Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Informática
Programa de Pós-Graduação em Ciência da Computação

Modelagem e Produção de Documentos Personalizados com Conteúdo Variável

Maria Isabel Giusti Moreira

Dissertação apresentada como requisito parcial à obtenção do grau de mestre em Ciência da Computação

Orientador: Prof. Dr. João Batista S. de Oliveira

Porto Alegre
2007

PUCRS

Campus Central
Av. Ipiranga, 6681 – prédio 16 – CEP 90619-900
Porto Alegre – RS – Brasil
Fone: +55 (51) 3320-3544 – Fax: +55 (51) 3320-3548
Email: bceadm@pucrs.br
www.pucrs.br/biblioteca

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada ***"Modelagem e Produção de Documentos Personalizados com Conteúdo Variável"***, apresentada por Maria Isabel Giusti Moreira, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Computação Cientifica, aprovada em 21/12/2006 pela Comissão Examinadora:

Prof. Dr. João Batista Souza de Oliveira – Orientador — PPGCC/PUCRS

Prof. Dr. Márcio Serolli Pinho – — PPGCC/PUCRS

Profa. Dra. Carla Maria Dal Sasso Freitas – — UFRGS

Homologada em 08/03/2007, conforme Ata No. 005/2007 pela Comissão Coordenadora.

Prof. Dr. Fernando Luís Dotti
Coordenador.

PUCRS
Campus Central
Av. Ipiranga, 6681 – P. 16 – sala 106 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@inf.pucrs.br
www.pucrs.br/facin/pos

Resumo

Empresas e corporações se esforçam para atrair e manter seus clientes e um dos fatores que contribui para isso é conhecer as preferências e características dos seus clientes de forma individualizada. Contudo, de nada adianta essas empresas possuírem essas informações armazenadas em bases de dados sem aplicá-las a documentos com *marketing* individualizado, que apresentem informações referentes aos hábitos e costumes de um cliente. Para que esses conhecimentos sejam utilizados na prática, através da geração de documentos personalizados, é necessária a utilização da tecnologia de dados variáveis.

Essa tecnologia de dados variáveis pode ser considerada ainda bastante incipiente pela falta de pesquisas em relação a modelos, linguagens e até mesmo *software* que facilitem o processo de modelagem e geração de documentos personalizados utilizando conteúdos variáveis.

Por existir essa deficiência sobre a tecnologia, neste trabalho será desenvolvida uma revisão bibliográfica sobre os principais conceitos da tecnologia de dados variáveis, além de apresentar a sua utilização em conjunto com a técnica de estruturação de documentos, que se utilizadas juntas facilitarão o controle das escolhas dos diferentes conteúdos contidos em uma árvore estrutural, e com isso será possível fazer, de forma simplificada, a modelagem e geração dos documentos personalizados.

Com base nesses conceitos e principais aspectos positivos dessas tecnologias, será também idealizado e proposto um modelo que reduza as deficiências atualmente encontradas nessas tecnologias, através da formalização do processo de modelagem, mostrando que é possível gerar várias instâncias de documentos personalizados com conteúdos variáveis. Por fim, para validar o modelo formalizado na prática, será desenvolvido um protótipo que simule e execute as ações e comportamentos estipulados para os formalismos que compõem o modelo, mostrando que esse é capaz de gerar documentos personalizados com diferentes conteúdos variáveis com um alto grau de personalização.

Palavras-chave: modelo, documentos, personalizados, conteúdo variável.

Abstract

The companies and corporations are making efforts to attract and to keep customers and realized that it is important to know the preferences and characteristics of its customers. However, it is useless for these companies to possess the information stored in databases without applying them in documents, that present information about the habits of one client. To use this knowledge in practice, through personalized document generation, is necessary the use of variable data technology.

This variable data technology can be considered still incipient by the lack of research with respect to models, languages and even software that facilitates the process of modeling and personalized document generation using variable contents.

For existing this deficiency on the technology, in this work will be developed a bibliographical revision on the main concepts of variable data technology, besides presenting its use in set with the technique of structuring of documents, which used together will facilitate the control of the choices of the different contents contained in a structural tree, and with this will be possible to make, on a simplified form, the modeling and generation of personalized documents.

On the basis of these concepts and main positive aspects of these technologies, also will be idealized and proposed a model that reduces the currently found deficiencies in these technologies, through the legalization of modeling process, showing that it is possible to generate several personalized document instances with variable contents. Finally, to validate the legalized model in the practical way, it will be developed an prototype that simulates and executes the actions and behaviors stipulated for the formalisms that compose the model, showing that this one is capable to generate personalized documents with different variable contents with one high degree of personalization.

Keywords: model, document, personalized, variable contents.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

TI	Tecnologia da Informação	12
CRM	*Customer Relationship Management* ou Gestão de Relação com o cliente	12
VDP	*Variable Data Publishing* ou *Variable Data Printing*	17
CGATS	*Committee for Graphic Arts Technologies Standards*	26
PPML	*Personalized Print Markup Language*	26
PPML/VDX	*Personalized Print Markup Language/Variable Printing Data Exchange*	26
DAG	*Directed Acyclic Graph*	28
SVG	*Scalable Vector Graphics*	65
PNG	*Portable Network Graphics*	65
XML	*eXtensible Markup Language*	80
API	*Application Programming Interface* ou Interface de Programação de Aplicativos	83
SAX	*Simple API for XML*	83
DOM	*Document Object Model*	83
JDBC	*Java DataBase Connectivity*	89

Sumário

1 Introdução

Atualmente, se analisarmos a comunicação que visa atingir o consumidor, sob o ponto de vista evolutivo, podemos perceber que há um caminho progressivo para uma individualização e personalização cada vez mais voltadas às preferências e características de cada cliente.

Visto que os mercados estão cada vez mais competitivos, uma das maneiras mais eficazes de se chegar a essa comunicação e com isso convencer o consumidor a adquirir produtos e serviços de uma determinada empresa ou corporação é oferecendo-lhe propaganda personalizada e exclusiva acarretando assim uma maior fidelidade entre empresa e consumidor. Para se obter essa individualização é necessário utilizar a tecnologia de dados variáveis, a qual permite a geração de documentos personalizados a partir da combinação entre um *layout* básico e a escolha de diferentes conteúdos (como textos e imagens) que são retirados de uma base de dados e escolhidos conforme as características de cada usuário.

Utilizar essa tecnologia para personalização de documentos se assemelha muito com a maneira antiga de comércio local, quando o dono da loja atendia pessoalmente cada cliente e tinha na memória informações como nome, parentes próximos, data de aniversário, hábitos de compras e preferências. Com o advento do computador as empresas que mantêm um contato direto com o seu cliente armazenando em um banco de dados informações como nome, endereço e principalmente hábitos de compras, podem chegar a essa mesma fórmula de sucesso em uma escala bem maior empregando a tecnologia de dados variáveis para documentos personalizados.

Hoje, uma grande parte das empresas já possuem em seus departamentos de Tecnologia da Informação (TI), servidores que armazenam esses dados e que os atualizam constantemente. No passado a integração desse departamento com as outras áreas de negócios (como por exemplo, as áreas de *marketing* e vendas) não existia, mas hoje os profissionais de TI se tornaram pessoas-chave nas áreas estratégicas das empresas, pois é impossível implementar sistemas de CRM (*Customer Relationship Management*) sem essa integração, visto que é o departamento de informática o grande responsável pela manutenção e preservação dos dados, trazendo grande ganho operacional para muitas ações de *marketing*.

A comunicação usando conteúdo variável na geração de documentos personalizados pode ser considerada uma evolução do *marketing* direto, pois permite levar a individualização ainda mais longe, pois todas as informações disponíveis sobre o cliente podem servir de subsídio para a escolha de mensagens específicas de acordo com o perfil de cada público-alvo.

Essa nova abordagem de comunicação personalizada, especialmente a que emprega a técnica de conteúdo variável, deve ser vista como uma nova modalidade de comunicação que conviverá e se integrará com os meios de saída tradicionais sempre buscando explorar as preferências do consumidor, mostrando que essa comunicação é uma excelente forma de competir na economia globalizada e competitiva que se apresenta atualmente.

1.1 Motivação e Contexto de trabalho

Um dos grandes problemas existentes quando se trabalha com personalização de documentos utilizando dados variáveis está relacionado, principalmente, com o alto grau de complexidade de todo o seu design para se obter pequenas alterações nos documentos (como, por exemplo, nome, endereço de um cliente ou pequenas partes de texto) e com o elevado tempo para a geração dos vários documentos personalizados, além do custo para a empresa que deseja utilizar essa tecnologia.

Esses problemas surgem pela falta de modelos, linguagens e até mesmo *software* que facilite o processo de modelagem e geração através de tecnologias simples, ajudando a determinar, entre os conteúdos opcionais disponíveis, os que devem ser utilizados em cada versão de um documento personalizado.

Define-se um documento personalizado com conteúdo variável como qualquer documento composto por diferentes conteúdos e dados que são providos de uma base de dados, obtendo assim uma personalização ilimitada, visto que essas informações se alterarão conforme o perfil de cada cliente, aplicando ao documento diferentes mensagens, produtos, textos e figuras.

A principal finalidade de gerar documentos personalizados é utilizá-la na área de comunicação digital, ou seja, na área de *marketing* personalizado de uma empresa, permitindo que se estabeleça um contato direto e individualizado com seu público alvo, aproveitando os diferentes costumes e atitudes dos clientes, principalmente focando hábitos de compras e uso de serviços.

Baseado nessa finalidade, a principal motivação e proposta deste trabalho é explorar a área de documentos personalizados e propor soluções para as deficiências e problemas encontrados atualmente quando se trabalha com a tecnologia de dados variáveis. Esses problemas serão resolvidos através da criação e formalização de um modelo que reduza as deficiências encontradas e mostre a importância da conexão entre a área de *layout* de documentos e de dados variáveis, aplicadas ao seu principal uso, ou seja para o *marketing* individualizado.

A partir desse modelo, será também desenvolvido um protótipo que ajude a validar os formalismos criados nesse modelo e principalmente a sua capacidade em gerar documentos personalizados. A validação desse protótipo bem como do modelo só se tornará possível com a criação de diferentes documentos baseados em casos reais, os quais serão processados e avaliados através desse protótipo.

Tendo em vista esses aspectos, o ponto de partida deste trabalho visa fazer um estudo dos fundamentos teóricos disponíveis na literatura de engenharia de documentos, que serão descritos nos dois primeiros capítulos, para que se torne possível a formalização de um modelo que seja capaz de solucionar algumas das necessidades e problemas que hoje se tem quando se quer obter documentos personalizados.

Visto que a utilização de dados variáveis pode ser considerada uma nova tecnologia que contribui e trabalha para a geração de documentos com informações personalizadas, no capítulo dois deste trabalho será apresentado um pouco do histórico dessa tecnologia e seus principais conceitos, bem como o futuro e os benefícios que essa área trará quando passar a ser reconhecida como uma necessidade que trará um retorno de fidelidade para as empresas. Além disso, será abordado todo o *workflow* que compõe essa técnica para que se possa gerar um documento

com dados referentes às preferências de um cliente. Neste capítulo, também será dada ênfase ao principal componente desse *workflow*, ou seja, a base de dados, pois é através dela que se torna possível coletar e armazenar grandes quantidades de conhecimentos sobre os clientes, seus produtos, suas preferências e outras informações, e com isso aplicá-las nos documentos dando ao cliente a sensação de um tratamento privilegiado.

No capítulo três serão mostrados os conceitos necessários para transformar um documento não estruturado que utilize dados variáveis em um documento estruturado, para que o modelo a ser proposto, neste trabalho, possa ser representado com base em alguma sugestões encontradas na literatura dessa tecnologia. Essa técnica de transformação é chamada de estruturação de documentos e abordará a principal finalidade de se trabalhar com um documento em seu mais alto nível, fazendo com que as estruturas que compõem o documento passem a ter uma camada de relacionamento entre as suas partes, através da estrutura de representação de uma árvore hierárquica.

Já no quarto capítulo será apresentada a principal motivação deste trabalho, ou seja, a formalização de um modelo desenvolvido com base nos dois temas estudados e tratados nos capítulos anteriores que abordam a fundamentação teórica. Esse modelo, denominado **Modelo para o Projeto de Documentos Personalizados**, tem como principal finalidade representar as partes do documento através de uma árvore genérica e com isso reduzir os problemas encontrados atualmente para a obtenção de várias instâncias personalizadas de um documento, visto que visa facilitar o processo de estruturação e, através do conjunto de formalismos criados para o modelo, evitar que seja necessário o controle de múltiplas árvores, tornando a geração de documentos personalizados com dados variáveis mais ágil, menos custosa e com um maior grau de personalização.

Baseado nesse modelo formalizado será necessário mostrar, através de uma validação, que o mesmo pode ser aplicado na prática obtendo de forma correta as diferentes versões dos documentos personalizados com conteúdo variável.

Inicialmente para que essa validação possa ocorrer é necessário determinar como as árvores modelando os documentos serão representadas para que um sistema seja capaz de reconhecê-las. Através do capítulo cinco serão apresentadas essas definições, estipulando a ferramenta de editoração em que as mesmas serão estruturadas. Será também apresentada a biblioteca criada baseada no modelo proposto, que tornará as árvores capazes de serem desenhadas, composta por um conjunto de formas que representam os formalismos desse modelo. Além disso, será mostrado como essas árvores serão transformadas para que um sistema ou protótipo possa reconhecer as suas partes e os formalismos que classificam os nodos que as compõem.

No sexto capítulo deste trabalho, serão apresentados os principais aspectos e partes de um protótipo desenvolvido na linguagem JAVA, que receberá como entrada as árvores apresentadas no capítulo cinco. Esse protótipo terá como principal finalidade simular os comportamentos estipulados pelo modelo na prática e através disso gerar automaticamente documentos personalizados com diferentes conteúdos variáveis para cada cliente de uma empresa ou instituição. Esses documentos obtidos ao término do processo de geração não serão documentos com conteúdos visualmente disponibilizados em um *layout* prontos para serem impressos ou publicados para o cliente final. No término desse processo de geração através do protótipo serão obtidas árvores

estruturadas contendo os diferentes conteúdos que devem compor os documentos personalizados.

No final deste trabalho, no capítulo sete, será mostrada a parte final desta validação do modelo, que será feita através de um conjunto de três documentos desenvolvidos e inspirados em casos reais que serão processados pelo protótipo provando que esse é capaz de gerar diferentes versões desses documentos baseado em diferentes preferências. Esses documentos criados também servirão de estudos de casos que permitirão avaliar o desempenho e o tempo de todo o processo de geração do protótipo implementado e os aspectos mais relevantes desse modelo. A partir dessa validação será possível, através de experimentos, demonstrar os resultados obtidos durante o processamento do protótipo.

1.2 Estado da arte

Dentro da Ciência da Computação a área de Engenharia de Documentos é relativamente recente e por esse motivo, as referências e principalmente os trabalhos práticos relacionados a essa área utilizando a técnica de dados variáveis, aplicadas a geração e personalização de documentos, ainda são bastantes recentes e limitados.

A maioria dos trabalhos e artigos encontrados referentes às áreas de dados variáveis e de estruturação de documentos estão relacionados com fundamentos teóricos sobre os componentes necessários para utilizar essas técnicas (como aborda artigo *ABC da Impressão Digital de Dados Variável* [EFI do Brasil 2005] desenvolvido pela EFI) ou relacionados com os potenciais e benefícios que essas técnicas serão capazes de causar se utilizadas para a produção de documentos na área de *marketing* individualizado.

Já trabalhos relacionados com o lado mais prático da aplicação dessas tecnologias para se obter algum nível de personalização em documentos para impressão ou publicação e os reais resultados dessa utilização ainda são bastante incipientes e difíceis de ser encontrados. Recentemente, alguns dados referentes à utilização dessas tecnologias aplicadas à impressão digital foram apresentados em artigos desenvolvidos com base em dados de um relatório, denominado *"Variable Data Printing 2006: growth and changes in the marketplace"*, publicado pela *TrendWatch Graphic Arts*.

Segundo o artigo [Duraes 2006], alguns números apresentados nesse relatório apontam que atualmente 37% das gráficas e empresas dos Estados Unidos já produzem algum tipo de impressão digital utilizando dados variáveis para a personalização de documentos, contra 28% verificados em 2004. Já no Brasil, esse setor vem demonstrando indícios de crescimento mas ainda não se tornou realidade para grande parte das empresas, visto que, segundo esse artigo, qualquer novidade demora um pouco para se disseminar e sustentar um bom crescimento no mercado brasileiro.

Atualmente uma grande quantidade de eventos (como o Digitally Cool, promovido pela Xerox) têm sido realizados com a finalidade de apresentar, tanto para o mercado internacional quanto para o brasileiro, os recursos e possibilidades de aplicações que oferecem impressões e publicações de documentos personalizados utilizando uma das principais tendências para o mercado gráfico, ou seja, os dados variáveis. Além desses eventos, para impulsionar a área de

Engenharia de Documentos existe o **DOCENG**, o qual pode ser considerado a principal e mais importante conferência existente nessa área.

2 Dados Variáveis

Empresas e corporações esforçam-se para atrair e manter clientes. Milhões foram gastos em pesquisas e com consultores para descobrir que um dos fatores primordiais para o sucesso do negócio é **conhecer o seu cliente** [Komasinski 2005].

Como conseqüência, sistemas de Gestão de Relação com o Cliente (CRM), arrecadam e armazenam grande quantidade de informações sobre a demografia, padrões de compras e preferências dos usuários.

Porém essas informações arrecadadas e armazenadas em uma base de dados geralmente não são aproveitadas plenamente, principalmente por causa da inexistência de uma tecnologia que gere diferentes documentos personalizados e determine as informações mais importantes dentre as que os sistemas de CRM oferecem. A tecnologia que permite essa geração, mesmo que hoje seja pouco explorada, é baseada no uso de **dados variáveis**, a qual será abordada neste capítulo através do seu conceito e dos componentes que compõem o seu *workflow* típico.

2.1 Conceito de dados variáveis

Os primeiros conceitos relacionados com dados variáveis, de uma forma mais geral, foram introduzidos no contexto de pesquisa e desenvolvidos há mais ou menos seis anos, repercutindo perante pesquisadores e designers com muito entusiasmo. Porém, hoje em dia, pode-se notar que apesar de ser uma técnica bastante interessante provou ser mais difícil e complexa do que era esperado e por esse motivo os dados variáveis não são muito utilizados na prática.

Esse novo segmento da indústria de impressão e publicação não tem ainda um nome específico e por esse motivo é tratado na literatura por diferentes nomes como *variable information printing*, *variable information publishing*, *personalized publishing*, *personalized printing*, *customized printing*, *on-demand printing*, *one-to-one publishing*, entre outros. Além desses, os mais descritivos e os escolhidos para serem utilizados neste trabalho serão ***Variable Data Printing*** ou ***Variable Data Publishing***, ambos também conhecidos por **VDP**.

Pode-se definir **VDP** como uma técnica que gera documentos a partir de uma combinação da fusão de textos e imagens que se alteram, provenientes de uma base de dados específica, com um *layout*, tornando o produto final personalizado e sofisticado conforme as características do usuário. Essa tecnologia proporciona, entre fornecedores e seus clientes, qualidade mais elevada de comunicação já que utiliza bases de dados e dispositivos de saída digitais, como impressoras e monitores, para criar documentos de alta qualidade, para vários meios de saída, mais personalizados e dirigidos ao usuário que se deseja atingir [EFI do Brasil 2005].

Há dois tipos de documentos a serem gerados com dados variáveis: os **transacionais** e os **promocionais**. Os transacionais são documentos cujo o principal objetivo é mostrar um

conjunto de dados que se julga importante a quem irá receber esse documento personalizado, como por exemplo: formulários de inscrição, faturas de materiais vendidos, políticas, contratos, contas declarações e ordens de pagamento. Esses documentos não precisam ter um *layout* criativo ou dinâmico e têm como principal característica a perpetuidade, ou seja, poderão ser utilizados por muito tempo, já que o foco é o dado utilizado e não o formato ou a comunicação visual empregada. Já com os documentos promocionais acontece exatamente ao contrário. O *layout*, nesse caso, será dispensável em curto espaço de tempo, pois a meta é vender uma idéia ou produto de forma exata e única, com criatividade para atrair a atenção do público a ser atingido, como por exemplo: catálogos, encartes, *whitepapers*, matérias promocionais e notícias/matérias para publicação.

Um exemplo de documento promocional gerado com a tecnologia de dados variáveis é um folheto de propaganda para uma loja, onde é necessário que a mesma possua uma base de dados com informações sobre seus clientes (como nome, preferências, produtos adquiridos e os produtos similares que o cliente pode desejar futuramente comprar). Esses conteúdos da base de dados, serão os dados variáveis, visto que mudarão a cada documento conforme o histórico do cliente. É também através dessa base de dados que se torna possível determinar quais conteúdos são os mais relevantes para cada folheto de propaganda, para que o mesmo consiga atingir o objetivo desejado. Esses dados relevantes serão associados a um *layout* pré-definido e desenvolvido por um designer, o qual será composto por *copy holes* vazios, que através de uma referência entre a base de dados e o *layout* receberão os dados variáveis na etapa final, apresentando como resultado um documento com mercadorias diferentes e personalizadas conforme o gosto de cada cliente dessa loja.

A personalização dos documentos com dados variáveis baseia-se no processo de adicionar a um documento um ou mais dados associados a um indivíduo, a uma empresa ou a uma organização. Essa personalização do documento final que varia a cada impressão ou publicação é o principal diferencial do uso de dados variáveis para a maneira tradicional de publicação e impressão estática, onde todos os documentos são idênticos.

Porém, são esses métodos tradicionais que originaram, utilizaram e impulsionaram o que hoje chega mais próximo de personalização com dados variáveis. No método utilizado tradicionalmente, consegue-se obter algum grau de personalização através da técnica de *overprinting*, onde ocorre adição de textos em *layouts* pré-prontos de forma estática, onde não é possível se obter uma seleção de dados mais relevantes e nem a habilidade de adicionar imagens ou gráficos distintos a cada novo documento gerado para os diferentes usuários. Um exemplo atual dessa tecnologia pode ser visto na geração dos documentos mais simples de conteúdo variável como as contas de luz e telefone, onde apenas são alterados textos sobre os clientes sem grandes conseqüências no *layout*.

Para que se obtenha maior aceitabilidade da técnica e um maior nível de personalização deve-se sempre idealizar uma página estática e ser capaz de identificar quais dos dados que têm a capacidade de aumentar o poder da mensagem a ser passada e quais podem ser mudados de acordo com o indivíduo que está lendo o documento. O desenvolvedor também deve possuir a noção de quais dados sobre o usuário são mais apropriados para que sejam explorados conforme com o que se deseja inserir no documento variável.

Segundo [Davis 2001], o alto grau de complexidade dos dados variáveis está relacionado com o fato de que como cada documento é produzido diferentemente, podem ocorrer desde alterações simples, como um nome e um endereço até alterações complexas como imagens e mudança na disposição dos elementos nas páginas, os quais podem influenciar uns aos outros, tornando-os fora dos padrões determinados por seu autor e necessitando de modelos e algoritmos que possam detectar e resolver esse problema.

2.2 *Workflow* dos dados variáveis

Para que um documento gerado com **VDP** seja completo e consistente no seu *workflow* existem componentes principais que devem ser utilizados durante a criação de um documento como os dispositivos de saída, os sistemas de base de dados, as regras de negócio, um *layout* pré-pronto e os conteúdos.

O *workflow*, conforme mostra a Figura 1, começa baseando-se nas regras de negócio que serão as diretrizes para o documento e que influenciarão na escolha dos conteúdos contidos na base de dados ou em um arquivo estático. Esses conteúdos serão adicionados em um *layout* pré-pronto, que contém em seu interior apenas *copy holes* vazios que receberão os conteúdos conforme a disposição determinada no projeto criado pelo desenvolvedor. Ao término, quando todos os conteúdos estiverem prontos para serem inseridos no *layout*, a etapa de processo final é gerada através das diferentes versões do documento para cada usuário, apresentando-se através de um dispositivo de saída, o qual pode ser por impressão ou publicação.

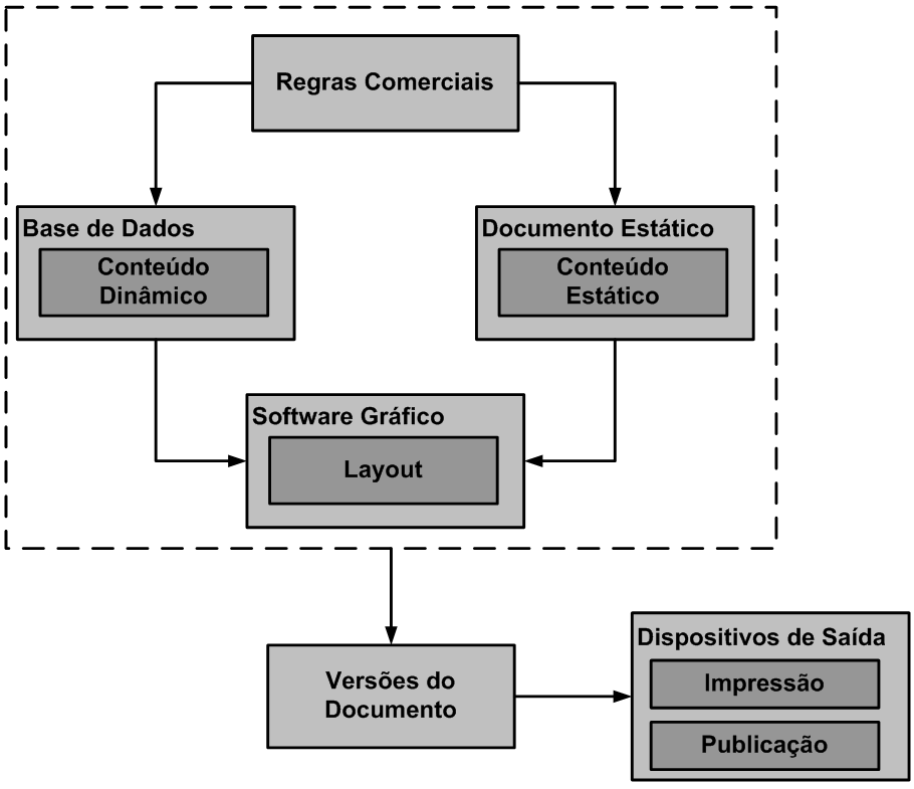


Figura 1 – *Workflow* dos dados variáveis.

Porém esse *workflow* para dados variáveis ainda é bastante precário e pouco desenvolvido, apresentando inexistência de opções sobre uma arquitetura de um modelo, *software* ou linguagem que facilite a escolha e a inserção dos conteúdos no *layout* pré-pronto. Além disso, há deficiência em relação às estratégias para a utilização de cada componente do *workflow*, visto que a maioria das empresas e usuários ainda não estão preparados para possuir, manter ou integrar todos os elementos necessários. Pode-se citar ainda como tarefas críticas e complexas do *workflow*:

- desenvolvimento preciso e apropriado para o *layout* e do projeto, visto que para isso deve haver conhecimento e informações consistentes da base de dados, para os quais a maioria das empresas ainda não estão preparadas ou nem mesmo as possuem;
- automatização de todos os processos (definição das regras de negócio, extração dos conteúdos, inserção desses conteúdos no *layout* e geração do documento final) que compõem o *workflow*, principalmente no processo final em relação aos diferentes dispositivos existentes;
- fornecimento e mineração de informações da base de dados, onde deve-se minerar os dados mais úteis através das regras de negócio;
- otimização da base de dados para gerenciar recursos dos mais simples aos mais complexos, como textos, gráficos, *layout* ou imagens variantes;
- a existência de uma ferramenta de fácil uso para que seja feita a composição dos conjuntos de conteúdos e ligá-los da base de dados ao *layout*;
- tratar o problema da disposição de cada elemento dentro do *layout*, sem que os mesmos influenciem na estética ou na localização dos demais dados que o compõe;
- entrega do documento final ao cliente garantindo a eficiência do projeto, além do baixo custo de cada documento personalizado.

Para amenizar os problemas existentes quando se trabalha com **VDP** deve-se fazer com que os componentes do *workflow* trabalhem em conjunto com serviços que façam tarefas auxiliares, como [EFI do Brasil 2005]:

- **Mescla de bases de dados:** combina informações proveniente de várias bases de dados em apenas uma;
- **Limpeza de dados:** detecta e remove informações obsoletas, incorretas e entradas redundantes;
- **Prospecção de dados:** procura em um grande volume de dados e informações relevantes para uma finalidade específica baseado nas regras de negócio;
- **Serviço de design especializado:** cria *layout* de documentos que irão receber dados variáveis;
- **Gerenciamento de Recursos:** gerencia onde posicionar e como tratar os arquivos e bases de dados com informações;

- **Serviços de mídias cruzadas:** refinará as saídas para diferentes mídias tais como jornais, malas-diretas, folhetos, manuais, publicações eletrônicas e outras.

No decorrer dessa seção os componentes do *workflow* dos dados variáveis serão detalhadamente apresentados.

2.2.1 Regras de negócio

As **regras de negócio** têm a capacidade, dentro deste contexto do *workflow*, de determinar as regras e instruções que especificam quais os conteúdos (textos e imagens, por exemplo) devem ser usados e onde colocá-los no documento a ser gerado [EFI do Brasil 2005]. Essas devem conseguir descrever, de forma prática, as diretrizes dos documentos e por esse motivo devem ser baseadas em aspectos principais como:

- determinar os objetivos e as métricas relevantes sobre o negócio ou foco que se quer atingir escolhendo, por exemplo, os dados mais relevantes contidos na base de dados (como as imagens mais significativas a serem utilizadas);
- selecionar e determinar o público que se deseja atingir;
- basear-se em técnicas eficazes para o sucesso da campanha de propaganda ou da criação de um documento e em técnicas sobre a disposição dos dados dentro do *layout*. Um exemplo desse aspecto seria a fase onde determina-se, dentre os *layouts* disponíveis, qual deve ser escolhido para ser utilizado durante a geração final do documento;
- analisar a coerência do projeto, como por exemplo, fazer estudo das informações que devem ser analisadas na base de dados (como viagens já feitas para não serem repetidas);
- avaliar os resultados dos documentos gerados, fazendo um estudo ou verificar quais produtos aumentarão mais as vendas da empresa;
- basear-se em métodos eficazes de personalização e de produção apropriados para cada tipo de aplicação que se deseja obter, como por exemplo, o tipo de dispositivo de saída em que o documento será gerado e o tamanho do mesmo para que se possa determinar a quantidade de imagens ou tamanho dos textos a serem inseridos.

Especificar e interagir com esses aspectos e principalmente com dados variáveis permite que as regras de negócio consigam atender as exigências e gerem, através do *workflow*, o desenvolvimento de documentos com dados variáveis de uma forma eficiente, lógica e com um bom retorno para empresas [Clinkunbroomer 2005].

2.2.2 Projeto

Paralelamente a todo desenvolvimento do *workflow* de **VDP**, existe a criação de um **projeto**, com base no *layout* e nas demais etapas do processo como um todo, que terá a finalidade de de-

terminar quais serão os dados variáveis mais importantes e onde eles serão melhor representados dentro da disposição do documento personalizado (*layout*).

Conforme a Figura 2, para criação desse projeto devem ser fornecidas informações como:

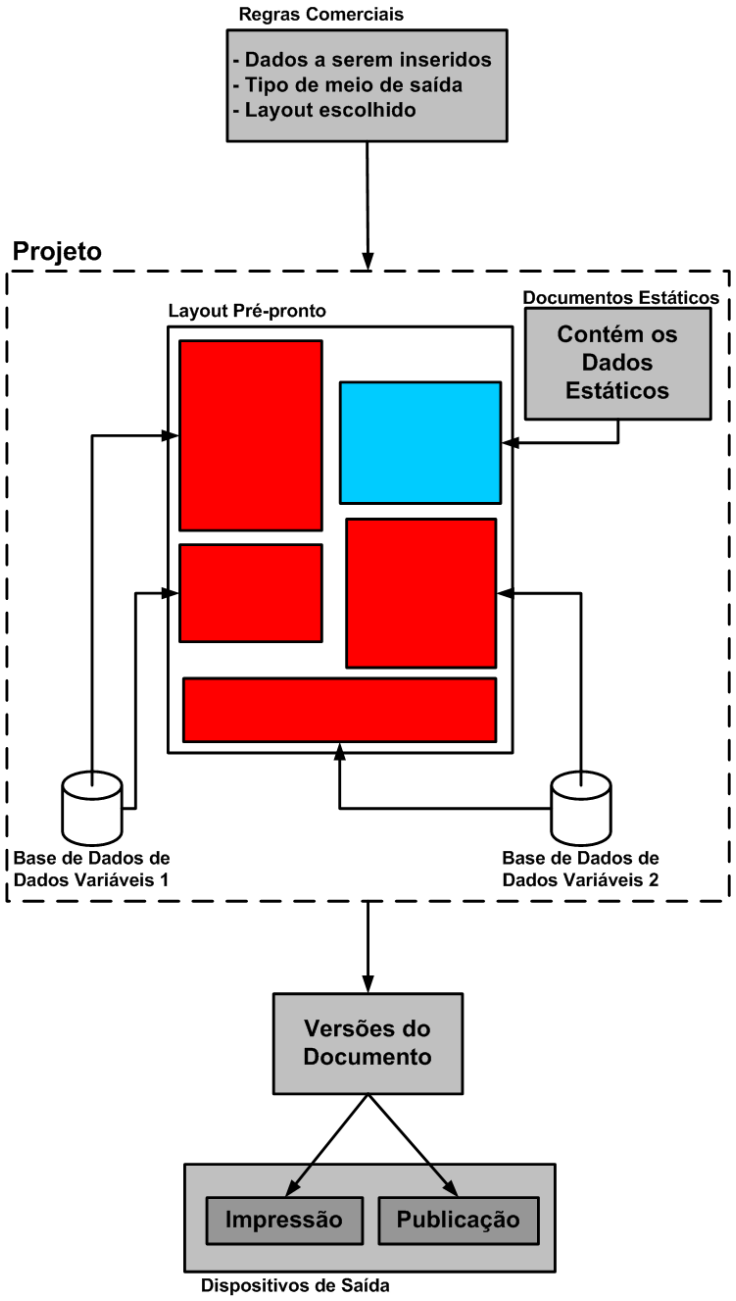


Figura 2 – Informações necessárias para criar um projeto.

- as amostras da base de dados com os dados variáveis que serão utilizados, incluindo os nomes dos campos no arquivo;
- o *layout* já criado, que deve conter espaços livres (denominados *copy holes*) com referências para as regras de negócio, os quais receberão as imagens ou textos e também referências de quais e onde serão colocados os conteúdos utilizados;
- um arquivo de conteúdos estáticos que serão utilizados no *layout*;

- o tipo de dispositivo de saída para que seja possível ter as dimensões do documento e os locais onde os conteúdos podem ser inseridos.

Após essas definições, o projeto repassará ao *layout* tais informações sobre os dados, ou seja, os principais conteúdos que deverão ser inseridos a cada documento, os quais serão sobrepostos nos lugares já determinados (*copy holes*) para os conteúdos.

2.2.3 Conteúdos

Os **conteúdos** são compostos por dados (como textos, imagens e gráficos) que serão inseridos dentro do documento a ser gerado, os quais podem ser obtidos através de uma grande variedade de *software* como editores de textos, câmeras ou uma entrada digital (*scanner*) para as imagens ou fotografias e *software* gráfico para criar diferentes tipos de imagens, entre outros.

Dentro dos dados variáveis, esses conteúdos e seus respectivos dados são classificados em dois tipos principais, denominados **Conteúdo Estático** e **Conteúdo Variável**, os quais podem ser utilizados em conjunto num mesmo documento, criando assim um documento misto [EFI 2005].

Classifica-se como **conteúdo estático** aquele dado que aparece sempre idêntico em todos os documentos personalizados e que é fornecido para o documento através de um arquivo digital estático, para que não seja necessária a declaração do seu conteúdo várias vezes na base de dados. Na geração de muitos documentos, principalmente aqueles que tiverem uma grande quantidade de dados estáticos, utilizar o conteúdo estático separadamente da base de dados será fundamental para tornar a geração das diferentes versões de documentos com **VDP** mais simples. Um exemplo de conteúdo estático são os dados referentes a uma empresa (como seu nome, endereço, telefone e site), os quais serão idênticos em todas as várias versões personalizadas de um mesmo documento.

Já o **conteúdo variável** é aquele dado que aparece diferentemente em cada documento criado e é informado através de uma base de dados que permitirá essas alterações. Trabalhar com o conteúdo variável aumenta consideralmente o número de dados que devem ser alterados nos documentos que utilizam **VDP** em relação ao número utilizado para impressões e publicações estáticas tradicionais, onde o conteúdo estático da página sofre apenas pequenas alterações.

2.2.4 *Layout*

Um ***layout*** é a representação da disposição dos dados dentro do documento, o qual pode ser criado utilizando um *software* comum para publicações e criações ou até mesmo por um processador de texto. Porém, durante o seu projeto devem ser analisadas as possíveis soluções para melhor acomodar os conteúdos avançados como os textos, imagens e dados variáveis.

Esse *layout* que será criado pode ser **único**, que será utilizado em todos os documentos só alterando os dados extraídos da base de dados, como, por exemplo, quando se quer criar um folheto de propaganda para uma loja, a quantidade e a disposição dos produtos será igual para todos os usuários só alterando as imagens e textos. Além disso, esse poderá ser **diferenciado**,

quando são criados vários *layouts* que irão variar de documento a documento conforme as especificações e níveis de personalização que se quer atingir quando, por exemplo, deseja-se criar um folheto de propaganda variando o número de produtos para cada cliente, onde para atender essa maior quantidade de *copy holes* será necessária a geração de diferentes *layouts* que serão utilizados conforme o número de compras anteriores que o cliente fez, apresentando um maior número de produtos para os clientes que consomem mais.

É também através desse processo e dessa etapa que o criador do documento deve exercer uma maior criatividade, para torná-lo diferente e personalizado. Além disso, o *layout* diferenciado deve ser desenvolvido de uma forma que possa ser reutilizado em diferentes meios de saída, como por exemplo, um documento digital (página web) ser reutilizado em uma campanha de *marketing* impressa [Clinkunbroomer 2005].

2.2.5 Base de dados

Quando trabalha-se com dados variáveis o sucesso da criação de documentos deixa de ser exclusivamente centralizado na escolha de *layout*, papéis, tamanhos, cores, fontes e imagens únicas e passa a depender diretamente de uma **base de dados** que irá prover informações em diferentes aspectos tornando o processo personalizado.

Esses dados ou informações contidas nas bases podem ser dados simples como nomes, idades e endereços de clientes ou podem ser dados bem mais complexos como gráficos, imagens, *layout* ou informações específicas e personalizadas do usuário que se quer atingir com o documento. Esses podem ser utilizados em conjunto, o que torna mais específica a personalização do documento final, como por exemplo, criar uma propaganda dirigida a pessoas mais idosas, dando importância maior para o campo idade, com *layout* e dados mais tradicionais ou para pessoas mais jovens com *layouts* criativos, imagens e textos coloridos.

Os dados simples ou complexos, para serem utilizados em um documento com **VDP**, são extraídos diretamente da base de dados com a ajuda de ferramentas específicas de acordo com regras de negócio pré-definidas conforme os objetivos que se quer atingir. Com base nessas regras, o desenvolvedor do documento decide quais campos da base de dados são mais relevantes do que os outros [Smith 2005].

Com isso, as informações variáveis podem ser diretamente inseridas em qualquer tipo de documento. Porém o lugar em que as mesmas devem ser inseridas normalmente gera mudanças complexas na disposição do documento, em especial diferentes posicionamentos em relação aos demais componentes.

Além disso, a base de dados pode apresentar problemas como a falta de dados em campos sobre o cliente, o que acarretará a geração de um documento inconsistente, visto que os *copy holes* que forem associados a esses campos da base de dados não apresentarão textos ou imagens. Outro problema que deve ser cuidadosamente analisado durante a criação da base de dados é o fato de que os dados podem estar errados ou trocados entre os clientes, o que geraria documentos incompatíveis com o que deseja-se atingir e perdendo principalmente o objetivo de se utilizar **VDP**, já que o seu propósito e a sua vantagem é mostrar ao cliente que conhecemos, através da

personalização, as suas preferências.

Para resolver esses problemas deve-se, antes de associar os campos da base de dados aos *copy holes*, fazer a utilização de ferramentas que trabalhem com mescla, mineração, limpeza e prospecção nas bases de dados, as quais deverão ser capazes de eleger, extrair, transformar e carregar apenas os dados mais relevantes ao documento que deseja-se controlar para que se tenha a segurança nos dados extraídos e nas regras determinadas, verificando se os mesmos estão completos e consistentes.

2.2.6 Versões de documentos e dispositivos de saída

O processo de gerar as **versões de documentos**, trabalhando dentro do *workflow* de **VDP**, consiste na etapa onde são criados os vários documentos com os diferentes dados variáveis personalizados para cada usuário, os quais serão repassados a um **dispositivo de saída**.

Durante a geração das versões de documentos, as informações e *layout* personalizados podem mudar de saída a saída ou de folha em folha e esse processo permite que as diferentes alterações sejam adaptadas para um dispositivo de saída, o qual pode ser uma impressora ou um documento digital para publicações.

Atualmente, a fase da visualização para os dispositivos de saída é bem mais avançada para o meio impresso, e por esse motivo as empresas vêm desenvolvendo, durante os últimos anos, impressoras que facilitem esse processo e suportem através de seu *software*, conteúdo variável, através de impressoras como a NexPress da Kodak, DocuColor iGen3 da Xerox, HP Indigo da Hewlett-Packard e a DCP Press da Xeikon. Essas são os quatro melhores tipos de dispositivo de saída para a impressão com dados variáveis existentes no mercado.

2.3 Futuro dos dados variáveis

A tecnologia de dados variáveis, nos dias atuais, não é extensivamente utilizada pelo fato de que a maioria das empresas e desenvolvedores da comunidade de artes gráficas ainda não adaptou seus negócios às características e necessidades vindas dessa tecnologia [Kita 2003].

Segundo [Kita 2003], somente 3% dos designers gráficos e de outros profissionais vêm atualmente utilizando em seus trabalhos alguma forma de conteúdo variável para *marketing* ou vendas. Porém acredita-se que o futuro do **VDP** e o seu crescimento está diretamente ligado ao crescimento das compras pela internet, onde cada vez mais existirá uma necessidade de agradar o cliente personalizando seus documentos.

Esse aumento fará parte dos aspectos que ajudam na conscientização de que é cada vez mais necessária a utilização da tecnologia visto que essa possui grandes benefícios, como o aumento das taxas de resposta sobre determinados produtos e também da lealdade do cliente com a empresa, pois isso reflete como se a empresa soubesse detalhadamente as necessidades de seu usuário tratando-o com o valor merecido, trazendo identidade à empresa através de seus documentos personalizados, visto que o usuário ao ler esses documentos percebe que a empresa realmente

quer tê-lo como cliente e conhece suas características [Seybold Publications 1998].

Além disso, utilizar **VDP** pode ser importante conforme pesquisas realizadas que comprovam que uma documentação personalizada aumenta muito as taxas de resposta e aproveita melhor os gastos feitos com os clientes, incentivando as relações em direção as vendas repetitivas ou fidelização e por esse motivo há um melhor e maior retorno dos investimentos com as campanhas ou documentos gerados com dados variáveis [EFI do Brasil 2005].

Para impulsionar o seu futuro o ***Comitê de Padrões Técnicos de Artes Gráficas***, ou ***Committee for Graphic Arts Technical Standards*** (CGATS) tem agora a responsabilidade de desenvolver o avanço progressivo através de padrões ou linguagens menos limitadas como as atuais (como ***Personalized Print Markup Language***, ou **PPML** e ***Personalized Print Markup Language/Variable Printing Data Exchange***, ou **PPML/VDX**) que facilite a utilização dos dados variáveis na prática. Para isso, tem idealizado para o futuro o desenvolvimento de linguagens menos restritas e de tarefas que sustentarão o padrão e a indústria de publicação e impressão de dados variáveis [MacDowell 2004].

Para [Monkerud 2003], o crescimento da tecnologia depende diretamente de fabricantes, dos colaboradores do *software*, dos pesquisadores em bases de dados e a outros, os quais devem gerar melhoras significativas e desenvolver uma infraestrutura para solucionar muitas das dificuldades que necessitam ser superadas antes que o **VDP** possa tornar-se uma técnica difundida.

Aprimorar principalmente a geração de *software* dará um grande impulso na sua utilização, visto que atualmente existem poucos que permitem trabalhar com **VDP**, os quais na maioria das vezes necessitam de pessoas especializadas e voltadas para programação e administração de banco de dados para implementar uma solução de **VDP** [Miley 2003].

Exclusivamente, o futuro do **VDP** depende de que três aspectos sejam explorados: fazer com que os usuários e as empresas tornem-se cientes do potencial da tecnologia e das maneiras de explorá-la; convencer comerciantes sobre o maior retorno no investimento; fazer com que as empresas possuam dados ou bases de dados de seus clientes em um formato utilizável e que seja capaz de aumentar, através de pesquisas, a capacidade desses dados.

Com esses avanços a sua utilização deixará de ser exclusivamente das grandes empresas e de desenvolvedores e passará a ser feita pelas companhias menores, quando o processo de *workflow* será bem menos custoso, fazendo com que assim aumente a competição entre pequenas e grandes gráficas. Esse fato ocorrerá porque, com a nova área de *marketing* personalizado, as grandes tiragens em escala não serão mais uma boa estratégia de comunicação e as pequenas gráficas podem se diferenciar com vantagens sobre o conteúdo personalizado em pequenas tiragens.

3 Estruturação de Documentos associada a Dados Variáveis

Pelo fato de existir grande quantidade de documentos personalizados que serão transmitidos através de diferentes meios de saída, é necessária uma forma que permita modelar o relacionamento entre as partes desse documento, garantindo transformações nos seus conteúdos de forma ágil e sem que se perca a sua consistência lógica. A maneira mais eficaz existente na bibliografia para atender esse relacionamento e garantir a fidelidade aos documentos denomina-se **Estruturação de Documentos**.

Essa tecnologia, que será explorada neste capítulo em conjunto como os dados variáveis terá a finalidade de, futuramente, criar um modelo que gere documentos personalizados onde sejam preservadas as características mais relevantes dentro do contexto de transmissão de informação e de consistência lógica para diferentes meios de saída ou para diferentes usuários.

3.1 Conceito de estruturação de documentos

Define-se basicamente um documento não estruturado como um conjunto de caracteres em seqüência, modificando sua posição na seqüência sempre que ocorrer alguma alteração no documento. Já um documento estruturado, ou seja, que passa pela etapa de **Estruturação de Documentos** ou ***Document Model Representation***, é definido como um documento de mais alto nível que irá possuir relacionamentos entre as partes que o compõem, uma descrição lógica e componentes adicionais que descrevem e representam os conteúdos estáticos e variáveis contidos nas partes do documento perante a uma representação hierárquica e ao término dessa etapa será obtido um documento estruturado hierarquicamente [Groves e Brailsford 1993].

Considera-se um documento estruturado como de alto nível porque em vez de ser baseado na representação física e estética, ele passa a ser baseado na sua organização e formação lógica, sendo composto por elementos também de alto nível chamados assim por estarem contidos dentro de um documento estruturado. Um exemplo bastante utilizado de documento de alto nível pode ser visto em qualquer documento nas linguagens LaTeX, XML, HTML ou RTF.

Utilizar estruturação em documentos que trabalhem com conteúdo variável traz como principal vantagem o fato de permitir a criação e manipulação de conteúdos executando alterações e personalizações em documentos e nas suas características de herança, os quais abordados em conjunto com as definições de dados variáveis, poderão suprir a necessidade de gerar documentos personalizados através da criação de um modelo, *software* ou linguagem que torne mais fácil a sua existência na prática, executando alterações auxiliares em documentos e nas suas características. Além disso, existem outras vantagens que agilizam o processo de estruturação como a capacidade de se reutilizar conteúdos variáveis ou estáticos das partes que compõem o documento.

