sim |
| Template inicial | Sim | Sim | Sim | |
| Licença | MIT | MIT | MIT | - |
| Multiplataformas | Sim | Sim | Sim | Sim |
| plug-ins | Sim | Sim | Sim | Sim |
| Browsers que suportam | Chrome, Firefox, Safari, IE9+; iOS, Android, Windows Phone 7+ | Firefox, Chrome, Safari, IE8 ou superior | Firefox, Chrome, Safari, IE8 ou superior | Firefox, Chrome, Safari, IE8 ou superior |

*Tabela 3 - Comparação das várias frameworks HTML/CSS* (19) (20).

## 6.8 API

O modelo orientado por objetos permite construir rapidamente diversas funcionalidades com as ferramentas fornecidas pela plataforma.

Os programadores por vezes necessitam de modificar dados riais numa aplicação e usar serviços de terceiros, para criar comportamentos personalizados para as aplicações.

Para o fazer, eles precisam de usar diversas APIs para efetuar uma integração com a plataforma.

**Apex**

A Apex é a primeira linguagem de programação para computação em *cloud* do mundo introduzida pela salesforce.com, cuja sintaxe é semelhante ao Java para aplicações da Web, funciona nos servidores da plataforma Force.com.

A Apex foi desenvolvida com o objetivo de gestão dos dados e processos da plataforma Force.com.

Esta linguagem torna se eficiente e produtiva para criar aplicações utilizando funcionalidades lógicas predefinidas, para que os programadores se concentrem apenas nos elementos específicos das suas aplicações.

**Visualforce**

O Visualforce permite construir e fornecer qualquer tipo de interação ou até mesmo design de interface do utilizador feito inteiramente na *cloud*. A Visualforce pode melhorar a aparência da plataforma Force.com ou substituí-la por um estilo completamente único, permite ainda aos programadores usarem marcas do Visualforce juntamente com HTML, JavaScript, Flash, ou quaisquer outros códigos que funcionam em páginas HTML na plataforma.

A base de qualquer boa aplicação empresarial é uma boa interface com o utilizador.

**O diretório AppExchange**

O AppExchange diferencia a plataforma Force.com de outras plataformas, sendo um diretório da Web, no qual aplicações construídas na plataforma Force.com estão disponíveis para navegação, demonstração, revisão e instalação para clientes do salesforce.com. Caso os programadores queiram compartilhá-los com a comunidade podem enviar as suas aplicações para listagem no diretório AppExchange.

**Vantagens**

- As APIs abertas permitem facilmente integração em qualquer aplicação, simplificando o desenvolvimento das mesmas.
- Force.com requer o mínimo de codificação, reúne as aplicações em forma de blocos de construção usando ferramentas visuais e biblioteca de componentes. Inclui, também, componentes pré-construídos para perfis, conversas, atualizações, compartilhamento de arquivos, Organização de departamento de vendas, programar reuniões e enviar e-mails sem sair da plataforma.
- Pode organizar todo o material como apresentações, documentos e outros para acesso de toda equipa de venda (5).

## 6.9 Ardina.api

Faz parte da família ardina.com que é produzida com base na force.com, Ardina.api é um modelo de negócio e uma tecnologia que promove a inovação na sua empresa, encontrando novas formas de monitorizar os seus conteúdos.

Tendo como principais **vantagens** a criação de um ecossistema de parceiros para ajudarem a monitorizar os conteúdos. Esta ferramenta permite a uma empresa programar mais rápido e com mais segurança.

Pode-se ter total controlo dos conteúdos, incluindo a definição das regras de utilização e os níveis de acesso.

**O ardina.api é composto por dois módulos:**

**Módulo de negócio:** ajuda uma empresa a encontrar os melhores modelos de negócio para rentabilizar ao máximo os seus conteúdos através da API.

**Módulo técnico:** desenha tecnicamente a sua API, integrando-a com plataforma informática da sua empresa ou com a família de produtos ardina.com (21).

# 7 Tecnologias utilizadas

Neste capítulo é dada uma explicação mais completa das tecnologias que vão ser utilizadas.

## 7.1 Backbone.js

Backbone.js nasceu para desenvolver aplicações de uma única página, ou, em inglês, Single-Page Applications, mantendo os dados de um sistema ou aplicação sincronizados com a interface do utilizador sem que o programador tenha que fazer muito esforço.

A arquitetura da *framework* Backbone.JS é estruturada com diversas classes com objetivos específicos. As principais classes são: collection, event, View, entre outras.

### 7.1.1 Backbone.Collection

Uma coleção é um conjunto de dados ordenados. A classe Backbone.Collection representa uma coleção de Models, fornecendo diversos métodos úteis para se trabalhar.

Na classe Backbone.Collection todos os Models são mantidos em um array, definido no atributo Models.

A melhor prática para se trabalhar com os Models é utilização de métodos das classes. Pode ser utilizado no caso de ser necessário aceder diretamente a um array de Models.

### 7.1.2 Backbone.Models

É uma classe que armazena dados, sendo uma característica importante deste *framework*, onde os dados da aplicação não devem estar espalhados ao longo da estrutura HTML da página, mas sim organizados logicamente em estruturas de dados que manipulam estas informações do utilizador. Aproximando-se ao modelo MVC de engenharia de *software*.

Entre os métodos que Backbone.JS define, salientamos os que favorecem a validação dos dados quando um Model é modificado, e o suporte para uma API de persistência, através do middleware Backbone.sync.

### 7.1.3 Underscore.js

É possível utilizar 28 funções fornecidas pela biblioteca Underscore.js. Cada função tem um objetivo distinto e a lista é bem vasta, portanto será apresentada apenas uma tabela com cada função na documentação da Underscore.js, podendo obter mais informações sobre cada função individualmente.

| **Funções** | | | |
|---|---|---|---|
| chain | every | filter | find |
| First | forEach | groupBy | include |
| indexOf | initial | invoke | isEmpty |
| Last | lastIndexOf | map | max |
| Min | reduce | reduceRight | reject |
| Rest | size | some | sortBy |
| sortedIndex | shuffle | toArray | without |

*Tabela 4 - Funções da biblioteca Underscore.js*

### 7.1.4 Backbone.events

Ao longo do relatório foram apresentados diversas classes, uma das quais é a Backbone.Collection. Com ela é possível adicionar um ou mais Models a um conjunto de dados, remover Models, limpar o conjunto, sincronizar com o servidor, gravar no servidor, entre outras operações. Cada operação correrá um ou mais eventos. Para resumir, uma coleção poderá disparar os seguintes eventos: add: quando um Model for adicionado ao conjunto de Models; sync: quando a Collection sincronizar os dados com o servidor; reset: quando uma Collection for limpa; remove: quando um ou mais Models forem removidos do conjunto de dados; change: quando ocorrer alteração na Collection.

Eles auxiliam a manter o estado da View sempre atualizado, mostrando ao utilizador exatamente como estão os Models em cada momento. Apesar de no Backbone.js não existir uma maneira fácil de fazer o binding de atributos e View, trabalhar com eventos e callbacks é uma solução aceitável e que dará ao utilizador um feedback instantâneo de suas operações.

### 7.1.5 Backbone.View

Uma View representa o aspeto visual da aplicação. É, também, utilizada para executar eventos e agir de acordo com a sua especificidade (utilizando Backbone.Events). Os eventos servem na maioria das vezes, para quando se deseja alterar os dados de uma View. Para isso acontecer os eventos vão chamar as Views que atualizam completamente a parte da página onde o Model atua. É como se a parte da página onde o Model atua estivesse a ser carregada novamente, porém com os dados alterados. A vantagem desta aproximação é que elimina a complexidade do modelo convencional de arquiteturas de JavaScript, que eventualmente alteram pequenos elementos do ecrã em resposta a ações do utilizador. Resumindo, esta abordagem deixa o código de certa página mais lógico e de manutenção mais fácil, embora, se for mal programado, torna-se menos eficiente.

### 7.1.6 Backbone.Controller

Em Backbone.js o Controller cria uma ligação do URL com um evento, permitindo que a entrada do utilizador na página seja personalizada de acordo com a ligação selecionada. Este recurso é interessante para aplicações que desejam funcionalidades diferentes para o utilizador ligadas a URL. Esta funcionalidade do *framework* manifestada em URL, pode distribuir diversas funcionalidades da aplicação externamente, podendo divulgar mais facilmente, e armazenar em Favoritos, enviar por email.

### 7.1.7 Backbone.History

Armazena todas as referências de navegação do utilizador na página. Se todas as ações dos utilizadores fossem ligadas por Backbone.Controller's, o *framework* iria armazenar todas as ações executadas pelos utilizadores! Esta funcionalidade secundária é uma alternativa muito interessante para qualquer aplicação JavaScript e pode ser utilizada numa grande variedade de funcionalidades.

### 7.1.8 Backbone.Sync

Cada Model da aplicação tem um URL para sincronização, onde os dados do Model podem ser obtidos de um servidor ou mesmo de uma API para sincronização dos dados ou manutenção de dados (22).

## 7.2 Controlo de versões

O sistema de controlo de versões tem como objetivo registar alterações efetuadas durante o desenvolvimento de *software* das quais se destacam:

Controlo do histórico: possibilita a análise do histórico do desenvolvimento do projeto, como também possibilita a recuperação de versões mais antigas.

Para facilitar a organização do histórico é possível colocar comentários no envio das alterações do projeto.

Trabalho em equipa: muitos sistemas de controlo de versões permitem que diversas pessoas trabalhem no mesmo projeto, ao mesmo tempo. Alguns sistemas de versões têm um controlador de utilizadores embutido ou permitem a utilização de um outro sistema de autenticação separado sabendo assim que utilizador alterou o quê, o qual geralmente fica protegido por uma senha pessoal, ou alguma senha criada pelo administrador de sistemas minimizando assim alguns problemas com conflitos de versões.

Recuperação de versões seguras: muitos sistemas mostram em qual versão é que o projeto não tem erros que provoquem o não funcionamento da aplicação, facilitando a sua recuperação no futuro.

Ramificação de projeto: também muitos sistemas possibilitam a divisão do projeto em várias linhas de desenvolvimento, que podem ser trabalhadas paralelamente sem que entrem em conflito umas com as outras.

Comparação de versões

Em muitos casos os sistemas de controlo de versão permitem comparar versões, enviadas a qualquer momento, e saber o que foi alterado, acrescentado ou apagado e em que parte do ficheiro. O que permite saber o que cada um fez mais pormenorizadamente.

