UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

MÓDULO DE PLANEAMENTO DO PROCESSO DE GESTÃO NO SIAG-AP

Nuno Miguel Destapado de Sousa

Mestrado em Engenharia Informática

2008

UNIVERSIDADE DE LISBOA

Faculdade de Ciências
Departamento de Informática

MÓDULO DE PLANEAMENTO DO PROCESSO DE GESTÃO NO SIAG-AP

Nuno Miguel Destapado de Sousa

ESTÁGIO

Projecto orientado pelo Prof. Dr. Miguel Pupo Correia
e co-orientado por Eng. Nuno Pombo Rodrigues

Mestrado em Engenharia Informática

2008

FACULDADE · DE · CIÊNCIAS UNIVERSIDADE · DE · LISBOA

Declaração

Nuno Miguel Destapado de Sousa, aluno nº 31673 da Faculdade de Ciências da Universidade de Lisboa, declara ceder os seus direitos de cópia sobre o seu Relatório de Projecto em Engenharia Informática, intitulado "Módulo de Planeamento do Processo de Gestão no SIAG-A", realizado no ano lectivo de 2007/2008 à Faculdade de Ciências da Universidade de Lisboa para o efeito de arquivo e consulta nas suas bibliotecas e publicação do mesmo em formato electrónico na Internet.

FCUL, 24 de Outubro de 2008

Eng. Nuno Pombo Rodrigues, supervisor do projecto de *Nuno Miguel Destapado de Sousa*, aluno da Faculdade de Ciências da Universidade de Lisboa, declara concordar com a divulgação do Relatório do Projecto em Engenharia Informática, intitulado "Módulo de Planeamento do Processo de Gestão no SIAG-AP".

Lisboa, 24 de Outubro de 2008

Resumo

Este documento descreve o trabalho realizado no âmbito da disciplina de Projecto em Engenharia Informática do Mestrado em Engenharia Informática da Faculdade de Ciências da Universidade de Lisboa.

Este projecto, desenvolvido na GEDI S.A., teve como principal objectivo criar uma ferramenta que auxilie as organizações na tomada de decisões de maneira a melhorar o seu desempenho no negócio. A técnica utilizada para se conseguir isto denomina-se BPM (*Business Performance Management*). É uma técnica que se pode considerar uma evolução dos sistemas de BI (*Business Intelligence*) convencionais.

O projecto foi desenvolvido utilizando um conjunto de *frameworks open source*, sendo a linguagem de programação base o JAVA, na plataforma J2EE, e para as camadas de interface, negócio e dados: Struts ou JavaServer Pages, Spring e Hibernate correspondentemente. Para auxiliar no desenho de gráficos foi usada a ferramenta FusionCharts em conjunto com o Java. Foi ainda utilizada a tecnologia JavaServer Faces para a parte de *forecasting* do projecto.

A ferramenta desenvolvida funciona de forma integrada com o sistema SIAG-AP (Sistema Integrado de Apoio à Gestão para a Administração Pública) desenvolvido pela GEDI.

PALAVRAS-CHAVE:
Business Performance Management , *Business Intelligence*, Java, SIAG-AP

Abstract

This document describes the work done in the scope of the University of Lisbon Master Degree in Informatics Engineering discipline *Projecto em Engenharia Informática.*

The main goal of the project, developed at GEDI S.A. was to create a tool that would help organizations in decision making, in a way to improve their business performance. The technique used to achieve this goal is called BPM (Business Performance Management). This technique can be considered to be an evolution of the conventional BI (Business Intelligence) systems.

The project was made using a set of open source frameworks, being Java, with the J2EE framework, the main programming language, and for the interface, business and data layers: Struts or JavaServer Pages, Spring and Hibernate. To help in the graphics design, the FusionCharts tool and Java were used. JavaServer Faces was also used for the project's forecasting part.

The developed tool works integrated with GEDI's SIAG-AP (Sistema Integrado de Apoio à Gestão para a Administração Pública) system.

KEYWORDS:
Business Performance Management, Business Intelligence, Java, SIAG-AP

Conteúdo

Lista de figuras

Lista de tabelas

Capítulo 1

Introdução

Nos dias que correm já não é novidade nenhuma que soluções de BI (*Business Intelligence*) se situam na lista de prioridades dos CEOs (*Chief Executive Officer*), uma vez que permitem uma percepção mais abrangente de pontos críticos para o sucesso como a evolução do mercado, o acesso à informação em tempo real e quais os funcionários mais capacitados. Isto torna este projecto de extrema relevância e interesse para o leque de instituições que procura aumentar os seus rendimentos.

De modo a ser possível fazer uso de ferramentas de BI (*Business Inteligence*) é necessário que estejam desenvolvidos vários mecanismos para: recolha e extracção de dados de várias fontes; análise e tratamento de dados atendendo a diversos métodos e padrões; apresentação da informação tratada e compilada de forma simples para os agentes de negócio, nomeadamente a utilização de Portais para apresentação expedita de *dashboards* com relatórios, gráficos e os mais diversos indicadores.

O módulo desenvolvido neste projecto, que é explicado em pormenor no capítulo3, foi integrado no SIAG-AP (Sistema Integrado de Apoio à Gestão para a Administração Pública) desenvolvido pela GEDI.S.A. uma empresa cuja principal actividade é a concepção, desenvolvimento e implementação de soluções informáticas, sobre a qual se foca mais pormenorizadamente o capítulo 2 deste relatório.

O projecto propriamente dito, o módulo de Planeamento do Processo de Gestão (PPG), consistiu no desenvolvimento de uma *framework* de BPM (*Business Performance Management*), sobre a qual pode ser lido mais na secção 3.1 Fase Inicial, em que é feita uma identificação do tema mais aprofundada e uma comparação dos principais produtos que já a implementam, para um levantamento do conjunto de funcionalidades mais usadas no mercado.

De um modo geral, o termo BPM designa uma *framework*, considerada a nova geração dos sistemas BI (*Business Intelligence*), capaz de ajudar as organizações a melhorar a performance do seu negócio através de planos estratégicos traçados para alcançar os seus objectivos. Esta é uma *framework* que constitui um instrumento precioso de apoio à decisão, permitindo a análise e automatização de metodologias, processos, métricas e indicadores que guiarão os gestores na tomada de decisões que conduzirão ao aumento da performance das suas organizações.

A secção 3.2 Tecnologias e ferramentas utilizadas faz uma breve referência e descrição destas, seguida da secção 3.3 Implementação, em que é feita uma análise aprofundada e apresentada a informação relativa às componentes que foram implementados para o módulo de Planeamento do Processo de Gestão (PPG).

Este módulo encontra-se dividido em duas áreas – Instrumentos de Gestão e Avaliação e Controlo – e foram as componentes da área Instrumentos de Gestão (Orientações, Plano Estratégico, Objectivos, Indicadores, Iniciativas, Carta / Ficha de Avaliação, BSC e Actividades) que foram implementadas, e às quais é dedicada a secção 3.3.1 Componentes desenvolvidas.

A secção seguinte, 3.3.2 Forecasting, é dedicada a esta componente, uma vez que começou por ser desenvolvida como um projecto independente. O objectivo era mais tarde vir a ser integrada no mesmo módulo, mas acabou por ser desenvolvido apenas um protótipo de uma *pivottable*, como base de suporte para o seu desenvolvimento.

Esta é uma componente que, com base na experiência / dados de períodos anteriores, permite gerar uma previsão para os próximos períodos analisando padrões, tendências e diversos outros factores.

Após uma visão geral sobre a componente, uma sub-secção – 3.3.2.1 - fala sobre as tecnologias e ferramentas específicas que a implementação desta componente exigiu, com especial destaque para a tecnologia JSF (JavaServer Faces) e a implementação JSF *open source* utilizada – ICEfaces-1.7.0.

Utilizando esta tecnologia, e para implementar a componente de *Forecasting*, foi necessário desenvolver uma *interface* dinâmica (a *pivottable*) que permitisse a manipulação de valores e dados. Mais sobre as funcionalidades desta tabela pode ser lido na secção 3.3.2.2 Funcionalidades da *pivottable*, seguido da secção 3.3.2.3 Implementação da pivottable, dedicado às características de implementação desta.

No capítulo 4 Calendarização de tarefas são apresentados quer o plano detalhado para o projecto, que contém a identificação e calendarização das várias tarefas do projecto, quer o Mapa de Gantt correspondente, seguidos da justificação para o atraso existente na entrega deste projecto.

Por fim, no capítulo 5 Conclusão e trabalho futuro, além de uma descrição do trabalho realizado, é feito um apanhado do que seriam os passos seguintes – o trabalho que ficou por realizar.

Capítulo 2

Instituição de acolhimento

2.1 A Instituição

A GEDI, S.A. é uma empresa fundada em 1984 e cuja actividade principal é a concepção, desenvolvimento e implementação de soluções informáticas, nomeadamente, para diferentes áreas da Administração Pública. Para complementar as soluções informáticas são prestados serviços de apoio, serviços de inventariado e disponibilizados cursos de formação.

Após um longo trabalho de investigação e identificação das necessidades junto dos serviços e organismos públicos efectuados por uma equipa constituída de técnicos altamente especializados no desenvolvimento de soluções informáticas e de consultores de alguma forma ligados ao sector público, foi possível disponibilizar um conjunto de aplicações que constituem uma solução global com vista à satisfação de necessidades dos serviços e organismos. Estes organismos incluem, entre outros, Universidades, Institutos Politécnicos, Direcções-Gerais, Secretarias-Gerais, Câmaras Municipais e Escolas Secundárias, sendo contempladas, no lote de soluções oferecidas, áreas como a gestão orçamental, contabilidade pública, gestão patrimonial, gestão de pessoal, gestão de informação, etc.

O desenvolvimento das soluções referidas recorre a um ambiente gráfico, o que permite usar uma interface amigável e de fácil utilização, destinadas aos sistemas operativos Windows, Linux e Macintosh.

As principais linguagens de desenvolvimento utilizadas são o Java e o 4D – *4th Dimension* – um gestor de base de dados relacional, que suporta também a criação das interfaces da aplicação SIAG.

Os produtos da GEDI são, entre outros:

- Gestor – Gestão orçamental e Contabilidade Pública;
- Prep'OE – Elaboração de Projectos Orçamentais;
- RH+ – Gestãode Pessoal e Vencimentos;
- SPID – Sistema de Processamento de Deslocações;
- ISys – Gestão de Informação, Expediente e Arquivo;
- Facturação – Facturação de Bens e Serviços;
- Stocks – Gestão de *Stocks*;
- SIAG-AP – Ssolução integrada que pretende servir todas as áreas de apoio à gestão dos serviços e organismos públicos.

O módulo desenvolvido neste projecto foi integrado no principal produto da GEDI S.A. – o SIAG-AP – do qual falo em seguida.

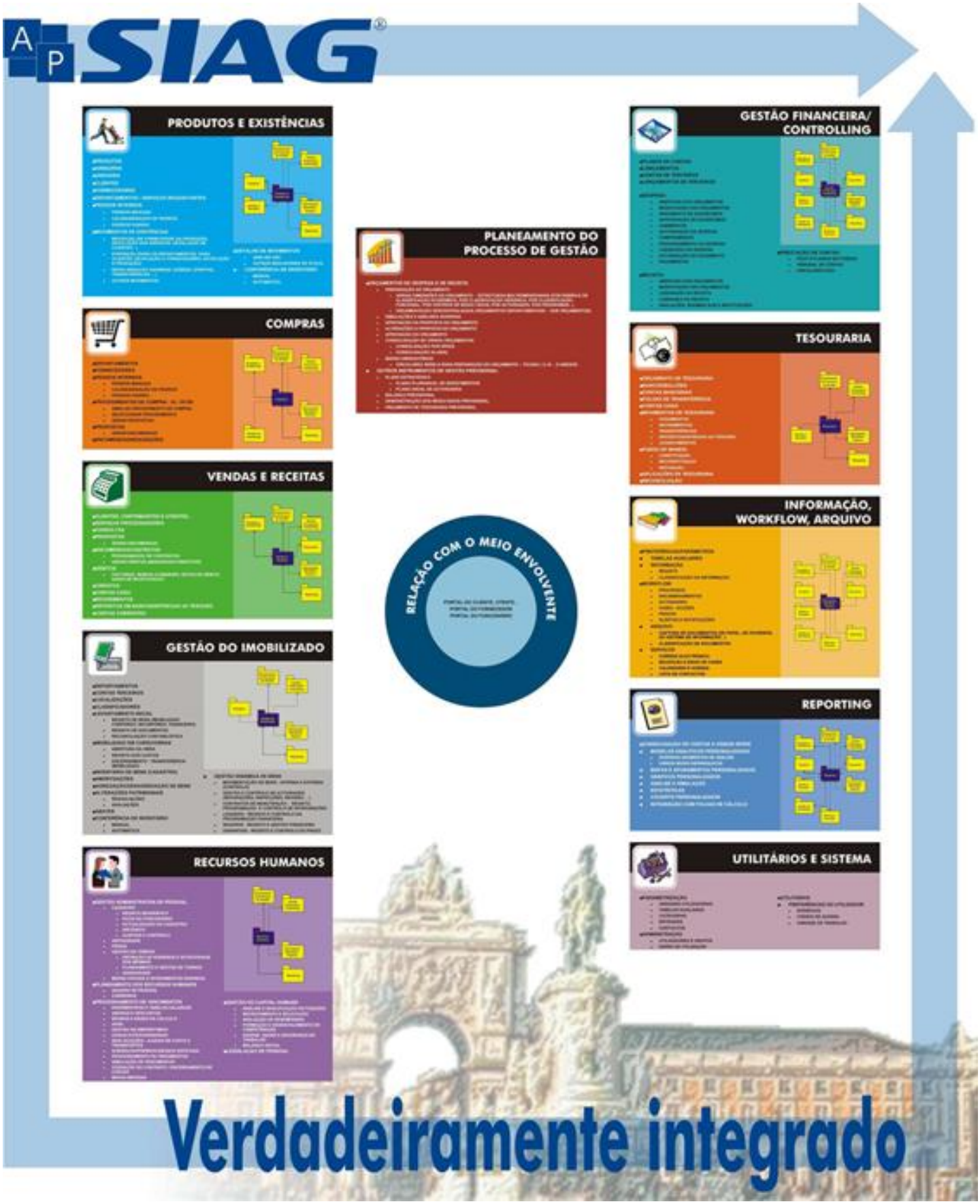

Figura 2.1:Esquema dos módulos que constituem o SIAG-AP

Fonte:GEDI [8]

2.2 SIAG-AP

A solução SIAG-AP pretende servir todas as áreas de apoio à gestão dos serviços e organismos públicos, permitindo dar resposta de forma integrada à implementação do POCP (Plano Oficial de Contabilidade Pública) e de qualquer plano sectorial dele decorrente, através de um variado conjunto de módulos. Entre estes módulos encontra-se aquele que irá conter a componente de BPM, nomeado de PPG (Planeamento do Processo de Gestão).

Os módulos desenhados para o sistema SIAG-AP são os seguintes:

- Planeamento do Processo de Gestão
- Produtos e Existências
- Compras
- Vendas e Receitas
- Gestão do Imobilizado
- Recursos Humanos
- Gestão Financeira – *Controlling*
- Tesouraria
- Informação - *Workflow* – Arquivo
- *Reporting*
- Relação com o Meio Envolvente
- Utilitários e Sistema

2.2.1 Arquitectura

Conforme se pode ver na Figura 2.2, a arquitectura do SIAG-AP divide-se em três camadas – Apresentação (UI *Layer*), Negócio (*Business Layer*) e Dados (*Persistence Layer*) – cujas funções, específicas a cada uma e com formas próprias de as resolver, detalho posteriormente, e que no SIAG-AP tomam a forma das *frameworks Struts/Tiles*, *Spring* e *Hibernate*, respectivamente.

