

BRUNA HITOMI FUKUYOSHI

PROGRAMA PARA GERENCIAMENTO DE REBANHO LEITEIRO

BRUNA HITOMI FUKUYOSHI

PROGRAMA PARA GERENCIAMENTO DE REBANHO LEITEIRO

Trabalho de Graduação apresentado ao Conselho de Curso de Graduação em Engenharia Elétrica da Faculdade de Engenharia do *Campus* de Guaratinguetá, Universidade Estadual Paulista, como parte dos requisitos para obtenção do diploma de Graduação em Engenharia Elétrica.

Orientador: Prof. Dr. Samuel E. de Lucena

Guaratinguetá
2012

F961p	<p>Fukuyoshi, Bruna Hitomi</p> <p>Programa para gerenciamento de rebanho leiteiro / Bruna Hitomi Fukuyoshi – Guaratinguetá : [s.n], 2012. 54 f : il. Bibliografia: f. 53-54</p> <p>Trabalho de Graduação em Engenharia Elétrica – Universidade Estadual Paulista, Faculdade de Engenharia de Guaratinguetá, 2012. Orientador: Prof. Dr. Samuel E. de Lucena</p> <p>1. C# (Linguagem de programação de computador) 2. USB (Barramento de computador) 3. Visual Studio (Programa de computador) I. Título</p> <p>CDU 519.682C</p>
-------	---

PROGRAMA PARA GERENCIAMENTO DE REBANHO LEITEIRO

BRUNA HITOMI FUKUYOSHI


ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO
PARTE DO REQUISITO PARA A OBTENÇÃO DO DIPLOMA DE
"GRADUANDO EM ENGENHARIA ELÉTRICA"

APROVADO EM SUA FORMA FINAL PELO CONSELHO DE CURSO DE
GRADUAÇÃO EM ENGENHARIA ELÉTRICA


Prof. Dr. LEONARDO MESQUITA
Coordenador

BANCA EXAMINADORA:


Prof. Dr. SAMUEL EUZEDICE DE LUCENA
Orientador/UNESP-FEG


Prof. Dr. FRANCISCO ANTONIO LOTUFO
UNESP-FEG


Prof. Msc. FERNANDO RIBEIRO FILADELFO
UNESP-FEG

Dezembro 2012

DADOS CURRICULARES

BRUNA HITOMI FUKUYOSHI

NASCIMENTO	19.02.1989 – SÃO PAULO / SP
FILIAÇÃO	Marcos Ken Itti Fukuyoshi Eliana Maria de Jesus
2008/2012	Curso de Graduação Universidade Estadual Paulista – “Júlio de Mesquita Filho”, <i>Campus</i> de Guaratinguetá.

DEDICATÓRIA

Dedico a minha família que sempre esteve ao meu lado em todos os momentos tristes e alegres e me ensinou a sempre correr atrás dos meus objetivos.

AGRADECIMENTOS

Agradeço aos meus tios, Edson e Isabeti, por serem mais que pais para mim, por sempre estarem do meu lado, acreditarem no meu potencial, me incentivarem a nunca desistir dos meus objetivos. Agradeço a eles também por nunca ter deixado me faltar nada e por terem passado para mim uma ótima base educacional.

Agradeço ao meu primo, Felipe, por ser como um irmão e por me mostrar o quanto um é importante para o outro. Agradeço a minha família por sempre me apoiarem e terem me ensinado muitas coisas.

Agradeço ao meu namorado, André, por estar do meu lado nesses cinco anos, ter me dado força nos momentos difíceis, por sempre me motivar a não desistir e sempre me dar bons conselhos. Agradeço aos meus amigos por estarem comigo nos meus momentos alegres e tristes.

Agradeço a República das Ursas por ter feito parte desses cinco anos tão importantes na minha vida, ter me proporcionado um grande amadurecimento e possibilitado o surgimento de grandes amizades.

FUKUYOSHI, B. H. Programa para gerenciamento de rebanho leiteiro. 2012. 54 f. Trabalho de Graduação (Graduando em Engenharia Elétrica) – Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2012.

RESUMO

Com o advento da tecnologia, muitos processos antes repetitivos passaram a ser simplificados com o auxílio de um simples computador. Para tanto a programação vem sendo desenvolvida e novos *softwares* aparecem a cada dia. Para o desenvolvimento desse projeto utilizou-se o Microsoft Visual Studio 2010, e com ele o programa foi feito na linguagem C# no modo Windows Form. Esse trabalho visa desenvolver um programa que consiga ler, receber e enviar dados através da porta *Universal Serial Bus* (USB) para o computador, e através disso realizar o tratamento dos dados. Com isso o produtor de leite possuirá o conhecimento da quantidade de leite produzido pelo animal. Com o tratamento de dados é possível gerar gráficos da produção do rebanho ou de um animal específico e também é possível gerar o relatório da data requerida.

PALAVRAS-CHAVE: Linguagem C#; USB; Interface Gráfica; Visual Studio

UKUYOSHI, B. H. Program for dairy herd management.2012. 54 p. Graduate Work (Graduate in Electrical Engineering) - Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2012.

ABSTRACT

With the advent of technology, many repetitive processes before now was be simplified with the support of a simple computer. Therefore programming has been developed and new softwares appear every day. For the development of this project was used the Microsoft Visual Studio 2010, and with it the program was made in C # Windows Form mode. This work aims to develop a program that can read, send and receive data via the Universal Serial Bus (USB) to the computer, and thereby perform the processing. Thus the milk producer will possess the knowledge of the amount of milk produced by the animal. With the processing of data it is possible to generate graphs of herd production or a specific animal and you can also generate the required report date.

KEYWORDS: C# Language; USB; Graphic Interface; Visual Studio

LISTA DE FIGURAS

Figura 1 – Fluxograma (Fonte: autor).....	20
Figura 2 -Bloco do Sistema (Fonte: autor).....	21
Figura 3 - Exemplo de Interface (Fonte: autor)	23
Figura 4 - Tela de Abertura do Microsoft Visual Studio (Fonte: autor)	26
Figura 5 - Tela Inicial do Visual Studio (Fonte: autor).....	27
Figura 6 – Gerar Novo Projeto (Fonte: autor).....	28
Figura 7 - Windows Form Application (Fonte: autor)	29
Figura 8 - Local da Montagem da Interface (Fonte: autor).....	29
Figura 9 - Caixa de Ferramentas (Fonte: autor)	30
Figura 10 - Barra Propriedades (Fonte: autor)	31
Figura 11 -Brinco de Identificação do animal (Fonte: http://www.abhb.com.br acessado em 17/12/2012)	33
Figura 12 - Animal com o brinco (Fonte: http://www.tecnologiaetreinamento.com.br acessado em 17/12/2012)	33
Figura 13 - Conexão Aparelho e Computador (Fonte: autor)	34
Figura 14 - Esquema Relatório (Fonte: autor)	35
Figura 15 - Esquema Histórico do Rebanho (Fonte: autor)	35
Figura 16 - Esquema Relatório Vaca (Fonte: autor)	35
Figura 17 - Passos para Gerar Relatório (Fonte: autor)	35
Figura 18 - Passos para Gerar Histórico (Fonte: autor)	36
Figura 19 -Passos para Gerar Relatório Vaca (Fonte: autor)	36
Figura 20 - Passos para Salvar os Dados (Fonte: autor)	36
Figura 21 - Interface Principal do Programa (Fonte: autor).....	42
Figura 22 - Interface Verificação Existência da Vaca (Fonte: autor)	42
Figura 23 - Caixa de dialogo (Fonte: autor).....	43
Figura 24 - Interface Relatório Vaca (Fonte: autor).....	44
Figura 25 - Interface Histórico do Rebanho (Fonte: autor).....	45
Figura 26 - Interface Relatório (Fonte: autor).....	46
Figura 27 - Inserção de Dados (Fonte: autor)	48
Figura 28 - Relatório da Vaca (Fonte: autor).....	50
Figura 29 - Relatório do Rebanho (Fonte: autor).....	50
Figura 30 - Relatório (Fonte: autor).....	51

LISTA DE QUADROS

Quadro 1 Tipos de Relação	18
Quadro 2 - Tipos de Lógica	18
Quadro 3 - Símbolos de Diagramas de Fluxo	19

LISTA DE TABELAS

Tabela 1 - Histórico.....	47
Tabela 2 - Atividade.....	47
Tabela 3 - Vaca	48

SUMÁRIO

1	INTRODUÇÃO	13
2	CONCEITUALIZAÇÃO DE PROGRAMAÇÃO.....	15
2.1	Algoritmo	15
2.1.1	Tipos de Instrução	15
2.1.2	Estrutura do Algoritmo	16
2.1.3	Fluxograma	18
2.2	Programa	21
2.3	Interface Gráfica.....	22
2.4	Linguagem de Programação C#.....	23
2.4.1	Estrutura da linguagem C#.....	24
2.5	Microsoft Visual Studio	25
2.5.1	Microsoft Visual C#.....	27
2.5.2	Windows FormsApplication	28
3	METODOLOGIA	32
3.1	Atividades Preliminares	32
3.2	Desenvolvimento do projeto	34
3.2.1	Pré-projeto.....	34
3.2.2	Desenvolvimento do Programa	36
3.2.2.1	Biblioteca	37
3.2.2.2	Programação.....	38
3.2.2.3	Interface.....	41
4	RESULTADOS E DISCUSSÃO	47
5	CONCLUSÃO	52
	REFERÊNCIAS BIBLIOGRÁFICAS.....	53
	BIBLIOGRAFIA CONSULTADA	54

1 INTRODUÇÃO

A partir do período em que se conseguiu diminuir o tamanho dos computadores de laboratório, e eles passaram a ser comercializados para uso em aplicações empresariais e pessoais, as linguagens de programação foram cada vez mais necessárias para poder comunicar informações aos computadores e ao usuário de uma forma que o usuário conseguisse entender; com isso foram surgindo novos tipos de linguagens cada vez mais aprimoradas, as quais facilitam a vida do usuário na utilização de todos os sistemas contidos nos computadores.

A linguagem de programação também é utilizada para realização de programas, os quais são desenvolvidos para facilitar a vida das pessoas, pois reduz a quantidade de rotinas a serem feitas e do tempo que seria gasto para o usuário obter a informação desejada.

O programa em questão é desenvolvido para que pessoas leigas possam utilizá-lo sem dificuldades; é um programa de fácil entendimento e simples utilização, pois possui uma interface gráfica simples e didática, além de ser baseado no sistema Windows, o mais utilizado e comercializado nos dias de hoje, o que facilita a implantação do programa.

O programa produz um gráfico de quanto o animal leiteiro em questão produziu, através do fornecimento dos parâmetros: quantidade de leite produzido, número de ordenhas, e número do animal. Também irá gerar o gráfico geral da produção de leite da fazenda. Os parâmetros para produzir os gráficos são enviados por um equipamento eletrônico que será conectado ao computador e assim enviará os dados para ele. Com os dados recebidos e salvos em arquivos texto, o programa realiza o tratamento destes dados e gera os gráficos da produção.

Para o desenvolvimento do projeto utilizou-se o Microsoft Visual Studio 2010, e com ele o programa foi feito na linguagem C# no modo Windows form.

O presente trabalho tem por objetivo desenvolver um programa que consiga receber dados da produção leiteira de uma fazenda, armazenados em um coletor de dados, através da porta *Universal Serial Bus* (USB), e realizar o tratamento dos dados, organizando a informação recebida na forma de tabelas e gráficos, para gerar relatórios que auxiliarão o produtor de leite no gerenciamento do rebanho, com vistas a aumentar a produtividade da fazenda.

Para o desenvolvimento do trabalho, o mesmo foi dividido em conceitualização de programação, metodologia, discussão e resultados e conclusão, onde:

- **Conceitualização de Programação:** neste tópico serão abordados os principais conceitos necessários para desenvolver a parte de programação do projeto;
- **Metodologia:** neste tópico será abordada a resolução da problemática proposta;
- **Discussão e Resultados:** neste tópico serão apresentadas as interfaces resultantes com suas devidas discussões;
- **Conclusão:** neste tópico será apresentada uma análise crítica do projeto.

2 CONCEITUALIZAÇÃO DE PROGRAMAÇÃO

Este capítulo tem como objetivo reunir todos os conceitos necessários para o desenvolvimento da programação do trabalho.

2.1 Algoritmo

Algoritmo é uma sequência de instruções que segue a lógica do problema abordado sem possuir ambiguidade e estas devem ser bem definidas, todavia antes de realizar o algoritmo deve ser definido o problema.

Segundo Campos (2007, p. 4)

A descrição de um algoritmo, por intermédio de uma notação algorítmica, melhora o seu entendimento, pois apenas os aspectos do raciocínio lógico são enfatizados, sem serem necessários os detalhes de implementação de uma linguagem de programação.

