UNIVERSIDADE SÃO FRANCISCO

Engenharia de Computação

**MARCELO LUIS LEME**

# DESENVOLVIMENTO DE UM *SOFTWARE* PARA AUTOMATIZAR AS COMANDAS DE PEDIDOS EM RESTAURANTES

Itatiba

2010

**MARCELO LUIS LEME - 002200600239**

# DESENVOLVIMENTO DE UM *SOFTWARE* PARA AUTOMATIZAR AS COMANDAS DE PEDIDOS EM RESTAURANTES

Monografia apresentada ao curso de Engenharia de Computação da Universidade São Francisco, como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Ms. Percival Silva de Lucena

Itatiba

2010

Marcelo Luis Leme

Desenvolvimento de um *software* para automatizar as comandas de pedidos em restaurantes

Orientador: Prof. Ms. Percival Silva de Lucena

55 páginas

Palavras-chave: *SOFTWARE*, AUTOMAÇÃO, *HARDWARE*, COMANDA ELETRÔNICA, PEDIDOS.

**MARCELO LUIS LEME**

# DESENVOLVIMENTO DE UM *SOFTWARE* PARA AUTOMATIZAR AS COMANDAS DE PEDIDOS EM RESTAURANTES

Monografia aprovada pelo curso de Graduação em Engenharia de Computação da Universidade São Francisco, como requisito parcial para a obtenção do título de Engenheiro de Computação.

**Data de aprovação: ___/___/_____**

**Banca examinadora:**

_____

Prof. Ms. Percival Silva de Lucena (Orientador)
Universidade São Francisco

_____

Examinador um
Universidade São Francisco

_____

Examinador dois
Universidade São Francisco

*A Carolina, Isadora e Elisangela, as estrelas que dão luz a minha vida.*

# AGRADECIMENTOS

Aos meus amigos, que sempre estiveram ao meu lado para que eu não desistisse da idéia de me graduar em engenharia de computação.

A minha família, principalmente meu pai e minha mãe.

Ao apoio do meu orientador.

A Deus por me guiar sempre por bons caminhos.

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| *ASK* | *Amplitude-Shift Keying* |
| *BIT* | *Binary digiT* |
| *CSMA* | *Carrier Sense Multiple Access* |
| *DBF* | *dBase Files* |
| *DLL* | *Dynamic Link Library* |
| *FSK* | *Frequency-Shift Keying* |
| *INI* | *INI file, arquivo de inicialização e configuração* |
| *IP* | *Internet Protocol* |
| *IR* | *InfraRed* |
| *ISM* | *Industrial, Scientific e Medical* |
| *LAN* | *Local Área Network* |
| *PC* | *Personal Computer* |
| *PDA* | *Personal Digital Assistant* |
| *QPSK* | *Query Phase-Shift Keying* |
| *RF* | *Radio Frequency* |
| *RS-232* | *Recommended-Standard 232, Padrão da Associação das Industrias Eletrônicas para comunicação binária serial* |
| *TCP* | *Transmission Control Protocol* |
| *UML* | *Unified Modeling Language* |
| *USB* | *Universal Serial Bus* |
| *VT-100* | *Virtual Terminal-100* |

# LISTA DE FIGURAS

FIGURA 1 - DISPLAY DO TERMINAL SHELLBY 915 ..... 16
FIGURA 2- EXEMPLO DE TERMINAL SEM COMUNICAÇÃO ..... 17
FIGURA 3 - CICLOS DE DESENVOLVIMENTO ITERATIVO ..... 20
FIGURA 4 - BASE RF CONECTADA DIRETAMENTE A PORTA USB DO PC ..... 21
FIGURA 5 - BASE RF CONECTADA A UM CONVERSOR RS-232 TCP/IP ..... 21
FIGURA 6 - CASO DE USO DE CADASTRO DE PRODUTOS E CADASTRO DE USUÁRIO ..... 24
FIGURA 7 - CASO DE USO DE REGISTRAR COMANDAS ..... 24
FIGURA 8 - TESTE DA BASE RF E TERMINAL SHELLBY (CONFIGURAÇÃO) ..... 29
FIGURA 9 - TESTE DA BASE RF E TERMINAL SHELLBY (COMUNICAÇÃO) ..... 30
FIGURA 10 - FLUXOGRAMA DE LANÇAMENTO DE COMANDA ELETRÔNICA ..... 31
FIGURA 11 - IDENTIFICAÇÃO DO USUÁRIO DO TERMINAL ..... 32
FIGURA 12 - LANÇAMENTO DO NÚMERO DA COMANDA/MESA ..... 33
FIGURA 13 - LANÇAMENTO DE PRODUTOS NA COMANDA ELETRÔNICA ..... 34
FIGURA 14 - LOGIN DO SISTEMA DE RETAGUARDA ..... 34
FIGURA 15 - ABERTURA DE MESA OU COMANDA ..... 35
FIGURA 16 - FECHAR MESA OU COMANDA ..... 35
FIGURA 17 - FORMAS DE PAGAMENTO ..... 36
FIGURA 18 - MAPA DE MESAS E COMANDAS ..... 36
FIGURA 19 - LANÇAMENTOS REALIZADOS ..... 37
FIGURA 20 - O PROCESSO DE TESTE ..... 39

# LISTA DE TABELAS

TABELA 1 - COMPARATIVO ENTRE CUSTO E BENEFÍCIOS ..... 13
TABELA 2 - COMPARAÇÃO ENTRE AS TECNOLOGIAS DE *LAN* SEM FIO ..... 22
TABELA 3 - COMANDOS IMPLEMENTADOS DE CONTROLE ..... 23

LEME, MARCELO LUIS. **Desenvolvimento de um *software* para automatizar as comandas de pedidos em restaurantes.** 2010. Monografia (Graduação em Engenharia de Computação) – Universidade São Francisco, Itatiba.