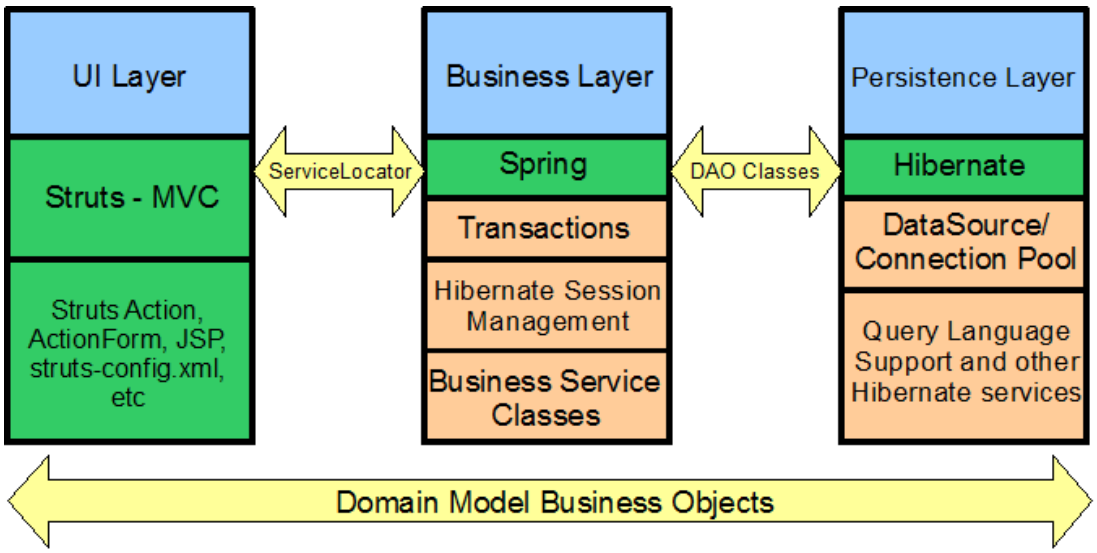

Figura 2.2: Visão global da arquitectura do SIAG-AP

A **Camada de apresentação** é responsável pelo controlo da parte da aplicação visível ao utilizador, mais concretamente:

- Gerir os pedidos e as respostas para um utilizador.
- Providenciar a delegação de chamadas à lógica de negócio.
- Disponibilizar os dados de uma forma coerente ao utilizador.
- Executar validações de introdução de dados.

As responsabilidades da **Camada de Negócio** são:

- Lógica de negócio da aplicação e por validações de negócio
- Gestão transaccional
- Permitir as interfaces comunicarem com as outras camadas
- Gestão das dependências entre objectos de negócio
- Adicionar flexibilidade entre a camada de apresentação e a camada de dados de forma a elas não comunicarem directamente.

A **Camada de Dados** é a camada responsável pela correcta gestão de todos os dados da aplicação. Mais concretamente:

- Transformar informação relacional em objectos através da *framework* Hibernate, já referida, utilizando uma linguagem OO chamada HQL.
- Gravação, actualização e remoção de informação guardada numa base de dados.
- O Hibernate suporta as principais bases de dados, transacções, herança e polimorfismo.

Como se pode ver na Figura 2.3, esta divisão por camadas da *framework* GEDI corresponde à divisão do processamento em três componentes distintas proposta no padrão de desenho MVC (*Model-View-Controller*). Neste padrão existem as camadas Modelo (*Model*), Apresentação (*View*) e Controlo (*Controller*).

O **Modelo** representa os dados da aplicação que no nosso caso são os POJOs (*Plain Old Java Object*) que são obtidos através dos serviços da camada de negócio.

A **Apresentação** é feita pelo intermédio de JSP (*JavaServer Pages*). Os dados que são apresentados na JSP são obtidos através de *actionForm's* que são construídos na componente de controlo e que são mapeados para a JSP de forma automática. Estas *actionForm's* são objectos Java que mapeiam uma propriedade para cada campo do formulário de HTML.

O **Controlo** é conseguido através das *Actions*. Estas *actions* são o ponto de entrada na aplicação e é onde se trata a lógica para determinada acção. As *actions* pertencem à camada de apresentação da aplicação e interagem com a camada de negócio através de serviços. Cada *action* tem designado um serviço que é responsável por tratar da sua lógica de negócio.

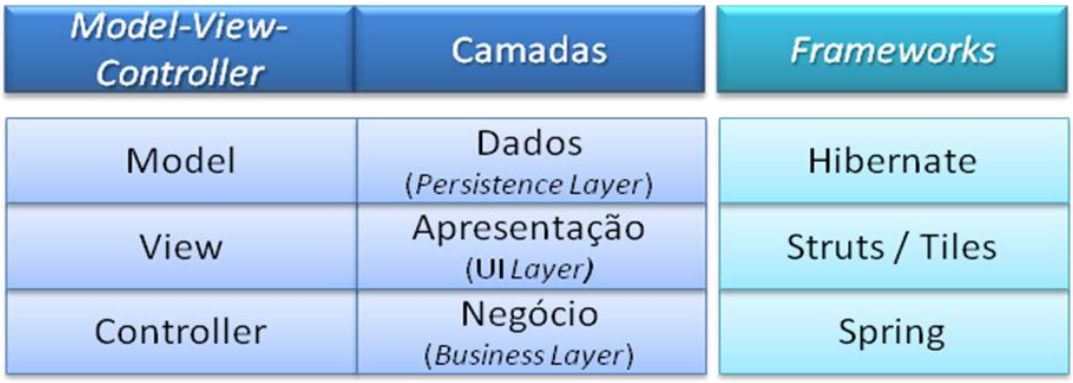

Figura 2.3: Relação entre as camadas do modelo MVC e a framework GEDI.

Como referido anteriormente, para cada camada descrita é utilizada uma *framework* de suporte, relação que se pode observar na figura Figura 2.3. Para fazer uso destas *frameworks* - mencionadas detalhadamente em Tecnologias no ponto 3.2 Tecnologias e ferramentas utilizadas - e conseguir esta divisão da aplicação por camadas foi utilizada a ferramenta GEDIDev, de que se fala em Ferramentas no mesmo ponto.

Capítulo 3

Projecto

3.1 Fase Inicial

Durante o primeiro dia de estágio, numa reunião com o supervisor na empresa, foi-me apresentado o plano de trabalho e quais os objectivos a atingir no projecto. A primeira fase do projecto consistiu numa pesquisa sobre o tema, correspondente à sub-secção 3.1.1 Identificação do tema, para perceber o que é realmente o BPM (designado por PPG no contexto do SIAG); seguida da escolha de alguns produtos relevantes existentes no mercado, recolha de funcionalidades comuns entre eles, tecnologias que cada um utiliza (a que é dedicada a secção 3.1.2 Principais produtos no mercado), de modo a poder elaborar um relatório e uma apresentação sobre a pesquisa efectuada, após duas semanas.

3.1.1 Identificação do tema

O termo BPM (*Business Performance Management*) identifica uma plataforma capaz de ajudar na tomada de decisões que visem o melhoramento do desempenho dos negócios de uma organização. Esta técnica é posta em prática através do uso de um conjunto de metodologias, indicadores e mapas estratégicos. Este conjunto designado por BPM também pode ser conhecido como CPM (*Corporate Performance Management*) ou EPM (*Enterprise Performance Management*) e engloba basicamente as acções de planeamento, orçamento, consolidação financeira, relatórios, estratégias, *balanced scorecard* e Seis Sigma (técnicas focadas na melhoria dos processos de negócio).

É necessário um certo cuidado com esta nova plataforma no que toca à aquisição dos seus produtos, uma vez que algumas empresas se aproveitam desta nova metodologia, que parece garantir resultados, para atribuir aos seus produtos que implementam soluções de BI, o nome BPM. Existe também o problema das duas expressões com a mesma sigla, *Business* ***Process*** *Management*, que diz respeito, pura e simplesmente, à organização dos processos operacionais, sem a inteligência analítica que caracteriza o *Business* ***Performance*** *Management*, que dá suporte às decisões em tempo real. Ambas usam a sigla BPM, mas este projecto incide sobre a segunda.

O BPM é visto como a próxima geração de BI, melhorando o uso dos recursos financeiros, humanos, materiais, assim como outros tipos de recursos. Envolve a consolidação de dados vindos de várias fontes, interrogando e analisando esses dados de modo a ser possível colocar os resultados em prática. As suas componentes permitem-nos perguntas como: Como estamos? (*Scorecard*) Como deveríamos estar? (*Planning*) Como iremos estar? (*Forecasting*) Porque estamos assim? (*Business Inteligence*)

Revisões contínuas e em tempo real ajudam a identificar e eliminar potenciais problemas antes que estes se tornem problemas reais. As capacidades de *forecasting* (estimar/prever) do BPM ajudam as organizações a tomar a atitude correcta na altura certa, através de uma análise de riscos. O *forecasting* é caracterizado pelo seu elevado grau de previsibilidade que pode ser usado para responder a cenários do tipo “e se”.

3.1.2 Principais produtos no mercado

Como mencionado anteriormente, uma parte da primeira fase do projecto incidiu numa pesquisa de mercado para recolher os principais produtos oferecidos na área de BPM, as tecnologias utilizadas por estes e funcionalidades existentes em comum, de modo a identificar quais as funcionalidades mais requeridas pelo mercado e, portanto, possivelmente a implementar.

Para a escolha dos produtos foram utilizados os seguintes critérios:

1. Oferta abrangente, cobrindo a maior parte, se não todas as áreas de BPM;
2. Focar-se no mercado de BPM (e não no mercado dos ERPs, uma vez que o BPM é apenas uma pequena parte do ERP);
3. Um histórico de sucesso na área;
4. Viabilidade e crescimento demonstrados pela organização;
5. Inovação contínua.

De acordo com estes critérios foram seleccionados 5 produtos - Applix, Business Objects, Clarity Systems, Pentaho e Hyperion – seguido de um breve levantamento individual de características e componentes de cada um. A relação entre estas componentes e os produtos é mais visível na tabela 3.1.

Applix TM1[2]:

Este produto assenta no alto desempenho do motor TM1 OLAP. As características mais apreciadas pelos clientes são a facilidade de implementação, uso e desempenho. O Applix tem componentes de *Planning, Reporting, Dashboard tools, Budgeting, Forecasting, Consolidation* e *Data Integration*.

Business Objects [4]:

Foi uma empresa de BI durante alguns anos, baseando-se apenas no uso de dashboards, o que por si só não era suficiente para se tornar numa empresa de BPM. Então a Business Objects adquiriu a SRC, uma pequena empresa que já fazia uso de *budgeting* e *consolidation* e actualmente oferece um serviço de BPM através do produto BusinessObjects EPM XI.

A Business Objects tem componentes de *Planning, Scoreboard, Dashboard, Budgeting, Forecasting, Consolidation, Reporting, Data Integration,* Serviços financeiros, Cuidados de saúde, Gestão de objectivos e Mapas estratégicos.

Clarity Systems [5]:

O Clarity 6, apesar de não ser uma solução tão conhecida no mercado como as outras que aqui apresento, oferece um vasto conjunto de ferramentas BPM e de um elevado grau de satisfação por parte dos clientes, tendo componentes de *Planning, Dashboard, Budgeting, Forecasting, Consolidation* e *Reporting.*

Pentaho[17]:

Esta solução, que nem oferece concretamente um produto de BPM, mas sim uma solução de BI com alguns componentes presentes nas soluções de BPM, foi escolhida não tanto por ser um dos produtos mais relevantes do mercado nesta área mas por ser uma solução *opensource* que assenta no J2EE. As componentes presentes são de *Reporting, Dashboard, Data Integration* e *Data Mining.*

Hyperion System 9[9]:

Líder no mercado de BPM, com uma boa e completa oferta na área de BPM, o *Hyperion System 9* está normalmente no topo da lista das vendas nesta área. Permite que os profissionais de finanças desenvolvam estratégias, modelem situações e orçamentos, desenvolvam planeamentos operacionais e estratégicos, comparem o desempenho com os planos e consolidem informações financeiras para a emissão de relatórios gerados e exigidos por lei.

A Hyperion System oferece componentes de *Planning, Scoreboard, Dashboard, Budgeting, Forecasting, Consolidation, Data Integration, Reporting, Essbase Analytics, Enterprise Analytics, Web Analysis* e *Visual Explorer.*

Desta caracterização e identificação de funcionalidades dos produtos, resultou a tabela 3.1., onde são visíveis de uma forma esquemática e resumida as funcionalidades encontradas, com destaque a negrito para as mais comuns entre os produtos.

Funcionalidades/ Produtos	Applix	Business Objects	Clarity Systems	Pentaho	Hyperion
Planning	■	■	■		■
Scorecard	■	■			■
Dashboard	■	■	■	■	■
Budgeting	■	■	■		■
Forecasting	■	■	■		■
Reporting	■	■	■	■	■
Consolidation	■	■	■		■
Data Integration	■	■		■	■
Visual Explorer					■
Essbase Analytics					■
Enterprise Analytics					■
Web Analysis					■
Financial Services		■			
Health Care		■			
Objective Management		■			
Strategical Maps		■			
Data Mining				■	

Tabela 3.1: Quadro comparativo de funcionalidades dos diversos produtos de BPM estudados

Legenda:

■ Funcionalidade disponível no produto

3.2 Tecnologias e ferramentas utilizadas

Tecnologias

A implementação da solução SIAG utilizou as tecnologias Java, J2EE, JavaScript, HTML, Apache Struts, Tiles, Spring, Hibernate e JavaServer Pages, detalhadas em seguida.

- **Java**

O **Java Standard Edition** (**SE**), no projecto designado apenas **Java**, utilizado no desenvolvimento da aplicação SIAG, é uma linguagem de programação orientada a objectos que tem como principal vantagem ser independente da plataforma de execução. Esta independência permite que a aplicação seja executada nos mais populares sistemas operativos.

- **J2EE**

Java EE (ou **J2EE**, ou Java 2 Enterprise Edition, ou em português Java Edição Empresarial) é uma edição da plataforma Java voltada para aplicações multi-camada, baseadas em componentes que são executados num servidor aplicacional. É usado principalmente para desenvolvimento de aplicações empresariais.

- **JavaScript**

O **JavaScript** é uma linguagem de programação bastante utilizada no desenvolvimento de páginas web cujo código é executado na parte cliente da aplicação e, por isso, bastante utilizado para efectuar validações de formulários. Pode também ser utilizado para interagir com objectos de outras aplicações.

- **HTML**

HTML é uma linguagem de marcação utilizada para produzir páginas web.

- **Apache Struts**

O **Struts** é uma *framework* de código livre para aplicações web desenvolvidas em Java que concretiza o, já apresentado, padrão de desenho MVC.

A configuração da componente de Struts é feita num ficheiro XML (Extensible Markup Language) designado struts-config.xml que indica, para cada uma das acções que podem ser submetidas ao servidor, qual a classe responsável por tratar o pedido, bem como um conjunto de redireccionamentos possíveis quer a acção seja bem sucedida ou não. Tem de se indicar também qual o ActionForm, referido anteriormente, que representa o formulário que vai ser apresentado ao utilizador.

O modelo de interacção do Struts está representado na Figura 2.3.