2.1.1 Tipos de Instrução

As instruções básicas e que podem ser aplicadas de modo geral no algoritmo e podem ser utilizadas em qualquer linguagem são:

- Instrução de início/fim;
- Instrução de atribuição;
- Instrução de leitura;
- Instrução de escrita;
- Instrução de bifurcação: ela pode ser negativa ou positiva e pode ser do tipo condicional ou incondicional.

Essas instruções serão mais bem explicadas no tópico seguinte.

2.1.2 Estrutura do Algoritmo

A estrutura do algoritmo deve conter um começo, um meio e um fim, e deve seguir algumas regras:

- Iniciar: **Algoritmo** <nome_do_algoritmo>
- Finalizar: **fim algoritmo**
- Objetivo: {**Objetivo:**<objetivo_do_algoritmo>}
- Parâmetros de entrada: **parâmetro de entrada** <lista_de_variáveis>
- Parâmetros de saída: **parâmetros de saída** <lista_de_variáveis>

Segundo Campos (2007, p. 5)

Uma variável corresponde a uma posição de memória do computador onde está armazenado um determinado valor. As variáveis são representadas por identificadores que são cadeias de caracteres alfanuméricos, podendo os elementos de vetores e matrizes serem referenciados por subscritos ou índices. Por exemplo, v_i ou $v(i)$ e m_{ij} ou $m(i,j)$.

Para realizar um comentário em qualquer parte do algoritmo, este deve ser delimitado por chaves {<comentário>}.

Para atribuir algumas expressões e comandos, devem ser seguidas as seguintes regras:

- Expressões aritméticas: <variável> \leftarrow <expressão>
- Comando de leitura: **leia**<lista_de_variáveis>
- Comando de escrita: **escreva**<lista_de_variáveis>
- Estruturas condicionais

➤ Simples: **se** <condição>**então**

<comandos>

fim se

➤ Composta: **se** <condição>**então**

<comando_1>

senão

<comando_2>

fim se

- Estruturas de repetições

- Com número indefinido de repetições:

repita

<comando_1>

se<condição>então

interrompa

fim se

<comando_2>

fim repita

<comando_3>

- Com número definido de repetições:

para<controle>←<valor_inicial>até<valor_final>passo<delta>faça

<comandos>

fim para

Para realizar operações de relação dentro de uma estrutura, podem ser usados os símbolos do quadro 1.

Quadro 1 Tipos de Relação

Símbolo	Descrição
$>$	Maior que
\geq ou $> =$	Maior ou igual a
$<$	Menor que
\leq ou $< =$	Menor ou igual a
$=$	Igual a
\neq ou \diamond	Diferente de

Para realizar operações lógicas dentro de uma estrutura, os símbolos a serem usados estão no quadro 2.

Quadro 2 - Tipos de Lógica

Símbolo	Descrição
E	Usado quando deve ocorrer uma coisa e outra
Ou	Usado quando pode ocorrer uma coisa ou outra
Não	Usado em caso de negação

O resultado das operações lógicas e das de relação podem ser verdadeiro ou falso dentro do comando que essas estão inseridas.

Campos (2007, p. 12) fala que o comando de falha no algoritmo é usado para indicar a ocorrência de uma falha evidente na execução do algoritmo. Então, neste caso, a execução será cancelada. Caso o programa utilizado não possua o comando **abandono**, o programador deve implementá-lo.


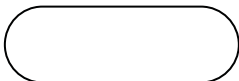
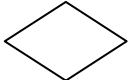






2.1.3 Fluxograma

O fluxograma ou diagrama de fluxo é um artifício utilizado que ajuda o programador a visualizar melhor as ações que irão ocorrer na implementação. Ele mostra graficamente a sequência em que as operações são executadas no problema que está sendo trabalhado e como cada passo está interligado com os demais. Com o diagrama, é possível se ter uma descrição muito clara do problema.

O fluxograma é constituído por símbolos (caixas) padrão, onde cada um possui a sua função.

No quadro3 temos alguns símbolos que são:

Quadro 3 - Símbolos de Diagramas de Fluxo

Símbolo	Função	Descrição
	Linha de fluxo	Utilizado para realizar a ligação entre as operações e mostrar o fluxo da lógica do programa.
	Terminal	É utilizado para indicar o início e fim do processo.
	Decisão	Usado para tomada de decisões (possui duas saídas, uma verdadeira e outra falsa).
	Processo	Utilizado para realização de operações que cause mudança.
	Conector de rotina	Liga diferentes linhas de fluxo.
	Entrada/Saída	Usado para ações de entrada e saída, que pode ser leitura e impressão.
	Conector de página	Quando ocorre a mudança de página do fluxograma, utiliza-se esse conector para ligar o final de uma página com o início de outra.
	Entrada Manual	Indica entrada de dados via teclado.
	Exibir	Mostra informações ou resultados.

O fluxograma, depois de montado, permite que o programador consiga realizar a programação com mais facilidade, pois ele mostra passo a passo as ações que ele deve implementar para conseguir que o programa funcione corretamente. Em muitos casos a construção de um fluxograma é necessária para o programador não se perder no meio da programação. Apesar da construção do diagrama requerer tempo, com a construção dele haverá uma diminuição de tempo no desenvolvimento do programa.

A figura 1 mostra um exemplo de um diagrama de fluxo, onde ele mostra a leitura de três parâmetros (A, B, C) e a verificação de qual é o maior e o menor entre os três números.

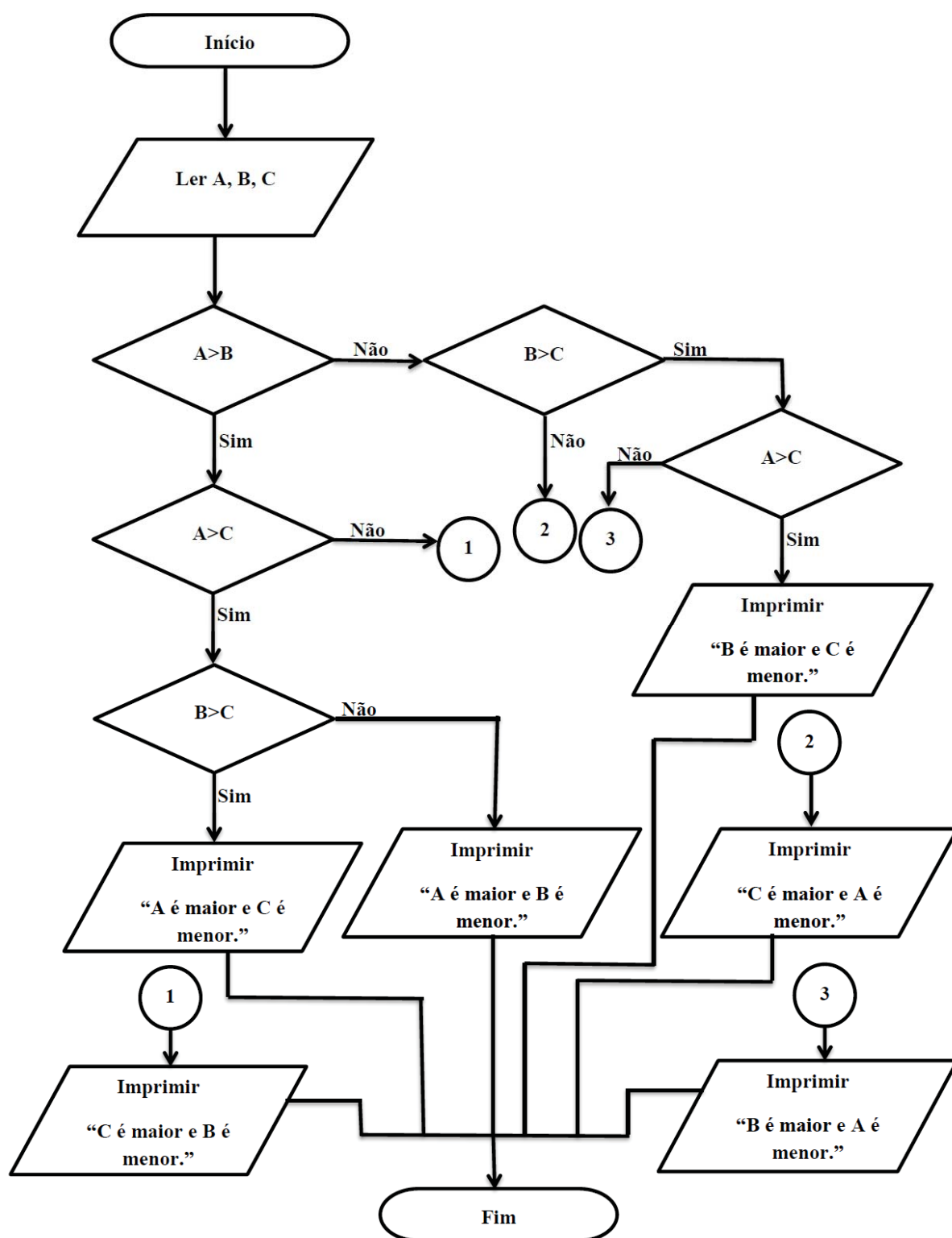


Figura 1 – Fluxograma (Fonte: autor)

Apesar de o exemplo ser bem simples, ele dá uma noção de como se montam os diagramas de fluxo e, como dito anteriormente, pode-se perceber que ele facilita quando da realização da programação, pois o diagrama explicita a sequência que é preciso ser seguida na programação.

As grandes vantagens da utilização do diagrama é que ele facilita a leitura e o entendimento do que deve ser feito, dá uma visão melhor da localização e da identificação dos aspectos que são mais importantes, possibilita uma análise mais adequada e é muito mais fácil descrever os métodos empregados. Entretanto a decisão do nível de detalhes a ser colocado nos fluxogramas fica a critério de quem o está montando.

2.2 Programa

Um programa inclui três blocos fundamentais: o das entradas, o das saídas, e do processamento, sendo que esse último contém o algoritmo propriamente dito, como ilustrado na figura 2 abaixo.

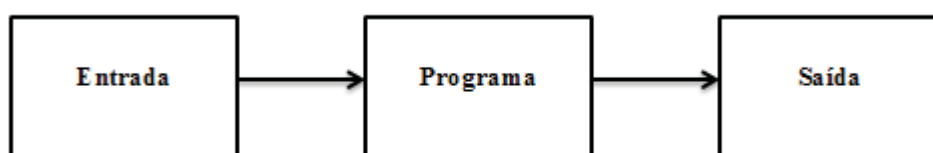


Figura 2 -Bloco do Sistema (Fonte: autor)

Para garantir o êxito no desenvolvimento de um programa, é necessário analisar o problema. Depois de analisadas as especificações de entrada e saída, o algoritmo pode ser montado. Com essas atividades realizadas, é possível assim começar a codificar o programa e, quando essa parte for finalizada, teremos como realizar testes e obter resultados, os quais poderão ser analisados.

A programação segue a estrutura dos algoritmos. A única coisa que muda é o modo de como é escrito. Os elementos básicos são:

- Constantes;
- Variáveis;
- Expressões;
- Instruções;
- Caracteres especiais (como ponte e vírgula que indica o fim da linha.);

- Indicadores (define o nome do programa, da função, etc.);
- Palavras reservadas (com o *if-else*, *do-while*, etc.).

Além dos elementos mencionados anteriormente, existem alguns que são muito utilizados na hora de realizar o programa, como os laços, os contadores e as estruturas sequenciais.

O programa é constituído de expressões que podem ser manipuladas, isso só depende do modo como a operação é estruturada. As expressões são classificadas em aritmética, relacionais, lógicas e caractere.

O resultado das expressões lógicas e relacionais é do tipo lógico; o das aritméticas é do tipo número, e o de caractere é do tipo caractere.

Quando trabalhamos com as expressões aritméticas é necessário seguir algumas regras de ordem de prioridade das operações que são praticamente iguais às utilizadas em matemática. A precedência é:

- Exponenciais;
- Multiplicação e divisão;
- Divisão inteira (retorna o quociente inteiro de uma divisão) e módulo;
- Soma e subtração.

2.3 Interface Gráfica

Interface gráfica é o modo que o usuário e o programa se comunicam através de uma representação visual, que pode ser por meio de gráficos, desenhos, imagens, etc. Ela faz com que o usuário consiga se integrar com a aplicação de modo mais eficaz, pois ela tenta fazer com que o indivíduo consiga compreender rapidamente o programa, quase intuitivamente, e assim utilizá-la de modo mais produtivo.

A interface é praticamente intuitiva, pois os ícones empregados indicam e representam a operação que será efetuada; devido a isso, não fica tão massacrante aprender a utilizar o aplicativo e faz com que o usuário queira aprender mais sobre o funcionamento do mesmo. O

indivíduo interage com a interface por intermédio do teclado ou do mouse e, deste modo, é possível obter resultados práticos.

Exemplos de interfaces gráficas muito usadas por muitas pessoas sem se darem conta são as páginas de internet, os aplicativos do Office em geral, o próprio desktop dos computadores, etc. A figura 3 nos exemplifica uma interface, onde pode ser notado que cada componente é autoexplicativo, o que facilita muito a vida do usuário e torna o sistema cada vez mais otimizado.

