

**Universidade do Minho**

Departamento de Informática

Pedro Miguel Pereira Tomé Branco

**Digital Signage – Aplicação Baseada em Eventos  
Usando a API Google Calendar**

Dissertação de Mestrado  
Mestrado em Engenharia Informática

Trabalho realizado sob a orientação do  
**Professor Doutor António Nestor Ribeiro**

Outubro de 2011



**Digital Signage – Aplicação Baseada em Eventos  
Usando a API Google Calendar**

# Agradecimentos

Agradeço ao meu orientador, Prof. Dr. António Nestor Ribeiro, por ter estado sempre disponível para o esclarecimento das minhas dúvidas na escrita desta dissertação. Agradeço também ao meu orientador na empresa Ubisign, Dr. Helder Manuel da Silva Pinto, pelo fornecimento da informação e esclarecimento de dúvidas para o desenvolvimento da aplicação de *software*, sem esquecer o papel relevante da Ubisign na disponibilização do enunciado do projecto de *software*. Finalmente, deixo uma palavra de apreço aos meus familiares, colegas e amigos, que sempre me estimularam e apoiaram na realização deste trabalho.



# Resumo

O uso de painéis de *digital signage* ou *displays* (ou, em português, painéis digitais) está cada vez mais a ser implementado nos locais públicos e semi-públicos, uma vez que é uma forma actual de apresentar, de um modo dinâmico, informação, publicidade e conteúdos de entretenimento. Desta forma, as pessoas expostas a esta tecnologia passam a ter um acesso mais rápido e actualizado à informação sobre tudo o que as rodeia.

Os ecrãs dos *displays* têm evoluído no sentido da melhoria da sua qualidade. Além disso, a descida dos preços torna esta tecnologia bastante mais acessível para ser aplicada em vários sectores, nomeadamente na indústria, no comércio, na educação, na saúde, etc., substituindo os meios tradicionais publicitários e informativos, baseados em papel [Müller, 2009].

Uma rede de painéis digitais permite a actualização dos seus conteúdos de forma remota, com base num servidor central que controla toda a informação apresentada na rede, enquanto que nos meios tradicionais a actualização de conteúdos é muito mais cara, demorada e de difícil gestão.

Como as pessoas olham pouco tempo para os painéis digitais, exige-se um esforço muito grande no estudo do local onde estes são instalados e na modelação dos conteúdos a serem apresentados, de forma a que o ambiente se torne o mais atractivo possível e, deste modo, se promova a atenção de quem passa pelos painéis [Huang et al., 2008]. Também existem *displays* que usam tecnologias mais sofisticadas, permitindo a adaptação automática dos conteúdos apresentados em função do contexto envolvente, fazendo com que a informação seja mais direccionada ao público, sem a necessidade de controlo humano [Müller, 2007, Payne et al., 2006].

Dado o crescente sucesso, à escala planetária, da *digital signage*, têm surgindo cada vez mais soluções de *software* aplicadas nesta vertente. Por conseguinte, o tema principal desta dissertação incidirá sobre o desenvolvi-

mento de uma aplicação *web* para painéis de *digital signage*, sugerida pela empresa Ubisign, com o objectivo de permitir configurar visualizações com informação sobre eventos provenientes do Google Calendar.

Uma vez que a promoção de eventos geralmente exige custos elevados de *design* e produção, é necessário minimizá-los no contexto das redes de *digital signage*. Para tal, existem determinadas ferramentas *on-line* de gestão de eventos, como o Google Calendar, com a função de permitir que o utilizador especifique eventos que vão ocorrer e efectue o seu escalonamento com base em calendários. Do ponto de vista de um *developer*, estas ferramentas evitam a necessidade de implementar outros *softwares* de gestão de eventos, permitindo também desenvolver outras aplicações que comuniquem com estas ferramentas, através de APIs apropriadas e bem documentadas. Para tirar partido disto, a aplicação a desenvolver terá de recorrer à API Google Calendar para disponibilizar, de forma personalizada, informação sobre eventos que estejam planeados para um dado sítio com uma rede de *digital signage* instalada e, para esse efeito, terá de ser integrada no serviço Ubisign.com.

**Área de Aplicação:** Interfaces de informação e apresentação.

**Palavras-chave:** Painéis de *Digital Signage*; Google Calendar; Informação sobre Eventos.

# Abstract

The usage of digital signage displays in public and semi-public places is increasing, because they are a modern way of dynamically displaying information, advertisements and entertainment content. In this way, those exposed to digital signage have a quick and up to date access to information around them.

Display screens have been improving in quality. Also prices have been falling, making this technology more affordable in order to be applied in various areas (particularly in industry, commerce, education, health, etc.), replacing the paper-based traditional means of advertising and information [Müller, 2009].

A digital signage network allows the remote update of its contents, based on a central server that manages all the information presented on the network, while in traditional means the contents' updating process is much more costly, time consuming and difficult to manage.

Because people glance briefly into the displays, their content and locus must be carefully studied, so that the environment becomes more attractive. Thus it is a way to promote the attention of the passersby [Huang et al., 2008]. Some displays use more sophisticated technologies, allowing a context-based automatic adaptation of their content. So the information is targeted to the public, without the need for human control [Müller, 2007, Payne et al., 2006].

Ever more software solutions applied to digital signage have emerged due to its increasing success in a worldwide scale. Therefore the development of a digital signage web application that allows the set up of views with event information coming from Google Calendar (proposed by Ubisign company) will be the main theme of this dissertation.

Since event promotion generally requires high costs of design and produc-



tion, they must be minimized in digital signage networks. To this end, some online tools for managing events, such as Google Calendar, allow calendar-based event scheduling and its specification. From a developer's perspective, these tools avoid the need to implement other event management software, allowing one to develop other applications that communicate with these tools through appropriate and well documented APIs. To take advantage of this, the application that will be developed will need to use the Google Calendar API to provide information in a personalized way about events that are planned for a given place with an installed digital signage network. For this purpose, the application must be integrated in the Ubsign.com service.

**Area of Application:** Information Interfaces and Presentation.

**Keywords:** Digital Signage Displays; Google Calendar; Event Information.

# Conteúdo

<b>Lista de Siglas e Acrónimos</b>	<b>xi</b>
<b>Lista de Figuras</b>	<b>xiii</b>
<b>Lista de Tabelas</b>	<b>xix</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Motivação . . . . .	1
1.2 Objectivos . . . . .	2
1.3 Organização . . . . .	4
<b>2 Estado da Arte</b>	<b>7</b>
2.1 Introdução . . . . .	7
2.2 Redes de Digital Signage – DSNs . . . . .	9
2.2.1 DSNs: Factores Técnicos, Estratégicos e Comerciais . .	10
2.2.2 DSNs: Factores Psicológicos . . . . .	14
2.2.3 DSNs: Framework Psico-Estratégica . . . . .	17
2.3 <i>Surveys</i> sobre o <i>design</i> em painéis digitais . . . . .	19
2.3.1 <i>Survey</i> sobre <i>digital signage</i> em ambientes exteriores .	19
2.3.2 <i>Survey</i> sobre <i>digital signage</i> num ambiente académico	22
2.3.3 <i>Survey</i> sobre <i>digital signage</i> “semi-público” para gru- pos pequenos e co-localizados . . . . .	24
2.4 Problemas inerentes à publicidade em <i>digital signage</i> . . . . .	26
2.4.1 Publicidade com alvos definidos . . . . .	27
2.4.2 Externalidades negativas da publicidade . . . . .	28
2.5 <i>Displays</i> públicos inteligentes baseados no contexto e em lei- lões publicitários . . . . .	29
2.5.1 O <i>BluScreen</i> . . . . .	29

2.5.2	O <i>iDisplay</i> . . . . .	32
2.5.3	Conclusões . . . . .	34
<b>3</b>	<b>Especificação do Problema</b>	<b>35</b>
3.1	Introdução . . . . .	35
3.2	Ferramentas <i>on-line</i> de Gestão de Eventos . . . . .	36
3.2.1	Google Calendar . . . . .	37
3.2.2	Microsoft Exchange . . . . .	42
3.2.3	Eventbrite . . . . .	46
3.2.4	Criação de eventos no Facebook . . . . .	47
3.3	Descrição da Aplicação . . . . .	47
3.3.1	Visualização em ecrã inteiro (Full) . . . . .	49
3.3.2	Visualização em coluna (Column) . . . . .	49
3.3.3	Visualização em barra horizontal (Bar) . . . . .	50
3.3.4	Visualização em XML . . . . .	50
3.3.5	Parâmetros de configuração . . . . .	51
3.3.6	Formulário de configurações . . . . .	51
3.3.7	Persistência de configurações e configurações <i>default</i> . . . . .	51
3.3.8	Integração da aplicação no serviço Ubisign.com . . . . .	52
3.3.9	Outros aspectos importantes . . . . .	52
3.4	Modelo do Domínio . . . . .	54
3.5	Análise de Requisitos . . . . .	57
<b>4</b>	<b>A Aplicação DSEventApp</b>	<b>59</b>
4.1	Introdução . . . . .	59
4.2	Requisitos Detalhados . . . . .	61
4.2.1	Visualização em ecrã inteiro (Full) . . . . .	61
4.2.2	Visualização em coluna (Column) . . . . .	62
4.2.3	Visualização em barra horizontal (Bar) . . . . .	62
4.2.4	Visualização em XML . . . . .	64
4.2.5	Parâmetros de Configuração . . . . .	65
4.3	Especificação da Aplicação . . . . .	70
4.3.1	Camada de Interface . . . . .	73
4.3.2	Camada de Negócio . . . . .	97
4.3.3	Camada de Dados . . . . .	105
4.4	Implementação da Aplicação . . . . .	111
4.4.1	Processo de Implementação . . . . .	111

4.4.2	Problemas e decisões tomadas durante a fase de desenvolvimento . . . . .	113
4.4.3	Exemplos de utilização da aplicação . . . . .	116
<b>5</b>	<b>Aplicação da DSEventApp a Casos de Estudo</b>	<b>127</b>
5.1	Introdução . . . . .	128
5.2	Utilização da DSEventApp num Hotel . . . . .	129
5.3	Utilização da DSEventApp num Aeroporto . . . . .	132
5.4	Utilização da DSEventApp em quartéis de bombeiros . . . . .	136
5.5	Integração da DSEventApp no serviço Ubisign.com . . . . .	140
<b>6</b>	<b>Conclusões</b>	<b>149</b>
6.1	Sumário . . . . .	149
6.2	Discussão . . . . .	150
6.3	Trabalho Futuro . . . . .	154
	<b>Bibliografia</b>	<b>157</b>



# Lista de Siglas e Acrónimos

AJAX	<i>Asynchronous Javascript and XML</i>
API	<i>Application Programming Interface</i>
ASP.NET MVC	<i>.NET Active Server Pages (Model-View-Controller)</i>
CRUD	<i>Create Read Update Delete</i>
CSS	<i>Cascading Style Sheets</i>
DSEventApp	<i>Digital Signage Event Application</i>
DSN	<i>Digital Signage Network</i>
ECL	<i>Everyday Computing Lab</i>
GUID	<i>Globally Unique Identifier</i>
HTML	<i>HyperText Markup Language</i>
LAN	<i>Local Area Network</i>
LCD	<i>Liquid Crystal Display</i>
LED	<i>Light Emitting Diode</i>
OAuth	<i>Open Authorization</i>
OLED	<i>Organic Light-Emitting Diode</i>
OpenID	<i>Open Identification</i>
ORM	<i>Object Relational Mapping</i>
RFID	<i>Radio Frequency Identification</i>
RGB	<i>Red-Green-Blue</i>
RSS	<i>Really Simple Syndication</i>
SaaS	<i>Software as a Service</i>
SATIN	<i>Toolkit for making informal ink-based applications</i>
SQL	<i>Structured Query Language</i>
URL	<i>Uniform Resource Locator</i>
WAN	<i>Wide Area Network</i>
WPF	<i>Windows Presentation Foundation</i>
XML	<i>Extensible Markup Language</i>



# Lista de Figuras

2.1	Exemplo de arquitectura para uma DSN . . . . .	12
2.2	Figura que ilustra uma topologia em estrela . . . . .	12
2.3	As pessoas param e sentam-se à frente de pequenos <i>displays</i> , mesmo que na vizinhança existam grandes <i>displays</i> . . . . .	21
2.4	Times Square repleto de anúncios publicitários <i>vs.</i> publicidade moderada em Prinzpalmarkt . . . . .	27
2.5	Arquitectura do <i>BluScreen</i> baseada em agentes . . . . .	31
2.6	Processo de concretização de um leilão para alocar anúncios publicitários no <i>iDisplay</i> . . . . .	34
3.1	Formulário para criar um calendário no Google Calendar . . .	37
3.2	Eventos correspondentes a calendários diferentes (marcados com cor diferente) no Google Calendar . . . . .	38
3.3	Detalhes de um calendário do Google Calendar . . . . .	39
3.4	Exemplo de criar um evento no Google Calendar . . . . .	40
3.5	Formulário completo para criar um evento no Google Calendar	40
3.6	Formulário para criar eventos recorrentes no Google Calendar	41
3.7	Formulário de criação de um novo evento na Microsoft Exchange	43
3.8	Configuração de recorrência de um evento na Microsoft Exchange . . . . .	44
3.9	Calendário de eventos na vista mensal na Microsoft Exchange	44
3.10	Parte do formulário de criação de um evento no Eventbrite . .	45
3.11	Formulário de criação de um <i>ticket</i> no Eventbrite . . . . .	45
3.12	Página <i>web</i> de registo num evento acabado de criar no Eventbrite	46
3.13	Formulário de criação de um novo evento no Facebook . . . .	48
3.14	Formulário do Facebook para responder a um convite de participação num evento . . . . .	48



3.15	Modelo do Domínio . . . . .	55
4.1	Principais zonas correspondentes à localização da informação sobre um evento . . . . .	75
4.2	Dados correspondentes a um evento . . . . .	75
4.3	Visualização Full com toda a informação possível, com a sub-zona do ícone no lado esquerdo e a sub-zona dos dados no lado direito . . . . .	76
4.4	Visualização Full com toda a informação possível, com a sub-zona dos dados no lado esquerdo e a sub-zona do ícone no lado direito . . . . .	77
4.5	Visualização Full com toda a informação possível à excepção da descrição, com a sub-zona do ícone no lado esquerdo e a sub-zona dos dados no lado direito . . . . .	77
4.6	Visualização Full com toda a informação possível à excepção da descrição, com a sub-zona dos dados no lado esquerdo e a sub-zona do ícone no lado direito . . . . .	78
4.7	Visualização Full com o título, ícone do evento e datas/horas de início e fim . . . . .	78
4.8	Visualização Full com título e ícone do evento . . . . .	79
4.9	Visualização Full apenas com o ícone do evento . . . . .	79
4.10	Visualização Full com toda a informação possível à excepção do ícone . . . . .	80
4.11	Visualização Full com o título, data/hora e local do evento . . . . .	80
4.12	Visualização Column com título, data/hora de início, local e ícone dos eventos, localizados por cima do ícone . . . . .	81
4.13	Visualização Column com título e data/hora de início por cima do ícone; local dos eventos por baixo do ícone . . . . .	82
4.14	Visualização Column com o título por cima do ícone; data/hora de início e local dos eventos por baixo do ícone . . . . .	82
4.15	Visualização Column com título, data/hora de início e local por baixo do ícone . . . . .	83
4.16	Visualização Column com título, data/hora de início e local dos eventos . . . . .	83
4.17	Visualização Column com título e data/hora de início dos eventos . . . . .	84

4.18	Visualização Column com título e local dos eventos . . . . .	84
4.19	Visualização Column apenas com o título dos eventos . . . . .	85
4.20	Visualização Bar Fade que apresenta o título, a data/hora e local de cada evento . . . . .	86
4.21	Visualização Bar Fade que apresenta o título e a data/hora de cada evento . . . . .	86
4.22	Visualização Bar Fade que apresenta o título e o local de cada evento . . . . .	87
4.23	Visualização Bar Fade que apresenta apenas o título de cada evento . . . . .	87
4.24	Visualização Bar Scroll que pode apresentar praticamente toda a informação de cada evento à exceção do ícone . . . . .	88
4.25	Um exemplo simples no formato iCalendar . . . . .	89
4.26	Estrutura da visualização XML . . . . .	89
4.27	Parte do formulário para introduzir o URL do calendário e visibilidades/alinhamentos dos elementos de cada evento . . .	91
4.28	Parte do formulário para definir cores (de fundo e fonte), tamanho e família de fonte dos elementos de cada evento . . . .	92
4.29	Parte do formulário para definir parâmetros da componente de filtragem e comportamental . . . . .	94
4.30	Parte do formulário para carregar/persistir/remover configurações . . . . .	95
4.31	Diagrama de sequência que indica os passos que o utilizador deve tomar para fazer <i>update</i> de uma configuração e contém as respectivas reacções do sistema . . . . .	96
4.32	Diagrama de sequência que demonstra as interacções existentes no sistema, após submissão do formulário de configuração para obter uma visualização (Full, Column, Bar ou XML) . .	97
4.33	Diagrama de classes com os parâmetros da componente visual	99
4.34	Diagrama de classes constituído pelos <i>packages</i> Application-Configuration e ServiceAuthentication . . . . .	100
4.35	Diagrama de classes com o <i>package</i> ConfigControllers que contém classes ( <i>controllers</i> ) sobre os formulários de configuração	102
4.36	Diagrama com o <i>package</i> ConfigControllers que contém classes ( <i>controllers</i> ) sobre os formulários de configuração . . . . .	102

4.37	Diagrama com o <i>package</i> EventInfoModel que contém classes com informação de eventos e do respectivo calendário (no caso da visualização XML) . . . . .	103
4.38	Diagrama com o <i>package</i> EventManager que contém classes com a funcionalidade de efectuar <i>queries</i> e retornar eventos .	105
4.39	Diagrama com o <i>package</i> Utils que contém classes com utilidades diversas . . . . .	106
4.40	Diagrama de classes constituído pelos <i>packages</i> Application-Configuration e ServiceAuthentication, onde se realçam as classes persistentes . . . . .	107
4.41	Diagrama ER . . . . .	109
4.42	Diagrama de classes do <i>package</i> InitConfig, com a função principal de inicializar configurações . . . . .	110
4.43	Calendário para a componente de filtragem . . . . .	114
4.44	<i>Color Picker</i> para seleccionar cores de fundo ou de fonte . . .	115
4.45	Menu principal para o utilizador se autenticar num serviço e poder escolher a visualização que pretende configurar (botões bloqueados) . . . . .	117
4.46	Página para o utilizador efectuar a autenticação com uma conta Google . . . . .	117
4.47	Passo adicional para o utilizador efectuar a autenticação com uma conta Google . . . . .	118
4.48	Página para o utilizador autorizar o acesso da DSEventApp à sua conta do Google Calendar . . . . .	118
4.49	Menu principal para o utilizador se autenticar num serviço e poder escolher a visualização que pretende configurar (botões desbloqueados) . . . . .	119
4.50	Exemplo de visualização Full processada pela DSEventApp .	120
4.51	Exemplo de <i>query string</i> de uma configuração para visualização Full . . . . .	121
4.52	Exemplo de visualização Column processada pela DSEventApp	122
4.53	Exemplo de <i>query string</i> de uma configuração para visualização Column . . . . .	122
4.54	Exemplo de visualização Bar Fade processada pela DSEventApp	123
4.55	Exemplo de <i>query string</i> de uma configuração para visualização Bar Fade . . . . .	124

4.56	Exemplo de visualização Bar Scroll processada pela DSEventApp . . . . .	124
4.57	Exemplo de <i>query string</i> de uma configuração para visualização Bar Scroll . . . . .	125
4.58	Formulário para configurar visualizações XML . . . . .	126
4.59	Exemplo de visualização XML processada pela DSEventApp .	126
4.60	Exemplo de <i>query string</i> de uma configuração para visualização XML . . . . .	126
5.1	Exemplo de oito eventos desportivos especificados no Google Calendar . . . . .	130
5.2	Exemplo de eventos apresentados ciclicamente num <i>display</i> .	131
5.3	Exemplo de eventos apresentados ciclicamente num <i>display</i> e com alterações gráficas por parte de uma aplicação externa .	132
5.4	Mapa de voos . . . . .	133
5.5	Exemplo 1 de mapa de voos processado pela DSEventApp . .	134
5.6	Exemplo 2 de mapa de voos processado pela DSEventApp . .	135
5.7	Incêndios na região do Minho . . . . .	137
5.8	Três exemplos do mesmo incêndio, reflectindo o seu ciclo de vida . . . . .	139
5.9	Três instâncias de XML geradas pela DSEventApp para o mesmo incêndio em estados diferentes . . . . .	139
5.10	Exemplos de filtragem de incêndios . . . . .	140
5.11	<i>Layout</i> utilizado para os primeiros dois exemplos do serviço Ubisign.com . . . . .	143
5.12	Visualização Full enquadrada na região 3 do <i>layout</i> da Figura 5.11 . . . . .	143
5.13	Destaque da visualização Full existente no exemplo da Figura 5.12 . . . . .	144
5.14	Visualização Column enquadrada na região 2 do <i>layout</i> da Figura 5.11 . . . . .	145
5.15	Destaque da visualização Column existente no exemplo da Figura 5.14 . . . . .	145
5.16	<i>Layout</i> utilizado para o último exemplo do serviço Ubisign.com	146
5.17	Visualização Bar Scroll enquadrada na região 4 do <i>layout</i> da Figura 5.16 . . . . .	146

5.18 Destaque da visualização Bar Scroll existente no exemplo da  
Figura 5.17 . . . . . 146

# Lista de Tabelas

4.1	Parâmetros da componente visual (parte 1)	67
4.2	Parâmetros da componente visual (parte 2)	68
4.3	Parâmetros da componente visual (parte 3)	69
4.4	Parâmetros da componente de filtragem	70
4.5	Parâmetros da componente comportamental	71
4.6	Parâmetros adicionais	72



# Capítulo 1

## Introdução

### 1.1 Motivação

A *digital signage*, também designada por *dynamic signage* ou, em português, sinalética digital, consiste numa forma moderna de apresentar dinamicamente conteúdos publicitários, de entretenimento ou informativos, recorrendo a painéis digitais, localizados normalmente em locais públicos, como por exemplo, locais de retalho, escolas, bibliotecas, departamentos, aeroportos, etc..

Com o constante desenvolvimento da tecnologia (sobretudo nas tecnologias que têm vindo a ser adoptadas nos ecrãs, como por exemplo: LCD's, plasmas e LED's), os painéis de *digital signage*, também designados por *displays*, tendem a ser cada vez mais acessíveis em termos de preço e há cada vez mais empresas a adoptar este novo meio tecnológico, pondo de parte os meios tradicionais publicitários/informativos, como por exemplo, o *paper signage* (placares informativos baseados em papel) [Müller, 2009].

Como se sabe, os painéis de *digital signage* têm características electrónicas e digitais, permitindo a actualização dos seus conteúdos em tempo real, quando conectados, por exemplo, a um servidor central, recorrendo à Internet ou a redes proprietárias. Uma outra característica, é poderem fornecer vários conteúdos publicitários no mesmo ecrã, uma vez que estes painéis podem ser de grandes dimensões.

Nos espaços públicos, há várias razões para recorrer a painéis digitais em vez de painéis baseados em papel [Müller, 2009]. Uma delas é a possibilidade de actualizar os seus conteúdos de forma remota, enquanto que os painéis tradicionais requerem mudança dos conteúdos de forma individual, e é necessário ir pessoalmente ao local onde estes se localizam. Outra razão importante é o facto dos painéis digitais não requererem papel nem tinta, sendo, assim, mais “amigos” do ambiente em relação aos meios tradicionais.

Num âmbito publicitário, uma vez que se pretende que as pessoas absorvam a informação transmitida, os agentes publicitários necessitam de utilizar



as estratégias mais apropriadas para cativar a audiência. Por isso, deve-se estudar o local onde os painéis vão ser instalados e modelá-lo de forma a que se promova um ambiente atractivo para o público alvo [Huang et al., 2008]. Para além disso, deve-se modelar os conteúdos a apresentar nos painéis, de forma a que cativa a atenção das pessoas, visto que os painéis de *digital signage* se distinguem bastante da televisão, rádio, computadores, etc., levando a que antes de se instalar um ou vários painéis digitais, se deva ter em conta os factores referidos neste parágrafo.

Em certos painéis digitais que usam recursos mais sofisticados à base de sensores (câmara, sensores *bluetooth*, etc.) [Müller, 2007, Payne et al., 2006], é possível obter uma adaptação dinâmica dos conteúdos apresentados em função do contexto, sem a necessidade de monitorização humana, uma vez que se recorre a heurísticas que têm em conta as variáveis tempo, espaço, audiência, etc..

Como a *digital signage* é uma tecnologia cada vez mais acessível em termos económicos e com sucesso indiscutível à escala planetária, cada vez existem mais projectos de *software* aplicados nesta vertente. Portanto, o cerne desta dissertação está no desenvolvimento de uma aplicação *web*, sugerida pela empresa Ubisign<sup>1</sup>, para permitir configurar visualizações sobre eventos em painéis digitais e, uma vez concretizada a solução final, pretende-se integrar a aplicação no serviço Ubisign.com, para ser utilizado num contexto real e à escala global.

A aplicação a desenvolver vai recorrer à API Google Calendar para apresentar, com um aspecto visual personalizado, informação sobre eventos que estejam planeados para um determinado local coberto por uma rede de *digital signage*.

Tendo em conta que a produção de conteúdo promocional sobre eventos geralmente implica custos elevados de *design* e produção, é importante minimizá-los no contexto das redes de *digital signage*. A existência de determinadas ferramentas *on-line* de gestão de eventos, como o Google Calendar, facilita muito o desenvolvimento de aplicações neste domínio.

## 1.2 Objectivos

Em primeiro lugar, deve ser feito um estudo do estado da arte associado ao conceito de *digital signage*, destacando-se os seguintes objectivos:

- Descrever o conceito de redes de *digital signage*, salientando os factores técnicos, estratégicos, comerciais e psicológicos que devem ser considerados na instalação desse tipo de redes;
- Realizar um levantamento das melhores práticas a ter na modelação dos locais onde os *displays* públicos se encontram, dos próprios *displays*

---

<sup>1</sup><http://www.ubisign.com>

e dos conteúdos a apresentar nestes, com base em vários estudos sobre o comportamento das pessoas em geral ou pequenos grupos de pessoas, face à passagem pelos *displays*, em ambientes exteriores e interiores;

- Fazer uma introdução sobre os problemas que a publicidade excessiva provoca no contexto dos painéis de *digital signage* e identificar algumas soluções;
- Dar exemplos de *displays* públicos inteligentes que tenham em conta o contexto envolvente e seleccionem os conteúdos a apresentar face a diversas situações, sem existir qualquer intervenção humana no escalonamento dos conteúdos.

De seguida, deve ser feita uma análise sobre o problema relacionado com a aplicação a desenvolver para o serviço Ubisign.com, obedecendo aos seguintes objectivos:

- Detalhar algumas ferramentas *on-line* de gestão de eventos, mas detalhar principalmente o Google Calendar.
- Descrever a aplicação a elaborar, dando a conhecer as principais funcionalidades que a aplicação irá suportar;
- Apresentar e analisar o Modelo do Domínio da aplicação a desenvolver;
- Fazer a análise de requisitos ou tarefas que a aplicação terá de cumprir, de forma a que disponibilize todas as funcionalidades essenciais para o seu bom funcionamento.

Quanto à forma como se planeia desenvolver a solução para o problema, isto é, a especificação e a discriminação das fases de desenvolvimento da aplicação *web* para apresentar informação sobre eventos, recorrendo à API Google Calendar, é importante referir os seguintes objectivos:

- Detalhar e desenvolver os requisitos propostos para a implementação da aplicação;
- Especificar a aplicação *web* a elaborar, ou seja, apresentar esboços para a sua interface, modelar a lógica de negócio e modelar a camada de acesso à base de dados;
- Desenvolver a aplicação *web* proposta pela Ubisign, capaz de produzir visualizações de informação sobre eventos pertencentes ao Google Calendar, a partir de configurações especificadas pelo utilizador, sendo que estas poderão ser armazenadas na base de dados para posterior utilização e gestão (actualização ou remoção);

- Descrever o processo de desenvolvimento da aplicação *web* proposta, identificar os problemas e decisões tomadas durante a fase de desenvolvimento e apresentar alguns exemplos simples de utilização.

Para finalizar, é importante provar que a aplicação já desenvolvida está preparada para ser utilizada em contextos reais, através da elaboração de alguns casos de estudo, destacando-se os seguintes objectivos:

- Definir alguns exemplos hipotéticos de utilização da aplicação, que poderão vir a ter utilidade efectiva, de modo a considerar as principais visualizações de informação sobre eventos (processadas pela aplicação). Não ser consideradas aplicações externas (hipotéticas) que utilizam a aplicação desenvolvida, sendo que, nesta fase, ainda não se pretende entrar no contexto do serviço Ubisign.com;
- Provar o bom funcionamento da aplicação num contexto real, ou seja, integrar a aplicação no serviço Ubisign.com, e apresentar alguns exemplos concretos de visualizações processadas pela aplicação, juntamente com outros conteúdos existentes no serviço.

### 1.3 Organização

No capítulo 2 vai ser analisado o estado da arte sobre *digital signage*, ou seja, vai haver um estudo de vários aspectos ligados a esse novo meio tecnológico, com o objectivo de enquadrar o leitor no tema e, assim, compreender melhor os capítulos subsequentes, sobre a aplicação proposta pela Ubisign.

Ao longo deste capítulo vai ser descrita a noção de redes de *digital signage*, serão realçados os principais factores inerentes a esse tipo de tecnologia, vai-se dar destaque às melhores práticas a ter no *design* de painéis de *digital signage*, vai-se fazer uma introdução sobre os problemas que a publicidade excessiva traz ao mundo dos *displays* públicos e, finalmente, dar-se-ão exemplos de *displays* inteligentes que têm em conta o contexto envolvente (através de sensores) e seleccionam os conteúdos a apresentar, consoante as situações presenciadas.

Considerando o capítulo 3, importa dizer que vai ser apresentado o problema relacionado com a aplicação a elaborar, permitindo, dessa forma, que o leitor conheça o projecto proposto pela Ubisign (aplicação *web* para apresentar eventos do Google Calendar) e todos os contornos inerentes ao problema em questão.

Ao longo deste capítulo vão ser detalhadas algumas ferramentas *on-line* de gestão de eventos (com principal foco no Google Calendar), será descrita a aplicação a elaborar, é apresentado e analisado o respectivo Modelo do Domínio e, por fim, vão ser declarados os requisitos a suportar pela aplicação.

Passando para o capítulo 4, é importante referir que é onde vai estar descrita a solução de *software* desenvolvida, com o objectivo de proporcionar ao leitor uma ideia clara sobre todas as decisões tomadas na especificação e implementação da aplicação *web*, de acordo com o que foi definido na análise de requisitos.

Durante este capítulo vão ser detalhados e desenvolvidos os requisitos propostos, será especificada a aplicação segundo a camada de interface, negócio e dados, bem como a descrição de todo o processo de desenvolvimento, juntamente com os problemas confrontados e, ainda, a apresentação de alguns exemplos simples de utilização da aplicação.

No capítulo 5 serão destacados exemplos de utilização da aplicação *web* elaborada, permitindo que o leitor tenha a noção das potencialidades efectivas da aplicação, para além de poder ver o resultado real da sua integração no serviço Uesign.com.

O capítulo 6 apresentará as conclusões sobre todo o trabalho desenvolvido, efectuará uma análise crítica dos objectivos já propostos, assim como ideias com vista à melhoria da qualidade e utilidade da aplicação desenvolvida.



## Capítulo 2

# Estado da Arte

Neste capítulo vai ser efectuado um estudo do estado da arte sobre *digital signage*, que consiste numa forma muito frequente de apresentar conteúdos publicitários em locais públicos e informação sobre eventos, substituindo os tradicionais *paper signs* (publicidade baseada em papel) [Müller, 2009]. As tecnologias utilizadas nos painéis de *digital signage* têm vindo a evoluir na qualidade (LCDs, *plasmas*, LEDs, ecrãs sensíveis ao toque, etc.) e, ao mesmo tempo, os preços têm vindo a descer, o que tem facilitado o seu uso generalizado.

Para que a informação seja facilmente captada pelo público alvo, é necessário que os conteúdos apresentados sejam bem escalonados, disponibilizados em quantidades moderadas e suficientemente visíveis, uma vez que, segundo estudos realizados [Huang et al., 2008], as pessoas em locais públicos olham muito brevemente para os *displays*.

Tendo em vista o controlo dos conteúdos a apresentar, pode-se recorrer a sistemas sofisticados, que dado o contexto inserido (local, tempo, etc.), utilizam estratégias de venda de espaços publicitários, à base da selecção criteriosa e automática dos conteúdos a apresentar a cada momento (*digital signage* inteligente [Müller, 2008]) e, como consequência, aumenta-se o grau de interesse das pessoas que passam pelos *displays* (ou painéis digitais).

É de notar que os fundamentos sobre *digital signage*, estudados neste capítulo, vão servir como introdução ao tema da dissertação, possibilitando uma melhor compreensão dos capítulos subsequentes, sobre o desenvolvimento de uma aplicação *web* para painéis de *digital signage*, proposta pela empresa Ubisign.

### 2.1 Introdução

Na secção seguinte, vai-se dar destaque às redes de painéis digitais (*Digital Signage Networks* – DSNs) [Bügg, 2005], tratando-se de um termo genérico dado a um sistema de painéis digitais que recebe conteúdos de um sistema

de controlo centralizado que permite controlar conteúdos multimédia de uma forma dinâmica e flexível. São uma nova forma de expor informação em relação a outros meios, como a televisão, a rádio ou a Internet e, por isso, há factores técnicos, estratégicos, comerciais e psicológicos, que devem ser considerados. Essencialmente, é de realçar o último factor mencionado, pois quando se lida com pessoas, o importante é que gostem dos conteúdos apresentados, para que a informação seja absorvida e não seja ignorada. Tendo em conta os factores mencionados anteriormente, vai ser proposta uma *framework* (teórica).

De seguida, vai-se dar destaque a três *surveys* sobre as melhores práticas a ter no *design* dos locais públicos onde os painéis digitais são instalados e no *design* do conteúdo apresentado nos próprios painéis.

O primeiro *survey* analisado consiste num estudo aprofundado de 46 painéis de grande dimensão, localizados em várias cidades da Europa Ocidental [Huang et al., 2008], sendo que o objectivo principal deste trabalho é o de oferecer aos investigadores e *designers*, conhecimento ecologicamente válido sobre as melhores práticas na instalação de *displays* de grande dimensão, em ambientes públicos.

O *survey* seguinte, ao contrário do anterior, enfatiza painéis públicos localizados em ambientes interiores, mais especificamente num complexo universitário. Para obter opiniões por parte dos utilizadores, foi feito um questionário sobre a qualidade e sobre a forma como os conteúdos são apresentados pelos painéis, para além de outro tipo de perguntas [Rashid and Quigley, 2009].

O último *survey* considera painéis destinados a pequenos grupos co-localizados (painéis semi-públicos) [Huang and Mynatt, 2003], em que o objectivo principal é apresentar informação que auxilie na interacção e colaboração entre membros de um grupo que trabalham num laboratório. Para tal, foram desenvolvidas quatro aplicações informáticas para um *display* e, depois da instalação das mesmas, foi feita uma recolha de opiniões, com vista a melhorar os aspectos menos positivos identificados.

Na secção 4 vai-se introduzir as externalidades negativas existentes na publicidade em *digital signage* [Müller and Krüger, 2007], onde se realça o facto de haver sítios em que não há uma regulação criteriosa da publicidade e, por isso, há uma compra agressiva dos espaços publicitários. Assim, sempre que um agente compra um espaço publicitário, o orçamento investido não tem em conta a atenção que as pessoas prestam a essa publicidade. Para resolver este problema são propostos métodos como os *maximum permissible values, fees* e *tradable certificates* (baseado em leilões), que definem métricas para quantificar a atenção consumida pelas pessoas a observar a publicidade.

Na última secção vai-se considerar dois exemplos de *displays* públicos inteligentes, o *BluScreen* [Payne et al., 2006] e o *iDisplay* [Müller, 2007], que apresentam anúncios publicitários, com base no contexto envolvente, recorrendo a um sistema baseado em leilões para venda de espaços publicitários.

O *BluScreen* utiliza um mecanismo de selecção de conteúdos publicitários (baseado em leilões de segundo-preço [Vickrey, 1961]) que, juntamente com o histórico de exposição dos utilizadores em certos anúncios publicitários e com a informação sobre que utilizadores actualmente estão a ver o *display*, determina qual é o anúncio que tem direito a ser apresentado no próximo ciclo publicitário. Regra geral, o anúncio apresentado é aquele que foi menos visto pelos utilizadores que estão a observar o *BluScreen*, não havendo uma desnecessária repetição de conteúdos publicitários para aqueles que já os viram. Para o *BluScreen* saber que pessoas estão em frente ao *display*, elas têm que utilizar dispositivos com Bluetooth e, por isso, são utilizados sensores para detectá-los.

Por fim, o *iDisplay* é um *display* público inteligente que, tal como o *BluScreen*, se baseia em leilões de segundo-preço para seleccionar os anúncios publicitários mais apropriados, segundo o contexto envolvente. O que distingue estes dois *displays*, é o facto de o *iDisplay* considerar apenas anúncios que têm potencial para desencadear uma acção (por exemplo, ir às compras), por parte das pessoas que observam o *display*, enquanto que o *BluScreen* é mais generalista. Outro aspecto a considerar sobre o *iDisplay*, é o facto deste incluir vários *slots* publicitários no ecrã do *display*, enquanto que o *BluScreen* apresenta um anúncio de cada vez.

## 2.2 Redes de Digital Signage – DSNs

Uma rede de painéis digitais (Digital Signage Network – DSN) é um termo genérico dado a qualquer sistema de painéis digitais que recebe conteúdos de um sistema de controlo centralizado [Bügg, 2005]. Esse sistema é capaz de controlar o conteúdo de uma forma dinâmica e flexível. Tais redes de painéis digitais têm vindo a alargar-se e a substituir os tradicionais mecanismos de publicidade (*paper signage*, rádio, televisão, etc.) [Müller, 2009], muito devido ao constante avanço tecnológico e conseqüente descida dos preços deste tipo de *hardware* (plasmas, LCDs, LEDs, etc.). É de notar que em paralelo ao crescimento deste tipo de tecnologia, as empresas fornecedoras de DSNs têm crescido tanto em termos de tamanho como em número, e tendem a continuar a crescer a curto/médio prazo.

Algumas companhias que fornecem serviços nestas áreas, não têm o conhecimento necessário sobre a melhor forma de expor informação em *displays* públicos e têm a tendência de apresentar o mesmo tipo de conteúdos característicos dos meios tradicionais, como a televisão, a Internet, etc.. Por isso, o público alvo tende, muitas vezes, em não dar a devida atenção aos painéis informativos, visto que, por exemplo, o tipo de conteúdo e a sua quantidade não se enquadra tão bem no contexto. Assim, é necessário ter em consideração vários factores (técnicos, estratégicos, comerciais e psicológicos) para haver uma adaptação das DSNs ao contexto onde estão inseridos ou o in-



verso, ou seja, moldar o contexto de forma a que este seja o mais adequado possível para os *displays* e para o tipo de conteúdos neles apresentados.

### 2.2.1 DSNs: Factores Técnicos, Estratégicos e Comerciais

Com as DSNs a serem cada vez mais utilizadas a nível mundial (é um mercado que já excede mil milhões de dólares e tem tido um crescimento de 30% por ano [Bügg, 2005]), é bastante importante ter em consideração aspectos característicos destas e de nunca esquecer que se trata de um tipo de tecnologia ubíqua [Weiser, 1993].

Antes de instalar uma DSN, deve-se ter em conta os factores técnicos, uma vez que estes podem implicar mudanças na concepção e na implementação dos sistemas instalados nos painéis e, também, no tipo de *hardware* utilizado. Também deve haver uma preocupação efectiva com os restantes factores, tanto nos que têm a ver com os objectivos comerciais e estratégicos, como os que têm a ver com as ciências humanas, isto é, os factores psicológicos (que serão abordados mais adiante).

#### Avanços Técnicos

O evolução tecnológica do *hardware* dos *displays* tem permitido uma diminuição dos preços e, ao mesmo tempo, a qualidade e brilho dos ecrãs tem evoluído consideravelmente, tornando as DSNs cada vez são mais acessíveis e atractivas, em termos de custos e qualidade, para muitas grandes e médias empresas.

O custo das redes, ao nível do *hardware*, também tem evoluído muito nestes últimos anos, sendo que agora as redes com fios e sem fios são soluções óptimas em termos de preço e na qualidade de transmissão de dados. Além disso, o capital investido nestas redes acaba por ser muito compensador, uma vez que é bastante menor do que se fossem introduzidos os conteúdos, em cada *display* da rede, de forma manual.

Outro aspecto muito importante para tornar as DSNs uma realidade viável, tem sido a evolução constante ao nível do *software* de controlo. Assim, este tipo de *software* tem como objectivo controlar e customizar os conteúdos dentro do local onde se encontra a DSN (*in-house*) ou de forma remota (em qualquer local do mundo) por equipas especializadas.

Também existem outros avanços importantes a nível de *software* que possibilitam a uma DSN recolher dados de fontes externas, como por exemplo: monitorizadores de temperatura, *feeds* de vídeo, bases de dados proprietárias, etc..

No caso concreto da publicidade, existem *softwares* de gestão de publicidade e sistemas de detecção que ajudam os agentes publicitários a monitorizar, customizar e cobrar pelos anúncios publicitários.

## Tipos de DSNs

Existem várias características que variam nas DSNs, apesar de, à primeira vista, parecerem todas idênticas, estas podem ser categorizadas segundo várias dimensões, tais como: *hardware*, *software*, conteúdos e local.

No que diz respeito ao *hardware*, pode-se encontrar painéis de diversas tecnologias, tais como: LCDs, plasmas, LEDs, OLEDs, etc.. Tal como os computadores, os painéis podem ter conectividade com ou sem fios, e quanto ao seu posicionamento, estes podem ser instalados nas paredes, no tecto ou até mesmo em suportes próprios.

Ao nível do *software*, pode-se considerar que o sistema de gestão de conteúdos é local, quando este apenas pode ser utilizado no sítio onde se encontra a DSN, e considera-se *hosted*, quando há a possibilidade de efectuar o controlo dos conteúdos em qualquer local do mundo (normalmente através de *websites*).

Também é de relevar que quando o sistema de gestão de conteúdos é *hosted*, muitas vezes é fornecido por entidades externas, sendo um caso típico de utilização de *software* como um serviço (SaaS – *Software as a Service* [Gold et al., 2004]).

Ao nível dos conteúdos, estes podem estar todos contidos nos próprios painéis (*inhouse*), como podem ser produzidos em sistemas externos aos painéis (*outsourced*), assim como podem estar contidos tanto nos painéis como nos sistemas externos a este (*mixed*).

Por fim, quanto ao local onde as DSNs são instaladas, podemos reparar que normalmente estas redes localizam-se em locais de entretenimento (por ex.: restaurantes, cinemas, estádios, auditórios, etc.), de *shopping* (por ex.: centros comerciais, avenidas, etc.) ou de retalho (por ex.: pequenos comércio, super-mercados, etc.).

## Típica solução de DSN

Para ilustrar o que é de facto uma DSN, pode-se observar a arquitectura proposta na Figura 2.1<sup>1</sup>.

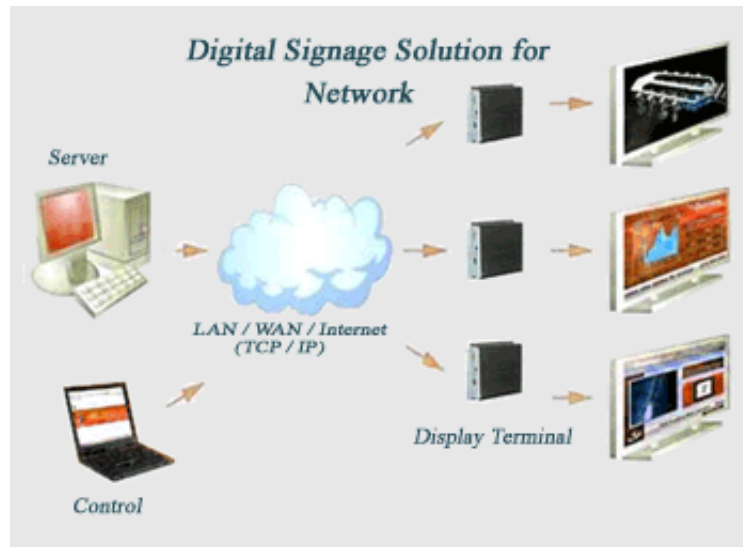
Como já foi dito no tópico anterior, existem várias dimensões a considerar na instalação de uma DSN. Neste caso particular, vão ser consideradas as seguintes: *hardware*, *software* e conteúdos.

Tendo em conta o *hardware*, pode-se considerar os seguintes dispositivos da Figura 2.1: servidores, *players*, *displays* e os dispositivos que compõem a rede distribuída.

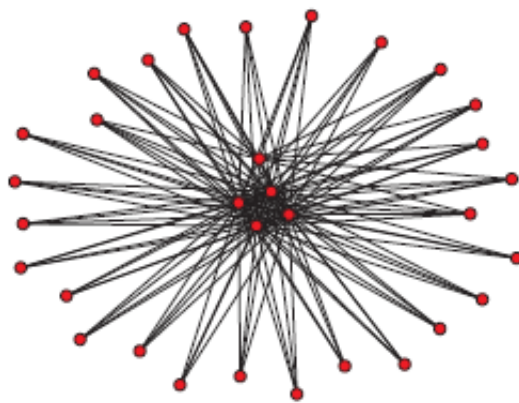
Quanto aos servidores (observar ícone representativo na Figura 2.1, designado por *Server*), destaca-se o papel de armazenar conteúdos multimédia, de permitir a gestão destes e, ainda, a possibilidade de enviá-los para os *players* a partir da rede distribuída.

---

<sup>1</sup>Este tópico é baseado em: [http://ettss.com/network\\_solution.php](http://ettss.com/network_solution.php)



**Figura 2.1:** Exemplo de arquitetura para uma DSN (Figura retirada de [http://ettss.com/network\\_solution.php](http://ettss.com/network_solution.php))



**Figura 2.2:** Figura que ilustra uma topologia em estrela [Guimerà et al., 2002]

Considerando os *players*, é importante realçar que são tipicamente computadores com a função de armazenar e entregar conteúdos multimédia aos *displays*, segundo um escalonamento temporal definido à priori.

Como já foi dito, os *displays* podem assumir várias tecnologias (LCD, plasma, LED, etc.) e têm a óbvia função de apresentar os conteúdos multimédia, entregues pelo respectivo *player*.

O principal meio que torna viável qualquer que seja a solução de DSN, é a existência de uma rede distribuída que é composta por uma infraestrutura que permite que os dispositivos existentes nela, troquem informação entre si. Neste caso, o papel fundamental da rede distribuída, é o de permitir que os servidores centrais enviem os conteúdos até ao alvos definidos, os *players*. Como se sabe, uma rede distribuída pode recorrer a uma ou mais tecnologias (consoante as necessidades ou possibilidades técnicas ou económicas), tais como: satélite, Internet, LAN, WAN, etc..

Na Figura 2.2, pode-se observar a topologia de rede em estrela, que é a considerada na típica solução de DSN, em que os servidores que armazenam conteúdos e permitem a sua gestão, estão localizados no centro da estrela e, assim, são acedidos pelos *players* que se encontram nos vértices da estrela e que solicitam conteúdos aos servidores centrais, com o objectivo de serem apresentados nos *displays* respectivos.

Passando para a dimensão sobre o *software*, é importante destacar o sistema de gestão de conteúdos, que tem um papel crucial numa DSN. Este *software* especializado tem a função de permitir o escalonamento no envio de conteúdos para os *displays* e também serve para definir a ordem temporal na qual os conteúdos são apresentados. Ainda existem outras funcionalidades importantes de um sistema de gestão de conteúdos, tais como: a monitorização da performance dos *players* e dos eventos escalonados (incluindo *logs*), a possibilidade de definir e escolher *layouts* para especificar os conteúdos que são apresentados em cada região dos ecrãs dos *displays*, a possibilidade de acrescentar *feeds* informativos, etc..

Quanto aos conteúdos, estes podem ser apresentados no formato vídeo, áudio e/ou imagem e podem assumir-se como conteúdos publicitários, eventos, notícias, informação sobre preços de produtos, etc..

