MÍRIAN CRISTINA DE OLIVEIRA

REENGENHARIA DE SOFTWARE: REENGENHARIA DE UM SISTEMA LEGADO BASEADO EM PADRÕES

Trabalho de conclusão de curso apresentado ao Curso de Ciência da Computação.

UNIVERSIDADE PRESIDENTE ANTÔNIO CARLOS

Orientadora: Profa. Lorena Sophia
Co-orientador: Prof. Eduardo Bhering

BARBACENA
2003

MÍRIAN CRISTINA DE OLIVEIRA

REENGENHARIA DE SOFTWARE: REENGENHARIA DE UM SISTEMA LEGADO BASEADO EM PADRÕES

Este trabalho de conclusão de curso foi julgado adequado à obtenção do grau de Licenciado em Ciência da Computação e aprovado em sua forma final pelo Curso de Ciência da Computação da Universidade Presidente Antônio Carlos.

Barbacena – MG, 08 de julho de 2003.

_____

Prof. Lorena Sophia - Orientador do Trabalho

_____

Prof. Eduardo Bhering - Membro da Banca Examinadora

_____

Prof. Elio Lovisi Filho - Membro da Banca Examinadora

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me dado a oportunidade de estar aqui e ter me dado forças para continuar este curso, quando muitos problemas me fizeram ter vontade de parar. A meus orientadores, o meu muito obrigado pela compreensão, dedicação e incentivo durante todo o semestre. E a todos os que direta ou

indiretamente colaboraram para a efetivação deste trabalho. Obrigado.

RESUMO

Este trabalho apresenta um processo de reengenharia de um sistema legado desenvolvido em Clipper para um sistema orientado a objetos, escrito em linguagem Delphi, baseado nos padrões da Família de Padrões de Reengenharia Orientada a Objetos de Sistemas Legados Procedimentais, proposta por Edson Luiz Recchia (Universidade Federal de São Carlos/SP). A etapa de engenharia reversa foi feita com a utilização da ferramenta *Rational Rose* para a elaboração dos modelos orientados a objetos e a etapa de engenharia progressiva foi feita com a utilização da linguagem orientada a objetos *Delphi*.

Palavras-chave: Reengenharia, Engenharia Reversa, Sistema Legado.

SUMÁRIO

1 MOTIVOS QUE LEVARAM À ABORDAGEM DO TEMA.....10

1.1 OBJETIVO.....12
1.2 ORGANIZAÇÃO DO TRABALHO.....13

2 REVISÃO BIBLIOGRÁFICA.....14

2.1 O QUE É A REENGENHARIA.....14
2.2 A ENGENHARIA REVERSA E A ENGENHARIA AVANTE.....17
2.2.1 CATEGORIAS DA ENGENHARIA REVERSA.....21
2.2.2 ENGENHARIA AVANTE.....23
2.3 PLANEJAMENTO PARA REENGENHARIA.....24
2.4 O MÉTODO FUSION/RE.....26
2.5 A FAMILIA DE PADRÕES DE REENGENHARIA ORIENTADA A OBJETOS – FAPRE/OO.....27
2.6 CONCLUSÕES.....32

3 REENGENHARIA DO SISTEMA DE UMA IMÓBILIÁRIA BASEADA NOS PADRÕES DA FAMÍLIA DE PADRÕES DE REENGENHARIA ORIENTADA A OBJETOS – FAPRE/OO.....33

3.1 INTRODUÇÃO.....33
3.2 O SISTEMA.....34
3.3 OS PADRÕES.....34
3.4 A ENGENHARIA REVERSA DO SISTEMA DE UMA IMOBILIÁRIA UTILIZANDO OS PADRÕES DA FAMÍLIA DE PADRÕES DE REENGENHARIA PARA SISTEMAS LEGADOS EM CLIPPER.....35
Cluster – modelar os dados do legado:.....35
1. Nome: Iniciar a Análise dos Dados:.....35
2. Nome: Definir Chaves:.....38
3. Nome: Identificar Relacionamentos:.....40
4. Nome: Criar Visão OO dos Dados:.....43
Cluster – modelar a funcionalidade do sistema:.....45
5. Nome: Obter Cenários:.....45
6. Nome: Construir Diagramas de Use Cases:.....48
7. Nome: Elaborar a descrição de Use Cases:.....50
8. Nome: Tratar Anomalias:.....52
Cluster – modelar o sistema orientado a objetos:.....58
9. Nome: Definir as Classes:.....58
10. Nome: Definir Atributos:.....61

Figura 3.10 – Diagrama de Classes (Definição dos Atributos das Classes).....63
11. Nome: Analisar Hierarquias:.....63
12. Nome: Definir Métodos:.....69
13. Nome: Construir Diagramas de Seqüência:.....72
3.5 CONCLUSÃO:.....73

4 PROCESSO DE ENGENHARIA AVANTE DO SISTEMA PARA CONTROLE DE LOCAÇÕES DE UMA IMOBILIÁRIA BASEADO EM PADRÕES.....74

Cluster – Gerar o sistema orientado a objetos:.....75
1. Nome: Definir a Plataforma:.....75
2. Nome: Converter o Banco de Dados:.....77
O Banco de dados do Sistema não foi efetivamente implementado, visto que o objetivo principal deste projeto não era a implementação e sim apenas o processo de engenharia reversa do mesmo.....77
3. Nome: Implementar os Métodos:.....77
4. Nome: Realizar melhorias na Interface:.....78

5 CONCLUSÕES.....81

5.1 REVISÃO:.....81
5.2 CONSIDERAÇÕES FINAIS:.....82
5.3 TRABALHOS FUTUROS:.....83

REFERÊNCIAS BIBLIOGRÁFICAS.....84

ANEXO A/TABELAS E FIGURAS DA ETAPA DE ENGENHARIA REVERSA UTILIZANDO OS PADRÕES DA FAMÍLIA DE PADRÕES DE REENGENHARIA ORIENTADA A OBJETOS.....86

PADRÃO 1: INICIAR A ANÁLISE DOS DADOS:.....86
FIGURA 1: TABELA PROGRAMAS X ARQUIVOS.....86
PADRÃO 2: DEFINIR CHAVES:.....89
FIGURA 2 – ARQUIVOS DE DADOS COM SEUS RESPECTIVOS ARQUIVOS DE ÍNDICES:.....89
FIGURA 3- TABELA ENTIDADES X CHAVES.....90
PADRÃO 6: CONSTRUIR DIAGRAMAS DE USE CASES:.....91
FIGURA 4 – DIAGRAMAS DE USE CASES PARA OS SEIS CENÁRIOS OBTIDOS A PARTIR DO ESTUDO DO PADRÃO 5:.....91
PADRÃO 7: ELABORAR A DESCRIÇÃO DE USE CASES:.....94
FIGURA 5 – DESCRIÇÃO DOS DIAGRAMAS DE USE CASES:.....94
PADRÃO 13: CONSTRUIR DIAGRAMAS DE SEQUÊNCIA:.....114
FIGURA 6 – DIAGRAMA DE SEQUÊNCIA DOS USE CASES:.....114

FIGURAS DA ETAPA DE ENGENHARIA PROGRESSIVA DO SISTEMA DE CONTROLE DE LOCAÇÕES DE IMÓVEIS BASEADO EM CONCEITOS DE USABILIDADE.....131

PADRÃO 4: REALIZAR MELHORIAS NA INTERFACE:.....131
FIGURA 1: TABELA PROGRAMAS X ARQUIVOS.....131
FIGURA 2:TELA MENU INCLUSÕES: 132
FIGURA 3: TELA INCLUSÕES LOCADORES:.....133
FIGURA 4: TELA INCLUSÕES IMÓVEIS: 135
FIGURA 5: TELA INCLUSÕES LOCATÁRIOS: 136
FIGURA 6: TELA ALTERAÇÕES LOCADORES: 137
FIGURA 7: TELA ALTERAÇÕES IMÓVEIS: 137
FIGURA 8: TELA ALTERAÇÕES LOCATÁRIOS: 137
FIGURA 9: TELA CONSULTAS LOCADORES:.....137
FIGURA 10: TELA CONSULTAS IMÓVEIS ALUGADOS:.....139
FIGURA 11: TELA CONSULTAS IMÓVEIS DISPONÍVEIS: 140
FIGURA 12: TELA CONSULTAS LOCATÁRIOS: 140
FIGURA 13: TELA CONSULTAS AGENDA DEPÓSITOS: 141
FIGURA 14: TELA CONSULTAS ANIVERSARIANTES DO MÊS: 142
FIGURA 15: TELA CONTRATOS: 144
FIGURA 16: TELA COMUNICADOS:.....145
FIGURA 17: TELA LOCATÁRIOS EM DÉBITO:.....146
FIGURA 18: TELA DEMONSTRATIVO DE DEPÓSITO (SUBMENU):.....148

FIGURA 19: TELA LISTAGEM DOS LOCADORES:.....149
FIGURA 20:TELA DEPÓSITOS NÃO REALIZADOS: 149
FIGURA 21: TELA COMPROVANTE IMPOSTO DE RENDA: 150
FIGURA 22: TELA CARTA COBRANÇA LOCATÁRIO:.....151
FIGURA 23: TELA LISTAGEM LOCATÁRIOS:.....152
FIGURA 24: TELA CONTRATOS VENCIDOS:.....153
FIGURA 25: TELA CARTA PEDIDO DE IMÓVEL:.....153
FIGURA 26: TELA CARTA DE DISTRATO:.....154
FIGURA 27: TELA CARTA COBRANÇA FIADOR:.....155
FIGURA 28: EMISSÃO ETIQUETA INQUILINO:.....156
FIGURA 29: EMISSÃO ETIQUETA PROPRIETÁRIO:.....157
FIGURA 30: TELA ORGANIZAR ARQUIVOS:.....158
FIGURA 31: TELA TIPO DE REAJUSTE:.....159

1 MOTIVOS QUE LEVARAM À ABORDAGEM DO TEMA

Muitos softwares são desenvolvidos sem nenhuma documentação escrita, pois o cliente precisa que ele esteja funcionando em pouquíssimo tempo, e esse fato normalmente obriga os desenvolvedores a não executarem a fase de análise e projeto, e irem direto para a fase de implementação.

Com o passar dos anos, problemas com o uso do software começam a surgir, pois ele já está se tornando obsoleto, suas funções já não atendem totalmente os anseios da firma. Surge então a necessidade de manutenção do mesmo, pois como o sistema já possui um bom tempo de uso e seus usuários já estão acostumados, é mais viável mantê-lo do que trocá-lo.

Mas, mantê-lo pode ser dispendioso e trabalhoso, visto que este não possui nenhuma documentação, e sua manutenção terá que ser feita através de entrevistas com os usuários, exame de código e interface, e outras técnicas.

A reengenharia de software trata o assunto de manutenção de sistemas que foram desenvolvidos de forma errônea, ou que já estão precisando ser adaptados às necessidades do usuário, pois sistemas que não sofrem atualizações estão fadados a serem abandonados, visto perderem utilidade no contexto para que foi desenvolvido.

Alguns grandes problemas em executar manutenção em softwares podem ser:

- Desestruturação e dificuldade de entendimento do código, visto que este não possui nenhuma explicação;
- Normalmente, quem desenvolveu o sistema não é quem realiza a manutenção do mesmo; sendo assim o processo de entendimento é mais demorado; e,
- Nenhuma documentação, ou documentação desatualizada.

A idéia da reengenharia é que, a partir do software obsoleto, seja feito um estudo, mantendo sua base de dados, e ao final seja feito um novo software com as mesmas características do antigo, só que desta vez, mais preparado para futuras atualizações.

A linguagem usada para o desenvolvimento do sistema também será uma linguagem mais moderna que a antiga, resultando num programa mais fácil de ser manuseado pelo usuário, ficando este pronto a qualquer modificação, talvez até para uma outra plataforma, de acordo com a necessidade do mesmo.

O campo da reengenharia tem crescido muito, visto que a maioria dos sistemas utilizada nas indústrias foi desenvolvida sem nenhum estudo, e estes estão em uso há mais de 10 anos, sendo necessárias adaptações, como mudança de interfaces, que agora deverão ser gráficas, mudança para desenvolvimento orientado a objeto, etc. Sistemas desenvolvidos atualmente também estão sujeitos à reengenharia, dessa forma sempre estarão atualizados e atendendo as necessidades dos usuários.

No caso específico da reengenharia do sistema escolhido, este se deu pelo fato de que o sistema é de suma importância para o cliente, já possui uma base de dados suficientemente grande para ser modificado por um outro sistema, e não poderia ser trocado por um utilitário que não atendesse às necessidades do mesmo. Sendo feita a reengenharia sobre o sistema, o cliente terá um novo sistema que não lhe será tão novo assim, visto que as suas funções principais serão mantidas.

1.1 OBJETIVO

O objetivo deste trabalho é, a partir da reengenharia de software, reconstruir um sistema que seja de utilidade para alguma organização, mas que não deve ser substituído, visto ser importante para a mesma.

A reengenharia envolve duas etapas: **a engenharia reversa**, que é justamente o contrário da engenharia de software, onde a partir do estudo do sistema pronto, faz-se uma análise do que é realmente necessário, o que está sendo usado e o que se tornou obsoleto, e reúne-se tudo isto em uma documentação formal; e **a engenharia avante**, onde se faz o papel de engenharia de software, dessa forma refazendo e reimplementando o sistema com o que é necessário e útil ao usuário.

A partir de um sistema escolhido será feito todo o trabalho de reengenharia, mas apenas a parte de análise e desenvolvimento do projeto; a implementação do novo código ficará de fora do trabalho, visto não ser a programação o objetivo aqui explicitado.

O sistema escolhido foi o sistema de controle de locação de uma imobiliária da cidade de Barbacena. Este sistema é muito importante para o cliente (visto já estar implantado há muitos anos e possuir muitas informações que não podem ser desprezadas), mas justamente por ser mais antigo, já não condiz com todos os interesses do mesmo. É um programa que foi escrito em Clipper (versão Summer 87) e possui aproximadamente 8.000 mil linhas de código.

Para executar a engenharia reversa do sistema utilizou-se padrões da FaPRE/OO (Família de Padrões de Reengenharia Orientada a Objetos), uma metodologia que possibilita a transformação de sistemas procedimentais em sistemas orientados a objeto. E para executar a fase de engenharia progressiva utilizou-se a ferramenta Rational Rose. As novas interfaces implementadas com base no estudo realizado, foram feitas na linguagem Delphi, por ser uma linguagem fácil de ser entendida e mantida posteriormente.

O objetivo principal e final deste trabalho será realizar a reengenharia de software no sistema denominado Sistema para Controle de Locações de Imóveis, visando a modelagem

completa deste (etapa de engenharia reversa), com pretensão de uma futura implementação do mesmo (etapa de engenharia progressiva), para que este tenha de fato passado por todo o processo da reengenharia

1.2 ORGANIZAÇÃO DO TRABALHO

O trabalho será organizado em capítulos. A este primeiro capítulo coube uma introdução sobre o projeto, seu objetivo e motivações.

O capítulo 2 trata a revisão bibliográfica, os principais conceitos de reengenharia, engenharia reversa e engenharia avante e a Família de Padrões de Reengenharia Orientado a Objetos FaPRE/OO (Recchia et al 2002).

O capítulo 3 trata especificamente sobre a engenharia reversa do sistema, com aplicação da família de padrões escolhida.

O capítulo 4 descreve a engenharia progressiva do sistema, baseado nos modelos obtidos no capítulo 3.

O capítulo 5 nos traz as conclusões tiradas através de todo o projeto e as propostas para trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Antes de seguir adiante, será feita uma breve pausa, para citar alguns conceitos e definições sobre a reengenharia de software, engenharia reversa, Método Fusion e finalmente a FaPRE/OO.

2.1 O QUE É A REENGENHARIA

Michael Hammer, o pai da reengenharia, diz que a reengenharia não é uma ficção, está bem entranhada na realidade. De fato esse processo está relacionado com a construção de algo do mundo real, reformulando-o e entregando-o com uma qualidade melhor que a anterior.

A reengenharia pode ser aplicada não só a sistemas informatizados, mas também em organizações e locais de trabalho, o que vem sendo executado e muito bem aceito, por causa das suas melhorias.

Uma definição de reengenharia de processos dada por Hammer e Champy [1993], está a seguir: “reengenharia é o repensar fundamental e o reprojeto radical dos processos de uma organização, para realizar uma dramática melhoria em termos de performance, custo, qualidade e serviços”.

Mas o próprio Hammer reconhece que dentro de uma empresa há inimigos à reengenharia de processos; “são os que têm medo da mudança, todos os que querem que as coisas fiquem como estão. Dentro das organizações, eu diria que há dois tipos de inimigos. Por um lado, o” top management “que não sabe realmente exercer a liderança, os gestores do topo que não sabem ser líderes e que, no fundo, não passam de meros “administradores” de negócios. Se não atuarem como líderes, não conseguirão sucesso. Por outro, o “middle management”. Para estes, manter o velho sistema é o ideal, pois é o mais cômodo. Mas, eu diria que, se houver uma forte liderança por parte dos gestores do topo, os do meio cooperarão”.

A reengenharia de processos consiste em remodelar uma organização, já na reengenharia de sistemas, o que se faz é remodelar um sistema, considerado já obsoleto, mas que não se pode, ou não é viável, trocá-lo, por ser de grande importância para a instituição.

Estão relatadas a seguir algumas definições de reengenharia de software, dadas por outros autores:

Para Chikofsky e Cross [1990], IEEE CS-TCSE [1997] e GT-REG [1998], a reengenharia é a reconstrução ou renovação de um software. Através do exame e alteração deste, pode-se reconstruí-lo em uma nova forma. Um processo de reengenharia normalmente inclui as etapas de engenharia reversa e engenharia avante.

Para Warden [1992], a reengenharia de software tem o objetivo de proporcionar alguma melhoria em um sistema, com alterações significantes, mas conservando-se suas funcionalidades principais. Apenas extrair as descrições de uma aplicação através do código e reescreve-la em uma outra linguagem, não é considerada reengenharia pelo autor, e considerada transcrição de código. Este autor também considera que a reengenharia possui duas etapas: a engenharia reversa e a engenharia progressiva.

Premerlani e Blaha [1994] citam que o objetivo da reengenharia é reutilizar automaticamente os esforços de desenvolvimento passados, objetivando reduzir custos de manutenção e melhoria na flexibilidade do software.

Segundo Pressman [1995], a reengenharia é considerada uma reconstrução de um sistema, a partir de sua documentação e exame do mesmo, sem alterar suas funcionalidades, apenas adicionando o que se julga necessário, no sentido de devolve-lo ao usuário, com uma melhor qualidade.

Das definições acima fica claro de perceber a diferença entre reengenharia e engenharia de software; e ainda, a diferença entre reengenharia e engenharia reversa. A engenharia de software busca a execução de um sistema totalmente novo, sem nada ainda formal descrito sobre ele, enquanto que a reengenharia busca exatamente o contrário: a partir do software, já existente, formular uma nova versão, mais eficiente que a anterior, e com uma documentação de qualidade, que possibilite futuras manutenções.

Falando agora sobre a diferença entre reengenharia e engenharia reversa, tem-se que a segunda é um subconjunto da primeira, visto que ela é feita a partir do exame minucioso do código do produto, e será de grande valia no contexto total da reengenharia, que é considerada mais abrangente. A figura 2.1 ilustra bem essa questão.

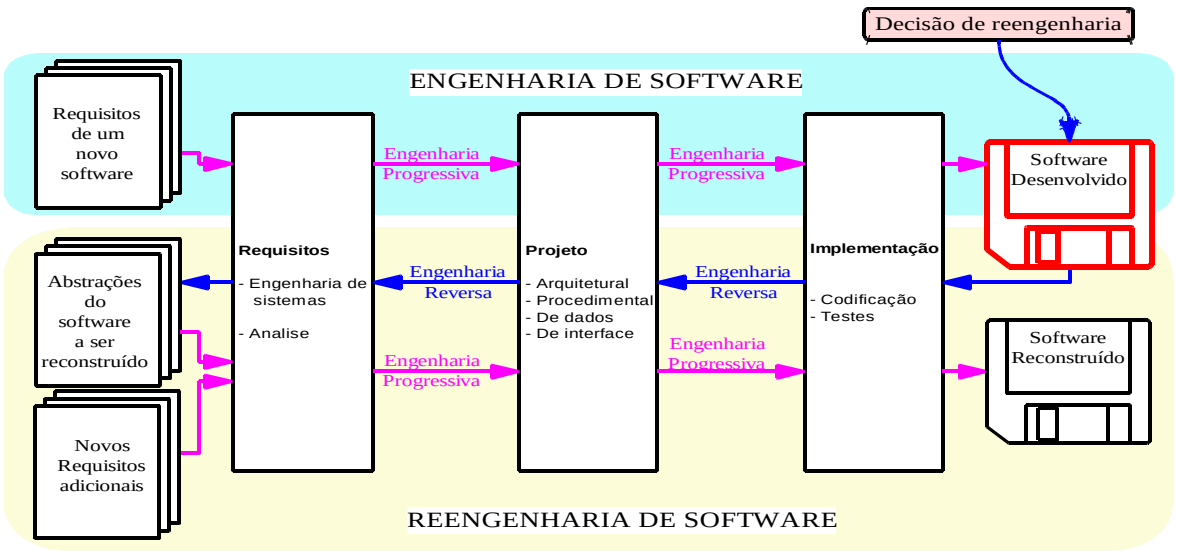


Figura 2.1 – Engenharia e Reengenharia de Software [Quinaia, 1998]

A figura acima representa todas as etapas da reengenharia, o entendimento do programa, análise e projeto, com o intuito de fornecer um nível de abstração maior que o software antigo.

A reengenharia normalmente é aplicada a sistemas orientados a função, sistemas procedimentais, mudando-o para uma plataforma orientada a objetos, na qual futuras manutenções serão mais fáceis de serem aplicadas.

2.2 A ENGENHARIA REVERSA E A ENGENHARIA AVANTE

Um software possui um ciclo de vida. Este ciclo de vida pode ser de vários modelos: em cascata, evolutivo, espiral, entre outros. Independentemente do ciclo de vida adotado, este software passa pela especificação de requisitos, análise e implementação do projeto. Nessas etapas, quando se passa de especificação para análise e de análise para implementação, tem-se uma diminuição do nível de abstração.

A *abstração* é a propriedade de concentrar-se nos assuntos principais do sistema, sem necessidade de se aprofundar em questões de implementação.

O que a engenharia reversa faz é, justamente, partir de um baixo nível de abstração para um alto nível, pois quando é iniciada a manutenção tem-se apenas o sistema antigo, e então vai se criando alguma documentação, até chegar à engenharia progressiva, para o desenvolvimento de um novo software.

Inicialmente a engenharia reversa era utilizada em equipamentos de hardware, com o intuito de desenvolver um produto melhor que o do concorrente no mercado. Ela era feita como um clone do produto original; já a engenharia reversa para software possui o intuito de melhorar um sistema no sentido de conseguir um nível mais alto de abstração do software.

O objetivo da engenharia reversa é possibilitar um conhecimento geral do sistema através da documentação gerada, facilitando assim manutenções futuras. Se o software

examinado já possuísse uma boa documentação, que é feita com uma boa engenharia de software, o trabalho dos analistas seria muito mais simples no que diz respeito à manutenção do mesmo.

Para que a engenharia reversa seja executada é necessário o uso de ferramentas que possibilitem o processo. Existem as ferramentas estáticas, que examinam o código fonte do programa, e as ferramentas dinâmicas, que examinam o programa em execução.

A metodologia adotada neste trabalho é a FaPRE/OO, Família de Padrões de Reengenharia Orientada a Objetos (Recchia et al. 2002a), a qual se baseia no modelo Fusion/RE e na linguagem de padrões proposta por Demeyer.

A metodologia proposta por Demeyer et al. (2002a) envolve diversos padrões agrupados em 4 clusters, de acordo com a familiaridade que cada padrão possui. Os clusters têm a seguinte denominação:

1. Iniciação ao Sistema Legado: agrupa os padrões que dizem o que se deve fazer no primeiro contato com o sistema. Esse cluster possui três padrões:

   a. Ler todo o código em uma hora;

   b. Estudar superficialmente a documentação;

   c. Entrevistar o usuário durante o sistema em operação;

2. Iniciação ao Sistema Legado: agrupa os padrões que dizem o que se deve fazer no primeiro contato com o sistema. Esse cluster possui três padrões:

   a. Ler todo o código em uma hora;

   b. Estudar superficialmente a documentação;

   c. Entrevistar o usuário durante o sistema em operação;

3. Iniciação ao Sistema Legado: agrupa os padrões que dizem o que se deve fazer no primeiro contato com o sistema. Esse cluster possui três padrões:

a. Ler todo o código em uma hora;

b. Estudar superficialmente a documentação;

c. Entrevistar o usuário durante o sistema em operação;

4. Entendimento inicial: agrupa padrões que dizem como obter um entendimento inicial do sistema. Esse cluster possui 4 padrões:

a. Presumir prováveis objetos;

b. Examinar a base de dados;

c. Inspecionar as maiores construções;

d. Explorar possíveis modificações;

5. Detalhamento do sistema: agrupa padrões que dizem como obter o entendimento detalhado dos componentes do sistema. Esse cluster possui 2 padrões:

a. Verificar as invocações de métodos;

b. Observar a execução dos componentes.

6. Preparação da reengenharia: este último cluster agrupa padrões que mostram como preparar os passos subseqüentes da reengenharia. Possui apenas um padrão:

a. Refazer para entender.

Todos os padrões apresentados nesta linguagem possuem o seguinte formato: **nome, intuito, problema, solução, avaliação (*trade-offs*), exemplo, justificativa, usos conhecidos e padrões relacionados.**

A figura 2.2 ilustra graficamente a linguagem proposta por Demeyer.

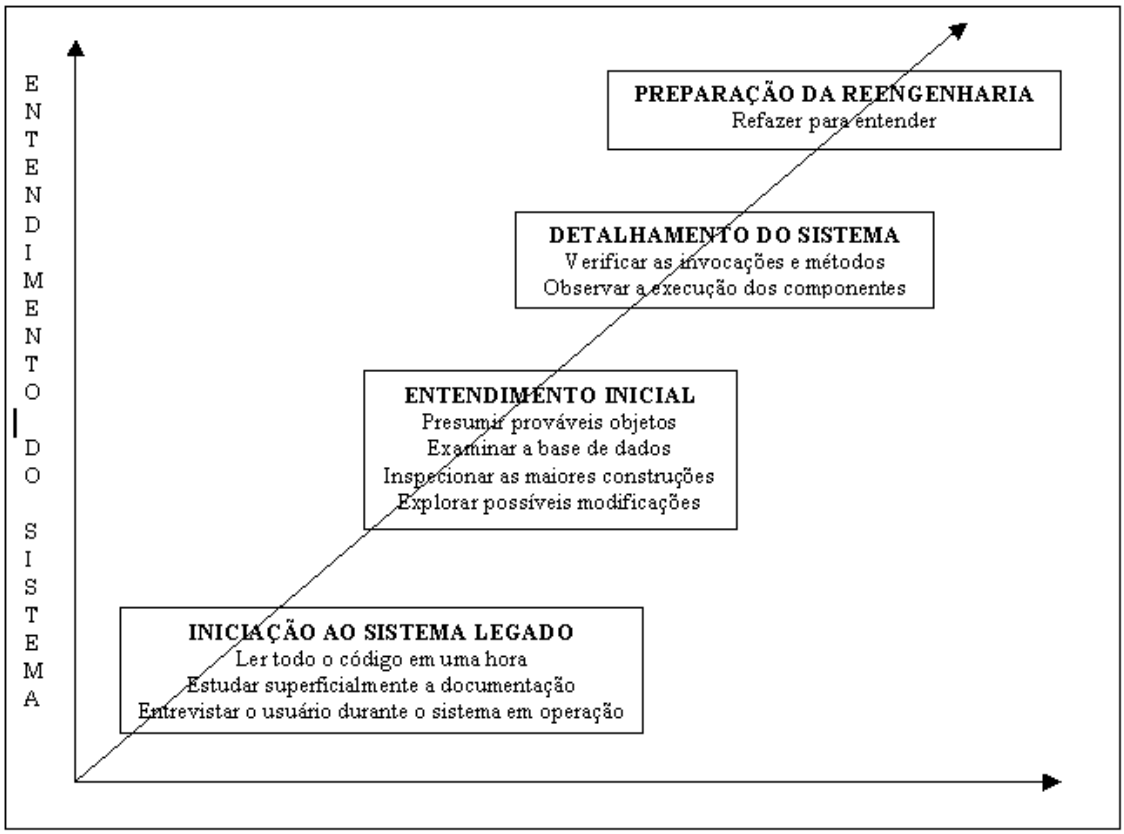


Figura 2.2 – Visão Geral da Linguagem proposta por Demeyer [Recchia, 2002]

Um dos problemas encontrados no modelo especificado acima é que ele é voltado para sistemas legados orientados a objetos, desenvolvidos em linguagens como Java, C++, Delphi, etc., e a maioria dos softwares legados manuseados são procedimentais, elaborados em linguagens tais como Cobol, Clipper, Fortran, etc.

Dessa forma, o método FaPRE/OO (Recchia et al. 2002a) (Recchia et al. 2002b) incorporou algumas modificações e melhoramentos no modelo acima, para que possa ser usado para a reengenharia não só de sistemas orientados a objeto, mas também em sistemas procedimentais. Um detalhamento maior sobre o FaPRE/OO será feito mais adiante.

A engenharia reversa é feita com mais facilidade quando a ferramenta muda suavemente entre as etapas de análise, projeto e desenvolvimento. No método Fusion, essas transições são feitas de forma suave, por isso o Fusion/RE é utilizado com tanta facilidade.

Uma definição formal para a engenharia reversa de software seria a de que esta consiste em analisar um programa, na tentativa de se criar uma representação do mesmo, em um nível de abstração mais alto que o código-fonte [Pressman, 1995].

2.2.1 CATEGORIAS DA ENGENHARIA REVERSA

Dependendo do entendimento do sistema a ser manipulado, duas categorias de engenharia reversa podem ser definidas:

A. Visualização de Código [Oman, 1990]: Esta é a forma mais antiga de se realizar engenharia reversa. A partir do exame do código, deve-se recuperar a documentação que já existiu, ou que deveria existir, sobre o sistema. A ênfase nessa fase é a criação de visões adicionais, principalmente visões gráficas, que já deviam existir desde a engenharia progressiva.

As visões mais profundas como, propósitos, função do sistema, são conhecidas na fase de entendimento do programa.

B. Entendimento de programa [Chikofsky & Cross, 1990]: Como dito antes, nessa fase, o objetivo é conseguir informações de abstração mais alta, se comparado com as informações da etapa anterior. Aqui extrai-se informações que vão além do código, tais como informações externas, de usuários e algumas deduções sobre o mesmo. Em resumo, o entendimento do programa deve fornecer informações sobre *o que*, *como e por que* o sistema faz.

O entendimento de programa diferencia-se da visualização de código, porque, como o próprio nome diz, ele busca *entender* o programa, e não apenas visões alternativas para auxílio ao entendimento. Esse entendimento vai além do conhecimento em *nível implementacional e estrutural*, obtendo o conhecimento em *nível-funcional* e até mesmo em *nível-de-domínio* (ambiente de operação do sistema).

A figura 2.3 mostra a diferença e a amplitude entre as duas categorias da engenharia reversa.

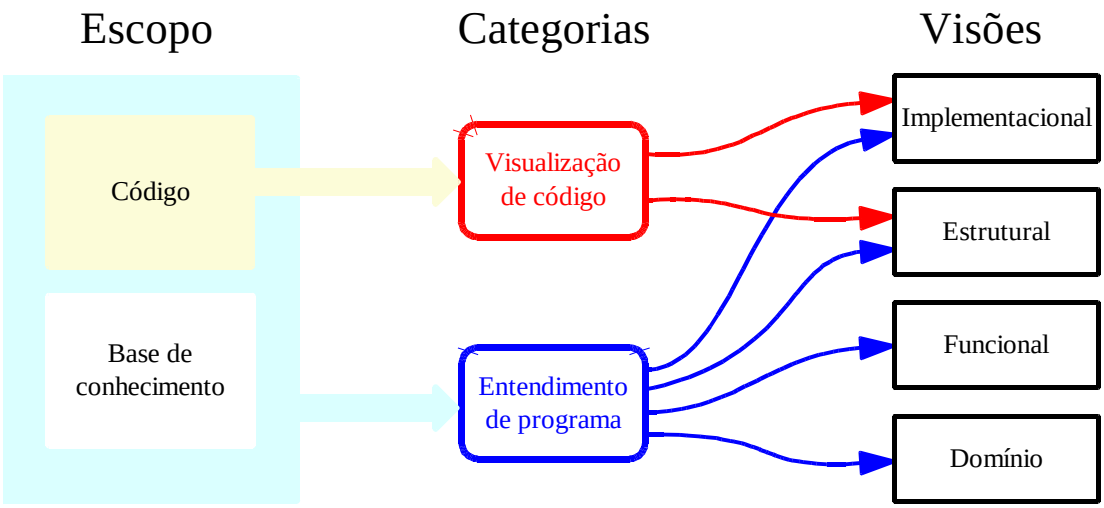


Figura 2.3 – Categorias da Engenharia Reversa [Quinaia, 1998]

Jacobson e Lindström em [Jacobson & Lindström 1991], definem a reengenharia de software como:

Reengenharia = Engenharia Reversa + Δ + Engenharia Progressiva

Onde Δ pode ser:

1. Alterações de funcionalidade: Ocorrem alterações no sistema devido a necessidade do usuário, ou algumas mudanças no negócio.

2. Alterações de implementação: Ocorrem alterações no ambiente de operação sistema ou alterações da linguagem de implementação (protocolos, sistema operacional, etc);

A explicação da equação acima, também dada por Jacobson e Lindström, é dada a seguir:

Engenharia Reversa: Nesse processo faz-se uma identificação dos relacionamentos entre os componentes do sistema. Um exemplo pode ser a dependência entre os arquivos e funções. O resultado poderá ser um diagrama de fluxo de dados para as funções, e um modelo de entidade e relacionamento para os arquivos. Com esse primeiro passo obtém-se um modelo abstrato das funcionalidades do sistema original.

Decidir sobre alterações na funcionalidade do sistema: Nesse passo as alterações de requisitos necessitadas pelo usuário são adicionadas ao sistema. Essas alterações são conseguidas com maior facilidade devido à fase anterior, visto já haver informações em um nível de abstração mais alto. Se não houvesse o processo de engenharia reversa, seria necessário modificar a funcionalidade do sistema, através de um nível mais baixo na abstração.

Reprojetar o sistema: Nesse passo, utiliza-se o resultado obtido nos passos anteriores para concretizar a engenharia progressiva, último passo da reengenharia. A partir dos modelos de abstrações o sistema é reimplementado, com suas novas funções. Aqui deve ser levado em consideração que se apenas parte do software for reimplementado, esta deve ser compatível com a parte antiga, de modo a não gerar conflitos no resultado final do produto.

O estudo para a reengenharia de software começa identificando-se os possíveis objetos nos programas procedimentais. Gall e Klosch [1995] falam que o processo de identificação de objetos se baseia em informações extraídas do código fonte, do conhecimento da aplicação e do domínio do sistema. Essa identificação de objetos inicia-se com a geração de diagramas estruturados, diagramas de fluxo de dados, etc.. E estes documentos irão constituir a base para a identificação de objetos do sistema.

2.2.2 ENGENHARIA AVANTE

A segunda parte da reengenharia de software é a engenharia avante, onde, de posse da documentação obtida no processo de engenharia reversa, é possível reformular o sistema legado, juntamente com as melhorias necessárias ao bom funcionamento do sistema.

O processo de engenharia avante, segundo Jacobson et al. (1991), é o passo do desenvolvimento normal de sistemas, isto é, a implementação do software em alguma linguagem de programação.

Para Chikofsky et al.(1990) a engenharia avante é o processo tradicional de transferir abstrações, modelo lógico e projeto independente da implantação para uma implementação física.

2.3 PLANEJAMENTO PARA REENGENHARIA

Para que a reengenharia de software seja um trabalho com bons resultados, é necessário estabelecer passos que busquem tal objetivo.

Sneed [1995], apresenta uma idéia para o planejamento da reengenharia em cinco passos:

1. *Justificativa do projeto*: Aqui deve ser feito um minucioso estudo sobre os sistemas da organização e a importância do sistema em questão, com o objetivo de justificar a reengenharia do mesmo. Nesta justificativa deve ser enfatizada a importância da reengenharia para o aumento da qualidade do software, a facilidade de futuras manutenções e o conseqüente aumento do valor do software para a organização.

2. *Análise do portifólio*: Deve-se estabelecer critérios para a escolha das aplicações que serão submetidas à reengenharia, de acordo com a importância destas para a organização. Essa análise pode ser feita com a ajuda da figura 2.3 para a determinação das aplicações de software candidatas a reengenharia.

3. *Estimativa de custo*: Deve ser feito ainda uma estimativa de custo do projeto total da reengenharia, através de planilhas e relatórios. Estimar a avaliação-ponderação dos componentes do software a serem reconstruídos. Estes componentes podem incluir: reengenharia do software, teste e reengenharia de dados.

4. *Análise do custo-benefício*: Com base na estimativa de custo feita no passo anterior, pode-se fazer uma análise do custo-benefício, para decidir se os custos da reengenharia irão valer a pena quando comparados ao resultado esperado na manutenção, ou se será mais viável comprar um novo software ou o seu próprio redesenvolvimento.

5. *Contrato externo*: Dois tipos de contratos podem ser usados em processos de reengenharia: o contrato por tempo e material e o contrato por resultados. Os contratos são feitos com o objetivo de conseguir um nível máximo de distribuição de projetos para evitar gargalos.

A figura 2.4 mostra a matriz de decisão sobre a estimativa de custo para análise da viabilidade do processo. Deve-se decidir entre substituir, manter, melhorar ou efetuar a reengenharia de um sistema de software legado.

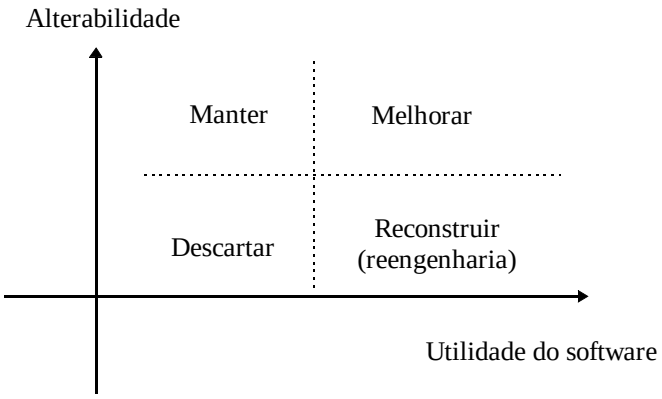


Figura 2.4 – Matriz de Decisão sobre o destino do Software Legado
[Jacobson & Lindström, 1991]

A partir de uma análise da figura pode-se ver que sistemas fáceis de alterar, mas com pouca utilização na empresa, devem ser mantidos sobre manutenção. Sistemas difíceis de alterar e com pouca utilização devem ser descartados. Por outro lado, sistemas difíceis de alterar mas com grande utilidade para a empresa, devem ser reconstruídos, e sistemas de grande utilização e fáceis de alterar, esses é que devem continuar evoluindo.

2.4 O MÉTODO FUSION/RE

Um novo método de se fazer engenharia reversa a partir de sistemas legados é mostrado em *Penteado et al. (1996, 1996a).* Suas principais características são extensibilidade de integrar heurísticas novas e ferramentas para engenharia reversa de dados e ainda algumas métricas que avaliam o esforço da conversão de sistemas procedimentais em orientados a objetos. O Fusion/RE consiste em 4 passos:

1. Revitalização da arquitetura do sistema atual;
2. Criar um modelo de análise do sistema atual (MASA);
3. A partir do MASA criar um modelo de análise (MAS);
4. Fazer o mapeamento do MAS para MASA.

Os passos descritos acima são ilustrados na figura 2.4.

Mais tarde (1998) foram apresentados mais dois passos que foram incorporados ao método Fusion/RE que dizem respeito aos modelos de projeto e segmentação. A base para esses dois passos provem dos modelos obtidos quando da realização da engenharia reversa.

A necessidade de ferramentas para suporte a reengenharia de software é claramente difundida. Sneed (1996) descreve um processo com o apoio de uma ferramenta para reengenharia de sistemas em Cobol. Cremer (1998), Fioravante et al. (1998) e Gall et al. (1993) também apresentam exemplos para suportes. A ferramenta usada nesse projeto na fase de engenharia avante do sistema, será a Rational Rose (Rational Software Corporation, 2000).

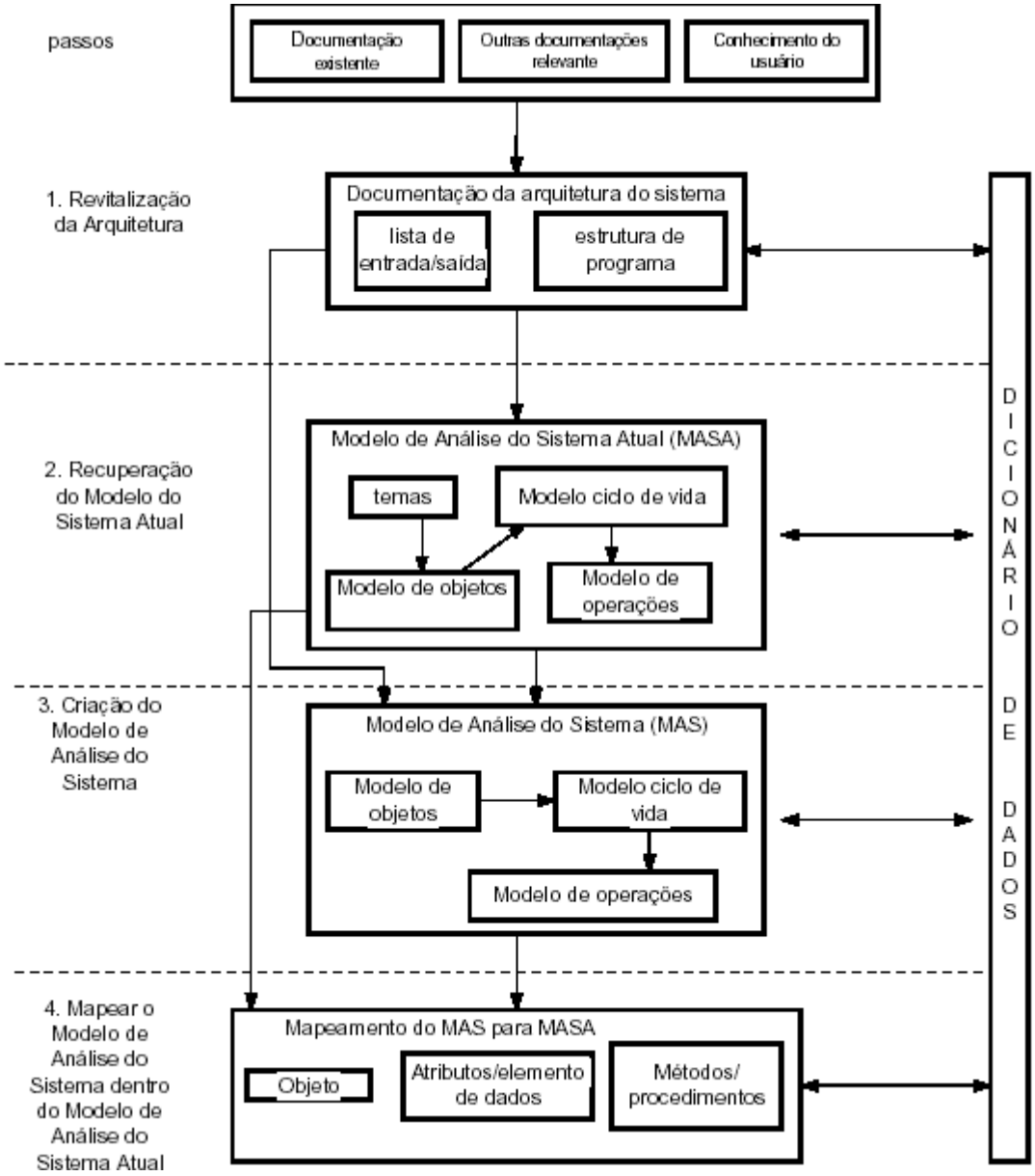


Figura 2.5 – Passos da Abordagem Fusion/RE [Penteado]

2.5 A FAMILIA DE PADRÕES DE REENGENHARIA ORIENTADA A OBJETOS – FAPRE/OO

Recchia et al. (2002), baseado na Linguagem de Padrões de Engenharia Reversa proposta por Demeyer (et al 2002a), e também no método Fusion/RE (Penteado et al. 1995), (Penteado 1996), (Penteado et al. 1996a), (Penteado et al. 1998), propõe uma nova linguagem

de padrões para a reengenharia de software para sistemas legados procedimentais, visto que ficou provado em sua dissertação de mestrado que a linguagem de padrões de Demeyer não se aplica a este tipo de software, e sim apenas a sistemas legados orientados a objetos.

A FaPRE/OO é constituída no seu todo de uma família de 17 padrões, sendo 13 padrões divididos em 3 clusters na etapa de engenharia reversa, e 4 padrões agrupados em 1 cluster na etapa de engenharia avante. Esses clusters e seus padrões são explicados mais detalhadamente a seguir:

Cluster 1 – MODELAR OS DADOS DO LEGADO: Este cluster agrupa 4 padrões que extraem informações a partir dos dados e do código fonte do sistema legado, gerando o MER (Modelo de Entidade e Relacionamento) e o MASA (Modelo de Análise do Sistema Atual) – Diagrama de Pseudo-Classes (visão orientada a objeto dos dados). Os padrões deste cluster auxiliam o engenheiro de software no seu primeiro contato com o sistema, são eles:

- Iniciar a análise dos dados;
- Definir chaves;
- Identificar relacionamentos;
- Criar visão OO dos dados;

Cluster 2 – MODELAR A FUNCIONALIDADE DO SISTEMA: Este cluster também agrupa 4 padrões que ajudam a obter a funcionalidade do sistema, através de modelos que possibilitam ao engenheiro de software o entendimento detalhado dos componentes do software, aprofundando assim seu conhecimento do mesmo. São padrões deste cluster:

Obter cenários;

- Construir diagramas de Use Cases;
- Elaborar a descrição de Use Cases;
- Tratar anomalias;

Cluster 3 – MODELAR O SISTEMA ORIENTADO A OBJETOS: Cinco padrões são agrupados neste cluster, onde estes fazem a obtenção do Diagrama de Classes e de Seqüência do sistema, através da interação com os modelos obtidos nos padrões dos clusters anteriores. Esse cluster também permite ao engenheiro obter o MAS – Modelo de Análise do Sistema, sendo como um produto final para ser efetuada a transição da engenharia reversa para a engenharia avante. Os padrões deste último cluster do módulo engenharia reversa são:

- Definir as classes;
- Definir atributos;
- Analisar hierarquias;
- Definir métodos;
- Construir diagramas de sequência;

No decorrer deste trabalho será observado que muitos padrões serão entrada ou saída para outros padrões, dando assim a idéia de ser um método sequencial, mas como ilustrado na figura 2.5, apenas os padrões do Cluster 1 são feitos de modo sequencial. Este é um método evolutivo, podendo-se avançar ou recuar em qualquer padrão.

Os padrões da FaPRE/OO, seguem o mesmo formato dos padrões propostos por Demeyer et al. (2000a), são eles: Nome, Intuito, Problema, Contexto (Influências), Solução, Avaliação, Justificativa, Usos Conhecidos, Padrões Relacionados e Produtos Obtidos. O item **Solução**, sempre que necessário deve ser apresentado na forma de passos, os itens de **Avaliação** e **Justificativa** normalmente contém a opinião dos engenheiros quando à utilização destes padrões no processo de Engenharia Reversa. E no item **Usos Conhecidos** é apresentado o uso de padrões similares, documentados ou não sob a forma de padrão.

