UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE UM SISTEMA DE GERENCIAMENTO DE INFORMAÇÕES DE LIVROS UTILIZANDO OBJETOS DISTRIBUÍDOS

ISRAEL RODNEI DE FAUSTINO

BLUMENAU
2006

2006/1-19

ISRAEL RODNEI DE FAUSTINO

PROTÓTIPO DE UM SISTEMA DE GERENCIAMENTO DE INFORMAÇÕES DE LIVROS UTILIZANDO OBJETOS DISTRIBUÍDOS

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Paulo Fernando da Silva - Orientador

BLUMENAU
2006

2006/1-19

PROTÓTIPO DE UM SISTEMA DE GERENCIAMENTO DE INFORMAÇÕES DE LIVROS UTILIZANDO OBJETOS DISTRIBUÍDOS

Por

ISRAEL RODNEI DE FAUSTINO

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Paulo Fernando da Silva – Orientador, FURB
Membro:	Prof. Paulo César Rodacki Gomes – FURB
Membro:	Prof. Alexander Roberto Valdameri – FURB

Blumenau, 20 de março de 2006

A minha esposa Cristiana e a minha linda e querida filha Ana Caroline pelo amor e compreensão, aos meus familiares, principalmente minha mãe Carmelina e meu padrasto Lauro, pela força e dedicação prestadas.

AGRADECIMENTOS

Agradeço a Deus, que fez com que trilhássemos esse caminho para que pudéssemos descobrir que quando queremos, somos capazes de tudo. Também nos encheu de força e fez com que os amigos e familiares dissessem as palavras certas, nas horas exatas.

Agradeço ao professor Paulo Fernando da Silva, que proporcionou a oportunidade de elaborar um trabalho de enriquecimento individual e profissional, nesta primeira etapa de minha vida acadêmica. Sem sua orientação, este trabalho e esta monografia não seriam possíveis.

Agradeço ao professor Alexander Roberto Valdameri, que me auxiliou nas dúvidas que insistiam em aparecer durante o desenvolvimento deste trabalho.

Para finalizar agradeço ao meu grande amigo Fábio Augusto Bruns, que me auxiliou nas horas difíceis, proporcionando orientação nos problemas que ocorreram no decorrer do desenvolvimento deste protótipo.

“O que é feito por amor está sempre além do bem e do mal”

Nietzsche

RESUMO

Com o crescimento de editoras e livrarias e, conseqüentemente, o lançamento de novos livros e atualização constante de edições, torna-se cada vez mais necessária a implantação de um sistema que reúna em um só local todas as informações possíveis e indispensáveis para que as livrarias possam acessar os dados referente a livros mais rapidamente, com isso repassar estas informações com mais agilidade e confiabilidade aos seus clientes. A ferramenta proposta tem por finalidade, através de um ambiente distribuído e utilizando-se de um padrão de formatação de dados de livros, colocar os dados atualizados pelas editoras disponíveis para acesso pelas livrarias de uma forma rápida e completa. Este protótipo tem por finalidade o cadastramento de dados pelas editoras em um sistema distribuído utilizando CORBA, estes dados de cadastramento de livros seguem o padrão de catalogação internacional MARC 21. As editoras terão um módulo onde poderão incluir novos livros ou alterar os mesmos. O banco de dados das editoras é disponibilizado para as livrarias utilizando também de um ambiente distribuído, através de uma tela de consultas que será utilizada pelos funcionários ou clientes das livrarias, depois de efetuada a consulta o sistema gravará um arquivo em XML que servirá como um *cache* para novas consultas, ou seja, buscando informações deste arquivo que já foram consultadas.

Palavras-chave: Objetos distribuídos. Multicamadas. Livrarias. Editoras. Livros.

ABSTRACT

The publishing houses and bookstores development and, consequently, the publishing of new books and editions updating have increased the necessity of a new system able to concentrated every information, such as possible or useful, in a same place, with the main objective of supporting bookstores with data books in real time, quickly and reliable, making in mind to get a best link into the relation store assistant/customer. This tool, as a proposal, means to have updating data from publishing houses to free access by bookstores, in a fast way, using a distributed environment in a data format standard. This model is thought for a better way of organizing data list into the system by the publishing houses, in distributed system, using CORBA. Such data follow a kind of catalog international standard, called MARC 21. The publishing houses will be able to include new books using a simple module, or even modify them, like categories, skills and so on. The data base of these publishing houses will be available to bookstores using a distributed environment too. For this, a consultation screen can be used by customers or bookstore assistants. After the consultation be done, the system will record, in XML, all changes. All then will serve with cache for new consultation that is, researching information from that file which had already been consulted.

Key-words: Distributed objects. Multicamadas. Bookstores. Publishing companies. Books.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

CORBA – *Common Object Request Broker*

MARC – *Machine Readable Cataloging*

XML – *Extensible Markup Language*

UML – *Unified Modeling Language*

OD – Objetos Distribuídos

ORB – *Object Request Broker*

IDL – *Interface Definition Language*

OMG – *Object Management Group*

SQL – *Structured Query Language*

FURB – Fundação Universidade Regional de Blumenau

HTML – *Hypertext Markup Language*

SGML – *Standard Generalized Markup Language*

IP – *Internet Protocol*

PEP – Prontuário Eletrônico de Pacientes

SUMÁRIO

1 INTRODUÇÃO

Segundo Mindlin (2004), livros transmitem pensamentos, traduzem emoções, estimulam a imaginação e o sonho, permitindo que a vida cotidiana tenha um sentido intelectual e espiritual de inestimável valor. Atualmente, o mercado editorial brasileiro está conseguindo satisfazer esses interesses. As tiragens ainda são restritas e a distribuição precária, mas o número não é pequeno, alcançando cerca de 3.000 títulos mensais, e a publicação de autores novos e consagrados cresceu consideravelmente.

Na medida que a informação referente a livros e inclusão de novos autores se expande, as editoras enfrentam o desafio de prover fácil acesso dessas informações às livrarias. Além dos lançamentos, é necessário disponibilizar informações sobre a edição do livro, ou seja, se o mesmo está sendo reeditado, esgotado, temporariamente esgotado ou fora de catálogo, além das alterações de preços que são constantes.

Muitas editoras disponibilizam as informações sobre seus livros em suas páginas na Internet, e por sua vez as livrarias tendem a acessar diariamente estas páginas para atualizar seus bancos de dados. Com o acesso a diferentes editoras, surgem vários tipos de informações, muita delas incompletas, por exemplo: arquivos com listas de preços em diferentes formatos (txt, xls, pdf), livros sem resenha, autor incompleto, assunto não informado e outros. Esse processo realizado pelas livrarias é árduo, pois além da demora na obtenção das informações, estas nem sempre estão disponíveis de imediato. Muitas vezes há a necessidade de contatar as editoras para obter mais informações por outros meios (telefone, fax, *e-mail*), gerando mais custos para as livrarias.

Tendo em vista o problema apresentado, este trabalho propõe o desenvolvimento de um protótipo de sistema de gerenciamento de informação de livros utilizando Objetos Distribuídos (OD).

O sistema é distribuído, oferecendo grande faixa de recursos computacionais para a aplicação, independente da localização do usuário ou dos equipamentos computacionais que sejam requeridos. Esses recursos permitem que a aplicação ofereça ao usuário, melhor desempenho e uso mais eficiente dos serviços computacionais do sistema.

Em um sistema distribuído, a arquitetura do sistema está organizada em camadas de software, onde o processamento e os dados manipulados pelo mesmo ficam distribuídos e acessíveis aos usuários a partir de qualquer localidade. Segundo Poloniato (1999, p. 16), a arquitetura cliente/servidor multicamadas *tree-tier*, oferece uma interface formal entre a camada do aplicativo (interface com o usuário) e o banco de dados, através das regras de negócios, garantindo assim, estruturas de dados melhores e mais confiáveis.

Para comunicação entre processos utilizou-se o padrão *Common Object Request Broker Architecture* (CORBA), principalmente porque, é uma arquitetura conhecida e muito utilizada para gerenciar ambientes distribuídos.

Com o intuito de manter um padrão nos dados armazenados pelas editoras, foi utilizado um formato de intercâmbio de informação para livros, denominado *Machine Readable Cataloging* (MARC 21), em sua versão em *Extensible Markup Language* (XML) (MARC STANDARDS, 2004).

Toda a tecnologia utilizada para o desenvolvimento desta aplicação visa formar uma rede de distribuição de informações para livrarias, onde as mesmas poderão observar o comportamento do setor editorial com uma agilidade que hoje é cada vez mais necessária.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um protótipo que possibilite livrarias de todo o Brasil acessarem informações referente a livros de diversas editoras em um só local.

Os objetivos específicos do trabalho são:

a) disponibilizar um módulo para editoras gerenciarem os dados cadastrados referente a livros e um módulo para que as livrarias possam obter esses dados;
b) controlar o acesso das livrarias através de usuários e senhas, para obter maior segurança em relação aos dados disponibilizados pelas editoras;
c) utilizar o MARC 21, modelo para metadados de livros amplamente conhecido e utilizado no mundo, padronizando assim o cadastramento dos mesmos pelas editoras.

1.2 ESTRUTURA DO TRABALHO

No primeiro capítulo é feita a introdução ao trabalho, demonstrando brevemente as deficiências atuais bem como as necessidades às quais a aplicação se destina a resolver.

No segundo capítulo serão abordados os tópicos relacionados à parte técnica da aplicação, demonstrando as tecnologias e produtos utilizados na confecção do protótipo.

No terceiro capítulo será apresentado o protótipo em si, suas especificações, seu desenvolvimento, seu funcionamento e sua distribuição.

No quarto capítulo serão apresentadas as conclusões finais sobre o trabalho, alguns problemas enfrentados durante o desenvolvimento e algumas sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO

Neste capítulo são apresentados alguns aspectos teóricos relacionados ao trabalho, tais como: objetos distribuídos, CORBA, ORB, IDL, XML, MARC e ao final, são apresentados alguns trabalhos correlatos.

2.1 OBJETOS DISTRIBUÍDOS

Capeletto (1999, p. 12) menciona que objetos distribuídos é um paradigma da computação que possibilita que objetos sejam distribuídos através de uma rede heterogênea, e permita a cada componente a total interoperabilidade e acesso a outros objetos. Para uma aplicação escrita em um ambiente de objetos distribuídos, objetos remotos são acessados como se estivessem na máquina que efetua as requisições.

Segundo Coulouris, Dollimore e Kindberg (2001, p. 171), o estado de um objeto consiste nos valores de suas variáveis instanciadas. No paradigma baseado em objetos o estado de um programa é particionado, cada um associado com um objeto. Em um sistema distribuído, desde que os programas baseados em objetos sejam divididos logicamente, a distribuição física dos objetos em processos ou em computadores é uma extensão natural.

A diferença entre objetos clássicos e objetos distribuídos é que, um objeto clássico possui propriedades e métodos e é gerado através de uma linguagem de programação como Delphi, C++, Visual Basic, entre outras. Esses objetos só existem dentro de um programa, apenas o compilador que os criou conhece a sua existência.

Segundo Riccioni (2000, p. 12), unidas ao paradigma de orientação a objetos, as aplicações distribuídas podem referenciar-se a objetos do sistema em qualquer lugar da rede, proporcionando maior flexibilidade e reutilização de código, tornando mais eficiente o

desenvolvimento de aplicações.

Um objeto distribuído também pode ser uma classe, que torna disponíveis tanto suas propriedades quanto seus métodos, mas a linguagem e o compilador usados para criar objetos distribuídos são totalmente transparentes para a implementação, ou seja, são armazenadores de códigos que executam uma determinada tarefa. Sua principal característica refere-se a sua localização, eles podem ser abrigados todos em uma única máquina, ou distribuídos em máquinas distintas de uma rede de computadores (*Local Area Network* (LAN), *Wide Area Network* (WAN) e Internet). Em conjunto, esses objetos são capazes de executar funções para um sistema distribuído.

A tecnologia de objetos distribuídos permite a comunicação e a cooperação entre objetos (aplicações), independentemente da arquitetura de hardware, sistema operacional, banco de dados, linguagem de programação e rede de computadores. Pode-se dizer que os objetos distribuídos são uma evolução do objeto convencional.

Objetos distribuídos possuem uma interface específica onde os computadores geram um código a mais para serem acessados por outros objetos, de maneira que o programa que o solicite desconheça o local onde o objeto chamado está abrigado, o sistema operacional que está sendo utilizado e a linguagem na qual foi criado.

“Atualmente, existem na indústria dois grandes padrões para desenvolvimento de objetos distribuídos que separam a interface de um objeto de sua implementação: *Distributed Component Object Model* (DCOM) e CORBA” (RODRIGUES, 2003).

As tecnologias de objetos distribuídos são utilizadas nesta aplicação para o desenvolvimento de um sistema distribuído. Segundo Tanenbaum e Steen (2001, p. 2), um sistema distribuído, é uma coleção de computadores independentes que parecem como um único sistema para o usuário.

2.2 CORBA

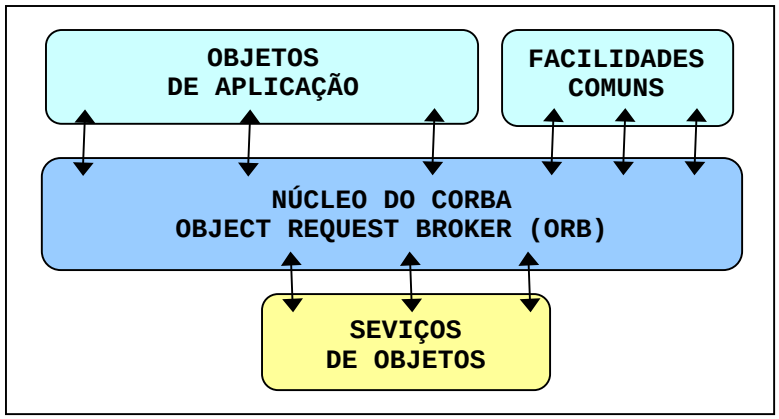
Segundo Graham (1994), em 1989 foi fundado o *Object Management Group* (OMG), uma organização que se preocupa de forma primordial com o estabelecimento de padrões para a industria de software. O OMG é uma associação comercial internacional, com fins não lucrativos, mantida por cerca de 800 companhias de hardware e software.

A missão do OMG é, segundo Martin (1994):

- dedicar-se a maximizar a portabilidade, a capacidade de reaproveitamento e a interoperabilidade de software. O OMG é a organização que lidera mundialmente a iniciativa de proporcionar um quadro contextual e as especificações para os ambientes baseados em objetos disponíveis no mercado;
- prover uma arquitetura referencial com termos e definições sobre os quais se baseiam todas as especificações. Implementação dessas especificações tornar-se-ão disponíveis sob condições e termos justos e com eqüidade. O OMG cria padrões industriais para os sistemas baseados em objetos, focalizando-se no acesso remoto a objetos, no encapsulamento dos aplicativos e nas interfaces dos bancos de dados baseados em objetos;
- prover um fórum aberto para discussões no que diz respeito a indústria em questão, à educação e à promoção da tecnologia baseada em objetos. O OMG coordena suas atividades com organizações correlatas e atua como um centro de tecnologia e marketing para os softwares baseados em objetos.

Ainda, segundo Martin (1994), CORBA permite que objetos invoquem métodos em objetos distribuídos em redes, como se estes fossem locais. Em uma implementação CORBA é necessário que haja, no mínimo, dois *Object Request Brokers* (ORB), localizados em ambos os lados do sistema cliente/servidor, para criar e gerenciar a comunicação entre os objetos.

Segundo a OBJECT MANAGEMENT GROUP (1998, p. IV), a chave para entender a estrutura da arquitetura CORBA é o *Reference Model*, que consiste nos componentes apresentados no quadro 1.



Quadro 1 – Arquitetura de modelo de referência da OMG.

O quadro 1 apresenta os componentes que formam o *Reference Model* e são descritos abaixo.

a) ORB: permite aos objetos a transparência de fazer e receber solicitações e respostas em um sistema distribuído. É a fundação para construir aplicações para objetos distribuídos e possibilitar a interoperabilidade entre aplicações em ambientes heterogêneos;

b) Serviços de objetos: coleção de serviços (interfaces e objetos) que suportam funções básicas para usar e implementar objetos. Serviços são necessários para construir qualquer aplicação distribuída e são sempre independentes da aplicação;

c) Facilidades comuns: é uma coleção de serviços que muitas aplicações podem compartilhar, mas que não são tão fundamentais como o *Object Services*;

d) Objetos de aplicação: são os produtos de um único fornecedor que controla suas interfaces. São produtos não padronizados pela OMG e constituem a camada mais alta do *Reference Model*;

2.2.1 OBJECT REQUEST BROKER (ORB)

Segundo Capeletto (1999, p. 19), objetos clientes requisitam serviços às implementações de objetos através de um ORB. O ORB é responsável por todos os mecanismos requeridos para encontrar o objeto, o mesmo prepara a implementação de um objeto para que receba a requisição que será executada. O cliente vê a requisição de forma independente de onde o objeto está localizado, em qual linguagem de programação ele foi implementado, ou qualquer outro aspecto que não esteja refletido na interface do objeto.

