



AppTour

Instituto Superior de Engenharia do Porto

2011 / 2012

1090702 Pedro Luís Cardoso Vasques Ferreira

1090704 Ricardo Jorge Moutinho Soares Carneiro



AppTour

Instituto Superior de Engenharia do Porto

2011 / 2012

1090702 Pedro Luís Cardoso Vasques Ferreira

1090704 Ricardo Jorge Moutinho Soares Carneiro



Licenciatura em Engenharia Informática

Julho 2012

Orientador ISEP: **Doutor Paulo Alexandre Gandra de Sousa**

*«Aos meus pais e amigos. Pedro Ferreira.
Aos meus pais, irmã e namorada. Ricardo Carneiro»*

Agradecimentos

Agradeço a todos os que tornaram possível a realização deste projecto, por muito ínfima que possam considerar a sua ajuda.

Agradeço, em particular, ao **Doutor Paulo Gandra de Sousa** pelo tempo despendido durante a realização do projecto em acompanhamento e orientação, assim como pela ajuda prestada durante as várias fases do mesmo;

Ao meu colega e amigo, **Ricardo Jorge Carneiro**, por toda a paciência necessária nos momentos menos produtivos do projecto e consequente motivação, assim como o fornecimento de todas as condições necessárias para que este projecto pudesse ser realizado até ao final;

Finalmente, agradeço aos meus **pais e irmãos** pelo apoio dado durante as alturas em que o cansaço se começava a instalar, assim como naturalmente por todos os sacrifícios que fizeram possível a minha entrada no Instituto.

A todos, o meu sincero agradecimento.

Pedro Luís Cardoso Vasques Ferreira

Tenho o privilégio de agradecer às pessoas que me ajudaram ao longo do percurso, que acreditaram em mim e que, de uma forma ou de outra, me apoiaram.

Ao **Doutor Paulo Gandra de Sousa**, pelo empenho, dedicação e competência demonstrada na orientação deste projecto, pela disponibilidade e tempo disponibilizado e por todas as críticas feitas em tempo oportuno.

Ao **Pedro Ferreira**, colega e amigo, pela dedicação, desempenho neste projecto e por todos os momentos passados ao longo do mesmo.

Aos meus **PAIS** e **IRMÃ**, pela paciência, por todo o apoio, pelos gestos, palavras e por todo o amor incondicional.

À minha **NAMORADA**, pela extrema compreensão, pelos momentos adiados e pelo apoio incondicional no decorrer do curso e particularmente no percurso deste projecto.

Por todos os gestos e palavras, o meu sincero obrigado.

Ricardo Jorge Moutinho Soares Carneiro

Resumo

Com o surgimento dos novos *smartphones* e com as potencialidades de mobilidade e comunicação, assim como a quantidade enorme de sensores que estes oferecem, emergiu uma nova necessidade em criação de aplicações que dessem uso a essas mesmas funcionalidades. Um exemplo dessas aplicações passa pelo uso das potencialidades de localização geográfica e interacção com o meio ambiente que os rodeia, vulgarmente apelidadas de geo-aplicações.

Com esse objectivo, propõe-se neste projecto, a elaboração de uma aplicação que fornece informação distinta para turistas acerca do contexto de uma cidade, assim como os vários pontos de interesse que estas possam fornecer ou eventos que possam ocorrer, baseando-se na localização destes. Esta informação será previamente recolhida e tratada, assim como dividida em diversos temas por forma a auxiliar a escolha de resultados relevantes.

O desenvolvimento deste projecto atingiu os objectivos primordiais a que se propôs, embora tivessem sido encontradas algumas dificuldades que tivessem de ser ultrapassadas. Embora ainda careça de algum desenvolvimento futuro, as principais funcionalidades propostas inicialmente pela aplicação foram cumpridas, o que permite a um qualquer utilizador o uso deste sistema de uma forma imediata.

Palavras Chave (Tema): Geo-Aplicações, Geo-Localização, Turismo, Pontos Interesse, Eventos.

Palavras Chave (Tecnologias): Android, .NET, MVC, WCF, Entity

Abstract

With the arising of smartphones and the potential they bring in mobility and communications, as well as the huge number of sensors they offer, a new need for applications that use these same features emerged. An example of these applications is the use of geographical location and potential interaction with the environment around them, commonly named geo-applications.

With this at its goal, this project proposes the development of an application that can provide distinct and useful information for a tourist about the context of a city, as well as the various points of interest it can provide or events that may occur, based on the user's location. This information is previously collected and treated, as divided in several subjects in order to assist the selection of relevant results.

The developments of this project meet the primary objectives it set itself, although they were found some difficulties to be overcome. Although still lacking some future development, the main features initially proposed by the application have been met, which allows any user to use this system in an immediate way.

Keywords (Subject): Geo-Applications, Geo-Location, Tourism, Points of Interest, Events.

Keywords (Technology): Android, .NET, MVC, WCF, Entity

Índice

<i>Agradecimentos</i>	<i>vi</i>
<i>Resumo</i>	<i>ix</i>
<i>Abstract</i>	<i>xi</i>
<i>Índice</i>	<i>xiii</i>
<i>Índice de Figuras</i>	<i>xvi</i>
<i>Índice de Tabelas</i>	<i>xxi</i>
<i>Notação e Glossário</i>	<i>xxiii</i>
1 Introdução.....	1
1.1 Enquadramento	1
1.2 Objectivos	2
1.3 Contributos deste Trabalho	3
1.4 Processo de Desenvolvimento	4
1.4.1 Planeamento de projecto	4
1.4.2 Reuniões de acompanhamento	1
1.5 Linguagens e Tecnologias	4
1.6 Organização do relatório	8
2 Análise	9
2.1 Termos e Conceitos	9
2.2 Solução Pretendida	11
2.2.1 Requisitos Básicos	11
2.2.2 Requisitos Complementares	16
2.2.3 Requisitos não Funcionais.....	17
2.3 Visão Geral.....	19
2.3.1 Repositório	20
2.3.2 Lógica de Negócio	20
2.3.3 Web Services.....	20
2.3.4 Web Portal	21
2.3.5 App Android	22
2.3.6 Agentes	22
2.4 Modelo de Domínio.....	23
2.5 Casos de Uso	25
2.5.1 Efectuar uma pesquisa.....	27

2.5.2	Alterar o ponto de localização	27
2.5.3	Considerar apenas Pontos disponíveis	28
2.5.4	Comentar um Ponto/ Evento.....	28
2.5.5	Classificar um Ponto / Evento.....	28
2.5.6	Denunciar um comentário	29
2.5.7	Criar um novo Perfil de Pesquisa	29
2.5.8	Configurar um Perfil de Pesquisa.....	29
2.5.9	Eliminar um perfil de pesquisa	30
2.5.10	Calendarizar Eventos.....	30
2.5.11	Publicar Pontos e Eventos em redes sociais	30
2.5.12	Resolver denúncias	31
2.5.13	Inserir Pontos e Eventos	31
2.5.14	Alterar conteúdos em Pontos e Eventos	32
2.5.15	Alterar permissões Utilizador	32
2.5.16	Bloquear e desbloquear utilizadores	33
2.5.17	Procurar e resolver Pontos e Eventos duplicados	33
2.5.18	Recolha de informação automática	34
2.5.19	Classificação e armazenagem de informação.....	34
3	Desenvolvimento e Implementação	36
3.1	Arquitectura.....	37
3.2	Modelo de Dados	61
3.3	Aplicação Android	73
3.3.1	Arquitectura em Android	73
3.3.2	Actividades em Android	75
3.4	Implementação das funcionalidades.....	77
3.4.1	Efectuar uma pesquisa.....	77
3.4.2	Alterar o ponto de localização	80
3.4.3	Considerar apenas Pontos disponíveis	82
3.4.4	Comentar um Ponto/ Evento.....	83
3.4.5	Classificar um Ponto / Evento.....	86
3.4.6	Denunciar um comentário	88
3.4.7	Criar um novo Perfil de Pesquisa	88
3.4.8	Configurar um Perfil de Pesquisa.....	92
3.4.9	Eliminar um perfil de pesquisa	93

3.4.10	Calendarizar Eventos.....	93
3.4.11	Publicar Pontos e Eventos em redes sociais	94
3.4.12	Resolver denúncias	95
3.4.13	Inserir Pontos e Eventos	96
3.4.14	Alterar conteúdos em Pontos e Eventos	98
3.4.15	Alterar permissões Utilizador	102
3.4.16	Bloquear e desbloquear utilizadores	103
3.4.17	Procurar e resolver Pontos e Eventos duplicados	103
3.4.18	Recolha de informação automática.....	104
3.4.19	Classificação e armazenagem de informação.....	108
3.4.20	Multilingue	113
3.5	Testes Efectuados.....	118
4	Conclusões	119
4.1	Objectivos realizados	119
4.2	Limitações e trabalho futuro.....	123
4.3	Apreciação final	125
	Bibliografia.....	127
	Anexos.....	129
	Anexo 1 <i>Diagrama de Sequência de Pesquisa de Pontos</i>	129
	Anexo 2 <i>Diagrama de Sequência Inserir Comentário</i>	129
	Anexo 3 <i>Diagrama de Sequência Classificar um Ponto</i>	129
	Anexo 4 <i>Diagrama de Sequência Criar Perfil de Pesquisa</i>	129
	Anexo 5 <i>Diagrama de Sequencia Agente Google Locals</i>	129
	Anexo 6 <i>Diagrama de Sequencia Agents Controller.....</i>	129
	Anexo 7 <i>Diagrama de Sequência Agent Operations</i>	130
	Anexo 8 <i>Diagrama de Sequência de Inserção de Ponto</i>	130
	Anexo 9 <i>Diagrama de Sequência de Edição de Ponto</i>	130
	Anexo 10 <i>Documento de Casos de Uso</i>	130

Índice de Figuras

<i>Figura 1. Aplicação Foursquare</i>	1
<i>Figura 2. Planeamento do Projecto.....</i>	1
<i>Figura 3. Padrão MVC</i>	6
<i>Figura 4. Exemplo de Perfil de Pesquisa.....</i>	10
<i>Figura 5. Diagrama de Contexto.....</i>	19
<i>Figura 6. Modelo de Domínio.....</i>	24
<i>Figura 7. Casos de Uso</i>	26
<i>Figura 8. Logotipo AppTour</i>	36
<i>Figura 9. Sistema Backend AppTour (Black-Box)</i>	37
<i>Figura 10. Projectos de Backend - AppTour</i>	38
<i>Figura 11. Solução AppTour</i>	38
<i>Figura 12. Exemplo de registo de um agente</i>	39
<i>Figura 13. Exemplo do padrão Observer implementado nos Agentes</i>	40
<i>Figura 14. Organização do projecto ServiceContracts.....</i>	41
<i>Figura 15. Interface IPointService</i>	42
<i>Figura 16. Organização do projecto Services.....</i>	43
<i>Figura 17. Pequeno exemplo do serviço implemento para a entidade Ponto</i>	43
<i>Figura 18. Projecto Entity.....</i>	44
<i>Figura 19. Entity Framework – Modelo parcial AppTour.....</i>	45
<i>Figura 20. Projecto de Repositório</i>	46
<i>Figura 21. Exemplo de um método para devolver um Ponto</i>	47
<i>Figura 22. Exemplo da Classe PointRepository.....</i>	47
<i>Figura 23. Consulta filtrada no pedido à Base de Dados (boa abordagem)</i>	48
<i>Figura 24. Consulta filtrada no pedido à Base de Dados (má abordagem).....</i>	48
<i>Figura 25. Projecto Models</i>	49
<i>Figura 26. Exemplo parcial de um modelo.....</i>	49
<i>Figura 27. Exemplo de Serviços para o Ponto</i>	50
<i>Figura 28. Definição da classe BaseModel.....</i>	50
<i>Figura 29. Definição do método SelfValidate() do Modelo PointModel.....</i>	51
<i>Figura 30. Instalação do Fluent Validation no VS 2010</i>	51
<i>Figura 31. Definição do Validator do PointModel</i>	51
<i>Figura 32. Exemplo de validação de um ponto</i>	52

<i>Figura 33. Pasta Tools da solução com os respectivos projectos</i>	52
<i>Figura 34. Pasta da UI</i>	53
<i>Figura 35. Projecto UI Agents</i>	54
<i>Figura 36. Exemplo da aplicação Agentes em execução.</i>	54
<i>Figura 37. Projecto Web Portal</i>	55
<i>Figura 38. Conteúdo da pasta Content do Web Portal</i>	55
<i>Figura 39. Controllers no Web Portal</i>	56
<i>Figura 40. Pasta Helpers da aplicação Web Portal</i>	56
<i>Figura 41. Modelos da aplicação Web Portal</i>	57
<i>Figura 42. Resources na aplicação Web Portal</i>	57
<i>Figura 43. Scripts do Web Portal</i>	58
<i>Figura 44. Views do Web Portal</i>	58
<i>Figura 45. City Controller e respectivas Actions</i>	59
<i>Figura 46. Views da City</i>	59
<i>Figura 47. Método InsertComment do controlador PointController.</i>	59
<i>Figura 48. Projecto Web Services</i>	60
<i>Figura 49. Interface IAppTourWS que define os métodos WCF</i>	60
<i>Figura 50. Tabelas de base de dados AppTour</i>	61
<i>Figura 51. Instância da solução AppTour</i>	62
<i>Figura 52. Modelo E-R</i>	63
<i>Figura 53. Definição da tabela Agents</i>	64
<i>Figura 54. Definição da tabela City</i>	64
<i>Figura 55. Definição da tabela Classification</i>	65
<i>Figura 56. Definição da tabela Comments</i>	65
<i>Figura 57. Definição da tabela Country</i>	66
<i>Figura 58. Definições da tabela Event</i>	66
<i>Figura 59. Definições da tabela Event_Point</i>	66
<i>Figura 60. Definições da tabela Event_Topic</i>	67
<i>Figura 61. Definição da tabela Hour</i>	67
<i>Figura 62. Definição da tabela Hours</i>	68
<i>Figura 63. Definição da tabela Language</i>	68
<i>Figura 64. Definição da tabela Picture</i>	68
<i>Figura 65. Definição da tabela Point</i>	69
<i>Figura 66. Definição da tabela Pont_Topic</i>	69

<i>Figura 67. Definição da tabela Points_Attributes.....</i>	69
<i>Figura 68. Definição da tabela Search_Profile.....</i>	70
<i>Figura 69. Definição da tabela Search_Profile_Topic</i>	71
<i>Figura 70. Definição da tabela Theme</i>	71
<i>Figura 71. Definição da tabela Topic</i>	71
<i>Figura 72. Definição da tabela Translation.....</i>	72
<i>Figura 73. Definição da tabela User.....</i>	72
<i>Figura 74. Estrutura do projecto AppTour Android</i>	73
<i>Figura 75. package com.apptour.app.....</i>	73
<i>Figura 76. package com.apptour.framework</i>	74
<i>Figura 77. Singleton na classe ApplicationValues.....</i>	74
<i>Figura 78. Diagrama de Actividades em Andoid.....</i>	76
<i>Figura 79. Assinatura e especificação do método Search do Web Service WCF</i>	77
<i>Figura 80. Aplicação AppTour- Pesquisa.....</i>	78
<i>Figura 81. Efectuar uma pesquisa no Web Portal</i>	79
<i>Figura 82. Exemplo de pesquisa através do Web Portal.....</i>	79
<i>Figura 83. Definição do Stored Procedure Search.....</i>	80
<i>Figura 84. Aplicação AppTour – Localização</i>	80
<i>Figura 85. Pedido de obtenção de localização no Google Chrome</i>	81
<i>Figura 86. Exemplo de um comentário de um ponto.....</i>	83
<i>Figura 87. Pedido AJAX, em jQuery, para inserir comentário</i>	84
<i>Figura 88. Assinatura do método InsertComment.....</i>	84
<i>Figura 89. Diagrama de Sequência da funcionalidade de inserir comentário.....</i>	85
<i>Figura 90. Classificação de um Ponto</i>	86
<i>Figura 91. Diagrama Sequência InsertVote</i>	87
<i>Figura 92. Botão de denunciar um comentário</i>	88
<i>Figura 93. Activar vista de perfis de pesquisa.....</i>	89
<i>Figura 94. Definir perfis de pesquisa em Android</i>	89
<i>Figura 95. Menu de Topo do Web Portal.....</i>	90
<i>Figura 96. Separador de Pesquisas no Perfil de Utilizador</i>	91
<i>Figura 97. Método Create do SearchProfileController.....</i>	91
<i>Figura 98. Eliminar um perfil de pesquisa no Android</i>	93
<i>Figura 99. Partilha de Pontos em redes sociais</i>	94
<i>Figura 100. Exemplo dos scripts que permitem a partilha em redes sociais</i>	95

<i>Figura 101. Exemplo de interface de inserção de novo ponto.</i>	96
<i>Figura 102. Código do pedido POST de inserção de um ponto.</i>	97
<i>Figura 103. Formulário de edição de um ponto.</i>	98
<i>Figura 104. Página de Atributos do ponto ISEP.</i>	99
<i>Figura 105. Formulário de Atributos.</i>	99
<i>Figura 106. Método InsertAttribute do controlador AttributeController</i>	100
<i>Figura 107. Diagrama de Sequência InsertAttribute</i>	101
<i>Figura 108. Membership da Microsoft</i>	102
<i>Figura 109. Gestão de Roles no Membership</i>	102
<i>Figura 110. Bloquear e Desbloquear Utilizadores no Membership</i>	103
<i>Figura 111. Windows Task Scheduler</i>	104
<i>Figura 112. Estrutura do projecto AgentsService</i>	105
<i>Figura 113. Início do processo de importação de dados pelos Agentes</i>	106
<i>Figura 114. Processo de Criação de Agentes</i>	107
<i>Figura 115. Esquema do processo de gravação de informação recolhida pelos Agentes.</i>	109
<i>Figura 116. Fluxograma de operações do AgentOperationService</i>	112
<i>Figura 117. Entidade Language com os ficheiros de resources</i>	113
<i>Figura 118. Conteúdo do resource Language em Português</i>	114
<i>Figura 119. Conteúdo do resource Language em Inglês</i>	114
<i>Figura 120. Utilização de resources numa View</i>	114
<i>Figura 121. Resources em Android</i>	115
<i>Figura 122. Exemplo de traduções no Android em Inglês</i>	116
<i>Figura 123. Exemplo de traduções no Android em Português</i>	116
<i>Figura 124. Estrutura da tabela Translations</i>	117
<i>Figura 125. Leitura de cidades com conteúdos traduzidos</i>	117
<i>Figura 126. Pasta de projectos de teste</i>	118

Índice de Tabelas

<i>Tabela 1 - Reunião de Kick-Off</i>	1
<i>Tabela 2 - Apresentação e discussão de tecnologias a utilizar.</i>	1
<i>Tabela 3 - Apresentação de Arquitectura a utilizar.</i>	2
<i>Tabela 4 - Ponto de Situação</i>	2
<i>Tabela 5. Apresentação da solução</i>	2
<i>Tabela 6 - Ponto de situação do relatório</i>	3
<i>Tabela 7. Objectivos cumpridos em requisitos</i>	120
<i>Tabela 8. Objectivos cumpridos em Casos de uso</i>	122

Notação e Glossário

AJAX	<i>Asynchronous JavaScript and XML.</i>
	Tecnologia que, entre outros aspectos, permite adicionar interactividade a páginas Web.
API	<i>Application Programming Interface.</i>
	É um conjunto de especificações e “regras” (código) que programas de <i>software</i> seguem, de forma a comunicar entre si. Serve de interface entre diversos programas e facilita a sua interacção.
ASP.NET	Plataforma da Microsoft para desenvolvimento de aplicações Web.
Backend	Termo que se refere ao painel administrativo de uma aplicação.
BLL	<i>Business Logic Layer.</i>
Browser	Aplicação que permite a visualização de páginas web.
C#	C Sharp. Linguagem de Programação.
DAL	<i>Data Access Layer.</i>
DLL	<i>Dynamic-link library</i> ou biblioteca de ligação dinâmica. Implementação das comuns bibliotecas nos sistemas Microsoft e OS/2. Estas podem conter no seu interior extensões, recursos, código e dados partilháveis por diversas aplicações.
GET	Método HTTP.
IDE	<i>Integrated Development Environment.</i> Ambiente integrado para desenvolvimento de software.
Jquery	Biblioteca <i>JavaScript</i> desenvolvida para simplificar os scripts do lado do cliente.
Json	<i>JavaScript Object Notation.</i> Formato para comunicação de dados entre aplicações.
MVC	<i>Model-View-Controller.</i> É um tipo de arquitectura de <i>software</i> , considerada actualmente um padrão arquitectural. Este padrão isola a lógica de domínio do interface ao utilizador permitindo desenvolvimento, testes e manutenção isoladamente.
ORM	<i>Object Relational Mapping.</i> Técnica de programação orientada a objectos que permite converter dados entre dois sistemas de paradigmas incompatíveis.

PESTI	Projecto-Estágio.
Plugin	Conjunto de bibliotecas que incrementam as funcionalidades das bibliotecas já existentes.
POST	Método HTTP.
REST	<p><i>Representational State Transfer.</i></p> <p>Arquitectura para distribuição de hipermédia, uma das utilizações é a implementação de <i>Web Services</i>.</p>
SDK	<p><i>Software Development Kit.</i></p> <p>Tipicamente, é um conjunto de ferramentas de desenvolvimento que ajudam à criação de aplicações para um <i>software</i> específico.</p>
SGDB	<p>Sistema de Gestão de Base de Dados</p> <p>Aplicação ou conjunto de aplicações responsáveis pela gestão de uma base de dados e por disponibilizar funcionalidades que permitem que interaja com elas.</p>
Site ou Website	Sítio electrónico na Internet composto por uma ou mais páginas acessíveis normalmente através de um navegador.
Smartphone	Telefone móvel de alta tecnologia que combina as funções de um assistente digital (PDA) e de um telefone móvel regular.
SOAP	<p><i>Simple Object Access Protocol</i></p> <p>Protocolo para troca de informação em <i>Web Services</i>.</p>
SVN	<i>Subversion.</i> É um sistema de controlo de versão, desenhado para ser o substituto do predecessor CVS.
UML	<i>Unified Modeling Language.</i>
W3C	<i>World Wide Web Consortium</i>
WBS	<p><i>Work breakdown structure</i></p> <p>Ferramenta de decomposição do trabalho de um projecto em partes executáveis.</p>
Web Service	Componente de software que permite a outras aplicações enviar e receber dados.
XML	<p><i>Extensible Markup Language.</i></p> <p>Linguagem de marcação capaz de descrever tipos de dados, que define um conjunto específico de regras, para que seja possível serem interpretadas por máquinas e sistemas.</p>

1 Introdução

No presente capítulo é feita uma abordagem global do projecto, entendida como o conjunto constituído pela descrição do problema a solucionar, pelos objectivos, planeamento, ferramentas e tecnologias utilizadas.

1.1 Enquadramento

Este relatório visa documentar o processo de desenvolvimento do projecto realizado no âmbito da unidade curricular de Projecto/Estágio da Licenciatura em Engenharia Informática do Instituto Superior de Engenharia do Porto. A disciplina, a última a ser realizada no curso, propõe a elaboração de um projecto de aplicação informática com vista à satisfação de uma necessidade ou à melhoria de determinadas funcionalidades existentes e possibilita a aplicação da experiência adquirida nas unidades curriculares do curso e naturalmente, o contacto com novas tecnologias.

A *AppTour* enquadra-se no contexto de *Geo Applications* que se referem a aplicações, normalmente Web, que aproveitam as potencialidades oferecidas por diversas *APIs* geográficas, para desta forma poder apresentar informação relevante e localizada geograficamente.



Existem diversas outras aplicações no mercado que aproveitam essas mesmas potencialidades, porém cada uma delas apresenta-se com um propósito bem definido. Desde aplicações sociais como o *FourSquare*¹ a aplicações cujo objectivo é específico a determinadas áreas de negócio, várias são actualmente as aplicações que visam fornecer informação relativa a um ou vários locais específicos.

Figura 1. Aplicação Foursquare

¹ <https://foursquare.com>

1.2 Objectivos

Na aplicação proposta, o conceito a aplicar passa pela disponibilização de informação que auxilie um turista numa visita a uma cidade desconhecida. Dessa forma, actuando como um guia turístico, é possível saber, literalmente na ponta dos dedos, informação pertinente sobre locais disponíveis e interessantes, sem que para isso seja necessário a consulta de variados sítios na net, tantos quanto os possíveis temas de interesse que possam existir.

Esta aplicação apresenta-se como uma plataforma web que permite uma visualização e manutenção de informação distinta, agrupada em diversos temas e identificada como pontos geográficos ou eventos temporais que ocorrem nesses mesmos pontos.

Paralelamente a essa plataforma, existe uma segunda plataforma dedicada a aparelhos móveis, normalmente designados de *smartphones*, que visa oferecer uma maior mobilidade assim como aproveitar as potencialidades oferecidas por esses aparelhos. A vantagem em utilizar uma aplicação nativa consiste em oferecer um interface mais simplificado e directo às funcionalidades a utilizar, assim como maximizar o desempenho e a experiência do utilizador.

A informação poderá ser mantida por utilizadores regulares da aplicação assim como alimentada através de diversas fontes de informação que possa ser encontradas pela internet.

Outro aspecto a ter em conta centra-se num processo alternativo de recolha e uniformização da informação a disponibilizar. Embora todos os aspectos da aplicação possam e devam ser mantidos de uma forma directa pelos utilizadores, tal solução por si só, peca por insuficiente, já que dessa forma, seria provavelmente necessário muito tempo até que fosse recolhida um mínimo de informação e se tornasse uma solução viável.

Assim sendo e tendo em conta que existem muitos portais na internet que já dispõem dessa mesma informação, embora que de uma forma distinta, será necessário arranjar um meio de recolha e uniformização dessa mesma informação.

Isso implica aproveitar de alguma forma as diversas *APIs* disponibilizadas para o efeito, reaproveitando essa mesma informação através de um processo que se pretende automático e autónomo.

1.3 Contributos deste Trabalho

A pesquisa de informação relativamente a pontos específicos de um local ou cidade tem estado em crescimento, o que motiva tanto à procura dessa mesma informação, como ao desenvolvimento de soluções diversas com vista a satisfazer essa mesma demanda. Vários serviços são disponibilizados ao utilizador comum que possuem muita dessa informação, tais como o *Google Places*², entre outros, onde são apresentados vários pontos agrupados sob diversos temas.

Os principais problemas destas soluções prendem-se com o facto de muitos dos pontos considerados interessantes por habitantes locais tais como monumentos históricos, locais de interesse ou mesmo zonas pitorescas ou características não poderem ser mantidos directamente por utilizadores comuns, que muitas vezes são aqueles que detêm o melhor conhecimento popular sobre esses mesmos locais.

Outra situação menos agradável é o facto de que muitos portais que fornecem essa mesma informação, apenas abrangem alguns temas específicos relativos ao tema desse mesmo portal, o que implica uma pesquisa específica em cada um desses mesmos portais para cada tema, assim como um prévio conhecimento da sua existência.

Assim sendo, a georreferenciação de variados temas de uma forma homogénea e de manutenção directa pelos utilizadores é ainda uma situação que não se encontra prevista. Este projecto de estágio apresenta-se como uma possível solução para resolver esse problema, actuando como um aglutinador de diversas fontes dispersas e um homogeneizador da diversa informação encontrada sobre diversos temas considerados interessantes para um estranho de visita a uma cidade e que podem ser livremente mantidos por habitantes locais com vista a anunciar aspectos que considerem relevantes.

² <http://www.google.com/places/>

1.4 Processo de Desenvolvimento

O processo de desenvolvimento utilizado para a implementação da solução final foi um processo iterativo. Muito devido ao desconhecimento sobre a matéria a abordar, mas também para dessa forma poder ir adaptando esse mesmo desenvolvimento às necessidades em questão, o projecto foi sendo desenvolvido através da divisão em subtarefas, que foram depois divididas pelos dois elementos da equipa.

Foram também implementadas várias reuniões entre os elementos, assim como entre estes e o coordenador de projecto, para cruzamento de informação e ponto de situação do desenvolvimento efectuado até então.

1.4.1 Planeamento de projeto

Na fase inicial deste projecto foi realizado um planeamento de forma a proporcionar um melhor conhecimento das funcionalidades a implementar, assim como do tempo máximo disponível para cada uma das mesmas. Foram delineadas cinco fases distintas para a realização deste projecto:

- Análise de Requisitos
 - Definição de Casos de Uso
 - Definição de Funcionalidades
- Design da Solução
 - Arquitectura
 - Definição de Modelo de Dados
 - Aplicações e Componentes
- Desenvolvimento e Documentação
 - Web Portal
 - Agentes
 - Aplicação Android
 - Diagramas UML
 - Testes
- Elaboração do Relatório

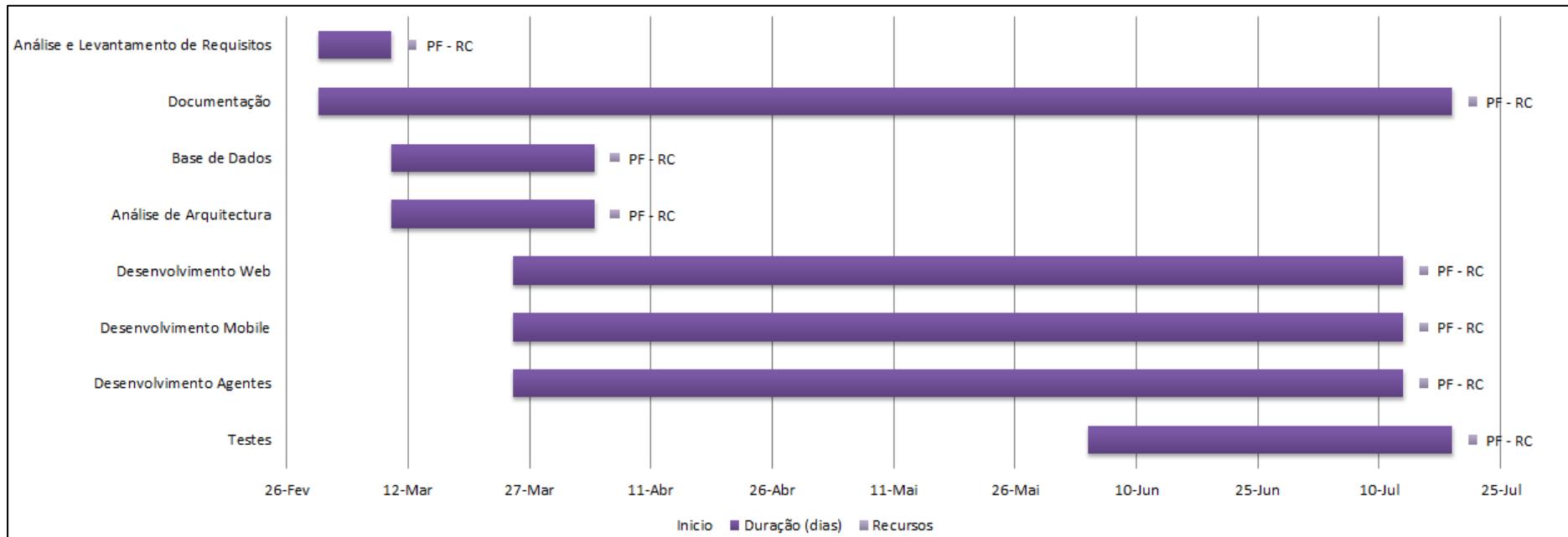


Figura 2. Planeamento do Projecto

1.4.2 Reuniões de acompanhamento

Durante a realização do projecto houve várias reuniões que permitiram a discussão de temas relativos à concepção e desenvolvimento da solução a implementar com o Orientador de Estágio, Professor Paulo Gandra de Sousa.

Segue um resumo dos pontos discutidos e respectivas reuniões

Tabela 1 - Reunião de Kick-Off

Reunião Kick-Off	
Data Realização	13 Março, 2012
Intervenientes	Professor Paulo Gandra de Sousa Pedro Ferreira Ricardo Carneiro
Local	Instituto Superior de Engenharia do Porto
Temas Discutidos	Apresentação do projecto inicial. Discussão de conceitos, ideias e complexidade de solução.

Tabela 2 - Apresentação e discussão de tecnologias a utilizar.

Reunião Apresentação/Discussão de Tecnologias	
Data Realização	04 Abril, 2012
Intervenientes	Professor Paulo Gandra de Sousa Pedro Ferreira Ricardo Carneiro
Local	Instituto Superior de Engenharia do Porto
Temas Discutidos	Discussão de benefícios/desvantagens do uso de tecnologias Microsoft; Escolha da tecnologia ASP .NET MVC da Microsoft. Discussão da viabilidade do uso da plataforma dedicada Android.

Tabela 3 - Apresentação de Arquitectura a utilizar.

Reunião Apresentação da Arquitectura da utilizar	
Data Realização	12 Abril, 2012
Intervenientes	Professor Paulo Gandra de Sousa Pedro Ferreira Ricardo Carneiro
Local	Instituto Superior de Engenharia do Porto
Temas Discutidos	Apresentação do Diagrama de Contexto. Apresentação da WBS e planeamento de desenvolvimento Apresentação da primeira abordagem da arquitectura a utilizar na solução. Reformulação da Arquitectura apresentada.

Tabela 4 - Ponto de Situação

Reunião Ponto de Situação	
Data Realização	5 Junho, 2012
Intervenientes	Professor Paulo Gandra de Sousa Pedro Ferreira Ricardo Carneiro
Local	Instituto Superior de Engenharia do Porto
Temas Discutidos	Ponto de situação do desenvolvimento efectuado à data. Demonstração de funcionamento dos componentes e a sua especificação de desenvolvimento. Discussão de desenvolvimentos futuros.

Tabela 5. Apresentação da solução

Reunião Apresentação da Solução	
Data Realização	14 Junho, 2012
Intervenientes	Professor Paulo Gandra de Sousa Pedro Ferreira Ricardo Carneiro
Local	Instituto Superior de Engenharia do Porto
Temas Discutidos	Apresentação da estrutura a elaborar do relatório

Tabela 6 - Ponto de situação do relatório

Reunião Ponto de situação do Relatório	
Data Realização	20 Junho, 2012
Intervenientes	Professor Paulo Gandra de Sousa Pedro Ferreira Ricardo Carneiro
Local	Instituto Superior de Engenharia do Porto
Temas Discutidos	Ponto de situação do relatório; alterações a efectuar para novos desenvolvimentos

1.5 Linguagens e Tecnologias

Na realização deste trabalho foi necessário recorrer a diversas tecnologias e linguagens que possibilitessem um bom desenvolvimento do trabalho a realizar.