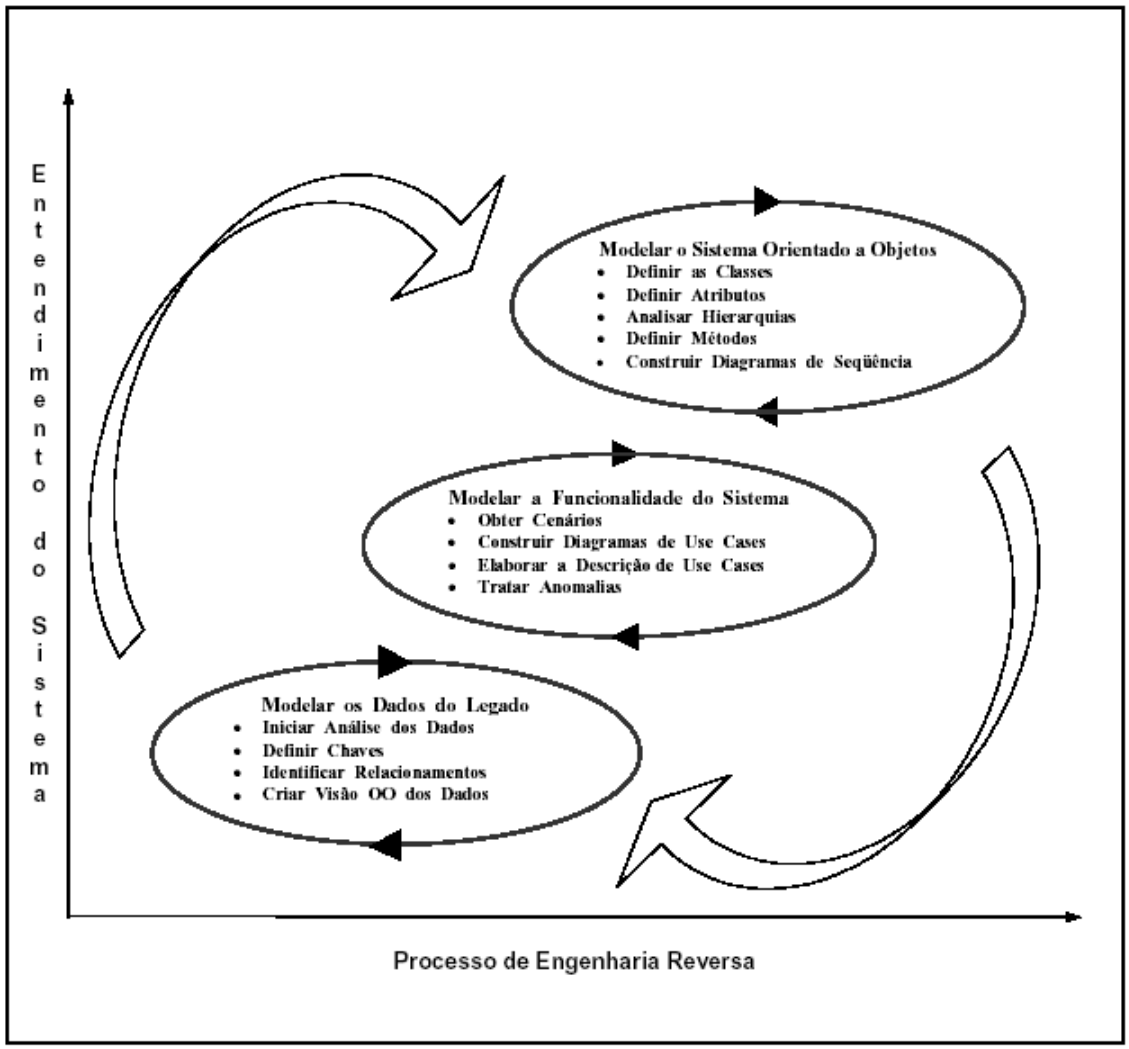


Figura 2.6 – Padrões de Engenharia Reversa da Família de Padrões de Reengenharia Orientado a Objetos [Recchia, 2002]

O quarto cluster já é o que trata os padrões de engenharia avante. A denominação deste cluster é *Gerar o Sistema Orientado a Objetos*. O formato dos padrões para engenharia avante da Família de Padrões de Reengenharia Orientado a Objetos são implementados da seguinte maneira: Nome, Intuito, Aplicabilidade, Contexto (Influências), Solução, Avaliação, Justificativa, e Produtos Obtidos. Os itens **Avaliação** e **Justificativa** normalmente contém a opinião dos engenheiros quando à utilização destes padrões no processo de Engenharia Avante.

Cluster 4 – GERAR O SISTEMA ORIENTADO A OBJETOS: Este cluster agrupa 4 padrões que completam o processo de engenharia avante de um sistema, mudando o sistema legado, do paradigma procedimental para o paradigma orientado a objetos. Os padrões são apresentados a seguir:

- Definir a plataforma;
    - Escolher o ambiente operacional;
    - Escolher o Banco de Dados Relacional;
    - Escolher o pacote da linguagem de implementação;
- Converter o Banco de Dados;
- Implementar os métodos;
- Realizar melhorias na interface;

A ilustração dos padrões de engenharia avante juntamente com a etapa de engenharia reversa é mostrada a seguir:

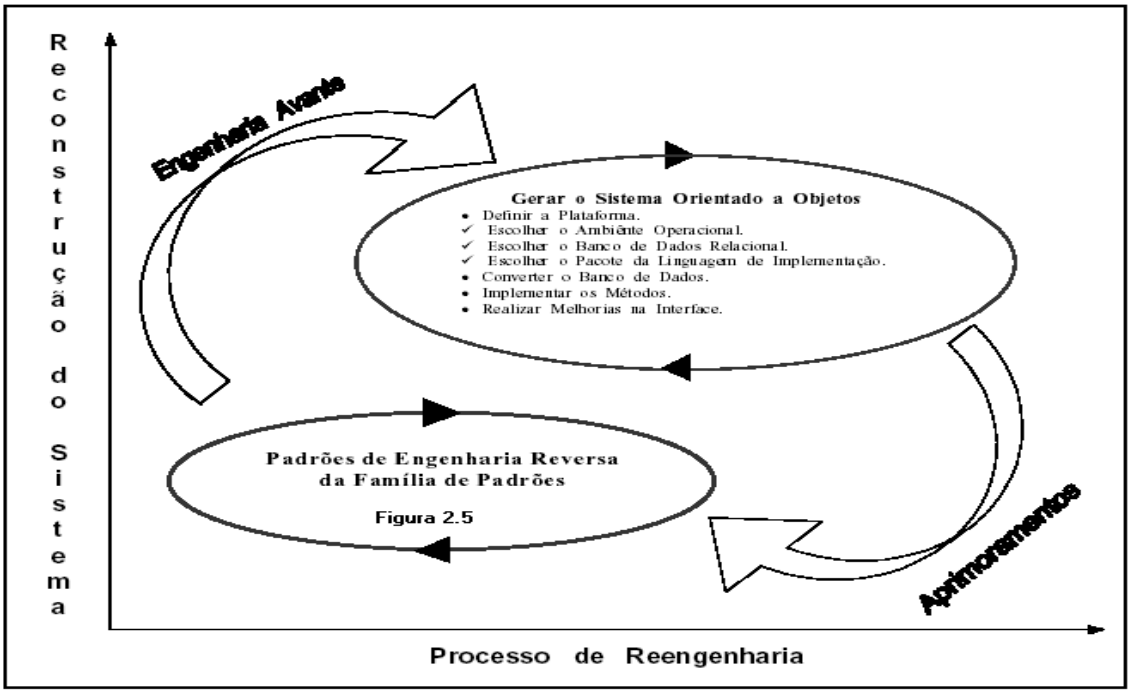


Figura 2.7 – Padrões da Engenharia Avante da Família de Padrões de Reengenharia Orientado a Objetos[Recchia, 2002]

É importante salientar que sem a posse do código fonte do sistema, será mais difícil para o analista de sistemas realizar a reengenharia a partir da FaPRE/OO. Apenas alguns de seus padrões poderiam ser utilizados, além do mais, o analista teria que utilizá-lo em conjunto com uma outra metodologia (por exemplo o método Fusion/RE-I, [Penteado]). Para que todos os padrões aqui utilizados tenham seus efeitos obtidos, é necessário que o analista possua o código fonte e o aplicativo executável do sistema.

2.6 CONCLUSÕES

Neste capítulo foram apresentadas várias definições da literatura sobre Reengenharia de Software, suas divisões, categorias e objetivos. Os temas Engenharia Reversa, Método Fusion/RE e ainda a FaPRE/OO, Família de Padrões de Reengenharia Orientada a Objetos também são abordados de forma a ser dada uma visão inicial do que será usado no desenvolvimento deste trabalho.

3 REENGENHARIA DO SISTEMA DE UMA IMÓBILIÁRIA BASEADA NOS PADRÕES DA FAMÍLIA DE PADRÕES DE REENGENHARIA ORIENTADA A OBJETOS – FAPRE/OO

3.1 INTRODUÇÃO

Nesse capítulo, serão apresentadas as etapas do processo de engenharia reversa efetuado no sistema de controle de locação de uma imobiliária. Para a efetivação desse processo foram utilizados os padrões da Família de Padrões de Reengenharia Orientado a Objetos FaPRE/OO (Recchia et al. 2001), (Recchia et al. 2002a), (Recchia et al. 2002b).

3.2 O SISTEMA

Há alguns anos atrás (meados de 1990, ou antes) foi desenvolvido por projetistas da cidade de Barbacena, um sistema para controle de locações de uma imobiliária, sendo utilizado nesta mesma cidade. O objetivo deste sistema, é gerenciar as locações dos imóveis cadastrados na organização, através de inclusões de locadores, imóveis, locatários, alterações destes itens, consultas a dados destes itens, aos imóveis que estão alugados, e os disponíveis para locação. O sistema ainda emite vários relatórios, como recibo de pagamento de aluguel, confecção de contratos de locação, relatório de locatários em débito, listagem de locadores, listagem de depósitos não realizados, comprovante de imposto de renda, cartas de cobrança para locatários em débito, lista de locatários, lista de contratos vencidos, e vários tipos de cartas modelo. O sistema possui ainda uma função que agenda os depósitos da parte do aluguel que cabe aos locadores, gerando todos os dias uma lista com todos os depósitos referentes àquele dia.

O sistema foi desenvolvido em Clipper (versão Summer 87), possui aproximadamente 8.000 mil linhas de código e comporta várias características de sistema legado. É um sistema razoavelmente antigo e não possui documentação alguma, visto que o projetista foi também o programador. Sendo assim ele constitui um ótimo sistema para ser submetido à reengenharia de software.

3.3 OS PADRÕES

Não falaremos aqui novamente sobre os padrões, visto já ter sido amplamente falado sobre eles no capítulo 2, seção 2.5. Portanto, as seções daqui em diante serão com o objetivo de apresentá-los efetivamente e aplicá-los no sistema de controle de locações de uma imobiliária.

3.4 A ENGENHARIA REVERSA DO SISTEMA DE UMA IMOBILIÁRIA UTILIZANDO OS PADRÕES DA FAMÍLIA DE PADRÕES DE REENGENHARIA PARA SISTEMAS LEGADOS EM CLIPPER

Nessa seção será elaborada a etapa de Engenharia Reversa do sistema com base no modelo de padrões FaPRE/OO. Como dito anteriormente essa etapa envolve 13 padrões agrupados em 3 clusters e, após esta etapa, o resultado será uma documentação apta para se fazer a etapa de Engenharia Avante do sistema. E finalmente se terá um novo sistema com as mesmas funcionalidade essenciais do antigo e ainda novas aplicações que poderão ser implantadas ao longo da Reengenharia.

Cada padrão da Engenharia reversa é composto de 10 itens que serão preenchidos conforme a verificação do sistema, são eles: **Nome, Intuito, Problema, Contexto (Influências), Solução, Avaliação, Justificativa, Usos Conhecidos, Padrões Relacionados e Produtos Obtidos.**

CLUSTER – **MODELAR OS DADOS DO LEGADO:**

1. Nome: **Iniciar a Análise dos Dados:**

Intuito: Iniciar a construção do MER, Modelo de Entidade e Relacionamento, utilizando uma tabela que relaciona todos os programas/módulos que fazem parte do sistema, com seus respectivos arquivos de dados.

Problema: Existe uma relação de programas/módulos e de arquivos de dados na Organização, porém não se sabe como eles se relacionam.

Influências:

- Ausência de documentação, exceto o código fonte (programas) e os arquivos de dados.

- Em sistemas complexos, há grande quantidade de arquivos de programas e de dados, dificultando o seu entendimento.

- O engenheiro de software tem conhecimento suficiente da linguagem de implementação para encontrar arquivos de dados no código fonte.

Solução:

1. Construir uma tabela, denominada Programas X Arquivos, com duas colunas. A primeira coluna contém os nomes dos programas e, a segunda, o nome dos arquivos de dados a que esses programas têm acesso. O nome original desses arquivos deve ser mantido.

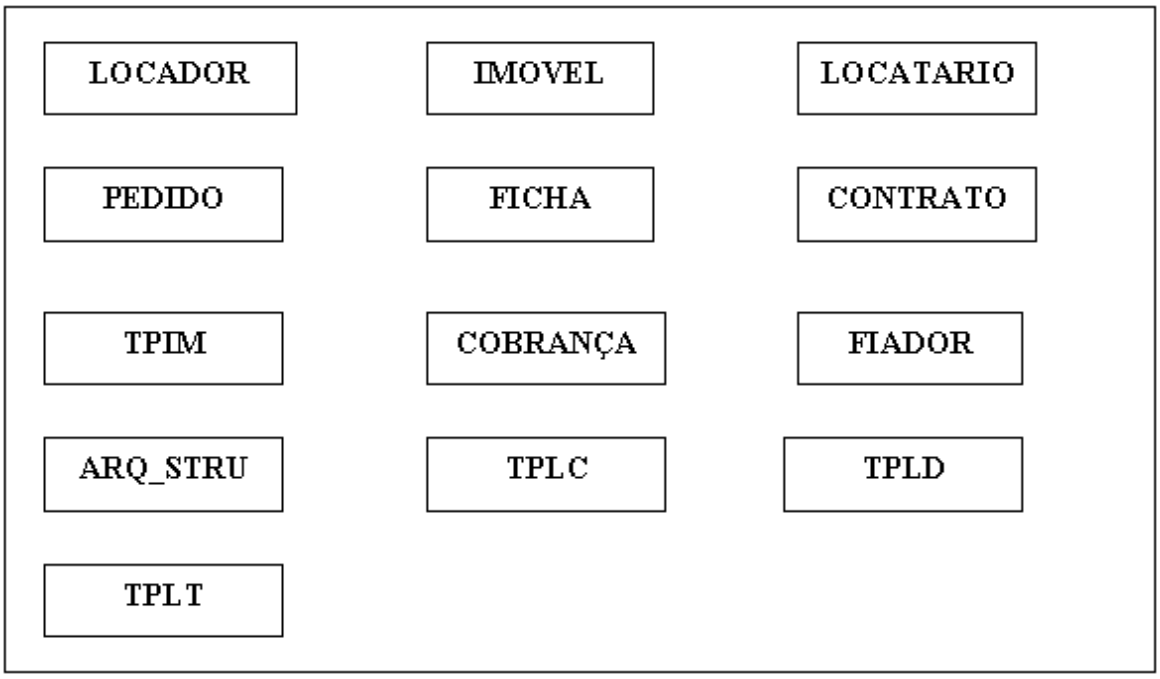
A tabela de Programas X Arquivos está explicitada completamente na figura 1 do Apêndice 1. A figura 3.1 está exemplificando parte desta tabela.

Tabela 1 – Tabela de Programas X Arquivos do Sistema Legado:

Programas (.prg)	**Arquivos (.dbf)**
Fimo.prg	
Imo.prg	
Imo1000.prg	
Imo1001.prg	Locador.dbf
Imo1002.prg	Locador.dbf
Imo1003.prg	Locador.dbf Imovel.dbf Locatario.dbf Fiador.dbf
Imo1004.prg	Imovel.dbf Tpim.dbf
•••	•••

2. Iniciar a construção do MER a partir da tabela construída no passo 1 (fig. 1 do Apêndice 1). Considere cada arquivo de dados do sistema legado (.dbf), como uma entidade do MER (figura 3.2).

Figura 3.1 – Modelo de Entidade e Relacionamento Inicial:



Avaliação:

A construção da tabela ajuda no primeiro contato com o sistema, pois é o primeiro passo para a construção do MER.

Justificativa:

Esse padrão foi utilizado para obter um primeiro contato com o sistema, e seu resultado será usado em padrões que são apresentados mais adiante.

Padrões Relacionados: Esse padrão é a entrada para o padrão Identificar Relacionamentos.

Produtos Obtidos:

- Tabela Programas X Arquivos;
- MER – Modelo de Entidade Relacionamento Inicial.

2. Nome: **Definir Chaves:**

Intuito: Obter as chaves primárias e estrangeiras para cada entidade identificada pelo padrão Iniciar Análise dos Dados, a partir de arquivos de dados do sistema legado (.dbf), a fim de identificar os relacionamentos entre as entidades que compõem o MER, iniciado no passo anterior.

Problema: Em sistemas procedimentais não existe o conceito de chaves primárias e estrangeiras. Nesses sistemas as chaves podem estar em arquivos separados ou em cláusulas especiais dentro do próprio arquivo de dados.

Influências:

- Sistemas implementados em linguagens como Clipper, Cobol, etc., utilizam arquivos para persistir os dados.
- Para obter a chave primária, a partir dos arquivos de dados, é necessário analisar arquivos de índices (por exemplo, Clipper) ou o código fonte (por exemplo, Cobol).
- O engenheiro de software tem à sua disposição utilitários que acompanham a linguagem de implementação, os quais viabilizam a análise dos arquivos de dados e de índices. conhecimento suficiente da linguagem de implementação para encontrar arquivos de dados no código fonte.
- Tem-se todos os índices de acesso aos dados do sistema, nos quais estão identificadas as chaves de acesso aos dados.

Solução:

1 . Construir uma tabela, denominada Entidades X Chaves, com duas colunas. A primeira coluna contém os nomes das entidades e, a segunda, as chaves primárias identificadas para essas entidades.

Por meio do comando USE, da linguagem Clipper, obter todos os arquivos de índices (.ntx) para cada arquivos de dados (.dbf). Um arquivo .dbf pode ter até 15 arquivos de índices.

Identifica-se a Chave Primária para cada entidade analisando o respectivo arquivo de dados .dbf com a ferramenta DBU, que acompanha a linguagem Clipper, observando-se a linha que contém a palavra Key.

A figura 2 do Apêndice 1 ilustra todos os arquivos de dados (.dbf) com seus respectivos arquivos de índices (.ntx), que foram usados para a construção da tabela Entidades Chaves.

A construção final da tabela de Entidades X Chaves está explicitada completamente na figura 3 do Apêndice 1.

2 . Utilizar o código fonte e a ferramenta DBU para identificar as chaves estrangeiras das entidades, adicionando-as na Tabela Entidades X Chaves em uma nova coluna (Chave Estrangeira). Para o sistema em análise, não foram encontradas chaves estrangeiras propriamente ditas. O desenvolvedor do sistema, quando buscava chaves primárias de outros arquivos, as concatenava na chave primária do arquivo em uso, e tudo se agrupava em uma chave só. Por exemplo, o arquivo Locatario possui a chave primária Loc_Imo; esta chave seria a junção das chaves de Locador (Codigo) e Imovel (Codigo), mas na verdade, a chave primária de Locatario é uma concatenação da chave Código, de Locador, e da chave Código, de Imovel; e essa junção é armazenada na variável Loc_Imo.

Para a estrutura Imovel, o caso é o mesmo, sua chave (Codigo) seria a chave de Locador (Codigo) e mais um campo de autonumeração, que identifica quantos imóveis determinado locador possui. Mas na verdade esta chave é feita também com a concatenação dos dois campos em apenas uma variável – Codigo.

A tabela 2 mostra a tabela Entidades X Chaves completa, com a 3.ª coluna, chamada de Chave Estrangeira.

Tabela 2 – Tabela de Entidades X Chaves Completa:

Entidades	Chaves Primárias	Chaves Estrangeiras
Arq_Stru	———	———
Cobrança	———	———
Contrato	———	———
Fiador	Loc_Imo(chave de locatário)	Loc_Imo
Ficha	———	———
Imóvel	Código(locador)+Código	Código(locador)
Locador	Código	———
Locatári	Loc_Imo (chave de imóvel)	Loc_Imo
Pedido	———	———
Tpim	Código(locador)+Código	Código(locador)
Tplc	Codigo(locador)	Código(locador)
Tpld	Codigo(locador)	Código(locador)
Tplt	Dia	———

Avaliação:

A partir da análise inicial dos dados, pudemos extrair um MER inicial, onde foram identificadas as entidades do sistema. Com a identificação destas entidades, podemos então localizar as chaves primárias e estrangeiras que o sistema possui.

Justificativa:

Esse padrão nos possibilita obter os atributos das entidades e suas chaves, e é utilizado em padrões posteriores.

Padrões Relacionados: Esse padrão é a entrada para o padrão Identificar Relacionamentos.

Produtos Obtidos:

- Tabela Entidades X chaves;

3. Nome: **Identificar Relacionamentos:**

Intuito: Identificar os relacionamentos entre as entidades do MER.

Problema: Não se sabe como os arquivos de dados do sistema legado estão relacionados entre si.

Influências:

- Sistemas implementados em linguagens como Clipper, Cobol, etc., em geral não utilizam bancos de dados relacional, no qual os relacionamentos entre as tabelas são explícitos.
- A partir da inspeção dos dados e da análise do código fonte é possível reconstruir o modelo de dados do sistema legado.

Solução:

1. Utilizando-se a Tabela Entidades X Chaves e trechos do código fonte que auxiliaram na elaboração desta tabela, pode-se identificar os relacionamentos entre entidades. Sempre que o campo de um arquivo de dados, que representa a chave primária de alguma entidade, for atribuído a um campo de um outro arquivo de dados, que representa a chave estrangeira de outra entidade (ou dela mesmo, no caso de auto-relacionamento), há um relacionamento entre essas entidades, sendo este representado no MER.
2. Para explicitar a cardinalidade deve-se observar, nos trechos do código fonte, se há condição restritiva quanto à gravação de informação. Caso haja, a própria condição informa a cardinalidade mínima e máxima. Caso não haja, a cardinalidade mínima é “Zero” e a máxima é “N”.

O sistema em questão, como foi dito anteriormente, não possui chaves estrangeiras, mas através de trechos do código fonte podemos identificar que as entidades se relacionam de alguma forma. Assim será possível construir o MER necessário para as próximas etapas.

Avaliação:

A partir dos produtos já obtidos podemos identificar os relacionamentos entre as entidades do sistema e dessa forma compor mais um elemento do trabalho: o MER.

Justificativa:

A partir dos padrões utilizados anteriormente, podemos usar seus resultados como entrada para o padrão especificado.

Padrões Relacionados:

Esse padrão tem como entrada os seguintes padrões: Iniciar a Análise dos Dados e Definir Chaves.

Produtos Obtidos:

- MER – Modelo de Entidade e Relacionamento do Sistema Legado.

Para as entidades ARQ_STRU, TPLC, TPLD, TPIM e TPLT não foram encontrados relacionamentos, visto que na análise do código fonte do programa não foi efetivamente encontrado algum uso para o arquivo ARQ_STRU e os outros 4 arquivos são cópias geradas dos arquivos Locador, Locador, Imovel e Locatario, respectivamente, com apenas algumas variáveis dos arquivos originais, provavelmente para gerar relatórios.

A figura 3.2 ilustra os relacionamentos das entidades, formulando o MER do sistema legado.

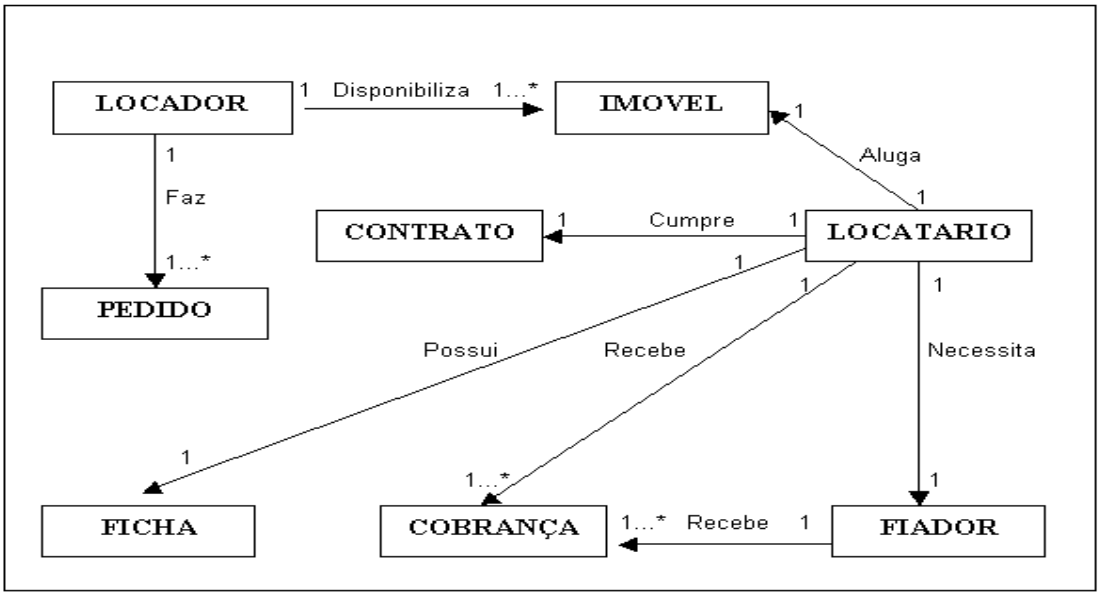


Figura 3.2 – MER do Sistema Legado

4. Nome: **Criar Visão OO dos Dados:**

Intuito: Criar uma visão orientada a objetos dos dados.

Problema: Transformar um modelo de dados desenvolvido no paradigma procedimental em um modelo de dados no paradigma orientado a objetos.

Influências:

- Reconhecer classes/objetos e seus relacionamentos a partir do código procedimental é difícil..
- Um engenheiro de software tem conhecimento dos conceitos da UML, Linguagem Unificada de Modelagem, para gerar um modelo orientado a objetos, a partir do MER.

Solução:

A partir do MER, construir o Diagrama de Pseudo-Classes (candidatas a classes) do sistema legado, gerando assim o MASA, Modelo de Análise do Sistema Atual. O nome original das entidades do MER deve ser mantido

1. Considerar cada entidade do MER como uma pseudo-classe.

2. Buscar pares de relacionamento n-para-n no MER, que podem ser representados como link de atributo no modelo orientado a objetos, de acordo com a funcionalidade do sistema legado e com os conceitos da orientação a objetos.

3. Buscar nos relacionamentos um-para-n no MER, aqueles que podem representar o princípio Todo-Parte (agregação por: Referência ou Valor), considerando a funcionalidade do sistema legado e os conceitos da orientação a objetos.

4. Casos de especialização serão tratados no padrão Analisar Hierarquias.

A figura 3.5 ilustra o MASA para o MER do sistema legado.

Avaliação:

Esse padrão nos fornece o produto final deste Cluster: o MASA. Através dele, poderemos seguir a etapa de engenharia reversa do sistema em busca de torná-lo um sistema orientado a objetos.

Justificativa:

O resultado obtido neste padrão será usado mais adiante em outro padrão e poderá assim continuar a análise do sistema.

Padrões Relacionados:

Esse padrão é a entrada para o padrão Tratar Anomalias.

Produtos Obtidos:

MASA – Modelo de Análise do Sistema Atual: Diagrama de Pseudo-Classes.

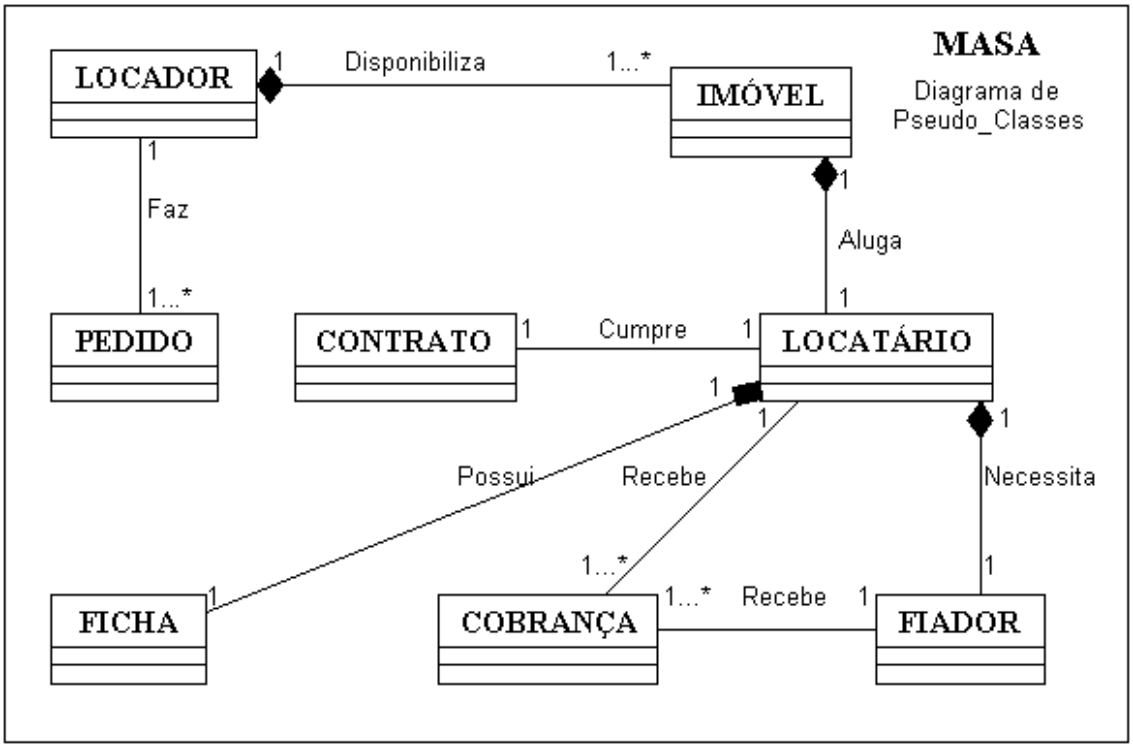


Figura 3.3 – MASA do Sistema Legado

CLUSTER – **MODELAR A FUNCIONALIDADE DO SISTEMA:**

5. Nome: **Obter Cenários:**

Intuito: Obter os cenários do sistema através da análise das interfaces do sistema em operação.

Problema: Há grande variedade de interface nos sistemas legados. No entanto, é necessário identificar os Cenários do Sistema para obter um modelo que represente a sua funcionalidade.

Influências:

- Sistemas implementados em linguagens como Clipper, Cobol, etc., utilizam telas de menus para acesso à funcionalidade.
- Não existe uma forma padronizada para a construção desses menus.

- Cada cenário, em sistemas legados como Clipper, Cobol, etc., é representado por um Menu e cada opção do menu corresponde a um ou mais programas fonte.

- O engenheiro de software tem experiência para encontrar cenários em interfaces poluídas.

Solução:

1. Construir a tabela de Cenários do sistema com duas colunas. A primeira coluna contém as opções do menu principal, e a segunda, os submenus de cada opção do menu principal. Para essa construção, obter os cenários do sistema através da observação do sistema legado em operação.

A figura 3.4 exibe a interface inicial do sistema de controle de locações da imobiliária, estando em prompt do DOS, posicionado em menu INCLUSÕES > opção 1 – LOCADORES. A figura 3.5 mostra os cenários obtidos dessa interface.

Avaliação:

O objetivo deste padrão é obter todas as interfaces do sistema através de seus menus, que são os cenários. Aqui capturamos também os submenus e suas ramificações, se existirem.

Justificativa:

Esse padrão é utilizado para a construção dos diagramas de *Use-Case* do sistema.

Padrões Relacionados:

Esse padrão deve ser aplicado em conjunto com o padrão Entrevistar o Usuário durante o Sistema em Operação (*Interview During Demo*) proposto por Demeyer et. al (2000a) e é a entrada para os padrões Construir Diagramas de Use Case e Elaborar a Descrição de Use Case.

Produtos Obtidos:

- Tabela Cenários do Sistema;

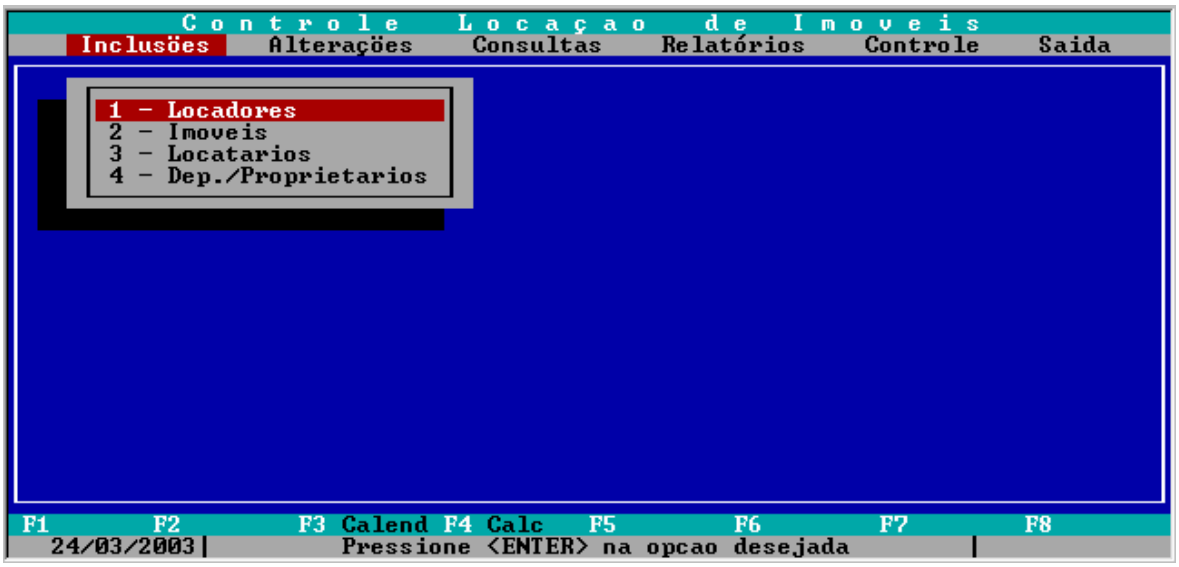


Figura 3.4 – Menu inicial do Sistema de Controle de Locações da Imobiliária

CENÁRIOS Opções do Menu Principal	CONTEÚDO Opções dos sub-menus de cada opção principal
Inclusões	Locadores Imóveis Locatários Dep./Proprietários
Alterações	Locadores Imóveis Locatários Ficha
Consultas	Locadores Imóveis Alugados Imóveis Disponíveis Locatários Agenda Depósitos Aniversarian. do Mês
Relatórios	Recibo de Pgto de Aluguel Contrato de Locação *(Contrato Imovel (1), Contrato Telefone, Comunicado Imóvel, Comunicado Telefone, Contrato Imóvel (2))* Locatários em Débito Demonstrativo Depósito (*Individualmente, Global*) Listagem dos locadores Depósitos Ñ. Realizados Comprovante Imposto de Renda Carta Cobrança Locatário (*Carta de Número 1, Carta de Número 2, Carta de Número 3*) Listagem Locatários Contratos Vencidos Carta Pedido de Imóvel Carta de Distrato Carta Cobrança Fiador Emissão Etiqueta (*Proprietários, Inquilinos*)
Controle	Organizar Arquivos Tipo de Reajuste
Saída	Sair do Sistema

Figura 3.5 – Tabela Cenários do Sistema

6. Nome: **Construir Diagramas de Use Cases:**

Intuito: Construir todos os diagramas de Use Cases do Sistema a partir da Tabela Cenários do Sistema elaborada no padrão Obter Cenários.

Problema: Documentar as informações da funcionalidade de um sistema procedimental durante a engenharia reversa orientada a objetos.

Influências:

- Obter o Use Cases a partir do código procedimental é difícil.
- Os cenários obtidos a partir das opções de menu, viabilizam a construção dos respectivos diagramas de Use Cases.
- O engenheiro de software tem conhecimento da UML.

Solução:

Considerar como Use Case cada item da segunda coluna da Tabela Cenários do Sistema. Observe que cada item da coluna “Cenários” possui um Conteúdo com várias opções, como mostrado na figura 3.4. Cada opção do conteúdo de um Cenário passa a corresponder a um Use Case no Diagrama em construção.

A figura 3.6 exibe parte do Diagrama de Use Cases do Sistema para Controle de Locações de uma Imobiliária. Os nomes para cada Use Case deve ser definido pelo próprio engenheiro de software, bem como os eventos associados a cada Use Case, pela observação do sistema legado em operação, quando da ativação de cada opção do Menu correspondente. O Diagrama de Use Cases para todos os conteúdos dos seis Cenários é mostrado na figura 4 do Apêndice 1.

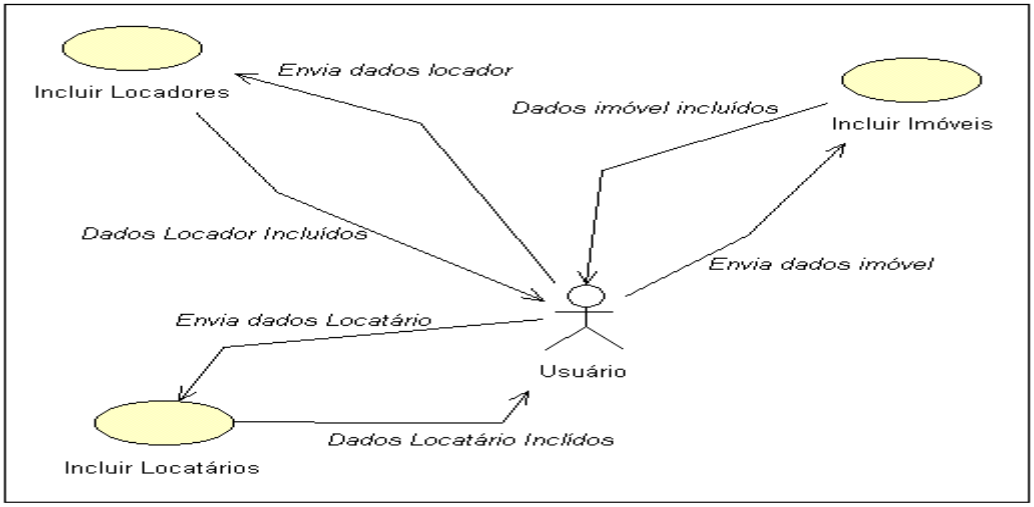


Figura 3.6 – Diagrama de Use Case para o Cenário Inclusões

Avaliação:

A partir deste padrão podemos elaborar uma documentação da funcionalidade do sistema.

Justificativa:

O produto deste padrão é utilizado para descrever as *Use-Cases*.

Padrões Relacionados:

Esse padrão deve ser aplicado em conjunto com o padrão Entrevistar o Usuário Durante o Sistema em Operação (Interview During Demo) proposto por Demeyer et. al (2000a) e é utilizado como entrada para o padrão Elaborar a Descrição de Use Cases.

Produtos Obtidos:

- Diagramas de Use Case do Sistema.

7. Nome: **Elaborar a descrição de Use Cases:**

Intuito: Elaborar a descrição correspondente a cada Use Case obtido pelo padrão Construir Diagramas de Use Cases.

Problema: É necessário registrar a lógica da funcionalidade do sistema para facilitar sua futura reengenharia.

Influências:

- Há falta de documentação do sistema que registre a lógica da funcionalidade.
- Tem-se o código fonte disponível para obter a lógica da funcionalidade.
- O engenheiro de software tem experiência de uso da linguagem de programação.

Solução:

Para cada Use Case obtido pelo padrão Construir Diagramas de Use Case, elaborar sua descrição a partir de trechos do código fonte correspondente à opção do Menu ativada, quando da construção do respectivo Use Case. A descrição correspondente ao Diagrama de Use Case Inclusões de Locadores é descrito na figura 3.7. Todas as outras descrições estão na figura 5 do Apêndice 1.

Avaliação:

Com o uso desse padrão poderemos entender de forma mais detalhada a funcionalidade do sistema.

Justificativa:

Sem o uso desse padrão seria mais difícil dar continuidade ao processo de engenharia reversa utilizando os padrões adotados.

Padrões Relacionados:

Esse padrão tem como entrada o padrão Construir Diagramas de Use-Cases e é utilizado como entrada para o padrão Tratar Anomalias.

Produtos Obtidos:

- Descrição de cada Use Case do sistema.

Figura 3.7 – Descrição do Use Case incluir Locadores, curso normal:

Use Case: Incluir Locadores
Curso: Normal

1 Usuário solicita inclusão de um locador;
2 Sistema gera automaticamente um código autonumeração para o locador a ser incluído;
3 Usuário entra com os dados do locador;
4 Sistema solicita ao usuário gravar ou alterar os dados;
5 Usuário aceita gravar os dados;
6 Sistema grava os dados que foram registrados no arquivo Locador.dbf
7 Sistema mostra dicas de como gravar e abandonar a tela, e exibe espaço para usuário inserir observações inerentes ao locador.

8. Nome: **Tratar Anomalias:**

Intuito: Analisar as descrições de Use Cases para tratar as anomalias, definindo assim, os possíveis métodos das pseudo-classes (candidatas a classe) do sistema, obtidas pelo padrão Criar Visão OO dos Dados.

Anomalia: Quando a descrição do Use Case faz acesso/atualização a arquivos de dados que não pertencem à pseudo-classe a que ele se relaciona, então infere-se que esse procedimento (Descrição do Use Case) é anômalo. A anomalia pode ser do tipo:

o+ Quando o procedimento é observador de duas ou mais classes;

c+ Quando o procedimento é construtor de duas ou mais classes;

oc Quando o procedimento é observador de uma classe e construtor de outra;

o+c Quando o procedimento é observador de duas ou mais classes e construtor de outra;

oc+ Quando o procedimento é observador de uma classe e construtor de duas ou mais classes;

o+c+ Quando o procedimento é observador de duas ou mais classes e construtor de duas ou mais classes;

i Quando o procedimento é dependente da implementação e que não se refere a classe alguma.

Problema: Em sistemas procedimentais pode existir um número elevado de anomalias em cada procedimento (código fonte). Deve-se eliminar essas anomalias uma vez que, em sistemas orientados a objetos, os métodos estão associados a uma única classe. Então, é necessário transformar o código fonte correspondente ao procedimento anômalo (Descrição do Use Case) em métodos.

Influências:

- Quando o sistema não tem implementação orientada a objetos, a análise dos procedimentos pode ser complexa.
- Requer do engenheiro de software experiência para extrair anomalias a partir da descrição do Use Case.
- Têm-se as descrições de Use Case geradas a partir do código fonte.

Solução:

Para a descrição de Use Case, obtida pelo padrão Elaborar Descrição de Use Case, construa a tabela Detalhes de Implementação, com seis colunas:

- A primeira coluna contém o nome do Use Case;
- A segunda coluna contém o nome de todos os programas/módulos (códigos fonte) utilizados na construção do Use Case;
- A terceira coluna contém o nome das Pseudo-Classes do MASA correspondentes aos arquivos de dados a que se tem acesso em cada programa/módulo;
- A quarta coluna contém a classificação quanto ao tipo de acesso ao arquivo de dados, realizado pelo procedimento;
- A quinta coluna contém os nomes dos possíveis métodos quando as anomalias forem eliminadas;
- A sexta coluna contém os nomes das pseudo-classes revisadas, a que os respectivos métodos estão associados;

1. Iniciar a construção da tabela Detalhes de Implementação, verificando para cada descrição de Use Case o tipo de acesso aos arquivos. Transformar o trecho fonte que consulta/altera diversos arquivos em possíveis métodos, eliminando assim o múltiplo

acesso a arquivos. A parte do código que somente consulta um arquivo é retirada desse, e uma chamada a esse método é feita para que a funcionalidade do sistema seja mantida.

2. Continuar a construção da tabela Detalhes de Implementação, renomeando as pseudo-classes que não representam completamente a informação dentro do contexto do sistema.

3. Acrescentar na tabela Detalhes de Implementação, os nomes de entidades do MER (arquivos de dados) existentes no código fonte que foram usados apenas para atender a necessidades de implementação. Esses arquivos são temporários e não fazem parte da funcionalidade do sistema. Portanto, toda pseudo-classe que é criada por algum procedimento classificado com a anomalia **i** deve ser desconsiderada.

4. Acrescentar na coluna pseudo-classes revisadas, da tabela Detalhes de Implementação, as pseudo-classes correspondentes a trechos do código fonte usados para implementar informações adicionais em memória. Por exemplo, tabelas de descontos em função da quantidade comprada, que foram implementadas em trecho de código ao invés de estarem persistidas em arquivos de dados. Em sistemas procedimentais esses artifícios eram comuns para melhorar o desempenho do sistema. Como essas pseudo-classes são geradas diretamente do código fonte, elas não aparecem como entidades do MER e, conseqüentemente, como pseudo-classes do MASA, pelo fato de não existir um arquivo de dados correspondente.

Avaliação:

Através desse padrão podemos visualizar quais arquivos fontes se relacionam com quais arquivos de dados e classificar o tipo das anomalias existentes no código.

Justificativa:

Esse padrão é importante pois, com o resultado final obtido dele, podemos continuar a análise do sistema, nos padrões do próximo Cluster (Modelar o Sistema Orientado e Objetos).

Padrões Relacionados:

Esse padrão tem como entrada o padrão Elaborar Construção de Use Cases e é utilizado como entrada para os padrões do *Cluster* Modelar o Sistema Orientado a Objetos.

Produtos Obtidos:

- Tabela Detalhes de Implementação.

