

Mestrado em Engenharia Informática
Estágio
Relatório Final

Plataforma de Gestão de Identidades de Marcas

Tiago Mota
tmota@student.dei.uc.pt

Orientadores:
Ernesto Costa
João Cunha
28 de Janeiro de 2014



FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Resumo

O modo tradicional em como agências de *design*, empresas e instituições lidam com as suas marcas há muito se tornou obsoleto. Até muito recentemente as soluções encontradas pelas marcas como meios facilitadores para divulgação e partilha das suas identidades abarcavam um leque que ia desde a criação de plataformas *online* (específicas para cada marcas) até ao passar em mão informação pretendida através de um suporte físico (e.g.: pen ou CD). De facto, a não existência de soluções web genéricas associada às necessidades da Ferrand, Bicker & Associados (FBA), empresa de *design* gráfico, foram os elementos catalisadores que estiveram na génese deste projecto. Como prova da crescente necessidade deste tipo de soluções constatou-se, no decorrer deste projecto, o surgimento de duas plataformas web que tentam dar resposta a muitas das questões que aqui se serão levantadas.

O que se propõe com este projecto é a criação de uma Plataforma de Gestão de Activos de Marcas para uso global centrada, primeiramente, nas necessidades das agências de *design*. É este enfoque que faz com que este projecto sobressaia em relação aos outros.

Tomadas em consideração as especificações iniciais que identificavam as tecnologias a serem usadas para o desenvolvimento, PHP e MySQL, a abordagem ao problema debruçou-se sobre a utilização de *frameworks* que simplificassem a implementação e o desenvolvimento, mais especificamente: a *Framework* Laravel, para o desenvolvimento em PHP de aplicações *web* baseadas no paradigma MVC, e que contem um *Active Record* para trabalhar com bases de dados relacionais; e o Bootstrap, que se baseia no paradigma *Mobile First*, para desenho de páginas e aplicações *web*.

Palavras-Chave

“Plataforma de Gestão de Activos de Marcas” “PHP 5” “MySQL 5.6” “Javascript” “Laravel 4.1” “Bootstrap 3” “Brand Identity” “Corporate Identity”

Índice

Resumo.....	1
Palavras-Chave	3
Índice	5
Lista de Tabelas.....	6
Lista de Figuras	7
Lista de Diagramas	8
Lista de Fragmentos de Código.....	9
Capítulo 1 Introdução.....	1
1.1. Enquadramento.....	1
1.2. Problema.....	2
1.3. Proposta.....	2
Capítulo 2 Estado da Arte.....	4
2.1. Lingo	4
2.2. Manuais de Identidade de Marcas	4
2.3. Aplicações Existentes	6
Capítulo 3 Planeamento.....	10
3.1. Requisitos Funcionais.....	10
3.2. Requisitos Não Funcionais.....	16
3.3. Diagramas de Casos de Uso	17
3.4. Planeamento do Projecto.....	23
Capítulo 4 Arquitetura	27
4.1. Arquitectura do Sistema.....	27
4.2. <i>Frameworks</i>	30
4.3. Ambiente de Desenvolvimento	32
Capítulo 5 Implementação	34
5.1. Modelação de Dados	34
5.2. Laravel Routing & Filters.....	58
5.3. MVC (<i>Model-View-Controller</i>)	62
Capítulo 6 Conclusões	93
6.1. Aspectos positivos	93
6.2. Aspectos a relatar	93
6.3. Trabalho Futuro	94
Referências.....	95

Anexos.....	95
Anexo 1. Lingo: Definições e Conceitos.....	96
Anexo 2. Wireframes - Mockups.....	101
Anexo 3. Código SQL para geração BD.....	115
Anexo 4. Bootstrap vs Foundation: User Interfaces and Widges.....	134
Anexo 5. Especificações Iniciais.....	136
Anexo 6. Conclusões Pessoais.....	137

Lista de Tabelas

Tabela 1.....	6
Tabela 2.....	11
Tabela 3.....	13
Tabela 4.....	13
Tabela 5.....	16
Tabela 6.....	17
Tabela 7.....	18
Tabela 8.....	19
Tabela 9.....	20
Tabela 10.....	21
Tabela 11.....	22
Tabela 12.....	22
Tabela 13.....	23
Tabela 14.....	23
Tabela 15.....	23
Tabela 16.....	23
Tabela 17.....	23
Tabela 18.....	23
Tabela 19.....	31
Tabela 20.....	35
Tabela 21.....	36
Tabela 22.....	36
Tabela 23.....	36

Tabela 24.....	37
Tabela 25.....	38
Tabela 26 – Relações de funções de utilizador por entidade.....	42
Tabela 27 – Atributos da entidade User.....	43
Tabela 28 – Atributos da tabela ‘user_roles’.....	43
Tabela 29 – Atributos de ‘users’	45
Tabela 30 – Exemplos de endereços da plataforma	45
Tabela 31 – Atributos de Contactos	46
Tabela 32 – Relações de ‘brands’.....	48
Tabela 33 – Atributos de ‘brands’	48
Tabela 34 – Atributos de ‘logo_measurements’	50
Tabela 35 – Atributos de ‘logo_files’	51
Tabela 36 – Atributos de ‘colourway_colours’.....	52
Tabela 37 – Atributos de ‘typography_typefaces’	53
Tabela 38 – Atributos de ‘brand_applications’.....	56
Tabela 39 – Atributos de ‘brand_app_items’.....	56
Tabela 40 – Atributos de ‘brand_app_item_files’	56
Tabela 41.....	64
Tabela 42.....	67
Tabela 43.....	67
Tabela 44.....	80
Tabela 45.....	81
Tabela 46.....	82
Tabela 47.....	82
Tabela 48.....	88
Tabela 49.....	88
Tabela 50.....	89

Lista de Figuras

Figura 1 – Cliente Servidor.....	27
Figura 2 – LAMP Stack.....	27

Figura 3 – Página publica.....	78
Figura 4 – Página publica com barra vertical direita activa para autenticação na plataforma	79
Figura 5 – Página publica com formulário de criação de conta para utilizadores <i>Freemium</i>	79
Figura 6 – Barra de ferramentas com operações visíveis para os vários utilizadores	80
Figura 7 – Menu de navegação no contexto de marca.....	81
Figura 8 – Ecrã de edição do perfil de uma marca	83

Lista de Diagramas

Diagrama 1 – Estados de um pedido.....	12
Diagrama 2 – Casos de uso – Actor ‘Anonymous’.....	18
Diagrama 3 – Casos de uso – Actor ‘Authenticated User’	19
Diagrama 4 – Casos de uso – Actor genérico ‘User Role’ e suas especializações.....	20
Diagrama 5 – Casos de uso – Actor ‘BAMP Admin’.....	21
Diagrama 6 – Casos de uso – Actores ‘BASA’, ‘BOU’, ‘BOA’ e ‘BU’	22
Diagrama 7 – Gantt, fases do projecto.....	24
Diagrama 8 – Gantt, 13 Fev. a 14 Mar.....	25
Diagrama 9 – Gantt, 14 Mar. a 26 Abr.....	25
Diagrama 10 – Gantt, 14 Mar. a 25 Jun.	26
Diagrama 11 – Entidades e Relações – diagrama simplificado.....	39
Diagrama 12 – Relações – Um para um.....	39
Diagrama 13 – Relações – Um para muitos.....	40
Diagrama 14 – Relações – Muitos para muitos	40
Diagrama 15 – Relações – Herança	41
Diagrama 16 – Entidades e Relações – Conta e Sessão	42
Diagrama 17 – Entidades e Relações – ‘Entities’	44
Diagrama 18 – Entidades e Relações – ‘agencies’	46
Diagrama 19 – Entidades e Relações – Marcas.....	47
Diagrama 20 – Entidades e Relações – Identidade Visual.....	49
Diagrama 21 – Entidades e Relações – Logo.....	50
Diagrama 22 – Entidades e Relações – Colourway	51
Diagrama 23 – Entidades e Relações – Typography.....	53
Diagrama 24 – Entidades e Relações – Imagery	54

Diagrama 25 – Entidades e Relações – “Brand Applications”	55
Diagrama 26 – Entidades e Relações – Partilha	57

Lista de Fragmentos de Código

Código 1 – Laravel Routing – Para uma vista (<i>blade</i>).....	59
Código 2 – Laravel Routing – Usagem.....	59
Código 3 – Laravel Routing – Para o controlador, com argumentos (get json).....	59
Código 4 – Laravel Routing – Esqueleto de um destino com argumentos	60
Código 5 – Laravel Routing – Para o controlador, com argumentos (post).....	60
Código 6 – Laravel Route Filters – ‘auth’ para utilizadores autenticados	61
Código 7 – Laravel Route Filters – Agrupamento de reencaminhamentos por um filtro	61
Código 8 – Reflexão MySQL – Tabelas	63
Código 9 – Reflexão MySQL – Chaves primárias (PK).....	63
Código 10 – Reflexão MySQL – Chaves-forasteiras (FK)	63
Código 11 – Reflexão MySQL – Colunas	63
Código 12 – Bibliotecas Java – MySQL Connector J – Ligação	65
Código 13 – Bibliotecas Java – MySQL Connector J – Execução de uma consulta	66
Código 14 – Bibliotecas Java – Inflector – Singular vs. Plural.....	66
Código 15 – Bibliotecas Java – Guava – Conversão de notação de texto	67
Código 16 – Bibliotecas Java – Guava – Coleções	67
Código 17 - Bibliotecas Java – Guava – Coleções – Modelo de classe PHP.....	68
Código 18 - Bibliotecas Java – Guava – Coleções – Assinatura de uma classe PHP	68
Código 19 - Bibliotecas Java – Guava – Coleções – Assinatura de classes PHP	68
Código 20 - Bibliotecas Java – Guava – Coleções – Membros da classe PHP	68
Código 21 - Bibliotecas Java – Guava – Coleções – Classe PHP completa.....	68
Código 22 – Notação JSON	69
Código 23 - Bibliotecas Java – GSON – Serialização em JSON – Base	69
Código 24 - Bibliotecas Java – GSON – Serialização em JSON – Pretty Print	69
Código 25 - Bibliotecas Java – GSON – Serialização em JSON – Esconder atributos.....	69
Código 26 – Laravel Model – Código-base e possíveis instanciações	70
Código 27 – Laravel Model – Com identificação de tabela.....	70

Código 28 – Laravel Model – Colunas fillable	70
Código 29 – Laravel Model – Para dispensar timestamps	70
Código 30 – Laravel Model – Relações Um-para-Um	71
Código 31 – Laravel Model – Relações um para um – Perspectiva inversa	71
Código 32 – Laravel Model – Relações um para muitos	72
Código 33 – Laravel Model – Relações um para muitos – Perspectiva inversa	72
Código 34 – Laravel Model – Relações muitos para muitos -	72
Código 35 – Laravel Model – Relações de Herança – Objectivo.....	72
Código 36 – Laravel Model – Relações de Herança – Relações polimorfas.....	73
Código 37 – Laravel Model – Relações de Herança – Código final.....	74
Código 38 – Laravel Model – Exemplo ‘Page’.....	74
Código 39 – Laravel Model – Exemplo ‘Entity’	75
Código 40 – Laravel Model – Exemplo ‘Agency’	75
Código 41 – Laravel Model – Exemplo ‘User’.....	75
Código 42 – Laravel Model – Exemplo ‘Role’	76
Código 43 – Javascript Controller – Chamada periódica à leitura de sessão	91
Código 44 – Javascript Controller – Invocação assíncrona do serviço de consulta	91
Código 45 – Laravel Controller – Entrega das notificações e pedidos em JSON	91
Código 46 – Laravel Controller – Lazy initialisation da chave de página.....	92
Código 47 – Laravel Controller – Endereço dinâmico com identificação da página	92

Capítulo 1

Introdução

Este documento visa relatar o desenvolvimento de uma plataforma de gestão de conteúdos de marcas, enquadrada no Projecto de Estágio de Mestrado do Curso de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra pelo aluno Tiago João Sário Mota, para a Empresa FBA. - Ferrand, Bicker & Associados.

Contextualiza-se o âmbito do projecto, a sua real importância descrita sob a forma de problema, bem como o que pretende alcançar.

1.1. Enquadramento

Desde tempos remotos até muito recentemente, e especialmente em locais atingidos pelo analfabetismo, mercadores e vendedores utilizaram símbolos na representação de produtos. Posteriormente, os produtos começaram a incluir símbolos que representassem igualmente os seus fabricantes, denotando não somente o local de origem, mas também o de propriedade. Mais tarde, autoridades governamentais instituíram conceitos de patente, marca registada e direitos de autor, protegendo produto, propriedade e produtor, e incentivando o mercado.

O desenvolvimento da indústria e consequente produção em série foi colocando um sem número de produtos embalados à disposição. As embalagens eram utilizadas não só para identificar produto e produtor, mas também para os publicitar, usando para isso uma imagem visual apelativa. Com o aparecimento da Imprensa surgiram os primeiros anúncios, aumentando a necessidade das empresas saberem exprimir e publicitar produtos e serviços com brevidade e precisão. Numa sociedade cada vez mais industrializada, tornou-se imperioso o uso de uma representação gráfica facilmente memorizável, de reconhecimento garantido e que retratasse produto e empresa com sucesso. Impulsionado pelo desenvolvimento dos caminhos-de-ferro, rádio, televisão, fotografia e *design* gráfico, pelo decréscimo da iliteracia e por um acréscimo de condições socioeconómicas favoráveis ao consumo, o conceito de marca foi-se desenvolvendo.

Nos dias de hoje, a marca é o ingrediente chave da estratégia comercial de qualquer empresa ou organização. Tem por intuito atribuir-lhe um corpo concreto através de uma representação gráfica, por meio de símbolo ou logo, imagem ou palavra, diferenciando-a das outras empresas e conferindo-lhe uma presença reconhecível e única. O objectivo é o seu destaque imediato no mercado, atraindo e fidelizando clientes, e tornando-a líder na sua categoria. Toda a estratégia envolvente foca-se na arquitectura da personificação-mistificação da própria marca: personificando-a atribui-lhe carácter, responsabilidades, sonhos e valores, e estabelece uma relação pessoal com os seus clientes; mistificando-a torna-a num bem glamoroso a ser conquistado.

Este processo é moroso e requer que seja determinado como poderá a marca suportar uma estratégia comercial, que seja efectuada uma pesquisa e um inventário de marcas existentes no mercado, que sejam analisadas as perspectivas e necessidades do cliente, e que se estabeleça que tipo de promessa se quer ver cumprida. No que concerne ao *design*, tendo por base a saturação do mercado e o crescente aumento da competição entre empresas, foi-se deixando de lado o conceito tradicional de marca que aposta unicamente numa

representação gráfica imediata. Esforços foram feitos no sentido de transformar esta representação gráfica num alicerce sólido e de a fazer acompanhar de uma mensagem coesa e inabalável, maximizando a sua força e impacto.

Muitas empresas recorrem a Guias de Identidade de Marcas que são documentos, livros ou páginas web construídas por agências de *design*, que possibilitam a construção de marcas robustas ao fornecerem especificações, conselhos e regras para melhor utilizar os elementos que compõem a identidade da marca, em termos de logo, paleta de cores, tipografia, imagem e texto. Estes guias permitem às empresas uniformizar o tom de voz das suas marcas de acordo com os seus interesses, redefinir a comunicação impressa em todas as suas variantes e trabalhar eficientemente a sua presença em todos os canais disponíveis, convidando o cliente a interacção visual e a uma resposta emocional intensas.

A representação gráfica passa assim de objectivo final a ponto de partida.

1.2. Problema

A FBA é um estúdio de *design* que conta com um extenso trabalho, reconhecido, realizado sobre sistemas de identidade visual. Após conclusão do processo de criação da marca ou sistema de identidade visual é criado um manual de normas gráficas que deve ser respeitado por todos os agentes que utilizam e aplicam a marca. Presentemente a FBA fornece aos seus clientes esses manuais, apenas, em suporte físico. No entanto esta forma de distribuição tem algumas limitações, sendo as principais a **difusão**, as **actualizações** e **disponibilidade**; que passamos a detalhar:

- Difusão das normas: implica fornecer uma cópia a cada potencial utilizador;
- Actualizações: sempre que se verifica a necessidade de actualizar alguma parte do manual corre-se o risco de existirem em circulação diferentes versões (algumas desactualizadas) do manual.
- Disponibilidade: cada vez mais empresas e clientes confiam em serviços online para armazenamento de dados, esta tendência tem importantes repercussões tanto a nível dos acessos como à da partilha.

Na generalidade, clientes e ateliers de design experimentam estes e outros problemas quando toca à gestão de marcas e activos visuais. Pretende-se com este projecto cobrir os problemas apresentados tendo como base as seguintes propostas.

1.3. Proposta

Pensado inicialmente para ser uma “Plataforma de Manuais de Normas gráficas de marcas online” com objectivo de vir a ser uma ferramenta que ajudasse a FBA e os seus clientes a gerirem mais eficazmente os activos visuais das suas marcas; o projecto rapidamente evoluiu no sentido de desenvolver algo mais abrangente e global que cobrisse não só as necessidades da FBA mas também de outras empresas do ramo, bem como a de qualquer entidade que pretenda fazer a gestão da sua marca. Por estes motivos, o projecto sofreu alterações¹ ao plano inicial que lhe veio conferir uma maior abrangência no contexto de gestão de marcas e utilizadores finais; desta forma a FBA deixou de ser considerada o utilizador final mas apenas mais um utilizador da plataforma.

Enuncia-se de seguida as principais características projecto que se pretende desenvolver:

¹ Consultar Anexo 5 para leitura das especificações iniciais

1. Criação de um *Software as a Service* (SaaS) cuja filosofia inerente é a da gestão de activos de marca de uma forma generalista; abarcando a *Brand Identity*, *Brand Framework* e *Tone-of-Voice*. Pretendesse numa primeira fase dar resposta à *Brand Identity* deixando para fases futuras de desenvolvimento a *Brand Framework* e *Tone-of-Voice*;
2. Aplicação compatível com o uso em plataformas móveis;
3. Criação de perfis de utilizadores de acordo com o público-alvo e possíveis cenários de utilização:
 - Administrador da aplicação,
 - Cliente do tipo Agência (poderá criar e gerir quantas marcas quiser – utilizador *Premium*),
 - Cliente do Independente (poderá criar e gerir uma marca, poderá também partilhar a sua marca, ou partes da mesma, com os utilizadores que pretender – utilizador *Freemium*)
 - Utilizador da Marca dependente da Agência (a cada marca criada por uma agência está associado um utilizador ao qual a marca pertence. Este poderá partilhar a sua marca, ou partes da mesma, com os utilizadores que pretender)
 - Utilizadores de Marcas (acesso a partilhas criadas por terceiros)
4. Criação de um suporte de gestão partilhas de conteúdos com utilizadores escolhidos pelas marcas.
5. Uso exclusivo de termos em inglês na definição de conceitos relacionados com *design* uma vez que é a língua em que estes se encontram mais bem definidos e estabilizados. Como consequência e uma vez que se pretende que seja uma solução global, todas as interfaces serão desenvolvidos em inglês;

Capítulo 2

Estado da Arte

Serve o presente capítulo para apresentação do conhecimento fundamental sobre o qual versa a temática deste projecto. Este conhecimento foi adquirido após consulta de inúmeros documentos de normas visuais, apenas serão referidos os mais relevantes relegando para as referências os restantes, e de análise de algumas plataformas web.

2.1. Lingo

Como desafio inicial tomou-se como objectivo o encontrar de termos e definições comuns que pudessem ser usados para melhor entender conceitos de design e aos quais fosse possível recorrer para desenvolver de forma coerente a plataforma.

O que seria à partida seria uma tarefa fácil revelou não ser assim tão linear. Veja-se o seguinte exemplo: *Logo* é muitas vezes referido como *Signature*, *Logotype*, *Word Sign* ou *Mark*.

Como forma de ultrapassar esta confusão linguística, tendo em mente que a solução final resultante deste projecto é uma plataforma em inglês, foram analisados “guias de marcas”, “manuais de identidade de marcas” e plataformas *online* de gestão de identidades que foram sendo identificadas como relevantes para o estudo. Como resultado deste estudo foram identificados um conjunto de conceitos, do qual depende o domínio da aplicação.

Estes conceitos foram incluídos em anexo, por se tratar de um documento produzido em inglês e pela sua extensão, e recomenda-se a sua consulta para melhor compreensão dos termos e conceitos usados ao logo de todo o documento.

2.2. Manuais de Identidade de Marcas

Este foi o ponto de partida para um conhecimento mais alargado do contexto do projecto.

O estudo de manuais de normas visuais e guias de identidade de marcas manuais permitiu conhecer os principais temas e conteúdos que estão sempre presentes na definição da identidade de uma marca, são estes:

Identidade Visual:

- Versões de logos;
- Codificação Cromática
- Codificação Tipográfica
- Imagens

Aplicações de Marca:

- Estacionários
- *Templates* de Documentos

2.2.1. *Versões de Logos*

Uma versão pode ser composta por múltiplas opções que servem para classificar o logo.

- **Formatos:** com o intuito de manter as proporções e boa leitura do logo, por vezes são criados logos adaptados para diferentes tamanhos. Forma identificados três tipos:
 - Normal
 - Pequeno
 - Grande
- **Varições de Cor:** este tipo de classificação aplica-se para que o logo seja sempre bem visível quando aplicado em diferentes suportes.(e.g.: é comum a aplicação de logos em documentos para serem impressos a preto e branco, para estes cenários as marcas vulgarmente dispõem de versões a preto e branco ou monocromaticas do seu logo principal.).
- **Tipo:** não sendo um nome generalizado para estas classificações, põem-se no mesmo saco classificações como orientação do logo(horizontal, vertical), forma como é constituído o logo(se tem só palavras[e.g.: FBA. e IBM], se tem sempre palavras e elementos graficos,...)
- **Medidas:** para que a apresentação de um logo seja sempre a mais correcta certas medidas têm sempre de ser levadas em consideração. Identificados como sempre presentes são: margens de segurança e tamanho mínimo. Menos frequente mas identificado por diversas vezes é a definição de ângulos (e.g.: Best Buy e Samsung).
- **Dos & Don'ts:** para que não se corra o risco de uma má utilização da marca é comum se identificarem alguns erros passíveis de serem cometidos. Geralmente estes são identificados na forma de imagens com aplicações incorrectas.

2.2.2. *Codificação Cromática*

A Cor é uma componente fundamental da identidade visual, pelo que deve ser aplicada com o maior rigor possível. A codificação cromática é definida com base em sistema de impressão de cores directas Pantone, e deve servir como referência para todos os restantes sistemas de cor.

Ainda que haja a obrigatoriedade da existência de um conjunto de cores principais, designada por paleta principal, é comum encontrar-se mais do que uma paleta de cores sendo vulgar a existência de pelo menos uma de cores secundárias.

Já no que diz respeito aos sistemas mais comuns para identificação das cores, foi tomado em consideração os exemplos presentes nos manuais da FBA cujos proporcionam uma perspectiva mais focalizada das necessidades das agências. Os sistemas identificados são:

- Pantone
- RGB
- HEX
- CMYK
- RAL
- 3M

Em conclusão poder-se-á dizer que a codificação cromática é composta por uma ou mais paletas de cores e que cada cor é identificada pelo menos por um dos sistemas de cores.

2.2.3. *Codificação Tipográfica*

A inclusão de elementos tipográficos no desenho de um logo e a escolha de tipos de tipos de letra (en: *typefaces*) para comunicações foram as principais observações recolhidas.

Por regra as marcas têm um ou mais tipos de letra que devem ser usadas, sempre que possível, e quando não o é as marcas oferecem alternativas.

Na codificação tipográfica identificam-se todos os tipos de letra usados, agregando-os em grupos de tipos, identificando formas e ambientes correctos para a sua utilização.

2.2.4. Imagens

Sendo uma imagem um meio de comunicação por excelência as marcas tendem a possuir galerias de albuns com registos fotograficos, texturas, icons, entre outros. Estas imagens transmitem uma identificação com a marca servindo muitas vezes de guias para trabalhos futuros.

2.2.5. Aplicações de marca

Trata de qualquer tipo de uso generalizado de uma marca sendo as aplicações mais comuns os estacionarios (e.g.: papeis de carta e envelopes), material publicitário (e.g.: canetas e pens) e *templates* de documentos (e.g.: apresentações em PowerPoint).

2.3. Aplicações Existentes

No inicio do projecto não existia nenhuma plataforma sobre a temática na qual este projecto se debruça. Contudo, durante o decorrer projecto, surgiram duas plataformas que respondem parcialmete ao problema. A primeira de nome “Brandisty responde a alguns dos requisitos propostos para o BAMP. Já em Março de 2014 tomou-se conhecimento de um outro projecto de nome “BrandFolder, do qual também falaremos.

Talvez por falta de soluções ou motivados por outros factores, muitas empresas e instituições possuem plataformas próprias para gerir a(s) suas(s) marca(s). A titulo de exemplo nomearemos em seguida aqueles que escolhemos para casos de estudo: “Government of Canada”, “British Council”, “Valspar”, “Best Buy” e “Heineken Company”. Os casos apontados foram recolhidos e seleccionados, por responsaveis da FBA e pelo estagiário, pela sua qualidade e completude em termos de conteudos.

Tabela 1

	Canada ²	British Council	Valspar	Best Buy	Heineken Company	Brandisty	Brand Folder
Acesso	Público	Público	Público	Público	Privado	Misto	Misto
Gestão de Utilizadores	Não	Não	Não	Não	Sim	Sim	Sim
Base de Dados	MS SQL Server ³	MySQL	unknown	MS SQL Server ⁴	MS SQL Server ²	MongoDB	Postgres

² Government of Canada

³ O uso de MS SQLServer foi deduzido. Embora não obrigatório normalmente o uso de ASP.NET está associado ao uso da base de dados mencionada.

Cliente	ASP	PHP	JSP	PHP	ASPX	JS	JS
Servidor	ASP	PHP	JSP	PHP	ASPX	Node.js	Rails

Das plataformas de acesso público optamos por fazer uma descrição mais detalhada apenas do “British Council”.

2.3.1. British Council

Desenvolvido com recurso ao Drupal⁵, é sem dúvida o exemplo mais completo de um guia online de marcas; foi o principal caso de estudo que serviu de inspiração no desenvolvimento da ontologia⁶ do BAMP e para análise de conteúdos.

“The British Council Brand Guidelines” encontra-se estruturado em cinco assuntos principais: *Brand Framework*; *Tone of Voice*; *Visual Identity*; *Partnerships*; *Downloads*. Em que cada um se subdivide em assuntos coerentes relacionados com cada item.

Exemplo:

Visual Identity → **Our Logo** {Corporate logo/Preferred logo, Positive logo, Negative logo, PMS 295 logo, Symbol and name, Favicon, Do not redraw the logo, Download the logo};

- **Logo exceptions** { UK country versions, Translated logos, Legal exceptions, Tax and status logo exceptions, Friends of the British Council USA, Anniversary logos, Department or programme logos};
- **Logo colourways** {Corporate colours, Single colour logo};
- **Logo size and positioning** {Minimum size, Space around the logo (exclusion zone), Positioning our logo};
- **When to use the black or white version of the logo** {When to use the negative logo, Incorrect positive logo coloured white, Correct use of negative logo, Correct use of positive logo};
- **How to identify a correct logo** {Redrawn logo (circles), Redrawn typeface, Incorrect version of the positive logo, Incorrect version of the negative logo, Incorrect logo examples};
- Downloading the logo;
- Animating the British Council logo;
- Trademark management {Where is the British Council logo currently registered?, What does it cost and who pays for it?, How long does it take?, How long does it last?, What classes do we register our mark in?, What do these classes stand for?, Can we register the word “British Council” to avoid issues with search engines, such as Google?}.

2.3.2. Heineken Company

Apesar de todas as diligências efectuadas, não se conseguiu obter muita informação relacionada com esta plataforma. Ainda assim conseguiu-se visualizar o conceito geral

⁴ O uso de MS SQLServer foi deduzido a partir de diversos anúncios de emprego da empresa relacionados com bases de dados. Essa informação não se encontra em lado nenhum.

⁵ Drupal é uma “Content Management Framework” desenvolvida em PHP e MySQL

⁶ Consultar Anexo

recorrendo ao manual de identidade visual da Heineken⁷ e a *screenshots* disponíveis no *web site* empresa “Addison Group”, responsável pelo desenvolvimento da plataforma.

A “Heineken Company” é constituída por diversas marcas⁸, têm como política a de fornecer a sua identidade visual a empregados e a terceiros que precisem de realizar trabalhos em nome da Heineken. Inferiu-se desta forma que o que os levou a criar esta plataforma foi a centralização numa única plataforma de todas as identidades tendo em vista a uma melhor gestão das marcas possibilitando um melhor controlo sobre os acessos.

Para além da identidade visual, a Heineken integrou no seu website *office branding*, materiais, relatórios externos, entre outros.

A informação relativa à *Visual Identity* está estruturada da seguinte forma: *Logo, Colours, Typeface, Supporting elements, Photography, Correspondence, Presentation, Web, Publications, Signage*.

2.3.3. Brandisty

Como já foi anteriormente referido, este é um projecto recente com o qual a plataforma BAMP se pode assemelhar. Presentemente o Brandisty⁹ já se encontra em pleno funcionamento mas durante a fase inicial do BAMP ainda se encontrava em fase desenvolvimento e testes. Depois de tomarmos conhecimento da existência do projecto foi criada uma conta sob o pretexto de estudar as suas potencialidades, facto que só ocorreu após contactos entre a FBA e a Brandisty. Esta conta permitiu-nos avaliar as funcionalidades e extrapolar um publico alvo tipo, cujo pensamos que sejam pequenas empresas/instituições.

As funcionalidades principais são:

- Possibilidade de um utilizador ter/gerir mais do que uma Marca;
- A marca pode ser acedida por qualquer um que tenha *link* para a marca no Brandisty;
- No momento da transferência de imagens é possível efectuar a conversão de ficheiros EPS em PDF, JPG e PNG;
- Possibilidade de transferir a paleta de cores como “Adobe Swatch Exchange” (ASE) ;
- Três conjuntos de logos: *Primary, Alternate e Icon*. Cada conjunto pode conter mais do que uma imagem;
- Redimensionamento proporcional de Imagens;
- Definição de dois conjuntos de amostras de cores: Primary e Secondary. Cada amostra pode conter múltiplas cores;
- Definição de dois conjuntos de tipografia. Cada conjunto pode conter mais do que uma fonte.
- Possibilidade de carregar um pdf que contenha as *Brand Guidelines*.
- Algumas limitações e falhas:
 - Sistema de cores está limitado a Pantone, CMYK, HEX e RGB;
 - As fontes só são identificadas após carregamento das mesmas para a plataforma. O uso e distribuição de fontes está sujeito a termos e condições de contractos de

⁷ Heineken Company Visual Identity Guidelines

⁸ Curiosidade: algumas marcas portuguesas integram a “Heineken Company” são exemplos disso a Sagres, a Imperial e a Cergal.

⁹ Desenvolvido pela empresa “Tiny Factory”.

licenciamento, o que quer dizer que qualquer um que possua o *link* para a marca tem acesso a fontes que possivelmente não estão licenciadas para esse efeito.

2.3.4. BrandFolder

A par com o Brandistly o seu surgimento no mercado demonstra a necessidade de ferramentas/plataformas que respondam às necessidades das empresas de gestão das suas identidades visuais. Embora existam semelhanças entre plataformas têm públicos-alvos um pouco distintos, dar resposta às necessidades do marketing é o cerne do BrandFolder.

Principais funcionalidades:

- Criação de várias marcas associadas a num único utilizador, com link personalizável (e.g.: <https://brandfolder.com/fba>);
- Gestão dos seguintes activos com enumeração dos campos dos formulários:
 - *Logos*: nome, guias de utilização, carregamento de ficheiros múltiplos, imagem principal;
 - *Imagens*: nome, guias de utilização, carregamento de ficheiros múltiplos, imagem principal;
 - *Documents*: nome, guias de utilização, carregamento de ficheiros múltiplos, ficheiro principal
 - *Media*: url, nome, descrição;
 - *People*: nome, título, email, número de telefone, twitter, linkdIn, biografia, carregamento de ficheiros múltiplos;
 - *Press*: nome, data de publicação, link, descrição, carregamento de ficheiros múltiplos;
 - *Fonts*: escolha de uma web font ou nome da fonte, guias de utilização
 - *Colors*: nome, identificação da cor nos sistemas de cores [hex, rgb, cmyk e pantone], guias de utilização;
 - *Info*: título, área de texto livre que permite o uso de alguns atributos html.

É possível criar múltiplos activos.

- Activos privados partilháveis com os utilizadores inseridos em *People*, esta funcionalidade só está activa na versão paga.
- Todos os conteúdos são públicos, para contas gratuitas, e acessíveis através do link escolhido.

Capítulo 3

Planeamento

Recolhido o conhecimento necessário e analisadas as soluções existentes, este capítulo trata de documentar a preparação feita para a definição da arquitectura e as várias questões de implementação (nomeadamente requisitos e casos-de-uso).

3.1. Requisitos Funcionais

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

3.1.1. Utilizadores

Não pode haver utilizadores repetidos. Nesta análise, presume-se que, para que um utilizador possa fazer alguma coisa, tem de estar autenticado na plataforma.

Todos os utilizadores terão um espaço para alterarem o seu perfil pessoal e palavra-passe, fazem parte desse perfil os seguintes itens: nome, fotografia, morada, contactos telefónicos.

Todos os Utilizadores terão acesso às marcas que lhe estão associadas como utilizador ou administrador. O perfil é o mesmo para as marcas e agências.

3.1.2. Funções de Utilizadores (Users Roles)

Função que um utilizador desempenha num determinado contexto.

3.1.2.1. Descrição

Os próximos pontos são a descrição das funções dos utilizadores, nomeadamente BAMP Admin, BASA, BOA, BOU e BU.

3.1.2.1.1. Administrador da Plataforma (BAMP Admin)

Utilizador responsável pela criação e gestão dos utilizadores BASA e BOA. Como administrador, tem acesso a um conjunto de informação estatística e descritiva sobre a utilização da toda plataforma:

- Marcas;
- Utilizadores e funções;
- Relações e dependências entre utilizadores;
- Relações entre utilizadores e marcas.

3.1.2.1.2. Administrador de Agência (BASA)

Feito a pensar no uso por *ateliers* de *design*, este utilizador *Premium* é criado pelo “BAMP Admin” após acordo (e.g. confirmação de pagamento). Este utilizador é responsável por:

- Criar e gerir as suas marcas e respectivos utilizadores BOU (por norma, clientes do atelier de design (a agência));

- Pela inserção e gestão dos conteúdos nas contas dos seus clientes (BOUs).

Tem ainda acesso a um conjunto de informação estatística e descritiva sobre a utilização da plataforma da agência:

- marcas; utilizadores e funções;
- relações e dependências entre utilizadores;
- relações entre utilizadores e marcas.

3.1.2.1.3. Administrador de Marca (BOA)

Utilizador *Freemium* (Freemium – Wikipédia, a enciclopédia livre, 2014), i.e. livre pelo pacote-base. Regista-se directamente na plataforma. A estes utilizadores ser-lhes-ão atribuídos automaticamente uma marca com o nome pretendido. Poderá criar novas marcas mediante acordo (tipicamente, pagado por estas).

Responsável pela gestão das suas próprias marcas e respectivos conteúdos e permissões de acesso. Este utilizador gere os utilizadores do tipo BU.

3.1.2.1.4. Utilizador Dono de Marca (BOU)

Utilizador criado pelo BASA. Gere utilizadores do tipo BU.

Este utilizador é responsável pelos acessos aos seus conteúdos, i.e., com base nos seus activos, pode atribuir, a utilizadores distintos, acessos diferenciados aos conteúdos da marca.

3.1.2.1.5. Utilizador autorizado (BU)

Utilizador criado pelo BOA e pelo BOU. Este utilizador consulta os activos da marca a que está associado. Pode visualizar os conteúdos que lhe forem atribuídos para consulta.

3.1.2.2. Gestão de Funções

A plataforma terá que disponibilizar uma secção para gestão de utilizadores e suas permissões de acesso à plataforma. Permissões de gestão, de acordo com o especificado nos pontos anteriores, estão descritas na seguinte tabela.

Tabela 2

	BAMP Admin				BASA				BOA				BOU				BU			
	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
BAMP Admin	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
BASA									C	R	U	D	C	R	U	D	C	R	U	D
									<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
BOA													C	R	U	D	C	R	U	D
													<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
BOU																	C	R	U	D
																	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

3.1.2.3. Diagrama de Estados de um Pedido

Um pedido para atribuição de uma função passa por alguns estados, evoluindo conforme o tratamento que o destinatário aplicar a esse pedido.

Embora tenha sido desenhado apenas para funções, os demais pedidos (de agência e quantidade de marcas) entre utilizadores aproveitam este modelo para definirem o seu próprio ciclo-de-vida, embora

O diagrama seguinte permite observar o ciclo de vida de um destes pedidos.

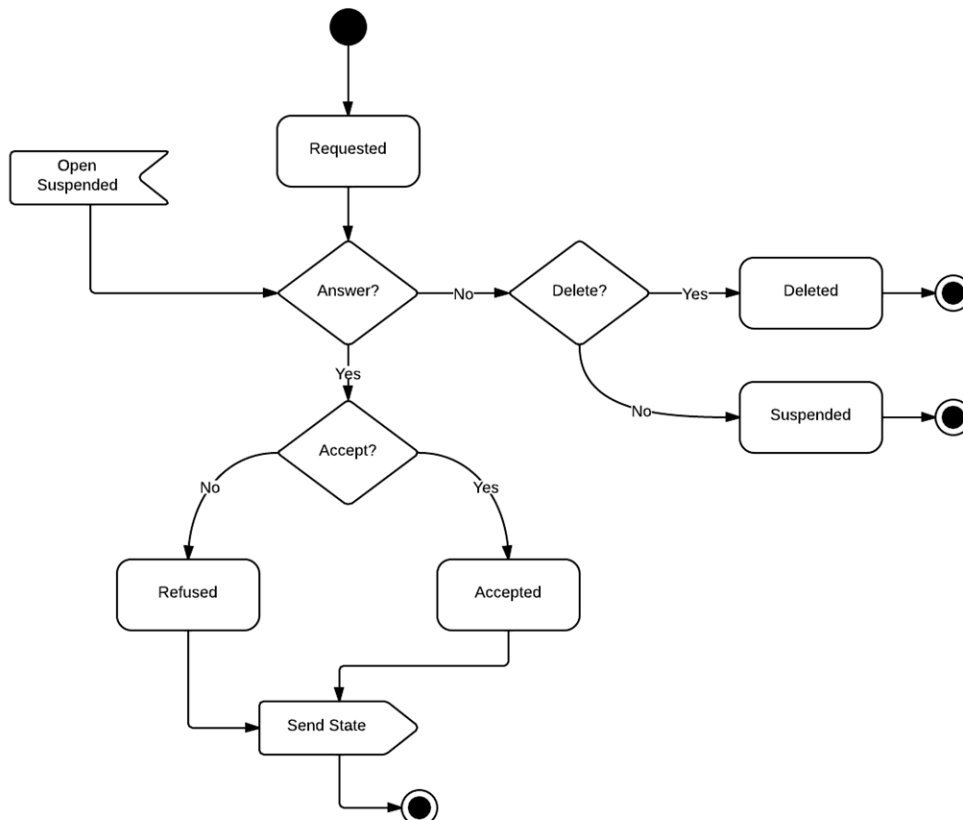


Diagrama 1 – Estados de um pedido.

Um convite para uma função é iniciado no estado “Requested” no momento da sua criação pelo remetente do mesmo. Quando chega “às mãos” do destinatário, é neste estado que ele se encontra e a partir do qual haverá evolução.

Quando questionado quanto ao tratamento que o convite deverá receber, este utilizador tem a hipótese de mandar uma resposta imediata ao remetente (aceitando ou recusando o convite), bem como de prescindir de responder.

Optando por responder, uma notificação automática com a mesma é enviada para o remetente; caso aceite, a acção sugerida no mesmo convite é executada. No caso

3.1.3. Marcas (Brands)

A criação de uma nova marca é feita após a criação de um utilizador tipo BOA ou BOU. Os utilizadores autorizados terão acesso a uma página de gestão de marcas onde poderão consultar, as marcas sob a sua alçada, fazer alterações e eliminar.

Associado à criação da marca, existe um perfil da marca com informação de nome e contactos, que podem ser alterados. Os conteúdos só podem ser preenchidos depois de a marca ser criada.

Tabela 3

	BAMP Admin				BASA				BOA				BOU				BU			
Marca	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

3.1.3.1. Identidade Visual (Visual Identity)

Poderá ser carregado um ou mais documentos pdf de manuais de normas que ficarão disponíveis para transferência. Esses ficheiros podem ser apagados e substituídos por outros.

Tabela 4

	BAMP Admin				BASA				BOA				BOU				BU			
Manual de Normas	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Visual Identity	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Brand Applications	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

3.1.3.1.1. Logo

Ao inserir um logo, há que ter em conta que este tem de ser classificado. Cada classificação terá um texto para enquadrar as escolhas efectuadas.

Inserir Imagem → Classificação de: Tamanho, Profundidade de Cor, Tipo e Medidas

- Formato: Standard, Pequeno e Grande
- Cor Base: Positivo, Negativo e Monocromático
- Tipo de Logo: Horizontal, Vertical, Strapline/Tagline, Símbolo, Texto
- Medidas: Margem de Segurança, Dimensões Mínimas, Ângulo

As Artes Finais poderão ser disponibilizadas para transferência

3.1.3.1.1.1. Excepções Logo

Ao inserir uma excepção tem que se identificar e classificar o tipo de excepção em questão. Cada classificação terá um texto para enquadrar as escolhas efectuadas. As excepções identificadas são as seguintes:

- Web: Favicon e Animações

- Icon;
- Traduções: Selecção da Língua
- Comemorativos: Aniversários, Excepções Legais

As Artes Finais poderão ser disponibilizadas para transferência.

3.1.3.1.1.2. Nota

A plataforma deverá aceitar diferentes tipos de ficheiros para que sejam disponibilizados para transferência. Os ficheiros que contêm as Artes Finais são por norma em formato EPS ou AI. Se for feito o carregamento para o BAMP de um EPS a própria plataforma deve poder converter e disponibilizar, para transferência, a imagem nos seguintes formatos: PDF, JPG e PNG.

3.1.3.1.2. Codificação Cromática (Colourways)

A identificação das cores de uma marca será conseguida através da criação de uma ou mais paletes de cores acompanhado por um texto descritivo.

Uma paleta de cores é constituída por uma ou mais cores. Uma cor é identificada recorrendo a sistemas de cores sendo os principais:

- Pantone;
- RGB;
- Hexadecimal (*Web Colour*);
- CMYK;
- RAL
- 3M.

3.1.3.1.2.1. Aplicações das Cores

Associado com a codificação cromática, existem regras para aplicação dessas cores:

- Sobreposição de Tons (*Tone-on-Tone*)
- Sobreposição por Camadas (*Overlaid*)
- Cor com Fotografia (*Colour with Photography*)
- Gradientes (*Gradient*)

Estas regras são demonstradas através uma ou mais imagens ilustrativas acompanhadas com um texto descritivo da aplicação.

3.1.3.1.3. Tipografia (*Typography*)

Definição de fontes; podem ser definidas uma ou mais fontes e variantes. Isto é conseguido fazendo conjuntos de fontes.

Existe sempre um conjunto de fontes consideradas primárias. Outros conjuntos podem ser definidos (e.g. tipografia secundária e substituta).

- Conjuntos primários e secundários: são iguais, na sua essência, podendo ser classificadas várias fontes como: general, wordmark, strapline/tagline (o utilizador poderá adicionar classificações).
- Conjunto substituição: grupo para situações como: web, desktop e outras excepções (o utilizador poderá adicionar outras classificações).

Cada fonte terá de ter associado um texto informativo sobre o seu uso.

Espaço para carregamento da(s) fonte(s). Para os casos em que não é possível a disponibilização das fontes, por motivos de licenciamento, terá que ser carregada uma imagem com a “font cascade”. Em ambos os casos terá de se disponibilizar campos de identificação da fonte como: nome da fonte, autor, família, estilo, link.

3.1.3.1.4. Imagens (*Imagery*)

Existem galerias bem definidas de imagens a incluir no projecto, são essas as seguintes: *Photography; Patterns; Textures; icons; Identity Elements*. O utilizador poderá adicionar mais. Cada conjunto de galerias terá de ter um texto descritivo associado.

