CENTRO UNIVERSITÁRIO FACVEST
CURSO DE CIÊNCIAS DA COMPUTAÇÃO
GABRIEL PEREIRA RODRIGUES

PEST CONTROL - SISTEMA PARA CONTOLE DE PRAGAS

LAGES
2013

GABRIEL PEREIRA RODRIGUES

PEST CONTROL - SISTEMA PARA CONTOLE DE PRAGAS

Trabalho de conclusão de curso, apresentado ao Centro Universitário UNIFACVEST, como pré-requisito para obtenção do título de Bacharel em Ciências da Computação.

Orientador: Prof. MSc. Márcio José Sembay

LAGES

2013

GABRIEL PEREIRA RODRIGUES

PESTM CONTROL – SISTEMA PARA CONTROLE DE PRAGA.

Trabalho de conclusão de curso, apresentado ao Centro Universitário UNIFACVEST, como pré-requisito para obtenção do título de Bacharel em Ciências da Computação.

Lages SC ____/____/2013 Nota_____ _____

Prf. MSc. Márcio José Sembay - Orientador

_____

Márcio José Sembay (Coordenador do Curso de Ciências da Computação)

LAGES

2013

AGRADECIMENTOS

Agradeço em primeiro lugar Deus, que por anos tem ouvido minhas orações pedindo a realização deste grande sonho.

A minha esposa pela grande oportunidade que me foi dada através de sua bondade e por acreditar na minha competência, na minha capacidade em desenvolver um programa de *software*.

Aos professores que transmitiram seus conhecimentos e ofereceram sugestões para o meu aprimoramento como profissional.

E a todos que de uma forma ou outra contribuíram para a realização desta trabalho.

Muito obrigado!

PEST CONTROL - SISTEMA PARA CONTOLE DE PRAGAS

Gabriel Pereira Rodrigues[1]
Márcio José Sembay[2]

RESUMO

No atual contexto competitivo, a informação é um grande diferencial, portanto as organizações têm investido em sistemas de informação, cujos mesmos, devem ser totalmente aderentes aos processos de negócio da organização, e essa adesão só é obtido se houver uma correta integração entre usuários e desenvolvedores. O passo fundamental para o sucesso, em qualquer modelo de desenvolvimento de software é a definição e análise de requisitos, pois mesmo se o sistema estiver bem projetado e codificado, porém mau especificado, com certeza, irá causar danos e aborrecimentos para os envolvidos. Nesse sentido, o objetivo deste estudo foi o de desenvolver um programa de software com informações direcionadas ao controle de pragas nas fazendas da região da Serra Catarinense. O curso de Ciência da Computação do Centro Universitário UNIFACVEST, permitiu a nós acadêmicos no processo de desenvolvimento de sistemas de informação e definir os processos e atividades oferecidas nas aulas teóricas.

Palavras- chave: Programação. Ciências da Computação. Desenvolvimento de Software.

[1] Acadêmico do Curso de Ciências da Computação do Centro Universitário UNIFACVEST

[2] Professor do Curso de Ciências da Computação do Centro Universitário UNIFACVEST.

PEST CONTROL - SISTEMA PARA CONTOLE DE PRAGAS

Gabriel Pereira Rodrigues[3]
Márcio José Sembay[4]

ABSTRACT

In today's competitive environment, information is a big difference, so companies have invested in IT, whose same systems should be fully adherent to the organization's business processes, and that membership is earned only if a correct integration between users and developers. The key step to success in any model of software development is the definition and requirements analysis, because even if the system is well designed and coded, but bad Specified surely will cause damage and hassles for those involved. Accordingly, the aim of this study was to develop a software program with information aimed at the control of pests in farms of the Sierra Santa Catarina State . The course of Computer Science of the University Center UNIFACVEST allowed us academics in the development of information systems process and define processes and activities offered in the lectures.

Keywords: Programming. Computer Sciences. Software Development.

[3] Acadêmico do Curso de Ciências da Computação do Centro Universitário UNIFACVEST.
[4] Professor do Curso de Ciências da Computação do Centro Universitário UNIFACVEST.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

A revista Business Week, no inicio da década de 1980, apresentou a seguinte manchete: "*Software*: A Nova Força Propulsora". O *software* amadurecera - tornara-se um tema de preocupação administrativa. Em meados da década de 1980, uma reportagem de capa da Fortune lamentava "Uma Crescente Defasagem de Software" e, ao final da década, a Business Week avisava os gerentes sobre a "Armadilha do Software - Automatizar ou Não?" . No começo da década de 1990, uma reportagem especial da *Newsweek* perguntava: "Podemos Confiar em Nosso Software?" enquanto o *Wall Street Journal* relacionava as "dores de parto" de uma grande empresa de software com um artigo de primeira página intitulado "Criar Software Novo: Era Uma Tarefa Agonizante..." . Essas manchetes, e muitas outras iguais a elas, eram o anuncio de uma nova compreensão da importância do software de computador - as oportunidades que ele oferece e os perigos que apresenta.

O software agora ultrapassou o hardware como a chave para o sucesso de muitos sistemas baseados em computador. Seja o computador usado para dirigir um negocio, controlar um produto ou capacitar um sistema, o software é um fator que diferencia. A inteireza e a oportunidade. das informações oferecidas pelo software (e bancos de dados relacionados) diferenciam uma empresa de suas concorrentes. O projeto e a capacidade de ser "amigável ao ser humano" de um produto de software diferenciam-no dos produtos concorrentes que tenham função idêntica em outros aspectos. A inteligência e a função oferecidas pelo software muitas vezes diferenciam dois produtos de consumo ou industrias idênticas. E o software que pode fazer a diferença.

1.1 JUSTIFICATIVA

Durante as três primeiras décadas da era do computador, o principal desafio era desenvolver um hardware que reduzisse o custo de processamento e armazenagem de dados. Ao longo da década de 1980, avanços na microeletrônica resultaram em maior poder de

computação a um custo cada vez mais baixo. Hoje, o problema é diferente. O principal desafio durante a década de 2013 é melhorar a qualidade (e reduzir o custo) de soluções baseadas em computador - soluções que são implementadas com software.

A engenharia de software possibilita aos cientistas da computação os passos necessários para o desenvolvimento de um sistema. Bauer (1969) *apud* Pressman (2006, p.17) afirma que "[Engenharia de software é] a criação e a utilização de sólidos princípios de engenharia a fim de obter softwares econômicos que sejam confiáveis e que trabalhem eficientemente em máquinas reais".

O uso do computador esta aumentando em todos os campos de trabalho. Em uma era de custos sempre crescentes os custos da computação têm diminuído substancialmente, em virtude de rápidos desenvolvimentos na tecnologia de *hardware e software*. Os computadores que ocupavam enormes salas e custavam milhões de dólares há apenas duas décadas agora são feitos na superfície de chips de silício menores do que uma unha, custando poucos dólares por unidade.

A tecnologia do chip de silício barateou a computação de tal maneira que centenas de milhões de computadores de propósito geral estão em uso no mundo todo, ajudando as pessoas no comércio, na indústria, na agricultura, no governo e em uso particular. Dada a taxa corrente de desenvolvimento tecnológico, esse número pode facilmente dobrar nos próximos anos.

1.2 IMPORTÂNCIA

O poder de um computador da década de 1980 agora está a disposição sobre uma mesa. As assombrosas capacidades de processamento e armazenagem do moderno *hardware* representam um grande potencial de computação. O *software* é o mecanismo que nos possibilita aproveitar e dar vazão a esse potencial.

Com o avanço tecnológico as empresas tendem cada vez mais utilizar sistemas de informação automatizando assim seus serviços. Softwares agilizam os processos em vários setores das empresas que o utilizam mas, os mesmos precisam estar em constante melhoramento para atender as necessidades dos usuários.

O programa Pest Control – Sistema para controle de pragas é um bando de dados, que pode ser visto como o equivalente eletrônico de um armário de arquivamento. Em outras palavras, é um repositório ou recipiente para uma coleção de arquivos de dados

computadorizados. Os usuários do sistema poderão executar diversas operações sobre tais arquivos.

1.3 OBJETIVO

1.3.1 Objetivo Geral

Desenvolvimento de um software para controle de pragas para facilitar o trabalho dos proprietários das fazendas no controle de pragas em seus pomares.

1.3.2 Objetivos Específicos

Os objetivos específicos consistem no desenvolvimento de um sistema com as seguintes funcionalidades:

α) Geração de um programa na linguagem C#.

β) Geração de um software que seja capaz de ajudar na tomada de decisão em seu dia a dia.

1.4 METODOLOGIA

1.4.1 Estudo de Caso

Para o desenvolvimento do trabalho foi realizada uma entrevista com um gerente de uma fazenda de plantio de maçã que tinha a necessidade de um programa de software para o controle de pragas, foi elaborado um projeto para levantar os requisitos básicos do software. Após esta etapa o gerente informou quais programas a fazenda já utilizava e quais as facilidades que o sistema poderia proporcionar ao usuário.

O trabalho foi dividido em quatro etapas, citadas:

1) Levantamento dos *patterns* utilizados na empresa;

2) Acompanhamento das rotinas de programação;

3) Padronização da estrutura dos projetos;

1.4.2 Estudo bibliográfico

Para a realização deste trabalho foram pesquisados em livros e periódicos os conceitos de linguagem de programação, como implementá-los em sistemas legados e desacoplamento para utilização de ferramentas que facilitem a codificação de sistemas.

Realizou-se um estudo sobre engenharia de software para compreensão dos processos de software. Levou-se em consideração ergonomia, eficiência, eficácia para que os usuários possam utilizar o sistema desenvolvido evitando assim os trabalhos repetitivos.

1.4.3 Cronograma

Cronograma utilizado para a realização deste trabalho.

Atividades realizadas	**J**	**J**	**A**	**S**	**O**	**N**	**D**
Pesquisa	**X**	**X**					
Desenvolvimento e apresentação da pré-proposta		**X**					
Modelagem do sistema		**X**					
Implementação do software			**X**	**X**	**X**		
Fase de testes			**X**	**X**	**X**		
Entrega e defesa do TCC à banca avaliadora						**X**	
Correções						**X**	**X**
Entrega e defesa do TCC á banca avaliadora							**X**

1.5 ESTRUTURA DO TRABALHO

O presente trabalho foi desenvolvido da seguinte forma: pesquisa de material bibliográfico, revisão da literatura, modelagem do sistema, implementação do software e testes.

Na primeira fase foi realizado um levantamento bibliográfico, coletando os dados e informações necessários para o início do trabalho. Na revisão bibliográfica fundamentou-se teoricamente e proporcionou embasamento técnico, justificando os tópicos abordados no TCC e o padrão de processo adotado para o desenvolvimento do sistema. A modelagem do sistema

foi realizada com auxílio de ferramenta CASE. Foram feitas pesquisas em livros, periódicos, páginas da Web, trabalhos de conclusões de curso, entre outras modalidades.

Para implementação e testes do sistema foi utilizado o C#,banco de dados Sql Server e para a geração de relatórios o crystal reportes. A IDE utilizada para a codificação foi o Visual Studio 2010 premium.

Hardware utilizado:

- Notebook para implementação;
- Processador Intel Core2Due™ P7550, 2,27GHz;
- 2 GB de memória RAM;
- HD 500 GB.

2 REVISÃO DA LITERATURA

2.1 O PAPEL EVOLUTIVO DO SOFTWARE

O contexto em que o software foi desenvolvido está estreitamente ligado a quase cinco décadas de evolução dos sistemas computadorizados. O melhor desempenho de hardware, menor tamanho e custo mais baixo precipitaram o aparecimento de sistemas baseados em computadores mais sofisticados. Mudamos dos processadores a válvula para os dispositivos microeletrônicos que são capazes de processar 200 milhões de instruções por segundo. Em livros populares sobre a "revolução do computador", Osborne caracterizou uma "nova revolução industrial", Toffler chamou o advento da microeletrônica de "a terceira onda de mudança" na historia humana e Naisbitt previu que a transformação de uma sociedade industrial em uma "sociedade de informação" terá um profundo impacto sobre nossas vidas. Feigenbaum e McCorduck sugeriram que a informação e o conhecimento (controlados por computador) serão o foco principal de poder no século XXI, e Stoll argumentou que a "comunidade eletrônica" criada por redes e software e a chave para a troca de conhecimentos em todo o mundo. Quando se iniciava a década de 1990, Toffler descreveu uma "mudança de poder", em que as velhas estruturas de poder (governamental, educacional, industrial, econômico e militar) se desintegrarão enquanto os computadores e o software levarão a uma "democratização do conhecimento" (PAREIRA JÚNIOR, 2005, p. 5).

Primeiros Anos 1950 a 1960	Segura Era 1960 a 1970	Terceira Era 1970 a 1980	Quarta Era 1980 a 2000
Orientação Batch	Multiusuário Interativo	Sistemas Distribuídos	Sistemas de Desktop poderosos
Distribuição Limitadas	Tempo Real	Hardware de baixo custo	Tecnologias Orientadas à Objetos
Software Customizado	Banco de Dados	Microprocessadores	Sistemas Especialistas
Programação Artesanal	Produto de software	Impacto de Consumo	Computação paralela
Sem Administração Específica	Software House	"inteligênia" embutida	Ferramentas CASE
Sem documentação			Reutilização
			Redes Neurais artificias

Quadro 1: Evolução do software
Fonte: (PARREIRA JÚNIOR, 2005)

O Quadro 1 descreve a evolução do software dentro do contexto das áreas de aplicação de sistemas baseados em computador. Durante os primeiros anos do desenvolvimento de sistemas computadorizados, o hardware sofreu continuas mudanças, enquanto o software era visto por muitos como uma reflexão posterior. A programação de computador era uma arte "secundaria" para a qual havia poucos métodos sistemáticos. O desenvolvimento do software era feito virtualmente sem administração - ate que os prazos começassem a se esgotar e os custos a subir abruptamente. Durante esse período, era usada uma orientação batch (em lote) para a maioria dos sistemas. Notáveis exceções foram os sistemas interativos, tais como o primeiro sistema de reservas da *American Airlines* e os sistemas de tempo real orientados a defesa, como o SAGE. Na maior parte, entretanto, o hardware dedicava-se a execução de um único programa que, por sua vez, dedicava-se a uma aplicação específica (PARREIRA JÚNIOR; TEODORO NETO, 2002).

Durante os primeiros anos, o hardware de propósito geral tornara-se lugar comum. O software, por outro lado, era projetado sob medida para cada aplicação e tinha uma distribuição relativamente limitada. O produto software (isto e, programas desenvolvidos para serem vendidos a um ou mais clientes) estava em sua infância (SOMMERVILLE, 2003. A maior parte do software era desenvolvida e, em ultima analise, usada pela própria pessoa ou organização. Você escrevia-o, colocava-o em funcionamento e, se ele falhasse, era você quem o consertava. Uma vez que a rotatividade de empregos era baixa, os gerentes podiam dormir

tranquilos com a certeza de que você estaria lá se defeitos fossem encontrados (PARREIRA JÚNIOR, 2002).

Por causa desse ambiente de software personalizado, o projeto era um processo implícito realizado no cérebro de alguém, e a documentação muitas vezes não existia. Durante os primeiros anos, aprendemos muito sobre a implementação de sistemas baseados em computador, mas relativamente pouco sobre engenharia de sistemas de computador. Por justiça, entretanto, devemos reconhecer os muito surpreendentes sistemas baseados em computador desenvolvidos durante essa época. Alguns deles permanecem em uso ate hoje e constituem feitos que são um marco de referência e que continuam a justificar a admiração (CARNEIRO, 2000).

