Faculdade de Engenharia da Universidade do Porto

Mestrado Integrado em Engenharia Informática e Computação

Reengenharia de ferramenta CASE para ambiente Web 2.0 no INESC Porto

Relatório de Projecto de Mestrado do MIEIC 2007/08

Hernâni Filipe Dias Fernandes

Orientador na FEUP: Prof. João Pascoal Faria

Responsável pelo acompanhamento no INESC Porto: Eng.º Rui Barros

Abril de 2008

À minha Alma Gémea Inês Sá

Aos meus Pais

Aos meus Amigos, minha segunda família

Ao Futebol, Final Fantasy e Saint Seiya

Aos bons amigos que o curso me trouxe, à minha família e a todos os que me querem bem

Resumo

O presente relatório descreve o projecto de conclusão do Mestrado Integrado em Engenharia Informática e Computação (MIEIC), da Faculdade de Engenharia da Universidade do Porto (FEUP), realizado pelo autor no Instituto de Engenharia de Sistemas e Computadores do Porto (INESC Porto).

O projecto consistiu na evolução de uma ferramenta de desenvolvimento de aplicações para ambiente Web. A ferramenta a evoluir, o Saga – Sistema Assistido de Geração e Gestão de Aplicações, foi desenvolvida pelo INESC Porto no final dos anos 80, com o intuito de facilitar o desenvolvimento e manutenção de aplicações de *backoffice* de autarquias e reduzir dependências de plataformas. O SAGA é utilizado presentemente pela *software house* Medidata, S.A., líder em aplicações informáticas para o mercado autárquico. Devido à procura crescente no mercado autárquico de aplicações empresariais com interface Web, a Medidata contratou ao INESC Porto o projecto de evolução do Saga. Devido à arquitectura do SAGA, o caminho escolhido pela equipa de projecto passou pela implementação de uma nova camada de interface.

Numa primeira fase, foi feito o estudo da ferramenta original e efectuado o levantamento dos requisitos do projecto. Após uma pré-selecção de tecnologias candidatas à utilização na implementação da nova camada, foi efectuado um *benchmarking* sobre um conjunto de tecnologias. Deste benchmarking resultou a selecção da tecnologia a utilizar no desenvolvimento do projecto, *Gaia Ajax Widgets*. A solução encontrada passa por encapsular o componente do SAGA responsável pela lógica de negócio e acesso a dados, e passagem da gestão de interface para a nova camada, desenvolvida sobre a framework Ajax *Gaia Ajax Widgets*. A ferramenta implementada mantém a usabilidade e eficiência da sua versão anterior, sem exigir alterações nas aplicações existentes, o que possibilita a migração das aplicações para um novo ambiente, sem que daí advenham quaisquer problemas de readaptação para o utilizador final.

Na fase final do projecto (a nível académico), a equipa do projecto continua a evoluir a ferramenta, no sentido de completar a camada de interface e a integração desta com a camada de dados. Esta ferramenta será, tal como a sua antecessora, uma mais-valia para a Medidata, no sentido em que descentralizará o acesso a aplicações criadas ao longo de (quase) 2 décadas, recorrendo a tecnologias Web que permitem manter e, em alguns aspectos, melhorar a qualidade da experiência do utilizador, o que serão importantes factores competitivos para a empresa.

Agradecimentos

Gostaria de agradecer às pessoas que fizeram parte do meu dia-a-dia durante o projecto, em especial à equipa com a qual trabalhei. Assim, um muito obrigado à Anabela Monteiro, minha colega de equipa, ao Rui Barros, gestor de projecto e ao coordenador científico, Prof. João Pascoal Faria, a quem devo agradecer não só pelo apoio manifestado durante todo o projecto, mas também pela ajuda na elaboração do presente documento.

Às pessoas do INESC Porto que facilitaram a minha integração, obrigado pela paciência, simpatia e disponibilidade.

À Medidata, gostaria de agradecer o terem possibilitado a minha participação neste projecto.

Obrigado ao Prof. Ademar Aguiar pelo tempo perdido na leitura do presente relatório e pelas sugestões dadas.

À FEUP e a todos os professores e colegas que me ajudaram a crescer, como Engenheiro e como indivíduo, o meu obrigado.

Aos amigos encontrados durante o curso, principalmente Luis Ferreira, Nuno Sanches e Daniel Cordeiro, um muito obrigado.

A nível pessoal, desejo agradecer do fundo do coração às 3 pessoas que mais contribuíram para que me tornasse na pessoa que sou e para que estivesse onde estou: Hernâni de Jesus Romano Fernandes (Pai), Maria Arminda Sampaio Dias Fernandes (Mãe) e Inês Sofia Ferraz Moreira de Sá (Namorada e Melhor Amiga).

Índice de Conteúdos

Índice de Figuras

Lista de Acrónimos

AJAX – Asynchronous JavaScript And XML

API – Application Programming Interface

ASP – Active Server Pages

CASE - Computer-Aided Software Engineering

CGI – Common Gateway Interface

CLR – Common Language Runtime

CRUD – Create, Retrieve, Update and Delete

CSS – Cascading Style Sheets

DBML – Data Base Manipulation Language

DLL – Dynamic-Link Library

GUI – Graphical User Interface

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

IDE – Integrated Development Environment

LINQ – Language Integrated Query

MP3 – MPEG-1/2 Audio Layer 3

MVC – Model-View-Controller

ODBC – Open DataBase Connectivity

PNG – Portable Network Graphics

RIA – Rich Internet Applications

SAGA – Sistema Assistido de Geração e Gestão de Aplicações

SGBD – Sistemas de Gestão de Bases de Dados

UI – User Interface

VCS – Vista Compilada do Saga

W3C – World Wide Web Consortium

WPF – Windows Presentation Foundation

WPF/E – Windows Presentation Foundation / Everywhere

WMV – Windows Media Video

WMA – Windows Media Audio

WYSIWYG – What You See Is What You Get

XAML – Extensible Application Markup Language

XML – Extensible Markup Language

1 Introdução

Este capítulo destina-se à descrição do contexto do projecto com o título original 'Reengenharia de ferramenta CASE[12] para ambiente Web 2.0', realizado no Instituto de Engenharia de Sistemas e Computadores do Porto (**INESC Porto**) no âmbito do plano de estudos do Mestrado Integrado em Engenharia Informática e Computação (**MIEIC**) da Faculdade de Engenharia da Universidade do Porto (**FEUP**). Será feita uma breve apresentação do INESC Porto (instituição onde se realizou o projecto), assim como do projecto e dos seus principais objectivos. Por motivos de legibilidade, sempre que for feita alguma referência ao projecto 'Reengenharia de ferramenta CASE para ambiente Web 2.0', será utilizado o seu nome comercial, 'SagaWeb'.

1.1 Apresentação do INESC Porto

O INESC Porto – Instituto de Engenharia de Sistemas e Computadores do Porto – é uma associação privada sem fins lucrativos reconhecida como instituição de utilidade pública, tendo adquirido em 2002 o estatuto de Laboratório Associado.

Figura 1 - INESC Porto

Desenvolve actividades de investigação e desenvolvimento, consultoria, formação avançada e transferência de tecnologia nas áreas de Telecomunicações e Multimédia, Sistemas de Energia, Sistemas de Produção, Sistemas de Informação e Comunicação e Optoelectrónica.

O INESC Porto é uma instituição criada para constituir uma interface entre o mundo académico e o mundo empresarial da indústria e dos serviços, bem como a administração pública, no âmbito das Tecnologias de Informação, Telecomunicações e Electrónica, dedicando-se a actividades de investigação científica e desenvolvimento tecnológico, transferência de tecnologia, consultoria e formação avançada.

[1] CASE - Computer-Aided Software Engineering

[2] Ferramenta de software utilizada para ajudar no desenvolvimento e manutenção de software

O INESC Porto tem os seguintes objectivos estratégicos:

- Levar a cabo a produção de ciência e de tecnologia capazes de competir a nível nacional e mundial
- Colaborar na formação de recursos humanos de qualidade científica e técnica, motivados para apostar nas capacidades nacionais e na modernização do País
- Contribuir para a evolução do sistema de ensino científico e tecnológico, modernizando-o e adaptando-o às necessidades do tecido económico e social
- Promover, facilitar e incubar iniciativas empresariais que possibilitem a valorização das suas actividades de I&D e promovam o espírito de iniciativa e de risco entre os seus jovens investigadores
- Contribuir, pela realização dos objectivos anteriores, para a construção de um Portugal moderno, de uma economia sólida e de uma sociedade de qualidade. [1]

1.2 O projecto SagaWeb

No final da década de 80, no âmbito de um projecto de informatização autárquica, foi desenvolvida no INESC Porto uma ferramenta CASE, o Sistema Assistido de Geração e Gestão de Aplicações (SAGA), que tinha como principais objectivos facilitar o desenvolvimento e manutenção de aplicações de *backoffice* das autarquias e reduzir dependências de plataformas. A primeira versão do Saga surgiu em 1989 e, desde então, várias versões foram desenvolvidas. A versão actual data de 1997 e consiste numa ferramenta de desenvolvimento rápido de aplicações interactivas de bases de dados relacionais. As aplicações desenvolvidas com o Saga são aplicações cliente-servidor com interface gráfica em Windows, possuindo semelhanças com Oracle Forms e Microsoft Access.

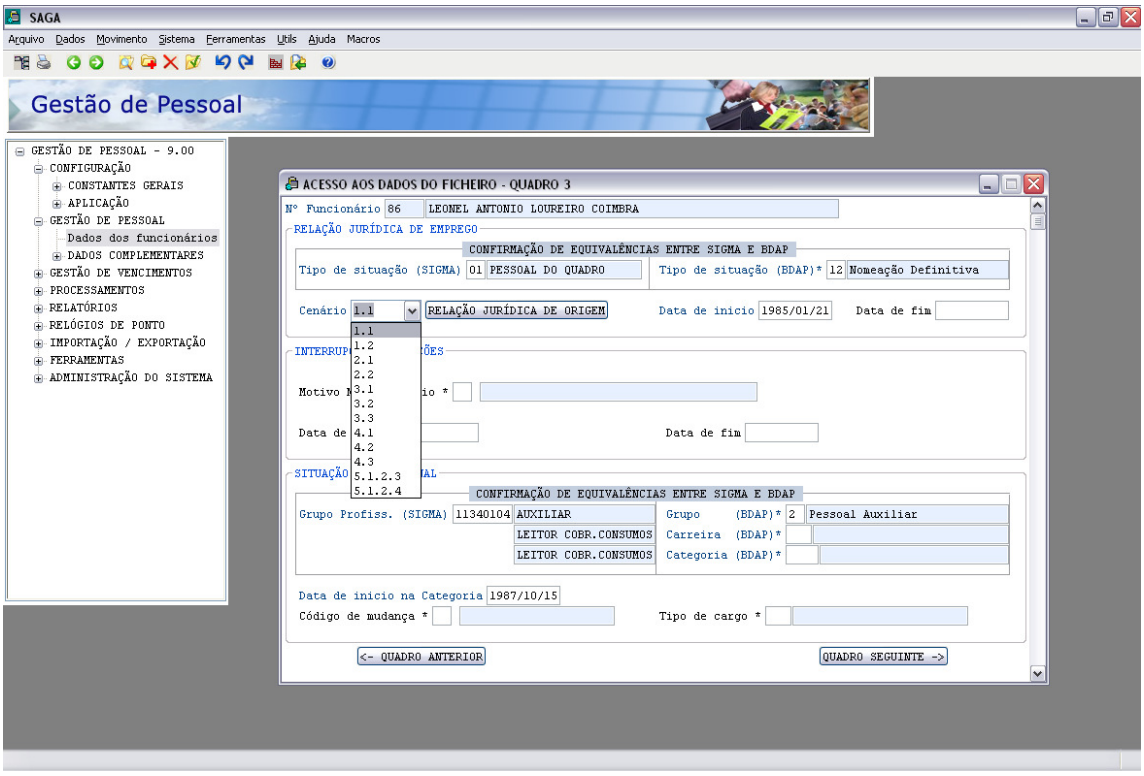


Figura 2 - Aplicação Saga

As primeiras aplicações de informatização autárquica e a própria ferramenta Saga foram transferidas para a *software house* Medidata, S.A.. A Medidata é, presentemente, líder em aplicações informáticas para o mercado autárquico, possuindo mais de 55% de cota do mercado em Portugal e uma presença significativa em Angola. A Medidata utiliza o Saga no desenvolvimento de aplicações nos seus clientes. Recentemente, a Medidata deparou-se com a necessidade urgente de migrar as suas aplicações desenvolvidas com o Saga (ambiente Windows) para ambiente Web. Nesse sentido, contratou ao INESC Porto um projecto de reengenharia da ferramenta Saga para ambiente Web.

Uma vez que a definição das aplicações criadas com o Saga é feita de uma forma declarativa e independente da tecnologia da interface para o utilizador, a solução escolhida passou por implementar uma nova camada de interface Web. Esta solução visa, essencialmente, conseguir a migração das aplicações já desenvolvidas na íntegra, sem que seja necessário a sua rescrita.

1.3 Equipa e actividades do projecto

Para além do autor, Hernâni Fernandes, a equipa do projecto é constituída por Anabela Monteiro e Rui Barros (gestor de projecto), tendo como coordenador científico João Pascoal Faria.

O autor participou activamente em todas as fases do projecto, nomeadamente:

1. Análise dos requisitos do projecto
2. Estudo da ferramenta original
3. Estudo e experimentação de tecnologias candidatas
4. Selecção da tecnologia a utilizar na implementação da camada Web
5. Desenho arquitectural e funcional da solução
6. Implementação da solução
7. Elaboração de documentação (o presente relatório)

O autor participou em todas as reuniões mensais com o cliente do projecto (Medidata). Durante o projecto ocorreram também reuniões e sessões de trabalho com fornecedores e especialistas de tecnologias candidatas (Microsoft, Gaiaware e Ruby On Rails) e um período de formação em tecnologia Microsoft .Net.

Apesar da componente académica do projecto terminar com a realização do presente relatório e da apresentação final (Abril de 2008), este prolonga-se até meados de Novembro de 2008.

A Figura 3 apresenta o calendário de actividades do projecto, que é descrito mais detalhadamente no Anexo A.

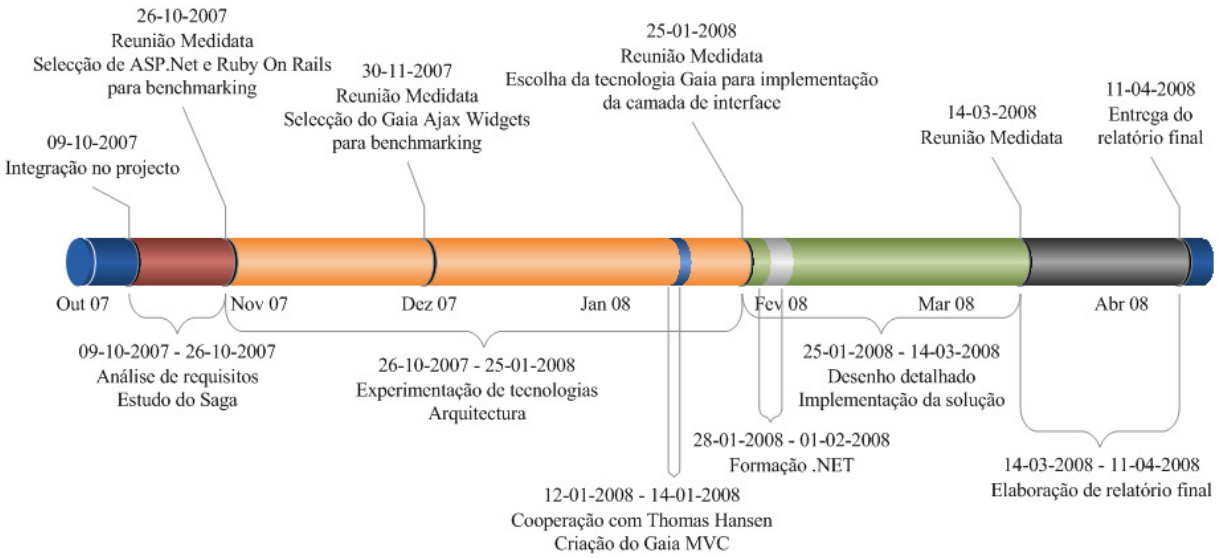


Figura 3 - Calendário do projecto

1.4 Temas abordados no presente Relatório

O presente relatório foi estruturado tendo em conta a sequência das principais actividades realizadas no decurso do projecto. Assim, o 'núcleo' do relatório é constituído por 6 capítulos.

No capítulo 2 é analisado o problema geral que deu origem ao projecto. Nesta fase é apresentada a ferramenta Saga, focando essencialmente a sua origem, funcionalidades, arquitectura e posição no mercado, com o intuito de contextualizar o leitor e introduzir conceitos essenciais à compreensão do projecto. Descrevem-se, de seguida, a motivação e os requisitos da migração para ambiente Web, finalizando o capítulo com a explicação da estratégia de migração utilizada no projecto.

O capítulo 3 diz respeito à análise e experimentação de tecnologias, onde se detalham individualmente as tecnologias consideradas para a implementação da camada de interface, as experiências realizadas com cada uma e as ilações retiradas destas. Para além das tecnologias estudadas pelo autor (ASP.Net, Silverlight e Gaia Ajax Widgets), e para que o relatório sirva também de documentação do projecto, será mencionada uma tecnologia estudada por outro membro da equipa, igualmente considerada no processo de selecção (Ruby On Rails). Por fim, é feita uma análise comparativa, justificando a decisão tomada relativamente à(s) tecnologia(s) a utilizar.

A arquitectura da nova camada Web é apresentada no capítulo 4. Para além de uma visão global da arquitectura, são explicadas as decisões de desenho de software mais relevantes e enumeradas algumas das dificuldades e soluções encontradas.

O capítulo 5 destina-se a demonstrar a estrutura e o funcionamento geral da nova camada de interface Web através de alguns exemplos de utilização do SagaWeb. Estes exemplos servirão de ponto de comparação com a experiência original de utilização do Saga e serão a base da avaliação dos resultados. Neste capítulo são, também, apresentadas as perspectivas de trabalho futuro, que serão uma referência para o trabalho a desenvolver nas seguintes fases do projecto.

O capítulo final destina-se às conclusões retiradas do projecto pelo autor, tanto a nível profissional, como pessoal.

2 Análise do problema

Este capítulo destina-se à análise em detalhe do problema que originou o projecto SagaWeb. Serão descritos a ferramenta original, o Sistema Assistido de Geração e Gestão de Aplicações (SAGA), a sua arquitectura e desenvolvimento e alguns conceitos essenciais à sua compreensão. De seguida, serão apresentadas a motivação do projecto e a estratégia adoptada na sua migração. Os requisitos do projecto são apresentados no final deste capítulo.

2.1 A ferramenta Saga original

O Saga (Figura 4) é uma ferramenta CASE para o desenvolvimento de aplicações interactivas de bases de dados relacionais constituídas essencialmente por formulários de ecrã, relatórios e alguns processamentos. [2]

Desenvolvido inicialmente no INESC Porto no âmbito do projecto 'Autarquias', para apoiar o desenvolvimento do pacote SIGMA – Sistema Integrado de Gestão Municipal – o Saga é presentemente utilizado pela Medidata.

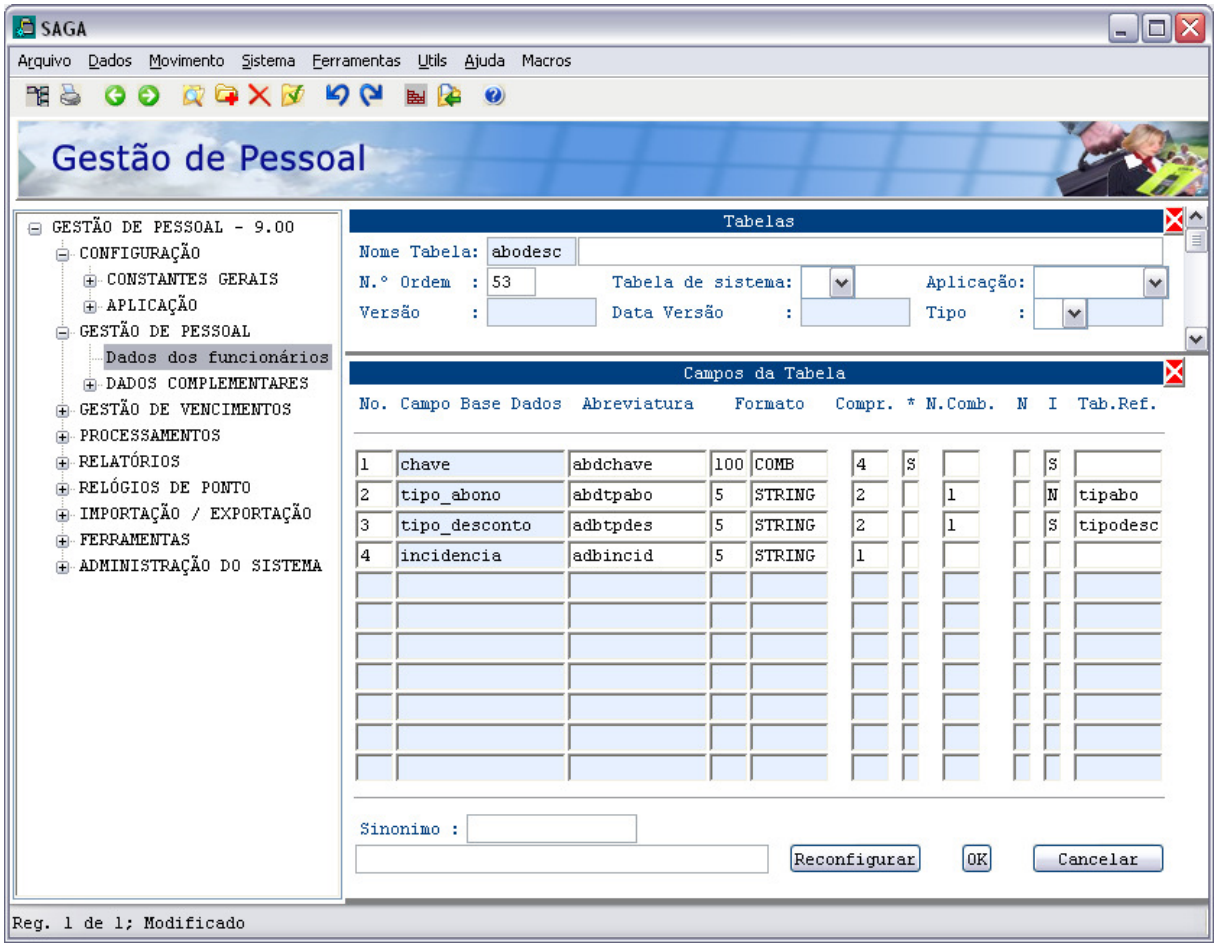


Figura 4 - Exemplo de utilização do Saga

2.1.1 Portabilidade e conectividade

O Saga funciona sobre múltiplas plataformas:

- Vários Sistemas de Gestão de Bases de Dados (SGBD) relacionais: SQL Server, Oracle, Informix e outros via ODBC[3]
- Vários sistemas operativos e ambientes de interacção com o utilizador: Windows (com interface gráfico) ou Unix (com interface alfanumérico baseado em janelas alfanuméricas)
- Diferentes arquitecturas: cliente-servidor ou centralizada

O Saga oferece independência da plataforma:

- Aplicações desenvolvidas sobre uma plataforma (SGBD e/ou sistema operativo) podem ser transportadas para qualquer outra plataforma suportada, sem qualquer alteração das aplicações;
- Diferentes sessões da mesma aplicação, partilhando a mesma base de dados, podem estar a correr simultaneamente em diferentes sistemas de interface para o utilizador (alfanuméricos e gráficos).

A independência das aplicações em relação à plataforma, por intermédio do Saga, é ilustrada na Figura 5.

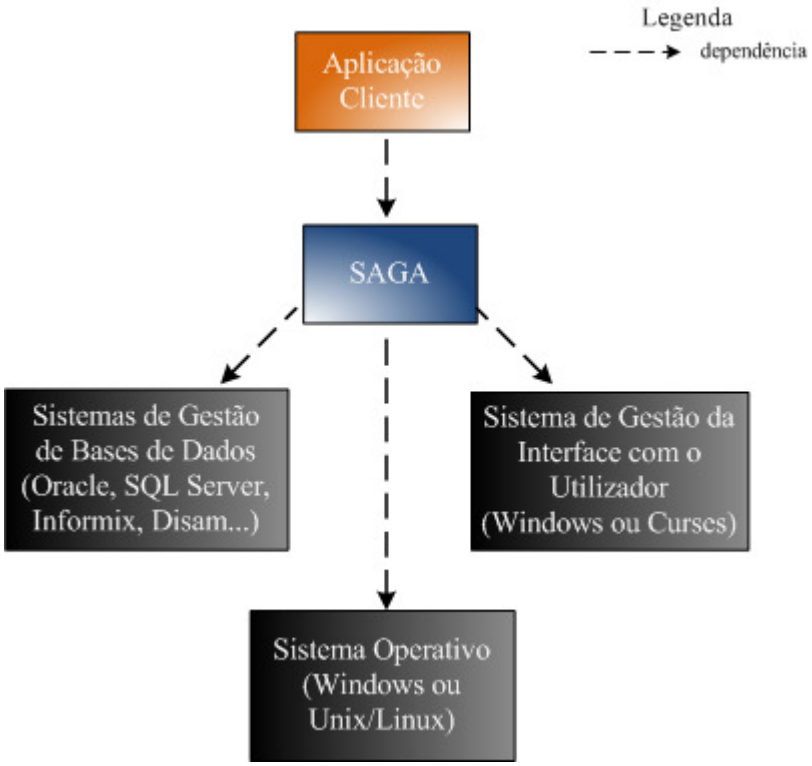


Figura 5 - Arquitectura por camadas do ambiente de trabalho assente no Saga.

[3] ODBC é um acrónimo para Open DataBase Connectivity, uma API (Application Program Interface) standard para acesso a bases de dados relacionais.

2.1.2 Vistas

No Saga, formulários e relatórios são unificados através do conceito de vista[4]. As vistas são os componentes principais das aplicações e são assim designadas porque permitem visualizar, agregar e relacionar de múltiplas formas a informação armazenada na base de dados de forma normalizada.

Destacam-se as seguintes funcionalidades:

- Todas as vistas se podem manipular interactivamente no ecrã (para consulta ou alteração) e imprimir, havendo poucas variações entre a formatação para o ecrã ou para a impressora, segundo o lema WYSIWYG ('What You See Is What You Get');
- Podem ser feitas pesquisas flexíveis por quaisquer campos das vistas, incluindo campos calculados, campos de sumário e campos de sub-vistas sem que, para tal, seja necessária qualquer programação;
- Vistas simples podem ser geradas automaticamente (também chamadas vistas por defeito) e refinadas depois através de um utilitário de desenho de *layout* (paint);
- Através de uma vista, podem-se manipular várias tabelas da base de dados, sem necessidade de qualquer programação, devido à geração automática das condições de junção, critérios de actualização da base de dados e regras de integridade relacional correspondentes por parte do Saga;
- As vistas podem ser 'combinadas' para construir outras mais complexas, com relações mestre-detalhe (Figura 6);
- Uma vista pode estar organizada hierarquicamente, tendo secções dentro de secções, em que cada secção corresponde a um agrupamento de registos;
- Navegação eficiente através de grandes quantidades de dados, devido a mecanismos automáticos de selecção incremental de dados da base de dados.

[4] Apesar de diversas semelhanças, as vistas do Saga não devem ser confundidas com vistas definidas em SQL com o comando "create view".

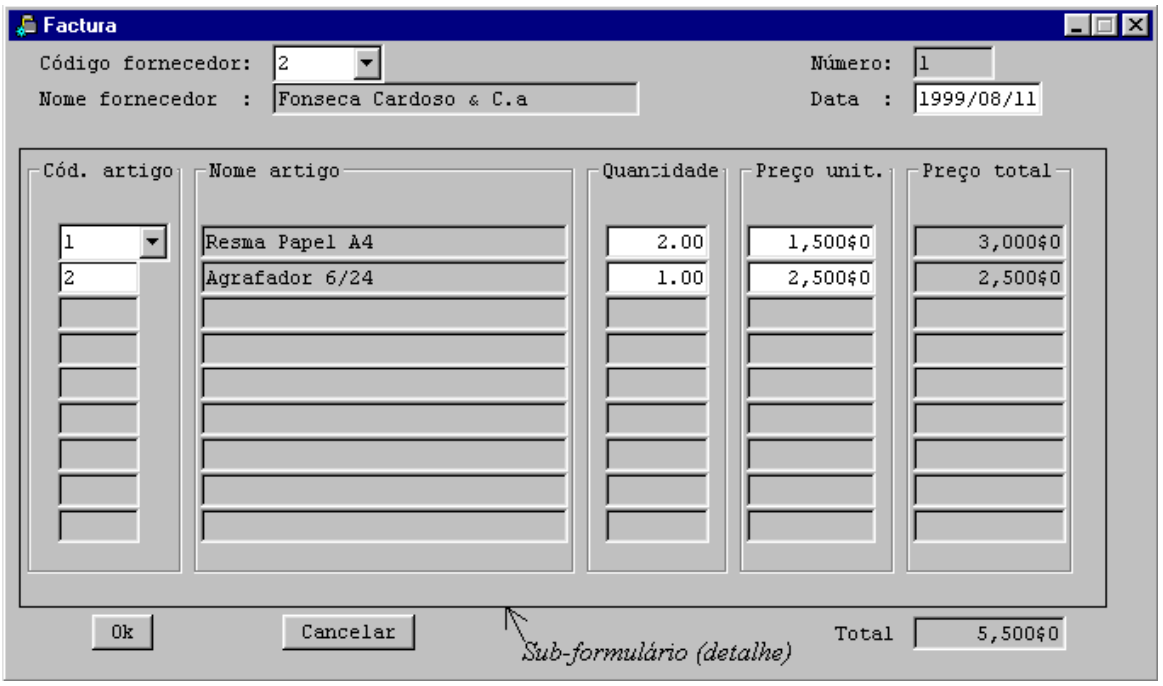


Figura 6 - Vista para introdução de facturas com sub-vista tabular.