3.1.3.2. Aplicações da Marca (*Brand Applications*)

Considerado mais como um repositório de documento, as aplicações à marca deverão permitir a inserção de ficheiros para transferência. Ao inserir os documentos terão de ser classificados e organizados pelos seguintes itens:

- Impressões (*Print application*): Advertisements; Banners; Brochures; Calendars; CDs and DVDs; Certificates; Flyers; Folders; Intros and outros; Invitations; Name badges; Newsletters; Postcards; Posters;
- Estacionários (*Stationary*): business card; envelop; Letterhead;
- Meios digitais (*Digital Media*): templates; email; signature; newsletter; office app (desktop/online; text editor; spreadsheet; presentation);
- Merchandising: Item

3.1.4. Conteúdos

Elementos de uma marca partilháveis com Bus.

3.1.4.1. Acesso aos Conteúdos e Notificações

Para acesso aos diversos conteúdos, BOA e BOU poderão partilhá-los a utilizadores do tipo BU. Grupos de partilha são criados com um ou mais conteúdos e partilhados a estes utilizadores.

Exemplo 1: partilhar o logo e a tipografia com os utilizadores João e Maria:

- Criar o grupo de partilhas “Logo + Tipografia”;
- Associar o Logo e a Tipografia;
- Caso o João e a Maria não estejam no sistema, convidá-los;
- Associar os mesmos ao grupo “Logo + Tipografia”.

Exemplo 2: grupo de partilhas como grupo de utilizadores:

- Criar o grupo de partilhas “Gráfica”;
- Adicionar ou convidar os utilizadores pretendidos;
- Ir adicionando conteúdos conforme a necessidade.

Exemplo 3: grupo de partilhas como grupo de conteúdos:

- Criar o grupo de partilhas “Natal”;
- Adicionar fontes, cores, e versões de Logo pretendidos;

- Ir adicionando utilizadores conforme a necessidade.

Deverá ser possível apagar grupos; adicionar e remover, tanto utilizadores como conteúdos.

Tabela 5

	BAMP Admin				BASA				BOA				BOU				BU			
Autorização de Acesso	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Grupos de Conteúdos	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Sempre que um utilizador é criado ou lhe são dadas permissões para aceder a novos conteúdos, deve ser notificado ou por email ou notificação na plataforma. O utilizador deverá poder consultar uma listagem das notificações e visualizar a existência de novas.

3.1.4.2. Grupos de Conteúdos

Embora o utilizador possa criar grupos de partilha, existirão grupos de sistema desde que exista pelo menos um item inserido nesse grupo, são esses conteúdos os seguintes: Brand Identity

- Visual Identity
 - Logos
 - Typographies
 - Colourways
 - Colour Applications
 - Imagery
- Brand Applications
- Brand Framework
- Brand Tone of Voice

O objectivo destes grupos é o de facilitar a partilha de conteúdos por utilizadores autorizados a aceder à plataforma e/ou definir os conteúdos de acesso publico.

3.2. Requisitos Não Funcionais

Requisitos de utilização e desempenho.

3.2.1. Escalabilidade

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

3.2.2. 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 imagens é fundamental, irá sempre existir um processamento e informação enviada para o utilizador com o qual teremos sempre de contar.

3.2.3. Segurança

Todos os acessos à plataforma serão feitos após autenticação por email e password encriptada.

Havendo necessidade de fazer a recuperação da acesso, a plataforma gerará um link para aceder a uma página de recuperação da password, link esse que será enviado por email ao utilizador.

3.2.4. Usabilidade

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

3.3. Diagramas de Casos de Uso

Para poupar ao diagrama, serão usados os seguintes “abusos”:

- **Manage <item>**: create, update e delete aquele item;
- **Assign <role>**: convidar para a função; senão existir, envia um email a convidá-lo a registar-se na plataforma e assumir a função.

3.3.1. Identificação dos Actores

São actores do sistema os papéis que a pessoa cumpre na plataforma.

Tabela 6

Actor	Nome	Descrição
Anónimo	Anonymous	Qualquer pessoa que visite o sistema sem estar autenticada.
Utilizador Autenticado	Authenticated User	Utilizador do sistema com autenticação efectuada, independente da(s) função(ões) que desempenhe.
Função	User role	Base de todas as funções (gestão de pedidos e de notificações).
	BAMP Admin	Administrador geral: responde a pedidos de clientes mediante acordo.
	BOA	Do utilizador independente: gere marcas pessoais.
	BASA	Administrador de uma agência: gere marcas e pessoas.

	BOU	Gestor de uma marca: gere conteúdos e utilizadores
	BU	Utilizadores de uma Marca

3.3.2. Anónimo

Sem ser reconhecido pela plataforma.

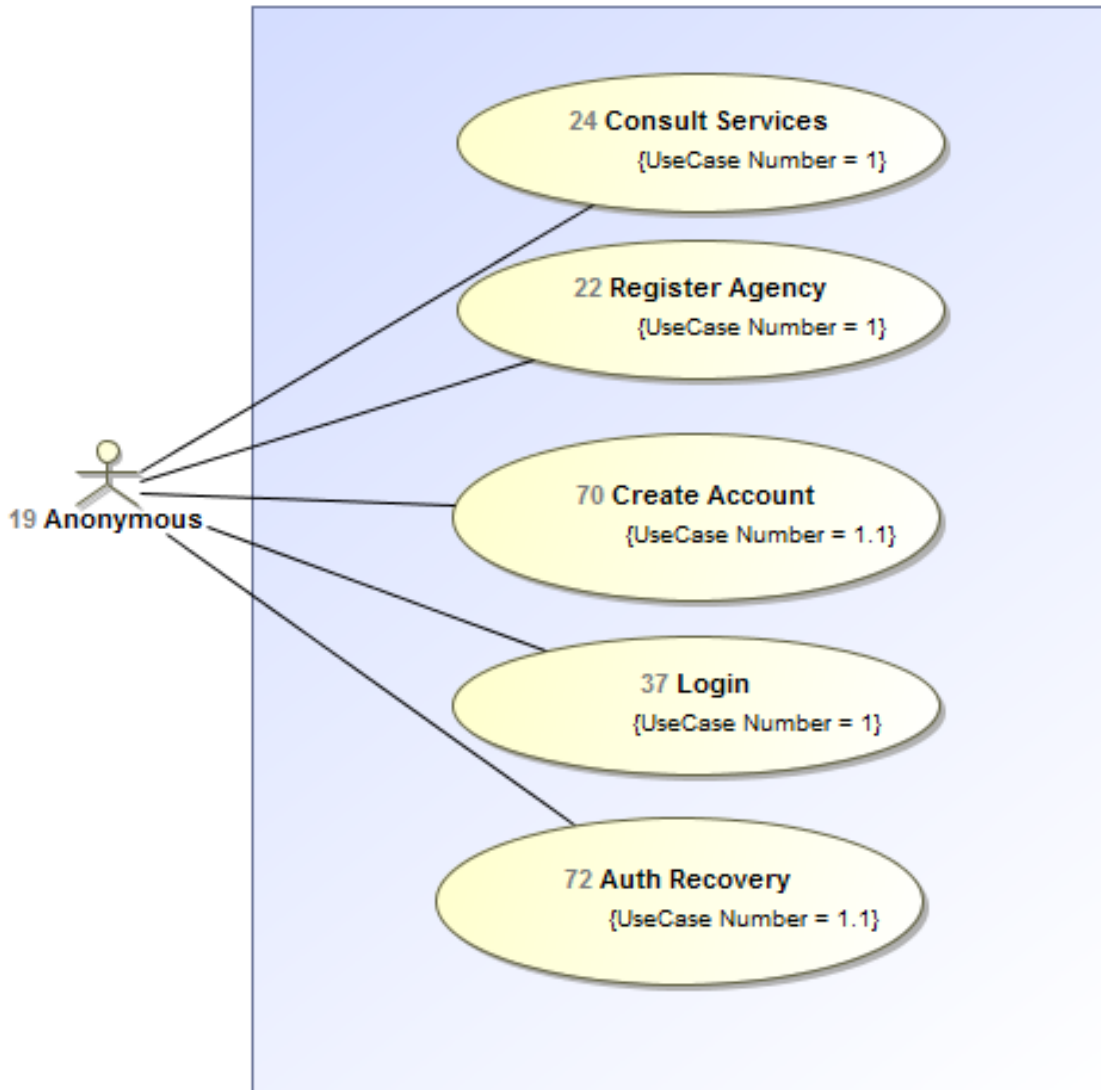


Diagrama 2 – Casos de uso – Actor ‘Anonymous’

Tabela 7

Consult Services	Consultar Serviços	Vê a página de visita com todos os conteúdos considerados pertinentes para os visitantes da plataforma (basicamente, serviços).
Registry Agency	Registar agência	Pede para registar uma agência. Mediante acordo, esta é criada e os dados do visitante transformam-no num BASA.
Create Account	Criar conta	Cria o utilizador e a sua primeira Marca.

Recovery Access	Recuperação de Acesso	Tipicamente por esquecimento de uma palavra-passe, pede, para um email especificado, instruções para voltar a aceder à plataforma.
Login	Iniciar Sessão	Mediante endereço electrónico e palavra-passe, é reconhecido pelo sistema e redireccionado para a sua área de entrada.

3.3.3. Utilizador Autenticado

O utilizador autenticado terá uma ou várias funções na plataforma. A maioria dos casos de uso referem-se a uma função em específico (e a um contexto em concreto), enquanto estas não têm enquadramento em qualquer função ou contexto.

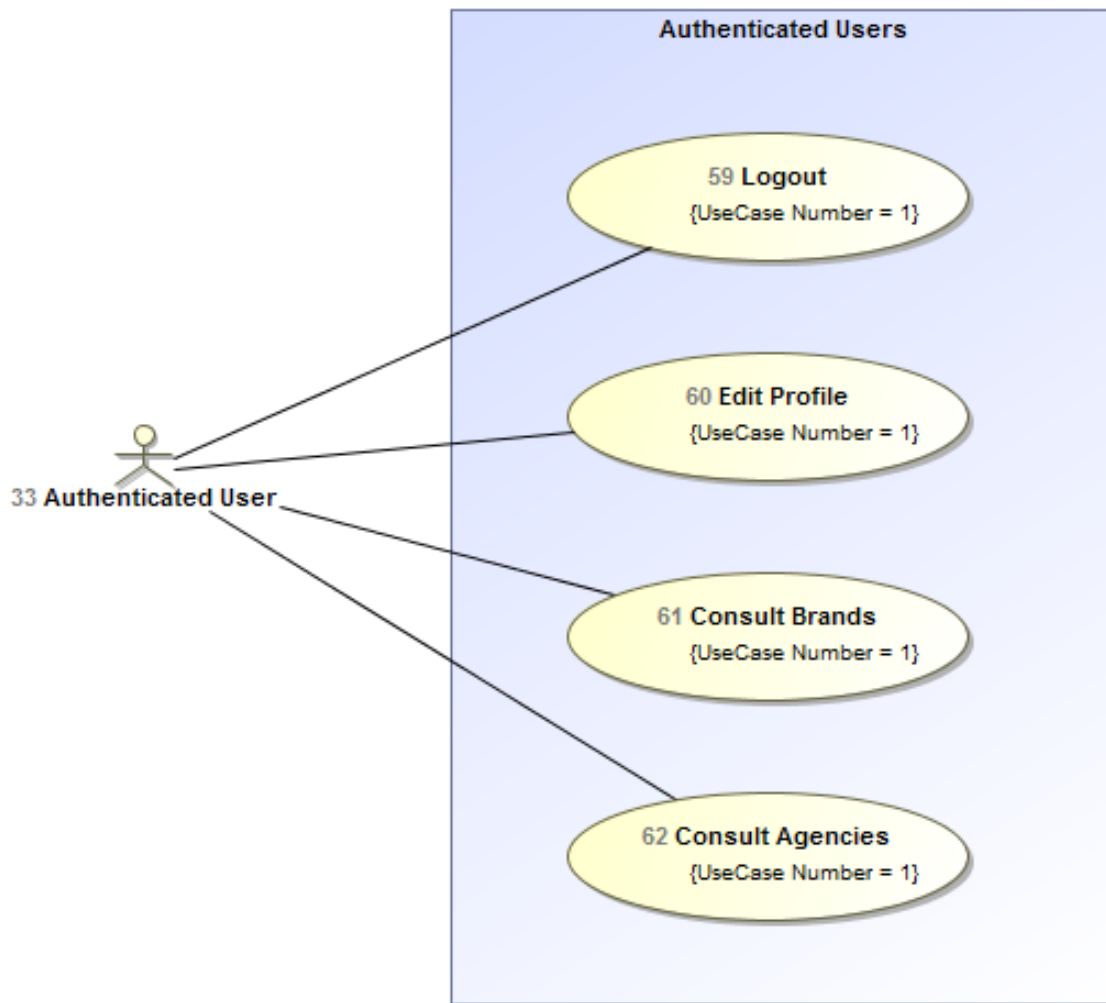


Diagrama 3 – Casos de uso – Actor ‘Authenticated User’

Tabela 8

Logut	Terminar sessão	Fazendo com que a plataforma deixe de o reconhecer e assumindo o papel (actor) Anonymous.
Edit profile	Editar dados pessoais	Preencher ou editar dados pessoais ou contactos.
See	Listar	Tipicamente, um utilizador só terá uma agência (seja BASA,

Agencies	agências	BOU ou BU). No entanto, nada impede que não tenha funções em mais que uma. Na listagem, pode entrar no contexto de uma em específico.
See Brands	Listar marcas	Listar marcas sobre as quais tem funções. Selecionando uma, entra no contexto dela.

3.3.4. Funções

As funções que um utilizador pode assumir num sistema são: BAMP Admin, BASA, BOA, BOU e BU. São bastante distintas, embora tenham parte comum. Por isso começa-se por definir os casos-de-uso comuns num (pseudo) actor-base representado no diagrama com o nome “User Role”.

3.3.4.1. Genérico

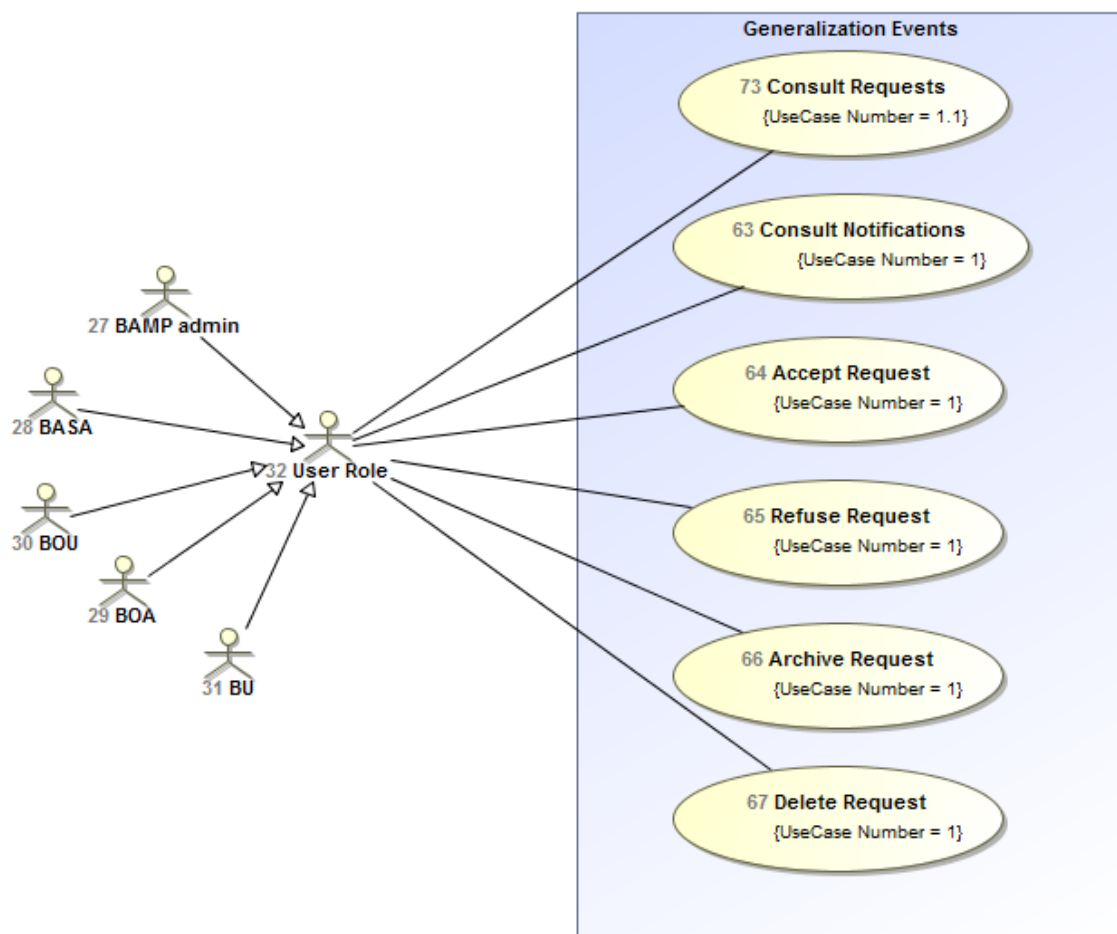


Diagrama 4 – Casos de uso – Actor genérico ‘User Role’ e suas especializações

Tabela 9

See Notifications	Listar notificações	Nomeadamente, respostas a pedidos e avisos de partilha.
See Requests	Listar pedidos	<ul style="list-style-type: none"> • Convites feito ao actor de uma nova função; • Pedido (ao BAMP Admin) de:

			<ul style="list-style-type: none"> ○ Mais marcas (fora do contexto de agência); ○ Inscrição de Agência.
Handle Request	Accept	Aceitar	Envia a resposta “Aceite”. A acção sugerida no pedido é despachada.
	Refuse	Recusar	Envia a resposta “Recusado”. A acção sugerida é anulada.
	Suspend	Suspender	Não há resposta: o tratamento do pedido é adiado.
	Delete	Apagar	Não há resposta: o pedido deixa de existir.

3.3.4.2. BAMP Admin

Responde aos pedidos a ele dirigidos, tipicamente após confirmação de cumprimento de acordo (e.g. pagamento):

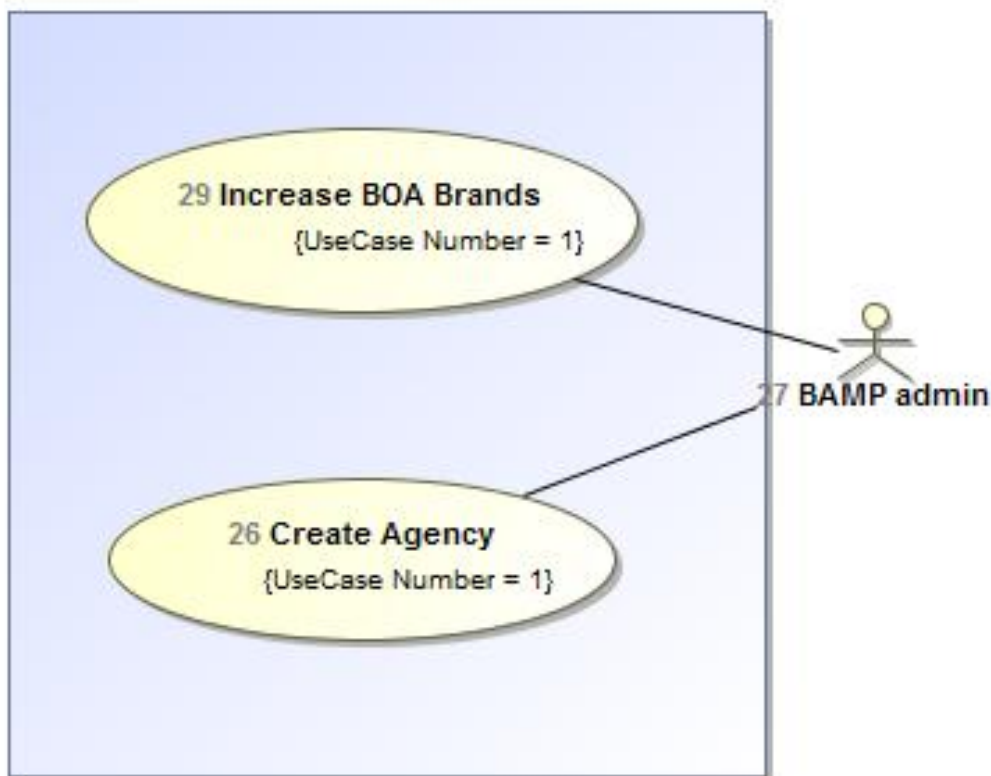


Diagrama 5 – Casos de uso – Actor ‘BAMP Admin’

Tabela 10

Increase BOA Brands	Aumentar número de marcas pessoais	Uma agência pode ter muitas marcas. Fora do contexto de agência, um qualquer utilizador só pode ter uma, até que “pague” por mais.
Create agencies	Listar pedidos	<ul style="list-style-type: none"> • Convites feito ao actor de uma nova função; • Pedido (ao BAMP Admin) de:

		<ul style="list-style-type: none"> ○ Mais marcas (fora do contexto de agência); ○ Inscrição de Agência.
--	--	---

3.3.4.3. Agentes de Partilha (BASA, BOU, BOA e BU)

Dentro do contexto de Marca, conteúdos podem ser criados, partilhados e consultados. Se a marca for de uma agência, o BASA cria e o BOU partilha. Se for pessoal, o BOA cria e partilha. O actor que consulta (nos dois casos) chama-se BU.



Diagrama 6 – Casos de uso – Actores ‘BASA’, ‘BOU’, ‘BOA’ e ‘BU’

3.3.4.3.1. Manage Brand

Tabela 11

Nome	Gerir marcas. Concreto para o BOA e abstracto para o BASA, pois terá ainda de designar um BOU
Actores	BASA e BOA

3.3.4.3.2. Assign BOU

Tabela 12

Nome	Convidar alguém para o designar como BOU de uma Marca de Agência.
Pré-condição	Marca já criada
Actores	BASA

3.3.4.3.3. Manage Agency Brand

Tabela 13

Nome	Concretização da criação de marca de agência. Inclui dois passos: criação de marca e designação de BOU.
Actores	BASA
Used cases	<ul style="list-style-type: none"> • Manage Brand • Assign BOU

3.3.4.3.4. Manage Content

Tabela 14

Nome	Gerir conteúdos a serem partilhados com BUs.
Actores	BASA e BOA

3.3.4.3.5. Share content

Tabela 15

Nome	Partilhar conteúdos previamente criados com BUs.
Actores	BOU e BOA

3.3.4.3.6. Assign BU

Tabela 16

Nome	Pedir a um utilizador que seja BU dentro de uma marca.
Actores	BOU e BOA

3.3.4.3.7. Request Additional Brands

Tabela 17

Nome	Um BOA começa com uma marca; se quiser mais, terá de as pedir mediante acordo (leia-se pagamento).
Actores	BOA

3.3.4.3.8. Consult Content

Tabela 18

Nome	Consultar conteúdos partilhados para si.
Actores	BU

3.4. Planeamento do Projecto

Nesta secção é apresentado o diagrama Gantt, construído para ilustrar o avanço das diferentes etapas do projecto. Como se poderá observar os intervalos de tempo representando o início e fim de cada fase, iniciam-se apenas no mês de Fevereiro. Este lapso temporal entre a data de início do projecto, Setembro de 2013, e o início temporal do diagrama explica-se pelo seguinte motivo:

Durante a primeira metade deste projecto todo o projecto foi pensado e planeado a pensar no uso do *Content Management Framework* (CMF) Drupal, escolhida pelo estagiário e aceite pelos orientadores como solução ao qual se recorreria como base para a implementação do projecto, no entanto a sua utilização foi descartada por receio de falta de complexidade técnica. Motivo pelo qual neste documento não se fazer menção nem às análises nem às estratégias seguidas até esse ponto cujo tinham sido pensadas para o uso do Drupal.

Este evento, foi bastante prejudicial para o decorrer do projecto obrigou a uma reestruturação completa do projecto.

Aproveitado da primeira metade do projecto foram todas as análises e estudos que constam deste documento até este ponto, nomeadamente o Estado da Arte, Requisitos e Diagramas de Casos de Uso.

	📌	Nome	Início	Fim	Recursos	Fev 2 - Fev 8 '14			
						D	S	T	Q
1	📌	📁 Elaboração do Relatório de Estágio	13/02/2014	26/07/2014					
2	📌	-INTRODUÇÃO	13/02/2014	20/02/2014					
3	📌	-CONCEITO E ESTADO DA ARTE	20/02/2014	08/03/2014					
4	📌	- ARQUITECTURA	10/03/2014	25/03/2014					
5	📌	-TESTES	07/05/2014	06/06/2014					
6	📌	-CONCLUSÃO	06/07/2014	26/07/2014					
7	📌	📁 Análise de Requisitos	13/02/2014	22/02/2014					
8	📌	- identificação de requisitos, casos de uso doc	13/02/2014	22/02/2014					
9	📌	📁 Especificação Funcional	22/02/2014	20/03/2014					
10	📌	- tecnologias	22/02/2014	25/02/2014					
11	📌	- DB Model	25/02/2014	10/03/2014					
12	📌	- arquitetura (modulos, ...)	10/03/2014	17/03/2014					
13	📌	- maquete / wireframe	17/03/2014	19/03/2014					
14	📌	- redação de documento	10/03/2014	20/03/2014					
15	📌	📁 Design	20/03/2014	05/04/2014					
16	📌	- templates (projeto grafico)	20/03/2014	30/03/2014	FBA Designers				
17	📌	- implementação de layouts	28/03/2014	05/04/2014					
18	📌	📁 Implementação dos modulos	07/04/2014	18/06/2014					
19	📌	tarefas anteriores	07/04/2014	08/05/2014					
20	📌	📁 Marca	09/05/2014	09/06/2014					
21	📌	Ficha da marca	09/05/2014	09/05/2014					
22	📌	📁 Identidade Visual	09/05/2014	23/05/2014					
23	📌	Logos	09/05/2014	13/05/2014					
24	📌	tipografia	14/05/2014	16/05/2014					
25	📌	Colourways	16/05/2014	21/05/2014					
26	📌	imagens	20/05/2014	23/05/2014					
27	📌	Aplicações da Marca	26/05/2014	30/05/2014					
28	📌	Grupos de Conteudo	02/06/2014	06/06/2014					
29	📌	Utilizadores/acessos	09/06/2014	09/06/2014					
30	📌	Gestão de Perfil	09/06/2014	09/06/2014					
31	📌	Pagina de Entrada/estatisticas	10/06/2014	10/06/2014					
32	📌	📁 ADMINISTRAÇÃO	11/06/2014	13/06/2014					
33	📌	Gestão de Utilizadores	11/06/2014	13/06/2014					
34	📌	📁 Area Publica	16/06/2014	18/06/2014					
35	📌	Pagina BAMP	16/06/2014	18/06/2014					
36	📌	Directório de Marcas	16/06/2014	18/06/2014					
37	📌	📁 Testes	02/06/2014	20/06/2014					
38	📌	- plano de testes	02/06/2014	10/06/2014					
39	📌	- execução / relatório de execução	10/06/2014	20/06/2014					
40	📌	📁 Documentação	16/06/2014	27/06/2014					

Diagrama 7 – Gantt, fases do projecto

Este segundo plano de trabalhos não conta com nenhuma reestruturação feita à posteriori pelo que as fases aqui descritas não espelham de forma concreta o trabalho realizado. O diagrama foi usado apenas como um guia e não deve ser analisado como um mapa concreto do projecto.

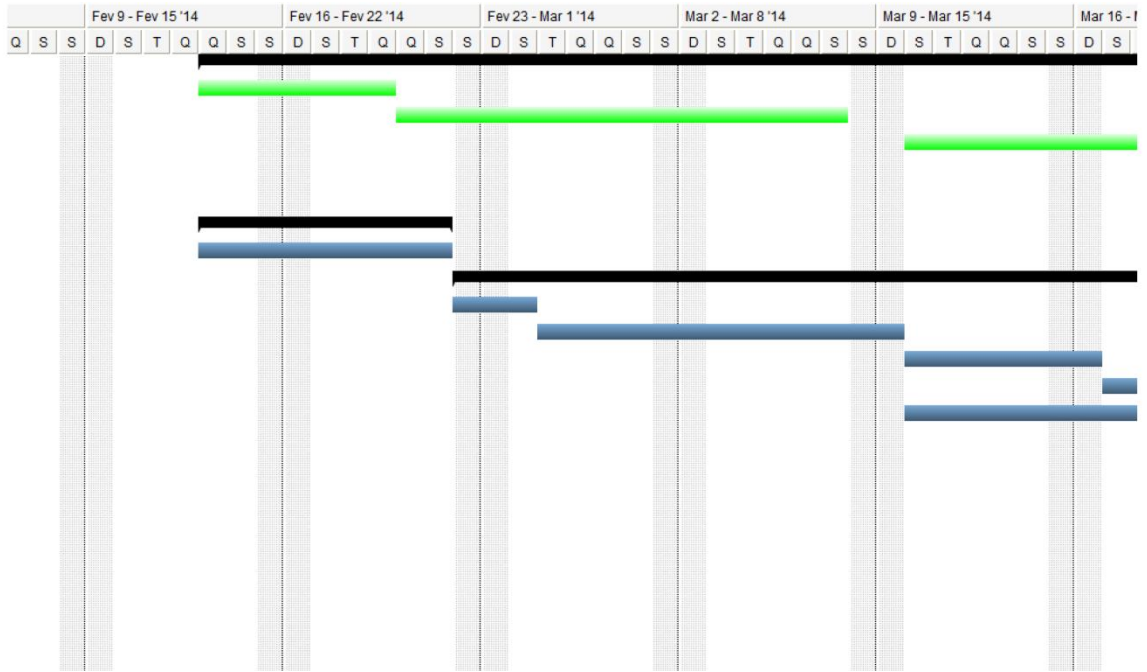


Diagrama 8 – Gantt, 13 Fev. a 14 Mar

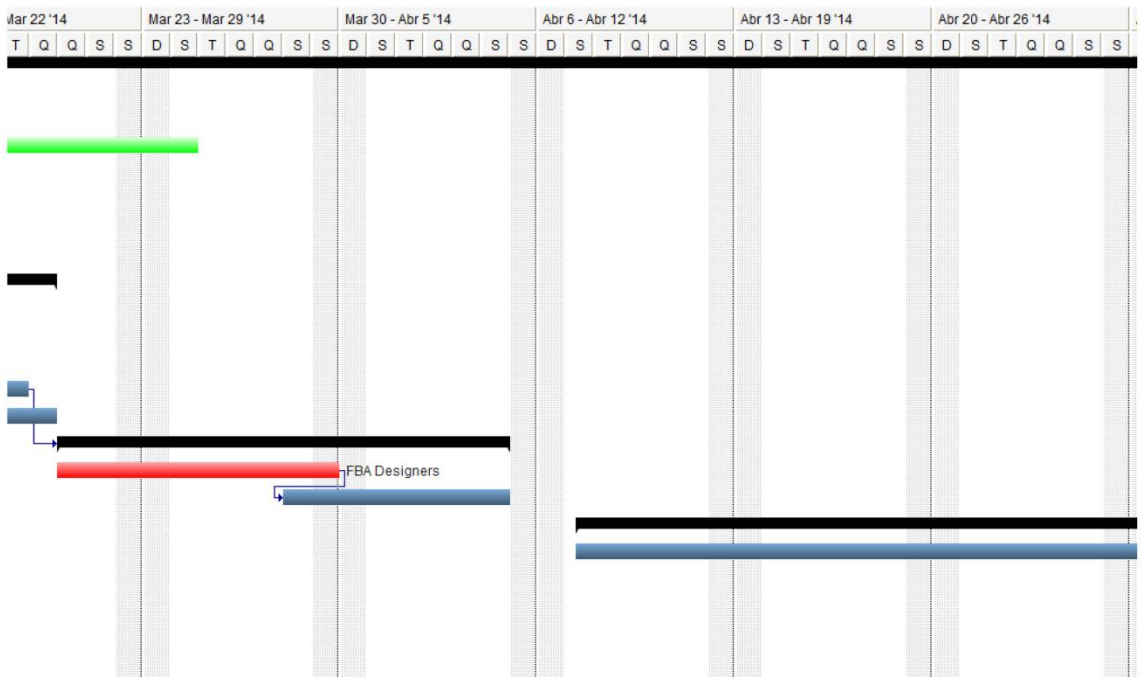


Diagrama 9 – Gantt, 14 Mar. a 26 Abr.

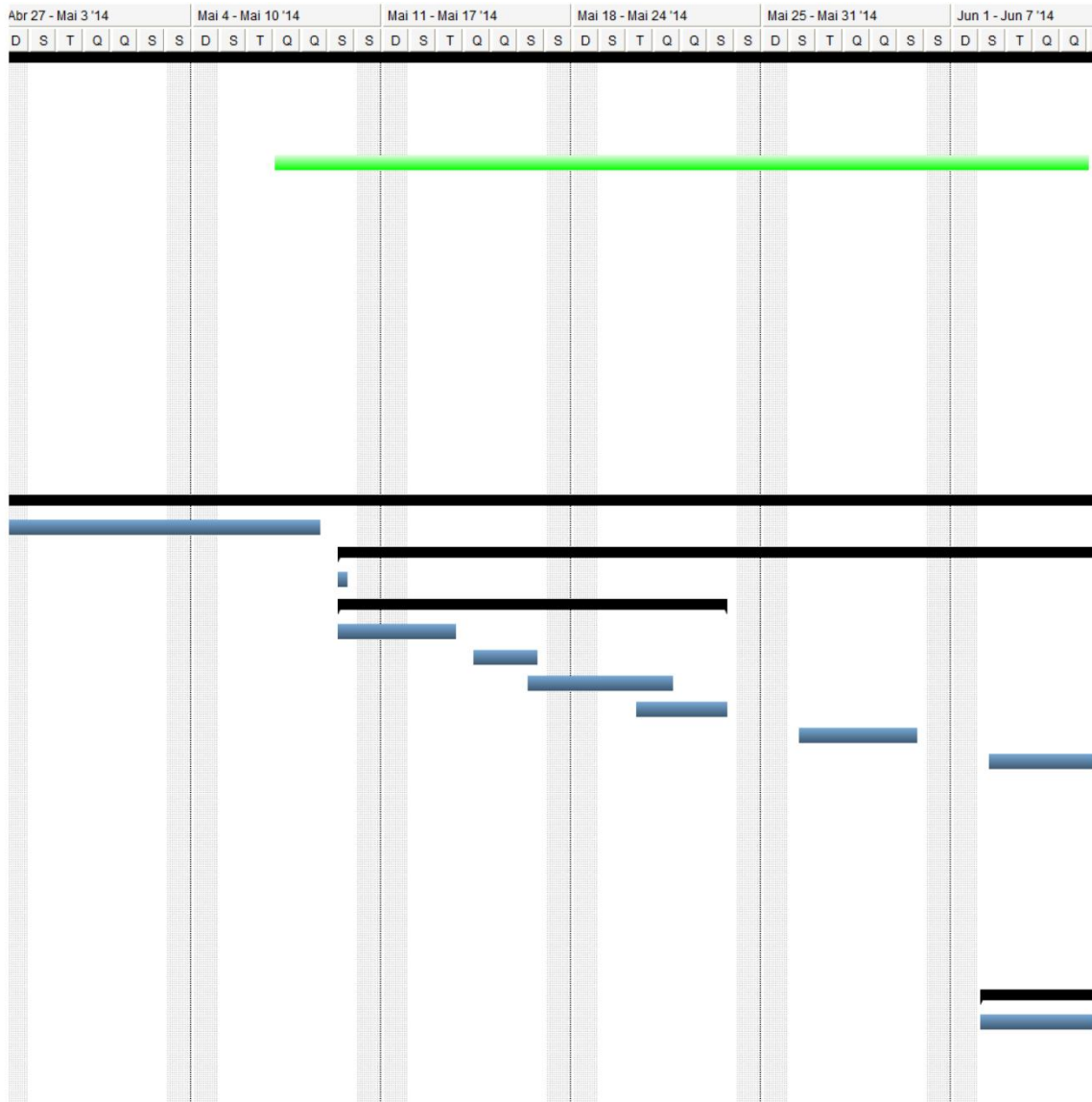


Diagrama 10 – Gantt, 14 Mar. a 25 Jun.

A falta de percepção do que acarretaria a reestruturação do projecto fez com que o trabalho realizado, cujo está descrito nos capítulos seguintes, se perlongasse por mais tempo do que aquele esperado e ao mesmo tempo obriga-se a uma selecção do que viria ser implementado.

Capítulo 4

Arquitetura

Este capítulo apresenta as definições da plataforma quanto a sistemas, *frameworks* e ambiente de trabalho.

4.1. Arquitectura do Sistema

A arquitectura da plataforma baseia-se no paradigma de computação **cliente-servidor**, mais especificamente na sua versão web, como se ilustra na figura seguinte.

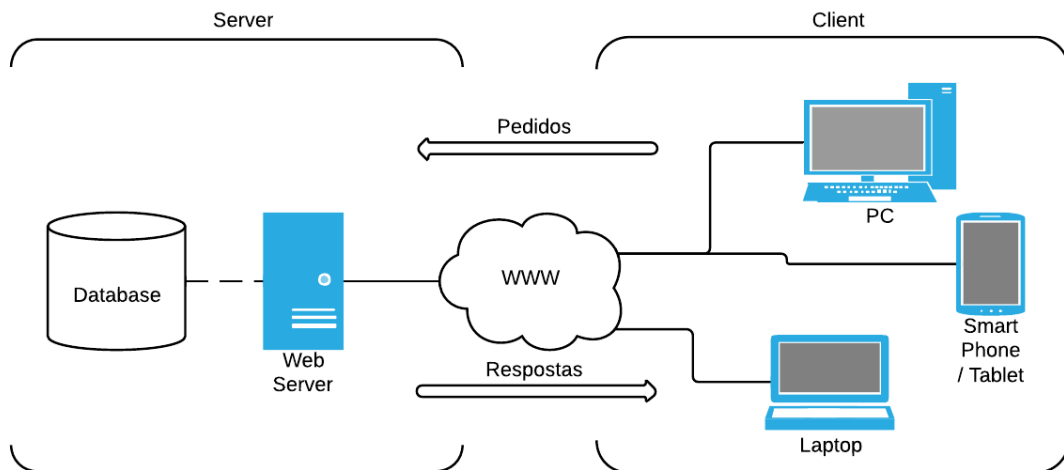


Figura 1 – Cliente Servidor

4.1.1. Cliente

Sendo web, o cliente pode ser executado em qualquer sistema operativo, ficando para o capítulo “4 - Implementação“ a adaptação do mesmo para os vários dispositivos (móveis, portáteis e PCs). Ter de correr em vários navegadores é uma questão do servidor, pois é ele que trata da compatibilidade dos mesmos.

4.1.2. Servidor

Conhecidas as limitações, ao nível das linguagens a serem utilizadas na implementação do projecto, a escolha “natural” de tecnologias a serem usadas do lado do servidor foi de um conjunto de soluções LAMP (*Linux+Apache+MySQL+PHP*), que se apresenta de seguida (com cada versão utilizada no desenvolvimento):

- Linux: Kubuntu 14.04,
- Apache: 2.4.7,
- MySQL: 5.5.37,

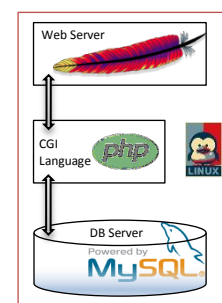


Figura 2 – LAMP Stack

- PHP: 5.5.9 com suporte activo:

Quaisquer alternativas, para viabilizar o desenvolvimento de aplicações web de propósito geral, de alta disponibilidade e de alto desempenho, poderiam ter vingado (e.g. Apache poderia ter sido substituído, ou usado em conjunto, pelo Nginx; bem como o uso duma solução da Microsoft com o servidor IIS (*Internet+Information+Services* - ainda que o acrónimo fosse WIMP).

Justificações para a solução tomada são:

- Uso exclusivo de soluções livres, o que exclui as da Microsoft;
- Os consumos de memória nessa fase foram considerados uma preocupação secundária;
- Experiência pessoal com a solução LAMP escolhida.

Os componentes escolhidos são ainda os mais populares de cada leque de opções, arrastando assim uma quantidade enorme documentação tanto formal, como na forma de resultados da experiência e discussão da comunidade dos seus inúmeros utilizadores e programadores.

Ainda nas mais-valias recolhidas, pela popularidade da *solution stack* escolhida, é a existência de vários modelos de instalação e configuração da solução completa (em detrimento de cada componente em separado e conseqüente integração), bem como da quantidade de utensílios de manipulação, optimização e consulta disponíveis.

4.1.2.1. *Linux*

As razões para usar Linux nas fases de desenvolvimento e teste foram:

- SO do servidor de testes;
- Obediência à *solution pack* escolhida;
- Se funciona em Linux, cujo sistema de ficheiros é *case-sensitive*, deverá funcionar em Windows (que é *case-insensitive*) - à data do projecto, não era conhecido o SO de publicação da plataforma.

4.1.2.2. *Apache*

Históricamente responsável por desempenhar um papel chave no crescimento da WWW, de acordo com um estudo de mercado da Netcraft (Netcraft, 2014), os servidores Apache são, ainda, globalmente os mais usados. Trata-se de um software livre e de código aberto, distribuído sob uma licença Apache, que é desenvolvido e mantido por uma comunidade aberta sob auspício da *Apache Software Foundation*.

Tecnicamente, o Nginx consome menos memória que o Apache, por lidar com pedidos Web através do conceito de “event-based web server” (o Apache é baseado no conceito “process-based server”). Não são necessariamente “concorrentes”, pois Apache e Nginx podem trabalhar juntos: é possível diminuir o consumo de memória do Apache fazendo os pedidos passarem primeiro pelo Nginx; desta forma, o Apache não precisa servir arquivos estáticos, e pode depender do bom controlo de cache feito pelo Nginx.

O custo da dependência de processos do Apache pode ser contrariado, se modelos e paradigmas de programação forem usados (e foram neste projecto, pelo menos parcialmente) para a produção de protótipos mais eficientes, embora não seja uma questão fulcral deste projecto. São exemplos:

- AJAX e WebSockets: clientes assíncronos, dissipando os efeitos da latência do servidor;
- Cache no cliente: reduzindo pedidos idempotentes e consumo do servidor;
- Validação no cliente: reduzindo pedidos inválidos;
- Serviços PHP: passando a produção da vista para Javascript, reduzindo as funções do PHP na produção de vistas e ajustando a sua jurisdição à transferência de dados.

4.1.2.3. *MySQL*

Embora nascido para ser livre, desde a sua aquisição pela Oracle em 2009 que tal estatuto vem sendo comprometido (ou pelo menos posto em causa). No entanto, o seu cumprimento com os standards e a preocupação deste projecto em não se afastar dos mesmos (e.g.: para extensões proprietárias do MySQL) permitirá que uma eventual substituição do motor não seja pesada.

4.1.2.4. *PHP*

Trata-se de uma linguagem de programação web server-side multi-plataforma (compatível com “todos” os navegadores). Embora as alternativas notáveis sejam muitas (e.g.: ASP.NET, C, C++, CMFL, Haskell, Java, Pascal, Perl, Python, Ruby, Scala), a comparação mais popular com alternativas, opõem-no ao Java (Servlets + JSP) e à plataforma .Net (da Microsoft: ASP + alguma linguagem da plataforma, e.g. VB e C#), com vantagens e desvantagens para todos os lados, sendo estas as mais repetidas.

Vantagens:

- Fácil aprendizagem: ser fácil é irrelevante, por já ser conhecida e haver experiência do estagiário;
- Acesso a dados: importante mas irrelevante, uma vez que os outros também acedem;
- Velocidade e robustez: importante, embora a diferença venha a minimizar-se;
- Multiplataforma: fundamental: é um requisito;
- Código fonte aberto: gratuito e ser livre também são requisitos;

Desvantagens:

- Compatibilidade entre versões: funções e funcionalidades que deixam de existir e mudam de nome podem causar futuras dores-de-cabeça, embora não seja um problema para agora;
- Documentação incompleta: nas novidades, pelo menos, embora não exploradas pelo projecto;
- Suporte a datas: implementa, mas de forma menos esclarecida ou limpa como .Net (e o Java desde a v8);
- Segurança: PHP é interpretado - os especialistas invocam que a pré-compilação nos outros é mais segura;
- Aplicativos de Servidor: vantagem dos outros, com a diferença a minimizar-se;
- Serviços web: como o PHP a apanhar os outros.

Ao contrário do motor-de-dados, esta opção não é fácil de se revogar dado a abissal diferença para com as alternativas. No entanto:

- É um requisito;
- A opção .Net está fora de questão
- O custo da aprendizagem da solução Java pode constituir um largo entrave.

