UNIVERSIDADE CATÓLICA DE PELOTAS
ESCOLA DE INFORMÁTICA
CURSO DE ANÁLISE DE SISTEMAS

Sistema de Gestão via Web para Pomar de Pessegueiros, baseado em Software Livre

Por
ANDRÉ RIBEIRO CAMARGO

Projeto de Graduação submetido como requisito parcial à obtenção do grau de Bacharel em Análise de Sistemas.

Orientador: Prof. Ramiro Saldaña
Co-Orientador: Prof. Guilherme Netto

Pelotas, Dezembro de 2000

Dedico este trabalho a meus pais Adelci e Solange, por sempre acreditarem na minha capacidade e me disponibilizarem todo o possível, a minha irmã Carla e meus irmãos Mateus e Adelci Júnior, por todo apoio e paciência e finalmente, a minha namorada Aline, por todo suporte extra familiar oferecido gratuitamente, demonstrando ser uma ótima companheira. Eu amo todos vocês.

III

“Porque nos inquietamos?
...
Até onde a nossa vista alcança?
...
Qual é a origem dos pensamentos?
Podemos pensar no que não nos interessa?
Porque nos dói a cabeça?
Porque não estamos nunca satisfeitos?
...
O cérebro precisa de alimento?
O cérebro do talentoso é maior que o do imbecil?
...
Já se descobriu o mundo todo?
...
Convém ter sempre alguma coisa que fazer?
Porque nos esquecemos de umas coisas e lembramos de outras?
Porque não temos tudo o que precisamos?
O que é uma garrafa térmica?
...
Onde se escondem as moscas no inverno?
Poderíamos ver se não tivéssemos cérebro?”

O livro dos Porquês, faixa 7 do álbum “À Beça” do escritor, cantor e compositor pelotense Vitor Ramil

AGRADECIMENTOS

Ao meu orientador, Prof. Ramiro Saldaña, pelos ensinamentos, pela atenção, confiança e por todo o apoio que sempre me deu.
Ao meu co-orientador, Prof. Guilherme Netto, pelo incentivo.
Aos professores da Escola de Informática (ESIN) da Universidade Católica de Pelotas (UCPel), principalmente ao Kratz, Clara, Regina, Lúcia, CAC, Meirelles, André Freitas, André Caruso, Sérgio Santi, Cláudio Gastal e Daniel Lichtnow, pela orientação oferecida no decorrer do meu curso.
Ao Eng. Agrônomo Vilson Helbig, pela revisão e sugestões na área agrícola.
Ao criador da OOHDM, Prof. Gustavo Rossi, pelo brilhante método e pela atenção dispensada comigo.
A Daniel Schwabe, Natacha Güell, Patricia Vilain e Mark Douglas, pela ajuda oferecida no aprendizado da OOHDM.
Aos meus cyberfriends, Carlos Villela (leecho) e Ricardo Sugawara Júnior (Sugawara), pela revisão da monografia e pelas dicas sempre bem-vindas.
Aos meus colegas Sandro Fernandes, Daniel Brahm, Rodrigo Gruppelli e Cristiano Amadori pelos momentos extrovertidos durante esses 5 anos de faculdade.
Aos meus amigos de Canguçu, pela compreensão e por toda a atenção e amizade.
Finalmente a toda comunidade de software livre pelo belo trabalho que tem desenvolvido.

SUMÁRIO

LISTA DE FIGURAS

RESUMO

Este trabalho apresenta a modelagem de um sistema de gestão para pomares de pessegueiros, para rodar via WWW, baseado inteiramente em software livre. Para isso, foi feito um estudo sobre software livre e o estado da arte em ferramentas distribuídas dessa forma. Foi utilizado como objeto de estudo o Pomar de Pessegueiros Arroio das Pedras, situado em Canguçu-RS. A modelagem da aplicação foi feita utilizando-se o método para projeto de aplicações hipermídia OOHDM. Também foi implementado um protótipo a fim de assegurar que esta especificação é passível de construção.

Este estudo tem como meta contribuir com a comunidade de software livre e fornecer meios para o desenvolvimento regional, visto que nossa região é grande produtora de pêssegos e não há muitas ferramentas que apóiem os agricultores na gestão de seu pomar.

Palavras-chave: sistema de gestão, pessegueiro, WWW, software livre, OOHDM

ABSTRACT

This work presents the modeling of a web peach tree orchard management system, based totally in free software. To do this work, it was made a study about free software and the state of the art in free software tools. Our case study was the Arroio das Pedras peach tree orchard, situated in Canguçu-RS-Brazil. In the application modeling was used the Object-Oriented Hypermedia Design Method (OOHDM). Also, it was implemented a prototype to assert that this specification can to be built.

This study aims to contribute with free software community and to provide ways to regional growth, inasmuch as our region is a big peach producer and there aren't many tools that aid the farmers in the management of your orchard.

Keywords: management system, peach tree, WWW, free software, OOHDM

1 INTRODUÇÃO

Quando estudei geografia no primeiro grau (hoje ensino fundamental), a professora ensinou-me que a região sul do Brasil (constituída pelos estados do Paraná, Santa Catarina e Rio Grande do Sul) era considerada como a "região-celeiro" deste país, devido a sua grande produção no setor primário (agropecuária). Com o passar do tempo, minha vocação não despertou para o lado agropecuário, tomando o rumo da informática. Mesmo assim, não esqueci-me disso e tendo vários familiares que tiram seu sustento da terra, percebi um mercado pouco desenvolvido dentro da minha área: o mercado da informática aplicada ao setor agropecuário.

O Sr. Adelci Camargo, que é meu pai, é proprietário de um pomar de pessegueiros no município de Canguçu-RS. Já que atualmente nada está informatizado, toda a gestão do pomar é feita manualmente, utilizando-se um caderno. Como o pomar está em fase de expansão, ele está procurando novas formas para gerenciamento de custos.

Devido ao nosso estreito relacionamento, ele veio a mim e pediu para que eu desenvolvesse um sistema de gestão para o pomar. A necessidade dele caiu como uma luva no que eu estava procurando desenvolver como projeto. Propus a ele a idéia de desenvolver um sistema de gestão para pomares de pessegueiros como meu projeto de graduação e ele aceitou o desafio de tornar-se minha principal "cobaia". Como nossa região é produtora de pêssegos e meu pai tem um ótimo relacionamento com outros produtores, a idéia inicial foi ampliada para que desenvolvêssemos um sistema baseado também nas necessidades de outros pomares.

O problema foi definir uma maneira de disponibilizar facilmente este sistema de forma que atingisse o maior número de produtores, independente da localização do proprietário do pomar. Foi aí que surgiu a idéia de hospedar a aplicação na Internet, para ser acessada através da web.

Atualmente, com a popularização da Internet, todas as atenções estão voltadas para o comércio eletrônico. Isto me despertou um interesse muito grande por esta área. Além disso, também sou um apaixonado pelo movimento de software livre e como existem ótimas ferramentas para web disponibilizadas livremente, proponho desenvolver esta aplicação utilizando somente "free software"e, é claro, licenciar o aplicativo através da licença de software livre GNU General Public License (GNU GPL), uma cópia traduzida para o português da licença é apresentada no ANEXO A. Convém lembrar, que essa

forma de distribuição também beneficiaria produtores de pequeno porte, já que soluções baseadas em software livre possuem um custo reduzido quando comparado a soluções proprietárias.

1.1 Objetivos e contribuições

O objetivo deste trabalho é modelar um sistema de gestão para pomares de pessegueiros de forma que possa ser acessado via web. Feito isso, estudar a viabilidade de implementação deste modelo utilizando somente software livre. E disponibilizar o sistema na web compartilhando o conhecimento adquirido com a comunidade, além de, contribuir com o desenvolvimento da nossa região sul do estado do Rio Grande do Sul.

Este trabalho tem como objetivos específicos:

- Estudar ferramentas livres de desenvolvimento para web;
- Estudar e escolher um método para desenvolvimento de software;
- Modelar um sistema de gestão para pomares de pessegueiros;
- Implementar, um protótipo do sistema modelado;
- Criar uma classe de documento LaTeX[1] para monografias da Escola de Informática (ESIN) da Universidade Católica de Pelotas (UCPel);
- Disponibilizar conhecimento adquirido para a comunidade;
- Contribuir para o desenvolvimento da região, já que nossa região é grande produtora de pêssegos;
- Licenciar o sistema de gestão através da GNU GPL;

Em minha universidade, todos os projetos de graduação devem ter necessariamente três tipos de contribuições básicas, a saber:

- Contribuições para o ensino: os resultados deste projeto poderão ser utilizados para ensino de graduação em diversas disciplinas do curso relacionadas a engenharia de software, principalmente modelagem de sistemas, e redes de computadores no que tange a web;

[1] **LaTeX** é um conjunto de macros na linguagem tipográfica TEX que facilitam a escritura de documentos científicos.

- Contribuições para pesquisa: não encontrei nenhum outro software que desempenhe o que este projeto está se propondo, tornando-se assim uma aplicação ímpar e desafiante. Até mesmo pela tecnologia e pelas ferramentas que utilizei. Contribuindo também com subsídios para futuros estudos sobre ferramentas livres.

- Contribuições para extensão: Auxiliar produtores de pêssego na gestão do seu negócio, melhorando a produção principalmente através de um melhor controle de custos para alcançar melhor competividade no mercado e possibilitando o acesso a tudo que a Internet oferece (comunicação com outras pessoas, comércio eletrônico, ensino a distância, etc), conseqüentemente contribuindo com o desenvolvimento da nossa região.

1.2 Organização do trabalho

O capítulo 2 desta monografia apresenta a filosofia GNU, o movimento de software livre e o cuidado que se deve ter com a licença sob a qual o software que utilizamos é distribuído.

No capítulo 3, focalizarei o estado da arte em software livre em algumas áreas específicas da informática.

Feito o estudo das ferramentas, o capítulo 4 trata sobre o modelo de desenvolvimento orientado a objetos. Abordarei o processo de desenvolvimento de software, algumas metodologias orientada a objetos e apresentarei o tipo de aplicação que este trabalho propõe construir.

O capítulo 5 apresenta um estudo de caso sobre o pomar de pessegueiros “Arroio das Pedras”, focalizando desde a história do pessegueiro até os processos desenvolvidos no cultivo agrícola.

No capítulo 6 começa a formalização do processo de desenvolvimento de software com a realização da fase de elaboração, onde é feito a modelagem em caso de usos da aplicação e o levantamento dos riscos de projeto.

O capítulo 7 é o resultado da fase de projeto. Nesta fase utilizei a OOHDM como método de apoio para a construção da aplicação de gestão para pomar de pessegueiros. Neste capítulo é feito um estudo sobre o método de projeto utilizado, bem como, a especificação da aplicação através da utilização do método.

Por fim, o capítulo 8 apresenta a conclusão do trabalho, o capítulo 9 contém os anexos e o capítulo 10 as referências bibliográficas.

2 SOFTWARE LIVRE

2.1 Introdução

> "**livre** adj. 1. Que não está sujeito a algum senhor. 2. Que não está, ou já não está, prisioneiro; solto. 3. Desprendido, solto. 4. Independente. 5. Que goza dos seus direitos civis e políticos. 6. Cujo funcionamento sem coerção ou discriminação é garantido por lei. 7. Permitido. 11. Desimpedido. 14. Sem limites, imenso." [Fer 93]
>
> "Se os usuários de computadores pudessem modificar programas para os ajustar às suas necessidades, e liberdade para compartilhar estes programas, acredito que o avanço da informática não seria travado por empresas que assumem o lucro como principal meta, não importando se com isso elitizam o acesso à informação.
>
> A idéia de se compartilhar software nasceu junto com os computadores, da mesma forma que compartilhar receitas é tão antigo como cozinhar. Não é um devaneio socialista. É a constatação de que o desenvolvimento da tecnologia da informação é fruto do acúmulo do conhecimento da humanidade e, portanto, não passível de patenteamento."[Gas 00]

O termo **free software** (em inglês free = livre ou grátis) às vezes é mal interpretado — não tem nada a ver com o preço. E sim com liberdade. A Free Software Foundation define que um software é livre, quando sua licença contempla as quatro condições abaixo:

1. Você tem liberdade para executar o programa, com qualquer propósito;
2. Você tem a liberdade para modificar o programa e adaptá-lo às suas necessidades. (Para fazer esta liberdade ser efetiva e prática, você deve ter acesso ao código fonte, porque modificar um programa sem ter a fonte de código é excessivamente difícil.)
3. Você tem liberdade para redistribuir cópias, tanto grátis como com taxa.
4. Você tem a liberdade para distribuir versões modificadas do programa, de tal modo que a comunidade possa beneficiar-se com as suas melhorias.

Como "free" (livre) refere-se a "freedom" (liberdade) e não a preço, não existe contradição entre a venda de cópias e o software livre. De fato, a liberdade para vender cópias é crucial para obtenção de fundos para o desenvolvimento de software livre.

Se um programa é software livre quando deixa as mãos de seu autor, isto não significa que será software livre para qualquer um que tenha uma cópia dele. Por exemplo, o software de domínio público (software que não está sujeito aos direitos autorais de qualquer pessoa) é software livre; mas qualquer um pode fazer uma versão modificada proprietária dele. Igualmente, são registrados muitos programas livres mas distribuídos por meio de licenças simples que permitem versões modificadas proprietárias.

Por causa da ambigüidade da palavra "free", as pessoas a muito têm procurado uma alternativa, mas ninguém encontrou um outro termo apropriado. O idioma inglês tem mais palavras e nuances que qualquer outro, mas falta uma palavra simples, não ambígua, palavra que signifique "free" (livre), como em "freedom" (liberdade) - "unfettered" (sem correntes) e a palavra que mais entra no íntimo significando. Outras alternativas como "liberated" (liberado), "freedom" (liberdade) e "open" (aberto) têm o significado errado ou alguma outra desvantagem.

2.2 A opinião da Free Software Foundation sobre licenças livres

A Free Software Foundation (FSF) é uma organização dedicada a eliminar restrições na cópia, redistribuição, entendimento e modificações de programas de computador. Isso é feito através da promoção do desenvolvimento e uso de software livre em todas as áreas da computação - particularmente, ajudando no desenvolvimento do Sistema Operacional GNU.

Conforme a FSF, há duas categorias de licenças de software, que são: **copyleft** e **não copyleft**. Licenças copyleft são como a GNU GPL, insistem que versões modificadas de um programa devem continuar sendo livres. Licenças não copyleft não fazem questão disso.

A FSF classifica as licenças livres de acordo com alguns princípios básicos, que são:

- Se a licença se considera uma licença livre;
- Se a licença é copyleft;
- Se a licença é compatível com a GNU GPL (Isto significa que você pode combinar um módulo que foi lançado sobre esta licença com um módulo licenciado pela GPL para construir um programa maior);

- Se a licença causa qualquer problema ao utilizá-la;

Segue uma lista de observações sobre licença, conforme [GNU 00]:

- A Licença GNU General Public License, ou abreviadamente, GNU GPL
  Esta licença é livre e copyleft. A FSF a recomenda para a maioria dos pacotes de software.

- A Licença GNU Lesser General Public License, ou abreviadamente, GNU LGPL
  Esta licença é livre, mas não copyleft, porque permite ligar seus módulos a software não livre. É compatível com a GNU GPL. É recomendada para algumas circunstâncias especiais.

- A Licença do Guile
  Esta licença consiste da GNU GPL, mais uma declaração especial permitindo ligar com software não livre. A FSF recomenda sua utilização apenas em algumas circunstâncias — praticamente as mesmas que deveriam ser consideradas ao utilizar a GNU LGPL.

- A Licença das Unidades de Run-Time do compilador GNU Ada
  Idem a licença do Guile.

- A Licença X11
  Esta é uma licença simples, de software livre não copyleft permissiva, compatível com GNU GPL. O XFree86[2] usa essa licença.

- A Licença original BSD
  Esta licença é simples, não copyleft, com várias falhas. A "cláusula de advertência BSD" não chega a ser um grande problema; que seria, a não submissão a software não-livre. Isso causa problemas na prática dessa licença, sendo incompatível com a GNU GPL.

- A Licença BSD modificada
  Esta é a licença BSD original, modificada para remover a cláusula de advertência. É simples, não copyleft e sem problemas particulares. É compatível com a GNU GPL.

- A Licença do Apache
  Simples, não permite copyleft, mas possui os mesmos problemas da licença BSD para praticá-la, sendo incompatível com a GNU GPL.

[2] **XFree86** é o nome de um projeto que desenvolve um ambiente gráfico para sistemas operacionais Unix.

- A Licença do Zope
  A licença pela qual o Zope é distribuído é simples, não permite copyleft e possui os mesmos problemas de aplicação da licença BSD, sendo incompatível com a GNU GPL.

- A Licença Pública IBM
  Esta é uma licença livre, mas incompatível com a GNU GPL.

- A Licença Pública do Projeto LaTeX
  Esta licença é uma declaração incompleta dos termos de distribuição do LaTeX. Enquanto pode, ainda é uma licença livre, mas incompatível com a GNU GPL porque possui várias exigências que não estão na GPL.

- A Licença do Perl
  Esta licença é a disjunção da licença Artística e da GNU GPL. Ela é qualificada como licença livre, mas pode não ser uma copyleft verdadeira. Ela é compatível com a GNU GPL, porque a GNU GPL é uma das suas alternativas.

- A Mozilla Public License (MPL)
  Esta é uma licença livre que não possui um copyleft forte, diferente da licença X11, ela possui algumas restrições complexas que a tornam incompatível com a GNU GPL. Por exemplo, um módulo GPL não pode legalmente ser ligado a um módulo MPL.

- A licença Open Source Netizen (NOSL), Versão 1.0
  Esta licença é essencialmente a mesma MPL, versão 1.1. Assim como a MPL, a licença NOSL possui as mesmas restrições complexas que a tornam incompatível com a GNU GPL.

- A Licença Pública Sun
  Esta licença é essencialmente a mesma MPL: uma licença de software livre incompatível com GNU GPL. Não confunda esta licença com a Sun Community Source License que não é uma licença livre.

- A Netscape Public License (NPL)
  Esta licença é livre, não copyleft, e incompatível com a GNU GPL. Ela consiste da Mozilla Public License com uma cláusula adicional que permite a Netscape usar o código que você adicionou até em versões proprietárias do programa. É claro, eles não dão a você permissão de usar o código deles.

- A Licença do Javascript Netscape
  Esta licença é uma junção da NPL e da GNU GPL. Devido a isso, ela é uma

licença livre, compatível com GNU GPL, mas não copyleft. Esta licença é uma boa escolha se você deseja que seu software seja compatível com GPL e MPL. De qualquer maneira, você também pode executá-la usando as licenças LGPL ou Guile.

- A PHP License, versão 2.02
  Esta é a licença usada pelo PHP4, não é copyleft e possui alguns problemas praticais, como aqueles da licença BSD Original, sendo incompatível com a GNU GPL.

- A QT Public License (QPL)
  Esta é uma licença livre não copyleft que é incompatível com a GNU GPL. Esta licença também tem um inconveniente ao praticá-la, já que código fonte modificado deve ser somente distribuído como patch[3].
  Desde que QPL é incompatível com a GNU GPL, não é possível ligar um programa licenciado pela GPL com outro programa licenciado com QT.

- A Licença da Biblioteca Padrão de Funções da iMatix
  Esta é uma licença livre e compatível com GPL.

- A Licença da ZLib
  Esta é uma licença livre, compatível com GPL.

- A Licença FreeType
  A licença FreeType é uma licença livre não copyleft, que é incompatível com a licença GPL por razões técnicas. Normalmente seu uso se restringe a fontes tipográficas, e neste tipo de uso, a incompatibilidade não causa problemas.

2.3 O Projeto GNU

Baseado nas informações contidas em [GNU 00] e [Gas 00] podemos dizer que em janeiro de 1984, Richard M. Stallman deixou seu trabalho no Massachusetts Institute of Technology (MIT) e deu início ao Projeto GNU. A meta do Projeto GNU é criar um sistema operacional livre, já que sem isso, não é possível nem fazer funcionar um computador. Deixar seu trabalho foi necessário para que o MIT não pudesse interferir na distribuição do software GNU. Se ele tivesse continuado como parte da equipe, o MIT poderia ter reivindicado propriedade no trabalho, e poderia ter imposto as próprias condições de distribuição, ou até mesmo poderia transformar o trabalho em um pacote de software proprietário. Não era intenção fazer um trabalho enorme somente para vê-lo tornar-se inútil

[3] **patch** como termo técnico de informática, não foge muito a definição literal de “remendo”, podendo ser entendido também como “atualização”

a seu almejado propósito: criar uma nova comunidade de software compartilhado. Ele sabia que com um sistema operacional livre, poderia haver uma comunidade de hackers[4] cooperando.

Como desenvolvedor de sistema operacional, Richard M. Stallman tinha as habilidades apropriadas para esta tarefa. Então decidiu fazer com que o sistema fosse compatível com Unix porque deste modo seria portável, e assim aqueles usuários de Unix poderiam migrar para ele com facilidade. O nome GNU foi escolhido seguindo uma tradição hacker, como acrônimo recursivo para "Gnu is Not Unix" (GNU não é Unix).

O objetivo do Projeto GNU é dar liberdade aos usuários. Em função disso procurou-se criar condições de distribuição que prevenissem que o software GNU pudesse vir a se tornar proprietário.

O método utilizado foi denominado **"copyleft"**. O copyleft usa a lei protegida por direitos autorais, mas dá a volta para servir ao oposto de seu propósito habitual: em vez de ser um meio de privatizar o software, se torna um meio de manter livre o software. A idéia central do copyleft é dar a qualquer um permissão para executar o programa, copiar o programa, modificar o programa e redistribuir versões modificadas - sem permitir que o usuário adicione restrições de sua propriedade. Deste modo, as liberdades cruciais que definem o "software livre" são garantidas a qualquer um que tenha uma cópia; elas tornam-se direitos inalienáveis.

Para um copyleft efetivo, as versões modificadas também devem ser livres. Isto assegura que todo o trabalho baseado em software livre fica disponível para nossa comunidade se é publicado. Quando os desenvolvedores trabalham como programadores voluntários para melhorar o software GNU, é o copyleft que impede que os empregadores digam: "não pode compartilhar essas mudanças, porque nós queremos usá-las para fazer nossa versão proprietária do programa".

A exigência de que as mudanças devem ser livres é essencial se nós quisermos assegurar a liberdade para cada usuário do programa.

Um tópico relacionado trata a combinação de um programa livre com um de código não livre. Tal combinação será inevitavelmente não livre; qualquer liberdade que perdeu a parte não livre, também perderá o todo. Permitir tais combinações abriria um

[4] O uso de **hacker** para se referir ao "violador de segurança" é uma confusão que vem por parte dos meios de comunicação de massa, sendo que a verdadeira definição seria "alguém que ama programar e que gosta de ser hábil e engenhoso".

buraco grande o suficiente para afundar um navio. Para isto, é uma exigência crucial ao copyleft tapar este buraco: qualquer coisa somada ou combinada com um programa copyleft deve ser de tal forma que a versão total combinada também seja livre e copyleft.

A implementação específica de copyleft que é usada na maioria do software GNU é o "GNU General Public License" (Licença Pública Geral GNU) ou GNU GPL para abreviar. Há outros tipos de copyleft que são usados em circunstâncias específicas. Manuais GNU também são copyleft, mas usam um copyleft muito mais simples, porque a complexidade da GNU GPL não é necessária para manuais.

2.4 Conclusão do capítulo

A escolha da utilização de software livre neste trabalho não é acaso. Após compreender realmente o que significa o termo "free software" e verificar que ele é capaz de criar um comunidade cooperativa de usuários e desenvolvedores, não tem como resistir a esta "nova" forma de construção/distribuição de software.

Só não sei até quando software livre conseguirá manter a gratuidade de sua distribuição, vejo que é necessário que os membros da comunidade contribuam (doações?) com o movimento. Vejo empresas migrando para software livre devido ao seu baixo custo, preferencialmente, custo zero é o que eles querem.

É importante ficar atento ao surgimento de uma nova classe de gigolôs. Os "gigolôs de software livre" são aquelas pessoas ou empresas que estão interessadas em tirar vantagem, extraindo dinheiro fácil na implantação de software livre sem que retorne um "tostão furado" para os desenvolvedores. Minha sugestão é que as contribuições sejam dentro das condições do usuário, me parece óbvio que uma empresa multinacional possa contribuir com mais do que um estabelecimento comercial do meu município Canguçu.

Vivemos em um mundo capitalista em que diariamente convivemos com contrastes sociais enormes, uma sociedade que não dá oportunidade àqueles que não possuem recursos financeiros para pagar (geralmente um preço alto) por saúde, moradia, educação, etc. Acredito que o software livre é capaz de socializar o acesso ao software e ao conhecimento nele empregado. Possibilitando que com estudo e dedicação, alcancemos nosso "lugar ao sol" na área da informática.

3 ESTADO DA ARTE EM SOFTWARE LIVRE

3.1 Introdução

No desenvolvimento desse projeto, procurarei utilizar somente ferramentas cuja licença seja livre. Sendo assim, inicialmente pesquisei até obter uma suite de ferramentas passíveis de serem utilizadas na implementação do estudo de caso proposto neste trabalho.

A pesquisa sobre as ferramentas citadas nesta seção são frutos de pesquisa nos sites *AppWatch* [APP 00], *freshmeat* [FRE 00] e *FreeOS* [FOS 00].

3.2 Sistemas operacionais livres

Sistemas operacionais são necessários para possibilitar a operação de qualquer sistema computacional ou de alguns dispositivos.

Alguns sistemas operacionais populares são Windows, MacOS, Unix, GNU/Linux e Solaris. Dos casos citados, GNU/Linux é um sistema operacional livre, enquanto que os demais não.

Sistemas operacionais livres são sistema operacionais que podem ser copiados, redistribuídos e que ainda oferecem o código fonte para que o usuário possa adaptá-lo as suas necessidades.

A seguir, abordaremos brevemente alguns sistemas operacionais livre existentes para plataforma PC, já que esta é a única plataforma que tenho acesso para implementar este trabalho.

- FreeBSD [BSD 00]
  FreeBSD é um avançado Sistema Operacional Unix BSD para plataforma PC, desenvolvido e mantido por uma coletividade de pessoas. Oferece performance, segurança, avançado suporte a redes e características de compatibilidade que ainda hoje falta em outros sistemas operacionais, até mesmo em alguns sistemas operacionais comerciais e proprietários. Provê robustos serviços de rede, até mesmo sobre grandes cargas de acesso e usa eficientemente a memória para

manter uma boa resposta a centenas, ou milhares, de processos simultâneos.
A qualidade do FreeBSD combinado com o baixo custo de hardware PC faz do FreeBSD um ótima alternativa para estações comerciais de trabalho UNIX.
Exige conhecimentos avançados para manutenção e é de difícil configuração, já que não existem ferramentas *user-friendly* para ajudar nessa tarefa.

- GNU/Linux [LNX 00]
  GNU/Linux é um sistema operacional livre do tipo Unix, sendo uma implementação independente de POSIX[5] que inclui multitarefa real, memória virtual, bibliotecas compartilháveis, carregamento em-demanda, gerenciador de memória otimizado, suporte a TCP/IP e outras características encontradas em sistemas do tipo Unix.
  Desenvolvido sobre a GNU GPL, o código fonte é distribuído livremente para qualquer um.

- Alliance [ALL 00]
  Alliance é um projeto "open source" que está desenvolvendo um sistema operacional baseado sobre o conceito de Cache Kernel (desenvolvido pela Universidade de Stanford) e CORBA da OMG.
  Alliance usa o conceito de Cache Kernel para criar um ambiente operacional para funcionar em direção a emulação de outros sistemas operacionais e plataformas de hardware. A meta não é somente ter um fantástico emulador de sistemas operacionais, mas prover um ambiente onde os dados transitem entre as fronteiras do sistema operacional. Ele também provê muitos recursos avançados para aplicações nativas (suporte a processos de tempo real, suporte a computação distribuída, entre outras características).
  Este sistema operacional encontrava-se em fase inicial de desenvolvendo quando este documento foi escrito, onde 75% da características planejadas para a primeira versão já estavam finalizadas, sendo distribuído conforme a licença GNU GPL.

- FreeDOS [DOS 00]
  FreeDOS é um sistema operacional "open source", sendo desenvolvido para atender à demanda dos usuários que não necessitam de um sistema operacional "complexo" como os disponíveis atualmente, aos que tem computadores velhos e desejam "ressuscitá-los" e para, finalmente, quem não quiser pagar nada por isso... Ao mesmo tempo, os desenvolvedores do FreeDOS estão procurando manter 100% de compatibilidade com o popularíssimo DOS!

[5] **POSIX** é um conjunto de padrões que definem funções e standards para um sistema operacional visando portabilidade.

Por ser um sistema monotarefa, monousuário, ele não é indicado no desenvolvimento deste projeto. Mesmo assim, citei-o apenas por motivos informativos.

- EROS [ERO 00]
  EROS é um novo sistema operacional sendo implementado na Universidade da Pensilvânia. O sistema funde algumas idéias antigas em sistemas operacionais com algumas novas idéias sobre performance e gerenciamento de recursos. O resultado é um pequeno, seguro, sistema operacional de tempo real que provê persistência ortogonal.
  O sistema se encontra na versão 0.8.3 atualmente e achei deveras interessante as suas características.