Segundo Riccioni (2000, p. 57), o modelo do ORB é flexível o suficiente para permitir que módulos e aplicações não orientadas a objetos possam ser conectadas ao sistema. Dado que nem todos os componentes do sistema sejam orientados a objetos, o poder de expressão da IDL deve ser independente da linguagem e do seu modelo, sem possuir características de uma dada linguagem que não possa ser compartilhada por outras, tais como: ponteiros de C, eventos *callback*, *broadcasting/multicasting*.

2.2.2 INTERFACE DEFINITION LANGUAGE (IDL)

Segundo a OBJECT MANAGEMENT GROUP (1998, p. 56), a IDL é a linguagem usada para descrever as interfaces que os objetos clientes solicitam e as implementações que os objetos distribuídos disponibilizam. Uma definição de interface escrita em IDL define completamente a interface e especifica totalmente cada parâmetro das operações, disponibilizando toda a informação necessária para desenvolver clientes que usem as operações desta interface. Os clientes não são escritos em IDL, que é puramente uma linguagem descritiva, mas sim em linguagens nas quais já tiveram mapeamentos definidos pelo OMG. O tipo de mapeamento depende das facilidades disponíveis na linguagem do

cliente.

Segundo Souza e Mazzola (1998, p.5), existem muitas vantagens em utilizar CORBA, podendo-se resumir como principal benefício a concentração dos esforços do desenvolvedor sobre as funcionalidades e requisitos da aplicação, ou seja, este poderia deixar de se preocupar com os problemas que podem surgir da distribuição dos processos e das eventuais conseqüências advindas da heterogeneidade do sistema, uma vez que as mesmas seriam resolvidas pela utilização de qualquer plataforma de hardware ou software.

2.3 EXTENSIBLE MARKUP LANGUAGE (XML)

Segundo Tittel (2003, p. 11), a *Extensible Markup Language* (XML) foi criada em 1996 e é um subconjunto da *Standard Generalized Markup Language* (SGML). O projeto original descreve uma linguagem de marcação que pudesse ser lida por software, que fosse orientada para a distribuição ampla na Internet, que pudesse ser integrada nas linguagens de marcação, tais como *Hypertext Markup Language* (HTML) e SGML, e que fosse facilmente legível. A XML torna-se especial, pois é: flexível, escalonável e adaptável, ou seja, XML pode se tornar qualquer coisa que um documento necessite que ela seja para distribuir informações na *web* ou entre aplicativos (software) – sem as restrições e limitações de HTML.

Também segundo Silva (2001, p. 18), o grande diferencial de XML é ser extensível, possibilitando a criação de elementos, assim, podem-se criar padrões, que sejam de interesse do desenvolvedor, facilitando o processo de interpretação das informações para os mais variados sistemas.

Os dados em XML podem ser unicamente "etiquetados", o que permite que, por exemplo, uma busca por livros seja feita em função do nome do autor. Atualmente, uma busca pelo nome do autor poderia levar a qualquer *site* que tivesse referência a tal nome, não

importando se fosse o autor do livro ou simplesmente um livro sobre o autor. Sem a XML é necessário para a aplicação de procura saber como é construído cada banco de dados que armazena os dados de interesse. A XML permitiria definir livros por autor, título, assunto e outros campos, o que facilitaria enormemente o cadastramento.

Segundo Light (1999, p. 74), os caracteres em documento XML representam uma combinação de caracteres de dados que é o verdadeiro texto do documento, e a marcação, que é informação que adiciona valor ao texto. A XML reconhece os seguintes tipos de marcação: *tags* inicias, *tags* finais, elementos vazios, referência a entidades, referência a caracteres, comentários, seções "CDATA", declarações do tipo de documentos, instruções de processamento.

2.3.1 Seções CDATA

Como "&" e "<" são caracteres especiais, estes não podem ser usados em qualquer lugar. Eles precisam ser "escapados[1]", para serem usados como parte do texto de um documento. Às vezes é monótono ter de "escapar" sempre que os caracteres "&" e "<" aparecem no texto, principalmente se o documento é muito extenso. A XML fornece seções "CDATA" para resolver esse problema.

Também segundo Light (1999, p. 76), seções "CDATA" podem ocorrer em qualquer parte em que dados de caracteres ocorrem. Elas são usadas para escapar blocos de textos que de outra forma seria interpretados como marcação. Seções "CDATA" começam com a seqüência "<! [CDATA[e terminam com a seqüência]]>", como no exemplo do Quadro 2.

```
<! [CDATA] [ <p> Once more we consider the wise words of
L&eacute;ger: ]]>
```

Fonte: Light (1999, p. 76).

Quadro 2 – Exemplo de instrução CDATA em um documento XML

[1] N.R.T.: Do inglês *escaped*¸ no sentido de "liberados".

Seções “CDATA” não podem ser aninhadas, ou seja, ocorrer dentro de outras seções “CDATA”.

2.3.2 Instruções de processamento

Segundo Light (1999. p. 79), instruções de processamento, contém instruções para as aplicações que processarão o documento XML. As instruções de processamento começam com a seqüência “<?”, seguida de um *name*, e eles são finalizados com “?>”. Como no exemplo do Quadro 3.

```
<?XML version = “1.0”?>
```

Fonte: Light (1999, p. 79).

Quadro 3 – Exemplo de instruções de processamento em um documento XML

As instruções de processamento não são parte dos dados de caracteres do documento, mas (ao contrário dos comentários) precisam passar pela aplicação.

2.3.3 O Elemento do documento

Segundo Light (1999, p. 85), o núcleo de um documento XML consiste exatamente de um elemento: o elemento do documento. Dentro deste elemento pode ser aninhado qualquer número de sub-elementos em qualquer nível e também pode ser incluída qualquer quantidade de texto. O ponto-chave é que um documento não pode consistir de mais de um elemento, e não pode ser parte de um elemento.

No Quadro 4, é apresentando uma seqüência de instruções que não se caracteriza como sendo um documento XML, porque contém dois elementos.

```
<?XML version = “1.0”?>
<greeting> Hello, world!</greeting>
<response>Hello, XML!</response>
```

Fonte: Light (1999, p. 85).
Quadro 4 – Exemplo de um documento XML formatado errado utilizando elementos

No Quadro 5, é apresentado um documento XML que tem um único elemento conversation que contém o greeting e a response.

```
<?XML version = “1.0”?>
<conversation>
<greeting> Hello, world!</greeting>
<response>Hello, XML!</response>
</conversation>
```

Fonte: Light (1999, p. 85).
Quadro 5 – Exemplo de um documento XML formatado corretamente utilizando elementos

2.3.4 Elementos vazios

Segundo Light (1999, p. 89), elementos vazios têm um formato especial. Como eles não têm conteúdo, não precisam ter uma *tag* inicial e uma *tag* final para ancorar esse conteúdo. Esse formato inclui uma “/” antes do final da *tag* inicial do elemento vazio. Como mostra o exemplo do Quadro 6.

```
<emptytag/>
```

Fonte: Light (1999, p. 89).
Quadro 6 – Exemplo de elementos vazios em um documento XML

2.3.5 *Tags* iniciais e *tags* finais

Segundo Light (1999, p. 88), para ser capaz de conter coisas, elementos precisam ter um tamanho finito. O tamanho finito de um documento é obtido através de *tags* inicias e *tags* finais, as quais marcam as delimitações de um elemento (seu começo e seu fim). No Quadro 7, temos um exemplo de utilização de *tags* inicias e *tags* finais.

```
<mytag> [o conteúdo vai aqui] </mytag>
```

Fonte: Light (1999, p. 89).

Quadro 7 – Exemplo de utilização de *tags* iniciais e finais em um documento XML

2.3.6 Comentários

Comentários podem ser usados para inserir anotações em seus documentos XML. Eles começam com "<! --", seguido do texto do comentário e finalizam com "-->". No Quadro 8, é apresentado um exemplo de comentário em um documento XML.

```
<!-- This is a comment. -->
```

Fonte: Light (1999, p. 78).

Quadro 8 – Exemplo de comentário em um documento XML

Na Figura 1 é apresentado um exemplo de dados textuais e o correspondente em linguagem XML.

Dados textuais

```
Catálogo de endereços
João Silva
Rua Carijós, 135
Belo Horizonte, MG 30.000
Brasil
31 3335-5556(preferido)
31 3549-4446
joaosilva@net.com.br
José Almeida
jalmeida@net.com.br
```

Correspondente em XML

```
<?xml version="1.0"?>
<catálogo de endereços>
   <entrada>
     <nome>João Silva</nome>
     <endereço>
        <rua>Carijós, 135</rua>
        <estado>MG</estado>
        <cep>30.000</cep>
        <país>Brasil</país>
     </endereço>
     <telefone preferido="true">31 3335-4456</telefone>
      <telefone>31 3594-4446</telefone>
      <email>joaosilva@net.com.br</email>
   </entrada>
   <entrada>
      <nome><prim>José</prim>
            <sobren>Almeida</sobren>
      <email>jalmeida@net.com.br</email>
   </entrada>
</catalogo de endereços>
```

Fonte: Almeida e Cendon (2003, p. 7).

Figura 1 – Exemplo de dados textuais e o correspondente em XML

2.4 MACHINE READABLE CATALOGING (MARC)

Com a automatização das bibliotecas a partir da década de 70, MARC foi adotado como padrão para representar documentos em bases de dados catalográficas. Novas necessidades aparecerem a partir da emergência da Internet e da explosão da disseminação da informação nos anos 90.

Segundo Almeida e Cendon (2003, p.3), o padrão MARC é composto por diversos campos padronizados, que contém representação de dados e metadados bibliográficos. Cada campo é identificado por uma seqüência de três dígitos (etiqueta), por exemplo: 100 para o campo autor, 130 para o campo título, 300 para o campo descrição física. Os campos podem conter subcampos.

O registro MARC contém sinalizadores que marcam o registro armazenado e auxiliam na leitura e interpretação desse registro. Os sinalizadores indicam o início e o término dos campos e subcampos. Por exemplo, ao invés de palavras, usam-se os códigos 260 $a $b $c, para marcar o campo que contém os subcampos “área de publicação”, “local de publicação”, “nome da editora” e “data de publicação” em cada registro.

Os sinalizadores MARC auxiliam os computadores na leitura e interpretação do registro, marcando o registro bibliográfico para armazenamento em meio magnético. Contudo os registros do padrão MARC não podem ser publicados na *web*, pois seu formato complexo não pode ser interpretado pelos navegadores.

A figura 2 apresenta um fragmento de um registro, mostrando os sinalizadores de texto e seus correspondentes no padrão MARC.

Dados	Sinalizadores de texto	Sinalizadores MARC		
		Campos	Indicadores	Subcampos
Arnosky, Jim	Entrada principal, nome pessoal com sobrenome simples	100	1#	$a
Raccoons and ripe corn / Jim Arnosky	Área de título Meção de responsabilidade	245	10	$a $c
New York : Lothrop, Lee & Shepard Books, c1987.	Local de publicação Nome da editora Data de publicação	260	##	$a $b $c
25 p.; col. Ill.; 26 cm.	Paginação Ilustrações Dimensão	300	##	$a $b $c

Fonte: Almeida e Cedon (2003, p. 4).

Figura 2 - Simbologia do padrão MARC

Almeida e Cedon (2003, p. 4), descreve que, a figura 2 ilustra os sinalizadores para campos, indicadores e subcampos. O numero “100” corresponde à etiqueta que representa o campo onde está o nome do autor.

Na primeira linha da Figura 2, os indicadores para o campo “100” são os caracteres “1” e “#”, o símbolo “#” indica que o indicador não é usado e o valor “1” significa que deverá haver uma entrada de título no catálogo.

Além dos campos, existe também os sub-campos, que são representados por letras minúsculas, como na Figura 2, onde o campo “300” tem o sub-campo “a” que representa o número de páginas, os sub-campos são precedidos por delimitadores que podem ser representados por diferentes símbolos.

Em MARC Standards (2004), um registro bibliográfico MARC 21 é uma atualização do MARC tornando-se um padrão para representação e comunicação de informações bibliográficas relatada em forma legível por máquina (*machine-readable*).

2.5 TRABALHOS CORRELATOS

Em Ferrari (2000) é descrito um protótipo de sistema de consulta de preços de supermercados, utilizando objetos distribuídos via Internet. Neste trabalho o autor demonstra

as principais tecnologias envolvidas como objetos distrubuídos, *Active Server Pages* (ASP), *applications server* e as técnicas de modelagem em multicamadas. Ferrari (2000) menciona em seu trabalho a utilização de outras tecnologias e produtos que, segundo o próprio autor, não foram tão relevantes para o desenvolvimento do projeto e não precisaram ser profundamente estudadas, nas quais cito o SQL Server 7.0 para o banco de dados e o modelo de componentes distribuídos DCOM. Segundo Ferrari (2000), com um "banco de dados" extremamente grande como se tornou a Internet ficou muito difícil de reunir as informações sobre determinados produtos, e ao mesmo tempo fazer comparações entre os produtos pesquisados, com isso surgiu à idéia de desenvolver uma aplicação web que permitiria aos usuários dos *sites* pesquisar e descobrir quais os estabelecimentos que teriam os melhores preços.

Em Crispim e Fernandes (2004), baseado no Prontuário Eletrônico de Pacientes (PEP), foi desenvolvido um sistema de prontuário eletrônico integrado do paciente para as clínicas do centro de saúde da Universidade do Vale do Itajaí (PEP/UNIVALI). O PEP é uma aplicação computacional que visa fornecer facilidades e serviços para que os profissionais da área médica possam manter um acompanhamento efetivo do histórico clínico dos seus pacientes. Em um PEP distribuído, a arquitetura do sistema está organizada em camadas de software, que podem estar localizadas em máquinas geograficamente distantes, interligadas por redes de comunicação. O processamento e os dados manipulados pelo PEP ficam distribuídos e acessíveis aos profissionais da área médica a partir de qualquer localidade, sejam hospitais, centros de saúde, policlínicas e unidades conveniadas. Este prontuário foi desenvolvido em PHP e o banco de dados utilizado foi o Oracle. Para integrar as informações das diversas clínicas foi necessário criar um vocabulário unificado com base na Classificação Internacional de Doenças (CID) e a partir disto, criar visões específicas para cada área dentro do sistema. Além disso, para permitir troca de informações entre instituições, foi necessário utilizar XML

como uma forma de representação de documentos que fosse padronizada e independente de plataforma de hardware e software.

3 DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo são descritos os requisitos e as especificações do protótipo. São apresentados também, alguns detalhes dos softwares e componentes utilizados, bem como os demais detalhes para a implementação do protótipo. Em seguida, é apresentada a operacionalidade do protótipo. Por fim, os resultados obtidos durante os testes de funcionamento e conexão.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

A ferramenta deverá:

a) permitir cadastramento de usuários no módulo para editoras (requisito funcional – RF);
b) permitir gerenciamento das informações referente aos livros cadastrados no módulo para editoras (RF);
c) permitir gerenciamento das informações referente ao padrão de catalogação no módulo para editoras (RF);
d) permitir escolha das editoras que serão consultadas no módulo para livrarias (RF);
e) possuir uma área para visualização das informações cadastradas (RF);
f) possuir controle de acesso para usuários do módulo para editoras (RF);
g) possuir controle de acesso para usuários do módulo para livrarias (RF);
h) utilizar tecnologia de gerenciamento de objetos CORBA (requisito não funcional – RNF);
i) ser implementado utilizando o ambiente Delphi 6.0 e banco de dados Interbase 6.0 (RNF);

j) utilizar padrão para metadados de livros MARC 21 (RNF).

3.2 ESPECIFICAÇÃO

Nesta seção é apresentada a especificação dos objetos para o desenvolvimento do protótipo, e também a realização de um estudo do padrão MARC 21, onde houve necessidade de adaptá-lo para que o mesmo pudesse ser utilizado para armazenar as informações de livros das editoras e livrarias.

3.2.1 Adaptação do registro MARC 21

A adaptação do registro MARC 21 se deu através de uma análise criteriosa do padrão que hoje é utilizado internacionalmente por bibliotecas, o qual, precisou ser adequado para as exigências de implementação desta aplicação, ou seja, como o MARC 21 é um padrão utilizado para diversos tipos de mídias, houve a necessidade de adaptá-lo para utilização somente com dados referente a livros e que fossem pertinentes à utilização pelas editoras. Após o trabalho de adaptação do registro MARC 21, foi determinado que os melhores campos para esta aplicação seriam os descritos no anexo A.

3.2.2 Especificação dos objetos

A especificação dos objetos para a implementação foi desenvolvida usando a UML, através de quatro diagramas distintos: diagramas de caso de uso, diagrama de classes, diagramas de seqüência e o diagrama de estados. Para a confecção destes diagramas, utilizou-se a ferramenta Enterprise Architect 4.0.