4.2. Frameworks

Porque não vale a pena inventar a roda, a escolha de uma *framework* era essencial. Descartada a opção de uso do Drupal, houve a necessidade de encontrar uma solução que auxiliasse a fase de implementação. O estudo efectuado, sobre o assunto, revelou a existência de inúmeras *frameworks* das quais se enumeram aqui algumas apenas para enquadramento:

- Codeigniter
- Symfony
- Yii
- PHPCake
- Laravel

A escolha inicial recaiu no Codeigniter que, pelos comentários da comunidade, revelava-se uma *framework* de uso intuitivo, bem documentada e de fácil adaptabilidade. No entanto, à data da escolha da *framework*, já não sofria nenhuma actualização desde 2012 e ameaçava ser descontinuada¹⁰, motivo pelo qual foi posto de parte o seu uso. No entanto, acrescentou requisitos à escolha da *framework*: ser orientado a objectos e à arquitectura MVC. Esta procura levou à constatação de que parte da comunidade do Codeigniter estava a optar pelo Laravel.

4.2.1. Laravel

Laravel é uma *framework* desenhada para o desenvolvimento em PHP de aplicações web baseadas no paradigma MVC. É uma solução livre e de código aberto publicada sob licença do MIT.

“laravel attempts to take the pain out of development by easing common tasks used in the majority of web projects, such as authentication, routing, sessions, and caching.”
(Laravel - The PHP Framework For Web Artisans., 2014)

Os seus recursos e o facto de estar altamente bem documentada torna a curva de aprendizagem mais rápida que outras *frameworks* semelhantes. Estes motivos a par com a crescente comunidade de utilizadores motivaram a escolha da mesma.

Dentro dos seus recursos nomeiam-se alguns dos mais importantes:

- Composer nativo: instalação e gestão de dependências;
- Suporte para MySQL, Postgres, SQLite e SQL Server;
- Ico Container: gestão de dependências de classes;
- Segurança: Autenticação de Utilizadores, Protecção de Rotas;
- Blade: motor para *templating*;
- Construtor de Consultas (en: queries);
- Eloquent ORM: implementação *Active Record* para trabalhar com bases de dados relacionais.

Para este projecto foi usado o Laravel 4.1¹¹.

¹⁰ Codeigniter 2.2.0, nova versão, foi lançada a 5 de Junho de 2014.

¹¹ À data da escrita deste documento a última versão estável do Laravel é a 4.2

4.2.2. Front end Frameworks

Constatando-se que existia a impossibilidade, por parte da FBA, de entregar os *templates* em tempo útil, foi necessário arranjar um meio rápido, eficaz e que cumprisse com os requisitos do projecto, nomeadamente ser uma aplicação *responsive*. Por sugestão do orientador da FBA, duas *frameworks* foram colocadas para análise: Foundation e Bootstrap. Embora o leque de *frameworks* existentes seja vasto, o facto do tempo disponível não ser muito, levou a que mais nenhuma *framework* tenha sido considerada.

4.2.2.1. Análise das frameworks

Ambas as *frameworks* pareciam adequar-se às necessidades do projecto. A tabela seguinte espelha algumas principais informações recolhidas sobre ambas:

Tabela 19

	Bootstrap	Foundation
Versão Estável (data)	v3.1.1 (13 Fev 2014)	v5.2.2 (4 Apr 2014)
Licença	MIT License	MIT License
Suporte Oficial de Browsers	Chrome (Mac, Windows, iOS, and Android) Safari (Mac and iOS only) Firefox (Mac, Windows) Opera (Mac and Windows) IE8+	Desktop: Chrome, Firefox, Safari, IE9+ Mobile: iOS (iPhone), iOS (iPad), Android 2, 4 (Phone), Android 2, 4 (Tablet), Windows Phone 7+, Surface
Boa Documentação	Sim	Sim
Grelhas e Responsividade	Sim	Sim
UI e Widgets	Sim	Sim

Nota: Para mais informação sobre User Interfaces e Widgets suportados por ambas as *frameworks* consultar tabela em anexo.

Durante este processo, foram efectuadas algumas experiencias com as *frameworks* que resultaram na descoberta de temas e *templates* livres e gratuitos.

4.2.2.2. Escolha da Framework e Templates

Analisadas as *frameworks*, encontrando-se ambas candidatas a serem seleccionadas, a escolha foi efectuada recorrendo ao método menos científico de todos. Foi referido no ponto anterior a existência de *templates* e temas gratuitos, dois dos que captaram maior atenção eram para o Bootstrap 3, o que acabou por se revelar a gota de água na escolha da framework.

Os *templates* em questão são:

- **Stylish Portfolio:** base para a página pública com menu de autenticação;
- **SB Admin:** base para a página de gestão de aplicação, visualização e consulta de conteúdos.

Colocados os *templates* no servidor, procedeu-se um rápido estudo dos mesmos e subsequente adaptação à visão idealizada para a plataforma e adaptação dos wireframes¹² concebidos durante a análise de requisitos.

Importante referir que, com os *templates*, apareceu uma “*framework*”, até então desconhecida pelos elementos deste projeto, chamada “**Font Awesome**”.

4.2.2.2.1. Font Awesome

“*The iconic font and CSS framework*”. O trocadilho inerente a esta frase, que é usada como lema, é explicativo da sua utilidade e funcionalidade. No entanto parece-me um pouco excessivo o uso do termo *framework*.

Inicialmente desenhada para o Bootstrap em código aberto e GPL Friendly, actualmente vai na sua versão 4.1, trata-se de uma biblioteca de ícones vectoriais customizáveis por CSS.

Escolhida a *framework* e *templates* a usar no desenvolvimento do projecto, passa-se em seguida a uma descrição em mais detalhe da mesma focalizada nos elementos, estilos, componentes e scripts incluídos.

4.2.2.3. Bootstrap

É uma *framework* ou um conjunto de ferramentas de software livre, baseadas no paradigma Mobile First, para desenhar Páginas e Aplicações Web. Desenvolvida por Mark Otto and Jacob Thornton na Twitter como uma *framework* para encorajar a consistência por todas as ferramentas internas, foi lançado como open source em 2011 sob uma licença MIT.

Contém *templates* em HTML e CSS para tipografia, formulários, botões, navegação, entre outros componentes de interacção, bem como extensões de javascript.

Trata-se de uma *framework* muito bem documentada o que facilitou o seu estudo e incorporação no projecto. A documentação encontra-se contextualizada nos seguintes pontos dos quais se foca as principais componentes a ser usadas:

- CSS: Grid System; Typography; Forms; Buttons e Images
- Components: Glyphicons; Dropdowns; Button Groups; Button Dropdowns; Input Groups; Navs; Navbar; Pagination; Labels; List Group;
- Javascript: Modal; Dropdown; Tab; Button; Collapse; Carousel

Recorreu-se extensivamente a estes componentes para construir os *templates* semi-dinâmicos, finais. Entenda-se por semi-dinâmicos o facto do Bootstrap incluir componentes em javascript na forma de plugins de jquery, que lhe confere a parte dinâmica que aqui se fala.

4.3. Ambiente de Desenvolvimento

As seguintes aplicações foram usadas neste projecto, algumas de cuja presença foi constante (e.g.: MySQL Workbench) enquanto outras foram usadas no início e gradualmente abandonadas em detrimento de outras (e.g.: NetBeans → PHPStorm):

Base de Dados:

¹² Consultar o Anexo 2.

- MySQL Workbench 6.1 (*Community Edition - open source*): Modelação e gestão

Programação:

- SublimeText 2 (*freeware “ma non troppo”*) com plugin blade syntax: Laravel templating
- Net Beans 8 (*freeware*): PHP e HTML
- PHPStorm 7 (*trial*): PHP, HTML e Javascript
- IntelliJ IDEA 13 (*Community Edition - freeware*): Java

Edição de Imagem:

- GIMP 2.8 (*free/opensource*)

Edição de Texto:

- Kingsoft Writer (*freeware*)
- Google Docs (*free*)
- Microsoft Word 2013

Gestão de Projecto:

- Gantter for Google Drive (*free*): diagrama das etapas do projecto

Desenho de Maquetes:

- MockFlow Wireframe Pro (*Primum Edition*, disponibilizado pela FBA):

Diagramas UML:

- MagicDraw 17.4 (*Trial*): Casos de Uso
- Lucidchart Diagrams for Google Drive (*free version*)

Capítulo 5

Implementação

Este capítulo trata da fase de implementação não apenas do proposto mas também de um gerador para Modelos Laravel. Abordar-se-ão com o detalhe necessário escolhas, tecnologias e processos de implementação.

5.1. Modelação de Dados

Estruturação da informação conforme há-de ser utilizada, tanto pela BD como pela aplicação.

5.1.1. Nomenclatura e Diagramas

Parece haver, *a priori*, duas arquiteturas a modelar: a da BD e as classes do PHP. Os *standards* apresentados pela comunidade são, respetivamente, diagramas ER e de diagramas UML de classes.

No entanto, os diagramas ER são de difícil consumo, pelo que é vulgar usarem-se híbridos não-*standard* para modelar as BD, e.g. Models (no MySQL Benchmark), Conceptua Diagrams (no PowerDesigner) e ERD (no Visual Studio). De forma redutora, trata-se de usar os diagramas de classes com a nomenclatura dos diagramas ER).

Estes diagramas estão àquem dos diagrama UML de classes por não definirem os seguintes pontos:

- Interfaces
- Operações (métodos ou funções de classes);
- Herança (derivação, generalização ou especialização);
- Associações especializadas (como composição, agregação e navegação);
- Etc.

Os modelos Laravel fazem mapeamento direto a tabelas pelo que dispensam quaisquer destes pontos e, conseqüentemente, dispensariam diagramas de classes. No entanto, as questões da herança são trabalhadas além da ajuda do Laravel, pelo que será usada a respetiva nomenclatura dos diagramas UML de classes para o efeito:

5.1.2. Metodologias

O desenvolvimento do plano de dados seguiu várias metodologias que convém identificar antes de apresentar o esquema final. Podem ser resumidas da seguinte forma:

- Adoptadas *a priori*
- “Impostas” pelo Laravel
- Tomadas durante o percurso

5.1.2.1. *Adoptadas a priori*

Não foram adoptadas muitas metodologias *a priori* e que tenham sobrevivido para a posteriori. No entanto, os seguintes pontos apresentam algumas das que vingaram.

5.1.2.1.1. Inteiros

O tipo escolhido para todos os inteiros foi o tipo INT (*signed* com 4 bytes), i.e., sem pensar a respeito, não se usaram os tipos longs/BIGINT, shorts/SMALLINT, INT(3), INT(4) ou afins.

O tipo INT funciona porque o resultado de várias operações aritméticas (e.g. soma e multiplicação) é também do tipo INT. Há outros casos em isso não é garantido nem pelo PHP nem pelo MySQL.

A versão de inteiro escolhida pelo MySQL Benchmark é INT(11) que dá no mesmo. Não há impacto em uns serem INT e outros INT(11).

Neste projeto, todas as chaves (PK e FK) são inteiras; logo, do tipo INT.

5.1.2.1.2. Texto

Neste projeto, o tipo escolhido para texto é VARCHAR(255). 255 é o máximo e não se impôs, a priori, restrições a este aspecto. Esta regra é genérica para:

- **Títulos:** title, name, designation;
- **Endereços:** email, website, path, thumbnail.

Serve de exceção o recorrente campo ‘description’ que, por ser extenso, é do tipo TEXT.

5.1.2.1.3. Chaves forasteiras

A nomenclatura para FK (en: Foreign Key; pt: chave forasteira) depende do tipo de relação pretendida: neste projeto há especializações (inversa da generalização) e associações.

Especializações: caso em que uma ou mais entidades são especializações de uma entidade genérica (e.g. humanos e bovinos são especializações de mamíferos). A entidade genérica tem as propriedades comuns a todas as suas especializações enquanto cada especialização define as suas propriedades específicas. No mapeamento a tabelas, a PK da tabela especialista é também FK da tabela genérica. O nome dado a ambas as PK é id.

Associação: relação entre duas entidades com role (pt: papel ou função) em cada ponto de ligação. O nome é composto por um radical em snake_case com o sufixo ‘_id’.

Para se encontrar o radical, seguiu-se sempre uma das seguintes regras (a primeira que funciona, invalida as outras):

Regra 1: o *role* da associação está identificado junto da FK; usa-se o role.

Tabela 20

Entidade com FK	Entidade referida	FK
------------------------	--------------------------	-----------

agencies	users (creator)	creator_id
agencies	user_roles (basa)	basa_id
agency_brands	user_roles (bou)	bou_id
independent_brands	user_roles (boa)	boa_id

Regra 2: tabelas cujo o nome NÃO TÊM um prefixo comum → nome da entidade referida no singular:

Tabela 21

Entidade com FK	Entidade referida	FK
visual_identities	brand_identities	brand_identity_id
logos	visual_identities	visual_identity_id
contents	brands	brand_id

Regra 3: tabelas cujo o nome TÊM um prefixo comum → nome da entidade referida no singular sem o prefixo:

Tabela 22

Entidade com FK	Entidade referida	FK
colourway_colours	colourway_palettes	palette_id
typography_typeface	typography_types	type_id
content_in_group	content_groups	group_id

Nota: user_roles e users não partilham prefixo: partilham o nome inteiro de uma das entidades: aplica-se a regra 2.

5.1.2.2. “Impostas” pelo Laravel

O Laravel não impõe nada: fornece configuração a gosto. No entanto, a verdade é que o Laravel tem predefinições para quase tudo e contrariar essas predefinições implica escrever mais código (por detalhe, por tabela, por coluna, por ligação) e mais código implica mais trabalho e mais bugs. Assim, onde as predefinições não constituíram fricção com o programador, foram aceites. Nomeadamente nos seguintes casos:

Tabela 23

Nomes das tabelas (excepto pivots)	Plural + <i>snake case</i>	User: users Agency brand: agency_brands
------------------------------------	----------------------------	--

PK simples	id (inteiro)	
Associação polimórfica	Singular em <i>camel case</i> + 'able_type'	Content: contentable_type; User role: userRoleable_type Entity: entityable_type
Timestamps	created_at updated_at deleted_at (timestamp)	

Nota: além de transparentes as tabelas-pivots não têm nomenclatura fixa (e.g. user_role (Laravel), users_roles, users_in_roles, user_has_role (MySQL Workbench)); uma vez que este esquema tem poucos casos e já muito trabalho estava feito sobre eles, ignorou-se as predefinições do Laravel e configurou-se cada um.

5.1.2.3. Tomadas durante o percurso

A importância de isolar esta secção prende-se por estas considerações contrariarem algum do trabalho já feito, pelo que nem todo foi emendado.

O caso já referido, do nome das tabelas-pivot (que nem chegou a receber uma metodologia) serve de exemplo.

5.1.2.3.1. Entidades genéricas são abstratas

Regra: se houver especialização, a entidade genérica é trada como abstrata; desta forma, distingue-se a entidade pela coluna polimorfa (coluna na tabela genérica que identifica a tabela especialista).

Casos:

Tabela 24

Generic entity	Morphable column	Specializations
Entities	entityable_type	<ul style="list-style-type: none"> Brands, Users e Agencies
Brands	brandable_type	<ul style="list-style-type: none"> Agency brand e Independent brands
Contents	contentable_type	Visual identities, logos, palettes, typefaces, galleries, albums, ...
Galleries	galleryable_type	<ul style="list-style-type: none"> Imagery galleries e Colourway colour applications

Exceção: dos vários user_roles (BAMP Admin, BASA, BOA, BOU e BU), apenas BU tem características adicionais (“Brand user” tem FK para Brand); especializar os outros seria criar 4 tabelas apenas com a chave; e a distinção do tipo de user role já estava implementado através da entidade Roles.

5.1.2.3.2. Normalização da coluna type

Tipicamente, a coluna type pede normalização (uma tabela, só com a coluna type, referida por todos os registos do mesmo tipo). No entanto, o esquema explodiria em tabelas quase vazias (quase só chaves) como no caso do Logo:

Sendo a base:

- Logo: sem atributos
- Versions: title, image e thumbnail
- Files: path

Tabela 25

Como está	Nova tabela	Como ficaria
Formats <ul style="list-style-type: none"> • type • description 	Formats <ul style="list-style-type: none"> • designation 	Version format <ul style="list-style-type: none"> • description
Colours <ul style="list-style-type: none"> • type • description 	ColourDepths <ul style="list-style-type: none"> • designation 	Version colour depth <ul style="list-style-type: none"> • description
Types <ul style="list-style-type: none"> • type • description 	Types <ul style="list-style-type: none"> • designation 	Version type <ul style="list-style-type: none"> • description
Measurements <ul style="list-style-type: none"> • type • description • image • Thumbnail 	Measurement type <ul style="list-style-type: none"> • designation 	Measurements <ul style="list-style-type: none"> • description • image • thumbnail

5.1.2.4. Relações entre entidades

As relações entre as entidades são conceptuais e têm significado físico na implementação que

5.1.2.4.1. Exemplos

Esta secção irá referir-se ao seguinte diagrama para exemplos :

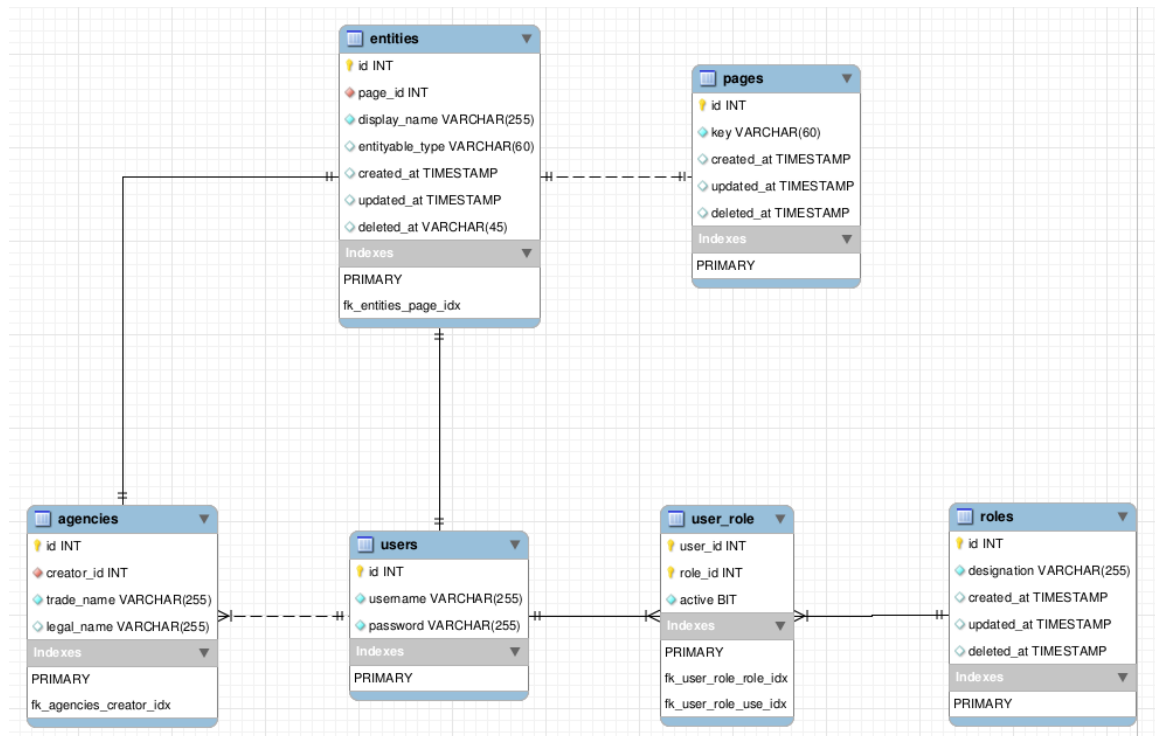


Diagrama 11 – Entidades e Relações – diagrama simplificado

Trata-se duma secção simplificada do diagrama do projecto que ilustra cada caso apresentado aqui.

5.1.2.4.2. Relações um-para-um

As relações UM-para-UM são conseguidas através de uma FK numa coluna com índice UNIQUE. Sem este índice, passa a UM-para-MUITOS.

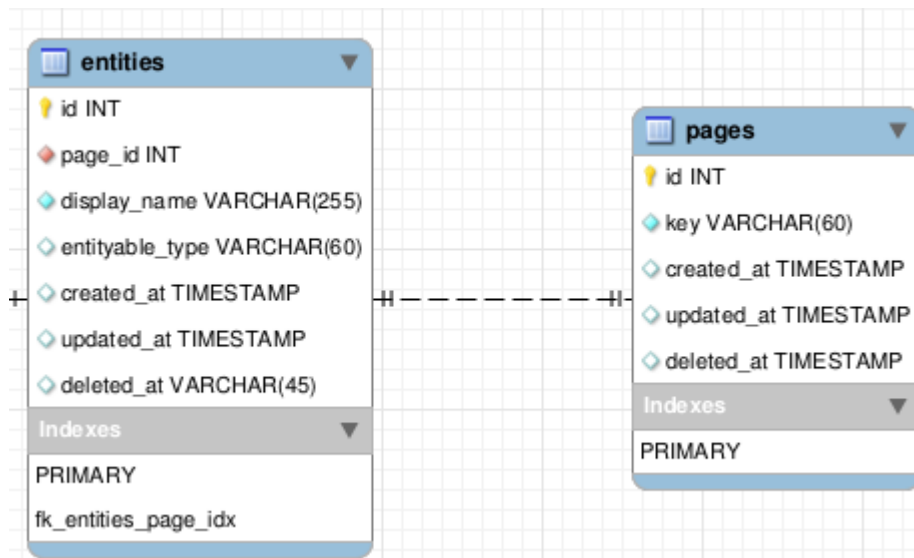


Diagrama 12 – Relações – Um para um

Leitura:

- uma 'entity' tem uma 'page';
- uma 'page' pertence a uma 'entity';

Nota 1: existe um índice UNIQUE sobre a FK 'user_id'; sem ele passaria a um-para-muitos.

Nota 2: a coluna ‘page_id’ é obrigatória; caso não fosse (um-para-um opcional) a entidade poderia existir sem ter uma página.

5.1.2.4.3. Relações um-para-muitos & muitos-para-um

Os termos um-para-muitos e muitos-para-um designam perspectivas inversas da mesma relação.

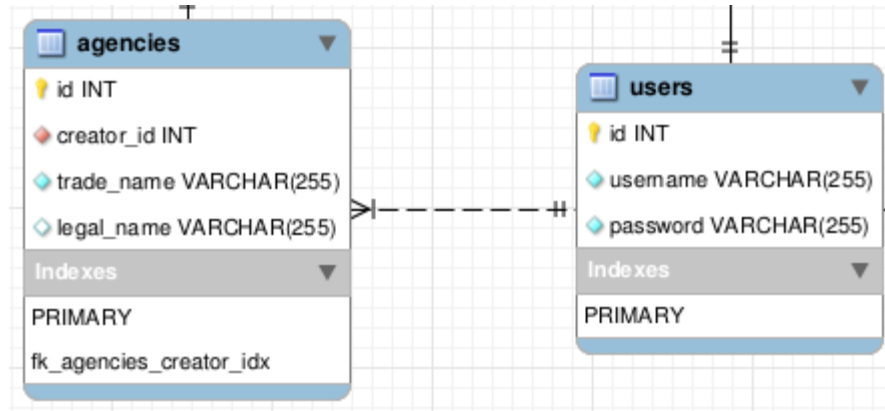


Diagrama 13 – Relações – Um para muitos

Leitura:

- um ‘user’ criou muitos ‘agencies’;
- cada ‘agency’ foi criada apenas por um ‘user’;

Nota 1: Não existe um índice UNIQUE sobre a FK ‘creator_id’; com esse índice passaria a um-para-um.

Nota 2: a coluna ‘creator_id’ é obrigatória; caso não fosse (um-para-muitos opcional) uma ‘agency’ poderia existir sem ter um ‘creator’.

5.1.2.4.4. Relações muitos-para muitos

As relações muitos-para-muitos são conseguidas por tabelas-pivot, i.e., são tabelas com PK binária (com duas colunas) em que cada coluna é também uma FK para sua tabela. Ao contrário destas, as demais tabelas produzem modelos.

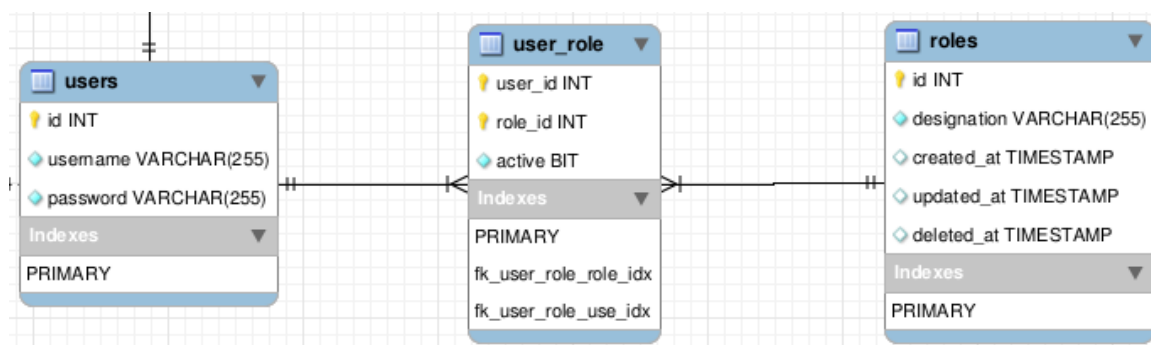


Diagrama 14 – Relações – Muitos para muitos

Leitura:

- cada ‘user’ tem muitos ‘roles’;
- cada ‘role’ tem muitos ‘users’;

Nota 1: o nome da tabela pivot ‘user_role’ é o nome implícito para o Laravel; pode ser explicitado qualquer outro nome.

Nota 2: a tabela pivot user_role tem o campo extra ‘active’ a bem do exemplo (é costume não haver campos extra).

5.1.2.4.5. Herança

Herança faz uma entidade herdar atributos, propriedades e relações de outra entidade. Isso é possível através do registo em múltiplas tabelas. A especialização faz-se de um modelo-base ou genérico para um modelo-derivado ou especialista. Diz-se que:

- “o modelo-base deriva no modelo-derivado”
- “o modelo-derivado foi derivado a partir do modelo-base”

Tabelas não derivam, pelo que não são base, nem são derivadas; não obstante, usam-se aqui os seguintes abusos de linguagem:

- **tabela-base:** tabela corresponde à entidade-base;
- **tabela-derivada:** tabela corresponde à entidade-derivada;

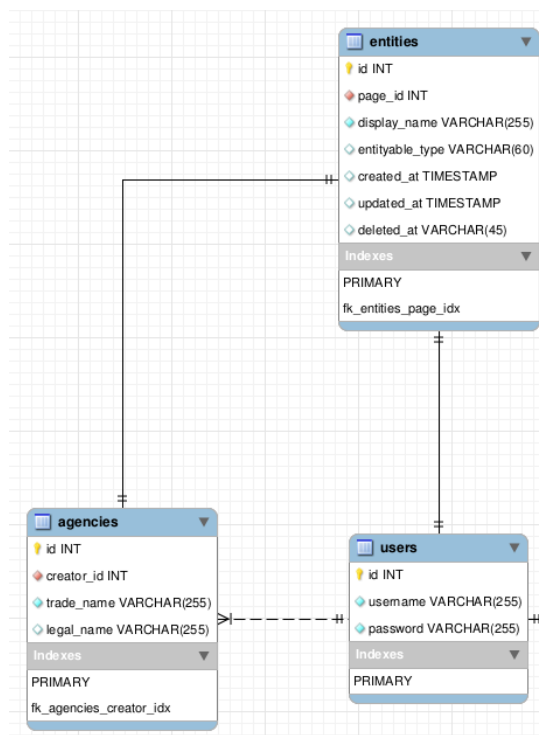


Diagrama 15 – Relações – Herança

contrariam o Laravel por não se chamar ‘entity_id’; o Laravel presume o polimorfismo apenas para associações enquanto aqui é usado em heranças.

5.1.3. Entidades por contexto

Listas extensas planas, seja qual for a ordenação e independentemente do assunto, são difíceis de consumir. Assim sendo (e para contar melhor a história do esquema) a descrição de entidades é arrumada em função dos contextos com que a plataforma se rege.

Solução: Uma tabela-base deriva numa tabela-derivada se a PK da derivada tiver uma única coluna e for também FK para a tabela-base.

Leitura:

- ‘entities’ derivou em ‘users’ e ‘agencies’;
- ‘users’ foi derivado de ‘entities’;
- ‘agencies’ foi derivado de ‘entities’.

Nota 1: as tabelas ‘users’ e ‘agencies’ não têm os timestamps por já existirem na tabela ‘entities’.

Nota 2: a tabela ‘entities’ tem uma coluna ‘entityable_type’ para registar o nome da extensão (‘User’ ou ‘Agency’) e, assim, conhecer a restante entidade a partir apenas de ‘entities’.

Nota 3: as chaves de dependência (‘id’: PK e FK para ‘entities’) em ‘users’ e ‘agencies’

5.1.3.1. Conta e Sessão

Para garantir o domínio das tecnologias que vieram a ser utilizadas, usou-se este contexto para criar o primeiro circuito completo de código (do navegador à BD e volta através da aplicação). As funcionalidades foram:

- Autenticação
- Criação de conta
- Recuperação de acesso

O seguinte diagrama ilustra as tabelas e relações que suportam as funcionalidades aqui apresentadas.

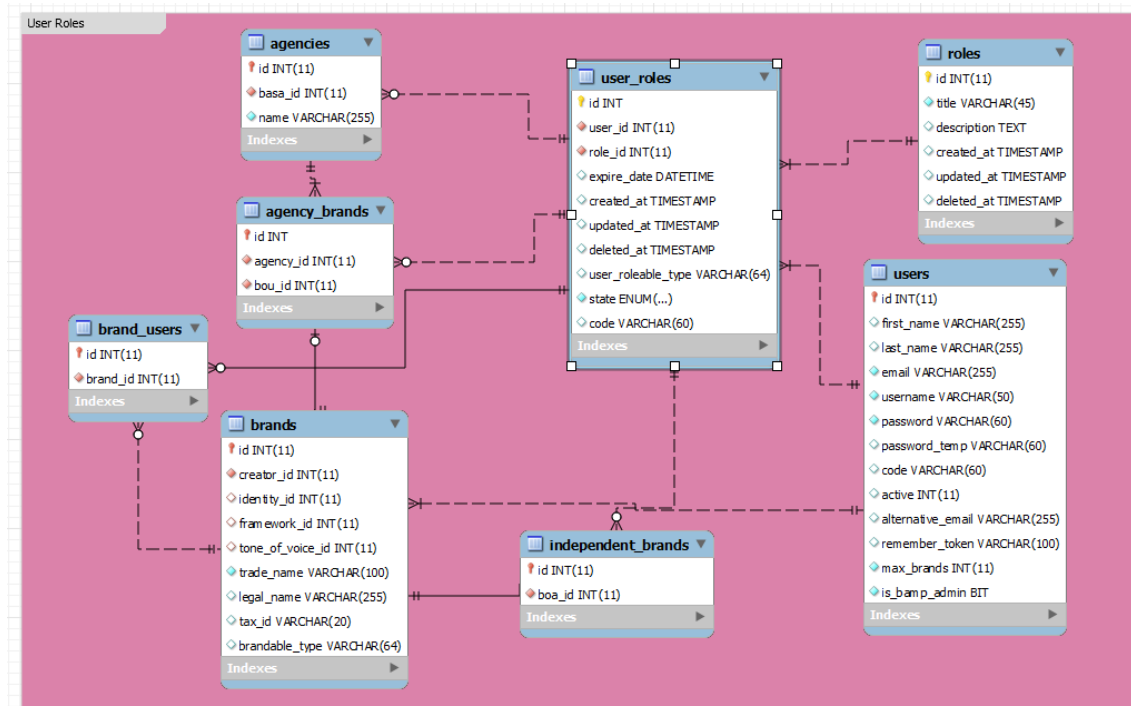


Diagrama 16 – Entidades e Relações – Conta e Sessão

As tabelas principais deste diagrama são ‘users’ e ‘user_roles’ e serão apresentadas nas seguintes divisões deste ponto.

A tabela ‘roles’ serve apenas para normalizar o nome da função na tabela ‘user_roles’ embora tenham um campo para descrição para a eventualidade de uma explicação da função.

As demais tabelas ilustram essencialmente o proveito tirado deste esquema, conforme a seguinte tabela:

Tabela 26 – Relações de funções de utilizador por entidade

Tabela	Relação	Descrição
agencies	Coluna basa_id	Um BASA com muitas agências
agency_brands	Coluna bou_id	Um BOU com muitas marcas de agência
independente_brands	Coluna boa_id	Um BOA com muitas marcas independentes.
brands	Tabela brand_users	Muitos BUs em muitas Marcas

5.1.3.1.1. Tabela ‘users’

Define os utilizadores quanto ao acesso à plataforma (quanto aos conteúdos, existem as funções).

Tabela 27 – Atributos da entidade User

Nome	Tipo	Descrição
email	varchar	Para autenticação (nada a ver com algum contacto)
username first name last name	varchar	Típicos, mas não usados.
password	varchar	
code	varchar	Enviado para o <i>email</i> para validação de conta
active	bit	0: não foi validado; 1: foi validado.
remember_token	varchar	Gravado no cliente (em <i>cookie</i>) para memorização de sessão (caso pretendido)
is_bamp_admin	bit	0: utilizador normal; 1: administrador da plataforma.

Não estão definidos nesta tabela os restantes dados do utilizador (e.g. perfil, endereços e contactos), uma vez que ‘User’ é uma generalização de ‘Entity’, que já trata dos conceitos em questão. ‘Entity’, por relacionar ainda agências e marcas, é demasiado genérica para incluir atributos como idade ou sexo (não há esse interesse; se houvesse, esses campos seriam aqui definidos por não existir uma entidade ‘Person’).

5.1.3.1.2. Tabela ‘user_roles’

Atribuição de uma função a um utilizador. Além dos campos Laravel, cada registo desta tabela identifica o utilizador bem como a função.

Tabela 28 – Atributos da tabela ‘user_roles’

Nome	Tipo	Descrição
state	enum	<ul style="list-style-type: none"> • ‘requested’, • ‘accepted’, • ‘refused’, • ‘suspended’
code	varchar	Código de validação caso passe por <i>email</i> .

É de notar que o campo não tem o valor ‘deleted’ porque o Laravel trata já desse estado (através da coluna ‘deleted_at’).

5.1.3.2. Entidades

Já se falou de ‘User’ na qualidade de especialização da entidade genérica ‘Entity’. Este número serve para definir o conceito ‘Entity’ na plataforma, bem como das restantes especializações (‘Agency’ e ‘Brand’).

Ao longo da plataforma BAMP, um utilizador está sempre enquadrado nalgum contexto. Quando o utilizador se autentica na plataforma, o contexto é o próprio utilizador. Na visita a uma marca, agência ou outro utilizador, o visitante sai do contexto para entrar no contexto da entidade visitada. Estas entidades têm, de semelhante: um perfil, um título, um avatar, contactos e uma página de entrada.

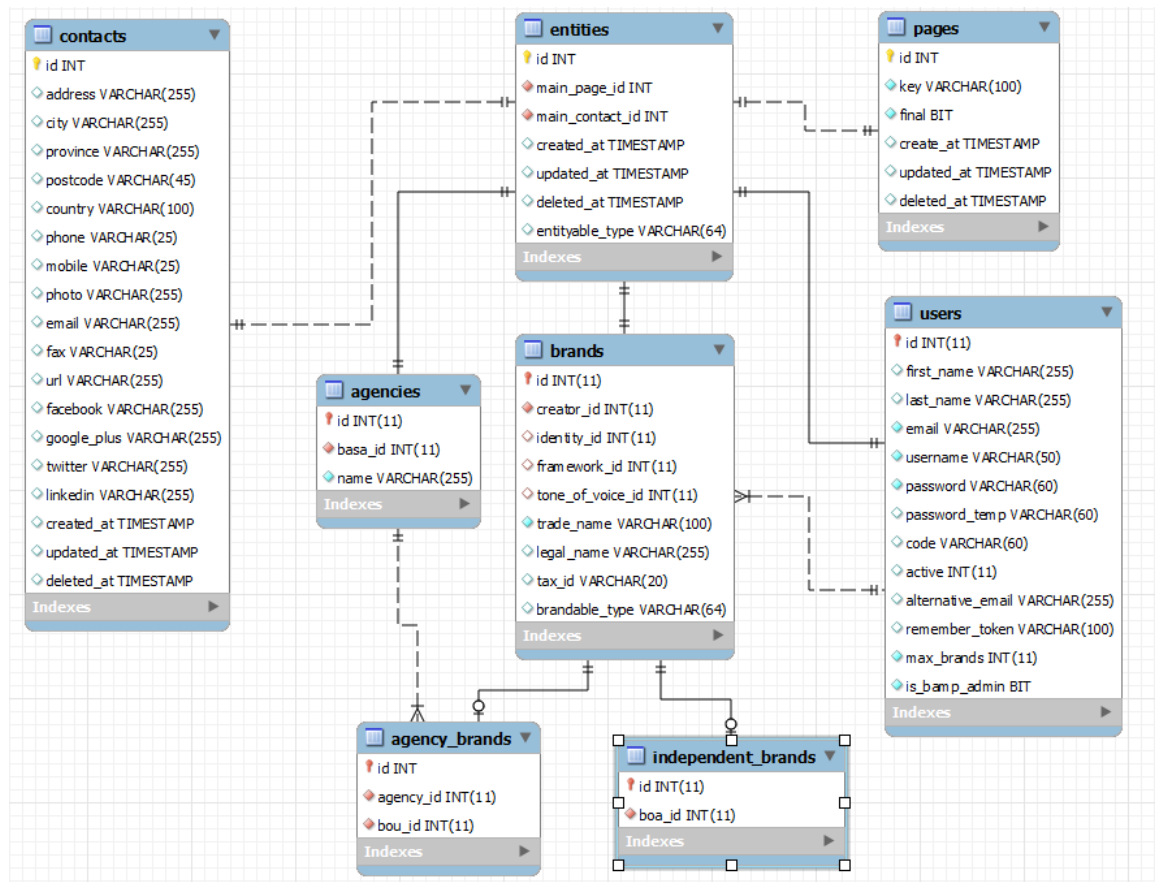


Diagrama 17 – Entidades e Relações – ‘Entities’

5.1.3.2.1. Tabela ‘entity’

(Users / Agencies / Brands)

Abstracto. Serve de aglutinador de contactos e de páginas. No momento, só tem associadas a página principal e a ficha principal de contactos.

Associações:

- main_page
- main_contact

Nome	Tipo	Descrição
avatar	varchar	Endereço da imagem que serve de avatar
Display_	bit	0. Temporário 1. Final

Atributos:

- avatar
- display_name

5.1.3.2.2. Tabela ‘pages’

Define páginas de uma entidade.

Tabela 29 – Atributos de ‘users’

Nome	Tipo	Descrição
key	varchar	Chave para produção de endereços
final	bit	2. Temporário 3. Final

Cada entidade tem as suas páginas definidas por endereços que podem ser guardados ou partilhados. Para cada entidade, a base do endereço é constante e gerada através da adição de uma chave pessoal ao domínio da plataforma:

Tabela 30 – Exemplos de endereços da plataforma

Entidade	Página	Endereço
Utilizador	Página pessoal	http://dominio-bamp/fulano/main
	Perfil	http://dominio-bamp/fulano/profile
	Marcas (em JSON)	http://dominio-bamp/fulano/brands.json
Agência	Página pessoal	http://dominio-bamp/xpto/main
	Página das Marcas	http://dominio-bamp/xpto/brands
Marca	Perfil	http://dominio-bamp/we2r345t645/profile
	Identidade visual	http://dominio-bamp/we2r345t645/visual_identity

Na tabela acima apresentada, os termos a negrito são exemplos de chaves mencionadas. Esta chave. Também é de notar que as chaves de Utilizador e Agência são legíveis em relação à chave da marca. Quando criada a entidade, é-lhe associada uma chave (gerada aleatória e automaticamente e de cariz temporário) para os seus endereços. Esta chave pode ser mais

tarde alterada por outra definitiva. Neste exemplo sugere-se que o utilizador e a agência já tenham alterado enquanto a marca mantenha a chave original.

5.1.3.2.3. Tabela ‘contacts’

Contactos de uma entidade (user, brand ou agency).

Esta tabela foi preenchida com colunas que guardam as formas de contacto típicas e suficientes para oferecer alguma coisa à plataforma. No entanto, por motivos de desempenho e escalabilidade, deverá ser evoluída em trabalho futuro.

Tabela 31 – Atributos de Contactos

Tipo	Campos
Morada	Address, city, province, postcode e country
Telefones	Phone, mobile e fax
Email	
Social	Website, facebook, twitter e google_plus

5.1.3.3. Agências

Este contexto serve para definir as agências. É composto pelas tabelas agencies e agency_brands.

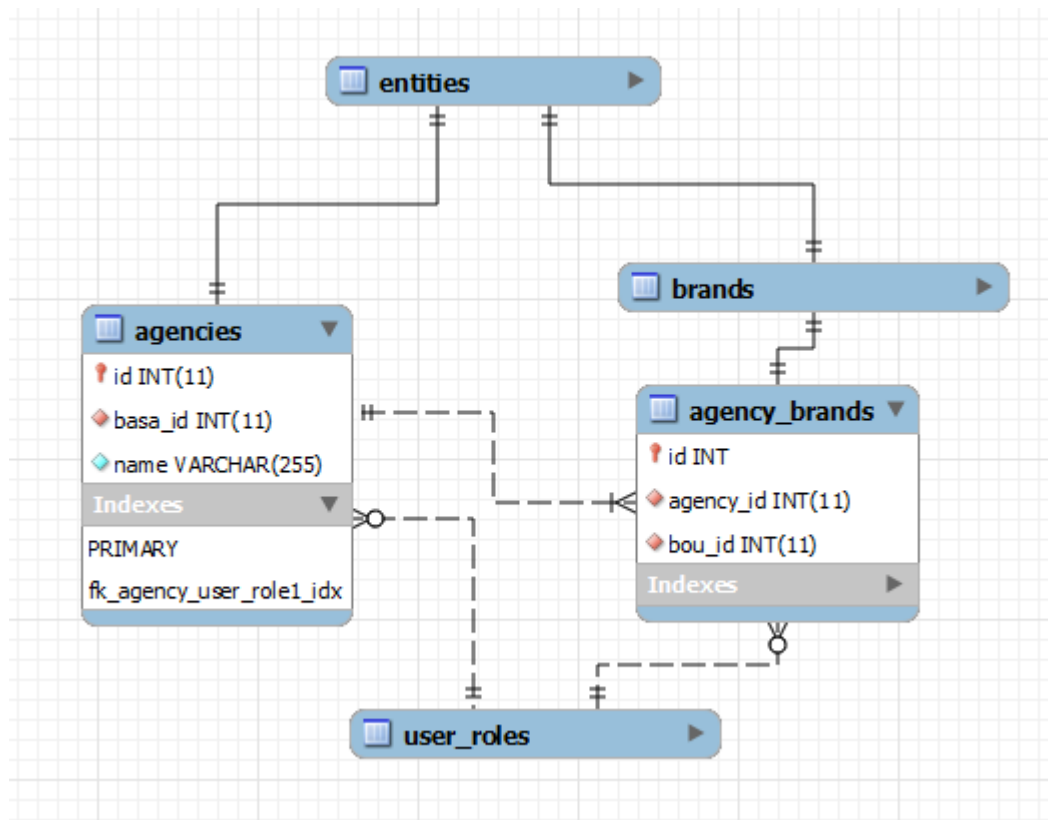


Diagrama 18 – Entidades e Relações – ‘agencies’

5.1.3.3.1. Tabela ‘agencies’

Como especializa ‘entities’, herda um avatar e um display_name. Tem, para um efeito mais formal, uma coluna ‘name’ (e.g.: display_name: FBA; name: “FBA. - Ferrand, Bicker & Associados”).

Define ainda o role BASA que corresponde ao administrador da mesma, na forma da FK basa_id.

5.1.3.3.2. Tabela ‘agency_brands’

Deriva de ‘brands’, criando distinção entre as “Marcas de uma Agência” das “Marcas de um utilizador”, onde se associa o role BOU através da coluna ‘bou_id’.

5.1.3.4. Marcas

Este é o contexto central de toda a plataforma. Se os clientes preferenciais da plataforma são as agências, o principal serviço prestado é a organização das marcas dessas agências.

A plataforma não gere apenas as marcas de agências. Qualquer utilizador tem, nesta plataforma, a possibilidade de gerir uma marca sua da mesma forma que as agências. O tratamento das marcas é muito semelhante para agências e para utilizadores independentes, havendo diferença apenas nas funções dadas aos utilizadores de cada um.