Tabela 2 – Tabela Detalhes de Implementação:

USE CASE	CODIGOS FONTE CORRESPONDENTE	PSEUDO-CLASSE (MASA)	TIPO DA ANOMALIA	POSSÍVEIS MÉTODOS	PSEUDO-CLASSES REVISADAS
Incluir Locadores	Imo1001.prg	Locador		IncluirLocadores()	Locador
Incluir Imóveis	Imo1002.prg (oc)	Locador	o	SelecionarDadosLocador()	Locador
		Imóvel	c	IncluirImovel()	Imóvel
Incluir Locatários	Imo1003.prg (o+c+)(i)	Locador	o	SelecionarDadosLocador()	Locador
		Imóvel	o	SelecionarDadosImóvel()	Imóvel
		Locatário	c	IncluirLocatário()	Locatário
		Fiador	i	_____	Fiador
	Imo1301.prg (c+)	Fiador	c	IncluirFiador()	Fiador
	Imo1302.prg (c+)	Ficha	c	CadastrarFicha()	Ficha
Depósito de Proprietários	Imo1004.prg	Imóvel		SelecionarDadosImóvel()	Imóvel
	Imo1401.prg	Ficha		AlterarFicha()	Ficha
Alterar Locadores	Imo2001.prg (c+)	Locador	c	Alterar/ExcluirLocador()	Locador
		Imóvel	c	ExcluirImóvel()	Imóvel
		Locatário	c	ExcluirLocatário()	Locatário
		Fiador	c	ExcluirFiador()	Fiador
		Ficha	c	ExcluirFicha()	Ficha
Alterar Imóveis	Imo2002.prg (oc+)	Locador	o	SelecionarDadosLocador()	Locador
		Imóvel	c	Alterar/ExcluirImóvel()	Imóvel
		Locatário	c	ExcluirLocatário()	Locatário
		Fiador	c	ExcluirFiador()	Fiador
Alterar Locatários	Imo2003.prg (c+)(i)	Locador	o	SelecionarLocador()	Locador
		Imóvel	o	SelecionarImóvel()	Imóvel
		Locatário	c	Alterar/ExcluirLocatário()	Locatário
		Fiador	i	_____	Fiador
	Imo2301.prg	Fiador		ExcluirFiador()	Fiador
Alterar Ficha	Imo2004.prg	Locatário		SelecionarDadosLocatário()	Locatário
	Imo2401.prg	Ficha		AlterarFicha()	Ficha
Consulta Locadores	Imo3001.prg (o+)	Locador	o	SelecionarDadosLocador()	Locador
		Locatário	o	SelecionarDadosLocatário()	Locatário
Consulta Locatários	Imo3004.prg	Locatário		SelecionarDadosLocatário()	Locatário
Consulta Imóveis Alugados	Imo3002.prg	Imóvel		SelecionarImovelAlugado()	Imóvel
Consulta Imóveis Disponíveis	Imo3003.prg	Imóvel		SelecionarImovelDisponvel()	Imóvel
Agenda de Depósitos	Imo3005.prg (o+)	Locatário	o	ComparaDataDepósito()	Locatário
		Imóvel	o	CalculaDepósitolocador()	Imóvel
		Locador	o	SelecionarDadosLocador()	Locador
Aniversariantes do Mês	Imo3006.prg	Locador		SelecionarDadosLocador()	Locador
Emitir Recibo	Imo4001.prg (o+c)	Imóvel	o	SelecionarTipoImóvel()	Imóvel
		Locatário	o	SelecionarDadosLocatário()	Locatário
		Ficha	c	AlterarFicha()	Ficha
	Imo4101.prg (o+c)	Imóvel	o	SelecionarTipoImóvel()	Imóvel
		Locatário	o	SelecionarNomeLocatário()	Locatário
		Ficha	c	AlterarFicha()	Ficha
	Imo4002.prg (i)		i	_____	

Contratos	Imo4201.prg (o+)	Locador	o	SelecionarNomeLocador()	Locador
		Fiador	o	SelecionarDadosFiador()	Fiador
		Imóvel	o	SelecionarTipoImóvel()	Imóvel
		Locatário	o	SelecionarLocatário()	Locatário
	Imo4211.prg (o+c)	Locador	o	SelecionarNomeLocador()	Locador
		Fiador	o	SelecionarNomeFiador()	Fiador
		Locatário	o	SelecionarDadosLocatário()	Locatário
		Contrato	c	SelecionarContrato() Alterar/ImprimirContrato()	Contrato
	Imo4202.prg (o+)	Locador	o	SelecionarNomeLocador()	Locador
		Fiador	o	SelecionarNomeFiador()	Fiador
		Imóvel	o	SelecionarTipoImóvel()	Imóvel
		Locatário	o	SelecionarDadosLocatário()	Locatário
	Imo4222.prg (o+c)	Locador	o	SelecionarNomeLocador()	Locador
		Fiador	o	SelecionarNomeFiador()	Fiador
		Locatário	o	SelecionarDadosLocatário()	Locatário
		Contrato	c	SelecionarContrato() Alterar/ImprimirContrato()	Contrato
	Imo4255.prg (o+c)	Locador	o	SelecionarNomeLocador()	Locador
		Fiador	o	SelecionarNomeFiador()	Fiador
		Locatário	o	SelecionarDadosLocatário()	Locatário
		Contrato	c	SelecionarContrato() Alterar/ImprimirContrato()	Contrato
	Imo4203.prg	Contrato		SelecionarComunicado() Alterar/ImprimirComunicado()	Contrato
	Imo4204.prg	Contrato		SelecionarComunicado() Alterar/ImprimirComunicadoTel()	Contrato
	Imo4205.prg (o+)	Locador	o	SelecionarNomeLocador()	Locador
		Fiador	o	SelecionarNomeFiador()	Fiador
		Imóvel	o	SelecionarTipoImóvel()	Imóvel
		Locatário	o	SelecionarDadosLocatário()	Locatário
Locatários em Débito	Imo4003.prg (o+)	Imóvel	o	SelecionarPGTOAtrasado()	Imóvel
		Locatário	o	ImprimirLocatárioemDébito()	Locatário
Demonstrativo Depósito	Imo4004.prg (i)			———————	
	Imo4401.prg (o+)	Locatário	o	SelecionarNomeLocatário()	Locatário
		Locador	o	SelecionarNomeLocador()	Locador
		Imóvel	o	SelecionarTipoImóvel()	Imóvel
		Ficha	o	SelecionarMes()	Ficha
	Imo4402.prg (o+)	Locador	o	SelecionarNomeLocador()	Locador
		Imóvel	o	SelecionarTipoImóvel()	Imóvel
		Locatário	o	SelecionarNomeLocatário()	Locatário
		Ficha	o	SelecionarMes()	Ficha
Listagem Locadores	Imo4005.prg (o+)	Imóvel	o	SelecionarImóvel()	Imóvel
		Locador	o	SelecionarLocador()	Locador
Depósitos não Realizados	Imo4006.prg (o+)	Imóvel	o	SelecionarTipoImóvel()	Imóvel
		Locatário	o	SelecionarNomeLocatário()	Locatário
		Ficha	o	SelecionarMes()	Ficha
Imposto de Renda	Imo4007.prg (o+)	Locador	o	SelecionarNomeLocador()	Locador
		Imóvel	o	SelecionarTipoImóvel()	Imóvel
		Locatário	o	SelecionarNomeLocatário()	Locatário
Listagem Locatários	Imo4009.prg (o+)	Imóvel	o	SelecionarImóvel()	Imóvel
		Locatário	o	SelecionarNomeLocatário()	Locatário
Carta Cobrança Locatário	Imo4008.prg (i)			———————	
	Imo4801.prg (o+c)	Imóvel	o	SelecionarTipoImóvel()	Imóvel
		Locatário	o	SelecionarNomeLocatário()	Locatário
	Imo4811.prg	Cobrança	c	SelecionarCartaCobrança() Alterar/ImprimirCartaCobrança()	Cobrança
Contratos Vencidos	Imo4010.prg (o+)	Imóvel	o	SelecionarTipoImóvel()	Imóvel
		Locatário	o	SelecionarNomeLocatário()	Locatário
Carta Pedido	Imo4011.prg (o+)	Imóvel	o	SelecionarTipoImóvel()	Imóvel
		Locatário	o	SelecionarNomeLocatário()	Locatário

Imóvel	Imo4111.prg	Pedido		Alterar/ImprimirCartaPedido()	Pedido
Carta de Distrato	Imo4012.prg (o+)	Imóvel	o	SelecionarTipoImóvel()	Imóvel
		Locatário	O	SelecionarNomeLocatário()	Locatário
	Imo4112.prg	Pedido		Alterar/ImprimirCartaPedido()	Pedido
Carta Cobrança Fiador	Imo4013.prg (o+)	Fiador	o	SelecionarFiador()	Fiador
		Locatário	o	SelecionarNomeLocatário()	Locatário
	Imo4113.prg	Pedido		Alterar/ImprimirCartaCobrança()	Pedido
Emissão Etiqueta	Imo4014.prg (i)			_____	
	Imo4141.prg	Locador		ImprimirEtiquetaLocador()	Locador
	Imo4142.prg	Locatário		ImprimirEtiquetaLocatário()	Locatário
Organizar Arquivos	Imo5001.prg (o+)	Locador	o	IndexarArquivos()	Locador
		Imóvel	o	IndexarArquivos()	Imóvel
		Locatário	o	IndexarArquivos()	Locatário
		Fiador	o	IndexarArquivos()	Fiador
Reajuste	Imo5002.prg	Contrato		TipoReajusteContrato()	Contrato
Sair do Sistema	Imo.prg (i)				

CLUSTER – **MODELAR O SISTEMA ORIENTADO A OBJETOS:**

9. Nome: **Definir as Classes:**

Intuito: Iniciar a construção do Diagrama de Classes do Sistema.

Problema: Existem várias informações levantadas pelo processo de engenharia reversa. MER, MASA, Diagramas de Use Cases, Descrição de Use Cases e Tabela Detalhes de Implementação. No entanto, elas precisam ainda ser consolidadas e apresentadas na visão de orientação a objetos.

Influências:

- As classes são obtidas por meio da coluna Pseudo-Classes Revisadas, da Tabela Detalhes de Implementação.
- O engenheiro de software tem experiência na linguagem UML.

Solução:

1. Transformar cada pseudo-classe revisada, da Tabela Detalhes de Implementação, em uma Classe no Diagrama de Classes.

A transformação citada no item 1, (acima) é demonstrada na figura 3.8.

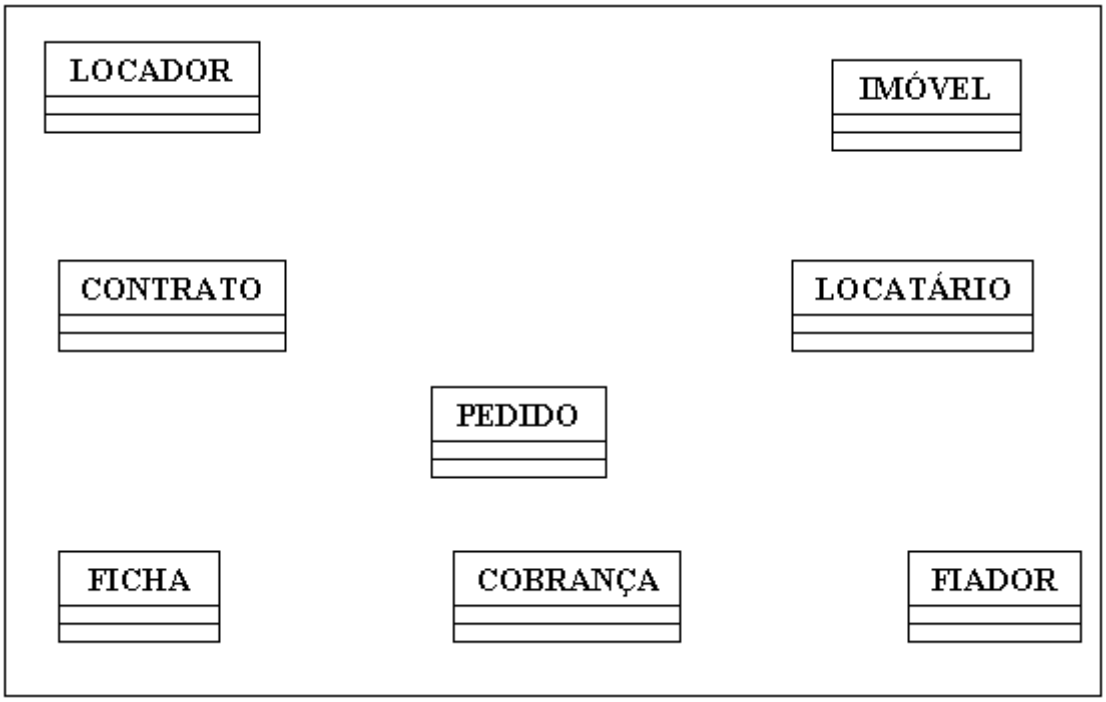


Figura 3.8 – Diagrama de Classes (Construção das Classes)

2. Gerar os relacionamentos no diagrama de classes a partir dos relacionamentos existentes do MASA. Além disso, para as pseudo-classes que foram geradas diretamente do código-fonte, criar relacionamentos que representam o princípio Todo Parte (Agregação por Referência ou Valor), bem como *link* de atributo, conforme os conceitos da orientação a objetos, discutidos pelo padrão Criar Visão OO dos Dados. A figura 3.9 ilustra os respectivos relacionamentos.

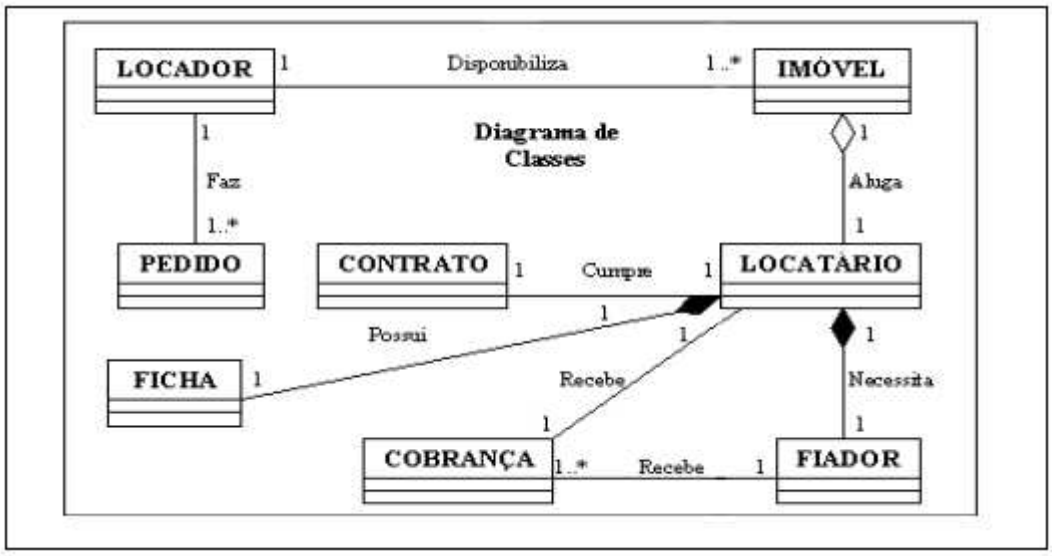


Figura 3.9 – Diagrama de Classes (Construção dos Relacionamentos)

Avaliação:

O objetivo deste padrão é obter todas as classes e seus relacionamentos, visando já uma transformação para a abordagem orientada a objetos.

Justificativa:

Esse padrão é um princípio para que possamos definir os atributos e continuar a construção do diagrama de classe final.

Padrões Relacionados:

Esse padrão tem como entrada o padrão Tratar Anomalias e serve como entrada para o padrão Definir Atributos.

Produtos Obtidos:

- Diagrama de Classes (inicial) do sistema: classes definidas;

10. Nome: **Definir Atributos:**

Intuito: Definir os atributos das classes pertencentes ao Diagrama de Classes em construção.

Problema: Identificar os atributos das classes definidas pelo padrão Definir Classes, a partir dos arquivos de dados do sistema legado existente.

Influências:

- Transformar os campos dos arquivos de dados em atributos de classes pode requerer uma análise complexa das estruturas de dados.
- Existem ferramentas que auxiliam a análise das estruturas de dados do legado.

Solução:

1. Os campos dos arquivos de dados (.dbf), representados como pseudo-classes no MASA são, na sua maioria, transformados em atributos das classes correspondentes. Dessa forma, representar esses atributos das classes, no Diagrama de Classes.
2. Os casos em que não há correspondência direta deverão ser tratados em paralelo, avaliando-se as questões de hierarquias abordadas pelo padrão Analisar Hierarquias.

Avaliação:

Este padrão é uma continuação do padrão anterior, onde detalhamos melhor o Diagrama de Classes.

Justificativa:

Sem o uso desse padrão seria mais difícil encontrar os atributos das classes do sistema.

Padrões Relacionados:

Esse padrão tem como entrada o padrão Definir as Classes e serve como entrada para o padrão Definir Métodos. Pode ser aplicado, se necessário, em paralelo ao padrão Analisar Hierarquias.

Produtos Obtidos:

- Diagrama de Classes (continuação) do Sistema: classes e atributos definidos.

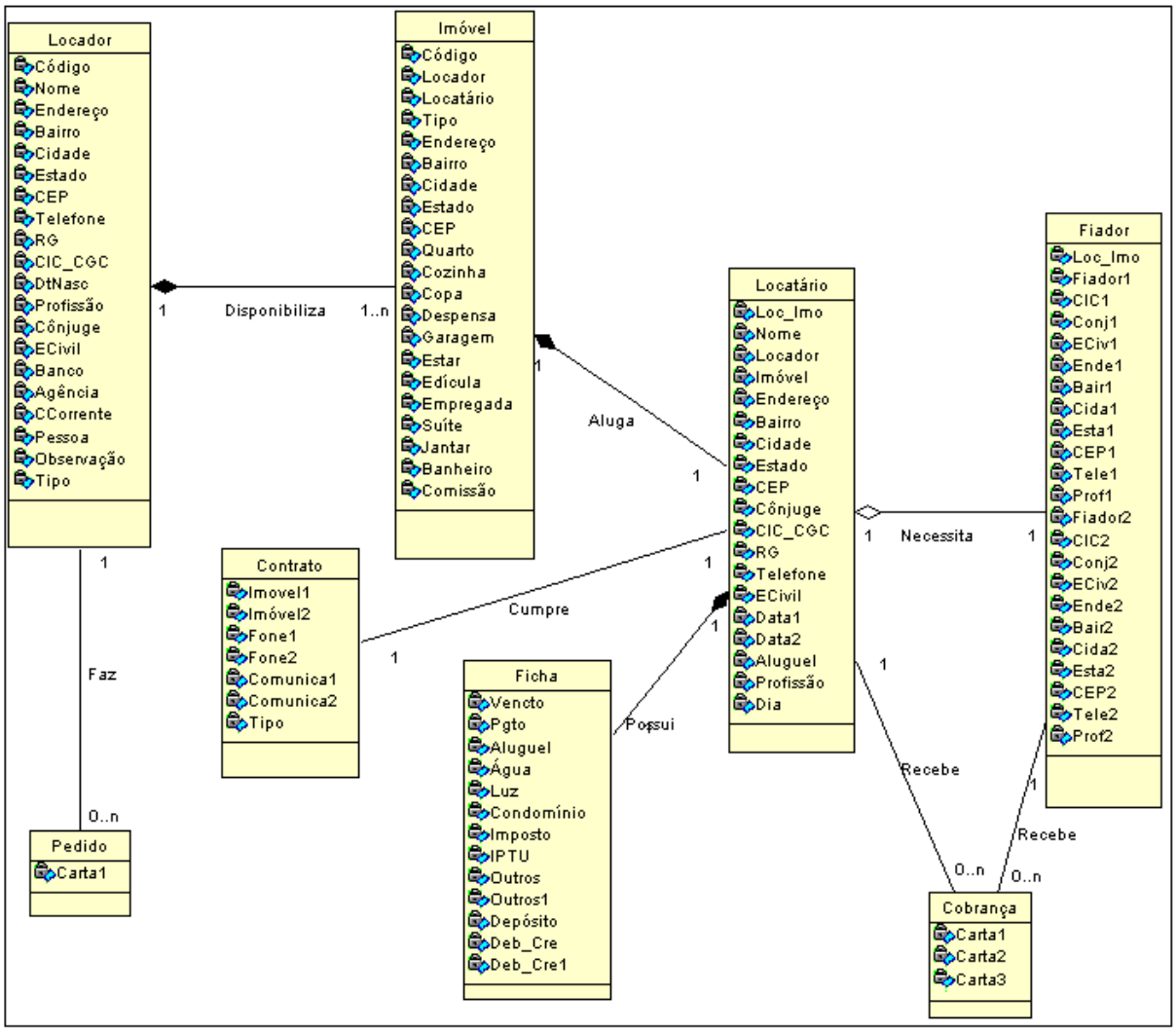


Figura 3.10 – Diagrama de Classes (Definição dos Atributos das Classes)

11. Nome: **Analisar Hierarquias:**

Intuito: Fornecer o mecanismos para descobrir possíveis papéis (mais de uma função) e hierarquias de herança que possam estar contidas nos arquivos de dados do sistema legado.

Problema: Encontrar papéis e hierarquias de herança que possam estar persistidos como campos em vários arquivos de dados do sistema legado.

Influências:

- Hierarquias de herança podem estar contidas nos arquivos de dados das seguintes formas:

- Por meio de conjunto de campos opcionais (campos que assumem valores dependendo de um contexto. Por exemplo, um arquivo de dados de nome Cliente possui os campos CPF e CGC, sendo que para cada cliente, somente um desses campos assumirá valor, dependendo do tipo do cliente – pessoa física ou jurídica);

- Por meio de conjuntos de campos semelhantes (mesmo campos em diferentes arquivos de dados);

    - Hierarquias de Herança podem também estar contidas em relacionamentos do tipo um para um, ligando várias pseudo-classes do MASA.

    - Papéis podem estar contidos nos arquivos de dados, por exemplo, por meio da cláusula Redefines do Cobol.

    - O Engenheiro de Software possui experiência para definir super-classe, a partir de vários arquivos de dados, e também de transformar super-classe em várias sub-classes, a partir da pseudo-classe do MASA

Solução:

1. Encontrar, no sistema legado, os arquivos de dados (.dbf) com campos opcionais. Isso indica uma situação em que uma hierarquia de classe completa está representada em um único arquivo de dados. Nesse caso, transformar esse arquivo de dados numa super-classe e gerar tantas classes quanto forem necessárias para representar esse conjunto de informações. A figura 3.11 mostra que os arquivos de dados Locatário.dbf (entidade Locatário) e Locador.dbf (entidade Locador) são considerados arquivos com atributos opcionais, contendo informações tanto de cliente Pessoa Física quanto Pessoa Jurídica. Portanto esse arquivo de dados é definido como uma super-classe, sendo criadas as sub-classes: Pessoa Física e Pessoa Jurídica.

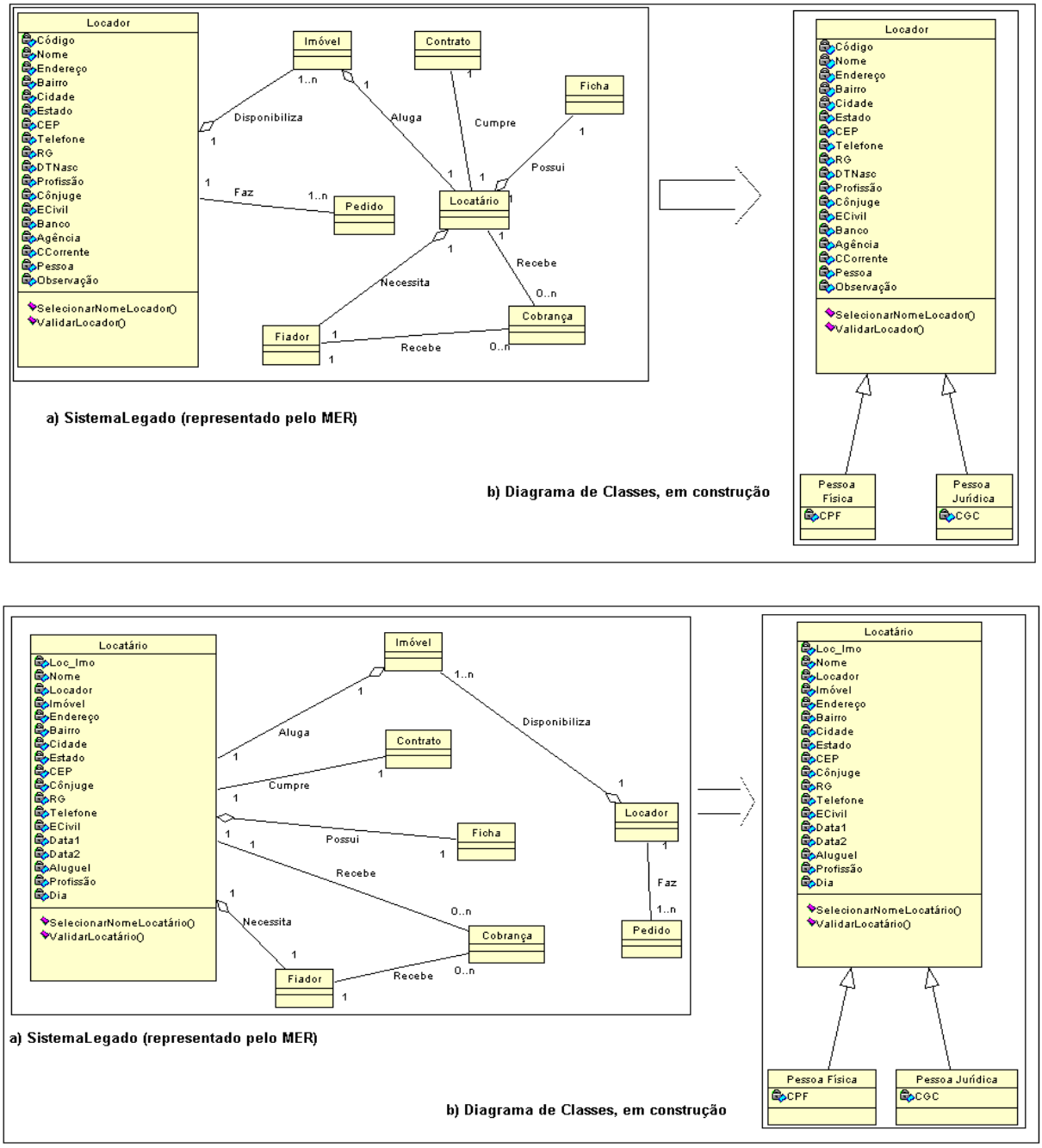


Figura 3.11 – Arquivos de Dados com atributos opcionais

2. Encontrar, no sistema legado, os arquivos de dados com campos semelhantes. Isso indica que a hierarquia de classe está distribuída entre vários arquivos de dados. Nesse caso, definir uma super-classe

movendo os campos comuns para ela e, os não comuns, continuam pertencendo às respectivas sub-classes.

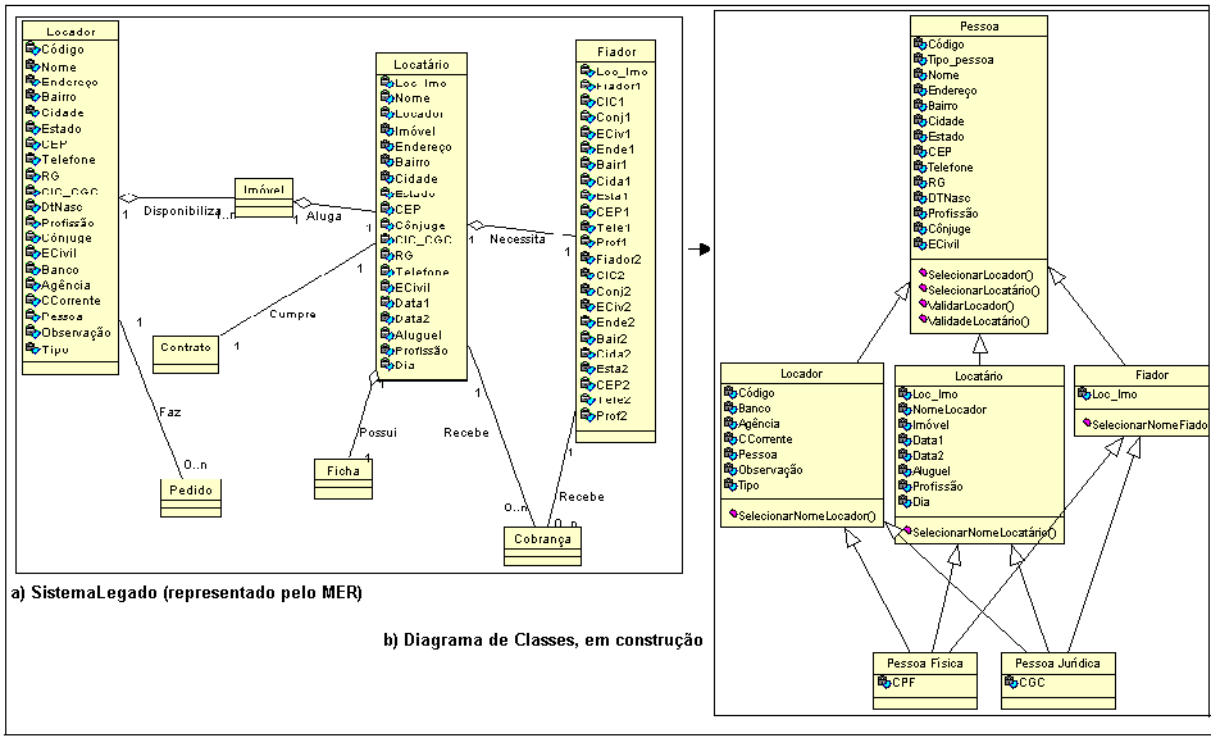


Figura 3.12 – Arquivos de Dados com atributos semelhantes

3. Encontrar, no MASA, as pseudo-classes nas quais a chave primária também serve como chave estrangeiras de outra pseudo-classe. Isto pode representar um relacionamento um para um, indicando talvez, um relacionamento de herança. A figura Nesse caso, deve-se analisar a funcionalidade do sistema legado, e representar esse conjunto de pseudo-classes como uma Hierarquia de Herança entre Classes, quando a essa característica for validade.A figura 3.13(a) mostra que foram encontradas no MASA, seis pseudo-classes relacionadas (por relacionamentos um-para-um), trata-se das pseudo-classes Pessoa, Locador, Locatário, Fiador, Pessoa Física e Pessoa Jurídica. Nesse caso, após analisar a funcionalidade do sistema legado, representa-se diretamente esse conjunto de pseudo-classes como uma hierarquia de herança entre classes, figura 3.13(b). O engenheiro de software deverá

observar, no MASA, situações de relacionamento desse tipo, devendo sempre representá-los como uma hierarquia de heranças no diagrama de classes em construção, para assim consolidar cada vez mais esse diagrama no conceito de orientação a objetos.

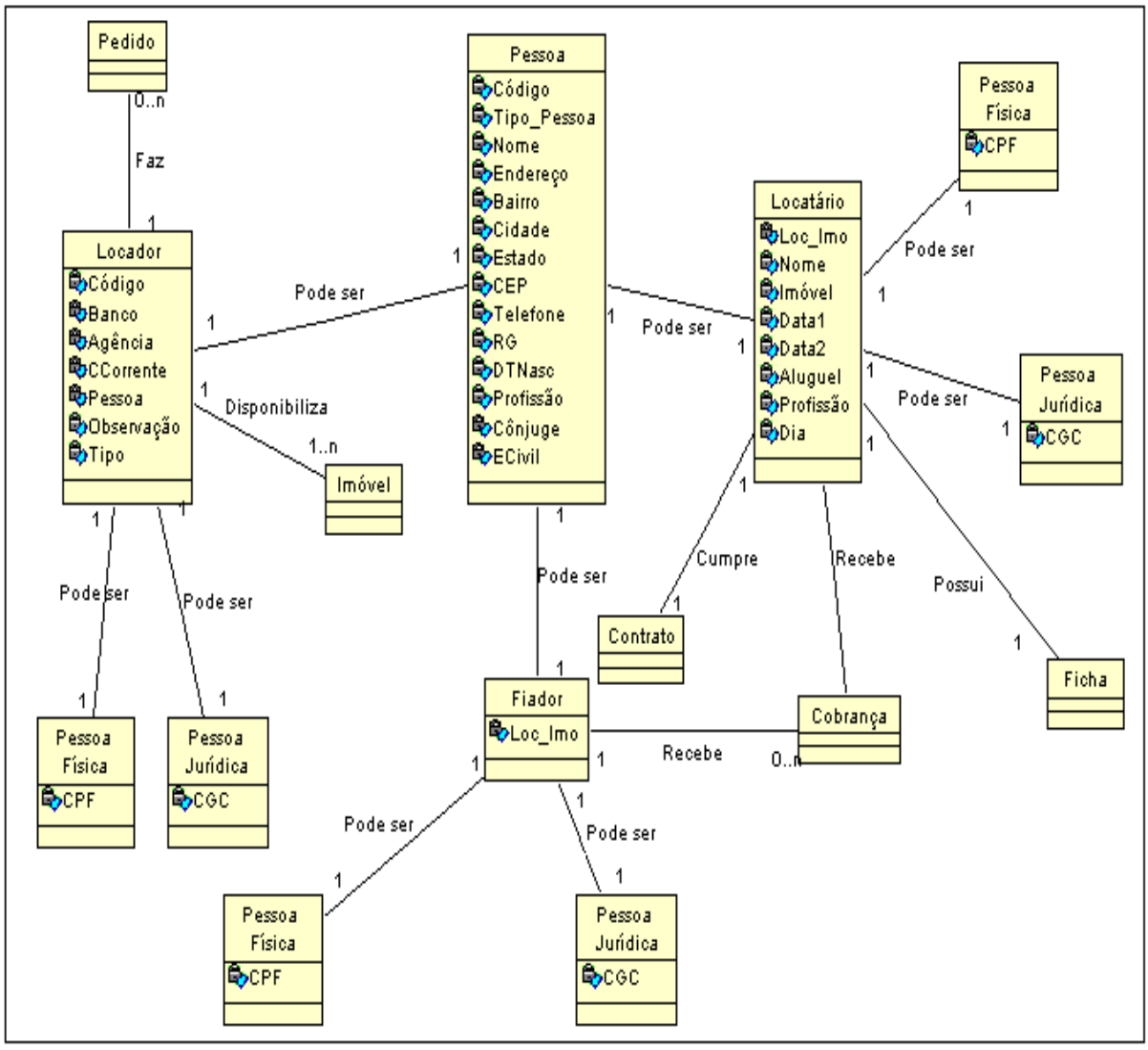


Figura 3.13 (a) – MASA

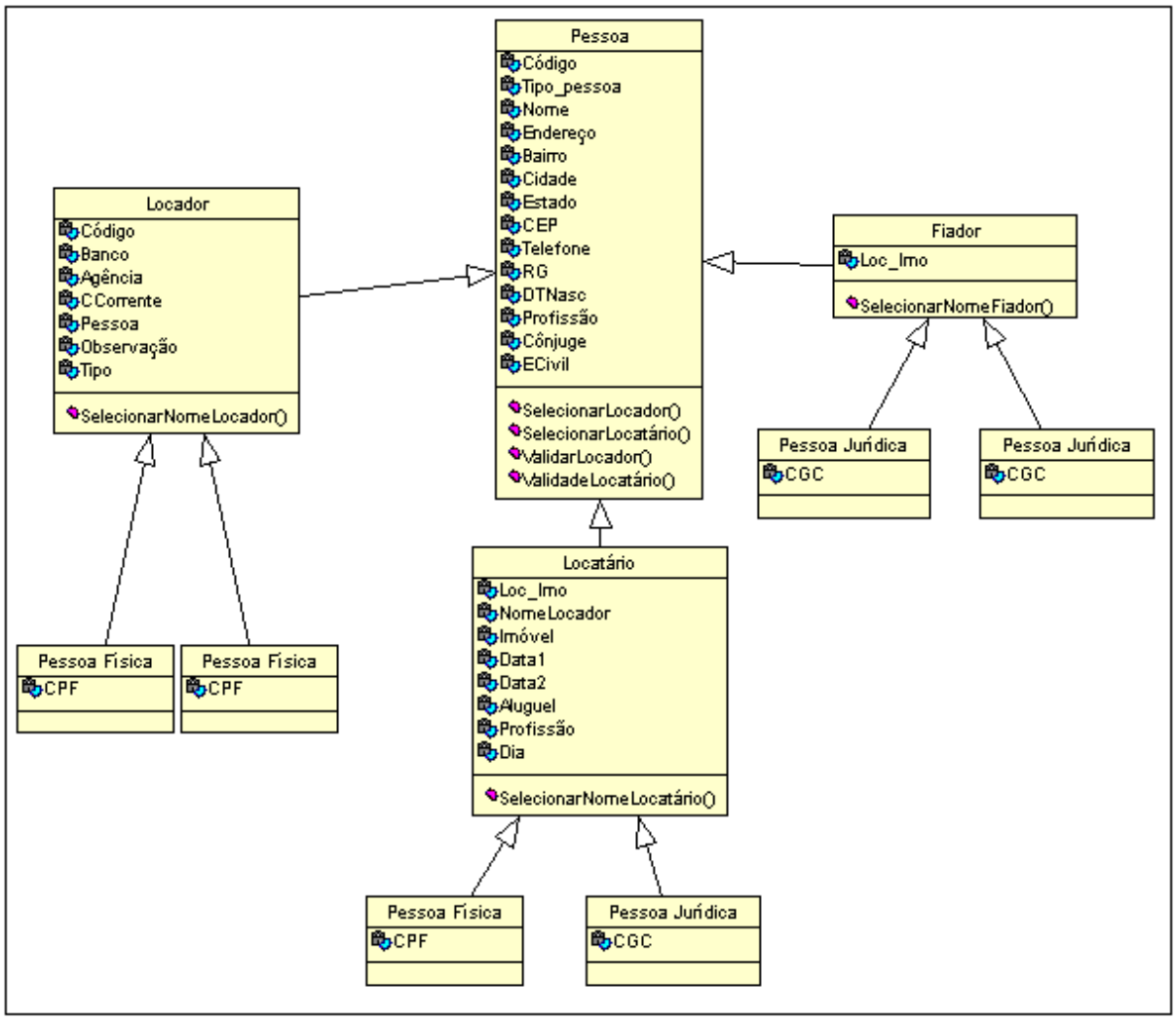


Figura 3.13 (a) – MASA

4. Encontrar, no sistema legado, os arquivos de dados com os campos possuindo um ou mais papéis. Nesse caso, para cada papel, gerar uma nova classe no Diagrama de Classes. Essa situação ocorre com freqüência em programas Cobol (cláusula *Redefines*), no entanto, nos programas em Clipper os campos de cada arquivo de dados possui unicamente um papel, logo esse passo não requer detalhamento nesta extensão para Clipper na Família de Padrões de Engenharia Reversa.

Avaliação:

Este padrão é uma continuação do padrão anterior, onde detalhamos melhor o Diagrama de Classes.

Justificativa:

Sem o uso desse padrão seria mais difícil encontrar os atributos das classes do sistema.

Padrões Relacionados:

Esse padrão tem como entrada o padrão Definir as Classes e Definir Visão OO dos Dados. Pode ser aplicado, se necessário, em paralelo ao padrão Definir Atributos.

Produtos Obtidos:

- Diagrama de Classes (continuação) do Sistema: classes e atributos definidos.

12. Nome: **Definir Métodos:**

Intuito: Definir os métodos das classes pertencentes ao Diagrama de Classes em construção.

Problema: Associar métodos às classes do Diagrama de Classes a partir da tabela Detalhes de Implementação.

Influências:

- Definir cada um dos métodos a partir dos procedimentos anômalos existentes no código fonte legado é difícil.
- Existem indícios dos prováveis métodos na Tabela Detalhes de Implementação.

Solução:

Representar os métodos que estão relacionados na coluna Possíveis Métodos, da Tabela Detalhes de Implementação, nas classes respectivas às pseudo-classes, especificadas na coluna Pseudo Classes Revisadas, da mesma tabela. Assim, obtém-se os métodos das classes no Diagrama de Classes em construção por meio da descrição do Use Case correspondente. A figura 3.14 ilustra o Diagrama de Classes já com os métodos definidos.

Informações Adicionais:

O engenheiro de software deve observar trechos do código fonte que possam ser caracterizados como Polimorfismo, gerando tantos métodos quanto forem necessários, consolidando assim, o Diagrama de Classes do sistema no paradigma orientado a objetos.

Avaliação:

Com a aplicação deste padrão podemos então terminar a construção do Diagrama de Classes, com seus métodos e atributos.

Justificativa:

Este padrão auxilia, juntamente com o padrão Tratar Anomalias, a identificar os métodos e construir o Diagrama de Classes.

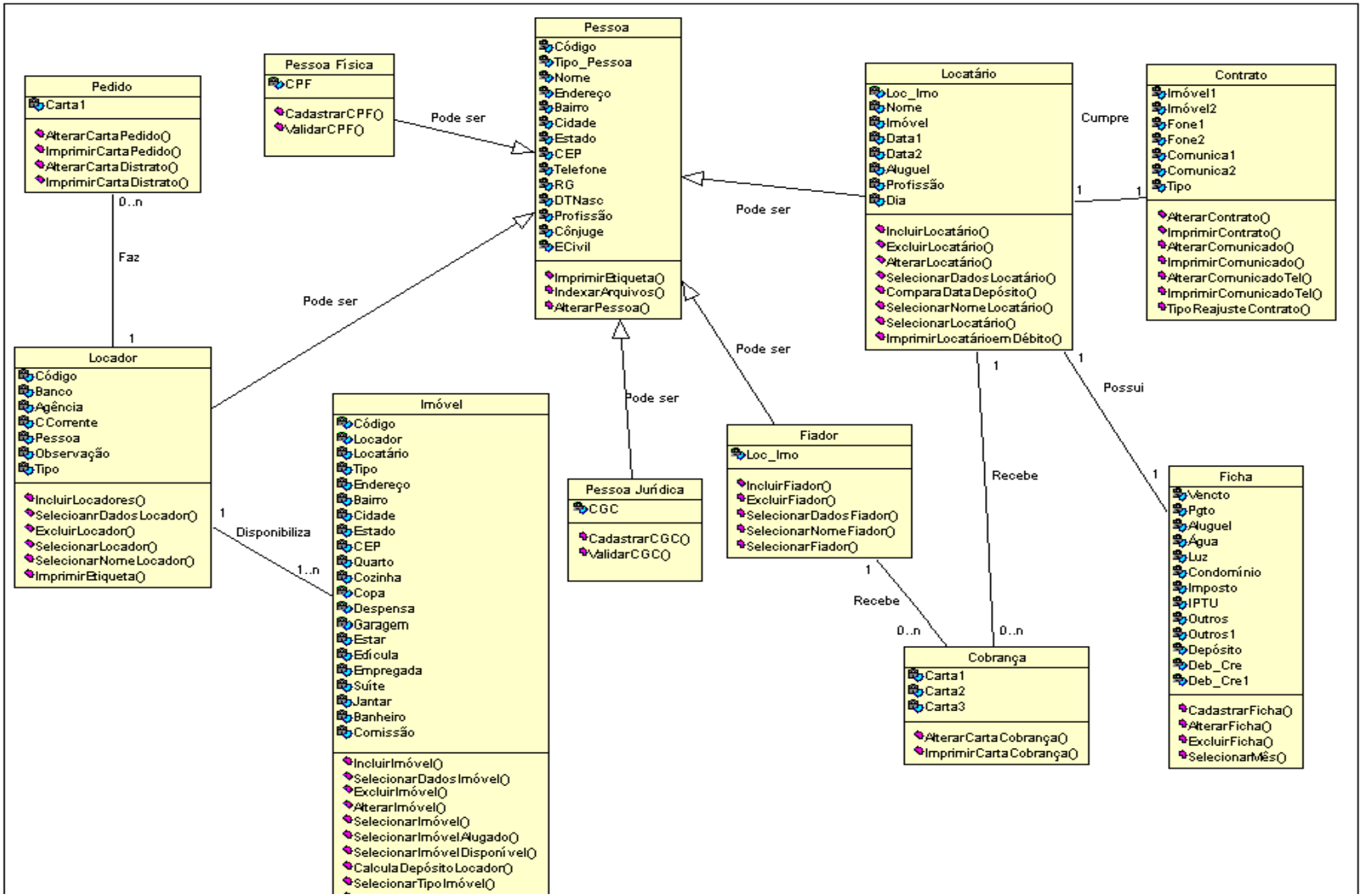
Padrões Relacionados:

Esse padrão tem como entrada o padrão Tratar Anomalias.

Produtos Obtidos:

- Diagrama de Classes do Sistema.

Figura 3.14 – Diagrama de Classes (Definição dos Métodos das Classes):


Pessoa
Código
Tipo_Pessoa
Nome
Endereço
Bairro
Cidade
Estado
CEP
Telefone
RG
DTNasc
Profissão
Cônjuge
ECivil
ImprimirEtiqueta()
IndexarArquivos()
AlterarPessoa()
Pessoa Física
CPF
CadastrarCPF()
ValidarCPF()
Pode ser
Pedido
Carta1
AlterarCartaPedido()
ImprimirCartaPedido()
AlterarCartaDistrato()
ImprimirCartaDistrato()
0..n
Faz
1
Locador
Código
Banco
Agência
CCorrente
Pessoa
Observação
Tipo
IncluirLocadores()
SelecioanrDadosLocador()
ExcluirLocador()
SelecionarLocador()
SelecionarNomeLocador()
ImprimirEtiqueta()
Pode ser
1
Disponibiliza
1..n
Imóvel
Código
Locador
Locatário
Tipo
Endereço
Bairro
Cidade
Estado
CEP
Quarto
Cozinha
Copa
Despensa
Garagem
Estar
Edícula
Empregada
Suíte
Jantar
Banheiro
Comissão
IncluirImóvel()
SelecionarDadosImóvel()
ExcluirImóvel()
AlterarImóvel()
SelecionarImóvel()
SelecionarImóvelAlugado()
SelecionarImóvelDisponível()
CalculaDepósitoLocador()
SelecionarTipoImóvel()
Pessoa Jurídica
CGC
CadastrarCGC()
ValidarCGC()
Pode ser
Pode ser
Fiador
Loc_Imo
IncluirFiador()
ExcluirFiador()
SelecionarDadosFiador()
SelecionarNomeFiador()
SelecionarFiador()
1
Recebe
0..n
Locatário
Loc_Imo
Nome
Imóvel
Data1
Data2
Aluguel
Profissão
Dia
IncluirLocatário()
ExcluirLocatário()
AlterarLocatário()
SelecionarDadosLocatário()
ComparaDataDepósito()
SelecionarNomeLocatário()
SelecionarLocatário()
ImprimirLocatárioemDébito()
Pode ser
Cumpre
1
1
Contrato
Imóvel1
Imóvel2
Fone1
Fone2
Comunica1
Comunica2
Tipo
AlterarContrato()
ImprimirContrato()
AlterarComunicado()
ImprimirComunicado()
AlterarComunicadoTel()
ImprimirComunicadoTel()
TipoReajusteContrato()
1
Recebe
0..n
1
Possui
1
Cobrança
Carta1
Carta2
Carta3
AlterarCartaCobrança()
ImprimirCartaCobrança()
Ficha
Vencto
Pgto
Aluguel
Água
Luz
Condomínio
Imposto
IPTU
Outros
Outros1
Depósito
Deb_Cre
Deb_Cre1
CadastrarFicha()
AlterarFicha()
ExcluirFicha()
SelecionarMês()

13. Nome: **Construir Diagramas de Seqüência:**

Intuito: Documentar as regras de negócio do sistema legado para facilitar o seu futuro processo de engenharia avante.

Problema: A necessidade de explicitar as Regras de Negócio.

Influências:

- Representar a iteração dos métodos obtidos no padrão Definir Métodos, pode ser complexa.
- Métodos já definidos a partir das Descrições de Use Cases, viabilizam a solução deste problema.

Solução:

Construir todos os Diagramas de Seqüência, a partir de cada descrição de Use Case, obtida pelo padrão Elaborar Descrição de Use Case. Considere, para a sua construção, o processo de eliminação de anomalias efetuado pelo padrão Tratar Anomalias, quando foram gerados tantos métodos quanto foram necessários, para manter a mesma funcionalidade do sistema legado. Agora, no Diagrama de Seqüência, é representada graficamente a iteração destes métodos. Os diagramas de seqüência estão ilustrados nos itens da figura 6 do Apêndice 1.

Avaliação:

Com a aplicação deste padrão pudemos detalhar mais as funções que começaram a ser explicitadas no padrão Elaborar Descrição de Use Case.

Justificativa:

Este padrão nos dá a última parcela de contribuição para a etapa de Engenharia Reversa de acordo com o padrão adotado.

Padrões Relacionados:

Esse padrão tem como entrada os padrões Elaborar Descrição de Use Cases, Definir Métodos e Tratar Anomalias.

Produtos Obtidos:

- Diagramas de Seqüência do Sistema.

3.5 CONCLUSÃO:

O capítulo 3 conduziu passo a passo o processo de Engenharia Reversa baseada na Família de Padrões de Reengenharia. Aplicou-se como exemplo o Sistema de Controle de Locações de uma Imobiliária. A seguir é mostrada a etapa de Engenharia Avante baseada na mesma Família de Padrões.

4 PROCESSO DE ENGENHARIA AVANTE DO SISTEMA PARA CONTROLE DE LOCAÇÕES DE UMA IMOBILIÁRIA BASEADO EM PADRÕES

O capítulo 4 traz as etapas do processo de Engenharia Avante para sistemas legados baseado na Família de Padrões de Reengenharia Orientada a Objetos. Neste contexto o engenheiro de software aplicará a mudança de plataforma, de Clipper para Delphi. Neste capítulo só há um cluster denominado Gerar o Sistema Orientado a Objetos, sendo que este possui 4 padrões a serem aplicados.

Os padrões de Engenharia Avante são descritos de acordo com o seguinte formato: **Nome, Intuito, Aplicabilidade, Contexto (Influências), Solução, Avaliação, Justificativa e Produtos Obtidos**.

CLUSTER – **GERAR O SISTEMA ORIENTADO A OBJETOS:**

1. Nome: **Definir a Plataforma:**

Intuito: Definir a plataforma de desenvolvimento do novo sistema conforme as exigências do negócio e dos objetivos estratégicos da organização empresarial.

Aplicabilidade: Em organizações empresariais que desejam modernizar a Tecnologia da Informação usada como suporte às realizações do seu negócio.

Influências:

- O atual e o futuro ambiente operacional da organização empresarial são objetos de estudo que devem incluir as seguintes análises:

- O quê de fato já existe.
- Quais os atuais e os futuros recursos operacionais disponibilizados.
- A cultura da organização.
- O histórico de desenvolvimentos realizados.
- O foco estratégico da organização.
- Quais os processos de negócio que serão suportados.

- Avaliar as considerações anteriores, juntamente com a área estratégica da organização, pois a mudança de um ambiente operacional envolve custos, tais como:

- Treinamento de usuários na nova tecnologia.
- Investimentos em infra-estrutura que envolvem a solução adotada, podendo ser necessário adquirir: novos computadores servidores, e estações de trabalho, *routhers* e *switchs* para interligação remota destas estações. Também pode ocorrer a necessidade

de reestruturar toda a rede de informática, efetuando-se a mudança de cabos coaxiais para a tecnologia de par-trançado.

- Investimento em todo o software necessário para a implementação da solução escolhida, podendo ser necessário adquirir: um novo sistema operacional, tal como: Novell, Windows NT; um novo Banco de Dados Relacional; uma ferramenta de desenvolvimento, tal como Java, Delphi, softwares de apoio, tais como backup, correio interno, etc.

- Investimento no desenvolvimento do aplicativo. Poder-se-á optar pela terceirização desses serviços, ou treinar toda a equipe técnica da empresa na nova tecnologia, não desconsiderando que essa equipe deverá manter o atual sistema legado em operação durante toda a fase de desenvolvimento do novo aplicativo.

    - O engenheiro de software tem à sua disposição cópias de demonstração de todos os pacotes, utilitários e ferramentas de suporte técnico.

    - O engenheiro de software tem um bom conhecimento de técnicas de reuniões para se obter um consenso da área estratégica da organização.

Solução:

1. **Escolher o ambiente operacional:**

2. **Escolher o Banco de Dados Relacional**:

3. **Escolher o pacote da linguagem de programação**:

Para o trabalho aqui desenvolvido, não será implementada a conversão do Banco de Dados, visto não ser o objetivo deste trabalho a implementação do sistema e sim apenas o processo de reengenharia. A linguagem escolhida para a nova plataforma será a linguagem Delphi.

Produtos Obtidos:

Plataforma de implementação definida:

- Ambiente operacional;
- Banco de Dados Relacional;
- Pacote da Linguagem de Implementação.

2. Nome: **Converter o Banco de Dados:**

Intuito: Criar as estruturas de dados do Sistema, fisicamente, de acordo com o Diagrama de Classes.

Produtos Obtidos:

- Banco de dados do Sistema;

O Banco de dados do Sistema não foi efetivamente implementado, visto que o objetivo principal deste projeto não era a implementação e sim apenas o processo de engenharia reversa do mesmo.

3. Nome: **Implementar os Métodos:**

Intuito: Escrever os métodos obtidos na engenharia reversa na linguagem de programação escolhida para o processo de engenharia avante.

Produtos Obtidos:

Métodos programados na linguagem de implementação escolhida para reengenharia. Ambiente operacional;

Os métodos identificados no Diagrama de Classes desenvolvido no processo de Engenharia Reversa do sistema não foram efetivamente implementados, visto não ser a implementação o objetivo principal deste trabalho.

4. Nome: **Realizar melhorias na Interface:**

Intuito: Conceber sistemas que atendam, cada vez melhor, às necessidades dos usuários em relação não apenas a critérios de funcionalidade (conjunto de tarefas desempenhadas pelo sistema), mas também à usabilidade.