Estas foram escolhidas com vista à facilidade de uso e rapidez de desenvolvimento, relativamente ao ambiente Windows, assim como o custo inexistente inerente ao desenvolvimento móvel para *Android*. Outro dos motivos que levaram à escolha destas mesmas tecnologias foi a vontade de especialização nas mesmas. Seguidamente apresenta-se uma lista dessas tecnologias escolhidas:

- Sistema de Gestão de Base de Dados: Microsoft SQL Server 2008;
- Linguagens de programação: C#.NET, Java;
- Plataforma: .NET 4.0, Android SDK;
- Ambiente de desenvolvimento: Microsoft Visual Studio 2010, Eclipse;
- Sistemas Operativos: Windows 7, Android 2.2 (Froyo);
- Desenvolvimento de páginas Web: ASP.NET MVC 3.0;
- Outras tecnologias: HTML5, CSS 3, JavaScript, JSON, jQuery;
- Controlo de versões SVN sobre todos os projectos .NET e Android

Android SDK

O *Android* é um Sistema Operativo do *Google*, que tem como base um núcleo *Linux*, divergido da versão do *kernel* original.

Existem vários modos de desenvolvimento de aplicações para a plataforma *Android*, porém apenas foi considerado para o projecto o ambiente de desenvolvimento *Android SDK*³ que corresponde a um conjunto de ferramentas disponibilizadas pela Google para o desenvolvimento de aplicações baseadas em linguagem Java e que são depois executadas na máquina virtual nativa do *Android* (*Dalvik*).

Entre outras, as ferramentas disponibilizadas pelo *plugin* são:

³ <http://developer.android.com/tools/sdk/tools-notes.html>

- *ADT plugin for Eclipse*⁴: *Android Developer Tools* utilizado no ambiente integrado de desenvolvimento Eclipse para desenvolvimento de aplicações *Android* em *Java*. Este *plugin* permite ao Eclipse a integração de diversas vistas que permitem o desenvolvimento e depuramento das aplicações *Android*;
- *AVD Manager*⁵: *Android Virtual Device manager*. Permite a criação e gestão de dispositivos virtuais *Android* para teste e simulação em máquinas de desenvolvimento;
- *ADB*⁶: *Android Debug Bridge*. É a ferramenta de ligação ao *AVD* assim como a dispositivos *Android* externos que permite o *debug* das aplicações;
- *DDMS*⁷: *Dalvik Debug Monitoring Service*. É uma ferramenta de diagnóstico e depuramento da máquina virtual *Dalvik*, onde correm as aplicações desenvolvidas para *Android* através da linguagem *Java*. Permite funcionalidades como a monitorização das *threads*, memória e restantes componentes que fazem parte do dispositivo *Android* onde está a correr a aplicação.

Existem várias versões do sistema operativo *Android* que podem ser utilizadas, dependendo das funcionalidades pretendidas. A escolha deste projecto recaiu sobre o *Android 2.2 Froyo*, uma vez que existem casos de aumento de performance na ordem dos 450% para versões anteriores. Foi também implementado o *addon* do *Google APIs* de forma a poder aproveitar as aplicações disponibilizadas pela *Google*, mais nomeadamente o *Google Maps*.

Entity Framework

Esta *framework* facilita o acesso aos dados e abstrai o programador das regras e validações da base de dados, garantido a sua integridade e exactidão dos dados. Para além dessas garantias de integridade, a *framework* possibilita o uso de uma série de recursos que aumentam a produtividade no desenvolvimento de aplicações persistentes.

⁴ <http://developer.android.com/tools/sdk/eclipse-adt.html>

⁵ <http://developer.android.com/tools/devices/managing-avds.html>

⁶ <http://developer.android.com/tools/help/adb.html>

⁷ <http://developer.android.com/tools/debugging/ddms.html>

*ADO .NET Entity Framework*⁸ utiliza uma variante da linguagem *SQL*, denominada *Entity SQL*, que é responsável por codificar consultas declarativas e actualizações sobre entidades e as suas relações, no nível conceitual. É distinto do *SQL* na medida em que não tem construções explícitas para junções porque o *Entity Data Model (EDM)* foi desenhado para abstrair o particionamento dos dados nas tabelas.

ASP .NET MVC 3

O padrão de arquitectura *Model-View-Controller (MVC)* separa a aplicação em três grandes componentes: O modelo, a vista e o controlador.

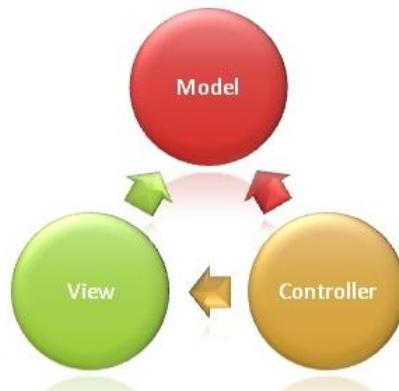


Figura 3. Padrão MVC

Modelo: os modelos são objectos usados para definir e gerir o domínio da informação. São uma representação detalhada da informação utilizada pela aplicação. Podem implementar lógica de negócio, adicionando dessa forma valor semântico aos dados.

Vistas: as vistas são componentes que mostram a interface da aplicação ao utilizador (*UI*). Normalmente, essa *UI* é criada a partir dos dados do modelo. Um exemplo é o modo de edição de um tabela de pontos, onde mostra caixas de texto, *drop-down lists* e *checkbox*, com base no estado um objecto *PointModel*.

Controladores: os controladores são os componentes responsáveis pela interacção do utilizador, trabalham o modelo e finalmente utilizam uma vista para

⁸ <http://msdn.microsoft.com/en-us/library/bb399572.aspx>

mostrar ao utilizador. Numa aplicação *MVC* a vista apenas exibe informações, o controlador manipula e responde às interacções do utilizador.

A *Framework ASP .NET MVC*⁹ oferece uma alternativa ao *ASP .NET Web Forms* para criar aplicações *Web*, baseados em *MVC*. É uma *framework* leve, com uma estrutura de apresentação altamente testável, totalmente integrado com os recursos existentes do *ASP.NET*, como páginas mestre e associações baseada em autenticação.

A versão 3 do *ASP .NET MVC* foi escolhida devida ao facto de ser a última versão estável e suportada pela Microsoft e consegue garantir a funcionalidade de todos as características da *framework .NET 4.0*. Inicialmente tentamos a versão *ASP .NET 4*, juntamente com a versão 4.5 da *plataforma .NET*, mas devido à instabilidade e incompatibilidade dos sistemas integrantes, foi sensato optar pela versão 3.

WCF

Windows Communication Foundation (WCF) é um modelo de programação unificado e ambiente de execução (*framework*) criado pela *Microsoft* cujo principal objectivo é a construção de aplicações baseadas em serviços, também conhecidas como *SOA (Service Oriented Applications)*.

O *WCF* suporta a interoperabilidade com aplicações *WCF* em execução num mesmo computador Windows ou em execução em máquinas Windows diferentes ou ainda serviços *Web* padrão construídos em plataformas como *Java* ou outras.

O *WCF* suporta comunicações em protocolos *SOAP*, como também dados *XML* não envoltos em *SOAP*. Pode ainda ser usado para suportar outros formatos como *JSON*, o que o tornam flexível para as necessidades actuais e futuras.

De uma maneira geral a comunicação entre serviços *WCF* funciona da seguinte maneira: Um cliente *WCF* conecta-se a um serviço *WCF* através de um *endpoint*. Cada serviço expõe seu contrato por um ou mais *endpoints*. Um ponto de extremidade (*endpoint*) tem um endereço (*URL* que especifica onde o ponto final

⁹ <http://www.asp.net/mvc/mvc3>

pode ser acedido) e propriedades de ligação que especifica como os dados serão transferidos.

1.6 Organização do relatório

O presente relatório está escrito em Português convencional, não contemplando o novo Acordo Ortográfico. Está também organizado em cinco capítulos, cada um deles com os conteúdos que a seguir se apresentam.

No primeiro capítulo – Introdução – é feita uma breve apresentação do projecto, do seu enquadramento e da sua finalidade. São, ainda, descritas as reuniões de acompanhamento que existiram, o planeamento efectuado para o projecto assim como as tecnologias utilizadas.

O segundo capítulo – Análise – apresenta os aspectos e conceitos relacionados com o trabalho realizado, principais ideias consideradas e como este foi inicialmente idealizado, descreve os requisitos identificados, assim como a especificação dos casos de uso a implementar e apresenta uma análise global à solução pretendida.

No terceiro capítulo – Implementação e Desenvolvimento – é descrito a forma de desenvolvimento e metodologias adoptadas para a solução final. São referenciadas as resoluções dos casos de uso, acompanhados com a documentação específica. Por fim, são descritos como foram executados e implementados testes à solução criada.

O quarto e último capítulo – Conclusões – apresentam os objectivos realizados e as limitações existentes, assim como possíveis desenvolvimentos no futuro. São ainda referidos outros trabalhos realizados durante o estágio e é produzida uma apreciação final por parte dos estagiários intervenientes no projecto.

2 Análise

Neste capítulo serão abordados vários pontos considerados essenciais para que se possa considerar que a aplicação cumpriu o seu objectivo. Para além daqueles que são os requisitos básicos necessários, são previstos alguns requisitos complementares para que se possa de alguma forma dar um valor acrescido a essa mesma solução. Por fim tentamos também estabelecer alguns requisitos não funcionais que permitam que a experiência dos futuros utilizadores possa ser valorizada e possam de alguma forma complementar e favorecer a opinião final da solução proposta.

Embora o projecto seja puramente académico, já que não existe nenhuma proposta de lhe dar um seguimento comercial, deve existir um cuidado especial em tornar o mesmo escalável e permitir de alguma forma uma maior especialização deste, como se de um projecto comercial se tratasse.

2.1 Termos e Conceitos

Agentes

Os Agentes são componentes da solução *AppTour* cuja função consiste na importação de dados através de *APIs* externas. Estes agentes são controlados pelo projecto *AppTour.Agents.Services*, também falado como *AgentsServices* que disponibiliza os modelos para as *APIs*, os serviços e garante o fluxo de controlo e de dados.

Pontos

O conceito de ponto implica uma localização, imóvel, ao qual está inherente um negócio, ideia ou hábito. Implicam uma coordenada de forma a poderem ser localizados e apresentados num mapa e um nome que os possam identificar de alguma forma. Também podem ter associadas outras informações, como por exemplo, descrições do negócio ou conceito desse mesmo ponto, imagens que permitam uma melhor identificação, horários de funcionamento, ligações para sítios

na internet que permitam uma maior informação sobre o ponto em questão ou mesmo uma maior interacção, entre outras que seja consideradas pertinentes. Associados aos pontos podem também estar Eventos.

Eventos

Eventos são acções ou acontecimentos que ocorrem num determinado espaço de tempo e em um ou mais Pontos. Cada evento tem assim associado um nome, uma duração e um ponto mínimo obrigatório. Assim sendo ao efectuar uma pesquisa sobre um determinado ponto é também possível saber quais os eventos que lhe estão atribuídos ou que decorrerão nesse mesmo ponto e em que dias.

Perfis de Pesquisa

Parte principal da solução a implementar baseia-se na pesquisa de pontos e/ou eventos. Todavia os métodos de pesquisa tradicionais (por texto) são insuficientes para uma aplicação geográfica. Assim sendo deve ser possível definir perfis de pesquisa, criados e configurados por cada utilizador, cuja principal função é guardar e reutilizar variadas pesquisas que sejam consideradas frutíferas. Estes perfis de pesquisa poderão ser parametrizados com distância (em quilómetros) entre pontos, assim como dias em que podem ocorrer futuros eventos, pesquisar em determinados tópicos e filtrar por palavras-chave, conforme se pode ver na Figura 4.

Nome	Distância dos Pontos	Tempo de Eventos (dias)	Critério de Pesquisa	Tópicos	Activo	
Pesquisa	32	2		0		

Figura 4. Exemplo de Perfil de Pesquisa

Web Portal

O *Web Portal* é uma aplicação pertencente ao *AppTour* que complementa o sistema na sua componente *web*. Está alojado numa máquina virtual no Departamento de Engenharia Informática do ISEP¹⁰.

¹⁰ <http://apptour.dei.isep.ipp.pt/AppTour>

2.2 Solução Pretendida

O projecto *AppTour* é um projecto que visa a georreferenciação de diversos pontos e eventos, para desta forma poder fornecer informação diversa de uma forma acessível. Assim sendo alguns requisitos são necessários para que se cumpram estes objectivos. Outros requisitos há que embora não sejam absolutamente necessários, gostaríamos de ver considerados para dessa forma poder fornecer valor acrescentado à solução. Seguidamente são apresentados esses mesmos requisitos.

2.2.1 Requisitos Básicos

Os requisitos básicos são aqueles que são considerados os mínimos da aplicação. Por forma a poder se considerar uma solução viável, todos os pontos aqui descritos deverão ser considerados.

Resposta em tempo útil

Como a solução final deverá fornecer informação de uma forma rápida e homogénea para o utilizador, existe a necessidade de haver um repositório central onde essa mesma informação é previamente guardada para mais tarde poder ser disponibilizada. Dessa forma agiliza-se o processo de consulta das diversas fontes assim como de moldagem da informação.

Uma solução possível passaria pela criação de um repositório central que aglutinasse essa mesma informação.

Web Portal

A apresentação da solução final através de um *Web Portal* é quase uma obrigatoriedade nos dias que correm, tanto pela grande exposição que permite, como pela facilidade de manuseamento. Assim espera-se que a solução proporcione um interface web que permita a realização de todos os casos de uso previstos para o utilizador.

Android

Uma plataforma num formato móvel é necessária dada a mobilidade inherente num conceito de turismo. A aplicação Android deverá ter algumas funcionalidades iniciais mínimas. Embora se possam prever funcionalidades adicionais, numa primeira fase, o mínimo deverá estar assegurado, para que esta possa ser considerada viável. São elas:

- Login de um utilizador já existente na aplicação
- Registo de um novo utilizador
- Manutenção de perfis de pesquisa
- Efectuar pesquisas
- Visualização de detalhes de um ponto
- Visualização de detalhes de um evento

Agentes

O processo de preenchimento automático da informação disponível deverá ser efectuado por Agentes. Agentes são programas que usam informação disponibilizada em diversas *APIs* e a transformam de uma forma homogénea para que possa ser incluída no repositório central. Estes devem ser autónomos na medida em que deverão dispensar a intervenção de pessoas para que possam ser executados. Deverão também permitir uma fácil escalabilidade para novas *APIs* que se apresentem no mercado.

Pontos

A solução final passa pela apresentação de informação relativa a determinados pontos geográficos. Algumas regras relativas aos pontos devem ser cumpridas:

O sistema deve permitir a criação de Pontos pelo utilizador corrente, assim associar informação distinta a esse mesmo Ponto.

Os Pontos devem ter associados um nome e uma coordenada para poderem ser considerados credíveis.

O sistema deve permitir a entrada da localização de um Ponto através do fornecimento de uma latitude e longitude. Alternativamente ou em complemento, o Sistema deve permitir a informação da localização ser fornecida através de um mapa.

O sistema deve permitir a edição da informação desse mesmo Ponto, mas apenas por utilizadores com privilégios para isso.

Essa informação relativamente a pontos deverá ser mantida tanto pelos utilizadores, como também através de um processo autónomo.

Eventos

Eventos são acontecimentos que decorrem num determinado local durante um determinado espaço de tempo.

Como tal o sistema deve permitir fornecer a um Evento um nome, um ou mais pontos onde este ocorre e uma data de início assim como uma de fim.

Deve ser permitido criar um Evento por um utilizador corrente, desde que fornecida a informação supra citada.

Deve ser possível editar a informação relativa a um Evento já criado, por utilizadores com permissão para isso.

O sistema deve permitir criar essa informação relativa a um Evento tanto pelos utilizadores, como através de um processo automático.

Temas / Tópicos

Tanto os pontos como os eventos deverão estar associados a tópicos, para assim permitir uma melhor especificação dos aspectos base inerentes. Esses tópicos serão representados por ícones que os representarão de uma forma visual para desta forma serem mais facilmente identificáveis.

Os tópicos, por sua vez deverão estar agrupados em temas, para assim simplificarem as pesquisas a efectuar. Esses mesmos temas serão visualmente identificados por cores respectivas.

Uma descrição dos temas e/ou tópicos abordados é a que segue:

- Alojamento
 - Hotéis
 - Motéis
 - Residenciais
 - Pousadas
 - Estalagens
 - Albergaria
- Restauração
 - Restaurantes
 - Cafés
 - Snacks
 - Eventos Gastronómicos
- Transportes
 - Aviação
 - Comboios
 - Metro
 - Autocarros
 - Táxis
 - Aluguer Veículos
- Cultura
 - Museus
 - Exposições
 - Monumentos
 - Locais Interesse
- Entretenimento
 - Bares & Discotecas
 - Cinema
 - Concertos
 - Teatro
 - Espectáculos
- Desportos
 - Eventos Desportivos
 - Ginásios
 - Actividades Desportivas
- Saúde & Bem-estar
 - Spas
 - Termas
 - Estética e Cabeleireira
 - Serviços Públicos
- Serviços Saúde
 - Policia
 - Educação
 - Farmácias
 - Informações Turísticas
 - Câmbios

Multilingue

Como a solução final é destinada a turistas, esta deverá contemplar outras línguas que não o Inglês normal. Como tal o *Web Portal* assim como a aplicação *Android* deverão ser multilingues e permitir a escalabilidade para várias linguagens.

Partilha em sítios Sociais

Uma constante nos dias de hoje são os sítios sociais e as interacções que estes permitem. Assim sendo uma grande mais-valia para a aplicação seria a partilha de determinados pontos e eventos para as maiores e mais conhecidas dessas aplicações sociais como *Facebook*¹¹, *Google+*¹² e *Twitter*¹³.

Gestores conteúdo

Devido ao facto da informação poder ser fornecida por todos os utilizadores, existe uma necessidade de controlar essa mesma informação para desta forma evitar informação errónea ou mesmo repetida. Assim sendo existe a necessidade de haverem moderadores dessa mesma informação, para que em caso de necessidade, poderem efectuar as correcções necessárias ao bom funcionamento da aplicação. A esses utilizadores com privilégios especiais são apelidados de gestores de conteúdos.

Para além dessa principal função, os gestores de conteúdo podem também controlar outros aspectos secundários como a moderação de comentários e classificações aplicados a um ponto ou evento.

Perfis de pesquisa

A aplicação deverá permitir ao utilizador a manutenção de diversos perfis de pesquisa. Dessa forma poderão ser gravadas vários tipos de pesquisa que de uma forma ou de outra possam corresponder a pesquisas com resultados relevantes ou que abranjam temas pré-definidos. Num exemplo prático podemos considerar um utilizador que efectua uma determinada pesquisa que abranja lugares de divertimentos nocturno assim como táxis, para poder pesquisar num horário nocturno, ao passo que durante o dia usará um perfil que pesquisa eventos culturais, museus e transportes públicos.

¹¹ <http://www.facebook.com>

¹² <https://plus.google.com/>

¹³ <https://twitter.com/>

2.2.2 Requisitos Complementares

Como requisitos complementares entendem-se as funcionalidades que embora não sejam essenciais para o normal funcionamento da aplicação, possam de alguma forma ser implementados ou previstos, permitindo assim uma escalabilidade das funções desta. Alguns dessas funcionalidades podem ser as que seguem:

Multicanal

Dada a grande concorrência existente actualmente no mercado móvel, deveremos considerar que a solução final seja escalável para outras aplicações móveis. Assim devem ser evitadas as regras de negócio do lado desses interfaces, centralizando as regras de negócio num servidor para minimizar o consequente esforço de criação de aplicações para outras plataformas.

Horários

A inclusão de horários nos pontos, pode permitir uma melhor eficácia nas pesquisas efectuadas, já que pode permitir filtrar os pontos que se encontram em funcionamento e ignorar aqueles que se encontram fechados. Para o utilizador final isso resume-se numa melhor e mais eficaz pesquisa, já que podem ser descartados à partida aqueles que não possam garantir um serviço na altura em que este é pedido.

Deve no entanto ser considerado que esse mesmo horário pode divergir mediante dias da semana, ou mesmo por horários sazonais.

Atributos

Embora os pontos / eventos possuam uma estrutura de informação mínima que deve ser garantida, deverá haver a hipótese de implementar uma política de atributos que permitirá uma atribuição de diversos valores que sejam considerados pertinentes e que podem divergir em tipo de dados a apresentar.

Deverão ser contemplados dados que possam ser guardados em texto, datas, valores numéricos ou decimais, assim como valores booleanos (sim/não).

Calendarização de Eventos

Como os eventos são acontecimentos que ocorrem em determinados períodos temporais pode haver a necessidade de um utilizador poder querer ser informado de algum evento atempadamente, para desta forma poder tomar as precauções necessárias, tais como compra de bilhetes. Uma potencialidade a ter em conta seria então a calendarização desses mesmos eventos, através de aplicações já existentes como o Google calendar.

2.2.3 Requisitos não Funcionais

Requisitos não funcionais são aqueles que embora não apresentem uma funcionalidade inerente, contemplam situações que deverão ser previstas para uma melhor fluidez e interacção dos utilizadores com o sistema.

Assim sendo alguns pontos foram previstos que possam trazer mais-valias para a aplicação.

Usabilidade

A aplicação deve ter uma fácil usabilidade e interacção. Como muitos dos utilizadores da aplicação podem ser turistas e não ter um nível muito avançado de interacção com aparelhos informáticos, funcionalidades básicas e principais como efectuar uma pesquisa ou consultar os detalhes de um ponto devem ser o mais simples e intuitivo possível.

Segurança

Devem ser previstas regras mínimas de segurança para garantir a integridade da informação disponibilizada. Dessa forma aspectos como a edição não autorizada da informação de pontos e eventos, assim como acesso à informação relativa de cada utilizador através da manipulação de *URLs* deve ser prevista e validada.

Outro aspecto importante passa pelo controle de instruções indevidas sobre a base de dados conhecidos como injecção de comandos.

A nível do *Android* deve ser evitado guardar informação privada no telemóvel, como por exemplo palavras passe, para evitar o uso indevido destas em caso de perca ou extravio do telefone. No entanto não deve ser descurada a funcionalidade de login automático através deste dispositivo numa tentativa de agilização de processo.

Confiabilidade

O uso da aplicação através de um terminal *Android* deve ter uma série de condições asseguradas para que todo o processo possa decorrer sem problemas. Problemas como falta de rede, impossibilidade de obter localizações, ou erros de comunicação devem ser previamente verificados e o utilizador deve ser informado atempadamente, para desta forma, evitar a naveabilidade errónea e o uso do terminal sem conseguir obter a informação correcta.

Desempenho

Tal como referido previamente a solução final terá um componente cuja principal função é a recolha e aglutinação de informação para um repositório central. Essa recolha pode ser feita sobre várias *APIs* e deverá permitir uma escalabilidade ao número de fontes que possam ser consultadas. Atendendo a esses factores um processo de migração pode-se tornar num processo bastante moroso e com um processamento muito pesado.

Para tentar atenuar tal situação, deve-se tentar implementar políticas de melhoramento de desempenho nos Agentes para minimizar esse custo. Essas políticas passarão pelo paralelismo das tarefas de migração, assim como a detecção e melhoramento dos vários processos mais custosos.

Autonomia

Tal como referido no ponto anterior o processo de migração pode ser um processo penoso e demorado. A altura mais lógica de os efectuar deverá ser nas alturas de menor uso do sistema geral. Isso implica que esse processo deverá ser

capaz de iniciar autonomamente, sem intervenção de ninguém, bastando para isso, agendar processos de migração.

Os Agentes deverão ser serviços que possam ser agendados e que não requeiram intervenção externa. Devem também criar um sistema de registo das intervenções efectuadas para desta forma permitir a avaliação do seu desempenho numa altura futura.

2.3 Visão Geral

Neste capítulo são abordados os diversos componentes que constituem a solução final. Apresenta-se de uma maneira geral quais as estruturas adoptadas, suas principais funções e objectivos que estes se visam a cumprir.

Na figura seguinte, podemos verificar também os componentes num diagrama de contexto geral assim como as ligações entre eles.

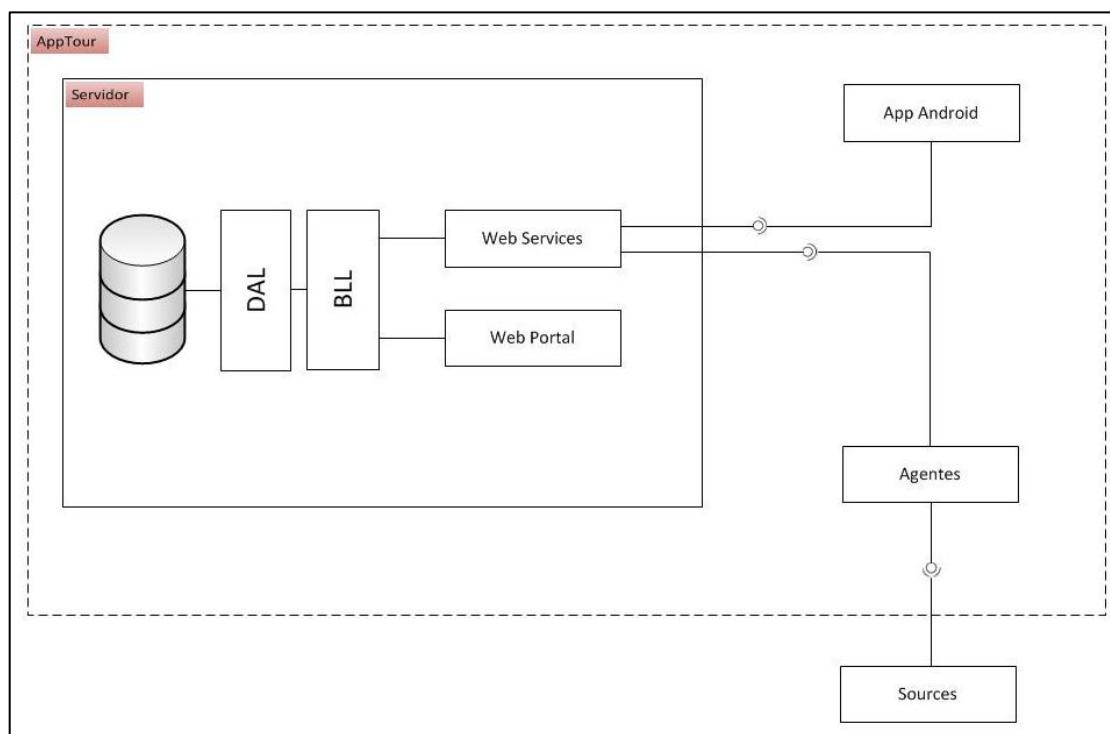


Figura 5. Diagrama de Contexto

2.3.1 *Repositório*

O repositório é um dos componentes principais da solução. Este componente tem como função guardar e disponibilizar os dados que são alimentados tanto pelos utilizadores, como pelos Agentes. Como repositório entende-se não só o sistema gestor de base de dados, como também toda a estrutura de acesso a essa mesma Base de Dados, também chamada de *DAL*.

A *DAL* serve para implementar uma estrutura em camadas que permita um isolamento sobre a Base de Dados, assim como uma maior segurança sobre o acesso aos dados, como algumas das políticas de segurança discutidas nos requisitos não funcionais.

2.3.2 *Lógica de Negócio*

O componente que controla toda a lógica de negócio é apelidado de *BLL*. Neste são controladas todas as regras de negócio através de validadores implementados nos modelos de negócio, assim como, e através da utilização de serviços, controlar e implementar regras de integridade entre os vários modelos, ao mesmo tempo que estes serviços servem de porta de entrada para pedidos externos.

Os pedidos externos à aplicação são efectuados através de dois outros componentes, os *Web Services* e o *Web Portal*, sendo que cada um tem um papel respectivo.

2.3.3 *Web Services*

Os *Web Services* são o componente que permite o uso dos serviços disponibilizados pela *BLL* da aplicação através da internet. Dessa forma conseguimos manter todas as tarefas relativas ao negócio de um modo centralizado e disponibilizar apenas ao público uma série de serviços finais que podem ser utilizados.

Utilizados pela aplicação *Android* para comunicações com o sistema central, os *Web Services* permitem assim uma interoperabilidade entre a aplicação móvel e o

sistema central, sem descurar o facto referido previamente, e deixando à aplicação móvel, um papel mais de apresentação de dados e interacção com o utilizador final.

Outro componente que visa a utilização dos *Web Services* é os Agentes, tanto para pedidos de dados, como para efectuar a persistência de nova informação, permitindo assim um reaproveitamento da estrutura já previamente montada.

2.3.4 Web Portal

O *Web Portal* é o principal e mais completo interface do sistema com os utilizadores. Neste são permitidas todas as funcionalidades implementadas e disponibilizadas pela aplicação. O *Web Portal* irá ser implementado através do padrão *MVC*, mais concretamente pelo *ASP.NET MVC* versão 3 da *Microsoft*. Tal escolha prende-se não só com a facilidade de implementação oferecida por este como também pela simplicidade e fiabilidade, o que permite uma construção de aplicações web com uma separação distinta entre a lógica de negócio e a apresentação dos dados.

Uma das particularidades do *Web Portal* na implementação do *MVC* será a reutilização dos modelos de negócio disponibilizados pelo sistema, nos *model* utilizados pelo próprio padrão, permitindo dessa forma uma maior consistência entre os dados e facilidade de implementação de alterações futuras, já que qualquer alteração sobre os modelos da arquitectura, serão imediatamente visualizados no *Web Portal*.

Outras ferramentas são disponibilizadas e utilizadas no *Web Portal* com vista à rapidez de implementação e robustez do portal como validadores, linguagens de Mapeamento objecto-relacional e *lambda-expressions*.

2.3.5 App Android

A aplicação para *Android* é a interface móvel dos utilizadores com a aplicação. Criada com vista a um sistema *Android* para aproveitamento das diversas funcionalidades oferecidas pelos *smartphones*, assim como pela mobilidade e simplicidade de interacção oferecida por estes.

A aplicação *Android* funcionará mais como um interface móvel para o sistema, já que usará os *Web Services* para interagir com este, delegando assim todas as regras de negócio assim como responsabilidades para o sistema central.

Não são guardados nenhum tipo de dados locais, para além de informações de sessão, o que implica que necessita de uma permanente ligação à internet em todo o processo de utilização das diversas funcionalidades.

Embora limitada às funcionalidades principais, tais como registo e autenticação de utilizadores, assim como manutenção de perfis de pesquisa e disponibilização dos resultados das pesquisas, a expansão para a total disponibilidade do conjunto completo de serviços disponibilizado pela aplicação seria uma mais-valia, que embora fosse uma séria opção a considerar, não se encontra no âmbito deste projecto, dado o pouco tempo disponível para a implementação da solução.

2.3.6 Agentes

Os Agentes são o componente que provavelmente passará mais despercebido ao utilizador comum da aplicação. Já que não oferece qualquer interface com estes e tem como função apenas a recolha de informação em diversas fontes na internet e correspondente persistência dos dados no repositório central. Pese embora esse factor, estes componentes têm um papel critico na credibilidade e aceitação da solução final, já que permitem ter um sistema funcional e de uso imediato, assim como constantes melhoramentos a nível da informação prestada.

Os Agentes não serão disponibilizados aos utilizadores comuns, pelo que não são necessários cuidados especiais, nem na apresentação nem a nível de segurança. Deve no entanto haver um cuidado especial de permitir uma grande escalabilidade

destes mesmos componentes dada a volatilidade das fontes usadas e constante aparecimento de novas soluções.

Os Agentes deverão ser implementados sob a forma de uma aplicação em consola ou mesmo em *WinForms*. Seja qual o modo escolhido para a implementação dos Agentes, estes deverão fornecer de uma forma visual ou através de ficheiros de log, informação sobre o processo de migração, assim como de dificuldades encontradas, erros detectados ou simplesmente casos de migração com sucesso. No caso de implementação em consola o registo de actividade poderá ser guardado em ficheiros de texto, ao passo que, no caso de *WinForms*, este registo poderá apenas ser mostrado numa simples caixa de texto.

Devido ao laborioso processo que implica uma migração de dados, deverá ser possível aos Agentes retomar um processo a partir de um ponto que tenha sido suspenso anteriormente. Dessa forma será possível efectuar migrações faseadas o que permite delegar esses processos para horas de menor uso do sistema.

Outra particularidade a ter em conta seria a implementação das funcionalidades de processamento em paralelo disponibilizado pela *framework .NET* potenciando assim um melhor aproveitamento dessa mesma “janela” de tempo disponível.

2.4 Modelo de Domínio

Depois de analisadas todas as entidades envolvidas e as suas dependências, foi desenhado o modelo de domínio, que se pode ver na Figura 6.