O seguinte diagrama representa as tabelas e relações deste assunto:

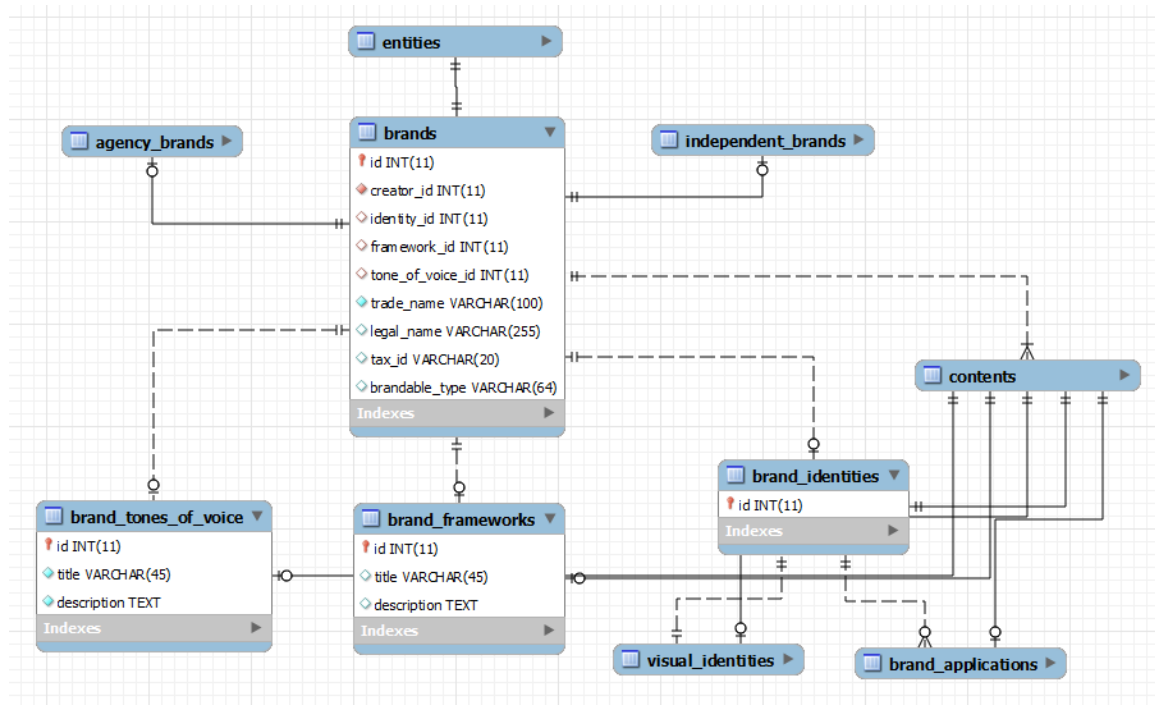


Diagrama 19 – Entidades e Relações – Marcas

Pode observar-se, no diagrama acima, que algumas tabelas estão colapsadas. Isto acontece por terem ponto próprio para descrição. Aparecem aqui apenas para as suas relações serem descritas.

Tabela 32 – Relações de ‘brands’

Genérica	entities		Herda os assuntos de ‘entities’, tal como ‘users’ e ‘agencies’.
Especializações	agency_brands		Marca de uma agência.
	independente_brands		Marca de um utilizador.
Relações (recursos)	contents		Conteúdos partilháveis
	brand_identities	visual_identities	Identidade Visual
		brand_applications	Aplicações
	brand_tones_of_voice		Tons de voz (trabalho futuro)
	brand_frameworks		(trabalho futuro)

Tabela 33 – Atributos de ‘brands’

Nome	Tipo	Descrição
trade_name	varchar	Nome comum da marca
legal_name	varchar	Nome oficial da marca
tax_id	varchar	Número de identificação fiscal

5.1.3.5. Identidade Visual

A identidade visual é conjunto dos elementos formais que representam uma marca e que definem a sua identidade. Inclui o logo, o código de cores (colourway), a tipografia bem como galerias de imagens (imagery).

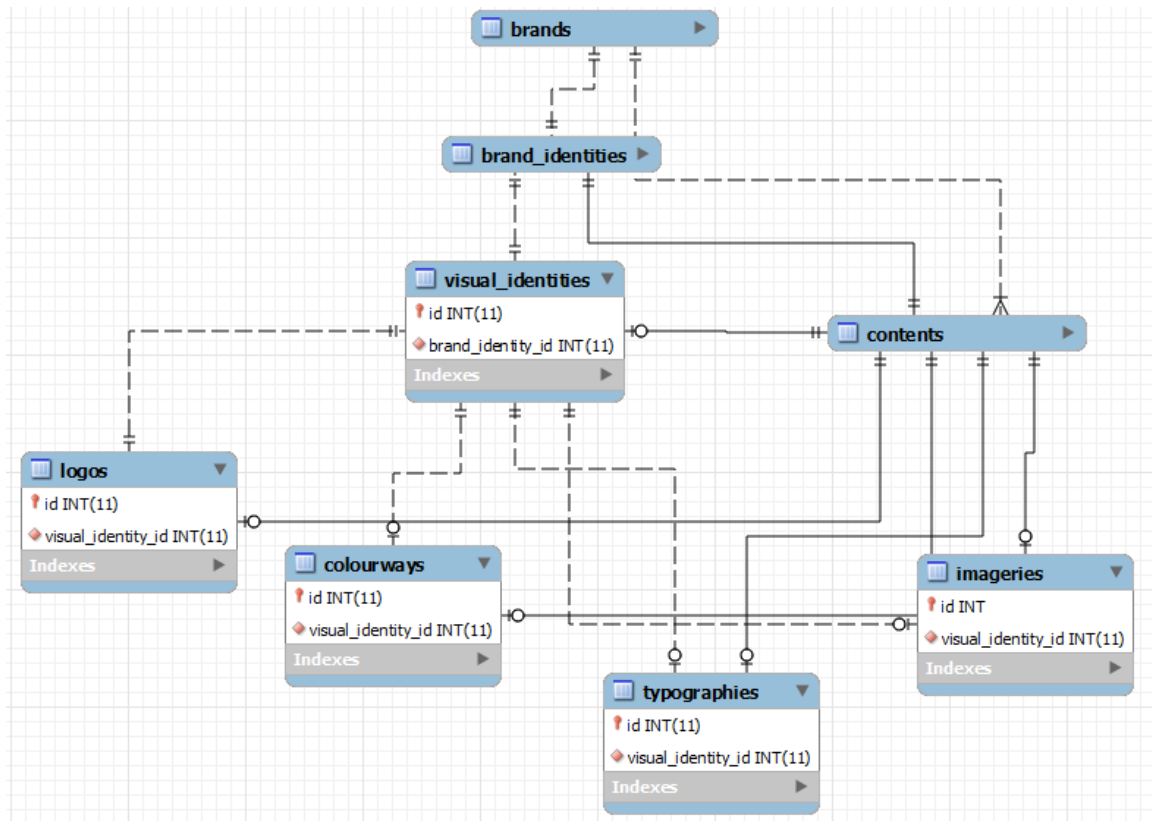


Diagrama 20 – Entidades e Relações – Identidade Visual

Tanto a identidade visual, como cada um dos seus grupos, são conteúdos partilháveis pelo que especializam a tabela ‘contents’. A definição de cada um desses grupos é apresentada nos próximos pontos.

5.1.3.5.1. Logo

Um logo é o conjunto de letras de texto e imagem que simboliza uma marca. Embora único, pode ter várias versões variando em tamanho, versão de cor, posição etc.

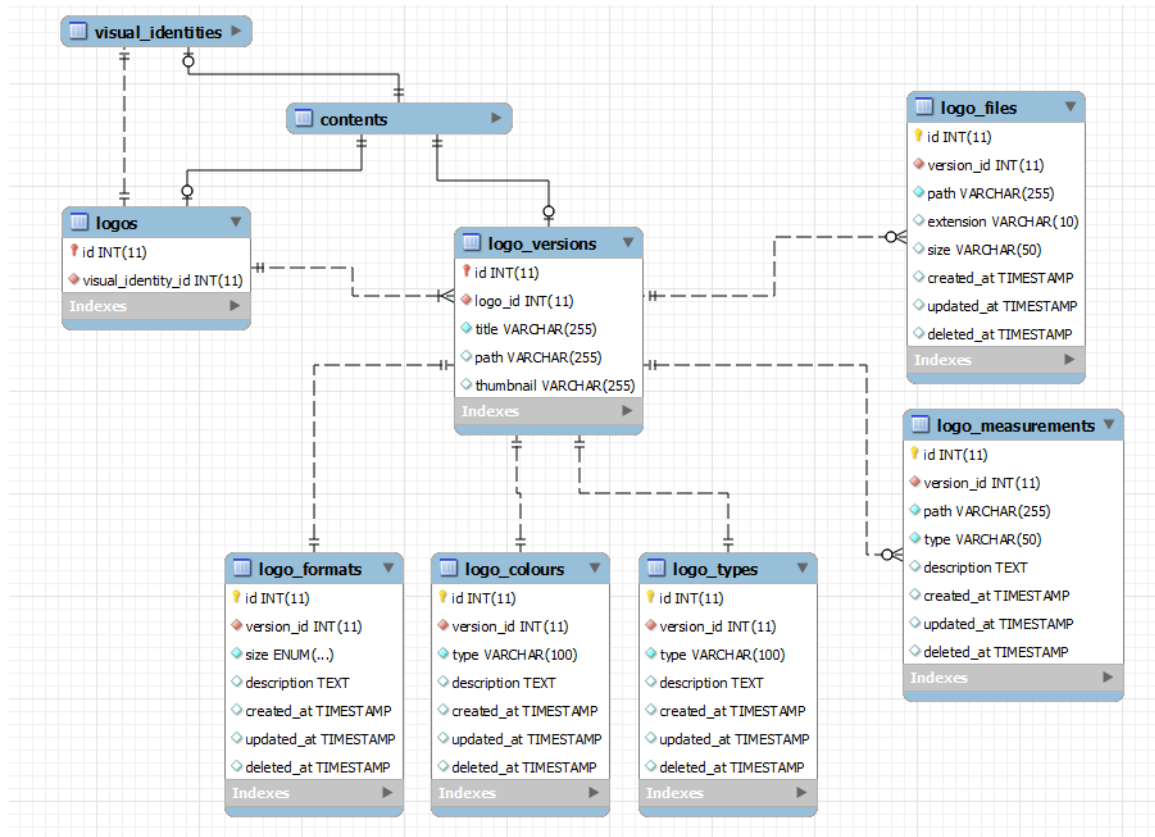


Diagrama 21 – Entidades e Relações – Logo

5.1.3.5.1.1. Logo versions ('logo_format' + 'logo_colour' + 'logo_type').

A versão de um logo é, como já dito, uma variação do logo em qualquer das suas propriedades (formato, cores e tipo). Tem um título e uma imagem.

As variações são as seguintes:

Variante	Descrição	Exemplos
Formato	O formato do logo é a definição do seu tamanho.	E.g. Padrão, Pequeno e grande
Versão de Cores	Opção por uma ou mais cores, bem como preto-e-branco ou negativo.	E.g. monocromático, verde, azul, verde e azul, três cores e negativo.
Tipo	Posição, efeito, aplicação, ...	Horizontal, vertical, símbolo, wordmark, strepline, etc.

5.1.3.5.1.2. Tabela 'logo_measurements'

Definições ou sugestões na utilização de cada versão do logo.

Tabela 34 – Atributos de 'logo_measurements'

Atributo	Tipo	Descrição
type	varchar	E.g. "protective space", "minimum size" e

		“angle”
path	varchar	Endereço da ilustração
description	varchar	Texto a acompanhar

5.1.3.5.1.3. Tabela ‘logo_files’

Ficheiros vários a anexar uma versão.

Tabela 35 – Atributos de ‘logo_files’

Atributo	Tipo	Descrição
path	varchar	Endereço do ficheiro
extension	varchar	Extensão do ficheiro
size	varchar	Dimensão do ficheiro

5.1.3.5.2. Colourway

Cores e paletas de cores e seus sistemas de representação (e.g. rgb, pantone, cmyk).

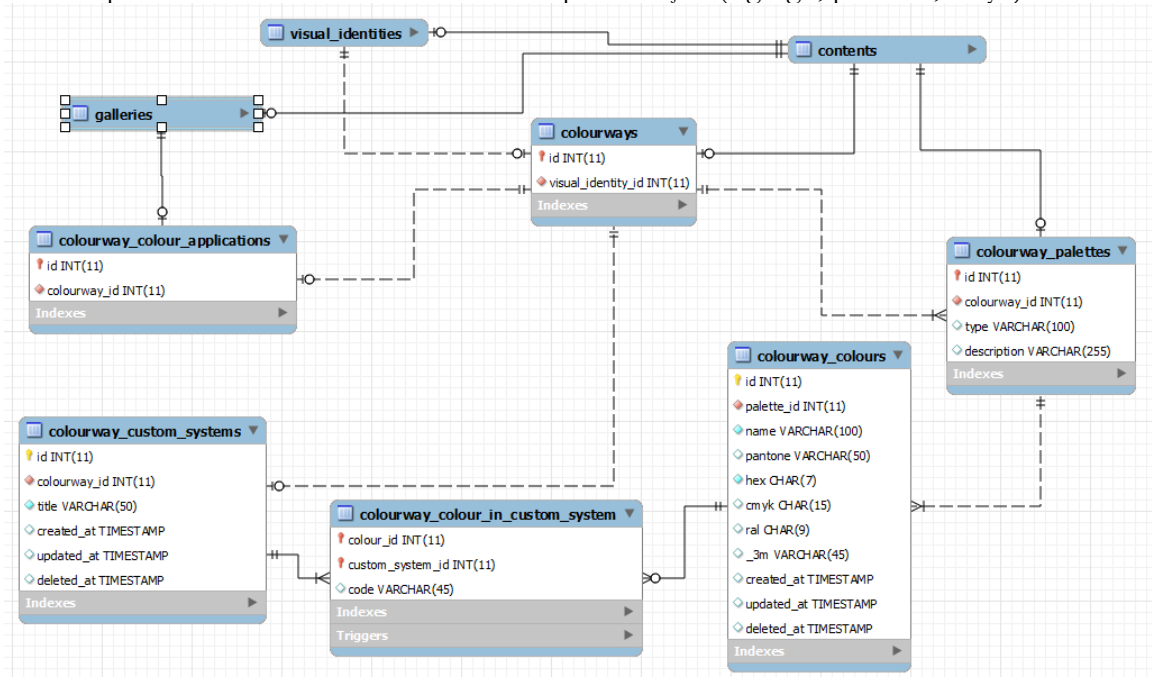


Diagrama 22 – Entidades e Relações – Colourway

Tanto o plano de cores como cada paleta podem ser partilhados, razão pela qual especializam ‘contents’.

Além disso, tem associado uma galeria de imagens (tabela ‘colourway_colour_applications’), especialização de galeria (tabela ‘galleries’) que só vem bem-definida em 5.1.3.5.4. Imagery (espaço bem definido para o galeria de imagens).

5.1.3.5.2.1. Tabela 'colourway_palettes'

A paleta é um conjunto de cores com um nome e uma descrição. A aplicação sugere 'Primary' e 'Secondary', embora possam o nome seja livre.

5.1.3.5.2.2. Tabela 'colourway_colours'

Definição de uma cor de acordo com vários sistemas. A classificação dentro dos mais populares encontram-se na nesta tabela enquanto classificações alternativas podem ser acrescentadas através da criação de sistemas personalizados (tabela, colourway_custom_system) com classificações livres.

Tabela 36 – Atributos de 'colourway_colours'

Atributo	Tipo	Exemplos
Nome	varchar	Amarelo, salmão
Pantone	varchar	Pantone 326C PANTONE Cool Gray 1 M PANTONE 31-4-6 U
Hex	varchar	#afafaf, #ff0000, #fba
CMYK	varchar	0-0-0-100, 23-148-0-50
RAL	varchar	6048
3M	varchar	30 -730 Apple green

5.1.3.5.3. Typography

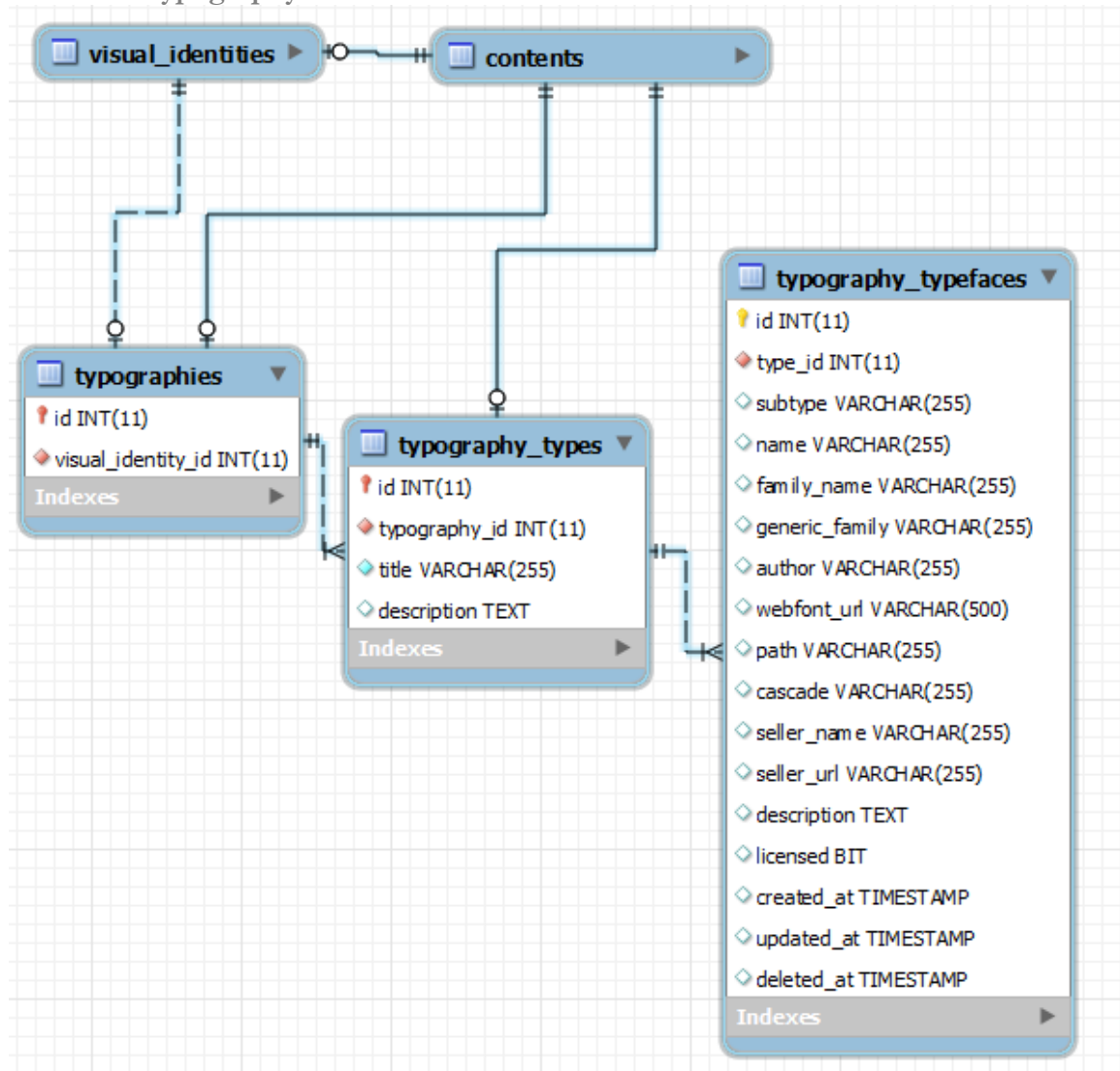


Diagrama 23 – Entidades e Relações – Typography

5.1.3.5.3.1. Tabela ‘typography_types’

Grupo de typefaces. Tem um título e uma descrição. São exemplos: primary, secondary e substitute.

5.1.3.5.3.2. Tabela ‘typography_typefaces’

Tabela 37 – Atributos de ‘typography_typefaces’

Atributo	Type	Descrição
subtype	varchar	
name	varchar	
family_name	varchar	
generic_family	varchar	
author	varchar	
webfont_url	varchar	

path	varchar	
cascade	varchar	
seller_name	varchar	
seller_url	varchar	
description	varchar	
licensed	bit	

5.1.3.5.4. Imagery

Aqui não se definem-se as galerias da identidade visual. Tratando-se também de uma galeria, aquela que ilustra aplicações de cores da colourway é definida aqui.

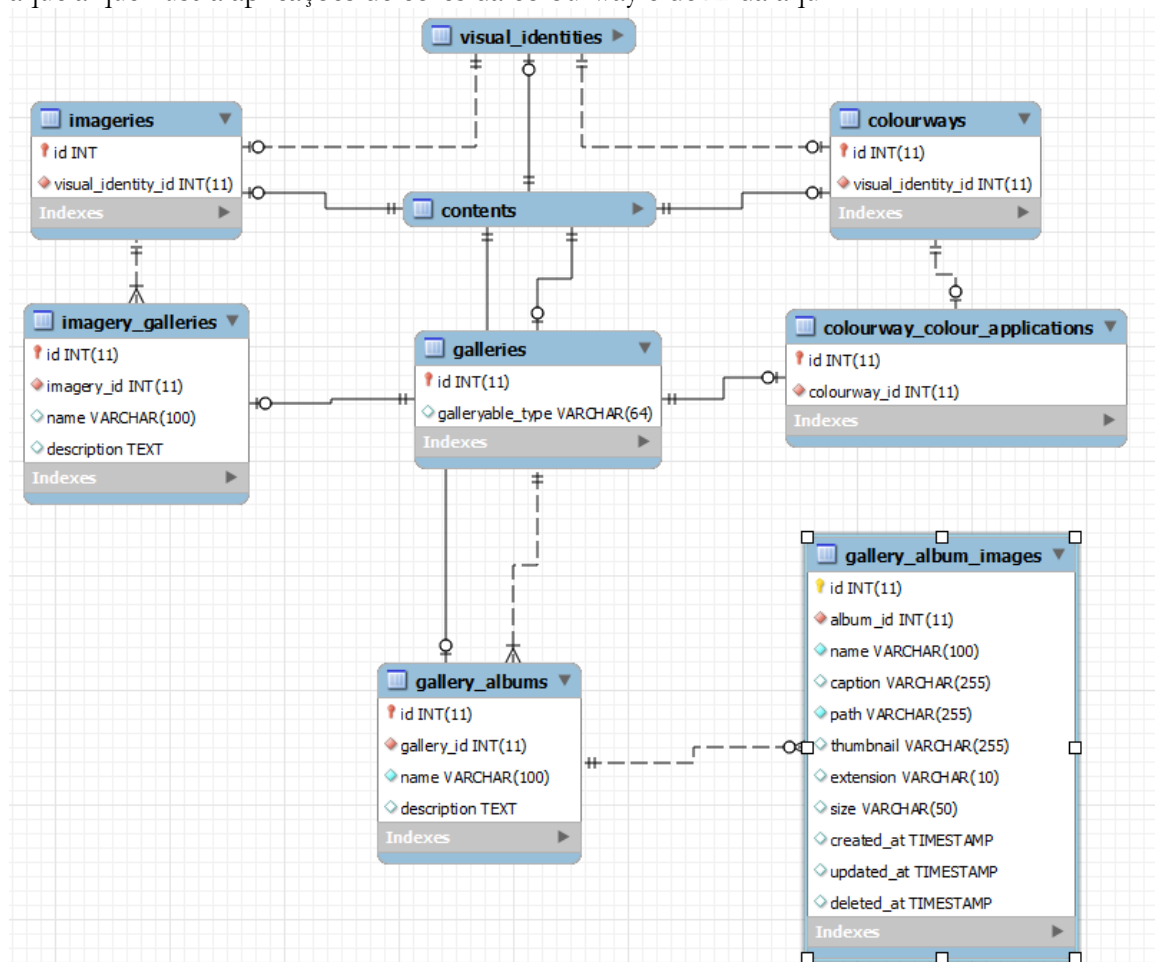


Diagrama 24 – Entidades e Relações – Imagery

5.1.3.5.4.1. Tabelas ‘galleries’

A tabela ‘gallery’ é genérica e serve de base, tanto para ‘imagery_galleries’ como ‘colourway_colour_application’. Desta forma, ambas possuem álbuns com imagens, embora com as seguintes diferenças:

	Imagery galleries	Colourway colour applications
--	-------------------	-------------------------------

Cardinalidade	Várias dentro de Imagery	Única em Colourway
Atributos adicionais	Nome e descrição (varchar)	Não há

5.1.3.5.4.2. Tabela 'gallery_albums'

Uma galeria pode ter os álbuns que lhe forem adicionados. Estes definem-se por nome e têm uma descrição.

5.1.3.5.4.3. Tabela 'galley_album_images'

Atributo	Tipo	Descrição
name	varchar	Título simples
caption	varchar	Subtítulo
path	varchar	Endereço do ficheiro
thumbnail	varchar	Endereço da miniatura
extension	varchar	Extensão da imagem
size	varchar	Tamanho do ficheiro

5.1.3.6. "Brand Applications"

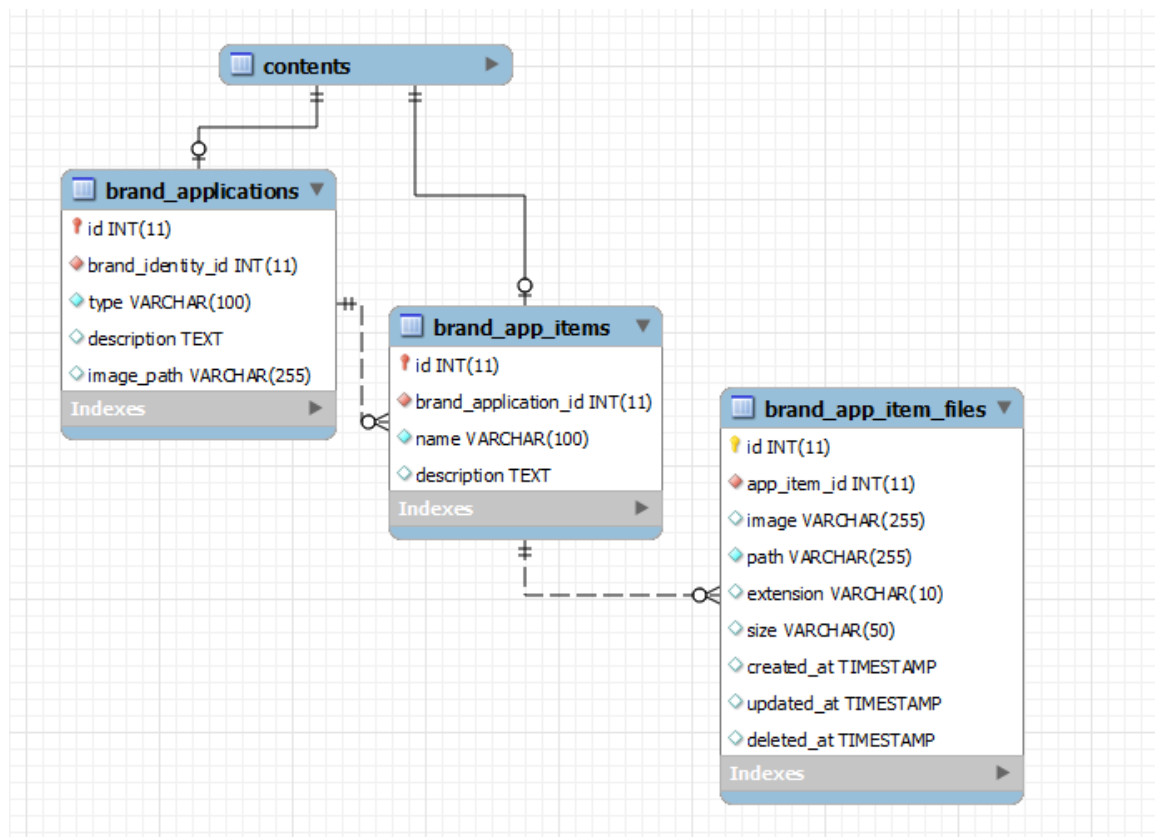


Diagrama 25 – Entidades e Relações – “Brand Applications”

Aplicações associadas à marca. Aqui define-se “Brand Applications” como sendo grupos de tipos de elementos tipicamente usados como modelos de trabalho para a produção gráfica (e.g. cartas, camisolas, canetas) e documentação (e.g. documentos, apresentações).

Relações:

- Uma ‘brand identity’ pode ter várias ‘brand applications’.
- Cada ‘brand application’ pode ter vários itens;
- Cada item, pode ter vários ficheiros.

5.1.3.6.1. Tabela ‘brand_applications’

Tabela 38 – Atributos de ‘brand_applications’

Nome	Tipo	Descrição
type	varchar	Exemplos: <ul style="list-style-type: none"> • Estacionário (envelopes e cartas) • Modelos de Documento (folha de cálculo e apresentações)
description	varchar	Texto livre
image_path	varchar	Imagem representativa

5.1.3.6.2. Tabela ‘brand_app_items’

Tabela 39 – Atributos de ‘brand_app_items’

Nome	Tipo	Descrição
name	varchar	Exemplos de Estacionários: <ul style="list-style-type: none"> • Envelopes • Cartas Exemplos de Modelos de Documento: <ul style="list-style-type: none"> • Documentos; • Apresentações.
description	varchar	Texto livre

5.1.3.6.3. Tabela ‘brand_app_item_files’

Tabela 40 – Atributos de ‘brand_app_item_files’

Nome	Tipo	Descrição
image	varchar	Endereço da previsualização
path	varchar	Endereço do ficheiro a transferir
extension	varchar	Extensão do ficheiro a transferir
size	varchar	Tamanho do ficheiro a transferir

5.1.3.7. Partilha

Foi para conseguir a partilha, que se introduziu pela primeira vez o conceito de polimorfismo aplicado à BD. As várias entidades partilháveis foram generalizadas na entidade-base “contents” e da qual herdam a capacidade de ser partilhadas.

O seguinte diagrama apresenta o esquema da BD para a realização do conceito de partilha:

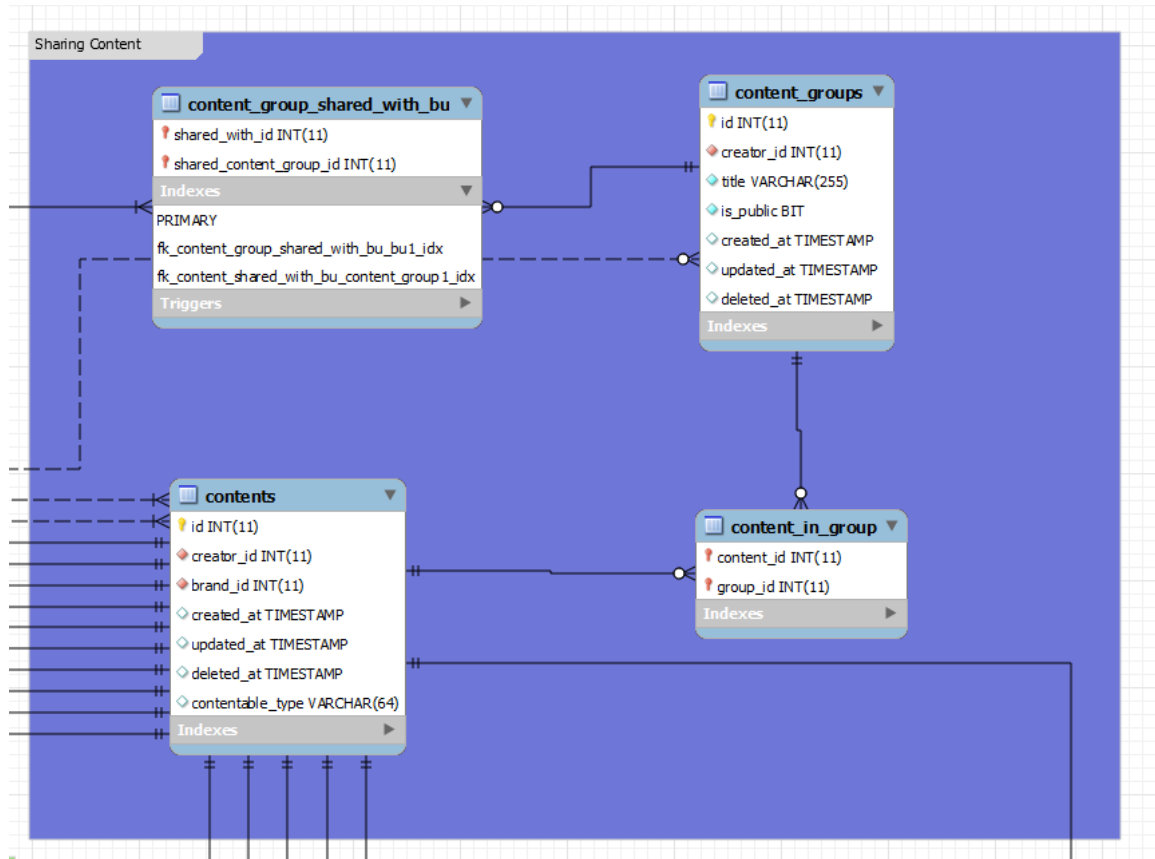


Diagrama 26 – Entidades e Relações – Partilha

Designam-se as tabelas acima apresentadas para os seguintes efeitos:

Entidades

- **contents:** base dos conteúdos partilháveis;
- **content_groups:** grupos de partilha;

Associações MUTOS-para-MUITOS:

- **content_in_group:** entre grupos e conteúdos;
- **content_group_shared_with_bu:** entre grupos e BUs.

5.1.3.7.1. Tabela ‘contents’

A tabela ‘contents’ não tem propriedades, além da identificação do criador e marca, bem como das colunas Laravel.

5.1.3.7.2. Tabela ‘groups’

A tabela ‘content_groups’, além da identificação do criador, precisa de um título.

A opção de publicação do grupo (coluna ‘is_public’) permitiria que um endereço para a visualização dos conteúdos em questão funcione sem que o utilizador esteja sequer autenticado (e consequentemente, possa ser transmitido para terceiros). Na realidade, estas páginas de visualização são apanhadas pelo filtro ‘auth’ impossibilitando esta situação. Dá, no entanto, para apresentar conteúdos que sejam públicos a utilizadores autenticados estranho à Marca do conteúdo.

5.2. 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 funcionamento desta responde a pedidos na forma de endereços URL geridos por estes elementos.

5.2.1. Routing

No desenvolvimento *web* básico, os URLs endereçam ficheiros ou pastas de um servidor. Ainda dentro do básico, é normal haver direcionamento da pasta para um ficheiro predefinido dentro daquela pasta (e muitas vezes a falhar na inexistência do mesmo).

Ultrapassando o básico, é possível “romper” com o endereçamento típico e criar novos significados para quaisquer URLs dentro do domínio gerido pelo Laravel. Com esta habilidade, pode-se obter os seguintes efeitos para um mesmo pedido:

Exemplo do Pedido: `http://exemplo.pt/utilizadores.txt`

- Devolver um ficheiro chamado “utilizadores.txt”;
- Gerar um resultado compatível com o esperado pelo navegador (um ficheiro de texto que nunca existiu no servidor);
- Executar qualquer outra acção, criando uma máscara em relação ao verdadeiro propósito do endereço (e.g., apresentar uma imagem ou criar uma transacção).

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

O endereçamento desenvolvido neste projecto realiza-se tipicamente nos seguintes modelos:

- Páginas web (método get: resultado em formato HTML);
- Serviços web de consulta (método get: resultado em formato JSON);
- Comandos / submissões (método post; termina redireccionando para algum URL).

5.2.1.1. Páginas web (*views*)

Uma página *web* é uma *view* (simples ou composta) conforme definido na secção MVC deste capítulo.

Na versão típica de um pedido *web* a uma página, não há sumissão de dados eventualmente sensíveis, pelo que é usado o HTTP Method POST. Tanto o navegador fica possibilitado de guardar a chamada no histórico, como o utilizador de guardar o endereço para futura utilização (e.g., na forma de marcadores / favoritos).

O seguinte exemplo serve para apresentar a página de entrada no sistema mediante autenticação.


```
Route::get('/account/sign-in', array(
    'as' => 'account-sign-in',
    function () {
        return View::make('account.signin');
    }
));
```

Código 1 – Laravel Routing – Para uma vista (*blade*)

O fragmento, acima apresentado, concilia os seguintes elementos:

- **URL:** <http://dominio-bamp/public/index.php/account/sign-in>
- **HTTP Method:** GET;
- **Named route:** “account-sign-in”
- **Destino:** página “signin.blade.php” na pasta “app/views/account”

O “named route” não é obrigatório; aproveita-se o exemplo para apresentar o conceito: trata-se de uma referência simplificada para o mesmo URL para uso em código Laravel, i.e., para uma hiperligação àquela página usa-se o seguinte interrupção ao código html dentro de um qualquer blade:

```
<a href="{{ URL::route('account-sign-in') }}">Sign in</a>
```

Código 2 – Laravel Routing – Usagem

Desta forma não há preocupação com endereços relativos ou absolutos; o Laravel trata de apresentar o URL correcto.

5.2.1.2. *Serviços web de consulta (JSON)*

Serviços web são uma solução de integração entre sistemas tipicamente diferentes. Com esta tecnologia, é possível integrar, com o servidor, vários tipos de cliente que apenas tenham necessidade de comunicação de dados e comandos. No início da tecnologia a comunicação era feita em formato XML, enquanto aqui se usa JSON (mais simples para Javascript e idêntico para o PHP).

O único cliente implementado é o cliente *web* da plataforma. Uma vez carregada pelo navegador, as páginas consultam estes serviços para preencher graficamente os seus conteúdos minimizando o processamento do servidor e a transferência de objectos gráficos.

Não sendo uma vista, não é redireccionado para um blade, mas sim para o Controlador que produz o ficheiro a apresentar.

Usa-se o HTTP Method GET por não haver submissão de dados sensíveis podendo ter mais-valia de histórico e favoritos dos navegadores.

```
Route::get('/{user}/session.json', array(
    'as' => 'session-json',
    'uses' => 'SessionController@getJson'
));
```

Código 3 – Laravel Routing – Para o controlador, com argumentos (*get json*)

Note-se o termo “{user}” da declaração do endereço; trata-se de um parâmetro (Laravel route parameter) de nome ‘user’ a ser passado para o método invocado por este endereço.

O fragmento, acima apresentado, concilia os seguintes possíveis elementos:

- **URL:** <http://dominio-bamp/public/index.php/fulano/session.json>
- **HTTP Method:** GET;
- **Named route:** “session-json”;
- **Destino:** método ‘getJson’ na classe ‘SessionController’;

Para o exemplo apresentado, o parâmetro ‘user’ receberia o valor “fulano”, chave hipotética de identificação de um utilizador igualmente hipotético.

O implementação do método ‘getJson’ na classe ‘SessionController’ segue o seguinte modelo:

```
class SessionController extends BaseController {
    // ...
    function getJson($user){
        // ...
    }
    // ...
}
```

Código 4 – Laravel Routing – Esqueleto de um destino com argumentos

Em que o parâmetro ‘user’, no exemplo sugerido anteriormente, tomaria o valor ‘fulano’.

5.2.1.3. Comandos e Submissões

Comandos: serviços invocados assincronadamente pelo Javascript sem sair da página – o resultado aparece em JSON para ser interpretado pelo invocador.

Submissões: de formulários por mudança de página (e.g. registo de utilizador) - a consequência é o redireccionamento para uma página já com a submissão tratada (e.g. sucesso no registo).

Este modelo também funciona como serviço *web*. Como tal é igualmente redireccionado para um método específico do controlador em questão.

Por haver submissão de dados, o HTTP Method é POST impedindo a invocação directa na barra do endereçamento do navegador e ocultando a passagem dos dados, tanto da barra de endereços como do histórico do navegador.

Segue-se o fragmento necessário para criar o endereço de submissão dos dados de um perfil (comum para utilizadores, agências e marcas):

```
Route::post('/{page}/profile', array(
    'as' => 'profile-post',
    'uses' => 'PageController@postProfile'
));
```

Código 5 – Laravel Routing – Para o controlador, com argumentos (post)

O fragmento, acima apresentado, concilia os seguintes possíveis elementos:

- **URL:** <http://dominio-bamp/public/index.php/xpto/profile>
- **HTTP Method:** POST;
- **Named route:** “profile-post”;
- **Destino:** método ‘postProfile’ na classe ‘PageController’;

Onde um parâmetro ‘page’ recebe o texto “xpto” (chave de identificação de um utilizador, agência ou marca).

Após a execução destas acções, o controlador redireciona a execução do processo corrente conforme o tipo:

- **Comandos:** para um serviço *web* de consulta (em JSON) para ser interpretado pelo Javascript;
- **Submissões:** para uma página de sucesso, erro ou mesmo para a página anterior conforme o plano.

5.2.2. Filters

Os filtros criam uma camada de segurança permitindo condicionar o endereçamento (os Laravel Routes) ao reconhecimento do utilizador. Implementaram-se três filtros:

- **Guest:** utilizadores anónimos;
- **Auth:** utilizadores autenticados;
- **CSRF:** cross-site request forgery (detecção de sessões inválidas).

Os filtros são implementados no ficheiro ‘filters.php’ da pasta ‘app’ da aplicação e implementam-se como o seguinte caso:

```
Route::filter('auth', function()
{
    if (Auth::guest())
        return Redirect::guest(URL::route('account-sign-in'));
});
```

Código 6 – Laravel Route Filters – ‘auth’ para utilizadores autenticados

O caso acima apresentado cria o termo ‘auth’ para agrupamento de endereços. Pode observar-se a predefinição do comportamento para utilizadores anónimos: redirecciona-os para a página de autenticação.

A criação destes filtros permite agrupar os endereços do ‘routes.php’ em secções que só são válidas se a respectiva condição se verificar, impedindo acessos negados.

Exemplo do agrupamento dos endereços para utilizadores autenticados:

```
/**
 * Only authenticated users may enter...
 */
Route::group(array('before' => 'auth'), function () {
    // Route page-main
    // ...
    // Route session-json
    // ...
    // Route profile-post
    // ...
})
```

Código 7 – Laravel Route Filters – Agrupamento de reencaminhamentos por um filtro

Os filtros apresentados vêm implementados na instalação do Laravel. Outros filtros ficaram implementados para ajustar o acesso aos endereços às funções do utilizador (e.g. o BU não pode editar conteúdo, pelo que a route ‘brand-typography’ (submissão da tipografia) deveria estar vedada já aqui). A preocupação com a implementação de funcionalidades foi adiando a questão de as restringir, pelo que fica em trabalho futuro.

5.3. MVC (*Model-View-Controller*)

O Laravel assenta na arquitectura MVC (Model+View+Controller).

Enquanto as Views e os Controllers têm presença na camada de apresentação, os componentes do tipo Modelo implementam a camada de acesso a dados, refletindo o Modelo de Dados e, conseqüentemente, a BD.

5.3.1. Models

Os modelos Laravel são classes que reflectem e manipulam o estado da BD. Existe um modelo para cada tabela (que não seja *pivot*), escrito conforme a mesma, suas colunas, relações, índices e respectivas propriedades. Nesta secção descrevem-se os processos de geração dos mesmos.

Cada modelo é implementado na forma de classe PHP que deriva da classe Eloquent (da biblioteca do Laravel) e mapeia directamente a correspondente entidade, do Modelo de Dados, e tabela na BD. A título de exemplo, para a entidade “User”, do Modelo de Dados, existe uma tabela “users” na BD e um modelo “User” 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 implementação de classes com estes membros.

5.3.1.1. Introdução

O modelo de entidades define algumas 40 entidades traduzidas em perto de 50 tabelas. São muitos modelos para serem implementados em rotinas monotonicamente similares.

Qualquer alteração à BD obriga a conseqüente actualização dos respectivos modelos; enquanto a BD não estiver estável, os modelos terão de sofrer edição a tempo inteiro.

A dimensão da base-de-dados e as suas recorrentes alterações fizeram surgir a necessidade de procurar por uma ferramenta que gerasse e actualizasse os modelos a serem usados. A pesquisa por essa ferramenta revelou-se infrutífera: não há nada oficial e os projectos encontrados não respondiam às necessidades ou tinham (à altura do projecto) ligações quebradas.

A opção que sobrou foi implementar uma série de rotinas em Java que foram produzindo e a actualizando os Modelos Laravel.

As próximas secções deste capítulo dedicam-se a relatar o desenvolvimento dos Modelos a partir da BD e através do Java.