- **Tiles**

A *framework* de código livre **Tiles** permite dividir a página a apresentar num conjunto de zonas que podem ser definidas através de um ficheiro XML. No nosso caso a *framework* é usada para definir uma página de formulário com funcionalidade de base que é partilhada por todos e onde são adicionados o titulo da página, o endereço para onde o formulário deve ser submetido e qual JSP a incluir com a apresentação especifica de determinado ecrã.

- **Spring**

Spring é uma *framework* que trata de instanciar objectos Java e respectivas relações através de uma configuração escrita em XML. Isto permite recorrer apenas a ficheiros XML para definir as relações entre objectos tornando o código mais simples.

Permite também definir *interceptor's* para as chamadas aos objectos, fazer algo, e depois redireccionar a chamada para o objecto real.

- **Hibernate**

O **Hibernate** efectua o mapeamento entre as tabelas de uma base de dados no esquema relacional para o modelo *object oriented* através da introdução de objectos Java, Pojos (*Plain Old Java Objects*), que representam registos. A relação que existe entre tabelas também é transposta para os objectos através de relações *many-to-one*

onde o hibernate acrescenta ainda a possibilidade de ter relações *one-to-many* através de *set's* de pojos.

- **JavaServer Pages**

JavaServer Pages é uma tecnologia Java que permite gerar conteúdo HTML dinamicamente em tempo de execução. Através do uso de etiquetas JSP onde pode ser inserido código Java e de um servidor de aplicações que suporte JSP e Servlets conseguem-se criar páginas HTML dinâmicas. Quando o servidor de aplicações recebe o pedido de uma página JSP o conteúdo das etiquetas presentes no ficheiro é compilado e o seu resultado é código HTML.

Ferramentas

A produção de uma solução informática requer a preparação de um ambiente de desenvolvimento adaptado às tecnologias utilizadas na solução. Assim, começo por fazer um sumário das ferramentas utilizadas, que se encontram mais à frente em detalhe.

O Ambiente de Desenvolvimento Integrado utilizado neste projecto foi o MyEclipse, até pela sua integração com ferramentas de controlo de versões, que no caso deste projecto foi numa primeira fase o CVS (*Concurrent Version System*), e posteriormente o SVN (*Subversion*).

Para a criação dos gráficos animados que integraram a aplicação foi usada a ferramenta FusionCharts que utiliza a tecnologia Flash.

Uma vez que este projecto se encontra relacionado com uma aplicação web é igualmente necessário configurar um ambiente de execução adequado, isto é, instalar um servidor de aplicações e um SGBD (Sistema de Gestão de Bases de Dados).

O *Application Server* que foi utilizado, para permitir que o código Java do projecto fosse executado no lado do servidor, foi o JBoss. Sendo um dos *Application Servers* mais utilizados no mundo Java e sendo a JBoss uma empresa com alguma ligação à GEDI, S.A. por questões de suporte, esta razão foi decisiva para tomar esta decisão.

Quanto ao suporte de dados, este é oferecido pelo SGBD MySQL, ferramenta que utiliza a linguagem SQL (*Structured Query Language*) de modo a gerir a base de dados já usada pelo SIAG.

Por último, para permitir que o projecto seguisse facilmente o padrão de desenho MVC, foi utilizada a ferramenta GEDIDev, que foi também utilizada para o desenho de ecrãs da aplicação.

- **Eclipse**

O **Eclipse** é uma ferramenta *open source* conhecida pelo seu Java IDE (Integrated Development Environment), um ambiente integrado para desenvolvimento de software em Java. Pode funcionar com vários plugins que alargam o leque de funcionalidades desta ferramenta. Para o desenvolvimento deste projecto foi utilizado o plugin **MyEclipse** que oferece um conjunto de funcionalidades extra para J2EE, tais como, efectuar o *deploy* de servidores aplicacionais e geri-los.

- **Concurrent Version System**

O **CVS**, ou **Concurrent Version System** (Sistema de Versões Concorrentes) é um sistema de controlo de versões que permite que se trabalhe com diversas versões de ficheiros organizados em directorias e localizados local ou remotamente, mantendo-se as versões antigas e os logs de quando e por quem foram alterados os ficheiros.

É especialmente útil para se controlar versões de um software durante o seu desenvolvimento.

- **Subversion**

O **SVN**, ou **Subversion** é um sistema de controlo de versões desenhado especificamente para ser um substituto moderno do CVS, que se considera ter algumas limitações.

- **FusionCharts**

Desenvolvido pela InfoSoft para charting, é uma componente Flash usada para renderizar dados e gráficos animados para aplicações web e apresentações.

É multi-plataforma e pode ser utilizada com diferentes linguagens de programação, sem ser necessário ter conhecimentos de Flash, um software primariamente de desenho vectorial - apesar de suportar imagens bitmap e vídeos - utilizado geralmente para a criação de animações interactivas que funcionam embutidas num browser web.

- **JBoss Application Server**

JBoss Application Server (ou **JBoss AS**) é um servidor aplicacional *open source* desenvolvido pela JBoss, uma divisão da Red Hat que sendo baseado na tecnologia Java pode ser utilizado em qualquer sistema operativo que suporte o Java.

- **MySQL**

O **MySQL** é um sistema de gestão de bases de dados (SGBD), que utiliza a linguagem SQL (Structured Query Language) como interface. Foi desenvolvido pela MySQL AB e é um sistema multi-plataforma.

- **GEDIDev**

O GEDIDev é uma ferramenta de apoio ao desenvolvimento, feita na linguagem 4D, da GEDI que permite desenhar os ecrãs da aplicação e fazer uso do padrão de desenho MVC trabalhando com as *frameworks* referidas anteriormente (Struts/ Tiles, Spring e Hibernate). Para que isto aconteça a ferramenta gera as seguintes componentes de código:

Para o Modelo de dados:

- HBMs
- DTDs
- Contadores
- *DataSets*
- Tabelas Auxiliares

Para a Estrutura de Aplicação:

- Menus
- *Actions*
- Serviços

Para os ficheiros de configuração:

- *Struts*
- *Spring*
- *Tiles*

Para as *Resources:*

- Mensagens
- Listas Estáticas
- Listas de Tabelas

Para o Desenho:

- JSP
- JS
- XML
- *Action Form*
- *Validators*

3.3 Implementação

Este projecto consiste na implementação de um novo módulo que irá integrar o SIAG-AP, módulo esse que dá pelo nome de PPG (Planeamento do Processo de Gestão) e que se encontra dividido em duas áreas:

- Instrumentos de Gestão;
- Avaliação e Controlo.

As componentes desenvolvidas correspondem à primeira àrea, Instrumentos de Gestão, e é a estas que é dedicada a próxima secção, seguida de uma secção sobre a componente de *forecasting*.

Esta componente encontra-se numa secção separada porque começou por ser desenvolvida num projecto independente, com o objectivo de mais tarde vir também a integrar o módulo PPG do SIAG, mas acabou por ser desenvolvido apenas um protótipo de uma *pivottable*, como base de suporte para o desenvolvimento da componente de *forecasting*.

Figura 3.1: Listagem das componentes do módulo Instrumentos de Gestão

3.3.1 Componentes desenvolvidas

O trabalho realizado consistiu apenas na implementação de componentes relativas à área Instrumentos de Gestão, que se podem ver na Figura 3.1. Como a implementação de alguns destes componentes foi realizada em paralelo com outro programador da empresa serão apenas referidas as componentes que implementei, ou em que participei activamente na implementação. Estas, a azul na figura, foram:

- Orientações;
- Plano Estratégico;
- Objectivos;
- Indicadores;
- Iniciativas;
- Carta / Ficha de Avaliação;
- BSC;
- Actividades.

Iniciativas, Indicadores, Objectivos e Orientações são quatro componentes que podem ser agrupadas, uma vez que estão todas interligadas e contribuem para o mesmo fim: o fornecimento de dados ao componente BSC para que este ajude no cumprimento do plano estratégico traçado. Caso este plano estratégico não esteja a cumprir com o traçado, o BSC deve reportar que existem irregularidades o quanto antes, ou até mesmo com alguma antecedência.

Um Plano Estratégico traça a missão de uma organização, quais os objectivos gerais a atingir. Está dividido em várias Orientações, que por sua vez são influenciadas por vários Objectivos. Para alcançar cada um destes Objectivos existe um conjunto de Iniciativas que devem ser cumpridas, e que resultam do levantamento de necessidades em termos de produção de produtos, vendas e compras, entre outras, e cuja contribuição pode provir de Iniciativas com diferentes pesos mas que devem somar 100%. Além

disto, existe um conjunto de Indicadores ou métricas que medem o estado destas, através do que podemos ver como os Objectivos estão a ser alcançados.

O BSC, cuja representação parcial das tabelas de suporte na base de dados do SIAG pode ser vista na ., é uma ferramenta que ajuda as organizações a implementar uma estratégia e fazer a avaliação dos Objectivos definidos. Após a criação dos Objectivos, Indicadores e Iniciativas é necessário definir as Perspectivas e associar os Objectivos a essas Perspectivas. É também necessário definir um mapa estratégico com as respectivas relações de causa-efeito entre os Objectivos das várias perspectivas.

A esta componente, BSC, e à componente Indicadores, são dedicadas respectivamente as próximas duas secções.

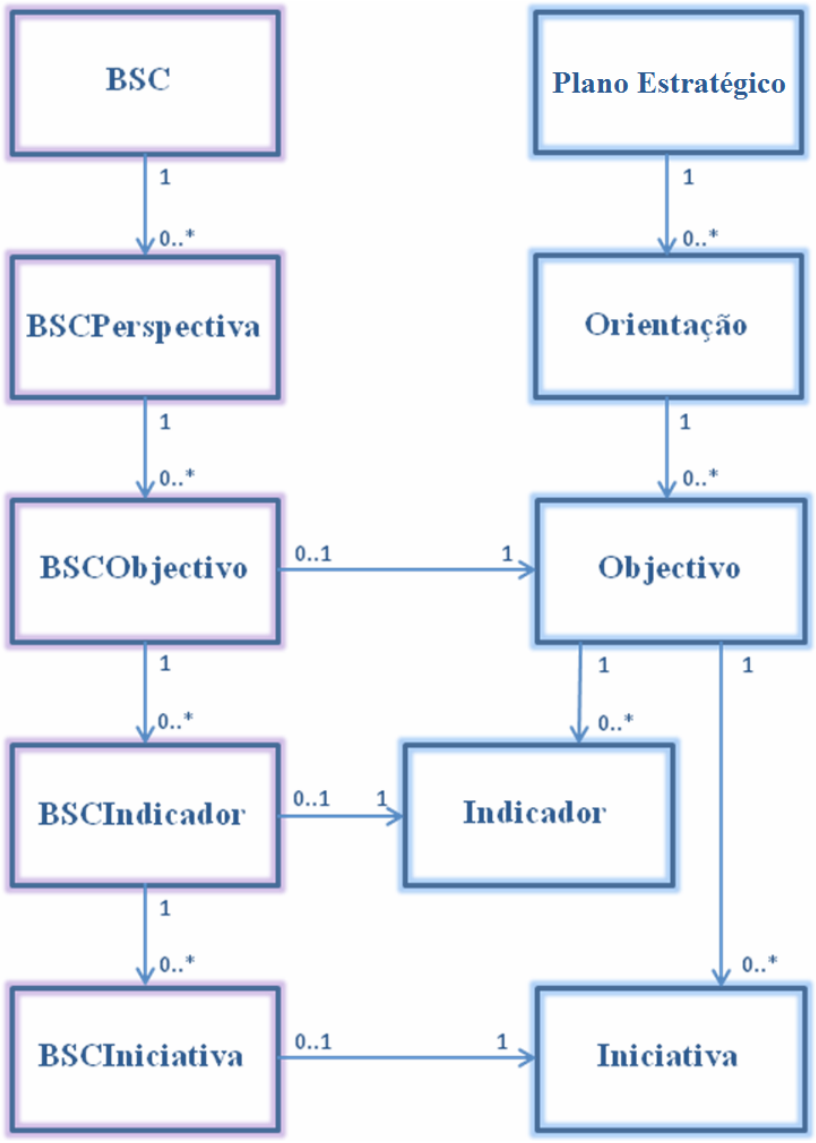

Figura 3.2: Representação parcial da base de dados do SIAG que suporta o BSC

3.3.1.1 Balanced Scorecard

Uma metodologia desenvolvida pelos professores Robert Kaplan e David Norton em 1992, o *Balanced Scorecard*, ou BSC, foi inicialmente apresentado como um modelo de avaliação de performance empresarial, porém, a sua aplicação em empresas proporcionou a sua evolução para uma metodologia de gestão estratégica.

É também classificado como uma ferramenta de suporte à decisão, pois pretende reunir elementos-chave para poder acompanhar o cumprimento da estratégia. Esta definição recebe críticas, pois abrange mais do que a tomada de decisões, focando também a comunicação da estratégia e o *feedback* do seu cumprimento.

O BSC decompõe a estratégia de uma maneira lógica, baseando-se em relações de causa e efeito, vectores de desempenho e relação com factores financeiros. É decomposto em objectivos, indicadores, e iniciativas, nas quatro perspectivas de negócio:

- **Financeira**: As medidas financeiras essenciais podem-se destacar entre o retorno dos investimentos, valor agregado ao património, lucratividade, e outras que a cultura organizacional considerar relevantes.

- **Clientes**: Analisar o negócio sob o ponto de vista dos clientes, através de indicadores tradicionais como satisfação, participação no mercado, tendências, retenção e aquisição de clientes, e outros como valor agregado aos produtos/serviços, posicionamento no mercado, nível de serviços agregados à comunidade pelos quais os clientes indirectamente contribuem, motiva a organização a manter-se focada na sua missão e na certeza de que estará desdobrando a sua visão em estratégias adequadas aos seus verdadeiros propósitos.

- **Processos internos**: Esta perspectiva garantirá a qualidade intrínseca aos produtos e processos, à inovação, à criatividade de gestão, à capacidade de produção, logística e optimização dos fluxos, assim como a qualidade das informações, da comunicação interna e das interfaces.

- **Aprendizagem e crescimento**: É esta a perspectiva que irá direccionar a atenção da empresa ao que é básico para alcançar o futuro com sucesso, considerando as pessoas em termos de capacidades, competências, motivação, alinhamento, e a estrutura da organização em termos investimentos no seu futuro. Gestão do conhecimento, mapeamento e gestão de pessoas por competência, enfim, o desenvolvimento da verdadeira "organização de aprendizagem" dá suporte a outras perspectivas que, se desalinhadas desses aspectos, apresentarão resultados efémeros. Essa perspectiva garante a solidez, valor fundamental para as empresas no futuro.

A componente essencial do BSC é o Mapa da Estratégico, que evidencia quais são os objectivos estratégicos e como estes se relacionam entre si ao longo das quatro perspectivas através de relações lógicas de causa-efeito.