As etapas quando utilizadores trabalham ao mesmo tempo no mesmo projeto são as seguintes:

1.ª Etapa – cada programador atualiza o seu projeto

2.ª Etapa – vários programadores desenvolvem o projeto ao mesmo tempo

3.ª Etapa – quando terminam o seu trabalho submetendo o seu projeto

4.ª Etapa – quando se faz a atualização cada programador recebe uma mensagem a dizer que tem uma versão mais recente do projeto

5.ª Etapa – os programadores podem descarregar o projeto

6.ª Etapa – último passo submetemos a versão final

Por vezes quando se partilha um projeto numa rede diretamente num sistema de ficheiros simples, sem um sistema de controlo de versão, pode acontecer que alguns ficheiros fiquem bloqueados somente para escrita enquanto estão a ser utilizados por outros utilizadores. Quando não se utiliza um sistema de controlo de versão, existe muita informação pela rede, por cada modificação parcial é necessário reenviar o projeto e a probabilidade de alguns ficheiros ficarem danificados aumenta, o que não acontece com um sistema de controlo de versão, pois cada desenvolvedor possui uma cópia local do projeto, trabalhando localmente, baixando as atualizações e enviando as alterações (23).

## 7.3 Bitbucket

Neste projeto utilizei o Bitbucket para controlo de versões.

Foi escolhido este sistema de controlo de versões porque é o adotado na empresa.

O controlo de versões é muito importante porque pode-se guardar o projeto, sem correr o risco de se perder o trabalho, podendo saber sempre as alterações realizadas ao longo do projeto. Por algum motivo se ocorrer algum erro no projeto tem-se sempre a possibilidade de voltar com o projeto atrás no tempo.

Podia-se ter utilizado o Bitbucket diretamente pelo website como mostra na figura abaixo.

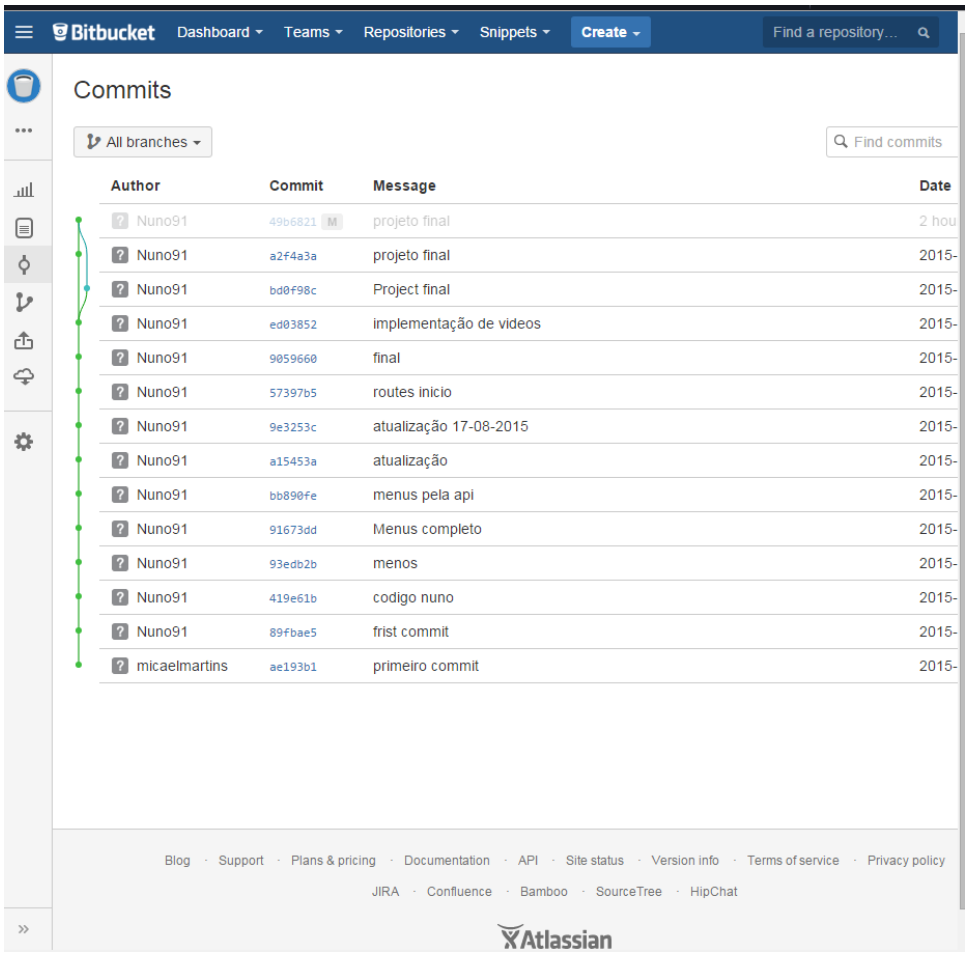


*Figura 5 - Website Bitbucket.*

Optou-se por utilizar uma interface gráfica chamada Sourcetree que permite manipular um sistema de versões á nossa escolha, tornando mais acessível, só tendo que especificar o diretório dos nossos projetos. A única coisa que se tem que fazer é atribuir um comentário para saber o que foi feito nessa versão e carregar no botão *commit* e a seguir o *push*, para carregar os ficheiros para o website do Bitbucket.

O Sourcetree mostra os ficheiros adicionados com o Símbolo verde, os que foram eliminados a símbolo vermelho e os que não foram alterados a símbolo amarelo.

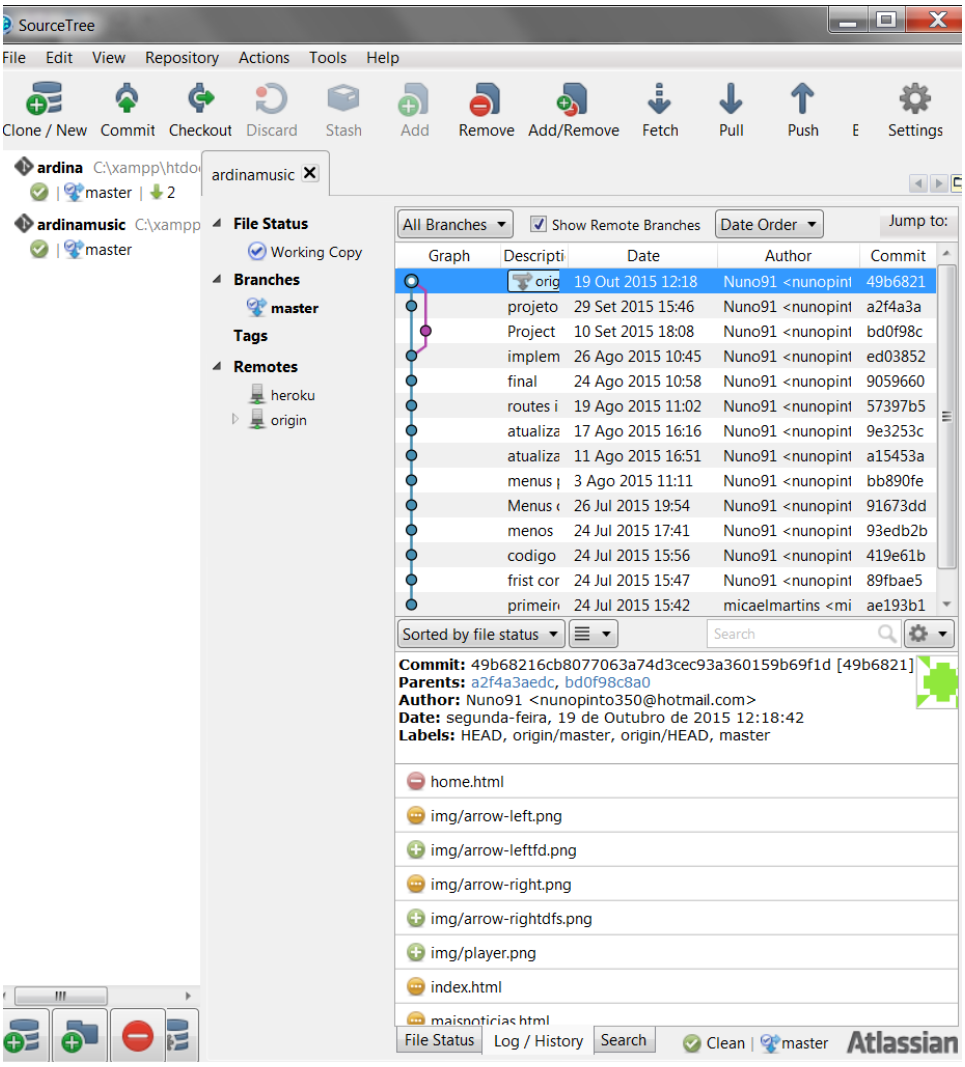


*Figura 6 - Programa Sourcetree.*

No Sourcetree as linhas de código que foram eliminadas a vermelho e as que foram adicionadas a verde. Permite, ainda, ter acesso a todas as versões, podendo fazer recuperação de versões antigas. A principal vantagem do Sourcetree é permitir ter tudo o que é necessário numa única aplicação.

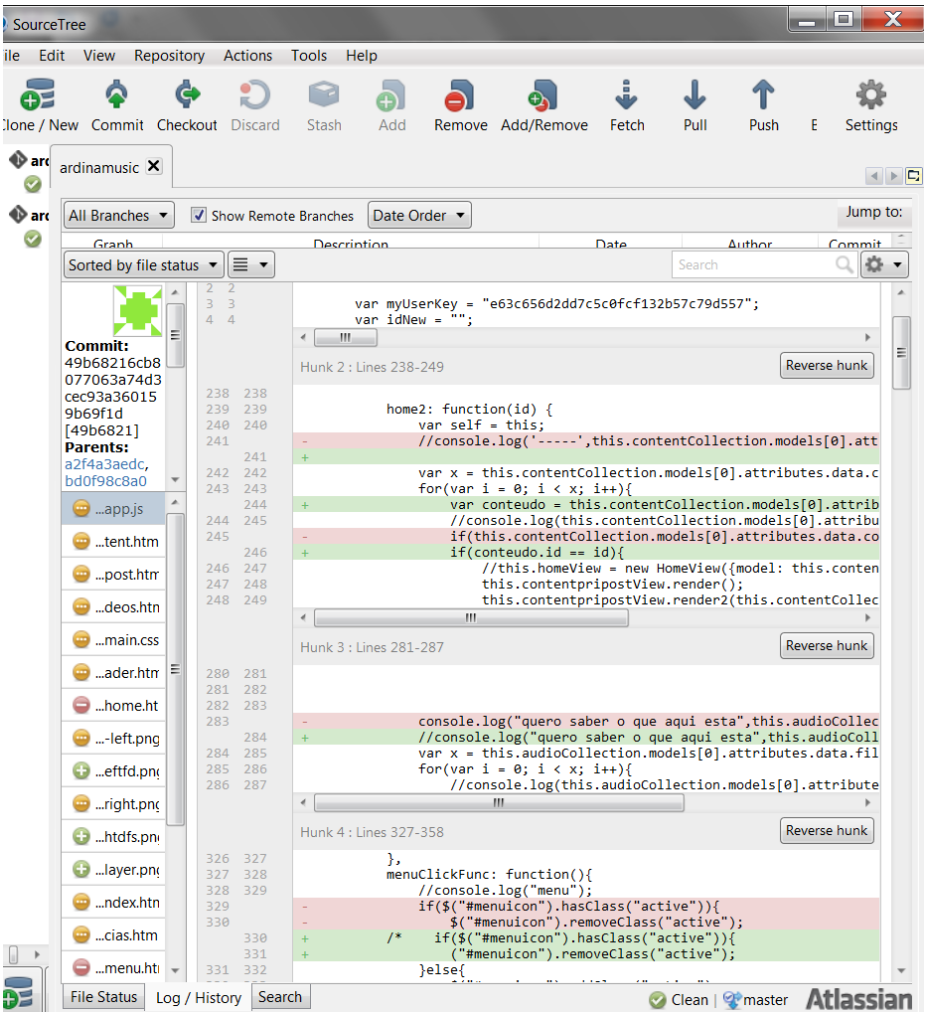


*Figura 7 - Programa Sourcetree fazendo alterações.*

## 7.4 Ardina.press

### 7.4.1 Características do ardina.press

O ardina.press faz parte da família ardina.com que é produzida com base no force.com, a plataforma de desenvolvimento de aplicações na *cloud*.