Para que seja possível gerar um documento estruturado com uma consistência lógica aceitável é necessário que seja determinado o seu tipo de representação física (conforme será visto na Seção 3.2) e que o mesmo seja composto por componentes que auxiliarão e representarão os dados dentro de uma representação física (conforme será mostrado no decorrer da Seção 3.3).

3.2 Estruturas de representação para um documento estruturado

Antes que se comece a etapa de estruturação de um documento e que seus componentes sejam definidos é necessário que se defina a melhor maneira de representá-lo fisicamente através de uma estrutura de dados.

Segundo [André, Furuta e Quint 1989] utilizar representações através de estruturas de dados faz com que se tenha uma melhor visualização do documento estruturado de uma forma fiel ao que o seu desenvolvedor deseja e utilizando abstração para representá-los, onde todos os conteúdos estáticos e variáveis são enumerados, estabelecendo a ordem entre eles para se chegar a uma hierarquia que traduz a estrutura do documento. As estruturas de dados que são utilizadas na bibliografia para suprir esses aspectos necessários são **árvores**, **grafos dirigidos**, **unidirecionais** ou **acíclicos dirigidos (DAG)**.

Dentre essas, um tipo de representação bastante utilizada é a estrutura de **grafos acíclicos dirigidos**, porque permite que sejam reaproveitadas partes do documento. Assim, esse tipo de representação torna-se a maneira mais simples de mostrar compartilhamento entre as partes do documento estruturado.

Além dos grafos acíclicos dirigidos, a representação mais utilizada para atender as necessidades desse tipo de estrutura são as **árvores**, por apresentarem a capacidade de hierarquia e herança entre os conteúdos e fazer com que o documento seja dividido em partes, que serão subdivididas em novas partes e assim sucessivamente. Esse tipo de representação permite a criação de várias árvores, o que ajudará na relação entre os conteúdos que formam o documento.

Trabalhar com esse tipo de estrutura de dados em modelos de documentos, permitindo herança, traz como vantagem o fato de não ser necessário incluir todos os tipos de diretrizes em todas as partes que formam o documento. Uma vez definidos os conteúdos a serem inseridos e essas diretrizes (cores, estilo, tamanho e tipo de fonte) do documento, através do mecanismo de herança eles propagam-se por toda a estrutura facilitando a sua formatação de uma forma rápida e simples. Como exemplo disso, todas as partes de um documento indicadas como sendo títulos, através de uma estrutura implícita, recebem diretrizes que determinarão que essas partes sejam em fonte *bold* e tamanho 14 pt.

Para esses tipos de estruturas de dados (tanto para um grafo acíclico dirigido quanto para uma árvore) a varredura e a busca dos conteúdos pelo estrutura ocorrerá, para melhor controlar a hierarquia, de forma *top-down*. Essa forma de percorrer a árvore ou grafo irá começar a criar o documento final baseado nos primeiros conteúdos ou partes que o compõem, passando pelas relações entre os conteúdos estáticos e variáveis filhos, para após isso incluir as propriedades estéticas de um documento estruturado, as quais determinam a aparência final, como por exemplo, diretrizes de posicionamento dos dados no *layout*, cores, tamanho e estilos da fonte [Groves

e Brailsford 1993].

Quando utiliza-se essas estruturas de dados é necessário também que se estabeleça o tipo de relação entre as partes ou dados (estáticos ou variáveis) e por esse motivo esses componentes podem ser classificados como dois tipos:

- **homogênea:** ocorre quando todos os conteúdos ou partes que formam um documento utilizam o mesmo tipo de relacionamento dentro do espaço de informação, como por exemplo, uma página HTML que possui *links* apenas para documentos HTML;
- **heterogênea:** ocorre quando os diferentes dados ou partes que compõem o documento estruturado se relacionam ou são descritos em diferentes espaços de informações, como por exemplo, uma página HTML com *links* para documentos com um formato distinto de HTML, como um arquivo PDF.

3.3 Componentes da estruturação de documentos

Um documento estruturado é formado por dois componentes principais denominados **estrutura lógica** e **objetos** (ou conjunto de objetos).

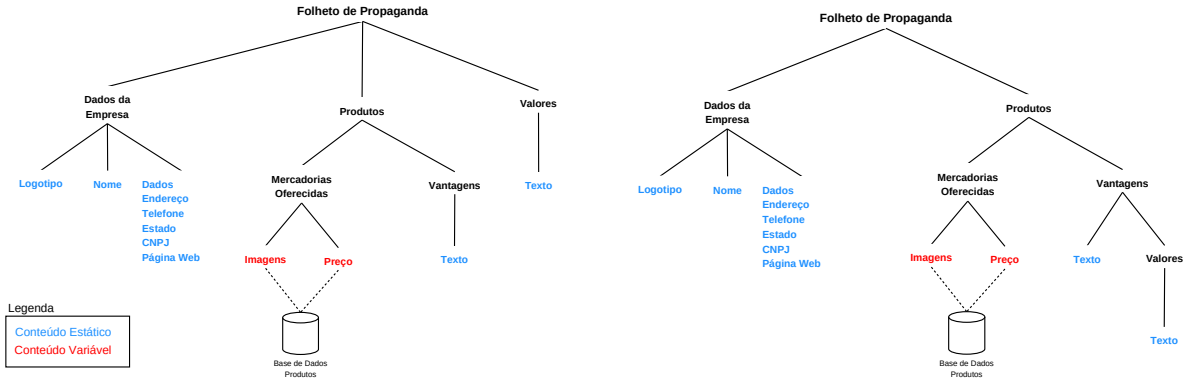
Essa estrutura lógica, conforme será mostrado na Subseção 3.3.1, visa dividí-lo em partes, onde um artigo, por exemplo, seria particionado em seções e subseções. Compondo essa estrutura lógica está presente uma estrutura primária, que representa as partes do documento, e as estruturas secundárias, as quais associam as mesmas e seus relacionamentos que não podem ser representados na primeira estrutura.

3.3.1 Estrutura lógica

Um documento estruturado possui uma **estrutura lógica**, cujo seu principal objetivo é dar significado e relacionamento entre os conteúdos (estáticos ou variáveis) e as partes em que o documento estruturado for dividido.

Pode-se dizer que a estrutura lógica de um folheto de propaganda, por exemplo, é composta por várias partes (Dados da Empresa, Mercadorias Oferecidas, Vantagens e Modo de Pagamento) onde cada uma dessas partes é subdividida em subseções ou itens, compostas por conteúdos estáticos ou variáveis que formam a estrutura lógica do documento.

O conceito e a forma de representação da estrutura lógica podem variar em relação às diferentes pessoas que vão estruturar e utilizar os documentos. Um exemplo pode ser visto quando um primeiro desenvolvedor analisa o projeto e as regras de negócio que darão as diretrizes para um folheto de propaganda e o estrutura através de três partes principais denominadas **Dados da Empresa**, **Produtos** com duas seções chamadas de **Mercadorias Oferecidas**, a qual é dividida em subseções para cada classificação das Mercadorias, e **Vantagens**, além da parte final do documento variável denominado **Valores**. Porém, um segundo desenvolvedor analisa as mesmas regras de negócio e projeto, mas cria outro tipo de estrutura lógica para o mesmo documento, passando a seção **Valores** para uma subseção de **Vantagens** (Figura 3).



(a) Desenvolvedor 1 (b) Desenvolvedor 2

Figura 3 – Diferentes estruturações para um mesmo documento por diferentes desenvolvedores.

Nesse tipo de exemplo as estruturas modificadas geram pequenas alterações na hierarquia e poucos conflitos durante a estruturação do documento, já que o cliente final continuará entendendo da mesma forma o contexto completo do folheto.

A estrutura lógica de um documento estruturado é primeiramente composta por uma estrutura inicial, denominada **Estrutura Primária**. Essa possui como principal objetivo descrever as ligações e relacionamentos usados para representar dados (textos, imagens e gráficos) que formam a estrutura lógica do documento (Figura 4).

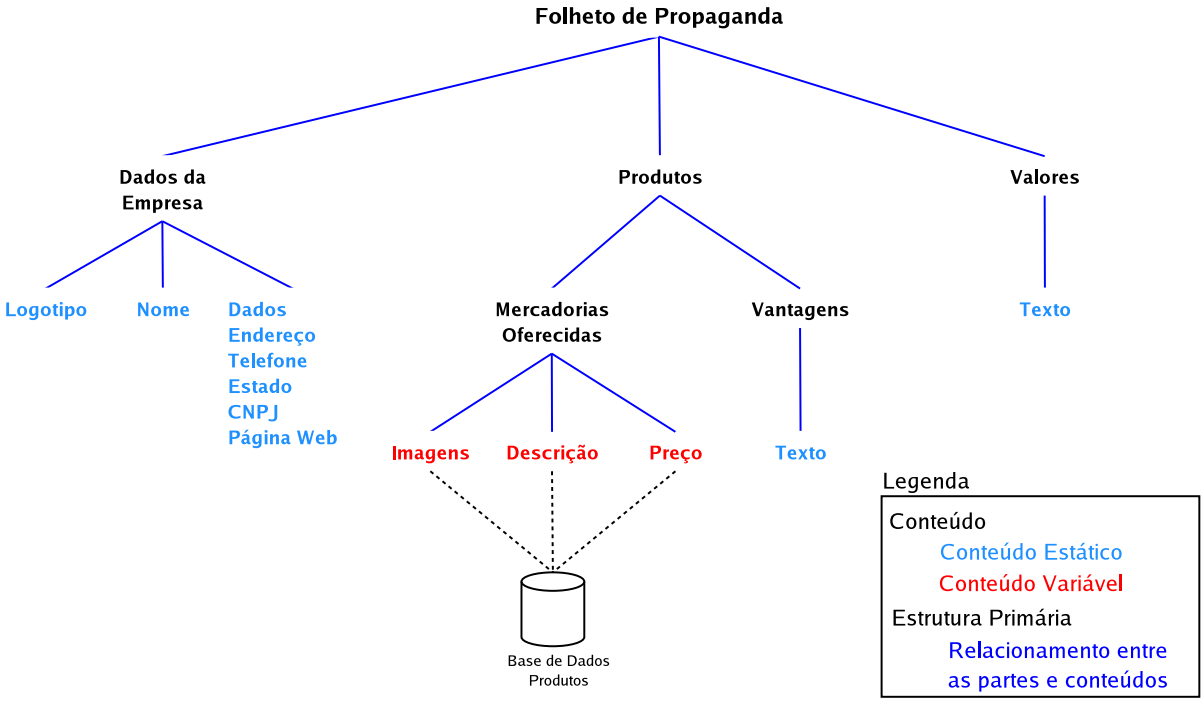


Figura 4 – Componentes da Estrutura Primária.

Segundo [Heeman 1992], a composição da estrutura primária só torna-se possível através da utilização separada ou combinada de cinco tipos de estruturas:

1. **Seqüência:** é uma coleção de conteúdos estáticos ou variáveis requisitados em seqüência,

os quais pertencem a tipos distintos. Um exemplo desse tipo de estrutura seria de um documento onde as imagens devem sempre ser apresentadas logo após um texto;

2. **Agregado:** trata-se de uma coleção de conteúdos estáticos ou variáveis classificados como objetos desordenados que podem pertencer a tipos diferentes. Um exemplo desse tipo de construção são os dados variáveis que devem ser apresentados no documento final logo após o texto;

3. **Lista:** dados são classificados como listas quando são requeridos em seqüência e são do mesmo tipo. A principal diferença existente entre esse tipo de estrutura e seqüência está em relação aos tipos que os dados possuem, se forem de mesmo tipo será uma lista, porém se os mesmos forem de tipos distintos será uma seqüência;

4. **Escolha:** trata-se dos dados associados a uma lista de tipos pré-determinados;

5. **Combinação:** tem por objetivo, dentro da estrutura primária, tornar esses quatro tipos de estruturas mais eficazes e com capacidade de atender todas as necessidades que possam surgir durante a construção da árvore lógica do documento. Possui como característica permitir que as construções citadas acima sejam utilizadas em conjunto.

É também nesse componente de estruturação que o documento, em vez de uma única parte, passa a ter várias partes que representam os dados (textos, imagens, figuras, gráficos e tabelas), as quais precisam estar relacionadas entre si para garantir a consistência lógica. Para facilitar essa consistência, permitir o reaproveitamento das partes e determinar se essas podem ou não serem relacionadas através da estrutura primária existem dois níveis de agrupamento classificados como:

- **Genéricos:** onde são agrupados documentos ou as partes dos documentos que tiverem estruturas similares, como letras, fontes e imagens (para agrupar partes de um documento) ou tipo de meios de saída como relatórios e manuais (para agrupar documentos);

- **Específicos:** quando partes ou o próprio documento não forem similares a nenhuma outra estrutura.

A segunda estrutura que compõe a estrutura lógica de um documento, denominada **Estrutura Secundária**, tem por objetivo conter os relacionamentos adicionais entres os dados variáveis e estáticos, como fontes, imagens que podem mudar de posição e referenciamento entre diferentes documentos. Esses relacionamentos da estrutura secundária não têm a capacidade de influenciar a formação de um documento de alto nível.

Conforme pode ser visto na Figura 5, a definição das estruturas secundárias é composta por três componentes adicionais que são utilizados para sua construção e composição [Furuta, Quint e André 1988]:

- **Atributos:** a principal função dos atributos é armazenar informações que definirão padrões (como nomes, tamanhos, estilos, cores da fonte, tipo de alinhamento e idioma) na forma de informações semânticas, as quais não podem ser representadas dentro da estrutura primária.

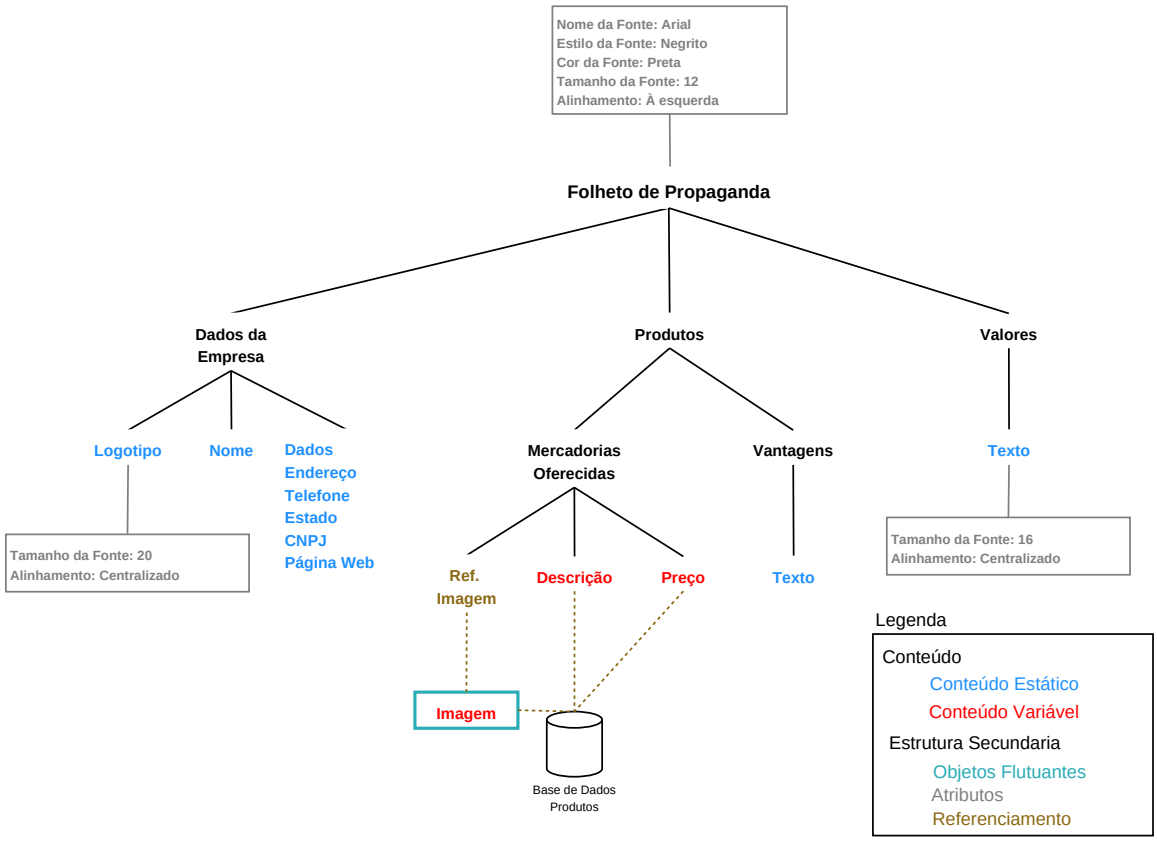


Figura 5 – Componentes da Estrutura Secundária.

Visto que os conteúdos estáticos devem aparecer sempre iguais em todos os documentos, os atributos associados aos dados estáticos devem ser diretamente definidos no nodo da árvore que representa a imagem, objeto, texto ou gráfico que será inserido no documento e uma vez definidos não podem ser alterados para que se mantenham iguais em todos os documentos gerados. Já quando se trabalha com conteúdos variáveis os atributos poderão se alterar a cada dado a ser inserido no documento personalizado, trazendo à árvore estrutural uma grande quantidade de atributos, os quais podem ser descritos diretamente em uma base de dados de atributos variáveis onde é determinado o campo onde se deseja armazenar essas características. Isso tornará cada documento mais personalizado, com diferentes fontes, cores, estilos e tamanhos, já que o mesmo não precisa ficar preso a padrões que foram estabelecidos diretamente na estrutura fixa do *layout* ou nos seus nodos fixos.

Representar um documento estruturado através da hierarquia faz com que se trabalhe com a propriedade de herança, onde os atributos associados a um nó pai da árvore fazem com que seus nodos filhos, caso não seja estabelecido novo padrão, carreguem os mesmos atributos. Herdar atributos pode ocorrer também através dos atributos que são capazes de utilizar os tipos e valores de objetos de outros documentos ou através de outro atributo pertencente ao mesmo documento. Um exemplo disso, seria estipular que todos os títulos que compõem o documento devem possuir atributos iguais ao do título inicial. Então durante a estruturação da árvore novos atributos não devem ser associados a títulos seguintes, porque esses herdam todos os atributos do título principal.

- **Elementos flutuantes:** caracterizam-se como elemento flutuante figuras, notas de rodapé ou outros conteúdos variáveis e estáticos que não precisam necessariamente ter uma posição fixa dentro dos documentos gerados.

  Utilizar documentos estruturados para diferentes dispositivos ou meios de saída torna-se mais fácil, através desses elementos flutuantes, pois permite que conteúdos sofram transformações em suas disposições até encontrar uma posição arbitrária no documento conforme as exigências estabelecidas nas regras de negócio.

  Além disso, quando se trabalha com dados variáveis classificados como elementos flutuantes, como por exemplo as figuras, pode ocorrer de uma figura ser mais larga ou mais alta do que as demais. Tratando-as como elementos flutuantes elas podem percorrer o documento até encontrar uma disposição no *layout* esteticamente aceitável para o usuário.

- **Referências:** tem a finalidade de ligar os conteúdos e variáveis que compõem a árvore estruturada com os demais, formando a construção final, lógica e hierárquica do documento estruturado. Além disso, fazem a ligação entre os nodos da árvore e as bases de dados de onde são extraídos os dados variáveis.

  Existem duas principais maneiras de se referenciar conteúdos dentro de uma árvore estruturada. A primeira acontece quando as referências entre os dados estáticos ou variáveis ocorrem dentro de um mesmo documento, a qual é bastante utilizada em documentos personalizados que fazem referências no texto, como por exemplo, uma determinada figura ou nota de rodapé. Já a outra forma de se referenciar ocorre quando as referências entre os dados aparecem entre diferentes documentos, a qual cresceu expressivamente com o surgimento da Web, através dos hipertextos.

3.3.2 Objetos

O último componente existente para auxiliar o processo de estruturação de documentos denomina-se **Objetos**, definidos como a forma mais simples de se estruturar certos tipos de conteúdos estáticos e variáveis, tornando tanto a estrutura primária (ou seja, o relacionamento entre as partes) quanto a própria definição e estruturação dos dados relativamente simples, bem compreendida e de fácil manipulação através de árvores.

Conteúdos que são melhores estruturados em formato de objetos podem ser tabelas, diagramas, fórmulas, gráficos ou dados com alto grau de complexidade de representação, os quais podem variar a cada documento (vindos de uma base de dados) ou serem os mesmos em todos os documentos.

Os objetos são estruturados separadamente da estrutura lógica do documento principal, ou seja, possuem uma estrutura de dados para a representação própria e a estrutura principal possui uma chamada para inserir esses objetos quando a estrutura for percorrida.

O fato de tratá-los e estruturá-los separadamente traz como principal vantagem reutilizar-se os conteúdos estruturados como objetos em diversos documentos, principalmente se esse objeto for estático. Nesse caso se os dados forem estruturados como objetos ocorrerá a diminuição do

tempo para gerar os diferentes documentos finais, visto que a reutilização agilizará esse processo estruturando de uma única vez as árvores desses conteúdos complexos e utilizando-as em vários documentos ao contrário de quando esses conteúdos fazem parte da própria árvore principal do documento [Heeman 1992].

Com isso, partes que formam a árvore desses objetos podem ser utilizadas em documentos distintos. Um exemplo desse processo de reutilização seria de uma base de dados com 77 tabelas que variam a cada documento gerado, porém 50 dessas tabelas são idênticas para um grupo de usuários, 20 para outro e o restante diferentes para cada um dos usuários. Em vez de ocorrer a estruturação de 77 tabelas diretamente como um nodo da árvore principal, o que elevaria o tempo de geração desses documentos personalizados, utiliza-se a estruturação dessas tabelas através dos objetos, as quais serão reutilizadas nos documentos através do encapsulamento na árvore.

Essas árvores distintas dos objetos possuem estruturas e três principais componentes que ajudarão na sua estruturação, conforme mostrado na Figura 6, e são classificados como [Groves e Brailsford 1993]:

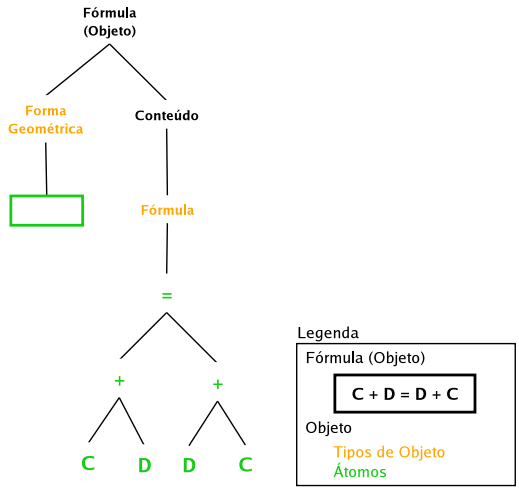


Figura 6 – Componentes dos Objetos.

- **Átomos:** são conteúdos ou dados concretos que são reconhecidos dentro de um objeto estruturado, os quais podem ser caracteres soltos, textos, círculos, linhas, colunas ou diferentes figuras geométricas, e que na maioria das vezes costumam ser relativamente grandes.

  Como mostrado na bibliografia e nas aplicações que utilizam objetos, os átomos costumam ser relativamente grandes e heterogêneos em relação ao tipo de estrutura de dados utilizadas [Furuta 1994].

- **Conteúdo Básico:** responsável pelo agrupamento dos objetos similares como caracteres ou gráficos.

- **Tipo de Objetos:** entidade não concreta que apenas representa os componentes agrupados e construídos a partir do conteúdo básico e de outros objetos utilizados na estrutura interna. Esses tipos de objetos são classificados como:

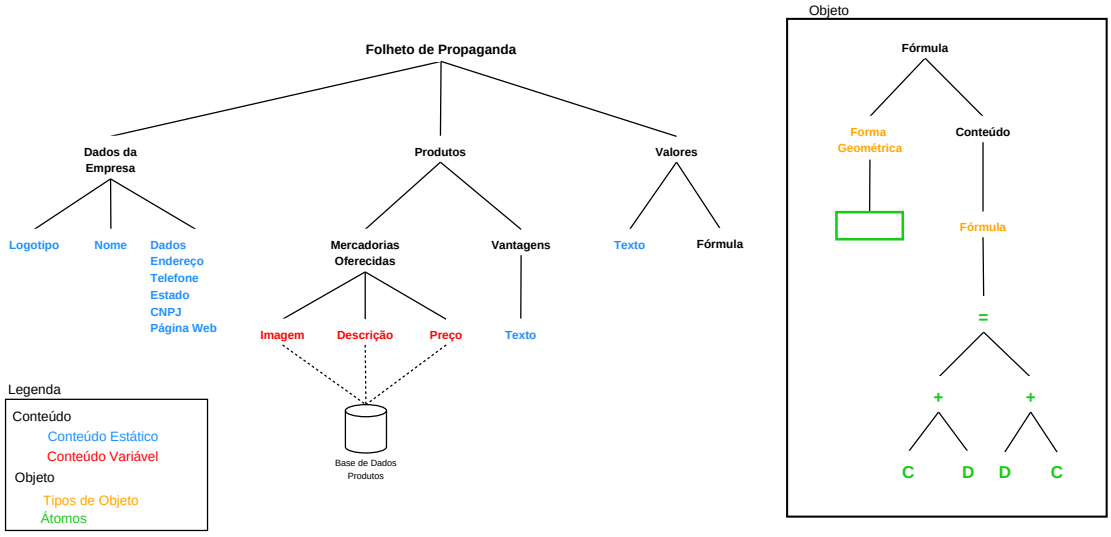


Figura 7 – Estrutura principal pedindo encapsulamento e objeto que será encapsulado.

- **Fórmula:** consiste na hierarquia formada por um conjunto de caracteres;
- **Imagem:** consiste em uma matriz de pixels;
- **Diagrama:** consiste em gráficos estruturados dentro de objetos ou representações gráficas;
- **Tabela:** consiste em um conjunto de formas geométricas concatenadas e também caracteres para formar os textos;
- **Forma Geométrica:** consiste em dados que podem ser agrupados como outros gráficos para formar e dar mais liberdade aos objetos, para poder gerar conteúdos complexos.

Esses tipos de objetos podem ajudar um tipo a completar os demais. Como por exemplo, as tabelas ou diagramas são compostos por formas geométricas diferentes seguidos por textos, caracteres soltos que contenham imagens ou fórmulas (como a legenda das imagens).

O inserção dos objetos variáveis e estáticos na árvore lógica principal, processo o qual denomina-se **encapsulamento**, soluciona os problemas com conteúdos do tipo tabelas e diagramas, onde as árvores dos objetos são encapsulados em uma folha da árvore geral gerando novos filhos e tornando-se mais simples de se trabalhar, avaliar e reutilizar (Figuras 7 e 8).

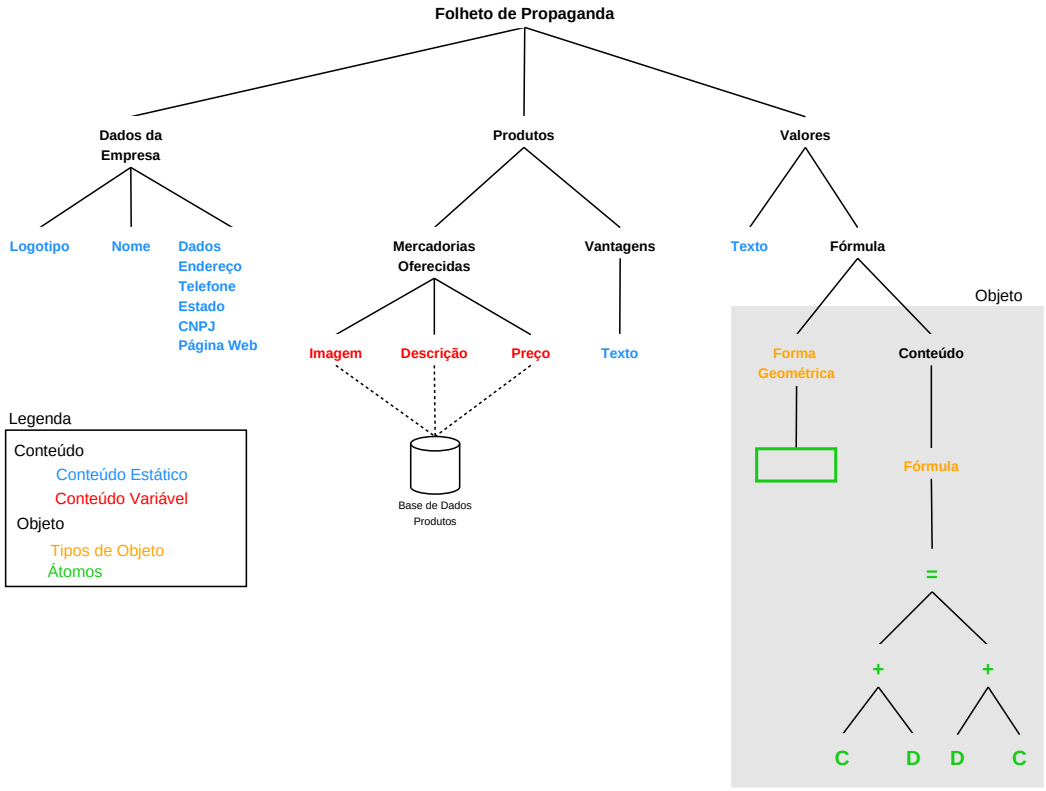


Figura 8 – Encapsulamento do objeto na estrutura principal.

4 Um modelo para o projeto de Documentos Personalizados com Conteúdo Variável

4.1 Proposta

Baseado nas áreas apresentadas nos Capítulos 2 e 3, foi proposto o desenvolvimento de uma abordagem mostrando como as técnicas estudadas (dados variáveis e estruturação de documentos) comportam-se trabalhando juntas, e a partir disso fossem avaliados os principais aspectos positivos na prática e para que alguns dos problemas encontrados sejam analisados e eventualmente solucionados.

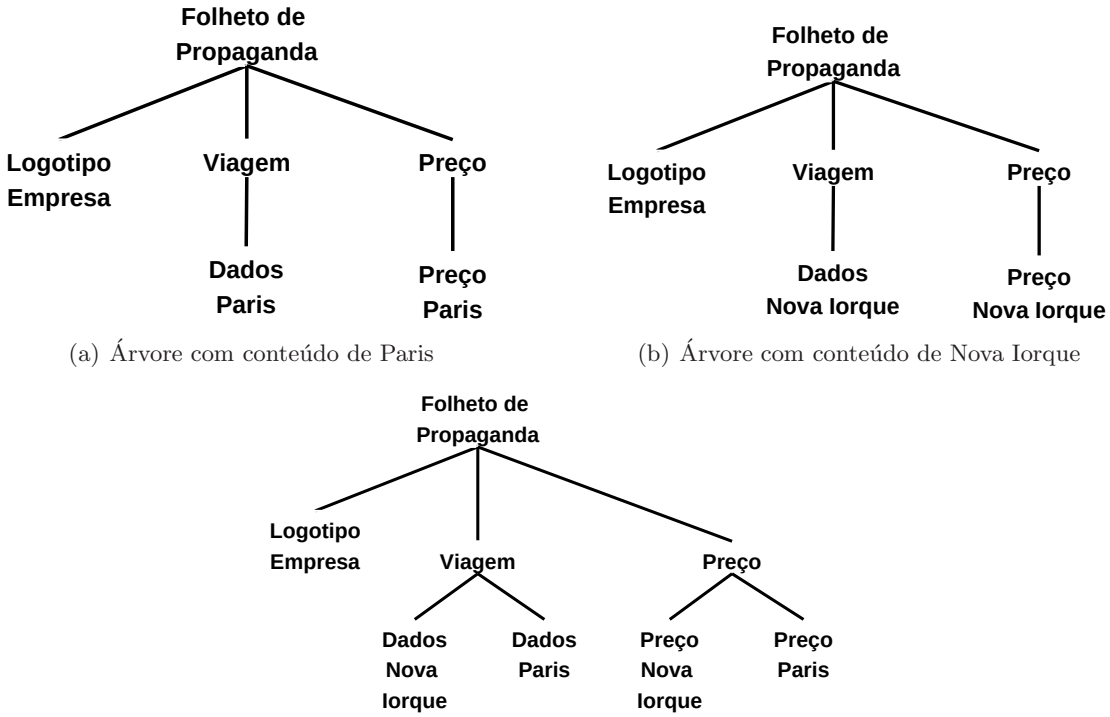
Durante a avaliação do funcionamento dessas técnicas em conjunto foi observado que para gerar documentos personalizados com conteúdos variáveis é necessária a criação de várias árvores estruturadas para atender os conteúdos disponíveis como opções (imagens, textos ou gráficos que diferem a cada documento) que serão utilizadas nos documentos personalizados finais.

Um exemplo desse problema pode ser visto quando se quer gerar um documento para um folheto de propaganda que oferece ofertas de viagens para destinos como Nova Iorque e Paris, onde para um grupo de usuários o documento será composto somente pelas informações de Paris, para outro, só os conteúdos de Nova Iorque e para um terceiro ambos os conteúdos irão ser inseridos nos documentos. Essas variações de conteúdo para cada grupo de usuários só ocorreriam se houvesse a criação de três árvores estruturadas (Figura 9) que contemplem essas diferentes opções de conteúdo, onde para cada um desses três grupos seria escolhida uma dessas árvores.

Gerar essas várias árvores para atender cada um dos diferentes conteúdos é um dos principais fatores que tornam a utilização dessas técnicas em conjunto bastante complexa e pouco difundida para ser utilizada na prática.

Para facilitar esse processo e evitar o controle de múltiplas origens dos dados (diferentes documentos estáticos, várias bases de dados e principalmente várias árvores) desejamos desenvolver e formalizar, baseado nessas duas técnicas, um **modelo para o projeto de documentos personalizados**, o qual traz como principal vantagem o fato de que em vez de criar diferentes árvores para cada documento que possuir opções de conteúdo, apenas estrutura-se e desenvolve-se uma única árvore desde o início. Essa possuirá em suas ramificações diferentes opções com conteúdos que alteram-se entre os documentos, que só serão percorridos e inseridos se os conteúdos que as compõem forem selecionados como preferência dos usuários.

Esse modelo será idealizado de forma que o seu funcionamento e formulação sejam pouco complexos e que com isso se consiga facilmente extrair os conteúdos relevantes para cada documento personalizado. Essa idealização também fará com que o mesmo seja utilizado por um profissional da área de documentos como projetistas, desenvolvedores ou designers de documentos.



(a) Árvore com conteúdo de Paris

(b) Árvore com conteúdo de Nova Iorque

(c) Árvore com conteúdo de Paris e Nova Iorque

Figura 9 – Formalização de múltiplas árvores para atender os diferentes conteúdos.

Propomos formalizar um modelo para o projeto de documentos personalizados, através de definições necessárias para o seu funcionamento, determinando as representações para a estruturação dos documentos (Seção 4.2), os modos de busca dentro dessa representação (Seção 4.3), além de conceitos e formalismos necessários para que o funcionamento desse modelo torne-se mais simples de ser utilizado em prática.

4.2 Estrutura de representação: árvores genéricas

Para o desenvolvimento do modelo, primeiramente é necessário definir como a estrutura lógica, os componentes e os objetos que formam os documentos a serem gerados vão ser representados dentro do formalismo. Por esse motivo e para que obtenha-se uma melhor visualização dos documentos a serem gerados foi determinado que dentre as estruturas disponíveis na literatura as **Árvores Genéricas** são as que melhor adaptam-se às características e necessidades que o modelo irá exigir e que terão o motivo da sua escolha explicado a seguir.

Uma característica que facilita a representação dos documentos nesse tipo de estrutura de dados será o fato de árvores genéricas possuírem um relacionamento hierárquico entre as partes que as compõem e por esse motivo suprem a necessidade que existe quando se trabalha com geração de documentos estruturados com dados variáveis, onde deve-se ter uma ordem entre as partes que garanta, perante o usuário final, a construção lógica de seus documentos. Além disso, as árvores genéricas têm a capacidade de representar grandes conjuntos de dados e conteúdos (uma necessidade dos casos reais de documentos personalizados) e alguns tipos de documentos

como por exemplo, a organização das páginas de um site, divisão de um livro em capítulos, seções, tópicos, etc.

Segundo [Moraes 2001], uma árvore genérica pode ser definida como uma árvore que possui um conjunto finito e não vazio de nodos que assumem as seguintes características (Figura 10):

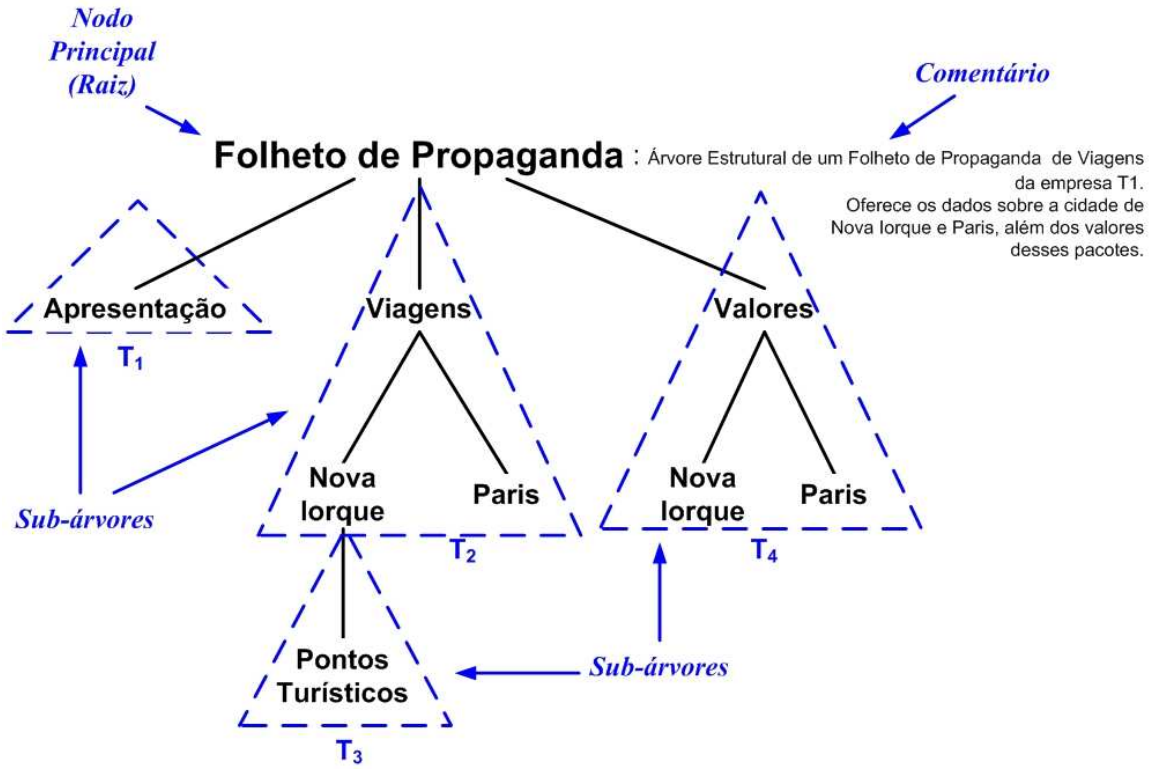


Figura 10 – Representação das características de uma árvore genérica.

- existe um nodo principal da árvore denominado **raiz**, que no modelo representará o tipo de documento a ser gerado (um poster, um e-mail, um folheto de propaganda ou de notícias, uma página web personalizada, um jornal ou revista, entre outros).

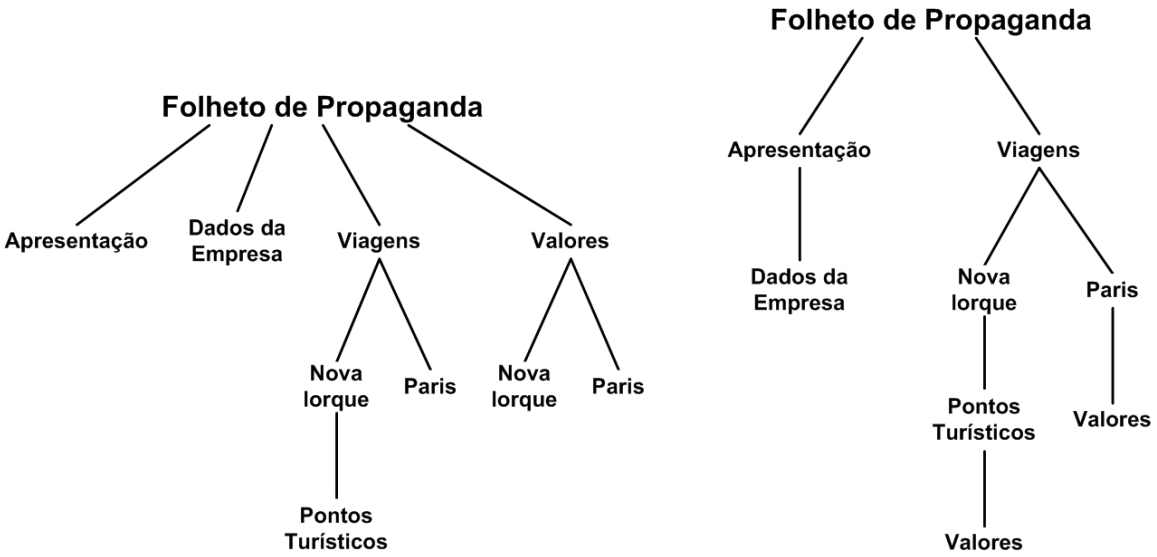
  Além disso, acompanhando o nodo raiz, o desenvolvedor pode adicionar **comentários** referentes à estrutura lógica do documento durante a sua estruturação, como por exemplo, os tipos de classificação dos conteúdos existentes e quais as opções foram atribuídas a esses conteúdos. Adicionar esses comentários ao nodo raiz fará com que eles funcionem como diretrizes para a decisão da escolha dos conteúdos para cada usuário e, com isso outro desenvolvedor pode utilizá-lo sem que tenha que analisar toda a sua estrutura para poder gerar diferentes versões dos documentos. Na Figura 10 pode ser visto um exemplo de comentário adicionado à raiz, onde são detalhados os conteúdos que compõem a árvore e as viagens que são oferecidas nesse folheto além dos valores a serem pagos;

- os demais nodos da árvore estão divididos em $T_1, T_2, ..., T_k$ estruturas ou sub-conjuntos disjuntos de árvores, denominado **sub-árvores**. Essas estruturas disjuntas irão garantir que um nodo não aparecerá em mais de uma sub-árvore.

  Nessas sub-árvores e em seus nodos descendentes estarão contidas e armazenadas as partes que compõem a estrutura do documento e os conteúdos (como textos, imagens, tabelas,

gráficos) que serão inseridos no mesmo, deixando-os personalizados conforme as preferências de cada cliente.

Um exemplo de construção de documentos pode ser visto na Figura 11, onde dois desenvolvedores constroem árvores estruturadas com posições distintas para as partes que compõem um mesmo documento, as quais perante o conceito de árvores genéricas são equivalentes em relação ao conteúdo e que portanto, apesar de não possuírem uma mesma ordem, apresentarão documentos ao usuário final que serão iguais em relação as suas partes e conteúdos sem perder a consistência lógica.



(a) Desenvolvedor 1 (b) Desenvolvedor 2

Figura 11 – Exemplo de um folheto de propaganda estruturado por dois desenvolvedores diferentes.

Outra vantagem dessa escolha de representação é que as árvores genéricas, ao contrário das árvores binárias, não estabelecem um limite de números de sub-árvores que cada nodo pode ter. Enquanto numa árvore binária o número de filhos de nodos é limitado a dois, nas árvores genéricas essa restrição não existe, podendo haver um número arbitrário de filhos associados a um nodo pai. Esse fato irá auxiliar o modelo, pois quando estrutura-se um documento sabe-se que o mesmo pode ser composto por várias partes.

4.3 Modos de travessia do modelo

Para utilizar as árvores genéricas que representam os documentos, deve-se determinar a maneira de visitar sistematicamente todos os nodos que irão compor essas árvores que representam a estrutura lógica dos documentos a serem gerados. Essa tarefa será denominada **Varredura da Árvore Genérica** e determina a ordem em que o modelo irá percorrer a árvore tanto em relação à disposição dos nodos em sua largura quanto em sua altura.

Entre os métodos de travessia existentes na literatura [Wirth 1999], o que melhor adapta-se ao funcionamento esperado pelo modelo para buscar os nodos em relação à sua disposição na

altura da árvore é a travessia de **pré-ordem**, uma especialização da busca em profundidade. Nesse tipo de travessia, o primeiro nodo visitado é o que encontra-se na raiz, seguido das sub-árvores desse nodo, ou seja, visitando cada nodo antes dos seus filhos. Após isso, são examinados os nodos dessa sub-árvore, até que essa não possua mais nodos e então o modelo irá percorrer a próxima sub-árvore existente na árvore genérica. Dessa forma garante-se que a árvore estrutural do documento seja percorrida da raiz para as folhas, ou seja, de forma ***top-down***, assim como sugere-se na literatura quando ocorre a formação de documentos que passaram pela etapa de estruturação [André, Furuta e Quint 1989].

Outro método de travessia que também precisa ser definido é em relação à disposição dos nodos na largura da árvore. Esse método está relacionado com o fato de que durante a estruturação desse documento, por um desenvolvedor, essas árvores deverão ser previamente **ordenadas** conforme deseja-se que o modelo faça a varredura de seus nodos (que representam os conteúdos).

Essa ordenação nos nodos da árvore determina que a ordem dos filhos é relevante não apenas baseando-se na importância da hierarquia, mas também na posição que cada um dos filhos de um nodo ocupa, identificando assim os filhos como um sendo o primeiro, o outro o segundo, o terceiro e assim por diante. Os filhos nessa árvore são ordenados, baseados na literatura [Szwarcfiter e Markenzon 2000], assumindo a busca da **esquerda para direita**, implicando diretamente na garantia da consistência da representação final do documento.

Os métodos de travessia que são utilizados para o funcionamento do modelo podem ser observados na Figura 12. Nesse caso, o modelo irá visitar primeiramente o nodo raiz e segue a busca dos conteúdos até formar o documento, primeiramente na sub-árvore T_1 (nodo ***Apresentação***), em seguida a sub-árvore T_2, visto que para o funcionamento do modelo a árvore deve ser lida da esquerda para direita, e assim sucessivamente até T_4, fazendo com que toda a árvore seja percorrida.

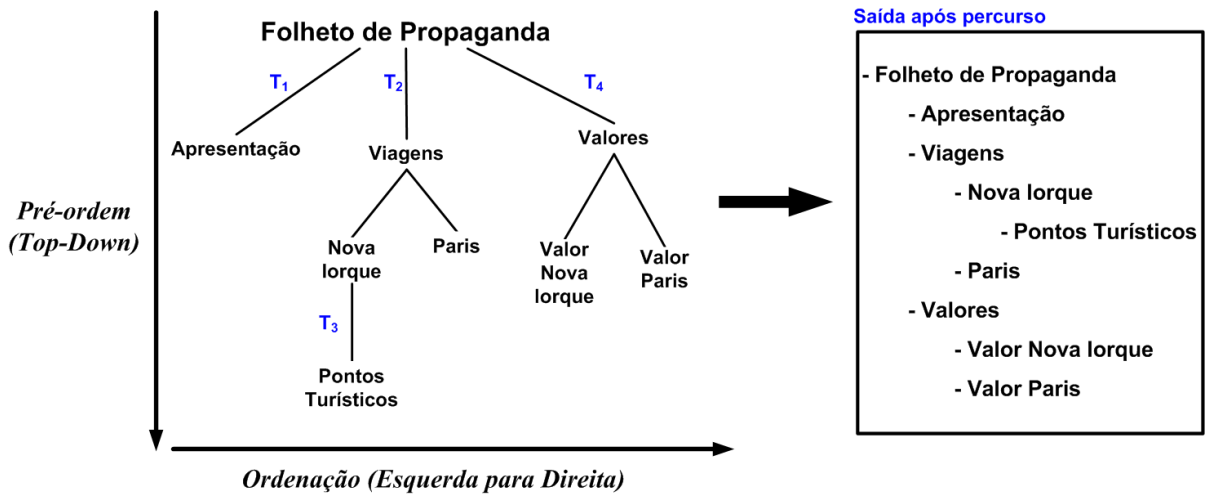


Figura 12 – Ordem de varredura de uma árvore genérica para o modelo formalizado.