# RESUMO

*Softwares* de diversas áreas tem passado por mudanças. Em específico, os desenvolvidos para automação comercial vem se tornando mais completos e complexos.
Este projeto é uma extensão do sistema TabRestaurante, versão 2.22, desenvolvido pela Microtab Sistemas & Computadores. Este sistema tem a finalidade de atender a demanda de estabelecimentos comercias do setor alimentício.
Este trabalho pretende desenvolver um sistema de *hardware* sem fio e de *software* embarcado que permitirá aos atendentes de estabelecimentos alimentícios coletar pedidos de refeições dos clientes in loco. O sistema de comanda eletrônica é capaz de captar os pedidos e transferi-los para o gerenciador do estabelecimento que informará ao setor responsável os dados sobre o que deve ser produzido e entregue aos clientes.
O *hardware* é o Terminal Shellby 915, desenvolvido pela empresa Wilbor Tech. A escolha por este *hardware* deve-se a tecnologia sem fio integrada.

**Palavras-chave:** *SOFTWARE*, AUTOMAÇÃO, *HARDWARE*, COMANDA ELETRÔNICA, PEDIDOS.

# *ABSTRACT*

Software from various areas have undergone some changes. In particular, those developed for business automation is becoming more extensive and complex.
This project is an extension of TabRestaurante system, version 2.22, developed by Microtab Sistemas & Computadores. This system aims to meet the demand of commercial establishments in the food industry.
This work aims to develop a system of wireless hardware and software specifications that enable food establishments attendants collect meal orders from customers on the spot. The system of electronic controls is capable of capturing the requests and transfer them to the manager of the establishment informing the responsible sector about what should be produced and delivered to customers.
The hardware used is the Terminal Shellby 915, developed by Wilbor Tech. The choice of this hardware is due to its integrated wireless technology.



**Key words**: SOFTWARE, AUTOMATION, HARDWARE, ELECTRONIC CONTROLS, APPLICATIONS.

# SUMÁRIO


1 INTRODUÇÃO ..... 7

2 OBJETIVO ..... 8

3 METODOLOGIA ..... 9

4 REVISÃO BIBLIOGRÁFICA ..... 10

4.1 Automação Comercial ..... 10
4.1.1 Objetivos e Benefícios da Automação Comercial ..... 11
4.2 Automação no ramo alimentício ..... 13
4.3 Análise de softwares ..... 14
4.4 Análise de *hardware* para terminal de pedido ..... 14
4.4.1 *Pocket PC*s ..... 14
4.4.2 Shellby ..... 15
4.5 A Linguagem de Modelagem Unificada (UML) ..... 18

5 DESENVOLVIMENTO ..... 19

5.1 Desenvolvimento do módulo de pedidos ..... 19
5.1.1 Modelo de Processo de desenvolvimento ..... 19
5.1.2 Protocolo de comunicação do Terminal com a Base RF ..... 20
5.1.3 Descrição dos casos de uso do módulo ..... 23
5.1.4 Linguagem de programação ..... 28
5.1.5 Armazenamento dos dados ..... 28
5.2 Programação e configuração do terminal de pedidos ..... 29
5.3 Etapas do desenvolvimento ..... 31
5.4 Visão Geral do sistema ..... 32

6 TESTES ..... 38

7 CONCLUSÃO ..... 40

REFERÊNCIAS BIBLIOGRÁFICAS ..... 41

APÊNDICES ..... 43

APÊNDICE A – *DLL* Shellby e *DLL* Wtechlpt Configuração do terminal Shellby ..... 44
APÊNDICE B – Figuras com os códigos fontes da rotinas do sistema em Delphi 7 ..... 52

# 1 INTRODUÇÃO

Nos últimos anos, a importância do *software* como ferramenta de automação comercial tem passado por mudanças significativas. As empresas necessitam cada vez mais de controle sobre suas operações, o que aumenta a complexidade e integração dos sistemas. O foco específico deste trabalho é prover uma solução de automatização para o processo dos pedidos feitos pelos clientes em *fast-foods*, restaurantes, padarias, locais onde clientes fazem pedidos para seu consumo.

O protótipo para o projeto é uma extensão do sistema TabRestaurante, versão 2.22, desenvolvido pela Microtab Sistemas & Computadores. Este sistema tem a finalidade de atender a demanda de estabelecimentos comercias do tipo restaurante e possui os módulos de cadastros de produtos, controle de entrada e saídas de mercadorias, consultas e relatórios diversos. Nesta versão os pedidos são digitados manualmente via um terminal computacional responsável por transferir os dados que constam em comandas de pedidos preenchidas manualmente por atendentes.

O projeto permite que atendentes de estabelecimentos alimentícios criem pedidos de clientes utilizando *hardware* apropriado. Os dados entrados pelos atendentes através de um terminal, são encaminhados automaticamente via conexão sem fio para o computador centralizador do sistema. Desta forma, é eliminada a necessidade de escrever um pedido manualmente e depois ter que digitar os mesmos dados novamente.

A solução de *hardware* é baseada no terminal Shellby 915, desenvolvido pela empresa Wilbor Tech. Cada base de transmissão permite a comunicação com até 16 terminais e pode ser conectada aos computadores do sistema através de entradas seriais, USB ou de rede.

# 2 OBJETIVO

Este projeto tem como objetivo automatizar o atendimento de pedidos feitos pelos clientes em restaurantes em geral, substituindo a comanda manual, utilizando o terminal sem fio Shellby 915. A automatização do processo engloba desenvolver e implantar um módulo de um sistema para a digitação e envio de pedido através do *hardware* escolhido.

# 3 METODOLOGIA

Este projeto do módulo do sistema foi dividido nas seguintes fases:

- Análise de *softwares*
    - TabRestaurante versão 2.22
    - *Cheff Solutions*
    - Comanda Ideal

- Análise de *hardwares* (terminais de pedidos)
    - Terminal Shellby 915
    - *Pocket PC*

- Comparativo entre as soluções

- Desenvolvimento do módulo de pedidos
    - Modelo de Processo de Desenvolvimento
    - Protocolo de Comunicação do Terminal com a Base RF
    - Descrição do caso de uso do módulo
    - Linguagem de Programação
    - Armazenamento de dados

- Programação e configuração do terminal de pedidos

- Etapas do desenvolvimento

- Visão geral do sistema

- Testes

- Conclusão

# 4 REVISÃO BIBLIOGRÁFICA

Este capítulo destina-se a uma revisão sobre os principais conceitos englobados neste trabalho.