2.1.3 Regras

As vistas são refinadas através da adição de regras e comandos, escritas numa linguagem de 4ª geração, a 'linguagem de regras e comandos' (Figura 7). Os comandos definem acções que o utilizador pode invocar explicitamente através de botões ou opções de menus. Em contrapartida, as regras definem acções a executar automaticamente pelo sistema em resposta à ocorrência de eventos de manipulação de vistas (modificação do valor de um campo, inserção de um registo, abertura de uma vista, etc.), servindo para diversas finalidades:

- Manutenção de dados derivados: definição de campos calculados em função de outros campos, registos seleccionados em função de outros registos, etc., segundo um paradigma semelhante ao das folhas de cálculo;
- Verificação de restrições de integridade (validações): definição de restrições envolvendo um único campo, vários campos do mesmo registo ou vários registos;
- Definição de valores iniciais por omissão;
- Conversão e correcção de dados introduzidos pelo utilizador;
- Realização de acções e validações no início ("before") ou no fim ("after") de operações de manipulação de vistas (inserir, actualizar ou eliminar registos, salvar alterações, etc.);
- Controlo de autorizações e obrigações (campos alteráveis, campos obrigatórios, etc.)
- Monitorização e alerta de situações;

- Definição de vistas desligadas da base de dados que não tiram partido dos mecanismos automáticos de selecção e actualização.

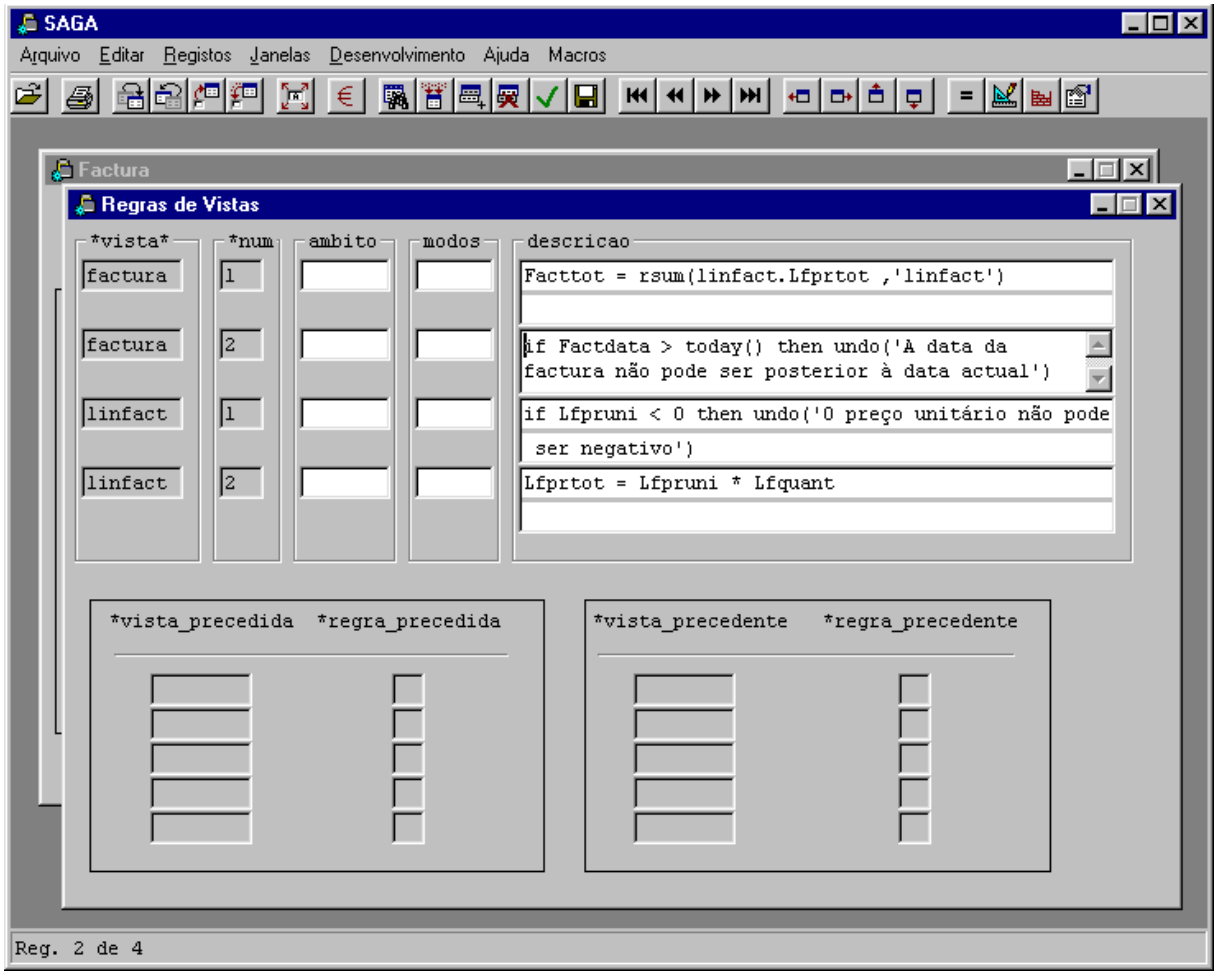


Figura 7 - Definição de regras

A programação através de regras tem várias vantagens: é mais modular, está mais próxima da especificação e está mais adaptada a sistemas interactivos dirigidos por eventos.

2.1.4 Integração de ambientes de desenvolvimento e utilização das aplicações

No Saga, os ambientes de desenvolvimento e de utilização das aplicações encontram-se integrados sob diversos aspectos:

- As especificações das aplicações (tabelas, vistas, regras, menus, utilizadores, permissões, etc.) são armazenadas numa base de dados relacional (em tabelas de sistema), tal como os dados manipulados por essas aplicações[5].

[5] O Saga pode trabalhar directamente sobre as especificações das vistas armazenadas na base de dados (útil em ambiente de desenvolvimento), ou sobre especificações pré-compiladas para ficheiros (útil em ambiente de produção, porque torna a abertura das vistas mais rápida).

- Na mesma sessão de trabalho, pode-se passar uma vista do modo de execução para o modo de desenho e vice-versa, e podem coexistir vistas em modo de desenho e em modo de execução[6].
- Grande parte do trabalho de desenvolvimento de aplicações é efectuada através de vistas criadas com o Saga (chamadas vistas de sistema), assemelhando-se assim ao trabalho de utilização final das aplicações[7]. A título de exemplo, veja-se o que se passa na Figura 7, em que as regras associadas a vistas são introduzidas através da vista de sistema "Regras de Vistas".

A integração tem diversas vantagens:

- O desenvolvimento é mais expedito, pois o ciclo codificação-teste é encurtado;
- Os utilizadores finais ou administradores de sistemas podem realizar pequenas acções de manutenção;
- O desenvolvimento de aplicações beneficia dos serviços fornecidos pelos SGBD relacionais, nomeadamente serviços de interrogação e de controlo de concorrência;
- A ferramenta é mais compacta e, consequentemente, de mais fácil manutenção e evolução.

2.1.5 Arquitectura do Saga

O Saga foi desenvolvido em C, com uma pequena porção de código em C++ (para ligação a algumas *features* do Windows), e faz a ligação ao sistema de janelas através da Win32 API[8]. Conta com cerca de 120.000 linhas de código fonte.

O Saga funciona normalmente numa arquitectura cliente-servidor sobre ODBC, tanto em ambiente de desenvolvimento como em ambiente de utilização final das aplicações, conforme ilustra, de forma simplificada, o diagrama da Figura 8.

---

[6] No entanto, o modo de desenho pode ser vedado aos utilizadores finais através de um sistema de permissões.

[7] As vistas de sistema são fornecidas numa aplicação "master", que deve servir de base a qualquer nova aplicação.

[8] API com a qual todos os programas Windows (excepto os programas de consola) devem interagir, independentemente da linguagem utilizada

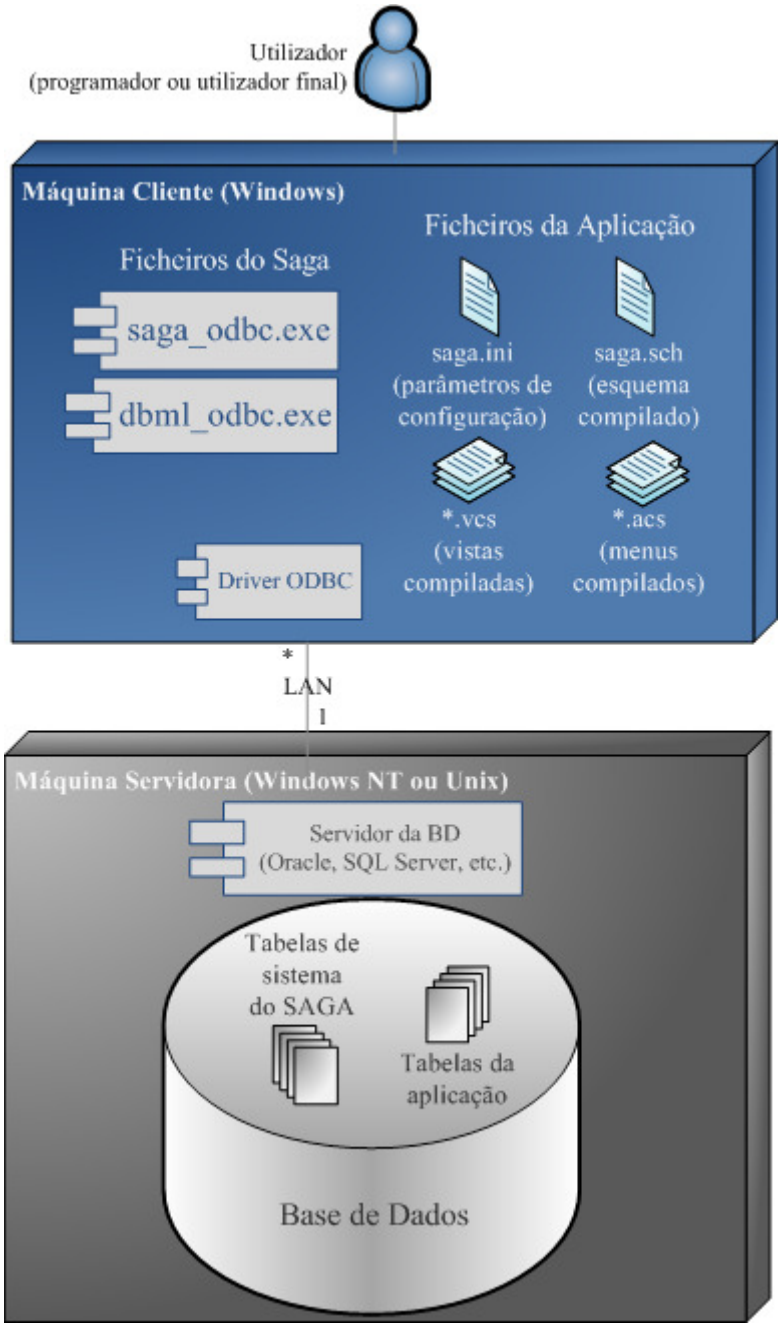


Figura 8 - Arquitectura cliente-servidor do Saga sobre ODBC

Segue-se uma pequena explicação de cada um dos elementos apresentados na figura anterior.

saga_odbc.exe

O executável "saga_odbc.exe" constitui o Saga propriamente dito, com interface gráfica para o utilizador. Corre em cada máquina cliente, podendo estar instalado no disco local de cada máquina cliente ou num disco partilhado por rede.

dbml_odbc.exe

O executável "dbml_odbc.exe" é uma consola (tipo ‘shell’) com interface alfanumérica para o utilizador que executa comandos escritos num subconjunto da linguagem de regras e comandos do Saga, sem as funções de manipulação de vistas (apenas com as funções comuns e as funções de manipulação da base de dados). O acrónimo ‘DBML’ vem de ‘Data Base Manipulation Language’. Este utilitário não se destina aos utilizadores finais, mas apenas aos programadores e administradoras das aplicações. Serve para realizar operações de instalação e administração das aplicações, bem como executar processamentos que não envolvem interacção com o utilizador.

Ficheiro com parâmetros de configuração (saga.ini)

O ficheiro "saga.ini" é um ficheiro de texto (específico de cada aplicação) onde devem ser definidos parâmetros de configuração diversos que são depois lidos e utilizados pelo executável "saga_odbc.exe". Estes parâmetros indicam a localização de ficheiros importantes, a localização da base de dados e de tabelas específicas, preferências da interface para o utilizador, dimensionamento de limites diversos, etc.

Base de Dados, Servidor da Base de Dados e driver ODBC

A base de dados com as tabelas específicas da aplicação e as tabelas de sistema do Saga reside numa máquina servidora (com sistema operativo UNIX ou Window), juntamente com o software que gere a base de dados (servidor da base de dados: Oracle, SQL Server, etc.).

As tabelas de sistema são tabelas utilizadas pelo Saga para guardar, de forma estruturada, definições de:

- Tabelas da aplicação e de sistema
- Vistas da aplicação e de sistema
- Relatórios e impressoras
- Menus da aplicação
- Utilizadores, grupos e permissões
- Barras de menus e barras de ferramentas do próprio Saga.

Devido às limitações de alguns SGBD mais antigos a manipular campos de comprimento variável, pode ser necessário guardar alguma informação em ficheiros de texto separados, geridos especificamente pelo Saga.

O servidor da base de dados é acedido pelo Saga através de ODBC, para o que é necessário que esteja instalado o "driver" apropriado na máquina cliente.

Note-se que a arquitectura apresentada na figura não é a única possível. Nomeadamente, as tabelas da aplicação podem estar distribuídas por várias bases de dados localizadas em máquinas diferentes.

Ficheiro com esquema compilado (saga.sch)

Por razões de eficiência, o Saga utiliza um ficheiro "saga.sch" onde guarda (em formato binário muito compacto e independente da máquina) o esquema "compilado" da base de dados (informação sobre a estrutura das tabelas existentes). Este ficheiro é carregado para memória pelo "saga_odbc.exe".

Vistas compiladas (.vcs)*

Os constituintes principais de qualquer aplicação desenvolvida com o Saga são as tabelas e as vistas (correspondentes a formulários e relatórios). Conforme já foi referido, a especificação das vistas (incluindo o código de regras e comandos associados a essas vistas), é guardada em tabelas de sistema. Quando se abre uma vista, o Saga tem de carregar para memória a especificação dessa vista, podendo fazê-lo acedendo directamente às tabelas de sistema. Em alternativa, para tornar a abertura das vistas mais rápida e diminuir a carga no servidor (note-se que a especificação de uma vista ocupa várias registos em várias tabelas), podem-se compilar previamente as vistas para ficheiros (um ficheiro para cada vista com extensão ".vcs"). Para abrir uma vista, o Saga só tem de carregar para memória o respectivo ficheiro pré-compilado. Esta solução é útil sobretudo em ambiente de produção. As vistas pré-compiladas podem ser instaladas em cada máquina cliente ou num único sítio partilhado por rede.

Menus compilados (.acs)*

A especificação das barras de menus e barras de ferramentas (*toolbars*) é igualmente guardada em tabelas de sistema. Quando se arranca o Saga, este carrega para memória essa especificação, podendo fazê-lo acedendo directamente às tabelas de sistema. Em alternativa, para tornar o arranque do Saga mais rápido e diminuir a carga no servidor, podem-se compilar previamente os menos para ficheiros (com extensão ".acs").

2.2 Objectivo

O objectivo do projecto SagaWeb é o de implementar uma nova camada de apresentação para a ferramenta Saga para a Web, de forma a que as aplicações desenvolvidas com a ferramenta original possam correr no novo ambiente sem rescrita.

2.3 Motivação e estratégia de migração

O Saga é usado presentemente pela Medidata para o desenvolvimento e manutenção de aplicações cliente-servidor com interface Windows para mais de 55% do mercado autárquico em Portugal e um número significativo de autarquias em Angola. Tal como tem acontecido noutros mercados, existe uma procura crescente no mercado autárquico de aplicações empresariais com interface Web, por diversas razões:

- Facilidade de acesso às aplicações a partir de qualquer local (a partir de qualquer edifício da Câmara Municipal, a partir de qualquer junta de freguesia, a partir de casa do funcionário ou dirigente, etc.) e qualquer tipo de terminal;
- Evolução das tecnologias Web (nomeadamente através de Ajax) que permitem ter no *browser* Web uma interactividade e desempenho semelhantes aos obtidos na utilização do ambiente *desktop*;
- Potencial de integração com outras aplicações e conteúdos em portais e intranets;
- *Deployment* simplificado das aplicações (sem qualquer instalação de software nas máquinas clientes).

Do ponto de vista da Medidata, a evolução tecnológica das suas aplicações é também urgente para não perder competitividade face à concorrência.

A solução escolhida passou por implementar uma nova camada de interface Web na ferramenta Saga, uma vez que as suas aplicações são definidas de uma forma declarativa e independente da tecnologia da UI e o Saga possui uma arquitectura modular na qual os componentes que gerem a UI estão relativamente bem isolados. Esta solução permite a migração das aplicações para o novo ambiente sem que seja necessário rescrevê-las.

Soluções alternativas de rescrita das aplicações numa nova plataforma, desenvolvimento de um conversor para uma nova plataforma ou desenvolvimento de um novo *runtime* seriam demasiado onerosas e arriscadas, devido à grande complexidade e maturidade quer das aplicações quer da ferramenta Saga. O Saga é um sistema legado, antigo (surgiu no final da década de 80) e complexo mas que, no entanto, continua a fornecer serviços críticos.

Tendo em conta que o principal requisito da migração da ferramenta era o de garantir a compatibilidade das aplicações (secção 2.4), a opção pela evolução da ferramenta, implementando apenas uma nova camada de apresentação, minimiza o risco de incompatibilidade e protege o grande investimento feito pelos clientes nas suas aplicações.

Para além de ser uma ferramenta cujo código é muito extenso (cerca de 120 mil linhas), o Saga possui um elevado nível de complexidade. O seu desenvolvimento envolveu, entre outras, a criação de raiz de linguagens específicas de domínio (linguagem de regras e comandos e meta-modelo para definição de vistas), e a implementação de automatismos de gestão de regras e de vistas (já referidos). Não existem razões para reimplementar estas funcionalidades, visto continuarem a servir o seu propósito de uma forma plenamente satisfatória e o investimento nas mesmas ter sido elevado.

Tal como acontece frequentemente com outros sistemas legados, a reduzida disponibilidade de documentação e pessoas com conhecimento sobre a organização interna do Saga, a sua elevada complexidade, e a tecnologia antiquada em que está desenvolvido (linguagem C), são factores que dificultam um projecto de reengenharia desta natureza, em que há necessidade de aceder e modificar algumas partes internas da ferramenta.

2.4 Requisitos da migração para ambiente Web

O conjunto de requisitos do projecto SagaWeb divide-se em 3 grupos, de acordo com o seu grau de prioridade: muito alta, alta e média.

- Compatibilidade (muito alta)

O requisito de prioridade mais alta é o de garantir que as aplicações já existentes funcionem com o mínimo esforço de migração (alteração das aplicações) possível na nova versão Web. Estão previstas algumas alterações, nomeadamente ao nível de regras relacionadas com a interface para o utilizador, como por exemplo regras que requeiram *input* do utilizador, ou que mudem o foco de um campo para outro.

- Portabilidade (alta)

Ao nível do utilizador final das aplicações, é importante que as aplicações corram nos Web *browsers* mais utilizados (Internet Explorer, Mozilla Firefox, …). O novo ambiente deve ser independente do sistema operativo da máquina cliente. A utilização de standards como HTML, Javascript e AJAX permitirão responder a este requisito. Ainda que com menor prioridade, é desejável que o *software* do lado do servidor possa correr nos sistemas operativos mais utilizados (Windows e Linux).

- *Web Design*, Usabilidade e Acessibilidade (alta)

Durante o desenvolvimento, deve procurar-se a evolução das aplicações ao nível de *Web Design*, no sentido de as tornar mais competitivas em termos da experiência de utilizador, tirando partido do novo ambiente (Web). O novo ambiente deverá manter a interactividade do seu antecessor. É desejável que o tratamento de alguma de interacção com o utilizador (validações, etc.) seja feito do lado do cliente, o que irá exigir a tradução de um conjunto de regras para Javascript. Apesar de, actualmente, a conformidade com as normas W3C de acessibilidade não ser, ainda, obrigatória no sector público, a curto/médio prazo esta obrigatoriedade vai-se verificar, pelo que é importante garantir a acessibilidade das aplicações do SagaWeb.

- Desempenho (alta)

O desempenho das aplicações no ambiente Web deve ser semelhante ao verificado no ambiente antecessor, permitindo que os utilizadores das aplicações tenham uma transição sem problemas para o novo ambiente.

- Segurança (média)

As aplicações Saga funcionam em Intranet, pelo que a segurança é um requisito de média prioridade.

- "Componentarização" e integração (média)

Deseja-se que seja possível transformar aplicações em componentes de interface para o utilizador, para que vistas individuais possam ser inseridas num ambiente do tipo portal, podendo estar associadas a passos de *workflow* de orquestração de processos do utilizador e do negócio.

- Ambiente de desenvolvimento adequado (média)

Manter ou melhorar a rapidez e facilidade de desenvolvimento e manutenção de aplicações na nova versão. Uma das possibilidades será desenvolver aplicações num ambiente aberto de desenvolvimento (como um *Integrated Development Environment* - IDE) no qual as vistas do Saga fossem componentes disponibilizados numa *toolbox*. Outra possibilidade é migrar a actual ferramenta de desenho de *layout* (paint) para ambiente Web (webpaint).

3 Análise, experimentação e selecção de tecnologias para a Interface Web

O SagaWeb é um projecto que consiste no desenvolvimento de uma camada de interface Web para aplicações de formulários. A camada deve ser capaz de interpretar a definição da apresentação e estrutura das vistas que constituem as aplicações definidas com o Saga e representá-las em ambiente Web. A definição das vistas encontra-se em ficheiros de vistas compiladas do Saga, e é feita recorrendo a uma linguagem específica de domínio. A gestão de interface passa do Saga para a nova camada. No entanto, a gestão da camada de 'modelo' (dados) permanece responsabilidade do gestor de vistas do Saga.

Assim, tornou-se importante determinar qual a tecnologia Web de interfaces para aplicações de formulários a utilizar no desenvolvimento desta nova camada. Para este efeito, a equipa do projecto realizou um *benchmarking* de tecnologias, constituído por um estudo e o desenvolvimento de uma solução sobre cada uma delas. O benchmarking teve como objectivo avaliar as características das tecnologias estudadas e a sua capacidade de resposta a um conjunto de critérios, de entre os quais se destacam os seguintes, constituídos por requisitos de projecto e características tecnológicas:

- Compatibilidade das aplicações existentes com o novo ambiente;
- Portabilidade das aplicações e da ferramenta;
- Web Design e Usabilidade das aplicações;
- Desempenho das aplicações;
- Suporte ao desenvolvimento com a tecnologia;
- Rapidez, robustez e facilidade de desenvolvimento da tecnologia.

O resultado do benchmarking é descrito em detalhe na secção 3.5.

Este capítulo consiste na descrição do *benchmarking* de tecnologias Web candidatas ao desenvolvimento da nova camada de interface de aplicações baseadas em formulário, realizada pela equipa do SagaWeb. Este processo culminou na determinação da tecnologia que soluciona o problema descrito anteriormente da forma mais satisfatória.

3.1 ASP.Net

Esta secção destina-se à análise e descrição das experiências realizadas com a tecnologia pertencente à framework .Net da Microsoft, o ASP.Net. Esta foi a primeira tecnologia estudada e experimentada pelo autor, tendo o desenvolvimento sido realizado com o recurso à linguagem de programação C#.

Devido ao facto de se terem analisado e experimentado outras tecnologias (Gaia Ajax Widgets, secção 3.2) que tinham por base o ASP.Net, é importante apresentar alguns conceitos essenciais para a compreensão, não só das próprias tecnologias, mas também dos desafios arquitecturais e de implementação que advêm da sua utilização.

Assim, nesta secção será feita uma introdução ao protocolo HyperText Transfer Protocol (HTTP) e explicada a dificuldade de manutenção do estado associada ao mesmo. De seguida será apresentado o ASP.Net, as suas páginas, os ficheiros *aspx*, e os User Controls, os

ficheiros *ascx*. Será dada uma explicação sobre a diferença entre os modelos de programação utilizados na construção de aplicações com controlos dinâmicos e com controlos estáticos. Por fim, será descrita a experiência efectuada com a tecnologia ASP.Net.

3.1.1 Introdução ao HTTP

A utilização do protocolo HTTP (HyperText Transfer Protocol) transforma a construção de aplicações Web num desafio interessante. O principal problema associado à construção deste tipo de aplicações resume-se à manutenção de estado entre pedidos. Quando construímos uma aplicação do tipo 'Windows Forms', conseguimos manter facilmente todos os dados necessários ao correcto funcionamento da aplicação. Contudo, tal não acontece nas aplicações Web. Do ponto de vista do servidor, cada pedido de um recurso é um novo pedido, independentemente do facto de ser ou não o primeiro acesso a esse recurso por parte do utilizador.

O protocolo HTTP efectua uma distinção entre os dois intervenientes na comunicação: um desempenha o papel do cliente e o outro do servidor. Uma sessão Web será sempre iniciada por um *browser* Web no computador cliente que efectua um pedido à aplicação/*site* Web que desempenha o papel de servidor. Todas as ligações entre cliente e servidor são mantidas apenas durante o tempo necessário à satisfação do pedido do cliente.

O protocolo HTTP apenas define as regras relacionadas com a comunicação entre os dois sistemas, não sendo por isso responsável por lidar com detalhes de carácter mais específico (como, por exemplo, o formato dos pacotes utilizados para enviar uma mensagem do cliente para o servidor).

3.1.2 Introdução ao ASP. Net

A plataforma ASP.Net introduz uma camada de abstracção que nos permite trabalhar com os valores obtidos através dos pedidos HTTP no lado do servidor. Uma das vantagens decorrentes da utilização desta plataforma reside no facto de esta conseguir transformar eventos cliente (gerados no *browser*) em eventos servidor que podem ser tratados através da código escrito numa linguagem .Net (tudo isto de uma forma quase transparente para o programador). [3]

A geração de eventos servidor (como consequência de eventos ocorridos no lado do cliente) é conseguida através da utilização de formulários e controlos servidor (elementos anotados com o atributo *runat='server'*) que são capazes de, automaticamente, iniciarem um novo pedido a partir do cliente para a própria página do lado do servidor. Estas operações são designadas de *postbacks*. Para que este tipo de operações funcione correctamente, é necessário garantir que o estado dos controlos é mantido e/ou actualizado no lado do servidor entre pedidos (algo que, como seria de esperar, é feito automaticamente pela plataforma).

Durante o *postback*, a plataforma começa por recuperar o estado associado a cada controlo guardado no final do pedido anterior. Em seguida, actualiza os controlos de servidor para que os valores modificados no lado do cliente sejam replicados no lado do servidor. De seguida, a plataforma gera os eventos servidor adequados à situação actual (normalmente, tem

de gerar pelo menos o evento servidor equivalente ao evento cliente que iniciou toda a operação de *postback*) (Figura 9).

Figura 9 - Modelo usado pelos formulários Web associados a uma página ASP. Net

3.1.3 Páginas ASP.Net (*.aspx)

Uma página ASP.Net (ficheiro *.aspx*) é, sem qualquer dúvida, o tipo de recurso mais utilizado durante o desenvolvimento de uma aplicação Web .Net. No Anexo B pode ser visto um exemplo de uma página *aspx*. Estas páginas contêm vários tipos de blocos.

- Directivas

Permitem controlar vários aspectos associados à compilação de uma página *aspx*. Apesar de poderem ser colocadas em qualquer parte da página, é normal definirmos este tipo de elementos no topo. A directiva mais importante é a directiva `@Page`, que apenas pode ser usada em páginas ASP.Net, e define vários elementos importantes. Alguns desses elementos são o nome da classe em que a página *aspx* é transformada antes de ser compilada (atributo 'ClassName'), a classe base da página *aspx* (atributo 'Inherits') que, por omissão, é System.Web.UI.Page, a linguagem de programação utilizada na página (atributo 'Language') e o ficheiro de Code-Behind ('Inherits' e 'CodeFile').