### Características únicas das DSNs

Ao contrário da televisão, em que os programadores dos canais apenas têm conhecimento da variável tempo, a programação dos conteúdos de uma DSN pode ser feita tendo em conta vários factores, tais como: o espaço, o tempo, os eventos (fusão entre o espaço e o tempo), a audiência, os conteúdos dinâmicos, etc..

- **O Espaço:** Tendo em conta que o sistema central “sabe” onde cada painel se encontra, pode haver uma selecção mais apropriada de con-

teúdos (ou canais) para apresentar em cada painel, em função do local onde se encontra. Por exemplo, se um *display* está localizado num super-mercado, perto de um determinado produto, é natural que este apresente conteúdos relacionados com esse produto (preço, descontos, características do produto, etc.).

- **O Tempo:** Para um dado dia da semana, cada *display* pode fornecer conteúdos diferentes em função da hora do dia. Por exemplo, num aeroporto, durante a semana de trabalho, pode existir uma DSN que forneça informação sobre oportunidades de negócio em vários locais do mundo, enquanto que nos fins-de-semana, pode publicitar lugares turísticos e propícios ao convívio familiar ou social.
- **Os Eventos:** Sabendo os eventos que vão ocorrer, cada *display* pode divulgá-los e apresentar informação sobre eles.
- **A Audiência:** Sabendo a data actual e o espaço, a informação demográfica da audiência pode ser bem especificada.
- **Os Conteúdos Dinâmicos:** Outra característica muito importante das DSNs é poderem fornecer conteúdos dinâmicos, tornando a constante mudança de conteúdos bastante barata quando comparada à publicidade tradicional baseada em papel. Permite ainda personalizar os conteúdos rapidamente e em tempo real, recorrer a animações para cativar o público e, em casos particulares, permite a interacção com a audiência (por exemplo, os *displays* sensíveis ao toque).

### Fragmentação da Audiência

As DSNs, juntamente com a Internet, têm vindo a substituir os tradicionais meios de comunicação (rádio e televisão) [Müller, 2009], devido ao facto destes serem mais eficientes no que diz respeito à capacidade de enviar mensagens a segmentos específicos da população (jovens, idosos, desportistas, etc.). Mas é de salientar que isto acontece, porque actualmente as pessoas têm interesses muito variados, havendo uma grande fragmentação da audiência, para além da existência de diferentes culturas.

#### 2.2.2 DSNs: Factores Psicológicos

Para além dos factores técnicos, estratégicos e comerciais, relacionados com as DSNs, descritos na subsecção anterior, os factores psicológicos são os mais importantes a ter em conta. Por exemplo, a colocação de um *display* grande fixado à parede ou a colocação de um *display* mais pequeno apoiado numa base, pode ter efeitos psicológicos diferentes nas pessoas (o *display* grande dá a sensação de autoridade, imponência ou de riqueza, enquanto que o *display* pequeno pode fazer com que as pessoas se aproximem mais do ecrã). Outras

considerações psicológicas, como a interface com o utilizador e a personalização, são relevantes no contexto do *software* escolhido para controlar a DSN.

No entanto, a informação disponibilizada no ecrã dos *displays* é o factor que tem mais impacto psicológico nas pessoas. Assim, durante o processo de modelação dos conteúdos a apresentar numa dada DSN, devem ser considerados muitos aspectos da psicologia humana. Estes têm importância na decisão do conteúdo a ser apresentado (as mensagens intelectuais e emocionais a serem enviadas) e na decisão da forma como o conteúdo é apresentado (as cores [Birren, 1961], os tamanhos e os movimentos usados).

### A percepção

Quando há informação transmitida num ecrã de um painel de *digital signage*, o factor mais relevante a ter em conta é a percepção, isto porque se a informação não pode ser vista nem ouvida, as pessoas acabam por ignorar o *display*.

Apesar da percepção da audição ser importante em alguns contextos de DSNs, a percepção visual é a que costuma ter mais peso psicológico e, nesse sentido, deve-se dar especial ênfase aos aspectos visuais, tais como: a cor, a forma e o movimento.

Sobre a percepção da cor, é de referir que convém utilizar cores que contrastem bem (por ex.: letras de cor escura num fundo branco) e não apenas cores escuras ou cores claras (por ex.: letras azuis num fundo preto ou letras amarelas num fundo branco), isto para que o olho humano capte de forma eficiente a mensagem transmitida pelo *display*.

Quanto à percepção da forma, a informação transmitida por um painel digital consiste frequentemente em figuras geométricas ou textos. Convém que se utilizem as mais adequadas, para que haja o impacto psicológico desejado, por exemplo, as letras grandes transmitem um sentimento amigável ou de diversão, enquanto que as letras arredondadas transmitem um sentimento de sensualidade ou efervescência.

Por fim, sobre a percepção do movimento, pode-se dizer que é uma das mais básicas características do nosso sistema visual e, por isso, nenhum de nós falha nessa percepção. De forma intuitiva, parece que um alvo em movimento serve para captar a atenção, no entanto, um estudo relativamente recente [Abrams and Christ, 2003] concluiu que o movimento, por si só, não capta a atenção (esse estudo também concluiu que os objectos estáticos e os que estão em movimento, têm o mesmo índice de atracção). O que promove a atenção das pessoas são os objectos que estejam a iniciar um movimento.

Considerando esse estudo, quando numa DSN é aplicado o conceito de movimento, deve-se privilegiar o movimento de objectos que inicialmente estavam parados, para que se chame a atenção do público. No entanto, é importante que não existam demasiados objectos a mover-se simultaneamente,

uma vez que pode cansar a audiência e, conseqüentemente, há uma natural despromoção da atenção (factor que vai ser abordado de seguida).

### A atenção

Para além do movimento, existem outras formas de cativar a atenção, tais como: a mudança de brilho ou iluminação, mudanças bruscas de cor, caras emocionais ou relevantes, fenómenos únicos (por ex., um objecto amarelo num sítio cheio de objectos azuis), etc..

Um estudo recente [Franconeri et al., 2005] concluiu que o facto de aparecer um novo objecto não é suficiente para captar a atenção das pessoas. Outros estudos [Kristjansson et al., 2001, Nakayama and Mackeben, 1989] mostram que as imagens intermitentes ou oscilantes não são muito bem sucedidas na promoção da atenção por períodos longos, tornando-as facilmente ignoradas pelas pessoas.

A atenção e a percepção do tempo estão bastante ligadas [Tse et al., 2004], por exemplo, se um evento, que passa no ecrã de um *display*, prende a atenção de alguém, então esse evento dá a sensação de durar mais tempo. Ao mesmo tempo que o evento visual parece “mover-se devagar”, o resto do ambiente envolvente parece “mover-se depressa”.

### A memória

Como já foi dito, uma DSN é usada para publicitar, promover uma marca, criar bom ambiente, construir uma relação de lealdade ou apenas tem a função de informar. Para que essa informação seja memorizada pelas pessoas, é necessário que esta seja: relevante, emocional, variada, credível, repetida, cativante, oportuna, multi-sensorial, etc..

Algumas memórias são retidas num formato de imagem (memórias episódicas) e outras são retidas num formato mais factual (memórias semânticas). Assim sendo, se se pretende promover memórias episódicas, a apresentação de informação visual é a mais adequada. Por outro lado, se se pretende dar relevo às memórias semânticas, a informação verbal é a mais apropriada. Por exemplo, se uma DSN promove um pacote de férias, é mais eficaz recorrer a imagens (memória episódica), enquanto que se uma DSN existe com o objectivo de enviar uma mensagem pública, nesse caso recorre-se a memória semântica.

Para além das memórias episódicas e semânticas, também se deve ter em conta as memórias de curto e longo prazo. Tal como o nome indica, as memórias de longo prazo persistem por muito mais tempo que as de curto prazo (exemplos de memórias de longo prazo: memórias de criança, coisas que uma pessoa deseja, as marcas favoritas, etc.), sendo as de longo prazo as mais importantes a ter em conta na apresentação de conteúdos nas DSNs (por exemplo, usar um *slogan* que caracterize de uma forma genuína uma

certa marca, realçar aspectos marcantes de uma marca, etc.).

### 2.2.3 DSNs: Framework Psico-Estratégica

Como é sabido, as DSNs são um novo meio de comunicação e, contrariamente aos *paper signs*, apresentam informação dinâmica e, em relação à Internet, são fisicamente distintas. Graças à sua facilidade de integrar dados (através de sensores, bases de dados, relógios, etc.) são capazes de servir, de forma inteligente e dinâmica, diversos conteúdos multimédia. Deste modo, a sua capacidade para usar o conhecimento do local, tempo e outras variáveis, permite-lhes apresentar conteúdos de uma forma flexível e com um alvo bem definido.

Sabendo que as estratégias e os modelos tradicionais de publicidade, informação e entretenimento não se enquadram no contexto das DSNs, deve-se desenvolver uma nova *framework* teórica que tenha em conta os factores psicológicos, sociais, culturais e de negócio mais relevantes para as DSNs. Antes de se criar uma DSN, deve-se ter em conta os seus objectivos, o ambiente onde está inserida, as pessoas e as actividades que têm lugar nesse ambiente.

#### Objectivos das DSNs

Um dos principais objectivos das DSNs é o de publicitar produtos/serviços existentes no próprio local e num determinado tempo (por ex., produtos que se encontram em *stock* e que estão prontos a ser vendidos), produtos/serviços disponíveis no futuro, ou produtos/serviços disponíveis noutra local (por ex., a abertura de uma nova loja). Poderão ser considerados, ainda, outros objectivos, tais como:

- reforçar, criar ou alterar uma marca;
- criar um bom ambiente psicológico nas pessoas (havendo uma boa atmosfera/clima no local);
- mudar o comportamento de compra das pessoas;
- induzir a vontade de experimentar o produto;
- entreter e informar as pessoas.

É de realçar que o objectivo de entreter e informar as pessoas é muitas vezes misturado com a publicidade (tal como acontece muitas vezes com a televisão e rádio), fazendo com que as pessoas prestem mais atenção e captem melhor a mensagem transmitida. Por isso, uma boa estratégia é usar este formato de publicidade em locais de longa espera, para que as pessoas fiquem com a sensação de que o tempo passa mais depressa.



### Actividades e Mentalidades nas Compras

Não raras vezes, as pessoas vão às compras para adquirir bens essenciais, explorar novas tendências, socializar ou divertir-se. Uma dimensão relevante na compra de produtos, é a mentalidade do comprador, pois um bom conhecimento da sua mentalidade é susceptível de ajudar a modelar uma estratégia adequada para uma DSN, para além de ajudar na escolha dos conteúdos a apresentar. Outro aspecto importante é auxiliar o processo de compras, ou seja, fornecer informação útil acerca do local onde se situam certos mercados ou certos produtos (com particular interesse) num super-mercado, fornecer dados sobre determinadas características de produtos, etc..

### As Pessoas

Um dos aspectos mais críticos de qualquer estratégia, aplicada a uma DSN, é o conhecimento profundo sobre pessoas que se deparam perante painéis digitais. Um estudo das suas personalidades, objectivos e pensamentos deve proporcionar todo o tipo de variáveis a ter em conta numa estratégia de uma DSN. Assim, uma abordagem útil para estudar o comportamento das pessoas é ter em conta as seguintes variáveis: os medos, as incertezas, as dúvidas, os sonhos e as aspirações. Como ponto de partida, convém fazer vários inquéritos de informações básicas (idade, salário, estilo de compras que efectua, etc.) a todos os segmentos da população (idosos, jovens, homens, mulheres, etc.) e a partir desses inquéritos extrair toda a informação importante, para que se obtenham todas as variáveis referidas anteriormente. No fim deste processo, há uma maior preparação para criar o conteúdo mais adequado a introduzir numa DSN.

### O Local

Estudar o local/espço onde se pretende colocar uma DSN, é importante pelos seguintes motivos:

- dá a informação sobre qual o *hardware/software* a instalar numa DSN (parte inicial da estratégia de DSN);
- permite conhecer o tipo de pessoas e personalidades existentes no espaço;
- permite perceber o efeito que o local provoca nas pessoas (do ponto de vista psicológico);
- o local, muitas vezes, evita que a DSN transmita o sentimento desejado às pessoas (por exemplo, criar um ambiente calmo num estádio de futebol não é uma solução viável).

## 2.3 *Surveys* sobre o *design* em painéis digitais

Existem vários trabalhos no ramo do *design* em painéis digitais, tanto em locais públicos, como por exemplo, estações de comboio e cafés [Carter et al., 2004, Churchill et al., 2003], como em locais privados [Czerwinski et al., 2003, MacIntyre et al., 2001], e até em locais semi-públicos, como por exemplo, salas de aula ou outros locais trabalho [Fass et al., 2002, Huang and Mynatt, 2003, McCarthy et al., 2001, Salber et al., 1999b, Terrell and McCrickard, 2006].

Nesta secção, vai-se dar destaque a três *surveys* sobre as melhores práticas a ter no *design* dos locais públicos onde os painéis digitais são instalados e no *design* do conteúdo apresentado nos próprios painéis. O primeiro *survey* consiste num estudo aprofundado de 46 painéis de grande dimensão, localizados em várias cidades da Europa Ocidental [Huang et al., 2008]. O *survey* seguinte, ao contrário do anterior, enfatiza painéis públicos localizados em ambientes interiores, mais especificamente num complexo universitário. Para obter *feedback* por parte dos utilizadores, foi feito um questionário sobre a qualidade e forma dos conteúdos apresentados pelos painéis, para além de outras perguntas mais concretas [Rashid and Quigley, 2009]. Finalmente, o último *survey* é mais orientado para painéis destinados a pequenos grupos co-localizados (painéis semi-públicos) [Huang and Mynatt, 2003].

### 2.3.1 *Survey* sobre *digital signage* em ambientes exteriores

Este *survey* consiste num estudo aprofundado de 46 *displays* digitais de grandes dimensões localizados em vários sítios da Europa Central, a apresentar informação “calma”<sup>2</sup> (não perturbadora para as pessoas, ao contrário da publicidade excessiva). O objectivo deste trabalho foi oferecer aos investigadores e *designers* conhecimento ecologicamente válido sobre as melhores práticas no uso de grandes painéis digitais em ambientes públicos [Huang et al., 2008]. É dada ênfase a *displays* com a função de fornecer conteúdos ambientais e de cariz não urgente, tais como: conteúdos informativos, publicitários (moderados), artísticos, ou de entretenimento.

### Resultados e Recomendações ao nível do *design*

No fim deste estudo, obtiveram-se os seguintes resultados e recomendações nos painéis de grandes dimensões, ao nível do *design*:

- *Brevidade nos olhares para os displays*: Geralmente as pessoas olham muito brevemente para os *displays* (1 ou 2 segundos). Para além disso, as pessoas raramente abrandam ou param para prestar atenção ao conteúdo dos painéis. Apesar de tudo, os *displays* que apresentam

---

<sup>2</sup><http://www.ubiq.com/hypertext/weiser/calmtech/calmtech.htm>

vídeos, tendem a captar uma maior atenção das pessoas, mas raramente fazem com que parem, uma vez que na rua, geralmente, há pressa de ir a algum sítio. No entanto, com base em estudos anteriores, concluiu-se que para os olhares com uma duração superior a 800 milissegundos, há intencionalidade da pessoa no conteúdo apresentado [Müller and Rabbitt, 1989], o que do ponto de vista do factor atenção é óptimo. *Recomendação*: evitar o uso excessivo de texto nos ecrãs dos painéis, uma vez que as pessoas não perdem muito tempo para ler os conteúdos. Utilizar imagens ou vídeos para não haver demasiado texto.

- *Posicionamento dos displays*: Geralmente os *displays* são colocados ao nível dos olhos ou posicionados consideravelmente acima da cabeça. Verifica-se que os primeiros são mais eficazes em atrair as pessoas, independentemente do tipo de conteúdo apresentado, enquanto que os últimos raramente atraem a atenção.
- *Tipos de conteúdo*: Os tipos de conteúdo observados são variados, de entre os quais se destacam: conteúdos educativos, artísticos, publicitários (o mais abundante), curiosidades, notícias e eventos. Em geral, as observações feitas não permitem saber qual o conteúdo de maior interesse para as pessoas.
- *Formato dos conteúdos*: Há maior interesse em conteúdos em formato vídeo do que em formato texto com animações ou imagem. Os conteúdos estáticos são mais adequados em formato papel do que em painéis de *digital signage*. A informação do estilo *screensaver*, cuja ordem temporal dos conteúdos é previamente imposta e que não permite navegação (*browsing*), não é de particular interesse por parte da audiência. *Recomendação*: produzir, continuamente, conteúdos dinâmicos e permitir, por parte do utilizador, maior controlo sobre os conteúdos apresentados.
- *Captar a atenção*: Geralmente as pessoas olham para os *displays* se o ambiente envolvente é atractivo. Por isso, os painéis de grande dimensão têm um papel secundário na captação da atenção quando existem outros objectos nas suas vizinhanças. *Recomendação*: tirar partido do material existente no ambiente para atrair a atenção do público e não assumir que um *display* de grandes dimensões é o factor de atracção número um.
- *Cativar a audiência*: Em algumas situações o público é mantido “preso” em frente a um *display* (por exemplo, *displays* localizados nos terminais de escadas rolantes). Apesar deste facto, um *display*, por si só, não garante uma maior duração da atenção prestada pelo público (por exemplo, nas lojas onde há filas destinadas ao pagamento, as pessoas, apesar de estarem a maior parte do tempo paradas, não prestam muita



**Figura 2.3:** As pessoas param e sentam-se à frente de pequenos displays, mesmo que na vizinhança existam grandes displays [Huang et al., 2008]

atenção aos *displays*, uma vez que, geralmente, o material comercial envolvente é mais atractivo). Outros trabalhos de investigação, como em [Agamanolis, 2003], apelam ao uso de imagens das pessoas que olham para os *displays*, com o objectivo de cativá-las e, assim, ficarem mais tempo em frente aos ecrãs.

- *Painéis de pequena vs. grande dimensão:* De uma forma geral, as pessoas ficam mais tempo em frente a *displays* com um ecrã de pequena dimensão, possivelmente porque têm maior privacidade e é um ambiente mais íntimo (ver exemplo da Figura 2.3). No entanto, isso não quer dizer que os painéis de pequena dimensão difiram muito em termos de atracção, em relação aos de grande dimensão. *Recomendação:* utilizar *displays* de diferentes dimensões de ecrã, para balancear os sentimentos de privacidade e de exposição ao público.

## Conclusões

A melhor forma de promover a atenção da audiência, em relação aos *displays*, é através do acoplamento do aspecto ambiental que rodeia o *display* e o processo de *design*. Assim, o contexto que envolve um *display* deve ser considerado durante a fase de *design* e não depois, uma vez que o ambiente tem um grande impacto na atenção das pessoas.

Neste *survey* faltou efectuar um estudo sobre as experiências que as pessoas têm com os *displays* e, consoante as opiniões dadas, haver uma adapta-

ção dos mesmos de forma a que sejam mais atractivos. No entanto, no caso específico da publicitação de marcas, os estudos no mercado mostram que breves olhares para os *displays* são suficientes para incrementar o conhecimento das marcas<sup>3</sup>.

### 2.3.2 *Survey sobre digital signage num ambiente académico*

Este trabalho investiga o uso de *displays* digitais (num complexo universitário localizado em Dublin, na Irlanda) que apresentam, ciclicamente, páginas *web* sobre perfis de membros do complexo e conteúdos sobre investigação científica. Para tal, foi feito um *survey*, à base de um questionário, para explorar a forma como os membros do complexo utilizam os *displays* localizados no edifício da universidade [Rashid and Quigley, 2009].

É importante sublinhar que os resultados obtidos são menos precisos, quando se efectuam questionários para que os participantes se lembrem das experiências que tiveram com os *displays* do complexo, ao contrário do que acontece quando se efectuam observações no contexto (método adoptado no anterior *survey* [Huang et al., 2008]).

#### Resultados com base nas experiências dos participantes

Quanto a experiências vividas pelos participantes, com os *displays*, obtiveram-se os seguintes resultados:

- Os conteúdos mais memorizados foram fotos e perfis de membros do complexo;
- O conteúdo mais útil foi a informação sobre os perfis de membros do complexo;
- O melhor local para observar os *displays* foi a cantina, porque é um local onde as pessoas estão sentadas e têm mais tempo para prestar atenção aos ecrãs;
- Os participantes foram da opinião de que os *displays* serviram para tornar o ambiente visualmente mais apelativo e para melhorar o conhecimento dos membros do complexo universitário.

#### Resultados: Expectativas e Recomendações

Como é natural, na parte final do questionário, os participantes foram interrogados para dizerem quais as mudanças que, a seu ver, poderiam ser feitas para melhorar a experiência proporcionada pelos painéis de *digital signage*, localizados no complexo, tendo-se destacado as seguintes sugestões:

---

<sup>3</sup>[http://www.economist.com/node/10431119?story\\_id=10431119](http://www.economist.com/node/10431119?story_id=10431119)

- *Tipos de conteúdo*: Quanto aos tipos de conteúdo sugeridos, os participantes consideraram que os painéis deveriam mostrar mais informação sobre eventos e notícias, diminuindo a informação sobre perfis de membros do complexo. Mesmo as notícias disponibilizadas (sob a forma de um rodapé rolante – Scrolling News Ticker<sup>4</sup>) eram bastante estáticas no sentido em que eram actualizadas muito raramente. Outro problema mencionado foi o facto das páginas *web* não estarem adaptadas para painéis públicos, uma vez que não faz sentido que exista uma secção de “Links” num painel não interactivo. *Recomendação*: disponibilizar conteúdos dinâmicos e constantemente actualizados (como por exemplo, *web feeds*).
- *Contextualização do conteúdo*: O conteúdo apresentado pelo *display*, localizado na recepção do complexo, deve ser mais geral, tornando-o mais apelativo para os visitantes. *Recomendação*: adaptar o conteúdo em função do contexto onde o *display* está inserido.
- *Posicionamento dos displays*: Verificou-se que a maioria dos *displays* estavam mal localizados, uma vez que três dos cinco *displays* eram pouco vistos. Um dos participantes do questionário sugeriu que seria melhor localizar os *displays* ao pé de sítios onde as pessoas normalmente esperam, como por exemplo: perto de impressoras, elevadores, filas, etc.. *Recomendação*: Antes de posicionar os *displays*, identificar o fluxo de movimento das pessoas e os locais de congestão existentes dentro de um edifício.
- *Tornar os displays mais interactivos*: A maioria dos participantes não gostou da proposta de tornar os *displays* mais interactivos e com a possibilidade de fazer *upload* de conteúdos à sua escolha. *Recomendação*: controlo moderado no *upload* de conteúdos.
- *Controlo de presença*: Foi feita uma proposta aos participantes com o intuito de permitir o acesso a informação de controlo de presença, para saber que pessoas estão ou não dentro do edifício universitário (tipo de informação inspirada em aplicações como o In/Out Board [Salber et al., 1999a] e o Active Portrait [Huang and Mynatt, 2003]). No entanto não acharam a ideia boa, uma vez que esse tipo de informação é bastante privada e, apenas, deve ser partilhada com colegas ou amigos.

### Conclusões e trabalhos futuros

Foram formulados alguns princípios, com o objectivo de maximizar a utilização de *displays* em ambientes públicos fechados (neste caso trata-se de um

---

<sup>4</sup>Scrolling News Ticker em JavaScript: [http://www.mioplanet.com/rsc/newsticker\\_javascript.htm](http://www.mioplanet.com/rsc/newsticker_javascript.htm)

ambiente académico). Quanto a trabalhos futuros, vão ser implementadas as mudanças propostas ao nível do *design*, *layout* e posicionamento dos *displays* dentro do complexo universitário.

### 2.3.3 *Survey* sobre *digital signage* “semi-público” para grupos pequenos e co-localizados

Este trabalho focaliza-se em painéis destinados a grupos pequenos de pessoas co-localizadas, mais propriamente num ambiente laboratorial académico [Huang and Mynatt, 2003]. Os painéis, utilizados neste ambiente, são designados como semi-públicos, porque a informação disponibilizada por estes, é destinada a auxiliar membros de um grupo pequeno e co-localizado, dentro de um local confinado (laboratório) e não acessível ao público.

Para explorar o *design* de painéis semi-públicos, desenvolveram-se quatro aplicações para um laboratório (Everyday Computing Lab – ECL), com o intuito de mostrar informação importante, para promover o conhecimento e a colaboração entre membros de um grupo.

De seguida, vão ser analisadas as quatro aplicações desenvolvidas e as opiniões dadas pelos membros do laboratório, com o objectivo de saber as qualidades e defeitos das aplicações e os aspectos a melhorar no futuro.

#### Attendance Panel

Esta aplicação consiste numa visualização abstracta sobre a presença ou não de membros do grupo em próximos eventos, considerando os interesses do grupo. Essa visualização abstracta consiste numa “flor”, cujo o centro é um círculo com o título de um evento (quando um utilizador toca dentro do círculo, a descrição do evento aparece) e cada “pétala” representa um membro do grupo. Como cor de fundo, escolheu-se o azul, cada “pétala” tem três estados e uma cor diferente para cada estado: um azul escuro, camuflado pela cor de fundo, é utilizado para membros que não pretendem comparecer no evento; um cor-de-rosa claro (cor de ênfase num fundo azul) é utilizado para membros que querem participar no evento; por fim, um branco para membros que ainda não decidiram nada. É de sublinhar que, por questões de privacidade, cada “pétala” colorida não dá informações sobre a identidade dos membros em questão.

#### Reminder Panel

Aplicação do estilo *slideshow*<sup>5</sup> destinada a apresentar informação sobre pedidos de ajuda ou lembretes e com o propósito de incentivar a troca de palavras ou discutir ideias entre os membros do grupo. Outro objectivo é reunir toda a informação, de carácter importante, num único ponto partilhado para ser

---

<sup>5</sup>Exemplo de *slideshow*: <http://www.ajaxdaddy.com/demo-interface-slideshow.html>

acedido facilmente por qualquer utilizador (espécie de fórum de discussão visível num *display*).

### **Collaboration Space**

A aplicação Collaboration Space tem como objectivo proporcionar um espaço dinâmico, com a possibilidade de edição por qualquer membro do grupo. É criado um item por cada pedido de ajuda de membros e, desta forma, esta aplicação funciona ciclicamente, iterando sobre todos os pedidos, sendo que cada item é apresentado durante vários minutos.

Os membros do grupo podem interagir com o *display* e adicionar conteúdos, usando uma caneta própria, de forma a responder aos itens colocados por outros membros que necessitam de ajuda. Essa funcionalidade é proporcionada pelo SATIN [Hong and Landay, 2000], tratando-se de um *toolkit*, baseado em Java, que permite a criação de aplicações que possibilitam a escrita informal através de canetas próprias.

Existem outros trabalhos relacionados, como por exemplo, o Notification Collage [Greenberg and Rounding, 2001], onde cada utilizador pode adicionar itens de potencial interesse para o grupo (vídeos, imagens, notas informativas, eventos, etc.), e o “*What’s Happening*” *screen-saver* [Zhao and Stasko, 2002], que apresenta informação sobre pessoas (rede social em *digital signature*) para promover a criação de grupos, tendo como alvo uma audiência muito vasta, contrariamente ao que acontece neste *survey*.

### **Active Portrait**

Aplicação que apresenta de forma abstracta a actividade do grupo, ou seja, é utilizada uma fotografia com os membros do grupo para representar a sua actividade nos computadores do laboratório. Na fotografia, os membros têm a sua imagem com coloração total, caso estes estejam ou tenham estado recentemente activos. As imagens dos membros menos activos, ao longo do tempo, começam a perder coloração e a ficar mais esbranquiçadas, permitindo que os membros do grupo tenham um conhecimento abstracto da actividade de cada um no laboratório.

Contrariamente a outras ferramentas, como por exemplo, calendários partilhados ou *in-out boards* [Salber et al., 1999a], a imagem apresentada no *display* não apresenta informação exacta sobre os tempos em que um membro sai ou entra no laboratório, proporcionando, assim, um maior grau de privacidade.

### **Estudo feito após a instalação das aplicações**

A aplicação Attendance Panel foi largamente aceite pelos utilizadores, uma vez que acharam as visualizações úteis para determinar os eventos mais importantes para o grupo. A pouca informação fornecida pela aplicação não foi



um factor de decisão no que diz respeito à vontade dos planos de cada um em participar em eventos. No entanto, os utilizadores disseram que seria mais prático se pudessem ter acesso aos planos de cada colega, visto que se conheciam bem e, neste caso particular, como se trata de um painel semi-público (reservado a um público restrito), não há problemas de privacidade.

A aplicação Reminder Panel também foi bem aceite pelos utilizadores, pois estes a acharam bastante útil no que diz respeito à manutenção do conhecimento e à troca de ideias sobre o seu trabalho diário, permitindo a entre-ajuda nas suas tarefas, uma vez que os pedidos eram facilmente vistos nos *displays*.

A aplicação Collaboration Space provou ser um pouco problemática, visto que os utilizadores a acharam mais divertida do que propriamente útil e, por isso, escreviam coisas interessantes, mas não de carácter profissional. Deste modo, como espaço destinado à colaboração, deixa um pouco a desejar, uma vez que é difícil utilizar a caneta no *display* e a colaboração é bastante mais eficaz quando esta é efectuada através do diálogo directo, até porque se trata de um grupo pequeno em que as pessoas se conhecem bem.

A aplicação Active Portrait sofreu problemas técnicos, uma vez que os membros do grupo acharam os níveis de visibilidade da imagem difíceis de distinguir. Também houve problemas ao nível da imprecisão trazida pelo facto de se monitorizar apenas a actividade nos computadores de cada um, uma vez que o grupo trabalha muitas vezes sem a sua utilização. Como solução, foi sugerida a monitorização à base de RFID em vez de monitorização de computadores e também propôs-se a melhoria do contraste da imagem, para a aplicação ter mais utilidade para os membros do laboratório.

## 2.4 Problemas inerentes à publicidade em *digital signage*

Como se sabe, há locais no mundo como Shibuya Crossing ou Times Square (observar a parte do lado esquerdo da Figura 2.4) onde as pessoas são “inundadas por rios” de anúncios publicitários [Müller and Krüger, 2007]. Teoricamente, as pessoas podem ir para sítios diferentes com menos publicidade, mas na prática as pessoas são forçadas a frequentarem certos locais onde os agentes publicitários fazem uma campanha feroz para que a sua publicidade seja vista (para tal utilizam várias estratégias, já desenvolvidas nas secções anteriores, para promover a atenção do público).

As pessoas que passam por painéis de *digital signage*, em locais como Times Square, têm muitas vezes dificuldade em distinguir a informação que é transmitida, uma vez que é muito densa e demasiado dinâmica, dificultando as suas capacidades visuais [Intille, 2002].

Sabendo que cada vez há mais publicidade em certos espaços públicos (grandes avenidas), pergunta-se: haverá sempre uma divergência entre os



**Figura 2.4:** *Times Square (Estados Unidos) repleto de anúncios publicitários vs. publicidade moderada em Prinszipalmarkt (Alemanha) [Müller and Krüger, 2007]*

agentes publicitários e as pessoas que tentam escapar da publicidade? Certamente que esta questão não é apenas técnica mas, principalmente, económica. Se virmos a evolução da publicidade na Internet, é de reparar que, no início, os anúncios publicitários eram apenas imagens estáticas que podiam ser facilmente ignoradas pelas pessoas. Com a evolução da tecnologia, começaram a ser adoptadas imagens dinâmicas baseadas em *flash* e *gifs* contendo animações, fazendo com que as pessoas em geral, tivessem cada vez mais dificuldade em ignorar essa forma de publicidade. As coisas ainda pioraram mais quando surgiram os *e-mails* com *spam* (lixo electrónico), o excesso de janelas de *pop-up* com informação desinteressante e animações que ocupam o ecrã inteiro, o que gerou um sentimento de revolta por parte de muitos utilizadores. Portanto, como medidas de prevenção, foram desenvolvidos filtros de *spam*, *plugins* que bloqueiam janelas de *pop-up*, etc., fazendo com que a situação seja uma espécie de guerra entre os agentes publicitários a tentar vender os seus produtos e os utilizadores a tentar evitar a publicidade.

O mais importante a reter é que, considerando a *digital signage*, não vale a pena enveredar pelo caminho da Internet, em que há demasiada informação publicitária que irrita os utilizadores. Em vez disso, deve-se adoptar publicidade que tenha em conta o contexto envolvente (*targeted advertising*), para que se forme uma boa imagem do local onde os *displays* públicos se encontram (observar a parte do lado direito da Figura 2.4).

### 2.4.1 Publicidade com alvos definidos

A publicidade com alvos definidos (*targeted advertising*) tem-se tornado predominante em relação a sistemas de publicidade pessoais, tais como os sistemas de recomendação ou anúncios publicitários *web based* [Amiri and Menon, 2003]. Os sistemas de publicidade pessoais funcionam muito bem quando aplicados aos utilizadores de forma individual, pois existe, à priori, informação bastante rica sobre o perfil de cada utilizador. Contrariamente a esses sistemas, os sistemas com alvos definidos, como o *BluScreen display*

[Payne et al., 2006], que vai ser abordado na secção seguinte, seleccionam os anúncios publicitários, baseando-se na presença de vários utilizadores, onde existe pouca informação de perfil de utilizador disponível. Tendo em conta essa falta de informação (típico da *digital signage*), deve-se usar sensores para obter máxima informação possível do contexto (tempo, local, dispositivos bluetooth, câmaras, etc.), para se conhecer melhor o público alvo, num dado momento e local.

### 2.4.2 Externalidades negativas da publicidade

A atenção de cada indivíduo é um recurso limitado, quando uma pessoa observa um anúncio publicitário, presta menos atenção a outras tarefas. Por outro lado, a informação publicitária pode ser útil e trazer, por isso, benefícios. Deve-se fazer tudo o que for possível para que esses benefícios sejam superiores aos custos, havendo, para isso, um regulamento mais rigoroso do mercado.

Caso o mercado não esteja regulado, a quantidade de publicidade é muito superior ao valor óptimo, porque os agentes publicitários não pagam pelos custos de atenção das pessoas, mais concretamente, a atenção prestada por cada indivíduo é externa e não é tida em conta nos cálculos efectuados pelos agentes publicitários (externalidade negativa da publicidade). Neste caso o único custo que afecta os cálculos dos publicitários é o proveniente dos espaços publicitários, fazendo com que haja uma compra agressiva desses espaços e, como consequência, as grandes avenidas ficam repletas de publicidade.

Para resolver o problema da externalidade negativa, há três métodos que se devem ter em conta: *maximum permissible values, fees* e *tradable certificates* [Müller and Krüger, 2007]. Cada um destes métodos definem métricas para quantificar a atenção consumida pelas pessoas a observar a publicidade.

#### Maximum Permissible Values

É um método frequentemente usado que consiste em regular a quantidade e o estilo da publicidade existente num local. Tem o problema de obrigar a que toda a publicidade atraia a mesma quantidade de atenção, sendo, por isso, pouco eficaz e não produz uma distribuição óptima da atenção.

#### Fees

É um método alternativo que consiste em investir um determinado orçamento por cada unidade de atenção que é atraída. Em teoria, este método pode resultar numa óptima quantidade de publicidade para a sociedade, mas é muito difícil de gerir.

### Tradable Certificates

É um método que combina as vantagens dos dois anteriores. Nesta abordagem, para cada local é calculada uma certa quantidade de atenção, e esta é representada por certificados, que posteriormente são vendidos num leilão, fazendo com que cada agente publicitário valorize a atenção comprando um maior número de certificados. O principal problema deste método é o alto custo transaccional envolvido na execução do leilão, tornando-o inviável para os actuais esquemas analógicos de publicidade. Contudo, quando os leilões são usados para vender espaço publicitário em *digital signage*, existe muito menos complexidade no processo de leiloar os certificados.

A utilização do mecanismo baseado em leilões, em *digital signage*, vai ser abordado na próxima secção.

## 2.5 *Displays* públicos inteligentes baseados no contexto e em leilões publicitários

Qualquer rede de *digital signage* deve apresentar conteúdos de interesse para o utilizador e, para além disso, deve ser pro-activa, antecipando as necessidades e os interesses do utilizador, fornecendo-lhe a informação adequada. Num dado local e tempo, os sistemas de *digital signage* devem analisar o contexto envolvente, decidir os conteúdos a ser apresentados e medir o efeito da sua apresentação, para que optimizem a utilidade da informação fornecida ao público.

No caso em que as redes de *digital signage* são de grande dimensão, o problema do escalonamento dos conteúdos publicitários torna-se bastante complexo para ser feito pelo ser humano. Para resolver este problema, deve-se recorrer a *digital signage* inteligente [Müller, 2008], ou seja, deve-se utilizar painéis de *digital signage* com a inteligência artificial necessária para poder seleccionar automaticamente os conteúdos a apresentar, em função das situações que ocorrem.

De seguida, vão ser analisados dois exemplos de *displays* inteligentes, o *BluScreen* [Payne et al., 2006] e o *iDisplay* [Müller, 2007], que têm em conta os factores acima mencionados e que utilizam um sistema baseado em leilões publicitários, com o objectivo de escalonar a publicidade de forma automática e eficaz.

### 2.5.1 O *BluScreen*

Como já foi dito anteriormente, os painéis públicos de *digital signage* estão em constante crescimento e são usados para fornecer informação, entreter ou publicitar produtos. Os agentes publicitários tipicamente utilizam uma variedade de métodos para aumentar o número de anúncios publicitários

apresentados, para que tenham o máximo de exposição possível para o público alvo. Contudo, muitas vezes, esses métodos não têm em conta o público que frequentemente passa pelos *displays*.

Para resolver esses problemas, já foram propostos painéis públicos interactivos que suportam a comunicação com o utilizador através de dispositivos móveis ou que são sensíveis ao toque. O problema desses sistemas é que assumem um conhecimento prévio do público alvo e requerem que cada utilizador tenha acesso exclusivo ao *display*. Como consequência, essas abordagens não servem no contexto dos espaços públicos, onde não há qualquer conhecimento prévio dos utilizadores que observam os *displays*, e estes necessitam de reagir à presença de muitos utilizadores em simultâneo.

Tendo em conta um contexto de espaços públicos, vai-se considerar um *display* inteligente, o *BluScreen* [Payne et al., 2006], que utiliza uma nova abordagem para detectar pessoas, com o intuito de melhorar a selecção de conteúdos a apresentar. O objectivo dessa selecção é a de maximizar o número de conteúdos distintos vistos pelas pessoas. Tendo isto em consideração, o sistema utiliza o histórico da exposição de utilizadores em certos conjuntos de publicidade, com o fim de evitar a repetição de anúncios já vistos, juntamente com informação sobre que utilizadores estão a ver o *display* (utilizam-se sensores para detectar dispositivos com Bluetooth). Em particular, foi desenvolvido um mecanismo multi-agente baseado em leilões, para seleccionar, de forma eficiente, um anúncio publicitário para cada *slot* temporal (ou ciclo publicitário<sup>6</sup>).

O *BluScreen* utiliza sensores para detectar dispositivos com tecnologia Bluetooth e com a finalidade de identificar os utilizadores. Assim, sempre que um utilizador passa pelo *display*, o sistema armazena, em histórico, o tempo durante o qual este observa o anúncio publicitário. Pode-se assumir que quanto maior for esse tempo, maior é o nível de interesse por parte do utilizador no conteúdo publicitado. Normalmente um utilizador que esteja realmente interessado no material publicitário apresentado, fica exposto ao *display* durante a totalidade da apresentação.

### ***BluScreen*: Arquitectura baseada em agentes**

Como se pode observar na Figura 2.5, a arquitectura do *BluScreen* é composta pelos seguintes agentes:

- **Agente de detecção de dispositivos com Bluetooth:** Este agente monitoriza o ambiente, verificando periodicamente se há dispositivos Bluetooth detectáveis;
- **Agente Publicitário:** Cada Agente Publicitário representa um único anúncio publicitário e é responsável por comprar espaço publicitário,

---

<sup>6</sup>Um ciclo publicitário é um intervalo de tempo durante o qual um anúncio publicitário é apresentado.

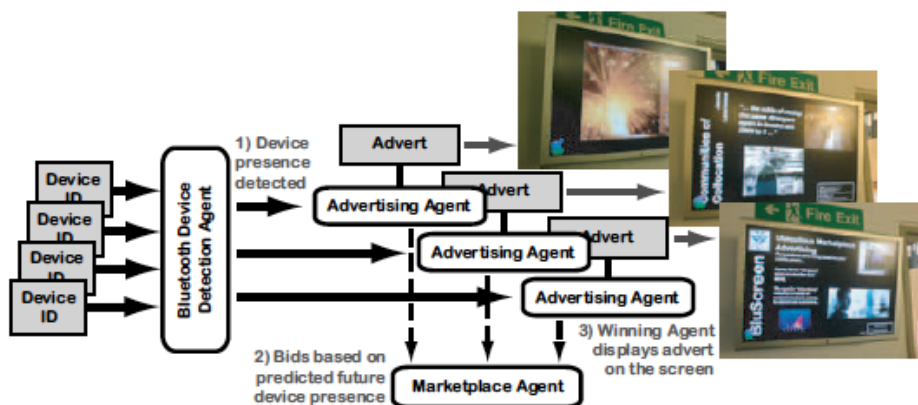


Figura 2.5: Arquitetura do BluScreen baseada em agentes [Payne et al., 2006]

gerando propostas com um valor previsto para o próximo ciclo publicitário. Este valor é baseado na exposição do anúncio para o maior número de utilizadores possível e, para esse propósito, cada agente armazena informação de histórico dos ciclos publicitários de sucesso (quando o agente ganha o leilão, com o fim de transmitir a sua publicidade associada). Em adição a este histórico recolhido, para cada ciclo publicitário, é armazenado o tempo durante o qual cada dispositivo Bluetooth foi detectado pelos sensores do *display*.

- **Agente de *MarketPlace*:** Este agente facilita a venda de *slots* publicitários e, como mecanismo de selecção, utiliza um leilão de segundo-preço (ou leilão de Vickrey [Vickrey, 1961]) para determinar qual é o agente publicitário que vai disponibilizar o seu conteúdo no próximo *slot* de tempo, para além do correspondente pagamento.

### ***BluScreen*: Mecanismo baseado em leilões**

O *BluScreen* foi desenhado para suportar uma *framework* publicitária escalável e dinâmica, com o objectivo de expor o máximo número de anúncios publicitários possíveis para o maior número de pessoas possível, num ambiente de pouco conhecimento (não se sabe o perfil de cada pessoa). Por conseguinte, o sistema, como um todo, decide qual o agente vencedor da maior exposição do seu conteúdo, no próximo ciclo publicitário. É assumido que cada agente propõe um valor que reflecte o seu desejo de expor os seus conteúdos publicitários para o próximo ciclo. Com esse propósito foi implementado um leilão de segundo-preço [Vickrey, 1961] que tem lugar antes de cada ciclo, e o vencedor do leilão corresponde àquele agente que fez a maior licitação e que pagará o valor correspondente à segunda maior licitação.

Durante cada ciclo publicitário, há apenas um agente (o que ganhou o leilão) que apresenta o seu conteúdo, sendo este o único com a habilidade de obter a informação dos dispositivos com Bluetooth. Contudo, no fim de cada ciclo, o agente de *MarketPlace* informa todos os agentes publicitários sobre os dispositivos detectados à frente do ecrã naquele intervalo de tempo.

Para cada agente publicitário,  $a_j$ , assume-se que é gerido por um certo cliente que o credita com um orçamento fixo para publicitar. Para um dado ciclo, cada agente publicitário, para além de ter o orçamento fixo, gera um valor privado que é convertido para a moeda em uso durante o leilão. Quando um agente gasta todo o seu orçamento, este é removido do sistema, mas a qualquer momento podem ser adicionados novos agentes ao sistema.

É de referir que o agente de *MarketPlace* tem a função de leiloeiro, sendo este que executa o leilão antes de cada ciclo publicitário. É este agente que deve definir o mínimo valor aceitável para licitação, e no caso de nenhum agente publicitário atingir o valor mínimo, o ecrã apenas apresenta informação padrão.

Para o correcto funcionamento do sistema, o agente de *MarketPlace* não pode permitir que os agentes publicitários tenham acesso às restantes propostas, visto que se trata de um leilão de segundo-preço com as licitações seladas (ou ocultas) [Vickrey, 1961].

### ***BluScreen*: Simulações efectuadas**

Por fim, foram feitas simulações para comparar o mecanismo de leilões do *BluScreen* com os mecanismos de selecção *round-robin* e *random*. O importante a dizer sobre essas simulações é que se concluiu que o mecanismo de leilões é mais eficiente quando comparado com os outros dois [Payne et al., 2006].

#### **2.5.2 O *iDisplay***

O *iDisplay* [Müller, 2007], tal como o *BluScreen*, é um painel digital público que se baseia em leilões de segundo-preço, com a finalidade de alocar os anúncios publicitários mais apropriados ao contexto inserido (recorrendo a câmaras, dispositivos Bluetooth, etc.). Há, assim, uma maior probabilidade de as pessoas acederem a anúncios com utilidade efectiva, evitando que as mesmas ignorem os *displays*.

Comparando as diferenças entre o *iDisplay* e o *BluScreen*, é de sublinhar que o primeiro apenas considera os anúncios publicitários que têm potencial para desencadear uma acção, por parte das pessoas que observam o *display*, num determinado contexto temporal e espacial (caso típico da publicitação de eventos). Por exemplo, quando um dado *display* apresenta um anúncio de um filme que vai estrear, pode-se dizer que esse anúncio tem potencial para provocar uma acção, isto é, estimular a ida ao cinema para assistir a um

novo filme, à data e hora marcada. No caso do *BluScreen*, não há qualquer distinção do tipo de publicidade, sendo, por isso, mais generalista.

Outro aspecto a considerar sobre o *iDisplay* é o facto de se incluírem vários *slots* publicitários no ecrã do painel, enquanto que no *BluScreen* apenas é apresentado um anúncio de cada vez.

### ***iDisplay*: Mecanismo baseado em leilões**

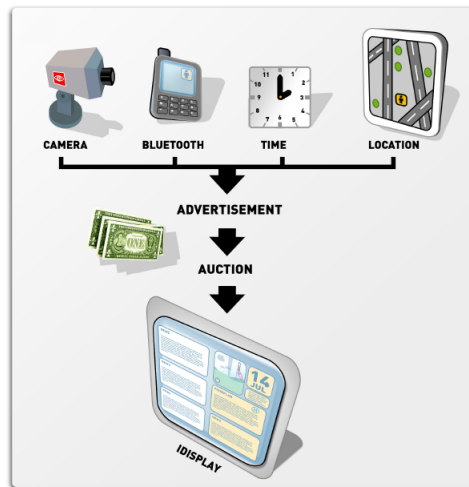
É importante adoptar o melhor mecanismo de venda de *slots* publicitários e, por isso, deve-se ter em conta três formas de pagamento: *per-impression*, *per-view* e *per-attendance* [Müller, 2007]. No pagamento *per-impression* o agente publicitário paga por cada segundo que o seu anúncio é apresentado, não tendo em conta se alguém o viu. No pagamento *per-view* o agente publicitário paga por cada pessoa que efectivamente viu a publicidade. Por fim, no pagamento *per-attendance* o agente publicitário apenas retribui por cada pessoa que viu a publicidade e que, de seguida, tenha executado a acção. Por causa das dificuldades técnicas de implementação dos pagamentos *per-view* e *per-attendance*, foi escolhido, como ponto de partida, o pagamento *per-impression*.

Como alternativa semelhante à publicidade feita pela Google, são vendidos múltiplos *slots* publicitários ao mesmo tempo e, deste modo, estes são apresentados simultaneamente no *display*, com o propósito de aumentar a probabilidade de que o utilizador esteja interessado na publicidade disponibilizada. Daí que, tenha sido implementado um leilão de segundo-preço generalizado, similar ao implementado pela Google [Edelman, 2005] (no entanto a Google utiliza um sistema de pagamento *per-click*). Como no caso da *digital signage*, não há acesso a *keywords* como na Google, deve-se adoptar um mecanismo que fornece, com base no contexto, informação sobre o que cada agente publicitário deve licitar no leilão, para que possa ser concretizada a melhor opção em termos de investimento.

Analisando a Figura 2.6, é possível ver o processo para a concretização do leilão e a alocação dos anúncios publicitários, vencedores do leilão, nos *slots* publicitários do *iDisplay* [Müller and Krüger, 2007].