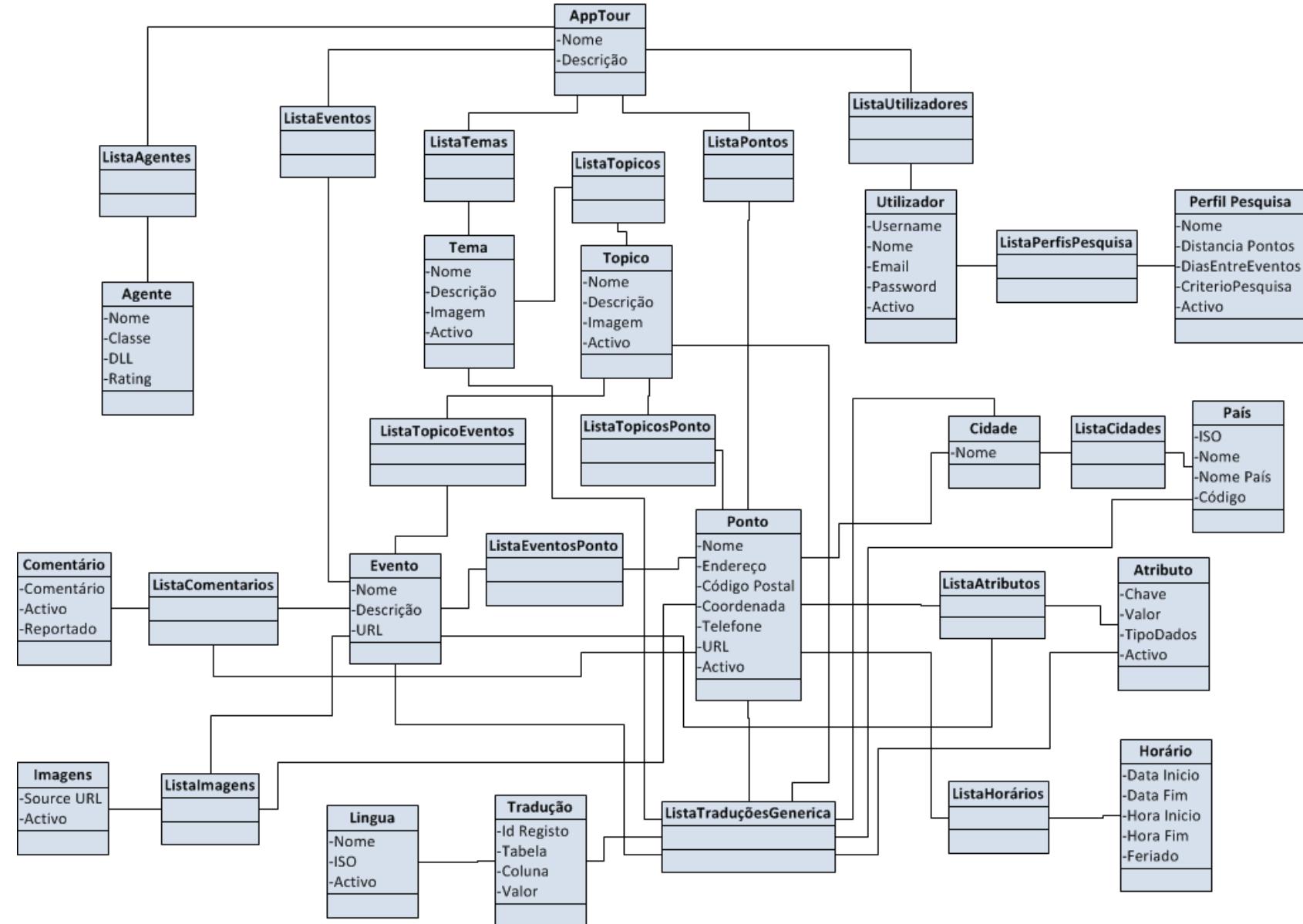


Figura 6. Modelo de Domínio

2.5 Casos de Uso

Em seguida podemos ver os casos de uso relativos à aplicação *AppTour* sob a forma de um diagrama de casos de uso. Neste diagrama podemos ver quais os principais modos de intervenção com a aplicação assim como os actores implicados nesses mesmos processos.

Em seguida segue uma descrição sucinta do objectivo de cada caso de uso, assim como o comportamento esperado de cada um destes. Atente-se que esta descrição apenas refere os mesmos casos de uso de uma forma genérica e textual, pelo que informações mais detalhadas sobre esses mesmos casos de uso, assim como das respostas esperadas do sistema são apresentados no Anexo 10.

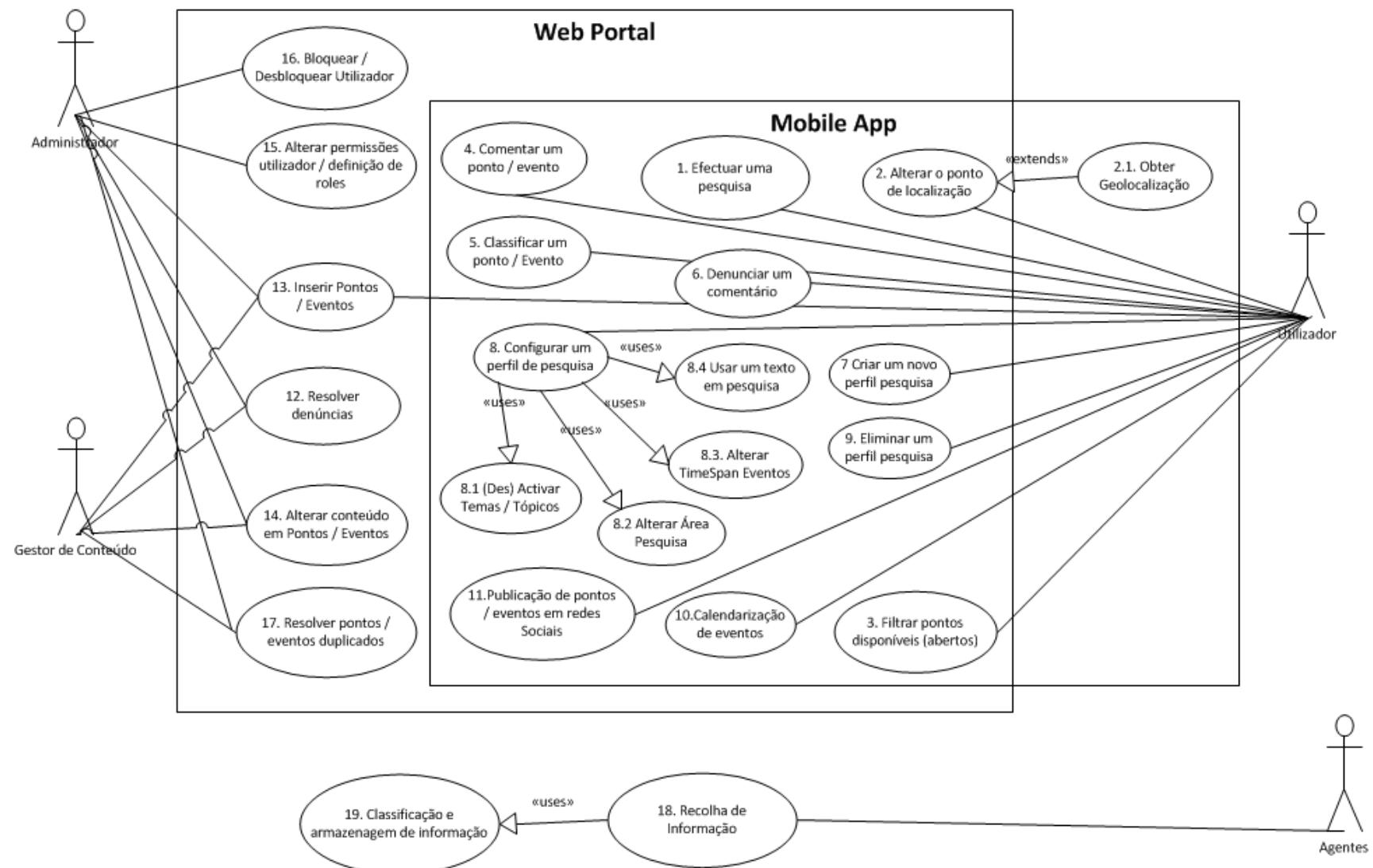


Figura 7. Casos de Uso

2.5.1 Efectuar uma pesquisa

O principal e provavelmente o mais importante caso de uso da aplicação. Neste caso de uso o objectivo passa por efectuar uma pesquisa que obedeça aos critérios de um perfil de pesquisa definido. Depois de enviado ao sistema o perfil a utilizar, este deverá devolver ao utilizador uma lista de pontos que cumpram os requisitos desse mesmo perfil. Deve-se ter em atenção que apenas poderá ser referido um perfil existente e associado ao utilizador.

Tanto no sistema *Android* como no *Web Portal*, essa lista de pontos será apresentada num mapa, em que são representados esses mesmos pontos encontrados, através de marcadores dispostos no local respectivo à posição geográfica de cada ponto, devidamente identificados pelos ícones respectivos ao tópico a que estes se referem.

Após estes marcadores serem clicados estes deverão expandir uma pequena caixa de texto com o nome desse mesmo ponto, assim como permitir uma ligação para uma página de detalhe desse mesmo ponto.

Opcionalmente poderá ser fornecida uma lista dos pontos em vez de os dispor num mapa, onde essa mesma lista poderá eventualmente ter critérios de ordenação.

2.5.2 Alterar o ponto de localização

Este caso de uso baseia-se em saber a localização do utilizador, seja através do uso de funcionalidades oferecidas pelos *browsers* de internet actuais, seja através de funcionalidades de geolocalização dos *smartphones Android*. Após obtida essa localização, esta deve ser guardada de alguma forma, no dispositivo em uso, para poder ser enviada ao sistema no momento de pedir uma pesquisa, juntamente com o perfil de pesquisa escolhido.

Caso essa funcionalidade não esteja disponível, deverá poder ser fornecida directamente pelo utilizador, seja por uma morada, seja por indicação directa num

mapa. Em último caso, deverá ser assumida uma localização por omissão, sendo por exemplo o centro da cidade em questão.

2.5.3 Considerar apenas Pontos disponíveis

Neste caso de uso pretende-se refinar a pesquisa de pontos obtidos. O objectivo passa por impedir que uma pesquisa devolva pontos que devido ao horário, não se encontrem em funcionamento.

Uma vez dada a indicação por parte do utilizador que deseja apenas pontos disponíveis, indicação essa que pode ser fornecida no momento de efectuar a pesquisa, ou associada ao próprio perfil de pesquisa, o sistema ao efectuar a pesquisa deve considerar os pontos que possuam horários e se estes se enquadram à hora correspondente dessa mesma cidade. Caso o Ponto não possua um horário associado, deverá ser dada uma indicação ao utilizador que não existe nenhum horário disponível para esse mesmo Ponto.

2.5.4 Comentar um Ponto/ Evento

Comentar um ponto deverá ser uma funcionalidade que deverá ser disponibilizada ao utilizador quando este consulta os detalhes de um Ponto ou um Evento. Após a apresentação desse Ponto ou Evento, deverá existir um modo de recolher um comentário em texto que será automaticamente incluído em cada nova consulta a esse mesmo Ponto / Evento.

2.5.5 Classificar um Ponto / Evento

A classificação de um Ponto ou Evento é outra funcionalidade que deverá ser permitida ao utilizador na altura em que este visualiza os detalhes desses mesmos Pontos ou Eventos. A classificação consiste numa avaliação de um a cinco, números inteiros naturais enviada pelo utilizador para o sistema. Esta será guardada e usada para mostrar a informação da média dessas mesmas classificações quando é pedido os detalhes desse mesmo Ponto, através da representação de cinco estrelas iluminadas ou não mediante a pontuação calculada.

Deverá apenas ser permitida uma pontuação por utilizador para cada Ponto / Evento, para evitar a pontuação abusiva para um aumento deliberado da média final.

2.5.6 Denunciar um comentário

Uma vez que os comentários podem ser livremente inseridos, e são automaticamente disponibilizados nos detalhes desse mesmo Ponto / Evento, estes deverão ser moderados pelos gestores de conteúdo, para que se evitem casos de comentários indevidos ou despropositados.

Numa tentativa de auxiliar os gestores de conteúdo no processo de moderação, deverá ser disponibilizada uma forma de qualquer utilizador poder denunciar um determinado comentário para que este possa ser mais facilmente identificado e resolvido. Assim sendo estes comentários ficarão marcados para revisão e poderão posteriormente ser vistos por esses mesmos gestores.

2.5.7 Criar um novo Perfil de Pesquisa

Embora no momento de registo de um novo utilizador, seja sempre criado um Perfil de Pesquisa por omissão, este deverá ter a hipótese de criar novos perfis, identificados por um nome. Não há nenhuma necessidade de existir um limite para os perfis existentes. Depois da criação desse mesmo perfil, este será sugerido para a próxima pesquisa, já que é o de criação mais recente.

2.5.8 Configurar um Perfil de Pesquisa

A configuração de um perfil de pesquisa permite ao utilizador configurar os resultados que deseja visualizar. No momento de configuração de um perfil deverão poder ser fornecidos pelo utilizador vários critérios que irão influenciar os resultados das pesquisas a efectuar. Esses critérios deverão respeitar valores estipulados e poderão ser sugeridos pela aplicação. Esses critérios serão os que seguem:

- Um Texto de pesquisa: texto usado para procurar no nome ou descrição de um ponto;

- Área de pesquisa de Pontos: Deverá ser fornecida uma área máxima de pesquisa de Pontos. Essa área será definida em quilómetros e terá de estar compreendida entre os valores de um a vinte, sendo dez um valor sugerido;
- *TimeSpan* de Eventos: Tal como a distância nos Pontos, o *timespan* visa controlar o tempo de pesquisa de Eventos. Neste caso os valores definem dias e deverão estar compreendidos entre um e sete, sendo três um valor sugerido;
- Tópicos abrangidos: As pesquisas apenas irão considerar os Pontos que estejam incluídos nos tópicos seleccionados.

2.5.9 Eliminar um perfil de pesquisa

A eliminação de um perfil de pesquisa é um caso de uso que permite ao utilizador inactivar esse mesmo perfil. Embora não exista uma eliminação efectiva do perfil, este deixará de estar disponível para efectuar pesquisas. No entanto esta inactivação desse mesmo perfil deverá poder ser reversível.

2.5.10 Calendarizar Eventos

Como calendarização de eventos entende-se o acto de permitir ao utilizador, depois de consultar os dados de um determinado evento, exportar esses mesmos dados, tal como nome do evento, descrição e dia e hora de acontecimento do mesmo para uma das aplicações de calendarização existentes. Idealmente essa opção recairá sobre o *Google Calendar*¹⁴ embora outras hipóteses poderão ser consideradas.

2.5.11 Publicar Pontos e Eventos em redes sociais

Dada a influência nos dias de hoje das chamadas redes sociais tanto na comunicação como divulgação de informação, torna-se quase obrigatório para qualquer aplicação que consiga utilizar e partilhar informação sobre estas plataformas.

O utilizador deve então e depois de consultado o detalhe de um ponto e/ou evento, poder ter uma forma de divulgar informação relativa a esse mesmo ponto ou

¹⁴ <https://www.google.com/calendar>

evento sobre as redes sociais mais conhecidas. Essa informação deverá ser no mínimo o nome e uma breve descrição.

Considera-se as redes sociais a partilhar aquelas mais preponderantes nos dias de hoje, como sejam, *Facebook*¹⁵, *Twitter*¹⁶, e *Google+*¹⁷.

2.5.12 *Resolver denúncias*

Tal como referido no ponto 2.5.6 os utilizadores podem auxiliar o gestor de conteúdos a controlar e moderar os comentários que são inseridos. Embora os utilizadores marquem os comentários para moderação, estes continuam visíveis para todos os que peçam informação relativamente ao ponto em questão. Para que tal comentário seja efectivamente removido é necessário que um gestor de conteúdos assuma essa decisão.

Assim todos os comentários marcados para verificação deverão estar disponíveis e evidenciados para verificação pelo gestor de conteúdos e consequente eliminação caso assim se entenda.

2.5.13 *Inserir Pontos e Eventos*

O acto de inserir Pontos e Eventos deve ser permitido a todos os utilizadores da aplicação. Deverá ser disponibilizado um ecrã que permite a recolha dos dados relativos a um ponto e deverão ser validados os atributos obrigatórios relativos a cada Ponto ou Evento.

Deverão também ser validados pormenores como a existência de outros Pontos na mesma coordenada geográfica ou com o mesmo nome, para se evitarem conflitos. Nesse caso, deverá ser mostrado qual o(s) ponto(s) que se apresenta(m) em conflito e permitir a escolha ao utilizador sobre continuar ou não com a operação.

¹⁵ <https://www.facebook.com/>

¹⁶ <http://twitter.com/>

¹⁷ <https://plus.google.com/>

Após a inserção com sucesso do Ponto / Evento, este deverá ser disponibilizado para pesquisas imediatamente. Não carece de nenhuma confirmação por parte de nenhum utilizador com permissões superiores, a informação do ponto, pelo que se confia na informação introduzida e na boa-fé dos utilizadores. Qualquer excepção a esta regra será tratada pelo caso de uso seguinte.

2.5.14 *Alterar conteúdos em Pontos e Eventos*

Como a informação presente nos pontos será maioritariamente inserida ou pelos utilizadores da aplicação ou por processos automáticos é de prever alguma informação errónea ou menos certa em alguns Pontos / Eventos.

Caberá aos gestores de conteúdo ou mesmo ao Administrador a rectificação dessa mesma informação para garantir a credibilidade da aplicação.

Para tal a aplicação deverá permitir um modo de edição dos Pontos apenas para os dois tipos de utilizador descritos acima onde será possível editar toda a informação, de um ponto, ou mesmo se necessário a eliminação desse mesmo ponto.

Opcionalmente poderá ser facilitada uma listagem dos pontos disponíveis, ordenada pela data de criação ou mesmo pela data de edição desse mesmo ponto (inexistente para os pontos novos) para dessa forma permitir um melhor controlo dos Pontos já verificados e quais os novos Pontos que ainda nunca tinham sido validados.

2.5.15 *Alterar permissões Utilizador*

A promoção de determinados tipos de utilizador deverá ser outra situação a contemplar, para assim poderem auxiliar no processo de manutenção da informação.

Apenas os administradores poderão alterar os tipos de utilizador de comum, para gestores de conteúdo ou para administradores e vice-versa.

Poderá existir uma lista de todos os utilizadores, onde poderão ser mencionados o número de acções que cada um tenha efectuado (inserção de pontos, comentários ou denúncias), já que poderá auxiliar na escolha dos utilizadores mais activos.

2.5.16 *Bloquear e desbloquear utilizadores*

O caso de uso de bloquear e/ou desbloquear utilizadores servirá para evitar a reincidência de abusos efectuados pelos utilizadores.

No caso de ser necessário bloquear um utilizador, o administrador, e apenas este, efectuará a pesquisa desse mesmo utilizador, e poderá efectuar esse mesmo bloqueio, onde deverá ser fornecida uma explicação do motivo. Nessa altura o utilizador será notificado que a sua conta ficou bloqueada e será informado desse mesmo motivo, não sendo mais permitido o uso da aplicação por esse mesmo utilizador.

No caso de desbloqueio, o processo deverá ser o mesmo, embora poderá ser dispensada a explicação do motivo do desbloqueio.

2.5.17 *Procurar e resolver Pontos e Eventos duplicados*

A resolução de pontos e eventos duplicados será uma situação a prever, já que não existe um controlo efectivo no momento de criação destes. Assim é necessário que os gestores de conteúdo possam de alguma forma, poder aceder a uma listagem dos pontos disponíveis e possam efectuar pesquisas sobre determinados pontos.

Listagens com vários tipos de ordenação (nome, coordenadas, etc.) poderão fornecer dicas e provas óbvias desses mesmos pontos. Também deverá ser contemplada a pesquisa de pontos que contenham um determinado texto no nome. Outro método poderá passar pela listagem de pontos que possuam uma pequena distância especificada entre eles.

2.5.18 *Recolha de informação automática*

Este caso de uso dedica-se apenas à recolha de informação feita através dos Agentes sobre as diversas fontes que existam na Internet.

Deverá assim poder ser agendado e iniciado um serviço automaticamente, que instancie um número variável de Agentes, um para cada *API* disponível.

Cada um desses Agentes deverá estar encarregado de todos os métodos de autenticação nessas mesmas *APIs*, assim como leitura desses mesmos dados e validação dos limites de informação disponibilizados por cada uma.

Também deverá tarefa desses mesmos Agentes o recomeço de tarefas que tenham ficado de alguma forma incompletas da última migração.

Os Agentes deverão ser independentes entre si embora devam ser controlados de alguma forma por alguma aplicação que receba a informação recolhida por estes, e a forneça ao sistema central. Essa mesma aplicação deve também ser responsável por informar o utilizador do progresso ou dos erros encontrados.

Toda a informação acerca do processo de migração, como o sucesso de uma importação, a falha de comunicação com uma *API*, um erro que possa ser devolvido por essa mesma *API*, ou um erro obtido no processo de gravação da informação do Ponto ou Evento obtido deverá ser guardada em ficheiros de registo para posterior consulta.

2.5.19 *Classificação e armazenagem de informação*

Cada Agente deverá ter em si a informação e o conhecimento necessários para poder fornecer ao sistema central uma informação homogénea e já devidamente formatada e o mais completa possível.

Deverá ser contemplado um método que permita associar os pontos aos tópicos a que estes se inserem, pelo que cada Agente deverá ter acesso a um dicionário ou algo que lhe permita estabelecer uma relação entre as classificações

dadas por cada *API* aos pontos importados e aqueles que são os tópicos respectivos da aplicação *AppTour*.

Outro aspecto a ter em conta é de tentar evitar inserir pontos duplicados. Embora este processo possa ser considerado falível, dada a disparidade possível entre a informação do mesmo ponto por duas *APIs* diferentes, devem ser tentadas várias abordagens, tais como evitar nomes com o mesmo nome, com a mesma coordenada (pode até ser contemplado algum pequeno desvio entre as coordenadas dos vários pontos), ou até mesmo ambos.

3 Desenvolvimento e Implementação

A solução apresentada é um sistema com base em tecnologias *Microsoft* que assenta sobre uma arquitectura *n-layer*. Isto permite a escalabilidade da solução, tanto a nível de volume de dados como a nível de aplicações, mas também uma garantia de integridade dos dados a nível de negócio, garantidos por regras implementadas nos próprios modelos.



Figura 8. Logotipo AppTour

Este sistema é composto por vários componentes de *backend*, sendo estes a parte de *business layer* e *data access layer*, o que permite uma maior robustez e estabilidade, já permite a criação de várias aplicações finais sobre estas camadas, sem que estas possam interferir sobre as regras de negócio e de integridade de dados. Estas interacções são assim isoladas por serviços, funcionando desta forma como uma *black-box*.

A fiabilidade e a segurança desta arquitectura é garantida tanto pela utilização de validadores que garantem as regras de negócio a nível dos modelos como também pela utilização de serviços únicos que asseguram a correcta sequência de trabalho e de acções entre esses mesmos modelos.

Outra das grandes vantagens numa implementação em várias camadas está na facilidade com que se conseguem detectar e resolver problemas tais como baixa performance, erros em tempo de execução ou *bugs* de implementação. Isto deve-se à divisão de responsabilidades entre os vários projectos, o que permite uma maior facilidade desse mesmo diagnóstico.

3.1 Arquitectura

A solução *AppTour* foi pensada na escalabilidade e na possibilidade de criação de várias aplicações sobre uma camada de lógica de negócio e acesso a dados.

Deste modo, com a separação de camadas e a garantia da lógica de negócio na parte interna do sistema, conseguimos garantir que exista várias aplicações a serem criadas sobre o nosso sistema.

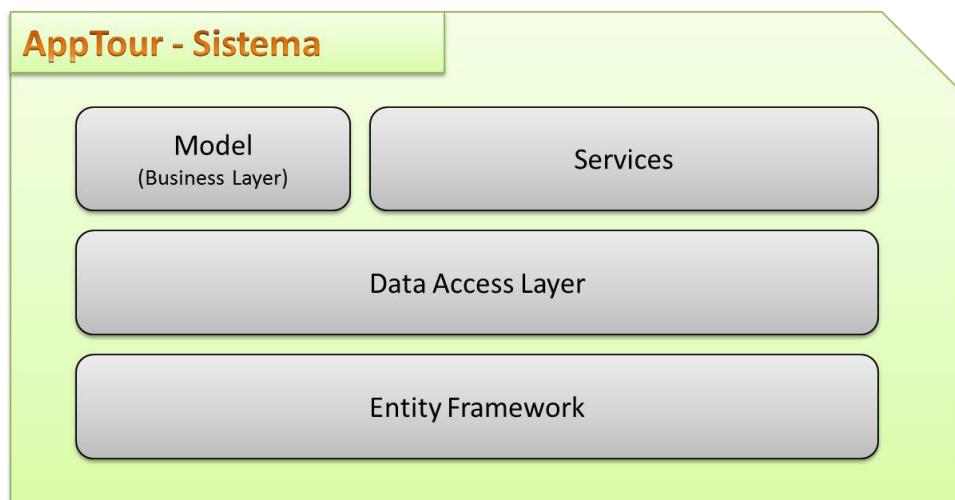


Figura 9. Sistema *Backend* *AppTour* (*Black-Box*)

Uma grande vantagem no isolamento da implementação deste sistema de *backend* é não só esconder os métodos de implementação e regras de funcionamento do negócio em si, mas também a possibilidade de disponibilizar os serviços e os modelos para utilizações externas, sem nos preocuparmos de preocupar com eventuais desvios ao conceito original.

Esta arquitectura foi desenvolvida em tecnologias *Microsoft*, nomeadamente *Microsoft .NET 4.0*.

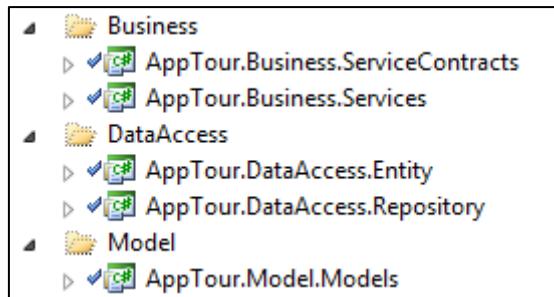


Figura 10. Projectos de *Backend* - AppTour

Estes 5 projectos representam a parte de dados e de lógica de negócio que permite garantir a lógica entre todas os serviços e funcionalidades.

Como solução completa existem cerca de 14 projectos .NET e 1 projeto Java (*Android App*).

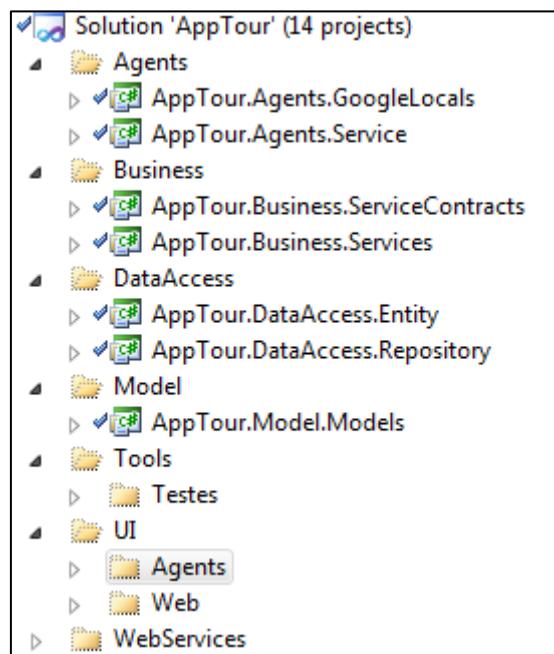


Figura 11. Solução AppTour

A seguir descrevem-se os vários projectos que são visíveis na Figura 11:

Agents

Esta pasta contém todo o sistema de agentes assim como os projectos que incluem as consultas às *APIs*, inserção e verificação de duplicados de pontos e eventos.

AgentsServices

O *AgentsService* é o responsável pela criação de Agentes, pela sua instância e pela inserção de novos pontos e eventos. Como estes não possuem contacto directo com o sistema central, a importação dos dados recolhidos, assim como a sua homogeneização, verificação de duplicados e persistência destes no sistema é só possível através deste componente central.

Os agentes estão guardados na base de dados e são instanciados dinamicamente. São guardados o *namespace* completo mais a classe que é instanciada e o ficheiro *DLL* que contém a referência.

NAME	FULL_CLASS_NAME	DLL_FILE
Google Locals API	AppTour.Agents.GoogleLocals.GoogleLocalsAdapter	AppTour.Agents.GoogleLocals.dll

Figura 12. Exemplo de registo de um agente

Com isto é possível criar dinamicamente os agentes e sempre que houver alguma alteração a ser feita, basta substituir os ficheiros *DLL* correspondentes.

Este serviço disponibiliza uma interface para cada *API* desenvolvida, assim como os modelos que cada *API* tem que preencher para os pontos. Usamos o padrão de desenvolvimento *Adapter* para implementar estar funcionalidade. Assim cada *API* desenvolvida é um *Adaptee*, sendo o *AgentsService* responsável por fornecer os modelos, receber os pontos/eventos criados e os introduzir na base de dados.

É também utilizado o padrão *Observer*, para permitir a comunicação entre qualquer um desses mesmos *Adaptees* com o sistema vigente na altura para apresentação do progresso realizado.

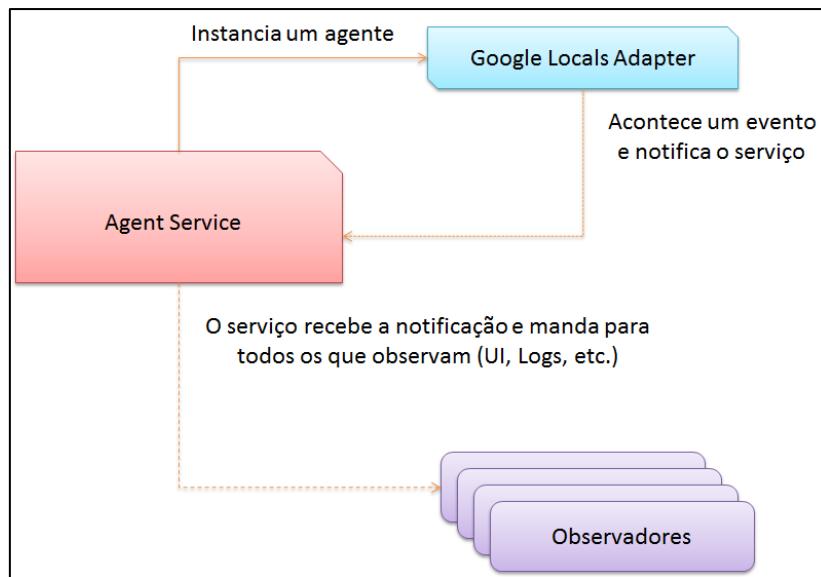


Figura 13. Exemplo do padrão *Observer* implementado nos Agentes

GoogleLocals

Exemplo de um agente que vai buscar os pontos à *API* do *Google Locals* (*Google Maps*).

Este agente tem a periodicidade de 24 horas devido ao limite de pedidos impostos pelo *Google*: 1000 pedidos diários.

Esta *API* consegue ir buscar os pontos no raio de 1000 metros de uma dada coordenada. A lógica que estamos a utilizar é através da localização por códigos postais, ou seja, dado um código postal, é chamada um serviço externo¹⁸ que transforma o código postal numa coordenada e daí é pedido à *API* do *Google Locals* os locais num raio de 1000 metros dessa coordenada.

A partir daí é efectuado toda a validação e mapeamento dos dados recebidos pela *API* para os modelos disponibilizados pelo serviço.

O resumo de processos suspensos é garantido através de um registo na base de dados do último código postal efectuado com sucesso. Como os códigos postais são “alimentados” através de um ficheiro CSV, que tem uma ordem pré-definida de códigos postais a utilizar, é assim possível continuar o processo através do último

¹⁸ Serviço disponibilizado em codigospostais.appspot.com

guardado, bastando para isso continuar por utilizar o seguinte disponível nesse mesmo ficheiro.

Business

Estes componentes, como o próprio nome indica, fazem parte da camada que garante as regras de negócio, juntamente com os Modelos. Tal como referido no ponto 3 e 3.1 os serviços validam a correcta sequência de trabalho entre os modelos. Os serviços são também a porta de entrada para as funcionalidades proporcionadas pelo sistema.

Paralelamente aos serviços existem também *ServiceContracts* que definem a assinatura desses mesmos serviços.

ServiceContracts

Este projecto contém as interfaces que os serviços implementam. Dessa forma todos os serviços são assim identificados e assinados, o que facilita a comunicação via *Web Services*, por *WCF*.

Os *interfaces* estão organizados por entidades, da seguinte maneira:

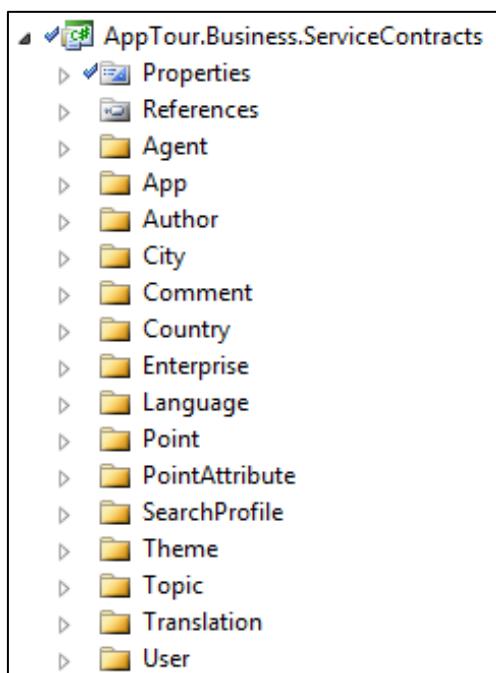


Figura 14. Organização do projecto *ServiceContracts*

Este tipo de padrão de desenvolvimento é denominado *Service Interface*, suportado pela *Microsoft Patterns & Practices*¹⁹.

Segue um exemplo da solução, do interface implementado para os pontos.

```
namespace AppTour.Business.ServiceContracts.Point
{
    public interface IPointService
    {
        [OperationContract]
        IList<PointModel> GetPoints();

        [OperationContract]
        IList<PointModel> GetActivePoints();

        [OperationContract]
        PointModel GetPoint(Guid id);

        [OperationContract]
        void UpdatePoint(PointModel point);

        [OperationContract]
        Guid InsertPoint(PointModel point);

        [OperationContract]
        void DeletePoint(PointModel point);

        [OperationContract]
        IList<PointModel> GetActivePoints(TopicModel Topic);

        [OperationContract]
        PointModel GetActivePoint(Guid id);
    }
}
```

AppTour.Business.ServiceContracts.Point

Figura 15. Interface IPointService

Services

Os serviços são as classes que implementam os *Services Contracts*, explicado acima, definem a lógica e podem conter algumas validações de integridade e segurança.

¹⁹ <http://msdn.microsoft.com/en-us/library/ff647559>

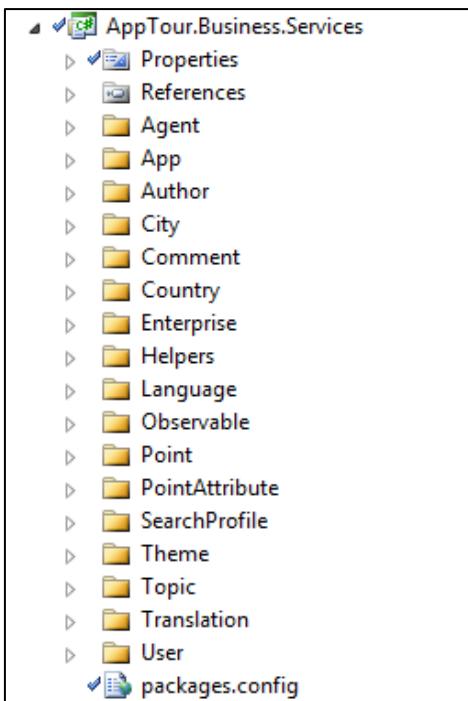


Figura 16. Organização do projeto Services

O projeto está dividido por entidades, conforme acontece nos contractos e todas as classes têm que implementar uma interface respectiva para garantir a implementação do padrão referido anteriormente.

```
namespace AppTour.Business.Services.Point
{
    public class PointService : IPointService
    {
        public IList<PointModel> GetPoints()
        {
            return new PointRepository().GetPoints();
        }
    }
}
```

Figura 17. Pequeno exemplo do serviço implemento para a entidade Ponto.