O algoritmo em português estruturado desse método de varredura é mostrado no Pseudo-Código 1 e demonstra como a árvore genérica será percorrida. Nesse algoritmo a variável $T_{principal}$ representa a árvore principal, a variável Raiz representa o nodo raiz, primeiramente da árvore principal e após as chamadas recursivas os nodos raízes das sub-árvores e a variável T_k

representa as sub-árvores filhas do nodo Raiz.

Algoritmo 1: Algoritmo recursivo da travessia de **pré-ordem**.

```
1 preOrdem (T_principal, Raiz)
2 Visite Raiz
3 para cada filho T_k de Raiz faça
4 |   preOrdem(T_principal,T_k)
5 fim
```

4.4 Classificação dos nodos

Baseado em [Wirth 1999], os nodos que compõem as árvores genéricas além do nodo raiz são classificados em dois tipos. Se um nodo não apresentar descendentes, ou seja, não possuir sub-árvores, será classificado como **nodo terminal** ou **folha** da árvore. Já todo nodo que apresentar pelo menos um filho será classificado como **nodo interior** ou **nodo interno**.

Essas classificações podem ser analisadas no exemplo apresentado na Figura 13, onde a árvore estruturada possui um nodo principal raiz, denominado ***Folheto de Propaganda***, com quatro sub-árvores. Dentre essas sub-árvores os nodos ***Viagens*** e ***Valores*** são nodos internos por possuírem dois filhos, denominados ***Nova Iorque*** e ***Paris*** que, assim como o nodo ***Apresentação*** e ***Dados da Empresa***, são classificados como folhas por não apresentarem filhos.

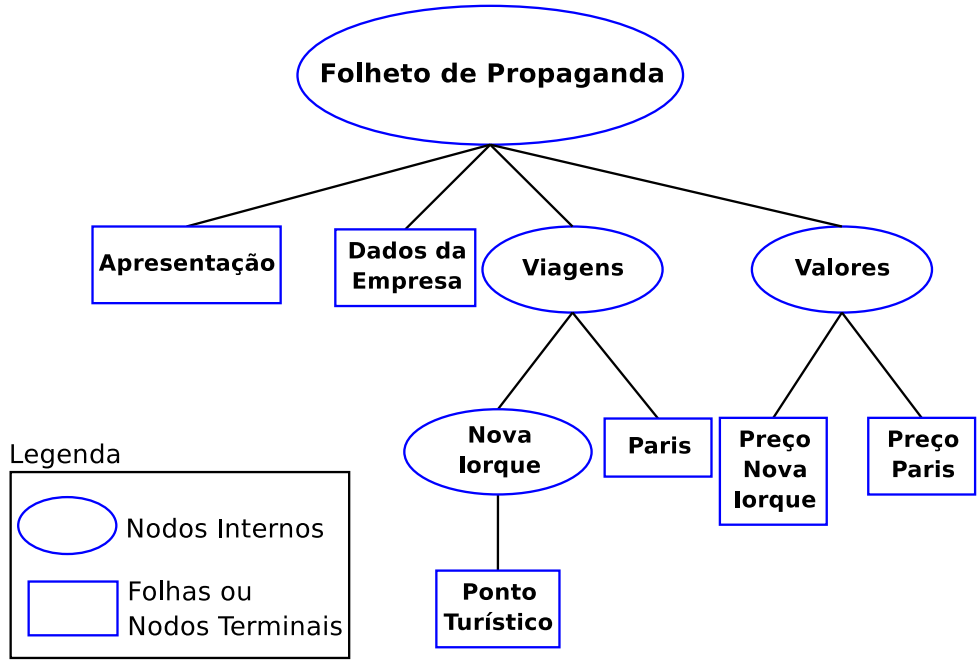


Figura 13 – Classificação dos nodos das árvores genéricas.

4.4.1 Nodo interno

Os nodos internos, nesse modelo, não possuem conteúdos (textos ou figuras) a serem incluídos no documento final e são usados apenas para a organização lógica, representando as partes do documento. Portanto, os nodos internos são entidades implícitas, como por exemplo, o nodo interno ***Viagem*** (Figura 13) que é uma representação sem conteúdos para as viagens que serão oferecidas no folheto de propaganda, no caso, Nova Iorque e Paris.

No modelo os nodos internos podem possuir vários filhos, que podem ser folhas ou outros nodos internos, os quais apresentam sempre ordem (existindo assim primeiro filho, segundo filho e assim sucessivamente) conforme apresentado na Seção 4.3.

4.4.2 Folhas

As folhas em uma árvore armazenam os conteúdos (textos, imagens ou gráficos) que são inseridos no documento final. Esses conteúdos, perante o modelo que está sendo proposto, podem ser classificados em quatro tipos distintos (conforme mostra a Figura 14) denominados folhas com conteúdo estático, variável, seletivo e seletivo/variável, os quais serão descritos nas seções a seguir.

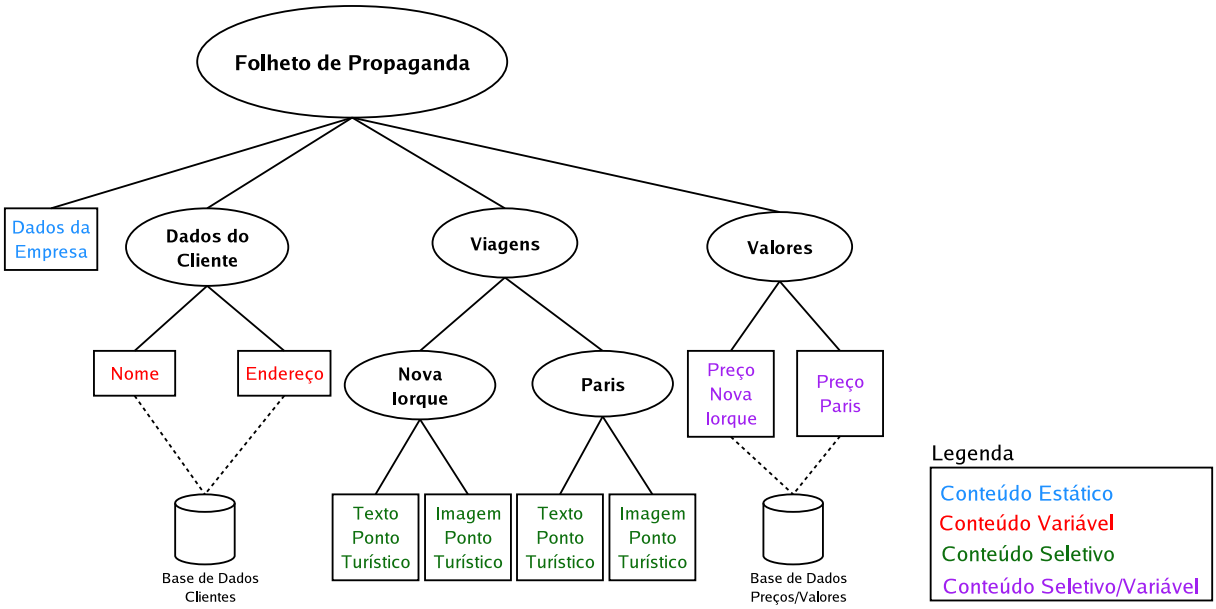


Figura 14 – Classificação dos conteúdos das folhas no modelo formalizado.

4.5 Conteúdo estático

As folhas associadas com o conteúdo estático, representadas na Figura 14 em azul claro, comportam-se como estruturas que nunca se alteram nos documentos, independente das prefe-

rências dos usuários e que portanto devem aparecer de forma idêntica em todas as versões finais dos documentos gerados para os diferentes usuários.

Os dados que compõem os conteúdos estáticos, ao contrário do apresentado no Capítulo 2, não precisam ser informados através de um arquivo estático externo e passam a compor a árvore genérica como uma folha, pois são dados constantes nas diferentes versões do documento final.

Alguns exemplos desse tipo de conteúdo são um logotipo ou os dados informativos de uma empresa (Figura 15) que sempre terá de ser inserido nos documentos para que o cliente reconheça qual empresa está enviando aquele documento e mantendo a identidade da mesma.

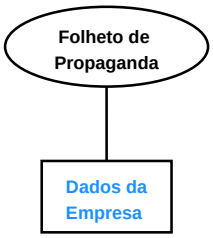


Figura 15 – Exemplo de conteúdo estático.

4.6 Conteúdo variável

As folhas que estão associadas com o conteúdo variável comportam-se como estruturas que se modificam a cada documento gerado (como por exemplo o nome, endereço, telefone de um cliente) e por esse motivo são extraídos diretamente de uma base de dados. Com isso, pode-se definir os conteúdos variáveis como uma das partes do documento que modificam-se a cada documento personalizado.

Os dados que compõem os conteúdos variáveis (representados na árvore estrutural em vermelho, conforme a Figura 14) são extraídos da base de dados através da referência do seu nodo folha a essa base. As partes ou conteúdos que compõem a árvore são classificados como variáveis quando forem dados que devem ser alterados a cada documento, como, por exemplo, o nome e o endereço dos diferentes usuários ou clientes (conforme a Figura 16). Por serem grandes quantidades de dados, ficam melhores organizados em uma base de dados, tornando o modelo mais simples e fácil de ser controlado.

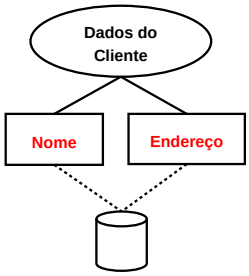


Figura 16 – Exemplo de conteúdo variável.

4.7 Conteúdo seletivo

Com os formalismos e tipos de conteúdos até agora apresentados, pode-se notar o surgimento de problemas relacionados com alguns nodos da árvore que contenham duas ou mais sub-árvores para serem inseridas no documento final, como por exemplo, a sub-árvore **Viagem** da Figura 14, reproduzida na Figura 17. Baseado apenas nas classificações de conteúdos existentes e nos seus comportamentos de funcionamento, o modelo irá apresentar todos os conteúdos disponíveis que sejam filhos da sub-árvore, ou seja, no caso desse exemplo, tanto a sub-árvore com os conteúdos sobre a cidade de Nova Iorque quanto os conteúdos sobre Paris serão inseridos nas diferentes versões finais.

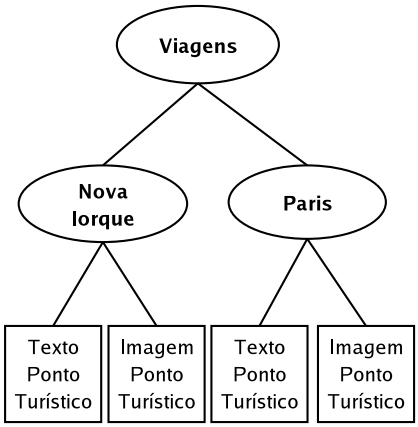


Figura 17 – Exemplo do problema existente quando se tem um nodo com duas sub-árvores onde ambas serão inseridas no documento final.

Porém, deseja-se que esse modelo permita selecionar dentre as diferentes opções disponíveis de uma sub-árvore (no caso do exemplo, entre as opções Nova Iorque e Paris) qual dessas melhor encaixa-se nas preferências do usuário e apresentar apenas algumas dessas no documento final, evitando assim o problema inicial onde havia a criação de múltiplas árvores para atender esse processo de escolha. Para resolver esse problema e para que essa escolha torne-se possível é necessária a classificação de conteúdos a ser inserido nas folhas como **Conteúdo Seletivo**.

As folhas que possuírem esse conteúdo seletivo, representados na árvore estrutural em verde conforme mostra a Figura 14, comportam-se como estruturas que estão disponíveis na árvore estrutural e podem ou não ser inseridas no documento final conforme as preferências na geração do documento. Com isso, os conteúdos seletivos, assim como o conteúdo variável, são a outra parte do formalismo do modelo que podem ser alteradas a cada documento personalizado, perante uma escolha dentre os conteúdos disponíveis.

Esse processo de escolha acontecerá com a introdução de um novo formalismo ao modelo que possibilite as decisões de escolhas de conteúdos para que o documento final seja gerado. Esse novo formalismo será denominado **Seletor**, descrito a seguir.

4.7.1 Seletor

O seletor é um novo nodo interno que terá como finalidade decidir se as suas sub-árvores deverão ou não ter seus conteúdos explorados e conseqüentemente inseridos num documento gerado para cada um dos usuários.

A inserção desse formalismo como um nodo fará com que as folhas abaixo desse seletor sempre sejam classificadas como folhas com conteúdos seletivo. Esse seletor será mostrado na representação física desse modelo através da figura geométrica de um losango, conforme a Figura 18.

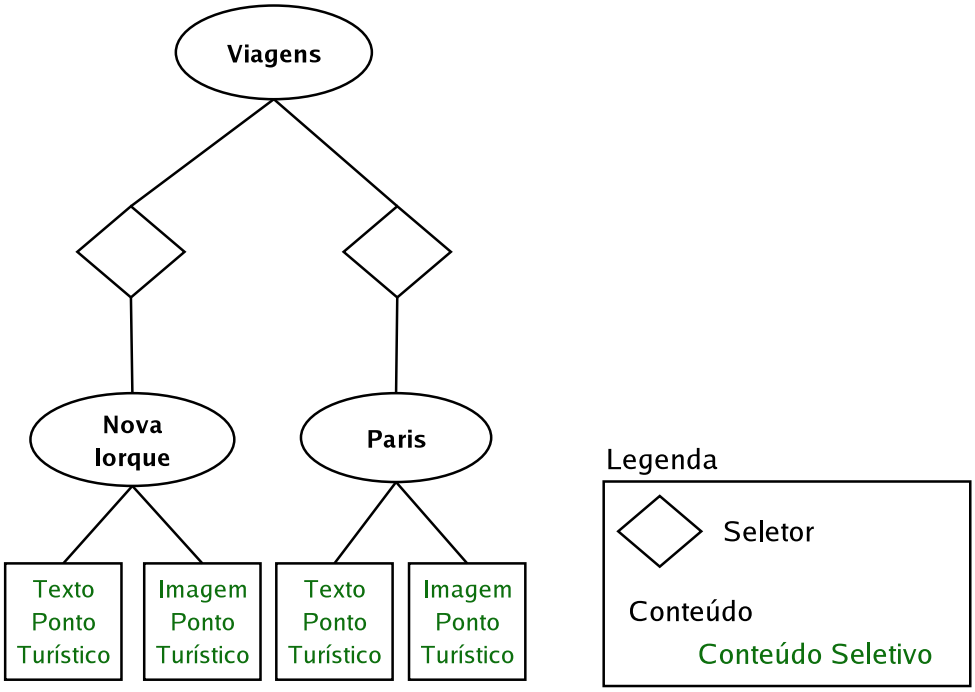


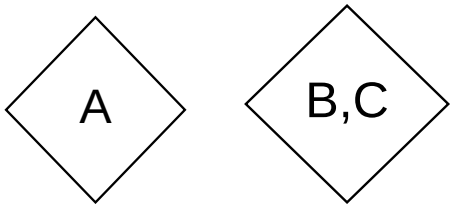
Figura 18 – Inserção do nodo seletor no exemplo da Figura 17.

No interior desse seletor estarão contidas expressões booleanas atribuídas pelo projetista ou desenvolvedor do documento durante a sua estruturação em uma árvore genérica. Essas expressões booleanas deverão ser formadas por variáveis (conforme mostra a Figura 19) que serão denominadas de **Opcionalidades**.

Essas opcionalidades terão o objetivo de verificar, através da validação das expressões, se elas são verdadeiras ou falsas e, portanto determinar, se a sub-árvore e os conteúdos filhos desse seletor devem ou não ser inseridos no documento final.

Pelo fato dos seletores poderem conter uma expressão booleana, precisamos utilizar os **operadores lógicos**, que terão como finalidade combinar as opcionalidades entre si e auxiliar para que os documentos gerados possam ser construídos de forma mais livre e diferente entre eles para cada um dos clientes ou usuários.

Dentre os operadores lógicos existentes, para esse modelo foram escolhidos apenas os principais, como os operadores lógicos **E**, representado nesse modelo pelo símbolo $\wedge$, **OU**, representado pelo símbolo $\vee$ e **Negação**, representado pelo símbolo $\sim$ antes da opcionalidade (Figura 20). A escolha desses operadores lógicos deu-se pelo fato de que apenas com a utilização desses princi-

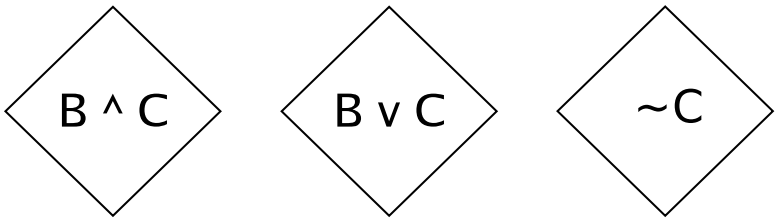


(a) Uma opcionalidade

(b) Conjunto de opcionalidades

Figura 19 – Tipos de opcionalidades atribuídas a um seletor.

pais consegue-se atender a geração da grande parte dos exemplos reais existentes com um alto grau de personalização para cada cliente.



(a) Expressão booleana com o operador lógico E

(b) Expressão booleana com o operador lógico OU

(c) Expressão booleana com o operador lógico de NEGAÇÃO

Figura 20 – Opcionalidades com os operadores lógicos.

O operador lógico de **Negação** sempre aparece implicitamente quando dentro de um seletor existir uma expressão booleana que exclua uma ou mais opcionalidades que foram utilizadas em toda a árvore estruturada principal. Um exemplo disso pode ser visto quando numa árvore estrutural existem as opcionalidades A, B e C e seus seletores contiverem as expressões booleanas **A**, **A** $\vee$ **B** e **C**. Neste caso, para o primeiro seletor existe implicitamente a negação de B e C (como se a expressão fosse A $\wedge$ $\sim$B $\wedge$ $\sim$C), para o segundo seletor a negação implícita de C (A $\vee$ B $\wedge$ $\sim$C) e o último seletor exclui implicitamente A e B ($\sim$A $\wedge$ $\sim$B $\wedge$ C).

Neste momento torna-se possível demonstrar como será o funcionamento e comportamento dos seletores trabalhando em conjunto com as opcionalidades e operadores lógicos utilizados no modelo. Esses aspectos serão explorados a seguir.

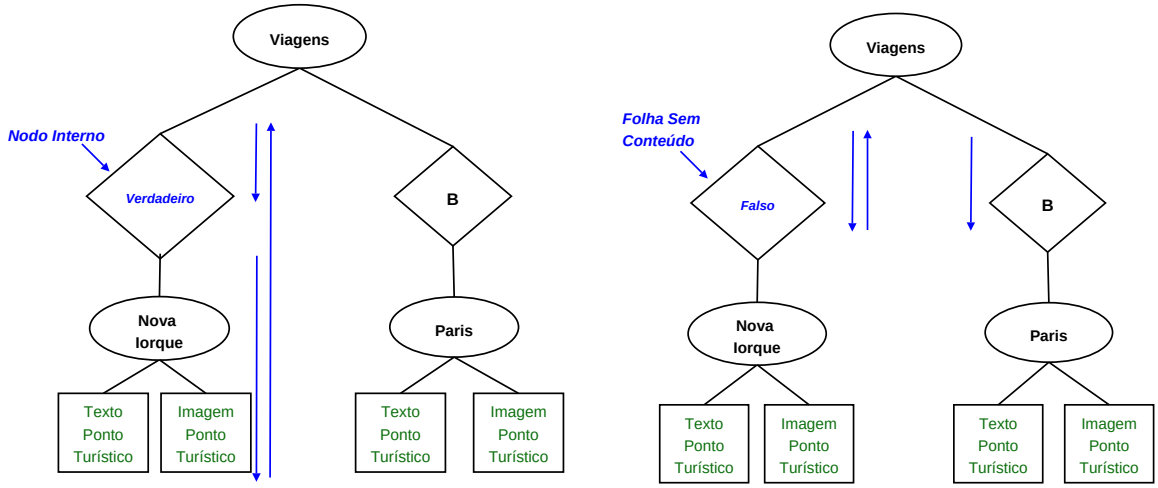
4.7.2 Funcionamento e comportamento dos seletores

O funcionamento dos seletores acontecerá através da avaliação de cada expressão booleana contida nos seletores verificando se o valor retornado dessas expressões é verdadeiro ou falso, e com isso determinar se deve-se inserir ou não as sub-árvores desses seletores.

Essa avaliação em cada seletor não ocorrerá antes de percorrer a árvore e sim durante o percurso. Essa escolha de não fazer as verificações previamente se deu pelo fato de que, apesar disso poder tornar o funcionamento prático do modelo mais lento, futuramente, com a criação de novos formalismos, será necessária a avaliação no momento do percurso para tomar decisões

automáticas para a inserção de novos conteúdos (Seção 4.11).

Dependendo do valor retornado (verdadeiro ou falso) das expressões booleanas, o comportamento que o seletor irá assumir pode mudar (conforme mostra a Figura 21):



(a) Expressão booleana retorna verdadeiro (b) Expressão booleana retorna falso

Figura 21 – Comportamento dos seletores dependendo do valor retornado da expressão booleana.

- se a expressão booleana retornar **verdadeiro**, o seletor será classificado como um **nodo interno** na árvore. Nesse caso seus filhos devem ser inseridos no documento personalizado e o modelo faz a varredura até o fim da sub-árvore descendente desse seletor, inserindo os conteúdos seletivos no documento final.

  Um exemplo pode ser visto através da árvore estruturada do lado esquerdo na Figura 21, no qual a expressão booleana é verdadeira: o modelo classifica o seletor como um nodo interno e percorre sua sub-árvore até o fim, inserindo os conteúdos sobre Nova Iorque;

- se a expressão booleana retornar **falso**, o seu seletor assumirá o comportamento de uma **folha sem conteúdo**. Essa classificação fará com que o modelo faça a varredura apenas até o seletor e não será necessário ir até suas folhas porque as mesmas não serão inseridas no documento.

  Um exemplo desse tipo de comportamento pode ser visto através da árvore do lado direito na Figura 21, na qual a expressão booleana tem um valor falso: o seletor será classificado como folha sem conteúdo e o modelo irá retornar, descartando o restante dessa sub-árvore (visto que essa não deve ser inserida no documento) e passa a avaliar, o próximo filho do nodo Viagem, no caso o seletor para nodo Paris.

Para determinar qual comportamento o seletor deve assumir, é necessário que ocorra a avaliação das expressões booleanas mediante a atribuição de valores booleanos feito pelo produtor do documento.

Essas preferências do projetista ou obtidas através de mineração dos dados que a empresa possui sobre o cliente, serão de fundamental importância para que a decisão automática ocorra nesse modelo, determinando se os conteúdos seletivos devem ou não ser inseridos nos documentos

finais. Nesse modelo, essas preferências serão conhecidas como **Parâmetros de Entrada**, os quais serão abordados na próxima seção.

4.7.3 Parâmetros de entrada

Os parâmetros de entrada são usados pelos seletores para que esses consigam determinar quais expressões booleanas são verdadeiras e conseqüentemente quais sub-árvores devem ser percorridas, obtendo assim a inserção apenas dos conteúdos seletivos que expressem as preferências do projetista para o documento a ser produzido.

Esses parâmetros são informações externas ao modelo e que representarão as preferências do cliente sobre aquele documento. Um exemplo acontece quando se deseja a inserção do conteúdo sobre Nova Iorque num documento e esse conteúdo seletivo foi atribuído na árvore estrutural pela opcionalidade **A**, então nesse caso o parâmetro de entrada a ser declarado é **A = Verdadeiro**.

4.7.4 Obtenção dos parâmetros de entrada

Dentre as maneiras existentes para obter-se os parâmetros de entrada existem duas que melhor adaptam-se ao funcionamento do modelo. Esses dois métodos serão denominados de **obtenção por grupos** e **obtenção exclusiva**.

A **obtenção por grupos** é obtida através de características comuns que existem entre os clientes contidos na base de dados. Com essas características mais comuns entre os usuários consegue-se determinar grupos de clientes com preferências similares, podendo-se gerar documentos parecidos para os componentes de um mesmo grupo. Quando trabalha-se com esse método obtém-se um processo menos personalizado, visto que as opcionalidades não são atribuídas especificamente a um único cliente e sim a um grupo.

Como exemplo de grupos formados por características em comum dos clientes, pode-se citar grupos conforme a renda mensal ou o número de compras, grupos de famílias que tenham filhos em determinadas faixas etárias ou famílias que tenham ou não filhos, grupos que comprem mais roupas ou mais eletrônicos, grupos que possuem ou não animais de estimação, grupos determinados pelo valor gasto em compras, grupos com diferentes classes sociais, onde para cada um desses grupos serão atribuídos e oferecidos diferentes viagens ou preços, conforme a condição econômica e interesse de cada pessoa.

A **obtenção exclusiva** é obtida através de campos previamente escolhidos na base de dados (pelo desenvolvedor, também baseado em conhecimentos em *marketing* e principalmente nas regras de negócio que trazem os objetivos a serem atingidos através do documento) conforme as preferências de cada um dos clientes, tornando assim cada parâmetro exclusivo para um usuário.

Esse fato acarreta que os diferentes documentos tenham um alto grau de personalização, o que torna a personalização mais dirigida ao cliente. Porém, a geração dos diferentes documentos torna-se mais lenta, visto que para cada um dos clientes haverá a avaliação de opcionalidades e diferentes parâmetros que existirão na árvore genérica.

Como exemplo desse método pode-se citar um folheto de propaganda personalizado, onde para cada cliente dessa empresa será informado um parâmetro de entrada específico, gerando assim diferentes versões do folheto exclusivas e personalizadas. Essas versões, por exemplo, podem conter somente viagens já realizadas por cada cliente, viagens que cada cliente deseja realizar, ou ainda oferecer destinos que cada cliente da empresa nunca penso conhecer, mas pode passar a querer analisando os pontos turísticos demonstrados nesse folheto.

Com esses dois principais métodos, torna-se necessário determinar o formato em que esses parâmetros de entrada serão fornecidos ao modelo, como será abordado a seguir.

4.7.5 Declaração dos parâmetros de entrada

Para fazer a varredura na árvore estrutural e avaliar os seletores, a maneira mais óbvia com que os parâmetros de entrada devem ser informados é declarando tanto as variáveis booleanas usadas nos seletores quanto os valores que essas devem assumir, ou seja, verdadeiro quando deseja-se a inserção do conteúdo seletivo e falso caso esse conteúdo não deva ser inserido no documento final. Dessa maneira tanto se o valor for verdadeiro ou falso deve ocorrer a declaração de todas as variáveis.

Porém, em vez de declarar todos os parâmetros de entrada e o valor que devem assumir, optamos por declarar apenas as que representam os conteúdos seletivos que assumam o valor verdadeiro e para as que não forem declaradas fica implícito que o valor é falso.

Assim, torna-se possível fazer as comparações e avaliações usando esses parâmetros de entrada nos seletores e determinando quais conteúdos seletivos devem ou não serem inseridos. Esse processo será demonstrado a seguir.

4.7.6 Exemplo do funcionamento dos conteúdos seletivos

Para demonstrar como comporta-se a inserção de conteúdo seletivo em um documento personalizado, será apresentado um exemplo de geração de um folheto de propaganda que ofereça viagens diferentes aos clientes conforme suas preferências.

Baseado na árvore que representa esse folheto de viagens, demonstrada na Figura 22, e também sabendo que o modelo recebeu como informação externa os parâmetros de entrada **B, C** (ou seja, **A = Falso**, **B = C = Verdadeiro**) é possível mostrar como o modelo funciona perante o conteúdo seletivo.

Inicialmente o modelo irá percorrer a primeira sub-árvore que contém informações sobre a empresa, classificada como conteúdo estático, e que por esse motivo serão inseridos em todos os documentos independente dos parâmetros de entrada. Após isso, o modelo passa a percorrer a segunda sub-árvore, denominada Dados do Cliente, e insere no documento os conteúdos variáveis como nome e endereço de cada cliente, que são providos de uma base de dados.

Ao percorrer o nodo denominado Viagens, o modelo passará a avaliar cada um dos seletores descendentes desse nodo durante o seu percurso. Ao avaliar o primeiro seletor que contém a

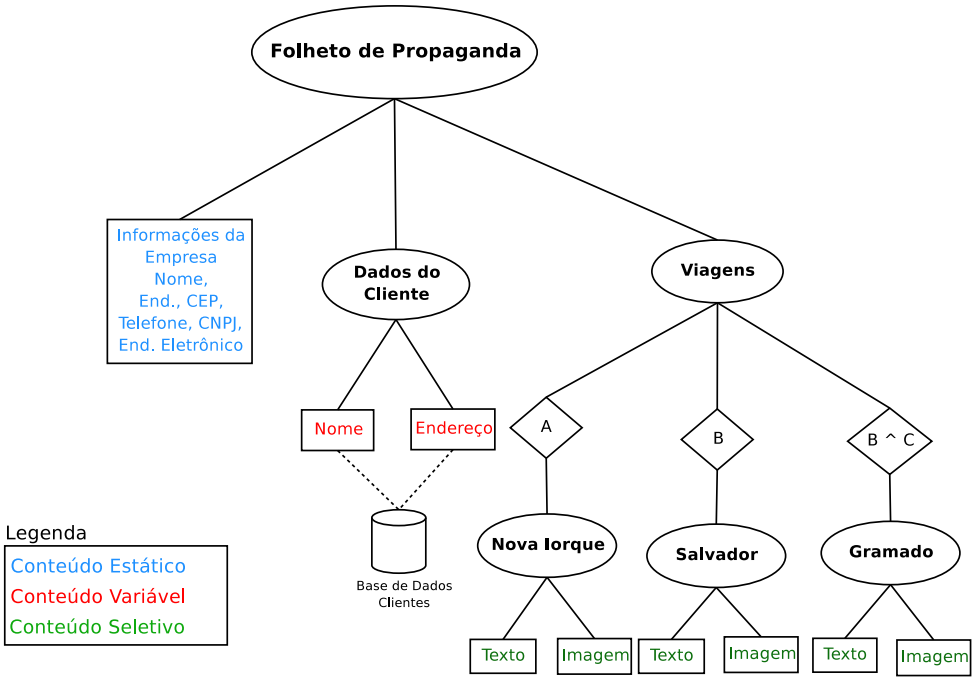


Figura 22 – Exemplo de um folheto de propaganda de viagens.

expressão booleana **A**, o modelo irá determinar, baseado no parâmetro de entrada, que essa expressão booleana para esse caso é falsa. Com isso, o seletor é classificado como folha sem conteúdo, e não deixa que sua sub-árvore seja percorrida até o fim, já que o seu conteúdo não deve ser inserido no documento final.

Após isso, o modelo passa a percorrer a próxima sub-árvore, avaliando o seletor seguinte, que possui expressão booleana **B**. Já que essa variável, no parâmetro de entrada foi atribuída como verdadeira, a expressão booleana é verdadeira. O seletor será classificado como um nodo interno e sua sub-árvore será percorrida até o final inserindo as suas folhas com conteúdo sobre a cidade de Salvador.

Ao inserir esses conteúdos no documento, o modelo passa a percorrer a última sub-árvore do nodo Viagem. Essa possui em seu seletor a expressão booleana $\mathbf{B} \wedge \mathbf{C}$. Visto que **B** foi atribuído como parâmetro de entrada e **C** também (então ambas são verdadeiras) essa expressão booleana é verdadeira e o conteúdo seletivo da viagem a Gramado será inserido no documento final e o modelo termina de fazer seu percurso.

Ao término o documento gerado apresentará, além das informações da empresa, nome e endereço do cliente, os dados personalizados das viagens a Salvador e Gramado. Além da inserção desses conteúdos personalizados pode-se obter a inserção de conteúdos seletivos diferentes alterando apenas os parâmetros de entrada. Essas diferentes seleções de conteúdos seletivos, podem ser vistas na Tabela 1, onde é mostrado que com diferentes parâmetros de entrada e com o mesmo exemplo da árvore estrutural gera-se documentos diferentes.

Parâmetros de Entrada				Conteúdos Seletivos Inseridos
Formato de Declaração	**A**	**B**	**C**	
A	Verdadeiro	Falso	Falso	Sub-árvore Nova Iorque
B	Falso	Verdadeiro	Falso	Sub-árvore Salvador
C	Falso	Falso	Verdadeiro	-
B,C	Falso	Verdadeiro	Verdadeiro	Sub-árvore Salvador Sub-árvore Gramado
A,B	Verdadeiro	Verdadeiro	Falso	Sub-árvore Nova Iorque Sub-árvore Salvador
A,C	Verdadeiro	Falso	Verdadeiro	Sub-árvore Nova Iorque
A,B,C	Verdadeiro	Verdadeiro	Verdadeiro	Sub-árvore Nova Iorque Sub-árvore Salvador Sub-árvore Gramado

Tabela 1 – Tabela mostrando quais sub-árvores serão inseridas de acordo com os parâmetros de entrada no exemplo da Figura 22.

4.8 Conteúdo seletivo/variável

O último tipo de classificação que os conteúdos contidos nas folhas podem assumir é o conteúdo seletivo/variável, representado na Figura 14 em roxo. Esse tipo de conteúdo não representa a criação de novos conceitos nem tampouco pode ser considerado novo e imprescindível ao formalismo do modelo, isto porque o mesmo unifica os conceitos e comportamentos do conteúdo variável e seletivo.

A principal diferença de funcionamento entre o conteúdo seletivo e o conteúdo seletivo/variável é que no primeiro o conteúdo é sempre igual e só é decidido se o mesmo será ou não inserido no documento, já no conteúdo seletivo/variável, além de escolher se o conteúdo irá compor ou não o documento, o mesmo se alterará conforme as características e preferências do cliente, visto que esses vêm de uma base de dados, assim como o conteúdo variável.

Um exemplo desse tipo de conteúdo pode ser visto na sub-árvore da Figura 23, onde existem duas opções (conteúdos seletivos) para determinar qual viagem deve ser oferecida, escolhendo se esse destino deve ser Nova Iorque ou Paris. Nas sub-árvores dessas viagens estão contidos os conteúdos que irão compor documento final, o quais trazem um texto sobre o destino escolhido seguido de informações e preços sobre diferentes pontos turísticos da região, os quais vão alterar-se conforme a época do ano ou as preferências do cliente (conteúdo variável). Essa variação ocorrerá baseado em dados obtidos previamente e armazenados na base de dados, que permitirá essa mudança a cada documento.

4.9 Múltiplas bases de dados

Visto que os dados classificados como variáveis ou como seletivos/variáveis são extraídos de uma base de dados, em alguns casos uma única base poderá não ser suficiente para representar

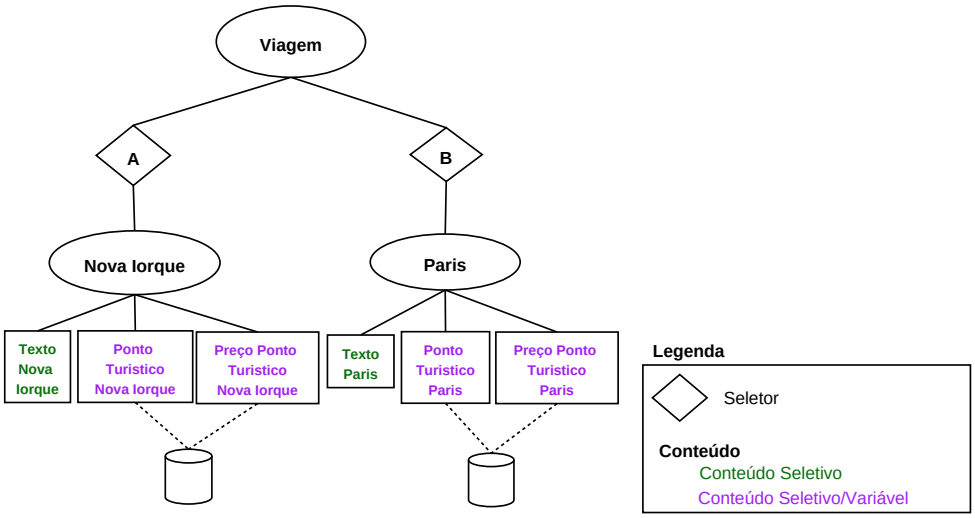


Figura 23 – Exemplo de conteúdo seletivo/variável.

alguns documentos. Para deixar o modelo mais completo e próximo de casos reais incluímos a utilização de **múltiplas bases de dados**, independentes em relação aos conteúdos disponíveis, mas que poderão ter ou não um relacionamento entre si (baseado nos seus campos como uma base que tenha cidade do usuário e outra que tenha nome de lojas de diferentes regiões, onde deve-se escolher a loja baseado na cidade do cliente) para tomar decisões sobre a escolha e extração da base de dados de certos conteúdos variáveis ou seletivos/variáveis.

Utilizar múltiplas bases de dados permite a geração de documento personalizado que necessite de um maior grau de complexidade, sem ter seletores em todas as ramificações.

Trabalhar com essas dependências entre as bases de dados para fazer as escolhas de conteúdo fará com que o processo de geração de documento final seja mais automático e dependa menos do desenvolvedor do documento, sem que o mesmo precise ter conhecimento prévio de algumas informações relacionadas a cada cliente ou grupo de clientes. Esse relacionamento entre as bases de dados permitirá alterar os conteúdos variáveis e seletivos/variáveis (como fotos, imagens, tabelas de pagamento retiradas de uma base de dados) conforme características do cliente como, por exemplo, mudar o banco onde deverá ser efetuado o pagamento de compras conforme a cidade ou estado onde o cliente reside, oferecer viagens para lugares diferentes de onde o usuário vive (se o cliente mora no RS oferecer viagens para o norte ou nordeste), divulgar produtos referentes aos costumes e hábitos utilizados na região (para o norte evitar oferecer roupas de inverno e para a parte sul evitar disponibilizar roupas de verão durante o inverno).

4.10 Objetos

Definem-se **objetos** como sub-árvores criadas independentemente da árvore principal do documento, contendo em sua estrutura elementos (como textos, imagens, tabelas ou o agrupamento desses) que irão compor o documento final.

As partes do documento que perante a estruturação tornam-se grandes, muito extensas e conseqüentemente deixem a árvore estruturada principal confusa podem ser estruturadas como

objetos, o que ajuda o modelo a tornar-se mais simples. Baseado nisso, tratar essas sub-árvores complexas como objetos faz com que a árvore principal apresente menos nodos.

A inserção desses objetos na árvore que representa o documento completo ocorre através do **encapsulamento** de um nodo da árvore principal, que faz uma chamada para essas sub-árvores independentes (Figura 24, onde os nodos que fazem o encapsulamento dos objetos apresentam um símbolo especial, ou seja, uma seta). Um aspecto importante relacionado aos objetos é que perante a árvore principal essas estruturas serão tratadas como nodos internos, visto que mesmo que na árvore principal os nodos objetos não apresentem filhos, quando o encapsulamento dessa sub-árvore ocorrer, esses nodos passarão a ter filhos.

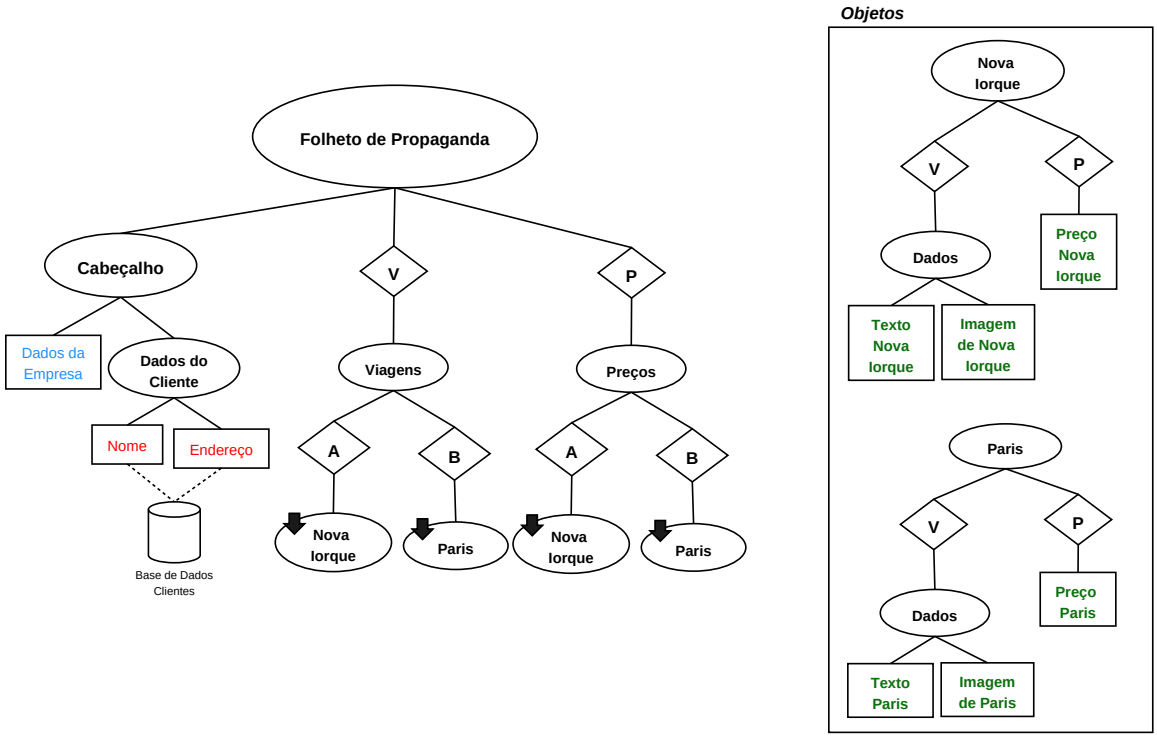


Figura 24 – Árvore com os nodos de chamada para o encapsulamento e objetos a serem encapsulados.

O funcionamento do encapsulamento dos objetos ocorre quando o modelo faz a busca pela árvore principal e explora um nodo objeto. Ao explorar esse nodo objeto ocorre a chamada e a inserção dessa sub-árvore na árvore principal, como se esse objeto fosse uma sub-árvore já estruturada pelo desenvolvedor previamente ali.

No modelo, formalizar as sub-árvores como objetos independentes da árvore principal acarreta duas principais vantagens:

- permite o agrupamento de nodos ou sub-árvores como se fossem pacotes fechados de conteúdos que sejam parecidos ou apresentem fatos em comum (como por exemplo um texto sobre uma cidade seguido de uma imagem da mesma cidade) e que obrigatoriamente devem aparecer juntos e de forma consistente em um documento. Abordar esse formalismo em uma sub-árvore evita que ocorra a inconsistência lógica, onde um cliente receberia um texto de uma cidade **A** seguido de uma imagem de um ponto turístico de uma cidade **B**;
- objetos estruturados de forma separada da árvore principal podem ser reutilizados em

outros documentos bem como em mídias ou meios de saída diferentes. Isso permite que esses objetos sejam novamente oferecidos em folhetos de viagens em meses diferentes ou gerar uma mesma propaganda através de meios de saída distintos como e-mail, poster, folder ou site personalizado.

4.10.1 Transformação de árvores para grafos

Quando se trabalha atribuindo objetos a uma sub-árvore pode-se observar o surgimento de alguns problemas em casos de documentos mais complexos. Esses problemas devem ser resolvidos através de restrições ou através de uma nova abordagem de representação para esse documento, que melhor traduza a sua estrutura.

O primeiro problema encontrado está na representação de árvore onde, por exemplo, dois nodos distintos de uma mesma árvore pedem o encapsulamento de um mesmo objeto. Ao fazer a referência, conforme pode ser visto na Figura 25, a **árvore genérica** deixa de ser a forma correta de apresentar esse documento estruturado, e passa a ser representada através de um **grafo**.

Outro problema que pode surgir com a utilização de objetos é quando ocorrem referências circulares entre os encapsulamentos ou até entre os próprios objetos. Um exemplo desse problema seria se um nodo **A** da árvore principal contiver uma referência para um objeto **B**, o qual possui outra referência para **A**, acarretando que a varredura formalizada para o modelo acabe em *loop*. Para que se evite esse tipo de problema, o melhor é permitir apenas que o modelo seja representado como um **grafo acíclico dirigido (DAG)**, ou seja, um grafo que não possui ciclos.

Além disso, durante a estruturação e construção do documento em uma estrutura de grafo acíclico, através de uma ferramenta de edição, é necessário que seja garantindo a consistência das partes que formam esse documento.

4.11 Ações

Com os formalismos até agora estipulados para o modelo, notou-se que o mesmo seria incapaz de tomar decisões automáticas sem os parâmetros de entrada, como por exemplo alterar textos conforme a soma de produtos oferecidos, adicionando textos que abordem mais tipos de pagamento quando a soma de produtos for maior que um valor estipulado, ou contabilizar o número de viagens que irá compor os documentos finais, valor que muda conforme os parâmetros de entrada.

Para resolver essa restrição e para finalizar o modelo foi introduzido um novo formalismo, denominado **Ações**, o qual funcionará como um mecanismo para calcular valores ou validar expressões dinamicamente, durante o processo de varredura pela árvore, sem a influência do projetista.

Para a inclusão das ações é necessário que ocorra a inserção de um novo nodo, representado no modelo por um retângulo (em azul, conforme mostra a Figura 26), o qual é adicionado

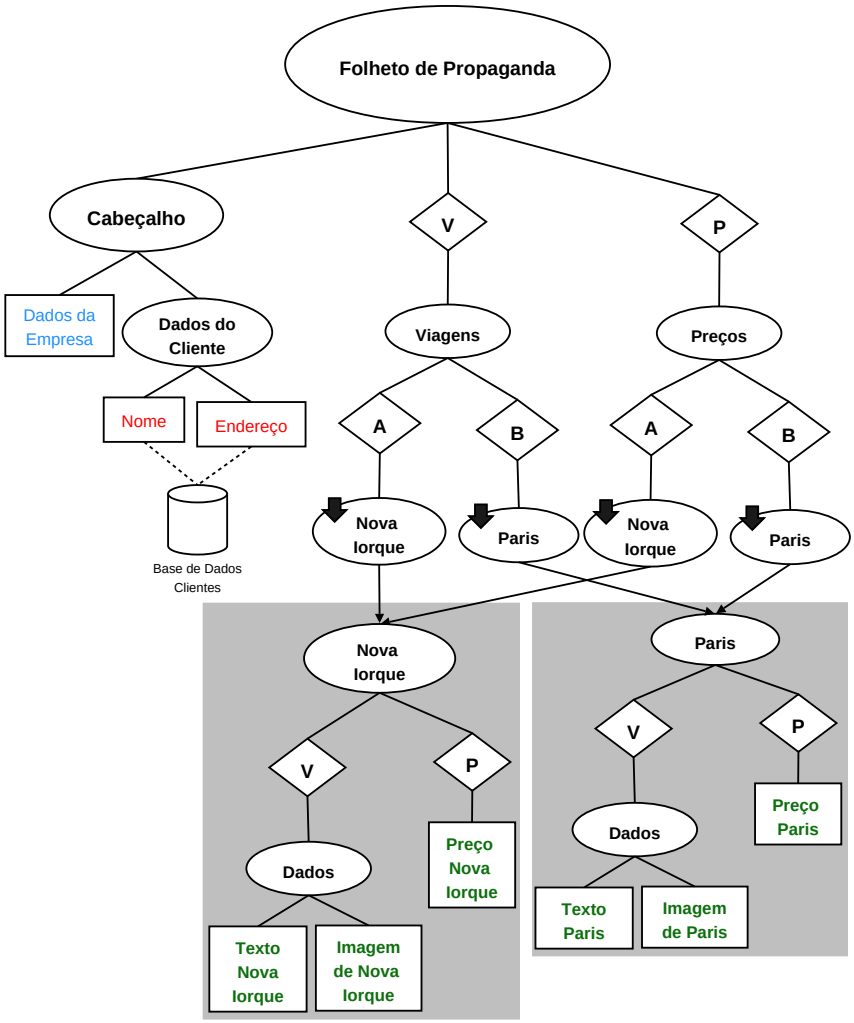


Figura 25 – Representação de um caso onde um documento estruturado é representado através de grafo acíclico dirigido.

paralelamente a qualquer folha, nodo objeto ou nodo interno que compõem a árvore estrutural.