## 4.1 Automação Comercial

Conforme Regensteiner [14], “A informática e a automação podem ajudar de forma importante na coleta e na organização dos dados necessários para a identificação dos fatores críticos, e oferecer informações que ajudem na gerencia do estabelecimento a tomar decisões nas diferentes situações. Um plano de automação comercial tem por objetivo aumentar a eficiência da empresa por meio da melhoria de processos.”

A automação é o conceito de tornar automáticas atividades repetitivas com uso de sistemas e equipamentos que efetuam coleta de dados e atuam nos processos, minimizando a necessidade de interferência humana.

Automatizar é obter um melhor gerenciamento operacional em todas as áreas da empresa, inclusive em seu relacionamento com parceiros comerciais e clientes. Inicia-se com a implantação de equipamentos e sistemas, substituindo processos e rotinas feitas manualmente por processos automáticos, até chegar ao uso de ferramentas que trazem mais controle na gestão da empresa, reduzindo custos, erros e alcançando maior competitividade e rentabilidade. [6]

Existem 3 passos importantes para uma automação eficiente:

Planejamento: Deve-se listar todas as atividades que acontecem dentro da empresa, desde a mais simples até a mais complexa. Através da lista resultante é possível identificar as necessidades de automação.

Preparação: Deve-se dimensionar o quanto será investido para o primeiro momento. É necessário escolher o fornecedor de soluções.

Implantação: As primeiras atividades a serem automatizadas dever ser as menos críticas. As áreas de pedidos, estoques, recebimento e deposito possuem atividades que impactam diretamente com o cliente. O treinamento das pessoas envolvidas é também fundamental para o sucesso do projeto.[6]

### 4.1.1 Objetivos e Benefícios da Automação Comercial

Acompanhar as transformações que estão acontecendo e fazer da tecnologia sua aliada inseparável, é o principal foco do meio empresarial nos dias de hoje. O mercado exige que as empresas reorientem suas estratégias à necessidade do consumidor. É importante conhecer os hábitos de consumo dos clientes, e estar atento a mudanças de mercado.

Com a automação em pleno funcionamento, o empresário terá tempo para focar na estratégia do negocio e poderá dirigir sua atenção à capacitação profissional de seu pessoal, aos seus clientes, etc. [6]

A Automação Comercial exige investimentos que causam um profundo impacto nos processos da empresa e proporcionam benefícios concretos. Os benefícios da Automação Comercial iniciam na rapidez das informações obtidas dos processos e do controle sobre a movimentação dos produtos. Como conseqüência, a Automação Comercial confere mais produtividade e confiabilidade aos processos das empresas.

O gerenciamento de estoque eficaz permite comercializações e reposições de produtos, reduz a obsolescência e o nível do estoque, trazendo assim maior lucratividade e aumento de vendas.[6]

Existem diversos fatores determinantes para um projeto de automação, como por exemplo:

a) Solução mais adequada: é preciso atenção especial e dedicação em todo o projeto de automação para a escolha correta da solução de *hardware, software* e infra-estrutura. Uma

escolha malfeita pode comprometer o investimento transformando-se em custo e elevando os gastos na automação.

b) Capacitação dos Usuários: Necessário treinamento dos usuários que participam das atividades automatizadas. Não há como a automação funcionar adequadamente se os usuários não alimentarem os processos automatizados corretamente.

c) Manutenção de Aplicativos e Equipamentos: Apesar do custo envolvido a manutenção é indispensável para o funcionamento dos equipamentos e software.

d) Aspectos Culturais: A automação de uma empresa pode impactar a execução dos processos. A adaptação a novas tarefas enfrenta resistências, sempre presentes e que não podem ser negligenciadas. Todas as resistências devem ser tratadas de forma a não prejudicar a automação.

e) Tratamento das Informações: Há uma demanda por cadastros eficientes. Isso implica a necessidade de um rigoroso planejamento e administração desses cadastros, incluindo o alinhamento de dados. Se os cadastros viabilizam os benefícios da automação, eles exigem, em compensação, disciplina e tempo adicional das pessoas responsáveis pelos processos. Pode ocorrer, inicialmente, uma sobrecarga de informações, dada a abundância de dados antes indisponíveis. Assim, é preciso repensar os relatórios antigos e planejar seletivamente os novos, levando em conta conteúdo, formato e periodicidade que os tornem eficazes, bem como sua tecnologia de apresentação (*on line, batch*, consulta etc.). Ainda em razão da nova carga de informações, é indispensável que estas sejam avaliadas discriminando-se o que é relevante do que é apenas interessante saber. [6]

A Tabela 1 mostra um comparativo destas informações.

**Tabela 1 - Comparativo entre Custo e Benefícios**

| CUSTOS | BENEFÍCIOS |
|---|---|
| INVESTIMENTOS<br>• Equipamentos e *softwares*<br>• Redes de comunicação<br>• *Layout* e mobiliário | PRODUTIVIDADE<br>• Mais vendas devido aos melhores serviços e comodidades oferecidas<br>• Redução de custos<br>• Eliminação de papel |
| TREINAMENTO DE PESSOAL<br>• Na implantação<br>• De manutenção | |
| MANUTENÇÃO<br>• Equipamentos<br>• Aplicativos<br>• Cadastros e bancos de dados | QUALIDADE<br>• Melhoria no atendimento<br>• Redução de erros<br>• Aumento de segurança |
| IMPACTO CULTURAL<br>• Novos processos e tarefas<br>• Muito mais informação para ser tratada e absorvida<br>• Resistências | COMUNICACAO VISUAL<br>• Facilidade e atratividade para clientes<br>• Identidade corporativa moderna e personalizada |
| CAPACITAÇÃO TÉCNICA<br>• Atualização permanente<br>• Manter equipe mínima | GESTÃO EFICAZ<br>• Facilidade para apuração de resultados<br>• Segurança e agilidade das informações<br>• Eficiência na administração do fluxo de caixa |

Fonte: GS1 Brasil (2007)

## 4.2 Automação no ramo alimentício

Segundo GS1 Brasil [6], o consumo de alimentos fora do lar, o conhecido mercado de *Food Service*, vem crescendo significativamente no Brasil. A globalização dos mercados, o surgimento de novos modelos de gestão estão fazendo com que as empresas invistam cada vez mais na sua eficiência e assim se tornam mais competitivas, em virtude do acesso mais rápido e fácil às informações.