- GNU Hurd [HUR 00]
  GNU Hurd é o futuro kernel Unix para o sistema operacional GNU. O Hurd consiste de uma coleção de servidores que rodam sobre um micro-kernel Mach, gerenciando sistemas de arquivos, protocolos de rede, controle de acesso a arquivos, e outras características que são implementadas por kerneis Unix ou similares (como Linux, por exemplo).
  Atualmente, Hurd roda em máquinas i386. Futuramente, será portado para outras plataformas.
  Como todo software GNU, o GNU Hurd também é licenciado utilizando a GNU GPL.

- Minix [MNX 00]
  MINIX é um clone de UNIX livre. Com obrigação de ocupar poucos recursos, projeto baseado em micro-kernel e ampla documentação, ele é indicado para pessoas que querem rodar um sistema operacional Unix em seu computadores pessoais e aprender como o sistema funciona.
  Este sistema operacional tem fins didáticos, não sendo recomendado seu uso em máquinas de produção.

- NetBSD [NET 00]
  NetBSD é um sistema operacional UNIX livre, altamente portável, disponível para várias plataformas (quando este documento foi escrito, suportava 28 plataformas de hardware!), compreendendo desde estações Alpha de 64 bits até handhelds. Projetado com bom-senso, suportando uma larga variedade de hardware, distribuído através da licença BSD, com código fonte distribuído, seguro, maduro e estável, estes adjetivos tornam o NetBSD uma opção tanto para ambientes de produção como de pesquisa. A principal desvantagem, vem de não ser *user-friendly*.

- OpenBSD [OPE 00]
  O projeto OpenBSD produz um sistema operacional livre, multiplataforma, sendo um UNIX baseado no BSD4.4. Portabilidade, padronização, segurança e criptograficamente integrado, OpenBSD suporta emulação binária da maioria dos programas para SVR4(Solaris), FreeBSD, BSD/OS, SunOS e HP-UX. Atualmente roda em 10 plataformas diferentes de hardware e é considerado o sistema operacional mais seguro.

3.3 Banco de dados

Sistemas gerenciadores de banco de dados serão utilizados para armazenar eficaz e eficientemente todos os dados que a aplicação do estudo de caso deste trabalho utilizará.

A seguir, apresentaremos alguns bancos de dados livres.

- GNU SQL Server [SQL 00]
  SQL Server é um sistema gerenciador de bancos de dados multiusuário, portável e livre. Suporta completamente o dialeto SQL89 e implementa algumas extensões de SQL92. Ele provê acesso multiusuário e isolamento de transações através de "predicative locks". Roda somente em UNIXes. Também usa Remote Procedure Call (RPC), memória compartilhada e agendamento de mensagens. É distribuído conforme a GNU GPL e enquanto este documento foi escrito, encontrava-se em fase de desenvolvimento.

- Lago [LAG 00]
  Lago é um projeto que almeja prover um sistema gerenciador de bancos de dados livre licenciado através da GNU GPL. Lago é multi-thread, portável e escrito em C++.
  Devido as características atuais que este banco de dados implementa, não é possível utilizarmos no desenvolvimento deste projeto.

- Interbase [INT 00]
  Até mesmo as companhias de software proprietário estão procurando "aderir" ao Movimento de Software Livre, a Interbase abriu o código fonte de seu banco de dados distribuindo-o através de sua licença IPL (Interbase Public Licence).
  A versão 6 implementa algumas características interessantes como a criação de dialetos (adicionaram novas cláusulas ao padrão SQL, como por exemplo a cláusula ALTER COLUMN) e replicação.

- LEAP [LEA 00]
  LEAP é um sistema gerenciador de bancos de dados relacional que é utilizado como uma ferramenta educacional, ajudando estudantes, assistindo pesquisadores e professores como estudar e ensinar banco de dados.
  É distribuído através da licença GNU GPL e escrito em linguagem C padrão, sendo portável para várias plataformas operacionais.
  Implementa uma linguagem de consulta baseada em **álgebra relacional**, não suportando SQL. O que o deixa de fora da possibilidade de utilizá-lo no desenvolvimento desse projeto.

- PostgreSQL [PGS 00]
  PostgreSQL é um sofisticado sistema gerenciador de bancos de dados objeto-relacional, suportando quase todas cláusulas ANSI SQL, incluindo sub-selects, transações e tipos e funções definidas pelo usuário. É o banco de dados livre mais avançado atualmente, já que implementa herança, valores não-atômicos, entre outras características interessantes.
  A distribuição do PostgreSQL é baseada nos termos da Berkeley licence, que também é uma licença livre.

- mySQL [MYS 00]
  mySQL é um sistema gerenciador de banco de dados relacional. Distribuído conforme a GNU GPL, este banco de dados é apontado como um dos mais velozes bancos de dados da atualidade. Baseado na arquitetura cliente-servidor, este sistema consiste de um servidor SQL multi-threaded que suporta diferentes back-ends, vários programas e bibliotecas cliente, ferramentas de administração e interfaces de programação. O SQL implementado procura seguir o padrão ANSI SQL92.

3.4 Ferramentas de desenvolvimento para Internet

Como a proposta deste trabalho é modelar e parcialmente implementar uma aplicação para web, esta seção ilustra algumas ferramentas que poderão ser utilizadas na implementação do estudo de caso proposto.

3.4.1 Common Gateway Interface

A Common Gateway Interface (CGI) é um padrão para interfaceamento de aplicações externas com servidores web. Um documento HTML que um servidor web entrega é estático, o que significa que existe em um estado constante: um arquivo de texto que não

se altera. Um programa CGI, é executado em tempo real, então ele pode criar informação (páginas web) dinamicamente.

Um programa CGI pode ser escrito utilizando qualquer linguagem que pode ser executada no sistema, como:

- C/C++
- Fortran
- PERL
- TCL
- Qualquer Unix shell

3.4.2 Pré-processadores

Pré-processadores basicamente são programas que executam scripts que podem ou não estar embutidos no código HTML de páginas web. São estes pré-processadores que geram a parte dinâmica das páginas. Dependendo do pré-processador, é possível executar as mais diversas operações como: acesso a bases de dados e/ou recursos do sistema.

- HB [HB 00]
  HB age como uma extensão ao HTML. Rodando junto ao servidor web, HB depura os documentos conforme eles são requisitados. Com HB é possível criar conteúdo dinâmico de maneira fácil e poderosa.
  Licenciado através da GNU GPL, tem suporte a PERL e ao banco de dados PostgreSQL.

- PHP [PHP 00]
  PHP é uma linguagem script embutida em HTML. Basicamente, qualquer coisa que pode ser feita por algum programa CGI pode ser feita também com PHP, como coletar dados de um formulário, gerar páginas dinamicamente ou enviar e receber cookies.
  PHP também tem como uma das características mais importantes o suporte a um grande número de bancos de dados como: “dBase”, Interbase, mSQL, mySQL, Oracle, Sybase, PostgreSQL e vários outros. Construir uma página baseada em um banco de dados torna-se uma tarefa extremamente simples com PHP.
  Além disso, PHP tem suporte a outros serviços através de protocolos como

IMAP, SNMP, NNTP, POP3 e, logicamente, HTTP. Ainda é possível abrir sockets e interagir com outros protocolos.
Esta ótima ferramenta, até a versão 3, é licenciada através da GNU GPL.

3.4.3 Servidores de aplicações

Servidores de Aplicações oferecem um ambiente completo para o desenvolvimento de aplicações, "facilitando" o processo de construção de sites dinâmicos através de módulos pré-programados.

A seguir, brevemente introduzirei alguns servidores de aplicações de distribuição livre.

- Midgard [MID 00]
  Midgard é uma ferramenta livre para desenvolvimento de aplicações via web, baseado em na linguagem script PHP. É um projeto Open Source, dando a você liberdade para criar suas soluções em um ambiente aberto. Midgard é uma ferramenta para criar, modificar e gerenciar serviços web dinâmicos com suporte a banco de dados.
  Este servidor de aplicações é licenciado através da LGPL

- ZOPE [ZOP 00]
  Zope é um servidor de aplicações web que permite o trabalho colaborativo entre equipes, na criação e gerenciamento de aplicações dinâmicas baseada na web, como intranets e portais. Este servidor de aplicações torna fácil construir *search* e *news* personalizados, além de, aplicações e-commerce via web.
  Zope consiste de vários componentes que trabalham cooperativamente para prover um servidor de aplicações completo e flexível, incluindo um servidor de internet, um banco de dados objeto-transacional, um mecanismo de procura, um sistema de modelos de páginas web, uma ferramenta para gerenciamento de desenvolvimento e suporte a extensões.
  Este servidor de aplicações é distribuído conforme a licença própria, que não é compatível com GNU GPL.

3.4.4 Java Server Pages (JSP)

Java Server Pages é uma tecnologia desenvolvida pela Sun Microsystems que permite embutir código escrito na linguagem Java em páginas HTML, é possível também fazer com que o código Java das páginas HTML executem servlets[6] no lado do servidor

[6]**Servlets** são componentes de software escritos em Java.

web.

- GNUJSP [GJS 00]
  GNUJSP é uma implementação livre do Java Server Pages da Sun Microsystems, suportando a especificação 1.0 do JSP.
  Este produto é distribuído sob a licença GNU GPL.

- Resin [RES 00]
  Resin é uma veloz implementação de servlet e JSP que suporta "load balancing" para incrementar a performance e disponibilidade de serviços.
  Possui um servidor web compatível com o protocolo HTTP/1.1 dedicado a servir conteúdo dinâmico em Java.
  Suporta a especificação 2.2 para Servlet e a 1.1 para JSP da Sun.
  Este software é distribuído conforme a licença QPL.

- Tomcat [TOM 00]
  Tomcat é uma implementação na forma de software livre das especificações Java Servlet 2.2 e JSP 1.1. Esta implementação é usada no servidor web Apache [APA 00] bem como em outros servidores web e ferramentas de desenvolvimento.
  Este produto é licenciado através da licença Apache.

- PolyJSP [POL 00]
  PolyJSP é uma implementação de Java Server Pages projetada para suportar múltiplas linguagens script e múltiplas versões de JSP.
  Completamente baseada em XML e XSL, PolyJSP atualmente suporta scripts em Java, JavaScript e WebL. Suporta parcialmente a especificação 0.92 do JSP.
  Segundo informações retirados do seu website, PolyJSP é um produto distribuído conforme a licença Apache. Quando entrei em contato com Ricardo Rocha (ricardo@apache.org), que é o seu desenvolvedor, ele informou-me que o desenvolvimento deste produto está parado.

3.5 Servidor de HTTP

Servidor de HTTP[7] é um programa que utilizando o modelo cliente/servidor e o protocolo HTTP, serve a requisições de páginas de usuários da World Wide Web.

A seguir, brevemente e sem intenção de esgotar o assunto, apresentarei alguns servidores de HTTP que pesquisei.

[7] **Servidores de HTTP** também podem ser chamados de "servidores web"

- Apache [APA 00]
  Apache é o servidor HTTPD mais popular, sendo possivelmente o melhor em termos de funcionalidade, eficiência, segurança e velocidade. Foi originalmente baseado no código e idéias do NCSA httpd 1.3. Extremamente extensivo, suporta módulos PERL, PHP, XML entre outros.
  Este servidor de HTTP é licenciado através de licença própria.

- ghttpd [GHT 00]
  ghttpd é um servidor HTTP rápido e eficiente. Suporta CGI, virtual hosts e inetd, sendo capaz de manipular centenas de conexões simultaneamente. De fácil configuração, ghttpd roda na maioria dos sistemas UNIX e é distribuído sob a licença GNU GPL.

- Xitami [XIT 00]
  Desenvolvido e mantido pela empresa iMatix, Xitami é um servidor web que roda em todas plataformas UNIX, OS/2, OpenVMS, Windows 3.x, Windows 95 e Windows NT. De fácil instalação, já que o software vem totalmente pré-configurado, não exigindo a configuração de uma se quer opção para funcionar. Xitami aloca poucos recursos, é rápido e robusto: ele é baseado em sólidas ferramentas de engenharia de software que a iMatix desenvolveu para servidores TCP/IP multithreaded. Ao contrário de outros servidores web, Xitami foi projetado como um conjunto de componentes reusáveis altamente portável. Tornando fácil a extensão e reincorporação de seus módulos.
  Distribuído sob licença Open Source este servidor web é gratuito.

- thttpd [THT 00]
  thttpd é um servidor web simples, pequeno, rápido e seguro. Possui algumas características extremamente úteis (como URL-traffic-based throttling) que nenhum outro servidor possui. Adicionalmente, suporta IPv6 nativamente.
  Este servidor web é licenciado sob licença BSD

3.6 Implementações de Máquina Virtual Java

Uma Máquina Virtual Java é uma suite de ferramentas, que formam um ambiente capaz de interpretar e executar programas escritos na linguagem Java.

- Kaffe [KAF 00]
  Kaffe é uma implementação livre da Máquina Virtual Java e de suas classes. É um projeto de software livre porque pratica a licença GNU GPL.

Kaffe é compatível com a especificação do Java Development Kit[8] 1.1, faltando apenas algumas partes, sendo que outras já suportam a especificação 1.2.

- Japhar [JAP 00]
  Japhar é "um interpretador portável para bytecodes Java"[JAP 00] distribuído sob a licença LGPL. Enquanto esta monografia está sendo escrita, Japhar encontrava-se em um estágio inicial de desenvolvimento, apesar de já implementar algumas funcionalidade dos JDKs 1.1 e 1.2.

3.7 Navegadores Web

Navegadores Web são programas que permitem a visualização de páginas web.

Estes programas poderão ser utilizados pelos usuários da aplicação que este trabalho pretende construir.

- Amaya [AMA 00]
  Amaya é um completo navegador e ambiente de autoria com interface "WYSIWYG" [9]. Com esse estilo de interface possibilita gerar facilmente páginas HTML e XHTML.
  Este navegador suporta as tecnologias estado da arte para web, licenciado sob uma licença do tipo BSD, este projeto é hospedado pelo W3C[10].

- Armadillo [ARM 00]
  Armadillo é um navegador distribuído sob GNU GPL, escrito completamente em C utilizando a biblioteca gráfica GTK+. Seu criador pretende torná-lo rápido, eficiente, altamente extensível e compatível com todos padrões para web.

- Chimera [CHI 00]
  Chimera é um navegador web baseado em UNIX para computadores que rodam o ambiente gráfico X Window. Utiliza a biblioteca gráfica Athena, é horrível de feio e suporta algumas características da versão 3.2 do HTML.

- Galeon [GAL 00]
  Galeon é um navegador para o ambiente GNOME[GNO 00] baseado no "gecko"

[8] **Java Development Kit (JDK)** é uma suite de ferramentas que possibilitam o desenvolvimento de aplicações em Java.

[9] **What You See Is What You Get (WYSIWYG)**, nome dado a ferramentas que permitem que o usuário edite documentos e visualize o resultado final, sem que seja necessário executar qualquer passo adicional para isso.

[10] O **World Wide Web Consortium (W3C)**, desenvolve tecnologias interoperáveis (especificações, normas, software e ferramentas) para conduzir a Web para todo seu potencial como fórum de informações, comércio, comunicações e entendimento coletivo.

(renderizador HTML do Mozilla). É rápido, tem uma interface limpa e é compatível com os padrões atuais.

- Konqueror [KON 00]
  Konqueror faz parte do ambiente KDE[KDE 00] e além de ser um navegador web, agrega outras características como: gerenciador de arquivos e visualizador universal. Suporta as tecnologias estado da arte da web e é licenciado através da QPL e GNU GPL.

- Mozilla [MOZ 00]
  Mozilla surgiu quando a Netscape Corporation liberou o código fonte do seu navegador Netscape. Atualmente é licenciado através da MPL, mas há planos de liberarem o código através da GNU GPL.

- SkateBrowser [SK8 00]
  Skate é um navegador escrito em Java, que não requere muitos recursos do sistema, oferece suporte a toda especificação HTML 3.2, servidor proxy e suporte a Java (não suporta Javascript). É licenciado através da GNU GPL.

3.8 Conclusão do capítulo

Como pode ser visto no decorrer deste capítulo, opções de ferramentas livres é o que não falta. Concordo que algumas são de manuseio bem mais complicado do que outras (principalmente se considerarmos soluções proprietárias), mas acredito que com o aumento da comunidade usuária, os desenvolvedores acabarão procurando facilitar a utilização de seus produtos (através de front-ends gráficos, etc), sempre mantendo a filosofia do software livre e encorajando os usuários a utilizarem seus cérebros.

4 METODOLOGIA DE ENGENHARIA DE SOFTWARE

4.1 Introdução

> "Software é um lugar onde sonhos são plantados e pesadelos colhidos, uma abstração, um pântano místico onde demônios terríveis competem por panacéias mágicas, um mundo de lobisomens e balas de prata" [Cox 90]

Conforme vou adquirindo maior conhecimento sobre engenharia de software e trabalhando no desenvolvimento de aplicações, concordo com a definição de software apresentada acima e concluo que a utilização de um método para desenvolvimento de software é um fator indispensável para construção de sistemas. Quando o desenvolvimento é feito por uma só pessoa e o ambiente a ser informatizado é simples, talvez a utilização de uma metodologia até possa ser dispensada. Mas quando uma equipe composta por várias pessoas com funções diversas se unem no desenvolvimento de uma aplicação acredito ser indispensável a utilização de um método de engenharia de software. Neste caso, o método principalmente organiza os processos de desenvolvimento, permitindo um gerenciamento mais eficaz através de uma distribuição de tarefas mais elaborada e bem definida. Além de facilitar a comunicação entre os componentes da equipe. O principal problema que vejo na adoção de uma metodologia é principalmente com treinamento, visto que cada componente da equipe possui uma curva de aprendizado própria e o custo com consultoria pode ser elevado. Mesmo assim, acredito que se a equipe for composta pelas pessoas certas, o investimento vale a pena e se paga rapidamente. "O problema está deixando de ser o software e o hardware - cada dia mais poderosos do que no dia anterior - para se tornar os métodos de trabalho que empregamos no uso dessa tecnologia, bem como seu alvo de utilização." [Fur 98]

Segundo [Ber 00] as melhores técnicas para engenharia de software seriam aquelas que:

- poderiam ser descritas quantitativamente, bem como qualitativamente;
- poderiam ser usadas repetidamente, cada vez arquivando resultados similares;
- poderiam ser ensinadas a outras pessoas dentro de um razoável espaço de tempo;
- poderiam ser aplicadas por outras pessoas com uma razoável nível de sucesso;

- alcançassem significativamente e consistentemente, melhores resultados do que qualquer outra técnica ou abordagem “ad hoc”;
- seja aplicável em uma larga porcentagem de casos;

A seguir, apresentarei brevemente algumas metodologias que utilizam uma abordagem orientada a objetos para a construção de software. A escolha por esta abordagem, e não outra, se deu devido ao referencial teórico citar, que atualmente, a orientação a objetos é a abordagem estado da arte no desenvolvimento de aplicações.

4.2 Abordagem para desenvolvimento de software orientado a objetos

> “O projeto Orientado a Objetos (OO) tem se consolidado como a melhor maneira de alcançar a reusabilidade, visto que ele permite definir por abstração e por composição diferentes componentes de uma aplicação, que são interligados a fim de atender o comportamento esperado do sistema. Os dados da aplicação são distribuídos entre os componentes, que tornam-se responsáveis pela sua manutenção e pelo alcance dos comportamente estabelecido.”[Piz 98]

O enfoque tradicional de modelagem para a construção de sistemas de informação baseia-se na compreensão desse sistema como um conjunto de programas que, por sua vez, executam processos sobre dados. O enfoque da modelagem por objetos, vê o mundo como uma coletânea de objetos que interagem entre si, apresentam características próprias que são representadas pelos seus atributos (dados) e operações (processos).

Essa abordagem Orientada a Objetos é justificada pelo fato de que objetos existem na natureza muito antes de haver qualquer tipo de aplicação deles pelo negócio: equipamentos, pessoas, minerais, petróleo etc., existem por si só, apresentam características peculiares representadas pelos seus atributos e, adiante, pelo seu comportamento no mundo real. Essa abordagem oferece como principais benefícios:

- Manter a modelagem do sistema e, em decorrência, sua automação o mais próximo possível de uma visão conceitual do mundo real;
- Servir de base à decomposição e modelagem do sistema nos dados, que é o elemento mais estável de todos aqueles que compõem um sistema de informação;
- Oferecer maior transparência na passagem da fase de modelagem para a de construção através da introdução de detalhes, não requerendo uma reorganização do modelo.

Programas orientados a objetos são feitos de objetos. Estes objetos empacotam tanto os dados como os procedimentos que operam sobre esses dados. Os procedimentos são chamados de métodos ou operações. Um objeto executa uma operação, quando recebe uma solicitação (ou mensagem) de um cliente. As solicitações são a *única* maneira de conseguir que um objeto execute uma operação. As operações são a *única* maneira de mudar os dados internos de um objeto. Devido a essas restrições, dizemos que o estado interno de um objeto está *encapsulado*; ele não deve ser acessado diretamente e sua representação é invisível do exterior do objeto.

A parte difícil sobre projeto orientado a objetos é a decomposição de um sistema em objetos. A tarefa é difícil porque muitos fatores entram em jogo: encapsulamento, granulidade, dependência, flexibilidade, desempenho, evolução, reutilização e assim por diante.

Entram então as metodologias de projeto orientado a objetos, procurando simplificar o processo de decomposição do sistema em objetos através da divisão do processo em atividades, com tarefas a serem executadas em cada atividade.

A seguir, apresentarei brevemente alguns métodos de desenvolvimento de software orientado a objetos.

4.3 Métodos para desenvolvimento de software orientado a objetos

Nas últimas décadas houve o aprimoramento constante das técnicas e metodologias para a engenharia de software visando acompanhar a evolução natural do conhecimento, incluindo a análise e o projeto estruturado, a análise essencial e a Engenharia da Informação.

Métodos orientado a objetos procuram tanto redefinir quanto estender métodos existentes. Não obstante sejam comuns no mercado metodologias híbridas combinando vários aspectos da orientação a objetos e métodos estruturados, alguns críticos têm defendido a posição de que não é possível haver uma continuidade das técnicas estruturadas alegando não serem “objetificáveis”. De qualquer forma, a orientação a objetos é fundamentalmente uma nova maneira de se pensar, um novo paradigma, e não simplesmente uma linguagem de programação.

A seguir abordaremos brevemente alguns métodos que surgiram no campo da orientação a objetos.

- BOOCH
  "Grady Booch propôs um método que consistia no emprego de técnicas de desenho orientado a objetos, apesar de ter sido estendido para contemplar também a análise orientada a objetos. Booch descreve um objeto como sendo um modelo do mundo real que consiste de dados e habilidades para o tratamento desses dados. Uma vez tendo sido localizados e definidos os objetos, esses passam a servir de base para os módulos do sistema ou são considerados módulos relacionados. Descreve o projeto estruturado e o projeto orientado a objeto como sendo visões ortogonais de como um sistema deveria ser criado. Enquanto o projeto orientado a objetos estuda o problema com objetos que existem em um domínio de problema, o projeto estruturado separa o sistema em módulos. Argumenta que o projeto estruturado funciona bem com linguagens de programação estruturada, mas o enfoque estruturado direciona a concentração de esforços na lógica do módulo com pouca ênfase no uso dos dados. O resultado, de acordo com Booch, é a geração de código adicional e menos dados, pois as estruturas são derivadas dos processos e esses necessitam interagir com dados. O projeto orientado a objeto é organizado via abstrações algorítmicas de forma parecida à programação orientada a objeto. Cada módulo representa um objeto ou uma classe de objetos com reutilização e encapsulamento ao longo dos processos. Essa estreita relação existente entre desenho e programação orientadas a objeto permite aos projetistas tomar decisões de negócio antes da criação do código, contribuindo para a melhoria do sistema gerado." [Fur 98]

- OMT
  "Rumbaugh, com seu método que fora desenvolvido pela GE Corporation, também conhecido como Técnica de Modelagem de Objetos (OMT - *Object Modeling Technique*). Baseado na modelagem semântica de dados, o método Rumbaugh tornou-se um enfoque testado e maduro, cobrindo as diversas fases do desenvolvimento orientado a objeto. A notação empregada por Rumbaugh é parecida com a dos métodos estruturados e utiliza a notação de modelo de objeto que suporta conceitos de modelagem de dados (exemplo: atributos e relacionamento), objetos (composição/agregação) e herança. Também é empregada para categorizar classes e instâncias. O ponto forte do método Rumbaught é a notação utilizada e o enfoque relativamente conservador no uso da teoria de objetos. Por outro lado, um problema apresentado é a falta de notação específica para representar a passagem de mensagens de um objeto a outro." [Fur 98]

- OOSE

  "Jacobson, com sua técnica, criou as bases para os métodos *Object-Oriented Software Engineering* (OOSE) e o Objectory. O que diferencia Jacobson de outros métodos orientados a objeto é o seu foco em casos de uso e categorização de pessoas e equipamentos dependendo de seu papel no sistema global. A análise no método de Jacobson é baseada em modelos de requerimentos e análise de problema e de uma descrição da interface do sistema. Seguindo a criação do modelo de análise, são gerados diagramas baseados no modelo que também descrevem a comunicação entre blocos. O modelo de blocos é então implementado utilizando-se linguagens apropriadas e ferramentas orientadas a objeto. Um dos pontos fracos do enfoque original da OOSE de Jacobson é a notação simplista usada para objetos de domínio (objetos simplesmente são representados como círculos). Por outro lado, o método Objectory tem sido adaptado para a engenharia de negócio, onde as idéias são usadas para modelar e melhorar processos." [Fur 98]

- SHLAER/MELLOR

  "Sally Shlaer e Stephen Mellor criaram um método orientado a objetos que pode utilizar ferramentas tradicionais, ressuscitando até mesmo os diagramas de fluxo de dados como técnica de visualização do modelo de comportamento do objeto." [Fur 98]

  "Trata-se de um método bem definido e disciplinado, orientado para o desenvolvimento de software a uma escala industrial. Este método inclui um conjunto de modelos de análise orientados a objeto, que podem ser simulados para verificação, e uma aproximação inovadora à fase de projeto, denominada *Recursive Design* (RD), que produz o projeto do sistema através da tradução dos modelos de análise. O modelo de informação de objetos é baseado no modelo relacional dos dados. Os modelos de estado são autômatos determinísticos finitos. Os diagramas de fluxo de dados mostram a seqüência, podendo cada processo ser definido numa linguagem formal. Todos os modelos podem ser construídos utilizando ferramentas CASE, sem a necessidade de serem desenhados diretamente pelo analista. Isto é especialmente verdade para o modelo de comunicações de objetos, pois este modelo mostra as comunicações entre os objetos. Uma vez construídos os modelos fundamentais, os modelos restantes podem ser derivados, garantindo a consistência entre eles." [LAl 97]

  "O método Shlaer/Mellor difere da abordagem estruturada ao se utilizar de um enfoque de modelagem *bottom-up*. Os métodos estruturados empregam técnicas *top-down* que, conforme argumentado por Booch, tendem a criar mais

código do que o necessário para o desenho orientado a objeto. O enfoque de Shlaer/Mellor é uma variação da modelagem de dados, concentrando-se nos dados ou objetos e como eles são agrupados. Mensagens passadas de um objeto a outro determinam a ordem das operações. Especialistas têm criticado o método Shlaer/Mellor por não oferecer um esquema para o tratamento de mensagens e encapsulamento, considerados como quesitos fundamentais da teoria de objetos. Seu suporte a conceitos de tipo e herança também é pobre, quando comparado a outros métodos." [Fur 98]

- COAD/YOURDON

  "Peter Coad e Ed Yourdon, com seu enfoque simples, porém eficaz, dividiram a análise orientada a objetos como sendo classes e objetos. Objetos são abstrações de um domínio de problema que reflete a habilidade de um sistema em reter dados, interagir com dados e encapsular valores de atributos e serviços. Classe é uma coleção de um ou mais objetos com atributos e serviços, abrangendo o entendimento de como um novo objeto é criado em uma classe. A técnica para análise orientada a objeto de Coad/Yourdon usa quatro elementos adicionais para descrever um sistema: estrutura, sujeito, atributo e serviço. Estrutura é uma representação de um domínio de problema que está diretamente relacionado às competências do sistema. Coad/Yourdon utilizam-se dessa forma em termos de estruturas de generalização/especificação ou agregações. Os sujeitos organizam o trabalho baseado nos requerimentos iniciais através de um mecanismo de agrupamento de classes e objetos. Atributo é algum tipo de dado onde cada objeto possui um valor independente (cada objeto na classe possui seu valor próprio). E, finalmente, os serviços são o que o objeto pode fazer ou que é de sua responsabilidade fazer. Coad/Yourdon propõem a identificação e definição de classes e objetos, estruturas, sujeitos, atributos e serviços como passos básicos para a realização da análise orientada a objeto. O diagrama que Coad/Yourdon sugerem para representar tais conceitos é basicamente um único e a característica significativa de um objeto/classe é definida dentro do símbolo do objeto. Para relacionar um objeto a outro, Coad/Yourdon empregam conectores e associações - dois ou mais objetos são associados através de conectores. O conector ou a associação pode ser uma generalização/especialização, todo/parte, instância-a-instância ou uma conexão de mensagem. A conexão de objetos utilizando-se das associações de Coad/Yourdon permite estudar como as classes/objetos relacionam-se entre si. Essa habilidade em conectar objetos utilizando diferentes associações faz do método de Coad/Yourdon algo flexível e capaz de reagir à maioria das situações de negócio." [Fur 98]

- MARTIN/ODELL

  "Martin/Odell propuseram, na visão de muitos observadores de mercado, uma extensão orientada a objeto da Engenharia da Informação. Ambos os métodos compartilharam de várias notações comuns, tais como tipo de objeto e classes que são representados pelo mesmo símbolo. O método Martin/Odell procura, evidentemente, capitalizar sobre o conhecimento já existente na comunidade de informática, aplicando notações similares e confortáveis aos profissionais que empregam a Engenharia da Informação. Esse método também utiliza uma notação semelhante à de Coad/Yourdon, pelo menos conceitualmente. Possui forte ênfase na modelagem de dados, conforme é notado pelo modelo de esquema de objeto. O propósito desse método, que poderíamos denominar Engenharia da Informação orientada a objeto, é definir áreas comuns entre dados e serviços dos objetos para, conseqüentemente, obter a máxima reutilização possível. Apesar de o método Martin/Odell não estar sendo muito utilizado, pode ser empregado em certas organizações, em decorrência da popularidade da Engenharia da Informação." [Fur 98]

- WIRFS-BROCK

  "Rebecca Wirfs-Brock propõe uma metodologia dirigida a responsabilidades cujo maior diferencial frente aos demais métodos é o uso de cartões CRC - *Class-Responsability-Collaboration*, desenvolvidos por Ward Cunningham. Tais cartões permitem um mapeamento de classes para execução de certas responsabilidades através de colaboração mútua. Wirfs-Brock torna explícitos os relacionamentos de herança empregando grafos de hierarquia e diagrama de Venn, mas não oferece a possibilidade de modelar associações de composição. É satisfatório enquanto que tange a conceitos de contratos e colaboração e apresenta bom potencial de reutilização ao propor que atributos de classes não serão totalmente conhecidos até a fase de implementação." [Fur 98]

- EMBLEY/KURTZ

  "Embley/Kurtz, da Universidade de Bigham Young, enfatizam uma forte fundamentação na modelagem de dados, o que inclui um modelo objeto/relacionamento que suporta vários relacionamentos. Também suporta notações para conjunto de relacionamentos multidimensionais que podem incluir regras, como é o caso da mútua exclusividade. É interessante notar que esse método, além de ser menos popular do que os demais, não modela prontamente os atributos dos objetos, fato que o classifica em uma categoria própria." [Fur 98]

Não é minha intenção esgotar o assunto neste trabalho, portanto apresentei somente alguns, dos vários, métodos orientado a objetos. Lembrando que a proposta é desenvolver uma aplicação para ser acessada via web, utilizando um navegador, pesquisei sobre métodos orientado a objetos para construção de aplicações hipermídia.

