UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO
EM CIÊNCIA DA COMPUTAÇÃO

Romário Lopes Alcântara

Sobre o Desenvolvimento de Sistemas de Informação Utilizando Software Livre - Uma Experiência no Serviço Público

Dissertação submetida à Universidade Federal de Santa Catarina como requisito final para a obtenção do grau de Mestre em Ciência da Computação

Prof. Rosvelter João Coelho da Costa
Orientador

Florianópolis, Novembro de 2002

Sobre o Desenvolvimento de Sistemas de Informação Utilizando Software Livre - Uma Experiência no Serviço Público

Romário Lopes Alcântara

Esta dissertação foi julgada adequada para obtenção do título de Mestre em Ciência da Computação Área de Concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Prof. Fernando A. O. Gauthier, Dr.
Coordenador do Curso de Pós-Graduação
em Ciência da Computação

Banca Examinadora

_____

Prof. Rosvelter João Coelho da Costa, Dr. - UFSC
Orientador e Presidente da Banca

_____

Prof. Murilo Silva de Camargo, Dr. - UNB

_____

Prof. Roberto Willrich, Dr. - UFSC

Agradecimentos

Agradeço ao Curso de Pós-Graduação em Ciência da Computação e à Universidade Federal de Santa Catarina pela oportunidade, organização e infra-estrutura possibilitando a viabilidade do desenvolvimento deste trabalho.

Agradeço aos funcionários do CPGCC pela competência e pelo atendimento de qualidade. Em especial a Sra. Vera Sodré pela compreensão e atenção.

Agradeço a Prefeitura Municipal de Ijuí RS que proporcionou o espaço para que este trabalho pudesse ser desenvolvido e a Universidade de Ijuí, pelo apoio.

Agradeço aos professores Rosvelter João Coelho da Costa, Roberto Willrich, Murilo Silva de Camargo por participarem do julgamento desse trabalho.

Agradeço em especial ao Prof. Rosvelter João Coelho da Costa, pela paciência, dedicação e extrema competência com que me orientou, além dos conhecimentos que me transmitiu durante este projeto de dissertação.

Agradeço aos meus pais que sempre me apoiaram em todos os momentos.

Agradeço a minha esposa Roselma que sempre esteve ali dando força e estímulos para que eu conseguisse concluir este trabalho. Demonstrando com certeza o verdadeiro sentido do amor.

Ao meu filho Romário, que teve um pai ausente, mas galgando um degrau para que ele tenha todas as condições de ser um homem digno e honrado. Tu és uma das razões do meu esforço.

Sumário

Lista de Figuras

Lista de Quadros

Resumo

Esta dissertação descreve uma experiência de reengenharia de sistema caracterizado pela utilização de software livre no desenvolvimento de sistemas de informação para instituições públicas de pequeno porte.

No estudo foi desenvolvido um modelo de análise, considerando-se a implantação de um sistema totalmente novo e um modelo de dados legados construído a partir da base de dados do sistema existente. Este último foi proposto para permitir uma implantação gradual do novo sistema a partir de uma integração ortogonal com o sistema existente. O sistema existente continua funcionando normalmente enquanto o novo sistema é gradualmente implantado.

Para efeitos de avaliação prática, uma parte do sistema que foi proposto efetivamente foi implantado. Trata-se de um sistema de divulgação de dados da Secretaria da Saúde via Web pelo qual, usuários poderão acessar as informações geridas pelo sistema existente a partir de qualquer computador conectado a Internet. A implantação foi a custo zero para a prefeitura, pois é composto de componentes de software disponíveis gratuitamente.

Abstract

This dissertation presents a case study of system reengineering characterized by the free software utilization in the development of information systems for public institutions of small size.

In this study, a model of analysis was developed, considering the implementation of a completely new system and a legacy data model constructed from the existing system database. The last one was proposed to allow a gradual implementation of the new system, beginning with an orthogonal integration with the existing system. This system keeps on working normally while the new system is gradually implemented.

To a practical evaluation effect, part of the proposed system was effectively implemented. It is a system of *Secretaria da Saúde* (Healthy Department) data diffusion on the web, which allows the users to access the information produced by the system from any computer connected to the Internet. The implementation was costless to the municipal government as the system is composed by free cost software.

Capítulo 1

Introdução

O extraordinário avanço dos meios de comunicação e de tratamento da informação, em particular, o advento da Internet, trouxe consigo uma profunda mudança no relacionamento entre as pessoas e entre essas e as empresas e instituições publicas. As fronteiras foram expandidas aos quatro cantos da Terra e a informação, que antes costumava fazer a diferença, tornou-se banal. As empresas se viram diante de uma verdadeira “corrida espacial” com as suas concorrentes cujo prêmio era a sua própria sobrevivência no mercado. Esta corrida é claro, ainda esta longe de terminar, mas no nosso cotidiano já podemos ver boa parte do resultado.

Por outro lado, embora grandes empresas ou instituições públicas com grande orçamento já possam oferecer aos seus clientes ou usuários todos os benefícios da tecnologia Web, muitas instituições públicas, devido a suas restrições orçamentárias, mal conseguem manter um pequeno sistema de informação para seu uso interno. Como conseqüência, há hoje no Brasil dois tipos de serviço publico: o conectado na Web, onde podemos acompanhar obras, preencher formulários online, etc., e o conectado no balcão, com longas filas de espera. Diferentemente de empresas privadas, essas instituições públicas não podem quebrar ou sair do mercado.

Do lado da tecnologia de software, novas tecnologias foram desenvolvidas e disponibilizadas. Dessas, duas grandes vertentes têm ganhado a atenção do mercado nos últimos anos, a vertente Microsoft, através da tecnologia ASP, e a vertente Sun Microsystems, através da tecnologia JSP. Embora similares sob muitos aspectos técnicos, distinguem-se quanto ao relacionamento com o mercado. Diferentemente da Microsoft que guarda sobre si o essencial do controle no desenvolvimento da tecnológica ASP – o chamado padrão proprietário, a Sun Microsystems desenvolve a tecnologia JSP dentro de

uma perspectiva muito mais aberta – o chamado padrão aberto – e próxima de uma outra grande vertente de desenvolvimento de software que são os softwares livres – *freeware*.

Este trabalho de dissertação foi desenvolvido com o propósito principal de avaliar o uso de software livre no desenvolvimento de sistemas de informação de instituições publicas de pequeno porte como é o caso de muitas prefeituras do interior do Brasil. Especificamente, tratamos o caso da Prefeitura Municipal de Ijuí, no interior do Rio Grande do Sul. O sistema de informação existente nessa prefeitura foi desenvolvido em linguagem Cobol no ambiente Unix HP-UX sem nenhuma facilidade para a incorporação de novas funcionalidades ligadas a tecnologia Web.

A substituição desse sistema por outro totalmente novo esbarraria em uma grande dificuldade que é o custo, completamente proibitivo para uma prefeitura desse porte, pois exigiria a contratação de uma empresa terceirizada. Neste contexto, a solução adotada foi a especificação de um novo sistema, mas com sua implantação sendo feita de maneira gradual e continuamente integrada ao sistema existente.

Os Capítulos 2 e 3 apresentam um apanhado geral dos conceitos e ferramentas que foram envolvidos neste trabalho de dissertação.

O Capítulo 4 trata da especificação do novo sistema a partir da definição de um modelo de análise e da definição de um modelo de dados legados. O modelo de análise, efetivamente feita apenas para um pequeno grupo de funcionalidades, correspondendo às atividades de atendimento da Secretaria de Saúde, foi desenvolvido considerando-se a implantação de um sistema totalmente novo, mas ainda muito próxima ao sistema existente. Seguiu-se uma abordagem de reengenharia fortemente caracterizada pelo re-projeto. O modelo de dados legados construídos a partir da base de dados do sistema existente foi proposto para permitir a implantação gradual do novo sistema a partir de uma integração ortogonal com o sistema existente.

O Capítulo 5 descreve a implantação de uma parte do novo sistema. Trata-se de um sistema de divulgação de dados da Secretaria da Saúde via Web integrado ao sistema existente. Como resultado, usuários poderão acessar as informações disponibilizadas a

partir de qualquer computador ligado a Internet. A implantação foi a custo zero para a prefeitura, pois é composto de componentes de software disponíveis gratuitamente, do sistema operacional, Linux [CON] até a plataforma de desenvolvimento, J2SE da Sun Microsystems [J2SE].

O Capítulo 6 trata da conclusão e perspectivas de continuação do trabalho.

O custo zero não foi a única característica importante na abordagem de desenvolvimento de sistema proposta nesse trabalho de dissertação. Buscou-se também um modelo de integração completamente ortogonal. O sistema existente continua funcionando normalmente enquanto o novo sistema é gradualmente implantado. Um sistema não toma conhecimento do outro.

Capítulo 2

Conceitos e Métodos Aplicados ao Desenvolvimento de Software

2.1 Introdução

Há atualmente um grande número de técnicas que são empregadas no desenvolvimento de software de qualidade. Englobam um certo número de conceitos e métodos que aplicados às diferentes etapas do desenvolvimento ajudam-nos a obter softwares flexíveis, robustos e reutilizáveis.

Este capítulo destaca alguns dos conceitos e métodos mais comuns aplicado ao desenvolvimento de software.

2.2 Software

Há diversas definições, uma das mais empregadas é que software é um elemento do sistema lógico, sendo desenvolvido por engenharia e não manufaturado como o hardware [PRE-95].

O software vem evoluindo desde a década de 50, passando por diversas transformações em termos de técnicas e metodologias, sendo que no início havia poucos métodos sistemáticos e o desenvolvimento era feito sem qualquer administração causando sérios problemas em relação a custos e prazos. A Fig. 2.1 ilustra a evolução do software nos últimos 50 anos.

Primeira geração	**Segunda geração**	**Terceira geração**	**Quarta geração**
Proc. em batch	Multiusuários	Sistemas distribuídos	Sistemas desk-tops
Distribuição limitada	Tempo real	“Inteligência” embutida	Tecnologias O.O.
Software customizado	Banco de dados	Hardware baratos	Sistemas especialistas
	Produto de software	Impacto de consumo	Redes neurais artificiais
			Computação paralela

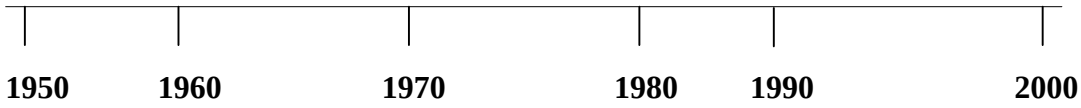

Figura 2.1 – Evolução do Software [PRE-95].

2.3 Sistemas

Em Informática é comum deparar-se com definições nas quais sistemas são considerados componentes que interagem atendendo necessidades específicas no ambiente em que estão presentes, atingindo objetivos comuns.

2.4 Engenharia de Software - princípios e objetivos

A grande preocupação nas décadas de 60 e 70 era ter um hardware que reduzisse o custo e armazenasse grandes volumes de dados e um software mais padronizado dentro de uma metodologia que pudesse proporcionar o cumprimento de cronogramas, controle de custos e prazos. Essa última parte compreende o que hoje em dia denomina-se por Engenharia de Software.

Em [PRE-95], a Engenharia de Software é definida da seguinte forma:

> *"É um rebento da engenharia de Sistemas e de Hardware. Ela Abrange um conjunto de três elementos fundamentais - Métodos, Ferramentas e Procedimentos que possibilita ao gerente o controle do processo de desenvolvimento do software e oferece ao profissional uma base para construção de software de alta qualidade"*.

A Engenharia de Software tem como objetivo auxiliar o processo de produção de software, de forma a gerar produtos de alta qualidade, produzidos mais rapidamente e a um custo cada vez menor.

Sabe-se que existem muitos problemas a serem tratados pela Engenharia de Software, pois tanto o processo de produzir o software quanto o produto produzido possuem vários atributos que devem ser considerados para que se tenha sucesso, por exemplo, a complexidade, a visibilidade, a aceitabilidade, a confiabilidade, a manutenibilidade e a segurança figuram entre os mais importantes [PRE-95].

A Engenharia de Software herdou da engenharia o conceito de disciplina na produção de software, através do uso de *metodologias* que, por sua vez, seguem *métodos* que utilizam *ferramentas* automatizadas para englobar as principais atividades do processo de produção de software. Alguns métodos focalizam as funções do sistema; outros se concentram nos objetos que povoam; outros, ainda, baseiam-se nos eventos que ocorrem durante o funcionamento do sistema. Portanto são vários conceitos que se completam [PRE-95].

Esses princípios referem-se tanto ao processo como ao produto final e descrevem algumas propriedades gerais dos processos e produtos. O *processo* correto ajudará a produzir o *produto* correto e o *produto* almejado também afetará a escolha do *processo* a ser utilizado.

Para isso o engenheiro de software deve estar equipado com metodologias apropriadas e com métodos e ferramentas especificas que o ajudarão a incorporar as propriedades desejadas aos processos e aos produtos, além do mais, os princípios devem

guiar a escolha das metodologias, métodos e ferramentas apropriadas para o desenvolvimento de software.

Os princípios seriam os seguintes: formalidade, abstração, decomposição, generalização e a flexibilização.

2.5 Reengenharia de software

A reengenharia de software ocorre quando um sistema de software existente tenha passado por sucessivas atualizações, tornando a manutenção difícil e dispendiosa.

Com o passar do tempo, um software legado pode tornar-se tecnicamente obsoleto devendo ser substituído ou reestruturado para que possa atender as necessidades atuais. Um sistema legado é aquele que foi desenvolvido para satisfazer as necessidades de uma empresa mas que não acompanhou a evolução tecnologica.

Em [PRE-95], para o software a engenharia reversa é o processo de analisar um programa, num esforço para criar uma representação do programa em nível de abstração maior do que o código fonte, sendo um processo de recuperação de projeto.

Em [CHI-90] a reengenharia também chamada de renovação ou recuperação, não somente recupera informações de projeto de software existente, mas usa essas informações para alterar ou reconstituir o sistema existente num esforço para melhorar a qualidade global.

Um outro aspecto que deve ser observado é que na maioria dos casos, o software que sofre uma reengenharia refaz as funções do sistema existente melhorando o desempenho global.

A reengenharia de um sistema implica examinar e alterar um sistema de software de forma a reconstituí-lo sob um novo formato. Isso significa que deverá ser aprimorando a documentação (compreendendo e descrevendo o que o sistema faz; as suas entradas, saídas e as suas ações), projetar e reestruturar um sistema existente de forma a obter uma forma mais aceitável do mesmo [PET-01].

2.5.1 Subprocessos do modelo de reengenharia de software

Os dois principais subprocessos de um projeto de reengenharia são [PET-01]:

- Engenharia progressiva;

- Engenharia reversa.

Na engenharia progressiva, há uma seqüência bem definida de atividades que levam o projetista da análise e descrição dos requisitos para o projeto e desenvolvimento.

A engenharia reversa, parte de um sistema existente, analisando-o de forma a identificar seus componentes e as inter-relações dos mesmos. A engenharia reversa tem como um dos principais benefícios, a obtenção de resultados na recuperação de informações e estruturas úteis.

A engenharia reversa é executada com o objetivo de obter uma melhor compreensão de um sistema existente, sendo um processo de análise. Como conseqüência disso a engenharia de software reconstrói as especificações de projeto (aprimorando a descrição de sua própria arquitetura e de sua descrição) de forma a preparar o terreno para um esforço de engenharia progressiva [PET-01].

A fig. 2.2 mostra um modelo de reengenharia.