Após automatizar as operações de uma empresa, o empresário poderá focar em aspectos estratégicos de seu negocio e poderá dirigir sua atenção à capacitação profissional de seu pessoal e às necessidades de seus clientes.[6]

## 4.3 Análise de softwares

Um dos softwares presentes no setor de automação do setor alimentício é o *Pocket Cheff* da *Cheff Solutions*. Tal software já possui em sua solução a comanda eletrônica, utilizando *PocketPC*. Sua desvantagem é o custo da implantação. [8]

Outro sistema é o Comanda Ideal, software desenvolvido pela PLSoft. Utiliza *PocketPC* em sua solução e seu custo de implantação também é elevado. [13]

Estes sistemas possuem características que atendem a necessidade de se usar comandas eletrônicas para atender pedidos. O mais sofisticado é o Pocket Cheff, seguido pelo Comanda Ideal, que trazem na implantação de suas soluções, a necessidade de infra estrutura maior, com computadores de retaguarda mais potentes e impressoras sem fio para operações de crédito e fechamento de mesas entre o computador de mão e elas.

O software TabRestaurante versão 2.22 da Microtab Sistemas é um sistema voltado para atender necessidades operacionais de restaurantes, bares, padarias e redes de *fast food*. Possui módulos para cadastro de produtos, controle de estoques, controle de vendas e relatórios. Algumas das vantagens do sistema são a facilidade de abertura e fechamento de mesas e o custo da implantação. Também é de fácil treinamento e utilização pelos usuários. Sua desvantagem é que os pedidos são feitos manualmente, ocorrendo erros eventuais de preenchimento, identificação do produto vendido e atrasos no atendimento.

## 4.4 Análise de *hardware* para terminal de pedido

### 4.4.1 *Pocket PC*s

Os *Pockets PC* são terminais de uso mais abrangentes. Os *pockets* possuem processadores, memória, câmera fotográfica e muitas outras características. Conhecidos também como *PDAs* ou *Handhelds*, os *Pocket PC* são computadores de dimensões reduzidas, dotados de grande capacidade computacional, com funções de agenda, sistema de escritório elementar, com possibilidade de interconexão com um computador pessoal e uma rede sem fios, para acesso de email, internet e *Bluetooth*. [9]

O sistema *Windows Mobile* compatível com o sistema operacional *Windows* de computadores e servidores, foi adotado por alguns fabricantes de *PDAs*.

Os *Pocket PC* são uma solução de preço elevado e seus requisitos computacionais e de comunicação são superdimensionados para as tarefas de gerenciamento de pedidos das mesas.

### 4.4.2 Shellby

O Shellby 915 funciona como um terminal simples. Tudo que é digitado em seu teclado é enviado pra o computador através da Base RF, onde um programa processa estes dados decidindo se serão apresentados ou não no display do Terminal. Todas as funcionalidades que o terminal faz, são processadas pelo sistema hospedado no computador servidor. Se o terminal precisa solicitar código do produto, operador, quantidade, ou qualquer outro campo para entrada de dados, ou se será utilizado para efetuar vendas ou outro tipo de operação, o servidor remoto determinará o que deve ser feito e exibido para o usuário. [3]

O terminal Shellby funciona através de radio frequência na faixa de 915 MHz, comunicando com um computador através de uma Base RF com *interface USB* ou serial RS-232. Possui teclado numérico além das teclas *enter*, *backspace*, esc, vírgula e função. As configurações de parâmetros de RF da Base, o gerenciamento de dados de entrada e saída do terminal é feita através de uma *DLL* para Win32 fornecida pelo fabricante do terminal.

O Shellby possui em sua ultima linha do *display*, indicadores de *Status* que nos dão diversas informações úteis sobre suas configurações e funcionamento, conforme Figura 1:

Fonte: Manual Shellby (2010)
**Figura 1 - Display do Terminal Shellby 915**

1- Link de Rádio: indica que o terminal possui um link de rádio com a Base RF. Este ícone deve estar sempre presente em momentos de utilização. A falta deste ícone se deve ao fato da perda de comunicação com a Base RF, neste caso verifique as configurações de terminal e base, e certifique que o sistema gerenciador está sendo executado.

2- Sincronismo: indica o sincronismo com a Base RF. Este ícone deve piscar de tempos em tempos, caso contrário significa que o terminal está sem comunicação.

3- Status de Bateria: indica a carga de bateria.

4- ID do Terminal: mostra o Número I do Shellby configurado.

5- Número de Base: mostra o Número da Base configurado.

6- Número de Canal: mostra o Canal configurado.

7- Status de Comunicação: indica qual situação o terminal se encontra na rede RF. As setas indicam o sentido da comunicação. Se as duas setas estiverem presentes, a comunicação entre Base RF e Terminal está presente. Se o ícone for um “X”, não há comunicação com a Base RF. Se o ícone for “O” indica interferência na rede.

8- Indicador de Autenticação: Indica que o Terminal está no processo de autenticação com a *DLL*. Após a checagem a *DLL* efetuará a liberação da comunicação com o aplicativo, apagando o Indicador de Autenticação seguido de um *beep*. Se o parâmetro de Autenticação não estiver habilitado na *DLL* a verificação não será realizada e ocorrerá a liberação automaticamente, apagando o Indicador de Autenticação seguido de um *beep*.

9- Indicador de Função: indica que a tecla Fcn foi pressionada e o teclado encontra-se em modo “Função”. Caso este ícone não esteja presente, significa que o teclado está em modo “digitação”. [11]

Ausência do ícone nº 1 e a presença de um X no ícone 7, é ilustrada na Figura 2.

Fonte: Manual Shellby (2010)
**Figura 2- Exemplo de Terminal sem comunicação**

O terminal Shellby 915 é um *hardware* de baixo custo, de simples utilização, configuração e uso. Tais características tornaram o Shellby 915 a solução de *hardware* apropriada para o desenvolvimento deste trabalho.

## 4.5 A Linguagem de Modelagem Unificada (UML)

Com o surgimento de sistemas mais complexos e completos, veio a necessidade de uma reavaliação da forma de se desenvolver sistemas. Sendo assim, desde o primeiro computador até hoje, as técnicas utilizadas para a construção de sistemas computacionais têm evoluído de forma impressionante, notavelmente no que tange à modelagem de sistemas. [15]

Segundo Bezerra [15], nas décadas de 1950/60, os sistemas de *software* eram bastante simples, assim como as técnicas de modelagem. Era a época dos fluxogramas e dos diagramas de módulos.