Estes serviços disponibilizam métodos que são utilizados pelos *Web Services* para interacção com aplicações externas (*Android App*) e o resultado destes é automaticamente serializado, para formatos *standard (Json)*.

Data Access

Este componente é responsável pela persistência dos dados, assim como o acesso e disponibilização da informação. Está dividido em dois projectos: *Entity* e *Repository*.

O projecto *Entity* é um projecto que apenas contém modelos *Object-Relational (ORM)*. Este mapeamento é efectuado automaticamente pela *Entity Framework* e automatizado pela *framework .NET 4.0*, pelo que não requer intervenção externa por parte do desenvolvimento.

O projecto *Repository* contém todos os métodos *CRUD*²⁰ que permitem a manipulação das entidades fornecidas pela *Entity Framework*, e estes são implementados através de *LINQ*²¹, o que permite a manipulação de consultas sobre a forma de objectos em detrimento de declarações *SQL* convencionais.

Entity

O projecto *Entity* é o mais simples e o mais directo de todos. Apenas suporta a *Entity Framework*, que faz o mapeamento de uma base de dados relacional para objectos. A *Entity Framework* também garante a integridade dos dados assim como as regras definidas em base de dados (*constraints, foreign keys, etc.*).

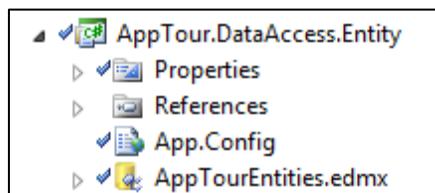


Figura 18. Projecto Entity

Podemos ver na Figura 19 o mapeamento feito pela *Entity Framework* à base de dados, transpondo as tabelas em entidades. São apresentadas as entidades correspondentes aos modelos de negócio.

²⁰ Create, Read, Update e Delete

²¹ Language Integrated Query

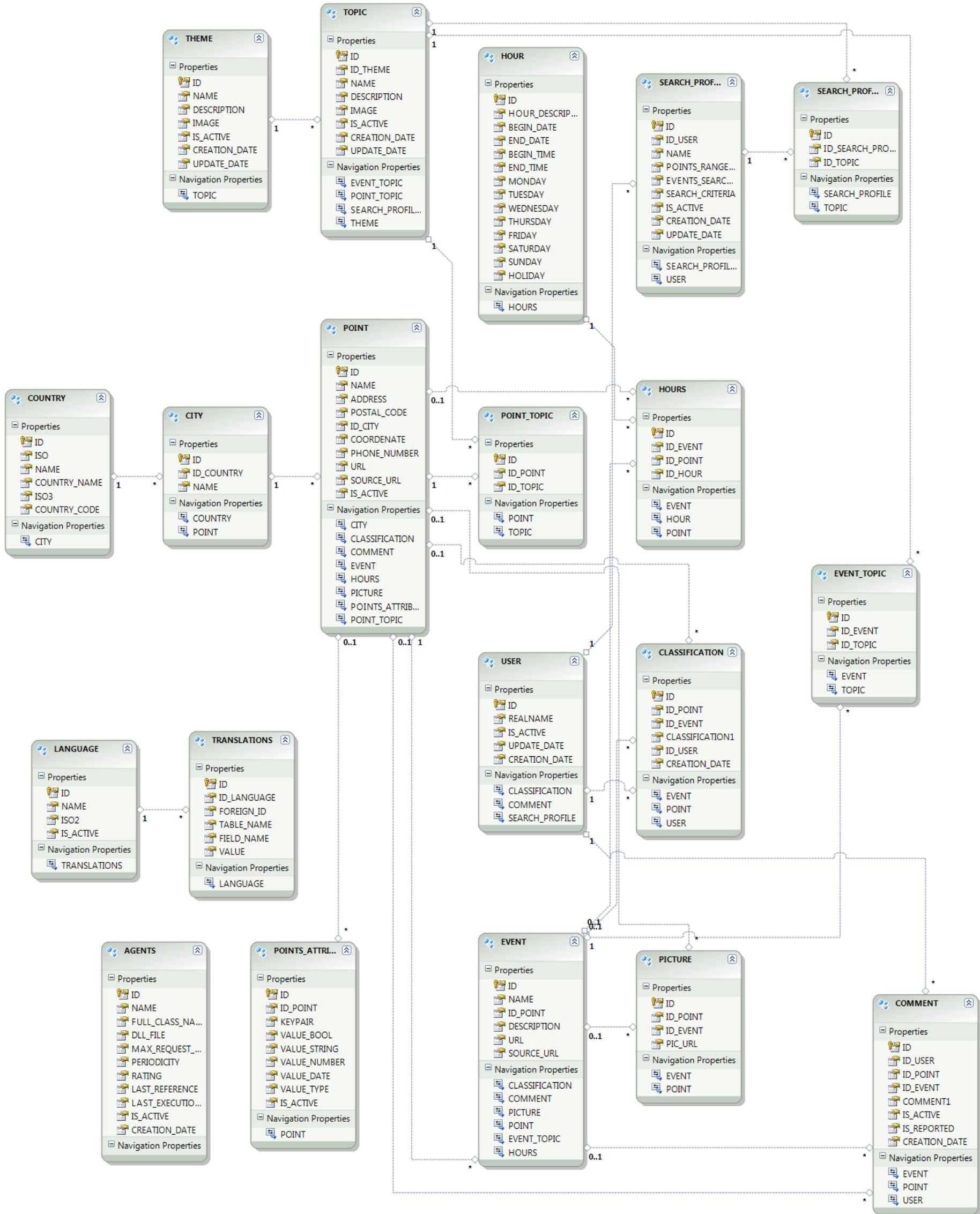


Figura 19. Entity Framework – Modelo parcial AppTour

Repository

O repositório é o único projecto que tem acesso directo à base de dados, ou seja, acesso directo à *Entity*, se considerarmos esta como a nossa base de dados, dada a abstracção que provoca.

Este projecto contém as classes (organizado por entidades tal como os outros projectos), com os respectivos métodos para acesso e manipulação dos dados (vulgarmente conhecidos como métodos *CRUD*) e faz o respectivo mapeamento para os modelos.

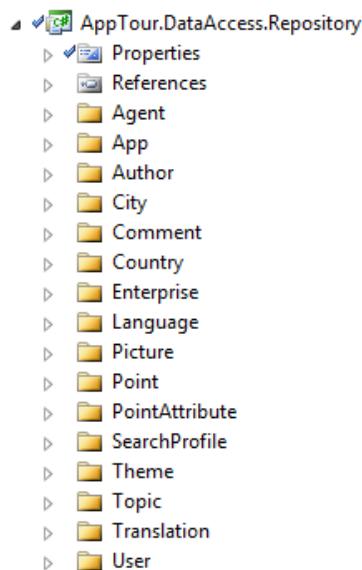


Figura 20. Projecto de Repositório

Consultas a modelos constituídos por colecções de outros objectos, como por exemplo na leitura de Pontos, que implicam a consulta a tópicos, atributos, comentários, imagens e outras tabelas por forma a preencher o modelo, são optimizadas através da utilização de paralelismo. Veja-se a seguir na Figura 21 o exemplo dessa utilização para obter as diversas entidades pertencentes a um ponto, quem apesar de serem diferentes, são reutilizadas entre si, tornando assim possível a separação de responsabilidades entre cada uma dessas entidades.

```

#region + GetPoint(Guid Id)
public PointModel GetPoint(Guid Id)
{
    using (AppTourEntities data = new AppTourEntities())
    {
        PointModel point = this.GetPoints(data).SingleOrDefault(x => x.Id == Id);

        Parallel.Invoke(() =>
        {
            point.Topics = new TopicRepository().GetTopicsForPoint(point.Id);
            point.Attributes = new PointAttributeRepository().GetAttributeForPoints(point.Id);
            point.Pictures = new PictureRepository().GetPicturesFromPoint(point.Id);
            point.Comments = new CommentRepository().GetCommentsForPoint(point.Id);
        });

        return point;
    }
#endregion

```

Figura 21. Exemplo de um método para devolver um Ponto

Com esta separação de camadas de atribuição de responsabilidades é mais fácil e mais produtivo o desenvolvimento de novas funcionalidades, ou até mesmo a optimização das mesmas já que a responsabilidade das tarefas está atribuída a uma entidade e apenas daí vem a informação.

No exemplo seguinte podemos verificar que todos os acessos a dados, respectivos a um ou mais pontos estão nesta classe.

```

public sealed class PointRepository
{
    - IQueryable GetPoints(AppTourEntities data)
    - IQueryable<PointModel> GetActivePoints(AppTourEntities data, TopicModel Topic)
    + GetPoints()
    + GetPoint(Guid Id)
    + GetActivePoint(Guid Id)
    + GetActivePoints()
    + GetActivePoints(TopicModel Topic)
    + UpdatePoint(PointModel point)
    + InsertPoint(PointModel point)
    + DeactivatePoint(PointModel point)
}

```

Figura 22. Exemplo da Classe *PointRepository*.

Embora seja possível efectuar uma consulta a um ou vários Pontos, foram criados alguns métodos adicionais com o objectivo de melhorar a performance. Obtida uma vez que é mais eficaz efectuar uma consulta filtrando à partida alguns resultados esperados, como se pode ver na consulta de pontos por tópicos na Figura 23, do que devolver uma listagem com todos os Pontos para depois serem filtrados e devolvidos apenas os pretendidos, como se pode observar na Figura 24.

```
#region + GetActivePoints(TopicModel Topic)
public IList<PointModel> GetActivePoints(TopicModel Topic)
{
    using (AppTourEntities data = new AppTourEntities())
    {
        var points = this.GetActivePoints(data, Topic).ToList();

        Parallel.ForEach(points, x =>
        {
            x.Topics = new TopicRepository().GetTopicsForPoint(x.Id);
            x.Attributes = new PointAttributeRepository().GetAttributeForPoints(x.Id);
        });
    }

    return points.ToList();
}
#endregion
```

Figura 23. Consulta filtrada no pedido à Base de Dados (boa abordagem)

```
#region + GetActivePoints(Guid Topic)
public IList<PointModel> GetActivePoints(Guid TopicId)
{
    using (AppTourEntities data = new AppTourEntities())
    {
        var points = this.GetPoints(data).ToList();

        Parallel.ForEach(points, x =>
        {
            x.Topics = new TopicRepository().GetTopicsForPoint(x.Id);
            x.Attributes = new PointAttributeRepository().GetAttributeForPoints(x.Id);
        });

        return points.ToList().Where(x => x.Topics.All(y => y.Id.Equals(TopicId))).ToList();
    }
}
#endregion
```

Figura 24. Consulta filtrada no pedido à Base de Dados (má abordagem)

Não existe uma receita mágica para estas soluções. Terão que ser medidos os tempos e baseado no volume de dados verificar o que é mais satisfatório.

Model

Esta pasta contém o projecto *Models*, onde são agregados os modelos de negócio da aplicação, organizado conforme os outros projectos, separado por entidades, como se pode ver na Figura 25.

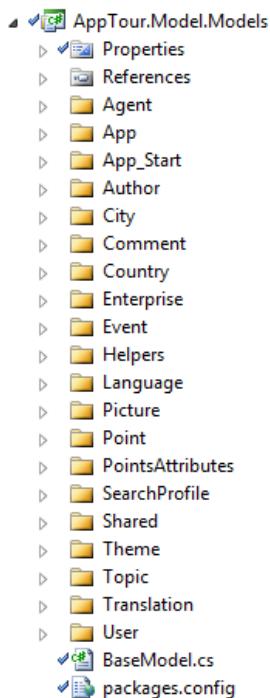


Figura 25. Projecto Models

Os modelos são as classes que reflectem as entidades da base de dados, podendo conter mais ou menos atributos que a própria entidade.

Estes Modelos estão preparados para serem serializados para desta forma poderem ser transmitidos por *Web Services*, através de atributos nativos da *framework 4.0*, conforme se pode ver na Figura 26.

```
namespace AppTour.Model.Models.Point
{
    [DataContract]
    public class PointModel : BaseModel
    {
        public PointModel() { }

        [DataMember]
        public Guid Id { get; set; }

        [DataMember]
        public string Name { get; set; }

        [DataMember]
        public string Address { get; set; }
    }
}
```

Figura 26. Exemplo parcial de um modelo.

Os Modelos juntamente com os serviços ao implementarem um contrato que permite serem acedidos pelo exterior, ver Figura 27, estabelecem um *standard* de comunicação global. Com isto é possível uma comunicação, através de *Web Services*, utilizando esses mesmos contractos para definir uma política de comunicação entre

a nossa aplicação e outras que a utilizem, indiferentemente das plataformas onde estas se encontrem.

```
namespace AppTour.Business.ServiceContracts.Point
{
    [ServiceContract]
    public interface IPontService
    {
        [OperationContract]
        IList<PointModel> GetPoints();

        [OperationContract]
        IList<PointModel> GetActivePoints();

        [OperationContract]
        PointModel GetPoint(Guid id);

        [OperationContract]
        void UpdatePoint(PointModel point);
    }
}
```

Figura 27. Exemplo de Serviços para o Ponto

Isto é uma funcionalidade da *Framework 4.0* com inclusão dos serviços *WCF* e é uma garantia é implementada pela *plataforma .NET*.

Conforme se pode verificar na Figura 26 todas as classes de modelo implementam uma classe *BaseModel*.

Esta classe *BaseModel* é uma classe abstracta que contém os métodos que fazem as validações em tempo de execução. Isto é vantajoso porque, independente da plataforma e aplicação que esteja a utilizar estes modelos é sempre possível garantir a integridade e validação das regras de negócio.

```
namespace AppTour.Model.Models
{
    [DataContract]
    public abstract class BaseModel : IDataErrorInfo
    {
        + bool IsValid

        + virtual ValidationResult SelfValidate()

        IDataErrorInfo Members
    }
}
```

Figura 28. Definição da classe *BaseModel*

No final da classe *PointModel* definimos o método *SelfValidate()* que escreve para aquele modelo, a maneira como vai garantir as regras de negócio, conforme se pode ver na Figura 29.

```

public override ValidationResult SelfValidate()
{
    return ValidationHelper.Validate<PointValidator, PointModel>(this);
}

```

Figura 29. Definição do método *SelfValidate()* do Modelo *PointModel*.

Assim é instanciado um objecto da classe *PointValidator* onde contém todas as validações necessárias. Estas validações são efectuadas por classes externas à aplicação, que podem ser obtidas pelo *NuGet*. Neste caso a extensão utilizada foi o *Fluent Validation*.



```
PM> Install-Package FluentValidation
```

Figura 30. Instalação do Fluent Validation no VS 2010

Com esta extensão instalada é possível definir, de forma fácil e intuitiva, as regras de negócio. Existem no entanto algumas limitações a ter em conta. Não se pode fazer consultas à base de dados no modelo. Assim essa responsabilidade fica atribuída aos serviços, o que reforça a ideia que a camada de negócio se encontra não só nos modelos, mas também nos serviços.

```

using FluentValidation;

namespace AppTour.Model.Models.Point
{
    public class PointValidator : AbstractValidator<PointModel>
    {
        public PointValidator()
        {
            RuleFor(x => x.Name).NotEmpty().WithMessage(ViewRes.ErrorMessages.Required);
            RuleFor(x => x.Topics.Count).GreaterThanOrEqualTo(1).WithMessage(ViewRes.ErrorMessages.Required);
        }
    }
}

```

Figura 31. Definição do Validator do PointModel

Conforme se pode ver na Figura 31 são implementadas duas regras de negócio no modelo do Ponto. Neste exemplo, o nome do Ponto deve estar preenchido, assim como o número de tópicos associados a este deve ser igual ou superior a um (1).

```

try
{
    if (point.IsValid)
    {
        service.UpdatePoint(point);
        return RedirectToAction("Index");
    }
    else
    {
        ModelState.AddModelError("", point.Error);
    }
}
catch (Exception e)
{
    ModelState.AddModelError("", e.Message);
}

```

Figura 32. Exemplo de validação de um ponto

O exemplo mostrado na Figura 32 exemplifica o uso do validador. A invocação da propriedade *IsValid* é suficiente para despoletar a validação dessas mesmas regras previamente definidas.

Todos os modelos contêm um validador, independentemente se tem regras de negócio criadas ou não.

Tools

Esta pasta denominada *Tools* foi pensada para conter projectos de teste implementados e outros projectos auxiliares a todo o sistema *AppTour*.

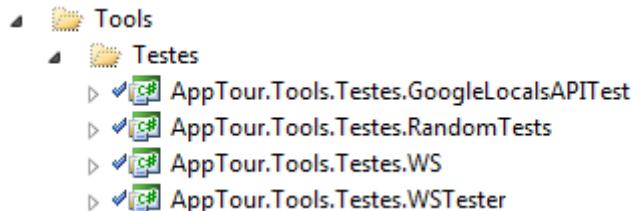


Figura 33. Pasta *Tools* da solução com os respectivos projectos

Inicialmente existiam outras soluções que reescreviam o *Membership* da Microsoft, de modo a personalizar a autenticação à nossa medida.

Concluiu-se que não era uma boa solução e optou-se por fazer apenas uma extensão do *Membership*, colocando-o nos serviços. Deste modo simplificou-se o processo de autenticação permitindo ultrapassar esse obstáculo, mantendo os requisitos e funcionalidades iguais e os mesmos atributos pretendidos.

Actualmente a pasta contém projectos de testes, tanto para os *Web Services* como para os Agentes. Estes projectos são testes implementados por experiencias nossas com objectivos específicos a cada componente. Por norma são projectos do tipo *Console Application*. No entanto serão abordados mais detalhadamente no capítulo 0.

UI

Nesta secção estão organizados as várias aplicações gráficas que fazem parte da solução. Estas aplicações agregam toda a arquitectura e utilizam os serviços por ela disponibilizados. Assim se garante a separação da camada gráfica de toda a camada de negócio e de persistência de dados.

A organização desta pasta, como se pode ver na Figura 34, está dividida por tipo de componentes, isto é, estão agrupados por funcionalidades conceptuais.

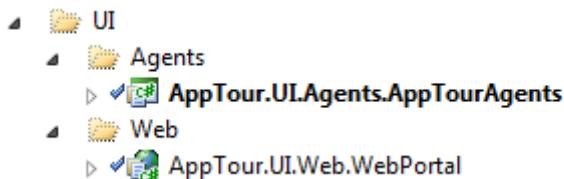


Figura 34. Pasta da UI

Agents

O projecto de *Agents* é composto por uma aplicação *WinForm* que utiliza o serviço *AgentsService*. Caso criássemos também uma *Console Application*, este seria o local da solução onde este projecto seria criado.

Como a estrutura da nossa arquitectura está desenhada para permitir escalabilidade e baixo acoplamento e como toda a parte de negócio está do lado dos serviços, esta aplicação de Agentes, tanto pode ser feita em *WinForms*, *WPF* ou outro tipo de *User End Application* que não influencia em nada o modo como actua o serviço e apenas terá que utilizar os métodos correctos.

O reflexo do descrito no parágrafo anterior é a aplicação *UI* de Agentes (Figura 35), que é provavelmente, das aplicações mais básicas e simples que

compõem esta solução, pois a responsabilidade que tem é invocar o serviço e apresentar o resultado numa caixa de texto.

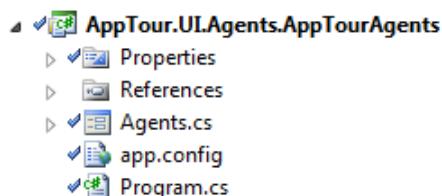


Figura 35. Projecto UI Agents

Tal como mencionado no capítulo 2.3.6 esta aplicação é apenas constituído por uma simples caixa de texto em que não existem muitos cuidados com a apresentação nem com a autenticação.

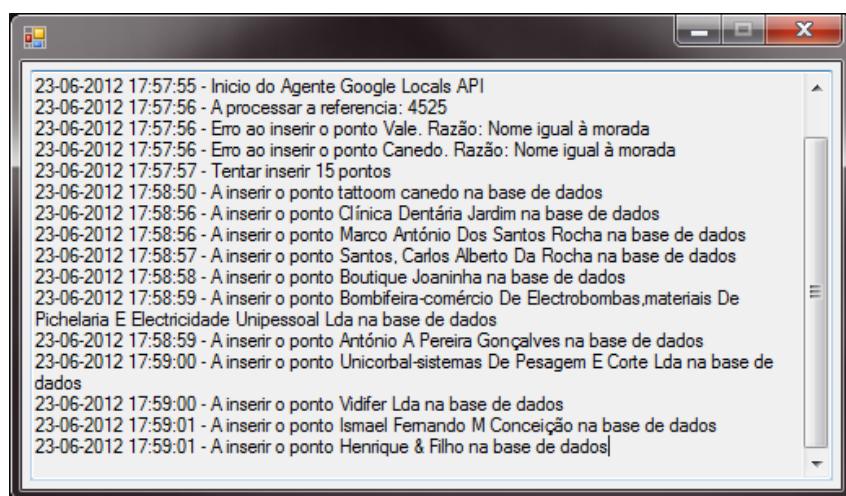


Figura 36. Exemplo da aplicação Agentes em execução.

Conforme se pode ver na Figura 36, a aplicação *UI* dos Agentes apenas recebe as notificações e é responsável por apresenta-las numa caixa de texto.

Web Portal

O Web Portal é uma aplicação Web, desenvolvida em *ASP .NET MVC 3* e como tal, é composta por uma organização bastante mais complexa.

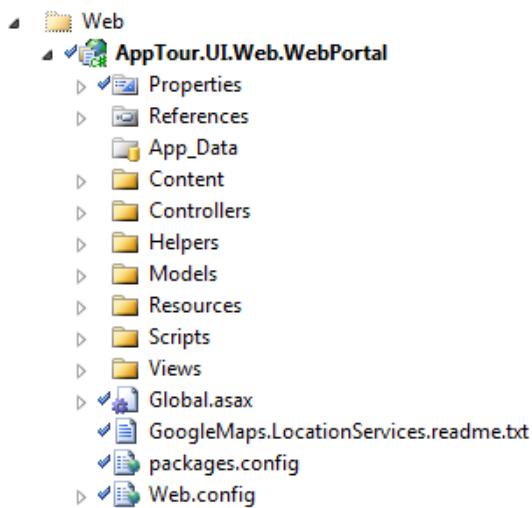


Figura 37. Projecto Web Portal

O projeto *Web Portal* encontra-se organizado conforme o padrão implementado pela *Microsoft*, em *MVC* com mais alguns recursos.

Content

Contém os ficheiros *CSS*, temas e imagens que definem o aspecto gráfico do portal.

Também é aqui que se inclui *plugins* extra, neste caso o *shadowbox*, que enriquece a experiência do utilizador no portal, através do uso de simples animações e renderização de imagens e página de uma forma fluida e dinâmica.

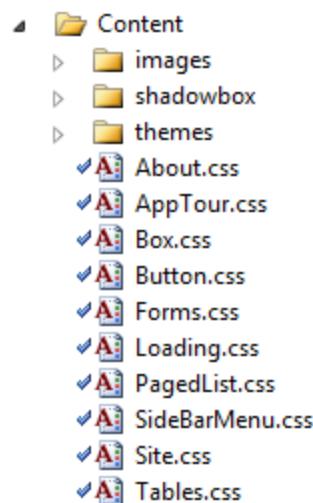


Figura 38. Conteúdo da pasta *Content* do Web Portal

Controllers

É um elemento constituinte do padrão *MVC* e é responsável pelo *workflow* do processo que se está a utilizar na respectiva entidade, como está detalhadamente explicado no capítulo 1.5.

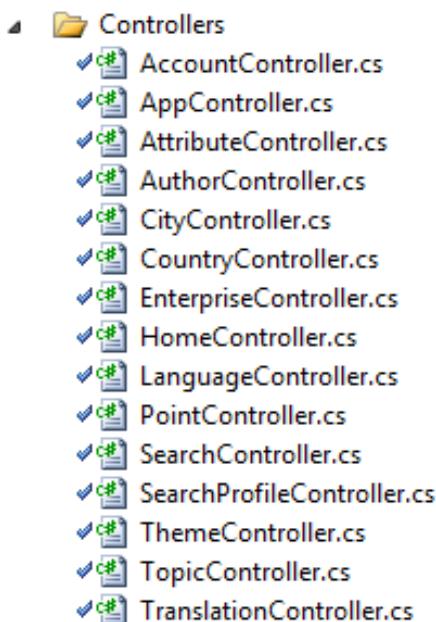


Figura 39. Controllers no Web Portal

Helpers

Inclui classes utilitárias que ajudam no tratamento e fluidez da aplicação. Normalmente são classes estáticas e auto-sustentadas. Podem também existir classes que são extensões a objectos e que ajudam no funcionamento da aplicação.

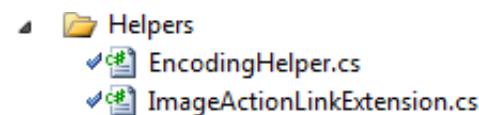


Figura 40. Pasta Helpers da aplicação Web Portal

Actualmente apenas temos *helpers* para tratamento do *URL* do portal, ou seja, codificação e descodificação das palavras de modo a ser transmissíveis em *URLs* válidos.

A outra classe *ImageActionLinkExtension* é uma extensão ao *MVC* que permite construir *tags* em *HTML* para imagens, nas *Views*.

Models

Estes modelos fazem parte da arquitectura *MVC*, mas nesta aplicação utilizamos os modelos da arquitectura do sistema, descritos neste mesmo capítulo (componente *Model*), de forma a manter coerentes as regras de negócio. Porém existe necessidade de utilizar outros modelos adaptados à aplicação do *Web Portal*.

Pode-se considerar estas classes como modelos adaptados para o portal e utilizados como extensões aos modelos reais do sistema, necessários para o correcto funcionamento da aplicação.

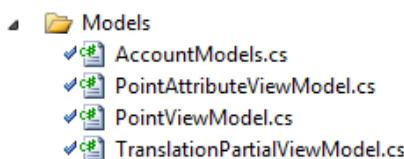


Figura 41. Modelos da aplicação Web Portal

Resources

Os *resources* são ficheiros que contêm as definições e traduções da multilingue do *web portal*. Estes ficheiros estão organizados por entidades e de momento estão definidas duas línguas.

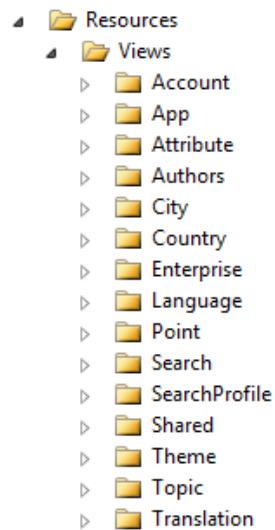


Figura 42. Resources na aplicação Web Portal

Mais detalhes podem ser vistos na implementação de *multilingue*, no Capítulo 3.4.20.

Scripts

Pasta onde se encontram os ficheiros *Javascript*, assim como as bibliotecas de *jQuery* para o web portal.

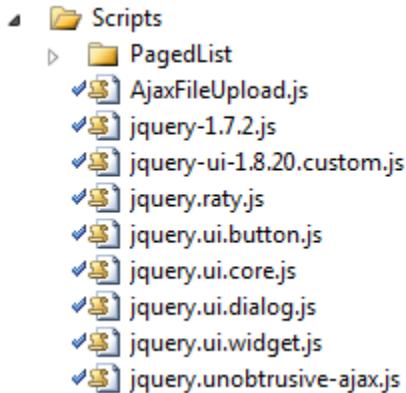


Figura 43. *Scripts* do Web Portal

Views

As *views* são componentes da arquitectura *MVC* que contém as páginas de apresentação para o utilizador final. Estas estão organizadas por entidades conforme os outros componentes na aplicação e no sistema.

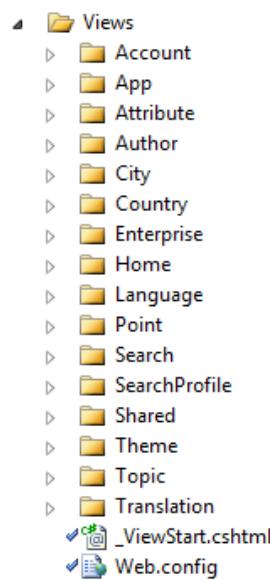


Figura 44. *Views* do Web Portal

Por norma, cada *view* está associada a uma *action* no *controller*, ou seja, o *controller* recebe um pedido na *action*, trata a lógica e manda para a vista, o resultado.

```

public class CityController : Controller
{
    Attributes
    Index()
    Create()
    Edit(Guid id)
    Delete(Guid id)
}

```

Figura 45. City Controller e respectivas Actions

Pode haver exceções, como por exemplo, no caso das *actions* que podem servir como pedidos AJAX, conforme pode ver na Figura 47, em que este caso não existe a renderização de uma *view* mas sim uma devolução de uma *string*.

```

#region + InsertComment(string userId, string pointId, string name, string msg)
[HttpPost]
public ActionResult InsertComment(string userId, string pointId, string name, string msg)
{
    if (userId == null || name == null || msg == null
        || pointId == null || userId == string.Empty
        || name == string.Empty || msg == string.Empty
        || pointId == string.Empty)
        return RedirectToAction("Index", "Home");

    if (isGuid.IsMatch(userId) && isGuid.IsMatch(pointId))
    {
        CommentModel comment = new CommentModel
        {
            Id = Guid.Empty,
            Point = new PointService().GetPoint(new Guid(pointId)),
            Comment = msg,
            User = new UserService(). GetUser(new Guid(userId)),
            IsActive = true,
            IsReported = false
        };
        if (comment.IsValid)
        {
            Guid id = new CommentService().AddComment(comment);
            return Content("sucesso");
        }
        return Content("Model Not Valid: " + comment.Error);
    }
    return RedirectToAction("Empty");
}
#endregion

```

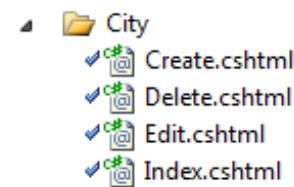


Figura 46. Views da City

Figura 47. Método *InsertComment* do controlador *PointController*.

Web Services

Os *Web Services* são desenvolvidos em tecnologia *Microsoft WCF* e definidos sobre uma arquitectura *REST*.

Os *Web Services* implementados estão adaptados para qualquer sistema ou plataforma utilizando como meio de transmissão *Json* ou apenas texto, em casos mais simples.

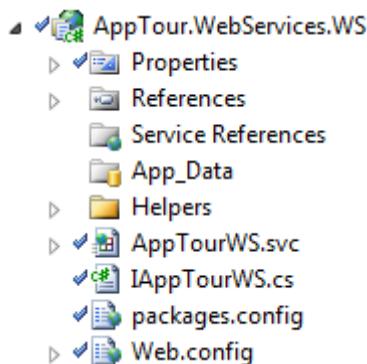


Figura 48. Projecto Web Services

Para definição dos métodos do serviço e também devido à arquitectura *WCF*, é utilizado um *interface* que define a assinatura dos métodos e o método de acesso.

```

[ServiceContract]
public interface IAppTourWS
{
    [OperationContract]
    [WebInvoke(UriTemplate = "/Authentication",
               ResponseFormat = WebMessageFormat.Json,
               BodyStyle = WebMessageBodyStyle.WrappedRequest)]
    Stream Authentication(Stream streamData);
}
  
```

Figura 49. Interface *IAppTourWS* que define os métodos *WCF*.

A definição dos métodos está reescrita na classe *AppTourWS*, que implementa a interface mostrada anteriormente.

Com estes *Web Services* activos a facilidade de desenvolvimento e a centralização dos pedidos torna-se muito mais fácil e mais robusta devido ao facto de se encontrarem no mesmo sistema e implementarem as ultimas tecnologias *Microsoft*, sendo os modelos, serviços e acessos totalmente compatíveis.

3.2 Modelo de Dados

O modelo de dados foi idealizado a partir do modelo de domínio, conforme se pode ver na Figura 6. Seguindo a abordagem de *Database First*, o modelo de dados foi, por assim dizer, o ponto inicial de construção da aplicação e modelação das entidades a utilizar.

Algumas entidades, tais como os pontos e futuramente os eventos, são desenhadas com o intuito de permitir a hipótese de guardar informação adicional sobre estas mesmas entidades, que não fosse prevista inicialmente, através de tabelas auxiliares, com a designação da tabela, seguida do sufixo “*ATTRIBUTES*”. A ideia consiste em ter um par *Key, Value* e a definição do tipo de dados do *value*, permitindo assim operações específicas ao tipo de dados usado. Alguns exemplos passam pelo cálculo de valores numéricos, diferença de datas, etc.

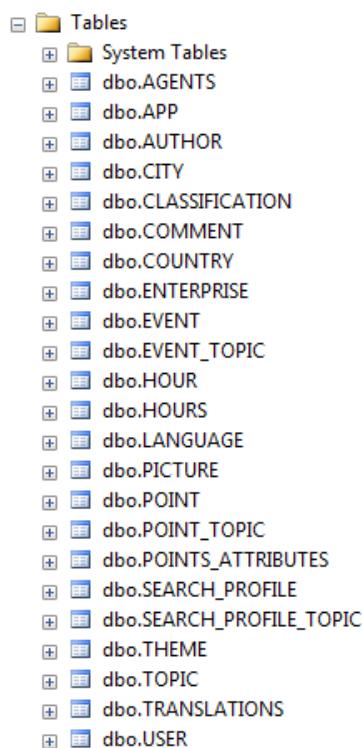


Figura 50. Tabelas de base de dados AppTour

Para melhorar o desempenho e repartir funcionalidades optou-se por separar os dados do *Membership* com os dados da *AppTour*. Foram criadas duas bases de dados, conforme se pode ver na Figura 51.