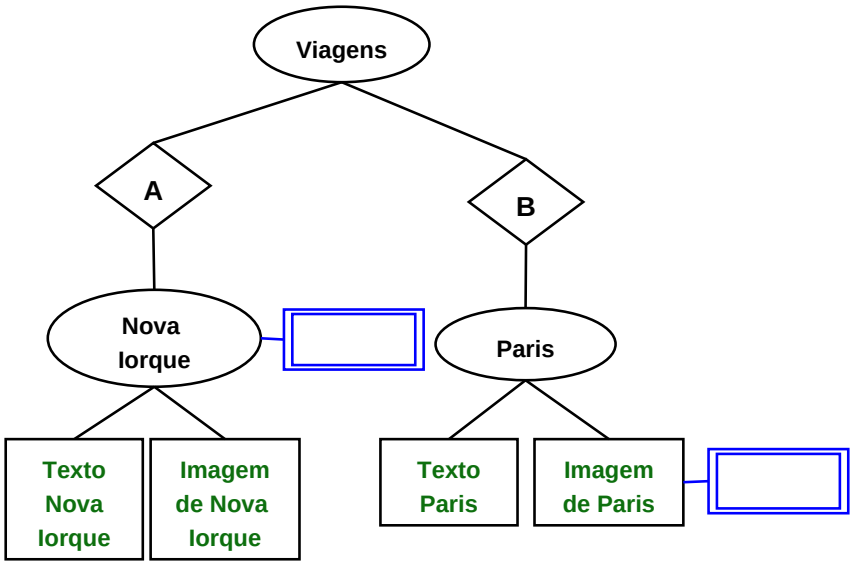


Figura 26 – Representação gráfica das ações nesse modelo.

No interior desses retângulos estarão contidas as ações (conforme mostra a Figura 27, em azul) que o modelo deverá executar caso o mesmo explore e insira no documento final os nodos que estão organizados paralelamente a esse nodo. Alguns exemplos de ações são a contagem do número de produtos, de capítulos, de textos ou de ramificações a ser inserido em cada documento personalizado, a soma de preços dos produtos, dos valores de disciplinas ou o número de créditos efetuados pelo aluno em cada semestre.

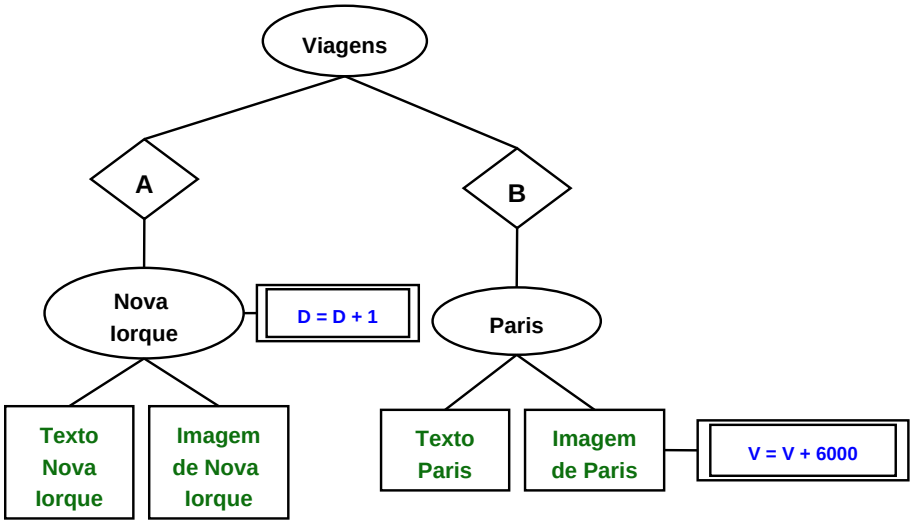


Figura 27 – Ações a serem executadas pelo modelo.

A execução dessas ações ocorrerá sempre durante o percurso do modelo pela árvore estrutural e quando o mesmo visita os nodos ou folhas que possuírem uma ação e explora seus conteúdos. Dentre ações permitidas pelo modelo, duas foram escolhidas para serem utilizadas e detalhadas conforme será abordado a seguir.

4.11.1 Tipos de ações - primeira abordagem

A primeira abordagem que traz os tipos de ações permitidas trabalha com uma expressão que tem um contador, como por exemplo para somar os preços das viagens ou produtos oferecidos, ou somar quantos nodos da árvore estrutural foram visitados pelo modelo.

Esse valor da soma será encontrado ao fim da execução do modelo, e poderá completar dinamicamente um conteúdo que espera esse valor encontrado, como um texto contido em uma folha. A folha que receberá esse conteúdo a ser completado vai ser classificada como um tipo especial de conteúdo estático, pois apesar desse valor alterar-se dinamicamente em cada documento gerado, esse texto deverá ser apresentado sempre em todos os documentos, só alterando o valor.

Um exemplo prático desse tipo de abordagem pode ser visto através da Figura 28, que aborda a utilização dessas ações como se fossem um contador para somar o número de viagens oferecidas num documento e a partir disso inserir esse valor em um texto final do documento personalizado.

Supondo que o parâmetro de entrada fornecido ao modelo é **A**, o mesmo percorrerá a árvore explorando seus conteúdos e adicionará no documento final os conteúdos sobre a cidade de Nova Iorque (executando uma ação para contabilizar), sobre a cidade de Salvador (contando mais

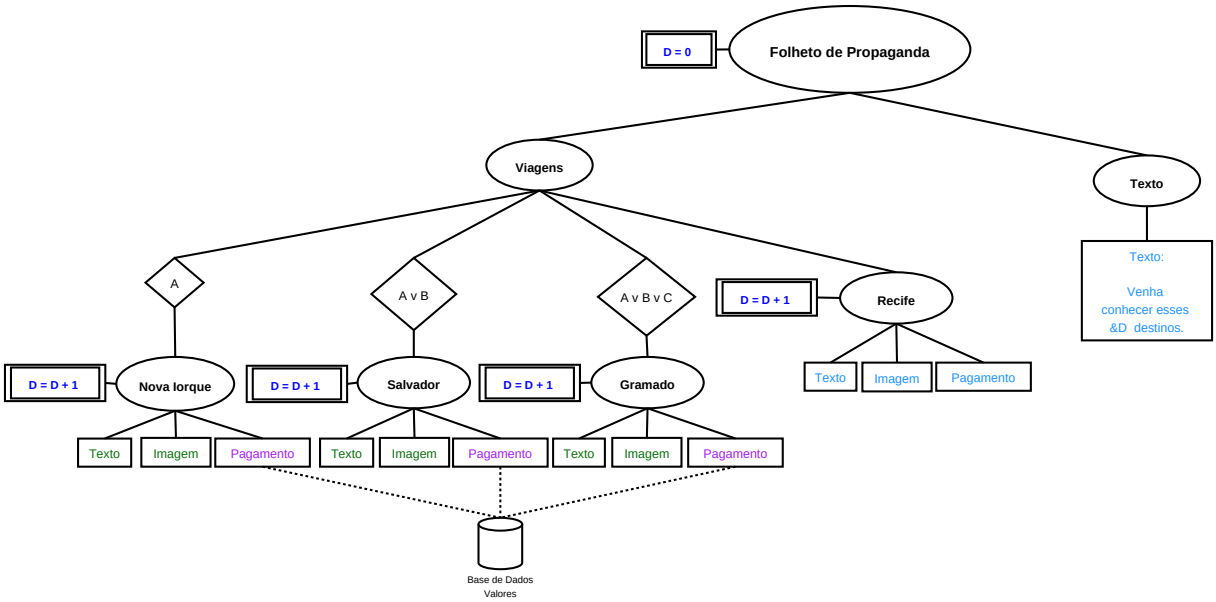


Figura 28 – Formalismo de ações em somador.

um) e sobre Gramado (contando mais um), visto que esses satisfazem o parâmetro de entrada fornecido. No fim, será inserido em todos os documentos, independente das preferências do usuário, o conteúdo sobre a cidade de Recife (conteúdo classificado como estático) e executa como ação a atribuição de mais um à variável que está contabilizando o total de viagens.

Quando o modelo adicionar no documento o texto final, contido no nodo Texto, colocará no espaço destino ao valor da variável interna (**&D**) o número de viagens contabilizadas como destinos diferentes oferecidos, ou seja, apresentará o seguinte texto: "`Venha conhecer esses 4 destinos!`". Porém, se o parâmetro de entrada oferecido for **C**, o texto que aparecerá no final do documento será "`Venha conhecer esses 2 destinos!`", visto que só será inseridos os conteúdos referentes as viagens a Gramado e Recife e com isso o mecanismo de ação só ocorrerá duas vezes.

Além de somar o número de viagens para compor um texto final, o mesmo processo pode somar o valor de pagamento de todas viagens oferecidas ao cliente. Em vez de conter as ações nos nodos internos, essas deveriam estar contidas nas folhas com conteúdo seletivo/variável **Pagamento** e o modelo somará os valores vindos da base de dados e inserido em um texto, ou até mesmo colocando no documento diferentes textos finais de parcelamento dependo do valor obtido.

4.11.2 Tipos de ações - segunda abordagem

A segunda abordagem para ações baseia-se em variáveis que recebem a atribuição de valores booleanos (verdadeiro ou falso). Essa árvore que contém esse tipo de ação, também deve possuir um seletor composto por essas variáveis contidas nas ações, para ser validado. Se todas as variáveis contidas no interior desse seletor tiver sofrido uma ação tornando-as verdadeiras, a ramificação descendente desse seletor será adicionada no documento final.

Nesse caso os conteúdos das folhas descendentes desse seletor serão conteúdos seletivos ou seletivos/variáveis, visto que os mesmos passarão por uma ação de escolha para selecionar se conteúdos devem ou não compor o documento final.

Um exemplo desse tipo de ação pode ser visto na Figura 29, e se o parâmetro de entrada é **A,B**, o modelo irá começar a fazer sua varredura por essa árvore estrutural.

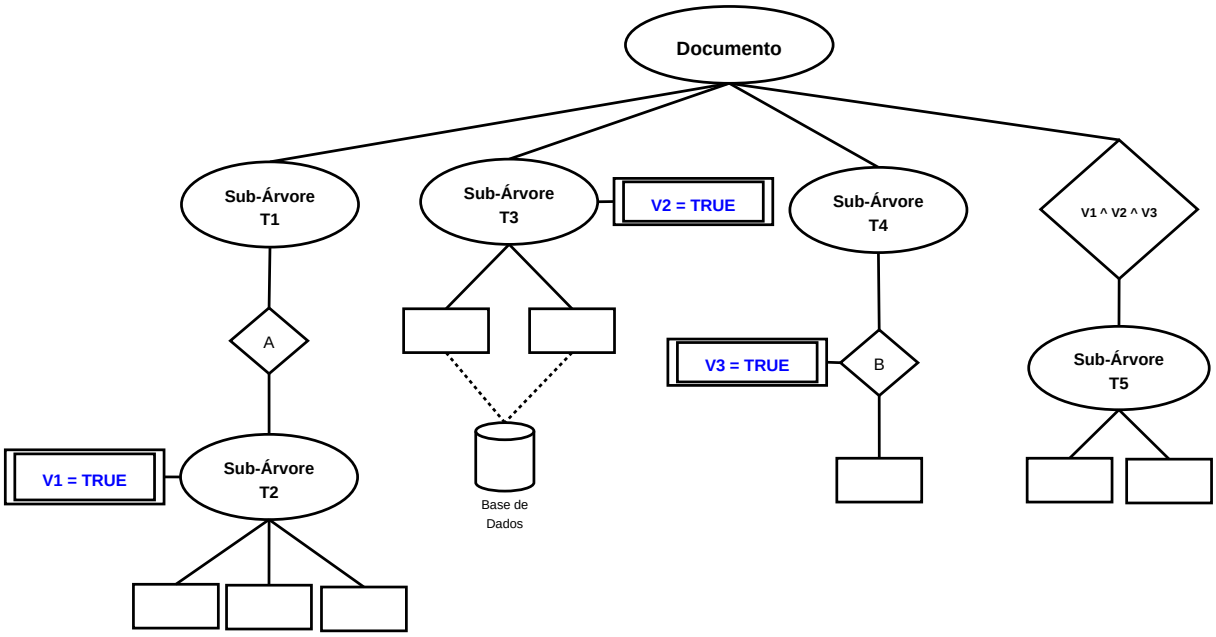


Figura 29 – Formalismo de ações para validar um seletor automaticamente.

Primeiramente, o modelo irá percorrer a sub-árvore T_1, e nessa encontrará um seletor com a expressão booleana **A**. Avaliando essa expressão como verdadeira, o modelo irá explorar os filhos desse seletor e executará uma ação atribuindo a variável V_1 contida nessa expressão ao valor verdadeiro. Após isso, a sub-árvore T_3 é percorrida e os conteúdos serão explorados e buscados da base de dados, e com isso a ação nessa sub-árvore irá ser executada tornando a variável V_2 verdadeira.

Com isso, o modelo irá percorrer a sub-árvore seguinte (sub-árvore T_4) e como no seletor existe a expressão booleana **B**, a qual é determinada no parâmetro de entrada como verdadeira, o seu conteúdo descendente é inserido no documento e ocorre a ação tornando a variável V_3 verdadeira.

Por fim, na última ramificação o modelo encontra um seletor com a expressão booleana $V_1 \wedge V_2 \wedge V_3$. Como as variáveis contidas no seletor foram atribuídas nas ações como verdadeiras, o seletor será classificado como um nodo interno com filhos, e seus conteúdos serão inseridos no documento final (ocorre a inserção da sub-árvore T_5).

Porém se o parâmetro de entrada fosse somente **A**, a sub-árvore T_5 não será inserida, pois o modelo não executará a ação da sub-árvore T_4, visto que não será explorada pelo modelo, e os valores para as variáveis não satisfazem a expressão booleana contida no último nodo.

4.12 Outra Abordagem

Durante o desenvolvimento do modelo apresentado anteriormente, foi idealizado outro formalismo que não foi incluído por problemas e detalhes específicos. Essa abordagem idealizada e não aplicada será apresentada nesta seção, abordando como é o seu funcionamento, os formalismos necessários bem como os motivos pela qual não foi utilizada.

4.12.1 Seletores

Baseado no que pode estar contido no interior dos seletores ocorreu o surgimento de uma nova abordagem, a qual foi estudada e será abordada nessa seção.

Essa abordagem trabalha restringindo a utilização de uma única variável no interior dos seletores, evitando assim a utilização dos operadores lógicos E e OU. As funções de avaliação passam a ser resolvidas graficamente durante a estruturação do documento em formato de árvore, onde deve-se utilizar os seletores em seqüência, para representar a avaliação feita pelo operador E, ou paralelamente, para representar a avaliação feita pelo operador OU, conforme será explicado a seguir.

Baseado nessa nova abordagem e aplicando-a sobre o operador lógico E seria simples fazer sua transformação da avaliação de expressões booleanas para a de forma gráfica, usando os mesmos princípios de varredura, avaliação e comportamento que o modelo deverá assumir diante das abordagens.

Neste caso, conforme mostra a Figura 30, o desenvolvedor do documento deverá separar as variáveis que formam as expressões booleanas (B $\wedge$ C) e colocá-las essas no interior de dois ou mais seletores, dependendo do número de variáveis, em seqüência, onde no caso desse exemplo da Figura 30, a árvore baseada na nova abordagem teria um seletor contendo a variável B seguido de outro com a variável C.

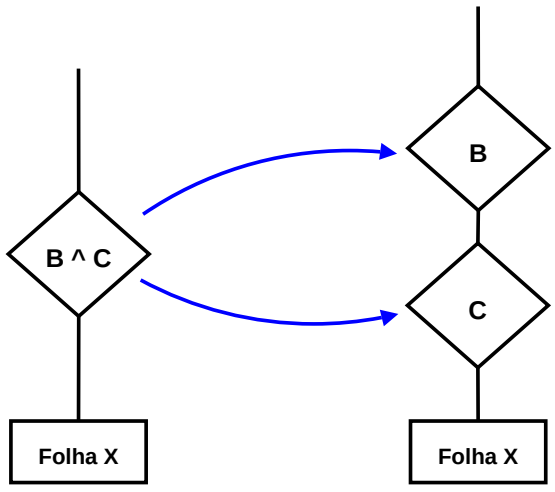


Figura 30 – Representação gráfica do operador E.

Com esses seletores em seqüência, um seletor só será percorrido se o seletor estruturado

acima dele for validado como verdadeiro e os conteúdos descendentes desses só serão explorados e inseridos nos documentos personalizados, caso todas as variáveis contidas no seu interior forem validadas como verdadeiras, como ocorre quando utilizamos o operador lógico E.

No caso do exemplo da Figura 30, o conteúdo *Folha X* só será inserido se o parâmetro de entrada fornecido for **B,C**, visto que ambos os seletores serão validados fazendo com que o modelo explore suas sub-árvores até as folhas. Caso contrário um desses seletores não será validado, o mesmo será classificado como uma folha sem conteúdo e suas sub-árvores descartadas.

Porém, aplicar essa nova abordagem sobre o operador lógico OU, torna-se um pouco mais complexa, apresentando problemas conforme será descrito a seguir.

Neste caso, conforme pode ser visto na Figura 31, o desenvolvedor do documento poderia tentar separar as variáveis que formam as expressões booleanas (B $\vee$ C) e colocá-las no interior de dois ou mais seletores, dependendo do número de variáveis, dispostos em paralelo, onde no caso do exemplo da Figura 31, na nova abordagem teria um seletor contendo só a variável B, seguido de outro só com a variável C.

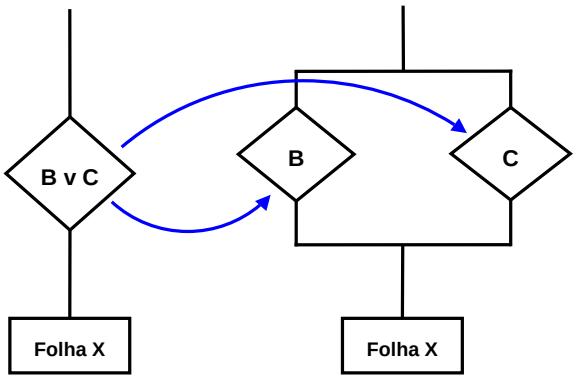


Figura 31 – Representação gráfica do operador OU.

Se o parâmetro de entrada é **B,C**, no entanto, o modo de varredura estipulado para o modelo fará com que o conteúdo *Folha X* seja inserido **duas vezes** no documento final. Isso ocorre porque ambos os seletores serão avaliados como verdadeiro e o modelo explora a sub-árvore descendente quando valida o seletor com variável B e com variável C, introduzindo ao modelo um novo problema de consistência de inserção de conteúdos.

Para tentar solucionar esse problema é preciso adicionar ao modelo uma forma de garantir que os conteúdos sejam inseridos apenas uma única vez e seria necessário fazer esse controle através de novos processos de varredura da árvore, verificando se o conteúdo já faz ou não parte do documento personalizado.

Além de gerar um alto custo no tempo para obter essas árvores finais, essa solução através de várias varreduras acarreta um outro problema para o modelo. Neste problema, conforme mostra a Figura 32, uma sub-árvore do documento (no caso do exemplo a sub-árvore T_3) não possui um seletor e possui uma chamada para esse conteúdo *Folha X*, que também é descendente dos seletores com variáveis B e C. Se restringirmos a adição desse conteúdo a uma única vez, o documento não será gerado de forma consistente.

Para resolver esses problemas será necessário para a avaliação gráfica do OU a utilização de ações que terão a finalidade de contabilizar quantas vezes o modelo explora uma ramificação,

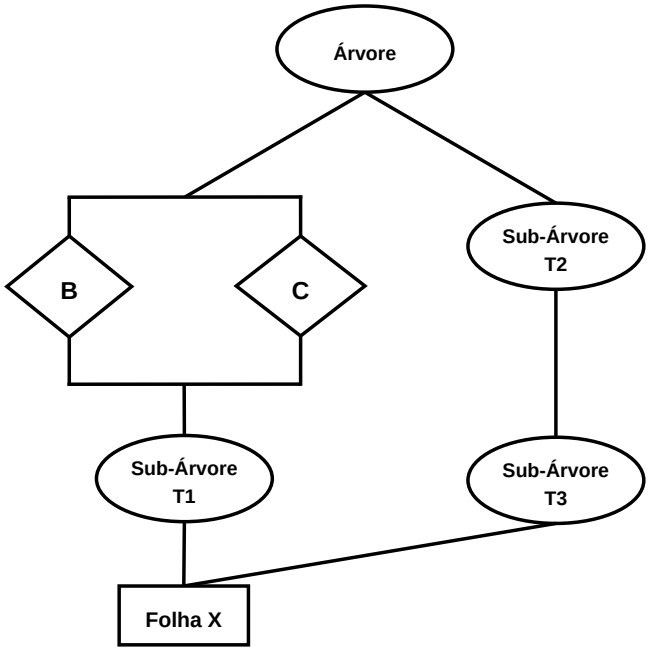


Figura 32 – Problema da representação gráfica do operador OU, quando restringe o número de vezes que o conteúdo aparece.

ou conforme o exemplo da Figura 33, quantas vezes o modelo passou pela sub-árvore T_1. Após as ramificações que obtiverem as ações deverá ser adicionado um seletor que teste o número de vezes que essas foram executadas. No caso do exemplo da Figura 33, será testado se o conteúdo foi inserido uma única vez (*Conta = 1*). Caso esse seletor retorne verdadeiro, o conteúdo *Folha X* vai ser inserido, e caso contrário (se o conteúdo já foi inserido uma vez), o seletor será classificado como uma folha sem conteúdo e seus descendentes serão descartados.

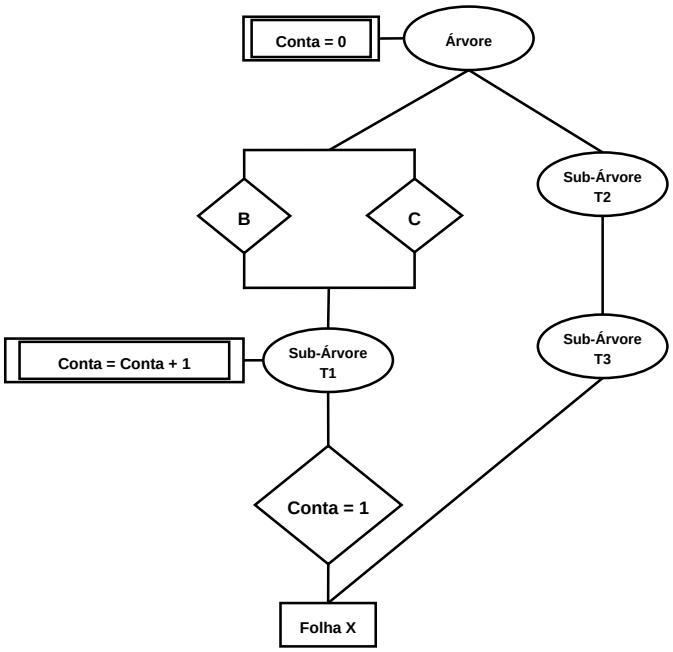


Figura 33 – Solução dos problema da representação gráfica do operador OU, através das ações.

No caso de uma outra sub-árvore (no caso do exemplo da Figura 33, a sub-árvore T_3) fazer a

chamada para esse mesmo conteúdo (conteúdo *Folha X*), o mesmo será inserido repetidamente, visto que a sua referência será feita direto com nodo folha, sem passar pelo seletor e com isso sem verificar, através do seletor ou de novos percursos pelas árvores personalizadas, se o mesmo já foi ou não inserido.

Por fim, nessa abordagem gráfica o operador lógico de negação não teria como ser representado, a não ser através do seu operador lógico onde a variável é antecedida pelo símbolo $\sim$.

Pelo fato do operador lógico de negação não ter uma representação gráfica aceitável e por esse tipo de abordagem deixar as árvores mais extensas e com mais nodos, além dos problemas encontrados e detalhados nessa seção essa representação gráfica não se mostrou apta a ser utilizada e por isso não foi usada no modelo. Outro motivo é que seria necessário a inserção de novos formalismos e ter um maior controle dos conteúdos a serem adicionados, principalmente no caso da representação gráfica OU, fazendo com que esse processo de avaliação do modelo torna-se mais complexo.

5 Geração automática das árvores dos documentos

Baseado no modelo desenvolvido para a produção de documentos personalizados, apresentado no Capítulo 4, é necessário mostrar que o mesmo pode ser aplicado na prática com o objetivo de gerar documentos personalizados com características e preferências exclusivas de cada cliente.

Porém, antes de provar esses conceitos através da implementação de um protótipo e aplicá-los na prática é preciso definir como serão criadas e representadas as árvores dos documentos que serão informadas como entrada para um protótipo que será implementado. Esse protótipo terá como principal finalidade executar os comportamentos estipulados para os formalismos criados no modelo e a partir disso ser capaz de gerar documentos personalizados com conteúdo variável conforme as preferências de cada usuário.

Para que esses arquivos de entrada que representam as árvores com as partes lógicas do documento não precisem ser criados manualmente pelos desenvolvedores e designers através da geração e manipulação de grandes trechos de código em uma linguagem específica, foi estabelecido que elas serão obtidas através de uma ferramenta de editoração que permita a criação de forma visual.

A ferramenta de editoração escolhida será apresentada a seguir, na Seção 5.1, bem como os benefícios, desvantagens e problemas encontrados durante a sua utilização como ferramenta para desenvolver as árvores dos documentos.

5.1 Ferramenta de editoração

Entre as diferentes ferramentas de editoração atualmente existentes (como XFIG, Microsoft Visio e AllClear), o *software* **DIA** foi escolhido para fazer de forma visual o intermediário entre os desenvolvedores e projetistas com a criação das árvores que representam os documentos e que serão repassados ao protótipo a ser desenvolvido.

O **DIA** é uma ferramenta *open-source* existente tanto para o sistema operacional LINUX [DIA 2006] quanto para Windows [DIA 2006], originalmente criado por Alexander Larsson e que atualmente faz parte do projeto GNOME [Foundation 2006].

Essa ferramenta possui como principal finalidade a criação e desenvolvimento de diferentes tipos de diagramas, fluxogramas, árvores ou grafos com a ajuda de diferentes formas ou *shapes*, que podem ser definidos como objetos a serem utilizados para montar os diagramas, como por exemplo linhas, retângulos, quadrados ou qualquer forma geométrica que possa ser utilizada em diagramas. A ferramenta de editoração **DIA** permite que as formas utilizadas nos diagramas possam ser criadas ou que sejam utilizadas as existentes no *software*, conforme a necessidade do desenvolvedor. Por exemplo, se o mesmo deseja desenvolver um diagrama que represente um circuito, pode usar as formas específicas desse tipo de diagrama existentes no **DIA** ou adicionar

novas formas para serem utilizadas nesta criação.

Esse *software* apresenta uma ampla diversidade e quantidade de bibliotecas, que são compostas por formas que representam as formas geométricas utilizadas em diagramas específicos, como por exemplo bibliotecas para criação de fluxogramas, de diagramas UML, de circuitos elétricos, de redes, de *workflow* de aplicações implementadas, entre outras. Por essa ampla quantidade existente pode-se observar o surgimento de um problema, o qual será detalhadamente abordado na seção a seguir, bem como a sua solução.

5.2 Biblioteca de formalismos

Um dos problemas encontrados ao utilizar o *software* **DIA** como ferramenta de editoração foi baseado no fato do mesmo apresentar uma ampla quantidade de bibliotecas, ou seja, formas que podem ser utilizadas como possíveis maneiras de representar os formalismos que compõem o modelo desenvolvido (seletores, folhas, nodos internos, bases de dados, ações, linhas para relacioná-los, entre outros).

Um exemplo disso seriam as linhas que fazem as ligações entre os nodos, os quais na ferramenta podem ser representadas por uma linha reta ou por uma linha zigzag, tipo de linha utilizada para criar diagramas UML contidos na biblioteca de formas UML. Outro exemplo seriam as bases de dados que estão disponíveis no **DIA** para serem utilizadas tanto na biblioteca AADL quanto na SyBases, as quais são representadas por nomes diferentes.

Pelo fato do **DIA** permitir a criação de novas bibliotecas, além das existentes, que abordem somente formas utilizáveis sobre um tipo de diagrama (por exemplo, uma biblioteca com formas somente para criar fluxogramas), e para solucionar esse problema existente foi desenvolvida uma biblioteca contendo uma coleção de formas que representam os formalismos do modelo. Esta biblioteca foi desenvolvida com o objetivo de delimitar as formas e os nomes dos símbolos que o protótipo a ser desenvolvido reconhecerá.

As novas formas criadas para compor essa biblioteca (que pode ser vista na Figura 34), denominada **Biblioteca para Modelagem de Documentos Personalizados**, foram desenvolvidos a partir de pequenos e simples trechos XML, composto por *tags* que somente o **DIA** reconhece (XML próprio da ferramenta), associados a um conjunto de gráficos vetoriais escaláveis, ou seja, *Scalable Vectorial Graphics* (SVG). Esses gráficos SVG tem o objetivo de descrever como serão as formas que representarão os formalismos do modelo, tornando a ferramenta **DIA** capaz de desenhar essas formas nos diagramas e permitindo que os desenvolvedores possam aumentá-las sem que elas percam qualidade, principal benefício de se trabalhar com SVG.

Associado a esse código SVG será informada uma imagem em miniatura no formato PNG, com o desenho igual ao obtido no SVG, de cada forma que compõe a biblioteca, que terá como principal finalidade preencher os ícones referentes aos símbolos que estão disponíveis para serem utilizados (conforme a Figura 34).

Um exemplo desse processo de criação pode ser visto na Figura 35, a qual traz o arquivo obtido ao término da criação de uma das formas que compõe o modelo (formalismo Seletor), onde o trecho do código em XML pode ser analisado em azul, o sub-conjunto SVG em vermelho

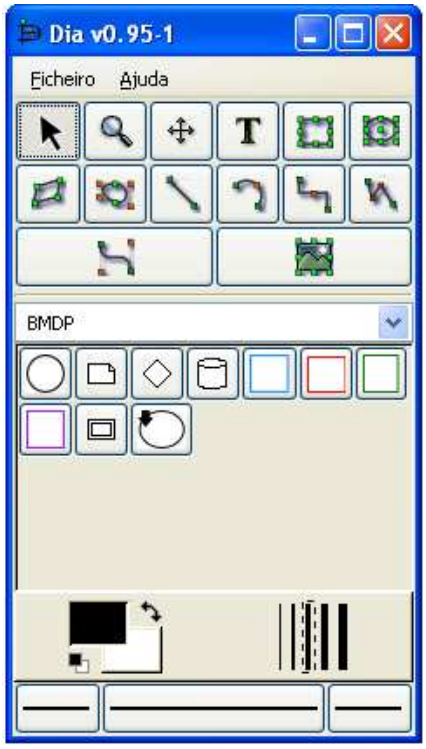


Figura 34 – **Biblioteca para Modelagem de Documentos Personalizados** e os símbolos disponíveis representados através dos ícones PNG.

e a chamada para a utilização da miniatura em PNG como um ícone da biblioteca criada.

```
<?xml version="1.0"?>
<shape xmlns="http://www.daa.com.au/~james/dia-shape-ns"
                    xmlns:svg="http://www.w3.org/2000/svg">
        <name>Modelo - Seletor</name>
        <description>Nodo Seletor </description>
        <icon>Seletor.png</icon>
        <connections>
                <point x="1.25" y="0"/>
                <point x="1.25" y="2.5"/>
                <point x="2.5" y="1.25"/>
                <point x="0" y="1.25"/>
                <point x="1.25" y="1.25" main="yes"/>
        </connections>
        <textbox x1="0.65" y1="0.65" x2="1.90" y2="1.90"/>

        <svg:svg width="2.5" height="2.5">
        <svg:polygon points="1.25,0 2.5,1.25 1.25,2.5 0,1.25"
          style="fill: default"/>
         </svg:svg>
</shape>
```

Minuatura em PNG da forma geométrica do Seletor

Figura 35 – Arquivo XML criado para o formalismo **Seletor**.

Além do exemplo mostrado na Figura 35, os outros componentes da biblioteca criada para representar os formalismos na ferramenta **DIA** e conseqüentemente serem reconhecidos pelo

protótipo serão detalhadamente descritos na subseção a seguir.

5.2.1 Formalismos e suas representações

Com a criação das formas que representam o modelo na ferramenta **DIA**, é necessário também estipular o que será permitido utilizar em cada símbolo e principalmente delimitar algumas restrições que devem ser usadas na criação dos diagramas que representam os documentos.

Na Tabela 2, podem ser vistos as formas de estruturações que os documentos podem ter (grafos e árvores), além do modo em que os nodos que compõem essa estrutura de dados deve ser ordenados e estruturados (da esquerda para direita e *top-down*) para que o protótipo modele corretamente os documentos.

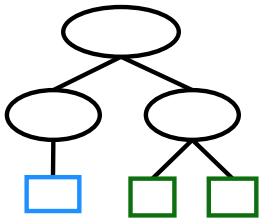
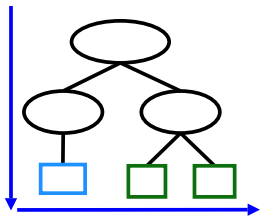
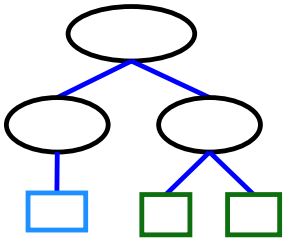
Estrutura de Árvores Genéricas ou Grafos		
Permissões	**Formato de Representação**	**Restrições**
Essas são as duas formas de estruturas de dados que os documentos podem ser estruturados e desenhados no **DIA**, para que o protótipo reconheça os documentos.		Não Há.
Modo de Travessia		
Permissões	**Formato de Representação**	**Restrições**
Essas árvores ou grafos que representam os documentos deverão ser estruturados e desenhados de forma ***top-down*** e ordenadas da **esquerda para direita**.		Não Há.

Tabela 2 – Permissões e restrições das formas de representação dos documentos na ferramenta de editoração **DIA**.

A partir das formas de representação definidas torna-se necessário conhecer as permissões e restrições de cada símbolo, as quais têm como principais finalidades determinar o que pode ou não ser utilizado em cada um, quando eles podem ser utilizados, a quais outros símbolos obrigatoriamente devem estar ligados para se ter um funcionamento completo e o que deve estar contido no interior de cada símbolo. Além disso, deve ser definida a forma com que essas

declarações no interior dos símbolos devem acontecer para que o protótipo reconheça e faça as execuções necessárias corretamente, respeitando, por exemplo, os espaços estipulados entre as variáveis que formam as expressões booleanas contidas no interior dos seletores.

Essas informações sobre cada um dos símbolos que compõem a biblioteca do modelo no **DIA** serão demonstradas através de um conjunto de tabelas, formado das Tabelas 3 a 17, onde cada uma aborda as permissões e restrições de cada um dos símbolos.

Linhas[1] **(Ligações)** 		
Permissões	**Formato de Representação**	**Restrições**
Tem a finalidade de fazer as ligações entre as partes e nodos que formam os documentos estruturados.		Outros tipos de linhas disponíveis em bibliotecas não deverão ser utilizadas na criação das árvores (por exemplo, as linhas disponíveis na biblioteca UML).

[1] Esse símbolo não está disponível na biblioteca e é representado pela linha padrão do **DIA**, disponível como o terceiro ícone na segunda fila do menu principal na versão 0.95-1.

Tabela 3 – Permissões e restrições do símbolo de ligação (linhas).

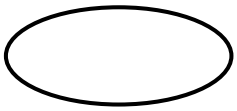
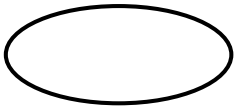
Nodo Raiz 		
Permissões	**Formato de Representação**	**Restrições**
O nodo raiz é um nodo interno que possuirá apenas ligações saindo do nodo.		Nenhuma linha de ligação deve chegar a esse nodo raiz.
No interior do nodo raiz deverão estar contidos somente textos que mostrem o tipo de documento que está sendo estruturado.		Os textos do interior desses nodos não vão compor o documento final.

Tabela 4 – Permissões e restrições do símbolo nodo raiz.

5.3 Desenvolvimento e transformação das árvores no DIA

A partir da **Biblioteca para Modelagem de Documentos Personalizados** e das permissões e restrições existentes para os símbolos que a compõe torna-se possível criar as primeiras árvores ou grafos estruturados que representam os documentos e para que posteriormente, após

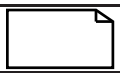
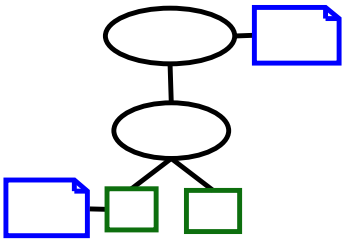
Comentários 		
Permissões	**Formato de Representação**	**Restrições**
Esse tipo de símbolo pode ser associado a qualquer nodo raiz, nodo interno, seletores, folhas ou objetos que compõem a árvore ou grafo estruturado.		
No interior de um comentário devem ser mantidos textos que descrevam as partes, conteúdos e opcionalidades atribuídas aos seletores.		
Uma árvore pode conter vários comentários entre seus nodos.		

Tabela 5 – Permissões e restrições do símbolo comentário.

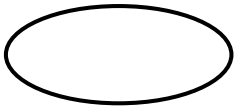
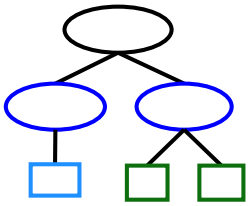
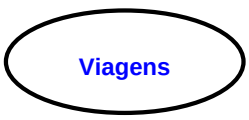
Nodo Interno 		
Permissões	**Formato de Representação**	**Restrições**
Qualquer nodo que não for terminal e que por isso apresente um ou mais descendentes deve ser classificado com esse tipo de símbolo.		Os textos do interior desses nodos não vão compor o documento final.
No interior dos nodos devem estar contidos textos que descrevam e organizem as partes lógicas do documento estruturado.		

Tabela 6 – Permissões e restrições do símbolo nodo interno.

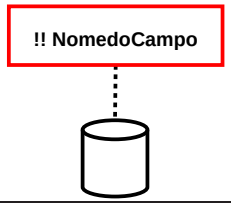
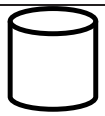
Folha com Conteúdo Variável		
Permissões	**Formato de Representação**	**Restrições**
Qualquer nodo que tiver conteúdo provindo de uma base de dados deverá ser classificado com esse símbolo.		
No interior desse nodo deverá estar contido dois caracteres especiais (**!!**) seguidos do **nome do campo** da tabela de uma base de dados em que os conteúdos a serem providos estão armazenados. Através desses caracteres **!!** sabe-se que a próxima informação é o nome do campo de uma base de dados. Ex: **!! NomedoCampo**		Não se deve inserir no interior do nodo um campo inexistente na base de dados.
Um ou mais conteúdos variáveis podem estar contidos no interior de um texto. Ex: Venha conhecer a cidade de **!! Viagem** só pelo valor de **!! Preço**.		
O símbolo de folhas variáveis deverá obrigatoriamente ser ligado, através das linhas de ligação, com uma base de dados (formalismo de base de dados).		Não se deve classificar um nodo como folha variável sem ligá-lo a uma base de dados.
Após fazer a busca na base de dados essas chamadas serão trocadas por textos, imagens ou conteúdos que mudam a cada documento já que são providos de uma base de dados.		

Tabela 7 – Permissões e restrições do símbolo de folhas com conteúdo variável.

Base de Dados



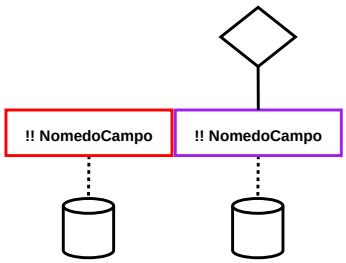
Permissões	Formato de Representação	Restrições
Deverá sempre acompanhar todos os nodos que forem classificados como conteúdo variável ou seletivo/variável		Não devem ser infomadas bases de dados inexistentes, nem tabelas que não estejam contidas dentro dessa mesma base de dados.
Junto com essa base de dados deverá ser informado o **nome da base** (na primeira linha após a imagem da base de dados) em que os dados estão armazenados e o **nome da tabela** (na segunda linha). Essa tabela deverá pertencer a essa base de dados onde os conteúdos estão armazenados.	 Nome da Base Nome da Tabela	
Esse símbolo deve ser conectado obrigatoriamente, através das linhas de ligações, com todos os nodos variáveis e seletivos/variáveis que tiverem conteúdo a serem providos de uma mesma tabela e mesma base de dados.		Dois ou mais nodos só podem ser ligados a uma mesma base caso os conteúdos a serem providos estejam armazenados na mesma tabela e base de dados.

Tabela 8 – Permissões e restrições do símbolo de base de dados.

Folha com Conteúdo Estático

Permissões	Formato de Representação	Restrições
Qualquer nodo que mantiver seu conteúdo idêntico em todos os documentos dever ser classificado com esse símbolo.		
No interior desse nodo deverão estar conteúdos como textos, imagens ou diagramas.		

Tabela 9 – Permissões e restrições do símbolo de folhas com conteúdo estático.

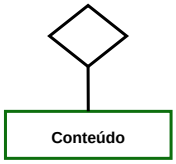
Folha com Conteúdo Seletivo		
Permissões	**Formato de Representação**	**Restrições**
Qualquer nodo descendente de um seletor e que por conta disso pode ou não ter seu conteúdo inserido deverá ser classificado com esse tipo de símbolo.		
No interior desse nodo deverão estar conteúdos como textos, imagens e diagramas que podem ou não ser inseridos.		
Os nodos devem obrigatoriamente ser filhos de um seletor ou descendentes de sub-árvores com seletores.		Não se deve classificar um nodo como conteúdo seletivo sem o mesmo ser filho ou descendente de um seletor.

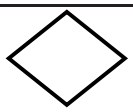
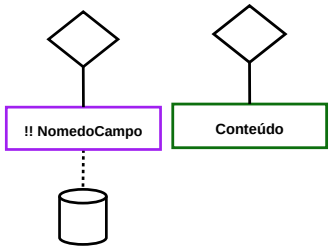
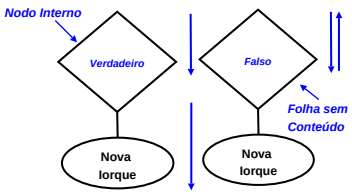
Tabela 10 – Permissões e restrições do símbolo de folhas com conteúdo seletivo.

uma transformação, o protótipo os reconheça.

Esse processo de desenvolvimento, apresentado na Figura 36, inicia quando um projetista, desenvolvedor ou designer de documentos, através da ferramenta **DIA**, começa a estruturar visualmente os documentos idealizados em formato de árvore ou grafo, obedecendo os modos de travessia que foram estipulados para o modelo (Seção 4.3). A fim de se ter um melhor controle sobre essa ordenação dos nodos durante o desenvolvimento dos documentos, a ferramenta **DIA** permite mostrar, através da *Árvore de Diagramas*, a hierarquia das formas utilizadas nos diagramas criados pelo desenvolvedor do documento.

Essas árvores ou grafos apenas poderão ser estruturadas utilizando as formas criadas e contidas na **Biblioteca para Modelagem de Documentos Personalizados** em conjunto com as ***linhas padrões*** do **DIA** para conectar os nodos que formam o diagrama. Na Figura 37 pode ser visto um exemplo de uma árvore gerada utilizando o conjunto de formas da biblioteca criada através do **DIA**, a qual representa um folheto de propaganda que utiliza nodos internos, conteúdo estático, conteúdos variáveis, base de dados, conteúdos seletivos, conteúdos seletivos/variáveis, seletores, expressões booleanas, ações com cálculos aritméticos e a troca desses valores encontrados.

Pelo fato das árvores só usarem as formas da biblioteca criada, para o desenvolvimento de documentos, a biblioteca de formalismos deverá estar adicionada ao **DIA** quando os desenvolvedores ou projetistas desejarem criar as árvores ou grafos estruturados e quando os mesmos quiserem alterar essas árvores ou grafos personalizados obtidas após a execução do protótipo sobre os arquivos originais. Isso porque no primeiro caso o protótipo não reconhecerá outras formas a não ser as contidas nessa biblioteca e no segundo caso a ferramenta ao carregar as

<table>
<tr><td colspan="3">Seletor[2]  </td></tr>
<tr><td>Permissões</td><td>Formato de Representação</td><td>Restrições</td></tr>
<tr><td>Esse símbolo deve ser estruturado antes de sub-árvores que possuírem nodos folhas com conteúdo seletivo ou seletivo/variável.</td><td> 
</td><td rowspan="3">A expressão booleana contida no seletor deve ser informada sem o valor booleano que deve assumir. Todas as expressões dos seletores sempre devem ter o valor verdadeiro para ter seus conteúdos explorados.</td></tr>
<tr><td>No interior desses seletores deve estar contida uma expressão booleana (Tabela 12).</td><td> 
</td></tr>
<tr><td>Essa expressão booleana possibilitará determinar o comportamento que os seletores devem assumir: comportamento de nodo interno com filhos (quando for verdadeiro) ou de folha sem conteúdo (quando for falso).</td><td> 
</td></tr>
</table>

[2] Esse mesmo símbolo representará os seletores de ações, que conterão no seu interior expressões booleanas com as variáveis iguais às contidas nas ações.

Tabela 11 – Permissões e restrições do símbolo seletor.

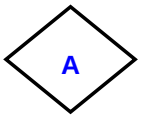
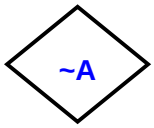
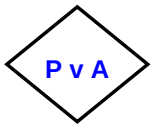
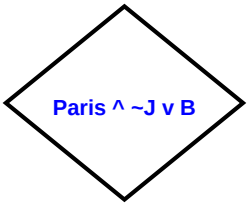
Expressões Booleanas		
Permissões	**Formato de Representação**	**Restrições**
As expressões booleanas devem estar contidas no interior dos seletores.		
Essas expressões booleanas devem ser estruturadas baseadas nos seguintes formatos: Para *única variável* **Negação** - representado pelo símbolo do til (∼)	 	
Para *múltiplas variáveis* **E** - representado pelo acento circunflexo (∧) **OU** - representado pelo letra v em minúsculo (∨)	 	Não deve ser utilizado parênteses na formação da expressão booleana.
O número de variáveis e operadores lógicos que podem compor uma expressão é ilimitado, além disso as variáveis podem ser palavras (por exemplo, **Nomeda-Cidade** em vez de **A**).		Os formatos de declaração devem ser estruturados conforme foram detalhados, com os determinados espaços entre cada variável e utilizar apenas os operadores lógicos permitidos.

Tabela 12 – Permissões e restrições do símbolo das expressões booleanas que devem estar contidas no interior dos seletores.