Computadores mais avançados e acessíveis começaram a surgir na década de 1970, e houve uma grande expansão no mercado computacional. Neste período surgem a programação estruturada e o projeto estruturado. [15]

Na década de 1980 os computadores se tornaram ainda mais avançados e baratos e surgiu a necessidade de interfaces mais sofisticadas, o que originou a produção de sistemas de *software*s mais complexos. Neste período surgiu a Análise Estruturada. [15]

No início da década de 1990 surgiu um novo paradigma de modelagem, a Análise Orientada a Objetos. [15]

No fim da década de 1990 o paradigma da orientação a objetos atingiu sua maturidade e surgiu a Linguagem de Modelagem de Dados Unificada, em inglês *Unified Modeling Language*, cuja abreviação é "*UML*". [15]

Além da *UML*, surgiram várias propostas de técnicas para modelagem de sistemas segundo o paradigma orientado a objetos. Era comum duas técnicas possuírem diferentes notações gráficas para modelar uma mesma perspectiva de um sistema. Ao mesmo tempo, cada técnica tinha seus pontos fortes e fracos em relação à notação utilizada. [15]

Percebeu-se então a necessidade da existência de uma linguagem que viesse a se tornar um padrão para a modelagem de sistemas.

Segundo Bezerra [15], uma notação definida para *UML* é uma união de diversas notações preexistentes, com alguns elementos removidos e outros elementos adicionados com o objetivo de torná-la mais expressiva.

# 5 DESENVOLVIMENTO

## 5.1 Desenvolvimento do módulo de pedidos

### 5.1.1 Modelo de Processo de desenvolvimento

Para resolver problemas reais, deve-se incorporar uma estratégia de desenvolvimento que envolva as camadas do processo. Um modelo de processo é escolhido com base na natureza do projeto e da aplicação, nos métodos e ferramentas a serem utilizados. [1]

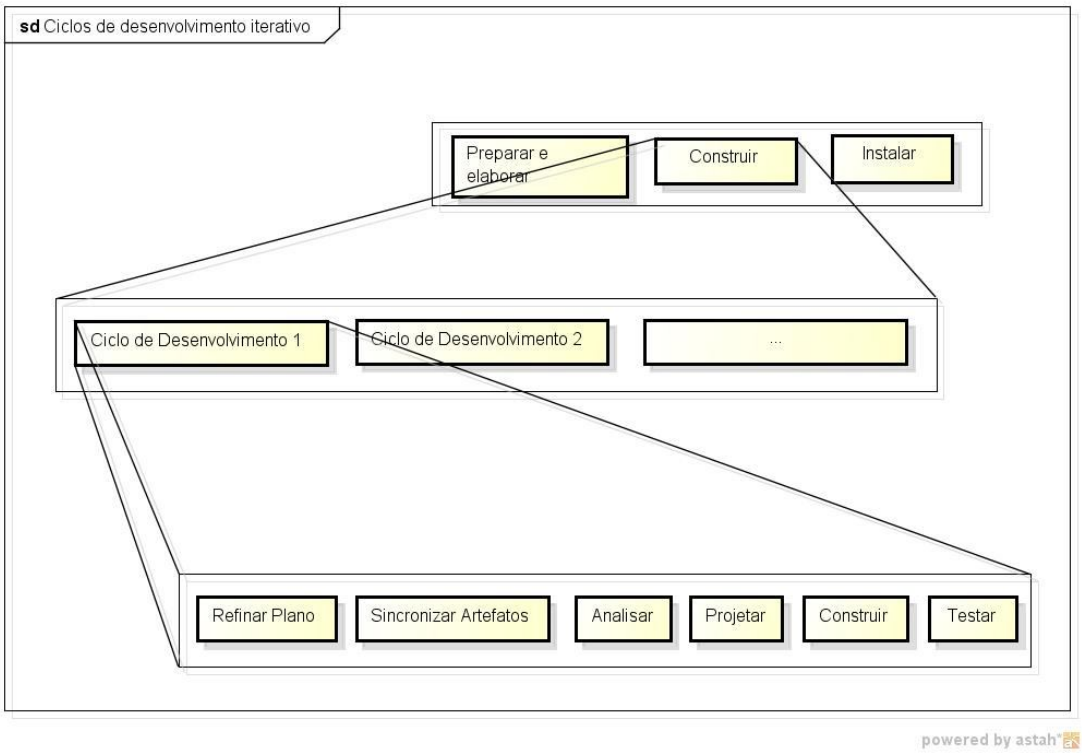
Para este processo de desenvolvimento foi utilizado um Modelo Evolucionário Incremental por ser um modelo iterativo que permite desenvolver versões cada vez mais completas do *software*.

O desenvolvimento iterativo possui algumas vantagens, dentre elas:

- a complexidade não se torna incontrolável .
- devido a velocidade de implementação, a realimentação é gerada cedo no processo. [4]

Um ciclo de vida iterativo se baseia no aumento e no refinamento sucessivo de um sistema através de múltiplos ciclos de desenvolvimento de análise, de projeto, de implementação e de teste [4].

Cada ciclo, conforme Figura 3, trata um conjunto relativamente pequeno de requisitos, trabalhando através da análise, do projeto, da construção e do teste.