Aplicabilidade: Existem diversas telas de acesso às funções do sistema legado, desenvolvidas para o ambiente DOS (caracter), totalmente peculiar às aplicações implementadas em Clipper, Cobol, etc. Então é necessário conceber nova interface para o ambiente Windows (gráfico), que atenda também aos critérios de usabilidade, e que possa ser construída na linguagem de implementação escolhida para o processo de engenharia avante.

Influências:

- Exige do engenheiro de software habilidade em conseguir transformar telas do ambiente caracter em telas no ambiente Windows.
- Exige do engenheiro de software, experiência no desenvolvimento de interfaces que:
    - Sejam consistentes;
    - Tenham atalhos para usuários experientes;
    - Tenham *Feedback* informativo;
    - Seja organizada quanto aos diálogos de interação;
    - Previna erros de usuários;
    - Seja possível a reversibilidade de ações;
    - Reduza a necessidade de memorização;
- O engenheiro de software tem à sua disposição todas as telas de tratamento da informação, do sistema legado.

- O engenheiro de software tem um bom conhecimento da linguagem de implementação para construir interfaces que viabilizem:
    - Facilidade de aprendizado;
    - Facilidade de utilização;
    - Ser intuitiva;
    - Diálogo simples e natural;
    - Velocidade na execução das tarefas;
    - Mensagens de erros consistentes;
    - Satisfação subjetiva;

Solução:

1. Para cada tela do ambiente caracter do sistema legado, gerar conjunto de interfaces na linguagem de implementação escolhida para a reengenharia, que realizem melhorias por meio dos conceitos de usabilidade.

Avaliação: Este padrão é muito importante, visto que o resultado obtido nele será o que o cliente efetivamente terá acesso, e estas melhorias na interface serão um dos critérios principais para a sua avaliação final sobre o sistema.

Justificativa: Este padrão nos ajuda a melhorar as condições de usabilidade do sistema, facilitando a interação do cliente/usuário com o sistema já modificado.

Produtos Obtidos:

Sistema que passou pelo processo de reengenharia, tendo interfaces concebidas, com base nos critérios de usabilidade.

As novas telas do sistema legado estão todas no Apêndice B deste trabalho.

5 CONCLUSÕES

5.1 REVISÃO:

O trabalho aqui desenvolvido foi organizado em 5 capítulos. No primeiro capítulo foi falado sobre a introdução sobre o projeto, seu objetivo e motivações.

No capítulo 2 foi tratada a revisão bibliográfica, os principais conceitos de reengenharia, engenharia reversa e engenharia avante e a Família de Padrões de Reengenharia Orientado a Objetos FaPRE/OO (Recchia et al 2002).

O capítulo 3 tratou do desenvolvimento principal do trabalho, o desenvolvimento da engenharia reversa do sistema, com aplicação da família de padrões escolhida.

O capítulo 4 trouxe o processo de engenharia progressiva do sistema, baseado nos modelos obtidos no capítulo 3. Foi visto também neste capítulo que apenas a modelagem das telas do novo sistema foi desenvolvido, pois o objetivo deste não era a implementação.

O capítulo 5 está trazendo as conclusões tiradas através de todo o projeto e as propostas para trabalhos futuros.

5.2 CONSIDERAÇÕES FINAIS:

Com o advento da Internet e das linguagens orientadas a objetos, as empresas estão decidindo mudar a plataforma de trabalho de seus sistemas. Os sistemas que estas empresas costumam utilizar são sistemas antigos que foram desenvolvidos especificamente para atender às suas necessidades há tempos atrás.

Com o intuito de continuar progredindo junto com as novas tecnologias, estes clientes acabam comprando pacotes de softwares prontos, pelas firmas de desenvolvimento de software e até mesmo pela Internet. Estes pacotes quase sempre não atendem totalmente os interesses de necessidades da empresa/cliente. Como resultado a empresa precisa se adaptar ao novo software para continuar trabalhando normalmente, quando na verdade, o software é que precisa ser adequado às necessidades da empresa.

O grande desafio dos engenheiros de software é elaborar um sistema novo, mas com todas as funcionalidades que o antigo possuía, e ainda com mais facilidades para que os usuários não criem uma certa aversão ao novo produto.

Os padrões utilizados aqui, comprovaram que o engenheiro de software pode executar de forma total e correta a reengenharia de um sistema legado, e não só os desenvolvidos em Clipper (como o caso deste trabalho), mas também os de todas as linguagens consideradas procedimentais.

A metodologia adotada, FaPRE/OO, se utiliza de 17 padrões divididos em 4 clusters, e no final, garante a reengenharia do sistema.

Os três primeiros clusters tratam da engenharia reversa do sistema, onde as informações são na maioria tiradas do código fonte do programa, visto que sistemas legados normalmente não possuem documentação.

O quarto cluster coloca os passos da engenharia progressiva do sistema, onde, com a utilização dos resultados da engenharia reversa, pode-se modelar o novo software e acoplar novas funções de funcionalidade e usabilidade no mesmo.

O resultado da implantação desta metodologia é um novo sistema com melhor qualidade, documentação completa do processo, que acompanha o novo software, e finalmente o novo sistema possui abertura para manutenções futuras que podem surgir de acordo com as novas necessidades do cliente.

5.3 TRABALHOS FUTUROS:

Como sugestão de trabalho futuro poderia ser colocado o fato de que o sistema aqui estudado passou por um processo de reengenharia, mas não foi efetivamente implementado. Fica a sugestão para os alunos que vierem a ter acesso ao trabalho desenvolvido, de que possam terminar o que foi começado, implementando de fato o sistema, a partir do seu diagrama de classes, e das telas que já foram mostradas no apêndice B.

Ainda como sugestão de trabalhos futuros poderia-se pensar em aplicar a metodologia aqui mostrada em outros sistemas que não implementados em Clipper, com mudança de plataforma também para Delphi ou para qualquer outra linguagem orientada a objetos, e assim ter certeza que o padrão se aplica a qualquer linguagem procedimental.

REFERÊNCIAS BIBLIOGRÁFICAS

[Monografia]BRAGA, Rosana Terezinha Vaccare. **Padrões de Software a Partir da Engenharia Reversa de Sistemas Legados**, 1998. Monografia - Instituto de Ciências Matemáticas e de Computação, USP, São Paulo.

[Monografia]RECCHIA, Edson Luiz. **Engenharia Reversa e Reengenharia Baseada em Padrões**, 2002. Monografia - Instituto de Ciências Matemáticas e de Computação, USP, São Paulo.

[Monografia]QUINAIA, Marcos Antônio. **Diretrizes para Reengenharia de Software com Características de Software Legado**, 1998. Monografia - Instituto de Ciências Matemáticas e de Computação, USP, São Paulo.

[Parte de uma obra com autor específico]PENTEADO, R. A. D.; GERAMNO, F. W. R.; MASIERO, P. C. Engenharia Reversa Orientada a Objetos do Ambiente StatSim: método e resultados obtidos. In: IX Simpósio Brasileiro de Engenharia de Software, Recife, 1995.

[Parte de uma obra com autor específico]PENTEADO, R. A. D.; GERAMNO, F. W. R.; MASIERO, P. C. Melhria da Manutenibilidade de Software Legado com Base em Engenharia Reversa. In: Anais da Conferência Internacional de Tecnologia de Software: Qualidade de Software, Curitiba, 1996.

[Informação de Internet (www)]CLIPPER ON LINE,. **Curso de Clipper, Comandos Básicos**. <http://www.pctoledo.com.br/col_cdr.htm> Acesso em 28 abr. 2003.

[Informação de Internet (www)]MUCURI, O SITE DO PROGRAMADOR. **Clipper, Comandos Básicos**. <http://www.mucuri.cjb.net> Acesso em 28 abr. 2003.

[Informação de Internet (www)]ANAIS SBES. **Reengenharia de software**. <http://www.inf.ufsc.br/sbes99/anais/sbes-Completo/26.pdf> Acesso em 28 abr. 2003.

[Informação de Internet (www)]RATIONAL. **Rational Rose – Tutorial – CES-63.htm**. <http:// www.comp.ita.br/~drikams/tutorial.htm > Acesso em 12 mai. 2003.

[Informação de Internet (www)]. **UML, modelagem estática**. <http:// wwwedit.inf.ufsc.br:1194/users/grupo2/fin/page01.html > Acesso em 12 mai. 2003.

[Informação de Internet (www)]UFSC. **UML, tutorial**. <http://www.edit.inf.ufsc.br/1194/users/grupo2/fin/page01.html#3> Acesso em 12 mai. 2003.

[Informação de Internet (www)]ITA. **Tutorial ferramenta rose**. <http://www.compita.br/~olany/rose_tut.htm> Acesso em 12 mai. 2003.

[Informação de Internet (www)]BARROS, Pablo. **UML – Linguagem de Modelagem Unificada**. <http://www.cc.usu.edu/~slqz9/uml/. Acesso em 12 mai. 2003.

[Informação de Internet (www)]FOX HOUSE, A TOCA DA RAPOSA. **UML – linguagem de Modelagem Unificada**. <http://www2.foxbrasil.com.br:8080/paulino/Projetos/OOP/UML.htm> Acesso em 12 mai. 2003.

ANEXO A/TABELAS E FIGURAS DA ETAPA DE ENGENHARIA REVERSA UTILIZANDO OS PADRÕES DA FAMÍLIA DE PADRÕES DE REENGENHARIA ORIENTADA A OBJETOS

CLUSTER 1: MODELAR OS DADOS DO LEGADO

PADRÃO 1: INICIAR A ANÁLISE DOS DADOS:

FIGURA 1: TABELA PROGRAMAS X ARQUIVOS

Programas (.prg)	Arquivos (.dbf)
Fimo.prg	
Imo.prg	
Imo1000.prg	
Imo1001.prg	Locador.dbf
Imo1002.prg	Locador.dbf
Imo1003.prg	Locador.dbf Imovel.dbf

	Locatario.dbf Fiador.dbf
Imo1004.prg	Imovel.dbf Tpim.dbf
Imo1301.prg	Fiador.dbf
Imo1302.prg	Ficha.dbf
Imo1401.prg	
Imo2000.prg	
Imo2001.prg	Locador.dbf Imovel.dbf Locatario.dbf Fiador.dbf
Imo2002.prg	Locador.dbf Imovel.dbf Locatario.dbf Fiador.dbf
Imo2003.prg	Locador.dbf Imovel.dbf Locatario.dbf Fiador.dbf
Imo2004.prg	Locatario.dbf
Imo2301.prg	Fiador.dbf
Imo2401.prg	Imovel.dbf
Imo3000.prg	
Imo3001.prg	Locador.dbf Locatario.dbf
Imo3002.prg	Imovel.dbf Tpim.dbf
Imo3003.prg	Imovel.dbf Tpim.dbf Locatario.dbf
Imo3004.prg	Locatario.dbf
Imo3005.prg	Locatario.dbf Imovel.dbf Tplt.dbf
Imo3006.prg	Locador.dbf Tpld.dbf
Imo3101.prg	Tplt.dbf
Imo4000.prg	
Imo4001.prg	Imovel.dbf Locatario.dbf
Imo4002.prg	
Imo4003.prg	Imovel.dbf Locatario.dbf .dbf
Imo4004.prg	
Imo4005.prg	Imovel.dbf Locador.dbf
Imo4006.prg	Imovel.dbf

	Locatario.dbf .dbf
Imo4007.prg	Locador.dbf Imovel.dbf Locatario.dbf
Imo4008.prg	
Imo4009.prg	Imovel.dbf Locatario.dbf .dbf
Imo4010.prg	Imovel.dbf Locatario.dbf
Imo4011.prg	Imovel.dbf Locatario.dbf
Imo4012.prg	Imovel.dbf Locatario.dbf
Imo4013.prg	Fiador.dbf Locatario.dbf
Imo4014.prg	
Imo4101.prg	Locador.dbf
Imo4111.prg	Pedido.dbf
Imo4112.prg	Pedido.dbf
Imo4113.prg	Pedido.dbf
Imo4141.prg	Locador.dbf Tplc.dbf
Imo4142.prg	Locatario.dbf Tplt.dbf
Imo4201.prg	Locador.dbf Fiador.dbf Imovel.dbf Locatario.dbf
Imo4202.prg	Locador.dbf Fiador.dbf Imovel.dbf Locatario.dbf
Imo4203.prg	Contrato.dbf
Imo4204.prg	Contrato.dbf
Imo4205.prg	Locador.dbf Fiador.dbf Imovel.dbf Locatario.dbf
Imo4211.prg	Contrato.dbf
Imo4222.prg	Contrato.dbf
Imo4255.prg	Contrato.dbf
Imo4401.prg	Locatario.dbf Locador.dbf Imovel.dbf Tplt.dbf
Imo4402.prg	Locador.dbf Imovel.dbf

	Locatario.dbf
Imo4801.prg	Imovel.dbf Locatario.dbf
Imo4811.prg	Cobranca.dbf
Imo5000.prg	
Imo5001.prg	Locador.dbf Imovel.dbf Locatario.dbf Fiador.dbf
Imo5002.prg	Contrato.dbf
Pimo.prg	

PADRÃO 2: DEFINIR CHAVES:

FIGURA 2 – ARQUIVOS DE DADOS COM SEUS RESPECTIVOS ARQUIVOS DE ÍNDICES:

ARQUIVOS (.DBF)	**ÍNDICES (.NTX)**
Arq_Stru	
Cobrança	
Contrato	
Fiador	Fdcd.ntx
Ficha	
Imovel	Imcd.ntx
Locador	Lccd.ntx Lcnm.ntx Imlc.ntx
Locatari	Ltcd.ntx Ltnm.ntx Ltlc.ntx
Pedido	
Tpim	
Tplc	
Tpld	
Tplt	Tpdia.ntx

.

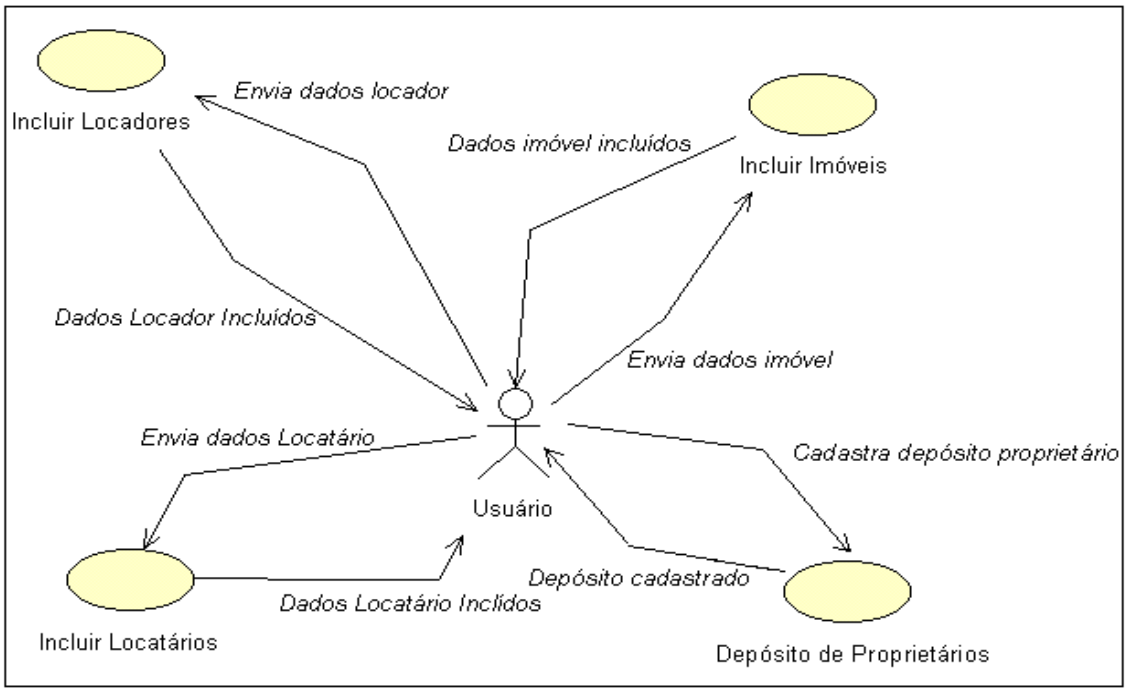
FIGURA 3- TABELA ENTIDADES X CHAVES

ENTIDADES	CHAVE PRIMÁRIA
Arq_Stru	————
Cobranca	————
Contrato	————
Fiador	Loc_Imo(chave de locatário)
Ficha	————
Imovel	Código(locador)+Código
Locador	Código
Locatari	Loc_Imo (chave de imóvel)
Pedido	————
Tpim	Código(locador)+Código
Tplc	Codigo(locador)
Tpld	Codigo(locador)
Tplt	Dia

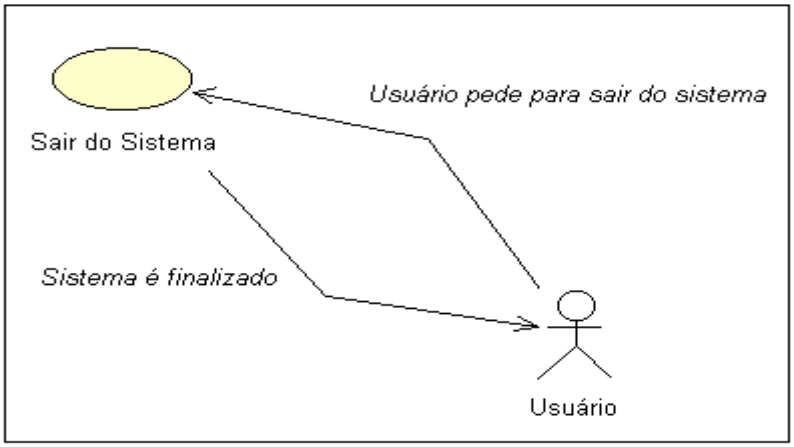
CLUSTER 2: MODELAR A FUNCIONALIDADE DO SISTEMA

PADRÃO 6: CONSTRUIR DIAGRAMAS DE USE CASES:

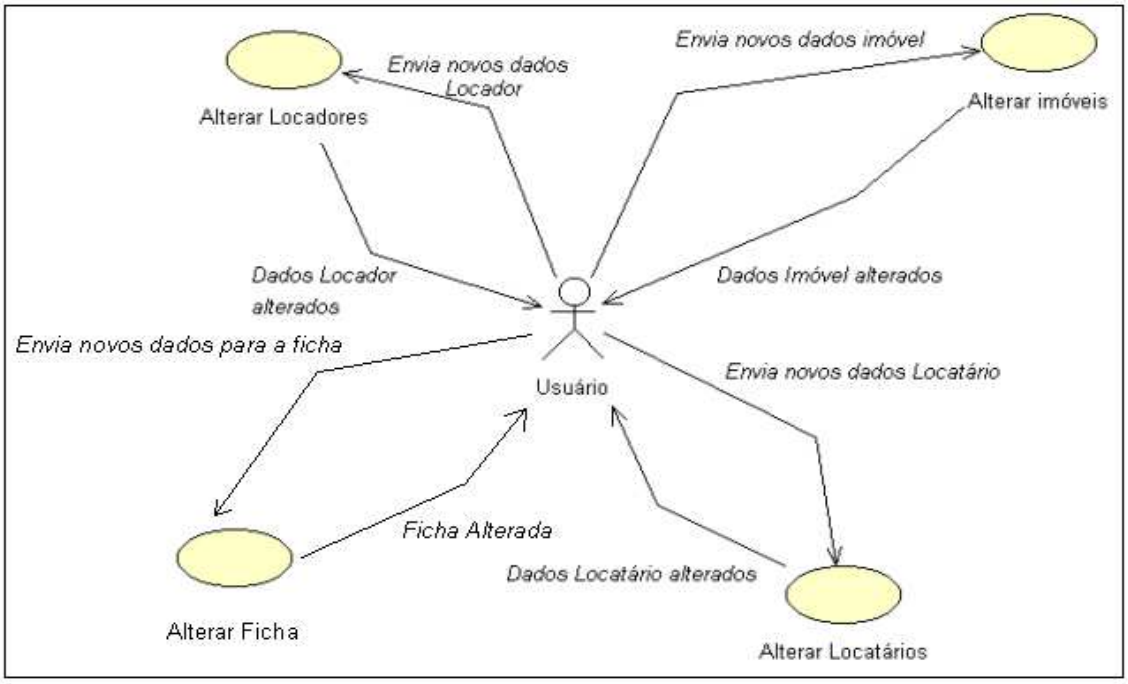
FIGURA 4 – DIAGRAMAS DE USE CASES PARA OS SEIS CENÁRIOS OBTIDOS A PARTIR DO ESTUDO DO PADRÃO 5:



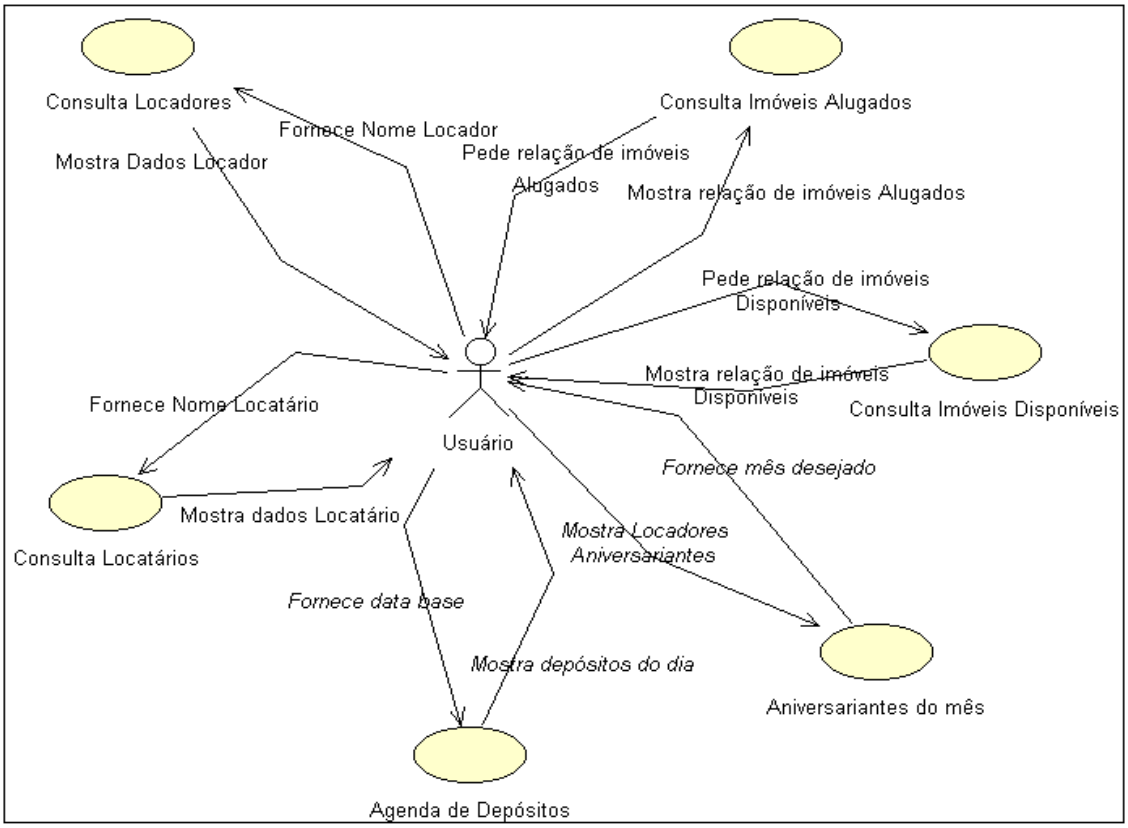
USE CASE DO CENÁRIO INCLUSÕES



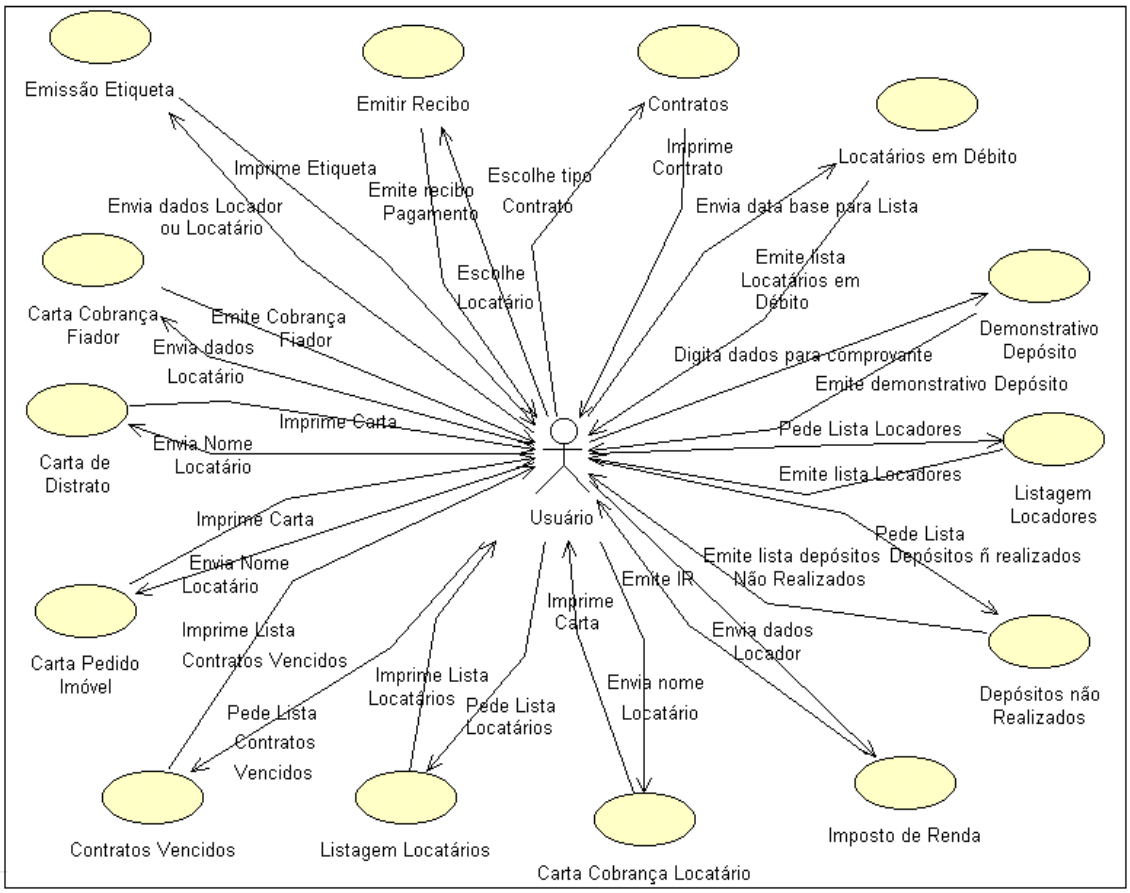
USE CASE DO CENÁRIO SAIR



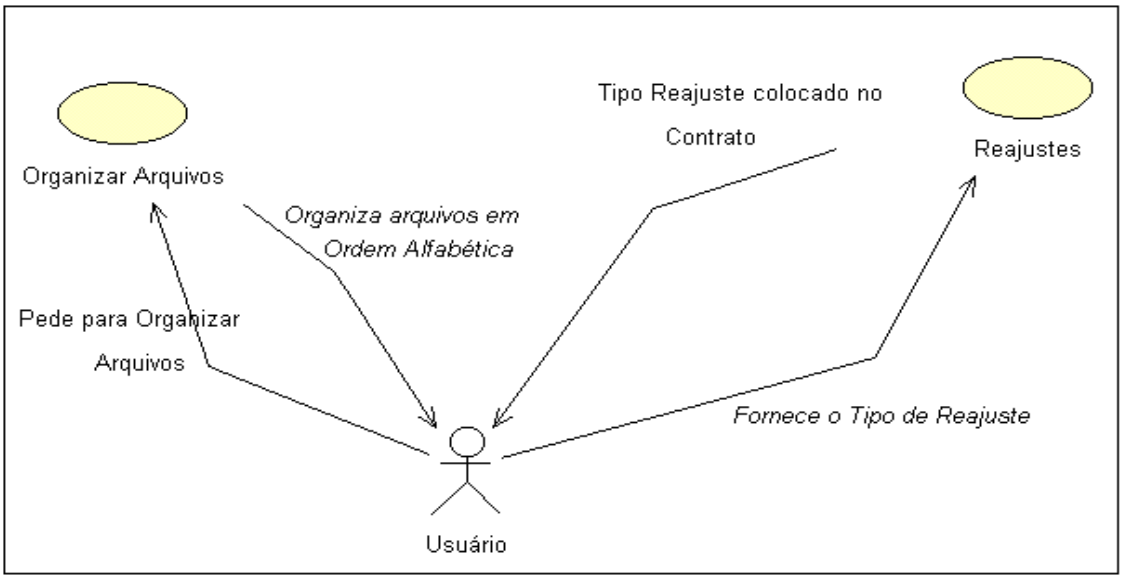
USE CASE DO CENÁRIO ALTERAÇÕES



USE CASE DO CENÁRIO CONSULTAS



USE CASE DO CENÁRIO RELATÓRIOS



USE CASE DO CENÁRIO CONTROLE

PADRÃO 7: ELABORAR A DESCRIÇÃO DE USE CASES:

FIGURA 5 – DESCRIÇÃO DOS DIAGRAMAS DE USE CASES:

Use Case: Incluir Locadores

Curso: Normal

1. Usuário solicita inclusão de um locador;
2. Sistema gera automaticamente um código autonumeração para o locador a ser incluído;
3. Usuário entra com os dados do locador;
4. Sistema solicita ao usuário gravar ou alterar os dados;
5. Usuário aceita gravar os dados;
6. Sistema grava os dados que foram registrados no arquivo Locador.dbf
7. Sistema mostra dicas de como gravar e abandonar a tela, e exibe espaço para usuário inserir observações inerentes ao locador.
8. Usuário termina a gravação dos dados, pressionando as teclas CTRL + W.
9. Sistema retorna à tela de inclusão de locadores, para uma nova inclusão.

Use Case: Incluir Locadores

Curso: Alternativo

1. Usuário solicita inclusão de um locador;
2. Sistema gera automaticamente um código autonumeração para o locador a ser incluído;
3. Usuário entra com os dados do locador;
4. Sistema solicita ao usuário, gravar ou alterar os dados que foram inseridos.
5. Usuário escolhe alterar dados;
6. O sistema volta na tela de inserção de dados, para usuário altera-los.
7. Sistema solicita gravação dos dados ao usuário
8. Usuário aceita gravar os dados;
9. Sistema grava os dados que foram registrados no arquivo Locador.dbf
10. Sistema mostra dicas de como gravar e abandonar a tela, e exibe espaço para usuário inserir observações inerentes ao locador.
11. Usuário termina a gravação dos dados, pressionando as teclas CTRL + W.
12. Sistema retorna à tela de inclusão de locadores, para uma nova inclusão.

Use Case: Incluir Imóveis

Curso: Normal

1. Usuário solicita inclusão de um imóvel;
2. Sistema solicita o código do locador, dono do imóvel a ser incluído.
3. Usuário informa o código do locador.
4. Sistema busca no arquivo Locador.dbf a confirmação do código informado pelo usuário.
    a. Se o código estiver incorreto, sistema informa ao usuário e volta para a tela de inclusão de imóveis.
5. Sistema busca no arquivo Locador.dbf o nome do locador e gera a segunda parte do código do imóvel que está sendo incluído, através da autonumeração; o código do imóvel então, será a concatenação do código do locador com a autonumeração que o próprio sistema gera automaticamente.
6. Usuário informa dados do imóvel a ser incluído;
7. Sistema solicita ao usuário gravar ou alterar os dados:
8. Usuário aceita gravar os dados;
9. Sistema grava os dados que foram registrados no arquivo Imóvel.dbf
10. Sistema retorna à tela de inclusões, para uma nova inclusão.

Use Case: Incluir Imóveis

Curso: Alternativo

1. Usuário solicita inclusão de um imóvel;
2. Sistema solicita o código do locador, dono do imóvel a ser incluído.
3. Usuário informa o código do locador.
4. Sistema busca no arquivo Locador.dbf a confirmação do código informado pelo usuário.
    a. Se o código estiver incorreto, sistema informa ao usuário e volta para a tela de inclusão de imóveis.
5. Se código do locador estiver correto, sistema busca no arquivo Locador.dbf o nome do locador e gera a segunda parte do código do imóvel que está sendo incluído, através da autonumeração; o código do imóvel então, será a concatenação do código do locador com a autonumeração que o próprio sistema gera automaticamente.
6. Usuário informa dados do imóvel a ser incluído;
7. Sistema solicita ao usuário gravar ou alterar os dados:
8. Usuário escolhe alterar dados;
9. O sistema volta na tela de inserção de dados, para usuário altera-los.
10. Sistema solicita gravação dos dados ao usuário
11. Usuário aceita gravar os dados;
12. Sistema grava os dados que foram registrados no arquivo Imóvel.dbf
13. Sistema retorna à tela de inclusão de locadores, para uma nova inclusão.

Use Case: Incluir Locatários
Curso: Alternativo

1. Usuário solicita inclusão de um locatário;
2. Sistema solicita o código do locador.
3. Usuário informa o código do locador.
4. Sistema busca no arquivo Locador.dbf a confirmação do código informado pelo usuário.
    a. Se o código estiver incorreto, sistema informa ao usuário e volta para a tela de inclusão de locatários.
5. Se código do locador estiver correto, sistema busca no arquivo Locador.dbf o nome do locador e pede o número de código do imóvel a ser alugado.
6. Usuário informa código do imóvel a ser alugado;
    a. Se código estiver incorreto, sistema informa ao usuário e volta para a tela de inclusão de locatários.
7. Usuário informa dados do locatário;
8. Sistema abre a tela de inserção de dados de 02 fiadores;
9. Usuário insere dados dos 02 fiadores;
10. Sistema solicita gravar ou alterar dos dados;
11. Usuário escolhe alterar dados;
12. O sistema volta na tela de inserção de dados, para usuário altera-los.
13. Sistema solicita gravação dos dados ao usuário
14. Usuário aceita gravar os dados;
15. Sistema grava os dados que foram registrados no arquivo Locatario.dbf, sendo que a chave de Locatario.dbf é a concatenação da chave de Locador.dbf + Imovel.dbf; Ainda com a mesma chave de locatario.dbf, LOC_IMO, o Sistema grava os dados dos 02 fiadores no arquivo Fiador.dbf.
16. Sistema retorna à tela de inclusão de locadores, para uma nova inclusão.

Use Case: Incluir Locatários
Curso: Normal

1. Usuário solicita inclusão de um locatário;
2. Sistema solicita o código do locador.
3. Usuário informa o código do locador.
4. Sistema busca no arquivo Locador.dbf a confirmação do código informado pelo usuário.
    a. Se o código estiver incorreto, sistema informa ao usuário e volta para a tela de inclusão de locatários.
5. Se código do locador estiver correto, sistema busca no arquivo Locador.dbf o nome do locador e pede o número de código do imóvel a ser alugado.
6. Usuário informa código do imóvel a ser alugado;
    a. Se código estiver incorreto, sistema informa ao usuário e volta para a tela de inclusão de locatários.
7. Usuário informa dados do locatário;
8. Sistema abre a tela de inserção de dados de 02 fiadores.
9. Usuário insere dados dos 02 fiadores;
10. Sistema solicita gravar ou alterar os dados ao usuário
11. Usuário aceita gravar os dados;
12. Sistema grava os dados que foram registrados no arquivo Locatario.dbf, sendo que a chave de Locatario.dbf é a concatenação da chave de Locador.dbf + Imovel.dbf; Ainda com a mesma chave de locatario.dbf, LOC_IMO, o Sistema grava os dados dos 02 fiadores no arquivo Fiador.dbf.
13. Sistema retorna à tela de inclusão de locadores, para uma nova inclusão.

Use Case: Depósito de Proprietários

Curso: Normal

1. Usuário solicita baixa no depósito realizado para um locador;
2. Sistema exibe uma tela com o nome o Locador e o tipo do imóvel que este possui a partir do arquivo Imovel.dbf.
3. Usuário pode localizar um Locador desejado utilizando a tecla F8.
4. Sistema pede o nome do Locador.
5. Usuário informa o nome do Locador.
6. Sistema posiciona cursor no nome digitado ou no nome que mais se aproxima do digitado pelo usuário, a partir do arquivo Imovel.dbf.
7. Usuário escolhe Locador desejado pressionando ENTER na opção desejada.
8. Sistema abre uma tela, a partir do arquivo Ficha.dbf, contendo dados do pagamento do aluguel, data do pagamento e um campo para a baixa do depósito. Todos os dados podem ser editados.
9. Usuário edita os dados.
10. Sistema grava os dados no arquivo Ficha.dbf e retorna ao menu inclusões.

Use Case: Alterar Locadores

Curso: Normal

1. Usuário solicita alteração de um locador.
2. Sistema pede o código do locador a ter seu cadastro alterado;
3. Usuário entra com o código do locador;
    a. Se o código estiver errado, Sistema avisa usuário e retorna à tela alterações locadores.
4. Sistema exibe dados do locador, a partir do arquivo Locador.dbf, e pergunta se usuário quer alterar ou excluir os dados;
5. Usuário altera os dados;
6. Sistema regrava os dados no arquivo Locador.dbf;
7. Sair da tela de edição de dados, Sistema retorna ao menu alterações.

Use Case: Alterar Locadores

Curso: Alternativo

1. Usuário solicita alteração de um locador.
2. Sistema pede o código do locador a ter seu cadastro alterado;
3. Usuário entra com o código do locador;
    a. Se o código estiver errado, Sistema avisa usuário e retorna à tela alterações locadores.
4. Sistema exibe dados do locador, a partir do arquivo Locador.dbf e pergunta se usuário quer alterar ou excluir os dados;
5. Usuário escolhe excluir o registro.
6. Sistema exibe mensagem dizendo que irá excluir também imóveis, locatários e fichas referentes a esse Locador.
7. Usuário aceita. Sistema exclui todos os registros referentes ao Locador, nos arquivos Locador.dbf, Imovel.dbf, Locatario.dbf e Fiador.dbf.
8. Sistema retorna as novas alterações

Use Case: Alterar Imóveis

Curso: Normal

1. Usuário solicita alteração de um imóvel;
2. Sistema pede o código do locador, dono do imóvel;
3. Usuário entra com o código do locador;
4. Sistema verifica o código no arquivo Locador.dbf.
    a. Se o código estiver errado, Sistema avisa usuário e retorna à tela de edição do imóvel.
5. Sistema exibe o nome do locador e pede o código do imóvel.
6. Sistema verifica no arquivo Imovel.dbf.
    a. Se o código do imóvel estiver errado, Sistema avisa e retorna à tela de edição de imóvel.
7. Sistema exibe dados do imóvel e pergunta se usuário quer alterar ou excluir os dados;
8. Usuário altera os dados;
9. Sistema regrava os dados no arquivo Imovel.dbf;
10. Sai da tela de edição de dados. Sistema retorna as novas alterações

Use Case: Alterar Imóveis

Curso: Normal

1. Usuário solicita alteração de um imóvel;
2. Sistema pede o código do locador, dono do imóvel;
3. Usuário entra com o código do locador;
4. Sistema verifica o código no arquivo Locador.dbf.
    a. Se o código estiver errado, Sistema avisa usuário e retorna à tela de edição do imóvel.
5. Sistema exibe o nome do locador e pede o código do imóvel.
6. Usuário entra com o código do Imóvel.
7. Sistema verifica no arquivo Imovel.dbf.
    a. Se o código do imóvel estiver errado, Sistema avisa e retorna à tela de edição de imóvel.
8. Sistema exibe dados do imóvel e pergunta se usuário quer alterar ou excluir os dados;
9. Usuário escolhe excluir o registro;
10. Sistema exclui registro do arquivo Imovel.dbf, e registros referentes nos arquivos Locatario.dbf e Fiador.dbf.

Use Case: Alterar Locatário

Curso: Normal

1. Usuário solicita alteração de um locatário;
2. Sistema solicita o código do locatário.
3. Usuário informa o código do locatário.
4. Sistema busca no arquivo Locatario.dbf a confirmação do código informado pelo usuário.
    a. Se o código estiver incorreto, sistema informa ao usuário e volta para a tela de inclusão de locatários.
5. Sistema busca no arquivo Locatario.dbf os dados do locatário e pergunta se usuário deseja alterar ou excluir dados;
6. Usuário escolhe alterar dados;
7. Sistema abre a tela de edição de dados;
8. Usuário altera os dados do locatário e do fiador;
9. Sistema regrava os dados nos arquivos Locatario.dbf e Fiador.dbf;
10. Sistema retorna ao menu alterações.

Use Case: Alterar Locatário

Curso: Alternativo

1. Usuário solicita alteração de um locatário;
2. Sistema solicita o código do locatário.
3. Usuário informa o código do locatário.
4. Sistema busca no arquivo Locatario.dbf a confirmação do código informado pelo usuário.
    a. Se o código estiver incorreto, sistema informa ao usuário e volta para a tela de inclusão de locatários.
5. Sistema busca no arquivo Locatario.dbf os dados do locatário e pergunta se usuário deseja alterar ou excluir os dados;
6. Usuário escolhe excluir o registro;
7. Sistema exclui os registros dos arquivos Locatário.dbf e Fiador.dbf, a partir da chave primária LOC_IMO do registro escolhido.
8. Sistema retorna ao menu alterações

Use Case: Alterar Ficha

Curso: Normal

1. Usuário solicita alterar a ficha de um locatário;
2. Sistema exibe uma tela com o nome de todos locatários e o tipo do imóvel alugado a partir do arquivo Locatario.dbf.
3. Usuário pode localizar um Locatário desejado utilizando a tecla F8.
4. Sistema pede o nome do Locatário procurado.
5. Usuário informa o nome do Locatário.
6. Sistema posiciona cursor no nome digitado ou no nome que mais se aproxima do digitado pelo usuário, a partir do arquivo Locatario.dbf.
7. Usuário escolhe Locatário desejado pressionando ENTER na opção desejada.
8. Sistema abre uma tela, a partir do arquivo Ficha.dbf, contendo todo o histórico do pagamento do aluguel do Locatário.
9. Usuário pode escolher entre apagar, imprimir e inserir novos dados.
10. Se usuário escolher apagar ou inserir, poderá editar o registro do arquivo Ficha.dbf que necessita, e o sistema gravará os novos dados no arquivo.

Use Case: Alterar Ficha

Curso: Alternativo

1. Usuário solicita alterar a ficha de um locatário;
2. Sistema exibe uma tela com o nome de todos locatários e o tipo do imóvel alugado a partir do arquivo Locatario.dbf.
3. Usuário pode localizar um Locatário desejado utilizando a tecla F8.
4. Sistema pede o nome do Locatário procurado.
5. Usuário informa o nome do Locatário.
6. Sistema posiciona cursor no nome digitado ou no nome que mais se aproxima do digitado pelo usuário, a partir do arquivo Locatario.dbf.
7. Usuário escolhe Locatário desejado pressionando ENTER na opção desejada.
8. Sistema abre uma tela, a partir do arquivo Ficha.dbf, contendo todo o histórico do pagamento do aluguel do Locatário.
9. Usuário pode escolher entre apagar, imprimir e inserir novos dados.
10. Se usuário escolher imprimir (pressionando F5), sistema imprime todos os registros do arquivo Ficha.dbf , como num histórico de pagamentos do Locatário.

Use Case: Consulta Locadores

Curso: Normal

1. Usuário solicita consulta de locadores;
2. Sistema abre uma tela com dados de todos os locadores ordenados pelo seu código, a partir do arquivo Locador.dbf.
3. Usuário pode localizar um Locador na lista, pressionando F8.
4. Sistema pede nome do Locador.
5. Usuário informa nome do Locador.
6. Sistema posiciona cursor no nome digitado ou no nome que mais se aproxima do digitado pelo usuário
7. Usuário consulta dados do Locador escolhido, e se quiser detalhar dados, pode faze-lo pressionando ENTER.
8. Usuário pressiona ENTER.
9. Sistema exibe tela com alguns dados do Locador, e a partir do arquivo Locatario.dbf, exibe todos os locatários referentes aquele locador.
10. Usuário aperta ESC e retorna ao menu de consultas.

Use Case: Imóveis Alugados

Curso: Normal

1. Usuário solicita consulta de imóveis alugados;
2. Sistema abre uma tela com dados de todos os disponíveis a partir do arquivo Imovel.dbf, organizados pelo seu código.
3. Usuário aperta ESC e retorna ao menu de consultas.

Use Case: Imóveis Disponíveis

Curso: Normal

1. Usuário solicita consulta de imóveis disponíveis;
2. Sistema abre uma tela com dados de todos os imóveis alugados a partir do arquivo Imovel.dbf, organizados pelo seu código.
3. Usuário aperta ESC e retorna ao menu de consultas.

Use Case: Agenda de depósitos

Curso: Normal

1. Usuário solicita ver a agenda de depósitos;
2. Sistema solicita data base para listagem.
3. Usuário digita data base.
4. Sistema abre uma tela com os depósitos devidos para a data digitada comparado com a variável dia do arquivo Locatario.dbf (dia do depósito do proprietário), e utiliza o arquivo Imovel.dbf para calcular o valor devido a ser depositado para o locador (arquivo Locador.dbf).
5. Usuário pressiona qualquer tecla para retornar ao menu consultas.
6. Sistema retorna ao menu consultas

Use Case: Consulta Locatários

Curso: Normal

1. Usuário solicita consulta de locatários;
2. Sistema abre uma tela com dados de todos os locatários ordenados pelo seu código, a partir do arquivo Locatario.dbf.
3. Usuário pode localizar um Locatário na lista, pressionando F8.
4. Sistema pede nome do Locatário.
5. Usuário informa nome do Locatário.
6. Sistema posiciona cursor no nome digitado ou no nome que mais se aproxima do digitado pelo usuário.
7. Usuário consulta dados do Locatário escolhido.
8. Usuário aperta ESC e retorna ao menu de consultas.

2. Sistema solicita mês base para listagem.
3. Usuário digita mês desejado.
4. Sistema abre uma tela com todos os dados dos LOCADORES aniversariantes do mês digitado pelo usuário a partir da comparação com a variável DTNASC do arquivo Locador.dbf .

Use Case: Consulta Locatários

Curso: Alternativo

1. Usuário solicita emitir recibo de pagamento de aluguel;
2. Sistema abre tela com todos os locatários e o imóvel alugado por eles a partir dos arquivos Locatario.dbf e Imovel.dbf;
3. Usuário pode localizar um Locatário desejado utilizando a tecla F8.
4. Sistema pede o nome do Locatário.
5. Usuário informa o nome do Locatário.
6. Sistema posiciona cursor no nome digitado ou no nome que mais se aproxima do digitado pelo usuário, a partir do arquivo Locatario.dbf.
7. Usuário escolhe Locatário desejado pressionando ENTER na opção desejada.
8. Sistema abre uma tela com alguns dados do locatário e a partir do arquivo Ficha.dbf exibe dados do histórico de pagamentos do Locatário.
9. Usuário pode edita-los e ao final imprimir um recibo em 03 vias do pagamento de tributos referente ao mês desejado. Além disso, o usuário pode ainda colocar um aviso no rodapé do recibo e este será impresso no final de cada uma das 03 vias. Sistema grava os dados no arquivo Ficha.dbf.

Use Case: Demonstrativo de Depósito

Curso: Normal

1. Usuário solicita um demonstrativo de depósito;
2. Sistema abre um submenu perguntando se usuário deseja o demonstrativo, individualmente ou globalmente.
3. Usuário escolhe o individualmente.
4. Sistema solicita o mês/ano referente ao repasse, o código do Locador para o qual será impresso o demonstrativo e exibe um memo para o usuário digitar quaisquer observações que julgar necessárias.
5. Usuário digita dados solicitados e no fim pressiona CTRL + W.
6. Sistema confirma se usuário deseja imprimir relatório.
    a. Se usuário escolher não, Sistema retorna ao menu relatórios.
7. Usuário escolhe imprimir.
8. Sistema procura o mês digitado no arquivo Ficha.dbf, e busca dados nos arquivos Locador.dbf, Locatário.dbf e Imovel.dbf, para confeccionar o relatório.
9. Sistema retorna ao menu relatórios

Use Case: Locatários em Débito

Curso: Normal

1. Usuário solicita consulta de locatários em débito;
2. Sistema solicita uma data base para listagem;
3. Usuário digita data base.
4. Sistema pergunta se usuário deseja confirmar impressão.
    a. Se usuário escolher não, Sistema retorna ao menu relatórios.
5. Usuário escolhe imprimir.
6. Sistema abre o arquivo Imovel.dbf, para saber quais registros estão atrasados, e através da chave primária de Imovel.dbf, localiza o registro procurado no arquivo Locatario.dbf. A partir dessas informações o Sistema imprime uma listagem de todos os locatários em débito.
7. Sistema retorna ao menu relatórios.

Use Case: Demonstrativo de Depósito

Curso: Alternativo

1. Usuário solicita um demonstrativo de depósito;
2. Sistema abre um submenu perguntando se usuário deseja o demonstrativo, individualmente ou globalmente.
3. Usuário escolhe o globalmente.
4. Sistema solicita o mês/ano referente ao repasse.
5. Usuário digita o mês solicitado.
6. Sistema confirma se usuário deseja imprimir o relatório.
    a. Se usuário escolher não sistema, retorna ao menu relatórios.
10. Usuário escolhe imprimir;
11. Sistema procura o mês digitado no arquivo Ficha.dbf, e busca dados nos arquivos Locador.dbf, Locatário.dbf e Imovel.dbf, para confeccionar o relatório de todos os locadores.