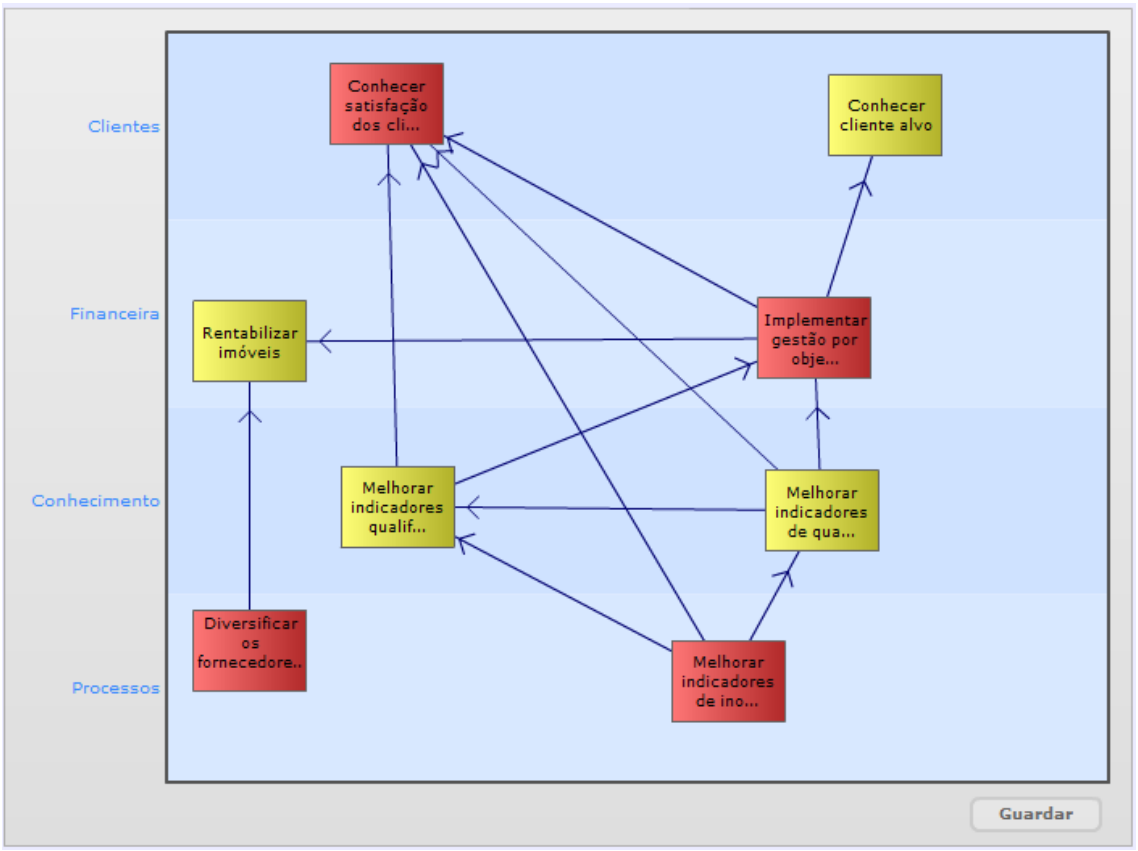

Figura 3.3: Conjunto de objectivos, divididos pelas quatro perspectivas

Durante o desenvolvimento deste projecto, foram criados vários tipos de gráficos animados, que integraram a aplicação utilizando a ferramenta FusionCharts, e através dos quais se consegue ter uma percepção mais imediata do estado das diferentes componentes estratégicas desenvolvidas.

Para o desenvolvimento destes gráficos é necessário utilizar uma linguagem de programação base, que neste caso foi o Java, este gera um ficheiro XML (*eXtensible Markup Language*) com as configurações que queremos dar ao gráfico. Cada tipo de gráfico usa uma diferente componente gráfica do FusionCharts. Estes gráficos são incorporados em ecrãs através do GEDIDev, ferramenta de desenho de ecrãs, que tem uma *tag* específica para permitir adicionar gráficos ao ecrã.

Um mapa estratégico utilizando gráficos animados gerado pela aplicação pode ser visto na Figura 3.3. Para desenhar este gráfico foi usado um gráfico da secção PowerCharts, um subconjunto de gráficos animados, do FusionCharts. O gráfico utilizado foi o **DragNode**. Segue-se uma amostra e respectiva explicação do ficheiro XML necessário para a sua construção.

```
<chart>
      <dataset>
            <set x="10" y="150" width="40" height="30" color="Yellow"
            name="Rentabilizar Imóveis" id="financeira1"/>
            <set x="250" y="350" name="Implementar gestão por
            objectivos" color="Red" id="financeira2"/>
            <set x="10" y="350" name="Diversificar os fornecedores"
            color="Red" id="processos1"/>
      </dataset>

      <connectors color="Blue">
            <connector strength="0.55" from="processos1"
            to="financeira1" arrowAtStart="0" arrowAtEnd="1"/>
            <connector strength="0.55" from="financeira2"
            to="financeira1" arrowAtStart="0" arrowAtEnd="1"/>
      </connectors>
</chart>
```

O conteúdo do ficheiro XML é sempre iniciado com a *tag chart*; para o **DragNode** existe um *dataset* com a lista de elementos (rectângulos neste caso) que compõem o gráfico e uma lista de *connectors*, as setas que interligam os elementos do *dataset*.

Cada *set* ou elemento necessita de um par de coordenadas absoluto dentro da área do gráfico, que é dado por x e y, sendo o ponto (0,0) o canto superior esquerdo da área de desenho. Para calcular estas coordenadas foi criada uma função que calcula dinamicamente o espaço disponível para colocar os elementos e, consoante o número de elementos que cabem numa linha, atribui as coordenadas a cada elemento, de modo a evitar que estes se sobreponham. Cada elemento recebe ainda uma largura (width) e altura (height), bem como um nome (name), e um atributo id que identifica univocamente cada elemento.

Segue-se a lista de *connectors*, as setas que ligam os elementos. Para cada *connector* foi definida a largura da linha, dada pelo atributo strength, o elemento de partida (from) e o elemento de chegada (to), que são referentes aos ids dos elementos. Os atributos *arrowAtStart* e *arrowAtEnd* definem se queremos um conector bidireccional ou em que sentido se quer definir a seta.

Existe ainda a possibilidade de guardar as posições dos objectivos nas perspectivas, o que permite que o utilizador os "arranje", isto é, os arraste livremente pela área de desenho, e guarde as suas novas posições, para que quando o mesmo mapa estratégico seja aberto, as coordenadas gravadas sejam lidas da base de dados e o gráfico seja mostrado com uma organização exactamente igual à versão previamente guardada.

Para cada BSC podem ser consultados os estados de cada uma das suas Perspectivas, e desagregar estas por Objectivos e Indicadores, para uma análise mais profunda de modo a perceber a razão do estado de cada Perspectiva. Na Figura 3.4 pode-se ver qual a visão que um BSC tem sobre as suas Perspectivas, e mais concretamente sobre o histórico de estados dos Indicadores de uma Perspectiva.

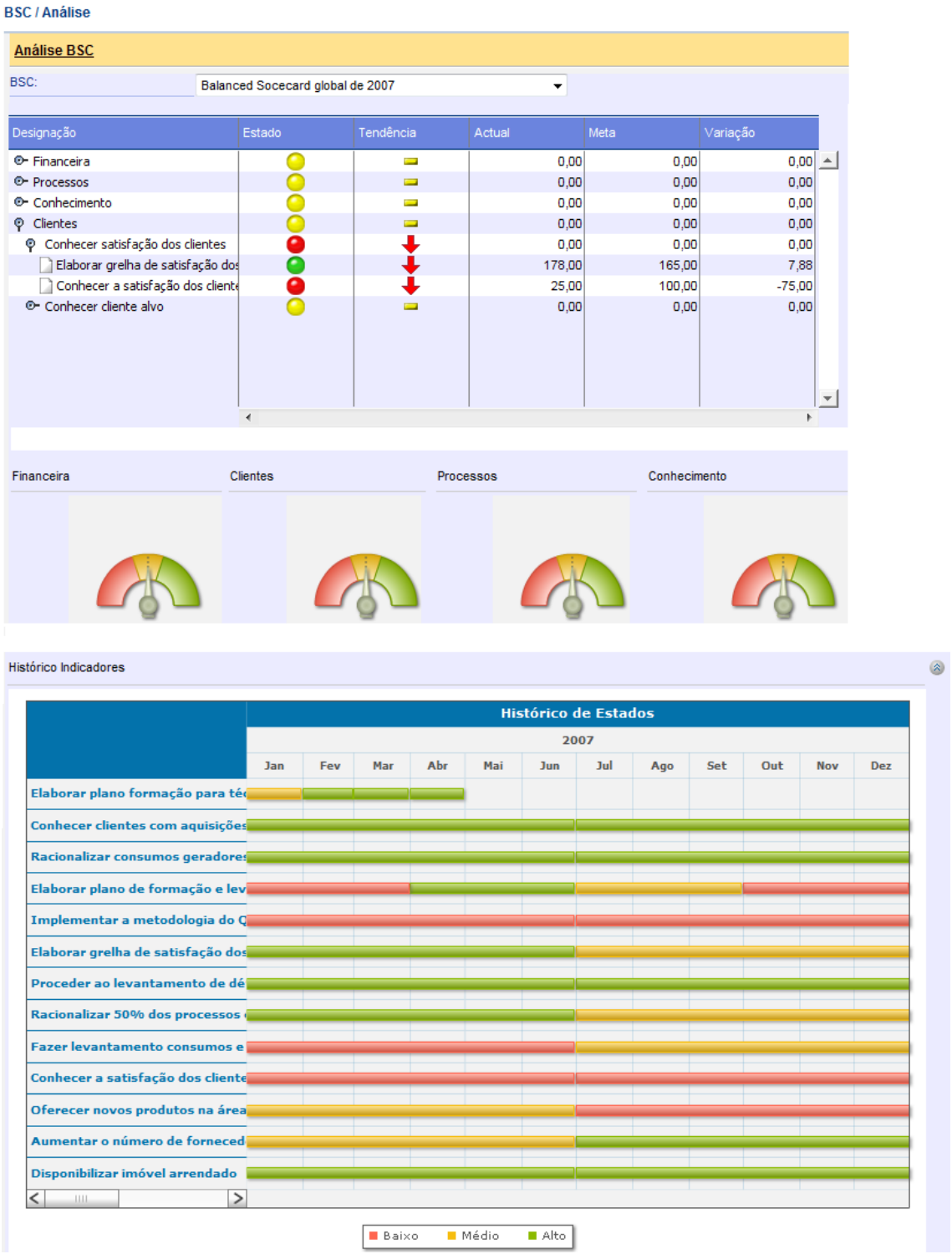

Figura 3.4: Ecrã de análise de um BSC

Neste ecrã de análise de BSC, foi adicionada uma *combobox*, para permitir seleccionar um BSC para análise. Consoante o BSC seleccionado, a *arealist* (tabela por baixo da *combobox*), é actualizada com as perspectivas referentes a esse BSC. Esta *arealist* contém, na primeira coluna "designação", uma estrutura hierárquica em árvore, que permite desagregar as perspectivas em objectivos e por sua vez em indicadores. É portanto uma estrutura hierárquica que pode ter no máximo 3 níveis.

 Indica que a perspectiva ou objectivo desta linha ainda tem subníveis, que não estão visíveis, e podem ser visto ao clicar nesta "lupa".

 Indica que a perspectiva ou objectivo desta linha já tem o seu próximo subnível todo visível.

 Neste caso estamos perante um elemento que não tem sucessores, elemento folha, este pode ser uma perspectiva, um objectivo ou um indicador.

Para a coluna de estado existem 3 tipos de estados: verde – se o valor actual superou as expectativas do valor meta, isto é, se está pelo menos 10% acima do valor definido como meta; amarelo – se o valor actual não se afastou muito da meta, nem para cima, nem para baixo. Foi definida uma margem de 10% tanto acima como abaixo da meta, assim se o valor estiver dentro deste intervalo, está amarelo; vermelho – se o valor está abaixo da margem de 10% inferior à meta.

Quanto à coluna tendência, são também 3 os tipos de tendências: seta verde para cima – se o estado do período actual é superior ao estado do período anterior; linha amarela – se o estado se manteve na mesma em relação à meta, em comparação com o período anterior; seta vermelha para baixo – se o estado piorou em relação ao período anterior.

Tanto as imagens para os estados como para as tendências, foram desenhadas com o *software* Adobe Photoshop. Para permitir que estas fossem renderizadas na *arealist*, foi necessário fazer alterações na classe Java responsável pela renderização da *arealist*.

Apenas os Indicadores têm um estado e uma tendência próprias, assim sendo, de acordo com o estado de cada Indicador e as suas contribuições para o respectivo Objectivo, o estado e a tendência deste é o resultado do que resulta dos seus

Indicadores. Por sua vez os Objectivos propagam também os seus estados e tendências para as respectivas perspectivas.

A coluna variação representa a diferença entre o valor actual e o valor meta, para variações positivas o estado é positivo, para variações negativas, o estado é negativo.

Para desenhar o gráfico Histórico Indicadores, foi utilizado o gráfico **Gantt**, também pertencente à secção dos FusionWidgets. Segue-se uma amostra e respectiva explicação do ficheiro XML necessário para a construção deste gráfico.

```
<chart dateFormat='dd/mm/yyyy' >
<categories>
      <category start='1/1/2007' end='31/12/2007' />
</categories>
<categories>
      <category start='1/1/2007' end='31/1/2007' name='Jan' />
      <category start='1/2/2007' end='28/2/2005' name='Fev' />
      (...)
      <category start='1/11/2007' end='30/11/2007' name='Nov' />
      <category start='1/12/2007' end='31/12/2007' name='Dez'/>
</categories>
<processes>
      <process Name='Elaborar plano de formação para técn.' id='A'  />
      <process Name='Conhecer clientes com aquisições sup.' id='B' />
      <process Name='Racionalizar consumos geradores.' id='C' />
</processes>
<tasks >
      <task name='Médio' processId='A' start='1/1/2007'
      end='31/1/2007' color='Yellow' />
      <task name='Alto' processId='A' start='1/2/2007' end='30/4/2007'
      color='Green' />
```

```
      <task      name='Alto'      processId='B'      start='1/1/2007'
      end='31/12/2007' color='Green' />

      <task name='Idle' processId='C' start='1/1/2007'
      end='31/12/2007' color='Green' />

      <task name='Baixo' processId='D' start='1/1/2007'
      end='31/3/2007' color='Red' />
      <task name='Alto' processId='D' start='1/4/2007' end='30/6/2007'
      color='Green' />
      <task name='Médio' processId='D' start='1/7/2007' end='0/9/2007'
      color='Yellow' />
      <task name='Baixo' processId='D' start='1/10/2007'
      end='31/12/2007' color='Red' />
</tasks>

<legend>
      <item label='Baixo' color='Red' />
      <item label='Médio' color='Yellow' />
      <item label='Alto' color='Green' />
</legend>
</chart>
```

Este gráfico está dividido em 4 secções: *categories* – onde é definida a data de início e de fim representada no gráfico, e a duração de cada período deste intervalo; *processes* – onde se definem as designações dos Indicadores e um identificador único para cada, o *id* ; *tasks* – define o histórico dos estados dos Indicadores, ou seja, através do *id* definido, por cada período de cada processo, define-se uma das 3 cores definidas na *legend*, consoante o estado desse Indicador em cada período; *legend* – faz a correspondência entre o nome de cada um dos 3 estados e a respectiva cor.

3.3.1.2 Indicadores

Como já foi referido, as Iniciativas correspondem às acções que se tomam para alcançar os Objectivos, que por sua vez, são medidos através dos Indicadores. Estes valores para os Indicadores são introduzidos através de uma tabela (esta informação é introduzida de forma manual o automática através de extracção de informação do ERP ou outras fontes de dados), e o desempenho é medido e mostrado ao utilizador sob a forma de gráficos. Na Figura 3.5 está ilustrado o ecrã de registo e edição de novos Indicadores.