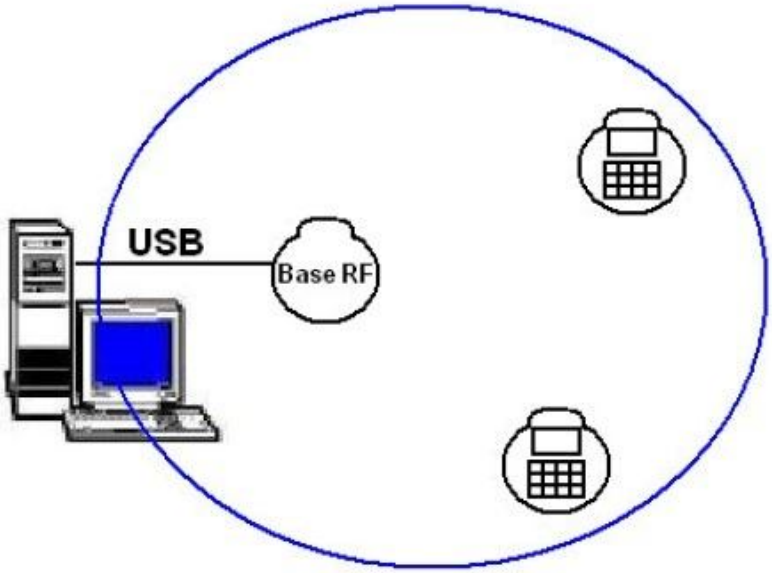

Fonte: LARMAN (2007)
**Figura 3 - Ciclos de desenvolvimento iterativo**

### 5.1.2 Protocolo de comunicação do Terminal com a Base RF

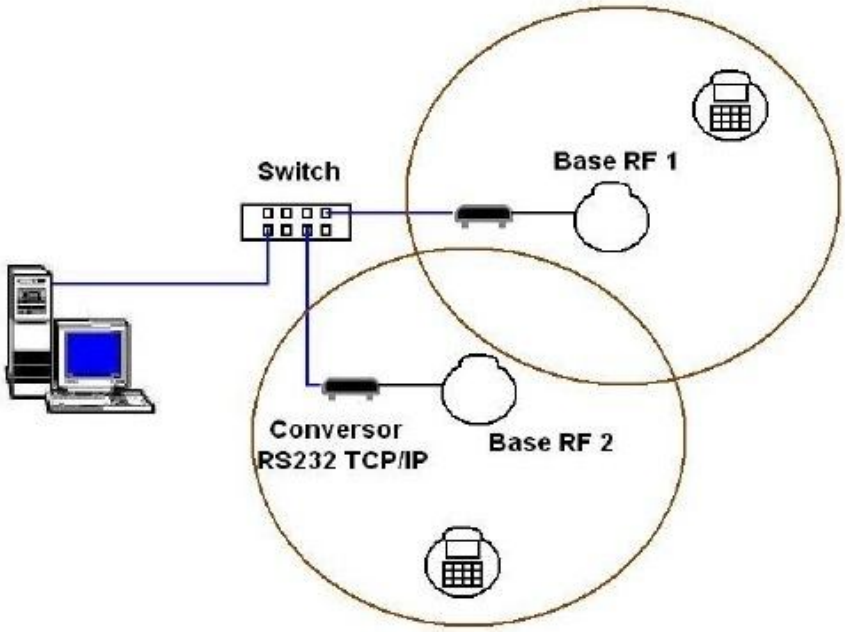
O protocolo de comunicação dos terminais oferece funções para escrita de dados, posicionamento do cursor e identificação dos terminais reconhecidos na rede.

A Base Shellby para os terminais Shellby 915 é conectada no computador da seguinte forma:

- USB: na porta USB do computador Figura 4.
- *Ethernet*: utilizando um conversor *Ethernet* para Serial RS-232 Figura 5.




Fonte: Manual Base RF (2009)
**Figura 4 - Base RF conectada diretamente a porta USB do PC**




Fonte: Manual Terminal Shellby (2009)
**Figura 5 - Base RF conectada a um Conversor RS-232 TCP/IP**

**Protocolo de Comunicação**

Para que os computadores de uma rede possam trocar informações é necessário que todos adotem as mesmas regras para o envio e o recebimento de informações. Este conjunto de regras é conhecido como Protocolo de comunicação.[7]

A *LAN* sem fio precisa atender os mesmos tipos de requisitos típicos de qualquer comunicação, incluindo alta capacidade, capacidade de cobrir curtas distâncias, total conectividade entre as estações conectadas e capacidade de *broadcast*. [12]

Ainda segundo STALLINGS [12], as *LAN*s sem fio geralmente são categorizadas de acordo com a técnica de transmissão que é usada.

Os produtos de *LAN* sem fio pertencem a uma das seguintes categorias:

- *LAN*s infravermelhas (*IR*): uma célula individual de uma *LAN IR* é limitada a uma única sala, pois a luz infravermelha não penetra em paredes opacas.
- *LAN*s de amplo espectro: esse tipo de *LAN* usa a tecnologia de transmissão de amplo espectro. Na maioria dos casos, essas *LAN*s operam nas bandas *ISM*.
- Microondas de banda estreita: essas *LAN*s operam em freqüências de microondas, mas não usam amplo espectro.

A Tabela 2 resume algumas das características principais dessas três categorias.

**Tabela 2 - Comparação entre as tecnologias de *LAN* sem fio**

| | **Infravermelho** | | **Amplo espectro** | | **Rádio** |
|---|---|---|---|---|---|
| | **Infravermelho difuso** | **Infravermelho de feixe dirigido** | **Salteamento de freqüência** | **Sequência direta** | **Microondas de banda estreita** |
| **Velocidade de dados (Mbps)** | 1 a 4 | 1 a 10 | 1 a 3 | 2 a 50 | 10 a 20 |
| **Mobilidade** | Estacionaria/móvel | Estacionaria | Movel | Estacionaria/móvel | |
| **Faixa (m)** | 15 a 60 | 25 | 30 a 100 | 30 a 250 | 10 a 40 |
| **Detecbilidade** | Mínima | | Pequena | | Regular |
| **Comprimento de onda/freqüência** | A: 800 a 900 nm | | 902 a 928 MHz<br>2,4 a 2,4835 GHz<br>5,725 a 5,85 GHz | | 902 a 928 MHz<br>5,2 a 5,775 GHz<br>18,825 a 19,205 GHz |
| **Técnica de modulação** | ASK | | FSK | QPSK | FSK/QPSK |
| **Energia irradiada** | - | <1W | 25 mW | | |
| **Método de acesso** | CSMA | Token Ring, CSMA | CSMA | Reserva ALOHA, CSMA | |
| **Licença dirigida** | Não | | Não | | Sim, exceto *ISM* |

Fonte:STALLINGS (2005)

A base do Terminal Shellby é um equipamento de comunicação por radio freqüência na faixa de 915 MHz, sendo ela responsável pelo envio e recebimento de dados dos terminais, como também do gerenciamento de pacotes da rede RF. [3]