Cloud Content Management System (Gestor de conteúdos na cloud): permite criar, editar e publicar conteúdos de formato texto, ou conteúdos multimédia de uma forma rápida e sem complicações. Como funciona totalmente na *cloud*, significa que não é necessário ter os ficheiros ou pastas nos computadores em que se trabalha, porque toda a nossa informação está no mesmo sítio na internet.

Podem aceder e trabalhar nos conteúdos a qualquer momento, em qualquer lugar e de qualquer dispositivo (desktop, tablet ou smartphone), basta ter acesso à internet e um desses dispositivos.

Multimedia Asset Management (Gestão de elementos multimédia): o ardina.press tem uma base de dados completa de ficheiros multimédia organizada, o que permite pesquisar facilmente pelos ficheiros e aceder a um relatório estatístico sobre os autores e utilização dos ficheiros. Torna-se fácil carregar ficheiros multimédia e organizá-los para que possam ser fáceis de encontrar ou até mesmo reutilizar. Aceita os ficheiros em formato vídeo, áudio e imagem.

Tag Management System (Organização de artigos): possui um sistema de tags que possibilita a atribuição de uma tag aos artigos, ou aos ficheiros, permitindo desta forma encontrar o conteúdo desejado sem dificuldade.

Criação de tabelas: tem a funcionalidade de criar tabelas diretamente no ardina.press ou utilizar tabelas já criadas, as quais já têm a funcionalidade de comparação, sem necessidade de utilizar folhas de cálculo.

Social Collaboration (Rede social interna): ardina.press tem incluído uma rede social privada tão fácil de utilizar como o Facebook. Utiliza tecnologias sociais como feeds, perfis e grupos para que possa comunicar ou até mesmo partilhar os nossos conteúdos mais facilmente com colegas.

Profile management (gestão de perfis): ardina.press possibilita a criação de perfis diferentes para cada trabalhador da empresa.

Inicialmente tem perfis para jornalistas e editores com acesso a funcionalidades distintas. Mas podemos adicionar outros perfis de acordo com as necessidades de cada trabalhador (24).

A figura a seguir mostra todas as opções do ardina.press.

Existe a opção contents para inserir conteúdos na base de dados.

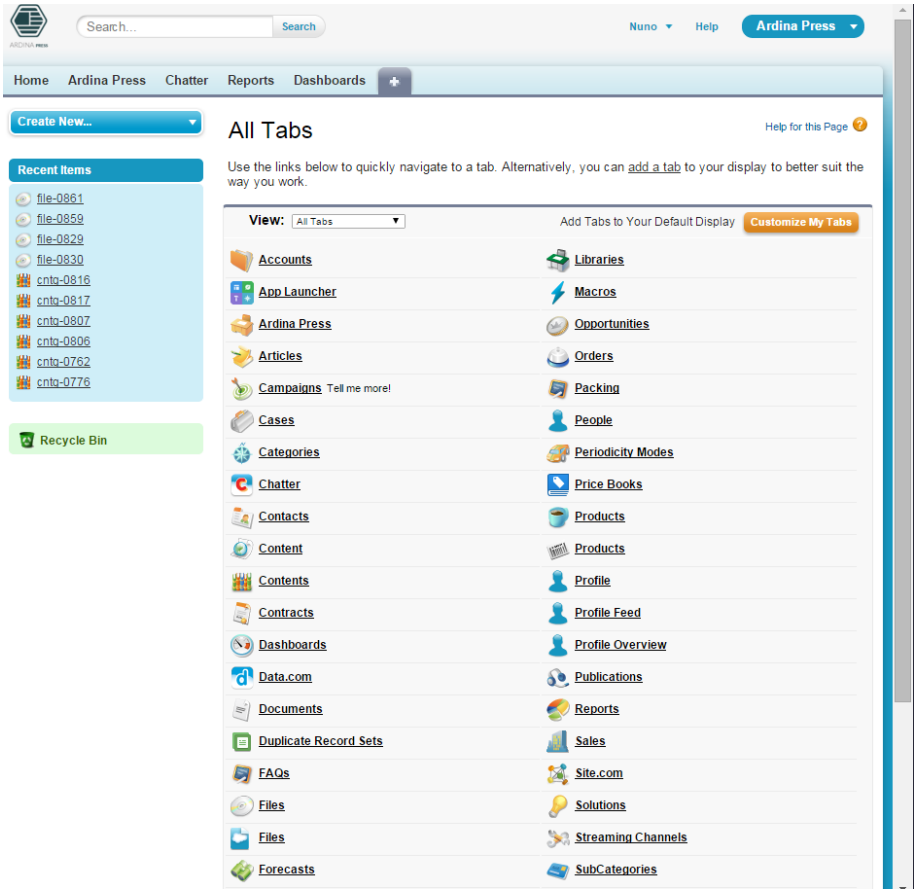


*Figura 8 - Website ardina.press.*

A Figura 9, mostra a página de criação de um novo conteúdo.

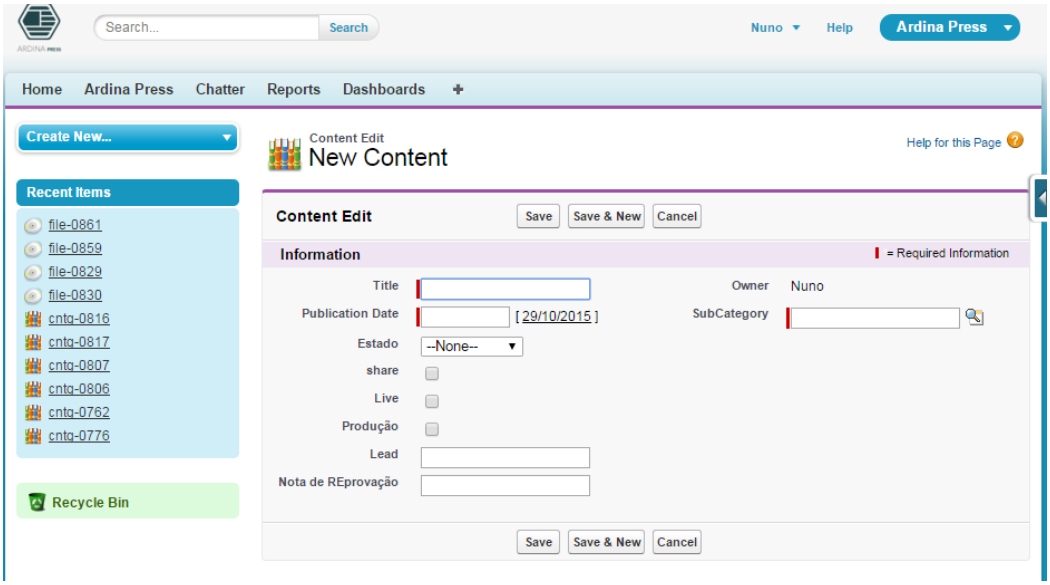


*Figura 9 - Website ardina.press criação de um novo conteúdo.*

Depois de se criar um conteúdo podemos adicionar um artigo e ficheiros multimédia.

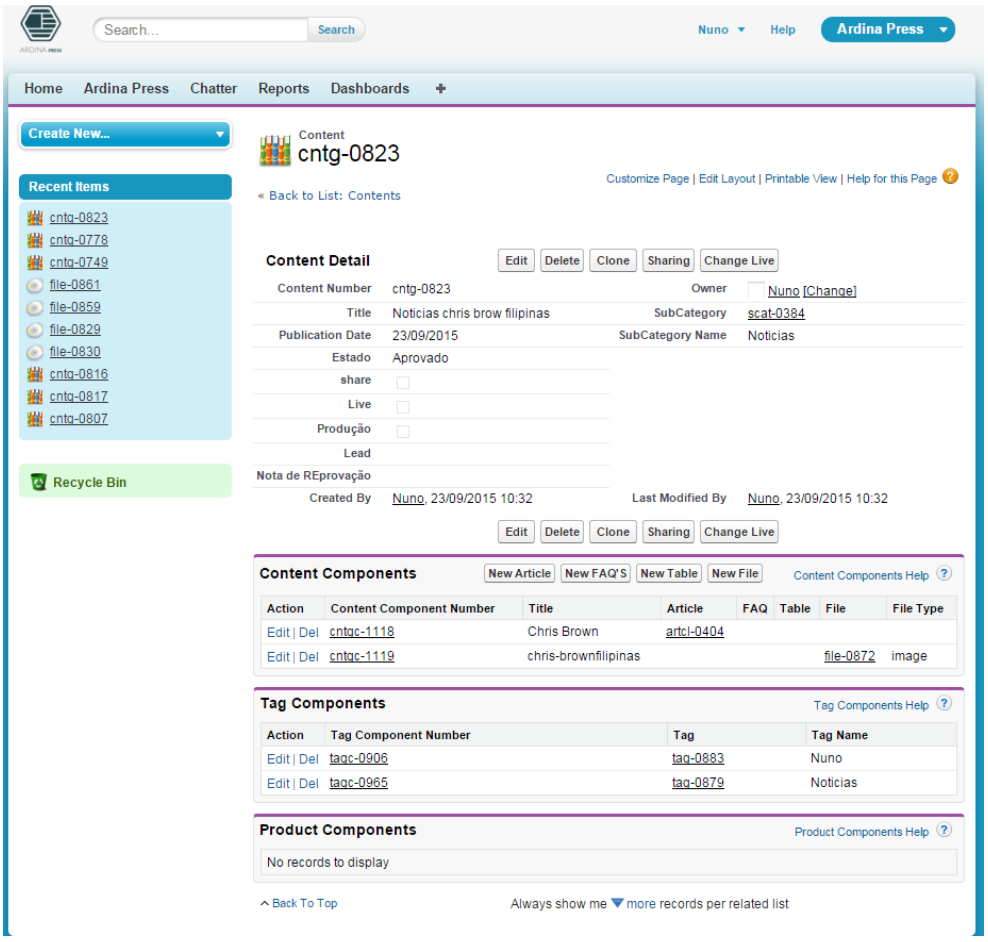


*Figura 10 - Website ardina.press depois da criação de um novo conteúdo.*

Nesta Figura 11, mostra os parâmetros que podemos preencher quando se cria um novo artigo.