5.3.1.2. Reflexão da BD

À semelhança dos motores de dados mais comuns (Oracle, PostgreSQL, SQL Server), o MySQL disponibiliza o seu “Information schema”. Trata-se de um conjunto de vistas que reflete a estrutura do motor fornecendo informação sobre esquemas, tabelas, vistas, relações, procedimentos, etc.

As seguintes consultas a estas vistas permitem obter toda a informação necessária para compreender e utilizar qualquer motor.

5.3.1.2.1. Tabelas

Por cada tabela, constrói-se um modelo Laravel. A seguinte consulta dá o nome das tabelas presentes no esquema 'bamp_db'

```
select table_name
from information_schema.tables
where
    table_schema like 'bamp_db' and
    table_type not like 'VIEW';
```

Código 8 – Reflexão MySQL – Tabelas

Nota 1: que são excluídos resultados com table_type 'VIEW'. Sem a exclusão, obtém-se também o nome das vistas que não são interessantes para a produção dos modelos.

Nota 2: este resultado devolve também as tabelas *pivot*, de ligação que implementam a relação muitos-para-muitos entre modelos; estas tabelas não produzem modelos, mas a análise das chave primárias e forasteiras irá distingui-las.

5.3.1.2.2. Chaves Primárias

A seguinte consulta devolve o nome das colunas por tabela:

```
select
    table_name,
    column_name
from
    information_schema.key_column_usage
where
    table_schema like 'bamp_db' and
    constraint_name like 'PRIMARY'
```

Código 9 – Reflexão MySQL – Chaves primárias (PK)

Nota: as chaves primárias dos modelos tem sempre o nome "id" e tipo "int"; no entanto, as tabelas-pivot têm chaves de duas colunas cada uma com o seu nome.

5.3.1.2.3. Chaves Forasteiras

Para a extração das chaves forasteiras, recorreu-se ao seguinte código.

```
select
    constraint_name,
    table_name,
    column_name,
    referenced_table_name,
    referenced_column_name
from
    information_schema.key_column_usage
where
    table_schema like 'bamp_db' and
    referenced_table_name is not null
```

Código 10 – Reflexão MySQL – Chaves-forasteiras (FK)

Nota: embora provenham da mesma vista que as chaves primárias, distinguem-se por terem colunas que identificam a tabela referida (a primária tem estas colunas a null).

5.3.1.2.4. Colunas

O comando "describe" permite consultar a informação necessária para conhecer as colunas das tabelas e suas propriedades. No seguinte exemplo, o comando para descrever a tabela "users".

```
describe users;
```

Código 11 – Reflexão MySQL – Colunas

Que fornece o seguinte resultado:

Tabela 41

	Field	Type	Null	Key	Default	Extra
1	id	int	NO	PRI	[NULL]	
2	first_name	varchar(255)	YES		[NULL]	
3	last_name	varchar(255)	YES		[NULL]	
4	email	varchar(255)	NO	UNI	[NULL]	
5	username	varchar(50)	NO	UNI	[NULL]	
6	password	varchar(60)	NO		[NULL]	
7	password_temp	varchar(60)	YES		[NULL]	
8	code	varchar(60)	YES		[NULL]	
9	active	int	YES		0	
10	alternative_email	varchar(255)	YES		[NULL]	
11	remember_token	varchar(100)	YES		[NULL]	

O campo key pode ter os seguintes valores:

- “”: coluna normal;
- **PRI**: chave primária;
- **MUL**: chaves forasteiras do tipo muitos-para-um
- **UNI**: índices de unicidade (UNIQUE) - nas chaves forasteiras, define-as como

O campo extra valores automáticos; usaram-se os seguintes:

- auto_increment - para PK do tipo int (User deriva de Entity: a PK é atribuída);
- timestamp - em inserções.

5.3.1.3. Bibliotecas Java

Foram quatro as bibliotecas usadas para o apoio à programação:

- MySQL Connector J: integração com MySQL, canal de instruções e dados;
- Inflector: conversor de número (singular/plural);
- Guava: manipulação de Strings e Coleções;
- Gson: serialização em JSON.

5.3.1.3.1. MySQL Connector J

Biblioteca Java de um cliente MySQL para envio de comandos e obtenção de dados.

Exemplo de obtenção de uma ligação ao MySQL

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

Connection conn = null;
...
try {
    conn =
        DriverManager.getConnection("jdbc:mysql://localhost/test?" +
```

```

        "user=monty&password=greatsqlldb");

    // Do something with the Connection

    ...
} catch (SQLException ex) {
    // handle any errors
    System.out.println("SQLException: " + ex.getMessage());
    System.out.println("SQLState: " + ex.getSQLState());
    System.out.println("VendorError: " + ex.getErrorCode());
}

```

Código 12 – Bibliotecas Java – MySQL Connector J – Ligação

Exemplo de execução de uma consulta.

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.ResultSet;

// assume that conn is an already created JDBC connection (see previous examples)

Statement stmt = null;
ResultSet rs = null;

try {
    stmt = conn.createStatement();
    rs = stmt.executeQuery("SELECT foo FROM bar");

    // or alternatively, if you don't know ahead of time that
    // the query will be a SELECT...

    if (stmt.execute("SELECT foo FROM bar")) {
        rs = stmt.getResultSet();
    }

    // Now do something with the ResultSet ....
}
catch (SQLException ex){
    // handle any errors
    System.out.println("SQLException: " + ex.getMessage());
    System.out.println("SQLState: " + ex.getSQLState());
    System.out.println("VendorError: " + ex.getErrorCode());
}
finally {
    // it is a good idea to release
    // resources in a finally{} block
    // in reverse-order of their creation
    // if they are no-longer needed

    if (rs != null) {
        try {
            rs.close();
        } catch (SQLException sqlEx) { } // ignore

        rs = null;
    }

    if (stmt != null) {
        try {
            stmt.close();
        } catch (SQLException sqlEx) { } // ignore
    }
}

```

```

        stmt = null;
    }
}

```

Código 13 – Bibliotecas Java – MySQL Connector J – Execução de uma consulta

5.3.1.3.2. Inflector

Os nomes das tabelas na BD estão no plural. Os respectivos Modelos Laravel estão no singular. As propriedades mágicas estão algumas no plural e outras no singular. Graças a Jared Crapo, que verteu para Java a classe Inflector da biblioteca de ruby ActiveSupport, esse assunto foi resolvido.

```

String[] singulars = {"content", "gallery"};
String[] plurals = {"agencies", "users"};

System.out.println("Singular to plural:");
for(String singular : singulars)
    System.out.printf("%t%s: %s%d", singular, Inflector.pluralize(singular));

System.out.println("Plural to singular:");
for(String plural : plurals)
    System.out.printf("%t%s: %s%d", plural, Inflector.singularize(plural));

// Singular to plural:
// content: contents
// gallery: galleries
// Plural to singular:
// agencies: agency
// users: user

```

Código 14 – Bibliotecas Java – Inflector – Singular vs. Plural

As variantes recorrentes são os pares de terminações {'', 's'} e {'y', 'ies'}. Porém, a coleção de regras por terminação é mais extensa. A classe tem ainda uma coleção de irregulares (e.g. {'person', 'people'}, {'man', 'men'} e {'child', 'children'}) e de invariáveis (e.g. 'equipment', 'information' e 'rice'). Independentemente do alcance da classe e da necessidade de tanta regra a verdade é que a segurança do problema resolvido para futuros casos apareceu.

Nota: a classe não conseguiu verter a expressão 'brand tone(s) of voice', caso que constitui a única exceção ao uso da mesma.

5.3.1.3.3. Guava

Para a produção do código PHP, cada detalhe foi transformado em classe para que os vários métodos toString combinados expelissem o código final com menor esforço. A biblioteca Guava deu uma ajuda larga no que toca a manipulação de texto e coleções

Manipulação de texto

O código PHP foi produzido pelo Java em mais do que um formato. Para conversão entre formatos, usou-se esta biblioteca.

Os formatos usados foram:

- Snake case: e.g. mais_que_uma_palavra;
- Lower camel: e.g. maisQueUmaPalavra;
- Upper camel: e.g. MaisQueUmaPalavra;

Na BD, as tabelas e colunas: snake case.

No PHP:

- Classes: upper camel;
- Métodos: lower camel;
- Atributos: snake case;

Guava traduz estes formatos com alguma facilidade:

Tabela 42

```
static import com.google.guava.Format.*
String tableName = "post_comments";

CaseFormat snakeCase = from(LOWER_UNDERSCORE);

String phpClassName = snakeCase.to(UPPER_CAMEL, tableName); // PostComments

String phpMethodName = snakeCase.to(LOWER_CAMEL, tableName); // postComments
```

Código 15 – Bibliotecas Java – Guava – Conversão de notação de texto

Repare-se que o Guava chama lower underscore ao snake case, mas este tem muito mais pinta. Fica a lista completa dos formatos suportados pelo Guava:

Tabela 43

Nome Guava	Alias	Exemplo	Aplicações
Lower underscore	Snake case	user_photos	Tabelas Colunas Laravel magic properties
Upper underscore		USER_PHOTOS	Constantes Enumerações
Lower camel		userPhotos	Métodos
Upper camel	Pascal Case	UserPhotos	Classes
Lower underscore	Train case	user-photos	HTML id CSS classes

Manipulação de coleções

O guava ajuda ainda a compilar coleções de objectos em texto através da class Joiner que o seguinte funcionamento genérico:

```
return Joiner.on("; ").skipNulls().join("Harry", null, "Ron", "Hermione").
```

Código 16 – Bibliotecas Java – Guava – Coleções

Que devolve o texto: "Harry; Ron; Hermione".

A título de exemplo do proveito tirado pelo (ab)uso desta classe, apresenta-se, de seguida, o código para escrever uma classe PHP.

Seja o próximo código, a estrutura Java que representa uma classe PHP, admitindo a implementação prévia e conveniente das estruturas {Import, PhpDoc, Visibility, PhpClass, PhpInterface, PhpAttribute e PhpMethod}:

```
class PhpClass{
    List<Import> imports;
    PhpDoc doc;
    Visibility visibility;
    boolean isAbstract;
    String name;
    PhpClass parentClass;
    List<PhpInterface> interfaces;
```

```

List<PhpAttribute> attributes;
List<PhpMethod> methods;

// (methods)
}

```

Código 17 - Bibliotecas Java – Guava – Coleções – Modelo de classe PHP

Este é um modelo simplificado e limpo para o propósito de exemplo.

Para a produção da assinatura da classe (a primeira linha até à abertura da chaveta).

```

public String getSignature(){
    return Joiner
        .on(" ")
        .skipNulls()
        .join(
            visibility == null ? null : visibility,
            isAbstract ? "abstract" : null,
            "class",
            name,
            baseModel == null ? null : "extends " + parentClass.name,
            interfaces.isEmpty() ? null : "implements " + Joiner.on(", ").join(interfaces)
        );
}

```

Código 18 - Bibliotecas Java – Guava – Coleções – Assinatura de uma classe PHP

Almejando produzir as variantes necessárias para escrever as seguintes classes PHP:

```

class Empty {}

public abstract class Mammal extends Tetrapod {}

class User extends Eloquent implements UserInterface, ReminderInterface {}

```

Código 19 - Bibliotecas Java – Guava – Coleções – Assinatura de classes PHP

Agora, para o código interno (entre chavetas):

```

public String getInnerCode(){
    List<PhpClassMember> members = new ArrayList<>(attributes);
    members.addAll(methods);

    return Joiner
        .on("\n\n")
        .join(members);
}

```

Código 20 - Bibliotecas Java – Guava – Coleções – Membros da classe PHP

E finalmente a compilação final:

```

public String toString(){
    return Joiner
        .on("\n")
        .skipNulls()
        .join(
            imports.isEmpty() ? null : Joiner.on("\n").join(imports) + "\n",
            doc,
            getSignature() + "{",
            getInnerCode(),
            "}"
        );
}

```

Código 21 - Bibliotecas Java – Guava – Coleções – Classe PHP completa

Eis portanto a justificação do uso da biblioteca com seis utilizações numa única chamada sem ciclos ou condições.

5.3.1.3.4. Gson

A reflexão da BD foi implementada de forma independente do escritor de código, de forma a que se pudesse gravar e carregar o estado intermédio e, assim, minimizar as reflexões idênticas à BD.

Optou-se por serialização em texto pela ausência de dados sensíveis (é apenas estrutura) e para se observar o progresso da implementação desta ferramenta.

Dos vários modelos de serialização em texto, preferiu-se JSON por serializar de forma compacta mas legível, como no seguinte exemplo aproveitado da Wikipedia:

```
{
  "Alunos": [
    {
      "nome": "João",
      "notas": [ 8, 9, 7 ]
    },
    {
      "nome": "Maria",
      "notas": [ 8, 10, 7 ]
    },
    {
      "nome": "Pedro",
      "notas": [ 10, 10, 9 ]
    }
  ]
}
```

Código 22 – Notação JSON

A biblioteca GSON serializa e de-serializa de forma simples e funcional:

```
// Criação base do serializador

Gson gson = new GsonBuilder().create();

// Serialização

new PrintWriter(new FileWriter(FILENAME)).print(gson.toJson(dataModel));

// De-serialização

DataModel dataModel = gson.fromJson(new FileReader(file), DataModel.class);
```

Código 23 - Bibliotecas Java – GSON – Serialização em JSON – Base

Assim construído, o JSON sai compactado todo para a mesma linha. Um exemplo de configuração adicional é:

```
Gson gson = new GsonBuilder()
    .setPrettyPrinting()
    .create();
```

Código 24 - Bibliotecas Java – GSON – Serialização em JSON – Pretty Print

Esta configuração deixa o JSON quase como no exemplo da Wikipedia (esta forma estraga mais linhas).

A grande preocupação, na produção do código, é garantir a ausência de referências circulares (e.g. { dataModel.entities, entity.dataModel }, {entity.attributes, attribute.entity}) pois o GSON não as resolve. No entanto, não há castração do programador, uma vez que o GSON ignora atributos marcados com o operador ‘transient’.

Exemplo:

```
class DataModel {
    Entity[] entities;
}

class Entity {
    transient DataModel dataModel;
    Attribute[] attributes;
}

class Attribute {
    transient Entity entity;
}
```

Código 25 - Bibliotecas Java – GSON – Serialização em JSON – Esconder atributos

5.3.1.4. Produção do Código

Nota: o Laravel tem *predefinições* para muita estruturação da BD (nomes de tabelas, colunas, chaves, etc.); todos podem ser reconfigurados, mas privilegiou-se a obediência em prole da redução de código.

5.3.1.4.1. Modelo básico

Um modelo Laravel tem, na sua implementação, o seguinte código-base:

```
class User extends Eloquent {
}
```

```
$user1 = new User;
$user2 = User::find(1);
$user3 = Auth::user();
```

Código 26 – Laravel Model – Código-base e possíveis instanciações

Nota: acima não se identifica o nome da tabela; o Laravel assume a existência de uma tabela com o nome no plural e em “snake case” (minúsculas unidas por traço-baixo ‘_’ - e.g. “user_comments”). Privilegiou-se essa predefinição definindo o nome das tabelas como o Laravel espera encontrá-los. A alternativa passa por explicitar o nome da tabela como no exemplo seguinte:

```
class User extends Eloquent {
    protected $table = 'my_users';
}
```

Código 27 – Laravel Model – Com identificação de tabela

5.3.1.4.2. Colunas

Quanto às colunas, deve-se explicitar ao Laravel que colunas devem ser tratadas como atributos através da propriedade ‘fillable’:

```
class User extends Eloquent {
    protected $fillable = array('username', 'password');
}
```

```
$user = new User(array(
    'username' => 'user31',
    'password' => 'secret'
));
```

Código 28 – Laravel Model – Colunas fillable

Excluem-se as colunas automáticas, de relação e de suporte ao Laravel (e.g. timestamps).

5.3.1.4.3. Timestamps

Outra assunção do Laravel é que as tabelas têm as seguintes colunas para guardar momentos (timestamp) de edição:

- created_at: instante da inserção;
- inserted_at: instante do último update.

Para desligar esse efeito, recorre-se à alteração como ao seguinte exemplo:

```
class User extends Eloquent {
    public $timestamps = false;
}
```

Código 29 – Laravel Model – Para dispensar timestamps

5.3.1.4.4. Relações um-para-um

Nas relações um-para-um, tal como nas relações um-para-muitos, o Laravel presume que se trata de uma relação de posse com direcção do dono (en: owner) para a pertença (en: belong), em que a tabela da pertença é a que contém a FK para a outra. Do ponto de vista do MySQL, a relação tem direcção, mas não tem significado. Depois do modelo construído, o significado dilui-se.

```
class Entity extends Eloquent {

    public function page()
    {
        return $this->belongsTo('Page');
    }

}

$entity = Entity::find(1)
$page = $entity->page; // or...
$page = $entity->page()->first(); // same thing
```

Código 30 – Laravel Model – Relações Um-para-Um

Entity recebeu uma propriedade dinâmica através do método page (em lowerCamelCase), que constitui um “Inner Join” entre as tabelas “entities” e “pages” e que será modelo para a construção, em runtime, da attribute ‘page’ (em snake_case).

Para a perspectiva inversa da relação.

```
class Page extends Eloquent {

    public function entity()
    {
        return $this->ownsOne('Entity');
    }

}

$page = Page::find(1);
$entity = $page->entity;
```

Código 31 – Laravel Model – Relações um para um – Perspectiva inversa

Como se pode observar, a direcção e os significados de posse diluíram-se:

- navegação: existe nos dois sentidos;
- significados: não se distingue, se é a entity que possui a page ou a page que possui o entity.

O Laravel presume a existência de uma coluna ‘entity_id’ na tabela ‘pages’ FK da coluna ‘id’ em ‘entities’. Dá para contrariar todas as presunções. No entanto, estas questões não foram relevantes para este projecto

5.3.1.4.5. Relações um-para-muitos

A diferença, entre esta relação e a anterior, está na perspectiva do owner que passa a ter muitos (en: has many).

```
class User extends Eloquent {

    public function agencies()
    {
        return $this->hasMany('Agency');
    }

}
```

```

$user = Auth::user()->get();
foreach($user->agencies as $agency){
    // ...
}

```

Código 32 – Laravel Model – Relações um para muitos

A perspectiva inversa da relação só identifica um owner.

```

class Agency extends Eloquent {

    public function creator()
    {
        return $this->belongsTo('User');
    }

}

```

```

$agency = Agency::find(1)
$user = $agency->creator

```

Código 33 – Laravel Model – Relações um para muitos – Perspectiva inversa

Desta vez o Laravel procura uma coluna 'creator_id' na tabela 'brands'

5.3.1.4.6. Relações muitos-para-muitos

```

class User extends Eloquent {

    public function roles()
    {
        return $this->belongsToMany('Role')->withPivot('active');
    }

}

```

```

$roles = $user->roles()

```

```

class Role extends Eloquent {

    public function users()
    {
        return $this->belongsToMany('User')->withPivot('active');
    }

}

```

```

$users = $role->users

```

Código 34 – Laravel Model – Relações muitos para muitos -

5.3.1.4.7. Herança

O Laravel não implementa herança. Cada entidade corresponde a um modelo e cada modelo a sua classe. No paradigma OO (orientação a objectos), uma classe derivada corresponde a várias entidades o que não é aplicável aqui.

Abdicando do paradigma OO e respeitando o que realmente acontece na BD, a relação terá de ser implementada através de relações de posse e obter efeitos como os seguintes.

```

$user = Auth::user();
$entity = $user->entity;

$agency = Agency::find(1);
$entity = $agency->entity;

```

Código 35 – Laravel Model – Relações de Herança – Objectivo

No sentido oposto, o modelo-base é suposto não conhecer os modelos seus derivados e os esquemas já apresentados exigem o conhecimento das entidades relacionadas.

No entanto o Laravel tem o conceito de relação polimorfa que faz quase o pretendido:

```
class Photo extends Eloquent {

    public function imageable()
    {
        return $this->morphTo();
    }

}

class Staff extends Eloquent {
    public function photos()
    {
        return $this->morphMany('Photo', 'imageable');
    }
}

class Order extends Eloquent {

    public function photos()
    {
        return $this->morphMany('Photo', 'imageable');
    }

}

$staff = Staff::find(1);
$photos = $staff->photos;

$order = Order::find(1);
$photos = $order->photos;

$photo = Photo::find(1);
if ($photo->imageable instanceof Staff){
    $staf = $photo->imageable;
}
else if ($photo->imageable instanceof Order){
    $order = $photo->imageable;
}
}
```

Código 36 – Laravel Model – Relações de Herança – Relações polimorfias

O exemplo acima mostra uma relação polimorfa de um-para-muitos. Para que este código funcione, na tabela “photos” tem de haver uma coluna com o nome “imageable_type” (a predefinição seria ‘photoable’) dum tipo de texto com um mínimo de 60 caracteres para guardar o nome do modelo (respetivamente “Staff” e “Order”).

Para implementar uma derivação, a relação será de um-para-um.

```
class Entity extends Eloquent {

    public function entityable()
    {
        return $this->morphTo();
    }

}

class User extends Eloquent {

    public $timestamps = false;

    protected $touches = array('entity');

    public function entity()
    {
        return $this->morphOne('Entity', 'entityable');
    }

}
```

```

}
class Agency extends Eloquent {
    public $timestamps = false;
    protected $touches = array('entity');
    public function entity()
    {
        return $this->morphOne('Entity', 'entityable');
    }
}
}

$user = Auth::user();
$page = $user->entity->page;

$agency = Agency::find(1);
$screen_name = $agency->entity->screen_name;

$entity = Entity::find(1);
if ($entity->entityable instanceof User){
    $user = $entity->entityable;
}
else if ($entity->entityable instanceof Agency){
    $agency = $entity->entityable;
}
}

```

Código 37 – Laravel Model – Relações de Herança – Código final

Nota 1: A propriedade ‘timestamps’ iniciada a false serve para ‘users’ e ‘agencies’ dispensarem timestamps e respectivas colunas.

Nota 2: A propriedade ‘touches’ nos modelos ‘User’ e ‘Agency’ serve para que os respectivos registos em ‘entities’ atualizem os seus timestamps.

5.3.1.5. Modelos finais

Aqui, apresentam-se os modelos gerados para o diagrama que serviu de exemplo.

5.3.1.5.1. Page

```

class Page extends Eloquent {
    property $fillable = array('key');
    public function entity()
    {
        return $this->onwsOne('Entity');
    }
}
}

```

Código 38 – Laravel Model – Exemplo ‘Page’

5.3.1.5.2. Entity

```

class Entity extends Eloquent {
    property $fillable = array('screen_name');
    public function page()
    {
        return $this->belongsTo('Page');
    }
    public function entityable()
    {

```



```

        return $this->morphTo();
    }
}

```

Código 39 – Laravel Model – Exemplo 'Entity'

5.3.1.5.3. Agency

```

class Agency extends Eloquent {

    protected $fillable = array('trade_name', 'legal_name')

    public $timestamps = false;

    protected $touches = array('entity');

    public function creator()
    {
        return $this->belongsTo('User');
    }

    public function entity()
    {
        return $this->morphOne('Entity', 'entityable');
    }

}

```

Código 40 – Laravel Model – Exemplo 'Agency'

5.3.1.5.4. User

```

class User extends Eloquent {

    protected $fillable = array('username', 'password')

    public $timestamps = false;

    protected $touches = array('entity');

    public function roles()
    {
        return $this->belongsToMany('Role')->withPivot('active');
    }

    public function agencies()
    {
        return $this->hasMany('Agency');
    }

    public function entity()
    {
        return $this->morphOne('Entity', 'entityable');
    }

}

```

Código 41 – Laravel Model – Exemplo 'User'

5.3.1.5.5. Role

```

class Role extends Eloquent {

    protected $fillable = array('designation')

    public function users()
    {
        return $this->belongsToMany('User')->withPivot('active');
    }

}

```

```
}  
}
```

Código 42 – Laravel Model – Exemplo 'Role'

5.3.2. Views

Uma vista é uma área visível. Neste trabalho, três grupos podem ser distinguidos:

1. Pode ser completa (e.g. alguma página inteira),
2. Template
3. Parcial (e.g. menus, controlos, painéis).

A produção das Vistas é feita em HTML e CSS, essencialmente produzidos pelo Laravel (chamado de blades) ou pela framework Bootstrap.

Refletem os contextos da plataforma, i.e., devem mudar drasticamente entre contextos e manterem-se coesas dentro de cada contexto, de forma à participação dos seus utilizadores ser intuitiva.

O contexto mais alargado é o de sessão ou de conta, o que sugere dois *sites* completamente distintos: um para os anónimos e outro para os membros. Um pormenor radical entre estas variantes é o menu vertical estar à direita, para os anónimos, e à esquerda, para os utilizadores autenticados. Para estes, as opções de sessão passam para a barra de sessão (horizontal no topo).

Praticamente não há vista implementada no cliente. São conhecidas as desvantagens da produção de vistas no servidor:

- Centraliza o processamento: podia cada cliente (navegador) tratar do que conseguisse;
- Impede a cache do cliente: o resultado de um pedido PHP não é guardado para que se possa reutilizar no futuro (ao contrário de ficheiros estáticos como: html, js, css, json, imagens, etc.).

Que são ofuscadas pelas vantagens:

- Laravel implementa a maioria do código PHP;
- Os IDEs gratuitos são melhores para PHP do que para Javascript;
- Integram melhor com os modelos e controladores do que no cliente (e.g. o cliente não tem noção da estruturação dos modelos que hão-de chegar do servidor).

5.3.2.1. Anónimos

Os três requisitos fundamentais para a área pública são:

1. Apresentação dos serviços
2. Aquisição de serviços
3. Autenticação

5.3.2.1.1. Apresentação dos serviços

Este espaço não foi explorado por não figurar nos requisitos nem no contexto deste projeto.

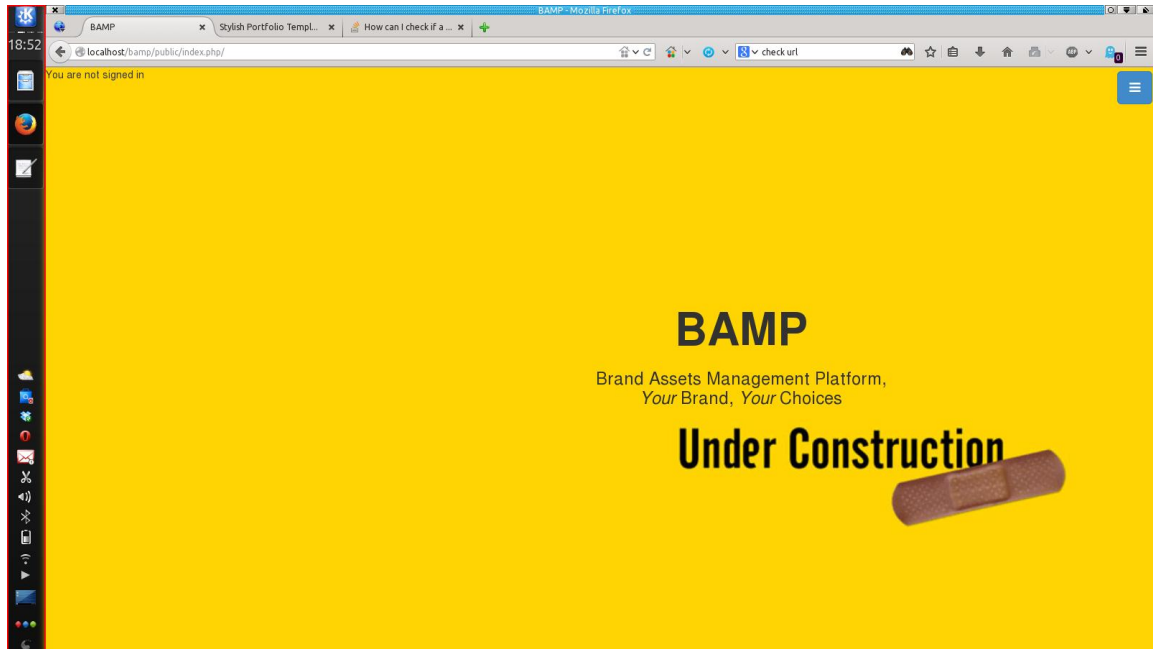


Figura 3 – Página pública

5.3.2.1.2. Aquisição dos serviços

O serviço principal a ser adquirido neste espaço é a inscrição de uma Agência na plataforma. Em tudo semelhante à criação de conta, incluída no ponto seguinte (5.3.2.1.3 - Autenticação) com a diferença de haver aprovação automática pelo sistema (após pagamento ou outra forma de acordo).

Os seguintes fazem parte dos campos do formulário de inscrição de uma agência:

- Identificação do BASA (será o próprio);
- Identificação da Agência;

5.3.2.1.3. Autenticação

Barra vertical na lateral direita com de aquisição de acesso a utilizadores anónimos.

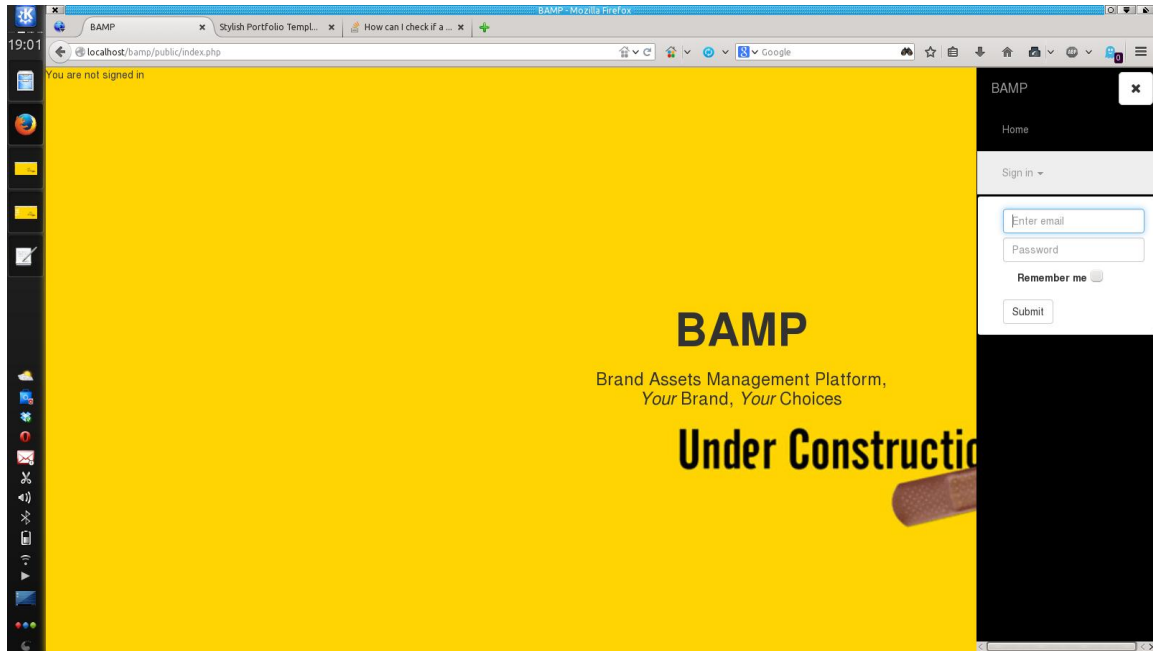


Figura 4 – Página pública com barra vertical direita activa para autenticação na plataforma

Operações principais:

- Autenticação
- Registo
- Recuperação de acesso

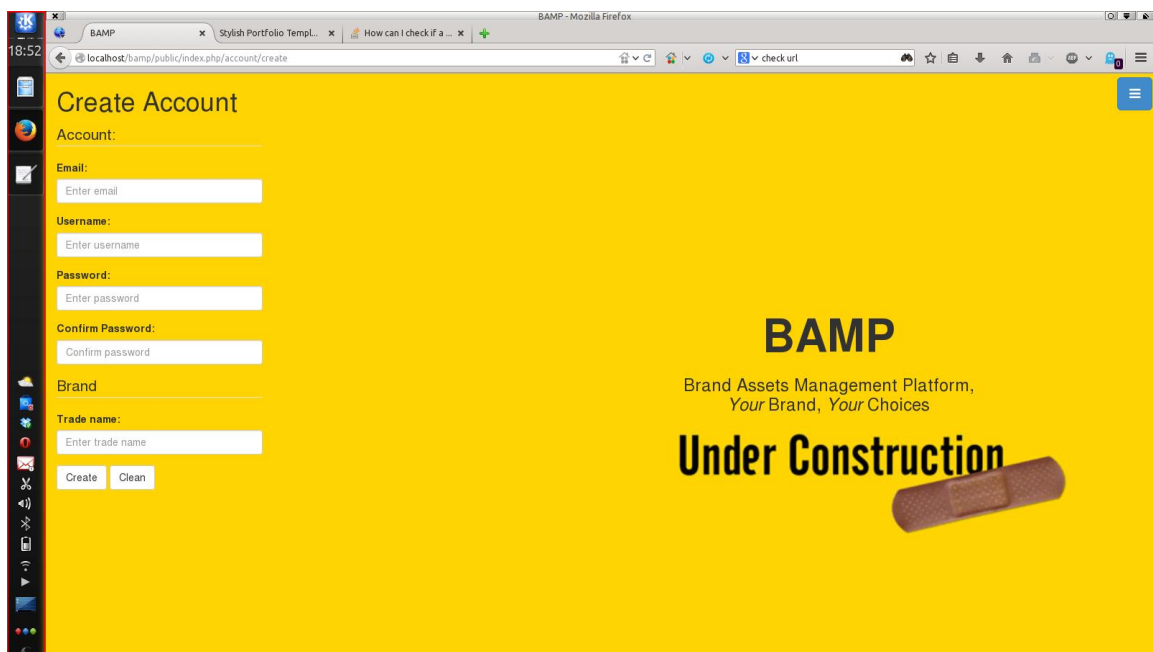


Figura 5 – Página pública com formulário de criação de conta para utilizadores *Freemium*

5.3.2.2. Autenticados

Após o início de sessão.

5.3.2.2.1. Barra de Sessão

Barra de ferramentas (en: toolbar) horizontal no topo da página com convites, notificações e operações relativas ao utilizador autenticado.

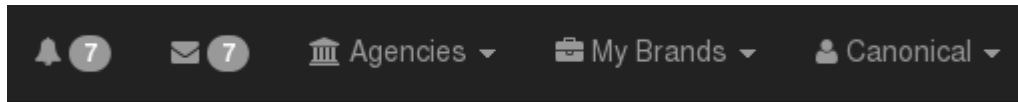


Figura 6 – Barra de ferramentas com operações visíveis para os vários utilizadores

Imagem

5.3.2.2.1.1. Notificações

Qualquer informação apresentada a um utilizador, sem quaisquer consequências. Exemplos:

- Respostas a convites (apresentados de seguida);
- Avisos de partilha;

5.3.2.2.1.2. Convites

São pedidos de realização de uma determinada ação através de confirmação; são exemplos de convite:

- Inscrição de Agência: (a um BAMP Admin) que crie uma agência em nome do requerente;
- Alteração do Limite de Marcas: (a um BAMP Admin) que aumente o limite de marcas associado ao requerente;
- Aceitar uma função:
 - **BASA:** por um BAMP Admin na inscrição de uma Agência;
 - **BOU:** por um BASA na inscrição de uma Marca numa Agência;
 - **BU:**
 - por um BOU, enquadrado numa Marca de uma Agência;
 - por um BOA, enquadrado numa Marca do BOA

Operações genéricas disponíveis sobre convites:

Tabela 44

Ação	Resposta	Estado	Previne adicionais
Aceitar	Aceite	Lido	Sim
Recusar	Recusado	Lido	Não
Suspender	Pendente	Lido	Sim
Ignorar	---	Oculto	Sim
Apagar	---	Apagado	Não
Arquivar	---	Lido	Sim

Acesso a consultas

- seu perfil
- As suas Agências
- As suas Marcas

- Ajuda

Operações de sessão e de conta

- Término de sessão
- Edição de conta

5.3.2.2.2. Menu de Navegação

Já autenticado, aparece o contexto de página ou entidade. A área que reflete o contexto de entidade é o menu esquerdo que mostra o avatar (com nome e ligação à página principal da entidade) e o consoante plano de navegação.

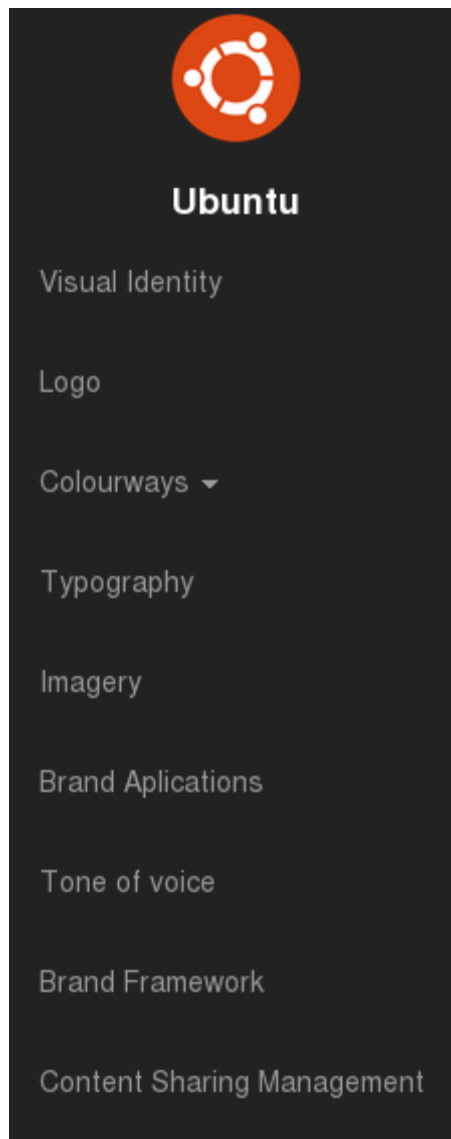


Figura 7 – Menu de navegação no contexto de marca

Tabela 45

Utilizador	Agência	Marca
------------	---------	-------

<ul style="list-style-type: none"> • Brands • Agencies 	<ul style="list-style-type: none"> • Brands 	<ul style="list-style-type: none"> • Visual identity • Brand applications • Tones of voice • Brand Framework • Users • Groups
--	--	---

A informação relativa a uma entidade enquadra-se em dois tipos: propriedades (e.g. dados, perfil e contactos) e recursos (cada tipo de entidade tem os seus). A página principal serve para tratar do primeiro tipo enquanto o menu serve para navegar sobre o segundo.

5.3.2.2.3. Main Page

Os dados pessoais da entidade estão estruturados por tipo de entidade.

Tabela 46

Agency	Brand	User
<ul style="list-style-type: none"> • Name 	<ul style="list-style-type: none"> • Trade name • Legal name • Tax id 	<ul style="list-style-type: none"> • First name • Last name

Por esta razão, a página principal de uma entidade não pode ser completa tendo que enquadrar uma vista interna especializada a estes campos.

Além destes, há a informação de perfil e contatos, transversal ao tipo de entidade, permitindo implementar a restante vista.

Tabela 47

Perfil	Contactos
<ul style="list-style-type: none"> • Avatar • Display name 	<ul style="list-style-type: none"> • Addresses • Phones • Urls

Para edição destes dados. existe ainda um botão que troca esta vista pelo editor de perfil.

5.3.2.2.4. Profile editor

Página para edição de um perfil. Pode ser o perfil do próprio, de uma agência, do qual seja BASA, ou de uma marca do qual seja BASA ou BOA.

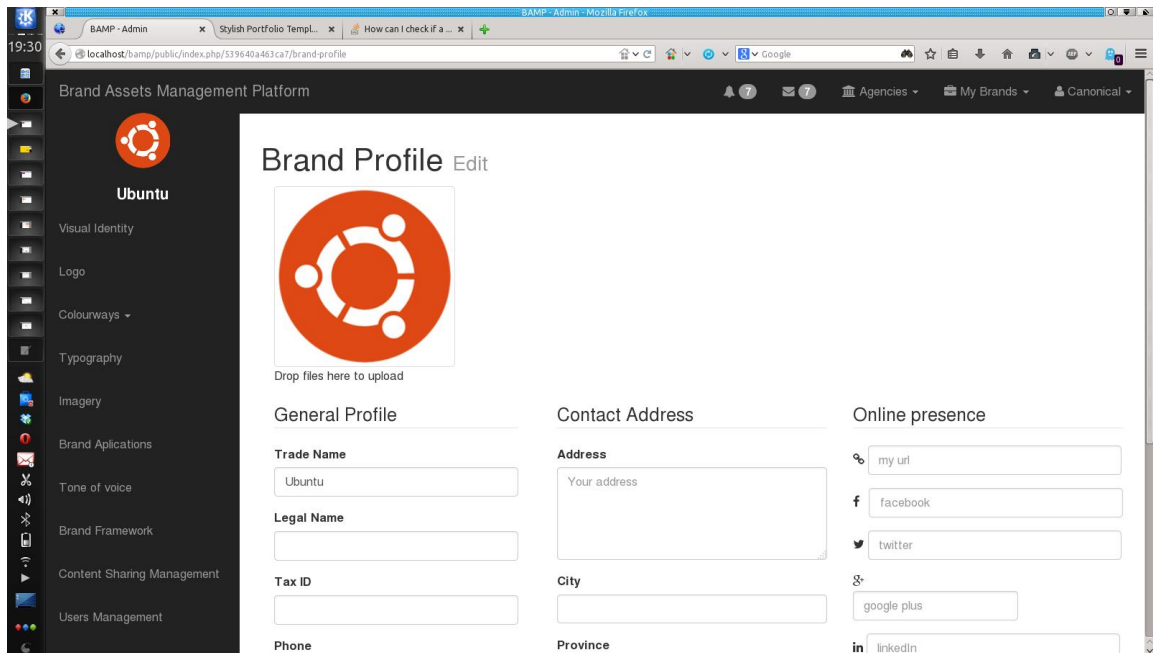


Figura 8 – Ecrã de edição do perfil de uma marca

5.3.2.3. Conteúdos

O restante espaço da página (área branca ao centro) é reservado para os subcontextos ou conteúdos da entidade corrente.

A estrutura que conterà estes conteúdos é produzida dinamicamente em PHP (sob a forma dos Blades do Laravel), mas só será preenchida pelo trabalho coordenado dos controladores PHP e Javascript, reduzindo a carga do servidor.

Segue-se a exposição dos vários cenários da plataforma.