A segunda era da evolução dos sistemas computadorizados estendeu-se de meados da década de 1960 até o final da década de 1970. A multiprogramação e os sistemas multiusuários introduziram novos conceitos de interação homem-máquina. As técnicas interativas abriram um novo mundo de aplicações e novos níveis de sofisticação de software e hardware. Sistemas de tempo real podiam coletar, analisar e transformar dados de múltiplas fontes, dai controlando processos e produzindo saída em milissegundos, e não em minutos. Os avanços da armazenagem on-line levaram à primeira geração de sistemas de gerenciamento de bancos de dados (PARREIRA JÚNIOR, 2005).

A segunda era também foi caracterizada pelo uso do produto de software e pelo advento das "*software houses*". O software era desenvolvido para ampla distribuição num mercado interdisciplinar. Programas para mainframes e minicomputadores eram distribuídos para centenas e, às vezes, milhares de usuários. Empresários da industria, governos e universidades puseram-se "desenvolver pacotes de software" e a ganhar muito dinheiro (PARREIRA JÚNIOR, 2005).

A medida que o número de sistemas baseados em computador crescia, bibliotecas de software de computador começaram a se expandir. Projetos de desenvolvimento internos nas empresas produziram dezenas de milhares de instruções de programa (SOMMERVILLE, 2003). Os produtos de software comprados no exterior acrescentaram centenas de milhares de novas instruções. Uma nuvem negra apareceu no horizonte. Todos esses programas - todas essas instruções – tinham de ser corrigidos quando eram detectadas falhas, alterados conforme as exigências do usuário se modificavam ou adaptados a um novo hardware que fosse comprado. Essas atividades foram chamadas coletivamente de manutenção de software. O

esforço despendido na manutenção de software começou a absorver recursos em índices alarmantes .

A terceira era da evolução dos sistemas computadorizados começou em meados da década de 1970 e continua ate hoje. Os sistemas distribuídos - múltiplos computadores, cada um executando funções concorrentemente e comunicando-se um com o outro - aumentaram intensamente a complexidade dos sistemas baseados em computador. As redes globais e locais, as comunicações digitais de largura de banda (*handwidth*) elevada e a crescente demanda de acesso "instantâneo" a dados exigem muito dos desenvolvedores de software (SOMMERVILLE, 2003).

A terceira era também foi caracterizada pelo advento e generalização do uso de microprocessadores, computadores pessoais e poderosas estações de trabalho (*workstations*) de mesa. O microprocessador gerou um amplo conjunto de produtos inteligentes - de automóveis a fornos de microondas, de robôs industriais a equipamentos para diagnostico de soro sanguíneo. Em muitos casos, a tecnologia de software esta sendo integrada a produtos por equipes técnicas que entendem de hardware mais que frequentemente são principiantes em desenvolvimento de software (PARREIRA JÚNIOR, 2005).

O computador pessoal foi o catalisador do crescimento de muitas empresas de software. Enquanto as empresas de software da segunda era vendiam centenas ou milhares de copias de seus programas, as empresas da terceira era vendem dezenas e até mesmo centenas de milhares de cópias. O hardware de computador pessoal está se tornando rapidamente um produto primário, enquanto o software oferece a característica capaz de diferenciar. De fato, quando a taxa de crescimento das vendas de computadores pessoais se estabilizou em meados da década de 1980, as vendas de software continuaram a crescer. Na indústria ou em âmbito domestico, muitas pessoas gastaram mais dinheiro em software do que aquilo que despenderam para comprar o computador no qual o software seria instalado (PARREIRA JÚNIOR; TEODORO NETO, 2002).

A quarta era do software de computador esta apenas começando. As tecnologias orientadas a objetos estão rapidamente ocupando o lugar das abordagens mais convencionais para o desenvolvimento de software em muitas áreas de aplicação. Autores tais como Feigenhaum, McCorduck e Allman (*apud* PARREIRA JÚNIOR, 2005) prevêem que os computadores de "quinta geração", arquiteturas de computação radicalmente diferentes, e seu software correlato exercerão um profundo impacto sobre o equilíbrio do poder político e industrial em todo o mundo. As técnicas de "quarta geração" para o desenvolvimento de

software já estão mudando a maneira segundo a qual alguns segmentos da comunidade de software constroem programas de computador. Os sistemas especialistas e o software de inteligência artificial finalmente saíram do laboratório para a aplicação pratica em problemas de amplo espectro do mundo real. O software de rede neural artificial abriu excitantes possibilidades para o reconhecimento de padrões e para capacidades de processamento de informações semelhantes as humanas.

Quando nos movimentamos para a quinta era, os problemas associados ao software de computador continuam a se intensificar segundo Deitel (2003):

- A sofisticação do software ultrapassou nossa capacidade de construir um software que extraia o potencial do hardware.
- Nossa capacidade de construir programas não pode acompanhar o ritmo da demanda de novos programas.
- Nossa capacidade de manter os programas existentes é ameaçada por projetos ruins e recursos inadequados.

Em resposta a esses problemas, estão adotadas práticas de engenharia em todos os seguimentos de trabalho.

2.2 APLICAÇÕES DE SOFTWARE

O software pode ser aplicado a qualquer situação em que um conjunto previamente especificado de passos procedimentais (um algoritmo) tiver sido definido. O conteúdo de informações e a previsibilidade são fatores importantes na determinação da natureza de um aplicativo.

Desenvolver categorias genéricas para as aplicações de softwares é uma tarefa muito difícil. Quanto mais complexo é o sistema, mais difícil é determinar de forma clara os vários componentes do software.

As aplicações dividem-se, segundo Deitel (2003), em:

- Software Básico – que é um conjunto de programas para dar apoio a outros programas. Tem como característica uma forte interação com o hardware, operações concorrentes, compartilhamento de recursos, uso por múltiplos usuários e múltiplas interfaces.

- Software de Tempo Real – são programas que monitora, analisa e controla eventos do mundo real, devendo responder aos estímulos do mundo externo com restrições de tempo pré-determinadas.
- Software Comercial – é a maior área de aplicação de softwares, são aplicações que gerenciam as operações comerciais de modo a facilitar o gerenciamento comercial do negócio da empresa, permitindo também a tomada de decisões.
- Software Cientifico e de Engenharia – são caracterizados por algoritmos de processamento numérico, dependentes da coleta e processamento de dados para as mais variadas áreas do conhecimento.
- Software Embutido – são desenvolvidos para executar atividades muito específicas e inseridos em produtos inteligentes tanto para atividades comerciais como para atividades domesticas.
- Software de Computador Pessoal – são produtos desenvolvidos para o uso pessoal do computador, tais como planilhas eletrônicas, processadores de textos, jogos, etc.
- Software de Inteligência Artificial – faz uso de algoritmos não-numéricos para resolver problemas complexos que não apresentam facilidades computacionais numéricas ou de análise direta.

2.3 CONCEITO DE ENGENHARIA DE SOFTWARE

Uma primeira definição de engenharia de software foi proposta por Fritz Bauer (*apud* BRAZ JÚNIOR, 1997), na primeira grande conferencia dedicada ao assunto: "o estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que funcione eficientemente com máquinas reais".

A engenharia de software é uma derivação da engenharia de sistemas e de hardware. Ela abrange um conjunto de três elementos fundamentais - métodos, ferramentas e procedimentos - que possibilita ao gerente o controle do processo de desenvolvimento do software e oferece ao profissional uma base para a construção de software de alta qualidade produtivamente (SOMMERVILLE, 2003).

Os métodos de engenharia de software proporcionam os detalhes de "como fazer" para construir o software. Os métodos envolvem um amplo conjunto de tarefas que incluem: planejamento e estimativa de projeto, análise de requisitos de software e de sistemas, projeto

da estrutura de dados, arquitetura de programa e algoritmo de processamento, codificação, teste e manutenção. Os métodos da engenharia de software muitas vezes introduzem uma notação gráfica ou orientada a linguagem especial e introduzem um conjunto de critérios para a qualidade do software (SOMMERVILLE, 2003).

As ferramentas de engenharia de software proporcionam apoio automatizado ou semiautomatizado aos métodos. Atualmente, existem ferramentas para sustentar cada um dos métodos anotados anteriormente. Quando as ferramentas são integradas de forma que a informação criada por uma ferramenta possa ser usada por outra, é estabelecido um sistema de suporte ao desenvolvimento de software chamado engenharia de software auxiliada por computador (CASE - *Computer-Aided Software Engineering*). O CASE combina software, hardware e um banco de dados de engenharia de software (uma estrutura de dados contendo importantes informações sobre analise, projeto, codificação e teste) para criar um ambiente de engenharia de software que seja análogo ao projeto auxiliado por computador/engenharia auxiliada por computador para o hardware (PAULA FILHO, 2001).

Os procedimentos da engenharia de software constituem o elo de ligação que mantém juntos os métodos e as ferramentas e possibilita o desenvolvimento racional e oportuno do software de computador. Os procedimentos definem a sequência em que os métodos serão aplicados, e se exige que sejam entregues (documentos, relatórios, formulários etc.), os controles que ajudam a assegurar a qualidade e a coordenar as mudanças, e os marcos de referencia que possibilitam aos gerentes de software avaliar o progresso (PAULA FILHO, 2001).

A engenharia de software compreende um conjunto de etapas que envolve métodos. ferramentas e os procedimentos. Essas etapas muitas vezes são citadas como paradigmas da engenharia de software. Um paradigma de engenharia de software e escolhido tendo-se como base a natureza do projeto e da aplicação, os métodos e as ferramentas a serem usados, os controles e os produtos que precisam ser entregues (DEITEL, 2003).

Engenharia de software segundo Parreira Júnior (2005, p. 9) é “um conjunto integrado de métodos e ferramentas utilizadas para especificar, projetar, implementar e manter um sistema”.

Segundo Ariadne Carvalho & Thelma Chiossi no livro Introdução à Computação, a Engenharia de Software, *in* Parreira Júnior (2005, p. 10) é "uma disciplina que reúne metodologias, métodos e ferramentas a ser utilizados, desde a percepção do problema até o momento em que o sistema desenvolvido deixa de ser operacional, visando resolve problemas

inerentes ao processo de desenvolvimento e ao produto de software". Pode-se definir como: a) um método é uma prescrição explícita de como chegar a uma atividade requerida por um modelo de ciclo de vida, visando otimizar a execução das atividades que foram especificadas; b) as ferramentas proporcionam apoio automatizado ou semi-automatizado aos métodos,

Os Métodos de Desenvolvimento de Sistema se diferenciam pela maneira como o problema deve ser visualizado e como a solução do problema deve ser modelada, São eles:

2.3.1 Método baseado na decomposição de funções

Abordagem estruturada caracterizada pela decomposição das funções, Os tipos de modelos que representam as funções são:

- DFD (Diagrama de Fluxo de Dados) - se caracteriza pela decomposição hierárquica de processos.
- MHT (Modelo Hierárquico de Tarefas) - se baseia na decomposição hierárquica de tarefas.

2.3.2 Método baseado na estrutura de dados

Abordagem baseada na decomposição de um problema a partir dos dados. Exemplos de tipos de modelos dessa classe:

- MER (Modelagem Entidade-Relacionamento)
- Técnica de Warnier.

2.3.3 Método de Análise baseado na Orientação a Objeto.

Os tipos de modelos que representam essa classe são:

- UML (*Unified Process*) - notação de modelagem, independente de processos de desenvolvimento.
- Cenários

2.4 PARADIGMAS DE ENGENHARIA DE SOFTWARE

Segundo Pressman (2006, p. 175), paradigmas são os modelos de processos que possibilitam:

- ao gerente: o controle do processo de desenvolvimento de sistemas de software,
- ao desenvolvedor: a obter a base para produzir, de maneira eficiente, software que satisfaça os requisitos preestabelecidos.

Os paradigmas especificam algumas atividades que devem ser executadas, assim como a ordem em que devem ser executadas. A função dos paradigmas é diminuir os problemas encontrados no processo de desenvolvimento do software, e deve ser escolhido de acordo com a natureza do projeto e do produto a ser desenvolvido, dos métodos e ferramenta a ser utilizado e dos controles e produtos intermediários desejados (PRESMAN (2006).

Segundo Sommerville (2011, p. 176), “um processo de software é um conjunto de atividades e resultados associados que levam à produção de um produto de software." Embora existam muitos processos de software diferentes, há atividades fundamentais comuns a todos eles, tais como:

- Especificação de software;
- Projeto e implementação de software;
- Validação de software;
- Evolução do software.

2.5 OS PROCESSOS DE SOFTWARE

Um processo de software é um conjunto de atividades e resultados associados que levam à produção de um produto de software. Esse processo pode envolver o desenvolvimento de software desde o início, embora, cada vez mais, ocorra o caso de um software novo ser desenvolvido mediante a expansão e a modificação de sistemas já existentes (PRESSMAN, 2006).

2.5.1 Modelos de processos de software

Um modelo de processo representa um processo a partir de uma perspectiva particular, de uma maneira que proporciona apenas informações parciais sobre o processo.

Esses modelos genéricos não são descrições definitivas de processos de software, mas são abstrações úteis, que podem ser utilizadas para explicar diferentes abordagens do desenvolvimento do software.

2.5.2 Modelo em cascata

É o primeiro modelo publicado do processo de desenvolvimento de software.

O resultado de cada fase envolve um ou mais documentos que são aprovados e assinados. A fase seguinte só é iniciada após a conclusão da fase precedente, mas na prática eles se sobrepõem e trocam informações. Durante o projeto, são identificados problemas com os requisitos; durante a codificação são verificados problemas do projeto, e assim por diante. O processo não é um modelo linear simples, mas envolve uma sequencia de iterações das atividades de desenvolvimento.

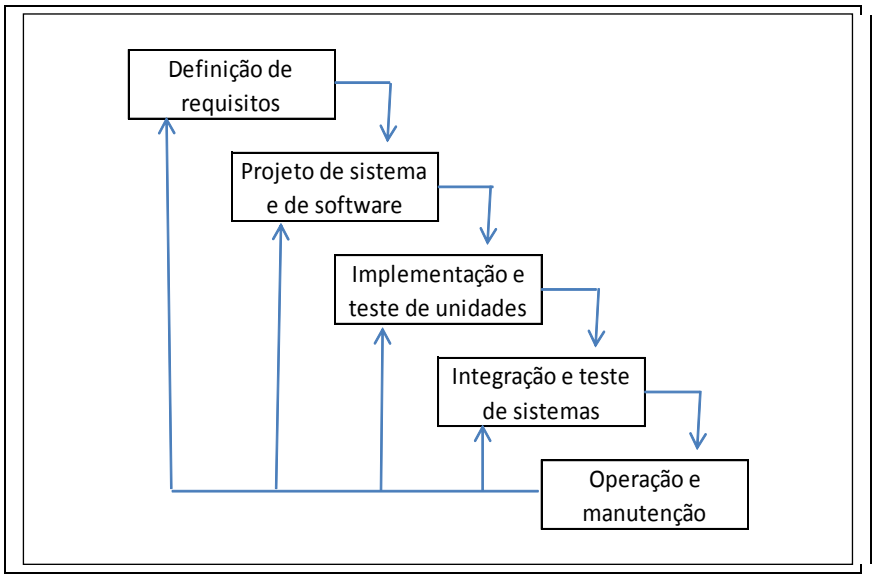


Figura 1: Modelo software em cascata

Fonte: (PRESSMAN, 2010)

Devido aos custos de produção e aprovação de documentos, as iterações são onerosas e envolvem muito retrabalho. Depois de algumas iterações é normal suspender partes do desenvolvimento da fase e passar para o próximo estágio, postergando soluções ainda não encontradas. Essa suspensão prematura pode resultar em problemas futuros, quando da finalização do software (PARREIRA JÚNIOR, 2013).