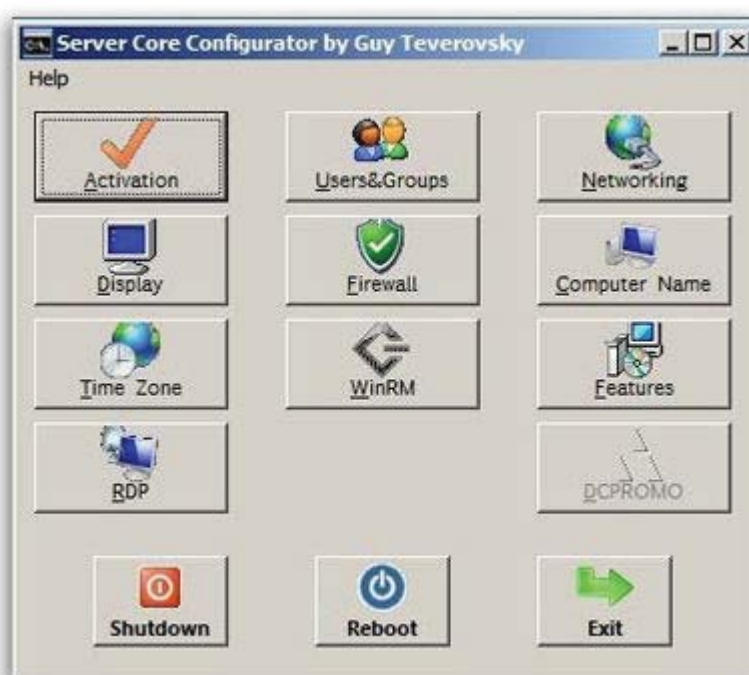


Figura 3 - Exemplo de Interface (Fonte: autor)

2.4 Linguagem de Programação C#

O C# também conhecido como C Sharp é uma linguagem de programação orientada a objetos. É uma derivação do C++, mas não possui as mesmas limitações deste, da qual utilizou a idéia da orientação a objetos, mas também possui grande influência das outras linguagens. O C# foi criado pela Microsoft para fazer parte da plataforma .NET.

Essa linguagem é bem simples de se implementar e fácil de se aprender. O C# fornece recursos poderosos, um deles é o acesso direto à memória, que também são recursos que não são encontrados no Java e no C++. As pessoas que possuem algum conhecimento de C, C++ ou de Java terão facilidade em realizar implementações com o C#, pois é muito parecido com essas outras linguagens.

Hejlsberg et al.(2010, p. 1) falam que o C Sharp possui construções de linguagem que apoiam diretamente os conceitos de fornecer os componentes de *software* em forma de pacotes autossuficientes e a autodescrição da funcionalidade, o que a torna muito natural para usar e criar componentes do *software* e também possui várias características que ajudam na construção de aplicativos robustos e duráveis.

Segundo Hejlsberget al. (2010, p. 1)

Para garantir que os programas de C# e suas bibliotecas possam evoluir ao longo do tempo de forma compatível, foi colocado muita ênfase no desenvolvimento da versão de C#. Muitas linguagens de programação dão pouca atenção a essa questão. Como resultado, os programas escritos nessas linguagens quebram mais vezes quando novas versões de bibliotecas são introduzidas.[...]

Por ser uma linguagem orientada a objetos, ela pode possuir em seus programas o encapsulamento, a herança e o polimorfismo. Todos os métodos e variáveis são encapsuladas em definições de classes.

A interface gráfica no C# é bem mais fácil que no Java, pois nela os componentes a serem usados já possuem seus ícones prontos e só basta carregá-los na implementação que está sendo desenvolvida; neste momento o usuário arruma o *layout* de modo que o agrada e depois só precisa programar o modo em que os ícones irão atuar; já em Java os componentes e o *layout* da interface gráfica devem ser programados assim como o modo em que eles irão atuar. A linguagem que está sendo estudada consegue diferenciar letras maiúsculas de minúsculas.

Segundo Nagel, citado por Hejlsberget al. (2010, p. 2), “C# não é uma linguagem pura orientada a objetos, mas sim uma linguagem que se estende ao longo do tempo para obter mais produtividade nas principais áreas que o C# é usado.[...]”

2.4.1 Estrutura da linguagem C#

O C# é constituído de uma estrutura que possui uma classe externa, o nome do projeto, o nome da classe e o procedimento principal.

Segundo Santos (2007) o que cada componente da estrutura faz é:

1. **Classes externas:** são as classes básicas que são necessárias para que o programa funcione corretamente, e essas já são inseridas na criação do projeto. Na linguagem usada cada instrução do programa possui a sua definição para o compilador que é adquirido por meio das classes básicas.

2. **Nome do projeto:** quando criamos um novo projeto devemos escolher um nome para este, e também será criada uma pasta que conterà todos os arquivos e subdiretórios da aplicação em questão.
3. **Nome da classe:** especifica o nome da classe que está sendo criada e este nome pode ser alterado a qualquer momento, podemos dizer que é o cabeçalho.
4. **Procedimento principal:** é aqui que se localiza o programa em si; é neste ponto que o programador irá inserir o código.