Parâmetros de Entrada		
Permissões	**Formato de Representação**	**Restrições**
Na declaração dos parâmetros de entrada, deve-se declarar apenas as variáveis que assumam o valor verdadeiro (sem atribuí-las a esse valor), deixando implícito que as que não forem declaradas são falsas.	**A,B** A = B = True, C = D = False	
Para esse modelo as variáveis serão **separadas por vírgulas**, para que essas não precisem ser letras isoladas e também possam ser palavras.	**Paris,F,G,Valores,O**	
Os parâmetros podem ser declarados tanto em minúsculas ou maiúsculas, independente de como foram escritas dentro dos seletores.	**A,b,f,G,k**	
O número de parâmetros de entrada é ilimitado.	**A,B,F,G,K,O,L...**	
Esses parâmetros serão passados ao protótipo por linha de comando ou por um arquivo externo ao protótipo.		

Tabela 13 – Permissões e restrições do símbolo parâmetros de entrada.

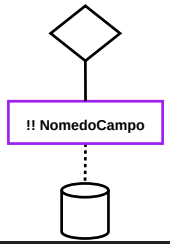
Folha com Conteúdo Seletivo/Variável		
Permissões	**Formato de Representação**	**Restrições**
Qualquer nodo que tiver seu conteúdo provindo de uma base de dados e for descendente de um seletor, pode ou não ter seu conteúdo inserido e deverá ser classificado com esse formalismo.		Não se deve inserir no interior do nodo um campo inexistente na base de dados.
No interior desse nodo deverão estar contidos dois caracteres especiais (**!!**) seguidos do **nome do campo** da tabela de uma base de dados em que os conteúdos estão armazenados. Através desses caracteres **!!** sabe-se que a próxima informação é o nome do campo de uma base de dados. Ex: **!! NomedoCampo**		
Um ou mais conteúdos variáveis podem estar contidos no interior de um texto. Ex: Venha Conhecer a cidade de **!! Viagem** somente pelo valor de **!! Preço**.		
A utilização desse símbolo deverá obrigatoriamente ser ligado com uma base de dados e também ser filho de um seletor ou descendente de sub-árvores com seletores.		Não se deve classificar um nodo como conteúdo seletivo/variável sem ligá-lo a uma base de dados e sem o mesmo ser filho ou descendente de um seletor.
Após fazer a busca na base de dados, essas chamadas serão trocadas por textos, imagens ou conteúdos que mudam a cada documento já que são providos de uma base de dados.		

Tabela 14 – Permissões e restrições do símbolo de conteúdo seletivo/variável.

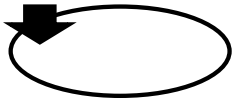
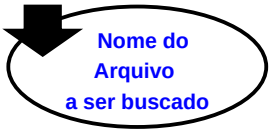
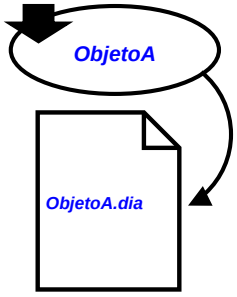
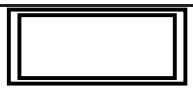
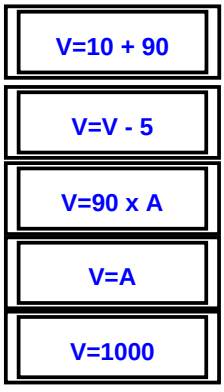
Objetos		
Permissões	**Formato de Representação**	**Restrições**
Utilizado para fazer chamadas às sub-árvores estruturadas independentemente do documento principal, fazendo com que elas sejam inseridas no documento principal.		Não colocar a extensão junto com o nome do arquivo no interior do nodo.
No interior desses nodos objetos devem estar contidos o **nome do arquivo** que possui a sub-árvore objeto a ser encapsulada. A extensão desse arquivo (.dia) não deve ser inserida junto no interior desse seletor (por exemplo, nomedoarquivo.dia).		
Essas sub-árvores a serem inseridas devem ser criadas no **DIA** em um arquivo distinto. O seu nome, ao ser salvo, deve ser igual ao nome inserido na árvore principal.		Não devem ser colocados nomes diferentes entre as chamadas de encapsulamento dos objetos (texto inserido no seu interior) e os nomes dos arquivos que contém a sub-árvore.

Tabela 15 – Permissões e restrições do símbolo objeto.

<table>
<tr><th colspan="3">Ações  </th></tr>
<tr><th>Permissões</th><th>Formato de Representação</th><th>Restrições</th></tr>
<tr><td>Esses nodos podem estar associados a qualquer nodo interno, objeto ou folha da árvore principal ou das sub-árvores objetos.</td><td>  </td><td rowspan="3">Não devem ser associados aos nodos seletores. Essa restrição ocorre porque as ações atribuídas aos seletores serão sempre executadas, visto que esses seletores são sempre visitados pelo modelo, para após isso serem validados.</td></tr>
<tr><td>Nesse nodo estarão as ações que o modelo irá executar caso o nodo que esteja paralelo a ação seja explorado para ter seus conteúdos inseridos nos documentos finais.</td><td> 
</td></tr>
<tr><td>Dois tipos de ações serão aceitas para o modelo.

Aritméticas
As operações aceitas serão: +, *, -, /, %, cos, sin, tan, acos, asin, atan, sqrt, sqr, log, min, max, ceil, floor, abs, neg, rnd

Atribuição de Valores Booleanos
Os valores aceitos serão: True e False.</td><td> 

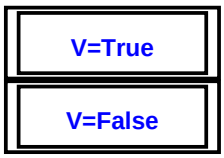

</td></tr>
</table>

Tabela 16 – Permissões e restrições do símbolo de ações.

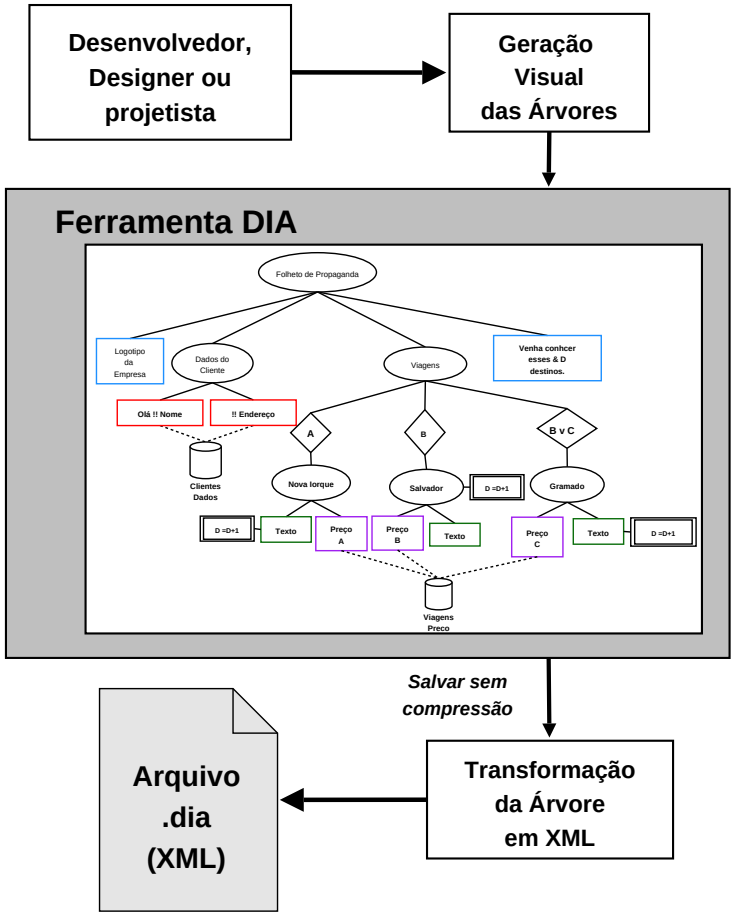


Figura 36 – Processo de desenvolvimento e geração de uma árvore genérica representativa do documento no **DIA**.

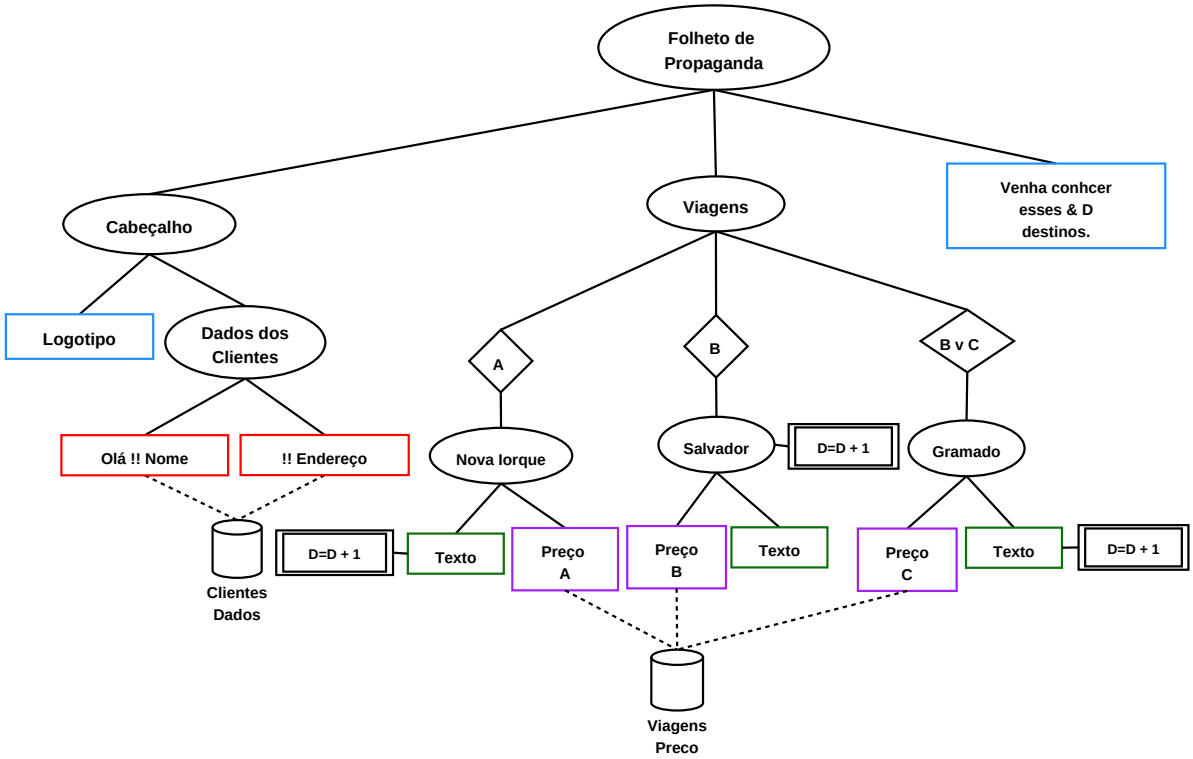


Figura 37 – Árvore de um documento desenvolvido no **DIA**.

Variáveis a serem substituídas pelos valores das ações		
Permissões	**Formato de Representação**	**Restrições**
Para que a inserção dos valores calculados ocorra nas folhas estáticas deverá ser utilizado um caracter especial (&) acompanhado de uma **variável**. Por exemplo: **& Variável**	**& Variável**	
Esse caracter especial acompanhado de uma variável pode ser inserido dentro de uma folha, no interior de um texto.	**Venha conhecer esses & D destinos.**	
Isso fará com que o modelo leia o texto de cada folha e ao encontrar esse caracter especial troca-o pelo valor armazenado na variável calculado nas ações contidas depois dele.	**Venha conhecer esses 10 destinos.**	

Tabela 17 – Permissões e restrições do símbolo de ações baseado em variáveis que devem ser substituídas pelos valores calculados nas ações mostradas na Tabela 16.

árvores ou grafos personalizadas, irá buscar na biblioteca os formatos SVG de cada objeto que compõe o diagrama personalizado e se a mesma não encontrá-los dentre as bibliotecas existentes, a árvore ou grafo personalizado não será carregado.

Após a árvore estar estruturada a etapa final do desenvolvimento deve ocorrer, onde a mesma precisa ser transformada para que, posteriormente, o protótipo seja capaz de reconhecer, modificar e alterar essas árvores.

Esse processo de transformação será feito através da própria ferramenta **DIA**, onde as árvores geradas são salvas no próprio formato da ferramenta, ou seja, a extensão `.dia`, porém sem compressão. Salvar os diagramas gerados no **DIA** sem compressão faz com que o arquivo `.dia` torne-se compreensível para leitura tanto para os desenvolvedores e projetistas dos documentos quanto para o protótipo que irá explorá-lo, pelo fato dos diagramas serem salvos em um formato próprio da ferramenta (ou seja, com tags somente reconhecidas por ela) na linguagem de marcação XML.

Ao término da transformação da árvore estruturada no **DIA** para um arquivo XML, será obtido um arquivo texto organizado hierarquicamente como se fosse uma árvore montada de forma *top-down* e que tratará cada formalismo do modelo como se fosse um nodo independente, tornando mais fácil a busca das informações e conteúdos pelo XML. O resultado desse processo de transformação, ou seja, o XML obtido no **DIA** de uma árvore genérica pode ser visto através da Figura 38.

Esses arquivos, como o representado na Figura 38, irão conter tags de marcação autodescri-

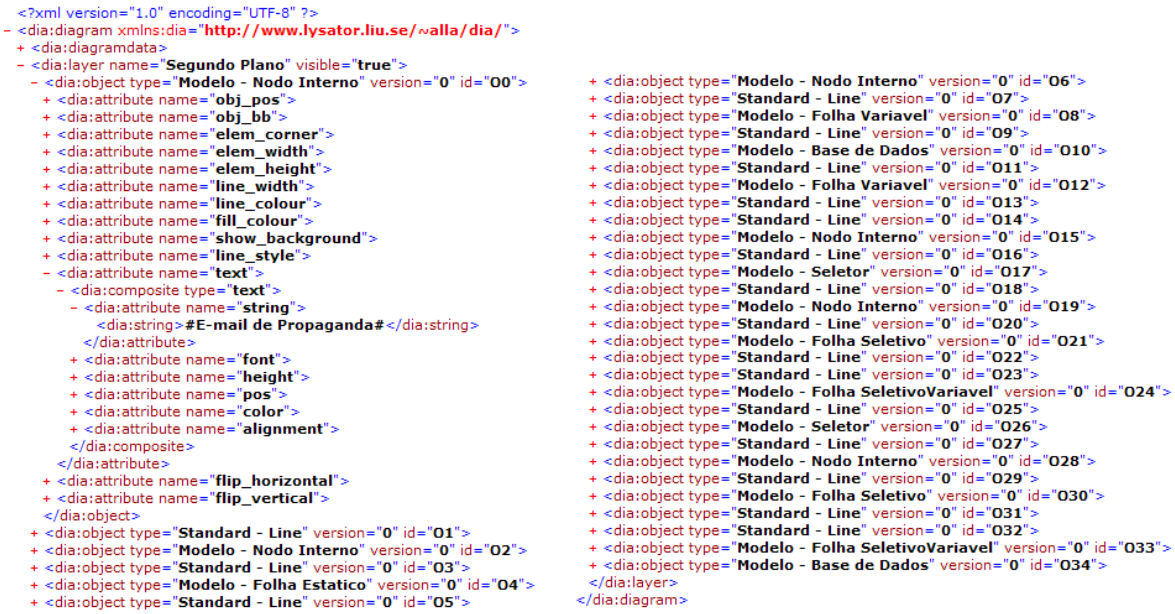


Figura 38 – XML gerado automaticamente ao ser salvo na extensão padrão do **DIA**.

tíveis, ou seja, tags XML próprias da ferramenta **DIA**, que irão fornecer o tipo de informação que está por vir (como por exemplo a tag *type* que irá representar o tipo de classificação do nodo: nodo interno, folha, seletor). Cada nodo deve estar aninhado corretamente, com sua tag de abertura e fechamento correspondente, as quais serão interpretadas conforme a aplicação (no caso o protótipo do modelo) que irá ler e explorar esse documento XML.

Com base nessas características, pode-se observar a vantagem da utilização do **DIA** como a forma de criar e transformar automaticamente as árvores representadas na ferramenta em um arquivo XML, visto que faz com que qualquer desenvolvedor ou projetista possa criar e gerar documentos personalizados de forma bastante simples e sem ter um contato direto com a linguagem XML. Além disso, a ferramenta **DIA** permite que esse arquivo XML seja alterado, através do protótipo, e recarregado na própria ferramenta para que os desenvolvedores consigam analisar as alterações ocorridas.

Após a transformação do documento estruturado para a linguagem XML, as árvores estão prontas para serem informadas como entrada ao protótipo, que será apresentado no próximo capítulo, o qual tem a finalidade de representar o funcionamento dos formalismos do modelo em prática.

6 O protótipo

Com as árvores ou grafos representativos dos documentos desenhados com o intermédio da ferramenta **DIA** e transformados para a linguagem XML, tornando-os legíveis para qualquer tipo de aplicação (conforme abordado no Capítulo 5), é necessária a implementação, a partir do modelo proposto, de um mecanismo que seja capaz de mostrar que a geração automática de documentos personalizados para diferentes clientes pode ser possível na prática, obtendo-os de forma simples e em um tempo razoável.

Esse protótipo para validar as árvores e simular os comportamentos do modelo terá como principal finalidade validar os formalismos para modelagem de documentos personalizados, mostrando o funcionamento deles na prática para realizar experimentos com casos reais, obtendo, após todo o processo do protótipo, documentos personalizados.

Esse protótipo validará e interpretará as árvores em XML, aplicará sobre elas os comportamentos e ações que cada formalismo deve assumir diante de diferentes documentos gerados no **DIA** e, baseado nesses aspectos, funcionará como um mecanismo de seleção e validação que percorrerá as árvores XML, escolhendo entre todas as opções existentes quais conteúdos devem ser inseridos, validando expressões e ações e completando os dados necessários até que sejam gerados diferentes versões de um documento.

Neste capítulo serão apresentadas as características desse protótipo, o qual teve seu desenvolvimento baseado nas finalidades abordadas, explicando quais tecnologias usadas para sua implementação (Seção 6.1), o funcionamento dessa aplicação desde a criação das árvores até a geração das diferentes versões do documento personalizado (Seção 6.2), além de outra abordagem que não foi aplicada na prática (Seção 6.3).

6.1 Linguagem de programação e modelo de objetos

Para o desenvolvimento e a implementação do protótipo foi escolhida a linguagem de programação JAVA [Microsystems 2006]. Essa escolha deu-se, principalmente, pelo fato dela ser uma plataforma de desenvolvimento de aplicações bastante poderosa, que trabalha de forma simples com processamento de arquivos e processamento de banco de dados, fato que ajuda a trabalhar com os conteúdos variáveis e seletivos/variáveis. Além disso, funciona em qualquer sistema operacional e assim qualquer desenvolvedor que trabalhe tanto no sistema LINUX quanto no sistema Windows pode ser capaz de criar e gerar documentos personalizados, visto que a ferramenta **DIA** também está disponível para esses dois sistemas operacionais.

Para que o protótipo desenvolvido em JAVA reconheça os arquivos XML que contém as árvores genéricas dos documentos, é necessária a utilização de um processador de documentos XML que funcione em conjunto com a linguagem JAVA, com o objetivo de fazer a validação e a

leitura desses arquivos bem como navegar entre todos nodos que compõem a árvore estruturada, acessando, analisando e alterando as suas partes e conteúdos.

Para isso, a aplicação precisa utilizar uma API auxiliar que torne possível a realização dessas tarefas de forma que os conjuntos de informações que compõem as árvores sejam projetados em um modelo de objetos que permita ao protótipo manipulá-los. Define-se uma API (*Application Programming Interface*) como um conjunto de rotinas acessíveis por meio de programação, que pode ter suas funcionalidades utilizadas em outros *software*, os quais não precisem envolver-se em detalhes da implementação, mas apenas usar os serviços que a API oferece.

As APIs existentes e mais difundidas para realizar esse processamento dos arquivos XML em JAVA são a *Simple API for XML* (**SAX**) e *Document Object Model* (**DOM**). Dentre essas, a escolhida para auxiliar o processo de desenvolvimento do protótipo foi o modelo de objetos **DOM**, fornecido pela **W3C**, pelo fato dele trabalhar diretamente com a estrutura do documento, armazenando-o inteiro em uma árvore reusável de nodos enquanto o **SAX** é apenas utilizado quando se quer trabalhar com alguns nodos e não se quer navegar por toda a árvore XML.

Baseado nesse aspecto, pode-se definir a API **DOM** com um conjunto de interfaces que decompõem um documento XML em uma árvore transversal hierárquica de nodos, a qual é sempre percorrida através de uma varredura baseada em pré-ordem, tornando mais fácil a interpretação do XML para o protótipo [Veloso 2003].

Através da definição dessa linguagem de programação e do modelo de objetos para auxiliar seu processo de navegação é que o protótipo será implementado. Para que essa implementação ocorra, foi necessário abordar como será todo o seu processo de *workflow*, mostrando todo o processo de modelagem e produção até obter-se diferentes versões de documentos personalizados.

6.2 *Workflow* da aplicação

O *workflow* completo dessa aplicação implementada, conforme mostra a Figura 39, desde a criação dos documentos no **DIA** até a obtenção de múltiplos documentos personalizados com as preferências e características de cada cliente, serão abordados nesta seção.

Primeiramente, um desenvolvedor, designer ou um usuário leigo faz a criação de um documento na forma de uma das estruturas de dados estipuladas (árvores genéricas ou grafos), através da ferramenta **DIA**, em conjunto com os formalismos que compõem a **Biblioteca para Modelagem de Documentos Personalizados**. Após essa árvore ou grafo estar visualmente estruturada, com todas as informações necessárias inseridas, ela deverá ser convertida através da própria ferramenta **DIA** em um formato que o protótipo seja capaz de reconhecer, ou seja, a linguagem XML. Esse processo inicial do *workflow*, onde se tem a geração e transformação das árvores foi abordado no Capítulo 5.

O documento XML, obtido após a transformação, será repassado para o protótipo como entrada, em conjunto com bases de dados e parâmetros de entrada. O protótipo utiliza a proposta de trabalhar com os parâmetros baseado no **método de obtenção por grupo** (Seção 4.7.4). Isso acarreta que seja informado para a aplicação apenas um único conjunto de parâmetros de entrada que determinará a inserção de conteúdos seletivos iguais para um grupo de clientes.

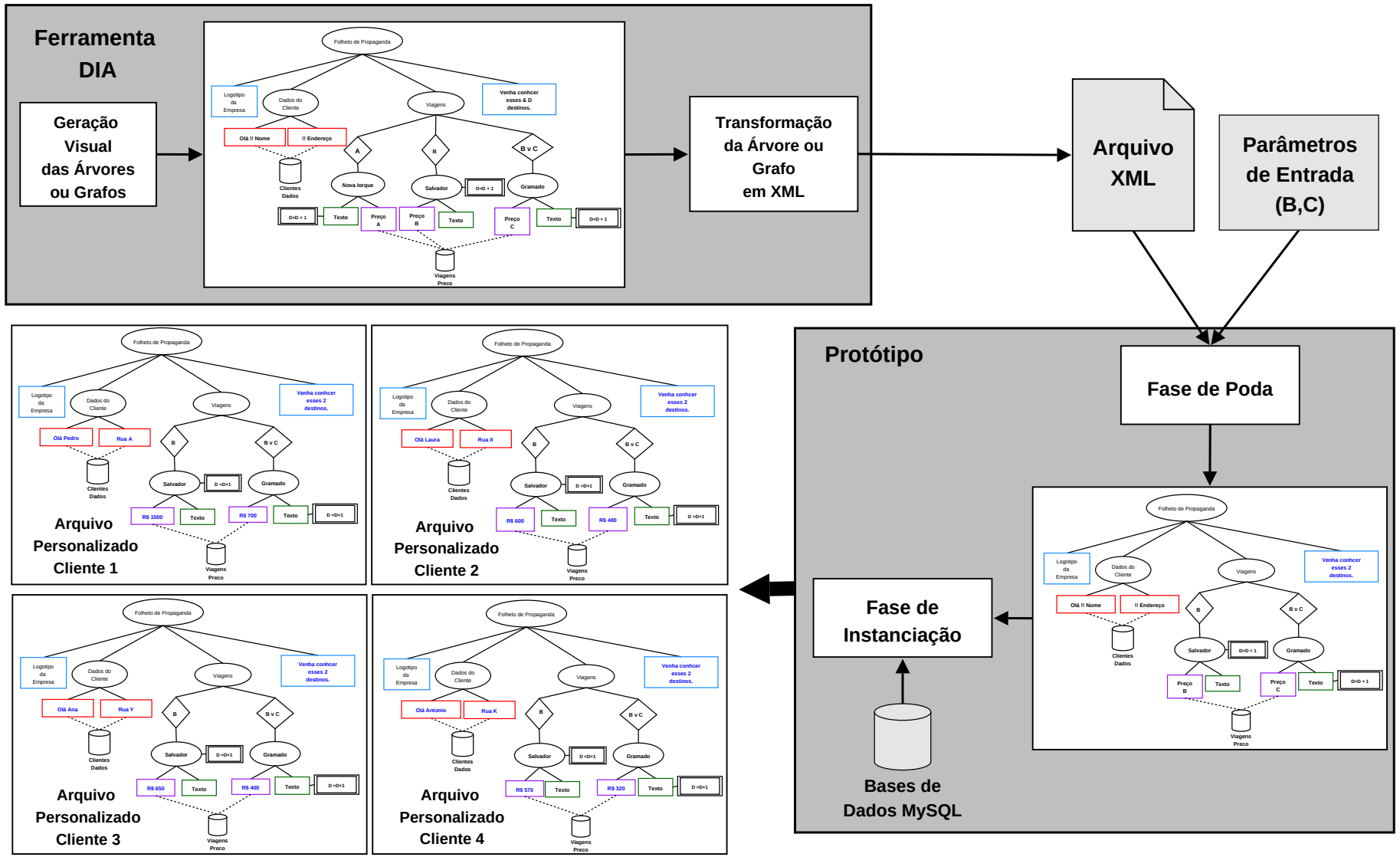


Figura 39 – *Workflow* da aplicação.

Com essas informações repassadas para a aplicação, a mesma pode começar a aplicar seu funcionamento de remoção, inserção e execuções dos formalismos, além da inserção dos conteúdo variáveis sobre cada usuário nos documentos. Baseado nessas tarefas, o protótipo, ao ser implementado, foi dividido em duas principais etapas, que terão responsabilidades diferentes sobre essas árvores.

As duas etapas serão explicadas nas seções a seguir, demonstrando o que será informado como entrada a cada etapa, as tarefas que estas realizarão e o que será obtido na saída.

6.2.1 Implementação do protótipo

O protótipo foi modelado e implementado baseando-se na sua divisão em duas etapas em que os arquivos informados como entrada sofrem transformações e alterações até se obter as diferentes versões de um documento inicial. Essas duas etapas, mostradas na Figura 40, serão denominadas **Fase de Poda** e **Fase de Instanciação**.

A decisão de projetar em duas fases deu-se pelo fato de que com essa idéia consegue-se obter um melhor desempenho de processamento e um baixo custo na geração dos diferentes documentos personalizados, visto que a **Fase de Poda** será executada uma única vez, gerando apenas um documento XML genérico com algumas ramificações removidas e ações executadas, para que somente a **Fase de Instanciação** seja executada diversas vezes personalizando os documentos com as informações dos clientes e gerando várias instâncias do documento, uma para cada cliente.

A seguir essas duas fases serão apresentadas, abordando quais conjuntos de formalismos compõem cada fase e a finalidade das mesmas.

6.2.2 Fase de Poda

A fase que inicia o processamento, a **Fase de Poda**, recebe como informação de entrada parâmentos de entrada (abordados na Seção 4.7.3, como por exemplo *A,Paris,J,Salvador*) e o arquivo que representa o documento, já transformado na linguagem XML (processo de criação e transformação abordado no Capítulo 5).

A partir dessas informações, essa fase terá a finalidade de (conforme mostra a Figura 41), baseado na árvore XML inicial, determinar quais nodos ou sub-árvores devem ser mantidos nos documentos (baseado em comparações com os parâmetros de entrada fornecidos) e quais devem ser descartados da árvore principal, através da aplicação de uma tarefa de poda, onde deve ocorrer a remoção desses nodos ou sub-árvores indesejados. Além disso, a **Fase de Poda** será responsável pela maioria das execuções de comportamento dos formalismos do modelo, executando qualquer tipo de tarefas que possam ser resolvidas para todos os usuários, sem usar informações exclusivas de cada cliente que refletirão na personalização exclusiva final.

Baseado nessa finalidade, os formalismos implementados (apresentados em negrito) e as tarefas em relação ao comportamento desses formalismos, que essa **Fase de Poda** executará,

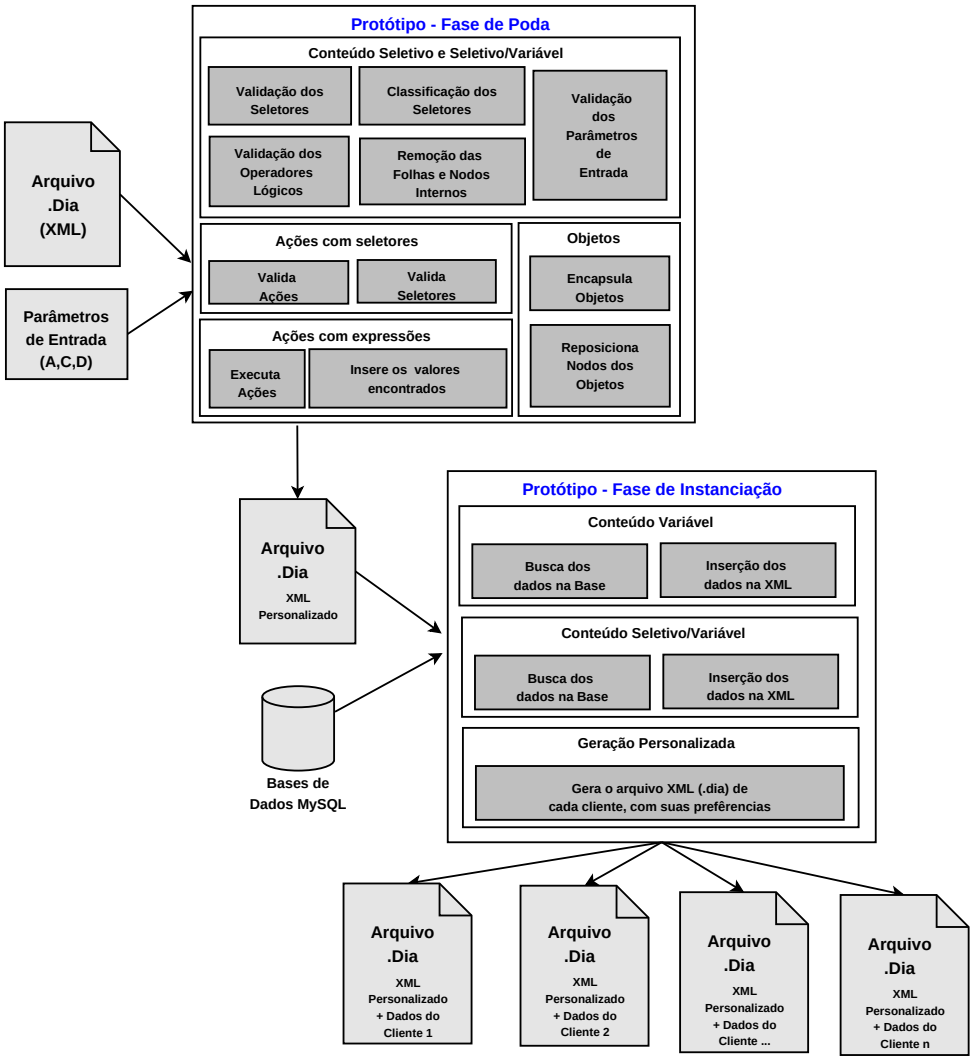


Figura 40 – **Fase de Poda e de Instanciação** que foram modeladas e implementadas para o protótipo.

serão:

1. Validação do XML, verificando as estruturas de dados em que os documentos podem ser formalizados (**árvore genéricas** ou **grafos**);
2. Determina quais nodos são ligados a outros nodos, formando as **sub-árvores**;
3. Validação do **nodo raiz**;
4. Validação dos **comentários**;
5. Percurso pela árvores XML, em **pré-ordem** (o DOM faz esse tipo de varredura) e de forma ***top-down***;
6. Validação dos **nodos internos**;
7. Inserção dos **conteúdos estáticos**;

8. Validação das **expressões booleanas** contidas nos **seletores** de acordo com os parâmetros de entrada ou com os valores atribuídos nas ações;
9. Determinação do **comportamento** que os **seletores** devem assumir diante da validação das expressões booleanas;
10. Inserção ou remoção das sub-árvores ou nodos com **conteúdo seletivo**;
11. Armazena o nome das **bases de dados** e das tabelas em que os dados a serem inseridos na fase seguinte serão providos;
12. Inserção ou remoção das sub-árvores ou nodos com **conteúdo seletivo/variável**;
13. Procura dos **arquivos que contém os objetos** independentes a serem inseridos;
14. Inserção das sub-árvores dos **objetos**;
15. Calcula as **ações** baseadas em expressões aritméticas;
16. Troca esses valores encontrados nas **ações de expressões aritméticas** nos textos contidos nas folhas;
17. Executa as **ações** baseadas nas atribuições de valores booleanos;
18. Valida os seletores baseados nas **ações de atribuições de valores booleanos**.

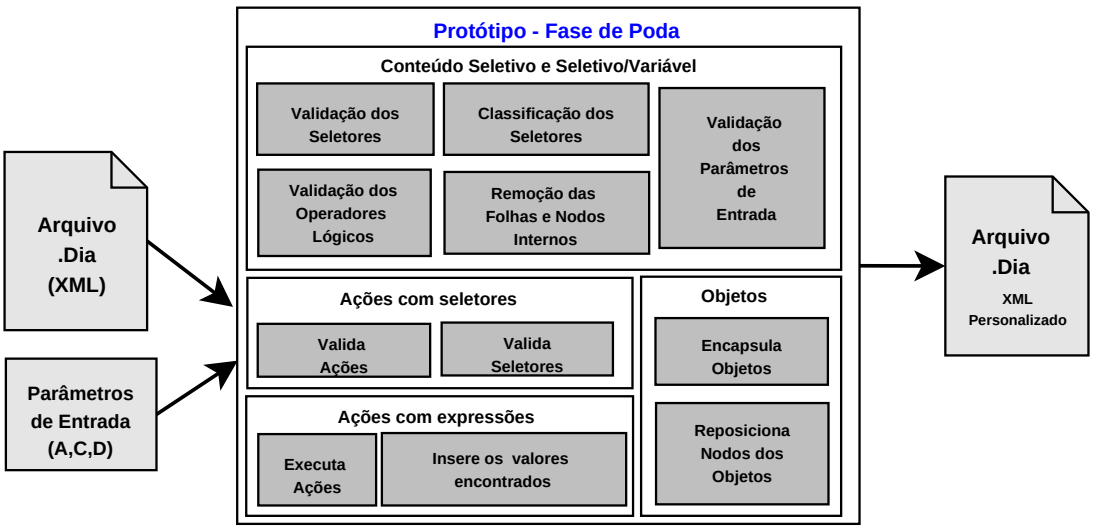


Figura 41 – Principais tarefas executadas pela **Fase de Poda**.

Após a execução de todas essas tarefas que a **Fase de Poda** deve realizar, será obtido um único arquivo XML, bastante parecido com o informado como entrada, porém com algumas podas e sem alguns nodos que faziam parte dele. Esse arquivo possuirá apenas as ramificações e sub-árvores escolhidas de acordo com os parâmetros de entrada (as que não devem compor os documentos finais são descartadas), seus objetos encapsulados e as ações executadas com seus valores alterados.

Um problema encontrado durante a implementação dessa **Fase de Poda** do protótipo ocorreu durante a idealização das ações que trabalham resolvendo expressões com cálculos aritméticos (primeira abordagem de ações).

Esse problema ocorreu porque expressões aritméticas contidas no interior do formalismo de ações são providas para o protótipo num formato texto, ou seja, como *string* visto que são armazenadas em XML. Como a linguagem JAVA não apresenta, por *default*, uma classe ou interface que retorne automaticamente o valor dessas expressões, foi necessária a utilização de uma API externa.

Essa API escolhida, denominada ***Java Math Expression Evaluator***, terá a finalidade de avaliar dinamicamente e sem transformações uma expressão matemática contida dentro das ações de primeira abordagem e retornar o valor dessa expressão, armazenando em uma outra variável. Além disso, ela permite que o protótipo trabalhe com uma ampla quantidade de operadores básicos (apresentado na Tabela 16 do Capítulo 5) como a soma, subtração, divisão, resto da divisão, multiplicação, seno, cosseno, tangente, raiz quadrada, exponencial, seno inverso, cosseno inverso e tangente inverso, além de ser capaz de informar valores constantes como o Π e gerar números aleatórios, entre outros [Java Math Expression Evaluator - Versão 1.01 2001].

O funcionamento da API ***Java Math Expression Evaluator*** fará com que o protótipo receba uma *string*, ou seja, uma expressão aritmética extraída do XML, e ao recebê-la não faz nenhuma análise sintática ou a separa em partes, visto que essa análise e avaliação das expressões ficará a cargo da API. Essa expressão recebida poderá ser composta por dois tipos de expressões a serem calculadas, onde o primeiro apenas trabalha fazendo cálculos de valores (por exemplo, `A=10 + 20` ou `A=10`) e o segundo tipo é responsável por fazer cálculos entre variáveis (por exemplo `A=J + D`) ou entre variáveis e valores (por exemplo, `A=D + 10` ou `A=10 + D`).

6.2.3 Fase de Instanciação

A segunda fase que compõe o processamento desse protótipo, ou seja, a **Fase de Instanciação**, recebe como entrada o arquivo XML obtido como resultado da **Fase de Poda** em conjunto com informações de múltiplas bases de dados geradas em MySQL [MySQL Database 2006].

Conforme mostra a Figura 42, essa fase terá o objetivo de buscar todas as informações que serão providas das bases de dados e que por conta disso, serão exclusivas a cada documento personalizado. Baseado nisso, pode-se dizer que a **Fase de Instanciação** será responsável por gerar as diferentes versões dos documentos personalizados, com as características (como nome, endereço, telefone) de cada cliente e preferências e informações que alteram-se para os mesmos (por exemplo, valores de viagens que mudam conforme a classe social do cliente), tornando-os documentos finais personalizados.

Com esse objetivo, os formalismos que compõem essa fase (mostrados em negrito) e as tarefas que essa executará sobre esses formalismos, serão:

1. Inserção dos **conteúdos variáveis** que têm seus conteúdos ou informações providos das bases de dados;

2. Geração de múltiplos arquivos XML (`.dia`), com informações e preferências de cada cliente.

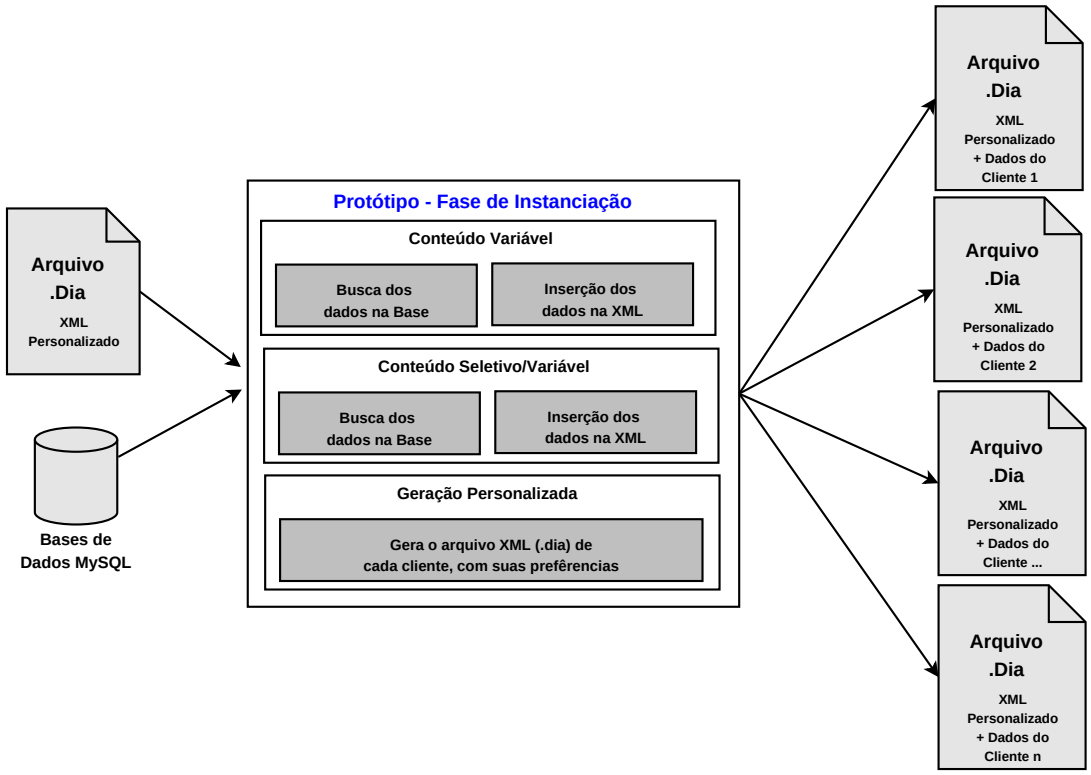


Figura 42 – Principais tarefas executadas pela **Fase de Instanciação**.

Para que o protótipo implementado em JAVA busque dados e informações dos conteúdos variáveis e faça a sua inserção nas folhas, gerando diferentes versões personalizadas dos documentos, foi necessário a utilização de um conjunto de classes e interfaces (API) para trabalhar com banco de dados, conhecida como *Java Database Connectivity* (**JDBC**), escritas também em JAVA e que fazem o envio de instruções SQL para qualquer banco de dados relacional [Microsystems 2006].

Essa conexão entre o banco de dados e o protótipo, através da interface **JDBC**, só acontece se houver a utilização de um driver específico e adequado para conectar-se a um tipo de banco de dados relacional (como o MySQL) [Ramon 2000].

O banco de dados MySQL foi utilizado para armazenar as informações variáveis do protótipo, e dentre os drivers **JDBC** existentes, o driver nativo *open-source* conhecido por **MySQL Connector/J** seria o utilizado [MySQL Connector/J 2006]. Esse terá a finalidade, no protótipo, de interagir com a base de dados MySQL buscando, extraindo e inserindo nos documentos, os diferentes conteúdos variáveis e seletivos/variáveis, através de conversões das chamadas geradas pela interface **JDBC** no protocolo de rede que utiliza a base de dados MySQL.

Após todos os conteúdos variáveis e seletivos/variáveis, que são providos de múltiplas bases de dados MySQL, forem extraídos por intermédio dessa API e inseridos nos documentos personalizados acontecerá o término dessa **Fase de Instanciação**, obtendo-se a geração de vários arquivos XML personalizados, também no formato padrão do **DIA**, permitindo assim que esses documentos obtidos possam ser relidos novamente na ferramenta **DIA** para que os desenvolvedores possam visualizar as alterações ocorridas, analisando as mudanças das características e

preferências de cada um dos usuários.

6.3 Outra abordagem de implementação

A implementação do protótipo baseia-se na idéia de trabalhar com duas fases que fazem escolhas e podas na árvore uma única vez, de forma genérica para todos os usuários e a partir dessa árvore podada é que são alterados os diferentes dados e informações exclusivas gerando vários documentos personalizados para cada um dos clientes. Nesse caso a base de dados não afeta as escolhas de quais conteúdos seletivos devem ou não ser inseridos, e só provém dados e informações para os conteúdos variáveis, completando textos com dados de um cliente, produtos ou viagens exclusivas, preços dependendo dos passeios que o cliente irá conhecer ou qualquer informação que alteram-se em todos os documentos gerados.

Além dessa abordagem escolhida, a qual baseia-se em parâmetros de entrada fornecidos de forma genérica aos clientes, e que foi implementada para ser aplicada com experimentos baseados em documentos reais (conforme será mostrado no próximo capítulo), existe outra abordagem que poderia ter sido utilizada nessa implementação, a qual seria baseada no fornecimento dos parâmetros de entrada de forma individual para cada um dos clientes, ou seja, através do **método de obtenção exclusiva** (demonstrado na Seção 4.7.4).

Nessa nova abordagem, demonstrada na Figura 43, o modelo receberia como entrada o mesmo arquivo XML com a árvore gerada e transformada no **DIA**, só que o conjunto de parâmetros de entrada seria provido de uma base de dados, onde cada cliente teria os seus parâmetros exclusivos. Nesse caso, a base de dados passaria a afetar as escolhas e podas feitas na primeira fase sobre a árvore, fazendo com que a **Fase de Poda** e **Instanciação** fossem executadas múltiplas vezes para cada um dos parâmetros de entrada.

Para simplificar o funcionamento dessa abordagem, evitando que a **Fase de Poda** fosse múltiplas vezes executada quando os parâmetros de entrada entre dois ou mais usuários fossem iguais, seria necessário que a base de dados passasse por uma mineração com ordenação, onde todos os clientes com parâmetros de entrada iguais ficassem agrupados.

Antes da **Fase de Poda**, a cada execução a essa abordagem testaria se o parâmetro de entrada de cada cliente é igual ao anterior. Caso ele seja, a abordagem passa a executar direto a **Fase de Instanciação** só alterando os dados e informações do cliente sobre a árvore já podada para o cliente anterior que possui parâmetro igual. Porém, se o parâmetro de entrada for diferente, o protótipo irá refazer a **Fase de Poda** sobre a mesma árvore inicial, só que com o parâmetro de entrada diferente, para após passar para **Fase de Instanciação**.

Essa nova abordagem aumentaria o grau de personalização dos documentos, mas em contrapartida aumentaria tempo de execução até obter-se a geração de todos os documentos personalizados, deixando o processamento mais lento. Além disso, nessa possível abordagem o funcionamento dessas duas fases bem como os formalismos e comportamentos deles serão iguais ao da abordagem implementada na prática, utilizando parâmetros de entrada genéricos, provando que a validação desse modelo idealizado é viável.