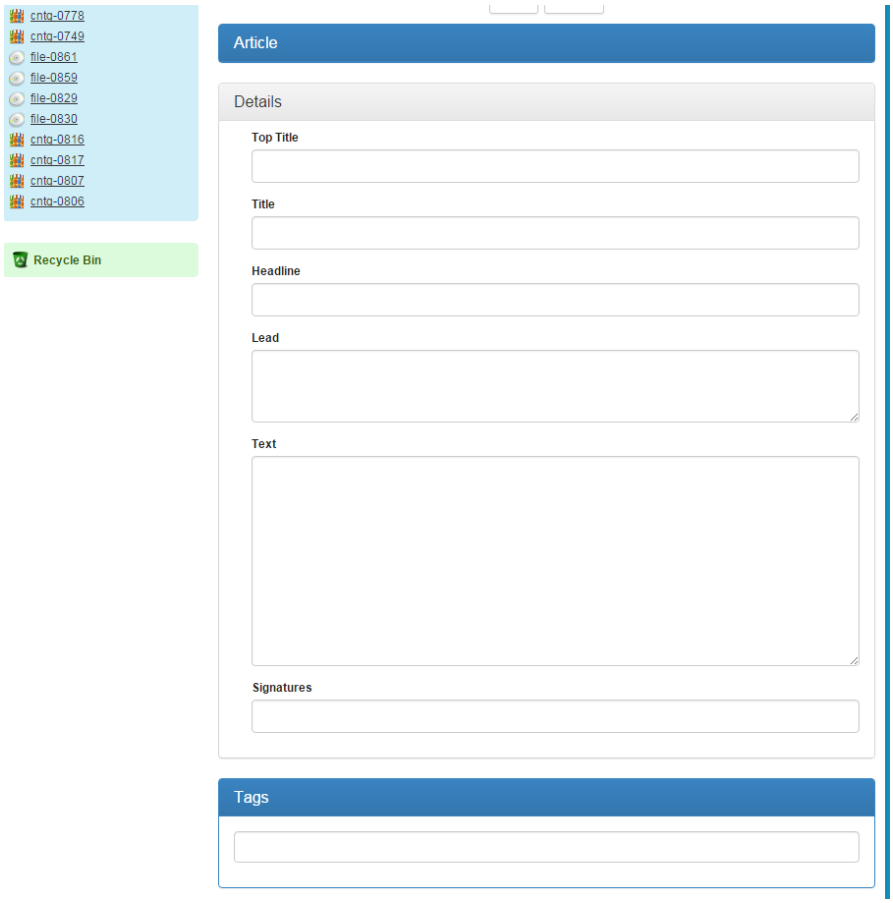


*Figura 11 - Website ardina.press criação de um artigo.*

Esta figura mostra a página para carregarmos ficheiros multimédia.

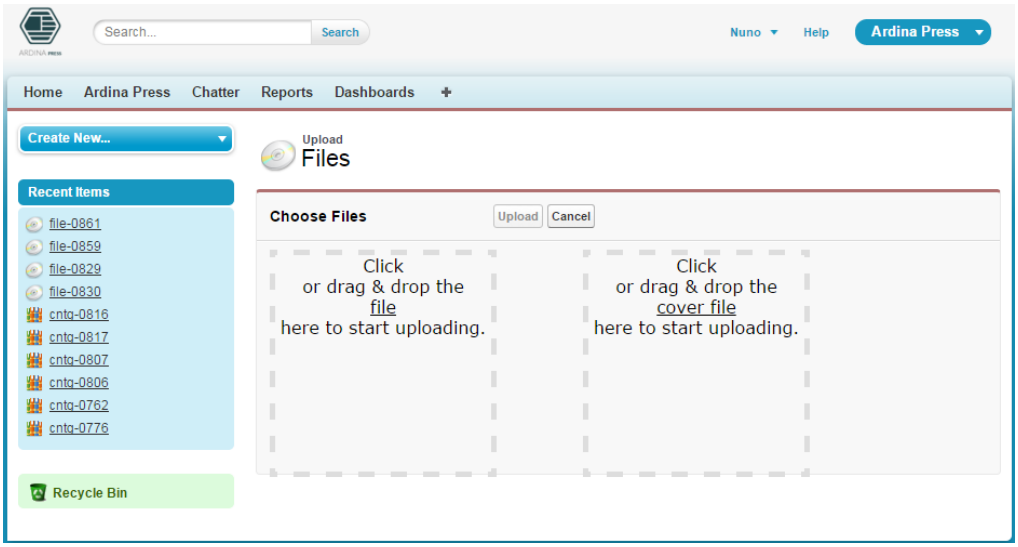


*Figura 12 - Website ardina.press para carregarmos conteúdo multimédia.*

# 8 Implementação

O objetivo principal do projeto é o de criar um website sobre música.

Este trabalho foi iniciado com a leitura de alguns tutoriais no website udacity para aprender a linguagem.

## 8.1 HTML/CSS

O primeiro tutorial foi sobre HTML/CSS para ter uma boa noção de como o HTML/CSS é constituído.

Para utilizar a *framework backbone.js* é necessário que o HTML seja todo dividido em linhas e colunas. Foi elaborado um pequeno exemplo de um website utilizando somente classes de CSS criadas, para se aprender como é efetuada a divisão entre as linhas e colunas.


*Figura 13 - Website de implementação do tutorial HTML/CSS.*

## 8.2 JavaScript

Outro tutorial que foi proposto foi sobre JavaScript como nunca se tinha programado nesta linguagem seria-me útil saber todas as suas funções e de como é constituída.

Visto que a linguagem base da *framework* é feita em JavaScript, foi feito um pequeno exemplo, para pôr em prática os conhecimentos adquiridos.

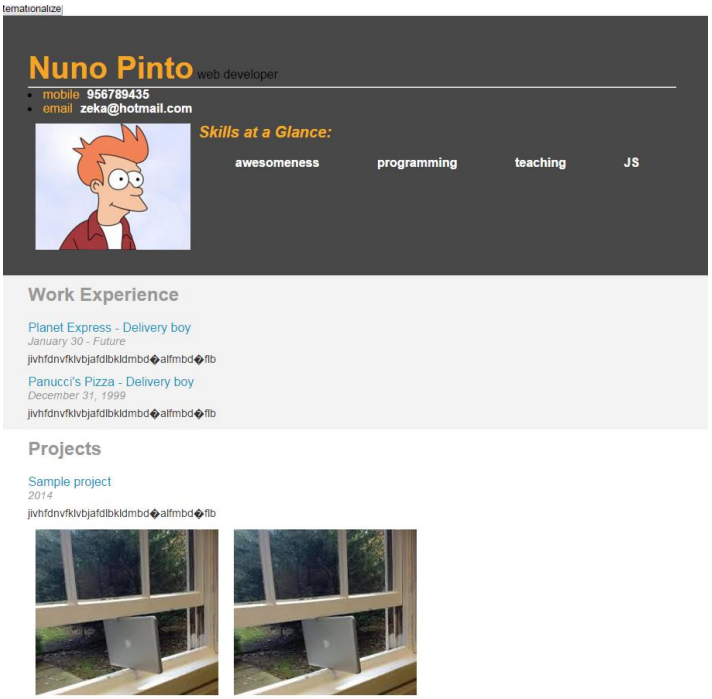


*Figura 14 - Website de implementação do tutorial JavaScript.*

## 8.3 Backbone.js

Na penúltima etapa foi dado um design, para começar a programar em backbone.js, com ajuda dos colaboradores da empresa para começar a perceber como é constituída a *framework*.

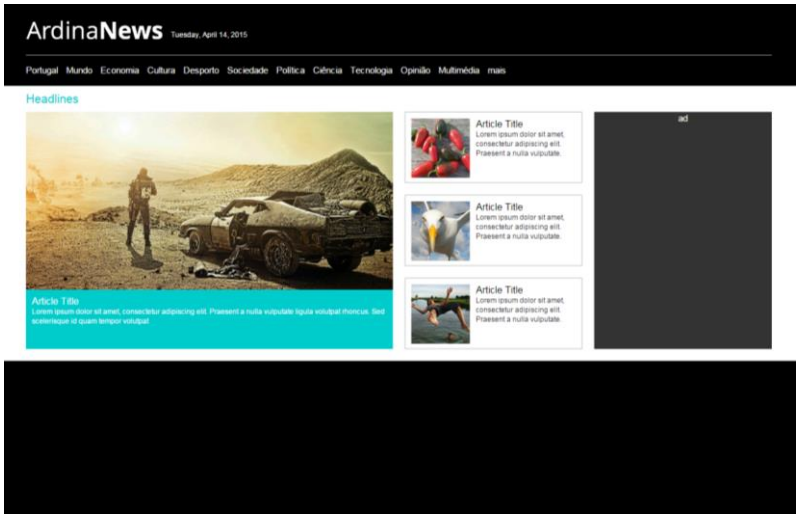


*Figura 15 - Design dado pela empresa como exemplo.*

Depois de convertido em HTML/CSS, dividido em Views e com os dados vindos da ardina.api ficou com este aspeto.