3.2.2.1 Especificação do módulo para editoras

Na figura 3 tem-se o diagrama de casos de uso referente ao acesso pelo ator denominado de editor, ao sistema de manutenção de dados das editoras.

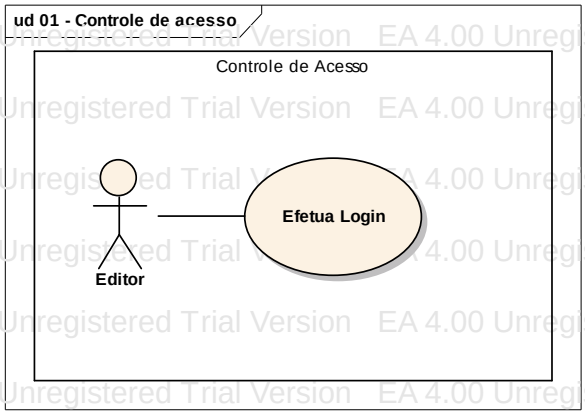


Figura 3 – Diagrama de Casos de uso acesso módulo editoras

No diagrama da figura 3 tem-se o acesso do editor, ou seja, a pessoa autorizada para utilização do sistema. O mesmo efetua *login* na máquina, mediante a um usuário e uma senha, assim pode efetuar a manutenção do banco de dados.

Na figura 4, tem-se ilustrado o diagrama de casos de uso referente aos cadastros que o editor pode efetuar no módulo para editoras.

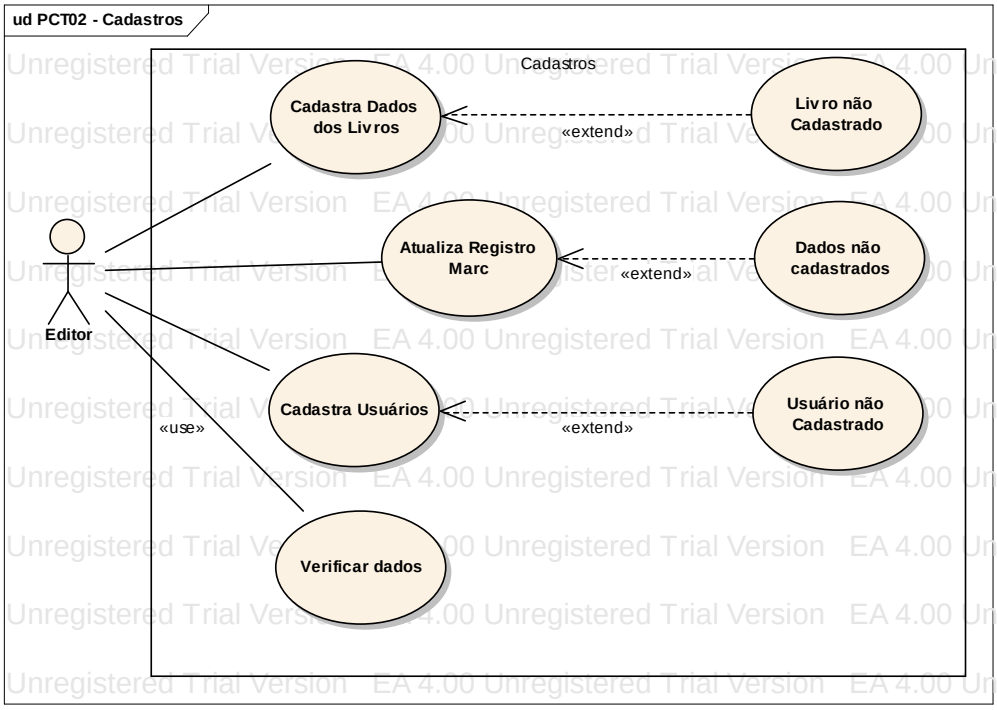


Figura 4 – Diagrama de casos de uso referente aos cadastros do módulo editoras

O diagrama da figura 4 mostra-nos que, através de uma situação inesperada (*extends*), ou seja, de uma necessidade externa o editor poderá efetuar o cadastramento dos dados referente a livros, também poderá realizar atualizações no registro MARC que está armazenado no banco, bem como poderá cadastrar os usuários que terão acesso à manutenção ou consultas ao banco de dados. É importante salientar que, a verificação de dados por estar sendo compartilhada com o módulo de livrarias utilizada uma relação *uses*.

3.2.2.2 Especificação do módulo para livrarias

Na figura 5, é apresentado o diagrama de casos de uso referente ao acesso pelo ator denominado de livreiro, ao sistema de consulta de dados das livrarias.

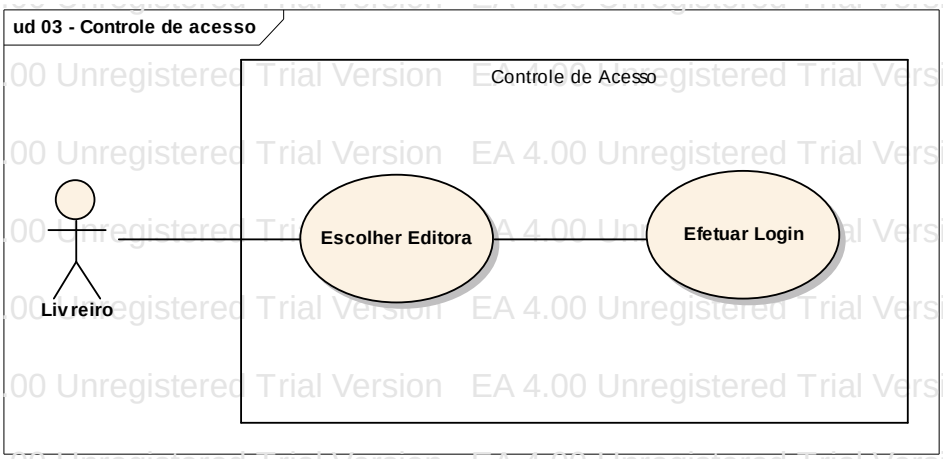


Figura 5 – Diagrama de caso de uso referente ao acesso ao sistema de consulta das livrarias

No diagrama da figura 5, tem-se o acesso do livreiro, ou seja, a pessoa autorizada para utilização do sistema, o mesmo primeiramente escolhe a editora que deseja se conectar para acessar os dados para consulta. Após escolher a mesma, o livreiro efetua o *login* na máquina, mediante a um usuário e uma senha.

Na figura 6 é apresentado o diagrama de casos de uso referente à consulta de dados.

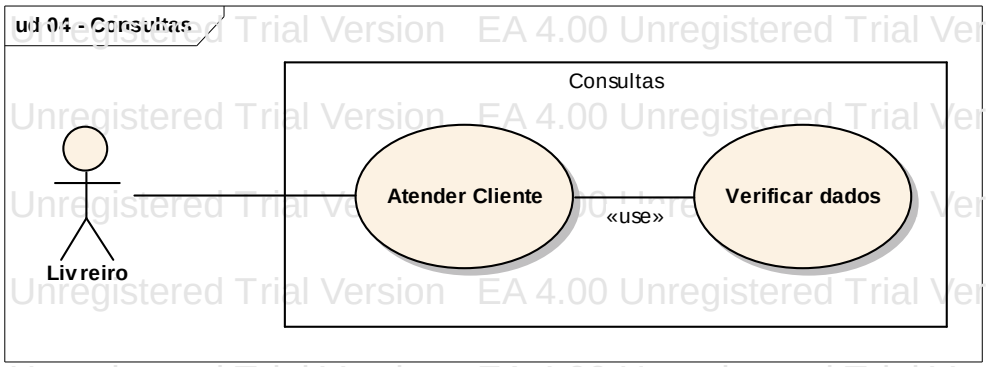


Figura 6 – Diagrama de casos de uso referente à consulta de dados

No diagrama da figura 6 tem-se a representação do livreiro atendendo um cliente e efetuando uma consulta aos dados armazenados no banco que será atualizado e disponibilizado pelas editoras.

A figura 7 ilustra o diagrama de estados referente à consulta de livros disponibilizada no módulo de livrarias.

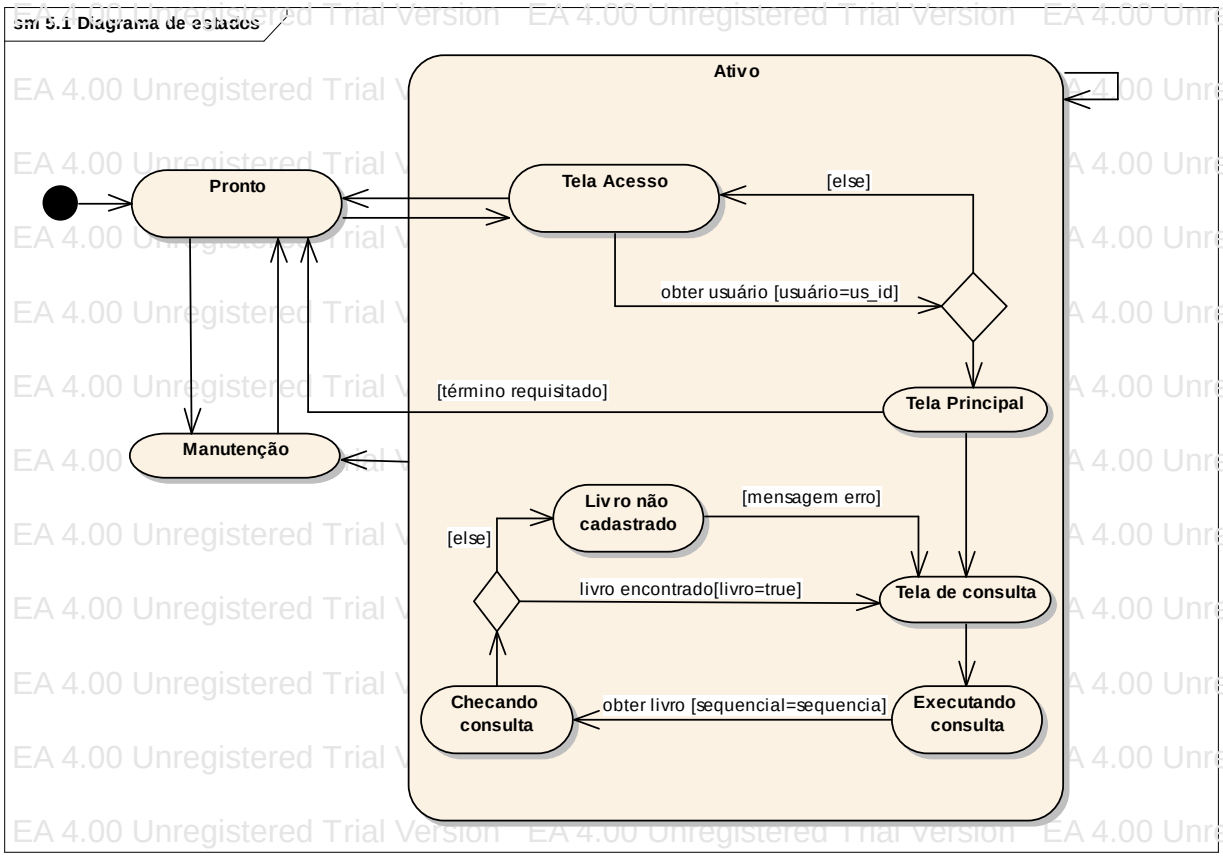


Figura 7 – Diagrama de estados referente à consulta de livros do módulo para livrarias

Na figura 7 é possível observar que, o sistema parte de um estado inicial, que neste caso seria o terminal disponibilizado para utilização (pronto), pois o mesmo poderia estar

indisponível por algum problema técnico e, conseqüentemente, estar em manutenção. Se estiver tudo funcionando corretamente na estação de consulta, o usuário poderá fazer o acesso através de seu nome de usuário e sua senha, o mesmo estando cadastrado no banco o sistema acessa a tela principal, senão, volta para o estado anterior de acesso. Ao acessar a tela de consulta o usuário deve clicar no menu para efetuar a pesquisa referente ao livro. Através de um parâmetro fornecido pelo usuário, o sistema busca a informação no banco de dados, se o livro for encontrado, o mesmo apresenta os dados na tela de consulta, senão aparece mensagem de erro.

3.3 IMPLEMENTAÇÃO

Nesta seção são apresentados os principais componentes e as ferramentas utilizadas para a implementação do protótipo.

3.3.1 Técnicas e ferramentas utilizadas

Nesta seção são apresentadas as técnicas e as ferramentas utilizadas para a construção do protótipo. O conteúdo é dividido em partes, sendo que a primeira parte mostra a construção do banco de dados utilizado na aplicação, seguindo o padrão de catalogação adaptado do MARC que pode ser visualizado no anexo A, a segunda apresenta o desenvolvimento do servidor que hospeda o banco de dados que é distribuído para os módulos e a terceira e última parte mostra a implementação dos módulos para editoras e livrarias.

3.3.1.1 Estrutura do banco de dados após adaptação do registro MARC

A figura 8 apresenta o diagrama de entidade e relacionamento (ER) do protótipo, onde pode ser observada a estrutura das tabelas criadas e seus respectivos relacionamentos.

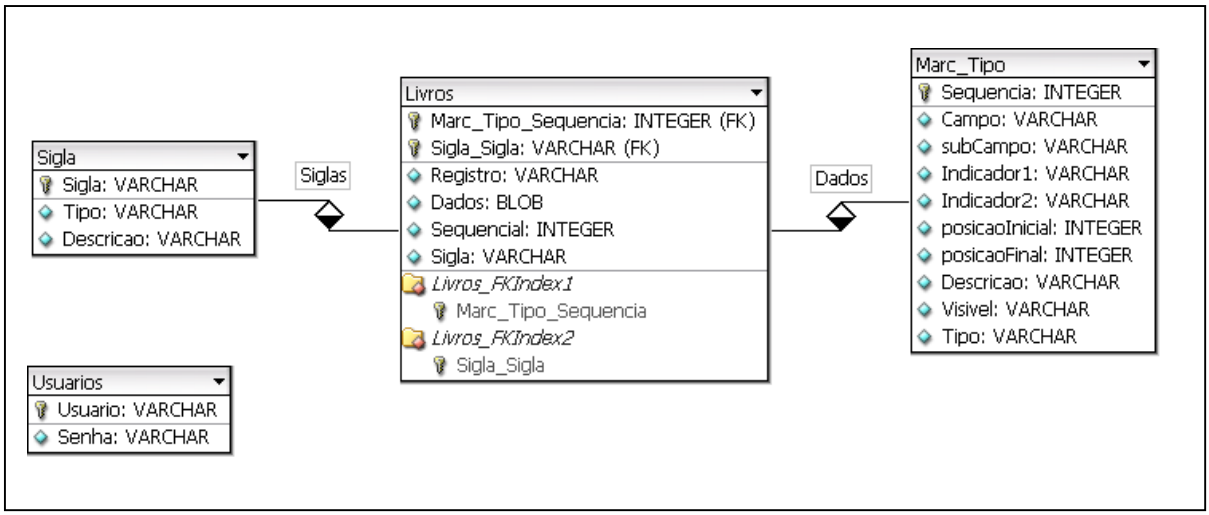


Figura 8 – Diagrama de entidade e relacionamento do protótipo

Assim através da figura 8 pode-se observar que o banco de dados foi estruturado de acordo com os campos retirados do padrão MARC 21, onde foi utilizada a nomenclatura original do padrão para armazenamento dos dados na tabela “Marc_Tipo”.

Na estrutura das tabelas foram criados campos e subcampos além de indicadores e um campo para armazenamento dos dados em três tabelas: a tabela “Marc_Tipo”, “Sigla” e a tabela “Livros”, onde, a tabela “Marc_Tipo” contém todo o padrão cadastrado com seus respectivos campos, subcampos e indicadores e um campo para ligação com a tabela “Livros”.

No quadro 9 tem-se a descrição dos campos e suas respectivas tabelas conforme diagrama apresentado na figura 8.