No início de cada ciclo publicitário é recolhida informação sobre o contexto, sendo que cada anúncio publicitário tem acesso ao número de utilizadores a observar o *iDisplay*, a sua identidade, o tempo e o local. Os anúncios publicitários, com base nesta informação, estimam a probabilidade dos utilizadores actuarem e computam a utilidade expectável [Müller, 2007] de serem alocados nos *slots* publicitários, com o objectivo de gerar as licitações para o próximo ciclo. Por fim, os anúncios vencedores do leilão são escolhidos para serem apresentados no *iDisplay*, durante o ciclo publicitário.





**Figura 2.6:** Processo de concretização de um leilão para alocar anúncios publicitários no *iDisplay* [Müller and Krüger, 2007]

### 2.5.3 Conclusões

O factor determinante da *digital signage* é a utilização do contexto envolvente como meio de selecção de material publicitário e informativo. Assim, o público passa a observar material do seu interesse e a ter uma ideia bem mais positiva da publicidade, facto que raramente ocorre em algumas cidades de grande dimensão, onde há um excesso de publicidade e não há controlo do tipo de conteúdos apresentados.

Tal como acontece na televisão, também os agentes publicitários devem ter acesso à procura e interesse que realmente existe, a cada momento, nos seus anúncios publicitários disponibilizados em painéis de *digital signage*, com vista a saberem qual o valor justo do respectivo investimento. Para isso, deve-se ter em conta os vários dados existentes no contexto onde os painéis estão inseridos e, deste modo, os conteúdos serão apresentados para as pessoas certas e nas melhores alturas possíveis.

## Capítulo 3

# Especificação do Problema

Actualmente, no mundo comercial, devido à grande competitividade existente, é necessário que se utilize a tecnologia como um meio essencial para cativar a atenção dos clientes e, assim, promover todo o tipo de produtos de forma eficaz, recorrendo ao menor número de recursos possível e diminuindo os custos de promoção destes produtos. Daí que se utilizem, muitas vezes, painéis informativos (*digital signage*) em locais públicos (como por exemplo, em centros comerciais, aeroportos, restaurantes, etc.) para propagandas publicitárias ou promoção de certos eventos de interesse público.

Deste modo, a informação sobre eventos que estejam a decorrer ou estejam planeados para um determinado local coberto por uma rede de *digital signage* é de extrema relevância para os seus utentes. Atendendo a que a produção de conteúdo promocional faz incorrer geralmente em custos elevados (design e produção), torna-se importante minimizar os custos de promoção de eventos em redes de *digital signage*.

A popularidade e carácter aberto de certas ferramentas *on-line* de gestão/promoção de eventos facilita grandemente o desenvolvimento de aplicações neste domínio. Por isso, é bastante útil recorrer a APIs para interagir com esse tipo de ferramentas *on-line* que fornecem interfaces à base de calendários (e não só<sup>1</sup>), para gerir e escalonar eventos. Como exemplo desse tipo de ferramentas, destacam-se as seguintes: o Google Calendar (principal ferramenta a ser integrada com a aplicação a desenvolver), a Microsoft Exchange, o Facebook, a Eventbrite, entre muitas outras.

### 3.1 Introdução

Neste capítulo, uma vez terminada a abordagem ao estado da arte sobre painéis de *digital signage*, vai ser introduzido o projecto de *software* a ser

---

<sup>1</sup>No caso da ferramenta *Eventbrite* a interface disponibilizada é à base de um *wizard* com todos os passos para criar um evento.

desenvolvido no âmbito empresarial (sugerido pela Ubisign<sup>2</sup>). Assim sendo, este projecto vai consistir no desenvolvimento de uma aplicação *web-based* destinada a apresentar, aos utentes de painéis de *digital signage*, informação sobre eventos disponíveis no serviço Google Calendar.

Como ponto de partida, vão ser analisadas algumas ferramentas *on-line* de gestão de eventos, como a Microsoft Exchange, o Eventbrite, o Facebook, com particular destaque para o Google Calendar. Também vai ser efectuada uma introdução sobre as APIs fornecidas pela Google, mais concretamente a API Google Calendar e aprofundar-se-á a questão, de particular importância, sobre o tema da autenticação e autorização nos serviços da Google. Dar-se-á especial ênfase aos protocolos OpenID e OAuth, de modo a constituírem a melhor solução para permitir o acesso ao serviço Google Calendar, por parte da aplicação *web* a desenvolver.

De seguida, vai-se abordar a descrição do problema proposto pela Ubisign, isto é, vão ser descritos os quatro tipos de apresentação (ou visualização) de eventos que a aplicação *web* terá que proporcionar. Desta forma, a aplicação, na sua essência, irá possibilitar a visualização de eventos no modo Full (ecrã inteiro), Column (em coluna), Bar (barra horizontal ou rodapé) e XML. Também é de salientar que as visualizações Full, Column e Bar vão permitir uma customização/personalização de uma forma simples, mas ao mesmo tempo eficaz por parte do utilizador. Assim, essa customização irá ser feita através de um formulário *web*, para cada uma das visualizações referidas anteriormente (inclusive a visualização XML). Após o preenchimento de um formulário e respectiva submissão, a aplicação recebe os parâmetros de configuração, com o fim de processar a respectiva visualização de eventos, tendo em conta as preferências do utilizador. A aplicação também deverá permitir o armazenamento de configurações numa base de dados e receber parâmetros de configuração através de aplicações externas, sem ser necessário qualquer tipo de formulário.

Uma vez terminada a descrição sobre o que é o projecto, vai ser apresentado o Modelo do Domínio do problema e os respectivos requisitos funcionais e não funcionais.

### 3.2 Ferramentas *on-line* de Gestão de Eventos

Como já foi dito na introdução deste capítulo, existem várias ferramentas *on-line* de gestão de eventos. Daí que nesta secção será dada particular ênfase ao Google Calendar e vai-se introduzir, com menos detalhe, a Microsoft Exchange, o Eventbrite e a criação de eventos no Facebook.

---

<sup>2</sup><http://www.ubisign.com>

**Criar novo calendário**

Detalhes do calendário

[« Regressar ao calendário](#)

Nome do calendário:

Descrição:

Local:   
ex.: "Lisboa", "Porto" ou "Portugal". Ao especificar um local geral irá ajudar outras pessoas a encontrarem eventos no seu calendário (ca:

**Fuso Horário do Calendário:**  
Primeiro, seleccione um país para seleccionar o conjunto de fusos horários pretendido. Para ver todos os fusos horários, assinala a caixa.

Pais:  (escolha um país diferente para ver outros fusos horários)

Agora, seleccione um fuso horário:   Apresentar todos os fusos horários

**Tornar este calendário público**  
Este calendário aparecerá nos resultados públicos de pesquisa do Google.

Partilhar apenas a minha informação disponível/ocupado (Ocultar detalhes)

**Partilhar com pessoas específicas**

Pessoa	Definições de autorização
<input type="text" value="Introduza o endereço de e-mail"/>	<input type="text" value="Ver todos os detalhes do evento"/>

**Figura 3.1:** Formulário para criar um calendário no Google Calendar

### 3.2.1 Google Calendar

Este serviço disponibilizado pela Google é bastante útil, uma vez que permite uma gestão de eventos *on-line* à base de um ou mais calendários, com uma interface *user-friendly* e bastante simples de se usar.

De seguida vão ser abordadas as principais funções fornecidas pelo serviço Google Calendar.

#### Criação de Calendários

Para efectuar a criação de um calendário, a Google disponibiliza um formulário para atribuir o nome do calendário, uma descrição (informação opcional para saber com maior detalhe em que consiste o calendário), o local (para ajudar outras pessoas a encontrarem eventos no calendário criado, no caso do calendário ser público) e o fuso horário do calendário. Permite, também, tornar o calendário público (para aparecer nos resultados de pesquisa do Google) e partilhar eventos com pessoas específicas (através da introdução dos seus *e-mails*). Para uma visualização mais esclarecedora, pode-se observar o formulário na Figura 3.1.

Sendo possível elaborar mais do que um calendário, é frequente criar um calendário para cada assunto em particular, de forma a que os eventos estejam bem organizados. Por exemplo, no calendário X pode-se introduzir apenas eventos de carácter desportivo, no calendário Y pode-se introduzir um horário de trabalho, etc..

Por uma questão de melhor distinção de cada calendário, a ferramenta de gestão de eventos da Google também permite atribuir cores diferentes para cada um. Na mesma visualização pode-se juntar dois ou mais calendários,

Mon 6/6	Tue 6/7	Wed 6/8	Thu 6/9	Fri 6/10
	11 – 1p Natação	11 – 12p Ginásio	11 – 1p Natação	
		12p – 1p Descanso		
1p – 2p Almoço	1p – 2p Almoço	1p – 2p Almoço	1p – 2p Almoço	1p – 2p Almoço
2p – 3p Descanso	2p – 3p Descanso	2p – 7p Torneio de Tênis	2p – 8:30p Vários Eventos Desportivos	
3p – 4p Novo Evento	3p – 9p Campeonato de Futebol			3p – 4p Novo Evento
		7p – 8p Descanso		
	9p – 10p Descanso		8:30p – 9:30p Descanso	

**Figura 3.2:** *Eventos correspondentes a calendários diferentes (marcados com cor diferente) no Google Calendar*

conseguindo distingui-los através da cor atribuída, como se pode observar na Figura 3.2.

### Definições de um Calendário

Cada calendário tem associada informação que foi inserida no momento da sua criação (nome, descrição, local, etc.) e que pode ser alterada no formulário de definições de um calendário, apresentado na Figura 3.3. Para além desta informação, também é possível adicionar/aceitar convites de eventos, incorporar um calendário num *website* (com possibilidade de personalizar o calendário) e pode-se obter 3 tipos de endereços do calendário: um que fornece uma página em XML (útil para leitores de *feeds*), outro no formato iCalendar<sup>3</sup> (para o calendário ser facilmente integrado noutras aplicações que usam este formato estandardizado) e um último que fornece uma página no formato HTML (para visualizar no *browser*).

### Criação de Eventos

O processo de criação de eventos é bastante simples, uma vez que basta clicar numa célula do calendário para aparecer um “balão” com o fim de nele ser introduzido o nome do evento e associá-lo ao calendário pretendido (como

<sup>3</sup><http://www.infowester.com/blog/voce-sabe-o-que-e-ical/>

### Detalhes de Calendário 1

[Detalhes do calendário](#)
[Partilhar este calendário](#)
[Notificações](#)

[« Regressar ao calendário](#)

**Nome do calendário:**


**Proprietário do calendário:** "pg15223@alunos.uminho.pt" <pg15223@alunos.uminho.pt>

**Descrição:**

**Local:**   
ex.: "Lisboa", "Porto" ou "Portugal". Ao especificar um local geral irá ajudar outras pessoas a encontrarem e

**Fuso Horário do Calendário:** Este calendário está a utilizar o seu fuso horário actual: (GMT+00:00) Lisboa [Definir o fuso horário](#)

**Incorporar este calendário**  
Incorpore este calendário no seu Web site ou blogue, colando este código na sua página Web. Para incorporar vários calendários, clique na hiperligação Personalizar



Cole este código no seu Web site.  
[Personalize a cor, tamanho e outras opções](#)

```
<iframe src="https://www.google.com/calendar/embed?src=pg15223@alunos.uminho.pt" width="800" height="600" frameborder="0" scrolling="no"></iframe>
```

**Endereço do calendário:** [XML](#) [iCAL](#) [HTML](#) (ID do calendário: pg15223@alunos.uminho.pt)  
Este é o endereço para o seu calendário. Ninguém poderá utilizar esta hiperligação, a menos que tenha t

**Endereço privado:** [XML](#) [iCAL](#) [HTML](#) [Repor URLs privados](#)  
Este é o endereço privado para este calendário. Não partilhe este endereço com outras pessoas, a não se

**Figura 3.3:** *Detalhes de um calendário do Google Calendar*

se pode visualizar na Figura 3.4). Para alterar a duração do evento basta ajustar o tamanho do rectângulo correspondente ao evento. Para introduzir mais informação sobre o evento, o utilizador pode clicar num *link* existente no “balão”, com a finalidade de ser apresentado um formulário mais completo.

Nesse formulário (Figura 3.5) é possível introduzir informação sobre a data/hora de início e fim, o local, a descrição, a cor da célula do evento, o nível de privacidade, etc.. Também existe a possibilidade de configurar eventos de forma a que estes sejam recorrentes, ou seja, pode-se criar um evento que repita durante um certo período de tempo e num certo dia, ou vários dias da semana/mês/ano. Por exemplo, é possível criar um evento com recorrência de 2 em 2 semanas, às segundas e sextas, com o início no dia 6/6/2011 e que termine após 35 ocorrências (Figura 3.6).

### APIs da Google

Mudando para um contexto mais técnico e que está mais relacionado com o que a aplicação *web* a desenvolver vai utilizar, a Google fornece APIs para comunicar com vários dos seus serviços, como por exemplo: o já referido Google Calendar, Google Docs, Email Settings, YouTube, etc..<sup>4</sup>. Uma vez que a aplicação *web* vai ser programada utilizando a .NET Framework, para esta poder interagir com o serviço Google Calendar, é necessário importar

<sup>4</sup><http://code.google.com/p/google-gdata/>

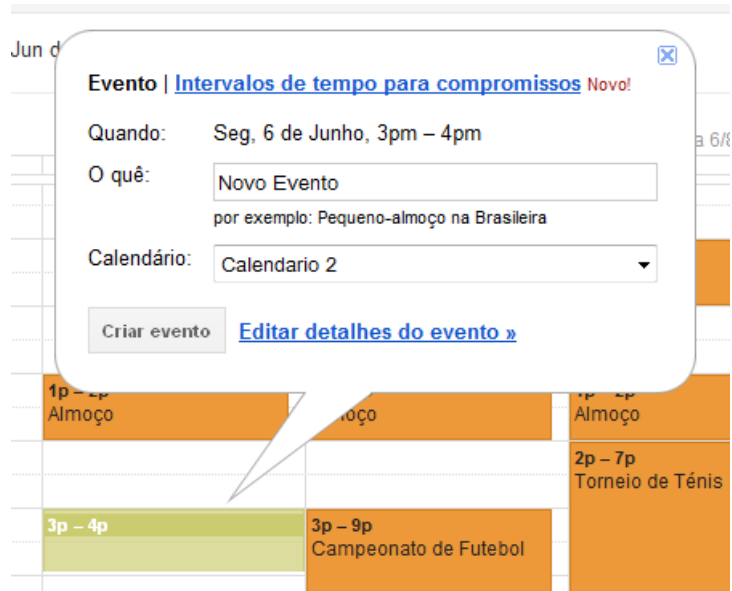


Figura 3.4: Exemplo de criar um evento no Google Calendar

**Novo Evento**

5/17/2011 2:00pm a 3:00pm 5/17/2011 [Fuso horário](#)

Todo o dia  Repetir...

Detalhes do evento [Encontrar uma hora](#)

Onde

Calendário **Calendário 1**

Descrição

Cor do evento

Lembretes **E-mail** 10 minutos   
**Pop-up** 10 minutos   
[Adicionar um lembrete](#)

Mostrar-me como  Disponível  Ocupado

Privacidade  Predefinido  Público  Privado

Figura 3.5: Formulário completo para criar um evento no Google Calendar

**Repetir**

Repete-se: Semanal

Repetir todos os(as): 2 semanas

Repetir no(a):  D  S  T  Q  Q  S  S

Tem início às: 6/6/2011

Termina no dia:  Nunca  Após 35 ocorrências  No dia

Resumo: A cada 2 semanas à Segunda-feira, Sexta-feira, 35 vezes

Concluído Cancelar

**Figura 3.6:** Formulário para criar eventos recorrentes no Google Calendar

DLLs (*Dynamic-link libraries* com a API Google Calendar).

No que diz respeito à API Google Calendar, é importante referir que a Google fornece a documentação<sup>5</sup> necessária, para .NET, sobre as funções de autenticação e sobre as funções que provocam o retorno dos eventos consoante a *query* utilizada. Todas as outras funções, como por exemplo, criar calendários, criar eventos, actualizar eventos, remover eventos, etc., não vão ser relevantes para a aplicação a desenvolver, uma vez que não se vai tratar de uma aplicação de gestão de calendários/eventos (gestão que já é feita pelo próprio Google Calendar), tratando-se sim de uma aplicação que vai apresentar eventos com vários tipos de visualizações e com possibilidade de customização da interface.

### Autenticação e Autorização

Numa questão tão sensível e confidencial, como é o caso da autenticação realizada em *web sites*, é importante evitar, sempre que for possível, o armazenamento de dados de autenticação de utilizadores. Com o objectivo de combater a persistência de *passwords*, deve-se recorrer a certos protocolos de autenticação standardizados, como por exemplo, a autenticação via OpenID. Também é bastante relevante sublinhar que a utilização do OpenID, evita o incómodo do utilizador ter que criar uma nova conta para cada *web site* que requeira autenticação.

A Google, sendo ela uma empresa à escala mundial, proporciona uma autenticação via OpenID e, por isso, qualquer que seja a aplicação *web* que

<sup>5</sup>[http://code.google.com/intl/pt-PT/apis/calendar/data/2.0/developers\\_guide\\_dotnet.html](http://code.google.com/intl/pt-PT/apis/calendar/data/2.0/developers_guide_dotnet.html)



necessite de proporcionar um serviço de autenticação, pode recorrer à Google. Quando um utilizador se autentica numa aplicação *web*, via OpenID, o servidor de OpenID retorna um *token* persistente (identificador do utilizador) que pode ser gerido pela aplicação *web* (por exemplo: gerir a sessão do utilizador, armazenar dados específicos pertencentes ao utilizador, etc.).

No que diz respeito à autorização, pode-se dizer que é um mecanismo em que um sistema determina qual o nível de acesso aos seus recursos, por parte de um utilizador. Por exemplo, um caso muito típico é o que ocorre nas bases de dados, em que os utilizadores com privilégios administrativos podem alterar os dados existentes, enquanto que os utilizadores sem esse tipo de privilégios apenas podem efectuar *queries* para obter informações sobre os dados e, eventualmente, processar esses dados (no entanto, não podem remover ou modificá-los fisicamente na base de dados)<sup>6</sup>.

Como se pode concluir, os mecanismos de autenticação e de autorização estão fortemente acoplados, uma vez que quando um utilizador se autentica está a “dizer” ao sistema qual é a sua identidade e, assim, o próprio “sabe” quais são os recursos sobre quais o utilizador tem autorização para aceder ou que tipo de operações pode efectuar.

A Google suporta o protocolo padrão de autorização, OAuth, que tem vindo a ser frequentemente utilizado por *web sites* conhecidos, como por exemplo, o Twitter e o Facebook. No que diz respeito aos serviços fornecidos pela Google, é possível uma aplicação utilizar o protocolo OAuth, para obter as credenciais de acesso a um ou mais serviços, desde que o utilizador o permita. E é nesse sentido em que a aplicação *web* sobre eventos a ser desenvolvida, vai tirar partido dos protocolos OpenID e OAuth, suportados pela Google, para obter acesso aos eventos de um calendário do utilizador.

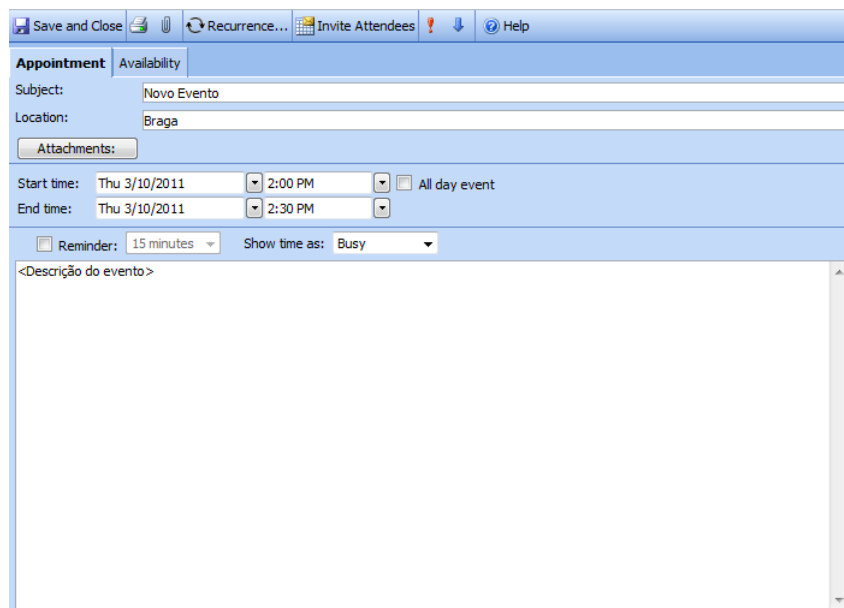
Uma vez que os protocolos OpenID e OAuth estão fortemente interligados, é possível utilizar uma solução híbrida de autenticação/autorização (protocolo híbrido). De uma forma sucinta, é necessário que, em primeiro lugar, uma aplicação *web* efectue a autenticação via OpenID para que esta obtenha um *token* de pedido. A partir do protocolo OAuth, a aplicação troca o *token* de pedido por um *token* de acesso ao serviço (mecanismo de autorização)<sup>7</sup>. Como a Google permite o uso do protocolo híbrido, vai ser esta a solução de autenticação/autorização a utilizar na aplicação *web* a desenvolver, de modo a que se verifique se o utilizador permite ou não o acesso ao serviço Google Calendar.

### 3.2.2 Microsoft Exchange

Abordando a ferramenta de gestão de eventos, Microsoft Exchange, é possível visualizar na Figura 3.7, o formulário de criação de eventos, com os seguintes

<sup>6</sup><http://www.duke.edu/~rob/kerberos/authvauth.html>

<sup>7</sup>[http://code.google.com/intl/pt-PT/googleapps/domain/sso/openid\\_reference\\_implementation.html](http://code.google.com/intl/pt-PT/googleapps/domain/sso/openid_reference_implementation.html)

The image shows a screenshot of the Microsoft Exchange 'Appointment' form. The form is titled 'Appointment' and has a sub-tab 'Availability'. It contains several fields: 'Subject' with the value 'Novo Evento', 'Location' with the value 'Braga', and an 'Attachments' button. Below these are 'Start time' (Thu 3/10/2011, 2:00 PM) and 'End time' (Thu 3/10/2011, 2:30 PM) fields, with an 'All day event' checkbox. There is also a 'Reminder' section with a '15 minutes' dropdown and a 'Show time as: Busy' dropdown. At the bottom is a large text area for the event description, currently containing the placeholder text '<Descrição do evento>'. The top of the window shows a menu bar with 'Save and Close', 'Recurrence...', 'Invite Attendees', and 'Help'.

**Figura 3.7:** Formulário de criação de um novo evento na Microsoft Exchange

campos: assunto, local, data/hora de início e fim, descrição do evento e se o evento dura todo o dia.

Tal como no Google Calendar, também existe uma opção de recorrência que permite repetir um evento durante determinado período de tempo. O padrão de recorrência pode ser diário, semanal, mensal ou anual. Também é possível estabelecer um limite de ocorrências, uma data limite ou nem sequer haver limite. Na Figura 3.8 pode-se visualizar o formulário para definir opções de recorrência para um evento no Microsoft Exchange.

Considerando a Figura 3.9, é possível ver que o evento designado por “Novo Evento” repete-se todas as quintas-feiras às duas horas da tarde (evento recorrente). Outro evento que se destaca é o “Evento Todo o Dia” que, tal como o nome indica, dura um dia inteiro e é delimitado por um rectângulo para se distinguir dos outros.

Como é de reparar, a Microsoft Exchange é muito semelhante ao Google Calendar e também permite a criação de vários calendários, o que faz desta uma ferramenta *on-line* muito útil para gerir eventos sobre temas variados.

Para finalizar, certamente há bastantes mais funcionalidades disponibilizadas pela Microsoft Exchange que não são aqui referidas, pois o principal foco da dissertação é o Google Calendar (o mesmo se poderá dizer nas próximas duas subsecções, sobre o Facebook e o Eventbrite).

The screenshot shows the 'Appointment time' dialog box in Microsoft Exchange. It is configured with the following settings:

- Appointment time:** Start: 2:00 PM, End: 2:30 PM.
- Recurrence pattern:** Recurs every 1 week(s) on Thursday.
- Range of recurrence:** Start: Thu 3/10/2011, No end date.

Buttons at the bottom: OK, Cancel, Remove Recurrence.

Figura 3.8: Configuração de recorrência de um evento na Microsoft Exchange

The screenshot shows the monthly calendar view for March 2011. The calendar displays events for each day of the month. The events are as follows:

Mon	Tue	Wed	Thu	Fri	Sat	Sun
Feb 28	Mar 1	2	3	4	5	6
7	8	9	10 2:00 PM Novo Evento (Braga)	11 5:00 PM Outro Evento 1 (Porto) 5:30 PM Outro Evento 2 (Lisboa)	12	13
14	15	16	17 2:00 PM Novo Evento (Braga)	18	19 Evento Todo o Dia (Braga)	20
21	22	23	24 2:00 PM Novo Evento (Braga)	25	26	27
28	29	30	31 2:00 PM Novo Evento (Braga)	Apr 1	2	3
4	5	6	7 2:00 PM Novo Evento (Braga)	8	9	10

Figura 3.9: Calendário de eventos na vista mensal na Microsoft Exchange

**Create an Event** Save As Draft Save & Publish Preview

**STEP 1: ADD EVENT TITLE**

Novo Evento

**STEP 2: ADD TICKET INFORMATION**

Create a Ticket You must create at least one ticket for your event to be published.

**STEP 3: ADD EVENT DETAILS**

Font family Font size

Detalhes do evento...

**LOGO** EDIT REMOVE

Upload the logo for your event:

branco/Desktop/Eventicon.png Procurar...

Must be JPG, GIF, or PNG smaller than 100Kb. Dimensions are limited to 450 x 200 px. Images larger than this will be resized.

Upload

**STEP 4: ADD WHEN**

**EVENT STARTS**

06/01/2012 at 01:00 PM

Hide start date on registration page?

**EVENT ENDS**

06/01/2012 at 04:00 PM

Hide end date on registration page?

**TIME ZONE**

(GMT) Greenwich Mean Time: Dublin, ...

[view US time zones](#)

**EVENT REPEATS?**

Yes, this event repeats

Figura 3.10: Parte do formulário de criação de um evento no Eventbrite

**Create Ticket** X

Create tickets for each price, sales date, or other options.

\* Required Field

Ticket Name: \* Grátis

Examples: Member, Non-member, Student, Early Bird

Price: \* 0.00 U.S. Dollars (\$)

Free

Donation Format (Attendee can specify the payment amount)

Quantity Available: \* 20

Advanced Options: [Show](#)

Cancel Save

Figura 3.11: Formulário de criação de um ticket no Eventbrite

**Novo Evento**  
Friday, June 1, 2012 from 1:00 PM to 4:00 PM (GMT)  
Braga, Portugal

**Ticket Information**

TYPE	REMAINING	END	QUANTITY
Grátis	19 tickets	Jun 1, 2012	Free <input type="text" value="1"/>

**Register**

**When & Where**

Rua X  
4710 Braga  
Portugal  
Friday, June 1, 2012 from 1:00 PM to 4:00 PM (GMT)

Add to my calendar

**Hosted By**

**Organização Y**

Contact the Host

Descrição da organização...

View other Organização Y events

Subscribe to receive notifications of future events by this host

Share this! | Email | Share | Tweet | Like | Sign Up to see what your friends like.

**Event Details**

Detalhes do evento...

Figura 3.12: Página web de registo num evento acabado de criar no Eventbrite

### 3.2.3 Eventbrite

A ferramenta de gestão de eventos *on-line*, Eventbrite, ao contrário do Google Calendar e da Microsoft Exchange, não fornece um calendário para organizar os eventos criados. No entanto, permite exportar os eventos para o Microsoft Exchange, Google Calendar, Yahoo! Calendar e, ainda, para o iCal Calendar. Também permite a criação rápida de uma página *web*, com bastante grau de customização, com a informação sobre o evento criado, para que os interessados na participação do evento se possam inscrever nele.

Para criar um evento no Eventbrite, é fornecido um grande formulário com os passos necessários. Na Figura 3.10 observa-se parte do formulário de criação de um novo evento. O primeiro passo é atribuir um título ao evento e de seguida deve-se criar um *ticket* para possibilitar a participação dos interessados no evento.

Na Figura 3.11 destaca-se o formulário de criação de um *ticket*, onde constam os campos nome do *ticket*, preço, quantidade existente e ainda há opções avançadas que não vão ser analisadas. Neste caso, atribuiu-se o nome *Grátis*, uma vez que se optou por disponibilizar vinte *tickets* de graça, para interessados em participar no evento.

Após a criação de um *ticket*, pode-se atribuir uma descrição do evento, carregar um ícone para o evento, definir a data de início e de fim (juntamente

com o respectivo fuso horário) e especificar recorrência, caso o evento se repita mais que uma vez.

Outros passos que não estão apresentados na Figura 3.11 são a especificação da morada (ou local) onde vai ocorrer o evento, a identificação do(s) organizador(es) do mesmo e respectiva descrição, a indicação de um tema de cores para decorar a página *web* de registo no evento, a especificação das credenciais de acesso a este, ou seja, definir se ele vai ser público ou restrito a certas pessoas e, por fim, declarar opções para a página do evento, como por exemplo, apresentar o número de *tickets* restantes.

Após todo este processo, já é possível publicar o evento e o resultado obtido é uma página *web* para o público interessado se poder registar nele, como se pode observar na Figura 3.12 com o evento acabado de criar.

### 3.2.4 Criação de eventos no Facebook

Sendo o Facebook uma rede social à escala global, esta proporciona várias funcionalidades. Uma das funcionalidades mais utilizadas no Facebook é a criação de eventos para combinar reuniões importantes, jantares, festas de convívio, etc.. Contudo, tal como o Eventbrite, o Facebook não fornece um calendário para gerir eventos, mas também permite exportá-los para outros programas, como o iCal Calendar, a Microsoft Exchange e o Google Calendar. Uma outra desvantagem do Facebook é a impossibilidade de criar eventos recorrentes e, desta forma, o utilizador tem que criar um novo sempre que este ocorrer.

Na Figura 3.13 pode-se visualizar um formulário bastante simples para criar um evento, onde se destacam os seguintes campos informativos, típicos de qualquer evento: datas de início e de fim, título, local, descrição e ícone. Também há a possibilidade de tornar o evento público ou privado. Caso seja privado, pode-se convidar amigos, permitir que estes convidem outros amigos, e assim sucessivamente.

Quando se acaba de criar o evento, os convidados são automaticamente notificados e, por isso, podem responder ao convite, escolhendo uma das seguintes três opções existentes na Figura 3.14: aceitar, talvez aceitar ou rejeitar convite. Também é possível deixar uma mensagem, anexada ao evento, para agradecer o convite ou justificar a não possibilidade de comparecer no evento ou outra coisa qualquer.

## 3.3 Descrição da Aplicação

Como já foi dito no início deste capítulo, vai ser desenvolvido um projecto de software no âmbito empresarial, mais concretamente uma solução de *software web-based*, sugerida pela empresa Ubisign<sup>8</sup>, com o principal objectivo

---

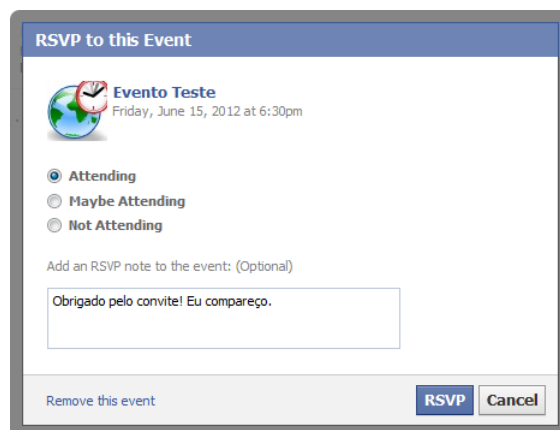
<sup>8</sup><http://www.ubisign.com>



The screenshot shows the 'Criar um evento' (Create an event) form on Facebook. On the left, there is a globe icon with a clock overlay and a button labeled '+ Alterar foto do evento'. The main form area is titled 'Criar um evento' and contains the following fields and options:

- Quando?**: Date field set to 6/15/2012 and time field set to 18:30.
- Hora de fim:**: End time field set to 20:30.
- O que é que estás a planear?**: Text input field containing 'Evento Teste'.
- Onde?**: Text input field containing 'Portugal'.
- Mais informação?**: Text input field containing '...'.
- Quem foi convidado?**: A button labeled 'Seleccionar mais convidados (1)'.
- Privacy and visibility options:**
  - Qualquer pessoa pode ver e responder (evento público)
  - Os convidados podem convidar amigos
  - Exibir a lista de convidados na página do evento
- Submit button:** 'Criar evento'.

Figura 3.13: Formulário de criação de um novo evento no Facebook



The screenshot shows the 'RSVP to this Event' dialog box on Facebook. It contains the following information and options:

- Title:** 'RSVP to this Event'.
- Event Details:** 'Evento Teste' with a globe icon and 'Friday, June 15, 2012 at 6:30pm'.
- Response Options:**
  - Attending
  - Maybe Attending
  - Not Attending
- RSVP Note:** A text input field with the note 'Obrigado pelo convite! Eu compareço.'
- Buttons:** 'Remove this event', 'RSVP', and 'Cancel'.

Figura 3.14: Formulário do Facebook para responder a um convite de participação num evento

de apresentar, aos utentes de painéis de *digital signage*, informação sobre eventos disponíveis no serviço Google Calendar. Daí que, utilizando como fonte de informação a API Google Calendar, pretende-se que se desenvolva uma aplicação que exiba informação sobre eventos planeados ou em curso para um determinado local (tipicamente, o espaço ou a zona geográfica onde o painel de *digital signage* está a operar). Também deverá permitir que o utilizador especifique o aspecto visual dos conteúdos apresentados, com base em formulários próprios.

A aplicação deve consultar o Google Calendar e obter um conjunto de eventos que satisfaçam os critérios geográficos e cronológicos especificados na consulta. O conteúdo da informação a apresentar para cada evento deve incluir, pelo menos: o nome do evento, a(s) data(s)/horário de ocorrência, o local, a descrição, e uma fotografia/ícone associado ao evento. Caso haja mais do que um evento a exibir, a visualização deve apresentar iterativamente os eventos ordenados cronologicamente. A visualização deve ser concebida de modo a suportar os formatos de apresentação mais comuns em *digital signage*: ecrã inteiro (visualização Full), coluna (visualização Column), e barra horizontal (visualização Bar). Para além destes três formatos de apresentação frequentes, a aplicação também vai ter que permitir a visualização dos eventos num formato estandardizado, como é o caso do XML.

### 3.3.1 Visualização em ecrã inteiro (Full)

Este tipo de visualização é a que vai apresentar a informação dos eventos com maior detalhe, uma vez que vai ocupar a totalidade do ecrã do painel de *digital signage*. É uma visualização muito propícia a apresentar eventos num modo *slide-show*, ou seja, quando uma visualização Full é carregada no *browser*, esta itera sobre os eventos que tem para exibir, segundo um intervalo de tempo especificado na configuração. Enquanto a visualização estiver carregada no *browser*, esta vai apresentar a mesma sequência de eventos de forma repetida (encontra-se em *loop*).

Uma vez que este tipo de visualização ocupa o ecrã inteiro, é normal que se queira incluir, muitas vezes, uma imagem associada ao evento e descrições adicionais de forma a aproveitar o espaço existente no ecrã. No entanto, é muito importante esclarecer que, como se trata de *digital signage*, não é conveniente utilizar um excesso de informação no ecrã, pois, segundo estudos feitos em [Huang et al., 2008], as pessoas olham, em geral, muito brevemente para os ecrãs em locais públicos (na ordem dos 1 ou 2 segundos).

### 3.3.2 Visualização em coluna (Column)

Ao contrário da anterior, esta visualização exibe a informação sobre eventos de forma mais resumida, dado que o espaço disponível é menor (normalmente trata-se de uma coluna localizada num dos lados do ecrã). Além disso, sendo



o espaço horizontal bastante menor do que o espaço vertical, esta visualização deve permitir exibir informação sobre mais do que um evento ao mesmo tempo, em função de um parâmetro especificado na configuração. Neste caso, ao contrário da visualização Full, não há a necessidade de temporização da iteração, isto é, enquanto a visualização estiver carregada, exibe constantemente os mesmos eventos.

Visto que este tipo de visualização exige um espaço reduzido para vários eventos, é normal que cada evento seja apresentado com menos informação. Assim sendo, se a coluna contiver poucos eventos, cada um deve apresentar, no máximo, um ícone simples para ser facilmente identificado pelas pessoas que visualizam o painel digital. Caso a coluna contenha muitos eventos, a utilização de ícone está fora de questão, sendo suficiente utilizar informação básica sobre o evento, como por exemplo, o título, local e data/hora.

### 3.3.3 Visualização em barra horizontal (Bar)

Este tipo de visualização, sendo a mais restrita de todas, exibe a informação sobre eventos, o mais resumidamente possível, dado que o espaço disponível é muito limitado. Nesta visualização, só se vai poder exibir um evento de cada vez, iterando sobre os eventos que tem para exibir, segundo um intervalo de tempo especificado na configuração.

Dentro do tipo de visualização em barra horizontal, o deslizamento de texto sobre eventos vai poder ser feito na horizontal ou na vertical. Caso este seja feito na horizontal há a possibilidade de que cada evento apresente bastante mais informação (à base de texto), uma vez que a informação é mostrada de forma gradual, segundo a velocidade de deslizamento especificada na configuração. No entanto, caso o deslizamento seja feito na vertical, como cada evento é apresentado num espaço muito reduzido, segundo um intervalo de tempo especificado na configuração, é natural que cada um apresente muito pouca informação.

### 3.3.4 Visualização em XML

Este tipo de visualização, como o próprio nome indica, trata-se de XML puro. Não vai ter qualquer tipo de informação decorativa, expondo apenas os eventos retornados pela *query* feita ao Google Calendar, na forma de um documento XML.

A informação apresentada em XML deve ter como referência o formato iCalendar (um formato *standard* usado em ficheiros para armazenar informação sobre eventos) para que aplicações externas tenham a facilidade de importar o XML. Por isso, este tipo de visualização é útil para quando se pretende que o aspecto decorativo fique totalmente na responsabilidade de uma aplicação externa, e para quando se pretenda realizar *queries* ao Google Calendar, usando como intermédio a aplicação *web* a ser desenvolvida.

### 3.3.5 Parâmetros de configuração

Como já foi dito, a aplicação vai permitir gerar quatro tipos de visualizações e o utilizador vai poder configurá-las através de parâmetros de configuração da componente visual (com a excepção da visualização XML), da componente comportamental (com a excepção da visualização XML) e da componente de filtragem.

Quanto aos parâmetros da componente visual, é de realçar que vai ser possível alterar a interface para cada uma das visualizações em função de várias configurações, como por exemplo, alterar a cor de fundo, de fonte, utilizar tamanhos ou estilos diferentes de fonte para diferentes componentes de um evento (título, data, local, etc.), de entre outras coisas que vão ser definidas na fase de especificação da aplicação (próximo capítulo).

No que diz respeito aos parâmetros da componente comportamental, o utilizador vai poder configurar o comportamento das visualizações e da aplicação, como por exemplo, na visualização Full poder-se-á especificar o tempo durante o qual cada evento é apresentado, o número de eventos apresentados, de entre outras coisas que vão ser especificadas no próximo capítulo.

Por fim, quanto aos parâmetros da componente de filtragem, o utilizador poderá efectuar *queries*, de maior ou menor complexidade, ao serviço Google Calendar. Essas *queries* serão sobretudo de cariz temporal ou espacial, mas também poderão surgir outro tipo de *queries* no processo de especificação da aplicação (próximo capítulo).

### 3.3.6 Formulário de configurações

Como as configurações dos quatro tipos de visualizações se vão basear em vários parâmetros especificados pelo utilizador, torna-se útil a possibilidade de poder efectuar configurações com a ajuda de um formulário *web*. De preferência, o formulário deverá ter mecanismos *user-friendly* para não confundir o utilizador com excesso de informação. Por exemplo, no que diz respeito a cores de fonte ou de fundo de um evento, é importante que a interface do formulário apresente uma janela com um lote de cores frequentemente usadas. Para além disso, há muitos outros aspectos que poderão ser tidos em conta durante a fase de especificação mais detalhada da aplicação.

### 3.3.7 Persistência de configurações e configurações *default*

Uma solução de grande utilidade para o utilizador é a possibilidade de poder armazenar as suas configurações numa base de dados, para posteriormente as poder usar com mais facilidade. Também é muito importante que outras operações, como a actualização ou a remoção de configurações, sejam permitidas.

As configurações *default* vão consistir num conjunto de configurações pré-definidas que vêm armazenadas, desde início, na base de dados da aplicação.

Estas vão permitir ao utilizador poder seleccionar aquelas que achar mais úteis, aproveitá-las para efectuar algumas alterações convenientes e, por fim, armazená-las na base de dados, se for caso disso.

### 3.3.8 Integração da aplicação no serviço Ubisign.com

Como a aplicação a desenvolver vai ser *web-based*, a integração com o serviço Ubisign.com vai ser bastante simples, uma vez que a aplicação vai poder receber parâmetros de configuração, para os quatro tipos de visualização, através de *query strings*.

No que diz respeito ao processo de autenticação, o serviço Ubisign.com apenas terá que utilizar na *query string* um *token* GUID (Globally Unique Identifier) que corresponde a uma *string* em hexadecimal de 32 caracteres, identificando de forma inequívoca o utilizador da aplicação. Para que tal seja possível, é necessário que a aplicação armazene, numa base de dados, os *tokens* dos utilizadores da aplicação. É também de salientar que cada *token* irá estar associado a uma conta OpenID pertencente à Google<sup>9</sup>.

Por fim, é importante deixar claro que a aplicação não vai depender do serviço Ubisign.com para a sua utilização. Outras aplicações externas também vão ter o trabalho facilitado no que diz respeito à utilização da aplicação como *plugin*, visto que, como já foi dito, esta é *web-based* e as configurações e o *token* de autenticação são enviados via *query string*.

A aplicação também vai poder ser utilizada de forma autónoma, uma vez que, por exemplo, a visualização Full poderá ser, por si só, bastante apelativa para apresentar eventos em *loop* e, além disso, todas as configurações vão poder ser submetidas através de um formulário *web*, o que facilita a introdução de novas configurações. Como já foi dito, também vão existir configurações pré-definidas que ajudam o utilizador a especificar a interface e o seu comportamento.

### 3.3.9 Outros aspectos importantes

Para além dos aspectos principais, referidos nas anteriores subsecções, outros aspectos como a manutenção de estado, a localização (cultura regional do *browser*) e a abstracção da fonte de informação, também são muito importantes numa aplicação que ambicione ter uma utilização à escala mundial, como é o caso do serviço Ubisign.com.

Um problema que se vai enfrentar, uma vez que a aplicação *web* vai ser embutida na camada de visualização do serviço Ubisign.com, é o facto de se utilizar um navegador *web* de código aberto, o WPF Chromium WebBrow-

---

<sup>9</sup>Para mais informação, consultar a subsecção sobre autenticação/autorização da secção anterior: Ferramentas *on-line* de Gestão de Eventos.

ser<sup>10</sup>, como forma de comunicar com a aplicação.

De seguida, estes aspectos vão ser abordados com maior detalhe.

### Manutenção de estado

Uma vez que a aplicação vai servir como uma espécie de *plugin* para ser integrado no serviço Ubsign.com (e não só, visto que a aplicação vai poder ser utilizada de forma autónoma), é necessário que esta faça um controlo rigoroso do estado das iterações entre cada pedido à aplicação. Com isto pretende-se que a aplicação, por cada pedido que é feito, retorne uma visualização com os próximos N eventos, para que a aplicação externa controle o momento de apresentar conjuntos de eventos distintos.

Vai ser necessário utilizar a sessão de forma a guardar o estado das iterações entre cada pedido à aplicação. No entanto, é preciso ter em conta (para uma dada *query*) o comportamento do Google Calendar na forma como retorna os eventos. Mas como o Google Calendar permite retornar todos os resultados da *query* de uma só vez, a aplicação terá também que armazenar em sessão os eventos, para que em cada pedido seja retornada uma visualização com diferentes conjuntos de eventos. De certa forma também não seria muito eficiente efectuar vários pedidos ao serviço Google Calendar, para uma dada *query*, visto que esses pedidos exigem o recurso à Internet e, por isso, são bastante caros.

### Localização

Sabendo que a aplicação vai ter que suportar uma utilização à escala global, esta deverá ser capaz de exibir datas/horas de um evento no formato correcto, segundo as definições regionais do cliente que faz o pedido. Por exemplo, na Europa a data tem o formato DD/MM/AAAA (dia/mês/ano), enquanto que nos Estados Unidos o formato é MM/DD/AAAA (mês/dia/ano).

### Abstracção da fonte de informação

A aplicação deve, ao nível da camada de negócio, ser independente do serviço de gestão de eventos utilizado (ou fonte de informação). Isto implica que a arquitectura da aplicação contenha abstracções que permitam facilmente, no futuro, estender a aplicação para consumir eventos da Microsoft Exchange, Facebook, Eventbrite, etc.. Para tal, deve ser especificado um modelo de dados sobre eventos que seja o mais genérico possível e potencialmente compatível com outras fontes de informação. Neste sentido, o formato iCalendar é uma boa referência, no entanto é conveniente definir abstracções em função das necessidades da aplicação.

---

<sup>10</sup><http://chriscavanagh.wordpress.com/2009/08/25/wpf-chromium-webbrowser-source-code/>

### WPF Chromium WebBrowser

As visualizações Full, Column e Bar devem ser suportadas pelo WPF Chromium WebBrowser (navegador *web* de código aberto), uma vez que é o *browser* que a Ubisign utiliza, de forma embutida, nos seus serviços, com o fim de processar várias aplicações *web* num só painel de *digital signage*. No entanto, é preciso ter em conta todo o tipo de limitações que esse *browser* tem em termos de HTML, JavaScript e CSS (Cascading Style Sheets), para que as visualizações sejam apresentadas de forma fluída e sem qualquer tipo de problema.

## 3.4 Modelo do Domínio

Nesta secção vai-se analisar o domínio do problema através de um diagrama designado por Modelo do Domínio. Assim, na Figura 3.15 encontra-se o Modelo do Domínio para a aplicação que se pretende elaborar, e onde constam as entidades principais do domínio do problema, para além das relações entre elas.

Fazendo uma análise do Modelo do Domínio da Figura 3.15, verifica-se que a entidade Utilizador deve, antes de tudo, autenticar-se com a conta OpenID da Google e obter a autorização para o acesso ao serviço Google Calendar. Para tal, é necessário que a aplicação interaja com a Google, tanto no processo de autenticação (protocolo OpenID), como no processo de autorização de acesso ao serviço Google Calendar (protocolo OAuth).

Uma vez autenticado e autorizado, o Utilizador, recorrendo aos campos existentes na entidade Formulário, pode introduzir Parâmetros de Configuração. É importante referir que, apesar de no Modelo do Domínio da Figura 3.15 apresentar apenas uma entidade genérica designada por Formulário, essa mesma entidade é estendida pelas entidades subentendidas Formulário para a Visualização Full, Column, Bar e XML<sup>11</sup>. Daí extrai-se que vão existir quatro formulários de configuração, com bastantes semelhanças e algumas diferenças, para os quatro tipos de visualização existentes.

Observando a entidade Parâmetros de Configuração, facilmente se verifica que esta é estendida por três entidades: “Params. Componente Visual”, “Params. Componente Comportamental” e “Params. Componente de Filtragem”. Por isso, é fácil deduzir que a partir do nome de cada uma delas se pode saber que se trata de parâmetros da componente visual, comportamental e de filtragem. A explicação detalhada destas três componentes é feita na anterior secção, Descrição da Aplicação.

Concluindo o preenchimento do Formulário por parte do Utilizador, este pode e deve, antes de mais, clicar no Botão de Visualização<sup>12</sup>, existente no

<sup>11</sup>Estas quatro entidades não foram acrescentadas ao diagrama por motivos de espaço e de compreensão do mesmo.