2.5.3 Desenvolvimento evolucionário

Tem como ideia o desenvolvimento da versão inicial que é exposta aos comentários do usuário e a partir destes é desenvolvido uma versão intermediária que é exposta aos comentários do usuário e assim sucessivamente (gera várias versões) até que um sistema adequado tenha sido desenvolvido.

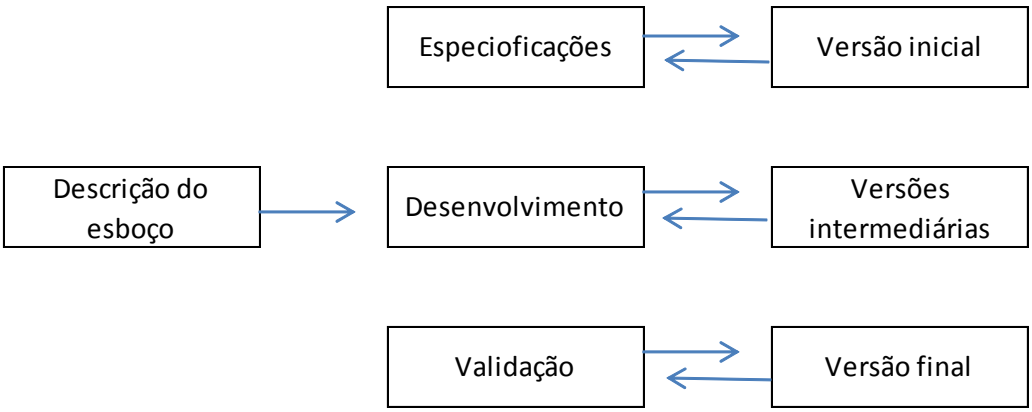


Figura 2: Atividades concorrentes

Fonte: (PRESSMAN, 2010)

Há duas formas de desenvolvimento:

- Desenvolvimento exploratório - tem como objetivo trabalhar com o cliente e explorar os seus requisitos e entregar um sistema final;
- Prototipos descartáveis - tem corno objetivo compreender os requisitos através dos prototipos e depois desenvolver urna definição dos requisitos para o sistema.

2.5.4 Desenvolvimento formal de sistemas

É uma abordagem que tem pontos em comum com o modelo em cascata, mas cujo processo de desenvolvimento tem como base a transformação matemática formal de uma especificação de sistemas em um programa executável. O desenvolvimento formal de sistema:

Figura 3: Desenvolvimento formal de sistema

Fonte: (PRESMAN, 2010)

Este modelo tem a vantagem de reduzir a quantidade de software a ser desenvolvido, portanto de reduzir custos e riscos, permitindo desta forma a entrega mais rápida do software Contudo, as adequações nos requisitos podem ser inevitáveis e pode resultar em um sistema que não atenda as necessidades do usuário, além de que novas versões dos componentes reutilizáveis podem não estar sob controle da equipe de desenvolvimento.

2.6 A LINGUAGEM DE CONSULTA SQL

O nome SQL[5] (*Structured Query Language* - Linguagem Estruturada de Consulta) explica sua finalidade. Não é uma linguagem de programação de computadores especificamente criada para desenvolver sistemas, como são as linguagens PASCAL, C. BASIC, COBOL, MODULA-2. LUA (linguagem brasileiras http://www.lua.org/ portugues.html), entre outras. É tão somente uma linguagem declarativa utilizada para facilitar o acesso a informações (por meio de consultas, atualizações e manipulações de dados armazenadas em banco de dados relacional (MANZANO, 2009).

Um banco de dados relacional armazena os dados e informações em tabelas que são formadas por linhas (seus registros) e colunas (seus campos). Atualmente os programas de gerenciamento de bancos de dados (SGBD - Sistema de Gerenciamento de Bancos de Dados) disponíveis no mercado mundial são ferramentas baseadas em banco de dados relacional (MANZANO, 2009).

A linguagem de consulta estruturada SQL foi desenvolvida primeiramente pela empresa IBM (International Business Machine), e apresentada em sua primeira versão no ano de 1974, com o nome *Structured English QUEry Language* (SEQUEL). A linguagem de consulta SEQUEL foi desenvolvica pelo Ph.D. Donald D. Chamberlin, e outros profissionais da IBM.

O próximo capítulo trataremos do projeto de desenvolvimento do programa de computação, dirigido as fazendas para análise das pragas encontradas na região.

---

5 O termo SQL deveria ser pronunciado como “siquel” e não “esse que el” como normalmente acontece.

3 PROJETO

Para o desenvolvimento de qualquer sistema computacional é necessário um estudo prévio para definir a abrangência do sistema, suas funcionalidades, requisitos e o *hardware* necessário para que o sistema promova os resultados esperados.

Fez-se necessário a escolha de uma linguagem de programação orientada a objetos, foi escolhido o C# por se tratar de uma linguagem já consolidada no mercado que muitos sistemas que já a utilizam.

Para o levantamento de requisitos foi utilizado a própria experiencia do autor, além de consultas a Embrapa no qual foi seguido o material de produção integrada de maçã (PIM).

3.1 DIAGRAMA DE BANCO DE DADOS

Diagrama de banco de dados é uma coleção de dados logicamente coerente que possui um significado implícito cuja interpretação é dada por uma determinada aplicação. Representa abstratamente uma parte do mundo real, conhecida como Mini-mundo ou Universo de discurso, que é de interesse de uma certa aplicação e mantido em dispositivos de armazenamento secundário de um sistema de computação (DATE, 2000).

Assim, o Sistema Gerenciador de Bancos de Dados (SGBD), é um software construído para facilitar as atividades de definição, construção e manipulação de bancos de dados.

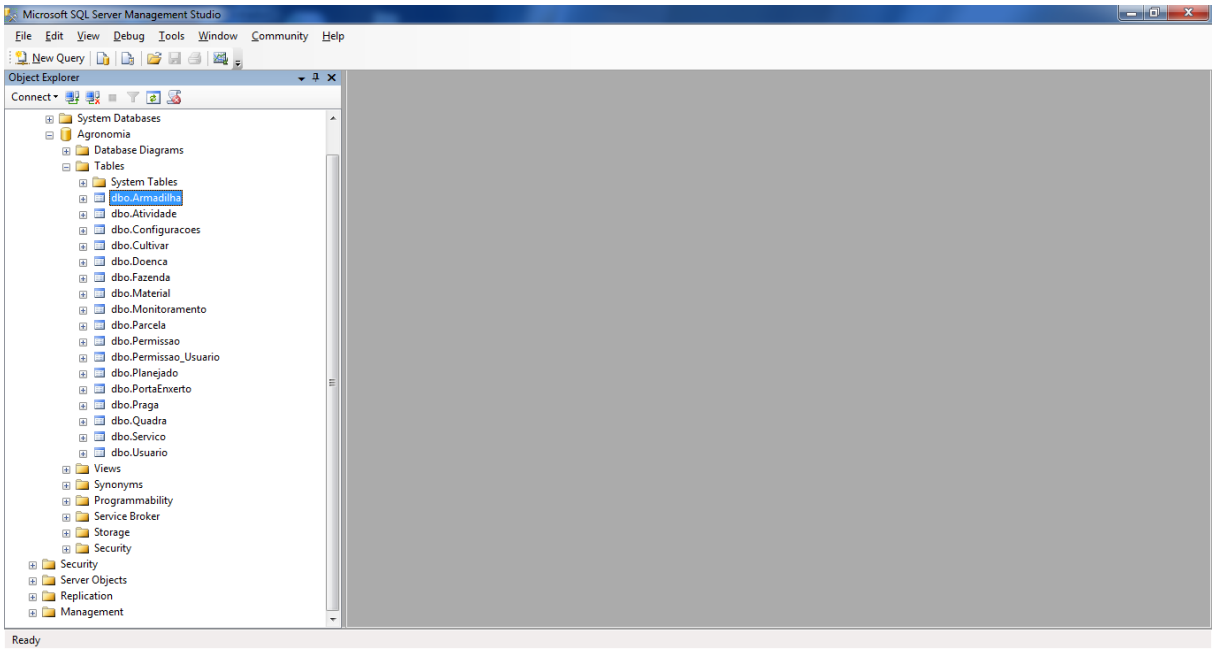


Figura 4: Banco de dados

Fonte: Autor (2013)

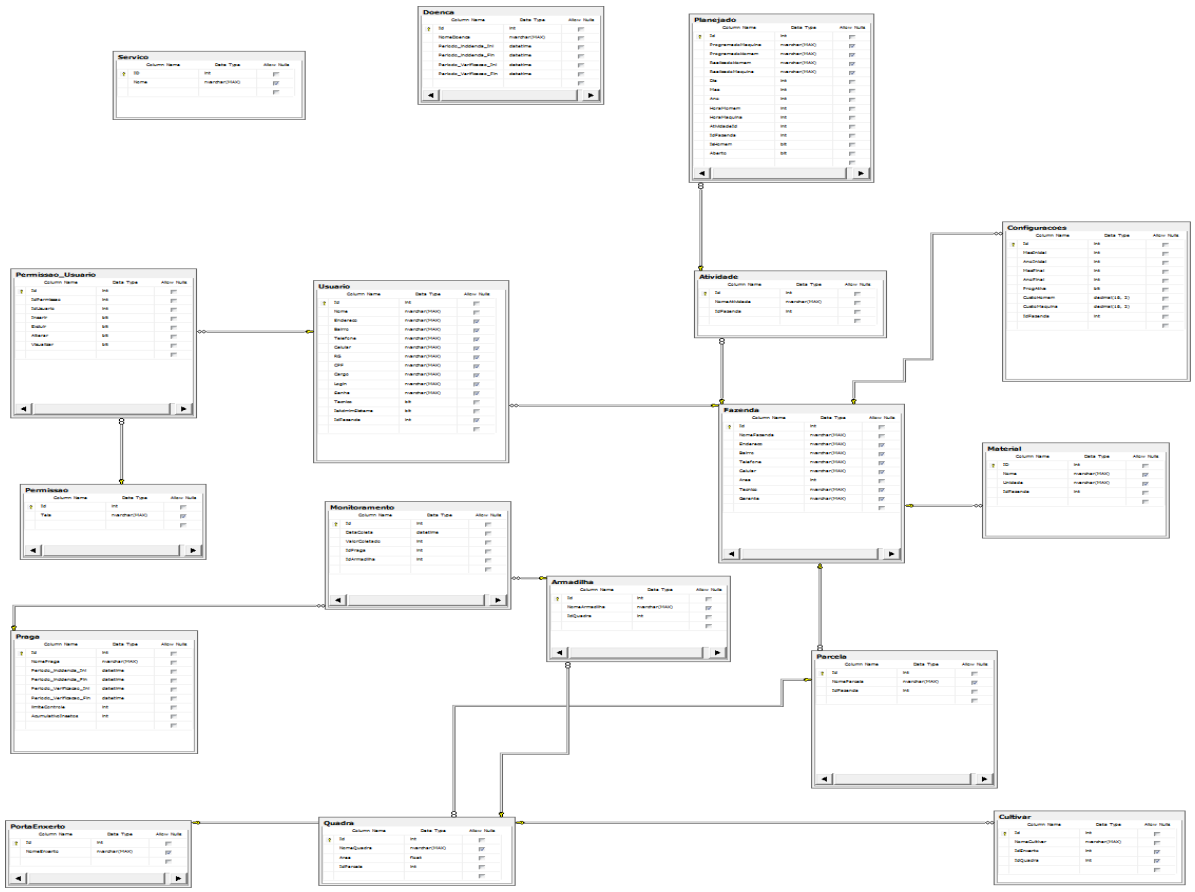


Figura 5: Diagrama de Banco de Dados

Fonte: Próprio Autor (2013)

3.2 CASO DE USO

O diagrama de caso de uso descreve a funcionalidade proposta para um novo sistema que será projetado. Segundo Ivar Jacobson (citado por DATE, 2000), podemos dizer que um caso de uso é um documento narrativo que descreve a sequência de eventos de um ator que usa um sistema para completar um processo. Um caso de uso representa uma unidade discreta da interação entre um usuário (humano ou máquina) e o sistema. Um caso de uso é uma unidade de um trabalho significante. Por exemplo: o login para o sistema, registrar no sistema e criar pedidos são todos casos de uso. Cada caso de uso tem uma descrição da funcionalidade que será construída no sistema proposto. Um caso de uso pode usar outra funcionalidade de caso de uso ou estender outro caso de uso com seu próprio comportamento.

Casos de uso são tipicamente relacionados a "atores". Um ator é um humano ou entidade máquina que interage com o sistema para executar um significante trabalho (DATE, 2000).

Esse diagrama documenta o que o sistema faz do ponto de vista do usuário. Em outras palavras, ele descreve as principais funcionalidades do sistema e a interação dessas funcionalidades com os usuários do mesmo sistema. Nesse diagrama não nos aprofundamos em detalhes técnicos que dizem como o sistema faz.

Este artefato é comumente derivado da especificação de requisitos, que por sua vez não faz parte da UML. Pode ser utilizado também para criar o documento de requisitos.

Diagramas de Casos de Uso são compostos basicamente por quatro partes:

- Cenário: Sequência de eventos que acontecem quando um usuário interage com o sistema.
- Ator: Usuário do sistema, ou melhor, um tipo de usuário.
- Use Case: É uma tarefa ou uma funcionalidade realizada pelo ator (usuário)

Comunicação: è o que liga um ator com um caso de uso

UML é um acrônimo para a expressão *Unified Modeling Language*. Pela definição de seu nome, vemos que UML é uma linguagem que define uma série de artefatos que nos ajuda na tarefa de modelar e documentar os sistemas orientados a objetos que desenvolvemos (DATE, 2000).

Ela possui nove tipos de diagramas que são usados para documentar e modelar diversos aspectos dos sistemas.

A maioria dos problemas encontrados em sistemas orientados a objetos tem sua origem na construção do modelo, no desenho do sistema. Muitas vezes as empresas e

profissionais não dão muita ênfase à essa fase do projeto, e acabam cometendo diversos erros de análise e modelagem. Isso quando há modelagem, pois nós profissionais da área sabemos que muitas vezes o projeto começa já na fase de codificação.

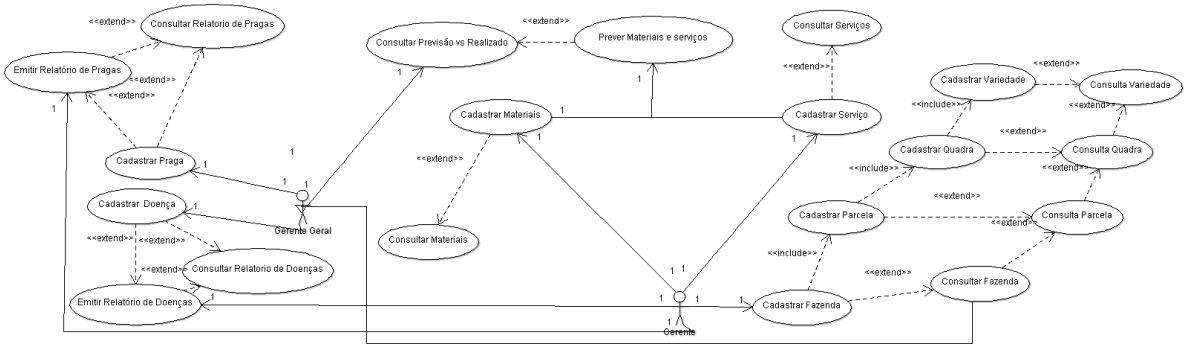


Figura 6: Diagramas de caso de uso (Anexo 2)

Figura: Autor (2013)

3.3 INTERFACES DO PEST CONTROL – SISTEMA PARA CONTROLE DE PRAGAS

Aqui serão apresentados as interfaces gráficas do sistema.