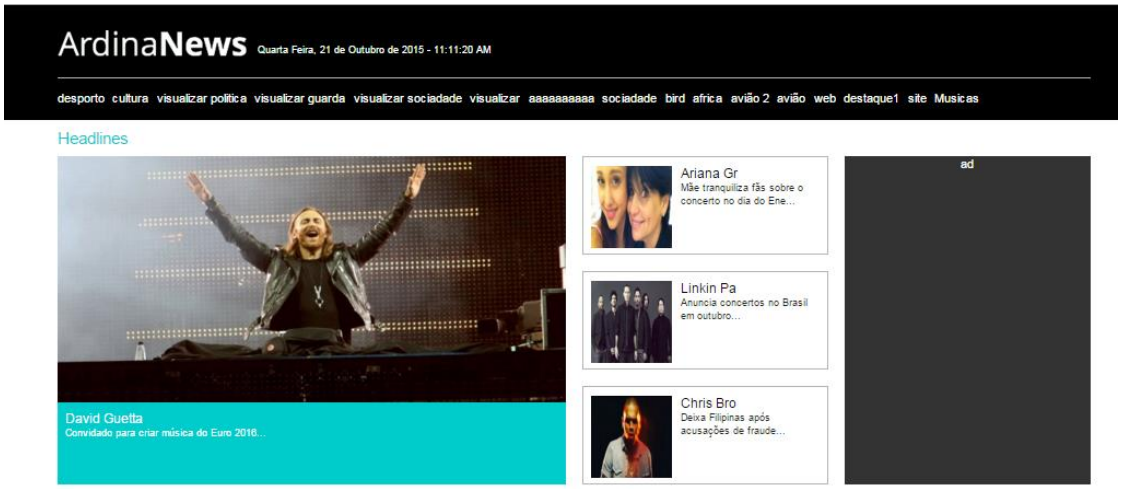


*Figura 16 - Design dado pela empresa convertido.*

Na empresa as tarefas de desenvolvimento do website são divididas pelos vários colaboradores. Um colaborador trata dos designs dos websites, outro converte tudo em HTML/CSS e por fim o desenvolvedor torna o produto funcional e interativo através de JavaScript.

Quando o modo de trabalho com a *framework* backbone.js todo o website é dividido por Views, é necessário dividir todo o HTML por Views.

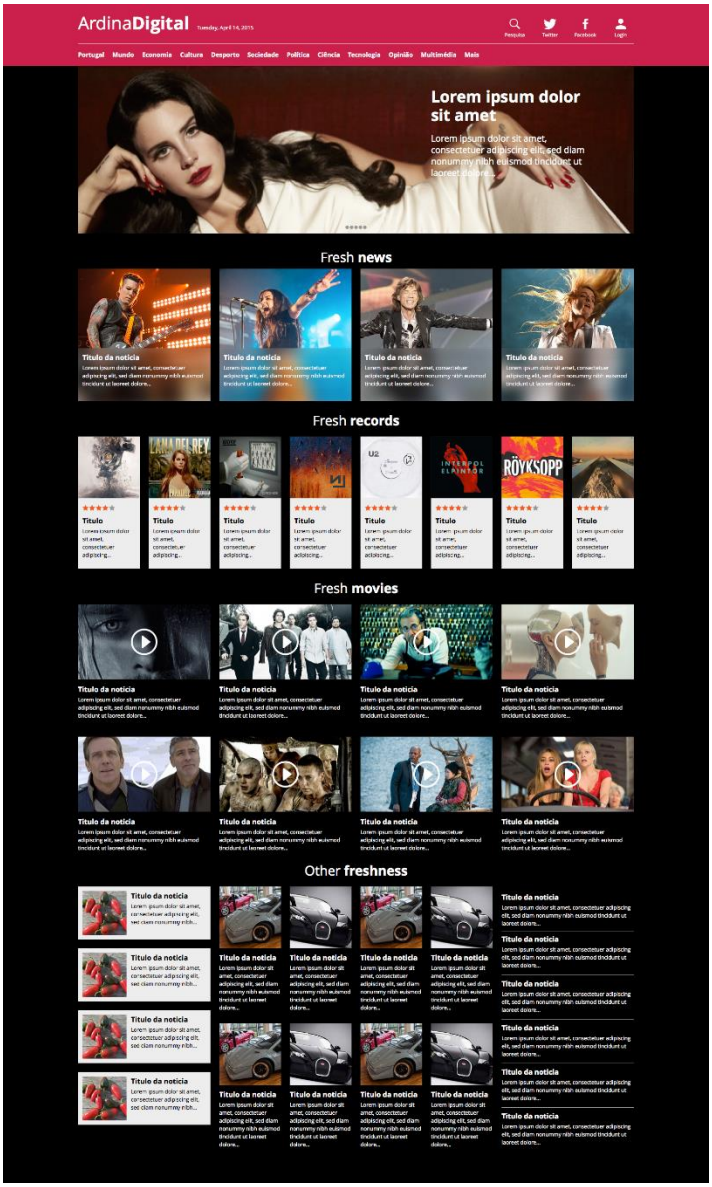


*Figura 17 - Design de exemplo para o meu projeto.*

Olhando para o design e sabendo a *framework* de HTML/CSS da empresa é feita com base em colunas e linhas, para que seja possível que ao trocar as classes do CSS, seja possível moldar todo o aspeto mas a base ser a mesma, decidiu-se criar um design e assim saber um pouco da área de design.

Começou-se por elaborar um esboço de como se achava que o website ficaria bem, o qual foi aprovado pelo colaborador responsável pelo design da empresa.

Após a aprovação do colaborador, foi alterado o HTML/CSS do exemplo de treino que foi feito anteriormente, de modo a que corresponde-se ao design do esboço feito.

Depois desta etapa, foi necessário dividir todo o HTML em Views, e implementar todo o código que está por detrás.

## 8.4 Arquitetura

A arquitetura, como mostra a Figura 18, descreve os componentes de *hardware* e *software* e respetiva interação com outros elementos de suporte ao processamento.


*Figura 18 - Arquitetura*

## 8.5 Divisão do HTML em Views

O conceito desta fase é dividir todo o HTML de um ficheiro em várias partes, da forma que acharmos mais adequado, que vão ser colocados em outros ficheiros HTML, podendo serem divididos ainda mais.

O Backbone.js funciona por (IDs) para saber a que elemento do HTML se refere.

O que o backbone.js vai fazer é colocar o HTML nesses ficheiros que por sua vez vai ser inserido no HTML da página principal.

Fazendo a analogia é como pensar que Portugal está dividido em diferentes distritos e cada distrito tem vários concelhos mas todos pertencem a Portugal.

O backbone.js é como se fosse as estradas.

Na Figura 19 é mostrada a constituição do ficheiro HTML principal:

- No head é onde se coloca todas as bibliotecas, título do website e todos os links.
- Na div é atribuído o id="header" para que a View saiba que vai ser nesta div que vou imprimir o cabeçalho com os menus.
- Na div com o id="content" é onde vai ser visualizada toda minha estrutura.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>Ardina Music</title>
    <meta name="description" content="Write an awesome description for your new site here. You can edit this line in _config.yml. It will appear in your document head meta (for Google search results) and in your feed.xml site description.
  ">
    <link rel="stylesheet" href="css/main.css">
    <link rel="canonical" href="http://yourdomain.com/">
    <link rel="alternate" type="application/rss+xml" title="Your awesome title" href="http://yourdomain.com/feed.xml" />

    <script type="text/javascript" src="libs/require.min.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
    <script type="text/javascript" src="http://cdnjs.cloudflare.com/ajax/libs/underscore.js/1.5.2/underscore-min.js"></script>
    <script type="text/javascript" src="http://cdnjs.cloudflare.com/ajax/libs/backbone.js/1.1.0/backbone-min.js"></script>
    <script type="text/javascript" src="//cdnjs.cloudflare.com/ajax/libs/draggabilly/1.2.0/draggabilly.pkgd.min.js"></script>
    <script  type="text/javascript" src="app.js"></script>
  </head>

  <body>
    <header>
        <div id="header" class="header-container">

        </div>
    </header>
      <div id="content"  class="container">

      </div>
  </body>
</html>
```

*Figura 19 - Constituição do ficheiro HTML principal*

Na Figura 20 é apresentado um exemplo de outro ficheiro HTML que vai ser impresso na div com o id content na página principal.