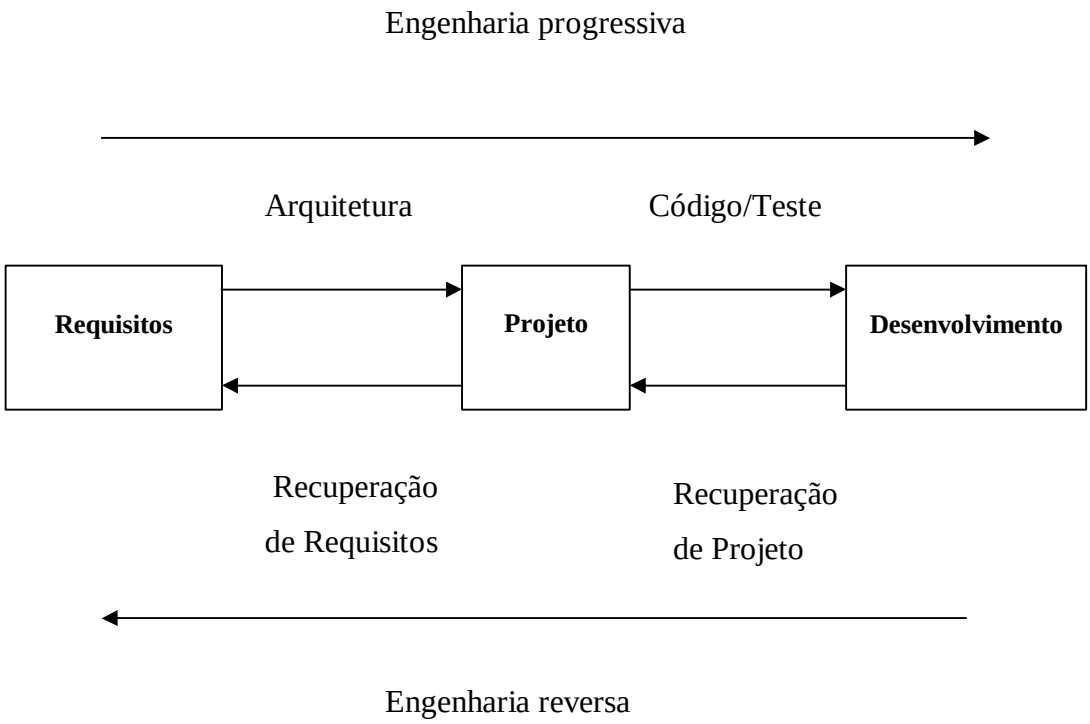

Figura 2.2 – Modelo de reengenharia de software [PET-01]

2.5.2 Propósitos da reengenharia de software

A reengenharia traz para um sistema legado, diversos benefícios como:

- Aumento da manutenibilidade de software;
- Identificação dos candidatos ao reuso;
- Valorização do sistema que sofre o processo de reengenharia;
- Aprimoramento da gestão de riscos do projeto.

Neste projeto a reengenharia contempla dois aspectos que é a implantação gradual e a integração ortogonal.

2.6 A abordagem orientada a objetos

Tudo começou no final dos anos sessenta pelas pessoas que utilizavam a linguagem Simula, que por sua vez nos anos setenta integrou-se à linguagem Smalltalk, desenvolvida pela empresa Xerox PARC, considerada a primeira linguagem totalmente orientada a objetos.

Objetos

A palavra objeto refere-se a elementos pertencentes ao mundo real, que se pode ver, sentir ou utilizar, mas que não se tem o conhecimento preciso de seus componentes internos e de seu comportamento. Por exemplo, uma mesa, um carro, um animal.

Em [COA-93], um objeto é uma abstração de alguma entidade no domínio do problema ou em sua realização, refletindo a capacidade de um sistema manter informações sobre ela, interagir com ela, ou em ambos os casos.

Na programação orientada a objetos pode-se definir objeto como uma estrutura que agrupa dados (geralmente denominados atributos) e funções para manipulação de dados (geralmente denominados métodos).

Sendo as funções definidas sobre os dados do objeto, o sistema todo funciona de maneira mais segura e confiável, portanto, um objeto do tipo animal não é composto somente por seus atributos (cor, tamanho, pelagem, etc.), mas também por funções tais como: crescer, reproduzir-se, etc. Um círculo é uma figura geométrica assim como o quadrado ou o triângulo, mas cada um tem as suas peculiaridades e comportamento.

Classes

São categorias de objetos. Todo objeto é uma instância de alguma classe de objetos. A classe descreve um grupo de objetos com as mesmas propriedades (atributos), e comportamento (métodos) similares.

Pode-se dizer que a classe é um tipo definido pelo usuário que contém o molde de seus objetos, suas especificações, assim como o tipo inteiro contém o molde para valores declarados como inteiros.

Ao construir uma classe, descreve-se um tipo de objeto. Ela especifica uma estrutura de dados e um conjunto de operações para permitir o acesso e a modificação dos dados do objeto. Por exemplo, uma classe de nome Empregado poderia incluir operações, tais como “contratar”, “promover”, “demitir”.

Os detalhes sobre o método da operação são especificados pela classe e são compartilhados por todos os objetos da classe.

A Fig. 2.3 representa uma classe na notação UML que descreve objetos do tipo `Ponto`.

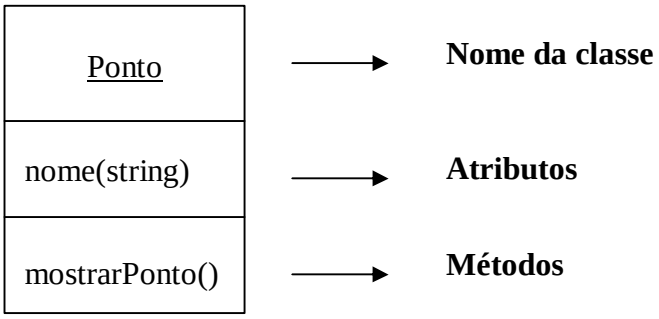

Figura 2.3 – Representação padrão de classes na notação UML.

A representação desta mesma classe em código Java é apresentado no Quadro 2.1.

```
class Ponto   {
    private int x, y;

    public Ponto (int x, int y) {
        this.x  = x;
        this.y  = y;
    }

    public int  get X () {
          return (this.x);
     }

     public void mostra  () {
           System.out.println("X= "+ x + ",  Y= "+y);
     }
...

}
```

Quadro 2.1– Declaração de classe na linguagem Java.

Encapsulamento

Um objeto pode ocultar seus dados de outros objetos e permitir que eles sejam acessados somente por intermédio de seus próprios métodos. Este procedimento é utilizado para evitar que sejam inadvertidamente corrompidos ou utilizados de forma inadequada.

Em [LEW-00], encapsulamento é uma das características da orientação a objetos através da qual os atributos de um objeto são alterados somente através dos métodos do próprio objeto.

O encapsulamento é isso, oculta detalhes de um objeto para oferecer ao seu ambiente um modelo de utilização do objeto seguro e eficiente. Isso permite que o desenvolvimento do objeto seja modificado sem exigir que os aplicativos que as usam, sejam também modificados.

O comportamento do tipo de objeto pode ser modificado e testado, independentemente de outros tipos de objetos.

Somente os métodos especificados são acessíveis. O encapsulamento impede a interferência nas partes internas e oculta a complexidade de seus componentes.

Mensagens

Objetos cooperam entre si através da troca de mensagens. Ao receber uma mensagem, um objeto reage executando um comportamento que freqüentemente resulta no envio de outras mensagens e modificação de seus atributos.

Por exemplo, ao acionar o comando dos *flaps* de um avião, o piloto percebe uma ação sendo executada e recebe a resposta através de seu painel de controle.

Todos os objetos aviões são controlados dessa forma, mas o piloto não poderá mudar a temperatura do *cockpit* usando o comando dos *flaps*, porque aqui outro tipo de mensagem será necessário.

Herança

O conceito de herança é um dos que fazem com que a abordagem orientada a objetos tenha como grande atrativo à aceleração do processo de desenvolvimento. Se já existir uma classe que pode responder a uma parte das mensagens desejadas, porque construir uma nova classe apenas para responder a algumas mensagens adicionais?

As linguagens de programação orientada a objetos mais comuns permitem que se crie uma classe que herda todas as características de uma outra classe. Respectivamente, essas classes são denominadas de subclasses e superclasses. Isso permite que um novo programa, possa utilizar classes já existentes, modificando-as através do acréscimo de novos atributos ou métodos. Subclasses podem também sobrescrever métodos da superclasse se desejarem fornecer um comportamento mais especializado para o método.

A Fig. 2.4 ilustra um exemplo de herança e o Quadro 2.3 ilustra o código em Java para este mesmo exemplo.

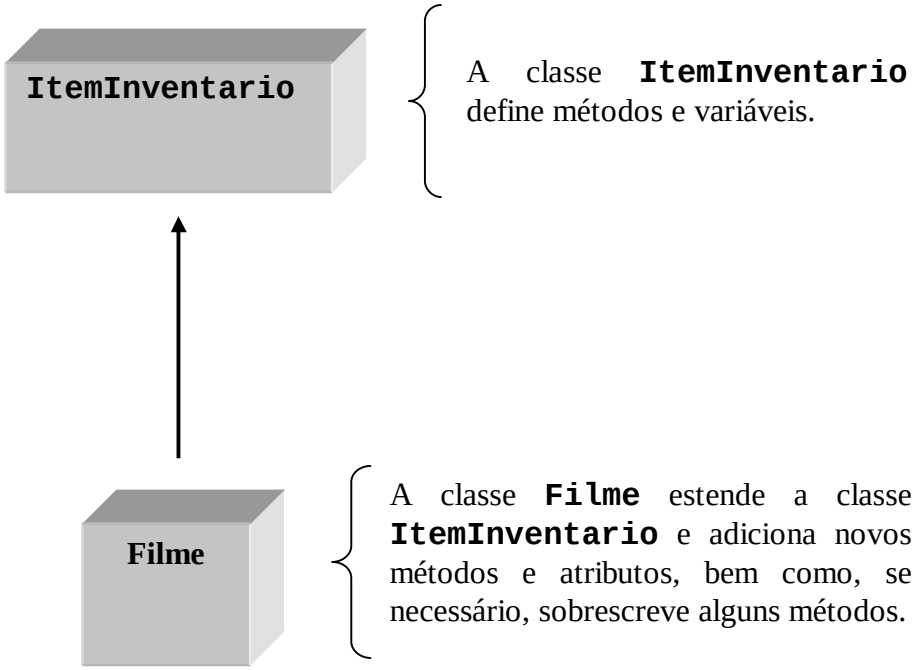

Figura 2.4 – Herança.

```
public class  ItemInventario
{
   // Atributos e métodos da classe ItemInventario
}


public class Filme extends ItemInventário
{
  // Atributos e métodos adicionais da classe Filme.
}
```

Quadro 2.2- Herança em Java.

Polimorfismo

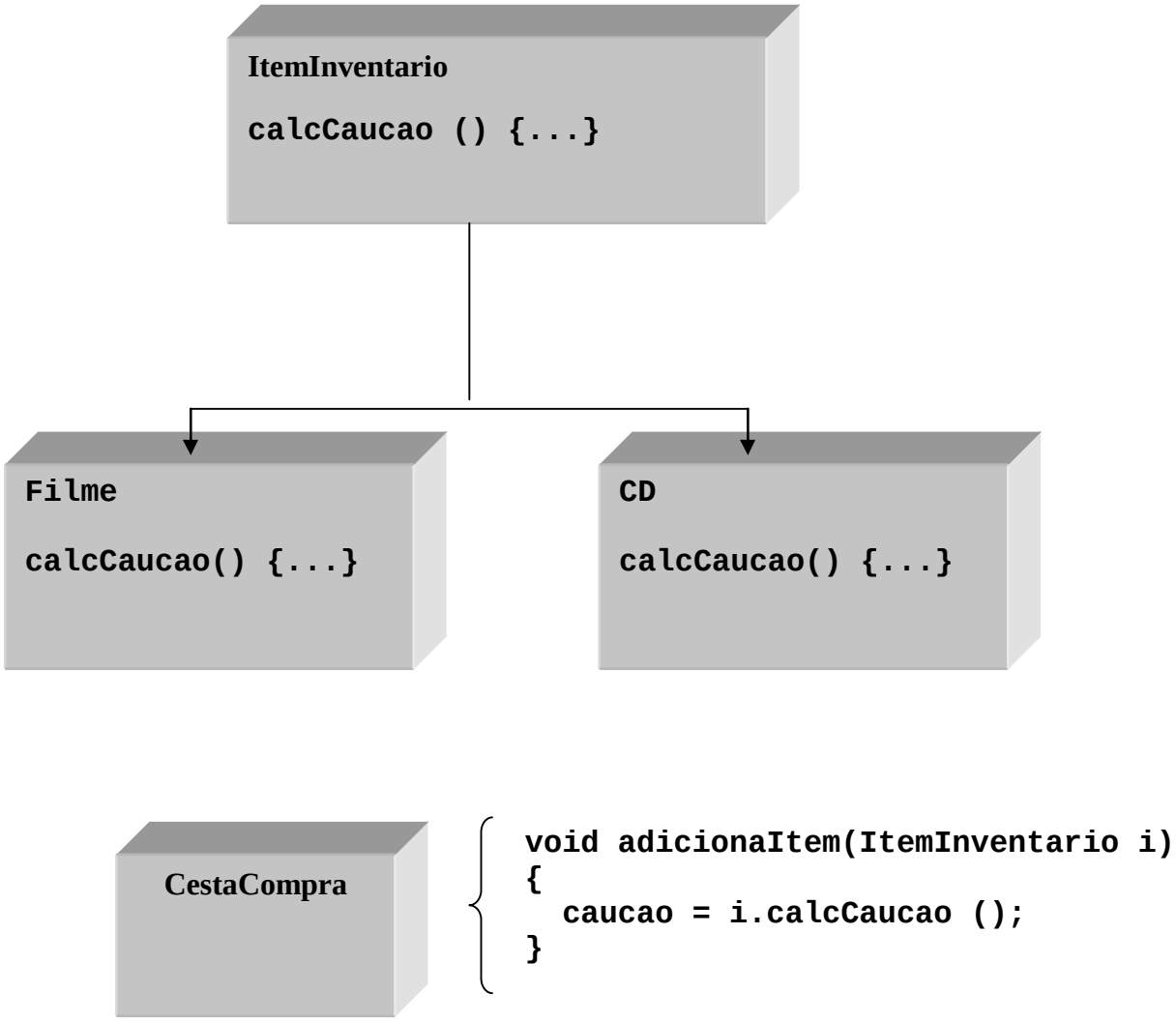

Figura 2.5 - Estrutura do polimorfismo do exemplo.

Pode ser definido como a capacidade de objetos de classes diferentes responderem à mesma solicitação, cada um a seu modo. Por exemplo a mensagem “imprima”, pode gerar resultados diferentes dependendo do tipo de impressora à qual foi enviada.

A Fig. 2.5 mostra um exemplo (em Java) em que o vendedor tem que determinar o preço de cada um dos itens, bem como a caução de um determinado produto que será alugado.

A classe **`CestaCompras`** foi projetada para simplesmente aceitar e processar itens de inventário (objeto do tipo **`ItemInventário`**), sejam quais forem os seus tipos. Java se encarregará de determinar o tipo do item e chamará o método correto baseado neste tipo.

Usando-se esta técnica, se pode adicionar quantos itens forem necessários sem a necessidade de alterar ou recompilar o código antigo.

O método **`calcCaucao`**() da classe **ItemInventário** é sobrescrito pelas classes Filme e CD, cada qual fornecendo uma versão especializada do método.

A classe **`CestaCompra`** possui um método **`adicionaItem(ItemInventárioitem)`** que chama o método **`valorAPagar()`** e **`calcCaucao()`** através da referência recebida a um objeto **`ItemInventário`**.

O exemplo acima mostra que o mais importante é que as classes **`CestaCompra`** e **`ItemInventario`** não precisam ser modificadas quando novos tipos de itens de inventário forem adicionados ao negócio.

2.7 Benefícios da abordagem orientada a objetos

A mudança de paradigma, de estruturada para orientada a objetos, trouxe benefícios bastante significativos também. Entre os mais importantes, destacam-se:

Rapidez

A demora no desenvolvimento implica em elevação de custos e perda de oportunidades de mercado. A abordagem orientada a objetos permite que aplicativos, mesmo pequenos, sejam criados a partir de componentes pré-existentes, aumentando a velocidade no desenvolvimento.

Confiabilidade

A construção de softwares utilizando-se componentes já testados e aperfeiçoados aumenta a confiabilidade de novos produtos.

Reuso

Classes podem ser projetadas de modo a serem utilizadas em uma grande variedade de casos. Diversos mecanismos têm sido proposto a esse respeito, entre eles, os *frameworks* e os padrões de projeto (*patterns*).