Use Case: Contratos

Curso: Normal

1. Usuário solicita contrato de locação;
2. Sistema abre submenu com as seguintes opções:
   2.1. Contrato Imóvel (1);
   2.2. Contrato Telefone;
   2.3. Comunicado Imóvel;
   2.4. Comunicado Telefone;
   2.5. Contrato Imóvel (2);

   Todos os contratos e comunicados de Imóvel e Telefone são digitados pelo próprio usuário em estruturas do tipo memo, e mais tarde essas estruturas podem ser alteradas de acordo com a necessidade do mesmo.
3. Usuário escolhe Contratos.
4. Sistema abre os arquivos Locador.dbf, Locatario.dbf, Fiador.dbf e Imovel.dbf, e uma tela com o nome de todos os locatários, seus locadores e outros dados.
5. Usuário pode localizar um Locatário desejado utilizando a tecla F8.
6. Sistema pede o nome do Locatário.
7. Usuário informa o nome do Locatário.
8. Sistema posiciona cursor no nome digitado ou no nome que mais se aproxima do digitado pelo usuário.
9. Usuário escolhe Locatário desejado pressionando ENTER na opção desejada.
10. Sistema abre a tela do contrato com alguns dados do locatário, e a partir do arquivo Contrato.dbf exibe o memo referente à opção escolhida no submenu do item 4.
11. Usuário edita o contrato e dá o comando CTRL+W para gravar e imprimir o contrato, se não quiser imprimi-lo, é só apertar ESC e o sistema retornará ao menu de contratos.
12. Se o usuário digitar CTRL+W o sistema grava as alterações e abre uma janela perguntando quantas cópias do contrato serão impressas.
13. Usuário digita o número de cópias.
14. Sistema confirma se usuário quer imprimir contrato.
    14.1. Se usuário escolher não, Sistema não imprime contrato.
15. Usuário escolhe imprimir;
16. Sistema imprime contrato, e pergunta se usuário quer imprimir comunicado de locação de imóvel.
    16.1. Se usuário escolher não, Sistema não imprime comunicado.
17. Usuário escolhe sim.
18. Sistema imprime Comunicado e volta ao menu de relatórios.

Use Case: Contratos

Curso: Alternativo

1. Usuário solicita contrato de locação;
2. Sistema abre submenu com as seguintes opções:
   2.1. Contrato Imóvel (1);
   2.2. Contrato Telefone;
   2.3. Comunicado Imóvel;
   2.4. Comunicado Telefone;
   2.5. Contrato Imóvel (2);

   Todos os contratos e comunicados de Imóvel e Telefone são digitados pelo próprio usuário em estruturas do tipo memo, e mais tarde essas estruturas podem ser alteradas de acordo com a necessidade do mesmo.
3. Usuário escolhe as opções de comunicado;
4. Sistema abre a tela de edição de comunicados, através do arquivo Contrato.dbf.
5. Usuário edita comunicado e aperta CTRL+W para gravar e imprimir o Comunicado.
6. Sistema pergunta quantas cópias deseja e usuário.
7. Usuário digita o número de cópias.
8. Sistema imprime Comunicado e retorna à tela relatórios

Use Case: Listagem dos Locadores

Curso: Normal

1. Usuário solicita a lista de todos os locadores;
2. Sistema solicita uma confirmação da impressão.
   a. Se Usuário escolher não, Sistema volta ao menu relatórios.
3. Se Usuário escolher sim, Sistema abre os arquivos Locador.dbf e Imovel.dbf, imprime dados do locador e a comissão que cada um paga para a Imobiliária.
4. Sistema retorna ao menu relatórios

Use Case: Listagem dos Depósitos não Realizados

Curso: Normal

1. Usuário solicita uma listagem dos depósitos não realizados;
2. Sistema pergunta se usuário deseja confirmar a impressão;
    a. Se usuário escolher não, Sistema retorna ao menu relatórios;
3. Se usuário escolher sim, Sistema, abre os arquivos Ficha.dbf, para saber se Locatário pagou alugueis, e a partir dessa informação, coloca no relatório o nome do Locador, (arquivo Locador.dbf), o nome do Locatário (arquivo Locatario.dbf), o valor do depósito devido, a partir do valor do aluguel – valor da comissão + débitos/créditos, e a data que ele deveria ter sido efetuado.
4. Sistema imprime um relatório com todos os dados descritos acima.
5. Sistema retorna ao menu relatórios.

Use Case: Imposto de Renda

Curso: Normal

1. Usuário solicita impressão do comprovante de imposto de renda;
2. Sistema abre uma tela com o nome do Locador, nome do Inquilino e descrição do imóvel, a partir dos arquivos, Locador.dbf, Locatario.dbf e Imovel.dbf;
3. Usuário pode encontrar o nome procurado pressionando a tecla F8, onde o sistema abrirá uma janela que o usuário digita o nome desejado e o sistema se posiciona no nome que mais se aproxima;
4. Se usuário encontrar o nome procurado, pressiona F5, para prosseguir a operação;
5. Sistema abre uma janela pedindo ao usuário o ano base do demonstrativo e o número de cópias que deseja imprimir;
6. Usuário entra com os dados necessários.
7. Sistema pergunta se usuário deseja confirmar a impressão;
    a. Se Usuário escolher não, Sistema retorna ao menu relatórios.
8. Usuário escolhe imprimir;
9. Sistema imprime um relatório com os dados, nome do Proprietário, nome do Locatário, imóvel alugado, endereço do imóvel, valor do aluguel, valor recebido pelo proprietário, e outros dados complementares.
10. Sistema retorna ao menu relatórios.

Use Case: Carta Cobrança Locatário

Curso: Normal

1. Usuário solicita impressão de uma carta de cobrança de Locatário;
2. Sistema abre um submenu para usuário escolher entre 3 tipos de carta de cobrança;
3. Usuário escolhe o número da carta e o Sistema, acessa o registro no arquivo Cobrança.dbf;
4. Independente do número da carta que o Usuário escolher, Sistema abrirá uma tela com o nome do Locatário e a descrição do imóvel locado, a partir dos arquivos Locatario.dbf e Imovel.dbf;
5. Usuário pode encontrar o nome procurado pressionando a tecla F8, onde o sistema abrirá uma janela que o usuário digita o nome desejado e o sistema se posiciona no nome que mais se aproxima;
6. Se usuário encontrar o nome procurado, pressiona F5, para prosseguir a operação;
7. Sistema abre uma tela com o texto da carta escolhida no submenu, com o nome do Locador, nome do Locatário e endereço do imóvel alugado;
8. Usuário pode editar a carta exibida na tela;
9. Usuário aperta CTRL+W para gravar as alterações no arquivo Cobrança.dbf;
10. Sistema pede que o Usuário entre com o número de cópias a ser impressa;
11. Usuário digita o número de cópias;
12. Sistema pergunta se usuário deseja confirmar a impressão
    a. Se Usuário escolher não, Sistema retorna ao menu relatórios.
13. Usuário escolhe imprimir.
14. Sistema imprime a carta de cobrança.

Use Case: Contratos Vencidos

Curso: Normal

1. Usuário solicita a lista de todos os contratos vencidos;
2. Sistema solicita uma confirmação da impressão.
    a. Se Usuário escolher não, Sistema volta ao menu relatórios.
3. Usuário escolhe sim.
4. Sistema abre os arquivos Locatario.dbf e Imovel.dbf, compara se a data2 (data final do contrato) é menor que a data do sistema que o computador está marcando no dia, e imprime o nome do Locador, tipo do imóvel, nome do locatário e valor do aluguel.
5. Sistema retorna ao menu relatórios

a. Se Usuário escolher não, Sistema volta ao menu relatórios.
3. Usuário escolhe sim.
4. Sistema abre os arquivos Locatario.dbf e Imovel.dbf, imprime dados dos Locatários, valor do aluguel e valor comissão que a Imobiliária recebe.

Use Case: Carta Pedido de Imóvel

Curso: Normal

1. Usuário solicita impressão de uma carta de pedido de imóvel;
2. Sistema abre uma tela com o nome do Locatário e a descrição do imóvel locado, a partir dos arquivos Locatario.dbf e Imovel.dbf;
3. Usuário pode encontrar o nome procurado pressionando a tecla F8, onde o sistema abrirá uma janela que o usuário digita o nome desejado e o sistema se posiciona no nome que mais se aproxima;
4. Se usuário encontrar o nome procurado, pressiona F5, para prosseguir a operação;
5. Sistema pede que o Usuário entre com o número de cópias a ser impressa;
6. Usuário digita o número de cópias;
7. Sistema abre uma tela com o texto da carta, a partir do registro Carta1 do arquivo Pedido.dbf, exibindo ainda o nome do Locador, nome do Locatário e endereço do imóvel;
8. Usuário pode editar a carta exibida na tela;
9. Usuário aperta CTRL+W para gravar as alterações;
10. Sistema pergunta se usuário deseja confirmar a impressão.
    a. Se Usuário escolher não, Sistema retorna ao menu relatórios.
11. Usuário escolhe sim.
12. Sistema imprime a carta de pedido de imóvel

Use Case: Carta de Distrato

Curso: Normal

1. Usuário solicita impressão de uma carta de distrato;
2. Sistema abre uma tela com o nome do Locatário e a descrição do imóvel locado, a partir dos arquivos Locatario.dbf e Imovel.dbf;
3. Usuário pode encontrar o nome procurado pressionando a tecla F8, onde o sistema abrirá uma janela que o usuário digita o nome desejado e o sistema se posiciona no nome que mais se aproxima;
4. Se usuário encontrar o nome procurado, pressiona F5, para prosseguir a operação;
5. Sistema pede que o Usuário entre com o número de cópias a ser impressa;
6. Usuário digita o número de cópias;
7. Sistema abre uma tela com o texto da carta, a partir do registro Carta2, também do arquivo Pedido.dbf, exibindo ainda o nome do Locador, nome do Locatário e endereço do imóvel alugado;
8. Usuário pode editar a carta exibida na tela;
9. Usuário aperta CTRL+W para gravar as alterações;
10. Sistema pergunta se usuário deseja confirmar a impressão
    a. Se Usuário escolher não, Sistema retorna ao menu relatórios.
11. Usuário escolhe sim.
12. Sistema imprime a carta de pedido de imóvel.
13. Sistema retorna ao menu relatórios

Use Case: Carta Cobrança Fiador

Curso: Normal

1. Usuário solicita impressão de uma carta de cobrança de Fiador;
2. Sistema abre uma tela com o nome do Locatário e a descrição do imóvel locado, a partir dos arquivos Locatario.dbf e Fiador.dbf; utiliza o código do locatário para encontrar o fiador desejado;
3. Usuário pode encontrar o nome procurado pressionando a tecla F8, onde o sistema abrirá uma janela que o usuário digita o nome desejado e o sistema se posiciona no nome que mais se aproxima;
4. Se usuário encontrar o nome procurado, pressiona F5, para prosseguir a operação;
5. Sistema abre uma janela e pede que o Usuário entre com o número de cópias a ser impressa;
6. Usuário digita o número de cópias;
7. Sistema abre uma nova janela para que o usuário escolha se a carta irá para o primeiro ou segundo fiador.
8. Usuário digita para qual fiador será emitida a carta de cobrança.
9. Sistema pergunta se usuário deseja confirmar a impressão
    a. Se Usuário escolher não, Sistema retorna ao menu relatórios.
10. Usuário escolhe sim.
11. Sistema imprime a carta de cobrança, a partir do registro Carta3, do arquivo Pedido.dbf, exibindo nome e endereço completo do fiador.
12. Sistema retorna ao menu relatórios.

Use Case: Emissão Etiqueta

Curso: Normal

1. Usuário solicita impressão de etiquetas com endereços cadastrados no sistema;
2. Sistema abre um submenu perguntando se usuário deseja imprimir endereço de proprietários ou de inquilinos.
3. Usuário escolhe proprietários.
4. Sistema abre uma tela com dois campos a serem preenchidos, cada um com 4 caracteres: Código do primeiro Locador a ser impresso e código do último locador. A partir desses dois limites, o Sistema procura no arquivo Locador.dbf se os códigos existem e imprime, nome e endereço dos locadores.
5. Sistema confirma se usuário deseja imprimir etiquetas.
    a. Se usuário escolher não, Sistema retorna ao menu relatórios.
6. Usuário escolhe sim.
7. Sistema imprime as etiquetas selecionadas.
8. Sistema retorna ao menu relatórios

Use Case: Emissão Etiqueta

Curso: Alternativo

1. Usuário solicita impressão de etiquetas com endereços cadastrados no sistema;
2. Sistema abre um submenu perguntando se usuário deseja imprimir endereço de proprietários ou de inquilinos.
3. Usuário escolhe inquilinos.
4. Sistema abre uma tela também com dois campos a serem preenchidos, só que dessa vez, cada campo possui 8 caracteres: Código do primeiro Locatário a ser impresso e código do último locatário. A partir desses dois limites, o Sistema procura no arquivo Locatario.dbf se os códigos existem e imprime, nome e endereço dos locatários.
5. Sistema confirma se usuário deseja imprimir etiquetas.
    a. Se usuário escolher não, Sistema retorna ao menu relatórios.
6. usuário escolhe sim.
7. Sistema imprime as etiquetas selecionadas.
8. Sistema retorna ao menu relatórios

Use Case: Organizar Arquivos
Curso: Normal

1. Usuário solicita organizar os arquivos de dados do sistema;
2. Sistema exibe uma mensagem dizendo que para realizar a operação necessita de exclusividade dos arquivos, e pede para usuário certificar-se disto antes. Sistema ainda pergunta se usuário deseja confirmar a atividade.
    a. Se Usuário escolher não, Sistema retorna ao menu Controle.
3. Usuário escolhe sim.
4. Sistema utiliza os arquivos Locador.dbf, Imovel.dbf, Locatario.dbf e Fiador.dbf, cada um exclusivamente, e indexa os arquivos aos seus respectivos arquivos de índice:
    a. Para Locador.dbf, indexa pelo código o arquivo Lccd.ntx e pelo nome o arquivo Lcnm.ntx.
    b. Para Imovel.dbf, indexa pelo código o arquivo Imcd.ntx.
    c. Para Locatario.dbf, indexa pelo código (LOC_IMO) o arquivo Ltcd.ntx, e pelo nome o arquivo Ltnm.ntx
    d. Para Fiador.dbf, indexa pelo código (LOC_IMO) o arquivo Fdcd.ntx.
5. Sistema retorna ao menu Controle

Use Case: Tipo de Reajuste
Curso: Normal

1. Usuário solicita modificar o tipo de reajuste adotado na imobiliária;
2. Sistema exibe uma tela para que o usuário entre com o nome do novo reajuste a ser usado.
3. Usuário entra com o nome do reajuste.
4. Sistema modifica apenas o valor da variável TIPO no arquivo Contrato.dbf, para o novo valor digitado pelo Usuário.
5. Sistema retorna ao menu Controle

Use Case: Sair do Sistema
Curso: Normal

1. Usuário solicita o menu saída;
2. Sistema é encerrado corretamente.

PADRÃO 13: CONSTRUIR DIAGRAMAS DE SEQUÊNCIA:

FIGURA 6 – DIAGRAMA DE SEQUÊNCIA DOS USE CASES:

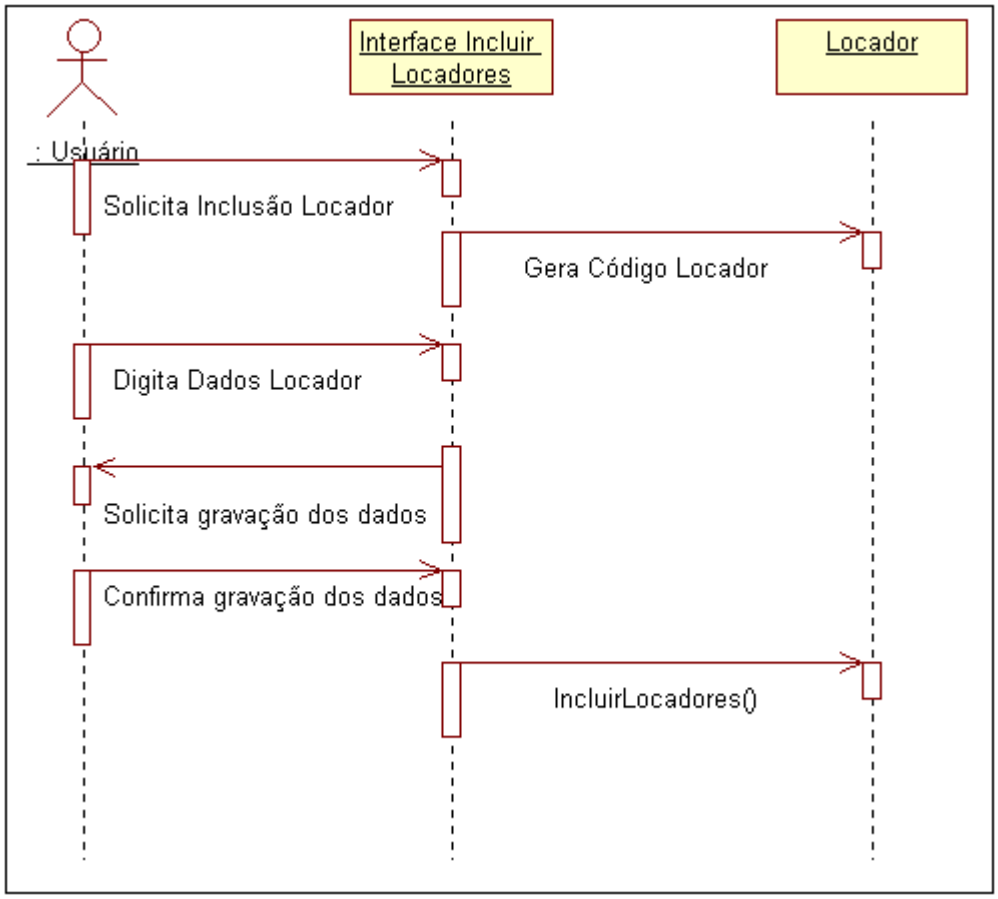


DIAGRAMA DE SEQUÊNCIA DO USE CASE INCLUIR LOCADORES, CURSO NORMAL

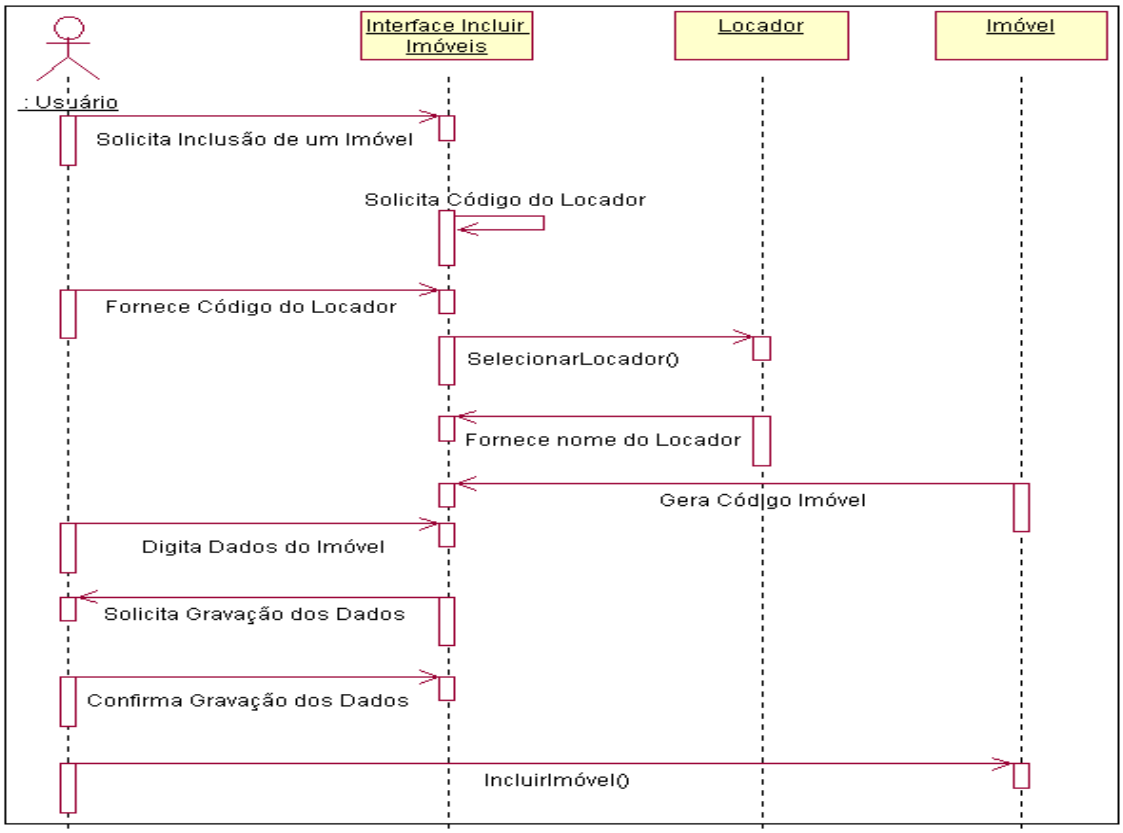


DIAGRAMA DE SEQUÊNCIA DO USE CASE INCLUIR IMÓVEIS, CURSO NORMAL

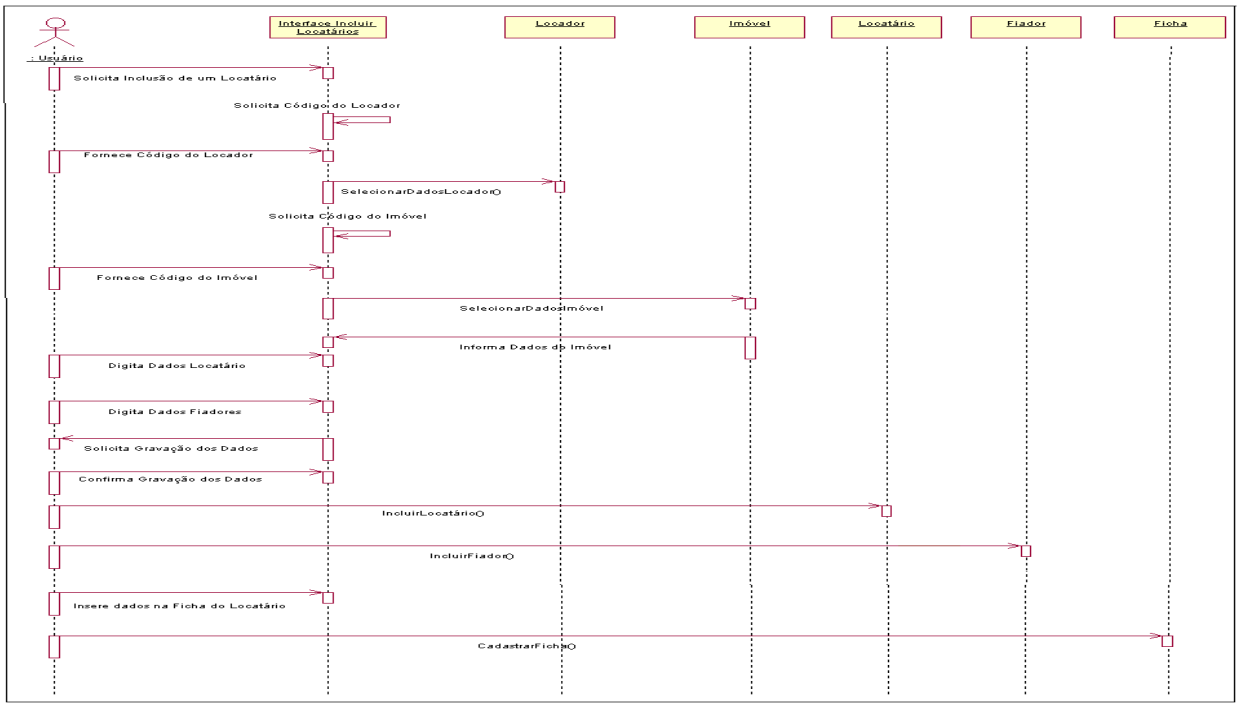


DIAGRAMA DE SEQUÊNCIA DO USE CASE INCLUIR LOCATÁRIOS,
CURSO NORMAL

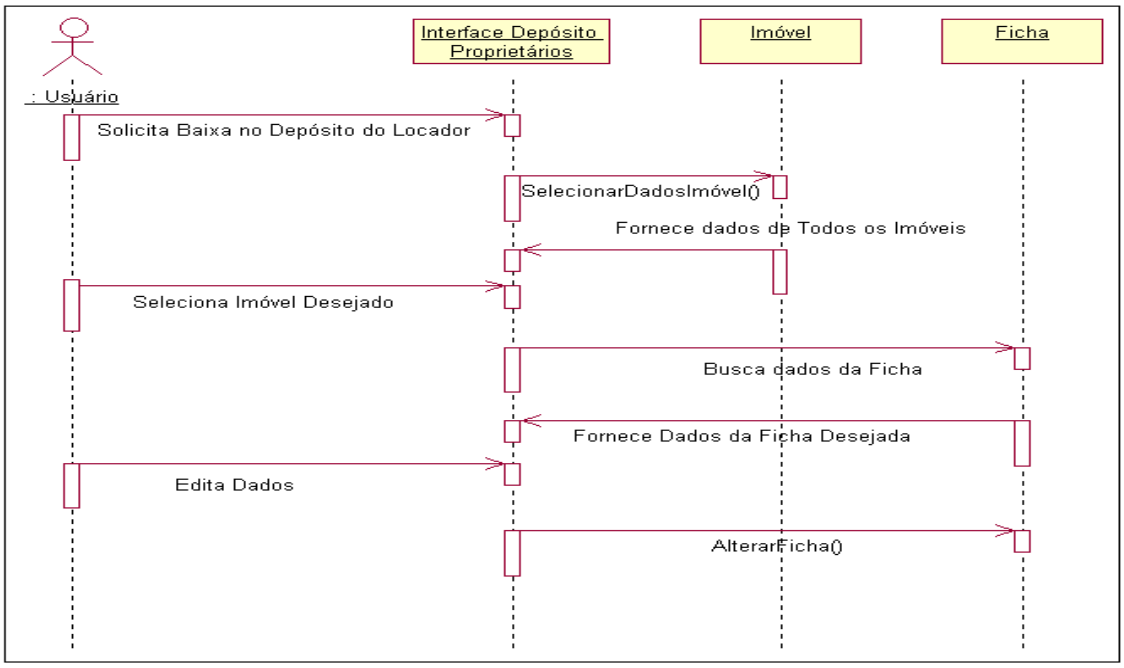


DIAGRAMA DE SEQUÊNCIA DO USE CASE DEPÓSITO DE PROPRIETÁRIOS, CURSO NORMAL

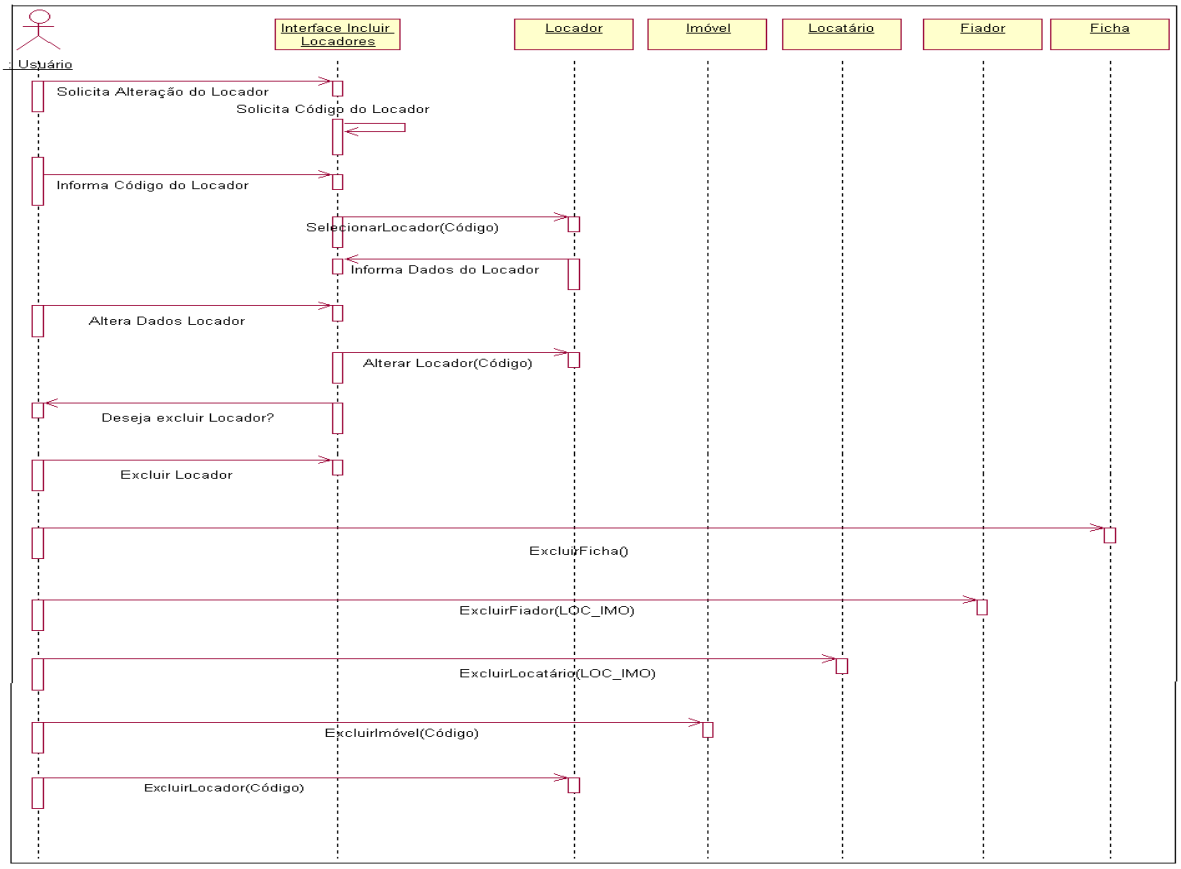


DIAGRAMA DE SEQUÊNCIA DO USE CASE ALTERAR LOCADORES, CURSO NORMAL

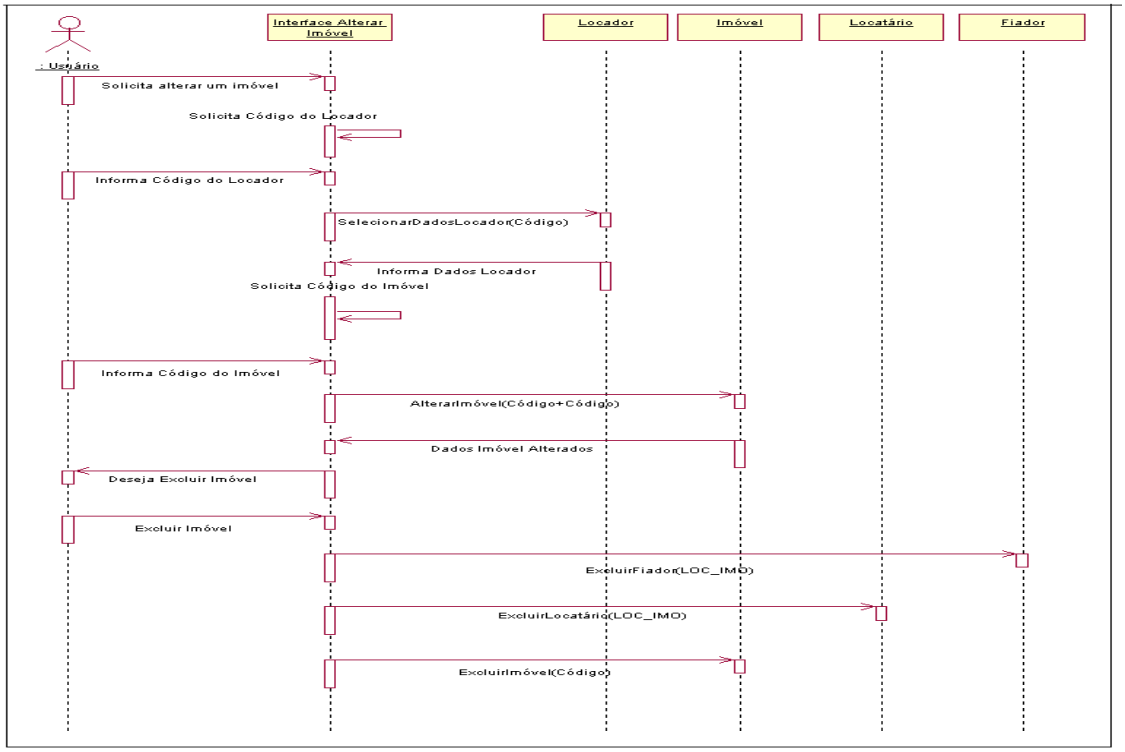


DIAGRAMA DE SEQUÊNCIA DO USE CASE ALTERAR IMÓVEIS, CURSO NORMAL

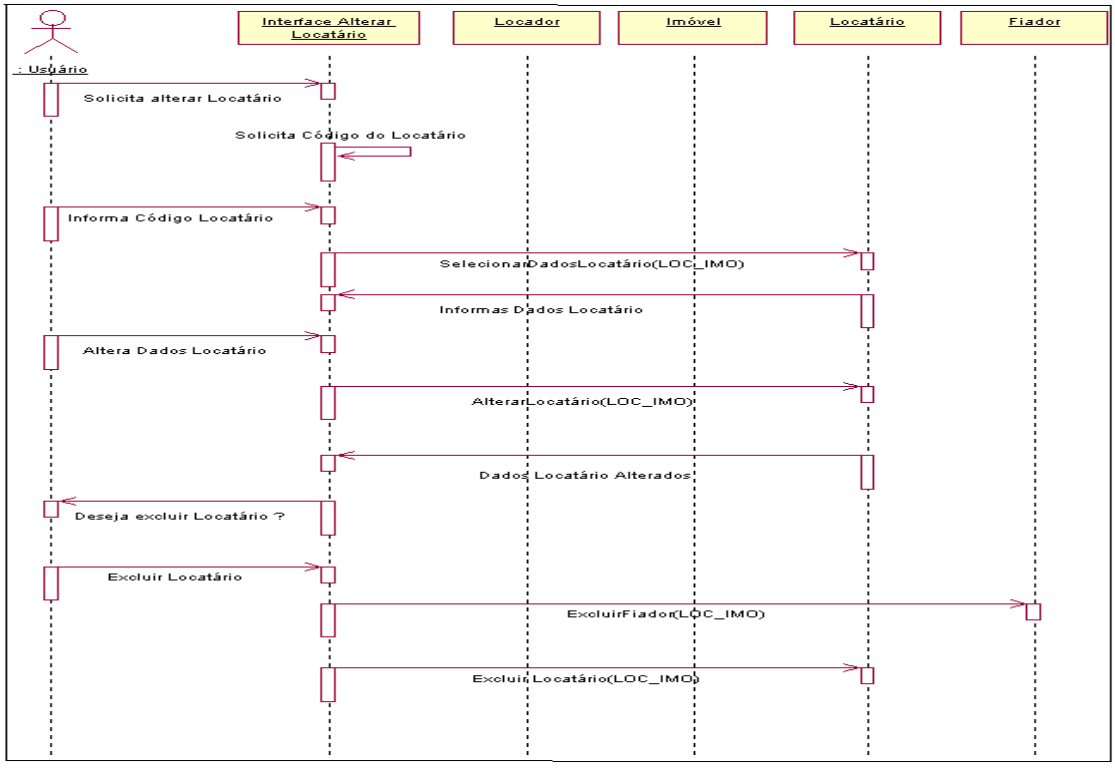


DIAGRAMA DE SEQUÊNCIA DO USE CASE ALTERAR LOCATÁRIOS, CURSO NORMAL

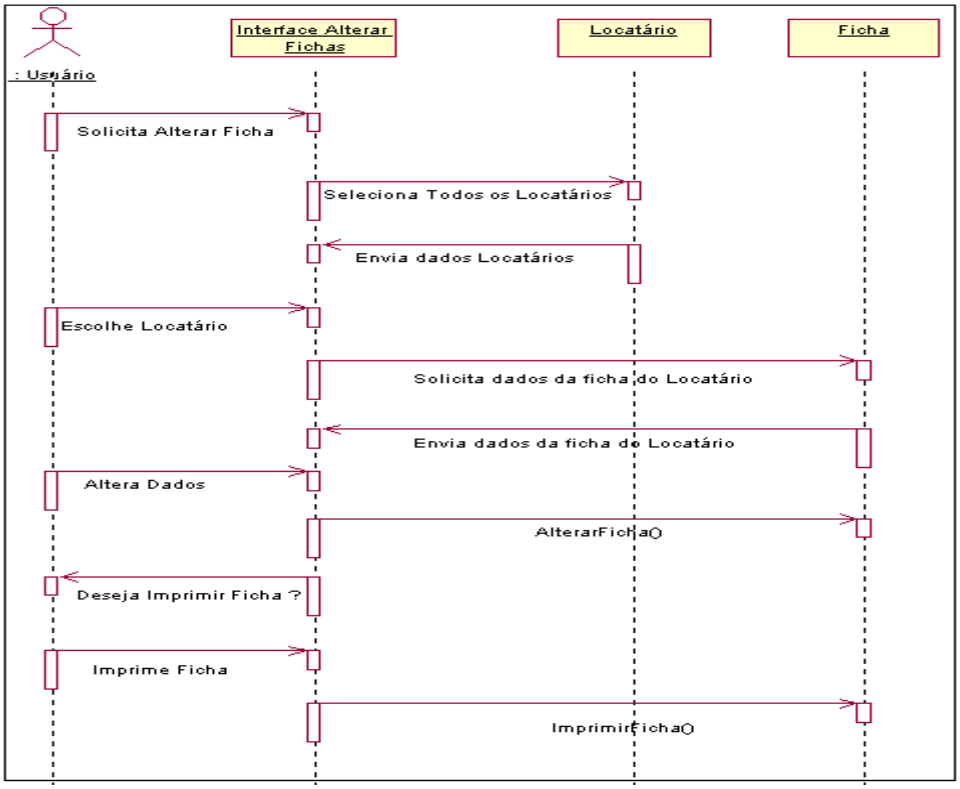


DIAGRAMA DE SEQUÊNCIA DO USE CASE ALTERAR FICHA, CURSO NORMAL

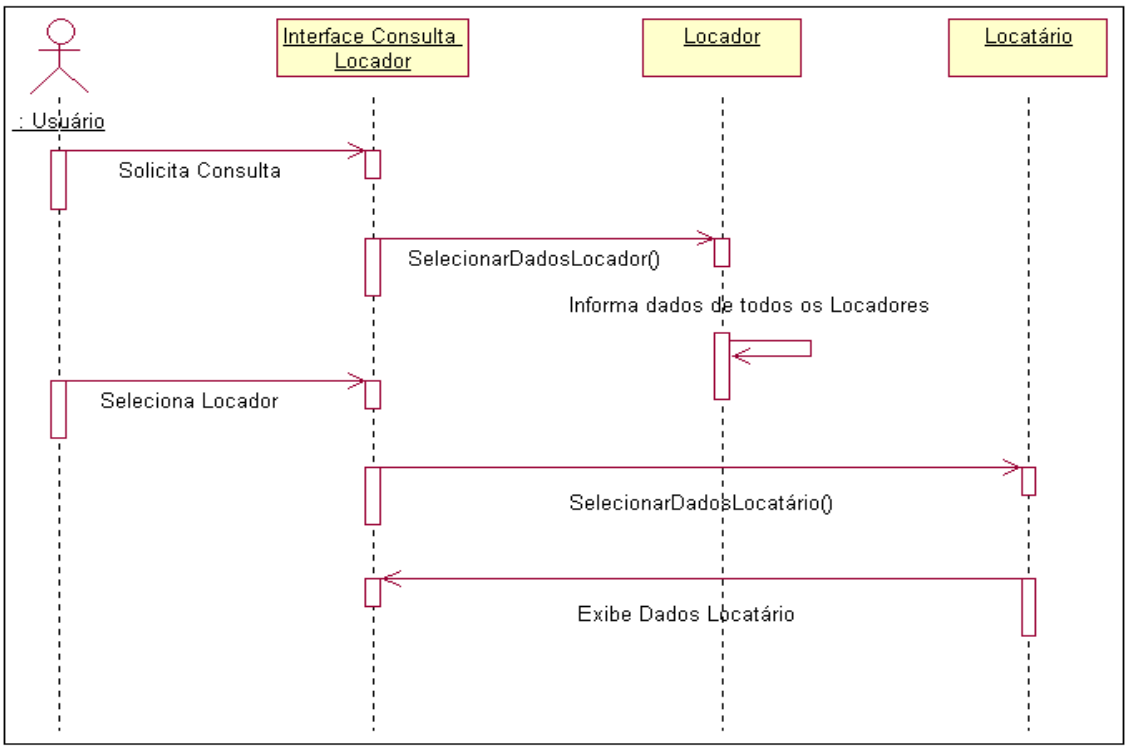


DIAGRAMA DE SEQUÊNCIA DO USE CASE CONSULTA LOCADORES, CURSO NORMAL

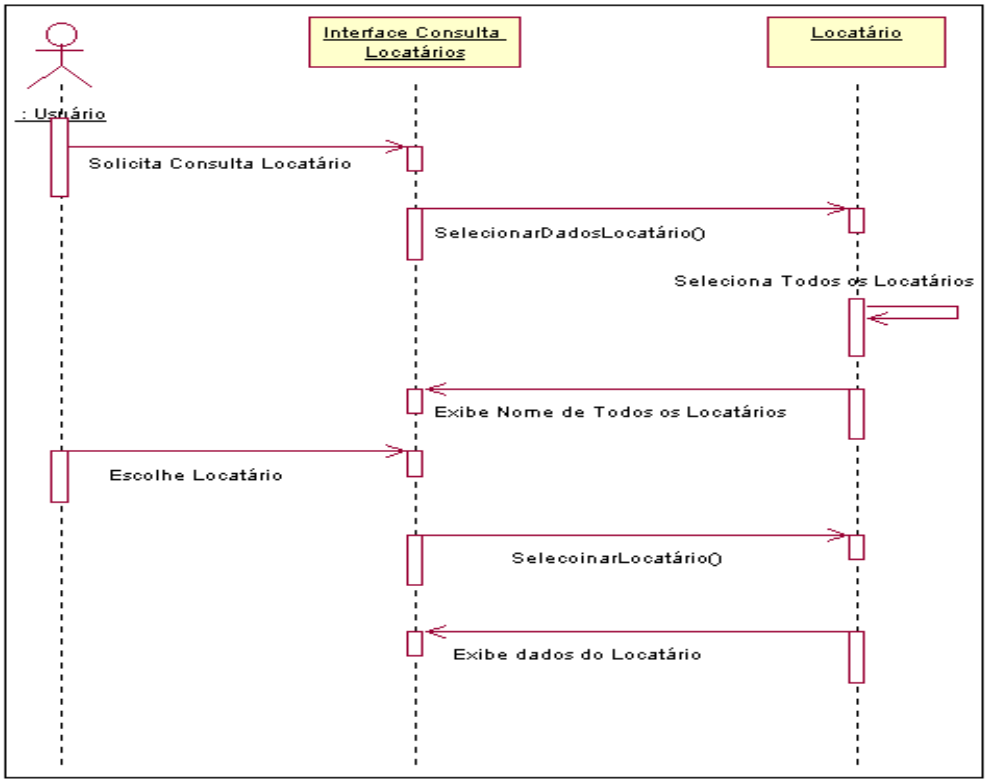


DIAGRAMA DE SEQUÊNCIA DO USE CASE CONSULTA LOCATÁRIOS, CURSO NORMAL

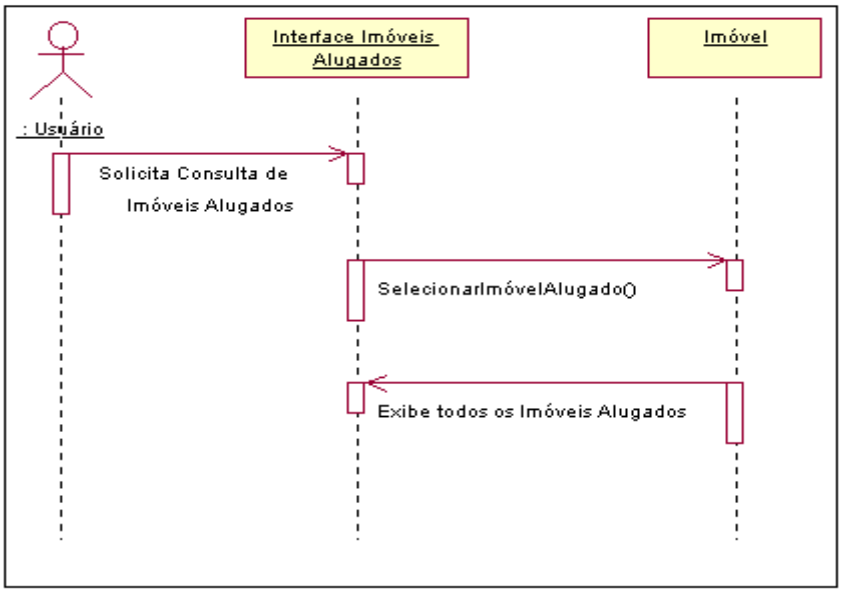


DIAGRAMA DE SEQUÊNCIA DO USE CASE CONSULTA IMÓVEIS ALUGADOS, CURSO NORMAL

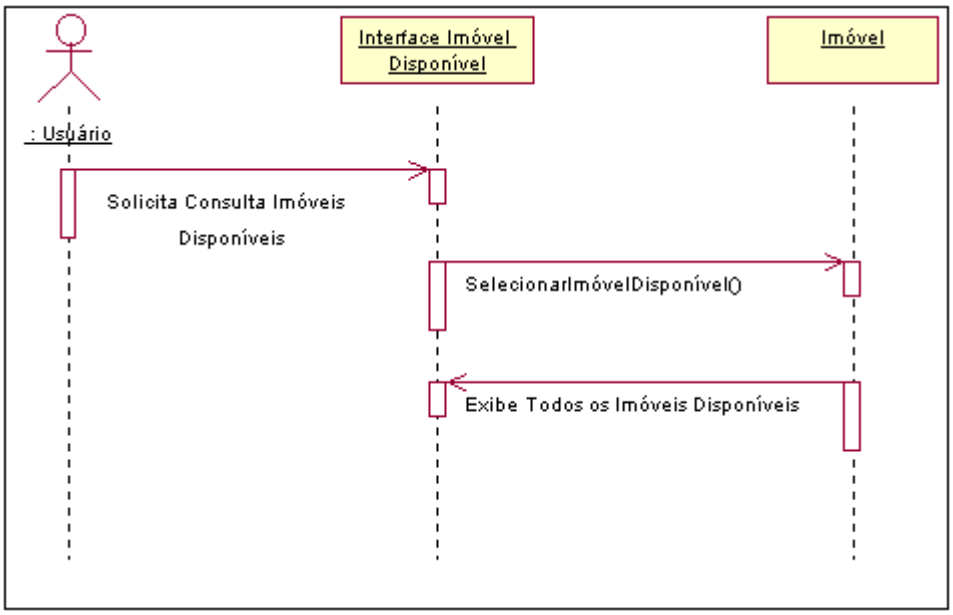


DIAGRAMA DE SEQUÊNCIA DO USE CASE CONSULTA IMÓVEIS DISPONÍVEIS, CURSO NORMAL

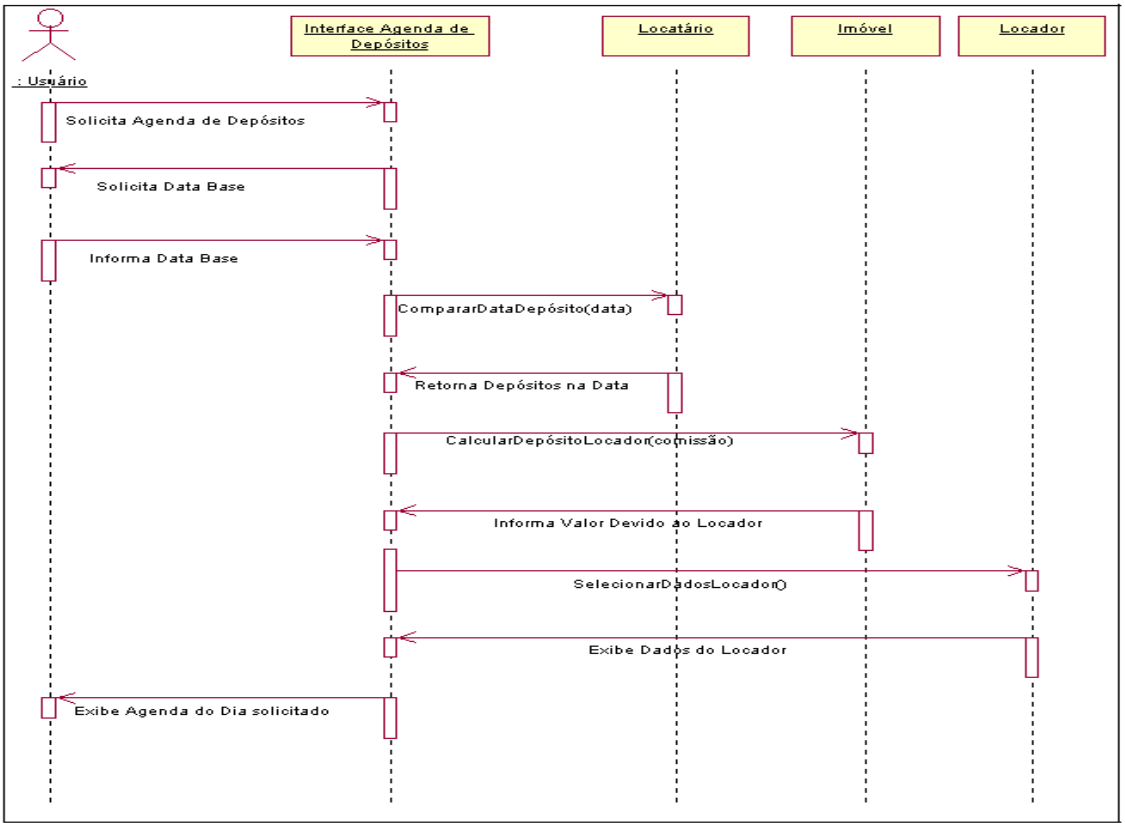


DIAGRAMA DE SEQUÊNCIA DO USE CASE AGENDA DE DEPÓSITOS,

CURSO NORMAL

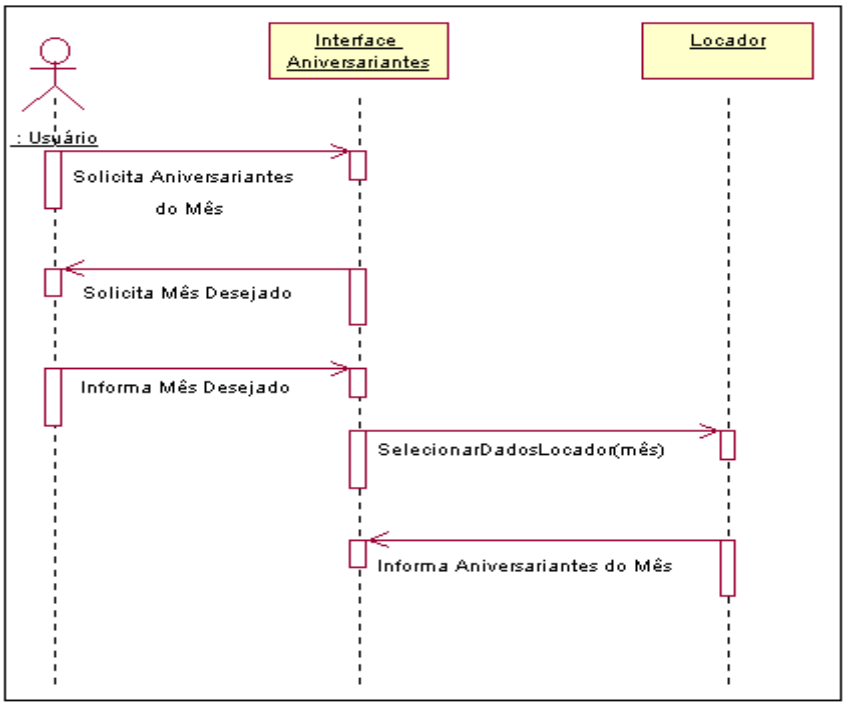


DIAGRAMA DE SEQUÊNCIA DO USE CASE ANIVERSARIANTES DO MÊS, CURSO NORMAL

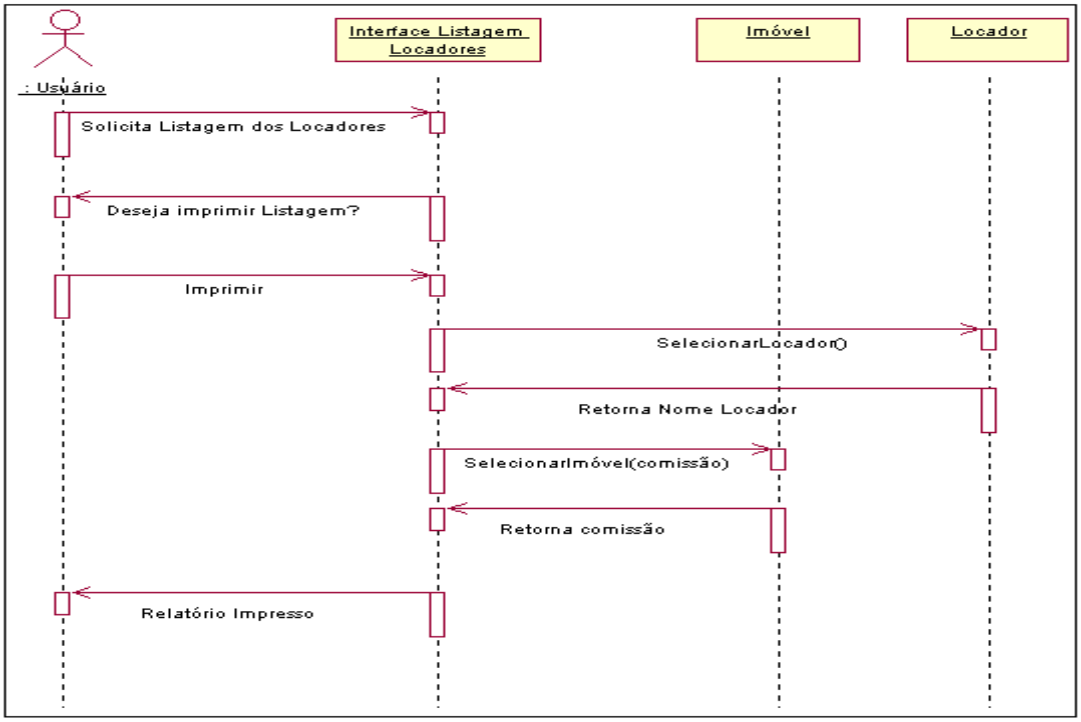


DIAGRAMA DE SEQUÊNCIA DO USE CASE LISTAGEM LOCADORES,

CURSO NORMAL

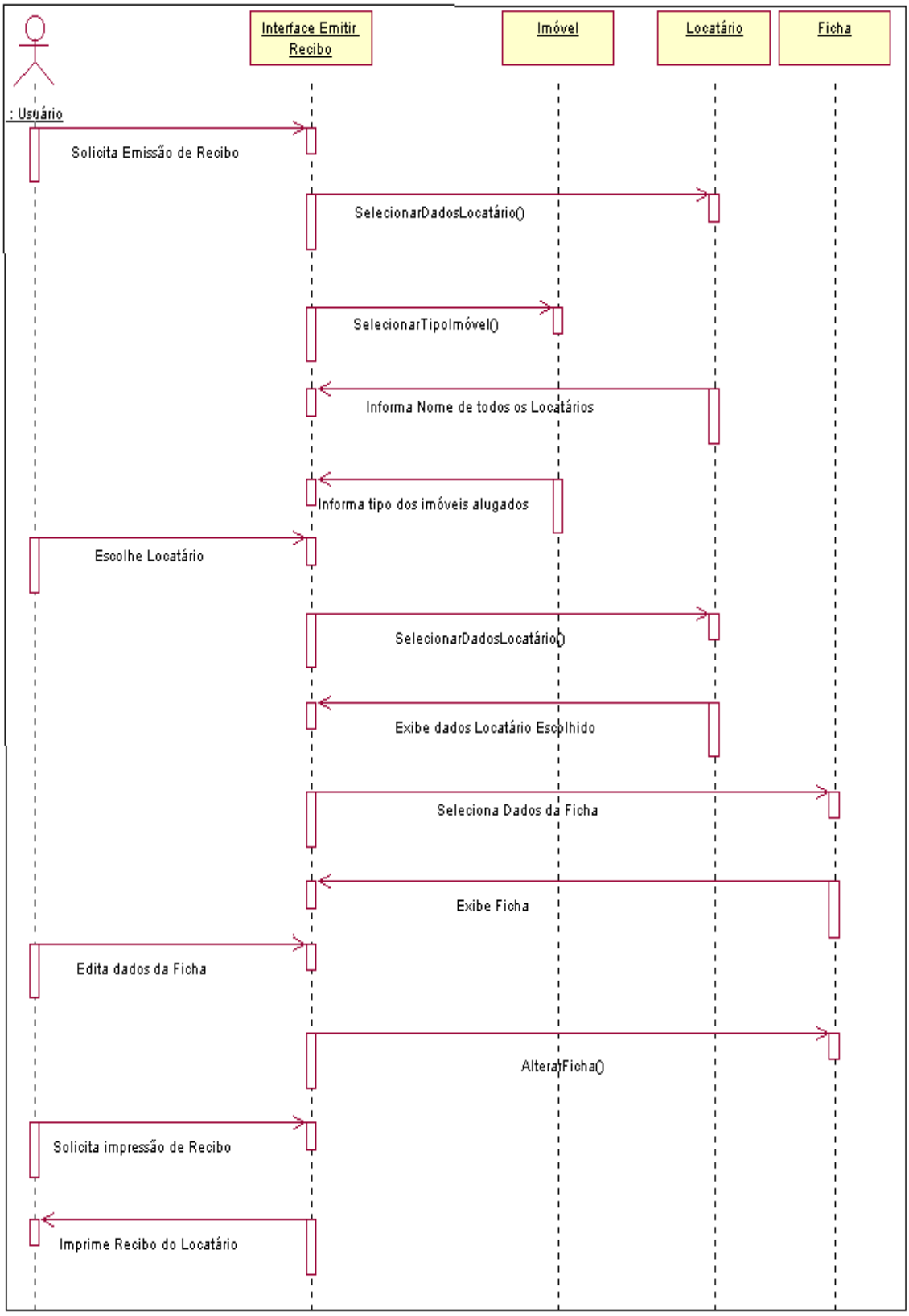


DIAGRAMA DE SEQUÊNCIA DO USE CASE EMITIR RECIBO, CURSO NORMAL

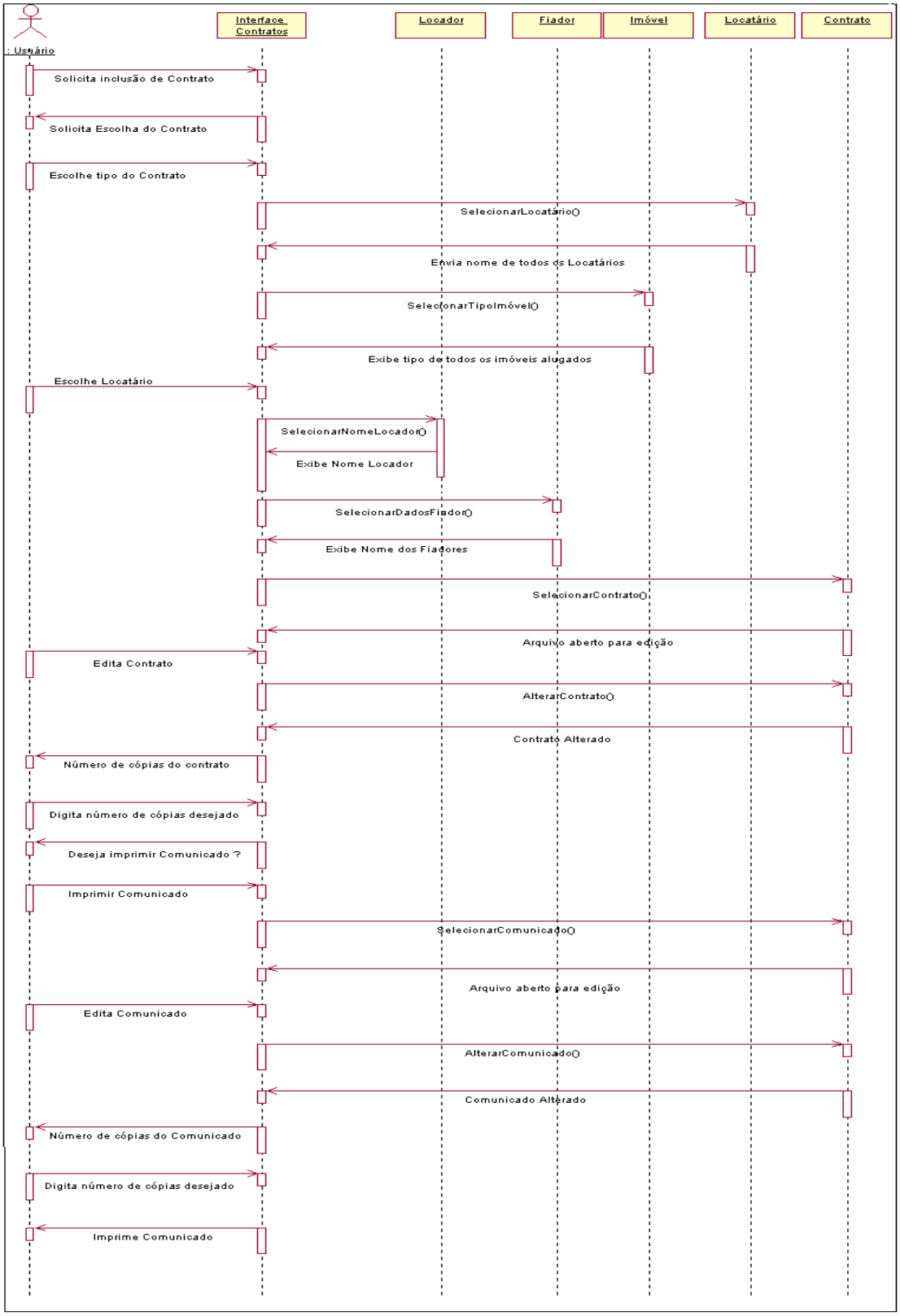


DIAGRAMA DE SEQUÊNCIA DO USE CASE CONTRATOS, CURSO NORMAL

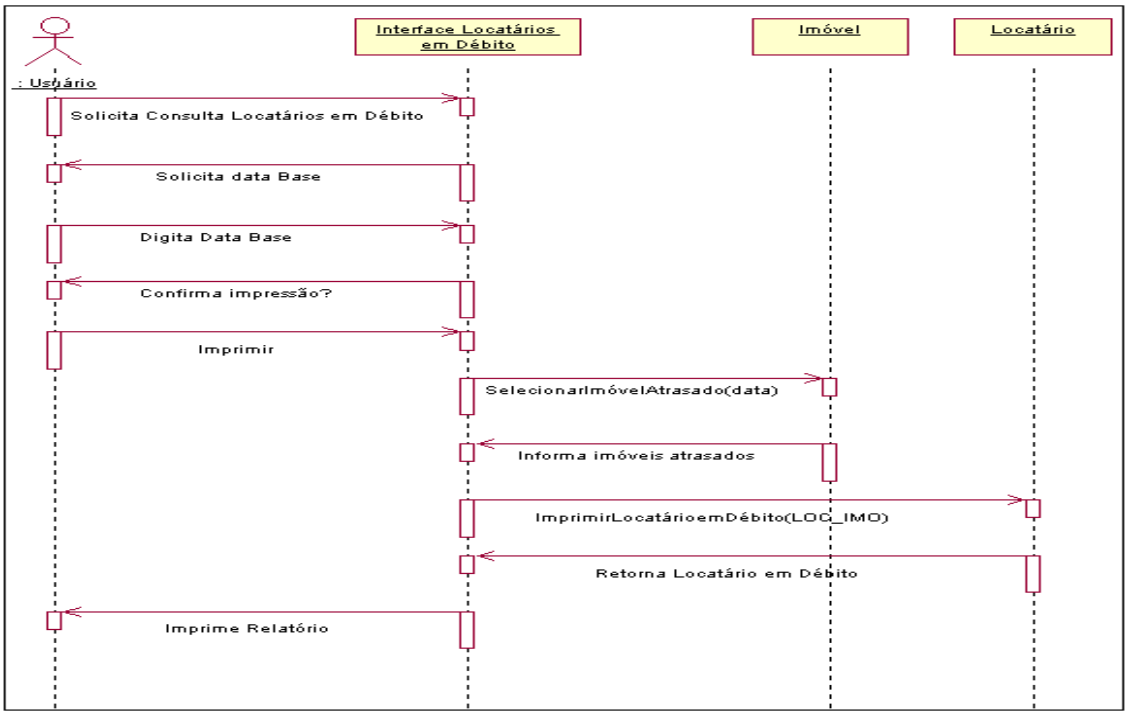


DIAGRAMA DE SEQUÊNCIA DO USE CASE LOCATÁRIOS EM DÉBITO, CURSO NORMAL

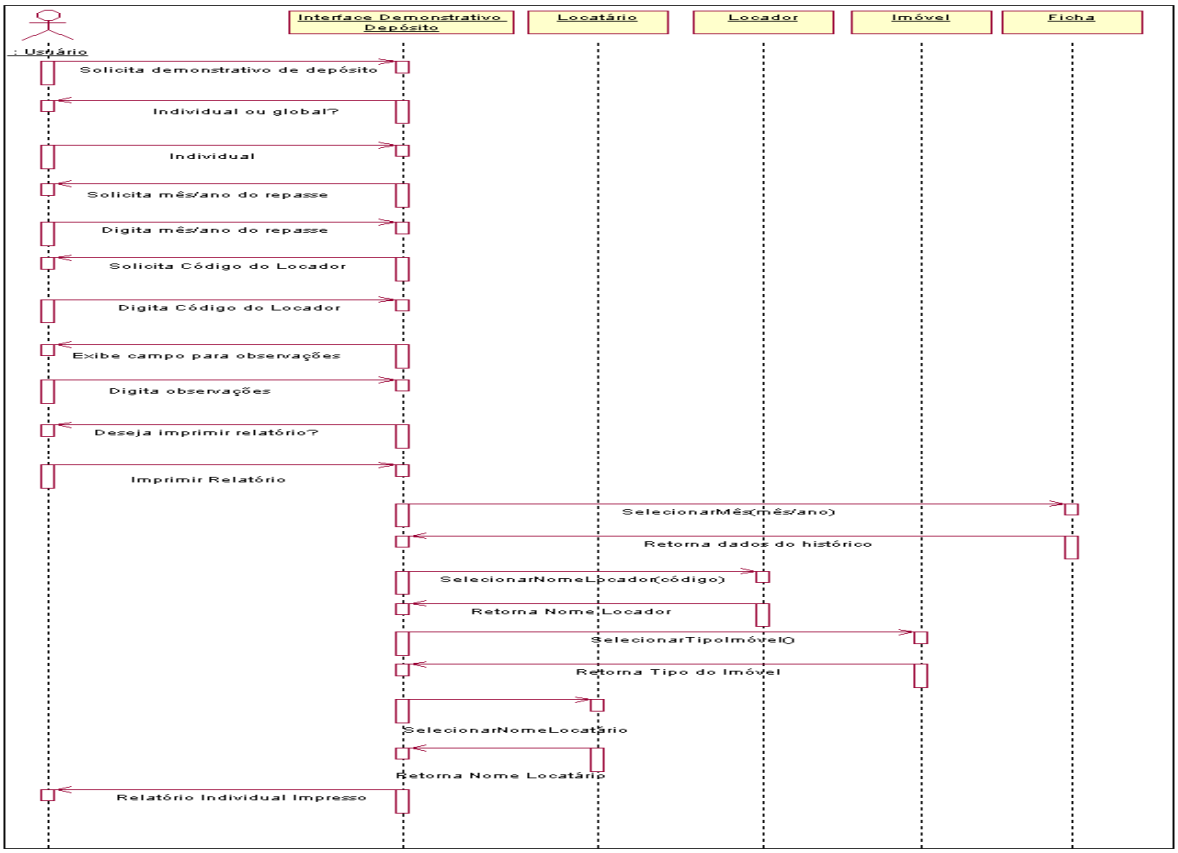


DIAGRAMA DE SEQUÊNCIA DO USE CASE DEMONST. DE DEPÓSITO, CURSO NORMAL

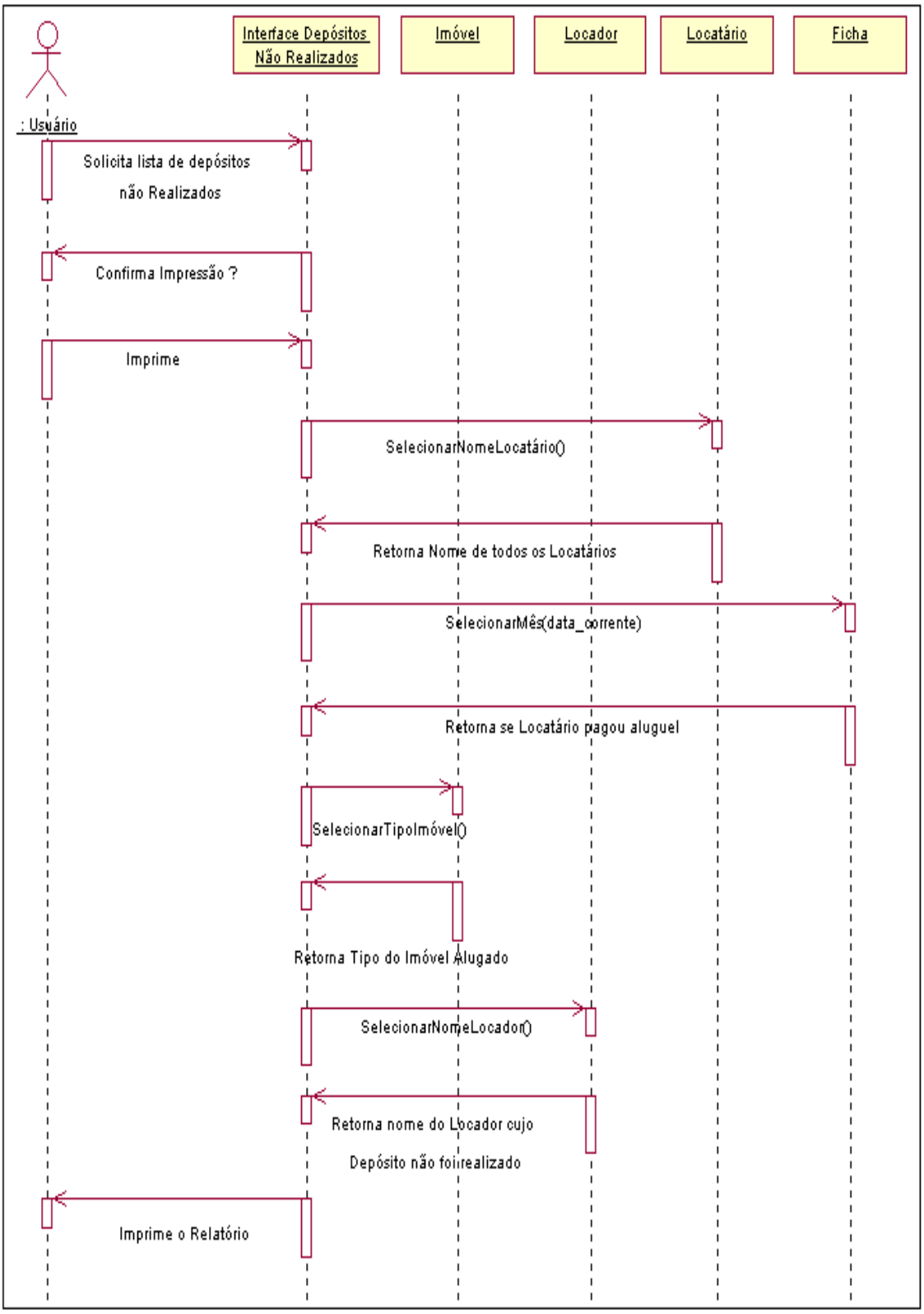


DIAGRAMA DE SEQUÊNCIA DO USE CASE DEPÓSITOS NÃO REALIZADOS,

CURSO NORMAL

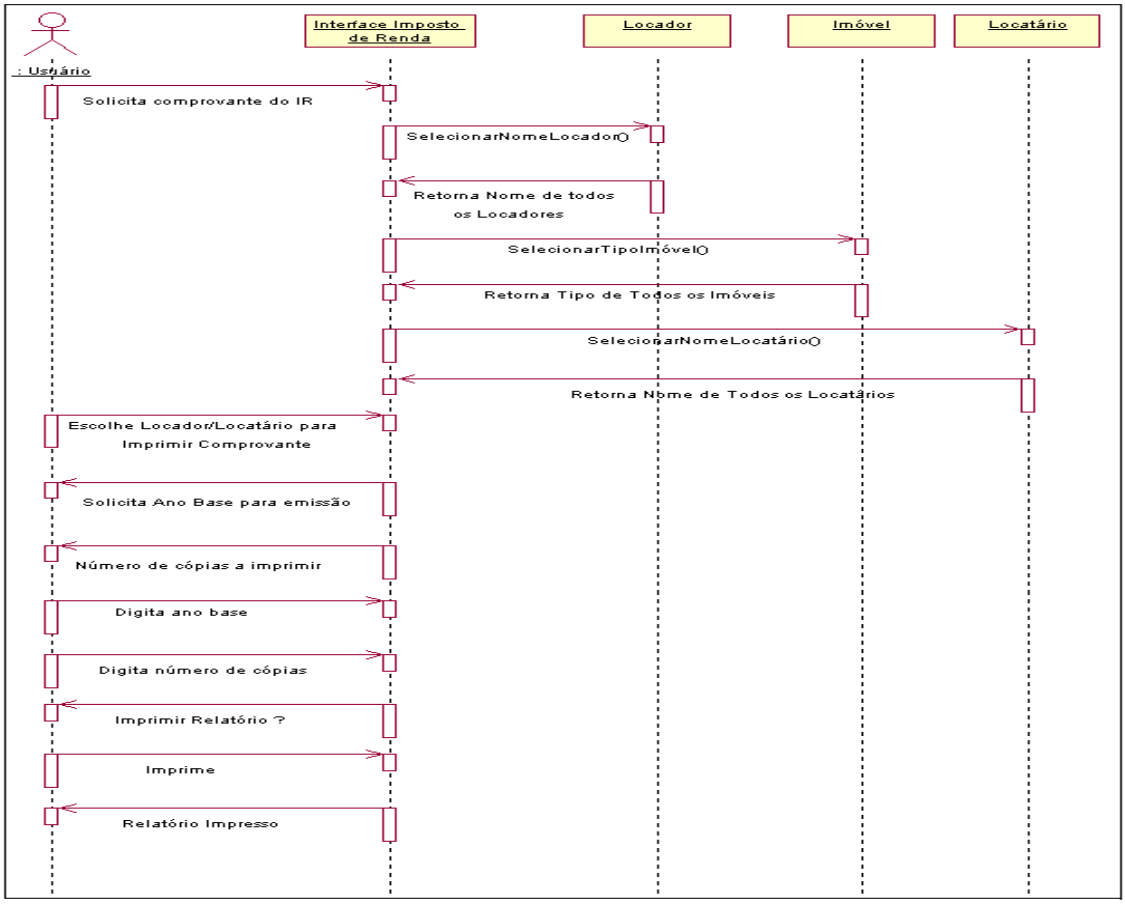


DIAGRAMA DE SEQUÊNCIA DO USE CASE IMPOSTO DE RENDA, CURSO NORMAL

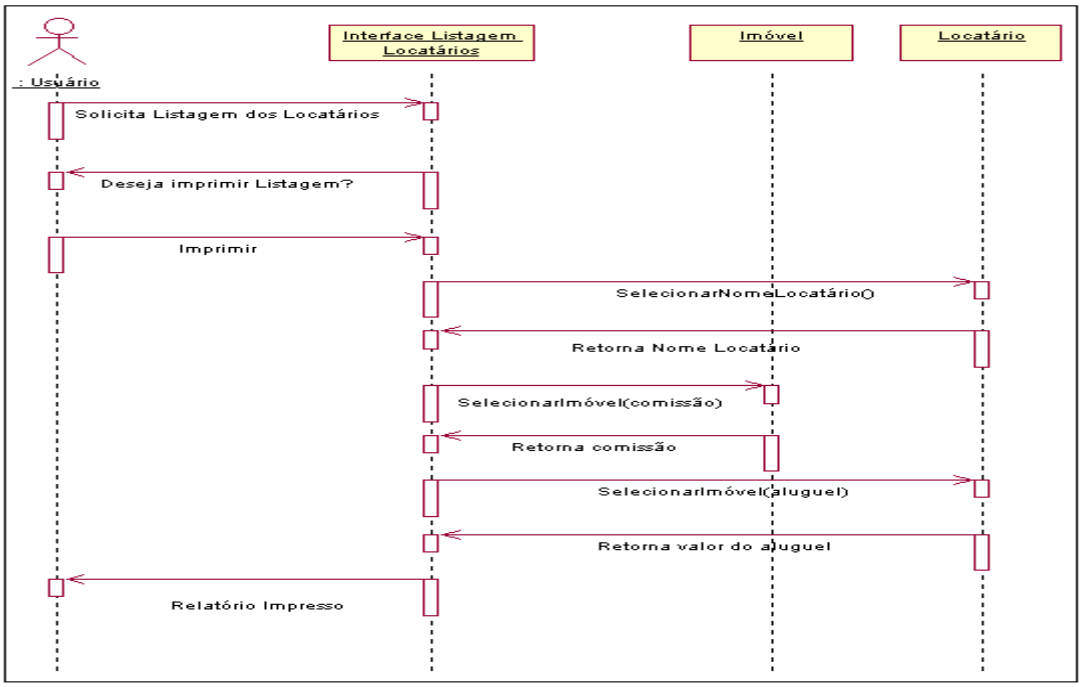


DIAGRAMA DE SEQUÊNCIA DO USE CASE LISTAGEM LOCATÁRIOS, CURSO NORMAL

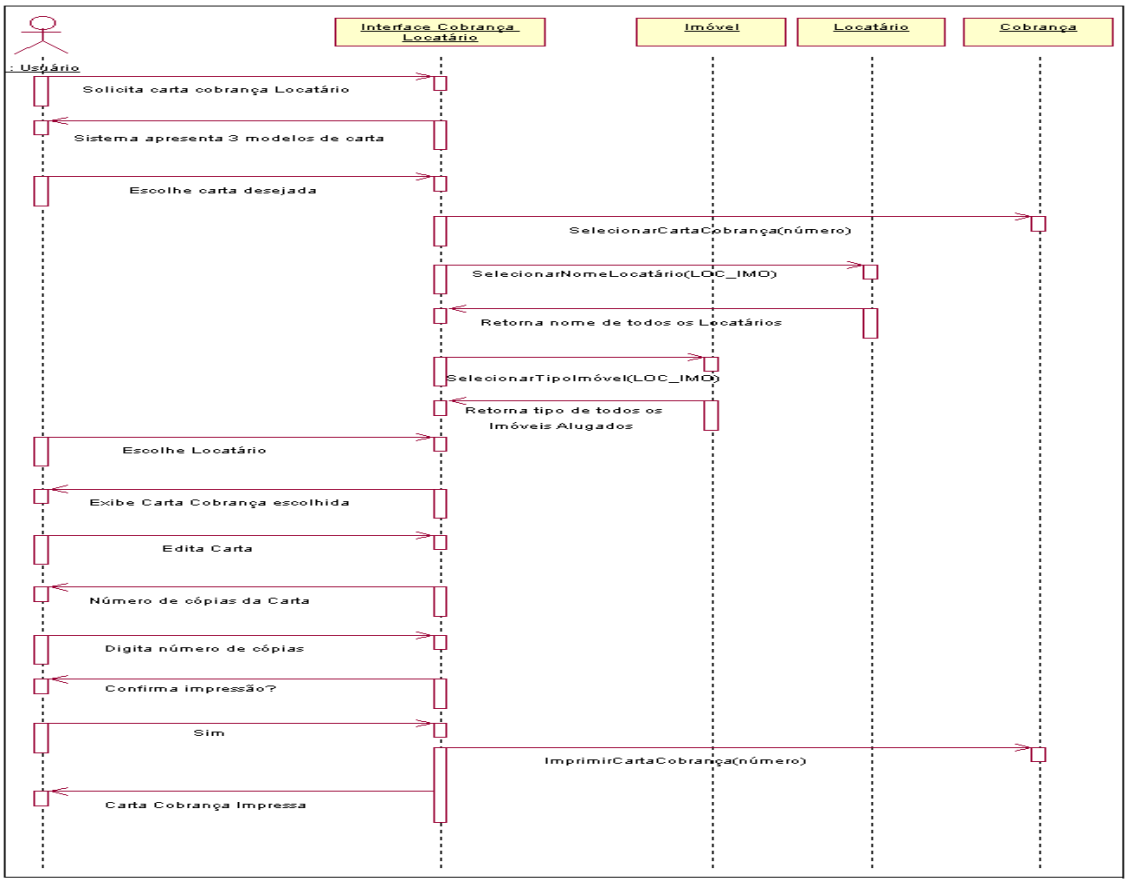


DIAGRAMA DE SEQUÊNCIA DO USE CASE CARTA COB. LOCATÁRIO, CURSO NORMAL

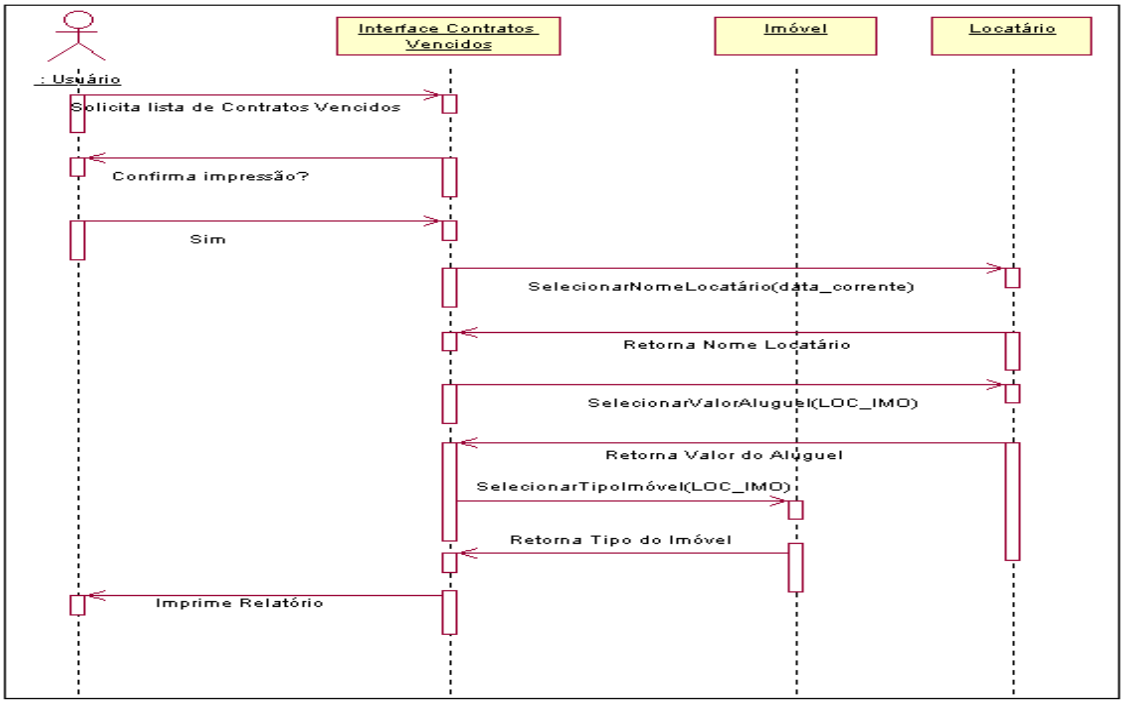


DIAGRAMA DE SEQUÊNCIA DO USE CASE CONTRATOS VENCIDOS, CURSO NORMAL

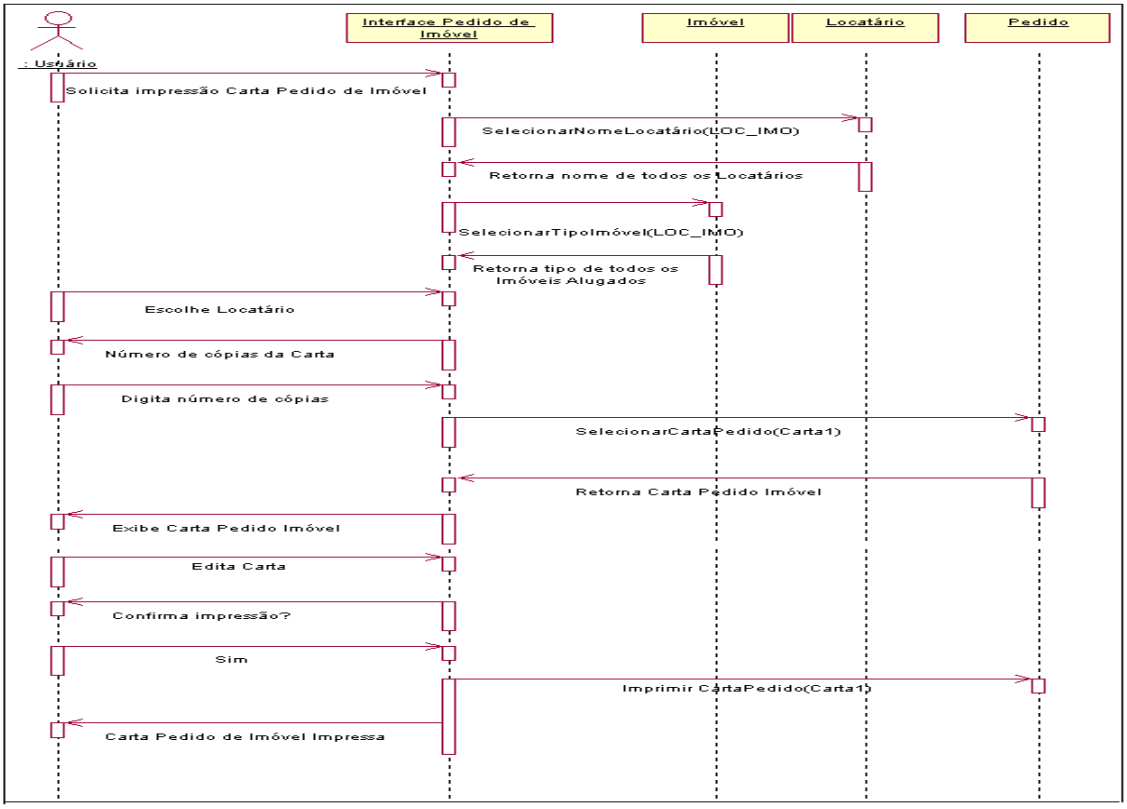


DIAGRAMA DE SEQUÊNCIA DO USE CASE CARTA PED. DE IMÓVEL, CURSO NORMAL

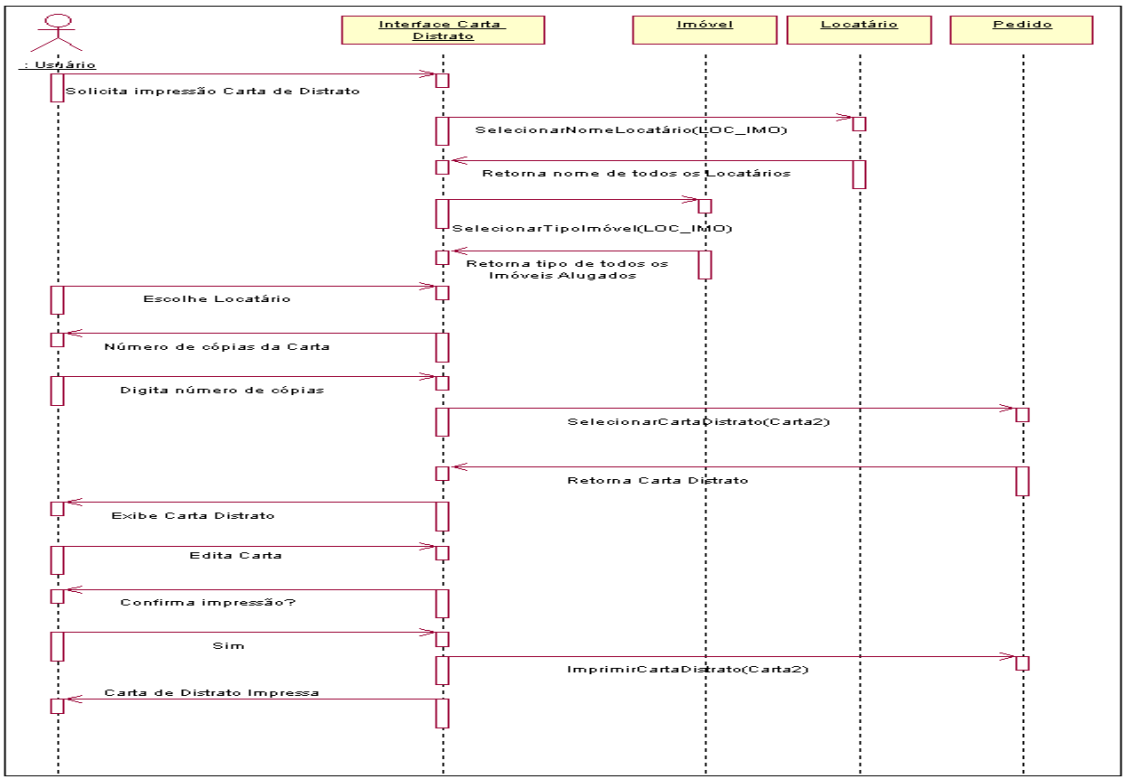


DIAGRAMA DE SEQUÊNCIA DO USE CASE CARTA DE DISTRATO, CURSO NORMAL

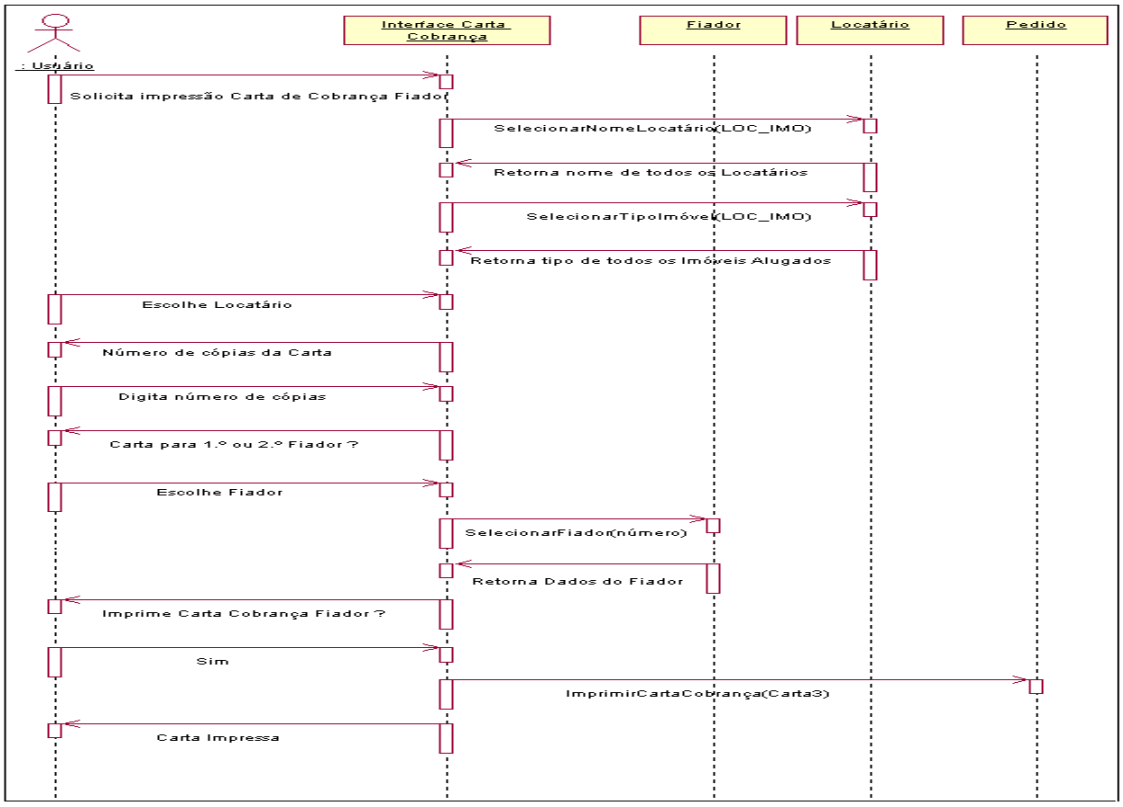


DIAGRAMA DE SEQUÊNCIA DO USE CASE CARTA COB. FIADOR, CURSO NORMAL

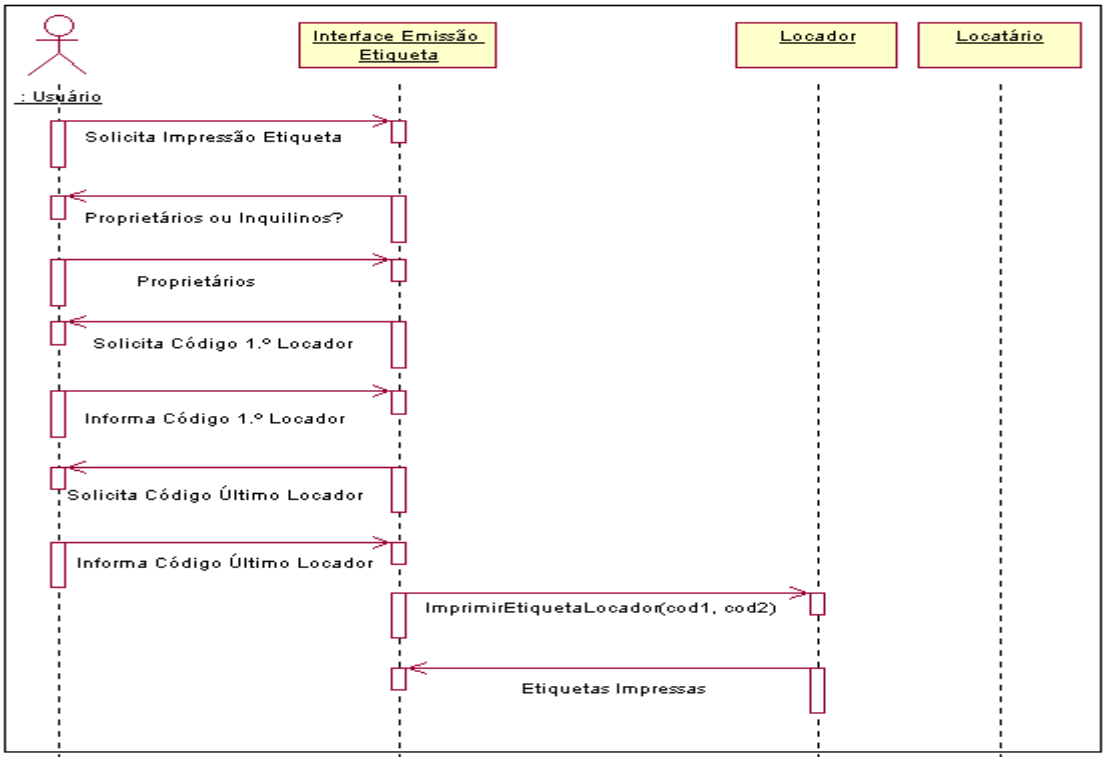


DIAGRAMA DE SEQUÊNCIA DO USE CASE EMISSÃO ETIQUETA, CURSO NORMAL

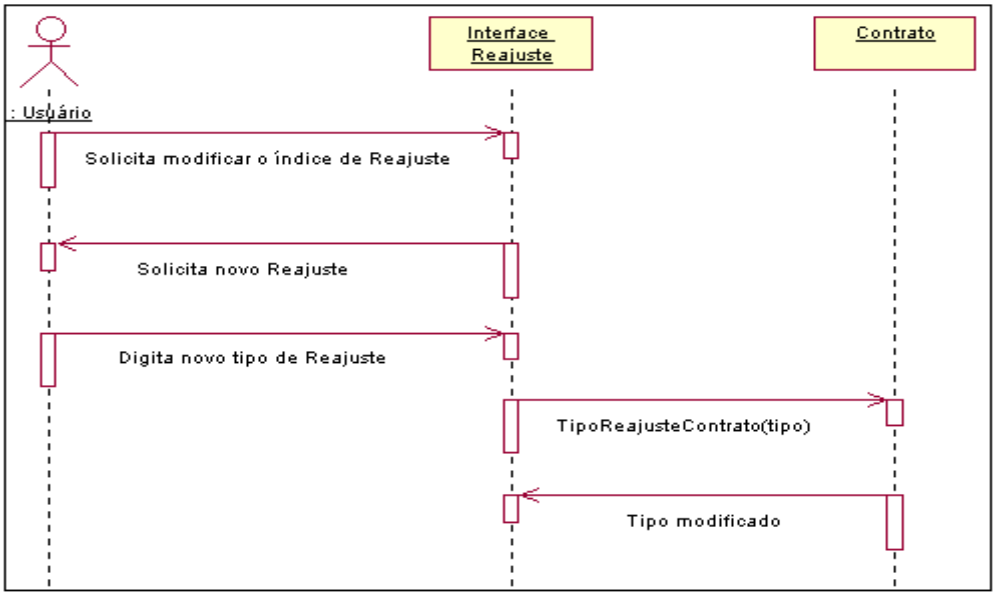


DIAGRAMA DE SEQUÊNCIA DO USE CASE REAJUSTE, CURSO NORMAL

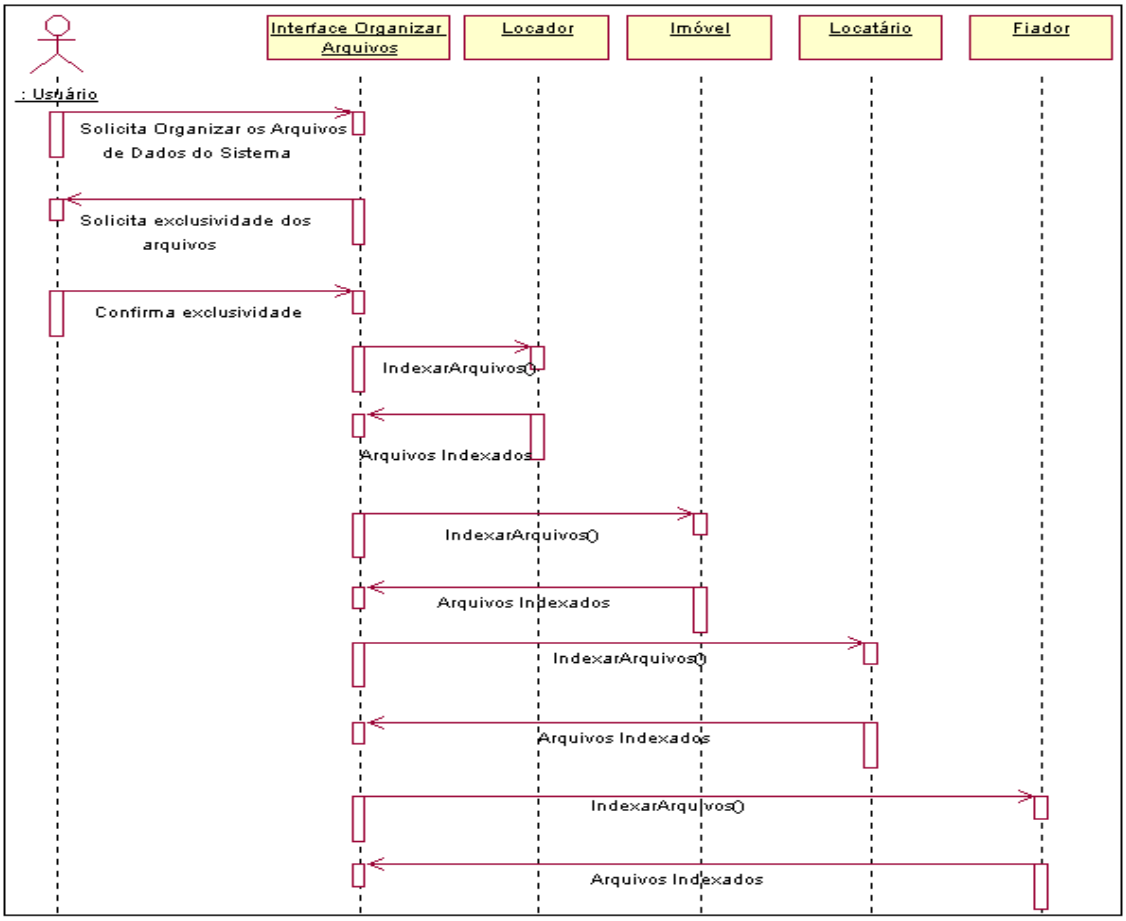


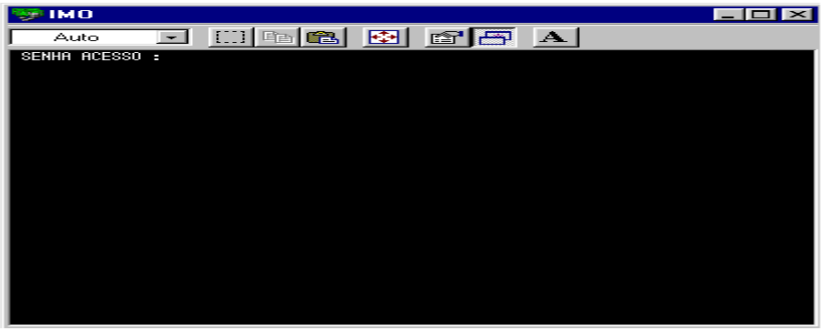
DIAGRAMA DE SEQUÊNCIA DO USE CASE ORGANIZAR ARQUIVOS, CURSO NORMAL

FIGURAS DA ETAPA DE ENGENHARIA PROGRESSIVA DO SISTEMA DE CONTROLE DE LOCAÇÕES DE IMÓVEIS BASEADO EM CONCEITOS DE USABILIDADE