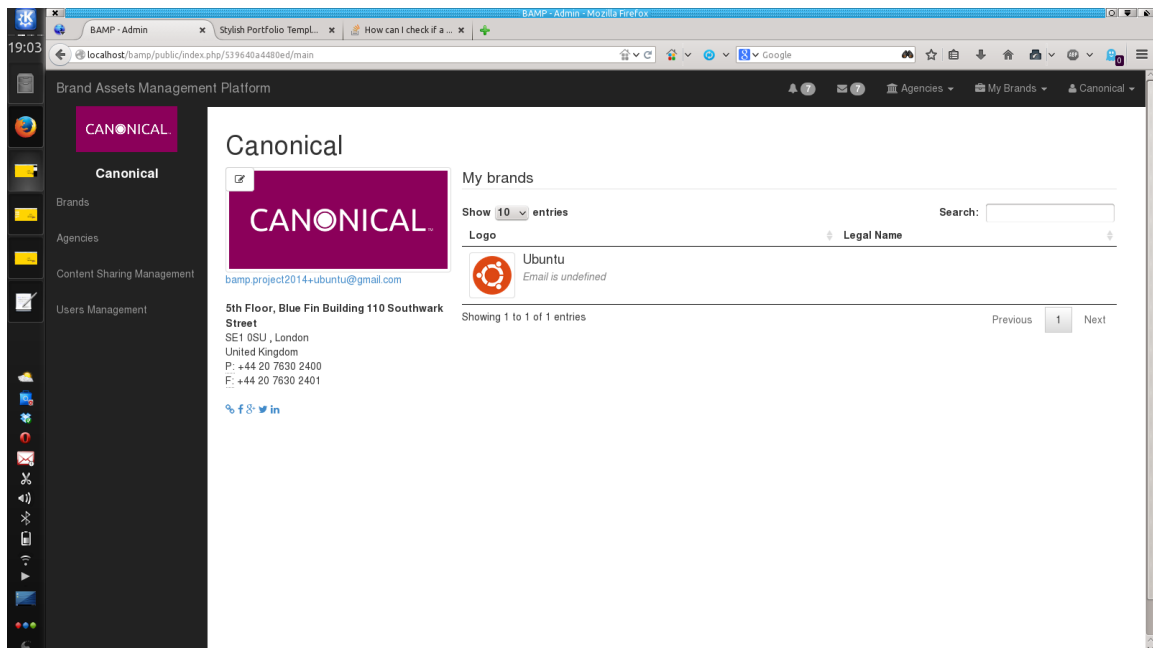


Figura 9 – Cenário de escolha de uma marca

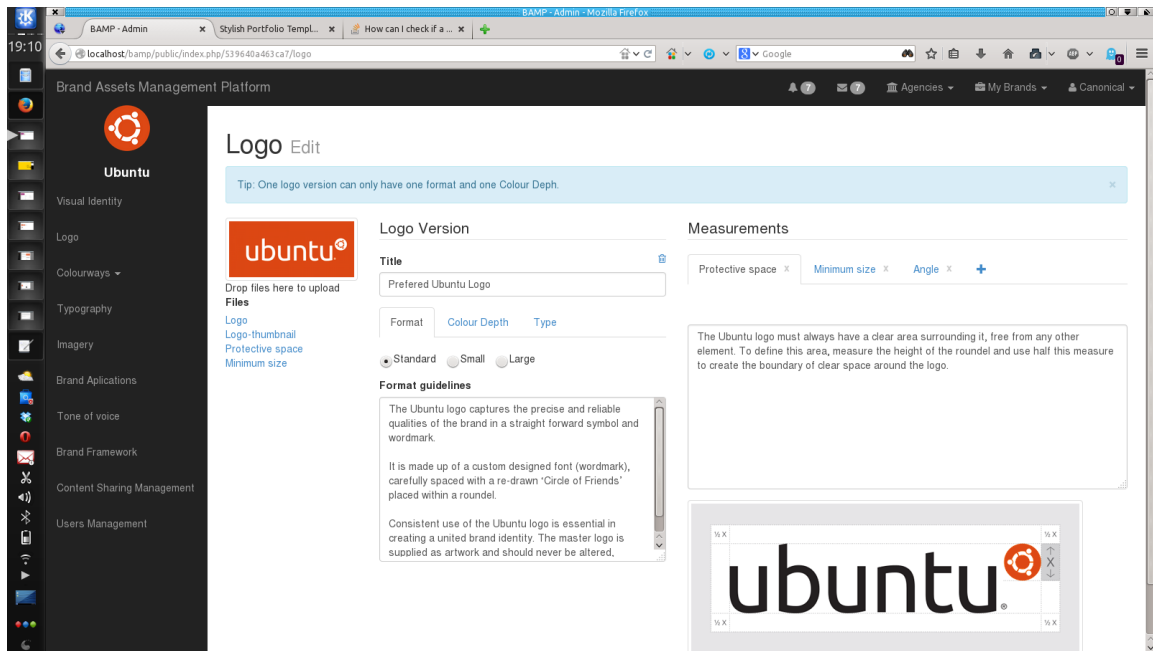


Figura 10 – Cenário de edição de Logos

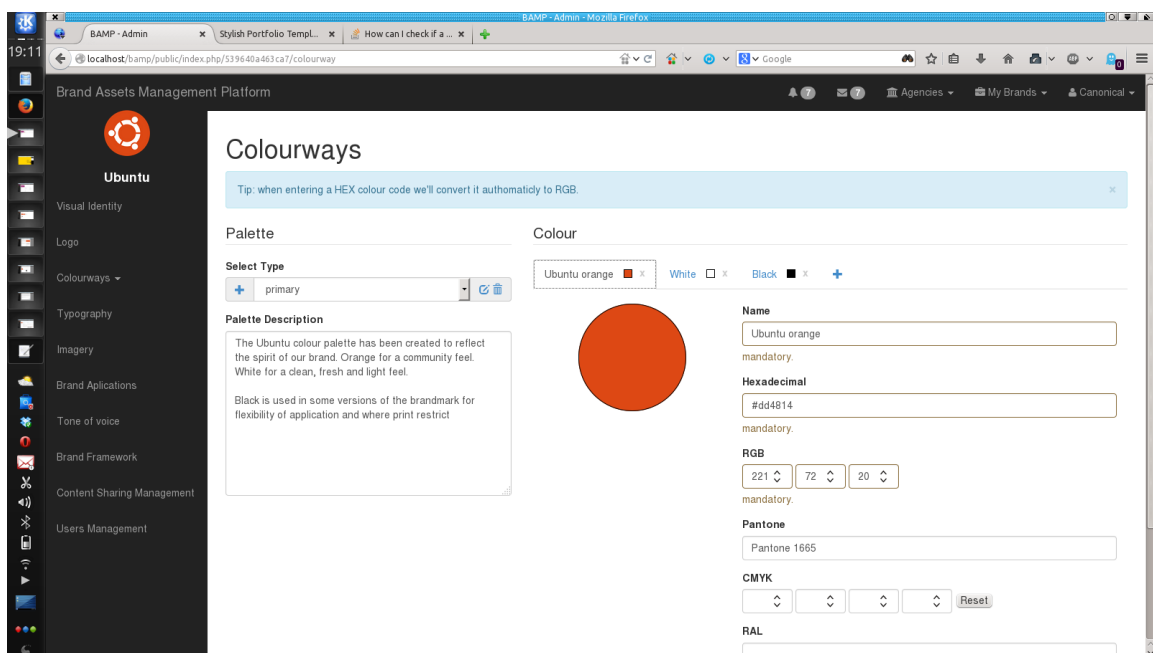


Figura 11 – Cenário de edição de Colourway

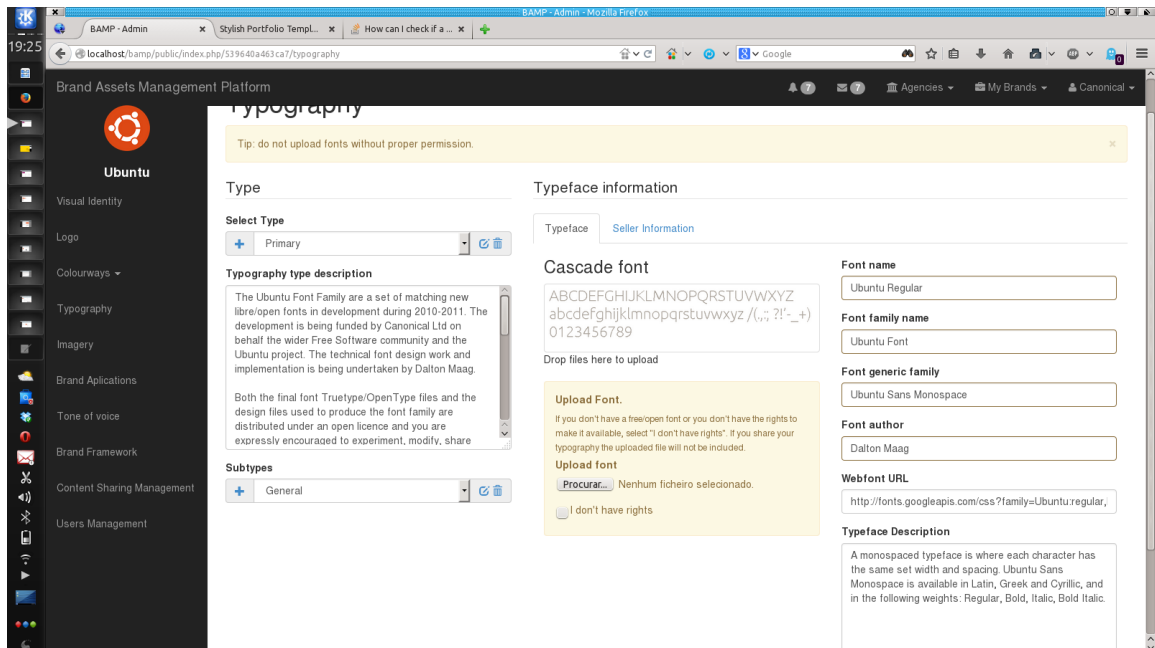


Figura 12 – Cenário de Edição da Tipografia

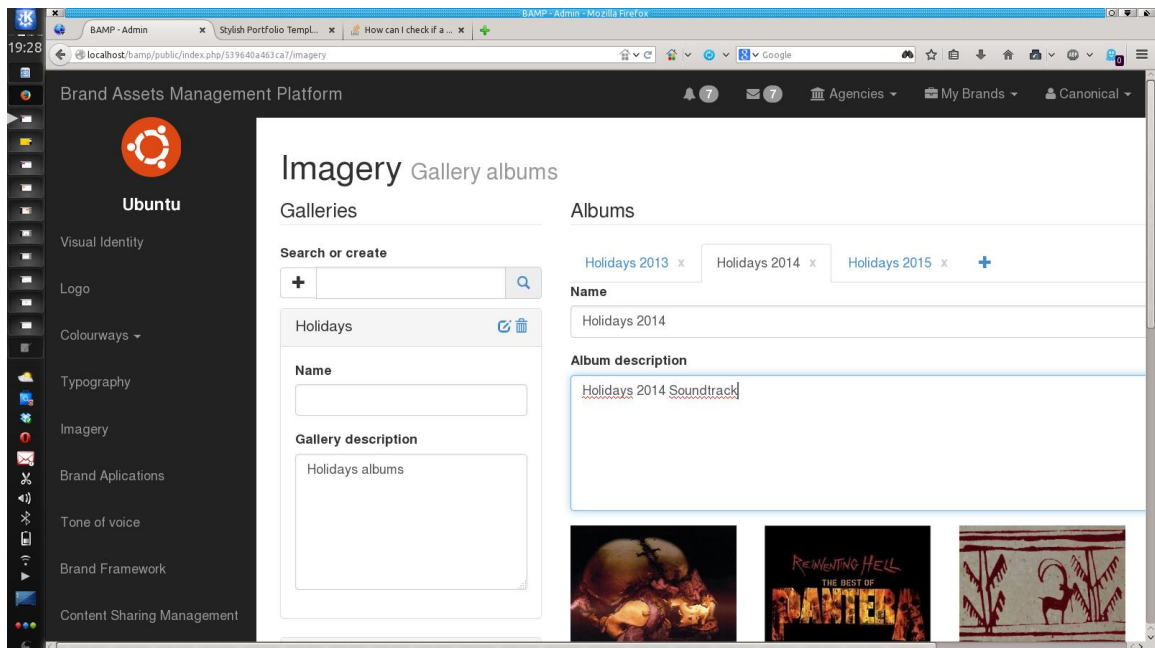


Figura 13 – Cenário de Edição de Galerias de Imagens

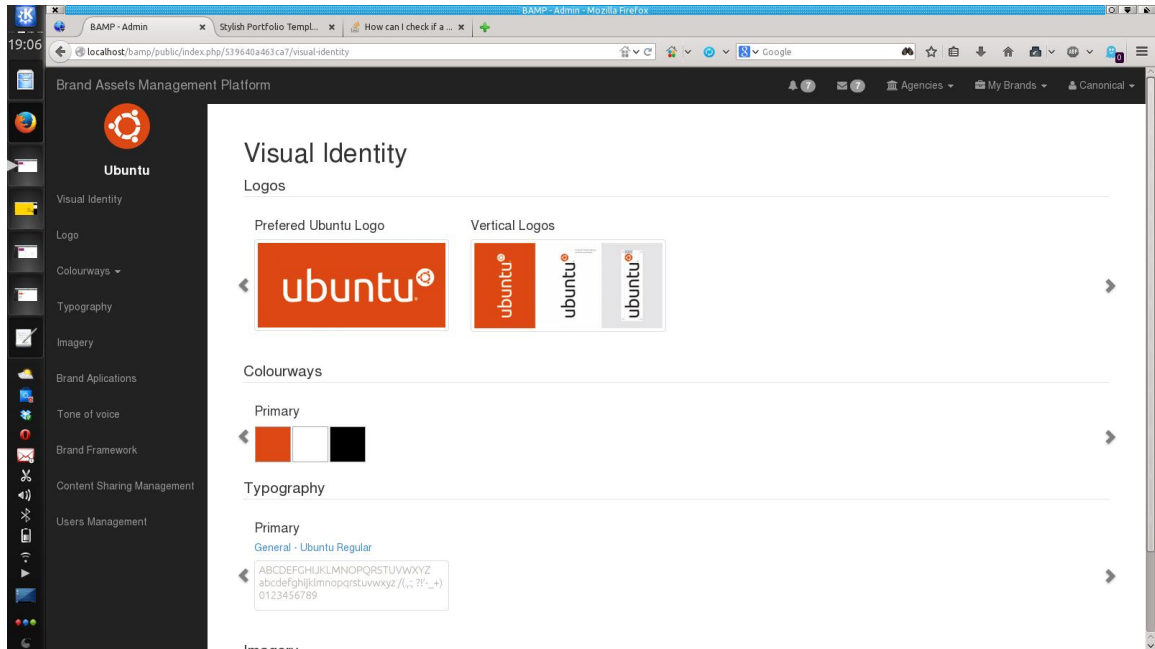


Figura 14 – Cenário de Visualização da Identidade Visual

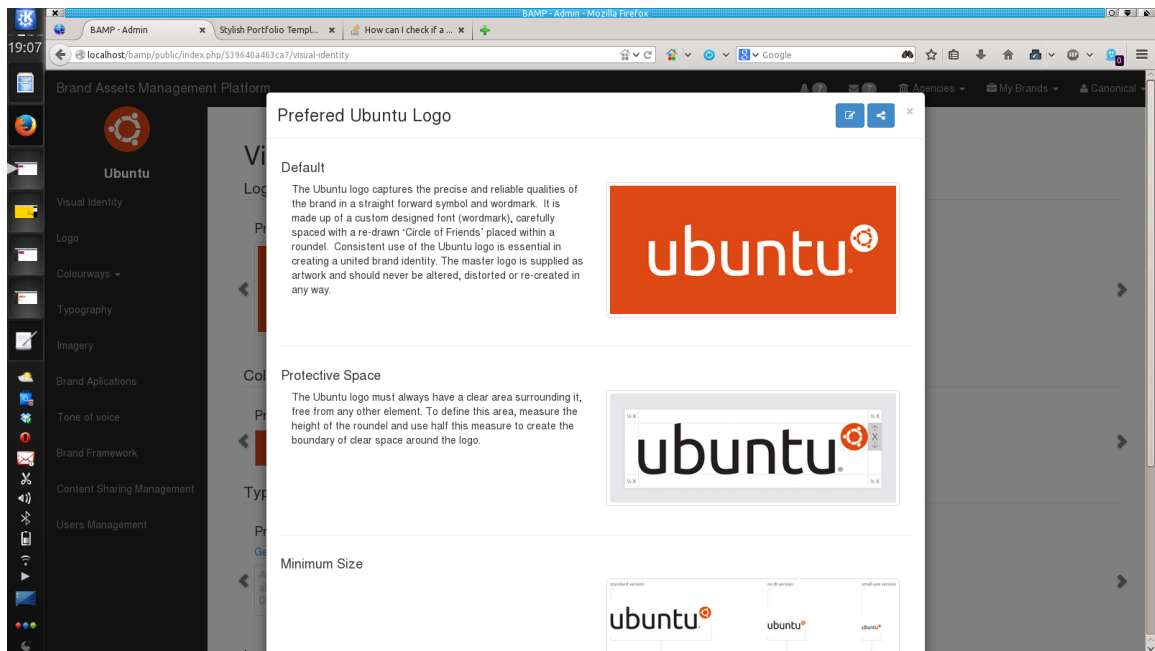


Figura 15 – Cenário de Visualização de Logos

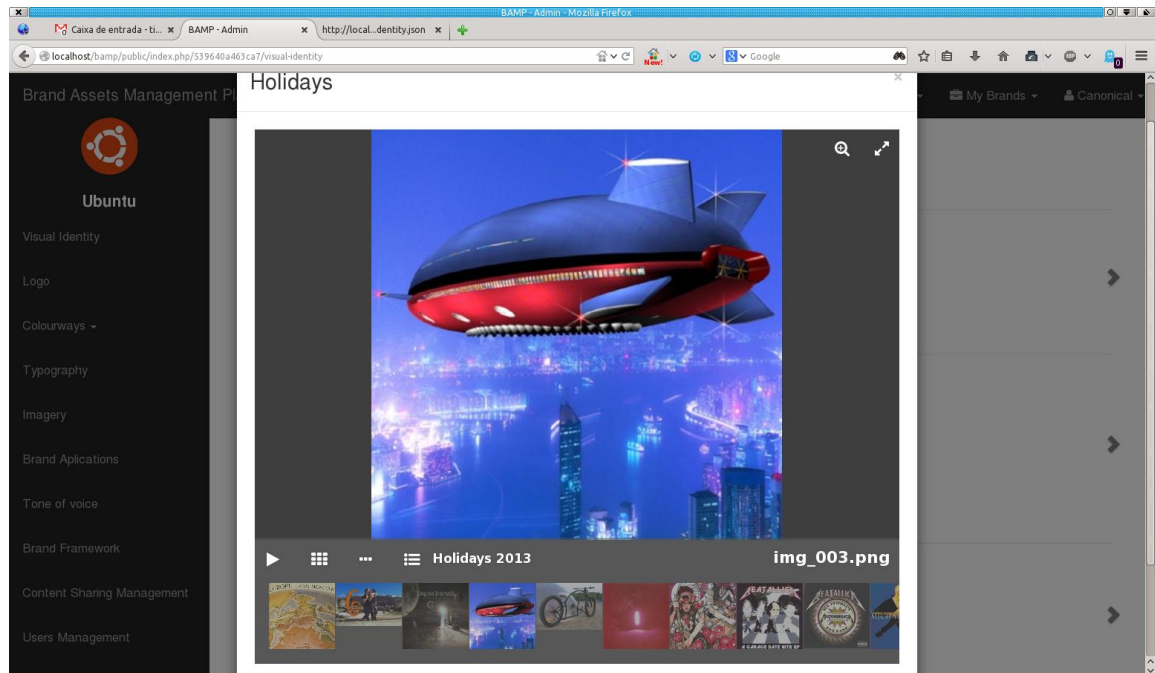


Figura 16 - Cenário de Visualização de uma Galeria de Imagens

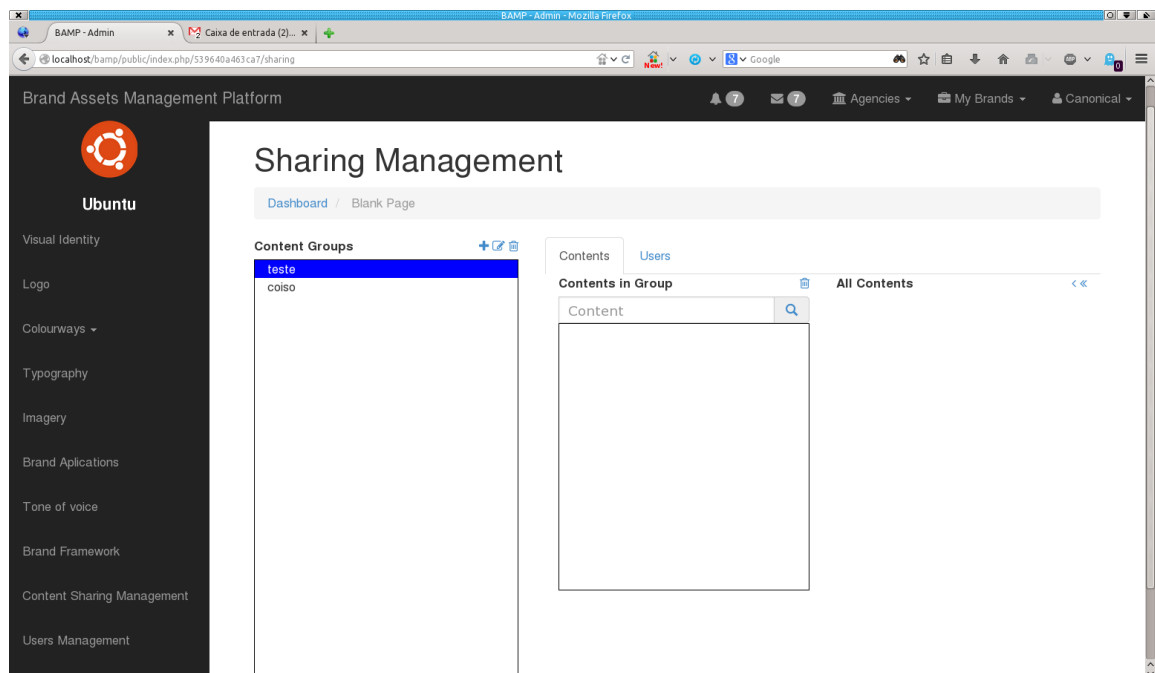


Figura 17 - Cenário de Partilha de Conteudos e Criação de Grupos de Conteudos por Utilizadores

5.3.3. Controllers

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

Não há relação direta entre controladores e vistas ou modelos. No entanto, entre vistas e controladores é fácil reconhecer afinidades entre alguns deles:

Tabela 48

Vistas	Controladores
Menu de Autenticação Barra de Sessão	Conta
Manu de Navegação – opções gerais	Entidade
Menu de Navegação – opções de Agência Subcontextos e conteúdos de Agência	Agência
Menu de Navegação – opções de Marca Subcontextos e conteúdos de Marca	Marca Logo, Colourway, Typography e Imagery

5.3.3.1. Base

Todos os controladores têm rotinas muito semelhantes entre si. Para evitar a repetição de código, foram implementadas técnicas de reutilização de código, tanto em Javascript como PHP.

5.3.3.1.1. Javascript

Os controladores foram colocados na pasta `js/bamp`, enquanto as rotinas transversais aos controladores, foram colocadas na pasta `js/dev`. Eis a estrutura:

Tabela 49

prototype.js Adição de métodos aos objectos Javascript	Number.hex()	Converte DEC 2 HEX
	Number.pad(n)	Acrescenta Zeros à esquerda
	String.replaceAll(search, new)	String.replace só substitui 1x
	String.trainCase()	Converte para train-case
	String.snakeCase()	Converte para snake_case
ui.js Rotinas para grafismos	formatIds(key)	Corrige ID e HREF
	loadFragments(url, id)	Carrega ficheiro como <i>template</i>
misc.js Rotinas avulsas	createPath(path)	Corrige paths vindos da BD
	isSet(arg)	Testa undefined e null
	hasText(arg)	Existe e tem texto
	is(arg)	Existe e é true

Os Widgets de terceiros são altamente configuráveis. No entanto, a configuração pretendida, dentro de cada tipo, é muito constante. Além disso, afectam sempre elementos gráficos, vulgarmente identificados por css selectores, html elements e jqueries. Para o tratamento dos

Widgets de terceiros, bem como para a produção de novos, há ainda a pasta `js/dev/widgets` conforme se apresenta de seguida:

Tabela 50

Base	Widget (genérico)	Interpreta argumentos. Todos os outros passam por aqui.
De raiz	Template	Retira um elemento HTML da vista, guarda-o como template e produz clones com ids (e respectivos hrefs) formatados.
	List	Trata um UL como um select multiline sem o contexto de FORM.
Bootstrap	Carousel	Slide horizontal de elementos HTML
	Modal	Popups cross-browser
	Tabs	Uma única área de conteúdos, com múltiplos páineis, cada um associado a um cabeçalho.
jQuery UI	Autocomplete	Autocomplete sobre um input do tipo text mediante fonte em json.
Dropzone	DropzoneJS	Drag and drop de imagens e ficheiros

5.3.3.1.2. Reutilização de PHP

A fim de poder ser usado ao longo de todos os controladores, a base destes foi preenchida com as seguintes rotinas:

<code>handleUploadedImage()</code>		Trata da submissão de imagens, redimensionamento, produção de miniaturas e colocação em pastas devidas
<code>responseJson(\$arguments)</code>		Entrega para devolução os argumentos convertidos para JSON, em estilo <i>pretty-print</i> e codificação em JSON para simulação de ficheiro.
Rotinas de teste avulsas	<code>burst(\$arguments)</code>	Forçar interrupção com argumentos em JSON
	<code>output(\$arguments)</code>	Gravação de dados em ficheiro (formato JSON).

5.3.3.2. Validação

Os pontos de validação dos formulários são tão mais importantes quanto mais próximos do efeito final (seja ação ou alteração da BD). Os principais pontos de validação são os seguintes:

- **HTML:** onde for possível (e.g. atributos `required` e `pattern` nos elementos do tipo `input`);
- **Javascript:** impedindo o formulário de se submeter para o servidor caso não valide;
- **PHP:** com acesso a informação sensível, de sessão ou de aplicação acrescentando validação adicional;
- **MySQL:** através de triggers que invalidem ou adaptem alterações caso seja necessário (e.g. preenchimento de campos redundantes como totais, nome visível ou logo do tipo 'standard' como avatar de marca).

Há motivos para validar cada elemento em cada ponto, assim como há motivos para validação em todos com custos de desempenho. No entanto, a exaustão da validação em cada ponto possível é demasiado extensa levando a sobrecarga nas tarefas do projeto.

5.3.3.3. Controladores Javascript e PHP

Dado que as vistas destes contextos são muito carregadas, cada controlador PHP fica encarregue de encapsular a informação em JSON para ser tratada assincronamente pelo controlador Javascript.

O tratamento das vistas pelos controladores tem o seguinte percurso genérico:

- As vistas são carregadas inicialmente com elementos gráficos exemplares (aqui chamados de *templates*) de onde sairão cópias em função da informação que há-de ser recolhida;
- Quando o controlador Javascript é lançado, retira esses *templates* da vista e arruma-os em cache;
- Quando pronto, pede então assincronamente os dados ao controlador PHP que os entrega em JSON;
- Correndo essa informação, o Javascript produz tantos clones de *templates* quanto os necessários, preenche-os com os dados adequados e coloca-os onde é devido.

Este processo permite que o grosso da vista seja implementado no cliente pelo navegador com custos reduzidos de transferência de dados e minimizando o processamento do servidor.

5.3.3.4. Controlador de conta

O PHP mantém cache dos dados relativos do utilizador autenticado naquele posto (e.g. credenciais, conta e privilégios), de forma a melhorar a resposta das vistas.

Responde a qualquer operação ou consulta de conta iniciada em qualquer vista, tanto para anónimos como para membros, como em páginas específicas para configuração daqueles dados:

- Início de sessão;
- Término de sessão;
- Criação de conta;
- Ativação de conta;
- Recuperação de acesso;
- Retorno da chave do utilizador;
- Convites e Notificações.

Quanto ao tratamento de convites e notificações, por se tratarem de elementos que necessitam de atualização, é responsabilidade do controlador reler essa informação periodicamente procedendo a eventuais ações. Para libertar o servidor desta carga, fica o

controlador Javascript com o papel de assincronamente pedir e recolher os dados, com eventual atualização da vista:

Chamada periódica à função de leitura de sessão:

```
setInterval(function() { loadSession() }, 2000); // 2000 milliseecs = 2 secs
```

Código 43 – Javascript Controller – Chamada periódica à leitura de sessão

Chamada AJAX ao events.json

```
function loadSession(){
    $.getJSON(page + "/session.json", function (events) {
        if (typeof session.notifications !== 'undefined')
            handleNotifications(events.notifications);

        if (typeof session.requests !== 'undefined')
            handleRequests(session.requests);
    })
}
```

Código 44 – Javascript Controller – Invocação assíncrona do serviço de consulta

O código apresentado é um modelo incompleto do código implementado. Não se apresenta as funções de tratamento de cada tarefa (handleNotifications e handleInvitations).

Enquanto o controlador PHP se limita a encapsular aqueles dados em JSON aquando um pedido:

```
function getJson($page)
{
    $result = [];

    // find user_id
    $user_id = Auth::user()->id;

    // query for notifications
    $notifications = Notification::where('user_id', '=', $user_id)
        ->get();

    if ($notifications)
        $result["notifications"] = $notifications;

    // query for invitations
    $requests = Auth::user
        ->roles()
        ->whereIn('state', array('requested', 'suspended'))
        ->get();
    if ($requests)
        $result["requests"] = $requests;

    return self::responseJson($result)
}
```

Código 45 – Laravel Controller – Entrega das notificações e pedidos em JSON

5.3.3.5. Controlador de página ou entidade

O contexto a que este controlador se refere é definido pela entidade dentro da qual o utilizador escolheu navegar. Trata de manter pontos estáticos de acesso aos campos mais usados de uma entidade, nomeadamente:

- Chave de página: presente no URL;
- Tipo de entidade;
- Avatar;
- Nome de apresentação (consoante a entidade);

- Contactos
- etc.

Rege-se pelos princípios de inicialização atrasada (en: lazy initialization), em que um campo só é inicializado no momento da primeira consulta, dado que:

- nem todos os seus campos precisam de ser usados para todas as páginas;
- alguns campos podem ser utilizados massivamente nalguma página (e.g. caso da chave de página (en: key) para produzir endereços de reencaminhamento dentro do contexto da entidade).

```
class PageController extends BaseController
{
    private static $key;

    // other fields

    public static function key()
    {
        if (!self::$key) {
            $path = $_SERVER['PHP_SELF'];
            $splits = explode("/", $path);
            self::$key = $splits[count($splits) - 2];
        }
        return self::$key;
    }

    // other methods
}
```

Código 46 – Laravel Controller – Lazy initialisation da chave de página

```
return Redirect::route('brand-profile', PageController::key());
```

Código 47 – Laravel Controller – Endereço dinâmico com identificação da página

Trata ainda de validar e submeter pedidos de alteração aos dados de perfil bem como de contatos.

5.3.3.6. Controladores de Identidade Visual

Os controladores de Identidade Visual ouvem e resolvem os pedidos de edição dos contextos de mais alto nível dentro da Identidade Visual da Marca. São eles:

- Logo: versões e medidas;
- Colourway: paletas e cores;
- Typography: fontes e suas aplicações;
- Imagery: galerias, albuns e imagens.

Capítulo 6

Conclusões

Este capítulo serve para apresentar as conclusões do desenvolvimento do projecto.

6.1. Aspectos positivos

Quanto ao estagiário: este projecto permitiu que o estagiário desenvolvesse:

- Experiência nas tecnologias, no desenvolvimento de projectos informáticos, bem como do conhecimento do *design* gráfico e gestão da marca, suas identidades e conceitos;
- Maior coordenação; mais cuidado com a gestão de tempos, períodos e metas; noção de produção intensiva de documentação e maior percepção das condições de desenvolvimento em ambiente real.

Quanto à plataforma:

- Está estruturada: os contextos, navegação, acessos e elementos da plataforma conjugam-se fluidamente e com capacidade para acoplação dos módulos futuros;
- A gestão de utilizadores e respectivas funções cumpre com o planeado;
- Os conteúdos são partilháveis com os utilizadores pretendidos que têm uma navegação simples e intuitiva sobre os mesmos.

Quanto à empresa acolhedora:

- Processos arquitectados: este projecto precisou estruturar todos os processos no tratamento de marcas e de agências de *design* – os mesmos processos podem ser formalizados a um nível de detalhe maior do que existia antes, de acordo com a documentação recolhida a respeito;
- A automação dos processos viabiliza agora respostas rápidas, organizadas e com menos necessidade da presença dos vários intervenientes, bem como minimiza a comunicação entre eles.

6.2. Aspectos a relatar

Conforme apresentado no Anexo 6 – Conclusões pessoais, o desvio à cronologia planeada, nomeadamente pelo período imprevisto para a arte para, ou no domínio das tecnologias desconhecidas, impediu a existência uma fase de testes adequada dentro do previsto. Esta terá de acontecer após o término do estágio que enquadrou este projecto.

Além do ponto anterior, falta ainda enunciar as funcionalidades e requisitos pensados *a priori* mas insatisfeitos na altura da entrega deste documento:

- Gestão de “Brand Applications”;
- Gestão de Agências;
- Acessos mais consistentes;
- Participação do utilizador ‘BAMP Admin’.

6.3. Trabalho Futuro

Os pontos identificados na secção anterior (7.2 – Aspectos a relatar) constituem o principal impedimento para avançar com a plataforma no mercado. Urge rematá-los de forma a manter este produto na janela de oportunidade possível.

Fora dos requisitos deste estágio, ficaram requisitos para outros módulos que aguardam vez para desenvolvimento. Nomeadamente:

- Brand Framework
- Brand Tone of Voice.

Não se termina esta secção sem enunciar alguns pontos que, embora não tenham sido previstos a priori, tenham sido imaginados no decorrer do projecto e que possam constituir mais-valia:

- Internacionalização: embora pensada para um mercado global, a sua tradução para outras línguas (sendo para a portuguesa a primeira a ser efectuada) seria uma mais valia assim como potenciaria o alargamento da sua utilização;
- Pagamento online: facilitar aos clientes a aquisição de serviços (e.g. inscrição de agências e aumento do numero de marcas) é sem dúvida outra mais-valia para todos os intervenientes.

Com base na análise efectuada no Estado da Arte, pode afirmar-se que existe uma carência de soluções do genero, pelo que a plataforma desenvolvida poderá vir ter um futuro de sucesso caso se opte pela sua conclusão. É minha convicção que poderia sair deste projecto a criação de uma *Startup* com possibilidade de vir a singrar, focalizada no aperfeiçoamento e desenvolvimento da solução desenvolvida.

Referências

(20 de 06 de 2014). Obtido de Laravel - The PHP Framework For Web Artisans.:
<http://laravel.com/docs/introduction#laravel-philosophy>

Freemium – *Wikipédia, a enciclopédia livre*. (28 de 06 de 2014). Obtido de Wikipédia, a enciclopédia livre:
<http://pt.wikipedia.org/w/index.php?title=Freemium&oldid=38389262>

Netcraft. (06 de 06 de 2014). Obtido de Netcraft:
<http://news.netcraft.com/archives/2014/06/06/june-2014-web-server-survey.html>

VA. (2014). *Comparison of web application frameworks* - *Wikipedia, the free encyclopedia*. Obtido em 23 de 06 de 2014, de
http://en.wikipedia.org/w/index.php?title=Comparison_of_web_application_frameworks&oldid=613775888

Anexos






Anexo 1.


Lingo: Definições e Conceitos

Brand Identity: is the sum of all forms of corporate outward show. In other words: in addition to the visual image a company provides, it also involves all the verbal expressions, behaviours, and structures that a company uses when interacting both within its own corporate structure and with the public.


Visual Identity: Visible elements of a brand, such as logo, colour and typography, which encapsulate and convey the symbolic meanings that cannot be imparted through words alone.





Logo Versions:

<p>Logo: Typically consists of text, graphic image, or a combination of both. It is an important element of corporate design in that it formulates the visual identity of the entity or institution it represents.</p>	 <p>City of Sabugal, in Portugal</p>
<p>Logo Format: It's common for brand to have alternative logos applicable in different situations and needs. Those Formats are: Standard, Small and Large.</p>	<p>Wit Software logo includes the word "software" as standard logo, a smaller version has design to be used in some special occasions.</p> 
<p>Logo Colour Variations: Logos can have several variations in its colours with the purpose of easy adaptation.</p> <p>Main colour variations: Positive, Negative, Monochromatic</p>	<p>FCTUC main version or positive:</p>  <p>FCTUC negative version:</p>  <p>FCTUC monochromatic version:</p> 
<p>Logo Measurements: Measurements are needed to assure the correct use, and visibility of a logo. The most common well defined measurements are: protective</p>	<p>Wit Software logo Protective space:</p>



<p>space, minimum size and angle.</p>	 <p>Portugal Telecom logo minimum size: 7 mm </p> <p>Best Buy logo angle: </p>
<p>Type of logo: the main, or preferred logo, is horizontal or vertical but, in many occasions, brands have other elements in their logos to be used in specific situations or as alternative logos.</p> <p>Some of this types are: horizontal, vertical, strapline/tagline, symbol and wordmark</p>	<p>Firefox horizontal and vertical logo: </p> <p>Valspar logo with strapline: </p> <p>IBM logo with register symbol included: </p> <p>Canada wordmark logo: </p>

Logo Exceptions:





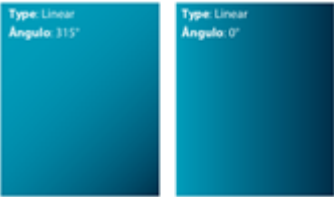
<p>Favicons: Used only to identify a webpage; favicons are very special logos because of their extremely limited size. They tend to bend all the rules associated with the design of the main logos.</p>	<p>British Council favicon: </p> <p>In this example we can see the only time British Council Brand is disassociated of the wordmark.</p>
<p>Animations: Some brands have very strict policies concerning animation of their identities</p>	

<p>Icons: Special logos to be used to identify a brand with an application.</p>	<p>Youtube Icon, Android application:</p> 
<p>Translations and Legal Exceptions: Global brands have their own preferred logos to be used in different parts of the globe so they can be easily identified by the locals.</p> <p>Language: logo translation language.</p>	<p>Coca-Cola logo translated to Hebrew:</p>  <p>Approved British Council Logo:</p> 
<p>Commemorative: logos design to be used in short periods of time. Common example: anniversaries</p>	<p>Ordem dos Engenheiros 75 anniversary logo:</p> 

Identity Elements

<p>Flag and coat of arms: To illustrate this element we have the examples of Portugal towns; they all have a flags.</p>	<p>Vila of Sabugal Flag and Coat of Arms:</p> 
<p>Graphic: graphic elements that can be used alone and clearly identify a brand.</p>	<p>Casa da Lusofonia graphic elements:</p> 

Colourways: Any of a range of combinations of colours in which a style or design is available.

<p>Palettes: A palette of colours can contain one or more colours well identified by colour systems. Brands usually have a primary and a secondary palette.</p>	<p>Colour identified with the main colour systems:</p>  <p>CMYK: C 59 M 0 Y 100 K 0 <i>Referência de cor para impressão offset em quadricromia</i> RGB: R 100 G 183 B 55 <i>Referência de cor para meios digitais</i> HEXADECIMAL: #64b737 <i>Referência de cor para internet</i> PAL: 6018 <i>Referência de cor para tintas e lacados</i> 3M (serie 30 Scotchcal): 30 - 730 <i>apple green</i> <i>Referência de cor para vinyl</i></p>
<p>Applications: Applying colours to different scenarios can be a challenge. Brands usually have those scenarios covered.</p>	<p>Tone-on-tone:</p>  <p>Overlaid:</p>  <p>Colour with Photography:</p>  <p>Gradient:</p> 

Main Colour Systems

<p>PANTONE (PMS)</p>	<p>The Pantone Colour Matching System is largely a standardized colour reproduction system” used primarily for printing.</p>
<p>RGB</p>	<p>“The RGB colour model is an additive colour model in which red, green, and blue light are added together in various ways to reproduce a broad array of colours. The main purpose of the RGB colour model is for the sensing, representation, and display of images in electronic systems.” [Wikipedia]</p>

HEXADECIMAL	Also called web colours, “a colour is specified according to the intensity of its red, green and blue components, each represented by eight bits.” [Wikipedia]
CMYK	“Is a subtractive colour model, used in colour printing, and is also used to describe the printing process itself.” [Wikipedia]
RAL	“Is a colour matching system used in Europe. In colloquial speech RAL refers to the RAL Classic system, mainly used for varnish and powder coating but nowadays there are reference panels for plastics as well.” [Wikipedia]
3M	Colour Reference used for vinyl prints.

Typography: The general term “typography” refers to the functions of typeface design and the arrangement of type and other elements on a page. A typography consist in sets of typefaces, with one or more elements, classified according to its name, family and general family.

- **Font:** In traditional typography, a font is a particular size, weight and style of a typeface. The term font it’s exclusively used in context of software application.

Imagery: Image galleries like Photography, Pattern or Texture.

Brand Framework: brand attributes such as Values, Positioning and Personality.

Brand Tone-of-Voice: how a brand "talks" both in written of spoken formats

Brand Applications: these applications are the touchstone for creating high-impact communications that present a unified brand image.

Brand Applications examples:

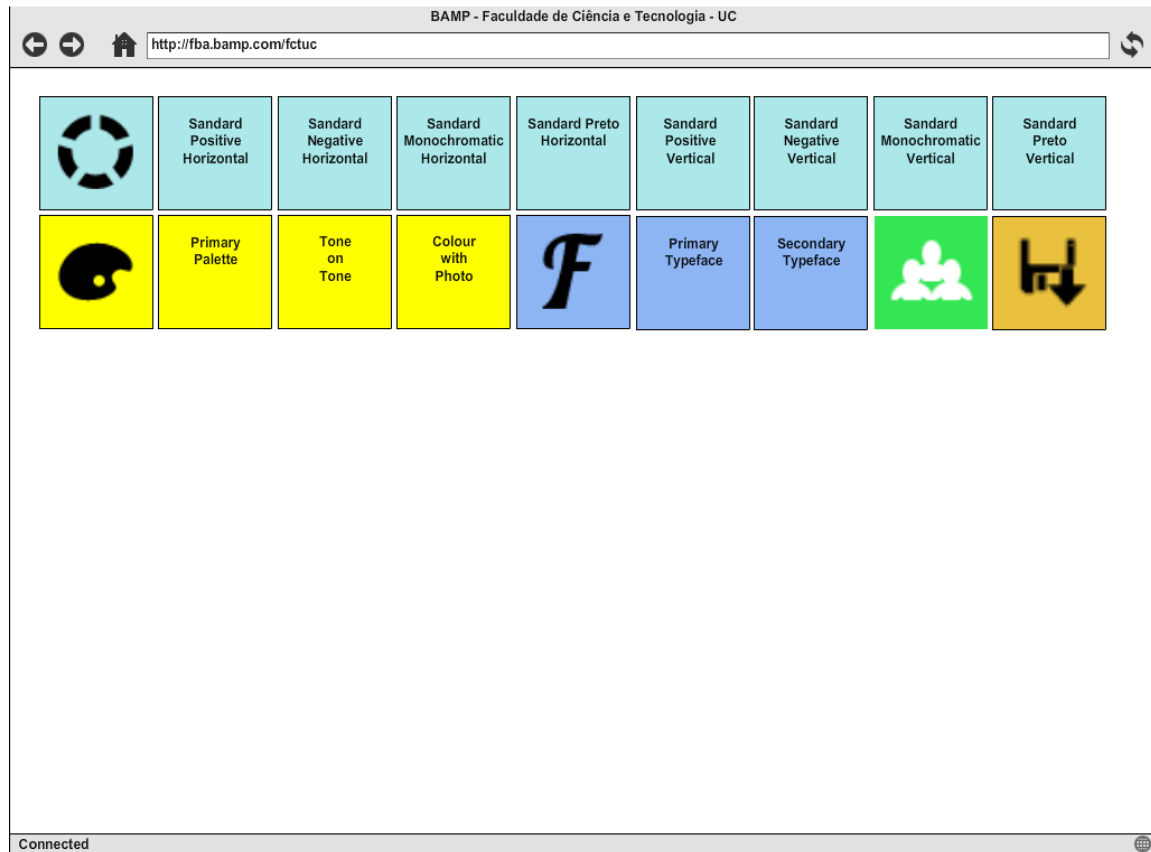
- Brochures
- Corporate Folder
- Documents templates: PowerPoint, Word, etc
- Branded Merchandise: Mugs, Hats, Clothes, etc

Anexo 2. Wireframes - Mockups

Páginas Iniciais Pré e Pós Autenticação

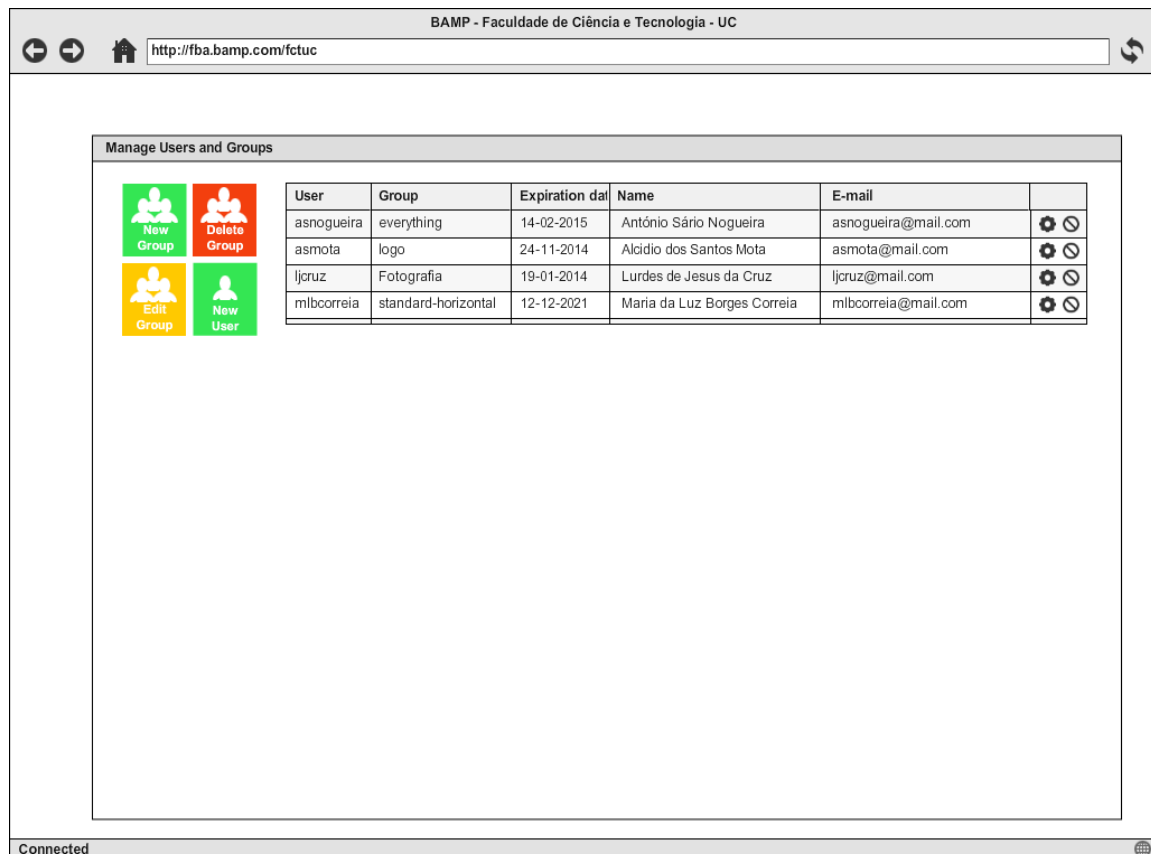
The wireframe shows a browser window with the address bar containing 'http://fba.bamp.com'. The page layout is divided into two main vertical sections. The left section contains the 'FBA BAMP' logo at the top, followed by an 'Email address' input field, a 'Password' input field, a 'Remember me' checkbox, and a blue 'Sign in' button. The right section is a large, empty white space. At the bottom left of the browser window, the text 'Connected' is visible next to a globe icon.