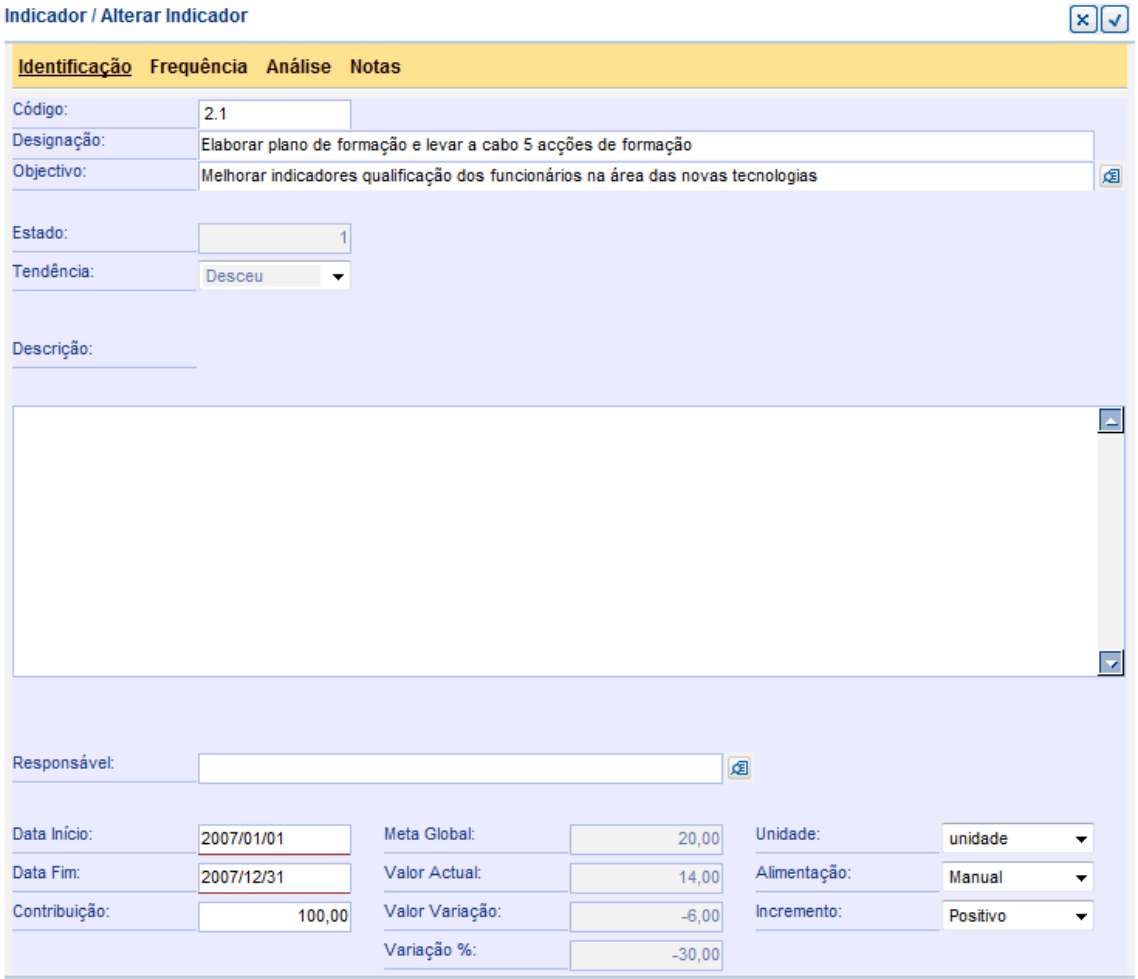

Figura 3.5: Tabulador Identificação do ecrã de registo e alteração de Indicadores

A frequência com que são efectuadas as medições destes Indicadores é definida pelos utilizadores. Na Figura 3.6 pode ser vista uma tabela de introdução manual de valores para os Indicadores, com uma frequência trimestral.

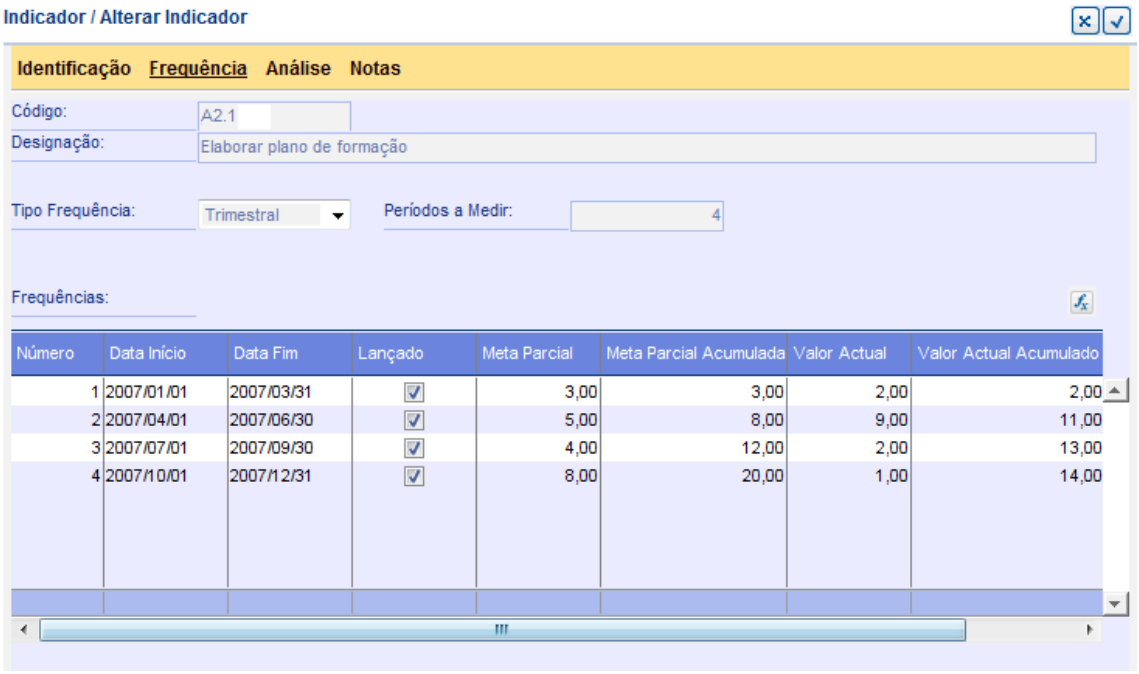

Figura 3.6: Tabulador frequência do ecrã Indicador

As frequências visíveis na tabela são calculadas a partir das datas de início e fim introduzidas no ecrã de registo de Indicador, e pelo tipo de frequência escolhido no tabulador de frequência. A partir destas datas, e de acordo com a frequência escolhida, ao ser detectado um evento de alteração quer da frequência, quer de uma das datas, uma função que calcula a nova lista de frequências é chamada, para de acordo com as novas especificações, calcular o início e fim de cada período e actualizar a tabela de frequências. Estas frequências ou períodos servem como sub-etapas para cada Indicador, tendo cada uma, uma meta parcial – objectivo a atingir durante esse período; uma meta parcial acumulada – soma das metas parciais até esse período; valor actual – valor atingido nesse período; e valor actual acumulado – soma dos valores actuais até esse período. O campo lançado serve para indicar à aplicação que, no caso de tanto as metas como os valores serem 0, esse 0 é um valor e não a falta de valor.

Para cada Indicador pode ser consultado o seu estado num intervalo de tempo para o qual este tenha sido definido, assim como qual é a tendência do seu estado, isto é, se o seu estado tem vindo a melhorar em relação aos períodos anteriores ou pelo contrário, tem uma tendência negativa, o que indica que o estado do Indicador se tem vindo a afastar cada vez mais da meta, no sentido negativo. Um conjunto de gráficos que resulta da análise de um Indicador é mostrado na Figura 3.7

Figura 3.7: Tabulador análise do ecrã de Indicadores

As datas de inicio e de fim podem ser alteradas e consoante o período escolhido os gráficos são redesenhados.

Para desenhar os gráficos da Figura 3.7 foi necessário recorrer ao gráfico **AngularGauge**, pertencente à secção FusionWidgets da FusionCharts. Segue-se uma amostra e respectiva explicação do ficheiro XML necessário para a construção do gráfico do "Estado Acumulado".

```
<chart lowerLimit='0' upperLimit='60' numberSuffix='' >
   <colorRange>
      <color minValue='0' maxValue='30' code='Red'/>
      <color minValue='30' maxValue='60' code='Green'/>
   </colorRange>

   <dials>
      <dial value='50' />
   </dials>

   <trendpoints>
      <point value='30' useMarker='1' />
   </trendpoints>
</chart>
```

Este gráfico tem definido um limite inferior e superior, nomeadamente *lowerLimit* e *upperLimit*, que ditam qual o conjunto de valores abrangido. A propriedade *numberSuffix* não tem um valor definido, mas poderia ser por exemplo "%", como é o caso dos outros gráficos presentes nesta imagem, de modo a colocar o sufixo "%" a cada valor. De seguida, o elemento *colorRange* indica quais os intervalos de cores do gráfico, e qual a cor de cada um destes.

Foi necessário definir um dial, que é o valor para onde o ponteiro fica a "apontar" e um *trendpoint*, para, neste caso mostrar qual seria o estado esperado, para poder ser feita uma comparação com o estado real.

Às diferenças entre os valores reais e aqueles que foram definidos como valores meta dá-se o nome de variações, na Figura 3.8 temos um conjunto de gráficos que permite verificar estas variações. Se a variação é positiva, significa que a meta foi superada, caso contrário, o valor meta não foi atingido.

Figura 3.8: Representação das variações entre valores reais e meta de um Indicador

Para desenhar os gráficos da Figura 3.8, usou-se o gráfico **Col3DLineSY2**, que pertence à biblioteca *standard* da FusionCharts. Segue-se uma amostra e respectiva explicação do ficheiro XML necessário para a construção do gráfico do "Valores Reais e Metas Parciais".

```
<chart caption='Valores Reais e Metas Parciais' >
      <categories>
            <category label='F1' />
            <category label='F2' />
      </categories>

      <dataset seriesName='Valores Reais' >
            <set value='15' />
            <set value='33' />
      </dataset>

      <dataset seriesname='Meta Parcial' renderAs='Line' >
            <set value='8' />
            <set value='20' />
      </dataset>
</chart>
```

As categorias correspondem ao eixo dos XX, e o primeiro *dataset*, corresponde aos valores reais, sendo que o segundo é referente à linha da meta parcial.

3.3.2 Forecasting

Forecast (**Previsão**) é uma componente da *framework* BPM que, com base na experiência / dados de períodos anteriores, permite gerar uma previsão para os próximos períodos, analisando padrões, tendências, e diversos outros factores.

É utilizada maioritariamente para prever volumes de vendas e custos.

Esta componente não chegou a ser totalmente implementada, apenas um protótipo de uma *pivottable* de suporte ao *forecast* foi feito. Foi criada uma tabela (uma *pivottable* - termo vulgarmente utilizado para designar tabelas que representam várias dimensões de dados), que permite introduzir valores para vários períodos, e a partir desses valores gerar uma previsão de valores para os períodos seguintes. A parte de gerar previsões não chegou a ser implementada, apenas a parte de introdução e manipulação de dados.

Seguidamente, antes de aprofundar a implementação, são apresentadas as tecnologias específicas a esta parte do projecto, com especial destaque para o JavaServer Faces (JSF) sobre o qual recaiu grande parte da investigação.

Foi com a ajuda desta tecnologia que foi possível construir uma *interface* dinâmica, a *Pivottable*, implementada a pensar na necessidade de manipulação de valores, tipicamente valores para despesas / receitas, e custos / proveitos por unidade temporal, permitindo ainda que esse detalhe temporal seja definido pelo utilizador em tempo real, através do *browser*, por acção de eventos de expansão e/ou contracção de colunas da tabela. Estes eventos de contracção possibilitam visualizar os valores através de somatórios de uma unidade temporal maior, por exemplo, por trimestre, semestre, ano, etc.

3.2.3.1 Ferramentas e tecnologias específicas

O desenvolvimento desta componente exigiu a utilização de algumas tecnologias mais específicas. O servidor aplicacional utilizado foi o **jboss-4.2.2.GA**, configurado para correr no porto **8081**. Assim, após a iniciação do servidor aplicacional, o URL para aceder à aplicação é:

http://localhost:8081/desenv_pivottable/home.seam

Quanto às tecnologias, além das utilizadas no projecto em geral, foram utilizadas as frameworks *JBoss Seam* e *JavaServer Faces*, descritas nos próximos sub-pontos.

- **JBoss Seam [14]**

Esta é uma poderosa *framework* para construção de aplicações Web 2.0, unificando e integrando tecnologias tais como Asynchronous JavaScript and XML (AJAX), JavaServer Faces (JSF), Enterprise Java Beans (EJB3), Java Portlets e Business Process Management (BPM). O Seam foi desenhado de raiz com o intuito de eliminar a complexidade a nível da arquitectura e da API.

Permite aos programadores construir aplicações web complexas com mecanismos simples, tais como *annotated Plain Old Java Objects* (POJOs), componentes de interface, e XML.

A versão do Jboss seam utilizada foi a **jboss-seam-2.0.1.GA**.

- **JavaServer Faces (JSF)**

Desenvolvida pela Sun, a **JSF** é uma *framework* para aplicações web baseada em Java que surgiu com o intuito de simplificar o desenvolvimento de interfaces para aplicações Java. Ao contrário das *frameworks* web tradicionais que fazem uso do padrão de desenho MVC, o JSF é baseado em componentes.

Para permitir o desenvolvimento desta componente utilizou-se esta nova tecnologia, que foi escolhida porque, para além de ser *open source* e usar a mesma linguagem de programação utilizada no resto do projecto - o Java -, oferece uma grande facilidade e versatilidade em termos de interfaces.

O seu uso foi inicialmente pensado também com o intuito de actualizar outras tecnologias existentes no produto SIAG-AP, substituindo as applets existentes na aplicação.

Com diversas implementações, característica de que falarei adiante, foi a implementação **ICEfaces-1.7.0 [10]** a utilizada no desenvolvimento deste projecto. Esta é uma implementação JSF open-source desenvolvida pela ICEsoft Technologies Inc., que disponibiliza um vasto conjunto de componentes JSF.

Para que fosse possível fazer uso desta tecnologia foi necessário seguir todo um processo de instalações e configurações. Assim, é o ficheiro de configuração da aplicação **faces-config.xml** contém as regras de navegação e os *managed beans* de uma aplicação. Por exemplo, para a página **home.xhtml**, as **regras de navegação** processam-se da seguinte forma:

```
<navigation-rule>
  <from-view-id>/home.xhtml</from-view-id>
  <navigation-case>
    <from-outcome>login</from-outcome>
    <to-view-id>/login.xhtml</to-view-id>
  </navigation-case>
```

```
    <navigation-case>
      <from-outcome>error</from-outcome>
      <to-view-id>/error.xhtml</to-view-id>
    </navigation-case>
  </navigation-rule>
```

Caso a página **home.xhtml** receba do servidor a resposta "login", as regras de navegação actuam redireccionando a aplicação para a página **login.xhtml**. No caso da resposta ser "error", a aplicação é redireccionada para a página **error.xhtml**.