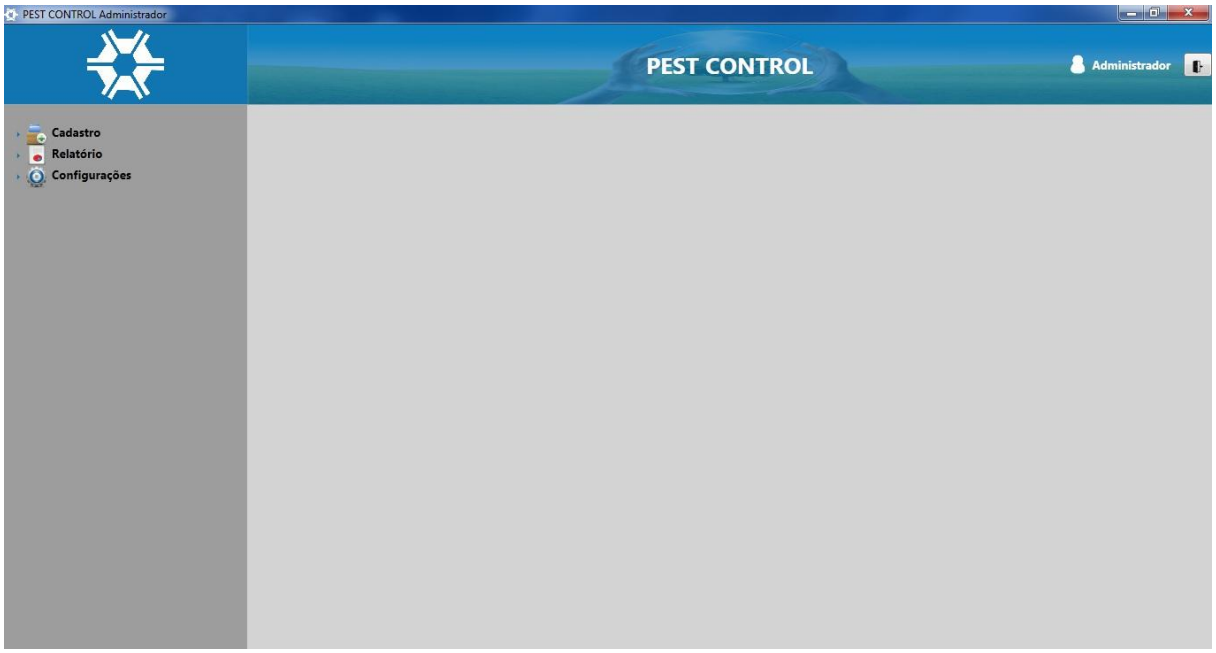


Figura 7: Tela inicial

Fonte: Autor (2013)

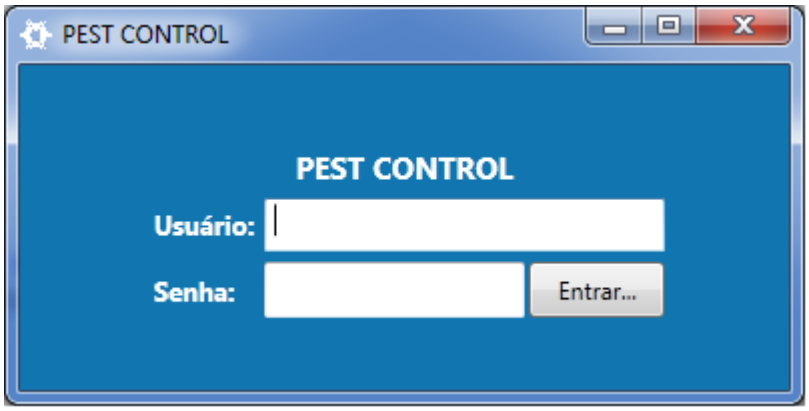


Figura 8: Tela Inicial do PEST CONTROL

Fonte: Próprio Autor

A Figura 8 apresenta a tela inicial do sistema. Nesta tela solicita o nome do usuário e sua senha, para que o sistema inicie e permita ao usuário a sua navegação.

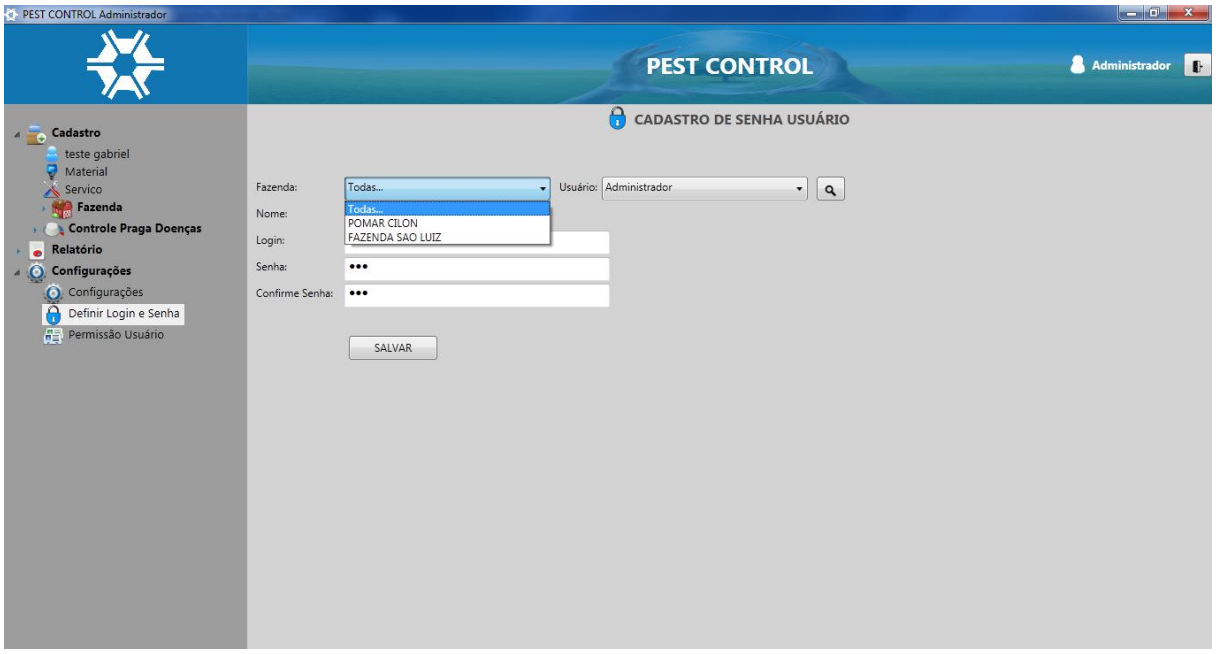


Figura 9: Cadastro de senha usuário

Fonte: Autor (2013)

A Figura 9 ilustra a tela de cadastro de senha do usuário para permitir a este que faça escolha da fazenda que deseja o controle e/ou verificação de pragas.

3.4 RESULTADOS

A seguir apresentamos algumas telas do sistema PEST CONTROL que após o usuário se identificar, o programa permite a esse que navegue por todas as informações que o programa lhe apresenta conforme sua necessidade e permissões de consulta.

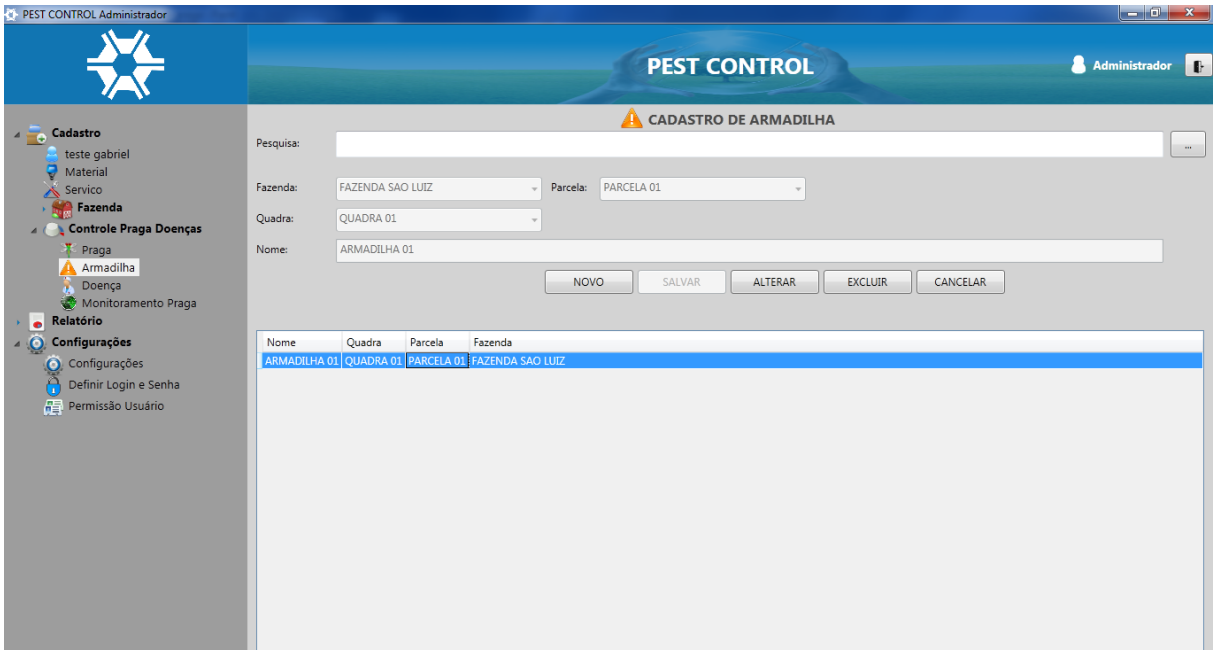


Figura 10: Cadastro de armadilhas

Fonte: Autor (2013)

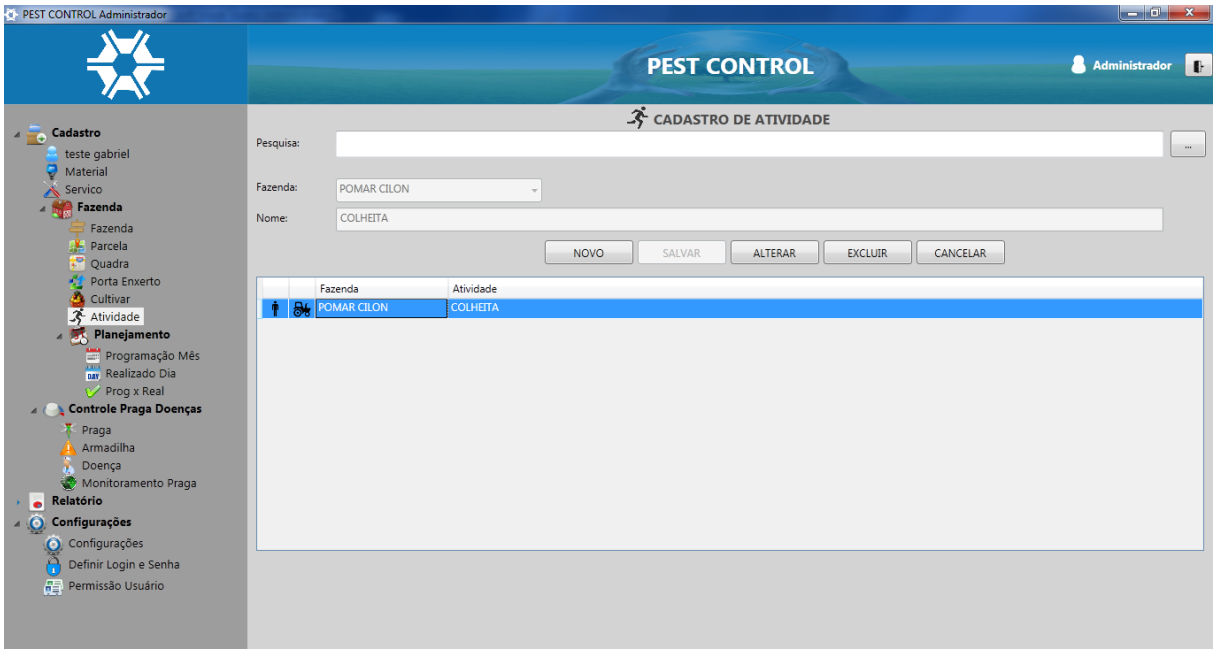


Figura 11: Cadastro de atividades

Fonte: Autor (2013)

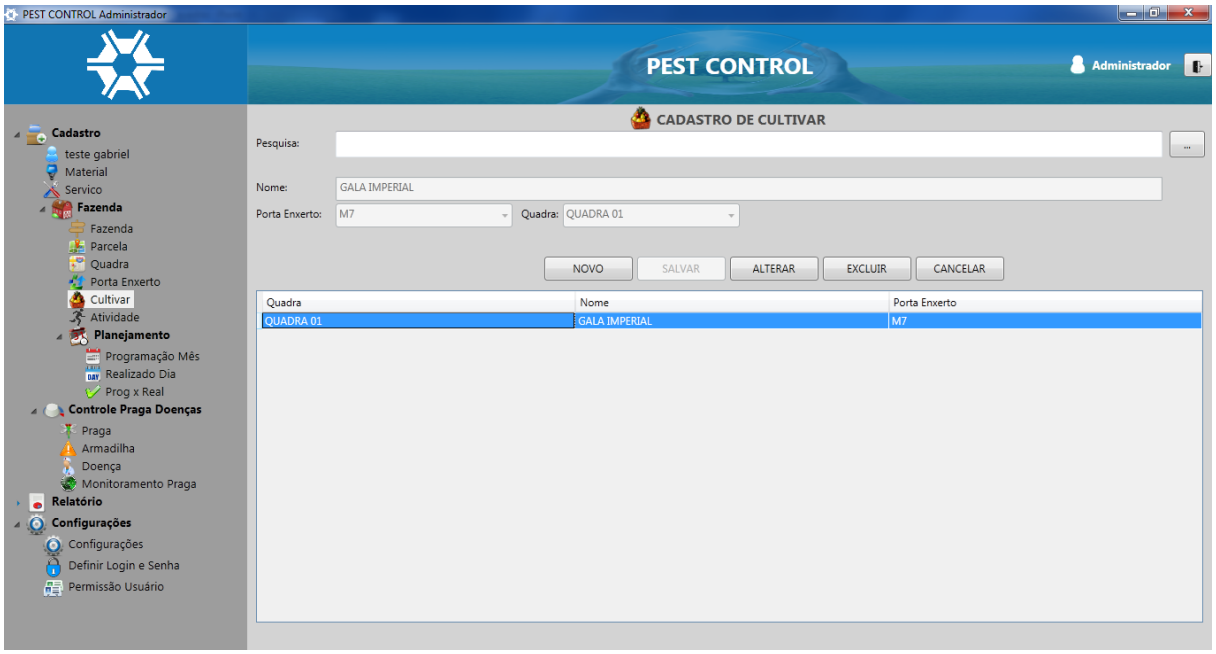


Figura 12: Cadastro de cultivar

Fonte: Autor (2013)