```
<div id ="exemplo" class="cols two">
  <article class="fifteen-rows">
    <div class="inner-cols fullwidth">
      <figure class="fourteen"><img src="img/140x140a.jpg" alt=""></figure>
    </div>
    <div class="inner-cols fullwidth has-paddingtopbottom">
      <h1 class="headline-title"><a href="post.html">Article Title</a></h1>
      <p class="headline-lead">Lorem ipsum dolor sit amet.</p>
    </div>
  </article>
</div>
```

*Figura 20 - Constituição de um ficheiro HTML secundário*

A div com o id exemplo vai ficar neste ficheiro, tudo o resto vai ser colocado no ficheiro JavaScript, especificamente numa View para ser impresso nesta div.

## 8.6 Resultado do meu projeto

Como página principal é onde se encontra a notícia principal com botões para mudar de notícia.

Um reprodutor de música.

Uma secção de notícias de destaque, a seguir outra secção com mais notícias.

Por último um *scroll* com alguns dos videoclips recentes.

**Divisão de Views:**

1ª View - A notícia principal, destaques e mais Noticias.

2ª View – O reprodutor de música.

3ª View – Secção de vídeos.

Isto porque as notícias vêm de um URL da API, as músicas vêm de outro e os vídeos de outro URL.

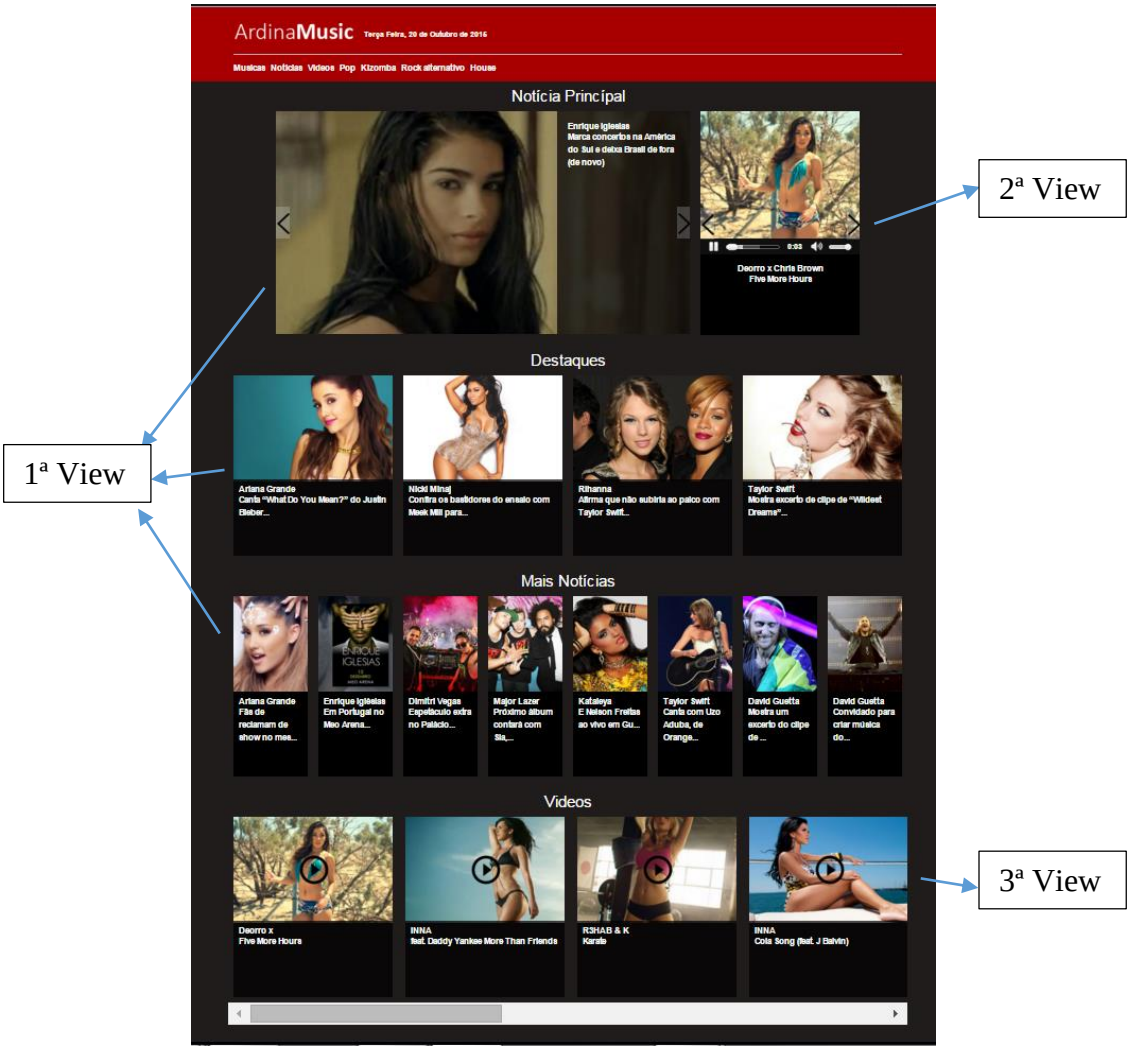


*Figura 21 – Página principal do meu website.*

Quando se escolhe um dos menus apaga as Views do corpo da página e mostra esta outra View e vai mudando de View consoante o menu que se escolher.

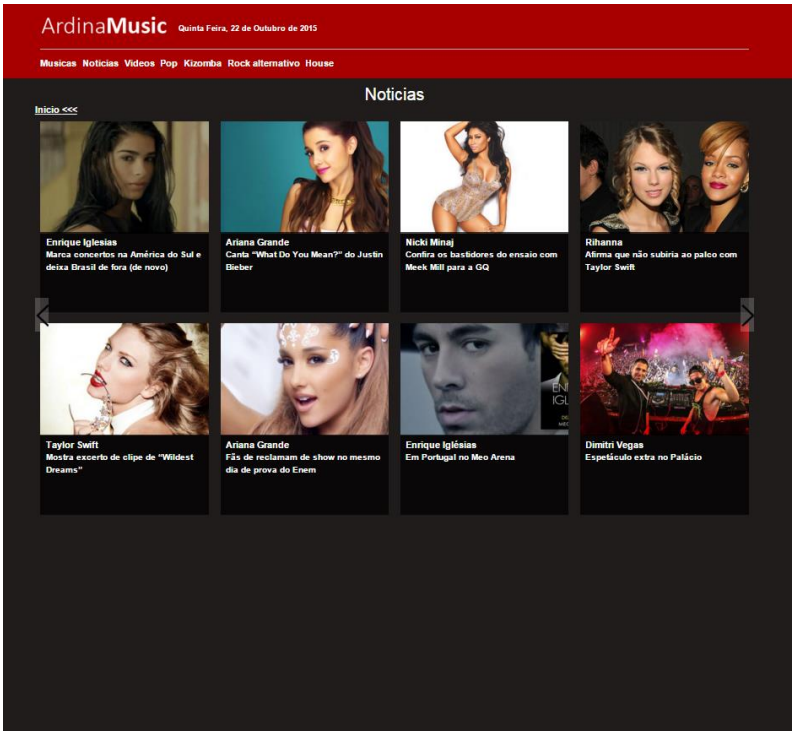


*Figura 22 - Captura de ecrã da página intermédia do meu website.*

Ao pressionar um dos menus de escolha rápida filtra os videoclips conforme o tipo de música utilizando a tecnologia de tags.

A View a seguir é devolvida quando se carrega no tipo de música POP.

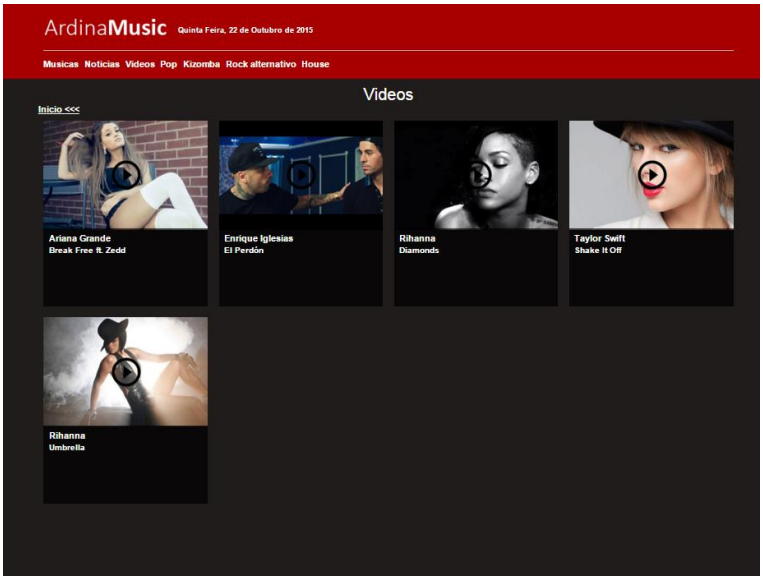


*Figura 23 - Página intermédia utilizando tags do meu website.*

Quando se pressiona numa notícia mostra estas Views.

**Divisão de Views:**

1.ª View – Noticias relacionadas.

2.ª View – Mais noticias.

3.ª View – Tudo o resto.

Podendo se mudar de notícia tanto na secção Noticias relacionada como na secção mais noticias.

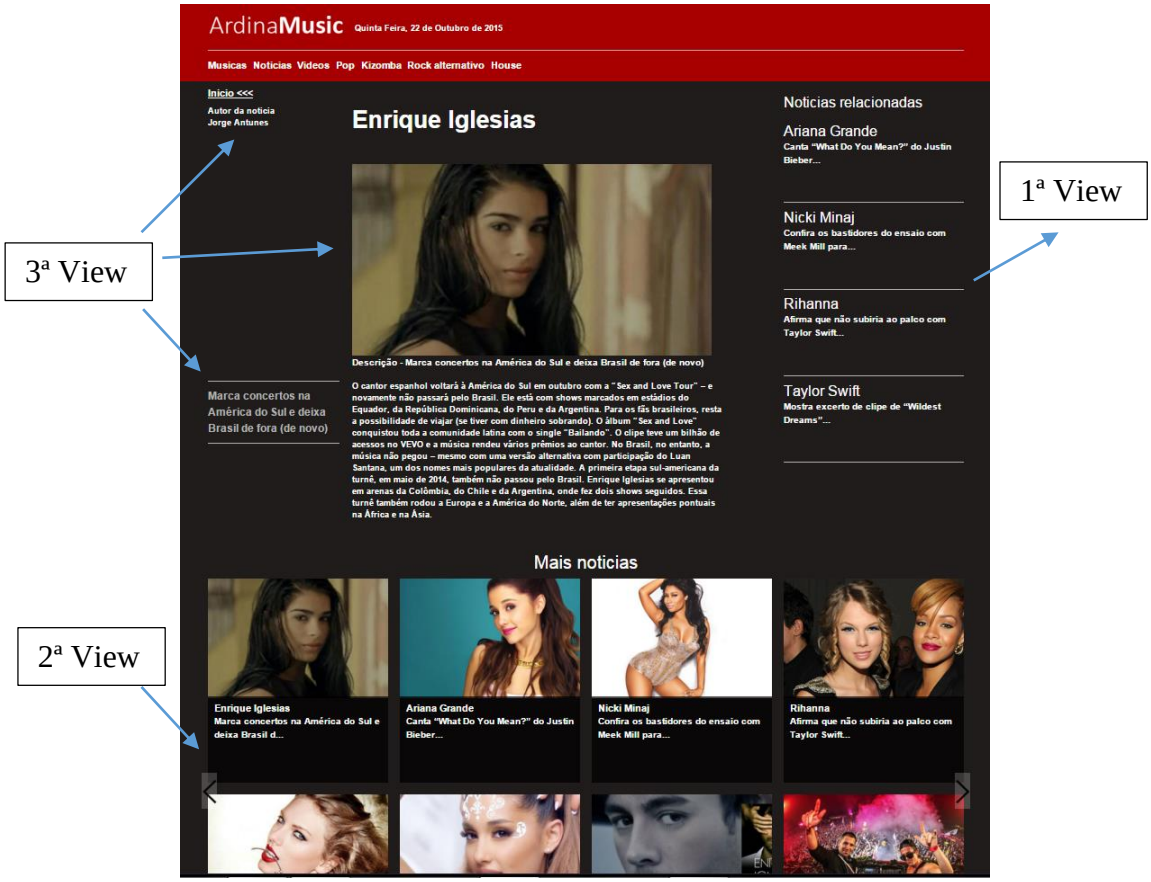


*Figura 24 - Página de notícias do meu website.*

Quando se seleciona algum vídeo mostra esta View para visualização de videoclips com a opção de mudar de videoclips, no *scroll* possibilitando mudar de vídeos sem ter que deixar de ver o que se está a visualizar.

Pode-se ainda visualizar todos os vídeos na secção Vídeos.

**Divisão de Views:**

1.ª View – Reprodutor de vídeo.

2.ª View – *Scroll* de videos.

3.ª View – Secção de vídeos.

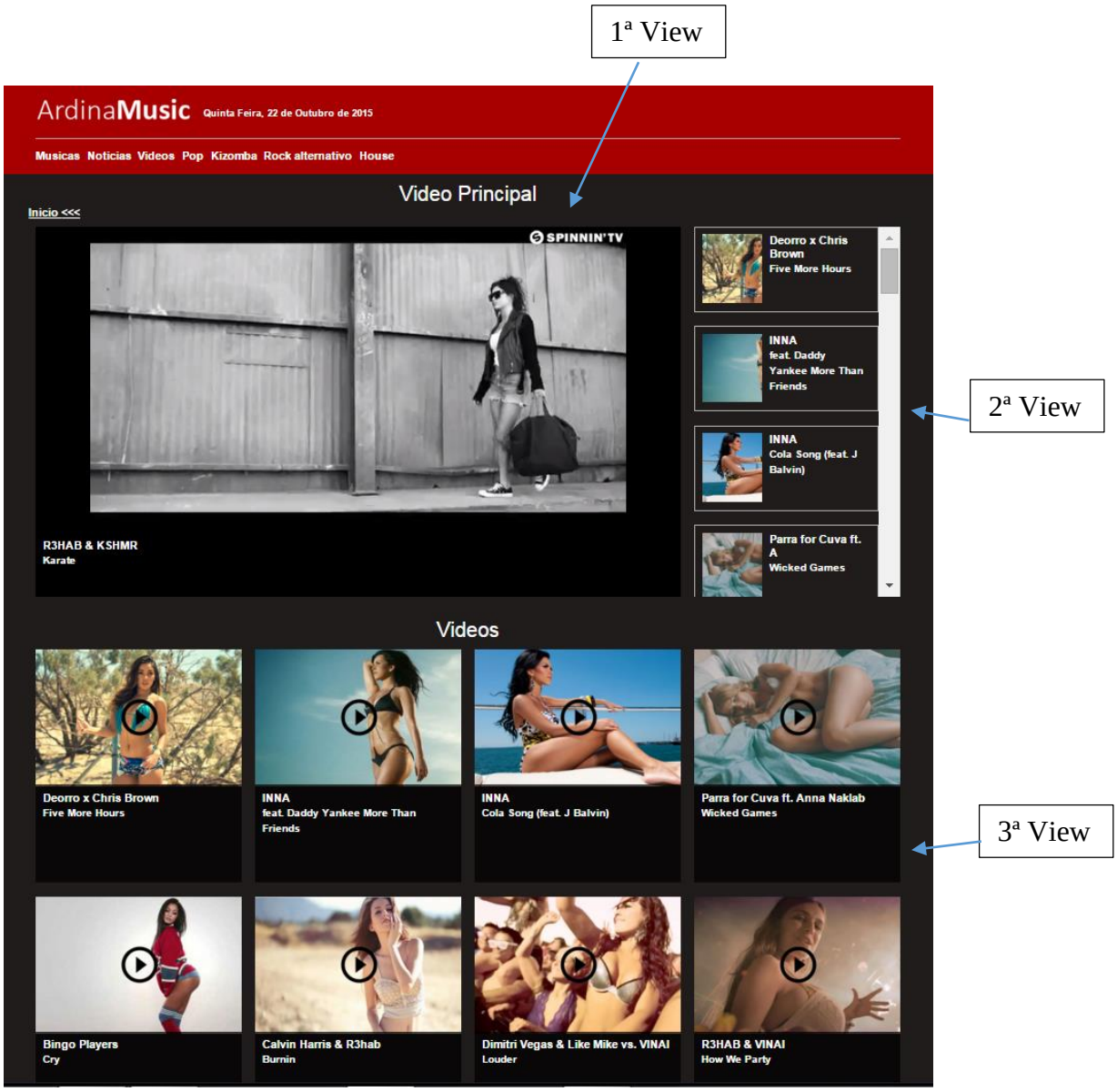


*Figura 25 - Página de visualização de vídeos do meu website.*

Quando se seleciona uma música mostra estas Views de reprodução de músicas, com a opção de mudar para a música a seguir ou anterior através de botões ou pelo scroll.

Pode-se ainda visualizar todas as músicas na secção músicas.

**Divisão de Views:**

1.ª View – Reprodutor de música.

2.ª View – Scroll de música.

3.ª View – Secção de música.

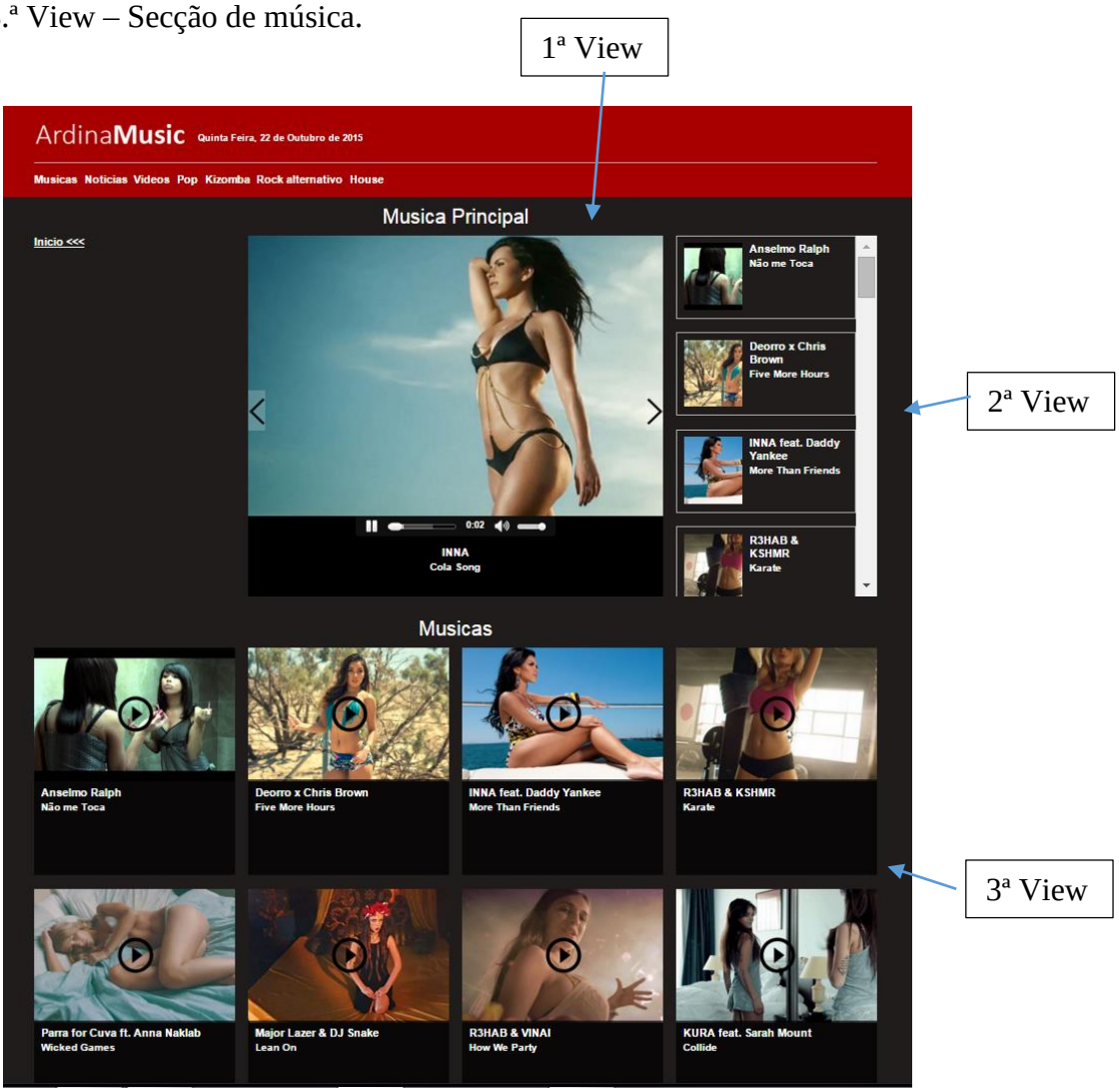


*Figura 26 - Página de reprodução de músicas do meu website.*

## 8.7 Funcionamento o backbone.js

Na figura a seguir é exemplificado de como se cria uma das coleções.

Para se criar uma coleção dá-se o nome da coleção e diz-se que é *framework backbone*, que é uma coleção seguida de extend.

Declara-se a função initialize que vai servir para inicializar alguma variável ou criar algum filtro para os dados que vão ser retornados.

A função sync é onde sincroniza com a ardina.api, tem uma condição para devolver 30 parâmetros e uma função em ajax que diz o tipo de URL, com uma variável myUserKey que é a minha *password* que me identifica e digo que me devolva todos os conteúdos que tenham a tag Vídeos.

Tenho uma função para o caso de receber os parâmetros e outra para o caso de dar erro.