Wireframe 1- Página Publica com menu de autenticação

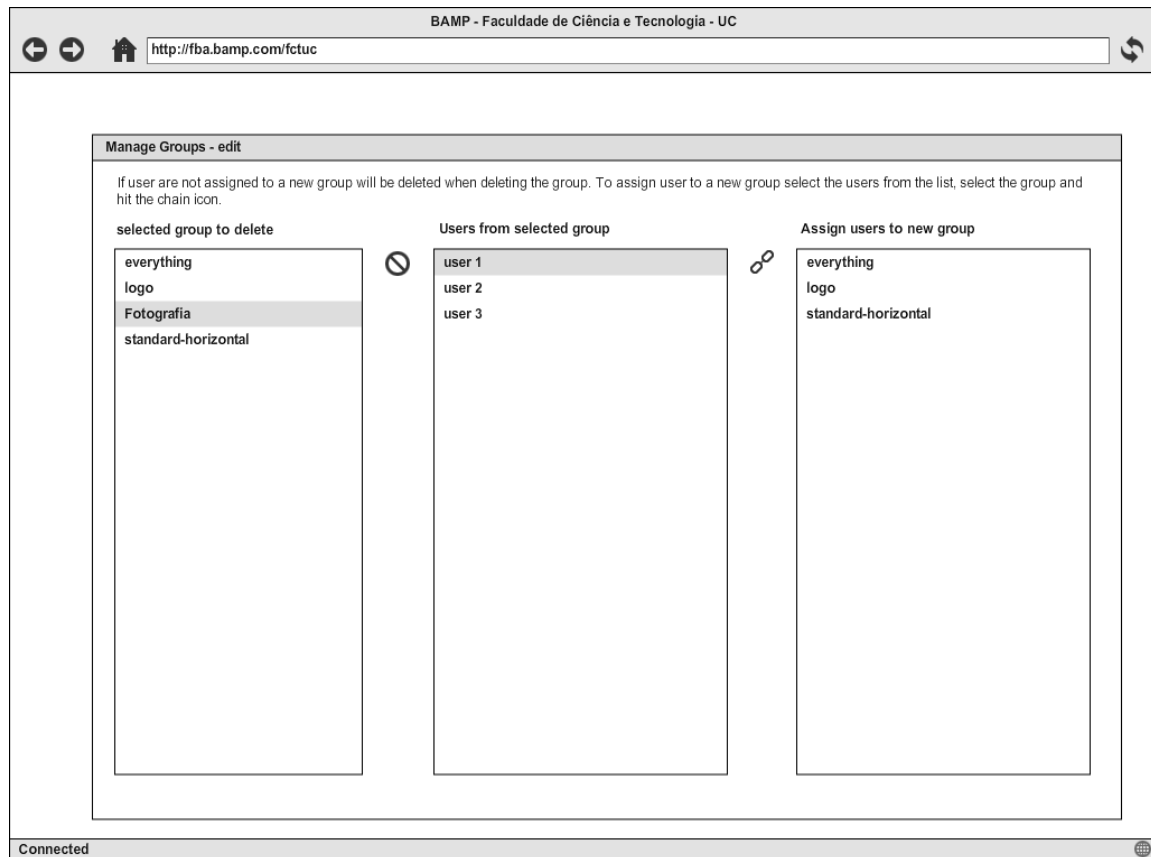


Wireframe 2 – Página Principal da Marca após autenticação

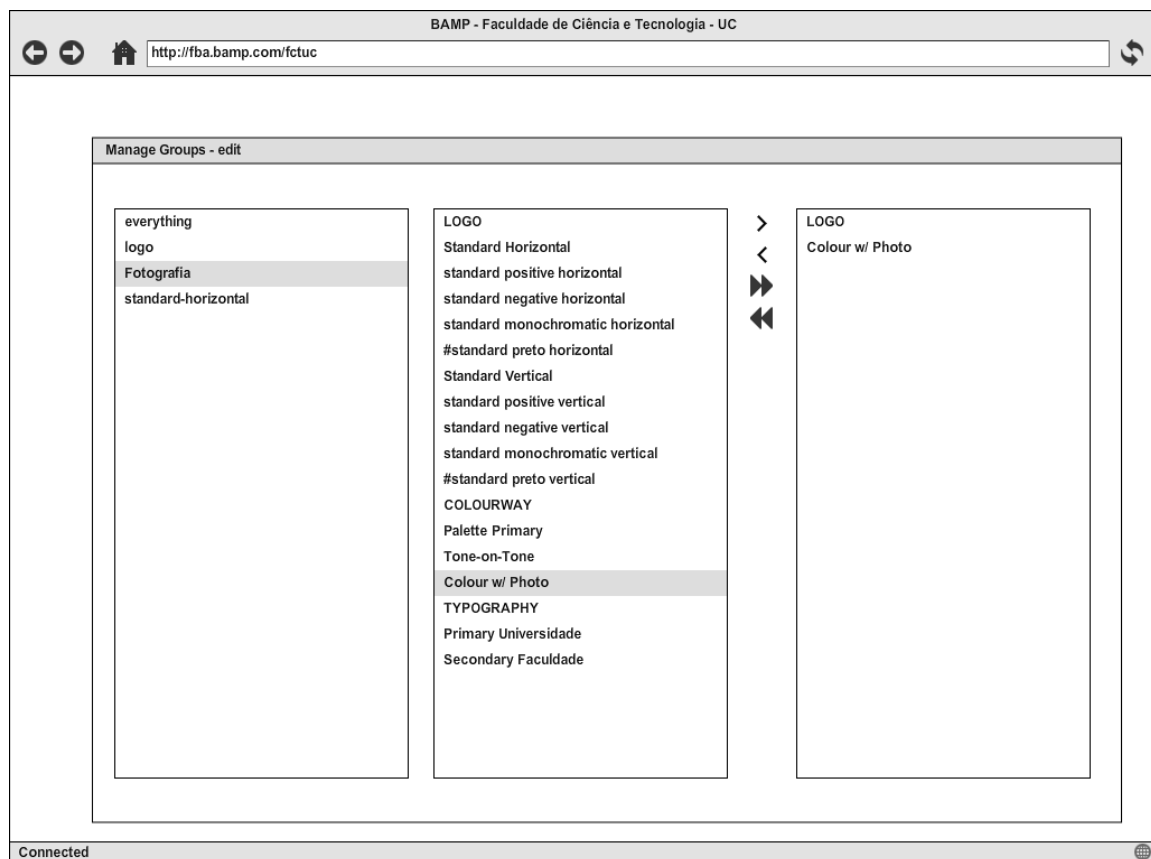
Gestão de grupos de conteúdos



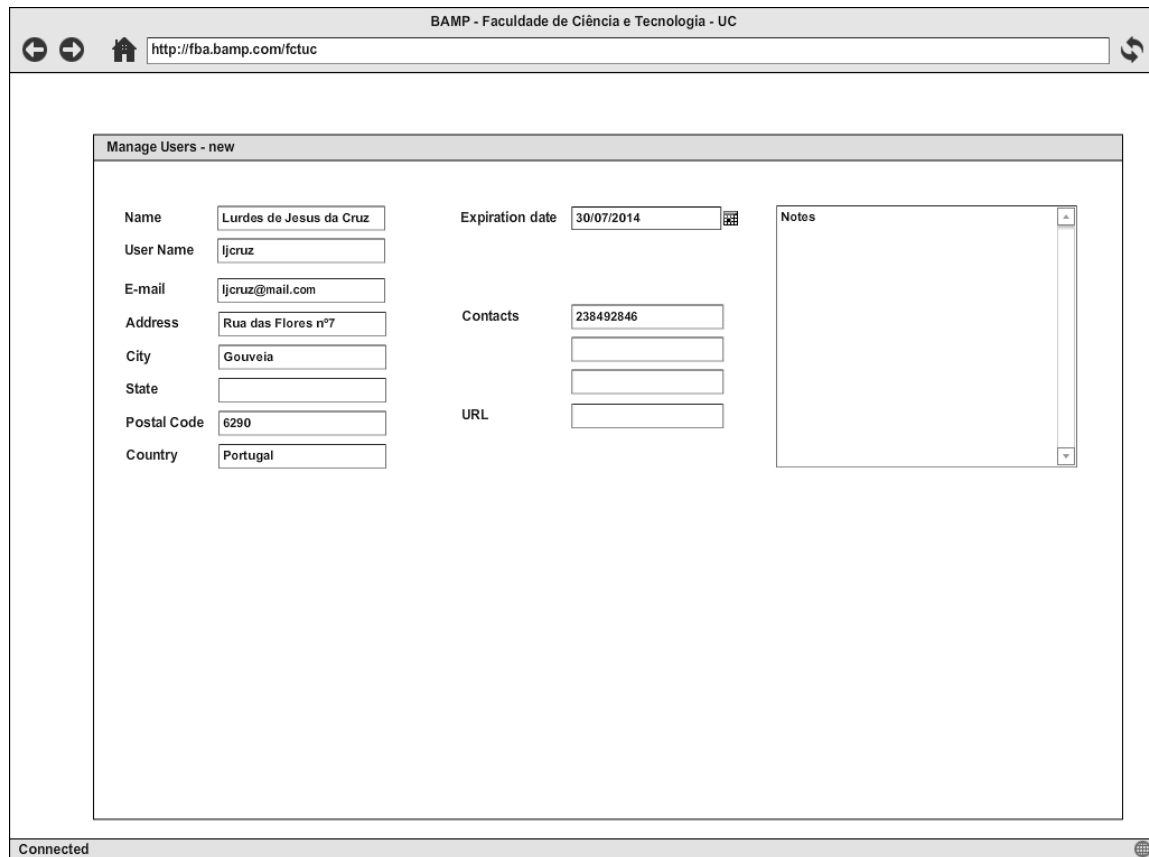
Wireframe 3 – Página Principal de gestão de utilizadores e grupos de partilha de conteúdos



Wireframe 4 – Página de gestão de grupos de utilizadores

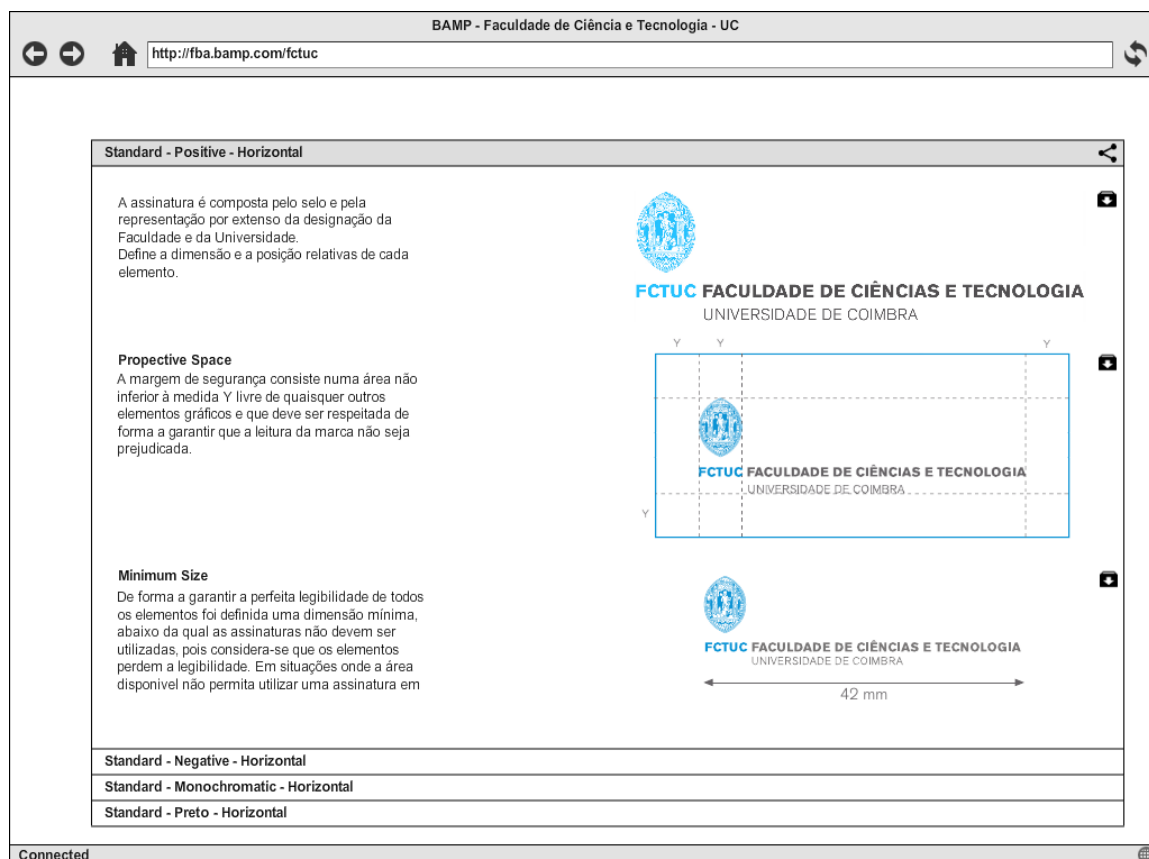


Wireframe 5 – Página de gestão de grupos de conteúdos



Wireframe 6 – Página de criação de novo utilizador

Visualização e consulta




Wireframe 7 – Página de consulta de uma versão do logo

BAMP - Faculdade de Ciência e Tecnologia - UC

http://fba.bamp.com/fctuc

Standard - Positivo - Vertical

A assinatura é composta pelo selo e pela representação por extenso da designação da Faculdade e da Universidade. Define a dimensão e a posição relativas de cada elemento.



FCTUC FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Propective Space
A margem de segurança consiste numa área não inferior à medida Y livre de quaisquer outros elementos gráficos e que deve ser respeitada de forma a garantir que a leitura da marca não seja prejudicada.



Minimum Size
De forma a garantir a perfeita legibilidade de todos os elementos foi definida uma dimensão mínima, abaixo da qual as assinaturas não devem ser utilizadas, pois considera-se que os elementos perdem a legibilidade. Em situações onde a área disponível não permita utilizar uma assinatura em



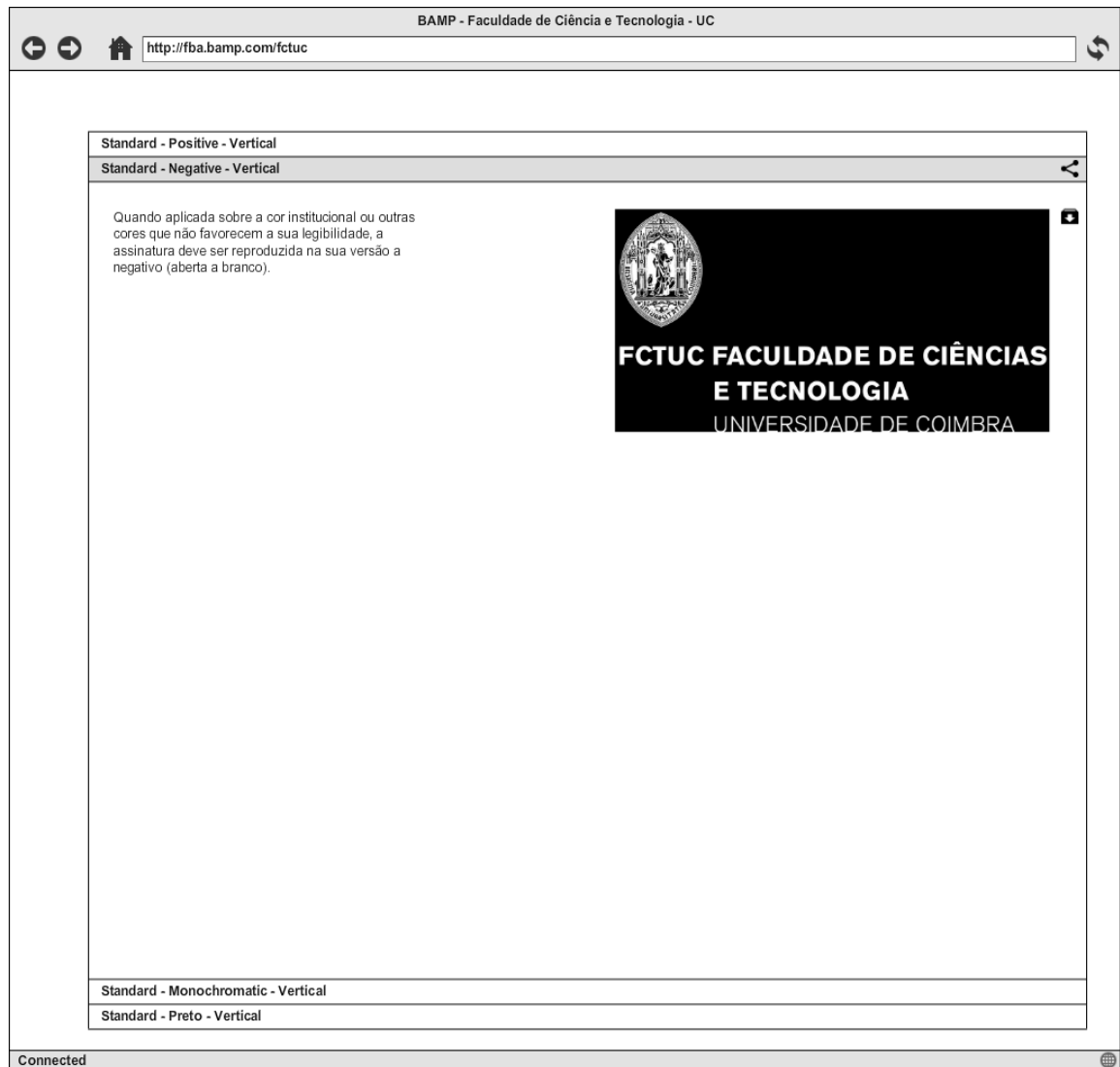
Standard - Negative - Vertical

Standard - Monochromatic - Vertical

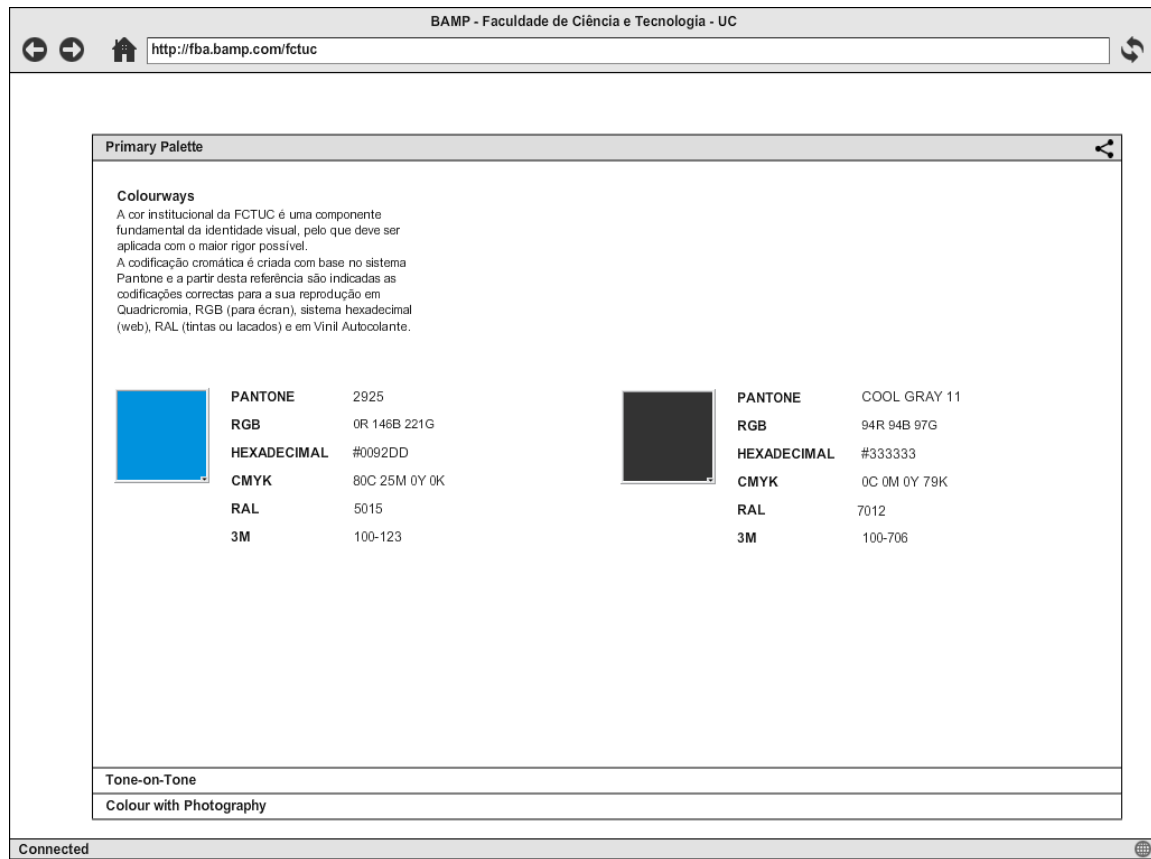
Standard - Preto - Vertical

Connected

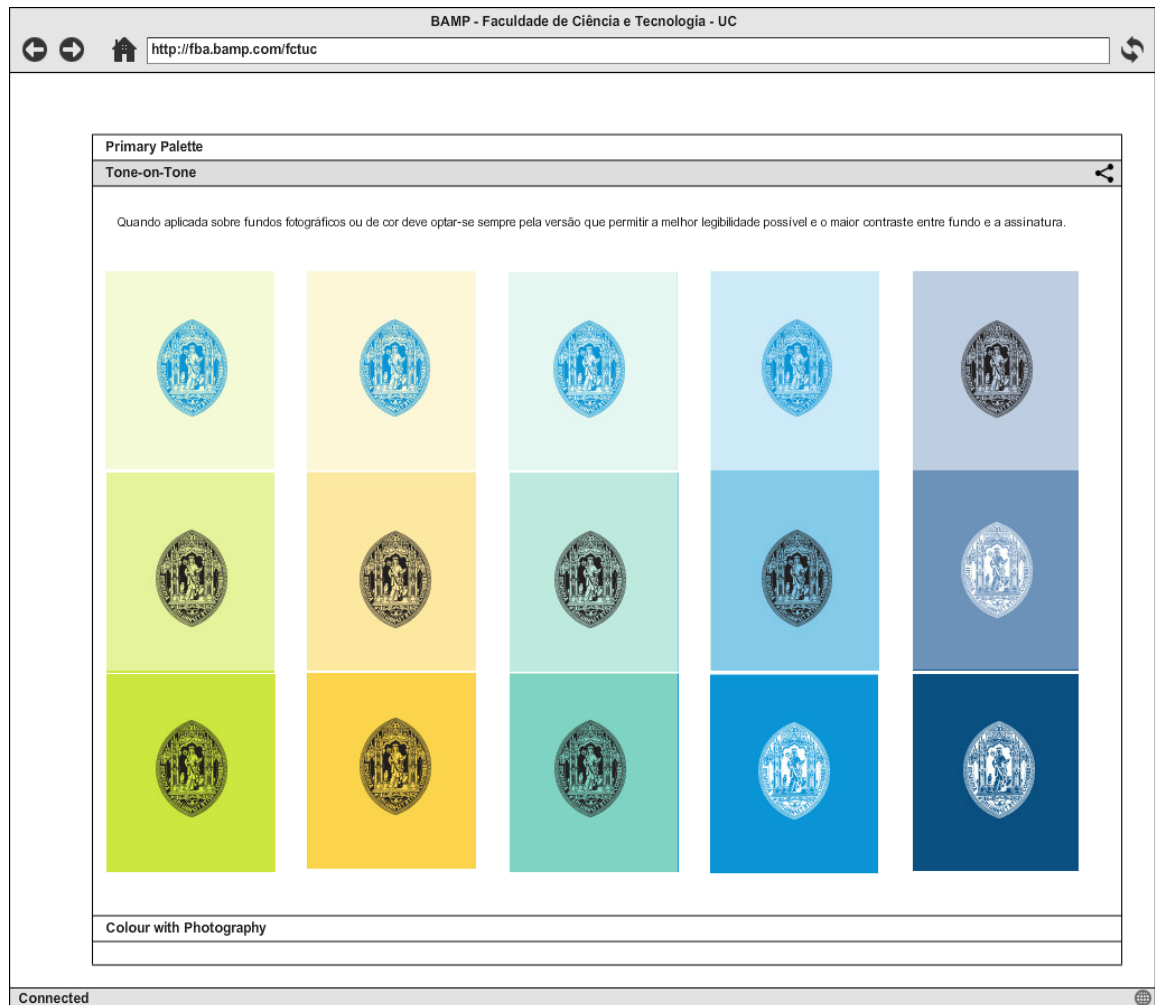
Wireframe 8 - Página de consulta de uma segunda versão do logo



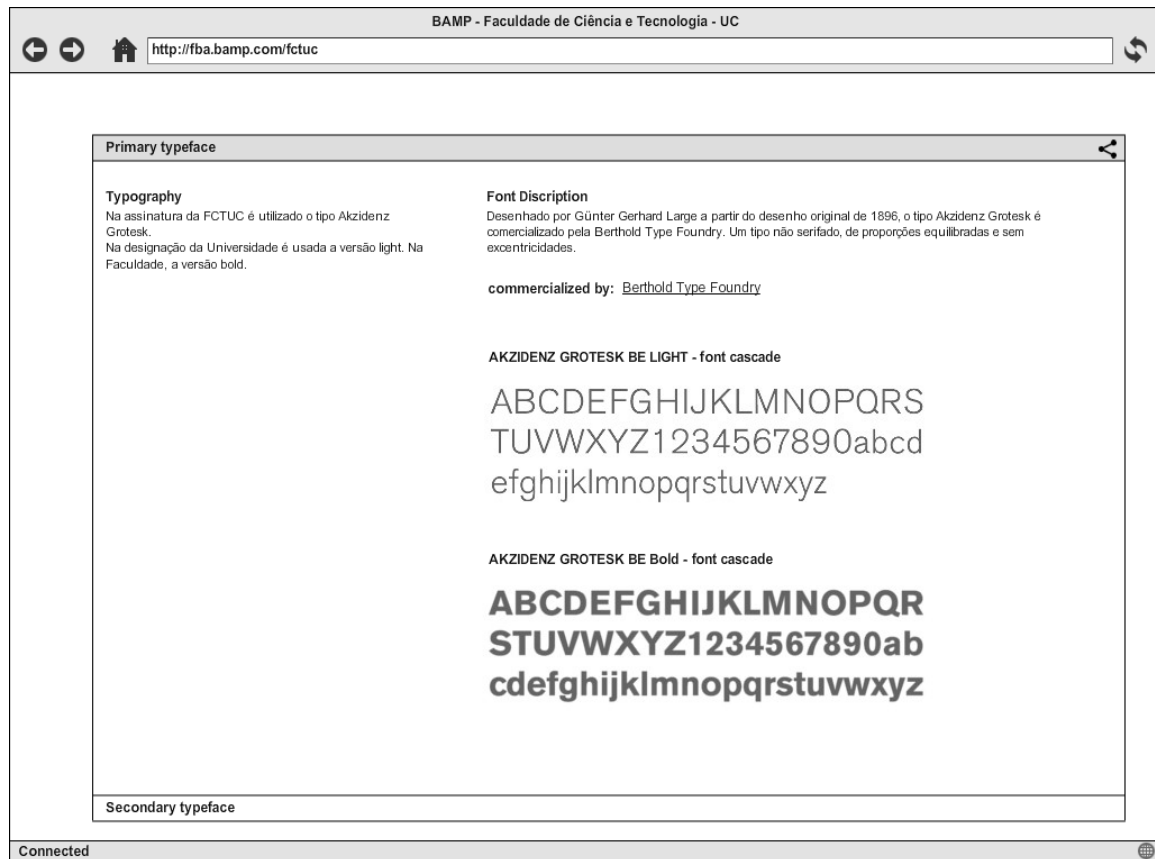
Wireframe 9 - Página de consulta de uma versão do logo com menos informação



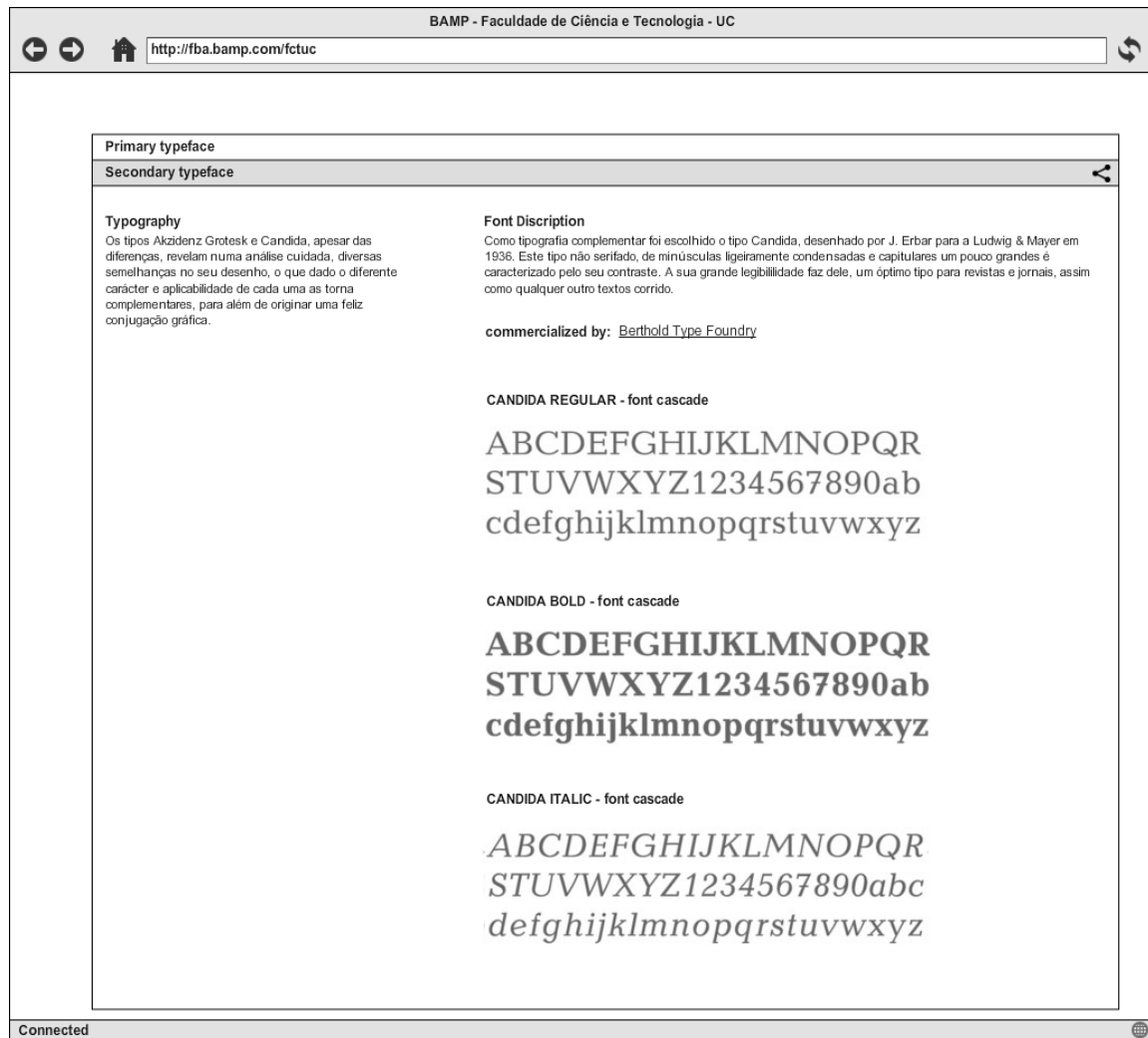
Wireframe 10 - Página de consulta de uma paleta de cores



Wireframe 11 - Página de consulta de uma das possíveis aplicações de cores

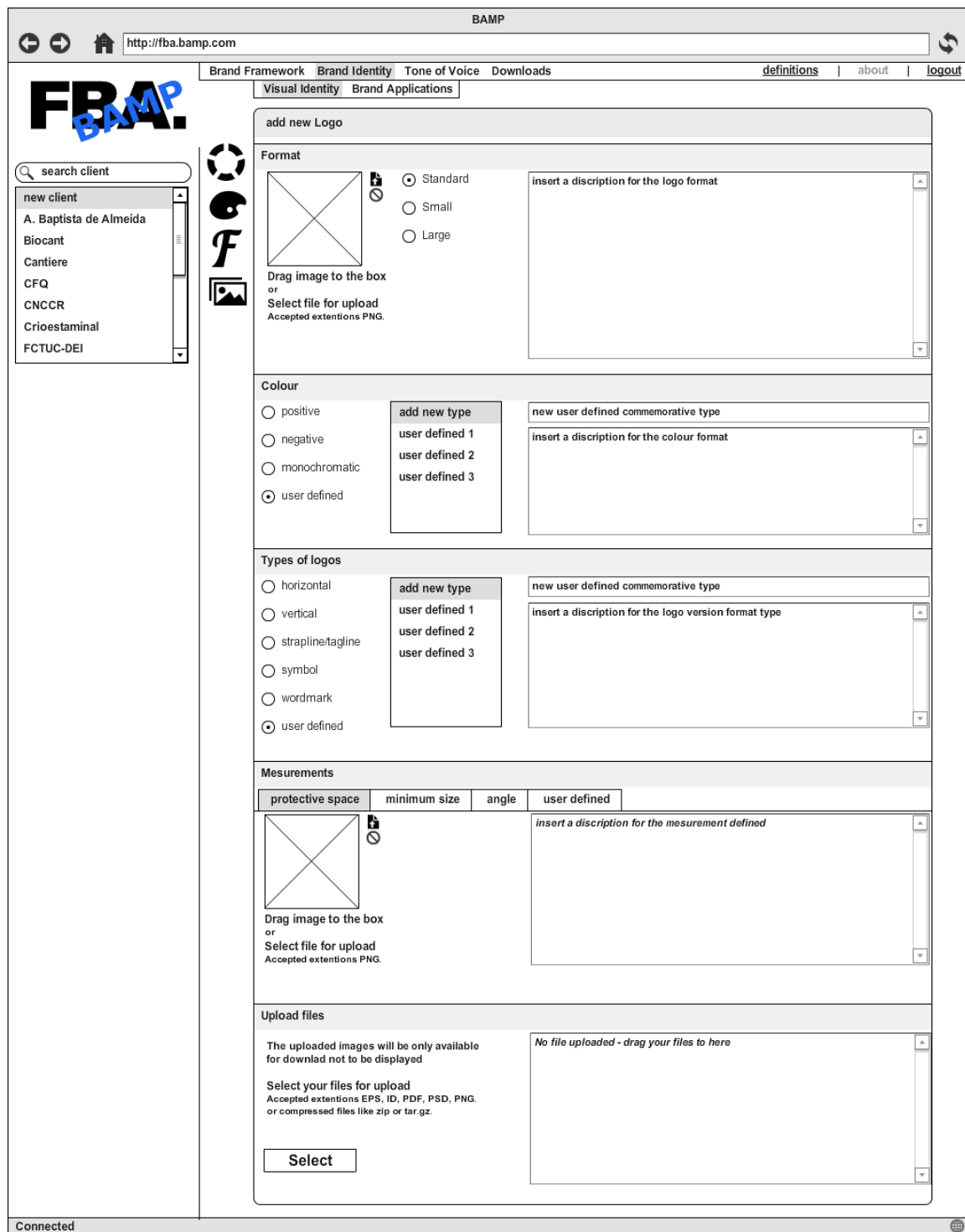


Wireframe 12 - Página de consulta da tipografia principal

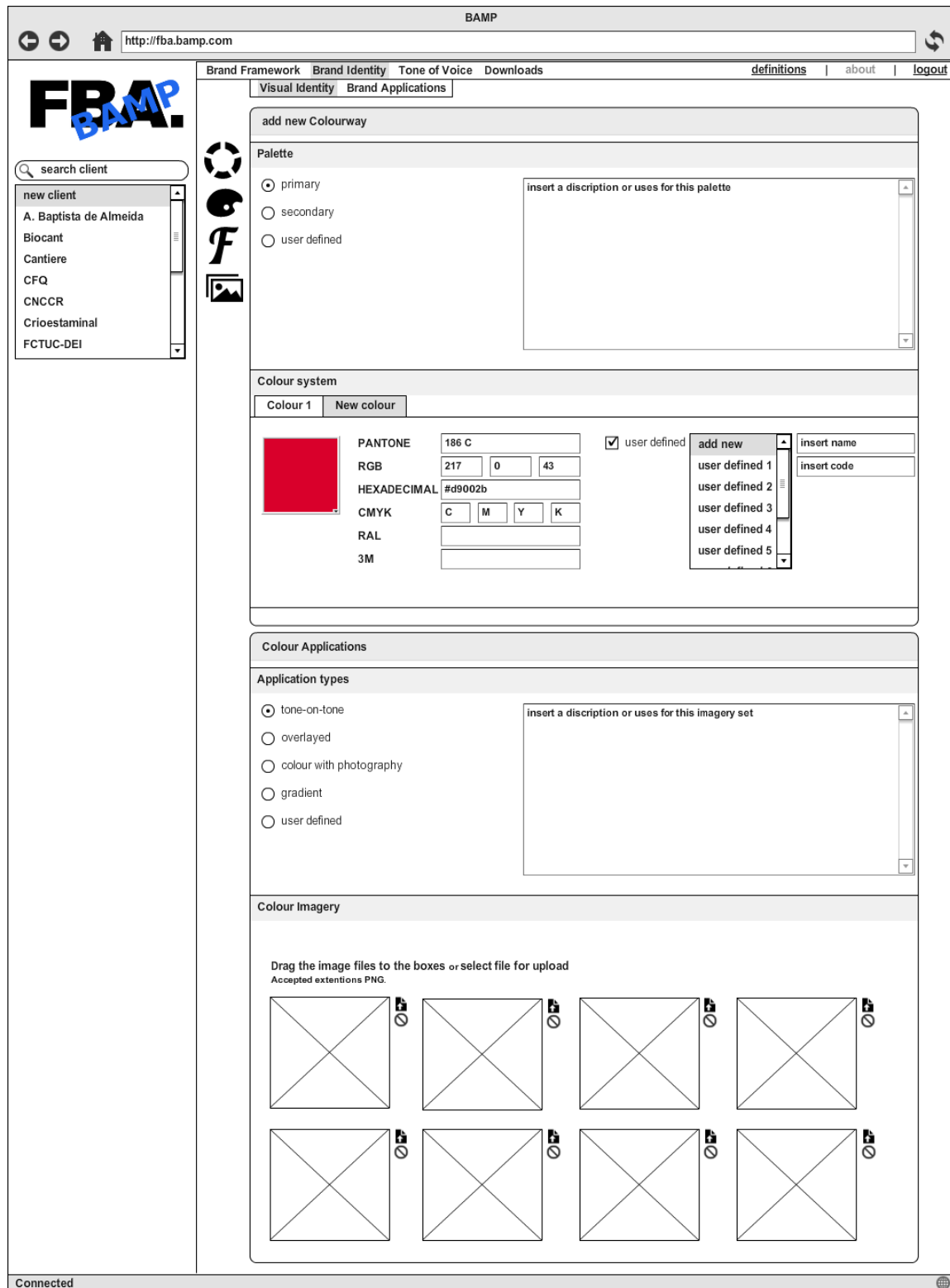


Wireframe 13 - Página de consulta da tipografia secundária

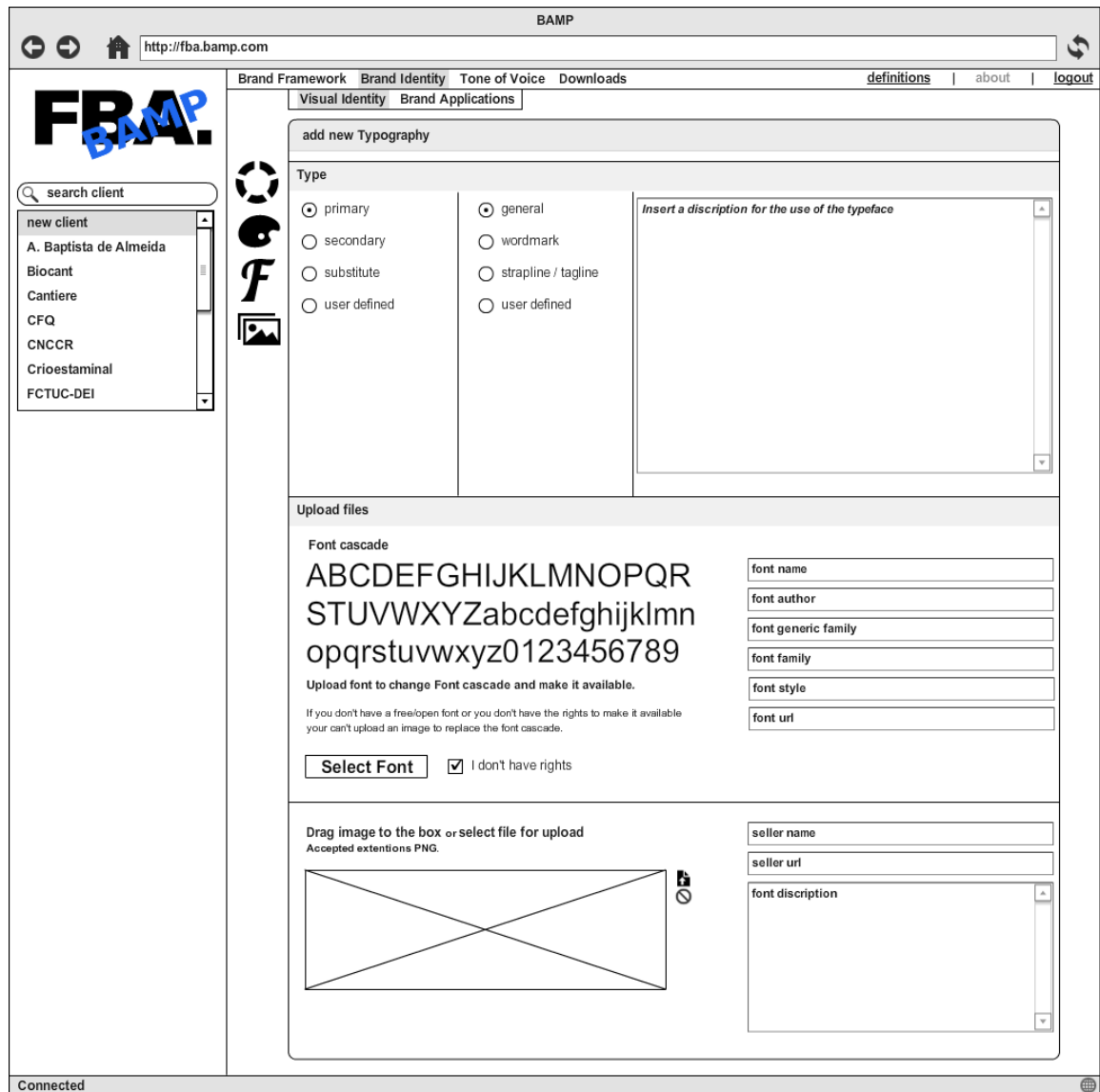
Edição



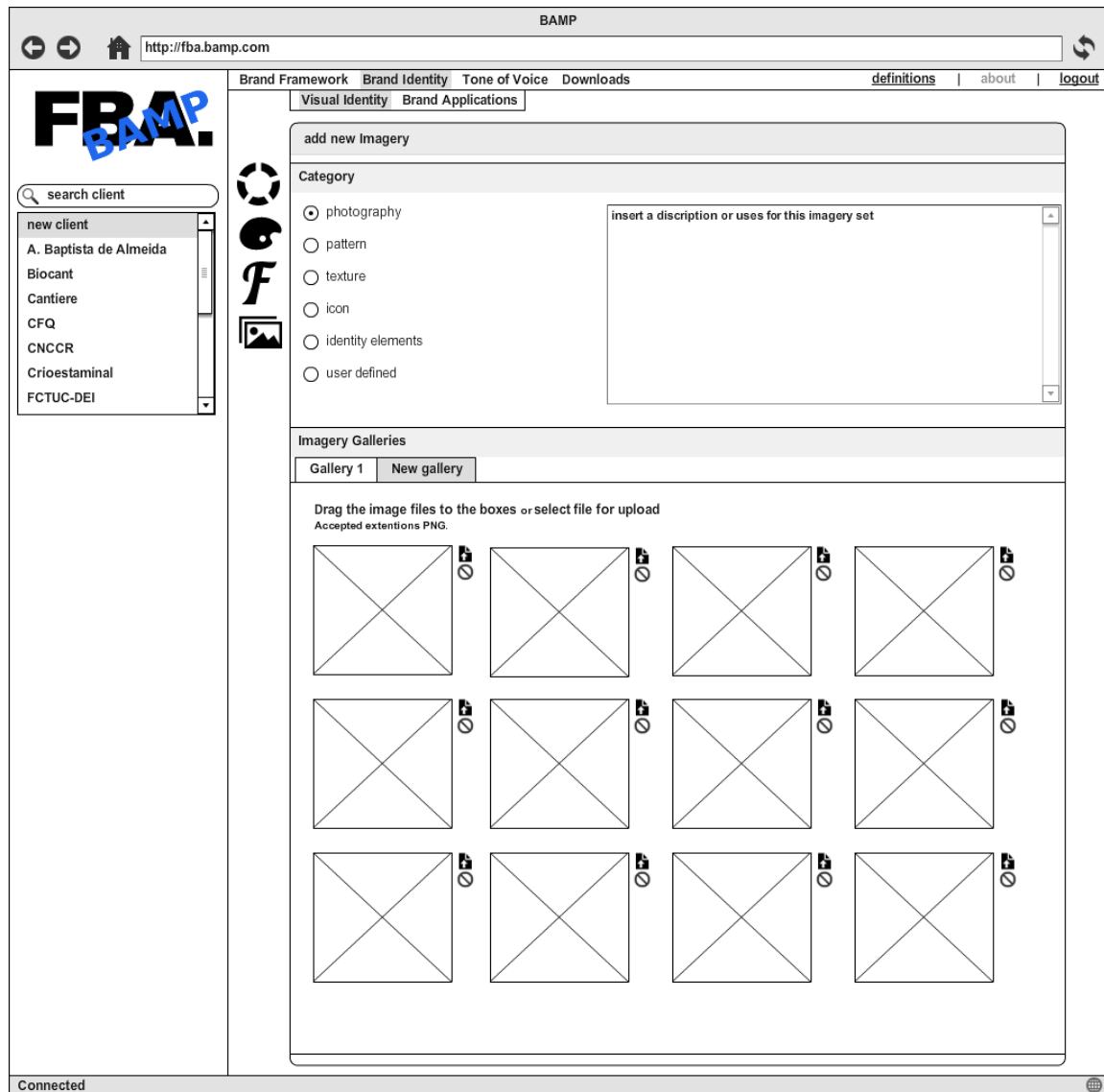
Wireframe 14 - Página de edição para versões de logós



