Departamento de Engenharia Informatica da

Universidade de Évora

Mestrado em Engenharia Informática

XEOReports

António Manuel Filipe de Matos Pinto

Orientadora: Prof. Doutora Irene Pimenta Rodrigues

Maio de 2009

Esta dissertação não inclui as críticas e sugestões feitas pelo júri

UE
171
307

Departamento de Engenharia Informatica da

Universidade de Évora

Mestrado em Engenharia Informática

XEOReports

António Manuel Filipe de Matos Pinto

Orientadora: Prof. Doutora Irene Pimenta Rodrigues

171907

Maio de 2009

Esta dissertação não inclui as críticas e sugestões feitas pelo júri

Prefacio

Es documento contem um trabalho intitulado XEOReports, um trabalho do aluno António Manuel Filipe de Matos Pinto[1], estudante do mestrado em engenharia informática da Universidade de Évora

A orientadora deste trabalho é a Prof. Irene Pimenta Rodrigues[2], do departamento de Informática da Universidade de Évora.

O autor do trabalho é licenciado em Engenharia Informática pela Universidade de Évora.

Esta dissertação foi entregue em Maio de 2009.

[1]L16756@alunos.uevora.pt

[2]ipr@di.uevora.pt

Agradecimentos

A realização deste trabalho contou com a colaboração, directa ou indirecta, de um conjunto amplo de familiares e amigos, aos quais gostaria de agradecer especialmente.

O meu mais profundo agradecimento é dirigido à Eng. Susana Pratas e Eng. João Carreira, Responsável pela Unidade de Serviços e Elemento da Unidade de Engenharia respectivamente, da empresa ITDS – Internet Tecnologias e Desenvolvimento de Software, pelo apoio e estímulo que me transmitiram para a realização deste trabalho.

Gostaria de agradecer aos meus colegas de trabalho e amigos, Francisco Câmara, Túlio Brasil, Marcos Larsen, Olga Ribeiro, António Cruz, Nuno Ferreira, Daniel Silva, Pedro Campos, Ricardo Vieira, Bruno Teixeira, Luís Barreira, Sérgio Precatado, Susana Real, Hugo Pinho, Victor Cardoso, Eliseu Cartaxo e Edgar Delfino pelo apoio sempre demonstrado.

Um agradecimento especial à Professora Irene Pimenta Rodrigues, pelo apoio, disponibilidade e entusiasmo que demonstrou em orientar a minha tese de Mestrado.

Aos meus avós e irmã Ana o meu sincero obrigado por toda a ajuda e carinho demonstrado.

Aos meus pais, apenas tenho a dizer:

"Obrigado por serem os meus pais"

Resumo

Esta dissertação apresenta um projecto em engenharia de software para o desenvolvimento e implementação de um modulo parte integrante da plataforma XEO, denominado XEOReports.

Este módulo destina-se à construção de relatórios dinâmicos, no formato .pdf tendo como base ecrãs de edição da plataforma XEO

Foi utilizada uma plataforma de geração de relatórios em diversos formatos, de nome JasperReports, de forma a que o modulo desenvolvido fosse a integração entre as duas plataformas, XEO e JasperReports.

O desenvolvimento deste modulo foi feito tendo em conta os requisitos que a plataforma jasperReports apresentava para a geração de relatórios tendo como base os ecrãs da plataforma XEO.

O estudo foi feito respeitando a metodologia de desenvolvimento de software UML, repeitando as boas praticas de desenvolvimento de software a ela inerentes.

Abstract

This thesis consists in a software engineering project that deals with the development, functioning and implementation of the XEOReports module, which later became a component of the XEO platform.

The XEOReports module aims the construction of dynamic reports in the Portable Document Format (PDF), based on edition screens of the XEO platform.

JasperReports, an open source reporting engine, which generates reports in several file formats, was also used in the project development. Therefore, the XEOReports module is the result of the two platforms integration, namely XEO and JasperReports.

It is also important to refer that this study took into account the JasperReports platform requirements in the creation of reports based on edition screens of the XEO platform.

Moreover, the development methodology of the UML software, as well as the good development software practices inherent in it, were respected and followed in the progression of this project.

Índice

Índice das Tabelas

Índice das Figuras

1 - INTRODUÇÃO

Em áreas tão variadas como, o meio empresarial ou o mundo académico, a concepção de relatórios, afirmou-se ao longo dos anos como uma ferramenta de grande utilidade para que o leitor obtenha conhecimento sobre uma determinada matéria.

Um relatório pode ser utilizado como elemento de apoio na resolução de uma determinada problemática, levando o leitor mais facilmente a aferir uma conclusão.

A criação de um relatório pode ser resultado de um trabalho de investigação que tenha a necessidade de ser apresentado de forma simples a um determinado leitor alvo. Por norma os relatórios são utilizados como ferramenta informativa, sendo que poderão apresentar algumas características próprias com informações e sugestões adicionais para o utilizador.

Um relatório poderá ser constituído pode os seguintes elementos:

- Capítulos
- Tabelas
- Figuras
- Imagens
- Tabelas de conteúdos
- Sumários
- Apêndices
- Notas de rodapé
- Hiperligações
- Referencias

A construção de um relatório deverá ser feita de forma séria e eficiente e correcta, respeitadas as normas aconselhadas.

Através de um relatório poderão retirar-se muitas conclusões, pelo que devido a este facto, impera a necessidade de que a geração de um relatório seja feita correctamente, caso contrário o leitor pode ser induzido em erro e retirar conclusões erradas da sua leitura.

Existem diversos tipos de relatórios:

- Relatórios científicos
- Relatórios de recomendações
- Relatórios de apoio à decisão
- Relatórios anuais
- Relatórios de auditorias
- Relatórios de trabalho
- Relatórios de censos
- Relatórios de viagem
- Relatórios de investigações
- Relatórios demográficos
- Relatórios militares
- Relatórios vinculativos.

Com o impressionante desenvolvimento das empresas e da competitividade dos mercados, a importância das empresas disporem de conjuntos de informação actualizada e fidedigna, tornou-se uma das principais prioridades para o meio empresarial.

Ao longo dos tempos, tem-se verificado um investimento na área das tecnologias de informação, com o objectivo de se conseguir aferir mais facilmente a informação necessária a uma empresa.

Com uma maior quantidade de informação disponível esta poderá ser cruzada, tirando-se com isso, conclusões mais rápida e facilmente.

Com este desenvolvimento, o termo "Relatório empresarial" ganha forma.

Um relatório empresarial é um conjunto de informação trabalhada e organizada para que, este fique legível para o leitor comum. A informação é retirada de bases de dados, sendo posteriormente trabalhada e organizada para ficar em condições de ser utilizada num relatório.

A plataforma XEO – eXtensivle Enterprise Object é a principal ferramenta de desenvolvimento e criação de aplicações da empresa ITDS – Internet tecnologias e desenvolvimento de software. Esta plataforma é uma ferramenta fundamental no desenvolvimento do projecto da dissertação, o modulo XEOReports que foi feito nas instalações desta empresa.

A ITDS foi fundada em finais de 2002 com a junção de duas empresas.

Desde a sua fundação que se dedica ao desenvolvimento de aplicações com base em tecnologia Web destinadas aos mais variados tipos de negócios e temáticas.

A plataforma XEO começou a ser utilizada no ano 2000, mas inicialmente com o nome Netgest v1.0. Esta versão da plataforma foi utilizada até 2003, e era basicamente desenvolvida em DHTML.

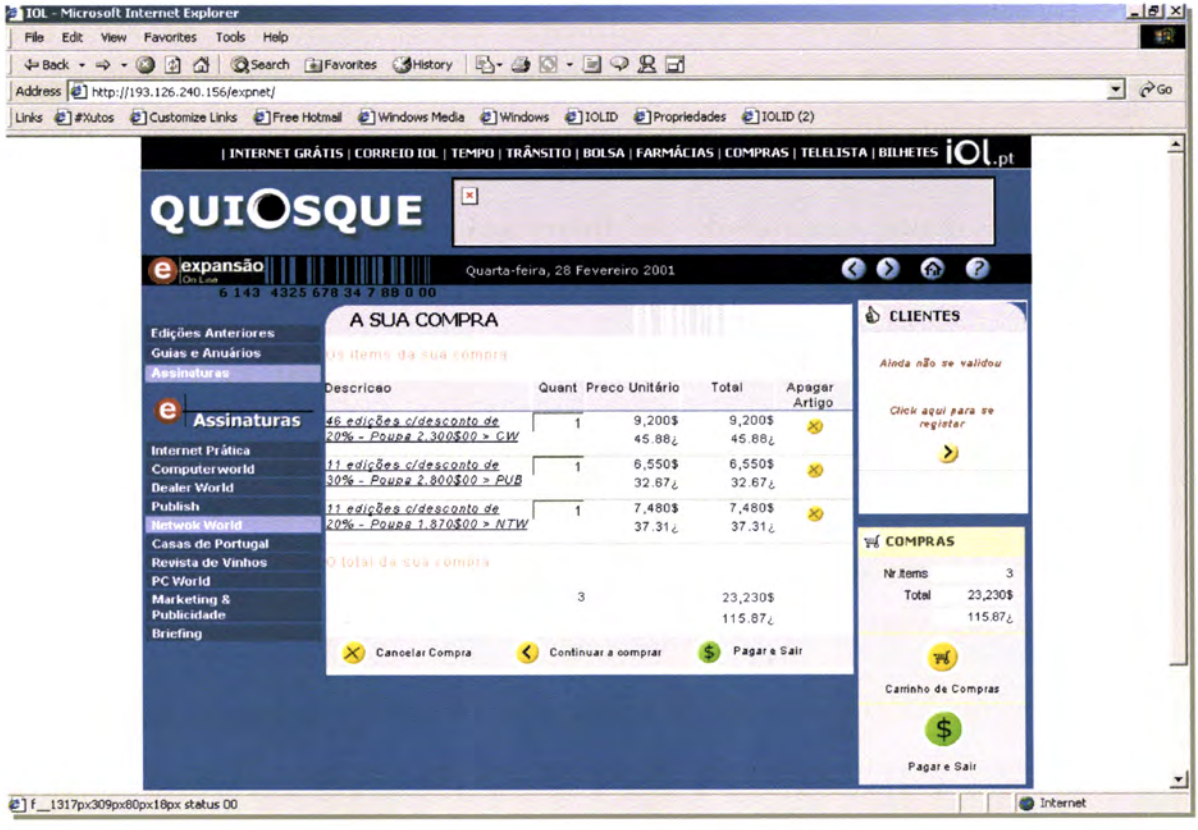


Figura 1: Ecrãs da plataforma XEO - Netgest v1.0

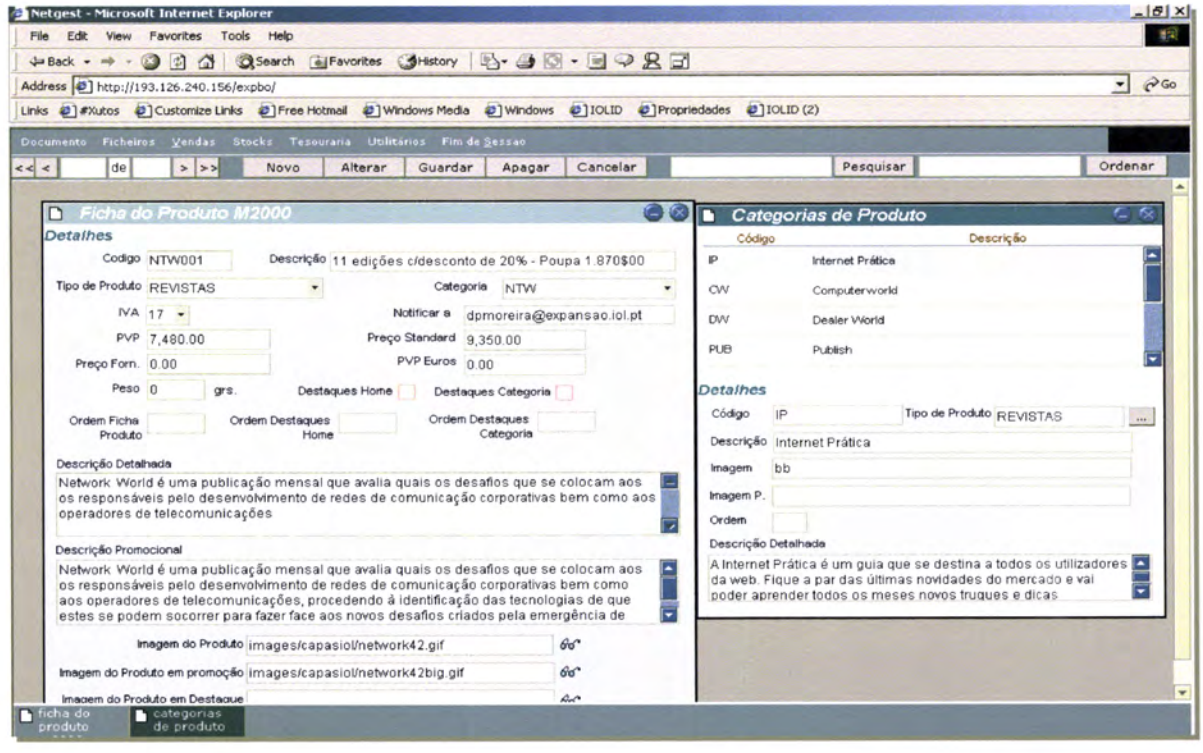


Figura 2: Ecrãs da plataforma XEO - Netgest v1.0

Em 2004, já com a empresa ITDS, a plataforma Netgest v1.0 passou a chamar-se XEO v1.0. Esta versão foi utilizada até ao ano de 2007. Neste ano apareceu a versão 2.0 da plataforma XEO que é a versão utilizada no desenvolvimento deste projecto.

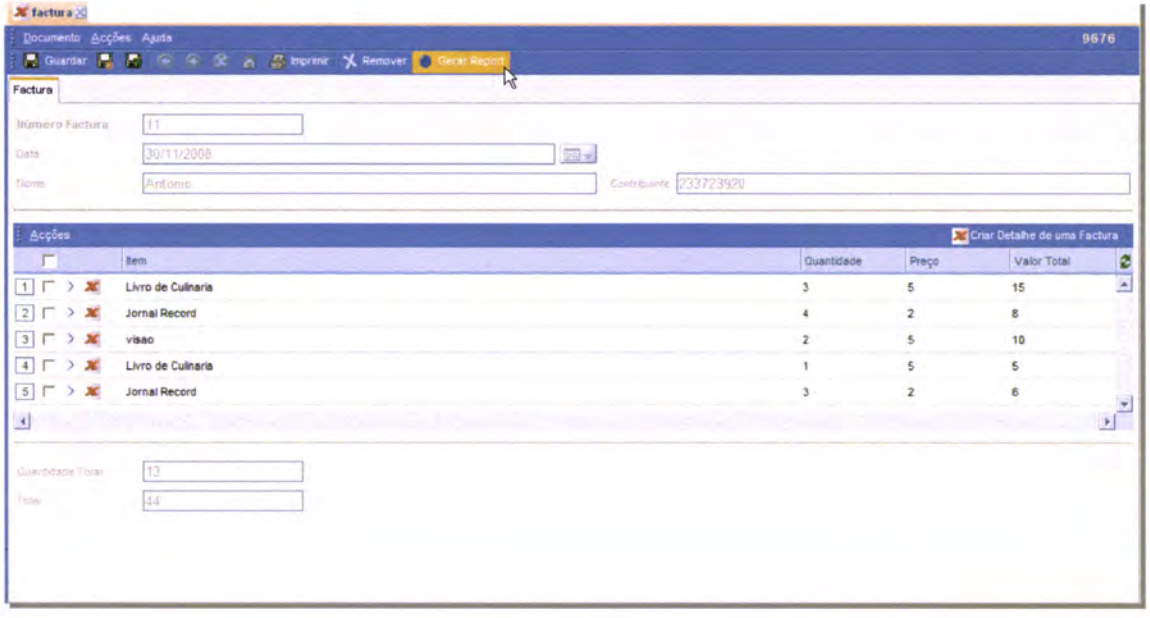


Figura 3: Exemplo de ecrã editável de objecto XEO versão 2.0

Desde a sua fundação, a plataforma XEO foi uma aposta da empresa ITds com o objectivo de proporcionar um desenvolvimento rápido e eficiente das aplicações dos clientes. O desenvolvimento e melhoramento da plataforma XEO foi e é umas das principais linhas de trabalho da ITDS.

Hoje em dia, o mercado empresarial exige que as aplicações apresentem funcionalidades capazes de gerar informação, assegurando deste modo as necessidades do utilizador.

Assim sendo, as empresas que se dediquem à criação de aplicações à medida das necessidades do cliente devem ter presente que possuir uma forma de

criação de relatórios de informação torna-se cada vez mais importante nos dias que correm.

Com o aumento da necessidade de criação e cruzamento de informação, para as diversas empresas que recorrem aos serviços da ITDS, esta passou a ter a necessidade de ver a sua plataforma dotada da capacidade de criação de relatórios que suprimissem essa necessidade.

Certamente atenta a estas questões, a ITDS propôs-me a criação de um módulo de geração de relatórios integrado na plataforma XEO.

Este módulo deverá apresentar a capacidade de replicar, o mais aproximadamente possível, os ecrãs de edição da plataforma, uma vez que estes já são dotados de funcionalidades que oferecem aos utilizadores diversas formas de visualizarem informação.

Com este módulo o programador terá apenas de se focar na forma como os ecrãs serão apresentados, não tendo de se preocupar com a criação de relatórios, cabendo à plataforma XEO essa responsabilidade.

É intenção da ITDS aproveitar as funcionalidades da ferramenta Jasper Relatórios. Esta ferramenta dedica-se à criação de relatórios com informação armazenada em bases de dados, sendo também dotada de muitas funcionalidades que permitem criar relatórios à medida.

Assim sendo, a criação deste ponto de integração torna-se bastante aliciante, uma vez que terá de ser feito um trabalho de investigação, ao nível das funcionalidades e requisitos da plataforma Jasper Relatórios para fazer parte deste módulo.

Por outro lado, o facto do grau de complexidade dos ecrãs de edição da plataforma XEO ser elevado, faz com que o número casos a serem estudados seja elevado, o que leva a um aumento do grau de complexidade do desenvolvimento.

Criar uma aplicação que suprima as necessidades de uma empresa, e que possa ser utilizado em aplicações para os diversos clientes da ITDS, torna o desenvolvimento deste módulo ainda mais interessante.

Esta dissertação encontra-se organizada em cinco capítulos:

- 1º Capitulo – Introdução
- 2º Capitulo - Trabalho relacionado. Neste capítulo serão apresentados temas relacionados com esta dissertação. Tecnologias utilizadas bem como formas de desenvolvimento serão alvo de estudo.
- 3º Capitulo – Trabalho desenvolvido. Este capítulo é dedicado ao estudo detalhado do trabalho efectuado, através da tecnologia UML para o desenvolvimento deste projecto.
- 4º Capitulo - exemplos de utilização. Neste capítulo são apresentados exemplos de uso do módulo XEOReports depois de desenvolvido.
- 5º Capitulo - Conclusões e trabalho futuro. Conclusões sobre o trabalho realizado, e possíveis melhoramentos de futuro, serão temas abordados neste capítulo.
- 6º Capitulo – Referencias bibliográficas. Capitulo onde são apresentadas as referencias bibliografias utilizadas durante a redacção desta dissertação.
- 7º Capitulo – Anexos. Capitulo onde se encontra informação complementar à apresentada na dissertação.

2 - TRABALHO RELACIONADO

Neste capítulo será apresentado a investigação necessário necessidade de ser feito, para o desenvolvimento deste projecto. Ferramentas utilizadas, linguagens de programação e também metodologias de desenvolvimentos serão alvo de estudo neste capítulo.

2.1 XEO – Objectos extensíveis para empresas

O XEO (eXtensible Enterprise Objects) é uma ferramenta de programação ágil e inovadora que a partir de uma notação do tipo UML permite muito rapidamente criar soluções de base tecnológicas para os seus clientes, quase sem programação.

Este processo de interacção entre quem está a desenvolver e quem conhece o negócio permite a criação de soluções adaptadas à realidade das organizações e do seu meio envolvente. Esta plataforma de desenvolvimento foi construída com o recurso às mais recentes tecnologias desenvolvidas para a Internet, apresenta-se como uma plataforma que faz uso intensivo destas tecnologias de informação de modo a obter um bom ambiente de desenvolvimento.

Permite uma rápida e simples integração com as ferramentas da Oracle, nomeadamente com a Base de Dados Oracle e o Application Server.

2.1.1 Integração com outras Aplicações

A integração com outras aplicações pode ser feita através de serviços web, de conectores ou das interfaces das próprias aplicações.

A plataforma XEO é composta por um conjunto de ferramentas de desenvolvimento de fácil e rápida utilização, que permitem o desenvolvimento para aplicações baseadas em ambiente Web.

A tecnologia utilizada no XEO coloca o poder e a funcionalidade que se espera de uma aplicação tradicional em ambientes cliente/servidor com instalação de componentes locais, num Web browser comum sem qualquer instalação de componentes na máquina cliente.

A performance, assim como as potencialidades, é comparável a uma aplicação cliente/servidor.

Permite ciclos de desenvolvimento extremamente rápidos. O desenvolvimento rápido é potenciado por um desenvolvimento baseado em objectos XML parametrizáveis e extensíveis. A plataforma base tem já um conjunto de objectos desenvolvidos que podem ser alterados, estendidos ou integrados com novos objectos de forma a responder às necessidades das aplicações em vista.

Todos os objectos e código desenvolvido podem ser reutilizáveis no desenvolvimento de outras aplicações ou de novas funcionalidades na aplicação existente.

Dispõe de funcionalidades ao nível dos ecrãs de edição que, permitem facilmente ao utilizador desenhar fluxos de trabalho, criar regras de segurança, acrescentar novos campos e parametrizar a solução desenvolvida

sem recurso a elementos técnicos. Estas facilidades permitem uma rápida resposta a alterações registadas como necessárias para a aplicação.

A Base de Dados do XEO constitui um repositório único e centralizado de informação para os negócios de uma determinada organização, onde é possível encontrar os objectos criados numa aplicação criada com esta plataforma. Este repositório, significa que uma fonte de informação poderá ser tanto consultada, como afectada, mediante a utilização da plataforma XEO, de uma forma completamente transparente para o utilizador, beneficiando este de uma maior interoperabilidade, simplicidade e eficiência nas tarefas por si realizadas.

2.1.2 Soluções

Foi desenvolvido um conjunto de produtos, tendo como base a plataforma XEO que de uma forma integrada ou isolada se destacam:

Portal Gestão de Conteúdos – a ferramenta para a gestão de portais permite a gestão dos sites mais simples, onde somente se encontra informação institucional e notícias, até complexos portais de serviços particulares ou governamentais.

Gestão Documental – permite fazer a gestão de todos os documentos da organização e toda a gestão de expediente com entrada e saída de correspondência. Permite a integração com dispositivos de digitalização (scanners), com fax, sistemas de e-mail e SMS.

Gestão Formulários – o XEO dispõe de grandes capacidades para a geração e disponibilização de formulários electrónicos, com vista a sistemas de e-Government.

Workflow – uma das grandes capacidades do XEO é o desenho e implementação de fluxos de trabalho permitindo o seguimento e avaliação

de processos dentro da organização. O Workflow do XEO permite a gestão de filas de trabalho, de grupos de trabalho e a geração de notificações para garantir a qualidade de serviço.

Gestão de Reclamações – tem como principal objectivo a gestão de pedidos, reclamações, participações ou sugestões, com vista a uma rápida resposta das organizações às solicitações dos seus clientes. Para além do registo dos diversos pedidos, a Gestão de Reclamações assenta sobre um sistema de Workflow, que permite fazer o seguimento dos pedidos até à sua satisfação ou resolução, com a resposta ao cliente.

Plataforma de e-Learning – A plataforma de e-learning da ITds permite a criação de cursos on-line, através da Internet ou das Intranets das empresas. Com esta plataforma os conteúdos podem ser geridos de uma forma simples, criando cursos modulares.

Help Desk e Manutenção – A gestão de Help Desk e Manutenção permitem gerir pedidos dentro das organizações e equipas de intervenção para a resolução desses mesmos pedidos. Esta aplicação permite também gerir equipas de intervenção para a manutenção de equipamento como computadores, cadeiras, lâmpadas, ou qualquer outro equipamento dentro da organização.

Com o objectivo de situar a plataforma XEO, no panorama das tecnologias de informação, será benéfico, fazer-se um paralelismo com outras ferramentas que de alguma forma apresentem semelhanças com a plataforma referida. Para uma maior organização, é possível separar duas fazes distintas de procedimentos na plataforma XEO:

Criação e mapeamento de classes Java com tabelas em bases de dados

Construção de um conjunto de JSPs – JavaServer Pages que representam os ecrã de edição de cada objecto.

Em relação ao primeiro ponto, são várias as ferramentas que podemos escolher para fazer um paralelismo das funções das mesmas com o XEO, entre as quais temos por exemplo:

Hibernate – Estrutura de mapeamento dos atributos de uma base de dados relacional e os atributos de um objecto de uma aplicação.

Outsystems – Plataforma de desenvolvimento ágil que permite uma fácil criação, integração e alteração de aplicações.

Sendo que ambas têm objectivos e processos de trabalho muito próximos, de seguida é feita uma comparação entre as duas plataformas, Hibernate e XEO, de forma a auxiliar os leitores a situarem a plataforma XEO nas ferramentas de construção e mapeamento de classes java com tabelas em bases de dados.

2.1.3 Hibernate vs XEO

"O Hibernate é um ambiente para mapeamento objeto-relacional escrito na linguagem Java, mas também está disponível em .Net com o nome Hibernate. Este programa facilita o mapeamento dos atributos entre uma base tradicional de dados relacionais e o modelo objecto de uma aplicação, mediante o uso de arquivos (XML) para estabelecer esta relação.

O Hibernate é um software livre de código aberto distribuído com a licença LGPL."

Wikipédia, a enciclopédia livre - http://pt.wikipedia.org/wiki/Hibernate

Tendo inicialmente começado como um projecto, desenvolvido por programadores espalhados por todo o mundo, o Hibernate, não sendo uma estrutura com as mesmas funções de negocio que a plataforma XEO, assemelha-se contudo à mesma, uma vez que esta ferramenta tem como

Uma vez feitas todas as alterações o programador poderá de forma simples proceder à construção das mesmas no servidor, através da funcionalidade 1-Clik Publish.

A versão final da aplicação ficará imediatamente disponível para ser executada.

2.1.4.1 Pontos em comum entre as duas plataformas.

- Criação de mapeamento de objectos em bases de dados
- Criação de ecrãs de edição específicos referentes a esses mesmos objectos

Centrando-se mais no tema principal desta tese, importa desviar um pouco a temática XEO para contornos que nos aproximam mais da principal temática desta dissertação, como por exemplo fazer uma análise mais detalhada a factores de especial importância para a compreensão deste projecto.

Se entrarmos num estudo mais pormenorizado da plataforma XEO, há a salientar que esta se apresenta desenvolvida e faz uso de quatro principais linguagens de programação:

- Java
- Jsp
- Sql/Boql
- Xml

2.1.5 JAVA

Tendo sido a linguagem de programação que reuniu um maior número de simpatizantes, num curto espaço de tempo, o JAVA apresenta-se nos dias de hoje como uma linguagem de programação que não passa despercebida à maioria dos programadores de software, tendo mesmo sido estabelecidos na internet grupos de desenvolvimento, alguns deles de grandes dimensões, denominados JUGs (Java User Groups).

Na plataforma XEO, Java apresenta-se como uma linguagem usada em processos centrais da plataforma. O núcleo central encontra-se implementado em java, facilitando e proporcionando ao programador, regra geral, um ambiente de programação mais familiar.

2.1.6 JSP (JavaServer Pages)

Fazendo um breve resumo do que consiste esta linguagem de programação, ela tem como objectivo, fazer o controlo do fluxo dinâmico de processos relacionados com WEB Clients, construídos por exemplo em HTML, ou outros tipos de documentos, esta tecnologia permite ao utilizador incluir a linguagem de programação Java para que se consiga produzir acções previamente tipificadas em conteúdos estáticos.

Como já foi dito anteriormente, através da utilização da plataforma XEO, o utilizador não tem necessidade de estar familiarizado com um ambiente demasiadamente técnico, sendo apenas necessário ter conhecimentos na utilização de um Web browser para a interacção com a plataforma XEO.

Methods (métodos)

Esta etiqueta poderá conter diversas etiquetas com um método cada. Estas, por sua vez, destinam-se à inserção de métodos que serão executados mediante a chamada por parte do utilizador através de um botão que aparecerá na interface gráfica correspondente ao método em causa.

A chamada da acção em causa será feita a uma classe java que desencadeará um rol de acções mediante as necessidades do utilizador.

Events (eventos)

Tal como no ponto anterior, esta etiqueta destina-se à inserção de diversos eventos, tantos conforme a necessidade do programador, eventos esses que diferem entre si na altura em que são executados, nomeadamente antes de depois das seguintes acções:

- Carregamento de um objecto já existente.
- Criação de um novo objecto.
- Gravação de um objecto.
- Destruição de um objecto.

onBeforeLoad	Evento que será executado imediatamente antes de ser feito o load de um determinado objecto.
onAfterLoad	Evento que será executado imediatamente depois de ser feito o load de um determinado objecto.
onBeforeCreate	Evento que será executado imediatamente antes de

	ser criado um determinado objecto.
onAfterCreate	Evento que será executado imediatamente depois de ser criado um determinado objecto
onBeforeSave	Evento que será executado imediatamente antes de um determinado objecto ser gravado.
onAfterSave	Evento que será executado imediatamente depois de um determinado objecto ser gravado.
onBeforeDestroy	Evento que será executado imediatamente antes de um determinado objecto ser eliminado.
onAfterDestroy	Evento que será executado imediatamente depois de um determinado objecto ser eliminado.