4.3.1 Métodos orientado a objetos para aplicações hipermídia

> "Hipermídia é a junção dos tipos de dados da multimídia com os mecanismos e semânticas dos hipertextos, ou seja, um sistema ou aplicativo hipertexto que, além de texto e gráficos, suporta outros tipos de mídia, tais como desenhos, imagem, som e vídeo." [CZL 00]

Aplicações Hipermídia permitem que o usuário da aplicação navegue através de "nós de informação" de maneira intuitiva.

- OOHDM

  Gustavo Rossi [Ros 96] desenvolveu um Método para Projeto Orientado a Objetos de Aplicações Hipermídia que consiste de uma abordagem baseada em modelos para construção de grandes aplicações hipermídia. Esta metodologia vem sendo utilizada no projeto de diversos tipos de aplicações como: web sites e sistemas de informação, quiosques interativos, apresentações multimídia, etc. OOHDM considera o processo de desenvolvimento da aplicação hipermídia como um processo de quatro atividades, desempenhadas em uma mistura de estilos iterativos e incrementais de desenvolvimento; em cada etapa um modelo é construído ou enriquecido. Como foi citado, tratando o modelo conceitual, navegacional e de interface como atividades separadas, permite que nos concentremos em diferentes interesses de cada vez. Conseqüentemente, obteremos projetos mais reusáveis e modulares. Além do modelo de interface facilmente ser mapeado para linguagens ou ambientes que não sejam orientado a objetos (como HTML), deste modo OOHDM pode ser usada independentemente da plataforma alvo ser um ambiente puramente orientado a objetos ou híbrido (como usualmente encontramos na Internet).

- HDM

  "HDM é o primeiro modelo amplamente conhecido para o projeto de hipermídia. Tem sido usado por mais de 20 grupos de desenvolvimento em seis países diferentes. Em HDM há poucas primitivas de modelagem, portanto, o modelo resultante é bastante conciso. Contudo, desde modo, muitas decisões de projeto permanecem não documentadas. Um esquema hipermídia em HDM é descrito como um conjunto de tipos de entidades e de elos; as entidades são constituídas por componentes hierarquicamente estruturados e cada componente pode

ser visto sob diferentes perspectivas, usando Unidades. São usados Elos Estruturais para relacionar diferentes componentes de uma entidade ou diferentes perspectivas de uma unidade. São usados, em HDM, estruturas de acesso como índices e roteiros guiados para ajudar o autor a simplificar o acesso a entidades na aplicação hipermídia. Extensões subseqüentes do modelo de dados HDM original (como HDM2) acrescentaram novas primitivas de modelagem, como 'slots' dando às entidades uma estrutura no estilo dos 'frames'. Mais recentemente, foi proposto o uso de coleções multimídia como uma forma de organizar melhor a estrutura navegacional da aplicação. HDM é uma estrutura em evolução, sendo continuamente melhorada." [Ros 96]

- EORM
  "O *Enhanced Object Relationship Model* (EORM) é uma abordagem de modelagem hipermídia orientada a objetos que utiliza mecanismos de abstração conhecidos para a construção de modelos de relacionamentos entre objetos de domínio. Em EORM, um esquema conceitual é construído usando as primitivas de OMT, ou seja, entidades do mundo real são modeladas como objetos coletados em classes. EORM é bastante similar a OOHDM no fato de usar o modelo de orientação a objetos como a filosofia central para a descrição de aplicações hipermídia." [Ros 96]

4.4 O processo de desenvolvimento

Martin Fowler em [Fow 00a], aborda uma visão geral do processo de desenvolvimento de software, ele não define um método, mas dá uma visão de alto nível do processo.

O processo de desenvolvimento proposto é iterativo e incremental. Iterativo porque o software não é implementado em um instante no fim do projeto, mas desenvolvido e implementado em partes, e incremental porque o desenvolvimento vai adicionando complexidade aos modelos conforme o avanço das fases.

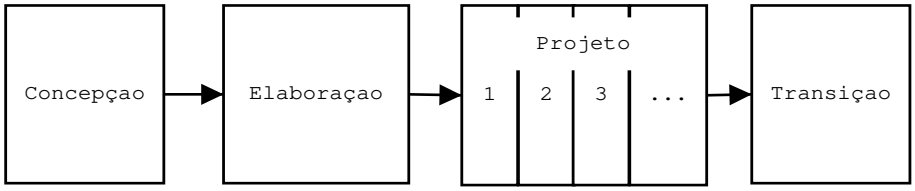

Figura 1: Visão geral do processo de desenvolvimento de software.

Os projetos variam quanto ao volume de **formalidade**, onde quanto maior o projeto, mais formalidade é necessária. Projetos com pouca formalidade podem ter uma fase de concepção que cabe em uma folha de papel e foi feita durante um bate-papo, enquanto que projetos formais possuem documentos formais a serem entregues, reuniões formais e encerramentos formais.

Segundo Fowler, as fases de um processo de desenvolvimento são:

1. **Concepção**
   Pode ser feita de várias formas, devendo ser um trabalho de poucos dias para considerar se vale a pena continuar o desenvolvimento. Desenvolve-se um plano de negócio do projeto — a grosso modo quanto custará e quanto terá de retorno. Talvez seja necessário fazer alguma análise inicial para se ter uma idéia do escopo e tamanho do projeto.

2. **Elaboração**
   Nesta fase procura-se uma melhor compreensão do problema.

   - O que é que vai ser realmente construído?
   - Como vai ser construído?

   É feito também o levantamento de riscos do projeto, Fowler comenta três tipos de riscos:

   - Riscos de requisitos
     Levanta-se os requisitos da aplicação para que seja construído um sistema que atenda as necessidades do cliente. Se dá pela feitura dos famosos caso de uso da UML, descobrindo-se os casos de uso importantes e arriscados. Pode também ser construído um diagrama de classes, feito a partir de uma perspectiva conceitual. Fowler aconselha que seja construído um protótipo de todas as partes difíceis dos casos de uso. “A criação de protótipos é uma técnica valiosa para obter uma melhor compreensão de como situações mais dinâmicas funcionam.” [Fow 00a]
   - Riscos Tecnológicos
     A construção de protótipos na avaliação dos riscos tecnológicos é fundamental para que se experimente as partes da tecnologia que está pensando em utilizar. Procure juntar elementos que permitam sua troca com relativa facilidade. Pense sobre o que fazer se:
     - Uma peça de tecnologia não funcionar.
     - Não for possível conectar as duas partes do quebra-cabeça.

  - Algo der errado.

- Riscos de Especialização
  Tem-se a equipe certa para a construção da aplicação? É necessário treinamento? “Treinamento é um modo de evitar erros, porque os instrutores já cometeram aqueles erros. Errar toma tempo, e tempo custa dinheiro”[Fow 00a]

Quando os desenvolvedores forem capazes de estimar o tempo para construção de cada caso de uso e todos os riscos significativos já foram devidamente identificados significa que esta fase chegou ao fim.

3. **Projeto** ou **Construção**
   Inicialmente deve ser planejado como o sistema vai ser construído, estipulando prazos para desenvolvimento dos casos de uso. Assim é possível estar ciente do progresso do desenvolvimento. Nesta fase o sistema é construído em uma série de iterações. Cada iteração deve ser considerada um miniprojeto onde é feito análise, projeto, codificação, teste e integração para os casos de uso designados para cada iteração. A iteração termina com uma demonstração para o usuário e com a realização de testes de sistema para confirmar que os casos de uso foram construídos corretamente. “Uma característica-chave para o desenvolvimento iterativo é que ele é congelado nas datas — você não pode deixar passar nenhuma data.”[Fow 00a] Interessante que a única coisa que é possível afirmar sobre um plano é que as coisas não vão acontecer de acordo com ele. O gerenciamento de um plano tem tudo a ver com a maneira como você lida efetivamente com as mudanças.

4. **Transição**
   Nesta fase não é feito desenvolvimento para acrescentar funcionalidade, a não ser que seja pequena e essencial. Este é o momento de otimizar o código e corrigir bugs. “Um exemplo da fase de transição é o período entre a versão beta e a versão final do produto.” [Fow 00a]

4.5 Conclusão do capítulo

Neste capítulo foi apresentado brevemente várias metodologias para o desenvolvimento de aplicações orientada a objetos, além de uma visão geral do processo de construção de aplicações.

Decidi utilizar OOHDM como método de projeto, já que esta metodologia foi a que, na minha opinião, mais se adequá ao que pretendo desenvolver com este trabalho.

O problema é que, como o próprio nome da metodologia diz, OOHDM é um método de projeto e já vimos que o processo de desenvolvimento engloba mais do que isso.

Dessa forma, neste trabalho vou adicionar a OOHDM as fases de concepção, elaboração e transição, proposta por Fowler em [Fow 00a].

É importante ressaltar que não estou propondo a extensão do método OOHDM, já que dada a complexidade e a formalidade necessária para isso, foge do escopo desta monografia, portanto estou adicionando as fases de maneira empírica e intuitiva.

Convém lembrar que o intuito dessa monografia é modelar o sistema de gestão e implementar um protótipo, dessa forma, não atingirei a fase de transição porque o protótipo construído não possuirá todas as funcionalidade de uma versão beta.

5 ESTUDO DE CASO

5.1 Introdução

O estudo de caso desenvolvido neste trabalho é a modelagem de um sistema de gestão para pomares de pessegueiros, sendo este capítulo o resultado de várias reuniões que buscaram formalizar e definir o escopo deste sistema. Foi feito um levantamento de requisitos baseado na visão do cliente sobre o que seria o seu negócio.

5.2 Objeto de estudo

Nosso objeto de estudo é o pomar de pessegueiros *"Arroio das Pedras"*, de propriedade do Sr. Adelci dos Santos Camargo, localizado no 5º distrito do município de Canguçu-RS. Enquanto esta monografia estava sendo escrita, o pomar Arroio das Pedras possuía em torno de 47.000 plantas distribuídas em 58 hectares de pomar.

5.3 O pessegueiro

Esta monografia estaria incompleta sem um histórico da planta que é o principal motivo deste trabalho existir.

> "O pessegueiro é uma espécie nativa da China, tendo sido encontradas referências na literatura chinesa de 20 séculos a.C. O nome, entretanto, é originário da Pérsia, que foi erroneamente tomada como país de origem dessa espécie. O pessegueiro era também conhecido no mundo greco-romano, no século que antecedeu a Cristo. Provavelmente, teria sido levado da China à Pérsia e daí espalhado pela Europa. No Brasil, o pessegueiro foi introduzido em 1532 por Martin Afonso de Souza, por meio de mudas trazidas da Ilha da Madeira e plantadas em São Vicente, São Paulo.
>
> Atualmente, em nosso País, o cultivo do pessegueiro ocupa área superior a 20 mil hectares, com produção anual que ultrapassa 100 mil toneladas.
>
> Na região Sul e, em particular, no Rio Grande do Sul, a cultura do pessegueiro só passou a ter maior importância a partir da década de 60. Antes disso, mais de 80% do pêssego consumido era importado, sobretudo, da Argentina. Foi, provavelmente, na década de 50 que um agricultor notou que a planta de pessegueiro oriunda de caroços jogados ao solo apresentava boa adaptação e produção. Essa planta foi multiplicada, recebendo o nome de "Aldrighi", cultivar que deu grande impulso à expansão da espécie.
>
> No início dos anos 50, Orlando Rigitano iniciou, em São Paulo, um programa de melhoramento genético de fruteiras de clima temperado, visando a obter adaptação

às condições de inverno ameno. Poucos anos depois, um programa semelhante era iniciado em Taquari/RS, na Estação Fitotécnica, por Sérgio Sachs. Em 1962, esse programa foi transferido para a Estação Experimental de Pelotas, hoje parte do Centro de Pesquisa Agropecuária de Clima Temperado-CPACT, da Embrapa.

Esses programas deram grande estímulo à cultura, chegando a haver, só no Rio Grande do Sul, mais de 2.500 produtores e mais de 100 indústrias de conserva, que tinham, no processamento do pêssego, a principal atividade.

Atualmente, o quadro é pouco distinto. Existem ainda, na região, pouco mais de 22 indústrias, que empregam cerca de 12 mil pessoas, na época de safra, as quais têm no pêssego a principal e, em alguns casos, a única atividade. Persistem ainda, no sul do RS, aproximadamente, 1.500 pequenos produtores. Observa-se, entretanto, que essa cultura começa a expandir-se para áreas tradicionalmente cultivadas com soja, trigo e fumo, ou exploradas com a criação de gado.

O consumo de pêssegos no Brasil ainda é muito pequeno; segundo dados da Associação Gaúcha dos Produtores de Maçã - apenas 0,85 kg por habitante/ano. Esse baixo consumo é explicado, em grande parte, pelo reduzido poder aquisitivo da população e também pela falta de investimentos em propaganda e em esclarecimentos ao consumidor, que considera o pêssego ainda como sobremesa, quando deveria considerá-lo um complemento alimentar.

Análises realizadas nos laboratórios do Centro de Pesquisa Agropecuária de Clima Temperado-CPACT, em algumas cultivares locais, comprovam que o teor de vitamina C, no fruto fresco, está entre 26,6 a 30,0 mg/100g. A quantidade dessa vitamina recomendada pelo Conselho Nacional de Pesquisa dos Estados Unidos é de 45 mg/dia, o que significa que duas frutas são suficientes para suprir as necessidades diárias do organismo humano. Além da vitamina C, o pêssego de polpa amarela é também rico em carotenóides, importantes por serem antioxidantes, estimuladores das funções imunológicas e protetores de determinados tipos de câncer.

Portanto, o pêssego alia à sua beleza, ao sabor delicado e aroma agradável um alto valor nutritivo, ainda não devidamente conhecido.

A produção de pêssegos, no Brasil, é considerada suficiente para atender à atual demanda; entretanto, com a melhoria do poder aquisitivo e com investimentos em marketing e publicidade, certamente, em pouco tempo, a produção estaria muito aquém do necessário." [MRa 98]

Em 2000 verificamos a permanência na atividade de apenas 12 indústrias na região, porém, tendo sua capacidade instalada ampliada, com a renovação do parque de máquinas.

5.4 O pomar de pessegueiros

Durante várias reuniões com o Sr. Adelci, procuramos formalizar o que seria um pomar de pessegueiros. Deste processo surgiu a descrição textual que é citada abaixo.

Segundo [Fer 93], um pomar é um conjunto de plantas frutíferas.

O Sr. Adelci decidiu cultivar pessegueiros, definindo um pomar de pessegueiros como *"um conjunto de plantas frutíferas (pessegueiros), explorado economicamente"*.

Esse conjunto de plantas ficam distribuídas em lotes ou talhões.

Esses lotes podem ser identificados pela sua localização, área (mensurada em hectares), número de plantas, idade e um cultivar (também chamado por "variedade").

Cultivar é uma variedade de pêssego com características próprias. Dessas características próprias a que mais nos interessa para gestão do pomar é o **ciclo produtivo do cultivar**.

O ciclo produtivo do cultivar envolve uma coleção de fases onde são realizados uma série de "tratos culturais"[11].

Esses tratos culturais, através de tarefas, utilizam um ou mais recursos para sua realização.

Para facilidade de compreensão, planejamento e utilização, estes recursos são identificados, codificados, quantificados e, ainda, classificados.

Os recursos podem ser classificados como:

- **Humanos**: Os recursos humanos compreendem cada pessoa que trabalha no pomar (mão-de-obra utilizada na execução das tarefas).
- **Materiais**: Recursos materiais são identificados por cada máquina, implemento, utensílio, insumo, etc, que pode ser utilizados para realizar alguma tarefa no pomar.
- **Financeiros**: Recursos financeiros são representados pelos valores em moeda nacional envolvidos no processo.
- **Naturais**: Recursos naturais compreende os elementos que se encontram na natureza a disposição para uso no processo produtivo, como solo, água, plantas, etc.

[11] **Tratos Culturais** são atividades desenvolvidas no cultivo agrícola.

5.5 O ciclo produtivo de um pomar de pessegueiros

Conversando sobre o ciclo produtivo do pessegueiro e sobre atividades executadas no pomar, concluímos que o ciclo produtivo do pomar é composto das seguintes fases:

1. Dormência
2. Floração
3. Frutificação
4. Maturação
5. Colheita
6. Comercialização
7. Entresafra

Embora a época de início da dormência seja semelhante, independente do cultivar, a fase de floração e maturação diferem influindo diretamente na época de realização das outras fases. Sendo que podemos classificar, quanto a época de maturação, em cultivar de ciclo:

- **Precoce** quando ocorre a maturação até 25 de Novembro;
- **Médio** com maturação após 25 de Novembro até 25 de Dezembro e
- **Tardio** com maturação após 25 de Dezembro.

A seguir detalharei cada fase do ciclo produtivo do pomar.

5.5.1 A fase de dormência

A fase de Dormência, também conhecida por "repouso", corresponde ao período em que diminui a atividade vegetativa do pessegueiro com a conseqüente queda das folhas.

Na fase de Dormência desenrola-se os seguintes tratos culturais:

- **Roçada**: consiste numa limpeza (corte de pequenos arbustos, gramíneas, etc). Essa limpeza é feita utilizando-se roçadeira mecânica (tratorizada) ou manual. Para execução deste trato cultural utiliza-se recursos humanos e materiais (máquinas e implementos)

- **Tratamento de Inverno**: aplicação de produtos a base de óleo mineral, cobre, inseticida, etc, nas plantas em dormência. Essa aplicação é feita com uso de pulverizadores costal e tratorizado. Para execução deste trato cultural utiliza-se recursos humanos e materiais (máquinas, implementos, insumos e utensílios).

- **Poda Hibernal**: consiste no corte dos ramos indesejáveis para formação/condução da planta facilitando a aeração e penetração de luz no interior da mesma. Feito através do uso de tesouras de poda manual e serrote de poda. Para execução deste trato cultural utiliza-se recursos humanos e materiais (utensílios).

5.5.2 A fase de floração

Esta fase corresponde ao início da atividade vegetativa do pessegueiro com o desenvolvimento de botões florais, desabrojar das flores até a queda das pétalas.

A fase de Floração possui os seguintes tratos culturais:

- **Tratamento da Flor**: se dá desde o desenvolvimento do botão floral até a queda das pétalas, com produtos a base de fungicidas, inseticidas, etc. Aplicação feita com o uso de pulverizadores costal e tratorizados. Os recursos utilizados na execução deste trato cultural são os mesmo do Tratamento de Inverno da fase de Dormência.

- **Adubação de Base**: aplicação de adubo conforme orientação da análise de amostra de solo ou foliar, na projeção da copa das plantas, aplicado manualmente ou com o emprego de semeadeiras tratorizadas. Recursos humanos e materiais (máquinas, implementos e insumos) são utilizados na execução deste trato cultural.

5.5.3 A fase de frutificação

Esta fase inicia com a queda da "*corola*" até a completa formação do fruto.

Na fase de Frutificação são executados os seguintes tratos culturais:

- **Capina**: consiste na eliminação das ervas daninhas no interior das linhas do pomar. Pode ser manual (quando realizada com uso de enxadas), mecânica (quando realizada com capinadeiras, grades tratorizadas, etc) ou química (uso de herbicidas). Para aplicar o herbicida utiliza-se pulverizador tratorizado ou costal.

- **Tratamento dos Frutos**: controle de doenças e pragas com o uso de produtos químicos e/ou orgânicos (fungicidas, inseticidas, etc) que são aplicados com uso de pulverizadores (costal ou tratorizado). São utilizado os mesmo recursos do Tratamento de Inverno da fase de Dormência.

- **Raleio**: consiste na eliminação do excesso de frutos para obter maior tamanho e qualidade. Essa eliminação é feita manualmente. Retirando-se os frutos excedentes, defeituosos e mal localizados, operação feita antes do endurecimento do caroço. Somente recursos humanos são utilizados neste trato cultural.

- **Adubação de Cobertura**: distribuição de adubos nitrogenados durante o período vegetativo (aproximadamente 60 dias antes da colheita dependendo do cultivar). São utilizados os mesmos recursos da Adubação de Base da fase de Floração.

- **Poda Verde**: consiste na eliminação dos ramos mal localizados, facilitando a formação de ramos produtivos para a próxima safra e na formação de plantas novas. Na execução deste trato cultural são utilizados recursos humanos e materiais (utensílios).

- **Irrigação**: quando necessário e possível, efetua-se a rega do solo com o uso de equipamentos próprios para irrigação. Se for necessário bombear água usa-se motor elétrico ou a combustão (motor estacionário ou trator). Este trato cultural possui características ímpar, por exigir uma série de recursos de utilidade única a este fim. Pode ser efetuado de diversas formas (gotejamento, aspersão, sulcos, etc), utilizando recursos humanos e materiais (máquinas, implementos e utensílios).

5.5.4 A fase de maturação

Esta fase dá com o desenvolvimento da polpa do fruto, adquirindo textura, sabor e cor característica do cultivar.

A fase de Maturação compreende o seguinte trato cultural:

- **Tratamento dos Frutos**: Idem ao tratamento de frutos da fase de frutificação, com atenção especial ao combate da mosca dos frutos e podridão parda.

- **Irrigação**: idem ao trato cultural de Irrigação da fase de Frutificação.

5.5.5 A fase de colheita

Esta fase corresponde a coleta dos frutos com maturação completa.

Na fase de Colheita são realizadas as seguintes atividades:

- **Apanha dos Frutos**: recolhimento manual dos frutos que estão com maturação completa em cestos e/ou bolsas especiais. Utiliza recursos humanos e materiais do tipo utensílios.
- **Transporte interno**: consiste na distribuição de caixas vazias nos lotes e a recondução delas cheias para classificação em galpões. Utiliza recursos humanos e materiais do tipo máquina (trator e carreta) e utensílios (caixas).
- **Classificação dos Frutos**: consiste na classificação de acordo com o tamanho, sendo eliminados os defeituosos e avariados (amassados, podres, etc). A eliminação é feita manualmente, enquanto que a classificação é feita por mesas classificadoras elétricas ou não. Os frutos são acondicionados em caixas especiais com capacidade de 20kg para aguardar comercialização. Utiliza recursos humanos e materiais do tipo utensílios.

5.5.6 A fase de comercialização

A fase de Comercialização consiste na entrega dos frutos embalados para a indústria, ou para o comércio *in natura* em feiras e supermercados. A atividade executada nessa fase seria compostos pelo registro de venda de frutos com emissão de notas fiscais e o agendamento dos recebimentos, quando a prazo.

5.5.7 A fase de entresafra

Entresafra é a fase em que a planta, após a produção dos frutos recompõe-se, acumulando reservas em suas gemas, raízes e ramos para a produção na próxima safra.

A fase de Entresafra possui os seguintes tratos culturais:

- **Adubação de Cobertura**: distribuição de adubos nitrogenados durante o período vegetativo. Na execução deste trato cultural são empregados os mesmo recursos da Adubação de Base da fase de Floração.
- **Combate as Pragas**: combate principalmente a formiga, grafolita, etc. Aplicação de formicidas, inseticidas com uso de pulverizadores (costal ou tratorizado) ou manualmente. Utiliza-se recursos humanos e materiais (máquinas, implementos, insumos e utensílios).
- **Adubação Orgânica**: consiste no emprego de matéria orgânica (cama de aviário, esterco de curral, etc) no outono e no período de pré-dormência.

- **Poda Verde**: idem a atividade de Poda Verde da fase de Frutificação.
- **Calagem do Solo**: consiste no uso de calcário de acordo com as indicações da análise de solo.

5.6 A decomposição dos tratos culturais em tarefas

A execução de um trato cultural, é considerado uma ou mais tarefas que utilizam recursos no pomar.

Baseado na descrição detalhada dos tratos culturais no ciclo produtivo do pomar, observei que muitos recursos são utilizados repetidamente, em fases diferentes e com características diferentes, recebendo inclusive, denominações diferentes. Mas para efeito de gestão, notei a possibilidade de agrupar estes tratos culturais, facilitando deste modo, a compreensão, modelagem e futura implementação da aplicação proposta neste trabalho.

Note, por exemplo, que o trato cultural denominado "poda" pode ocorrer 2 ou mais vezes no ciclo produtivo: poda hibernal (fase de dormência) e poda verde (fase de frutificação e entresafra).

São efetuadas várias adubações no decorrer do ciclo produtivo (Adubação de Base, fase de Floração; Adubação de Cobertura, fases de Frutificação e Entresafra; Adubação Orgânica, fase de Entresafra). Convém ressaltar que o trato cultural de Calagem possui forma de aplicação semelhante a uma adubação, logo, para efeito de gestão do pomar, podemos tratar como tal.

Também é possível agrupar as várias pulverizações desenvolvidas nos tratamentos (de Inverno, fase de Dormência; da Flor, fase de Floração; do Fruto, fase de Maturação) e combate a pragas (em qualquer fase).

Nas fases de colheita e comercialização são executadas tarefas que também merecem gerenciamento, apesar de não fazerem parte do ciclo produtivo do pessegueiro (como classificação, transporte, etc).