Wireframe 15 – Página de edição de paletas de cores



Wireframe 16 – Página de edição para tipografias



Wireframe 17 – Página de edição para galerias de imagens

Anexo 3. Código SQL para geração BD

```

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

CREATE SCHEMA IF NOT EXISTS `bamp_db` DEFAULT CHARACTER SET utf8 ;
USE `bamp_db` ;

-----
-- Table `bamp_db`.`pages`
-----
DROP TABLE IF EXISTS `bamp_db`.`pages` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`pages` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `key` VARCHAR(100) NOT NULL,
  `final` BIT NOT NULL DEFAULT b'0',
  `create_at` TIMESTAMP NULL,
  `updated_at` TIMESTAMP NULL,
  `deleted_at` TIMESTAMP NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

-----
-- Table `bamp_db`.`contacts`
-----
DROP TABLE IF EXISTS `bamp_db`.`contacts` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`contacts` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `address` VARCHAR(255) NULL,
  `city` VARCHAR(255) NULL,
  `province` VARCHAR(255) NULL,
  `postcode` VARCHAR(45) NULL,
  `country` VARCHAR(100) NULL,
  `phone` VARCHAR(25) NULL,
  `mobile` VARCHAR(25) NULL,
  `photo` VARCHAR(255) NULL,
  `email` VARCHAR(255) NULL,
  `fax` VARCHAR(25) NULL,
  `url` VARCHAR(255) NULL,
  `facebook` VARCHAR(255) NULL,
  `google_plus` VARCHAR(255) NULL,
  `twitter` VARCHAR(255) NULL,
  `linkedin` VARCHAR(255) NULL,
  `created_at` TIMESTAMP NULL,
  `updated_at` TIMESTAMP NULL,
  `deleted_at` TIMESTAMP NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

-----
-- Table `bamp_db`.`entities`
-----
DROP TABLE IF EXISTS `bamp_db`.`entities` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`entities` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `main_page_id` INT NOT NULL,
  `main_contact_id` INT NOT NULL,

```

```

`created_at` TIMESTAMP NULL,
`updated_at` TIMESTAMP NULL,
`deleted_at` TIMESTAMP NULL,
`entityable_type` VARCHAR(64) NULL,
PRIMARY KEY (`id`),
INDEX `fk_entities_pages1_idx` (`main_page_id` ASC),
INDEX `fk_entities_contacts1_idx` (`main_contact_id` ASC),
UNIQUE INDEX `main_contact_id_UNIQUE` (`main_contact_id` ASC),
UNIQUE INDEX `main_page_id_UNIQUE` (`main_page_id` ASC),
CONSTRAINT `fk_entities_pages1`
  FOREIGN KEY (`main_page_id`)
  REFERENCES `bamp_db`.`pages` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
CONSTRAINT `fk_entities_contacts1`
  FOREIGN KEY (`main_contact_id`)
  REFERENCES `bamp_db`.`contacts` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----
-- Table `bamp_db`.`users`
-----

DROP TABLE IF EXISTS `bamp_db`.`users` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`users` (
  `id` INT(11) NOT NULL,
  `first_name` VARCHAR(255) NULL,
  `last_name` VARCHAR(255) NULL,
  `email` VARCHAR(255) NOT NULL,
  `username` VARCHAR(50) NOT NULL,
  `password` VARCHAR(60) NOT NULL,
  `password_temp` VARCHAR(60) NULL,
  `code` VARCHAR(60) NULL,
  `active` INT(11) NULL DEFAULT 0,
  `alternative_email` VARCHAR(255) NULL DEFAULT NULL,
  `remember_token` VARCHAR(100) NULL,
  `max_brands` INT(11) NOT NULL DEFAULT 1,
  `is_bamp_admin` BIT NOT NULL DEFAULT b'0',
  PRIMARY KEY (`id`),
  UNIQUE INDEX `username_UNIQUE` (`username` ASC),
  UNIQUE INDEX `email_UNIQUE` (`email` ASC),
  CONSTRAINT `fk_users_1`
    FOREIGN KEY (`id`)
    REFERENCES `bamp_db`.`entities` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`roles`
-----

DROP TABLE IF EXISTS `bamp_db`.`roles` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`roles` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `title` VARCHAR(45) NOT NULL,
  `description` TEXT NULL DEFAULT NULL,
  `created_at` TIMESTAMP NULL,
  `updated_at` TIMESTAMP NULL,
  `deleted_at` TIMESTAMP NULL,
  PRIMARY KEY (`id`))

```

```

ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`user_roles`
-----
DROP TABLE IF EXISTS `bamp_db`.`user_roles` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`user_roles` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `user_id` INT(11) NOT NULL,
  `role_id` INT(11) NOT NULL,
  `state` ENUM('requested', 'accepted', 'refused', 'suspended') NOT NULL DEFAULT
'requested',
  `code` VARCHAR(60) NULL DEFAULT NULL,
  `created_at` TIMESTAMP NULL,
  `updated_at` TIMESTAMP NULL,
  `deleted_at` TIMESTAMP NULL,
  `user_roleable_type` VARCHAR(64) NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_user_role_user1_idx` (`user_id` ASC),
  INDEX `fk_user_role_role1_idx` (`role_id` ASC),
  CONSTRAINT `fk_user_role_user1`
    FOREIGN KEY (`user_id`)
      REFERENCES `bamp_db`.`users` (`id`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_user_role_role1`
    FOREIGN KEY (`role_id`)
      REFERENCES `bamp_db`.`roles` (`id`)
      ON DELETE CASCADE
      ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`agencies`
-----
DROP TABLE IF EXISTS `bamp_db`.`agencies` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`agencies` (
  `id` INT(11) NOT NULL,
  `basa_id` INT(11) NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_agency_user_role1_idx` (`basa_id` ASC),
  CONSTRAINT `fk_agency_user_role1`
    FOREIGN KEY (`basa_id`)
      REFERENCES `bamp_db`.`user_roles` (`id`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_agencies_1`
    FOREIGN KEY (`id`)
      REFERENCES `bamp_db`.`entities` (`id`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`contents`
-----
DROP TABLE IF EXISTS `bamp_db`.`contents` ;

```

```

CREATE TABLE IF NOT EXISTS `bamp_db`.`contents` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `creator_id` INT(11) NOT NULL,
  `brand_id` INT(11) NOT NULL,
  `created_at` TIMESTAMP NULL,
  `updated_at` TIMESTAMP NULL,
  `deleted_at` TIMESTAMP NULL,
  `contentable_type` VARCHAR(64) NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_content_brand1_idx` (`brand_id` ASC),
  INDEX `fk_content_user1_idx` (`creator_id` ASC),
  CONSTRAINT `fk_content_brand1`
    FOREIGN KEY (`brand_id`)
      REFERENCES `bamp_db`.`brands` (`id`)
      ON DELETE CASCADE
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_content_user1`
    FOREIGN KEY (`creator_id`)
      REFERENCES `bamp_db`.`users` (`id`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`brand_identities`
-----
DROP TABLE IF EXISTS `bamp_db`.`brand_identities` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`brand_identities` (
  `id` INT(11) NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_brand_identity_content1`
    FOREIGN KEY (`id`)
      REFERENCES `bamp_db`.`contents` (`id`)
      ON DELETE CASCADE
      ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`brand_frameworks`
-----
DROP TABLE IF EXISTS `bamp_db`.`brand_frameworks` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`brand_frameworks` (
  `id` INT(11) NOT NULL,
  `title` VARCHAR(45) NULL DEFAULT NULL,
  `description` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_brand_framework_content1_idx` (`id` ASC),
  CONSTRAINT `fk_brand_framework_content1`
    FOREIGN KEY (`id`)
      REFERENCES `bamp_db`.`contents` (`id`)
      ON DELETE CASCADE
      ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`brand_tones_of_voice`
-----

```

```

DROP TABLE IF EXISTS `bamp_db`.`brand_tones_of_voice` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`brand_tones_of_voice` (
  `id` INT(11) NOT NULL,
  `title` VARCHAR(45) NOT NULL,
  `description` TEXT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_brand_tone_of_voice_content1_idx` (`id` ASC),
  CONSTRAINT `fk_brand_tone_of_voice_content1`
    FOREIGN KEY (`id`)
    REFERENCES `bamp_db`.`contents` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`brands`
-----
DROP TABLE IF EXISTS `bamp_db`.`brands` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`brands` (
  `id` INT(11) NOT NULL,
  `creator_id` INT(11) NOT NULL,
  `identity_id` INT(11) NOT NULL,
  `framework_id` INT(11) NULL,
  `tone_of_voice_id` INT(11) NULL,
  `trade_name` VARCHAR(100) NOT NULL,
  `legal_name` VARCHAR(255) NULL DEFAULT NULL,
  `tax_id` VARCHAR(20) NULL DEFAULT NULL,
  `brandable_type` VARCHAR(64) NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_brand_user1_idx` (`creator_id` ASC),
  INDEX `fk_brands_brand_identities1_idx` (`identity_id` ASC),
  INDEX `fk_brands_brand_frameworks1_idx` (`framework_id` ASC),
  INDEX `fk_brands_brand_tones_of_voice1_idx` (`tone_of_voice_id` ASC),
  UNIQUE INDEX `identity_id_UNIQUE` (`identity_id` ASC),
  UNIQUE INDEX `framework_id_UNIQUE` (`framework_id` ASC),
  UNIQUE INDEX `tone_of_voice_id_UNIQUE` (`tone_of_voice_id` ASC),
  CONSTRAINT `fk_brand_user1`
    FOREIGN KEY (`creator_id`)
    REFERENCES `bamp_db`.`users` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_brands_brand_identities1`
    FOREIGN KEY (`identity_id`)
    REFERENCES `bamp_db`.`brand_identities` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_brands_brand_frameworks1`
    FOREIGN KEY (`framework_id`)
    REFERENCES `bamp_db`.`brand_frameworks` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_brands_brand_tones_of_voice1`
    FOREIGN KEY (`tone_of_voice_id`)
    REFERENCES `bamp_db`.`brand_tones_of_voice` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_brands_1`
    FOREIGN KEY (`id`)
    REFERENCES `bamp_db`.`entities` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB

```

```

DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`agency_brands`
-----
DROP TABLE IF EXISTS `bamp_db`.`agency_brands` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`agency_brands` (
  `id` INT NOT NULL,
  `agency_id` INT(11) NOT NULL,
  `bou_id` INT(11) NOT NULL,
  INDEX `fk_agency_brand_brand1_idx` (`id` ASC),
  INDEX `fk_agency_brand_agency1_idx` (`agency_id` ASC),
  INDEX `fk_agency_brand_user_role1_idx` (`bou_id` ASC),
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_agency_brand_brand1`
    FOREIGN KEY (`id`)
      REFERENCES `bamp_db`.`brands` (`id`)
      ON DELETE CASCADE
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_agency_brand_agency1`
    FOREIGN KEY (`agency_id`)
      REFERENCES `bamp_db`.`agencies` (`id`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_agency_brand_user_role1`
    FOREIGN KEY (`bou_id`)
      REFERENCES `bamp_db`.`user_roles` (`id`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`brand_applications`
-----
DROP TABLE IF EXISTS `bamp_db`.`brand_applications` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`brand_applications` (
  `id` INT(11) NOT NULL,
  `brand_identity_id` INT(11) NOT NULL,
  `type` VARCHAR(100) NOT NULL,
  `description` TEXT NULL DEFAULT NULL,
  `image_path` VARCHAR(255) NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_brand_application_content1_idx` (`id` ASC),
  INDEX `fk_brand_application_brand_identity1_idx` (`brand_identity_id` ASC),
  CONSTRAINT `fk_brand_application_content1`
    FOREIGN KEY (`id`)
      REFERENCES `bamp_db`.`contents` (`id`)
      ON DELETE CASCADE
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_brand_application_brand_identity1`
    FOREIGN KEY (`brand_identity_id`)
      REFERENCES `bamp_db`.`brand_identities` (`id`)
      ON DELETE CASCADE
      ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`brand_app_items`
-----

```

```

DROP TABLE IF EXISTS `bamp_db`.`brand_app_items` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`brand_app_items` (
  `id` INT(11) NOT NULL,
  `brand_application_id` INT(11) NOT NULL,
  `name` VARCHAR(100) NOT NULL,
  `description` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_brand_app_item_content1_idx` (`id` ASC),
  INDEX `fk_brand_app_item_brand_application1_idx` (`brand_application_id` ASC),
  CONSTRAINT `fk_brand_app_item_content10`
    FOREIGN KEY (`id`)
    REFERENCES `bamp_db`.`contents` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_brand_app_item_brand_application10`
    FOREIGN KEY (`brand_application_id`)
    REFERENCES `bamp_db`.`brand_applications` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`brand_app_item_files`
-----
DROP TABLE IF EXISTS `bamp_db`.`brand_app_item_files` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`brand_app_item_files` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `app_item_id` INT(11) NOT NULL,
  `image` VARCHAR(255) NULL DEFAULT NULL,
  `path` VARCHAR(255) NOT NULL,
  `extension` VARCHAR(10) NULL,
  `size` VARCHAR(50) NULL,
  `created_at` TIMESTAMP NULL,
  `updated_at` TIMESTAMP NULL,
  `deleted_at` TIMESTAMP NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_brand_app_files_brand_app_item1_idx` (`app_item_id` ASC),
  CONSTRAINT `fk_brand_app_files_brand_app_item1`
    FOREIGN KEY (`app_item_id`)
    REFERENCES `bamp_db`.`brand_app_items` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`brand_users`
-----
DROP TABLE IF EXISTS `bamp_db`.`brand_users` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`brand_users` (
  `id` INT(11) NOT NULL,
  `brand_id` INT(11) NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_user_has_brand1_brand1_idx` (`brand_id` ASC),
  INDEX `fk_bu_user_role1_idx` (`id` ASC),
  CONSTRAINT `fk_user_has_brand1_brand1`
    FOREIGN KEY (`brand_id`)
    REFERENCES `bamp_db`.`brands` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,

```

```

CONSTRAINT `fk_bu_user_role1`
  FOREIGN KEY (`id`)
  REFERENCES `bamp_db`.`user_roles` (`id`)
  ON DELETE CASCADE
  ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`visual_identities`
-----
DROP TABLE IF EXISTS `bamp_db`.`visual_identities` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`visual_identities` (
  `id` INT(11) NOT NULL,
  `brand_identity_id` INT(11) NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_visual_identity_brand_identity1_idx` (`brand_identity_id` ASC),
  UNIQUE INDEX `brand_identity_id_UNIQUE` (`brand_identity_id` ASC),
  CONSTRAINT `fk_visual_identity_content`
    FOREIGN KEY (`id`)
    REFERENCES `bamp_db`.`contents` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_visual_identity_brand_identity1`
    FOREIGN KEY (`brand_identity_id`)
    REFERENCES `bamp_db`.`brand_identities` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`colourways`
-----
DROP TABLE IF EXISTS `bamp_db`.`colourways` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`colourways` (
  `id` INT(11) NOT NULL,
  `visual_identity_id` INT(11) NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_colourway_visual_identity2_idx` (`visual_identity_id` ASC),
  UNIQUE INDEX `visual_identity_id_UNIQUE` (`visual_identity_id` ASC),
  CONSTRAINT `fk_colourway_visual_identity2`
    FOREIGN KEY (`visual_identity_id`)
    REFERENCES `bamp_db`.`visual_identities` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_colourway_content`
    FOREIGN KEY (`id`)
    REFERENCES `bamp_db`.`contents` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`colourway_palettes`
-----
DROP TABLE IF EXISTS `bamp_db`.`colourway_palettes` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`colourway_palettes` (
  `id` INT(11) NOT NULL,

```



```

`colourway_id` INT(11) NOT NULL,
`type` VARCHAR(100) NULL DEFAULT NULL,
`description` VARCHAR(255) NULL DEFAULT NULL,
PRIMARY KEY (`id`),
INDEX `fk_colourway_palette_colourway1_idx` (`colourway_id` ASC),
CONSTRAINT `fk_colourway_palette_colourway1`
  FOREIGN KEY (`colourway_id`)
  REFERENCES `bamp_db`.`colourways` (`id`)
  ON DELETE CASCADE
  ON UPDATE NO ACTION,
CONSTRAINT `fk_colourway_palette_content`
  FOREIGN KEY (`id`)
  REFERENCES `bamp_db`.`contents` (`id`)
  ON DELETE CASCADE
  ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`colourway_colours`
-----
DROP TABLE IF EXISTS `bamp_db`.`colourway_colours` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`colourway_colours` (
  `id` INT(11) NOT NULL AUTO_INCREMENT COMMENT ' ',
  `palette_id` INT(11) NOT NULL,
  `name` VARCHAR(100) NOT NULL,
  `pantone` VARCHAR(50) NULL DEFAULT NULL,
  `hex` CHAR(7) NOT NULL,
  `cmyk` CHAR(15) NULL DEFAULT NULL,
  `ra1` CHAR(9) NULL DEFAULT NULL,
  `_3m` VARCHAR(45) NULL DEFAULT NULL,
  `created_at` TIMESTAMP NULL,
  `updated_at` TIMESTAMP NULL,
  `deleted_at` TIMESTAMP NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_colour_palette1_idx` (`palette_id` ASC),
  CONSTRAINT `fk_colour_palette1`
    FOREIGN KEY (`palette_id`)
    REFERENCES `bamp_db`.`colourway_palettes` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`galleries`
-----
DROP TABLE IF EXISTS `bamp_db`.`galleries` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`galleries` (
  `id` INT(11) NOT NULL,
  `galleryable_type` VARCHAR(64) NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_gallery_content`
    FOREIGN KEY (`id`)
    REFERENCES `bamp_db`.`contents` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
-----

```

```

-- Table `bamp_db`.`colourway_colour_applications`
-----
DROP TABLE IF EXISTS `bamp_db`.`colourway_colour_applications` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`colourway_colour_applications` (
  `id` INT(11) NOT NULL COMMENT '',
  `colourway_id` INT(11) NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_colourway_colour_applications_colourways1_idx` (`colourway_id` ASC),
  UNIQUE INDEX `colourway_id_UNIQUE` (`colourway_id` ASC),
  CONSTRAINT `fk_colourway_gallery_image_gallery1`
    FOREIGN KEY (`id`)
    REFERENCES `bamp_db`.`galleries` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_colourway_colour_applications_colourways1`
    FOREIGN KEY (`colourway_id`)
    REFERENCES `bamp_db`.`colourways` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----

-- Table `bamp_db`.`colourway_custom_systems`
-----
DROP TABLE IF EXISTS `bamp_db`.`colourway_custom_systems` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`colourway_custom_systems` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `colourway_id` INT(11) NOT NULL,
  `title` VARCHAR(50) NOT NULL,
  `created_at` TIMESTAMP NULL,
  `updated_at` TIMESTAMP NULL,
  `deleted_at` TIMESTAMP NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_colourway_custom_system_colourway1_idx` (`colourway_id` ASC),
  CONSTRAINT `fk_colourway_custom_system_colourway1`
    FOREIGN KEY (`colourway_id`)
    REFERENCES `bamp_db`.`colourways` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----

-- Table `bamp_db`.`colourway_colour_in_custom_system`
-----
DROP TABLE IF EXISTS `bamp_db`.`colourway_colour_in_custom_system` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`colourway_colour_in_custom_system` (
  `colour_id` INT(11) NOT NULL,
  `custom_system_id` INT(11) NOT NULL,
  `code` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`colour_id`, `custom_system_id`),
  INDEX `fk_colourway_colour_has_colourway_custom_system_colourway_c_idx`
    (`custom_system_id` ASC),
  INDEX `fk_colourway_colour_has_colourway_custom_system_colourway_c_idx1` (`colour_id`
  ASC),
  CONSTRAINT `fk_colourway_colour_has_colourway_custom_system_colourway_col1`
    FOREIGN KEY (`colour_id`)
    REFERENCES `bamp_db`.`colourway_colours` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,

```

```

CONSTRAINT `fk_colourway_colour_has_colourway_custom_system_colourway_cus1`
  FOREIGN KEY (`custom_system_id`)
  REFERENCES `bamp_db`.`colourway_custom_systems` (`id`)
  ON DELETE CASCADE
  ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COMMENT = 'colourway_custom_system_id';

-----
-- Table `bamp_db`.`content_groups`
-----
DROP TABLE IF EXISTS `bamp_db`.`content_groups` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`content_groups` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `creator_id` INT(11) NOT NULL,
  `title` VARCHAR(255) NOT NULL,
  `is_public` BIT NOT NULL DEFAULT 0,
  `created_at` TIMESTAMP NULL,
  `updated_at` TIMESTAMP NULL,
  `deleted_at` TIMESTAMP NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_content_group_user1_idx` (`creator_id` ASC),
  CONSTRAINT `fk_content_group_user1`
    FOREIGN KEY (`creator_id`)
    REFERENCES `bamp_db`.`users` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`content_group_shared_with_bu`
-----
DROP TABLE IF EXISTS `bamp_db`.`content_group_shared_with_bu` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`content_group_shared_with_bu` (
  `shared_with_id` INT(11) NOT NULL,
  `shared_content_group_id` INT(11) NOT NULL,
  PRIMARY KEY (`shared_with_id`, `shared_content_group_id`),
  INDEX `fk_content_group_shared_with_bu_bu1_idx` (`shared_with_id` ASC),
  INDEX `fk_content_shared_with_bu_content_group1_idx` (`shared_content_group_id` ASC),
  CONSTRAINT `fk_content_group_shared_with_bu_bu1`
    FOREIGN KEY (`shared_with_id`)
    REFERENCES `bamp_db`.`brand_users` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_content_shared_with_bu_content_group1`
    FOREIGN KEY (`shared_content_group_id`)
    REFERENCES `bamp_db`.`content_groups` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`content_in_group`
-----
DROP TABLE IF EXISTS `bamp_db`.`content_in_group` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`content_in_group` (
  `content_id` INT(11) NOT NULL,

```

```

`group_id` INT(11) NOT NULL,
PRIMARY KEY (`content_id`, `group_id`),
INDEX `fk_content_group_has_content_content1_idx` (`content_id` ASC),
INDEX `fk_content_group_has_content_content_group1_idx` (`group_id` ASC),
CONSTRAINT `fk_content_group_has_content_content_group1`
  FOREIGN KEY (`group_id`)
    REFERENCES `bamp_db`.`content_groups` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
CONSTRAINT `fk_content_group_has_content_content1`
  FOREIGN KEY (`content_id`)
    REFERENCES `bamp_db`.`contents` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`gallery_albums`
-----
DROP TABLE IF EXISTS `bamp_db`.`gallery_albums` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`gallery_albums` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `gallery_id` INT(11) NOT NULL,
  `name` VARCHAR(100) NOT NULL,
  `description` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_image_album_image_gallery1_idx` (`gallery_id` ASC),
  CONSTRAINT `fk_image_album_image_gallery1`
    FOREIGN KEY (`gallery_id`)
      REFERENCES `bamp_db`.`galleries` (`id`)
      ON DELETE CASCADE
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_gallery_album_content`
    FOREIGN KEY (`id`)
      REFERENCES `bamp_db`.`contents` (`id`)
      ON DELETE CASCADE
      ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`gallery_album_images`
-----
DROP TABLE IF EXISTS `bamp_db`.`gallery_album_images` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`gallery_album_images` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `album_id` INT(11) NOT NULL,
  `name` VARCHAR(100) NOT NULL,
  `caption` VARCHAR(255) NULL DEFAULT NULL,
  `path` VARCHAR(255) NOT NULL,
  `thumbnail` VARCHAR(255) NULL DEFAULT NULL,
  `extension` VARCHAR(10) NULL,
  `size` VARCHAR(50) NULL,
  `created_at` TIMESTAMP NULL,
  `updated_at` TIMESTAMP NULL,
  `deleted_at` TIMESTAMP NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_image_image_album1_idx` (`album_id` ASC),
  CONSTRAINT `fk_image_image_album1`
    FOREIGN KEY (`album_id`)
      REFERENCES `bamp_db`.`gallery_albums` (`id`)

```

```

    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`imageries`
-----
DROP TABLE IF EXISTS `bamp_db`.`imageries` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`imageries` (
  `id` INT NOT NULL,
  `visual_identity_id` INT(11) NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_imagery_visual_identity2_idx` (`visual_identity_id` ASC),
  UNIQUE INDEX `visual_identity_id_UNIQUE` (`visual_identity_id` ASC),
  CONSTRAINT `fk_imagery_visual_identity2`
    FOREIGN KEY (`visual_identity_id`)
    REFERENCES `bamp_db`.`visual_identities` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_imagery_content`
    FOREIGN KEY (`id`)
    REFERENCES `bamp_db`.`contents` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`imagery_galleries`
-----
DROP TABLE IF EXISTS `bamp_db`.`imagery_galleries` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`imagery_galleries` (
  `id` INT(11) NOT NULL,
  `imagery_id` INT(11) NOT NULL,
  `name` VARCHAR(100) NULL DEFAULT NULL,
  `description` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_imagery_gallery_imagery1_idx` (`imagery_id` ASC),
  CONSTRAINT `fk_imagery_gallery_image_gallery1`
    FOREIGN KEY (`id`)
    REFERENCES `bamp_db`.`galleries` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_imagery_gallery_imagery1`
    FOREIGN KEY (`imagery_id`)
    REFERENCES `bamp_db`.`imageries` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`independent_brands`
-----
DROP TABLE IF EXISTS `bamp_db`.`independent_brands` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`independent_brands` (
  `id` INT(11) NOT NULL,
  `boa_id` INT(11) NOT NULL,
  PRIMARY KEY (`id`),

```

```

INDEX `fk_single_brand_brand1_idx` (`id` ASC),
INDEX `fk_independent_brand_user_role1_idx` (`boa_id` ASC),
CONSTRAINT `fk_single_brand_brand1`
  FOREIGN KEY (`id`)
  REFERENCES `bamp_db`.`brands` (`id`)
  ON DELETE CASCADE
  ON UPDATE NO ACTION,
CONSTRAINT `fk_independent_brand_user_role1`
  FOREIGN KEY (`boa_id`)
  REFERENCES `bamp_db`.`user_roles` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`logos`
-----
DROP TABLE IF EXISTS `bamp_db`.`logos` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`logos` (
  `id` INT(11) NOT NULL,
  `visual_identity_id` INT(11) NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_logo_visual_identity1_idx` (`visual_identity_id` ASC),
  UNIQUE INDEX `visual_identity_id_UNIQUE` (`visual_identity_id` ASC),
  CONSTRAINT `fk_logo_visual_identity1`
    FOREIGN KEY (`visual_identity_id`)
    REFERENCES `bamp_db`.`visual_identities` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_logo_content`
    FOREIGN KEY (`id`)
    REFERENCES `bamp_db`.`contents` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`logo_versions`
-----
DROP TABLE IF EXISTS `bamp_db`.`logo_versions` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`logo_versions` (
  `id` INT(11) NOT NULL,
  `logo_id` INT(11) NOT NULL,
  `title` VARCHAR(255) NOT NULL,
  `path` VARCHAR(255) NULL,
  `thumbnail` VARCHAR(255) NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_logo_version_logo1_idx` (`logo_id` ASC),
  CONSTRAINT `fk_logo_version_logo1`
    FOREIGN KEY (`logo_id`)
    REFERENCES `bamp_db`.`logos` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_logo_version_1`
    FOREIGN KEY (`id`)
    REFERENCES `bamp_db`.`contents` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

```

```

-----
-- Table `bamp_db`.`logo_colours`
-----
DROP TABLE IF EXISTS `bamp_db`.`logo_colours` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`logo_colours` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `version_id` INT(11) NOT NULL,
  `type` VARCHAR(100) NOT NULL,
  `description` TEXT NULL DEFAULT NULL,
  `created_at` TIMESTAMP NULL,
  `updated_at` TIMESTAMP NULL,
  `deleted_at` TIMESTAMP NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `version_id_UNIQUE` (`version_id` ASC),
  INDEX `fk_logo_colour_logo_version1_idx` (`version_id` ASC),
  CONSTRAINT `fk_logo_colour_logo_version1`
    FOREIGN KEY (`version_id`)
      REFERENCES `bamp_db`.`logo_versions` (`id`)
      ON DELETE CASCADE
      ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`logo_exceptions`
-----
DROP TABLE IF EXISTS `bamp_db`.`logo_exceptions` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`logo_exceptions` (
  `id` INT(11) NOT NULL,
  `logo_id` INT(11) NOT NULL,
  `group` ENUM('web','icon','translation','commemorative') NOT NULL,
  `subgroup` VARCHAR(50) NOT NULL,
  `thumbnail` VARCHAR(255) NULL DEFAULT NULL,
  `description` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_logo_exception_logo1_idx` (`logo_id` ASC),
  CONSTRAINT `fk_logo_exception_logo1`
    FOREIGN KEY (`logo_id`)
      REFERENCES `bamp_db`.`logos` (`id`)
      ON DELETE CASCADE
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_logo_exception_1`
    FOREIGN KEY (`id`)
      REFERENCES `bamp_db`.`contents` (`id`)
      ON DELETE CASCADE
      ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`logo_exception_files`
-----
DROP TABLE IF EXISTS `bamp_db`.`logo_exception_files` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`logo_exception_files` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `exception_id` INT(11) NOT NULL,
  `path` VARCHAR(255) NOT NULL,
  `extension` VARCHAR(10) NULL,
  `size` VARCHAR(50) NULL,

```

```

`created_at` TIMESTAMP NULL,
`updated_at` TIMESTAMP NULL,
`deleted_at` TIMESTAMP NULL,
PRIMARY KEY (`id`),
INDEX `fk_logo_file_copy1_logo_exception1_idx` (`exception_id` ASC),
CONSTRAINT `fk_logo_file_copy1_logo_exception1`
  FOREIGN KEY (`exception_id`)
  REFERENCES `bamp_db`.`logo_exceptions` (`id`)
  ON DELETE CASCADE
  ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`logo_files`
-----
DROP TABLE IF EXISTS `bamp_db`.`logo_files` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`logo_files` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `version_id` INT(11) NOT NULL,
  `path` VARCHAR(255) NOT NULL,
  `extension` VARCHAR(10) NULL,
  `size` VARCHAR(50) NULL,
  `created_at` TIMESTAMP NULL,
  `updated_at` TIMESTAMP NULL,
  `deleted_at` TIMESTAMP NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_logo_file_logo_version1_idx` (`version_id` ASC),
  CONSTRAINT `fk_logo_file_logo_version1`
    FOREIGN KEY (`version_id`)
    REFERENCES `bamp_db`.`logo_versions` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`logo_formats`
-----
DROP TABLE IF EXISTS `bamp_db`.`logo_formats` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`logo_formats` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `version_id` INT(11) NOT NULL,
  `size` ENUM('standard','small','large') NOT NULL,
  `description` TEXT NULL DEFAULT NULL,
  `created_at` TIMESTAMP NULL,
  `updated_at` TIMESTAMP NULL,
  `deleted_at` TIMESTAMP NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_logo_formats_logo_versions1_idx` (`version_id` ASC),
  UNIQUE INDEX `versions_id_UNIQUE` (`version_id` ASC),
  CONSTRAINT `fk_logo_formats_logo_versions1`
    FOREIGN KEY (`version_id`)
    REFERENCES `bamp_db`.`logo_versions` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`logo_measurements`

```



```

-----
DROP TABLE IF EXISTS `bamp_db`.`logo_measurements` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`logo_measurements` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `version_id` INT(11) NOT NULL,
  `path` VARCHAR(255) NOT NULL,
  `type` VARCHAR(50) NOT NULL,
  `description` TEXT NULL DEFAULT NULL,
  `created_at` TIMESTAMP NULL,
  `updated_at` TIMESTAMP NULL,
  `deleted_at` TIMESTAMP NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_logo_measurement_logo_version1_idx` (`version_id` ASC),
  CONSTRAINT `fk_logo_measurement_logo_version1`
    FOREIGN KEY (`version_id`)
    REFERENCES `bamp_db`.`logo_versions` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`logo_types`
-----
DROP TABLE IF EXISTS `bamp_db`.`logo_types` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`logo_types` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `version_id` INT(11) NOT NULL,
  `type` VARCHAR(100) NOT NULL,
  `description` TEXT NULL DEFAULT NULL,
  `created_at` TIMESTAMP NULL,
  `updated_at` TIMESTAMP NULL,
  `deleted_at` TIMESTAMP NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `version_id_UNIQUE` (`version_id` ASC),
  INDEX `fk_logo_type_logo_version1_idx` (`version_id` ASC),
  CONSTRAINT `fk_logo_type_logo_version1`
    FOREIGN KEY (`version_id`)
    REFERENCES `bamp_db`.`logo_versions` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`typographies`
-----
DROP TABLE IF EXISTS `bamp_db`.`typographies` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`typographies` (
  `id` INT(11) NOT NULL,
  `visual_identity_id` INT(11) NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_typography_visual_identity2_idx` (`visual_identity_id` ASC),
  UNIQUE INDEX `visual_identity_id_UNIQUE` (`visual_identity_id` ASC),
  CONSTRAINT `fk_typography_visual_identity2`
    FOREIGN KEY (`visual_identity_id`)
    REFERENCES `bamp_db`.`visual_identities` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_typography_content`
    FOREIGN KEY (`id`)

```

```

REFERENCES `bamp_db`.`contents` (`id`)
ON DELETE CASCADE
ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`typography_types`
-----
DROP TABLE IF EXISTS `bamp_db`.`typography_types` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`typography_types` (
  `id` INT(11) NOT NULL,
  `typography_id` INT(11) NOT NULL,
  `title` VARCHAR(255) NOT NULL,
  `description` TEXT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_typography_type_typography1_idx` (`typography_id` ASC),
  CONSTRAINT `fk_typography_type_typography1`
    FOREIGN KEY (`typography_id`)
    REFERENCES `bamp_db`.`typographies` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_typography_type_content`
    FOREIGN KEY (`id`)
    REFERENCES `bamp_db`.`contents` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `bamp_db`.`typography_typefaces`
-----
DROP TABLE IF EXISTS `bamp_db`.`typography_typefaces` ;

CREATE TABLE IF NOT EXISTS `bamp_db`.`typography_typefaces` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `type_id` INT(11) NOT NULL,
  `subtype` VARCHAR(255) NULL,
  `name` VARCHAR(255) NULL DEFAULT NULL,
  `family_name` VARCHAR(255) NULL DEFAULT NULL,
  `generic_family` VARCHAR(255) NULL DEFAULT NULL,
  `author` VARCHAR(255) NULL DEFAULT NULL,
  `webfont_url` VARCHAR(500) NULL DEFAULT NULL,
  `path` VARCHAR(255) NULL DEFAULT NULL,
  `cascade` VARCHAR(255) NULL DEFAULT NULL,
  `seller_name` VARCHAR(255) NULL DEFAULT NULL,
  `seller_url` VARCHAR(255) NULL DEFAULT NULL,
  `description` TEXT NULL DEFAULT NULL,
  `licensed` BIT NULL DEFAULT NULL,
  `created_at` TIMESTAMP NULL,
  `updated_at` TIMESTAMP NULL,
  `deleted_at` TIMESTAMP NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_typeface_type1_idx` (`type_id` ASC),
  CONSTRAINT `fk_typeface_type1`
    FOREIGN KEY (`type_id`)
    REFERENCES `bamp_db`.`typography_types` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

```

```
SET SQL_MODE=@OLD_SQL_MODE;  
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;  
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Anexo 4.

Bootstrap vs Foundation: User Interfaces and Widges

UI e Widgets	Bootstrap	Foundation
Alerts	SIM	SIM
Badges	SIM	SIM .label.round
Breadcrumbs	SIM	SIM
Buttons	SIM	SIM
Carousel	SIM	SIM
Collapse/Accordion	SIM	SIM
Dropdown	SIM	SIM
Forms	SIM	SIM
Form Validation	NÃO	SIM
Grids	SIM	SIM
Icons	SIM	SIM
Labels	SIM	SIM
Lists	SIM	SIM
Lists (horizontal)	SIM	SIM
Media Object	SIM	NÃO
Modal window	SIM	SIM
Navigation	SIM	SIM
Pagination	SIM	SIM
Panels	SIM	SIM
Popovers	SIM	SIM
Pricing tables	NÃO	SIM
Print styles	SIM	SIM
Progress bars	SIM	SIM
Responsive media	NÃO	SIM
Right-to-Left	NÃO	SIM

Scrollspy	SIM	SIM
Tables	SIM	SIM
Tabs	SIM	SIM
Thumbnails	SIM	SIM
Tooltips	SIM	SIM
Tour	NÃO	SIM
Typeahead	NÃO	NÃO
Typography	SIM	SIM
Video scaling	NÃO	SIM

Anexo 5.

Especificações Iniciais

Especificações iniciais fornecidas ao estagiário no início do projecto:

O objectivo é criar uma plataforma *online* em que seja possível criar diferentes contas para as diferentes marcas/sistemas de identidade visual. O acesso a cada conta será sempre restrito e limitado aos utilizadores dessa marca em particular. Esta aplicação será de acesso restrito e deverá ser desenvolvida tendo em conta os seguintes pontos genéricos:

Existem inicialmente 3 perfis de acesso à aplicação, com permissões distintas:

6. FBA (os utilizadores deste perfil serão os únicos com autorização para efectuar alterações na informação disponível no sistema. Deverá existir um Administrador dentro deste perfil que terá os privilégios de criar os restantes utilizadores dos diferentes perfis previstos)
7. Cliente (terá acesso ao manual da sua (Marca(s) para consulta e download de ficheiros da marca ou modelos de documento. Deverá igualmente existir a possibilidade de um utilizador mestre do cliente criar utilizadores do perfil Cliente ou Terceiros com possibilidade de definição de restrições de acesso a diferentes partes do Manual)
8. Terceiros (serão utilizadores pontuais a quem o cliente possa querer dar acesso limitado no tempo e âmbito para utilizar informação ou documentos do manual – por exemplo uma entidade externa que tem necessidade de aplicar a marca do cliente). Na prática este perfil pode ser considerado um perfil semelhante ao do cliente mas com restrições de acesso maiores.

Anexo 6. Conclusões Pessoais

Com o término do estágio poder-se-ia dizer que muito ficou por cumprir. A verdade porém é outra! Tendo em conta que os percalços ocorridos no decorrer do projecto obrigaram a uma substancial reestruturação do mesmo e consequentes atrasos; todos os esforços foram feitos para conseguir entregar uma solução estruturalmente bem pensada, planeada e com qualidade. Sendo a proposta inicial a elaboração de uma plataforma web para gestão de manuais de normas para a FBA e o objectivo final uma plataforma web de gestão de marcas pode afirmar-se que foi alcançado um ponto intermédio de concretização.

A reestruturação mais penosa, em termos temporais, está relacionada com a escolha do *Content Management Framework* (CMF) Drupal; escolhida pelo estagiário e aceite pelos orientadores como solução ao qual se recorreria como base para a implementação do projecto, no entanto a sua utilização foi descartada a meio do estágio por falta de complexidade técnica. Até ao seu abandono, o projecto estava a ser estruturado centrado no uso do CMF o que invalidou os estudos, testes e identificação de módulos existentes para uso durante a implementação. Realça-se o facto de neste documento não se fazer menção nem às análises nem às estratégias seguidas até esse ponto cujo tinham sido pensadas para o uso do Drupal.

Motivado pelo grande volume de trabalho da agência, embora planeado para ser da responsabilidade da FBA, a elaboração dos *templates* acabou por recair sobre o estagiário. Este acontecimento veio influenciar pela negativa o avanço do projecto, dado que ocorreu na fase de implementação e obrigou o estagiário a procurar soluções para poder ultrapassar mais este contratempo.

Em consequência do exposto, escolhas tiveram de ser que efectuadas afectando sobretudo a implementação de funcionalidades da plataforma. Conjuntamente com os orientadores, foi decidido que se optaria por uma implementação mais fina mas ao mesmo tempo que demonstrasse as capacidades do projecto. Desta forma dois dos casos de utilização foram seleccionados para fazer o fazer, são estes os utilizadores com funções de administração de uma marca que não dependem de uma agência (BOA) e os utilizadores com funções apenas de consulta de conteúdos partilhados (BU). A escolha do BOA recaiu pela similaridade de funcionalidades entre utilizadores com funções de BASA, BOA e BOU o que tornará a sua implementação futura uma questão de adaptabilidade das funções já implementadas. Já para o caso dos utilizadores BU, a escolha é justificada por ser uma demonstração das possibilidades de partilha da plataforma.

Em síntese de conclusão ao alcançado neste projecto poder-se-á então dizer que a FBA poderá transformar os manuais de normas de identidade visual em manuais online podendo pô-los ao dispor dos seus clientes para consulta.

Anexo 7. Diagrama ER completo