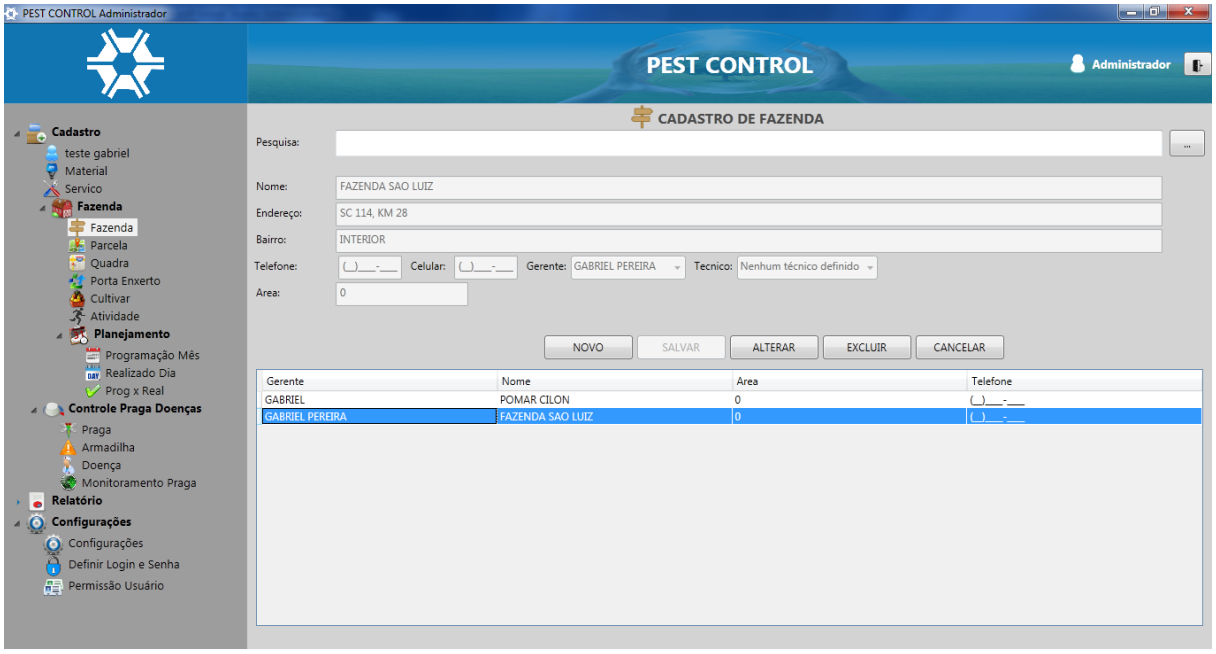


Figura 13: Cadastro de fazenda

Fonte: Autor (2013)

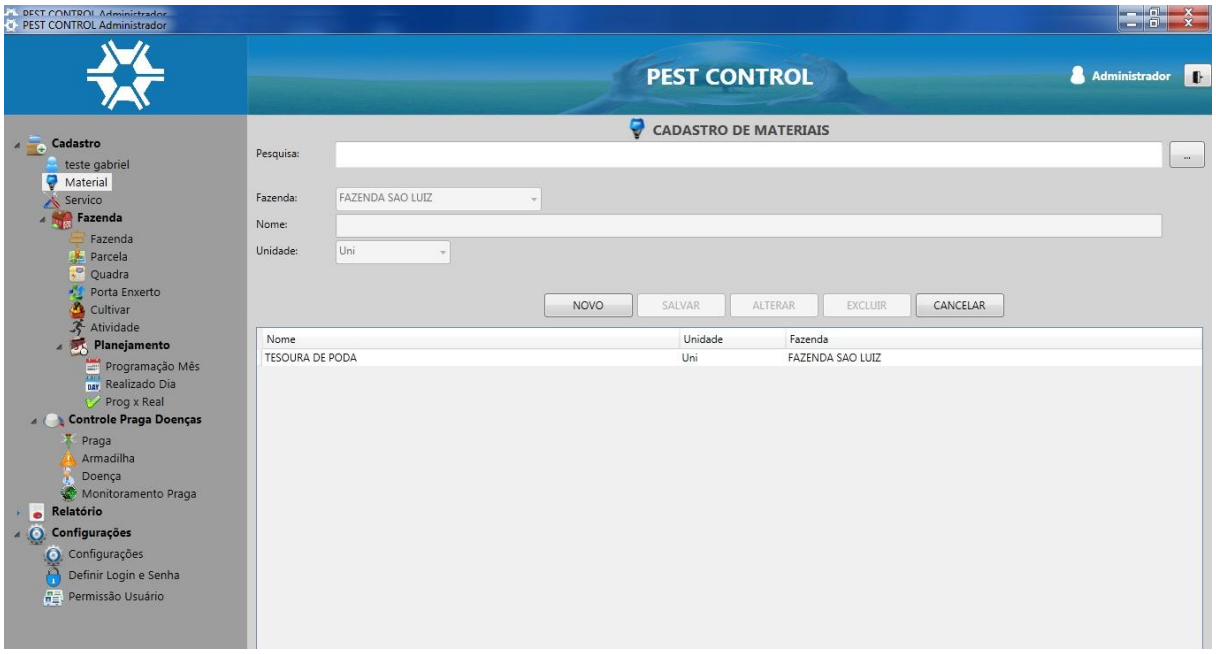


Figura 14: Cadastro de materiais

Fonte: Autor (2013)

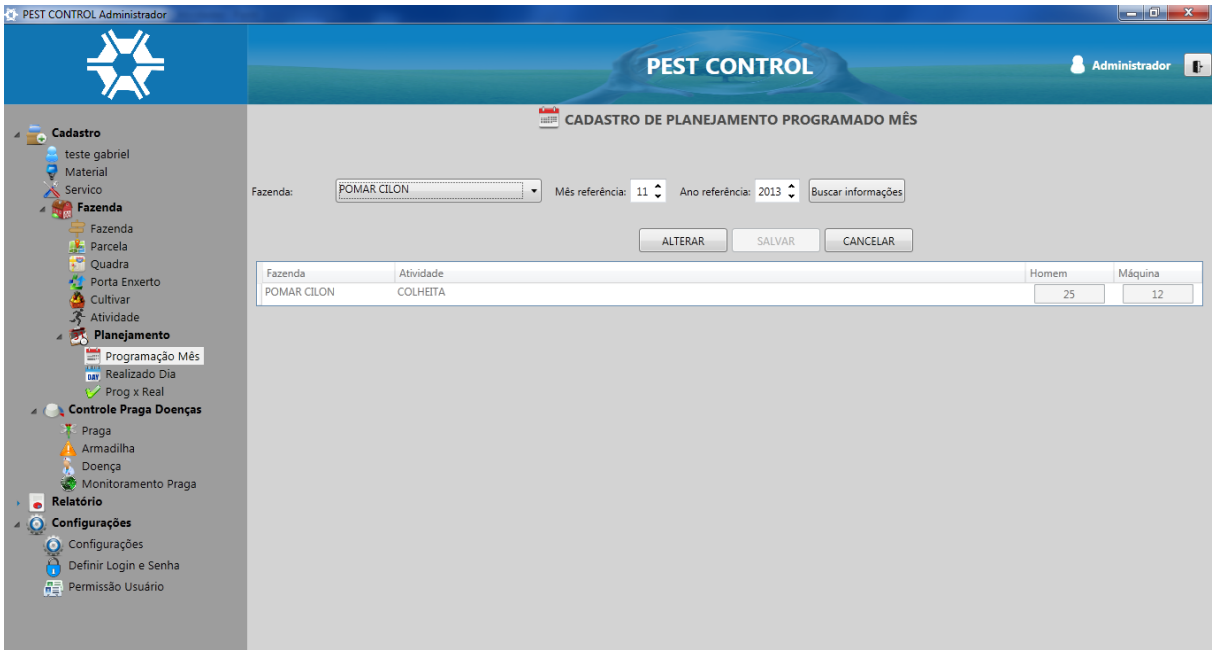


Figura 15: Cadastro de planejamento programado mês

Fonte: Autor (2013)

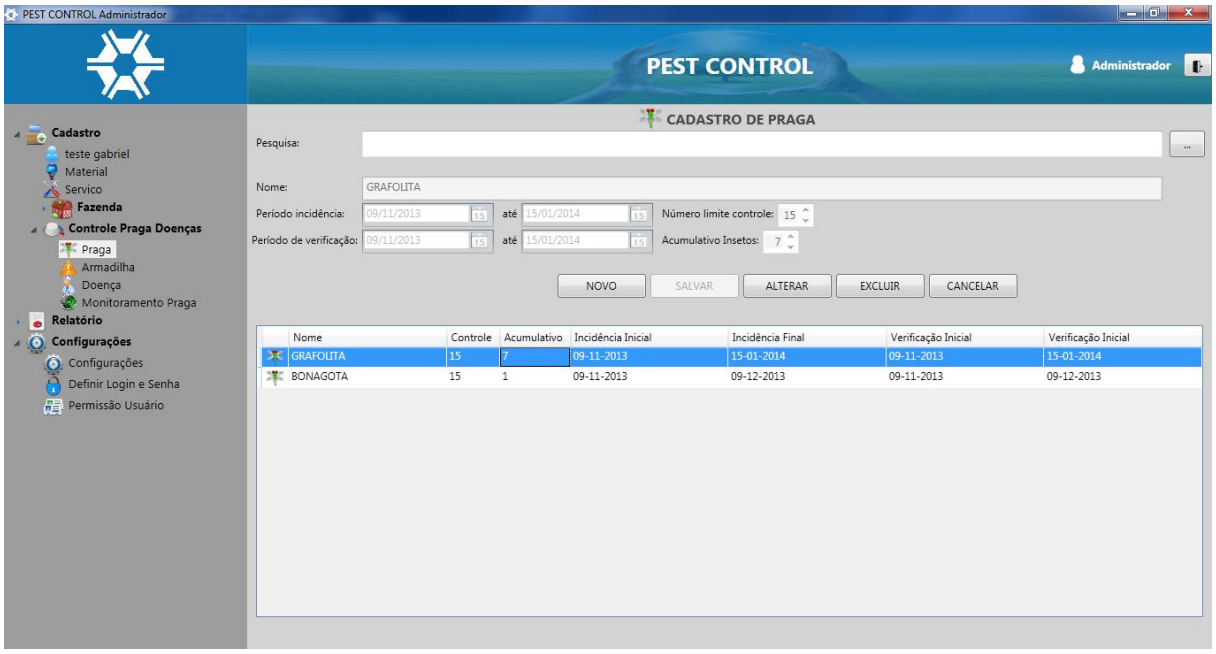


Figura 16: Cadastro de praga

Fonte: Autor (2013)

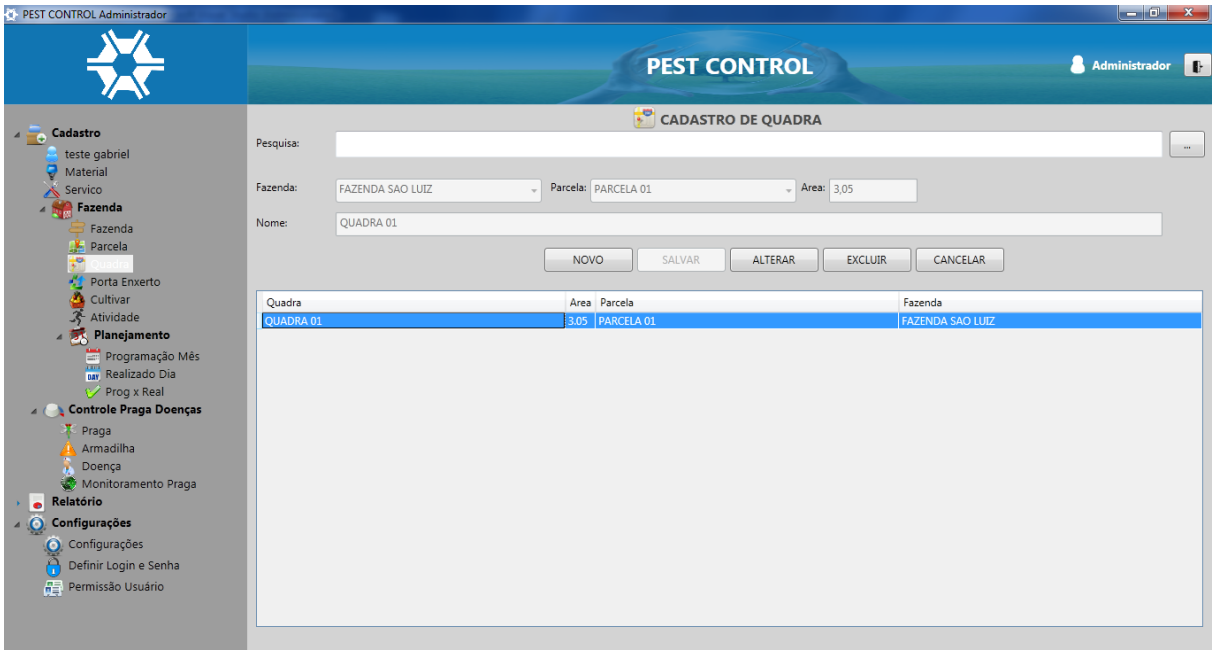


Figura 17: Cadastro de quadra

Fonte: Autor (2013)

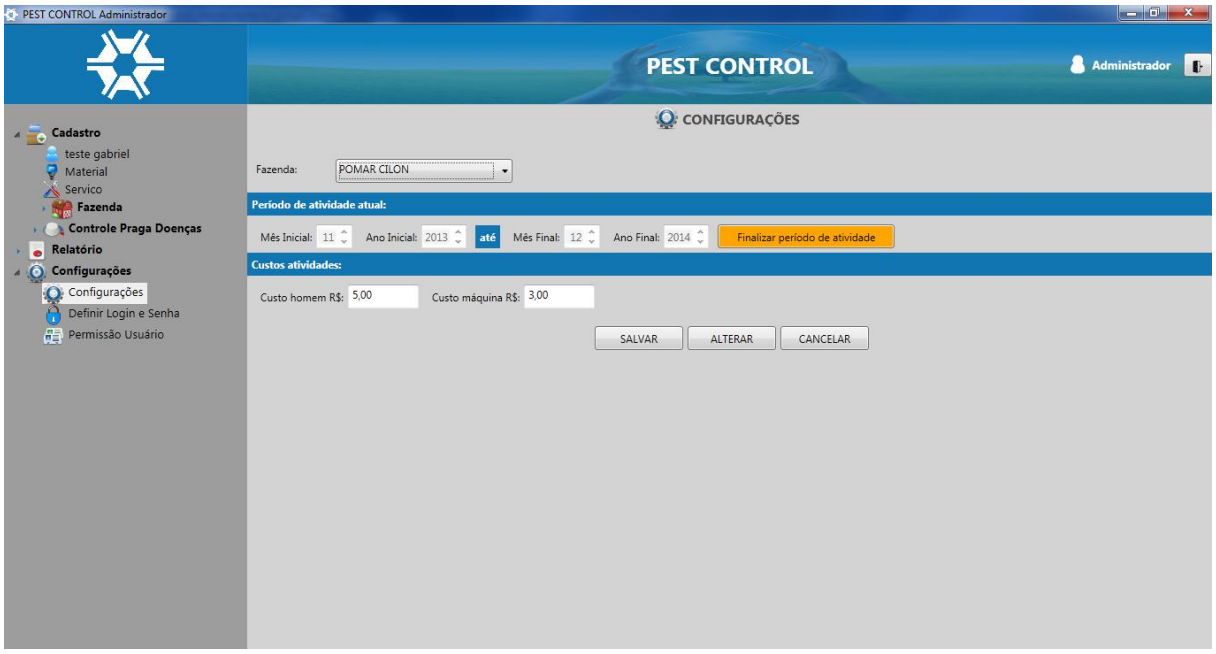


Figura 18: Configurações

Fonte: Autor (2013)

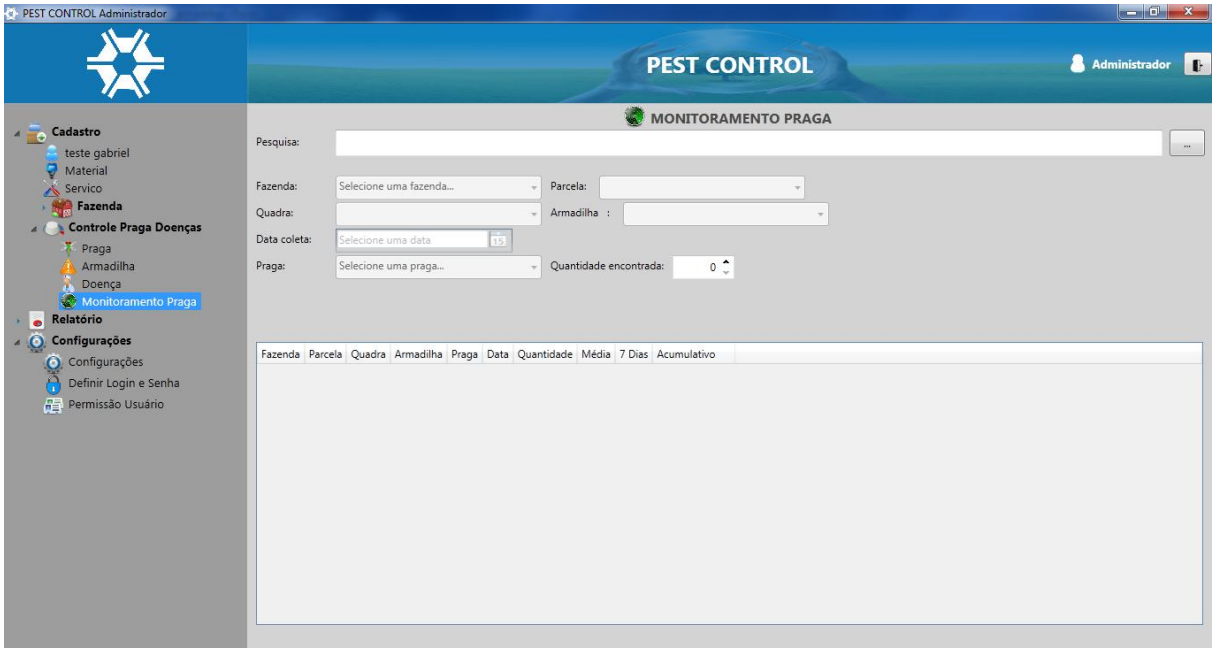


Figura 19: Monitoramento praga (Anexo 3 – Código Fonte)

Fonte: Autor (2013)

Na Figura 19, a tela mostra os resultados dos monitoramentos realizados, na qual apresenta os resultados para as tomadas de decisões por parte da equipe técnica de controle das pragas.