Tabela – Constituição relevante da etiqueta Events

Em todos estes casos, será chamada uma classe JAVA responsável pelo método que executará o conjunto de acções que o utilizador pretender para um evento em particular.

Attributes (Atributos)

Esta etiqueta é composta por diversas etiquetas que compõem o conjunto dos atributos que o objecto contém.

Estes atributos podem ser de vários tipos.

attributeBinaryData	Representa, um atributo que permite ao utilizador associar a um objecto um determinado ficheiro.
attributeBoolean	Representa um atributo de tipo booleano.
attributeCurrency	Atributo que representa um valor monetário.
attributeDate	Atributo que representa uma data.
attributeDateTime	Atributo que representa uma data e uma hora associada a uma determinada propriedade do objecto.
attributeDuration	Atributo que representa uma determinada quantidade temporal.
attributeLongText	Atributo que representa um texto de dimensões possivelmente superiores a um atributo de texto normal.
	Atributo que representa um determinado valor.

attributeNumber	
attributeObject	Atributo que representa um outro objecto XEO. Este objecto encontra-se associado ao objecto que nos encontramos a definir e terá as mesmas características que definem um objecto XEO.
attributeObjectCollection	Atributo que representa uma colecção de objectos XEO que se encontram associados ao objecto que nos encontramos a definir. Os objectos que compõem este conjunto obedecem a todas as características de um objecto XEO.
attributeSequence	Atributo que representa um valor numerário, mas que por sua vez é calculado automaticamente representando como um valor identificativo de um objecto, como por exemplo o numero de uma factura.
attributeText	Atributo que representa um texto de pequenas dimensões.

Tabela – Constituição relevante da etiqueta Attributes

Viewers (ecrãs)

Esta etiqueta contém os diversos ecrãs de edição que um objecto XEO pode ter. Entenda-se como ecrã de edição o aspecto gráfico que um determinado objecto apresentará ao utilizador através de um Web browser.

Estes viewers podem ser de três tipos:

Edit	Apresenta um determinado objecto escolhido pelo utilizador detalhadamente, ou seja

	indicando todos os atributos que compõem objecto em causa.
List	Apresenta uma listagem de objectos com o resultado de uma pesquisa à base de dados. Esta listagem poderá ser utilizada para posterior selecção de objectos como intuito de o utilizador fazer uma consulta ao detalhe de um objecto. Um objecto poderá ter uma listagem associada, através de uma utilização de uma attributeCollection. Um attributeCollection representa um conjunto de objectos associados a um outro objecto.
Explorer	Apresenta uma listagem de objectos com o resultado de uma pesquisa à base de dados, feita em BOQL (Linguagem apresentada anteriormente). Ao contrario da listagem anterior, esta tem carácter apenas informativo, e apresenta um aspecto diferente da anterior. Este tipo de listagem contém também algumas opções, entre as quais

2.2 JasperReports

Quantos de nós, não nos deparamos já com o problema de colocar um conjunto de informação armazenado numa base de dados, num relatório com o formato PDF?

A JasperReports poderá ser a solução mais adequada para a maioria de nós.

Inicialmente a maioria dos programadores quando tinha de fazer a geração de relatórios de grandes quantidades de informação, utilizavam a ferramenta Itext. Esta é ainda utilizada pela biblioteca JasperReports, que por sua vez veio adicionar uma serie de funcionalidades à geração de relatórios, e de um modo bastante simples e em poucas linhas de código. O JasperReports não é uma aplicação capaz de ser instalada, mas sim uma aplicação para ser incluída numa outra desenvolvida pelo programador, que permitirá mais facilmente a criação de relatórios à "medida".

Tendo como data para o começo do seu desenvolvimento o ano de 2001, através do programador Teodor Danciu, os utilizadores de ferramentas deste tipo, puderam tomar conhecimento do JasperReports, versão 0.1.5, através do site http://sourceforge.net, despertando desde logo muita curiosidade e ganhando uma grande popularidade entre os internautas.

Até aqui o JasperReports resumiu-se a um projecto desenvolvido apenas por uma pessoa, Teodor Danciu, mas depois de se tornar conhecido no mundo, e da formação da companhia JasperSoft, este passou a ser desenvolvido pelos programadores desta companhia. Esta também disponibiliza aos utilizadores outro tipo de suporte e ferramentas ligadas ao JasperReports, como é exemplo o Ireport Visual Designer.

Neste momento, foram descarregadas da internet certa de 300.000 cópias estando a ser utilizado em cerca de 10.000 empresas independentes produtoras de software.

Na mesma lógica que anteriormente, será extremamente benéfico fazer um paralelismo entre o JasperReports e outras ferramentas que se enquadrem também no conceito de ferramentas relacionadas com o design e geração de relatórios.

Assim sendo, e dentro de uma vasto lote de ferramentas deste género, penso que deverão ser estabelecidos alguns critérios de comparação, para que esta seja focada em requisitos deste projecto.

Numa altura em que a criação de software livre está cada vez mais emergente, muitos são os casos em que empresas do mais variado tipo começam a apostar em software deste tipo, sendo a primeira vantagem e talvez a mais evidente, o facto de este ser livre de custos, o que num mercado cada vez mais exigente muitas vezes pode fazer a diferença. Não sendo esta uma regra, uma vez que muitas empresas continuam a apostar em ferramentas comerciais. Se por um lado nos poderá causar alguma estranheza o facto de muitas companhias continuarem a gastar milhares de euros em ferramentas comerciais, por outro, será facilmente compreensível que isso aconteça, já que um investimento poderá ser extremamente rentável, se ele for feito numa ferramenta que ofereça mais garantias de sucesso em relação a outra que seja livre. Ainda assim, e através de alguma investigação pela Internet, facilmente se consegue chegar à conclusão que para este paradigma, se consegue encontrar soluções fiáveis e com muitas funcionalidades sendo ferramentas open source.

Com o objectivo de situar a aplicação jasperReports no contexto das ferramentas de construção de relatórios, será vantajoso fazer-se um paralelismo desde já com alguns critérios de selecção, como por exemplo, a

obrigatoriedade da ferramenta de geração de relatórios a ser escolhida suportar forçosamente a possibilidade de integração em aplicações JAVA.

Assim sendo e depois de alguma investigação, decidi fazer este "cotejo", tendo como referencia o JasperReports, com outra ferramenta gerador de relatórios, O BIRT (Business Intelligence and Reporting Tools).

2.2.2.1 JasperReports vs BIRT

Datado de 2005 o, Business Intelligence and Reporting Tools (BIRT), foi inicialmente um projecto suportado pela Actuate Corporation. Quando esta se juntou à Eclipse Fundation, este projecto passou a ser um projecto desenvolvido pela comunidade Eclipse. Actualmente este é apoiado também pela IBM e pela Actuate como referido anteriormente.

O BIRT é um framework integrado no IDE Eclipse que é responsável pela geração de relatórios em larga escala.

Antes de começar este projecto, e depois de feita uma análise ao objectivo final deste projecto e às tecnologias envolvidas, aferiram-se alguns pontos que têm especial relevância para o desenvolvimento deste projecto.

2.2.1.1 Ferramentas de design de Relatórios

A existência de ferramentas deste género é um ponto bastante importante e que deve ser levado em conta. Para agilizar e facilitar o desenho de relatórios, é necessária uma ferramenta que facilite esse processo. Ainda que o objectivo deste projecto seja a criação dinâmica de relatórios. Neste ponto a acção de um utilizador será necessária, uma vez que será este que

irá fazer a construção desse mesmo relatório com a posterior utilização de um dataSource dedicado constituído por objectos XEO. Através de uma boa ferramenta de design de relatórios, esse processo de construção dos mesmos fica bastante facilitado, não sendo necessário um programador para a elaboração do mesmo. Assim é feita uma separação do trabalho de programador para o trabalho de designer, havendo assim uma distinção de tarefas, ganhando com isso o projecto de uma forma global.

Acerca deste tema, tanto o BIRT como o JasperReports apresentam associados a si ferramentas de design de relatórios que facilitam bastante a criação dos mesmos. Através destas ferramentas é possível fazer a criação de relatórios em minutos. No caso do BIRT este utiliza "Eclipse's graphical editing framework" enquanto que no caso do JasperReports este apresenta um grupo de ferramentas deste género, sendo a mais conhecida o Ireports, uma vez que, é uma ferramenta bastante fácil de utilizar e igualmente como o projecto pai, o JasperReports, uma ferramenta gratuita.

Existem também outras ferramentas comerciais como o Jasper Assistant, baseada em eclipse, mas não com tanto sucesso como o Ireports. Relativamente à ferramenta de design de relatórios do BIRT o Ireports, de certa forma, fica em desvantagem uma vez que em termos visuais esta não tem um efeito suficientemente "amigável", se compararmos com a ferramenta de design de relatórios do BIRT.

Outro dos pontos necessários a ter em conta na escolha de uma ferramenta de relatórios, é a forma como é feita a utilização de fontes de informação, normalmente denominadas de dataSources. Este é talvez o ponto mais importante e com grande peso nesta escolha.

2.2.1.2 Fontes de dados

Antes de descrever os factores que tornam este ponto tão importante, será útil revelar de uma forma geral os tipos de fontes de dados suportados pelas duas plataformas:

	JasperReports	BIRT
JDBC	Sim	Sim
Xml	Sim	Sim
MDX – XML\A	Cross tables são suportadas	Sim
Web Service	Sim	Não
Hibernate	Não	Sim
EJB	Não	Sim
Semi-colon separated values – SSV	Sim	Não
Tab separated values – TSV	Sim	Não
Pipe separated values – PSV	Sim	Não
Comma separated Values – CSV	Sim	Sim
POJO/JavaBeans	Sim	Sim

Tabela - Comparação JasperReports / BIRT

Sendo o XEO uma plataforma autónoma e com objectos de características próprias, o acesso a bases de dados, ou mais pormenorizadamente, a criação de dataSources que sejam responsáveis por providenciar a informação necessária ao preenchimento correcto dos diversos campos de um relatório, afirma-se como um ponto minucioso. A ferramenta que venha a ser inserida neste projecto terá de ter a capacidade da criação de dataSources específicos para diversas situações, e deve permitir também a inserção de objectos XEO nesses dataSources ficando com isso garantida a possibilidade de acedermos às propriedades próprias desses objectos.

Em relação a este ponto, interessa-nos que qualquer que seja a plataforma geradora de relatórios que venha a ser escolhida, ela terá de ter a capacidade de receber um dataSource específico. Neste aspecto tanto o BIRT como o JasperReports, suportam conexões a bases de dados do tipo JDBC (Java Database Connectivity), mas para além disso também apresentam a possibilidade de utilizar um dataSource específico, através do uso de objectos Java com a informação futuramente utilizada no preenchimento de um relatório. Através desta funcionalidade e visto que este projecto se destina a fazer a integração com uma plataforma com características bastante especificas, é possível aquando da geração de relatórios acederemos a características ou funcionalidades próprias de cada objecto XEO, sendo o resultado, relatórios o mais fiáveis e com a maior semelhança possível em relação aos conteúdos dos objectos XEO. Através do JasperReports podemos mesmo personalizar a nossa classe Java representativa de um dataSource, através da implementação de uma interface existente na API do jasperReports chamada JRDataSource.

"The Java Database Connectivity (JDBC) API is the industry standard for database-independent connectivity between the Java programming language and a wide range of databases – SQL databases and other tabular data sources, such as spreadsheets or flat files. The JDBC API provides a call-level API for SQL-based database access."

SUN Developer Network(SDN)
http://java.sun.com/javase/technologies/database/

2.2.2.2 Formatos de relatórios

Tendo presente que uma aplicação útil hoje, amanhã já não o poderá ser, na criação, mais concretamente no projecto de uma aplicação, deve existir a preocupação, não só desta respeitar todos os requisitos necessários no momento, mas também que seja projectada de uma forma a que de futuro esta possa ser facilmente alterável, se assim for necessário. Neste aspecto torna-se importante o factor "formato do relatório". Uma ferramenta de geração de relatórios deverá ter a capacidade de o fazer em diversos formatos, de maneira a que com isso satisfaça o maior número de requisitos. Assim sendo se presentemente temos uma ferramenta que nos constrói relatórios num determinado formato, no futuro poderá vir a ser necessário, outro tipo de formato, devendo para isso haver a preocupação de escolher uma ferramenta que nos dê garantias que isso será possível fazer com o mínimo de dificuldade possível.

Neste aspecto o jasperReports é mais vantajoso uma vez que são vários os formatos em que é possível gerar relatórios:

- HTML,
- PDF,

- XML,
- Excel (multi sheet e single sheet),
- CSV,
- RTF,
- TXT
- ODT

Já em relação ao BIRT este apresenta como formatos possíveis apenas:

- HTML,
- PDF
- CSV
- EXCEL
- RTF
- TXT

Igualmente como o JasperReports, o desenho dos relatórios é também feito sob a forma de ficheiros do formato XML.

Em relação ao ciclo de vida da geração de um relatório, utilizando o BIRT, podemos realçar duas fases distintas, o design de relatório e o preenchimento com a informação necessária do relatório anteriormente "desenhado".

Esse desenho inicial será representado por um ficheiro de extensão *.rptdesign que depois de preenchido dará origem a outro ficheiro desta vez com a extensão *.rptdocument.

Em relação ao jasperReports podemos distinguir 6 fases:

Criação do ficheiro XML com o layout do relatório em construção

Geração da classe JAVA a partir do layout anterior

Compilação da classe com consequente geração de um ficheiro(*.jasper)

Carregamento da classe

Execução da classe e consequente preenchimento dos diversos campos do relatório

Impressão para o formato pretendido.

2.2.2.3 Funcionalidades

Como a maioria das aplicações que se dedicam a criação de relatórios dinamicamente, o jasperReports utiliza os seus templates estruturados em secções como por exemplo:

(traduzir os nomes das secções)

- Title (Titulo)
- Page header (Cabeçalho da página)
- Column header (Cabeçalho da coluna)
- Detail (Detalhe)
- Column footer (Rodapé da coluna)
- Page footer (Rodapé da pagina)
- Summary (Sumário)

Cada secção pode ser trabalhada de forma diferente permitindo que, cada resultado final seja independente dos outros, podendo ser colocados os mais variados elementos, como por exemplo, imagens, blocos de texto estático e dinâmico, bem como, linhas e rectângulos.

2.2.2.4 Construção do ficheiro de formato .jrxml

Como foi mencionado anteriormente, a construção de um primeiro ficheiro que representa o "esqueleto" do relatório que pretendemos construir, não necessita da participação do programador, deixando esta fase apenas para quem desenha os relatórios.

Uma ferramenta bastante útil para utilização nesta fase é o ireports. Esta ferramenta permite ao desenhador criar um relatório em ambiente gráfico e atribuir-lhe todas as funcionalidades que o jasperReports suporta.

Para além disso o designer pode também, e através do mesmo ambiente gráfico testar essas mesmas funcionalidades, criar o ficheiro .jrxml que a aplicação jasperReports necessita para a geração do relatório e compilar este ficheiro, validando-o.

Com isto o tempo que se poderia despender com a criação e teste de um template na geração de relatórios, diminuirá significativamente.

2.2.2.5 Compilação do ficheiro de formato .jrxml / Geração de ficheiro de formato .jasper

A compilação do ficheiro *.xml que representa o desenho de um relatório permite-nos validar possíveis erros que este tenha, mas para além disso, permite-nos também adicionar informação com expressões que o relatório utiliza em runtime.

Com este processo de compilação, é gerado um ficheiro *.jrxml que será utilizado no preenchimento do relatório com a informação vinda da base de dados utilizada.

2.2.2.6 Preenchimento do ficheiro de formato .jasper com informação de uma fonte de dados.

A informação que o jasperReports utiliza para fazer o preenchimento de um determinado relatório, poderá ser acedida de várias formas.

Bases de dados relacionais
Vectores com objectos JAVA
Informação contida em ficheiros XML

Os utilizadores podem criar dataSources à sua "medida", bastando para isso a implementação da interface JRDataSource.

2.2.2.7 Construção do ficheiro final.

Depois da fase de preenchimento, um novo objecto foi gerado, encontrando-se o documento preparado para ser impresso.

Este objecto poderá ser utilizado para gerar documentos nos seguintes formatos:

- HTML
- RTF
- XLS
- ODT
- CSV
- PDF
- XML

2.3 UML

"A Unified Modeling Language (UML) é uma linguagem de modelagem não proprietária de terceira geração. A UML não é uma Métodologia de desenvolvimento, o que significa que ela não diz para você o que fazer primeiro e em seguida ou como projetar seu sistema, mas ela lhe auxilia a visualizar seu desenho e a comunicação entre objectos."

Wikipédia, a enciclopédia livre - http://pt.wikipedia.org/wiki/UML

Uml é uma linguagem específica que se destina a facilitar a análise de um determinado projecto de engenharia de software, separando de uma forma clara duas fases bastante importantes no ciclo de vida de um projecto, a de desenho e levantamento de requisitos, da fase de implementação.

Diagramas de UML são representações gráficas de diagramas onde estão descritas minuciosamente, por exemplo, a forma de implementação do projecto ou também, os seus fluxos tanto de trabalho como de informação que, permitem aos utilizadores obterem uma maior facilidade em visualizar, especificar, construir e documentar todo um conjunto de "peças" que no seu conjunto ou associação resultam num projecto de software.

O uml não deve ser confundido com uma notação que indica ao programador como, nem quais as melhores técnicas a utilizar na implementação de um determinado projecto.

Esta linguagem pode ser utilizada segundo diferentes metodologias, tais como:

- RUP - Rational Unified Process
- FDD - Feature Driven Development
- RAD - Rational Application Developer

2.3.1 RUP

O RUP - Rational Unified Process, foi construído pela Rational Software Corporation, e posteriormente adquirido pela IBM. Na actualidade ele é denominado IRUP – IBM Rational Unified Process.

Com o objectivo de aumentar a produtividade e eficiência no desenvolvimento de um projecto de software, seguindo as linhas mestras deste processo, permite aos membros de uma equipa de desenvolvimento atingir esse ponto de excelência.

Principais linhas de trabalho:

Gestão de requisitos - documentos minuciosamente descritos com todas as restrições e requisitos que o sistema apresente.

Uso de arquitectura baseada em componentes – A utilização de uma arquitectura baseada em componentes permite que a aplicação final seja facilmente extensível. Através do uso de uma arquitectura em componentes, uma aplicação será facilmente alterada, caso se verifique o aparecimento de novos requisitos funcionais.

Uso de software de modelos visuais - Permite uma visão simplificada da implementação de um software, o que permite, a pessoas que estejam menos

habituadas ao domínio das linguagens de programação, uma maior facilidade em compreender como está, ou irá ser implementada a aplicação. Consegue-se também ter uma visão global da aplicação.

Verificação da qualidade do software – É uma das fases mais importantes do ciclo de vida de um software. A qualidade de um software, é verificada envolvendo todos os elementos da fase de desenvolvimento.

Gestão e Controle de Mudanças do Software – As alterações futuras de um software nos dias que correm são inevitáveis. Através da utilização da Métodologia RUP, essas possíveis alterações são tidas em conta aquando da projecção de um software, minimizando com isso o risco e a dificuldade de implementação das alterações referidas.

Conceitos como a implementação por objectos, herança, atributos, etc. devem ser parte importante do conhecimento quotidiano do programador.

2.3.2 FDD

Feature Driven Development – FDD, em português, desenvolvimento dirigido por funcionalidades, tem como elementos principais no desenvolvimento de um software as suas funcionalidades.

Este modelo apresenta as seguintes fases:

Construção de um modelo generalista - este processo destina-se à analise de requisitos, analise orientada por objectos, modelagem lógica de dados, deforma a que seja possível fazer-se um "mapa" de alto nível que a equipa de programadores utilizará durante o desenvolvimento do software.

Construção de um elenco de funcionalidades – nesta fase será feita a divisão de funcionalidades por três áreas distintas:

- Áreas de negócio
- Actividades de negócio
- Passos automatizados da actividade

O resultado deste processo é um conjunto de funcionalidades organizadas hierarquicamente que representam o produto final a ser construído.

Organizações por funcionalidade – nesta fase, são estabelecidas prioridades em relação à implementação das funcionalidades. Nesta avaliação são tidos em conta parâmetros como por exemplo a complexidade da funcionalidade, dependências entre funcionalidades e também quais as funcionalidades que se assumem como prioritárias para o negocio.

Detalhar por funcionalidade – neste ponto da planificação será feita a análise detalhada de como será organizada a a construção da funcionalidade em causa. Pontos como os requisitos da funcionalidade e os seus testes são levados em conta.

Construção da Funcionalidade - fase em que é desenvolvido o código necessário à construção de funcionalidade e são feitos os testes necessários para que esta passe a fazer parte da aplicação em construção.

2.3.3 RAD

Rapid Application Development – RAD, em português, desenvolvimento rápido de aplicações, é um modelo de desenvolvimento de software com um ciclo de vida relativamente curto, entre 60 a 90 dias.

Este tipo de desenvolvimento divide-se nas seguintes fases:

Modelagem de Negócio - Durante esta fase algumas questões deverão ser respondidas:

- Qual o contexto em que se insere o negocio?
- Que informação é gerada no negocio?
- Que entidade gera essa informação?
- Para onde se dirige esta informação?
- Quem processa a informação?

Modelagem dos dados – Nesta fase começa já a ser definidas alguns dos principais objectos, nomeadamente a composição dos mesmos bem como a relação entre eles.

Modelagens do processo – Nesta fase começam a ser implementadas as funcionalidades que permitam realizar operações como adicionar, modificar, recuperar um conjunto de dados.

Geração da aplicação – Este modelo utiliza, na fase de geração da aplicação, componentes já testados que, através da sua utilização, proporcionem uma maior rapidez de desenvolvimento da aplicação.

Testes e modificação – Apesar de este modelo fomentar a reutilização de componentes já testados, o que reduz o tempo de testes, todo o desenvolvimento efectuado terá de ser exaustivamente testados.

3 - TRABALHO DESENVOLVIDO

Este capítulo será dedicado, à análise e concepção de um conjunto de diagramas, utilizando a notação UML, que possibilitem, neste caso ao programador, uma visão abstracta e que demonstre o sistema que se encontra a ser modelado.

Este capítulo será apresentado os seguintes pontos:

Apresentação de uma breve descrição de conceitos como a Notação UML, as suas vertentes e os diagramas utilizados.

Os diversos tipos de Metodologias de trabalho, envolvendo o modelo de linguagem UML serão também analisados.

Todas as funcionalidades do sistema implementado vêm descritas neste capítulo através de diagramas UML. Estes diagramas são gráficos que espelham o conteúdo da visão apresentada ao programador.

Entre os nove tipos de diagramas possíveis de utilizar na linguagem de modelação UML, neste projecto foram utilizados apenas quatro:

- Diagrama de Casos de Uso
- Diagrama de Actividade
- Diagrama de Sequência
- Diagrama de Classes

Diagramas de casos de uso

Diagramas de casos de uso representam um conjunto de acções entre utilizadores e o sistema e como é feita a iteração entre os dois. Este tipo de diagrama é composto pois dois elementos principais, os casos de uso e os actores, sendo que os últimos podem representar não só utilizadores mas também outros sistemas que ajam com a aplicação que esta a ser projectada.

Diagramas de actividade

Diagramas de actividade descrevem a sequência de actividades de um sistema. Estes diagramas têm como objectivo a decomposição de uma actividade em várias actividades tendo em conta o factor temporal na organização das mesmas.

Elementos como casos de uso, operações de uma classe ou uma actividade de mais alto nível, são elementos possíveis de serem descritos num diagrama de actividade.

No projecto em causa, o diagrama de actividades será utilizado para fazer a descrição de actividades que se encontram nos casos de uso de descrevem criação de elementos que constituem a árvore intermédia na construção de um relatório.

Diagrama de Sequência

Este diagrama mostra uma visão simplificada da troca de mensagens entre diversos objectos que façam parte de um sistema, organizando-as por ordem cronológica, de forma a proporcionar ao programador uma noção de como os objectos comunicam entre si para completar uma determinada acção.

Diagramas de classes

Um diagrama de classes tem como objectivo a descrição dos diversos objectos que compõem um sistema e todas as operações e atributos a eles inerentes.

Um diagrama de classes pode ser interpretado de três formas diferentes:

- Conceitual - Forma utilizada especialmente para apresentação a clientes, baseada nos conceitos em estudo e de fácil intepretação.
- Especificação - Apresenta alguns conceitos mais técnicos, relacionados com a arquitectura do sistema a ser implementado.
- Implementação - Aborda diversos aspectos da implementação de uma forma mais específica. Diagramas deste tipo destinam-se à equipa de implementação.

Será apresentada uma descrição exaustiva de todos os casos de uso considerados essenciais neste projecto, sendo que sempre que necessário será também apresentado um diagrama de actividade que facilitará a compreensão de um caso de uso em concreto.

Serão também descritas algumas funcionalidades através da utilização de diagramas de sequência que ajudam a compreender a iteração entre objectos.

No final deste capítulo será apresentada uma estrutura estática das classes que compõem o sistema a ser implementa através de um diagrama de classes.

3.1 Diagramas de Sequência

Antes de fazermos uma análise mais detalhada, de todos os casos de uso aferidos neste projecto, importa ter uma visão mais abstracta de principais processos existentes no modulo XEOReports:

- Visão geral de da construção de um relatório
- Visão sobre a construção de um relatório referente a um ecrã de edição do tipo Edit
- Visão sobre a construção de um relatório referente a um ecrã de edição do tipo List
- Visão sobre a construção de um relatório referente a um ecrã de edição do tipo Explorer

3.1.1 Visão geral de da construção de um relatório

Este ponto destina-se à analise e descrição geral da construção de um relatório, sem ter em conta o tipo de ecrã de edição ao qual se está a construir o relatório.

Todo este processo começa no objecto motor. Este objecto terá de ter a capacidade de determinar qual o tipo de relatório que o utilizador pretende construir. Assim sendo, e depois de interpretar correctamente o tipo de relatório pretendido, o objecto motor deverá também evocar o método responsável pela construção referida.

Chegamos então ao primeiro ponto de selecção, onde a construção do relatório poderá seguir 3 caminhos diferentes.

- Construção de um relatório acerca de um ecrã de edição do tipo Edit
- Construção de um relatório acerca de um ecrã de edição do tipo Explorer
- Construção de um relatório acerca de um ecrã de edição do tipo List.

Neste último ponto o utilizador responsável pelo pedido de construção de um relatório é diferente dos dois anteriores, uma vez que o pedido de impressão de um relatório deste tipo apenas será feito caso na constituição de um ecrã de edição a ser reportado, se encontre um ecrã de edição do tipo List. O pedido de construção de um relatório do tipo List, será sempre da responsabilidade do módulo XEOReports.

Os métodos repontáveis pela construção de relatórios dos ecrãs referidos, serão da responsabilidade do objecto Heart. Esses métodos, serão também eles alvo de uma analise mais detalhada, através de diagramas de sequência nos pontos ...

Depois de terminada toda a acção relativa aos métodos mencionados, estes terão como retorno um objecto do tipo InputStream que será enviado para a plataforma JasperReports, para ser compilado e todo o seu conteúdo ser valido.

Caso essa validação tenha ocorrido com sucesso, a plataforma JasperReports devolverá um objecto do tipo JasperReports, que representa um relatório já compilado e capaz de ser preenchido com informação.

Para isso deverá proceder-se primeiro à construção de uma fonte de dados, adequada para o efeito. O inicio deste processo será também da responsabilidade do objecto Motor. O método necessário para a construção adequada do conjunto de informação necessário, encontra-se no objecto

XEOJRDataSource. Este objecto será um objecto que implementa uma interface presente na plataforma JasperReports, e que proporcionará a utilizador de informação que é também utilizada na plataforma XEO. No final desta chamada será devolvido um objecto XEOJRDataSource que representa a informação para preenchimento do relatório.

Uma vez criada a fonte de informação, o objecto Motor já terá todos os elementos para proceder ao preenchimento do relatório. Para isso este terá de mais uma vez evocar um método da plataforma JasperReports, que através do relatório previamente compilado e da fonte de informação criada, fará o preenchimento do relatório.

Desta acção será retornado um objecto do tipo JasperPrint. Este objecto será utilizado da impressão do relatório para o formato que o utilizador pretender. No nosso caso, formato PDF.

O acto de impressão de um relatório é, igualmente e como a maioria dos métodos referidos, da responsabilidade da plataforma JasperReports, e a sua chamada será mais uma vez da responsabilidade do objecto Motor.

3.1.2 Visão sobre a construção de um relatório referente a um ecrã de edição do tipo Edit

Este ponto destina-se a dar uma visão mais ampla de como decorre a construção de um relatório de um ecrã de edição do tipo Edit.

No inicio deste processo deverá ser criado uma nova instancia do objecto ao qual se está a fazer o relatório. Dele serão retiradas algumas definições necessárias a todo este processo de construção.