<sup>12</sup>É de notar que, por uma questão de simplificação do Modelo do Domínio da Figura

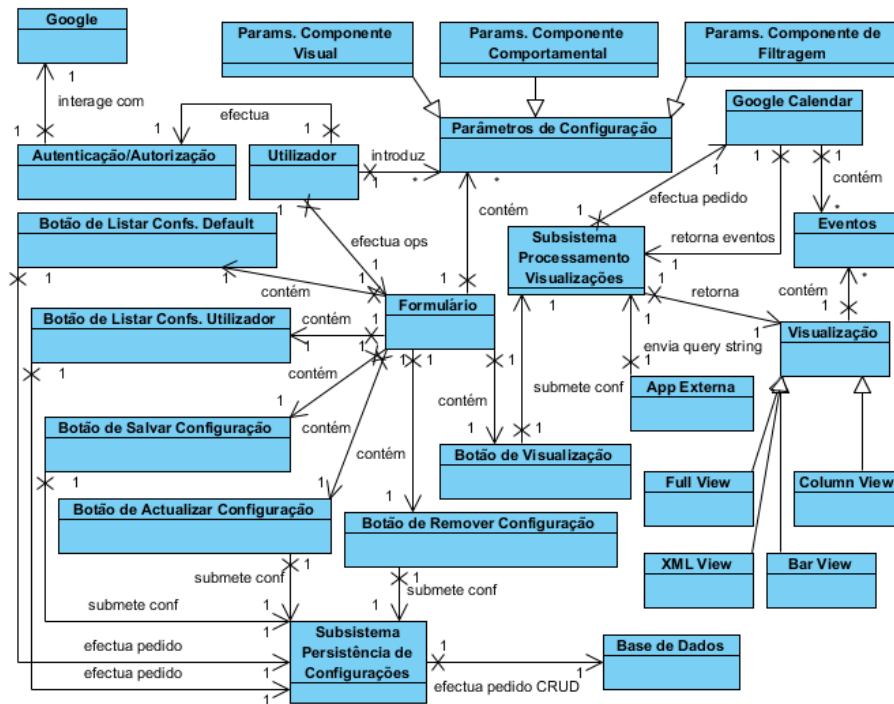


Figura 3.15: *Modelo do Domínio*

Formulário, para a aplicação apresentar uma visualização de acordo com o que o Utilizador especificou. Assim sendo, a entidade “Subsistema Processamento Visualizações”, que corresponde a uma parte do sistema que processa configurações, recebe os parâmetros de configuração especificados pelo Utilizador e efectua um pedido à entidade Google Calendar para esta retornar Eventos em função dos “Params. Componente de Filtragem”. Por fim, a entidade “Subsistema Processamento Visualizações” retorna uma Visualização Full, Column, Bar ou XML, consoante o Formulário que o Utilizador submeteu.

Ainda quanto à entidade “Subsistema Processamento Visualizações”, é importante dizer que a Visualização retornada é “moldada”, em termos visuais, segundo os “Params. Componente Visual” especificados pelo Utilizador. Quanto ao comportamento da visualização, este é definido segundo os “Params. Componente Comportamental” também definidos pelo Utilizador.

Este processo todo, descrito desde o início desta secção até aqui, também pode ser feito através de uma “App Externa” (uma aplicação externa que pode corresponder ao serviço UbiSign.com ou a outra qualquer aplicação), enviando todos os parâmetros de configuração e um *token* de autenticação. Como resposta, a aplicação retorna a Visualização desejada e a “App

3.15, a acção de clicar nos botões existentes no Formulário, por parte da entidade Utilizador, é apresentada por “A entidade Utilizador efectua operações na entidade Formulário”.

Externa” pode fazer o que for pretendido com essa Visualização (por exemplo, melhorar esteticamente a Visualização ou enquadrar a Visualização num *layout* da “App Externa”).

Para além da operação comum de clicar no Botão de Visualização, ainda existem outras operações muito importantes que estão relacionadas com a entidade Base de Dados. Deste modo, o Utilizador pode efectuar operações básicas, disponibilizadas pelas bases de dados relacionais, como as de listagem, inserção, actualização e remoção de configurações (operações CRUD<sup>13</sup>).

Quando o Utilizador clica no “Botão de Listar Confs. Utilizador” é efectuado um pedido à entidade “Subsistema Persistência de Configurações” para listar configurações. Essa entidade, que corresponde a uma parte do sistema que interage directamente com um motor de Base de Dados relacional (esse subsistema irá fazer parte da camada de dados da aplicação), efectua uma *query* para a entidade Base de Dados devolver a lista das configurações pertencentes ao Utilizador (operação SQL de SELECT).

Se o Utilizador pretende persistir os Parâmetros de Configuração preenchidos no Formulário, basta clicar no “Botão de Salvar Configuração”. Assim, os Parâmetros de Configuração são submetidos à entidade “Subsistema Persistência de Configurações” e esta, por sua vez, efectua o pedido de inserção de nova configuração de Utilizador à entidade Base de Dados (operação SQL de INSERT).

Quanto à operação de actualização de uma configuração existente na Base de Dados e pertencente ao Utilizador, é necessário, antes de mais nada, clicar no “Botão de Listar Confs. Utilizador” para que o Utilizador possa seleccionar a configuração a actualizar. Depois do Utilizador efectuar a selecção pretendida, tem que haver um mecanismo para preencher os campos do Formulário com a informação pertencente à configuração seleccionada. Neste momento, o Utilizador já pode efectuar as alterações nos campos que achar conveniente e, por fim, submeter as alterações, clicando no “Botão de Actualizar Configuração”. A configuração actualizada é submetida para a entidade “Subsistema Persistência de Configurações” e esta, por sua vez, efectua o pedido de actualização de configuração à entidade Base de Dados (operação SQL de UPDATE).

No que diz respeito à operação de remover uma configuração de Utilizador, tal como na operação de actualização de configuração, é necessário clicar no “Botão de Listar Confs. Utilizador” para o Utilizador seleccionar a configuração a remover. Uma vez feita essa operação, o Utilizador já pode clicar no “Botão de Remover Configuração” e, como resposta, o identificador da configuração é submetido para a entidade “Subsistema Persistência de Configurações”. Finalmente, a entidade “Subsistema Persistência de Configurações” efectua o pedido de remoção de configuração à entidade Base de Dados (operação SQL de DELETE).

---

<sup>13</sup>CRUD – Create, Remove, Update e Delete.

Para finalizar a análise do Modelo do Domínio da Figura 3.15, falta apenas analisar a entidade “Botão de Listar Confs. Default” que efectua uma operação semelhante ao “Botão de Listar Confs. Utilizador”. Assim, quando o Utilizador clica nesse botão, é efectuado um pedido de listagem de configurações *default* da aplicação. Esse pedido é delegado pela entidade “Subsistema Persistência de Configurações” para a entidade Base de Dados (operação SQL de SELECT). Logo que esteja a listagem disponível na interface da aplicação, o Utilizador pode seleccionar a configuração *default* que desejar para poder efectuar operações de visualização e inserção, correspondentes às entidades “Botão de Visualização” e “Botão de Salvar Configuração”.

Tal como na operação de actualização de configuração de Utilizador, neste último caso (listar configurações *default*) também tem que haver um mecanismo para preencher os campos do Formulário com a informação pertencente à configuração seleccionada. Assim, o Utilizador pode visualizar, salvar configuração e, até, alterar os Parâmetros de Configuração do Formulário que achar conveniente, para persistir uma nova configuração que obteve com base em uma *default*.

### 3.5 Análise de Requisitos

Nesta secção vão ser analisados os requisitos (tarefas) que a aplicação terá de cumprir, de forma a que esta disponibilize todas as funcionalidades essenciais para o seu bom funcionamento.

Os requisitos funcionais da aplicação são os seguintes:

- Permitir ao utilizador autenticar-se e obter a autorização de acesso ao serviço Google Calendar (recorrendo ao protocolo híbrido OpenID/OAuth);
- Suportar quatro tipos de visualização para painéis de *digital signage*: em ecrã inteiro (visualização Full), em coluna (visualização Column), em barra horizontal (visualização Bar) e em XML;
- Definir todos os parâmetros de configuração necessários para cada um dos quatro tipos de visualização (Full, Column, Bar e XML);
- Incluir suporte total a parâmetros de configuração da componente visual, que diz respeito à personalização do aspecto da visualização (não incluir a visualização XML, uma vez que o aspecto visual é sempre o mesmo);
- Incluir suporte total a parâmetros de configuração da componente comportamental, que engloba todo o comportamento definido para as visualizações (Full, Column e Bar) e certos comportamentos da aplicação (gestão de sessão);



- Incluir suporte total a parâmetros de configuração da componente de filtragem, que devem ser parâmetros de ordem temporal ou espacial para efectuar pedidos de eventos ao Google Calendar. Cada um dos quatro tipos de visualizações (Full, Column, Bar e XML) vão ter que suportar parâmetros de filtragem;
- Disponibilizar um formulário *user-friendly* para que o utilizador possa, facilmente, preencher os campos relativos a parâmetros de configuração;
- Permitir a persistência de configurações recorrendo a uma base de dados relacional;
- Permitir operações de actualização e de remoção de configurações;
- Possibilitar que a aplicação tenha um conjunto de configurações pré-definidas (configurações *default*) para auxiliar o utilizador;
- Permitir que aplicações externas (principalmente o serviço Ubsign.com) utilizem a aplicação a elaborar e possam autenticar-se, obter autorização de acesso ao Google Calendar e enviar parâmetros de configuração através de *query strings*;
- Incluir manutenção de estado para os eventos que são retornados pelo serviço Google Calendar.

No que diz respeito aos requisitos não funcionais da aplicação a desenvolver, importa destacar os seguintes:

- Suportar a localização das datas/horas dos eventos apresentadas nas visualizações Full, Column e Bar, ou seja, a aplicação deverá ser capaz de exibir datas/horas de um evento no formato correcto, segundo as definições regionais do cliente que faz o pedido;
- Implementar abstrações ao nível da camada de negócio, de forma a que seja possível, no futuro, estender a aplicação para consumir eventos da Microsoft Exchange, Facebook, Eventbrite, etc.;
- Durante a implementação das visualizações Full, Column e Bar, é essencial que haja compatibilidade com o WPF Chromium WebBrowser, uma vez que o serviço Ubsign.com vai utilizar esse *browser* para embutir a aplicação a elaborar.

## Capítulo 4

# A Aplicação DSEventApp

Uma vez terminada a especificação do problema (capítulo 3), ou seja, tudo o que se pretende com o projecto de *software* a desenvolver (aplicação *web* sobre apresentação de eventos em painéis de *digital signage*), vai-se dar início à solução do projecto.

Neste capítulo, após uma introdução mais completa, vai-se começar por detalhar os requisitos da aplicação a elaborar, no que diz respeito à quantidade de informação sobre eventos para as visualizações (Full, Column, Bar e XML). Também vão ser detalhados todos os parâmetros de configuração que cada tipo de visualização suporta.

De seguida, inicia-se a fase de especificação da aplicação, isto é, apresentar-se-ão os vários diagramas e esboços que auxiliam o desenvolvimento da aplicação. Também é importante sublinhar que todas as boas práticas de desenvolvimento de *software* vão ser cumpridas (separação entre a camada de visualização, lógica aplicacional e da camada de base de dados).

Na última secção deste capítulo, vai ser descrito todo o processo de desenvolvimento e dar-se-á a conhecer todas as decisões relevantes tomadas. No final, vai-se demonstrar o funcionamento da aplicação.

### 4.1 Introdução

Ao longo deste capítulo vai ser descrita a solução a desenvolver, isto é, tudo o que está relacionado com a aplicação *web* que apresenta informação sobre eventos disponíveis no serviço Google Calendar. É de destacar que a aplicação será desenvolvida recorrendo à .NET Framework 3.5, utilizando a tecnologia ASP.NET MVC 2, juntamente com a linguagem de programação C#.

Como este capítulo vai ser dedicado exclusivamente à aplicação a desenvolver, frequentemente esta será designada por DSEventApp (Digital Signage Event Application). É de notar que o seu desígnio não está relacionado com o Google Calendar, uma vez que, no futuro, esta irá ser estendida para outros serviços de gestão de eventos, como por exemplo: Microsoft Exchange,

Eventbrite, Facebook, etc.. Além disso, a sua lógica de negócio vai estar preparada para que o processo de extensão seja facilitado.

Num primeiro momento, serão detalhados os requisitos da DSEventApp no que diz respeito à quantidade de informação sobre eventos para as visualizações em ecrã inteiro (Full), em coluna (Column), em barra horizontal (Bar) e em XML. Também vão ser identificados e detalhados todos os parâmetros de configuração que cada tipo de visualização vai suportar.

De seguida, vai ser feita a especificação da DSEventApp ao nível de três camadas muito conhecidas no mundo do desenvolvimento de *software*: camada da interface do utilizador, camada da lógica de negócio e camada de acesso à base de dados.

Ao nível da camada de interface, é importante referir que esta é caracterizada pelas interações efectuadas pelo utilizador ao nível dos formulários de configuração e, também, pelas visualizações (Full, Column, Bar e XML) disponibilizadas pela DSEventApp, com base em parâmetros de configuração. Desta forma, nesta primeira fase do capítulo, vão ser apresentados alguns *mockups* (esboços) relacionados com os componentes existentes na interface do sistema. De seguida, para uma mais fácil compreensão das interações e comportamentos existentes nos formulários de configuração, vão ser apresentados dois diagramas de sequência de alto nível, que apresentam a sequência de acções que o utilizador pode efectuar (em resposta a essas acções, são apresentados os comportamentos do sistema).

No que diz respeito à camada de negócio, vai ser apresentado um diagrama de classes por cada *package* especificado, sendo que cada um é composto por várias classes e respectivas interações, e os seus métodos vão traduzir as funcionalidades que a aplicação terá de suportar.

Por fim, relativamente à camada de base de dados, como se vai recorrer à *framework* NHibernate para .NET (ferramenta ORM – Object Relational Mapping – que faz o mapeamento entre classes C# e relações de uma base de dados), vão ser especificadas as entidades (ou classes persistentes) e as tabelas que dizem respeito à base de dados (diagrama ER). Também importa dizer que se utilizará o motor de base de dados SQL Server 2005, e as tabelas vão servir, essencialmente, para armazenar parâmetros de configuração e dados de autenticação/autorização de utilizadores da DSEventApp.

Na segunda e última secção deste capítulo será descrito todo o processo de implementação, tomando decisões e apresentando soluções para os problemas encontrados. Para terminar, vai ser apresentado um exemplo para cada visualização (Full, Column, Bar e XML) processada pela DSEventApp, juntamente com a respectiva *query string* de configuração.

## 4.2 Requisitos Detalhados

Nesta secção define-se a quantidade de informação sobre eventos que cada tipo de visualização disponibiliza e detalha-se todos os parâmetros que a DSEventApp consome para cada uma das visualizações.

### 4.2.1 Visualização em ecrã inteiro (Full)

Uma vez que esta visualização foi definida para ocupar o ecrã por inteiro, tem a vantagem de permitir apresentar a informação dos eventos com maior detalhe.

É de relembrar que enquanto estiver carregada no browser, a visualização itera sobre os eventos que tem para exibir, segundo um intervalo de tempo especificado na configuração. Deste modo, por cada iteração, é apresentado um evento de cada vez, possibilitando que a totalidade de espaço, de um dado ecrã de um painel de *digital signage* (que costuma ser de grandes dimensões), seja alocado exclusivamente para apresentar informação sobre um determinado evento.

Tendo em conta o grande espaço disponível, este tipo de visualização poderá exibir a seguinte informação sobre cada evento:

- Título do evento
- Local (opcional)
- Data de início (opcional)
- Hora de início (opcional)
- Data de fim (opcional)
- Hora de fim (opcional)
- Descrição do evento (opcional)
- Imagem associada ao evento (opcional)

Como requisito mínimo, cada evento deve ter um título e, pelo menos, mais alguma informação adicional, de modo a ocupar um ecrã na sua totalidade. Os restantes elementos informativos vão ser opcionais, ou seja, o utilizador é que irá definir, através de parâmetros de configuração, quais são aqueles que pretende que cada evento disponibilize.

Quanto à obrigatoriedade de apresentação dos títulos de eventos, vai haver apenas uma pequena exceção, que é quando se pretende apresentar eventos exclusivamente sob a forma de imagens. Assim, a informação disponibilizada é bastante flexível, uma vez que as imagens permitem um total controlo da informação, da componente estética e do *layout*, para cada evento em concreto.

### 4.2.2 Visualização em coluna (Column)

Este tipo de visualização é bastante restrito, na medida em que são apresentados vários eventos em coluna.

Deste modo, comparando com a visualização Full, há menos espaço, pelo facto de uma coluna ter sempre dimensões inferiores em relação a um ecrã inteiro, mas também porque, ao contrário da visualização Full, são apresentados vários eventos de uma só vez (o número de eventos apresentado é baseado num parâmetro especificado na configuração). Outro aspecto em que difere este tipo de visualização, é o facto de ser totalmente estática, uma vez que os eventos não são apresentados em *loop* segundo um intervalo de tempo pré-definido.

Considerando que o espaço horizontal é bastante menor que o espaço vertical, este tipo de visualização deve exibir a seguinte informação sobre cada evento:

- Título do evento
- Local (opcional)
- Data de início (opcional)
- Hora de início (opcional)
- Ícone/imagem associada ao evento (opcional)

Neste caso, tal como na visualização Full, o título do evento é obrigatório, a não ser que se pretenda apenas mostrar uma imagem associada a cada evento apresentado em coluna. Essa imagem poderá ter toda a informação de interesse que o utilizador queira disponibilizar.

É de notar que, ao contrário da visualização Full, a visualização Column não vai apresentar a data e hora de fim de um evento, nem nenhuma descrição associada. A imagem também deverá ter menos detalhe, uma vez que esta irá ser apresentada numa célula de uma coluna. Quanto às dimensões da imagem, esta pode até apresentar um tamanho muito considerável, pois a DSEventApp trata de adaptá-la ao tamanho da célula.

### 4.2.3 Visualização em barra horizontal (Bar)

A visualização Bar corresponde, normalmente, a um rodapé ou barra horizontal com deslizamento de texto sobre eventos. O deslizamento pode ser feito na horizontal (visualização Bar Scroll), proporcionando mais conteúdo sobre cada evento, ou na vertical. No entanto, a DSEventApp não vai considerar o deslizamento de texto na vertical, uma vez que irá adoptar um rodapé, com características idênticas, designado por *fade-in/fade-out*, que corresponde ao aparecimento e desaparecimento de texto, segundo um intervalo de tempo pré-definido (visualização Bar Fade).

### Deslizamento horizontal

Dadas as características deste estilo de barra horizontal, ultrapassa-se a grande limitação de espaço, devido ao facto de ir aparecendo, gradualmente, texto a uma certa velocidade, especificada pelo utilizador.

Em comparação com as visualizações Full e Column, a visualização Bar Scroll é mais dinâmica, tendo a desvantagem de a informação disponibilizada ser mais difícil de captar pelos utilizadores, mas, por outro lado, permite apresentar uma maior quantidade de informação de cada evento (na maioria das vezes superior à da visualização Column).

Este tipo de visualização deve exibir a seguinte informação sobre cada evento:

- Título do evento
- Local (opcional)
- Data de início (opcional)
- Hora de início (opcional)
- Data de fim (opcional)
- Hora de fim (opcional)
- Descrição do evento (opcional)

Comparada com a visualização Full, a Bar Scroll não fica muito atrás no que diz respeito à quantidade de elementos informativos apresentados, uma vez que só não apresenta uma imagem para cada evento. No entanto, há casos em que as imagens são fundamentais e aí a visualização Bar fica a perder em relação às visualizações Full e Column.

### Deslizamento vertical

Este tipo de deslizamento é apenas feito nas transições entre eventos e de forma a que pare, durante um certo intervalo de tempo, no momento em que a informação do evento está bem formatada e visível (tipos de rodapés informativos muito frequentemente usados em canais de televisão).

No entanto, para a DSEventApp foi escolhida a animação *fade-in/fade-out* na transição entre eventos. Este modo de apresentação em rodapé consiste no aparecimento/desaparecimento gradual da informação sobre cada evento, obedecendo a dois intervalos de tempo especificados na configuração (um para o aparecimento/desaparecimento e outro para o tempo durante o qual a informação do evento fica totalmente visível).

Esta visualização, sendo a mais restrita de todas, exhibe a informação o mais resumidamente possível, dado que o espaço disponível é muito limitado.

Tal como na visualização Full, só será exibido um evento de cada vez, havendo iteração sobre os eventos que tem para exibir, segundo um intervalo de tempo especificado na configuração.

A visualização Bar Fade deve exibir a seguinte informação sobre cada evento:

- Título do evento
- Local (opcional)
- Data de início (opcional)
- Hora de início (opcional)

Como se verifica, apenas o título, o local e a data/hora de início podem ser apresentados, dado o espaço muito reduzido existente. Também é de destacar que não faz sentido que haja uma descrição associada ao evento, uma vez que o título, muitas vezes, ocupa bastante espaço e já é suficientemente elucidativo sobre o mesmo.

#### 4.2.4 Visualização em XML

Considerar XML como um tipo de visualização pode ser estranho, mas faz sentido devido ao facto de ser enviada pela DSEventApp após um pedido HTTP GET e, por conseguinte, pertence à camada de visualização da aplicação.

Sendo esta uma visualização totalmente distinta das anteriores, uma vez que não tem o objectivo de ser apresentada directamente em painéis de *digital signage*, a quantidade de informação suportada é bastante superior. Por isso, será essencialmente utilizada para retornar os eventos em formato XML, resultantes de *queries* realizadas ao serviço Google Calendar. Quanto ao objectivo principal desta visualização, é o de poder ser consumida por qualquer outra aplicação dedicada a apresentar eventos ou com outra função qualquer relacionada com os eventos retornados.

Este tipo de visualização vai exibir a seguinte informação:

- URL do calendário ao qual foi feito o pedido de eventos
- Nome do calendário
- Fuso horário utilizado no calendário
- Serviço (neste caso corresponde ao Google Calendar)
- Número de eventos retornados
- Um identificador para cada evento

- Título de cada evento
- Local
- Data de início
- Hora de início
- Data de fim
- Hora de fim
- Descrição
- URL da imagem associada a cada evento

Como se pode verificar, a quantidade de informação fornecida é superior à das outras visualizações e os eventos retornados pelo Google Calendar são apresentados todos de uma só vez.

Por fim, é importante destacar que a informação que a visualização XML apresenta, baseia-se no formato estandardizado iCalendar.

#### 4.2.5 Parâmetros de Configuração

Como já foi dito, a DSEventApp recebe parâmetros de configuração, via *query string*, para poder processar as visualizações e devolvê-las da forma como o utilizador pretende. Para uma melhor compreensão, os parâmetros vão ser apresentados em tabelas com os campos: nome do parâmetro, tipo de dados (*string*, *inteiro* ou *date*), visualizações suportadas e descrição.

Como foi referido no capítulo 3, a DSEventApp suporta três tipos de parâmetros de configuração: da componente visual, de filtragem e comportamental. Ainda foram considerados alguns parâmetros adicionais onde consta um parâmetro com o *token* de autenticação/autorização, outro com o URL do calendário a ser utilizado e os restantes com a função de efectuar operações na base de dados, no que diz respeito às configurações *default* e do utilizador.

De seguida, vão ser apresentados e descritos todos os parâmetros de configuração para as visualizações existentes (Full, Column, Bar e XML).

##### Parâmetros da componente visual

Os parâmetros da componente visual têm o objectivo de proporcionar ao utilizador a modelação da interface apresentada pela DSEventApp, segundo as suas preferências, para as visualizações Full, Column e Bar.

Na Tabela 4.1 pode-se observar parâmetros de configuração da componente visual para as visualizações Full, Column e Bar. É de notar que o parâmetro *EventNameVisibility* não é utilizado na visualização Bar porque



o título é um elemento obrigatório e não é necessário especificar nenhum tipo de alinhamento.

A Tabela 4.2 apresenta todos os parâmetros (da componente visual) exclusivos para as visualizações Full e Bar Scroll. Esses parâmetros dizem respeito à visibilidade/alinhamento da data de término dos eventos e da sua descrição, sendo que estes elementos informativos não constam nas visualizações Column e Bar Fade.

Relativamente aos parâmetros descritos na Tabela 4.3, pode-se dizer que são exclusivamente destinados à visualização Column e que correspondem às configurações visuais da informação sobre eventos apresentados nas células pares da coluna. Por isso, por uma questão de distinção, estes parâmetros têm, no geral, o prefixo “Sec”.

É de destacar que existe um parâmetro adicional designado por `EventIconRelativeLocalization` (exclusivo para a visualização Column) que é utilizado com o fim de especificar a localização do ícone relativamente ao título, local e data de início dos eventos (ver descrição mais detalhada deste parâmetro na Tabela 4.3).

### Parâmetros da componente de filtragem

Os parâmetros da componente de filtragem têm como função essencial disponibilizar ao utilizador a possibilidade de efectuar *queries* ao serviço Google Calendar e este retorna os eventos segundo os parâmetros especificados na *query* (por exemplo: obter eventos que apenas dizem respeito a uma data específica).

Como se pode ver na Tabela 4.4, foram especificados cinco parâmetros, de entre os quais dois servem para delimitar o período temporal dos eventos (`FromDate` e `ToDate`), um para definir um local comum para todos (`Where`), outro para obter todos os eventos com um dado título (`EventName`) e o último (`FullTextMatching`), mais poderoso, com a função de filtrar eventos, através da procura de um dado pedaço de texto existente nestes.

Como é de esperar, estes parâmetros são disponibilizados para todos os tipos de visualização existentes, inclusive a XML.

### Parâmetros da componente comportamental

Os parâmetros da componente comportamental têm a função de permitir ao utilizador especificar o comportamento das visualizações ou da própria aplicação (ao nível da camada de negócio). Assim, estes parâmetros são úteis para quando se pretende definir o número de eventos apresentado (`ItemCount`), a duração de cada um (`ItemDuration`), o máximo número de eventos retornados pelo Google Calendar (`MaxEventsCount`), de entre outros que podem ser também vistos na Tabela 4.5.

Parâmetro	Tipo	Views	Descrição
BackgroundColor	<i>string</i>	Full;Column; Bar	Cor de fundo.
EventNameVisibility	<i>string</i>	Full;Column	Opção de visibilidade do título do evento. <b>Valores:</b> Left, Center e Right.
EventNameColor	<i>string</i>	Full;Column; Bar	Cor do texto usada para o título do evento.
EventNameFontSize	<i>inteiro</i>	Full;Column; Bar	Tamanho da fonte usada para o título do evento.
EventNameFontFamily	<i>string</i>	Full;Column; Bar	Estilo de fonte a usar no título do evento.
EventStartDateVisibility	<i>string</i>	Full;Column; Bar	Opção de visibilidade da data de início do evento. <b>Valores p/ v. Full e Column:</b> Hidden, DateRight, DateLeft, DateCenter, TimeLeft, TimeRight, TimeCenter, DateTimeLeft, DateTimeRight e DateTimeCenter. <b>Valores p/ v. Bar:</b> DateVisible, TimeVisible e DateTimeVisible.
EventStartDateColor	<i>string</i>	Full;Column; Bar	Cor do texto usada para a data de início do evento.
EventStartDateFontSize	<i>inteiro</i>	Full;Column; Bar	Tamanho da fonte usada para a data de início do evento.
EventStartDateFontFamily	<i>string</i>	Full;Column; Bar	Estilo de fonte a usar na data de início do evento.
EventPlaceVisibility	<i>string</i>	Full;Column; Bar	Opção de visibilidade do local do evento. <b>Valores p/ v. Full e Column:</b> Hidden, Right, Left e Center. <b>Valores p/ v. Bar:</b> Hidden e Visible.
EventPlaceColor	<i>string</i>	Full;Column; Bar	Cor do texto usada para o local do evento.
EventPlaceFontSize	<i>inteiro</i>	Full;Column; Bar	Tamanho da fonte usada para o local do evento.
EventPlaceFontFamily	<i>string</i>	Full;Column; Bar	Estilo de fonte a usar no local do evento.

**Tabela 4.1:** Parâmetros da componente visual (parte 1)

Parâmetro	Tipo	Views	Descrição
EventEndDateVisibility	<i>string</i>	Full; Bar Scroll	Opção de visibilidade da data de fim do evento. <b>Valores p/ v. Full:</b> Hidden, DateRight, DateLeft, DateCenter, TimeLeft, TimeRight, TimeCenter, DateTimeLeft, DateTimeRight e DateTimeCenter. <b>Valores p/ v. Bar Scroll:</b> DateVisible, TimeVisible e DateTimeVisible.
EventEndDateColor	<i>string</i>	Full; Bar Scroll	Cor do texto usada para a data de fim do evento.
EventEndDateFontSize	<i>inteiro</i>	Full; Bar Scroll	Tamanho da fonte usada para a data de fim do evento.
EventEndDateFontFamily	<i>string</i>	Full; Bar Scroll	Estilo de fonte a usar na data de fim do evento.
EventDescVisibility	<i>string</i>	Full; Bar Scroll	Opção de visibilidade da descrição do evento. <b>Valores p/ v. Full:</b> Hidden, Right, Left e Center. <b>Valores p/ v. Bar Scroll:</b> Hidden e Visible.
EventDescColor	<i>string</i>	Full; Bar Scroll	Cor do texto usada para a descrição do evento.
EventDescFontSize	<i>inteiro</i>	Full; Bar Scroll	Tamanho da fonte usada para a descrição do evento.
EventDescFontFamily	<i>string</i>	Full; Bar Scroll	Estilo de fonte a usar na descrição do evento.
EventIconVisibility	<i>string</i>	Full;Column	Opção de visibilidade do imagem/ícone do evento relativamente aos restantes elementos do evento (excepto o título). <b>Valores:</b> Hidden, Right, Left, Single e Only. Caso seja Single, os restantes elementos do evento não são exibidos (excepto o título). Caso seja Only, apenas é exibida a imagem/ícone.

Tabela 4.2: Parâmetros da componente visual (parte 2)

<b>Parâmetro</b>	<b>Tipo</b>	<b>Views</b>	<b>Descrição</b>
SecondaryBackgroundColor	<i>string</i>	Column	Cor de fundo (secundária), a usar apenas na v. Column.
SecEventNameColor	<i>string</i>	Column	Cor do texto (secundária) usada para o título do evento.
SecEventNameFontSize	<i>string</i>	Column	Tamanho da fonte (secundário) usada para o título do evento.
SecEventNameFontFamily	<i>string</i>	Column	Estilo de fonte (secundário) a usar no título do evento.
SecEventStartDateColor	<i>string</i>	Column; Bar	Cor do texto (secundária) usada para a data de início do evento.
SecEventStartDateFontSize	<i>inteiro</i>	Column; Bar	Tamanho da fonte (secundário) usada para a data de início do evento.
SecEventStartDateFontFamily	<i>string</i>	Column; Bar	Estilo de fonte (secundário) a usar na data de início do evento.
SecEventPlaceColor	<i>string</i>	Column	Cor do texto (secundária) usada para o local do evento.
SecEventPlaceFontSize	<i>inteiro</i>	Column	Tamanho da fonte (secundário) usada para o local do evento.
SecEventPlaceFontFamily	<i>string</i>	Column	Estilo de fonte (secundário) a usar no local do evento.
EventIconRelativeLocalization	<i>inteiro</i>	Column	Determina a localização do ícone em relação aos outros elementos do evento. Varia entre -1 e 3, sendo que -1 é usado apenas quando o parâmetro EventIcon-Visibility é Hidden. Por exemplo, caso todos os elementos dos eventos sejam visíveis (não sejam Hidden): valor 0 caso o ícone se encontre no topo de cada célula da coluna; valor 1 caso o título se encontre por cima do ícone; valor 2 caso o título e a data de início se encontre por cima do ícone; valor 3 caso o título, data de início e local se encontre por cima do ícone.

**Tabela 4.3:** Parâmetros da componente visual (parte 3)

Parâmetro	Tipo	Views	Descrição
EventName	<i>string</i>	Full;Column; Bar; XML	Filtragem por título ou parte de título do evento (opcional).
Where	<i>string</i>	Full;Column; Bar; XML	Restrição geográfica (opcional).
FromDate	<i>date</i>	Full;Column; Bar; XML	Data de início da <i>query</i> (opcional; <i>default</i> é a data corrente).
ToDate	<i>date</i>	Full;Column; Bar; XML	Data de fim da <i>query</i> (opcional; <i>default</i> é a data corrente + 15 dias).
FullTextMatching	<i>string</i>	Full;Column; Bar; XML	Filtragem por qualquer pedaço de texto que possa existir em qualquer um dos elementos informativos dos eventos (opcional).

**Tabela 4.4:** Parâmetros da componente de filtragem

### Parâmetros adicionais

Para além dos parâmetros das componentes visual, comportamental e de filtragem, existem outros adicionais (como se pode observar na Tabela 4.6). Assim, há dois parâmetros essenciais, *Guid* e *GoogleCalendarURL*, que têm, respectivamente, o objectivo de proporcionar a autenticação/autorização através de um *token* (identificador único do utilizador) e especificar o calendário do Google Calendar que o utilizador vai usar, uma vez que podem haver vários.

Também existem outros parâmetros que foram acrescentados com o fim de possibilitar que aplicações externas possam fazer operações na base de dados, de guardar/carregar/actualizar configurações. Observando a Tabela 4.6, os parâmetros para guardar/carregar/actualizar configurações de utilizador são: *LoadConfig*, *SaveConfig*, *UpdateConfig* e *NewConfigName*. Os dois últimos parâmetros da Tabela 4.6 são destinados a permitir o carregamento e posterior visualização de configurações *default*.

## 4.3 Especificação da Aplicação

Nesta secção vai ser abordada a especificação da *DSEventApp* ao nível de três camadas: interface, negócio e dados.

Na camada de interface vão ser apresentados uns esboços das visualizações Full, Column, Bar e XML e, também, uns *mockups* dos respectivos formulários de configuração. Uma vez que a tecnologia escolhida é a Microsoft ASP.NET MVC 2.0<sup>1</sup>, que segue o padrão arquitectural Model-View-Controller (MVC), a interface vai corresponder à View da *DSEventApp*, ou

<sup>1</sup><http://stephenwalthers.com/blog/archive/2009/02/05/chapter-1-an-introduction-to-asp.net-mvc.aspx>

Parâmetro	Tipo	Views	Descrição
ItemDuration	<i>inteiro</i>	Full; Bar Fade	Duração, em milissegundos, de cada iteração entre eventos. <b>Nota:</b> na v. Bar Fade corresponde ao tempo durante o qual o evento está totalmente visível (não está na fase de <i>fading</i> ).
ItemCount	<i>inteiro</i>	Full;Column; Bar	Número de eventos a exibir. <b>Nota:</b> nas v. Full e Bar são feitas ItemCount iterações em ciclo infinito, enquanto que na v. Column são apresentados, de forma estática, ItemCount eventos em coluna.
MaxEventsCount	<i>inteiro</i>	Full;Column; Bar; XML	Máximo número de eventos retornados pelo serviço Google Calendar. <b>Nota:</b> nas v. Full, Column e Bar, por cada pedido, são apresentados sequencialmente ItemCount eventos dos MaxEventsCount retornados (há gestão de sessão), enquanto que na v. XML são apresentados os MaxEventsCount retornados (não há gestão de sessão).
ModelRefreshInterval	<i>inteiro</i>	Full;Column; Bar	Intervalo de tempo (em minutos) em que a aplicação vai refrescar os dados obtidos do serviço Google Calendar.
AnimationType	<i>string</i>	Bar	Tipo de animação utilizada. <b>Nota:</b> só se vai incluir dois tipos de animação para a v. Bar (Fade e Scroll), mas no futuro poderão ser incluídas nas v. Full e Column.
FadeDuration	<i>inteiro</i>	Bar Fade	Duração do <i>fading</i> . <b>Nota:</b> a duração de <i>fade-in</i> mais a de <i>fade-out</i> é igual ao dobro de FadeDuration.
ScrollSpeed	<i>inteiro</i>	Bar Scroll	Velocidade de deslizamento de texto em rodapé.

**Tabela 4.5:** Parâmetros da componente comportamental

Parâmetro	Tipo	Views	Descrição
Guid	<i>string</i>	Full;Column; Bar; XML	<i>Token</i> (obtido via OpenID) utilizado para efectuar autenticação na aplicação e obter as credenciais de acesso ao serviço Google Calendar (programaticamente via OAuth).
GoogleCalendarURL	<i>string</i>	Full;Column; Bar; XML	URL do calendário a consultar (também indica que o serviço usado é o Google Calendar).
LoadConfig	<i>string</i>	Full;Column; Bar	Nome da configuração a ser descarregada da base de dados para ser visualizada.
SaveConfig	<i>string</i>	Full;Column; Bar	Nome da configuração a ser armazenada na base de dados e de seguida é visualizada.
UpdateConfig	<i>string</i>	Full;Column; Bar	Nome antigo da configuração a ser alterada na base de dados e de seguida é visualizada.
NewConfigName	<i>string</i>	Full;Column; Bar	Novo nome da configuração a ser alterada na base de dados e de seguida é visualizada. <b>Nota:</b> este parâmetro vem juntamente com o parâmetro UpdateConfig.
DefaultConfigListName	<i>string</i>	Full;Column; Bar	Nome de uma lista de configurações que vêm por <i>default</i> na DSEventApp.
LoadDefaultConfig	<i>string</i>	Full;Column; Bar	Nome de uma configuração existente na lista de configurações <i>default</i> especificada no parâmetro DefaultConfigListName. Através dos parâmetros DefaultConfigListName e LoadDefaultConfig é possível carregar uma configuração <i>default</i> existente na DSEventApp.

Tabela 4.6: Parâmetros adicionais

seja, tudo o que tem a ver com páginas em HTML e com lógica de visualização. Numa fase final, vão ser analisados dois diagramas de sequência para explicar as interações existentes nos formulários de configuração.

Quanto à camada da lógica de negócio, vão ser apresentados diagramas de classes que especificam o relacionamento entre classes e os respectivos métodos e variáveis de instância. Também é importante sublinhar que, considerando a arquitectura MVC, as classes Model (modelos), não relacionadas com a base de dados, e todas as classes Controller (controladores) vão ser consideradas como pertencentes à lógica de negócio.

Por fim, vai ser especificada a camada de base de dados, em que vão ser apresentadas todas as entidades persistentes da DSEventApp. Como vai ser usada a *framework* NHibernate para .NET (faz o mapeamento ORM entre classes C# e relações de uma base de dados), as entidades persistentes vão ser apresentadas nas duas perspectivas: através de um diagrama de classes e o respectivo diagrama ER (diagrama com as relações pertencentes à base de dados). Também é importante referir que, considerando a arquitectura MVC, o diagrama de classes vai conter as classes Model que não dizem respeito à camada da lógica de negócio.

#### 4.3.1 Camada de Interface

Nesta subsecção vai ser apresentada e descrita a modelação efectuada para a camada de interface da DSEventApp. Como já foi mencionado, vão ser apresentados os quatro tipos de visualização que a DSEventApp vai suportar (Full, Column, Bar e XML) através de vários esboços para cada um destes, ilustrando, assim, a grande flexibilidade existente ao nível da configuração. A visualização XML vai apenas ser modelada a um nível estrutural, isto é, serão especificados todos os campos (neste caso *tags*) relevantes para a apresentação de eventos retornados pelo Google Calendar, numa interface sem qualquer elemento decorativo.

Uma vez apresentados e descritos os vários esboços para as visualizações, ir-se-á analisar os *mockups* para os formulários de configurações.

#### Visualizações

Como tem sido referido, a aplicação vai suportar quatro tipos de visualizações: Full, Column, Bar e XML.

Tendo em consideração essas visualizações é importante lembrar que a aplicação permite a mudança de cores, fontes, etc. no conteúdo informativo de cada evento, de acordo com os parâmetros passados à aplicação, por isso as cores utilizadas nestes exemplos não são obrigatórias. Também é de referir que não são usadas bordas nem outros elementos decorativos nas visualizações, para que a aplicação permita, por parte de peritos em *design*, posterior customização. Um último ponto a considerar é não usar texto em



português (nem em qualquer outra língua) para que a aplicação possa ser internacionalizada.

De seguida, serão apresentadas várias figuras para cada uma das três visualizações, que representam as várias alternativas de configuração que a aplicação suporta (consoante os parâmetros recebidos na *query string*).

Para a visualização XML será especificada uma estrutura fácil de entender, tendo sempre em conta o formato padrão, iCalendar. Por isso, a lista de eventos retornados pelo Google Calendar, vai poder ser utilizada por outras aplicações, através de *parsing* de XML.

**Visualização Full** A visualização Full, tal como o próprio nome indica, corresponde à utilização da totalidade do ecrã de um painel de *digital signage*. A vantagem deste tipo de visualização é a de permitir apresentar uma maior quantidade de informação e de, ao mesmo tempo, torná-la visível para o utilizador.

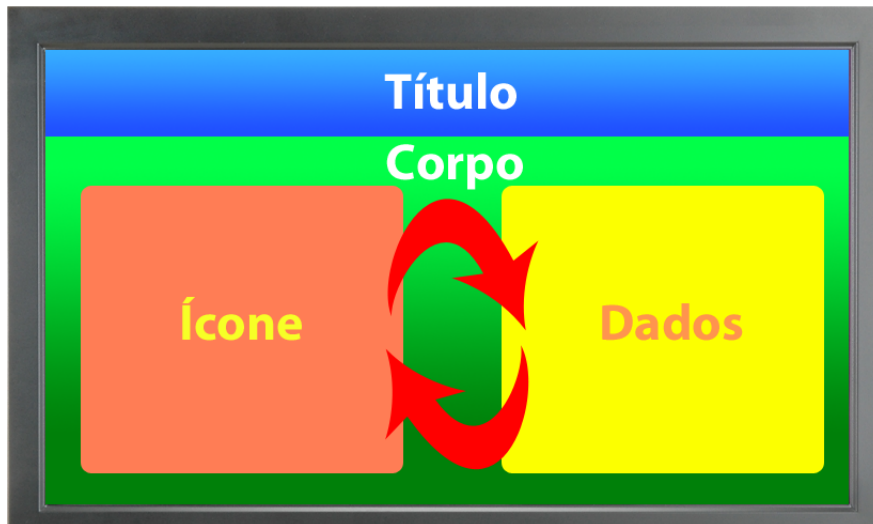
Com vista à organização da informação apresentada na visualização Full, resolveu-se criar um *layout* com duas zonas distintas: a do título e a do corpo do evento. Por sua vez, esta última é composta por duas sub-zonas: a do ícone e a dos dados. Na Figura 4.1 pode-se ver a divisão genérica dos conteúdos informativos sobre o evento, que constitui o *layout* da visualização Full, descrito nas duas frases anteriores.

Fazendo uma análise mais detalhada da Figura 4.1, destaca-se que a zona azul apenas contém o título do evento (cerca de 20% do ecrã) e a zona verde contém informação sobre o corpo do evento (cerca de 80% do ecrã). Quanto à sub-zona do ícone, tal como o nome indica, corresponde à zona onde é apresentada uma imagem ilustrativa acerca do evento. Quanto à sub-zona dos dados, esta pode ser constituída pelos seguintes elementos informativos: data/hora de começo e término, local e descrição do evento (consultar Figura 4.2).

O *layout* é bastante flexível, uma vez que o utilizador pode decidir excluir a sub-zona do ícone, fazendo com que a zona do corpo seja completamente preenchida pela sub-zona dos dados (o inverso também é válido). Também é possível especificar o lado onde aparece cada uma das duas sub-zonas (ver setas a vermelho incluídas na Figura 4.1).

De seguida, vão-se apresentar exemplos de várias visualizações possíveis, que a DSEventApp irá suportar, tendo em conta várias configurações para a visualização Full.

Na Figura 4.3 pode-se ver um exemplo de um evento com a maior quantidade de informação que pode conter. É de realçar o uso de tonalidades de vermelho diferentes na data/hora de começo e término do evento, para haver uma melhor distinção desses dois elementos informativos. No entanto, as cores, tamanho e estilo de fonte da informação



**Figura 4.1:** Principais zonas correspondentes à localização da informação sobre um evento



**Figura 4.2:** Dados correspondentes a um evento (os 3 campos são opcionais)



**Figura 4.3:** *Visualização Full com toda a informação possível, com a sub-zona do ícone no lado esquerdo e a sub-zona dos dados no lado direito*

dos eventos podem ser especificados de livre vontade por parte do utilizador.

De uma forma análoga ao exemplo da Figura 4.3, o da Figura 4.4 contém toda a informação de um evento, apenas diferindo no facto de trocar o lado da sub-zona do ícone com o da sub-zona dos dados.

Nas Figuras 4.5 e 4.6 podem-se ver exemplos de visualizações Full com toda a informação do evento à excepção da sua descrição.

Na Figura 4.7 pode-se ver um exemplo de visualização Full com título, ícone do evento e datas/horas de início e fim.

Na Figura 4.8 pode-se ver um exemplo de visualização Full com título e ícone do evento que poderá ter embutida informação sobre a data/hora, local, etc..

Na Figura 4.9 pode-se ver um exemplo de visualização Full apenas com o ícone do evento. Neste caso, tal como na Figura 4.8, o ícone poderá conter informação adicional sobre o evento (título, data, local, etc.).

Na Figura 4.10 destaca-se uma visualização Full sem ícone, mas com o resto da informação disponível.

Na Figura 4.11 pode-se verificar um exemplo de uma visualização Full com o título, data/hora e local onde o evento vai decorrer.

Feita a apresentação de vários exemplos de visualização Full suportados pela DSEventApp, é de sublinhar que não se foi mais exaustivo, devido aos exemplos serem suficientes para compreender a flexibilidade da



**Figura 4.4:** Visualização Full com toda a informação possível, com a sub-zona dos dados no lado esquerdo e a sub-zona do ícone no lado direito



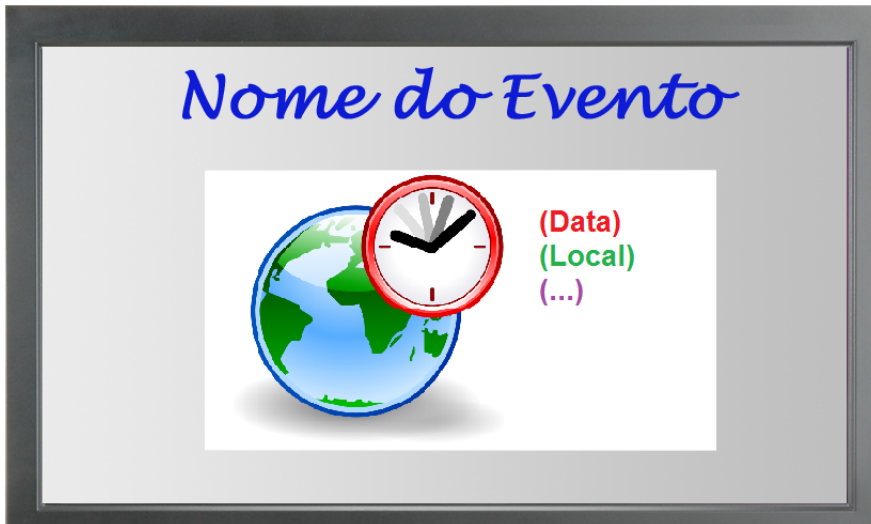
**Figura 4.5:** Visualização Full com toda a informação possível à exceção da descrição, com a sub-zona do ícone no lado esquerdo e a sub-zona dos dados no lado direito



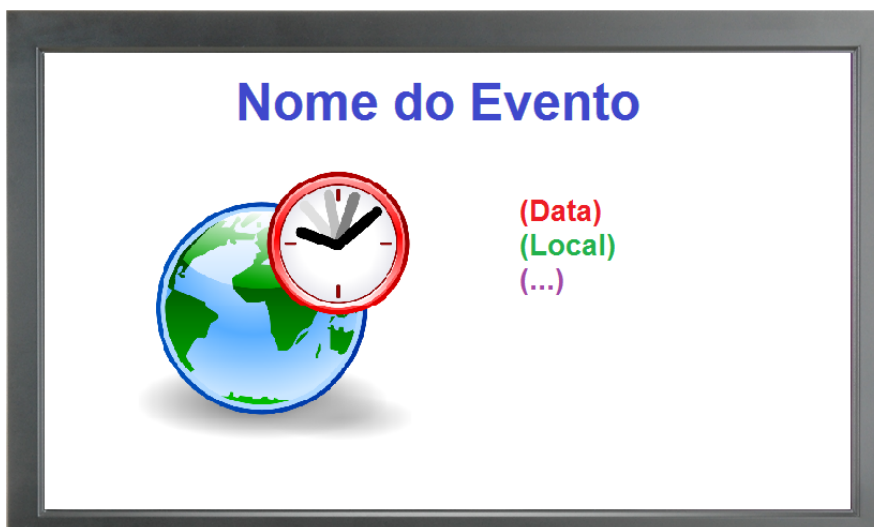
**Figura 4.6:** *Visualização Full com toda a informação possível à exceção da descrição, com a sub-zona dos dados no lado esquerdo e a sub-zona do ícone no lado direito*



**Figura 4.7:** *Visualização Full com o título, ícone do evento e datas/horas de início e fim*



**Figura 4.8:** *Visualização Full com título e ícone do evento (informação adicional encontra-se no ícone)*



**Figura 4.9:** *Visualização Full apenas com o ícone do evento*

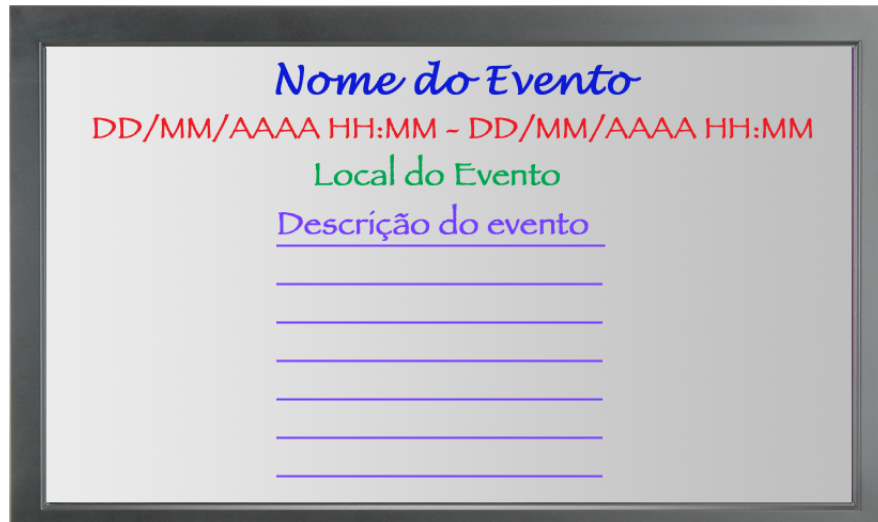
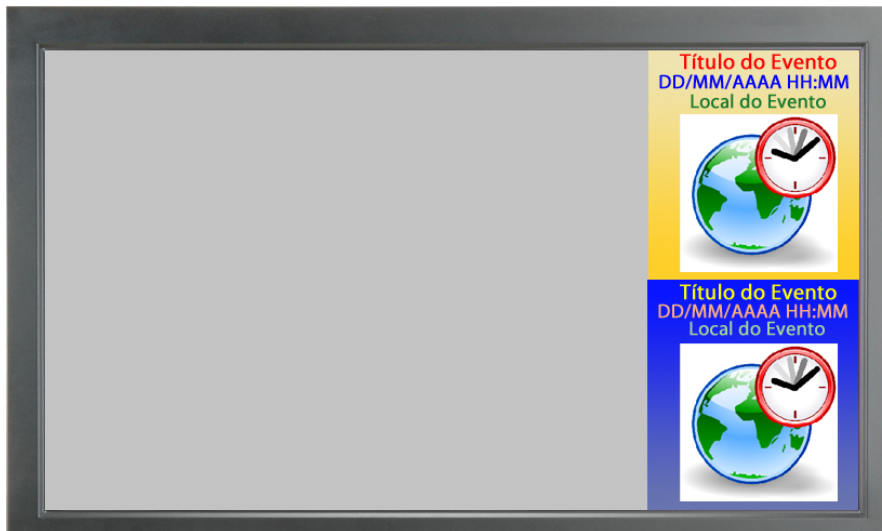


Figura 4.10: Visualização Full com toda a informação possível à excepção do ícone



Figura 4.11: Visualização Full com o título, data/hora e local do evento



**Figura 4.12:** Visualização Column com título, data/hora de início, local e ícone dos eventos, localizados por cima do ícone

aplicação, no que diz respeito à localização dos elementos informativos sobre eventos.