Considerando a similaridade de recursos utilizados e objetivos de cada trato cultural foi possível identificar as seguintes tarefas:

1. Capina Manual: utiliza recursos humanos e utensílios;
2. Capina Mecânica: utiliza recursos humanos, máquinas e implementos;

3. Capina Química: utiliza recursos humanos, máquinas e insumos;
4. Roçada Manual: utiliza recursos humanos e utensílios;
5. Roçada Mecânica: utiliza recursos humanos, máquinas e implementos;
6. Tratamento Manual: utiliza recursos humanos e utensílios;
7. Tratamento Mecânico: utiliza recursos humanos, máquinas, insumos e implementos;
8. Adubação Manual: utiliza recursos humanos, insumos e utensílios;
9. Adubação Mecânica: utiliza recursos humanos, máquinas, insumos e implementos;
10. Combate a Pragas Manual: utiliza recursos humanos, insumos e utensílios;
11. Combate a Pragas Mecânico: utiliza recursos humanos, máquinas, insumos e implementos;
12. Poda: utiliza recursos humanos e utensílios;
13. Raleio: utiliza recursos humanos;
14. Irrigação: utiliza recursos humanos, máquinas, utensílios e implementos;
15. Apanha: utiliza recursos humanos, máquinas, utensílios e implementos;
16. Transporte: utiliza recursos humanos, máquinas e implementos;
17. Classificação Manual: utiliza recursos humanos e utensílios;
18. Classificação Mecânica: utiliza recursos humanos e máquinas;
19. Comercialização: utiliza recursos financeiros;

5.7 A gestão completa do pomar

Até o presente momento focalizamos apenas o ciclo produtivo do pomar. Convém lembrar que a gestão completa abrange mais do que isso, como por exemplo:

- Precipitação pluviométrica ocorrida no pomar, para determinar a necessidade de irrigação nos lotes;

- Registro patrimonial: bens móveis (máquinas, implementos, ferramentas e/ou utensílios de trabalho manual), bens imóveis (terras, pomares, cercas, represas, casas, galpões), cálculo de depreciação de máquinas e equipamentos.
- Controle de Estoque: relacionando estoque de insumos (fertilizantes, inseticidas, fungicidas, formicidas, óleo diesel, lubrificantes, herbicidas, etc), peças para reposição e produtos para comercialização.
- Fluxo de Caixa: contabilizando entradas e saídas, contas a pagar e receber.
- Folha de pagamento: relatório de pagamentos mensais a empregados e encargos sociais.

5.8 Conclusão do capítulo

Dado a complexidade do ambiente a ser modelado e a restrição de tempo disponível para execução deste trabalho, decidi modelar e implementar a gestão do ciclo produtivo, deixando os tópicos da seção acima para um trabalho futuro.

Um fato interessante que merece ser ressaltado, é que a confecção deste capítulo propiciou ao proprietário do pomar momentos onde ele pode refletir profundamente sobre detalhes do seu negócio, normalmente ele não encontra “tempo” para fazer isso.

6 ELABORAÇÃO

6.1 Introdução

Nesta fase do processo de desenvolvimento do software, são identificados os riscos do projeto.

Conforme a classificação proposta na Seção 4.4, os riscos de projeto são:

- **Riscos de Requisitos**, compreende uma análise de requisitos apoiada principalmente em casos de uso e num modelo de domínio conceitual.
- **Riscos Tecnológicos**, escolhe-se as ferramentas que serão utilizadas no desenvolvimento do projeto.
- **Riscos de Especialização**, seleciona-se a mão-de-obra para trabalhar no projeto.

6.2 A gestão do ciclo produtivo de um pomar de pessegueiros modelada em casos de uso de negócio

> "A modelagem de casos de uso é uma técnica utilizada para descrever a funcionalidade de um sistema através de atores externos interagindo com casos de uso. Atores representam um papel e iniciam o caso de uso que, por sua vez, deve entregar um valor tangível de retorno ao ator. Atores e casos de uso estão conectados através de associações e podem ter relacionamentos." [Fur 98]

Casos de uso também fornecem a base da comunicação entre clientes e desenvolvedores, porque servem para captar as interações que os usuários tem com o sistema afim de atingir um objetivo.

Conforme [Fow 00a] os casos de uso quando centrados nas interações do usuário com o software são chamados de "caso de uso de sistema", enquanto que, quando centrados nas respostas do sistema ao cliente ou a um evento externo são chamados de "casos de uso de negócios".

Um detalhe interessante é que casos de uso não precisam ser detalhados, um texto curto (de um a três parágrafos) que seja suficientemente específico para que os usuários entendam a idéia básica e que os desenvolvedores tenham uma visão do que está escondido por dentro é o ideal. [Fow 00a]

6.2.1 Sistema de gestão do ciclo produtivo para pomares de pessegueiro

A gestão do ciclo produtivo se dá através do planejamento, registro de tarefas executadas e apuração de resultados sobre os recursos disponíveis no pomar. Para maior reutilização e flexibilidade do sistema, o cadastramento de recursos deve ser feito pelos próprios usuários. O sistema deve ainda auxiliar no planejamento para cada fase do ciclo produtivo através da geração de estimativas de custos com base em gestões anteriores, corrigidas monetariamente e/ou pela variação dos preços no mercado. O registro de tarefas se dá por ocasião da execução das mesmas para realizar cada trato cultural. Enquanto que a apuração de resultados é realizada automaticamente pelo sistema, através da mineração de informações ou comparando dados estimados com aqueles realmente executados.

O desenho do caso de uso de negócios do Sistema de Gestão do Pomar se deu, primeiramente, identificando os atores que interagem com o sistema. Foi identificado dois atores: o proprietário e o capataz do pomar. Definiu-se que o proprietário teria acesso irrestrito ao sistema e caberia a ele a manutenção dos processos críticos, enquanto que, o capataz teria acesso restrito ao cadastramento de recursos e registro de execução das tarefas. O diagrama está representado na Figura 2.

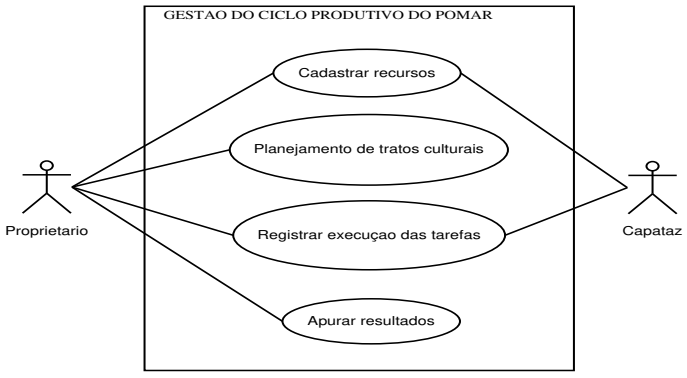

Figura 2: Caso de uso de negócios da aplicação de gestão do ciclo produtivo para pomares de pessegueiro

6.2.2 Caso de uso: Cadastrar recursos

A identificação e classificação dos recursos que o pomar dispõe para realização das tarefas é uma tarefa crucial para uma eficaz gestão. Visto que a estimativa, registro e apuração de resultados é baseada nos recursos cadastrados.

Este cadastramento deve ser flexível o suficiente para permitir que o usuário do sistema possa fazê-lo e extendê-lo, se necessário.

Como citado anteriormente, os recursos podem ser humanos, materiais, naturais e financeiros.

6.2.3 Caso de uso: Planejamento de tarefas

O planejamento para execução das tarefas tem tamanha importância na gestão do pomar. Com o planejamento é possível estimar a necessidade de recursos financeiros, criar um cronograma de tarefas através do agendamento das mesmas e, ainda, racionalizar a utilização de recursos.

Criado o cronograma, o sistema deve confeccionar as planilhas para acompanhamento, a campo, da execução das tarefas.

6.2.4 Caso de uso: Registro das tarefas executadas

O registro das tarefas executas no pomar se faz necessário para possibilitar a apuração de resultados, servir de base para estimativas futuras e manter um registro histórico dos acontecimentos. Este registro deve ser feito diariamente, relacionando todas tarefas que foram executadas em cada lote do pomar.

6.2.5 Caso de uso: Apuração de resultados

O sistema deve responder ao final do ciclo produtivo, no mínimo, as seguintes perguntas:

- Quanto foi gasto em transporte, empregados, encargos sociais, juros, reforma de máquinas e implementos?
- Quanto foi o total colhido por ciclo, cultivar e lote?
- Relatório de vendas da safra (para quem foi vendido, data de entrega, por qual valor e qual é a previsão de recebimentos)
- Qual foi o resultado final por ciclo, lote e cultivar?

6.3 Modelo de domínio da aplicação de gestão para pomares de pessegueiro

> "Modelo de domínio descreve qualquer modelo, cujo sujeito primário seja o mundo que o sistema de computação está suportando, qualquer que seja o estado do processo de desenvolvimento em que se esteja." [Fow 00a]

Neste trabalho a técnica utilizada para gerar o modelo de domínio foi o diagrama de classes, feito a partir de uma perspectiva conceitual. Saliento que foi utilizado uma notação mínima. Já que não procurou-se captar cada detalhe, este diagrama serve apenas para dar uma idéia global do sistema que está sendo construído.

Baseado na descrição textual do capítulo 4 e em primitivas de modelagem orientada a objetos, foi desenhado o modelo de domínio da aplicação na Figura 3.

6.4 Plataforma de hardware

No desenvolvimento desse projeto, utilizarei a plataforma de hardware PC, baseada na arquitetura Intel x86, devido ao seu custo inferior comparado ao de outras plataformas, facilidade de manutenção/suporte, abundância de software livre e popularidade.

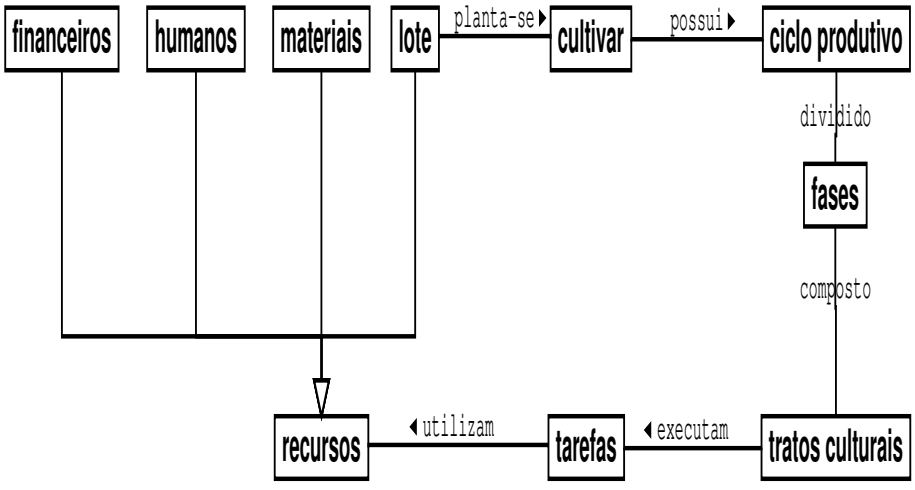

Figura 3: Modelo de domínio da aplicação de gestão para pomares de pessegueiro.

6.5 Plataforma de software

Nesta seção tratarei a respeito dos componentes de software que pretendo utilizar no desenvolvimento/implementação deste trabalho.

- Tecnologia de Sistema Operacional
  GNU/Linux [LNX 00] me parece uma opção acertada por possuir vasta documentação disponível, ser estável e servir de plataforma de desenvolvimento para a maioria das outras ferramentas que utilizarei neste projeto. A distribuição utilizada será Red Hat [RED 00] porque é a distribuição que tenho mais experiência.

- Tecnologia de Banco de dados
  PostgreSQL [PGS 00], por ser o único banco de dados híbrido livre, suportando parcialmente a orientação a objetos.

- Tecnologia de Servidor web
  Apache é o servidor escolhido. Robusto, veloz, extensível (suporta PHP) e de fácil configuração, o tornam uma boa opção para utilização neste projeto.

- Tecnologia de Geração dinâmica de páginas web
  PHP [PHP 00] é a ferramenta eleita por possuir suporte a orientação a objetos, PostgreSQL, Apache e por ser desenvolvido primeiramente sobre GNU/Linux.

- Tecnologia para Desenho de modelos
  O software Dia [DIA 00] foi a melhor opção encontrada, já que suporta a notação Unified Modeling Language (UML[12]) e é um software livre.

6.6 Mão-de-obra para desenvolvimento/implementação deste trabalho

No desenvolvimento deste trabalho estou procurando consultoria de pessoas que conheçam profundamente o assunto que estou enfrentando dificuldades de aprendizado.

Na implementação, a princípio, não será utilizado qualquer mão-de-obra externa.

[12] **Unified Modeling Language** é um padrão notacional para aplicações orientada a objetos.

7 PROJETO

7.1 Introdução

É nesta fase do processo de desenvolvimento que a aplicação vai começar a ser construída e implementada em fases iterativas e incrementais.

Primeiramente serão desenhados os casos de uso de sistema da aplicação, baseado no caso de uso de negócio apresentado na fase de elaboração. Feito isso, partirei para a execução das fases utilizando a OOHDM.

A próxima seção aborda rapidamente o método de projeto que utilizarei na construção deste sistema.

Na seção subseqüente teremos o projeto da aplicação proposta com este trabalho.

7.2 Object-Oriented Hypermedia Design Method

Como foi colocado na Seção 4.3.1, a OOHDM é um método de projeto para aplicações hipermídia que divide a construção do sistema em 4 atividades.

A seguir, explanarei de forma resumida, cada uma das atividades desenvolvidas em um projeto OOHDM.

7.2.1 Modelagem conceitual

A modelagem conceitual compreende a primeira atividade do método OOHDM, sendo responsável pela análise do domínio da aplicação, englobando assim todo o universo de informações relevantes para a aplicação em questão. [SVi 99]

Durante a modelagem conceitual é construído um modelo de domínio da aplicação utilizando princípios de modelagem orientada a objetos. A OOHDM não prescreve nenhuma notação em particular para produzir este esquema conceitual, podendo ser empregado UML, OMT, OOSE ou qualquer outra. Neste trabalho utilizarei a UML por acreditar que, atualmente, esta linguagem de modelagem é de fato um padrão notacional para modelagem de aplicações orientada a objetos.

A finalidade desta atividade é capturar a semântica do domínio da aplicação. Para isso, utiliza-se formalismos de modelagem orientada a objetos como classes, relacionamentos e casos de uso. Produz-se um esquema conceitual composto de classes, subsistemas, relacionamentos e perspectivas de atributos baseados em mecanismos como classificação, agregação, generalização e especialização.

As informações necessárias para criação do esquema conceitual foram extraídas dos casos de uso definidos na seção anterior. Para facilitar a tarefa de mapeamento de casos de uso para o esquema conceitual, existe uma ferramenta chamada User Interaction Diagram (UID) que descreve a troca de informações entre o sistema e o usuário em um alto nível de abstração. Maiores informações sobre UID são descritas em [VSS 00].

Devido a restrição de tempo para desenvolvimento deste trabalho, não foi possível aplicar a ferramenta de UID nos casos de uso desta aplicação, sendo que o mapeamento se deu de maneira empírica e intuitiva.

7.2.2 Projeto navegacional

Segundo [Ros 96] "Os aplicativos de hipermídia são projetados para efetuar navegação através de um espaço de informações".

Em OOHDM, uma aplicação é vista como um conjunto de visões navegacionais **derivadas** do modelo conceitual construído na atividade anterior. "Esta é uma das maiores inovações da OOHDM, já que reconhece objetos que o usuário navega **não** como objetos conceituais, mas sim, outros tipos de objetos que são "construídos" de um ou mais objetos conceituais" [SRo 98]. Logo, o modelo conceitual funcionará como um repositório compartilhado de modelagem, a partir do qual construiremos diferentes visões navegacionais do domínio do problema, baseado nos usuários que utilizarão a aplicação e as tarefas que realizarão nele [Ros 96].

> "Cada visão define um conjunto de contextos e classes navegacionais. Os contextos navegacionais expressam a estrutura navegacional geral do aplicativo hipermídia, enquanto as classes navegacionais, como os nós e os elos, especificam os objetos que serão vistos pelo usuário.
>
> Os nós representam 'janelas' lógicas em classes definidas no esquema conceitual através de uma mapeamento que fornece um 'corte e colagem' de atributos conceituais semelhantes às visões dos objetos.
>
> Os contextos navegacionais são um conjunto de nós, elos e outros contextos navegacionais (aninhados) que auxiliam na organização dos objetos navegacionais, fornecendo espaços de navegação consistentes e, deste modo, diminuindo as chances do usuário 'perder-se no hiperespaço"' [Ros 96].

O esquema de classes navegacionais da aplicação

> "Um dos produtos da atividade de projeto de navegação é o esquema navegacional, construído sobre nós e elos. As classes navegacionais definidas nesta atividade são especificadas como especializações de um conjunto de classes básicas que definem a semântica dos objetos navegacionais" [Ros 96].

Em OOHDM há um conjunto de tipos pré-definidos de classes navegacionais como: nós, links, âncoras e estruturas de acesso. A semântica dos nós, links e âncoras são as mesmas utilizadas em aplicações hipermídia usuais, enquanto que estruturas de acesso representam as possíveis maneiras de acessar nós.

O diagrama de contextos

> "Um contexto de navegação é um conjunto de objetos (nós, elos e, recursivamente, outros contextos navegacionais) que estão relacionados de acordo com algum aspecto (ex.: os nós do mesmo tipo que apresentam um atributo com o mesmo conteúdo; os nós do mesmo tipo que apresentam um relacionamento com um mesmo nó de outro tipo; os nós de tipos diferentes que apresentam uma característica em comum)." [SVi 99]

Os contextos de navegação da aplicação ficam documentados em cartões próprios e são representados graficamente no diagrama de contextos.

Os contextos possuem formas pré-definidas de acesso e de navegação entre seus elementos, bem como restrições de acesso a usuários.

Há um tipo de contexto de navegação chamado **estrutura de acesso** que age como índice para outros contextos. Em cada estrutura de acesso são especificados atributos "seletores", que são os objetos que ativam a navegação a outros contextos.

Classes em contexto

Classes em contexto são artifícios utilizados para acrescentar características a nós conforme o contexto em que estão sendo acessados. Estas classes em contexto só se fazem necessárias caso o nó deva possuir uma aparência diferente e âncoras distintas no contexto em questão.

Quando um nó possui atributos com múltiplas perspectivas, é um indício para utilização de classes em contexto.

Na representação de classes em contexto utiliza-se a mesma notação de classes da UML, diferenciando-se apenas pela adição de um compartimento onde é especificado os contextos nos quais a classe em contexto participa.

7.2.3 Projeto de interface abstrata

A atividade de Projeto de Interface Abstrata compreende uma abordagem para especificar a interface de usuário para aplicações hipermídia. Esta atividade deve ser desenvolvida antes de iniciar a atividade de implementação, de forma que seja independente do ambiente onde se pretende efetivamente construir o aplicativo, sempre procurando manter uma coerência com o ambiente de implementação (por exemplo, de nada adianta especificar algo que o ambiente de implementação não suporta).

Na especificação da interface abstrata utilizamos o conceito de Abstract Data Views (ADVs). ADVs são objetos que especificam a organização e procedimentos da interface, são abstratos porque sua real aparência física é feita na fase de implementação. ADVs possuem um estado e uma interface, onde a interface é utilizada para entrada e/ou saída de eventos externos gerados pelo usuário.

OOHDM também utiliza Abstract Data Objects (ADOs), que similarmente aos ADVs são objetos, só que não suportam eventos externos gerados pelo usuário.

> "Em uma típica aplicação utilizando ADVs, temos um conjunto de ADOs gerenciando estruturas de dados e de controle, e um conjunto de objetos de interface (instâncias de ADVs) gerenciando aspectos de interface da aplicação, como entrada de dados do usuário e saída de dados do sistema para o usuário." [RSL et al 95]

"No contexto da OOHDM, os objetos navegacionais (nós, elos, classes em contexto e estruturas de acesso) agirão como ADOs e estarão associados a ADVs que serão usados para especificar a sua aparência para o usuário." [RSL et al 95]

Utilizamos também uma variável chamada "ContextoPerceptivo" para indicar os objetos que são perceptíveis num dado momento. Todo e qualquer objeto que vier a se tornar visível é acrescido a "ContextoPerceptivo", enquanto que todos aqueles que forem excluídos da variável deixam de ser perceptíveis.

Diagramas de Configuração

Nesta fase constrói-se Diagramas de Configuração, que são utilizados para representar os relacionamentos entre os objetos de interface e os objetos navegacionais. [SVi 99]

Além de representarem os objetos de interface (ADVs), os diagramas de configuração especificam outras características da interface abstrata da aplicação, que conforme [SVi 99] são:

- os eventos externos iniciados pelo usuário e que serão manipulados pelos ADVs;
- os relacionamentos estruturais entre os ADVs;
- os relacionamentos entre os ADVs (que representam os objetos de interface) e os ADOs (que representam os objetos navegacionais).

ADVcharts

Outro diagrama também é construído nesta fase, os ADVcharts. Estes procuram apresentar as transformações ocorridas nos ADVs, sendo que cada ADVchart está relacionado com uma tela da interface.

Enquanto que os Diagramas de Configuração representam a estrutura estática da interface, utilizamos ADVcharts para representar a estrutura dinâmica.

7.2.4 Implementação

A atividade de Implementação é a última etapa de uma iteração em OOHDM, sendo que agora é o momento de transformarmos os artefatos abstratos produzidos nas fases anteriores em artefatos concretos de software.

Para feitura deste mapeamento utilizaria-se ferramentas que suportassem a notação OOHDM, bem como um framework[13] que possibilitasse a implementação de uma especificação OOHDM da maneira menos traumática possível.

Infelizmente não há ferramentas para desenho dos diagramas, fazendo necessário a adaptação das ferramentas existentes a notação da metodologia.

Quanto aos frameworks, existem atualmente duas opções que abordo brevemente a seguir.

- **OOHDM-Java**
  [Piz 98] propõe um framework em Java para implementação na WWW de aplicações hipermídia modeladas com OOHDM.

[13] Segundo [Piz 98] "um **framework** é o esqueleto de uma aplicação que pode ser customizada por um desenvolvedor de aplicação".

O framework é implementado com uma biblioteca de classes em Java, que representa as primitivas OOHDM na web.
OOHDM-Java é considerado um framework caixa-branca [14] , sendo portanto, de difícil aprendizado. Infelizmente, não há tempo para estudar todo o framework e ainda implementar um protótipo.
Apesar dos requisitos de sistema especificados na dissertação relacionarem apenas software proprietário, acredito que seja possível obter plataforma igual baseada em software livre.

- **OOHDM-WEB**
  [MOU 98] propõe um ambiente de desenvolvimento para dar suporte às aplicações WWW modeladas com OOHDM. Esse ambiente faz uso da linguagem script CGILua e o pacote DBLua (para acesso a bases de dados) e permite a implementação de script CGI, que geram páginas dinamicamente, baseando-se em templates pré-definidos e nos dados contidos em um banco de dados.
  Uma característica interessante deste ambiente é a possibilidade de compilação, gerando uma aplicação estática, eliminando desta maneira a necessidade de manter um banco de dados e o "overhead" causado pela geração dinâmica das páginas.
  Infelizmente não há implementação deste ambiente de desenvolvimento para rodar sobre software livre.

Acredito ser esta a atividade mais delicada da OOHDM na atualidade, a falta de um framework para implementação faz o desenvolvedor pensar duas vezes antes de utilizar um método, porque é sabido que todo o trabalho feito nas atividades anteriores não é mapeado de forma direta para artefatos concretos de software.

Como não existe um framework para PHP, decidi criar um para poder implementar o protótipo deste trabalho.

7.3 Casos de uso de sistema

Nesta seção vamos refinar os casos de uso de negócio propostos na fase anterior em casos de uso de sistema.

[14] "Um framework é considerado **caixa-branca** quando é necessário que programadores estendam o seu projeto e sua implementação, pelo menos até um certo grau de detalhe, para utilizá-lo. Por esta razão, esses tipos de frameworks geralmente são mais difíceis de aprender e usar." [Piz 98]

7.4 A construção do caso de uso “Cadastrar recursos”

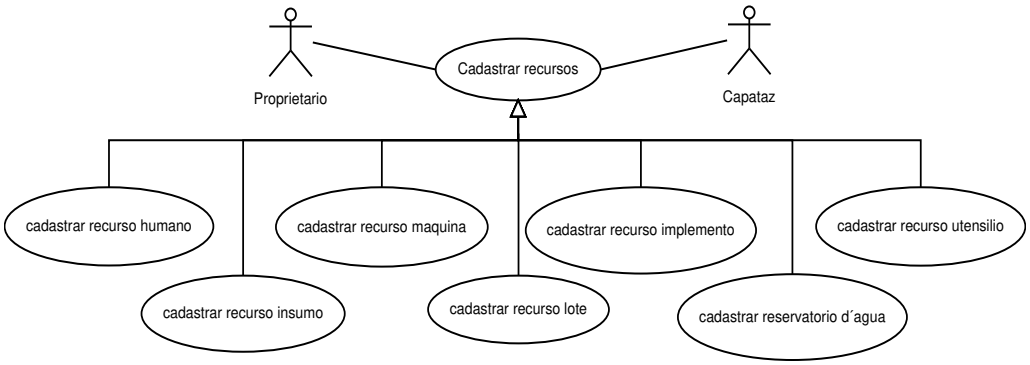

Figura 4: Caso de uso de “Cadastrar recursos”

Nesta seção será construído o caso de uso “Cadastrar recursos”, sendo que este caso de uso é utilizado tanto pelo proprietário como pelo capataz do pomar. Na Figura 4 procurei diagramar o caso de uso proposto nesta seção.

Os “casos de uso de sistema” (confesso que a especificação mais parece um dicionário de dados do que a formalização de casos de uso) estão textualizados abaixo:

- **Cadastrar Recursos Humanos**
  O cadastro da mão-de-obra empregada no pomar, deve ser feita cadastrando dados como:
  - Nome completo do empregado
  - Apelido
  - CPF
  - RG
  - Estado civil
  - Sexo
  - Forma de Pagamento (diarista, mensalista, empreiteiro, horista)
  - Salário
  - Dependentes (grau de dependência, nome e data de nascimento)
  - Observações (data, comentário)

- **Cadastrar Recursos Máquinas**
  O cadastro de recursos materiais da classe máquinas agrícolas, compreende os aparelhos capazes de produzir energia que pode ser utilizada na execução de

tarefas agrícolas.
Ex.: trator, moto-serra, motor estacionário (elétrico ou a combustão), etc.
O cadastro desses aparelhos deve conter dados como:

  - Código
  - Fabricante
  - Modelo
  - Número de Série
  - Cor
  - Número do Chassi
  - Ano de Fabricação
  - Classe
    * Trator
    * Moto-serra
    * motor estacionário

- **Cadastrar Recursos Implementos**
  Todos aqueles aparelhos que acoplados a uma máquina são capazes de executar tarefas agrícolas devem ser cadastrados como recursos materiais da classe implemento.
  Ex.: pulverizador, semeadeira, arado, carreta, etc.
  Desses implementos são armazenados os seguintes dados:

  - Código
  - Marca
  - Ano de Fabricação
  - Número de Fábrica
  - Classe
    * grade
      · número de discos
    * arado
      · número de discos
    * semeadeira
      · forma de distribuição: a lanço, 2, 3, 4, 5, 20 linhas, etc.
    * pulverizador

            - turbina, barra, pistola, etc.
        - carreta
            - número de rodas
            - capacidade (em toneladas)

- **Cadastrar Recursos Insumos**
  Os insumos compreendem aqueles produtos que se usa no desenvolver de um trato cultural no período de uma safra.
  Ex.: fungicidas, inseticidas, adubo, etc.
  Os seguintes dados fazem parte do cadastro de insumos:
  - Código
  - Nome do Produto
  - Tipo
    - Fungicida
    - Inseticida
    - Herbicida
    - Adubo (Fórmula)
  - Fabricante
  - Unidade

- **Cadastrar Recursos Utensílios**
  O cadastro dos recursos materiais da classe utensílios se dá relacionando todas pequenas ferramentas manuais utilizadas na execução de tarefas no pomar.
  Ex.: tesouras, serrotes, pulverizadores costal, enxada, etc.
  Os seguintes dados fazem parte do cadastro de utensílios:
  - Nome do Utensílio
  - Marca
  - Modelo
  - Classe (tesoura de poda, serrote de poda, serrote comum, enxada, etc)

- **Cadastrar Recurso Lote**
  O cadastro dos lotes é feito agrupado os talhões ou lavouras por cultivar e época de implantação.
  Para cadastro dos lotes considera-se os seguintes dados:
  - Área (em hectares)

  - Número de plantas
  - Localização
  - Cultivar

- **Cadastrar Recurso Reservatório d'Água**
  Dos reservatórios d'água instalados no pomar, cadastra-se os seguintes dados.
  - Localização
  - Capacidade (em m^3)
  - Classe
    * Açude
    * Barragem
    * Tanque
    * Poço Artesiano

7.4.1 Modelagem conceitual

O esquema conceitual do caso de uso de negócio "Cadastrar recursos" está representado na Figura 5.

7.4.2 Projeto navegacional

O produto da atividade de projeto navegacional do caso de uso "Cadastrar recursos" está documentado através de seu esquema navegacional (Figura 6) e diagrama de contextos (Figura 7).

7.4.3 Projeto de interface abstrata

A definição dos ADVs principais da aplicação, bem como, os ADVs do caso de uso em estudo estão representados na Figura 8.

Convém lembrar que alguns ADVs foram omitidos, dado que pretendo ilustrar com a Figura 8 a forma como é criado os ADVs e não especificar todas possibilidades.

7.4.4 Implementação

Neste trabalho implementarei um protótipo do cadastro de recursos humanos deste caso de uso, devido a restrição de tempo.