TABELA	CAMPO	DESCRIÇÃO
LIVROS	Registro	Número responsável pela identificação dos dados referente ao livro cadastrado.
	Dados	Campo responsável pelo armazenamento das informações de cada livro.
	Sequencial	Campo que liga a tabela de "Livros" com a tabela "Marc_Tipo", responsável pelos itens para o cadastramento das informações referente aos livros.
	Sigla	Campo que liga a tabela de "Livros" com a tabela "Siglas", responsável por obter as siglas correspondentes aos livros cadastrados.
MARC_TIPO	Sequencia	Campo responsável por disponibilizar as informações da tabela "Marc_Tipo" para a tabela “Livros” referente aos itens de cadastramento dos livros.
	Campo	Refere-se ao "Campo" do registro MARC 21 onde contém todas os números que identificam as informações que estão sendo cadastradas.
	subCampo	Refere-se ao "Sub-Campo" do registro MARC 21 onde contém todos os identificadores responsáveis pelos itens que devem ser seguidos para o cadastramento dos livros.
	Indicador1	Item padrão do registro MARC 21.
	Indicador2	Item padrão do registro MARC 21.
	posicaoInicial	Identifica a posição inicial nos registros 008 do padrão MARC 21 para os "Sub-Campos": Ilustrações e Natureza e conteúdo da obra.
	posicaoFinal	Identifica a posição final nos registros 008 do padrão MARC 21 para os "Sub-Campos": Ilustrações e Natureza e conteúdo da obra.
	Descrição	Refere-se à descrição de cada “sub-campo” do registro MARC 21.
	Visível	Campo responsável pela visualização dos itens do registro para consulta.
	Tipo	Responsável pelo armazenamento dos tipos de siglas referente à tabela "Sigla".
SIGLA	Sigla	Campo que armazena o código referente às siglas definida no padrão MARC 21.
	Tipo	Campo que armazena o tipo de sigla que será utilizado pela aplicação onde tem-se o código da área geográfica (GAC), código da linguagem (LC) e o código do relator (RC).
	Descrição	Campo que armazena a descrição dos itens contidos no campo “Sigla”.

Quadro 9 – Descrição dos campos das tabelas do protótipo

O quadro 10 mostra a parte do registro MARC que contém o campo e subcampos relativos ao autor de um livro.

```
100 – ENTRADA PRINCIPAL NOME PESSOAL – AUTOR (NR)
   Indicador
         0 – Parte Nome
         1 - Sobrenome
         3 – Nome de Família
   SubCampos
      $a – Nome pessoal (NR)
      $b - Numero(NR)
      $c – Os títulos e outros que exprimem associados com um conhecido
      $d – Datas associadas com um conhecido (NR)
      $e – Termo de relator (R)
      $f – Data de um trabalho (NR)
      $g – Informações variadas (NR)
      $j – Qualificador de atribuição (R)
      $k – Dar forma ao subtítulo (R)
      $l – Linguagem de um trabalho (NR)
      $n – Número da parte/Seção de um trabalho (R)
      $p – Nome da parte/Seção de um trabalho (R)
      $q – Formulário mais completo do nome contido no campo $a (NR)
      $t – Título de um trabalho (NR)
      $u - Afiliação (NR)
      $4 – Código de relator (R)
```

Quadro 10 – Campo 100 do MARC 21 e seus respectivos subcampos (traduzido)

No quadro 10 pode-se observar melhor de onde foram retirados os dados para montagem da tabela MARC_TIPO, como exemplo, temos o campo (100), responsável pelo armazenamento dos dados referente ao autor da obra, também possui um indicador que no caso deste item do MARC, possui só um indicador, mas pode conter até dois indicadores (Indicador1 e Indicador2), também pode-se observar a existência dos subcampos, onde cada um representa um dado que é armazenado na tabela referente ao livro.

As tabelas para o banco foram estruturadas no DBDesigner 4, e podem ser visualizadas no diagrama de entidade e relacionamento da figura 8.

3.3.1.2 Implementação do servidor CORBA

A implementação do servidor para distribuição da base de dados foi desenvolvida de acordo com as especificações apresentadas no item 3.2.2. O software foi desenvolvido no

Delphi 6 por conter maior suporte as tecnologias CORBA. Para base de dados utilizou-se o banco de dados Interbase 7.0. Também foi utilizado como ORB o VISIBROKER 3.3.

O Delphi 6 possui muitos componentes que facilitam a operação de programação deixando o sistema com uma interface amigável e a comunicação com banco de dados eficiente e portável. Para esta aplicação foram utilizadas muitas das facilidades que o Delphi 6 possui. Para efetuar a comunicação das interfaces clientes com o servidor, foi utilizado o CORBA *Data Module Wizard* que acompanha o Delphi 6 para gerar os objetos referente ao módulo de dados que será distribuído para as aplicações clientes.

Para efetuar as conexões com o banco estão sendo utilizados os componentes da paleta DBExpress, que acompanham o Delphi 6, e tem como principal característica a facilidade de acesso a múltiplos bancos de dados. Os componentes da paleta DBExpress utilizados foram: o TSQLConnection e o TSQLDataSet.

O TSQLConnection define os parâmetros utilizados para o estabelecimento da conexão com o banco de dados. Este componente interage com um *driver* DBExpress e com dois arquivos do tipo INI (DBXdrivers.ini e DBXconnections.ini). Estes arquivos listam:

a) os tipos de *drivers* instalados (InterBase, MsSql, Oracle, etc.);

b) os DLLs referentes a cada driver;

c) os atributos padrão a cada driver.

O arquivo DBXconnections.ini lista: os nomes dos conjuntos de conexão existentes, cada qual representando uma conexão específica a um banco de dados.

O TSQLDataSet é um importante componente de acesso a dados. Foi utilizada sua propriedade CommandType, para definir a utilização de *query*, ou seja, em sua propriedade CommandType foi adicionado instruções SQL utilizadas pelos módulos distribuídos da aplicação. Estes dois componentes ficam localizados no módulo de dados CORBA que foi gerado, fazendo a ligação das tabelas do banco de dados com os módulos distribuídos da

aplicação.

No quadro 11, são listados os componentes utilizados no módulo de dados CORBA deste protótipo.

TDataSetProvider	TSQLDataSet	TSQLConnection
sqlConsultaLivros	dsConsultaLivros	sqlCGerLivros
sqlDsLivros	dspLivros	
sqlDsIlustracoes	dspIlustracoes	
sqlDsNivel	dspNivel	
sqlDsFormato	dspFormato	
sqlDsNatureza	dspNatureza	
sqlDsPubGov	dspPubGov	
sqlDsForma	dspForma	
sqlDsBiografia	dspBiografia	
sqlDsUsuarios	dspUsuarios	
sqlDsNBN	dspNBN	
sqlDsISBN	dspISBN	
sqlDsEAN	dspEAN	
sqlDsIndEAN	dspIndEAN	
sqlDsIlus19	dspIlus19	
sqlDsIndAutor	dspIndAutor	
sqlDsAutor	dspAutor	
sqlDsAutCorp	dspAutCorp	
sqlDsIndAutCorp	dspIndAutCorp	
sqlDsIndTitAbreviado	dspIndTitAbreviado	
sqlDsTitAbreviado	dspTitAbreviado	
sqlDsTraducao	dspTraducao	
sqlDsIndTraducao	dspIndTraducao	
sqlDsFonteCat	dspFonteCat	
sqlDsLinguagem	dspLinguagem	
sqlDsIndLinguagem	dspIndLinguagem	
sqlDsAreaGeog	dspAreaGeog	
sqlDsIndTitulo	dspIndTitulo	
sqlDsTitulo	dspTitulo	
sqlDsEdicao	dspEdicao	
sqlDsDscFisica	dspDscFisica	
sqlDsDtPublicacao	dspDtPublicacao	
sqlDsIndDtPub	dspIndDtPub	
sqlDsAssunto	dspAssunto	
sqlDsMARC	dspMARC	
sqlDsSiglas	dspSiglas	
sqlDsGAC	dspGAC	
sqlDsLC	dspLC	
sqlDsRC	dspRC	
sqlDsNotas	dspNotas	
sqlDsIndSumario	dspIndSumario	
sqlDsSumario	dspSumario	
sqlDsCdUsuarios	dspCdUsuarios	
sqlDsPublicador	dspPublicador	
sqlDsIndPub	dspIndPub	
sqlDsSerie	dspSerie	

Quadro 11 – Lista de componentes utilizados no módulo de dados CORBA

O quadro 11 apresenta a lista de componentes utilizados no módulo de dados CORBA que é responsável pela comunicação dos módulos distribuídos do protótipo com a base de dados. No quadro 11 são apresentados os componentes TSQLConnection, TSQLDataSet e TDataSetProvider, que em conjunto fazem a obtenção dos dados do banco e conseqüentemente a distribuição destes para as telas dos módulos distribuídos.

O quadro 12, apresentada a rotina responsável por criar o módulo de dados CORBA.

```
implementation

{$R *.DFM}

uses CorbInit, CorbaVcl;

initialization
  TCorbaVclComponentFactory.Create('SrvCorbaFactory', 'SrvCorba',
'IDL:Servidor/SrvCorbaFactory:1.0', ISrvCorba,
    TSrvCorba, iMultiInstance, tmSingleThread);
end.
```

Quadro 12 – Rotina de criação de módulo de dados CORBA

No quadro 12 tem-se a rotina que cria os objetos que serão utilizados pelo módulo de dados através do componente *factory*, este é um componente nativo da arquitetura CORBA.

Quando se utiliza os componentes da arquitetura CORBA, o Delphi gera automaticamente o arquivo Servidor_TLB.pas que tem por finalidade obter a referência ao objeto para o servidor. Abrindo o arquivo Servidor_TLB.pas dentro do Delphi, este apresentará a tela do *Type Library*, no qual permite a especificação de todas as informações necessárias à definição da interface do objeto CORBA.

A figura 9 apresenta a tela do *Type Library* com a interface (ISrvCorba) e a classe (SrvCorba) especificadas para geração do arquivo Servidor.idl.

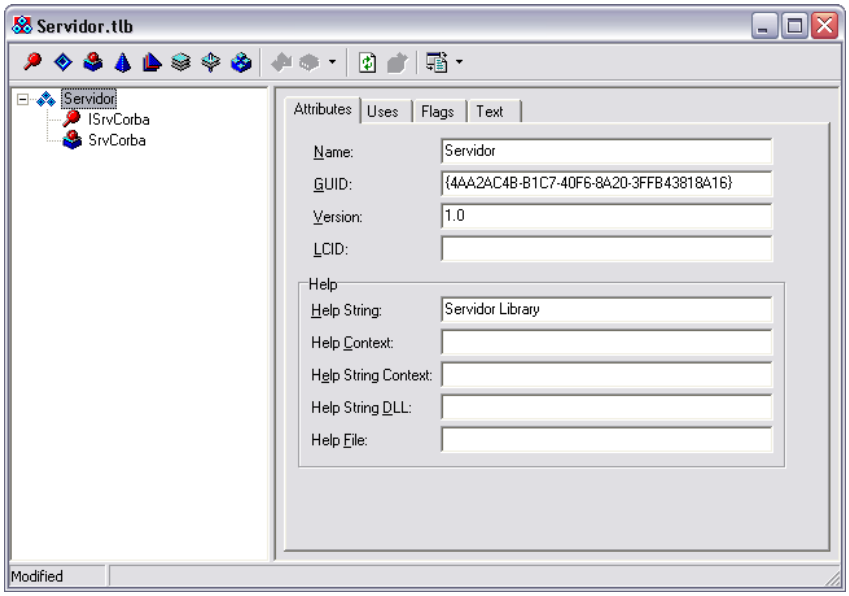


Figura 9 – Tela *Type Library* que gera o arquivo Servidor.idl

A partir do editor *Type Library* apresentado na figura 9, foi gerado o arquivo Servidor.idl, com a rotina descrita no quadro 13.

```
[
  uuid(4AA2AC4B-B1C7-40F6-8A20-3FFB43818A16),
  version(1.0),
  helpstring("Servidor Library")
]
library Servidor
{
  importlib("midas.dll");
  importlib("stdole2.tlb");
  importlib("stdvcl40.dll");
  [
    uuid(114DE038-E38C-4CD9-8EC9-F74A7D0CD149),
    version(1.0),
    helpstring("Dispatch interface for SrvCorba Object"),
    dual,
    oleautomation
  ]
   interface ISrvCorba: IAppServer
  {
  };
  [
    uuid(B7B9A7CC-250E-4277-9C95-5FD4749015ED),
    version(1.0),
    helpstring("SrvCorba Object"),
    custom(B0FC9343-5F0E-11D3-A3B9-00C04F79AD3A, 0)
  ]
  coclass SrvCorba
  {
    [default] interface ISrvCorba;
  };
};
```

Quadro 13 – Arquivo Servidor.idl gerado pelo editor *Type Library*

No quadro 13 é apresentada a rotina que descreve a interface criada neste protótipo. Esta rotina disponibilizará as informações necessárias para desenvolver clientes que necessitem utilizar as operações desta interface.

No quadro 14 é apresentada a rotina de criação do *skeleton* para ser utilizado pelo servidor CORBA, esta rotina se encontra no arquivo Servidor_TLB.pas.

```
constructor TSrvCorbaSkeleton.Create(const InstanceName: string; const
Impl: IUnknown);
begin
  inherited;
  inherited InitSkeleton('SrvCorba', InstanceName,
'IDL:Servidor/ISrvCorba:1.0', tmMultiThreaded, True);
  FIntf := Impl as ISrvCorba;
end;
```

Quadro 14 – Rotina de criação do *skeleton* utilizado pelo servidor

Na rotina apresentada no quatro 14, tem-se o construtor da classe *skeleton* onde é definido o nome do objeto CORBA, uma referência a instância do nome que se dá através de uma constante (InstanceName), a IDL que define a interface de informações que serão trocadas, a definição de que o servidor funcione com múltiplas *threads* e por fim a classe construtora através da variável “FIntf” recebera o objeto gerado.

No quadro 15, é apresentada a rotina referente à localização ao objeto “Servidor” implementado.

```
procedure TSrvCorbaSkeleton.GetImplementation(out Impl: IUnknown);
begin
  Impl := FIntf;
end;
```

Quadro 15 – Localização do objeto servidor CORBA

Após a criação do *skeleton* a variável “Impl” localiza o objeto implementado.

No quadro 16 tem-se a rotina que contém as classes e funções de criação do objeto, bem como a criação do objeto remoto e a instanciação da interface CORBA através do CORBA *factory*.

```
class function CoSrvCorba.Create: ISrvCorba;
begin
  Result := CreateComObject(CLASS_SrvCorba) as ISrvCorba;
end;

class function CoSrvCorba.CreateRemote(const MachineName: string):
ISrvCorba;
begin
  Result := CreateRemoteComObject(MachineName, CLASS_SrvCorba) as
ISrvCorba;
end;

class function TSrvCorbaCorbaFactory.CreateInstance(const InstanceName:
string): ISrvCorba;
begin
  Result := CorbaFactoryCreateStub('IDL:Servidor/SrvCorbaFactory:1.0',
'SrvCorba',
    InstanceName, '', ISrvCorba) as ISrvCorba;
end;
```

Quadro 16 – Rotina de criação de classes e funções do objeto CORBA

Na rotina do quadro 16, é apresentada primeiramente a criação do objeto que recebe a classe do objeto criado “SrvCorba”, em seguida é criado o objeto responsável pelos acessos remotos da aplicação com o objeto “SrvCorba”, por fim temos a criação e instanciação do *stub* que recebe a interface do objeto criado, todas as funções devolvem o objeto “ISrvCorba” às rotinas de registro.

O quadro 17 tem por finalidade mostrar a rotina responsável pelos registros do objeto CORBA.

```
initialization
  CorbaStubManager.RegisterStub(ISrvCorba, TSrvCorbaStub);
  CorbaInterfaceIDManager.RegisterInterface(ISrvCorba,
'IDL:Servidor/ISrvCorba:1.0');
  CorbaSkeletonManager.RegisterSkeleton(ISrvCorba, TSrvCorbaSkeleton);
end.
```

Quadro 17 – Rotina de registro do objeto CORBA

Na rotina do quadro 17, tem-se o registro do *stub*, da *interface* e do *skeleton* que foi criado na rotina apresentada no quadro 16.

Com isso tem-se o servidor CORBA implementado e devidamente conectado ao banco de dados que possibilita a integração com os módulos distribuídos do protótipo.

3.3.1.3 Implementação dos módulos para editoras e livrarias

A implementação do módulo para editoras foi desenvolvida de acordo com as especificações apresentadas no item 3.2.2.1 e o módulo para livrarias foi desenvolvido utilizando as especificações contidas no item 3.2.2.2.

Em CORBA há o método *early binding* ou *static binding* para obter uma instância de um objeto servidor, com isto a aplicação cliente tem conhecimento do tipo de objeto CORBA com o qual irá se comunicar, isso é possível através da classe *stub*, que é usada para controlar a passagem dos dados entre o cliente e o servidor (processos conhecidos como *marshaling* e *unmarshaling*).

O cliente *early binding* usa o arquivo gerado pelo *Type Library* para obter uma referência ao objeto CORBA que o servidor irá disponibilizar.

Para a comunicação com o servidor, foi utilizado o componente TCorbaConnection do Delphi 6, este componente, em conjunto com os componentes TClientDataSet e TDataSource, formam o conjunto para a conexão, obtenção e atualização dos dados no servidor.

O componente TCorbaConnection conecta-se ao servidor através de um *Internet Protocol* (IP) que é adicionado a propriedade HostName, também é definida a interface do objeto em seu RepositoryID e a especificação do nome do servidor em sua propriedade objetctName, assim é possível executar a aplicação “cliente”. O *ORB Smart Agent* deve estar executando em algum lugar da rede. Para executar o *ORB Smart Agent,* é necessário ter o Visibroker 3.3 instalado na máquina que executará o aplicativo servidor, e executar “osagent –c” a partir da linha de comando. Após a execução do *Smart Agent*, deve ser executado o aplicativo servidor do protótipo.

Para verificar se existe algum objeto servidor disponível na rede para ser utilizado pelo “cliente”, basta executar na máquina “cliente” o comando “osfind” a partir da linha de

comando. Este comando mostra uma lista dos objetos disponíveis na sub-rede da máquina. Isso verifica se o cliente tem acesso aos objetos do servidor.

Com o servidor em execução é possível conectar o TCorbaConnection que está em execução nas aplicações “clientes” do protótipo, assim foi conectado o TClientDataSet ao provedor de dados (TDataSetProvider) no lado do servidor, ao fazer esta conexão e ativar o componente TClientDataSet, o mesmo faz uma requisição de dados ao componente TDataSetProvider.

Na realidade, quando se ativa um componente TClienteDataSet, ele faz uma requisição de dados ao componente TDataSetProvider a ele associado.

Para possibilitar a navegação e edição de dados, o componente TClientDataSet mantém uma cópia dos dados na memória (*buffer*) para que eles possam ser alterados e consultados.

Neste contexto, qualquer operação de inclusão, alteração e exclusão é mantida em memória e os dados reais continuam inalterados.

Para que eventuais inclusões, alterações ou exclusões sejam efetivamente concluídas, é preciso ativar o método *ApplyUpdates* do componente TClientDataSet.

Os campos da aplicação são conectados ao T*ClientDataSet* através de um componente DataSource, com isto o resultado da *query* que foi introduzida nos componentes SqlDataSet podem ser utilizados por componentes tipo TData-Aware (DBGrid, DBEdit, etc) do Delphi.

Neste protótipo podem se destacar as rotinas que utilizam os métodos *FindKey* e *FindField* do componente TClientDataSet mostrados no quadro 18.

```
procedure TfrmAcesso.btOKClick(Sender: TObject);
   .
   .
   .
    {Se o nome digitado no controle edUsuario não for localizado
     no campo Usuario da tabela de Usuarios, faça}
    if not cdsUsuarios.FindKey([edUsuario.text]) then
    begin
      //Apresenta menssagem no panel
      pnlMensagem.Caption := 'Login Recusado, Usuário Desconhecido!';
      //Incrementa 1 à variável contador
      inc(Contador);
      //Se contador for igual a 4, feche o formulário principal
      if Contador = 4 then
        //Fecha o formulário principal
        frmPrincipal.Close;
      //O controle edUsuario recebe o foco
      edUsuario.SetFocus;
      //Sai do procedimento
      Exit;
    end;
    {A variavel valorSenha recebe o valor do campo senha
     da tabela usuario}
    valorSenha := cdsUsuarios.FindField('SENHA') as TstringField;
    {Se o valor armazenado na variavel valorSenha for diferente
     ao valor digitado no controle edUsuario, faça}
    if valorSenha.Value <> edSenha.Text then
      begin
        //Apresenta menssagem no panel
        pnlMensagem.Caption := 'Login Recusado, Senha Inválida!';
        //incrementa 1 à variável contador
        inc(Contador);
        //Se contador for igual a 4, feche o formulário principal
        if Contador = 4 then
          //Fecha o formulário principal
          frmPrincipal.Close;
        //o controle edSenha recebe o foco
        edSenha.SetFocus;
        //Sai do procedimento
        Exit;
      end;
    .
    .
    .
```

Quadro 18 – Rotina de acesso utilizando os métodos *FindKey* e *FindField*

No quadro 18 é apresentado o método *FindKey* onde, após o usuário digitar seu nome no componente TEdit (edUsuario) da tela de acesso, o parâmetro da pesquisa é enviado através do componente TClientDataSet (cdsUsuarios) através da sua propriedade *params* ao componente provedor de dados (TDataSetProvider) no servidor CORBA, onde, o método *FindKey* realiza uma busca em todos os registro da tabela usuários até encontrar ou não o registro que corresponda ao parâmetro passado para verificação.

Ao executar a linha que contém o método *FindField*, este obtém do banco de dados o valor do campo senha especificado no parâmetro e armazena o mesmo na variável “valorSenha” que retorna as mensagens de erro da aplicação ou executa a tela principal.

No módulo de livrarias é interessante observar duas rotinas de obtenção de dados. A rotina que efetua a busca de livros diretamente do banco de dados, e a rotina que busca livros do *cache* armazenado localmente em um arquivo XML (livros.xml), gerado pela aplicação após cada consulta efetuada no banco.

O quadro 19 apresenta a rotina de busca dentro do arquivo XML criado depois de efetuada uma consulta ao banco de dados.

```
function TfrmConsulta.pesquisaXML(pesquisa: boolean): boolean;
var
  dom : IXMLDOMDocument;
  endCsXML : IXMLDOMNodeList;
  i : integer;
  paramCsXML : widestring;
  resultado : boolean;
begin
  pesquisa := false;
  resultado := true;
  Resultado := verificaHora(Resultado);
  if Resultado = false then
    begin
      cdsXMLConsulta.Active := false;
      dom := CoDOMDocument.Create();
      dom.load('C:\TCC2\livros.xml');
      endCsXML :=
dom.documentElement.selectNodes('//ROWDATA/ROW[@DADOS="'+edPesquisa.Text
+'"]');
      if endCsXML.length = 1   then
        begin
          for i:=0 to endCsXML.length-1 do
          begin
            paramCsXML := endCsXML.item[i].attributes[2].text;
            cdsXMLConsulta.LoadFromFile('C:\TCC2\livros.xml');
            cdsXMLConsulta.Active := true;
            dbgResultado.DataSource := dsXMLConsulta;
            dbmDados.DataSource := dsXMLConsulta;
            dbmDados.DataField := 'DADOS';
            showmessage('Dados obtidos do cache!');
            pesquisa := true;
          end;
        end;
    end;
   result := pesquisa;
end;
```

Quadro 19 – Rotina de busca dos dados no arquivo XML (livros.xml)

Na rotina apresentada no quadro 19, após efetuada uma verificação no arquivo para saber se o mesmo está armazenado na máquina a mais de cinco horas, isto é necessário pois como os dados de livros são alterados constantemente pelas editoras (principalmente preços), foi estipulado um tempo para que o arquivo permanecesse em *cache* para uma nova pesquisa.

Caso o arquivo esteja de acordo, o sistema verifica se a informação solicitada se encontra no arquivo XML armazenado na máquina. Para realizar esta função, foi utilizado um dos componentes do Delphi para manipulação de arquivos XML, denominado *Document Object Model* (DOM), que tem por finalidade realizar pesquisas procurando por parâmetros dentro dos nós do arquivo criado. Através deste componente o arquivo é aberto e a pesquisa é realizada pelos nós, para obter o parâmetro digitado no componente TEdit (edPesquisa). Se o resultado for encontrado, o sistema utiliza a estrutura de dados criada no componente TClientDataSet (cdsXMLConsulta) que contém os campos da tabela livros, esta estrutura é necessária para que seja possível efetuar a manipulação em memória dos registros gravados no arquivo XML.

O componente TClientDataSet (`cdsXMLConsulta`) abre o arquivo XML, o mesmo é ativado e o conteúdo obtido do arquivo é disponibilizado para o usuário para visualização através dos componentes TData-Aware (dbgResultado e dbmDados) incluídos na aplicação, esta transferência é efetua através do componente TDataSource (dsXMLConsulta), se os dados foram obtidos do arquivo aparece uma mensagem avisando o usuário que os dados foram obtidos do cache e a função retorna *true*.

O quadro 20 apresenta a rotina de consulta a dados ao banco do módulo de livrarias, utilizando o método *ParamByName* .

```
procedure TfrmConsulta.bbExecutaClick(Sender: TObject);
var
  pesquisa : boolean;
begin
   pesquisa := false;
   if pesquisaXML(pesquisa) <> true then
     begin
       if edPesquisa.Text <> '' then
         begin
           cdsConsulta.Active := false;
           cdsConsulta.Params.ParamByName('parametro').AsString :=
edPesquisa.Text;
           dbgResultado.DataSource := dsConsulta;
           dbmDados.DataSource := dsConsulta;
           dbmDados.DataField := 'DADOS';
           cdsConsulta.Active := true;
           cdsConsulta.SaveToFile('C:\TCC2\livros.xml',dfXMLUTF8);
           showmessage('Não foi localizado em cache! Dados obtidos do
servidor');
         end
       else
         begin
           showmessage('Nenhum parametro de pesquisa adicionado!');
           cdsConsulta.Active := false;
         end;
     end;
end;
```

Quadro 20 – Rotina de pesquisa no banco de dados utilizando ParamByName

No quadro 20 tem-se, a rotina de consulta a dados que tem por finalidade obter um parâmetro em um componente TEdit (edPesquisa) e repassá-lo ao componente TClientDataSet através do método ParamByName. O componente TClientDataSet se encarrega de passar o parâmetro informado para o servidor e incluí-lo na instrução SQL do componente TSqlDataSet através do componente TDataSetProvider. Nesta rotina, após a apresentação dos dados para o usuário através de componentes TData-Aware, é gravada a pesquisa em um arquivo XML (livros.xml), que fica armazenado localmente na estação cliente.