CLUSTER: GERAR O SISTEMA ORIENTADO A OBJETOS

PADRÃO 4: REALIZAR MELHORIAS NA INTERFACE:

FIGURA 1**: TABELA PROGRAMAS X ARQUIVOS**



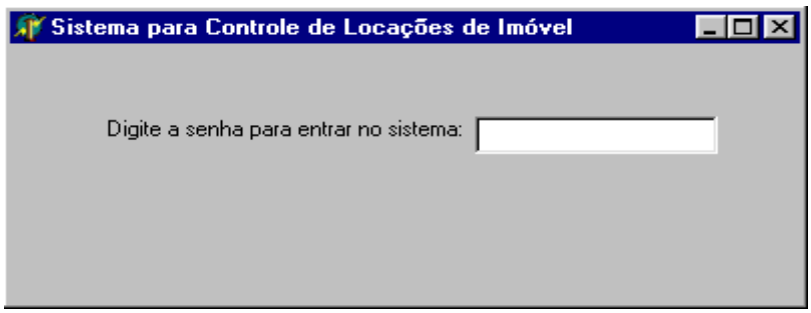
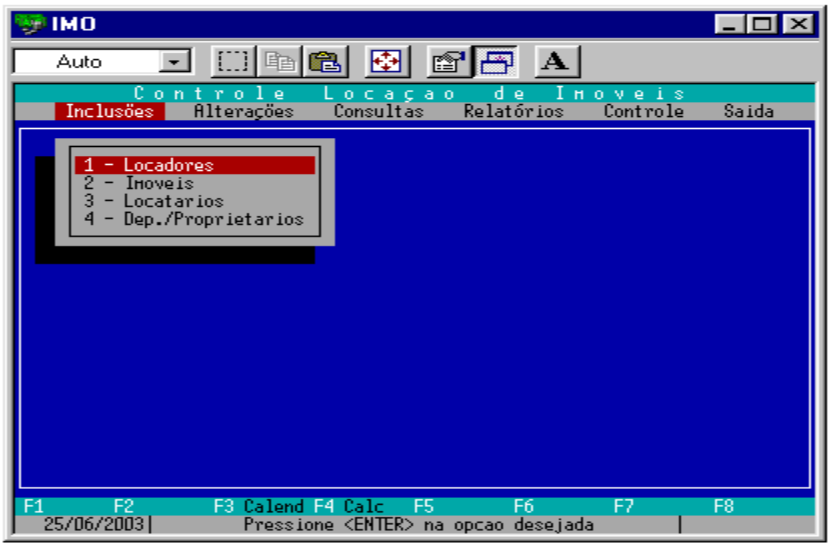


FIGURA 2**:TELA MENU INCLUSÕES:**

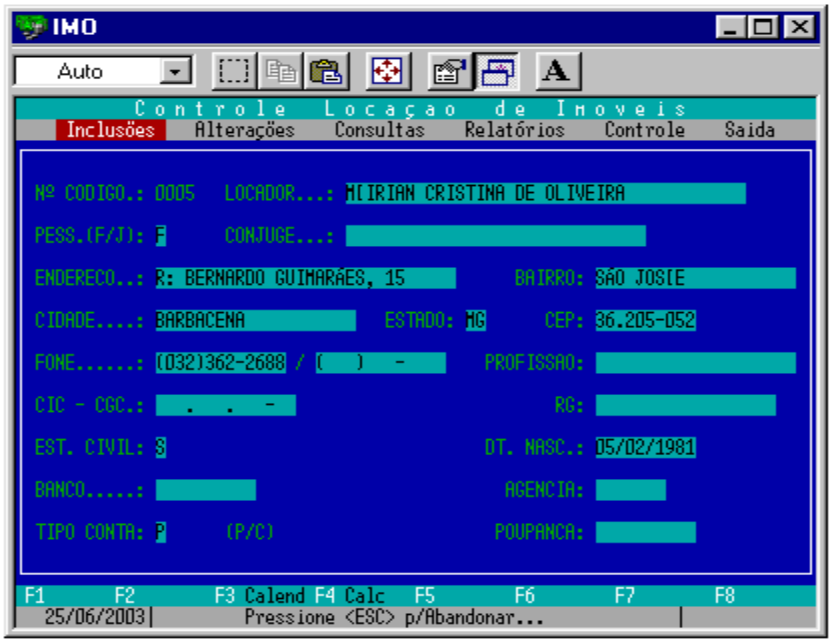
Aqui é mostrado o menu de abertura do sistema.



TELA MENU INCLUSÕES

FIGURA 3: TELA INCLUSÕES LOCADORES:

É aberta uma tela para inclusões de dados de locadores, se usuário confirmar a operação, os dados são gravados no arquivo Locador.dbf


IMO
Auto
Controle Locaçao de Imoveis
Inclusões Alterações Consultas Relatórios Controle Saida
Nº CODIGO.: 0005 LOCADOR...: MIRIAN CRISTINA DE OLIVEIRA
PESS.(F/J): F CONJUGE...:
ENDERECO..: R: BERNARDO GUIMARÃES, 15 BAIRRO: SÃO JOSIE
CIDADE....: BARBACENA ESTADO: MG CEP: 36.205-052
FONE.....: (032)362-2688 / ( ) - PROFISSAO:
CIC - CGC.: . . - RG:
EST. CIVIL: S DT. NASC.: 05/02/1981
BANCO.....: AGENCIA:
TIPO CONTA: P (P/C) POUPANCA:
F1 F2 F3 Calend F4 Calc F5 F6 F7 F8
25/06/2003 Pressione <ESC> p/Abandonar...

.

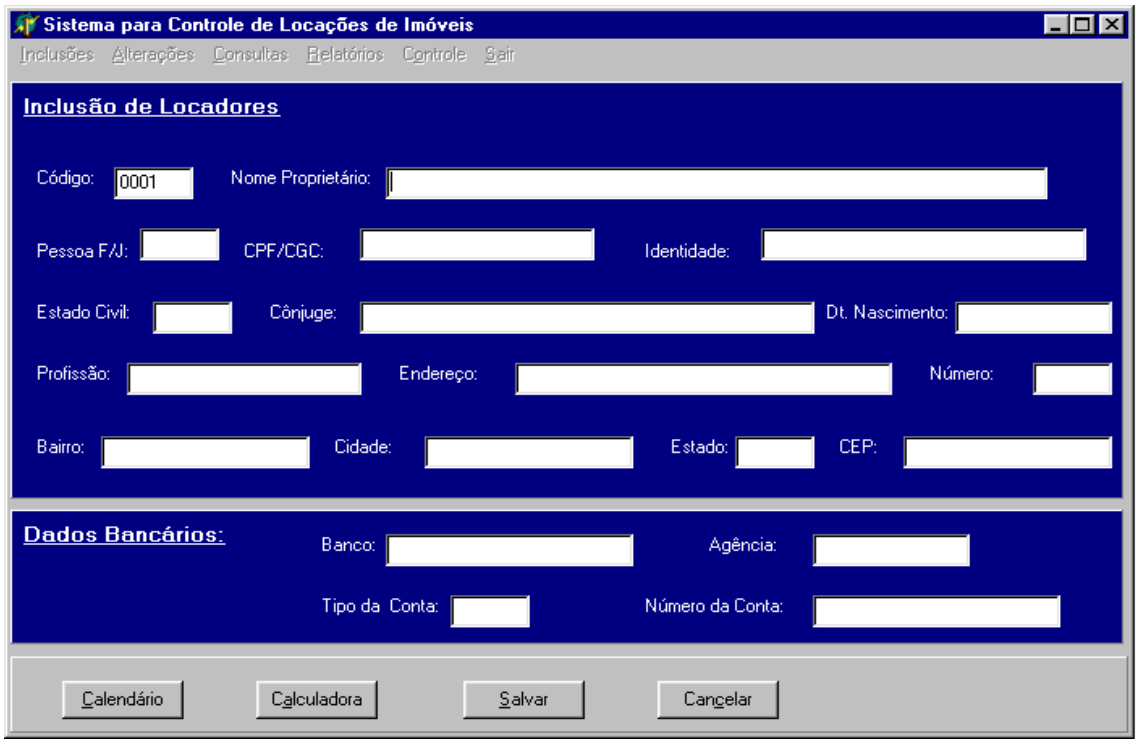
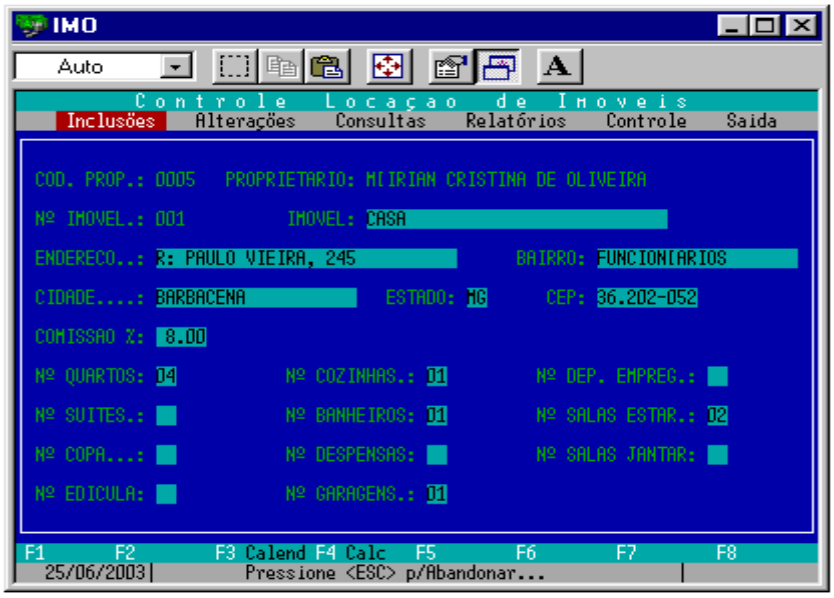

Sistema para Controle de Locações de Imóveis
Inclusões Alterações Consultas Relatórios Controle Sair
Inclusão de Locadores
Código: 0001
Nome Proprietário:
Pessoa F/J:
CPF/CGC:
Identidade:
Estado Civil:
Cônjuge:
Dt. Nascimento:
Profissão:
Endereço:
Número:
Bairro:
Cidade:
Estado:
CEP:
Dados Bancários:
Banco:
Agência:
Tipo da Conta:
Número da Conta:
Calendário
Calculadora
Salvar
Cancelar

FIGURA 4**: TELA INCLUSÕES IMÓVEIS:**

É aberta uma tela para inclusões de dados dos imóveis, se usuário confirmar a operação, os dados são gravados no arquivo Imóvel.dbf.



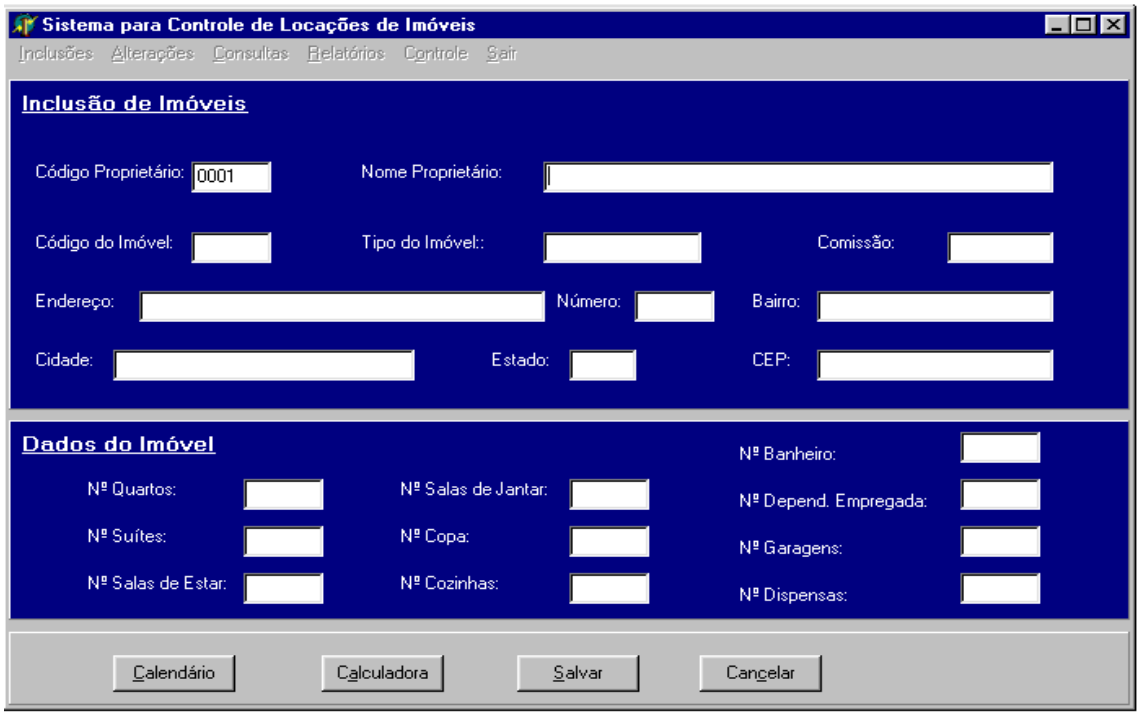


FIGURA 5**: TELA INCLUSÕES LOCATÁRIOS:**

É aberta uma tela para inclusões de dados de locatários e fiadores; se usuário confirmar a operação, os dados são gravados no arquivo Locatário.dbf e Fiador.dbf.

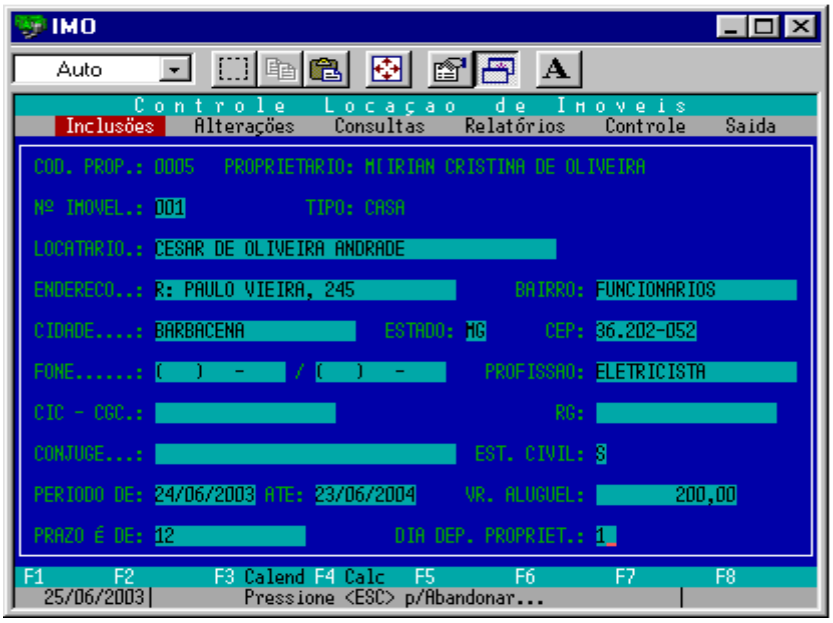


FIGURA 6**: TELA ALTERAÇÕES LOCADORES:**

É aberta uma tela idêntica à tela de inclusões de locadores, para que o usuário possa editar os dados. A única diferença é que o usuário precisa fornecer o código do locador que deseja alterar. Pergunta-se ainda se o usuário deseja excluir o registro referente ao locador. Usuário pode excluí-lo na tela de alterações de locadores.

FIGURA 7**: TELA ALTERAÇÕES IMÓVEIS:**

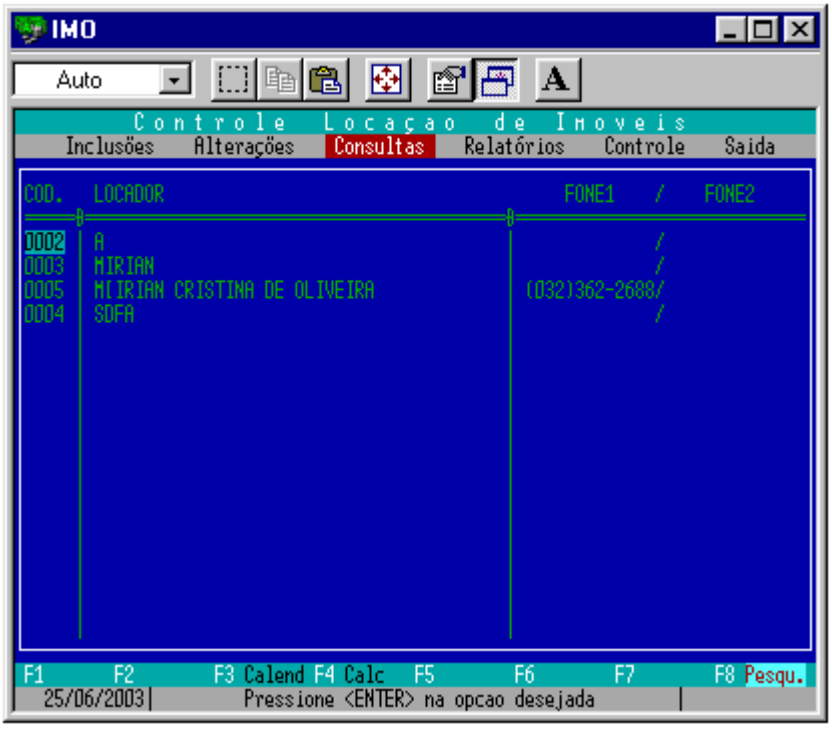
É aberta uma tela idêntica à tela de inclusões de imóveis, para que o usuário possa editar os dados. A única diferença é que o usuário precisa fornecer o código do locador e a parte do código do imóvel que deseja alterar. Pergunta-se ainda se o usuário deseja excluir o registro referente ao imóvel. Usuário pode excluí-lo na tela de alterações de imóveis.

FIGURA 8**: TELA ALTERAÇÕES LOCATÁRIOS:**

É aberta uma tela idêntica à tela de inclusões de locatários, para que o usuário possa editar os dados. A única diferença é que o usuário precisa fornecer o código do locatário que deseja alterar. Pergunta-se ainda se o usuário deseja excluir o registro referente ao locatário. Usuário pode excluí-lo na tela de alterações de imóveis.

FIGURA 9**: TELA CONSULTAS LOCADORES:**

É aberta uma tela contendo todos os dados dos locadores, existentes no arquivo Locador.dbf. Os locadores estão ordenados pelo código e o código de um locador que teve seu registro excluído, nunca é reaproveitado.Se o usuário quiser refinar sua pesquisa, pode encontrar todos os locatários relacionados com o locador consultado, bastando para isso pressionar a tecla ENTER.


IMO
Auto
Controle Locaçao de Imoveis
Inclusões Alterações Consultas Relatórios Controle Saida
COD. LOCADOR FONE1 / FONE2
0002 A /
0003 MIRIAN /
0005 MIRIAN CRISTINA DE OLIVEIRA (032)362-2688/
0004 SOFA /
F1 F2 F3 Calend F4 Calc F5 F6 F7 F8 Pesqu.
25/06/2003 Pressione <ENTER> na opcao desejada

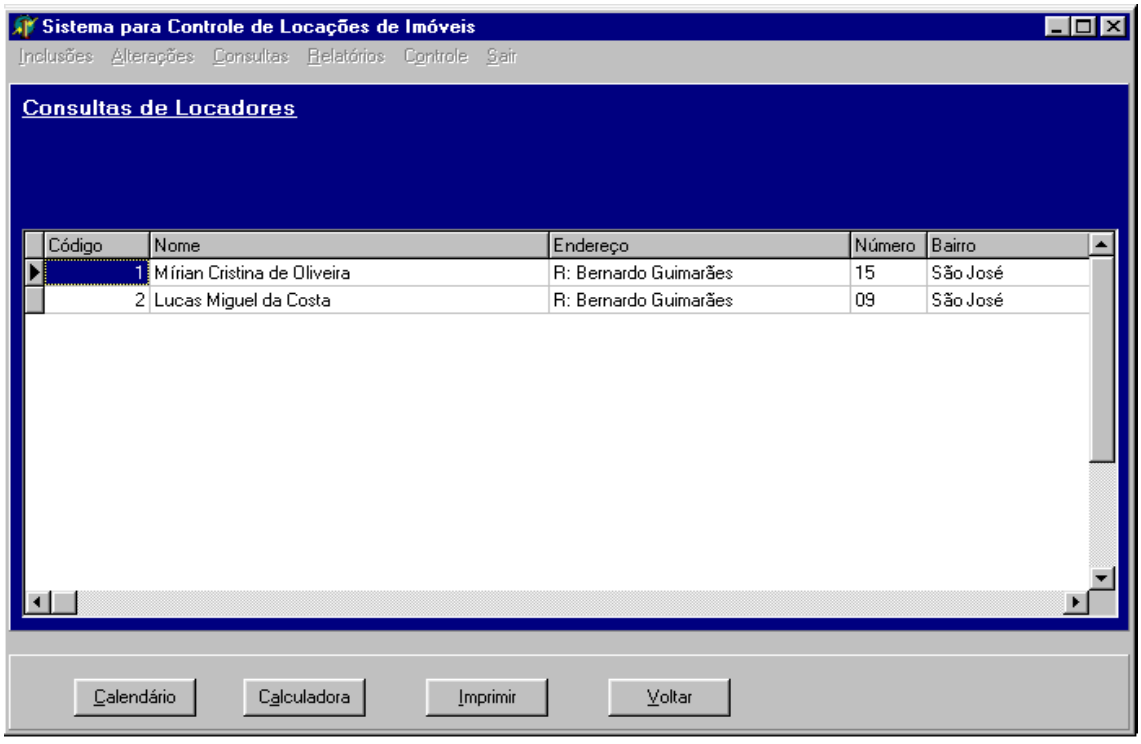
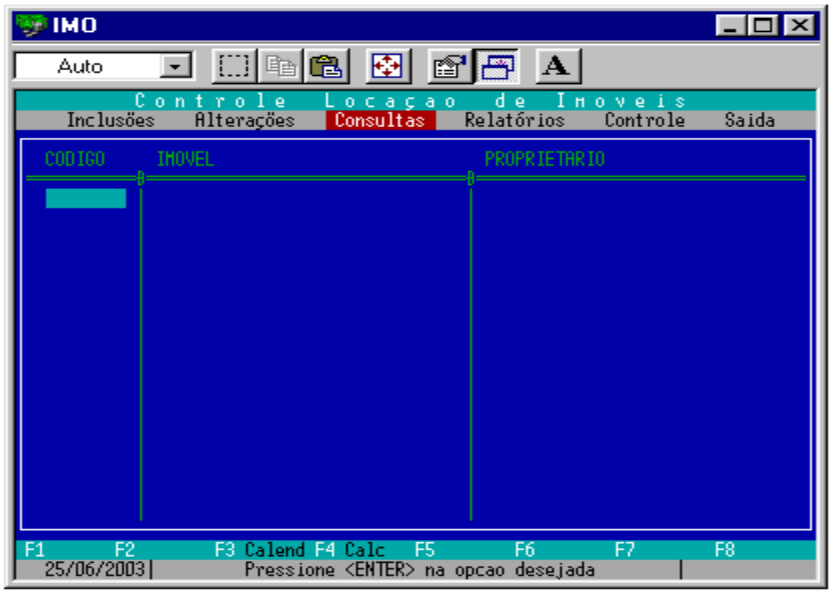

Sistema para Controle de Locações de Imóveis
Inclusões Alterações Consultas Relatórios Controle Sair
Consultas de Locadores
Código Nome Endereço Número Bairro
1 Mírian Cristina de Oliveira R: Bernardo Guimarães 15 São José
2 Lucas Miguel da Costa R: Bernardo Guimarães 09 São José
Calendário Calculadora Imprimir Voltar

FIGURA 10: TELA CONSULTAS IMÓVEIS ALUGADOS:

É aberta uma tela contendo todos os dados dos imóveis, a partir do arquivo Imóvel.dbf. Todos os imóveis são ordenados pela sua chave primária.



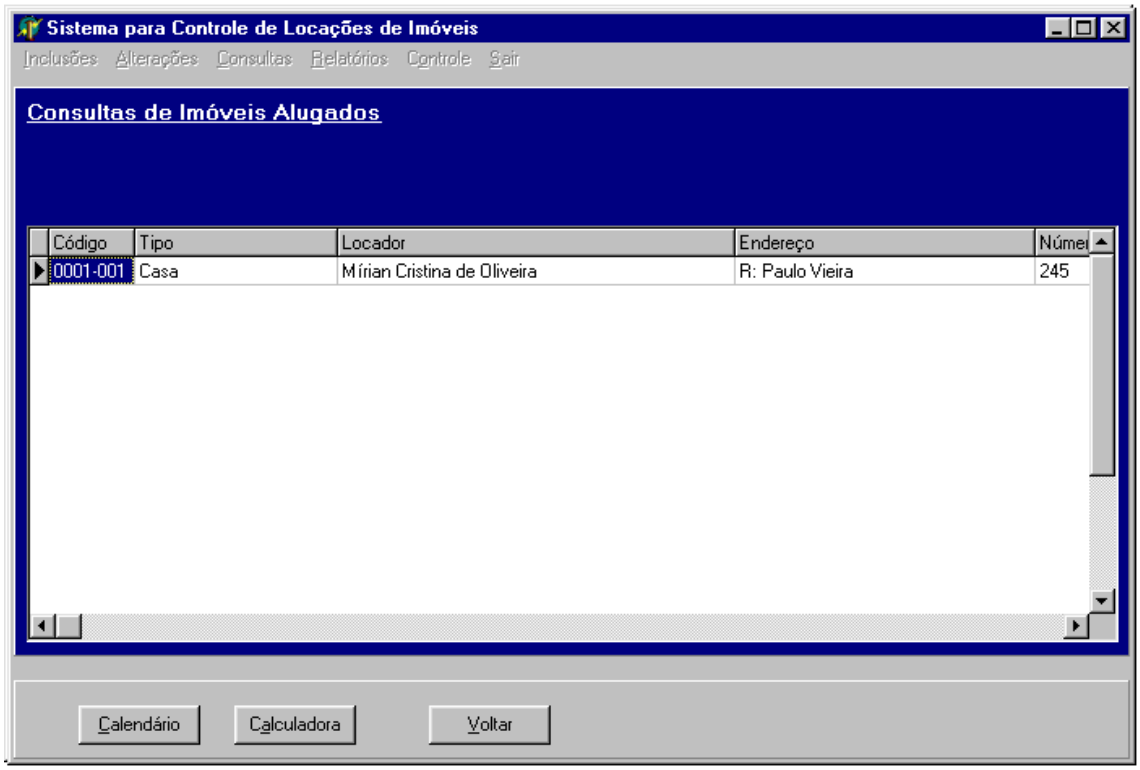


FIGURA 11**: TELA CONSULTAS IMÓVEIS DISPONÍVEIS:**

Uma tela idêntica à de consultas a imóveis alugados é aberta,só que dessa vez o sistema busca no arquivo Imovel.dbf os registros referente aos imóveis que não estão alugados. Todos os imóveis são ordenados pela sua chave primária.

FIGURA 12**: TELA CONSULTAS LOCATÁRIOS:**

É aberta uma tela contendo todos os dados dos locatários, existentes no arquivo Locatário.dbf. Os locatários estão ordenados pelo código e o código de um locatário que teve seu registro excluído, nunca é reaproveitado.

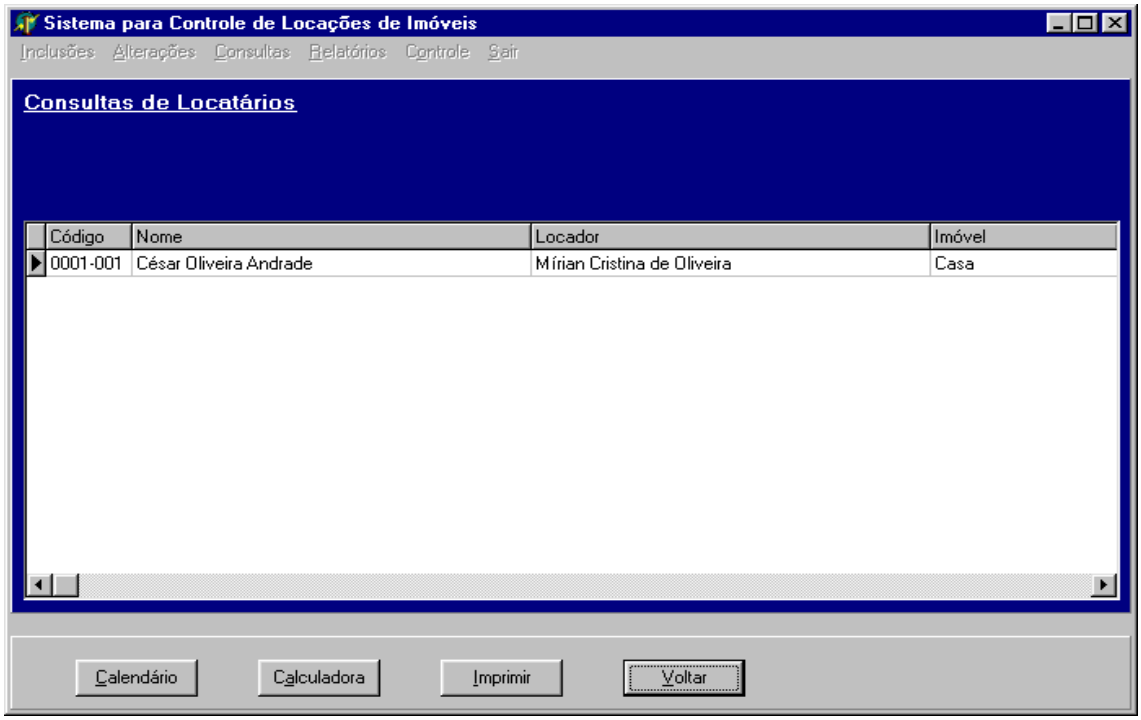
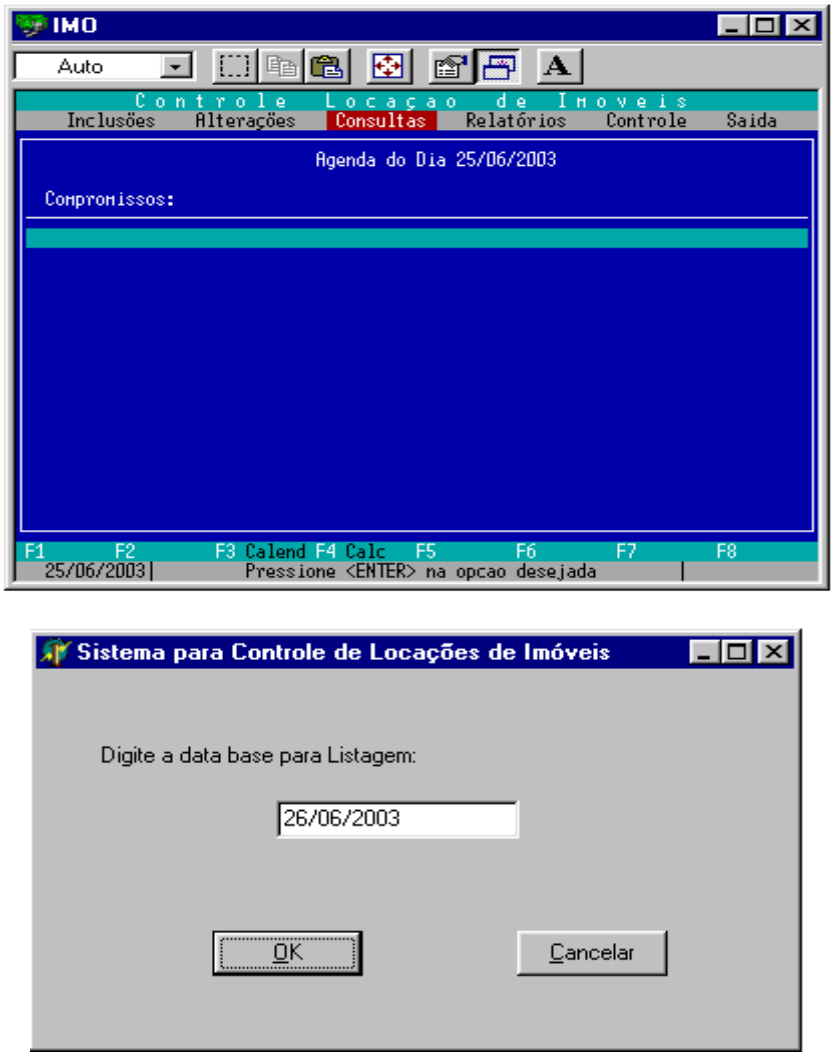


FIGURA 13: TELA CONSULTAS AGENDA DEPÓSITOS:

O sistema solicita antes ao usuário uma data base para listagem. Na figura, a data digitada foi 25/06/03. O campo de Compromissos mostra o nome dos locadores que possuem depósitos marcados para o dia e o valor que eles possuem para receber.



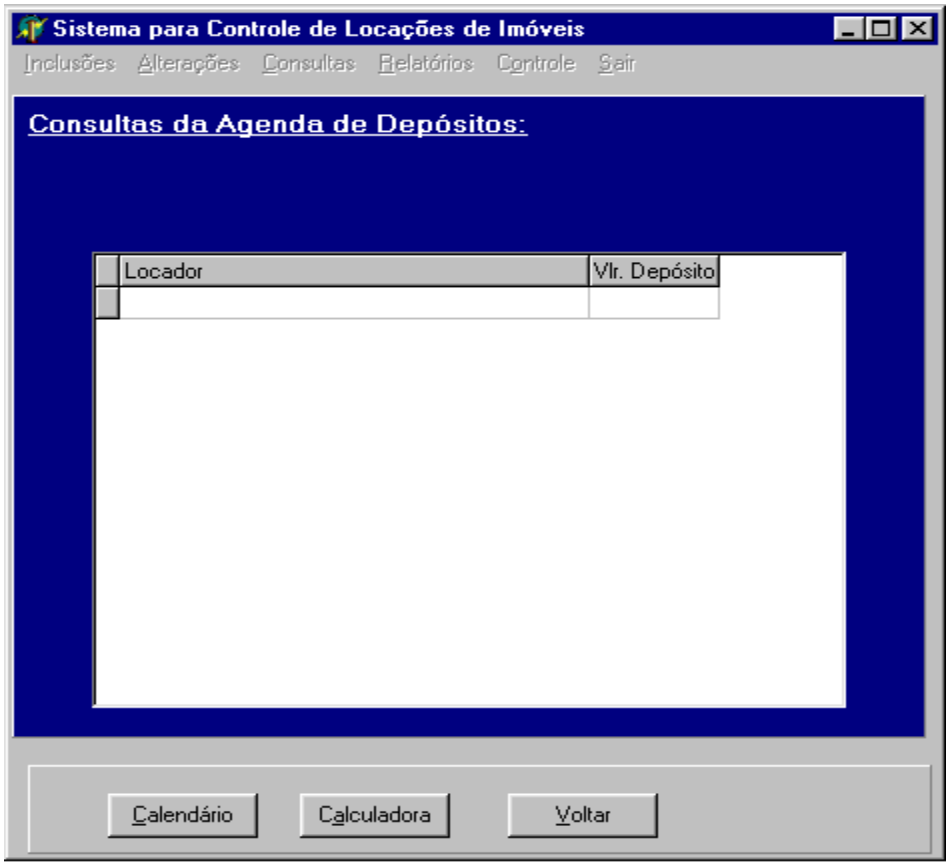
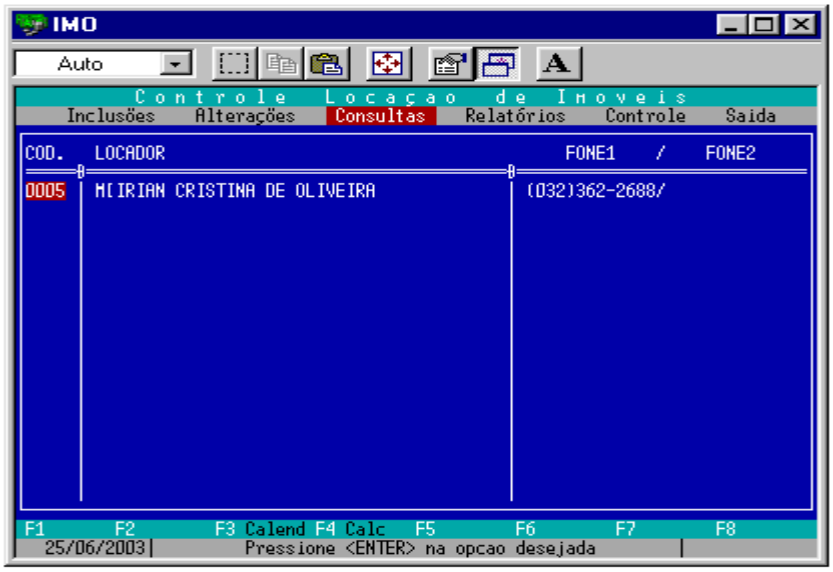


FIGURA 14: TELA CONSULTAS ANIVERSARIANTES DO MÊS:

O sistema solicita antes ao usuário o mês desejado. Na figura, o mês digitado foi 02. A partir daí, procura no arquivo Locador.dbf (só possui esta funcionalidade para locadores), os registros cujo mês coincide com o digitado pelo usuário. Na tela, o sistema mostra todos os dados dos locadores selecionados. Os locadores são ordenados pelo seu código.


IMO
Auto
Controle Locaçao de Imoveis
Inclusões Alterações Consultas Relatórios Controle Saida
COD. LOCADOR FONE1 / FONE2
0005 MIRIAN CRISTINA DE OLIVEIRA (032)362-2688/
F1 F2 F3 Calend F4 Calc F5 F6 F7 F8
25/06/2003 Pressione <ENTER> na opcao desejada

Sistema para Controle de Locações de Imóveis
Digite o mês desejado:
02
OK
Cancelar

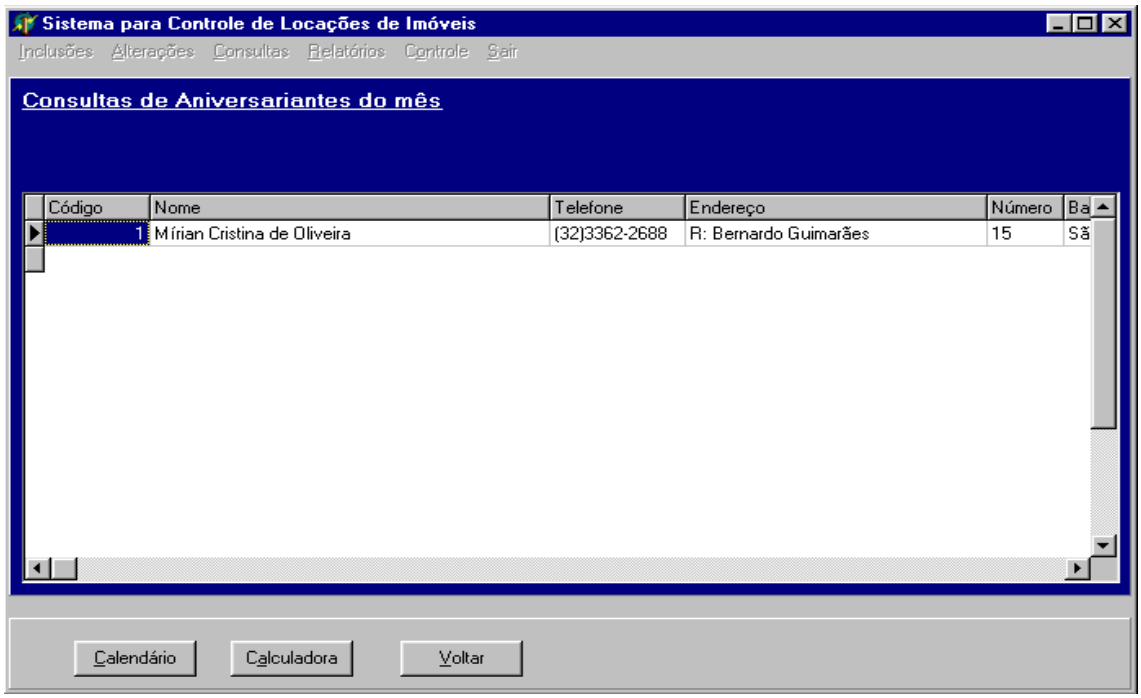
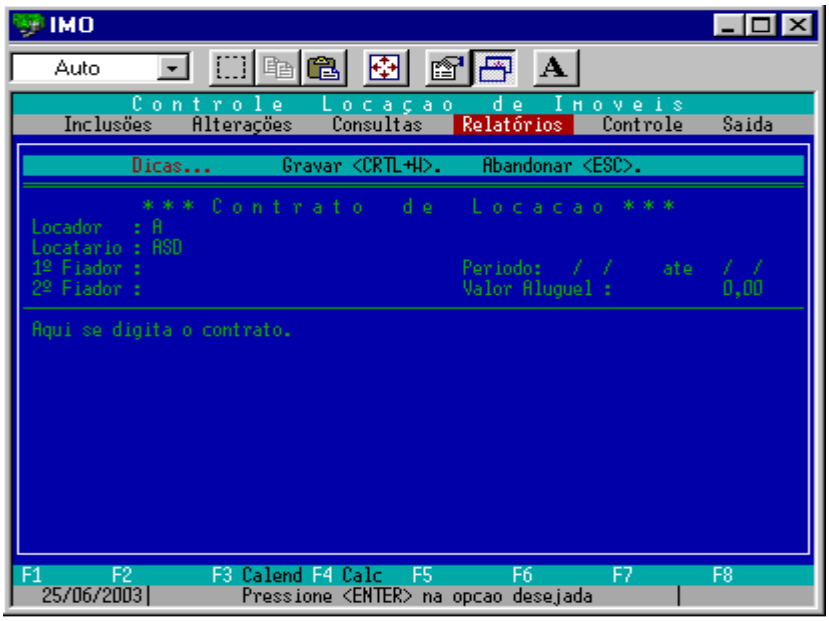

Sistema para Controle de Locações de Imóveis
Inclusões Alterações Consultas Relatórios Controle Sair
Consultas de Aniversariantes do mês
Código Nome Telefone Endereço Número Ba
1 Mírian Cristina de Oliveira (32)3362-2688 R: Bernardo Guimarães 15 Sã
Calendário Calculadora Voltar

FIGURA 15: TELA CONTRATOS:

Após escolhido o submenu, não importando qual tipo de contrato o usuário escolheu, o sistema abrirá uma tela com o nome de todos os locatários, ordenados pelo seu código. Assim que o usuário escolher o locatário desejado, sistema abre a tela contento o registro do contrato escolhido, a partir do arquivo Contrato.dbf, e no cabeçalho, os dados do locatário.