2.8 Metodologias de desenvolvimento de software

Objetos e mensagens são mecanismos chaves da abordagem orientada a objetos. De um modo geral, desenvolver um sistema orientado a objetos envolve, inicialmente, a identificação e especificação de objetos, e seus relacionamentos.

As funcionalidades exigidas do sistema devem estar equacionadas em termos de troca de mensagens entre os objetos e da especificação dos métodos correspondentes. Este esforço deve ser complementado pela incorporação de uma interface mais amigável para o usuário e pela integração do sistema ao ambiente de projeto.

Uma metodologia de desenvolvimento proporciona uma abordagem sistemática para o desenvolvimento e o detalhamento de sistemas de software, proporcionando elementos para assegurar que o sistema que será construído estará correto.

Inúmeras metodologias têm sido propostas na literatura [PRE-95]. Entre elas, destacam-se as seguintes: *Coad/Yourdon (OOA/OOD), OOSE (Object Oriented Software Enginiering) de Jacobson, OMT (Object Modeling Techinique) de Rumbaugh, Fusion de Coleman e a de Booch.* De uma forma geral, possuem as seguintes características [GIO-97]:

- Transições suaves entre as etapas de análise e projeto durante o ciclo de vida do desenvolvimento;

- Visualização do processo de desenvolvimento como uma atividade de modelagem, proporcionando diferentes, porém relacionados, tipos de modelos;

- Proporcionam conceitos, notações, regras e modelos bem definidos, completos e coerentes;

- Preservam, na etapa de projeto, as informações e as relações definidas na etapa de análise;

- Suportam aspectos técnicos e gerenciais para o desenvolvimento de sistemas computacionais (não pode ser apenas uma curiosidade de laboratório);

- Estão disponíveis em forma de livro didático, com conteúdo completo e consistente.

2.9 A Notação UML (*Unified Modeling Language*)

A UML unificou três grandes metodologias de desenvolvimento de software: o Método de Booch, o Método de Rumbaugh (OMT) e o Método de Jacobson. A UML foi padronizada pela OMG[1] [OMG, BOO-99].

Uma metodologia de desenvolvimento de software consiste de uma linguagem de modelagem, geralmente constituída de notações gráficas para representar os diferentes elementos de um software e de um processo que é a sugestão de quais os passos a serem seguidos na elaboração de um projeto. A UML é apenas a linguagem [OMG -99].

[1] A OMG é uma associação internacional onde participam diversas empresas, ligadas a produção de software, sendo constituída principalmente por empresas americanas e européias.

2.10 Diagramas

Diferente da linguagem natural, a notação UML descreve o propósito de uma ação construindo uma representação gráfica do seu conteúdo ou comportamento. Com este propósito a linguagem UML define diversos diagramas de descrição. São eles:

- Diagrama de Casos de Uso
- Diagrama de Classe
- Diagrama de Interação
    - ✍ Diagrama de Seqüência
    - ✍ Diagrama de Colaboração
- Diagrama de Estados (*Statechart*)
- Diagrama de Atividades
- Diagrama Físicos

Neste trabalho de dissertação utilizou apenas alguns diagramas. Um apanhado geral sobre alguns deles é apresentado na seqüência.

Diagrama de casos de uso

É com este diagrama que geralmente inicia-se o processo de desenvolvimento de um sistema orientado a objetos. Eles denotam cenários de utilização do software que será construído.

Um cenário é uma seqüência de passos que descreve uma interação entre um usuário e um determinado sistema. Por exemplo, um usuário requisitando um atendimento médico na Secretaria da Saúde:

O usuário solicita uma consulta com um determinado médico. O sistema verifica se o médico encontra-se disponível, marca um horário para o usuário, se for o caso, e envia um e-mail ao usuário confirmando a consulta.

Embora este cenário possa realmente acontecer, poderá haver problemas no que diz respeito ao médico estar ou não disponível (em férias, por exemplo), o usuário não foi autenticado pelo sistema, entre outros.

Os diagramas de casos de uso denotam um conjunto de cenários unidos por um objetivo comum. No exemplo acima, haveria um caso de uso para a solicitação de consulta e outro para a marcação de um horário.

Os diagramas de casos de uso também utilizam um outro tipo de elemento de descrição – os atores – que por sua vez podem estar ligados a mais de um caso de uso e um caso de uso pode estar reciprocamente ligado a vários atores. Os atores estimulam o sistema com eventos de entrada e recebem algo do mesmo [LAR-00]. Eles não são necessariamente seres humanos. Embora representados como bonecos nos diagramas de casos de uso, podem também ser um sistema externo que troca alguma informação com o sistema analisado. A Fig. 2.6 fornece a notação gráfica utilizada para representar atores.

Usuário Sistema de Atendimento

Figura 2.6 – Notação para atores em UML.

Diagramas de Classes

A notação UML define um grupo de notações para representar classes e seus relacionamentos. Como classes não existem isoladamente, relacionamentos entre as mesmas são fundamentais. A notação UML descreve vários tipos de relacionamentos. São eles:

- Generalização/Especialização
- Agregação
- Associação
- Dependência
- Refinamento
- Composição
- Realização

A Fig. 2.7 exemplifica alguns desses relacionamentos.

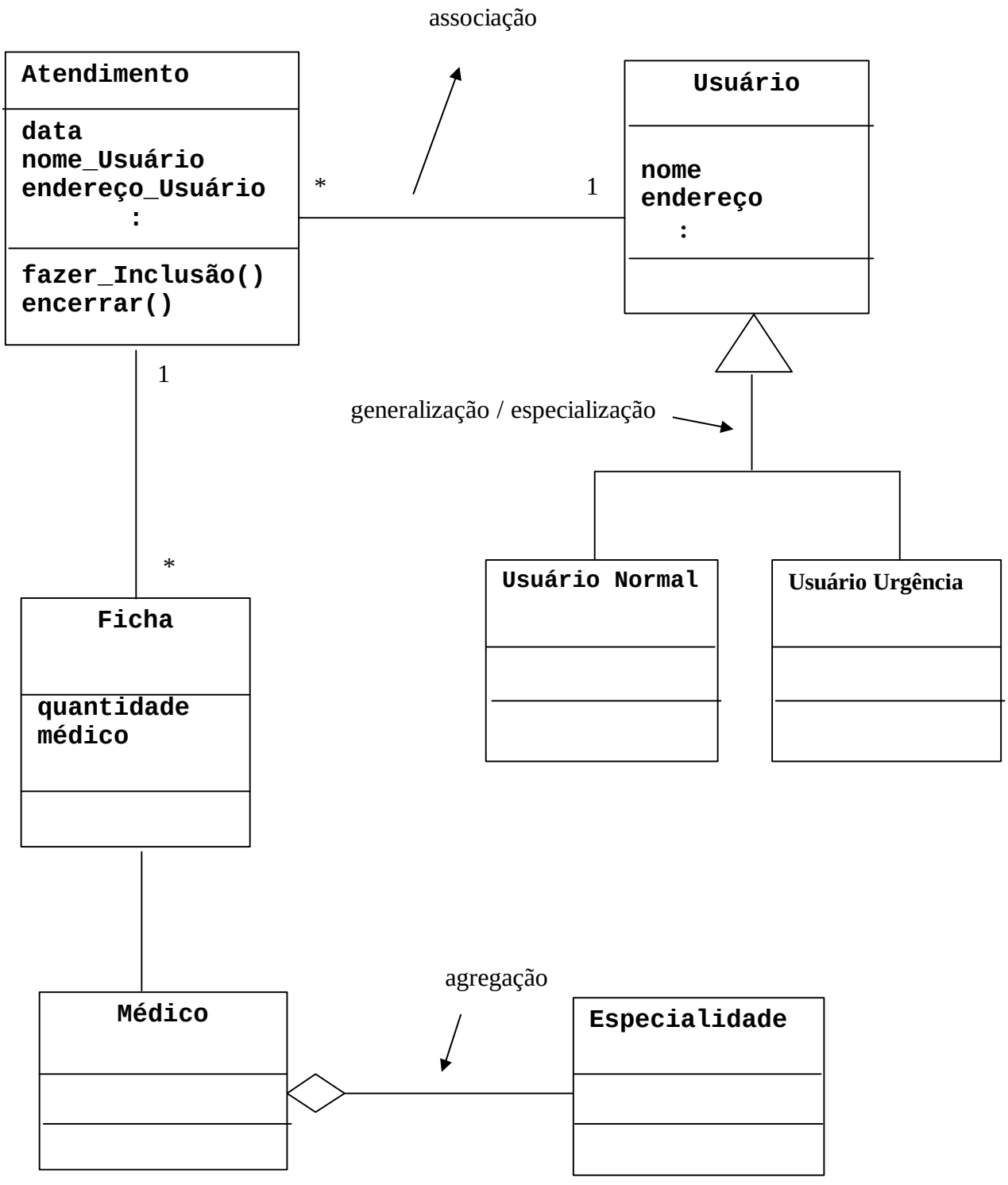

Figura 2.7 – Exemplo de diagrama de classes em UML

Diagrama de Seqüência

Os diagramas de seqüência descrevem uma possível colaboração finita entre um grupo de objetos. O comportamento de cada objeto é representado por uma linha (a linha da vida do objeto) sobre o qual descreve-se um ordenamento de recepção e emissão de mensagens.

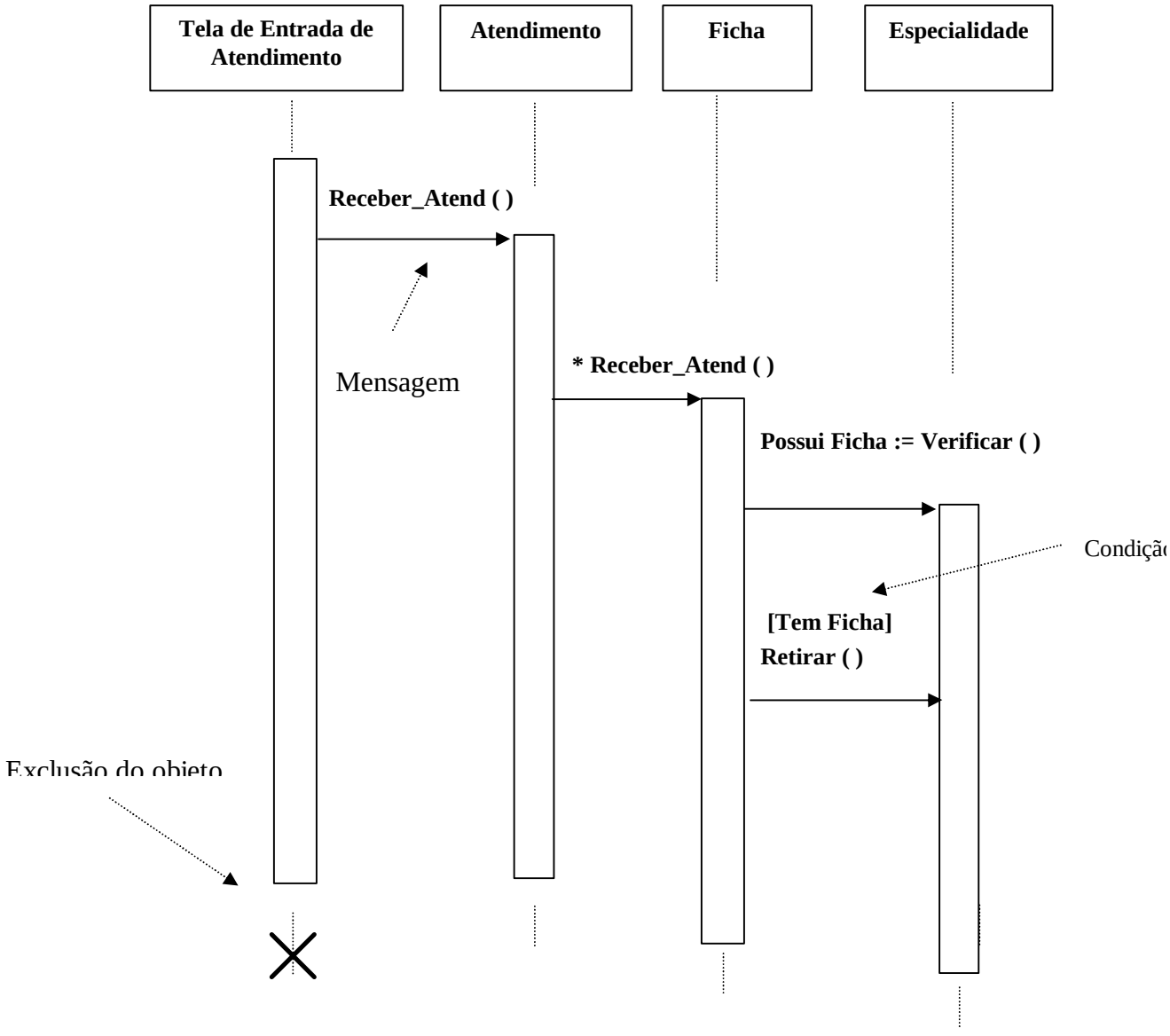

Figura 2.8 – Exemplo de diagrama de seqüência em UML.

A Fig. 2.8 apresenta um exemplo de diagrama de seqüência descrevendo o seguinte comportamento:

- Um objeto tela de entrada de atendimento envia uma mensagem **`Receber_Atend()`** para um objeto atendimento.

- O objeto atendimento então envia a mensagem **`Receber_Atend()`** para um objeto ficha que por sua vez verifica se há alguma disponível para tal especialidade.

- Se houver, retira uma do montante – mensagem **`Retirar()`**.

Diagramas de Estados (*Statechart*)

Os diagramas de estados descrevem o comportamento de um sistema, mostrando todos os estados possíveis de um objeto e como a mudança de estado ocorre. Na maioria das vezes este diagrama descreve como um objeto de uma classe se comportará ao longo do seu tempo de vida.

Diagramas de estados são mais conhecidos como *statecharts*, modelo originalmente introduzido por David Harel em 1987.

A Fig. 2.9 apresenta o diagrama de estado para um sistema de administração de saúde já visto anteriormente.

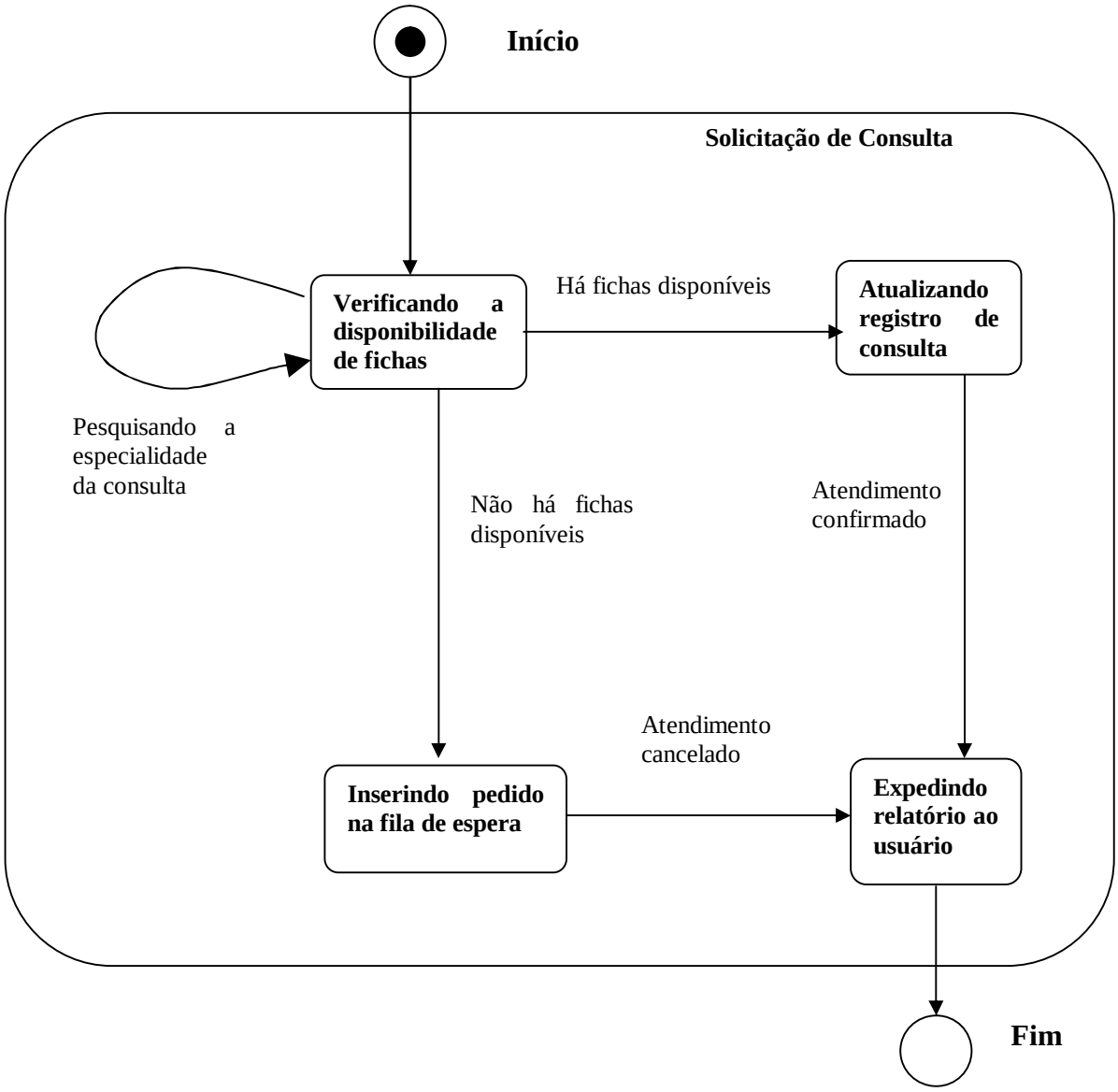

Figura 2.9 - Exemplo de diagrama de estados em UML.