Nesta altura, a aplicação estará pronta para iniciar a leitura e análise do código xml com a estrutura do ecrã de edição em causa.

Ao mesmo tempo que decorre a leitura e analise, vai sendo construída a estrutura de apoio em árvore. Este processo é da responsabilidade do objecto ConstroiArvoreRPT_Root, que posteriormente irá devolver o objecto RPT_Root, raiz da árvore construída.

Estes dois pontos serão analisados posteriormente no diagrama de casos de uso e também em diagramas de actividade.

Uma vez construída a árvore, deverá ser feita a sua leitura e consequente escrita de código xml, de que resultará um objecto do tipo StringBuffer e do qual será criado um outro objecto do tipo IntputStream.

3.1.3 Visão sobre a construção de um relatório referente a um ecrã de edição do tipo List

Caso se registe a existência, no contudo de um ecrã do tipo Edit, um ecrã do tipo List, este deverá ser tratado como um novo relatório, ou seja, será construído um sub-relatorio que representará o ecrã do tipo List e que irá fazer parte do relatório principal do ecrã de edição do tipo Edit.

Este processo é muito semelhante ao descrito no ponto anterior, com excepção de depois de ter sido construído o código xml necessário à plataforma JasperReports este deverá ser compilado. Desta compilação resultará um ficheiro de formato .jasper que será alojado numa directoria. Concluído todo este processo será devolvida um objecto do tipo String com a directoria onde estará alojado o ficheiro referido.

3.1.4 Construção de um relatório acerca de um ecrã de edição do tipo Explorer

Este processo, é um misto dos dois anteriores. Todo o processo de construção de código xml, é muito semelhante ao processo de construção para um ecrã de edição do tipo List. As diferenças entre estes dois processos poderão ser aferidas da análise do diagrama de casos de uso. As semelhanças com o mesmo processo mas para um ecrã de edição do tipo Edit, é o facto de que neste caso também será devolvido um objecto do tipo InputStream para ser posteriormente utilizado pela plataforma JasperReports.

3.2 Diagrama de casos de uso.

Para que o diagrama de caso de uso se torne mais claro é benéfico que o mesmo seja separado em componente com o nome de pacotes.

Pacotes do diagrama de casos de uso:

- Acções do utilizador
- Fonte de dados
- Leitor e construtor de XML
- Métodos de Jasper Relatórios.
- Motor

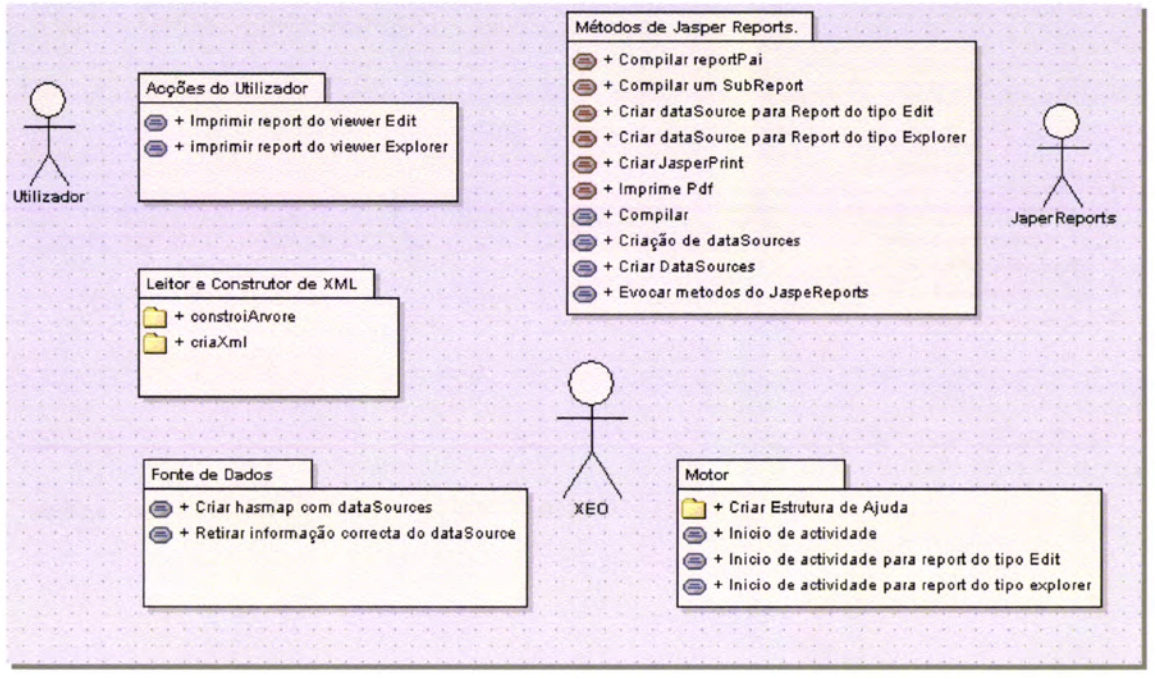


Figura 4: Organização em pacotes do diagrama de casos de uso.

3.2.1 Acções do Utilizador

Este pacote contém as duas actividades que o utilizador dispõe, a criação de um relatório de um ecrã de edição do tipo Edit e também do tipo Explorer.

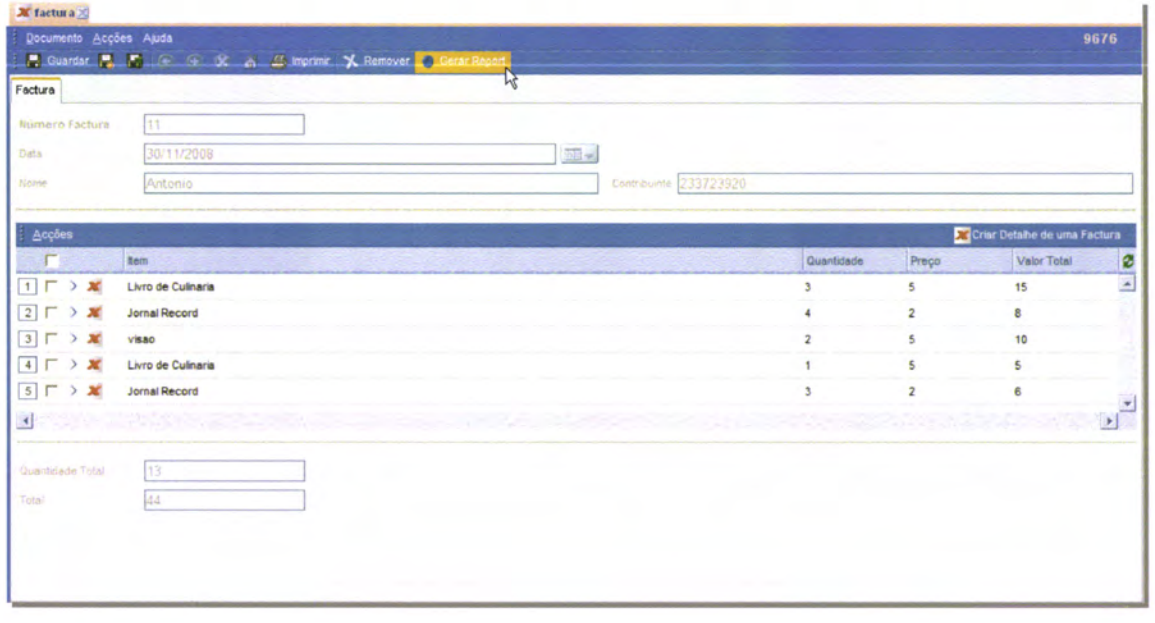


Figura 5: Criação de relatório de Ecrã do tipo Edit

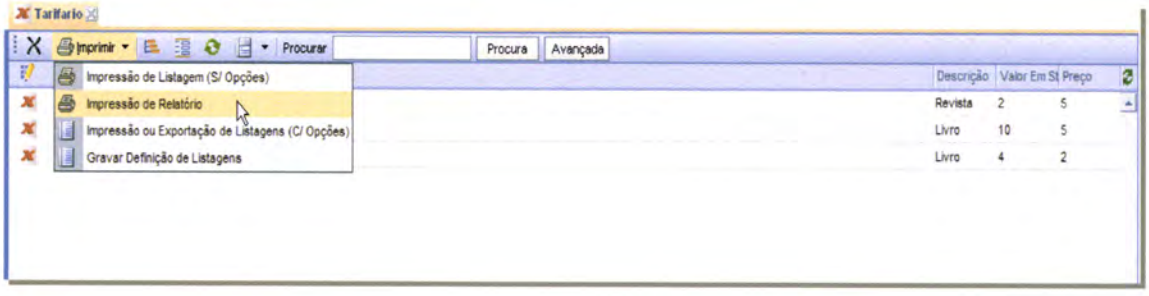


Figura 6: Criação de relatório de Ecrã de Edição do tipo "Explorer"

3.2.2 Heart

Neste pacote encontram-se os casos de uso que representam a interpretação, por parte da plataforma XEO, das acções que são pedidas pelo utilizador que resultam no começo da construção de relatórios do tipo Edit e Explorer respectivamente.

A plataforma XEO deverá estar preparada para receber um pedido de construção de um relatório identificando o tipo e iniciando a construção do mesmo.

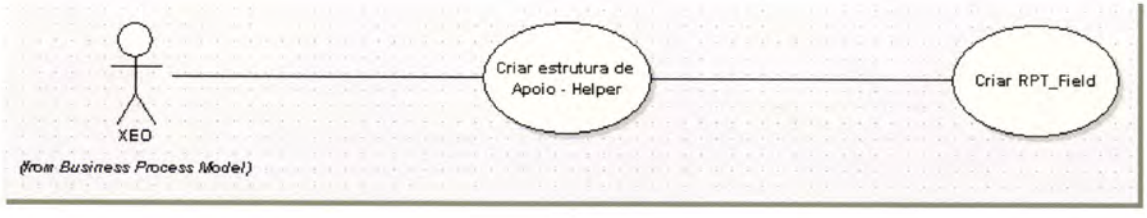


Figura 7: Use case – Criar estrutura de Ajuda

3.2.3 Criar Estrutura de Ajuda

Para a construção de um relatório, será necessário haver uma estrutura de apoio que de forma paralela acompanhe todo este processo de criação da estrutura principal em forma de árvore formando a representação intermédia do relatório. Esta estrutura deverá transportar consigo dados e executar outros casos de uso necessários a todo o processo.

Acções da responsabilidade do objecto RPT_Helper

Armazenar e tornar acessíveis os seguintes conteúdos:

- BOUI do objecto XEO
- Tipo de relatório
- Variável do tipo EboContext utilizada na plataforma XEO
- Directoria onde a aplicação irá aceder ao ficheiro de formato XML que constitui o objecto XEO em causa.
- Número de sub-relatórios criados.
- Nome do objecto XEO que se encontra descrito em um sub-relatório.
- Nome do objecto XEO que se encontra descrito no relatório principal.
- Vector com objectos do tipo RPT_Field.
- Repositório de parâmetros permanentes.

Criação de objectos:

- Criação de objectos do tipo RPT_Field.

Iniciar criação de um sub-relatório:

- Iniciar a criação de um sub-relatório caso a disposição dos atributos do objecto XEO o obrigue.

3.2.4 Fonte de dados

Para ser feito o preenchimento de um relatório com a informação pretendida, terá de ser construído uma estrutura que contenha a informação necessária a esse processo. As acções encarregadas desse processo encontram-se neste pacote.

Criar hashmap com dataSources

Este caso de uso baseia-se na construção de uma nova estrutura que contenha um conjunto de informação para ser utilizada na construção de um novo relatório. Os objectos que esta estrutura irá conter são objectos XEOJRDataSource, que são objectos que implementam uma interface da plataforma JasperReports de nome JRDataSource.

Retirar informação correcta do dataSource

Para fazer um preenchimento correcto o Actor XEO deverá ter a capacidade de retirar a informação que o jasperReports possa pedir para o preenchimento correcto do relatório. Todo este processo é representado por este caso de uso.

3.2.5 Métodos de JasperReports

Ao longo da construção de um relatório, por diversas vezes, como já foi referido, serão efectuadas chamadas à plataforma JasperReports, como o intuito que esta desenvolva determinadas acções, mediante certos requisitos. Assim sendo aplicação, representada neste caso pelo actor XEO, deverá ter a capacidade de fazer a evocação dessas operações.

Todos os casos de uso que sejam chamadas à plataforma JasperReports encontram-se agrupados no pacote "Métodos de jasperReports" e estes são acções da responsabilidade do actor XEO.

Evocar métodos da plataforma JasperReports

Este caso de uso representa a evocação de uma funcionalidade da plataforma JasperReports. Nela podem estar incluídas as seguintes acções:

Imprime PDF – Acção que evoca a impressão no formato PDF depois de ter sido criado o ficheiro no formato .xml com a representação do relatório.

Compilar – Depois de ter sido feito o ficheiro com a representação do relatório este deverá ser compilado pela plataforma JasperReports. Este caso de uso representa a evocação desse processo. Este caso de uso é extensível a duas acções distintas mas com o mesmo fundamento:

- Compilar reportPai
- Compilar SubReport

Criar DataSources – Este caso de uso representa, a evocação da criação de uma fonte dados. Esta fonte de dados é criada através de uma funcionalidade da plataforma JasperReports e a sua chamada é efectuada pelo actor XEO e interpretada pelo actor JasperReports. Este caso de uso é extensível a outros dois casos de uso:

- Criar fonte de dados para relatório de um ecrã de edição do tipo Edit.
- Criar fonte de dados para relatório de um ecrã de edição do tipo Explorer.

Criar JasperPrint – Acção responsável pela criação de um ficheiro de formato .jasperPrint. Este será o penúltimo passo antes da criação do relatório no formato PDF.

3.2.6 Leitor e construtor de XML

Leitor e construtor de XML, representa um conjunto de casos de uso que estão relacionados a leitura ou escrita de código em XML.

Esta aplicação interage com a linguagem XML em duas fases distintas:

Leitura de código XML que representa o objecto XEO e posterior construção gradual da estrutura intermédia em forma de árvore à medida que a leitura vai sendo efectuada.

Escrita de código XML que representa o ficheiro de formato ".jrxml" que será lido pela plataforma JasperReports.

Assim sendo podemos dentro deste pacote podemos criar dois sub – pacotes:

- Leitura de código XML / Construção de estrutura de dados intermédia
- Escrita de Código XML

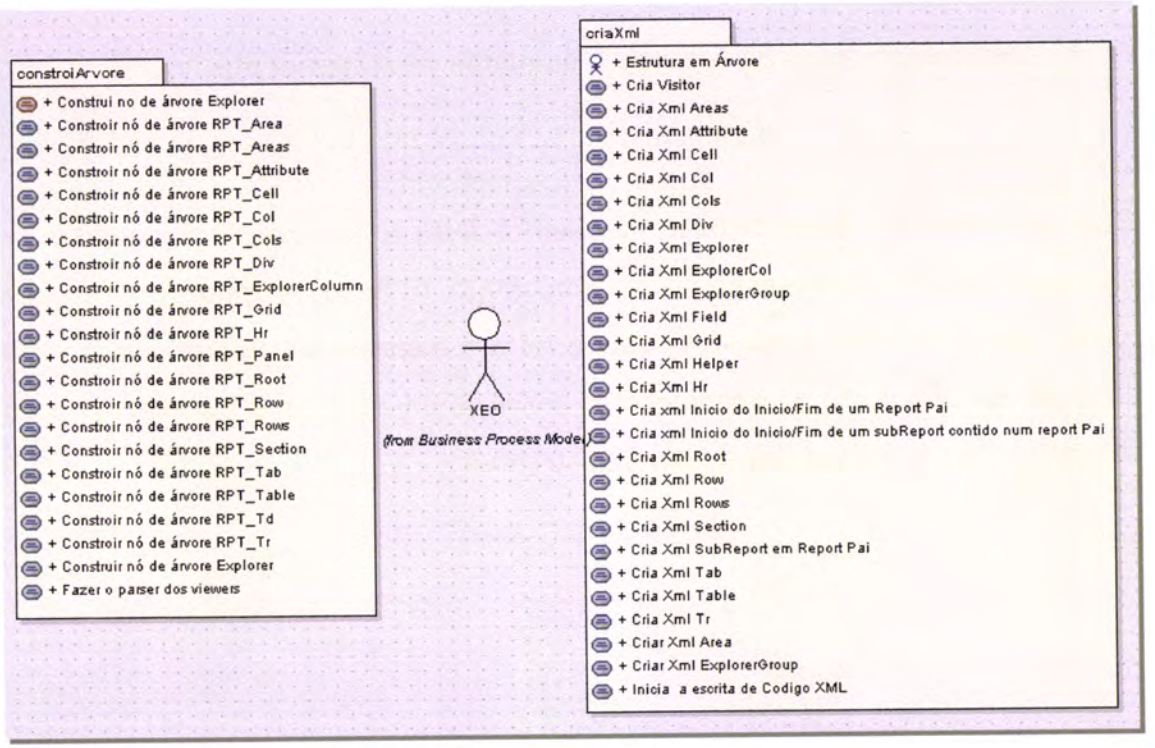


Figura 8: Casos de Uso – Acções de leitura e escrita de Código XML

3.2.6.1 Leitura de código XML / Construção de estrutura de dados intermédia

Através da utilização de classes já existentes na API - Application Programming Interface da plataforma XEO, o processo leitura do ficheiro de formato ".xeomodel" que representa o objecto XEO, fica desde já possível não havendo lugar qualquer tipo de implementação adicional.

Assim sendo este pacote é constituído pelos casos de uso que caracterizam o processo de criação de cada nó da estrutura de dados intermédia.

A criação de cada nó pode ser dividida em duas etapas:

- Leitura e identificação do nó XML que se encontra na descrição no ecrã de edição.
- Construção do nó da estrutura de dados correspondente.

Para uma melhor compreensão de como é feita a construção de um nó da estrutura intermédia devemos ter em conta o factor temporal para que assim seja mais fácil entender como é feita a construção. Assim sendo cada caso de uso referente a construção de cada nó da estrutura intermédia será descrito através de um diagrama de actividade.

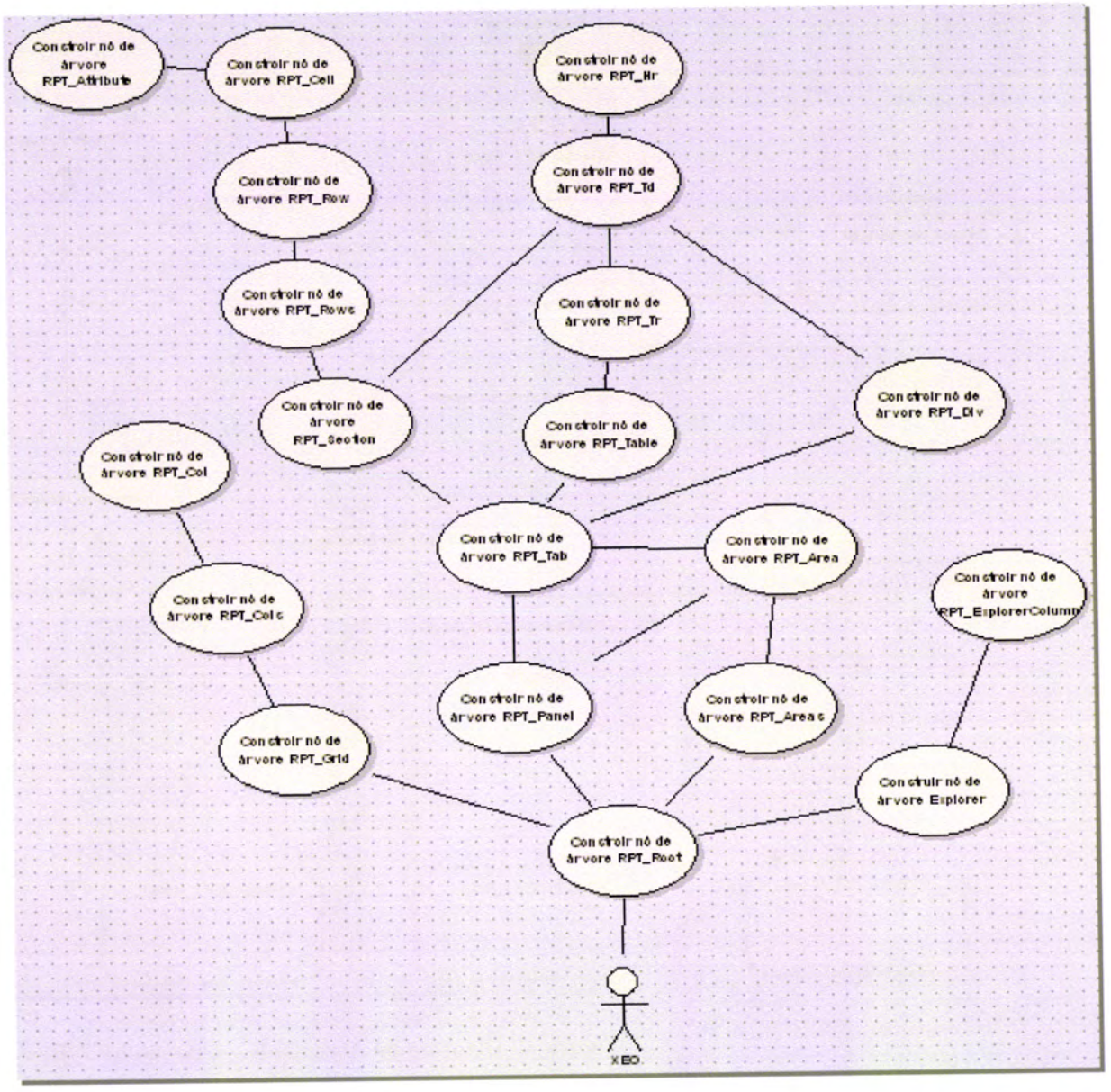


Figura 9: Use case – Constrói nó RPT_Root

Construir nó de árvore RPT_Root

Corresponde à raiz da árvore representativa do ecrã de edição, de um objecto XEO.

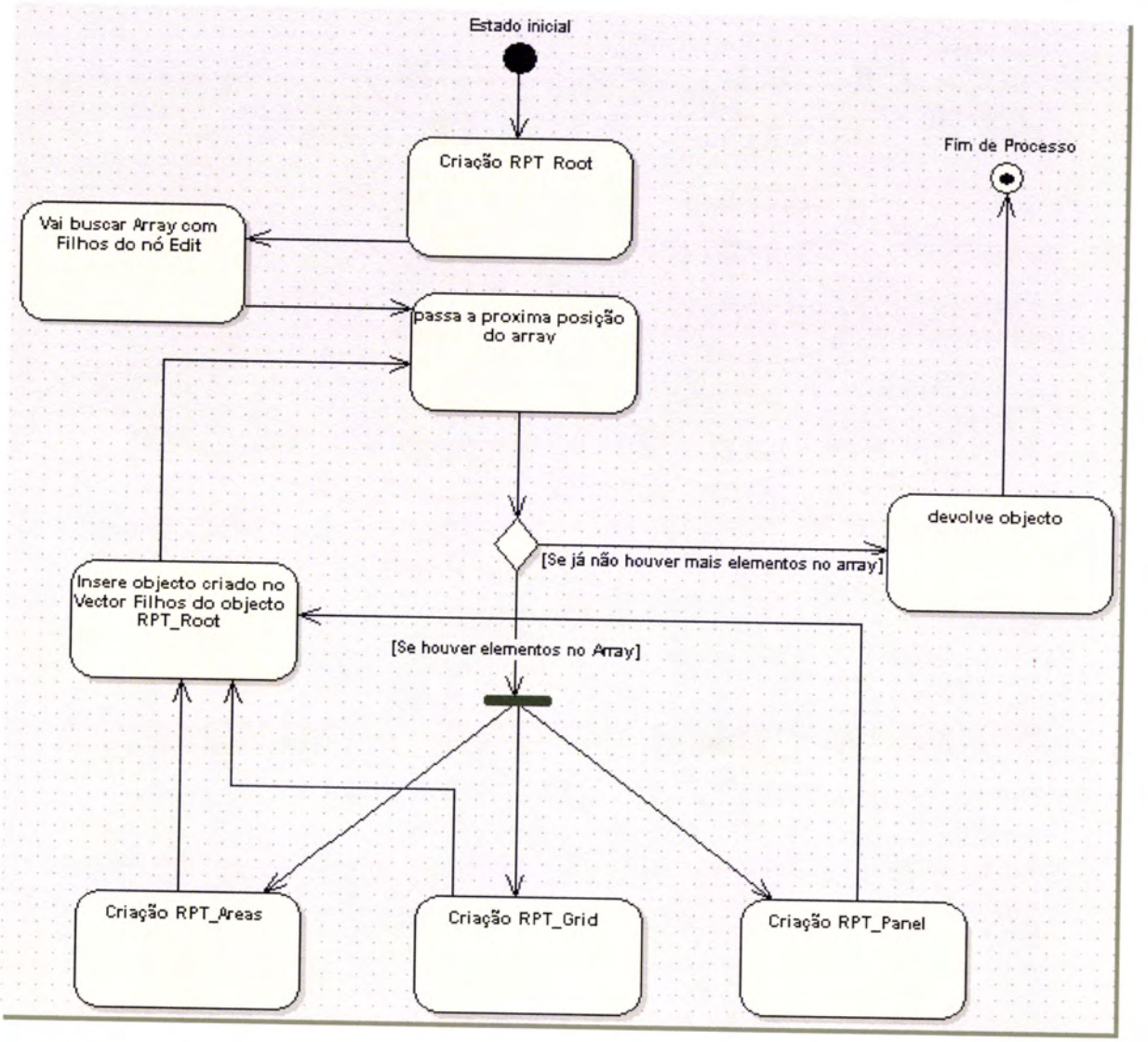


Figura 10:Diagrma de Actividade – Criação RPT_Root

Este caso de uso será descrito através de um diagrama de actividade organizado tendo em conta o factor temporal, com os seguintes pontos:

- Evocação do método de criação de um novo objecto RPT_Root
- Evoca o Vector com os nós XML que constituem o ecrã de edição EDIT
- Passa para o próximo elemento deste vector.
- Verifica se ainda existem ou não elementos no vector a serem tratados.

Depois do último ponto, é feita uma "bifurcação" no processo a ser descrito. Se ainda existirem elementos a ser tratados no vector com os nós xml, serão construídos objectos correspondentes a esses elementos existentes. Casos já não existam mais elementos no Vector, o objecto RPT_Root será devolvido.

Se o vector ainda possuir elementos eles podem representar três tipos de componentes:

- Áreas
- Grid
- Panel
- Explorer

Depois da identificação e de decorrer todo o processo associado a criação do objecto correspondente ao elemento, o objecto será devolvido e inserido no vector de objectos "filhos" do objecto RPT_Root.

Depois de completa esta operação, será executado de novo o ponto 3 descrito anteriormente.

Se já não existirem mais elementos no vector evocado, o objecto RPT_Root será devolvido e todo o processo estará finalizado.

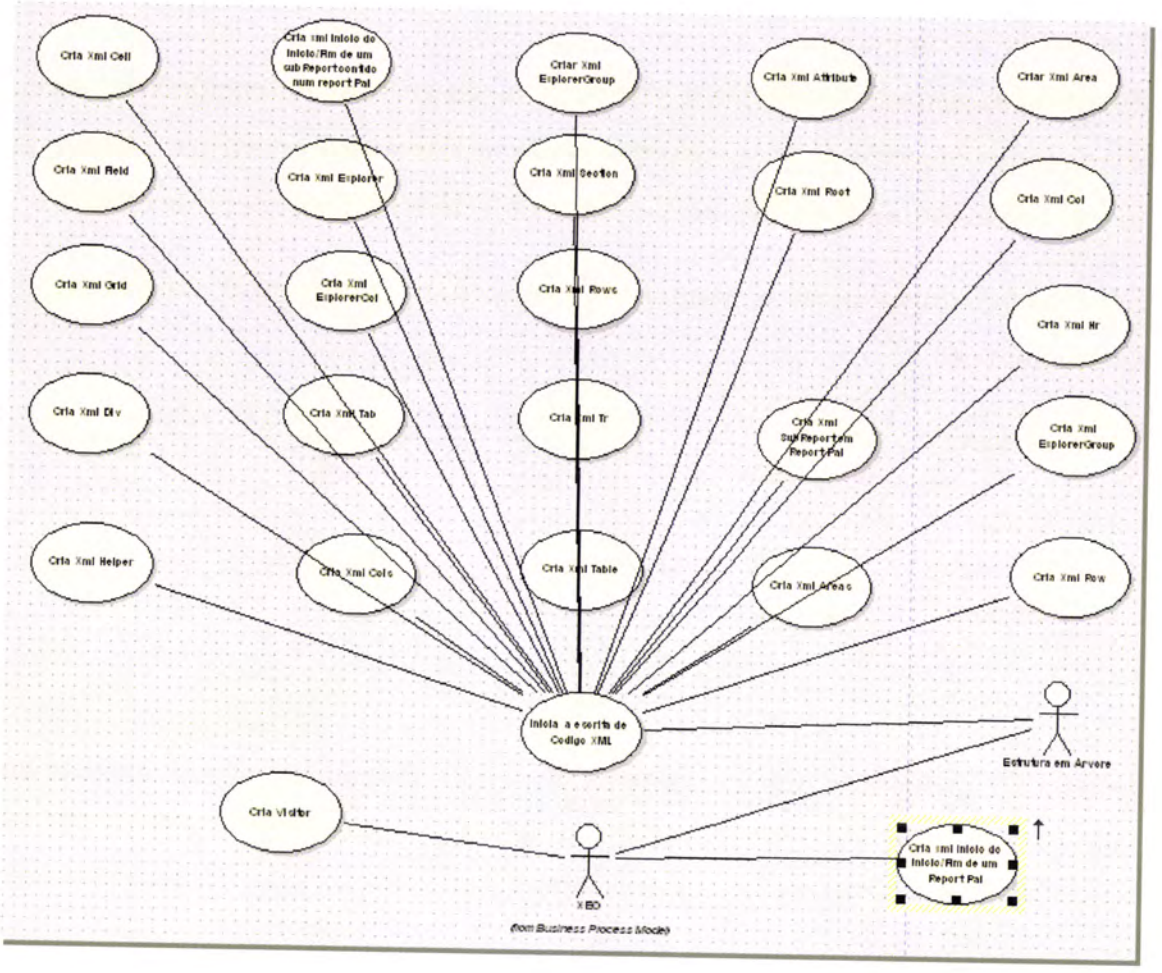


Figura 11: Use case – Cria Xml

3.2.6.2 Cria código em XML

Este pacote representa todas as acções que o Actor XEO desenvolve utilizando uma estrutura em árvore previamente construída.