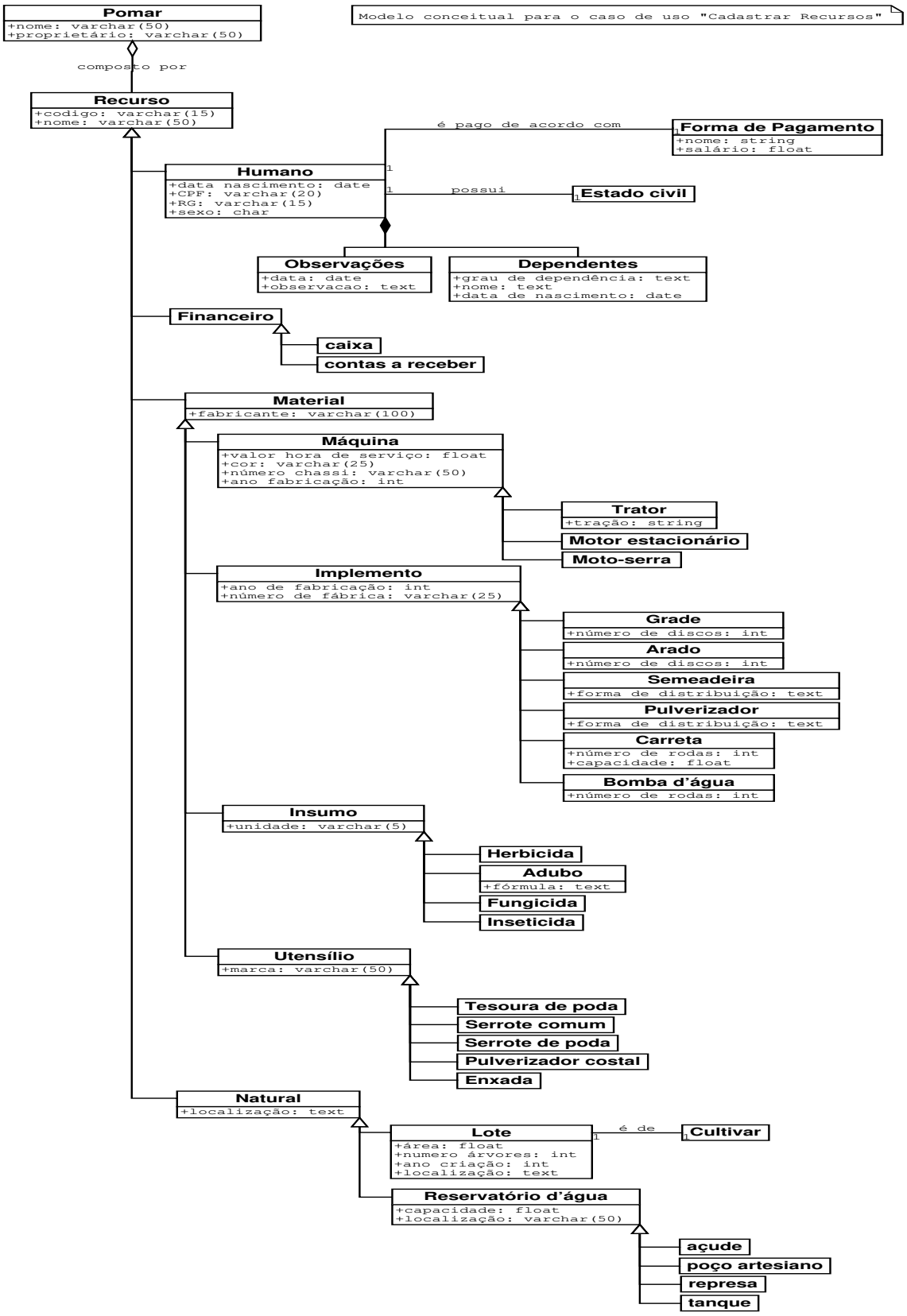

Figura 5: Esquema conceitual do caso de uso de “Cadastrar recursos”

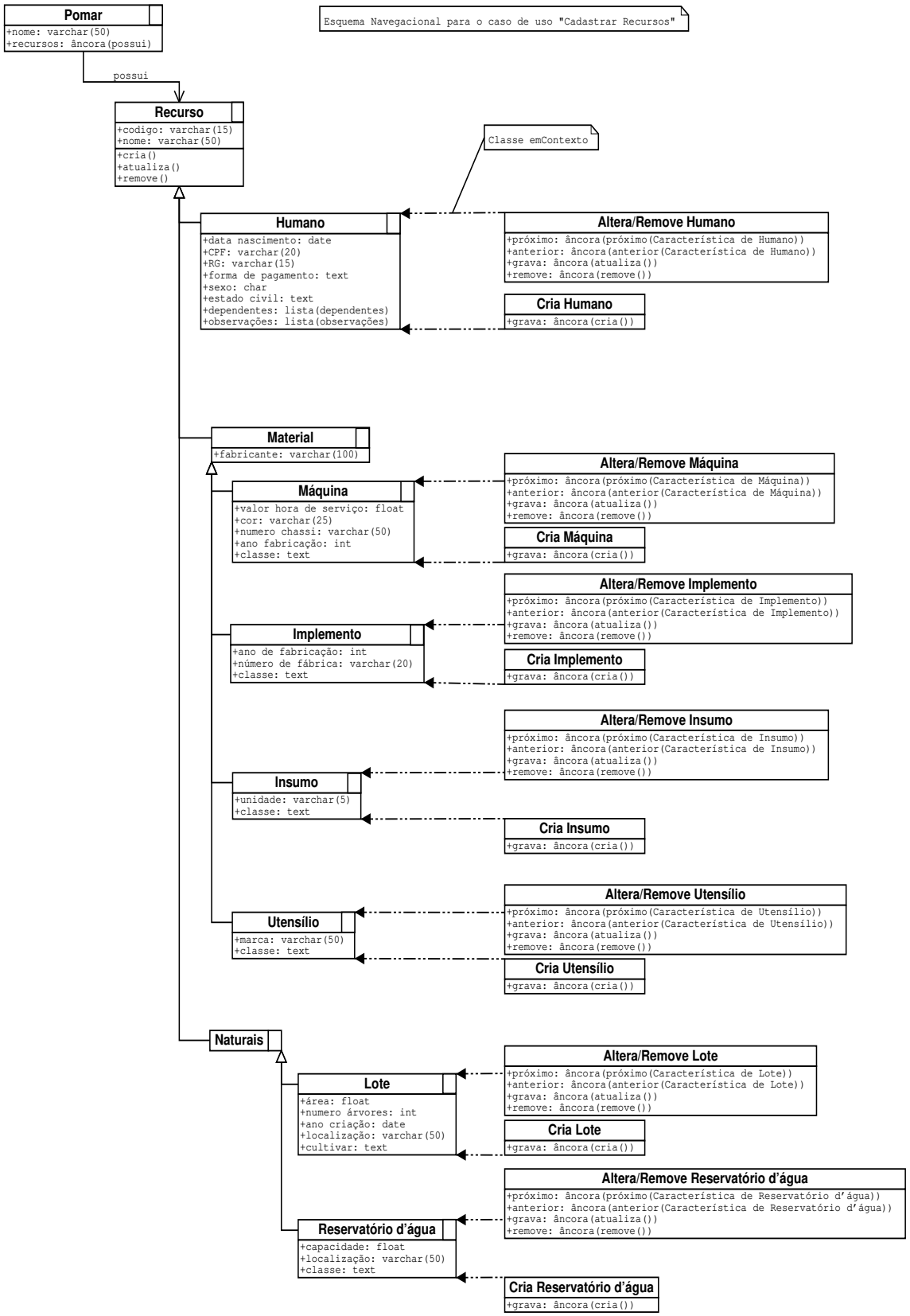

Figura 6: Esquema navegacional do caso de uso de “Cadastrar recursos”

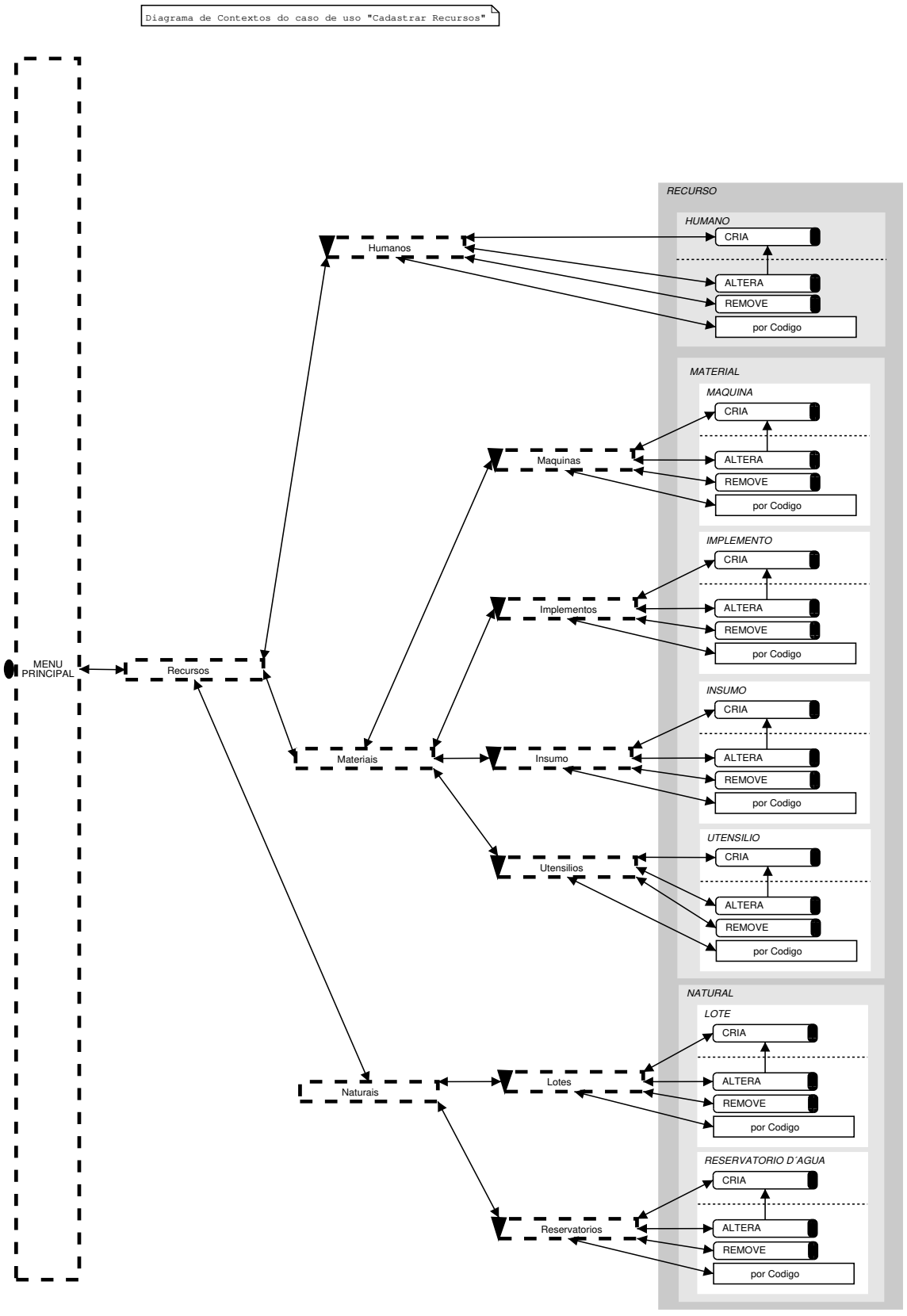

Figura 7: Diagrama de Contextos do caso de uso de “Cadastrar recursos”

Figura 8: Abstract Data Views do caso de uso “Cadastrar recursos”

A plataforma de software utilizada foi:

- Distribuição Red Hat Linux 6.2
- PostgreSQL 7.0.2
- PHP 4.0.1pl3
- Navegador Mozilla M18

A plataforma de hardware utilizada no desenvolvimento foi:

- Processador: AMD K6-II 333 Mhz
- Memória RAM: 128 Mb
- HD: Maxtor 91303D (13Gb) e QUANTUM FIREBALLlct10 (10Gb)
- Drive CD-ROM: Hewlett-Packard CD-Writer Plus 8100
- Modem: USRobotics Sportster Voice 33.6 External
- Placa de Vídeo: Diamond Stealth 6400 2 Mb (Chipset S3)
- Monitor de Vídeo: LG Studioworks 550M

O diagrama de classes produzido na atividade de Modelagem Conceitual foi traduzido para cláusulas SQL. Já que o PostgreSQL possui herança, utilizei esse recurso quando necessário. As relações entre classes foram mapeadas para campos que contém o Object Identification Number (OID[15]) do objeto ao qual se relaciona. Maiores informações sobre como mapear o esquema conceitual para uma base de dados podem ser encontradas em [SVi 99].

No Projeto Navegacional, o framework suporta a definição de "nós" para a implementação do Esquema de Classes Navegacionais; Contextos e Estruturas de Acesso (índices simples e dinâmicos) para implementação do Esquema de Contextos Navegacionais.

[15]**Object Identification Number**: conforme [MOM 00] cada registro no banco de dados PostgreSQL recebe um identificador único chamado OID.

Classes ADV definidas no framework dão suporte a implementação do Projeto de Interface Abstrata, sendo que as metáforas de interface são definidas em HTML. Apesar de não ter especificado *ADVcharts*[16] neste trabalho, foi necessário implementar o comportamento da interface em PHP (dentro dos Contextos e Estruturas de Acesso) e Java Script (dentro dos ADVs).

O framework não suporta completamente a OOHDM, até mesmo porque não é o objeto de estudo primário deste trabalho. Baseado nas definições encontradas em [MOU 98] considero o framework desenvolvido como sendo do tipo "caixa-branca", exigindo bom conhecimento da estrutura interna do framework para utilização do mesmo.

A Figura 9 apresenta a interface para usuário do ADV LOGIN. Não esqueça que os ADVs são aninhados, logo, o ADV LOGIN está contido no ADV APLICAÇÃO.

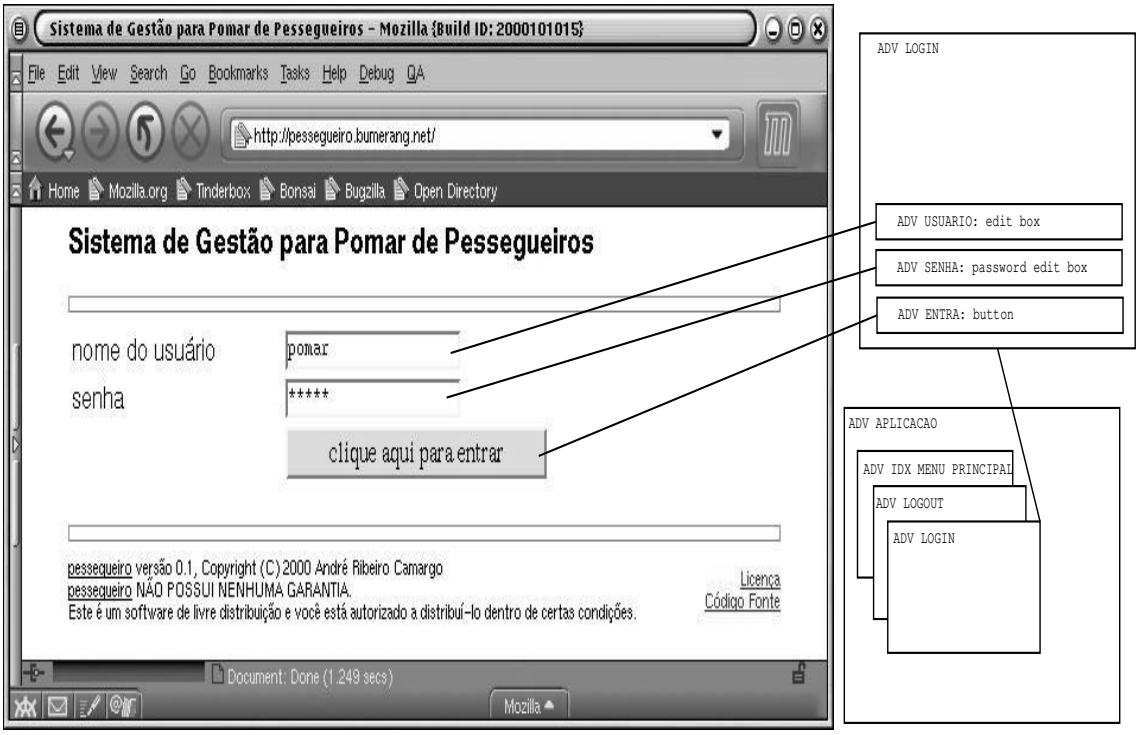

Figura 9: Interface para o usuário do ADV LOGIN

A Figura 10 apresenta a interface para usuário do ADV IDX MENU PRINCIPAL.

A interface para o usuário do ADV CTX CRIA HUMANO é apresentado na Figura 11.

[16] **ADVcharts** são diagramas que buscam formalizar o comportamento da interface da aplicação frente a estímulos externos (como cliques do mouse, etc.) Maiores informações podem ser obtidas na seção 7.2.3 e [RSL et al 95].

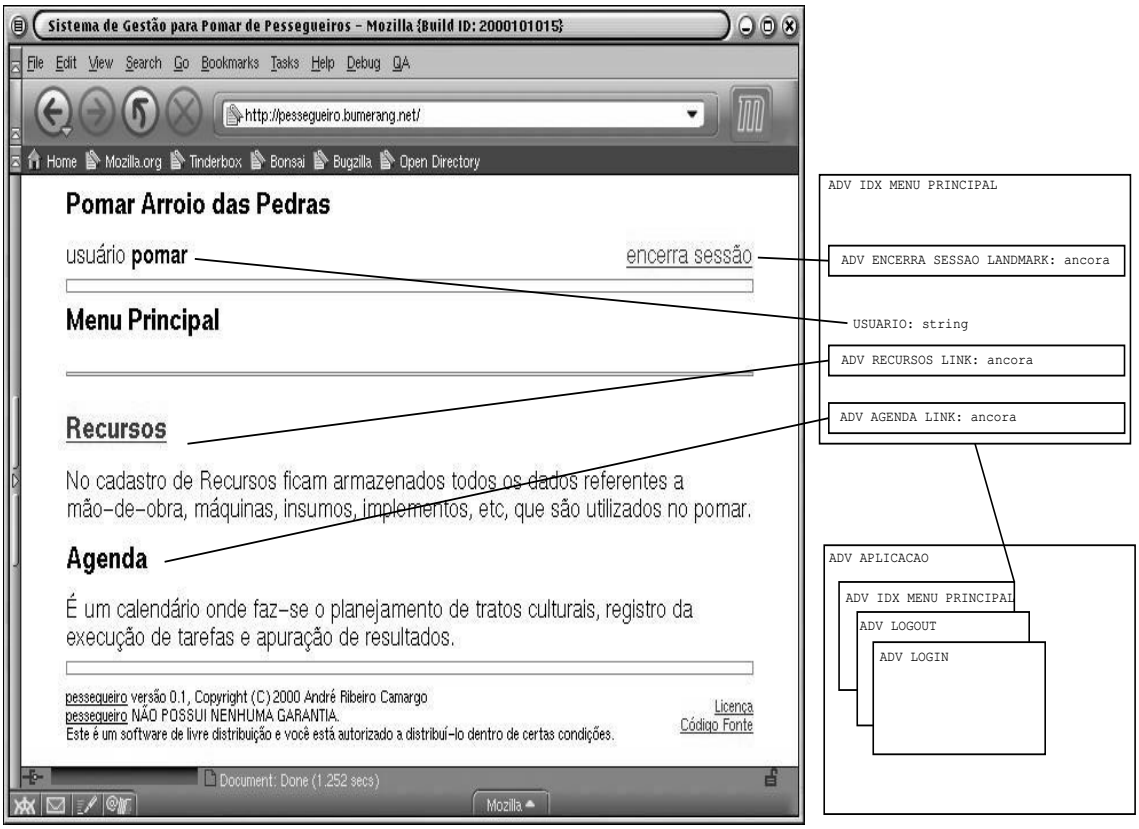

Figura 10: Interface para o usuário do ADV IDX MENU PRINCIPAL

Convém lembrar que os ADVs são objetos abstratos, sendo que a especificação de sua interface (em HTML) é armazenada em arquivos separados que são chamados de "metáforas de interface". O desenvolvedor deve carregar a metáfora do ADV em questão, instanciar os ADVs aninhados e tornar perceptíveis aqueles que o contexto da aplicação requere. Feito isso, o framework substitui as referências dos ADVs aninhados por suas respectivas metáforas.

A seguir apresento a especificação em HTML da metáfora de interface do ADV HUMANO (este ADV é utilizado sempre quando se deseja mostrar todos os dados sobre um Recurso Humano). Note que há ADVs aninhados (cujo nome está entre um par de pontos de interrogação). O resultado visual desta especificação é mostrado na figura 12.

Note que como a OOHDM pede, há uma separação entre a parte lógica e a parte visual da aplicação. A implementação deste modelo em PHP orientado a objetos, foi muito prejudicado devido a má implementação de polimorfismo nesta ferramenta. Acredito que a implementação com Java Server Pages seja mais tranqüila, até mesmo porque, JSP trabalha com esse conceito de separar a apresentação da implementação.

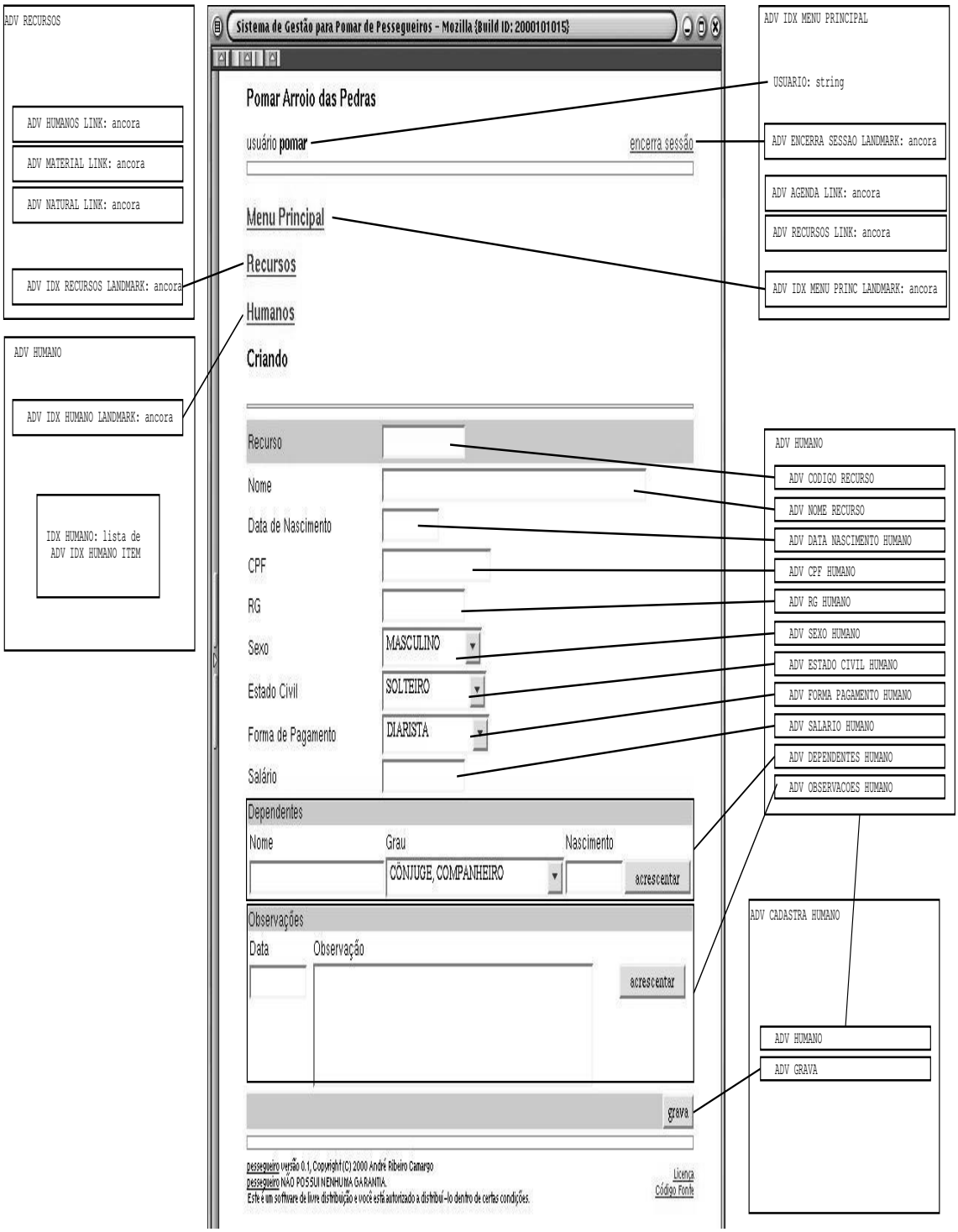

Figura 11: Implementação do ADV CTX CRIA HUMANO

7.5 A construção do caso de uso "Planejar tratos culturais"

O planejamento de tratos culturais se dá através da estimativa de custo para realização do mesmo. A Figura 13 ilustra os casos de uso de sistema que o caso de uso de

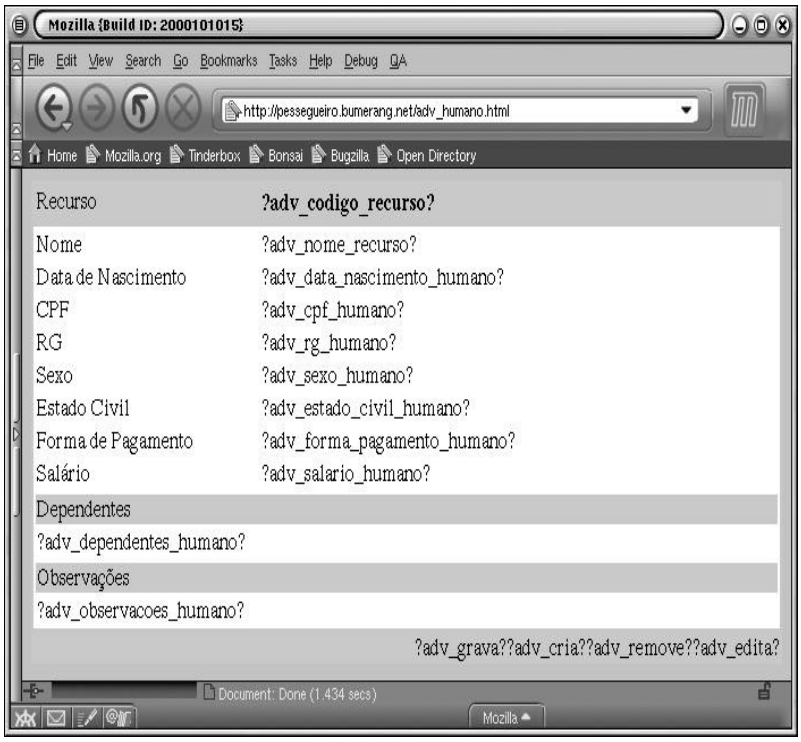

Figura 12: Metáfora de interface para o ADV HUMANO

negócio “Planejar tratos culturais” possui.

Considerações:

- Toda estimativa de trato cultural é feita por lote.
- O custo estimado do recurso máquina é feito multiplicando-se a quantidade de horas de serviço necessárias para execução da tarefa pelo valor da hora de serviço estipulada pelo proprietário do pomar.
- O custo estimado de recursos humanos (mão-de-obra) é feito multiplicando-se a quantidade de horas de serviço necessárias para execução da tarefa pelo valor da hora de mão-de-obra.
- O custo estimado de recurso insumo é feito multiplicando-se a dosagem do insumo por hectare pelo custo unitário de aquisição do mesmo (em kg ou l).
- Optou-se por não estimar Recursos materiais da classe utensílios.

Os casos de uso de sistema estão textualizados abaixo:

- **Planejar Roçada**
  A estimativa de custo de Roçada Manual é feito calculando-se o custo com

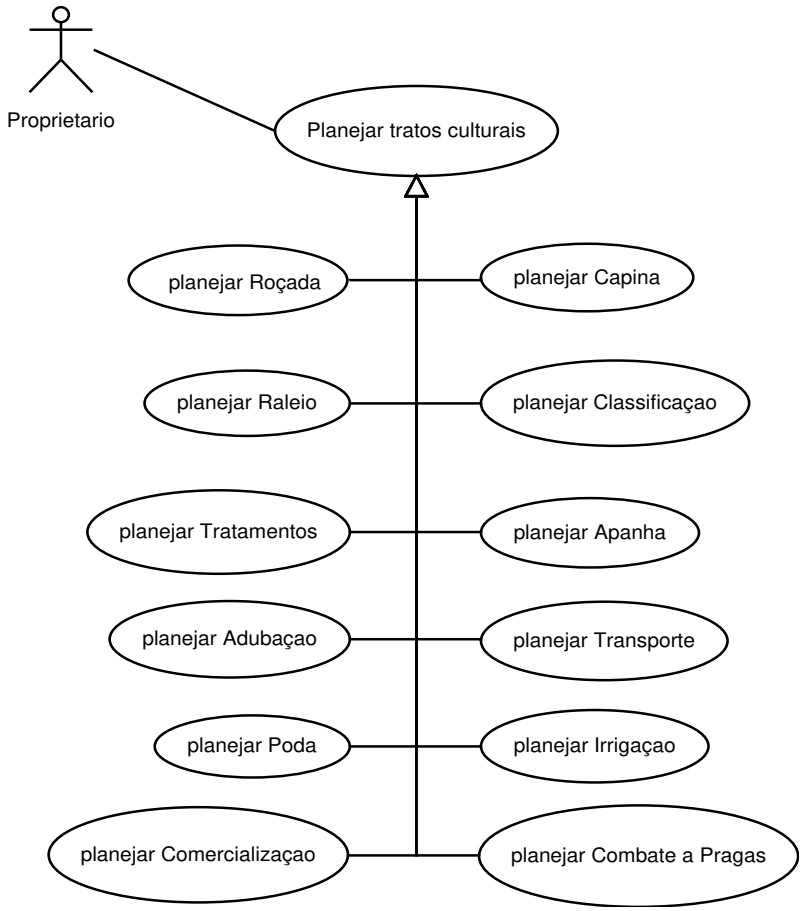

Figura 13: Caso de uso de “Planejar tratos culturais”

recursos humanos.
A estimativa de custo de Roçada Mecânica é feito somando o custo com recurso máquina e humano.