2.11 A Internet

A Internet representa sem dúvida um dos maiores avanços tecnológicos dos últimos tempos. Permite a troca de diversos tipos de informação aproximando pessoas e criando novas tendências de mercado. Por exemplo, o comércio eletrônico que estende os processos de negócios além das fronteiras físicas da empresa, integra clientes, fornecedores e faz com que as empresas reduzam os seus custos drasticamente.

A Internet é um mecanismo de disseminação da informação e um meio para colaboração e interação entre indivíduos e seus computadores, independentemente de suas localizações geográficas.

Começou em 1969, quando o Departamento de Defesa Americano sentiu que precisava construir uma rede de comunicação viável que unisse os seus diversos centros de decisão. A essa rede deu-se o nome de ARPANET e teve como ponto positivo o fato de admitir diferentes sistemas, apenas com a condição de que todos deveriam “falar” a mesma linguagem, a qual se chamava NCP (*Network Control Protocol*).

O projeto Interneting, datado de 1973 tinha como objetivo desenvolver um protocolo de comunicação entre computadores através de diferentes redes, do qual se derivou o nome *Internet* (Inter Redes). Deste projeto surgiu um conjunto de protocolos denominados TCP/IP.

O TCP subdivide (nos chamados pacotes de informação) as mensagens geradas na origem e os recompõe no destino. O IP é responsável pelo direcionamento dos pacotes que viajam por rotas diversas, atravessando várias estações (retransmissores) e diferentes redes e meios de transmissão.

Na década de 80, surgiram mais algumas redes científicas que permitiram que a utilização do protocolo TCP/IP fosse generalizada. Em 1983 três redes interligadas (entre elas a ARPANET) formavam aquilo que se convencionou chamar de *Internet*.

Com a diminuição do custo dos computadores pessoais, a sua utilização ao longo dos tempos foi se popularizando, assim como o aumento da potência dos processadores foi

permitindo novas formas de acesso e transmissão de dados em diversos formatos, tais como o texto, a imagem, o som e até o vídeo.

Surgiram também por isso interfaces que permitiam o acesso à informação multimídia e programas que permitem a ligação de um computador pessoal a um servidor de informação. Surgiu a Web.

2.12 A Web

No ano de 1993, surge a World Wide Web ou simplesmente Web. É a aplicação da Internet criada para organizar as associações de recursos de vários tipos, quer no mesmo arquivo, quer em diferentes arquivos em diversas partes do mundo.

A Web fornece a cada documento um endereço e um nome, indicando onde esta situada (servidor) e o modo de acesso. A esse conjunto, endereço mais modo de acesso, denomina-se URL.

O protocolo que permite a localização e transferência de recursos na Web é o HTTP. Este protocolo assume dois tipos de entidades, o cliente e o servidor.

O cliente interage com o servidor através de um documento HTML contendo ligações (links) para outros recursos na Internet (URL), em geral, um outro documento HTML.

Ao clicar em uma dessas ligações, o servidor contendo o recurso associado à ligação envia à plataforma cliente (reprodutor HTML) e o processo se reinicia.

2.13 Banco de dados

A disponibilidade de armazenamento maciço e barato de acesso direto causou uma quantidade tremenda de atividade de pesquisa e desenvolvimento na área de sistemas de banco de dados.

Em [SIL-99] aparece a seguinte definição de Banco de dados “Coleção de arquivos e programas inter-relacionados que permitem a um mesmo usuário o acesso para consultas e alterações a esses dados”.

Os sistemas bancos de dados fornecem diversas vantagens ao projeto de sistemas, são elas:

- Mecanismos de tratamento de redundância e inconsistência;
- Compartilhamento de dados;
- Utilizações de padrões;
- Aplicação de restrição de segurança;
- Manutenção de integridade;
- Adequação de requisitos contraditórios;
- Independência de dados.

Programas acessam as informações de um banco de dados através de uma linguagem de banco de dados. Esta, por sua vez, é geralmente composta de duas outras linguagens: DDL (*Data Definition Language*) representando a linguagem de definição de dados e a DDL (*Data Manipulation Language*) representando a linguagem de manipulação de dados. As linguagens SQL fornecem ambas.

2.14 Síntese

Este capítulo apresenta um apanhado geral sobre diferentes tópicos relacionados ao desenvolvimento de software e reflete em parte o estudo bibliográfico deste trabalho de dissertação.

Os conceitos e métodos que foram apresentados serviram de base para o projeto de software que foi desenvolvido.

Capítulo 3

Ferramentas de Desenvolvimento de Software

3.1 Introdução

Inúmeras ferramentas de software têm sido propostas para dar suporte ao desenvolvimento de aplicações distribuídas. Este capítulo dedica-se a apresentação de algumas delas. Em particular, no detalhamento das ferramentas que foram efetivamente utilizadas no projeto desenvolvido nesta dissertação.

3.2 O Sistema gerenciador de banco de dados MySQL

Sistemas gerenciadores de bancos de dados desempenham um papel preponderante no desenvolvimento de sistemas de informação. O sistema gerenciador de banco de dados MySQL foi escolhido porque além de atender a um dos principais requisitos não funcionais do projeto que é a gratuidade, o mesmo possui excelente desempenho ao acesso aos dados.

Possui todas as funcionalidades requisitadas pelo projeto, é rápido, eficiente, estável, experimentado em diversas organizações e gratuito (software livre). Por essas razões ele foi escolhido para integrar este projeto.

Além do MySQL, existem outros que poderiam ter sido usado, como exemplo o Postgres, que possui excelente desempenho, estável é gratuito, mas o MySQL é mais simples e possui uma boa documentação, sendo por estas razões o escolhido.

3.3 JDBC (*Java Data Base Conection*)

Visão geral

A plataforma Java oferece uma interface para acesso a bases de dados que é independente do sistema de banco de dados ou o mecanismo de conexão utilizado. Essa interface de acesso compreende na verdade toda uma API, conhecida pelo nome de JDBC, possibilitando através de comandos na linguagem SQL a integração do código na linguagem Java com sistemas de banco de dados [DEI-01].

A especificação do JDBC foi baseada no padrão X/Open SQL *Call Level Interface* (CLI), e o deixou muito parecido com um driver ODBC (*Open Data Base Connectivity*) que também foi baseado neste mesmo padrão.

A principal diferença é que o ODBC é um procedimento em linguagem C nativa enquanto que o JDBC pode ser desenvolvido independentemente da plataforma de desenvolvimento.

A API JDBC define um conjunto de classes que representam conexões à banco de dados, consultas na linguagem SQL e uma série de outras ações permitindo que o programador Java consulte o banco de dados e manipule o seu resultado.

A API funciona através de um gerenciador de drivers (classe da linguagem Java chamada DriverManager), compatível com múltiplos drivers e, por isso, permite a conexão com diferentes bancos de dados.

O JDBC é uma interface que foi planejada tanto para ser utilizada pelo programador através de comandos na linguagem SQL, como para servir de base para interfaces de mais alto nível. Existem basicamente dois tipos de API de alto nível desenvolvidas sobre o JDBC. São elas:

SQL embutido

O mapeamento é direto do banco de dados relacional para classes Java, procurando evitar a incompatibilidade de diversos tipos de linguagem SQL. O JDBC aceita o padrão ANSI SQL-92 que garante a portabilidade das aplicações.

API de terceiros

Neste caso uma das mais conhecidas é a API ODBC. Ela permite que qualquer string de consulta seja passada para algum driver específico de banco de dados, propiciando a aplicação, o uso de toda a funcionalidade do produto.

A arquitetura JDBC

Um programa Java utiliza uma API JDBC única que independe do banco de dados ou driver que estiver sendo utilizado.

Os drivers para conexão e acesso aos principais bancos de dados existentes são fornecidos pelos seus fabricantes. Neste caso, o programador só precisa saber utilizar o driver adequado e a API JDBC.

A Fig. 3.1 apresenta a estrutura funcional da API JDBC.

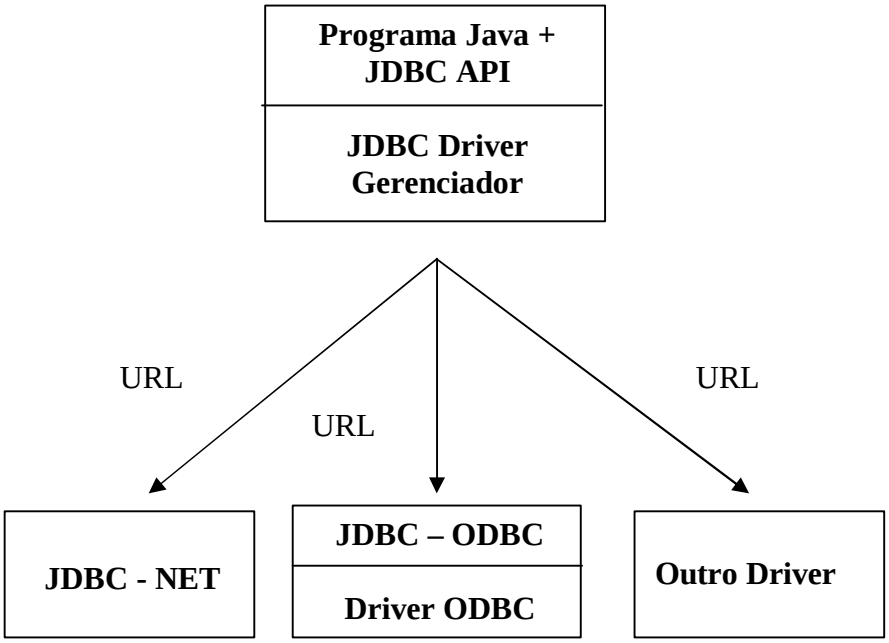

Figura 3.1 - Arquitetura JDBC.

O objetivo principal do JDBC é de ser uma interface, independentemente do sistema de gerenciamento de banco de dados (SGBD), que permita o acesso genérico ao banco de dados através da linguagem SQL fornecendo uma interface uniforme para diferentes fontes de dados.

3.3.1 Tipos de drivers

Os drivers JDBC podem ser escritos totalmente na linguagem Java, podendo ser instalados em uma máquina cliente através de um applet ou de métodos nativos que permitem a conexão com a biblioteca de acesso aos dados [DEI-01].

A API JDBC exige que o driver suporte no mínimo a funcionalidade do padrão ANSI SQL-92 para passar nos testes de compatibilidade do JDBC. Os drivers são divididos em quatro categorias, são elas:

Drivers com acesso para API nativa

Este tipo de driver efetua o mapeamento das chamadas JDBC para chamadas na API cliente. Encontrados nos principais bancos de dados comerciais (Oracle, DB2, Sybase, Informix) e também nos livres (MySQL, Postgress, entre outros). Exige a instalação de código binário em cada máquina cliente.

Drivers independentes de protocolos de rede

Conhecidos pelo nome de *Thin Drivers*. São considerados os mais flexíveis, pois possuem um protocolo de rede independente e podem ser traduzidos para os protocolos de acesso a bancos de dados específicos [DEI-01].

São ideais para intranets. Neste caso, devem satisfazer as normas de segurança impostas na Web para os applets.

Drivers dependentes de protocolos de rede

As chamadas JDBC são traduzidas para os protocolos de rede específicos utilizados pelos bancos de dados relacionais. Um exemplo é um driver JDBC, ligado a algum fabricante, que utiliza um protocolo diferente.

Drivers que utilizam ODBC

Funciona através de uma ponte, ligação JDBC-ODBC para manter a compatibilidade com uma grande variedade de bancos de dados existentes no mercado.

3.3.2 A API JDBC

As classes da API JDBC são totalmente em código Java e executadas em qualquer plataforma que tenha a máquina JVM instalada. Além das classes, o uso do JDBC requer um ou mais drivers JDBC.

As classes que formam a API JDBC estão no pacote *Java.sql* e são as seguintes:

DriverManager

Tem por função passar as restrições de conexão para os drivers JDBC.

DriverPropertyInfo

É uma instância de propriedades de uma conexão de um driver JDBC.

Types

Lista de constantes simbólicas de tipos de dados data SQL.

Time

Extensão da classe Java.util.Date mais apropriado ao tipo de dados Time da linguagem SQL.

Timestamp

Extensão da classe Java.util.Date mais apropriada ao tipo de dados Timestamp da linguagem SQL

Numeric

Tipo de dado numérico mais apropriada ao tipo de dados Numeric da linguagem SQL que proporciona um conjunto completo de operações aritméticas de grande precisão.

Driver

Classe através da qual o gerenciador do driver (DriverManager) faz a conexão.

Statement

Classe que executa comandos da linguagem SQL simples.

PreparedStatement

Classe especializada na manipulação de comandos da linguagem SQL com parâmetros.

CallableStatement

Classe especializada na manipulação de chamadas a procedimentos armazenados.

ResultSet

Classe para acessar os resultados obtidos pela execução de uma consulta em linguagem SQL.

DataBaseMetaData

Classe usada para acessar uma grande variedade de informações sobre uma conexão a um banco de dados.

ResultSetMetaData

Classe usada para acessar uma grande variedade de informações sobre os atributos de um ResultSet.

Conection

Classe que é responsável pela conexão com o banco de dados.

3.4 A Linguagem de consulta SQL

A SQL foi projetada especificamente para se comunicar com um banco de dados. No início não havia nenhum padrão para consultas. Cada fornecedor criava seus próprios métodos para permitir aos usuários procurar e selecionar registros e campos nas tabelas [DAT-89].

No início dos anos 70 a IBM criou uma linguagem chamada SQL *(Structured Query Language)* que em meados dos anos 80 foi padronizada pela empresa para consultas em seus *mainframes* e minicomputadores. Por exemplo, a sentença apresentada no Quadro 3.1, descreve a seleção de um determinado fornecedor em uma tabela de Produtos onde a quantidade seja menor que 100**.**

```
select * from Tbprodutos
     where
          Quantidade < 100
     and
           Fornecedor = “JORGE DA SILVA”
```

Quadro 3.1 – Exemplo de sentença em SQL .

A linguagem SQL é um padrão amplamente aceito em diversos SGDB, entre eles, Oracle, Informix, Open Ingress, IBM DB2, MySQL e Postgres.

A linguagem SQL proporciona a manipulação de dados utilizando um grupo de sentenças estruturadas, são elas: Select, Update, Insert e Delete.

3.5 A Web

A Web consiste de um conjunto de ferramentas para permitir que um cliente utilizando um reprodutor de documentos HTML (*browser*) possa interagir com um servidor HTTP ligado a Internet.

Uma página HTML nada mais é do um arquivo ASCII contendo o texto da página e os comandos HTML que definem a formatação visual da página e vínculos (*links*) com outras páginas.