No exemplo a seguir serão indicados os componentes da estrutura.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
    }
namespace Exemplo01
{
    class Program
    {
        static void Main(string[] args)
        {
            4
        }
    }
}

```

2.5 Microsoft Visual Studio

É um pacote de programas da Microsoft desenvolvido para elaboração de *softwares* e no mercado podem ser encontradas diversas edições.

O Visual Studio é usado para desenvolver programas tradicionais e também aplicações que se utilizam de interface gráfica. Um dos componentes deste pacote é o Windows Forms

que é utilizado para os casos com utilização de interfaces gráficas. Este componente será estudado mais à frente.

A versão utilizada será a 2010, a qual contém os seguintes componentes: Visual Basic .NET, Visual Basic, Visual C#, Visual C++, Visual J#, entre outros. Esta versão pode trabalhar no Windows 7 o que facilita pois os computadores mais novos já vêm configurados com essa versão de Windows, além desse Visual Studio possuir melhorias em seus pacotes, se for comparado com as edições anteriores.

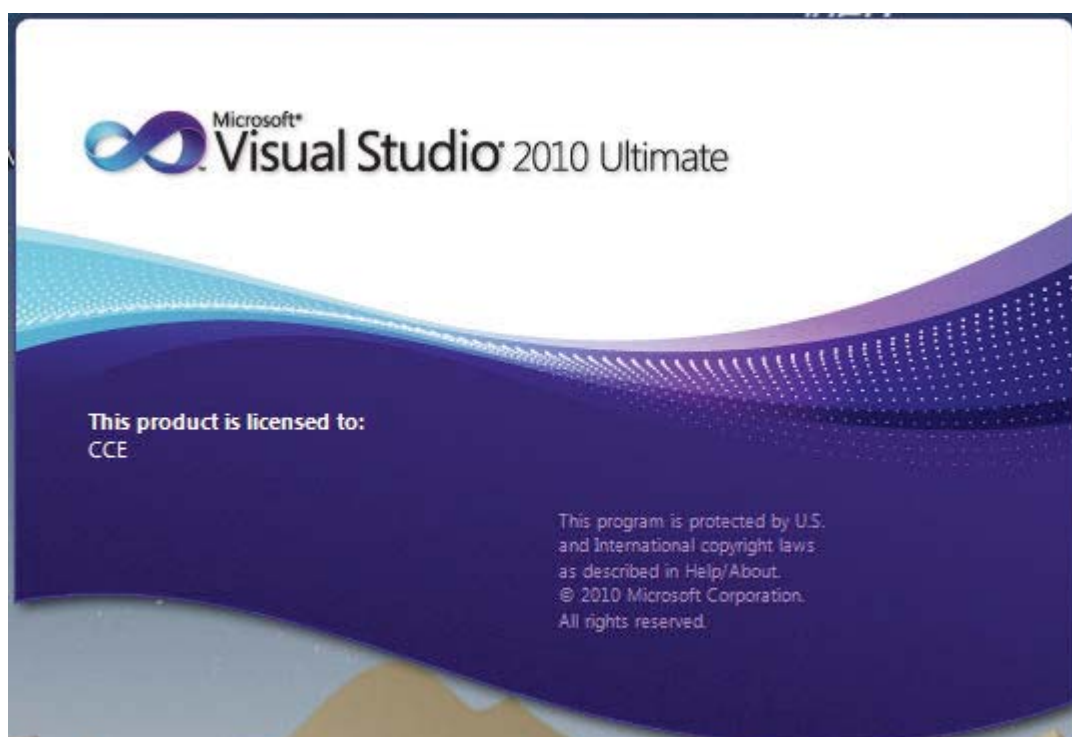


Figura 4 - Tela de Abertura do Microsoft Visual Studio (Fonte: autor)

O primeiro contato do usuário com o *software*, depois da tela de abertura que aparece enquanto o programa está se carregando, é mostrado na figura 5. O usuário deve se familiarizar com as ferramentas que estão localizadas na barra de ferramenta e como estão organizadas as coisas na página principal, para conseguir aproveitar de modo mais eficiente o *software* em questão.

No *layout* dessa janela, é possível visualizar os projetos que foram recentemente utilizados, os ícones de criação de novo projeto e abrir outros projetos, além de possuir dicas de como usar o *software* e muitas outras coisas.

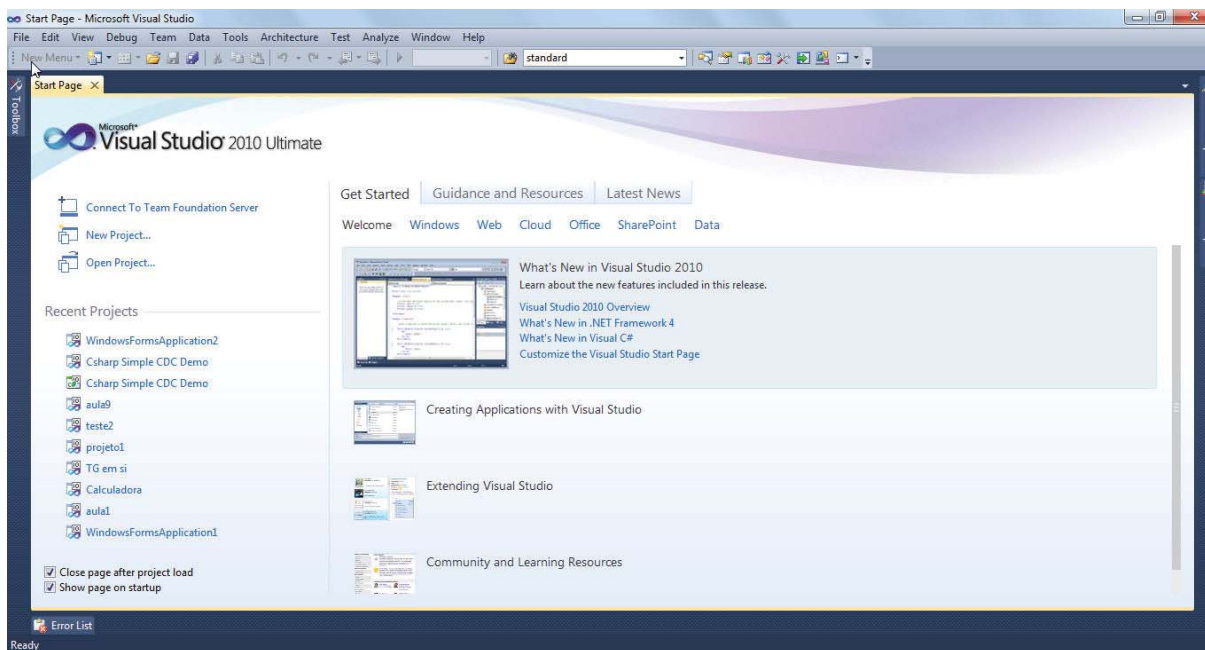


Figura 5 - Tela Inicial do Visual Studio (Fonte: autor)

2.5.1 Microsoft Visual C#

O Visual C# é uma implementação da linguagem C# produzida pela Microsoft. Possui um compilador, um editor de códigos completo, um depurador avançado e fácil de utilizar, e outras ferramentas. Pode-se acessar vários serviços do sistema operacional e de outras classes que aceleram o ciclo de desenvolvimento, através das bibliotecas da classe .NET Framework.

A figura 6 destaca a janela de geração de um novo projeto, onde há todos os tipos de projetos que podem ser realizados no Visual C# utilizando a biblioteca .NET Framework 4. Caso a biblioteca seja de versões anteriores, pode não possuir todos esses tipos de projetos.

A figura 6 também nos mostra o local onde escolhe-se o nome, o local onde será salvo e o nome de solução do novo projeto que está localizado na parte inferior do *layout*. Na parte superior esquerda, pode-se selecionar o tipo de linguagem do projeto e outras opções referentes ao projeto, e na parte superior direita temos uma breve descrição do tipo de projeto selecionado.

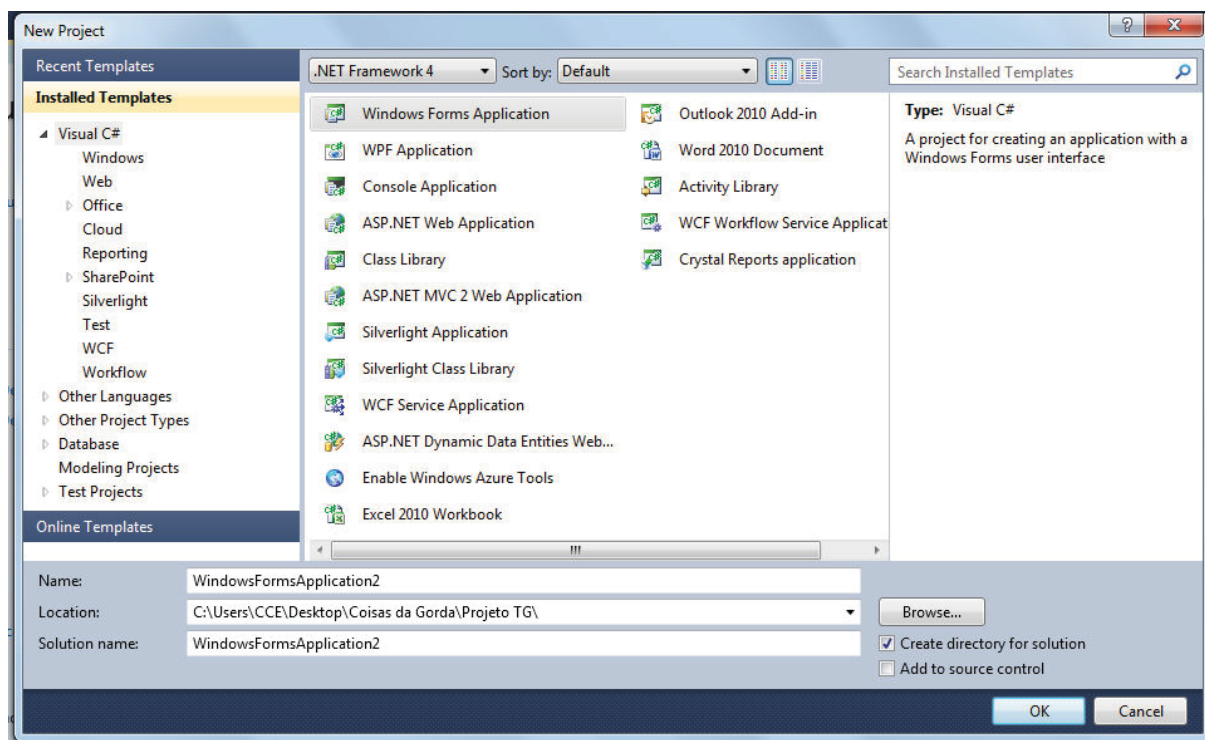


Figura 6 – Gerar Novo Projeto (Fonte: autor)

2.5.2 Windows FormsApplication

Essa aplicação que está contida no Visual C#, como pode ser visto na figura 7, é utilizada para implementar as interfaces gráficas, e a construção da interface é muito fácil de ser realizada.

A figura 7 mostra uma visão geral de como é a página inicial com que o usuário se depara quando seleciona essa aplicação para produzir o seu programa. Nesta figura há diversas janelas, cada uma com sua função e essas serão explicadas.

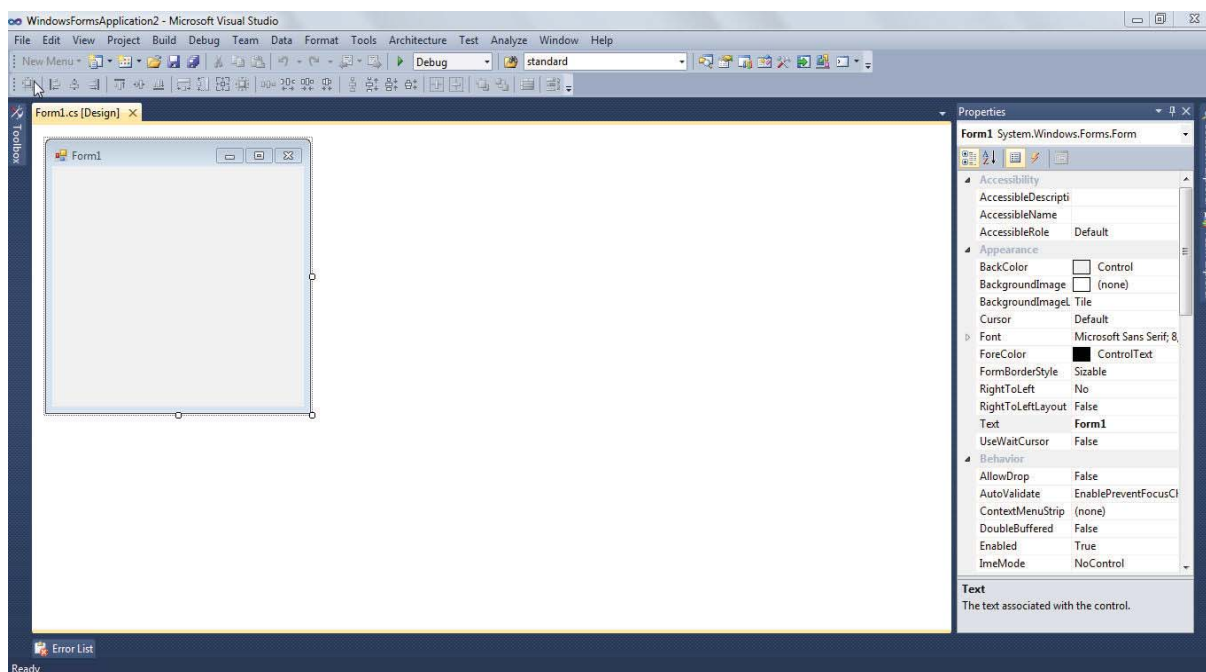


Figura 7 - Windows Form Application (Fonte: autor)

A figura 8 mostra onde é montada a interface gráfica. Os componentes que serão colocados nessa janela estão localizados na caixa de ferramentas (*Toolbox*), como mostra a figura 9. A figura 9a mostra como estão divididos os componentes, ao passo que a figura 9b mostra uma das subdivisões aberta para se ter uma ideia de como é o *layout* dos componentes dentro de cada pasta. Para colocar os componentes na janela que será montada na interface, basta selecionar o que se deseja colocar, puxar até a janela e arrumar cada componente do modo desejado. Depois de montada a parte gráfica, passa-se ao desenvolvimento da programação.

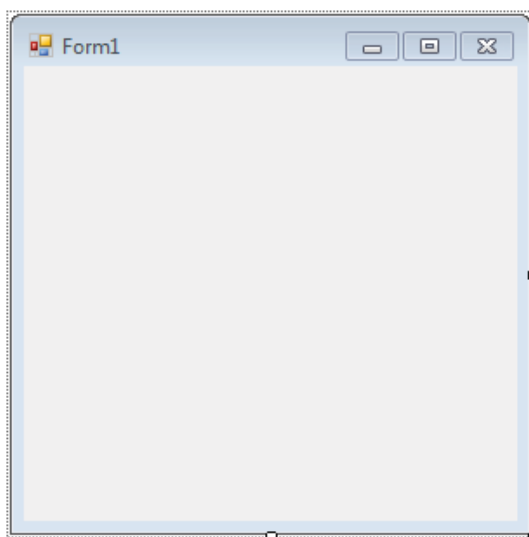


Figura 8 - Local da Montagem da Interface (Fonte: autor)

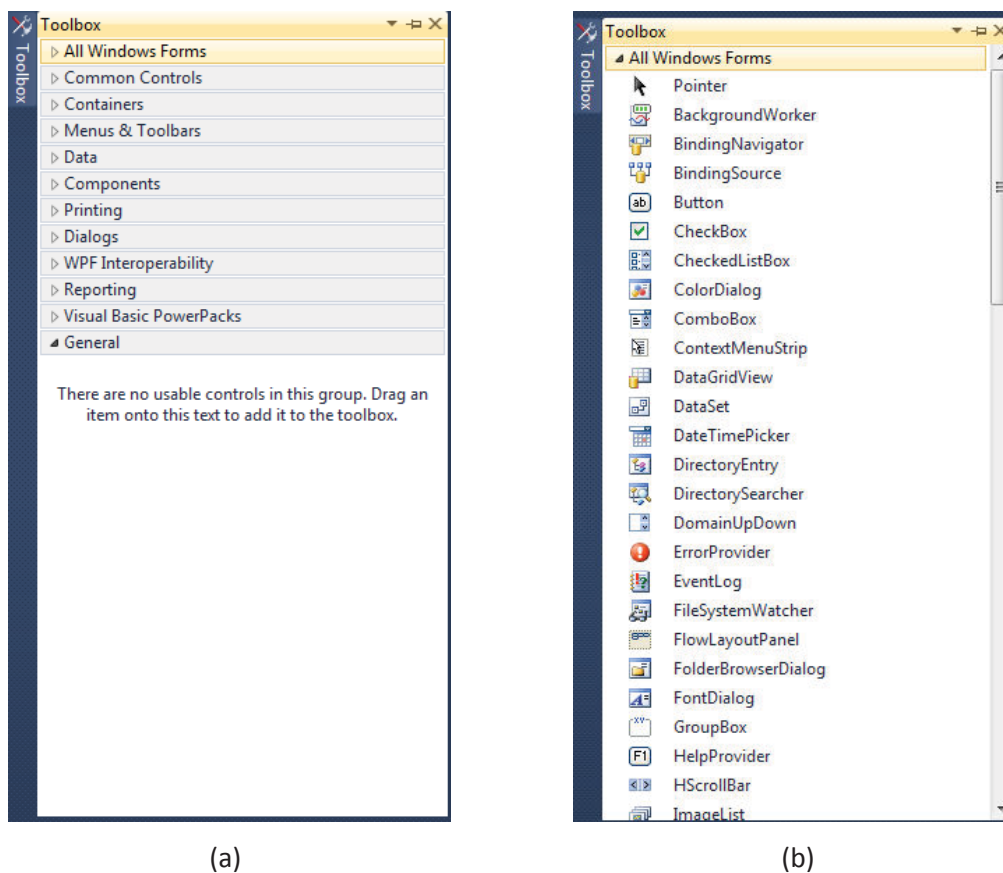


Figura 9 - Caixa de Ferramentas (Fonte: autor)

A figura 10 mostra a barra de propriedades da janela de interface, através da qual podem ser realizadas alterações em sua configuração. Caso a janela possua algum componente e esse seja selecionado, a janela de propriedades agora irá se referir a esse componente, portanto poderá ser ajustada a forma que se deseja.

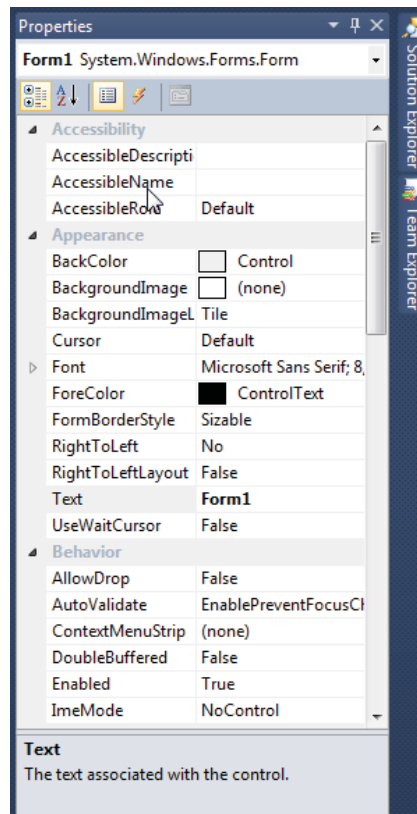


Figura 10 - Barra Propriedades (Fonte: autor)

3 METODOLOGIA

Neste capítulo é explicada a metodologia aplicada ao trabalho, desde a criação do programa até a sua finalização.

3.1 Atividades Preliminares

Antes de começar a desenvolver o programa, foi preciso estudar a linguagem C#, a qual é praticamente uma mistura da linguagem C++ e Java. Foi realizado também o reconhecimento do ambiente Visual Studio; porém antes de realizar o reconhecimento realizou-se a instalação.

Para realizar a instalação do Microsoft Visual Studio no computador em que o projeto foi desenvolvido, precisou-se verificar se a configuração do computador era compatível com as configurações mínimas essenciais para rodar o *software*. Uma das especificações é ter um sistema operacional Windows a partir do XP com o pacote de serviço 3 (SP3 – Service Pack 3) e outras configurações de *hardware* que são requisitadas.

Segundo o *site* da Microsoft (<http://www.microsoft.com>), as configurações de *hardware* requisitadas são:

- Processador a partir de 1,6 GHz;
- 1024 MB de memória RAM (*Random Access Memory*), caso executado em uma máquina virtual, 1,5 GB;
- 5,5 GB de espaço disponível no disco rígido (HD – *hard drive* ou *hard disk*);
- HD com velocidade 5400 rpm;
- Placa de vídeo DirectX 9 trabalhando com resolução a partir de 1024 x 768.

Depois de tudo instalado, do *software* funcionando e de ser feito um reconhecimento geral do ambiente Visual Studio, foi necessário realizar um reconhecimento mais detalhado no modo Windows Form do C#, pois foi nesta plataforma que o programa foi desenvolvido e estruturado.

Para construir um projeto adequado ao objetivo solicitado, precisou-se realizar algumas pesquisas em *sites* que realizam projetos similares; com isso pode-se ter alguma ideias como

seria efetuado o tratamento dos dados de entrada para obter resultados adequados e de fácil entendimento pelo usuário.

A coleta de dados com o aparelho eletrônico é realizada no momento que está sendo feita a ordenha, onde o usuário vê o número da vaca ordenhada por um brinco que cada uma possui, neste brinco há o código do animal em questão. A ordenha é feita por dia e todos os dias é realizada a ordenha nos animais, porém a coleta de dados utilizando o aparelho eletrônico só é feito a cada 15 dias ou uma vez por mês.



Figura 11 -Brinco de Identificação do animal (Fonte: <http://www.abhb.com.br> acessado em 17/12/2012)



Figura 12 - Animal com o brinco (Fonte: <http://www.tecnologiaetreinamento.com.br> acessado em 17/12/2012)

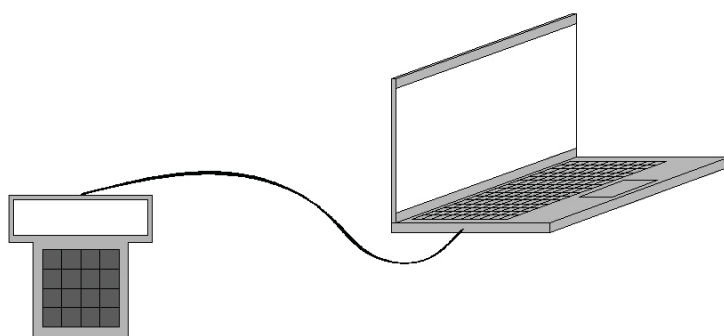


Figura 13 - Conexão Aparelho e Computador (Fonte: autor)

3.2 Desenvolvimento do projeto