- Secções de código

Uma declaração de código pode consistir na definição de um método ou na escrita de uma instrução simples. Todas as funções devem ser declaradas no interior do elemento `<script>`, conforme é ilustrado no excerto seguinte:

```
<script runat="server">
  void MyMethod() {
    //....
  }
</script>
```

Na maior parte das vezes, as secções de código definem métodos associados a eventos gerados pelos controlos e pelas páginas (note-se que a utilização de ficheiros de *code-behind* para escrever código deste tipo é recomendada). Todos os métodos declarados no interior das *tags* `<script>` são transformados em métodos associados à classe obtida a partir do *parsing* da página *aspx*.

- Controlos Servidor, *tags* HTML e texto

O código que define o apresentação das páginas é normalmente constituído por *tags* relativas a controlos ASP.Net e/ou elementos HTML tradicionais. A principal característica 'visual' de todos os controlos servidor reside na utilização do atributo 'runat' com o valor 'server'.

Todos os controlos HTML podem ser transformados em HTML Server Controls se forem anotados com o atributo `runat="server"`. O modelo de objectos definido por estes controlos tem como principal objectivo manter o modelo definido pelos elementos HTML correspondentes. (Figura 10)

Os Web Controls possuem um modelo de objectos muito mais rico do que o fornecido pelo outro subconjunto de controlos servidor existente, os HTML Server Controls. O nome Web Controls foi usado para denominar estes controlos devido ao facto de todos os controlos derivarem (directa ou indirectamente) da classe *WebControl*. Estes controlos protegem o programador de aplicações Web de ter de saber todos os pormenores relativos aos controlos HTML, uma vez que apresentam um modelo de objectos mais intuitivo e coerente. (Figura 10)

```
<!-- HTML Server Control -->
<input id="serverInput" value="htmlServerControl" runat="server" />

<!-- WebControl -->
<asp:TextBox ID="tbox" value="aspNetControl" runat="server"></asp:TextBox>
```

Figura 10 - Declaração de HTML Server Control e de WebControl

Os exemplos anteriores misturam todos os tipos de elementos que podemos encontrar numa página. Esta situação não é adequada à construção de aplicações profissionais. O ideal é separar o código da definição de *layout* da página.

O ASP.Net introduziu o conceito de *code-behind*. A sua utilização permite-nos definir todo o código necessário ao funcionamento de uma página num ficheiro à parte que será associado à página *aspx*. A utilização desta estratégia resulta, do ponto de vista prático, na criação de dois ficheiros: um de extensão '*.aspx'*, que contém as *tags* e controlos que definem o *layout* da página; e outro de extensão '*.aspx.cs'*, que contém o código utilizado pela página.

3.1.4 User Controls (*.ascx)

Os User Controls (ficheiros *ascx*) permitem efectuar a rápida construção de controlos que podem ser reutilizados ao longo de uma aplicação Web. Esta secção apresenta as particularidades associadas à construção deste tipo de elementos.

A fácil construção de componentes e controlos reutilizáveis foi uma das metas que a equipa do ASP.Net se propôs a atingir quando começou o seu desenvolvimento. Actualmente, a plataforma permite a construção de novos controlos, através de uma de duas estratégias: controlos servidor personalizados e User Controls.

Um controlo servidor personalizado não é mais do que uma classe que herda da classe *Control* ou *WebControl* (directa ou indirectamente). Portanto, nestas situações, somos obrigados a usar código para definir o aspecto final do controlo (normalmente à custa da geração de HTML no método *render* – ou num dos métodos relacionados).

Por outro lado, um User Control é, em última análise, um controlo baseado na classe *UserControl* e a sua construção é normalmente feita através do *designer* existente no Visual Studio. Na prática, um User Control será constituído por um ou mais controlos definidos de forma declarativa num ficheiro de extensão *ascx*.

A principal justificação para a utilização deste tipo de controlos reside na rápida construção de um novo controlo (constituído geralmente por outros controlos servidor e/ou instruções HTML) que pode ser reutilizado ao longo das várias páginas de uma aplicação.

3.1.5 Controlos estáticos e dinâmicos

A definição de controlos num formulário realiza-se, geralmente, através da utilização de uma sintaxe de *markup*, criando elementos com o atributo runat=’server’ em páginas '*aspx* ou *ascx* (User Controls). A estes controlos dá-se o nome de controlos estáticos, por já se encontrarem definidos antes do início da execução da página na qual se incluem (Figura 11). [4]

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="MyPage.aspx.cs"
Inherits="Infinity.Examples.MyPage" %>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>TRULY Understanding Dynamic Controls</title>
</head>
<body>
    <form id="form1" runat="server">
        Your name:
        <asp:TextBox ID="txtName" runat="server" Text="Enter your name" />
    </form>
</body>
</html>
```

Figura 11 - Declaração de um controlo TextBox numa página *aspx*

Quando a plataforma ASP.Net faz o *parse* do *markup*, cria uma classe ‘na hora’, responsável pelo trabalho ‘pesado’ (um a um, os controlos são criados e adicionados à arvore de controlos). A TextBox declarada na Figura 12 resulta no seguinte código gerado automaticamente:

```
private global::System.Web.UI.WebControls.TextBox @__BuildControltxtName()
{
    global::System.Web.UI.WebControls.TextBox @__ctrl;

    #line 11 "c:\projects\Truly\MyPage.aspx"
    @__ctrl = new global::System.Web.UI.WebControls.TextBox();

    #line default
    #line hidden
    this.txtName = @__ctrl;
    @__ctrl.ApplyStyleSheetSkin(this);

    #line 11 "c:\projects\Truly\MyPage.aspx"
    @__ctrl.ID = "txtName";

    #line default
    #line hidden

    #line 11 "c:\projects\Truly\MyPage.aspx"
    @__ctrl.Text = "Enter your name";

    #line default
    #line hidden
    return @__ctrl;
}
```

Figura 12 - Classe gerada automaticamente pela plataforma .Net

Apesar do código parecer um pouco complexo esta é apenas uma maneira cuidadosa de criar uma TextBox, definindo as suas propriedades 'ID' e 'Text', e devolvê-la. As expressões *#line* servem para o compilador alertar o programador de erros de compilação e conseguir dar a linha correcta do ficheiro *aspx* (ou *ascx*), em vez do número da linha do código gerado.

Existe um método do tipo 'construção de controlo' por cada controlo declarado estaticamente na página. Se um controlo existe dentro de outro, o método de 'construção de controlo' do controlo pai adiciona o filho à sua própria colecção de controlos. No exemplo da Figura 13, o controlo do tipo 'form' contém 3 controlos 'filho'.

```
private global::System.Web.UI.HtmlControls.HtmlForm @__BuildControlform1() {
    global::System.Web.UI.HtmlControls.HtmlForm @__ctrl;

    #line 10 "c:\projects\Truly\MyPage.aspx"
    @__ctrl = new global::System.Web.UI.HtmlControls.HtmlForm();

    #line default
    #line hidden
    this.form1 = @__ctrl;

    #line 10 "c:\projects\Truly\MyPage.aspx"
    @__ctrl.ID = "form1";

    #line default
    #line hidden
    System.Web.UI.IParserAccessor @__parser =
        ((System.Web.UI.IParserAccessor)(@__ctrl));

    #line 10 "c:\projects\Truly\MyPage.aspx"
    @__parser.AddParsedSubObject(
        new System.Web.UI.LiteralControl("\r\n   Your name: "));

    #line default
    #line hidden
    global::System.Web.UI.WebControls.TextBox @__ctrl1;

    #line 10 "c:\projects\Truly\MyPage.aspx"
    @__ctrl1 = this.@__BuildControltxtName();

    #line default
    #line hidden

    #line 10 "c:\projects\Truly\MyPage.aspx"
    @__parser.AddParsedSubObject(@__ctrl1);

    #line default
    #line hidden

    #line 10 "c:\projects\Truly\MyPage.aspx"
    @__parser.AddParsedSubObject(
        new System.Web.UI.LiteralControl("\r\n    "));

    #line default
    #line hidden
    return @__ctrl;
}
```

Figura 13 - Método de 'construção de controlo'

Os controlos 'filho' são:

1. Um LiteralControl com o texto "Your name:"
2. Uma TextBox com o ID "txtName"
3. Outro LiteralControl com o *whitespace* que se encontra antes do fim da *tag* 'form'

Para criar o controlo TextBox, o método da Figura 14 chama o método de 'construção de controlo' desse controlo.

```
global::System.Web.UI.WebControls.TextBox @__ctrl1;
#line 10 "c:\projects\Truly\MyPage.aspx"
@__ctrl1 = this.@__BuildControltxtName.@__BuildControltxtName();
```

Figura 14 - Chamada ao método de 'construção de controlo'

É, também, chamado um método que apenas adiciona o controlo à árvore de controlos (Figura 15).

```
@__parser.AddParsedSubObject(@__ctrl1);
```

Figura 15 - Inserção de controlo na árvore de controlos

É assim que a framework adiciona controlos declarados à árvore de controlos. Um controlo dinâmico é um controlo que o programador cria num ponto do ciclo de vida da página, adicionando-o programaticamente à mesma (Figura 16).

```
protected override void OnInit(EventArgs e) {
    TextBox txtName = new TextBox();
    txtName.ID = "txtName";
    txtName.Text = "Enter your name";
    this.form1.Controls.Add(txtName);

    base.OnInit(e);
}
```

Figura 16 - Criação de controlo dinâmico e adição à árvore de controlos da página .Net

Apesar de este código não ser tão complexo como o gerado pelo *parser* de páginas, faz precisamente a mesma coisa!

Isto significa que as mesmas funcionalidades podem ser alcançadas criando os controlos dinamicamente ou estaticamente, sendo que cada abordagem tem as suas vantagens e desvantagens, dependendo da utilização que se lhes pretende dar.

No entanto, ao utilizar controlos estáticos na construção de aplicações Web, o programador herda da plataforma .Net a responsabilidade de manter a(s) árvore(s) de controlos. Em ambas as abordagens (dinâmica e estática) os controlos da árvore devem ser completamente reconstruídos a cada pedido efectuado à página. Um erro comum cometido pelos programadores é o de adicionar um controlo dinâmico à árvore apenas quando se responde a um evento. Por exemplo, digamos que existe um controlo *Button* num formulário com o texto "clique aqui para inserir nome", e que o objectivo do botão, quando clicado, é o de adicionar dinamicamente uma *Textbox* ao formulário para inserção do nome. O código que processa o evento de clique do botão é algo de semelhante ao representado na Figura 17.

```
private void cmdEnterName_Click(object sender, EventArgs e) {
    TextBox txtFirstName = new TextBox();
    this.Controls.Add(txtFirstName);
}
```

Figura 17 - Método incorrecto de adição de controlo à árvore de controlos

Se o código da Figura 17 for executado, pode constatar-se que a TextBox, de facto, é acrescentada à UI. No entanto, a não ser que o utilizador clique no mesmo botão de novo, o

próximo *postback* faz com que a TextBox deixe de existir. Muitos programadores, quando confrontados com este tipo de erros, atribuem a culpa ao ViewState (erradamente).

O ViewState é responsável por manter o estado dos controlos presentes na árvore de controlos, mas não o estado da árvore em si. Essa responsabilidade, tal como referido, pertence ao programador. Os controlos dinâmicos não existirão no próximo *postback*, a não ser que sejam adicionados de novo à árvore de controlos. A partir do momento em que o controlo se encontra de novo na árvore de controlos, a framework trata de manter o seu estado.

3.1.6 Experimentação

A tecnologia ASP.Net, da Microsoft, foi a primeira a ser experimentada pelo autor. A primeira experiência consistia no desenvolvimento de uma aplicação que fosse capaz de permitir a um utilizador navegar ao longo dos registos de uma vista, com a possibilidade de editar, inserir e eliminar qualquer um deles. Os registos deviam ser apresentados num formato de 'grelha' (tabular, com múltiplos registos) ou de 'detalhe' (visualizar registos um a um).

Devido ao facto desta experimentação ter sido interrompida, os dois modos de visualização encontram-se em projectos diferentes. Como o modo de implementação foi semelhante nos 2 projectos, frequentemente será descrito apenas um, o da apresentação no modo 'detalhe'. Ocasionalmente, e se for considerado relevante algum aspecto do desenvolvimento do modo 'grelha', será mencionado esse projecto.

Nesta primeira solução não existe interacção com o Saga, pois tanto a definição como os dados das vistas se encontram no sistema de ficheiros. As definições das vistas encontram-se nos ficheiros de Vistas Compiladas do Saga (**.vcs*), enquanto que os dados de cada vista se encontram num ficheiro sem extensão com o nome igual ao da vista a que pertencem (por exemplo, à vista 'cpostal' correspondem o ficheiro de definição '*cpostal.vcs*' e o ficheiro de dados '*cpostal*'). A arquitectura desta primeira experiência encontra-se representada na Figura 18.

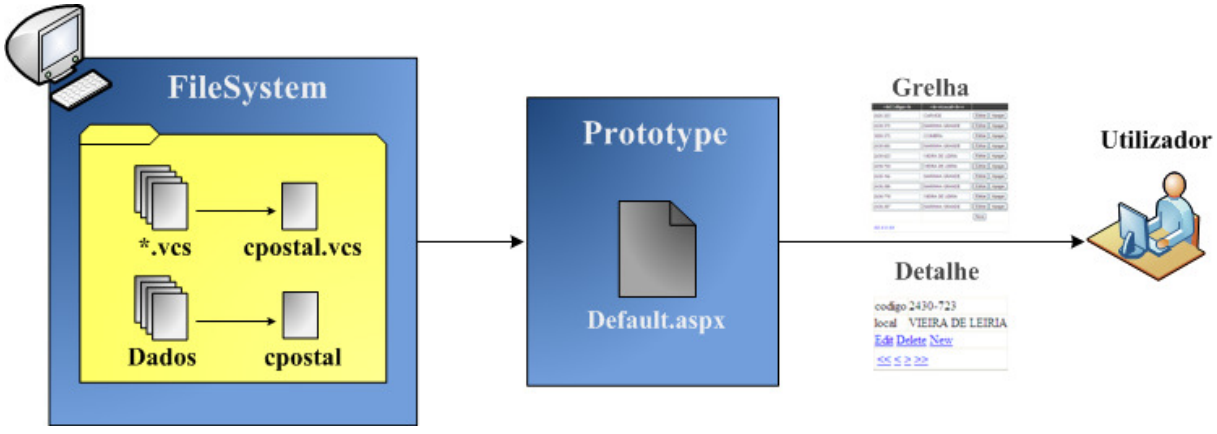


Figura 18 - Arquitectura geral da primeira experiência com ASP.Net

No caso da visualização no modo 'grelha', foram definidas 2 páginas: Default.aspx e Views.aspx. A página Default.aspx é responsável por procurar ficheiros com extensão '.vcs' (vistas compiladas do Saga) numa pasta do sistema de ficheiros pré-definida e apresentar ao utilizador uma lista com o nome das vistas cujos ficheiros de definição havia encontrado. Após o utilizador seleccionar qual das vistas deseja abrir, a aplicação inicia o processo de construção da interface Web da mesma.

A única diferença, neste ponto, entre o modo 'grelha' e o descrito ('detalhe') prende-se com o facto de, no último, ter sido criada apenas a página *Default.aspx*, que abre uma vista cujo nome se encontra pré-definida (*hard-coded*), sem antes apresentar a lista de vistas disponíveis para abertura.

Assim, na página inicial, *Default.aspx*, existe apenas um controlo do tipo *ObjectDataSource* (Figura 19). Um controlo *ObjectDataSource* é um objecto que constitui uma camada intermédia entre as camadas de apresentação e de dados, e que possui capacidades de apresentação e actualização de dados. O controlo *ObjectDataSource* funciona como interface de dados para controlos de apresentação e manipulação de dados, como o *GridView* ou o *DetailsView*. Este tipo de objectos pode ser utilizado para apresentar e editar dados de um objecto definido pelo utilizador numa página Web ASP.Net. [5]

```
<form id="form1" runat="server">
    <asp:ObjectDataSource ID="ds" runat="server" <!-- ObjectDataSource  -->
        DeleteMethod="removeRow" // método responsável pela eliminação de um registo
        SelectMethod="getDataTable" // método de selecção de dados
        TypeName="Prototype.classes.DadosRepository"> // objecto onde se encontram os dados

        // parâmetro obrigatório do método de eliminação de registo (indice de linha)
        <DeleteParameters><asp:Parameter Name="rowIndex" Type="Int32" />
        </DeleteParameters>
    </asp:ObjectDataSource>
</form>
```

Figura 19 - Código de Default.aspx de visualização de vista no formato ' detalhe'

No arranque da página, é instanciado um controlo do tipo DetailsView (para que possa vir a ser adicionado à página mais tarde), e definidas algumas propriedades do mesmo (activar paginação, por exemplo). É necessário subscrever alguns eventos, de modo a poder definir o comportamento de resposta da aplicação a acções do utilizador efectuadas na interface Web (Figura 20).

```
private void createDetailsView()
{
    dView = new DetailsView(); // Inicializar controlo DetailsView
      ...
      ...
    dView.AllowPaging = true; // DetailsView com paginação activada

    // Subscrição de eventos
    dView.ModeChanging += new DetailsViewModeEventHandler(changingMode);
    dView.PageIndexChanging += new DetailsViewPageEventHandler(this.pageIndexChanged);
      ...
      ...
}
```

Figura 20 - Código de definição do DetailsView com adição de *event handlers*

O passo seguinte na execução é criar um objecto do tipo *DataTable*, que guardará os dados da vista e que será utilizado pelo *ObjectDataSource* para os manipular. As colunas da *DataTable* serão preenchidas com os nomes dos campos da vista, enquanto as suas linhas são preenchidas com os dados da mesma.

De seguida, associa-se ao *DetailsView* criado o *ObjectDataSource* declarado em *Default.aspx*, que tem o ID "ds", através do comando:

```
dView.DataSourceID = "ds";
```

Os pedidos de manipulação de dados originam eventos, cujos *handlers* alteram os dados da *DataTable*. Após cada alteração dos dados da *DataTable*, e tal como sucede no fim do primeiro carregamento da página, é chamado o método *DataBind*, que faz com que o *DetailsView* actualize os seus dados, ao sincronizá-los com os da *DataTable*.

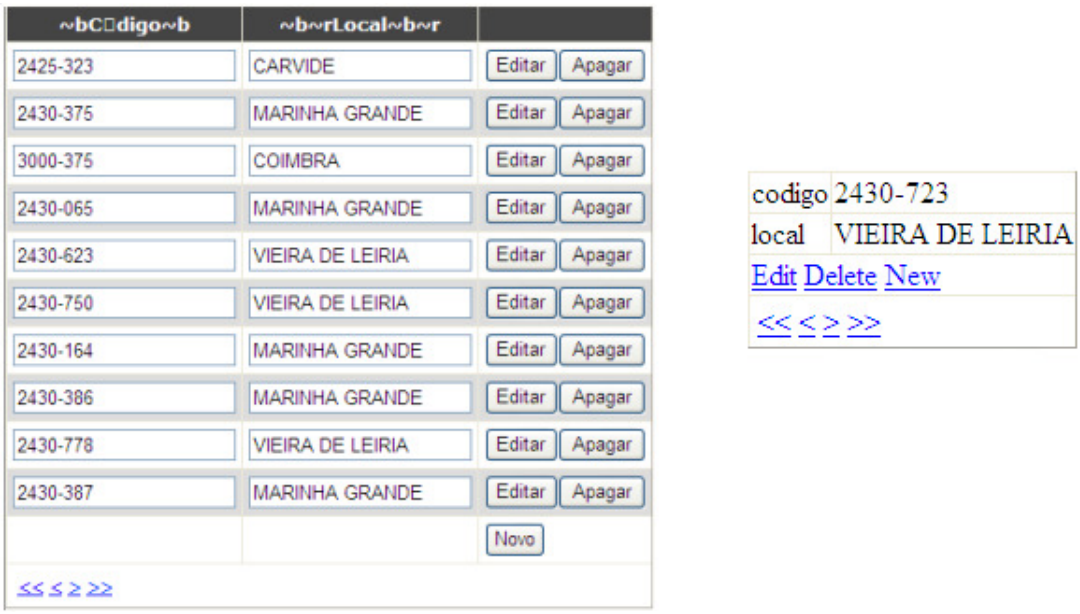


Figura 21 - Formatos de apresentação da vista *cpostal*: 'grelha' e 'detalhe'

A Figura 21 apresenta os resultados obtidos com as 2 soluções desenvolvidas com ASP.Net. O primeiro representa a aplicação que abre uma vista (*cpostal*) no formato 'grelha' (vários registos em simultâneo) e a segunda no modo 'detalhe' (um registo de cada vez).

3.1.7 Conclusões

Apesar de o ASP.Net possuir algumas características interessantes para o desenvolvimento rápido de aplicações baseadas em formulários como, por exemplo, controlos *data-bound* e a noção de *DataSource*, os seus controlos revelaram-se pouco flexíveis em termos de estrutura de apresentação (GridView não suporta posicionamento absoluto de controlos, por exemplo) e de tratamento de eventos (obrigatoriedade de construir a estrutura da árvore de controlos a cada pedido), perdendo também para outras tecnologias, em especial o Ruby On Rails, por este possuir maior facilidade de desenvolvimento e utilizar o padrão MVC. As conclusões retiradas da experimentação de ASP.Net e a análise comparativa desta com as restantes tecnologias estão descritas em maior detalhe na secção 3.5.

3.2 Gaia Ajax Widgets

Esta secção destina-se à descrição da framework Ajax com o nome Gaia Ajax Widgets. Esta framework foi desenvolvida por uma empresa Norueguesa, a Gaiaware, anteriormente conhecida por Frost Innovation. A equipa do projecto tomou conhecimento da existência desta ferramenta por intermédio de José António Silva, da Microsoft Portugal, no decurso da reunião que teve lugar a 30 de Novembro de 2007.

Esta tecnologia foi escolhida para estudo por apresentar características interessantes para o desenvolvimento da nova camada de interface Web, como abstrair o Javascript/Ajax necessário ao processamento de eventos e aos efeitos visuais da aplicação e a capacidade de simular um ambiente multi-janela dentro de uma única janela de *browser* (ambiente semelhante ao que os utilizadores do Saga utilizam actualmente).

3.2.1 AJAX

Sendo o Gaia Ajax Widgets uma framework Ajax, é determinante apresentar alguns conceitos essenciais à compreensão desta técnica, pelo que esta secção se destina a apresentar os principais conceitos do Ajax.

Ajax (Asynchronous JavaScript And XML) é uma técnica que permite criar aplicações Web mais rápidas e amigáveis para o utilizador através da combinação de diferente de standards, como JavaScript, XML e HTML, CSS. [6]

Com Ajax, o JavaScript consegue comunicar e trocar dados directamente com o servidor sem necessitar de recarregar a página completa. Desta forma, a experiência de utilizador é mais agradável, pelo facto de este não visualizar a página inteira a ser carregada, mas apenas uma pequena porção (actualização parcial). Com uma utilização correcta do Ajax, é possível tornar esta actualização transparente para o utilizador final.

Com o JavaScript tradicional, para se obter dados de uma base de dados ou ficheiro de um servidor ou enviar informação de utilizador para um servidor, é necessário construir um formulário HTML. O utilizador tem de pressionar o botão 'Submit' para enviar/obter a informação, esperar que o servidor responda, e que uma nova página carregue com o resultado do processamento do formulário por parte do servidor.

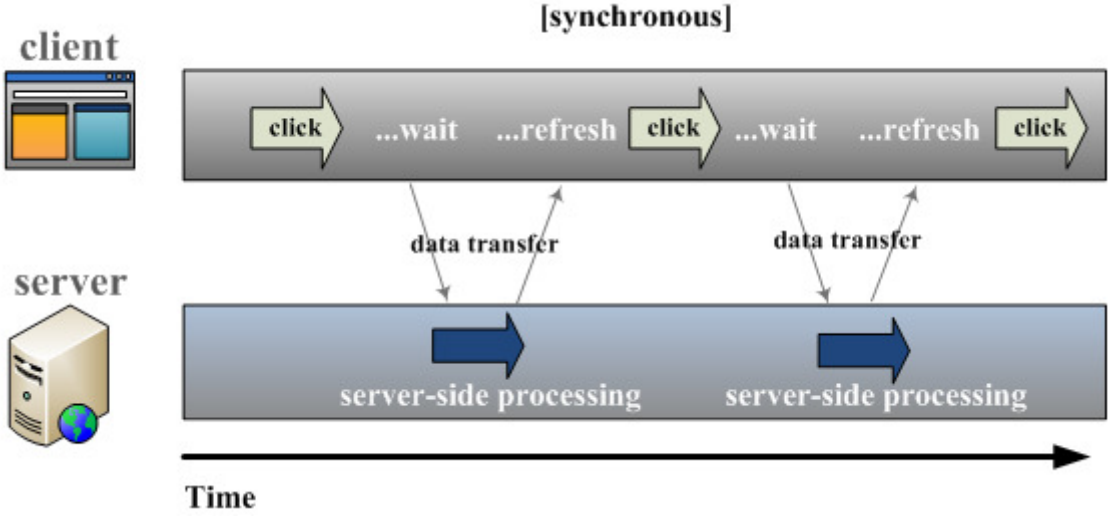


Figura 22 - Tratamento síncrono de eventos de utilizador

Devido ao facto do servidor retornar uma nova página de cada vez que o utilizador submete informação, as aplicações Web tradicionais podem tornar-se lentas e pouco amigáveis para o utilizador. Com AJAX e a utilização do objecto XMLHTTPRequest, um utilizador pode originar um pedido ao servidor Web, continuar a trabalhar, e obter a resposta sem recarregar a página, e sem se aperceber que os scripts da página estão a enviar pedidos para um servidor.

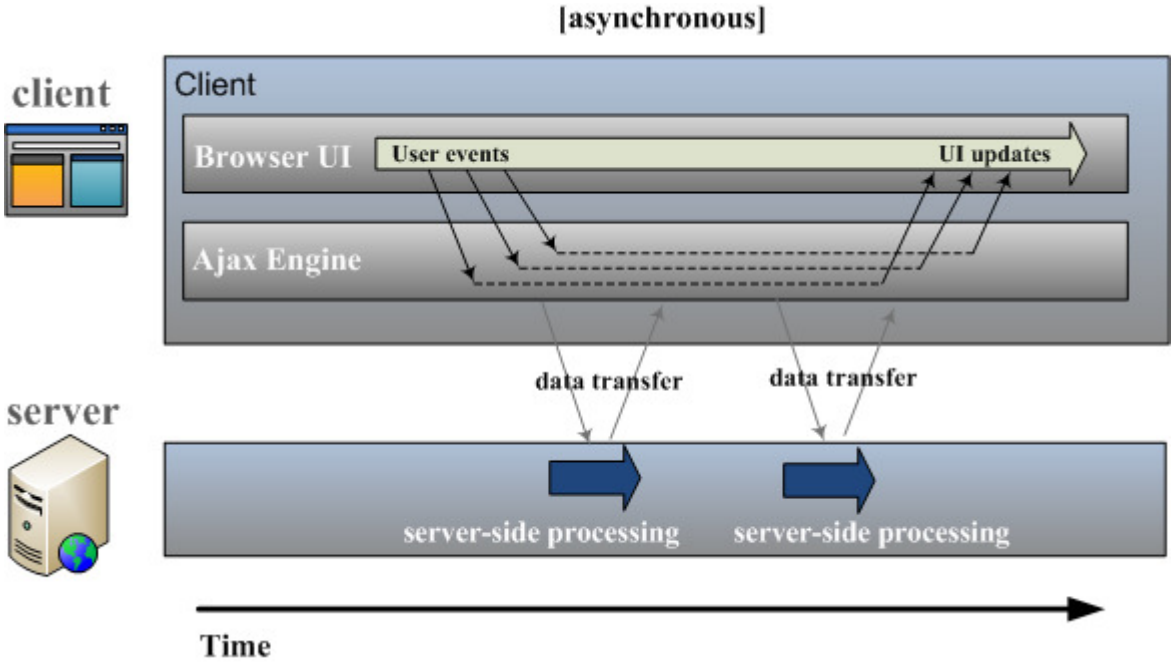


Figura 23 - Tratamento assíncrono de eventos de utilizador

3.2.2 Introdução ao Gaia

O Gaia Ajax Widgets (Gaia) (Figura 24) é uma framework Ajax *open source* de aplicações Web, criada na Noruega pela Gaiaware, anteriormente conhecida por Frost Innovation. [7]

Figura 24 - Gaia Ajax Widgets

O Gaia tem como principais características:

- Dispensar a escrita de Javascript/Ajax. O Gaia possui um conjunto de ficheiros Javascript responsável por executar funções avançadas do lado do cliente (criando uma aplicação 'rica'[9]) e efectuar pedidos Ajax ao servidor. Assim, o Gaia possui suporte avançado de Ajax, que permite o desenvolvimento simples de aplicações Web complexas com reduzido esforço do programador, pois este apenas necessita de escrever *code-behind* numa linguagem .Net à sua escolha.

- Dispor de ambiente de janelas. Os controlos nativos do Gaia Ajax Widgets (controlo *Window*) permitem a simulação de um ambiente multi-janelas no *browser* Web.

- Dispensar ScriptManager. Numa aplicação ASP.Net Ajax, o controlo ScriptManager é responsável pela gestão de script cliente e a sua utilização é obrigatória para o acesso a características do ASP.Net Ajax tais como a disponibilização das funcionalidades de script cliente da Microsoft Ajax Library e a actualização parcial de páginas, que permite que regiões da página sejam actualizadas independentemente do resto da página, e sem a ocorrência de *postback*[8]. Com o Gaia, não é necessário utilizar o controlo ScriptManager.

- *Rendering* parcial do Javascript. O Gaia envia ficheiros Javascript para o *browser* apenas quando necessário. O *browser* guarda estes ficheiros em *cache*, o que melhora significativamente o tempo de carregamento de páginas.

- Cliente com estado. Com o Gaia, o servidor apenas envia Javascript para o browser. Este *script* é executado do lado do cliente para, por exemplo, fazer pequenas alterações aos valores dos controlos. Desta forma, e porque não há envio de HTML, nem substituição de conteúdo, o Gaia faz com que o cliente mantenha o seu estado.

- Compatibilidade com ASP.Net 2.0. O Gaia permite a utilização de *postbacks* integrais (modelo comum de programação ASP.Net) e a combinação dos seus controlos com controlos do ASP.Net.

- Independente de plataforma. Para além do suporte a Windows, o Gaia é compatível com Linux, através do Mono, a framework .Net para Linux.

- Independente de Web Browser. O Gaia é 100% compatível com Microsoft Internet Explorer, Mozilla Firefox e Opera.

- Integrável. O Gaia permite integrar os seus controlos no SharePoint e DotNetNuke.

[9] Rich Internet Applications (RIA) são aplicações que possuem as características e funcionalidades das aplicações de *desktop* tradicionais. Este tipo de aplicações mantém o processamento da actividade da UI no lado do cliente e o processamento de dados no lado do servidor.

3.2.3 Experimentação

A experimentação do Gaia deu-se em duas fases, sendo que a primeira ocorreu no período compreendido entre 30 de Novembro de 2007 e 12 de Janeiro de 2008, e a segunda de 14 a 25 de Janeiro.

Durante a primeira fase, foi seguida uma abordagem semelhante à utilizada no passado para a integração do Saga com uma nova tecnologia de interface. Sempre que foi necessário suportar uma plataforma nova (Windows, Curses, …), criou-se um novo conector para um sistema de janelas. Uma versão simplificada da arquitectura modular do Saga com este tipo de abordagem pode ser vista na Figura 25.

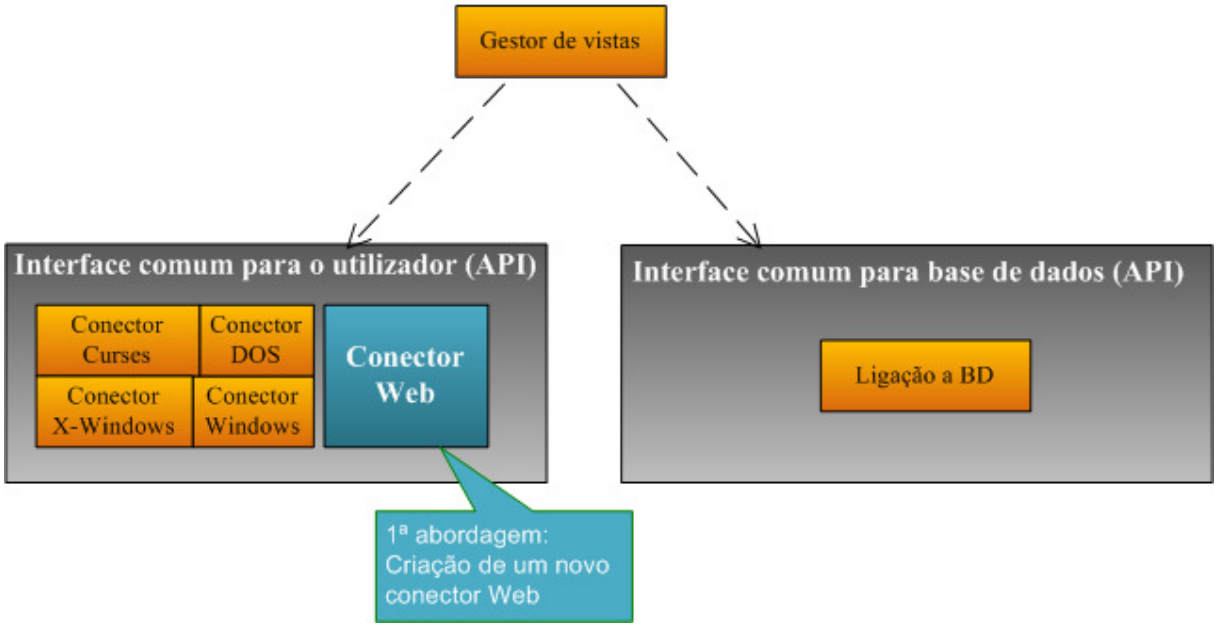


Figura 25 - Arquitectura simplificada do Saga na 1ª abordagem utilizada com o Gaia

Para o Saga, o novo conector disponibiliza uma API semelhante à que é disponibilizada pelo conector para Windows. A criação e alteração de estado de objectos da interface para o utilizador é solicitada pelo Saga, que invoca, através de *callback*, métodos do módulo em C#. O módulo em C#, por sua vez, cria objectos e invoca métodos do Gaia.

O módulo de C# possui referências para um conjunto de funções da Dynamic-Link Library (DLL) nativa, às quais passa apontadores de funções (métodos de callback do C#). Foram utilizados os Platform Invocation Services (PInvoke) da framework .Net para permitir que o código managed (C#) do novo módulo Web invoque funções unmanaged (C) implementadas na DLL. A DLL pode, assim, utilizar estes apontadores para chamar os métodos de abertura de janelas, alteração de valores de controlos, etc. do SagaWeb.

A aplicação de teste desenvolvida permite abrir as janelas das vistas com os respectivos controlos, simulando o ambiente multi-janela dentro do browser.

Para o tratamento de eventos de utilizador sobre os controlos da UI, o módulo C# define *event handlers*, que tratam de captar os eventos e os propagar para o Gaia (pelo conector gaia.dll). Deste modo, consegue-se uma interacção bidireccional entre a DLL e o módulo C#.

Foram encontradas algumas dificuldades no desenvolvimento desta solução, principalmente no que diz respeito à criação dinâmica de controlos e às diferenças existentes entre o modelo de programação do Gaia e a Win32 API, base do desenvolvimento do módulo anterior. Esta abordagem exige um conhecimento mais profundo da implementação original do Saga e aumenta a dependência da equipa de trabalho do conhecimento do Professor João Pascoal Faria.

Após uma reunião com Thomas Hansen, especialista da Gaiaware, a equipa de projecto decidiu tomar um novo rumo. Até então, estava a ser desenvolvido um módulo que interagia com o gestor de janelas do Saga, criando controlos dinamicamente e preenchendo-os com dados a pedido deste. Thomas Hansen criou, de 12 a 14 de Janeiro de 2007, as bases do projecto Gaia MVC. Este projecto, ainda que numa fase embrionária, deu origem a uma nova abordagem, semelhante à utilizada com o Ruby On Rails (Figura 26).

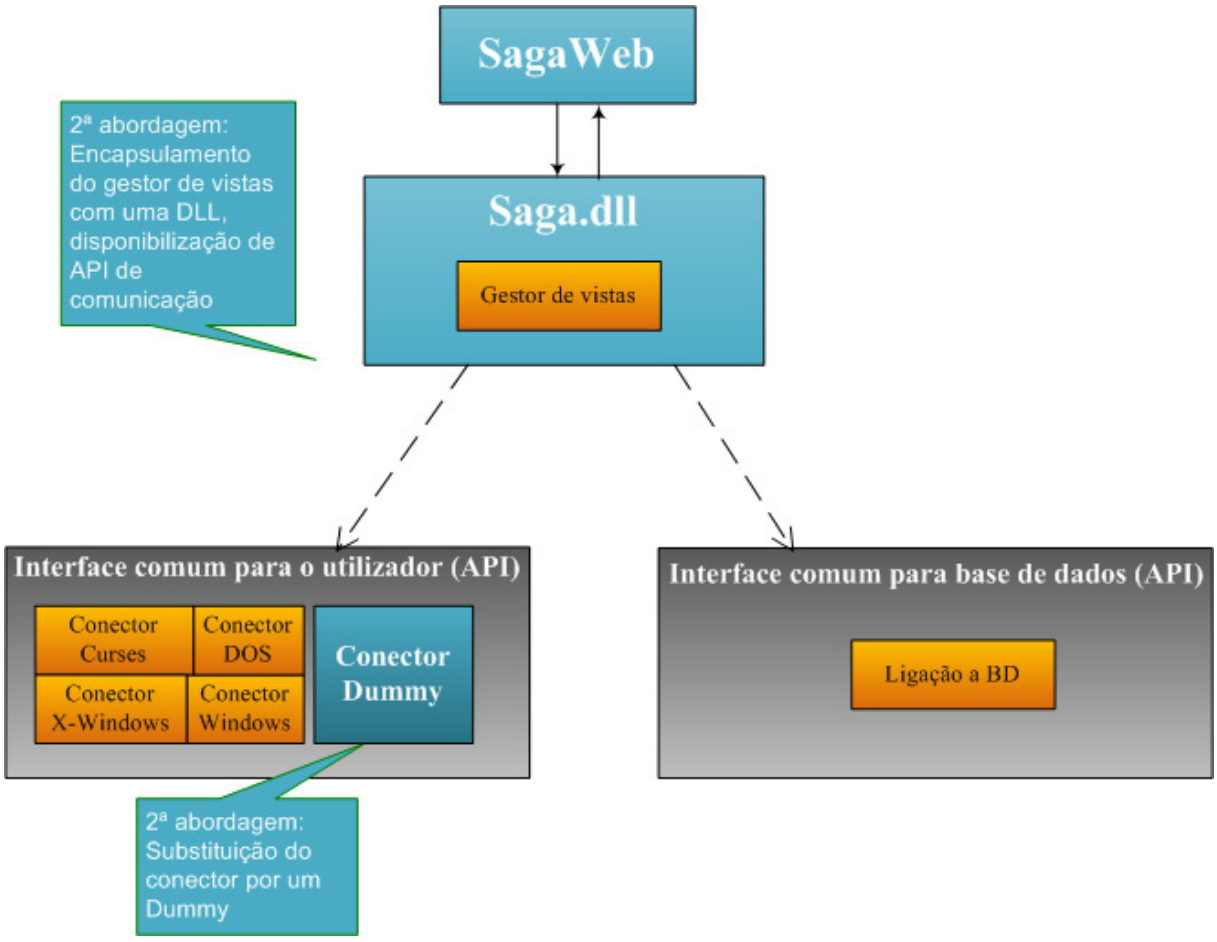


Figura 26 - Arquitectura simplificada do Saga na 2ª abordagem utilizada com o Gaia

Para o desenvolvimento do SagaWeb com o Gaia MVC, foi construída uma *façade* .Net sobre o servidor de vistas do Saga, com a qual o SagaWeb comunica para obter e manipular dados de vistas. A definição de vistas encontra-se no sistema de ficheiros em ficheiros de vistas compiladas (*.vcs). O SagaWeb gera dinamicamente um UserControl (*.ascx) por cada vista definida nos ficheiros de vistas compiladas. O conector UI é substituído por um conector *dummy,* retirando a funcionalidade de disponibilizar UI ao Saga. Isto acontece porque todos

os métodos chamados pelo gestor de janelas (abrir, fechar janelas, etc.) se encontram vazios neste novo conector. Os detalhes do desenvolvimento encontram-se descritos no capítulo 4.

3.2.4 Conclusões

A experimentação com a framework Ajax Gaia Ajax Widgets permitiu obter algumas conclusões importantes para a selecção da tecnologia a utilizar no desenvolvimento do SagaWeb. O Gaia Ajax Widgets permite dotar o SagaWeb de funcionalidades Ajax de uma forma simples e transparente para o programador. O conjunto de controlos nativos do Gaia Ajax Widgets permite simular no browser um ambiente multi-janela que proporciona uma experiência de utilizador semelhante à disponível actualmente no ambiente Windows, disponibilizando formas simples de melhorar o aspecto visual das aplicações (através da utilização de *skins*).

O projecto iniciado por Thomas Hansen, da Gaiaware, Gaia MVC, resolve o problema da gestão da estrutura da árvore de controlo, simplificando o tratamento de eventos do lado do servidor.

3.3 Silverlight

Este capítulo apresenta o estudo e a experimentação efectuados sobre a tecnologia Microsoft Silverlight. Apesar de ter sido relativamente curto, o período de experimentação desta tecnologia permitiu conhecer o potencial futuro que possui (principalmente a partir da sua versão 2.0), pelo que esta secção será dedicada a uma breve descrição da tecnologia Microsoft Silverlight.

Figura 27 - Exemplo de conteúdo Silverlight numa página Web

3.3.1 Introdução ao Silverlight

O Microsoft Silverlight (anteriormente Windows Presentation Foundation / Everywhere – WPF/E) é uma tecnologia independente de browser e plataforma para a construção de

conteúdo interactivo 'rico' para a Web[10], que exige apenas que seja instalado um *plugin* na máquina do cliente Web para que este possa suportar conteúdo desta tecnologia. O Silverlight é visto, por muitos, como a resposta da Microsoft ao Flash. [9][10]

A sua funcionalidade encontra-se totalmente encapsulada no *plugin* Silverlight. As aplicações Web exigem, tipicamente, que o servidor que contém a aplicação Web verifique um conjunto de requisitos. As aplicações Silverlight apenas requerem um servidor Web, da mesma forma que os documentos HTML

A execução de uma aplicação Silverlight num cliente Web é um processo de 2 passos. Inicialmente, a aplicação detecta se o *plugin* Silverlight se encontra instalado na máquina do cliente Web. Se o *plugin* não estiver instalado, será feito um pedido a partir do servidor Web para que se descarregue e o instale. O *plugin* Silverlight encontra-se num ficheiro executável *.dll* que é carregado para a memória do *browser* do cliente Web quando instalado. A única interacção exigida pelo cliente Web é permissão para instalar o *plugin*.

De seguida, quando o *plugin* estiver instalado na máquina do cliente Web, é necessário descarregar a aplicação Silverlight. Uma aplicação Silverlight pode consistir em vários ficheiros. Pode ser necessário modificar algumas configurações no servidor Web de forma a que os ficheiros Extensible Application Markup Language (XAML) sejam associados ao Silverlight e descarregados correctamente para a máquina do cliente Web quando necessário.

Quando o *plugin* Silverlight estiver instalado e a aplicação em si descarregada, esta última é executada na máquina do cliente Web. Existem alguns requisitos que a máquina do cliente Web deve cumprir. No entanto todos os *media players*, codecs áudio e vídeo, compiladores e *runtime* estão encapsulados no plugin Silverlight.

O seu conjunto de funcionalidades inclui:

- Formato declarativo de conteúdo baseado em XAML.
- Modelo de programação baseado em JavaScript.
- Diferentes opções de *deployment*: código embebido na página HTML, em ficheiros à parte ou compactados num ficheiro *zip*.
- Suporte multimédia, que inclui WMV, WMA, e MP3.
- Formatação e manipulação de texto.
- Suporte a imagens PNG e JPG, animações e gráficos.
- Tratamento de *input* de rato e teclado.

3.3.2 Arquitectura

A principal diferença (com relevo para o projecto SagaWeb) entre as versões 1.0 e 2.0 do Silverlight, consistem no suporte a .Net, presente apenas na versão 2.0. O facto do Microsoft Silveright 1.0 ser, de uma forma muito simplista, uma tecnologia de apresentação faz com que o seu envolvimento no projecto não seja tão interessante como seria o da versão 2.0, que terá um grande potencial na construção de aplicações de formulários Web. Na Figura 28

[10] O Silverlight também permite o desenvolvimento de conteúdo 'local'

encontra-se ilustrada a arquitectura do Silverlight, com particular relevo para as diferenças existentes entre as suas 2 versões.

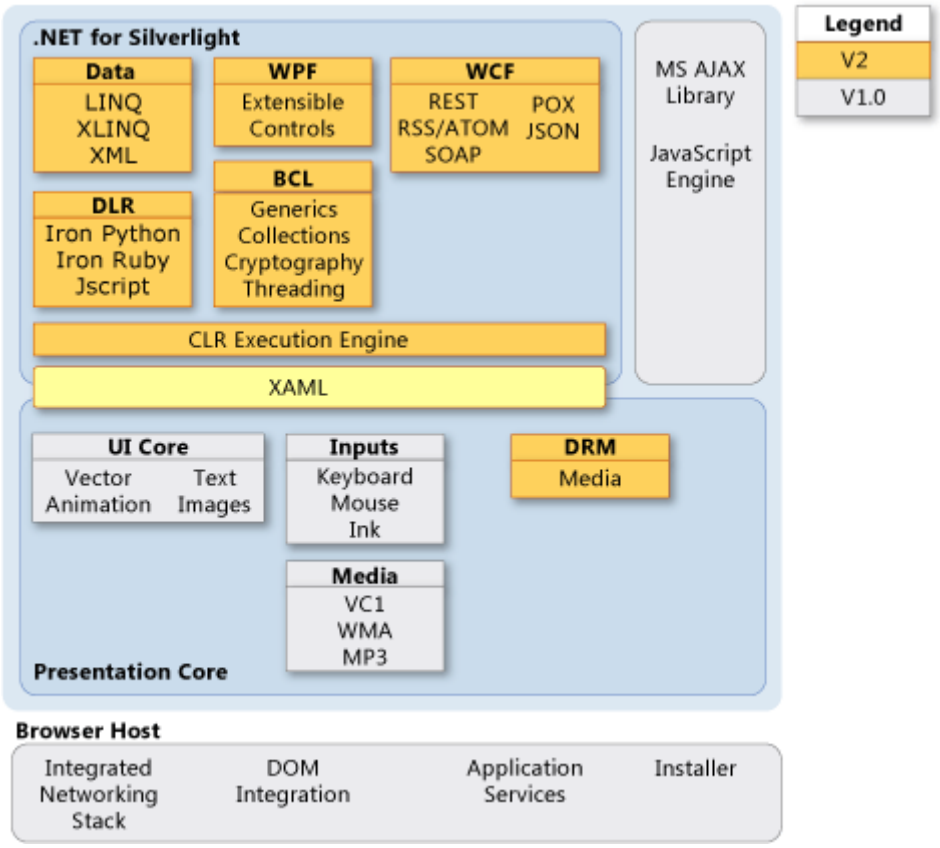


Figura 28 - Arquitectura do Microsoft Silverlight (1.0 e 2.0)

Entre as funcionalidades acrescentadas à versão 1.0 pelo Silverlight 2.0, estão o suporte a *Language Integrated Query* (LINQ), a presença do *Common Language Runtime* (CLR) do .Net e a possibilidade de utilização de controlos WPF (Button, DataGrid, DatePicker, etc.) [11].

3.3.3 Experimentação

Tal como mencionado anteriormente, a experimentação do Silverlight foi breve e permitiu à equipa do projecto perceber que a versão disponível (1.0) não se adequava à construção de aplicações de formulários, assemelhando-se a uma tecnologia de apresentação como o Flash.

Dado que, com o Silverlight 1.0, não se dispõe das funcionalidades (nem do conjunto de controlos) ASP.Net, a experimentação desta tecnologia consistiu no desenvolvimento de algumas aplicações Silverlight 1.0 com a ajuda de tutoriais Web da página oficial do Silverlight [12], que nos ajudaram a chegar às conclusões mencionadas anteriormente.

3.4 Ruby On Rails

Esta secção destina-se à descrição da tecnologia Ruby On Rails, estudada pelo elemento Anabela Monteiro da equipa do projecto SagaWeb. O Ruby On Rails foi a única tecnologia presente durante todo o período de experimentação.

3.4.1 Ruby

Ruby é uma linguagem *open source*, dinâmica[11] e orientada a objectos criada por Yukihiro Matsumoto, nos anos 90. O Ruby é multi-plataforma, podendo correr em Windows, Mac OS X, MS-DOS, Linux e Unix. [13]

O principal objectivo do Ruby é aumentar a produtividade do desenvolvimento de software. O Ruby propicia uma experiência que é não só produtiva, mas também divertida.

O Ruby é adequado à resolução de problemas dos seguintes domínios (entre outros):

- Processamento de texto – As classes *File*, *String* e *Regexp* do Ruby ajudam a processar texto de uma forma rápida e eficiente.
- Programação CGI – O Ruby tem as ferramentas necessárias para a programação CGI, incluindo classes de processamento de texto, uma biblioteca CGI, interface de base de dados, etc.
- Programação de Redes – As classes de *socket* do Ruby facilitam este tipo de programação.
- Programação GUI – O Ruby possui várias *toolkits* GUI que auxiliam o programador no desenvolvimento de interfaces.

3.4.2 Framework Ruby On Rails (MVC)

O Ruby On Rails é uma framework para o desenvolvimento rápido de aplicações Web que utiliza o padrão de arquitectura de software Model-view-controller (MVC). No desenvolvimento de aplicações Web complexas, é importante separar dados da apresentação, por forma a que as alterações feitas em qualquer uma delas não se reflictam na outra. O padrão MVC (Figura 29) consegue esta separação através da introdução de uma terceira camada entre elas.

[11] Linguagem de alto nível que consegue executar em *runtime* funções normalmente executadas na fase de compilação como, por exemplo, extensão de objectos e definições.

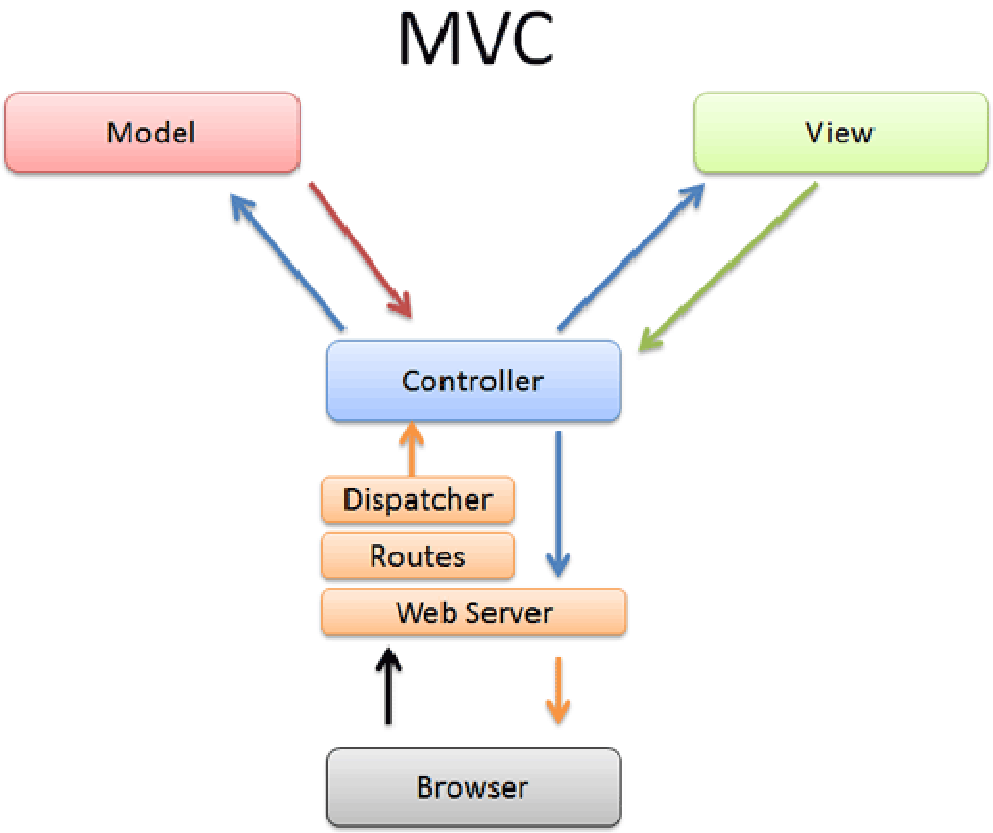


Figura 29 - Padrão de Arquitectura de Software MVC

O padrão MVC utiliza as seguintes camadas:

- Model – modelo de objectos, camada de dados (BD)
- View – apresentação de dados, interface para o utilizador (Browser)
- Controller – processamento de eventos da interface (View), alteração de dados (Model)

Esta framework disponibiliza, também, a funcionalidade de *scaffolding*, que constrói automaticamente a maioria dos *Controllers e Views* necessárias para o desenvolvimento de uma aplicação com funcionalidades CRUD a partir da especificação do *Model*.

3.4.3 Experimentação

Tal como referido anteriormente, a responsável pela experimentação desta tecnologia foi Anabela Monteiro. O desenvolvimento com o Ruby On Rails revelou-se mais fácil para o programador, e a aplicação desenvolvida foi uma das duas tecnologias finais consideradas para o desenvolvimento do SagaWeb.

3.5 Análise comparativa e selecção de tecnologias

O *benchmarking* teve como base 4 tecnologias: Ruby On Rails, Gaia Ajax Widgets, ASP.Net e Silverlight. No caso de Gaia Ajax Widget foram testadas 2 abordagens distintas,

tendo sido escolhida para a análise comparativa a última delas (Gaia MVC). O objectivo do *benchmarking* era o de verificar a resposta de cada tecnologia aos requisitos do projecto e determinar qual o projecto que seria capaz de cumprir os requisitos do projecto da forma mais satisfatória.

Apesar da grelha da Figura 30 dar a entender que a análise comparativa foi feita entre as 4 soluções em simultâneo no fim do período reservado à experimentação de tecnologias, isto não corresponde à verdade. Ao longo de toda a experimentação, foram sendo testadas 2 tecnologias em simultâneo, sendo que Ruby On Rails foi sempre uma delas.

Deu-se início ao processo de *benchmarking* com as tecnologias Ruby On Rails e ASP.Net. A equipa considerou, numa primeira instância, os resultados obtidos com o Ruby On Rails mais satisfatórios do que os obtidos com ASP.Net, o que levou ao abandono desta última, e ao início da experimentação das 2 abordagens Gaia Ajax Widgets. Entre as 2 abordagens do Gaia, fez-se uma curta experiência com Silverlight, que também foi abandonada devido ao facto da sua versão actual ser orientada à apresentação e, por isso, não ser capaz de conferir ao projecto o mínimo grau de funcionalidade que este exige. Assim, a derradeira comparação deu-se entre Ruby On Rails e Gaia Ajax Widgets, com a sua abordagem MVC.

As classificações fizeram-se com a utilização de uma escala de 'estrelas' que toma valores inteiros de 1 (⭐) a 5 (⭐⭐⭐⭐⭐). Para ilustrar a diferença entre as classificações obtidas por Ruby On Rails e Gaia, foram marcadas as estrelas em excesso na classificação de uma tecnologia num requisito em relação à sua 'adversária' (🟢). Em alguns casos, é atribuída uma classificação tendo em conta o potencial futuro da tecnologia. Nestes casos, a diferença entre a classificação actual e a que a equipa prevê que a tecnologia atinja no futuro é assinalada por estrelas entre parêntesis.

As classificações presentes na grelha da Figura 30 foram atribuídas depois de discussão e, apesar de, por vezes, se referirem a aspectos incomensuráveis (difusão da tecnologia, por exemplo), representam as ilações retiradas após o período de experimentação e devem ser tidas em conta apenas no âmbito deste projecto.

Requisito	Ruby On Rails	Gaia Ajax Widgets	ASP.Net	Silverlight
Compatibilidade	★★	★★★	★★	★★(★★★)
Portabilidade	★★★★★	★★★	★★	★(★)
Web Design e Usabilidade	★★★	★★★★★		★★(★★★)
Desempenho	★★	★★★		★(★★★)
Suporte	★★	★★★★	★★★★	★
Rapidez, robustez, facilidade de desenvolvimento	★★★	★★★	★	★★(★★★)
Maturidade, difusão	★★★	★★(★)	★★★★★	★
Componentarização, integração	★★★	★★★		
Facilidade de manutenção, extensão	★★★	★★★	★★	
Custos / licenças	★★★★★	★★(★)	★★★	★★★

Figura 30 - Grelha de comparação de tecnologias

De seguida serão comentadas algumas das classificações presentes na Grelha de comparação de tecnologias (factores de diferenciação das soluções Ruby On Rails e Gaia Ajax Widgets):

- Compatibilidade das aplicações da versão anterior

O facto da tecnologia Gaia se basear na framework .Net facilita a sua integração com o servidor de vistas, pela facilidade de utilizar .Net Remoting. É mais difícil encontrar uma solução para este problema com o Ruby On Rails. O facto de o Gaia permitir simular um ambiente de trabalho mais próximo do original também leva a crer que terão de ser feitas menos alterações nas aplicações para impedir que o utilizador tenha uma má experiência de migração (por não estar familiarizado com o ambiente de utilização).

- Portabilidade

O Ruby On Rails leva clara vantagem sobre as demais tecnologias por se tratar de uma tecnologia que garante independência de plataforma, tanto do lado do cliente como do servidor.

O Gaia contorna o facto de funcionar sobre ASP.Net através da utilização de Mono[12] para o seu funcionamento em Linux.

- Web Design e Usabilidade

No contexto deste projecto, a solução Gaia apresenta vantagens em relação às outras tecnologias analisadas. O Gaia disponibiliza um conjunto de *widgets* com um *design* atractivo e uma boa interactividade com o utilizador, graças ao suporte avançado de Ajax, e já suporta um ambiente de janelas atractivo no *browser*, com muito pouco esforço de programação. Permite também editar visualmente os User Controls (*.ascx) gerados (no Visual Studio, por exemplo). Apesar do Web Design, numa última instância, poder ser optimizado através da edição de *skins* ou ficheiros Cascading Style Sheet (CSS), o facto de se poder editar o aspecto das vistas num IDE e de esta tecnologia já dispor de controlos que permitem criar *rich applications* atribuem vantagem ao Gaia.

- Desempenho

Apesar do desempenho das duas soluções parecer semelhante, é natural que o Gaia, por utilizar uma linguagem compilada, leve vantagem, neste capítulo, sobre a framework de linguagem interpretada Ruby On Rails

- Suporte

A tecnologia Gaia Ajax Widgets tem um suporte muito forte da Microsoft e da própria Gaiaware, como ficou comprovado pelo constante acompanhamento de José António Silva, da Microsoft, e pela deslocação de um perito da Gaiaware, Thomas Hansen, a Portugal, para ajudar no desenvolvimento do SagaWeb.

- Custos / licenças

Ruby On Rails tem uma vantagem clara sobre o Gaia Ajax Widgets, devido ao facto de ser uma tecnologia grátis.

Após a discussão dos resultados obtidos a partir da análise e experimentação das tecnologias candidatas, a equipa de projecto e a Medidata tomaram a decisão de continuar a implementação do SagaWeb com o recurso a uma única tecnologia, o *Gaia Ajax Widgets*. A implementação será feita sobre o projecto embrionário *Gaia MVC*, da Gaiaware.

[12] Implementação *open source* da arquitectura Microsoft .Net

4 Arquitectura da nova camada Web em Gaia

Este capítulo tem como principal objectivo descrever a nova camada Web e as principais decisões de desenho e implementação tomadas no decurso do projecto.

Inicialmente, será apresentada a arquitectura geral do SagaWeb, seguida da descrição mais detalhada de alguns dos principais processos e características da nova camada. Por fim, serão descritas os principais problemas encontrados no seu desenvolvimento e as soluções encontradas para as mesmas.

4.1 Arquitectura geral

As principais decisões de desenho do SagaWeb consistem no encapsulamento do servidor de vistas do Saga (DLL, código legado em C) com o recurso a *managed code* (C#) e na conversão em *runtime* da definição de vistas (ficheiros *.vcs) para User Controls (*.ascx). A Figura 31 representa a arquitectura geral do SagaWeb.

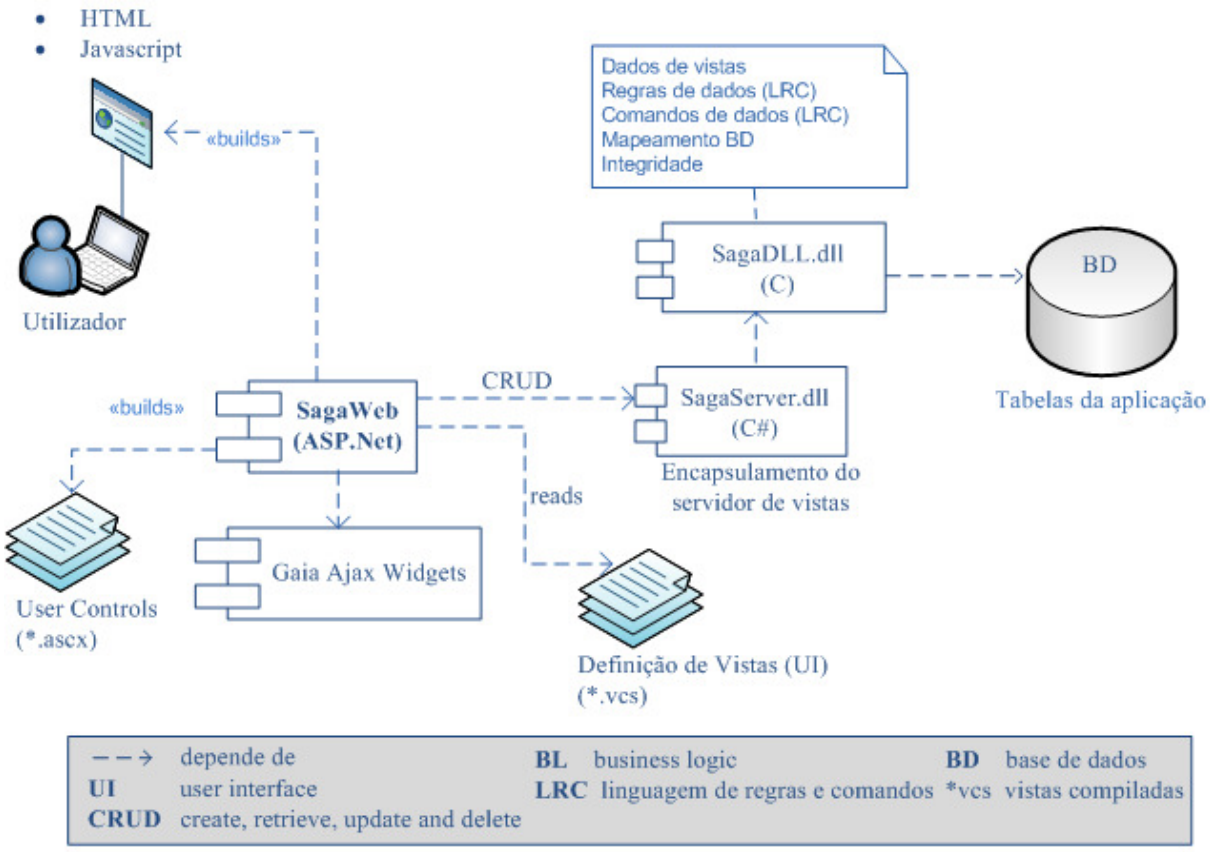


Figura 31 - Arquitectura geral do SagaWeb

O servidor de vistas do Saga permanece responsável pela lógica de negócio e acesso a dados das vistas. O seu encapsulamento permite disponibilizar uma API que é utilizada pela camada de interface para efectuar operações CRUD sobre os registos das vistas. A camada de Interface interpreta as definições de vistas presente em ficheiros '.vcs' e converte-as em User

Controls (ficheiros '*.ascx'), que são integrados em páginas .Net. Este processo mantém-se transparente para o utilizador final, que apenas recebe no Web browser HTML e Javascript.

4.2 Geração dinâmica de *user controls* correspondentes às vistas

Conforme já foi referido, é possível utilizar vistas compiladas (ficheiros *.vcs*) na execução do Saga para tornar a abertura das vistas mais rápida e diminuir a carga no servidor.

De acordo com o ciclo de vida dos controlos de servidor .Net dinâmicos[13], o tratamento de eventos no servidor originados pelo cliente apenas é possível se a estrutura de controlos da interface no servidor corresponder à presente no lado do cliente. Assim, se o SagaWeb utilizasse os ficheiros .vcs como representação das vistas, seria obrigado a interpretar os ficheiros em runtime e criar a estrutura de controlos de cada vez que ocorresse um evento. No caso dos controlos estáticos[14], a gestão da estrutura de controlos para tratamento de eventos é 'assumida' automaticamente pela framework .Net, de forma transparente para o programador.

Os User Controls são um tipo de controlos da framework .NET que podem incluir código HTML estático e ainda Web Controls. Codificados em ficheiros com extensão *.ascx*, os User Controls permitem a sua reutilização em várias páginas na mesma aplicação (são utilizados muitas vezes para representar cabeçalhos, rodapés, e outros elementos de interface comuns de uma página).

De forma a libertar o servidor da leitura e processamento de um ficheiro *vcs* por cada vista aberta por pedido, os responsáveis da Gaiaware iniciaram um projecto com o nome de *Gaia MVC* cuja funcionalidade mais relevante é a de criar (e carregar para a página) um *User Control* com a representação do aspecto e comportamento de uma vista.

O SagaWeb utiliza esta funcionalidade do *Gaia MVC* para implementar o seu mecanismo de *cache*. Quando o SagaWeb tem a *cache* desactivada, constrói, a cada pedido de abertura de vista, o User Control correspondente e carrega-o para a página *aspx*. Caso contrário, quando ocorre um pedido, o SagaWeb procura de imediato o User Control (ficheiro '*.ascx') correspondente à vista numa pasta do sistema de ficheiros pré-definida (*cachedViews*). Se este não se encontrar presente na pasta (não está em *cache*), o SagaWeb gera-o e guarda-o em *cache*, disponibilizando-o para os acessos subsequentes.

Para que a criação do User Control parta de um ficheiro *vcs*, e não de um ficheiro *xml*, como está implementado por omissão no *Gaia MVC*, o SagaWeb necessita de fazer *override* do método virtual *LoadNonCachedView* da classe principal do *Gaia MVC*, a *Gaia.MVC.Page*. Este método é chamado quando não são usadas vistas em *cache* (este comportamento é possível através da simples activação de uma *flag*), situação em que os *User Controls* são gerados em todas as chamadas, independentemente de já existirem ou não no sistema de ficheiros, ou quando são usadas vistas em cache, mas a definição da vista pedida ainda não foi materializada num *User Control* (1ª visualização da vista).

De forma a ter conhecimento de quais as vistas a recarregar (é importante relembrar que as vistas 'activas' devem ser adicionadas à página em todos os pedidos ao servidor, para que se possam processar os eventos), o SagaWeb mantém o conjunto dos identificadores das

[13] Controlos criados e adicionados a uma página *aspx* em *runtime*

[14] Controlos declarados numa página *aspx* ou *ascx*

vistas 'activas' no *ViewState*, que viaja continuamente entre o servidor e o cliente. De cada vez que recebe um *postback*, o SagaWeb determina no *ViewState* quais as vistas que estão abertas e volta a carregá-las.

O Anexo C mostra o conteúdo de um ficheiro de uma vista compilada do Saga. Um excerto do código resultante da conversão da vista compilada para um User Control pode ser visto no Anexo D.

Na Figura 32 pode ver-se a representação da vista (User Control) na interface para o utilizador.

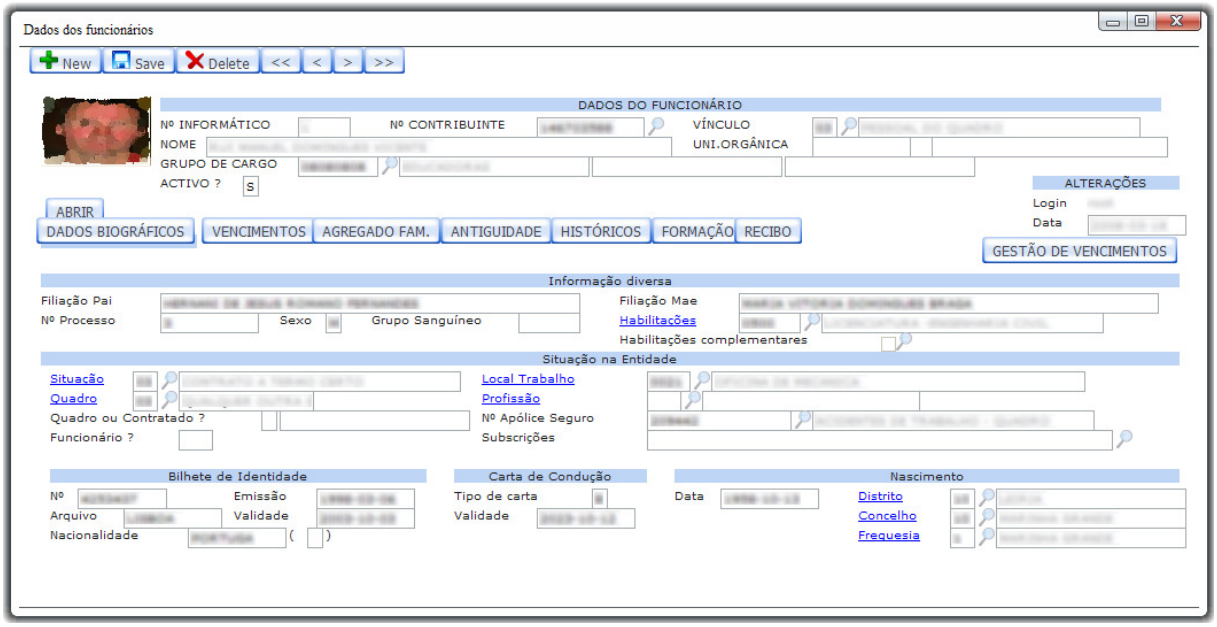


Figura 32 - Vista *cadastro* representada na interface para o utilizador

4.2.1 Código de vistas

No processamento das vistas compiladas (**.vcs*), o SagaWeb determina propriedades visuais das vistas através da interpretação das linhas marcadas no ficheiro *.vcs* com a *label* 'vista' (cada linha 'vista' corresponde à definição de uma vista), tais como:

- Nome
- Posição (distância do topo, distância da margem esquerda)
- Altura e largura
- Título

O SagaWeb representa cada uma das vistas num controlo Window.

4.2.2 Código de campos de vistas

As propriedades visuais dos campos das vistas encontram-se representadas nos ficheiros *.vcs* nas linhas marcadas com a *label* 'campista' (**campo** de **vista**, cada linha 'campista' corresponde à definição de um campo de vista). Dependendo da combinação de 3

propriedades, os campos de vista podem ser representados através de diferentes tipos de controlo. De seguida, são enumeradas as combinações possíveis das propriedades dos campos de vista que condicionam o tipo de controlo correspondente.

1. Se a combinação das propriedades 'formato' e 'formato_bd' indica que o formato corresponde a um dos valores 'STRING', 'FLOAT' ou 'NUMERIC', então:
    a. Se 'tipo_visual' for igual a 'bitmapbutton', campo é representado através de um controlo do tipo Image.
    b. Se 'tipo_visual' for igual a 'combobox', campo é representado através de um controlo do tipo Dropdown.
    c. Se 'tipo_visual' for igual a 'checkbox', campo é representado através de um controlo do tipo CheckBox.
    d. Se 'so_para_ver' for igual a 'S' (só de leitura), campo é representado através de um controlo do tipo Label.
    e. Senão, campo é representado através de um controlo do tipo TextBox.
2. Se a combinação das propriedades 'formato' e 'formato_bd' indica que o formato corresponde a um dos valores 'DATE', ou 'LDATE', então:
    a. Campo é representado através de um controlo do tipo Calendar.

Para além do tipo, o SagaWeb obtém informação sobre outras características do controlo que representa o campo na interface através das linhas 'campista' do ficheiro *vcs*:

- Nome
- Posição (distância do topo, distância da margem esquerda)
- Altura e comprimento

4.2.3 Código de zoom

O Saga define um mecanismo de encadeamento de vistas para navegação por 'zoom' (ou *lookup*), de um campo de uma vista para a totalidade doutra vista. O encadeamento das vistas encontra-se definido nas vistas compiladas nas linhas marcadas com a *label* 'zoomcmp'. Se um campo possui *Zoom* para uma vista, o SagaWeb assinala-o (através de uma imagem de uma lupa ao lado do controlo correspondente, ou transformando o controlo *Label* do campo num *LinkButton*). Ambos os métodos criam um controlo que origina (quando o utilizador 'clica' sobre o mesmo) uma chamada ao servidor, fazendo um pedido de abertura da vista de *Zoom*.

4.3 Comunicação com o servidor de vistas

Embora esteja prevista a sua modificação, o servidor de vistas do Saga, que consiste numa DLL nativa, desenvolvida em código *unmanaged*, não é, actualmente, *multithreaded.*

Como a versão actual necessita de manter estado (dados de vistas abertas), existe a necessidade de manter um processo isolado do servidor de vistas para cada sessão do utilizador. Se o SagaDLL fosse "linkado" dinamicamente, dentro da ambiente de execução .Net, não era garantida a existência de um processo por sessão.

A solução encontrada foi a de desenvolver uma "capa" .Net de forma a ter uma DLL em *managed* code, invocável através de .Net Remoting[15]. Desta forma, é possível lançar um processo do servidor de vistas por sessão do utilizador. O servidor de vistas deve escutar numa porta por pedidos do SagaWeb.

Foram desenvolvidas 2 classes com este propósito, SagaServer e SagaGateway.

SagaServer é um serviço (*client-activated*) que carrega a DLL nativa do Saga e disponibiliza alguns métodos que permitem a interacção com a mesma:

- SagaIni, método que inicializa a execução do servidor de vistas.
- SagaEnd, método que termina a execução do servidor de vistas.
- SagaEval, método que passa ao servidor de vistas um comando e retorna o resultado da execução do mesmo.
- SagaEvalError, método que obtém as mensagens de erro presentes no servidor de vistas.

É o SagaWeb (cliente) que lança o processo deste servidor, indicando a porta por onde decorre a comunicação. De seguida, o cliente instancia um objecto no servidor, sendo lançado, deste modo, um processo separado por cliente.

SagaGateway é uma classe que constituirá o serviço '*stateful'* (*singleton*[16]) e '*thread-safe*' que gere a concessão de portas dentro de uma gama aos clientes. Este serviço corre numa porta e concede portas cujo número varia entre uma gama de valores. Estas variáveis (nº de porta, intervalo de portas a conceder) são passadas ao SagaGateway por parâmetro aquando do início da sua execução.

A Figura 33 representa o diagrama de sequência da comunicação do SagaWeb com o servidor de vistas.

[15] API Microsoft para comunicação entre processos

[16] Padrão de desenho destinado a limitar a instanciação de uma classe a um único objecto

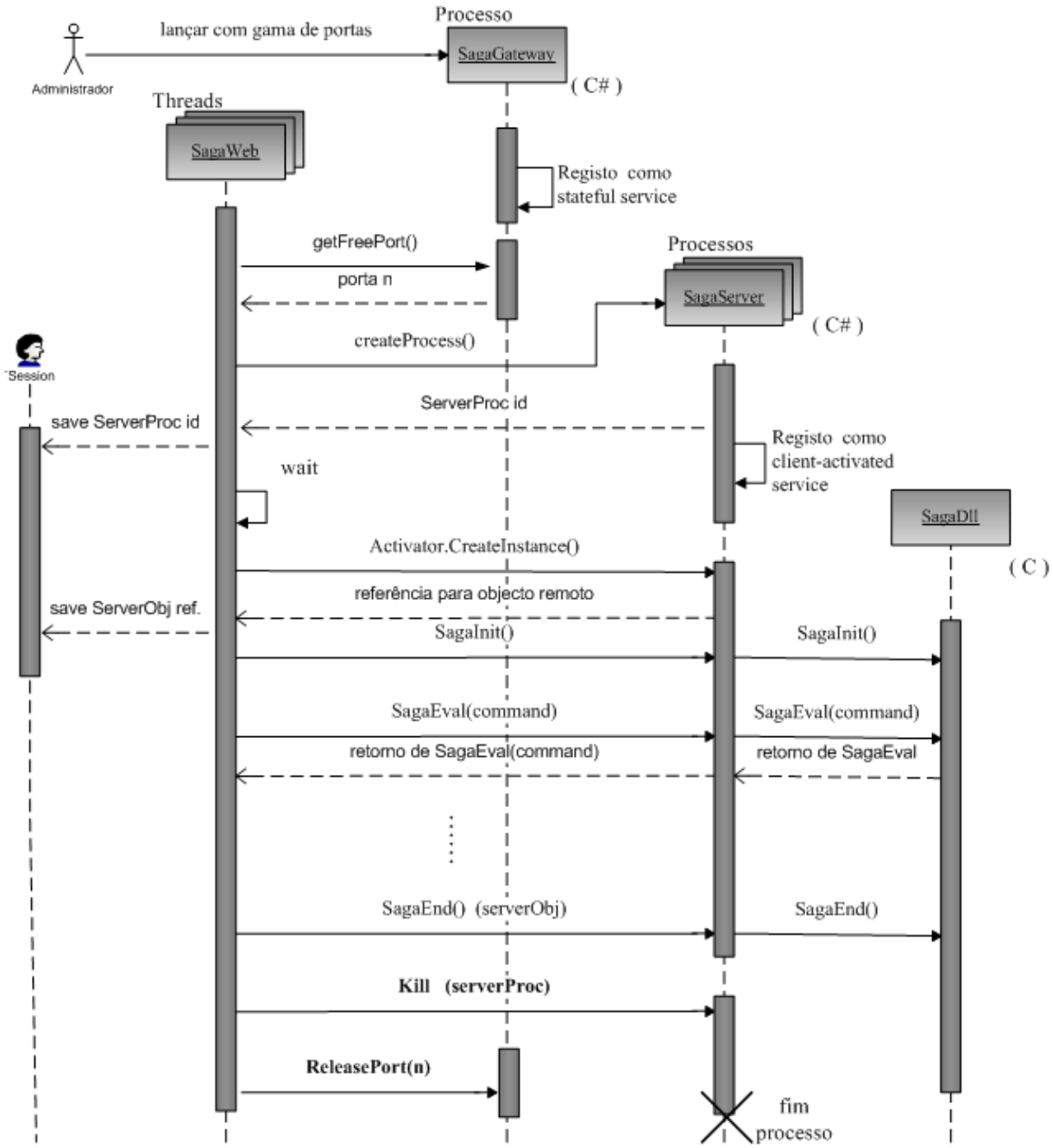


Figura 33 - Diagrama de sequência de comunicação do SagaWeb com o servidor de vistas

O SagaGateway corre como um processo único, sendo lançado por um administrador da máquina, que lhe indica qual a porta onde deve escutar pedidos, e qual a gama de portas que pode conceder. Depois de se registar como um *stateful service*, o SagaGateway está pronto para atender pedidos das *threads* da aplicação SagaWeb.

Cada thread SagaWeb efectua um pedido de uma porta disponível ao SagaGateway. Depois de obter uma porta disponível, o SagaWeb lança um processo SagaServer, guardando uma referência para o mesmo na Sessão (*serverProc*), e aguarda que este se registe como um serviço *client-activated*. De seguida, o SagaWeb activa o processo SagaServer, obtendo uma referência para o objecto SagaServer (*serverObj*), que também guarda na Sessão.

O SagaWeb utiliza o objecto *serverObj* para inicializar o servidor de vistas, através do método *SagaIni*. Desse ponto em diante, a comunicação com o servidor de vistas faz-se através do método *SagaEval*, que recebe comandos que o servidor de vistas interpreta e executa, devolvendo o resultado da execução. No fim da comunicação, o SagaWeb chama o método *SagaEnd*, que termina a execução do servidor de vistas.

Os passos finais na comunicação do SagaWeb com o servidor de vistas são: terminar o processo do SagaServer lançado e ordenar ao SagaGateway que liberte a porta ocupada pelo processo terminado.

4.4 Tratamento assíncrono de eventos e refrescamento parcial

Para além da geração de código de *Server Controls* dentro de ficheiros *ascx* (*User Controls*), o SagaWeb gera também *scripts* de tratamento de eventos.

Tanto o assincronismo do tratamento de eventos como o refrescamento parcial são garantidos pela utilização (transparente para o programador) de Ajax por parte da framework Gaia.

Durante o processamento assíncrono de um evento de alteração do valor de um campo, o Gaia impede que este seja alterado enquanto não chega a resposta do servidor, fazendo com que o controlo seja, momentaneamente, apenas de leitura.

Dependendo do tipo de controlo utilizado para representar um campo de vista, o SagaWeb gera o código de métodos que aguardam eventos de interface (alteração de valor de um controlo, clique num botão de *Zoom*,...) e originam chamadas ao método *ViewPropertyUpdatedMethod*[17] (Figura 34).