- **Planejar Raleio**
  Estima-se Raleio calculando a utilização de recursos humanos necessários para execução do trato cultural.

- **Planejar Tratamentos**
  A estimativa do custo de Tratamento é feito multiplicando-se o número de aplicações planejadas pelo custo por hectare de uma aplicação. O custo de uma aplicação é a soma do custo de distribuição e o custo com insumos. O custo de distribuição é a soma do custo da distribuição manual com o custo da distribuição mecanizada. A distribuição manual é feita com recursos humanos, enquanto que, distribuição tratorizada utiliza recurso trator. O custo com insumos é feito multiplicando-se a área do lote pelo somatório do custo de cada insumo (calda) por hectare. O custo de cada insumo é o custo de cada recurso

insumo utilizado por hectare. Na pulverização, o somatório do custo de cada insumo poderia ser chamado de custo da calda.

- **Planejar Adubação**
  Planejado da mesma forma de um Tratamento.

- **Planejar Poda**
  A poda é um trato cultural executado de forma manual, é estimado conforme a utilização de recursos humanos para efetuá-la.

- **Planejar Capina**
  Capina manual planeja-se conforme a necessidade de recursos humanos para efetuá-la, enquanto que, na capina mecânica estima-se a utilização de recursos humanos e maquinários, já o planejamento da capina química segue os moldes da estimativa de um tratamento.

- **Planejar Combate a Pragas**
  O planejamento do combate a pragas é semelhante a estimativa de um tratamento.

- **Planejar Irrigação**
  A estimativa do custo de Irrigação é o somatório dos custos estimados de água, bombeamento, recursos humanos e reposição de peças/manutenção de equipamentos. Para estimarmos o custo d'água e bombeamento é preciso definir previamente a necessidade d'água do lote. Esta necessidade d'água do lote é o produto da necessidade diária de água por planta (em m^3) pelo número de plantas do lote. Assim, o custo estimado com água é a necessidade d'água do lote multiplicado pelo valor do m^3 d'água (obs.: caso a água seja comprada, se não for, não há custo com água). O custo estimado de bombeamento é feito multiplicando-se o tempo de bombeamento pelo valor da hora de bombeamento. O tempo de bombeamento (em horas) é o quociente da necessidade d'água do lote (em m^3) pela capacidade de recalque d'água da bomba (m^3/h). A estimativa do custo com reposição de peças/manutenção de equipamentos é uma porcentagem calculada sobre o somatório dos custo de bombeamento e recursos humanos, a porcentagem padrão para cálculo é de 5%, podendo ser modificada.

- **Planejar Apanha**
  O custo estimado de apanha é o produto da produtividade média estimada por hectare, multiplicada pelo valor de apanha/kg.

- **Planejar Transporte Interno**
  O custo estimado de transporte interno é a soma da expectativa de uso de recursos materiais com a expectativa de uso de recursos humanos por hectare multiplicado pela área a ser colhida.

- **Planejar Classificação**
  O custo estimado de classificação é obtido pelo produto da produção bruta estimada (em kg) pelo custo de classificação por kg.

- **Planejar Comercialização**
  A estimativa de comercialização é o produto da expectativa de produção a ser vendida às indústrias e/ou comércio *in natura* pelo preço de mercado da fruta com base na safra anterior e às possíveis variações que possam vir a ocorrer em função de alterações de oferta e procura ou mesmo na variação monetária.

7.5.1 Modelagem conceitual

A modelagem conceitual do caso de uso "Planejar Tratos Culturais" produziu o diagrama de classes conceituais da Figura 14.

7.5.2 Projeto navegacional

Por ser um caso de uso simples, a navegação é pouca, como pode se demonstrado no seu esquema navegacional (Figura 15) e diagrama de contextos (Figura 16).

7.5.3 Projeto de interface abstrata

Os ADVs do caso de uso "Planejar Tratos Culturais" encontram-se documentados na Figura 17.

7.6 A construção do caso de uso "Registrar tarefas"

A execução de tratos culturais se dá através de tarefas que utilizam recursos do pomar. Conforme pode ser visto na Seção 5.6, decompus os tratos culturais em tarefas para facilitar a construção da aplicação e o registro das mesmas por parte dos usuários. A Figura 18 ilustra os casos de uso de sistema que o caso de uso de negócio "Registrar tarefas".

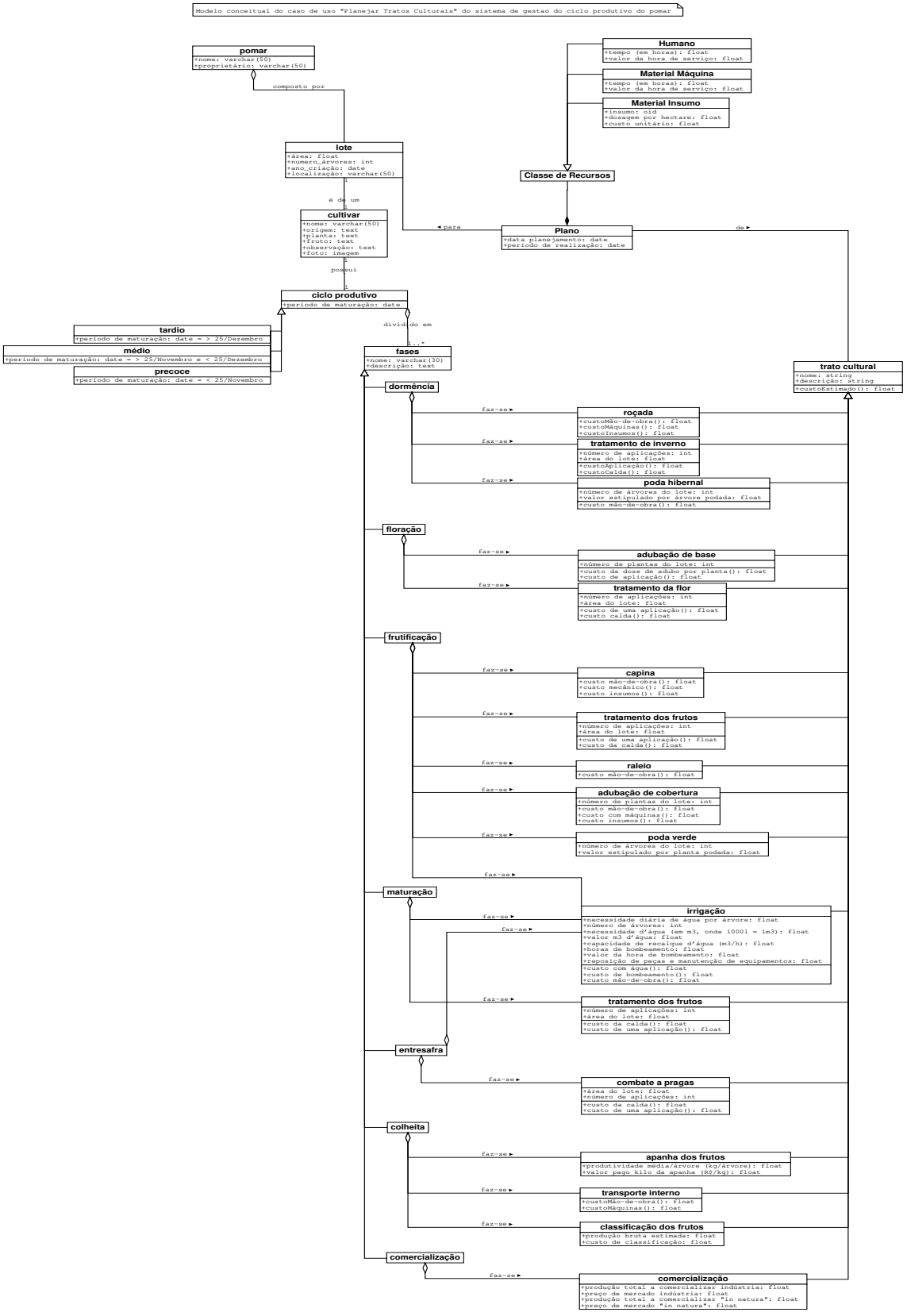

Figura 14: Modelo conceitual do caso de uso “Planejar Tratos Culturais”

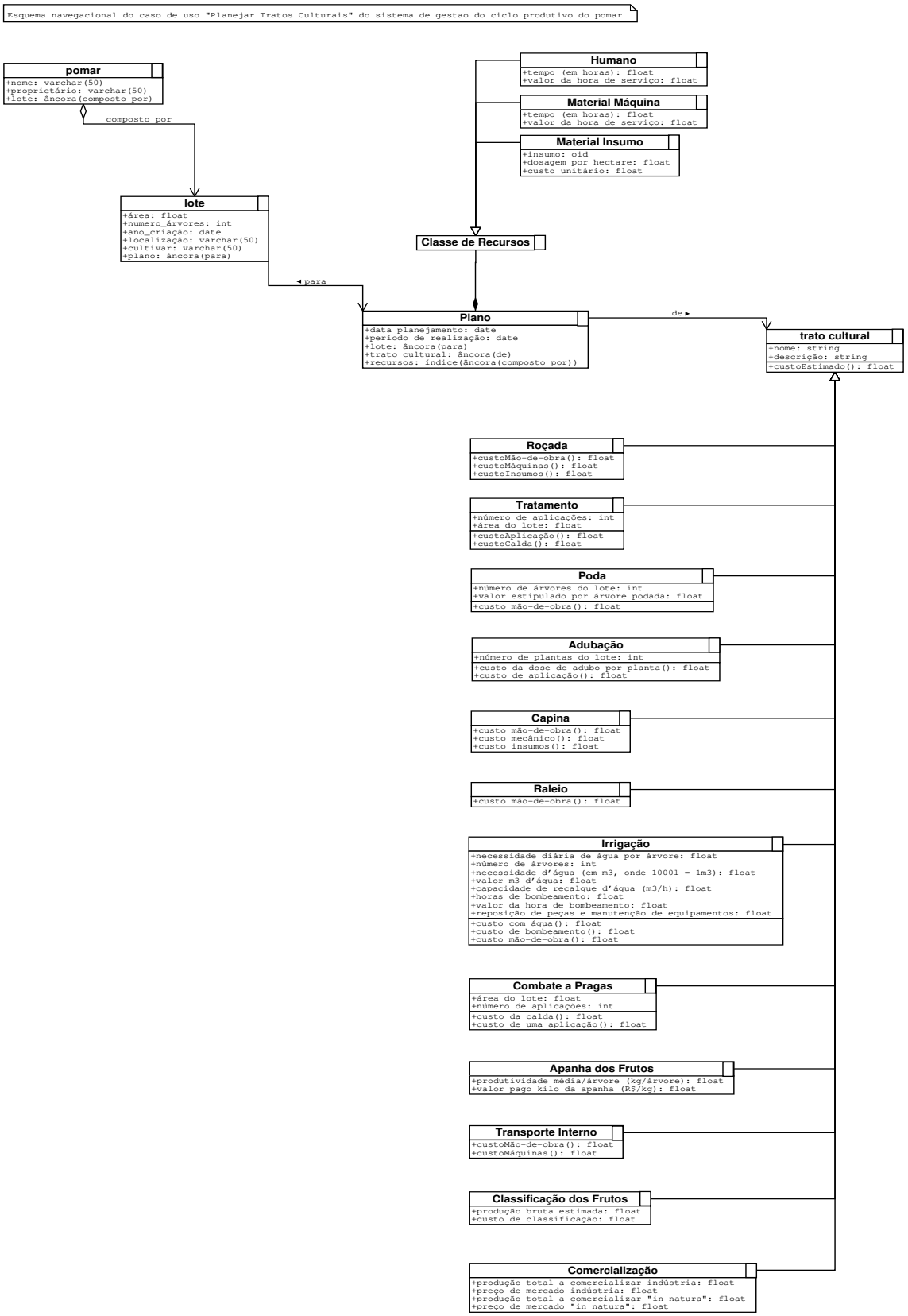

Figura 15: Esquema navegacional do caso de uso “Planejar Tratos Culturais”

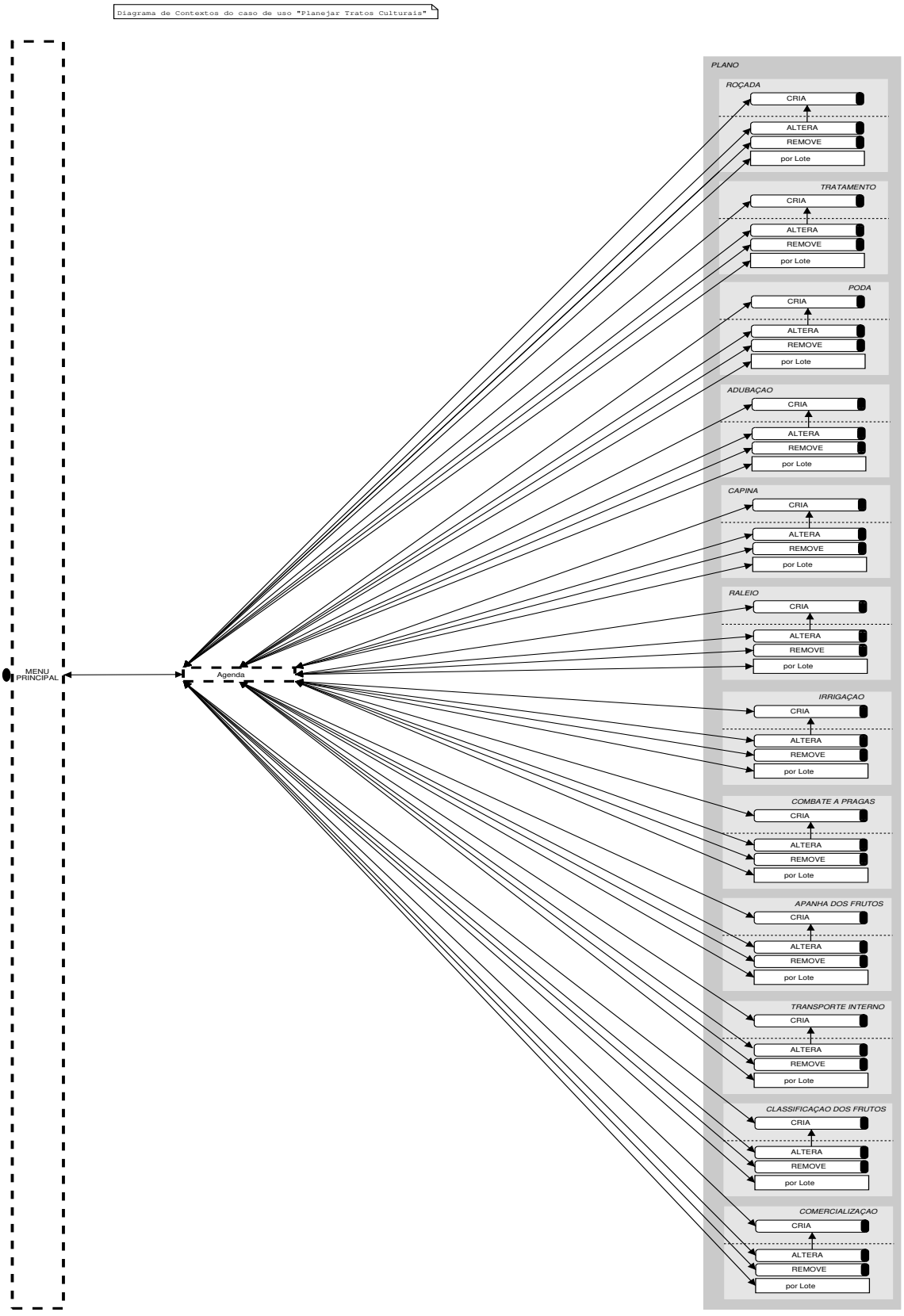

Figura 16: Diagrama de contextos do caso de uso “Planejar Tratos Culturais”

Figura 17: Abstract Data Views do caso de uso “Planejar Tratos Culturais”

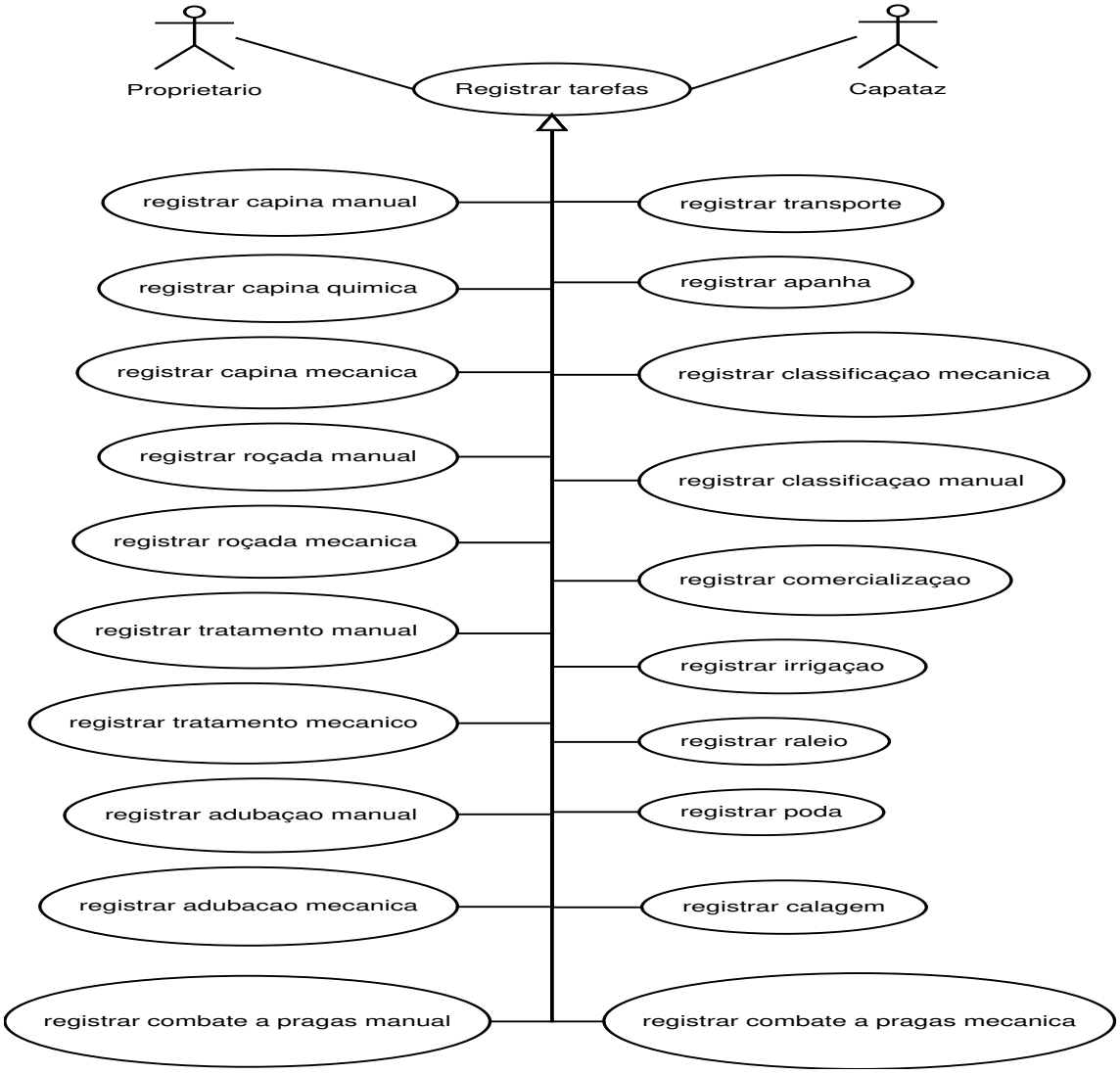

Figura 18: Caso de uso de “Registrar tarefas”

Considerações:

- O registro do uso de recursos humanos é feito documentando-se o tempo ocupado (em horas), o valor da hora trabalhada (hora normal e hora extra) com ou sem insalubridade e/ou periculosidade.
- O registro do uso de um recurso material máquina é feito anotando-se o tempo, em horas, e o valor da hora trabalhada.
- O registro do recurso insumo é feito especificando-se o nome do insumo, e a quantidade utilizada.
- O registro de uso do recurso material utensílio e implemento não é feito.

Os casos de uso de sistema estão textualizados abaixo:

- **Registrar Capina Manual**
  Registra-se o uso de recursos humanos;
- **Registrar Capina Mecânica**
  Registra-se a utilização de recursos humanos e máquinas;
- **Registrar Capina Química**
  Registra-se o uso de recursos humanos, máquinas e insumos;
- **Registrar Roçada Manual**
  Registra-se a utilização de recursos humanos;
- **Registrar Roçada Mecânica**
  Registra-se o uso de recursos humanos e máquinas;
- **Registrar Tratamento Manual**
  Registra-se a utilização de recursos humanos;
- **Registrar Tratamento Mecânico**
  Registra-se o uso de recursos humanos, máquinas e insumos;
- **Registrar Adubação Manual**
  Registra-se a utilização de recursos humanos e insumos;
- **Registrar Adubação Mecânica**
  Registra-se o uso de recursos humanos, máquinas e insumos;
- **Registrar Combate a Pragas Manual**
  Registra-se a utilização de recursos humanos e insumos;
- **Registrar Combate a Pragas Mecânica**
  Registra-se o uso de recursos humanos, máquinas e insumos;
- **Registrar Poda**
  Registra-se a utilização de recursos humanos;
- **Registrar Raleio**
  Registra-se o uso de recursos humanos;
- **Registrar Irrigação**
  Registra-se a utilização de recursos humanos e máquinas;
- **Registrar Apanha**
  O registro da apanha de fruto é feito anotando-se o tempo (em horas) e a quantidade (em kg) que cada peão conseguiu apanhar.

- **Registrar Transporte**
  O registro do transporte interno da colheita é feito anotando-se a quantidade transportada por lote e o tempo (em horas) utilizado dos recursos trator e humano.

- **Registrar Classificação Manual**
  O registro da tarefa de classificação manual é feito anotando-se o tempo (em horas) trabalhado por cada recurso humano e a quantidade (em kg) classificada nesse tempo.

- **Registrar Classificação Mecânica**
  Registra-se a utilização de recursos humanos, o tempo de trabalho da máquina e a quantidade (em kg) classificada.

- **Registrar Comercialização**
  O registro da comercialização da produção é feito anotando-se a quantidade (em kg) entregue, o valor unitário, o comprador, o responsável pela entrega, os dados do transportador (nome, placa do caminhão) e os vencimentos (data e valor).

7.6.1 Modelagem conceitual

O modelo conceitual do caso de uso de negócio "Registrar Tarefas" está representado na Figura 19.

7.6.2 Projeto navegacional

Baseado no modelo conceitual proposto anteriormente, foi criado o esquema navegacional da Figura 20 e o diagrama de contextos da Figura 21.

7.6.3 Projeto de interface abstrata

A Figura 22 ilustra os Abstract Data Views deste caso de uso.

7.7 A construção do caso de uso "Apurar Resultados"

Apurar resultados é o último caso de negócio deste sistema de gestão para pomar de pessegueiros.

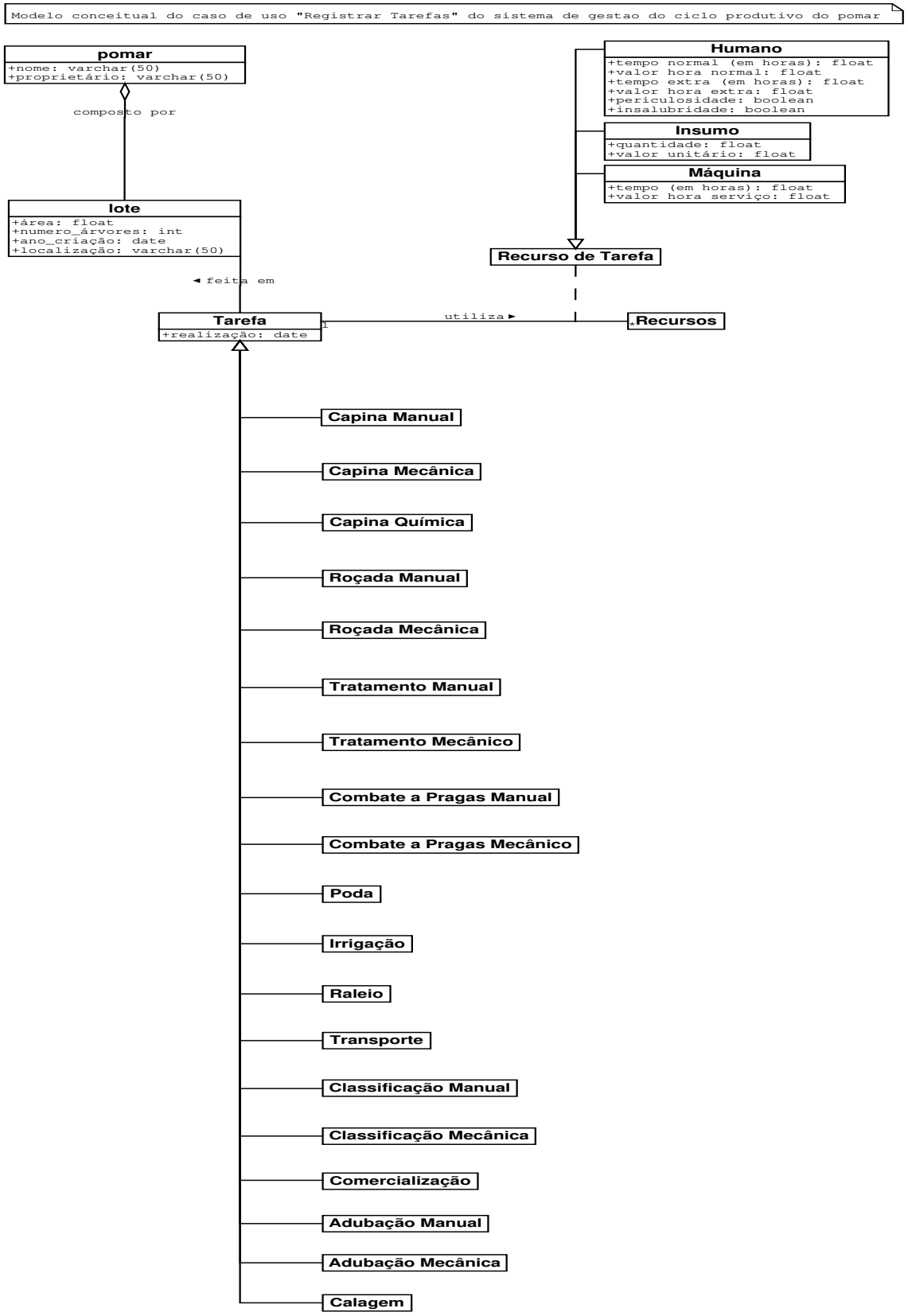

Figura 19: Modelo conceitual do caso de uso de “Registrar Tarefas”