Depois de todo o estudo realizado sobre o *software* e a linguagem empregados no desenvolvimento do projeto, passou-se a desenvolvimento de fato.

Segundo Santos (2010, p. 261)

“Desenvolver aplicativos em C# com classes é permitir ao desenvolvedor organizar o código, classificando as classes pela ordem de prioridade e definindo como suas informações podem ser utilizadas ou reaproveitadas em outras aplicações, quando desenvolvidas.”

Para poder explicar melhor o desenvolvimento do programa, o tópico foi dividido, para facilitar o entendimento de como o programa está estruturado.

3.2.1 Pré-projeto

Antes de começar a programar, foi montado um esquema do que deveria haver no programa e quais os passos que deveriam ser realizados no momento da programação. Pois realizando essas análises antes de começar a programar fica mais fácil de visualizar o que deve ser realizado na programação e como provavelmente o programa irá ficar.

Os esquemas são:

→ Relatório

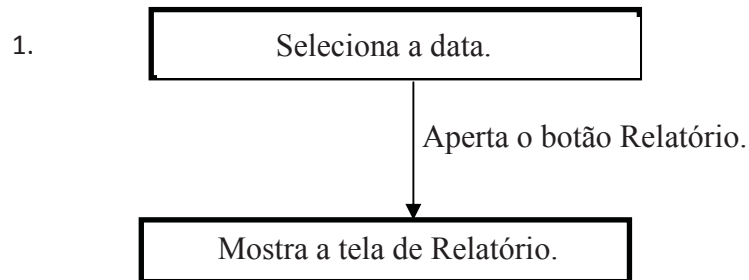


Figura 14 - Esquema Relatório (Fonte: autor)

→ Histórico do Rebanho

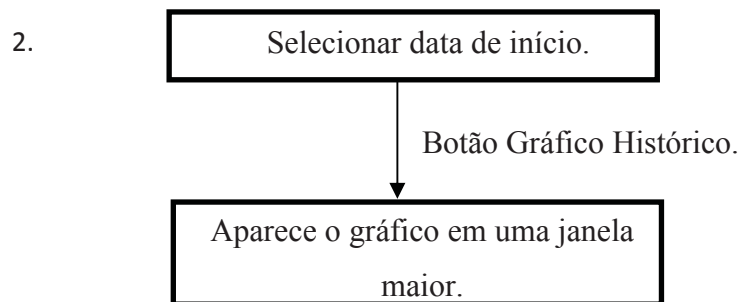


Figura 15 - Esquema Histórico do Rebanho (Fonte: autor)

→ Relatório da Vaca

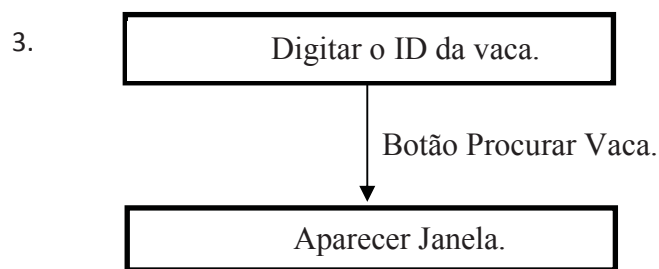


Figura 16 - Esquema Relatório Vaca (Fonte: autor)

Para realizar qualquer uma das operações anteriores, deve-se seguir alguns passos, para deste modo conseguir gerar os relatórios desejados. Os passos de cada uma das etapas são:

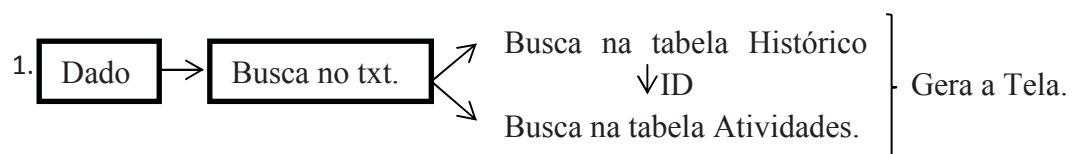


Figura 17 - Passos para Gerar Relatório (Fonte: autor)

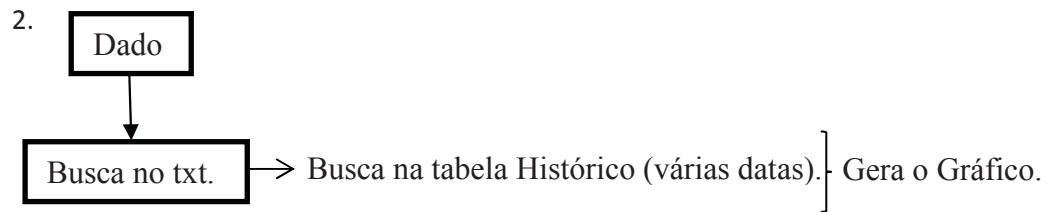


Figura 18 - Passos para Gerar Histórico (Fonte: autor)

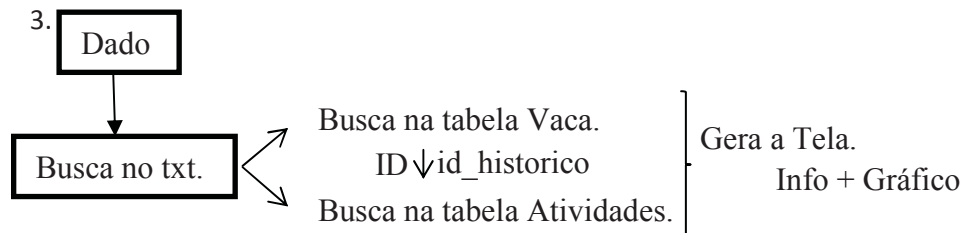


Figura 19 -Passos para Gerar Relatório Vaca (Fonte: autor)

Entretanto para conseguir realizar essas operações é necessário acessar os dados que serão recebidos pela porta USB e realizar os seguintes passos:

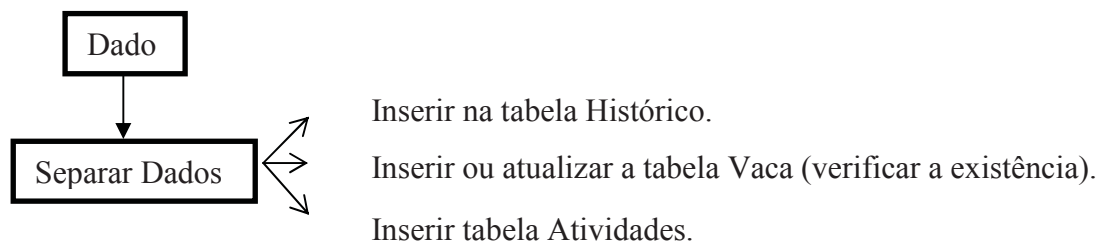


Figura 20 - Passos para Salvar os Dados (Fonte: autor)

Os txt são três arquivos textos e em cada arquivo está salvo uma tabela, as tabelas que estão e são salvas são as tabelas Vaca, Histórico e Atividade.

3.2.2 Desenvolvimento do Programa

Depois de determinado o que precisaria ser desenvolvido para conseguir o tratamento adequado dos resultados, foi possível começar a desenvolver o programa com mais eficiência, pois já se tem ideia das classes necessárias e da sequência que deve haver em cada uma delas.

3.2.2.1 Biblioteca

Antes de se iniciar o programa, as bibliotecas básicas são determinadas, para que não ocorra nenhum erro de compilação. As bibliotecas mais específicas são adicionadas no decorrer da programação para que a utilização dos elementos que necessitam dessas possam ser utilizados sem a ocorrência de falhas. Com isso quando for realizada a compilação não haverá erros relacionados às bibliotecas, mas ainda poderão existir erros na estrutura do programa, na lógica, na declaração dos parâmetros e possíveis erros de digitação.

As bibliotecas básicas são inseridas quando o projeto é criado. O próprio programa as insere, deste modo o programador não precisa ter a preocupação de colocá-las. São elas: System, System.Collection.Generic, System.ComponentModel, System.Data, System.Drawing, System.Linq, System.Text e System.Windows.Forms.

Segundo o *site* da Microsoft (<http://msdn.microsoft.com>), o que cada biblioteca possui, fornece ou realiza é:

- System – essa biblioteca contém classes fundamentais e classes base que definem os tipos de dados de referência e valor comumente usados, eventos e manipuladores eventos, interfaces, atributos e execuções de processamento.
- System.Collection.Generic – essa biblioteca contém interfaces que definem coleções genéricas, que permitem que os usuários criem coleções fortemente tipadas (a declaração do tipo é obrigatória) que fornece maior segurança do tipo e desempenho do que coleções fortemente tipadas-não-genéricas.
- System.ComponentModel – essa biblioteca contém os tipos que implementam o comportamento de componentes e controles em tempo de execução e em tempo de projeto.
- System.Data – essa biblioteca fornece acesso às classes que apresentam a arquitetura do ADO.NET. O ADO.NET permite a construção de componentes que são eficientes no gerenciamento de dados de várias fontes de dados.
- System.Drawing – essa biblioteca fornece acesso a funcionalidade de gráficos GraphicsDevice Interface Plus(GDI+).

- System.Linq – essa biblioteca fornece as classes e interfaces que suportam consultas que utilizam *Language-Integrated Query* (LINQ). O LINQ é uma síntese unificada para realização de consultas em fontes de dados variados, e ela foi inspirada na Linguagem de Consulta Estruturada (SQL).
- System.Text – essa biblioteca representa a codificação 8-bit *Unicode Transformation Format* (UTF-8) de caracteres Unicode. A UTF-8 apresenta cada ponto de código que é uma sequência de um a quatro bytes.
- System.Windows.Forms – essa biblioteca define a classe básica para os controles, os quais são os componentes com representação visual.

A biblioteca System.IO.Ports foi utilizada para poder ser realizada a conexão com a porta USB e a System.IO foi utilizada para a realização do uso de arquivos texto, podendo ser criação, modificação ou execução.

3.2.2.2 Programação

Para a leitura dos dados transmitidos pela USB e para o tratamento desses dados alguns processos de programação devem ser realizados. Por trás desses processos as bibliotecas explicadas anteriormente estão envolvidas.

A parte da programação é muito importante, pois é nela que será colocada toda a parte lógica do programa e também todos os parâmetros necessários para que no final seja possível alcançar os resultados desejados.

Foi necessário criar funções para atualizar as portas *Component Object Model* (COM). A USB é identificada como se fosse uma dessas portas, em uma *listbox*, essa função é lançada periodicamente com base no atributo do intervalo, o qual é definido no editor de formulários na janela de propriedades. Em seguida foi construída uma função para abrir as portas, que atua quando o usuário clica no botão *Connect*. Mas, para que isso ocorra, deve estar selecionada a porta COM, com a qual a conexão deve ocorrer. Além de abrir a porta, esta função também vai alterar o atributo de Habilitação (*Enable*) de vários objetos de formulário para evitar que o usuário abra uma nova porta COM.

Foi gerada uma função para fechar a porta COM, que atua quando o usuário clica no botão Fechar, sendo que essa função só pode ser utilizada se o programa já estiver conectado

com alguma porta. A função também pode ser diretamente chamada através de outras funções. A função em questão, além de fechar a porta, altera o atributo de Habilitação (*Enable*) de vários dos objetos de formulário para permitir que o usuário abra uma nova porta COM.

Criou-se uma função para imprimir os dados recebidos pela porta COM, que é chamada quando os dados são recebidos através da porta COM. Esta função tenta escrever os dados na caixa de texto. Caso ocorra uma exceção, a função que fecha a porta COM é chamada, que fecha a porta sem o usuário precise clicar no botão Close.

Todas essas funções comentadas anteriormente foram criadas em cima da interface principal. Foi necessário criar três classes muito importantes, pois elas são a base das ações que ocorrem nas interfaces, a fim de realizar o tratamento dos dados. Essas três classes estão mostradas a seguir.

1. `publicclassVaca`