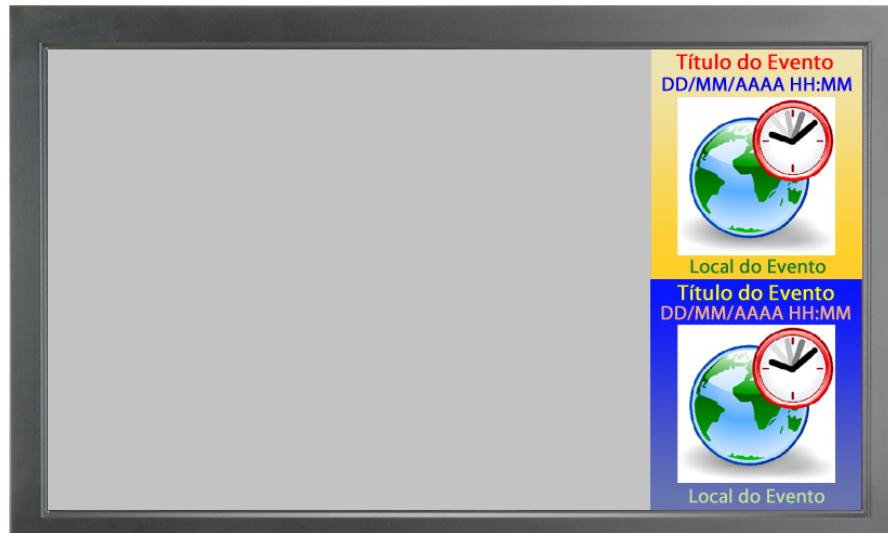
**Visualização Column** A visualização Column corresponde à apresentação de uma lista de eventos dispostos em coluna. A coluna vai ocupar o ecrã inteiro do painel de *digital signage*, de modo a permitir que outras aplicações possam utilizar a visualização num fragmento do ecrã em forma de coluna.

Uma das características importantes na visualização Column, é o facto de haver a necessidade de melhorar a visibilidade de eventos consecutivos e, para isso, existem parâmetros de configuração que permitem alternar as cores de fundo, os tamanhos, as cores e estilos de fonte.

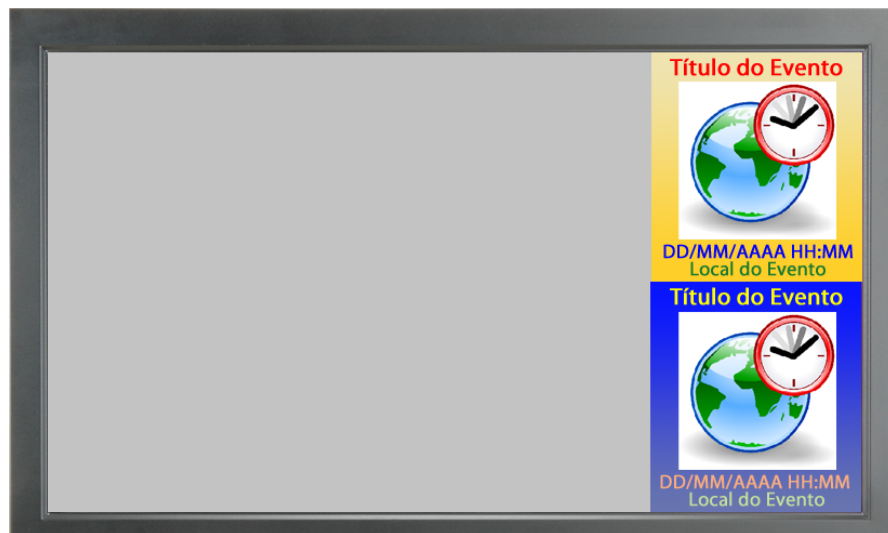
Nas Figuras 4.12, 4.13, 4.14 e 4.15, podem-se ver exemplos de visualizações do tipo Column com toda a informação possível. Vale a pena lembrar que, em relação à visualização Full, removeu-se a data/hora de fim do evento e a descrição do evento por falta de espaço.

Nas Figuras 4.16, 4.17, 4.18 e 4.19, pode-se ver os restantes exemplos de visualização Column que a aplicação vai possibilitar. Todas estas não apresentam o ícone do evento e, assim, há mais espaço para apresentar um maior número de eventos. É de notar que a visualização da Figura 4.19 permite a representação de oito (ou até mais) eventos em coluna, uma vez que só se apresenta o título como componente informativa de cada um. O número de eventos fica ao critério do utilizador, mas é sensato escolher um número equilibrado para que seja possível captar a informação mais facilmente.

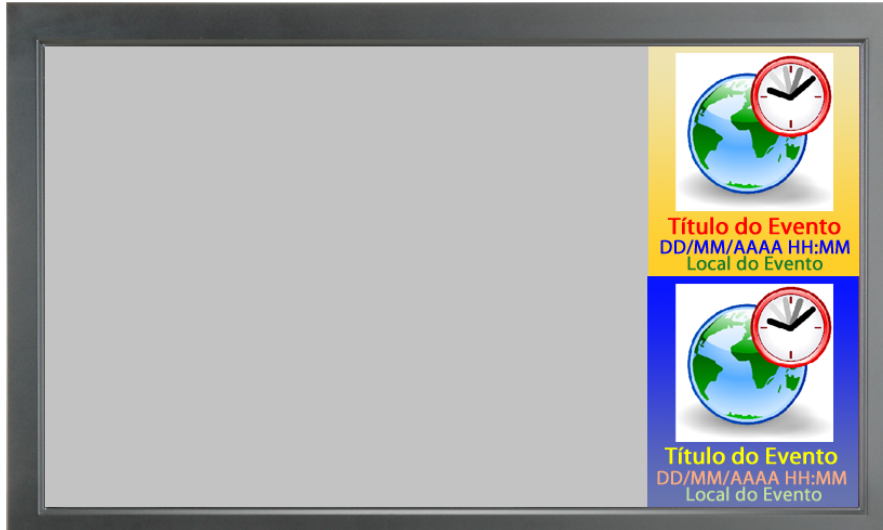




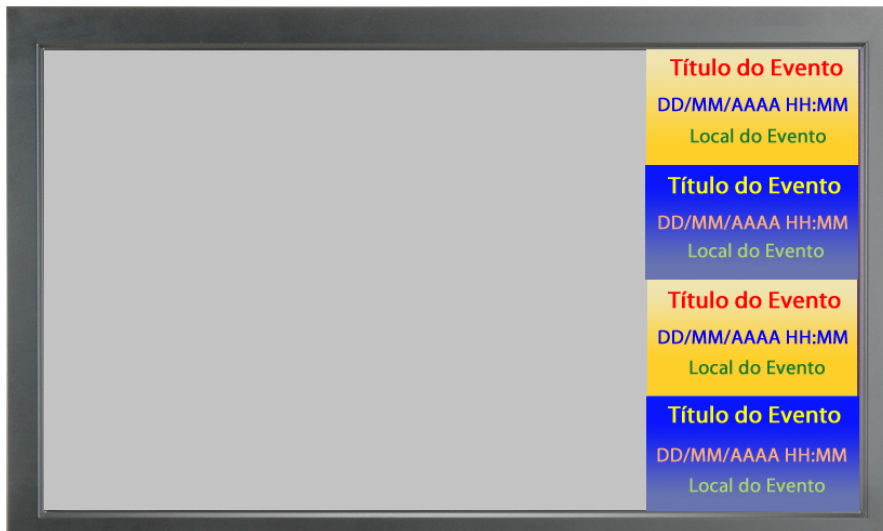
**Figura 4.13:** *Visualização Column com título e data/hora de início por cima do ícone; local dos eventos por baixo do ícone*



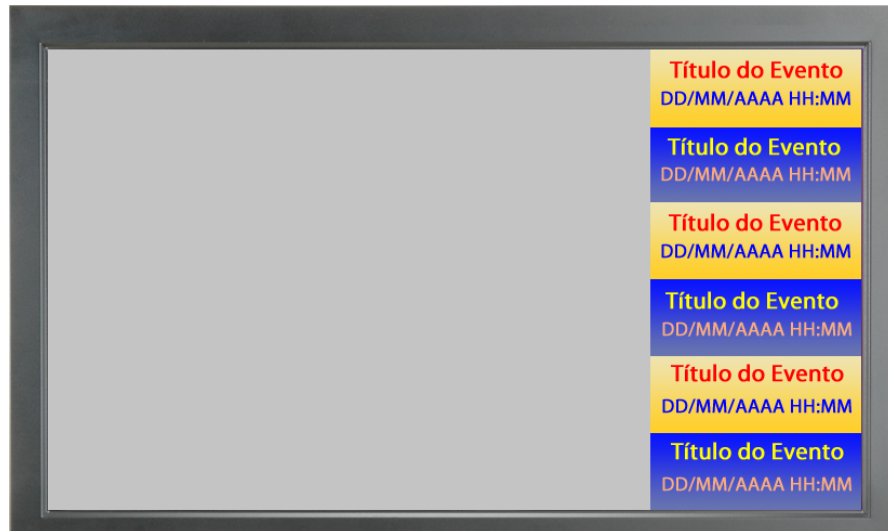
**Figura 4.14:** *Visualização Column com o título por cima do ícone; data/hora de início e local dos eventos por baixo do ícone*



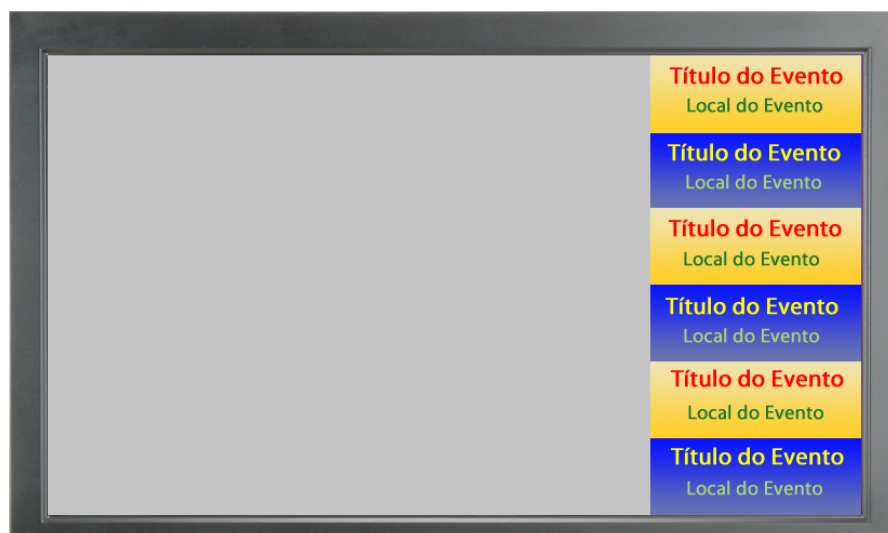
**Figura 4.15:** *Visualização Column com título, data/hora de início e local por baixo do ícone*



**Figura 4.16:** *Visualização Column com título, data/hora de início e local dos eventos*



**Figura 4.17:** Visualização Column com título e data/hora de início dos eventos



**Figura 4.18:** Visualização Column com título e local dos eventos



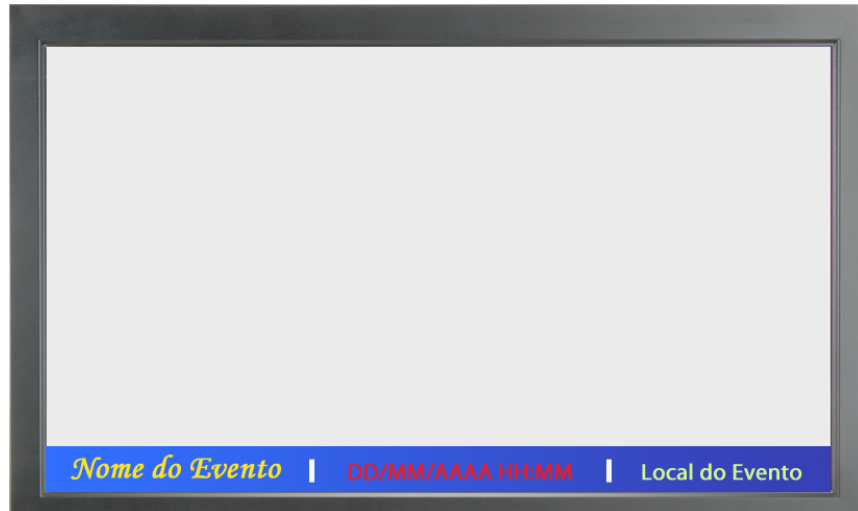
**Figura 4.19:** *Visualização Column apenas com o título dos eventos*

**Visualização Bar** A visualização Bar corresponde à apresentação de eventos sob a forma de um rodapé rolante com uma dada velocidade, especificada pelo utilizador (visualização Bar Scroll) ou com iteração de um evento de cada vez em rodapé, segundo um intervalo de tempo (visualização Bar Fade). Como já foi dito, a visualização Bar Scroll permite apresentar mais informação do que a Bar Fade, uma vez que o texto é apresentado de uma forma dinâmica.

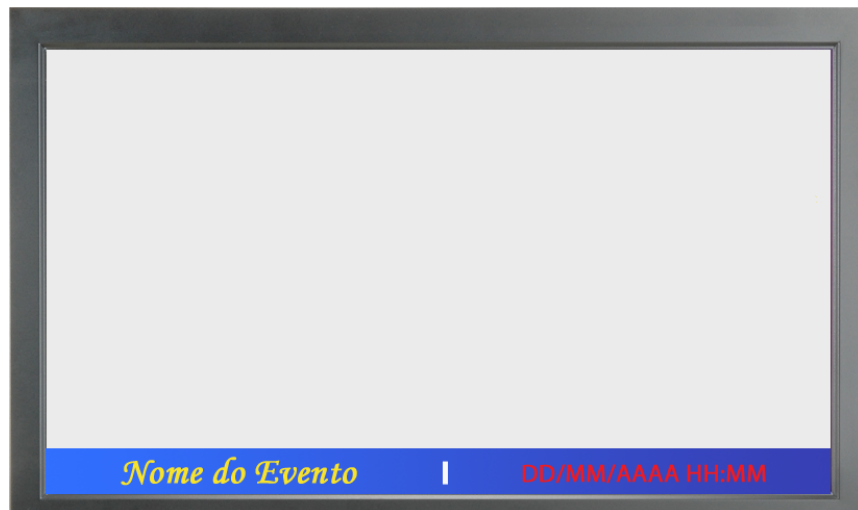
Dado que este tipo de visualização exige um espaço bastante reduzido, deve-se dar particular importância à forma como os conteúdos são apresentados, nomeadamente ao nível do tamanho de fonte e cores utilizadas, de modo a que o público alvo consiga observar, sem qualquer dificuldade, os eventos disponibilizados. Para o efeito, é fundamental seleccionar tamanhos de fonte com uma dimensão considerável para o espaço existente e as cores de fonte devem-se contrastar com a de fundo.

Tal como as visualizações Full e Column, a Bar tem de estar em modo *full-screen* para que quaisquer aplicações externas possam “encaixar” a visualização no sítio pretendido do ecrã (de uma forma geral aparece em rodapé).

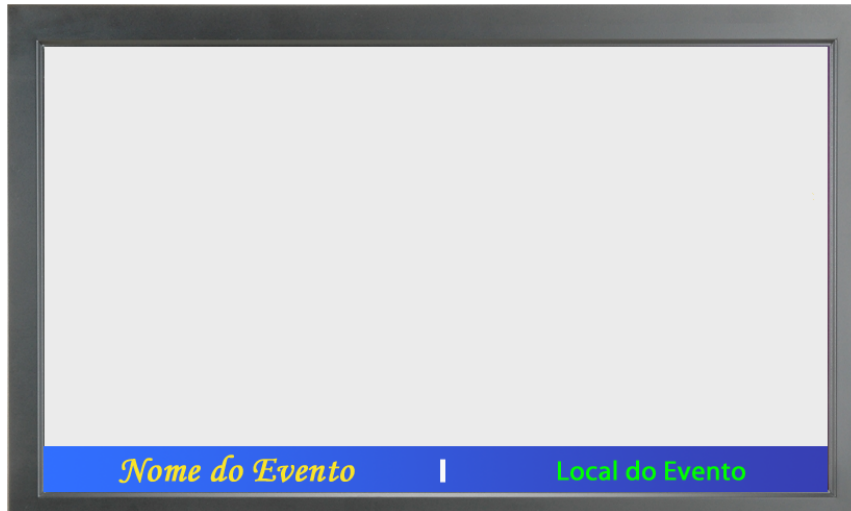
Nas Figuras 4.20, 4.21, 4.22 e 4.23 podem-se ver exemplos de visualização Bar Fade (em que apenas é disponibilizada a informação essencial de cada evento) e na Figura 4.24 é apresentado um exemplo genérico de visualização Bar Scroll (onde pode ser apresentada bastante informação sobre cada evento).



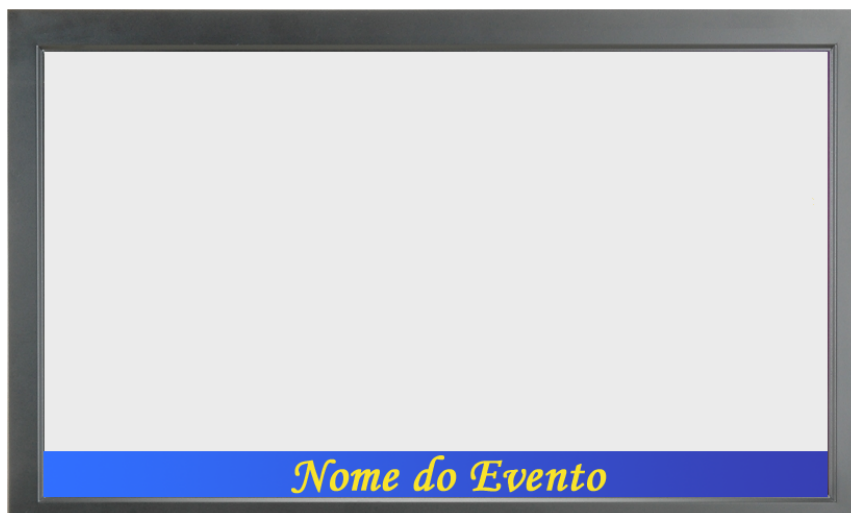
**Figura 4.20:** Visualização Bar Fade que apresenta o título, a data/hora e local de cada evento (toda a informação possível para este tipo de visualização)



**Figura 4.21:** Visualização Bar Fade que apresenta o título e a data/hora de cada evento



**Figura 4.22:** *Visualização Bar Fade que apresenta o título e o local de cada evento*



**Figura 4.23:** *Visualização Bar Fade que apresenta apenas o título de cada evento*



**Figura 4.24:** Visualização Bar Scroll que pode apresentar praticamente toda a informação de cada evento à exceção do ícone

**Visualização XML** A visualização XML, como o próprio nome indica, é destinada a apresentar informação de forma estruturada num formato XML. De modo a especificar uma estrutura simples de se entender e fácil de ser processada por aplicações externas, utilizou-se a informação mais importante existente no formato iCalendar, ou seja, fez-se um mapeamento de alguns dos campos do formato iCalendar para o formato XML.

Para dar uma ideia de como o formato iCalendar é representado e estruturado, pode-se visualizar a Figura 4.25, que contém apenas um evento (para simplificar a leitura).

Analisando a estrutura da visualização XML, primeiro definiram-se as *tags* de abertura e fecho, `<calendar>` e `</calendar>`, para englobar a informação mais importante sobre um dado calendário e todos os eventos com a respectiva informação (como se pode observar na Figura 4.26). Pode-se verificar que, para um dado calendário, existe o atributo com o respectivo URL (`calurl`), o atributo com o serviço utilizado (Google Calendar), a *tag* `<name>` com o seu nome e a *tag* `<timezone>` com o fuso horário considerado.

No que diz respeito à informação sobre os eventos, pode-se ver que existe o atributo `numevents` contida na *tag* `<events>`, que nos diz o número de eventos retornados pelo Google Calendar. Para cada evento retornado, existe um identificador designado por `uid`, o título (`<summary>`), a data de início e fim (`<dtstart>` e `<dtend>`), local (`<location>`), descrição (`<description>`) e, por fim, o URL da imagem associada ao evento (`<image-url>`).

```

BEGIN:VCALENDAR
PRODID:-//Google Inc//Google Calendar 70.9054//EN
VERSION:2.0
CALSCALE:GREGORIAN
METHOD:PUBLISH
X-WR-CALNAME:Calendario3
X-WR-TIMEZONE:Atlantic/Azores
X-WR-CALDESC:Calendário destinado a testes.
BEGIN:VEVENT
DTSTART:20110520T110000Z
DTEND:20110520T120000Z
DTSTAMP:20110519T161604Z
UID:4u4q7r3llqvta937geikaf467c@google.com
CREATED:20110519T161518Z
DESCRIPTION:Evento de teste de muito interesse.
LAST-MODIFIED:20110519T161518Z
LOCATION:Lisboa
SEQUENCE:0
STATUS:CONFIRMED
SUMMARY:EventoTeste1
TRANSP:OPAQUE
END:VEVENT
END:VCALENDAR

```

Figura 4.25: Um exemplo simples no formato iCalendar

```

<calendar calurl="https://www.google.com/(...)" service="Google Calendar">
  <name>(…)</name>
  <timezone>(…)</timezone>
  <events numevents="(…)">
    <event uid="(…)">
      <summary>(…)</summary>
      <dtstart>(…)</dtstart>
      <dtend>(…)</dtend>
      <location>(…)</location>
      <description>(…)</description>
      <image-url>(…)</image-url>
    </event>
    <event uid="(…)">
      (…)
    </event>
    (…)
  </events>
</calendar>

```

Figura 4.26: Estrutura da visualização XML



Para finalizar, é importante referir que para a visualização XML não há qualquer intenção de persistir configurações, uma vez que são utilizados poucos parâmetros de configuração, mais concretamente os parâmetros da componente de filtragem, para a DSEventApp poder efectuar *queries* personalizadas ao serviço Google Calendar.

### Formulários de Configuração

Os formulários de configuração vão servir essencialmente para ajudar o utilizador a especificar várias configurações para as visualizações Full, Column, Bar e XML e poder, posteriormente, visualizá-las. Para além dessa principal funcionalidade, também vai haver a possibilidade de persistir, actualizar ou remover configurações e até mesmo aproveitar configurações existentes de utilizador ou *default*.

Dada a importância dos formulários de configuração, serão apresentados e detalhados *mockups* do formulário da visualização Full, dado que as restantes visualizações não vão ser muito diferentes.

Por uma questão de organização, cada *mockup* vai corresponder apenas a um segmento do formulário e cada segmento vai ser analisado de forma individual. De seguida, vão ser apresentadas e analisadas as várias figuras correspondentes aos *mockups* do formulário.

Na Figura 4.27 observa-se um *mockup* que corresponde à zona de introdução de um URL de um dado calendário e à zona de especificação dos alinhamentos/visibilidades dos elementos informativos dos eventos a apresentar. Considerando os campos de alinhamento/visibilidade preenchidos, repara-se que o título será alinhado ao centro (valor `Center`), a data e hora de início vão ser alinhados à esquerda (valor `DateTimeLeft`), a hora de fim será alinhada à esquerda (valor `TimeLeft`), o local vai ser alinhado à esquerda (valor `Left`), o ícone vai estar alinhado à direita (valor `Right`) e a descrição vai ser o único elemento que estará invisível na apresentação (valor `Hidden`).

Dando agora destaque ao *mockup* da Figura 4.28 pode-se ver, em primeiro lugar, um campo, *Background Color*, para especificar a cor de fundo para todos os eventos, e de seguida existem campos gerais para definir as cores, tamanhos e família de fontes para todos os elementos de cada evento (campos *General Font Color*, *General Font Size* e *General Font Family*). Como se pode deduzir, os campos gerais servem para ajudar o utilizador a preencher os campos mais rapidamente, quando não pretende ter cores/tamanhos/famílias de fonte muito diferentes para cada elemento dos eventos. Para um maior grau de personalização, o utilizador sempre pode preencher os campos do formulário um a um.

É de destacar que vai existir a funcionalidade de expandir e colapsar os campos específicos de cor/tamanho/família de fontes. Assim, a ideia é que, no momento inicial de preenchimento de campos, os campos gerais estejam todos colapsados, com a finalidade de reduzir consideravelmente o tamanho

**Input a Calendar URL:**URL: **Visibilities/Alignments**

Title: <input type="text" value="Center"/>	Start Date: <input type="text" value="DateTimeLeft"/>
End Date: <input type="text" value="TimeLeft"/>	Place: <input type="text" value="Left"/>
Description: <input type="text" value="Hidden"/>	Icon: <input type="text" value="Right"/>

**Figura 4.27:** Parte do formulário para introduzir o URL do calendário e visibilidades/alinhamentos dos elementos de cada evento

do formulário, isto é, torná-lo mais simples. No exemplo da Figura 4.28, pode-se observar que os campos de cor e tamanho de fonte estão expandidos, enquanto que os campos sobre as famílias de fonte estão colapsadas.

Para elementos que foram definidos como escondidos (valor *Hidden*) na Figura 4.27, é natural que não façam parte dos campos expandidos da Figura 4.28 e, assim, é uma forma de não confundir o utilizador e de simplificar o formulário.

Também é importante ter em consideração que, ao clicar no *link Choose a Color* existente no *mockup* da Figura 4.28, deve aparecer uma janela pequena (um *pop-up*) com as principais cores que o utilizador tem à disposição e proporciona-se uma interface mais *user-friendly*.

Quanto aos campos já preenchidos no exemplo da Figura 4.28, pode-se ver que a cor de fundo é cinzento claro (valor *Lightgrey*). A cor geral de fonte é nula, uma vez que neste caso o utilizador pretendeu personalizar as cores dos elementos dos eventos uma a uma. A cor do título é azul marinho (valor *Navy*), a cor da data de início é vermelha (valor *Red*), a cor da data de fim é castanho-avermelhado (valor *Maroon*), a cor do local é verde (valor *Green*) e a cor da descrição é azul (valor *Blue*).

No que diz respeito ao campo, *General Font Size*, é de realçar que, no exemplo da Figura 4.28, está definido um tamanho de letra de 250% para todo o texto dos eventos. De notar que a percentagem vai ser relativa ao tamanho padrão de fonte existente em páginas *web*.

Para finalizar a análise do *mockup* da Figura 4.28, o campo *General Font Family* tem o valor *Calibri* que é uma família de fontes bastante conhecida. No entanto vão existir como opção outras famílias de fonte conhecidas como a *Arial*, *Helvetica*, *Cambria*, etc..

Uma vez terminada a análise dos parâmetros de configuração da componente visual (campos existentes nas Figuras 4.27 e 4.28, à excepção do campo de URL do calendário), vão ser analisados os parâmetros da componente de filtragem e da componente comportamental (campos existentes na

<b>Background and Font Configurations</b>		
Background Color:	<input type="text" value="Lightgrey"/>	<a href="#">Choose a Color</a>
General Font Color:	<input type="text" value="--"/>	<a href="#">Choose a Color</a> <a href="#">Collapse &lt;&lt;</a>
Title's Font Color:	<input type="text" value="Navy"/>	<a href="#">Choose a Color</a>
Start Date's Font Color:	<input type="text" value="Red"/>	<a href="#">Choose a Color</a>
End Date's Font Color:	<input type="text" value="Maroon"/>	<a href="#">Choose a Color</a>
Place's Font Color:	<input type="text" value="Green"/>	<a href="#">Choose a Color</a>
Description's Font Color:	<input type="text" value="Blue"/>	<a href="#">Choose a Color</a>
General Font Size:	<input type="text" value="250"/> (%)	<a href="#">Collapse &lt;&lt;</a>
Title's Font Size:	<input type="text" value="250"/> (%)	
Start Date's Font Size:	<input type="text" value="250"/> (%)	
End Date's Font Size:	<input type="text" value="250"/> (%)	
Place's Font Size:	<input type="text" value="250"/> (%)	
Description's Font Size:	<input type="text" value="250"/> (%)	
General Font Family:	<input type="text" value="Calibri"/>	<a href="#">Expand &gt;&gt;</a>

**Figura 4.28:** Parte do formulário para definir cores (de fundo e fonte), tamanho e família de fonte dos elementos de cada evento

Figura 4.29).

Considerando os campos de filtragem existentes no *mockup* da Figura 4.29, pode-se dizer que vai ser proporcionada filtragem por título (campo *Title*), local (campo *Place*), data de início e fim (campos *Start Date* e *End Date*), e por qualquer pedaço de texto existente na informação do evento (campo *Full Text Matching*). Estes campos, ao contrário dos restantes, são opcionais, uma vez que o utilizador pode não ter a intenção de exigir um critério de filtragem e, devido a isso, são retornados todos os eventos do calendário especificado.

Por uma questão de utilidade, o utilizador vai ter acesso a um pequeno calendário para poder inserir, mais facilmente, as datas de início e fim, uma vez que se tornaria muito inconveniente e convidativo a erros desnecessários, caso a introdução do formato das datas fosse totalmente manual.

O formato utilizado no preenchimento dos campos *Start Date* e *End Date* vai ser DD/MMM/AAAA. Como se pode ver na Figura 4.29, o mês vai ser introduzido no formato alfabético para não haver qualquer tipo de ambiguidade entre mês e dia. Os campos *Title*, *Place* e *Full Text Matching* terão um formato livre, uma vez que servem para procurar eventos com o respectivo texto introduzido.



Considerando os campos de filtragem preenchidos no formulário da Figura 4.29, pode-se interpretar que o serviço Google Calendar irá retornar eventos de Abril a Maio de 2011, com a sua ocorrência em Braga e com ligação a actividades desportivas.

Quanto às configurações da componente comportamental, pode-se observar na Figura 4.29 o campo *Item Duration*, que corresponde ao tempo de duração, em milissegundos, que cada evento tem alocado durante a sua apresentação. De seguida, o campo *Number of Events Displayed*, está relacionado com o número de eventos que vão ser apresentados durante cada ciclo. O campo *Max. Number of Events delivered by service*, relaciona-se com o número máximo de eventos retornados pelo serviço Google Calendar (os eventos são guardados na sessão do utilizador). Por fim, o campo *Refresh Interval of Events*, corresponde ao intervalo de tempo máximo em que a *DSEventApp* refresca os eventos, ou seja, quando houver outro pedido semelhante, a *DSEventApp* verifica se já ultrapassou o intervalo de tempo especificado na configuração, e caso isso se verifique, volta a efectuar a mesma *query* ao serviço Google Calendar. Caso contrário, retorna os mesmos eventos já guardados na sessão do utilizador.

Relativamente aos campos preenchidos sobre a componente comportamental (Figura 4.29), a *DSEventApp* vai apresentar uma visualização *Full*, em que cada evento é apresentado durante 7500 milissegundos, sendo que no total são apresentados 10 eventos diferentes (quando se chega ao último recomeça-se o ciclo). São guardados 40 eventos em sessão, de modo a que por cada pedido (igual) sejam retornados os próximos 10 eventos existentes na sessão. Por fim, é definido um intervalo de actualização de eventos de 15

**Service Filter**

Title =  Place =

Start Date =   End Date =  

Full Text Matching =

---

**Behavioral Configurations**

Item Duration:  (milliseconds)

Number of Events Displayed:

Max. Number of Events delivered by service:

Refresh Interval of Events:  (minutes)

---

**Figura 4.29:** Parte do formulário para definir parâmetros da componente de filtragem e comportamental

minutos.

Para finalizar a análise da Figura 4.29, é de referir que existem três botões de submissão do formulário: o botão de *Display* serve para apresentar a visualização Full, resultante dos campos preenchidos no formulário; o botão *Save* tem a função de persistir a configuração especificada no formulário; por fim, o botão *Update* permite ao utilizador actualizar uma configuração à escolha que já tenha sido armazenada na base de dados pelo utilizador.

Tendo em conta a Figura 4.30, observa-se um pequeno segmento que diz respeito às configurações de utilizador persistidas (*Your Stored Configurations*) e um outro destinado às configurações pré-definidas persistidas (*Default Configurations*), com a função de poupar trabalho ao utilizador.

Sempre que o utilizador pretenda carregar visualizações Full da sua autoria, basta clicar no botão *Load* e, assim, os nomes das configurações são listados. Desta maneira, ao seleccionar uma configuração da lista, um mecanismo automático preenche todos os campos de configuração do formulário, para o utilizador poder alterar os campos que achar conveniente, caso pretenda actualizar a configuração seleccionada.

Quanto aos botões *Display* e *Delete* (observar Figura 4.30), como é de esperar, o primeiro tem a função de apresentar e o último serve para remover a visualização seleccionada da lista.

Tendo em conta as configurações *default*, é relevante dizer que vai haver uma organização à base de várias listas de configurações e, no exemplo da

The image shows two sections of a web form. The first section, titled "Your Stored Configurations", contains a dropdown menu with "FVConf1" selected, a "Load" button, a "Display" button, and a "Delete" button. The second section, titled "Default Configurations", contains a dropdown menu with "Default Configuration List 1" selected, another dropdown menu with "Default Configuration X" selected, and a "Load" button.

**Figura 4.30:** Parte do formulário para carregar/persistir/remover configurações

Figura 4.30, a lista seleccionada chama-se *Default Configuration List 1* e a configuração seleccionada é designada por *Default Configuration X*. Tal como nas configurações de utilizador, os respectivos campos também vão ser preenchidos automaticamente, para que o utilizador possa visualizar todas as características da configuração seleccionada, podendo personalizá-la para se tornar numa configuração de utilizador armazenada na base de dados. O utilizador deve introduzir um URL do calendário que irá utilizar, uma vez que as configurações *default* não terão calendários associados, ao contrário das de utilizador.

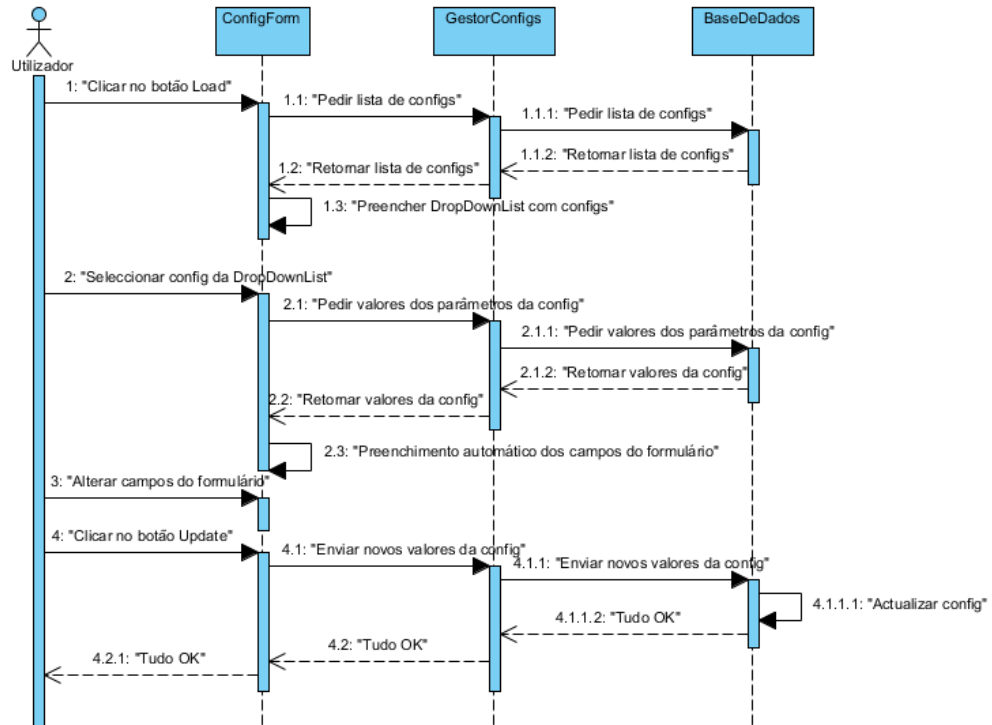
Para finalizar este tópico, é importante lembrar que os formulários para as visualizações Column, Bar e XML não foram considerados nos *mockups*, uma vez que os dois primeiros são bastante semelhantes ao da visualização Full e o da XML é mais simples, pois contém praticamente só campos da componente de filtragem.

### Diagramas de Sequência sobre os Formulários de Configuração

De modo a que se perceba o mecanismo de interacção homem-máquina, existente no preenchimento e submissão de formulários de configuração, vão ser apresentados dois diagramas de sequência de alto nível.

O primeiro diagrama (Figura 4.31) demonstra as etapas que o utilizador deve seguir para poder actualizar uma configuração à sua escolha, já persistida por este. O segundo diagrama (Figura 4.32) é mais pequeno, mas tem como função especificar, de forma simples, as interacções que ocorrem dentro do sistema, após submissão do formulário de configuração para obter uma visualização (Full, Column, Bar ou XML).

Começando pelo diagrama da Figura 4.31, o utilizador, em primeiro lugar, clica no botão *Load* para carregar configurações de utilizador (ver *mockup* da Figura 4.30). Como resposta, a entidade *GestorConfigs* efectua o pedido de obtenção de uma lista de configurações à base de dados. Se tudo



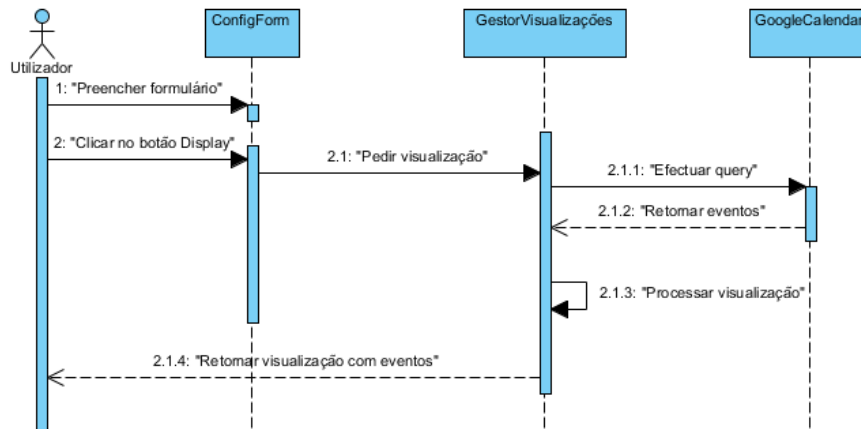
**Figura 4.31:** Diagrama de sequência que indica os passos que o utilizador deve tomar para fazer update de uma configuração e contém as respectivas reacções do sistema

correr bem, esta retorna a lista de configurações e a interface, correspondente à entidade `ConfigFom`, preenche a `DropDownList` (*widget* idêntico a uma *list box*) com a lista das configurações retornadas.

Quando o utilizador selecciona uma configuração da `DropDownList` (passo 2 da Figura 4.31), o sistema efectua um pedido de valores dos parâmetros da configuração seleccionada. A entidade `GestorConfigs` efectua uma *query* à base de dados para obter os valores da configuração. Para terminar, a entidade `ConfigFom` preenche automaticamente todos os seus campos de configuração.

Nesta fase, o utilizador já poderá alterar, à sua escolha, os campos do formulário (passo 3 da Figura 4.31), após isso deve clicar no botão de *Update* (passo 4 da Figura 4.31). Como resposta, são enviados os novos valores preenchidos no formulário para a entidade `GestorConfigs` e esta, por sua vez, efectua um pedido de *update* à base de dados, para os valores da configuração serem actualizados. Se tudo correr bem, o utilizador é notificado com uma mensagem de sucesso.

Passando para a análise do diagrama da Figura 4.32, que diz respeito ao processo de obtenção de uma visualização (Full, Column, Bar ou XML),



**Figura 4.32:** Diagrama de sequência que demonstra as interações existentes no sistema, após submissão do formulário de configuração para obter uma visualização (Full, Column, Bar ou XML)

o utilizador começa com o preenchimento do formulário de configurações. Depois basta clicar no botão *Display* para obter a visualização pretendida.

O processo de obtenção de uma visualização consiste em efectuar uma *query* ao Google Calendar e, se tudo correr bem, os eventos são retornados pelo serviço. De seguida, a entidade *GestorVisualizações* tem que recorrer a um mecanismo para processar a visualização, ou seja, recolher os parâmetros de configuração da componente visual e toda a informação sobre os eventos. Por fim, é gerada a visualização com os vários eventos e com o aspecto de acordo com o que foi especificado pelo utilizador.

Resta referir que não foram considerados diagramas de sequência para outras operações, como as de salvar ou remover configurações, uma vez que estas são simples e semelhantes às já referidas.

### 4.3.2 Camada de Negócio

Nesta subsecção vai ser descrita e analisada a modelação feita para a lógica de negócio da DSEventApp. Tendo em consideração a arquitectura Model-View-Controller, vão ser analisados os diagramas de classes sobre os controladores e também dos modelos que não fazem parte da camada de dados (com a excepção das classes de configuração de visualizações). Vão ser modeladas as interações existentes entre classes e serão apresentadas as variáveis de instância e métodos nelas presentes.

De forma a que os diagramas estejam bem organizados, resolveu-se dividir em *packages*, isto é, todas as classes com funcionalidades semelhantes são agrupadas e, assim, torna-se mais fácil a sua análise e compreensão.

Tendo em conta o diagrama de classes da Figura 4.33 (*package* CSSModel), pode-se observar as várias classes que a DSEventApp vai recorrer para



processar as visualizações Full, Column e Bar. Verifica-se que todas as classes têm o sufixo CSS, uma vez que vão ser usadas pela DSEventApp com o fim de gerar ficheiros CSS (*Cascading Style Sheets*). Recorrendo a esses ficheiros, a DSEventApp conseguirá processar as visualizações de acordo com o que o utilizador especificou.

Quanto ao mecanismo de herança utilizado, pode-se destacar que a super-classe ViewCSS tem como sub-classes a FullViewCSS, ColumnViewCSS e BarViewCSS, que são utilizadas pela DSEventApp para processar, respectivamente, as visualizações Full, Column e Bar. É de notar, também, que a classe ViewCSS contém as variáveis de instância *backgroundColor*, *nameCSS*, *startDateCSS* e *placeCSS*, que são os elementos comuns dos três tipos de visualização (como é óbvio, os elementos específicos de cada tipo de visualização já não constam na super-classe ViewCSS, existindo apenas nas suas sub-classes).

Terminando a análise do diagrama de classes da Figura 4.33, pode-se ver variáveis de instância correspondentes aos parâmetros da componente visual (já analisados anteriormente), como por exemplo: *visibility*, *color*, *fontSize*, *fontFamily*, *backgroundColor*, etc.. A DSEventApp vai ter que mapear essas variáveis em atributos CSS, de modo a que os ficheiros CSS sejam gerados correctamente.

Avançando para o diagrama da Figura 4.34, pode-se observar dois *packages* com funções complementares. O *package* ApplicationConfiguration representa as classes com os parâmetros de visualização da componente visual, comportamental e de filtragem. Por sua vez, o *package* ServiceAuthentication, contém as classes sobre a autenticação/autorização aos serviços pretendidos (neste momento a DSEventApp só irá suportar o acesso ao serviço Google Calendar, mas no futuro poderá ser estendida a outros serviços de eventos).

Analisando o *package* ApplicationConfiguration, pode-se verificar que a classe AppConfig contém o estereótipo «ORM Persistable», que corresponde à classe que armazena todos os dados de configuração de utilizador e que vai ser mapeada, via NHibernate, para uma relação também designada por AppConfig. Também se poderá dizer o mesmo sobre as classes DefaultConfList e DefaultConfig, só que neste caso armazenam-se configurações *default*. Estas classes persistentes e respectivas tabelas vão ser analisadas com maior pormenor na próxima subsecção (Camada de Dados).

Quanto ao *package* ServiceAuthentication resta apenas dizer que vai existir uma interface designada por GenericServiceAuthentication, que tem como função abstrair os vários serviços que a DSEventApp poderá vir a suportar no futuro (como por exemplo, Microsoft Exchange, Eventbrite, etc.). A classe persistente, GoogleCalendarAuthentication, com os dados de autenticação/autorização do serviço Google Calendar, vai ser considerada na próxima subsecção sobre a lógica de acesso à base de dados.