```
<gaia:TextBox AutoPostBack="true"  runat="server"
    ID="Cpcod_field" OnTextChanged="Cpcod_TextChanged"/>
<script runat="server">
    protected void Cpcod_TextChanged(object sender, EventArgs e)
    {
        ((sender as System.Web.UI.Control).Page as Gaia.MVC.Page).ViewPropertyUpdatedMethod(
                sender as System.Web.UI.Control,
                "cpostal.Cpcod",
                (sender as System.Web.UI.WebControls.TextBox).Text);
    }
</script>
```

Figura 34 - Definição de TextBox e *script* de tratamento do seu evento *TextChanged*

Na chamada ao método são passados como argumentos (para além do objecto que originou a chamada e que é indispensável ao lançamento de eventos, o *sender*) o identificador do campo e o seu valor. O método *ViewPropertyUpdatedMethod* verifica se existe algum *EventHandler* a subscrever o evento *ViewPropertyUpdated* e, em caso afirmativo, lança o evento (Figura 36).

[17] Este método é herdado por *Default.aspx* da classe *Gaia.MVC.Page*

```
public virtual void ViewPropertyUpdatedMethod(System.Web.UI.Control sender,
                                               string viewTypeDefinition, object value)
{
    if (ViewPropertyUpdated != null)
        ViewPropertyUpdated(sender, new ViewPropertyUpdatedEventArgs(viewTypeDefinition, value));
}
```

Figura 36 - Definição do método *ViewPropertyUpdated* da classe Gaia.MVC.Page

Para que a página principal (Default.aspx) possa ter conhecimento dos eventos de interface, apenas tem de criar um event handler e subscrever o evento ViewPropertyUpdated (Figura 35).