```
{
    publiclong id;
    publicdouble litros;
}
```

2. `publicclassAtividade`

```
{
    publiclongid_vaca;
    publicdouble litros;
    publicintnumOrdenhas;
    publiclongid_historico;
}
```

3. `publicclassHistorico`

```
{
    publiclong id;
    publicstring data;
    publiclongnumVacas;
    publicintnumOrdenhas;
    publicdouble litros;
}
```


As classes citadas e seus parâmetros são públicos, pois são utilizados em funções de outras classes. Também foram criadas três classes públicas que possuem funções públicas que são utilizadas nas interfaces. Estas funções não são criadas dentro das interfaces, para facilitar a visualização do que está acontecendo na interface e também para facilitar na hora de encontrar erros nas lógicas. Como cada função realiza uma ação, basta identificar qual ação está provocando erro e, assim, é possível focar a atenção em uma só função, o que ajuda muito a encontrar o erro com mais rapidez.

As classes e as suas funções são as seguintes:

1. `public class VacaDao`

```
{
    ...

    public Vaca buscarVaca(long id){...}
    public void inserirVaca(long id, double litros){...}
    public void atualizar(long id, double litros){...}
}
```

2. `public class AtividadeDao`

```
{
    ...

    public void inserirAtividade(string content, long id_historico){...}
    public ArrayList buscarAtividadePorHistorico(long id_historico){...}
    public ArrayList buscarAtividadePorVaca(long id_vaca, ArrayList hist){...}
}
```

3. `public class HistoricoDao`

```
{
    ...