Estes casos de uso, reflectem os processos de escrita de código do tipo xml. Cada um deles representa a construção de um pedaço de código, que depois de percorrida toda a estrutura em árvore, resultará na informação necessária para a plataforma jasperReports passar à sua compilação.

Estrutura em Árvore

Nesta altura podemos identificar um novo Actor.

A estrutura em árvore, construída através de todos os processos descritos, é será um novo actor e terá um papel de relevo na construção do código xml que será utilizado pela JasperReports.

Cria Visitante

Esta acção destina-se à criação de um patern que tem como objectivo de ser usado na estrutura em árvore. Este objecto irá "visitar" todos os objectos que sejam passíveis de ser visitados, ou seja que implementem a interface "visitable" e que contenham na sua constituição o método "accept", que da ao objecto em causa a propriedade de aceitar ser visitado.

Cria xml de inicio e fim de um relatório principal

Um ficheiro xml, terá certamente alguns pedaços de código que são estáticos.

Isto verifica-se no inicio e no fim do ficheiro, deixando todo o resto dependente da organização das estruturas utilizadas na criação de um relatório.

Este Caso de Uso representa essa escrita de código, no inicio e fim do ficheiro, pelo que pode ser executado em diferentes alturas da construção do relatório.

Inicia a escrita de Código XML

O actor XEO deverá evocar o método responsável pela criação de código xml existente pattern Visitor.

Para evocar esta impressão O Actor, XEO, deverá passar ao método a ser chamado um objecto RPT_Root, para que recursivamente, seja escrito o código xml deste objecto bem como dos objectos que se encontram no vector "vectorFilhos" do mesmo.

Cria Xml de um elemento Helper

Ao encontrar um objecto RPT_Helper na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

Para

Cria Xml de um elemento Root

Ao encontrar um objecto RPT_Root na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

Consultar restantes elementos em Anexo B.

3.3 Diagrama de Classes

Uma vez feito o digrama de casos de uso, estão criadas as condições para se determinar quais as classes a serem criadas para este projecto. De seguida

serão apresentados os objectos criados, bem como, todas as suas propriedades.

Tal como em diagramas anteriores, foi feita uma organização por conjuntos, dos diversos elementos que compõem este diagrama:

- Fonte de Dados
- Motor
- Métodos de Jasper Relatórios.
- Leitor e Construtor de XML
- Constrói Árvore
- Cria Xml
- Parser Objecto

3.3.1 Fonte de Dados

XEOJRDataSource – Objecto que representa a fonte de informação necessária ao relatório. Este objecto deverá implementar a Interface JRDataSource pertencente à plataforma JasperReports.

Atributos:

boObjectList olist – Objecto pertencente à plataforma XEO que representa uma lista de objectos XEO.

Construtor:

XEOJRDataSource(boObjectList); - Construtor responsável pela criação de um novo objecto XEOJRDataSource que tem como argumento de entrada um objecto do tipo boObjectList.

Métodos:

getFieldValue(JRField); - Método responsável por aceder e devolver a informação correcta para um determinado elemento do relatório.

Parâmetro de entrada – Objecto do tipo JRField

Parâmetro de retorno – Objecto do tipo Object

moveFirst(); - método responsável por colocar o cursor da lista de objectos no primeiro objecto da lista

next(); - Método responsável por mover o cursor da lista de objectos para o próximo objecto.

CreateHashMap – Objecto responsável pela criação de fontes de dados adequadas a diversas situações.

Atributos:

Map hashMap – Objecto que irá conter todas as fontes de dados, do tipo XEOJRDataSource, criadas para um determinado relatório.

Construtor:

CreateHashMap(); - Construtor responsável pela criação de um novo objecto CreateHashMap.

Métodos:

createDataSourcePai(RPT_Helper helper); - Método responsável por criar uma fonte de dados para o preenchimento de um relatório principal.

Parâmetro de entrada – Objecto do tipo RPT_Helper

3.3.3 Leitor e Construtor de XML

CoordenacaoParser – Objecto de coordenação entre a leitura de códi vindo de objectos XEO, criação da estrutura intermédia em árvore, de novo código xml para a plataforma JasperReports e chamada de métodos da plataforma JasperReports.

Atributos:

int numeroSubReports – Numero de sub - relatórios construídos.

String directoria – Directoria para colocação de ficheiros temporários.

RPT_Helper helper – Objecto de apoio ao processo do tipo RPT_Helpe

Construtor:

CoordenacaoParser(); - Construtor responsável pela criação de un objecto CoordenaçãoParser.

Métodos:

parserObject(boObject object, RPT_Helper helper); - Método de responsável pelos processos mencionados no inicio deste ponto, para de edição do tipo Edit respectivamente.

Parâmetro de entrada – Objecto XEO do tipo boObject, objecto de a tipo RPT_Helper.

Parâmetro de retorno – Objecto do tipo InputStream.

parserObject(Explorer explorer, RPT_Helper helper); - Método de responsável pelos processos mencionados no inicio deste ponto, para de edição do tipo Explorer respectivamente.

Parâmetro de entrada – Objecto XEO do tipo Explorer, objecto de a tipo RPT_Helper.

createDateSourceBridges(RPT_Helper helper, boObject objecto); - Método responsável por criar uma fonte de dados para o preenchimento de um relatório de um ecrã de edição do tipo List, caso este exista num relatório principal.

Parâmetro de entrada – Objecto do tipo RPT_Helper

criaLista(long bouiObjecto, String nomeObjecto, EboContext boctx); - Método responsável pela criação de uma lista de objectos XEO que vai ser utilizada na criação do objecto XEOJRDataSource.

Parâmetro de entrada – Objecto do tipo EboContext, nome do tipo de objectos que a lista irá conter, e o identificativo do objecto.

Parametro de retorno – Objecto do tipo boObjectList

devolveDataSourcesHashMap(RPT_Helper helper, boObject objecto) – Método responsável pela devolução do objecto do tipo Map, que deverá conter todas as fontes de dados necessárias para o preenchimento de um relatório.

Parâmetro de entrada – Objecto do tipo RPT_Helper, objecto XEO Auxiliar.

Parâmetro de retorno – Objecto do tipo Map

3.3.2 Motor

Motor – Objecto que inicia a criação de um relatório. Tem a capacidade para identificar e iniciar qualquer tipo de relatório.

Atributos:

EboContext boctx – Objecto XEO do tipo EboContext.

String directoria – Directoria para colocação de ficheiros temporários.

RPT_Helper helper – Objecto de apoio ao processo do tipo RPT_Hel

Construtor:

Motor(); - Construtor responsável pela criação de um novo objecto M

Métodos:

startPdfPrint(boObject objecto, String nomeForm, Outp outputStream); - Método de inicio à construção de um relatório de de edição do tipo Edit.

Parâmetro de entrada – Objecto XEO do tipo boObject, String com ecrã de edição, objecto do tipo OutputStream.

startPdfPrint(Explorer explorer, EboContext contexto, Outp outputStream); - Método de inicio à construção de um relatório de de edição do tipo Explorer.

Parâmetro de entrada – Objecto XEO do tipo Explorer, Objecto XE EboContext, objecto do tipo OutputStream.

startPdfPrint(String nomeFicheiro, String nomeForm); - Método c construção de um relatório de um ecrã de edição do tipo Explorer.

Parâmetro de entrada – String com o nome do ficheiro, String com ecrã de edição.

finalizaDetail(StringBuffer xmlCode, RPT_Root root)– Método que constrói o código xml relativo ao final de um "Detail", elemento de um relatório.

Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer, int comprimento_detail, objecto do tipo RPT_Root.

I_Visitable – Interface à qual todos os objectos anteriormente descritos deverão implementar.

Métodos:

accept(I_Visitor visitor); – Método que indica que um objecto aceita ser visitado, ou por outras palavras, é possível ser acedido e com isso desenvolver um conjunto de acções que no nosso caso são a construção de código xml, de acordo com o objecto em causa.

Parâmetro de entrada – interface do tipo I_Visitor

I_Visitor – Interface que o objecto "PrintVisitor" deve implementar.

Métodos:

visit(RPT_Root root); – Método que acciona a visita a um obejecto RPT_Root

visit(RPT_ElementAreas areas); - Método que acciona a visita a um objecto RPT_ElementAreas

visit(RPT_ElementArea area); - Método que acciona a visita a um objecto RPT_ElementArea

visit(RPT_ElementPanel panel); - Método que acciona a visita a um objecto RPT_ElementPanel

visit(RPT_ElementTab tab); - Método que acciona a visita a um objecto RPT_ElementTab

visit(RPT_ElementSection section); - Método que acciona a visita a um obejecto RPT_ElementSection

visit(RPT_ElementRows rows); - Método que acciona a visita a um objecto RPT_ElementRows

visit(RPT_ElementRow row); - Método que acciona a visita a um objecto RPT_ElementRow

visit(RPT_ElementCell cell); - Método que acciona a visita a um objecto RPT_ElementCell.

visit(RPT_ElementAttribute attribute); - Método que acciona a visita a um obejecto RPT_ElementAttribute.

visit(RPT_Helper helper); - Método que acciona a visita a um objecto RPT_Helper.

visit(RPT_Field field); - Método que acciona a visita a um objecto RPT_Field.

visit(RPT_ElementDiv div); - Método que acciona a visita a um objecto RPT_ElementDiv.

visit(RPT_ElementGrid grid); - Método que acciona a visita a um objecto RPT_ElementGrid.

visit(RPT_ElementCols cols); - Método que acciona a visita a um objecto RPT_ElementCols.

visit(RPT_ElementCol col); - Método que acciona a visita a um objecto RPT_ElementCol.

visit(RPT_ElementTable table); - Método que acciona a visita a um objecto RPT_ElementTable,

visit(RPT_ElementHr hr); - Método que acciona a visita a um objecto RPT_ElementHr,

visit(RPT_ElementTr tr); - Método que acciona a visita a um objecto RPT_ElementTr,

visit(RPT_ElementTd td); - Método que acciona a visita a um objecto RPT_ElementTd,

visit(RPT_ElementExplorerColumn explorerColmun); - Método que acciona a visita a um objecto RPT_ElementExplorerColumn.

visit(RPT_ElementExplorer explorer); - Método que acciona a visita a um obejecto RPT_ElementExplorer.

visit(RPT_ElementExplorerGroup explorerGroup); - Método que acciona a visita a um obejecto RPT_ExplorerGroup.

PrintVisitor – Objecto que implementa a interface anteriormente descrito. É nele que é feito o controle de da construção correcta de código xml, dependendo de qual o objecto que se encontra a ser visitado.

3.3.4 XEORptElements

Em relação aos objectos que compõem a estrutura em forma de árvore, eles apresentam muitas semelhanças entre eles, entre as quais o facto de todos estenderem o objecto RPT_Root e implementarem a interface I_Visitable.

RPT_Root – Objecto que representa a raiz da estrutura em arvore.

Atributos:

String nodeName; - nome do nó xml pertencente ao ecrã de edição

String label; - Nome identificativo de um determinado campo que aparece no relatório

Vector vectorFilhos; - vector onde vão ser colocados os objectos que constituem a disposição deste elemento.

int flag; - Variável que tem um papel importante no constrolo do fluxo de escrita de código xml relativo a este objecto. Determina se deve ser escrito o inicio ou final de uma etiqueta xml.

String ambiente; - Variável que contém o tipo de ecrã de edição ao qual se está a construir o relatório.

Construtor:

```
public RPT_Root()
 {
   this.nodeName    = null;
   this.label       = null;
   this.vectorFilhos = new Vector();
   this.flag        = 0;
   this.ambiente    = "";
 }
```

Métodos:

accept(I_Visitor visitor); – Método que acciona a construção de código xml relativo a este objecto.

getAmbiente(); - Método devolve o tipo de ecrã de edição que está a ser construído o relatório.

setAmbiente(String ambiente); - Método afecta a variável ambiente com o tipo de ecrã de edição, de que está a ser construído do relatório.

getLabel(); - Método que devolve uma "String" com o identificativo a constar no relatório, deste objecto.

setLabel(String label); - Método que afecta o atributo "label" com o identificativo a constar no relatório, deste objecto.

getFlag(); - Método que retorna a variável "flag" que faz o controlo de inicio e final de construção de uma etiqueta xml.

setFlag(String flag); - Método que afecta a variável "flag" que faz o controlo de inicio e final de construção de uma etiqueta xml.

getNodeName(); - Método que retorna a variável "NodeName" com o nome do nó representado neste objecto.

setNodeName(String nodeName); - Método que afecta a variável "NodeName" com o nome do nó representado neste objecto.

getVectorFilhos(); - Método que retorna o vector com todos os objectos que fazem parte deste objecto.

De seguida serão apresentados os objectos ao estenderem o objecto RPT_Root herdam tanto os atributos como os métodos deste:

- RPT_ElementAreas
- RPT_ElementArea
- RPT_ElementAttribute
- RPT_ElementCell
- RPT_ElementCol
- RPT_ElementCols
- RPT_ElementDiv
- RPT_ElementExplorer
- RPT_ElementExplorerCollumn
- RPT_ElementExplorerGroup
- RPT_ElementGrid
- RPT_ElementHr

- RPT_ElementPanel
- RPT_ElementRow
- RPT_ElementRows
- RPT_ElementSection
- RPT_ElementTab
- RPT_ElementTable
- RPT_ElementTd
- RPT_ElementTr
- RPT_Helper
- RPT_Field

3.3.5 Objectos de chamada à plataforma Jasper Relatórios.

ChamadasAoJasper – Objecto que contem métodos de chamada a operações da responsabilidade da plataforma JasperReports.

Métodos:

compilaReport(InputStream input); - Método responsável por evocar a compilação de um relatório.

Parâmetro de entrada – Objecto do tipo InputStream com o relatório a ser compilado.

compilaSubreport(InputStream input, OutputStream output); - Método responsável por evocar a compilação de um sub - relatório.

Parâmetro de entrada – Objecto do tipo InputStream com o relatório a ser compilado e objecto do tipo OuputStream.

criaDataSource(boObjectList lista); - Método responsável por evocar a criação de uma nova fonte de dados.

Parâmetro de entrada – Objecto XEO do tipo boObjectList.

devolveJasperPrint(JasperReport JasperReport, Map hm, XEOJRDataSource dataSource); - Método responsável por devolver um objecto do tipo jasperPrint.

Parâmetro de entrada – Objecto do tipo JasperReport com o relatório compilado, objecto do tipo "Map" com toda a informação necessária ao preenchimento de possíveis sub – relatórios e um objecto XEOJRDataSource com a informação necessária ao relatório principal.

imprimePdf(JasperPrint jasperPrint, String directoria, OutputStream outputStream); - Depois de concluídos todos os métodos anteriores este método é responsável pela criação de um ficheiro, no formato pdf, como o relatório pretendido.

Parâmetro de entrada – Objecto do tipo JasperPrint com o relatório compilado, objecto do tipo "String" com a directoria onde irá ficar o ficheiro criado e um objecto do tipo OutputStream.

Consultar restantes objectos em Anexo D.

4 - EXEMPLOS DE UTILIZAÇÃO

De seguida serão apresentados exemplos de relatórios que reproduzem ecrãs de edição da plataforma XEO, através da descrição dos passos que o utilizador deverá efectuar.

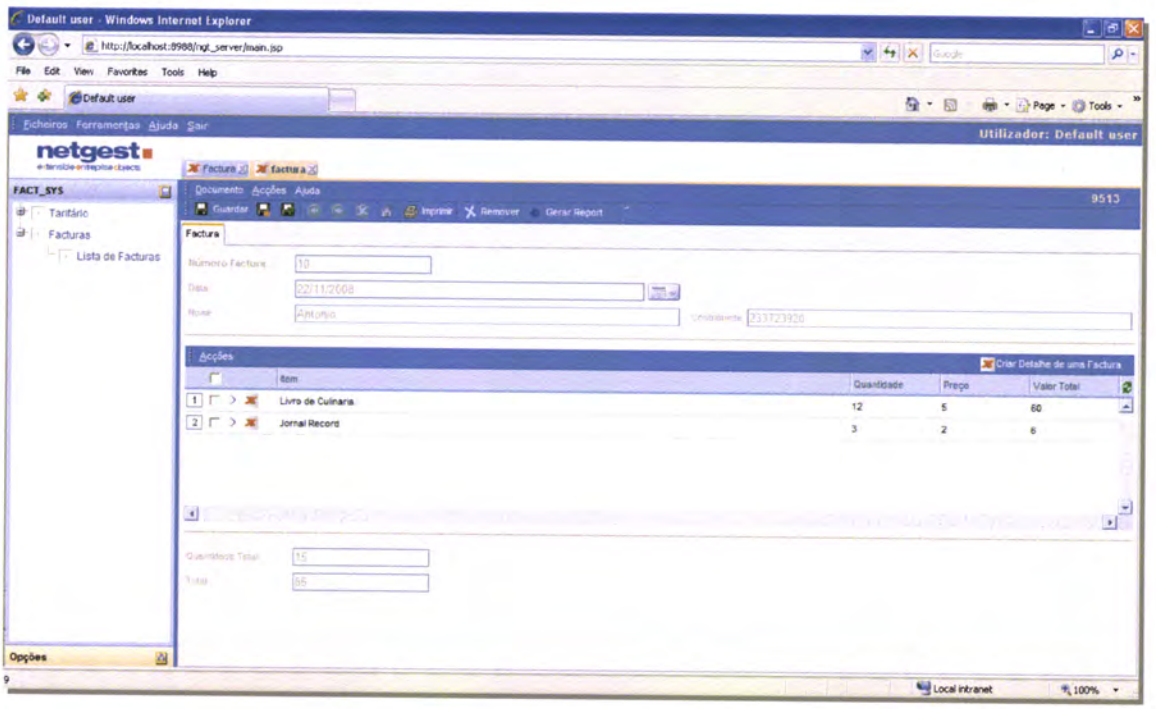


Figura 12: Exemplo de um ecrã do tipo Edit

A figura anterior demonstra um ecrã editável da plataforma xeo.

Este ecrã é composto por vários tipos de campos, que poderão ser editáveis ou não, entre os quais um ecrã do tipo list, com uma listagem de elementos.

Eles representam atributos de um determinado objecto, neste caso de um objecto factura.

Para um utilizador iniciar a criação de um relatório referente a este ecrã, basta carregar no botão situado na parte superior do ecrã, com o nome "Gerar Relatório".

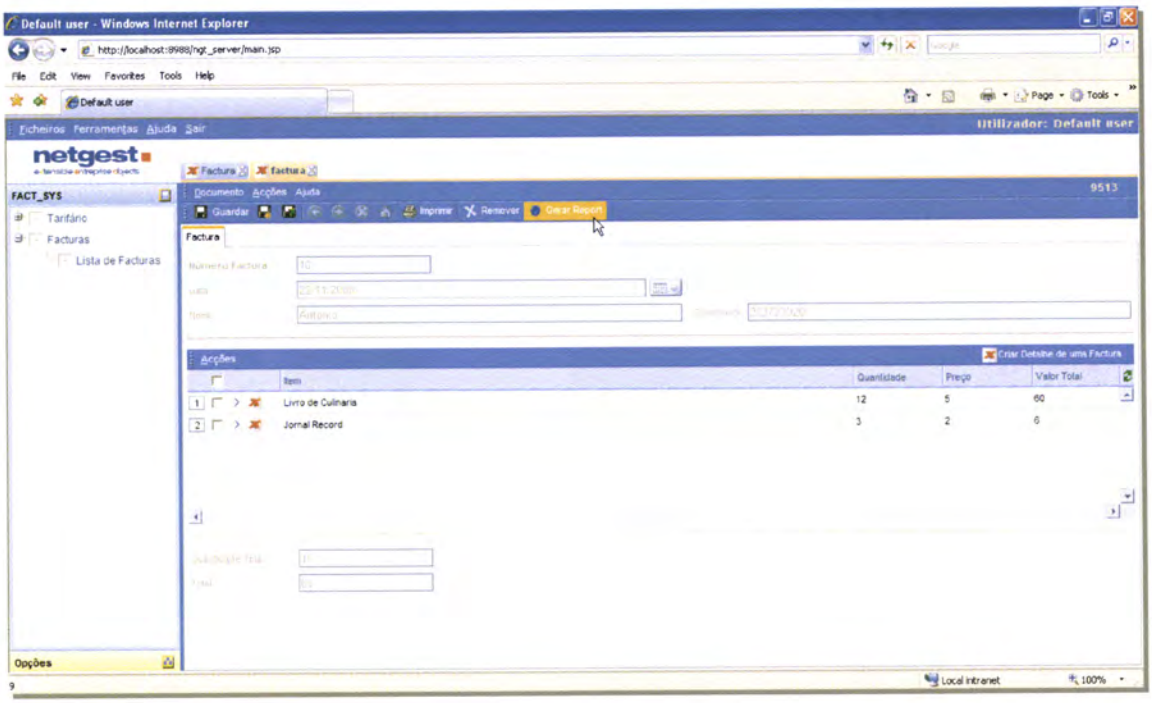


Figura 13: Botão de chamada da criação de um relatório de um ecrã Edit.

Depois de se iniciar a criação de um relatório, este será apresentado num novo ecrã.

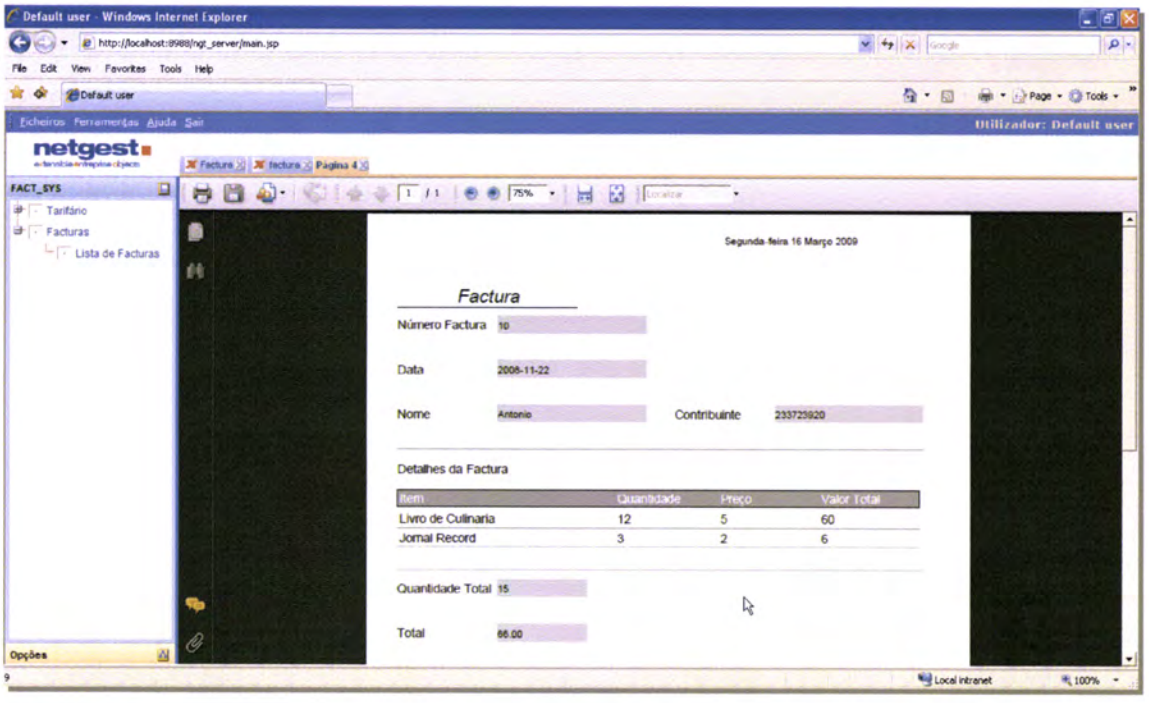


Figura 14: Relatório de um ecrã do tipo Edit

De seguida será apresentado um exemplo de criação de um relatório a partir de um ecrã do tipo Explorer.

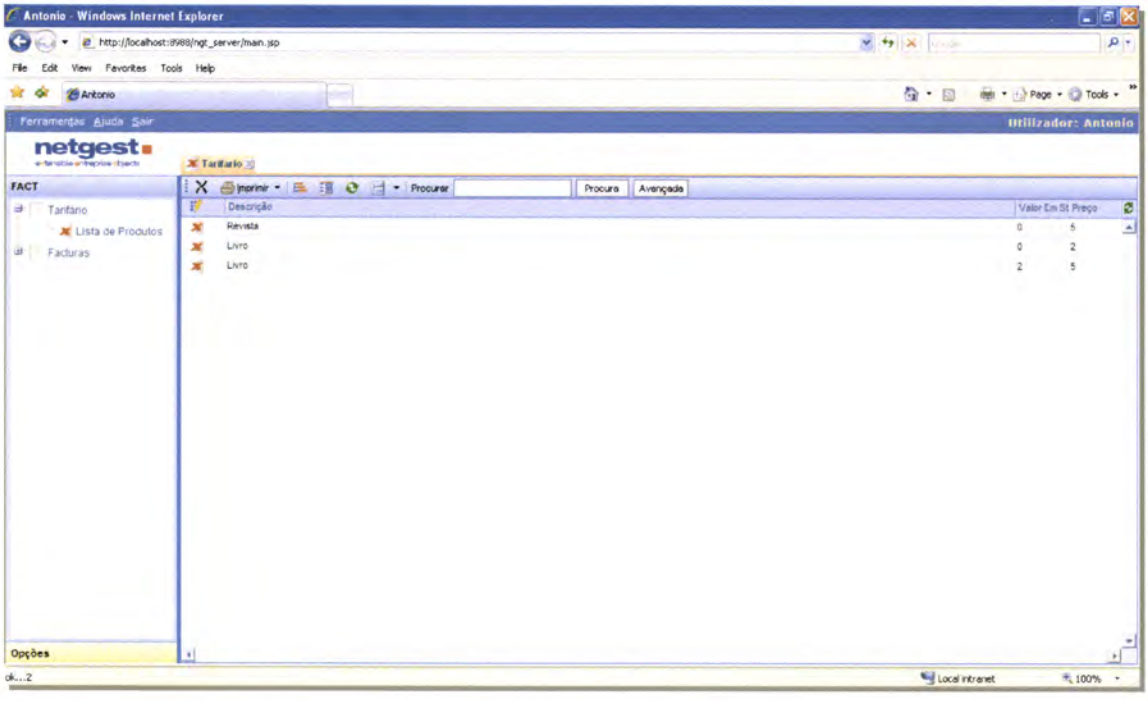


Figura 15: Exemplo de um ecrã do tipo Explorer

Para iniciar a criação do relatório, ao utilizador bastará carregar no botão que se situa na parte superior do ecrã com nome "imprimir".

Ao carregar neste botão, a plataforma XEO disponibiliza as diversas opções de impressão, entre as quais se encontra a criação de um relatório utilizando o módulo desenvolvido.

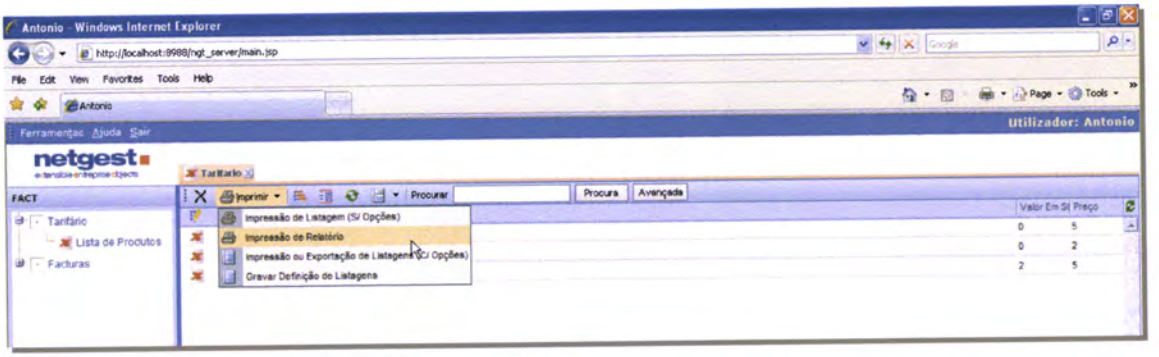


Figura 16: Botão de chamada da criação de um relatório de um ecrã Explorer

Ao carregar em "Imprimir relatório" o utilizador inicia a criação de um novo relatório relativo a este ecrã.