Passando para a análise do diagrama de classes da Figura 4.35, pode-se

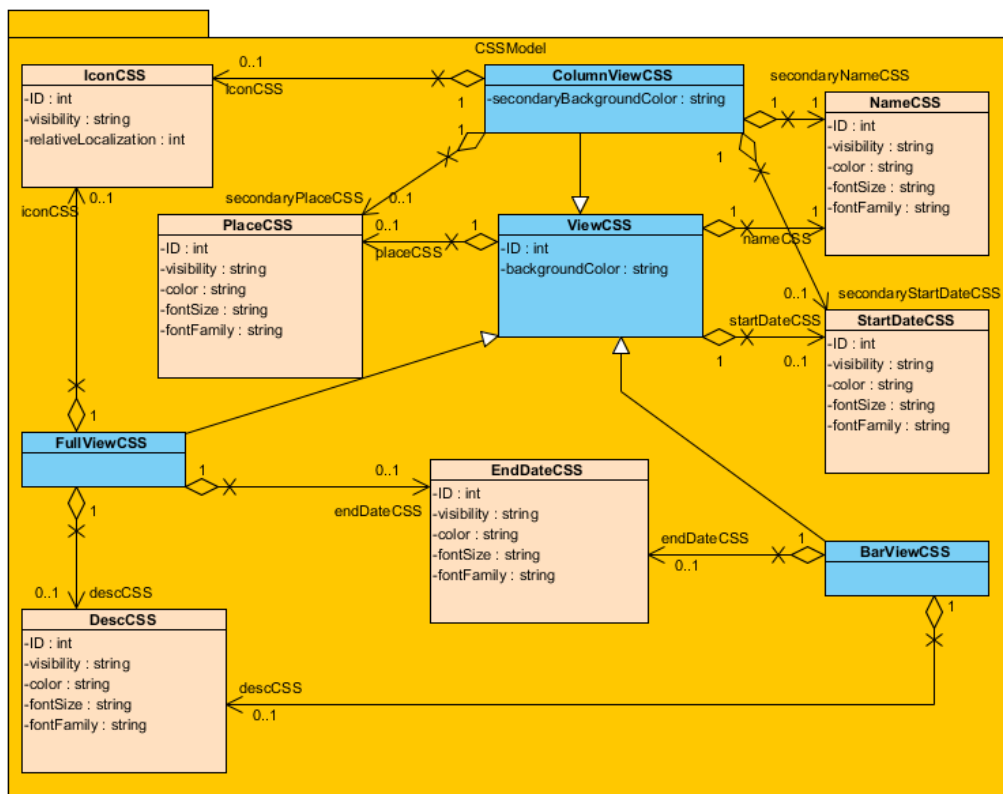
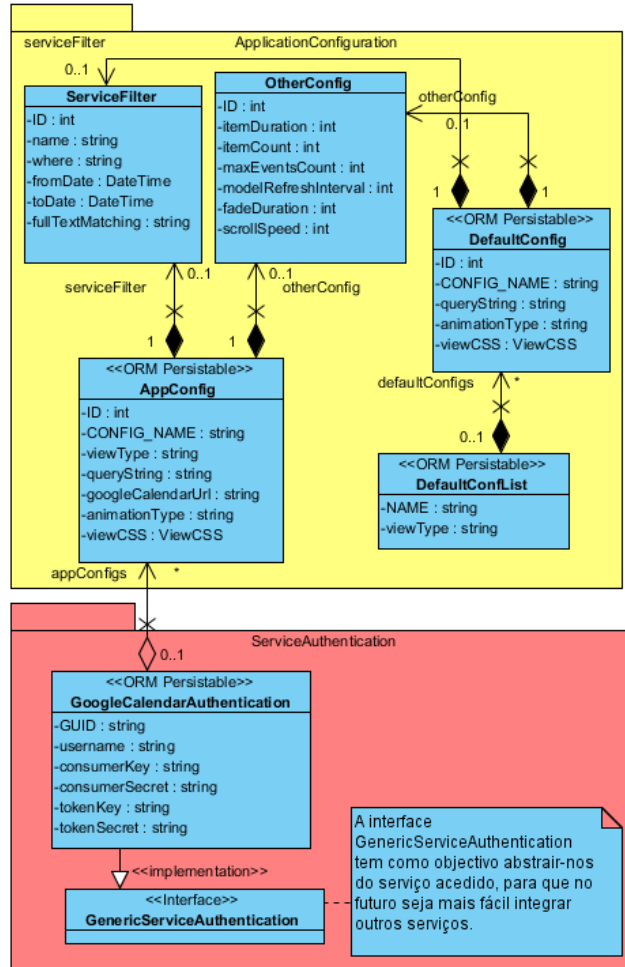


Figura 4.33: Diagrama de classes com os parâmetros da componente visual



**Figura 4.34:** Diagrama de classes constituído por dois packages. O package ApplicationConfiguration representa as classes com as configurações de visualização e o package ServiceAuthentication representa as classes com dados sobre a autenticação/autorização

dizer que contém as classes *controller*, considerando o padrão arquitetural Model-View-Controller, que interagem com os formulários de configuração para todos os tipos de visualização (Full, Column, Bar e XML). Também é de referir que quase todos os métodos têm um estereótipo para definir os que recolhem informação de um formulário de configuração (estereótipo «HttpPost») ou aqueles que apenas têm como função apresentar uma página *web* (estereótipo «HttpGet»).

Em primeiro lugar, vai existir uma interface simples para o utilizador poder fazer a sua autenticação/autorização no serviço Google Calendar. Assim, a classe *ConfigFormController* vai conter o método *OpenIdLogOn()* para efectuar a autenticação através do protocolo OpenID. Dentro desse mesmo método, logo após a autenticação ser feita, vai ser invocado o método privado, *OAuth(identifierUrl:string)*, que através do identificador de autenticação, *identifierUrl*, comunica com a Google, permitindo que a aplicação obtenha as credenciais de acesso aos calendários do utilizador, que constam no serviço Google Calendar. Uma vez que o utilizador já tenha sido autenticado e a aplicação tenha o certificado de autorização, o método *OAuthCallback()* permitirá o redireccionamento para os formulários de configuração. Para finalizar a análise da classe *ConfigFormController*, resta dizer que a variável de instância *tokenManager* serve para fazer a gestão do processo de autenticação/autorização.

A classe *XMLViewConfigController* tem como único objectivo apresentar visualizações XML, utilizando os dados preenchidos no formulário de configuração. Para tal, é utilizado o método *DisplayXMLView(coll:FormCollection)* que recebe os dados do formulário e devolve uma visualização XML.

As sub-classes *FVConfigController*, *CVConfigController*, *BVScrollConfigController* e *BVFadeConfigController* acrescentam funcionalidades aos métodos da super-classe *GenericConfigController*, apesar de, por uma questão de espaço, não terem sido representados esses métodos nas sub-classes. Esta super-classe contém funcionalidades comuns a todos os formulários de configuração (à excepção do formulário da visualização XML). As funcionalidades existentes são as já referidas operações de carregar, visualizar, remover, salvar e de actualizar configurações, com a adição dos métodos *GetChosenDefaultConfig(...)* e *GetYourChosenConfig(...)* para o preenchimento automático dos campos do formulário, após a selecção de uma configuração *default* ou de utilizador, respectivamente.

Quanto ao método *DisplayConfig(...)* da classe *GenericConfigController* da Figura 4.35, é importante referir que tem a função de construir uma *query string* através dos dados preenchidos no formulário. Essa *query string* é utilizada de modo a que a *DSEventApp* invoque um dos métodos *Index()* existentes numa das classes do *package* *ViewControllers* da Figura 4.36. Após esta invocação, a *DSEventApp* trata de processar a visualização através dos parâmetros recebidos da *query string*. Como consequência, é invocado o respectivo método com o sufixo *CSS* (com a excepção da visualização XML –

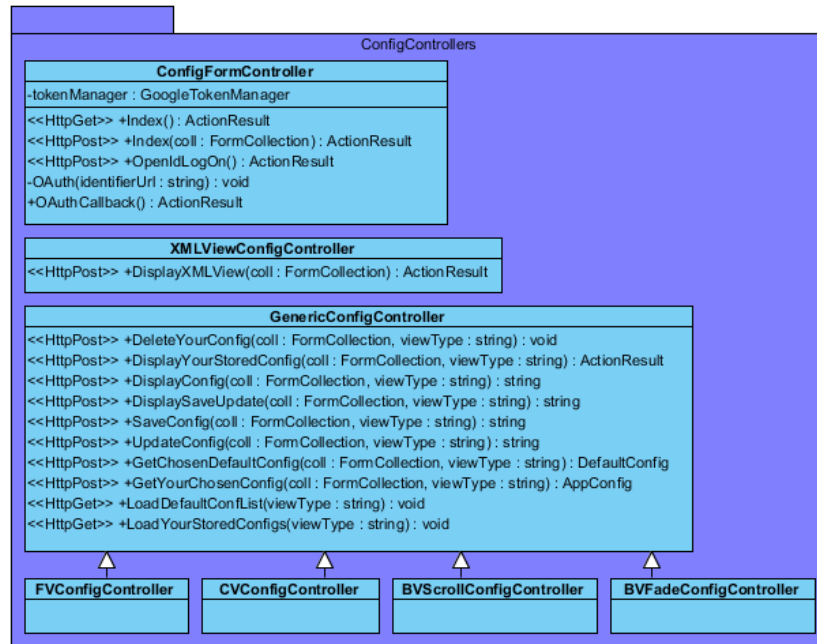


Figura 4.35: Diagrama de classes com o package ConfigControllers que contém classes (controllers) sobre os formulários de configuração

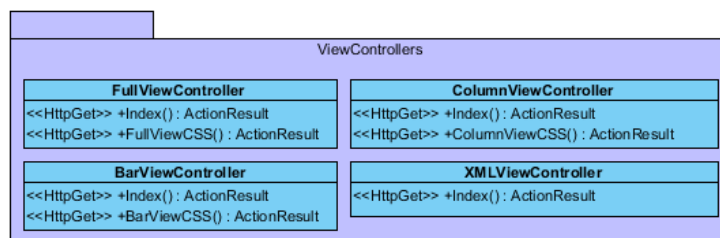
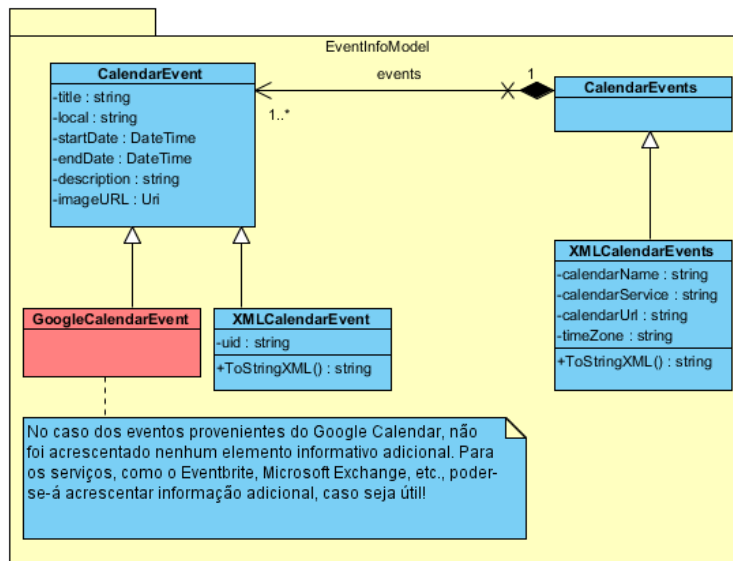


Figura 4.36: Diagrama com o package ConfigControllers que contém classes (controllers) sobre os formulários de configuração



**Figura 4.37:** Diagrama com o package *EventInfoModel* que contém classes com informação de eventos e do respectivo calendário (no caso da visualização XML)

ver classe *XMLViewController* da Figura 4.36 – pois esta não contém nenhum elemento decorativo) para a *DSEventApp* processar um ficheiro CSS, com base nos parâmetros da componente visual.

Para acabar o tópico das classes *controller*, resta referir que o package *ViewControllers* contém todas as classes com a função de processar as visualizações, utilizando *query strings* com vários parâmetros de configuração. Assim, vai ser possível especificar as *query strings* através de uma aplicação externa ou aproveitar o que já foi definido através do formulário de configuração e fazer o carregamento de uma dada configuração, utilizando o parâmetro *LoadConfig* (ver Tabela 4.6 da secção 4.2).

Avançando para a análise do package *EventInfoModel* da Figura 4.37, pode-se observar que irá existir uma classe chamada *CalendarEvents*, que contém uma lista de eventos (lista com objectos do tipo *CalendarEvent*) com a respectiva informação (título, local, data/hora, descrição e URL de uma imagem).

Também é relevante dizer que, por uma questão de organização, a classe *CalendarEvent* vai ser estendida pela classe *GoogleCalendarEvent*. No caso do Google Calendar, não se acrescentou informação sobre eventos, no entanto para outros serviços poder-se-á adicionar outros dados que sejam úteis e que sejam suportados pela API fornecida.

Por fim, a classe *XMLCalendarEvents* (que contém eventos do tipo *XMLCalendarEvent*) tem a função de guardar informação específica de um calendário e dos respectivos eventos, e contém o método *ToStringXML()* para gerar a visualização XML.

Considerando o *package* do diagrama da Figura 4.38, podem-se destacar as classes `EventManagerAllServices`, `GenericEventQueries` e `GoogleCalendarManager`. Começando pela última, verifica-se que diz respeito apenas ao serviço Google Calendar e que vai utilizar a API Google Calendar com o fim de efectuar *queries*. Deste modo, realiza-se a filtragem de eventos de acordo com o que o utilizador especificou no formulário.

Tendo ainda em conta a classe `GoogleCalendarManager`, pode-se observar dois métodos privados, `ListGoogleCalendarEvents(...)` e `ListGoogleCalendarEventsXML(...)` (invocados, respectivamente, pelos métodos `QueryGoogleCalendar(...)` e `QueryGoogleCalendarXML(...)`), com a função de encapsular a API do Google Calendar e de devolver os eventos de acordo com os filtros suportados pelo serviço.

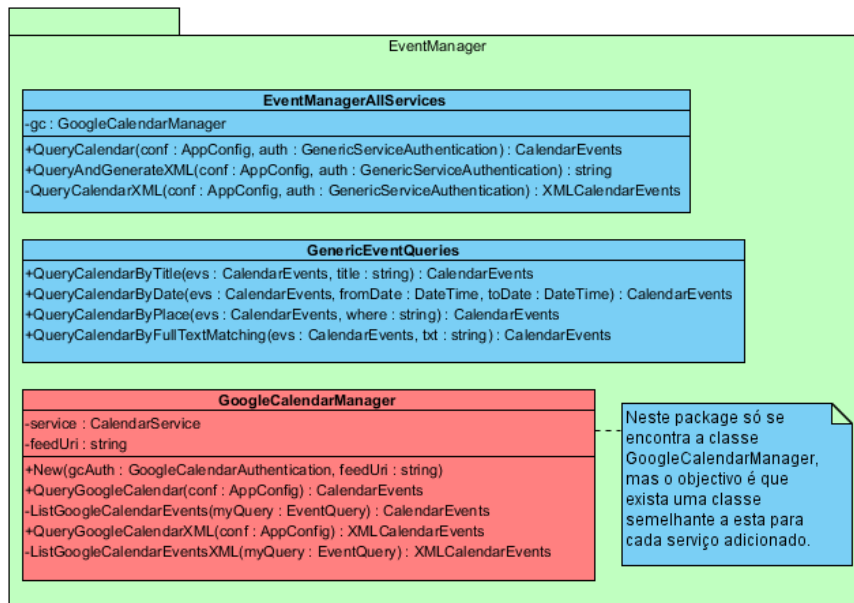
Como a API Google Calendar não suporta filtragem por título nem por local, os métodos `QueryGoogleCalendar(...)` e `QueryGoogleCalendarXML(...)` tratam de recorrer aos métodos `QueryCalendarByTitle(...)` e `QueryCalendarByPlace(...)`, da classe `GenericEventQueries`, para fazer a pós-filtragem, uma vez que há eventos, retornados pelo Google Calendar, que devem ser filtrados. Os métodos `QueryCalendarByDate(...)` e `QueryCalendarByPlace(...)` também estão incluídos na classe `GenericEventQueries`, com o fim de realizar a filtragem pretendida, caso as APIs de outros serviços não suportem esses critérios de filtragem.

Para terminar a análise do *package* `EventManager` da Figura 4.38, resta falar da classe `EventManagerAllServices`. Esta tem a função de encapsular os serviços que são acedidos na realização de *queries* e, por isso, os métodos `QueryGoogleCalendar(...)` e `QueryGoogleCalendarXML(...)`, da classe `GoogleCalendarManager`, são invocados, respectivamente, pelos métodos `QueryCalendar(...)` e `QueryCalendarXML(...)`, da classe `EventManagerAllServices`.

Para concluir a análise da lógica de negócio, falta descrever o *package* `Utils` da Figura 4.39, onde se observa que a classe `BrowserCulture` tem apenas um método para retornar a cultura do *browser*, de modo a que a `DSEventApp` apresente um formato adequado para as datas/horas, isto é, um formato que tenha em conta a cultura da região (por exemplo, nos Estados Unidos o formato da data é mês/dia/ano).

A classe `DateTimeConv` contém métodos utilitários de conversão de datas. Enquanto que o método `ConvertDateTime2Seconds(...)` converte uma data/hora em segundos, o método `ConvertSeconds2DateTime(...)` faz o inverso. O método `ConvertDateString2Seconds(...)` converte uma data num formato fácil de entender (por exemplo: "01-Jan-2011") em segundos e, por fim, o método `ConvertDateTime2DateString(...)` converte uma data numa *string* com o formato já referido.

Quanto à classe `ParsingUtilities`, é importante referir que tem quatro métodos que processam texto para ultrapassar problemas específicos. Os dois primeiros, `InputBrTagsInNewLines(...)` e `InputNBSPsInNewLines(...)`, substituem os caracteres de mudança de linha, existentes nas descrições de cada



**Figura 4.38:** Diagrama com o package *EventManager* que contém classes com a funcionalidade de efectuar queries e retornar eventos

evento, por *tags* `<br />` ou por espaços fixos, “&nbsp;”, respectivamente. Os dois últimos têm a função de ultrapassar uma grande limitação do Google Calendar, que consiste no facto de o serviço não suportar a introdução de imagens para eventos por via manual (apenas é possível introduzi-las programaticamente). De modo a que a *DSEventApp* ultrapasse esse problema, assume-se que o URL da imagem vai constar na descrição, entre as *tags* `<img>` e `</img>`. Então, o método `ObtainImgUrlFromDescription(...)` obtém apenas o URL da imagem, proveniente da descrição, e o método `ObtainAllFromDescriptionExceptImgUrl(...)` obtém a descrição sem o URL.

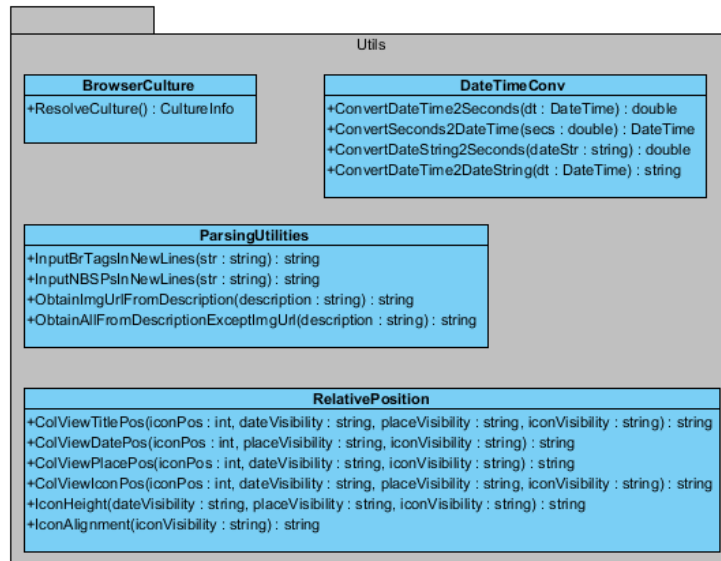
Por fim, a classe *RelativePosition* contém métodos que determinam o posicionamento relativo dos elementos de cada evento para a visualização *Column*. Para haver um melhor esclarecimento das suas funcionalidades, sugere-se a leitura da descrição do parâmetro *EventIconRelativeLocalization*, existente na Tabela 4.3, da secção 4.2.

### 4.3.3 Camada de Dados

A camada de acesso à base de dados vai ter um peso muito importante na *DSEventApp*, uma vez que vai permitir persistir e reutilizar configurações especificadas pelo utilizador, facilitando muito o trabalho. Além disso, vão existir configurações *default* para o utilizador as reutilizar ou até as alterar consoante as suas preferências.

Recorre-se à *framework* *NHibernate* para que haja um mapeamento bas-





**Figura 4.39:** Diagrama com o package Utils que contém classes com utilidades diversas

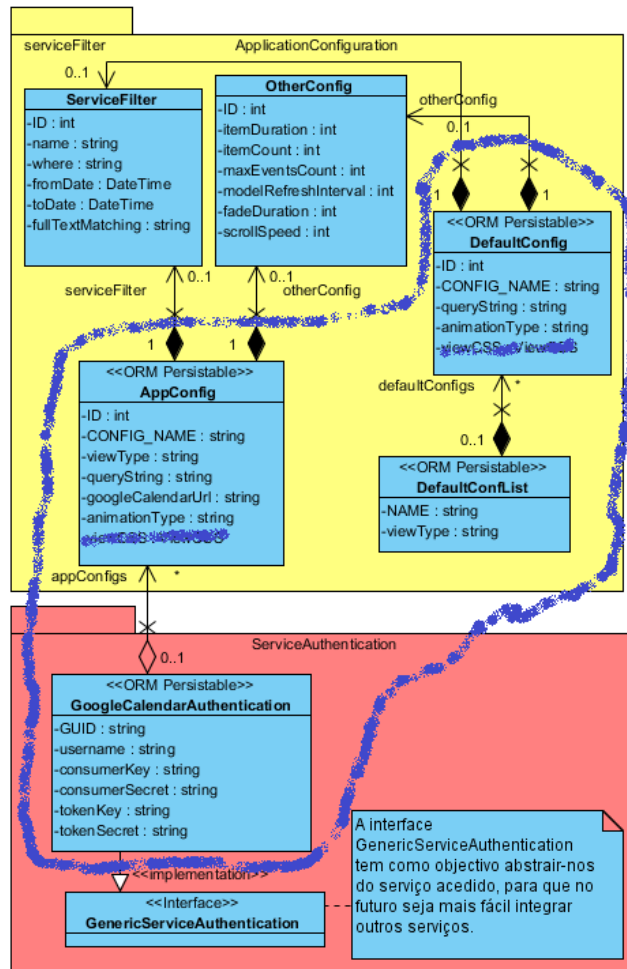
tante directo entre objectos e relações (mapeamento ORM) e utiliza-se o Visual Paradigm, que é uma ferramenta para modelar diagramas, mas que também permite realizar o mapeamento ORM (em termos de diagramas) e gera o código NHibernate das entidades persistentes e respectivas relações.

De seguida, apresenta-se o diagrama de classes que contém as entidades persistentes a serem mapeadas para relações ou tabelas. Observando este diagrama na Figura 4.40, podem-se destacar quatro classes persistentes, delimitadas por uma linha azul: GoogleCalendarAuthentication, AppConfig, DefaultConfList e DefaultConfig. No entanto, é de notar que as classes AppConfig e DefaultConfig contém uma variável de instância, *viewCSS*, que foi riscada a azul, uma vez que não vai ser considerada no mapeamento ORM. Na Figura 4.41, pode-se observar o diagrama ER obtido através do mapeamento ORM das quatro classes referidas.

A relação GoogleCalendarAuthentication (Figura 4.41) contém toda a informação necessária de autenticação e de autorização de acesso ao serviço Google Calendar.

A DSEventApp vai proporcionar comunicação (via OpenID) com a Google, com o fim de permitir ao utilizador inserir os seus dados de autenticação (*username* e *password*) na sua conta da Google. Uma vez autenticado, a Google fornece o *token* de resposta para a DSEventApp. Assim, a aplicação já tem a forma de saber a identificação do utilizador e, ao mesmo tempo, não armazenar os seus dados confidenciais.

Na primeira vez que o utilizador efectua a sua autenticação, é feito um redireccionamento para outra página da Google, com o fim do utilizador



**Figura 4.40:** Diagrama de classes constituído pelos packages *ApplicationConfiguration* e *ServiceAuthentication*, onde se realçam as classes persistentes

definir se permite ou não que a DSEventApp tenha a autorização de acesso ao serviço Google Calendar. Após o utilizador autenticar-se e dar a sua permissão de acesso, são armazenados o *token* de autenticação (campo *GUID* da tabela *GoogleCalendarAuthentication* da Figura 4.41), o seu *username* e as credenciais de acesso (campos *ConsumerKey*, *ConsumerSecret*, *TokenKey* e *TokenSecret*).

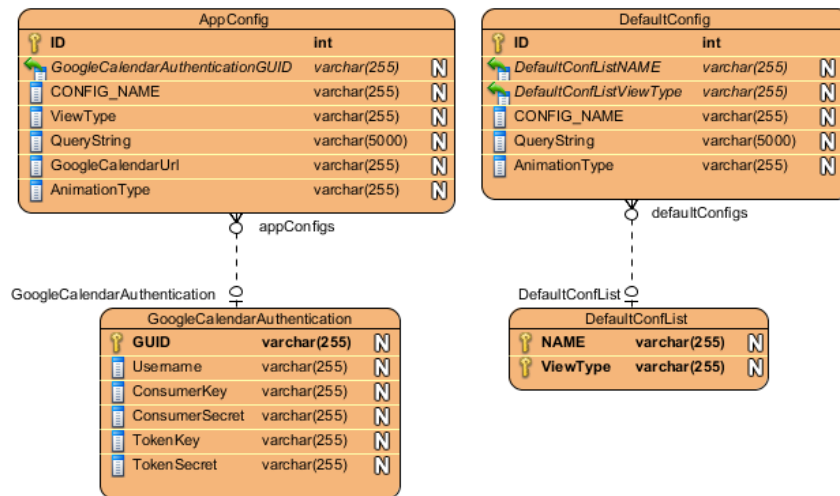
Quanto ao campo *ConsumerKey*, é de referir que se trata do domínio da aplicação DSEventApp e o campo *ConsumerSecret* corresponde a um identificador obtido pela Google para o domínio registado. No entanto, a DSEventApp vai adoptar, por defeito, o valor *anonymous* para esses dois campos, uma vez que a Google permite que o domínio não seja conhecido. De qualquer forma, o domínio sempre poderá ser especificado depois da fase de instalação da DSEventApp.

Concluindo a análise da tabela *GoogleCalendarAuthentication* da Figura 4.41, as colunas *TokenKey* e *TokenSecret* servem para armazenar as credenciais de acesso ao Google Calendar. Trata-se de dois *tokens* que confirmam a autorização dada pelo utilizador para que a DSEventApp possa ter acesso total aos calendários e respectivos eventos do utilizador.

Avançando para a tabela *AppConfig*, que armazena as configurações de utilizador, pode-se observar os seguintes campos:

- *ID*: chave primária que identifica uma certa configuração de utilizador;
- *GoogleCalendarAuthenticationGUID*: chave estrangeira que referencia a chave primária *GUID* da tabela *GoogleCalendarAuthentication* e que tem como objectivo identificar o dono da configuração;
- *CONFIG\_NAME*: nome da configuração a ser armazenada;
- *ViewType*: tipo de visualização a ser considerada (Full, Column ou Bar);
- *QueryString*: *query string* com todos os parâmetros de configuração necessários para a DSEventApp apresentar a visualização do tipo *ViewType*. É de notar que não se recorre a mais que uma tabela para armazenar esses parâmetros, por questões de eficiência;
- *GoogleCalendarUrl*: URL do calendário utilizado para a configuração armazenada;
- *AnimationType*: tipo de animação utilizada para visualização Bar, isto é, se se trata de um rodapé rolante (visualização Bar Scroll) ou de um rodapé *fade-in/fade-out* (visualização Bar Fade).

Salienta-se que, tal como acontece nas tabelas *GoogleCalendarAuthentication* e *AppConfig*, as tabelas *DefaultConfList* e *DefaultConfig* estabelecem uma relação de um para muitos, respectivamente.



**Figura 4.41:** Diagrama ER obtido através de classes persistentes (estereótipo «ORM Persistable») apresentadas na Figura 4.40

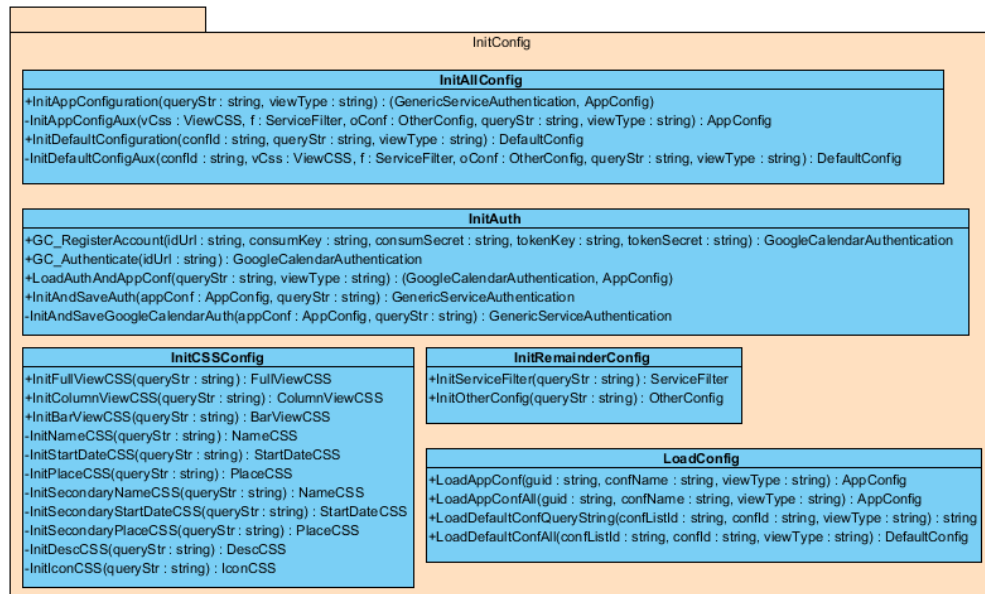
Analisando os campos existentes na tabela `DefaultConfList`, destaca-se o `NAME`, que é o nome da lista de configurações *default*, e o `ViewType`, que é o tipo de visualização associado à lista. É de relevar que os campos `NAME` e `ViewType` compõem a chave primária da tabela `DefaultConfList`.

Para finalizar a análise do diagrama ER da Figura 4.41, falta descrever os campos da tabela `DefaultConfig`. O campo `ID` é chave primária, identificando, assim, unicamente cada configuração *default* armazenada. Por sua vez, os campos `DefaultConfListNAME` e `DefaultConfListViewType` são chave estrangeira, referenciando os campos `NAME` e `ViewType` da tabela `DefaultConfList`. O campo `CONFIG_NAME` é o nome da configuração *default* pertencente a uma lista, o `QueryString` corresponde à *query string* com todos os parâmetros de configuração e o `AnimationType` define o tipo de animação para a visualização `Bar`.

Resta analisar o último diagrama de classes (*package* `InitConfig` da Figura 4.42) que tem a função de inicializar parâmetros de configuração a partir de uma *query string*. Esta pode ser obtida a partir da base de dados (tabela `AppConfig` ou `DefaultConfig`) ou através de parâmetros introduzidos num formulário de configuração.

Quanto ao *package* `InitConfig` da Figura 4.42, vai ser utilizada uma abordagem de baixo para cima, no que diz respeito à análise das suas classes, assim:

- A classe `InitCSSConfig` tem como objectivo inicializar todos os parâmetros da componente visual, isto é, as instâncias de classes com o sufixo `CSS`, para posteriormente serem utilizadas em ficheiros de estilo `CSS`;



**Figura 4.42:** Diagrama de classes do package *InitConfig*, com a função principal de inicializar configurações

- A classe *LoadConfig* tem métodos para inicializar configurações de utilizador (métodos *LoadAppConf(...)* e *LoadAppConfAll(...)*) e métodos para inicializar configurações *default* (métodos *LoadDefaultConfQueryString(...)* e *LoadDefaultConfAll(...)*);
- A classe *InitRemainderConfig* contém apenas dois métodos com a função importante de inicializar todos os parâmetros da componente de filtragem (método *InitServiceFilter(...)*) e os restantes parâmetros (método *InitOtherConfig(...)*);
- A classe *InitAuth* é composta por dois métodos (*GC\_RegisterAccount(...)* e *GC\_Authenticate(...)*) exclusivos para o serviço Google Calendar, sendo que o primeiro serve para armazenar na base de dados a informação sobre os dados de autenticação/autorização e o segundo tem como função carregar uma conta de utilizador já existente na base de dados. O método *LoadAuthAndAppConf(...)* tem como objectivo inicializar uma instância da classe *GoogleCalendarAuthentication* e uma outra instância da classe *AppConfig* (que podem ser observadas na Figura 4.40), a partir dos dados obtidos pela *query string*. É de relevar que, com base no *token* de autenticação (parâmetro *Guid* da *query string*), é efectuada uma *query* à tabela *GoogleCalendarAuthentication*, com o objectivo de inicializar, de uma forma completa, a instância da classe *GoogleCalendarAuthentication*. Para concretizar a completa análise desta classe, resta dar destaque ao método *AssociateApp-*

`Conf2AuthAndSave(...)`, que tem a função de atribuir uma configuração para o respectivo utilizador, ou seja, é preenchida a tabela `AppConfig` com uma configuração de utilizador e é efectuada uma referência para um registo (do utilizador em questão) da tabela `GoogleCalendarAuthentication`. De notar que quando a `DSEventApp` suportar outros serviços, para além do Google Calendar, terá que existir outras tabelas de autenticação para esses mesmos serviços;

- A classe `InitAllConfig` tem a função de invocar a maior parte dos métodos das restantes classes do *package* `InitConfig`, mais concretamente os métodos que inicializam os parâmetros de configuração da componente visual, comportamental e de filtragem. Após esse processo, a `DSEventApp` já pode apresentar uma visualização `Full`, `Column` ou `Bar`.

## 4.4 Implementação da Aplicação

A `DSEventApp`, como a maioria das soluções de *software*, tem em conta os requisitos funcionais e não funcionais, com o objectivo de ser usável por vários utilizadores, uma vez que se trata de uma aplicação *web*. Durante a fase de desenvolvimento da `DSEventApp` foram efectuados testes, de modo a evitar que os *bugs* se tornassem mais difíceis de corrigir, numa fase posterior.

Nesta secção abordar-se-á o processo de implementação da `DSEventApp`, ou seja, tudo o que foi efectuado com a ajuda da especificação até se obter a solução de *software* pretendida. Também vão ser analisadas algumas decisões tomadas e determinados problemas ocorridos durante a fase de desenvolvimento. Na última subsecção será dado um exemplo para cada visualização (`Full`, `Column`, `Bar`, `Bar Fade`, `Bar Scroll` e `XML`) que a `DSEventApp` suporta.

### 4.4.1 Processo de Implementação

Para que houvesse uma boa organização no período de desenvolvimento da `DSEventApp`, foi estabelecido, em conjunto com a `Ubisign`, um plano de trabalhos constituído por fases de desenvolvimento e os respectivos prazos.

De modo a que se tenha uma ideia geral sobre o processo de implementação, destacam-se as tarefas concretizadas durante o desenvolvimento da `DSEventApp`:

- *Protótipo da visualização Full*: Foi implementada a visualização `Full`, sem preocupações quanto ao processamento de configurações, isto é, a `DSEventApp` apenas apresentava uma única visualização `Full` com as componentes visual, comportamental e de filtragem totalmente pré-definidas e não parametrizáveis. Além disso, a informação de autenticação estava num parâmetro de configuração estático da `DSEventApp` e o tipo de autenticação era o convencional, ou seja, o `username` e a

*password* encontravam-se *hardcoded* na própria aplicação, de forma a que esta pudesse interagir com o serviço Google Calendar;

- *Suporte integral à visualização Full*: Foi desenvolvida a totalidade da visualização Full, isto é, a DSEventApp já processava todos os parâmetros de configuração que lhe eram passados via *query string*. Adicionalmente, já existia uma base de dados para que fosse possível enviar, por *query string*, um *token* de autenticação à DSEventApp, de modo a que esta obtivesse a respectiva informação de autenticação da base de dados;
- *Suporte integral à visualização Column*: Foi implementada a visualização Column no seu todo, ou seja, a DSEventApp já permitia o processamento de uma visualização Column com base nos parâmetros de configuração que lhe eram passados. Em acréscimo, tanto a visualização Column como a Full já eram capazes de manter o estado entre os pedidos (manutenção de sessão), de modo que as iterações continuassem a partir do ponto em que tinham ficado no último pedido;
- *Suporte integral à visualização Bar*: Foi desenvolvida a totalidade da visualização Bar capaz de processar todos os parâmetros que lhe eram passados, ou seja, implementou-se a visualização Bar Scroll que já apresentava informação sobre eventos em rodapé rolante, segundo um parâmetro de configuração que especifica a velocidade do deslizamento de texto. Adicionalmente, concretizou-se a visualização Bar Fade em que é fornecida a informação essencial de cada evento, com base num intervalo de tempo, especificado pelo utilizador, durante o qual cada evento é apresentado;
- *Suporte integral à visualização XML*: Foi concretizada a visualização XML que corresponde ao resultado de uma *query* de filtragem ao serviço Google Calendar e, deste modo, os eventos são apresentados no formato XML, com o objectivo de serem facilmente processados por outras aplicações;
- *Formulários de configuração*: Foi desenvolvido um formulário de configuração para cada tipo de visualização, permitindo a criação de configurações por parte do utilizador. Outras operações como persistir, actualizar e remover configurações também são permitidas através dos formulários. Adicionalmente, a DSEventApp suporta configurações *default*, de modo que o utilizador possa aproveitá-las ou, até mesmo, personalizá-las;
- *Solução de autenticação segura*: Por fim, foi disponibilizada uma solução de autenticação fidedigna através do protocolo OpenID e foi usado

o protocolo OAuth para que a DSEventApp aceda aos eventos, com a autorização do utilizador.

#### 4.4.2 Problemas e decisões tomadas durante a fase de desenvolvimento

Ao longo do processo de desenvolvimento da DSEventApp, é normal que se tenham que tomar decisões de implementação, com o objectivo de superar obstáculos que vão surgindo. Nesta subsecção vão ser descritas algumas das decisões tomadas e certos problemas encontrados.

##### Embutir código C# em ficheiros CSS

Com o objectivo de tornar o código do processamento das visualizações o mais legível possível, adoptou-se a solução de embutir código C# em cada ficheiro CSS (cada ficheiro corresponde a um tipo de visualização) para parametrizar as suas propriedades. No entanto, a solução mais óbvia, mas menos organizada, seria a de utilizar a API do JavaScript com o fim de parametrizar dinamicamente as propriedades CSS.

##### Divisão dos formulários de configuração em componentes – *Partial Views*

Ainda no que diz respeito à organização do código, na questão do desenvolvimento dos formulários de configuração, e dado que estes são relativamente grandes, resolveu-se efectuar uma divisão das suas componentes. Como a tecnologia ASP.NET o permite, foram utilizadas várias *partial views* (vistas parciais) com o intuito de tornar o código mais modulado.

A utilidade das vistas parciais é muito grande, uma vez que evita a repetição desnecessária do mesmo código HTML. Além disso, um aspecto ainda mais importante é o facto de sempre que haja mudanças no código HTML, não ter que se efectuar as alterações em vários ficheiros e, assim, basta efectuá-las num único ficheiro partilhado (vista parcial).

Um exemplo que ilustra a grande vantagem das vistas parciais é o dos estilos de fonte. Como este caso particular é utilizado em quase todos os formulários de configuração (exceptuando o formulário para a visualização XML), e pode estar sempre sujeito a alterações (por exemplo, acrescentar novos estilos de fonte), torna-se bastante útil separar essa componente num ficheiro à parte, para ser partilhado.

##### Calendários para a componente de filtragem


Para o utilizador não ter dificuldade a introduzir as datas de início e de fim, e com objectivo de efectuar *queries* ao Google Calendar, optou-se por introduzir um *widget* em JavaScript para apresentar um calendário. Assim,




**Service Filter**

Title =

Place =

Start Date =  

End Date =  


Full Text Ma

**Other Co**

Item Duration  1000 (milliseconds)

Number of E  1

Max. Numbe  5



The calendar widget shows the month of August 2011. It has a title bar with a close button (X) and navigation arrows. The days of the week are listed as Mon, Tue, Wed, Thu, Fri, Sat, Sun. The dates 1 through 31 are arranged in a grid. The date 29 is highlighted in red.

**Figura 4.43:** Calendário para a componente de filtragem

o utilizador poderá seleccionar um dia do mês e a DSEventApp reconhece a data introduzida sem haver problemas com erros de formatação.

Na Figura 4.43 pode-se ver um calendário bastante simples com os botões de avançar mês (uma seta) e de avançar ano (dupla seta). Quando o utilizador clicar num dia do mês, aparece a data na respectiva caixa de texto, no formato “dia-mês-ano” (por exemplo, “01-JAN-2011”).

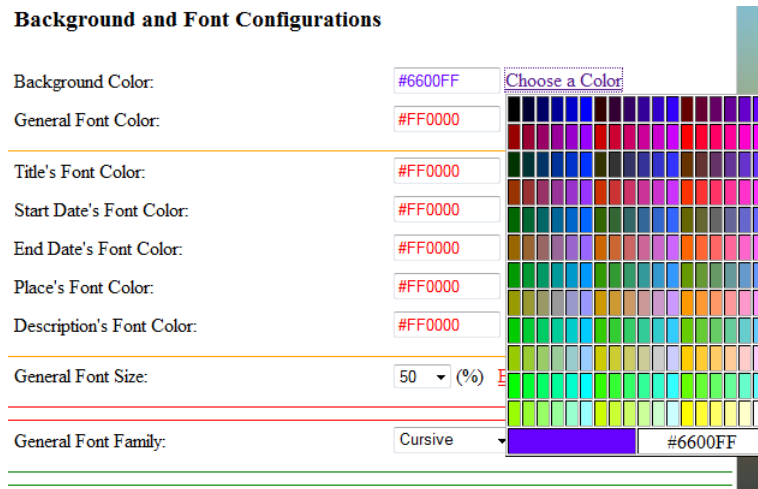
### **Color Picker para seleccionar cores**

Com o objectivo de ser fácil a introdução de cores de fundo e de fonte no formulário de configurações, é disponibilizado um *widget* designado por *Color Picker*. Como se pode observar na Figura 4.44, o *Color Picker* tem as cores mais frequentes de utilização, bastando clicar numa cor para preencher a caixa de texto com o valor RGB correspondente (valor de 3 *bytes* em hexadecimal que determinam a intensidade das componentes vermelho, verde e azul).

Ainda considerando a Figura 4.44, é visível que cada valor em hexadecimal contém como prefixo o carácter ‘#’ e, por isso, houve problemas com esse carácter ao ser utilizado em *query strings*. Como solução de remendo, a DSEventApp consome esses valores com o prefixo ‘\_’ em vez de ‘#’, e no processamento desses valores volta a colocar de volta o ‘#’, para que da parte dos ficheiros CSS se reconheça o formato de cor em hexadecimal, de modo a não haver qualquer tipo de ambiguidade com outros formatos.

### **Utilização de AJAX para preencher o formulário**

No que diz respeito à selecção de uma configuração persistida e ao consequente preenchimento automático de campos do formulário de configurações, decidiu-se recorrer à tecnologia AJAX para preencher o formulário de uma forma mais rápida e eficiente, isto é, sem ter que fazer *postback* à página do



**Figura 4.44:** *Color Picker para seleccionar cores de fundo ou de fonte*

formulário, o que provoca o carregamento integral (desnecessário) da mesma página com os valores da configuração seleccionada.

### Esclarecimento sobre os parâmetros de visibilidade/alinhamento

Tendo em consideração a Figura 4.1, existente na secção 4.3 e onde se aborda o tópico dos esboços para a visualização Full, é importante voltar a realçar que o *layout* do painel digital vai estar dividido em duas zonas (título e corpo do evento) e, por sua vez, o corpo do evento vai estar dividido em duas sub-zonas (ícone e dados).

O parâmetro `EventIconVisibility` vai determinar qual o posicionamento da sub-zona do ícone relativamente à sub-zona dos dados (datas, local e descrição). Se tiver o valor *Left* ou *Right*, a sub-zona do ícone aparece, respectivamente, à esquerda ou à direita, o que faz com que a sub-zona dos dados apareça em lados inversos. No entanto, a necessidade de parâmetros de alinhamento específicos para cada um dos dados, justifica-se pelo facto de, dentro da sub-zona reservada aos dados, o utilizador poder querer alinhamentos diferentes para as datas, descrição, etc.. Esse alinhamento é sempre relativo à própria sub-zona e não ao *layout* geral. Por exemplo, se `EventIconVisibility=Left` e `EventPlaceVisibility=Left`, o local do evento vai ficar à direita do ícone, mas alinhado à esquerda na sua sub-zona.

### Especificação de *query strings* durante o desenvolvimento

Ao longo da fase de desenvolvimento houve a especificação de *query strings* para testar, de uma forma mais exaustiva, os casos possíveis de configuração e, assim, saber se todos os parâmetros funcionam correctamente. Esse conjunto alargado de configurações encontram-se em ficheiros *.doc* e não se

encontram disponíveis nesta dissertação, uma vez que ocupam muito espaço e são pouco legíveis (são *query strings* de grande dimensão, devido ao elevado número de parâmetros de configuração).

Com o intuito de aproveitar as configurações especificadas, foi desenvolvido um *parser* para extrair *query strings* de ficheiros *.doc* e armazenar, de seguida, na base de dados como configurações *default*. Alarga-se, assim, de uma forma significativa o lote de configurações nativas e estas têm títulos sugestivos de como a configuração se comporta. Deste modo, o utilizador poderá escolher, de uma forma rápida, a configuração que pretende e alterá-la caso considere pertinente.

### 4.4.3 Exemplos de utilização da aplicação

Esta subsecção tem em vista destacar alguns exemplos de utilização da DSEventApp, isto é, vai-se dar ênfase ao modo como a DSEventApp funciona, com base em exemplos sobre o processo de autenticação/autorização e sobre as visualizações Full, Column, Bar e XML.

#### Processo de Autenticação/Autorização

Como já foi dito anteriormente, para o utilizador ter acesso total aos recursos que a DSEventApp disponibiliza, é necessário efectuar a sua autenticação e dar a autorização de acesso ao serviço Google Calendar.

Na Figura 4.45 pode-se observar o menu principal com os serviços que a aplicação poderá vir a suportar, sendo que o único serviço disponível, por enquanto, é o Google Calendar (serviço escolhido para este projecto).

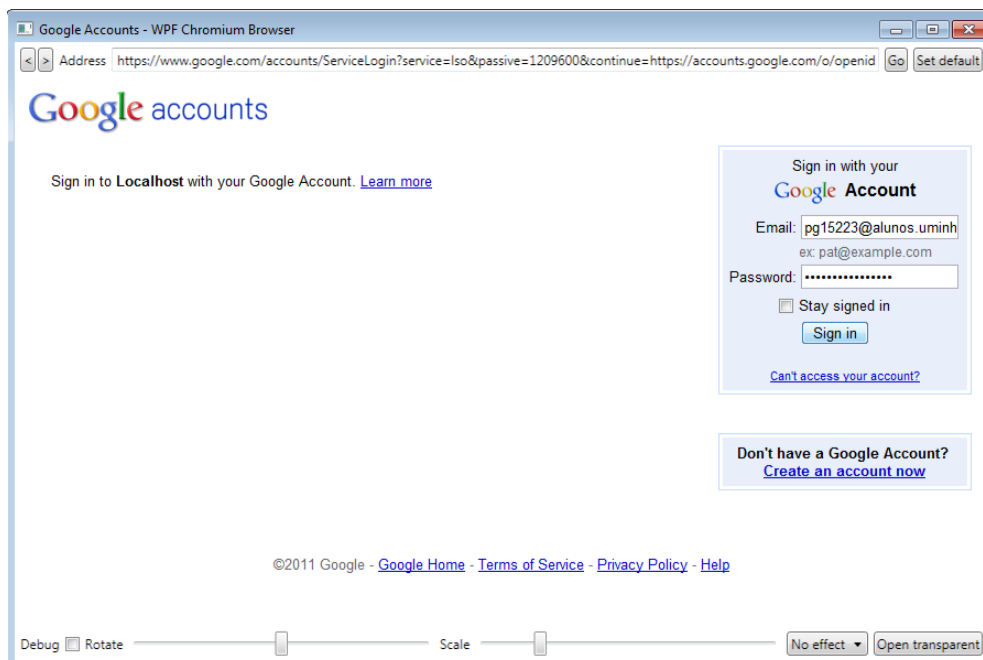
É de notar que os botões para aceder aos formulários de configuração das visualizações disponibilizadas, encontram-se inicialmente bloqueados, uma vez que, em primeiro lugar, o utilizador deve escolher o serviço com o qual pretende autenticar-se e permitir que a DSEventApp tenha um certificado para aceder aos calendários do utilizador.

Clicando no ícone da Google, existente no menu principal (Figura 4.45), é apresentada a bem conhecida página *web* que a Google fornece para o utilizador se autenticar, como se pode observar na Figura 4.46. A página *web* da Figura 4.47, diz respeito a outro passo para o utilizador efectuar a autenticação. Quanto à mensagem “Está a iniciar a sessão no Localhost com a sua Conta do Google...”, surge devido à DSEventApp estar a correr na própria máquina, mas, numa situação real de uso, irá aparecer o nome do domínio em vez de *localhost*.