    public long inserirHistorico(string content){...}
    public Historico buscarPorData(string data){...}
    public ArrayList buscarPor varias datas(string data){...}
}
```

Algumas das funções citadas são utilizadas nas interfaces que geram as telas com os resultados e as outras são utilizadas no momento de salvar em arquivo texto.

3.2.2.3 Interface

A primeira parte da interface foi montada antes da realização da programação, devido à facilidade e da programação ser montada baseada nela. Entretanto no decorrer da realização da codificação, a interface sofreu algumas modificações como acréscimos de alguns elementos, pois como cada componente na programação possui a sua função, a adição deles não precisava ser imediata, eles só precisariam existir caso as funções que já tenham sido ou está sendo codificadas utilizem-se do componente.

Para a montagem da interface só foi preciso escolher os componentes desejados na caixa de ferramentas (*toolbox*) e a organização deles na tela da montagem da interface é da forma que o programador deseja.

Os componentes que estão presentes na primeira caixa de interface são:

- Botões
 - Conectar: depois de selecionada a porta COM é só clicar nesse botão para ocorrer a conexão.
 - Fechar: esse botão permite fechar a porta que está conectada com o programa.
 - Salvar txt: permite separar os dados que estão na caixa de texto e salvá-los em três blocos de notas, sendo que cada elemento irá para o seu respectivo arquivo texto.
 - Buscar Vaca: abre outra interface que irá solicitar o número do animal que se deseja consultar.
 - Gerar Histórico: abre a interface que gerará o histórico.
 - Gerar Relatório: abre a interface que irá gerar o relatório
- *Listbox*: local onde estão dispostas as portas COM que podem ser acessadas.
- Caixa de texto (*textbox*) com múltiplas linhas: local onde serão recebidos pela porta COM do PC os dados sobre o rebanho.

O *layout* da interface é o que aparece na figura 21 que se encontra a seguir. Esta interface é a interface principal do programa e todas as outras interfaces surgem a partir desta, esta é a interface que o usuário possuirá o seu primeiro contato com o programa e será o que mais ele irá utilizar devido todas as outras surgirem a partir desta.

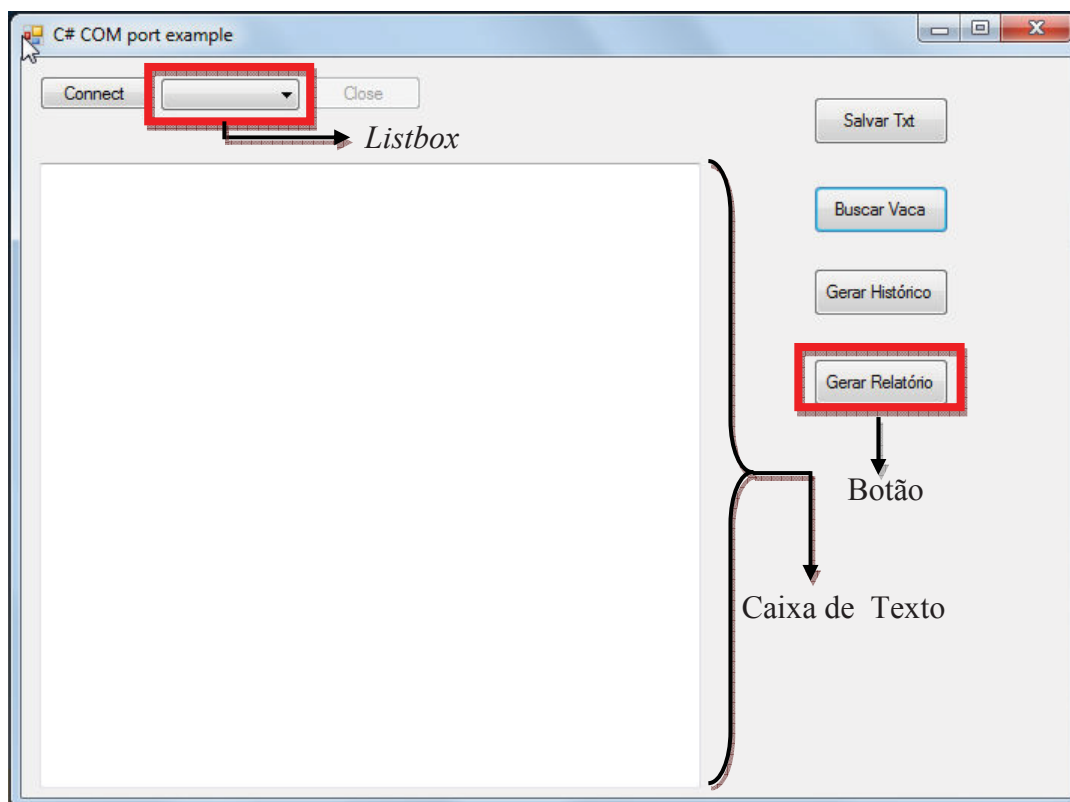


Figura 21 - Interface Principal do Programa (Fonte: autor)

Nesta interface também foi necessário colocar o componente da porta serial (*serialPort*) para que haja a conexão do programa com a porta USB, mas este não aparece no *layout* da interface.

A interface que abre quando se clica no botão *Buscar Vaca* é a mostrada na figura 22.

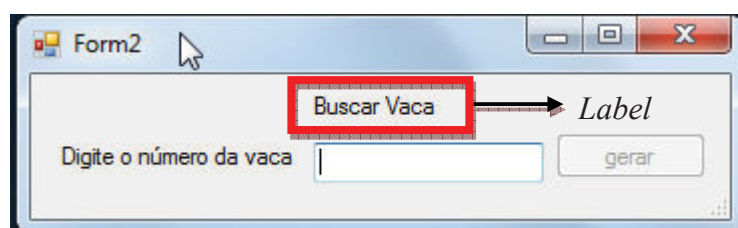


Figura 22 - Interface Verificação Existência da Vaca (Fonte: autor)

E nesta há os componentes a seguir:

- Dois *labels*
- Uma caixa de texto: o usuário digita o número da vaca que ele deseja pesquisar.

- Botão gerar: ele só será ativado se o usuário digitar alguma coisa na caixa de texto; caso o número digitado não exista, ele irá abrir uma caixa de diálogo e, caso o número digitado exista, irá abrir a interface que dará informações sobre a vaca.

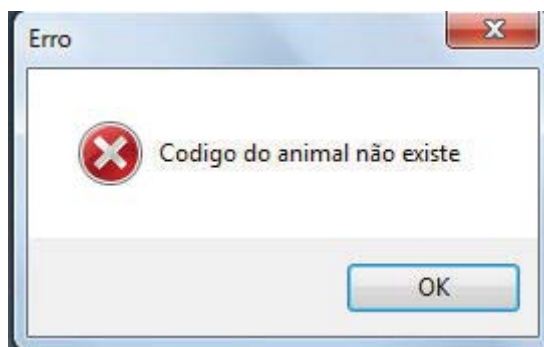


Figura 23 - Caixa de dialogo (Fonte: autor)

A figura 24 mostra a janela que abre caso o número da vaca digitado exista, e os componentes são:

- Cinco *labels*, sendo que dois desses *labels* são retirados do arquivo texto que possui o número da vaca e a quantidade de litros totais que a vaca em questão produziu.
- Caixa de texto onde será colocada a data de início para o gráfico que se deseja construir.
- Botão Ok: quando clicado e caso possua alguma data na caixa de texto que já exista gravado no arquivo texto ele irá gerar um gráfico a partir daquela data.
- Gráfico (Chart): esse espaço do gráfico é onde ele será gerado quando for digitada a data desejada para o início da produção da vaca.

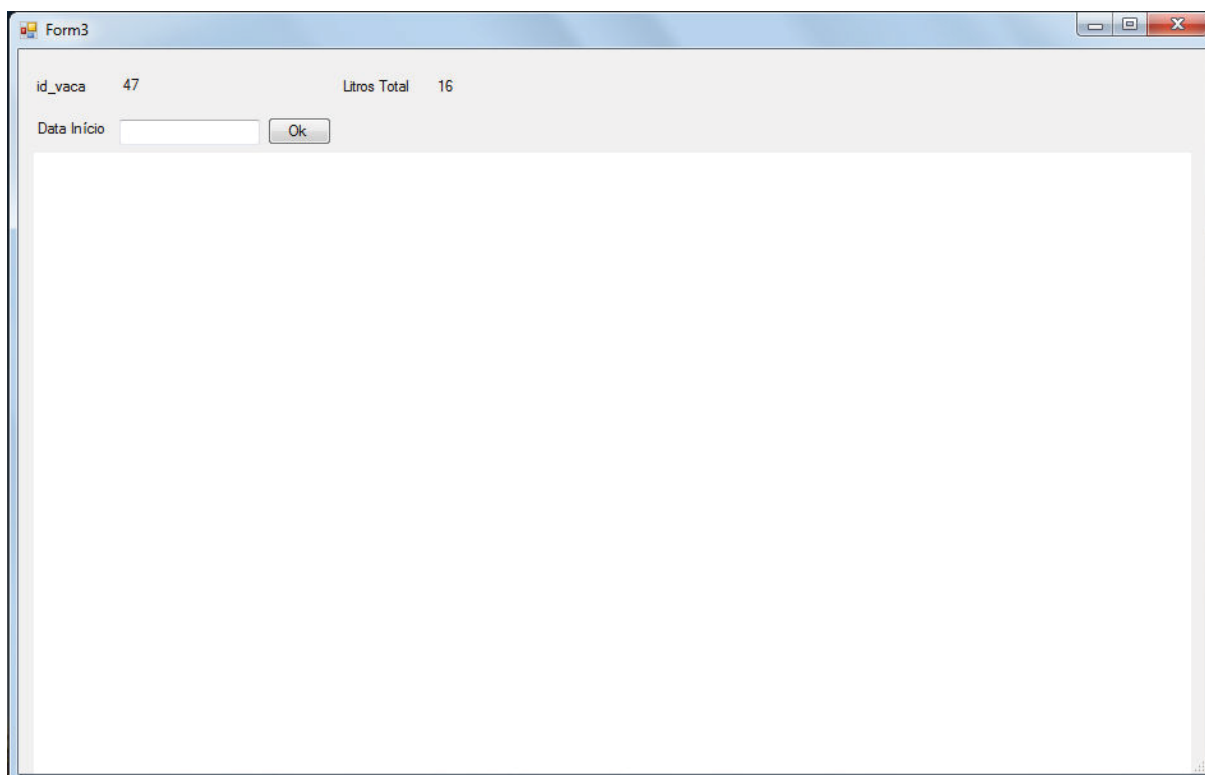


Figura 24 - Interface Relatório Vaca (Fonte: autor)

Quando o botão Gerar Histórico é acionado, a janela que aparece é mostrada na figura 25, com os seguintes elementos:

- *Label*;
- Caixa de texto onde será digitada a data a partir da qual se deseja gerar o gráfico;
- Gráfico;
- Botão Ok: quando clicado, e caso possua alguma data na caixa de texto que já exista gravado no arquivo texto, ele irá gerar um gráfico a partir daquela data.

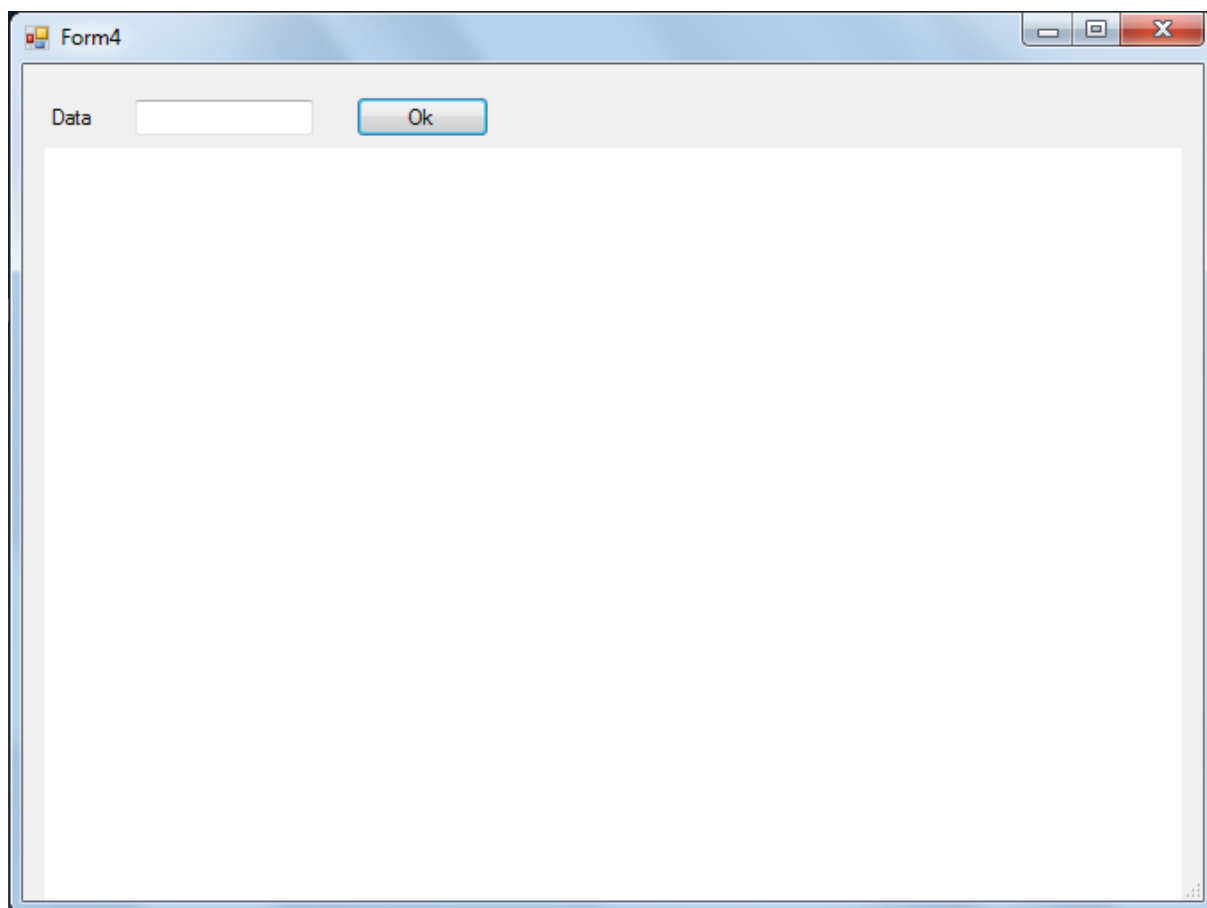


Figura 25 - Interface Histórico do Rebanho (Fonte: autor)

E se o botão Gerar Relatório for acionado, a interface que se abre possui os seguintes componentes:

- Dois *Labels*
 - Um dos *labels* se encontra sob a caixa de texto múltipla e será o destino das informações que estarão nessa mesma caixa de texto e esta ficará invisível e só será possível ver o *label*.
 - No outro *label* está gravada a palavra data.
- Caixa de texto
 - Simples: para inserir a data que se deseja para o relatório
 - Múltiplo: local em que serão gravadas as informações do relatório para depois enviar as informações para o *label* que fica abaixo dessa caixa de texto.
- Botão Ok: quando clicado e caso possua alguma data na caixa de texto que já exista gravado no arquivo texto ele irá gerar o relatório desta data.

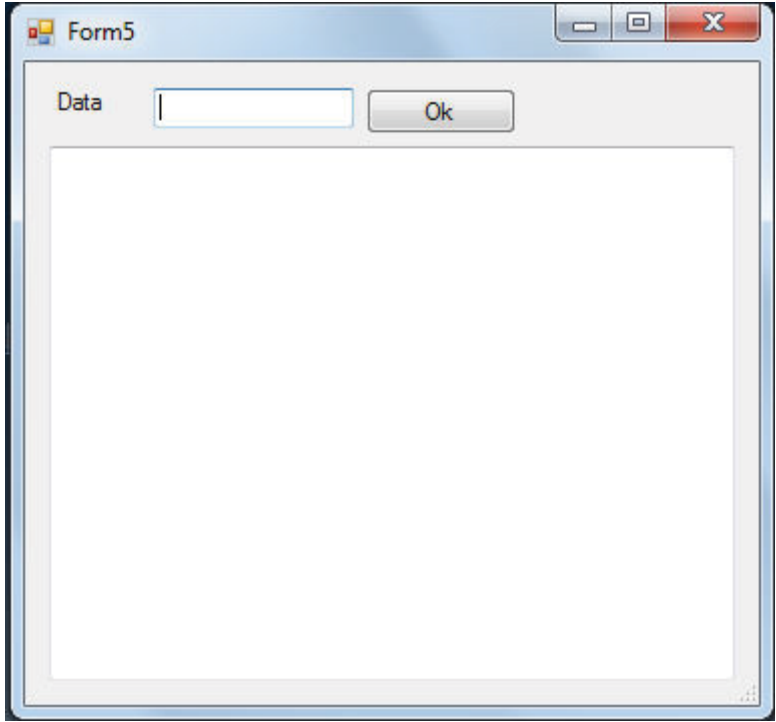


Figura 26 - Interface Relatório (Fonte: autor)

4 RESULTADOS E DISCUSSÃO

Neste capítulo serão mostrados alguns resultados obtidos, usando um conjunto de dados padronizados.

O aplicativo (projeto) gera três arquivos texto com tabelas distintas. Essas estão exemplificadas nas tabelas 1, 2 e 3:

Tabela 1 - Histórico

Id	Data	numVacas	numOrdenhas	Litros
1	190812	4	2	40
2	050912	4	2	42
3	200912	4	2	51
4	051012	4	2	51
5	201012	4	2	47
6	051112	4	2	45
7	201112	4	2	49,9

Tabela 2 - Atividade

id_vaca	Litros	numOrdenhas	id_hsitórico
47	16	2	1
11	14	2	1
523	12	2	1
1178	10	2	1
47	13	2	2
11	17	2	2
523	9	2	2
1178	15	2	2