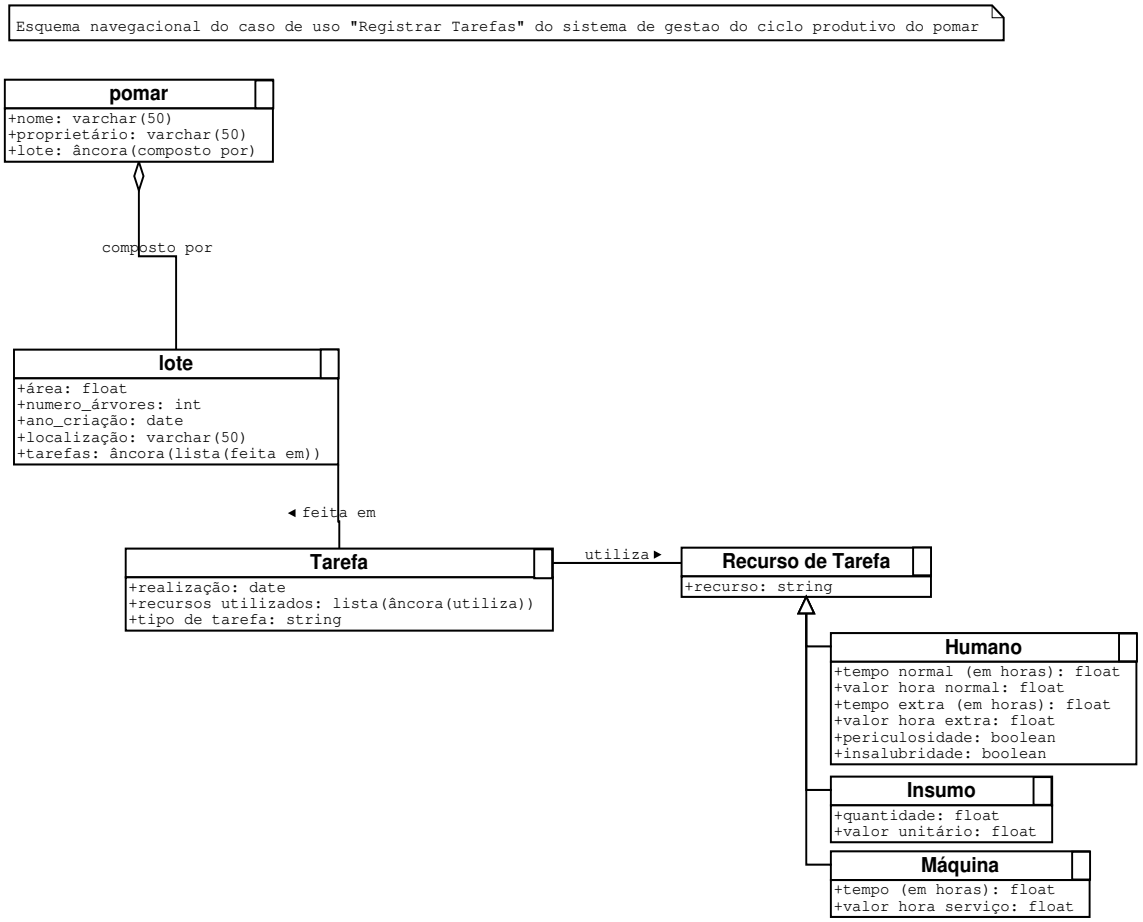

Figura 20: Esquema navegacional do caso de uso de “Registrar Tarefas”

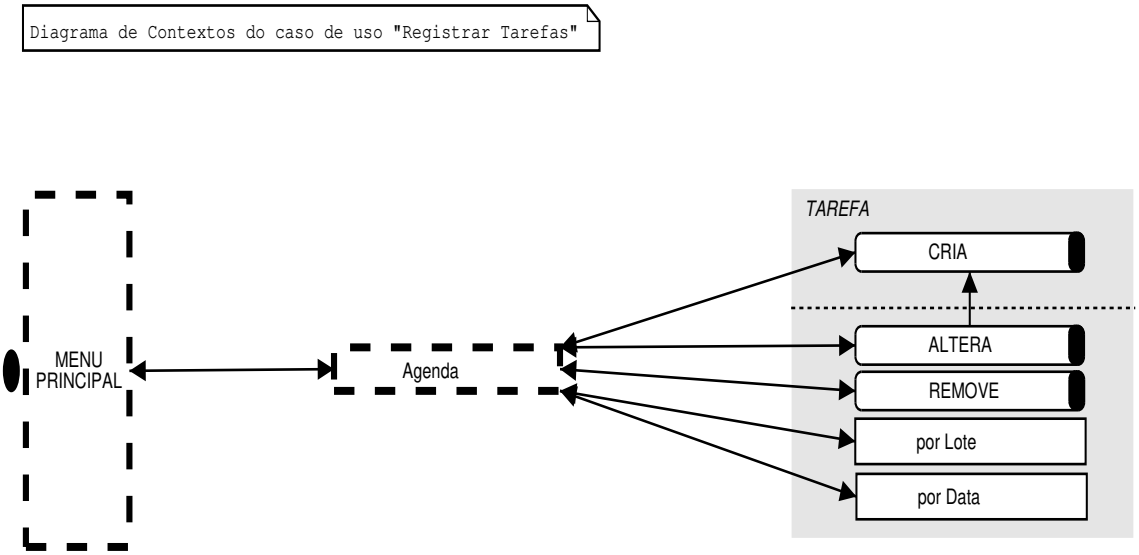

Figura 21: Diagrama de contextos do caso de uso de “Registrar Tarefas”

Em conversa com o Sr. Adelci, ficou decidido que a necessidade primária é obter relatórios que mostrem:

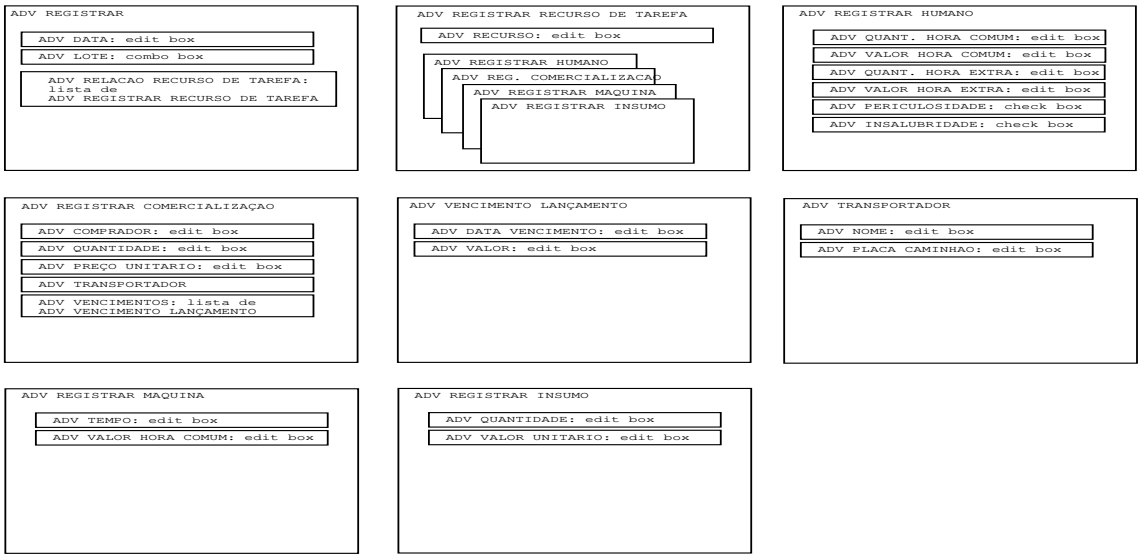

Figura 22: ADVs do caso de uso de “Registrar Tarefas”

- Quanto foi gasto por recurso?
  O sistema deve ser capaz de gerar extratos de quanto foi gasto por recursos dentro de um período de tempo especificado pelo usuário.

- Qual foi o total colhido?
  O sistema deve ser capaz de gerar relatório descriminados do total colhido por ciclo, cultivar e lote.

- Relatório de vendas da safra
  O sistema deve ser capaz de gerar relatório de vendas dentro de um período especificado pelo usuário. Neste relatório deveria conter dados como: para quem foi vendido, data da entrega, por quanto foi entregue e qual é a previsão de recebimentos.

- Qual foi o resultado final?
  O sistema deve ser capaz de gerar o relatório de resultado final por ciclo, lote e cultivar. Este relatório consiste de um balanço relacionando o valor da produção com o valor gasto para produzir.

- De quanto foi as perdas entre apanha e comercialização?
  O sistema deve ser capaz de apontar quanto do pêssego apanhado é rejeitado até a colheita.

A Figura 23 ilustra o caso de uso de negócio estudado nesta seção.

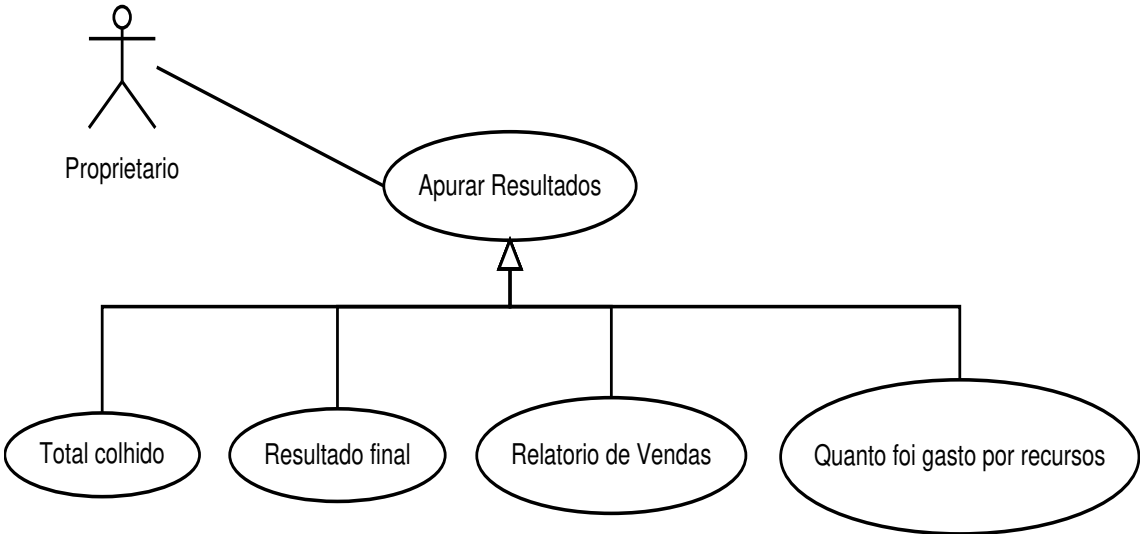

Figura 23: Caso de uso de negócio “Apurar resultados”

7.7.1 Modelagem conceitual

Por este caso de uso exclusivamente gerar informação baseada nos dados já cadastrados no sistema, o modelo conceitual deste caso de uso é o modelo conceitual da aplicação.

7.7.2 Projeto navegacional

O esquema navegacional usado é o mesmo esquema navegacional da aplicação.

O diagrama de contexto está representado na Figura 24.

7.7.3 Projeto de interface abstrata

Os ADVs dos relatórios deste caso de uso, estão representados na Figura 25.

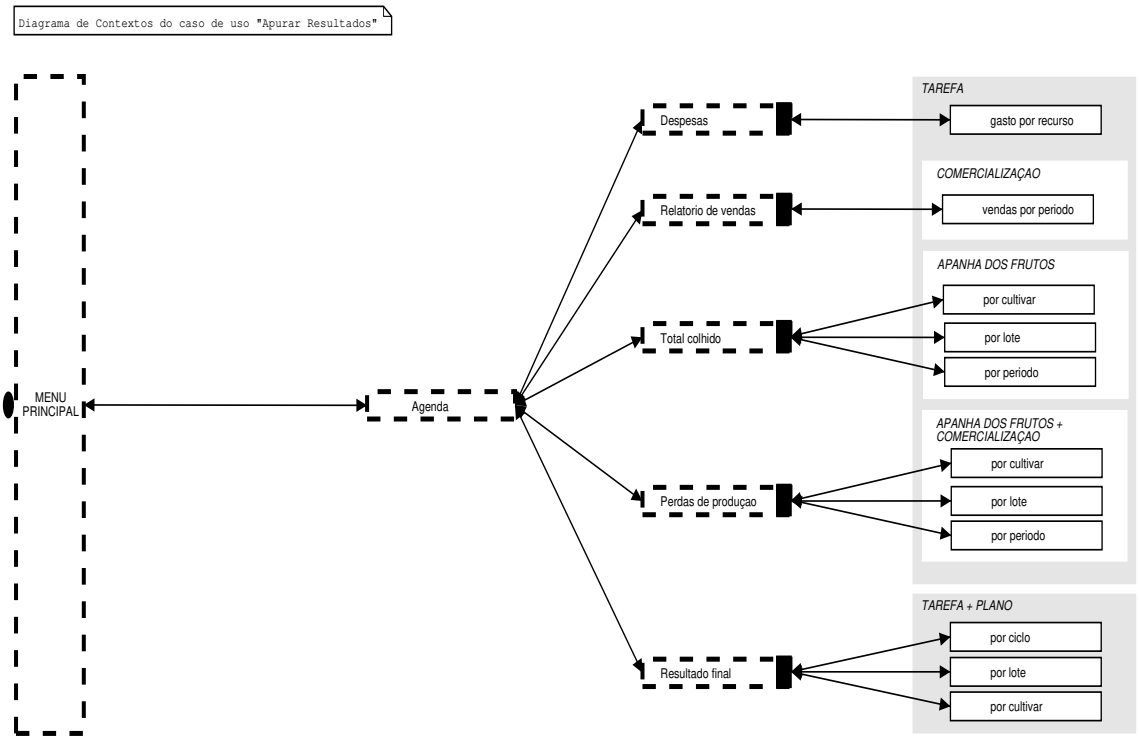

Figura 24: Diagrama de contextos do caso de uso de negócio “Apurar resultados”

Figura 25: ADVs do caso de uso de negócio “Apurar resultados”

8 CONCLUSÃO

Considerando que o desenvolvimento deste trabalho começou em Abril do ano de 2000, esta monografia é o resultado de 8 meses de estudo em diversas áreas do saber. Essa diversidade de matérias estudadas só foi possível dado a forma como a disciplina de Projeto de Graduação é conduzida. Acredito que o grande mérito desta disciplina seja a possibilidade do aluno poder escolher o tema do seu projeto, tema esse que deve possuir um cunho inovador e foco direcionado para a comunidade local. Essas duas exigências estabelecem um desafio interessante para a escolha do tema, já que a avaliação é baseada na capacidade do aluno de desenvolver raciocínio, integrar conceitos adquiridos ao longo do curso e realizar contribuição para ensino, pesquisa e extensão [Cos 99].

Para o desenvolvimento deste trabalho a Internet provou ser realmente uma grande biblioteca, basta verificar as referências bibliográficas e concluir que 80% do referencial teórico utilizado neste trabalho provem de Universal Resource Locators (URL[17]). Além de conteúdo estático, ainda foi possível entrar em contato com os autores de vários dos artigos utilizados, para esclarecimento de dúvidas e/ou troca de idéias, de forma barata e rápida. Sem essa ferramenta, certamente o desenvolvimento deste trabalho seria bem mais complicado e incompleto. É interessante ressaltar que o software livre somente conseguiu atingir o nível que está hoje graças a essa forma independente, "indisciplinada" e sem fronteiras para disseminação do conhecimento humano.

Quando fui escrever este trabalho surgiu a primeira dificuldade. Como o pretendido era somente utilizar software livre, que ferramenta eu iria utilizar para escrever esta monografia? Pesquisei alguns processadores de texto gráficos, mas infelizmente não encontrei uma ferramenta visual completa que atendesse aos requisitos tipográficos que esta monografia exige. Foi então que descobri o LaTeX e posso dizer que foi "amor a primeira compilação". Como o LaTeX é baseado em classes de documento e até então não existia uma classe de documento para monografias da ESIN, resolvi criar uma. Não sou perito em LaTeX mas tive que estudar mais profundamente a linguagem para conseguir criar a classe que utilizei para escrever o documento que estás lendo. Disponibilizarei a classe na Internet para que futuros formandos da ESIN possam usufruir dessa facilidade caso prefiram utilizar LaTeX como ferramenta para escritura de sua monografia.

[17] **Universal Resource Locator** é a forma de representação para endereço de informações na Internet. É disposta na forma *serviço://servidor/arquivo*. Um exemplo de URL seria http://projetoacanguacu.yi.org/index.php, onde o serviço é *http*, o servidor é *projetoacanguacu.yi.org* e o arquivo *index.php*.

Outra dificuldade encontrada foi a fraca implementação do modelo orientado a objetos nas ferramentas que utilizei na construção do protótipo da aplicação de gestão para pomar de pessegueiros, principalmente na implementação do PHP.

Possuir um proprietário de pomar de pessegueiros em casa é uma facilidade que merece ser ressaltada. Isso possibilitou uma resposta mais rápida e objetiva frente aos questionamentos e demonstrações que precisei efetuar.

Pretendo entrar em contato com órgãos que lidam com os produtores de pêssego, como EMBRAPA e EMATER, procurando buscar uma forma de viabilizar o diálogo entre estas entidades e os produtores, através da publicação de informações e até consultas virtuais (lembre-se que a Internet possibilita isso).

Paralelamente a este trabalho, motivado pela filosofia do software livre e por um grande senso de "querer fazer as coisas acontecerem" criei uma proposta para democratização da informática na cidade onde resido, chamada "Projeto Acanguaçu". Uma cópia desta proposta é apresentada no ANEXO B.

Após obter o título de bacharel em Análise de Sistemas, pretendo finalmente implantar o projeto, disseminando assim o software livre por lugares que ainda não tem idéia dos benefícios que a filosofia GNU junto da consciência civil dos cidadãos podem proporcionar a região onde vivo.

9 ANEXOS

9.1 ANEXO A

Licença pública GNU

GNU GENERAL PUBLIC LICENSE Version 2, June 1991

This is an unofficial translation of the GNU General Public License into Portuguese. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU GPL -- only the original English text of the GNU GPL does that. However, we hope that this translation will help Portuguese speakers understand the GNU GPL better.

F.3.1 Introdução

As licenças de muitos softwares são desenvolvidas para cercear a liberdade de uso, compartilhamento e mudanças. A GNU Licença Pública Geral ao contrário, pretende garantir a liberdade de compartilhar e alterar softwares de livre distribuição - tornando-os de livre distribuição também para quaisquer usuários. A Licença Pública Geral aplica-se à maioria dos softwares da Free Software Foundation e a qualquer autor que esteja de acordo de utilizá-la (alguns softwares da FSF são cobertos pela GNU Library General Public License).

Quando nos referimos a softwares de livre distribuição, referimo-nos à liberdade e não ao preço. Nossa Licença Pública Geral foi criada para garantir a liberdade de distribuição de cópias de softwares de livre distribuição (e cobrar por isso caso seja do interesse do distribuidor), o qual recebeu os códigos fonte, o qual pode ser alterado ou utilizado em parte em novos programas.

Para assegurar os direitos dos desenvolvedores, algumas restrições são feitas, proibindo a todas as pessoas a negação desses direitos ou a solicitação de sua abdicação. Essas restrições aplicam-se ainda a certas responsabilidades sobre a distribuição ou modificação do

software.

Por exemplo, ao se distribuir cópias de determinado programa, por uma taxa determinada ou gratuitamente, deve-se informar sobre todos os direitos incidentes sobre esse programa, assegurando-se que os fontes estejam disponíveis assim como a Licença Pública Geral GNU.

A proteção dos direitos envolve dois passos: (1) copyright do software e (2) licença que dá permissão legal para cópia, distribuição e/ou modificação do softwares.

Ainda para a proteção da FSF e do autor é importante que todos entendam que não há garantias para softwares de livre distribuição. Caso o software seja modificado por alguém e passado adiante, este software não mais refletirá o trabalho original do autor não podendo portanto ser garantido por aquele.

Finalmente, qualquer programa de livre distribuição é constantemente ameaçado pelas patentes de softwares. Buscamos evitar o perigo de que distribuidores destes programas obtenham patentes individuais, tornado-se seus donos efetivos. Para evitar isso foram feitas declarações expressas de que qualquer solicitação de patente deve ser feita permitindo o uso por qualquer indivíduo, sem a necessidade de licença de uso.

Os termos e condições precisas para cópia, distribuição e modificação seguem abaixo:

F.3.2 Licença Pública Geral GNU

TERMOS E CONDIÇÕES PARA CÓPIA, DISTRIBUIÇÃO E MODIFICAÇÃO

0

Esta licença se aplica a qualquer programa ou outro trabalho que contenha um aviso colocado pelo detentor dos direitos autorais dizendo que aquele poderá ser distribuído nas condições da Licença Pública Geral. O Programa, abaixo refere-se a qualquer software ou trabalho e a um trabalho baseado em um Programa e significa tanto o Programa em si como quaisquer trabalhos derivados de acordo com a lei de direitos autorais, o que significa dizer, um trabalho que contenha o Programa ou uma parte deste, na sua forma original ou com modificações ou traduzido para uma outra língua (tradução está incluída sem limitações no termo modificação).

Atividades distintas de cópia, distribuição e modificação não estão

cobertas por esta Licença, estando fora de seu escopo. O ato de executar o Programa não está restringido e a saída do Programa é coberta somente caso seu conteúdo contenha trabalhos baseados no Programa (independentemente de terem sidos gerados pela execução do Programa). Se isso é verdadeiro depende das funções executadas pelo Programa.

1

O código fonte do Programa, da forma como foi recebido, pode ser copiado e distribuído, em qualquer media, desde que seja providenciada um aviso adequado sobre os copyrights e a negação de garantias, e todos os avisos que se referem à Licença Pública Geral e à ausência de garantias estejam inalterados e que qualquer produtos oriundo do Programa esteja acompanhado desta Licença Pública Geral.

É permitida a cobrança de taxas pelo ato físico de transferência ou gravação de cópias, e podem ser dadas garantias e suporte em troca da cobrança de valores.

2

Pode-se modificar a cópia ou cópias do Programa de qualquer forma que se deseje, ou ainda criar-se um trabalho baseado no Programa, e copiá-la e distribuir tais modificações sob os termos da seção 1 acima e do seguinte:

1. Deve existir aviso em destaque de que os dados originais foram alterados nos arquivos e as datas das mudanças;
2. Deve existir aviso de que o trabalho distribuído ou publicado é, de forma total ou em parte derivado do Programa ou de alguma parte sua, e que pode ser licenciado totalmente sem custos para terceiros sob os termos desta Licença.
3. Caso o programa modificado seja executado de forma interativa, é obrigatório, no início de sua execução, apresentar a informação de copyright e da ausência de garantias (ou de que a garantia corre por conta de terceiros), e que os usuários podem redistribuir o programa sob estas condições, indicando ao usuário como acessar esta Licença na sua íntegra.

Esses requisitos aplicam-se a trabalhos de modificação em geral. Caso algumas seções identificáveis não sejam derivadas do Programa, e podem ser consideradas como partes independentes, então esta Licença e seus Termos não se aplicam àquelas seções quando distribuídas separadamente. Porém ao distribuir aquelas seções como parte de um trabalho baseado no Programa, a distribuição como um todo deve conter os termos desta Licença, cujas permissões estendem-se ao trabalho como

um todo, e não a cada uma das partes independentemente de quem os tenha desenvolvido.

Mais do que tencionar contestar os direitos sobre o trabalho desenvolvido por alguém, esta seção objetiva propiciar a correta distribuição de trabalhos derivados do Programa.

Adicionalmente, a mera adição de outro trabalho ao Programa, porém não baseado nele nem a um trabalho baseado nele, a um volume de armazenamento ou media de distribuição não obriga a utilização desta Licença e de seus termos ao trabalho.

3

São permitidas a cópia e a distribuição do Programa (ou a um trabalho baseado neste) na forma de código objeto ou executável de acordo com os termos das Seções 1 e 2 acima, desde que atendido o seguinte:

1. Esteja acompanhado dos códigos fonte legíveis, os quais devem ser distribuídos na forma da Seções 1 e 2 acima, em mídia normalmente utilizada para manuseio de softwares ou
2. Esteja acompanhado de oferta escrita, válida por, no mínimo 3 anos, de disponibilizar a terceiros, por um custo não superior ao custo do meio físico de armazenamento , uma cópia completa dos códigos fonte em meio magnético, de acordo com as Seções 1 e 2 acima.
3. Esteja acompanhada com a mesma informação recebida em relação à oferta da distribuição do código fonte correspondente. (esta alternativa somente é permitida para distribuições não comerciais e somente se o programa recebido na forma de objeto ou executável tenha tal oferta, de acordo com a sub-seção 2 acima).

O código fonte de um trabalho é a melhor forma de produzirem-se alterações naquele trabalho. Códigos fontes completos significam todos os fontes de todos os módulos, além das definições de interfaces associadas, arquivos, scripts utilizados na compilação e instalação do executável. Como uma exceção excepcional, o código fonte distribuído poderá não incluir alguns componentes que não se encontrem em seu escopo, tais como compilador, kernel, etc... para o SO onde o trabalho seja executado.

Caso a distribuição do executável ou objeto seja feita através de acesso a um determinado ponto, então oferta equivalente de acesso deve ser feita aos códigos fonte, mesmo que terceiros não sejam obrigados a copiarem os fontes juntos com os objetos simultaneamente.

4

Não é permitida a cópia, modificação, sublicenciamento ou distribuição do Programa, exceto sob as condições expressas nesta Licença. Qualquer tentativa de cópia, modificação, sublicenciamento ou distribuição do Programa é proibida, e os direitos descritos nesta Licença cessarão imediatamente. Terceiros que tenham recebido cópias ou direitos na forma desta Licença não terão seus direitos cessados desde que permaneçam dentro das cláusulas desta Licença.

5

Não é necessária aceitação formal desta Licença, apesar de que não haverá documento ou contrato que garanta permissão de modificação ou distribuição do Programa ou seus trabalhos derivados. Essas ações são proibidas por lei, caso não se aceitem as condições desta Licença. A modificação ou distribuição do Programa ou qualquer trabalho baseado neste implica na aceitação desta Licença e de todos os termos desta para cópia, distribuição ou modificação do Programa ou trabalhos baseados
neste.

6

Cada vez que o Programa seja distribuído (ou qualquer trabalho baseado neste), o recipiente automaticamente recebe uma licença do detentor original dos direitos de cópia, distribuição ou modificação do Programa objeto deste termos e condições. Não podem ser impostas outras restrições nos recipientes.

7

No caso de decisões judiciais ou alegações de uso indevido de patentes ou direitos autorais, restrições sejam impostas que contradigam esta Licença, estes não isentam da sua aplicação. Caso não seja possível distribuir o Programa de forma a garantir simultaneamente as obrigações desta Licença e outras que sejam necessárias, então o Programa não poderá ser distribuído.

Caso esta Seção seja considerada inválida por qualquer motivo particular ou geral, o seu resultado implicará na invalidação geral desta licença na cópia, modificação, sublicenciamento ou distribuição do Programa ou trabalhos baseados neste.

O propósito desta seção não é, de forma alguma, incitar quem quer que seja a infringir direitos reclamados em questões válidas e procedentes, e sim proteger as premissas do sistema de livre

distribuição de software. Muitas pessoas têm feito contribuições generosas ao sistema, na forma de programas, e é necessário garantir a consistência e credibilidade do sistema, cabendo a estes e não a terceiros decidirem a forma de distribuição dos softwares.

Esta seção pretende tornar claro os motivos que geraram as demais cláusulas destas Licença.

8

Caso a distribuição do Programa dentro dos termos desta Licença tenha restrições em algum País, quer por patentes ou direitos autorais, o detentor original dos direitos autorais do Programa sob esta Licença pode adicionar explicitamente limitações geográficas de distribuição, excluindo aqueles Países, fazendo com que a distribuição somente seja possível nos Países não excluídos.

9

A Fundação de Software de Livre Distribuição (FSF – Free Software Foundation) pode publicar versões revisadas ou novas versões desta Licença Pública Geral de tempos em tempos. Estas novas versões manterão os mesmos objetivos e o espírito da presente versão, podendo variar em detalhes referentes a novas situações encontradas.

A cada versão é dada um número distinto. Caso o Programa especifique um número de versão específico desta Licença a qual tenha em seu conteúdo a expressão – ou versão mais atualizada-, é possível optar pelas condições daquela versão ou de qualquer versão mais atualizada publicada pela FSF.

10

Caso se deseje incorporar parte do Programa em outros programas de livre distribuição de softwares é necessária autorização formal do autor. Para softwares que a FSF detenha os direitos autorais, podem ser abertas exceções desde que mantido o espírito e objetivos originais desta Licença.

11

AUSÊNCIA DE GARANTIAS

UMA VEZ QUE O PROGRAMA É LICENCIADO SEM ÔNUS, NÃO HÁ QUALQUER GARANTIA PARA O PROGRAMA. EXCETO QUANDO TERCEIROS EXPRESSEM-SE FORMALMENTE O PROGRAMA É DISPONIBILIZADO EM SEU FORMATO ORIGINAL, SEM GARANTIAS DE

QUALQUER NATUREZA, EXPRESSAS OU IMPLÍCITAS, INCLUINDO MAS NÃO LIMITADAS, A GARANTIAS COMERCIAIS E DO ATENDIMENTO DE DETERMINADO FIM. A QUALIDADE E A PERFORMANCE SÃO DE RISCO EXCLUSIVO DOS USUÁRIOS, CORRENDO POR SUAS CONTA OS CUSTOS NECESSÁRIOS A EVENTUAIS ALTERAÇÕES, CORREÇÕES E REPAROS JULGADOS NECESSÁRIOS.

EM NENHUMA OCASIÃO, A MENOS QUE REQUERIDO POR DECISÃO JUDICIAL OU POR LIVRE VONTADE, O AUTOR OU TERCEIROS QUE TENHAM MODIFICADO O PROGRAMA, SERÃO RESPONSÁVEIS POR DANOS OU PREJUÍZOS PROVENIENTES DO USO OU DA FALTA DE HABILIDADE NA SUA UTILIZAÇÃO (INCLUINDO MAS NÃO LIMITADA A PERDA DE DADOS OU DADOS ERRÔNEOS), MESMO QUE TENHA SIDO EMITIDO AVISO DE POSSÍVEIS ERROS OU DANOS.