```
protected override void OnInit(EventArgs e)
{
    this.ViewPropertyUpdated +=
        new EventHandler<Gaia.MVC.ViewPropertyUpdatedEventArgs>(_Default_ViewPropertyUpdated);

    /* ..........*/
}
```

Figura 35 - Subscrição do evento *ViewPropertyUpdated*

O método *_Default_ViewPropertyUpdated* de *Default.aspx* dispõe, assim, da informação necessária à determinação do tipo de evento ocorrido, bastando-lhe determinar qual a acção a realizar como resposta ao mesmo (actualização de dados, abertura de vista, etc.). A aplicação constrói e envia para o servidor de vistas um conjunto de comandos na linguagem de regras e comandos que traduz a acção a realizar. O conjunto de comandos modifica o estado das vistas do servidor de vistas, e obtém o novo estado do mesmo, que é retornado à aplicação, com o objectivo de propagar as alterações efectuadas sobre as vistas do servidor de vistas para a UI.

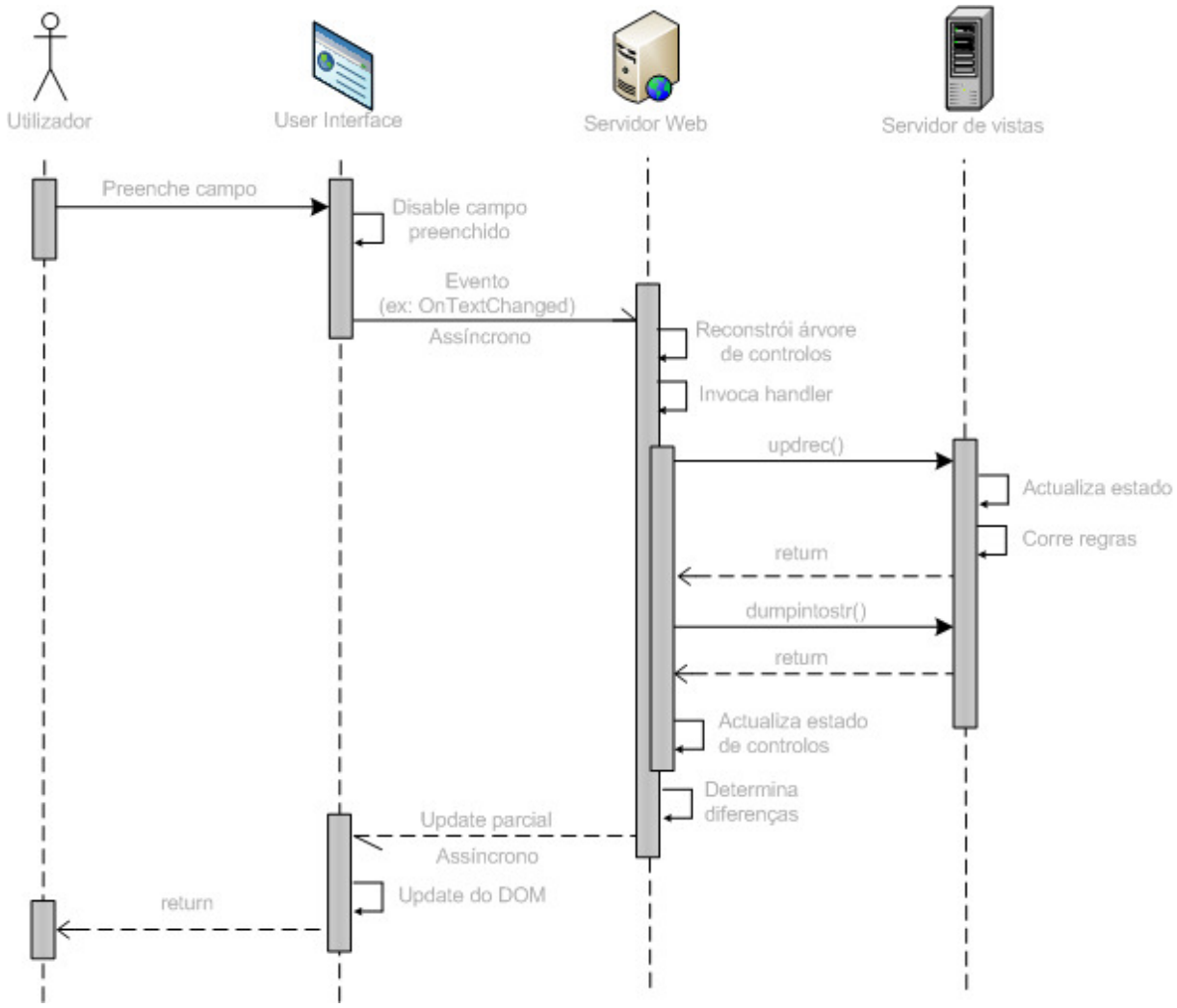


Figura 37 - Diagrama de sequência de alteração do valor de um campo na UI

A Figura 37 representa o diagrama de sequência dos eventos que advêm da alteração do valor de um campo na *User Interface*.

Após o preenchimento de um campo da UI por parte do utilizador, a interface faz o *disable* do mesmo, impedindo que o utilizador volte a alterá-lo enquanto não chega a resposta da aplicação Web. A UI lança de forma assíncrona o evento correspondente (por exemplo, *OnTextChanged*), do lado do servidor Web a framework .Net reconstrói a árvore de controlos (para poder tratar o evento originado) e invoca o handler do evento. Este handler é responsável por comunicar ao servidor de vistas a alteração de valor efectuada na interface, através do método *updrec()* do servidor de vistas. No lado do servidor de vistas, este actualiza o estado das vistas e corre o conjunto de regras relacionadas com a alteração efectuada na interface, devolvendo uma mensagem de sucesso à aplicação Web. De seguida, a aplicação pede o estado actualizado das vistas ao servidor de vistas e altera o estado dos seus controlos de acordo com a informação recebida. A framework Gaia determina as diferenças entre o estado actual dos controlos e o estado imediatamente anterior à ocorrência do evento e comunica assincronamente à UI o *update* parcial a efectuar ao seu *Document Object Model*, por forma a que o utilizador veja o(s) campo(s) actualizados.

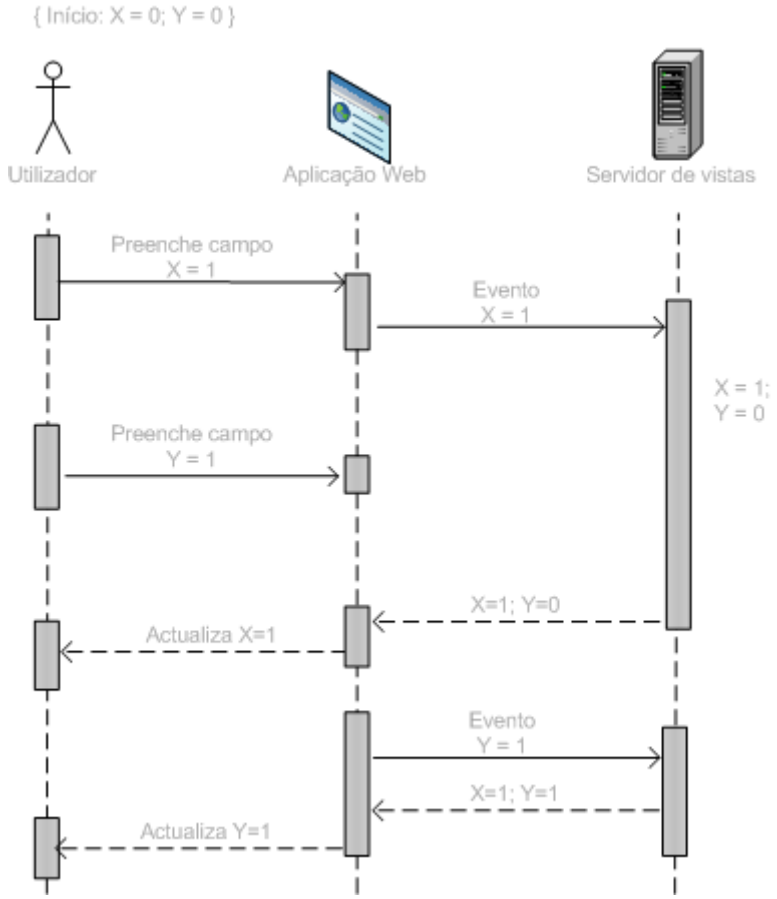


Figura 38 - Diagrama de sequência da alteração de valores de campos independentes na UI

A Figura 38 representa o diagrama de sequência da alteração de valores de campos independentes na UI.

Inicialmente existem dois campos na UI com valor igual a 0 (X e Y). O utilizador preenche o campo X com o valor 1 e, de imediato, preenche o campo Y com o valor 1, durante a comunicação da UI com a aplicação Web e antes da actualização da interface. A aplicação Web processa estas duas actualizações de uma forma sequencial, o que não levanta quaisquer problemas porque os campos são independentes.

4.5 Dificuldades

As principais dificuldades encontradas ao longo do desenvolvimento da nova camada de interface prenderam-se com vários factores, descritos nesta secção.

O facto de terem sido estudadas várias tecnologias Web, aliado à pouca (ou nenhuma, nalguns casos) experiência do autor na utilização das mesmas, revelou-se uma dificuldade na compreensão das especificidades de cada uma das tecnologias (esta dificuldade foi ultrapassada naturalmente com a evolução da experimentação).

Algumas soluções implementadas exigiram um grande envolvimento do Prof. João Pascoal Faria, pelo conhecimento que este possui da ferramenta original. Durante a implementação destas soluções, a equipa sentiu alguma dificuldade no progresso sempre que se verificava uma indisponibilidade prolongada do Prof. João Pascoal Faria.

As principais dificuldades sentidas a nível técnico tiveram a ver com a compreensão do modelo de programação do ASP.Net (principalmente quando utilizados controlos dinâmicos) e da manutenção do estado dos controlos da interface.

Também a integração do servidor de vistas com o SagaWeb se revelou um problema, tendo sido experimentadas algumas alternativas antes de chegar à presente solução, com a intervenção do Prof. João Pascoal Faria.

5 Apresentação e avaliação de resultados

Este capítulo destina-se à apresentação dos resultados obtidos no desenvolvimento do SagaWeb. Serão descritos a estrutura e o funcionamento geral da interface Web e exibidos alguns *screenshots* que ilustram o aspecto e funcionalidades da mesma. Estes exemplos, por vezes, far-se-ão acompanhar de imagens do Saga original no sentido de mostrar as diferenças (e também as semelhanças) entre ambas as ferramentas.

De seguida, serão avaliados os resultados, de acordo com as respostas dadas aos requisitos do projecto, nomeadamente ao nível da sua eficiência, interactividade, usabilidade e compatibilidade.

O capítulo será finalizado com uma perspectiva sobre os próximos passos a tomar no desenvolvimento do SagaWeb.

5.1 Estrutura da interface Web

A interface Web do SagaWeb (Figura 39) possui dois controlos que possibilitam ao utilizador abrir vistas das aplicações. O primeiro controlo é um AutoCompleter, que apresenta ao utilizador uma lista de sugestões (no formato *dropdown*) de vistas que este pode abrir. Esta lista é construída com base no texto introduzido pelo utilizador no controlo até então, sendo que o início dos nomes das vistas apresentados na lista coincide com o input do utilizador. O segundo controlo é do tipo TreeView e preenche os seus níveis de forma incremental, dependendo dos nós abertos pelo utilizador. Este controlo permite ao utilizador escolher uma vista para visualização de entre aquelas que são os pontos de entrada da aplicação, navegando ao longo da estrutura numa perspectiva do género *explorer*. *Screenshots* destes controlos podem ser encontrados na secção 5.2.

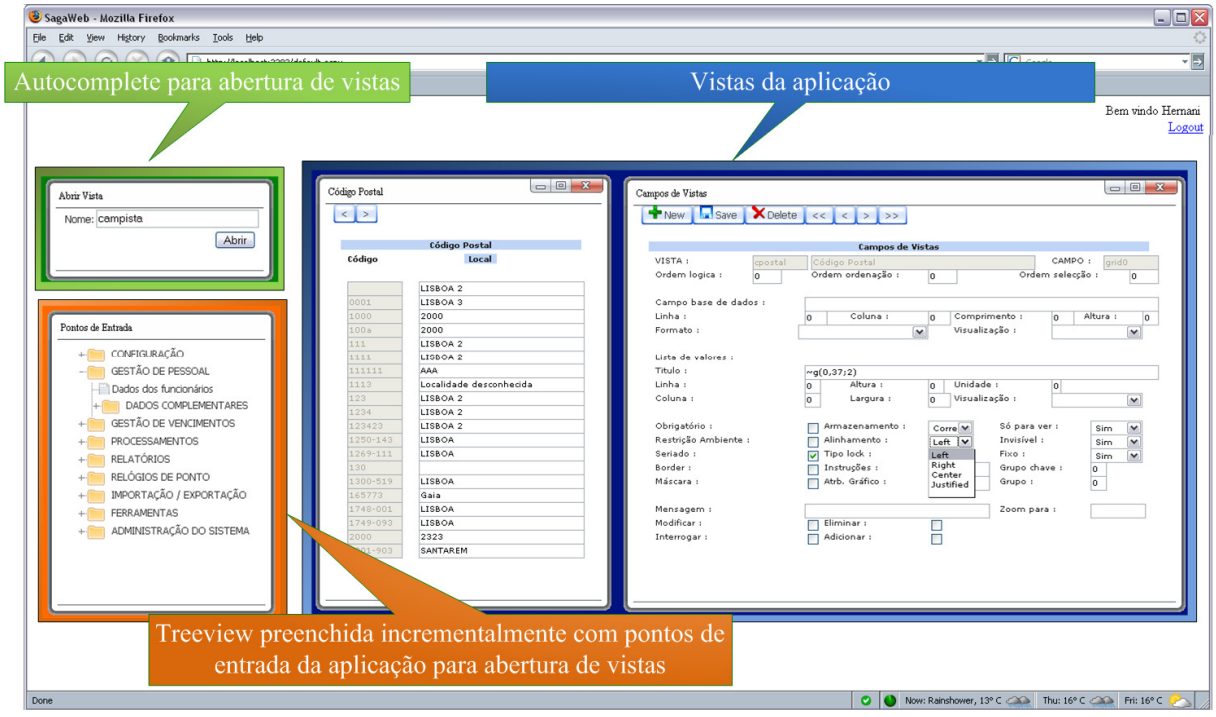


Figura 39 - Estrutura da interface SagaWeb

Cada vista aberta pelo utilizador é representada num controlo Window do Gaia, permitindo simular um ambiente multi-janela, característica familiar aos utilizadores finais das aplicações Saga. A diferença reside no facto de, agora, esta característica se encontrar presente numa aplicação Web.

Os controlos Window do Gaia permitem que o utilizador mova as vistas e as disponha no ecrã na posição que mais lhe convém. No entanto, o *layout* da interface foi desenhado de forma a que o utilizador disponha de uma área de trabalho na secção à direita dos controlos de abertura de vistas.

5.2 Funcionamento geral da interface Web

Na execução do Saga original, o utilizador depara-se com um ecrã inicial que lhe permite fazer a sua autenticação de forma a entrar na aplicação (Figura 40).

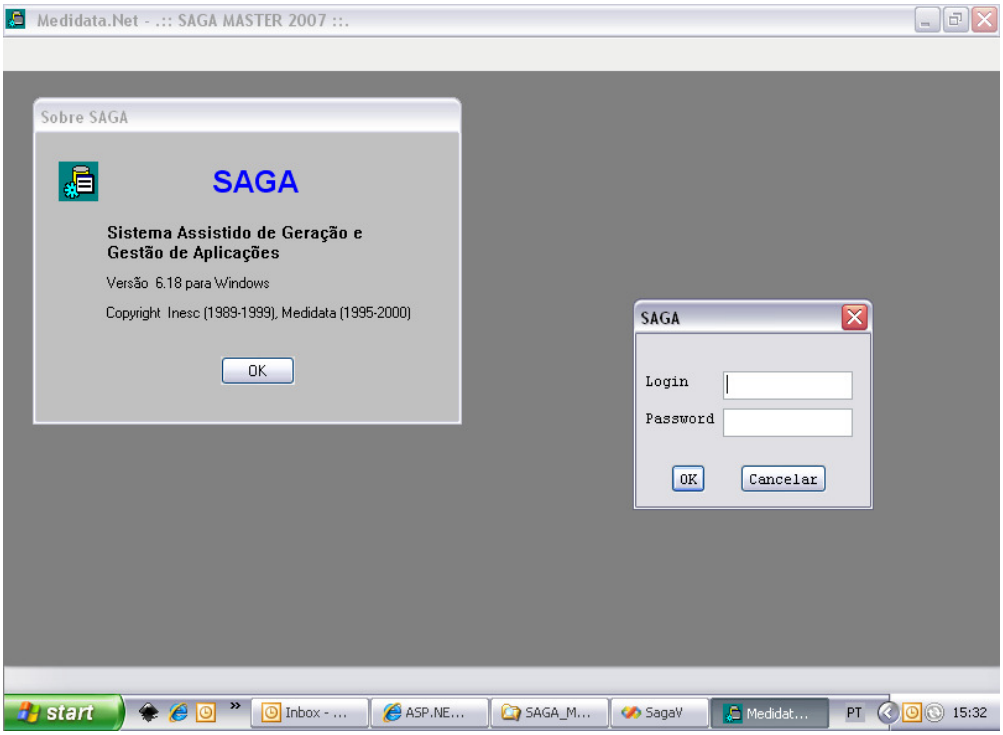


Figura 40 - Ecrã de login do Saga original

Ao iniciar o SagaWeb, o utilizador depara-se com um formulário onde, para além da sua autenticação, o utilizador pode fazer a selecção da aplicação à qual deseja aceder, pois a versão Web do Saga permite aceder a mais do que uma aplicação (Figura 41).

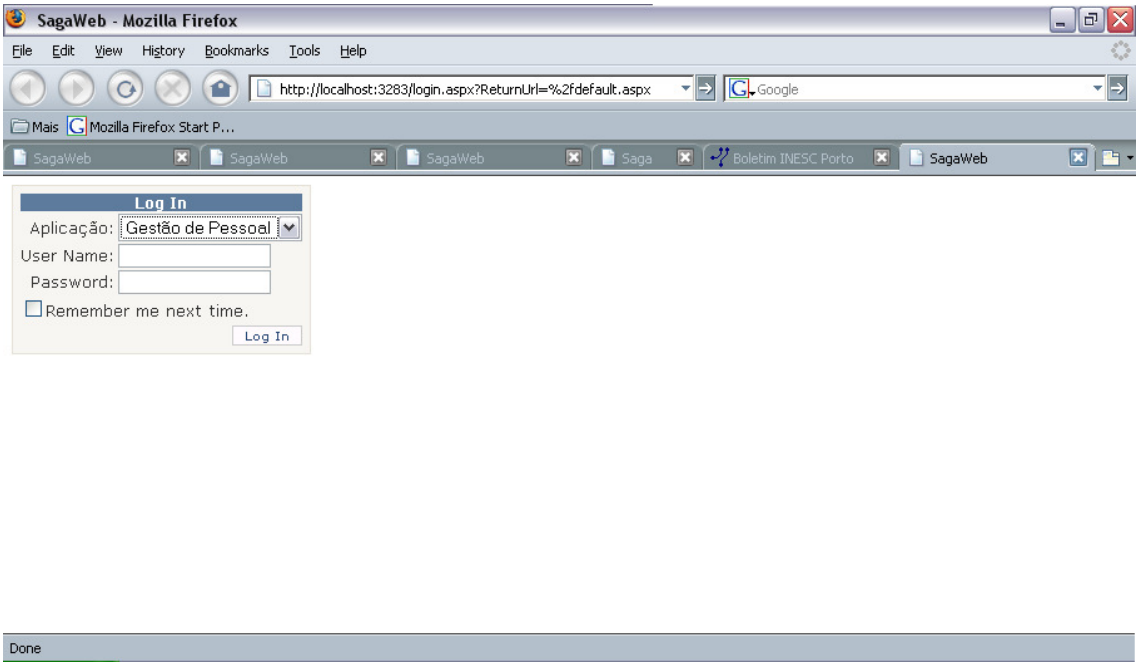


Figura 41 - Ecrã de login do SagaWeb

Ao entrar na aplicação seleccionada, o SagaWeb disponibiliza à esquerda da janela do browser dois controlos de abertura de vistas (AutoCompleter e TreeView). O AutoCompleter é um controlo que apresenta uma lista de vistas que depende do valor inserido até então (numa caixa de texto) pelo utilizador (Figura 42). Após seleccionar um dos valores da lista, o utilizador pressiona um botão que inicia a abertura da vista com o nome seleccionado. Esta funcionalidade é específica do SagaWeb, não se encontrando disponível na versão original do Saga.

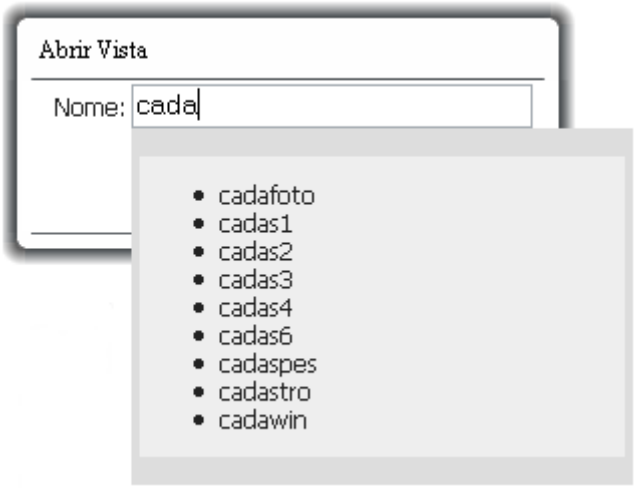


Figura 42 - AutoCompleter de abertura de vistas do SagaWeb

Para além de utilizar o AutoCompleter, o utilizador pode percorrer a TreeView, procurando as vistas que deseja abrir (Figura 43). O conteúdo da TreeView é 'construido' de uma forma incremental, pois apenas quando o utilizador abre um nó da árvore, o SagaWeb vai buscar o seu conteúdo. Desta forma, e porque a limitação de tráfego é importante no ambiente Web, reduz-se a quantidade de dados passada entre servidor e cliente.

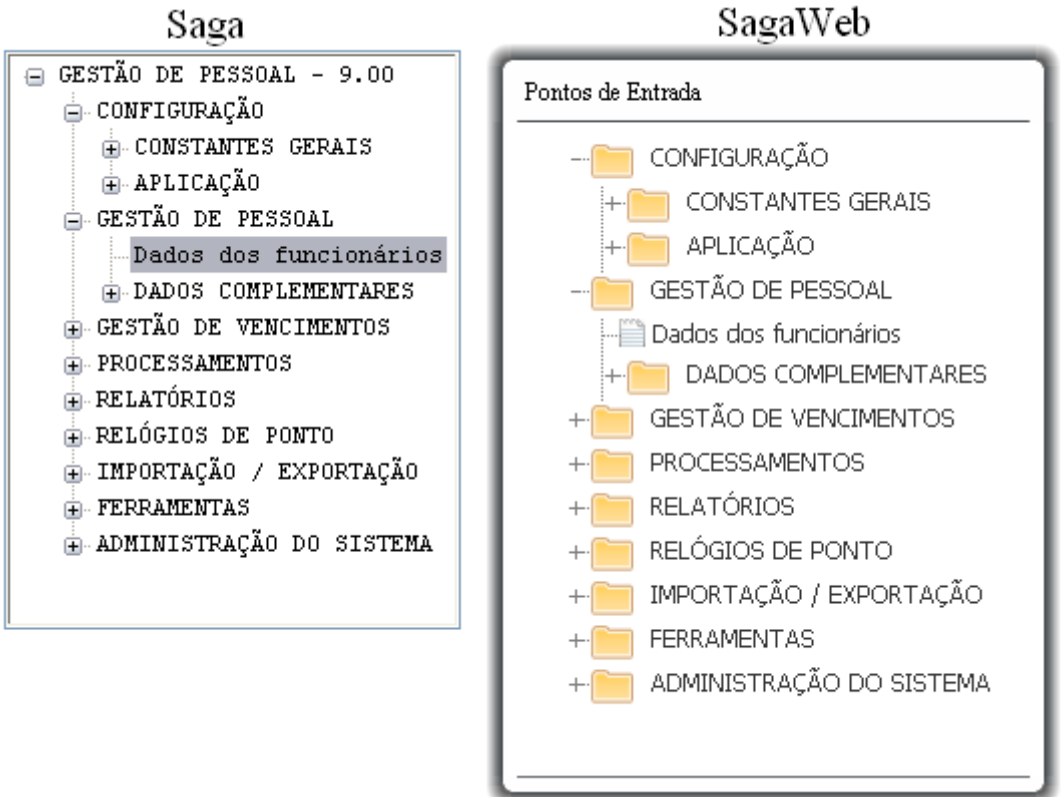


Figura 43 - Comparação de TreeView's de pontos de entrada (Saga e SagaWeb)

Cada vez que o utilizador abre uma vista, o SagaWeb abre um controlo Window na interface, atribuindo-lhe as características especificadas no ficheiro de definição de vistas correspondente. Deste modo, o *browser* oferece ao utilizador uma experiência semelhante à obtida no ambiente multi-janela, na Web (Figura 44).

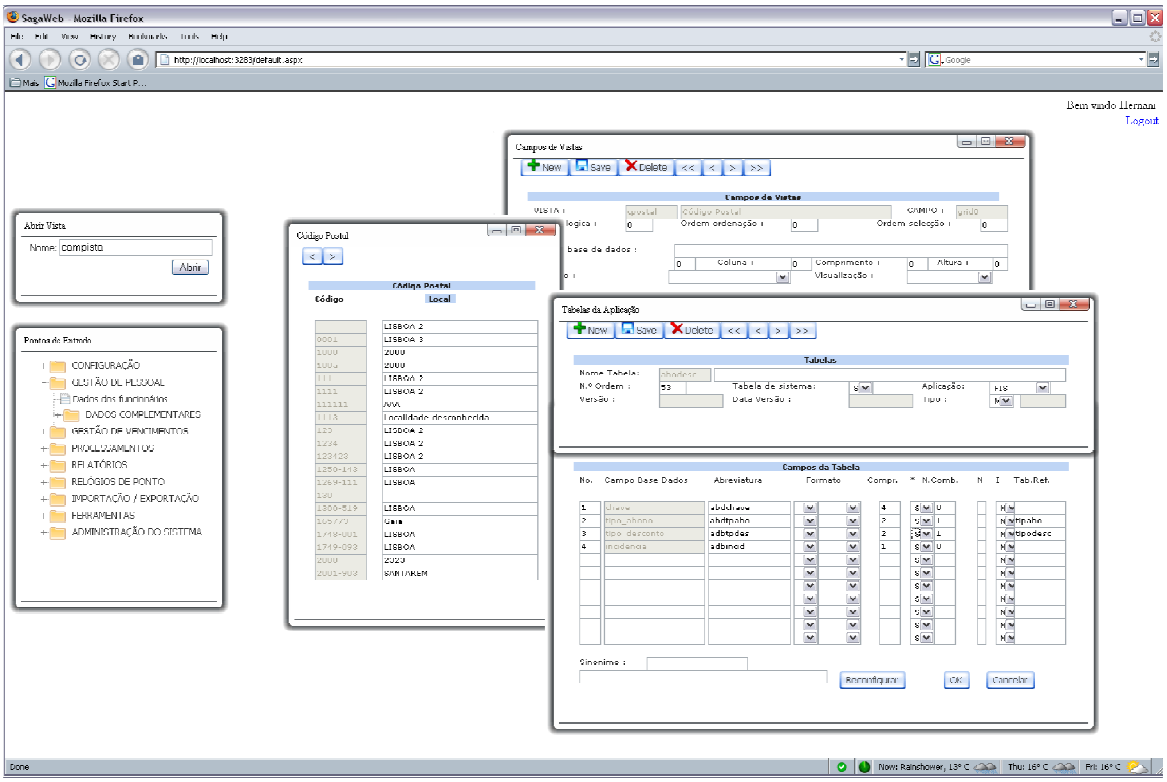


Figura 44 - Ambiente multi-janela do SagaWeb

Cada janela disponibiliza uma ToolBar/Menu (Figura 45) com opções de navegação e manipulação de registos. Através deste controlo, o utilizador pode criar um novo registo na vista, guardar as alterações efectuadas e eliminar o registo corrente. O utilizador pode, também, navegar directamente para o registo anterior e o seguinte, assim como para o primeiro e último registos de uma vista.

Figura 45 - ToolBar das vistas SagaWeb

A Figura 46 demonstra a forma como o SagaWeb apresenta os controlos que permitem actualmente ao utilizador abrir uma vista a partir de outra (*Zoom* e *Comadic*). Os comandos adicionais de abertura de vistas são representados por botões, enquanto os campos que têm *Zoom* associado são acompanhados por uma imagem de uma lupa que, quando clicado, abre a vista de *Zoom* correspondente.

Figura 46 - Exemplo de *Comadic* e *Zoom*

A tabela da Figura 47 representa os diversos tipos de controlos utilizados pelo SagaWeb para representar campos de vistas. O tipo de controlo utilizado depende do formato de dados e do tipo de representação visual definidos para o campo de vista em questão. Estes valores encontram-se definidos nos ficheiros de vistas compiladas do Saga.

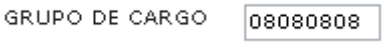
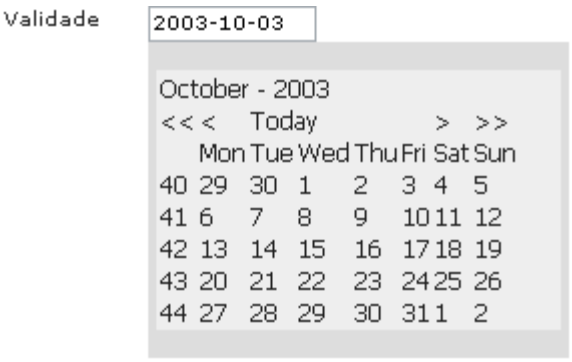
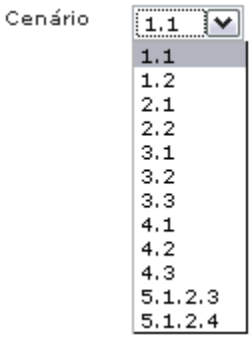
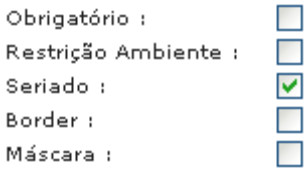
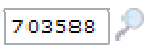
Tipos de controlo	Exemplo
TextBox	
Calendar	
DropDown	
CheckBox	
Image	

Figura 47 - Exemplos de Controlos do SagaWeb

5.3 Avaliação de resultados

Apesar do projecto SagaWeb se encontrar ainda no seu período de desenvolvimento, é possível retirar conclusões relativas aos objectivos que a equipa se propôs a atingir.

A solução implementada com o recurso ao projecto Gaia MVC consegue compatibilizar as aplicações desenvolvidas com o Saga original com o ambiente Web, sem que, até este ponto, tenha surgido a necessidade de alterar as aplicações. Através da utilização do controlo Window do Gaia, que apresenta uma janela no interior do *browser*, a aplicação pode simular o ambiente multi-janela com o qual os utilizadores estavam familiarizados no ambiente Windows. A nova camada Web disponibiliza operações CRUD sobre vistas simples, e a navegação entre registos das mesmas.

Após comparar o Saga original e o SagaWeb em termos de eficiência, constatou-se que não existem diferenças dignas de relevo, sendo satisfatória a rapidez do SagaWeb na execução das operações CRUD sobre as vistas de uma aplicação.

A framework Gaia permitiu evoluir o design das aplicações com a utilização de controlos 'ricos', e disponibilizando alguns *skins* nativos, que permitiram dar às aplicações existentes um aspecto ainda mais próximo daquele que as aplicações do Windows Vista possuem.

A tradução da definição de vistas para User Controls é um passo importante na 'componentarização' das vistas Saga, para que, no futuro, seja possível que uma vista isolada possa ser integrada em ambientes do tipo portal.

5.4 Trabalho futuro

Nesta secção será feita uma antevisão do trabalho a realizar nas fases seguintes do desenvolvimento do SagaWeb, nomeadamente ao nível da resolução de problemas, correcção de *bugs* e implementação de *features*.

Para que a equipa do SagaWeb obtenha um *feedback* mais completo do estado da UI e das funcionalidades implementadas, é necessário preparar uma versão do SagaWeb para ser acedida pelos utilizadores habituais do Saga. Ao compreender a impressão causada pelo novo ambiente Web nos indivíduos que o irão utilizar no dia-a-dia, torna-se mais fácil para a equipa do projecto determinar funcionalidades a implementar, melhorar ou retirar da ferramenta.

O SagaWeb apresenta, neste momento, um conjunto de problemas/bugs que a equipa do projecto se propôs a resolver. Alguns deles estão relacionados com a extensa utilização de AJAX. Um exemplo deste tipo de problemas é a perda de 'foco' dos controlos após o envio de um evento para processamento do lado do servidor através de Ajax.

Falta completar o tratamento de vistas em formato tabular, de forma a que o utilizador possa visualizar vários registos de uma vez, e de vistas complexas, com relações 'master-detail' e sincronização (transporte de dados) de vistas de 'zoom'.

A integração de vistas isoladas num ambiente do tipo portal é uma *feature* importante a implementar nas futuras iterações do desenvolvimento do projecto.

A equipa do projecto prevê que as regras de vistas que requerem *input* de utilizador tenham de ser processadas do lado do utilizador, o que exigirá a tradução de algumas regras para a linguagem Javascript.

6 Conclusões

O projecto 'Reengenharia de ferramenta CASE para ambiente 2.0' resultou numa aplicação ainda em desenvolvimento, mas que representa uma mais-valia a curto/médio prazo para a empresa cliente, a Medidata. O SagaWeb vai permitir à Medidata manter-se na vanguarda tecnológica e, em simultâneo, melhorar uma ferramenta legada, cujos serviços são essenciais, através da utilização de uma abordagem que representa um risco reduzido para a instituição.

A nível profissional, o projecto revelou-se extremamente rico, por um número variado de razões. O contacto com várias tecnologias e ferramentas permitiram ao autor desenvolver as suas aptidões técnicas, solidificar conhecimentos adquiridos durante o curso e adquirir novos conhecimentos. A interacção com indivíduos com grande experiência na área da Engenharia de Software, para além de um vasto leque de conhecimentos técnicos e tecnológicos foi uma fonte de conhecimento que, decerto, se revelará crucial em futuros projectos. Foi uma experiência interessante e enriquecedora poder interagir com profissionais de instituições nacionais e internacionais de renome, como a Medidata S.A., o INESC Porto, a Microsoft e a Gaiaware. O facto de poder terminar o Mestrado Integrado de Engenharia Informática e Computação com participação num projecto do INESC Porto é um marco importante na carreira profissional do autor.

A interacção pessoal com pessoas de diferentes culturas e posições na hierarquia das instituições, a cultura do INESC Porto de propiciar a socialização e, simultaneamente, impulsionar o desempenho individual e colectivo, permitiram ao autor desenvolver as suas capacidades sociais, estabelecer relações cordiais e de amizade e crescer como indivíduo enquanto evoluiu como profissional.

Referências e Bibliografia

[1] INESC Porto. *Apresentação do INESC Porto*. Obtido de http://www2.inescporto.pt/apresentacao

[2] Faria, João Pascoal. *Desenvolvimento de Aplicações em Saga – Sistema Assistido de Geração e Gestão de Aplicações*, Julho de 2002

[3] Abreu, Luís. *ASP.Net 2.0 – Curso Completo – 2ª Edição Revista,* Janeiro de 2006

[4] Infinities Loop. *Truly Understanding Dynamic Controls*. 25 de Agosto de 2006. Obtido de http://weblogs.ASP.Net/infinitiesloop/archive/2006/08/25/TRULY-Understanding-Dynamic-Controls-_2800_Part-1_2900_.aspx.

[5] Microsoft Corporation. *ObjectDataSource Web Server Control Overview*. Obtido de http://msdn2.microsoft.com/en-us/library/9a4kyhcx.aspx

[6] Wei, Coach K.. *AJAX: Asynchronous Java + XML?*. Obtido de http://www.developer.com/design/article.php/10925_3526681_1

[7] Hansen, Thomas. *7 minutes of Ajax [Video] that will change your life!*. 7 de Agosto de 2007. Obtido de http://ajaxwidgets.com/Blogs/thomas/7_minutes_of_ajax__video__that.bb

[8] Microsoft Corporation. *ScriptManager Control Overview*. Obtido de http://www.asp.net/ajax/documentation/live/overview/ScriptManagerOverview.aspx

[9] Microsoft Corporation. *Silverlight 1.0 Architecture.* Obtido de http://msdn2.microsoft.com/en-us/library/bb979825.aspx

[10] Webucator. *Silverlight Tutorial: Learn Silverlight*. Obtido de http://www.learn-silverlight-tutorial.com/

[11] Microsoft Corporation. *Silverlight Architecture*. Obtido de http://msdn2.microsoft.com/en-us/library/bb404713.aspx

[12] Microsoft Corporation, http://www.microsoft.com/silverlight/

[13] Informit. *Informit: The Ruby Programming Language.* Obtido de http://www.informit.com/articles/article.aspx?p=18225

[14] Azad, Kalid. http://betterexplained.com/wp-content/uploads/rails/mvc-rails.png

ANEXO A: Calendário de actividades

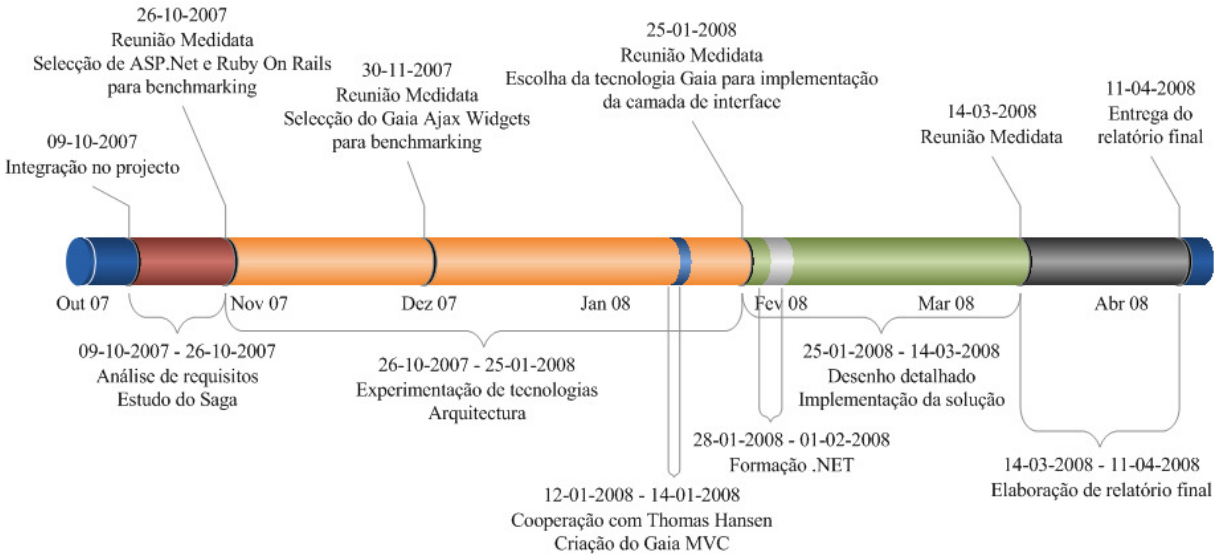


Figura 48 - Calendário de actividades do SagaWeb

O projecto 'SagaWeb' teve a sua reunião de arranque no início de Julho de 2007. No entanto, o ingresso do autor no INESC Porto para participação no projecto SagaWeb apenas aconteceu em Outubro, pelo que apenas serão descritas as actividades que ocorreram após essa data.

Outubro 2007

A 9 de Outubro, o autor iniciou a bolsa de investigação no INESC Porto e foi integrado de imediato na equipa do projecto.

Durante todo o mês de Outubro, e em simultâneo com o estudo da ferramenta Saga original, foi feita a análise de requisitos do projecto, seguida de um levantamento de padrões estruturais e comportamentais da interface para o utilizador. Após identificar os principais padrões, enumeraram-se tecnologias Web que pudessem constituir uma solução para o problema. Foi feita uma pré-selecção de tecnologias candidatas através da análise dos pontos fortes e fracos de cada uma das tecnologias enumeradas anteriormente. As tecnologias pré-seleccionadas foram o Microsoft ASP.Net e o Ruby On Rails.

A 26 de Outubro teve lugar uma reunião da equipa de projecto com a Medidata, José António Silva (*Architectural Advisor* da Microsoft) e Mário Lopes (Especialista em Ruby On Rails), na qual se debateram prós e contras das tecnologias candidatas e onde se decidiu avançar para um benchmarking das mesmas. O benchmarking, que tinha como objectivo encontrar a tecnologia de entre as candidatas que resolvesse o problema da migração da ferramenta Saga da forma mais satisfatória, ficou a cargo de Anabela Monteiro e do autor, Hernâni Fernandes.

Novembro 2007

Durante todo o mês de Novembro, procedeu-se ao estudo e exploração das tecnologias ASP.Net, pelo autor, e Ruby On Rails, por Anabela Monteiro, através do desenvolvimento de 2 soluções simples.

A 30 de Novembro ocorreu nova reunião com a Medidata, na qual se apresentaram as ilações tiradas na primeira fase da experimentação. Nesta altura, o Ruby On Rails apresentava mais pontos a seu favor do que o ASP.Net, principalmente pela sua facilidade de desenvolvimento e pela utilização do padrão MVC. Os controlos 'inteligentes' do ASP.Net (GridView e DetailsView) revelaram-se pouco flexíveis em termos de estrutura de apresentação e tratamento de eventos.

Durante a reunião, José António Silva apresentou uma *framework* AJAX de uma empresa Norueguesa (Gaiaware), o Gaia Ajax Widgets, que, à primeira vista, apresentava como principais pontos fortes o facto de abstrair o Javascript necessário ao processamento de eventos e aos efeitos visuais da aplicação e a capacidade de simular um ambiente multi-janela dentro de uma única janela de browser (ambiente semelhante ao que os utilizadores do Saga utilizam nas suas máquinas actualmente) (Figura 49).

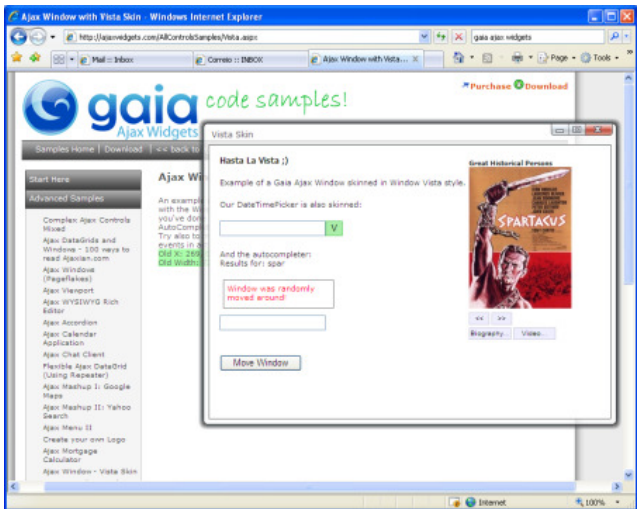


Figura 49 - Ambiente multi-janela de Gaia Ajax Widgets

Era importante determinar se o seu modelo de programação se aproximava do utilizado no desenvolvimento da ferramenta original (utilização da Win32 API, com gestão de controlos, menus, eventos). Se tal se verificasse, a migração seria mais fácil e menos disruptiva.

Dado que a solução ASP.Net estava em desvantagem em relação ao Ruby On Rails, decidiu-se abandoná-la e dar início à experimentação da framework Gaia, com o desenvolvimento de uma nova solução, a cargo do autor.

Dezembro 2007

Em Dezembro foram feitos o estudo e exploração paralelos das tecnologias Gaia e Ruby On Rails.

Janeiro 2008

Depois de encontrar algumas dificuldades na experimentação do Gaia, a equipa achou por bem fazer um breve estudo da tecnologia Silverlight, da Microsoft. Este estudo revelou rapidamente que o Silverlight (versão 1.0) não é adequado à construção de aplicações Web com formulários.

Nos dias 12, 13 e 14 de Janeiro, deslocou-se ao Porto um especialista da Gaiaware, Thomas Hansen, com o intuito de ajudar a equipa da SagaWeb na resolução de alguns problemas ao nível da utilização do Gaia para a geração dinâmica de controlos. Após tomar conhecimento do estado do projecto e dos problemas com que a equipa se havia deparado, Thomas Hansen lançou as bases de um projecto (que irá ser desenvolvido e comercializado no futuro pela Gaiaware) ao qual chamou 'Gaia MVC'. Este projecto, ainda que numa fase preliminar do seu desenvolvimento, resolveu alguns problemas pendentes críticos da solução implementada com o recurso ao Gaia, como o tratamento assíncrono de eventos. Foram resolvidos, também, problemas relativos ao ciclo de vida do servidor de vistas do Saga.

No dia 25 de Janeiro a equipa do projecto reuniu com a Medidata com o intuito de analisar os resultados da experimentação de tecnologias e seleccionar uma única tecnologia de entre as 2 em análise para a continuação do desenvolvimento do SagaWeb.

Com base nos resultados obtidos até esta data, e tendo em conta os requisitos do projecto, a solução que utiliza a *framework* Gaia Ajax Widgets combinada com o projecto (ainda em versão muito preliminar, mas com resultados satisfatórios) Gaia MVC levou vantagem sobre o Ruby On Rails. Desta forma, ficou decidido que, desta data em diante, o autor e Anabela Monteiro concentrariam esforços na solução 'Gaia'.

De dia 28 de Janeiro a 1 de Fevereiro, ocorreu na Faculdade de Engenharia da Universidade do Porto (FEUP) uma formação ASP.Net patrocinada pela Microsoft, leccionada por docentes do Centro de Cálculo do Instituto Superior de Engenharia de Lisboa. A formação foi frequentada pelo autor e por Anabela Monteiro e João Pascoal Faria.

Fevereiro 2008

O mês de Fevereiro foi dedicado ao desenvolvimento do SagaWeb com a framework Gaia. Foi estabelecido um conjunto de funcionalidades que se deveriam implementar até à data da reunião seguinte (14 de Março). Até à reunião seguinte, o SagaWeb deveria ser capaz de tratar uma vista isolada, permitir efectuar operações CRUD[18], navegar através dos registos de uma vista, apresentar diferentes tipos de controlos Web e fazer 'Zoom' de um campo de uma vista para uma outra vista.

[18] **CRUD** – **C**reate, **R**etrieve, **U**pdate and **D**elete – Criar, Obter, Actualizar e Apagar, as opções básicas de manipulação de dados

Março 2008

O desenvolvimento do SagaWeb (Figura 50) continuou durante o início do mês de Março, tendo-se prolongado até dia 14, altura em que teve lugar uma nova reunião com a Medidata. Esta reunião teve como propósito manter a Medidata ao corrente do desenvolvimento do SagaWeb, pelo que foram apresentados o estado da interface e as funcionalidades implementadas até então.

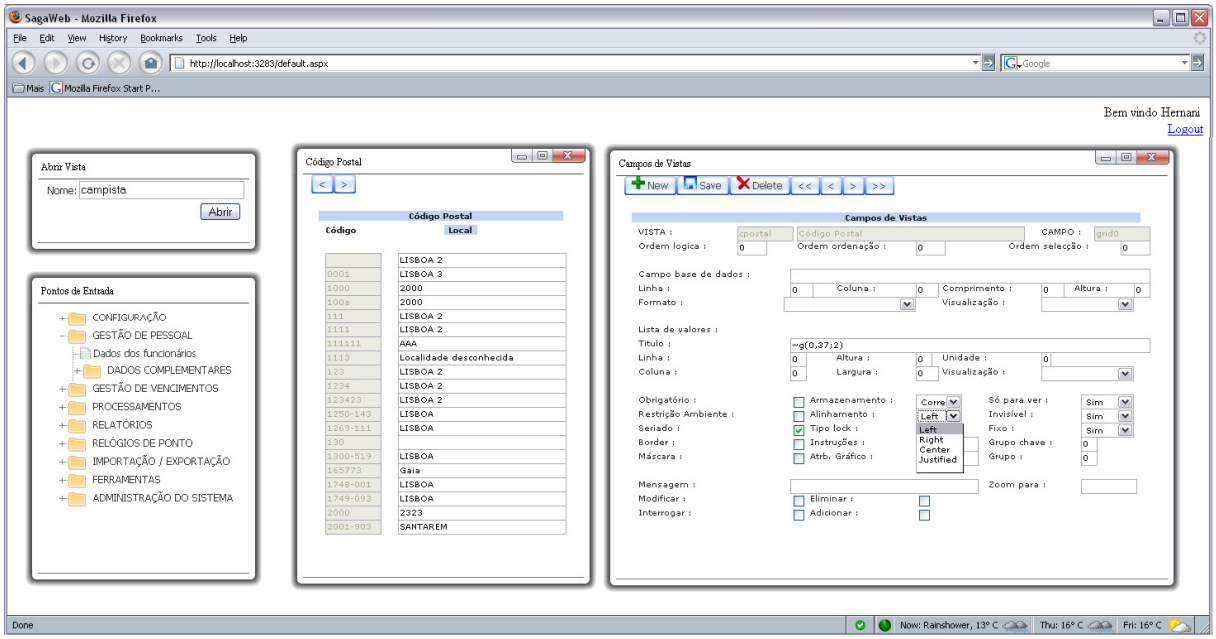


Figura 50 - Versão actual do SagaWeb

Foi estabelecido que, desta data em diante, o autor iria concentrar-se na elaboração do seu relatório final de projecto, ficando a cargo de Anabela Monteiro a continuação do desenvolvimento do SagaWeb.

ANEXO B: Exemplo de página ASP.Net (aspx)

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="myPage.aspx.cs"
    Inherits="SagaWeb.myPage" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>my ASPX page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        Nome:<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <br />
        Idade:<asp:TextBox ID="TextBox2" runat="server" style="margin-bottom: 0px"
            ontextchanged="TextBox2_TextChanged"></asp:TextBox>
        <asp:Button ID="Button1" runat="server" Text="Button"/>
    </div>
    </form>
</body>
</html>
```