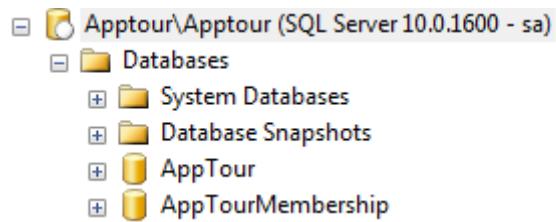


Figura 51. Instância da solução AppTour

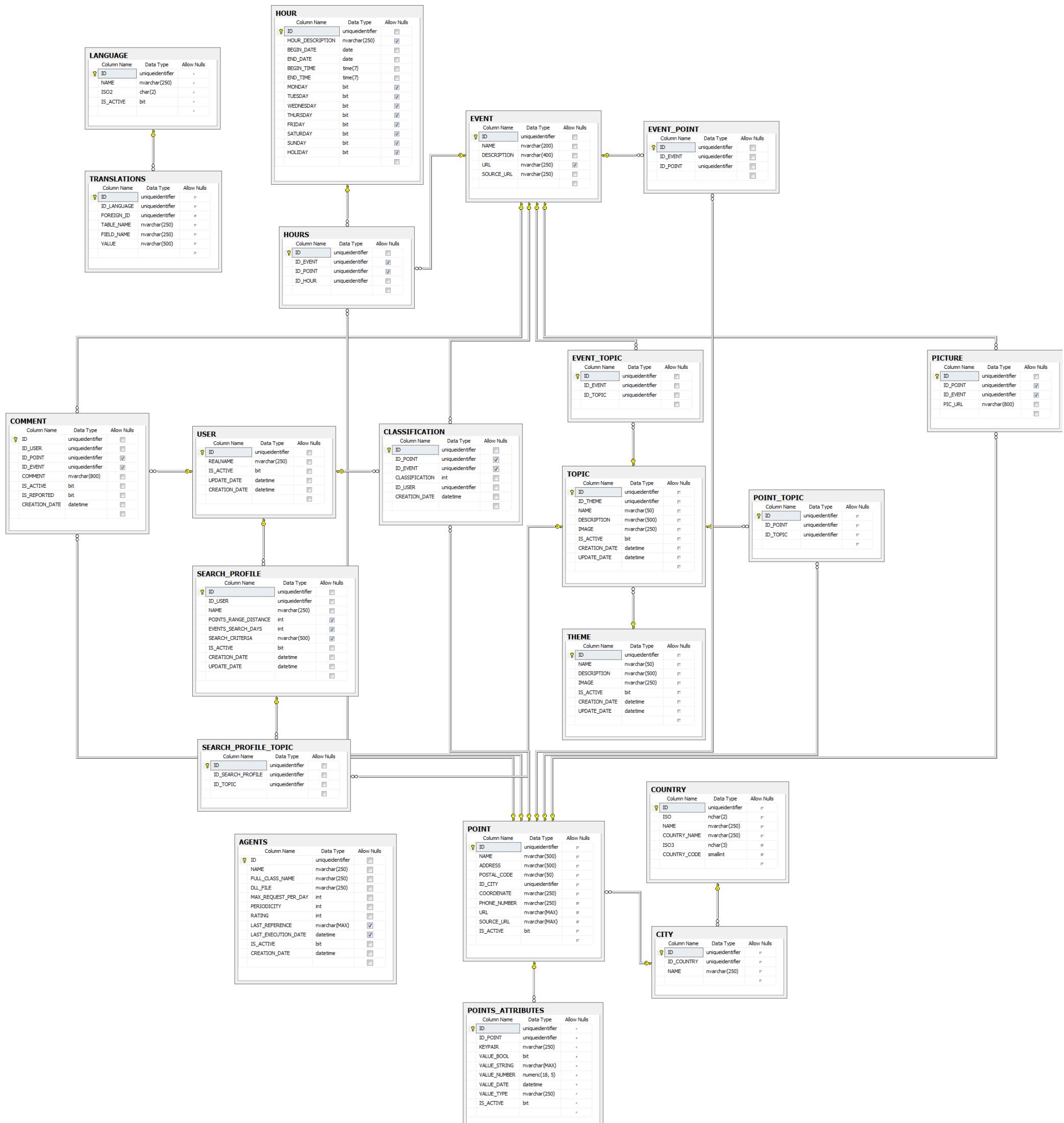


Figura 52. Modelo E-R

Agents

A tabela *Agents* contém a definição de todos os Agentes que são executados pelo nosso sistema de agentes. Nesta pode-se encontrar informação necessárias para garantir factores como a autonomia e escalabilidade, assim como campos necessários para a criação dinâmica dos mesmos.

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
NAME	nvarchar(250)	<input type="checkbox"/>
FULL_CLASS_NAME	nvarchar(250)	<input type="checkbox"/>
DLL_FILE	nvarchar(250)	<input type="checkbox"/>
MAX_REQUEST_PER_...	int	<input type="checkbox"/>
PERIODICITY	int	<input type="checkbox"/>
RATING	int	<input type="checkbox"/>
LAST_REFERENCE	nvarchar(MAX)	<input checked="" type="checkbox"/>
LAST_EXECUTION_DATE	datetime	<input checked="" type="checkbox"/>
IS_ACTIVE	bit	<input type="checkbox"/>
CREATION_DATE	datetime	<input type="checkbox"/>

Figura 53. Definição da tabela Agents

City

A tabela *City* contém o nome das cidades a que pertencem os pontos e eventos. Não é dada muita relevância ao detalhe das cidades, como se pode ver nos campos da tabela (Figura 54), já que inicialmente o projecto refere-se apenas a uma cidade, não havendo assim necessidade de estabelecer uma distinção a nível de negócio.

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
ID_COUNTRY	uniqueidentifier	<input type="checkbox"/>
NAME	nvarchar(250)	<input type="checkbox"/>

Figura 54. Definição da tabela City

Classification

A tabela *Classification* contém os valores que os utilizadores fornecem para estabelecer a classificação dos pontos ou eventos.

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
ID_POINT	uniqueidentifier	<input checked="" type="checkbox"/>
ID_EVENT	uniqueidentifier	<input checked="" type="checkbox"/>
CLASSIFICATION	int	<input type="checkbox"/>
ID_USER	uniqueidentifier	<input type="checkbox"/>
CREATION_DATE	datetime	<input type="checkbox"/>

Figura 55. Definição da tabela Classification

Contém restrições de negócio e de integridade de dados tais como:

- A classificação só pode ir de 1 a 5
- Os campos *ID_POINT* e *ID_EVENT* não podem ser os dois preenchidos

Comment

Esta tabela contém os comentários que os utilizadores podem inserir nos pontos ou nos eventos. Tal como na tabela de classificações não é possível o preenchimento de um *ID_POINT* e um *ID_EVENT* para um mesmo registo

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
ID_USER	uniqueidentifier	<input type="checkbox"/>
ID_POINT	uniqueidentifier	<input checked="" type="checkbox"/>
ID_EVENT	uniqueidentifier	<input checked="" type="checkbox"/>
COMMENT	nvarchar(800)	<input type="checkbox"/>
IS_ACTIVE	bit	<input type="checkbox"/>
IS_REPORTED	bit	<input type="checkbox"/>
CREATION_DATE	datetime	<input type="checkbox"/>

Figura 56. Definição da tabela Comments

Country

Esta tabela contém os países todos existentes e respectivos códigos. Este nível de detalhe é importante para a interacção com as APIs e os Agentes.

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
ISO	nchar(2)	<input type="checkbox"/>
NAME	nvarchar(250)	<input type="checkbox"/>
COUNTRY_NAME	nvarchar(250)	<input type="checkbox"/>
ISO3	nchar(3)	<input checked="" type="checkbox"/>
COUNTRY_CODE	smallint	<input checked="" type="checkbox"/>

Figura 57. Definição da tabela Country

Event

A tabela *Event* contém as definições de um evento. Na prática contém toda a informação necessária para o modelo evento.

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
NAME	nvarchar(200)	<input type="checkbox"/>
ID_POINT	uniqueidentifier	<input type="checkbox"/>
DESCRIPTION	nvarchar(400)	<input type="checkbox"/>
URL	nvarchar(250)	<input checked="" type="checkbox"/>
SOURCE_URL	nvarchar(250)	<input type="checkbox"/>

Figura 58. Definições da tabela Event

Event_Point

Esta tabela estabelece a relação de *N-N* que existe entre pontos e eventos. Dessa forma é possível a um ponto ter associado vários eventos, assim é possível a um Evento estar associado a vários pontos.

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
ID_EVENT	uniqueidentifier	<input type="checkbox"/>
ID_POINT	uniqueidentifier	<input type="checkbox"/>

Figura 59. Definições da tabela Event_Point

Event_Topic

A tabela *Event_Topic* é uma tabela que guarda os tópicos a que pertencem os eventos, ou seja, faz a relação de *N-N*. Esta relação é necessária, pois tanto os eventos podem ser associados a vários tópicos, como os tópicos podem ser associados a vários eventos.

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
ID_EVENT	uniqueidentifier	<input type="checkbox"/>
ID_TOPIC	uniqueidentifier	<input type="checkbox"/>

Figura 60. Definições da tabela Event_Topic

Hour

A tabela *Hour* contém as definições para detalhar os horários, tanto de um ponto como de um evento. Esta tabela contém alguns campos que visam alguma flexibilidade na atribuição de horários.

Assim sendo, os campos de *BEGIN_DATE* e *END_DATE* visam a atribuição do horário para um período estipulado entre datas, usado, por exemplo, para atribuição de horários de verão / inverno.

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
HOUR_DESCRIPTION	nvarchar(250)	<input checked="" type="checkbox"/>
BEGIN_DATE	date	<input type="checkbox"/>
END_DATE	date	<input type="checkbox"/>
BEGIN_TIME	time(7)	<input type="checkbox"/>
END_TIME	time(7)	<input type="checkbox"/>
MONDAY	bit	<input checked="" type="checkbox"/>
TUESDAY	bit	<input checked="" type="checkbox"/>
WEDNESDAY	bit	<input checked="" type="checkbox"/>
THURSDAY	bit	<input checked="" type="checkbox"/>
FRIDAY	bit	<input checked="" type="checkbox"/>
SATURDAY	bit	<input checked="" type="checkbox"/>
SUNDAY	bit	<input checked="" type="checkbox"/>
HOLIDAY	bit	<input checked="" type="checkbox"/>

Figura 61. Definição da tabela Hour

Os campos *BEGIN_TIME* e *END_TIME* correspondem ao horário efectivo de abertura, sendo que para atribuição de um período de almoço, se considerariam dois registos nesta tabela.

Os restantes campos booleanos visam a efectividade desse mesmo horário em caso de este se encontrar no dia de semana (ou feriado) respectivo. Dessa forma pode-se considerar ou não, determinados dias semanais no horário apresentado.

Hours

A tabela *Hours* contém as relações entre pontos e eventos com a tabela *Hour*.

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
ID_EVENT	uniqueidentifier	<input checked="" type="checkbox"/>
ID_POINT	uniqueidentifier	<input checked="" type="checkbox"/>
ID_HOUR	uniqueidentifier	<input type="checkbox"/>

Figura 62. Definição da tabela Hours

Language

A tabela *Language* persiste os idiomas a utilizar no sistema.

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
NAME	nvarchar(250)	<input type="checkbox"/>
ISO2	char(2)	<input type="checkbox"/>
IS_ACTIVE	bit	<input type="checkbox"/>

Figura 63. Definição da tabela Language

Picture

A tabela *Picture* guarda as referências (diga-se *URI*) para os pontos ou eventos. Com isto, pode-se guardar imagens que um ponto ou evento possam ter associados.

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
ID_POINT	uniqueidentifier	<input checked="" type="checkbox"/>
ID_EVENT	uniqueidentifier	<input checked="" type="checkbox"/>
PIC_URL	nvarchar(800)	<input type="checkbox"/>

Figura 64. Definição da tabela Picture

Point

A tabela *Point* contém as definições básicas de um ponto. Pode haver necessidade de existir outros atributos que não estejam nos campos actuais e por isso foi desenvolvido a tabela *Points_Attributes* para guardar essa informação.

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
NAME	nvarchar(500)	<input type="checkbox"/>
ADDRESS	nvarchar(500)	<input type="checkbox"/>
POSTAL_CODE	nvarchar(50)	<input type="checkbox"/>
ID_CITY	uniqueidentifier	<input type="checkbox"/>
COORDENATE	nvarchar(250)	<input type="checkbox"/>
PHONE_NUMBER	nvarchar(250)	<input checked="" type="checkbox"/>
URL	nvarchar(MAX)	<input checked="" type="checkbox"/>
SOURCE_URL	nvarchar(MAX)	<input checked="" type="checkbox"/>
IS_ACTIVE	bit	<input type="checkbox"/>

Figura 65. Definição da tabela Point

Point_Topic

A tabela *Point_Topic* mantém os registos de relação com os pontos e os tópicos, ou seja, verifica quais os tópicos a que pertencem o ponto – relação *N-N*, relação essa justificada pelo mesmo motivo que na tabela *Event_Topic*.

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
ID_POINT	uniqueidentifier	<input type="checkbox"/>
ID_TOPIC	uniqueidentifier	<input type="checkbox"/>

Figura 66. Definição da tabela Pont_Topic

Points_Attributes

A tabela *Points_Attributes* define os atributos extra que um ponto pode ter, isto é, permite a atribuição de informação extra, infinitamente escalável, que pode posteriormente ser mostrada ao utilizador.

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
ID_POINT	uniqueidentifier	<input checked="" type="checkbox"/>
KEYPAIR	nvarchar(250)	<input type="checkbox"/>
VALUE_BOOL	bit	<input checked="" type="checkbox"/>
VALUE_STRING	nvarchar(MAX)	<input checked="" type="checkbox"/>
VALUE_NUMBER	numeric(18, 5)	<input checked="" type="checkbox"/>
VALUE_DATE	datetime	<input checked="" type="checkbox"/>
VALUE_TYPE	nvarchar(250)	<input type="checkbox"/>
IS_ACTIVE	bit	<input type="checkbox"/>

Figura 67. Definição da tabela Points_Attributes

Esta tabela funciona sob o princípio que o valor de um atributo pode ser guardado num dos seguintes tipos:

- VALUE_BOOL guarda valores booleanos (Sim/Não)
- VALUE_STRING atributos de texto
- VALUE_NUMBER atributos numéricos, incluindo decimais
- VALUE_DATE guarda valores de Datas e Horas

Os tipo de valor desses mesmos atributos são depois identificados pelo campo *VALUE_TYPE*, que guarda o tipo de dados, respetivo, tipo esse que é identificado pela *framework .NET*, classe *System.Type*.

A razão desta diferenciação de valores, é não só pela correcta formatação dos dados a mostrar, como também para permitir um futuro reaproveitamento desses mesmos atributos para operações matemáticas ou mesmo de pesquisa.

Search_Profile

A tabela *Search_Profile* guarda os perfis que cada utilizador pode ter.

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
ID_USER	uniqueidentifier	<input type="checkbox"/>
NAME	nvarchar(250)	<input type="checkbox"/>
POINTS_RANGE_DIST...	int	<input checked="" type="checkbox"/>
EVENTS_SEARCH_DAYS	int	<input checked="" type="checkbox"/>
SEARCH_CRITERIA	nvarchar(500)	<input checked="" type="checkbox"/>
IS_ACTIVE	bit	<input type="checkbox"/>
CREATION_DATE	datetime	<input type="checkbox"/>
UPDATE_DATE	datetime	<input type="checkbox"/>

Figura 68. Definição da tabela Search_Profile

Search_Profile_Topic

Esta tabela guarda, para cada perfil de pesquisa, os tópicos que afecta.

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
ID_SEARCH_PROFILE	uniqueidentifier	<input type="checkbox"/>
ID_TOPIC	uniqueidentifier	<input type="checkbox"/>

Figura 69. Definição da tabela Search_Profile_Topic

Theme

A tabela *Theme* guarda os temas a que os tópicos pertencem, ou seja, para cada tema existe 1 ou mais tópicos - relação 1-N.

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
NAME	nvarchar(50)	<input type="checkbox"/>
DESCRIPTION	nvarchar(500)	<input type="checkbox"/>
IMAGE	nvarchar(250)	<input type="checkbox"/>
IS_ACTIVE	bit	<input type="checkbox"/>
CREATION_DATE	datetime	<input type="checkbox"/>
UPDATE_DATE	datetime	<input type="checkbox"/>

Figura 70. Definição da tabela Theme

Topic

A tabela *Topic* guarda os tópicos que existem no sistema.

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
ID_THEME	uniqueidentifier	<input type="checkbox"/>
NAME	nvarchar(50)	<input type="checkbox"/>
DESCRIPTION	nvarchar(500)	<input type="checkbox"/>
IMAGE	nvarchar(250)	<input type="checkbox"/>
IS_ACTIVE	bit	<input type="checkbox"/>
CREATION_DATE	datetime	<input type="checkbox"/>
UPDATE_DATE	datetime	<input type="checkbox"/>

Figura 71. Definição da tabela Topic

Translations

Esta tabela é genérica e guarda a definição de traduções para todos os campos das várias tabelas existentes na base de dados.

O campo *TABLE_NAME* corresponde à tabela onde se quer estabelecer a tradução, assim como o campo *FIELD_NAME* que corresponde à coluna. Para se identificar o *tuplo* a traduzir, é necessário identificar o *ID* dessa mesma tabela (existente em todas por convenção) com o campo *FOREIGN_ID*.

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
ID_LANGUAGE	uniqueidentifier	<input type="checkbox"/>
FOREIGN_ID	uniqueidentifier	<input checked="" type="checkbox"/>
TABLE_NAME	nvarchar(250)	<input type="checkbox"/>
FIELD_NAME	nvarchar(250)	<input type="checkbox"/>
VALUE	nvarchar(500)	<input type="checkbox"/>

Figura 72. Definição da tabela Translation

User

A tabela *User* guarda a informação extra, relativa à aplicação para os utilizadores.

Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	<input type="checkbox"/>
REALNAME	nvarchar(250)	<input type="checkbox"/>
IS_ACTIVE	bit	<input type="checkbox"/>
UPDATE_DATE	datetime	<input type="checkbox"/>
CREATION_DATE	datetime	<input type="checkbox"/>

Figura 73. Definição da tabela User

3.3 Aplicação Android

3.3.1 Arquitectura em Android

A Arquitectura da aplicação *Android* está convencionada segundo os padrões de desenvolvimento do *Android SDK*²², conforme se pode ver na Figura 74. Existem pastas distintas para os diversos elementos e componentes da aplicação.

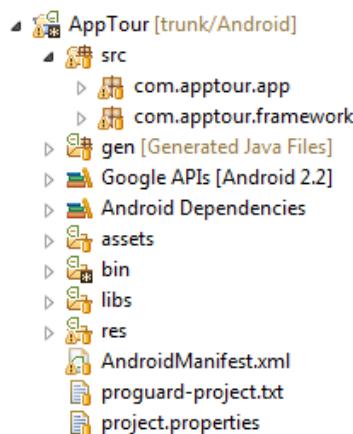


Figura 74. Estrutura do projecto AppTour Android

O projecto contém dois packages, dentro da pasta *src*, visto que esta é responsável por guardar os ficheiros código-fonte da aplicação, que contêm o desenvolvimento efectuado:

com.apptour.app

Este package contém as *Activities* e *Dialogs* que representam os ecrãs de interface com o utilizador, conforme se vê na Figura 75.

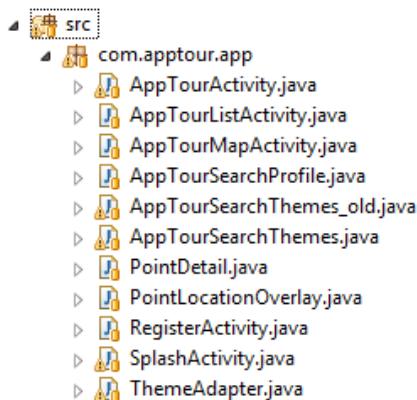


Figura 75. package com.apptour.app

²² <http://developer.android.com/tools/projects/index.html#ApplicationProjects>

com.apptour.framework

Este *package* inclui as classes de ajuda, utilitários, modelos de negócio e restantes classes que implementam o código para a comunicação do sistema central.

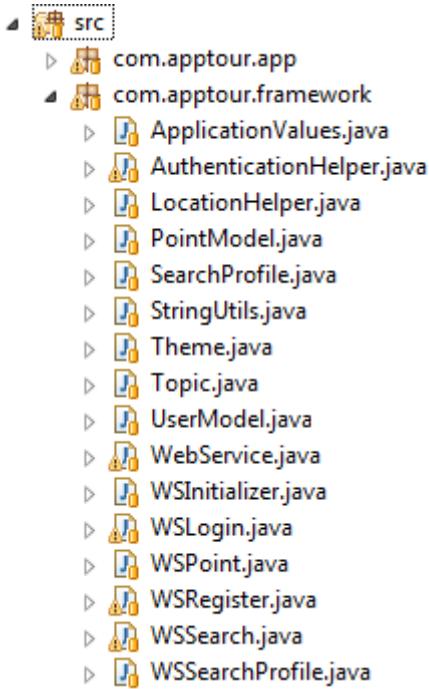


Figura 76. package com.apptour.framework

Neste projeto são utilizados alguns padrões *GoF*²³, tais como Singleton na classe *ApplicationValues*, onde esta actua como uma sessão da aplicação.

```

package com.apptour.framework;

import java.util.ArrayList;

public class ApplicationValues extends Application{

    private static ApplicationValues instance;

    // Singleton Pattern
    public static ApplicationValues getInstance(Context context){
        if (instance==null){
            instance = new ApplicationValues(context);
        }
        return instance;
    }

    private ApplicationValues(Context c){
        // Have we got Data to Load?
    }
}
  
```

Figura 77. Singleton na classe ApplicationValues

²³ Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides conhecidos como "Gang of Four" ([GoF](#))

Podemos também observar classes que representam os modelos de negócio da aplicação *Android* (*PointModel*, *UserModel*, *SearchProfile*, *Theme*, *Topic*), que não são necessariamente iguais aos modelos recebidos do sistema, tanto por excesso como também por falta de informação.

As classes responsáveis pela comunicação com o sistema são as classes *WebService.java* (classe abstracta) e as classes com o prefixo *WS*. Estas últimas herdam da classe *WebService.java*, que contém todo o código necessário à comunicação com os *Web Services* do sistema e apenas estabelecem alterações necessárias aos atributos a enviar, assim como a de-serialização dos objectos recebidos para os modelos *Android*.

3.3.2 Actividades em Android

Uma programação em *Android* difere ligeiramente de outros interfaces que estamos habituados normalmente, como por exemplo no *Web Portal*.

Tal deve-se ao facto de que uma programação em *Android* é feita através da sucessão e interacção de diversas *Activities*. Estas *Activities* representam ecrãs de interacção com o utilizador. Uma aplicação *Android* é por norma constituída por várias *Activities* que se sucedem umas às outras por uma ordem pré-determinada. Assim sendo não é possível a um utilizador o acesso directo a uma determinada *Activity* tal como podemos por exemplo aceder a uma determinada página usando um *URL* específico.

Devido ao descrito no parágrafo anterior, previamente ao desenvolvimento de uma aplicação *Android* é boa prática estipular as *Activities* existentes, assim como qual o fluxo permitido entre estas. Uma boa maneira de descrever esse mesmo fluxo e relações entre as diversas *Activities* pode passar por um diagrama de Actividades.

O Diagrama de actividades seguinte, descreve esse mesmo funcionamento das várias *Activities*, assim como da navegação possível entre estas.

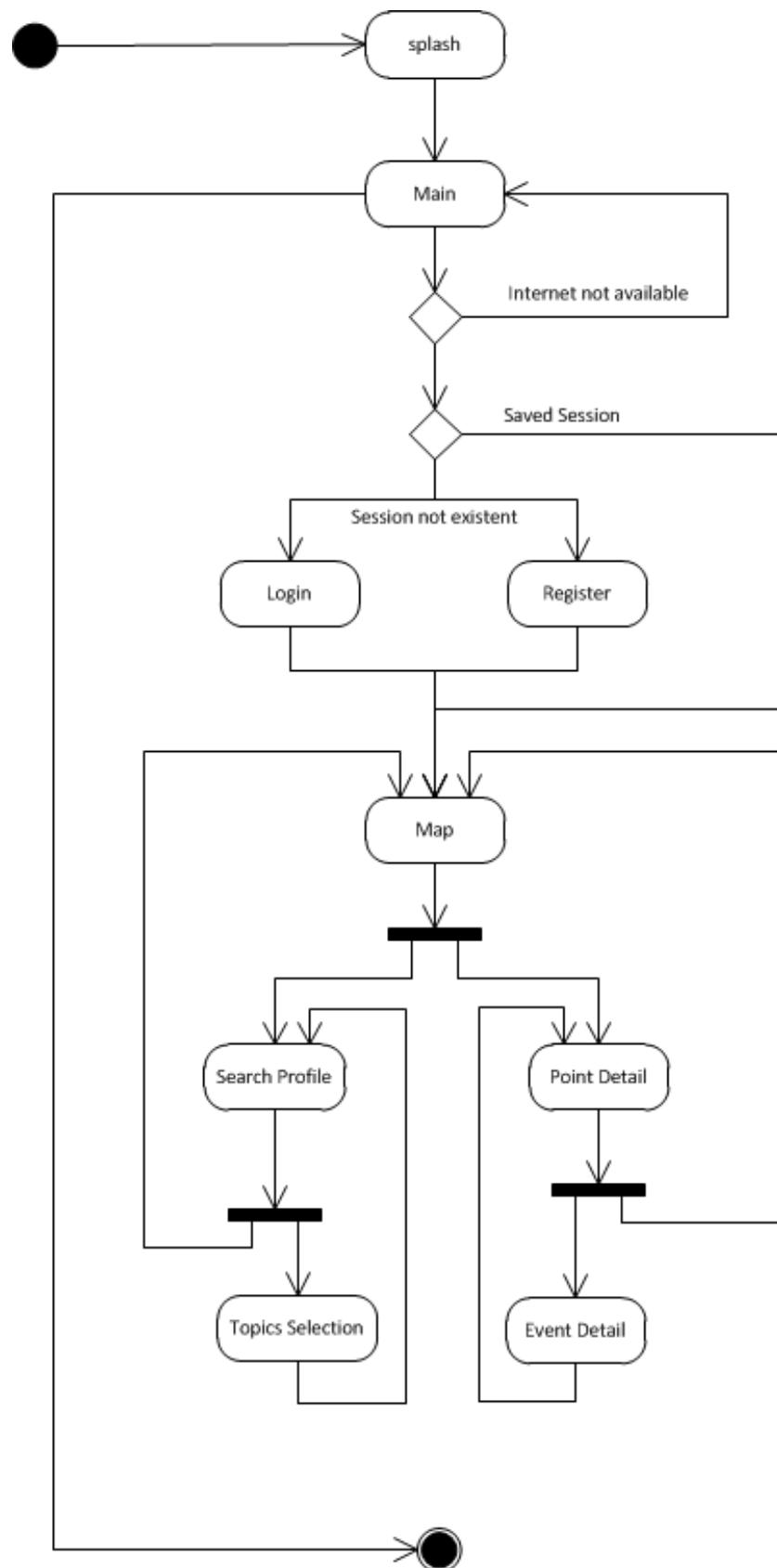


Figura 78. Diagrama de Actividades em Andoid

3.4 Implementação das funcionalidades

Nos capítulos seguintes serão discutidos os métodos de abordagem utilizados para a implementação dos casos de uso apresentados. Alguns dos casos de uso, poderão não estar implementados na totalidade, ou simplesmente não estarem implementados sequer. Tal facto deveu-se a limitações temporais o que impediram a conclusão com sucesso de todos os casos de uso planeados.

No entanto, no capítulo 4.2 far-se-á a descrição daquela que poderia ser uma abordagem a usar para a implementação dos métodos em falta.

3.4.1 Efectuar uma pesquisa

Componentes que despoletam a funcionalidade:

- Android App
- Web Portal

O método chamado pelo *Web Service* contém as seguintes especificações:

```
[OperationContract]
[WebInvoke( UriTemplate = "/Search", ResponseFormat = WebMessageFormat.Json,
            BodyStyle = WebMessageBodyStyle.WrappedRequest)]
Stream Search(Stream streamData);
```

Figura 79. Assinatura e especificação do método *Search* do *Web Service WCF*

Este é um método de pedido *POST* e como tal o parâmetro de entrada é do tipo *Stream*, sendo a saída uma serialização em *JSON* dos resultados obtidos.

A *Stream* de entrada terá de ter a seguinte estrutura:

- | | | |
|-------------------|--------------------------------|--------|
| • UserId | Id do Utilizador | Guid |
| • SearchProfileId | Id Perfil de Pesquisa | Guid |
| • Coordinates | Coordenada (<i>lat,long</i>) | string |

Exemplo:

```
UserId=3311F714-4B83-48F1-A143-2E15A7D0D376&SearchProfileId=D9CB6D2D-807B-41E8-
A4D5-2649142A387C&Coordinates=41,2641,-8,656
```

Android App

Na aplicação *Android*, no botão “*Do Search*” do menu de topo, é possível efectuar uma pesquisa, baseado na localização.

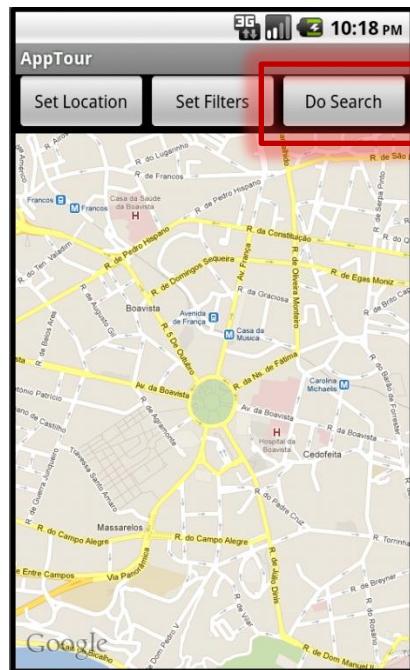


Figura 80. Aplicação AppTour- Pesquisa

Uma vez iniciada esta pesquisa será invocado o *Web Service* que utiliza os serviços respectivos para efectuar uma pesquisa, que podem ser vistos em Anexo 1.

Depois do pedido executado pelo *Web Service* com sucesso, a aplicação recebe uma lista de pontos em *Json*, onde estes são convertidos em marcadores, que serão posteriormente distribuídos pelo mapa e mostrados à medida que são recebidos.

Web Portal

No *Web Portal* a funcionalidade de pesquisa pode ser acedida pelo menu lateral, expandindo o menu *General* e clicando sobre *Search*. Será aberto uma página tal como se vê na Figura 81 que permite escolher um perfil de pesquisa sobre o qual se deseja basear a pesquisa. Opcionalmente pode ser enviada um texto que servirá como filtro de texto e que afectará os resultados obtidos juntamente com o perfil de pesquisa.

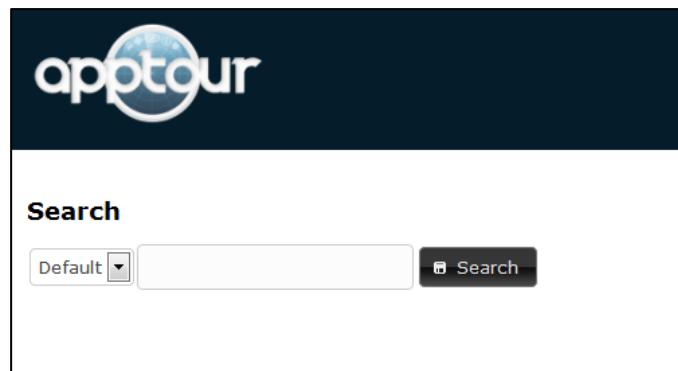


Figura 81. Efectuar uma pesquisa no Web Portal

Depois de recebido os pedidos do serviço, a aplicação processa a lista de pontos recebidos e mostra os resultados numa vista, como se pode ver na Figura 82.

Pesquisa

Default ▾ hotel

Hotel Star inn
Rua da Senhora do Porto 930
4250-453 - Porto, Portugal

[Hotel Star inn](#) | [228 347 000](#)

Complexo Hoteleiro de Sant'ana
Rua De Santana, 947
4465-742 - Leça do Balio, Portugal

[Complexo Hoteleiro de Sant'ana](#) | [229 069 780](#)

Figura 82. Exemplo de pesquisa através do Web Portal

Tanto o *Web Portal*, como a aplicação *Android* através dos *Web Services*, iniciam o serviço que efectua uma pesquisa de Pontos / Eventos. Podemos ver como é esse mesmo processo através do diagrama de sequência no Anexo 1.

Como o processo de pesquisa pode se tornar um pouco moroso à medida que o número de pontos e eventos crescem, foi optado por deixar que este processo se

desenrolasse o mais possível a nível da base de dados. Dessa forma as consultas às várias tabelas necessárias, assim como alguma manipulação de dados seria feita num procedimento apenas.

O método utilizado para pesquisa na base de dados é um *Stored Procedure* responsável com o nome *dbo.Search*. Na Figura 85 podemos ver os parâmetros de entrada do procedimento.

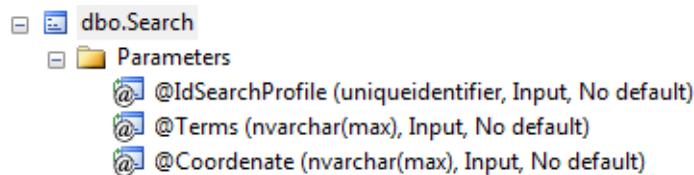


Figura 83. Definição do Stored Procedure Search

O resultado deste procedimento é uma lista de *PointId*, que são depois mapeados pelo repositório para a aplicação.

3.4.2 Alterar o ponto de localização

Componentes que despoletam a funcionalidade:

- Android App
- Web Portal

Android App

Na aplicação *Android*, no botão “Set Location” do menu de topo, é possível utilizar as potencialidades do *smartphone* para saber a localização efectiva deste.

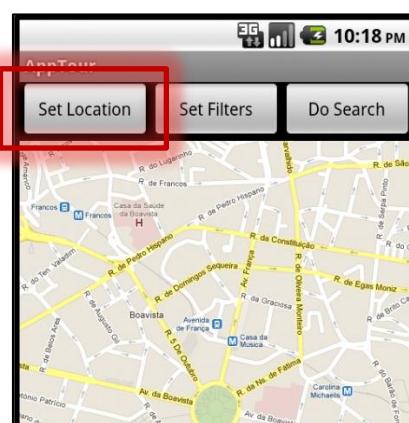


Figura 84. Aplicação AppTour – Localização

A localização pode ser obtida através do uso da rede de comunicações, ou através do GPS incorporado, caso o smartphone tenha um. A classe *LocationHelper.java* irá assim tentar obter a localização do telemóvel através da implementação de *listeners* que ficam à escuta de respostas a pedidos feitos à framework *Android*.

Uma vez obtida a localização, esta é guardada internamente no telefone e apenas é transmitida ao sistema, no momento em que faz um pedido de pesquisa. Não há uma necessidade de retransmissão permanente de novas coordenadas se não houver um propósito de as usar, como numa pesquisa.