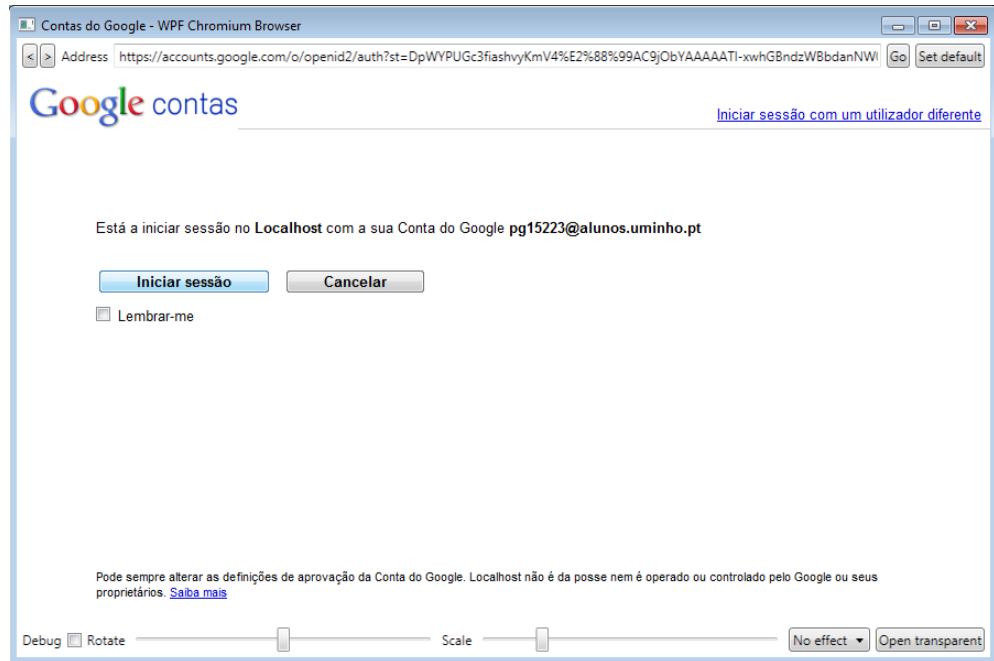
Com vista a que o utilizador permita o acesso da DSEventApp à sua conta do Google Calendar, basta clicar no botão *Grant access* (considerar Figura 4.48). No entanto, este passo é omitido quando o utilizador já tiver fornecido esta autorização, uma vez que a DSEventApp já armazenou na base de dados as credenciais de acesso.



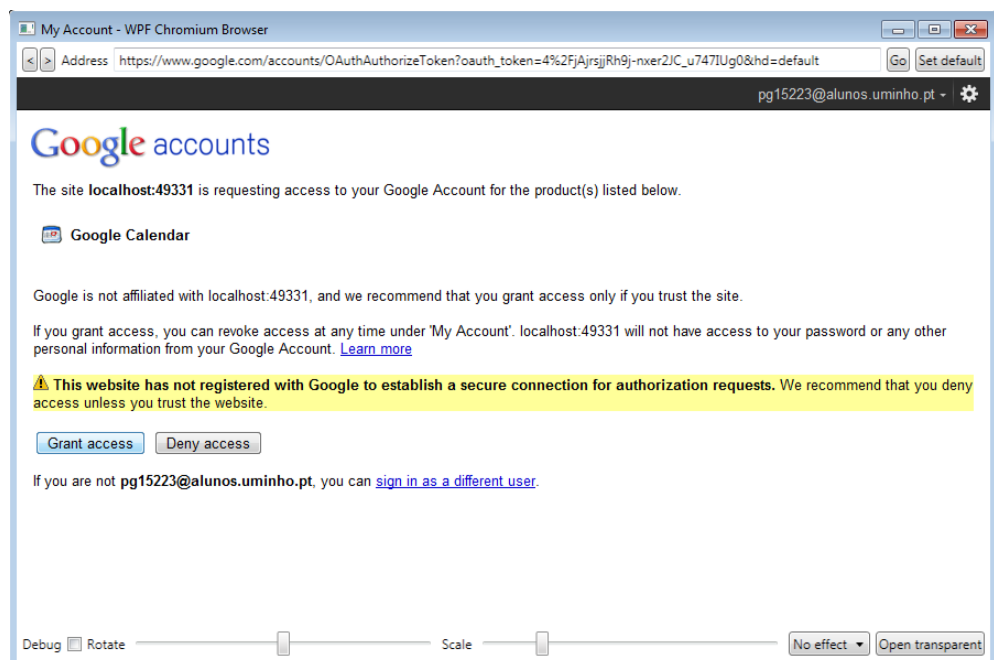
**Figura 4.45:** Menu principal para o utilizador se autenticar num serviço e poder escolher a visualização que pretende configurar (botões bloqueados)



**Figura 4.46:** Página para o utilizador efectuar a autenticação com uma conta Google



**Figura 4.47:** *Passo adicional para o utilizador efectuar a autenticação com uma conta Google*



**Figura 4.48:** *Página para o utilizador autorizar o total acesso da DSEventApp à sua conta do Google Calendar*



**Figura 4.49:** Menu principal para o utilizador se autenticar num serviço e poder escolher a visualização que pretende configurar (botões desbloqueados)

No fim deste processo, o utilizador já poderá aceder ao formulário de configuração pretendido, clicando num dos botões que foram desbloqueados (observar Figura 4.49).

### Visualização Full

Neste tópico vai-se apresentar um exemplo de visualização Full, processada pela DSEventApp. Também se vai mostrar a correspondente *query string* e, assim, dar-se uma ideia mais concreta dos parâmetros processados pela DSEventApp.

Na Figura 4.50 pode-se observar um dos eventos de uma visualização Full, apresentada no WPF Chromium WebBrowser (navegador *web* de código aberto utilizado pelo serviço Ubisign.com).

Considerando a Figura 4.51, observa-se a *query string* correspondente à visualização Full da Figura 4.50. Os dois parâmetros iniciais, Guid e GoogleCalendarURL, têm a função de permitir a autenticação do utilizador e de identificar o calendário pretendido, respectivamente. Também é importante dizer que esses dois parâmetros são fundamentais para que qualquer configuração torne possível o acesso a um calendário de utilizador.

Passando para os parâmetros que definem o aspecto da visualização, destacam-se os seguintes: o parâmetro BackgroundColor a laranja (cor de fundo laranja); os parâmetros sobre a visibilidade/alinhamento encontram-



Figura 4.50: Exemplo de visualização Full processada pela DSEventApp

se a negrito; os parâmetros sobre as propriedades da fonte (cor, tamanho e estilo de fonte) para o título dos eventos destacam-se a azul; as propriedades da fonte para as datas de início e fim encontram-se, respectivamente, a vermelho e a vermelho escuro; a verde destacam-se os parâmetros das propriedades da fonte para o local; a roxo destacam-se os parâmetros das propriedades da fonte para a descrição.

É de reparar que, para auxiliar a análise da *query string* da Figura 4.51, as cores utilizadas nos parâmetros ligados às propriedades de fonte estão relacionados com as cores dos elementos informativos de cada evento.

Na parte final da *query string*, pode-se visualizar, a azul claro, os parâmetros da componente comportamental. Neste caso, a duração de cada iteração é de 7 segundos ( $\text{ItemDuration}=7000$ ), o número de iterações é igual a 10 ( $\text{ItemCount}=10$ ), o máximo número de eventos retornados, pela *query* feita ao Google Calendar, é igual a 40 ( $\text{MaxEventsCount}=40$ ) e, por fim, a DSEventApp só deverá efectuar um pedido (equivalente a este) ao GoogleCalendar, após 10 minutos ( $\text{ModelRefreshInterval}=10$ ), uma vez que os eventos só serão refrescados depois desse intervalo.

Por fim, é de sublinhar que a *query string* da Figura 4.51 não contém parâmetros da componente de filtragem para não complicar ainda mais a sua análise. O único exemplo de *query string* que vai ter esse tipo de parâmetros, será o da visualização XML, uma vez que o número de parâmetros é bastante inferior, sendo, por isso, mais fácil a sua análise.

```
http://localhost:49331/FullView?Guid=AltOawklhMd6oTIW26-DumdFiQvRdYjnkR-
_GcA&GoogleCalendarURL=https://www.google.com/calendar/feeds/12c7gsp4rh0qesfr27hkjlq
1c%40group.calendar.google.com/private75dc608a01553284f0ae53f828d60726/full&Backgrou
ndColor=Orange&EventNameVisibility=Left&EventNameColor=Blue&EventNameFontSize=30
0&EventNameFontFamily=Cambria&EventStartDateVisibility=DateTimeCenter&EventStartD
ateColor=Red&EventStartDateFontSize=250&EventStartDateFontFamily=Calibri&EventEndDa
teVisibility=DateTimeCenter&EventEndDateColor=Maroon&EventEndDateFontSize=250&Eve
ntEndDateFontFamily=Calibri&EventPlaceVisibility=Center&EventPlaceColor=Green&EventP
laceFontSize=250&EventPlaceFontFamily=Calibri&EventDescVisibility=Center&EventDescC
olor=Purple&EventDescFontSize=250&EventDescFontFamily=Calibri&EventIconVisibility=Lef
t&ItemDuration=7000&ItemCount=10&MaxEventsCount=40&ModelRefreshInterval=10
```

**Figura 4.51:** Exemplo de query string de uma configuração para visualização Full

### Visualização Column

Na Figura 4.52 consta um exemplo de visualização Column processada pela DSEventApp. Tendo também como base a respectiva *query string* da Figura 4.53, verifica-se que a cor de fundo dos eventos ímpares é cinzento (BackgroundColor=Grey), enquanto que a dos pares é cinzento claro (SecondaryBackgroundColor=Lightgrey).

Quanto aos parâmetros que se encontram a negrito, é de destacar que servem para definir a visibilidade/alinhamento dos elementos informativos de cada evento, não havendo distinção quanto à sua paridade (se são ímpares ou pares). O parâmetro que poderá causar maior confusão é o EventIconRelativeLocalization. No entanto é bastante simples, pois define apenas a posição do ícone em relação ao título, data de início e local. Neste caso, EventIconRelativeLocalization=1 significa que o ícone se encontra na posição 1, sendo que o título posiciona-se na 0, a data de início na 2 e o local na 3. Na Figura 4.52, pode-se confirmar o posicionamento dos elementos informativos de cada evento.

Os parâmetros que definem as propriedades das fontes (cor, tamanho e estilo de fonte) da informação de cada evento, estão realçados com a cor correspondente. No entanto, é importante referir que, dentro do grupo desse tipo de parâmetros, existem dois sub-grupos. Um com o prefixo “Sec”, que afecta os eventos pares, e outro sem esse prefixo, mas com parâmetros equivalentes, onde apenas são afectados os eventos ímpares.

Quanto aos parâmetros da componente comportamental, destacados a azul claro, resta apenas dizer que o parâmetro ItemCount, aplicado à visualização Column, tem um significado distinto em relação aos outros tipos de visualização. É de notar que este parâmetro define o número de eventos apresentados em coluna. Observando a Figura 4.52, são apresentados 4 eventos em coluna (ItemCount=4).





Figura 4.52: Exemplo de visualização Column processada pela DSEventApp

[http://localhost:49331/ColumnView?Guid=AltOawklhMd6oTIW26-DumdFiQvRdYjnkR-\\_GcA&GoogleCalendarURL=https://www.google.com/calendar/feeds/l2c7gsp4rh0qesfr27hkjlq1c%40group.calendar.google.com/private75dc608a01553284f0ae53f828d60726/full&BackgroundColor=Grey&SecondaryBackgroundColor=Lightgrey&EventNameVisibility=Center&EventNameColor=Yellow&EventNameFontSize=100&EventNameFontFamily=Cambria&SecEventNameColor=Navy&SecEventNameFontSize=100&SecEventNameFontFamily=Cambria&EventStartDateVisibility=DateTimeLeft&EventStartDateColor=Orange&EventStartDateFontSize=100&EventStartDateFontFamily=Calibri&SecEventStartDateColor=Darkred&SecEventStartDateFontSize=100&SecEventStartDateFontFamily=Calibri&EventPlaceVisibility=Left&EventPlaceColor=LightGreen&EventPlaceFontSize=100&EventPlaceFontFamily=Calibri&SecEventPlaceColor=Darkgreen&SecEventPlaceFontSize=100&SecEventPlaceFontFamily=Calibri&EventIconVisibility=Center&EventIconRelativeLocalization=1&ItemCount=4&MaxEventsCount=40&ModelRefreshInterval=10](http://localhost:49331/ColumnView?Guid=AltOawklhMd6oTIW26-DumdFiQvRdYjnkR-_GcA&GoogleCalendarURL=https://www.google.com/calendar/feeds/l2c7gsp4rh0qesfr27hkjlq1c%40group.calendar.google.com/private75dc608a01553284f0ae53f828d60726/full&BackgroundColor=Grey&SecondaryBackgroundColor=Lightgrey&EventNameVisibility=Center&EventNameColor=Yellow&EventNameFontSize=100&EventNameFontFamily=Cambria&SecEventNameColor=Navy&SecEventNameFontSize=100&SecEventNameFontFamily=Cambria&EventStartDateVisibility=DateTimeLeft&EventStartDateColor=Orange&EventStartDateFontSize=100&EventStartDateFontFamily=Calibri&SecEventStartDateColor=Darkred&SecEventStartDateFontSize=100&SecEventStartDateFontFamily=Calibri&EventPlaceVisibility=Left&EventPlaceColor=LightGreen&EventPlaceFontSize=100&EventPlaceFontFamily=Calibri&SecEventPlaceColor=Darkgreen&SecEventPlaceFontSize=100&SecEventPlaceFontFamily=Calibri&EventIconVisibility=Center&EventIconRelativeLocalization=1&ItemCount=4&MaxEventsCount=40&ModelRefreshInterval=10)

Figura 4.53: Exemplo de query string de uma configuração para visualização Column

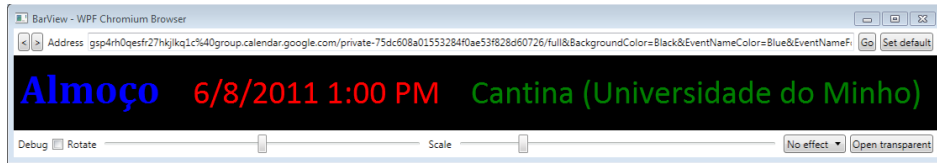


Figura 4.54: Exemplo de visualização Bar Fade processada pela DSEventApp

### Visualização Bar Fade

Considerando o exemplo da Figura 4.54, destaca-se um simples rodapé com informação básica sobre um evento (título, data de início e local). Como já foi dito, a visualização Bar Fade itera sobre cada evento, fazendo aparecer e desaparecer um evento de cada vez (*fade-in* e *fade-out*) e, por isso, tem um comportamento semelhante à visualização Full.

Comparando a *query string* da Figura 4.55 e a visualização da Figura 4.54, é fácil reparar que o parâmetro a negrito, `BackgroundColor`, define que a cor de fundo é preta, enquanto que os outros parâmetros a negrito determinam a visibilidade da data e do local. Não se considera nenhum parâmetro de visibilidade para o título, uma vez que este tem que ser obrigatoriamente visível. Também não se disponibiliza nenhum alinhamento customizável da informação do evento, pois num rodapé de pequenas dimensões não se justifica.

Os parâmetros que especificam as propriedades de fonte estão assinalados com a cor correspondente (azul para o título, vermelho para a data de início e verde para o local).

Por fim, dos parâmetros da componente comportamental, coloridos a azul claro, destacam-se os parâmetros `ItemDuration` e `FadeDuration`. Quanto ao primeiro parâmetro, pode-se dizer que determina o intervalo de tempo, em milissegundos, durante o qual cada evento é apresentado, sem nenhum tipo de transparência inerente ao *fading*. Enquanto que o segundo parâmetro define a duração, em milissegundos, do *fading*. Observando mais uma vez a Figura 4.55, tem-se `ItemDuration=2000` e `FadeDuration=1000`. Assim, para obter o tempo total de exposição de cada evento basta fazer o seguinte cálculo:

$$T_{total} = ItemDuration + FadeDuration * 2 \quad (4.1)$$

$$= 2000 + 1000 * 2 \quad (4.2)$$

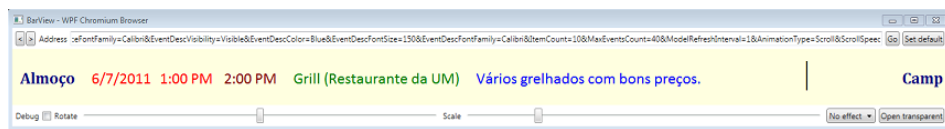
$$= 4000ms \quad (4.3)$$

### Visualização Bar Scroll

A visualização Bar Scroll corresponde a um rodapé com deslizamento de texto na horizontal, da direita para a esquerda. Tendo em conta a visuali-

```
http://localhost:49331/BarView?Guid=AltOawklhMd6oTIW26-DumdFiQvRdYjnkR-_GcA
&GoogleCalendarURL=https://www.google.com/calendar/feeds/12c7gsp4rh0qesfr27hkjkq1c%4
0group.calendar.google.com/private75dc608a01553284f0ae53f828d60726/full&BackgroundCo
lor=Black&EventNameColor=Blue&EventNameFontSize=300&EventNameFontFamily=Cambri
a&EventStartDateVisibility=DateTimeVisible&EventStartDateColor=Red&EventStartDateFont
Size=250&EventStartDateFontFamily=Calibri&EventPlaceVisibility=Visible&EventPlaceColor
=Green&EventPlaceFontSize=250&EventPlaceFontFamily=Calibri&ItemDuration=2000&ItemC
ount=10&MaxEventsCount=40&ModelRefreshInterval=1&AnimationType=Fade&FadeDur
ation=1000
```

**Figura 4.55:** Exemplo de query string de uma configuração para visualização Bar Fade



**Figura 4.56:** Exemplo de visualização Bar Scroll processada pela DSEventApp

zação Bar Scroll da Figura 4.56, verifica-se um período de transição entre dois eventos e também que, em relação à visualização Bar Fade, é possível apresentar mais informação sobre cada evento.

Considerando a *query string* da Figura 4.57, correspondente ao exemplo de visualização Bar Scroll da Figura 4.56, nota-se, sem dúvida nenhuma, uma maior quantidade de parâmetros em relação à visualização Bar Fade.

Para não se repetir praticamente o mesmo do que foi dito sobre os parâmetros da componente visual (nas anteriores análises), apenas se sublinha o facto do parâmetro a negrito, *EventEndDateVisibility*, ter o valor *TimeVisible*, enquanto que o parâmetro a negrito, *EventStartDateVisibility*, tem o valor *DateTimeVisible*. Assim, considerando a visualização Bar Scroll da Figura 4.56, vê-se claramente que é apresentada, a vermelho, a data e hora de início (*EventStartDateVisibility=DateTimeVisible*), enquanto que é apenas apresentada a hora de término de cada evento (*EventEndDateVisibility=TimeVisible*).

Quanto aos parâmetros, a azul claro, da componente comportamental, é de destacar o parâmetro *ScrollSpeed*, com a função de especificar a velocidade de deslizamento do texto em rodapé. Este parâmetro comporta valores superiores a 0, sendo que 1 especifica uma velocidade muito lenta, a partir de 2 a velocidade é moderada e, excedendo o valor 6, começa-se a verificar uma velocidade muito rápida.

## Visualização XML

Para terminar este capítulo, só falta analisar um exemplo de visualização XML e a sua respectiva *query string* com os parâmetros de configuração.

Em primeiro lugar, vai-se apenas mostrar o formulário para configurar

```

http://localhost:49331/BarView?Guid=AltOawklhMd6oTIW26-DumdFiQvRdYjnkR-_GcA
&GoogleCalendarURL=https://www.google.com/calendar/feeds/12c7gsp4rh0qesfr27hkjlq1c%4
0group.calendar.google.com/private75dc608a01553284f0ae53f828d60726/full&BackgroundCol
or=Lightyellow&EventNameColor=Navy&EventNameFontSize=150&EventNameFontFamily=Ca
mbria&EventStartDateVisibility=DateTimeVisible&EventStartDateColor=Red&EventStartDate
FontSize=150&EventStartDateFontFamily=Calibri&EventEndDateVisibility=TimeVisible&Eve
ntEndDateColor=Maroon&EventEndDateFontSize=150&EventEndDateFontFamily=Calibri&Eve
ntPlaceVisibility=Visible&EventPlaceColor=Green&EventPlaceFontSize=150&EventPlaceFon
tFamily=Calibri&EventDescVisibility=Visible&EventDescColor=Blue&EventDescFontSize=150
&EventDescFontFamily=Calibri&ItemCount=10&MaxEventsCount=40&ModelRefreshInterva
l=1&AnimationType=Scroll&ScrollSpeed=3

```

**Figura 4.57:** Exemplo de query string de uma configuração para visualização Bar Scroll

visualizações XML (Figura 4.58), uma vez que é o mais pequeno de todos e é fácil a análise dos parâmetros de configuração existentes (é a visualização que necessita de menos parâmetros).

É de reparar que o formulário contém alguns campos preenchidos e ao clicar no botão *Display*, obtém-se a visualização XML respectiva, como se pode observar na Figura 4.59. Utilizou-se o navegador Internet Explorer, uma vez que o WPF Chromium WebBrowser não suporta XML. De qualquer forma, não afecta em nada, visto que o serviço Ubisign.com não necessita do WPF Chromium WebBrowser para processar XML.

Analisando mais ao pormenor a *query string* da Figura 4.60, resultante do preenchimento do formulário, sublinham-se os parâmetros a cinzento (os parâmetros da componente de filtragem) e o parâmetro a azul claro, *MaxEventsCount*.

Quanto aos parâmetros da componente de filtragem, extrai-se a seguinte informação: o Google Calendar retorna todos os eventos que fazem pelo menos uma referência à palavra “Braga” e enquadram-se, no que diz respeito ao alcance temporal, entre o dia 1 de Junho e 1 de Julho de 2011 (observar valores, convertidos em segundos, dos parâmetros *FromDate* e *ToDate*).

Por fim, considerando o parâmetro *MaxEventsCount* com o valor 40, significa que é estabelecido o limite máximo de 40 eventos a retornar pelo serviço Google Calendar. No exemplo da Figura 4.59, não se chega a atingir esse limite e, como se pode ver na *tag* `<events>`, o atributo *numevents* diz-nos que foram retornados 21 eventos no total.

Figura 4.58: Formulário para configurar visualizações XML

```

<?xml version="1.0" encoding="UTF-8" ?>
<calendar calurl="https://www.google.com/calendar/feeds/l2c7gsp4rh0qesfr27hkjlq1c%40group.calendar.google.com/private-75dc608a01553284f0ae53f828d60726/full&FullTextMatching=Braga&FromDate=63442483200&ToDate=63445075200&MaxEventsCount=40" orderby="starttime" service="Google Calendar">
  <name>CalendarioTestes</name>
  <timezone>Europe/Lisbon</timezone>
  <events numevents="21">
    <event uid="3iqth643frqtnfqqr48la0lc8@google.com">
      <summary>Bilhar</summary>
      <dtstart>6/6/2011 8:00:00 AM</dtstart>
      <dtend>6/6/2011 9:30:00 AM</dtend>
      <location>Braga</location>
      <description>Preço: 50 Cêntimos</description>
      <image-url>http://downloads.open4group.com/wallpapers/bolas-de-bilhar-88a9e.jpg</image-url>
    </event>
    <event uid="dkn8mhd7d7vf0lh5fajcrvbg44@google.com">
      <summary>Natação</summary>
      <dtstart>6/7/2011 11:00:00 AM</dtstart>
      <dtend>6/7/2011 1:00:00 PM</dtend>
      <location>Piscinas da Rodovia - Braga</location>
      <description>Duas horas de grande intensidade física para evoluir as competências ligadas ao movimento de braços e pernas. Contamos consigo!</description>
      <image-url>http://1.bp.blogspot.com/-PQL9yIsOkMg/Tc_qo1Gf8n1/AAAAAAAAAAS4/LZMbyZ037CK/s400/natacao01.jpg</image-url>
    </event>
  </events>
</calendar>

```

Figura 4.59: Exemplo de visualização XML processada pela DSEventApp

[http://localhost:49331/XMLView?Guid=AltOawklhMd6oTIW26-DumdFiQvRdYjnkR-\\_GcA&GoogleCalendarURL=https://www.google.com/calendar/feeds/l2c7gsp4rh0qesfr27hkjlq1c%40group.calendar.google.com/private-75dc608a01553284f0ae53f828d60726/full&FullTextMatching=Braga&FromDate=63442483200&ToDate=63445075200&MaxEventsCount=40](http://localhost:49331/XMLView?Guid=AltOawklhMd6oTIW26-DumdFiQvRdYjnkR-_GcA&GoogleCalendarURL=https://www.google.com/calendar/feeds/l2c7gsp4rh0qesfr27hkjlq1c%40group.calendar.google.com/private-75dc608a01553284f0ae53f828d60726/full&FullTextMatching=Braga&FromDate=63442483200&ToDate=63445075200&MaxEventsCount=40)

Figura 4.60: Exemplo de query string de uma configuração para visualização XML

## Capítulo 5

# Aplicação da DSEventApp a Casos de Estudo

Ao longo deste capítulo serão analisadas algumas possibilidades de utilização da DSEventApp num contexto hipotético mas que pode ser transposto para a realidade, ou seja, vão ser dados três casos de utilização das visualizações (Full, Column, Bar e XML) por parte de aplicações externas, em painéis de *digital signage*, com o intuito de apresentar informação sobre eventos. No fim, será apresentado o resultado obtido após a instalação da DSEventApp no serviço Ubisign.com, com base em exemplos para as visualizações Full, Column e Bar.

É de salientar que cada uma das visualizações tem um tipo de utilização distinta. Considerando as visualizações Column e Bar, é fácil perceber que a primeira normalmente serve para se enquadrar num espaço com altura máxima mas pouca largura (em forma de coluna), enquanto que a visualização Bar costuma ser utilizada num espaço com pouquíssima altura mas, geralmente, com largura máxima. Como já foi dito, a visualização Full ocupa todo o ecrã (ou quase todo, caso haja uma coluna e/ou um rodapé) de um painel de *digital signage*.

De uma forma geral, as aplicações externas que interagem com a DSEventApp vão utilizar um ou mais tipos de visualização em simultâneo e, se for necessário, até poderá ser dado um toque artístico às visualizações, para que estas fiquem mais atraentes e captem, assim, o público alvo.

Por fim, a visualização XML é a mais abrangente de todas, uma vez que permite o consumo de eventos por parte de qualquer tipo de aplicação e essas aplicações podem ser destinadas não só a painéis de *digital signage*, mas também a outros dispositivos tecnológicos frequentemente utilizados, como por exemplo: televisão, computador, *tablets*, *smart phones*, etc.. As visualizações Full, Column e Bar também poderão ser utilizadas noutra tipo de tecnologias que não os painéis, só que com menos frequência.

## 5.1 Introdução

Como já foi dito, neste capítulo vão ser dados três casos hipotéticos de uso da DSEventApp, com possibilidades de utilização real, baseados em aplicações externas que tiram partido das visualizações, com o fim de apresentar, em painéis de *digital signage*, informação sobre eventos ao público alvo. Na última secção deste capítulo, será apresentado o resultado da integração da DSEventApp no serviço Ubisign.com, sendo dado um exemplo para cada visualização (Full, Column e Bar).

Em primeiro lugar, vai ser considerado um caso de apresentação de informação sobre eventos desportivos que vão ocorrer num hotel, com o objectivo de informar as pessoas interessadas de como podem participar nesses eventos. Também vai ser fornecida informação sobre a hora de início de cada evento desportivo e o local do hotel onde este é realizado.

A DSEventApp vai ter um papel muito importante, uma vez que vão ser utilizados, por intermédio de uma aplicação externa, dois tipos de visualização, a Full e a Column. A visualização Full irá servir para apresentar eventos desportivos ciclicamente, com o máximo de informação possível que cada evento pode conter, incluindo sempre uma imagem que ilustre claramente o desporto considerado. A visualização Column vai ter também um papel preponderante no que diz respeito à apresentação simultânea de todos os eventos que vão ocorrer durante o dia (com informação mais resumida em relação à visualização Full).

De seguida, vão ser considerados dois exemplos de mapas de voos num aeroporto, apresentados por painéis de *digital signage*. No primeiro exemplo, considera-se que existe uma aplicação externa que utiliza seis visualizações Bar Fade para que o conteúdo de cada evento (ou voos) seja orientado à linha. Quanto ao segundo exemplo, destacam-se três visualizações Column e a informação de cada voo é disposta numa das células, formando uma matriz. Em ambos os exemplos, consta um rodapé onde irá ficar alocada uma visualização Bar Scroll, com o fim de apresentar informação de cariz urgente, como por exemplo, o cancelamento ou o adiamento de voos.

Para acabar com a análise de todos os tipos de visualização, considera-se um caso de uso da visualização XML, onde vão ser analisados incêndios que estão, em determinado período, a ocorrer na região do Minho e a informação sobre o progresso dos trabalhos de extinção alcançados até ao momento, pelas corporações de bombeiros<sup>1</sup>. Para além disso, também é apresentada informação sobre a data de início de cada incêndio, número de corporações, veículos, local, etc..

Neste caso particular, vai ser considerado um painel de *digital signage*, sensível ao toque, para cada quartel de bombeiros, com o objectivo destes

---

<sup>1</sup>Esta ideia surgiu a partir de um trabalho elaborado em 2009, por alunos do terceiro ano da Licenciatura em Engenharia Informática, no âmbito da Unidade Curricular de Laboratórios de Informática 4 (<http://fogum.googlecode.com>).

poderem facilmente aceder a informação sobre os fogos que estão a ocorrer de momento, com a ajuda de um mapa de Portugal, disponibilizado pelo Google Maps.

O capítulo termina com uma secção que descreve o serviço Ubisign.com, assim como a apresentação de exemplos de utilização da DSEventApp, como resultado da integração desta no serviço Ubisign.com. Por isso, vão ser apresentados três exemplos dedicados às visualizações Full, Column e Bar, juntamente com outros conteúdos proporcionados pelo serviço, para que se prove a utilização real da DSEventApp no contexto da empresa Ubisign, responsável pelo projecto.

## 5.2 Utilização da DSEventApp num Hotel

Em hotéis de grandes dimensões e de qualidade, é frequente haver um escalonamento das actividades lúdicas, com o objectivo de proporcionar uma melhor estadia aos clientes do hotel.

Para que as pessoas sejam informadas dos eventos que vão ocorrer, é indispensável que se adopte um mecanismo eficaz. Como tal, este é um caso em que o uso de painéis de *digital signage* se torna bastante atractivo, uma vez que é uma tecnologia adequada para locais públicos. Além disso, as pessoas não costumam ficar indiferentes quando passam por *displays* de média ou grande dimensão.

Como já foi dito no capítulo 2, Estado da Arte, os painéis digitais permitem a mudança de conteúdos apresentados de uma forma rápida e fácil, pois basta haver um servidor central para que seja feito um controlo total sobre todos os *displays* existentes no hotel. Caso fosse utilizado um mecanismo informativo à base do papel (*paper signage*), teria que se efectuar várias mudanças diárias, não sendo nada eficiente e exigindo mais meios, como o consumo diário de muito papel, tinta, tempo e de recursos humanos.

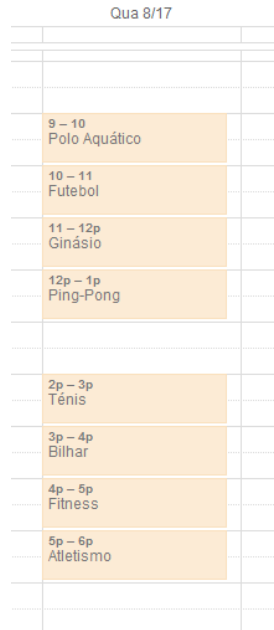
As ferramentas *on-line* de gestão de eventos, como o Google Calendar, ajudam muito no escalonamento dos eventos diários e permitem que este seja feito de um modo simples mas eficaz. Além disso, os eventos acrescentados são quase instantaneamente acedidos por todos os *displays* (típico das tecnologias ubíquas).

Passando para o caso concreto, é de sublinhar que vão ser considerados eventos desportivos que ocorrerão numa quarta-feira, dia 17 de Agosto, num hotel de consideráveis dimensões. Pode-se observar na Figura 5.1 oito actividades desportivas que vão ocorrer nesse dia, com a duração de uma hora cada.

Considerando a Figura 5.2, observa-se o resultado do acesso e processamento dos eventos especificados no Google Calendar (Figura 5.1), por parte da DSEventApp.

A DSEventApp vai proporcionar dois tipos de visualizações muito úteis





Qua 8/17	
9 – 10	Polo Aquático
10 – 11	Futebol
11 – 12p	Ginásio
12p – 1p	Ping-Pong
2p – 3p	Tênis
3p – 4p	Bilhar
4p – 5p	Fitness
5p – 6p	Atletismo

**Figura 5.1:** Exemplo de oito eventos desportivos especificados no Google Calendar

para apresentar os eventos desportivos, as Full e Column. Para se tirar partido dessas duas visualizações ao mesmo tempo, é necessário utilizar uma aplicação externa que divida um *display* em duas partes.

Como se pode observar na Figura 5.2, a parte do ecrã alocada para a visualização Full é consideravelmente maior do que a alocada para a visualização Column. No lado esquerdo do ecrã, destaca-se o evento desportivo, polo aquático, apresentado no modo de visualização Full. Do lado direito é apresentada, no modo de visualização Column, uma visão global dos oito eventos especificados.

É de salientar que, na visualização Full, os eventos vão ser organizados pela ordem definida na coluna do lado direito. Esta visualização vai efectuar iterações a partir do evento designado por *Polo Aquático* até ao evento com o título *Atletismo*, sendo que quando se chega ao último evento, retorna-se ao primeiro (*Polo Aquático*), continuando o processo de forma cíclica.

Devido às maiores dimensões reservadas para a visualização Full, pode-se ver que cada evento tem um título, uma data/hora de início e término, um local onde este irá ocorrer, uma descrição e uma imagem ilustrativa sobre o desporto considerado.

Na visualização Full do exemplo da Figura 5.2, extrai-se a seguinte informação:

- O título do evento é o nome do desporto planeado, que neste caso é polo aquático;

The screenshot displays a mobile application interface. On the left, a large orange panel contains the event title 'Polo Aquático' in blue. Below it, the date and time '8/17/2011 9:00 AM 10:00 AM' are shown in pink. The location 'Piscina Exterior' is listed in blue. A paragraph of text in green provides details about the event, including registration information and team formation rules. A photograph of a water polo player is positioned to the right of the text. On the right side of the screen, a vertical list of activities is shown in white text on a black background, including Polo Aquático (9:00 AM), Futebol (10:00 AM), Ginásio (11:00 AM), Ping-Pong (12:00 PM), Tênis (2:00 PM), Bilhar (3:00 PM), Fitness (4:00 PM), and Atletismo (5:00 PM).

**Figura 5.2:** Exemplo de eventos apresentados ciclicamente num display

- A data/hora de início do evento é dia 17 de Agosto às 9 horas da manhã;
- A hora de término do evento é às 10 horas da manhã;
- O local onde ocorre o jogo é na piscina exterior do hotel;
- A descrição esclarece os detalhes do jogo e explica onde pode ser feita a inscrição;
- A imagem, por si só, diz qual é o evento desportivo e procura ser atractiva. Neste exemplo o seu alinhamento é feito à direita, fazendo com que os restantes elementos informativos do evento fiquem alinhados à esquerda (excepto o título);

Para terminar a análise da Figura 5.2, a visualização Column apenas apresenta o título e a data de início de cada evento, sendo suficiente para saber quais os eventos desportivos que vão ocorrer durante o dia e qual a sua ordem temporal. Como se trata de uma visualização em coluna, é natural que se utilizem cores de fundo que se contrastem para eventos consecutivos (neste caso alterna-se entre a cor de fundo branca e preta).

Caso se pretenda modificar o resultado das visualizações fornecidas pela DSEventApp, pode-se, com recurso a uma aplicação externa, processar e alterar o código HTML devolvido pela DSEventApp. Na Figura 5.3 é possível visualizar algumas alterações feitas no exemplo apresentado anteriormente (Figura 5.2).



**Figura 5.3:** Exemplo de eventos apresentados ciclicamente num display e com alterações gráficas por parte de uma aplicação externa

Começando pela visualização Full, apesar de não ser muito relevante, foi acrescentado um traço entre a hora de início e de fim, a verde adicionou-se o campo *Local* e a cinzento acrescentou-se uma borda (onde tem escrito *Descrição*) para delimitar a descrição de cada evento.

Quanto à visualização Column, nota-se que o evento, com o título *Polo Aquático*, se encontra destacado com uma cor de fundo azulada, pretendendo-se com isto dizer que esse evento é o apresentado actualmente na visualização Full. Sempre que a visualização Full passa para a iteração seguinte, o evento seleccionado na coluna também avança, ajudando o utilizador a ter uma noção mais exacta do que está a ser apresentado no painel de *digital signage*.

Para terminar esta secção, é importante referir que se poderiam realizar numerosas alterações ao exemplo da Figura 5.2, consoante as preferências dos *designers* ou com base em inquéritos feitos a clientes do hotel.

### 5.3 Utilização da DSEventApp num Aeroporto

Os aeroportos são geralmente sítios por onde passam diariamente milhares de pessoas. Com vista a dar informações sobre voos que vão ocorrer, é necessário utilizar um mecanismo eficiente e com actualizações frequentes. A forma mais comum de o fazer, é recorrendo a painéis digitais com o mapa de voos.

Para aeroportos de grandes dimensões é costume haver várias divisões com voos distintos e, como consequência, também costumam existir vários painéis digitais com mapas de voos diferentes para cada divisão.

LH	4012	Genf	13:00
LH	4576	Brussel	13:00
LH	418	Washington	13:00
LH	1172	Paderborn Lippstadt	13:00
LH	648	Almaty	13:00
LH	470	Toronto	13:00
LH	180	Berlin-Tegel	13:00
LH	4376	Istanbul	13:00

**Figura 5.4:** *Mapa de voos*

Tipicamente um mapa de voos tem obrigatoriamente um código de voo, a hora de partida e o local de destino (como se pode verificar no exemplo da Figura 5.4). Adicionalmente também pode ser apresentado o número da porta, o estado do voo e uma observação caso necessário (por exemplo, voo cancelado devido às más condições climatéricas).

Para apresentar um mapa de voos, pode ser utilizada a DSEventApp, de modo a que seja possível especificar voos no Google Calendar e disponibilizar essa informação actualizada em painéis de *digital signage*.

Tal como no exemplo do hotel, neste caso também se vai assumir que existe uma aplicação externa que divide em partes o ecrã do *display*, para enquadrar as visualizações processadas pela DSEventApp.

Em primeiro lugar, vai ser considerado um exemplo de mapa de voos com seis visualizações Bar Fade, com o objectivo de apresentar, linha à linha, o conteúdo informativo de cada voo (exemplo da Figura 5.5). De seguida, apresenta-se um mapa de voos alternativo e mais raro, com três visualizações Column (exemplo da Figura 5.6). É de reparar que ambos os exemplos têm, em rodapé, informação de cariz urgente (por exemplo, cancelamento de voos) no modo de visualização Bar Scroll.

Tendo em conta o primeiro exemplo (Figura 5.5), pode-se observar que o ecrã tem um *layout* dividido em oito linhas. Partindo de cima, a primeira linha tem uma cor de fundo negra com os nomes dos campos (código do voo, hora de partida e destino), sendo apenas produzida pela aplicação externa. As restantes sete linhas contêm as visualizações produzidas pela DSEventApp, em que as primeiras seis são visualizações Bar Fade e a última é uma visualização Bar Scroll.

É de reparar que as seis visualizações Bar Fade têm como locais de destino países diferentes. Isso explica-se devido ao facto de ser feita uma filtragem, por país, para cada uma das visualizações. Assim, as iterações são feitas de forma a percorrer todos os voos de cada país. Neste caso, para simplificar a Figura 5.5, apenas foram considerados seis países (Portugal, Espanha, Inglaterra, França, Alemanha e Itália). No entanto, os painéis digitais têm, normalmente, grandes dimensões para apresentar o maior número de voos possível.

Quanto às cores utilizadas em cada visualização Bar Fade, destaca-se a evidente diferença de tonalidades de azul/violeta na cor de fundo, com a

<i>Flight</i>	<i>Time</i>	<i>Destination</i>
TP 1979	17:50	Lisbon (PT)
TP 736	20:05	Barcelona (SP)
TP 332	19:30	London (UK)
TP 458	19:45	Paris (FR)
LH 1179	17:00	Frankfurt (GR)
TP 852	21:50	Rome (IT)

**WARNING!** Flight TP 456 has been cancelled due to technical problems

**Figura 5.5:** Exemplo 1 de mapa de voos processado pela DSEventApp

finalidade de criar contraste entre linhas contíguas. A cor de fonte é bastante clara, uma vez que a cor de fundo é escura, criando um grande contraste. É também de realçar que o resultado da combinação das seis visualizações Bar Fade, com duas cores de fundo em linhas adjacentes, assemelham-se ao modo de visualização Column.

Finalizando a análise das seis visualizações Bar Fade da Figura 5.5, destacam-se uns separadores coloridos a vermelho e laranja entre cada campo informativo do voo, que são introduzidos pela aplicação externa, com o objectivo de dar um toque artístico. É de realçar que a DSEventApp não foi implementada, por defeito, com elementos decorativos (bordas, separadores, delimitadores, etc.), uma vez que se pretende dar uma liberdade total ao *designer*, no que diz respeito à customização das visualizações produzidas pela DSEventApp.

Considerando o rodapé da Figura 5.5, é importante dizer que tem deslização horizontal de texto da direita para a esquerda, isto é, trata-se da visualização Bar Scroll processada pela DSEventApp. Como já foi dito, a informação que passa nesse rodapé é de cariz urgente (como por exemplo, cancelamento de voos), não impedindo que seja utilizado outro tipo de informação (como por exemplo, a publicidade).

Outro aspecto que se deve sublinhar, é o facto de serem apresentados pseudo-eventos no rodapé da Figura 5.5, ou seja, são apresentados avisos de que algo aconteceu, não sendo nada programado, pois trata-se de imprevistos. Nesta situação, utilizar o Google Calendar não é comum, uma vez que os avisos não têm delimitação temporal (data de início e fim). No entanto,

Portugal	Spain	United Kingdom
<b>TP 1979</b> 5:50 PM Lisbon (PT)	<b>TP 736</b> 8:05 PM Barcelona (SP)	<b>TP 332</b> 7:30 PM London (UK)
<b>TP 1981</b> 8:15 PM Lisbon (PT)	<b>IB 8725</b> 8:20 PM Madrid (SP)	<b>EZY 5314</b> 8:10 PM London (UK)
<b>FR 5486</b> 9:45 PM Faro (PT)	<b>AB 7785</b> 10:25 PM Palma Mallorca (SP)	<b>S4 9740</b> 9:30 PM Birmingham (UK)
<b>WARNING!</b> Flight TP 456 has been cancelled due to technical problems		

**Figura 5.6:** Exemplo 2 de mapa de voos processado pela DSEventApp

nada impede que seja utilizada uma ferramenta *on-line* de gestão de eventos para introduzir informação de última hora.

Passando para o exemplo do mapa de voos da Figura 5.6, destaca-se, à primeira vista, uma organização diferente da informação (organização em matriz) em relação ao exemplo anterior (organização orientada à linha) e ao que costuma ser o *layout* frequentemente utilizado para os mapas de voos. Neste caso, há uma divisão do ecrã em três colunas, sendo que cada uma é destinada ao processamento de uma visualização Column. Também é de reparar que cada coluna apresenta informação sobre voos com países de destino diferentes, em que a primeira coluna apresenta voos para Portugal, a segunda para Espanha e a terceira para Inglaterra. Comparando este exemplo com o anterior, esta visualização apenas apresenta voos de três países (um país por coluna), enquanto que o primeiro disponibiliza voos de seis (um país por linha).

Esta forma de apresentar um mapa de voos tem a vantagem de permitir otimizar o espaço ocupado no ecrã, não perdendo a boa visibilidade, pois comparando com as seis visualizações Bar Fade do exemplo da Figura 5.5, as três visualizações Column apresentam, no total, mais três eventos (ou voos). No entanto, o facto de se utilizar visualizações Column implica que a aplicação externa efectue pedidos, de X em X tempo, de novas visualizações, uma vez que esse tipo de visualização é completamente estática e, por isso, não há iteração temporal sobre os eventos.

Para terminar a análise da Figura 5.5, resta referir que quanto às cores, é utilizada a mesma estratégia do exemplo do mapa de voos anterior, com

alternância de cores de fundo para eventos adjacentes e as cores utilizadas, tanto de fundo como de fonte, são iguais.

## 5.4 Utilização da DSEventApp em quartéis de bombeiros

Em países com climas relativamente quentes, como Portugal, é frequente existirem incêndios florestais. As causas desses incêndios podem ser várias, mas geralmente ou são por descuido humano, deliberados ou simplesmente por obra da natureza.

O papel desempenhado pelos bombeiros é muito importante, uma vez que combatem os incêndios das florestas do planeta, salvando animais e vidas humanas.

O trabalho efectuado pelas corporações de bombeiros é bastante difícil, principalmente quando se atravessam épocas de fortes incêndios e em grandes quantidades (normalmente no verão). É de extrema importância que haja meios eficazes para minimizar a perda de tempo em situações de máxima urgência, como por exemplo: haver boas condições de trabalho, materiais necessários e de qualidade, meios informativos eficientes, etc.. Dado que o tema desta dissertação incide na área da informação sobre eventos em painéis de *digital signage*, vai-se dar destaque à disponibilização de meios informativos eficientes e em tempo real em *displays* localizados em quartéis de bombeiros, de modo a apresentar os recursos humanos e materiais envolvidos no combate desses fogos, para além de outras informações, como por exemplo, o estado dos incêndios.

Nesta secção vai-se dar destaque a uma aplicação externa que fornece, com base na visualização XML, informação sobre o estado dos incêndios em Portugal, sendo que a melhor forma de o fazer é recorrendo a um mapa onde estão marcados os incêndios. Também é relevante dizer que, para a aplicação externa permitir interacção com o utilizador, os *displays* têm de permitir sensibilidade ao toque, apesar de se poder utilizar outro tipo de tecnologia, como por exemplo: computadores, *tablets*, ou até mesmo *smartphones*.

Analisando a Figura 5.7, podem-se destacar alguns incêndios assinalados no mapa correspondente à região do Minho. Através da sua legenda observam-se três estados que cada incêndio pode apresentar ao longo do seu ciclo de vida.

O ícone do fogo com cor de fundo vermelho corresponde a um incêndio que ainda não foi controlado pelos bombeiros, isto é, o incêndio ainda se encontra numa fase em que há potencial perigo. Passando para o ícone de fundo laranja, trata-se de um incêndio já controlado pelos bombeiros e, assim, chega-se a uma fase em que é necessário menor número de bombeiros, pois o incêndio já oferece pouco perigo. Por fim, o ícone de fundo violeta traduz-se num incêndio extinto, mas que ainda apresenta alguma probabilidade de



Figura 5.7: Incêndios na região do Minho

voltar a eclodir quando a temperatura exterior é bastante elevada (fenómeno comum em zonas de grande densidade florestal).

Passando para a análise da Figura 5.8, destaca-se um hipotético incêndio que ocorreu em Terras de Bouro no dia 1 de Agosto de 2011 e a respectiva informação ao longo do seu ciclo de vida. Consideraram-se três estados para este incêndio: não controlado, circunscrito e rescaldo. Assim, na Figura 5.8 apresentam-se três divisões, sendo que cada uma corresponde a um estado distinto do mesmo incêndio.

A DSEventApp tem como função, obter eventos (neste caso, fogos) do Google Calendar e produzir uma visualização XML. A aplicação externa processa o XML, de modo a que os fogos sejam apresentados correctamente no mapa e os balões também são preenchidos com a respectiva informação.