Figura 51 - Exemplo de página ASPX

ANEXO C: Exemplo de ficheiro de vista compilada do Saga (cadastro)

vistas|nome|tabela|n_max_reg_sel|scroll_margem|over_scroll|descricao|pode_modificar|pode_eliminar|pode_adicionar|pode_procurar|posicao_x|posicao_y|largura|altura|instrucoes_sos|n_max_reg_mem|margem_p_topo|margem_p_fundo|passo_entre_reg|orientacao_passo|vista_sistema|tem_border|tem_linha_status|tipo_lock|ajustavel|so_no_topo|modo_de_entrada|ponto_de_entrada|regua|pode_rolar|pode_mover|pode_esticar|pode_fechar|tipo_grafico|invisivel|formato|n_reg_sel_incr|auto_gravar_reg

vistas|cadastro|cadastro|0|0|0|Dados dos funcionários|S|S|S|S|0|3|124|23||0|0|0|23|V|N|N|S|N|N||B|||S|N|N|N||||0|

permgrp|grupo|vista|modificar|adicionar|interrogar|eliminar

permgrp|SYS2|cadastro|S|S|S|S

permgrp|VEN|cadastro|S|S|S|S

vistas|nome|tabela|n_max_reg_sel|scroll_margem|over_scroll|descricao|pode_modificar|pode_eliminar|pode_adicionar|pode_procurar|posicao_x|posicao_y|largura|altura|instrucoes_sos|n_max_reg_mem|margem_p_topo|margem_p_fundo|passo_entre_reg|orientacao_passo|vista_sistema|tem_border|tem_linha_status|tipo_lock|ajustavel|so_no_topo|modo_de_entrada|ponto_de_entrada|regua|pode_rolar|pode_mover|pode_esticar|pode_fechar|tipo_grafico|invisivel|formato|n_reg_sel_incr|auto_gravar_reg

vistas|cadastro|cadastro|0|0|0|Dados dos funcionários|S|S|S|S|0|3|124|23||0|0|0|23|V|N|N|S|N|N||B|||S|N|N|N||||0|

campista|vista|nome|comprimento|posicao_x|posicao_y|titulo|posicao_titulo_x|posicao_titulo_y|ordem_logica|campo_base_dados|obrigatorio|invisivel|so_para_ver|fixo|ordem_ordenacao|grupo_chave|restr_p_ambiente|instrucoes_sos|seriado|formato|grupo|lista_valores|altura|border|tipo_visual|tipo_lock|mensagem|ordem_seleccao|armazenamento|alinhamento|mascarado|atributo_grafico|altura_titulo|largura_titulo|tipo_visual_tit|unidade|zoom|formato_bd|comprimento_bd

campista|cadastro|grid0|0|0|0|~g(0,123;8)|0|0|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|prmp001|0|0|0|VÍNCULO|71|1|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|prmp002|0|0|0|UNI.ORGÂNICA|71|2|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|prmp003|0|0|0|Validade|45|21|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|prmp004|0|0|0|(|27|22|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|prmp005|0|0|0|)|31|22|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|prmp006|0|0|0|Nº Apólice Seguro|48|16|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|prmp007|0|0|0|Quadro ou Contratado ?|1|16|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|prmp008|0|0|0|~r Bilhete de Identidade ~r|1|19|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|prmp009|0|0|0|Habilitações complementares|63|12|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|prmp010|0|0|0|Nº INFORMÁTICO|13|1|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|prmp011|0|0|0|~r Nascimento ~r|69|19|0|||||0|0|||||0||0|||||0|||||6|0||0|||

campista|cadastro|prmp012|0|0|0|Nº CONTRIBUINTE|38|1|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|prmp013|0|0|0|~r ALTERAÇÕES ~r|108|4|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|prmp014|0|0|0|~r Carta de Condução ~r|45|19|0|||||0|0|||||0||0|||||0|||||4|0||0|||

campista|cadastro|prmp015|0|0|0|NOME|13|2|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|prmp016|0|0|0|GRUPO DE CARGO|13|3|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|prmp017|0|0|0|~r Situação na Entidade ~r|0|13|0|||||0|0|||||0||0|||||0|||||5|0||0|||

campista|cadastro|prmp018|0|0|0|ACTIVO ?|13|4|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|prmp019|0|0|0|~r Informação diversa ~r|0|9|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|prmp020|0|0|0|~r DADOS DO FUNCIONÁRIO ~r|13|0|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|prmp021|0|0|0|~r~b ~b~r|0|7|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|prmp022|0|0|0|Funcionário ?|1|17|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|prmp023|0|0|0|Subscrições|48|17|0|||||0|0|||||0||0|||||0|||||0|0||0|||

campista|cadastro|Foto|12|0|0||0|0|1|cadastro.foto|||S|S|0|0||||STRING|0||5||bitmapbutton|||0|||||0|0||0||STRING|50

campista|cadastro|Cdnum|5|28|1||0|0|2|cadastro.numero||N|||1|0|S|||NUMERIC|0||0|||||0|||||0|0||0||NUMERIC|5

campista|cadastro|Contrib|11|54|1||0|0|3|cadastro.n_contribuinte|||||0|0||||STRING|0||0|||||0|||||0|0||0|contrib|STRING|11

campista|cadastro|cadvinc|2|84|1||0|0|4|cadastro.vinculo|||||0|0||||STRING|0||0|||||0|||||0|0||0|vinculo|STRING|2

campista|cadastro|Vincdes|30|89|1||0|0|5|vinculo.descricao|||||0|0||||STRING|0||0|||||0|||||0|0||0||STRING|30

campista|cadastro|Nome|50|18|2||0|0|6|contrib.nome|||||0|0||||STRING|0||0|||||0|||||0|0||0||STRING|80

campista|cadastro|caduni|10|84|2||0|0|7|uniorg.codigo|||||0|0||||STRING|0||0|||||0|||||0|0||0||STRING|10

campista|cadastro|Uninome|25|97|2||0|0|8|uniorg.nome|||||0|0||||STRING|0||0|||||0||||0|0||0||STRING|100

campista|cadastro|Grupo|8|28|3||0|0|9|cadastro.grupo_cargo|||||0|0||||STRING|0||0|||||0|||||0|0||0|grupo|STRING|8

campista|cadastro|Grup|20|39|3||0|0|10|grupo.descricao|||||0|0||||STRING|0||0|||||0||||0|0||0||STRING|20

campista|cadastro|Carr|20|60|3||0|0|11|grupo.descr_carreira|||||0|0||||STRING|0||0|||||0|||||0|0||0||STRING|20

campista|cadastro|Cate|20|81|3||0|0|12|grupo.descr_categoria|||||0|0||||STRING|0||0|||||0|||||0|0||0||STRING|20

campista|cadastro|Proc|1|22|4||0|0|13|cadastro.processa_ordenad|||||0|0||||STRING|0||0|||||0|||||0|0||0||STRING|1

campista|cadastro|Login|8|114|5|Login|108|5|14|cadastro.login|||S||0|0||||STRING|0||0|||||0|||||0|0||0||STRING|8

campista|cadastro|Datal|10|114|6|Data|108|6|15|cadastro.data_login|||S||0|0||||LDATE|0||0|||||0|||||0|0||0||LDATE|10

campista|cadastro|cadpai|45|13|10|Filiação Pai|0|10|16|cadastro.filiacao_pai|N|N|N|N|0|0|N||N|STRING|0||0|||||0|||||0|0||0||STRING|65

campista|cadastro|cadmae|45|76|10|Filiação Mae|63|10|17|cadastro.filiacao_mae|N|N|N|N|0|0|N||N|STRING|0||0|||||0|||||0|0||0||STRING|65

campista|cadastro|Cdproc|10|13|11|Nº Processo|0|11|18|cadastro.processo|N|N|N|N|0|0|N||N||0||0|||||0|||||0|0||0||STRING|10

campista|cadastro|Cdsexo|1|31|11|Sexo|26|11|19|cadastro.sexo|S|N|N|N|0|0|N||N|STRING|0||0|||||0|||||0|0||0||STRING|1

campista|cadastro|Cdgrpsan|6|52|11|Grupo Sanguíneo|36|11|20|cadastro.grupo_sanguineo|N|N|N|N|0|0|N||N|STRING|0||0|||||0|||||0|0||0||STRING|6

campista|cadastro|cadhab|6|76|11|Habilitações|63|11|21|cadastro.codigo_habilitac|N|N|N|N|0|0|N||N|STRING|0||0|||||0|||||0|0||0|tiphabi|STRING|6

campista|cadastro|Habidesc|30|85|11||0|0|22|tiphabi.descricao|||||0|0||||STRING|0||0|||||0|||||0|0||0||STRING|40

campista|cadastro|Hab|1|91|12||0|0|23||||||0|0||||STRING|0|{'Sim','Nao',''}|0||checkbox|||0|||||0|0||0|habilit||

zoomcmp|zmcampo_cpvvista|zmcampo_cpvnome|vista|pode_modificar|pode_eliminar|pode_adicionar|pode_procurar

zoomcmp|cadastro|Hab|habilit||||

campista|vista|nome|comprimento|posicao_x|posicao_y|titulo|posicao_titulo_x|posicao_titulo_y|ordem_logica|campo_base_dados|obrigatorio|invisivel|so_para_ver|fixo|ordem_ordenacao|grupo_chave|restr_p_ambiente|instrucoes_sos|seriado|formato|grupo|lista_valores|altura|border|tipo_visual|tipo_lock|mensagem|ordem_seleccao|armazenamento|alinhamento|mascarado|atributo_grafico|altura_titulo|largura_titulo|tipo_visual_tit|unidade|zoom|formato_bd|comprimento_bd

campista|cadastro|Cadsitu|2|10|14|Situação|1|14|24|cadastro.tipo_situacao|S|N|N|N|0|0|S||N|STRING|0||0|||||0|||||0|0||0|tisitu|STRING|2

campista|cadastro|Sit|30|15|14||0|0|25|tisitu.descricao|N|N|N|s|0|0|N||N|STRING|0||0||||0|||||0|0||0||STRING|30

campista|cadastro|Cdloctra|0|66|14|Local Trabalho|48|14|26|cadastro.local_trabalho|N|N|N|N|0|0|N||N|STRING|0||0|||||0|||||0|0||0|loc_trab|STRING|4

campista|cadastro|Local|40|73|14||0|0|27|loc_trab.local|||||0|0||||STRING|0||0|||||0|||||0|0||0||STRING|40

campista|cadastro|Quadr|2|10|15|Quadro|1|15|28|cadastro.quadro|S|N|N|N|0|0|N||N|STRING|0||0|||||0|||||0|0||0|tip_quad|STRING|2

campista|cadastro|Quadescr|14|15|15||12|19|29|tip_quad.descricao|||||0|0||||STRING|0||0|||||0|||||0|0||0||STRING|20

campista|cadastro|cadqual|3|66|15|Profissão|48|15|30|cadastro.codigo_qualifica|N|N|N|0|0|N||N|STRING|0||0|||||0|||||0|0||0|tipquali|STRING|3

campista|cadastro|Profdesc|23|72|15||0|0|31|tipquali.descricao|||||0|0||||STRING|0||0||||0|||||0|0||0||STRING|25

campista|cadastro|Cdct|1|24|16||0|0|32|cadastro.contratado|||||0|0||||STRING|0||0|||||0|||||0|0||0||STRING|1

campista|cadastro|Des|20|26|16||0|0|33||||||0|0||||STRING|0||0|||||0|||||0|0||0|||

campista|cadastro|Apolice|15|66|16||0|0|34|cadastro.numero_apolice|||||0|0||||STRING|0||0|||||0|||||0|0||0|seguros|STRING|15

campista|cadastro|Apolde|30|84|16||0|0|35|seguros.ramo|||||0|0||||STRING|0||0|||||0|||||0|0||0||STRING|30

campista|cadastro|Orgeco1|3|15|17||0|0|36||||n||0|0||||STRING|0|multilist({'S','Sim','N','Não'},Orgeco,Orgeco1)|0|||||0|||||0|0||0|||

campista|cadastro|Orgeco|1|24|17||0|0|37|cadastro.orgeco||S|||0|0||||STRING|0||0|||||0|||||0|0||0||STRING|10