Já os **Managed Beans**, são definidos do seguinte modo:

```
<managed-bean>
     <managed-bean-name>geraTabela</managed-bean-name>
       <managed-bean-class>
   action.pt.gedi.base.jsf.tabela.main.GeraTabela
 </managed-bean-class>
     <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

Quando o cliente faz o pedido de uma página inicia-se o **ciclo de vida JSF**, cujos passos mudam de acordo com os pedidos do cliente, conforme se pode visualizar na Figura 3.9.

Estes pedidos do cliente podem ser *Faces request*, em que é requisitada uma página com componentes JSF, ou *Non-Faces request*, em que a requisição passa pelo Servlet controlador do JSF, mas não entra no ciclo de vida. A estes dois tipos de pedidos correspondem dois tipos de respostas – respectivamente *Faces response* e *Non-Faces response*. No primeiro a resposta envolve uma página que entra no ciclo de vida, enquanto no *Non-Faces response*, o ciclo de vida não está envolvido.

Para o caso, interessam particularmente os pedidos e respostas que entrem no ciclo de vida do faces.

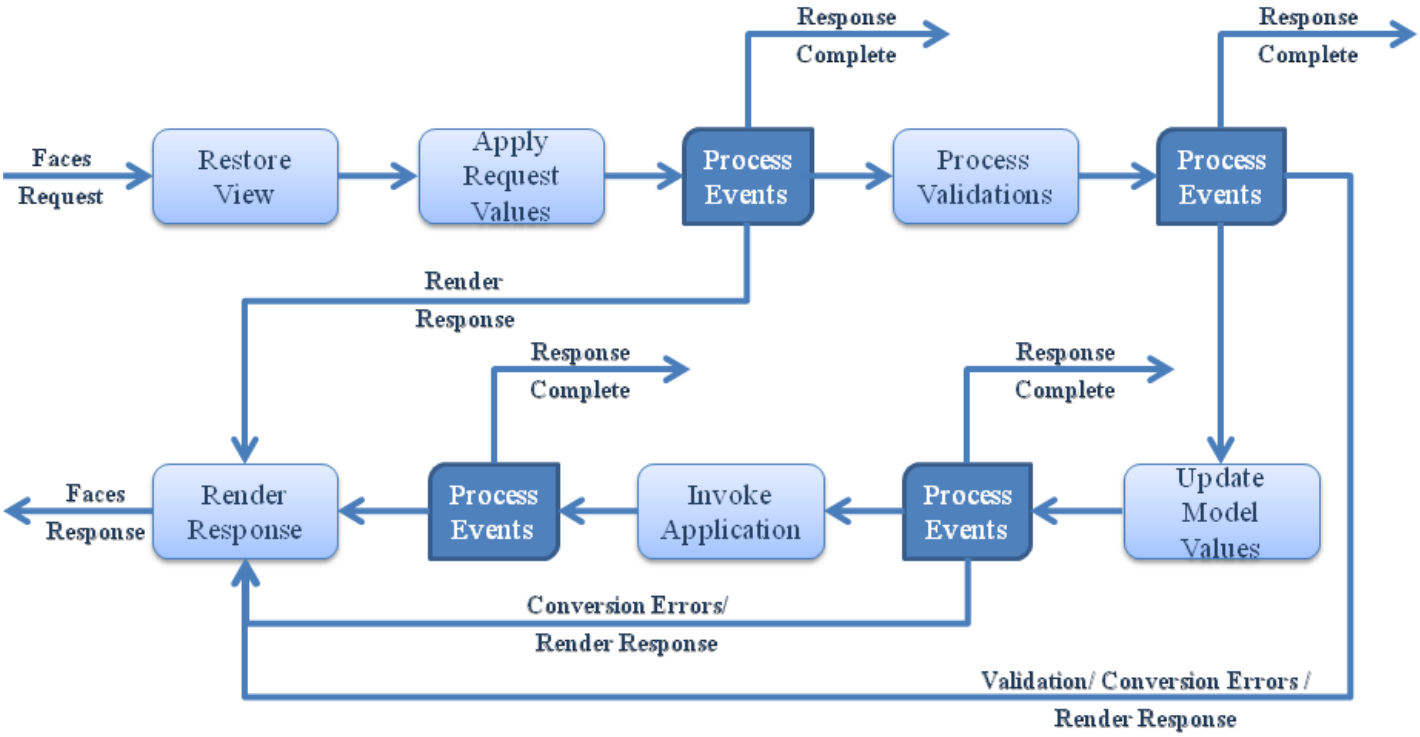

Figura 3.9: Ciclo de vida JSF

Como referido anteriormente existem diversas implementações JSF. A oferecida por defeito é a da Sun RI (*Reference Implementation*), uma implementação com um conjunto básico de componentes, mas é possível adicionar novas implementações ou até mesmo misturar um conjunto destas, de modo a enriquecer o leque de componentes oferecido, desde que tendo em conta que a existência de algumas incompatibilidades entre diferentes implementações.

Para se ter acesso aos componentes de outras implementações JSF é necessário adicionar os ficheiros .jar na pasta lib, e configurar o ficheiro **WEB-INF/web.xml**, adicionando-lhe os respectivos listeners, filters, context-params, e servlets. Neste projecto, como já referido, a implementação escolhida foi a ICEFaces e para permitir que esta implementação estivesse disponível foi necessário incluir na pasta lib os ficheiros: **icefaces.jar**, **icefaces-comps.jar**, **icefaces-facelets.jar.**

Como referido anteriormente, com esta implementação e a tecnologia JSF, foi implementada a *pivottable* cujas funcionalidades falamos na próxima secção.

3.3.2.2 Funcionalidades da pivottable

A pivottable, apresentada na figura Figura 3.10, tem algumas funcionalidades, descritas em seguida, que merecem destaque.

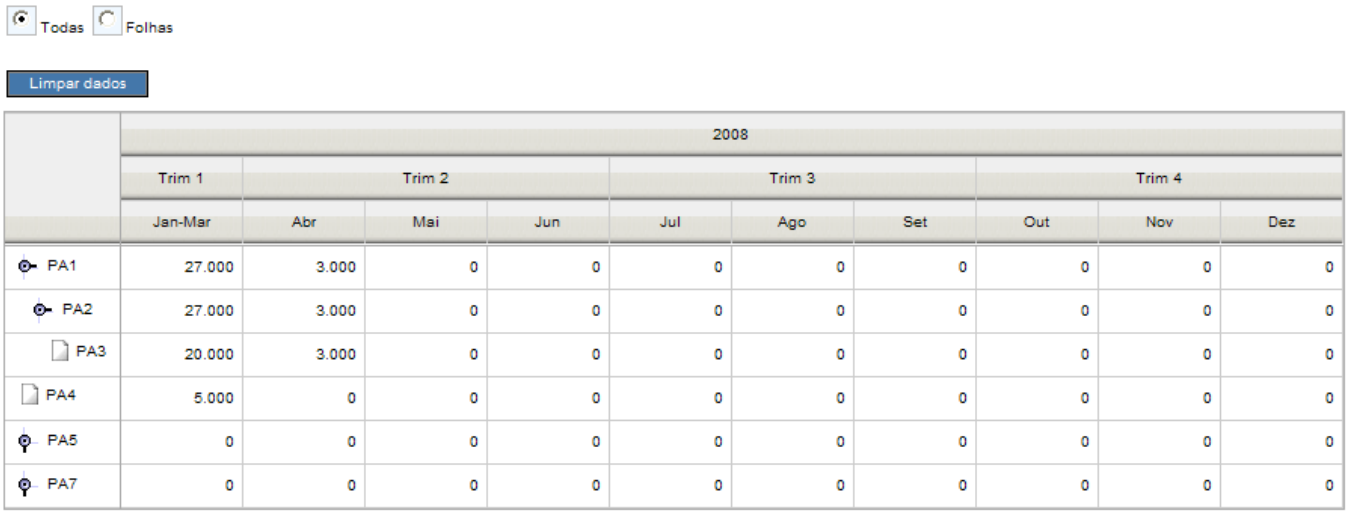

Figura 3.10: Ecrã da pivottable

De notar que na primeira coluna da pivottable existe a seguinte notação:

Nó da árvore contraído, isto é, com descendentes escondidos.

Nó da árvore expandido, isto é, com todos os seus descendentes do próximo nível visíveis.

Nó da árvore sem descendentes, isto é, é um nó de último nível.

Radio Buttons:

Estão na parte superior esquerda da imagem e disponibilizam as seguintes opções:

- **Todos**: quando seleccionado, todas as células da matriz são editáveis.
- **Folhas**: quando seleccionado, apenas as células correspondentes a linhas sem descendentes são editáveis.

Limpar dados:

Este botão encontra-se entre os *radio buttons* e a *pivottable* e serve para limpar todos os valores e somatórios de todas as células da matriz amarela, passando estes a conter o valor 0.

Propagar valores para as linhas ascendentes

Ao editar uma linha, se esta for uma linha descendente de outra na hierarquia de contas, o novo valor da célula editada vai ser propagado para as células ascendentes.

	Trim 1			Trim 2	
	Jan	Fev	Mar	Abr	Mai
PA1	30.000	8.000	7.000	3.000	0
PA2	30.000	8.000	7.000	3.000	0
PA3	12.000	8.000	0	3.000	0
PA4	0	5.000	0	0	0
PA5	0	0	0	0	0
PA7	0	0	0	0	0

Figura 3.11: Propagação de valores - Célula editada

Na Figura 3.11 encontra-se destacada uma célula que ao ser editada vai provocar alterações nas células antecedentes, por meio de propagação da nova diferença de valor. Um exemplo desta alteração é a Figura 3.12, que mostra os novos valores das células antecessoras (mais 3 unidades), após um aumento equivalente na célula editada.

	Trim 1			Trim 2	
	Jan	Fev	Mar	Abr	Mai
PA1	33.000	8.000	7.000	3.000	0
PA2	33.000	8.000	7.000	3.000	0
PA3	15.000	8.000	0	3.000	0
PA4	0	5.000	0	0	0
PA5	0	0	0	0	0
PA7	0	0	0	0	0

Figura 3.12: Propagação de valores - Resultado

Somatórios de colunas

	2008											
	Trim 1			Trim 2			Trim 3			Trim 4		
	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
PA1	12.000	8.000	7.000	3.000	0	0	0	0	0	0	0	0
PA2	12.000	8.000	7.000	3.000	0	0	0	0	0	0	0	0
PA3	12.000	8.000	0	3.000	0	0	0	0	0	0	0	0
PA4	0	5.000	0	0	0	0	0	0	0	0	0	0
PA5	0	0	0	0	0	0	0	0	0	0	0	0
PA7	0	0	0	0	0	0	0	0	0	0	0	0

Figura 3.13: Tabela sem colunas contraídas

Desde que não seja uma coluna de último nível – sem descendentes, não colapsável / expandivel, como é o caso das colunas correspondentes aos meses na Figura 3.13 – as colunas do *header* da *pivottable* são clicáveis, dando origem à contracção ou expansão de todas as colunas descendentes.

O colapso destas colunas origina o somatório das colunas contraídas, de modo a apresentá-lo na coluna que ficou visível. Na Figura 3.13 está em destaque uma coluna que ao ser clicada resultanda na tabela apresentada na Figura 3.14, com o trimestre 1 contraído. A Figura 3.15 mostra a mesma tabela, agora também com o ano 2008 contraído. A operação contrária, expansão de colunas contraídas, também é válida.

	2008									
	Trim 1	Trim 2			Trim 3			Trim 4		
	Jan-Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
PA1	27.000	3.000	0	0	0	0	0	0	0	0
PA2	27.000	3.000	0	0	0	0	0	0	0	0
PA3	20.000	3.000	0	0	0	0	0	0	0	0
PA4	5.000	0	0	0	0	0	0	0	0	0
PA5	0	0	0	0	0	0	0	0	0	0
PA7	0	0	0	0	0	0	0	0	0	0

Figura 3.14: Somatório de colunas - trimestre 1 contraído

	2008
	Trim 1-Trim 4
	Jan-Dez
PA1	30.000
PA2	30.000
PA3	23.000
PA4	5.000
PA5	0
PA7	0

Figura 3.15: Somatório de colunas - ano 2008 contraído

Distribuição de somatórios

Se uma célula pertencente a uma coluna de somatório, como a célula do exemplo da

Figura 3.16, for editada, surgirá um *popup* (na mesma figura) a perguntar se desejamos que esse novo valor de somatório seja **distribuído igualmente** ou **distribuído proporcionalmente**.

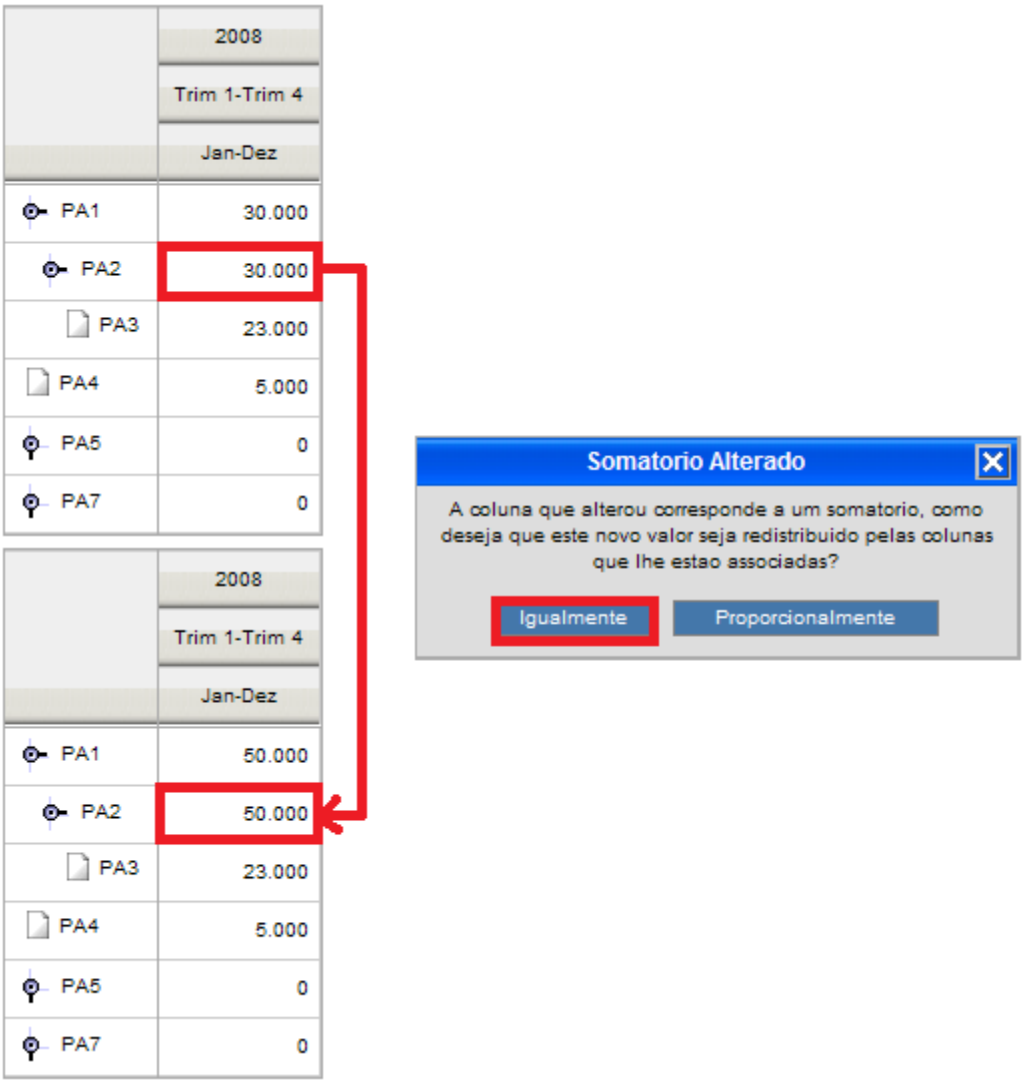

Figura 3.16: Distribuição de somatórios

Distribuído **igualmente** implica que o novo somatório seja dividido pelo número de células afectadas, contendo cada célula uma fracção desse somatório, resultando, no exemplo, na tabela da Figura 3.17.

	2008											
	Trim 1			Trim 2			Trim 3			Trim 4		
	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
PA1	4.17	4.17	4.17	4.17	4.17	4.17	4.17	4.17	4.17	4.17	4.17	4.17
PA2	4.17	4.17	4.17	4.17	4.17	4.17	4.17	4.17	4.17	4.17	4.17	4.17
PA3	12.000	8.000	0	3.000	0	0	0	0	0	0	0	0
PA4	0	5.000	0	0	0	0	0	0	0	0	0	0
PA5	0	0	0	0	0	0	0	0	0	0	0	0
PA7	0	0	0	0	0	0	0	0	0	0	0	0

Figura 3.17: Distribuição de somatórios por igualdade (Resultado)

A distibuição **proporcional** tem em conta os antigos valores das células, fazendo a diferença entre o somatório antigo e o novo somatório, dividindo essa diferença (quer seja positiva ou negativa) pelo número de células afectadas e somando essa fracção ao antigo valor de cada célula afectada. No exemplo a escolha desta opção resulta na tabela da Figura 3.18.

	2008											
	Trim 1			Trim 2			Trim 3			Trim 4		
	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
PA1	13.670	9.670	8.670	4.670	1.67	1.67	1.67	1.67	1.67	1.67	1.67	1.67
PA2	13.670	9.670	8.670	4.670	1.67	1.67	1.67	1.67	1.67	1.67	1.67	1.67
PA3	12.000	8.000	0	3.000	0	0	0	0	0	0	0	0
PA4	0	5.000	0	0	0	0	0	0	0	0	0	0
PA5	0	0	0	0	0	0	0	0	0	0	0	0
PA7	0	0	0	0	0	0	0	0	0	0	0	0

Figura 3.18: Distribuição de somatórios proporcionalmente (Resultado)

3.3.2.3 Implementação da pivottable

A *Pivottable* foi desenvolvida com o auxílio de quatro subtabelas interligadas, distinguidas com cores na Figura 3.19, e das quais se fala individualmente em seguida.

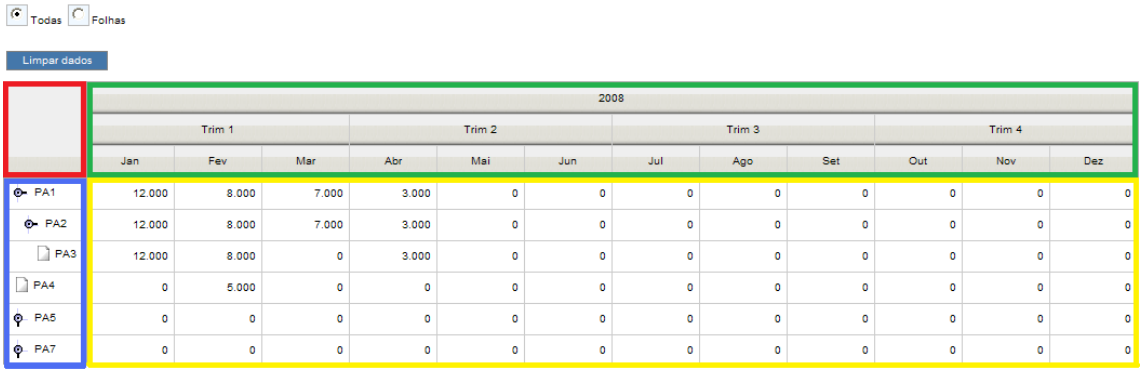

Figura 3.19: Ecrã da Pivottable

Tabela vermelha:

– Esta tabela serve apenas para questões de ajuste de *layout* e está definida na página home.xhtml, com o id **dataTableImg**.

Tabela verde:

– O *layout* desta tabela é dinâmico (depende da informação que lhe for passada), definido na classe **GeraTabela**, no método **getDataTableHeader**, e tem o id **dataTableHeader**.

– As classes Java que representam a tabela encontram-se nos *packages* pt.gedi.base.jsf.loadData, pt.gedi.base.jsf.tabela.main e pt.gedi.base.jsf.tabela.header.

Tabela azul:

– O *layout* da tabela está definido na página **home.xhtml**, com o id **dataTableContas**.

– As classes Java que representam esta tabela encontram-se nos *packages* pt.gedi.base.jsf.loadData **e** pt.gedi.base.jsf.tabela.contas.

Tabela amarela:

– Esta tabela tem o seu *layout* definido na classe GeraTabela, no método getDataTableMatriz, e tem o id **dataTableMatriz**.