A participação desta ferramenta no projeto será de suma importância devido ao fato de que estará associada aos códigos na linguagem JSP, que serão processadas pelo servidor Web antes de serem enviadas ao usuário.

A linguagem HTML é a ferramenta que acompanha todos os *browser* existentes no mercado, exatamente pela simplicidade, facilidade de interação com linguagens, bancos de dados e demais ferramentas. Tornou-se um padrão como interface do usuário.

O protocolo HTTP efetua um serviço TCP/IP orientado a conexão configurando-se principalmente pela transferência de páginas do servidor Web para o reprodutor.

A partir de um pedido HTTP efetuado no reprodutor HTML do cliente, o servidor reage enviando um recurso URL como resposta, geralmente um documento HTML.

Este é convenientemente interpretado pelo reprodutor proporcionando um novo ambiente de interação contendo outros pedidos HTTP.

Este protocolo efetua um serviço TCP/IP orientado a conexão, baseado num mecanismo de transferência de arquivos. O HTTP viabiliza a transferência de páginas WEB do Servidor WEB para o browser.

Ele possui um funcionamento muito simples, desde o momento que um URL é escrito no navegador, o "pedido" que é feito, se for entendido pelo servidor, ele "responde", enviando um documento em hipertexto que é composto por códigos que definem o seu código e ligações a outros documentos, sendo este formato chamado HTML.

Este documento em linguagem HTML é então mostrado ao utilizador, depois de interpretado pelo navegador, sob a forma gráfica.

3.6 Servlets

Os servlets inserem-se no contexto arquitetural cliente/servidor. O cliente solicita algum serviço ao servidor e este realiza a ação correspondente a resposta.

Servlets proporcionam um *framework* básico para manipulação de pedidos HTTP (solicitações *get* e *post* principalmente). Fornecem um conjunto de funcionalidades ligado à solicitação e à resposta a um cliente HTTP, aliviando o programador de detalhes associados ao processamento das solicitações e ao gerenciamento das seções de HTTP. Os servlets proporcionam a geração dinâmica de documentos em linguagem HTML [DEI-01].

Fornecem também um suporte operacional básico ao gerenciamento dos *threads* no servidor. A esse respeito, servlets substituem com vantagens a abordagem convencional para o tratamento de solicitações via processos CGI (*Commom Gateway Interface)*, pois *threads* exigem menos capacidade de processamento da máquina hospedeira do software do servidor.

A plataforma Java fornece dois pacotes associados ao desenvolvimento de servlets: Java.servlets e Java.servlets.http.

Os servlets são executados como parte de um servidor Web. De fato, os servlets são suportados pela maioria dos servidores Web, entre eles, o servidor da Netscape, o servidor

Internet Information Server (IIS) da Microsoft, o Jigsaw do WWW Consorcioum e o Jakarta-Tomcat da Apache.

3.6.1 A interface `java.servlet.Servlet`

Servlets constituem-se como realização da interface Java.servlet.Servlet especificando os seguintes métodos:

`void init (`*`ServletConfig config`*`)`

É invocada uma vez durante um ciclo de execução do servlet para a sua inicialização. O argumento ServletConfig é fornecido pelo servidor que executa o servlet.

`ServletConfig getServletConfig( )`

Esse método retorna uma referência para um objeto que realiza a interface ServletConfig. Esse objeto fornece acesso às informações de configuração do servlet, tais como os parâmetros de inicialização e o ServletContext do servlet, o qual fornece ao servlet acesso ao seu ambiente.

`void Service (ServletRequest Request,`
`ServletResponse Response)`

Define a resposta a uma solicitação de um cliente.

`String getServletInfo( )`

Retorna uma *string*, contendo informações do servlet, tais como o autor e a sua versão.

`void destroy( )`

Esse método é chamado quando um servlet é finalizado pelo servidor em que está sendo executado para liberação de recursos utilizados pelo servlet, tais como um arquivo ou uma conexão a um banco de dados.

3.6.2 A classe java.servlet.http.HttpServlet

A classe mais utilizada para avaliação do servlet interface é a HttpServlet. Sobrescreve o método *service* para distinguir as solicitações HTTP recebidas de um navegador Web. Os dois tipos mais comuns de solicitações HTTP são *GET* e *POST* que funcionam da seguinte maneira:

`Solicitação GET`

Obtém ou busca informações do servidor, geralmente um documento HTML ou uma imagem.

`Solicitação POST`

Solicitam a resposta do servidor a partir de parâmetros fornecido pelo cliente, geralmente informações de um formulário HTML em que o cliente insere dados associados a consultas a um banco de dados do servidor.

A classe **`HttpServlet`** define os métodos **`doGet`** e **`doPost`** para responder as solicitações *GET* e *POST* de um cliente, respectivamente. Esses métodos são chamados pelo método *service* da classe **`HttpServlet`** que é chamado quando uma solicitação chega ao servidor.

Além desses dois, a classe **`HttpServlet`** também define métodos correspondentes aos outros tipos de solicitação HTTP que são: *doDelete, doOptions, doPut e doTrace.*

Os argumentos **`HttpServletRequest`** e **`HttpServletResponse`** permitem a interação entre o cliente e o servidor. Os métodos **`HttpServletRequest`** tornam fácil o acesso aos dados fornecidos como parte da solicitação e os métodos **`HttpServletResponse`** tornam fácil o retorno da resposta ao cliente.

3.6.3 Um problema com o código dos servlets

Uma característica marcante com o código de servlets é o uso constante de um objeto *stream* para o envio do código HTML ao cliente. O Quadro 3.2 apresenta um fragmento de código evidenciando essa característica.

```
...
out.println("<html>");
out.println("<head>");
out.println("<title>"
     +  "Listagem  dos  contribuintes  por  atividades  do
     Município de Ijui
     + </title>");
out.println("</head>");
out.println("<body>");
...
```

Quadro 3.2 – Envio de código HTML ao cliente.

Qualquer alteração no formato da resposta associada ao servlet implica necessariamente em modificação no código do servlet. Isso representa um grande inconveniente, pois geralmente o pessoal ligado à apresentação da aplicação não é muito familiarizado com códigos da linguagem Java. Exemplo, o Web *designer*.

Uma solução amplamente adotada para este problema, é o uso de linguagem de manipulação de informações dinâmicas permitindo a geração de páginas dinâmicas.

3.7 Páginas dinâmicas

Consistem de porções de códigos (*scripts*) que são inseridos diretamente no código das páginas HTML e processadas pelo servidor Web antes de serem enviadas. Têm como objetivo principal, permitir a manipulação de dados dinâmicos gerados pelo servidor sem alteração do conteúdo textual das páginas HTML.

Entre os representantes mais populares desta tecnologia temos ASP da Microsoft; PHP da Apache Software Foundation, que veio para fortalecer a geração de páginas em servidores UNIX já possuindo versões para o LINUX e Windows; o Jscript Server Side da Netscape e a linguagem JSP (*Java Server Pages*) da SUN Microsystems [CON]. Todas elas, originárias ou controladas por empresas ou organismos norte-americanos.

3.7.1 JSP (*Java Server Pages*)

Visão geral

A linguagem JSP é uma tecnologia baseada na linguagem Java que tem por finalidade simplificar o processo de desenvolvimento dinâmico de servidores Web. Ela funciona com um compartilhamento que incorpora elementos dinâmicos, tornando os servidores mais atrativos para os usuários [JSP].

A linguagem JSP funciona como uma linguagem script no lado do servidor. As páginas *JavaServer* são arquivos texto, geralmente com a extensão ".jsp", que substituem os arquivos estáticos HTML. As páginas em código JSP, além de utilizar objetos do servidor, podem incorporar e manipular objetos próprios, como applets e servlets.

Quando a SUN lançou a plataforma Java no mercado, o propósito era o de gerar pequenos programas locais (*stand-alone*) denominados de applets, para prover um maior grau de funcionalidade ao reprodutor HTML do cliente. Entretanto, no ano de 1996, a SUN incorporou os servlets à plataforma Java provendo dinamismo também para o servidor [SUN].

A tecnologia JSP fornece um ambiente de integração de todas essas tecnologias anteriores, proporcionando simplicidade e flexibilidade ao desenvolvimento de software para Web a partir da plataforma Java. Embora a linguagem padrão da JSP seja Java, qualquer outra linguagem *script* pode ser utilizada.

A tecnologia JSP apresenta as seguintes características:

- Portabilidade, e independência de plataforma;
- Acesso a qualquer banco de dados e arquivos textos;
- Interação fácil com o servidor;
- Facilidade de codificação;
- Facilidade de manutenção dos códigos;
- Separação da programação lógica da programação visual, proporcionando que um desenvolvedor e um *designer* possam trabalhar no mesmo projeto de forma independente;
- Utiliza a filosofia de software livre;
- Possui códigos pequenos;
- Possui maior robustez por usar Java como sua linguagem de scripts;

3.7.2 Sobre o código da linguagem JSP

A linguagem JSP intercala a linguagem Java e a linguagem HTML, definindo os limites de cada linguagem no código, tornando organizada e clara a programação [JSP]. Esta é a principal razão porque a tecnologia Java foi escolhida para este projeto.

Existem outras linguagens para geração de páginas *freeware* como exemplo o PHP (Hypertext Preprocessor) que são multiplataforma, são também executados diretamente no Webserver, suporta diversos bancos de dados e uma série de outras características semelhantes à tecnologia Java, mas que possui a programação um pouco mais complexa do que a JSP.

No Quadro 3.3 é mostrado como o código Java é Intercalado com o código em linguagem HTML.

```
...
<%
    Código em Java
%>

    Código em HTML

<%
    Código em Java
%>
...
```

Quadro 3.3 – A estrutura de um código em JSP.

A programação torna-se bastante prática, pois o programador tem uma noção completa do que está acontecendo com o seu código em todas as etapas do desenvolvimento.

Alguns objetos, classes são utilizados no acesso a um banco de dados. São eles:

`Class.forName("org.gjt.mm.mysql.Driver"): Driver`

Define o driver do gerenciador de banco de dados MySQl .

```
DriverManager.getConnection
    "jdbc:mysql://localhost:3306/consultaweb","root","")
          : Connection
```

Define os elementos e a forma de conexão com o banco de dados.

```
Con.createStatement(): Statement
```

É um objeto que executa uma sentença em linguagem SQL simples, através de uma conexão **`con: Connection`** definida previamente.

```
stmt.executeQuery("select * from proficha"): ResultSet
```

Define o resultado de uma consulta em linguagem SQL;

3.8 O Servidor Apache

O servidor Apache é um servidor Web compatível com a padronização HTTP/1.1, sendo o servidor Web mais utilizado. A escolha do Apache em relação outros servidores se deu devido à vasta literatura existente e ao grande número de usuários, o que facilita muito a busca de informações em relação a instalação, configuração e outros aspectos não encontrado em outros servidores. Além da gratuidade, esta é uma das principais razões.

As principais funcionalidades do servidor Apache são:

- Hospeda múltiplos domínios;
- Gera informações sobre o usuário;
- Atua como um servidor *proxy*;
- Opera com múltiplos domínios em diferentes portas;
- Efetua o rastreamento de usuários;

- Inclui inúmeros recursos de segurança, como a possibilidade de restringir acessos seguindo critérios tais como: Domínios, usuários, endereços IP, diretórios e arquivos.

3.8.1 Serviço junto Tom Cat

É um serviço do servidor Apache que permite a execução de aplicações para Web. Sua principal característica é de estar centrado na plataforma Java, mais especificamente sobre as tecnologias Servlet e JSP.

A fundação Apache permite que o Tomcat (subprojeto do projeto Jakarta) seja usado livremente para fins de estudo e comerciais.

3.9 Arquitetura em Três Camadas

Esta arquitetura consiste de um conceito arquitetural de software definido por três camadas. A Fig. 3.2 mostra a estrutura funcional desta arquitetura.

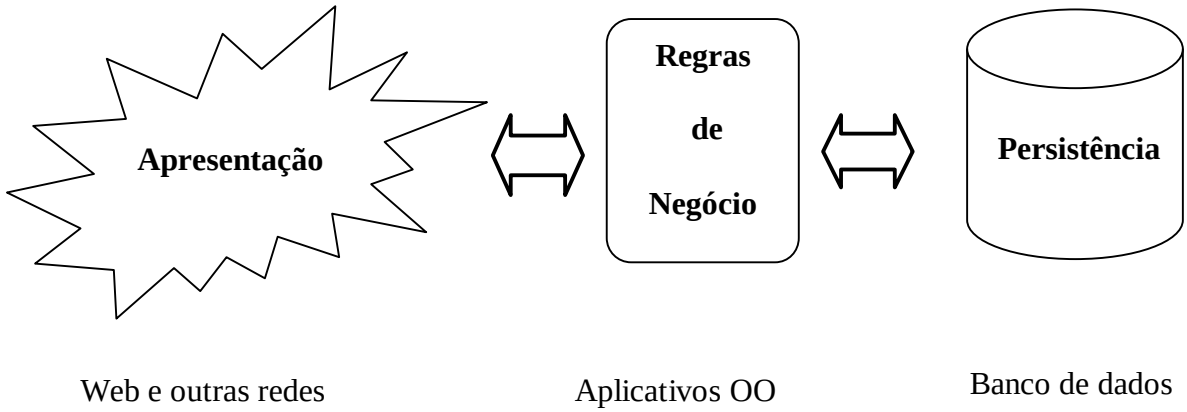

Figura 3.2 – Representação da arquitetura em três camadas.

A arquitetura em três camadas surgiu para facilitar o desenvolvimento e manutenção de sistemas. A camada de usuários disponibiliza uma interface gráfica que no caso de aplicações Web consiste de campos, imagens, botões e de outros componentes.

A camada de aplicativos ou de negócios encarrega-se das regras necessárias para disponibilizar os dados e a camada de dados é onde estão armazenadas todas as informações do sistema.

Entre os muitos aspectos importantes deste tipo de arquitetura, pode-se destacar a *modularização* entre as camadas e a *escalabilidade*.

O aspecto *modularização* torna-se presente quando, por exemplo, novas páginas são inseridas no sistema. Só a camada de apresentação é afetada, as restantes não sofrem nenhuma modificação.

Já a *escalabilidade* refere-se às diferentes dimensões de sistema que essa arquitetura pode ser aplicada. Para o caso das aplicações envolvendo a Internet, esse fator torna-se particularmente interessante, pois o número de usuários é uma variável dificilmente controlável.

3.10 Síntese

As ferramentas apresentadas neste capítulo compreendem o arcabouço tecnológico básico utilizado no desenvolvimento dessa dissertação.

O sistema gerenciador de banco de dados MySQL juntamente com dispositivo JDBC da plataforma Java formam o principal da camada de persistência. A tecnologia Servlets/JSP a principal ferramenta de programação. O Servidor Apache/TomCat a base de inserção do sistema no mundo Web.

Capítulo 4

Modelo de Análise do Novo Sistema de Saúde

4.1 Introdução

O sistema de informação existente na Prefeitura de Ijuí compreende a informatização de diversas atividades da prefeitura. A experiência apresentada neste capítulo considera apenas um grupo de atividades relacionadas à Secretaria de Saúde do município que são as atividades do atendimento médico nos postos de saúde, do serviço de vigilância sanitária e da administração dos cemitérios municipais. Sem perda de expressividade, o modelo de análise proposto nas Seções 4.3 e 4.4 contempla apenas as atividades do atendimento médico. Embora o modelo de análise tenha sido feito para um sistema totalmente novo, ele foi definido seguindo-se uma abordagem de reengenharia fortemente caracterizada pelo re-projeto, portanto, preservando as funcionalidades do sistema existente.

A Seção 4.5 apresenta um modelo de dados legados construído a partir da base de dados do sistema existente. Foi proposto para permitir a implantação gradual do novo sistema a partir de uma integração ortogonal com o sistema existente. Em particular, a implantação do Sistema de Divulgação via Web é apresentado no próximo capítulo.

A próxima seção fornece uma visão geral das atividades desenvolvidas pelo sistema proposto

4.2 Visão Geral

A Prefeitura Municipal de Ijuí possui um sistema de saúde que efetua o atendimento médico a população em geral de acordo com a cobertura do Sistema Único de

Saúde (SUS) através dos diversos postos de saúde espalhados pela cidade. Os usuários solicitam consultas em diversas especialidades médicas, após uma verificação da disponibilidade, é efetivada ou não a consulta ao usuário. Esse sistema compreende também o serviço de vigilância sanitária e a administração dos cemitérios municipais.