O relatório será aberto num novo ecrã.

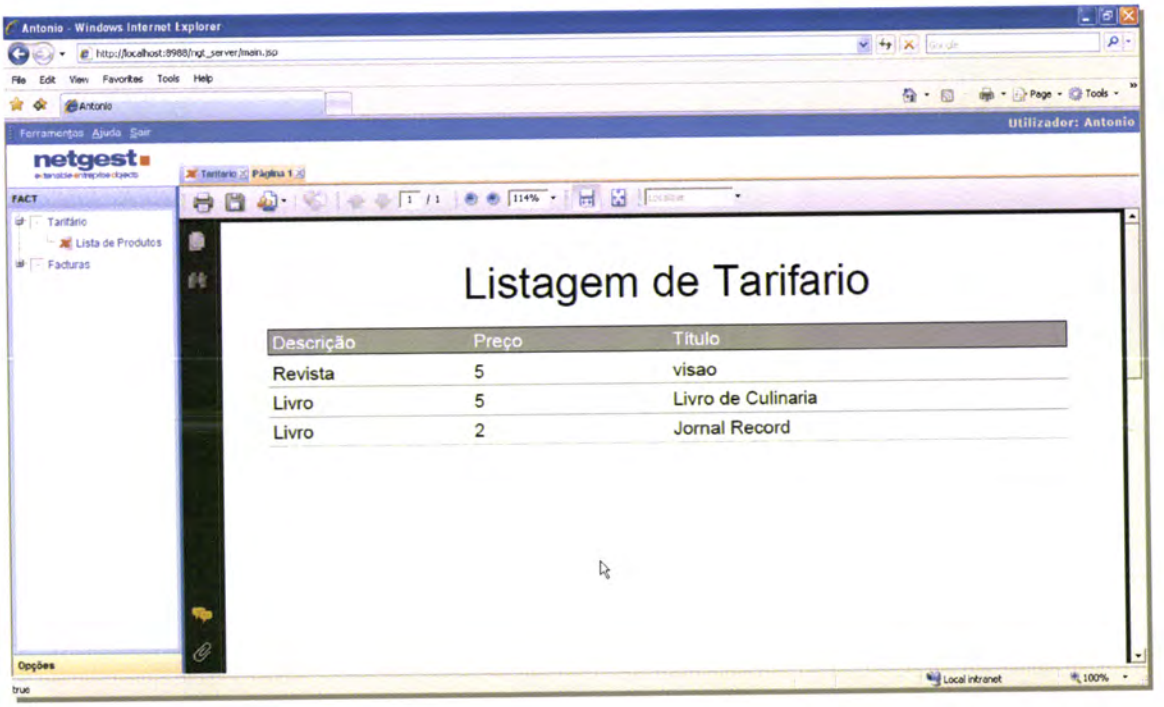


Figura 17: Relatório de um ecrã do tipo Explorer.

5 - CONCLUSÃO E TRABALHO FUTURO

Afirmando-se inicialmente como objectivo principal deste projecto, a reprodução de ecrãs da plataforma XEO em relatórios, no formato .pdf, utilizando para isso a plataforma JasperReports, obrigando para tal a haver uma integração entre as duas plataformas, podemos afirmar que esse objectivo foi conseguido.

Através do módulo XEOReports, passou a ser possível, na plataforma XEO, a criação de relatórios que reproduzam os seus ecrãs.

Outro objectivo, e tendo em conta que a plataforma dispõe de ecrãs que podem ser alterados pelo utilizador (ecrã do tipo Explorer em que o utilizador pode adicionar ou remover colunas de uma listagem, ou até agrupar informação que se repita), foi desenvolver a funcionalidade de detectar e criar relatórios que replicassem essas alterações de uma forma transparente para o utilizador.

No desenvolvimento deste projecto foi também alcançado o objectivo da criação de bancos de dados constituídos por objectos XEO, que possibilitou um acesso muito mais fácil a informação na forma como esta é tratada pela plataforma XEO.

Este ultimo ponto, permite também a utilização de bancos de dados constituídos por objectos XEO, neste caso em relatórios fixos. Entenda-se como relatórios fixos, relatórios específicos para um determinado caso, mas que sejam igualmente preenchidos com informação vinda de objectos XEO.

Em caso de desenvolvimento da plataforma XEO, este módulo teria de ser construído para que este sofresse o mínimo de alterações.

Estas alterações, a existirem, deveriam ser feitas de forma simples.

Este objectivo for alcançado, através da construção de uma estrutura intermédia em forma de árvore, entre a leitura da constituição dos ecrãs XEO e a criação do ficheiro XML para ser utilizado pela plataforma JasperReports. Através desta estrutura, este módulo torna-se fácil de ser alterado e adaptável a novas situações.

Assim sendo, podemos afirmar que os objectivos principais para o desenvolvimento do XEOReports foram alcançados.

Através deste projecto foi possível conhecer mais aprofundadamente, ferramentas de desenvolvimento de relatórios, como a JasperReports, a plataforma XEO e linguagens de programação e desenvolvimento como o Java, jsp, xml e a metodologia de desenvolvimento em UML.

Uma das questões em que este projecto pode ser melhorado, é a questão da performance aquando da compilação dos ficheiros xml pela plataforma JasperReports.

Como foi mencionado nesta dissertação, durante a criação de um relatório, existe uma fase em que o ficheiro de formato .xml que representa o relatório, é compilado para depois ser preenchido com informação vinda do banco de dados.

Caso a complexidade do ecrã XEO seja elevada, o tempo de compilação destes ficheiros aumenta também, podendo atingir tempos demasiado altos.

Este ponto poderá ser melhorado, detectando de se existiram alterações no ecrã XEO desde a ultima criação de um relatório para este ecrã.

Deste modo esta fase de compilação do ficheiro xml, apenas existirá quando se verifiquem alterações no ecrã XEO, diminuindo com isto o tempo de criação de um relatório.

Melhoramentos ao nível do grafismo dos relatórios, devem também ser feitos, de forma a dar aos relatórios um cariz mais adequado à realidade onde serão utilizados.

Uma vez que a plataforma XEO se encontra em constante desenvolvimento, de forma a acompanhar as necessidades que os utilizadores apresentam, o modulo XEOReports terá necessariamente de acompanhar esse desenvolvimento. Para tarefa poderá estar facilitada, uma vez que desde o inicio do desenvolvimento deste projecto, houve sempre a preocupação que este fosse facilmente alterado caso fosse necessário.

6 - REFERÊNCIAS BIBLIOGRÁFICAS

[URL1] http://en.wikipedia.org/wiki/Report (Visitado em 5 de Maio de 2008).

[URL2] http://sweet.ua.pt/~pf/Relatorios/relatorios.html (Visitado em 5 de Maio de 2008).

[URL3] http://pt.wikipedia.org/wiki/Hibernate (Visitado em 20 Maio de 2008).

[URL4] http://www.hibernate.org/5.html (Visitado em 25 Maio de 2008).

[URL5] http://agilior.pt/Default.aspx?tabid=107 (Visitado em 15 de Junho de 2008).

[URL6] http://en.wikipedia.org/wiki/OutSystems (Visitado em 15 de Junho de 2008).

[URL7] http://www.outsystems.com/agile/ (Visitado em 20 de Junho de 2008).

[URL8] http://pt.wikipedia.org/wiki/Java_(linguagem_de_programa%C3%A7%C3%A3o) (Visitado em 10 de Julho de 2008).

[URL9] http://java.sun.com/ (Visitado em 10 de Julho de 2008).

[URL10] http://java.sun.com/products/jsp/ (Visitado em 10 de Julho de 2008).

[URL11] http://en.wikipedia.org/wiki/JavaServer_Pages (Visitado em 12 de Julho de 2008).

[URL12] http://en.wikipedia.org/wiki/XML (Visitado em 20 de Julho de 2008).

[URL13] http://jasperforge.org/website/jasperreportswebsite/trunk/index.html?group_id=252 (Visitado em 4 de Setembro de 2008).

[URL14] http://www.javafree.org/wiki/JasperReports (Visitado em 5 de Abril de 2008).

[URL15] http://www.dsc.ufcg.edu.br/~jacques/cursos/daca/html/documentviews/relatorios.htm (Visitado em 4 Setembro de 2008).

[URL16] http://www.jaspersoft.com/JasperSoft_JasperReports.html (Visitado em 4 de Setembro de 2008).

[URL17] http://jasperforge.org/plugins/project/project_home.php?group_id=83 (Visitado em 15 de Setembro de 2008).

[URL18] http://www.eclipse.org/birt/phoenix/ (Visitado em 10 de Outubro de 2008).

[URL19] http://www.eclipse.org/birt/phoenix/intro/ (Visitado em 11 de Outubro de 2008).

[URL20] http://en.wikipedia.org/wiki/BIRT_Project (Visitado em 11 de Outubro de 2008).

[URL21] http://pt.wikipedia.org/wiki/UML (Visitado em 20 de Novembro de

2008).

[URL22] http://imasters.uol.com.br/artigo/2753/uml/modelando_sistemas_em_uml_-_casos_de_uso/ (Visitado em 15 de Janeiro de 2009)

[URL23] http://atlas.kennesaw.edu/~dbraun/csis4650/A&D/index.htm (Visitado em 15 de Janeiro de 2009).

[URL24] http://pt.wikipedia.org/wiki/RUP (Visitado em 20 de Janeiro de 2009).

[URL25] http://www.featuredrivendevelopment.com/ (Visitado em 5 de Fevereiro de 2009).

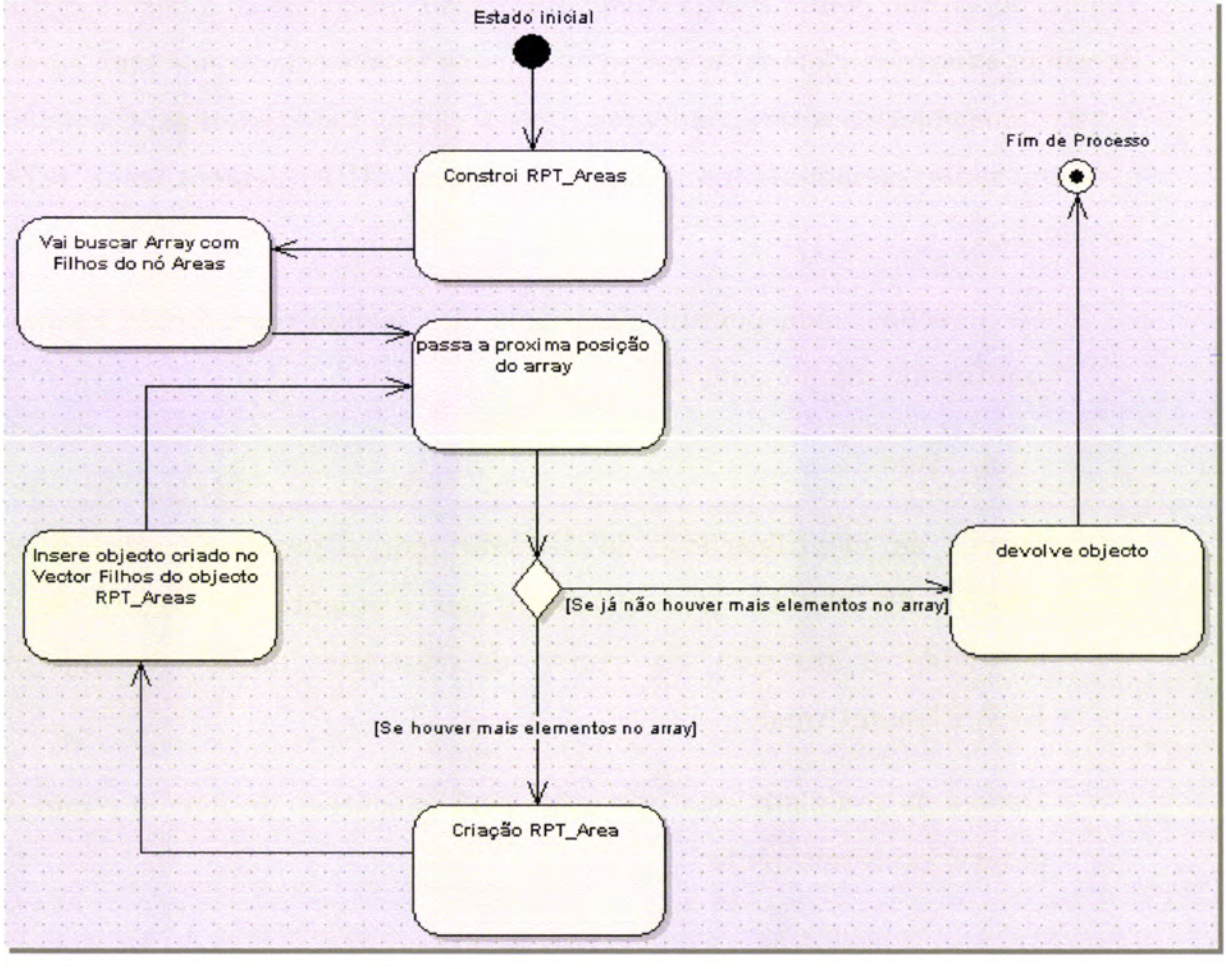
[URL26] http://pt.wikipedia.org/wiki/Rapid_Application_Development (Visitado em 1 de Março de 2009).

7 - ANEXOS

Anexo A

Em seguida é explicado como são construídos os diversos nós da estrutura intermédia em árvore.

- **Constrói RPT_ElementAreas**



Este caso de uso será descrito através de um diagrama de actividade organizado tendo em conta o factor temporal, com os seguintes pontos:

1. Evocação do método de criação de um novo objecto RPT_ElementAreas
2. Evoca o Vector com os nós XML que constituem o componente Areas presente o ecrã de edição EDIT.
3. Passa para o próximo elemento deste Vector.
4. Verifica se ainda existem ou não elementos no Vector a serem tratados.

Depois do último ponto, é feita uma "bifurcação" no processo a ser descrito. Se ainda existirem elementos a ser tratados no vector com os nós xml, serão construídos objectos correspondentes a esses elementos existentes. Caso não existam mais elementos no Vector, o objecto RPT_ElementAreas será devolvido.

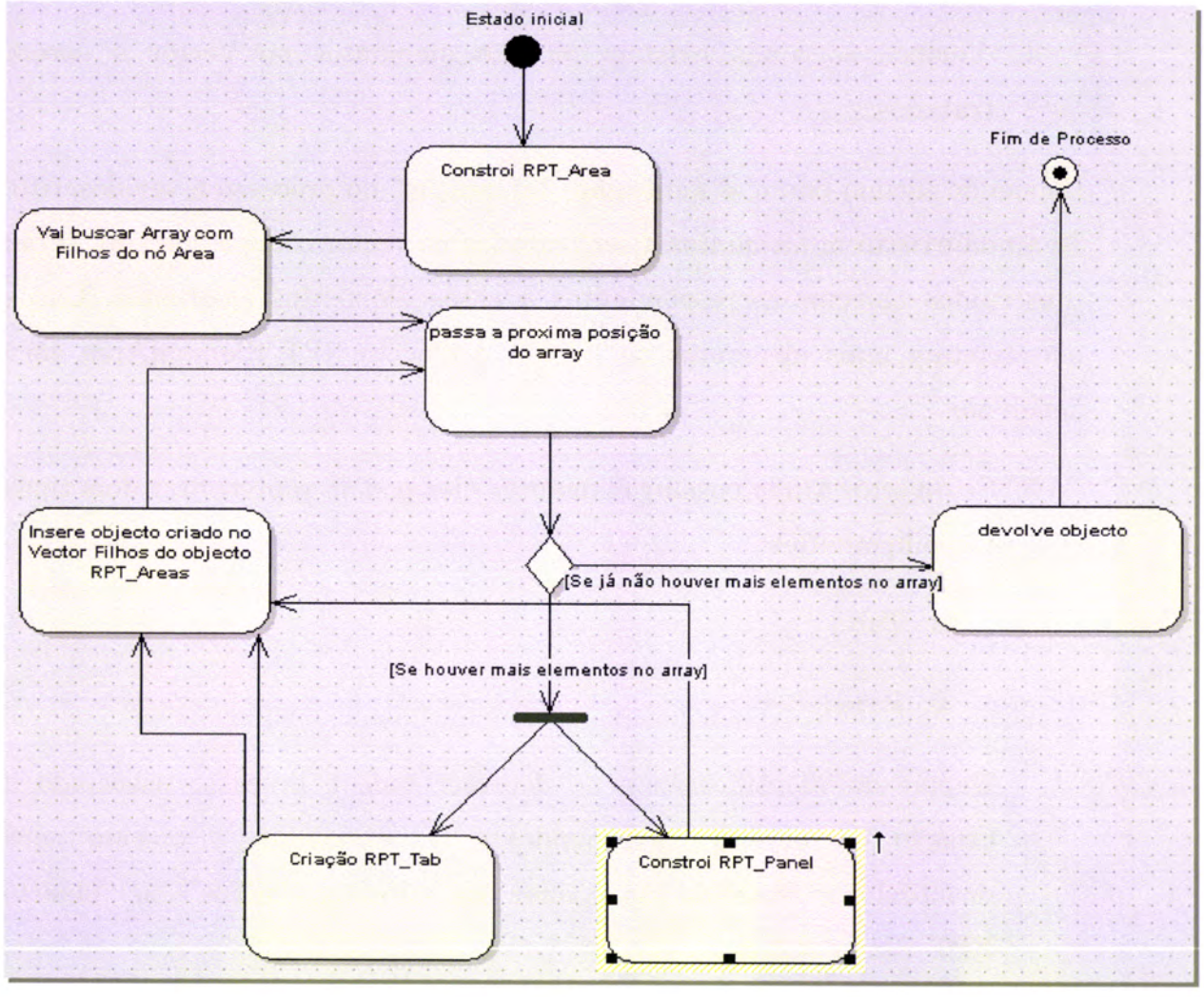
5. Se o vector ainda possuir elementos eles podem representar apenas um tipo de componente:

    a. Área

    Depois da identificação e de decorrer todo o processo associado a criação do objecto correspondente ao elemento, o objecto será devolvido e inserido no vector de objectos "filhos" do objecto RPT_ElementAreas.

    Depois de completa esta operação, será executado de novo o ponto 3 descrito anteriormente.

6. Se já não existirem mais elementos no vector evocado, o objecto RPT_ElementAreas e será devolvido e o processo de criação deste objecto encontra-se finalizado.



- Constrói RPT_ElementArea

Este caso de uso será descrito através de um diagrama de actividade organizado tendo em conta o factor temporal, com os seguintes pontos:

1. Evocação do método de criação de um novo objecto RPT_ElementArea

2. Evoca o Vector com os nós XML que constituem o componente Área presente no ecrã de edição EDIT.

3. Passa para o próximo elemento deste Vector.

4. Verifica se ainda existem ou não elementos no Vector a serem tratados.

Depois do último ponto, é feita uma "bifurcação" no processo a ser descrito. Se ainda existirem elementos a ser tratados no vector com os nós xml, serão construídos objectos correspondentes a esses elementos existentes. Casos não existam mais elementos no Vector, o objecto RPT_ElementArea será devolvido.

5. Se o vector ainda possuir elementos eles podem representar dois tipos de componentes:

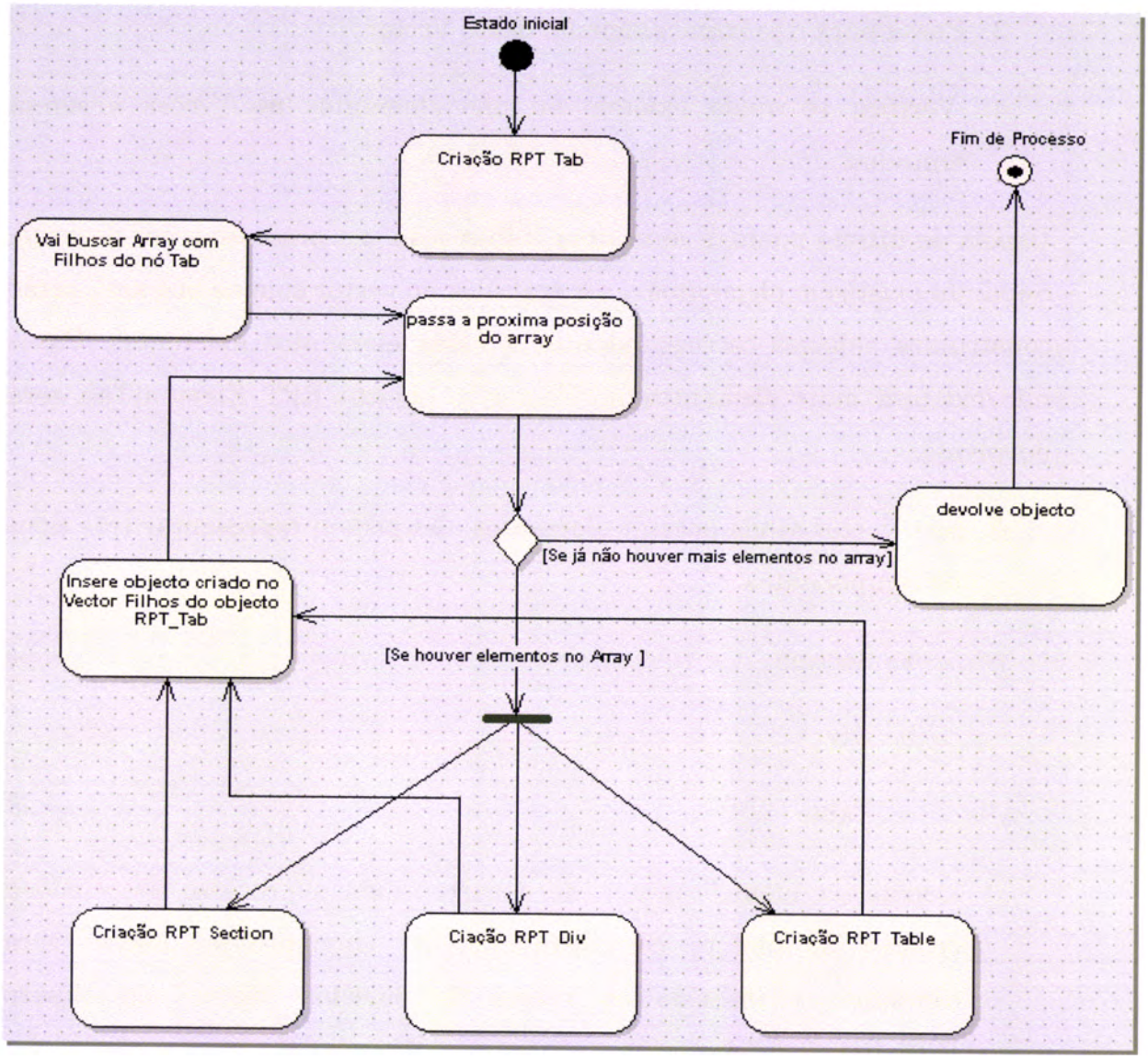
    a. Tab

    b. Panel

    Depois da identificação e de decorrer todo o processo associado a criação do objecto correspondente ao elemento, o objecto será devolvido e inserido no vector de objectos "filhos" do objecto RPT_ElementArea.

    Depois de completa esta operação, será executado de novo o ponto 3 descrito anteriormente.

6. Se já não existirem mais elementos no vector evocado, o objecto RPT_ElementArea e será devolvido e o processo de criação deste objecto encontra-se finalizado.



- **Criação RPT_ElementTab**

Este caso de uso será descrito através de um diagrama de actividade organizado tendo em conta o factor temporal, com os seguintes pontos:

1. Evocação do método de criação de um novo objecto RPT_ElementTab
2. Evoca o Vector com os nós XML que constituem o componente Tab presente no ecrã de edição EDIT.

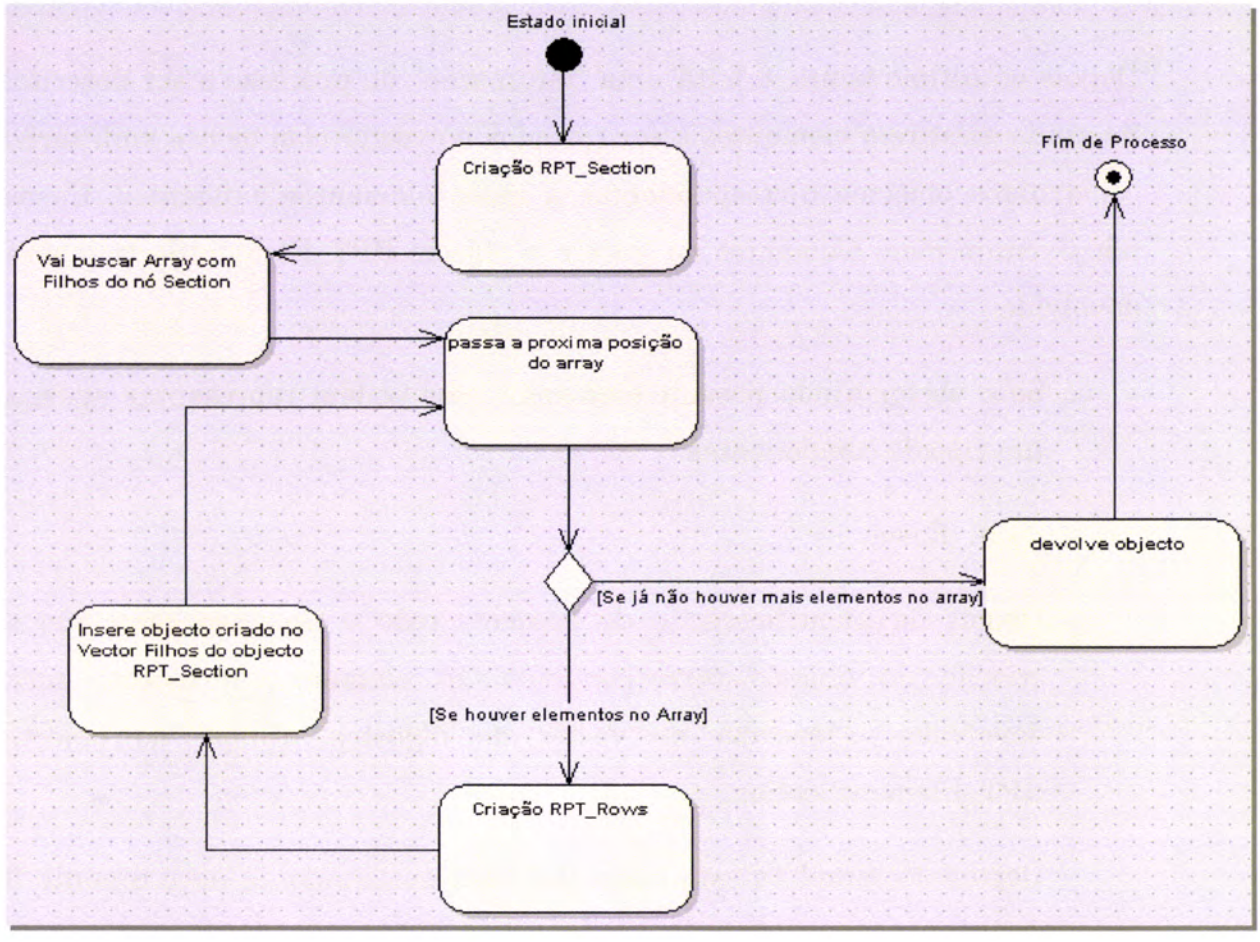
3. Passa para o próximo elemento deste Vector.
4. Verifica se ainda existem ou não elementos no Vector a serem tratados.

Depois do último ponto, é feita uma "bifurcação" no processo a ser descrito. Se ainda existirem elementos a ser tratados no vector com os nós xml, serão construídos objectos correspondentes a esses elementos existentes. Casos não existam mais elementos no Vector, o objecto RPT_ElementTab será devolvido.

5. Se o vector ainda possuir elementos eles podem representar três tipos de componentes:
    a. Section
    b. Div
    c. Table

    Depois da identificação e de decorrer todo o processo associado a criação do objecto correspondente ao elemento, o objecto será devolvido e inserido no vector de objectos "filhos" do objecto RPT_ElementTab.

    Depois de completa esta operação, será executado de novo o ponto 3 descrito anteriormente.
6. Se já não existirem mais elementos no vector evocado, o objecto RPT_ElementTab e será devolvido e o processo de criação deste objecto encontra-se finalizado.



- **Criação RPT_ElementSection**

Este caso de uso será descrito através de um diagrama de actividade organizado tendo em conta o factor temporal, com os seguintes pontos:

1. Evocação do método de criação de um novo objecto RPT_ElementSection
2. Evoca o Vector com os nós XML que constituem o componente Section presente no ecrã de edição EDIT.
3. Passa para o próximo elemento deste Vector.

4. Verifica se ainda existem ou não elementos no Vector a serem tratados.

Depois do último ponto, é feita uma "bifurcação" no processo a ser descrito. Se ainda existirem elementos a ser tratados no vector com os nós xml, serão construídos objectos correspondentes a esses elementos existentes. Casos não existam mais elementos no Vector, o objecto RPT_ElementSection será devolvido.

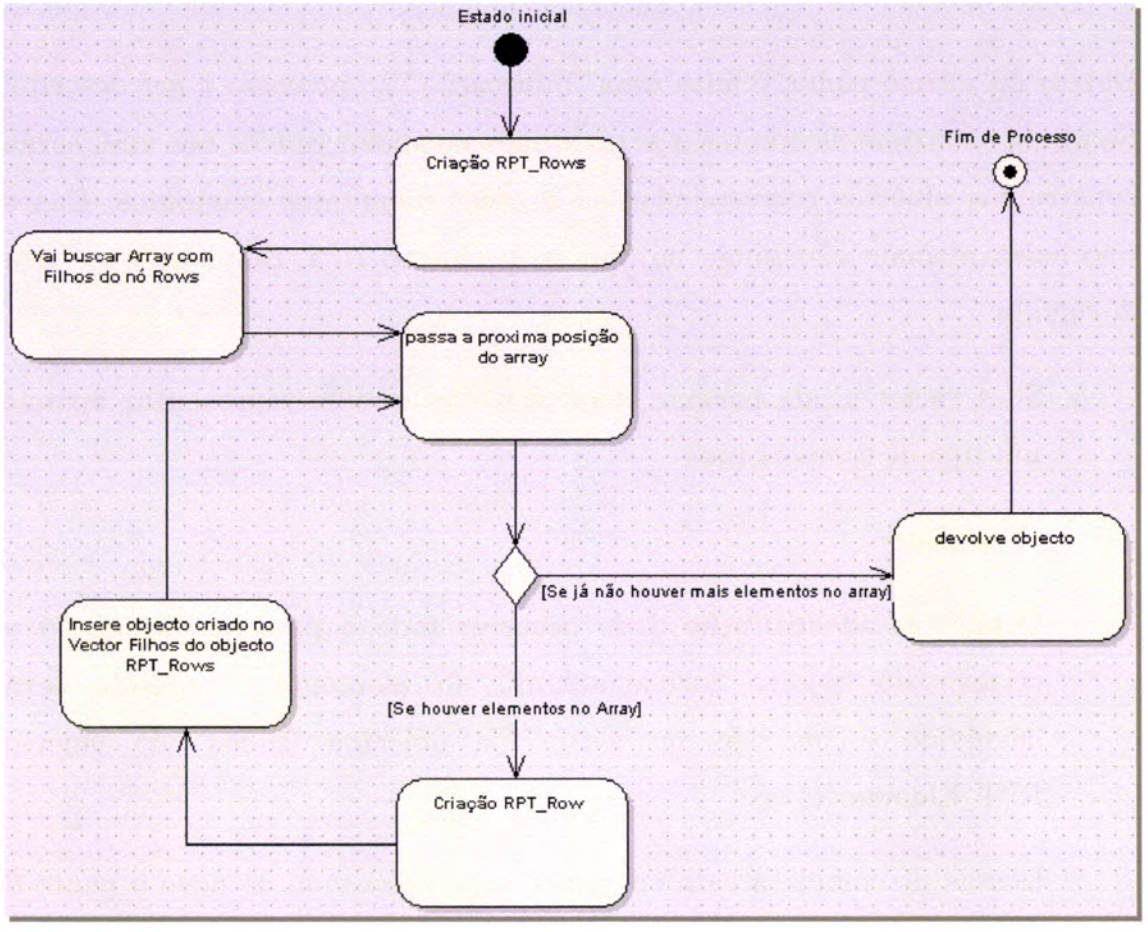
5. Se o vector ainda possuir elementos eles podem representar apenas um tipo de componentes:

    a. Rows

    Depois da identificação e de decorrer todo o processo associado a criação do objecto correspondente ao elemento, o objecto será devolvido e inserido no vector de objectos "filhos" do objecto RPT_ElementSection.

    Depois de completa esta operação, será executado de novo o ponto 3 descrito anteriormente.

6. Se já não existirem mais elementos no vector evocado, o objecto RPT_ElementSecion e será devolvido e o processo de criação deste objecto encontra-se finalizado.



- **Criação RPT_ElementRows**

Este caso de uso será descrito através de um diagrama de actividade organizado tendo em conta o factor temporal, com os seguintes pontos:

1. Evocação do método de criação de um novo objecto RPT_ElementRows
2. Evoca o Vector com os nós XML que constituem o componente Rows presente no ecrã de edição EDIT.
3. Passa para o próximo elemento deste Vector.

4. Verifica se ainda existem ou não elementos no Vector a serem tratados.

Depois do último ponto, é feita uma "bifurcação" no processo a ser descrito. Se ainda existirem elementos a ser tratados no vector com os nós xml, serão construídos objectos correspondentes a esses elementos existentes. Casos não existam mais elementos no Vector, o objecto RPT_ElementRows será devolvido.

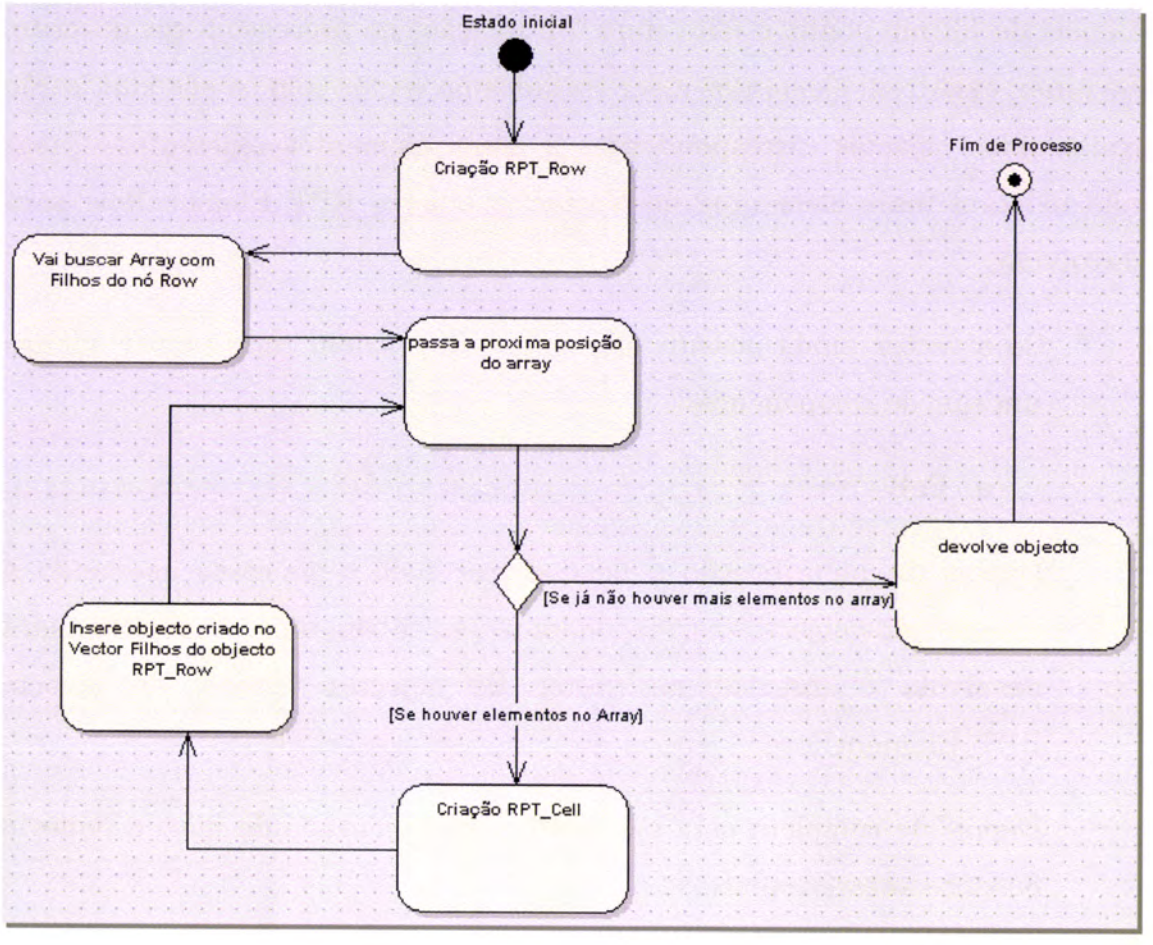
5. Se o vector ainda possuir elementos eles podem representar apenas um tipo de componentes:

    a. Row

    Depois da identificação e de decorrer todo o processo associado a criação do objecto correspondente ao elemento, o objecto será devolvido e inserido no vector de objectos "filhos" do objecto RPT_ElementRows.

    Depois de completa esta operação, será executado de novo o ponto 3 descrito anteriormente.

6. Se já não existirem mais elementos no vector evocado, o objecto RPT_ElementRows e será devolvido e o processo de criação deste objecto encontra-se finalizado.



- **Criação RPT_ElementRow**

Este caso de uso será descrito através de um diagrama de actividade organizado tendo em conta o factor temporal, com os seguintes pontos:

1. Evocação do método de criação de um novo objecto RPT_ElementRow
2. Evoca o Vector com os nós XML que constituem o componente Row presente no ecrã de edição EDIT.
3. Passa para o próximo elemento deste Vector.
4. Verifica se ainda existem ou não elementos no Vector a serem tratados.

Depois do último ponto, é feita uma "bifurcação" no processo a ser descrito. Se ainda existirem elementos a ser tratados no vector com os nós xml, serão construídos objectos correspondentes a esses elementos existentes. Casos não existam mais elementos no Vector, o objecto RPT_ElementRow será devolvido.

5. Se o vector ainda possuir elementos eles podem representar apenas um tipo de componentes:

    a. Cell

    Depois da identificação e de decorrer todo o processo associado a criação do objecto correspondente ao elemento, o objecto será devolvido e inserido no vector de objectos "filhos" do objecto RPT_ElementRow.

    Depois de completa esta operação, será executado de novo o ponto 3 descrito anteriormente.

6. Se já não existirem mais elementos no vector evocado, o objecto RPT_ElementRow e será devolvido e o processo de criação deste objecto encontra-se finalizado.

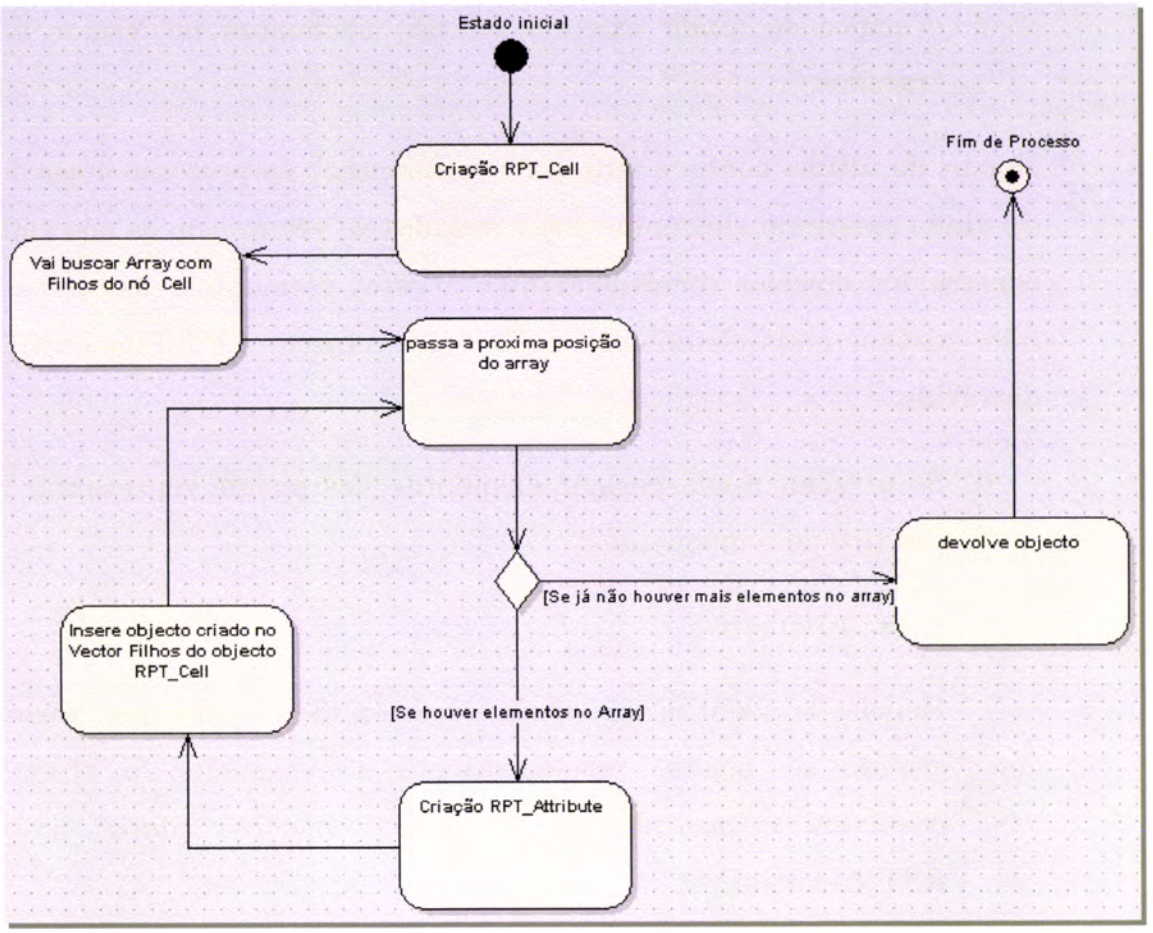


Figura 16:Activity Diagram– Criação RPT_ElementCell

- **Criação RPT_ElementCell**

Este caso de uso será descrito através de um diagrama de actividade organizado tendo em conta o factor temporal, com os seguintes pontos:

1. Evocação do método de criação de um novo objecto RPT_ElementCell
2. Evoca o Vector com os nós XML que constituem o componente Cell presente no ecrã de edição EDIT.
3. Passa para o próximo elemento deste Vector.

4. Verifica se ainda existem ou não elementos no Vector a serem tratados.

Depois do último ponto, é feita uma "bifurcação" no processo a ser descrito. Se ainda existirem elementos a ser tratados no vector com os nós xml, serão construídos objectos correspondentes a esses elementos existentes. Casos não existam mais elementos no Vector, o objecto RPT_ElementCell será devolvido.

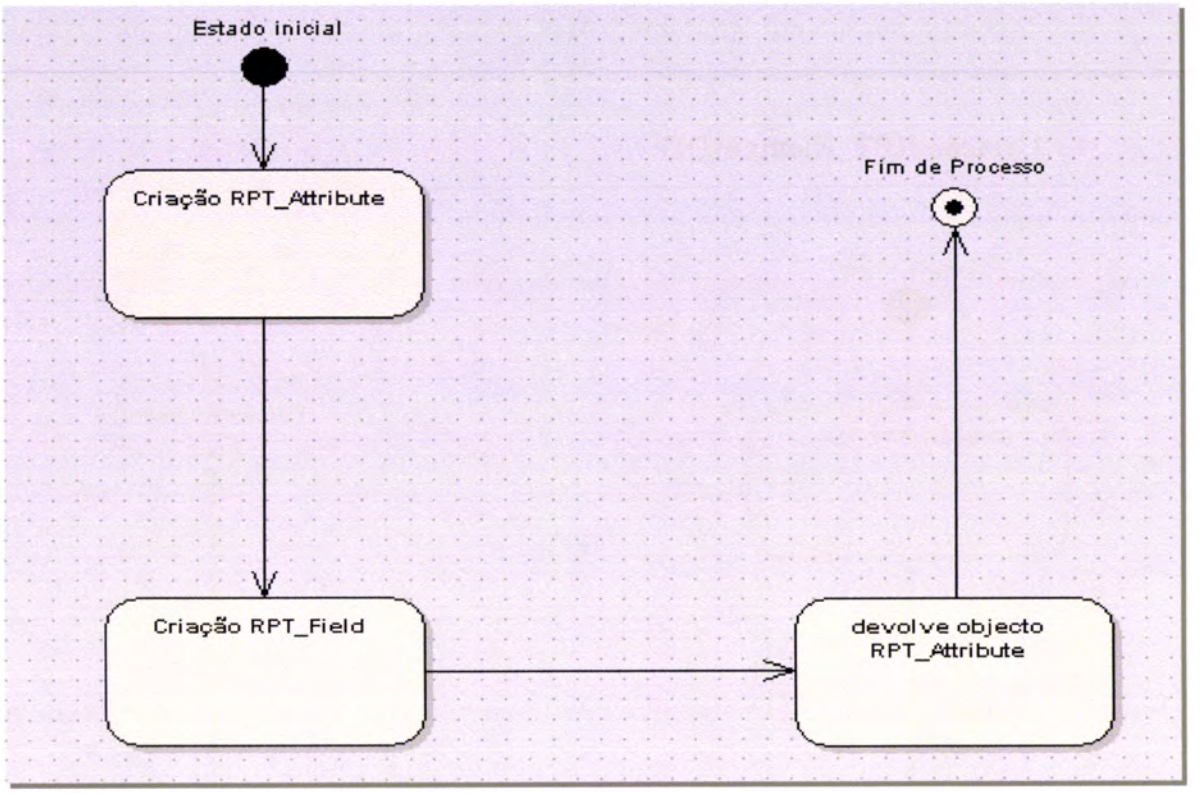
5. Se o vector ainda possuir elementos eles podem representar apenas um tipo de componentes:

    a. Attribute

    Depois da identificação e de decorrer todo o processo associado a criação do objecto correspondente ao elemento, o objecto será devolvido e inserido no vector de objectos "filhos" do objecto RPT_ElementCell.

    Depois de completa esta operação, será executado de novo o ponto 3 descrito anteriormente.

6. Se já não existirem mais elementos no vector evocado, o objecto RPT_ElementCell e será devolvido e o processo de criação deste objecto encontra-se finalizado.

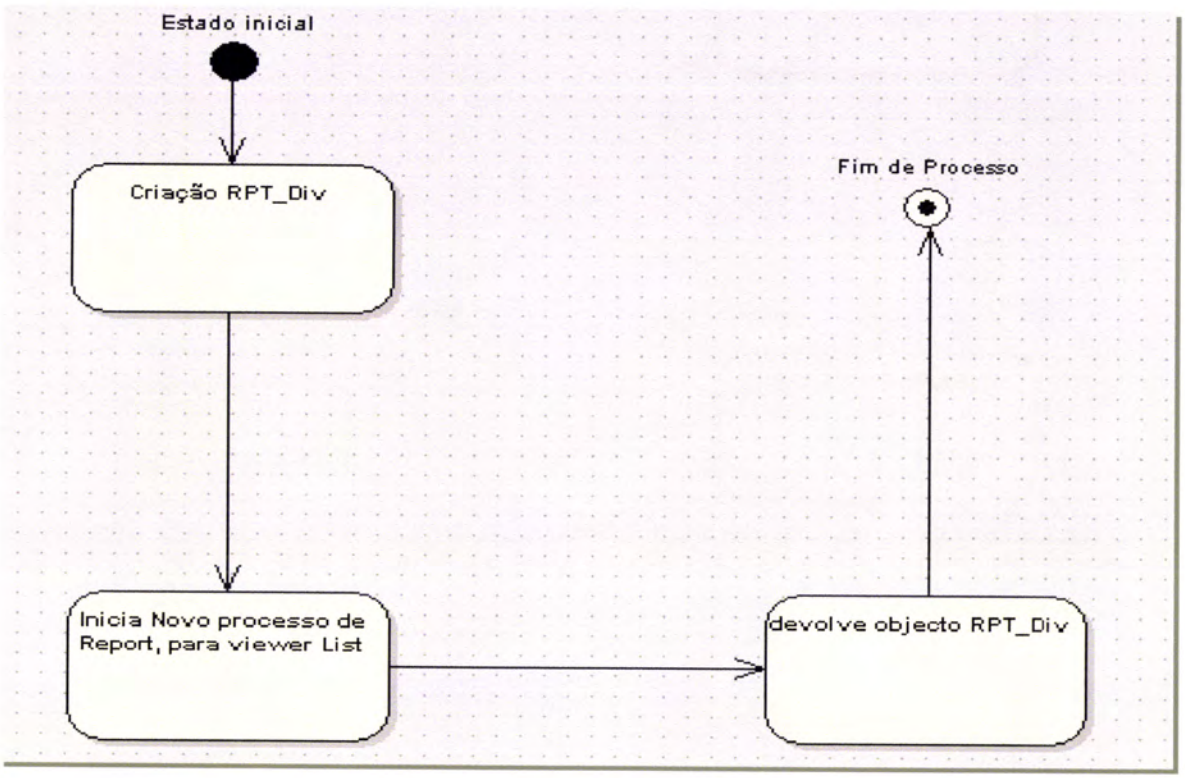


- **Criação RPT_ElementAttribute**

Este caso de uso será descrito através de um diagrama de actividade organizado tendo em conta o factor temporal, com os seguintes pontos:

1. Evocação do método de criação de um novo objecto RPT_ElementAttribute
2. Evocação do método de criação de um novo objecto RPT_ElementField
3. O objecto RPT_ElementAttribute e será devolvido e o processo de criação deste objecto encontra-se finalizado.

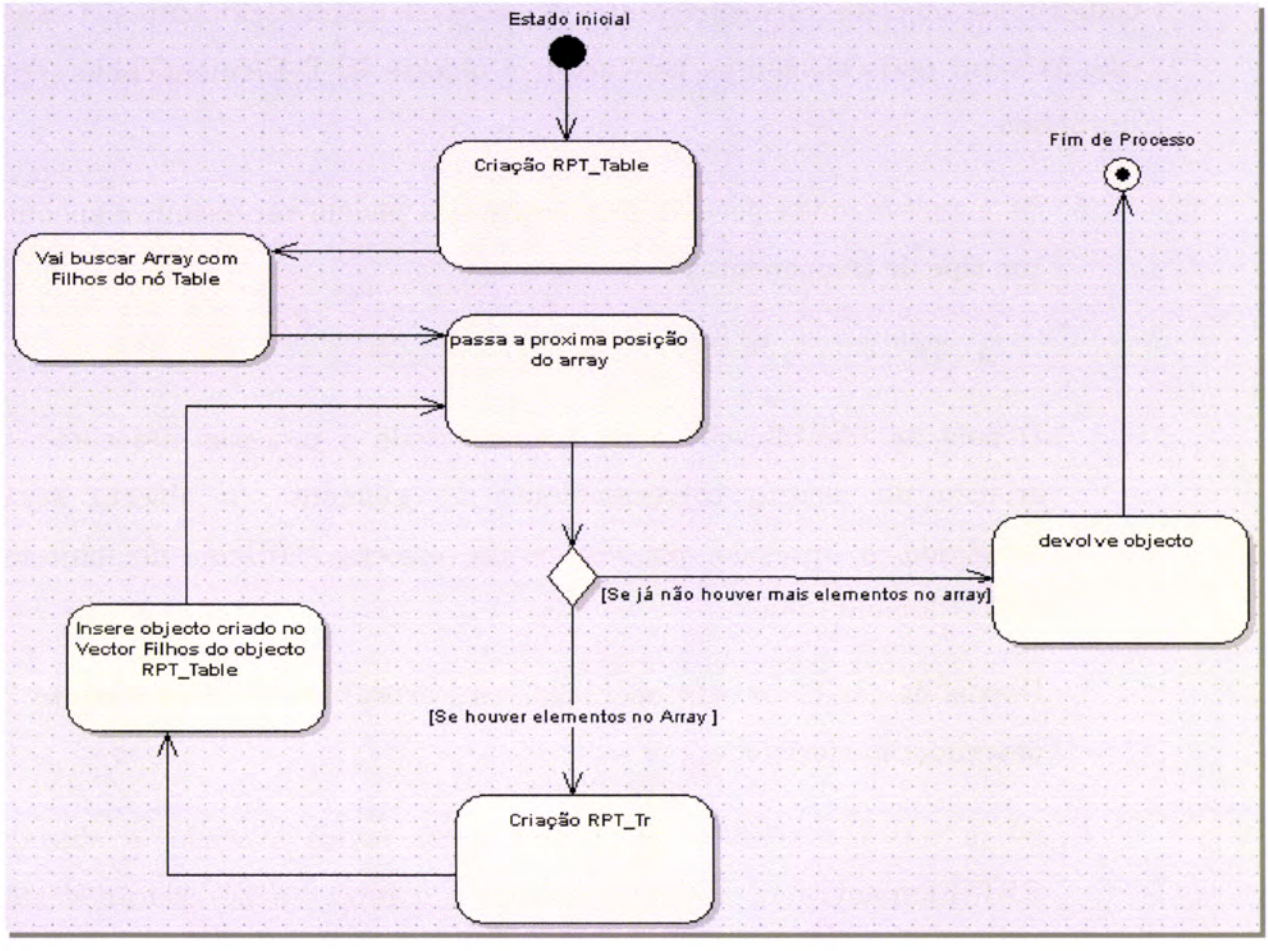
- **Criação RPT_ElementDIV**



Este caso de uso será descrito através de um diagrama de actividade organizado tendo em conta o factor temporal, com os seguintes pontos:

1. Evocação do método de criação de um novo objecto RPT_ElementDiv
2. Evocação do método responsável pela criação de um sub-relatório do tipo LIST
3. Depois de concluído o processo de construção de um sub-relatorio do tipo List o objecto RPT_ElementDIV é devolvido e o processo de criação do mesmo encontra-se terminado.

- **Criação RPT_ElementTable**



Este caso de uso será descrito através de um diagrama de actividade organizado tendo em conta o factor temporal, com os seguintes pontos:

1. Evocação do método de criação de um novo objecto RPT_ElementTable
2. Evoca o Vector com os nós XML que constituem o componente Table presente no ecrã de edição EDIT.
3. Passa para o próximo elemento deste Vector.
4. Verifica se ainda existem ou não elementos no Vector a serem tratados.

Depois do último ponto, é feita uma "bifurcação" no processo a ser descrito. Se ainda existirem elementos a ser tratados no vector com os nós xml, serão construídos objectos correspondentes a esses elementos existentes. Casos não existam mais elementos no Vector, o objecto RPT_ElementTable será devolvido.

5. Se o vector ainda possuir elementos eles podem representar apenas um tipo de componentes:

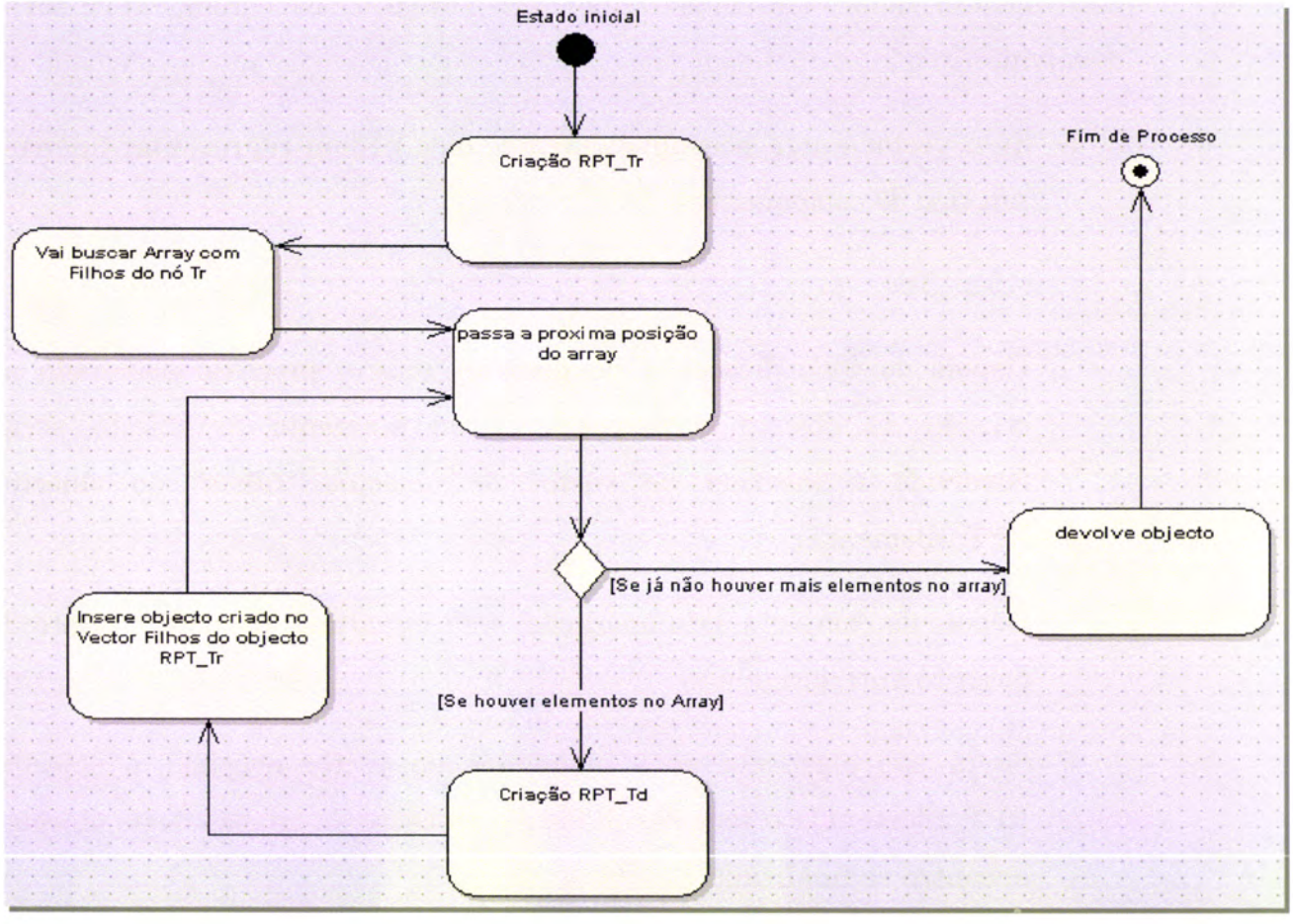
    a. Tr

    Depois da identificação e de decorrer todo o processo associado a criação do objecto correspondente ao elemento, o objecto será devolvido e inserido no vector de objectos "filhos" do objecto RPT_ElementTable.

    Depois de completa esta operação, será executado de novo o ponto 3 descrito anteriormente.

6. Se já não existirem mais elementos no vector evocado, o objecto RPT_ElementTable e será devolvido e o processo de criação deste objecto encontra-se finalizado.

- **Criação RPT_ElementTr**



Este caso de uso será descrito através de um diagrama de actividade organizado tendo em conta o factor temporal, com os seguintes pontos:

1. Evocação do método de criação de um novo objecto RPT_ElementTr
2. Evoca o Vector com os nós XML que constituem o componente Tr presente no ecrã de edição EDIT.
3. Passa para o próximo elemento deste Vector.

4. Verifica se ainda existem ou não elementos no Vector a serem tratados.

Depois do último ponto, é feita uma "bifurcação" no processo a ser descrito. Se ainda existirem elementos a ser tratados no vector com os nós xml, serão construídos objectos correspondentes a esses elementos existentes. Casos não existam mais elementos no Vector, o objecto RPT_ElementTr será devolvido.

5. Se o vector ainda possuir elementos eles podem representar apenas um tipo de componentes:

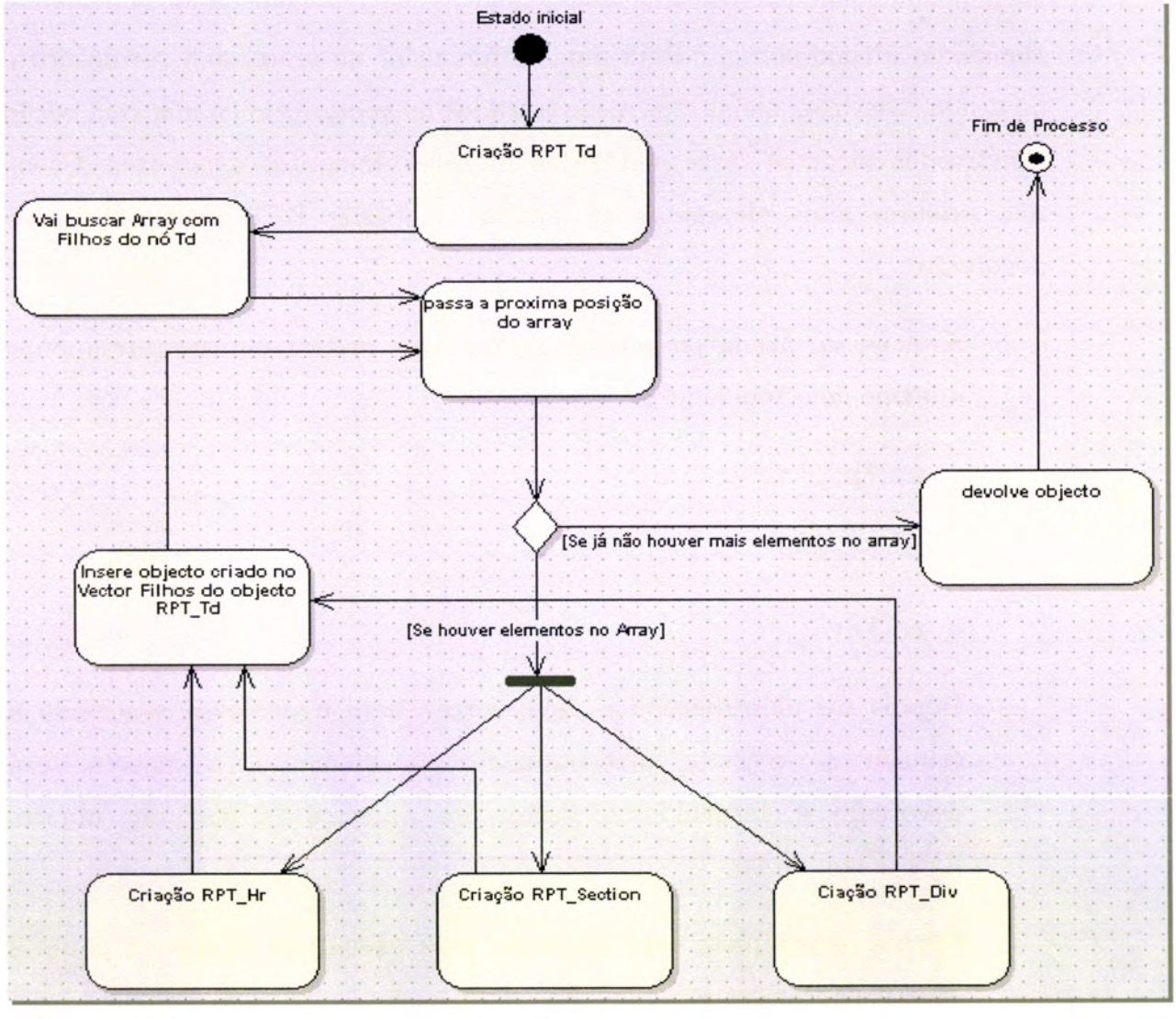
    a. Td

    Depois da identificação e de decorrer todo o processo associado a criação do objecto correspondente ao elemento, o objecto será devolvido e inserido no vector de objectos "filhos" do objecto RPT_ElementTr.

    Depois de completa esta operação, será executado de novo o ponto 3 descrito anteriormente.

6. Se já não existirem mais elementos no vector evocado, o objecto RPT_ElementTr e será devolvido e o processo de criação deste objecto encontra-se finalizado.

- **Criação RPT_ElementTd**



Este caso de uso será descrito através de um diagrama de actividade organizado tendo em conta o factor temporal, com os seguintes pontos:

1. Evocação do método de criação de um novo objecto RPT_ElementTd

2. Evoca o Vector com os nós XML que constituem o componente Td presente no ecrã de edição EDIT.

3. Passa para o próximo elemento deste Vector.

4. Verifica se ainda existem ou não elementos no Vector a serem tratados.

Depois do último ponto, é feita uma "bifurcação" no processo a ser descrito. Se ainda existirem elementos a ser tratados no vector com os nós xml, serão construídos objectos correspondentes a esses elementos existentes. Casos não existam mais elementos no Vector, o objecto RPT_ElementTd será devolvido.

5. Se o vector ainda possuir elementos eles podem representar apenas um tipo de componentes:

    a. Hr

    b. Section

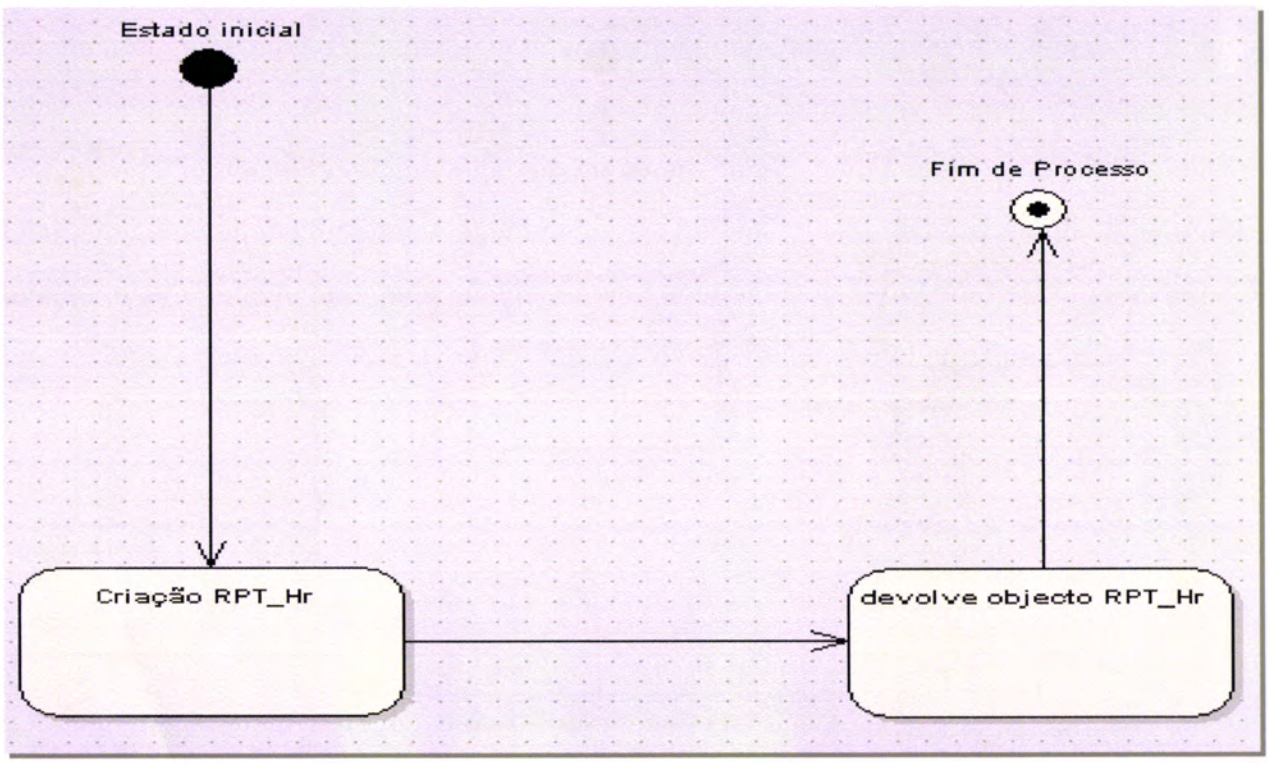
    c. Div

    Depois da identificação e de decorrer todo o processo associado a criação do objecto correspondente ao elemento, o objecto será devolvido e inserido no vector de objectos "filhos" do objecto RPT_ElementTd.

    Depois de completa esta operação, será executado de novo o ponto 3 descrito anteriormente.

6. Se já não existirem mais elementos no vector evocado, o objecto RPT_ElementTd e será devolvido e o processo de criação deste objecto encontra-se finalizado.

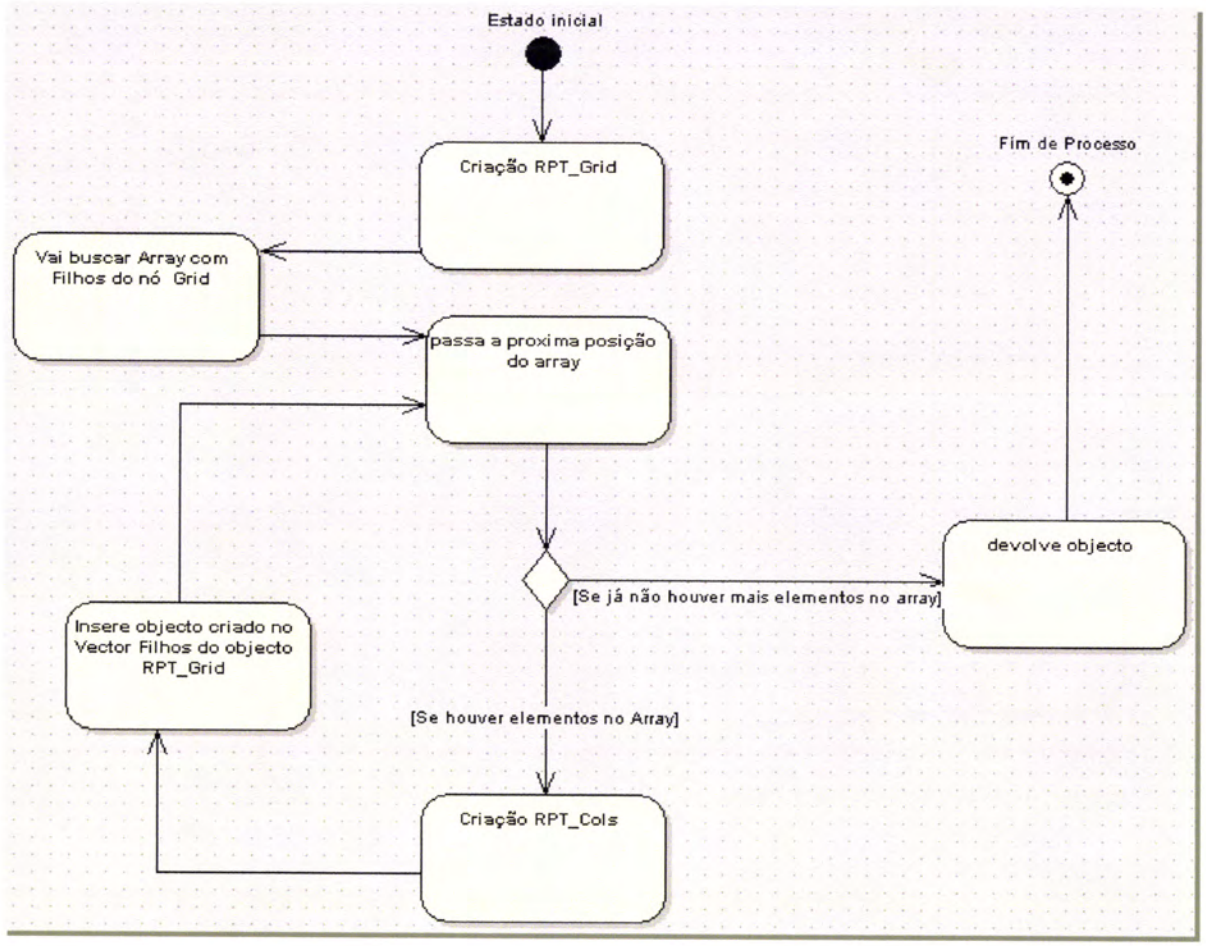
- **Criação RPT_ElementHr**



Este caso de uso será descrito através de um diagrama de actividade organizado tendo em conta o factor temporal, com os seguintes pontos:

1. Evocação do método de criação de um novo objecto RPT_ElementHr
2. O objecto RPT_ElementHr é devolvido e o processo de criação do mesmo fica concluído.

- **Construir nó de árvore RPT_ElementGrid**



Este caso de uso será descrito através de um diagrama de actividade organizado tendo em conta o factor temporal, com os seguintes pontos:

4. Evocação do método de criação de um novo objecto RPT_ElementGrid

5. Evoca o Vector com os nós XML que constituem o componente Grid presente no ecrã de edição LIST.

6. Passa para o próximo elemento deste Vector.

7. Verifica se ainda existem ou não elementos no Vector a serem tratados.

Depois do último ponto, é feita uma "bifurcação" no processo a ser descrito. Se ainda existirem elementos a ser tratados no vector com os nós xml, serão construídos objectos correspondentes a esses elementos existentes. Casos não existam mais elementos no Vector, o objecto RPT_ElementGrid será devolvido.

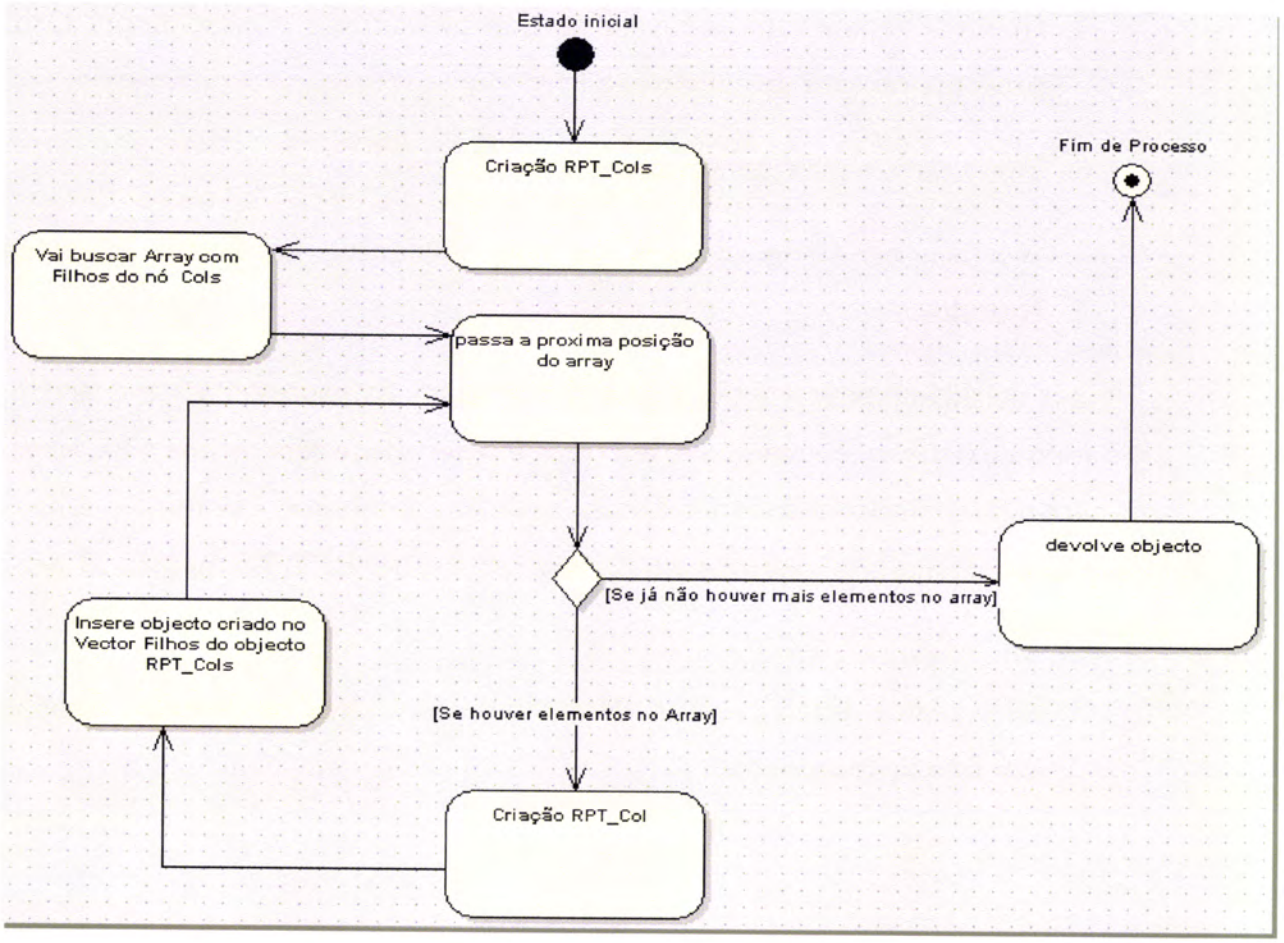
8. Se o vector ainda possuir elementos eles podem representar apenas um tipo de componentes:

    a. Cols

    Depois da identificação e de decorrer todo o processo associado à criação do objecto correspondente ao elemento, o objecto será devolvido e inserido no vector de objectos "filhos" do objecto RPT_ElementGrid.

    Depois de completa esta operação, será executado de novo o ponto 3 descrito anteriormente.

9. Se já não existirem mais elementos no vector evocado, o objecto RPT_ElementGrid e será devolvido e o processo de criação deste objecto encontra-se finalizado.



- **Criação RPT_ElementCols**

Este caso de uso será descrito através de um diagrama de actividade organizado tendo em conta o factor temporal, com os seguintes pontos:

1. Evocação do método de criação de um novo objecto RPT_ElementCols
2. Evoca o Vector com os nós XML que constituem o componente Cols presente no ecrã de edição LIST.
3. Passa para o próximo elemento deste Vector.
4. Verifica se ainda existem ou não elementos no Vector a serem tratados.

Depois do último ponto, é feita uma "bifurcação" no processo a ser descrito. Se ainda existirem elementos a ser tratados no vector com os nós xml, serão construídos objectos correspondentes a esses elementos existentes. Casos não existam mais elementos no Vector, o objecto RPT_ElementCols será devolvido.

5. Se o vector ainda possuir elementos eles podem representar apenas um tipo de componentes:

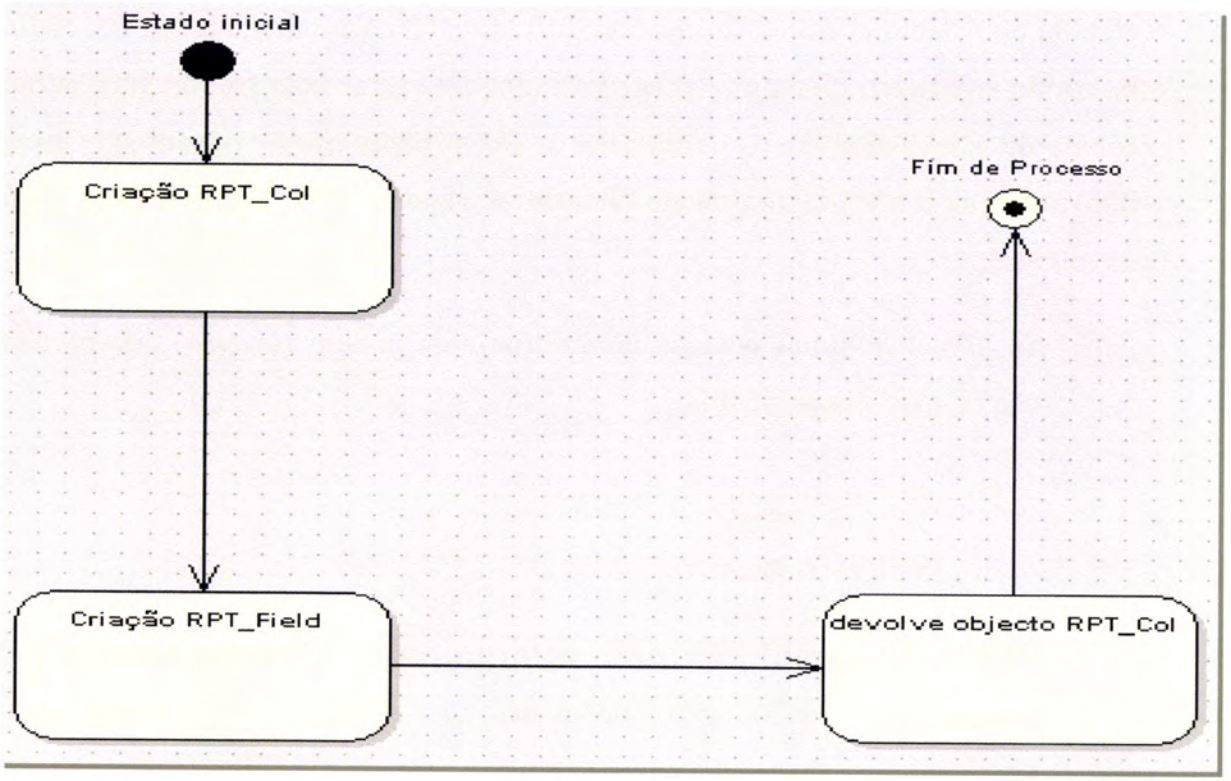
    a. Col

    b. ExplorerColumn

    Depois da identificação e de decorrer todo o processo associado à criação do objecto correspondente ao elemento, o objecto será devolvido e inserido no vector de objectos "filhos" do objecto RPT_ElementCols.

    Depois de completa esta operação, será executado de novo o ponto 3 descrito anteriormente.

6. Se já não existirem mais elementos no vector evocado, o objecto RPT_ElementCols e será devolvido e o processo de criação deste objecto encontra-se finalizado.



- **Construir nó de árvore RPT_ElementCol**

Este caso de uso será descrito através de um diagrama de actividade organizado tendo em conta o factor temporal, com os seguintes pontos:

1. Evocação do método de criação de um novo objecto RPT_ElementCol
2. Evocação do método de criação de um novo objecto RPT_Field
3. O objecto RPT_ElementCol e será devolvido e o processo de criação deste objecto encontra-se finalizado.

- **Construir nó de árvore RPT_ElementExplorerColumn**

Este caso de uso será descrito através de um diagrama de actividade organizado tendo em conta o factor temporal, com os seguintes pontos:

1. Evocação do método de criação de um novo objecto RPT_ElementExplorerColumn
2. Evocação do método de criação de um novo objecto RPT_Field
3. O objecto RPT_ElementExplorerColumn e será devolvido e o processo de criação deste objecto encontra-se finalizado.

- **Construir nó de árvore Explorer**

Este caso de uso será descrito através de um diagrama de actividade organizado tendo em conta o factor temporal, com os seguintes pontos:

1. Evocação do método de criação de um novo objecto RPT_ElementExplorer
2. Evoca o Vector com os nós XML que constituem o componente Cols presente no ecrã de edição Explorer.
3. Passa para o próximo elemento deste Vector.
4. Verifica se ainda existem ou não elementos no Vector a serem tratados.

Depois do último ponto, é feita uma "bifurcação" no processo a ser descrito. Se ainda existirem elementos a ser tratados no vector com os nós xml, serão construídos objectos correspondentes a esses elementos existentes. Casos não existam mais elementos no Vector, o objecto RPT_ElementExplorer será devolvido.

5. Se o vector ainda possuir elementos eles podem representar apenas um tipo de componentes:

    a. RPT_ElementExplorerGroup

    b. RPT_ElementCols

    Depois da identificação e de decorrer todo o processo associado à criação do objecto correspondente ao elemento, o objecto será devolvido e inserido no vector de objectos "filhos" do objecto RPT_ElementExplorer.

    Depois de completa esta operação, será executado de novo o ponto 3 descrito anteriormente.

Se já não existirem mais elementos no vector evocado, o objecto RPT_ElementExplorer e será devolvido e o processo de criação deste objecto encontra-se finalizado.

Anexo B

Em seguida é explicado como é criado o código em XML.

- **Cria Xml de um elemento Cols**

Ao encontrar um objecto RPT_ElementCols na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

- **Cria Xml de um elemento Table**

Ao encontrar um objecto RPT_ElementTable na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

- **Cria Xml de um elemento Areas**

Ao encontrar um objecto RPT_ElementAreas na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

- **Cria Xml de um elemento Row**

Ao encontrar um objecto RPT_ElementRow na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

- **Cria Xml de um elemento Div**

Ao encontrar um objecto RPT_ElementDiv na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

- **Cria Xml de um elemento Tab**

Ao encontrar um objecto RPT_ElementTab na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

- **Cria Xml de um elemento Tr**

Ao encontrar um objecto RPT_ElementTr na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

- **Cria Xml de correspondente a um sub - relatório em relatório Pai**

Caso de um que representa a criação de código Xml correspondente a um sub-relatório que faça parte da composição de um relatório principal.

- **Cria Xml de um elemento ExplorerGroup**

Ao encontrar um objecto RPT_ElementExplorerGroup na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

- **Cria Xml de um elemento Grid**

Ao encontrar um objecto RPT_ElementGrid na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

- **Cria Xml de um elemento ExplorerCol**

Ao encontrar um objecto RPT_ElementExplorerCol na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

- **Cria Xml de um elemento Rows**

Ao encontrar um objecto RPT_ElementRows na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

- **Cria Xml de um elemento Hr**

Ao encontrar um objecto RPT_ElementHr na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

- **Cria Xml de um elemento Field**

Ao encontrar um objecto RPT_Field na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

- **Cria Xml de um elemento Explorer**

Ao encontrar um objecto RPT_ElementExplorer na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

- **Cria Xml de um elemento Section**

Ao encontrar um objecto RPT_ElementSection na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

- **Cria Xml de um elemento Root**

Ao encontrar um objecto RPT_Root na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

- **Cria Xml de um elemento Col**

Ao encontrar um objecto RPT_ElementCol na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

- **Cria Xml de um elemento Cell**

Ao encontrar um objecto RPT_ElementCell na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

- **Cria Xml de um elemento Attribute**

Ao encontrar um objecto RPT_ElementAttribute na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

- **Criar Xml de um elemento Area**

Ao encontrar um objecto RPT_ElementArea na estrutura em árvore, este caso de uso representa a criação do correspondente código em xml.

Anexo C

Em seguida é explicado como são compostos os diversos objectos que constituem a árvore intermédia.

- **ConstroiArvoreRPT_Areas** – Objecto que faz a construção de um objecto RPT_ElementAreas.

  Construtor:

  - *ConstroiArvoreRPT_Areas();* - Construtor responsável pela criação de um novo objecto ConstroiArvoreRPT_Areas.

  Métodos:

  - *criaArvoreAreas(ngtXMLHandler areas, RPT_Helper helper);* - Método que constrói e preenche alguns atributos de um objecto do tipo RPT_ElementAreas. Poderá haver lugar ao inicio de construção de objectos do tipo RPT_ElementArea, assim estes possam ser inseridos no "vector filhos" do objecto RPT_ElementAreas e também na estrutura do código XML que se encontra a ser analisado.
    - Parâmetro de entrada – Objecto do tipo ngtXMLHandler dom o nó xml a ser analisado, objecto de apoio do tipo RPT_Helper.
    - Parâmetro de retorno – Objecto do tipo RPT_ElementAreas.

- **ConstroiArvoreRPT_Area** – Objecto que faz a construção de um objecto RPT_ElementArea.

  Construtor:

  - *ConstroiArvoreRPT_Area();* - Construtor responsável pela criação de um novo objecto ConstroiArvoreRPT_Area.