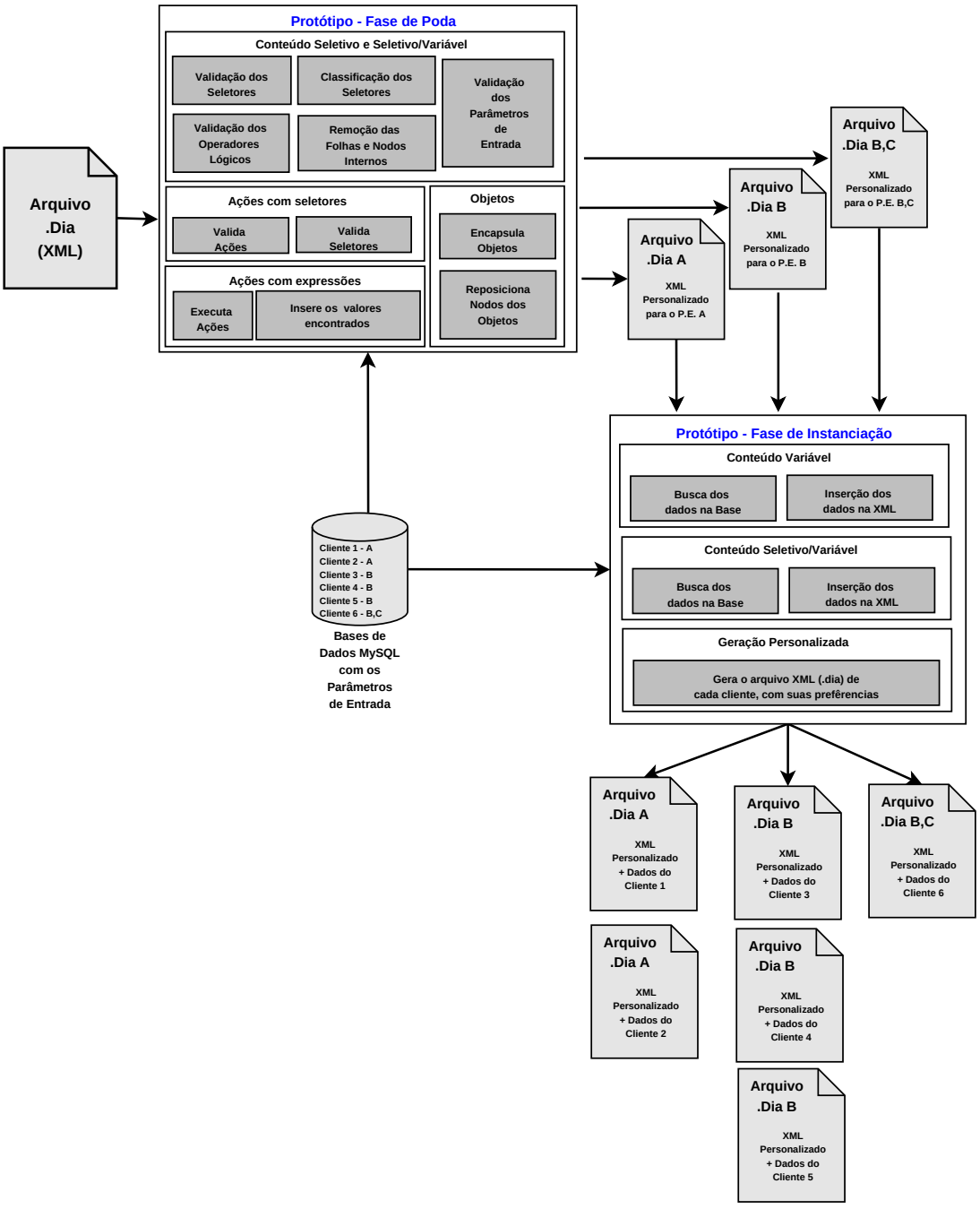


Figura 43 – *Workflow* de outra abordagem de possível implementação.

7 Experimentos e Resultados

Para validar os formalismos estipulados para o modelo, bem como o protótipo desenvolvido e apresentado no Capítulo 6, mostrando que esse é capaz de gerar documentos personalizados, foram realizadas criações de algumas árvores no **DIA** baseadas em exemplos de documentos de casos reais, as quais serão aplicadas a esse protótipo.

Através desses estudos de caso desenvolvidos para mostrar que o protótipo é apto a gerar documentos personalizados, será obtida uma avaliação sobre o desempenho desse sistema e uma análise em relação ao tempo de processamento atingido ao término de cada fase que compõe o protótipo para a geração de várias quantidades de documentos.

Neste capítulo serão apresentados os documentos de caso de uso que servirão de base para essa fase experimental, onde além de apresentar as árvores iniciais dos mesmos, serão mostradas instâncias personalizadas com os conteúdos variáveis inseridos, obtidas após o processamento do protótipo (Seção 7.1). Após isso, serão apresentados os resultados dos experimentos realizados sobre esses documentos, tornando-se possível a avaliação do desempenho do protótipo (Seção 7.2).

7.1 Documentos

Para validar o funcionamento do protótipo foram desenvolvidos três documentos baseados em exemplos reais: um folheto de propaganda de viagens, um documento impresso para matrícula de uma faculdade e um jornal com cadernos e notícias personalizadas conforme as preferências de cada leitor.

Esses documentos foram aplicados ao protótipo implementado na linguagem JAVA para que se obtivesse os documentos personalizados para determinados grupos de clientes. Além disso, essas árvores representativas dos documentos sofreram algumas duplicações, gerando assim árvores cada vez maiores, para que a avaliação de desempenho, que será demonstrado na Seção 7.2, pudesse ser realizado.

Poderão ser analisadas nas seções a seguir as informações e características desses três documentos desenvolvidos em conjunto com suas árvores geradas no **DIA**, determinando o que é possível gerar através delas e uma amostra de uma versão personalizada dos mesmos para um cliente específico.

7.1.1 Folheto de Propaganda

Dentre os três documentos que compõem os casos de uso para validar o protótipo está o documento que representa um folheto para propaganda de viagens para cinco destinos diferentes: Nova Iorque, Porto Seguro, Fortaleza, Canadá, Europa e Buenos Aires.

Ao gerar documentos personalizados a partir desse estudo de caso (que tem sua árvore mostrada na Figura 44), o desenvolvedor poderá gerar três tipos de folhetos diferentes em relação ao tipo de informação nas versões personalizadas:

1. um folheto contendo apenas os conteúdos relacionados às viagens oferecidas;
2. um folheto apresentando apenas os preços de um determinado conjunto de destinos;
3. um folheto que junta as informações apresentadas nos dois folhetos anteriores, ou seja, informando ao seu cliente tanto as informações sobre as viagens quanto seus preços.

O folheto de propaganda de viagens (Figura 44) é composto inicialmente por um cabeçalho que irá trazer as informações estáticas da empresa (logotipo, nome, endereço, CEP, cidade, estado, telefone e endereço eletrônico da mesma), que serão idênticos em todas as versões personalizadas do documento, seguido de informações sobre o usuário (como nome e endereço) que variam em todos os documentos e que são anexadas a textos que mudarão conforme o tipo de conteúdo. Os textos, marcados em laranja na Figura 44, mudarão quando o documento for formado por:

- **viagens ou pelo conjunto de viagens e preços das mesmas:** um texto com nome do cliente seguido de uma ação para mudar dinamicamente o número de destinos (`Olá !!nome, venha conhecer alguns dos nossos &D destinos!`);
- **preços:** um texto com o nome do cliente (`Olá !!nome, venha conhecer nossos destinos!`).

Independentemente do texto escolhido, o mesmo será seguido pelo endereço do cliente, que varia a cada versão do documento.

A próxima ramificação apresenta um seletor para determinar se suas sub-árvores devem ou não ser exploradas, onde o protótipo escolhe se as informações referentes às viagens devem ou não ser mantidas nas personalizações finais. Nas sub-árvores descendentes desse seletor estão contidos as cinco possíveis ofertas a compor o documento, onde apenas a sub-árvore referente a Buenos Aires não compõe a árvore principal, e sim é abordada como um objeto (Figura 45(a)), que ao ser explorado é encapsulado nessa árvore principal.

As sub-árvores das cinco possíveis viagens são todas compostas por um texto sobre cada destino, que será igual a todos os usuários com as mesmas preferências, além de um texto sobre um passeio que o cliente irá realizar caso faça a viagem e uma imagem de um dos pontos turísticos mais conhecidos da região, os quais se alterarão a cada documento personalizado já que serão providos de uma base de dados.

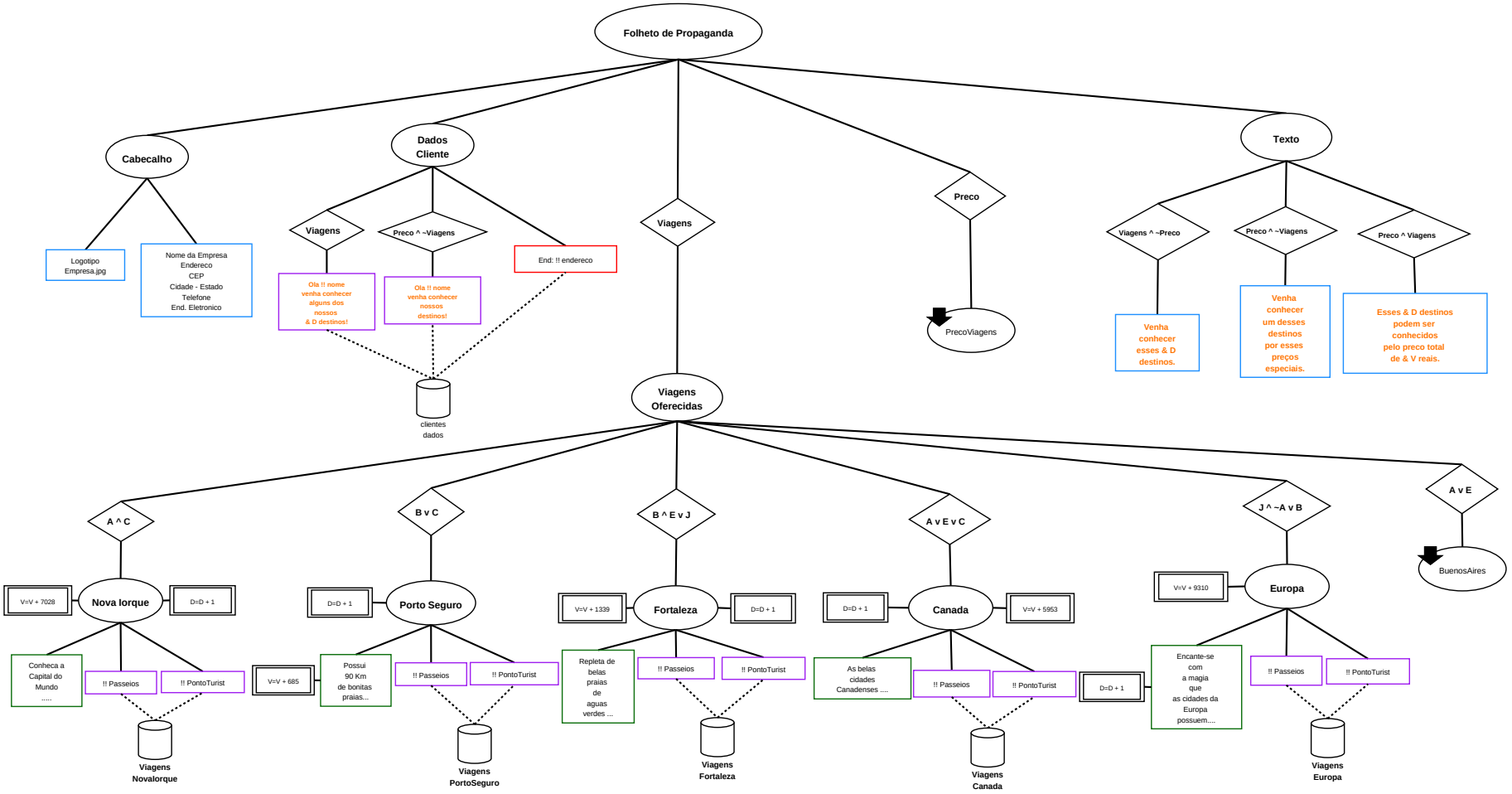
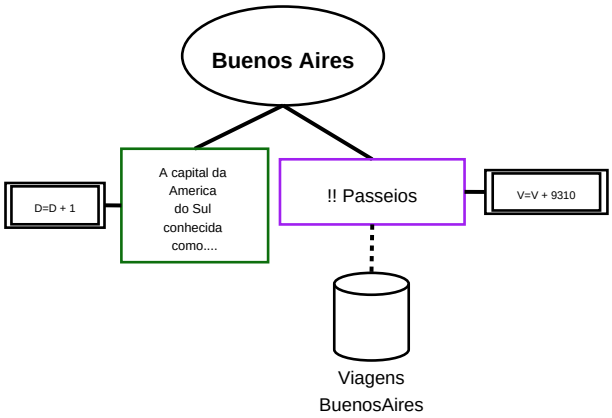
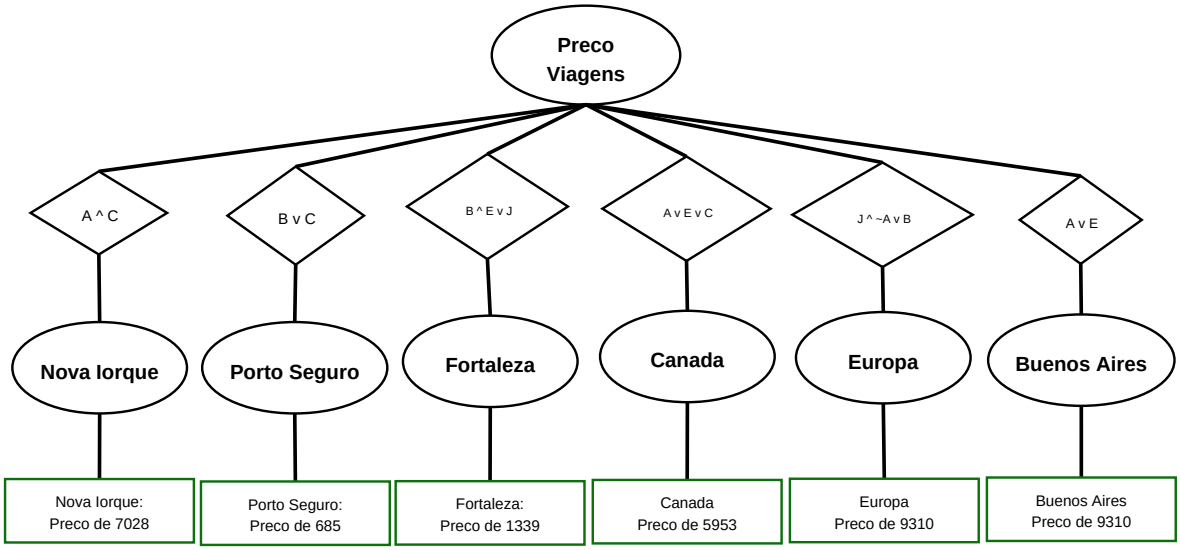


Figura 44 – Árvore representativa do estudo de caso **Folheto de Propaganda**.

Já na ramificação seguinte o mesmo processo de escolha feito para as viagens será feito para os preços desses destinos, visto que essa apresenta um seletor para determinar se suas sub-árvores devem ou não compor o documento final personalizado. Na sub-árvore desse seletor está contido um outro objeto (Figura 45(b)), que em suas ramificações conterá textos classificados como conteúdo seletivo com os preços referentes a cada viagem. Caso essa ramificação seja explorada esse objeto será encapsulado na árvore principal e irá acrescentar os preços para um determinado grupo de destinos escolhidos baseados nos parâmetros de entrada.



(a) Objeto Buenos Aires



(b) Objeto com os preços das viagens

Figura 45 – Objetos do experimento Folheto de Propaganda.

Para fechar a inserção de conteúdos ao documento, a última ramificação irá trazer frases (demonstradas na Figura 44 em laranja) que finalizarão o folheto de propaganda, as quais se alterarão conforme o tipo de conteúdo que o documento contenha, onde se o documento for composto por:

- **Viagens:** irá adicionar um texto com uma ação, que trocará automaticamente o número de viagens oferecidas (`Venha conhecer esses &D destinos.`);

- **Preços:** irá adicionar um texto estático (`Venha conhecer um desses destinos por esses preços especiais.`);
- **Viagens e Preços:** irá adicionar um texto seguido de duas ações, onde a primeira irá trocar dinamicamente o número de viagens oferecidas e a segunda alterará a soma total dos preços desses destinos (`Esses &D destinos podem ser conhecidos pelo preço total de &V reais.`).

A árvore estruturada no DIA foi aplicada ao protótipo para gerar diferentes documentos personalizados para clientes, baseado em um conjunto de 87 combinações entre os parâmetros de entrada. Uma dessas versões personalizadas obtidas para um determinado cliente com seus dados personalizados já inseridos, gerada para esse documento, pode ser vista na Figura 46, a qual foi obtida com base nos parâmetros `Viagens,Preco,A,J`.

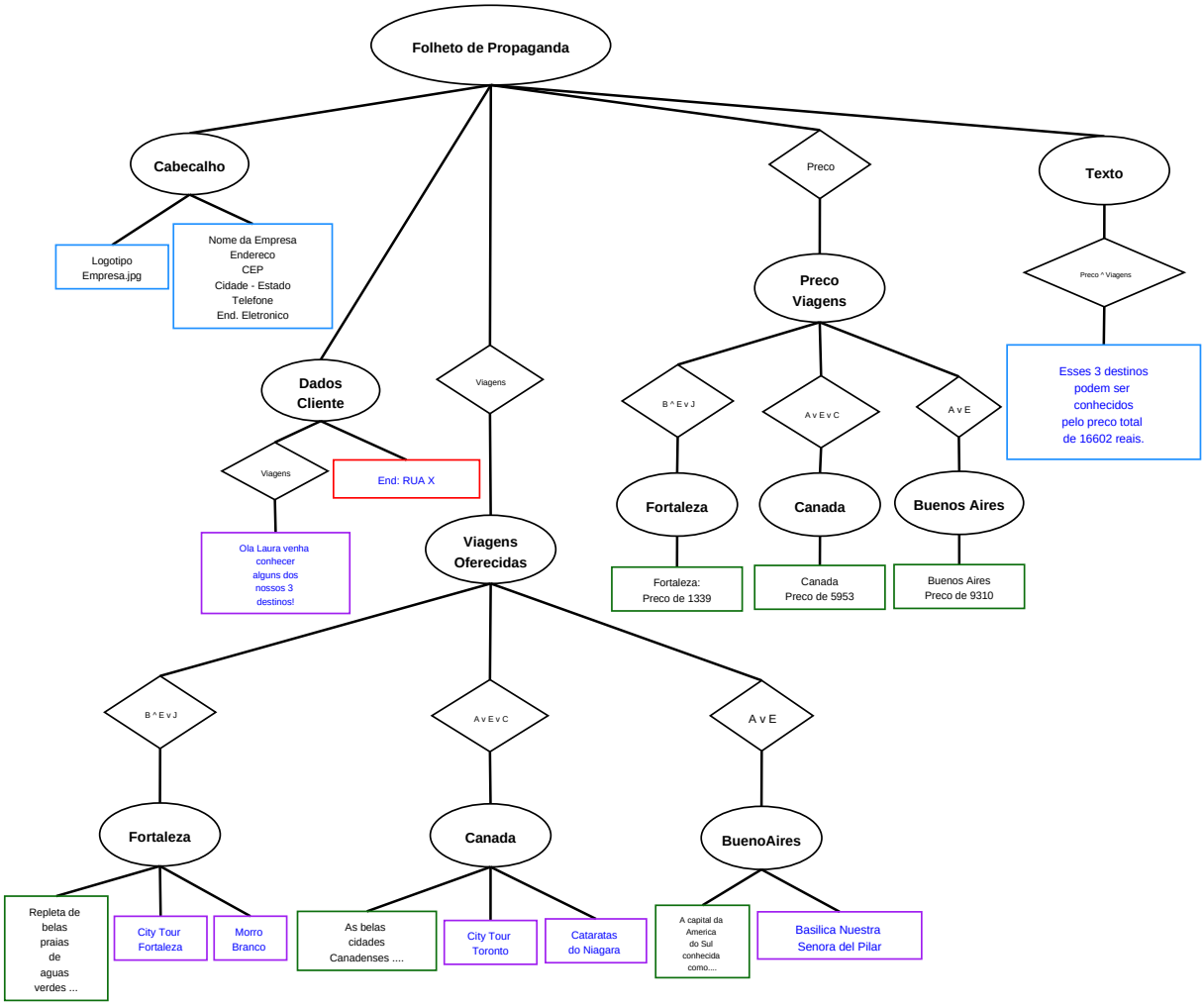


Figura 46 – Documento personalizado do Folheto de Propaganda para os parâmetros de entrada `Viagens,Preco,A,J`.

Além disso, na Figura 46 pode ser visto que alguns nodos têm seus textos representados em azul. Isso ocorre porque esses dados sofreram alguma alteração durante o processamento do protótipo, já que foram providos de uma base de dados (no caso dos conteúdos variáveis, como

o nodo com o nome e endereço do cliente) ou porque sofreram influências e foram completados através de valores contabilizados através do formalismo de ações.

7.1.2 Documento de Matrícula

O segundo documento desenvolvido para ser aplicado ao protótipo e validar o seu funcionamento, foi criado baseando-se em um documento de uma universidade que comprove e realize a matrícula dos alunos nas disciplinas que o mesmo irá cursar em um semestre.

O desenvolvedor poderá gerar versões desse documento de matrícula para cursos disponíveis para a graduação ou para a pós-graduação (mestrado ou doutorado) que essa universidade oferece. Dentre esses cursos que podem ter seus documentos de matrícula gerados de forma personalizada para cada aluno estão: sistemas de informação, direito, ciência da computação e administração (cursos na área de graduação); administração, psicologia, ciência da computação e direito (cursos na área de mestrado); e comunicação social, ciência da computação, letras e história (cursos na área de doutorado).

Com base na árvore desse estudo de caso, apresentada na Figura 47, esse documento começa pelos conteúdos estáticos referentes ao logotipo e aos dados da universidade (como nome, endereço, telefone, CEP, estado e endereço na web), seguidos dos dados informativos de cada faculdade (como nome da faculdade, prédio, telefone, número do fax e endereço da página web) que disponibiliza esses cursos de graduação, mestrado ou doutorado, os quais alteram-se a cada documento conforme o curso que está emitindo esse documento.

A próxima ramificação irá trazer as informações sobre cada aluno, apresentando o seu nome e endereço providos de uma base de dados e que irão alterar-se em todas as versões do documento, seguidos de um objeto que permite definir, através dos parâmetros de entrada, o curso que aluno está efetuando a matrícula (que pode ser analisado no Anexo A). Além disso, através dessa ramificação, se o aluno está matriculado nos cursos oferecidos nas áreas de mestrado ou doutorado, nesta parte do documento será adicionado o nome do orientador responsável por esse estudante.

Após isso, será inserido um texto sobre as matrículas idêntico em todas as versões do documento. Na sub-árvore seguinte estará disponível um conjunto de objetos referentes a cada curso oferecido (detalhados no Anexo A), que apresentarão as disciplinas em que o aluno está se matriculando naquele semestre, as quais mudam para cada curso e principalmente para cada aluno de um mesmo curso, e por isso serão providas de uma base de dados.

Esses objetos referentes às disciplinas só serão encapsulados caso os seletores posicionados antes desses sejam validados como verdadeiros com base nos parâmetros de entrada, e ao serem encapsulados será possível obter as informações sobre as disciplinas, determinando em quais bases de dados e tabelas essas serão providas para o protótipo ir buscá-las.

Por fim, o documento de matrícula será composto por um texto que irá se alterar dinamicamente conforme o número de disciplinas que o aluno está se matriculando (`O aluno está apto a fazer essas &N disciplinas.`).

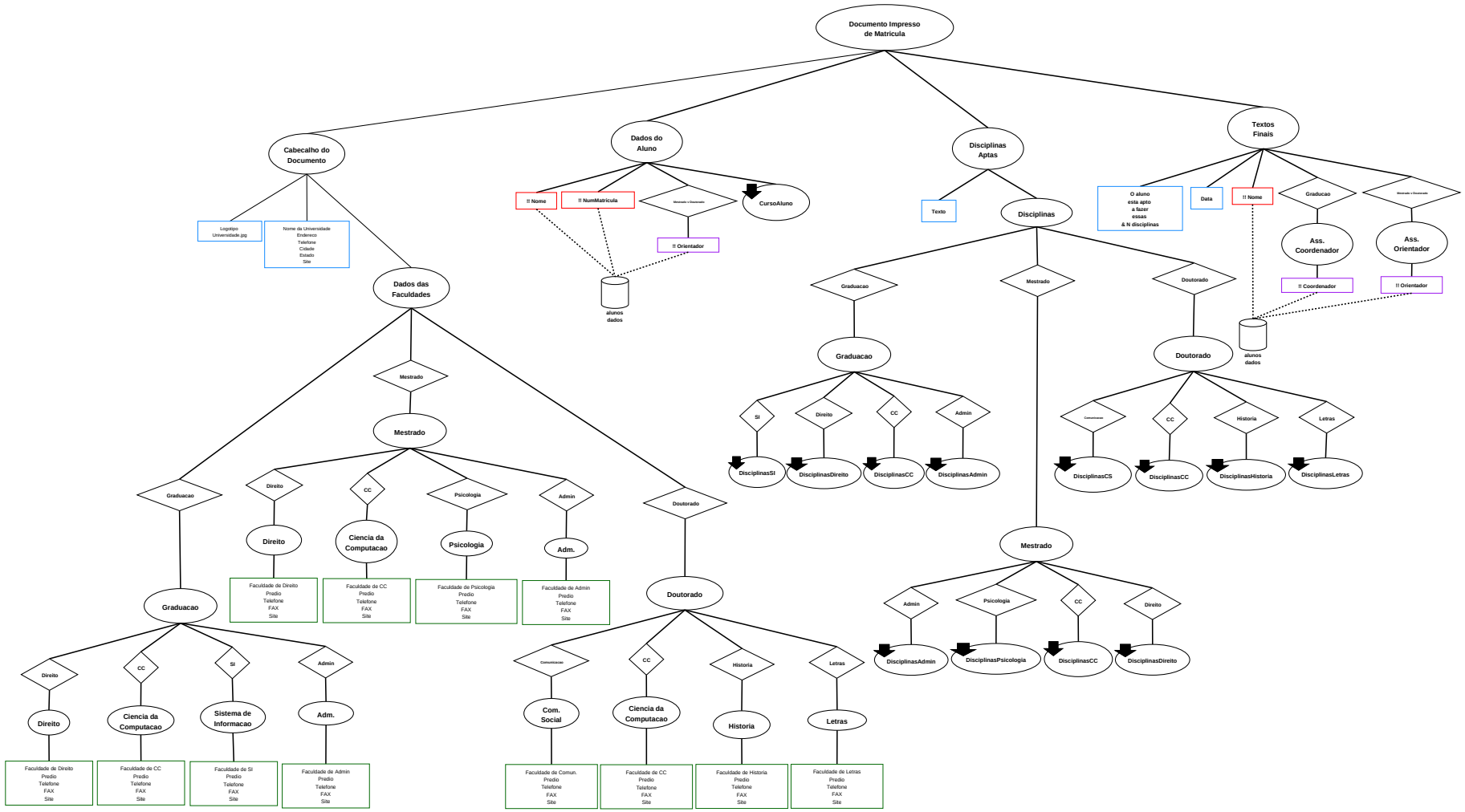


Figura 47 – Árvore representativa do estudo de caso **Documento de Matrícula**.

Caso o aluno esteja cursando um curso de graduação, o documento será composto pelo nome do coordenador do curso no lugar onde esse irá assinar, e caso esteja cursando um curso de mestrado ou doutorado, o documento será composto pelo nome do orientador do aluno, no lugar onde esse irá assinar.

Após a caracterização e criação desse documento, o mesmo foi processado no protótipo para gerar diferentes documentos de matrículas personalizados para vários alunos, baseado em um conjunto de 12 parâmetros de entrada, visto que para esse exemplo não seria muito útil fazer várias combinações entre os parâmetros, porque um aluno se matriculará em apenas um curso por semestre. Baseado nos parâmetros de entrada `Mestrado,CC` (ciência da computação) e aplicando esse documento ao protótipo teremos um documento gerado para um aluno somente com as informações desse curso, conforme pode ser visto na Figura 48.

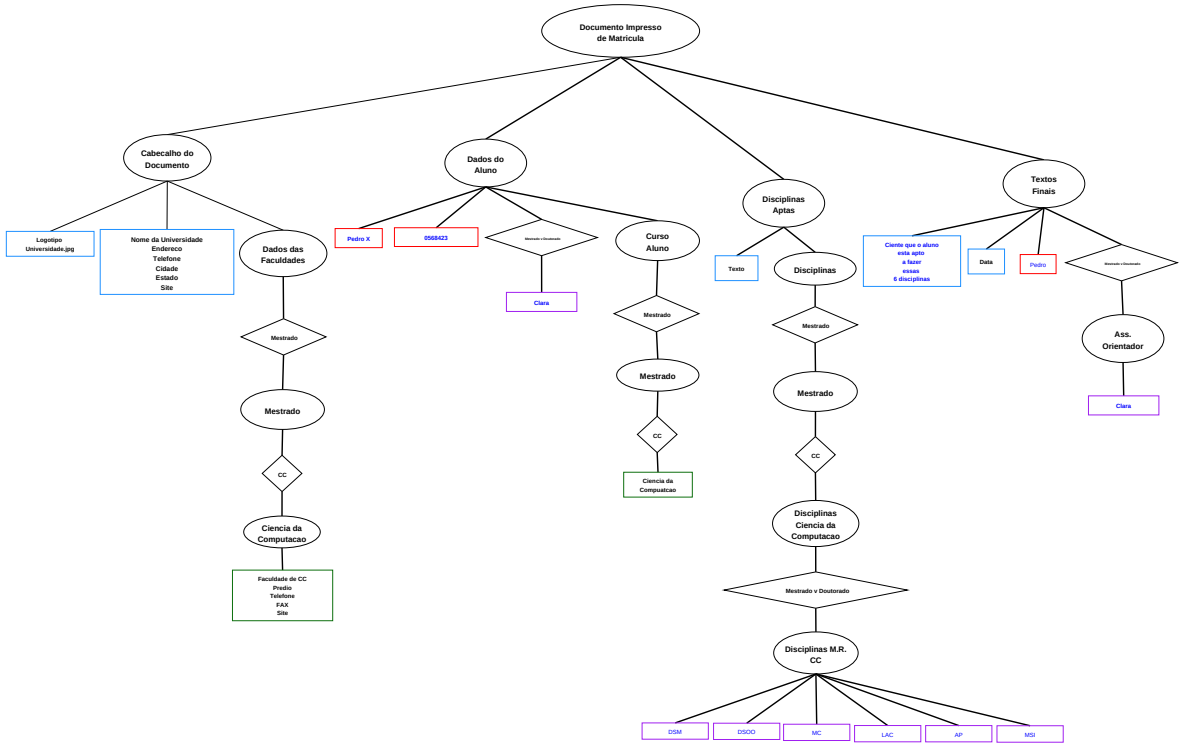


Figura 48 – Documento personalizado do Documento de Matrícula para os parâmetros de entrada `Mestrado,CC`.

7.1.3 Jornal

O último documento criado para validação do protótipo representa um jornal que tem como principal objetivo trazer ao seu leitor notícias personalizadas conforme as suas preferências e principalmente ser composto por vários cadernos personalizados, que o próprio leitor escolhe conforme suas necessidades. Os cadernos disponíveis que o leitor pode escolher para compor o seu jornal personalizado podem ser analisados na Tabela 18.

Ao desenvolver a árvore desse jornal personalizado, o qual pode ser analisada através da Figura 49, foi determinado que essa seria estruturada em três partes principais, onde a primeira

é a capa, a segunda parte é formada por um conjunto de cadernos personalizados que se alteram conforme o leitor deseja e a terceira parte idealizada como a sua contracapa.

Nome do Caderno	Nome do Caderno	Nome do Caderno
Policial	Esportivo	Carros
Econômico	Notícias do mundo	Imóveis
Político	Notícias gerais	Informática
Notícias do estado	Notícias da cidade	Educação
Vestibular	Mundo digital	Lazer
Cultural	Viagens	Emprego

Tabela 18 – Tabela mostrando os possíveis cadernos que podem compor um jornal personalizado.

A capa será iniciada por dados estáticos que aparecem iguais em todas as versões do documento, como o logotipo e data em que o jornal é emitido, seguido de um texto inicial onde será anexado o nome do leitor de cada documento (`Bom dia !!nome`), seu endereço e CEP. Após isso, ainda compondo a capa, estarão contidas duas principais notícias gerais ocorridas durante os últimos dias com suas respectivas fotos, as quais irão variar para cada um dos leitores e por isso são providas por uma base de dados.

Finalizando a capa há uma ramificação que apresenta um objeto (demonstrado no Anexo B), o qual terá em suas sub-árvores conteúdos seletivos, que ao serem validados, irão inserir no documento principal os cadernos que irão compor o jornal de cada leitor.

Nas páginas internas, o desenvolvedor poderá escolher os cadernos de notícias que irão compor o jornal de cada grupo de clientes, representados através de objetos, que ao serem validados fazem com que o protótipo encapsule as suas sub-árvores na árvore principal, adicionando ao documento notícias específicas a cada caderno. Os objetos e a classificação das suas notícias podem ser analisados no Anexo B deste trabalho.

Após os cadernos personalizados, a árvore principal apresenta uma sub-árvore que trabalha com a inserção dinâmica de outros cadernos. Dentre esses cadernos que serão adicionados dinamicamente e sem a influência direta dos parâmetros de entrada estão: caderno com classificados, caderno sobre cinema e um caderno com notícias sobre o Brasil e o exterior.

O caderno com classificados só será adicionado dinamicamente caso um desses cadernos sejam inseridos nas versões personalizadas: caderno de economia, o caderno de emprego, o caderno sobre o mundo digital, o caderno de informática, caderno sobre carros ou sobre imóveis. No interior desse caderno serão apresentados produtos diversos a serem vendidos, como carros e imóveis.

Se os cadernos sobre o mundo digital ou sobre informática estiverem contidos entre as preferências de um determinado leitor, além de carros, imóveis e produtos diversos, estarão contidos produtos eletrônicos para a venda.

Já o caderno sobre cinema será adicionado nos jornais personalizados quando os cadernos de cultura ou lazer fizerem parte das preferências do leitor e por esse motivo será mantido na versão personalizada do jornal, e o caderno com notícias do Brasil e do exterior será adicionada somente caso os cadernos sobre o estado ou cidade compuserem o jornal.

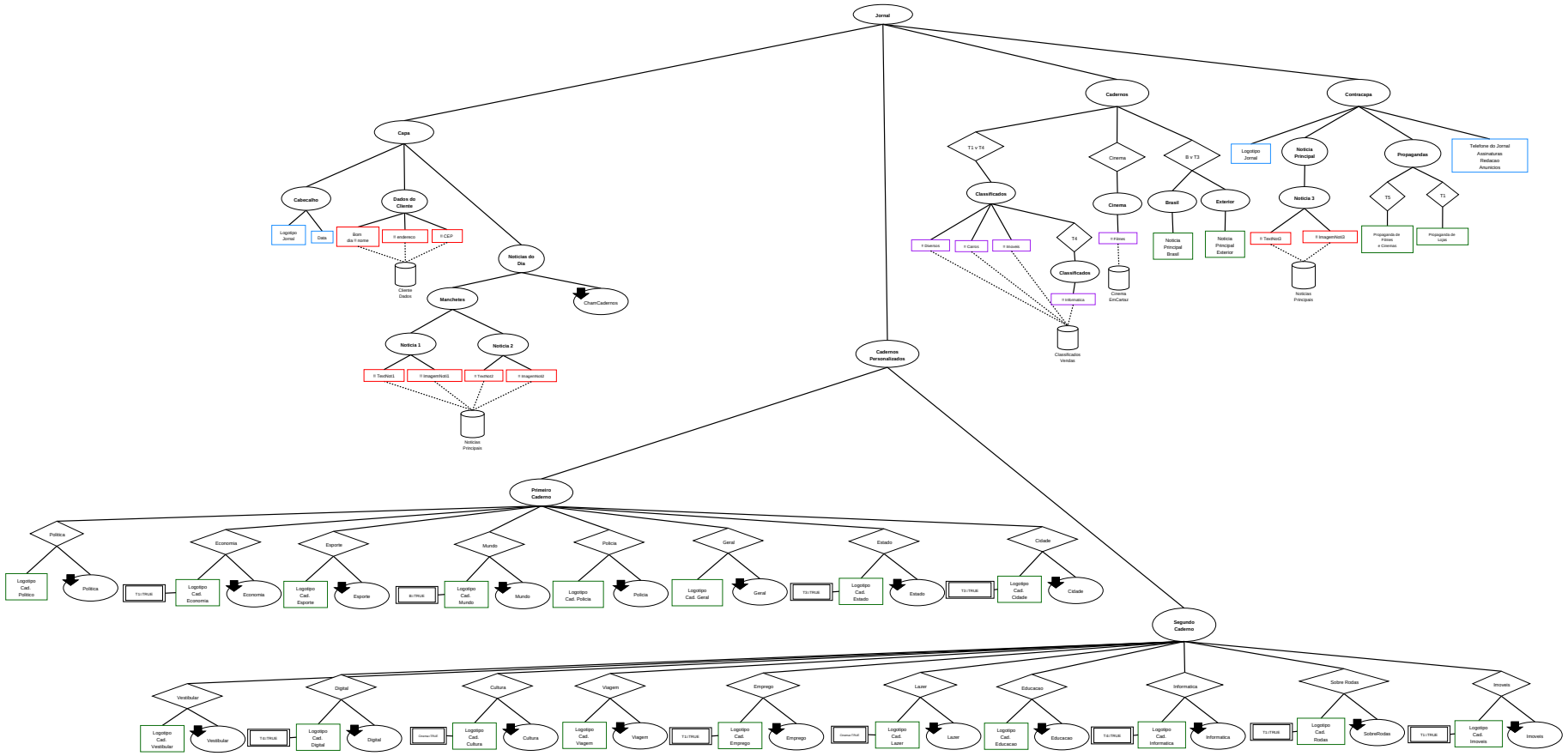


Figura 49 – Árvore representativa do estudo de caso **Jornal Personalizado**.

Por fim, na contracapa estarão contidas informações que serão iguais em todas as versões, como o logotipo (assim como na capa), com informações sobre os telefones para assinaturas, da sua redação e da central de anúncios. Além disso, essa contracapa conterá uma notícia principal personalizada que mudará dependendo do cliente, além de propagandas sobre filmes em cartaz e sobre cinemas, caso o caderno de lazer ou cultura tenha sido inserido nas versões personalizadas e sobre lojas de informática, caso o caderno sobre o mundo digital e sobre informática formarem parte do jornal do cliente.

Baseado em 16 combinações entre os parâmetros de entrada foi possível gerar documentos personalizados para diferentes leitores. Uma dessas versões personalizadas para um determinado leitor, com o conjunto de cadernos escolhidos especificamente para ele, pode ser analisada na Figura 50, a qual foi obtida com base nos parâmetros `Digital,Cultura,Viagem,Lazer,Imoveis`.

7.2 Resultados

Baseado nos documentos apresentados anteriormente e aplicando-os ao protótipo, sobre os diferentes conjuntos de parâmetros de entrada, foi possível realizar dois experimentos, os quais serão a seguir demonstrados, com o intuito de avaliar os tempos de processamento para diferentes versões do documento.

A obtenção desses documentos e a coleta de tempos de processamento para ambos os experimentos foram realizados sobre um computador Pentium 4 de 3.06 GHz com 512 MB de memória.

7.2.1 Primeiro Experimento

O primeiro experimento realizado tem como principal objetivo recolher os tempos de processamento do processo de geração de documentos personalizados, tanto na **Fase de Poda** quanto na **Fase de Instanciação** do protótipo sobre as três árvores dos documentos mostrados anteriormente.

Por meio desse experimento deseja-se provar que este protótipo, implementado a partir do modelo formalizado, neste trabalho, é capaz de gerar várias instâncias de um documento com alto grau de personalização em um tempo razoável. Além disso, deseja-se mostrar também qual dessas duas fases que compõem o protótipo é a que acarreta um maior tempo de processamento levando o desempenho do mesmo a tornar-se a parte mais cara do *workflow*, para que dessa forma, se consiga em trabalhos futuros, melhorar esse problema através de novas técnicas.

Para que se conseguisse obter esses resultados esperados sobre esse experimento foi necessário realizar no protótipo o processamento das seguintes árvores:

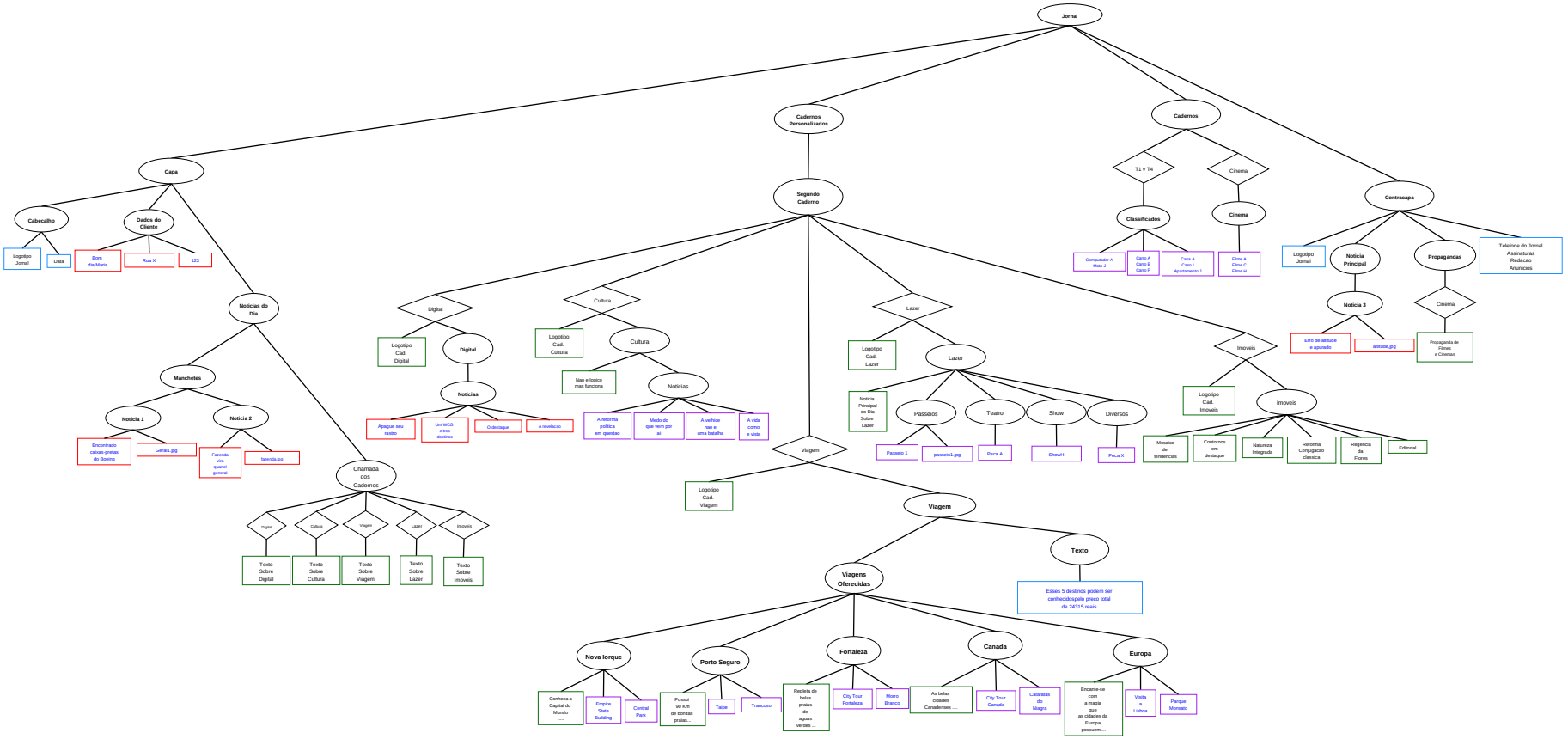


Figura 50 – Documento personalizado do Jornal para o parâmetro de entrada `Digital,Cultura,Viagem,Lazer,Imoveis`.

- **folheto de propaganda:** foram gerados diferentes documentos personalizados e recolhidos os tempos baseados em 87 conjuntos de parâmetros de entrada, como por exemplo, `Viagens,A,B,C`,
  `Preco,B,C` e `Viagens,Preco,A`;

- **documento de matrícula:** foram gerados documentos personalizados e recolhidos os tempos de processamento de ambas as fases do protótipo para 12 conjuntos de parâmetros de entrada, como por exemplo, `Graduacao,Direito`, `Mestrado,CC` e `Doutorado,Historia`;

- **jornal:** foram gerados documentos personalizados e recolhidos os tempos de ambas as fases do protótipo separadamente para 16 conjuntos de parâmetros de entrada, como por exemplo, `Geral, Estado, Cidade, Mundo`.

Para cada uma dessas combinações sobre esses três documentos foram recolhidos, separadamente, os tempos de processamento da **Fase de Poda** e **Fase de Instanciação** para vários grupos de clientes, obtendo assim a geração de 1, 10, 100, 1000 e 10000 documentos personalizados, adicionando a cada um deles características de cada cliente, providas das bases de dados MySQL. É necessário trabalhar com essa variação no número de documentos a ser gerados, nesse experimento, para que se possa assim analisar o comportamento e o desempenho da **Fase de Instanciação** do protótipo, visto que gerar várias instâncias de um documento é a principal característica para se trabalhar com conteúdos variáveis.

A partir desses tempos recolhidos em milésimos de segundo apresentamos gráficos para cada um dos documentos aplicados ao protótipo (conforme as Figuras 51, 52 e 53), que têm o objetivo de tornar mais acessível a análise dos resultados obtidos para cada uma dessas árvores e permitir a identificação de tendências. Esses gráficos foram gerados através da ferramenta ***Gnuplot***, baseados nos tempos de processamento obtidos de cada uma das árvores como foi explicado anteriormente [Gnuplot Homepage 2006].

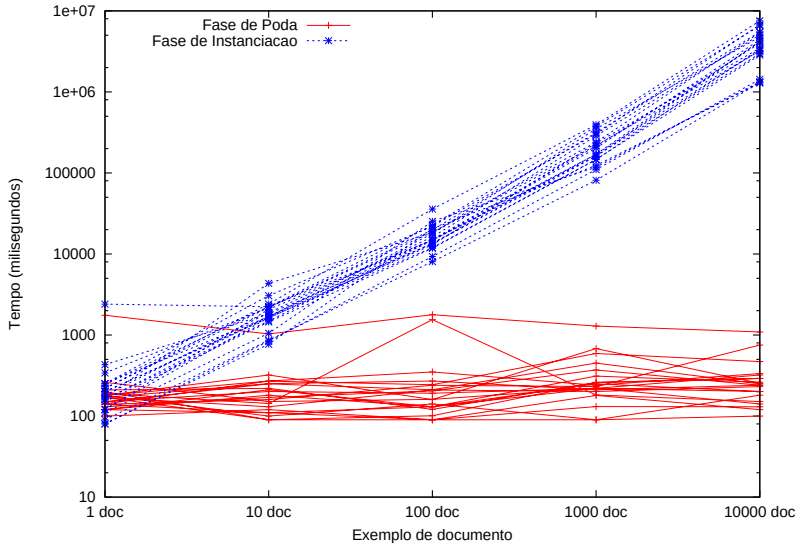


Figura 51 – Gráfico dos tempos de processamento das **Fases Poda e Instanciação** para o Folheto de Propaganda.

Para representar esses dados recolhidos foram criados gráficos de linhas, onde no eixo vertical

(eixo Y) está demarcada a variação dos tempos recolhidos em milésimos de segundo e no eixo horizontal (eixo X) estão demarcados os grupos de documentos gerados nesse experimento para diferentes números de clientes (1, 10, 100, 1000 e 10000 documentos).

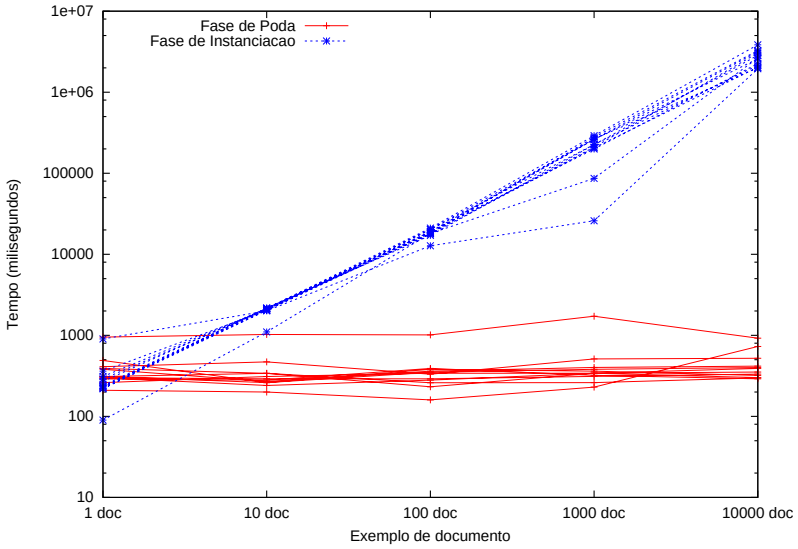


Figura 52 – Gráfico dos tempos de processamento das **Fases Poda e Instanciação** para o Documento de Matrícula.