Web Portal

No *Web Portal* a acção de recolha de localização é feita no momento de acesso ao mesmo, já que não existe necessidade de efectuar uma actualização constante, uma vez que normalmente esta não mudará. Ao entrar no *Web Portal* o utilizador será confrontado com uma questão que lhe permite enviar ou não a sua localização ao sistema. A forma como essa questão é colocada depende do *browser* utilizado para a visualização do *Web Portal* mas na Figura 85 podemos ver um exemplo de como ocorre no *Google Chrome*.

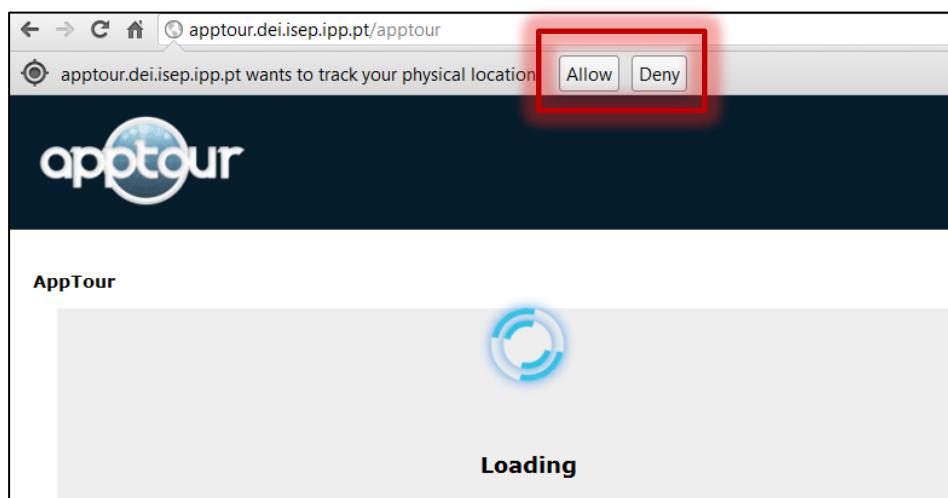


Figura 85. Pedido de obtenção de localização no Google Chrome

A localização em web *browsers* é conseguida através da *W3C Geolocation API*²⁴ desenvolvida pelo *W3C* com o objectivo de criar um protocolo *standard* da recolha de informação de localização geográfica para um dispositivo *client-side*.

As páginas web podem utilizar esta *API* de duas maneiras. Ou directamente caso os *browsers* a implementem (*Firefox 3.5+*, *Google Chrome*, *Opera 10.6+*, *Internet Explorer 9.0*) ou através do *plugin Google Gears Geolocation API*²⁵, que embora descontinuado em 2010, permite um suporte para *browsers* mais antigos.

Depois de recolhida esta informação relativamente à sessão, esta é guardada na sessão e utilizada de cada vez que é efectuada uma nova pesquisa.

3.4.3 Considerar apenas Pontos disponíveis

Esta funcionalidade não se encontra implementada e como tal é descrita com mais pormenor nos capítulos 4.1 e 4.2.

²⁴ http://en.wikipedia.org/wiki/W3C_Geolocation_API

²⁵ http://en.wikipedia.org/wiki/Google_Gears

3.4.4 Comentar um Ponto/Evento

Funcionalidade apenas implementada no *Web Portal* nesta fase inicial. Esta funcionalidade pode ser iniciada na página de detalhe de um ponto. Como podemos ver na Figura 86 é disponibilizado ao utilizador autenticado a possibilidade de comentar um ponto.

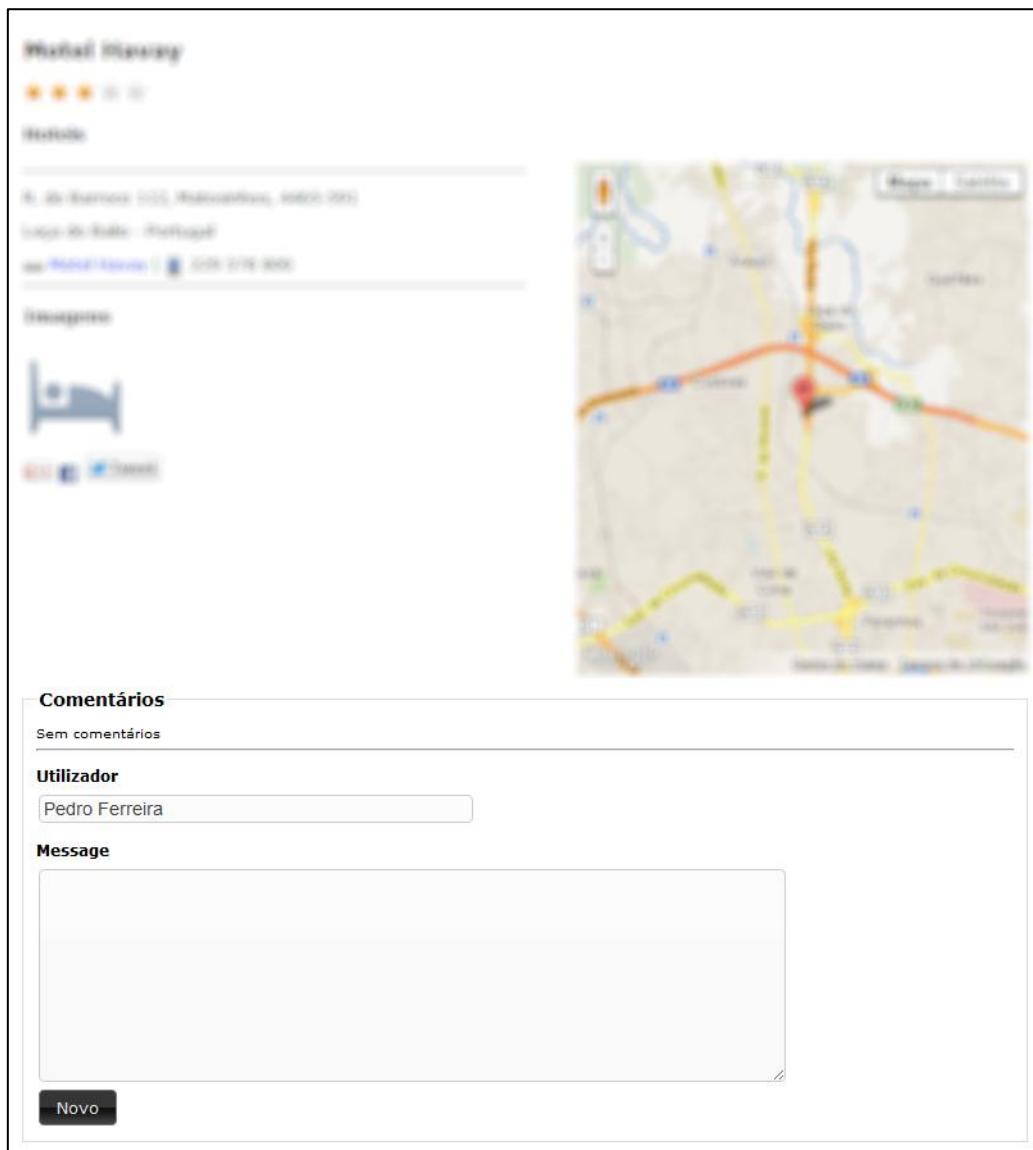


Figura 86. Exemplo de um comentário de um ponto

O processo inicia num pedido *AJAX* no momento que se pressiona o botão Novo. Para garantir que não são enviados comentários duplicados, ao carregar no botão este desaparece. Este código é programado em *jQuery* juntamente com *Razor Engine*, de modo a garantir a fiabilidade do código. Um exemplo, está na criação do caminho *URL*, através de uma função do *MVC .NET* (*Url.Content*), onde é

garantido que independentemente do *URL* que esteja a utilizar, o pedido é sempre feito para o endereço correcto. Pode-se ver o exemplo na Figura 87.

```

$("#save").click(function () {
    $("#save").hide("slow");
    var data = $("#sendcomment").serialize();
    $.ajax(
    {
        url: "@Url.Content("~/Point/InsertComment")",
        secureuri: false,
        data: data,
        dataType: 'text',
        type: "POST",
        success: function (data, status) {
            if (status == 'success') {
                if (data == 'sucesso') {
                    setTimeout(function() { location.reload(); }, 500);
                }
                else {
                    alert(data);
                    $("#save").show("slow");
                }
            }
        },
        error: function (data, status, e) {
            alert(e);
            $("#save").show("slow");
        }
    });
});

```

Figura 87. Pedido AJAX, em jQuery, para inserir comentário

Após o pedido, se for efectuado com sucesso, actualiza a página e assim o novo comentário já irá aparecer. Caso contrário, há um erro que será mostrada numa caixa de diálogo e reaparece o botão de enviar.

O *jQuery* serializa o formulário e invoca um método por *POST*, *InsertComment*, que pertence ao controlador *PointController*. O diagrama de sequência pode ser visto até à invocação do serviço, na Figura 89 e completo no Anexo 2. De seguida é mostrado o código que foi utilizado no controlador, para processar o pedido AJAX, conforme se pode ver na Figura 88.

```
public ActionResult InsertComment(string userId, string pointId, string name, string msg)
```

Figura 88. Assinatura do método *InsertComment*.

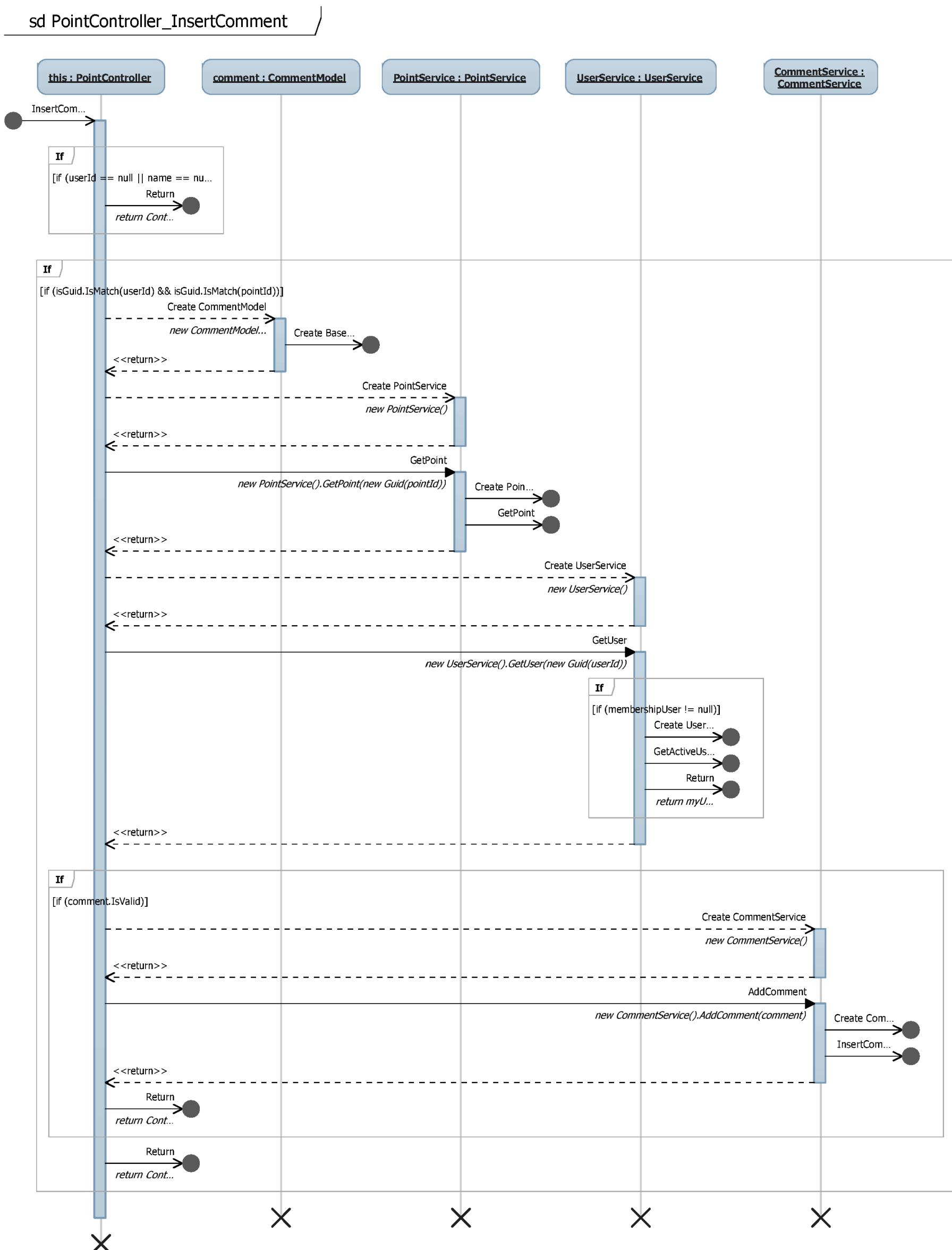


Figura 89. Diagrama de Sequência da funcionalidade de inserir comentário.

3.4.5 Classificar um Ponto / Evento

Esta funcionalidade está apenas implementada no *Web Portal* e funciona exclusivamente para os pontos. Embora a aplicação *Android* também a devesse contemplar nesta fase inicial, ainda não se encontra desenvolvido o código para tal.

No *Web Portal* para dar início à funcionalidade é necessário que o utilizador esteja a visualizar a página de detalhe de um ponto, onde são visíveis cinco estrelas, que correspondem à média das pontuações atribuídas. Na Figura 90 podemos ver com mais pormenor uma pontuação atribuída a um ponto.

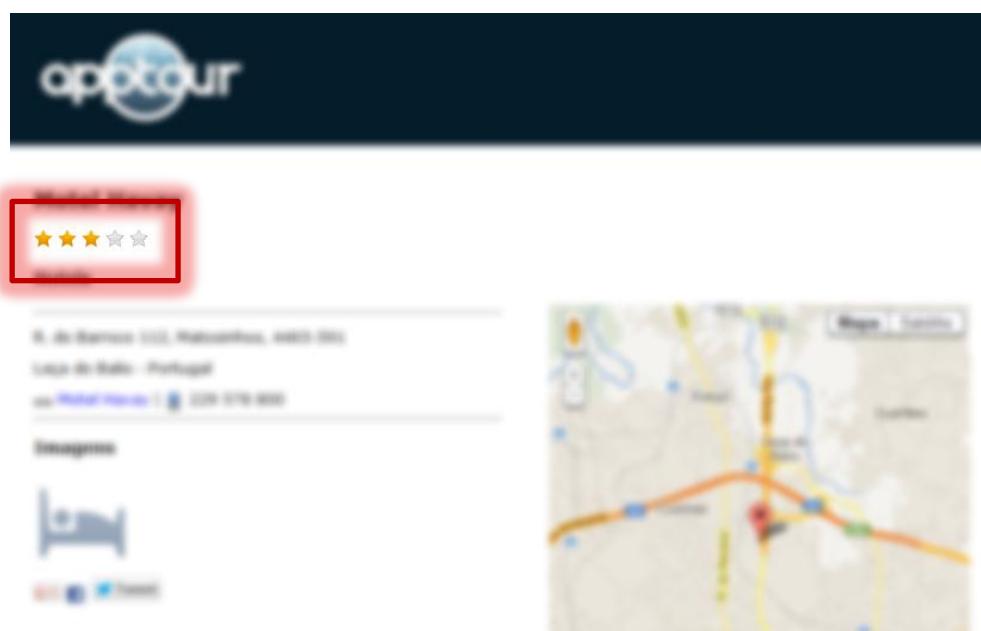
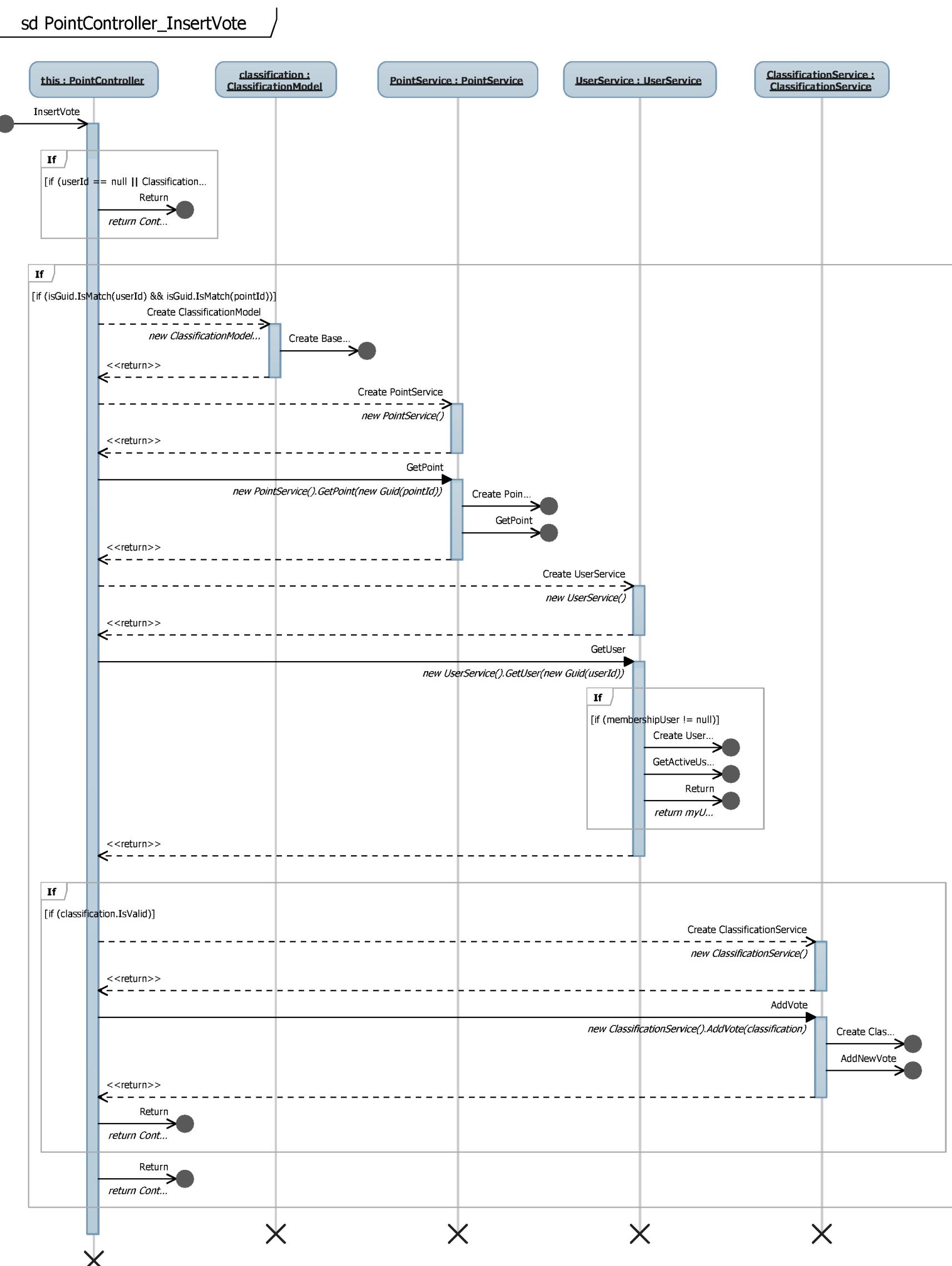


Figura 90. Classificação de um Ponto

Para um utilizador poder classificar um ponto, bastará passar com o rato sobre essas mesmas “estrelas” e caso ainda não tenha efectuado uma classificação sobre este ponto anteriormente, estas estarão desbloqueadas, o que permitirão a atribuição de um valor. Esse valor será assim enviado pelo controlador *PointController* para o serviço *ClassificationService*. Toda a sequência até ao serviço pode ser visualizada na Figura 91. A sequência completa pode ser vista no Anexo 3.

Figura 91. Diagrama Sequência *InsertVote*

3.4.6 Denunciar um comentário

Esta funcionalidade permite a um utilizador denunciar comentários ilícitos ou menos próprios para um ponto ou evento. A funcionalidade pode ser activada pelo botão destacado na Figura 92, que faz um pedido AJAX a um método no *PointController* onde vai alterar um campo do registo do comentário, do respectivo ponto.

The screenshot shows a web-based application interface. At the top left, there's a section titled "Comentários" (Comments) containing four entries:

- Ricardo Carneiro** 04/06/2012 00:21:59: Hello World!!
- Ricardo Carneiro** 04/06/2012 00:32:28: Another one bits the dust...
- Ricardo Carneiro** 04/06/2012 00:33:47: ...and another one bits, another one bits... another bits the dust...!! DUN DUN DUN!!

Each comment entry has a small square button with a white exclamation mark icon to its right. The first button is highlighted with a red rectangular box. Below the comments, there are two input fields: "Utilizador" (User) containing "Ricardo Carneiro" and "Message" (Message) which is currently empty. At the bottom left of the form is a "Novo" (New) button.

Figura 92. Botão de denunciar um comentário

Ao denunciar um comentário o mesmo não irá aparecer no detalhe do ponto, e será necessário uma intervenção do gestor de conteúdos ou do administrador que terá de decidir o que fazer com a denúncia.

3.4.7 Criar um novo Perfil de Pesquisa

Componentes que despoletam a funcionalidade:

- *Android App*
- *Web Portal*

Android App

Na aplicação *Android*, no botão “*Set Filters*” do menu de topo (ver Figura 93), é possível aceder à *activity* que permite controlar os perfis de pesquisa.

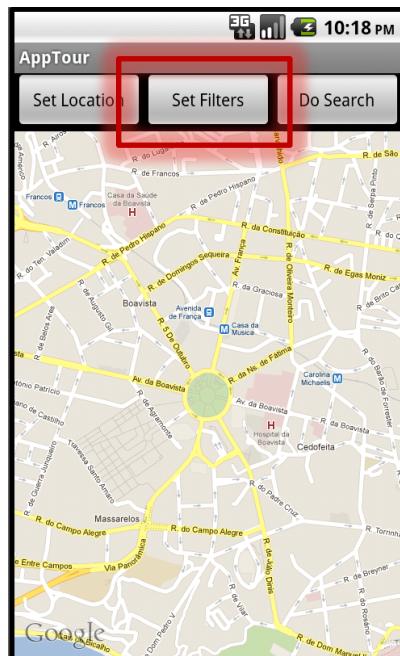


Figura 93. Activar vista de perfis de pesquisa

Uma vez clicado esse botão será aberto uma *Activity* que permite a definição de um perfil de pesquisa tal como se pode ver na Figura 94.

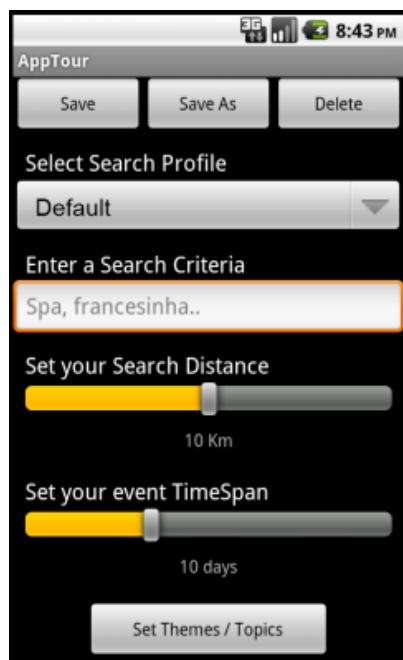


Figura 94. Definir perfis de pesquisa em Android

Este ecrã mostra os perfis de pesquisa que existem associados ao utilizador no *spinner (combobox)*, onde o utilizador pode escolher o que quer activar. Para a criação de um novo perfil de pesquisa basta definir os critérios a associar a esse perfil e premir o botão “Save As” que irá abrir uma janela onde será pedido ao utilizador que forneça o nome a associar a esse mesmo perfil a criar.

Neste caso o *Android* irá aceder ao método *SearchProfileNew()* fornecido pelo *Web Service*, que recebe por parâmetros todos os dados do perfil a gravar e devolve um *Guid* respectivo do perfil criado, usando para isso o serviço respectivo.

Web Portal

Para criar um novo perfil de pesquisa no *Web Portal* é necessário estar autenticado e no menu de topo ir ao seu perfil, conforme se pode ver na Figura 95.

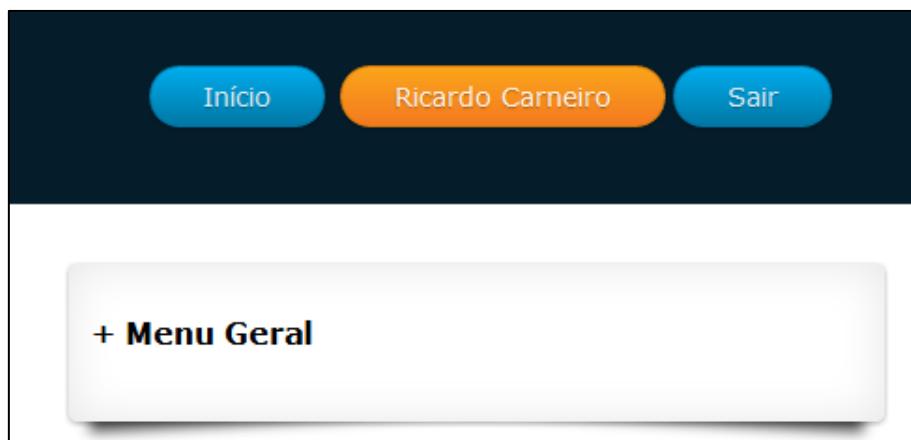


Figura 95. Menu de Topo do Web Portal

No separador *Pesquisas* tem uma listagem dos perfis que tem com os seus detalhes, conforme a Figura 96. Ao carregar no botão “Criar Novo Perfil Pesquisa” irá aparecer uma nova vista onde poderá definir os campos respectivos para detalhar o seu perfil de pesquisa. Do lado direito da página aparece uma listagem dos tópicos onde os pontos irão ser filtrados.

Nome	Distância dos Pontos (KM)	Tempo de Eventos (dias)	Critério de Pesquisa	Tópicos	Activo	
Pesquisa	32	2	Av	0	X	
Default	12	2		0	✓	

Figura 96. Separador de Pesquisas no Perfil de Utilizador

O código no *SearchProfileController* que permite a criação de um *SearchProfile* utiliza modelos intermédios para os tópicos, denominados *TopicsViewModel*, para permitir manipular os valores na vista, utilizando os controlos do *Razor Engine*, sem influenciar as regras de negócio, conforme se pode ver na Figura 97.

```
public ActionResult Create()
{
    IList<ThemeModel> temas = new ThemeService().GetActiveThemes();
    IList<TopicsViewModel> _topics = new List<TopicsViewModel>();

    IList<TopicModel> topics = new TopicService().GetActiveTopics()
        .OrderBy(x => x.Theme.Name)
        .ToList();

    topics.ToList().ForEach(x => _topics.Add(new TopicsViewModel
    {
        Name = x.Name,
        Check = false,
        Id = x.Id,
        ThemeId = x.Theme.Id,
        ThemeName = x.Theme.Name
    }));
}

SearchProfileModel model = new SearchProfileModel
{
    SearchProfileCheckboxes = _topics
};

return View(model);
}
```

Figura 97. Método *Create* do *SearchProfileController*

A sequência desta funcionalidade desde que é invocado o serviço, pode ser vista em Anexo 4 devido às grandes dimensões do documento.

3.4.8 Configurar um Perfil de Pesquisa

Componentes que despoletam a funcionalidade:

- *Android App*
- *Web Portal*

Android App

No *Android* a configuração de um perfil de pesquisa segue os mesmos passos que a criação de um novo perfil (3.4.7), porém a diferença passa no botão que o utilizador vai premir. Assim em vez de se usar o botão “Save As” deverá premir o botão “Save”.

A nível de comunicação com o sistema, muda o método chamado ao *Web Service*, que passa a ser o *SearchProfileSave()*.

Web Portal

No menu de perfil de utilizador, no separador de *Pesquisa*, é possível editar um perfil de pesquisa que se pretenda. Para tal, basta seleccionar o botão que tem o ícone *Editar* conforme se pode ver na Figura 96.

Ao carregar no botão editar é enviado um pedido ao *SearchProfileController* para o metodo *Edit*, com o *Guid* do *SearchProfile* seleccionado. Este faz um pedido aos serviços para devolver o respectivo *SearchProfile* baseado no *Guid* e faz a renderização da página para editar os dados. Isto é um procedimento normal em qualquer página de edição de uma tabela da base de dados.

Depois dos dados editados é efectuado um pedido *POST* ao método *Edit* com o respectivo modelo preenchido. Este modelo passa pelo processo de validação e é enviado ao serviço para actualização.

Este procedimento é válido para qualquer operação de edição em todos os controladores *MVC*, variando apenas alguns procedimentos, dentro do controlador.

3.4.9 Eliminar um perfil de pesquisa

Esta funcionalidade deverá estar activa tanto no *Android* como no *Web Portal*, porém não se encontra ainda implementada do lado do servidor. No entanto já se encontra previsto o interface do lado do *Android*, tal como se pode ver na Figura 98, onde apenas é necessário premir o botão “Delete”, o que vai despoletar uma chamada a um método do *Web Service* que vai eliminar o perfil seleccionado no *spinner*, o que no caso da figura é o perfil com o nome “Default”.

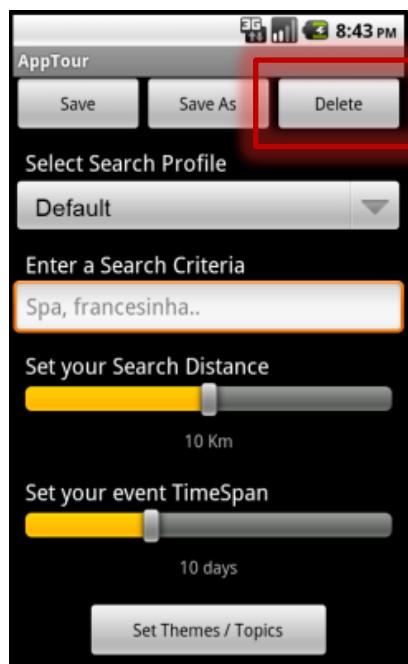


Figura 98. Eliminar um perfil de pesquisa no Android

No entanto, ainda não foi implementado nem o método do *Web Service*, que recebe esse mesmo pedido, nem mesmo as implementações necessárias a nível de serviços, modelos ou mesmo repositório para contemplar este caso.

Este desenvolvimento não está limitado por nenhum impedimento que não o tempo disponível, já que todas as condições necessárias para a sua implementação estão garantidas.

3.4.10 Calendarizar Eventos

Esta funcionalidade não se encontra implementada e como tal é descrita com mais pormenor nos capítulos 4.1 e 4.2.

3.4.11 Publicar Pontos e Eventos em redes sociais

A publicação de pontos ou eventos em redes sociais é uma funcionalidade que não requer tratamento especial da parte do sistema. Devido ao facto de ainda não ser possível a visualização de eventos, não é assim possível efectuar essa mesma partilha, pelo que se encontra limitada aos pontos.

Para poder efectuar essa partilha bastará carregar sobre o ícone da rede social respectiva, que se encontra disponível na vista de detalhe do ponto, como se pode ver na Figura 99.

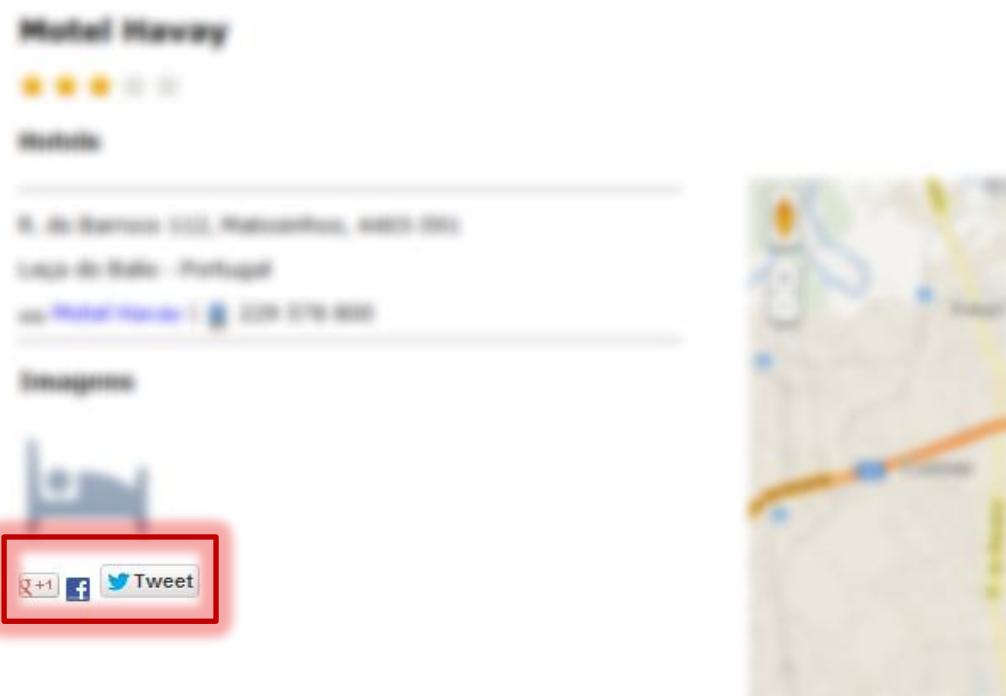


Figura 99. Partilha de Pontos em redes sociais

Não existe nenhuma limitação acerca do número de partilhas efectuadas. Essa responsabilidade cabe exclusivamente às redes sociais que disponibilizam esses mesmos serviços.

A publicação desses mesmos Pontos através das redes sociais resume-se à implementação de pequenos scripts que são incluídos na vista do detalhe dos Pontos. Podemos ver essa mesma implementação na Figura 100. A implementação

desses scripts seguiu as indicações fornecidas por essas mesmas redes sociais, tal como se pode ver nas páginas oficiais do *Facebook*²⁶, *Google+*²⁷ e *Twitter*²⁸.

```
<script src="http://static.ak.fbcdn.net/connect.php/js/FB.Share" type="text/javascript">
</script>

<script type="text/javascript">
    function doNoting() {
        return false;
    }

    (function () {
        var po = document.createElement('script'); po.type = 'text/javascript'; po.async = true;
        po.src = 'https://apis.google.com/js/plusone.js';
        var s = document.getElementsByTagName('script')[0]; s.parentNode.insertBefore(po, s);
    })();
</script>

<g:plusone size="small" annotation="none"></g:plusone>
<a name="fb_share" type="icon" share_url="@Request.Url.ToString()/Point/Detail/@Model.Id">
    Partilhar</a <a href="https://twitter.com/share" class="twitter-share-button" data-text="@Model.Name - @Model.Address"
    data-count="none" data-hashtags="apptour">Tweet</a>
<script type="text/javascript">
    !function (d, s, id) { var js, fjs = d.getElementsByTagName(s)[0]; if (!d.get
```

Figura 100. Exemplo dos scripts que permitem a partilha em redes sociais

3.4.12 Resolver denúncias

Esta funcionalidade não se encontra implementada e como tal é descrita com mais pormenor nos capítulos 4.1 e 4.2.

²⁶ <http://developers.facebook.com/docs/share/>

²⁷ <https://developers.google.com/+/plugins/share/>

²⁸ <https://dev.twitter.com/docs/tweet-button>

3.4.13 Inserir Pontos e Eventos

Esta funcionalidade, conforme explicado detalhadamente no capítulo 4.2, apenas está implementado para os pontos.

No formulário de inserção de ponto, através do *Web Portal* (Figura 101), é possível preencher todos os campos do ponto. Este formulário é permitido a todos os utilizadores, desde que autenticados no *Web Portal*. Isto é garantido através do atributo disponibilizado pela *framework .NET*, denominado *Authorize*²⁹.

Figura 101. Exemplo de interface de inserção de novo ponto.

Depois de preenchidos os campos do formulário a aplicação invoca o método *Create* do *PointController*, com o atributo *POST*. Como se pode ver na Figura 102, o método faz a conversão do modelo de aplicação *PointViewModel* para o modelo de negócio *PointModel*.

²⁹ <http://msdn.microsoft.com/en-us/library/system.web.mvc.authorizeattribute.aspx>

A utilização deste modelo intermédio é necessária para conseguir utilizar propriedades adaptadas na minha aplicação *MVC* num controlo *ListBox*, que devolve um *array de strings* que corresponde aos *Guid* dos tópicos seleccionados.

Como isto não existe no modelo de negócio (*PointModel*), foi necessário esta adaptação, para contornar essa limitação. Possivelmente existirá outro tipo de abordagem, porém esta implementação respeita as regras definidas pelo *ASP .NET MVC 3*.

```
[Authorize]
[HttpPost]
public ActionResult Create(PointViewModel point)
{
    if (ModelState.IsValid)
    {
        PointModel model = new PointModel
        {
            Id = point.Id != Guid.Empty ? point.Id : Guid.NewGuid(),
            Name = point.Name,
            Address = point.Address,
            PostalCode = point.PostalCode,
            City = point.City,
            Coordenate = point.Coordenate,
            PhoneNumber = point.PhoneNumber,
            URL = point.URL,
            SourceURL = point.SourceURL,
            IsActive = point.IsActive
        };

        if (point.SelectedTopicId != null && point.SelectedTopicId.Length > 0)
        {
            model.Topics = new List<TopicModel>();
            for (int i = 0; i < point.SelectedTopicId.Length; i++)
            {
                TopicModel temp = new TopicService().GetTopic(new Guid(point.SelectedTopicId[i].ToString()));
                model.Topics.Add(temp);
            }
        }
    }

    Validate Model After Conversion from ViewModel
}

ViewBag.Topics = new MultiSelectList(new TopicService().GetActiveTopics(), "Id", "Name");
ViewBag.City = new SelectList(new CityService().GetCities(), "Id", "Name");

return View(point);
}
#endregion + ActionResult Create
```

Figura 102. Código do pedido *POST* de inserção de um ponto.

Depois de convertido para o modelo de negócio, este é validado e é invocado o método *InsertPoint* do serviço *PointService* que é responsável pela operação de inserção de um ponto. O respectivo Diagrama de Sequência pode ser visualizado no Anexo 8.

3.4.14 Alterar conteúdos em Pontos e Eventos

A funcionalidade de alterar as informações dos pontos está limitada aos Gestores de Conteúdos (e futuramente ao criador do ponto – ver capítulo 4.2). O formulário de edição (ver Figura 103) é em tudo semelhante ao de inserção, excepto o pormenor de um botão “Lista de Atributos”. Este permite criar informação adicional não contemplada no formulário original.

Dados do Ponto

Nome
ISEP

Endereço
Rua Dr. António Bernardino de Almeida 431, 4200-072 Porto

Código Postal
4200-072

Cidade
Porto

Coordenada
41,177863,-8,608292

N.º Telefone
228 340 500

URL
<http://www.isep.ipp.pt/>

Fonte URL
<https://maps.googleapis.com/maps/api/place/details/json?reference=>

Topicos

- Cinema & Teatro
- Comboios
- Concertos & Espectáculos
- Educação**

Activo

Guardar **Lista de Atributos**

Figura 103. Formulário de edição de um ponto.

Ao carregar no botão Lista de Atributos é direcionado para uma página onde contém os atributos do respectivo ponto.

Nome	Endereço	Código Postal	Cidade	Coordenada	N.º Telefone	URL	Fonte URL	Atributos	Topics
ISEP	Rua Dr. António Bernardino de Almeida 431, 4200-072 Porto	4200-072	Porto	41,177863,-8,608292	228 340 500			2	1

Voltar ao Ponto

Atributos - ISEP

+ Adicionar Atributos

Chave	Valor	Tipo de Dados	Activo	Traduções
Engenharia Informática	True	System.Boolean		0
Engenharia Mecânica	True	System.Boolean		0

Figura 104. Página de Atributos do ponto ISEP.

Aí é possível inserir e editar atributos através de uma caixa de diálogo, construída com base em *jQuery*, como se pode ver na Figura 105.



Figura 105. Formulário de Atributos

Estas funcionalidades utilizam pedidos *AJAX* para processar os dados e redirecciona-los para o controlador onde é efectuado o pedido ao serviço de atributos.

Como se pode ver na Figura 106 o valor recebido é validado e guardado conforme o seu tipo de dados. A integridade desses dados é garantida por *jQuery*, que valida os valores conforme o tipo de dados seleccionado. Implementações

futuras poderiam alterar o controlo disponibilizado na caixa de diálogo para recolha desse valor, para desta forma ter melhor percepção a nível do utilizador, assim como garantir à partida, de uma forma gráfica, os dados recolhidos.

```

public ActionResult InsertAttribute(string PointId, string attrkey,
                                    string attrvalue, string attrtype, string active)
{
    if (PointId == null || PointId.Equals(string.Empty) || attrkey == null
        || attrkey.Equals(string.Empty) || attrvalue == null
        || attrvalue.Equals(string.Empty) || attrtype == null
        || attrtype.Equals(string.Empty))
        return Content(ViewRes.Attribute.MissingParams);
    try
    {
        bool isActive = (active != null ? true : false);
        PointAttributeModel model = new PointAttributeModel
        {
            Id = Guid.NewGuid(),
            Point = new PointService().GetPoint(new Guid(PointId)),
            KeyPair = attrkey,
            Value_bool = (attrtype.Equals("System.Boolean")
                ? (attrvalue.ToLower().Equals("true")
                    || attrvalue.ToLower().Equals("1") ? true : false)
                : (bool?)null),
            Value_number = (attrtype.Equals("System.Decimal")
                ? Convert.ToDecimal(attrvalue, CultureInfo.GetCultureInfo("en-US"))
                : (Decimal?)null),
            Value_Date = (attrtype.Equals("System.DateTime")
                ? DateTime.Parse(attrvalue) : (DateTime?)null),
            Value_string = (attrtype.Equals("System.String") ? attrvalue : null),
            Value_Type = attrtype,
            IsActive = isActive
        };
        Guid id = service.InsertAttribute(model);
        return Content("Sucesso");
    }
    catch (Exception e)
    {
        if (e.InnerException != null)
            return Content(e.InnerException.Message);
        return Content(e.Message);
    }
}

```

Figura 106. Método *InsertAttribute* do controlador *AttributeController*

Na seguinte figura, pode-se ver o diagrama de sequência da inserção de um atributo.

A edição de um ponto é idêntica ao funcionamento da inserção de um ponto, com a diferença do método a ser invocado pelo serviço, ser o método *UpdatePoint*. O diagrama de sequência relativa a este método está descrito no Anexo 9.

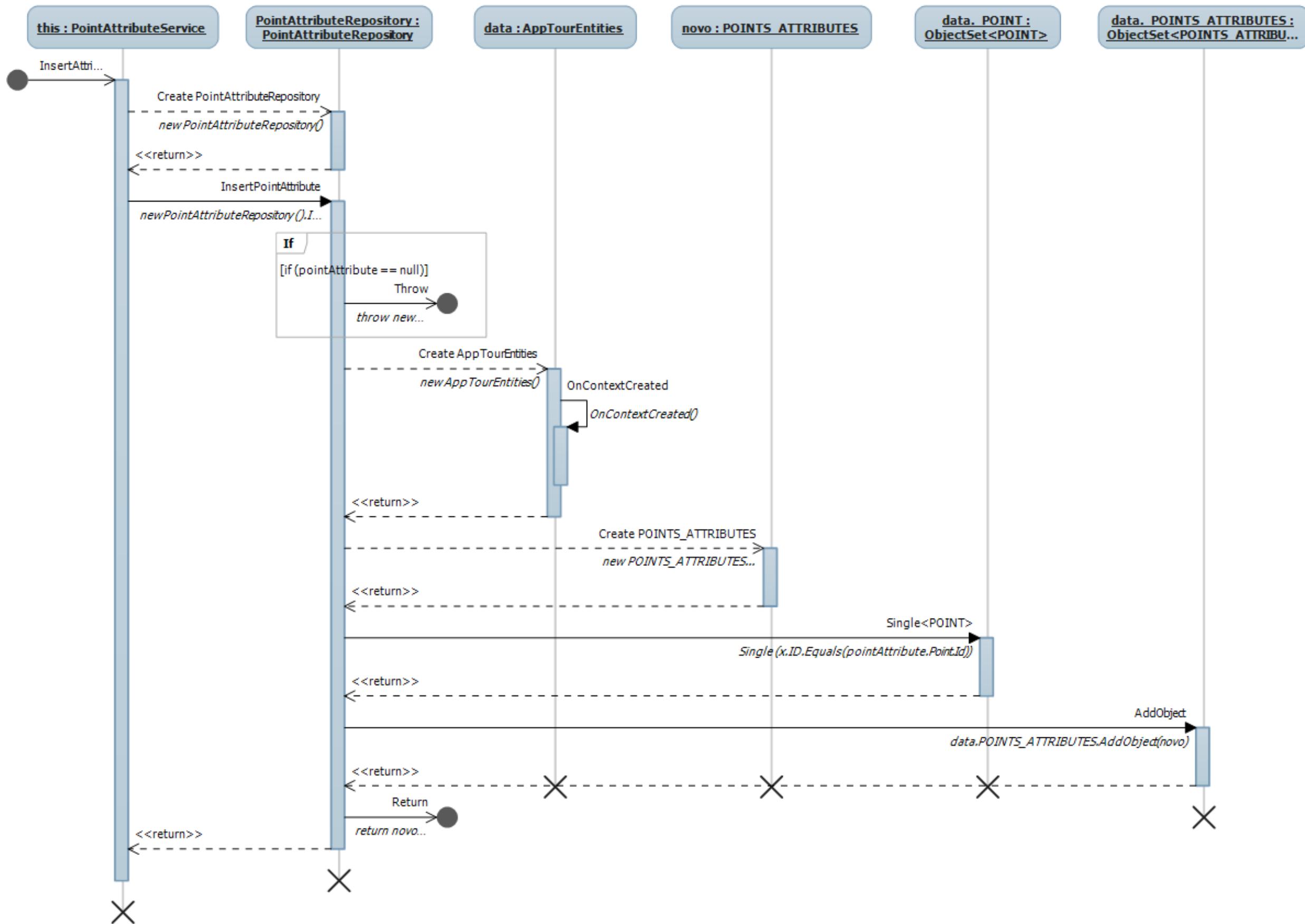


Figura 107. Diagrama de Sequência InsertAttribute

3.4.15 Alterar permissões Utilizador

A gestão de utilizadores foi delegada, num lançamento inicial da aplicação, para o *membership* da Microsoft. Embora esta não seja a solução ideal, serve os propósitos necessários para uma primeira *release*, evitando assim um desenvolvimento mais extensivo de uma situação que se encontra controlada à partida. Assim e para efectuar alterações de permissão de utilizadores, bastará lançar o *membership* o que abrirá no *browser* uma página tal como se pode ver na Figura 108.

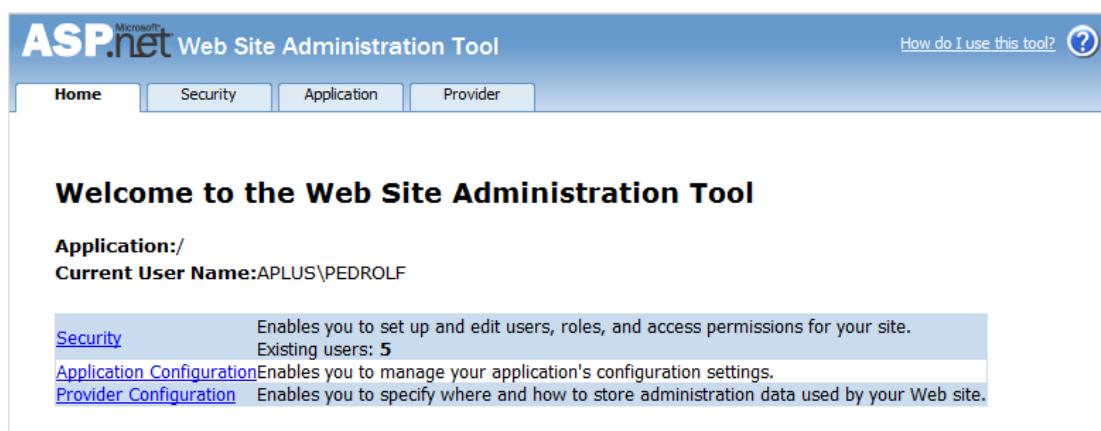


Figura 108. Membership da Microsoft

Acedendo à secção *Security* -> *Create or Manage Roles*, abrimos uma página com os vários perfis de utilizador disponíveis. Escolhendo a opção *Manage* (ver Figura 109) no perfil determinado, é possível atribuir ou retirar utilizadores desses mesmos perfis.

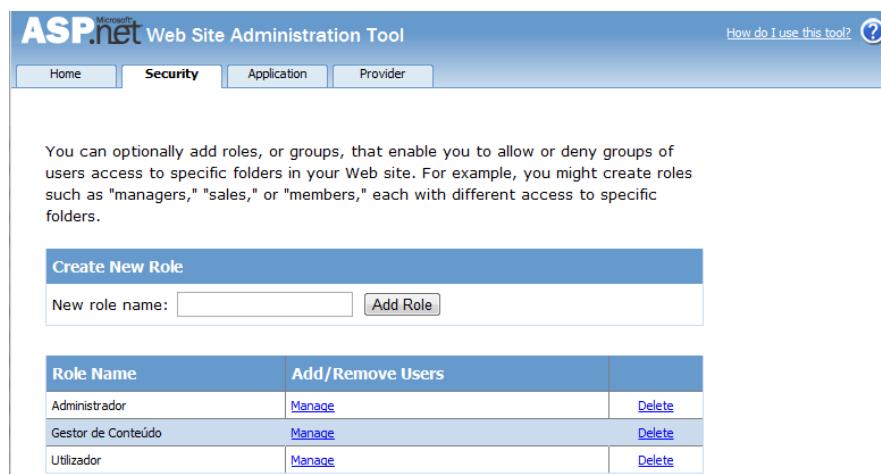


Figura 109. Gestão de *Roles* no Membership

3.4.16 Bloquear e desbloquear utilizadores

Tal como mencionado no ponto anterior, a gestão de utilizadores ficou delegada, numa fase inicial, para o *Membership* da *Microsoft*. Embora não uma solução ideal, esta aproximação é suficiente para a satisfação desta funcionalidade.

Assim para bloquear e desbloquear utilizadores, basta aceder ao *Membership* e através da opção *Security -> Manage Users* o que nos disponibiliza a página que podemos ver na Figura 110.

Active	User name			Roles
<input checked="" type="checkbox"/>	carneiro	Edit user	Delete user	Edit roles
<input checked="" type="checkbox"/>	joaquim	Edit user	Delete user	Edit roles
<input checked="" type="checkbox"/>	moura	Edit user	Delete user	Edit roles
<input checked="" type="checkbox"/>	pedroluisf	Edit user	Delete user	Edit roles
<input checked="" type="checkbox"/>	rjcarneiro	Edit user	Delete user	Edit roles

Figura 110. Bloquear e Desbloquear Utilizadores no *Membership*

Nesta página podemos observar uma listagem com os utilizadores disponíveis na aplicação, e na coluna mais à esquerda (coluna *Active*), podemos bloquear ou desbloquear um utilizador apenas por despistar ou picar a *checkbox* respectiva.

3.4.17 Procurar e resolver Pontos e Eventos duplicados

Esta funcionalidade depende da obtenção de Pontos de uma forma que permita saber se existem duplicados. Essa obtenção dessas possíveis duplicações está muito dependente da forma como a informação relativa a Pontos e Eventos é mostrada aos gestores de conteúdo e/ou Administradores.

No ponto actual de desenvolvimento da aplicação, é possível a edição de informação relativa a um Ponto (ver 3.4.14), assim como é possível inactivar esse

mesmo Ponto, o que significa que a tarefa de manutenção dos Pontos é efectivamente possível.

No entanto para se saber da existência de Pontos repetidos, é necessário que existam várias formas de os pesquisar. Listagens com diversas ordenações permitem assim uma maior capacidade de dar com esses mesmos Pontos, porém estas ainda não se encontram implementadas. Assim considera-se que este caso de uso não está implementado na sua totalidade.

3.4.18 Recolha de informação automática

A funcionalidade de recolha automática de informação corresponde à tarefa a executar pelos Agentes de importação.

Tal como descrito antes, o processo pode ser iniciado de distintas formas, uma das quais através do agendamento com o uso do *Windows Task Scheduler*, ferramenta fornecida com o Sistema Operativo da *Microsoft*.

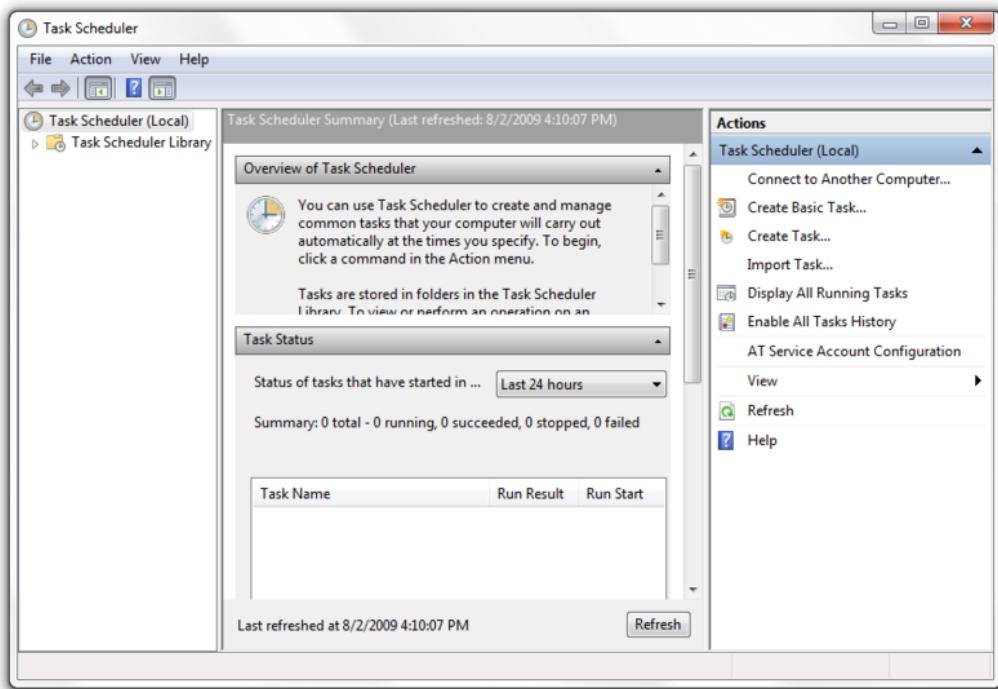


Figura 111. Windows Task Scheduler³⁰

³⁰ http://en.wikipedia.org/wiki/Windows_Task_Scheduler

Esta ferramenta permite definir quando se pretende iniciar um determinado programa, estipulando assim uma data/hora.

O programa a ser agendado, pode ser tanto uma aplicação *Winforms*, como uma *Console Application*, sendo que ambas utilizam uma *Class Library* denominada *AgentsService*. Essa *Class library* tem a estrutura que se pode ver na Figura 112.

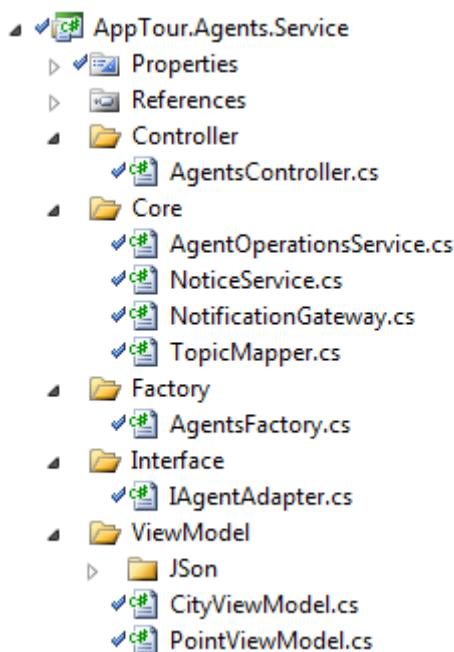


Figura 112. Estrutura do projecto *AgentsService*

AgentsController

O *AgentsController* é uma classe que faz parte do *AgentsService*, que tem como função dar início e controlar a gravação de estado no final de cada processo de importação de cada Agente. Na Figura 113 podemos verificar através de um diagrama de sequência o processo utilizado para saber dinamicamente quais os diversos Agentes que efectuarão os processos de importação de dados das diversas *APIs*. Esse mesmo diagrama está incluído no Anexo 6 com um detalhe melhorado, para uma melhor apreciação do procedimento descrito.

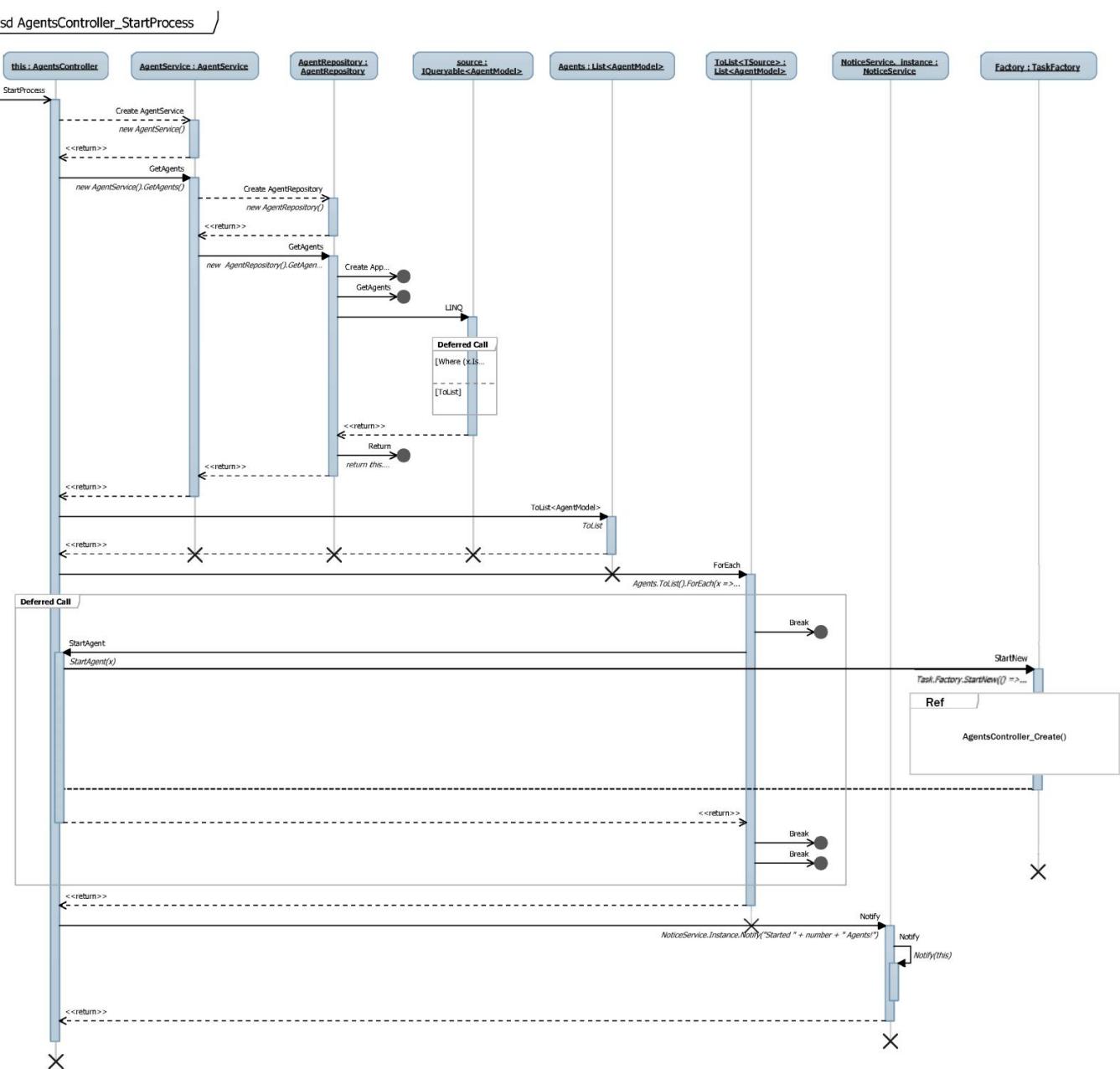


Figura 113. Início do processo de importação de dados pelos Agentes

No diagrama anterior podemos observar o processo de leitura dos diversos Agentes usando o serviço disponibilizado pela arquitectura (*AppTour.Business.Services.AgentsService*) que fornece uma lista de Agentes definidos na base de dados.

Esta listagem, uma vez obtida, irá dar origem a uma instância por registo desses mesmos Agentes, desde que estes cumpram as condições de periodicidade e limites impostos pelas próprias APIs. Esta instância é feita através da classe *AgentsFactory* que tem como função apenas essa criação. Ver Figura 114.

Em ambos os casos, os processos enviam sempre notificações das acções realizadas através do serviço de notificações.

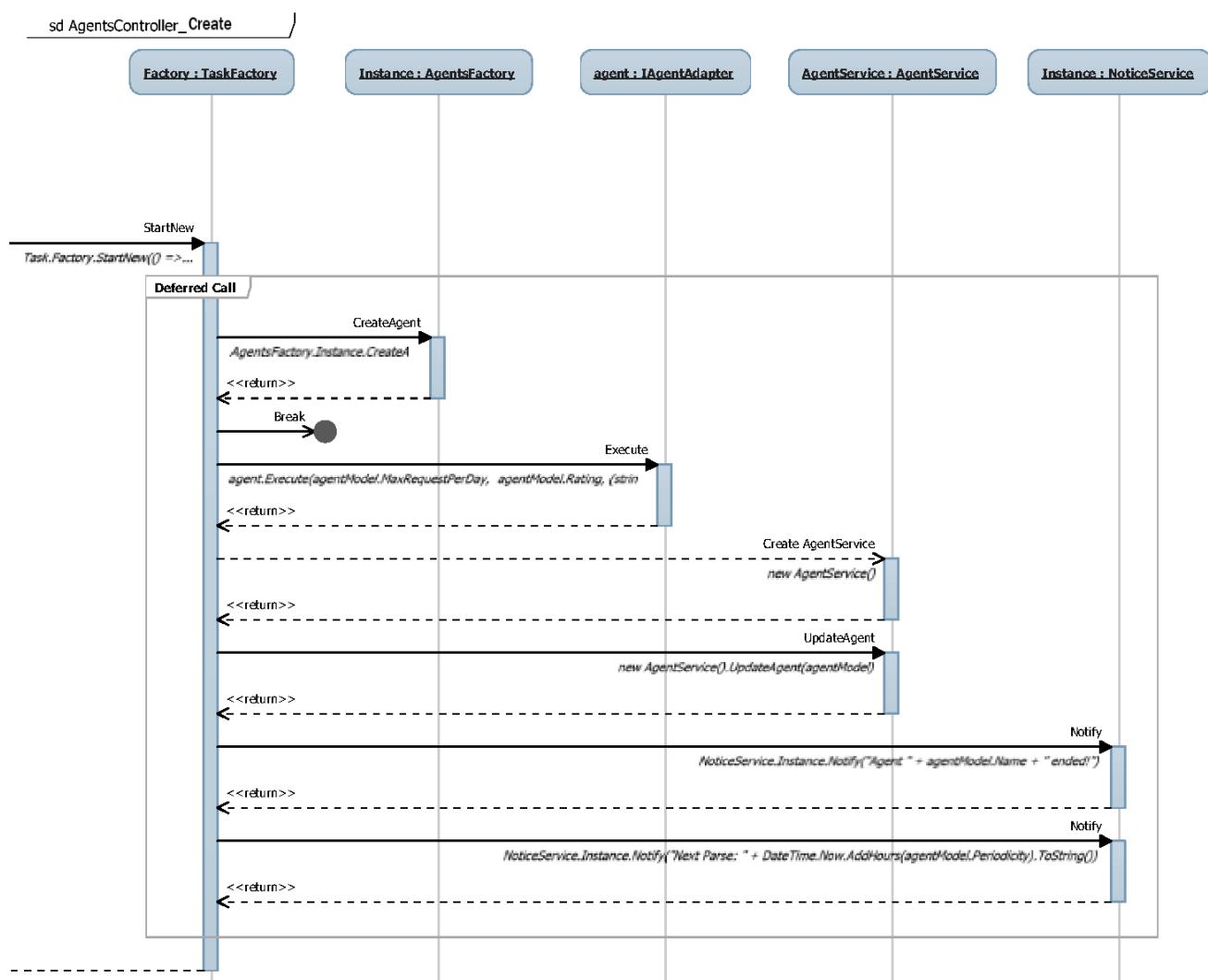


Figura 114. Processo de Criação de Agentes

Depois de instanciados todos os Agentes necessários, o *AgentsController* dá início ao processo de migração em cada um deles através do método de interface (*Execute*), sendo que cada um destes processos corre numa *Task* separada para maximização de desempenho.

No final de execução de cada um destes processos, é guardada uma referência ao ponto onde esta importação ficou, assim como a data de execução desse mesmo Agente, no modelo respectivo, e persistido na Base de Dados. Dessa forma é assim possível retomar a importação, numa altura futura.

Agentes

Cada um dos Agentes iniciados inicia um processo de importação de dados que pode ser completamente distinto de qualquer outro Agente. A lógica inerente a cada *API* consultada rege a forma como este Agente se irá comportar. Porém alguns pontos em comum se podem encontrar em todos os Agentes que merecem alguma referência. Assim podemos dizer que cada um desses Agentes terá de efectuar funções de autenticação, verificação de condições de operabilidade, leitura e recolha de dados.

Um exemplo da implementação de um desses agentes pode ser encontrado no Anexo 5, que corresponde ao Agente utilizado para comunicação com o *Google Locals*.

Neste caso a política a usar para esta *API* passou pela recolha de determinados pontos que se encontrassem à volta de uma zona específica na cidade, e com essa mesma lista, efectuar um pedido dos detalhes desses mesmos Pontos. Só então com esse detalhe era possível persistir a informação.

A forma como essa zona específica era definida, correspondia à leitura de um ficheiro *CSV* que continha uma lista de códigos postais, e que depois de lidos os pontos de um determinado código postal, seria passado ao seguinte. Este método oferece resultados satisfatórios, já que é possível a mudança da ordem de leitura dessas mesmas zonas, através da simples edição de um ficheiro de texto.

3.4.19 *Classificação e armazenagem de informação*

Este caso de uso corresponde ao tratamento dado aos dados lidos das *APIs* através da importação feita pelos Agentes. Como cada uma dessas *APIs* consultadas podem fornecer informação completamente dispare umas das outras, existe uma necessidade empírica de homogeneizar essa mesma informação. Depois de recolhida e tratada, existe a necessidade de a persistir na Base de Dados para que o processo de importação se torne assim definitivo. Não estão implementadas ainda nenhuma

importação de Eventos, pelo que esses não constarão da explicação disponibilizada em seguida.

Os modelos usados no sistema reflectem as entidades no seu estado final, já devidamente validados com as regras de negócio respectivas. No entanto os dados recolhidos podem não obedecer a estas mesmas regras, pelo menos enquanto a informação não é homogeneizada. Assim existe a necessidade de criação de modelos temporários, apelidados de *xxxViewModel*, cujo principal objectivo é servir de *containers* à informação lida pelas *APIs*.

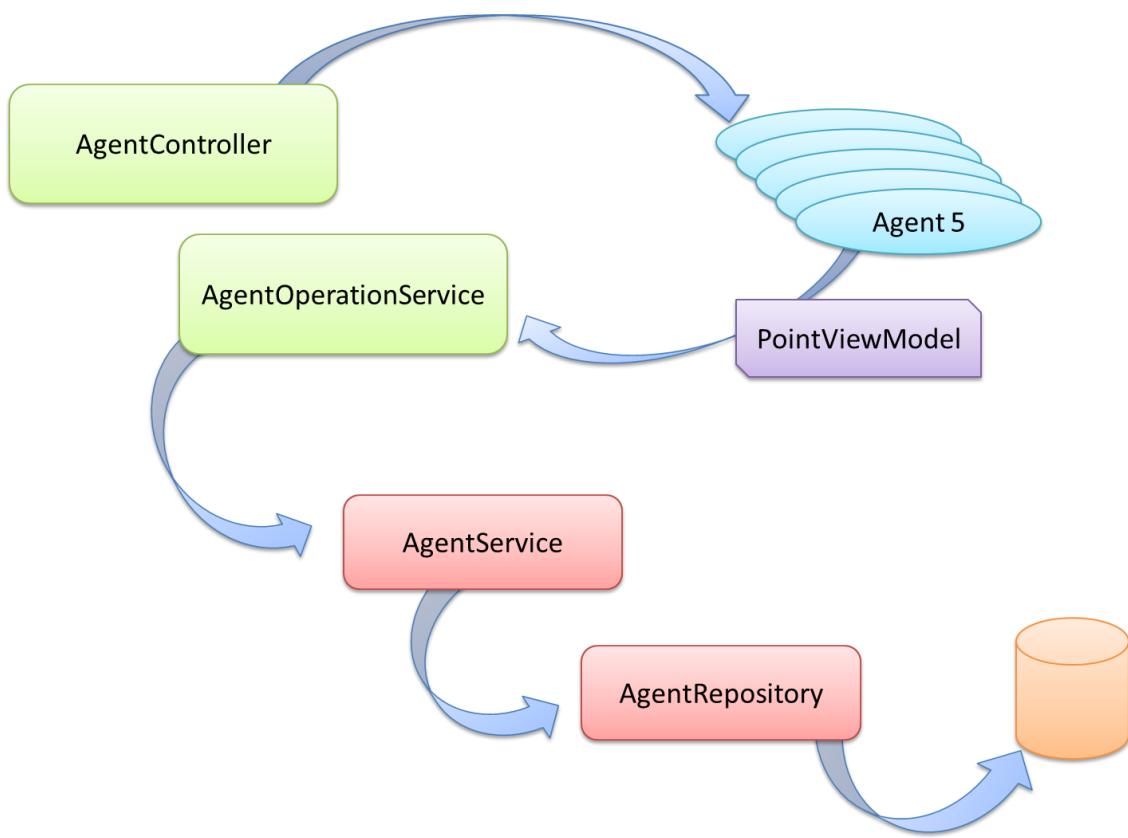


Figura 115. Esquema do processo de gravação de informação recolhida pelos Agentes.

No processo de recolha dos Pontos a gravar pelas *APIs* é necessário haver uma triagem sobre a informação recolhida. Isto porque não existem garantias que os Pontos lidos são efectivamente de interesse, já que Pontos com informação insuficiente como Códigos Postais inexistentes, Cidades não referenciadas, Pontos sem nome, ou mesmo Pontos que contenham informação insuficiente ou errónea, tal como por exemplo pontos sem uma classificação (tópico) associado, podem e devem ser descartados.

AgentOperationsService

Com a informação recolhida pelas *APIs* na nossa posse, torna-se necessária a homogeneização e mapeamento para os modelos do sistema de modo a ser possível a sua persistência. Assim para cada Ponto recolhido pelos Agentes é enviado ao *AgentOperationsService* que efectua esses importantes passos.

Antes porém de iniciar esses dois passos, existe ainda uma verificação final a ser feita, que consiste na detecção de possíveis pontos duplicados. Tal é necessário para tentar minimizar a tarefa dos gestores de conteúdos em detectar e corrigir essas situações. Essa operação é também efectuada no *AgentOperationsService*.

A verificação dessa possível duplicação foi feita inicialmente contra os pontos guardados na Base de Dados, porém este processo tornava-se muito morosa, já que para cada ponto seria necessário importar todos os já gravados e efectuar assim as verificações necessárias. Assim foi decidido que seriam carregados primeiramente para memória, todos esses pontos, e seguidamente seriam consultados esses mesmos pontos através de listas. Um factor importante a ter em conta passa pelo uso de *concurrentBags*³¹ para esse efeito, já que dessa forma podem ser aproveitadas potencialidade de paralelismo tanto nas pesquisas, como na gravação dos pontos recentemente persistidos na Base de Dados nessas mesmas listas de pesquisa.

Na parte de homogeneização de informação existem dois aspectos a considerar:

O primeiro passa pela validação dos códigos postais obtidos e se estes correspondem à cidade em questão. Para isso foram utilizados *Web Services* externos que devolvem uma localidade através do envio de um código postal.

O segundo, passa pela verificação da relevância dos tópicos desse Ponto para o nosso sistema. Não interessa efectuar a importação de um ponto que não esteja contemplado em nenhum dos tópicos da solução. Assim e tendo em conta que existe

³¹ <http://msdn.microsoft.com/en-us/library/dd381779.aspx>

uma infinidade de tópicos que possam existir em cada *API*, houve a necessidade de criar uma espécie de dicionário que relacionasse os tópicos existentes com as várias possibilidades encontradas. Esse “dicionário” pode ser encontrado na classe *TopicsMapper*.

A persistência dos dados recolhidos, depois de devidamente tratados é feita simplesmente pela invocação dos serviços da arquitectura, nomeadamente do serviço *PointService*, onde é enviado o *PointModel* entretanto preenchido e validado para inserção. Depois de invocado o serviço é notificado o acto de sucesso ou insucesso dessa mesma persistência.

Para uma melhor visualização deste processo recomenda-se a visualização do Anexo 7.

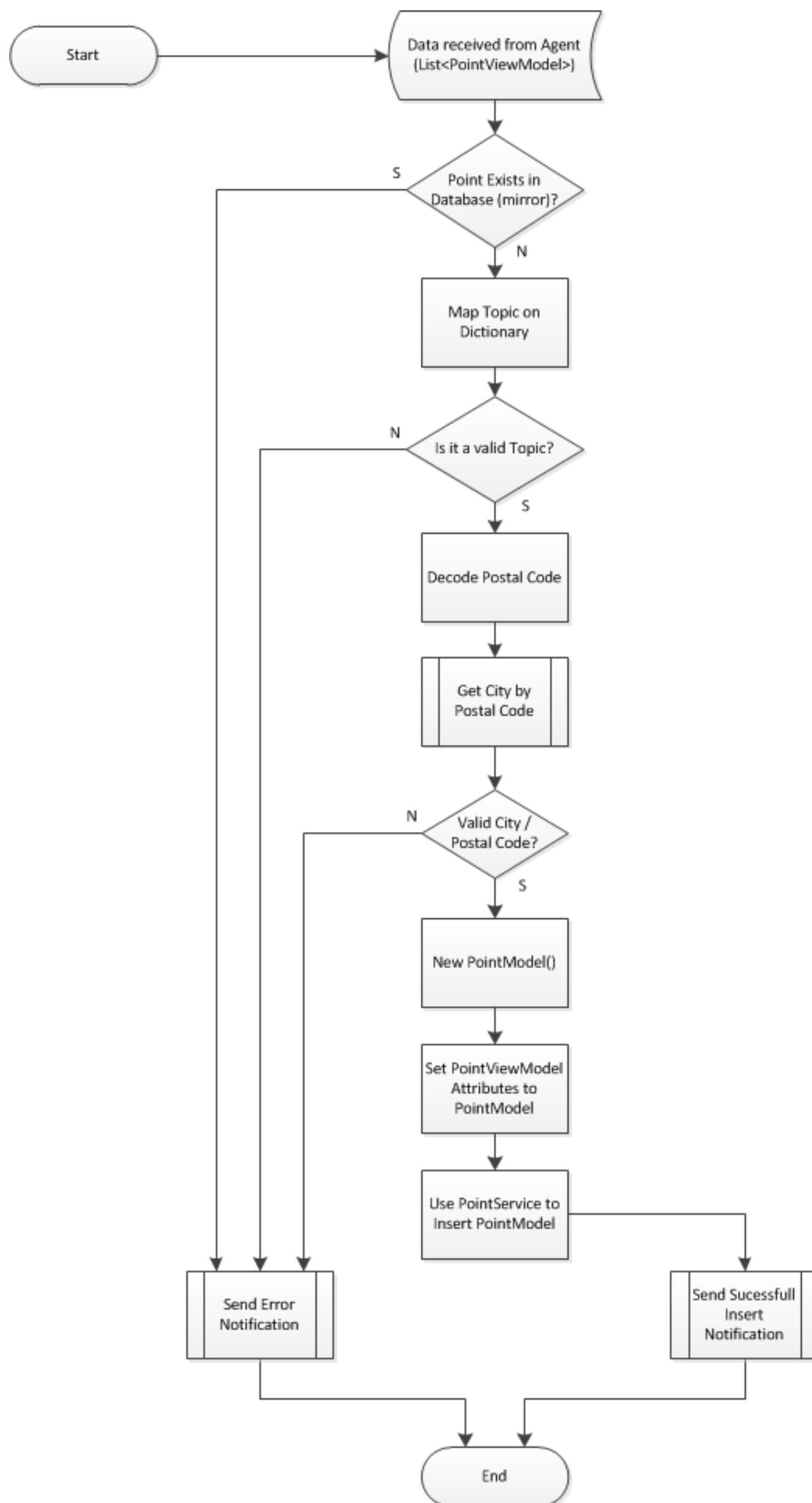


Figura 116. Fluxograma de operações do AgentOperationService

3.4.20 Multilingue

O sistema *multilingue* foi implementado ao nível das aplicações, tanto o *Web Portal* como o *Android* e ao nível de conteúdos.

Web Portal

A implementação de *multilingue* no Web Portal é efectuado através do uso de *resources*, tal como explicado no capítulo 3.1. Estes mesmos *resources* estão divididos por entidades e como se pode ver na Figura 117, cada entidade contém as suas próprias traduções, que são armazenadas em ficheiros *RESX*.

Esses ficheiros são constituídos pelo nome da entidade (*Language*) seguido pelo código *ISO2*³² da língua em questão, separado por um ponto final (*Language.EN*).

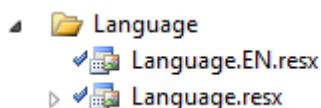


Figura 117. Entidade *Language* com os ficheiros de *resources*

A língua por omissão não contém o código *ISO*, derivando assim apenas as outras línguas. Podemos ver o conteúdo de um *resource* nas diferentes linguagens na Figura 118 e Figura 119.

³² http://en.wikipedia.org/wiki/ISO_2

Name	Value
Activo	Activo
Apagar	Apagar Idioma
AreYouSure	Tem a certeza?
ConfirmDelete	Tem a certeza que pretende eliminar o idioma
Criar	Criar Idioma
DeleteMessage	O idioma irá ser apagado e não há retorno.
Editar	Editar Idioma
Erro	Erro
Imagen	Imagen
ISO	Código ISO
LinguaHeader	Dados do Idioma
Lista	Lista de Idiomas
Nome	Nome
Nova	Novo Idioma
Titulo	Idiomas
UpdateException	Não pode apagar a lingua porque se encontram registos associados a ela.

Figura 118. Conteúdo do resource Language em Português

Name	Value
Activo	Active
Apagar	Delete Language
AreYouSure	Are you sure?
ConfirmDelete	Are you sure you want to delete de language?
Criar	Create New Language
DeleteMessage	The language will be deleted and there's no turn back
Editar	Edit Language
Erro	Error
Imagen	Image
ISO	ISO Code
LinguaHeader	Language Data
Lista	Languages List
Nome	Name
Nova	New Language
Titulo	Languages
UpdateException	You cannot delete the language because there's other records associated to them

Figura 119. Conteúdo do resource Language em Inglês.

Ao efectuar a construção de uma *view*, a *framework* verifica a cultura da *thread* que está a ser processada e vai usar o *resource* correspondente à língua escolhida.

```

<thead>
  <tr>
    <th class="ui-widget-header">
      @ViewRes.Language.Nome
    </th>
    <th class="ui-widget-header">
      @ViewRes.Language.ISO
    </th>
    <th class="ui-widget-header">
      @ViewRes.Language.Imagem
    </th>
    <th class="ui-widget-header">
      @ViewRes.Language.Activo
    </th>
    <th class="ui-widget-header" style="width: 150px;">
    </th>
  </tr>
</thead>

```

Figura 120. Utilização de resources numa View

A nível de desenvolvimento .NET é transparente para o programador a implementação desta funcionalidade, bastando referenciar que o texto a mostrar corresponde a uma *resource*, como se pode ver na Figura 120.

Android

As traduções ao nível de *Android* são também baseadas em *resources*. No entanto, ao contrário da implementação em .NET, estes *resources* não são diferenciadas por ficheiros, mas sim por pastas, em que, o nome destas é precedido de um sufixo correspondente ao ISO2 da linguagem a utilizar, conforme se pode ver na Figura 121.

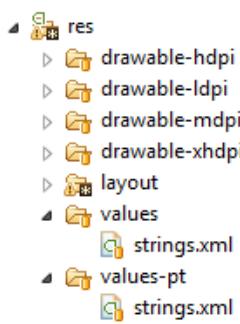


Figura 121. Resources em Android

A linguagem a utilizar corresponde à língua utilizada no Sistema Operativo do telefone, não sendo possível alterar a mesma em tempo de execução.

Na Figura 122 e Figura 123 podemos comparar as traduções entre Português e Inglês dos valores a usar na aplicação. Estes são armazenados em ficheiros XML com os nós correspondentes aos valores a utilizar.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="hello">Hello World, AppTourActivity!</string>
    <string name="app_name">AppTour</string>

    <string name="session_missing">Session Missing</string>
    <string name="session_required">A valid Session is required to continue</string>
    <string name="button_login">Login</string>
    <string name="button_register">Register</string>
    <string name="button_ok">OK</string>
    <string name="button_no">No</string>
    <string name="button_back">Back</string>
    <string name="button_cancel">Cancel</string>
    <string name="button_confirm">Confirm</string>
    <string name="button_map">See Map</string>
    <string name="button_location">Set Location</string>
    <string name="button_filters">Set Filters</string>
    <string name="button_search">Do Search</string>
    <string name="button_save">Save</string>
    <string name="button_save_as">Save As</string>
    <string name="button_delete">Delete</string>
    <string name="button_themes">Set Themes / Topics</string>

    <string name="menu_about">About AppTour</string>

    <string name="about_apptour">AppTour is a ISEP/PESTI project\nVisit us at our WebSite for more info</string>
    <string name="tap_2_continue">Tap the image to continue</string>
    <string name="register">Register a New User</string>
    <string name="username">Username</string>
    <string name="password">Password</string>
```

Figura 122. Exemplo de traduções no *Android* em Inglês

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="hello">Olá Mundo, AppTourActivity!</string>
    <string name="app_name">AppTour</string>

    <string name="session_missing">Sem Sessão</string>
    <string name="session_required">É necessária uma sessão válida para continuar</string>
    <string name="button_login">Login</string>
    <string name="button_register">Registrar</string>
    <string name="button_ok">OK</string>
    <string name="button_no">Não</string>
    <string name="button_back">Voltar</string>
    <string name="button_cancel">Cancelar</string>
    <string name="button_confirm">Confirmar</string>
    <string name="button_map">Ver mapa</string>
    <string name="button_location">Localizar</string>
    <string name="button_filters">Def. Filtros</string>
    <string name="button_search">Pesquisar</string>
    <string name="button_save">Gravar</string>
    <string name="button_save_as">Gravar como</string>
    <string name="button_delete">Apagar</string>
    <string name="button_themes">Def. Temas / Topicos</string>

    <string name="menu_about">Acerca de AppTour</string>

    <string name="about_apptour">AppTour é um projeto PESTI/ISEP \nVisite-nos no nosso WebSite para mais informações</string>
    <string name="tap_2_continue">Tap na imagem para continuar</string>
    <string name="register">Registrar um novo Utilizador</string>
    <string name="username">Username</string>
    <string name="password">Password</string>
```

Figura 123. Exemplo de traduções no *Android* em Português

Tradução de Conteúdos

As traduções implementadas ao nível de base de dados foram pensadas para traduzirem registo de tabelas mestre, ou seja, tabelas como Temas, Tópicos, Pontos e Atributos. As tabelas como os comentários não são previstas de contemplarem traduções.

A tabela das traduções tem a seguinte estrutura, conforme a Figura 124.

TRANSLATIONS		
Column Name	Data Type	Allow Nulls
ID	uniqueidentifier	False
ID_LANGUAGE	uniqueidentifier	False
FOREIGN_ID	uniqueidentifier	False
TABLE_NAME	nvarchar(250)	False
FIELD_NAME	nvarchar(250)	False
VALUE	nvarchar(500)	False

Figura 124. Estrutura da tabela *Translations*

Esta estrutura é muito generalista, permitindo assim a tradução de qualquer registo na base de dados, desde que mantenham coerente a informação que o identifica.

ID_LANGUAGE	Identificador da Língua
FOREIGN_ID	Identificador do Registo
TABLE_NAME	Nome da Tabela
FIELD_NAME	Nome do Campo
VALUE	Valor Traduzido

O mapeamento destas colunas é efectuado ao nível do repositório, no momento em que são recolhidos da base de dados. Um exemplo dessa implementação pode ser visto na Figura 125.

```
#region - IQueryable GetCitiesWithTranslations(AppTourEntities data, string ISO2)
private IQueryable<CityModel> GetCitiesWithTranslations(AppTourEntities data, string ISO2)
{
    var query = from c in data.CITY
                orderby c.NAME
                select new CityModel
                {
                    Id = c.ID,
                    Country = (from p in data.COUNTRY
                               where p.ID == c.COUNTRY.ID
                               select new CountryModel
                               {
                                   CountryCode = p.COUNTRY_CODE,
                                   Id = p.ID,
                                   CountryName = p.COUNTRY_NAME,
                                   ISO = p.ISO,
                                   ISO3 = p.ISO3,
                                   Name = p.NAME
                               }).FirstOrDefault(),
                    Name = c.NAME,
                    NameTranslated = (from l in data.TRANSLATIONS
                                      where l.FIELD_NAME == "NAME"
                                      && l.TABLE_NAME == "CITY"
                                      && l.FOREIGN_ID.Value == c.ID
                                      && l.LANGUAGE.ISO2 == ISO2
                                      select l.VALUE).ToString()
                };
    return query;
}
#endregion
```

Figura 125. Leitura de cidades com conteúdos traduzidos.

3.5 Testes Efectuados

Os testes foram efectuados no decorrer do desenvolvimento do código de modo específico a cada função a ser implementada.

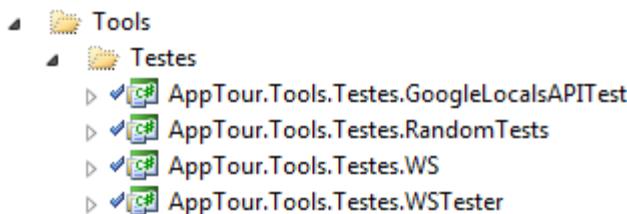


Figura 126. Pasta de projectos de teste

No entanto, houve casos em que foi necessário a criação de alguns projectos de testes para apoio ao desenvolvimento, visto que não ser viável a execução de um processo por inteiro para o teste de uma única função. Assim sendo, foram criados alguns projectos *Console Application*, que invocavam directamente a função a teste e imprimia os resultados na consola.

GoogleLocalsAPITest

Este projecto serve para testar a *API* utilizada pelo *AgenteService*, nomeadamente o *GoogleLocalsAPI*, sem ser necessário iniciar todo o processo de agentes de importação.

RandomTests

Este projecto contém chamadas a funções distintas e serve para apoiar o desenvolvimento do sistema.

WS e WSTester

Projectos que invocam experiências de implementação de métodos do *Web Service* para verificar se estão a ser cumpridos os parâmetros de entrada, saída e para possibilitar o *debug* na própria máquina, utilizando diferentes modos de acesso e protocolos.

4 Conclusões

No presente capítulo serão referidas as conclusões finais obtidas no final do projecto de estágio. Será dedicado um capítulo aos objectivos que foram concluídos, ou caso não o tenham sido na totalidade, será referido o seu grau de conclusão.

Dada a complexidade e envergadura do projecto, alguns casos de uso encontraram-se por efectuar, pelo que será dedicado uma secção a esses mesmos casos, onde se discutem as abordagens esperadas para a sua conclusão em trabalhos futuros.

Por fim será dada uma apreciação global a todo o projecto pelos intervenientes, assim como uma opinião mais pessoal a todo o processo de desenvolvimento e impressões obtidas.

4.1 Objectivos realizados

Tal como apresentado no capítulo 2.2 a solução pretendida passava pelo desenvolvimento de um sistema centralizador de informação de georreferenciação sobre pontos e eventos assim como de um modo de apresentação dessa mesma informação através de uma forma prática e de preferência móvel.

No processo de análise foram identificados quatro componentes essenciais para o sucesso da aplicação. Esses componentes seriam o repositório, o *Web Portal*, a plataforma *Android* e por fim os Agentes. Todos estes componentes foram implementados e encontram-se em funcionamento e em correcta interacção entre eles, pelo que podemos dizer que o principal objectivo se considerou cumprido.

Alguns requisitos foram definidos que seriam considerados imprescindíveis, enquanto outros apenas trariam valor acrescido à aplicação. Esses requisitos deram origem a alguns casos de uso que teriam de ser considerados. Embora muitos desses casos de uso se encontrem desenvolvidos na data de elaboração deste relatório, outros há que ainda se encontram em desenvolvimento ou mesmo nem se encontram implementados. Tais limitações deveram-se exclusivamente devido a falta de tempo disponível para a sua implementação.

Na Tabela 7 far-se-á uma descrição sobre quais os requisitos que eram necessários e quais os que foram efectivamente considerados:

Tabela 7. Objectivos cumpridos em requisitos

Requisitos	Conclusão	Comentário
Requisitos Básicos		
Resposta tempo útil (criação de repositório)	100%	
Web Portal	100%	
Android	100%	
Agentes	100%	
Pontos	100%	
Eventos	30%	Embora prevista a estrutura a nível de Base de Dados, ainda não foi possível a implementação do restante repositório, assim como a pesquisa desses mesmos Eventos.
Temas / Tópicos	100%	
Multilingue	100%	
Partilha em sítios Sociais	100%	
Gestores Conteúdo	100%	
Perfis de pesquisa	100%	
Requisitos Complementares		
Multicanal	100%	Embora prevista a implementação em multicanal apenas foi criada a aplicação móvel em <i>Android</i> .
Horários	20%	Embora previsto a nível de Base de Dados, não estão criadas as condições para a inserção desses horários, nem mesmo a consideração dos mesmos nas pesquisas a efectuar.
Atributos	100%	
Partilha em sítios Sociais	100%	
Calendarização de Eventos	0%	Não foi possível a implementação desta situação até ao momento de feitura deste relatório.
Requisitos não funcionais		
Usabilidade	Este requisito foi atingido. Não houve nenhum levantamento de opiniões acerca do melhor método de interacção pelo que se tentou seguir algumas das exemplos existentes no mercado	

Segurança	<p>Embora nunca se possa ter certeza a 100% nestes aspectos, os métodos de segurança mais básicos do Web Portal foram delegados para as ferramentas Microsoft, pelo que se consideram minimamente robustos.</p> <p>A nível da Android evitou-se guardar informação sensível.</p>
Confiabilidade	<p>A aplicação Android segue uma ordem de acções que permite a validação das condições mínimas até à obtenção de resultados.</p> <p>No entanto deveriam ser efectuados mais testes, nomeadamente a nível de perca de conexão de internet, para verificar o comportamento da mesma.</p>
Desempenho	<p>Através de alguns testes foram implementadas melhorias e conseguidos ganhos substanciais no processo de importação.</p> <p>Alguns métodos de paralelismo foram revistos, mas apenas medidos para as máquinas de desenvolvimento, o que se prevê que embora possuam um desempenho razoável, não são garantia em caso de mudança de servidor.</p>
Autonomia	<p>O processo de importação foi conseguido de uma maneira que se considera autónomo. Os agentes são controlados por um programa central que os instancia, executa e controla o seu processo.</p> <p>No entanto melhorias como a implementação de serviços Windows com logs em ficheiros poderiam ser contempladas.</p>

Seguidamente serão apresentados os casos de uso previstos, seu grau de implementação assim como uma breve descrição de como estes se encontram, ou não, a cumprir as funcionalidades requeridas:

Tabela 8. Objectivos cumpridos em Casos de uso

Caso de Uso	Conclusão	Comentário
Efectuar uma pesquisa	100%	
Alterar o ponto de localização	100%	
Considerar apenas Pontos disponíveis	0%	Não desenvolvido devido à não implementação dos horários.
Comentar um Ponto/ Evento	60%	Ainda não estão considerados os comentários de Eventos.
Classificar um Ponto/Evento	60%	Ainda não está considerada a classificação de Eventos
Denunciar um comentário	100%	
Criar um novo Perfil de Pesquisa	100%	
Configurar um Perfil de Pesquisa	80%	Devido à não implementação dos horários, ainda não é possível considerar este ponto como completo.
Eliminar um perfil de pesquisa	100%	
Calendarizar Eventos	0%	Caso ainda não previsto nem implementado, devido à não implementação de Eventos.
Publicar Pontos e Eventos em redes sociais	60%	Os pontos estão a ser partilhados, embora os Eventos ainda não.
Resolver denúncias	30%	Necessária a um modo de visualização das denúncias efectuadas assim como um modo de as resolver.
Inserir Pontos e Eventos	60%	Ainda não se inserem Eventos.
Alterar conteúdos em Pontos e Eventos	60%	Ainda não estão considerados os Eventos.
Alterar permissões Utilizador	100%	
Bloquear e desbloquear utilizadores	100%	
Procurar e resolver Pontos e Eventos duplicados	60%	Necessário implementar um modo de listagem desses mesmos pontos / eventos que permita uma ordenação específica assim como uma pesquisa por Pontos que possuam uma distância pequena entre eles. No entanto encontra-se previsto o modo de eliminação e edição desses mesmos pontos assim como listagens básicas dos mesmos.

Recolha de informação automática	80%	O processo de importação encontra-se planeado, estruturado e implementado. Este ponto no entanto nunca pode ser considerado acabado, pois esse é exactamente o objectivo dele. Poder “crescer” e estar permanentemente em desenvolvimento de novos Agentes para diferentes APIs.
Classificação e armazenagem de informação	80%	A classificação e armazenagem de informação necessita apenas algumas melhorias como a consideração de horários. Outros atributos não previstos no modelo podem também ser implementados para as APIs que os possuam.

4.2 Limitações e trabalho futuro

Tal como referido no capítulo 4.1 são mencionadas as funcionalidades que ainda não se encontram implementadas, assim como as que ficaram por finalizar. Muitas dessas limitações estão pendentes de um desenvolvimento crucial, mas que foi deixado de lado, por motivos exclusivos de falta de tempo.

Fala-se da implementação dos Eventos, assim como a sua associação aos Pontos e consequente apresentação. Este é sem dúvida um dos pontos mais importantes a desenvolver e sem o qual, muitos dos casos de uso não poderão ser finalizados ou executados, como por exemplo:

- Inserção de Eventos;
- Edição da informação de um Evento;
- Comentar um Evento;
- Classificar um Evento;
- Publicação de Eventos em redes sociais;
- Calendarização de Eventos;
- Procurar e resolver Eventos duplicados.

Outro desenvolvimento a considerar que faz parte dos requisitos complementares são a inclusão dos horários, tanto no processo de importação, como nas pesquisas a efectuar. Embora previstos a nível de Base de Dados, os horários não estão a ser considerados nem sequer está implementado em nenhum local da arquitectura. Assim sendo a implementação da política de horários envolve ainda a criação dessa mesma estrutura a nível de negócio, assim como a sua inclusão na criação e importação de pontos e a elaboração de testes.

Outro aspecto que foi verificado em falta foi a edição de pontos pelo utilizador que os criou, para além do gestor de conteúdos. Tal necessidade foi verificada na implementação dos atributos, mas também numa perspectiva de utilizador, já que faz todo o sentido que quem cria um ponto, faça a própria manutenção para permitir, entre outros, a veracidade do conteúdo ou actualizações constantes, tais como URL, imagens ou atributos diversos.

Os Agentes utilizados pelo sistema resumem-se neste momento a um Agente de ligação à *API* do *Google Locals*. Outro Agente estaria em desenvolvimento na altura de criação deste relatório, porém ainda não se pode considerar finalizado. A criação de mais Agentes é uma situação que embora prevista, se torna muito laboriosa, devido à pesquisa necessária para cada uma dessas *APIs* assim como da especificação das mesmas. Assim o processo de Agentes ainda se encontra numa fase inicial de desenvolvimento, quando nos referimos ao número de Agentes desenvolvido.

Os restantes pontos em falta poderiam ser ultrapassados através da criação de diversos modos de listagem para Pontos, Eventos e mesmo denúncias. Mesmo uma melhoria da informação disponibilizada do *Web Portal* poderia ser atingida através do reaproveitamento dessas mesmas listagens.

A implementação de testes unitários poderiam auxiliar na robustez, validação e garantia de integridade da solução final. Isto foi algo que consideramos mas devido à falta de experiência e a investigação inerente, foi adiado para uma segunda fase do projecto.

O uso de registo de *logs* foi umas das necessidades que se sentiu no processo de desenvolvimento do sistema. Extensões como *Log4NET*³³, *NLog*³⁴ ou outros, poderiam auxiliar na depuração de problemas e consequentemente na sua resolução. Mais uma vez, devido a restrições de tempo e por outras prioridades esta implementação foi constantemente adiada. Será algo a considerar numa futura intervenção.

³³ <http://nuget.org/packages/log4net>

³⁴ <http://nuget.org/packages/NLog>

A nível de *Android*, a implementação das funcionalidades previstas no *Web Portal*, tais como a criação de pontos poderia dar um valor acrescentado, já que dessa forma poderia ser possível uma criação dos pontos no local. Outra funcionalidade a ter em conta e que poderia trazer muito valor, seria o aproveitamento da câmara existente normalmente nos *smartphones*, para tirar e enviar fotos desses mesmos pontos para o servidor. No entanto isso implicaria que houvesse de alguma forma um alojamento dessas mesmas imagens, o que o sistema ainda não contempla.

4.3 Apreciação final

O projecto *PESTI* serviu essencialmente para colocar os intervenientes em contacto com o desenvolvimento de um projecto desde raiz, com as dificuldades daí inherentes, onde o objectivo principal seria a aplicação de conceitos e métodos de estruturação e esquematização assim como muitas das matérias aprendidas durante a licenciatura no *DEI*.

O projecto *AppTour* apresentou-se desde o início como um desafio, tanto pela dificuldade de implementação, devido à enorme envergadura que este apresentava, como também pela dificuldade em apresentar o conceito de criar uma solução cujo objectivo não fosse uma ideia nova, mas sim, uma abordagem diferente a soluções já implementadas.

Assim sendo, tornou-se uma dificuldade acrescida poder provar a viabilidade do projecto desde o início, sendo necessário, muitas vezes, uma reelaboração dos conceitos implementados.

Devido à vontade de explorar tecnologias novas para tornar o projecto *PESTI* numa forma de crescimento técnico acrescido, houve muitas vezes algumas curvas de aprendizagem que não só atrasavam como colocavam em risco algum do desenvolvimento. Muitas vezes foi necessário alguma reformulação da implementação de alguns métodos de uma forma que, embora não a ideal, permitiam a ultrapassagem dessa mesma situação.

Outra dificuldade apresentada passou pela aprendizagem feita de um modo autodidacta das tecnologias, *APIs* e restantes ferramentas necessárias para o desenvolvimento do projecto. Esse facto obrigou a alguma investigação acrescida, o que reduziu de uma forma considerável o tempo disponível para o desenvolvimento.

O desenvolvimento de todo o projecto fez-se em horário pós-laboral, já que os autores desempenham funções durante o dia, o que obrigou a alguns sacrifícios a nível pessoal, assim como a algumas cedências de foro privado.

Todas as situações descritas previamente provocaram muitas vezes algum desânimo e desmotivação, o que obrigou a alguma disciplina e esforço para conseguir retomar as funções.

Bibliografia

LEIC-FEUP, Guia de Elaboração de Relatórios LEIC. Texto académico.

DEI-ISEP (2002), Normas de elaboração de relatório de estágio. Normas de avaliação.

Sousa, Paulo (2002) Pequeno Guia de Elaboração de Relatórios, Unidade de Ensino
Instituto Superior de Engenharia do Porto.

Colin Campbell, Ralph Johnson, Ade Miller, Stephen Toub (2010), Parallel
Programming with Microsoft .NET: Design Patterns for Decomposition and
Coordination on Multicore Architectures.

Jon Galloway, Phil Haack, Brad Wilson, K. Scott Allen (2011), Professional ASP.NET
MVC 3

Charlie Collins, Michael Galpin, Matthias Kaepller (2011), Android in Practice version
6.

W. Frank Ableson, Robi Sen, Chris King (2011), Android in Action second Edition.

Explicação do uso de Geolocalização, em http://en.wikipedia.org/wiki/W3C_Geolocation_API.

Documentação do JQuery, em http://docs.jquery.com/Main_Page.

API do JQuery, em <http://api.jquery.com/>.

jQuery UI, em <http://jqueryui.com>

Developing in Android, em, <http://developer.android.com/index.html>

NuGet, em <http://nuget.org/>

Fluent Validators, em <http://fluentvalidation.codeplex.com/>

PageList MVC, em <http://nuget.org/packages/PagedList.Mvc>

Entity Framework, em <http://msdn.microsoft.com/en-us/library/bb399572.aspx>

Dúvidas e algumas consultas, em <http://stackoverflow.com/>

Anexos

Anexo 1 Diagrama de Sequência de Pesquisa de Pontos

Devido ao tamanho excessivo do diagrama, este segue num ficheiro em anexo com o nome *SD-Search_AppTour.Web Services.WS.pdf*.

Anexo 2 Diagrama de Sequência Inserir Comentário

Devido ao tamanho excessivo do diagrama, este segue num ficheiro em anexo com o nome *SD-InsertComment_AppTour.UI.Web.Web Portal.pdf*.

Anexo 3 Diagrama de Sequência Classificar um Ponto

Devido ao tamanho excessivo do diagrama, este segue num ficheiro em anexo com o nome *SD-InsertVote_AppTour.UI.Web.Web Portal.pdf*.

Anexo 4 Diagrama de Sequência Criar Perfil de Pesquisa

Devido ao tamanho excessivo do diagrama, este segue num ficheiro em anexo com o nome *SD-Create-SearchProfileController-AppTour.UI.Web.Web Portal.pdf*.

Anexo 5 Diagrama de Sequencia Agente Google Locals

Devido ao tamanho excessivo do diagrama, este segue num ficheiro em anexo com o nome *SD-GoogleLocalsAdapter.AppTour.AppTour.Agents.Service.pdf*.

Anexo 6 Diagrama de Sequencia Agents Controller

Devido ao tamanho excessivo do diagrama, este segue num ficheiro em anexo com o nome *SD-AgentsController.AppTour.AppTour.Agents.Service.pdf*.

Anexo 7 Diagrama de Sequência Agent Operations

Devido ao tamanho excessivo do diagrama, este segue num ficheiro em anexo com o nome *SD-AgentsOperation_InsertPoint.AppTour.AppTour.Agents.Service.pdf*.

Anexo 8 Diagrama de Sequência de Inserção de Ponto

Devido ao tamanho excessivo do diagrama, este segue num ficheiro em anexo com o nome *SD-PointService_InsertPoint.AppTour.Business.pdf*.

Anexo 9 Diagrama de Sequência de Edição de Ponto

Devido ao tamanho excessivo do diagrama, este segue num ficheiro em anexo com o nome *SD-PointService_UpdatePoint.AppTour.Business.pdf*.

Anexo 10 Documento de Casos de Uso

Este documento segue em anexo.