  Métodos:

  - *criaArvoreArea(ngtXMLHandler area, RPT_Helper helper);* - Método que constrói e preenche alguns atributos de um objecto do tipo RPT_ElementArea. Pode haver também lugar ao inicio de construção de outro tipo de objectos, assim estes possam ser inseridos no "vector filhos" do objecto RPT_ElementArea e também na estrutura do código XML que se encontra a ser analisado.

    - Parâmetro de entrada – Objecto do tipo ngtXMLHandler dom o nó xml a ser analisado, objecto de apoio do tipo RPT_Helper.
    - Parâmetro de retorno – Objecto do tipo RPT_ElementArea.

- **ConstroiArvoreRPT_Tab** – Objecto que faz a construção de um objecto RPT_ElementTab.

  Construtor:

- *ConstroiArvoreRPT_Tab();* - Construtor responsável pela criação de um novo objecto ConstroiArvoreRPT_Tab.

Métodos:

- *criaArvoreTab(ngtXMLHandler tab, RPT_Helper helper);* - Método que constrói e preenche alguns atributos de um objecto do tipo RPT_ElementTab. Pode haver também lugar ao inicio de construção de outro tipo de objectos, assim estes possam ser inseridos no "vector filhos" do objecto RPT_ElementTab e também na estrutura do código XML que se encontra a ser analisado.
  - Parâmetro de entrada – Objecto do tipo ngtXMLHandler dom o nó xml a ser analisado, objecto de apoio do tipo RPT_Helper.

    Parâmetro de retorno – Objecto do tipo RPT_ElementTab.

- **ConstroiArvoreRPT_Section** – Objecto que faz a construção de um objecto RPT_ElementSection.

Construtor:

- *ConstroiArvoreRPT_Section();* - Construtor responsável pela criação de um novo objecto ConstroiArvoreRPT_Section.

Métodos:

- *criaArvoreSection(ngtXMLHandler tab, RPT_Helper helper);* - Método que constrói e preenche alguns atributos de um objecto do tipo RPT_ElementSection. Poderá haver lugar ao inicio de construção de objectos do tipo RPT_ElementRows, assim estes

possam ser inseridos no "vector filhos" do objecto RPT_ElementSection e também na estrutura do código XML que se encontra a ser analisado.

- Parâmetro de entrada – Objecto do tipo ngtXMLHandler dom o nó xml a ser analisado, objecto de apoio do tipo RPT_Helper.

  Parâmetro de retorno – Objecto do tipo RPT_ElementSection.

- **ConstroiArvoreRPT_Rows** – Objecto que faz a construção de um objecto RPT_ElementRows.

  Construtor:

  - *ConstroiArvoreRPT_Rows();* - Construtor responsável pela criação de um novo objecto ConstroiArvoreRPT_Rows.

  Métodos:

  - *criaArvoreRows(ngtXMLHandler tab, RPT_Helper helper);* - Método que constrói e preenche alguns atributos de um objecto do tipo RPT_ElementRows. Poderá haver lugar ao inicio de construção de objectos do tipo RPT_ElementRow, assim estes possam ser inseridos no "vector filhos" do objecto RPT_ElementRow e também na estrutura do código XML que se encontra a ser analisado.

    - Parâmetro de entrada – Objecto do tipo ngtXMLHandler dom o nó xml a ser analisado, objecto de apoio do tipo RPT_Helper.

Parâmetro de retorno – Objecto do tipo RPT_ElementRows.

- **ConstroiArvoreRPT_Row** – Objecto que faz a construção de um objecto RPT_ElementRow.

  Construtor:

  - *ConstroiArvoreRPT_Row();* - Construtor responsável pela criação de um novo objecto ConstroiArvoreRPT_Row.

  Métodos:

  - *criaArvoreRow(ngtXMLHandler tab, RPT_Helper helper);* - Método que constrói e preenche alguns atributos de um objecto do tipo RPT_ElementRow. Poderá haver lugar ao inicio de construção de objectos do tipo RPT_ElementCell, assim estes possam ser inseridos no "vector filhos" do objecto RPT_ElementRow e também na estrutura do código XML que se encontra a ser analisado.

    - Parâmetro de entrada – Objecto do tipo ngtXMLHandler dom o nó xml a ser analisado, objecto de apoio do tipo RPT_Helper.

      Parâmetro de retorno – Objecto do tipo RPT_ElementRow.

- **ConstroiArvoreRPT_Cell** – Objecto que faz a construção de um objecto RPT_ElementCell.

  Construtor:

  - *ConstroiArvoreRPT_Cell();* - Construtor responsável pela criação de um novo objecto ConstroiArvoreRPT_Cell.

  Métodos:

  - *criaArvoreRow(ngtXMLHandler Row, RPT_Helper helper);* - Método que constrói e preenche alguns atributos de um objecto do tipo RPT_ElementCell. Poderá haver lugar ao inicio de construção de objectos do tipo RPT_ElementAttribute, assim estes possam ser inseridos no "vector filhos" do objecto RPT_ElementCell e também na estrutura do código XML que se encontra a ser analisado.

    - Parâmetro de entrada – Objecto do tipo ngtXMLHandler dom o nó xml a ser analisado, objecto de apoio do tipo RPT_Helper.

      Parâmetro de retorno – Objecto do tipo RPT_ElementCell.

- **ConstroiArvoreRPT_Attribute** – Objecto que faz a construção de um objecto RPT_ElementAttribute.

  Construtor:

  - *ConstroiArvoreRPT_Attribute();* - Construtor responsável pela criação de um novo objecto ConstroiArvoreRPT_Cell.

Métodos:

- *criaArvoreRow(ngtXMLHandler Attribute, RPT_Helper helper);* - Método que constrói e preenche alguns atributos de um objecto do tipo RPT_ElementAttribute. Esta construção inclui também o começo da construção de um objecto do tipo RPT_Field que deverá ser inserido no vector de fields, que se encontra no objecto RPT_Helper.
  - Parâmetro de entrada – Objecto do tipo ngtXMLHandler dom o nó xml a ser analisado, objecto de apoio do tipo RPT_Helper.

    Parâmetro de retorno – Objecto do tipo RPT_ElementCell.

- **ConstroiArvore**RPT_Div – Objecto que faz a construção de um objecto RPT_ElementDiv.

  Construtor:

  - *ConstroiArvoreRPT_Div();* - Construtor responsável pela criação de um novo objecto ConstroiArvoreRPT_Div.

  Métodos:

  - *criaArvoreDiv(ngtXMLHandler div, RPT_Helper helper);* - Método que constrói e preenche alguns atributos de um objecto do tipo RPT_ElementDiv. Esta construção inclui também o inicio de construção de um novo relatório, neste caso de um ecrã de edição do tipo LIST.

- Parâmetro de entrada – Objecto do tipo ngtXMLHandler dom o nó xml a ser analisado, objecto de apoio do tipo RPT_Helper.

  Parâmetro de retorno – Objecto do tipo RPT_ElementDiv.

- **ConstroiArvoreRPT_Table** – Objecto que faz a construção de um objecto RPT_ElementTable.

  Construtor:

  - *ConstroiArvoreRPT_Table();* - Construtor responsável pela criação de um novo objecto ConstroiArvoreRPT_Table.

  Métodos:

  - *criaArvoreTable(ngtXMLHandler table, RPT_Helper helper);* - Método que constrói e preenche alguns atributos de um objecto do tipo RPT_ElementTable. Poderá haver lugar ao inicio de construção de objectos do tipo RPT_ElementTr, assim estes possam ser inseridos no "vector filhos" do objecto RPT_ElementTable e também na estrutura do código XML que se encontra a ser analisado.

    - Parâmetro de entrada – Objecto do tipo ngtXMLHandler dom o nó xml a ser analisado, objecto de apoio do tipo RPT_Helper.

      Parâmetro de retorno – Objecto do tipo RPT_ElementTable.

- **ConstroiArvoreRPT_Tr** – Objecto que faz a construção de um objecto RPT_ElementTr.

  Construtor:

  - *ConstroiArvoreRPT_Tr();* - Construtor responsável pela criação de um novo objecto ConstroiArvoreRPT_Tr.

  Métodos:

  - *criaArvoreTr(ngtXMLHandler tr, RPT_Helper helper);* - Método que constrói e preenche alguns atributos de um objecto do tipo RPT_ElementTr. Poderá haver lugar ao inicio de construção de objectos do tipo RPT_ElementTd, assim estes possam ser inseridos no "vector filhos" do objecto RPT_ElementTr e também na estrutura do código XML que se encontra a ser analisado.

    - Parâmetro de entrada – Objecto do tipo ngtXMLHandler dom o nó xml a ser analisado, objecto de apoio do tipo RPT_Helper.

      Parâmetro de retorno – Objecto do tipo RPT_ElementTr.

- **ConstroiArvoreRPT_Td** – Objecto que faz a construção de um objecto RPT_ElementTd.

  Construtor:

  - *ConstroiArvoreRPT_Td();* - Construtor responsável pela criação de um novo objecto ConstroiArvoreRPT_Td.

  Métodos:

  - *criaArvoreTd(ngtXMLHandler td, RPT_Helper helper);* - Método que constrói e preenche alguns atributos de um objecto

do tipo RPT_ElementTd. Poderá haver lugar ao inicio de construção de objectos do tipo RPT_ElementHr, assim estes possam ser inseridos no "vector filhos" do objecto RPT_ElementTd e também na estrutura do código XML que se encontra a ser analisado.

- Parâmetro de entrada – Objecto do tipo ngtXMLHandler dom o nó xml a ser analisado, objecto de apoio do tipo RPT_Helper.
- Parâmetro de retorno – Objecto do tipo RPT_ElementTd

- **ConstroiArvoreRPT_Hr** – Objecto que faz a construção de um objecto RPT_ElementHr.

  Construtor:

  - *ConstroiArvoreRPT_Hr();* - Construtor responsável pela criação de um novo objecto ConstroiArvoreRPT_Hr.

  Métodos:

  - *criaArvoreHr(ngtXMLHandler hr, RPT_Helper helper);* - Método que constrói e preenche alguns atributos de um objecto do tipo RPT_ElementHr. Poderá haver lugar ao inicio de construção de objectos do tipo RPT_ElementHd, assim estes possam ser inseridos no "vector filhos" do objecto RPT_ElementHr e também na estrutura do código XML que se encontra a ser analisado.
    - Parâmetro de entrada – Objecto do tipo ngtXMLHandler dom o nó xml a ser analisado, objecto de apoio do tipo RPT_Helper.

- Parâmetro de retorno – Objecto do tipo RPT_ElementHr

- **ConstroiArvoreRPT_Panel** – Objecto que faz a construção de um objecto RPT_ElementPanel.

  Construtor:

  - *ConstroiArvoreRPT_Panel();* - Construtor responsável pela criação de um novo objecto ConstroiArvoreRPT_Panel.

  Métodos:

  - *criaArvorePanel(ngtXMLHandler panel, RPT_Helper helper);* - Método que constrói e preenche alguns atributos de um objecto do tipo RPT_ElementPanel. Pode haver também lugar ao inicio de construção de outro tipo de objectos, assim estes possam ser inseridos no "vector filhos" do objecto RPT_ElementPanel e também na estrutura do código XML que se encontra a ser analisado.

    - Parâmetro de entrada – Objecto do tipo ngtXMLHandler dom o nó xml a ser analisado, objecto de apoio do tipo RPT_Helper.

    - Parâmetro de retorno – Objecto do tipo RPT_ElementPanel

- **ConstroiArvoreRPT_Grid** – Objecto que faz a construção de um objecto RPT_ElementGrid.

  Construtor:

- *ConstroiArvoreRPT_Grid();* - Construtor responsável pela criação de um novo objecto ConstroiArvoreRPT_Grid.

Métodos:

- *criaArvoreGrid(ngtXMLHandler grid, RPT_Helper helper);* - Método que constrói e preenche alguns atributos de um objecto do tipo RPT_ElementGrid. Poderá haver lugar ao inicio de construção de objectos do tipo RPT_ElementCols, assim estes possam ser inseridos no "vector filhos" do objecto RPT_ElementGrid e também na estrutura do código XML que se encontra a ser analisado.
  - Parâmetro de entrada – Objecto do tipo ngtXMLHandler dom o nó xml a ser analisado, objecto de apoio do tipo RPT_Helper.
  - Parâmetro de retorno – Objecto do tipo RPT_ElementGrid

- **ConstroiArvoreRPT_Cols** – Objecto que faz a construção de um objecto RPT_ElementCols.

Construtor:

- *ConstroiArvoreRPT_Cols();* - Construtor responsável pela criação de um novo objecto ConstroiArvoreRPT_Cols.

Métodos:

- *criaArvoreCols(ngtXMLHandler cols, RPT_Helper helper);* - Método que constrói e preenche alguns atributos de um objecto do tipo RPT_ElementCols. Poderá haver lugar ao inicio de construção de objectos do tipo RPT_ElementCol, assim estes

possam ser inseridos no "vector filhos" do objecto RPT_ElementCols e também na estrutura do código XML que se encontra a ser analisado.

- Parâmetro de entrada – Objecto do tipo ngtXMLHandler dom o nó xml a ser analisado, objecto de apoio do tipo RPT_Helper.
- Parâmetro de retorno – Objecto do tipo RPT_ElementCols

- **ConstroiArvoreRPT_Col** – Objecto que faz a construção de um objecto RPT_ElementCol.

  Construtor:

  - *ConstroiArvoreRPT_Col();* - Construtor responsável pela criação de um novo objecto ConstroiArvoreRPT_Col.

  Métodos:

  - *criaArvoreCol(ngtXMLHandler col, RPT_Helper helper);* - Método que constrói e preenche alguns atributos de um objecto do tipo RPT_ElementCol. Esta construção inclui também o começo da construção de um objecto do tipo RPT_Field que deverá ser inserido no vector de fields, que se encontra no objecto RPT_Helper.
    - Parâmetro de entrada – Objecto do tipo ngtXMLHandler dom o nó xml a ser analisado, objecto de apoio do tipo RPT_Helper.
    - Parâmetro de retorno – Objecto do tipo RPT_ElementCol

- **ConstroiArvoreExplorer** – Objecto que faz a construção de um objecto RPT_ElementExplorer.

  Construtor:

  - *ConstroiArvoreExplorer();* - Construtor responsável pela criação de um novo objecto ConstroiArvoreExplorer.

  Métodos:

  - *constroiArvoreExplorer(Explorer explorer, RPT_Helper helper);* - Método que constrói e preenche alguns atributos de um objecto do tipo RPT_ElementExplorer. Poderá haver lugar ao inicio de construção de objectos do tipo RPT_ElementExplorerColumn ou RPT_ElementExplorerGroup assim estes possam ser inseridos no "vector filhos" do objecto RPT_ElementExplorer e também na estrutura do código XML que se encontra a ser analisado.

    - Parâmetro de entrada – Objecto do tipo ngtXMLHandler dom o nó xml a ser analisado, objecto de apoio do tipo RPT_Helper.

    - Parâmetro de retorno – Objecto do tipo RPT_ElementExplorer

- **ConstroiArvoreRPT_ExplorerColumn**– Objecto que faz a construção de um objecto RPT_ElementCol.

  Construtor:

- *ConstroiArvoreRPT_ExplorerColumn();* - Construtor responsável pela criação de um novo objecto ConstroiArvoreRPT_ExplorerColumn.

Métodos:

- *criaArvoreExplorercolumn(ColumnProvider coluna, RPT_Helper helper, int indice, int larguraMaxima)* - Método que constrói e preenche alguns atributos de um objecto do tipo RPT_ElementExplorerColumn. Esta construção inclui também o começo da construção de um objecto do tipo RPT_Field que deverá ser inserido no vector de fields, que se encontra no objecto RPT_Helper.
  - Parâmetro de entrada – Objecto XEO do tipo ColumnProvider, objecto de apoio do tipo RPT_Helper, int índice, int larguraMaxima.
  - Parâmetro de retorno – Objecto do tipo RPT_ElementExplorerColumn

- **ConstroiArvoreExplorerGroup** – Objecto que faz a construção de um objecto RPT_ElementExplorerGroup.

Construtor:

- *ConstroiArvoreExplorerGroup();* - Construtor responsável pela criação de um novo objecto ConstroiArvoreExplorerGroup.

Métodos:

- *constroiArvoreExplorerGroup(ColumnProvider colunaGroup, RPT_Helper helper)* - Método que constrói e preenche alguns atributos de um objecto do tipo RPT_ElementExplorerGroup.

Esta construção inclui também o começo da construção de um objecto do tipo RPT_Field que deverá ser inserido no vector de fields, que se encontra no objecto RPT_Helper.

- Parâmetro de entrada – Objecto XEO do tipo ColumnProvider, objecto de apoio do tipo RPT_Helper.
- Parâmetro de retorno – Objecto do tipo RPT_ElementExplorerGroup

Anexo D

Em seguida é explicado como são constituídos diversos objectos responsáveis pela criação de código xml.

- **CreateXMLArea** – Objecto que faz a construção de código xml relativo a um objecto RPT_Area.

  Construtor:

  - *CreateXmlArea()* - Construtor responsável pela criação de um novo objecto CreateXmlArea.

  Métodos:

  - *escreveXmlArea(StringBuffer xmlCode, RPT_ElementArea area, int comprimento_detail, int a, int b)* – Método que constrói o código xml relativo a de um "TextField", elemento de um relatório.

    - Parâmetro de entrada – Codigo construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer, int comprimento_detail, objecto do tipo RPT_ElementArea, objectos do tipo int com variáveis de colocação no relatório.

- **CreateXmlAttribute** – Objecto que faz a construção de código xml relativo a um objecto RPT_ElementAttribute.

Construtor:

- *CreateXmlAttribute()* - Construtor responsável pela criação de um novo objecto CreateXmlAttribute.

Métodos:

- *escreveXmlAttributeStaticText(RPT_ElementAttribute attribute, StringBuffer xmlCode, int a, int b, int alturaCampo)* – Método que constrói o código xml relativo ao inicio de um "Static Text", elemento de um relatório.
  - Parâmetro de entrada – Codigo construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer, int comprimento_detail, objecto do tipo RPT_ElementAttribute, objectos do tipo int com variáveis de colocação no relatório.
- *escreveXmlAttributeDynamicText(RPT_ElementAttribute attribute, StringBuffer xmlCode, int x_dinamico, int b, int alturaCampo, int comprimento_dinamico, String tipo)* – Método que constrói o código xml relativo a um "TextField", elemento de um relatório.
  - Parâmetro de entrada – Codigo construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer, objecto do tipo RPT_ElementAttribute, objectos do tipo int com variáveis de

colocação no relatório e o tipo do attributo de que esta a ser criado o codigo.

- *escreveXmlAttributeDynamicTextBoolean(RPT_ElementAttribute attribute, StringBuffer xmlCode, int x_dinamico, int b, int alturaCampo, int comprimento_dinamico, String tipo)* – Método muito semelhante ao anterior mas com a diferença de que este é utilizado para objectos RPT_ElementAttribute de tipo booleano,

- **CreateXmlCol** – Objecto que faz a construção de código xml relativo a um objecto RPT_ElementCol.

  Construtor:

  - *CreateXmlCol()* - Construtor responsável pela criação de um novo objecto CreateXmlCol.

  Métodos:

  - *escreveXmlStaticText(StringBuffer xmlCode, int comprimento_campo_list, int coord_x_list, RPT_ElementCol col)* – Método que constrói o código xml relativo a um "StaticText", elemento de um relatório.

    - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer, objecto do tipo RPT_ElementArea, objectos do tipo int com variáveis de colocação no relatório.

  - *escreveXmlTextField(StringBuffer xmlCode, int comprimento_campo_list, int coord_x_list, RPT_ElementCol col,*

String tipo) – Método muito semelhante ao anterior mas com a diferença que este constrói o código xml referente a um elemento dinâmico de um relatório.

- **CreateXmlCols** – Objecto que faz a construção de código xml relativo a um objecto RPT_ElementCols.

  Construtor:

  - *CreateXmlCols()* - Construtor responsável pela criação de um novo objecto CreateXmlCols.

  Métodos:

  - *abreCabecalhoColuna(StringBuffer xmlCode, RPT_ElementCols cols)* – Método que constrói o código xml relativo ao inicio de um "ColumnHeader", elemento de um relatório.
    - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer, objecto do tipo RPT_ElementCols.
  - *fechaCabecalhoColuna(StringBuffer xmlCode)* – Método que constrói o código xml relativo ao final de um "ColumnHeader", elemento de um relatório.
    - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer.

- *abreCabecalhoDetail(StringBuffer xmlCode, RPT_ElementCols cols)* – Método que constrói o código xml relativo ao inicio de um "Detail", elemento de um relatório.
  - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer, objecto do tipo RPT_ElementCols.

- *fechaCabecalhoDetail(StringBuffer xmlCode)* – Método que constrói o código xml relativo ao final de um "Detail", elemento de um relatório.
  - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer.

- **CreateXmlDiv** – Objecto que faz a construção de código xml relativo a um objecto RPT_ElementDiv.

  Construtor:

  - *CreateXmlDiv()* - Construtor responsável pela criação de um novo objecto CreateXmlDiv.

  Métodos:

  - *constroiXmlDiv(StringBuffer xmlCode, int y, RPT_ElementDiv div)* – Método que constrói o código xml relativo a um "subreport", elemento de um relatório.

- Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer, objecto do tipo RPT_ElementDiv.

- **CreateXmlExplorer** – Objecto que faz a construção de código xml relativo a um objecto RPT_ElementExplorer.

  Construtor:

  - *CreateXmlExplorer()* - Construtor responsável pela criação de um novo objecto CreateXmlExplorer.

  Métodos:

  - *iniciaXmlExplorer(StringBuffer xmlCode, RPT_ElementExplorer explorer)* – Método que constrói o código xml relativo a um "TextField", elemento de um relatório.

    - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer, objecto do tipo RPT_ElementExplorer.

- **CreateXmlExplorerCol** – Objecto que faz a construção de código xml relativo a um objecto RPT_ElementExplorerCol.

  Construtor:

  - *CreateXmlExplorerCol()* - Construtor responsável pela criação de um novo objecto CreateXmlExplorerCol.

Métodos:

- *escreveXmlStaticText(StringBuffer xmlCode, int comprimento_campo_list, int coord_x_list, RPT_ElementExplorerColumn explorerCol)* – Método que constrói o código xml relativo a um "StaticText", elemento de um relatório.
  - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer, objecto do tipo RPT_ElementExplorerColumn, objectos do tipo int com variáveis de colocação no relatório.
- *escreveXmlTextField(StringBuffer xmlCode, int comprimento_campo_list, int coord_x_list, RPT_ElementExplorerColumn explorerCol, String tipo)* – Método que constrói o código xml relativo a um "TextField", elemento dinâmico de um relatório.
  - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer, objecto do tipo RPT_ElementExplorerColumn, objectos do tipo int com variáveis de colocação no relatório e o tipo do atributo de que está a ser criado o código.

- **CreateXmlExplorerGroup** – Objecto que faz a construção de código xml relativo a um objecto RPT_ElementExplorerGroup.

  Construtor:

  - *CreateXmlExplorerGroup()* - Construtor responsável pela criação de um novo objecto CreateXmlExplorerGroup.

  Métodos:

  - *iniciaXmlExplorerGroup(StringBuffer xmlCode, RPT_ElementExplorerGroup explorerGroup)* – Método que constrói o código xml relativo a um "Group" e todos os seus elementos subjacentes.

    - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer, objecto do tipo RPT_ElementExplorerGroup.

- CreateXmlField – Objecto que faz a construção de código xml relativo a um objecto RPT_ Field.

  Construtor:

  - *CreateXmlField()* - Construtor responsável pela criação de um novo objecto CreateXmlField.

  Métodos:

  - *escreveXmlField(StringBuffer xmlCode, RPT_Field field, String tipoField)* – Método que constrói o código xml relativo a um "Field", elemento de um relatório.

- Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer, objecto do tipo RPT_Field, e o tipo do atributo de que está a ser criado o código.

- **CreateXmlHelper** – Objecto que faz a construção de código xml relativo a um objecto RPT_ Helper.

  Construtor:

  - *CreateXmlHelper()* - Construtor responsável pela criação de um novo objecto CreateXmlHelper.

  Métodos:

  - *escreveTitle(StringBuffer xmlCode, RPT_Helper helper)* – Método que constrói o código xml relativo a um "Title", elemento de um relatório.
    - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer, objecto do tipo RPT_Helper.
  - *escreveParametros(StringBuffer xmlCode, RPT_Helper helper)* – Método que constrói o código xml relativo todos os elementos "parameter", presentes no relatório.
    - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer, objecto do tipo RPT_Helper.

- **CreateXmlHr** – Objecto que faz a construção de código xml relativo a um objecto RPT_ElementHr.

  Construtor:

  - *CreateXmlHr()* - Construtor responsável pela criação de um novo objecto CreateXmlHr.

  Métodos:

  - *criaLinha(RPT_ElementHr hr, StringBuffer xmlCode,int y)* – Método que constrói o código xml relativo a um "Line", elemento de um relatório.
    - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer, objecto do tipo RPT_Helper e um objecto do tipo int com a coordenada onde será colocado o elemento em construção.

- **CreateXmlInicioFimPaiRPT** – Objecto que faz a construção de código xml relativo ao inicio e fim de um relatório. Trata-se de um pedaço de código estático.

  Construtor:

  - *CreateXmlInicioFimPaiRPT()* - Construtor responsável pela criação de um novo objecto CreateXmlInicioFimPaiRPT.

  Métodos:

- *iniciaXmlPaiRPT(StringBuffer xmlCode, String directoria)* – Método que constrói o código respeitante ao inicio de um relatório.
  - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer e um objecto do tipo "String" com a directoria onde estão colocados os ficheiros temporários.

- *finalizaXmlPaiRPT(StringBuffer xmlCode)* – Método que constrói o código respeitante ao final de um relatório.
  - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer.

- **CreateXmlInicioFimSubRPT** – Objecto que faz a construção de código xml relativo ao inicio e fim de um sub - relatório. Trata-se de um pedaço de código estático.

  Construtor:

  - *CreateXmlInicioFimSubRPT()* - Construtor responsável pela criação de um novo objecto CreateXmlInicioFimSubRPT.

  Métodos:

  - *iniciaXmlSubRPT(StringBuffer xmlCode, String directoria)* – Método que constrói o código respeitante ao inicio de um sub - relatório.

- Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer e um objecto do tipo "String" com a directoria onde estão colocados os ficheiros temporários.

- *finalizaXmlSubRPT(StringBuffer xmlCode, String ambiente)* – Método que constrói o código respeitante ao final de um sub - relatório.

  - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer e um objecto do tipo "String" com o tipo de relatório que está a ser construído, Edit, List ou Explorer.

- **CreateXmlRows** – Objecto que faz a construção de código xml relativo a um objecto RPT_ ElementRows.

  Construtor:

  - *CreateXmlRows()* - Construtor responsável pela criação de um novo objecto CreateXmlRows.

  Métodos:

  - *iniciaXmlRows(StringBuffer xmlCode, int comprimento_detail)* – Método que constrói o código xml relativo ao inicio de um "Detail", elemento de um relatório.

- Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer e um objecto do tipo "int" com o comprimento do elemento a ser construído.

- *finalizaXmlRows(StringBuffer xmlCode)*– Método que constrói o código xml relativo ao final de um "Detail", elemento de um relatório.

  - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer.

- **CreateXmlSubRPTPai** – Objecto que faz a construção de código presente num relatório, indicando a existência de um sub – relatório.

  Construtor:

  - *CreateXmlSubRPTPai()* - Construtor responsável pela criação de um novo objecto CreateXmlSubRPTPai.

  Métodos:

  - *escreveXmlSubRptPai(StringBuffer xmlCode, String nomeFicheiro, RPT_Helper helper)*– Método que constrói o código xml relativo ao inicio de um "Group", que tem subjacente um elemento "subReport" que indicara que o relatório principal terá na sua constituição um sub - relatório.

    - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer, objecto do tipo "String" com o nome do

ficheiro que contém o sub – relatório e um objecto de apoio ao processo do tipo RPT_Helper.

- **CreateXMLTab**– Objecto que faz a construção de código xml relativo a um objecto RPT_ ElementTab.

  Construtor:

  - *CreateXMLTab()* - Construtor responsável pela criação de um novo objecto CreateXmlTab.

  Métodos:

  - *escreveXmlTab(StringBuffer xmlCode, RPT_ElementTab tab, int y)*– Método que constrói o código xml relativo ao inicio de um "TextField", elemento de um relatório.
    - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer e um objecto do tipo RPT_ElementTab.
  - *finalizaXmlRows(StringBuffer xmlCode)*– Método que constrói o código xml relativo ao final de um "Detail", elemento de um relatório.
    - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer.

- **CreateXmlTr** – Objecto que faz a construção de código xml relativo a um objecto RPT_ElementTr.

  Construtor:

  - *CreateXmlTr()* - Construtor responsável pela criação de um novo objecto CreateXmlTr.

  Métodos:

  - *iniciaGrupoTr(RPT_ElementTr tr, StringBuffer xmlCode)* – Método que constrói o código xml relativo ao inicio de um "Group", elemento de um relatório.
    - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer e um objecto do tipo RPT_ElementTr.
  - *finalizaGrupoTr(StringBuffer xmlCode)* – Método que constrói o código xml relativo ao final de um "Group", elemento de um relatório.
    - Parâmetro de entrada – Código construído até ao momento de execução deste método sob a forma de um objecto do tipo StringBuffer.