O Sistema de Saúde proposto compõe-se dos seguintes subsistemas principais:

Subsistema de Atendimento

Sistema responsável pelo atendimento dos usuários, controlando consultas, exames e as especialidades médicas que os usuários solicitam.

Subsistema de Vigilância Sanitária

Sistema responsável pelo controle sanitário de empresas com sede no município de Ijuí.

Subsistema de Cemitérios

Sistema responsável pelo controle dos falecidos.

Com a implantação do Sistema de Divulgação via Web apresentado no próximo capítulo, os usuários poderão acessar informações associadas a este serviço tais como:

- Profissionais por unidades de serviços e especialidades;
- Totais de fichas retiradas nos atendimentos diários por profissionais e especialidades;
- Movimento do faturamento dos atendimentos ambulatoriais mensais;
- Atendimentos por faixa etária, especialidade e procedimento médico;
- Atendimentos por faixa etária, procedimento médico e tratamento;

- A lista de empresas fiscalizadas pelo serviço de vigilância sanitária;

- Falecidos do cemitério municipal e falecidos do cemitério jardim.

4.3 Diagramas de casos de uso

As Figs. 4.1 e 4.2 mostram os principais casos de uso do Sistema de Saúde proposto. Podemos notar que a escolha de uma especialidade, caso “Escolher uma Especialidade”, é usada apenas para a solicitação de uma consulta, caso “Solicitar uma Consulta”. Por outro lado, a marcação de horário sempre usa a escolha de um médico. Além desses, dois outros casos estão presentes no sistema. É o caso “Autenticar um usuário”, usado pelos casos “Solicitar uma Consulta” e “Alterar uma Consulta”, e “Consultar Banco de Dados” usados por quase todos eles.

Os diagramas de casos de uso dos subsistemas Vigilância Sanitária e Cemitérios foram omitidos desse texto.

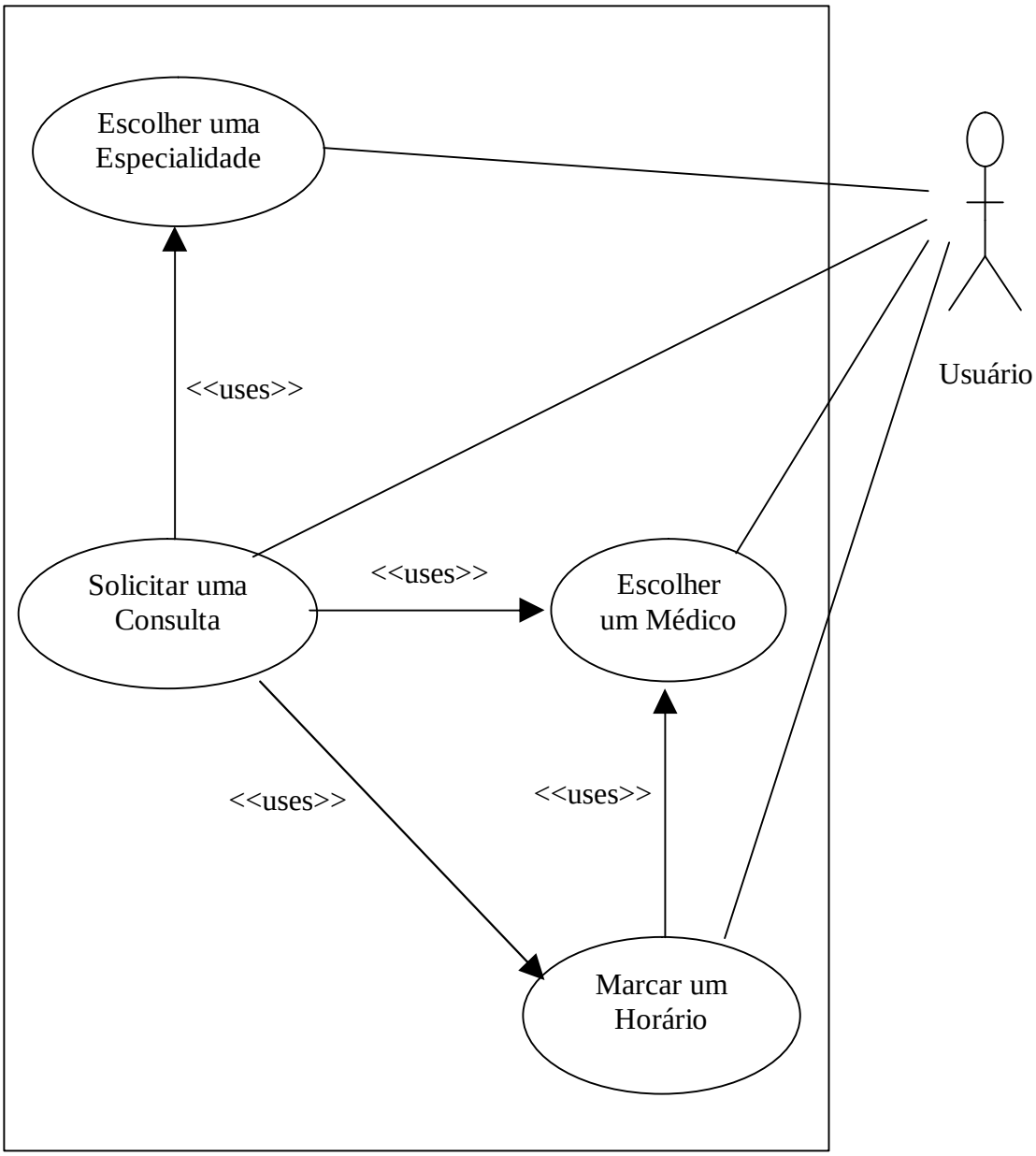

Figura 4.1 – Diagrama de casos de usos para a solicitação de consultas.

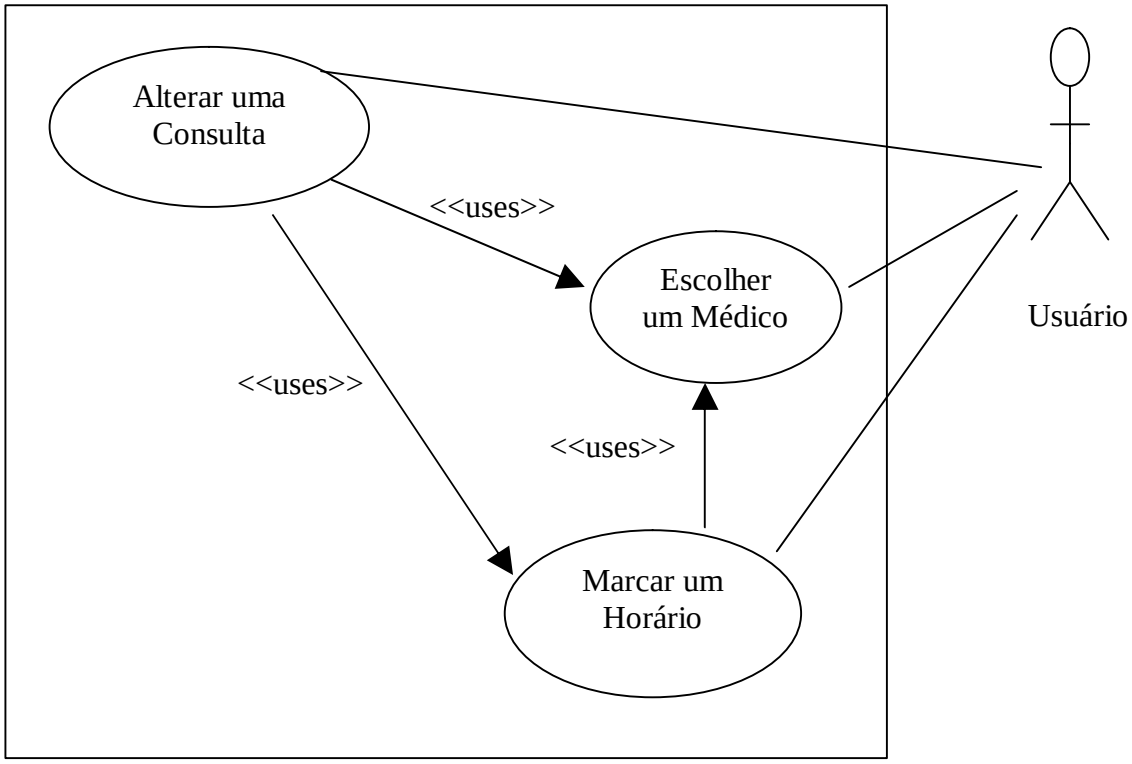

Figura 4.2 – Diagrama de casos de usos para a alteração de consultas.

4.4 Descrição dos casos de usos (cenários)

Autenticar um Usuário

Este caso de uso descreve a autenticação de um usuário no sistema.

Atores

- Usuários

Casos de usos relacionados

- ✍ Divulgar Médicos
- ✍ Solicitar uma Consulta
- ✍ Alterar uma Consulta
- ✍ Consultar Banco de Dados

Cenários primário - fluxos de eventos

1. O usuário fornece o seu código de acesso.
2. O sistema consulta o banco de dados e verifica se o código de acesso corresponde a um usuário registrado.
3. O sistema confirma a autenticação e fim do caso.

Cenários Secundários

1. O código de acesso não corresponde a nenhum usuário registrado.
2. O banco de dados encontra-se inacessível.

Solicitar uma Consulta

Este caso de uso descreve a solicitação de consulta médica por um usuário.

Atores

- ✍ Usuários

Casos de usos relacionados

- ✍ Autenticar um Usuário
- ✍ Escolher uma Especialidade
- ✍ Escolher um Médico
- ✍ Marcar um Horário
- ✍ Consultar Banco de Dados

Cenários primário - fluxos de eventos

1. O usuário solicita uma consulta médica.
2. O sistema autentica o usuário.
3. O usuário escolhe uma especialidade.
4. O usuário escolhe um médico com essa especialidade.
5. O usuário marca um horário com esse médico.
6. O sistema confirma a consulta e fim do caso.

Cenários Secundários

1. Não há profissionais disponíveis na especialidade escolhida pelo cliente.
2. Não há horário disponível para o médico escolhido pelo usuário.
3. O usuário não é autenticado pelo sistema.
4. O banco de dados encontra-se inacessível.

Escolher uma Especialidade

Este caso de uso descreve a escolha de um usuário por uma especialidade médica.

Pré-Condição

O usuário já foi autenticado.

Atores

- Usuários

Casos de usos relacionados

- Consultar Banco de Dados

Cenários primário - fluxos de eventos

1. O sistema consulta o banco de dados e fornece uma lista das especialidades disponíveis.
2. O usuário escolhe uma especialidade e fim do caso.

Cenários Secundários

1. O banco de dados encontra-se inacessível.

Escolher um médico

Este caso de uso descreve a escolha de um usuário por um médico.

Pré-Condição

? O usuário já foi autenticado.

Atores

✍ Usuários

Casos de usos relacionados

✍ Consultar Banco de Dados

Cenários primário - fluxos de eventos

1. O sistema consulta o banco de dados e fornece uma lista dos médicos ao usuário.
2. O usuário escolhe um médico e fim do caso.

Cenários Secundários

1. O médico escolhido encontra-se indisponível (em férias, em licença, etc.).
2. O banco de dados encontra-se inacessível.

Marcar um horário

Este caso de uso descreve a marcação de horário para uma consulta por um usuário.

Pré-Condição

? O usuário já foi autenticado e já escolheu um médico.

Atores

- Usuários

Casos de usos relacionados

- Consultar Banco de Dados

Cenários primário - fluxos de eventos

1. O sistema consulta o banco de dados e fornece uma lista de horários disponíveis para o médico escolhido pelo usuário.

2. O usuário escolhe um horário disponível (eventualmente nenhum).

3. O sistema confirma a marcação de horário e fim do caso.

Cenários Secundários

3. O médico encontra-se indisponível.

4. O banco de dados encontra-se inacessível.

Alterar uma Consulta

Este caso de uso descreve alteração de uma consulta médica (desmarcar, trocar de horário, etc.).

Pré-Condição

? O usuário já foi autenticado.

Atores

- Usuários

Casos de usos relacionados

- Escolher um Médico
- Marcar um horário
- Consultar o Banco de Dados

Cenários primário - fluxos de eventos

1. O Sistema consulta o banco de dados e fornece a lista de médicos para os quais o usuário marcou um horário para consulta.
2. O usuário escolhe um médico dessa lista.
3. O usuário marca um outro horário para o médico escolhido.
4. O sistema confirma a alteração e fim do caso.

Cenários Secundários

1. A consulta expirou.
2. O banco de dados encontra-se inacessível.

4.5 O modelo dos dados legados

O sistema existente foi desenvolvido em linguagem Cobol sobre o ambiente Unix HP-UX. Possui um bom desempenho em relação à segurança, à integridade dos dados e a ocupação de espaço em disco. Entretanto, exibe uma grande inflexibilidade na definição e manipulação dos dados além de não possuir nenhuma facilidade para a sua integração com a Web.

Os dados do sistema são armazenados em arquivos no formato "idx" da linguagem Cobol. Para que esses dados pudessem ser convenientemente examinados foi necessário convertê-los para um formato mais adequado. Para esse propósito, foi desenvolvido um programa em Cobol – **`Psmfatxt`** – para converter os arquivos no formato "idx" para o formato ASCII. O Quadro 4.1 apresenta o resultado para o caso dos dados dos atendimentos ambulatoriais. Os outros casos são tratados similarmente.

Nome dos arquivos (.idx)	**Descrição dos arquivos**
Arqmfi.idx	Movimento físico orçamentário mensal
Arquni.idx	Unidades de serviços
Arqgru.idx	Grupos de atendimento
Arqsgr .idx	Subgrupos de atendimento
Arqmaa.idx	Movimento dos atendimentos
Arqmfa.idx	Movimento do faturamento das unidades
Arqeap .idx	Especialidade dos profissionais da saúde
Arqtia.idx	Tipos de atendimentos
Arqeta.idx	Faixa etária dos pacientes

Quadro 4.1 – Relação de arquivos "idx" para o caso do atendimento ambulatorial.

Cada arquivo "idx" no Quadro 4.1 corresponde a uma tabela de dados que são gerados pelo sistema existente. O programa **`Psmfatxt`** faz uma seleção dos dados que serão convertidos para um único arquivo ASCII – **Psmfatur.txt**. A Fig. 4.4 ilustra toda essa operação.