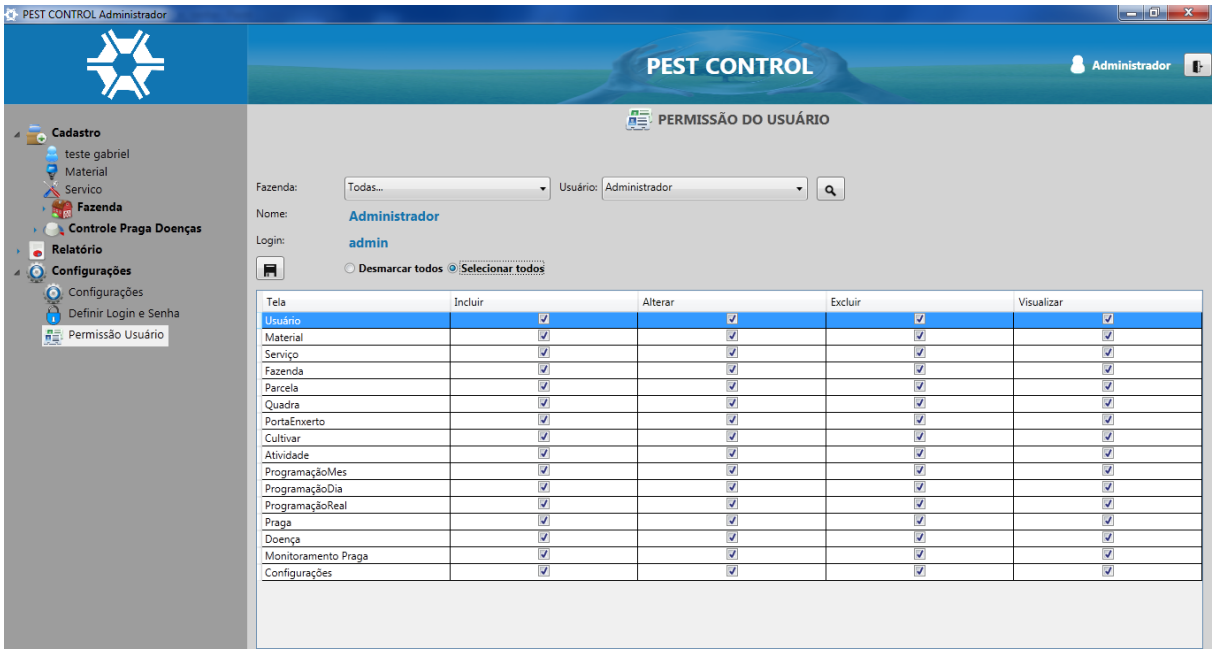


Figura: 20: Permissão do usuário

Fonte: Autor (2013)

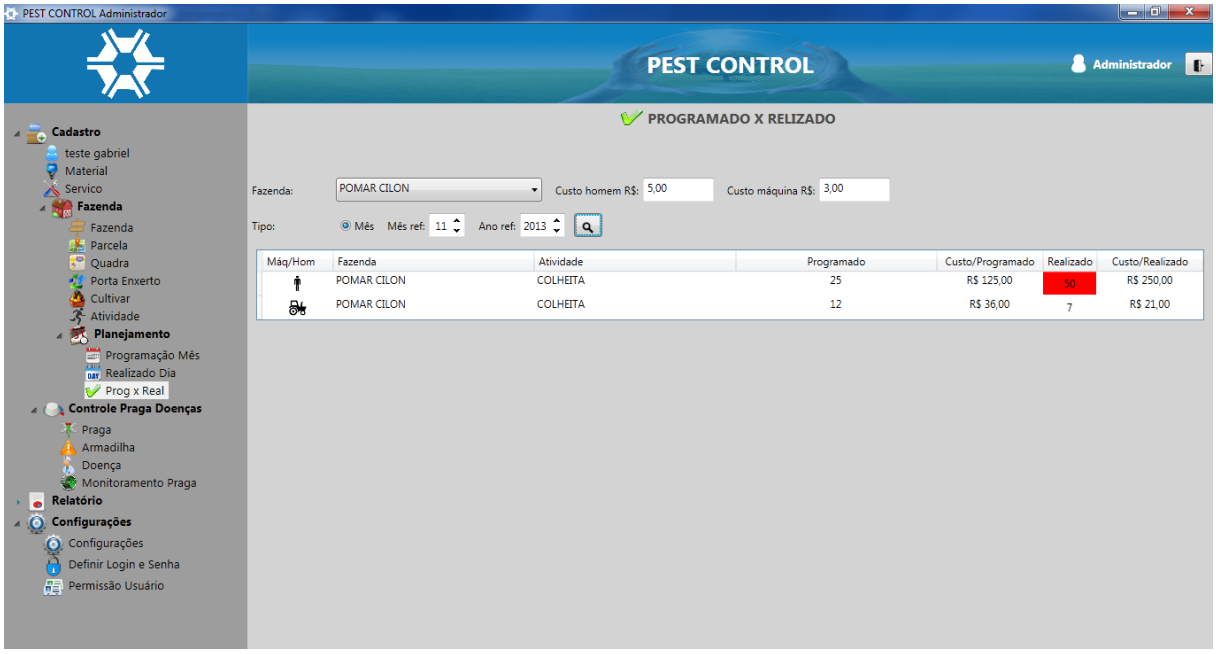


Figura 21: Programado x realizado

Fonte: Autor (2013)

A Figura 21, apresenta os resultado que servirão de parâmetros para tomada de decisão.

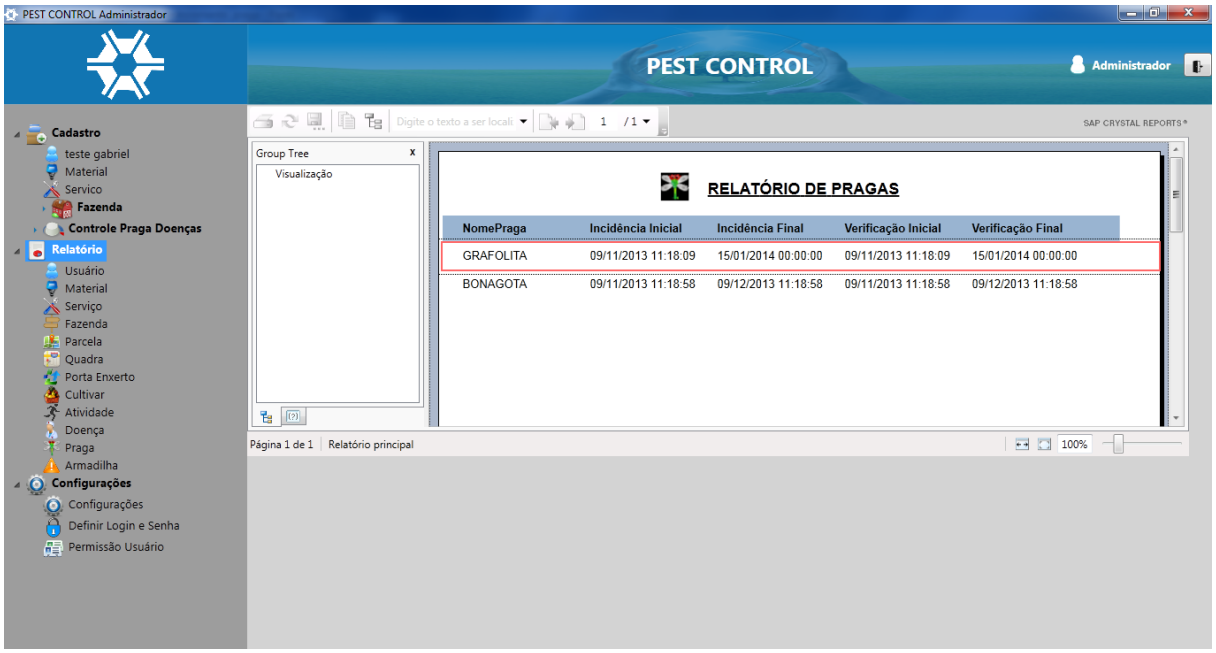


Figura 22: Relatório de pragas

Fonte: Autor (2013)

O PEST CONTROL é um programa de interesse nas fazendas para o controle de pragas, informando o início do surgimento da praga até o seu término.

O programa apresenta os fluxos de dados que entram e saem do sistema e sua interação. Tem como função delinear a área de observação.

O PEST CONTROL é uma representação em rede dos processos (funções) do sistema e dos dados que ligam esses processos. Ele mostra o que o sistema faz e não como é feito. As Figuras 10 até à 22, mostra os elementos essenciais para o perfeito funcionamento do programa PEST CONTROL.

4 CONSIDERAÇÕES FINAIS

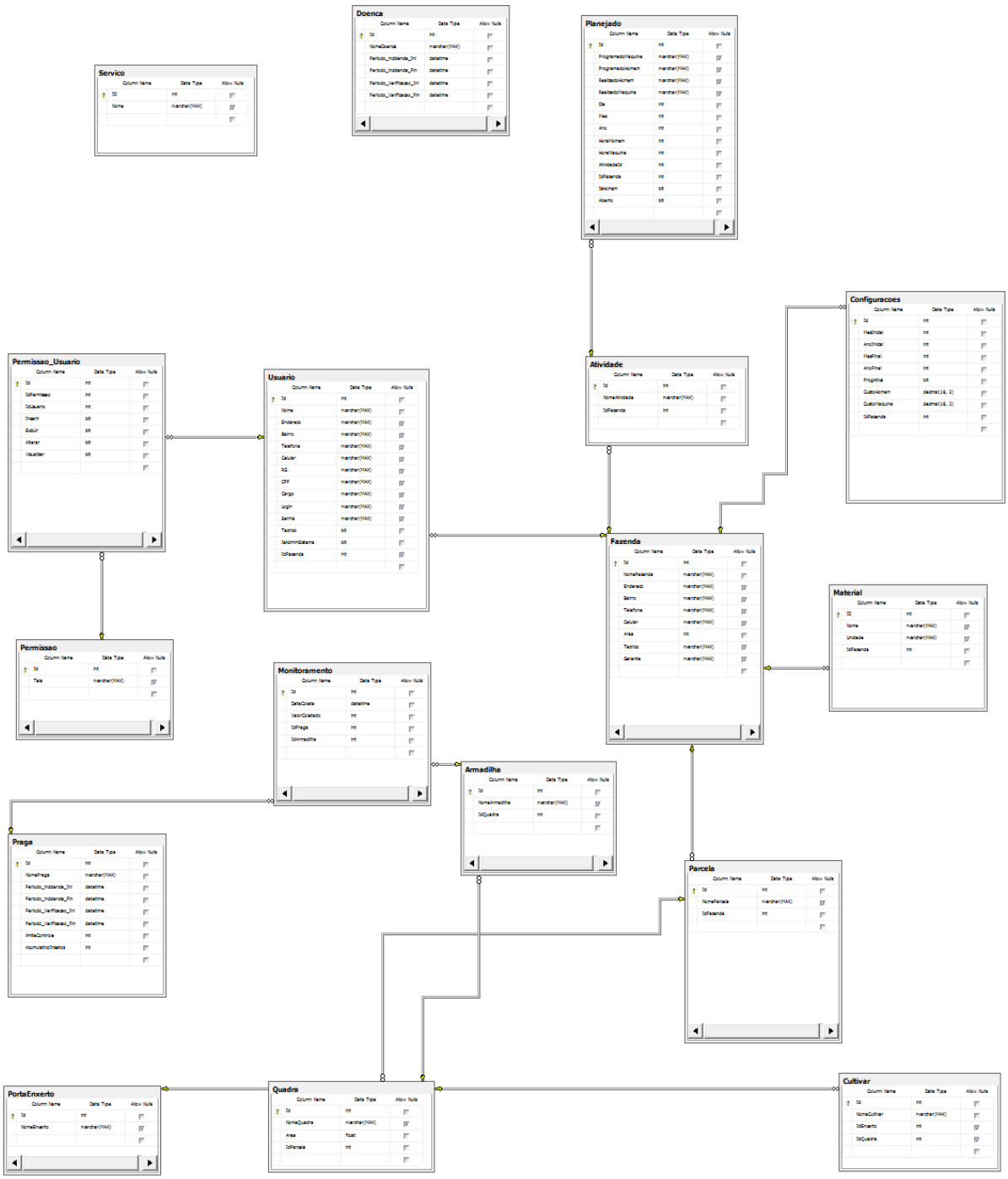
O desenvolvimento de software é uma atividade de crescente importância na sociedade contemporânea. A utilização de computadores nas mais diversas áreas do conhecimento humano tem gerado uma crescente demanda por soluções computadorizadas.

Para os iniciantes nas Ciências da Computação desenvolver software é, muitas vezes confundido com programação. Essa confusão inicial pode ser atribuída, parcialmente, pela forma como as pessoas são introduzidas nesta área de conhecimento, começando por desenvolver habilidades de raciocínio lógico, através de programação e estruturas de dados. Aliás, nada há de errado nessa estratégia. Começamos resolvendo pequenos problemas que gradativamente vão aumentando de complexidade, requerendo maiores conhecimentos e habilidades.

Na final do curso após adquirir conhecimentos, chegou-se ao final apresentando um programa desenvolvido com os conhecimentos adquiridos nas aulas teóricas do curso de Ciências da Computação do Centro Universitário UNIFACVEST, o programa foi desenvolvido em linguagem de fácil acesso e prático uso, para os proprietários de fazenda da região da Serra Catarinense, para o controle de pragas existentes nos pomares.