–As classes Java que representam esta tabela encontram-se nos *packages* pt.gedi.base.jsf.loadData e pt.gedi.base.jsf.tabela.main.

Em seguida é feita uma descrição geral dos métodos que servem de suporte à *pivottable*; o diagrama de classes que representa esta estrutura encontra-se na figura

Figura 3.20.

Descrição geral dos métodos que suportam a *pivottable*:

getDataTableHeader

O método *getDataTableHeader* da classe GeraTabela é responsável por:

- Invocar o método *createNodeList*

createNodeList

O método *createNodeList* da classe CarregaDados é responsável por:

- Se numEstrutura = 1
- Invocar o método *createHeaderNodeList*, que é responsável por ler os periodos da Base de Dados e criar uma lista do tipo EstruturaHeader com os elementos constituintes da tabela dataTableHeader.
- Se numEstrutura = 2
- Invocar o método *createContasNodeList*, que é responsável por ler as contas da Base de Dados e criar uma lista do tipo EstruturaConta com os elementos constituintes da tabela dataTableContas.
- Após a criação de qualquer uma destas listas, esta é passada à classe EstruturaHierarquicaController, onde o método *createHierarquicNodeList* trata de criar a respectiva estrutura hierarquica, com base na lista recebida.

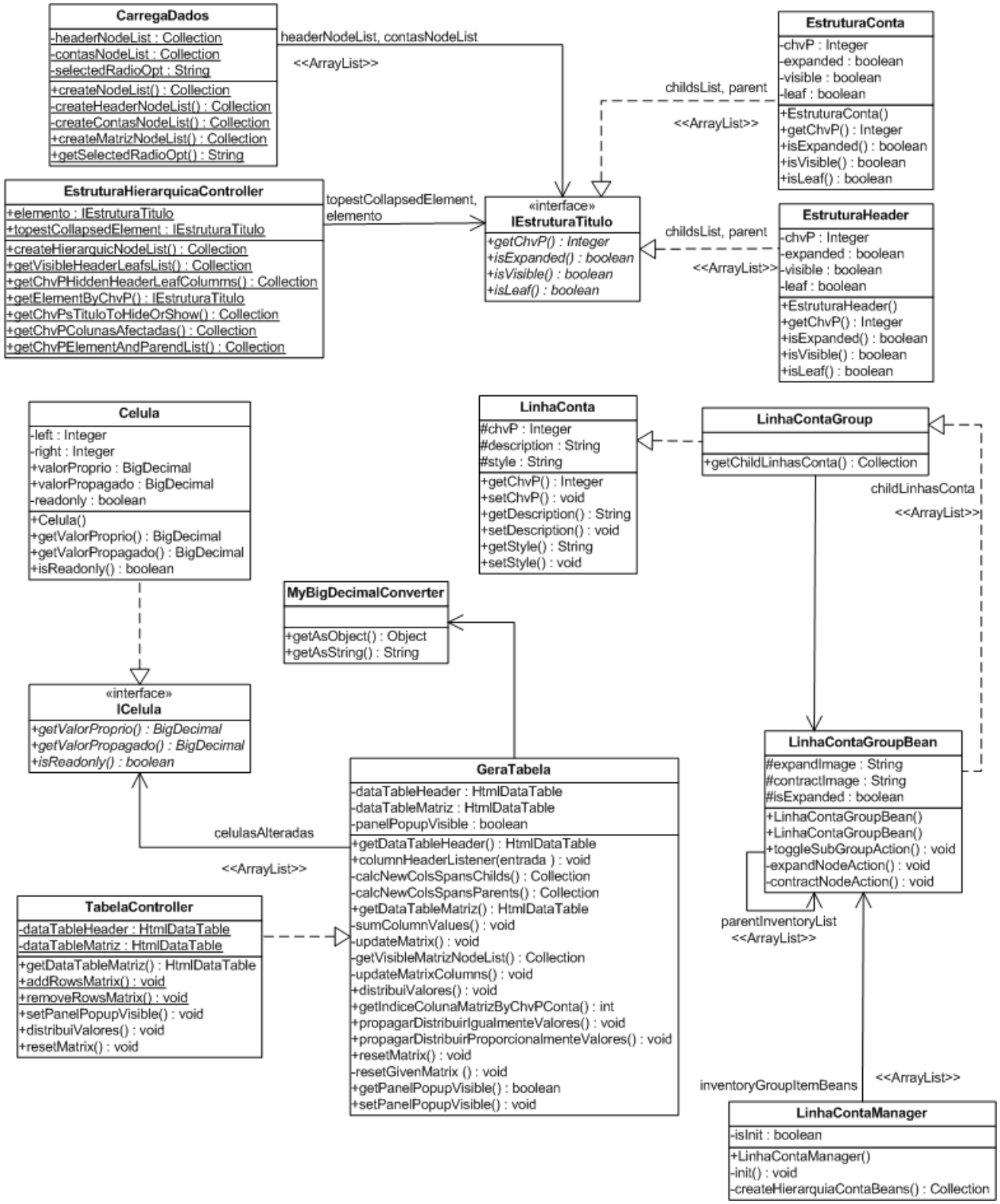

Figura 3.20:Diagrama de classes que representa a estrutura da pivottable

init

O método *init* da classe LinhaContaManager é responsável por:

- Inicializar a lista que contém em cada instante os beans (contas) visíveis da dataTableContas.
- Invocar o método *createNodeList* com o parâmetro 2.
- Invocar o método *createHierarquiaContaBeans* e passar-lhe como parâmetro a lista hierarquica e um paddingLeft. Esta lista hierarquica servirá de suporte para a criação de uma hierarquia de objectos do tipo LinhaContaGroupBean. Estes beans são criados com os seguintes argumentos: Uma imagem de expandir/contrair associada, a lista de beans que são visíveis na dataTableContas, que inicialmente é constituída pelos beans de nível um porque todos eles começam no estado contraído, um boolean que indica se o elemento está expandido ou não, e um boolean que diz se é um bean de nível um ou não.
- O paddingLeft é uma propriedade CSS (Cascade StyleSheet) utilizada para atribuir um style de indentação a cada linha da dataTableContas, consoante o seu nível hierárquico.

toogleSubGroupAction

O método *toggleSubGroupAction* da classe LinhaContaGroupBean é responsável por:

- Detectar os eventos e executar as acções de expansão/contracção dos beans da dataTableContas, invocando o método *expandNodeAction* no caso de executar uma expansão, ou o método *contractNodeAction* no caso de ser uma contracção.
- Reflectir estas acções na dataTableMatriz, no caso de ser uma expansão, invocar o método *addRowsMatrix* da classe TabelaController, para adicionar à tabela dataTableMatrix as linhas (contas descendentes) que passaram a ser visíveis. No caso de ser uma contracção, invocar o método *removeRowsMatrix* da classe TabelaController, para remover da tabela dataTableMatrix as linhas que deixaram de ser visíveis devido à contracção.

getDataTableMatriz

O método *getDataTableMatriz* da classe GeraTabela é responsável por:

- Invocar o método *getVisibleMatrizNodeList*, que obtém a lista de beans visíveis (cujos pais não estão contraídos) guardada na classe LinhaContaManager da dataTableContas. É também responsável através da invocação do método *createMatrizNodeList*, criar a matriz de células que vão preencher a dataTableMatriz.
- Para cada célula da dataTableMatriz, adicionar-lhe um componente HtmlInputText. Este componente tem um *converter* associado, do tipo *MyBigDecimalConverter*, cuja função é verificar se o input introduzido contém uma vírgula. Em caso afirmativo, trocar essa vírgula por um ponto, de modo a que o input seja aceite na fase de validação.
- Adicionar a cada inputText um evento *valueChangeListener*, de modo a que quando for detectada uma alteração num campo de inputText, através da invocação do método *updateMatrix* este possa propagar esse novo valor para os nós superiores.
- Binding de valores entre Celulas e InputTexts:

  O atributo valorPropagado de cada Celula passa a ter um binding com o atributo value de cada inputText da dataTableMatriz. Assim os valores propagados de cada Celula são mostrados na dataTableMatriz, e ao serem alterados no browser, são actualizados os correspondentes valoresPropagados das Celulas no servidor.

createMatrizNodeList

O método *createMatrizNodeList* da classe CarregaDados é responsável por:

- Invocar o método getVisibleHeaderLeafsList da classe EstruturaHierarquicaController, de modo a obter quais os nós da estrutura hierarquica da dataTableHeader que correspondem a uma coluna da dataTableMatriz, ou seja, os nós folha visíveis.
- Com esta lista de nós folha e a lista de todos os nós da dataTableContas, constrói uma matriz (ArrayList de ArrayLists), de objectos do tipo Celula. Um objecto do tipo Celula representa uma célula da dataTableMatriz, e contém as ChvPs (chaves

primárias, identificador único) da respectiva linha e coluna, bem como o valor a mostrar na tabela e um boolean que indica se a célula deve ser editável ou não. É sempre editável se corresponder a uma linha da dataTableContas que seja nó folha, ou se o valor obtido pela invocação do método getSelectedRadioOpt devolver "todos".

columnHeaderListener

O método columnHeaderListener é responsável por:

- Receber e tratar os eventos de expansão/contracção das colunas da dataTableHeader.

Funcionamento deste método:

- É detectado um evento de *click* numa coluna da dataTableHeader, através da fonte que originou o evento, obtém-se a ChvP da coluna que foi clicada.
- Com a ChvP da coluna, e através da invocação do método *getElementByChvP* da classe EstruturaHierarquicaController, obtém-se o objecto EstruturaHeader, que deu origem à coluna clicada.
- Invoca o método *getChvPColunasAfectadas* da classe EstruturaHierarquicaController, que cálcula a lista de ChvPs das colunas folha afectadas pela coluna clicada, por exemplo, se a coluna clicada for o Trim 1, as colunas afectadas serão Jan, Fev e Mar.
- Invoca o método *getChvPsTituloToHideOrShow* da classe EstruturaHierarquicaController, que calcula quais são as colunas cuja visibilidade na dataTableHeader deve ser alterada. Se for uma expansão, todos os elementos descendentes do elemento clicado devem ser mostrados, se todos os seus nós antecessores também estiverem expandidos. Se for uma contracção, então todos os elementos descendentes do elemento clicado devem ser ocultados, e surge uma nova coluna que representa essa contracção, por exemplo, se o elemento clicado for um Ano, ficam visíveis as colunas Trim 1-Trim 4 e Jan-Dez em baixo do ano.
- Invoca os métodos *calcNewColsSpansParents* e *calcNewColsSpansChilds*, que calculam os novos colspans para cada elemento da dataTableHeader, devido às alterações de visibilidade que surgiram por acção da expansão/contracção, por exemplo, se o Trim 1 for contraído o seu colspan é alterado de 3 para 1.

- Invoca o método *sumColumnValues* passando-lhe as colunas afectadas, para calcular o somatório das colunas que foram contraídas e coloca-lo na nova coluna, por exemplo, contrair o Trim 1 resulta na nova coluna Jan-Mar e o seu valor passa a ser o somatório das três colunas que foram escondidas.

  No caso de se tratar de uma expansão:

- O comportamento é oposto, voltando ao estado inicial.

- Por fim invoca o método *getChvPHiddenHeaderLeafColumns* da classe EstruturaHierarquicaController para obter uma lista com as colunas que deixaram de ser visíveis da dataTableHeader, e invoca o método *updateMatrixColumns*, para que as colunas correspondentes da dataTableMatriz também se tornem invisíveis.

distribuiValores

- Quando o valor de uma célula é alterado, o método *processValueChange* do evento *valueChangeListener* de um inputText é invocado, que por sua vez invoca o método *updateMatrix*. Se este método detectar que a célula alterada é uma célula de somatório (corresponde a uma coluna contraida) executa a instrução: *setPanelPopupVisible*(true), tornando visível o panelPopup com o id panelPopupDistribuicao definido na página home.xhtml.

- Este panelPopup apresenta duas opções: Distribuir Igualmente e Distribuir Proporcionalmente. Independentemente da opção escolhida será invocado o método distribuiValores.

O método *distribuiValores* é responsável por:

- Invocar o método *getChvPElementAndParentList* da classe EstruturaHierarquicaController, de modo a obter uma lista com as ChvPs da linha alterada da dataTableContas e das que serão afectadas pela propagação do novo somatório para os nós superiores.