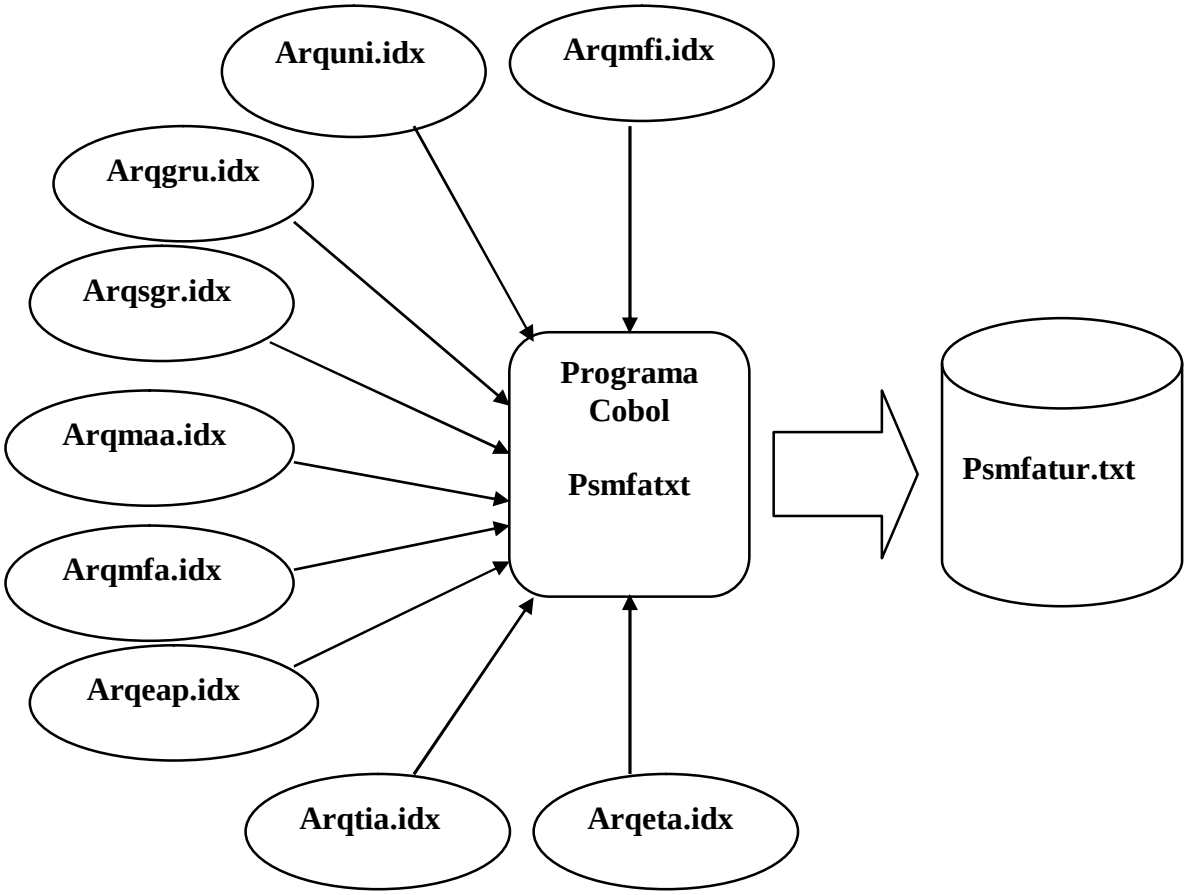

Figura 4.3 - Conversão dos arquivos “idx”.

4.6 Síntese

Foi apresentado um modelo de análise do novo Sistema de Saúde. Esse modelo foi definido seguindo-se uma abordagem de reengenharia com enfoque no re-projeto, isto é, o modelo de análise reflete as funcionalidades do sistema existente.

Também foi apresentado um modelo para a disponibilização de dados legados que consiste na seleção e conversão de dados no formato “idx” da linguagem Cobol para um arquivo ASCII. Estes dados serão posteriormente convertidos para um banco de dados relacional, MySQL, do Sistema de Divulgação via Web apresentado no próximo capítulo.

Capítulo 5

A Implantação do Sistema de Divulgação de Dados via Web

5.1 Introdução

Este capítulo descreve o projeto do sistema de divulgação de dados da Secretaria da Saúde da Prefeitura Municipal de Ijuí. A implantação desse sistema faz parte da estratégia de implantação gradual do Sistema de Saúde apresentado no capítulo precedente e divulgará os dados relativos ao atendimento médico, Subsistema de Atendimento, ao serviço de vigilância sanitária, Subsistema de Vigilância Sanitária, e à administração de cemitérios municipais, Subsistema de Cemitérios.

O Subsistema de Saúde terá como finalidade disponibilizar informações sobre consultas, médicos, tratamentos e demais atividades relacionadas, entre elas:

- Profissionais por unidades de serviços e especialidades;

- Totais de fichas retiradas nos atendimentos diários por profissionais e especialidades;

- Movimento do faturamento dos atendimentos ambulatoriais mensais;

- Atendimentos por faixa etária, especialidade e procedimento médico;

O Subsistema de Vigilância Sanitária terá como finalidade proporcionar informações sobre as empresas (contribuintes), ramos de atividades e demais dados.

Já o Subsistema de Cemitérios disponibilizará informações sobre a localização dos falecidos como quadra, fila e túmulo, parentesco, data de nascimento, data de falecimento e demais dados referentes aos falecidos.

O Sistema de Divulgação foi desenvolvido utilizando a tecnologia JSP/Servlets disponibilizada pela Sun Microsystems e utilizou o gerenciador de banco de dados MySQL disponibilizado pela T.c.X DataKonsultAB.

Todo o hardware utilizado pelo sistema já se encontrava disponível na prefeitura. Não foi necessário adquirir nenhum outro recurso de hardware. Tão pouco foi necessário a aquisição de qualquer software uma vez que foi utilizado apenas software livre (*freeware*).

Os dados que serão divulgados serão obtidos a partir de tabelas geradas na linguagem Cobol em formato texto ASCII padrão que posteriormente serão carregados em tabelas do gerenciador de banco de dados MySQL propiciando assim a divulgação dos mesmos através de páginas HTML na Web. Esse aspecto do sistema será detalhado na seção 5.3.

Qualquer usuário pode acessar o sistema de divulgação que foi desenvolvido através de um computador ligado à Internet. Ao entrar na página principal do sistema, o usuário escolhe qual dos três subsistemas de divulgação deseja obter informações.

5.2 Componentes básicos

O sistema de hardware disponível na Prefeitura de Ijuí compõe-se de dois servidores de aplicação HP modelo 827 e D-230 e de um computador PC ligados por uma rede *Ethernet* padrão.

Todo o sistema administrativo da prefeitura reside nos dois servidores de aplicação. Inclusive as aplicações responsáveis pela gerência dos dados que serão divulgados.

O sistema de divulgação reside no microcomputador PC, incluindo as ferramentas e subsistemas de suporte (servidor Apache/Tomcat, base de dados MySQL, etc.).

O sistema de software que possibilitará a implantação do projeto do sistema de divulgação compõe-se do Sistema Operacional Linux (versão Red-Hat 7.2), o servidor HTTP Apache, o serviço jakarta-Tomcat 4.0.1, o pacote Java (JSP e servlets) e o gerenciador de banco de dados MySQL.

A Fig. 5.1 apresenta a estrutura básica de funcionamento do hardware e do software na prefeitura. Como se pode perceber os usuários através de um microcomputador PC tem condições de acessar o sistema de divulgação que está residente em um microcomputador PC na prefeitura ligado a Web e que estará conectado aos servidores HP.

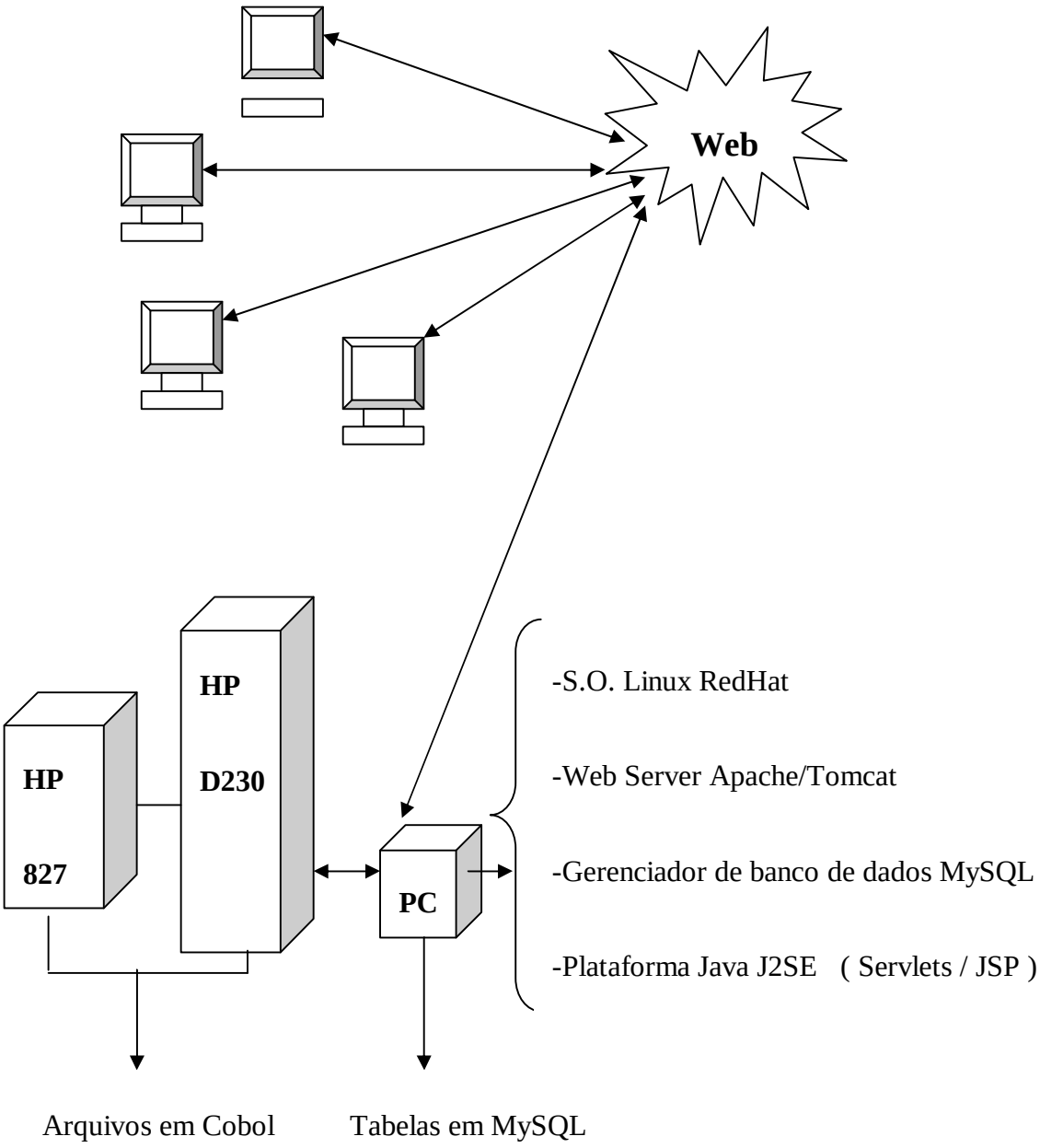

Figura 5.1 - Estrutura básica do sistema de hardware e software.

5.3 Integrando o modelo de dados legados ao sistema

A integração do sistema de divulgação e o sistema existente dar-se-á pela uso de um modelo de dados legados que já foi descrito no capítulo precedente que consiste na conversão periódica dos dados gerados e manipulados pelo sistema já existente em um formato de dados que podem ser utilizados pelo novo sistema.

A geração desse modelo de dados já foi explicada na Seção 4.5. Agora precisamos saber como esse modelo de dados será integrado ao Sistema de Divulgação via Web? Simples, padronizando os dados no formato arquivo ASCII para um formato adequado ao sistema gerenciador de banco de dados do sistema que é o MySQL. É fácil, o próprio MySQL já possui ferramentas para isso.

O Quadro 5.1 apresenta o código dos comandos MySQL para o exemplo da padronização da tabela *proficha* que fornece o total de fichas retiradas nos atendimentos diários por médicos e especialidade. As outras tabelas são padronizadas similarmente.

mysql -D consultaweb -e "delete from proficha" <ENTER>

Exclusão da base antiga .

mysql -D consultaweb -e "LOAD DATA LOCAL INFILE 'proficha.txt' INTO TABLE proficha" <ENTER>

Atualização da nova base.

Quadro 5.1- Atualização das tabelas.

Na banco de dados **consultaweb** estão presente todas as tabelas atualizadas e padronizadas para o sistema gerenciador de banco de dados MySQL. A título de ilustração, a Fig. 5.2 mostra como as tabelas de dados são representadas no arquivo em formato ASCII antes da padronização.

Ressalta-se que para obter o ajuste entre uma tabela Cobol em formato ASCII e uma mesma tabela no gerenciador de banco de dados em MySQL, o primeiro passo a ser feito foi descobrir através de testes qual era o delimitador de campos compatível entre as duas estruturas. No caso, a seqüência de caracteres " \ 9 ".

Jardim	Trajano Heitor Fernandes	1950-03-01	1992-11-05	1992-11-05
Jardim	Regis Diniz Sartori	1965-05-26	1993-04-08	1993-04-08
Jardim	Marli Jardim Sartori	1963-01-07	1993-04-08	1993-04-08
Jardim	Rejane Denise Sartori	1969-10-20	1993-04-08	1993-04-08
....	....	....	....	....

Figura 5.2 – Representação de dados de uma tabela Cobol.

5.4 Exemplo de utilização

Ao entrar no sistema pela Web, o usuário recebe no seu reprodutor html a tela apresentada pela Fig. 5.3 Através dela o usuário pode escolher o tipo de informação que deseja entre saúde, cemitério ou vigilância sanitária. As Figs. 5.4, 5.5 e 5.6 ilustram possíveis resultados.

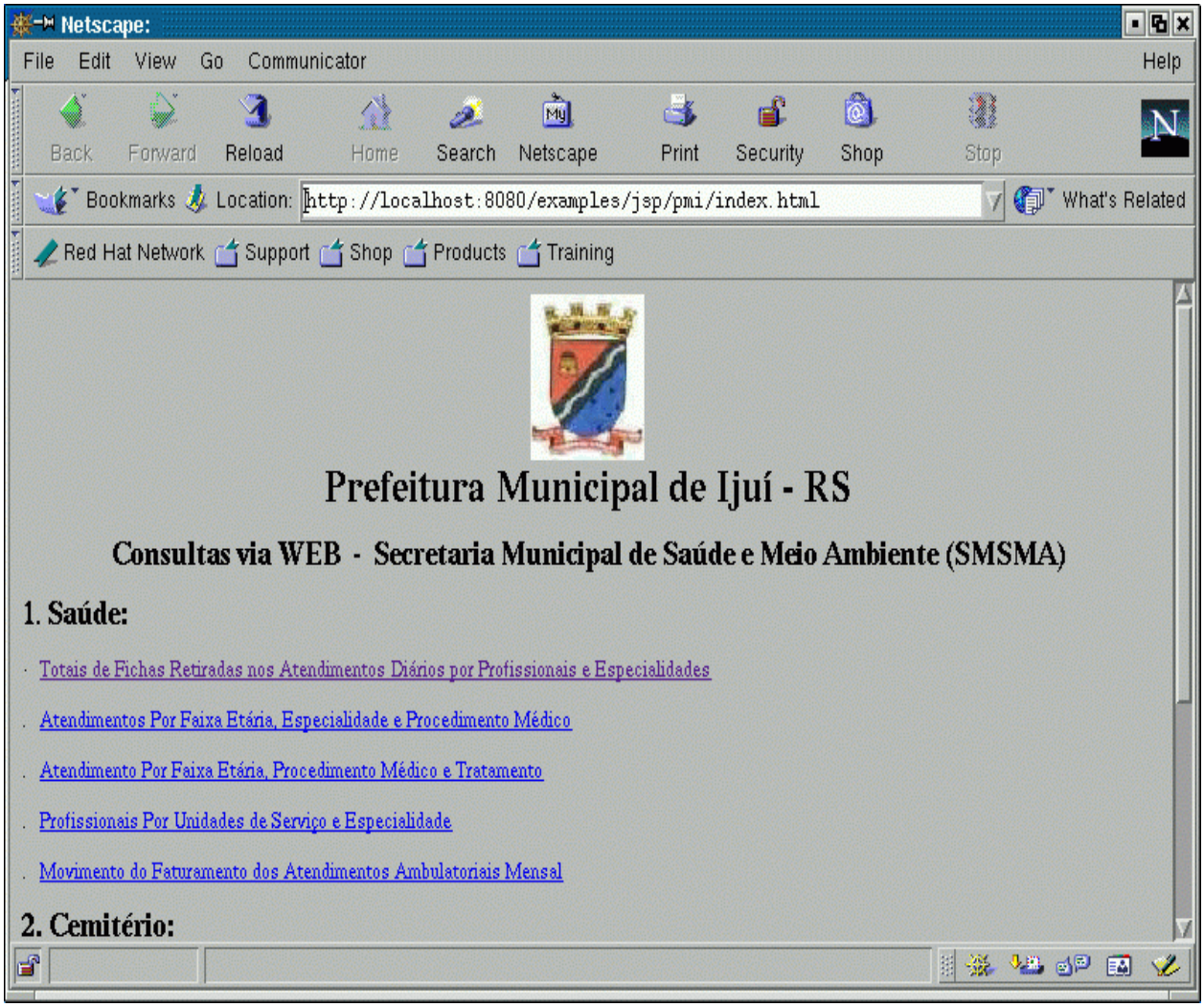

Figura 5.3 - Interface principal.