No quadro 21 é apresentada a estrutura do arquivo livros.xml.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
 <DATAPACKET Version="2.0">
 <METADATA>
   <FIELDS>
     <FIELD attrname="REGISTRO" fieldtype="string" WIDTH="25" />
     <FIELD attrname="DESCRICAO" fieldtype="string" WIDTH="70" />
     <FIELD attrname="DADOS" fieldtype="bin.hex" SUBTYPE="Text" WIDTH="1" />
     <FIELD attrname="TIPO" fieldtype="string" WIDTH="10" />
   </FIELDS>
    <PARAMS LCID="0" />
 </METADATA>
 <ROWDATA>
   <ROW REGISTRO="2" DESCRICAO="Assunto (Descrição)" DADOS="Linguagem de Programação" />
   <ROW REGISTRO="2" DESCRICAO="Autor (Nome pessoal)" DADOS="Cantú" />
   <ROW REGISTRO="2" DESCRICAO="Forma mais completa do nome contido no campo "Nome pessoal""
DADOS="Cantú, Marco" />
   <ROW REGISTRO="2" DESCRICAO="International Standard Book Number (ISBN)" DADOS="8534614083"
/>
   <ROW REGISTRO="2" DESCRICAO="Nome do publicador, do distribuidor, etc" DADOS="Makron Books
Ltda." />
   <ROW REGISTRO="2" DESCRICAO="Notas Geral" DADOS="Se você está procurando tirar o máximo
proveito dos poderosos recursos da versão mais recente do Delphi, você não pode ficar sem
este livro. Uma abordagem prática e com base em exercícios dirigidos o ajuda a desenvolver
habilidades importantes, a resolver problemas complicados e a construir e implementar
recursos sofisticados em seus aplicativos de banco de dados, cliente/servidor e para a
internet." />
   <ROW REGISTRO="2" DESCRICAO="Número ou código padrão" DADOS="9788534614085" />
   <ROW REGISTRO="2" DESCRICAO="Sumário, etc" DADOS="Introdução Parte I - Fundamentos
Capitulo 1 - O IDE do Delphi 6 Capitulo 2 - A Linguagem Object Pascal: Classes e Objetos
Capítulo 3 - A Linguagem Object Pascal: Herança e Polimorfismo Capítulo 4 - A Biblioteca de
tempo de execução ......" />
   <ROW REGISTRO="2" DESCRICAO="Série/Coleção (Título)" DADOS=""A Biblia"" />
   <ROW REGISTRO="2" DESCRICAO="Tamanho da unidade (Numero de volumes; Qtd. de páginas...)"
DADOS="935 páginas" />
   <ROW REGISTRO="2" DESCRICAO="Termos de disponibilidade (Normal; Fora Catálogo; Esgotado;
Temp Esg.)" DADOS="Fora de Catálogo" />
   <ROW REGISTRO="2" DESCRICAO="Termos de disponibilidade (Preço)" DADOS="R$ 190,00" />
   <ROW REGISTRO="2" DESCRICAO="Título" DADOS="Dominando o Delphi 6" />
  </ROWDATA>
 </DATAPACKET>
```

Quadro 21 – Arquivo livros.xml

O quadro 21 mostra o arquivo livros.xml gerado após o usuário efetuar uma consulta, onde armazena todos os dados referente aos livros pesquisados, neste arquivo aparece os campos da tabela de livros nas *tags* “FIELD” e os registros recebidos do banco nas *tags* “ROW”.

3.3.2 Operacionalidade da implementação

Nesta seção, são apresentadas as operacionalidades da implementação do aplicativo servidor e dos módulos para editoras e livrarias.

Na figura 10, tem-se uma visão geral da utilização do sistema.

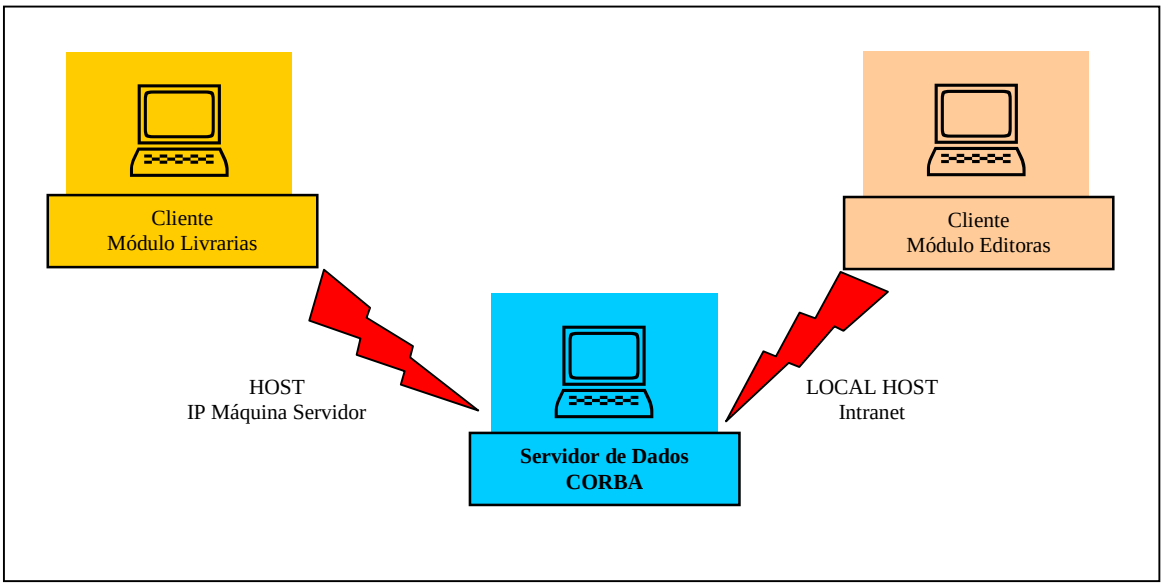


Figura 10 – Visão geral da utilização do sistema

Na figura 10, tem-se a máquina cliente do módulo para livrarias acessando o servidor de dados CORBA através do *host* com o endereço IP determinado para a conexão. Também tem-se o cliente do módulo para editoras acessando o servidor de dados CORBA através de uma Intranet, utilizando *localhost.*

3.3.2.1 Servidor

Para execução do servidor o VisiBroker Smart Agent precisa estar ativo, após a ativação do Smart Agent o servidor de dados deve ser executado, e o mesmo permanecerá minimizado, a figura 11 mostra a tela do servidor em execução.

Figura 11 – Tela servidor

3.3.2.2 Módulo para editoras

Na execução do sistema o usuário visualiza a tela apresentada na figura 12. Esta tela é visualizada após a aplicação efetuar a conexão com o servidor que está em execução na rede.

Figura 12 – Tela de acesso módulo editoras

Após a inserção dos dados, se for pressionado o botão “Cancelar”, o sistema fecha a aplicação e desconecta-se do servidor. Pressionando o botão “OK”, o sistema tenta efetuar a validação dos dados informados, se os dados não estiverem corretos, ou seja, se o usuário não estiver cadastrado ou foi digitado errado aparece a mensagem: “Acesso negado, usuário desconhecido!”, se a senha estiver incorreta aparece a mensagem: “Acesso negado, senha inválida!”, se os dados forem validados corretamente o sistema fecha a tela de acesso e permite acesso a tela posterior, mostrada na figura 13.

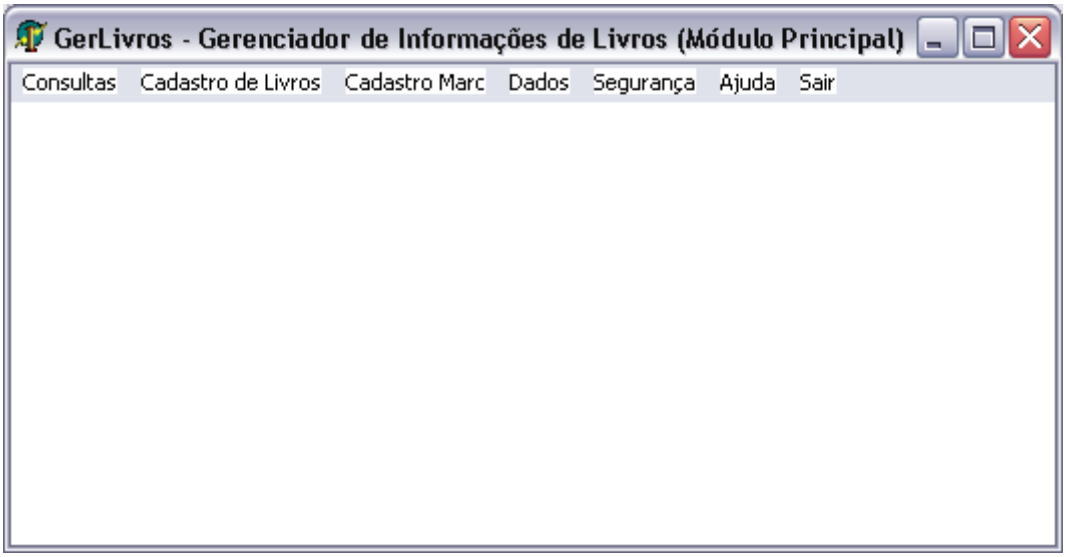


Figura 13 – Tela principal

Ao acessar o menu principal da aplicação o usuário visualiza os menus de acesso aos itens de manutenção do sistema e atualização dos dados. Clicando em “Consultas”, aparece a tela de consulta de livros, que é a mesma utilizada pelas livrarias e ilustrada na figura 22 do item 3.3.2.3. Clicando em “Cadastro de Livros”, aparecem os itens necessários para o cadastramento de um livro em formato padrão de catalogação. Estes itens podem ser observados na figura 14.

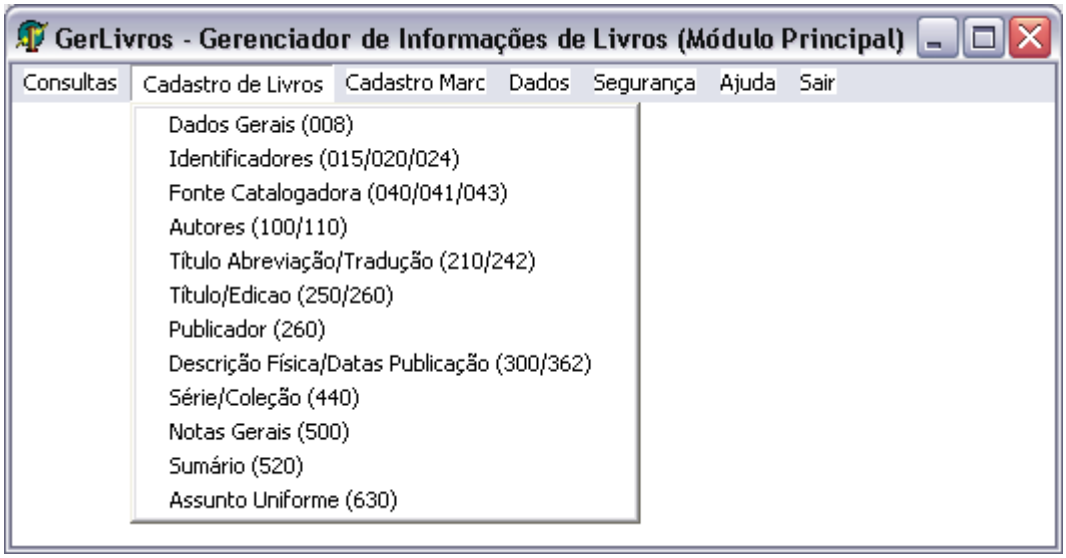


Figura 14 – Itens do menu “Cadastro de Livros”

Clicando nos itens abrirão as telas de cadastro para adicionar os dados referentes ao livro que está sendo cadastrado. As telas de cadastro são padronizadas, isto é, o modo de

utilizar é o mesmo para todas.

A figura 15 apresenta a tela de cadastro para armazenamento dos dados gerais referente a um livro que está sendo cadastrado.

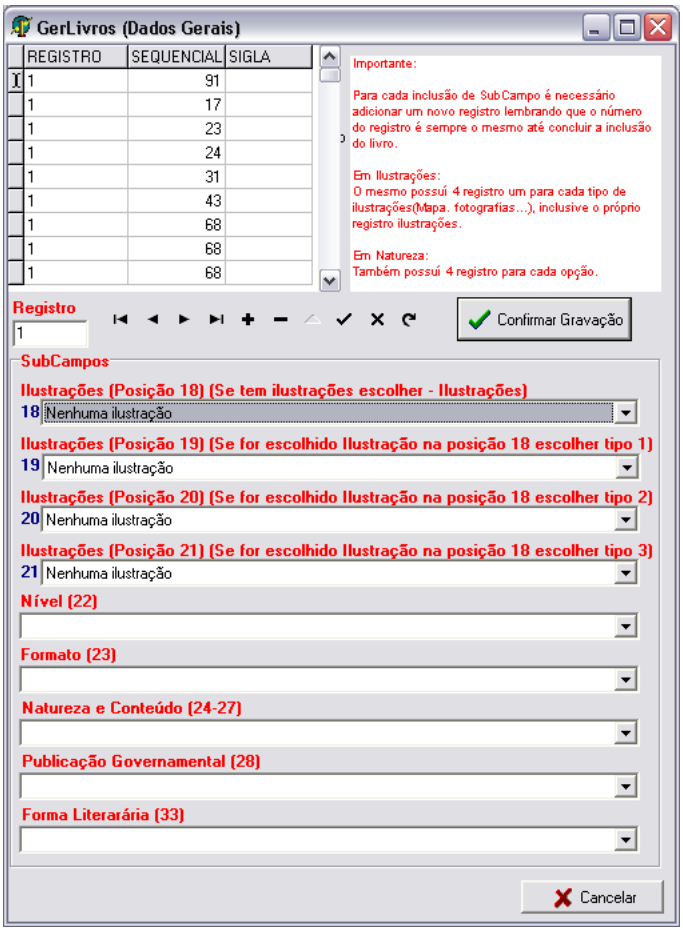


Figura 15 – Tela de cadastro de dados gerais

Na figura 15 pode ser vista a tela de cadastro de dados gerais. Nesta tela tem-se os dados já cadastrados referentes aos livros apresentados em um *grid* no canto superior esquerdo da aplicação. O usuário pode mover-se pelos registros através dos botões de navegação logo abaixo da tela. As funções de cada botão do navegador podem ser vistas na figura 15, o campo registro mostra o registro que está sendo selecionado no *grid* e também serve para digitar o valor do novo registro referente a um novo livro que será cadastrado, abaixo aparecem às opções de “sub-campos”, ou seja, os itens para cadastramento dos livros. Pode-se ressaltar que para cada “sub-campo” é feita uma inserção e o número do registro permanece inalterado até a confirmação da gravação do mesmo. Nesta tela não temos o

campo “Dados do livro referente ao subcampo”, pois as informações contidas nos componentes de rolagem já são pré-estabelecidos pelo registro MARC.

Na figura 16 é ilustrada a tela de cadastro de indicadores padrões (ISBN, EAN...).

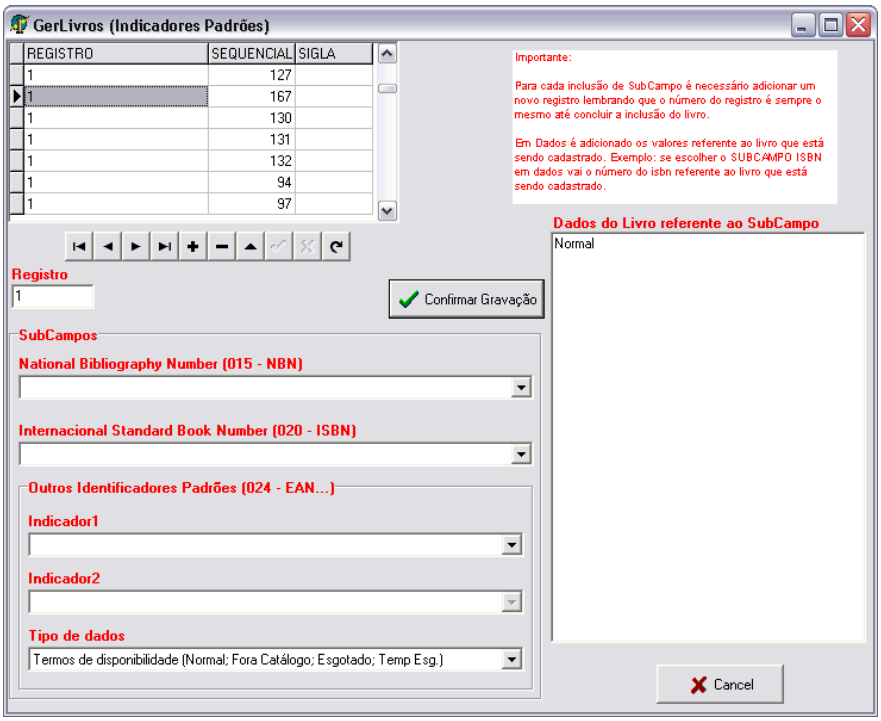
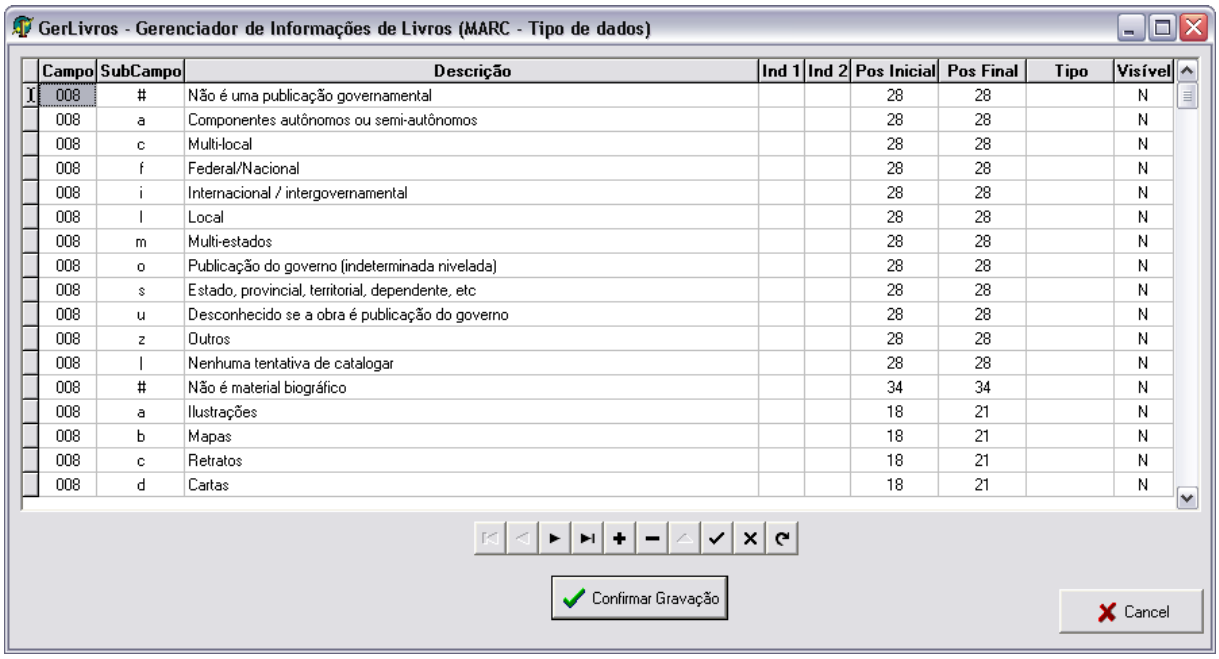


Figura 16 – Tela de cadastro de indicadores padrões

A figura 16 mostra a tela de cadastramento de indicadores, esta tela apresenta o campo “Dados do Livro referente ao sub-campo”, onde, o usuário digita os dados referentes às informações pertinentes ao livro. Nesta tela, após ser escolhido o “sub-campo” denominado “tipo de dados”, o usuário necessita digitar os dados referentes às informações solicitadas no mesmo e pertinentes ao livro que está sendo cadastrado. Os demais itens de cadastro do menu “Cadastro de Livros”, seguem o mesmo padrão de cadastramento sendo semelhante à tela apresentada na figura 16. O usuário precisa ter um certo conhecimento sobre o registro MARC para poder fazer as alterações e inclusões das obras, pelo menos em relação aos campos do registro.

Este protótipo possibilita a atualização do registro MARC através da opção do menu “Cadastro MARC”, tela mostrada na figura 17.



Campo	SubCampo	Descrição	Ind 1	Ind 2	Pos Inicial	Pos Final	Tipo	Visível
008	#	Não é uma publicação governamental			28	28		N
008	a	Componentes autônomos ou semi-autônomos			28	28		N
008	c	Multi-local			28	28		N
008	f	Federal/Nacional			28	28		N
008	i	Internacional / intergovernamental			28	28		N
008	l	Local			28	28		N
008	m	Multi-estados			28	28		N
008	o	Publicação do governo (indeterminada nivelada)			28	28		N
008	s	Estado, provincial, territorial, dependente, etc			28	28		N
008	u	Desconhecido se a obra é publicação do governo			28	28		N
008	z	Outros			28	28		N
008		Nenhuma tentativa de catalogar			28	28		N
008	#	Não é material biográfico			34	34		N
008	a	Ilustrações			18	21		N
008	b	Mapas			18	21		N
008	c	Retratos			18	21		N
008	d	Cartas			18	21		N

Figura 17 - Tela de atualização do registro MARC

A figura 17 apresenta a tela onde o usuário pode fazer as alterações no registro MARC. Esta tela possui um *grid* com todos os dados do registro cadastrados e conseqüentemente neste *grid* a opção de edição através do navegador, onde o usuário pode fazer as alterações necessárias para atualizar os dados do mesmo.

A figura 18, mostra a tela de manipulação dos dados referente às siglas.

Sigla	Descrição	Tipo
a	Asia	GAC
a-af	Afghanistan	GAC
a-ai	Armenia (Republic)	GAC
s-bl	Brazil	GAC
aar	Afar	LC
abk	Abkhaz	LC
ace	Achinese	LC
por	Portuguese	LC
act	Actor	RC
adp	Adapter	RC

Figura 18 – Tela de atualização de siglas

A figura 18 apresenta o cadastro de siglas que também foi retirado dos padrões de catalogação MARC. Esta tela também possibilita a atualização e inclusão dos dados através do *grid* e dos botões de "navegação".

Todos os dados contidos nesta tela e na tela de "Cadastro MARC" são utilizados para o cadastramento dos livros nas telas do menu "Cadastro de Livros".

Na figura 19 tem-se a tela que apresenta os usuários cadastrados para acesso ao servidor e onde o usuário poderá efetuar a manutenção dos mesmos.

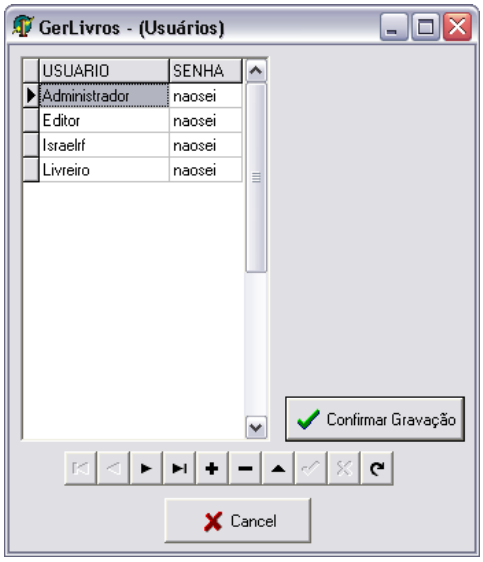


Figura 19 – Tela de cadastro de usuários

O usuário do módulo de editoras poderá na tela apresentada na figura 19 cadastrar e fazer a manutenção nos dados para o acesso dos sistemas tanto do módulo de editoras, como do módulo de livrarias.

3.3.2.3 Módulo para livrarias

Quando o sistema é executado, o usuário visualizará a tela da figura 20, onde o mesmo necessita escolher a editora para o qual o sistema efetuará a conexão.

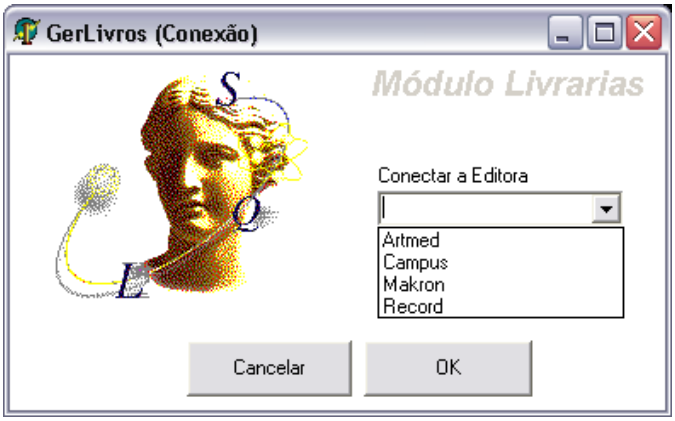


Figura 20 – Tela de escolha da editora para conexão

Tendo escolhido a editora na tela mostrada na figura 20, o sistema efetua a conexão através de um *host*, determinado em um arquivo texto (conexoes.txt) que fica armazenado no diretório da aplicação e contém o nome da editora e o *Internet Protocol* (IP), que é utilizado para a conexão. Após o servidor ter sido localizado e a conexão ter sido estabelecida o sistema apresenta uma tela de acesso igual à mostrada na figura 12.

Na figura 21 é apresentada a tela principal do sistema onde o usuário poderá fazer suas consultas utilizando o item “Consulta de livros” do menu.

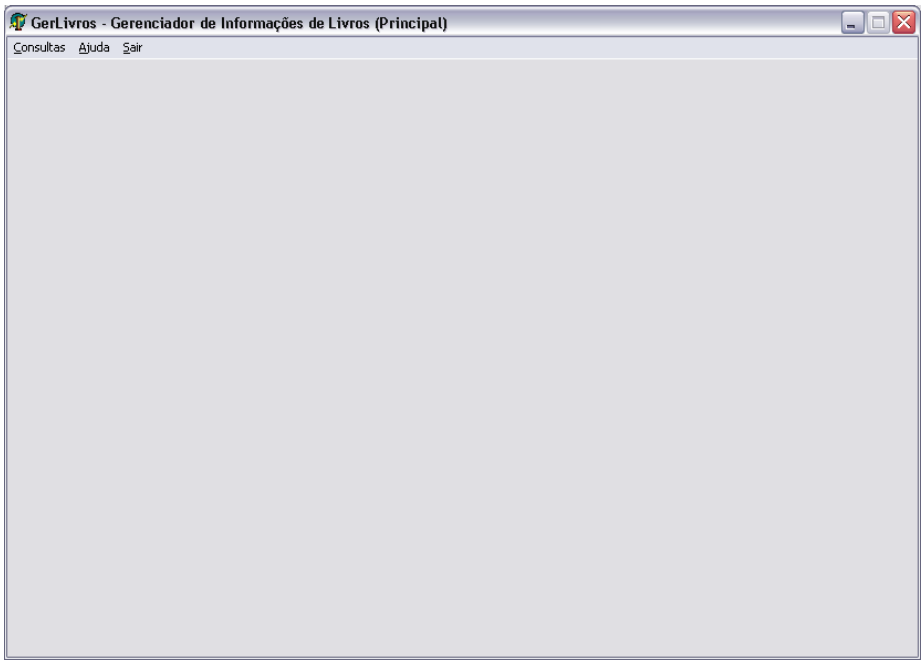


Figura 21 – Tela principal

Tendo acessado a tela principal, o usuário pode clicar no item “consultas” do menu, que apresenta a tela da figura 22, onde o mesmo pode efetuar a consulta dos dados utilizando

principalmente os seguintes parâmetros de pesquisa (Título, Autor, Editora, Coleção/Série, ISBN e EAN), a busca é efetuada digitando o parâmetro desejado no campo: “Parâmetros para consulta” e clicando no botão “Executa Pesquisa”, onde o resultado da pesquisa será apresentado no *grid* na parte superior da tela.

Para que o usuário tenha acesso aos dados ele poderá navegar dentro do *grid* e escolher qual a informação que ele deseja obter do registro encontrado, ou seja, se o mesmo desejar saber o título ou o preço, basta escolher a informação no *grid*, que o dado referente ao que ele deseja aparecerá no campo: “Dados”, ao lado do botão: “Executa Pesquisa”. Para sair da tela de consulta basta clicar no botão “Cancelar”.

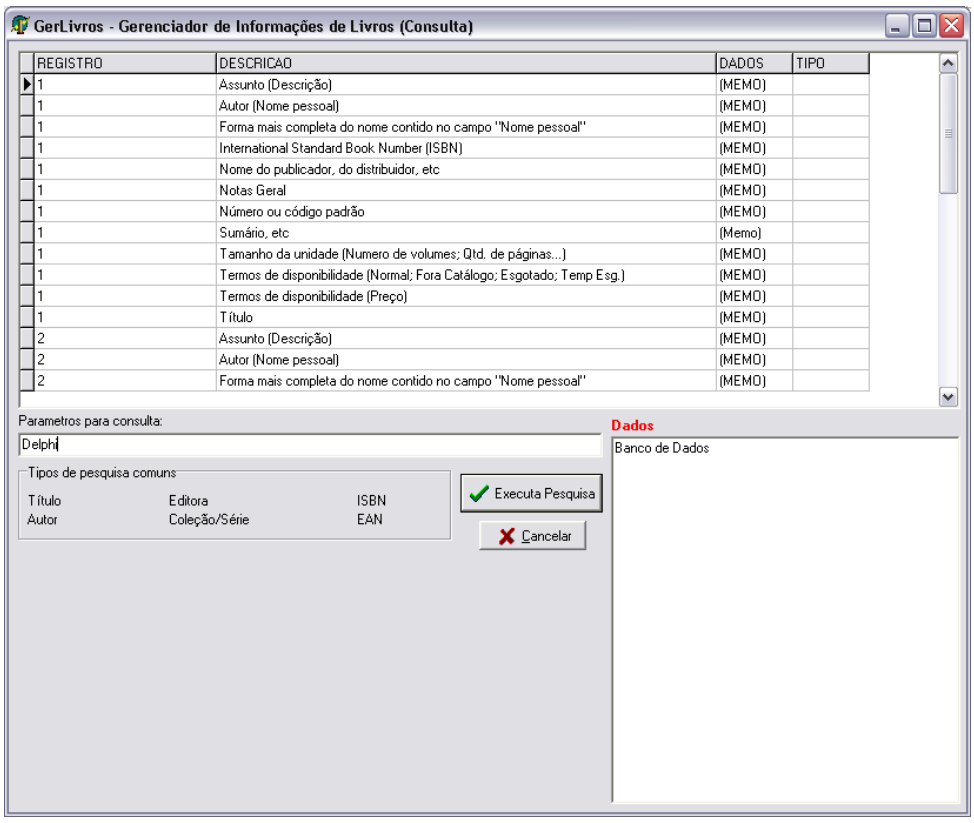


Figura 22 – Tela de consulta

Na tela da figura 22, caso o usuário não digite nenhum parâmetro para a consulta, aparece a mensagem “Nenhum parâmetro de pesquisa adicionado!”, também aparece mensagens referente a pesquisas realizadas diretamente no *cache* da máquina, isto se a informação pesquisada for encontrada no arquivo XML que está gravado na estação local.

3.4 RESULTADOS E DISCUSSÃO

Este trabalho atingiu seu objetivo de realizar o cadastro e consulta de informações de livros em um sistema distribuído utilizando um padrão de metadados. Foi implementado de forma a permitir que os usuários de uma editora possam cadastrar os dados referentes aos seus livros em uma aplicação cliente distribuída, da mesma forma como os usuários de uma livraria poderão consultar essas informações, também em um ambiente distribuído.

Foram realizados diversos testes com o protótipo. Entretanto, a maioria dos testes foram feitos utilizando uma rede ponto-a-ponto, utilizando uma conexão com cabo *crossover*, poucos testes foram realizados em uma rede com uma estrutura mais complexa, ou seja, utilizando *switchs* e *hubs*, porém os testes realizados tanto em uma, quanto na outra, surtiram os efeitos esperados, ou seja, a velocidade de obtenção dos dados foi satisfatória e a aplicação cliente detectou o ORB na rede para efetuar a conexão com o servidor utilizando os IPs das máquinas que estavam rodando o aplicativo servidor.

A utilização do padrão de catalogação MARC 21, foi de grande contribuição para uma melhor determinação das informações referente a livros que seriam utilizados no desenvolvimento do protótipo, porém como o padrão MARC 21 é um padrão para utilização em bibliotecas e possui uma enorme quantidade de campos e subcampos, foi feita uma análise a respeito de todos os campos contidos no padrão e retirado aqueles que mais se aproximavam do propósito da aplicação, estes dados foram de extrema importância para o desenvolvimento do banco de dados. Devido à complexidade do padrão escolhido, o trabalho de desenvolver o banco de dados teve que ser muito bem estudado. Sendo assim, foi realizada uma pesquisa na biblioteca da FURB, para melhor entender o padrão, pois o mesmo é utilizado para catalogação dos dados nesta instituição. Chegou-se a constatação de que as tabelas deveriam ser geradas com os nomes dos campos do padrão, ou seja, cada campo (008, 010, 024, etc),

seria uma tabela com seus subcampos (a, b, c, d, etc) os respectivos atributos. Com isso foi feita a modelagem dos dados da forma como foi constatado. Mas na modelagem observou-se que para fazer o relacionamento das tabelas seria muito complexo e dificultaria na geração do banco de dados, que implicaria em problemas em uma possível atualização do padrão. Assim, foi determinado que o banco de dados deveria ser modelado de forma que fosse possível para o usuário do protótipo, fazer as alterações ou inclusões no padrão que foi armazenado na tabela “Marc_Tipo”. Assim surgiu a idéia de cadastrar uma tabela com todos os campos e outros dados, os mesmos que foram escolhidos para essa aplicação e retirados do padrão MARC, assim, ficaria mais fácil uma possível atualização dos mesmos.

No decorrer do estudo referente ao padrão utilizado para catalogação foi possível observar o quanto é complexo desenvolver uma aplicação ou até mesmo trabalhar com informações referente a livros, pois estas informações, à medida que vão sendo padronizadas, geram uma série de controvérsias, pois, cada livro possui uma característica diferente e utilizando-se o padrão ficou mais claro o quanto são variados os tipos de informações e como estas informações divergem de um livro para outro. Isto acaba gerando um problema principalmente na hora de disponibilizar estas informações, de maneira prática e de fácil compreensão ao usuário de uma aplicação que efetuará a manutenção nas mesmas.