47	18	2	3
11	15	2	3
523	12	2	3
1178	18	2	3
47	11	2	4
11	17	2	4
523	19	2	4
1178	16	2	4
47	11	2	5
11	12	2	5
523	9	2	5
1178	15	2	5
47	8	2	6
11	13	2	6
523	12	2	6

1178	12	2	6
47	11,8	2	7
11	11,2	2	7
523	17,7	2	7
1178	9,2	2	7

Tabela 3 - Vaca

ID	Litros
47	88,8
11	99,2
523	90,7
1178	95,2

Essas tabelas são alteradas conforme vão sendo feitas novas transferências de dados e salvas nos arquivos textos.

Para gerar as tabelas acima os dados são inseridos na interface conforme mostrado na figura 27 e devem ser salvos pelo usuário, ele deve clicar o botão Salvar txt. Será deste modo que os dados serão enviados pelo equipamento eletrônico através da porta USB para o aplicativo, e esses dados são referentes a um dia de coleta.

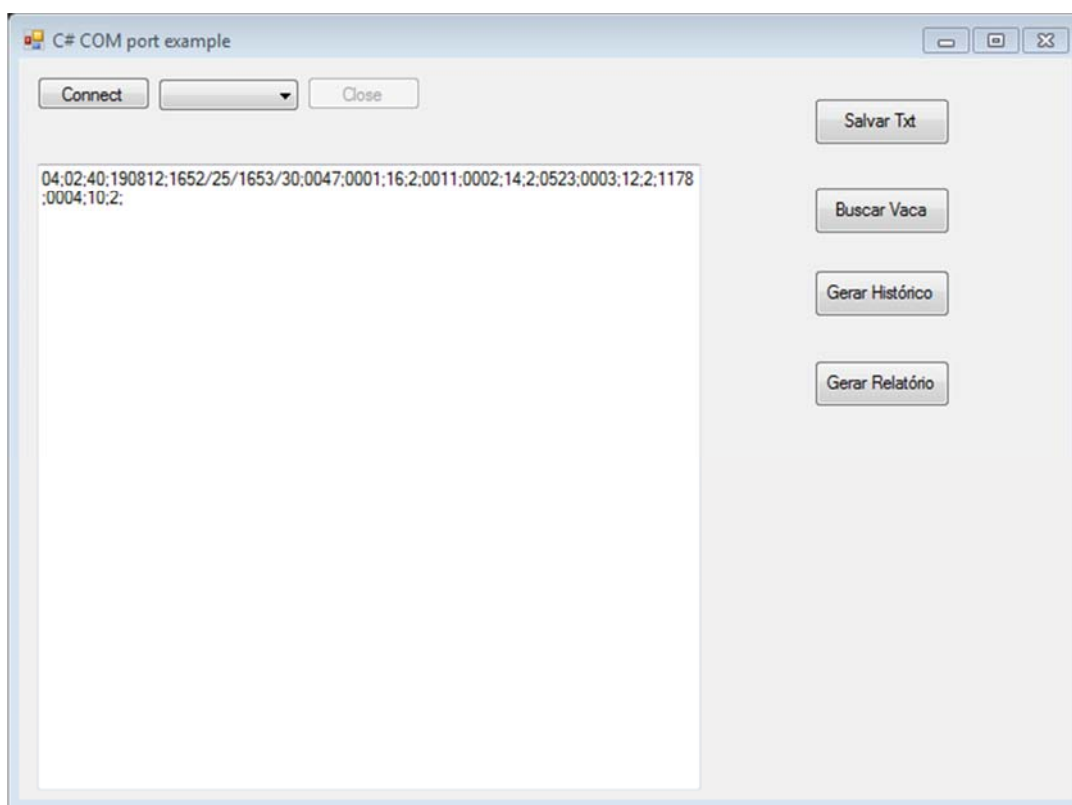
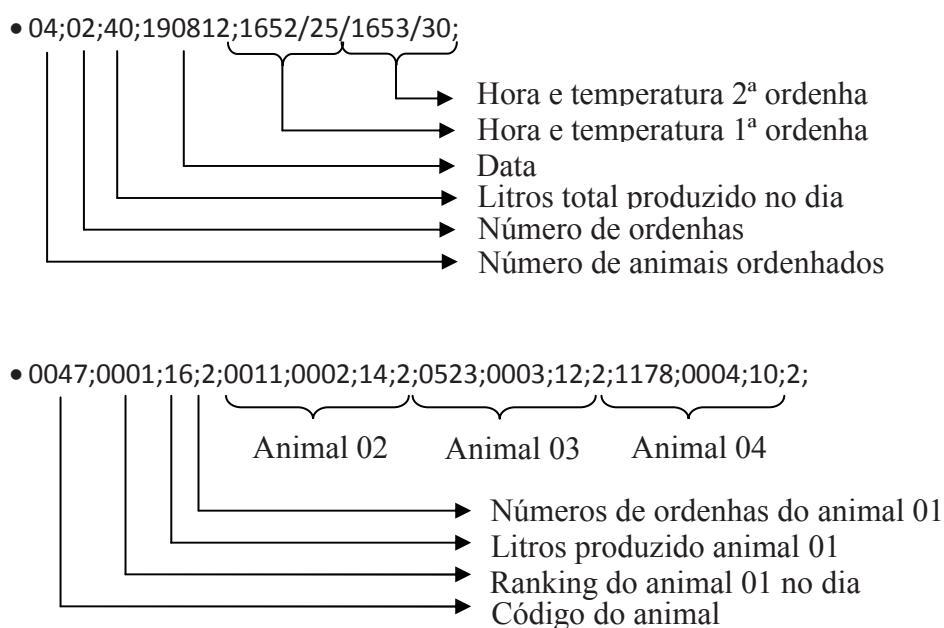


Figura 27 - Inserção de Dados (Fonte: autor)

É sabido que os dados foram transmitidos da seguinte maneira:

04;02;40;190812;1652/25/1653/30;0047;0001;16;2;0011;0002;14;2;0523;0003;12;2;1178;00
04;10;2;

Onde cada dado tem um significado:



Apesar de só terem 4 animais transmitidos, pode haver mais animais e os dados desses viriam posteriores ao do animal 04. Para gerar todos os dados das tabelas foram colocados na caixa de texto essas informações do modo que foi mostrado realizando alterações dos litros e da data. Foi a partir dessas simulações de dados que foi possível realizar as atualizações nos arquivos textos e testar o funcionamento do aplicativo desenvolvido.

Utilizando os dados das tabelas citadas e o programa desenvolvido, foi possível gerar os gráficos mostrados nas figuras 28, 29 e 30.

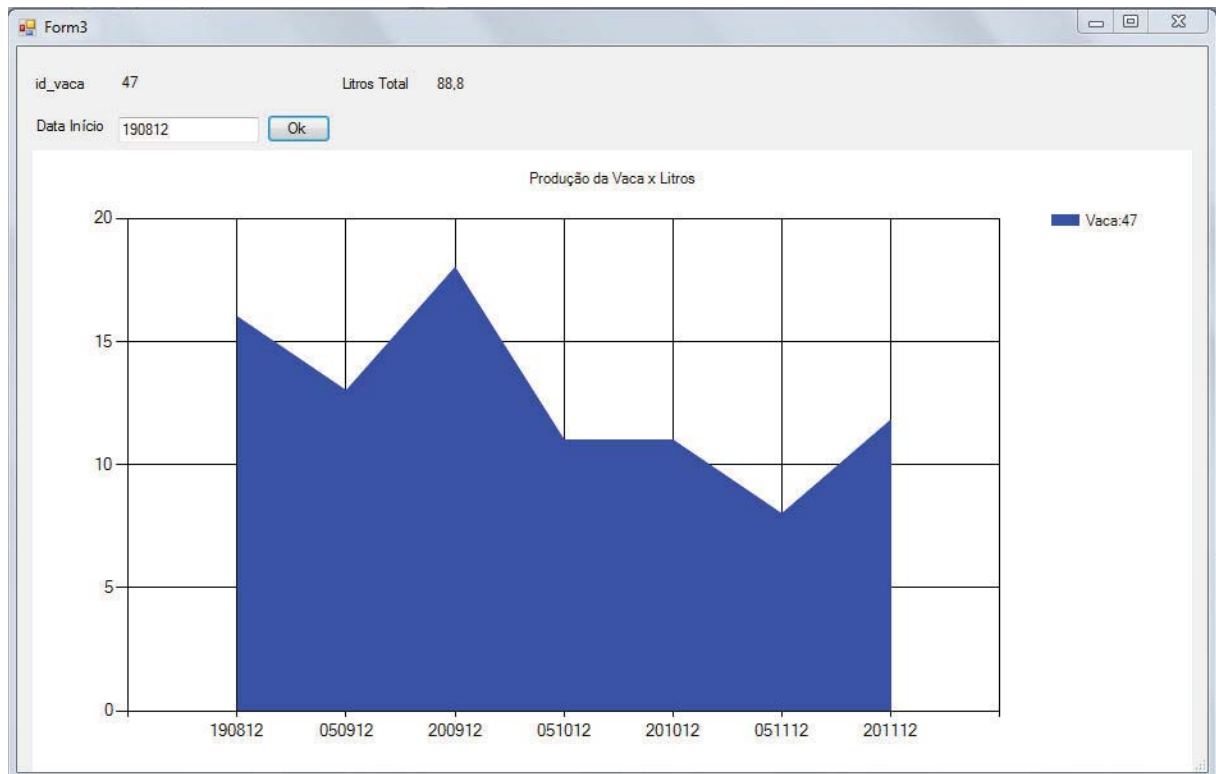


Figura 28 - Relatório da Vaca (Fonte: autor)

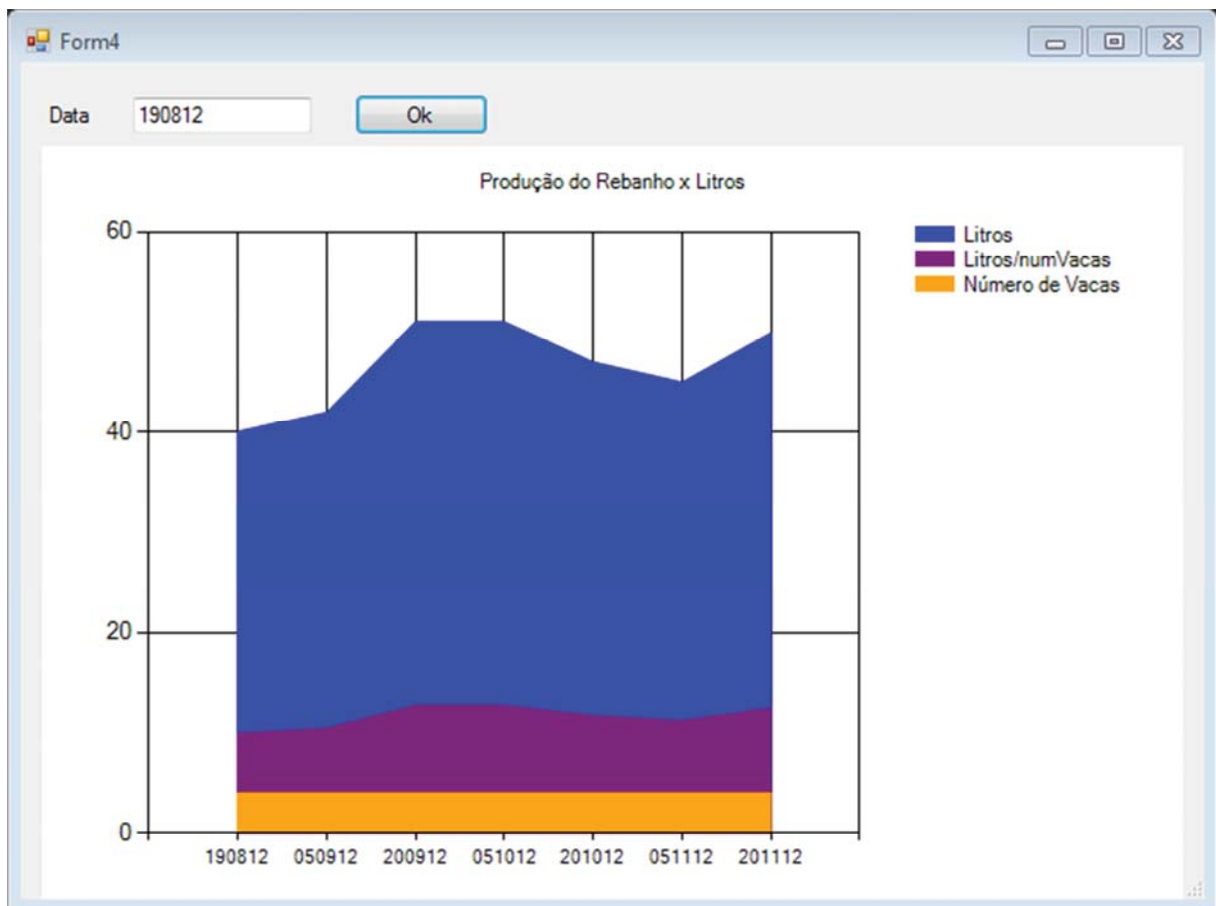


Figura 29 - Relatório do Rebanho (Fonte: autor)

Data: 190812	
Número de vacas: 4	
Número de Ordenhas: 2	
Produção total: 40	
47	16
11	14
523	12
1178	10

Figura 30 - Relatório (Fonte: autor)

A figura 28 está representando a produção da vaca 0047, onde os litros totais mostram o quanto a vaca produziu durante toda a sua vida e também gerou o gráfico da produção da vaca em litros pela data. O gráfico é gerado a partir da data escolhida pelo usuário e de mais quatorze datas consecutivas; caso tenha menos datas seguintes o gráfico será gerado até a última data existente.

O gráfico da figura 29 representa o histórico da produção total da fazenda, a quantidade de vacas e a produção média por vacas por data. O modo que o gráfico é gerado é semelhante ao do gráfico comentado anteriormente. Nesse gráfico o número de vacas possui uma relação retilínea, devido o número de vacas neste exemplo ser constante.

A figura 30 mostra o relatório que é gerado após o usuário escolher a data em que deseja ver as informações do relatório daquele dia. No relatório temos a data que foi selecionada, o número total de vacas que foram ordenhadas naquele dia, o número de ordenhas realizadas, a produção total de leite produzido no dia e também há o número de cada vaca e o quanto elas produziram.

Caso o usuário digite em qualquer uma das operações uma data que não exista no arquivo texto do histórico, não serão gerados os gráficos na interface da vaca e no do histórico, e na do relatório o mesmo não será gerado.

5 CONCLUSÃO

O programa proposto apresentou resultado eficiente, pois gerou uma interface de fácil uso e entendimento para o operador. A entrada de dados é feita via USB evitando retrabalhos, já que os dados são salvos em um equipamento eletrônico no qual ele consiga anotar em tempo real a produção de cada vaca evitando que necessite, no futuro, digitar esses dados para algum *software*.

Quando os dados recebidos são salvos, eles ficam armazenados em três arquivos texto, nos quais foram distribuídos de acordo com a função de cada um. Os três arquivos funcionam como um banco de dados, porém mais rústico. Caso fosse utilizado banco de dados o desenvolvimento do programa seria facilitado, entretanto iria dificultar a instalação, pois o usuário teria que instalar o programa de banco de dados e teria que possuir um computador que suportasse as configurações do mesmo.

A partir dos dados salvos no arquivo texto é possível gerar um histórico que posteriormente poderá ser utilizado para fazer estudos estatísticos. Esse estudo poderá influenciar nas decisões de melhoria de produção.

O estudo pode ser macro, quando há uma análise do rebanho, com a informação de quantas vacas há no rebanho e a produção geral. Ou o estudo específico de cada vaca possibilitando assim determinar quando, por exemplo, a vaca está mais produtiva, está no período fértil ou está na época de ir para o abate.

REFERÊNCIAS BIBLIOGRÁFICAS

- [1] Aguilar, L. J. **Fundamentos de Programação:** Algoritmos, Estruturas de Dados e Objetos. Tradução de Paulo Heraldo do Valle 3.ed. Madrid: McGraw-Hill, 2008.

- [2] Campos Filho, F. F. **Algoritmos Numéricos** 2.ed. Rio de Janeiro: LTC, 2007.

- [3] Dos Santos, L. C. **Microsoft Visual C# 2010 Express:** Aprenda a Programar na Prática. 1.ed. São Paulo: Editora Érica Ltda, 2011.

- [4] Hejlsberg, A. et al. **The C# Programming Language** 4.ed. Boston: Pearson Education, 2010.

BIBLIOGRAFIA CONSULTADA

[1] Albahari, B.; Drayton, P.; Merrill, B. **C# Essentials** 2.ed. Sebastopol: O'Reilly, 2002.

[2] Bonifácio Jr, J. M. **Borland C++ Bulder 3 para Principiantes** 1.ed. Rio de Janeiro: Axcel Books, 1998.

[3] ETEC Lauro Gomes. **Apostila C# - Módulo 1**

Disponível em:

<http://www.etelg.com.br/paginaete/downloads/informatica/apostila.pdf>> Data de Acesso: 23 de outubro de 2012.

[4] Griffiths, I.; Adams, M.; Liberty, J. **Programming C# 4.0** 6.ed. Sebastopol: O'Reilly, 2010.

[5] Murray III, W. H.; Pappas C. H. **Microsoft C/C++ 7.** 1.ed. McGraw-Hill, 1992.

[6] Tenenbaum, A. M. **Data Structures Using C.** 1.ed. Prentice Hall International Editions, 1990.