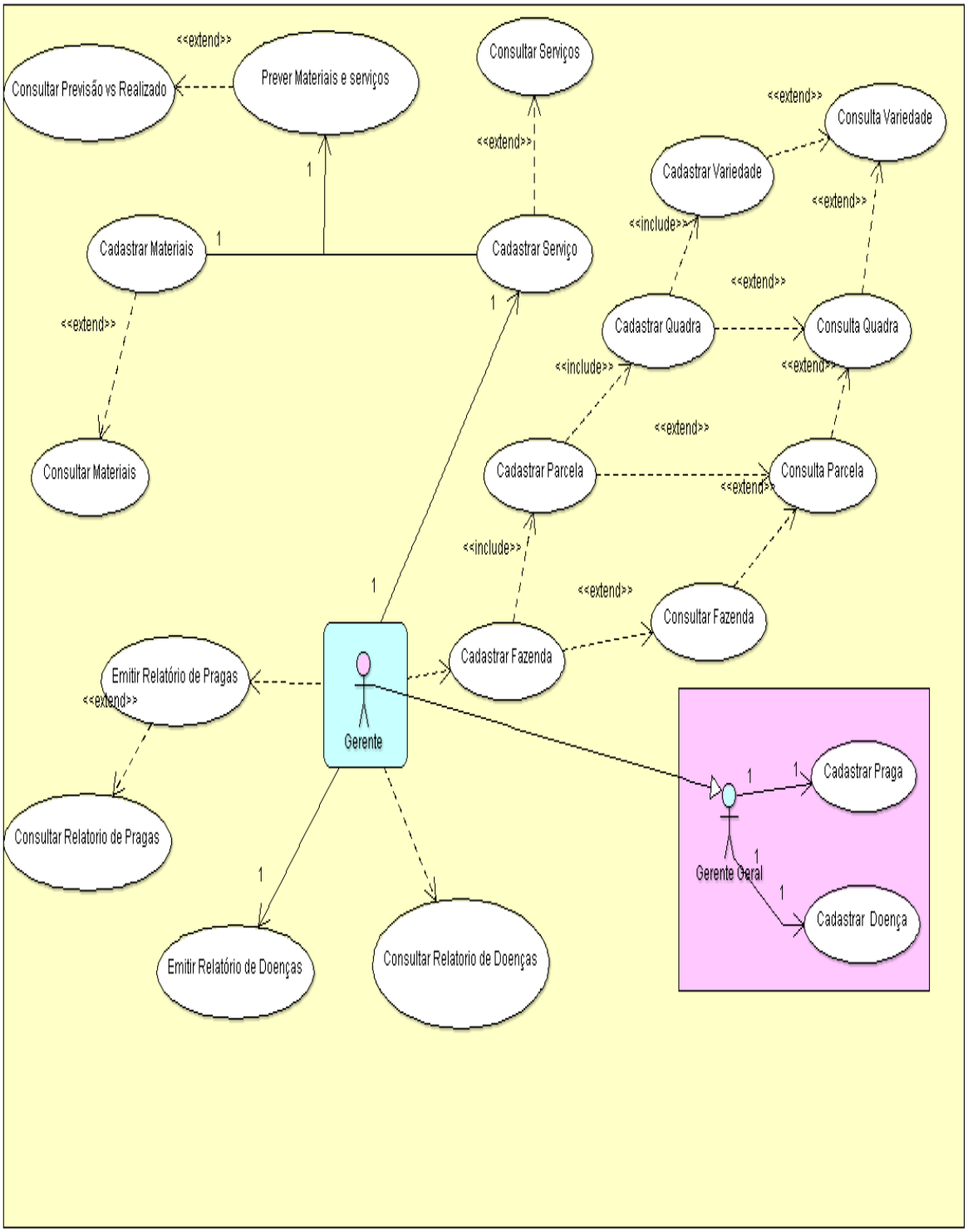
5.ANEXOS

5.1 ANEXO 1 – Diagrama de Banco de dados



Fonte Próprio Autor

5.2 ANEXO 2 – Diagrama de Caso de Uso



Fonte Próprio Autor

5.3 ANEXO 3 – Código Fonte Monitoramento de Pragas

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace Agronomia.Telas
{
    /// <summary>
    /// Interaction logic for PCadMonitoramentoPraga.xaml
    /// </summary>
    public partial class PCadMonitoramentoPraga : INotifyPropertyChanged
    {
        private bool alterar;
        private double _media;

        public double Media
        {
            get { return _media; }
            set
            {
                _media = value;
                OnPropertyChanged("Media");
            }
        }
        private List<Monitoramento> listaMonitoramento = new List<Monitoramento>();
        private int _idMonitoramento;
        private readonly Usuario _userLoged;
        public PCadMonitoramentoPraga(Usuario usu)
        {
            Usuario.PermissaoBotoes(usu, Permissions.VisualizarMoniPraga);
            InitializeComponent();
            RecarregarFazenda();
            CarregarPraga();
            CarregarMonitoramento();
        }

        private void RecarregarFazenda()
        {
            var db = new Contexto();
            var queryable = from f in db.Fazenda
                            select f;
            cbFazenda.Items.Add("Selecione uma fazenda...");
            foreach (var item in queryable)
                cbFazenda.Items.Add(item);


        }
```

```
        private void RecarregarParcela(int id)
        {
            var db = new Contexto();
            var queryable = from par in db.Parcela where par.IdFazenda == id select
par;
            cbParcela.Items.Clear();
            foreach (var item in queryable)
                cbParcela.Items.Add(item);
            cbParcela.SelectedIndex = 0;
        }

        private void RecarregarQuadra(int id)
        {
            var db = new Contexto();
            var queryable = from f in db.Quadra where f.IdParcela == id select f;
            cbQuadra.Items.Clear();

            foreach (var item in queryable)
                cbQuadra.Items.Add(item);
            cbQuadra.SelectedIndex = 0;
        }

        private void RecarregarArmadilha(int id)
        {
            var db = new Contexto();
            var queryable = from f in db.Armadilha where f.IdQuadra == id select f;
            cbArmadilha.Items.Clear();

            foreach (var item in queryable)
                cbArmadilha.Items.Add(item);
            cbArmadilha.SelectedIndex = 0;
        }

        private void CarregarPraga()
        {
            var db = new Contexto();
            var queryable = from f in db.Praga select f;
            cbPraga.Items.Clear();
            cbPraga.Items.Add("Selecione uma praga...");
            foreach (var item in queryable)
                cbPraga.Items.Add(item);
        }

        private void btNovo_Click(object sender, RoutedEventArgs e)
        {
            btNovo.IsEnabled = false;
            btSalvar.IsEnabled = true;
            btAlterar.IsEnabled = false;
            btExcluir.IsEnabled = false;
            HabilitarMonitoramento(true);
            LimparCamposMonitoramento();
            // tbNome.Focus();
        }

        private void LimparCamposMonitoramento()
        {
            cbArmadilha.SelectedIndex = 0;
            cbFazenda.SelectedIndex = 0;
            cbParcela.SelectedIndex = 0;
            cbPraga.SelectedIndex = 0;
            cbQuadra.SelectedIndex = 0;
            dpDateColeta.SelectedDate = DateTime.Now;
```

```
            NumQuantidade.Value = 0;
        }

        private void HabilitarMonitoramento(bool b)
        {
            cbArmadilha.IsEnabled = b;
            cbFazenda.IsEnabled = b;
            cbParcela.IsEnabled = b;
            cbPraga.IsEnabled = b;
            cbQuadra.IsEnabled = b;
            dpDateColeta.IsEnabled = b;
            NumQuantidade.IsEnabled = b;
        }

        private void btSalvar_Click(object sender, RoutedEventArgs e)
        {
            var fazenda = cbFazenda.SelectedItem as Fazenda;
            if (fazenda == null)
            {
                MessageBox.Show("Selecione uma fazenda válida.", "Fazenda",
MessageBoxButton.OK,
                                    MessageBoxImage.Information);
                return;
            }
            var parcela = cbParcela.SelectedItem as Parcela;
            if (parcela == null)
            {
                MessageBox.Show("Selecione uma parcela válida.", "Parcela",
MessageBoxButton.OK,
                                    MessageBoxImage.Information);
                return;
            }
            var quadra = cbQuadra.SelectedItem as Quadra;
            if (quadra == null)
            {
                MessageBox.Show("Selecione uma quadra válida.", "Quadra",
MessageBoxButton.OK,
                                     MessageBoxImage.Information);
                return;
            }
            var armadilha = cbArmadilha.SelectedItem as Armadilha;
            if (armadilha == null)
            {
                MessageBox.Show("Selecione uma armadilha válida.", "Armadilha",
MessageBoxButton.OK,
                                     MessageBoxImage.Information);
                return;
            }
            var praga = cbPraga.SelectedItem as Praga;
            if (praga == null)
            {
                MessageBox.Show("Selecione uma praga válida.", "Praga",
MessageBoxButton.OK,
                                    MessageBoxImage.Information);
                return;
            }


            var monitoramento = new Monitoramento();
            monitoramento.IdPraga = praga.Id;
            //monitoramento.IdQuadra = quadra.Id;
            monitoramento.IdArmadilha = armadilha.Id;
```

```
            if (dpDateColeta.SelectedDate != null) monitoramento.DataColeta =
dpDateColeta.SelectedDate.Value.Date;
            monitoramento.ValorColetado = NumQuantidade.Value.Value;

            if (!alterar)
                InserirMonitoramento(monitoramento);
            else
            {
                AlterarMonitoramento(monitoramento);
                    alterar = false;
            }
            btNovo.IsEnabled = true;
            btSalvar.IsEnabled = false;
            btAlterar.IsEnabled = false;
            btExcluir.IsEnabled = false;


        }

        private void AlterarMonitoramento(Monitoramento changeMon)
        {
            var db = new Contexto();
            var monitoramento = db.Monitoramento.First(e => e.Id ==
this._idMonitoramento);
            monitoramento.IdArmadilha = changeMon.IdArmadilha;
            monitoramento.DataColeta = changeMon.DataColeta;
            monitoramento.ValorColetado = changeMon.ValorColetado;
            monitoramento.IdPraga = changeMon.IdPraga;
            db.SaveChanges();
            MessageBox.Show("Monitoramento alterado com sucesso!");
            CarregarMonitoramento();
            LimparCamposMonitoramento();
            HabilitarMonitoramento(false);
        }

        private void InserirMonitoramento(Monitoramento monitoramento)
        {
            var db = new Contexto();
            db.Monitoramento.Add(monitoramento);
            db.SaveChanges();
            MessageBox.Show("Monitoramento cadastrada com sucesso!");
            CarregarMonitoramento();
            LimparCamposMonitoramento();
            HabilitarMonitoramento(false);
        }

        private void CarregarMonitoramento()
        {
            var db = new Contexto();
            var monitoramentos = from m in db.Monitoramento
                                 orderby
                                  m.DataColeta ascending,
                                 m.Armadilha.Quadra.Parcela.Fazenda.NomeFazenda,
                                 m.Armadilha.Quadra.Parcela.NomeParcela,
                                 m.Armadilha.Quadra.NomeQuadra,
                                 m.Armadilha.NomeArmadilha
                                 select m;
            listaMonitoramento.Clear();
            var list = new List<Monitoramento>();
            foreach (var item in monitoramentos)
            {
                list.Add(item);
```

```
                listaMonitoramento.Add(item);
            }
            //Calcular a média
            var listAcumulativa = new List<Monitoramento>();
            foreach (var item in listaMonitoramento)
            {
                //var nomeitem = item.Armadilha.Quadra.Parcela.Fazenda.NomeFazenda +
                //item.Armadilha.Quadra.Parcela.NomeParcela + item.DataColeta;
                double med =
                    list.Count(
                        l =>
                        l.Armadilha.Quadra.Parcela.Fazenda.Id ==
                        item.Armadilha.Quadra.Parcela.Fazenda.Id &&
                        l.Armadilha.Quadra.Parcela.Id ==
item.Armadilha.Quadra.Parcela.Id &&
                        l.DataColeta == item.DataColeta);
                var listSum = list.Where(l => l.Armadilha.Quadra.Parcela.Fazenda.Id ==
item.Armadilha.Quadra.Parcela.Fazenda.Id && l.Armadilha.Quadra.Parcela.Id ==
item.Armadilha.Quadra.Parcela.Id && l.DataColeta == item.DataColeta);
                double total = listSum.Aggregate<Monitoramento, double>(0, (current,
sum) => current + sum.ValorColetado);

                item.Media = Math.Round(total / med, 1);
                item.CorForeground = item.Media >= item.Praga.limiteControle ? new
SolidColorBrush(Colors.Red) : new SolidColorBrush(Colors.White);

                #region Calculo Sete Dias

                var enumerable =
                    listaMonitoramento.Where(l => l.DataColeta < item.DataColeta &&
l.Armadilha.Quadra.Parcela.Fazenda.Id ==
                        item.Armadilha.Quadra.Parcela.Fazenda.Id &&
                        l.Armadilha.Quadra.Parcela.Id ==
item.Armadilha.Quadra.Parcela.Id).OrderBy(p => p.DataColeta)
                        .ToList();
                if (enumerable.Count == 0)
                {
                    item.SeteDias = Math.Round(item.Media, 1);
                    item.CorForegroundSete = item.SeteDias >=
item.Praga.limiteControle
                                                ? new SolidColorBrush(Colors.Red)
                                                : new SolidColorBrush(Colors.White);
                }
                else
                {
                    item.SeteDias = Math.Round((enumerable.Last().Media + item.Media),
1);
                    item.CorForegroundSete = item.SeteDias >=
item.Praga.limiteControle ? new SolidColorBrush(Colors.Red) : new
SolidColorBrush(Colors.White);