Visto que os tempos de processamento, para ambas as fases do modelo, foram recolhidos separadamente, nesses gráficos podem ser analisados dois tipos de linhas, onde o tempo da **Fase de Poda** está demonstrado nas linhas vermelhas e o da **Fase de Instanciação** através das linhas azuis. A mudança de quantidade dos documentos gerados (1, 10, 100, 1000 e 10000 documentos) está representada nesses gráficos, através do símbolo de uma cruz (para a **Fase de Poda**) e através de asteriscos (para a **Fase de Instanciação**).

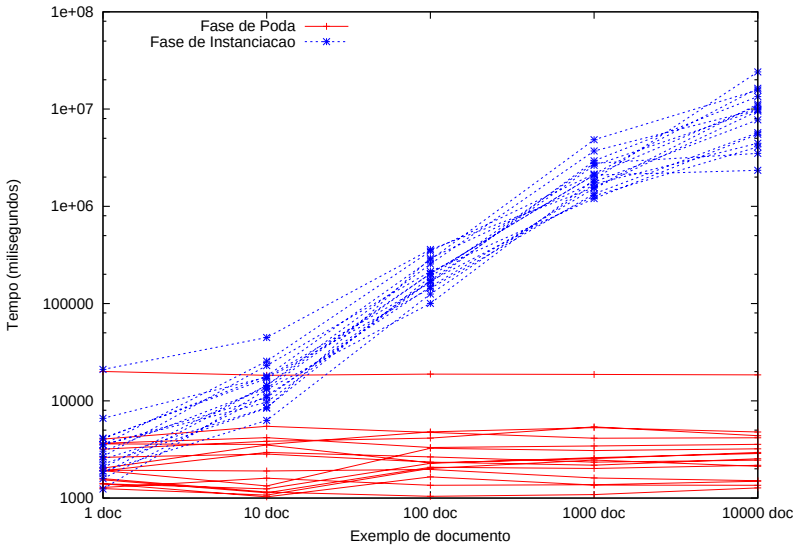


Figura 53 – Gráfico dos tempos de processamento das **Fases Poda e Instanciação** para o Jornal.

Baseado nesses três gráficos das Figuras 51, 52 e 53 e fazendo uma análise entre eles, foi possível observar que o tempo da **Fase de Poda** (representado nos gráficos em vermelho), a

qual obtém um único documento XML ainda sem os dados variáveis que será repassada para a **Fase de Instanciação**, é relativamente constante entre os diferentes parâmetros de entrada oferecidos para cada caso de uso, com pequenas variações nos tempos de processamento. O tempo de processamento para essa fase para o folheto de propaganda e documento de matrícula ficam em menos de 1 segundo para gerações sobre diferentes parâmetros e o tempo dessa fase para o último documento fica em torno de 3 segundos, custos que podem ser considerados baixos e chegando-se ao que se desejava provar, ou seja, mostrando que essa parte do *workflow* do protótipo é bastante ágil.

Porém, para gerar várias versões desses três documentos com as preferências e características de cada cliente, através da **Fase de Instanciação**, obtém-se tempos de processamento que crescem conforme aumenta a quantidade de versões de documentos personalizados a ser gerados. Isso pode ser visto, através dos tempos obtidos para a **Fase de Instanciação** sobre o parâmetro de entrada `Mestrado,CC` onde o tempo para a geração de 1 documento é 231ms, para 10 é 2143ms, para 100 é 21430ms, para 1000 é 214309ms e para 10000 é de 2143090 ms.

Além disso, pode-se concluir que fazer uma análise mais detalhada sobre o tempo obtido na **Fase de Instanciação** não é muito relevante, pois o protótipo levará o mesmo tempo gasto para obter um único documento, só que multiplicado pela quantidade de documentos estipulados naquele processo de geração, como por exemplo, se o tempo para obter um único documento é de 2674ms, quando houver a obtenção de 10 será de cerca de 26740ms. Isso porque, nesta fase, a cada novo documento a ser gerado será feita uma nova análise e percurso sobre a árvore, trocando todos os conteúdos variáveis, ou seja, no caso de 10 documentos, a árvore será percorrida 10 vezes.

Por esse motivo e analisando as linhas azuis desses gráficos é possível constatar que a **Fase de Instanciação** é a parte do protótipo que acarreta um tempo maior conforme aumenta a quantidade de documentos a ser gerados, podendo ser então considerada a parte mais complexa dessa aplicação, a qual em trabalhos futuros deverá ser aprimorada.

Apesar disso, esse protótipo foi capaz de obter as versões personalizadas dos documentos com diferentes características dos usuários entre as versões geradas, provando que na prática esse modelo e seu protótipo são aptos ao que se propõe, executando os comportamentos estipulados para cada formalismo de forma correta conforme foi proposto e formalizado, com o custo de processamento bastante baixo e com um *workflow* bastante simples.

7.2.2 Segundo Experimento

O segundo experimento realizado tem como principal finalidade recolher e analisar os tempos de processamento obtidos apenas na **Fase de Poda** baseados em algumas variantes dos documentos apresentados na Seção 7.1. Essas variantes, conforme mostra a Figura 54, foram obtidas através das árvores principais utilizadas no primeiro experimento, onde as mesmas foram duplicadas cinco vezes, para que assim, se obtivessem árvores maiores e com mais quantidade de nodos.

Por meio desse segundo experimento deseja-se determinar qual dos formalismos criados para

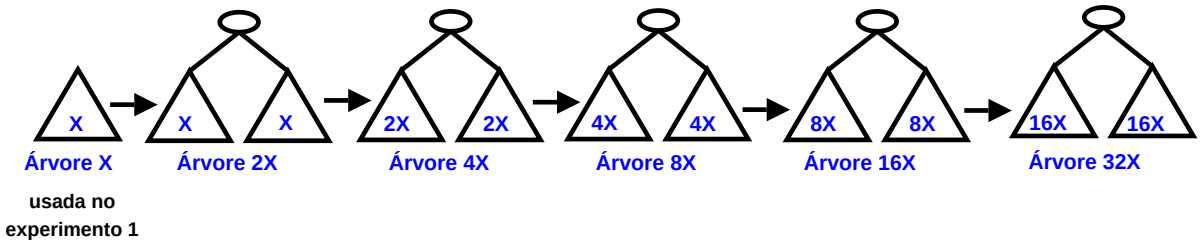


Figura 54 – Duplicação das árvores utilizadas no primeiro experimento, para que se tornasse possível a realização do segundo experimento.

o modelo e implementado nesse protótipo é o que apresenta maior tempo de execução para realizar seus comportamentos, tornando o tempo de processamento da **Fase de Poda** mais extenso. Além disso, deseja-se avaliar se uma árvore com um maior quantidade de nodos pode influenciar diretamente o comportamento e o processo de desempenho dessa ferramenta e do modelo.

Baseado no que foi demonstrado no experimento anterior, não será realizado, nesse experimento, o processamento dos documentos na **Fase de Instanciação**, isso porque os resultados que seriam obtidos nesse processo não permitem avaliar os aspectos desejados nesse experimento, pois nessa segunda fase acaba-se obtendo o tempo de percorrer o número de árvores a ser gerado.

Para que se consiga obter a comprovação do que se deseja alcançar com esse experimento, assim como foi dito anteriormente, foram utilizadas árvores com maior quantidade de nodos. Essas seis árvores duplicadas e com um maior número de nodos foram aplicadas ao protótipo para serem processadas sobre diferentes parâmetros de entrada, como:

1. **folheto de propaganda:** foram recolhidos os tempos de processamento dessas seis diferentes árvores sobre 87 conjuntos parâmetros de entrada;

2. **documento de matrícula:** foram recolhidos os tempos de processamento da **Fase de Poda** para 12 parâmetros de entrada;

3. **jornal:** forma recolhidos os tempos de processamento sobre 16 conjuntos de parâmetros de entrada.

Visto que será executada nesse experimento somente a **Fase de Poda** será obtido assim a geração de um único documento para os diferentes parâmetros de entrada de cada caso de uso criado, sem adicionar a esse documento os conteúdos variáveis referentes a cada cliente.

A quantidade de nodos referentes a cada um dos formalismos que compõem as árvores duplicadas, assim como o total de nodos que as compõem, podem ser analisados na Tabela 19. A quantidade de arestas que compõem as árvores passaram a ser contadas como formalismos, porque o XML gerado no **DIA** não demonstra quais nodos são filhos de quais, portanto é necessário validar as linhas determinando quais objetos estão ligados entre si formando as sub-árvores.

A partir desses tempos recolhidos em milésimos de segundos sobre o processamento dessas seis árvores de cada estudo de caso combinados com os diferentes conjuntos de parâmetros de entrada, foi possível desenvolver outros três gráficos (conforme as Figuras 55, 56 e 57) que fornecem a visualização dos tempos da **Fase de Poda** sobre esse experimento. Esses gráficos,

Nome do Documento	Nodos Int.	Cont. Est.	Cont. Var.	Cont. Sel.	Cont. Sel./ Var.	Seletor	Base de Dad.	Linhas	Ações	Obj.	Total de Nodos
Folheto de Propaganda X	10	5	1	5	12	13	6	71	10	2	135
Folheto de Propaganda 2X	21	10	2	10	24	26	12	143	20	4	272
Folheto de Propaganda 4X	43	20	4	20	48	52	24	287	40	8	546
Folheto de Propaganda 8X	87	40	8	40	96	104	48	575	80	16	1094
Folheto de Propaganda 16X	175	80	16	80	192	208	96	1151	160	32	2190
Folheto de Propaganda 32X	351	160	32	160	384	416	192	2303	320	64	4382
Documento de Matrícula X	27	6	2	12	3	33	2	100	0	13	198
Documento de Matrícula 2X	55	12	4	24	6	66	4	202	0	26	399
Documento de Matrícula 4X	111	24	8	48	12	132	8	406	0	52	801
Documento de Matrícula 8X	223	48	16	96	24	264	16	814	0	104	1605
Documento de Matrícula 16X	447	96	32	192	48	528	32	1630	0	208	3213
Documento de Matrícula 32X	895	192	64	384	96	1056	64	3262	0	416	6429
Jornal X	21	4	9	22	5	24	5	130	11	18	249
Jornal 2X	43	8	18	44	10	48	10	258	22	36	497
Jornal 4X	87	16	36	88	20	96	20	518	44	72	997
Jornal 8X	175	32	72	176	40	192	40	1038	88	144	1997
Jornal 16X	351	64	144	352	80	384	80	2078	176	288	3997

Tabela 19 – Tabela referente ao número de nodos que compõem as árvores duplicadas que foram utilizadas no segundo experimento.

assim como do primeiro experimento, foram obtidos através da ferramenta ***Gnuplot***, baseados nos tempos recolhidos no processamento das árvores através do protótipo [Gnuplot Homepage 2006].

Foram criados gráficos de linhas, onde no eixo vertical (eixo Y) está demarcada a variação do tempo em milésimos de segundo e no eixo horizontal (eixo X) está demarcado os diferentes parâmetros de entrada que serviram de diretrizes para a personalização de cada um dos documentos gerados. Essa variação no eixo X ocorre conforme o número de parâmetros de entrada aplicados ao protótipo sobre cada estudo de caso, ou seja, 87 variações para o primeiro gráfico (Figura 55), 12 variações para o segundo (Figura 56) e 16 variações para o terceiro (Figura 57).

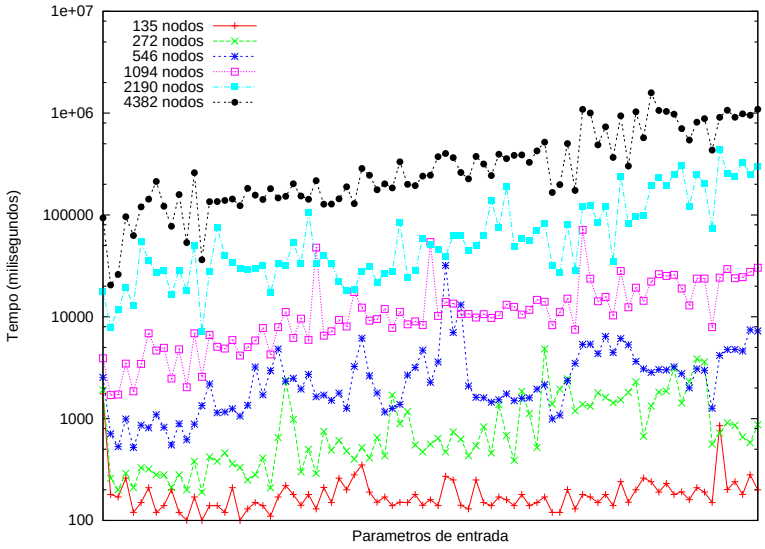


Figura 55 – Segundo experimento com os tempos de processamento da **Fase de Poda** para o folheto de propaganda sobre diferentes quantidades de nodos.

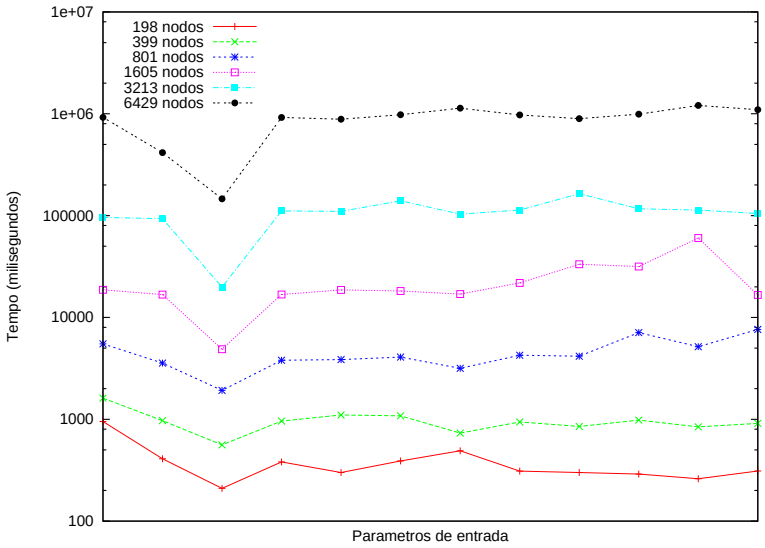


Figura 56 – Segundo experimento com os tempos de processamento da **Fase de Poda** para o documento de matrícula sobre diferentes quantidades de nodos.

A variação de cores representadas nesses gráficos, demonstram os tempos obtidos sobre as

diferentes versões dos documentos utilizadas nesse experimento, ou seja:

1. **linha vermelha:** representa a variação do tempo obtido para personalizar, sobre diferentes conjuntos de parâmetros de entrada, os documentos denominados **folheto de propaganda X** (que possui 135 nodos), **documento de matrícula X** (198 nodos) e **jornal X** (249 nodos), respectivamente em cada gráfico referente ao documento analisado;

2. **linha verde:** representa a variação do tempo obtido para personalizar, sobre diferentes conjuntos de parâmetros de entrada, os documentos denominados **folheto de propaganda 2X** (272 nodos), **documento de matrícula 2X** (399 nodos) e **jornal 2X** (497 nodos);

3. **linha azul escuro:** representa a variação do tempo obtido para personalizar, sobre diferentes conjuntos de parâmetros de entrada, os documentos denominados **folheto de propaganda 4X** (546 nodos), **documento de matrícula 4X** (801 nodos) e **jornal 4X** (997 nodos);

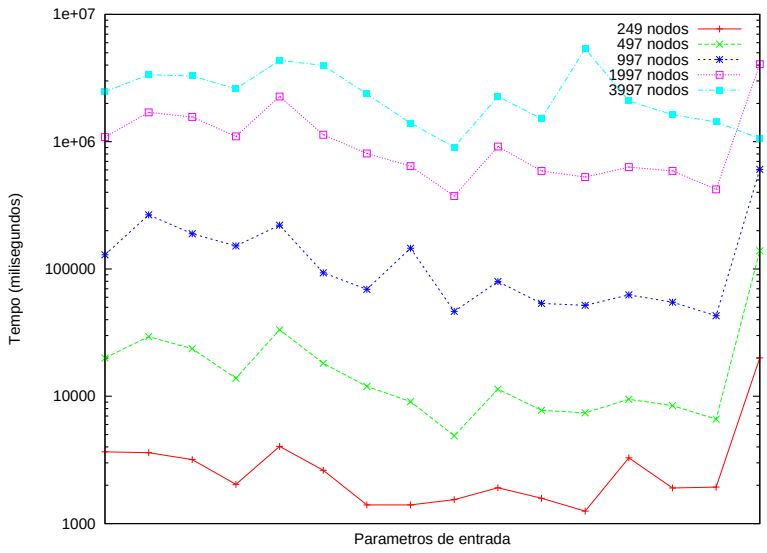


Figura 57 – Segundo experimento com os tempos de processamento da **Fase de Poda** para o jornal sobre diferentes quantidades de nodos.

4. **linha rosa:** representa a variação do tempo obtido para personalizar, sobre diferentes conjuntos de parâmetros de entrada, os documentos denominados **folheto de propaganda 8X** (1094 nodos), **documento de matrícula 8X** (1605 nodos) e **jornal 8X** (1997 nodos);

5. **linha azul:** representa a variação do tempo obtido para personalizar, sobre diferentes parâmetros de entrada, os documentos denominados **folheto de propaganda 16X** (2190 nodos), **documento de matrícula 16X** (3213 nodos) e **jornal 16X** (3997 nodos);

6. **linha preta:** representa a variação do tempo obtido para personalizar, sobre diferentes parâmetros de entrada, os documentos denominados **folheto de propaganda 32X**(4382 nodos) e **documento de matrícula 32X** (6429 nodos).

Baseado nas Figuras 55, 56 e 57 e analisando-as em conjunto, pode-se observar que conforme aumenta a quantidade de nodos, maior é o tempo de processamento para se definir as ramificações que devem compor o documento, isso porque o protótipo terá uma maior quantidade de seletores, ações, conteúdos seletivos e seletivos/variáveis para executar seus comportamentos, bem como objetos para serem validados e encapsulados na árvore principal. Com essa análise, consegue-se chegar a uma das finalidades desse experimento, que é verificar se uma maior quantidade de nodos sobre um mesmo documento afeta o desempenho da **Fase de Poda** desse protótipo.

A outra finalidade desse experimento é verificar se um dos formalismos conseguiria se sobressair sobre os demais, afetando o tempo de processamento da aplicação.

Fazendo uma observação comparativa entre esses três gráficos e nas suas quantidades de nodos pode-se se chegar a esse resultado, onde o formalismo que influencia diretamente o custo computacional do protótipo, tornando o tempo de geração da **Fase de Poda** mais lento, são os objetos. Isso porque quanto maior for a quantidade de objetos que um documento tiver (como ocorre no terceiro documento criado que possui inicialmente 18 nodos, enquanto os demais ficam em torno de 2 e 13, onde nunca serão utilizados os 13 juntos em um mesmo documento) maior será o tempo de processamento dessa **Fase de Poda**. O tempo para gerar um folheto de propaganda, inserindo seus dois objetos é de 140 ms, enquanto o tempo para inserir os 18 objetos do exemplo do documento de jornal é 20028 ms.

Isso ocorre porque cada objeto é tratado como uma nova árvore, que deve ser armazenada na memória (através da API DOM) sofrendo uma nova varredura para que se possa avaliar os seus nodos determinando quais seletores devem ser avaliados, recolhendo os nomes e tabelas de onde os dados variáveis, que na próxima fase serão providos, executando ações, entre outras tarefas para após isso serem encapsulados na árvore principal.

8 Conclusão

No mercado globalizado está cada vez mais difícil conquistar novos clientes com tantos concorrentes assediando e fornecendo vantagens. Para conquistar e manter novos consumidores é necessário mostrar que a empresa tem conhecimento sobre os mesmos, através de suas preferências e costumes.

Utilizar essas informações para auxiliar a escolha dos conteúdos variáveis aplicados na geração de documentos personalizados é uma nova maneira que está sendo empregada para que a pessoa que recebe os documentos se sinta prestigiada e continue fiel àquela empresa.

A produção de documentos com conteúdos variáveis é uma forma de geração e comunicação personalizada, que exige um novo modo de perceber os clientes através de individualidades, levando o conceito de segmentação a um nível profundo.

Ao contrário de outras tecnologias que vêm para substituir as anteriores, a geração de documentos personalizados com conteúdo variável continuará convivendo com outros meios de comunicação (como jornais, revistas, páginas web, e-mails) e, se for usada adequadamente, poderá complementá-los, sendo utilizada para a formação de imagem de determinadas marcas com mensagens individualizadas.

Para que um projeto que utilize a geração de documentos personalizados com uma grande quantidade de conteúdos variáveis torne-se bem sucedido é necessário que as tecnologias que permitam essa geração sejam unificadas para que o *workflow* seja rápido, simples e economicamente viável.

Por esses motivos, esse trabalho se propôs a mostrar como diminuir esses problemas através de um modelo que une as vantagens das técnicas de estruturação de documentos e conteúdos variáveis, visando a estruturação física dos documentos através de uma única árvore, em vez de várias para atender diferentes opções de conteúdos que podem compor um documento personalizado.

Além disso, o modelo é composto por um formalismo capaz de fazer o trabalho de busca, por meios de condicionais, para controlar as escolhas desses conteúdos e, por um formalismo que é capaz de reutilizar esses conteúdos (objetos) em vários documentos, faz com que a utilização do modelo na prática torne a geração e a estruturação mais dinâmica e simplificada.

Outro ponto forte que tornou o modelo e seu protótipo mais rápidos, com maior grau de personalização e diminuiu o custo efetivo do seu *workflow*, é o fato deles terem sido idealizados utilizando uma única ferramenta de edição para gerar e transformar as árvores, deixando o processo centrado apenas na ferramenta **DIA** e no protótipo implementado em JAVA, sem que fosse necessário a programação manual dessas árvores diretamente na linguagem XML e até mesmo sem a necessidade da programação de regras para que o protótipo tomasse as decisões de escolha dos conteúdos seletivos e a busca dos conteúdos variáveis nas bases de dados. Isso trará como principal vantagem a possibilidade desse modelo ser utilizado por qualquer desenvolvedor,

designer e até mesmos usuários leigos que nunca tiveram contato com as tecnologias utilizadas para sua formalização.

Além disso, diferente do que se tem atualmente, esse modelo é capaz de intercalar durante a estruturação das árvores tanto conteúdos estáticos quanto conteúdos variáveis, utilizando até mesmo bases de dados sofisticadas. Ele faz com que se consiga gerar documentos com uma ampla liberdade e diversidade nas escolhas de conteúdos para cada cliente, com alto grau de personalização pelo fato de necessitar receber como entrada comandos, no caso parâmetros de entrada, que podem ser considerados um dos aspectos mais fundamentais do modelo, visto que servirão de diretrizes para as decisões a serem tomadas.

A partir dos estudos de caso desenvolvidos e os resultados apresentados neste trabalho, pode-se observar que o modelo e o protótipo implementado são aptos ao que se propõem, sendo capazes de gerar instâncias de um documento personalizada utilizando conteúdos variáveis compatíveis com o perfil do público-alvo.

Pode-se concluir também que apenas trabalhar com esse modelo não é bastante eficiente para transformar as informações individualizadas dos clientes em documentos com conteúdo variável, visto que além da utilização desse protótipo é necessário percorrer um processo multidisciplinar composto por etapas sucessivas, que combinam, além do modelo formalizado, técnicas de mineração da base de dados obtendo um melhor aproveitamento das informações, estratégias de *marketing*, *design* de documento, maneiras de estruturar um documento e logística.

Apesar do bom desempenho obtido ao aplicar esse modelo na prática através do protótipo, os mesmos podem ser aprimorados em trabalhos futuros através da adição e inserção de novos formalismos para que esses sejam capazes de atender uma maior variedade de documentos personalizados com conteúdo variáveis, gerando instâncias cada vez mais complexas e com uma maior autonomia, fazendo com que o modelo seja cada vez menos dependente dos parâmetros de entrada e principalmente da intervenção dos desenvolvedores e projetistas.

Além disso, como principal ponto de aprimoramento dessa solução proposta e para que a mesma possa ser utilizada em gráficas ou empresas, está a necessidade de torná-la apta a combinar a parte lógica dos documentos (conteúdos como textos e imagens), a qual foi abordada nesse trabalho, com a parte que determina sua aparência física, estética e de posicionamento dos dados dentro de um *layout* (apresentação final).

Para que isso ocorra, será preciso adicionar um novo módulo à **Fase de Instanciação** do modelo, para que essa seja capaz de transferir os elementos (conteúdos) contidos no XML, obtido na **Fase de Poda**, para marcadores de posição contidos em *layout* desenvolvido em qualquer ferramenta de editoração ou *software* para diagramação de documentos que sejam compatíveis com a leitura e compreensão de XML.

Neste trabalho, esse aprimoramento não foi estudado, pois trabalhar com o posicionamento de conteúdos de forma que se garanta uma apresentação final aceitável e correta é ainda outro ponto bastante precário e em fase de pesquisas na área de engenharia de documentos, necessitando o desenvolvimento de complexos algoritmos que detectem esses problemas e dinamicamente ajustem os conteúdos ou mudem o *layout* conforme o documento melhor se adapte.

Por fim, para tornar a **Fase de Instanciação** do modelo (principalmente quando o número de documentos a ser gerado através da personalização individualizada ultrapassar 10000 ins-

tâncias) mais ágil e com uma redução no tempo de processamento para a geração de todos as versões pode-se propor que se idealize e trabalhe para que o protótipo implementado seja capaz de processar e gerar paralelamente vários conjuntos de documentos ao mesmo tempo. Assim, essa estratégia deve ser considerada para que resulte em um ganho no tempo de processamento, o qual ainda é a principal limitação desse projeto quando trabalha-se com grandes quantidades documentos.

A Material do Experimento - Objetos do Documento de Matrícula

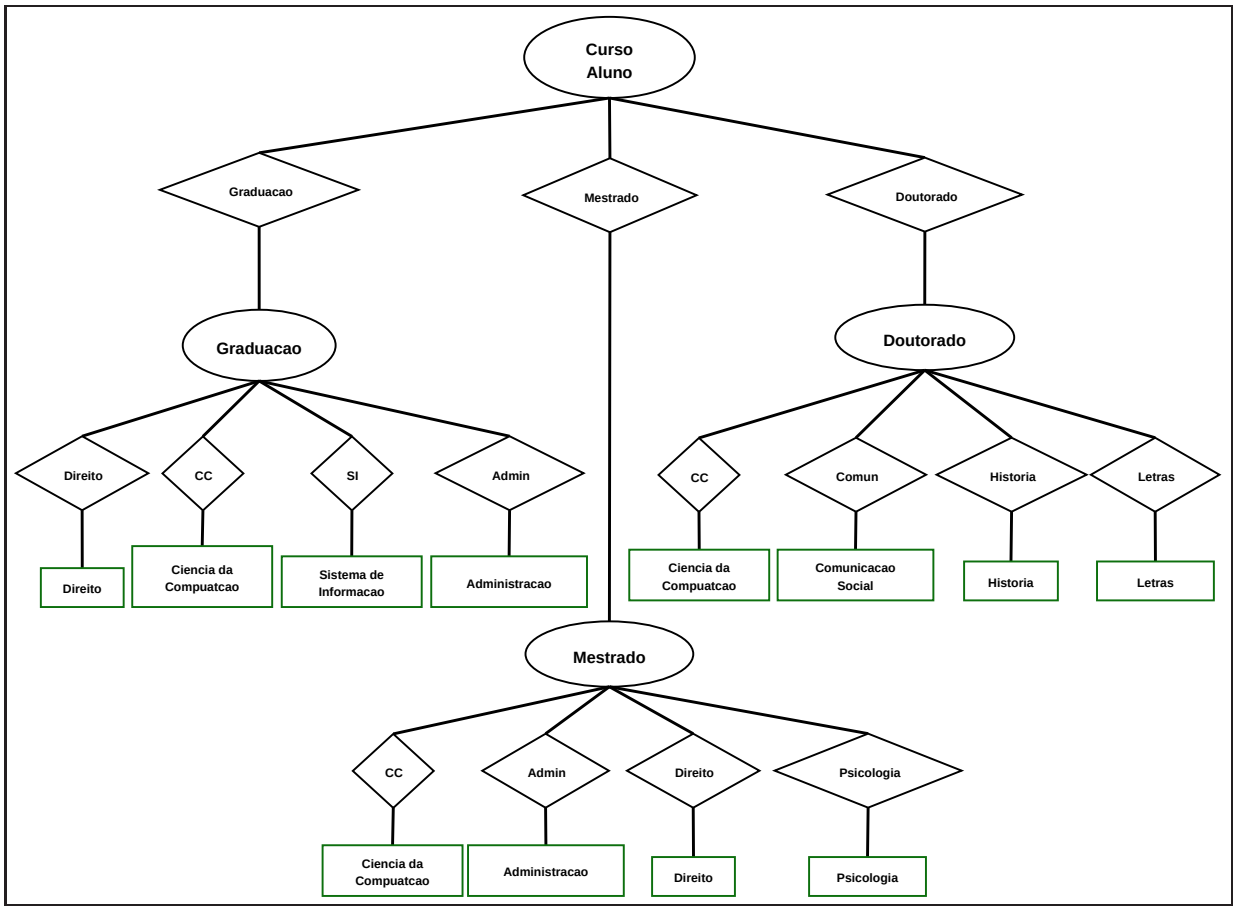


Figura 58 – Objeto com Curso do Aluno.

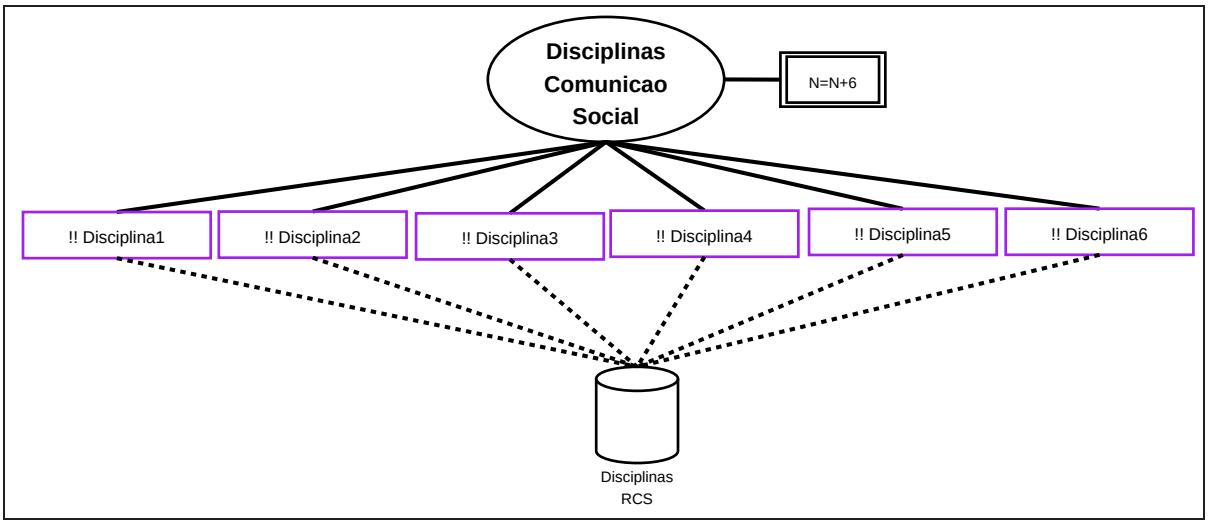


Figura 59 – Objeto com Disciplinas do curso de Comunicação Social.

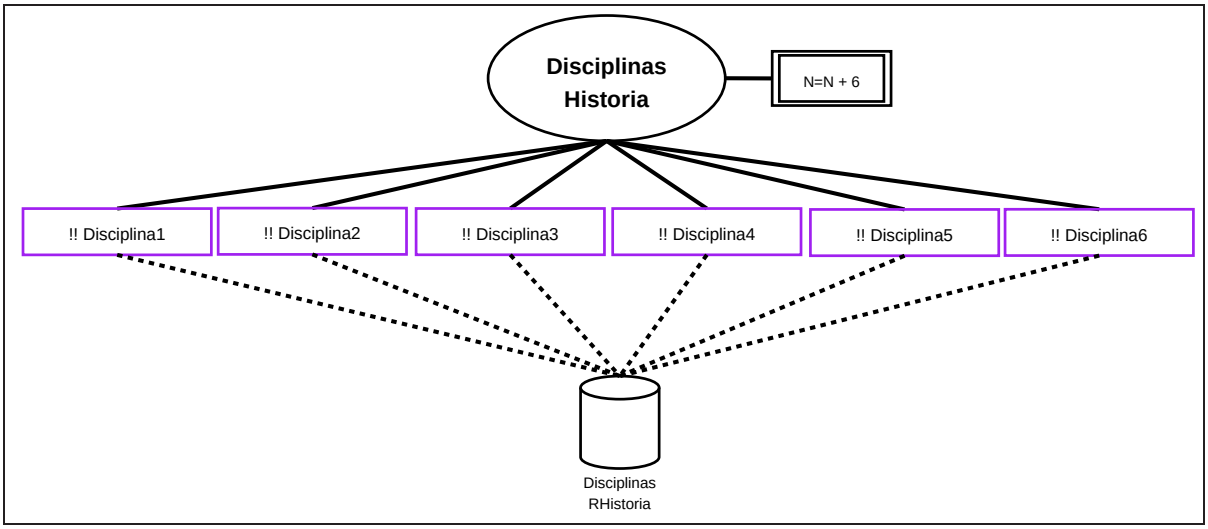


Figura 60 – Objeto com Disciplinas do curso de História.

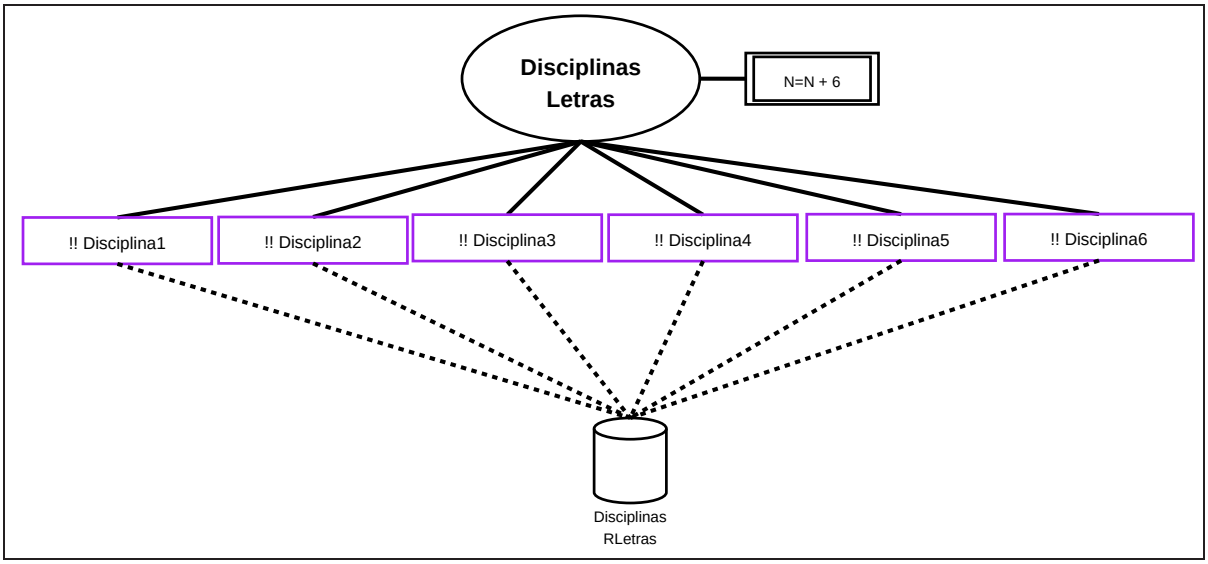


Figura 61 – Objeto com Disciplinas do curso de Letras.

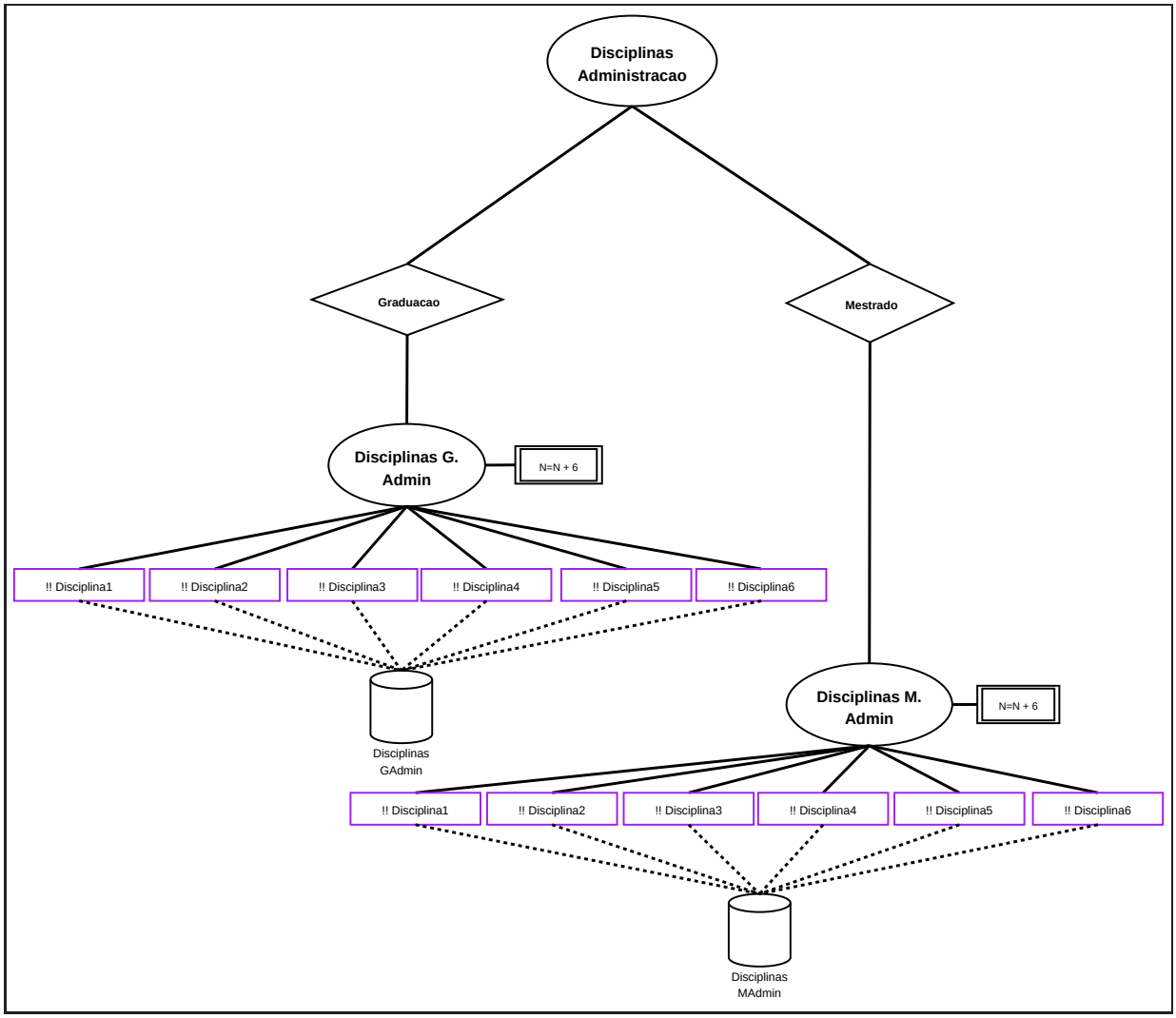


Figura 62 – Objeto com Disciplinas da Administração.

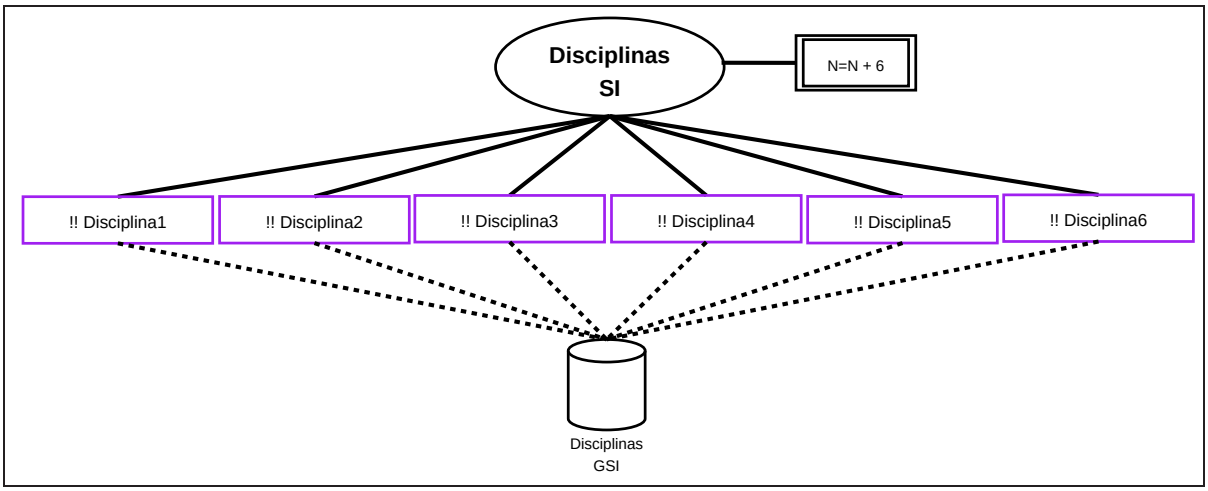


Figura 63 – Objeto com Disciplinas do curso de Sistemas de Informação.

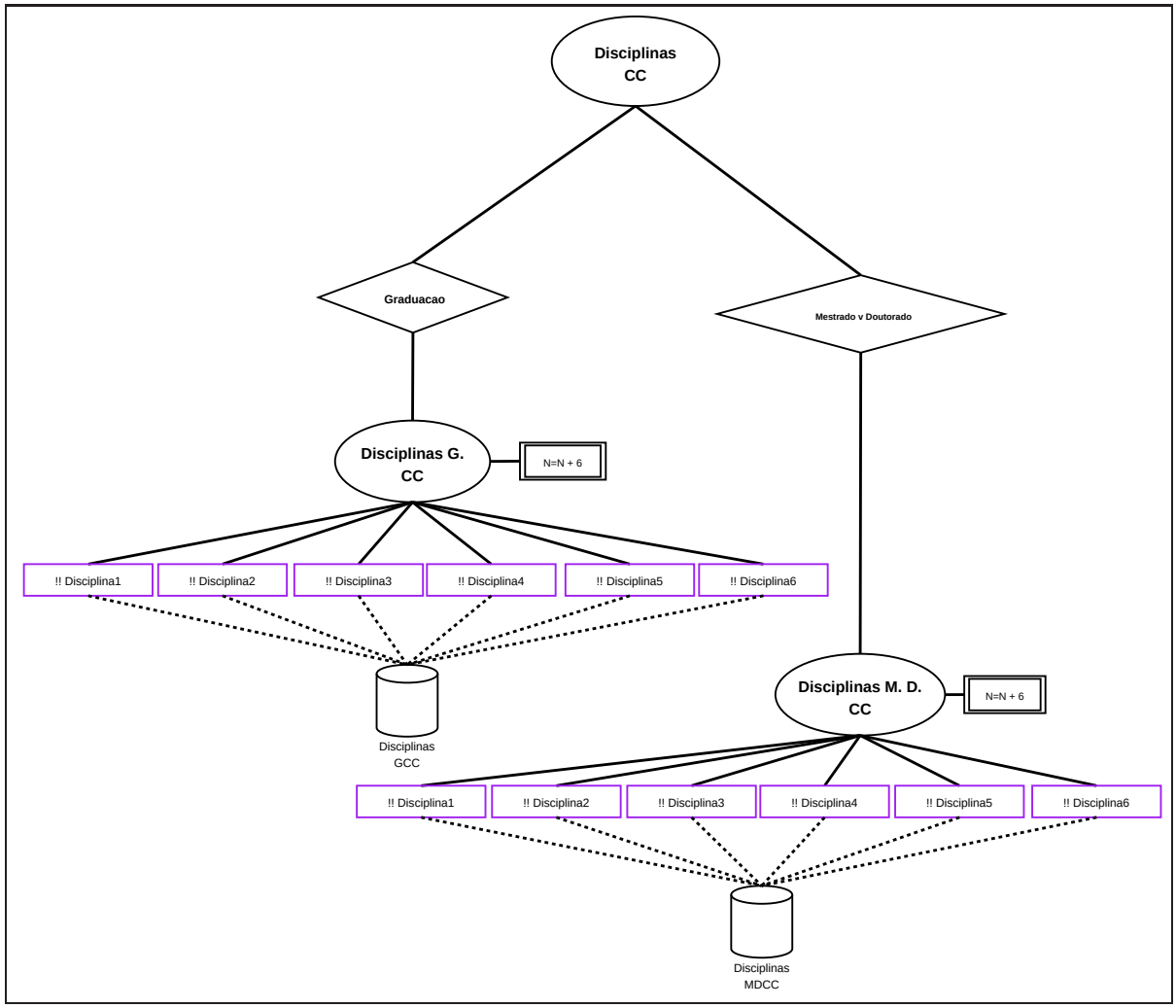


Figura 64 – Objeto com Disciplinas da Ciência da Computação.

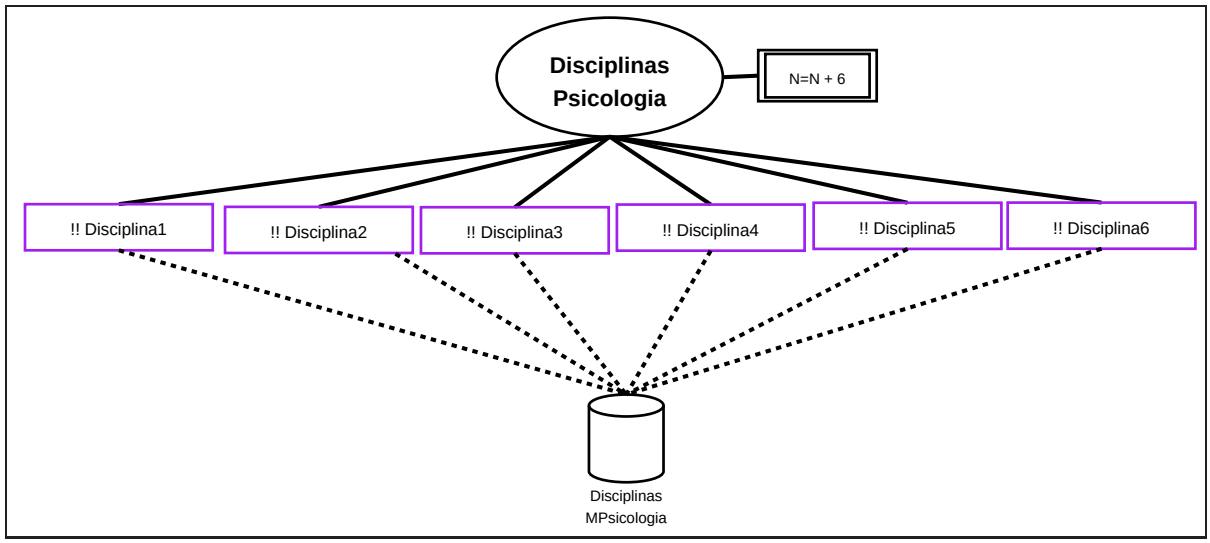


Figura 65 – Objeto com Disciplinas do curso de Psicologia.