```
var VideoCollection = Backbone.Collection.extend({

    initialize: function (options){

    },

    sync: function(method, collection, options){
        if(method == "read") {
            var pageData = {
                pageSize: 30};
        $.ajax({
            type: "GET",
            url: "https://ardina-api.herokuapp.com/api/1/videos?user_key="+myUserKey+"&tag=Videos",
            crossDomain: true,
            dataType: 'json',
            data: pageData,
        success: function (data) {
            //console.log("---->video",data);
            options.success(data, 'successcontent', null);
        },
        error: function(data) {
            //console.log("errorcontent::",data);
        }
    });


        }else{
            Backbone.sync(method, collection, options);
        }
    }

});
```

*Figura 27 - Criação de uma coleção do meu website.*

Como as Views são criadas no initialize das rotas, não é necessário o var e á frente basta especificar que é uma View.

O el é o id da página principal onde vai ser impresso o template da página que vou construir nesta View.

O initialize utilizo para ler a coleção.

Quando se clicar em algum video faz o evento selectvideopri que vai enviar o id do video selecionado para a View intermedia indexVideoView.

Na função listconf é onde vou imprimir o HTML com dados dinâmicos no id do template.

```
HomevideosView = Backbone.View.extend({
    el: "#content",
    template: "content.html",
    template: contentTpl,
        initialize: function(options) {
            this.listenTo(this.collection, 'sync', this.listcont);
        },

        events:{
        "click .clickContent" : "showContent",
        "click .articlevideos": "selectvideopri"
    },
    selectvideopri: function(ev){
        if(idNew != ev.currentTarget.attributes[0].value ){
             idNew = ev.currentTarget.attributes[0].value;
            var self = this;
            this.indexVideoView = new IndexVideoView({id: ev.currentTarget.attributes[0].value});
            this.indexVideoView.render3(ev.currentTarget.attributes[0].value);
            $('html, body').animate({ scrollTop: 0 }, 'fast');
             idNew = ev.currentTarget.attributes[0].value;
        }

    },

    listcont: function(){
            var cont = 0;
            var self = this;
            for(var i = self.collection.models[0].attributes.data.files.length; i > 0; i--){
                var video = self.collection.models[0].attributes.data.files[i];



                if(video != null){
                    if( cont>=0 && cont<10){
                        var resulttitle = video.title.substring(0,9);
                        self.$("#videopri").append('<div  data-type='+video.id+' class="cols four
                            articlevideos"><article class="is-absolute fifteen-rows colorinvert" ><
                            div  class="inner-cols fullwidth"><figure class="twenty"><img src='+
                            video.cover_url+' alt=""></figure></div><div  class="inner-cols
                            fullwidth has-padding colorinvert has-background black alpha" data-
                            article="article-text"><h1 class="headline-title kiwi"><a >'+resulttitle
                            +'</a></h1><p class="headline-lead">'+video.subtitle+'</p> </div><div
                            id="paging-player"></div></article></div> ');
                    }cont ++;
                }
            }

    },
```

*Figura 28 - Implementação de uma View do meu website.*

A função returndata mostrada na Figura 30 serve para devolver uma coleção.

A render é para desenhar na página o *template* e a *close* serve para remover a View da página.