                }

                #endregion

                var existeLista = listAcumulativa.Any(l => l.DataColeta ==
item.DataColeta && item.Armadilha.Quadra.Parcela.Fazenda.Id ==
l.Armadilha.Quadra.Parcela.Fazenda.Id && item.Armadilha.Quadra.Parcela.Id ==
l.Armadilha.Quadra.Parcela.Id);
                if (!existeLista)
                    listAcumulativa.Add(item);
```

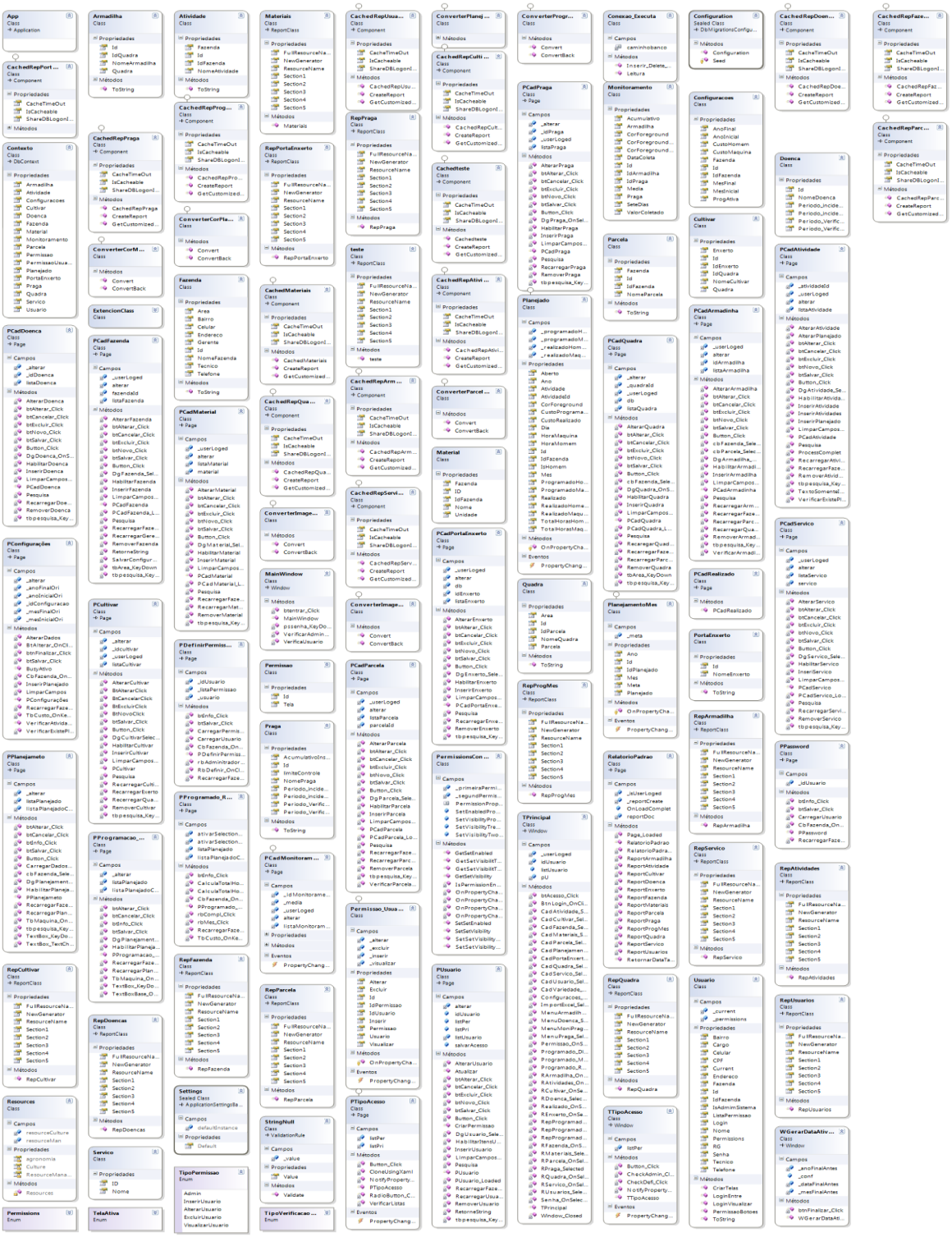
```
                //foreach (var item2 in list)
                //{
                //    var nomeitem2 =
item2.Armadilha.Quadra.Parcela.Fazenda.NomeFazenda +
                //                  item2.Armadilha.Quadra.Parcela.NomeParcela +
item2.DataColeta;
                //    if (item.Id != item2.Id)
                //        if (nomeitem.Equals(nomeitem2))
                //        {
                //            item.Media = Math.Round(total / med,1);
                //            item.CorForeground = item.Media >=
item.Praga.limiteControle ? new SolidColorBrush(Colors.Red) : new
SolidColorBrush(Colors.White);
                //            var enumerable =
                //                listaMonitoramento.Where(l => l.DataColeta <
item2.DataColeta).OrderBy(p => p.DataColeta)
                //                    .ToList();
                //            if (enumerable.Count == 0)
                //                item.SeteDias = item.Media;
                //            else
                //            {
                //                item.SeteDias = (enumerable.Last().Media +
item.Media);
                //                item.CorForegroundSete = item.SeteDias >=
item.Praga.limiteControle ? new SolidColorBrush(Colors.Red) : new
SolidColorBrush(Colors.White);

                //            }
                //        }
                //}
            }

            foreach (var item in listaMonitoramento)
                foreach (var item2 in listAcumulativa)
                    if (item2.DataColeta <= item.DataColeta &&
item.Armadilha.Quadra.Parcela.Fazenda.Id == item2.Armadilha.Quadra.Parcela.Fazenda.Id
&& item.Armadilha.Quadra.Parcela.Id == item2.Armadilha.Quadra.Parcela.Id)
                    {
                        item.Acumulativo = Math.Round(item.Acumulativo +
item2.SeteDias, 1);
                        item.CorForegroundAcumulativo = item.Acumulativo >
item.Praga.AcumulativoInsetos ? new SolidColorBrush(Colors.Red) : new
SolidColorBrush(Colors.White);
                    }



            var listCollectionView = new ListCollectionView(listaMonitoramento);
            if (listCollectionView.GroupDescriptions != null)
            {
                // listCollectionView.GroupDescriptions.Add(new
PropertyGroupDescription("Praga.NomePraga"));
                listCollectionView.GroupDescriptions.Add(new
PropertyGroupDescription("Armadilha.Quadra.Parcela.Fazenda.NomeFazenda"));
                // listCollectionView.GroupDescriptions.Add(new
PropertyGroupDescription("Armadilha.Quadra.Parcela.NomeParcela"));
            }

            Dgmonitoramento.ItemsSource = null;
            Dgmonitoramento.ItemsSource = listCollectionView;
```

```
            //Dgmonitoramento.ItemsSource = from m in listaMonitoramento
            //                              orderby
            //                              m.DataColeta descending,
            //
m.Armadilha.Quadra.Parcela.Fazenda.NomeFazenda,
            //                              m.Armadilha.Quadra.Parcela.NomeParcela,
            //                              m.Armadilha.Quadra.NomeQuadra,
            //                              m.Armadilha.NomeArmadilha
            //                              select m;
        }

        private void btAlterar_Click(object sender, RoutedEventArgs e)
        {
            alterar = true;
            btSalvar.IsEnabled = true;
            btAlterar.IsEnabled = false;
            btExcluir.IsEnabled = false;
            HabilitarMonitoramento(true);
        }

        private void btExcluir_Click(object sender, RoutedEventArgs e)
        {
            var mon = Dgmonitoramento.SelectedItem as Monitoramento;
            if (mon == null) return;
            var ret = MessageBox.Show("Deseja realmente excluir o Monitoramento da " +
mon.Praga.NomePraga + " do dia " + mon.DataColeta+ "?", "EXCLUIR",
MessageBoxButton.OKCancel, MessageBoxImage.Question);
            if (ret == MessageBoxResult.OK)
                RemoverMonitoracao();
        }

        private void RemoverMonitoracao()
        {
            var db = new Contexto();
            var monitoramento = db.Monitoramento.First(e => e.Id ==
this._idMonitoramento);
            if (monitoramento == null)return;
            db.Monitoramento.Remove(monitoramento);
            db.SaveChanges();
            MessageBox.Show("Monitoramento removido com sucesso!");
            CarregarMonitoramento();
            LimparCamposMonitoramento();
            HabilitarMonitoramento(false);
        }

        private void btCancelar_Click(object sender, RoutedEventArgs e)
        {
            btNovo.IsEnabled = true;
            btSalvar.IsEnabled = false;
            btExcluir.IsEnabled = false;
            btAlterar.IsEnabled = false;
            HabilitarMonitoramento(false);
            LimparCamposMonitoramento();
            Dgmonitoramento.SelectedItem = null;
        }

        private void Pesquisa()
        {
            var enumerable = listaMonitoramento.Where(s =>
s.Armadilha.NomeArmadilha.ToLower().Contains(tbpesquisa.Text.ToLower())||

s.Armadilha.Quadra.NomeQuadra.ToLower().Contains(tbpesquisa.Text.ToLower())||
```

```
s.Armadilha.Quadra.Parcela.NomeParcela.ToLower().Contains(tbpesquisa.Text.ToLower())||

s.Armadilha.Quadra.Parcela.Fazenda.NomeFazenda.ToLower().Contains(tbpesquisa.Text.ToLo
wer())||
                s.DataColeta.ToString().Contains(tbpesquisa.Text.ToLower()));

            Dgmonitoramento.ItemsSource = null;
            Dgmonitoramento.ItemsSource = enumerable;
        }

        private void tbpesquisa_KeyDown(object sender, KeyEventArgs e)
        {
            switch (e.Key)
            {
                case Key.Enter:
                    Pesquisa();
                    break;
                case Key.Escape:
                    tbpesquisa.Text = string.Empty;
                    Pesquisa();
                    break;
            }
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            Pesquisa();
        }

        private void Dgmonitoramento_OnSelectedCellsChanged(object sender,
SelectedCellsChangedEventArgs e)
        {
            var monitoramento = Dgmonitoramento.SelectedItem as Monitoramento;
            if (monitoramento != null)
            {
                this._idMonitoramento = monitoramento.Id;
                btExcluir.IsEnabled = true;
                btAlterar.IsEnabled = true;
                cbFazenda.SelectedValue =
monitoramento.Armadilha.Quadra.Parcela.Fazenda;
                cbParcela.SelectedValue = monitoramento.Armadilha.Quadra.Parcela;
                cbQuadra.SelectedValue = monitoramento.Armadilha.Quadra;
                cbArmadilha.SelectedValue = monitoramento.Armadilha;
                cbPraga.SelectedValue = monitoramento.Praga;
                NumQuantidade.Value = monitoramento.ValorColetado;
                dpDateColeta.SelectedDate = monitoramento.DataColeta;
            }
        }

        private void cbFazenda_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            var fazenda = cbFazenda.SelectedItem as Fazenda;
            if (fazenda != null)
            {
                cbParcela.Items.Clear();
                cbQuadra.Items.Clear();
                cbArmadilha.Items.Clear();
                RecarregarParcela(fazenda.Id);
            }
        }
```

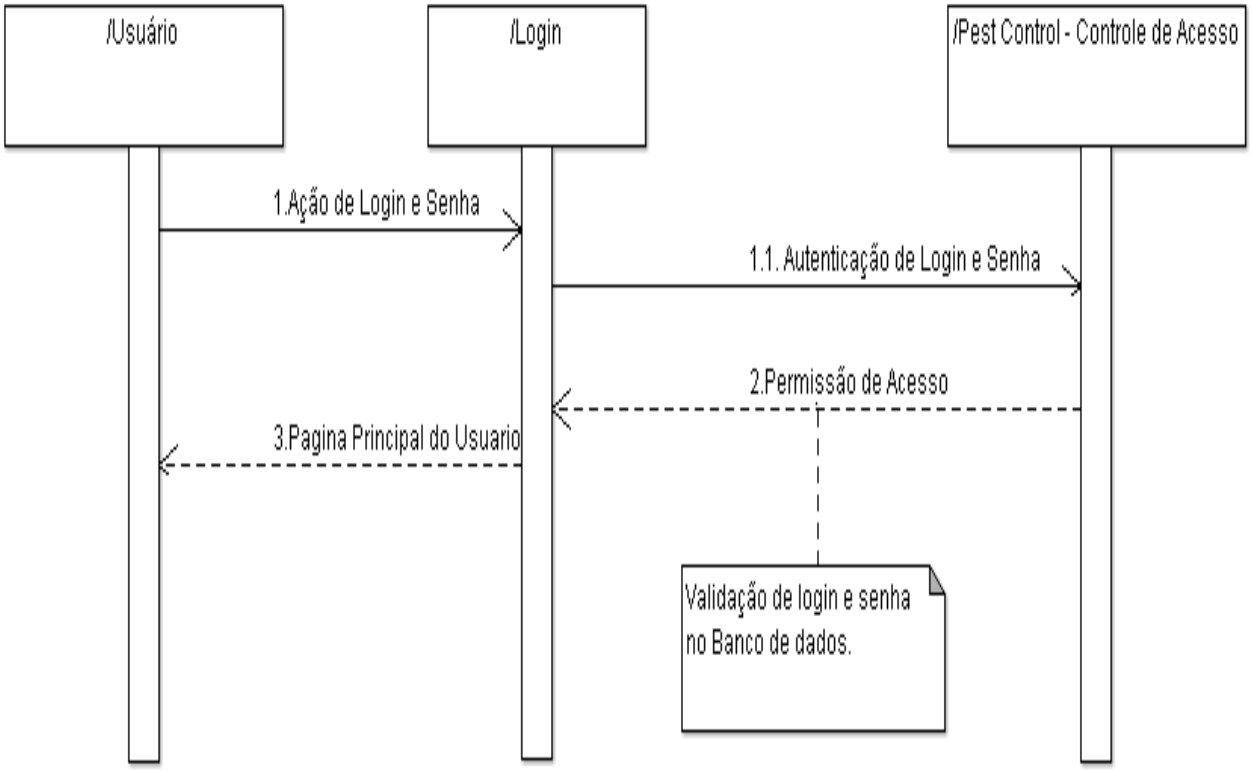
```
        private void cbParcela_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            var parcela = cbParcela.SelectedItem as Parcela;
            if (parcela != null)
            {
                cbQuadra.Items.Clear();
                cbArmadilha.Items.Clear();
                RecarregarQuadra(parcela.Id);
            }
        }

        private void cbQuadra_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            var quadra = cbQuadra.SelectedItem as Quadra;
            if (quadra != null)
            {
                cbArmadilha.Items.Clear();
                RecarregarArmadilha(quadra.Id);
            }
        }


        public event PropertyChangedEventHandler PropertyChanged;

        protected virtual void OnPropertyChanged(string propertyName)
        {
            PropertyChangedEventHandler handler = PropertyChanged;
            if (handler != null) handler(this, new
PropertyChangedEventArgs(propertyName));
        }
    }
}
```

5.4 ANEXO 4 – Diagrama de Classes



Fonte: Visual Studio 2010

5.4 ANEXO 4 – Diagrama de Sequência – Login do Usuário



Fonte: Próprio Autor 1

REFERÊNCIAS BIBLIOGRÁFICAS

BRAZ JÚNIOR, O. O. **Técnicas de análise de sistema** (apostila). Tubarão: UNISUL, 1997.

CARNEIRO, M. F. S. **Gerenciamento de projetos** (apostila), ENAP, 2000.

CARVALHO, Victorio Albani de; TEIXEIRA, Giovany Frossard. **Programação orientada a objetos:** Curso técnico de informática. Colatina: IFES, 2012. 134 p. Disponível em: <http://redeetec.mec.gov.br/images/stories/pdf/eixo_infor_comun/tec_inf/081112_progr_obj.pdf>. Acesso em: 16 out. 2013.

DATE, C. J. **Introdução a sistemas de bancos de dados**. Tradução da 7 ed. americana Vanderberg Dantas de Souza. Publicare consultoria e serviços. Rio de Janeiro: Campus, 2000.

DEITEL, H. M. **Como programar**. São Paulo: Makron Books, 2003.

GAMMA, E. et al. **Padrões de Projetos:** soluções reutilizáveis de software orientado a objetos. Porto Alegre: Bookman, 2000.

JAVA Virtual Machine e Seus Conceitos (para iniciantes). Disponível em: <http://www.vivaolinux.com.br/dica/Java-Virtual-Machine-e-seus-conceitos-(para-iniciantes)>. Acesso em: 21 out. 2013.

MANZANO, J. A. N. G. **Microsoft SQL Server 2008 express**: interativo: guia prático. São Paulo: Érica, 2009.

PARREIRA JÚNIOR, W. M. **Apostila engenharia de software.** Disponível em: <http://www.waltenomartins.com.br/ap_es_v1.pdf>. Acesso em: 07 out. 2013.

PAULA FILHO, W. P. **Engenharia de software**. Fundamentos e métodos e padrões. Rio de Janeiro: LTC Editora, 2001.

PRESSMAN, R. S. **Engenharia de software.** 6. ed. São Paulo: Mcgraw-hill, 2006.

_____. **Software Engineering:** a Practitioner's Approach. 7. ed. New York: Mcgraw-hill, 2010. Disponível em: <http://home.aubg.bg/students/ENI100/Software-Engineering.pdf>. Acesso em: 15 out. 2013.

SOMMERVILLE, Ian. **Engenharia de Software.** 9. ed. São Paulo: Pearson, 2011.

_____. **Engenharia de software**. 6. Ed. São Paulo: Addison Wesley, 2003.