campista|cadastro|Subscr|50|66|17||0|0|38||||n||0|0||||STRING|0|{foreach %subscr1 where %subscr1.numero==Cdnum do 'Nº '..%subscr1.num_subscritor..' - '..%subscr1.sigla..' - '..%entext.nome[%subscr1.sigla]}|0|||||0|||||0|0||0|subscr1||

zoomcmp|zmcampo_cpvvista|zmcampo_cpvnome|vista|pode_modificar|pode_eliminar|pode_adicionar|pode_procurar

zoomcmp|cadastro|Subscr|subscr1||||

campista|vista|nome|comprimento|posicao_x|posicao_y|titulo|posicao_titulo_x|posicao_titulo_y|ordem_logica|campo_base_dados|obrigatorio|invisivel|so_para_ver|fixo|ordem_ordenacao|grupo_chave|restr_p_ambiente|instrucoes_sos|seriado|formato|grupo|lista_valores|altura|border|tipo_visual|tipo_lock|mensagem|ordem_seleccao|armazenamento|alinhamento|mascarado|atributo_grafico|altura_titulo|largura_titulo|tipo_visual_tit|unidade|zoom|formato_bd|comprimento_bd

campista|cadastro|Cdbi|9|4|20|Nº|1|20|39|cadastro.numero_bi|S|N|N|N|0|0|N||N|NUMERIC|0||0|||||0|||||0|0||0||NUMERIC|9

campista|cadastro|Cddatbi|10|30|20|Emissão|21|20|40|cadastro.data_bi|S|N|N|N|0|0|N||N|DATE|0||0|||||0|||||0|0||0|cal|LDATE|10

zoomcmp|zmcampo_cpvvista|zmcampo_cpvnome|vista|pode_modificar|pode_eliminar|pode_adicionar|pode_procurar

zoomcmp|cadastro|Cddatbi|cal||||

campista|vista|nome|comprimento|posicao_x|posicao_y|titulo|posicao_titulo_x|posicao_titulo_y|ordem_logica|campo_base_dados|obrigatorio|invisivel|so_para_ver|fixo|ordem_ordenacao|grupo_chave|restr_p_ambiente|instrucoes_sos|seriado|formato|grupo|lista_valores|altura|border|tipo_visual|tipo_lock|mensagem|ordem_seleccao|armazenamento|alinhamento|mascarado|atributo_grafico|altura_titulo|largura_titulo|tipo_visual_tit|unidade|zoom|formato_bd|comprimento_bd

campista|cadastro|Cdtipcrt|1|60|20|Tipo de carta|45|20|41|cadastro.tipo_carta_cond|N|N|N|N|0|0|N||N|STRING|0||0|||||0|||||0|0||0||STRING|2

campista|cadastro|Cddatnas|10|74|20|Data|69|20|42|cadastro.data_nascimento|S|N|N|N|0|0|N||N|DATE|0||0|||||0|||||0|0||0|cal|LDATE|10

zoomcmp|zmcampo_cpvvista|zmcampo_cpvnome|vista|pode_modificar|pode_eliminar|pode_adicionar|pode_procurar

zoomcmp|cadastro|Cddatnas|cal||||

campista|vista|nome|comprimento|posicao_x|posicao_y|titulo|posicao_titulo_x|posicao_titulo_y|ordem_logica|campo_base_dados|obrigatorio|invisivel|so_para_ver|fixo|ordem_ordenacao|grupo_chave|restr_p_ambiente|instrucoes_sos|seriado|formato|grupo|lista_valores|altura|border|tipo_visual|tipo_lock|mensagem|ordem_seleccao|armazenamento|alinhamento|mascarado|atributo_grafico|altura_titulo|largura_titulo|tipo_visual_tit|unidade|zoom|formato_bd|comprimento_bd

campista|cadastro|Cdfgdist|2|99|20|Distrito|89|20|43|cadastro.distrito_nasc|||||0|0||||STRING|0||0|||||0|||||0|0||0|distrito|STRING|2

campista|cadastro|Nasdis|20|104|20||0|0|44|distrito.designacao[Cdfgdist]|||||0|0||||STRING|0||0|||||0|||||0|0||0||STRING|50

campista|cadastro|Cdarqbi|10|9|21|Arquivo|1|21|45|cadastro.arquivo_bi|S|N|N|N|0|0|N||N|STRING|0||0|||||0|||||0|0||0||STRING|10

campista|cadastro|Cdvalbi|10|30|21|Validade|21|21|46|cadastro.validade_bi|S|N|N|N|0|0|N||N|DATE|0||0|||||0|||||0|0||0|cal|LDATE|10

zoomcmp|zmcampo_cpvvista|zmcampo_cpvnome|vista|pode_modificar|pode_eliminar|pode_adicionar|pode_procurar

zoomcmp|cadastro|Cdvalbi|cal||||

campista|vista|nome|comprimento|posicao_x|posicao_y|titulo|posicao_titulo_x|posicao_titulo_y|ordem_logica|campo_base_dados|obrigatorio|invisivel|so_para_ver|fixo|ordem_ordenacao|grupo_chave|restr_p_ambiente|instrucoes_sos|seriado|formato|grupo|lista_valores|altura|border|tipo_visual|tipo_lock|mensagem|ordem_seleccao|armazenamento|alinhamento|mascarado|atributo_grafico|altura_titulo|largura_titulo|tipo_visual_tit|unidade|zoom|formato_bd|comprimento_bd

campista|cadastro|Cddatval|10|54|21||0|0|47|cadastro.data_validade|||||0|0||||LDATE|0||0|||||0|||||0|0||0|cal|LDATE|10

zoomcmp|zmcampo_cpvvista|zmcampo_cpvnome|vista|pode_modificar|pode_eliminar|pode_adicionar|pode_procurar

zoomcmp|cadastro|Cddatval|cal||||

campista|vista|nome|comprimento|posicao_x|posicao_y|titulo|posicao_titulo_x|posicao_titulo_y|ordem_logica|campo_base_dados|obrigatorio|invisivel|so_para_ver|fixo|ordem_ordenacao|grupo_chave|restr_p_ambiente|instrucoes_sos|seriado|formato|grupo|lista_valores|altura|border|tipo_visual|tipo_lock|mensagem|ordem_seleccao|armazenamento|alinhamento|mascarado|atributo_grafico|altura_titulo|largura_titulo|tipo_visual_tit|unidade|zoom|formato_bd|comprimento_bd

campista|cadastro|A_fgconc|2|99|21|Concelho|89|21|48|cadastro.cdfregna_fgconc|N|N|N|N|0|0|S||N||0||0|||||0|||||0|0||0|concelho|STRING|2

campista|cadastro|Disconc|20|104|21||0|0|49|concelho.nome[A_fgconc,Cdfgdist]|||||0|0||||STRING|0||0|||||0|||||0|0||0||STRING|30

campista|cadastro|Cdnatur|10|16|22|Nacionalidade|1|22|50|cadastro.nacionalidade|S|N|N|N|0|0|N||N|STRING|0||0|||||0|||||0|0||0||STRING|7

campista|cadastro|Origem|1|29|22||0|0|51|cadastro.origem|||||0|0||||STRING|0||0|||||0|||||0|0||0||STRING|1

campista|cadastro|Na_fgnum|2|99|22|Freguesia|89|22|52|cadastro.cdfregna_fgnum|N|N|N|N|0|0|N||N|NUMERIC|0||0|||||0|||||0|0||0|fregs|NUMERIC|2

campista|cadastro|Freg|20|104|22||0|0|53|fregs.nome[A_fgconc,Cdfgdist,Na_fgnum]|||||0|0||||STRING|0||0|||||0|||||0|0||0||STRING|30

regras|vista|numero|descricao|comentario|ambito

regras|cadastro|1|if after_open then _ins_vinc=_ins_sit=_ins_uni=_ins_hor=_ins_loc=0||

regras|cadastro|5|/*#REGRAS/cadastro.5*/if before_save_record && modo()~'M' && (p=%cadastro.unidade_organica[Cdnum])!~caduni then\\seltxt('A unidade organica foi modificada. Quer actualizar o historico ? ~bSim~b ~bNao~b')==1\?(_ins_uni=1;\\a=input('Qual a data da mudança (AAAA/MM/DD) ?');\dbadd('uniorg_h',Cdnum,caduni,a);\msg('O historico de unidades organicas foi modificado.')\):0;\\seltxt('A unidade organica foi modificada. Quer actualizar a conta da analítica ? ~bSim~b ~bNao~b')==2\?(_ins_uni=1;\msg('Preencha a conta analítica desejada.');\%cadastro.ct_anali[Cdnum]=%uniorg.ct_anali[caduni];\open('pcadas')\\

):%cadastro.ct_anali[Cdnum]=%uniorg.ct_anali[caduni]\\\||

regras|cadastro|6|/*#REGRAS/cadastro.6*/if before_save_record && modo()~'M' && (b=%cadastro.vinculo)!~cadvinc && seltxt('O vinculo foi modificado. Quer actualizar o historico ? ~bSim~b ~bNao~b')==1 then_ins_vinc=1;\a=input('Qual a data da mudança (AAAA/MM/DD) ?');\dbadd('vinc_his',Cdnum,cadvinc,a);\msg('O historico de vinculos foi modificado.')\/*\open('vinc_his','A')*/||

regras|cadastro|7|/*#REGRAS/cadastro.7*/if before_save_record && modo()~'M' && (b=%cadastro.tipo_situacao)!~Cadsitu && seltxt('A situacao foi modificada. Quer actualizar o historico ? ~bSim~b ~bNao~b')==1 then_ins_sit=1;\\a=input('Qual a data da mudança (AAAA/MM/DD) ?');\dbadd('histsitu',Cdnum,a,Cadsitu);\msg('O historico de situacao foi modificado.')\\/*\\open('histsitu','A')*/||

regras|cadastro|9|1 /*if after_save_record && novo then dbadd('uniorg_h',Cdnum,caduni,today());dbadd('vinc_his',Cdnum,cadvinc,today());dbadd('histsitu',Cdnum,today(),Cadsitu)||

regras|cadastro|10|1 /*if before_save_record then novo=modo()~'A'||

regras|cadastro|11|if modo()~'A' then Cddefic = 'N'||

regras|cadastro|12|if modo()~'[MA]' && Cdtitcoj->avail && Cdtitcoj!~'[SN]' then undo('S-Sim (Dois titulares) N-Nao (Unico titular)')||

regras|cadastro|13|if modo()~'[MA]' && Cdsexo->avail && Cdsexo!~'[FM]' then undo('M-Masculino F-Feminino')||

regras|cadastro|14|if modo()~'[MA]' && Cddefic->avail && Cddefic!~'[SN]' then undo('S-Sim N-Nao')||

regras|cadastro|16|if modo()~'[MA]' && Cdcrtcon->avail && Cdcrtcon!~'[SN]' then undo('S-Sim N-Nao')||

regras|cadastro|17|if modo()!~'I' && Origem->avail && Origem!~'[UPBO]' then undo('ORIGEM COMUNITARIA (excepto PORTUGAL) :\\n\\n U - Uniao Europeia\\n P - PALOP\\n B - Brasil\\n O - Outros')||

regras|cadastro|20|if modo()~'[AM]' then %ctgerais.CM_n_pessoa_col==680004327 ? ((Cadsitu->avail && Cadsitu!~'06' && Cadsitu!~'05' && !Cdadmfp->avail)?(msg('Falta preencher a data de admissao a funcao publica !');Cdadmfp->mandat=1):Cdadmfp->mandat=0):Cdadmfp->mandat=1||

regras|cadastro|22|before_save_record ? (Login = getuserid();Datal=today()) : 0||

regras|cadastro|23|/*#REGRAS/cadastro.23*/if before_save_record && modo()~'M' && (b=%cadastro.local_trabalho)!~Cdloctra && seltxt('O Local de Trabalho foi modificado. Quer actualizar o historico ? ~bSim~b ~bNao~b')==1 then_ins_loc=1;\\a=input('Qual a data da mudança (AAAA/MM/DD) ?');\dbadd('loc_hist',Cdnum,Cdloctra,a);\msg('O historico de locais de trabalho foi modificado.')\\/*\\open('loc_hist','A')*/||

regras|cadastro|25|/*#REGRAS/cadastro.25*/if Hor->avail then \Descr=switch(Hor,'D','Diurno','N','Nocturno','L','Livre','P','Parcial','R','Rigido','F','Flexivel','J','Jornada Continua','Z','Desfazado','T','Turnos','I','Isencao de Horario','E','Trab. estudante','A','Assist. desc. menores')\||regras|cadastro|26|/*#REGRAS/cadastro.26*/if modo()~'[MA]' && Estcivil->avail && Estcivil!~'[SCVDFU]' then \undo('S-Solteiro C-

Casado V-Viuvo D-Divorciado F-Separado de facto U-Uniao de facto') else (xx=switch(Estcivil,'S','Solteiro','C','Casado','V','Viuvo','D','Divorciado','F','Separado de facto','U','Uniao de facto'))\||

regras|cadastro|27|/*#REGRAS/cadastro.27*/before_save_record \?(\
dbadd('regalter',Cdnum,format(':',curtime()));\ %regalter.login = getuserid();\
%regalter.data=today()\) : 0\||

regras|cadastro|28|if after_load_record && dbset(1,'habilit.numero',Cdnum) then Hab='S'||

regras|cadastro|30|if modo()~'A' && Cdncont->avail && ^dbset(1,'cadastro.n_contribuinte',Cdncont) then msg('ATENÇÃO: Já existe um registo de cadastro com esse nº de contribuinte.')||

regras|cadastro|31|if modo()~'[MA]' && Cdct->avail && Cdct!~'[QC]' then undo('Q-Quadro, C-Contratado') else (Cdct~'Q'?(Des='Quadro';%cadastro.ncontrato[Num]=''):(Des='Contratado'))||

regras|cadastro|32|if after_query then gotofld(Cdnum)||

regras|cadastro|33|/*#REGRAS/cadastro.33*/if before_save_record && modo()~'A' /*&& (b=%cadastro.local_trabalho)!~Cdloctra */&& seltxt('O Local de Trabalho foi modificado. Quer actualizar o historico ? ~bSim~b ~bNao~b')==1 then_ins_loc=1;\\a=input('Qual a data da mudança (AAAA/MM/DD) ?');\dbadd('loc_hist',Cdnum,Cdloctra,a);\msg('O historico de locais de trabalho foi modificado.')\\/*\\open('loc_hist','A')*/||

regras|cadastro|34|/*#REGRAS/cadastro.34*/if before_save_record && modo()~'A'/* && (b=%cadastro.vinculo)!~cadvinc */&& seltxt('O vinculo foi modificado. Quer actualizar o historico ? ~bSim~b ~bNao~b')==1 then_ins_vinc=1;\a=input('Qual a data da mudança (AAAA/MM/DD) ?');\dbadd('vinc_his',Cdnum,cadvinc,a);\msg('O historico de vinculos foi modificado.')\/*\open('vinc_his','A')*/||

regras|cadastro|35|/*#REGRAS/cadastro.35*/if before_save_record && modo()~'A' /*(b=%cadastro.tipo_situacao)!~Cadsitu */&& seltxt(' A situação foi alterada. Quer actualizar o historico ? ~bSim~b ~bNao~b')==1 then_ins_sit=1;\\a=input('Qual a data da mudança (AAAA/MM/DD) ?');\dbadd('histsitu',Cdnum,a,Cadsitu);\msg('O historico de situacao foi modificado.')\\/*\\open('histsitu','A')*/||

regras|cadastro|36|/*#REGRAS/cadastro.31*/if before_save_record && modo()~'A' /*&& (p=%cadastro.unidade_organica[Cdnum])!~caduni */&& seltxt('A unidade organica foi modificada. Quer actualizar o historico ? ~bSim~b ~bNao~b')==1 then _ins_uni=1;\\a=input('Qual a data da mudança (AAAA/MM/DD) ?');\dbadd('uniorg_h',Cdnum,caduni,a);\msg('O historico de unidades organicas foi modificado.')\\/*\open('uniorg_h','A')*/\||

regras|cadastro|37|Orgeco1=switch(Orgeco,'S','Sim','N','Não')||

comadic|vista|numero|nome|tecla|nivel_minimo|nivel_maximo|accao|residente|posicao_x|posicao_y|sombra|invisivel|mensagem|altura|largura|icon

comadic|cadastro|1|Observacoes||0|100|open('obsss')||0|0||||0|0|

comadic|cadastro|2|Hist. Vinculo||0|100|open('vinc_his')||0|0||||0|0|

comadic|cadastro|3|Hist. Situacao||0|100|open('histsitu')||0|0||||0|0|

comadic|cadastro|4|Hist. Uni. Organica||0|100|open('uniorg_h')||0|0||||0|0|

comadic|cadastro|5|Hist. Horario||0|100|open('hist_hor')||0|0||||0|0|

comadic|cadastro|6|Hist. Local trabalho||0|100|open('loc_hist')||0|0||||0|0|

comadic|cadastro|19|-||0|100|0||0|0||||0|0|

comadic|cadastro|20|Gestao de Pessoal||0|100|open('sigpess1')||0|0||||0|0|

comadic|cadastro|21|Gestao de Vencimentos||0|100|open('sigvenc1')||0|0||||0|0|

comadic|cadastro|30|Z||0|100|gotofld(cadhab);zoom()|S|83|11||S||0|0|zoom.bmp

comadic|cadastro|31|Z||0|100|gotofld(Cadsitu);zoom()|S|13|14||S||0|0|zoom.bmp

comadic|cadastro|32|Z||0|100|gotofld(Quadr);zoom()|S|13|15||S||0|0|zoom.bmp

comadic|cadastro|33|Z||0|100|gotofld(cadqual);zoom()|S|70|15||S||0|0|zoom.bmp

comadic|cadastro|35|Z||0|100|gotofld(Cdloctra);zoom()|S|71|14||S||0|0|zoom.bmp

comadic|cadastro|36|C||0|100|gotofld(Cddatbi);zoom()|S|41|20||S||0|0|cal.bmp

comadic|cadastro|37|C||0|100|gotofld(Cdvalbi);zoom()|S|41|21||S||0|0|cal.bmp

comadic|cadastro|38|C||0|100|gotofld(Cddatval);zoom()|S|65|21||S||0|0|cal.bmp

comadic|cadastro|39|C||0|100|gotofld(Cddatnas);zoom()|S|85|20||S||0|0|cal.bmp

comadic|cadastro|40|Z||0|100|gotofld(Cdfgdist);zoom()|S|102|20||S||0|0|zoom.bmp

comadic|cadastro|41|Z||0|100|gotofld(A_fgconc);zoom()|S|102|21||S||0|0|zoom.bmp

comadic|cadastro|42|Z||0|100|gotofld(Na_fgnum);zoom()|S|102|22||S||0|0|zoom.bmp

comadic|cadastro|43|Z||0|100|gotofld(Vinculo);zoom()|S|87|1||S||0|0|zoom.bmp

comadic|cadastro|44|Z||0|100|gotofld(Uniorg);zoom()|S|95|2||S||0|0|zoom.bmp

comadic|cadastro|45|Z||0|100|gotofld(Apolice);zoom()|S|82|16||S||0|0|zoom.bmp

comadic|cadastro|91|Z||0|100|gotofld(Hab);zoom()|S|93|12||S||0|0|zoom.bmp

comadic|cadastro|92|ABRIR||0|100|_foto = trim(choosefile(getenv('IMAGEDIR')..'*.*p*'));length(_foto) > 0 ? Foto = _foto : 0|S|1|5||||0|10|

comadic|cadastro|93|Z||0|100|gotofld(Contrib);zoom()|S|66|1||S||0|0|zoom.bmp

comadic|cadastro|95|Z||0|100|gotofld(Grupo);zoom()|S|37|3||S||0|0|zoom.bmp

comadic|cadastro|96|F||0|100|^cadastro.isopen?close('cadastro'):0;^cadas1.isopen?close('cadas1'):0;^agre.isopen?close('agre'):0;^wcad4.isopen?close('wcad4'):0;^sighist.isopen?close('sighist'):0;^cursos.isopen?close('cursos'):0;close()|S|123|0||S||0|0|close.bmp

comadic|cadastro|97|DADOS BIOGRÁFICOS||0|100|^cadastro.isopen?select('cadastro'):open('cadastro')|S|0|6||||0|0|

comadic|cadastro|98|VENCIMENTOS||0|100|^cadas1.isopen?select('cadas1'):open('cadas1')|S|18|6||||0|0|

comadic|cadastro|99|AGREGADO FAM.||0|100|^agre.isopen?select('agre'):open('agre')|S|30|6||||0|0|

comadic|cadastro|100|ANTIGUIDADE||0|100|^wcad4.isopen?select('wcad4'):open('wcad4')|S|44|6||||0|0|

comadic|cadastro|101|HISTÓRICOS||0|100|open('sighist')|S|56|6||||0|0|

comadic|cadastro|102|FORMAÇÃO||0|100|^cursos.isopen?select('cursos'):open('cursos')|S|67|6||||0|0|

comadic|cadastro|103|GESTÃO DE VENCIMENTOS||0|100|if %ctgerais.CM_n_pessoa_col==500051054 && %users.grupo~'PES*' then open('falta') else open('abonos')|S|103|7||||0|0|

comadic|cadastro|104|RECIBO||0|100|open('rem_emp')|S|76|6||||0|0|

comadic|cadastro|105|Z||0|100|gotofld(Subscr);zoom()|S|119|17||S||0|0|zoom.bmp

relavist|vista|numero|titulo|filtro|largura|impressora|tipo|nome|tipo_caracteres|margem_superior|margem_inferior|margem_esquerda|ordem|imprimir_fundo|salto_pagina|camara|cabecalho_pagina

relavist|cadastro|1|Listagem do Cadastro|($cadastro)|126|WIN2|1|||0|0|0|0|||0|

relavist|cadastro|2|Listagem de Historicos por funcionario|($cadastro\\n($cadas1\\n)($agre\\n)($subscr1\\n)($clas_emp\\n)($habilit\\n)($prolou\\n)($concurs\\n)($cursos\\n)($histsitu\\n)($uniorg_h\\n)($vinc_his\\n)($acidente\\n)($gruphist)\\f)|126|WIN2|1|||0|0|0|0|||0|

relavist|cadastro|3|Listagem de Vencimentos por Funcionario|($cadastro\\n($cadas1\\n)$abonos\\n$desconto\\n$horasex\\n$falta\\n($comparti\\n)($ajudcust\\n)($ferias)\\f)|126|WIN2|1|||0|0|0|0|||0|

ANEXO D: Excerto de *User Control* gerado a partir de ficheiro de vista compilada do Saga (cadastro)

<%@ Control Language="C#" ClassName="cadastro" %>

<%@ Register Assembly="Gaia.WebWidgets" Namespace="Gaia.WebWidgets" TagPrefix="gaia" %>

<gaia:Window runat="server" ID="cadastro_window" Caption="Dados dos funcionários" Height="561px" Width="1112px" Left="0" Top="51" CssClass="vista" Resizable="False">

<gaia:Menu ID="cadastro_toolbar" CssClass="ivory_white_menu" runat="server" OnMenuClicked="Toolbar_Click" >

<gaia:MenuItem Value ="New" Title = " New "/> <gaia:MenuItem Value ="Save" Title = " Save "/> <gaia:MenuItem Value ="Delete" Title = " Delete "/> <gaia:MenuItem Value ="First" Title = " << "/> <gaia:MenuItem Value ="Previous" Title = " < "/> <gaia:MenuItem Value ="Next" Title = " > "/> <gaia:MenuItem Value ="Last" Title = " >> "/> </gaia:Menu>

<script runat="server">protected void Toolbar_Click(object sender, EventArgs e){ ((sender as System.Web.UI.Control).Page as Gaia.MVC.Page).ViewPropertyUpdatedMethod(sender as System.Web.UI.Control, "cadastro.Toolbar", (e as Gaia.WebWidgets.MenuClickedEventArgs).MenuItemValue);}</script>

<table>

/** Definição de Labels **/

<tr></tr>

<tr><td><gaia:Label ID="prmp001_lbl" runat="server" style="top:102px;left:608px; position:absolute;" Font-Names="Verdana" Font-Size="Smaller" Text="VÍNCULO" /></td></tr>

<tr><td><gaia:Label ID="prmp002_lbl" runat="server" style="top:119px;left:608px; position:absolute;" Font-Names="Verdana" Font-Size="Smaller" Text="UNI.ORGÂNICA" /></td></tr>

<tr><td><gaia:Label ID="prmp003_lbl" runat="server" style="top:442px;left:400px; position:absolute;" Font-Names="Verdana" Font-Size="Smaller" Text="Validade" /></td></tr>

<tr><td><gaia:Label ID="prmp004_lbl" runat="server" style="top:459px;left:256px; position:absolute;" Font-Names="Verdana" Font-Size="Smaller" Text="(" /></td></tr>

<tr><td><gaia:Label ID="prmp005_lbl" runat="server" style="top:459px;left:288px; position:absolute;" Font-Names="Verdana" Font-Size="Smaller" Text=")" /></td></tr>

<tr><td><gaia:Label ID="prmp006_lbl" runat="server" style="top:357px;left:424px; position:absolute;" Font-Names="Verdana" Font-Size="Smaller" Text="Nº Apólice Seguro" /></td></tr>

<tr><td><gaia:Label ID="prmp007_lbl" runat="server" style="top:357px;left:48px; position:absolute;" Font-Names="Verdana" Font-Size="Smaller" Text="Quadro ou Contratado ?" /></td></tr>

<tr><td><gaia:Label ID="prmp008_lbl" runat="server" style="top:408px;left:48px; position:absolute;" BackColor="#BCBCBC" Font-Names="Verdana" Font-Size="Smaller" Text=" Bilhete de Identidade " /></td></tr>

/* …… */

/* Definição de Controlos de Input e script de tratamento de eventos */

<tr><td><gaia:TextBox AutoPostback="true" OnTextChanged="Cdnum_TextChanged" runat="server" ID="Cdnum_field" Height="12" Width="40" style="top:102px;left:264px;position:absolute;" Font-Names="Verdana" Font-Size="Smaller"/></td>

<script runat="server">protected void Cdnum_TextChanged(object sender, EventArgs e){((sender as System.Web.UI.Control).Page as Gaia.MVC.Page).ViewPropertyUpdatedMethod(sender as System.Web.UI.Control, "cadastro.Cdnum", (sender as System.Web.UI.WebControls.TextBox).Text);}</script></tr>

<tr><td><gaia:TextBox AutoPostback="true" OnTextChanged="Contrib_TextChanged" runat="server" ID="Contrib_field" Height="12" Width="88" style="top:102px;left:472px;position:absolute;" Font-Names="Verdana" Font-Size="Smaller"/></td>

<script runat="server">protected void Contrib_TextChanged(object sender, EventArgs e){((sender as System.Web.UI.Control).Page as Gaia.MVC.Page).ViewPropertyUpdatedMethod(sender as System.Web.UI.Control, "cadastro.Contrib", (sender as System.Web.UI.WebControls.TextBox).Text);}</script></tr>

<tr><td><gaia:TextBox AutoPostback="true" OnTextChanged="cadvinc_TextChanged" runat="server" ID="cadvinc_field" Height="12" Width="16" style="top:102px;left:712px;position:absolute;" Font-Names="Verdana" Font-Size="Smaller"/></td>

<script runat="server">protected void cadvinc_TextChanged(object sender, EventArgs e){((sender as System.Web.UI.Control).Page as Gaia.MVC.Page).ViewPropertyUpdatedMethod(sender as System.Web.UI.Control, "cadastro.cadvinc", (sender as System.Web.UI.WebControls.TextBox).Text);}</script></tr>

/* …… */

<tr><td><gaia:LinkButton ID="cadastro_comadic92" class="squarebutton" OnClick="cadastro_comadic92_link_Click" CommandArgument="_foto = trim(choosefile(getenv('IMAGEDIR')..'*.*p*'));length(_foto) > 0 ? Foto = _foto : 0" CommandName="Comadic" runat="server" style="top:172px;left:44px; position:absolute;" >ABRIR</gaia:LinkButton></td>

<script runat="server">protected void cadastro_comadic92_link_Click(object sender, EventArgs e){((sender as System.Web.UI.Control).Page as Gaia.MVC.Page).ViewPropertyUpdatedMethod(sender as System.Web.UI.Control, "cadastro", (sender as System.Web.UI.WebControls.LinkButton).CommandArgument);}</script></tr>

<tr><td><gaia:LinkButton ID="cadastro_comadic97" class="squarebutton" OnClick="cadastro_comadic97_link_Click" CommandArgument="^cadastro.isopen?select('cadastro'):open('cadastro')" CommandName="Comadic" runat="server" style="top:189px;left:36px; position:absolute;" >DADOS BIOGRÁFICOS</gaia:LinkButton></td>

<script runat="server">protected void cadastro_comadic97_link_Click(object sender, EventArgs e){((sender as System.Web.UI.Control).Page as Gaia.MVC.Page).ViewPropertyUpdatedMethod(sender as System.Web.UI.Control, "cadastro", (sender as System.Web.UI.WebControls.LinkButton).CommandArgument);}</script></tr>

/* …… */

</table>

</gaia:Window>

<script runat="server">

protected void wnd_Moved(object sender, EventArgs e){((sender as System.Web.UI.Control).Page as Gaia.MVC.Page).ViewPropertyUpdatedMethod(sender as System.Web.UI.Control, "SagaWeb.View", e);}

</script>