Para o desenvolvimento deste protótipo, foi pesquisado o trabalho de Ferrari (2000), onde o mesmo descreve um protótipo de sistema de consulta de preços de supermercados, utilizando objetos distribuídos via Internet e o trabalho de Crispim e Fernandes, que se baseou no Prontuário Eletrônico de Pacientes (PEP), onde foi desenvolvido um sistema de prontuário eletrônico integrado do paciente para as clínicas do centro de saúde da Universidade do Vale do Itajaí (PEP/UNIVALI). Estes trabalhos, porém, tinham objetivos diferentes, de forma que os mesmos foram utilizados mais como inspiração base para o desenvolvimento desta aplicação.

4 CONCLUSÕES

Através deste trabalho foi possível explorar o funcionamento de uma aplicação distribuída. O desenvolvimento do trabalho contribuiu muito para o entendimento das tecnologias que estão por traz do funcionamento e da arquitetura dos sistemas distribuídos, que hoje são muito utilizados e que vêm crescendo a cada dia.

A facilidade de implementação e o funcionamento da aplicação utilizando as tecnologias apresentadas neste trabalho são um ponto a destacar, pois todos os recursos utilizados formam um conjunto de tecnologias que além de práticas, refletem em um resultado final bastante satisfatório. A utilização do Visibroker como ORB para esta aplicação também pode ser destacada, pois o mesmo serviu perfeitamente na função de *midlleare*, onde, independente do hardware que está sendo utilizado nas máquinas e a arquitetura envolvendo a rede que distribuí o protótipo, este se propôs a localizar de maneira rápida e sem necessidade de configurações extras, os dados disponibilizados pelo servidor e distribuídos para os módulos da aplicação.

A utilização de um padrão de metadados para o armazenamento das informações sobre os livros, também foi de extrema importância, pois sua utilização possibilitou chegar a uma estrutura de banco de dados que permite disponibilizar os mesmos de uma forma completa e padronizada, ou seja, através dos campos disponibilizados pelo padrão MARC, é possível saber exatamente quais informações precisam ser armazenadas. Com isso pretende-se levar essa tecnologia a ser implantada em todas as editoras, pois, traria benefícios não somente às livrarias, mas também às bibliotecas de todo o país, pois os dados poderiam ser exportados e importados entre os bancos em um padrão conhecido por todos.

Este trabalho foi desenvolvido com o propósito de tornar o processo de obtenção dos dados das livrarias o mais fácil e completo possível, o objetivo foi alcançado e ainda com a

introdução do padrão de metadados, pode-se ampliar a sua utilização, não ficando restrito somente as livrarias, mas também estendendo sua utilização a todas as instituições que de alguma forma necessitam obter informações referentes a livros e desejam estar em constante atualização sobre as mesmas. Este trabalho serve como um exemplo inicial, como uma base, sobre a qual outros projetos possam ser desenvolvidos.

4.1 EXTENSÕES

Como extensão para esse trabalho sugere-se:

a) a utilização de outras tecnologias como Java para fazer módulos que rodem em ambiente web;
b) a implantação de um projeto para integração de livrarias, editoras e bibliotecas utilizando o padrão mencionado, mas com a troca de informações sendo feitas entre as mesmas através da Internet;
c) utilizar a tecnologia Net Remoting para distribuição, em aplicações .NET, pois sendo uma tecnologia nova de distribuição de dados, precisaria ser explorada mais a fundo. Seu padrão de distribuição utiliza tecnologias inovadoras, principalmente na questão de segurança, o que é de extrema importância para as novas aplicações distribuídas que poderiam ser desenvolvidas;
d) trabalhar a utilização do padrão MARC junto às editoras e, conseqüentemente, desenvolver um protótipo que efetue importação e exportação de dados no formato XML para atualização das bases de dados, tanto de livrarias como também de bibliotecas, pois assim a disponibilização destes seriam mais dinâmicas e confiáveis.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, Mauricio Barcellos; CENDON, Beatriz Valadares. Pesquisa sobre ferramentas de conversão de registros catalográficos padrão MARC para a linguagem XML. In: ENCONTRO NACIONAL DE PESQUISA EM CIÊNCIA DA INFORMAÇÃO, 5., 2003, Belo Horizonte. **Anais...** Belo Horizonte: Escola de Ciência da Informação da UFMG, 2003. Disponível em: <http://www.eci.ufmg.br/mba/text/artigo_marcxml_sub_web.pdf>. Acesso em: 06 jun. 2005.

CAPELETTO, Johni Jéferson. **Comunicação entre objetos distribuídos utilizando a tecnologia CORBA**. 1999. 60 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

CERQUEIRA, Andréa. O modelo de arquitetura CORBA. **Linha de Código**. São Paulo, n.299, 2004. Disponível em: <http://www.linhadecodigo.com.br/artigos.asp?id_ac=299>. Acesso em: 15 maio 2006.

COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. **Distributed systems:** concepts and design. 3rd ed. Harlow: Pearson, 2001.

CRISPIM, Carlos Fernando; FERNANDE, Anita Maria da Rocha. Prontuário eletrônico integrado do paciente para as clínicas do centro de saúde da UNIVALI (PEP/UNIVALI). In: CONGRESSO REGIONAL DE INICIAÇÃO CIENTIFICA, 18., 2003, Itajaí. **Anais...** Itajaí: UNIVALI, 2003. Disponível em: <http://www.hu.ufsc.br/ix_cibs/trabalhos/arquivos/58.pdf>. Acesso em: 14 maio 2005.

FERRARI, Anderson Luiz. **Protótipo de sistema de consulta de preços de supermercados utilizando objetos distribuídos via Internet**. 2000. 58 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

GRAHAM, Ian. **Migrating to object technology**. California: Addison-Wesley, 1994.

LIGHT, Richard. **Iniciando em XML**. São Paulo: Makron Books, 1999.

MARC STANDARDS. **MARC 21 concise format for bibliographic data**. Washington, 2004. Disponível em: <http://www.loc.gov/mac/bibliographic/ecbdhome.html>. Acesso em: 18 maio 2005.

MARTIN, James. **Princípios de análise e projeto baseados em objetos**. Rio de Janeiro: Campus, 1994.

MINDLIN, José. Mindlin e a paixão pelos livros. **O Estado de São Paulo**, São Paulo, [2004?]. Disponível em: <http://www.vivaleitura.com.br/artigos.asp>. Acesso em: 01 maio 2005.

OBJECT MANAGEMENT GROUP. Preface. In: OBJECT MANAGEMENT GROUP. **The Common object request broker**: architecture and specification. Revision 2.0. St. Louis, 1998. Disponível em: <http://www.cs.wustl.edu/~cleeland/tao/corba_20_spec.pdf>. Acesso em: 14 set. 2005.

POLONIATO, Dário. Arquitetura de sistemas aplicativos multicamadas. In: SEMINÁRIOS DE PESQUISA, 7., 1999, São Paulo. **Anais...** São Paulo: USP, 1999. Disponível em: <http://www.fea.usp.br/ead457/Semin%C3%A1rios2S99.htm>. Acesso em: 17 maio 2005.

RICCIONI, Paulo Roberto. **Introdução a objetos distribuídos com CORBA**. Florianópolis: Visual Books, 2000.

RODRIGUES, Celso. **Objetos distribuídos.** São Paulo, [2003]. Disponível em: <http://www.activedelphi.com.br>. Acesso em: 10 abr. 2005.

SILVA, Osmar J. **XML**: aplicações práticas. São Paulo: Érica, 2001.

SOUZA, Anamélia Contente de; MAZZOLA, Vitório Bruno. Implementando aplicações distribuídas utilizando CORBA. **Infocomp Journal of Computer Science**, Lavras, v. 1, n. 1, 2000. Disponível em: <http://www.dcc.ufla.br/infocomp/artigos/v1.1/artigoCORBA.pdf>. Acesso em: 10 abr. 2005.

TANENBAUM, Andrew S.; STEEN, Maarten Van. **Distributed systems**: principles and paradigms. Seatle: Prentice Hall, 2001.

TITTEL, Ed. **Teoria e problemas de XML**. Porto Alegre: Bookman, 2003.

ANEXO A – Campos do registro MARC utilizados no protótipo

No quadro 22 são apresentados os campos que foram escolhidos do registro MARC, para serem utilizados na catalogação dos dados deste protótipo.

008--LIVROS

18-21 - Ilustrações
- # - Nenhuma ilustração
- a - Ilustrações
- b - Mapas
- c - Retratos
- d - Cartas
- e - Plantas
- f - Placas
- j – Tabelas Genealógicas
- k - Formulários
- l - Amostras
- o - Fotografias
- | - Nenhuma tentativa de codificar

22 – Nível da obra
- # - Desconhecido ou não especificado
- a – Pré-escola
- b - Primário
- c – Pré-adolescente
- d - Adolescente
- e - Adulto
- f - Especializado
- g - Geral
- j - Juvenil
- | - Nenhuma tentativa de codificar

23 – Formato da obra
- # - nenhuma das opções
- a - Microfilme
- b - Microficha
- c - Microopaque
- d – Reprodução grande de impressão
- f - Braille
- r – Reprodução de impressão regular
- s - Eletrônico
- | - Nenhuma tentativa de codificar

24-27 – Natureza e conteúdo da obra
- # - Nenhuma natureza específica dos conteúdos
- a – Abstracts/Sumários
- b - Bibliografias
- c - Catálogos
- d - Dicionários
- e - Enciclopédias
- f - Manuais
- g – Artigos Legais
- i - Índices
- j – Documentos de Patente Originais
- k - Discografias
- l – Legislação
- m - Teses
- n – Exame de literatura em uma área sujeita
- o - Revisões
- p – Textos programados
- q - Filmografias
- r - Diretórios
- s – Estatísticos
- t – Relatórios Técnicos
- u – Padrões/Especificações
- v – Casos legais e Notas de casos
- w – Relatórios e sumários da lei
- z - Tratados
- | - Nenhuma tentativa de codificar

28 – Publicação governamental
- Não é uma publicação governamental
a – Componentes autônomos ou semi-autônomos
c – Multi-local
f - Federal/nacional
i – Internacional intergovernamental
l - Local
m – Multi-estados
o – Publicação do governo (indeterminada nivelada)
s – Estado, provincial, territorial, dependente, etc.
u – Desconhecido se a obra for publicação do governo
z - Outros
| - Nenhuma tentativa de catalogar
33 – Forma Literária
0 – Não ficção
1 - Ficção
c – Quadrinhos
d - Dramas
e - Ensaios
f - Novelas
h - Humor, sátiras, etc.
i - Letras
j – Histórias curtas
m – Formulários misturados
p - Poesias
s - Discursos
u - Desconhecido
| - Nenhuma tentativa de catalogar
34 - Biografia
- Não é material biográfico
a - Autobiografia
b – Biografia individual
c – Biografia coletiva
d – Contem informação biográfica
| - Nenhuma tentativa de catalogar

015 - NATIONAL BIBLIOGRAPHY NUMBER (NBN) (R)
$a - National bibliography number (R)
$2 - Fonte (NR)

020 - INTERNATIONAL STANDARD BOOK NUMBER (ISBN) (R)
$a - International Standard Book Number (NR)
$c – Termos de disponibilidade (preço)(NR)
$z – Cancelado/Inválido(R)

024 – OUTRO IDENTICADOR PADRÃO (EAN) (R)
Indicador
0 - International Standard Recording Code (ISRC)
1 - Universal Product Code (UPC)
2 - International Standard Music Number (ISMN)
3 - International Article Number (EAN)
4 - Serial Item and Contribution Identifier (SICI)
7 – Fonte especificada no campo $2
8 – tipo não especifico de Identificador padrão
- No information provided
Campos
$a – Número ou código padrão(NR)
$c – Termos de disponibilidade (NR)
$d – Códigos adicionais depois do número ou do código padrão(NR)
$z – Número ou código padrão cancelado/inválido(R)
$2 – Fonte do número ou do código(NR)

040 – FONTE CATALOGADORA (NR)
$a – Agência catalogadora original(NR)
$b – Linguagem de catalogação (NR)

041 – CÓDIGO DA LINGUAGEM (R)
Indicador
0 – a obra não é uma tradução/não incluí uma tradução
1 – a obra é ou incluí uma tradução
- código de linguagem MARC
7 – Fonte especificada no campo $2
Campos
$a – Código da linguagem do texto/trilha ou título separado (R)
$b – Código da linguagem para sumário, abstract ou subtítulos (R)
$d – Código da linguagem do texto cantado ou falado (R)
$e – Código da linguagem dos livretos (R)
$f – Código da linguagem do índice (R)
$g – Código da linguagem do material complementar com exceção dos livretos (R)
$h – Código da linguagem de traduções originais e/ou intermediárias do texto (R)
$2 – Fonte do código (NR)

043 – CÓDIGO DA ÁREA GEOGRÁFICA (NR)
$a – Código da área geográfica (R)
$b – Código do Local CAG (R)
$c – Código ISO (R)

100 – ENTRADA PRINCIPAL NOME PESSOAL – AUTOR (NR)
Indicador
0 – Parte Nome
1 - Sobrenome
3 – Nome de Família
Campos
$a – Nome pessoal (NR)
$b - Numero(NR)
$c – Os títulos e outros que exprimem associados com um conhecido (R)
$d – Datas associadas com um conhecido (NR)
$e – Termo de relator (R)
$f – Data de um trabalho (NR)
$g – Informações variadas (NR)
$j – Qualificador de atribuição (R)
$k – Dar forma ao subtítulo (R)
$l – Linguagem de um trabalho (NR)
$n – Número da parte/Seção de um trabalho (R)
$p – Nome da parte/Seção de um trabalho (R)
$q – Formulário mais completo do nome contido no campo $a (NR)
$t – Título de um trabalho (NR)
$u - Afiliação (NR)
$4 – Código de relator (R)

110 – ENTRADA PRINCIPAL NOME CORPORATIVO - AUTOR (NR)
Indicador
0 – Nome invertido
1 – Nome de jurisdição
2 – Nome de ordem direta
Campos
$a – Nome incorporado ou nome de jurisdição como o elemento de entrada (NR)
$b – Unidade Subordinada (R)
$e – Termo de relator (R)
$f – Data de um trabalho (NR)
$g – Informações variadas (NR)
$k – Dar forma ao subtítulo (R)
$l – Linguagem de um trabalho (NR)
$n - Número da parte/Seção de um trabalho (R)
$p - Nome da parte/Seção de um trabalho (R)
$t - Título de um trabalho (NR)
$u - Afiliação (NR)
$4 - Código de relator (R)

210 – TÍTULO ABREVIADO (R)
Indicador
0 – Nenhuma entrada adicionada
1 – Entrada adicionada
- Título chave abreviado
0 – Outro título abreviado
Campos
$a – Título abreviado (NR)
$2 - Fonte (R)

242 – TRADUÇÃO DO TÍTULO CATALOGADO NA AGËNCIA (R)
Indicador
0 – Nenhuma entrada adicionada
1 – Entrada adicionada
Campos
$a - Título (NR)
$b – Restante do título (NR)
$c – Indicação da responsabilidade (NR)
$h - Meio (NR)
$n - Número da parte/Seção de um trabalho (R)
$p - Nome da parte/Seção de um trabalho (R)
$y – Código da linguagem do título traduzido (NR)**245 – INDICAÇÃO DO TITULO (NR)**
Indicador
0 – Nenhuma entrada adicionada
1 – Entrada adicionada
Campos
$a – Título (NR)
$b - Restante do título (NR)
$c - Indicação da responsabilidade, etc. (NR)
$f – Datas inclusivas (NR)
$g – Datas maiores (NR)
$h - Meio (NR)
$n - Número da parte/Seção de um trabalho (R)
$p - Nome da parte/Seção de um trabalho (R)
$s - Versão (NR)

250 – INDICAÇÃO DA EDIÇÃO (NR)
$a – Indicação da edição (NR)
$b – Restante da indicação da edição (NR)
260 – PUBLICAÇÃO, DISTRIBUIÇÃO, ETC (R)
Indicador
- Não ha informação de aplicação / não forneceu / pub. disponível mais cedo
2 – Publicação de intervenção
3 – Corrente / publicação disponível mais tarde
Campos
$a – Lugar da publicação, da distribuição, etc. (R)
$b – Nome do publicador, do distribuidor, etc. (R)
$c – Data de publicação, distribuição, etc. (R)
$e – Lugar de fabricação (R)
$f - Fabricante (R)
$g – Data de fabricação (R)
300 – DESCRIÇÃO FÍSICA (R)
$a - Extensão (R)
$b – Outros detalhes físicos (NR)
$c - Dimensões (R)
$e – Acompanha material (NR)
$f – Tipo de unidade (R)
$g – Tamanho da unidade (R)
362 – DATAS DE PUBLICAÇÃO E/OU DA DESIGNAÇÃO SEQÜENCIAL (R)
Indicador
0 – Estilo formatado
1 – Nota não formatada
Campos
$a – Datas de publicação e/ou da designação seqüencial (NR)
$z – Fonte de informação (NR)
440 – ENTRADA DE SÉRIE (R)
$a - Título (NR)
$n – Número da parte / Seção de um trabalho (R)
$p – Nome da parte / Seção de uma série (R)
$v – Numero de Volumes / designação seqüencial (NR)
500 – NOTAS GERAL (R)
$a – Nota geral (NR)
$3 – Materiais especificados (NR)

520 - SUMÁRIO, ETC. (R)
Indicador
- Sumário
0 - Assunto
1 - Revisão
2 – Escopo e conteúdo
3 - Abstract
8 – Não gerou nenhuma constante de exposição
Campos
$a - Sumário, etc. notas (NR)
$b – Expansão da nota do sumário (NR)
$u – Identificador uniforme do recurso (R)
$3 – Materiais especificados (NR)

630 – ENTRADA ASSUNTO – TÍTULO UNIFORME (R)
$a – Título (NR)
$n – Número da parte / Seção de um trabalho (R)
$p – Nome da parte / Seção de um trabalho (R)
$e – Termo do relator (R)
$f – Data do Trabalho (NR)
$g – Informações variadas (NR)
$v – Forma da subdivisão (R)
$x – Subdivisão geral (R)
$2 – Fonte do título ou do termo (NR)
$3 – Materiais especificados (NR)
$4 – Código do relator (R)

Quadro 22 - Campos do registro MARC utilizados no protótipo

ANEXO B – Campos iniciais de um registro MARC 21

No quadro 23 são apresentados os campos iniciais do registro MARC 21, o padrão completo poderá ser encontrado em MARC STANDARDS.