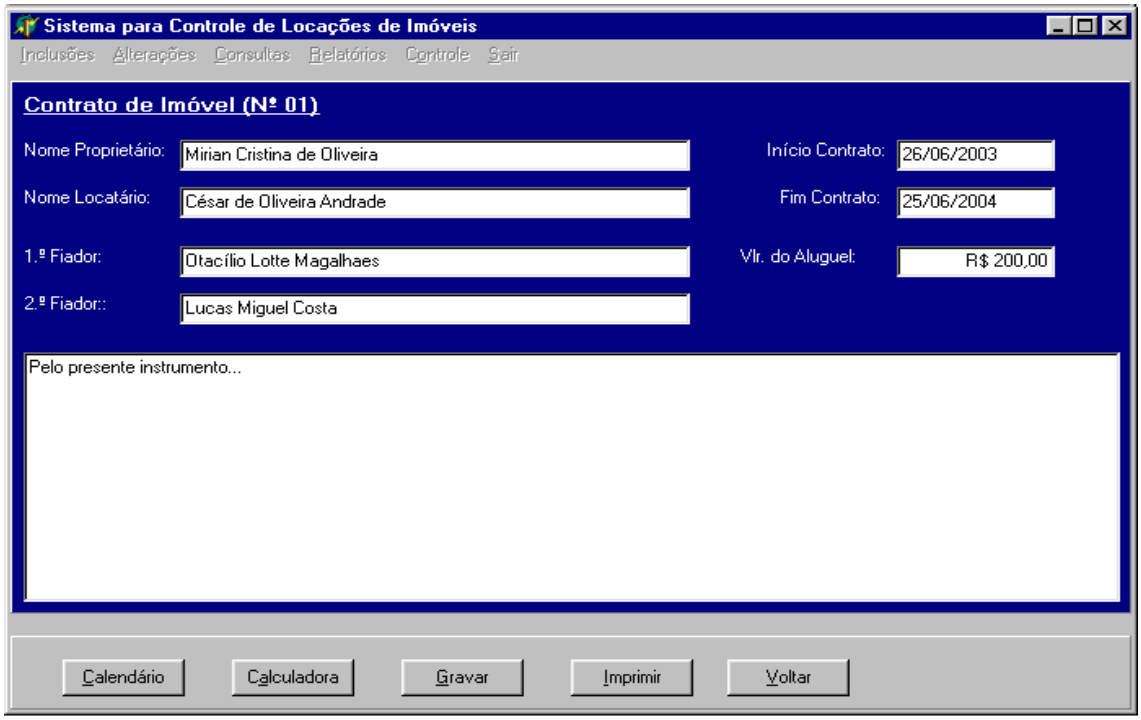
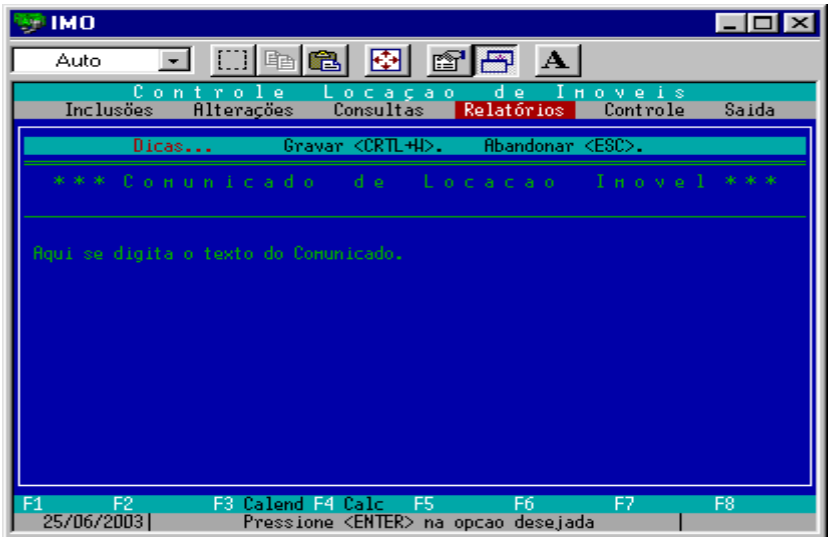


FIGURA 16**: TELA COMUNICADOS:**

Se usuário escolheu os submenus de comunicado, também não importando qual tipo de comunicado, o sistema abrirá uma tela com o nome de todos os locatários, ordenados pelo seu código. Assim que o usuário escolher o locatário desejado, sistema abre a tela contento o registro do comunicado escolhido, a partir do arquivo Contrato.dbf, e no cabeçalho, os dados do locador.



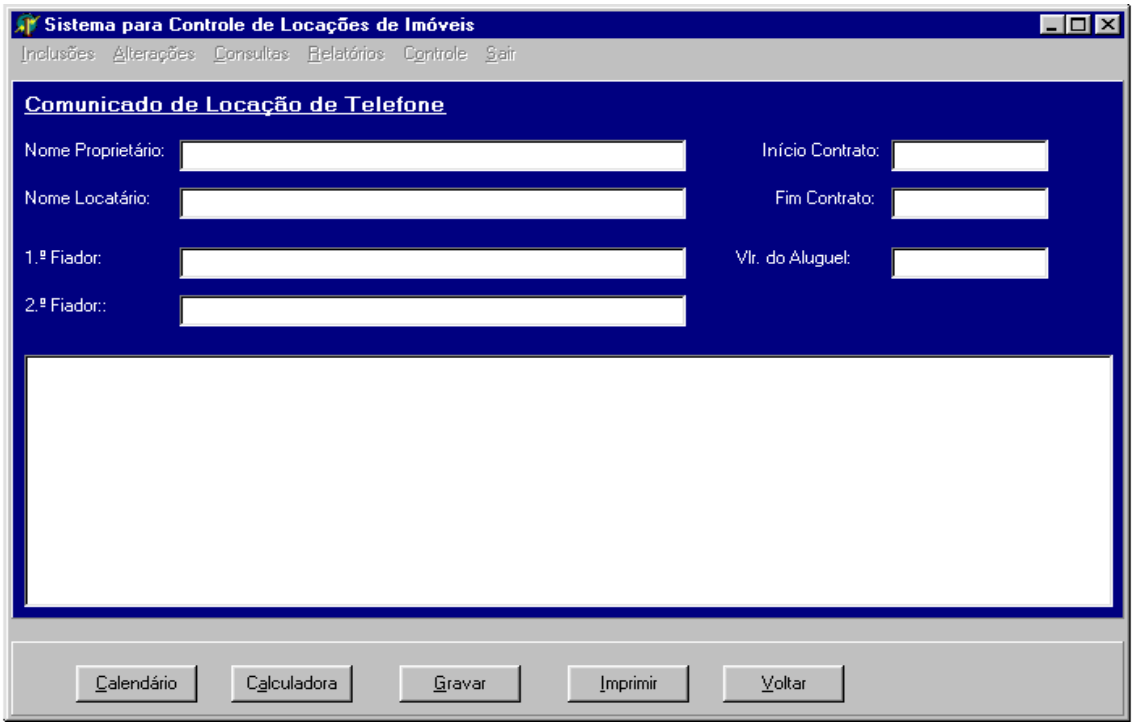
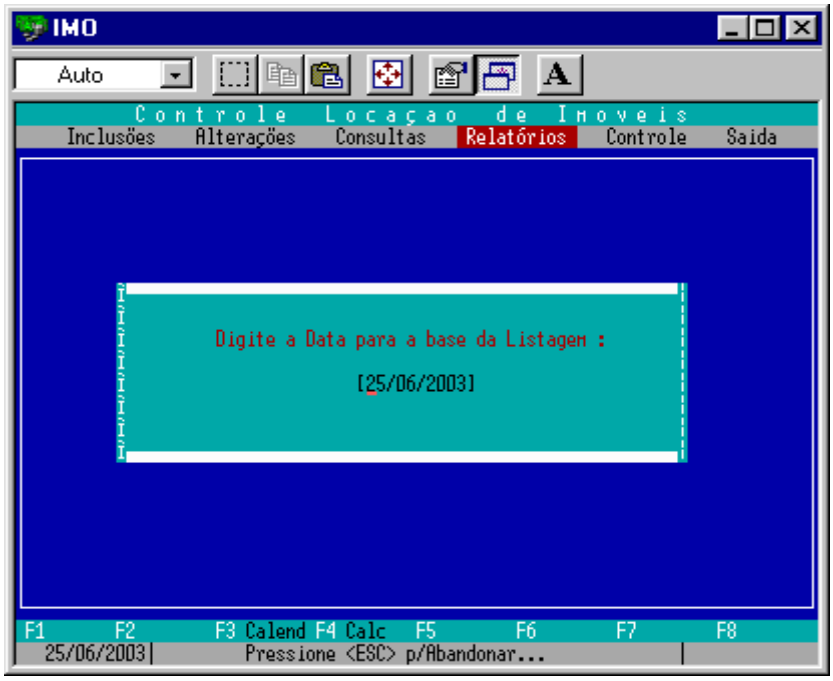


FIGURA 17**: TELA LOCATÁRIOS EM DÉBITO:**

O sistema pede ao usuário apenas a data base para listagem e já imprime a lista direto. Não a mostra antes na tela.



Sistema para Controle de Locações de Imóveis
Digite a data base para Listagem:
26/06/2003
OK
Cancelar

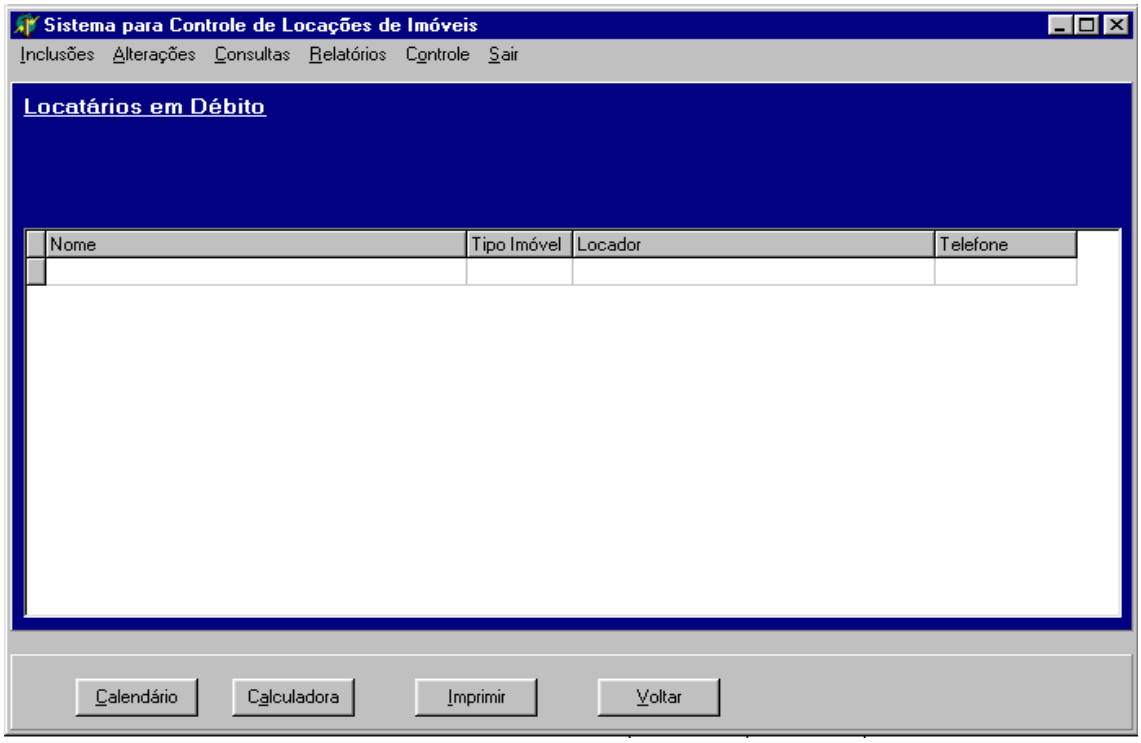
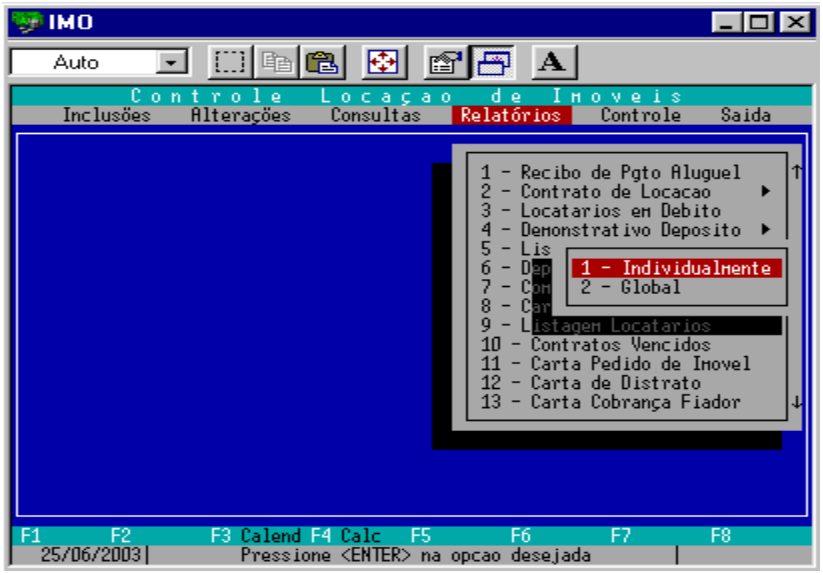
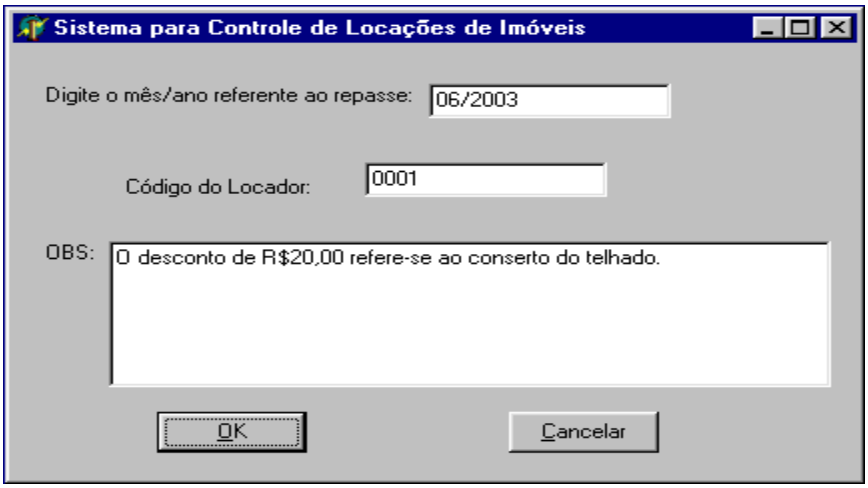

Sistema para Controle de Locações de Imóveis
Inclusões Alterações Consultas Relatórios Controle Sair
Locatários em Débito
Nome
Tipo Imóvel
Locador
Telefone
Calendário
Calculadora
Imprimir
Voltar

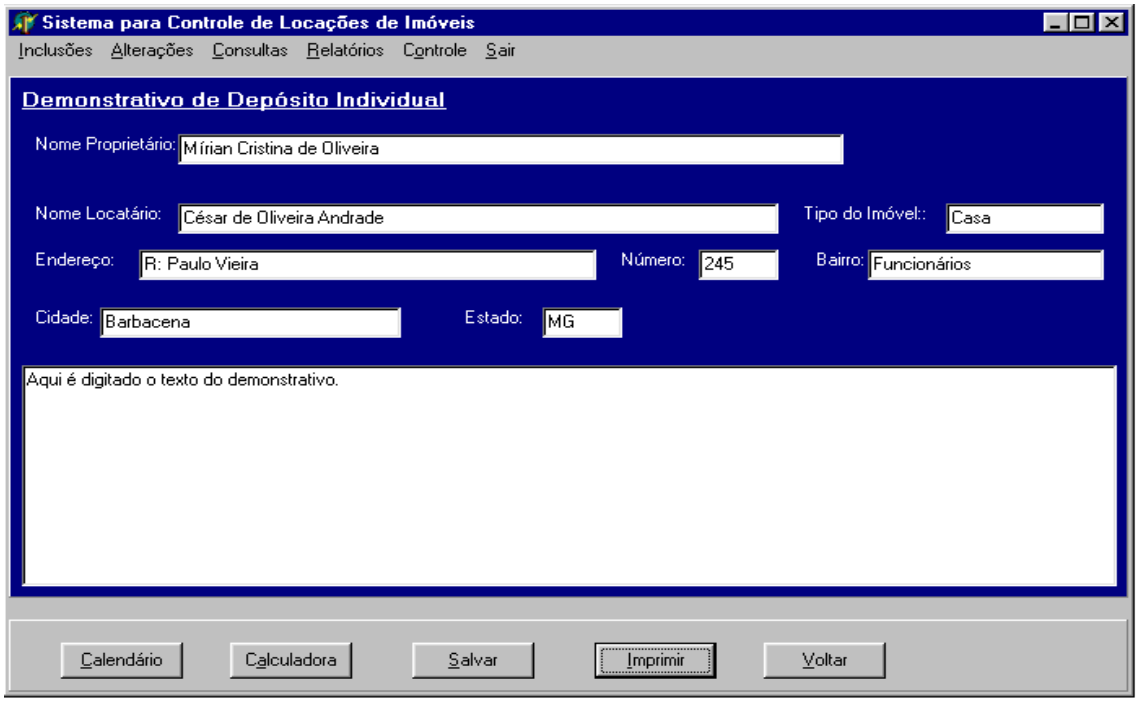
FIGURA 18**: TELA DEMONSTRATIVO DE DEPÓSITO (SUBMENU):**

Nessa tela, se usuário escolher individualmente, o sistema pedirá o mês referente ao repasse, o código do locador e exibirá um campo para observações, se o usuário escolher global o sistema exibirá o campo do mês do repasse e o campo para observações, e imprime diretamente, sem mostrar antes o relatório na tela, o demonstrativo de depósito.



Demonstrativo individual





Demonstrativo Global

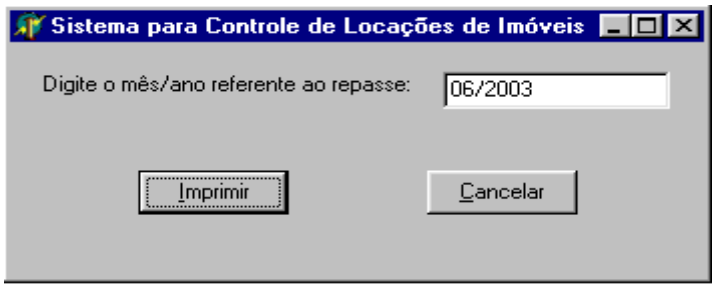


FIGURA 19**: TELA LISTAGEM DOS LOCADORES:**

O sistema apresenta apenas uma janela perguntando se usuário deseja confirmar a impressão. Não exibe este relatório na tela.Dessa forma, no novo sistema foi adicionado ao menu Consultas Locadores um botão “Imprimir” que permite ao usuário obter um relatório.

FIGURA 20**:TELA DEPÓSITOS NÃO REALIZADOS:**

Da mesma forma, O sistema apresenta apenas uma janela perguntando se usuário deseja confirmar a impressão. Não exibe este relatório na tela.

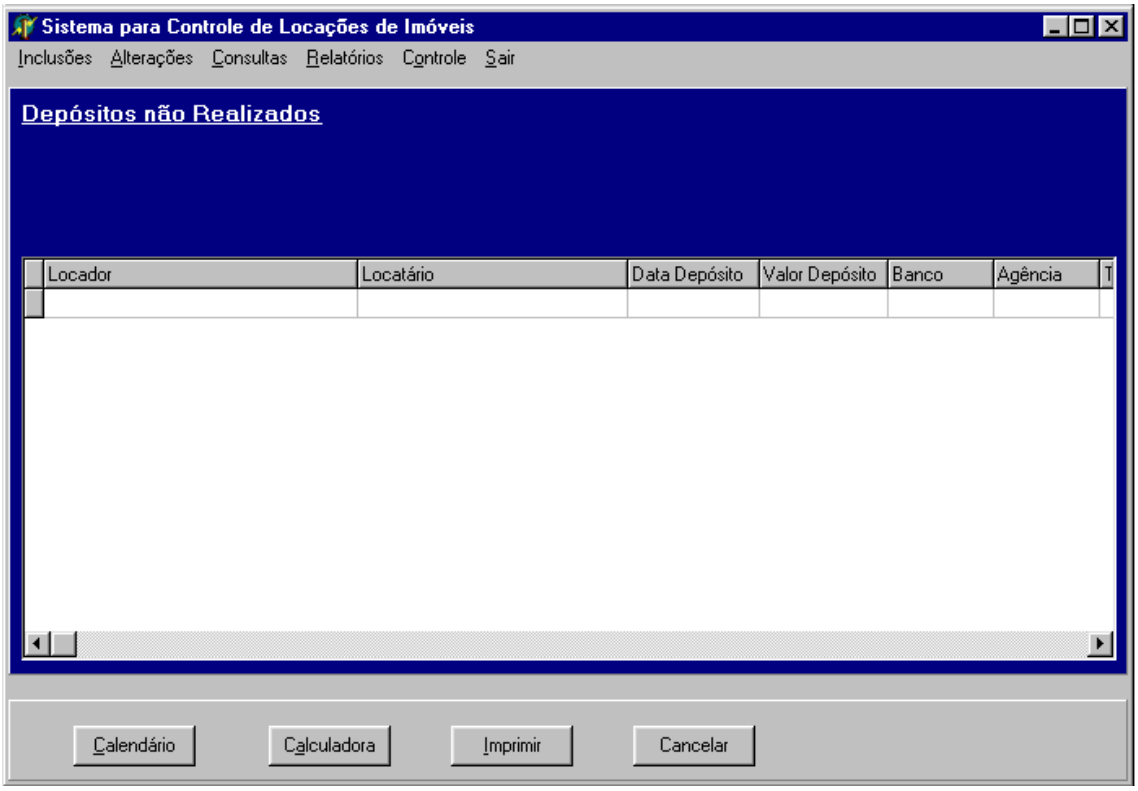
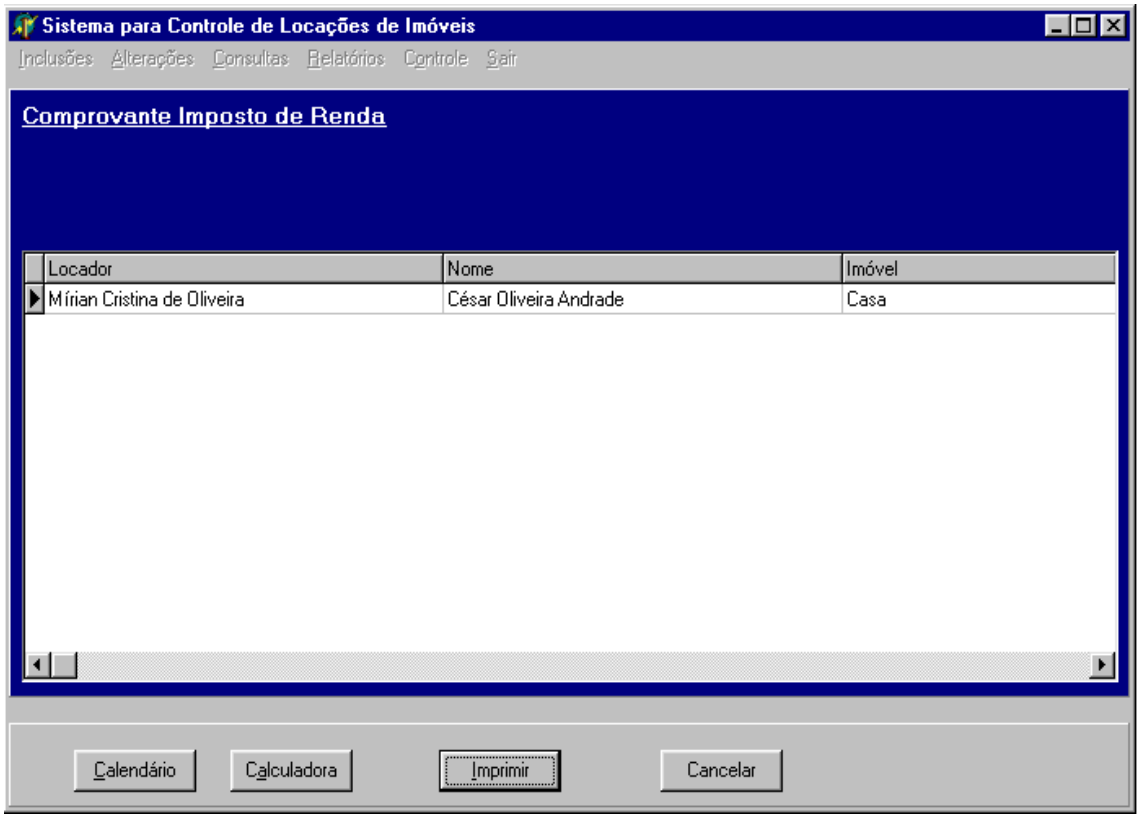


FIGURA 21**: TELA COMPROVANTE IMPOSTO DE RENDA:**

Sistema exibe uma tela com o nome de todos os proprietários, ordenados por ordem alfabética, nome de cada um de seus inquilinos, e o tipo do imóvel alugado. Usuário escolhe para qual imóvel irá tirar o IR, e o sistema pedirá o número de cópias e o ano base para declaração. Após essas ações, sistema imprime o comprovante.



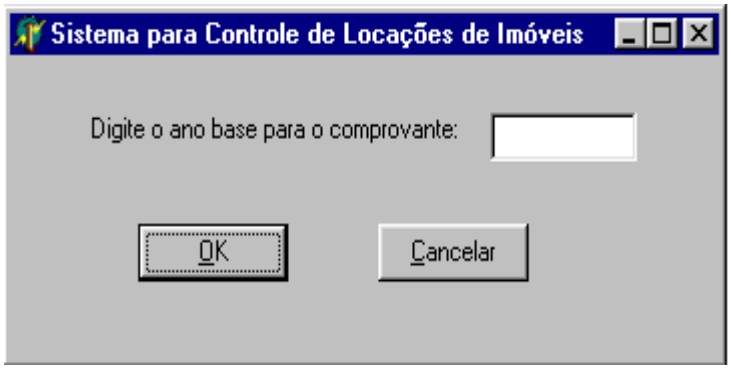
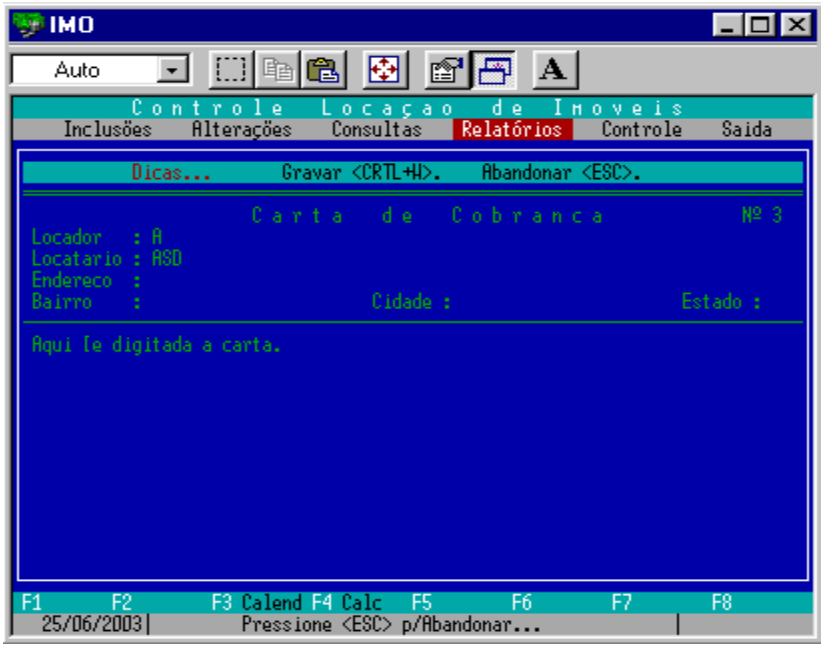


FIGURA 22**: TELA CARTA COBRANÇA LOCATÁRIO:**

Independente do número da carta escolhida, o sistema exibirá uma tela com o nome de cada locatário, por ordem alfabética e o tipo do imóvel alugado por ele. Usuário escolhe e o sistema busca no arquivo Cobrança.dbf a carta correspondente à escolhida.



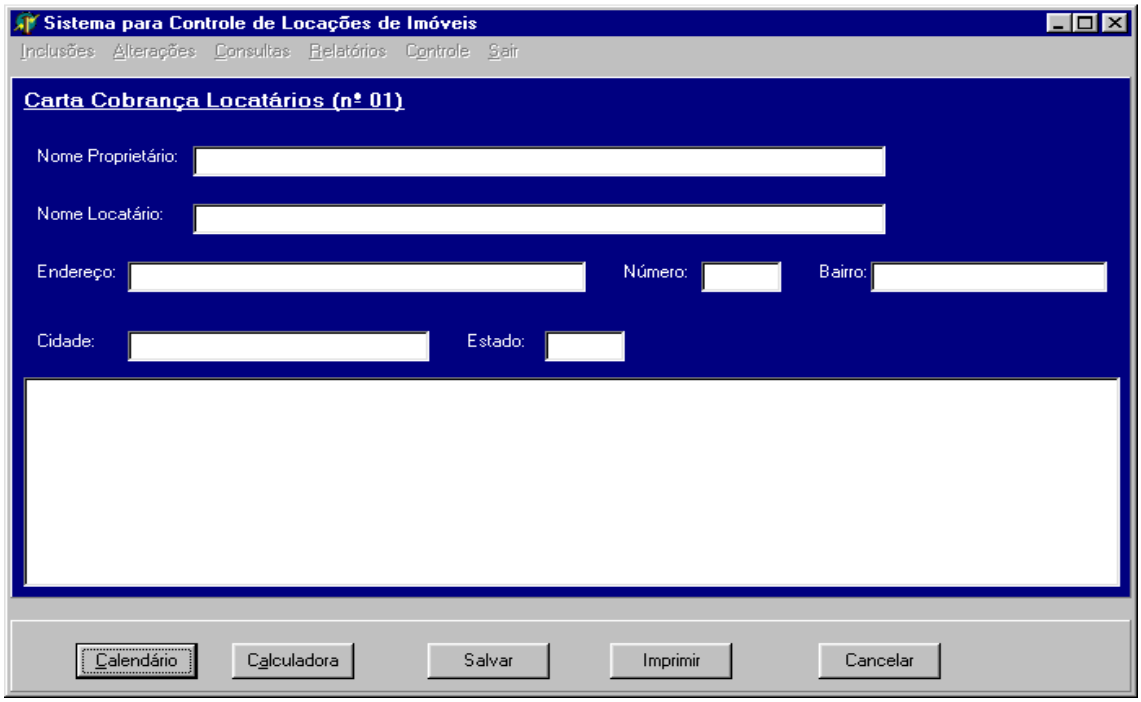


FIGURA 23**: TELA LISTAGEM LOCATÁRIOS:**

O sistema apresenta apenas uma janela perguntando se usuário deseja confirmar a impressão. Não exibe este relatório na tela. Dessa forma, na nova interface foi incluído um botão “Imprimir”, na tela do menu consultas locatários.

FIGURA 24**: TELA CONTRATOS VENCIDOS:**

O sistema apresenta apenas uma janela perguntando se usuário deseja confirmar a impressão. Não exibe este relatório na tela. O novo sistema exibe o relatório antes de imprimi-lo.

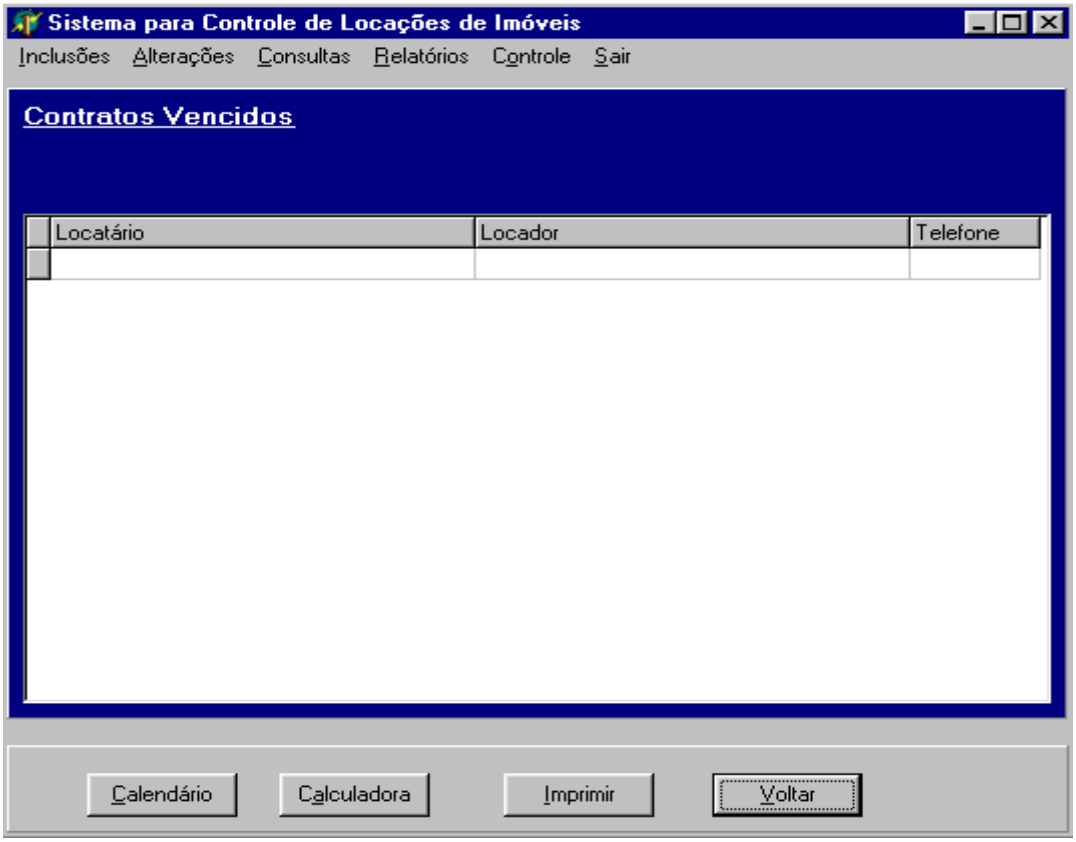
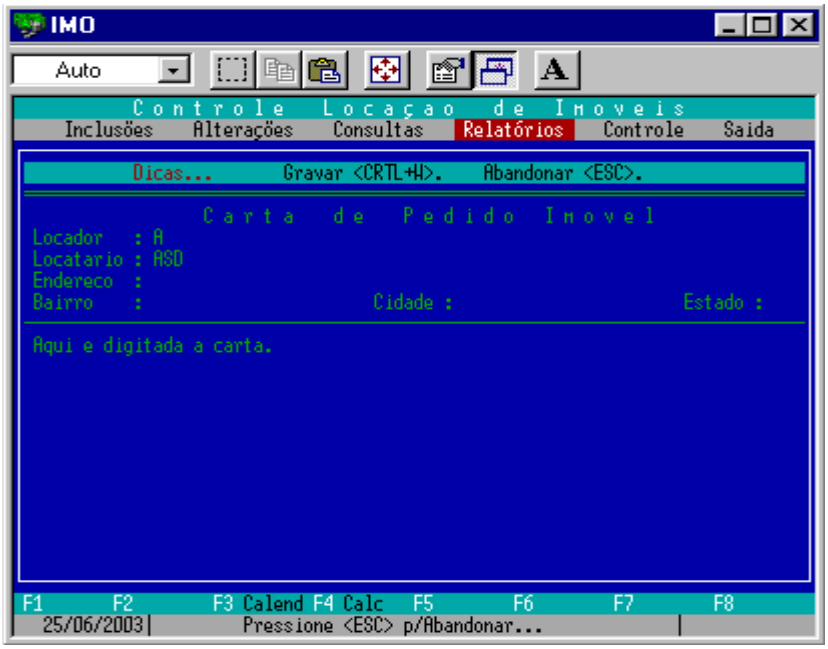


FIGURA 25**: TELA CARTA PEDIDO DE IMÓVEL:**

O sistema exibe uma tela com o nome de cada locatário, por ordem alfabética e o tipo do imóvel alugado por ele. Usuário escolhe e o sistema busca no arquivo Pedido.dbf a carta correspondente à escolhida.



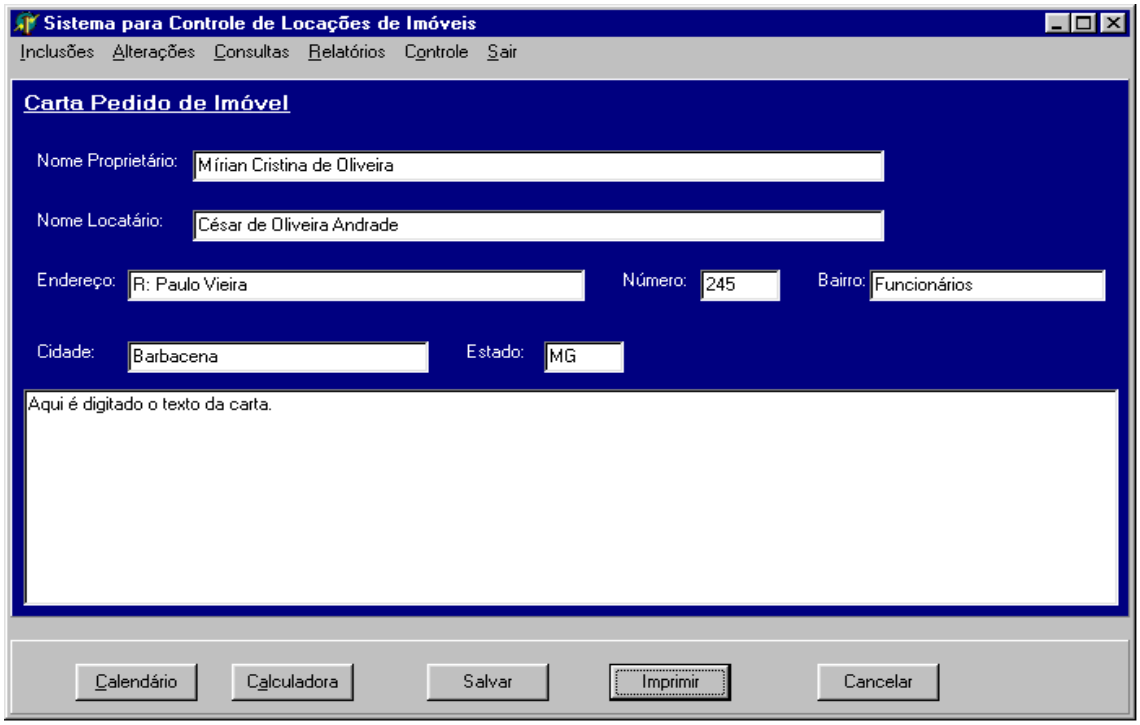


FIGURA 26**: TELA CARTA DE DISTRATO:**

O sistema exibe uma tela com o nome de cada locatário, por ordem alfabética e o tipo do imóvel alugado por ele. Usuário escolhe e o sistema busca no arquivo Pedido.dbf a carta correspondente à escolhida.

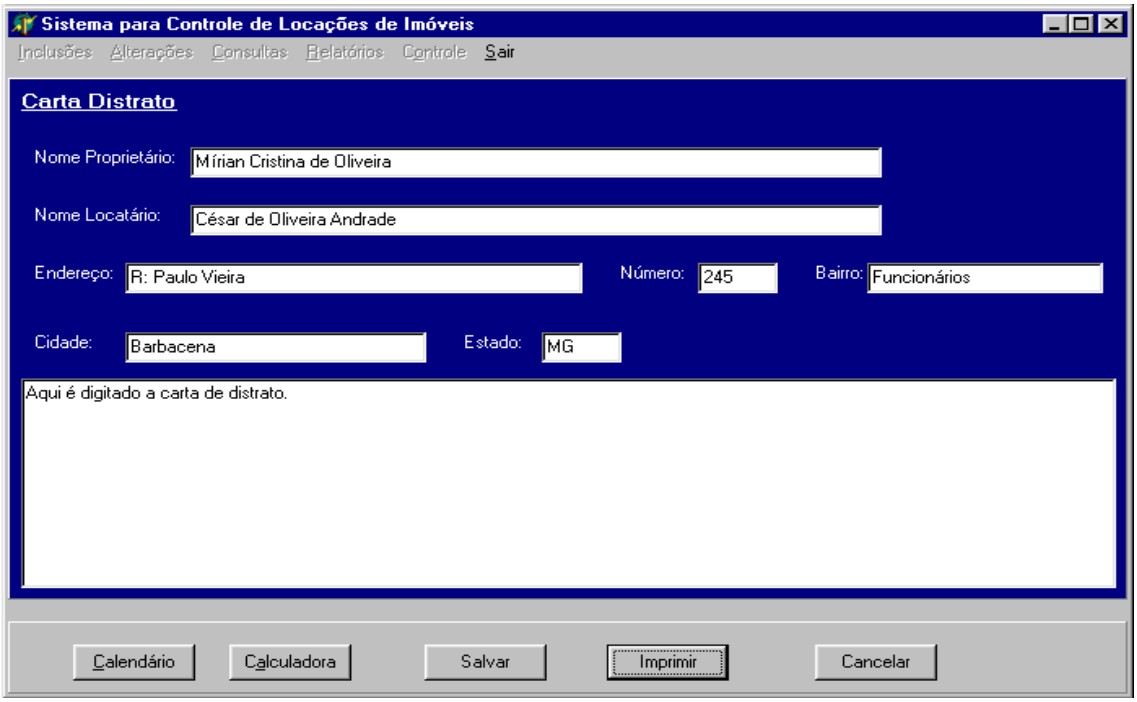
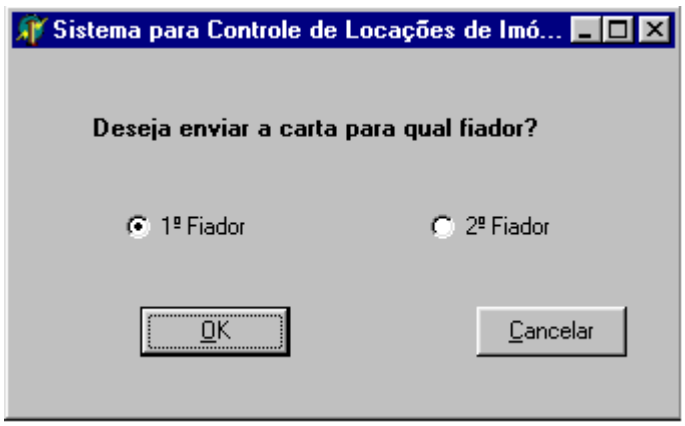


FIGURA 27**: TELA CARTA COBRANÇA FIADOR:**

O sistema exibe uma tela com o nome de cada locatário, por ordem alfabética e o tipo do imóvel alugado por ele. Usuário escolhe e o sistema busca no arquivo Pedido.dbf a carta correspondente à escolhida.Após escolhido, sistema abre outra janela perguntando o

número de cópias a ser impressa, e depois, uma outra janela perguntando se a carta será impressa para o 1.º ou 2.º fiador. Após responder estas perguntas, sistema pergunta se deseja confirmar a impressão e já imprime diretamente a carta, sem antes mostrá-la na tela. No novo sistema a carta é exibida na tela para possíveis alterações.



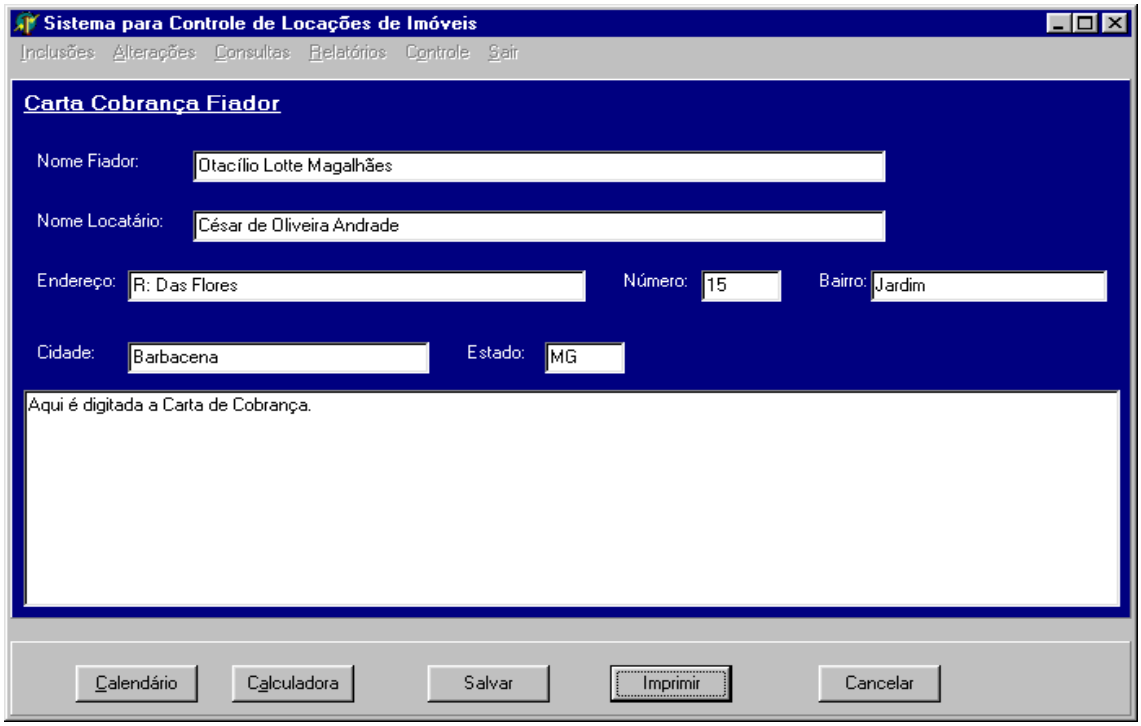


FIGURA 28: EMISSÃO ETIQUETA INQUILINO:

Imprime uma etiqueta contendo o nome do inquilino e seu endereço, para poder ser colocados nas cartas.

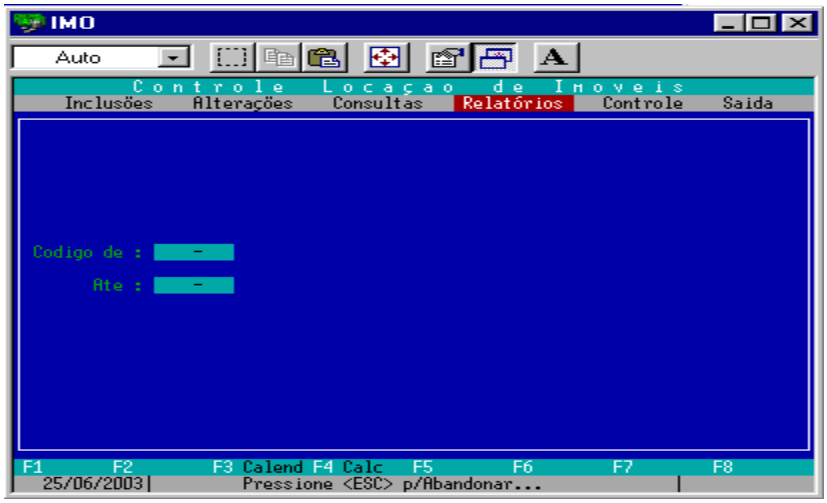


FIGURA 29**: EMISSÃO ETIQUETA PROPRIETÁRIO:**

Imprime uma etiqueta contendo o nome do proprietário e seu endereço, para poder ser colocados nas cartas.

FIGURA 30: TELA ORGANIZAR ARQUIVOS:

Essa funcionalidade foi retirada do novo sistema pois, visto que seu banco de dados, se implementado, seria feito em interbase, o próprio gerenciador do BD poderia fazer a organização dos arquivos, sem que o sistema necessite de uma funcionalidade para isto.

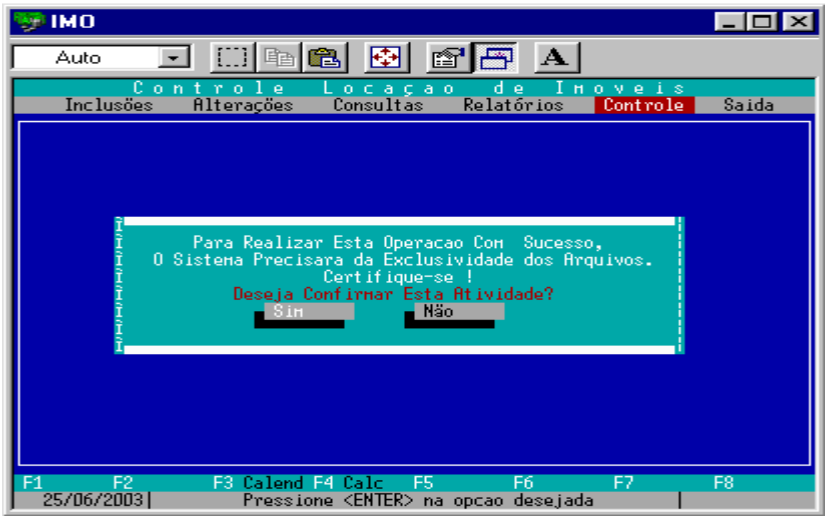
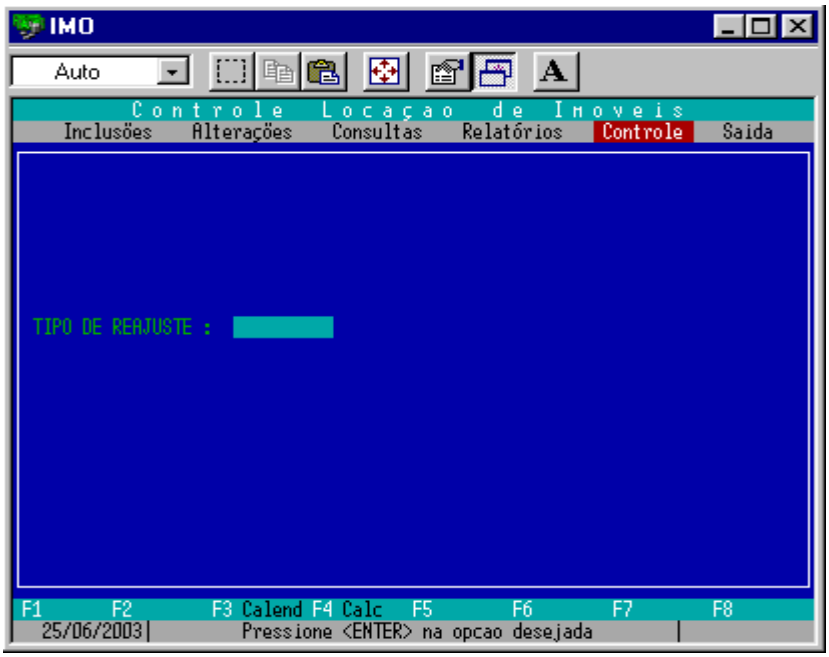



FIGURA 31: TELA TIPO DE REAJUSTE: