

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



FEUP

Aplicação Web para Classificação de Parcelas Agrícolas

Nelson Bilber Rodrigues

Dissertação

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Doutor António Fernando Vasconcelos Cunha Castro Coelho, Professor Auxiliar do Departamento de Engenharia Informática da Faculdade de Engenharia da Universidade do Porto

Julho de 2009

Aplicação Web para Classificação de Parcelas Agrícolas

Nelson Bilber Rodrigues

Dissertação
Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: João Francisco de Sousa Cardoso (Professor Auxiliar Convidado da Faculdade de Engenharia da Universidade do Porto)

Arguente: Luís Manuel Borges Gouveia (Professor Associado Universidade Fernando Pessoa)

Vogal: António Fernando Vasconcelos Cunha Castro Coelho (Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto)

14 de Julho de 2009

Resumo

Os Sistemas de Informação Geográfica são cada vez mais comuns e a sua aplicabilidade quotidiana difundiu-se por diversas áreas envolvendo inúmeras organizações e indivíduos em sistemas heterogéneos. O sector agrícola não foi excepção e a adopção deste tipo de sistemas originou um novo conceito de produção denominado por "agricultura de precisão".

O trabalho exposto nesta dissertação foi realizado no âmbito de um projecto em desenvolvimento pelo INESC Porto para o IVDP (Instituto dos Vinhos do Douro e Porto). O projecto tem como objectivo principal o desenvolvimento de um sistema que possibilite ao IVDP melhorar os processos de gestão e classificação das parcelas com vinha, tendo em consideração as especificidades da *Região Demarcada do Douro*. Um dos objectivos desta dissertação passa por realizar um estudo sobre este tipo de sistemas em diversas regiões do globo, analisando os procedimentos adoptados.

O sistema foi desenvolvido recorrendo à tecnologia RIA (Rich Internet Applications). Esta tecnologia apresenta um novo leque de oportunidades no desenvolvimento Web, terminado com as limitações apresentadas pelas aplicações Web tradicionais. Actualmente existem lacunas no que diz respeito a metodologias que possibilitem a transformação de aplicações Web tradicionais em aplicações do tipo RIA. Com a realização desta dissertação foi possível efectuar um estudo sobre esta temática e desenvolver uma metodologia que permite preservar toda a camada de negócio existente, implementando a GUI (Graphical User Interface) utilizando RIA. O sistema foi desenvolvido sobre uma arquitectura padronizada assente um sistema modular. Cada módulo agrupa um conjunto de componentes implementados segundo vários padrões de desenho de software.

Com a implementação do protótipo foi possível desenvolver métodos que possibilitam a visualização de informação georreferenciada relativa às parcelas agrícolas com vinha, utilizando serviços de *Web mapping*.

Concluindo, o resultado do estudo proporcionado pela realização desta dissertação permitiu desenvolver um sistema de informação geográfica dinâmico, interactivo e escalável adaptado às necessidades da *Região Demarcada do Douro*.

Abstract

Geographical Information Systems became widespread and their applicability is continuously expanding in several areas involving diverse organizations and individuals in heterogeneous systems. The agricultural sector is no exception and the adoption of such systems resulted in the development of a new concept named "precision agriculture".

The work outlined in this dissertation was conducted under a project in development stage by INESC Porto for the IVDP (Instituto dos Vinhos do Douro e Porto). The main objective of this project is the development of a system that allows IVDP to improve the management and classification of vineyard parcels taking into account the specificities of the *Região Demarcada do Douro*. One of the objectives of this dissertation is to conduct a study of such systems in various regions around the globe, examining the procedures adopted.

The system was developed using the design concepts offered by the RIA (Rich Internet Applications) technology. This technology leads to a range of new opportunities in Web development overcoming the limitations presented by traditional Web applications. Currently there is a lack of methodologies that enable the transformation of traditional Web applications (Web 1.0) to the RIA technology. This work leads to the development of a methodology that allows preserving all existing business layer and developing the GUI (Graphical User Interface) using RIA. The system was developed based on a standardized architecture in a modular system. Each module includes a set of components implemented in several software design patterns.

With the implementation of the prototype it was also possible to develop methods to integrate the visualization of geographic information of the agricultural parcels with vines, using *Web mapping* services.

In conclusion, the result of this work lead to the development of a geographical information system featuring a dynamic, interactive and scalable environment according to the needs of the *Região Demarcada do Douro*.

Agradecimentos

Gostava de agradecer ao meu orientador Dr. António Fernando Vasconcelos Cunha Castro Coelho, pela prontidão na resposta às minhas dúvidas e pela ajuda na revisão ortográfica.

Gostava de agradecer ao Engenheiro José Lino Oliveira pelos conselhos dados na resolução de problemas inerentes ao projecto.

Gostava de agradecer a todos os colegas de trabalho do INESC Porto, pelo bom ambiente de trabalho proporcionado e pela predisposição demonstrada, facilitando o processo de integração.

Gostava de agradecer à minha família e aos meus amigos que sempre me acompanharam ao longo deste percurso.

Nelson Bilber Rodrigues

Índice

1. Introdução.....	1
1.1 Contextualização.....	1
1.2 Problema	2
1.3 Objectivos	2
1.4 Metodologia	3
1.5 Organização da Dissertação	3
2. Sistemas de informação geográfica aplicados à "agricultura de precisão"	5
2.1 Sistemas de Informação Geográfica (SIG)	5
2.1.1 Software utilizado nos Sistemas de Informação Geográfica.....	7
2.1.2 Normalização da informação geográfica.....	8
2.1.2.1 Open Geospatial Consortium	8
2.1.2.2 ISO/TC 211 (Geographic information/Geomatics)	9
2.2 Sistemas de Informação Geográfica aplicados à Agricultura	9
2.2.1 Na Toscânia (Itália), novos estilos na gestão de vinhas provenientes da "agricultura de precisão"	12
2.2.2 British Columbia (Canadá), utilização de SIG como forma de melhorar a produtividade das vinhas nos vales de Okanagan e Similkameen	13
2.2.3 (Austrália), ARvino	14
2.2.4 Valdepenas (Espanha), estudo da diferenciação em zonas vitícolas na Denominação de Origem Valdepenas	15
2.2.5 Sumário	16
3. SIG Web e tecnologias Web	17
3.1 Arquitectura	18
3.1.1 Aplicações Thin client.....	18
3.1.2 Aplicações Thick client.....	19
3.2 Mapeamento Web	20
3.2.1 Projecção	21
3.2.2 Mashup.....	21
3.3 Rich Internet Applications	22
3.3.1 Modo de funcionamento.....	22

3.3.2	Conceitos implícitos no desenho de <i>Rich Internet Applications</i>	24
3.3.3	Metodologias.....	26
3.3.4	Adaptação de aplicações Web 1.0 para Web 2.0.....	30
3.3.5	Soluções	31
3.3.5.1	Tecnologias RIA <i>plugin-based</i>	33
3.3.6	Limitações	36
3.4	Sumário	36
4.	Aplicação Web para a gestão e classificação de parcelas agrícolas.....	37
4.1	Metodologia proposta	37
4.2	Análise do sistema existente	39
4.2.1	Visão geral.....	39
4.2.2	Arquitectura física.....	41
4.2.2.1	Vista de processos.....	43
4.2.3	Arquitectura lógica.....	43
4.2.3.1	Detalhes de implementação.....	45
4.3	Aplicação da metodologia.....	47
4.3.1	Camada intermédia de serviços.....	47
4.3.1.1	Padrões de desenho de software.....	48
4.3.2	Estruturas e tipos de dados	48
4.3.3	Desenvolvimento da GUI baseado num sistema hierárquico.....	52
4.3.4	Desenvolvimento da GUI utilizando <i>Composite Application Library</i>	53
4.3.4.1	Padrões de desenho de software na <i>Composite Application Library</i>	55
4.4	Visualização de Mapas	57
4.5	Sumário.....	59
5.	Implementação e análise de resultados	60
5.1	Enquadramento	60
5.2	Seleccção da arquitectura	60
5.2.1	Protótipo baseado no sistema hierárquico	60
5.2.2	Protótipo utilizando <i>Composite Application Library</i>	63
5.2.3	Síntese comparativa: hierarquia de namespaces <i>versus</i> CAL	64
5.3	Especificação do sistema modular.....	65
5.4	Implementação do sistema modular	67
5.5	Visualização das parcelas com vinha.....	70

5.6	Testes e resultados	71
5.7	Sumário	75
6.	Conclusões e trabalho futuro	76
6.1	Sumário das contribuições gerais	76
6.2	Trabalho futuro	77
7.	Referências bibliográficas.....	78
	Anexos.....	85
	Anexo A.1: Casting	86
	Anexo A.2: Custom Classes	87
	Anexo A.3: Reflection.....	90
	Anexo A.4: Singleton	93
	Anexo A.5: Licença de Utilização.....	94
	Anexo A.6: Informação complementar sobre os testes de carga realizados	95

Lista de Figuras

Figura 1- Categorização do software GIS [Steiniger09]	7
Figura 2 - Base de dados multi-critério [Smith98]	11
Figura 3 - Selecção de informação baseado em parâmetros multi-critério [Smith98] ...	11
Figura 4 - Modelação do ciclo de crescimento das castas de uvas [Smith98]	12
Figura 5 - Informação das parcelas com vinha [ESRI09b]	13
Figura 6 - Diferenciação das zonas de acordo com a temperatura [Bowen05]	14
Figura 7 - Realidade Virtual [King05]	14
Figura 8 - Realidade Aumentada [King05]	14
Figura 9 - Localização das parcelas de mosto [Riquelme06]	16
Figura 10 - Arquitectura SIG Web	18
Figura 11 - Server Side Application	19
Figura 12 - Client Side Applications	19
Figura 13 - Polígono Google Maps [GOOGLE09]	20
Figura 14 - Projecção Mercator [MSOFT09b]	21
Figura 15 - Arquitectura cliente-servidor das aplicações Web tradicionais	23
Figura 16 - Comunicação entre cliente e servidor utilizando RIA	23
Figura 17 - RIA design [Preciado07]	25
Figura 18 - RUX-Model: arquitectura e método [Linaje07]	30
Figura 19 - Descrição da metodologia	38
Figura 20 - Diagrama de casos de usos do Módulo de Classificação de Parcelas	40
Figura 21- Diagrama de distribuição	42
Figura 22 - Diagrama de componentes	42
Figura 23 - Vista a nível de processos	43
Figura 24 - Diagrama de Componentes	43
Figura 25 - Diagrama de sequência do caso de uso processar enquadramento legal	44
Figura 26 - Table Module + Table Data Gateway utilizando DataSets	45
Figura 27 - Distribuição do código	46
Figura 28 - Object Model do DataSet [MSOFT09f]	46
Figura 29 - Camada de serviços	47
Figura 30 - Singleton	48
Figura 31 - Mapeamento DataSet em Hash Table	49
Figura 32 - DataSetSupport	50
Figura 33 - Remoting End-to-End Time [MSOFT09e]	52
Figura 34 - Esqueleto da página do protótipo	52
Figura 35- Definição de namespaces	53
Figura 36- Sistema de pastas	53

Figura 37 - Composite Application integrando múltiplos componentes [MSOFT09g].	54
Figura 38 - Multi-polígonos	58
Figura 39 - Mapa multi-camadas.....	59
Figura 40-Sistema de feedback	61
Figura 41 - Divisão lógica e física do código.....	61
Figura 42 - Consultar Boletim de campo	62
Figura 43 - Janela Modal	63
Figura 44 - Protótipo baseado na biblioteca CAL (Silverlight)	63
Figura 45 - Protótipo baseado na biblioteca CAL (WPF)	64
Figura 46 - Sistema Modular sobre uma arquitetura padronizada.....	66
Figura 47 - Dependência entre módulos.....	67
Figura 48 - Agregação de módulos	68
Figura 49 - Protótipo utilizando a CAL (formulário)	68
Figura 50 - Aplicações CAL.....	69
Figura 51- Service Locator [MSOFT09h]	69
Figura 52 - Visualização da parcela com vinha.....	70
Figura 53 - Visualização multi-polígonos	71
Figura 54 - Descrição da parcela	71
Figura 55 - Comportamento da aplicação baseada em Ajax	72
Figura 56 - Aplicação baseada em Silverlight (sistema hierárquico).....	73
Figura 57 - Aplicação baseada em Silverlight (CAL)	73
Figura 58 – Teste de carga (HashTables)	74
Figura 59 – Teste de carga (DataSetSupport).....	75

Lista de Tabelas

Tabela 1 - Comparação entre aplicações Desktop, Web 1.0 e RIA [Bozzon06].....	24
Tabela 2 - Parâmetros de comparação [Preciado05]	27
Tabela 3 - Comparação entre os requisitos RIA e as metodologias Web[Preciado05] ..	29
Tabela 4- Comparação entre Flash e Silverlight 2.0	35

Abreviaturas e Símbolos

AJAX	Asynchronous Javascript And XML
API	Application Programming Interface
CRUD	Create, read, update and delete
CSS	Cascading Style Sheets
DLL	Dynamic-link library
DOM	Document Object Model
ESRI	Environmental Systems Research Institute
FEUP	Faculdade de Engenharia da Universidade do Porto
GIS	Geographic Information System
GUI	Graphical User Interface
HTML	HyperText Markup Language e Cascading Style Sheets
INESC Porto	Instituto de Engenharia de Sistemas e Computadores do Porto
IVDP	Instituto dos Vinhos do Douro e Porto
RDD	Região demarcada do Douro
RDF	Resource Description Framework
RIA	Rich Internet Applications
SI	Sistema de Informação
SIG	Sistema de Informação Geográfica
UI	User Interface
UML	Unified Modeling Language

1. Introdução

A globalização traz novos desafios aos produtores agrícolas. Estes necessitam de desenvolver novos mecanismos que proporcionem uma maior eficácia e eficiência na produção dos seus produtos de forma a obterem vantagens competitivas. A evolução das tecnologias de informação e dos sistemas geoespaciais aplicados à agricultura, originaram um novo conceito de produção – a agricultura de precisão. Este novo conceito pode ser definido segundo a seguinte citação:

"A agricultura de precisão consiste num conjunto de técnicas e procedimentos, que permite conhecer, localizar geograficamente e delimitar áreas de diferente produtividade utilizando as tecnologias de informação com o objectivo de melhorar a sustentabilidade económica e ambiental da produção" [Tschiedel02].

Existem já estudos e implementações de sistemas de agricultura de precisão. Estes permitem reduzir os custos de produção, reduzir a contaminação ambiental e otimizar os rendimentos das culturas.

Este capítulo começa por apresentar as entidades envolvidas na realização desta dissertação. De seguida, é efectuada a contextualização, definição do problema e descrição dos objectivos. Por fim, é apresentada a metodologia utilizada para levar o cabo o cumprimento dos objectivos propostos, assim como os resultados esperados.

1.1 Contextualização

O trabalho exposto nesta dissertação surge no âmbito de um projecto em desenvolvimento pelo Instituto de Engenharia de Sistemas e Computadores do Porto (INESC Porto) para o Instituto dos Vinhos do Douro e Porto (IVDP). O projecto tem como objectivo principal o desenvolvimento de um sistema que possibilite ao IVDP melhorar o processo de gestão e classificação das parcelas com vinha da RDD (Região

Demarcada do Douro), agilizando o processo de emissão de autorizações para a produção de Mosto Generoso¹ pelos vários viticultores.

As funcionalidades do sistema, de uma forma geral, são as de um Sistema de Informação Geográfica que permita realizar o cadastro de parcelas agrícolas e sua classificação, baseada em vários factores edafo-climáticos (localização, altitude, exposição, inclinação, abrigo, natureza do terreno, pedregosidade, castas, idade da parcela, produtividade, compasso e aramação) segundo o método definido na Portaria n.º 413/2001, de 18 Abril. O protótipo permitirá visualizar a informação georreferenciada utilizando um serviço de *Web mapping*.

1.2 Problema

O problema abordado nesta dissertação consiste no desenvolvimento de um sistema modular e dinâmico, baseado nos conceitos presentes nos sistemas de agricultura de precisão, aplicando-o à viticultura. Com este sistema pretende-se melhorar o processo de emissão de autorizações para a produção de Mosto Generoso, A classificação é realizada mediante a informação georreferenciada armazenada numa base de dados, com extensão espacial.

Pretende-se estudar metodologias que possibilitem o desenvolvimento de um protótipo informático, para gestão do processo de classificação das parcelas com vinha da RDD, assente numa arquitectura padronizada sobre um sistema modular, e aproveitando a flexibilidade e dinamismo proporcionado pelas RIA (Rich Internet Applications) no desenho de interfaces para aplicações Web. O protótipo a desenvolver deve preservar todas as regras de negócio da aplicação existente, aplicando uma nova GUI (Graphical User Interface) implementada segundo os princípios das RIA.

1.3 Objectivos

A utilização dos SIG (Sistemas de Informação Geográfica) na viticultura de precisão possibilita um maior rigor no controlo de qualidade, uma maior disponibilidade de informação sobre as culturas e factores de produção, bem como uma optimização do processo de cadastro e classificação das parcelas com vinha.

Esta dissertação abrange duas áreas distintas: os Sistemas de Informação Geográfica e as *Rich Internet Applications*. Após realizado um estudo prévio nas temáticas anteriormente referidas pretende-se que se desenvolva um protótipo para

¹ *Mosto Generoso* - sumo de uva obtido antes de concluir o processo de fermentação utilizado na produção de vinho do Porto.

gestão do processo de classificação de parcelas com vinha, baseado em vários factores edafo-climáticos. Este protótipo deverá ser implementado recorrendo à tecnologia RIA, explorando as suas capacidades no desenvolvimento de aplicações dinâmicas e usáveis.

Ao nível dos resultados esperados com este trabalho, pretende-se especificar uma arquitectura baseada em módulos independentes que interajam entre si, onde cada módulo é responsável por um determinado grupo de tarefas no processo de cadastro e classificação de parcelas com vinha. Pretende-se ainda o desenvolvimento de um protótipo que possibilite avaliar a viabilidade da tecnologia RIA no desenvolvimento de aplicações Web. Concretamente, o protótipo deverá possibilitar a interacção com uma base de dados espacial, permitindo a gestão do processo de cadastro e classificação das parcelas com vinha, de acordo com as regras de negócio estipuladas. O protótipo deverá possuir ainda um módulo responsável pela integração de um componente de *Web mapping* que possibilite a localização e visualização da informação georreferenciada.

1.4 Metodologia

A metodologia utilizada começa com a revisão do estado de arte, investigando sobre a temática das aplicações de cadastro e classificação das parcelas agrícolas. O estudo consiste numa análise dos sistemas já implementados em diversas regiões do globo, particularmente nos procedimentos adoptados e no software utilizado.

Tendo por base uma especificação prévia, pretende-se projectar uma nova arquitectura padronizada sobre um sistema modular. O projecto começa por estudar os conceitos implícitos no desenvolvimento de RIA e investigar metodologias que permitam a adaptação de aplicações Web tradicionais em aplicações Web dinâmicas e interactivas. Como resultado desse mesmo estudo pretende-se desenvolver uma metodologia que auxilie a transição da aplicação actual para uma aplicação do tipo RIA, tendo em conta o âmbito do problema em questão.

Segue-se a fase de implementação do protótipo aplicando a metodologia anteriormente formulada. Por fim, de forma a confirmar os objectivos propostos para esta dissertação seguem-se as fases de testes, de verificação e de validação do protótipo.

1.5 Organização da Dissertação

Esta monografia é composta por seis capítulos, sendo este o capítulo de introdução. No capítulo [2](#) são apresentados os principais conceitos envolvidos nos Sistemas de Informação Geográfica e são também debatidos e analisados alguns dos

sistemas já implementados em diversas regiões do globo. O capítulo [3](#) faz uma revisão sobre áreas relacionadas com as RIA e os sistemas de *Web GIS*. No capítulo [4](#) é efectuada uma análise ao sistema implementado anteriormente no IVDP e é apresentada uma metodologia que permite transformar aplicações Web 1.0 em Web 2.0. O capítulo [5](#) relata as experiências realizadas e os resultados obtidos na consequência da implementação do protótipo. No capítulo [6](#) são descritas as conclusões gerais e são apresentadas propostas de desenvolvimento futuro.

2. Sistemas de informação geográfica aplicados à "agricultura de precisão"

A aplicabilidade dos Sistemas de Informação Geográfica é bastante vasta e distinta. Neste capítulo são descritos vários conceitos relacionados com o funcionamento e estrutura dos Sistemas de Informação Geográfica. Seguidamente, são apresentadas regras de normalização da informação georreferenciada indispensáveis na interoperabilidade entre os vários sistemas.

É dada ênfase à aplicabilidade dos SIG na “agricultura de precisão” descrevendo qual a utilidade e vantagens da sua utilização. São vários os exemplos de SIG aplicados nos mais diversos sectores da agricultura. No entanto, tendo em conta o enquadramento da dissertação é descrita com especial atenção a empregabilidade dos SIG no sector vitícola. Seguidamente, são apresentados alguns estudos de sistemas implementados em regiões distintas do globo utilizando técnicas e tecnologias diversificadas.

2.1 Sistemas de Informação Geográfica (SIG)

Ao longo dos anos os Sistemas de Informação (SI) têm evoluído, originando e acompanhado as transformações na sociedade. “Actualmente é aceite de uma forma geral, que no mundo dos SI os factores humanos, sociais e organizacionais são tão importantes como os tecnológicos” [Avison01]. Segundo [Laudon04], “os sistemas de informação consistem em vários componentes interligados que recolhem, processam, distribuem e armazenam informação, podendo ser utilizados para organizar e fundamentar decisões”.

Já no passado o Homem sentiu a necessidade de registar e tratar a informação espacial (na exploração marítima, na geografia, na cartografia, entre outros). Com a evolução dos Sistemas de Informação surgiram os Sistemas de Informação Geográfica (SIG) originando uma revolução na forma de representação e distribuição espacial.

“Os sistemas designados por SIG cresceram da relação de múltiplas disciplinas e tecnologias que têm como objecto de estudo comum a informação geográfica. Os SIG não só potenciam a poderosa manipulação e análise da informação presente nos

sistemas de informação, reproduzindo-a com elevada precisão num mapa ou imagem, com também permite relacionar informação proveniente de sistema distintos.” [Martin91].

Um SIG também pode ser definido como, “ um software que fornece capacidades de armazenamento, recuperação, análise, visualização e mapeamento da informação proveniente de dados espaciais” [Church02]. Os sistemas de informação geográfica são capazes de processar variadas fontes de informação e representá-la de uma forma interactiva.

Normalmente os SIG estão associados a mapas. Mas os mapas apenas representam parte do sistema. Um Sistema de Informação Geográfica possui informação georreferenciada que associada a um programa de mapeamento simples ou adicionando dados a uma ferramenta de mapeamento on-line, pode fornecer um variado leque de resoluções de problemas [ESRI09a]. Um Sistema de Informação Geográfica pode ser visualizado de três formas diferentes, visão base de dados, visão de mapa e modelo de visão.

Visão de Base de Dados O SIG consiste numa base de dados estruturada que descreve o mundo em termos geográficos.

Visão de Mapa Um SIG é um conjunto de mapas inteligentes e outras representações gráficas, como por exemplo linhas e polígonos, que ilustram e descrevem características e funcionalidades na superfície terrestre. Os mapas podem ser vistos como “janelas” para as bases de dados, de forma a apoiar a pesquisa, análise e edição da informação. Este processo é designado por “geo-visualização”.

Modelo de visão Um SIG é um conjunto de ferramentas que transformam a informação geográfica segundo dados existentes. Este “geo-processamento” resulta da implementação de funções analíticas sobre dados existentes, originando nova informação.

Um mapa num SIG pode ser definido como um conjunto de imagens georreferenciadas com meta-informação. Os modelos mais comuns de representação da informação são o *raster* e o *vectorial*.

“Raster ou matricial” Grelha uniforme que representa a informação segundo um modelo celular, em que cada célula armazena o valor de um determinado atributo.

“Vector” O modelo é composto por formas geométricas (pontos, linhas e polígonos).

Os SIG estão presentes em vários sectores da sociedade que necessitem de processar informação georreferenciada. Sectores como o planeamento urbano, gestão das redes de transportes, agricultura, cartografia, criminologia, logística, marketing, são

exemplos de sectores que sofreram transformações com o aparecimento deste tipo de sistemas.

2.1.1 Software utilizado nos Sistemas de Informação Geográfica

Desde o início do aparecimento dos primeiros SIG, nos fins dos anos 70 e até aos dias de hoje, houve uma enorme evolução e difusão dos mesmos. Actualmente existe uma panóplia de sistemas que podem operar em plataformas heterogéneas. "O software SIG pode ser categorizado com ilustra a Figura 1" [Steiniger09]:

GIS tasks vs. GIS software category	viewing	creation	editing	storage	conflation	transformation	query	analysis	create maps
Desktop GIS									
Viewer	•			•			•		◊
Editor	•	•	•	•		◊	•		•
Analyst	•	•	•	•	•	•	•	•	•
Spatial DBMS				•		•	•	◊	
WebMap Server	•	◊	◊				•		•
Server GIS				•	•	•	•	•	•
WebGIS Client									
Thin Client	•						•		
Thick Client	•	•	•	•			•	•	•
Mobile GIS	•	•	•	•			•		
Libraries / Extensions				•	•	•		•	•

Figura 1- Categorização do software GIS [Steiniger09]

Como ilustra a Figura 1, o *Desktop GIS* agrupa o software com as funcionalidades de visualização (*GIS Viewer*), edição (*GIS Editor*) e análise (*GIS Analyst*). Para armazenar informação são utilizados bases de dados espaciais (Spatial Database Management Systems - DBMS). *WebMap Servers* são utilizados para "distribuir" mapas e informação pela Internet. Por sua vez, os clientes *WebGIS* são utilizados para consultar e visualizar a informação disponibilizada pelos servidores SIG. Existem ainda dois subgrupos: os "*Thin clients*" só realizam operações de consulta e visualização de informação geográfica, por exemplo mapas; e "*Thick clients*" que possuem ferramentas que permite a visualização e edição de informação geográfica. As bibliotecas são extensões que oferecem funcionalidades inexistentes no software SIG básico, como por exemplo funções para ler a informação num formato em particular. *MobileGis* funciona como repositório de informação no "terreno".

Seguidamente serão referenciados alguns exemplos de software SIG com alguma popularidade:

- **ESRI Suite of Products** [ESRI09c]: ArcInfo, ArcMap, ArcView, ArcSDE, ArcIMS, ArcGIS Explorer, software comercial produzido pela Environmental Systems Research Institute. Este software possibilita aos seus utilizadores a edição, análise, partilha e publicação de informação geoespacial.
- **GeoMedia**. Conjunto de componentes de software SIG da Intergraph².
- **Mapinfo** [MAPINFO09]: produto comercial Pitney Bowes MapInfo Corporation que fornece um conjunto de aplicações Desktop e de serviços que permitem, analisar, visualizar informação georreferenciada.
- **GRASS GIS** (Geographic Resources Analysis Support System) [GRASS09]: software OpenSource capaz de realizar mapeamento matricial, vectorial, processamento de imagem e informação gráfica. O GRASS foi um dos primeiros software GIS estando em constante desenvolvimento desde 1982 até aos dias de hoje.

2.1.2 Normalização da informação geográfica

O número de aplicações SIG tem crescido consideravelmente. Existe um variado leque de software envolvendo captura, manipulação, análise e visualização da informação geográfica. Os SIG são, actualmente, utilizados em diversas áreas, processando uma elevada quantidade de informação georreferenciada. No entanto, esta informação provém de múltiplas fontes, podendo diferenciar no formato do ficheiro, na resolução, no sistema de coordenadas e *datum*. Devido á proliferação de sistemas e serviços a processar informação georreferenciada, surge a necessidade de definir normas que possibilitem a integração e a interoperabilidade.

2.1.2.1 Open Geospatial Consortium

O OGC (Open Geospatial Consortium) [OGC09] anteriormente denominado por OpenGIS, é um consórcio internacional onde participam várias empresas, agências e universidades, com o objectivo de criar um consenso nas especificações desenvolvidas para interfaces espaciais de forma a desenvolver informação espacial e serviços complexos que possam ser utilizados livremente por qualquer tipo de aplicação.

WMS (Web Map Service) Protocolo *standard* utilizado para distribuir mapas com imagens através da Internet geradas por um servidor de mapas associado a uma base de dados espacial.

² <http://www.intergraph.com/>

WFS (Web Feature Service) Serviços cuja finalidade é padronização de uma interface de comunicação para a transmissão de informação geoespacial.

WCS (Web Coverage Service) É um serviço que permite a troca de informação geoespacial sob a forma de coberturas multi-dimensionais.

WPS (Web Processing Service) Tem como objectivo normalizar a forma como são disponibilizados os cálculos SIG através da internet.

CS-W (Catalog Service Web) Interface comum utilizado para pesquisar e consultar meta-dados sobre serviços e informação geoespacial.

SFS (Simple Features - SQL) Especifica o modelo de armazenamento da informação georreferenciada.

GML (Geography Markup Language) Linguagem de modelação baseada em XML para sistemas geográficos que serve também como forma de comunicação aberta para trocar informação geográfica.

KML (Keyhole Markup Language - versão 2.2) - Linguagem estruturada baseada em XML (*eXtensible Markup Language*) utilizada para modelar e armazenar informação com conteúdo geográfico, como por exemplo pontos, linhas, imagens e polígonos.

2.1.2.2 ISO/TC 211 (Geographic information/Geomatics)

ISO/TC 211 - 19100 series, é um conjunto de *standards* internacionais utilizados na partilha, processamento e análise de informação relacionada com uma localização relativa à Terra. “Estas normas podem especificar, para informação geográfica, os métodos, as ferramentas e serviços utilizados na gestão de dados (incluindo a definição e descrição), aquisição, processamento, análise, acesso, apresentação e transferência de tais dados no formato digital/electrónico entre os diferentes utilizadores, sistemas e localizações” [ISO21109]. A ISO/TC 211 partilha esforços com a OGC de forma a se obterem resultados idênticos, sendo alguns *standards* adoptados por ambas as organizações.

2.2 Sistemas de Informação Geográfica aplicados à Agricultura

Os SIG têm demonstrado ser uma ferramenta poderosa no planeamento e gestão de actividades agrícolas, como por exemplo na fertilização de solos, na redução da contaminação ambiental, na automatização de processos e na gestão de estimativas de produtividade. A facilidade na gestão de informação georreferenciada e a utilização dos

mapas temáticos proporcionados por estes sistemas, representam trunfos no bom planeamento da produção agrícola.

“A agricultura de precisão é um novo conceito na sustentabilidade e utilização dos recursos agrícolas. As principais características da agricultura de precisão estão na compreensão da variedade espacial, na identificação das razões dessa mesma variedade, na gestão da produção, na implementação de soluções de acordo com as especificidades do terreno e clima, na avaliação do tratamento de doenças que assolam as culturas agrícolas e no acumular de conhecimento de informação espacial com vista a facilitar a tomada de decisões. Novos desafios são lançados aos engenheiros para desenvolverem novas investigações e tecnologias de forma a promover a agricultura de precisão. Têm sido desenvolvidas novas abordagens, como o recurso à inteligência artificial, o uso de algoritmos genéticos, “*fuzzy controls*” e processamento de sinal digital. Grandes esforços estão a ser dirigidos no desenvolvimento de processos de integração destes sistemas de informação e no desenvolvimento de interfaces adequadas passíveis de serem utilizadas na prática da agricultura” [Maohua01].

São diversas as aplicações dos SIG na agricultura de precisão, nas mais variadas culturas e zonas do globo terrestre. Seguidamente são descritos dois exemplos onde foi possível obter vantagens com a adopção deste tipo de sistemas.

Wyoming, Estados Unidos da América. Recorreu-se aos SIG para determinar a viabilidade de produção de novas culturas de beterraba sacarina e de cevada. “Devido ao cultivo intensivo de beterraba apareceram novas pragas. Recorreu-se ao uso massivo de pesticidas mas não se obtiveram os resultados esperados. Uma das formas mais eficazes de combater as pragas é a rotatividade de culturas. Os SIG são utilizados para escolher os melhores locais de plantação de acordo com características ambientais, como por exemplo: temperatura, precipitação e qualidade do solo” [Young99]. A metodologia consistiu no estudo da topografia do terreno, da capacidade de irrigação, das características do solo e da precipitação. Este sistema possibilitou uma melhoria do meio ambiente devido ao menor uso defensivo, recorrendo a químicos, e uma maior disponibilidade de informação importante para a tomada de decisões.

Dinamarca, Europa. O sistema permite calcular o nível de nitrogénio nas zonas ribeirinhas, controlando assim a qualidade da água utilizada nas plantações aí situadas. “O aumento dos níveis de nitrogénio nas águas dinamarquesas é uma das causas principais da transformação da agricultura nesta região nos últimos quarenta anos. O sistema permite relacionar parâmetros como a qualidade da água (tendo em conta a quantidade de azoto existente) e o tipo de solo da plantação” [Skop98]. Desta forma foi possível realizar uma gestão eficiente dos recursos, adequando as necessidades dos diferentes tipos de culturas.

A aplicabilidade dos sistemas de informação geográfica na agricultura é vasta. No entanto, tendo em conta o enquadramento da dissertação será realizada uma focalização numa área em particular, a aplicação destes sistemas à viticultura.

A “viticultura de precisão” tem como objectivo otimizar a performance das vinhas, aumentando a sua produção e qualidade, de acordo com as especificidades de cada região. Os Australianos lideram a investigação e desenvolvimento da “viticultura de precisão”. Apesar, de os viticultores dos países da Europa ocidental e norte-americanos começarem a desempenhar um papel importante na área, libertando-se das suas raízes e práticas tradicionais. As tecnologias envolvidas nas “viticultura de precisão” são diversificadas, como por exemplo: sensores que controlam as condições ambientais, estações de meteorologia, GPS (Global Position Systems), Remote Sensing, DEM (Digital Elevation Models), sistemas topográficos e os SIG. Estes sistemas têm sido desenvolvidos com o intuito de prevenir doenças, melhorar o sistema de irrigação das vinhas, aumentar a produção, melhorar a qualidade da uva e do vinho produzido, melhorar a fertilização, minimizar os custos de produção e a contaminação provocada pelo uso excessivo de fertilizantes e pesticidas. Os SIG desempenham um papel relevante na “viticultura de precisão”. Com este sistemas é possível armazenar informação espacial, apresentar informação em mapas temáticos, tabelas e folhas de cálculo, de modo a promover a qualidade das uvas produzidas e facilitar aos viticultores a tomada de decisões. “Uma das formas de descrever a informação espacial na viticultura consiste na utilização de mapas temáticos que contenham uma elevada quantidade de informação. Estes mapas possuem informação detalhada, tornando complexa a sua representação nos mapas de papel estáticos tradicionais” [Smith98]. Os mapas computadorizados podem representar múltiplos temas em camadas e armazená-los em múltiplas bases de dados. Como é possível observar na Figura 2 é possível realizar uma análise multi-critério (variedade da casta da uva, parcelas com vinha, qualidade do solo, irrigação e drenagem do terreno), sendo o resultado final constituído pela conjugação dos diversos factores. Isto é visível na Figura 3.



Figura 2 - Base de dados multi-critério [Smith98]

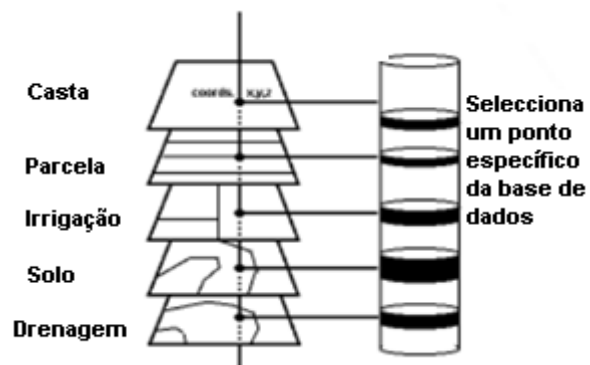


Figura 3 - Selecção de informação baseado em parâmetros multi-critério [Smith98]

A possibilidade de analisar informação espacial conjuntamente com a integração de outro tipo de informação, como por exemplo as características climáticas da região ou a topologia do terreno, entre outros fenómenos geográficos, facilita a compreensão da interacção entre os fenómenos naturais e as práticas de gestão das vinhas. Esta relação conduz a melhoria do produto final.

Com estes sistemas é possível acompanhar e registar todo o ciclo de crescimento das castas de uvas, desde a plantação até a sua colheita relacionado os factores ambientais com os factores de manutenção. Isto é visível na Figura 4.

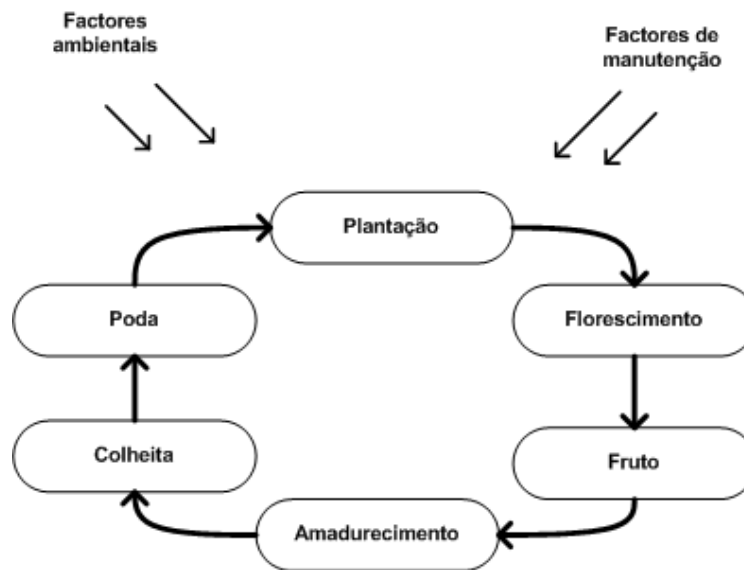


Figura 4 - Modelação do ciclo de crescimento das castas de uvas [Smith98]

Baseados neste conjunto de informação é possível desenvolver modelos de processos utilizados no desenvolvimento de sistemas de suporte a decisão (*Decision Support Systems*) que auxiliem a tomada de decisões por parte dos viticultores.

Nas secções seguintes são descritos alguns exemplos de SIG aplicados na viticultura, utilizando diferentes abordagens.

2.2.1 Na Toscana (Itália), novos estilos na gestão de vinhas provenientes da "agricultura de precisão"

A Toscana é mundialmente reconhecida pela sua produção de vinho. Sendo esta caracterizada pela utilização de técnicas ancestrais transmitidas ao longo de várias gerações. No entanto, alguns produtores decidiram introduzir os SIG com o objectivo de facilitar o processo de tomada de decisão. Na Figura 5 pode observar-se a delimitação das parcelas com vinha.



Figura 5 - Informação das parcelas com vinha [ESRI09b]

O SIG foi construído com a função de auxiliar na gestão das vinhas, principalmente na delimitação de terrenos, distinguindo as vinhas das restantes áreas de cultivo. Também foi desenvolvido outro sistema com o objectivo de auxiliar o planeamento estratégico das vinhas, permitindo a realização de simulações sobre a plantação de novas vinhas prevendo os custos associados. O sistema foi implementado com o recurso ao software ArcView [ArcView09] permitindo deste modo a visualização, a análise e a gestão de dados geográficos. Futuros desenvolvimentos irão permitir o uso de GPS e do ArcPad de forma a poder obter novas funcionalidades passando de um *Desktop GIS* para um *Mobile GIS*.

2.2.2 British Columbia (Canadá), utilização de SIG como forma de melhorar a produtividade das vinhas nos vales de Okanagan e Similkameen

A complexidade geológica do terreno presente nas regiões dos vales de Okanagan e Similkameen, deve-se ao solo o facto de possuir distintas origens vulcânicas e clima característico, o que dificulta a previsão dos melhores espaços para produção e gestão vitícola.

O SIG foi desenvolvido com o intuito de estudar a relação entre as condições locais, as praticas de gestão e a performance da vinha tendo em conta a qualidade da uva e vinho produzido. A metodologia de desenvolvimento passou por recolher informação sobre as necessidades específicas da vinha, de acordo com o local onde está situada. Desta forma, foi possível construir um modelo de dados que permitia relacionar as características do solo, do clima, das práticas de gestão (por exemplo o tipo de enxertos utilizados, espaçamento entre cada videira, equipamento de irrigação), com o tipo de uva produzido. O levantamento do terreno foi realizado utilizando um sistema GPS. Com base nos dados recolhidos foi possível construir mapas delimitando as áreas dos terrenos bem como a modelação e parametrização do terreno (exposição solar, temperatura ambiente e elevação. A Figura 6 ilustra a representação dos vales de acordo

com a diferenciação das zonas ao longo do terreno de acordo com a temperatura ambiente.

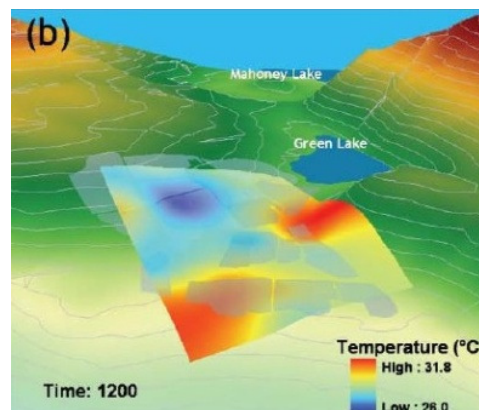


Figura 6 - Diferenciação das zonas de acordo com a temperatura [Bowen05]

Com a aplicação deste sistema foi possível determinar quais as melhores práticas a aplicar a cada vinha, tendo em conta as variedades de cada região e as diferentes qualidades de vinho produzidas.

2.2.3 (Austrália), ARvino

O ARvino consiste num protótipo que permite visualizar informação georreferenciada utilizando dispositivos directamente no terreno, rompendo com as tradicionais aplicações Desktop, em que a visualização e análise da informação têm lugar nos escritórios dos viticultores. O sistema auxilia os viticultores a determinar com precisão os parâmetros que afectam a qualidade das uvas nos diferentes campos de cultivo de vinhas. O ARvino disponibiliza a visualização da informação recorrendo a Realidade Virtual (ver Figura 7) e Aumentada (ver Figura 8).



Figura 7 - Realidade Virtual [King05]

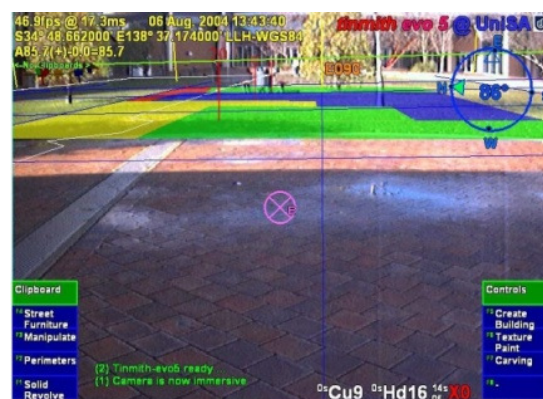


Figura 8 - Realidade Aumentada [King05]

De acordo com a informação contida nos *shape files* foi efectuado o mapeamento topológico, utilizando NDVI (Normalized Difference Vegetation Index)

Este é um indicador numérico que permite distinguir no alvo a ser observado, aquilo que é ou não vegetação.

A plataforma de Realidade Aumentada apresentou alguns problemas devido à dificuldade em aplicar transparências aos objectos virtuais 3D e dificuldades em visualizar o conteúdo apresentado no ecrã devido a incidência da luz solar no monitor do dispositivo. A solução encontrada para minimizar o problema foi a utilização de cores saturadas.

2.2.4 Valdepenas (Espanha), estudo da diferenciação em zonas vitícolas na Denominação de Origem Valdepenas

Os responsáveis executivos pela Denominação de Origem Valdepenas, necessitavam de um sistema que fosse capaz de gerir a informação vitícola, realizar ordenamento do território e otimizar a capacidade produtiva das vinhas. Decidiram iniciar uma proposta de actividade científica denominada de "Estudo de Zonificação e Vitícola na Denominação de Origem de Valdepenas", onde os principais intervenientes foram a Universidade de Castilla-La Mancha e algumas organizações agrárias locais. A ideia do estudo é desenvolver um sistema que permita auxiliar no planeamento estratégico recorrendo aos sistemas de informação e as novas tecnologias. Tendo como suporte um SIG o objectivo do projecto determinar zonas diferente aptidão para o cultivo de vinhas, combinando factores edafo-climáticos, topográficos e agrónomos.

“Para uma correcta delimitação e caracterização vitícola, é necessário estabelecer também a correcta correlação entre a qualidade do produto e condições do meio onde foi produzido. Como este problema não está completamente resolvido é necessário adoptar critérios de comparação regional e conhecer as metodologias para resolver problemas semelhantes em regiões distintas” [Riquelme05].

A metodologia na divisão de tarefas possui três grandes blocos:

1. Identificação, realização da cartografia da área de estudo.
2. Caracterização do território na Denominação de Origem.
3. Análise multi-critério permitindo a diferenciação dos vários vinhos agrupando-os em diferentes classes vitícolas de produção.

O desenho conceptual do SIG permite a caracterização vitícola, utilizando técnicas de análise multi-critério, tendo principalmente em conta as características do terreno, do clima e da vegetação. Os dados necessários para alimentar o sistema foram obtidos de diversas instituições, como por exemplo o Instituto Geográfico Nacional e o Cadastro Digital Rústico dos municípios que integram a zona Denominação de Origem de Valdepenas.

Após as primeiras aplicações do sistema, foi notório a variedade de diferentes zonas de produção existentes na Denominação de Origem. Foi possível disponibilizar mapas temáticos e resultados num contexto administrativo, interactivo, aberto, mediante consultas online, ver Figura 9.

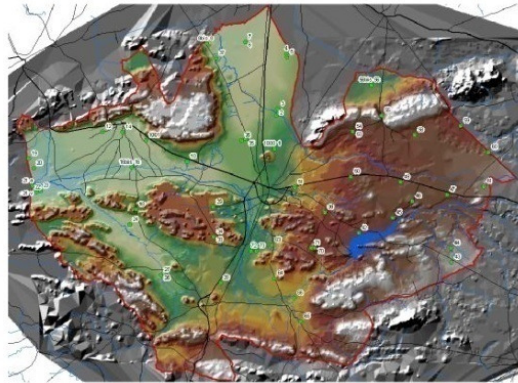


Figura 9 - Localização das parcelas de mosto [Riquelme06]

O SIG facilitou a análise multi-critério considerando factores edafo-climáticos, critérios espaciais, critérios temporais, definição de escalas e de parâmetros de avaliação indispensáveis para uma correcta diferenciação das zonas.

2.2.5 Sumário

Neste capítulo foi abordada da temática dos SIG, onde foi descrito o seu modo de funcionamento, quais as normas comuns utilizadas na normalização da informação com vista a garantir a interoperabilidade entre os vários sistemas. Seguidamente, foi introduzido o conceito de “agricultura de precisão”, onde foram apresentados vários exemplos de implementações deste tipo de sistemas em diversas regiões do globo.

Os SIG aplicados à viticultura estão intimamente ligados à região onde são aplicados. Os sistemas referidos nos tópicos anteriores iniciam o processo de desenvolvimento, efectuando um levantamento topográfico e realizando um estudo sobre os factores edafo-climáticos que afectam a região, como por exemplo, altitude, exposição, pedregosidade, castas, idade da parcela e produtividade. A obtenção de informação geográfica realizada durante o processo de levantamento e recolha de dados pode ser auxiliada pela utilização de sistemas externos, como por exemplo GPS (ver secção [2.2.2](#)). Assim como, pela utilização de dados provenientes de organizações agrárias locais (ver secção [2.2.4](#)). Após terminado o processo de recolha de informação segue-se a implementação do SIG propriamente dito.

Os SIG têm como vantagem a capacidade de cruzar várias fontes de informação, apresentando-a em mapas temáticos, facilitando a tomada de decisões.

3. SIG Web e tecnologias Web

Nos últimos anos o uso da Internet tem crescido exponencialmente, transformando a forma de comunicar e de transmissão de conhecimento. Os SIG não foram indiferentes a este fenómeno.

“ *Web GIS* tornou-se num método barato e de fácil utilização, contribuído assim para a disseminação da informação geoespacial e ferramentas de geoprocessamento SIG” [Alesheikh02].

A distribuição de mapas através de serviços de *Web mapping* de baixo custo estruturou a utilização dos SIG. Já não é necessário recorrer a software proprietário e dispendioso, com elevados recursos computacionais, para implementar um SIG. O factor económico encoraja a investigação e desenvolvimento de aplicações SIG recorrendo a tecnologias Web.

No entanto, os SIG baseados em tecnologias Web não estão isentos de problemas, necessitam de ligações à Internet relativamente rápidas devido ao facto de estarem presentes trocas de informação geográfica em quantidade considerável. As aplicações Web não disponibilizam mapas com o mesmo nível de detalhe e resolução das aplicações SIG tradicionais. Como forma de minimizar esta limitação, as aplicações *Web GIS* disponibilizam controlos que permitem navegar pelo mapa e aplicar vários níveis de zoom. Nos últimos anos, de uma forma generalizada a velocidade das ligações à Internet tende a aumentar, tornando estes problemas menos relevantes.

No presente capítulo é efectuado um estudo sobre os sistemas *Web GIS* apresentando a sua arquitectura e os métodos utilizados no mapeamento Web. Seguidamente, é justificado o potencial interesse das RIA na implementação de aplicações Web descrevendo o seu modo de funcionamento, metodologias de desenvolvimento e soluções tecnológicas actuais.

3.1 Arquitectura

Um SIG baseado em tecnologias Web possui uma arquitectura do tipo cliente/servidor. As responsabilidades do geoprocessamento da informação são repartidas pelo cliente e pelo servidor. Isto é visível Figura 10.

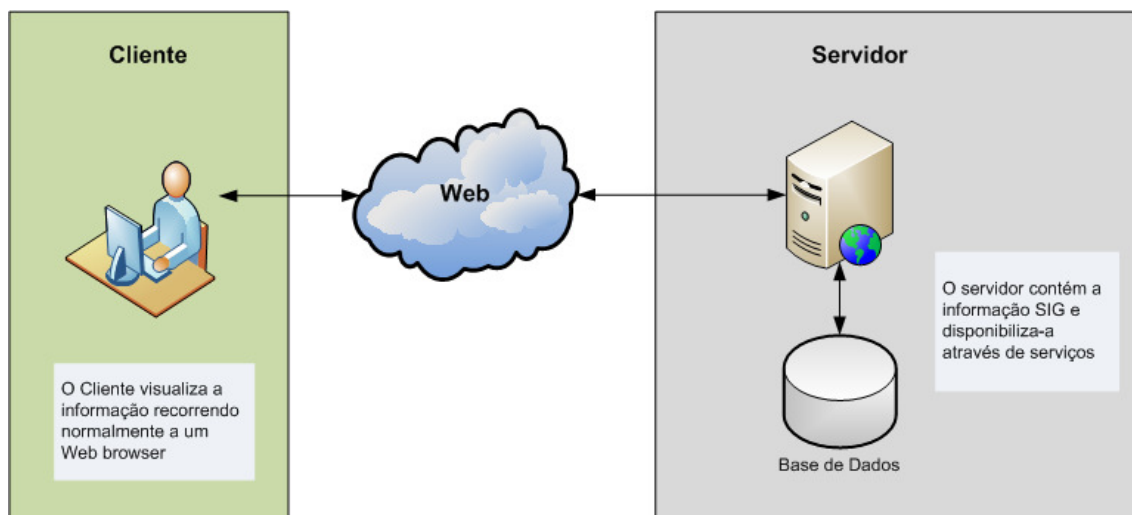


Figura 10 - Arquitectura SIG Web

O servidor é constituído por um sistema de gestão de base de dados e por software GIS, sendo também responsável pela gestão dos serviços a publicar. O cliente consome os serviços disponibilizados pelo servidor, construindo a representação da informação georreferenciada normalmente recorrendo a Web Browser. Existem algumas variantes da arquitectura apresentada na figura anterior, no entanto, há duas que se destacam: a *thick client* e *thin client*. Ambas apresentam vantagens e desvantagens que serão discutidas nos dois sub-tópicos seguintes.

3.1.1 Aplicações Thin client

Na variante *thin-client* ou *server side applications*, o cliente apenas tem como funcionalidades para comunicar com o servidor e apresentar os resultados. Todo o processamento e gestão da informação são da responsabilidade do servidor. Isto é visível na Figura 11.

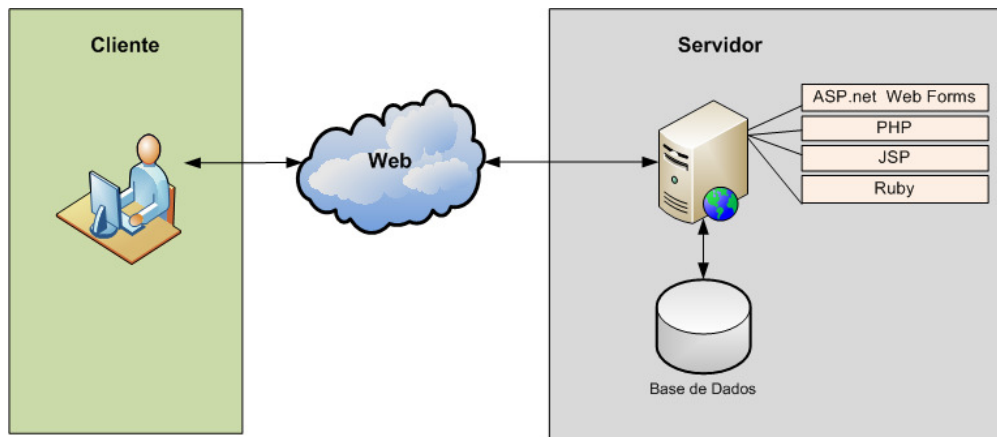


Figura 11 - Server Side Application

Esta arquitectura centrada no servidor tem como vantagens a facilidade na actualização de informação, a centralização do controlo das operações, a possibilidade de integração com outro software sem que isso represente um problema para o cliente, a facilidade na gestão da integridade e a segurança da informação. As principais desvantagens são o tempo de resposta elevado, necessidade de redes com baixo nível de latência, os servidores necessitarem de ter uma elevada performance pois são responsáveis pela maioria do processamento, os clientes não poderem trabalhar em modo *off-line* estando totalmente dependentes do servidor, menos interactividade e limitação na apresentação de conteúdos multimédia, como por exemplo vídeo.

3.1.2 Aplicações Thick client

Na variante *thick client* ou *client side applications* as responsabilidades do processamento são também partilhadas com o cliente. Isto é visível na Figura 12. As ligações ao servidor são reduzidas, podendo realizar tarefas sem que seja necessário a validação por parte do servidor.

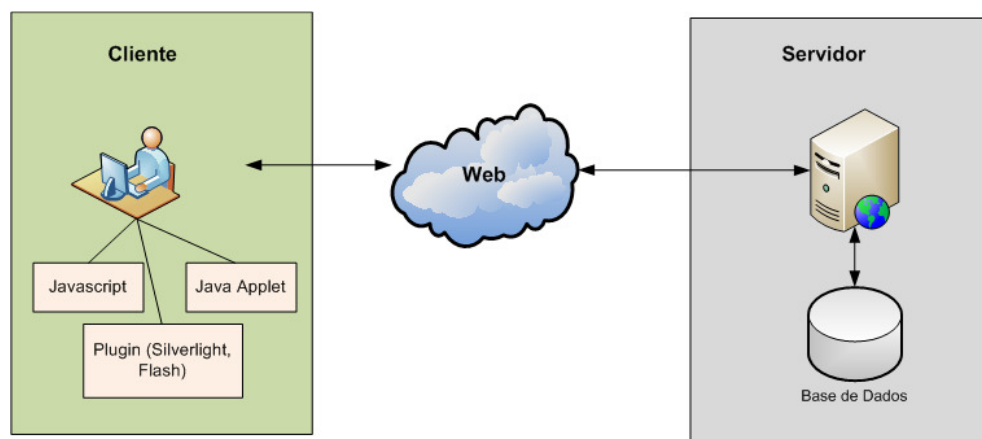


Figura 12 - Client Side Applications

As principais vantagens desta abordagem são: melhoria da interactividade e usabilidade, possibilidade de trabalhar *off-line*, melhor performance multimédia (como o processamento é partilhado pelos vários clientes os servidores não necessita de possuir uma elevada performance), melhoria na utilização dos recursos da rede devido à menor solicitação dos servidores, representação de conteúdos no formato vectorial e possibilidade de representar outros formatos de imagens além do JPEG e GIF. No entanto, esta abordagem não está isenta de críticas, dado as aplicações necessitarem de instalar um *Plugin* adicional ou transferir código para o lado do cliente, podendo representar uma limitação pois pode não ser compatível com todos os browsers e sistemas operativos. O facto de o processamento estar distribuído pelos diversos clientes pode trazer problemas no que diz respeito à integridades da informação armazenada no servidor.

3.2 Mapeamento Web

A utilização de serviços Web que disponibilizam informação georreferenciada tem ganho expressão no desenvolvimento de aplicações que auxiliam a tomada de decisão. Serviços de *Web mapping*, como por exemplo o Google Maps³ e o Live Search Maps⁴, incentivam a implementação de aplicações *Web GIS* fornecendo gratuitamente ou a baixo custo informação georreferenciada e disponibilizando aos programadores uma API (Application Programming Interface), com vista a facilitar o desenvolvimento de aplicações. Os serviços de *Web mapping* tecnologicamente mais evoluídos permitem a visualização da mesma localidade utilizando diferentes níveis de zoom e perspectivas. Podendo ainda interagir directamente com o mapa através do desenho de pontos, linhas, polígonos e inserção de imagens. Como pode ser observado na Figura 13.

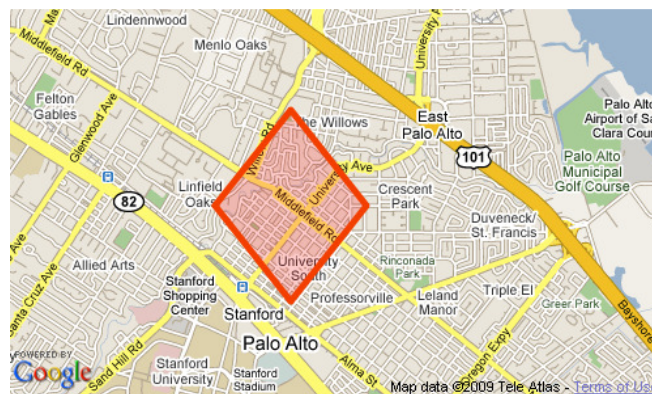


Figura 13 - Polígono Google Maps [GOOGLE09]

³ “Google Maps” - <http://maps.google.com/>

⁴ “Live Search Maps” - <http://maps.live.com/>

Alguns serviços fornecem ainda opções avançadas, como por exemplo, o cálculo da distância e tempo dos percursos entre duas localidades tendo em conta o meio de transporte e bem como a possibilidade de acompanhar o congestionamento do tráfego de algumas cidades do globo em tempo real.

3.2.1 Projecção

Por norma, estes serviços realizam a projecção dos mapas utilizando a projecção Mercator. Esta projecção cilíndrica representa em escala verdadeira a linha do Equador, os meridianos geográficos são representados por rectas paralelas entre si, sendo simultaneamente perpendiculares ao Equador, isto é visível na Figura 14. Apesar da projecção Mercator apresentar alguma distorção no que diz respeito à escala, tem como vantagem a capacidade de preservar a forma dos objectos. Aspecto relevante, por exemplo, na representação de edifícios quando se trabalha com diferentes níveis de *zoom*.

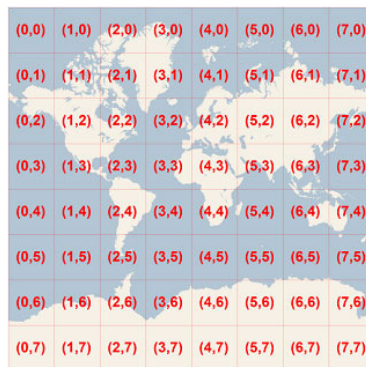


Figura 14 - Projecção Mercator [MSOFT09b]

3.2.2 Mashup

A disseminação de informação geográfica recorrendo aos serviços de *Web mapping* tem contribuído para o conceito de *Mashup* presente nas aplicações Web 2.0⁵. “Este conceito enquadra-se nos sites Web ou serviços que reúnem informação de várias fontes e transformam-na em nova informação ou serviços. Os *Mashup* estão em verdadeira expansão, especialmente no contexto de combinar informação geográfica e apresentá-la, de forma integrada em mapas” [Maged08]. Várias empresa e organizações que lutam pelo domínio do mundo da informação geoespacial na Web, sendo constantes

⁵ Os principais conceitos das aplicações Web 2.0: interoperabilidade, interactividade, partilha de informação e suporte multimédia. A Web 1.0 corresponde as aplicações Web tradicionais, normalmente caracterizadas por serem constituídas por sites estáticos, de fraca interactividade e desprovidos de dinamismo.

as novidades apresentadas na ânsia de cativar o maior número de utilizadores e explorar novas oportunidades de negócio.

3.3 Rich Internet Applications

Nos últimos anos a Web ganhou expressão como "plataforma" para o desenvolvimento de aplicações. No entanto, devido ao aumento do nível de exigência por parte dos utilizadores, a complexidade das aplicações aumentou, levando a que as tecnologias Web tradicionais comecem a demonstrar limitações no que respeita a interactividade e usabilidade.

"As *Rich Internet Applications* são aplicações Web que tiram partido do processamento realizado pelos clientes de forma a melhorar a capacidade de resposta e a usabilidade das *User Interfaces*, oferecendo funcionalidades semelhantes às das aplicações *Desktop*" [Brambilla08].

A ideia principal das RIA é reproduzir as funcionalidades e comportamento das aplicações *Desktop* na Web. As RIA procuram combinar o melhor de dois mundos, a arquitectura distribuída e os baixos custos de manutenção proporcionada pelas aplicações Web com as capacidades de interactividade e multimédia presentes nas aplicações *Desktop*.

3.3.1 Modo de funcionamento

As aplicações RIA enquadram-se na categoria *thick client* descrita na secção [3.1.2](#). "RIA seguem o paradigma cliente/servidor, mas contrariamente às aplicações Web tradicionais, as RIA são capazes de transferir o processamento da *User Interface* e da camada de negócio e a gestão da informação para o lado do cliente, com a possibilidade de utilizar a comunicação assíncrona com o servidor" [Brambilla08].

Nas aplicações Web tradicionais todo o processamento é realizado no lado do servidor, o cliente apenas apresenta um ou vários ficheiros HTML com conteúdo essencialmente estático. Este processo sobrecarrega os servidores pois toda a interacção com a aplicação e gestão da informação tem de passar pelo servidor. Como ilustra a Figura 15 o cliente envia um pedido ao servidor e em seguida o servidor envia a resposta ao cliente, que o cliente actualiza a página por inteiro.

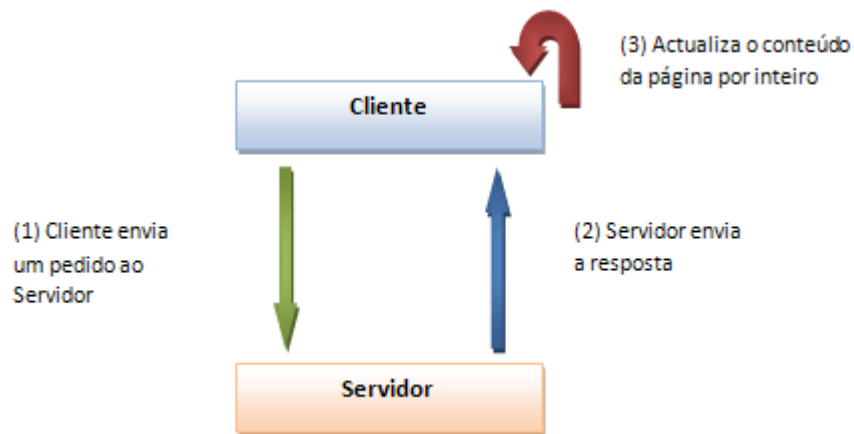


Figura 15 - Arquitectura cliente-servidor das aplicações Web tradicionais

Segundo o modelo das RIA servidores ficam mais libertos, diminuindo as transacções entre cliente e servidor, ver Figura 16.

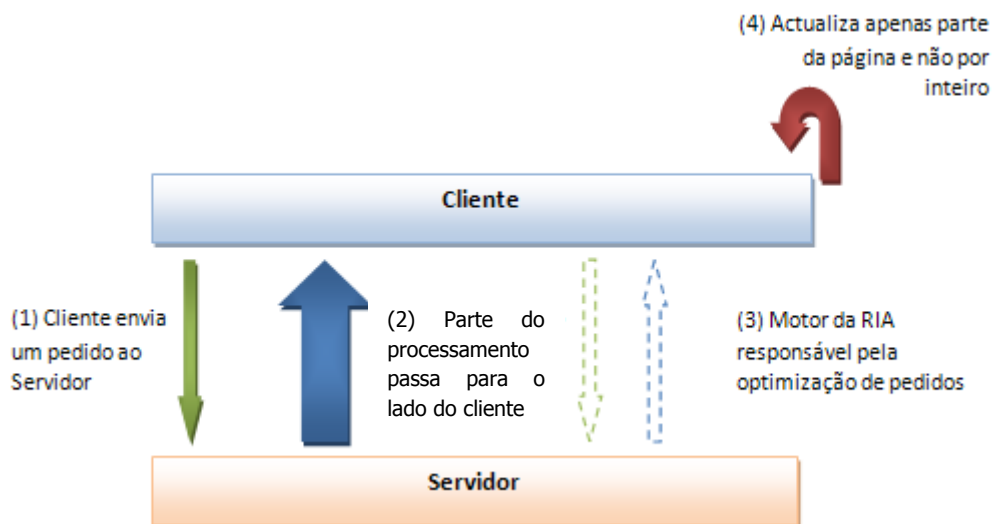


Figura 16 - Comunicação entre cliente e servidor utilizando RIA

Grande parte da informação é transferida para o cliente quando o browser carrega o conteúdo da página pela primeira vez. A actualização de conteúdos também difere das aplicações Web tradicionais, apenas actualiza partes da página e não a página por inteiro diminuindo os tempos de resposta na interacção com o utilizador. O motor da RIA encarrega-se por gerir e otimizar a actualização dos conteúdos.

A Tabela 1 compara sucintamente as RIA com as aplicações Desktop e Web convencionais. As funcionalidades em comparação são a interoperabilidade, a capacidades de interacção e a gestão da informação. É visível a semelhança entre as RIA e as aplicações Desktop, diferenciando apenas no processo de instalação da aplicação.

Funcionalidade	Desktop	Web	RIA
Cliente universal (browser)	Não	Sim	Sim
Instalação no cliente	Complexo	Simples	Simples
Capacidade de interacção	Rico	Limitado	Rico
Lógica de negócio no lado do servidor	Sim	Sim	Sim
Lógica de negócio no lado do cliente	Sim	Limitado	Sim
Necessita de actualizar todo o conteúdo da página	Não	Sim	Não
Acessos frequentes ao servidor	Não	Sim	Não
Comunicação <i>server-to-client</i>	Sim	Não	Sim
Possibilidade de trabalhar de forma desligada (<i>disconnected</i>)	Sim	Não	Sim ⁶

Tabela 1 - Comparação entre aplicações Desktop, Web 1.0 e RIA [Bozzon06]

3.3.2 Conceitos implícitos no desenho de *Rich Internet Applications*

O desenvolvimento e desenho de aplicações Web tradicionais apresentam os seguintes problemas [Preciado05]:

- **Problemas de navegação:** por vezes para completar uma tarefa simples, como por exemplo encomendar um livro, é necessário percorrer várias páginas.
- **Gestão da informação:** não suportam a exploração interactiva da informação. Normalmente o utilizador tem de navegar por vários formulários para encontrar a informação desejada.
- **Problemas de feedback:** não permite uma interacção contínua sem ter de actualizar a página por inteiro.

Os conceitos implícitos no desenho de RIA resolvem os problemas descritos nos tópicos anteriores. Na Figura 17 estão representadas as camadas presentes numa aplicação típica envolvendo RIA.

⁶ Pode trabalhar de forma desligada, no entanto, tem de estar ligado pelo menos uma vez quando pela primeira vez transfere o conteúdo para o cliente.

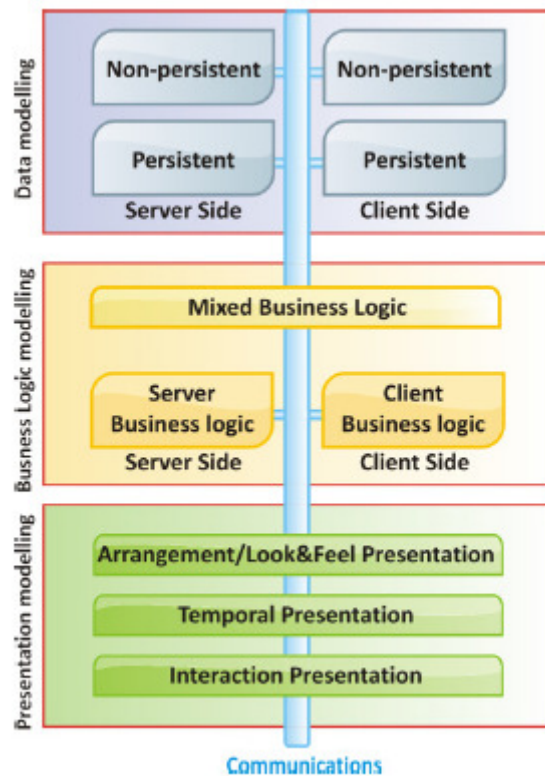


Figura 17 - RIA design [Preciado07]

Modelação dos dados (Data modeling): Com as RIA é possível utilizar a memória volátil ou persistente existente no cliente. O conteúdo pode ser armazenado e manipulado no lado do cliente sendo posteriormente enviado para o servidor após a operação estar concluída. “Este fluxo de dados também se verifica noutros campos da Engenharia de Software, sob o nome (*temporal*) *disconnected applications*” [Preciado07]. O uso da memória persistente no cliente está limitado consoante o tipo de RIA em utilização. Porém o facto de a informação estar distribuída requer mecanismos adicionais para garantir a integridade do modelo de dados, a integridade e a segurança da informação.

Modelação da camada de negócio (Business Logic modelling): a estrutura de navegação das RIA difere das aplicações Web 1.0, operando como *single page applications*. O cliente pode realizar operações de maior complexidade, acompanhado pela possibilidade de camada de negócio poder estendida pelo cliente e servidor.

Modelação da camada de apresentação (Presentation modeling): “a apresentação nas RIA concentra-se na definição do esboço da página, definição de estilos e na captura de comportamentos da aplicação” [Preciado07]. As RIA inovam comparativamente as aplicações Web 1.0 na interacção, na usabilidade e no suportam por defeito de conteúdo multimédia. As aplicações Web convencionais para apresentar interacções ricas como por exemplo *drag&drop*, reproduzir elementos multimédia,

necessitam de recorrer a código Javascript fortemente dependente do browser afectando a interoperabilidade da aplicação.

Comunicação (Communications): Não sendo uma camada isolada a troca de informação é transversal a todo o modelo da RIA. Na Figura 17 pode observa-se que o fluxo de dados é realizado dentro da mesma camada ou com as camadas adjacentes. Nas aplicações Web tradicionais a comunicação entre cliente e o servidor é síncrona. O cliente toma a iniciativa realizando um pedido ao servidor aguardando pela resposta. Nas RIA é possível utilizar a comunicação síncrona e assíncrona. A comunicação assíncrona possibilita várias transferências de informação em simultâneo entre o cliente e servidor sem ter de esperar que todas as transferências terminem na sua totalidade. Uma gestão eficiente dos mecanismos de comunicação é um ponto crucial para o bom funcionamento e sincronização da informação. O fluxo de dados pode ser despoletado pelo cliente (*pulling mode*) ou pelo servidor (*push mode*) através de eventos.

3.3.3 Metodologias

A utilização de RIA tem crescido de forma exponencial, porém existe um vazio no que diz respeito a metodologias de desenvolvimento, tanto na criação de novas aplicações, como na adaptação das aplicações Web 1.0 para aplicações Web 2.0.

Tem-se reunido esforços na criação e adopção de metodologias no desenvolvimento Web com o intuito de auxiliar na implementação, na organização e manutenção de sistemas baseados em tecnologias Web. No estudo realizado por [Preciado05], avaliou-se as metodologias existentes para aplicações Web 1.0 tendo como objectivo demonstrar se estas respondem às necessidades do desenvolvimento de RIA, ver Tabela 2.

Interacção	Possibilidade de especificar os comportamentos do utilizador
Multimédia	Possibilidade de suportar a representação de gráficos, áudio, vídeo, <i>streaming</i> multimédia
Ferramenta CASE⁷	Disponibilidade de uma ferramenta CASE no suporte da metodologia
Continuidade visual	Possibilidade de evitar actualização de conteúdo do ecrã por inteiro
Sincronização	Representação activa (relacionado com a interacção directa do utilizador) e passiva (relacionado com comportamentos predefinidos) dos elementos da interface do utilizador

⁷ Computer Aided Software Engineering

Desenvolvimento por camadas	Possibilidade de utilizar camadas separadas
Recuperação de informação de forma dinâmica	Possibilidade de fornecer/utilizar informação do servidor em <i>runtime</i>
Pedidos paralelos de origens diferentes	Possibilidade de recolher informação de origens diversas em simultâneo de forma assíncrona ou síncrona
Personalização	Opções de idioma e de região, acessibilidade, multi-dispositivo
Colaboração interactiva	Permite colaboração interactiva em tempo real por vários utilizadores com o objectivo de trabalharem na mesma tarefa em simultâneo

Tabela 2 - Parâmetros de comparação [Preciado05]

Foram definidos quatro níveis tendo como finalidade agrupar os resultados mediante o grau de cobertura dos parâmetros descritos na Tabela 2 por parte das metodologias Web. Os níveis são os seguintes:

- *Desired Coverage Degree (DES)*: Parâmetro corresponde idealmente aos requisitos do desenho de RIA. Foi atribuído peso 4.
- *Partial Coverage Degree (PAR)*: Determinado parâmetro não cobre na totalidade determinada funcionalidade no desenho de RIA. Foi atribuído peso 3.
- *Limited Coverage Degree (LIM)*: O parâmetro apresenta um suporte muito limitado relativamente a um determinado requisito de desenho de RIA. Foi atribuído peso 1.
- *None Coverage Degree (NON)*: O parâmetro não cobre determinado requisito no desenho de RIA. Foi atribuído peso 0.

No estudo realizado por [Preciado05] foram analisadas as seguintes metodologias e modelos Web: Hypertext Design Model, Web Site Design Method, Relationship Management Methodology, Hypermedia Model Based on Statecharts/Method, Object-Oriented Hypermedia Design Method, UML-based Web Engineering, Object-Oriented Hypermedia, Web Modeling Language e W2000.

“Hypertext Design Model ” (HDM) Modelo pioneiro na modelação da topologia da rede “*hipertextos*”. Modelo proposto como uma extensão do modelo Entidade-Relacionamento (Modelo E-R).

“Web Site Design Method” (WSDM)⁸ Modelo baseado no seguinte ciclo de vida: o modelo de desenho do utilizador, o desenho conceptual, o desenho de implementação e a implementação propriamente dita.

“Relationship Management Methodology” (RMM) Modelo que consiste num conjunto de sete passos que podem ser executados de forma iterativa ou em paralelo. Os passos são os seguintes: Representação do domínio da informação segundo um modelo E-R, divisão do diagrama principal em sub-diagramas, desenho de todos os caminhos que vão permitir a navegação no “hipertexto”, conversão dos vários diagramas num objecto para uma determinada plataforma, desenho da interface do utilizador, desenho dos comportamentos em tempo de execução, implementação acompanhada com a fase de testes.

“Hypermedia Model Based on Statecharts/Method” (HMBS/M) Modelo baseado em máquinas de estado. Define a estrutura do conteúdo e a navegação de um sistema Web, realizando a separação da estrutura da informação.

“Object-Oriented Hypermedia Design Method” (OODHM) Deriva da HDM acrescentando conceitos Orientados a Objectos. Utiliza mecanismos de composição e abstracção especificando padrões de navegação e transformação das interfaces do utilizador.

“UML-based Web Engineering” (UWE)⁹ Baseado no modelo Orientado a Objectos utilizando UML.

“Object-Oriented Hypermedia” (OOH)¹⁰ Metodologia utilizada para modelar e implementar processos de aplicações Web derivadas de abordagens Orientadas a Objectos.

“Web Modeling Language” - (WebML)¹¹ Descrição de aplicações Web baseadas num modelo de dados, modelo de “*hipertexto*”, modelo de apresentação e modelo de personalização. Os principais objectivos são: descrição de alto nível da estrutura da aplicação Web, fornecer múltiplas vistas do mesmo conteúdo, armazenar a meta-informação reunida durante o processo de desenho e permitir a especificação das operações de manipulação de informação (actualização de conteúdos, especificação dos processos de *workflow* e interacção com *Web Services*).

W2000 “ Estende o HDM com uma nova camada de negócio. Os Designers podem especificar operações invocadas pelo utilizador ou pelas transacções lógicas que definem os serviços fornecidos pela aplicação. Recorrendo a um meta-modelo “*Meta-*

⁸ “WSDM website” - <http://wsdm.vub.ac.be/>

⁹ “UWE – UML-based Web Engineering” - <http://uwe.pst.ifi.lmu.de/index.html>

¹⁰ “Object-Oriented Hypermedia” - http://gplsi.dlsi.ua.es/iwad/ooh_project/index.htm

¹¹ “Web Modeling Language” - <http://www.webml.org/>

Object Facility”, é possível realizar uma definição precisa e flexível de todos os conceitos de modelação” [Preciado05].

	HDM	WSDM	RMM	HMBS	OODHM	UWE	OO-H	WebML	W2000
Interacção	LIM	LIM	LIM	LIM	PAR	PAR	PAR	PAR	LIM
Multimédia	NON	NON	NON	LIM	NON	NON	NON	NON	NON
Ferramenta CASE	LIM	NON	LIM	LIM	PAR	PAR	PAR	PAR	PAR
Continuidade visual	NON	NON	NON	NON	NON	NON	NON	NON	NON
Sincronização	NON	NON	NON	NON	NON	NON	NON	NON	NON
Desenvolvimento por camadas	PAR	PAR	PAR	PAR	PAR	PAR	PAR	PAR	PAR
Recuperação de informação de forma dinâmica	PAR	LIM	LIM	LIM	PAR	LIM	PAR	PAR	PAR
Pedidos paralelos de origens diferentes	LIM	LIM	LIM	LIM	LIM	LIM	LIM	PAR	LIM
Personalização	LIM	LIM	LIM	LIM	LIM	PAR	PAR	PAR	LIM
Colaboração interactiva	NON	NON	NON	NON	NON	LIM	LIM	LIM	NON
Totais (pesos)	10	7	8	9	14	15	17	19	12

Tabela 3 - Comparação entre os requisitos RIA e as metodologias Web[Preciado05]

Analisando a Tabela 3 observa-se que o suporte multimédia, a modelação de processo de sincronização e a continuidade visual não são suportadas. Outra lacuna está na inexistência de uma ferramenta CASE consiga gerar automaticamente RIA, baseada em modelos Web. Apesar da falta de suporte multimédia e sincronização o WebML demonstrou ser um modelo extensível, flexível e o que mais se aproxima das necessidades de modelação impostas pelas RIA.

“A comunidade Web tem apresentado metodologias e ferramentas de forma a suportar o design, o desenvolvimento e manutenção das aplicações Web. Estas podem ser especificadas com elevado nível de abstracção e em alguns casos a aplicação final pode ser automaticamente gerada segundo essas especificações abstractas. Porém as metodologias e ferramentas actuais estão incompletas ou são inadequadas para

responder as novas exigências das aplicações Web, como por exemplo, a integração efectiva de áudio, vídeo e altos níveis de interacção” [Preciado05].

3.3.4 Adaptação de aplicações Web 1.0 para Web 2.0

Tendo como ponto de partida o legado deixado pelas metodologias da Web 1.0, tem-se realizado estudos no sentido adaptar as aplicações existentes às necessidades da Web 2.0, definindo novos modelos que auxiliem essa transição.

O “*Rich User eXperience Model*” (RUX-Model)¹² é um modelo visual que possibilita o desenho interfaces com o utilizador para RIA e todos os conceitos associados ao seu desenvolvimento utilizando representações gráficas. O RUX-Model descreve um método que possibilita a adaptação das aplicações Web 1.0 às interfaces Web 2.0, abstraindo toda a UI e preservando a camada de negócio da aplicação original. Este modelo foi pensado para ser aplicado às aplicações existentes baseadas em páginas HTML, que foram previamente geradas através de um modelo Web, como por exemplo WebML. Na Figura 18 no lado esquerdo está representada a arquitectura do RUX-Model e no lado direito encontra-se uma possível aplicação do método. As interfaces abstractas representam de uma forma comum, a UI para todos os dispositivos e plataformas sem preocupações de posicionamento de estilo e comportamento.

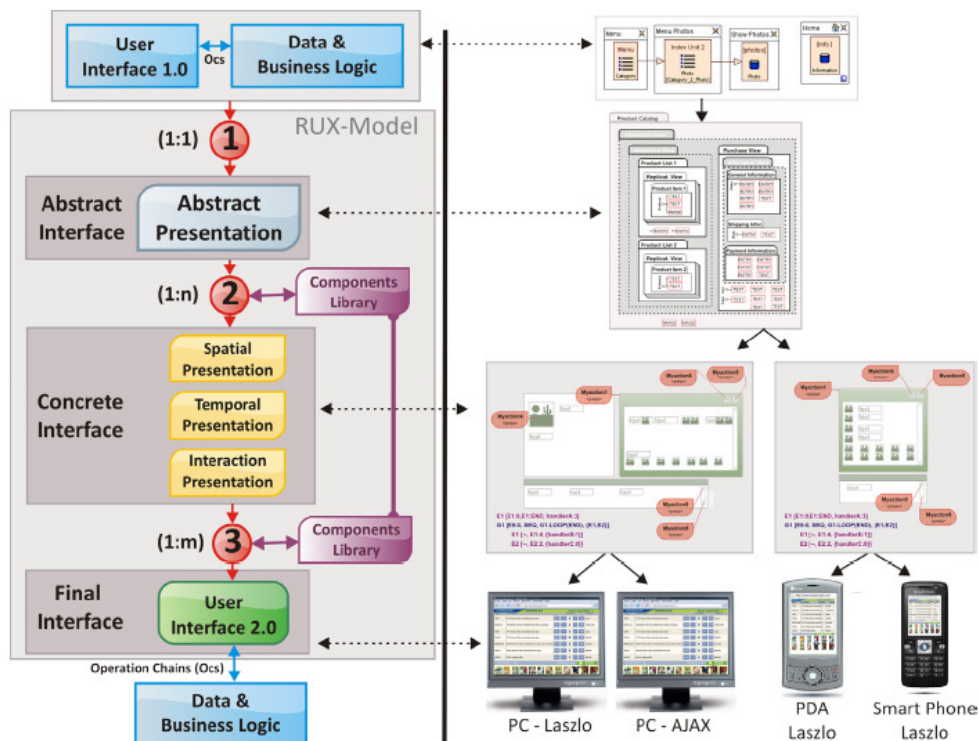


Figura 18 - RUX-Model: arquitectura e método [Linaje07]

¹² <http://www.ruxproject.org>

A *Abstract Interface*, representada na Figura 18, é constituída pelos seguintes elementos: *Connectors*, *Media* e *Views*. Os *Connectors* estabelecem a ponte entre a camada de dados, elementos multimédia da interface abstracta e a camada de negócio. O elemento *Media* representa a informação atómica que é independente da forma como vai ser apresentada no cliente. As *Views* simbolizam a informação que o utilizador final vai visualizar de forma simultânea. O RUX-Model agrupa a informação em quatro tipos de categorias distintas (simples, alternativa, replicação e hierárquica).

Na *Concrete Interface* a interface é otimizada em específico para um dispositivo ou grupos de dispositivos. A *Concrete Interface* está dividida em três níveis: apresentação espacial, apresentação temporal e apresentação da interação. A apresentação espacial representa a gestão do estilo e da distribuição dos vários elementos na interface. A apresentação temporal consiste na especificação dos comportamentos que requerem sincronização temporal, por exemplo animações. A apresentação da interação representa a modelação da interação do utilizador com a interface.

A *Final Interface* corresponde a fase final do processo. Nesta etapa é gerado o código do modelo criado para um determinado dispositivo ou grupo de dispositivos. O código é específico de acordo com a tecnologia envolvida na criação da RIA, como por exemplo AJAX, Laszlo ou XAML¹³.

Todo o processo de especificação e implementação da RIA utilizando do método RUX-Model pode ser auxiliado pela ferramenta CASE designada RUX-Tool. Esta ferramenta facilita a representação de gráficos e diagramas propostos pelo RUX-Model. “Conceptualmente, RUX-Model pode ser utilizado em vários modelos de desenvolvimento Web. A utilização de modelos Web é importante porque RUX necessita de compreender o legado da aplicação Web transformar. Ao nível da implementação, a RUX-Tool tem uma série de pré-requisitos tendo em conta que pode ser utilizada de forma a extrair toda a informação contida nesses modelos. Actualmente a RUX-Tool funciona em conjunto com WebML” [Preciado08].

3.3.5 Soluções

Existem várias tecnologias e Frameworks com a finalidade de introduzir dinamismo e interação nas aplicações Web. Para serem consideradas RIA terão de ter por base os seguintes princípios: capacidade de realizar processamento no lado do cliente, comunicação reduzida e otimizada com o servidor, os dados podem ser armazenados tanto no lado do cliente como no lado do servidor com diferentes níveis de persistência,

¹³ “Extensible Application Markup Language” - <http://msdn.microsoft.com/en-us/library/ms752059.aspx>

operações CRUD (Create, Read, Update and Delete) no servidor e cliente e capacidade de trabalhar tanto em modo *on-line* como em modo *off-line*.

As RIA podem ser divididas em quatro categorias tendo em consideração as tecnologias presentes no desenvolvimento e as funcionalidades por elas disponibilizadas.

No primeiro grupo estão reunidas as *script-based*, a lógica no cliente é desenvolvida utilizando linguagens de *scripting* e de anotação, recorrendo normalmente a AJAX (Asynchronous Javascript And XML). A interface é baseada em HTML e CSS e recorre-se ao Javascript para realizar chamadas assíncronas ao servidor. Para realizar a actualização parcial do conteúdo da página recorre-se ao DOM Scripting. Esta abordagem tem como vantagens não necessitar de elevados recursos e a simplicidade de instalação pois não requer uma pré-instalação no cliente. No entanto, algumas das tecnologias utilizadas, nomeadamente o Javascript, levantam problemas de compatibilidade com os vários browsers existentes. Para minimizar este problema existem várias Framework como por exemplos a *Dojo*¹⁴ e a *Prototype*¹⁵.

Numa segunda categoria agrupam-se as RIA *plugin-based*. As características de maior relevância estão na capacidade de representação gráfica e gestão de eventos proporcionadas pelo plug-in a correr num browser. No desenvolvimento deste tipo de RIA é inicialmente especificada qual a plataforma e qual o ambiente onde vai correr a aplicação. O conceito presente nas RIA *plugin-based* designado por *sandbox*, permite executar de forma seguro o código “não confiável” embebido em páginas Web. “Esta abordagem garante aos programadores uma plataforma de desenvolvimento única e assegura aos fornecedores que a aplicação vai produzir os mesmos resultados em múltiplas plataformas desde que o plug-in seja disponibilizado correctamente” [Farrell07]. A grande vantagem das RIA *plugin-based* reside na simplicidade de desenvolvimento. Porém terão de ser tomadas algumas considerações no que diz respeito a segurança. Como referido anteriormente o isolamento proporcionado pela *sandbox* a correr no browser impede que o sistema sofra qualquer dano. No entanto, esta protecção poderá ser corrompida sempre que a RIA necessite de utilizar recursos externos, como por exemplo, utilizar ficheiros ou pastas no lado do cliente, armazenar informação como as credenciais do utilizador e informação encriptada. Uma desvantagem está na necessidade de o cliente ter de instalar software para poder utilizar o plug-in que pode não ser compatível com todos os browsers e sistemas operativos. O Silverlight¹⁶, Flash¹⁷, Flex¹⁸ e OpenLaszlo¹⁹ são exemplos de RIA *plugin-based*.

¹⁴ <http://www.dojotoolkit.org/>

¹⁵ <http://www.prototypejs.org/>

¹⁶ <http://silverlight.net/>

¹⁷ <http://www.adobe.com/products/flash/>

¹⁸ <http://www.adobe.com/products/flex/>

¹⁹ <http://www.openlaszlo.org/>

As RIA *browser-based* representam outra categoria de RIA. Caracterizam-se por utilizarem uma linguagem para desenvolver GUI específica baseada em XML. Alguns browsers são capazes, de forma nativa, interpretar interfaces implementadas segundo uma linguagem declarativa. A XUL (XML User Interface Language)²⁰ criada pelo projecto Mozilla²¹ permite desenvolver aplicações ricas para múltiplas plataformas utilizando muitas normas XML e tecnologias *standart* como CSS, Javascript ou RDF, diminuindo a curva de aprendizagem para programadores com experiência na programação Web.

Não sendo unanimemente aceite por todos os autores, existem ainda uma quarta categoria de RIA, as *Web-based desktop applications*. Partilham a mesma arquitectura e princípios básicos das RIA. As aplicações são descarregadas da Web e correm sem o recurso a um browser. As aplicações Java Web Start²² são um exemplo típico, descarregando o conteúdo para o cliente utilizando um browser. De seguida, “o utilizador tem de aceitar um certificado de segurança e a aplicação comporta-se como uma aplicação *standalone*” [JAVA09].

3.3.5.1 Tecnologias RIA *plugin-based*

Para o desenvolvimento do protótipo foram estudadas com maior detalhe as RIA *plugin-based*. Ao longo dos anos, o Flash tem sido a tecnologia de escolha para a implementação de aplicações RIA mas, recentemente o Silverlight produzido pela Microsoft²³, tem disponibilizado novas funcionalidades para o seu desenvolvimento de RIA, despertando o interesse por este tipo de software na comunidade de programadores e clientes que utilizam produtos .NET.

O Flash, actualmente tecnologia proprietária da Adobe Systems²⁴, é um software utilizado no desenvolvimento de RIA, “anunciado pela Macromedia em 2001. É uma ferramenta multimédia que combina animação Web com capacidades de programação baseadas em métodos gráficos vectoriais.” [Mi-Jeong03]. Esta tecnologia pode ser utilizada nas mais variadas formas no mundo da Web, marcando uma viragem na forma de apresentar a informação ao utilizador final. Com o Flash é possível representar informação de forma interactiva no formato vectorial e também possibilita a integração de elementos multimédia nas páginas Web, como por exemplo vídeo e som. O Flash fornece ainda um modelo complementar para serviços Web facilitando a comunicação entre as interfaces dos utilizadores e a informação existente na Internet. A possibilidade de utilizar uma arquitectura baseada em serviços²⁵ facilita o desenvolvimento de

²⁰ <https://developer.mozilla.org/En/XUL>

²¹ <http://www.mozilla.org/>

²² <http://java.sun.com/javase/technologies/desktop/javawebstart/index.jsp>

²³ <http://www.microsoft.com>

²⁴ <http://www.adobe.com/>

²⁵ SOA- Service-Oriented Architecture

interfaces para aplicações já existentes permitindo a comunicação entre a interface e camada de negócio.

O Silverlight é uma Framework desenvolvida pela Microsoft que possibilita a criação de aplicações RIA. Segundo a Microsoft ” o Silverlight é um plug-in *cross-browser*, *cross-platform*, e *cross-device* que fornece suporte multimédia, novas funcionalidades de interacção e implementação de aplicações Web ricas baseadas na nova geração .NET. Utilizando Silverlight podem-se alcançar novas áreas de mercado utilizando a Web e dispositivos móveis” [MSOFT09]. “O principal objectivo do Silverlight é trazer a mesma fidelidade e qualidade das aplicações *Desktop* às aplicações Web, permitindo aos programadores e designers construir aplicações de acordo com as especificações dos clientes. O Silverlight foi desenhado para criar a ponte entre os designers e programadores dando a ambos um formato comum para trabalharem” [Moroney08]. Na Tabela 4 estão representados alguns tópicos de comparação entre o Flash e Silverlight.

Crítérios	Flash	Silverlight 2.0
Plataformas	Windows/ Linux / Solaris /Mac OS	Windows/MacOS/Linux ²⁶
Browser Suportados	Internet Explorer/Mozilla FireFox/Safari/Opera	Internet Explorer/Mozilla FireFox/Safari,
Dispositivos móveis	Suporta	Suporta
Linguagens de Programação suportadas	ActionScript	.NET(C#/ Visual Basic), Ajax, Iron Python, Iron Ruby
Serviço de <i>Streaming</i> grátis	Não	Sim
Multicast	Não suporta	Não Suporta
Suporte Alta Definição (HD)	Sim	Sim
Formatos de imagens suportados	Suporta praticamente todos os formatos de imagens	PNG, GIF e JPEG

²⁶ <http://www.mono-project.com/Moonlight> (Projecto open-source que visa dar suporte ao Silverlight no Linux)

Suporte a programação utilizando Socket	XMLSocket, possibilita a comunicação entre o <i>Flash Player</i> e o servidor	Suporta Sockets utilizando System.Net.Sockets ou System.Threading. o Silverlight suporta comunicação assíncrona
Gestão dos dados	Possibilidade de utilizar <i>Web Services</i> e REST ²⁷	Possibilidade de utilizar a Framework integrada <i>Windows Communication Foundation</i> (WCF) auxiliando o desenvolvimento de uma arquitectura SOA. Para manipular a informação no lado do cliente, os programadores podem usufruir do componente “ <i>Language Integrated Query</i> ” (LINQ) ²⁸
Indexação nos motores de pesquisa	Sim	Sim ²⁹

Tabela 4- Comparação entre Flash e Silverlight 2.0

Ambas as tecnologias possibilitam a representação de elementos multimédia, animações e desenho vectorial nas páginas Web. “ A Adobe tem no pensamento a comunidade dos designers, enquanto a Microsoft tem no pensamento a comunidade associada ao desenvolvimento. São ambas necessárias para poder construir RIA” [Ozer08].

Flex, produto desenvolvido pela Adobe, visa proporcionar aos programadores uma Framework para a construção de RIA *cross-plataform*. Baseado na tecnologia Flash é caracterizado por seguir uma vertente orientada para o desenvolvimento. Fornecendo aos programadores um meio de produção de aplicações ricas herdando as capacidades multimédia, desenho vectorial e interactividade do Flash.

²⁷ http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

²⁸ <http://msdn.microsoft.com/en-us/netframework/aa904594.aspx>

²⁹ <http://silverlight.net/learn/whitepapers/seo.aspx>

3.3.6 Limitações

Algumas críticas são, no entanto, apontadas as RIA (*plugin based*), nomeadamente a dificuldade de indexação do seu conteúdo por parte dos motores de pesquisa [Wenz08]. No entanto, têm sido desenvolvidos esforços tanto pelos principais motores de busca como pelas principais empresas a desenvolver tecnologias que permitem desenvolver aplicações RIA.

Outras das limitações apontadas às RIA é o fraco suporte e ausência de *standards* para a construção de interfaces para pessoas com deficiência. No entanto, o WC3 (World Wide Web Consortium) tem reunido esforços para desenvolver *standards* que permitam ultrapassar esta lacuna. O padrão chama-se WAI-ARIA (Accessible Rich Internet Applications) [W3CARIA09] e encontra-se actualmente na versão 1.0. “Com estas restrições de desenho na mente, ARIA define perfis, estados e propriedades que auxiliam os Web designers a descrever as funcionalidades do *widget* nas suas aplicações Web e ajudar em questões tecnológicas de forma a fornecer informação de interacção ao utilizador” [Velasco08].

3.4 Sumário

O presente capítulo introduz os SIG baseados em tecnologias Web, apresentando o conceito de *Web GIS* e apresentando as arquitecturas comuns no desenvolvimento deste tipo de sistemas.

De seguida é descrito o fenómeno crescente da Web 2.0, apresentando o conceito de MashUp.

Por fim, é apresentado um estudo pormenorizado sobre as RIA onde são apresentados os conceitos principais inerentes ao desenvolvimento deste tipo de aplicações e as metodologias que permitem transformar aplicações Web 1.0 em Web 2.0. Ainda nesta secção são comparadas as principais RIA *plugin-based*, terminado o capítulo com a apresentação algumas das limitações presentes no desenvolvimento de aplicações do tipo RIA.

4. Aplicação Web para a gestão e classificação de parcelas agrícolas

A utilização de meios informáticos na agricultura veio abrir novos horizontes nos sistemas de gestão da informação agrícola. O processo de atribuição de níveis de excelência e propriedades únicas na produção vitícola é cada vez mais exigente. Para obter a D.O.C (Denominação de Origem Controlada) é imprescindível um processo de cadastro e classificação de parcelas com vinha eficiente, de forma a garantir a qualidade do produto final.

As parcelas com vinha são porções de terreno, e como tal, possuem natureza geoespacial. Tirando partido da informação georreferenciada, é possível aplicar conceitos e tecnologias presentes nos SIG, proporcionando novas funcionalidades ao sistema, neste domínio.

O capítulo começa por apresentar a metodologia proposta para a realização da passagem de aplicações Web 1.0 para Web 2.0. Seguidamente, é realizada a especificação do sistema já existente, baseada na arquitectura e no modelo comportamental da aplicação (secção [4.2](#)). Na seguinte secção é apresentada a aplicação da metodologia ao sistema existente, utilizando como pano de fundo o desenho, a interactividade e a usabilidade proporcionada pelas RIA (secção [4.3](#)). Por fim, é descrito o processo de visualização das parcelas com vinha (secção [4.4](#)).

4.1 Metodologia proposta

Actualmente existe um vazio no que toca a metodologias que permitam a transformação de aplicações Web 1.0 para aplicações Web 2.0 (ver secções [3.3.3](#) e [3.3.4](#)). Um dos objectivos propostos para esta dissertação consiste em realizar a passagem da aplicação existente para uma aplicação do tipo RIA. É importante preservar a lógica de negócio, pois esta realiza um conjunto alargado de cálculos e validações de parâmetros já devidamente testados. O modelo descrito na secção [3.3.4](#) funciona para aplicações que tenham sido previamente desenvolvidas, baseadas em modelos Web, como por exemplo WebML. Como a aplicação existente não foi implementada segundo modelos Web foi necessário desenvolver uma metodologia que

descreva processos e mecanismos que permitam ultrapassar esta restrição e transformar a aplicação actual numa aplicação RIA. A Figura 19 ilustra a solução proposta.

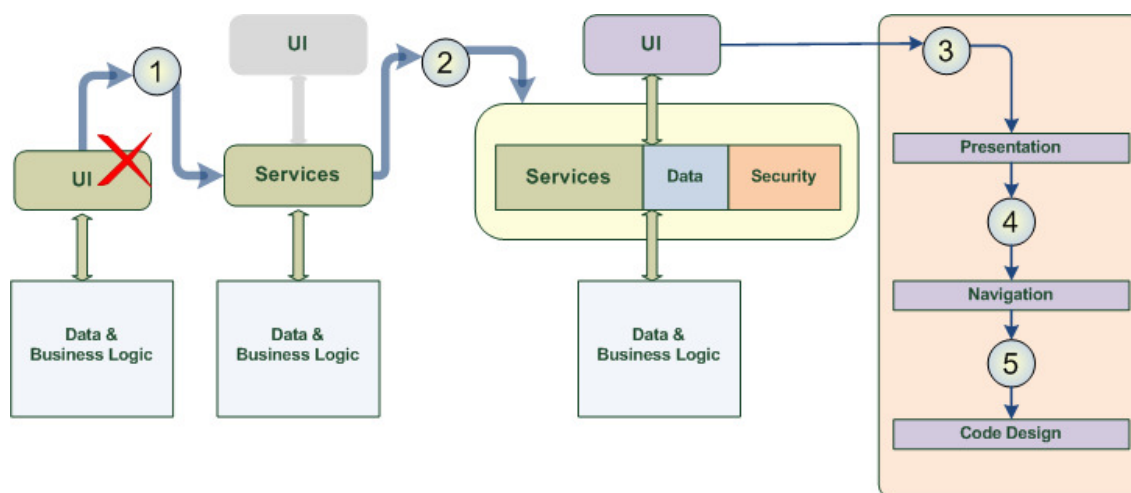


Figura 19 - Descrição da metodologia

A aplicação da metodologia baseia-se em dois conjuntos de operações distintos. Numa primeira fase realizar-se a abstracção das camadas de negócio e de dados, separando-as da camada de *interface* utilizando uma camada intermédia de serviços. Numa segunda fase, projectar-se a GUI de acordo com as especificidades da tecnologia envolvida na implementação da RIA. O processo de desenvolvimento proposto é descrito nos seguintes passos:

- 1) **Abstracção da User Interface:** Caso o código não esteja devidamente estruturado, efectuar a separação do código da camada de apresentação das restantes camadas. Nomeadamente, da camada de negócio e de acesso a dados. Sempre que possível recorrer a padrões de desenho de software de forma a otimizar a reestruturação do código.
- 2) **Camada de Serviços:** A desacoplação é realizada com a criação de uma camada de serviços. A camada de serviços isola a tecnologia que foi utilizada para implementar as camadas de negócio e de dados. Desta forma, é possível utilizar para a mesma camada de negócio vários tipos de RIA, desde que estas suportem a utilização de Web Services. Ainda nesta camada, caso se justifique criar estruturas de dados que auxiliem o tratamento de dados entre a UI e a camada de negócio. Como se trata de uma camada de serviços pode levantar problemas no que diz respeito à integridade da informação e aos processos de autenticação, daí a necessidade de implementar um módulo específico que auxilie este processo.

- 3) **Apresentação espacial:** Definir a distribuição dos vários elementos pela página. Para tal, utilizar métodos que permitam realizar prototipagem de interfaces.
- 4) **Sistema de navegação:** As RIA utilizam o paradigma *single-page application*. As aplicações Web tradicionais são constituídas por um conjunto de páginas onde é possível navegar entre elas. No caso das RIA o sistema de navegação e interação é restringido a uma só página. A implementação de mecanismos que possibilitem as funcionalidades anteriormente referidas depende da tecnologia envolvida na construção da RIA.
- 5) **Desenho do código:** Este passo depende da tecnologia de escolha na implementação da RIA. Realizar uma divisão de código quer física quer lógica de forma a facilitar a manutenção e escalabilidade do código. Podem ser utilizadas diversas técnicas, como por exemplo: a divisão utilizando *namespaces* e a implementação baseada em padrões de desenho.

4.2 Análise do sistema existente

Este tópico tem como objectivo descrever a arquitectura e o comportamento do sistema inicial que vai servir de suporte para aplicação da metodologia utilizada no desenvolvimento do protótipo. Ao longo deste tópico pretende-se dar uma visão geral de alto nível, sobre os componentes da arquitectura física, assim como a decomposição em camadas da arquitectura lógica. Na descrição comportamental são apresentados os casos de uso acompanhados dos diagramas de sequência. Na descrição estrutural são ilustrados os vários componentes presentes na aplicação e a forma como interação entre eles.

4.2.1 Visão geral

O sistema é apenas constituído por um único módulo designado por “Módulo de Classificação de Parcelas”. Neste módulo são realizadas todas as operações de processamento do enquadramento legal, preenchimento dos boletins de campo e gestão dos direitos de plantação. Na Figura 20 está representado o digrama UML de caso de usos do único módulo da aplicação.

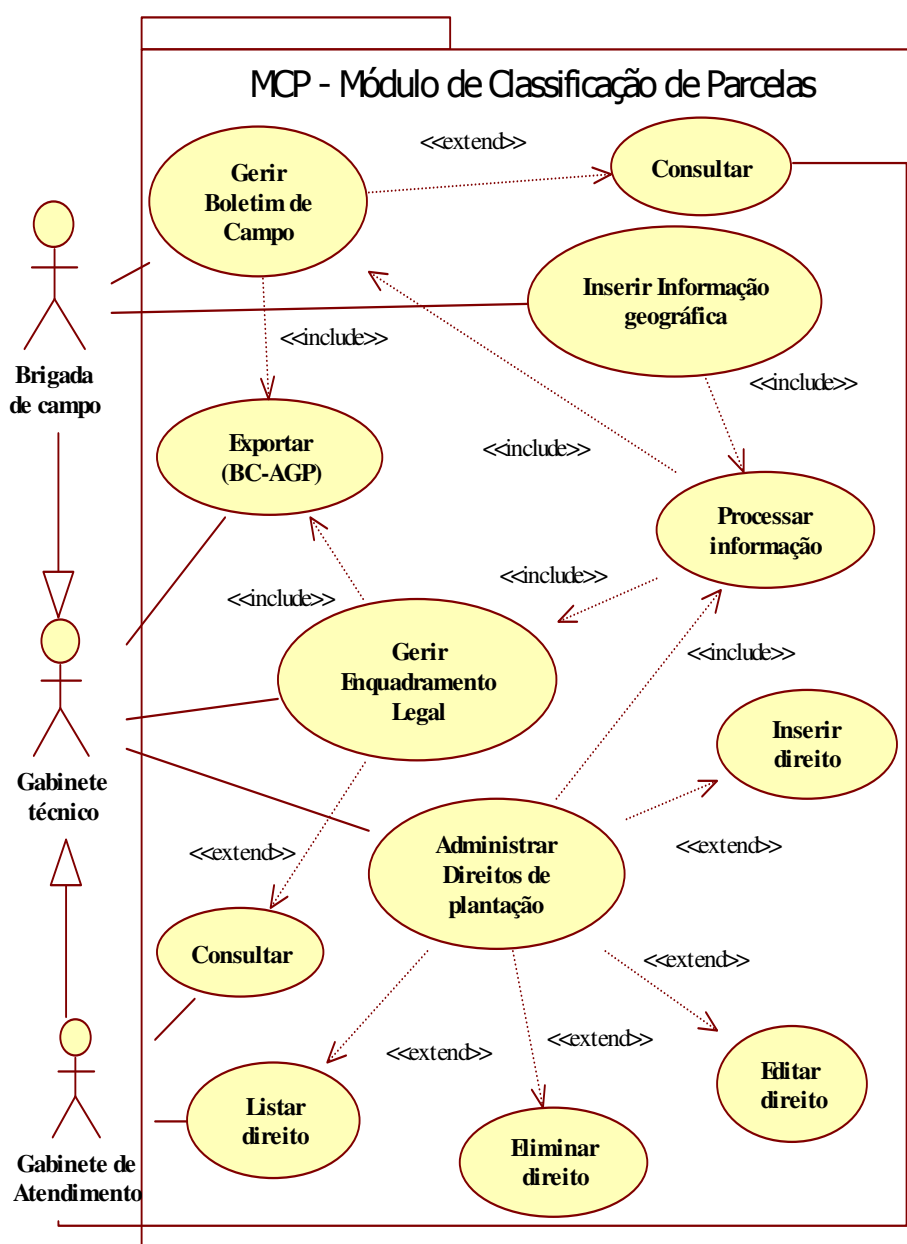


Figura 20 - Diagrama de casos de usos do Módulo de Classificação de Parcelas

Existem três grupos de actores que interagem directamente com a aplicação, o Gabinete Técnico, Brigada de Campo e o Gabinete de atendimento.

A **Brigada de Campo** descola-se ao terreno para efectuar a recolha dos dados. Tem ainda privilégio para processar o boletim de campo previamente preenchidos com informação recolhidos no terreno. Opcionalmente poderá realizar a exportação dos dados (BC-AGP) e consultar os boletins processados.

O **Gabinete de atendimento** representa o grupo de utilizadores que interage directamente com o público. O Gabinete de Atendimento tem permissões para consultar boletins de campo, listar direitos de plantação e consultar enquadramento legal.

O **Gabinete técnico** representa os utilizadores responsáveis pelo processamento do enquadramento legal. O processamento incluiu necessariamente dados com origem geográfica, como por exemplo a área da parcela, o tipo de castas, o tipo de terreno, o declive, a pedregosidade, a localização, a exposição e a aramação. Este grupo tem privilégios para processar os direitos de plantação, pode efectuar as operações de inserir, editar, eliminar e listar direitos. Têm ainda permissões para realizar a exportação da informação dos boletins de campo (BC-AGP) para ficheiros estruturados, com vista a serem utilizados por outros sistemas e aplicações.

O “Módulo de Classificação de Parcelas” é constituído por três casos de uso principais: Gerir Boletim de Campo, Gerir Enquadramento Legal e Administrar Direitos de Plantação.

A **Gestão Boletim de campo** representa a gestão do boletim de campo de acordo com os dados recolhidos no terreno pelas brigadas de campo.

A **Gestão do Enquadramento legal** é o processo de validar o enquadramento legal de um boletim de campo validando a informação geográfica e os direitos de plantação. Este processo resulta na exportação da informação para um ficheiro devidamente formatado.

O caso de uso **Administrar Direitos de plantação** representa a gestão dos direitos de plantação. Neste caso de uso é possível preencher novo direito, editar informação de um direito existente, eliminar e listar direitos de plantação mediante determinados parâmetros de consulta.

4.2.2 Arquitectura física

A topologia da aplicação segue o paradigma cliente-servidor presente nas aplicações Web. O servidor onde está alojada a aplicação pode também ser conter o sistema de gestão da base de dados. Caso se justifique, para uma melhor distribuição de carga, pode-se utilizar dois servidores como está ilustrado na Figura 21.

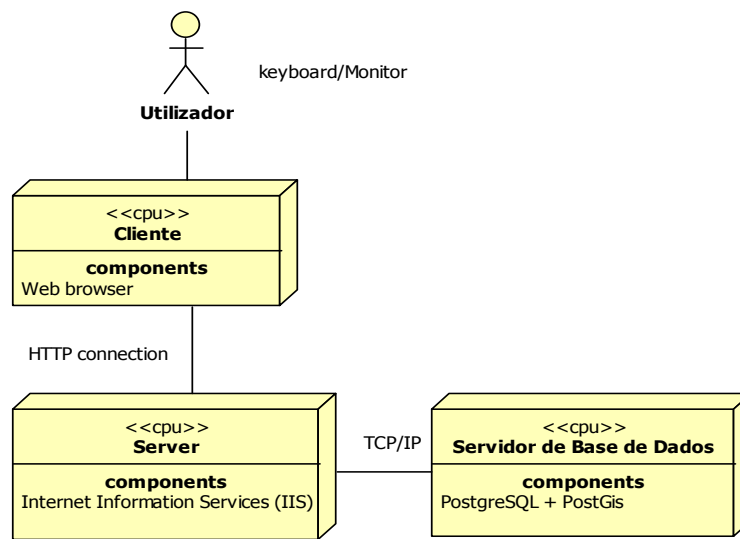


Figura 21- Diagrama de distribuição

A aplicação é constituída pelos três componentes representados na Figura 22. Realizando uma análise *top-down* o primeiro componente corresponde à GUI (Graphical User Interface), o utilizador interage com o sistema utilizando um Web browser. O componente intermédio contém todas as regras de negócio e estabelece a comunicação entre as camadas adjacentes de forma transparente para ambas. Toda a lógica de negócio está alojada num servidor onde os acessos são controlados pelo IIS (Internet Information Services)³⁰. Por fim, o componente que representa o servidor de base de dados. O sistema de gestão de base de dados é o PostgreSQL³¹ com extensão espacial PostGIS³².

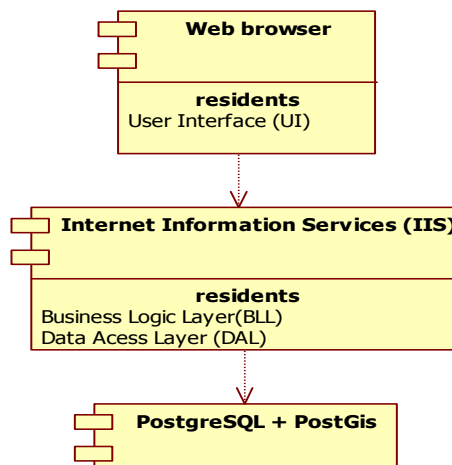


Figura 22 - Diagrama de componentes

³⁰ <http://www.iis.net/>

³¹ <http://www.postgresql.org/>

³² <http://postgis.refrations.net/>

4.2.2.1 Vista de processos

Na vista de processos pretende-se ilustrar a forma como os vários componentes interagem entre si, ver figura 23. A aplicação pode ser utilizada por vários utilizadores em simultâneo, o IIS é o responsável por gerir os múltiplos pedidos.

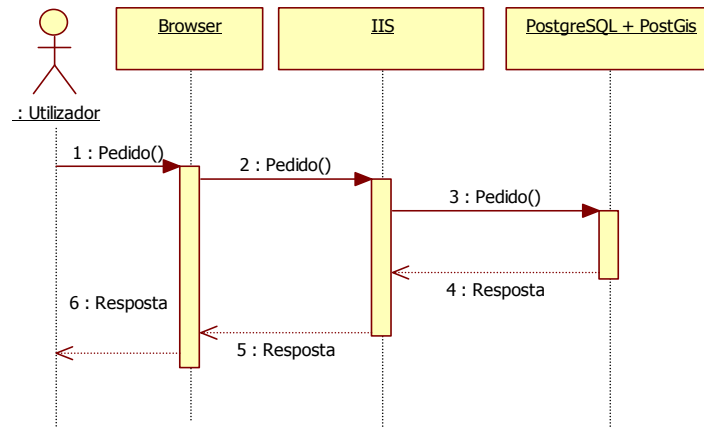


Figura 23 - Vista a nível de processos

4.2.3 Arquitectura lógica

Esta secção tem como finalidade descrever a arquitectura lógica da aplicação. Na Figura 24 estão representados diagramas de pacotes lógicos e as relações de composição e dependência entre pacotes. Cada pacote agrupa um conjunto de classes que partilham funcionalidades e responsabilidades comuns. A aplicação é constituída por três camadas: camada de interface, camada de negócio e camada de acesso a dados.

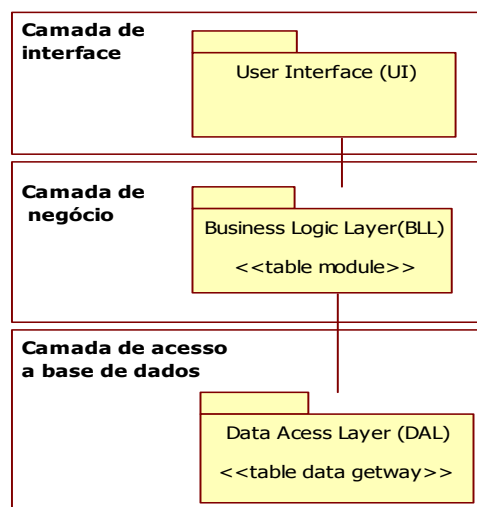


Figura 24 - Diagrama de Componentes

Na camada de interface (UI) o utilizador interage directamente com a aplicação. Quando um método é invocado, esta camada comunica com camada de negócio, passando apenas os parâmetros que foram recolhidos nos formulários da interface. Após processamento da informação o utilizador visualiza os resultados através da camada de interface. A lógica da camada de interface foi implementada recorrendo a C#³³, AJAX e CSS.

A camada de negócio (*Business Logic Layer*) tem como finalidade regularizar todos os processos de negócio. Na camada de negócio reside o núcleo da aplicação em termos de processamento. Recebe os dados da camada de interface realiza todas as validações e cálculos automáticos e enviando posteriormente, a informação para a camada de acesso a dados. A camada de negócio também é responsável por filtrar a informação proveniente da camada de acesso a dados e transferi-la para a camada de interface. A lógica desta camada foi igualmente desenvolvida utilizando C#.

A camada de acesso a dados (*Data Access Layer*) é responsável por gerir as consultas à base de dados e comunicar as respostas à lógica de negócio. O código fonte desta camada foi implementada com o recurso a C#.

A Figura 25 representa o fluxo de informação entre as várias camadas da aplicação ao realizar o processamento do enquadramento legal.

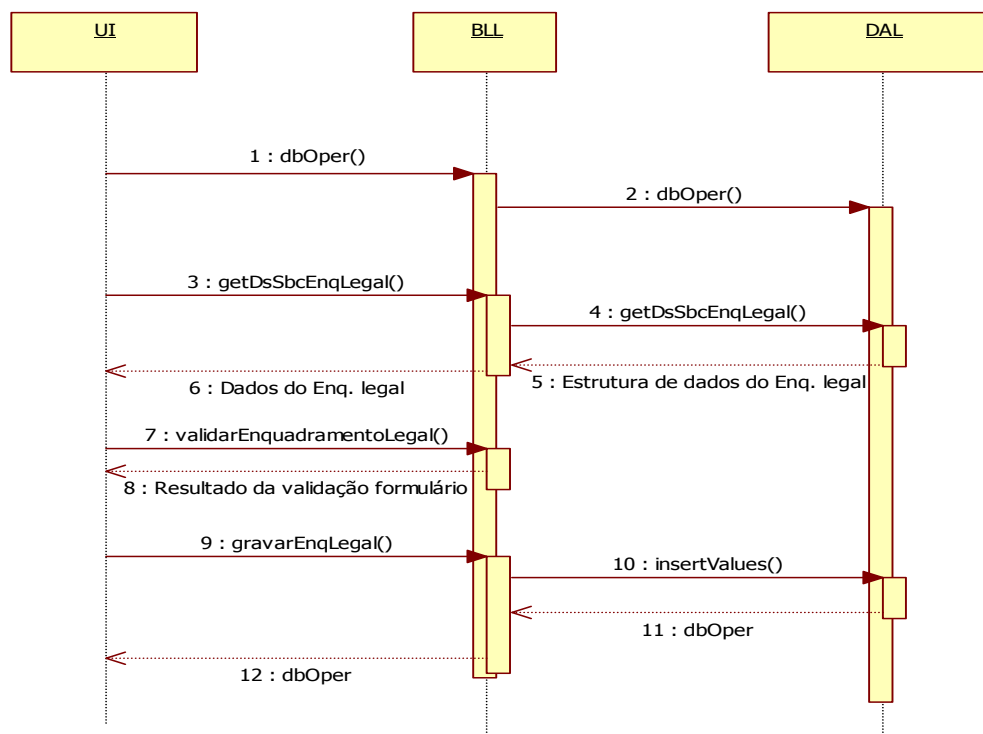


Figura 25 - Diagrama de sequência do caso de uso processar enquadramento legal

³³ <http://msdn.microsoft.com/en-us/vcsharp/default.aspx>

4.2.3.1 Detalhes de implementação

O padrão de desenho utilizado na construção da *Business Logic Layer* e da *Data Access layer* designa-se por “*Table Module + Table Data Gateway (DataSets)*”. Esta arquitectura baseia-se em dois padrões *Table Module* [MFOWLER09a] e *Table Data Gateway* [MFOWLER09b]. Na Figura 26 está representado o diagrama de classes com a divisão lógica do código. Esta figura ilustra o caso particular do enquadramento legal, mas o padrão repete-se para as restantes classes que implementam os outros casos de uso da aplicação.

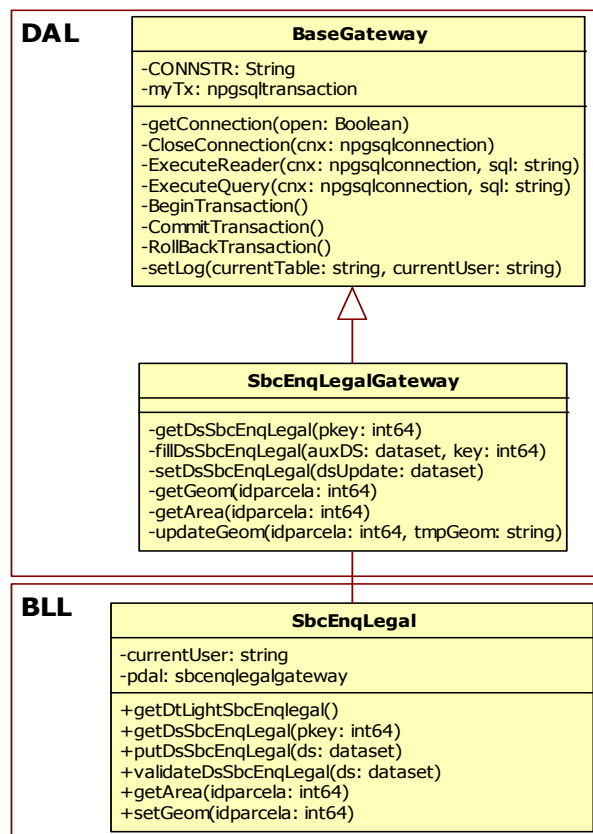


Figura 26 - Table Module + Table Data Gateway utilizando DataSets

Table Data Gateway: A classe *BaseGateway* é uma classe genérica que tem como função a gestão dos acessos à base de dados, verificação da ligação, a manutenção de logs e controlar as transacções. A classe *SbcEnqLegalGateway* agrupa todas as operações de acesso a base de dados relativas ao enquadramento legal, definindo os parâmetros e as declarações SQL (Structured Query Language). Desta forma, é possível isolar o código do SQL do restante, facilitando a implementação e a manutenção do mesmo. Os resultados das consultas são armazenados em estruturas do tipo *DataSet*³⁴.

³⁴ [http://msdn.microsoft.com/en-us/library/system.data.dataset\(VS.71\).aspx](http://msdn.microsoft.com/en-us/library/system.data.dataset(VS.71).aspx)

Table Module: Cada classe da lógica de domínio corresponde a uma tabela. A classe SbcEnqlegal representa toda a lógica de domínio relativa ao enquadramento legal. Na classe SbcEnqlegal o tipo de estruturas é o mesmo da classe SbcEnqLegalGateway.

Na camada de interface o código é dividido por um conjunto de páginas Web que partilham trechos de código comum através de *user controls*³⁵. Isto é visível na Figura 27. Cada página possui uma funcionalidade distinta.

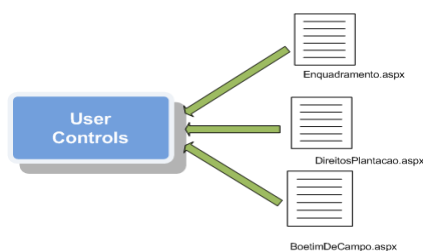


Figura 27 - Distribuição do código

DataSet: Genericamente um DataSet é uma colecção de dados estruturada em forma tabular, na qual cada coluna representa uma variável em e cada linha corresponde a um determinado valor dessa mesma variável. O tipo de dados DataSet surgiu na versão 1.0 da .NET Framework. Um DataSet é constituído por uma ou mais tabelas, designadas por DataTable. Um DataSet possui ainda restrições e relações entre as tabelas, ver Figura 28. A estrutura de dados DataSet possibilita a representação e utilização da base de dados em “modo desligado”. Ou seja, o DataSet faz persistir a informação utilizando a memória volátil, armazenando as tabelas e as relações, possibilitando a replicação da estrutura e do conteúdo da base de dados. Este cenário distribuído é reforçado com a utilização da arquitectura ADO .NET³⁶. Os DataSet podem armazenar, simultaneamente, informação proveniente de fontes distintas. A título de exemplo, o mesmo DataSet pode ser constituído por dados obtidos a partir de um ficheiro XML e de uma base de dados.

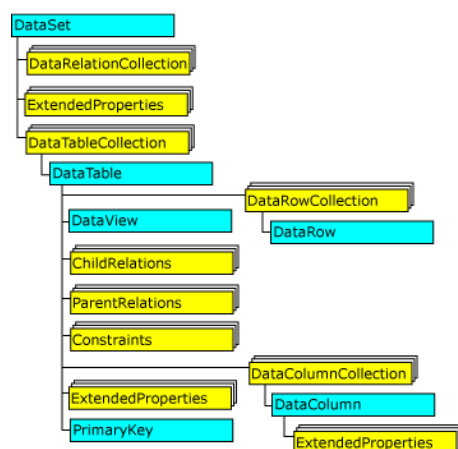


Figura 28 - Object Model do DataSet [MSOFT09f]

³⁵ [http://msdn.microsoft.com/en-us/library/y6wb1a0e\(VS.71\).aspx](http://msdn.microsoft.com/en-us/library/y6wb1a0e(VS.71).aspx)

³⁶ <http://msdn.microsoft.com/en-us/library/e80y5yhx.aspx>

4.3 Aplicação da metodologia

Na presente secção é exemplificado o processo de adaptação da aplicação existente numa aplicação do tipo RIA, através da metodologia descrita na secção 4.1. Inicialmente descreve-se a tecnologia, as estruturas e os tipos de dados utilizadas no desenvolvimento da camada intermédia, que possibilita o isolamento das camadas de negócio e de dados. Seguidamente, são apresentadas as duas soluções encontradas para o desenho e implementação da GUI.

4.3.1 Camada intermédia de serviços

De forma a garantir a independência da *User Interface* das restantes camadas foi implementada uma camada baseada em serviços, utilizando WCF (Windows Communication Foundation)³⁷, ver Figura 29.

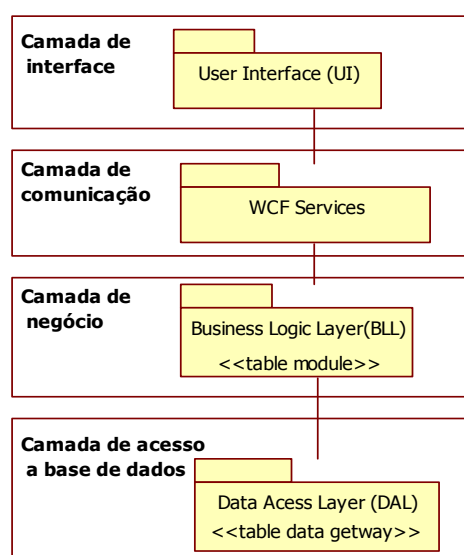


Figura 29 - Camada de serviços

O WCF é uma tecnologia presente na Microsoft .NET Framework desde a versão 3.0, que possibilita a construção de aplicações orientadas a serviços (*Service-Oriented Architecture*). Com o objectivo de otimizar a transferência de informação, o WCF disponibiliza as seguintes funcionalidades: comunicação binária otimizada entre aplicações Windows, comunicação síncrona e assíncrona, comunicação com o recurso a transacções, mecanismos de segurança que garantem a integridade da informação e MSMQ (Microsoft Message Queuing) que permite gerir filas de mensagens através da atribuição de prioridades. A interoperabilidade é garantida pela comunicação baseada no protocolo SOAP (Simple Object Access Protocol). Este foi o protocolo escolhido na

³⁷ <http://msdn.microsoft.com/en-us/netframework/aa663324.aspx>

implementação dos *Web Services* pois permite a troca de informação, de forma estruturada, baseando-se na associação de XML com uma linguagem descritiva WSDL (Web Services Description Language). O SOAP é um dos protocolos *padrão* adoptados na indústria na construção de serviços Web. Desta forma, isola-se completamente a camada de negócio da tecnologia utilizada na implementação da RIA. No entanto, a RIA tem de suportar a comunicação utilizando *Web Services* baseados em SOAP.

4.3.1.1 Padrões de desenho de software

Durante a fase de implementação foram utilizadas práticas de programação que facilitam a leitura, a manutenção e a reutilização do código. Foram seguidas as convenções de nomes do C#, assim como sempre que possível foram aplicados padrões de desenho de software. Neste caso particular utilizou-se o padrão *Singleton* para gerir as instâncias dos objectos criados na camada de serviços.

Singleton, padrão utilizado para restringir a instanciação de uma classe a um objecto, ver Figura 30. Cada serviço tem um Singleton associado, evitando que cada vez que um método de um determinado serviço é invocado, uma nova instância da classe seja criada. O padrão foi implementado segundo as recomendações “*Patterns & Practices*” da Microsoft. Para mais detalhes de implementação ver anexo A.4.



Figura 30 - Singleton

4.3.2 Estruturas e tipos de dados

A CLR (Common Language Runtime)³⁸, componente central da tecnologia .NET, é uma máquina virtual que disponibiliza serviços e cujo objectivo principal passa por auxiliar no processo de desenvolvimento de aplicações e garantir a interoperabilidade e compatibilidade entre as várias linguagens suportadas pelo .NET. O Silverlight é um subconjunto do WPF (Windows Presentation Foundation), garantindo a maioria das funcionalidades de desenho e suportando a maior parte dos tipos de dados. Consequentemente, o Silverlight possui uma “mini-CRL” de forma a não tornar pesado, as aplicações Web a correr nos browsers. A CLR do Silverlight é baseada na versão ASP.NET 3.5, suportando um conjunto alargado de tipos e estruturas de dados através da BCL (*Base-Class Library*)³⁹. Suporta ainda a compilação e execução de linguagens

³⁸ [http://msdn.microsoft.com/en-us/library/8bs2ecf4\(VS.71\).aspx](http://msdn.microsoft.com/en-us/library/8bs2ecf4(VS.71).aspx)

³⁹ <http://dmsdn.microsoft.com/en-us/netframework/aa569603.aspx>

dinâmicas, como por exemplo IronRuby⁴⁰ e IronPython⁴¹, importando as bibliotecas DLR (Dynamic Language Runtime)⁴².

No entanto, a CLR do Silverlight não suporta a referência de *assemblies* compiladas com a .NET Framework padrão. Logo, não é possível referenciar directamente, um projecto Silverlight, a DLL (Dynamic-link library) resultante da compilação da aplicação inicial, compilada na versão ASP .NET 2.0. A camada de negócio da aplicação existente foi implementada utilizando a estrutura de dados DataSet (ver secção 4.2.3.1), no entanto, a “mini-CLR” não suporta esse tipo de estrutura de dados, originando incompatibilidade entre as estruturas de dados da camada de negócio e da interface utilizando Silverlight. Dadas estas incompatibilidades, a camada intermédia de serviços além de permitir a independência total da camada de negócio da tecnologia utilizada na construção da interface, vai servir de “embrulho” (*wrapper*), gerindo as transformações das estruturas de dados entre as duas camadas.

Numa primeira abordagem implementaram-se classes auxiliares que tratam da transformação de DataSet em *Hash Tables* to tipo Dictionary<TKey, TValue>⁴³ (ver Figura 31) A chave corresponde ao nome da coluna do DataTable e o valor correspondente é uma lista de um tipo de dados genérico, ou seja, não “tipado”. No lado da UI, ao receber a estrutura de dados genérica tem que se realizar, programaticamente, todas as conversões de dados necessárias para realizar o *binding* dos controlos. O mesmo processo repete-se quando é necessário enviar dados da camada de interface para a camada de negócio.

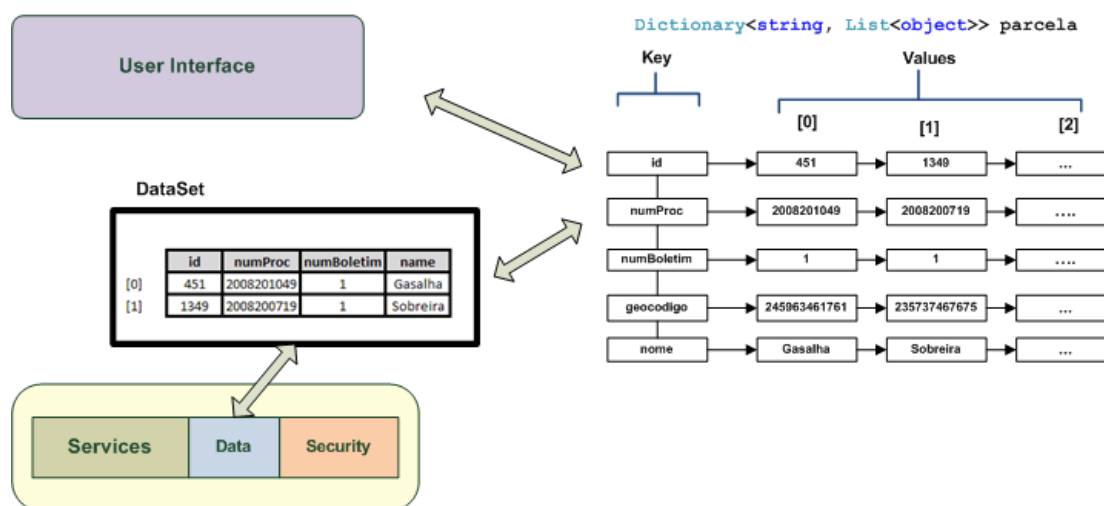


Figura 31 - Mapeamento DataSet em Hash Table

⁴⁰ <http://www.ironruby.net/>

⁴¹ <http://www.codeplex.com/Wiki/View.aspx?ProjectName=IronPython>

⁴² <http://www.codeplex.com/dlr>

⁴³ <http://msdn.microsoft.com/en-us/library/xfhwa508.aspx>

Este processo revela alguma inflexibilidade no tratamento dos dados. Caso o DataSet tenha vários DataTable é necessário trabalhar com estruturas de dados mais complexas, como por exemplo Dictionary< TKey, Dictionary<TKey, TValue>>, em que a chave associa ao nome do DataTable ao valor corresponde na Hash Table, como exemplifico na Figura 31. Estas estruturas tornam complexa a procura de um valor numa determinada chave de uma Hash Table, pois são necessárias diversas iterações e ciclos para encontrar o valor desejado. Outro problema associado a este tipo de transformação, são as conversões dos vários tipos de dados, que podem conter cada campo da DataTable. Por exemplo ao enviar uma variável da camada de negócio para a camada de apresentação do tipo inteiro é necessário converte-la num tipo genérico e a vinda proceder a transformação inversa, todo este processo ocupa recursos computacionais. Para consultar mais detalhadamente pormenores de implementação, ver anexo A.1.

Surgiu a necessidade de realizar um estudo detalhado sobre o comportamento da .NET Framework na forma de gestão e suporte no que respeita a tipo dados. Na camada intermédia de serviços, definiram-se classes auxiliares (*Custom Classes*) só com atributos de forma a imitar a esqueleto de um DataSet e DataTable, ver Figura 32. Estas *Custom Classes* possibilitam a desmaterialização do DataSet numa estrutura baseada em Strings⁴⁴ estruturadas de forma facilitar a conversão para XML. Este tipo de dados criados pelas *Custom Classes* é suportado tanto pela camada de serviços como pela camada de UI. No anexo A.2 podem ser consultadas as estruturas das classes auxiliares.

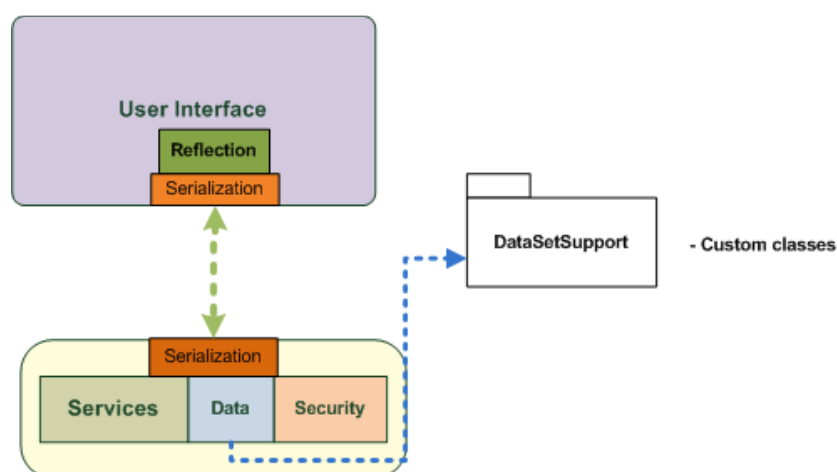


Figura 32 - DataSetSupport

Na camada de interface ao ser recebida a estrutura de dados fracamente “tipada” segue-se um processo de transformação complexo. Em tempo de execução os dados são convertidos no tipo de dados original, utilizando *Reflection*. Esta funcionalidade da

⁴⁴ [http://msdn.microsoft.com/en-us/library/362314fe\(VS.71\).aspx](http://msdn.microsoft.com/en-us/library/362314fe(VS.71).aspx)

.NET Framework permite “criar dinamicamente instâncias de um determinado tipo de dados, atribuir um tipo de dados a um objecto ou saber qual o tipo de dados de um objecto e invocar os seus métodos, acedendo aos seus campos e propriedades. *Reflection* fornece objectos (de um determinado tipo *Type*) que encapsula em *assemblies*, módulos e tipos.” [MSOFT09c]. “A *Reflection* API do C# .NET Framework possibilita que um programa se inspecione e se manipule a ele próprio. Pode ser utilizada para procurar todos os tipos de dados de um *assembly* e/ou invocar, dinamicamente, métodos de um *assembly*. Pode também imitar uma linguagem intermédia “*on-the-fly*” para que o código gerado possa ser executado directamente” [Suresh09]. Esta abordagem facilita a manutenção do código, pois não está limitada a um DataSet em específico, bastando invocar os métodos responsáveis pela transformação do DataSet num tipo de dados dinâmico, passando como parâmetro o DataSet ou DataTable. O processo repete-se quando é necessário converter uma estrutura de tipo de dados dinâmicos num DataSet. A conversão de dados é realizada de forma dinâmica, contrariamente à primeira abordagem em que cada campo da Tabela tem de ser convertido programaticamente de forma explícita. Ou seja, caso o tipo de dados de uma coluna do DataTable mude é necessário ir ao código refazer o *casting*. Detalhes de implementação podem ser consultados no anexo A.3.

Os DataSet são estruturas de dados complexas e verbosas. Transferir um DataSet ou uma estrutura equivalente através de serviços Web, dependendo da quantidade associada a essa mesma estrutura, pode ocupar demasiados recursos computacionais (memória, saturação da rede e tempo de processamento). “Ao transferir uma classe DataSet utilizando um Web Service .Net, internamente a Microsoft .NET Framework utiliza a serialização XML (*System.Xml.Serialization.XmlSerializer*)” [MSOFT09d]. Num estudo realizado por [MSOFT09e] esta solução provou não ser a mais adequada para DataSet com elevado número de linhas (ver Figura 33). A solução que oferece melhores resultados é a serialização binária. O formato binário é a forma mais simples e eficiente de transferir informação *by-wire*. Apesar de a estrutura das *Custom Classes* não ser exactamente um DataSet partilha muitas das suas propriedades. Com vista a otimizar a transferência de informação entre a camada intermédia de serviços e a camada de apresentação, implementaram-se classes auxiliares que tratam do processo serialização e “deserialização” binária dos objectos transferidos pelos serviços, como representado na Figura 32.

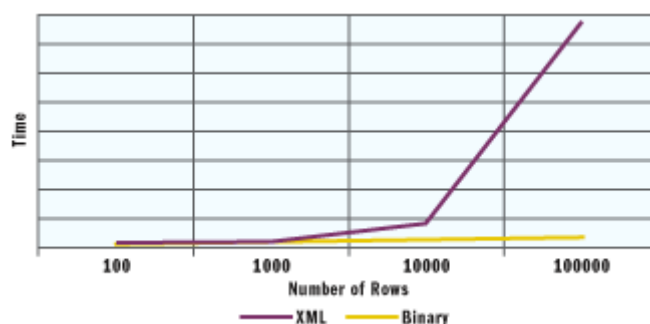


Figura 33 - Remoting End-to-End Time [MSOFT09e]

4.3.3 Desenvolvimento da GUI baseado num sistema hierárquico

A tecnologia seleccionada para no desenvolvimento da RIA foi o Silverlight (ver secção 3.3.5.1). A escolha deveu-se à facilidade de integração proporcionada pelas restantes tecnologias Microsoft, como por exemplo LINQ, e pela facilidade na manipulação de estruturas de dados era também do interesse do INESC Porto explorar as capacidades desta tecnologia no desenvolvimento Web com vista a sua utilização em aplicações futuras.

Nesta fase do desenvolvimento, os esforços foram direccionados para a construção da GUI. Realizando-se uma prototipagem simples, desenhando o esqueleto das principais áreas de apresentação de conteúdos (ver Figura 34). No topo situa-se o “Header”, contendo o sistema de navegação, cujo imita o comportamento do menu em cascata. No fundo situa-se o “Footer”, onde está implementado o sistema de *feedback*, que notifica o utilizador sobre o estado actual do sistema. Na zona central da aplicação, denominada por “Main”, está situado o contentor principal da aplicação. Neste local é descarregado o conteúdo correspondente ao item do menu seleccionado.

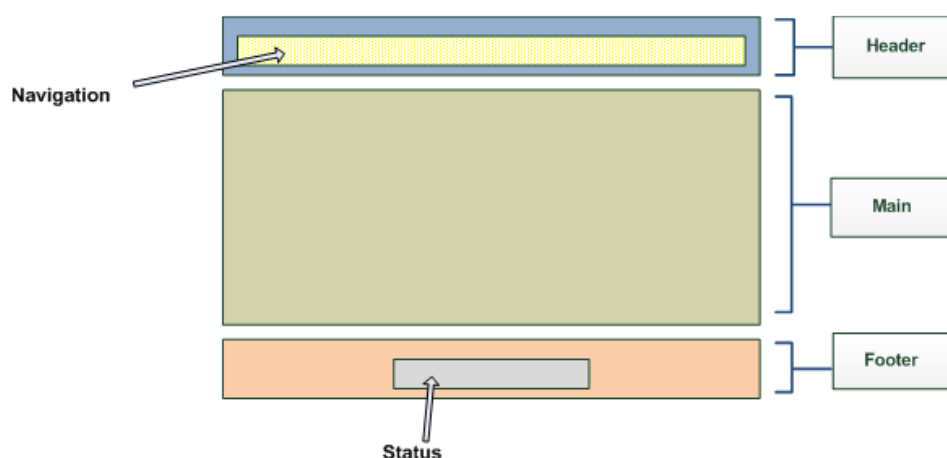


Figura 34 - Esqueleto da página do protótipo

Nesta abordagem a divisão lógica do código deve-se á utilização de namespaces. Cada *namespace* representa um grupo distinto de funcionalidades. Na Figura 35 foram definidos dois *namespaces* principais: o código pertencente à User Interface (SL.UI) e o código correspondente as classes auxiliares (SL.Utills.Data).

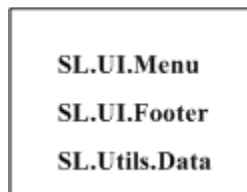


Figura 35- Definição de namespaces

A Figura 36 representa a hierarquia de pastas correspondem aos *namespaces* definidos na Figura 35.

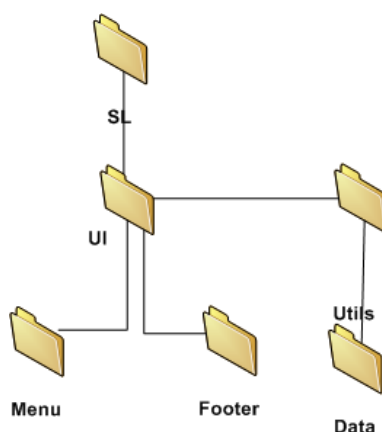


Figura 36- Sistema de pastas

Os resultados obtidos na construção do protótipo segundo a esta abordagem de desenvolvimento podem ser consultados na secção [5.2.1](#).

4.3.4 Desenvolvimento da GUI utilizando *Composite Application Library*

A CAL (Composite Application Library), também denominado por Prism 2.0, é um conjunto de bibliotecas, padrões e boas práticas para desenvolver aplicações compostas (*Composite Applications*) utilizando WPF e Silverlight. Estes tipos de aplicações caracterizam-se por serem flexíveis, desacopladas, autónomas e que, ao serem integradas, funcionam como uma aplicação única. “Durante a fase de desenvolvimento os requisitos da aplicação podem ser modificados, devido ao aparecimento de novas oportunidades de negócio e desafios, novas tecnologias ou por *feedback* por parte do utilizador. Tendo em conta estes imprevistos é importante construir aplicações flexíveis para que possam ser “estendidas” ao longo do tempo” [MSOFT09g]. Projectar e desenhar tal nível de flexibilidade pode ser uma tarefa

complicada de levar a cabo, requerendo uma arquitectura que possibilite desenvolver e testar partes individuais da aplicação de forma independente. A aplicação final deverá fornecer uma experiência de interacção rica e flexível agrupando a informação de múltiplas fontes proveniente de cada módulo, ver Figura 37.

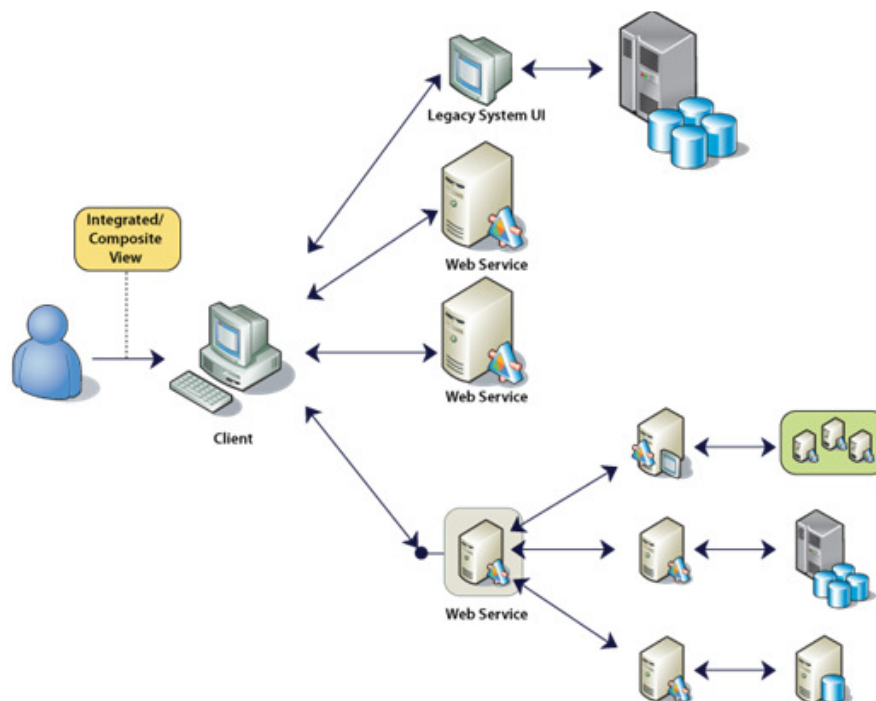


Figura 37 - Composite Application integrando múltiplos componentes [MSOFT09g].

A biblioteca CAL assenta em quatro conceitos principais: *Modularity*, *UI Composition*, *Multi-Targeting* e *Deployment*.

Modularity, desenho de um sistema utilizando módulos distintos que poderão ser posteriormente agregados a uma aplicação única. Cada unidade (módulo) é dividida de acordo com as suas funcionalidades e responsabilidades. Apesar de os módulos serem completamente independentes, deverão possuir mecanismos que possibilitem a comunicação entre eles.

UI Composition, corresponde à distribuição dos vários componentes pelo *layout* da aplicação. Cada componente visual, também designado por *view*, é definido em cada módulo. De seguida, as *views* dos múltiplos módulos são distribuídas em regiões do *layout* predefinidas em tempo de execução num único módulo agregador.

Multi-targeting, utilizando as bibliotecas CAL é possível com o mesmo código base, desenvolver para duas plataformas distintas. As tecnologias utilizadas no desenvolvimento são o Silverlight (Web) e o WPF (Desktop). Para alcançar este feito recorre-se a vários padrões de desenho de software a uma infra-estrutura que maximize a partilha de código entre dois ambientes.

Deployment, os binários do Silverlight e do WPF não são compatíveis. Logo é necessário configurar previamente um *deploy* específico para cada plataforma.

Os resultados obtidos na construção do protótipo segundo a esta abordagem de desenvolvimento podem ser consultados na secção [5.2.2](#).

4.3.4.1 Padrões de desenho de software na *Composite Application Library*

A construção de *composite applications* envolve a utilização de diversos padrões. Esta secção tem como objectivo descrever, de forma breve, os padrões comuns no desenvolvimento de aplicações utilizando a CAL. Estes podem ser divididos em três grandes grupos: *Composite User Interface Patterns*, *Modularity Patterns* e *Testability Patterns*.

Composite User Interface Patterns

Composite: composição dos vários objectos em estruturas hierárquicas. Este padrão permite tratar objectos individuais e composição de objectos de forma uniforme. Este padrão permite ainda criar uma hierarquia de classes de objectos simples e objectos compostos complexos, de forma transparente, no programa do cliente.

Composite view: Semelhante ao padrão anteriormente referido, mas dedicado à representação das *views*. Este padrão permite organizar as *views* de cada módulo de forma hierárquica. Este padrão é fundamental na utilização do CAL.

Command: utilização de objectos na representação de acções, encapsulando o seu pedido como um objecto, de forma a possibilitar diversas parametrizações de acordo com esse pedido. Este padrão separa quem realizou o pedido (*invoker*) do comando em si e dos seus *handlers*. A CAL possibilita a combinação de múltiplos comandos (*ICommand* e *DelegateCommand*) criando um *CompositeCommand*.

Adapter: consiste na adaptação da interface de uma classe de forma a corresponder com a interface esperada de uma outra classe.

Modularity Patterns

Separated Interface: introduzido por [MFOWLER02] separa a definição da interface e a implementação em diferentes *packages*. O cliente depende só da interface sem estar preocupado com pormenores de implementação. Na CAL este padrão é salvaguardado pela utilização da interface *IModule*.

Plugin: “este padrão fornece uma configuração centralizada e configuração em tempo de execução” [MFOWLER09c]. Logo é possível que uma implementação concreta de

uma classe seja ajustada em tempo de execução, evitando a recompilação. Na CAL este padrão é conseguido através da utilização das seguintes classes: *DirectoryModuleCatalog*, *ConfigurationModuleCatalog* e *ModuleInitializaer*.

Service Locator: Permite às classes localizarem serviços específicos sem ser necessário saber quem e como são implementados.

Event Aggregator: “eventos provenientes de múltiplos objectos reunidos num único objecto, de forma a simplificar o registo por parte dos clientes” [MFOWLER09d]. Por exemplo, um sistema com múltiplos objectos em que cada um é uma fonte potencial de eventos, o cliente teria de subscrever-los um a um. O *EventAggregator* funciona como um “repositório” único para os todos os eventos dos objectos. Na CAL o padrão é delegado nas classes *EventAggregator* e *CompositePresentationEvent*.

Façade: interface que reúne um conjunto de interfaces de um subsistema. Esta interface única simplifica a facilita a utilização isolando o grupo de interfaces do subsistema.

Dependency Injection: padrão semelhante ao *Service Locator*, utilizado para manter um baixo nível de acoplamento entre diferentes módulos do sistema. Permite que as classes localizem serviços específicos sem ser necessário saber pormenores de implementação dos mesmos. “O conceito do *Dependency Injection* passa por ter um objecto separado, um *assembler*, que preenche um campo de um objecto *listener* com uma implementação apropriada para a interface *finder*” [MFOWLER09e].

Testability Patterns

Inversion Control (IoC): conceito genérico que procura “estender” uma Framework. A sequência da chamada dos métodos é invertida comparada com a “programação tradicional”, o controle é delegado a outro componente, ficando este responsável pela sua execução.

Segundo [MFOWLER09g], IoC é em parte o que distingue uma Framework de uma biblioteca. Uma biblioteca é essencialmente um conjunto funções que podem ser chamadas, normalmente organizadas em classes. Cada chamada realiza uma determinada tarefa e retorna o controlo ao cliente. Uma Framework encarna alguma abstracção de design associado a uma ou vários comportamentos predefinidos. Para poder ser utilizado é necessário inserir o comportamento em vários pontos da Framework. A Framework toma a responsabilidade do controlo e não o cliente.

Alguns autores consideram o IoC um principio architectural e não um padrão.

Separated Presentation: “Assegurar que todo o código que manipula a apresentação só manipula a apresentação, colocando todo o código das camadas de domínio e dados em áreas claramente separadas do programa” [MFOWLER09f]. Representa ainda uma

categoria de padrões que serve para separar a lógica da UI da representação visual da mesma. Tem como principal vantagem testar a lógica da apresentação sem ser necessário envolver a representação visual. Existem alguns padrões conhecidos que seguem este conceito, como por exemplo: *Model-View-Controller* (MVC), *Model-View-Presenter* (MVP) e o *Model-View-ViewModel* (M-V-VM).

Model-View-Controller: este padrão permite isolar a lógica de negócio da lógica de apresentação. O *Model* representa os dados. A *View* corresponde ao aspecto gráfico da UI em si. O *Controller* é o coração do padrão gere a interacção entre o *Model* e a *View*.

Model-View-Presenter: derivação do padrão MVC, focalizando-se na total abstracção da apresentação, separando a lógica da apresentação (*Presenter*) da apresentação (*View*). Esta separação permite reutilizar a mesma lógica da apresentação em apresentações distintas: O *Presenter* recebe os dados do *Model* manipulando a informação através das regras de negócio estipuladas. O *Presenter* determina ainda como ela vai ser visualizada na *View*, que por sua vez, a envia comandos ao *Presenter* actuado sobre a informação, sendo posteriormente enviado os resultados ao *Model*.

Model-View-ViewModel: Introduzido por [GOSSMAN09] em 2005, baseado nos padrões *Model-View-Controller* e *Model-View-Presenter* (MVP) foi adaptado às necessidades actuais do desenvolvimento de *User Interfaces*, particularmente às que a linguagem XAML no seu desenvolvimento (Silverlight e WPF). A complexidade das UI obriga a que sejam desenvolvidas por equipas multi-disciplinares com filosofias de desenvolvimento díspares. O designer foca-se no grafismo enquanto o programador concentra-se no desenvolvimento. Tanto como no MVC e o MVP, o M-V-VM separa o *Model* da *View*. O *Model* é responsável pelo tratamento dos dados e pela gestão dos estados da aplicação. Por sua vez, a *View* está encarregue pelo desenho da UI. A diferença reside no terceiro componente, o *ViewModel* que gere a interacção entre o *Model* e a *View* não contém nenhuma referência directa para a *View*, serve-se do sistema de *DataBinding* da .NET Framework para popular a *View*. Desta forma, a *View* é totalmente independente do *ViewModel* não contendo qualquer *code-behind* relativo ao *Model* ou ao *ViewModel*.

4.4 Visualização de Mapas

Um dos requisitos do protótipo consiste na integração de um componente que possibilite a localização e a visualização da informação georreferenciada. Esta funcionalidade permite aos viticultores, ao gabinete de atendimento e ao gabinete técnico visualizar o posicionamento e a forma da área da parcela com vinha, facilitando a detecção de eventuais irregularidades.

Na secção 3.2 foi abordado a evolução que os serviços de *Web mapping* sofreram nos últimos anos. Para o desenvolvimento do protótipo, o estudo incidiu principalmente em dois serviços: Google Maps e Live Search Maps. A hipótese de utilização da API do Google Maps foi deixada para segundo plano devido a ambiguidades na licença de utilização, ver anexo A.5. Esta só permite a utilização directa dos serviços disponibilizados pela API, utilizando Flash ou Javascript. Devido à falta de clareza neste ponto, optou-se por utilizar os serviços fornecidos pelo Live Search Maps.

A Microsoft tem desenvolvido controlos que possibilitem a integração do Silverlight com o Live Search Maps. Na implementação do protótipo foi utilizado o controlo “*Virtual Earth Silverlight CTP*”⁴⁵.

Adicionando a extensão espacial PostGIS ao sistema de gestão de base de dados, é possível exportar para o formato KML (Keyhole Markup Language) o resultado das consultas realizadas às parcelas com vinha. Estes resultados consistem num conjunto de coordenadas que representam um polígono equivalente à área da parcela com vinha.

Uma das limitações da versão actual do Live Search Maps é a impossibilidade de desenhar directamente multi-polígonos de acordo com os resultados provenientes da base de dados, originando por exemplo, figuras geométricas incorrectas. A Figura 42 apresenta um exemplo incorrecto, no lado esquerdo, da forma incorrecta de representação do polígono. Quando o pretendido é o desenho de figuras geométricas sem sobreposições, semelhantes ao lado direito da Figura 38.



Figura 38 - Multi-polígonos

Para resolver este problema realiza-se uma filtragem dos pontos provenientes do KML. Seguidamente, aproveitando a capacidade do Live Search Maps possibilitar a representação de figuras geométricas em camadas distintas, inicia-se o processo de desenho da parcela com vinha. Inicialmente desenhavam-se os pontos previamente filtrados e unem-se utilizando linhas. De seguida, desenha-se o polígono sem representar as bordas. Ao sobrepor as camadas obtêm-se o efeito pretendido. Isto é visível na Figura 39.

⁴⁵ <https://connect.microsoft.com/silverlightmapcontrolctp>

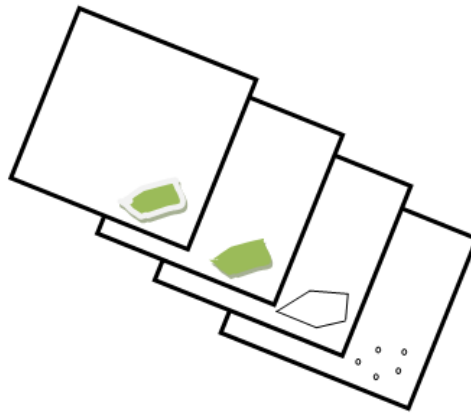


Figura 39 - Mapa multi-camadas

4.5 Sumário

Neste capítulo é apresentada a metodologia proposta que permite a adaptação de aplicações Web 1.0 em aplicações Web 2.0, utilizando as RIA para desenvolver as GUI.

De seguida, é realizado o levantamento da aplicação existente na qual vai ser aplicada a metodologia. Ainda neste capítulo é descrito o processo de aplicação da metodologia, explicando os mecanismos implementados para garantir o suporte a tipos e estruturas de dados, a aplicação da camada intermédia de serviços e o desenho e a separação de código da GUI utilizado.

Por fim, é apresentado o componente de *Web mapping* que permite a localização e visualização da informação georreferenciada.

5. Implementação e análise de resultados

Este capítulo tem como objectivo apresentar o protótipo construído mediante a metodologia apresentada no capítulo anterior. Serão também apresentada a especificação do sistema e alguns detalhes de implementação. Por fim, é elaborada uma análise sobre os testes de carga realizados e aos resultados obtidos.

5.1 Enquadramento

De forma a comprovar a viabilidade da metodologia proposta, construiu-se um protótipo baseado na mesma. Os principais requisitos do protótipo são os seguintes:

- Preservar as camadas de negócio e dados da aplicação já existente.
- Desenvolver uma arquitectura que permita realizar uma separação de código eficiente com vista a promover a reutilização de vários componentes (*User Controls*).
- Integração de um componente de *Web mapping*.
- Divisão modular.
- Interacção com uma base de dados espacial.

5.2 Selecção da arquitectura

Foram implementados dois protótipos seguindo duas arquitecturas distintas. A primeira baseia-se numa divisão simples, utilizando um sistema hierárquico de *namespaces*. A segunda arquitectura utiliza uma divisão mais elaborada baseando-se em padrões de desenho de software.

5.2.1 Protótipo baseado no sistema hierárquico

Neste tópico é apresentado o protótipo desenvolvido segundo o sistema hierárquico descrito na secção [4.3.4.](#)

Na figura 40 é possível observar a GUI resultante do esqueleto definido na fase de prototipagem, apresentando em detalhe, o sistema de *feedback* implementado.

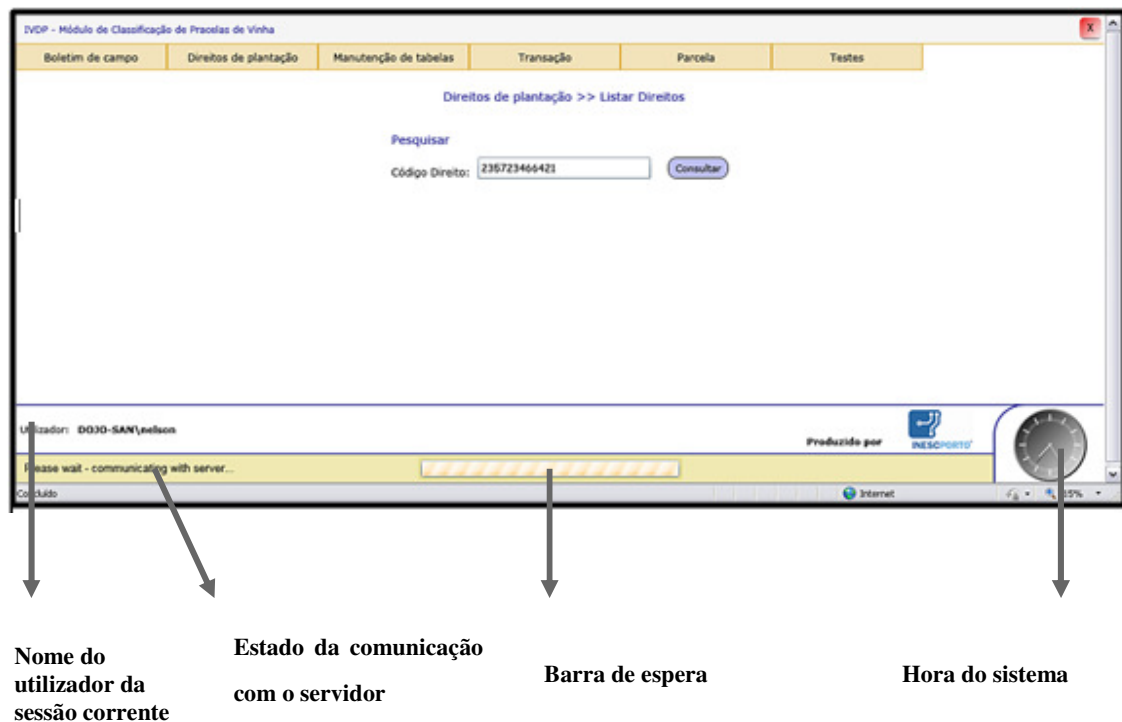


Figura 40-Sistema de feedback

Logicamente o código está dividido segundo *namespaces*, cada um reúne código com responsabilidades distintas. Fisicamente o código está agrupado por várias pastas, na qual, a hierarquia é definida tendo em conta suas funcionalidades. Isto é visível na Figura 41.

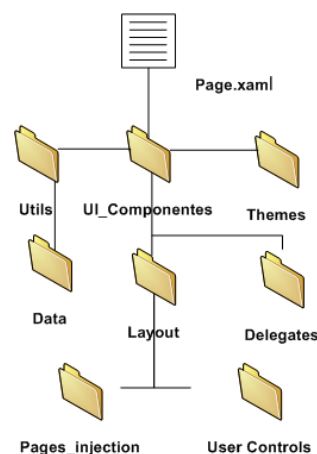


Figura 41 - Divisão lógica e física do código

Page.xaml Página principal da aplicação onde é realizada toda a interação com o utilizador.

UI_Componentes: Reúne o código relacionado com os componentes visuais da GUI.

Pages_injection Uma única página (Page.xaml) importa conteúdos de outros ficheiros XAML situados no *namespace* “Pages_injection”, apresentando-os na secção “Main”. Cada um destes ficheiros representa um caso de uso.

User Controls O *namespace* “User Controls” contem controlos que são partilhados por toda a aplicação.

Delegates Eventos personalizados que disparam de acordo com a interacção do utilizador. Por exemplo, o utilizador ao realizar uma consulta disparar pode um evento que envia uma mensagem para interface a notificar o estado do processo.

Layout Ficheiros correspondentes a cada secção do esqueleto projectado: “Header”, “Main” e ” Footer”;

Themes Personalização dos controlos, como por exemplo definição de cores e tipo de letra.

Utils: reúne todo o código auxiliar ao projecto.

Data Funções que auxiliam o tratamento das estruturas de dados.

De referir o menu em cascata que possibilita a navegação dentro do controlo Silverlight, carregando o item correspondente para o *container* principal. De forma a poder “deslizar” o conteúdo central foi necessário implementar uma funcionalidade extra (*Scroll*) que suportasse este requisito. Isto é visível na Figura 42.

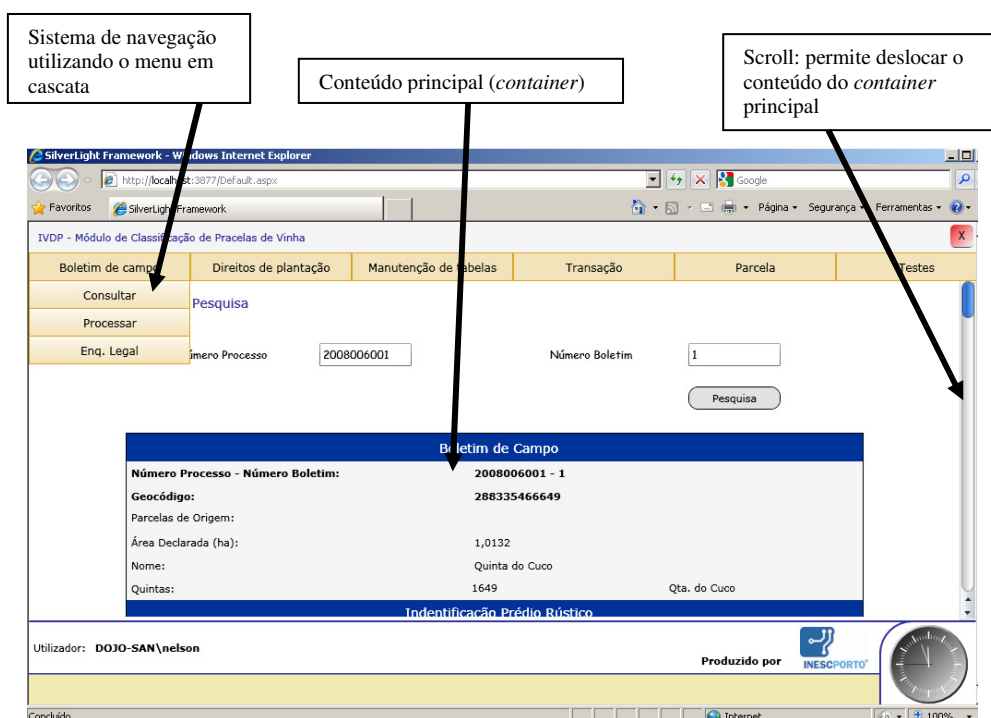


Figura 42 - Consultar Boletim de campo

Na Figura 43 está representado um comportamento típico das janelas modais presentes nas aplicações Desktop utilizando RIA (Silverlight).

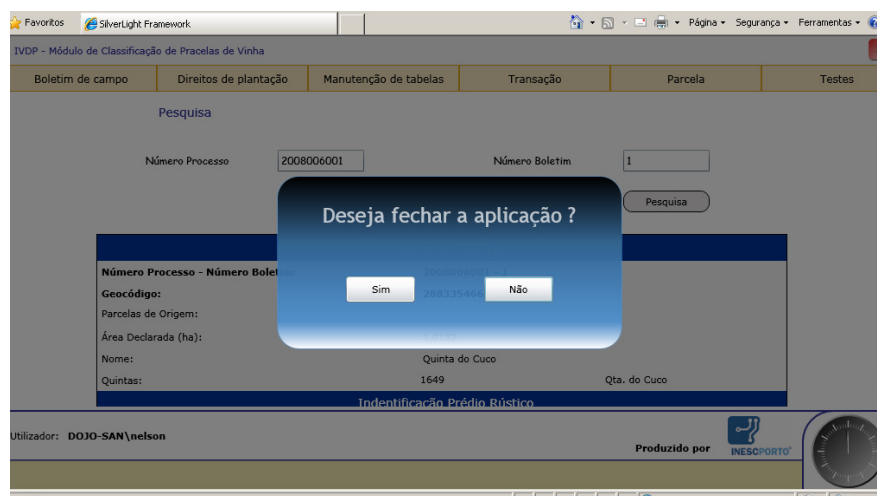


Figura 43 - Janela Modal

Ao longo do desenvolvimento da aplicação o processo de desenho dos controlos demonstrou ser bastante flexível e de fácil aprendizagem, principalmente quando auxiliado pela ferramenta Microsoft Blend⁴⁶. Todo o desenho das interfaces Silverlight foi desenvolvido utilizando XAML (eXtensible Application Markup Language).

5.2.2 Protótipo utilizando Composite Application Library

Na Figura 44 está representada a versão do protótipo desenvolvida utilizando a CAL, ver secção 4.3.4. Com este protótipo foi possível implementar uma aplicação baseada em componentes independentes agrupando-os em módulos.

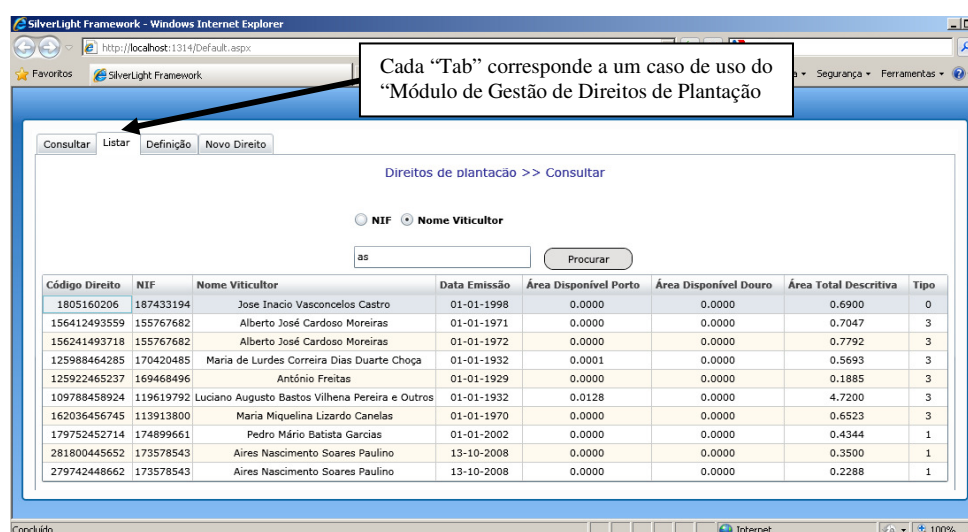


Figura 44 - Protótipo baseado na biblioteca CAL (Silverlight)

⁴⁶ <http://www.microsoft.com/Expression/products/Overview.aspx?key=blend>

Durante a realização do protótipo foi levantada a hipótese de implementar uma versão Desktop utilizando WPF. A utilização da biblioteca CAL e dos padrões de desenho de software a ela associados, tornou possível reutilizar grande parte do código produzido na versão Silverlight, acelerando o processo de desenvolvimento da versão Desktop. A grande vantagem da utilização da CAL está na possibilidade de, com o mesmo código, produzirem aplicações para as plataformas Web e Desktop sem que isso implique um esforço adicional substancial. As semelhanças no resultado obtido podem ser analisados comparando as Figuras 44 e 45.



Figura 45 - Protótipo baseado na biblioteca CAL (WPF)

5.2.3 Síntese comparativa: hierarquia de namespaces *versus* CAL

Na primeira abordagem foi possível provar a aplicação da metodologia proposta. Entretanto, a divisão utilizando *namespaces*, não se traduziu numa solução satisfatória na divisão modular pretendida. A utilização de *user controls* facilita a reutilização de código pelos vários controlos principais da aplicação, no entanto, esta arquitectura não foi desenhada para adicionar bibliotecas externas (*assembly*), limitando o uso e reutilização de componentes. Esta abordagem tem como vantagem a simplicidade ideal para aplicações de baixa complexidade.

Na segunda abordagem, com a utilização de biblioteca CAL foi possível construir uma arquitectura desacoplada, com elevado nível de independência entre módulos. Mantendo a camada intermédia de serviços e de suporte a dados a CAL foi essencialmente utilizada no desenho do código da GUI, na gestão espacial dos vários componentes da aplicação e na gestão dos serviços Web.

A CAL permite o desenvolvimento de uma arquitectura sustentável, ou seja, adicionar ou remover novas funcionalidades sem que para isso seja necessário

redesenhar toda a estrutura da aplicação. A CAL foi desenvolvida a pensar na optimização, efectuando uma gestão eficiente dos recursos na gestão dos vários componentes e módulos da aplicação em tempo de execução. A implementação dos padrões *Separated Presentation* (ver secção [4.4.1.1](#)) facilita a realização de testes à lógica da *User Interface*, assim como a utilização de uma estrutura modular possibilita a realização de testes aos módulos e componentes isoladamente, sem que isso afecte a aplicação principal. A construção de *composite applications* utilizando a CAL possibilita que várias equipas de desenvolvimento produzam simultaneamente módulos distintos minimizando a interdependência entre equipas. Outra das vantagens da CAL está na reutilização do código, possibilitando o desenvolvimento simultâneo de aplicações Web (Silverlight) e Desktop (WPF).

A CAL não é indicada para aplicações de pequena dimensão pois exige um período de aprendizagem, devido ao elevado número de padrões de desenho de software envolvidos. A CAL enquadra-se na construção de aplicações com algum grau de complexidade e dimensão, delegando a responsabilidade das várias funcionalidades da aplicação por vários módulos e componentes, integrando-os numa única aplicação final.

5.3 Especificação do sistema modular

Um dos objectivos propostos na implementação do protótipo consiste na projecção de uma arquitectura padronizada sobre um sistema modular. A CAL auxilia o desenho de uma arquitectura baseada em módulos independentes que interagem entre si. Das duas arquitecturas implementadas é a aquela que demonstra ser mais escalável e reutilizável, ideal para desenvolver aplicações modulares.

Da aplicação da metodologia descrita na [secção 4.3](#) mantém-se a independência entre a UI e a camada de negócio, garantida pela camada intermédia de serviços. O segundo conjunto de operações proposto na metodologia (*Presentation, Navigation e Code Design*), é alterado utilizando CAL. Os padrões presentes no conjunto de bibliotecas da CAL vão ser responsáveis pelo desenho e gestão eficiente do código, pelo sistema de navegação entre módulos e pela distribuição dos vários componentes pelo *layout* da aplicação.

Cada módulo funciona de forma independente agrupando um conjunto de tarefas em comum. De seguida cada módulo é compilado, originando um *assembly* (DLL), que posteriormente é agrupado num único componente (*Shell*). Este tem a função de agregar todos os *assemblies*, gerindo a comunicação e a interligação entre os vários módulos. Para o utilizador é um processo transparente, interagindo com a aplicação como um todo e não de uma forma isolada por módulos. Este funcionamento é visível na Figura 46.

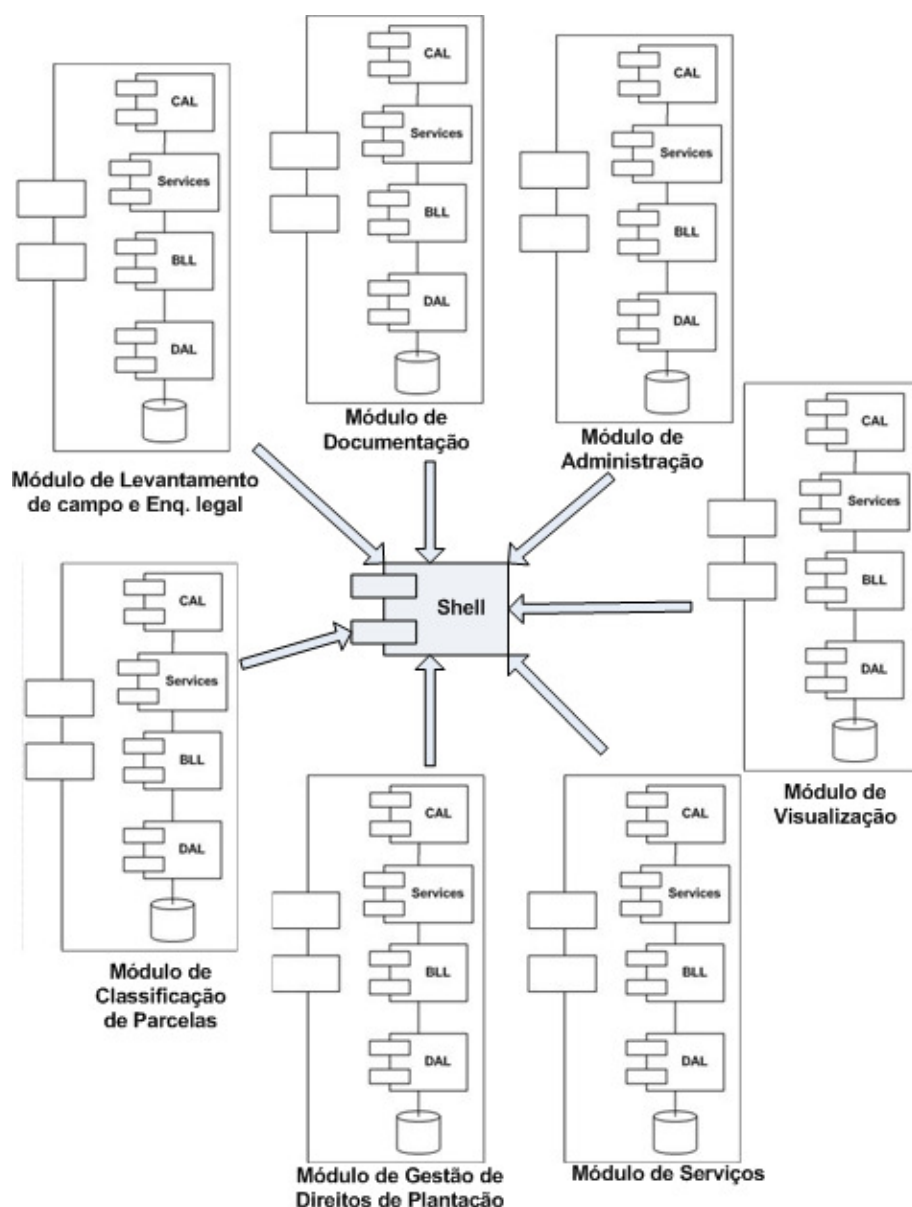


Figura 46 - Sistema Modular sobre uma arquitectura padronizada

A Figura 46 representa a divisão modular proposta, para o sistema a desenvolver. Cada módulo agrupa um conjunto de funcionalidades específicas. Os módulos são os seguintes:

Módulo de Levantamento de campo e Enquadramento Legal: Registo e validação dos dados provenientes das brigadas de campo. Realização do processamento do enquadramento legal.

Módulo de Gestão Direitos de Plantação: Delegação de autorizações na atribuição de *Mosto Generoso* pelos vários viticultores

Módulo de Classificação de Parcelas: Classificação da parcela com vinha, após efectuado o levantamento de campo e associado direitos. Ainda neste módulo é efectuada a gestão do histórico das parcelas.

Módulo de Administração: Operações de *backoffice*.

Módulo de Documentação: Criação e impressão de listagens e circulares previamente definidas. Exportação de dados para ficheiros.

Módulo de Visualização: Visualização da informação geográfica, integrando-a com serviços de *Web mapping*.

Módulo de Serviços: Gestão dos serviços responsáveis pela comunicação com o sistema interno do IVDP.

5.4 Implementação do sistema modular

Da especificação descrita na secção 5.3 implementou-se o “Módulo de gestão de Direitos de Plantação” como prova de conceito da abordagem, utilizando a CAL. Cada caso de uso foi dividido em pequenos componentes (ver Figura 47), onde para cada componente após o processo de compilação, o *assembly* resultante é injectado num componente único (IVDP.Direitos.Base). Este componente representa um módulo estanque que posteriormente vai ser agregado à aplicação principal (IVDP.Shell).

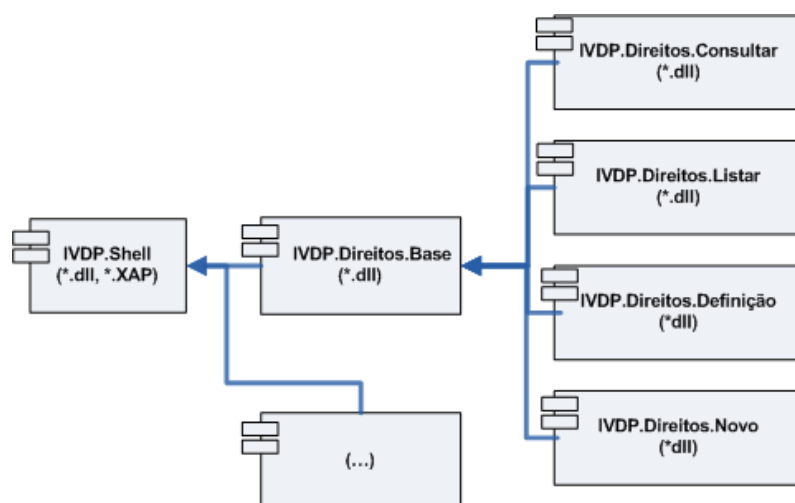


Figura 47 - Dependência entre módulos

IVDP.Shell: Módulo agregador que vai corresponder à aplicação final.

IVDP.Direitos.Base: Reúne todos os casos de uso relativos à Gestão dos Direitos de Plantação.

IVDP.Direitos.Consultar: Pesquisa de Direitos de Plantação tendo como parâmetro de pesquisa o código do direito.

IVDP.Direitos.Listar: Listagem dos Direitos de Plantação mediante pesquisa prévia tendo como parâmetro de pesquisa o nome do viticultor ou o Número de Identificação Fiscal (NIF).

IVDP.Direitos.Definição: Edição de um Direito de Plantação.

IVDP.Direitos.Novo: Criação de um novo direito de plantação

O esqueleto da *user interface* do módulo agregador baseou-se no padrão de desenho de *User Interfaces* “Module Tabs” [UIPATT09]. Cada componente corresponde a um “Tab” ver Figura 48.

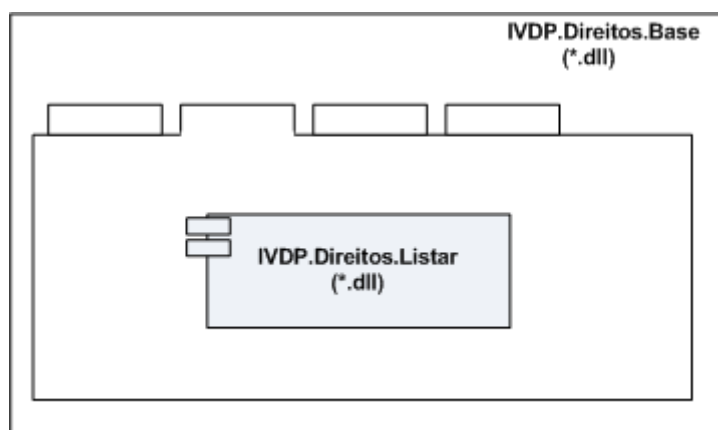


Figura 48 - Agregação de módulos

Na Figura 49 é apresentado o protótipo correspondente ao *layout* definido na Figura 49. Cada “Tab” corresponde a um componente independente. Sendo posteriormente agregado ao módulo final “Módulo de Gestão de Direitos de Plantação”.

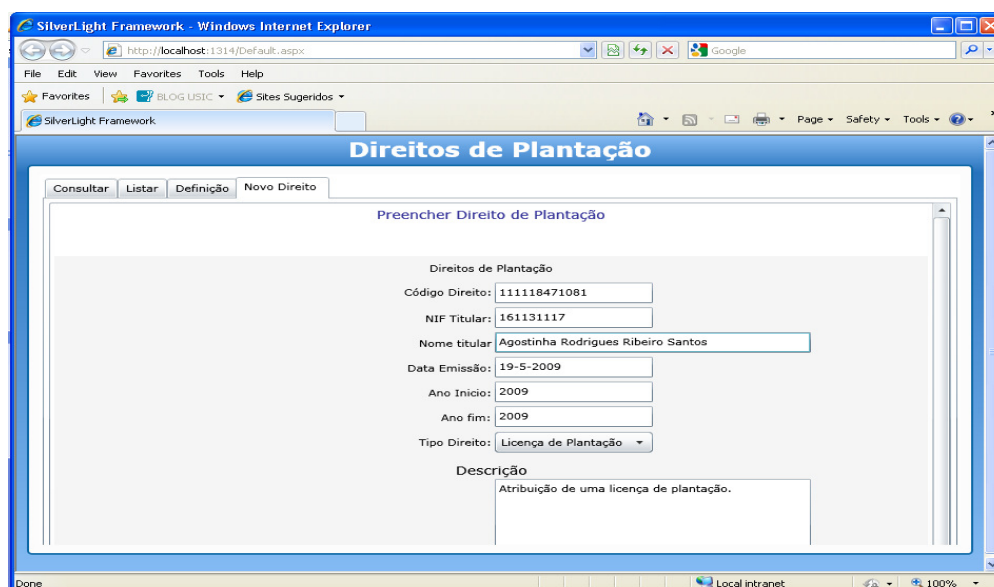


Figura 49 - Protótipo utilizando a CAL (formulário)

O desenvolvimento de aplicações utilizando a CAL segue o seguinte conjunto de operações: inicialização dos módulos, registo dos serviços e definição da GUI. O funcionamento deste processo é visível na Figura 50.

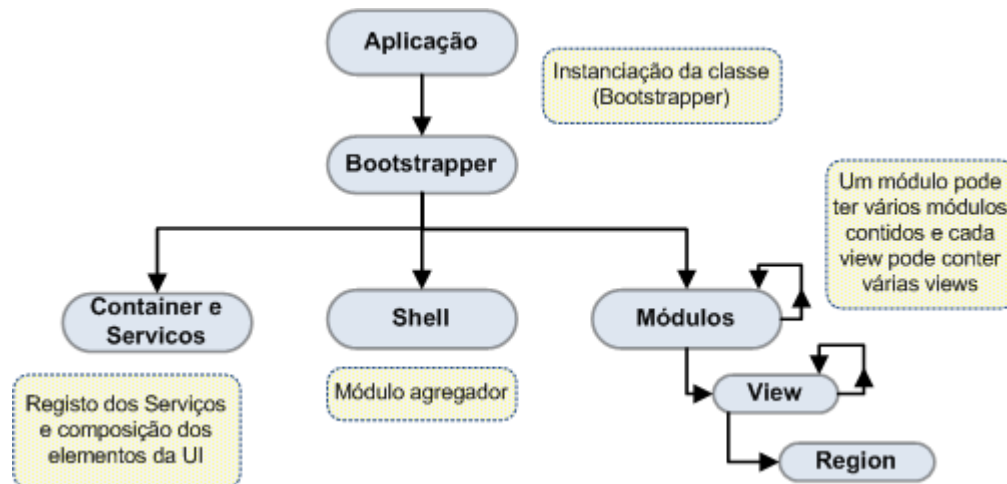


Figura 50 - Aplicações CAL

Inicialização dos módulos: A inicialização da aplicação deve-se à implementação da classe *Bootstrapper*. Esta classe é responsável por configurar os *containers*, configurar as regiões (*Regions*) onde vão ser injectados os restantes componentes, criar a *Shell* e inicializar os restantes módulos da aplicação. O ciclo de vida de um módulo passa pela definição/descoberta do módulo, carregamento do *assembly* correspondente para a memória e inicialização do módulo, instanciando a classe. A classe *Bootstrapper* implementa a classe abstracta *UnityBootstrapper* e as interfaces *IModuleManager*, *IModuleCatalog*, *IModuleInitializer*, *IRegionManager* e *IRegionViewRegistry*.

Registo dos serviços: Localização e registo dos serviços através da implementação da interface *IServiceLocator*. Esta não instancia directamente os serviços apenas, os regista para posteriormente poderem ser utilizados. Este funcionamento é visível na Figura 51.

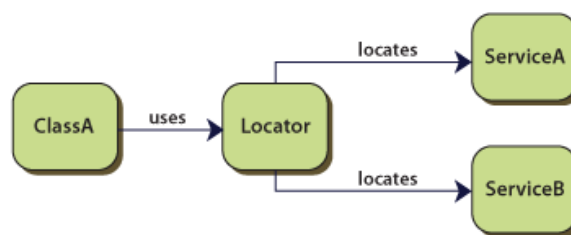


Figura 51- Service Locator [MSOFT09h]

Definição da user interface: Implementação da *User Interface* propriamente dita. No desenvolvimento do protótipo foi utilizado o padrão *Model-View-ViewModel*, (ver secção [4.3.4.1](#)).

- a) Definição do modelo: Especificação das acções dos eventos da aplicação. A título de exemplo, para a definição de uma determinada acção despoletada por

um botão é necessário implementar as interfaces *ICommand* e *IEventAggregator*.

- b) Definição da vista: Implementação da apresentação visual dos vários controlos que a aplicação vai conter. O desenho é realizado através da manipulação de ficheiros XAML.
- c) Associação do modelo à vista.

Tendo como objectivo, estudar a viabilidade desta abordagem na gestão da informação, este módulo realiza todas as operações de CRUD (Create, Read, Update and Delete).

5.5 Visualização das parcelas com vinha

A visualização da área ocupada pela parcela com vinha é possível utilizando um serviço de *Web mapping*. O controlo utilizado tem as seguintes funcionalidades: diferentes níveis de *zoom*, controlos de navegação, visualização interactiva de mapas e de imagens de satélites. Isto é visível na Figura 52.



Figura 52 - Visualização da parcela com vinha

Na Figura 53 está representada uma parcela com vinha cujo formato se baseia num multi-polígono. Esta imagem prova que a filtragem de pontos e a utilização do sistema de camadas utilizado produzem os resultados pretendidos.



Figura 53 - Visualização multi-polígonos

Além da visualização gráfica da parcela é possível visualizar a informação de forma descritiva, como indica a Figura 54.

Parcela Processada	
Número Processo - Número Boletim:	2008006001 - 1
Geocódigo:	288335466649
Distrito:	04-Bragança
Concelho:	09-Torre de Moncorvo
Freguesia:	16-Torre de Moncorvo
Local:	Torre de Moncorvo
Nome Explorador:	António Júlio Lopes
Nome Proprietário:	António Júlio Lopes
Área Potencial (ha):	1,0132

Figura 54 - Descrição da parcela

5.6 Testes e resultados

Com o objectivo de testar a performance da aplicação foram realizados testes de carga à aplicação anterior e aos protótipos implementados segundo o modelo RIA. A ferramenta utilizada na realização de testes foi o *Visual Studio Team System 2008*⁴⁷. Os sites foram devidamente configurados no IIS versão 6.0, numa máquina a correr o *Windows XP Professional*.

⁴⁷ “Visual Studio Team System 2008” - <http://msdn.microsoft.com/pt-br/teamssystem/default.aspx>

A aplicação vai ser utilizada num ambiente de Intranet. Com o objectivo de replicar com a maior fidelidade possível o cenário de utilização, a configuração correcta para os testes de carga num ambiente LAN (Local Area Network), segundo [Kumar08], baseia-se num número constante de utilizadores durante um determinado período de tempo.

Cenário 1

Objectivo: Testar a performance no acesso à base de dados e testar a agilidade e performance do desenho da *User Interface* da apresentação dos resultados por parte da interface.

Hardware: Pentium Centrino 1,73GHz com 1GB de RAM

Duração do teste: 20 minutos.

Número de utilizadores: 100 (o teste de carga envia pedidos ao IIS, imitando o comportamento de 100 utilizadores)

Caso de Uso: Direitos de Plantação → Consultar direito

Resultados esperados: Ao inserir o código correspondente a um direito de plantação, espera-se que retorne os dados correspondentes.

As figuras seguintes descrevem o tempo que demorou a responder cada pedido a um determinado URL. Na Figura 55 está representado o tempo que demoraram os acessos às páginas ‘Default.aspx’ (AJAX) e ‘ConsultarDireitosPlantacao.aspx’. Ao arrancar a aplicação, surge um pico que corresponde ao carregar das páginas, estabilizando de seguida, comportando-se de forma uniforme. Após realizado o teste de carga o tempo médio de resposta por parte do conjunto de páginas anteriormente referidas foi de 0,024 segundos. Ver informações complementares no anexo A.6.

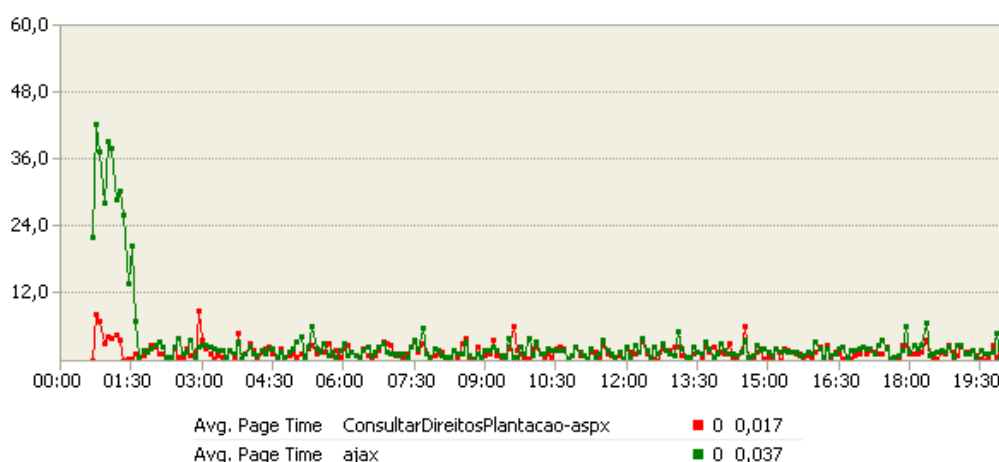


Figura 55 - Comportamento da aplicação baseada em Ajax

Na Figura 56 está representado o comportamento da aplicação, baseada em Silverlight. O teste de carga consiste em realizar pedidos directamente à página ‘Default.aspx’ (silverlight), ao ‘controlo Silverlight’ (SL_UI_Framework.xap) e ao ‘serviço’ (DataService.svc). Ao arrancar a aplicação, surge um pico que corresponde ao carregar do conteúdo para o *plugin* (típico deste tipo RIA), estabilizando de seguida comportando-se de forma uniforme. Após realizado o teste de carga o tempo médio de resposta foi de 0,021 segundos. Ver informações complementares no anexo A.6.

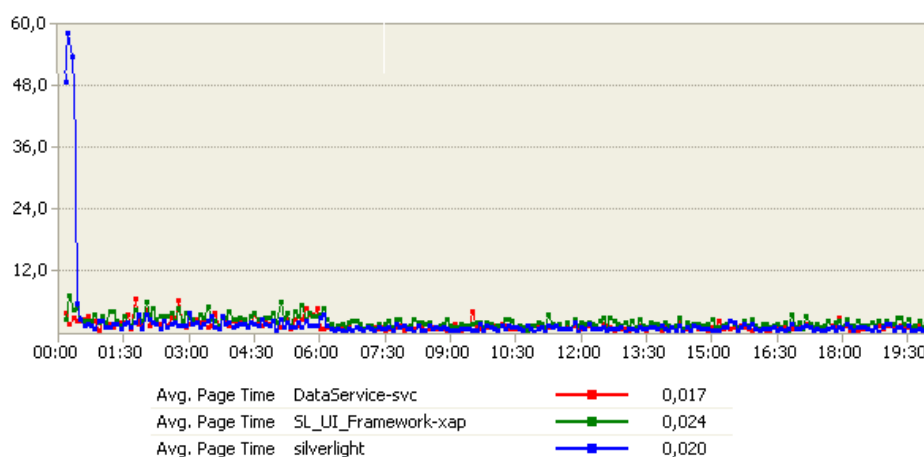


Figura 56 - Aplicação baseada em Silverlight (sistema hierárquico)

Na Figura 57 está representado o comportamento da aplicação baseada em Silverlight, utilizando a CAL. O comportamento é semelhante ao da aplicação anterior, com uma melhoria no tempo médio e com um comportamento mais uniforme. Neste teste o tempo registado foi de 0,015 segundos. Ver informações complementares no anexo A.6.

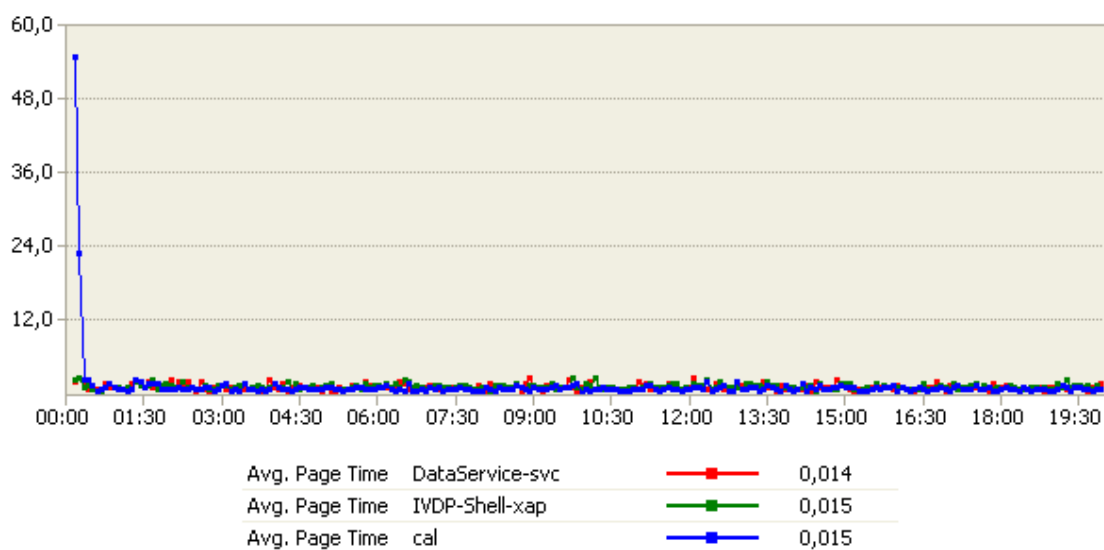


Figura 57 - Aplicação baseada em Silverlight (CAL)

Cenário 2

Objectivo: Testar a performance no acesso à base de dados e testar a agilidade da apresentação dos resultados por parte da interface. Este teste também pretende mensurar qual a estrutura de dados que oferece menor tempo de resposta. As estruturas de dados em comparação são as baseadas em *Hash Tables* e as classes *DatSetSupport* (ver secção [4.3.2](#)).

Hardware: Pentium Centrino 1,73GHz com 1GB de RAM

Duração do teste: 5 minutos.

Número de utilizadores: 100 (o teste de carga envia pedidos ao IIS, imitando o comportamento de 100 utilizadores).

Caso de Uso: Boletim de campo → Processar os boletins de campo pendentes

Resultados esperados: Apresentação de uma *Grid* (tabela) com aproximadamente 500 boletins de campo pendentes. Cada boletim de campo tem cinco atributos como características base.

As Figuras 58 e 59 representam respectivamente o comportamento da aplicação utilizando a estrutura de tipo de dados baseadas em *HashTable* e a baseada nas *custom classes* (*DataSetSupport*).

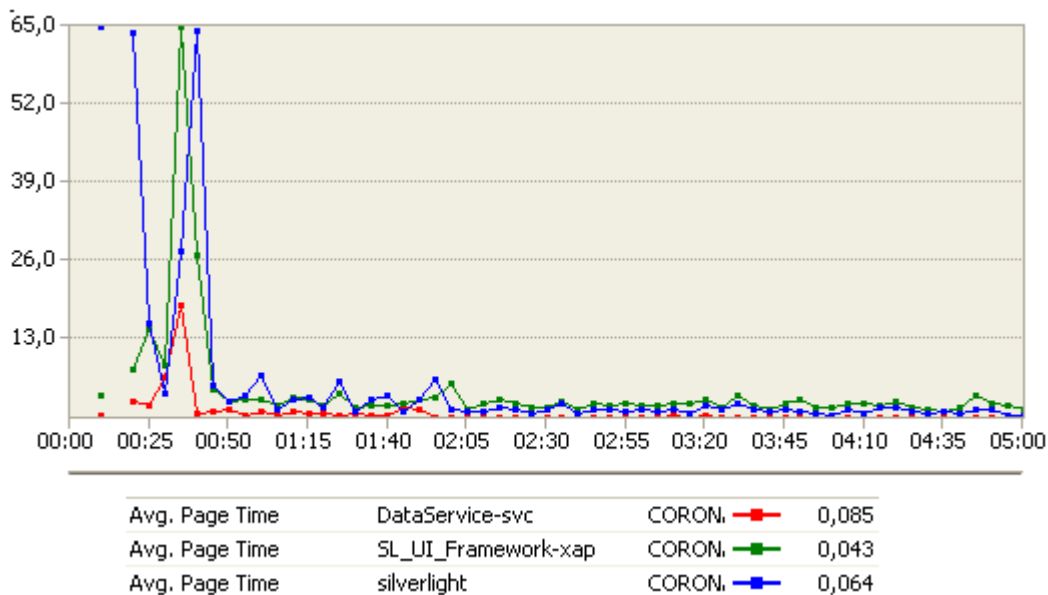


Figura 58 – Teste de carga (*HashTables*)

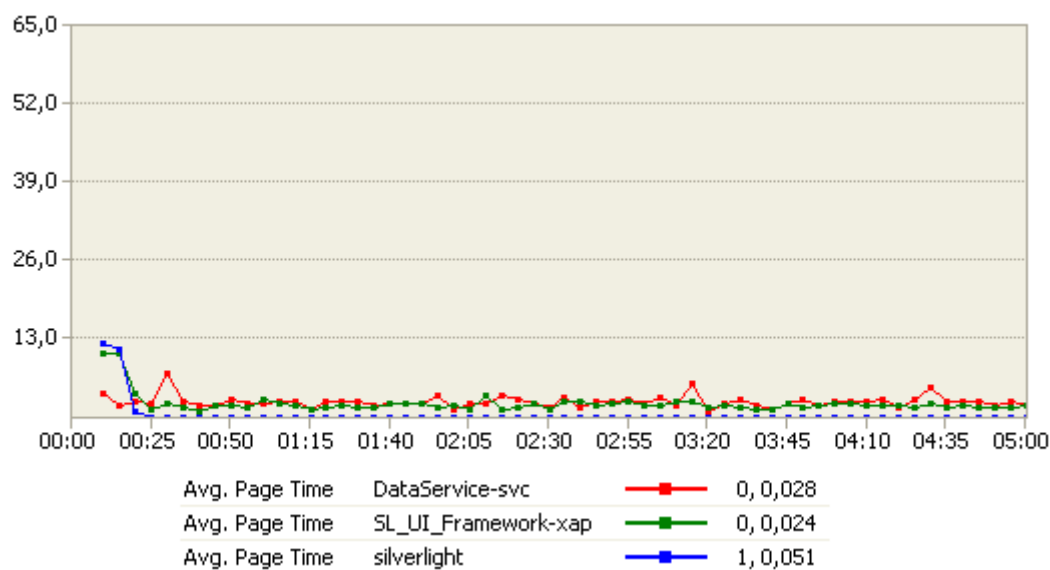


Figura 59 – Teste de carga (DataSetSupport)

A abordagem utilizando *DataSetSupport* demonstrou ser a mais eficaz apresentando um tempo de resposta inferior e um comportamento mais regular comparativamente a abordagem utilizando *HashTables*. O tempo médio foi de 0,034 segundos, ver informações complementares no anexo A.6. Enquanto o tempo médio de resposta dada pela aplicação utilizando uma estrutura baseada em *HashTables* foi de 0,064 segundos, ver informações complementares no anexo A.6.

5.7 Sumário

Através da implementação do protótipo foi possível comprovar o dinamismo proporcionado pelas RIA. Com utilização da CAL foi possível alcançar o nível de modularidade desejado, com a “mais-valia” de poder reutilizar o código e desenvolver aplicações em simultâneo para Silverlight e WPF. As estruturas de dados utilizadas na camada intermédia de serviços apresentaram um comportamento constante e consistente. Por fim, a utilização de serviços de *Web mapping* demonstrou ser uma opção válida na visualização de informação correspondente às parcelas com vinha, apresentando como grande vantagem o custo associado comparativamente com as tecnologias proprietárias comerciais.

6. Conclusões e trabalho futuro

Este capítulo apresenta as principais contribuições deste trabalho assim como apresenta propostas de desenvolvimento futuro.

6.1 Sumário das contribuições gerais

O trabalho descrito nesta tese assenta no desenvolvimento de aplicações Web para a gestão e classificação de parcelas agrícolas. As principais contribuições proporcionadas por este trabalho são: o desenvolvimento de uma metodologia que permite a transformação de aplicações Web 1.0 em Web 2.0 e a visualização de informação georreferenciada utilizando serviços de *Web mapping*.

A metodologia permite criar aplicações com um elevado nível de desacoplamento, possibilitando o desenho de módulos completamente independentes sobre uma arquitectura padronizada. Especificação de cada módulo do sistema modular agrupa um conjunto de funcionalidades específicas do Sistema de Informação Geográfico.

Foi ainda desenvolvida uma camada de serviços genérica que possibilita o tratamento de estruturas e tipos de dados de forma dinâmica. Podendo ser utilizada noutro tipo de aplicações que necessitem da transformação de DataSet em estruturas de dados genéricas.

A aplicação da metodologia proposta permitiu a preservação na totalidade, das camadas de negócio e dados. Desta forma, foi garantida a integridade de todas as regras de negócio utilizadas na classificação e gestão de parcelas com vinha.

Através do estudo realizado foi possível desenvolver métodos de visualização de informação georreferenciada relativa as parcelas agrícolas com vinha, utilizando serviços de *Web mapping*.

Com base nesta metodologia, foi o desenvolvido um protótipo onde foi possível avaliar a viabilidade das RIA no desenvolvimento de aplicações Web.

Analisando os objectivos iniciais e os resultados obtidos com o estudo realizado e a construção do protótipo, conclui-se que o problema inicialmente proposto para a realização desta tese foi ultrapassado.

6.2 Trabalho futuro

Na abordagem de desenvolvimento utilizando a CAL só foi implementado o módulo “Módulo de Gestão Direitos de Plantação”. Visto ser esta abordagem que apresenta um melhor desenho do código, sustentabilidade, facilidade na realização de testes e garantias reutilização de código deverá ser replicada para os restantes módulos do sistema.

No protótipo os mapas foram utilizados para desenhar a área de uma determinada parcela com vinha. Trabalho futuro passa por representar no mapa mais informação sobre a parcela ou conjunto de parcelas de uma determinada região. De forma a facilitar a análise e a tomada de decisões.

Com vista a desenvolver GUI dinâmicas e apelativas tendo em consideração o grupo de utilizadores que vai interagir com a aplicação, realizar uma integração com equipas multi-disciplinares, nomeadamente com uma equipa de *designers*.

Uma das limitações apontadas as RIA é o fraco suporte ao desenvolvimento de aplicações em que o factor acessibilidade é vital. Alguns sites públicos e institucionais são obrigados a desenvolver aplicações que tenham em conta a transmissão de informação de forma adequada a pessoas com deficiência. Desenvolvimento futuro, passa por estudar o trabalho desenvolvido pela [W3CARIA09] e implementar mecanismos que permitam a difusão de informação de forma universal sem restrições.

7. Referências bibliográficas

- [Alesheikh02] ALESHEIKH, A., HELALI, H. & BEHROZ, H. (2002) Web GIS: Technologies and Its Applications. *Proceedings, ISPRS Comission IV Symposium*. Canada.
- [ArcView09] ArcView – The geographic Information System for Everyone. <http://www.esri.com/software/arcview/>, acedido pela última vez em 17 de Março de 2009
- [Avison01] AVISON, D., FITZGERALD, G. & POWELL, P. (2001) Reflections on information systems practice, education and research: 10 years of the Information Systems Journal. *Information Systems Journal*, 11, 3-22.
- [Bowen05] BOWEN, P. A., BOGDANOFF, C. P., ESTERGAARD, B. F., MARSH, S. G., USHER, K. B., SNAITH, C. A. S. & FRANK, G. (2005) Geology and wine 10: Use of geographic information system technology to assess viticulture performance in the Okanagan and Similkameen valleys, British Columbia. *GEOSCIENCE CANADA*, 32, 161-176.
- [Bozzon06] BOZZON, A., COMAI, S., FRATERNALI, P. & CARUGHI, G. T. (2006) Conceptual modeling and code generation for rich internet applications. *Proceedings of the 6th international conference on Web engineering*. Palo Alto, California, USA, ACM.
- [Brambilla08] BRAMBILLA, M., PRECIADO, J. C., LINAJE, M. & SANCHEZ-FIGUEROA, F. (2008) Business Process-Based Conceptual Design of Rich Internet Applications. *Web Engineering, 2008. ICWE '08. Eighth International Conference on*.

- [Church02] CHURCH, R. L. (2002) Geographical information systems and location science. *Comput. Oper. Res.*, 29, 541-562.
- [ESRI09a] What is GIS?. <http://www.gis.com/whatisgis/index.html>, acessido pela última vez em 08 de Março de 2009
- [ESRI09b] In Tuscany, Italy, Old Style Vineyard Management Meets New Style Precision Farming. <http://www.esri.com/news/arcnews/spring04articles/in-tuscany.html>, acessido pela última vez em 17 de Março de 2009
- [ESRI09c] ESRI Products Overview. <http://www.esri.com/products/index.htm>, acessido pela última vez em 30 Março de 2009
- [Farrell07] FARRELL, J. & NEZLEK, G. S. (2007) Rich Internet Applications The Next Stage of Application Development. *Information Technology Interfaces, 2007. ITI 2007. 29th International Conference on.*
- [GOOGLE09] Google Maps JavaScript API Example: Simple Polygon. <http://code.google.com/apis/maps/documentation/examples/polygon-simple.html>, acessido pela última vez em 10 de Abril de 2009.
- [GOSSMAN09] Tales from the Smart Client: Introduction to Model/View/ViewModel pattern for building WPF apps. <http://blogs.msdn.com/johngossman/archive/2005/10/08/478683.aspx>, acessido pela última vez em 4 de Junho de 2009.
- [GRASS09] GRASS: Introduction. <http://grass.osgeo.org/intro/general.php>, acessido pela última vez em 30 de Março de 2009
- [ISO21109] ISO/TC 211 (Geographic Information and Geomatic). <http://www.isotc211.org/>, acessido pela última vez em 30 de Março de 2009
- [JAVA09] Rich Internet Applications - Java SE 7. <http://wikis.sun.com/display/JavaSE/Rich+Internet+Applications>, acessido pela última vez em 10 de Abril de 2009

- [King05] KING, G. R., PIEKARSKI, W. & THOMAS, B. H. (2005) ARVino - outdoor augmented reality visualization of viticulture GIS data. *Mixed and Augmented Reality, 2005. Proceedings. Fourth IEEE and ACM International Symposium on.*
- [Kumar08] KUMAR, S. & SUBASHINI, S. (2008) *Software Testing with Visual Studio Team System 2008*, Pack Publishing.
- [Laudon04] LAUDON K, C. & LAUDON J, P. (2004) *Management Information Systems: Managing the Digital Firm*, Prentice Hall.
- [Linaje07] LINAJE, M., PRECIADO, J. C. & SANCHEZ-FIGUEROA, F. (2007) Engineering Rich Internet Application User Interfaces over Legacy Web Models. *Internet Computing, IEEE*, 11, 53-59.
- [Maged08] MAGED, SCOTCH, M., CHEUNG, K.-H. & BURDEN, D. (2008) Web GIS in practice VI: a demo "playlist" of geo-mashups for public health neogeographers. *International Journal of Health Geographics*, 7, 38.
- [Maohua01] MAOHUA, W. (2001) Possible adoption of precision agriculture for developing countries at the threshold of the new millennium. *Computers and Electronics in Agriculture*, 30, 45-50.
- [MAPINFO09] “Essential tools from the location intelligence experts”. <http://www.mapinfo.com/products>, acessado pela última vez em 30 de Março de 2009
- [Martin91] MARTIN, D. (1991) *Geographic information systems and their socioeconomic applications*, Taylor & Francis.
- [MFOWLER02] FOWLER, M. (2002) *Patterns of Enterprise Application Architecture*, Addison-Wesley Professional.
- [MFOWLER09a] P of EAA: Table Module. <http://martinfowler.com/eaCatalog/tableModule.html>, acessado pela última vez em 20 de Maio de 2009

- [MFOWLER09b] P of EAA: Table Data Gateway.
<http://martinfowler.com/eaCatalog/tableDataGateway.html>,
acedido pela última vez em 20 de Maio de 2009
- [MFOWLER09c] P of EAA: Plugin.
<http://martinfowler.com/eaCatalog/separatedInterface.html>,
acedido pela última vez em 20 de Maio de 2009
- [MFOWLER09d] Event Aggregator.
<http://www.martinfowler.com/eaDev/EventAggregator.html>,
acedido pela última vez em 20 de Maio de 2009
- [MFOWLER09e] Inversion of Control Containers and the Dependency Injection.
<http://martinfowler.com/articles/injection.html#InversionOfControl>,
acedido pela última vez em 20 de Maio de 2009
- [MFOWLER09f] Separated Presentation.
<http://www.martinfowler.com/eaDev/SeparatedPresentation.html>,
acedido pela última vez em 20 de Maio de 2009
- [MFOWLER09g] Inversion of Control.
<http://martinfowler.com/bliki/InversionOfControl.html>,
acedido pela última vez dia 30 de Maio de 2009
- [MSOFT09] Microsoft Silverlight: Light Up the Web.
<http://www.microsoft.com/silverlight/overview/faq.aspx>,
acedido pela última vez em 10 de Abril de 2009.
- [MSOFT09b] Virtual Earth Tile System. <http://msdn.microsoft.com/en-us/library/bb259689.aspx>,
acedido pela última vez dia 20 de Abril de 2009
- [MSOFT09c] Reflection (C# Programming Guide).[http://msdn.microsoft.com/en-us/library/ms173183\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/ms173183(VS.80).aspx),
acedido pela última vez dia 20 de Maio de 2009

- [MSOFT09d] Improving DataSet Serialization and Remoting Performance. <http://support.microsoft.com/kb/829740>, acessido pela última vez dia 20 de Maio de 2009
- [MSOFT09e] Binary Serialization of DataSets. <http://msdn.microsoft.com/en-us/magazine/cc163911.aspx>, acessido pela última vez dia 20 de Maio de 2009
- [MSOFT09f] Application Development Challenges. <http://msdn.microsoft.com/en-us/library/dd490815.aspx>, acessido pela última vez dia 20 de Maio de 2009
- [MSOFT09g] ADO.NET DataSets. <http://msdn.microsoft.com/en-us/library/zb0sdh0b.aspx>, acessido pela última vez dia 20 de Maio de 2009
- [MSOFT09h] Service Locator. <http://msdn.microsoft.com/en-us/library/cc707905.aspx>, acessido pela última vez dia 30 de Maio de 2009
- [Mi-Jeong03] MI-JEONG, K., EUNKYU, L., BYOUNG-WOO, O. & MIN-SOO, K. (2003) A study on geographic data services based on dynamically generated flash in wireless Internet. *Geoscience and Remote Sensing Symposium, 2003. IGARSS '03. Proceedings. 2003 IEEE International*.
- [Moroney08] MORONEY, L. (2008) *Introducing Microsoft Silverlight 2*, Microsoft Press.
- [OGC09] OpenGIS Standards and Specifications. <http://www.opengeospatial.org/standards>, acessido pela última vez em 30 de Março de 2009
- [Ozer08] Ozer, J. Distribute Expertise: Flash vs. Silverlight. http://digitalcontentproducer.com/videoencodvd/revfeat/flash_vs_silverlight/index.html, acessido pela última vez 10 de Abril de 2009

- [Preciado05] PRECIADO, J. C., LINAJE, M., SANCHEZ, F. & COMAI, S. (2005) Necessity of methodologies to model Rich Internet Applications. Proceedings of the Seventh IEEE International Symposium on Web Site Evolution. IEEE Computer Society.
- [Preciado07] PRECIADO, J. C., LINAJE, M., COMAI, S. & SANCHEZ-FIGUEROA, F. (2007) Designing Rich Internet Applications with Web Engineering Methodologies. *Web Site Evolution, 2007. WSE 2007. 9th IEEE International Workshop on.*
- [Preciado08] PRECIADO, J. C., TRIGUEROS, M. L. & SÁNCHEZ-FIGUEROA, F. (2008) Adapting Web 1.0 User Interfaces to Web 2.0 Multidevice User Interfaces using RUX-Method. *J. UCS*, 14, 2239-2254.
- [Riquelme05] RIQUELME, F. J. M. & RAMOS, A. B. (2005) Land and water use management in vine growing by using geographic information systems in Castilla-La Mancha, Spain. Albacete, SPAIN, Elsevier Science Bv.
- [Riquelme06] Riquelme, F. J. M. (2006). “Estudo da divisão de zonas vitícolas denominação de origem Valdepenas”, Universidade de Castilla-La Mancha. Estudo disponível em http://www.dovaldepenas.es/content/docs/docsDoc_55.pdf, acedido pela última vez em 30 de Março de 2009.
- [Skop98] SKOP, E. & SØRENSEN, P. B. (1998) GIS-based modelling of solute fluxes at the catchment scale: a case study of the agricultural contribution to the riverine nitrogen loading in the Vejle Fjord catchment, Denmark. *Ecological Modelling*, 106, 291-310.
- [Smith98] SMITH, L. & PETER, F. (1998) Applied Research into the Integration of Spatial Information Systems with Viticultural Research & Vineyard Management System. *Presented at the 10th Colloquium of Spatial Information Research Center*. University of Otago, New Zeland.

- [Steiniger09] Stefan Steiniger and Robert Weibel (2009), "GIS Software - A description in 1000 words". University of Calgary (CA) and University of Zurich (CH), http://www.geo.unizh.ch/publications/sstein/gissoftware_steiniger2008.pdf, acessido pela última vez 30 de Março de 2009.
- [Suresh09] Suresh, G. C# Reflection and Dynamic Method Invocation. <http://gsraj.tripod.com/dotnet/reflection.html>, acessido pela última vez em 15 de Abril de 2009
- [Tschiedel02] TSCHIEDEL, M. & FERREIRA, M. F. (2002) INTRODUCTION THE PRECISION FARMING: CONCEPTS AND ADVANTAGES. *Ciência Rural*, 32, 159-163.
- [UIPATT09] User Interface Design Library. – Module Tabs, <http://ui-patterns.com/pattern/ModuleTabs>, acessido pela última vez em 30 de Maio de 2009
- [Velasco08] VELASCO, C. A., DENEV, D., STEGEMANN, D. & MOHAMAD, Y. (2008) A web compliance engineering framework to support the development of accessible rich internet applications. *Proceedings of the 2008 international cross-disciplinary conference on Web accessibility (W4A)*. Beijing, China, ACM.
- [W3CARIA09] W3C World Wide Web Consortium. Accessible Rich Internet Applications (WAI-ARIA) 1.0, Fevereiro de 2009. <http://www.w3.org/TR/wai-aria/>, acessido pela última vez em 15 de Junho de 2009
- [Wenz08] WENZ, C. (2008) Essential Silverlight 2 Up-to-Date, O'REILLY.
- [Young99] YOUNG, J. A., CHRISTENSEN, B. M., SCHAAD, M. S., HERDENDORF, M. E., VANCE, G. F. & MUNN, L. C. (1999) A geographic information system to identify areas for alternative crops in northwestern Wyoming. *J. Janick (ed.), Perspectives on new crops and new uses. ASHS Press*. Alexandria, VA.

Anexos

Anexo A.1: Casting

```

/*****
* Este trecho de código tem como objecto demonstrar a dificuldade na
* realização de conversões de tipos de dados. Pois sempre que muda um campo no
* DataSet é necessario modificar o código. Logo esta abordagem não é razoável
* quando o factor escalabilidade é importante. Assim como para poder encontrar
* um valor numa determinada DataTable num determinado DataSet é necessário
* realizar ciclos e conversões de tipos de dados simultaneamente, o que
* degrada a performance da aplicação.
*****/

//...

Dictionary<string, List<object>> insert_DB = new Dictionary<string, List<object>>();

//...

DataTable dtplantacaoDT = tmpDS.Tables["t_dtplantacao"];
DataRow pDR = dtplantacaoDT.Rows[0];

decimal ano_inicio =
Convert.ToDecimal(insert_DB["txt_ano_inicio"][0].ToString().Replace('.', ','));

decimal ano_fim = Convert.ToDecimal(insert_DB["txt_ano_fim"][0].ToString().Replace('.', ','),
',');
int tipo_dir = Convert.ToInt32(insert_DB["combo_dir"][0]);

pDR["anofim"] = ano_fim;
pDR["anoinicio"] = ano_inicio;
pDR["tipodireito"] = tipo_dir ;
pDR["iddtplantacao"] = insert_DB["txt_cod_dir"][0];
pDR["iddtorigem"] = insert_DB["txt_cod_dir_orig"][0];
pDR["descr"] = insert_DB["txt_area_tot_des"][0];

pDR["areatotaldescr"] =
Convert.ToDecimal(insert_DB["txt_area_tot_des"][0].ToString().Replace('.', ','));

pDR["nometitular"] = insert_DB["txt_nome_tit"][0];

//...

foreach (DataRow dr in tmpDS.Tables["t_pquintas"].Rows)
{

descricao = pBllDomain.getDescrFromIDSdominio(SdomTabela.enumTableSdominio.T_QUINTA,
int.Parse(dr["idquinta"].ToString()));

}

//...

```

Anexo A.2: Custom Classes

```

/*****
* Estas classes representam o esqueleto do DataSet e Datatable, podendo ser
* utilizadas tanto no Silverlight como no WCF.
* As classes foram adaptadas dos sites abaixo referenciados. Foi criados novos
* estados que permitem a uma gestão dos DataSet. Assim como, foram criadas métodos
* auxiliares que auxiliam a transformação do tipo de dados criados em collections
* como por exemplo a conversão de uma DataSetData num IEnumerable .
*
* "How to Display DataSet in Silverlight DataGrid"
* http://silverlight.net/forums/p/16733/55609.aspx
*
* "Dataset surrogate for silverlight"
* http://www.codeproject.com/KB/silverlight/Dataset_for_silverlight.aspx
*
*****/

/// <summary>
/// Os atributos desta classe representam uma coluna de uma DataTable
/// </summary>
public class DataColumnInfo
{
    [DataMember]
    public string ColumnName { get; set; }
    [DataMember]
    public string ColumnTitle { get; set; }
    [DataMember]
    public string DataTypeName { get; set; }
    [DataMember]
    public bool IsRequired { get; set; }
    [DataMember]
    public bool IsKey { get; set; }
    [DataMember]
    public bool IsReadOnly { get; set; }
    [DataMember]
    public int DisplayIndex { get; set; }
    [DataMember]
    public string EditControlType { get; set; }
    [DataMember]
    public int MaxLength { get; set; }
}

/// <summary>
/// Cria uma lista de colunas
/// </summary>
public class DataTableInfo
{
    [DataMember]
    public string TableName { get; set; }

    [DataMember]
    public ObservableCollection<DataColumnInfo> Columns { get; set; }
}

/// <summary>
/// Estrutura de dados que representa uma DataTable
/// </summary>
public class DataTableData
{
    [DataMember]
    public ObservableCollection<DataTableInfo> Tables { get; set; }

    //...
}

```

```

public static DataTableData FromDataTable(DataTable dt, bool isReadOnly)
{
    DataTableData dsd = new DataTableData();
    dsd.Tables = new ObservableCollection<DataTableInfo>();
    DataTable t = dt;
    DataTableInfo tableInfo = new DataTableInfo {TableName = t.TableName};
    dsd.Tables.Add(tableInfo);
    tableInfo.Columns = new ObservableCollection<DataColumnInfo>();
    foreach (DataColumn c in t.Columns)
    {
        DataColumnInfo col = new DataColumnInfo {
            ColumnName = c.ColumnName,
            ColumnTitle = c.ColumnName,
            DataTypeName = c.DataType.FullName,
            MaxLength=c.MaxLength,
            IsKey=c.Unique,
            IsReadOnly=(c.Unique || c.ReadOnly),
            IsRequired = !c.AllowDBNull
        };
        if (c.DataType == typeof(System.Guid))
        {
            col.IsReadOnly = true;
            col.DisplayIndex = -1;
        }
        tableInfo.Columns.Add(col);
    }

    //...

    /* Conversões de strings to xml porque o
    * DataTable não suporta este métodos
    */
    string resultXML;
    using (StringWriter sw = new StringWriter())
    {
        dt.WriteXml(sw);
        resultXML = sw.ToString();
    }
    dsd.DataXMLOriginal = resultXML;
    if (!isReadOnly)
    {
        string resultSchema;
        using (StringWriter sw = new StringWriter())
        {
            dt.WriteXmlSchema(sw);
            resultSchema = sw.ToString();
        }
        dsd.DataXMLMappings = "";
        dsd.DataXMLSchema = resultSchema;
    }
    return dsd; }
}

/// <summary>
/// Estrutura de dados que representa uma DataSet
/// </summary>
public class DataSetData
{
    [DataMember]
    public ObservableCollection<DataTableInfo> Tables { get; set; }

    //...

    public static DataSetData FromDataSet(DataSet ds, bool isReadOnly)
    {
        DataSetData dsd = new DataSetData();
        dsd.Tables = new ObservableCollection<DataTableInfo>();
        foreach (DataTable t in ds.Tables)
        {
            DataTableInfo tableInfo = new DataTableInfo {TableName = t.TableName};
            dsd.Tables.Add(tableInfo);
        }
    }
}

```

```
tableInfo.Columns = new ObservableCollection<DataColumnInfo>();
foreach (DataColumn c in t.Columns)
{
    DataColumnInfo col = new DataColumnInfo {
        ColumnName = c.ColumnName,
        ColumnTitle = c.ColumnName,
        DataTypeName = c.DataType.FullName,
        MaxLength=c.MaxLength,
        IsKey=c.Unique,
        IsReadOnly=(c.Unique || c.ReadOnly),
        IsRequired = !c.AllowDBNull
    };
    if (c.DataType == typeof(System.Guid))
    {
        col.IsReadOnly = true;
        col.DisplayIndex = -1;
    }
    tableInfo.Columns.Add(col);
}
dsd.DataXMLOriginal = ds.GetXml();
if (!isReadOnly)
{
    dsd.DataXMLMappings = "";
    dsd.DataXMLSchema = ds.GetXmlSchema();
}
return dsd;
}
```

Anexo A.3: Reflection

```

/*****
* Esta classe utiliza Reflection, transformando uma estrutura de dados
* fracamente "tipada" numa estrutura fortemente "tipada"
*****/

using System.Reflection;
using System.Reflection.Emit;

//...

/// <summary>
/// Cria dinamicamente a estrutura de dados equivalente ao DataSet, utilizando
/// Reflection recriando desta forma o tipo de dados origina do DataSet
/// </summary>
public class DynamicDataBuilder
{
    /// <summary>
    /// Cria dinamicamente o tipo de dados
    /// </summary>
    private static System.Type
    BuildDataObjectType(ObservableCollection<DataService.DataColumnInfo> Columns, string
    DataObjectName)
    {
        AssemblyBuilder assemblyBuilder = AppDomain.CurrentDomain.DefineDynamicAssembly(new
        AssemblyName("AprimoDynamicData"), AssemblyBuilderAccess.Run);
        ModuleBuilder moduleBuilder = assemblyBuilder.DefineDynamicModule("DataModule");
        TypeBuilder tb = moduleBuilder.DefineType(DataObjectName, TypeAttributes.Public |
        TypeAttributes.Class,
        typeof(DataObject));

        ConstructorBuilder constructor = tb.DefineDefaultConstructor(MethodAttributes.Public |
        MethodAttributes.SpecialName | MethodAttributes.RTSpecialName);
        foreach (var col in Columns)
        {
            string propertyName = col.ColumnName.Replace(' ', '_');
            Type dataType = System.Type.GetType(col.DataTypeName, false, true);
            if (dataType != null)
            {
                FieldBuilder fb = tb.DefineField("_" + propertyName, dataType, FieldAttributes.Private);
                PropertyBuilder pb = tb.DefineProperty(propertyName, PropertyAttributes.HasDefault,
                dataType, null);

                MethodBuilder getMethod = tb.DefineMethod("get_" + propertyName,
                MethodAttributes.Public |
                MethodAttributes.HideBySig |
                MethodAttributes.SpecialName,
                dataType,
                Type.EmptyTypes);
                ILGenerator ilgen = getMethod.GetILGenerator();
                //Emit Get property, return _prop
                ilgen.Emit(OpCodes.Ldarg_0);
                ilgen.Emit(OpCodes.Ldfld, fb);
                ilgen.Emit(OpCodes.Ret);
                pb.SetGetMethod(getMethod);
                MethodBuilder setMethod = tb.DefineMethod("set_" + propertyName,
                MethodAttributes.Public |
                MethodAttributes.HideBySig |
                MethodAttributes.SpecialName,
                null,
                new Type[] { dataType });
                ilgen = setMethod.GetILGenerator();
                LocalBuilder localBuilder = ilgen.DeclareLocal(typeof(String[]));
                //Emit set property, _Prop = value;
            }
        }
    }
}

```

```

        ilgen.Emit(OpCodes.Ldarg_0);
        ilgen.Emit(OpCodes.Ldarg_1);
        ilgen.Emit(OpCodes.Stfld, fb);
        //Notify Change:
        Type[] wlParams = new Type[] { typeof(string[]) };
        MethodInfo notifyMI = typeof(DataObject).GetMethod("NotifyChange",
        BindingFlags.NonPublic |
        BindingFlags.Instance,
        null,
        CallingConventions.HasThis,
        wlParams, null);

        //NotifyChange Property change
        ilgen.Emit(OpCodes.Ldc_I4_1);
        ilgen.Emit(OpCodes.Newarr, typeof(String));
        ilgen.Emit(OpCodes.Stloc_0);
        ilgen.Emit(OpCodes.Ldloc_0);
        ilgen.Emit(OpCodes.Ldc_I4_0);
        ilgen.Emit(OpCodes.Ldstr, propertyName);
        ilgen.Emit(OpCodes.Stelem_Ref);
        ilgen.Emit(OpCodes.Ldarg_0);
        ilgen.Emit(OpCodes.Ldloc_0);
        // call notifyChange function
        ilgen.Emit(OpCodes.Call, notifyMI, null);
        ilgen.Emit(OpCodes.Ret);
        pb.SetSetMethod(setMethod);
    }
}
System.Type rowType = tb.CreateType();
//assemblyBuilder.Save("DynamicData.dll");
return rowType;
}

//...

/// <summary>
/// Transforma a estrutura de dados DataSetData num tipo de dados System.Collection com
/// o objectivo de facilitar o preenchimento "binding" de controlos da User Interface,
/// como por exemplo uma ListBox ou DataGridView
/// </summary>
public static IEnumerable GetEnumFromComplexData(DataService.DataSetData data, string
tableName, ComplexData complex)
{
    if (data.Tables.Count() == 0)
        return null;

    DataService.DataTableInfo tableInfo = null;

    for (int i = 0; i < data.Tables.Count; i++)
    {
        if (data.Tables[i].TableName == tableName)
            tableInfo = data.Tables[i];
    }
    //caso não tenha encontrado a tabela
    if (tableInfo == null)
        return null;

    System.Type dataType = BuildDataObjectType(tableInfo.Columns, "MyDataObject");
    var listType = typeof(ObservableCollection<>).MakeGenericType(new[] { dataType });
    var list = Activator.CreateInstance(listType);

    for (int i=0; i < complex.ComplexDataList.Count; i++)
    {
        ComplexDataRow row = complex.ComplexDataList[i];
        var rowData = Activator.CreateInstance(dataType) as DataObject;
        if (rowData != null)
        {
            int j = 0;
            foreach (DataService.DataColumnInfo col in tableInfo.Columns)
            {
                string nome = row.RowData[j].NameField;
                string value = row.RowData[j].ValueField;

                if (nome == col.ColumnName &&

```

```

        col.DataTypeName != typeof(System.Byte[]).FullName
                                && col.DataTypeName != typeof(System.Guid).FullName
                                && col.DataTypeName !=
typeof(System.Decimal).FullName)
    {
        rowData.SetFieldValue(col.ColumnName, value, true);
        //rowData.NewRow();
    }
    else
    {
        if (col.DataTypeName == typeof(System.Decimal).FullName)
        {
            string tmp = value;
            decimal decimal_value = Convert.ToDecimal(tmp.ToString().Replace('.', ','));
            rowData.SetFieldValue(col.ColumnName, decimal_value, true);
        }
        j++;
    }
}

//estado da linha, perservar o estado das linhas dos DataSet
ComplexDataRow.DataStates state = row.State;

if (state == ComplexDataRow.DataStates.Added)
    rowData.NewRow();
if (state == ComplexDataRow.DataStates.Deleted)
    rowData.Delete();
if (state == ComplexDataRow.DataStates.Modified)
    rowData.Modified();
if (state == ComplexDataRow.DataStates.Unchanged)
    rowData.Unchanged();
listType.GetMethod("Add").Invoke(list, new[] { rowData });
}

return list as IEnumerable;
}

//...
}

```

Anexo A.4: Singleton

```
/* *****
 * Implementação do Padrão Singleton segundo as patterns & practices da
 * Microsoft
 * http://msdn.microsoft.com/en-us/library/ms998558.aspx
 * ***** */

public sealed class ServiceConvert
{
    static ServiceConvert instance = null;
    static readonly object padlock = new object();

    UIL() { }

    public static ServiceConvert Instance
    {
        get
        {
            lock (padlock)
            {
                if (instance == null)
                {
                    instance = new UIL();
                }
                return instance;
            }
        }
    }
}

// (...)

}
```

Anexo A.5: Licença de Utilização

Surgiram dúvidas sobre a possibilidade de utilizar Silverlight com o Google Maps. A licença de utilização não clara neste ponto. A Google só possibilita a utilização dos mapas através da utilização da sua API.

“ ...

10. License Restrictions. Except as expressly permitted under the Terms, or unless you have received prior written authorization from Google (or, as applicable, from the provider of particular Content), Google's licenses above are subject to your adherence to all of the restrictions below. Except as explicitly permitted in Section 7 or the Maps APIs Documentation, you must not (nor may you permit anyone else to):

10.1 access or use the Service or any Content through any technology or means other than those provided in the Service, or through other explicitly authorized means Google may designate;

...”

Citado de <http://code.google.com/apis/maps/terms.html>, acedido em 4 de Junho de 2009

Tópico de discussão em fóruns, sobre o tema “Utilização do Google Maps com Silverlight”:

<http://silverlight.net/forums/p/76721/189243.aspx#189243>

<http://forums.componentone.com/CS/forums/p/76430/212310.aspx#212310>

http://groups.google.com/group/Google-Maps-API/browse_thread/thread/4547d8dd30e46eeb/a67f7e6170602c5d?q=Googla+maps+And+silverlight#a67f7e6170602c5d

Anexo A.6: Informação complementar sobre os testes de carga realizados

1. Informação complementar sobre o teste de carga realizado a aplicação existente baseada em Ajax

Load Test Summary

Max User Load	100
Avg. Content Length (bytes)	4.583
Avg. Page Time (sec)	0,024

Page Results

URL	Scenario	Test	Avg. Page Time (sec)
http://localhost/ajax	Scenario1	Home20minAJAX	0,037
http://localhost/ajax/ConsultarDireitosPlantacao.aspx	Scenario1	Home20minAJAX	0,017

2. Informação complementar sobre o teste de carga realizado a aplicação construída em Silverlight desenhada sobre um sistema hierárquico

Load Test Summary

Max User Load	100
Avg. Content Length (bytes)	61.025
Avg. Page Time (sec)	0,021

Page Results

URL	Scenario	Test	Avg. Page Time (sec)
http://coronas/silverlight/ClientBin/SL_UI_Framework.xap	Scenario1	Home20minSilverlight	0,024
http://coronas/silverlight	Scenario1	Home20minSilverlight	0,020
http://coronas/silverlight/Services/DataService.svc	Scenario1	Home20minSilverlight	0,017

3. Informação complementar sobre o teste de carga realizado a aplicação construída em Silverlight desenhada utilizando a *Composite Application Library*

Load Test Summary

Max User Load	100
Avg. Content Length (bytes)	20.053
Avg. Page Time (sec)	0,015

Page Results

URL	Scenario	Test	Avg. Page Time (sec)
http://localhost/cal	Scenario1	Home20minCal	0,015
http://localhost/cal/ClientBin/IVDP.Shell.xap	Scenario1	Home20minCal	0,015
http://localhost/cal/Services/DataService.svc	Scenario1	Home20minCal	0,014

4. Informação complementar sobre o teste de carga realizado a aplicação que utiliza uma estrutura de dados baseada em *HashTables*

Load Test Summary

Max User Load	100
Avg. Content Length (bytes)	73.448
Avg. Page Time (sec)	0,064

Page Results

URL	Scenario	Test	Avg. Page Time (sec)
http://coronas/silverlight/Services/DataService.svc	Scenario1	Home5minHashTables	0,085
http://coronas/silverlight	Scenario1	Home5minHashTables	0,064
http://coronas/silverlight/ClientBin/SL_UI_Framework.xap	Scenario1	Home5minHashTables	0,043

5. Informação complementar sobre o teste de carga realizado a aplicação que utiliza as custom classes *DataSetSupport*

Load Test Summary

Max User Load	100
Avg. Content Length (bytes)	71.166
Avg. Page Time (sec)	0,034

Page Results

URL	Scenario	Test	Avg. Page Time (sec)
http://coronas/silverlight	Scenario1	Home5minDataSetSupport	0,051
http://coronas/silverlight/Services/DataService.svc	Scenario1	Home5minDataSetSupport	0,028
http://coronas/silverlight/ClientBin/SL_UI_Framework.xap	Scenario1	Home5minDataSetSupport	0,024