FIM DA LICENÇA

F.3.3 Apêndice

Como aplicar estes termos a novos softwares?

Caso se tenha desenvolvido um novo programa e se deseje a sua ampla distribuição para o público, a melhor forma de consegui-lo é torná-lo um software de livre distribuição, o qual qualquer um possa distribuí-lo nas condições desta Licença.

Para tanto basta anexar este aviso ao programa. É aconselhável indicar ainda no início de cada arquivo fonte a ausência de garantias e um apontamento para um arquivo contendo o texto geral desta Licença, como por exemplo:

<nome do programa e função> Copyright (C) 199X <Autor>

Este programa é um software de livre distribuição, que pode ser copiado e distribuído sob os termos da Licença Pública Geral GNU, conforme publicada pela Free Software Foundation, versão 2 da licença ou (a critério do autor) qualquer versão posterior.

Este programa é distribuído na expectativa de ser útil aos seus usuários, porém NÃO TEM NENHUMA GARANTIA, EXPLÍCITAS OU IMPLÍCITAS , COMERCIAIS OU DE ATENDIMENTO A UMA DETERMINADA FINALIDADE. Consulte a Licença Pública Geral GNU para maiores detalhes.

Deve haver uma cópia da Licença Pública Geral GNU junto com este software em inglês ou português. Caso não haja escreva para Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Autor@mail.com.br
Endereço

Caso o programa seja interativo, apresente na sua saída um breve aviso quando de seu início como por exemplo:

Gnomovision versão 69, Copyright ©199a Yoyodine

Softwares NÃO POSSUI NENHUMA GARANTIA; para detalhes digite mostre garantia. Este é um software de livre distribuição e você está autorizado a distribui-lo dentro de certas condições. Digite mostre condição para maiores detalhes.

Os comandos hipotéticos mostre garantia e mostre condição apresentarão as partes apropriadas da Licença Pública Geral GNU. Evidentemente os comandos podem variar ou serem acionado por outras interfaces como clique de mouse, etc...

9.2 ANEXO B

Proposta de Projeto para
Democratização da Informática em Canguçu

Versão 1.2 em 27/Novembro/2000
Versão 1.1 em 14/Junho/2000
Versão 1.0 em 09/Junho/2000
Versão 0.9 em 07/Junho/2000
Versão 0.3 em 29/Maio/2000
Versão 0.2 em 26/Maio/2000
Versão 0.1 em 20/Maio/2000

1. Introdução

"Democracia? É dar a todos o mesmo ponto de partida.
Quanto ao ponto de chegada, isso depende de cada um."
Mário Quintana

Com uma revolução chamada globalização acontecendo de certa forma através da popularização da Internet, a informática vem tornando-se cada vez mais parte do cotidiano humano. Isso acarreta mudanças sociais enormes na vida das pessoas, um exemplo direto dessas mudanças é saber manusear um novo equipamento: o computador.
O conhecimento de como utilizar esse equipamento, permitirá às pessoas manter-se em contato com o mundo, enquanto que, aqueles sem acesso a essa tecnologia correm o sério risco de exclusão social, profissional, etc.
Gilberto Dimenstein, em seu livro Aprendiz do Futuro, Cidadania hoje e amanhã discorre que:
"As chances de o Brasil gerar emprego na era da informatização, marcada por competição selvagem, dependem em boa parte não só do que se faz nas empresas, mas do que se produz hoje em sala de aula. Escola que não ensina a manejar computadores, entrar em redes de informações, mantendo o aluno em permanente reciclagem, cria novos analfabetos.
O sem-computador de hoje é o sem-terra do futuro. Vai ficar vagando à procura de auxílio oficial, com poucas chances de encontrar um bom emprego. Cedo ou tarde o drama dos sem-computador vai entrar no topo da agenda brasileira."
Mas como possibilitar que qualquer pessoa tenha acesso a informática?
Através de um projeto social visando ensinar informática, com custo zero para os participantes.
A educação dessas pessoas na área da informática, geraria um

retorno social muito grande como, por exemplo, capacitação de pessoal para o mercado de trabalho, descobrimento de novos talentos e ampliação da perspectiva de vida da população.
Ciente dos problemas da região, onde muitas pessoas não tem condições de vida digna e tantas outras sem condições de adquirir um computador, proponho com este documento uma forma de viabilizar o ensino da informática a essas pessoas.
Me refiro a ensinar, não apenas treinar, já que projeto deveria desenvolver o educando, assegurar-lhe a formação comum indispensável para o exercício da cidadania e fornecer-lhe meios para progredir no trabalho e em estudos posteriores, não sendo portanto, apenas um curso bitolado de como utilizar o computador.
Quando este projeto se encontrar numa fase mais adiantada, madura e auto-suficiente, poderia agir paralelamente a educação básica (ensino fundamental e ensino médio) incluindo mais uma área do saber na formação desses educandos.

2. O Projeto ACANGUAÇU

"ACANGUAÇU - Onça pintada que no passado viveu por estes pagos, segundo teoria existente, deu origem ao nome da cidade de Canguçu"

O Projeto ACANGUAÇU tem como meta proporcionar às pessoas contato com o mundo da informática. O Projeto ACANGUAÇU proporcionará este contato através da promoção de palestras, painéis, grupos de estudos e cursos de capacitação de pessoal.
A mão-de-obra utilizada no projeto, num primeiro momento, deve ser voluntária. Futuramente, se obtivermos recursos talvez seja empregado mão-de-obra assalariada.
A utilização e divulgação de software livre é indiscutível, já que o Movimento Software Livre valoriza os ideais de solidariedade humana incentivando a criação de uma comunidade de usuários de informática, além de ser uma forma de baratear o custo de implantação deste projeto.
Atualmente, a proposta do Projeto ACANGUAÇU não está ligada a nenhuma entidade, orgão ou partido político, sendo uma iniciativa voluntária e sem fins lucrativos.

2.1. Software Livre

Segundo a Revista do Linux, edição de Maio/2000:
"O estado do Rio Grande do Sul é, sem dúvida, o estado brasileiro mais entusiasta na discussão e disseminação do uso de software livre na informática pública. Governo Estadual e Prefeitura Municipal de Porto Alegre, através de suas empresas

de processamento de dados, a PROCERGS e a PROCEMPA, têm levantado essa bandeira, mostrando que o uso de programas com código aberto e de livre distribuição está entre as soluções mais viáveis para um país que precisa reduzir os vultuosos gastos públicos com a compra de software proprietário, universalizar o acesso à tecnologia e oferecer uma informatização segura e de qualidade aos cidadãos."
O Projeto ACANGUAÇU objetiva ser o agente estimulador da utilização de Software Livre na cidade de Canguçu.

2.2. Palestras

As palestras assumirão o papel de introduzir o projeto na sociedade. Depois da família, a escola é o principal semeador do saber, sendo responsável pelo pleno desenvolvimento do educando, seu preparo para o exercício da cidadania e sua qualificação para o trabalho. Baseado nisso, inicialmente o público alvo seriam turmas de 8ª série do ensino fundamental da rede escolar municipal, estadual e privada no município de Canguçu-RS, podendo posteriormente, ampliar-se para todo o ensino fundamental e médio.

2.3. Painéis

Através da promoção de painéis de discussão, o projeto proporcionará uma forma da sociedade debater assuntos relacionados a informática com convidados de outras instituições de ensino, empresas e membros do projeto.

2.4. Cursos de capacitação de pessoal

Devido a escassez de recursos, este é o item de mais difícil implantação. Seria necessário montar um laboratório com equipamentos que permitisse a realização de cursos de informática.
Os cursos abrangeriam áreas como informática básica, computação gráfica, administração de redes, desenvolvimento de software, etc. Sendo que o curso de informática básica deveria ser implantado primeiramente, visto que como o nome já diz, ele forneceria a base de conhecimento para os cursos seguintes. Isso possibilitaria a identificação de novos talentos, sendo que estes seriam convidados a participar de uma Cooperativa de Desenvolvimento de Software Livre.
A clientela alvo desses cursos seriam pessoas que não possuem computador e, portanto, precisam utilizar recursos de terceiros (neste caso, recursos do Projeto) para desenvolver a prática do

material exposto durante as aulas.
Nos horários vagos, o laboratório ficaria a disposição da comunidade canguçuense, propiciando que aquelas pessoas que não dispunham de computador possam ter um contato extra ou ainda aperfeiçoar seus conhecimentos.

2.5. Grupos de Estudos

O público alvo desses grupos de estudo seriam pessoas que dispunham de um computador para estudar os assuntos abordados durante os encontros.
Devido a dificuldade de se obter recursos para montagem do laboratório, os Grupos de Estudo surgem como resposta rápida para implantação deste projeto, já que os componentes utilizariam recursos próprios, fazendo-se necessário somente a locação de espaço físico para realização dos encontros.
Os grupos seriam organizados conforme o programa dos cursos de capacitação de pessoal.
O material poderia ficar disponível na Internet, para que os componentes dos grupos estudem por conta própria, reservando os encontros para debate daquele material.

2.6. A Cooperativa de Desenvolvimento de Software Livre

[1]Hugo Fogliano Gonçalves, em seu artigo [2]"O Cooperativismo de Trabalho e a Organização do Trabalho" diz que
"A união de pessoas objetivando a melhoria de todos, o crescimento de cada um, o alcance de maiores resultados, o aumento e a melhor distribuição de renda, o desenvolvimento da criatividade, a afirmação dos mesmos interesses e das mesmas dificuldades, caracterizam uma cooperativa".
A constituição de um provedor de serviços em informática, organizada na forma de uma cooperativa de trabalho utilizando a mão-de-obra oriunda do Projeto ACANGUAÇU, fecha o ciclo do projeto. Já que formaríamos pessoas para o mercado de trabalho e ainda absorveríamos parte dessa mão-de-obra.
Essa cooperativa prestaria serviços de informática para terceiros mediante remuneração justa.
Proponho ainda que seja criado além da taxa administrativa uma taxa de manutenção que seria aplicada no projeto para dar continuidade ao processo.

3. Quem é o proponente?

[3]André Ribeiro Camargo é técnico em Eletrônica pelo CEFET-RS e aluno formando do curso de Graduação em Análise de Sistemas

pela Universidade Católica de Pelotas. Trabalha na área da informática desde 1994, tendo desenvolvido serviços tanto na área de manutenção de computadores, desenvolvimento de software e instrutor em cursos de informática. Atualmente atua como desenvolvedor free-lancer de aplicações para Internet, já prestou serviços no Centro de Informática da Universidade Federal de Pelotas e para o Sindicado dos Municipários de Canguçu(SIMCA).
Entusiasta do Movimento Software Livre, curioso por natureza e aficionado em informática por vocação, André almeja através desse projeto um futuro melhor para região onde vive.

4. Conclusão

"É tudo muito simples, Anji-san. Apenas mude sua concepção de mundo."
Shogun, de James Clavell

A realização de palestras, painéis e grupos de estudo é viável a curto prazo, já que dispomos de pessoal e equipamentos para isso. Enquanto que cursos de capacitação de pessoal, devido a infra-estrutura envolvida, somente seria possível através de financiamento de instituições como Estado do Rio Grande do Sul, CDI (Comitê de Democratização da Informática), empresas privadas interessadas no desenvolvimento da região ou ainda doações de cidadãos entusiasmados com nossa causa.
De qualquer maneira, daremos início ao projeto através da realização das palestras.

References

1. mailto:hugo@domain.com.br
2. http://www.domain.com.br/clientes/hugo/pag3.html
3. mailto:acamargo@atlas.ucpel.tche.br

10 REFERÊNCIAS BIBLIOGRÁFICAS

[Ald 00] ALDRICH, Douglas F. Dominando o Mercado Digital. São Paulo: Makron Books, 2000.

[ALL 00] ALLIANCE website. Disponível por WWW em `http://www.allos.org`, [s.d.]. (jun. 2000)

[AMA 00] AMAYA website. Disponível por WWW em `http://www.w3.org/Amaya/`, [s.d.]. (nov. 2000)

[APA 00] APACHE Project website. Disponível por WWW em `http://www.apache.org`, [s.d.]. (jun. 2000)

[APP 00] APPWATCH.COM website. Disponível por WWW em `http://appwatch.zdnet.com`, [s.d.]. (abr. 2000)

[ARM 00] ARMADILLO website. Disponível por WWW em `http://www.gzilla.com/index.html`, [s.d.]. (nov. 2000)

[Bar s.d.] BARROS Pablo Fernando do Rêgo. UML Linguagem de Modelagem Unificada. Aracaju: Universidade Tirantes, [s.d.] 74 p. (Monografia, Centro de Ciências Formais e Tecnológicas)

[BBC et al. 00] BROD Cesar Augusto de Azambuja, BUSCH JR. Carlos Alberto, CASTRO Maurício de et al. SAGU-FMS - Sistema Aberto de Gestão Unificada para Escolas de Ensino Fundamental, Médio e Superior. In: 1º FÓRUM INTERNACIONAL SOFTWARE LIVRE 2000, 1, 2000, Porto Alegre. Anais Workshop sobre Software Livre. Porto Alegre: Sociedade Brasileira de Computação, 2000. p. 65-68

[BCu 89] BECK, Kent, CUNNINGHAM, Ward. *A laboratory for teaching object-oriented thinking.* Disponível por WWW em `http://c2.com/doc/oopsla89/papaer.html`, 1989. (ago. 2000)

[Ber 00] BERARD, Edward V. *What is a methodology?* Disponível por WWW em `http://www.toa.com/pub/methodology-article.txt`, [s.d.]. (ago. 2000)

[BSc 98] BARROSO, Natacha Güell, SCHWABE, Daniel. *Projeto de Navegação em Aplicativos Hipermídia Orientado ao Usuário.* Disponível por

WWW em http://www.telemidia.puc-rio.br/oohdm/Mod_Navegacao.pdf, 1998. (out. 2000)

[BSD 00] FREEBSD website. Disponível por WWW em http://www.freebsd.org, [s.d.]. (jun. 2000)

[CHI 00] CHIMERA website. Disponível por WWW em http://www.cs.unlv.edu/chimera/, [s.d.]. (nov. 2000)

[Coe 00] COELHO, Marcelo de Miranda. *O uso de estruturas navegacionais e vistas abstratas de dados no OOHDM e conceitos de objetos multimídia para a construção de uma aplicação*. Disponível por WWW em http://tathy.comp.ita.cta.br/~coelho/tgonline/cap1&2.htm, [s.d.]. (ago. 2000)

[Cos 99] COSTA, Cristiano André da. *Normas para a apresentação do Projeto de Graduação na Escola de Informática da UCPel*. Pelotas: Educat, 1999. 51 p.

[Cou 97] COUGO, Paulo. Modelagem conceitual e projeto de bancos de dados. Rio de Janeiro: Campus, 1997. 281 p.

[Cox 90] COX, Brad J. *Planning the Software Industrial Revolution*. Disponível por WWW em http://www.virtualschool.edu/cox/CoxPSIR.html, 1990. (ago. 1999)

[CYo 93] COAD Peter, YOURDON Edward. Projeto baseado em objetos. 4a. ed. Rio de Janeiro: Campus, 1993. 191 p.

[CZL 00] CIOCIOLA, Débora, ZAMBALDE, André Luiz, LOPES, Alfredo Scheid. *Desenvolvimento de um aplicativo hipermídia para o setor agropecuário: OOHDM e autoria toolbook.* Disponível por WWW em http://agrosoft.com/ag97/papers/c4w1730.htm, [s.d.]. (ago. 2000)

[DIA 00] DIA website. Disponível por WWW em http://www.lysator.liu.se/~alla/dia/, [s.d.]. (jun. 2000)

[DOS 00] FREEDOS website. Disponível por WWW em http://www.freedos.org, [s.d.]. (jun. 2000)

[EBi 00] ESTILO BibTex com as normas da ABNT. Disponível por WWW em http://www.if.ufrgs.br/hadrons/abnt/abnt.html, [s.d.]. (set. 2000)

[ERO 00] EROS website. Disponível por WWW em http://www.eros-os.org, [s.d.]. (jun. 2000)

[Fer 93] FERREIRA, Aurélio Buarque de Holanda. Minidicionário da Língua Portuguesa. 4a. ed. Rio de Janeiro: Nova Fronteira, 1993. 577 p.

[FOS 00] FREEOS website. Disponível por WWW em http://www.freeos.com, [s.d.]. (abr. 2000)

[Fow 00] FOWLER, Martin. *Techniques for Object Oriented Analysis and Design*. Disponível por WWW em http://www2.awl.com/cseng/titles/0-201-89542-0/techniques/, [s.d.] (ago. 2000)

[Fow 00a] FOWLER, Martin. *UML Essencial*. Porto Alegre: Bookman, 2000. 169p.

[FRE 00] FRESHMEAT.NET website. Disponível por WWW em http://freshmeat.net, [s.d.]. (abr. 2000)

[Fur 98] FURLAN, José Davi. Modelagem de Objetos através da UML - The Unified Modeling Language. São Paulo: Makron Books, 1998. 325 p.

[GAL 00] GALEON website. Disponível por WWW em http://galeon.sourceforge.net/, [s.d.]. (nov. 2000)

[Gas 00] GASS, Elvino Bohn. *Software Livre*. 1ª ed. Porto Alegre: Corag, 2000. 25 p.

[GHJ et al 00] GAMMA, Erich, HELM, Richard, JOHNSON, Ralph, VLISSIDES, John. *Padrões de Projeto: soluções reutilizáveis de software orientado a objetos*. Porto Alegre: Bookman, 2000. 364 p.

[GHT 00] GHTTPD website. Disponível por WWW em http://www.gaztek.co.uk/ghttpd, [s.d.]. (jun. 2000)

[GJS 00] GNUJSP website. Disponível por WWW em http://www.klomp.org/gnujsp, [s.d.]. (out. 2000)

[GNO 00] GNOME Desktop Environment website. Disponível por WWW em http://www.gnome.org, [s.d.]. (out. 2000)

[GNU 00] GNU Project website. Disponível por WWW em http://www.gnu.org, [s.d.]. (mai. 2000)

[HB 00] HB Pre-processor website. Disponível por WWW em http://bachue.com/hb/hb.cgi, [s.d.]. (jun. 2000)

[HUR 00] GNU/HURD website. Disponível por WWW em http://www.gnu.ai.mit.edu/software/hurd, [s.d.]. (jun. 2000)

[INT 00] INTERBASE website. Disponível por WWW em http://www.interbase.com, [s.d.]. (jun. 2000)

[JAP 00] JAPHAR website. Disponível por WWW em http://www.japhar.org, [s.d.]. (out. 2000)

[KAF 00] KAFFE website. Disponível por WWW em http://www.kaffe.org, [s.d.]. (out. 2000)

[KDE 00] K Desktop Environment website. Disponível por WWW em http://www.kde.org, [s.d.]. (out. 2000)

[Ken 89] KENT, Beck. A Laboratory For Teaching Object-Oriented Thinking. Disponível por WWW em http://c2.com/doc/oopsla89/paper.html, 1989. (ago. 1999)

[KON 00] KONQUEROR website. Disponível por WWW em http://www.konqueror.org, [s.d.]. (nov. 2000)

[LAG 00] LAGO website. Disponível por WWW em http://lago.sourceforge.net, [s.d.]. (jun. 2000)

[LAl 97] LOBATO Daniel Corrêa, ALVES Ronnie Cley de Oliveira. Análise e Projeto de Sistemas Método Shlaer-Mellor para OOA e OOD. Belém: Universidade Federal do Pará, 1997. 19 p. (Dissertação, Curso de Especialização em Análise de Sistemas)

[LEA 00] LEAP website. Disponível por WWW em http://leap.sourceforge.net, [s.d.]. (jun. 2000)

[LNX 00] LINUX website. Disponível por WWW em http://www.linux.org, [s.d.]. (jun. 2000)

[Loh 96] LOH, Stanley. Ambientes de desenvolvimento de software e ferramentas CASE. [s.l.]: Universidade Católica de Pelotas, 1996. 27 p.

[MID 00] MIDGARD project website. Disponível por WWW em http://www.midgard-project.org, [s.d.]. (jun. 2000)

[MNX 00] MINIX website. Disponível por WWW em http://www.cs.vu.nl/~ast/minix.html, [s.d.]. (jun. 2000)

[MOM 00] MOMJIAM, Bruce. *PostgreSQL Introduction and Concepts*. Disponível por WWW em `http://www.postgresql.org/docs/aw_pgsql_book/aw_pgsql_book.pdf`, [s.d.]. (out. 2000)

[MOU 98] MOURA, Isabela. *Um Ambiente para o Suporte ao Projeto e Implementação de Sistemas de Informação Baseados na WWW*. Disponível por WWW em `http://www.telemidia.puc-rio.br/oohdm/oohdmweb/docs/TeseIsabela.zip`, 1998. (out. 2000)

[MOZ 00] MOZILLA website. Disponível por WWW em `http://www.mozilla.org`, [s.d.]. (nov. 2000)

[MRa 98] MEDEIROS, Carlos Alberto Barbosa, RASEIRA, Maria do Carmo Bassols. *A cultura do Pessegueiro*. Pelotas: Embrapa-CPACT, 1998. 350p.

[MYS 00] MYSQL website. Disponível por WWW em `http://www.mysql.com`, [s.d.]. (jun. 2000)

[NET 00] NETBSD website. Disponível por WWW em `http://www.netbsd.org`, [s.d.]. (jun. 2000)

[OPE 00] OPENBSD website. Disponível por WWW em `http://www.openbsd.org`, [s.d.]. (jun. 2000)

[PCD et al 00] PARERA Antônio João, COPPETTI Clarice, DUTRA Claudio et al. DINHEIRO PARA QUEM PRECISA; Alguns motivos para o Projeto Software Livre do Rio Grande do Sul. In: 1º FÓRUM INTERNACIONAL SOFTWARE LIVRE 2000, 1, 2000, Porto Alegre.Anais Workshop sobre Software Livre... Porto Alegre: Sociedade Brasileiro de Computação, 2000. p. 9-13

[PGS 00] POSTGRESQL website. Disponível por WWW em `http://www.postgresql.org`, [s.d.]. (jun. 2000)

[PHP 00] PHP website. Disponível por WWW em `http://php.net`, [s.d.]. (jun. 2000)

[Piz 98] PIZZOL, Andréa Miranda. *Um Framework para Implementação na WWW de Aplicações Hipermídia Modeladas com OOHDM*. Disponível por FTP anônimo em webengs.lmf-di.puc-rio.br, no arquivo /oohdm/oohdmjava/tese.zip, 1998. (out. 2000)

[POL 00] PolyJSP website. Disponível por WWW em `http://www.plenix.org/polyjsp/`, [s.d.]. (out. 2000)

[RAT 97] RATIONAL SOFTWARE CORPORATION. *UML Notation Guide version 1.1*. Disponível por WWW em `http://www.platinum.com/corp/uml/nota_11.pdf`, 1997. (set. 2000)

[RAT 97a] RATIONAL SOFTWARE CORPORATION. *UML Summary version 1.1*. Disponível por WWW em `http://www.platinum.com/corp/uml/summ_11.pdf`, 1997. (set. 2000)

[RED 00] REDHAT Linux Distribuition Website. Disponível por WWW em `http://www.redhat.com`, [s.d.]. (out. 2000)

[RES 00] RESIN website. Disponível por WWW em `http://www.caucho.com/products/resin`, [s.d.]. (out. 2000)

[RLS 00] ROSSI, Gustavo, LYARDET, Fernando, SCHWABE, Daniel. *Patterns for E-commerce applications*. Disponível por WWW em `http://www.inf.puc-rio.br/~schwabe/papers/Europlop00.pdf`, 2000. (ago. 2000).

[Ros 96] ROSSI, Gustavo. *Um Método Orientado a Objetos para o Projeto de Aplicações Hipermídia.*. Disponível por WWW em `http://www-di.inf.puc-rio.br/~schwabe/Tese_Rossi/`, 1996. (ago. 2000).

[RSL et al 95] ROSSI, Gustavo, SCHWABE, Daniel, LUCENA, C.J.P., COWAN, D.D. *An Object-Oriented Model for Designing the Human-Computer Interface Of Hypermedia Applications.* Disponível por FTP anônimo em `ftp.inf.puc-rio.br`, no arquivo `/pub/docs/techreports/95_07_rossi.ps.gz`, 1995. (out. 2000)

[RSL 99] ROSSI, Gustavo, SCHWABE, Daniel, LYARDET, Fernando. *Web Application Models are more than Conceptual Models*. Disponível por WWW em `http://www.inf.puc-rio.br/~schwabe/papers/WWWCM99.pdf`, 1999. (set. 2000)

[RSL 00] ROSSI, Gustavo, SCHWABE, Daniel, LYARDET, Fernando. *Abstraction and Reuse Mechanisms in Web Application Models*. Disponível por WWW em `http://www.inf.puc-rio.br/~schwabe/papers/WWWCM002Expanded.pdf`, 2000. (ago. 2000).

[Sch 91] SCHULTZ, Ronald. OO project management. Disponível por WWW em `http://www.cs.ncl.ac.uk/people/chris.`

holt/home.formal/workroom/library/OO.project.management.htm, 1991. (ago. 1999)

[Sch 99] SCHARL, Arno. *A Conceptual, User-Centric Approach to Modeling Web Information Systems*. Disponível por WWW em http://ausweb.scu.edu.au/aw99/papers/scharl/paper.html, 1999. (ago. 2000)

[Sch 99] SCHWABE, Daniel. *"Just add Water" Applications: Hypermedia Application Frameworks*. Disponível por WWW em http://www.inf.puc-rio.br/~schwabe/papers/SchwabeHT99Workshop.pdf, 1999. (ago. 2000)

[SGu 00] SCHWABE, Daniel, GUËLL, Natacha. *Projeto de Navegação, baseado em cenários, use cases e diagramas de interação do usuário (UIDs)*. Disponível por WWW em http://www.telemidia.puc-rio.br/oohdm/Mod_Navegacao.pdf, [s.d.]. (out. 2000)

[SK8 00] SKATEBROWSER website. Disponível por WWW em http://freshmeat.net/projects/skatebrowser/homepage/, [s.d.]. (nov. 2000)

[SPo 98] SCHWABE, Daniel, PONTES, Rita de Almeida. *OOHDM-WEB: Rapid Prototyping of Hypermedia Applications in the WWW*. Disponível por WWW em http://www-di.inf.puc-rio.br/~schwabe/papers/MCC-08-98.pdf.gz, 1998. (out. 2000)

[SRE et al. 00] SCHWABE, Daniel, ROSSI, Gustavo, ESMERALDO, Luiselena et al. *Web Design Frameworks: An approach to improve reuse in Web applications*. Disponível por WWW em http://www.inf.puc-rio.br/~schwabe/papers/WWW9WebEngineering.pdf, [s.d.]. (set. 2000)

[SRo 98] SCHWABE, Daniel, ROSSI, Gustavo. *Developing Hypermedia Applications using OOHDM*. Disponível por WWW em http://www.inf.puc-rio.br/~schwabe/papers/ExOOHDM.pdf.gz, 1998. (ago. 2000).

[Sey 00] SEYBOLD, Patricia B. CLIENTES.COM. São Paulo: Makron Books, 2000.

[SOF 00] SOFTWARE LIVRE - PROJETO DE LEI No. 59/2000 DEPUTADO ELVINO BOHN BASS - PT. Porto Alegre: [s.n.], Mai. 2000.

[SQL 00] GNU SQL Server website. Disponível por WWW em http://www.ispras.ru/~gsql, [s.d.]. (jun. 2000)

[Sun 00] SUN Microsystems Inc. *Java Servlet and JavaServer Pages Technology: Comparing Methods for Server Side Dynamic Content*. Disponível por WWW http://javasoft.com/products/jsp/pdf/jspservlet.pdf, 2000. (out. 2000)

[SVi 99] SCHWABE, Daniel, VILAIN, Patrícia. *Notação da Metodologia OOHDM versão 1.1*. Disponível por WWW em http://www.telemidia.puc-rio.br/oohdm/notacao_OOHDM_UML.zip, 1999. (out. 2000)

[THT 00] THTTPD website. Disponível por WWW em http://www.acme.com/software/thttpd, [s.d.]. (jun. 2000)

[TOM 00] TOMCAT-JAKARTA website. Disponível por WWW em http://jakarta.apache.org/tomcat/index.html, [s.d.]. (out. 2000)

[Ven 99] VENETIANER, Tom. Como Vender seu Peixe na Internet. Rio de Janeiro: Campus, 1999.

[VSS 00] VILAIN, Patrícia, SCHWABE, Daniel, SOUZA, Clarisse Sieckenius de. *A Diagrammatic Tool for Representing User Interaction in UML*. Disponível por WWW em http://www.inf.puc-rio.br/~schwabe/papers/UML2000.pdf, 2000. (agosto 2000).

[XIT 00] XITAMI website. Disponível por WWW em http://www.xitami.com, [s.d.]. (jun. 2000)

[ZOP 00] ZOPE website. Disponível por WWW em http://www.zope.org, [s.d.]. (jun. 2000)