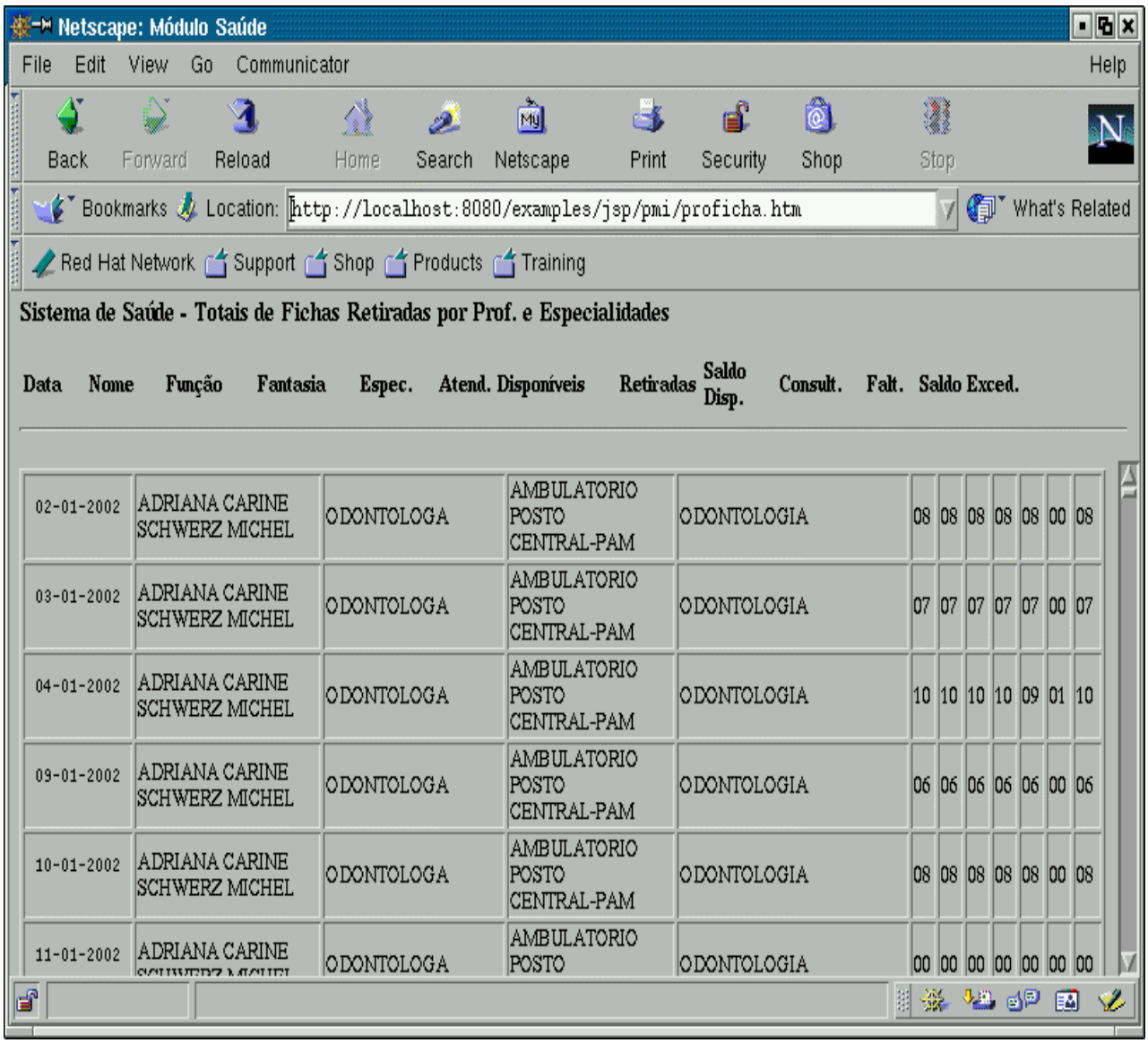

Figura 5.4 - Acesso aos dados da saúde.

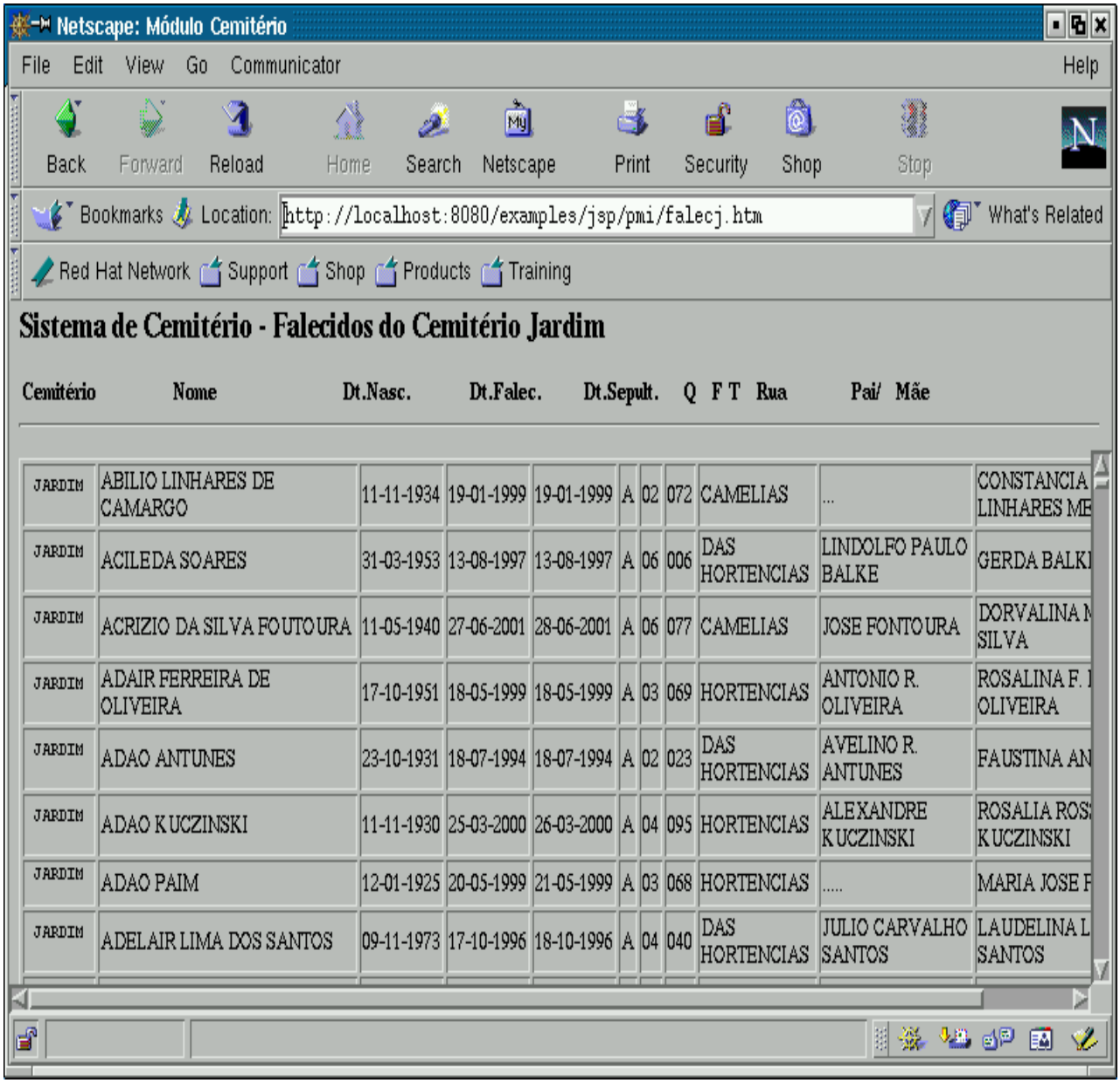

Figura 5.5 - Acesso aos dados dos cemitérios.

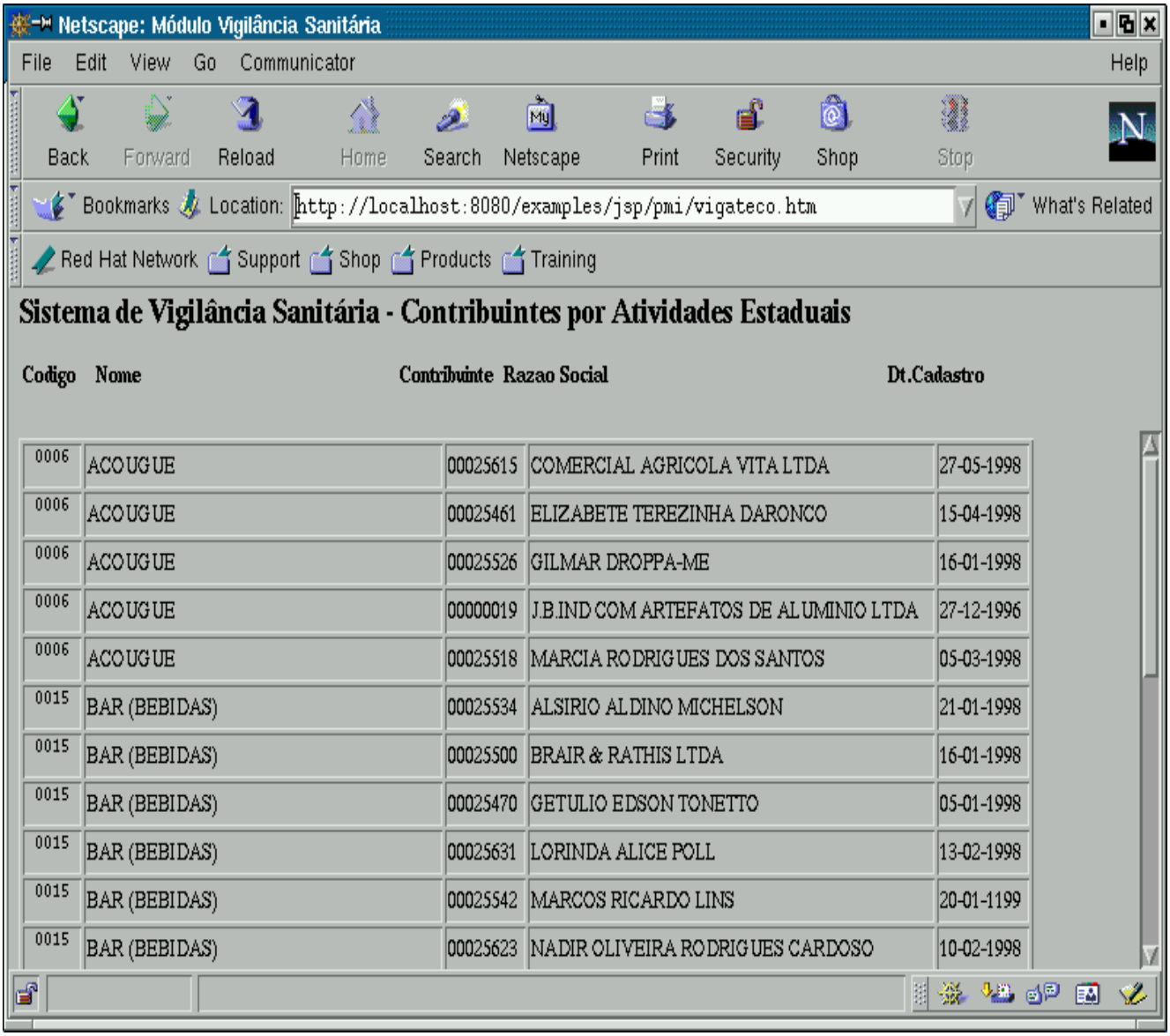

Figura 5.6 – Acesso aos dados da vigilância sanitária.

5.5 Síntese

Este capítulo descreve a estrutura de hardware e de software do sistema de divulgação da prefeitura de Ijuí. Descreve a maneira pelo qual as tabelas do sistema Cobol são integradas ao banco de dados do sistema gerenciador de banco de dados MySQL e apresenta um exemplo de funcionamento do sistema.

Algumas simulações refletindo operações que eram administrados no sistema legado revelaram que o sistema de divulgação via Web oferece respostas temporais adequadas ao contexto de utilização.

Capítulo 6

Conclusão e Perspectivas de Continuação

Esse trabalho de dissertação apresentou uma experiência de desenvolvimento de sistemas de informação. Evidenciou-se o uso de software livre como uma solução de desenvolvimento a custo zero. Avaliações práticas do resultado obtido através dessa experiência mostrou que essa abordagem é bastante viável e conveniente ao contexto de pequenas instituições públicas.

Também foi estudado um problema de reengenharia que contempla dois aspectos do desenvolvimento de novos sistemas, a saber: a implantação gradual e a integração ortogonal. Para esta experiência, o sistema de divulgação, a abordagem proposta mostrou-se bastante favorável às especificidades do ambiente que é encontrado em instituições públicas de pequeno porte.

O modelo de análise apresentado no Capítulo 4 compreende apenas às atividades da Secretaria da Saúde. Também, a própria divulgação contempla apenas aos dados de um pequeno grupo de atividades. Uma continuação natural desse trabalho é a extensão desse modelo de análise visando à incorporação de um número maior de funcionalidades e a implantação de novas partes do novo sistema.

O mecanismo de integração que foi utilizado é bastante simples. Trata-se da conversão periódica dos dados gerados e manipulada pelo sistema já existente em um formato de dados que podem ser utilizados pelo novo sistema. No caso do sistema de divulgação que foi implantado, a conversão diária mostrou-se bastante adequada porque a validade temporal dos dados é condizente com a sua utilização. A implantação de novas partes do novo sistema pode, entretanto exigir uma periodicidade menor. Neste caso, esse

mecanismo poderá não ser mais adequado. Ao mesmo tempo, quando consideremos ambos os sistemas, o existente e o que está sendo implantado, um outro problema vem à tona: a necessidade da duplicidade dos dados que exige uma capacidade de armazenamento duas vezes maior que a necessária para cada um dos sistemas individuais. O tratamento desses problemas é deixado como perspectiva de continuação desse trabalho de dissertação.

Bibliografia

[BOO-99] BOOCH, G. Rumbaugh J., Jacobson I. , The unifield modeling language user guide, Addison Wesley Reading, 1999.

[COA-93] COAD, P., Projeto baseado em objetos. Quarta Edição.Editora Campus, 1993.

[CHI-90] Chikofsky, E.J. e J. H. Cross, III, Reverse Engineering and Design Revery: A Taxonomy, IEEE Software, 1990.

[DAT-89] Date, C.J., Guia Para o Padrão SQL, Ed. Campus, 1989.

[DEI-01] Deitel, H.M., Deitel P.J., Java Como Programar. Bookman. Porto Alegre – Brasil, 2001.

[FOW-00] M. Fowler, Scott K., UML Essencial, Bookman, 2ª Edição, 2000.

[GIO-97] Fernando, A.C., Tecnologia de Orientação a Objetos, U. Mackenzie, 1997.

[LEW-00] LEWIS, J., Loftus W., Java software solutions : foundations of program design. Second Edition. Adison – Wesley, 2000.

[LAR-00] Larmann C, Utilizando UML e Padrões: uma introdução à análise e ao projeto orientado a objetos, Porto Alegre, Ed. Bookman 2000.

[OMG-99] OBJECT MANAGEMENT GROUP, UML specification version 1.3, junho, 1999.

[PET-01] Peters F. J., Pedrycz W., Engenharia de Software, Editora Campus, 2001.

[PRE-95] Pressman, Roger S., Engenharia de Software. Makron Books. São Paulo – Brasil, 1995.

[SIL-99] H.F, Silberschartz A., Sudarshan S, Sistema de Banco de Dados, Makron Books, 1999.

[JSP] Características da tecnologia JSP, http://Java.sun.com/produts/jsp/ em Janeiro de 2002.

[SUN] Incorporação de servlets pela plataforma Java, www.sun.org.com.br em Dezembro de 2001.

[J2SE] Plataforma de desenvolvimento Java, www.Sun.org.com.br em Janeiro de 2002.

[OMG] Notação UML, www.omg.com em Março de 2002.

[CON] Sistema Operacional Linux, www.conectiva.com.br em Dezembro de 2001.