```
    showContent: function(ev) {
        //console.log(ev);
    },
    returnData: function(){
        return this.collection;
    },
    render: function() {
        $(this.el).html(_.template(this.template));
    },

    close: function(){
        this.remove();
    }
});
```

*Figura 29 - Funções de uma View do meu website.*

Esta View intermédia faz a ligação da View principal com as rotas onde no render3 faz o tratamento dos dados e envia o id guardado na variável y para a rota home1.

```
IndexVideoView = Backbone.View.extend({
    el: "#content",
    template: "contentprivideos.html",
    template: contentprivideosTpl,

    initialize: function(options) {
    idtes: options.id ||"";
    },

    render: function() {
        $(this.el).html(_.template(this.template));
            app.navigate('home/'+this.idtes,  { trigger: true });

    },
    render3: function(y) {
        $(this.el).html(_.template(this.template));
            app.navigate('home1/'+y,  { trigger: true });


    },

    close: function(){
        this.remove();
    }
});
```

*Figura 30 - Implementação de uma View intermédia do meu website.*

A rota home1 recebe o id e envia-o para a home3.

No inicialize vai ser onde se vai criar as Views.

```
var ApplicationRouter = Backbone.Router.extend({
    routes: {
        "home": "home",
        'home/:id': 'home2',
        'home1/:id': 'home3',
        'home2/:id/:i': 'home4'   /*/"+i  para começar logo na musica que selecionei*/
    },
    initialize: function() {
        var tagsCollection = new TagsCollection();
        this.headerView = new HeaderView({collection: tagsCollection});
        this.headerView.render();
        tagsCollection.fetch();


        this.contentCollection = new ContentCollection();
        this.homeView = new HomeView({collection: this.contentCollection});
        this.homeView.render();


        this.contentpripostView = new ContentpripostView({collection: this.contentCollection});
        this.contentpripostView.render();
       // contentCollection.fetch();

        this.maisnotiView = new MaisnotiView({collection: this.contentCollection});
        this.maisnotiView.render();
      //  contentCollection.fetch();

        this.coldireitaView = new ColdireitaView({collection: this.contentCollection});
        this.coldireitaView.render();
        this.contentCollection.fetch();
```

*Figura 31 - Implementação das rotas e inicialização das Views.*

Nesta função compara o id do vídeo que selecionamos com todos os ids da coleção, para saber qual selecionei e a seguir vai imprimir o vídeo que selecionei na View, responsável pela reprodução de vídeos.

```
home3: function(id) {
    var x = this.videoCollection.models[0].attributes.data.files.length;
    for(var i = 0; i < x; i++){
        //console.log(this.videoCollection.models[0].attributes.data.files[i].id,':::::::',id);
        if(this.videoCollection.models[0].attributes.data.files[i].id == id){
            //this.homeView = new HomeView({model: this.videoCollection.models[0].attributes.
            data.contents[i]});


            this.videoView.render();
            this.videoView.render2(this.videoCollection.models[0].attributes.data.files[i]);
            this.videoCollection.fetch();


            }
    }


},
```

*Figura 32 - Implementação do destino de uma rota.*

## 8.8 Avaliação do website

Para teste do website foi utilizado um produto do Google developers que se chama PageSpeed Insights que analisa o desempenho a qualidade do website.

### 8.8.1 Resultados obtidos

**Pontos positivos.**

Nenhum problema encontrado.

O texto no website é legível, permite que seja exibido apropriadamente em todos os dispositivos.

Os conteúdos do website ajustam-se à janela de visualização, não usa plug-ins, o que previne o uso de conteúdos em muitas plataformas.

Anúncios intercalares para a instalação de aplicativos que ocultam uma parte significativa do conteúdo.

Todos os *links*/botões do website são grandes o suficiente para que um utilizador toque com facilidade no *touchscreen*.

**Sugestões**

Otimizar as imagens, mas se o website fosse vendido a algum cliente, seria o cliente a colocar conteúdo.

Utilizar a cache do navegador, no entanto não achei necessário utiliza-la porque o meu conteúdo do website é dinâmico.

De compactar o JavaScript, mas como o website é feito em backbone.js tudo o que é JavaScript está tudo num ficheiro, o que consideram ser muito extenso.

Compactar CSS, não apliquei porque é uma *framework* da empresa.

# 9 Conclusão

Neste projeto foram descritas algumas tecnologias para a construção de websites, nomeadamente *frameworks* para JavaScript, PaaS e HTML/CSS. No entanto, relativamente à framework para HTML/CSS, foi utilizada a framework da empresa Dom Digital.

Neste projeto foram referidas todas as etapas de desenvolvimento até ao produto final.

O objetivo era realizar um website responsivo sobre musica pronto a comercializar, o qual consegui cumprir.

A conclusão a que se chegou é que a framework backbone.js consegue ser um pouco confusa para quem não está habituado a esta forma de programar. Depois de se conseguir perceber o conceito ela é uma ferramenta muito útil para que o código seja mais lógico e com menos possibilidades de ocorrerem erros. Consegue ser uma boa base para quem quiser iniciar projetos, tendo este framework como base de desenvolvimento. Para quem já iniciou projetos, pode-se utilizar os conceitos do backbone.js para obter novas funcionalidades no seu projeto.

Resumindo, foi um excelente projeto que tivemos muito prazer em realizar apesar de no início considerar muito difícil, por esta forma de programar ser tão diferente de tudo o que já tinha até hoje aprendido.

## 9.1 Implementações futuras:

- Sistema de login para utilizadores;
- Website pode ser utilizado para uma rádio.

# 10 Bibliografia


1. modelo scrum. *devin.* [Online] [Citação: 3 de Junho de 2015.] http://www.devin.com.br/modelo-scrum/.

2. Scrum: A Metodologia Ágil. *MindMaster.* [Online] [Citação: 3 de Junho de 2015.] http://www.mindmaster.com.br/scrum.

3. computação em nuvem. *Wikipédia.* [Online] [Citação: 6 de Outubro de 2015.] https://pt.wikipedia.org/wiki/Computa%C3%A7%C3%A3o_em_nuvem.

4. MVC. *Wikipedia.* [Online] [Citação: 25 de Outubro de 2015.] https://pt.wikipedia.org/wiki/MVC.

5. Choi, De Phil, McGuire, Chris e Roth, Caroline. *Salesforce.* s.l. : salesforce.com,inc, 2013.

6. Salesforce.com. *Wikipedia.* [Online] [Citação: 14 de Junho de 2015.] https://en.wikipedia.org/wiki/Salesforce.com#Force.com.

7. AWS Elastic Beanstalk. *Amazon.* [Online] [Citação: 16 de Outubro de 2015.] https://aws.amazon.com/fr/elasticbeanstalk/.

8. O que é o Elastic Beanstalk. *Amazon.* [Online] [Citação: 24 de Outubro de 2015.] http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html.

9. heroku. *imasters.* [Online] [Citação: 15 de Outubro de 2015.] http://imasters.com.br/box/ferramenta/heroku/.

10. O que é o Google App Engine. *Google cloud.* [Online] [Citação: 24 de Outubro de 2015.] https://cloud.google.com/appengine/docs/whatisgoogleappengine.

11. Serviços de cloud computing PaaS. *InfoQ.* [Online] [Citação: 27 de Outubro de 2015.] http://www.infoq.com/br/articles/paas_comparison.

12. Principais Frameworks de Javascript. *linhadecodigo.* [Online] [Citação: 26 de Outubro de 2015.] http://www.linhadecodigo.com.br/artigo/3637/principais-frameworks-de-javascript.aspx.

13. Angular.JS. *caelum.* [Online] [Citação: 10 de Outubro de 2015.] http://blog.caelum.com.br/como-anda-o-angular-js-devo-embarcar-nessa/.

14. JavaScript MVC. *Funnyant.* [Online] [Citação: 20 de Outubro de 2015.] http://www.funnyant.com/choosing-javascript-mvc-framework/.

15. Top 10 Frameworks Javascript MVC. *CodeBrief.* [Online] [Citação: 24 de Outubro de 2015.] http://codebrief.com/2012/01/the-top-10-javascript-mvc-frameworks-reviewed/.

16. O que é o Bootstrap? *prestashop.* [Online] [Citação: 23 de Outubro de 2015.] https://www.prestashop.com/blog/pt/2014/03/06/o-que-e-o-bootstrap-verdades-e-mitos-parte-1-de-2/.

17. Conhecendo o framework front-end Foundation. *devmedia.* [Online] [Citação: 6 de Outubro de 2015.] http://www.devmedia.com.br/conhecendo-o-framework-front-end-foundation/27160.

18. Boilerplate um template rápido. *Claudio Felis.* [Online] [Citação: 8 de Outubro de 2015.] http://www.claudiofelis.com.br/blog/materia/boilerplate-um-template-rapido-html5/.

19. Os 5 Mais Popular Frontend Frameworks. *Sitepoint.* [Online] [Citação: 26 de Outubro de 2015.] http://www.sitepoint.com/5-most-popular-frontend-frameworks-compared/.

20. Coleção das melhores front-end. *CSS Front-end.* [Online] [Citação: 7 de Outubro de 2015.] http://usablica.github.io/front-end-frameworks/compare.html.

21. domdigital - ardina.api. *domdigital.* [Online] [Citação: 2 de Junho de 2015.] http://www.domdigital.pt/produtos/ardina.com/ardinaapi.asp.

22. Backbone.JS – Framework Estrutural. *2XT.* [Online] [Citação: 9 de Julho de 2015.] http://www.2xt.com.br/backbone-js-framework-estrutural-para-aplicacoes-javascript-ricas/.

23. Contolo de versões. *Wikipedia.* [Online] [Citação: 16 de Outubro de 2015.] https://pt.wikipedia.org/wiki/Sistema_de_controle_de_vers%C3%A3o.

24. domdigital - ardina.press. [Online] [Citação: 18 de Outubro de 2015.] http://www.domdigital.pt/produtos/ardina.com/ardinapress.asp.