```
v--Leader and Directory--

  LEADER
     Character Positions
     00-04 - Logical record length
     05 - Record status
          a - Increase in encoding level
          c - Corrected or revised
          d - Deleted
          n - New
          p - Increase in encoding level from prepublication
     06 - Type of record
          a - Language material
          b - Archival and manuscripts control [OBSOLETE]
          c - Notated music
          d - Manuscript notated music
          e - Cartographic material
          f - Manuscript cartographic material
          g - Projected medium
          h - Microform publications [OBSOLETE]
          i - Nonmusical sound recording
          j - Musical sound recording
          k - Two-dimensional nonprojectable graphic
          m - Computer file
          n - Special instructional material [OBSOLETE]
          o - Kit
          p - Mixed material
          r - Three-dimensional artifact or naturally occurring object
          t - Manuscript language material
     07 - Bibliographic level
          a - Monographic component part
          b - Serial component part
          c - Collection
          d - Subunit
          i - Integrating resource
          m - Monograph/item
          s - Serial
     08 - Type of control
          # - No specific type
          a - Archival
     09 - Character coding scheme
          # - MARC-8
          a - UCS/Unicode
     10 - Indicator count
     11 - Subfield code count
     12-16 - Base address of data
     17 - Encoding level
          # - Full level
          1 - Full level, material not examined
          2 - Less-than-full level, material not examined
          3 - Abbreviated level
          4 - Core level
          5 - Partial (preliminary) level
          7 - Minimal level
          8 - Prepublication level
          u - Unknown
          z - Not applicable
     18 - Descriptive cataloging form
          # - Non-ISBD
          a - AACR 2
          i - ISBD
          p - Partial ISBD (BK) [OBSOLETE]
          r - Provisional (VM MP MU) [OBSOLETE]
          u - Unknown
```

```
     19 - Linked record requirement
          # - Related record not required
          r - Related record required
     20-23 - Entry map
     20 - Length of the length-of-field portion
     21 - Length of the starting-character-position portion
     22 - Length of the implementation-defined portion
     23 - Undefined Entry map character position

DIRECTORY
   Character Positions
      00-02 - Tag
      03-06 - Field length
      07-11 - Starting character position

--Control Fields (001-006)--

001 - CONTROL NUMBER (NR)

003 - CONTROL NUMBER IDENTIFIER (NR)

005 - DATE AND TIME OF LATEST TRANSACTION (NR)

006 - FIXED-LENGTH DATA ELEMENTS--ADDITIONAL MATERIAL CHARACTERISTICS
        --GENERAL INFORMATION (R)
 006--BOOKS
  Character Positions
     00 - Form of material
        a - Language material
        t - Manuscript language material
     01-04 - Illustrations
     05 - Target audience
     06 - Form of item
     07-10 - Nature of contents
     11 - Government publication
     12 - Conference publication
     13 - Festschrift
     14 - Index
     15 - Undefined
     16 - Literary form
     17 - Biography
 006--COMPUTER FILES/ELECTRONIC RESOURCES
  Character Positions
     00 - Form of material
        m - Computer file/Electronic resource
     01-04 - Undefined
     05 - Target audience
     06-08 - Undefined
     09 - Type of computer file
     10 - Undefined
     11 - Government publication
  12-17 - Undefined
 006--MAPS
  Character Positions
     00 - Form of material
        e - Cartographic material
        f - Manuscript cartographic material
     01-04 - Relief
     05-06 - Projection
     07 - Undefined
     07 - Prime meridian [OBSOLETE]
     08 - Type of cartographic material
     09-10 - Undefined
     11 - Government publication
     12 - Form of item
     12 - Undefined [OBSOLETE]
     13 - Undefined
     14 - Index
     15 - Undefined
     16-17 - Special format characteristics
```

```
006--MIXED MATERIALS
  Character Positions
     00 - Form of material
        p - Mixed material
     01-05 - Undefined
     06 - Form of item
     07-17 – Undefined

006--MUSIC
  Character Positions
     00 - Form of material
        c - Notated music
        d - Manuscript notated music
        i - Nonmusical sound recording
        j - Musical sound recording
     01-02 - Form of composition
     03 - Format of music
     04 - Music parts
     05 - Target audience
     06 - Form of item
     07-12 - Accompanying matter
     13-14 - Literary text for sound recordings
     15 - Undefined
     16 - Transposition and arrangement
     17 - Undefined
 006--CONTINUING RESOURCES
  Character Positions
     00 - Form of material
        s - Serial/Integrating resource
     01 - Frequency
     02 - Regularity
     03 - ISSN Center [OBSOLETE]
     04 - Type of continuing resource
     05 - Form of original item
     06 - Form of item
     07 - Nature of entire work
     08-10 - Nature of contents
     11 - Government publication
     12 - Conference publication
      13-15 - Undefined
      16 - Original alphabet or script of title
      17 - Entry convention
 006--VISUAL MATERIALS
   Character Positions
      00 - Form of material
         g - Projected medium
         k - Two-dimensional nonprojectable graphic
         o - Kit
         r - Three-dimensional artifact or naturally occurring object
      01-03 - Running time
      04 - Undefined
      05 - Target audience
      06-10 - Undefined
      06-10 - Accompanying matter [OBSOLETE]
      11 - Government publication
      12 - Form of item
      12 - Undefined [OBSOLETE]
      13-15 - Undefined
      16 - Type of visual material
      17 – Technique
--Control Field 007--
```

Quadro 23 - Campos iniciais de um registro MARC 21