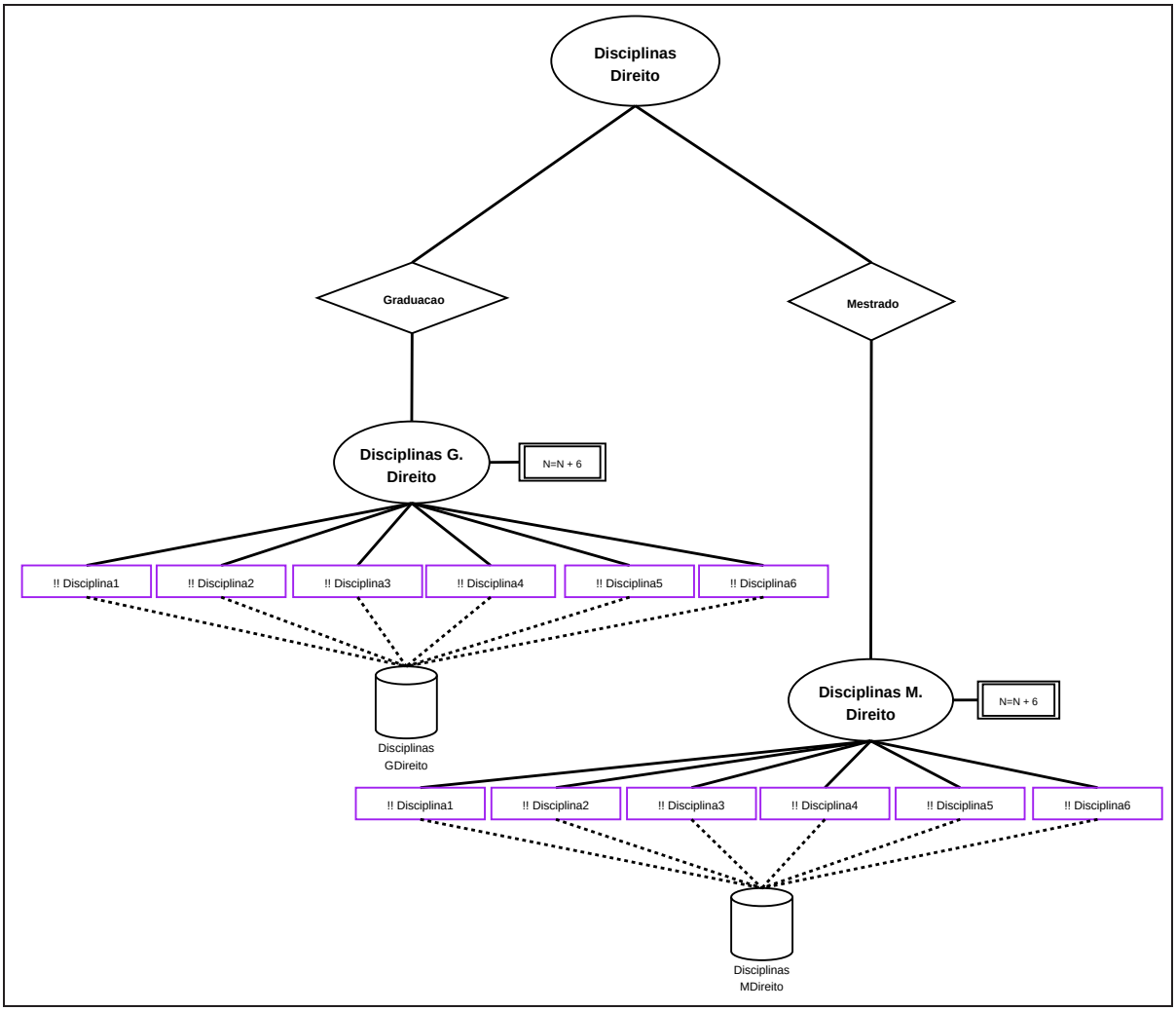


Figura 66 – Objeto com Disciplinas do curso de Direito.

B Material do Experimento - Objetos do Jornal

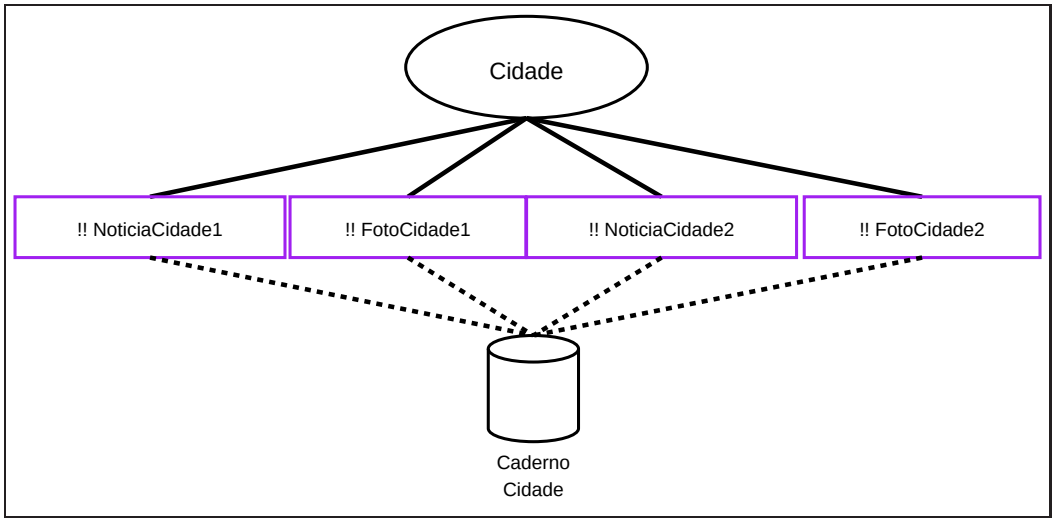


Figura 67 – Objeto caderno Cidade.

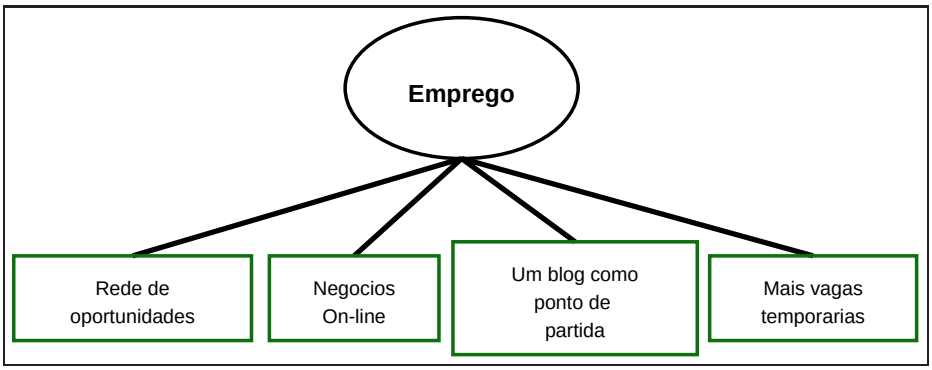


Figura 68 – Objeto caderno Emprego.

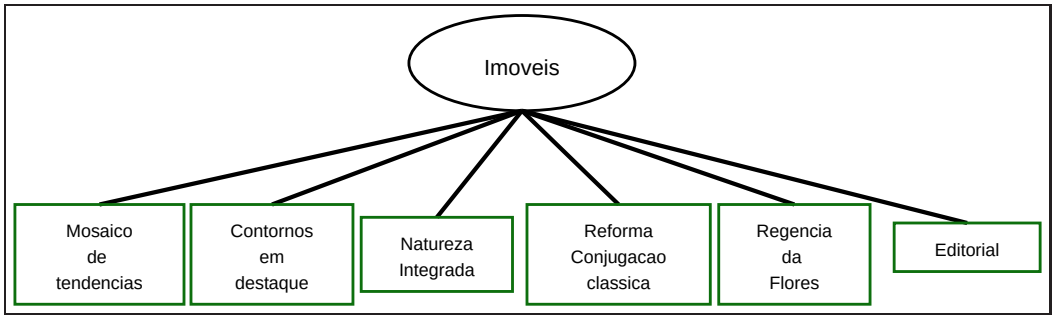


Figura 69 – Objeto caderno Imóveis.

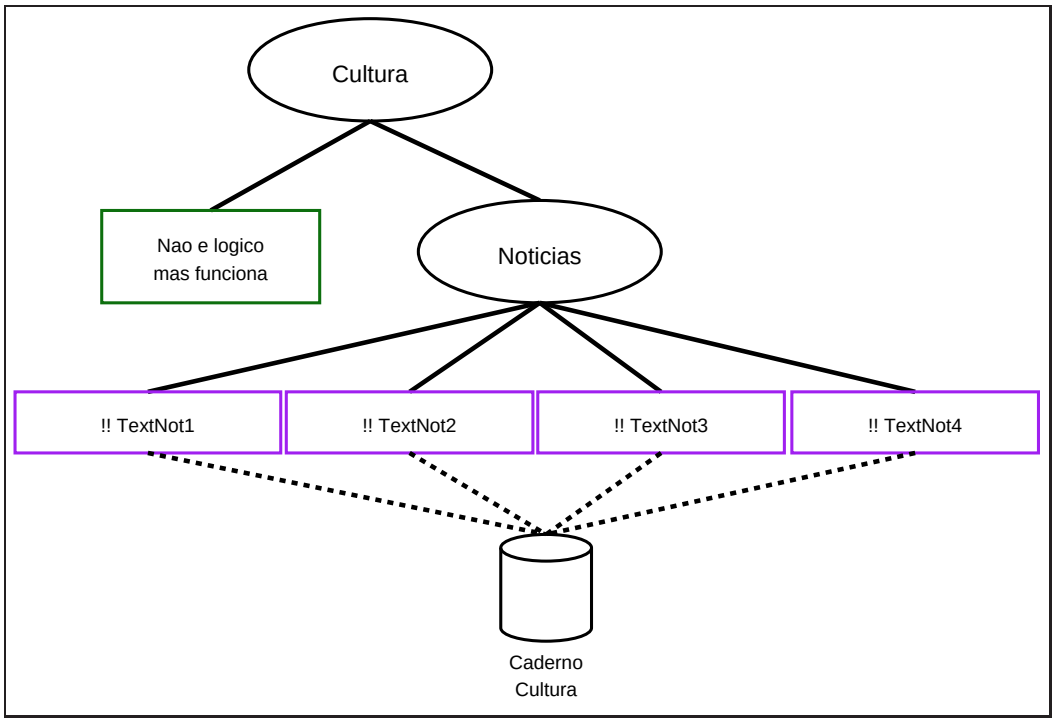


Figura 70 – Objeto caderno Cultura.

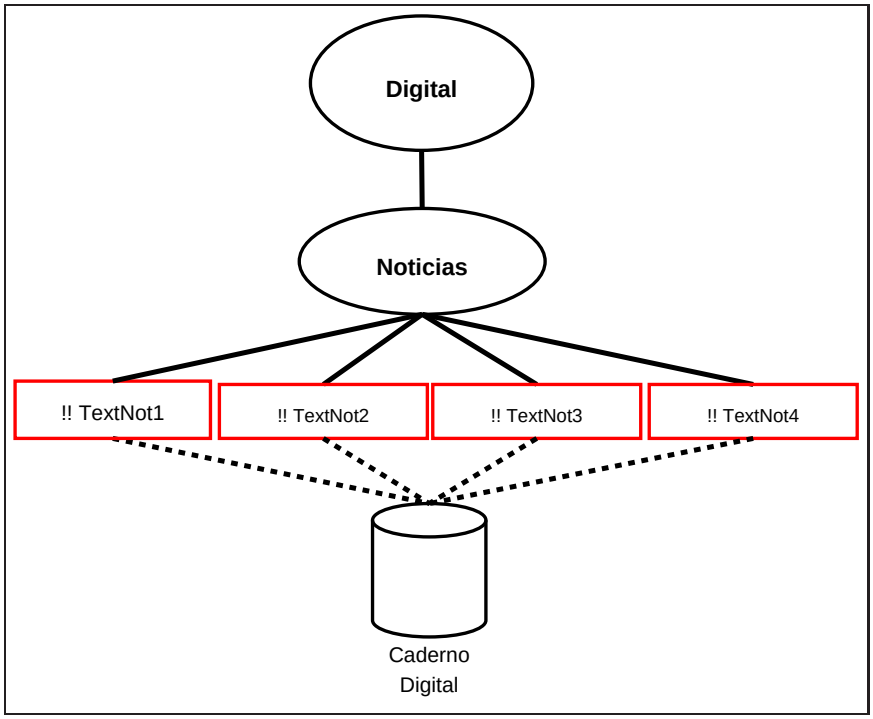


Figura 71 – Objeto caderno Digital.

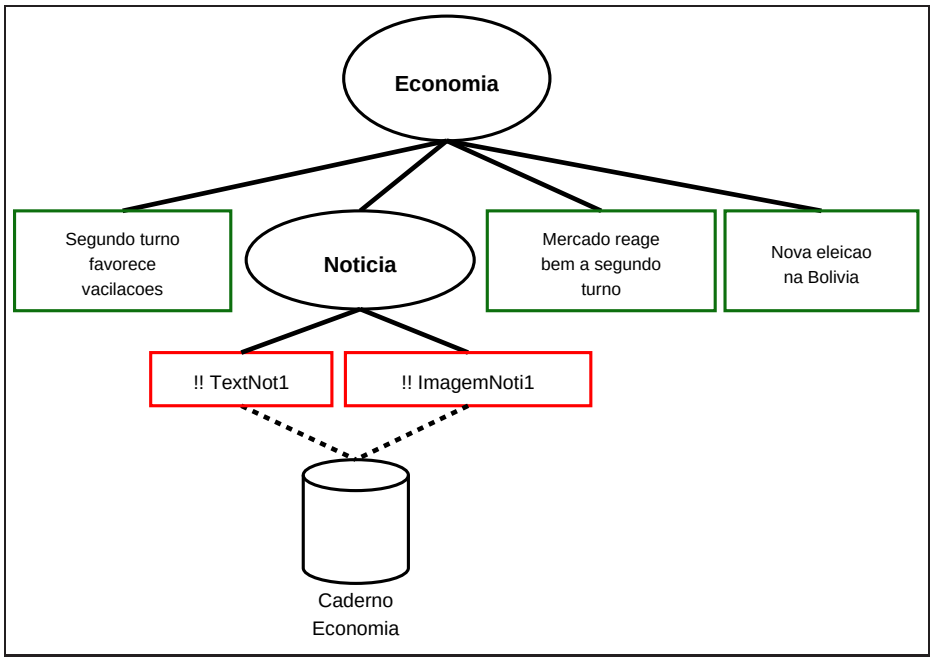


Figura 72 – Objeto caderno Economia.

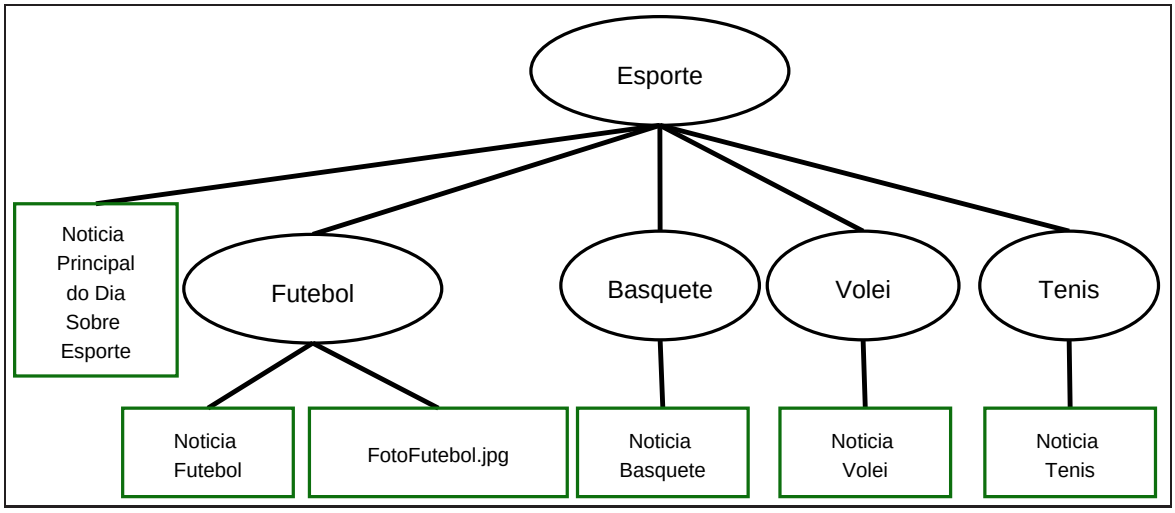


Figura 73 – Objeto caderno Esporte.

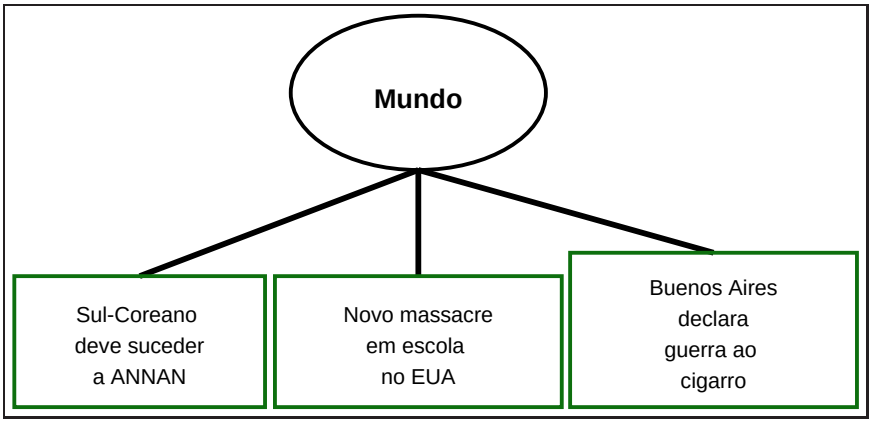


Figura 74 – Objeto caderno Mundo.

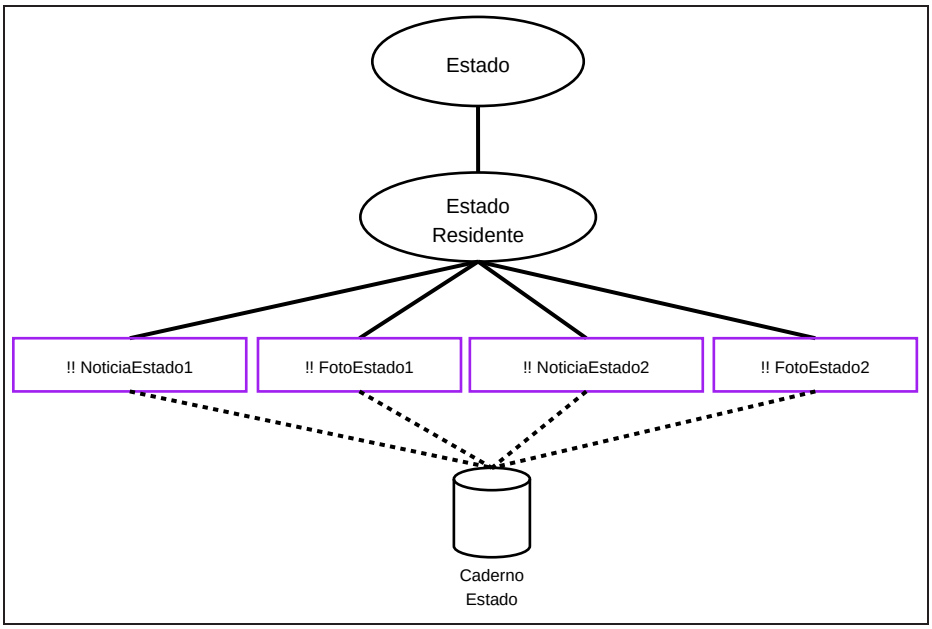


Figura 75 – Objeto caderno Estado.

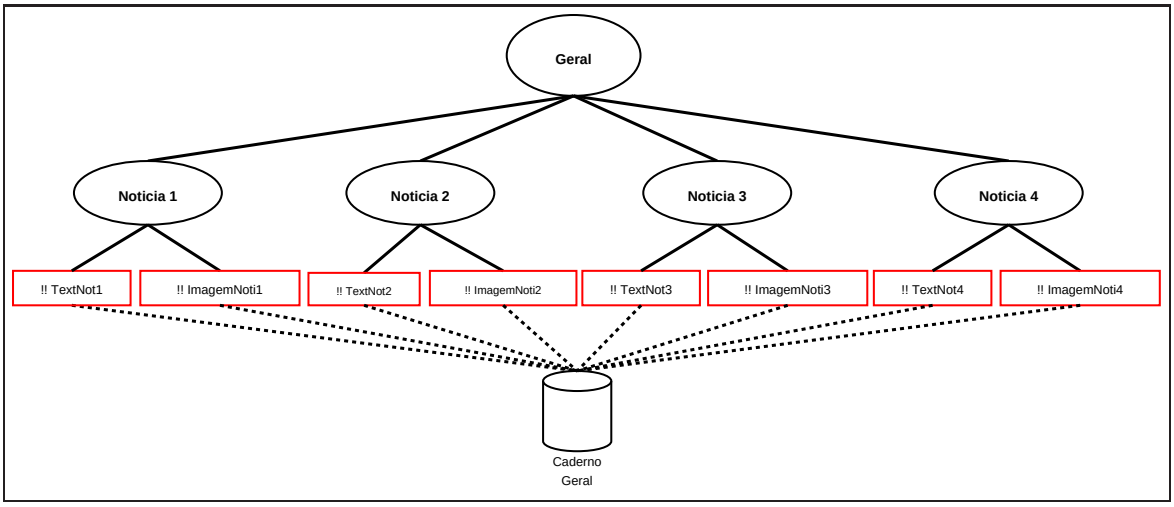


Figura 76 – Objeto caderno Geral.

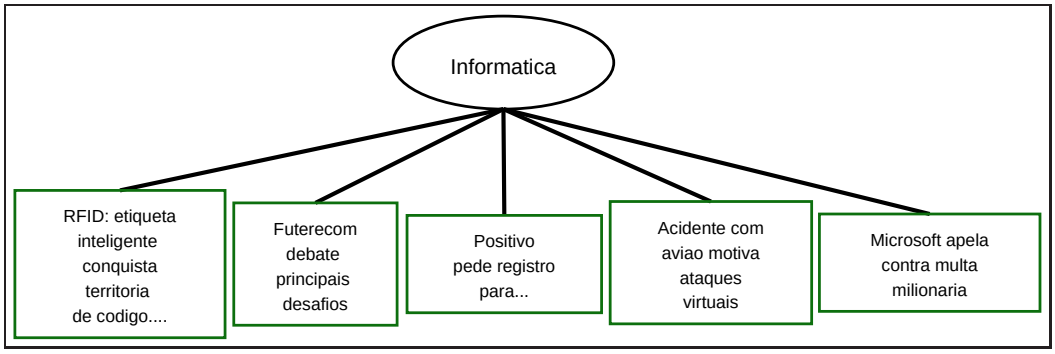


Figura 77 – Objeto caderno Informática.

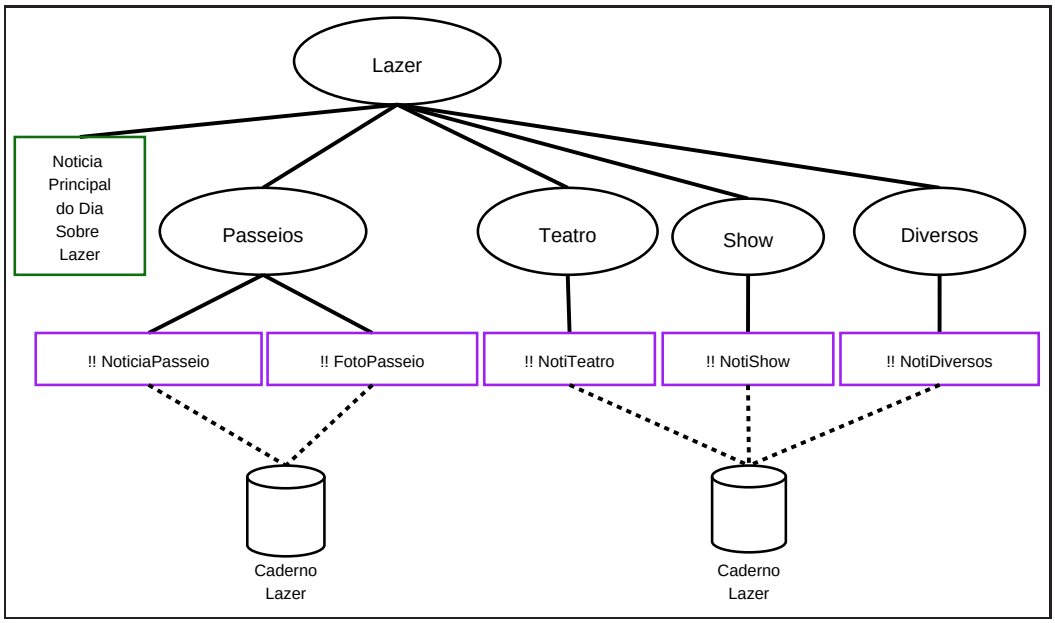


Figura 78 – Objeto caderno Lazer.

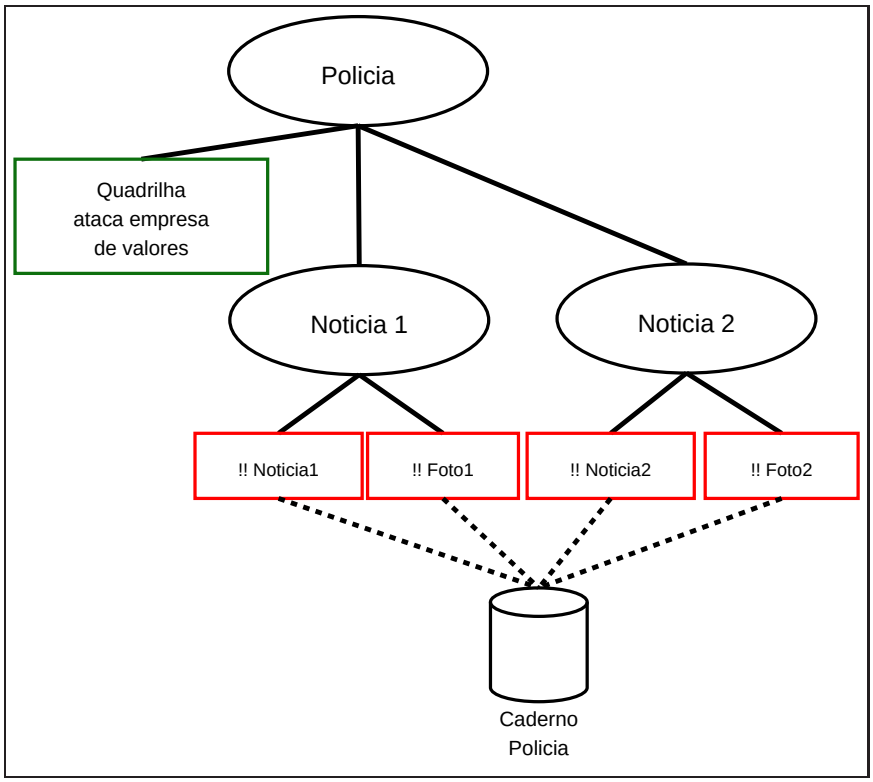


Figura 79 – Objeto caderno Polícia.

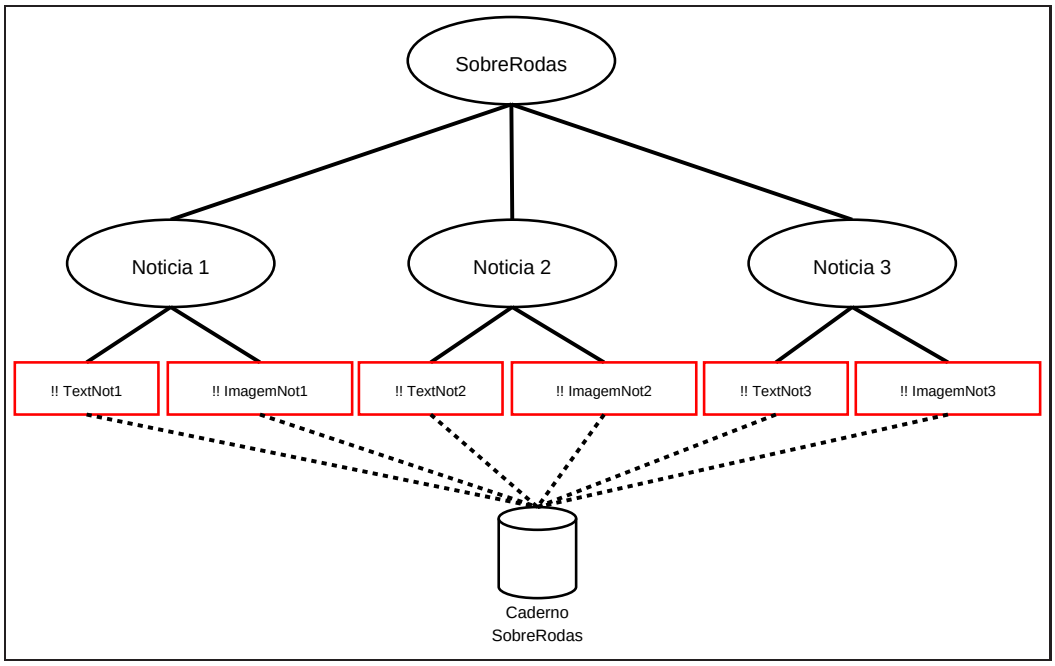


Figura 80 – Objeto caderno Sobre Rodas.

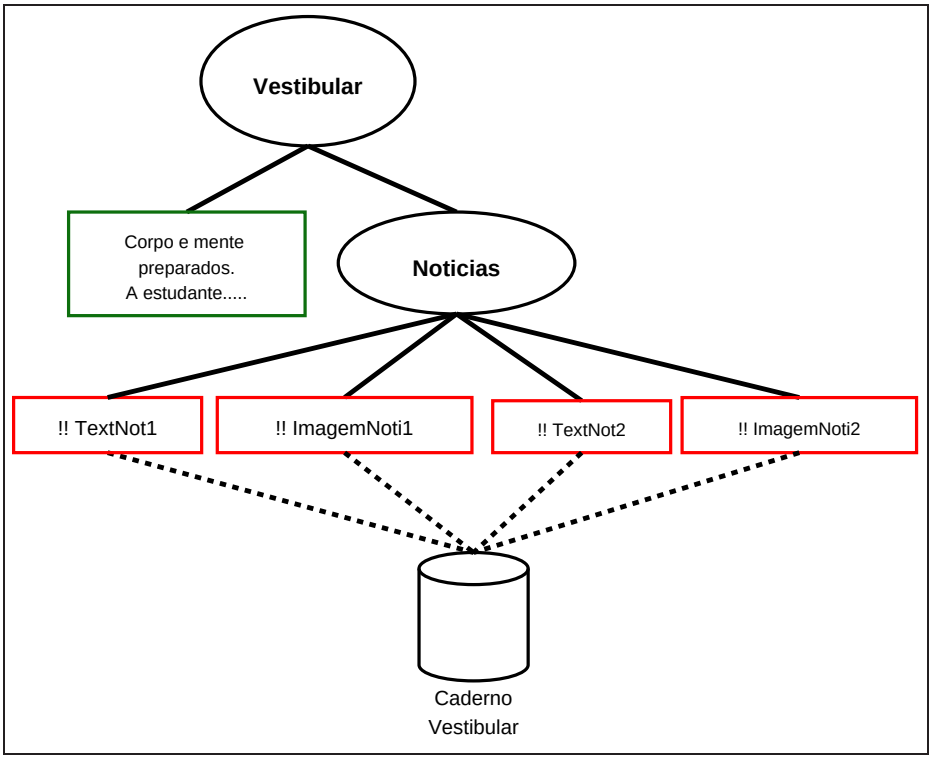


Figura 81 – Objeto caderno Vestibular.

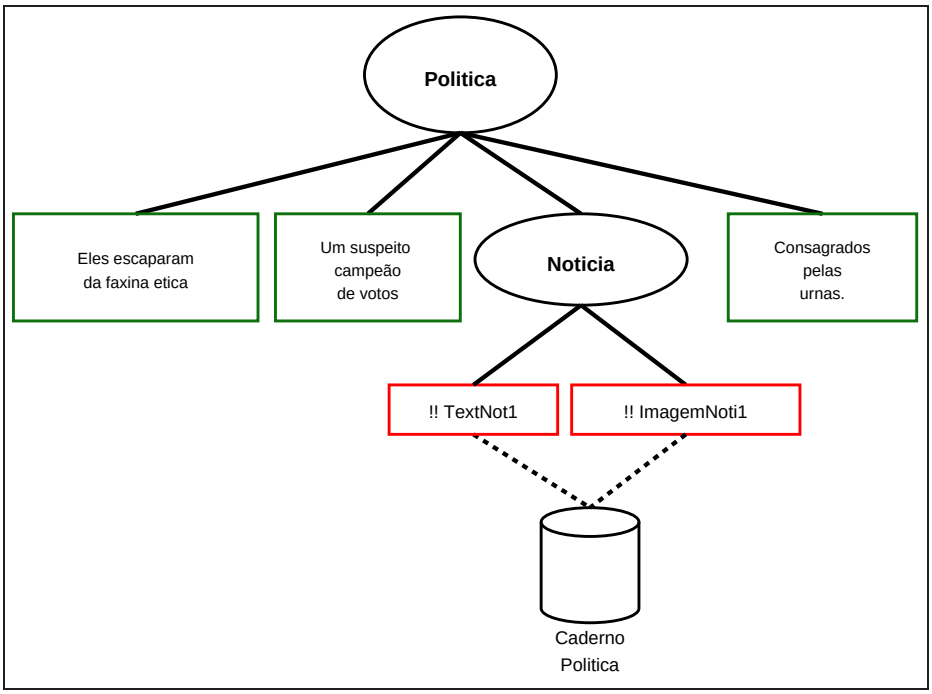


Figura 82 – Objeto caderno Política.

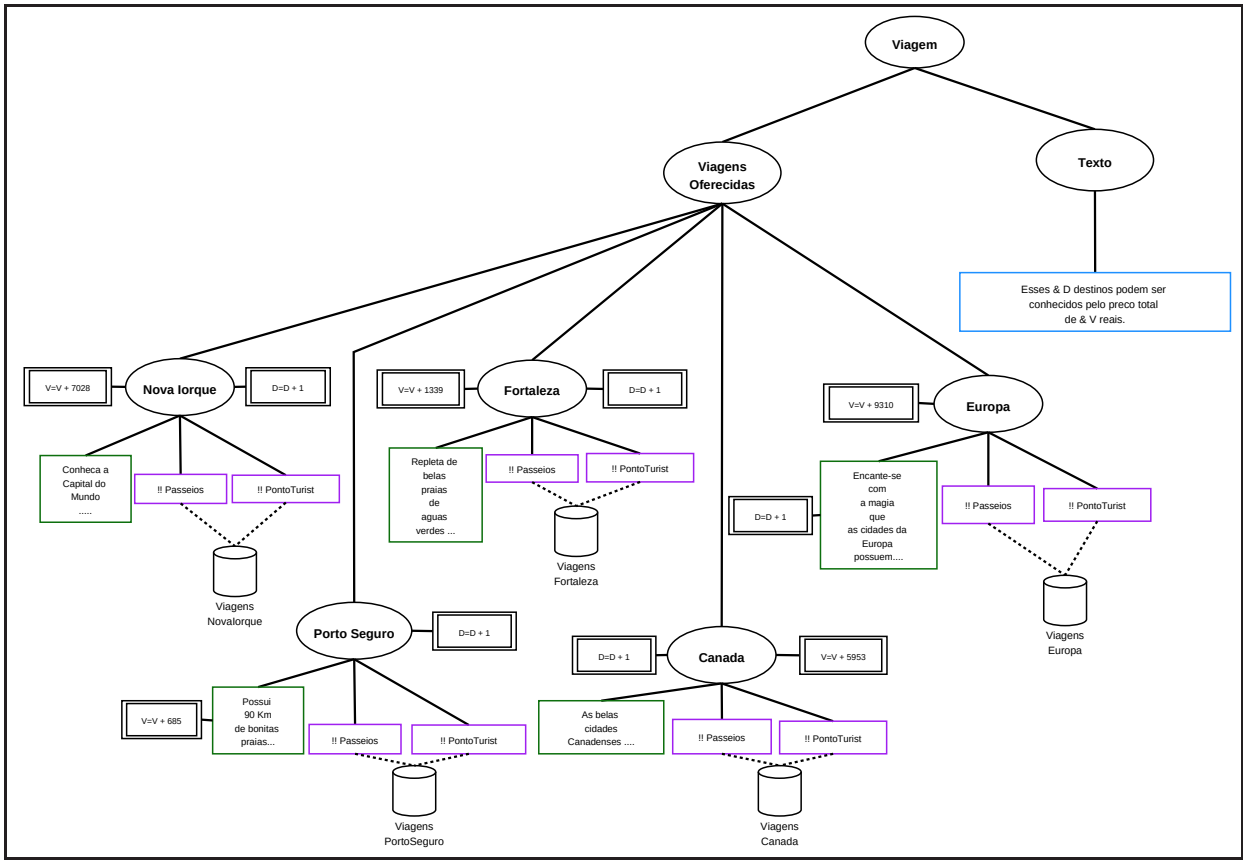


Figura 83 – Objeto caderno viagem.

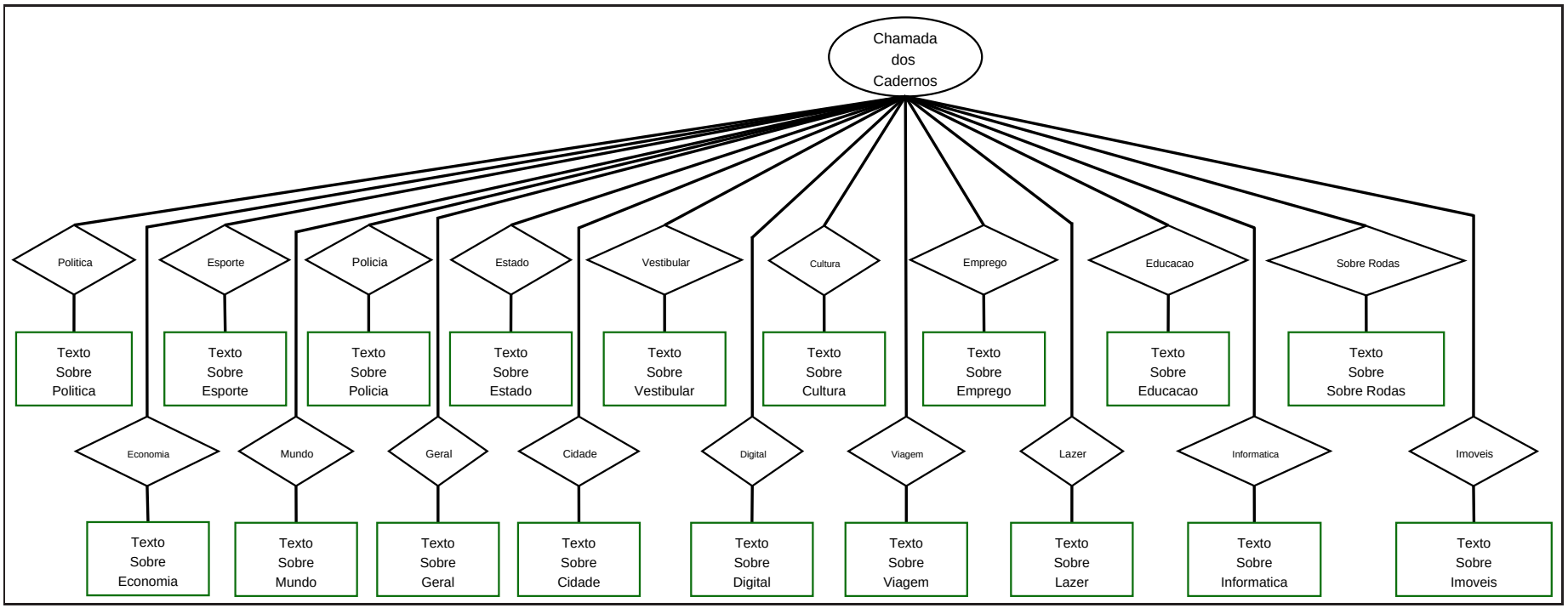


Figura 84 – Objeto com chamada de cadernos.

Referências

[André, Furuta e Quint 1989]ANDRÉ, J.; FURUTA, R.; QUINT, V. *Structured Documents*. New York: INRIA, 1989. (Cambridge series on eletronic publishing). ISBN 0-521-36554-6.

[Clinkunbroomer 2005]CLINKUNBROOMER, J. Vdp: Work you just can't do on offset. *Digital Publishing Solutions*, Maio 2005. Disponível em: <http://www.dpsmagazine.com/Content/ContentCT.asp?P=219>. Acesso em: 05 janeiro 2006.

[Davis 2001]DAVIS, D. One byte at a time: Variable-data printing in perspective. *In-Plant Printer*, p. 13–15 and 54, Julho 2001.

[DIA 2006]DIA. *Software Dia for Linux*. Maio 2006. Disponível em: <http://www.gnome.org/projects/dia/>. Acesso em: 04 maio 2006.

[DIA 2006]DIA. *Software Dia for Windows*. Maio 2006. Disponível em: <http://dia-installer.sourceforge.net/>. Acesso em: 04 maio 2006.

[Duraes 2006]DURAES, R. Dados variáveis como ferramenta de marketing direito. *Professional Publish*, Dabra, n. 83, p. 34, Setembro 2006.

[EFI 2005]EFI. *ABC's of VDP – A Variable Data Printing Basics Guide*. [S.l.], 2005. 1–52 p. Disponível em: <http://www.efi.com/promo/abcseries/>. Acesso em: 09 janeiro 2006.

[EFI do Brasil 2005]EFI DO BRASIL. Abc da impressão digital de dados variáveis – parte 1. *Professional Publish – Tecnologia aplicada as Artes Gráficas, Design e Criação*, Dabra, n. 77, p. 26–35, Julho 2005.

[EFI do Brasil 2005]EFI DO BRASIL. Abc da impressão digital de dados variáveis – parte 2. *Professional Publish – Tecnologia aplicada as Artes Gráficas, Design e Criação*, Dabra, n. 78, p. 24–30, Agosto 2005.

[EFI do Brasil 2005]EFI DO BRASIL. Abc da impressão digital de dados variáveis – parte 3. *Professional Publish – Tecnologia aplicada as Artes Gráficas, Design e Criação*, Dabra, n. 79, p. 25–29, Setembro 2005.

[Foundation 2006]FOUNDATION, G. *Projeto Gnome*. Maio 2006. Disponível em: <http://www.gnome.org/>. Acesso em: 03 maio 2006.

[Furuta 1994]FURUTA, R. Defining and using structured in digital documents. In: *Proceedings of the First Annual Conference on the Theory and Practice of Digital Libraries*. College Station, Texas, USA: [s.n.], 1994. Disponível em: <http://www.csdl.tamu.edu/DL94/paper/furuta.html>.

[Furuta, Quint e André 1988]FURUTA, R.; QUINT, V.; ANDRÉ, J. Interactively editing structured documents. *Electronic Publishing – Origination, Dissemination, and Design*, v. 1, n. 1, p. 19–44, Abril 1988. Disponível em: <citeseer.ist.psu.edu/furuta88interactively.html>.

[Gnuplot Homepage 2006]GNUPLOT Homepage. maio 2006. Disponível em: <www.gnuplot.info/>. Acesso em: 30 novembro 2006.

[Groves e Brailsford 1993]GROVES, M. J.; BRAILSFORD, D. F. Separate compilation of structured documents. *Electronic Publishing – Origination, Dissemination, and Design*, v. 6, n. 4, p. 315–326, Dezembro 1993. Disponível em: <citeseer.ist.psu.edu/groves93separate.html>.

[Heeman 1992]HEEMAN, F. C. Granularity in structured documents. *Electronic Publishing – Origination, Dissemination, and Design*, v. 5, n. 3, p. 143–155, Setembro 1992. Disponível em: <citeseer.ist.psu.edu/heeman92granularity.html>.

[Java Math Expression Evaluator - Versão 1.01 2001]JAVA Math Expression Evaluator - Versão 1.01. abril 2001. Disponível em: <http://lts.online.fr/dev/java/math.evaluator/>. Acesso em: 18 setembro 2006.

[Kita 2003]KITA, D. Variable-data printing: The key to realizing crm benefits. *ECM Insights*, p. 8–10, Março 2003. Disponível em: <http://www.edocmagazine.com/pdf/MarApr03ECMInsights.pdf>. Acesso em: 06 janeiro 2006.

[Komasinski 2005]KOMASINSKI, S. Variable data printing. *Image'n It – A Newsletter from Printing Services*, v. 2, n. 1, p. 3–4, Fevereiro 2005. Disponível em: <http://www.purdue.edu/physicalfacilities/pdf/prntvol2no1.pdf>. Acesso em: 05 janeiro 2006.

[MacDowell 2004]MACDOWELL, D. Variable data printing – what? where? standards! *IPA Bulletin*, p. 22–26, Abril 2004. Disponível em: <http://www.ipa.org/pdf/Stds-VDP.pdf>. Acesso em: 10 janeiro 2006.

[Microsystems 2006]MICROSYSTEMS, S. *Java*. Agosto 2006. Disponível em: <http://java.sun.com>. Acesso em: 20 agosto 2006.

[Microsystems 2006]MICROSYSTEMS, S. *Java SE - Java Database Connectivity (JDBC)*. Setembro 2006. Disponível em: <http://java.sun.com/javase/technologies/database.jsp>. Acesso em: 21 setembro 2006.

[Miley 2003]MILEY, M. Building a variable-data printing solution. *Eletronic Publishing*, Abril 2003.

[Monkerud 2003]MONKERUD, D. Variable-data printing: Personalized delivery of information for the 21st century. *Digital Output*, Setembro 2003. Disponível em: <http://www.digitaloutput.net/content/ContentCT.asp?P=78>. Acesso em: 08 janeiro 2006.

[Moraes 2001]MORAES, C. R. *Estruturas de Dados e Algoritmos – Uma abordagem didática*. São Paulo: Ed. Berkeley, 2001. ISBN 85-7251-585-2.

[MySQL Connector/J 2006]MYSQL Connector/J. Setembro 2006. Disponível em: <http://dev.mysql.com/downloads/connector/j/3.0.html>. Acesso em: 15 setembro 2006.

[MySQL Database 2006]MYSQL Database. Setembro 2006. Disponível em: <http://www.mysql.com/>. Acesso em: 13 setembro 2006.

[Ramon 2000]RAMON, F. *JDBC 2 - Acesso a bancos de dados usando a linguagem JAVA*. São Paulo: Ed. Novatec, 2000. ISBN 85-85184-74-4.

[Seybold Publications 1998]SEYBOLD PUBLICATIONS. Variable-data printing in practice. *The Seybold Report on Publishing Systems*, v. 27, n. 21, Agosto 1998. Disponível em: <http://www.seyboldreports.com/SRPS/subs/2721/PS272101.HTM>. Acesso em: 11 janeiro 2006.

[Smith 2005]SMITH, M. Variable data printing: Building the variable data base. *Printing Impressions*, Janeiro 2005. Disponível em: <http://www.piworld.com/doc/283268101149484.bsp>. Acesso em: 06 janeiro 2006.

[Szwarcfiter e Markenzon 2000]SZWARCFITER, J. L.; MARKENZON, L. *Estruturas de Dados e seus Algoritmos*. Rio de Janeiro: Ed. LTC - Livros Técnicos e Científicos, 2000.

[Veloso 2003]VELOSO, R. R. *Java e Xml - Processamento de documentos XML com Java*. São Paulo: Ed. Novatec, 2003. ISBN 85-7522-036-5.

[Wirth 1999]WIRTH, N. *Algoritmos e Estruturas de Dados*. Rio de Janeiro: Ed. LTC - Livros Técnicos e Científicos, 1999.