- Invocar o método *getIndiceColunaMatrizByChvPConta*, para obter o indice da coluna que contém o somatório, assim com as linhas e a coluna sabemos exactamente quais as células que vão sofrer alterações.

- Invocar o método *getChvPColunasAfectadas*, que dada a coluna em que o somatório foi alterado, vai buscar o nó pai mais acima que estiver contraído, e a partir desse obter as colunas afectadas, para se perceber qual somatório que foi editado, no caso de se ter colunas contraídas dentro de colunas contraídas e consequentemente somatórios de somatórios.
- Invocar o método *propagarDistribuirIgualmenteValores* se a opção escolhida no panelPopup tiver sido Distribuir Igualmente; Invocar o método *propagarDistribuirProporcionalmenteValores* se a opção for Distribuir Proprorcionalmente.

propagarDistribuirIgualmenteValores

- O método propagarDistribuirIgualmenteValores distribui igualmente o novo somatório pelas colunas afectadas, ignorando o antigo somatório. Por exemplo, se o novo somatório for 300 e se as colunas forem Jan, Fev, Mar, cada uma destas ficará com o valor 100.

propagarDistribuirProporcionalmenteValores

- O método propagarDistribuirProporcionalmenteValores guarda o somatório antigo e faz a diferença com o novo somatório. Por exemplo, Jan = 5, Fev = 7, Mar = 9, Total = 21. Se editarmos o total para 30, a diferença é 9 (30 - 21), divide-se esta diferença pelo número de colunas afectadas (3), 9/3 = 3, e distribui-se 3 para cada coluna afectada resultando em: Jan = 8, Fev = 10, Mar = 12.

resetMatrix

O método *resetMatrix* é invocado quando se clica no botão Limpar dados posicionado acima da pivottable e tem a responsabilidade de:

- Invocar o método *resetGivenMatrix* que coloca os valores e somatórios de todas as células a 0.

Capítulo 4

Calendarização de tarefas

Para a realização do projecto foram identificadas e calendarizadas várias tarefas. Esta identificação é visível no planeamento detalhado da

Figura 4.1; a calendarização com base neste planeamento encontra-se no Mapa de Gantt da Figura 4.2, que permite apreender de modo rápido as durações e dependências das várias tarefas identificadas.

Para o controlo do trabalho efectuado foram definidas, semanalmente, as tarefas e objectivos a atingir, e existiu um registo diário numa ferramenta para esse efeito – o XPlanner.

A observação quer do planeamento detalhado, quer do Mapa de Gantt, mostra que a data de fim do projecto seria 12 de Junho de 2007, como realmente acabou por acontecer. Salta à vista a disparidade de tempo entre esta data e a data de entrega do relatório final, situação que lamento e que se deve ao facto de ter tido de começar num novo emprego antes de ter terminado o relatório.

Esta mudança tornou necessário que me deslocasse um mês em trabalho e formação ao estrangeiro, sem computador pessoal ou condições que me tornassem possível continuar o relatório em tempo útil, influenciando a data de entrega.

ID	Nome da Tarefa	Início	Conclusão	Nomes de Recursos
1	**EPM - Enterprise Performance Management**	**10-09-07**	**12-06-08**	**Nuno Sousa**
2	**Investigação**	**10-09-07**	**12-10-07**	
3	Investigação BPM	10-09-07	21-09-07	
4	Apresentação	21-09-07	21-09-07	
5	Investigação Budgeting	24-09-07	12-10-07	
6	**BSC - Balanced ScoreCards**	**24-09-07**	**18-01-08**	
7	Investigação do tema	24-09-07	23-10-07	
8	Documento de proposta de implementação	08-10-07	11-10-07	
9	Apresentação sobre a pequisa e proposta de implementação	12-10-07	12-10-07	
10	Formação sobre plataforma desenvolvimento	17-10-07	17-10-07	
11	Configuração do ambiente de desenvolvimento	18-10-07	19-10-07	
12	Implementação dos ecrãs de suporte	22-10-07	02-11-07	
13	Correcções e actualizações no ecrã Indicadores	12-11-07	20-11-07	
14	Implementação BSC	23-11-07	20-12-07	
15	Formação	04-12-07	05-12-07	
16	Implementação Ecrã Mapa Estratégico	21-12-07	08-01-08	
17	Implementação Ecrã de Análise	08-01-08	18-01-08	
18	Carta de Avaliação	21-01-08	07-02-08	
19	Plano Estratégico	21-01-08	05-02-08	
20	**Planeamento**	**21-01-08**	**14-02-08**	
21	Plano Anual	21-01-08	14-02-08	
22	Plano Plurianual	21-01-08	14-02-08	
23	Testes e correcções	31-01-08	08-02-08	
24	Investigação Gantts	08-02-08	08-02-08	
25	**Forecasting**	**11-02-08**	**12-06-08**	
26	**Investigação**	**11-02-08**	**10-03-08**	
27	Estudo de ferramentas de Forecasting	11-02-08	04-03-08	
28	Demonstração	18-02-08	18-02-08	
29	Demonstração	26-02-08	26-02-08	
30	Demonstração	03-03-08	03-03-08	
31	Demonstração	10-03-08	10-03-08	
32	**Implementação de uma ferramenta de forecasting**	**10-03-08**	**12-06-08**	
33	Investigação JavaServer Faces	10-03-08	21-04-08	
34	Criação do modelo de dados	12-03-08	17-03-08	
35	Estudo das várias implementações JSF	24-03-08	01-04-08	
36	Preparação do ambiente de desenvolvimento e testes	02-04-08	16-04-08	
37	**Implementação de uma pivottable**	**22-04-08**	**10-06-08**	
38	Implementação do header da pivottable	22-04-08	06-05-08	
39	Reunião de discussão do estado da implementação	29-04-08	29-04-08	
40	Criação de uma classe para alimentar a pivottable	05-05-08	05-05-08	
41	Implementação da tabela de contas	06-05-08	15-05-08	
42	Implementação da matriz da pivottable	15-05-08	30-05-08	
43	Reunião de discussão do estado da implementação	29-05-08	29-05-08	
44	Correcções	30-05-08	02-06-08	
45	Optimizações / melhorias gráficas na pivot table	02-06-08	03-06-08	
46	Testes	03-06-08	04-06-08	
47	Correcções	05-06-08	10-06-08	
48	Documentação Pivottable	11-06-08	12-06-08	

Figura 4.1: Planeamento detalhado

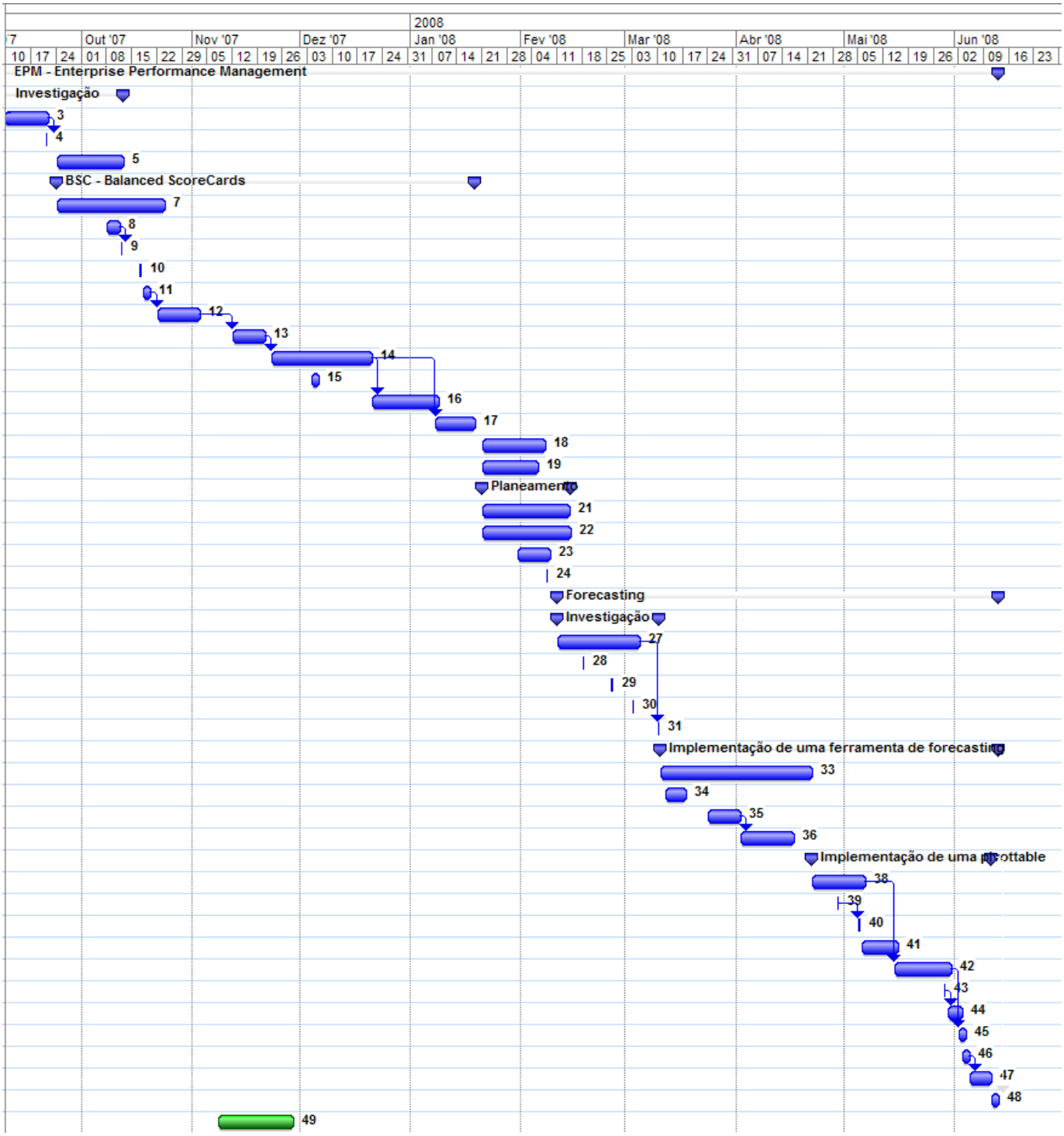

Figura 4.2: Mapa de Gantt

Capítulo 5

Conclusão e trabalho futuro

Neste projecto foi desenvolvida uma ferramenta de BPM (*Business Performance Management*) para apoio à decisão estratégica. Esta tecnologia, embora relativamente recente, já se tornou indispensável para muitas organizações.

No estado da aplicação, ao momento do fim do projecto, não é possível fazer uso da componente de *forecasting*, uma das principais componentes do BPM, uma vez que apenas um protótipo de suporte foi desenvolvido. Após o término do desenvolvimento desta componente seria necessário passar pela fase de integração desta com o sistema SIAG.

A juntar à integração da componente de *forecasting* fica a necessidade de desenvolver as componentes da área de Avaliação e Controlo, já que somente as da área de Instrumentos de Gestão foram desenvolvidas.

Acrónimos

4D	*4th Dimension*
BI	*Business Intelligence*
BPM	*Business Performance Management*
BSC	*Balanced Scorecard*
CEO	*Chief Executive Officer*
CPM	*Corporate Performance Management*
CVS	*Concurrent Version System*
DTD	*Document Type Definition*
EPM	*Enterprise Performance Management*
ERP	*Enterprise Resource Planning*
GEDI	*Gabinete de Estudos e Divulgação Informática*
HQL	*Hibernate Query Language*
HBM	*Hibernate-Mapping*
HTML	*HyperText Markup Language*
J2EE	*Java 2 Platform, Enterprise Edition*
JS	*JavaScript*
JSF	*JavaServer Faces*
JSP	*JavaServer Pages*
MVC	*Model-View-Controller*
OLAP	*On-Line Analytic Processing*
OO	*Object-Oriented*
POCP	*Plano Oficial de Contabilidade Pública*
POJO	*Plain Old Java Object*

PPG *Planeamento do Processo de Gestão*

SGBD *Sistema de Gestão de Base de Dados*

SIAG-AP *Sistema Integrado de Apoio à Gestão para a Administração* Pública

SQL *Structured Query Language*

UI *User Interface*

XML *eXtensible Markup Language*

Bibliografia

Manuais

Framework GEDI

Páginas Web

[1] Adaptive Planning; *Budgeting*, *Forecasting & Reporting*

http://www.adaptiveplanning.com/.

[2] Applix

http://www.applix.com.

[3] BSPG; *opensource Balanced Scorecard*

http://bspg.sourceforge.net/doc/sf.html#features.

[4] Business Objects

http://www.businessobjects.com.

[5] Clarity Systems

http://www.claritysystems.com.

[6] Cognos; *Balanced Scorecard*

http://www.cognos.com/products/cognos8businessintelligence/demos/scorecard/index.html.

[7] E-Visual Report; *Balanced Scorecard*

http://www.e-visualreport.com/files/CMI%20OV%20demo.html.

[8] GEDI

http://www.gedi.pt.

[9] Hyperion

http://www.hyperion.com.

[10] ICEFaces; guia de iniciação

http://www.icefaces.org/docs/v1_6_0/htmlguide/gettingstarted/GettingStartedTOC.html.

[11] Info PM; *Budgeting video*

http://media.infor.com/ProductDemos/PM_Budgeting/Web/997_PM_Budgeting.htm.

[12] JavaServer Faces tutorial

http://www.dsc.ufcg.edu.br/~jacques/cursos/daca/html/jsf/jsf.htm.

[13] JBoss seam framework

http://www.seamframework.org/.

[14] JBoss seam tutorial

http://docs.jboss.com/seam/1.0.1.GA/reference/en/html/tutorial.html.

[15] MBR; *Balanced Scorecard*

http://www.mbr.pt/main.html.

[16] MyEclipse; *ICEFaces tutorial*

http://www.myeclipseide.com/documentation/quickstarts/icefaces/.

[17] Pentaho

http://www.pentaho.com.

[18] QuadBase reporting

http://www.quadbase.com/espressreport/servletdemo.html#.