Considerando o balão da primeira divisão, pode-se analisar o incêndio no seu estado não controlado. Como a informação sobre o mesmo é obtida através do Google Calendar, pode-se observar o conteúdo do balão da seguinte maneira:

- O título do evento contém o local do incêndio e, de seguida, entre parêntesis contém o estado – o incêndio da primeira divisão tem o título *Terras de Bouro (Não Controlado)*. Uma vez que tanto o local, como o estado do incêndio já são apresentados no balão, não há a obrigatoriedade de apresentar o título;
- O local onde ocorre o incêndio é, como já foi dito, em Terras de Bouro;



- A data/hora de início do incêndio é dia 1 de Agosto às 11 horas e meia da manhã;
- A hora de término do incêndio ainda não é considerada, uma vez que o incêndio ainda não foi extinto;
- A descrição é tudo o que aparece no campo *Informação* (texto a azul e sublinhado), onde constam os seguintes dados: o estado do incêndio (não controlado), a característica da zona (neste caso trata-se de uma zona muito arborizada e propícia a incêndios), a superfície ardida por quilómetro quadrado ( $75Km^2$ ), o número de corporações (4), de bombeiros (35), de viaturas (8) e de helicópteros (2). Poder-se-ia mostrar mais informação, no entanto não se considera para simplificar o problema;
- O ícone tem o único objectivo de realçar o estado do incêndio, que neste caso se encontra num estado não controlado e, por isso, é apresentado um ícone chamadas de grande intensidade.

Comparando o balão do incêndio no estado circunscrito com o do estado não controlado, apenas se destaca que o primeiro tem um campo idêntico à data de fim, mas que neste caso corresponde obviamente à data de circunscrição (campo *Circ.*), enquanto que o outro, como foi dito anteriormente, não tem nenhuma data. As restantes diferenças incidem apenas nos valores atribuídos aos campos da descrição (superfície ardida, número de corporações, etc.) e no ícone utilizado.

Quanto ao último balão, correspondente ao incêndio na fase de rescaldo, também não há diferenças significativas quanto aos campos informativos em relação aos outros, exceptuando o facto deste ter uma real data de término em relação ao incêndio circunscrito (no entanto a DSEventApp interpreta os campos de datas de circunscrição e de extinção como se fossem ambas datas de término).

Uma vez terminada a análise da Figura 5.8, que representa o resultado do *parsing* de visualizações XML por parte de uma aplicação externa, vai-se analisar, de forma simples, o XML processado.

Considerando a divisão 1 da Figura 5.9, destaca-se a informação, no formato XML, sobre o incêndio descrito anteriormente, num estado ainda não controlado. Apesar de não haver data de término para um incêndio não controlado, como o Google Calendar obriga a definir esse campo, especifica-se essa data ao critério de cada um. Neste caso, adoptou-se o tempo de duração de uma hora, que é a duração por defeito de qualquer evento criado no Google Calendar. Também é de destacar que a descrição tem os vários campos (estado, zona, superfície ardida, etc.) separados por um ponto e vírgula e o nome do campo separado por dois pontos em relação ao valor correspondente. Torna-se, assim, mais simples efectuar o *parsing* dos campos e respectivos



Figura 5.8: Três exemplos do mesmo incêndio, reflectindo o seu ciclo de vida

```

<event uid="9jglbc982ok7jf4t3sm4k7fhkk@google.com">
  <summary>Terras de Bouro (Não Controlado)</summary>
  <dtstart>1/8/2011 11:30:00</dtstart>
  <dtend>1/8/2011 12:30:00</dtend>
  <location>Terras de Bouro</location>
  <description>Estado: Não Controlado; Zona: muito arborizada; Superfície ardida: 75Km2; Número de corporações: 4; Número de bombeiros: 35; Número de viaturas: 8; Número de helicópteros: 2; Latitude: 41.71557586546657; Longitude: -8.297456502914429</description>
  <image-url>http://www.ads-news.com/wp-content/uploads/2010/09/Incendio.jpg</image-url>
</event>
1

<event uid="9jglbc982ok7jf4t3sm4k7fhkk@google.com">
  <summary>Terras de Bouro (Circunscrito)</summary>
  <dtstart>1/8/2011 11:30:00</dtstart>
  <dtend>6/8/2011 10:00:00</dtend>
  <location>Terras de Bouro</location>
  <description>Estado: Circunscrito; Zona: muito arborizada; Superfície ardida: 120Km2; Número de corporações: 3; Número de bombeiros: 24; Número de viaturas: 5; Número de helicópteros: 2; Latitude: 41.71557586546657; Longitude: -8.297456502914429</description>
  <image-url>http://geoplantas.blogs.sapo.pt/arquivo/Fogo Controlado.jpg</image-url>
</event>
2

<event uid="9jglbc982ok7jf4t3sm4k7fhkk@google.com">
  <summary>Terras de Bouro (Rescaldo)</summary>
  <dtstart>1/8/2011 11:30:00</dtstart>
  <dtend>7/8/2011 11:50:00</dtend>
  <location>Terras de Bouro</location>
  <description>Estado: Rescaldo; Zona: muito arborizada; Superfície ardida: 122Km2; Número de corporações: 1; Número de bombeiros: 4; Número de viaturas: 1; Número de helicópteros: 0; Latitude: 41.71557586546657; Longitude: -8.297456502914429</description>
  <image-url>http://3.bp.blogspot.com/_y_Ei0lDsqNs/THufbHTxSPI/AAAAAAAC5g/2Rp4-DOhQwc/s1600/incendio.jpg</image-url>
</event>
3

```

Figura 5.9: Três instâncias de XML geradas pela DSEventApp para o mesmo incêndio em estados diferentes

valores, por parte da aplicação externa, com o objectivo, por exemplo, de realçar os campos a negrito (ver balões informativos da Figura 5.8). Além disso, existem dois campos que são ocultados nos balões informativos, que são as coordenadas onde ocorreu o incêndio. Estas são aproveitadas pela aplicação externa para introduzir o ícone do incêndio no mapa.

Quanto à divisão 2 da Figura 5.9, apenas há a destacar que o campo da data de término corresponde à data de circunscrição (6 de Agosto de 2011) e na divisão 3 o campo da data de término corresponde à data de extinção do incêndio (7 de Agosto de 2011). O URL do ícone, utilizado dentro do balão informativo de cada incêndio, está especificado na *tag* <image-url>.

Para finalizar esta secção, há que referir que existe a possibilidade de efectuar *queries* de filtragem ao serviço Google Calendar através da visualização XML e, considerando a aplicação externa sobre incêndios, é muito simples fazer uma filtragem por estado do incêndio.



**Figura 5.10:** Exemplos de filtragem de incêndios

A Figura 5.10 exemplifica o resultado da filtragem de incêndios pelo seu estado. Na divisão 1 observa-se os vários focos de incêndio existentes numa zona do Minho, a divisão 2 é o resultado da filtragem por incêndios no estado não controlado, na divisão 3 constam os incêndios no estado circunscrito e, por fim, na divisão 4 são apresentados os incêndios na fase de rescaldo.

## 5.5 Integração da DSEventApp no serviço Ubisign.com

Antes de serem apresentados exemplos sobre a integração da DSEventApp no serviço Ubisign.com, vai ser efectuado um resumo sobre o que consiste o serviço e respectivas funcionalidades.

O Ubisign.com<sup>2</sup> é um serviço *web* (SaaS – *Software as a Service*) que tem a função de monitorizar redes de *digital signage* e de gerir canais de comunicação multimédia. Assim sendo, é possível definir o conteúdo a ser apresentado, onde e quando é apresentado, e permitir escolher o público alvo de uma forma precisa e eficiente<sup>3</sup>.

Quanto à arquitectura utilizada pela Ubisign, é importante referir que existem duas componentes de base: o *backoffice* de gestão e o *player*. Para aceder ao *backoffice* de gestão, basta um simples navegador *web* para gerir todas as funcionalidades associadas à operação da rede de *displays*, mais concretamente, a definição visual e temporal de grelhas de programação dos canais que se pretende visualizar. Quanto ao *player*, pode-se dizer que é a componente responsável por apresentar os conteúdos nos respectivos *displays*, recorrendo a um acesso à Internet de banda larga para comunicar com o *backoffice* e, assim, poder ter acesso às grelhas de programação e respectivos conteúdos<sup>4</sup>.

<sup>2</sup><http://www.ubisign.com>

<sup>3</sup><http://www.ubisign.com/public/PortalRender.aspx?PageID=af2b112b-c4d9-436b-93dc-c1957b513eda>

<sup>4</sup><https://skydrive.live.com/view.aspx/Pública/ubisigncom-visaogeral2011.ppsx?cid=de2a96b005a494c3&sc=documents>

Para que o utilizador possa aceder ao serviço Ubisign.com, é necessário que efectue a sua autenticação por *login* e *password*, através do endereço <http://saas.ubisign.com>. A *homepage* permite ao utilizador aceder, de forma mais rápida, a um resumo das actividades recentes da sua conta (*domain*), realizar tarefas de gestão de permissões dos utilizadores do *domain* e, também, ser informado sobre os problemas que possam surgir na rede de *players*.

Quanto às áreas funcionais do serviço Ubisign.com, pode-se dizer que existem três áreas distintas: a de *networks*, de *assets* e de *monitor*. A área de *networks* permite que seja efectuada a segmentação lógica da rede de *players*, por forma a facilitar a sua gestão, para além de permitir segmentar um *domain* em várias *networks* e, dentro de cada uma delas, ter diferentes canais de conteúdos (*channels*), bem como diferentes responsáveis pela sua gestão. Os *assets* são conteúdos ou referências para conteúdos utilizados num canal e a área de *monitor* destina-se a funções de monitorização, permitindo saber o estado de ligação dos *players* da rede e os tempos de funcionamento e apresentação dos conteúdos.

De seguida, são apresentadas algumas das funcionalidades que o serviço Ubisign.com disponibiliza:

- Editar e criar novas redes de *displays*;
- Dividir um canal em blocos e atribuir momentos diferentes de agenda ou ter um único bloco que se mantém em exibição cíclica e permanente;
- Criar *layouts*, definir as regiões que compõem um *layout* e indicar os conteúdos a ser apresentados em cada região (os conteúdos podem ter o formato *flash*, vídeo, imagem, RSS, texto, *website*, *slideshow*, etc.);
- O serviço Ubisign.com monitoriza, em tempo real, o estado de cada *player*, possibilitando que o utilizador, remotamente, esteja sempre a par do que se passa com a sua rede de *players*;
- Cada *player* envia *logs* para o servidor, permitindo que o utilizador tenha acesso a eles e detecte possíveis anomalias;
- etc..

Passando para o contexto do uso de APIs no desenvolvimento de aplicações para o serviço Ubisign.com, é importante referir que facilita o desenvolvimento de aplicações *web* e, ao mesmo tempo, enriquece o serviço, no sentido em que aumenta o leque de opções na escolha dos conteúdos para as regiões de um *layout*. Pode-se dizer que do ponto de vista do serviço Ubisign.com, estas aplicações são mais um tipo de injeção de conteúdos, neste caso, aplicações *web*.

Tendo sido feito um resumo sobre o que consiste o serviço Ubisign.com, é altura de apresentar três exemplos de utilização da DSEventApp no serviço

Ubisign.com, sendo que cada exemplo representa um tipo de visualização distinta, isto é, as visualizações Full, Column e Bar. Também é de referir que esses três exemplos foram pensados para exibir eventos num *display* dentro da sala onde se encontra a trabalhar a equipa da Ubisign, de modo a que esta tenha um maior acompanhamento dos eventos que vão ocorrendo no dia a dia.

Para começar, na Figura 5.11 é possível ver o *layout* utilizado para os primeiros dois exemplos (visualizações Full e Column), sendo que se efectuou uma divisão do ecrã do *display* em três regiões distintas, com o objectivo de atribuir diferentes conteúdos para cada região. O mesmo se poderá dizer sobre o *layout* da Figura 5.16, com a excepção de se considerar mais uma região para colocar informação em rodapé (criado para o último exemplo – visualização Bar). Também é de referir que todos os exemplos partilham um *background* negro, com uma marca de moda (*Eureka*) evidenciada no canto superior esquerdo do *layout*, dando um pouco mais de estilo ao *background*.

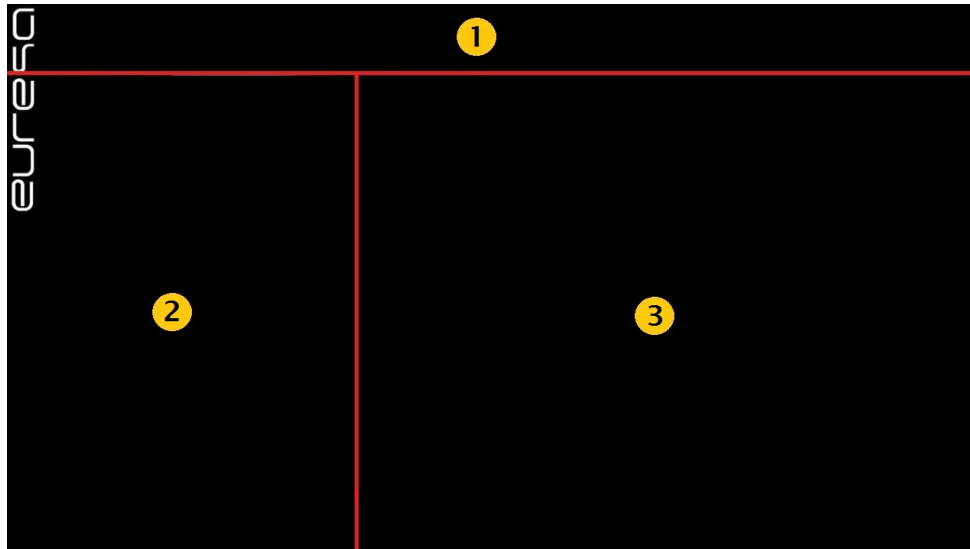
Considerando o exemplo da Figura 5.12, apenas se destacam conteúdos nas regiões 2 e 3 do *layout* (conferir regiões na Figura 5.11), sendo que na região 2 se encontra uma fotografia de uma modelo e na região 3 destaca-se uma visualização Full que apresenta, ciclicamente, eventos planeados pela Ubisign. É de salientar que a visualização Full é o único conteúdo processado pela DSEventApp (na Figura 5.13 destaca-se essa visualização) e a imagem da modelo é um conteúdo já existente no serviço Ubisign.com.

Quanto à configuração da visualização Full, resolveu-se utilizar uma cor de fundo negra para coincidir com a cor de fundo de todo o *layout* e optou-se pela cor de fonte branca para toda a informação do evento, uma vez que se pretende dar o maior contraste possível, com vista a melhorar a visibilidade.

No que toca à quantidade de informação dos eventos apresentados, optou-se por excluir a data/hora de fim (evento com duração indefinida), o local (não há razões para apresentar o local, visto que é no edifício da Ubisign) e a imagem (o *layout* já tinha sido desenhado para poder conter uma imagem estática do lado esquerdo e, neste caso, nem está relacionada com os eventos apresentados).

Avançando para a análise da Figura 5.14, verifica-se que as regiões 2 e 3 do *layout* da Figura 5.11 têm conteúdos em formato texto, apenas ficando a região 1 vazia. Portanto, na região 2 destaca-se a visualização Column, com informação sobre eventos que vão ocorrer na Ubisign e na região 3 corre informação sobre acontecimentos actuais, com a respectiva fonte e a data da sua ocorrência. Como é óbvio, vai-se dar particular destaque à visualização Column, uma vez que se trata do único conteúdo processado pela DSEventApp, como se pode observar na Figura 5.15.

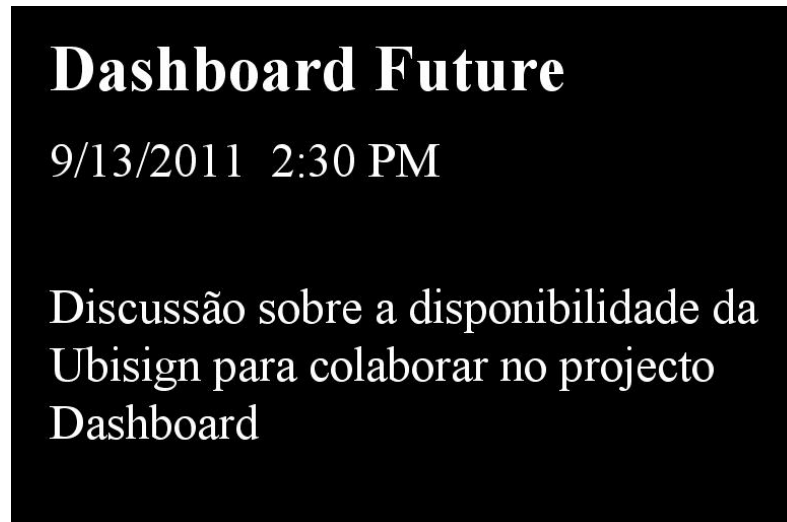
Considerando a configuração da visualização Column, é de referir que optou-se por utilizar cores de fundo iguais, para eventos consecutivos, para preservar a cor de fundo negra de todo o *layout*. Já no que diz respeito à cor de fonte de eventos consecutivos, resolveu-se utilizar cores distintas (a cor



**Figura 5.11:** *Layout utilizado para os primeiros dois exemplos do serviço Ubi-sign.com*



**Figura 5.12:** *Visualização Full enquadrada na região 3 do layout da Figura 5.11*



**Figura 5.13:** Destaque da visualização *Full* existente no exemplo da Figura 5.12

branca e amarela que se contrastam bem no fundo negro), uma vez que se pretende impedir confusões por parte do utilizador. Por exemplo, utilizando a mesma cor para os dois eventos da coluna, pode-se dar a ideia errada de que se trata de um único evento.

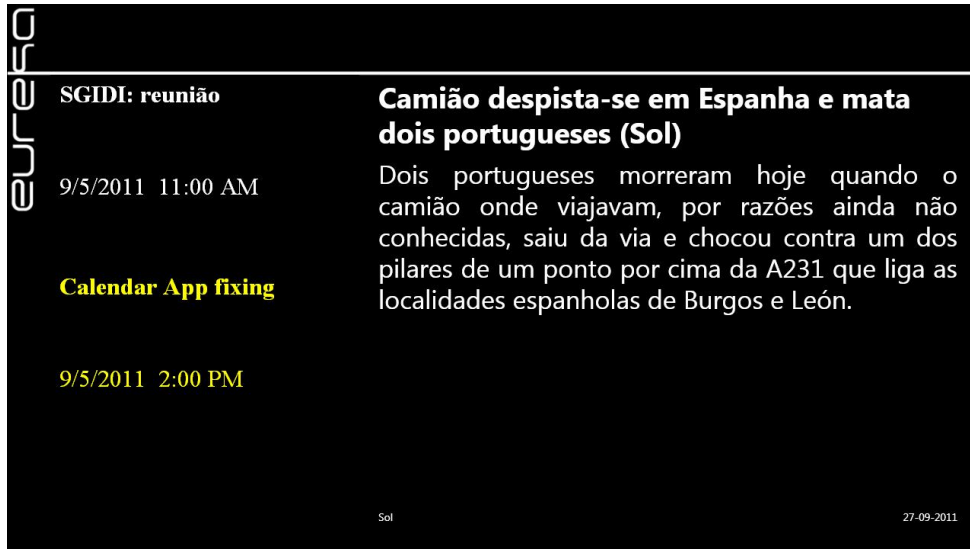
No respeitante à quantidade de informação dos eventos apresentados em coluna, resolveu-se não incluir o sítio, visto que são eventos locais, e também se optou por não incluir um ícone.

Outro aspecto a considerar, é o facto da visualização *Column* apenas conter dois eventos, certamente para não prejudicar a visibilidade, uma vez que em *digital signage* as pessoas não passam muito tempo “coladas” aos *displays*, contrariamente ao que acontece com os computadores pessoais.

Passando para o último exemplo (Figura 5.17), destacam-se conteúdos em todas as regiões do *layout* da Figura 5.16 (*layout* com quatro regiões). Na região 1 (canto superior direito do *layout*) observa-se um *widget* com a informação sobre o tempo, nas regiões 2 e 3 sobressaem duas imagens de modelos e na região 4 apresenta-se a visualização *Bar Scroll* (não se considera a visualização *Bar Fade*, porque é muito semelhante), processada pela *DSEventApp*, que é o único conteúdo concretizado por esta aplicação (como se pode observar a Figura 5.18).

Considerando a configuração da visualização *Bar Scroll*, também foi utilizada a cor de fundo negra e cor de fonte branca, pelos mesmos motivos existentes nos dois exemplos anteriores. Também só se apresenta informação sobre o título, data de início e descrição, não incluindo data de fim (certamente por opção).

Para concluir, pode-se dizer que a *DSEventApp* tem utilidade prática no serviço *Ubisign.com*, tornando-o mais rico, no sentido em que possibilita

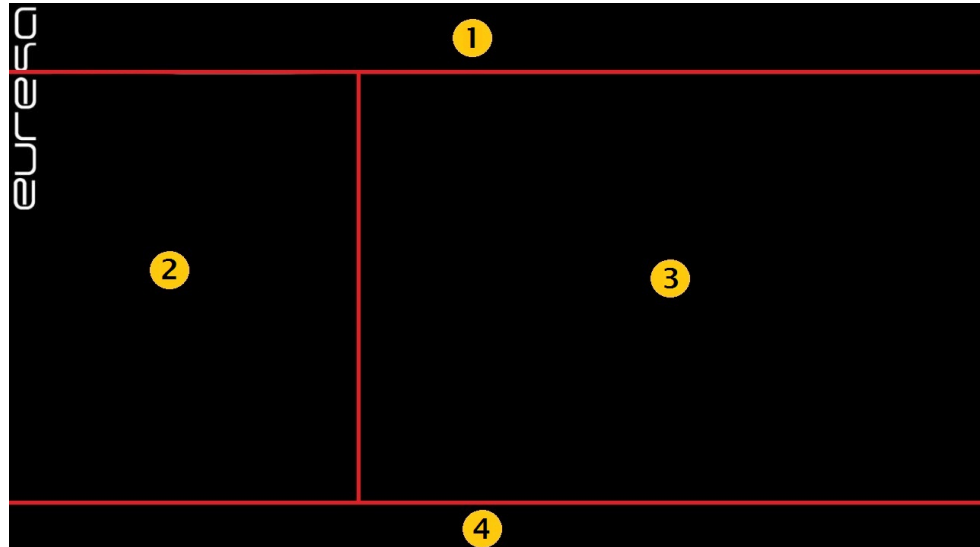


**Figura 5.14:** Visualização Column enquadrada na região 2 do layout da Figura 5.11

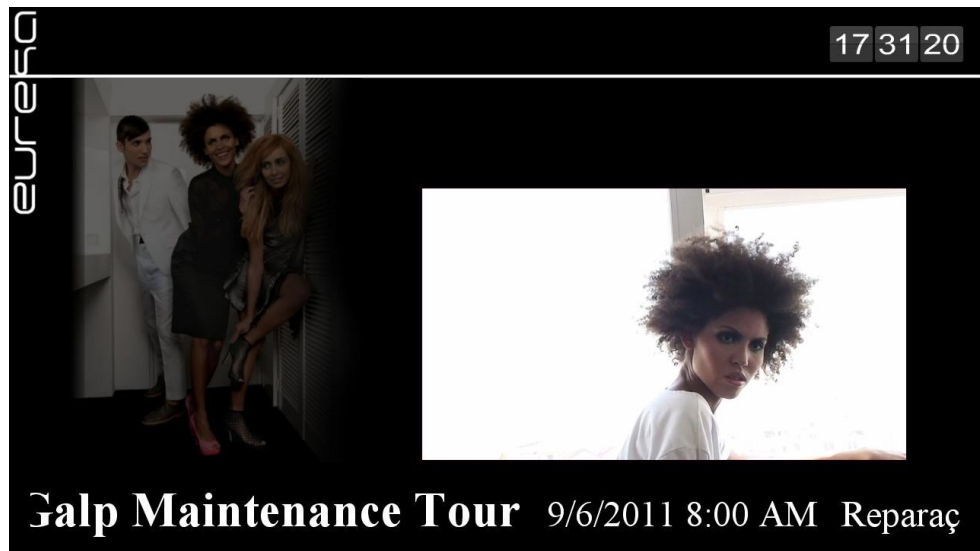


**Figura 5.15:** Destaque da visualização Column existente no exemplo da Figura 5.14





**Figura 5.16:** Layout utilizado para o último exemplo do serviço Ubisign.com



**Figura 5.17:** Visualização Bar Scroll enquadrada na região 4 do layout da Figura 5.16

**Galp Maintenance Tour 9/6/2011 8:00 AM Reparaç**

**Figura 5.18:** Destaque da visualização Bar Scroll existente no exemplo da Figura 5.17

a extracção de informação sobre eventos do Google Calendar e, também, permite que essa informação seja apresentada em três tipos de visualização comuns para qualquer *layout* criado pelo utilizador.

Ainda é importante dizer que o sucesso das aplicações desenvolvidas para painéis de *digital signage* depende muito da qualidade do *design* do *layout*, do *background*, dos conteúdos, etc.. Assim, é de notar que nos três exemplos apresentados pela Ubisign, verifica-se claramente que existe um *layout* bem pensado, um *background* atractivo e uma boa selecção dos conteúdos apresentados.



## Capítulo 6

# Conclusões

### 6.1 Sumário

Nesta secção vai ser feito um sumário sobre tudo o que foi concretizado nesta dissertação, sendo que cada um dos capítulos serão analisados de uma forma sucinta.

No capítulo 1 foi feita a introdução sobre painéis de *digital signage*, para enquadrar o leitor no tema do projecto, proposto pela empresa Ubisign, que é o desenvolvimento de uma aplicação *web* para painéis de *digital signage*, com o intuito de proporcionar informação sobre eventos existentes no Google Calendar e permitir que o utilizador customize o modo como a informação é apresentada, através de formulários próprios.

Durante este capítulo foi referida a motivação para o desenvolvimento da aplicação, designada por DSEventApp, que passa essencialmente pela integração desta no serviço Ubisign.com, com o propósito de alargar as fontes de informação proporcionadas pelo serviço. Para além disso, foram apresentados vários objectivos a atingir durante a escrita da dissertação e foi apresentada a organização que a dissertação deve cumprir.

Considerando o capítulo 2, pode-se dizer que se efectuou o levantamento sobre o estado da arte ligado à *digital signage*, com base em vários estudos efectuados na área, com vista a transmitir ao leitor os variados conceitos que giram à volta dessa tecnologia e de prepará-lo para os próximos capítulos (sobre o desenvolvimento da aplicação *web*).

Passando para o capítulo 3, é importante dizer que foi abordado o problema proposto pela Ubisign, tratando-se do desenvolvimento de uma aplicação capaz de apresentar informação sobre eventos, recorrendo à API Google Calendar. Assim sendo, foi detalhada a ferramenta Google Calendar e, adicionalmente, algumas ferramentas *on-line* de gestão de eventos, nomeadamente, a Microsoft Exchange, o Eventbrite e o Facebook. De seguida, foi descrita a aplicação a elaborar, tendo sido dado destaque às visualizações em ecrã inteiro (Full), em coluna (Column), em barra horizontal (Bar) e em XML

(com eventos retornados pelo Google Calendar, para serem consumidos por outras aplicações).

No final deste capítulo foi apresentado o Modelo do Domínio associado ao problema, juntamente com a descrição sobre as respectivas entidades e associações existentes entre estas. Também foram especificados os requisitos funcionais e não funcionais sobre o problema em questão.

No capítulo 4 foi analisado todo o processo inerente ao desenvolvimento da solução de *software*, apresentando toda a especificação ligada à DSEventApp. Nesse sentido, foi efectuado um maior desenvolvimento aos requisitos propostos, dando principal destaque aos parâmetros que a DSEventApp consome, com o objectivo de produzir visualizações Full, Column, Bar e XML.

De seguida, foi feita a especificação da DSEventApp ao nível da camada da interface, negócio e dados. Portanto, ao nível da camada da interface, foram apresentados esboços sobre as visualizações e sobre os formulários de configuração. Também foram apresentados dois diagramas de sequência de alto nível, que representam a sequência de acções que o utilizador pode efectuar ao nível dos formulários. Quanto à camada de negócio, foi apresentado um diagrama de classes por cada *package* especificado, contendo várias classes e respectivas interacções, sendo que os seus métodos compõem as funcionalidades que a aplicação suporta. Considerando a camada de dados, foram especificadas as entidades (ou classes persistentes) e as respectivas relações (diagrama ER), uma vez que se optou por utilizar a *framework* NHibernate para .NET. Por fim, foi descrito todo o processo de implementação, tomando decisões e apresentando soluções para certos problemas encontrados, e, ainda, foi dado um exemplo para cada visualização (Full, Column, Bar e XML) processada pela DSEventApp, juntamente com a respectiva *query string* de configuração.

No capítulo 5, destaca-se o facto de se apresentar três exemplos (sobre a instalação da DSEventApp em *displays* localizados em hotéis, aeroportos e quartéis de bombeiros) com grande potencial para serem aplicados na realidade, de modo a demonstrar a utilidade da DSEventApp. Para provar, de uma forma efectiva, que a aplicação está pronta para ser aplicada no mundo real, foram apresentados três exemplos resultantes da integração desta no serviço Ubsign.com. Cada exemplo corresponde ao uso de um tipo de visualização distinta (Full, Column e Bar), juntamente com outros conteúdos proporcionados pelo serviço.

## 6.2 Discussão

Nesta secção vão ser abordados os objectivos definidos no capítulo 1, Introdução, por forma a comprovar que estes foram integralmente cumpridos com sucesso e serão feitas as conclusões finais sobre a aplicação DSEventApp.

Dito isto, importa dizer que a primeira lista de objectivos corresponde ao

estudo do estado da arte associado ao conceito de *digital signage* e muitos outros aspectos relacionados com este tipo de tecnologia, podendo-se destacar os seguintes objectivos concretizados com sucesso:

- Foi desenvolvido o conceito de *digital signage*, juntamente com os principais factores associados, isto é, os factores técnicos, estratégicos, comerciais e psicológicos. Deu-se um destaque particular aos factores psicológicos, uma vez que o sucesso dos painéis de *digital signage* depende do nível de interesse que as pessoas lhes dão;
- Foram considerados três *surveys* sobre a modelação em *displays* públicos, sendo que no primeiro *survey*, foi feito um estudo de *displays* em ambientes exteriores, considerando uma audiência vasta; o segundo *survey* trata de obter opiniões sobre *displays* instalados num ambiente académico; o último tem em conta opiniões sobre *displays* localizados num ambiente laboratorial, restrito a grupos pequenos de pessoas;
- Efectuou-se uma introdução sobre os problemas que a publicidade excessiva provoca no contexto dos *displays* públicos, localizados em grandes cidades, e foram consideradas algumas formas para prevenir esses excessos;
- Consideraram-se dois exemplos de *displays* públicos inteligentes, o *BluScreen* e o *iDisplay*, que apresentam anúncios publicitários, com base no contexto envolvente, utilizando um sistema à base de leilões.

Passando para a segunda lista de objectivos, é importante dizer que está relacionada com o problema proposto pela empresa Ubisign, ou seja, o desenvolvimento de uma aplicação *web* que apresenta informação sobre eventos (existentes no Google Calendar), com vista a ser integrada no serviço Ubisign.com. De seguida, apresentam-se os objectivos cumpridos com sucesso:

- Foram abordadas algumas ferramentas *on-line* de gestão de eventos, como a Microsoft Exchange, o Eventbrite, o Facebook, mas detalhando, mais ao pormenor, as funcionalidades do Google Calendar. Também é feita uma introdução sobre APIs da Google, nomeadamente a do Google Calendar, e é desenvolvida a questão, muito importante, sobre a autenticação e autorização, dando particular destaque aos protocolos OpenID e OAuth, com vista a serem uma boa solução para permitir o acesso ao Google Calendar, por parte da aplicação *web* elaborada;
- Foi efectuada uma descrição sobre a aplicação desenvolvida, sendo detalhadas as visualizações que a aplicação processa (visualizações Full, Column, Bar e XML). Foram introduzidos três tipos de parâmetros de configuração (parâmetros da componente visual, comportamental e de filtragem) que a aplicação recebe para processar visualizações. Ainda

se consideraram os formulários de configuração, com vista a possibilitar a configuração de visualizações, de acordo com o que o utilizador pretende. Abordou-se a possibilidade de armazenar, actualizar ou remover configurações (especificadas pelo utilizador) e, também, a disponibilização de configurações que vêm de raiz com a aplicação (configurações *default*);

- Efectuou-se a apresentação e a análise do Modelo do Domínio, com vista a especificar as entidades importantes para o domínio do problema (neste caso a aplicação *web* sobre eventos);
- Foi feita a análise de requisitos da aplicação implementada, especificando os requisitos funcionais e não funcionais.

A terceira lista de objectivos está relacionada com a solução, ou seja, a especificação e o desenvolvimento da aplicação *web*, designada por DSEventApp. Assim, apresentam-se os seguintes objectivos concluídos com sucesso:

- Foi especificada a quantidade de informação apresentada para cada tipo de visualização (Full, Column, Bar e XML) e foram detalhados, à base de tabelas, os parâmetros de configuração da componente visual, de filtragem e comportamental;
- Especificou-se a DSEventApp ao nível da camada da interface, negócio e dados. Considerando a camada da interface, realizaram-se esboços para as visualizações Full, Column, Bar e XML, e alguns *mockups* que constituem o desenho dos formulários de configuração. Também foi realizado um diagrama de sequência, de alto nível, para indicar a operação de *update* de uma configuração e um outro para demonstrar as interacções existentes no sistema, após a submissão de um formulário de configuração. Quanto à camada de negócio, realizaram-se diagramas de classe organizados por *packages*, para que houvesse uma boa estruturação do código, com o objectivo de facilitar a fase de implementação. Para terminar, foi especificada a camada de dados com base na *framework* NHibernate (para efectuar o mapeamento ORM), sendo que foram modeladas as entidades (através de um diagrama de classes) e as correspondentes relações (através de um diagrama ER);
- Desenvolveu-se a DSEventApp com base num planeamento acordado com a Ubisign, contendo tarefas e respectivos prazos. Pode-se dizer que, regra geral, correu tudo de acordo com o planeado e os problemas que, por vezes, surgiram foram resolvidos;
- Descreveram-se as fases de implementação da aplicação *web* proposta, com base no planeamento acordado com a Ubisign, tomaram-se algumas decisões face aos problemas ocorridos durante a fase de implementação da DSEventApp e, por fim, foram apresentados alguns exemplos

simples das visualizações Full, Column, Bar e XML, juntamente com as respectivas *query strings* de configuração.

Quanto à última lista de objectivos, é importante dizer que tem a finalidade de provar que a DSEventApp pode ser utilizada em contextos reais, através da elaboração de alguns casos de estudo, destacando-se os seguintes objectivos concretizados com sucesso:

- Foram apresentados três casos hipotéticos de utilização da DSEventApp, com grande potencial para serem concretizados na realidade, com base em aplicações externas que tiram partido das visualizações processadas pela DSEventApp. O primeiro caso consiste na apresentação de informação sobre eventos desportivos que vão ocorrer num hotel, sendo que foram utilizadas as visualizações Full e Column em simultâneo. Passando para o segundo caso, apresentaram-se dois exemplos com um mapa de voos para um dado aeroporto, em que o primeiro consiste numa aplicação externa que utiliza seis visualizações Bar Fade (cada uma com informação de um voo) e uma visualização Bar Scroll (com informação de cariz urgente), e o segundo exemplo consiste numa aplicação externa que recorre a três visualizações Column (cada uma com três voos) e também uma visualização Bar Scroll, idêntica ao primeiro exemplo. Por fim, o terceiro caso diz respeito a uma aplicação externa sobre incêndios na região do Minho. Esta apresenta a localização de incêndios no Google Maps e a sua respectiva informação, sendo esta extraída através do *parsing* da visualização XML;
- Provou-se o bom funcionamento da DSEventApp num contexto real, ou seja, a aplicação foi integrada, com sucesso, no serviço Ubisign.com. Primeiro foi descrito o serviço Ubisign.com e, de seguida, foram dados três exemplos dedicados às visualizações Full, Column e Bar, juntamente com outros conteúdos proporcionados pelo serviço.

Concluindo, a DSEventApp é particularmente útil para ser utilizada em aplicações externas, principalmente no caso das visualizações Column, Bar e XML. As duas primeiras são apresentadas na totalidade do *display*, de forma a que outras aplicações possam integrá-las num *layout* adequado e a visualização XML é feita propositadamente para ser consumida por aplicações externas. Quanto à visualização Full, importa dizer que pode ser utilizada de forma isolada, desempenhando um papel importante, no sentido em que disponibiliza ao utilizador um grande número de alternativas de configuração.

Outro ponto importante é que nos painéis de *digital signage* não são só apresentados eventos e, nestes casos, a DSEventApp não é a solução ideal. Muitas vezes nos rodapés é apresentada informação de carácter urgente com o intuito de avisar as pessoas. Por exemplo, nas auto-estradas quando há



obras ou outro tipo de imprevistos, é frequente a utilização de *displays* com o objectivo de alertar para as situações de potencial perigo. Estes avisos não são programados nem têm uma delimitação temporal pré-definida e, por estes motivos, não são adequados para serem introduzidos no Google Calendar.

O sucesso da DSEventApp depende muito da forma como é usado pelos utilizadores, pois está extremamente relacionado com a configuração feita ao nível da componente visual, ou seja, é importante saber escolher bem as cores de fonte e de fundo de forma a que se contrastem, o tamanho de fonte deve ser suficientemente visível e a quantidade de texto existente nos eventos não deve ser exagerada. Além disso, a DSEventApp já foi implementada de forma a permitir que aplicações externas possam comunicar através de uma *query string* de configuração e, assim, pode haver uma customização ainda mais profunda das visualizações apresentadas. Geralmente, cabe aos *designers* definir o que pretendem para que as visualizações sejam mais atractivas para o público que passa pelos *displays*.

Numa fase inicial a Ubisign propôs que a DSEventApp possibilitasse a configuração de visualizações no modo Full, Column e Bar. Posteriormente, decidiu-se incluir o tipo de visualização XML e chegou-se à conclusão de que seria muito útil permitir o armazenamento de configurações numa base de dados. Desta forma, as aplicações externas não têm a necessidade de gerir configurações, sendo esta função proporcionada pela DSEventApp.

Para terminar, pode-se dizer que foram cumpridos todos os objectivos propostos no início da dissertação, juntamente com a implementação da aplicação *web*, DSEventApp, que apresenta informação sobre eventos em painéis de *digital signage*, recorrendo à API Google Calendar.

## 6.3 Trabalho Futuro

Nesta última secção vão ser sugeridas algumas ideias para melhorar a qualidade e utilidade da aplicação desenvolvida.

**Incluir animações nas transições** Estender a DSEventApp, de modo a que esta possibilite ao utilizador a escolha de animações diferentes, ao nível das transições, para as visualizações Full, Column e Bar, no sentido de aumentar, ainda mais, o número de possibilidades de configuração das visualizações, em que o utilizador poderá especificar transições mais suaves e interessantes no momento do aparecimento da informação sobre um dado evento e, também, no momento em que a informação desaparece, passando para a próxima iteração. No entanto, como a visualização Column é estática, só poderá ter uma animação no momento em que aparecem os vários eventos em coluna (não há a noção de transição entre eventos consecutivos, como ocorre nas visualizações Full e Bar).

**Noção de *themes*** Criar a noção de *themes* para as configurações *default*, juntamente com *thumbnails* para o utilizador saber mais facilmente o aspecto das visualizações *default*. O utilizador através do *thumbnail*, selecciona o *theme* que achar mais interessante e o formulário é preenchido automaticamente com os parâmetros da configuração seleccionada, podendo, desta forma, personalizar o *theme* (ou simplesmente deixá-lo como está) e utilizá-lo na apresentação dos eventos. Actualmente, a aplicação apenas suporta configurações *default* organizadas em várias listas, cada uma com um identificador para o utilizador saber do que tratam, não havendo organização à base de *themes* nem nenhum *thumbnail*.

**Actualização dinâmica dos eventos apresentados** Recorrer à tecnologia AJAX para que a DSEventApp, de X em X minutos, actualize os eventos apresentados. Neste momento, o parâmetro de actualização dos dados obtidos do serviço Google Calendar (ModelRefreshInterval) apenas refresca os eventos através de um novo pedido externo (por exemplo, a partir de uma aplicação externa) e, com este *upgrade*, esse parâmetro já funciona automaticamente e de forma cíclica, bastando definir o intervalo de tempo de actualização, para que sejam devolvidas, automaticamente, novas visualizações com os eventos refrescados (actualização cíclica).

**Aumentar o número de serviços suportados** Estender a DSEventApp de modo a suportar o acesso aos seguintes serviços: Eventbrite, Microsoft Exchange, Facebook e, possivelmente, outras mais. Para que tal aconteça, é necessário estudar as respectivas APIs, identificar as possíveis limitações e resolver os problemas detectados (processo semelhante ao realizado para o serviço Google Calendar).

**Possibilitar a criação de configurações genéricas** Uma vez suportadas várias APIs, permitir que o utilizador crie configurações genéricas, de modo a que possam ser utilizadas em serviços diferentes. Para isso a autenticação tem que ser independente dos serviços e, para o utilizador aceder aos serviços desejados, a aplicação deve ter os respectivos *tokens* de autenticação/autorização armazenados na base de dados. É de notar que as configurações *default* também são genéricas (podem ser utilizadas por qualquer serviço), mas a aplicação não permite a criação/actualização/remoção de configurações *default* pelo utilizador normal, é apenas o administrador da base de dados que tem essa possibilidade.

**Formulários para o administrador gerir configurações *default*** Para que não seja necessário um administrador, especialista em base de dados, a efectuar operações CRUD (operações de *create*, *read*, *update*

e *delete*) ao nível das configurações *default*, é necessário desenvolver formulários para configurar visualizações Full, Column e Bar, possibilitando uma gestão mais simples. Isto é idêntico à gestão que o utilizador pode realizar com as suas configurações, só que num contexto administrativo e não acessível para o utilizador normal.

# Bibliografia

- ABRAMS, R. A. AND CHRIST, S. E. 2003. Motion onset captures attention. *Psychological Science* 14, 427–432.
- AGAMANOLIS, S. 2003. Designing displays for human connectedness.
- AMIRI, A. AND MENON, S. 2003. Efficient scheduling of internet banner advertisements. *ACM Trans. Internet Techn.* 3, 4, 334–346.
- BIRREN, F. 1961. Color psychology color therapy. *Secaucus, New Jersey: University Books Inc.*
- BÜGG, B. 2005. Digital signage networks: Theory, psychology and strategy. *Pixel Inspiration Ltd.*
- CARTER, S., CHURCHILL, E. F., DENOUE, L., HELFMAN, J., AND NELSON, L. 2004. Digital graffiti: public annotation of multimedia content. In *CHI Extended Abstracts*, E. Dykstra-Erickson and M. Tscheligi, Eds. ACM, 1207–1210.
- CHURCHILL, E., NELSON, L., AND DENOUE, L. 2003. Multimedia fliers: Informal information sharing with digital community bulletin boards. *Proceedings of Communities and Technologies*.
- CZERWINSKI, M., SMITH, G., REGAN, T., MEYERS, B., ROBERTSON, G. G., AND STARKWEATHER, G. 2003. Toward characterizing the productivity benefits of very large displays. In *INTERACT*, M. Rauterberg, M. Menozzi, and J. Wesson, Eds. IOS Press.
- EDELMAN, B. 2005. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. *Working Paper*, <http://rwj.berkeley.edu/schwarz/>.
- FASS, A. M., FORLIZZI, J., AND PAUSCH, R. 2002. Messydesk and messyboard: two designs inspired by the goal of improving human memory. In *Symposium on Designing Interactive Systems*. 303–311.
- FRANCONERI, S. L., HOLLINGWORTH, A., AND SIMONS, D. J. 2005. Do new objects capture attention? *Psychological Science* 16, 275–281.

- GOLD, N., KNIGHT, C., MOHAN, A., AND MUNRO, M. 2004. Understanding service-oriented software. *IEEE Software* 21, 2, 71–77.
- GREENBERG, S. AND ROUNDING, M. 2001. The notification collage: posting information to public and personal displays. In *CHI*. 514–521.
- GUIMERÀ, R., ARENAS, A., DÍAZ-GUILERA, A., VEGA-REDONDO, F., AND CABRALES, A. 2002. Optimal network topologies for local search with congestion. *CoRR cond-mat/0206410*.
- HONG, J. I. AND LANDAY, J. A. 2000. Satin: a toolkit for informal ink-based applications. In *UIST*. 63–72.
- HUANG, E. M., KOSTER, A., AND BORCHERS, J. 2008. Overcoming assumptions and uncovering practices: When does the public really look at public displays? In *Pervasive*, J. Indulska, D. J. Patterson, T. Rodden, and M. Ott, Eds. Lecture Notes in Computer Science, vol. 5013. Springer, 228–243.
- HUANG, E. M. AND MYNATT, E. D. 2003. Semi-public displays for small, co-located groups. In *CHI*, G. Cockton and P. Korhonen, Eds. ACM, 49–56.
- INTILLE, S. S. 2002. Change blind information display for ubiquitous computing environments. In *UbiComp*, G. Borriello and L. E. Holmquist, Eds. Lecture Notes in Computer Science, vol. 2498. Springer, 91–106.
- KRISTJANSSON, A., MACKEBEN, M., AND NAKAYAMA, K. 2001. Rapid, object-based learning in the deployment of transient attention. *Perception* 30, 1375–1387.
- MACINTYRE, B., MYNATT, E. D., VOIDA, S., HANSEN, K. M., TULLIO, J., AND CORSO, G. M. 2001. Support for multitasking and background awareness using interactive peripheral displays. In *UIST*. 41–50.
- MCCARTHY, J. F., COSTA, T. J., AND LIONGOSARI, E. S. 2001. Unicast, outcast & groupcast: Three steps toward ubiquitous, peripheral displays. In *UbiComp*, G. D. Abowd, B. Brumitt, and S. A. Shafer, Eds. Lecture Notes in Computer Science, vol. 2201. Springer, 332–345.
- MÜLLER, H. J. AND RABBITT, P. M. A. 1989. Reflexive and voluntary orienting of visual attention: Time course of activation and resistance to interruption. *JOURNAL OF EXPERIMENTAL PSYCHOLOGY: HUMAN PERCEPTION AND PERFORMANCE* 15, 2, 315–330.
- MÜLLER, J. 2007. How much to bid in digital signage advertising auctions? *Adjunct proceedings of Pervasive 2007*.

- MÜLLER, J. 2008. Persuasion with smart digital signage. *Surrounded by Ambient Persuasion Workshop in Conjunction with CHI, Florence*.
- MÜLLER, J. 2009. Traditional and digital signage. In *GI Jahrestagung*, S. Fischer, E. Maehle, and R. Reischuk, Eds. LNI, vol. 154. GI, 3882–3887.
- MÜLLER, J. AND KRÜGER, A. 2007. Competing for your attention: Negative externalities in digital signage advertising. In *Ambient Information Systems*, W. R. Hazlewood, L. Coyle, and S. Consolvo, Eds. CEUR Workshop Proceedings, vol. 254. CEUR-WS.org.
- MÜLLER, J. AND KRÜGER, A. 2007. User Profiling for Generating Bids in Digital Signage Advertising Auctions. In *Workshop on Ubiquitous and Decentralized User Modeling, User Modeling 2007, Corfu*.
- NAKAYAMA, K. AND MACKEBEN, M. 1989. Sustained and transient components of focal visual attention. *Vision Research* 29, 1631–1647.
- PAYNE, T. R., DAVID, E., JENNINGS, N. R., AND SHARIFI, M. 2006. Auction mechanisms for efficient advertisement selection on public displays. In *ECAI*, G. Brewka, S. Coradeschi, A. Perini, and P. Traverso, Eds. Frontiers in Artificial Intelligence and Applications, vol. 141. IOS Press, 285–289.
- RASHID, U. AND QUIGLEY, A. J. 2009. Ambient displays in academic settings: Avoiding their underutilization. *IJACI* 1, 2, 31–38.
- SALBER, D., DEY, A., ORR, R., AND ABOWD, G. 1999a. Designing for Ubiquitous Computing: A Case Study on Context Sensing.
- SALBER, D., DEY, A. K., ORR, R. J., AND ABOWD, G. D. 1999b. Designing for ubiquitous computing: A case study in context sensing. Tech. rep., Usability Center.
- TERRELL, G. B. AND MCCRICKARD, D. S. 2006. Enlightening a co-located community with a semi-public notification system. In *CSCW*, P. J. Hinds and D. Martin, Eds. ACM, 21–24.
- TSE, P. U., RIVEST, J., INTRILIGATOR, J., AND CAVANAGH, P. 2004. Attention and subjective expansion of time. *Perception and Psychophysics* 66, 1171–1189.
- VICKREY, W. 1961. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 8–37.
- WEISER, M. 1993. Some computer science issues in ubiquitous computing. *Commun. ACM* 36, 7, 74–84.

- ZHAO, Q. A. AND STASKO, J. T. 2002. What's happening?: promoting community awareness through opportunistic, peripheral interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces*. AVI '02. ACM, New York, NY, USA, 69–74.