O Terminal Shellby 915 possui um subconjunto de comandos VT-100 um padrão utilizado para configurar e realizar tarefas de gerenciamento de sistemas em todos os estados de operação e falha. Os comandos implementados de controle estão descritos na Tabela 3. [10]

**Tabela 3 - Comandos Implementados de Controle**

| | |
|---|---|
| ESC[H | cursor home |
| ESC[J | apaga até o fim da tela |
| ESC[K | apaga até o fim da linha |
| ESC[A | cursor vai para a linha de cima |
| ESC[C | cursor desloca uma posição para direita |
| ESC[7m | habilita inversão de vídeo |
| ESC[27m | desabilita inversão de vídeo |
| ESC[4m | habilita modo sublinhado |
| ESC[24m | desabilita modo sublinhado |
| ESC[1m | desliga cursor |
| ESC[21m | liga curso |
| ESC[6m | display 4x16 |
| ESC[26m | display 7x20 |
| ESC[5i | habilita impressão simultânea |
| ESC[4i | desabilita impressão simultânea |
| ^H | *backspace* (08h) |
| ^J | *line feed* (0Ah) |
| ^G | *buzzer* (07h) |
| ESC[ll;ccH | posiciona cursor ll(linha) cc(coluna) |

Fonte: Manual Terminal Shellby (2009)

### 5.1.3 Descrição dos casos de uso do módulo

Um caso de uso fornece a sequência de um ator (agente externo) que utiliza um sistema para realizar um processo. Casos de uso são especificações de requisitos ou especificação funcional. [4]

As Figura 6 e Figura 7 mostram os diagramas de caso de uso utilizados.

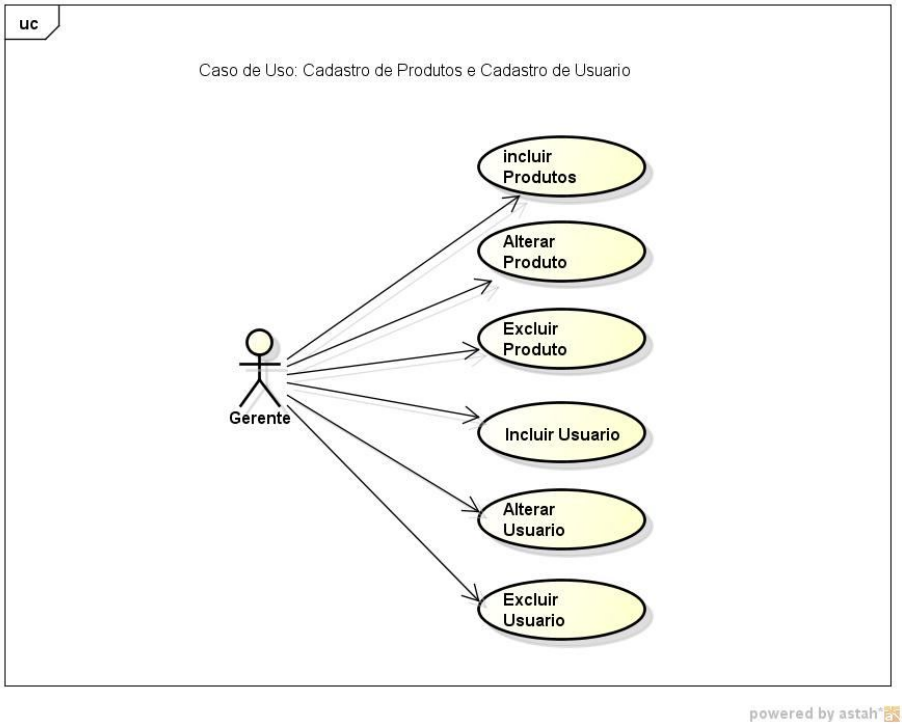



**Figura 6 - Caso de Uso de Cadastro de Produtos e Cadastro de Usuário**

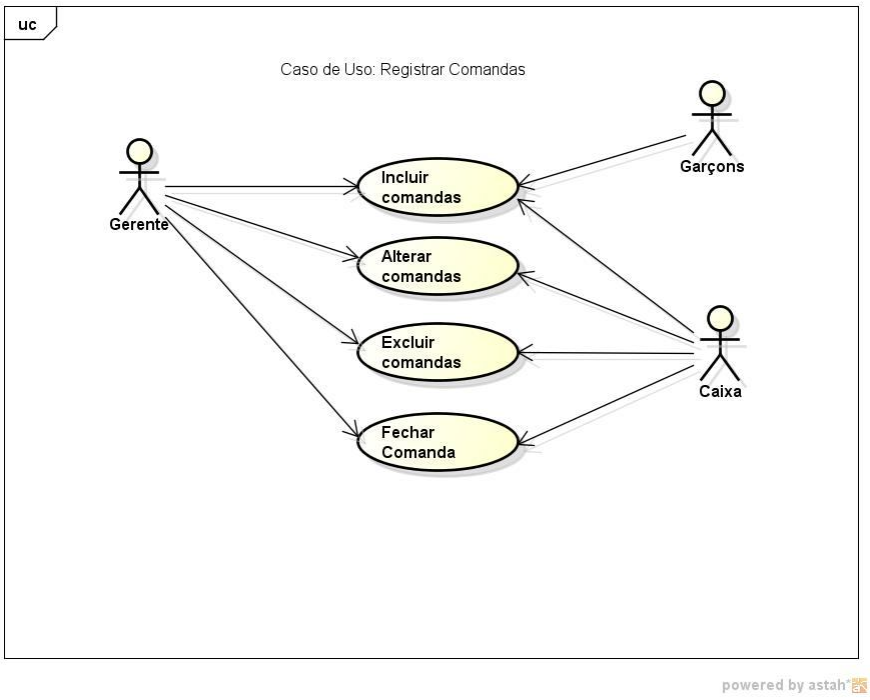



**Figura 7 - Caso de Uso de Registrar Comandas**

Os quadros a seguir descrevem os casos de uso das funcionalidades:

- Cadastro de Produtos
- Cadastro de Usuários
- Registro de Comandas

| **Nome do Caso de Uso:** Cadastro Produtos | **Funcionalidade:** Incluir de Produtos |
|---|---|
| **Ator(es) Envolvido(s):** Gerente | |
| **Pré Condição** | |
| - Dados do produto disponíveis | |
| **Descrição do Caso** | |
| - Verificar se o produto já se encontra cadastrado<br>- Se já esta cadastrado, termina<br>- Se não está, inserir dados do produto<br>- Gravar os dados | |
| **Pós Condição** | |
| - Produto incluído | |

| **Nome do Caso de Uso:** Cadastro Produtos | **Funcionalidade:** Alterar Produtos |
|---|---|
| **Ator(es) Envolvido(s):** Gerente | |
| **Pré Condição** | |
| - Dados do produto | |
| **Descrição do Caso** | |
| - Selecionar produto<br>- Se o cadastro do produto for encontrado, fazer as alterações e gravar os dados<br>- Se não encontrar, termina | |
| **Pós Condição** | |
| - Produto alterado no cadastro | |

| **Nome do Caso de Uso:** Cadastro de Produto | **Funcionalidade:** Excluir Produtos |
|---|---|
| **Ator(es) Envolvido(s):** Gerente | |
| **Pré Condição** | |
| - Dados do produto disponíveis | |
| **Descrição do Caso** | |
| - Selecionar produto<br>- Se o cadastro do produto for encontrado, excluir<br>- Se não encontrar, termina | |

| **Pós Condição** |
|---|
| - Produto excluído do cadastro |

| **Nome do Caso de Uso:** Cadastro de Usuário | **Funcionalidade:** Incluir Usuários |
|---|---|
| **Ator(es) Envolvido(s):** Gerente | |
| **Pré Condição** | |
| - Dados do usuário disponíveis | |
| **Descrição do Caso** | |
| - Verificar se o usuário já se encontra cadastrado<br>- Se já esta cadastrado, termina<br>- Se não está, inserir dados do usuário<br>- Gravar os dados | |
| **Pós Condição** | |
| - Usuário incluído | |
| **Nome do Caso de Uso:** Cadastro de Usuário | **Funcionalidade:** Alterar Usuários |
| **Ator(es) Envolvido(s):** Gerente | |
| **Pré Condição** | |
| - Dados do usuário | |
| **Descrição do Caso** | |
| - Selecionar usuário<br>- Se o cadastro do usuário for encontrado, fazer as alterações e gravar os dados<br>- Se não encontrar, termina | |
| **Pós Condição** | |
| - Usuário alterado no cadastro | |

| **Nome do Caso de Uso:** Cadastro de Usuário | **Funcionalidade:** Excluir Usuários |
|---|---|
| **Ator(es) Envolvido(s):** Gerente | |
| **Pré Condição** | |
| - Dados do usuário disponíveis | |
| **Descrição do Caso** | |
| - Selecionar usuário<br>- Se o cadastro do usuário for encontrado, excluir | |

| |
|---|
| - Se não encontrar, termina |
| **Pós Condição** |
| - Usuário excluído do cadastro |

| | |
|---|---|
| **Nome do Caso de Uso:** Registro Comanda | **Funcionalidade:** Incluir Comandas |
| **Ator(es) Envolvido(s):** Gerente, Garçons e caixas | |
| **Pré Condição** | |
| - Dados dos produtos disponíveis | |
| **Descrição do Caso** | |
| - Selecionar produtos<br>- Determinar quantidade de cada produto<br>- Gravar os dados | |
| **Pós Condição** | |
| - Comanda incluída | |
| **Nome do Caso de Uso:** Registro Comanda | **Funcionalidade:** Alterar Comandas |
| **Ator(es) Envolvido(s):** Gerente e caixas | |
| **Pré Condição** | |
| - Dados da comanda já cadastradas | |
| **Descrição do Caso** | |
| - Selecionar comanda<br>- Se o cadastro da comanda for encontrado, fazer as alterações e gravar os dados<br>- Se não encontrar, termina | |
| **Pós Condição** | |
| - Comanda alterada no cadastro | |

| | |
|---|---|
| **Nome do Caso de Uso:** Registro comanda | **Funcionalidade:** Excluir Comandas |
| **Ator(es) Envolvido(s):** Gerente e caixas | |
| **Pré Condição** | |
| - Dados da comanda já cadastrada | |
| **Descrição do Caso** | |
| - Selecionar comanda<br>- Se o cadastro da comanda for encontrado, excluir | |

| |
|---|
| - Se não encontrar, termina |
| **Pós Condição** |
| - Comanda excluída do cadastro |

| | |
|---|---|
| **Nome do Caso de Uso:** Registro Comanda | **Funcionalidade:** Fechar Comandas |
| **Ator(es) Envolvido(s):** Gerente e caixas | |
| **Pré Condição** | |
| - Dados da comanda já cadastrada | |
| **Descrição do Caso** | |
| - Selecionar comanda<br>- Se o cadastro da comanda for encontrado, fechar<br>- Se não encontrar, termina | |
| **Pós Condição** | |
| - Comanda fechada no cadastro | |

### 5.1.4 Linguagem de programação

A linguagem de programação definida foi o Delphi na sua versão 7.0, com esta é possível desenvolver aplicações no padrão *Windows Form* de forma prática, produtiva e simples. Por este motivo, atualmente esta linguagem é muito utilizada no mercado de desenvolvimento de *softwares*, sendo altamente indicada para o desenvolvimento de aplicações comerciais [5].

### 5.1.5 Armazenamento dos dados

O formato escolhido para armazenamento de dados *Dbase*, DBF. Este formato ainda é usado por aplicativos embarcados acessados de um único ponto.

O servidor do sistema armazena em banco de dados, todas as operações realizadas através dos pedidos feitos, garantindo assim maior confiabilidade e integridade das informações que chegam ao sistema gerenciador de pedidos, facilitando assim o fechamento de contas, inclusões e alterações de pedidos e produtos, como também *backup*.

## 5.2 Programação e configuração do terminal de pedidos

A Wilbor Tech fornece também um sistema para testes da Base RF e do Terminal Shellby 915. Com ele é possível configurar todos os parâmetros de comunicação RF, e enviar comandos para o terminal, como limpar *display*, mensagens na tela, tamanho da fonte, entre outros. Como ilustrado nas Figura 8 e Figura 9.

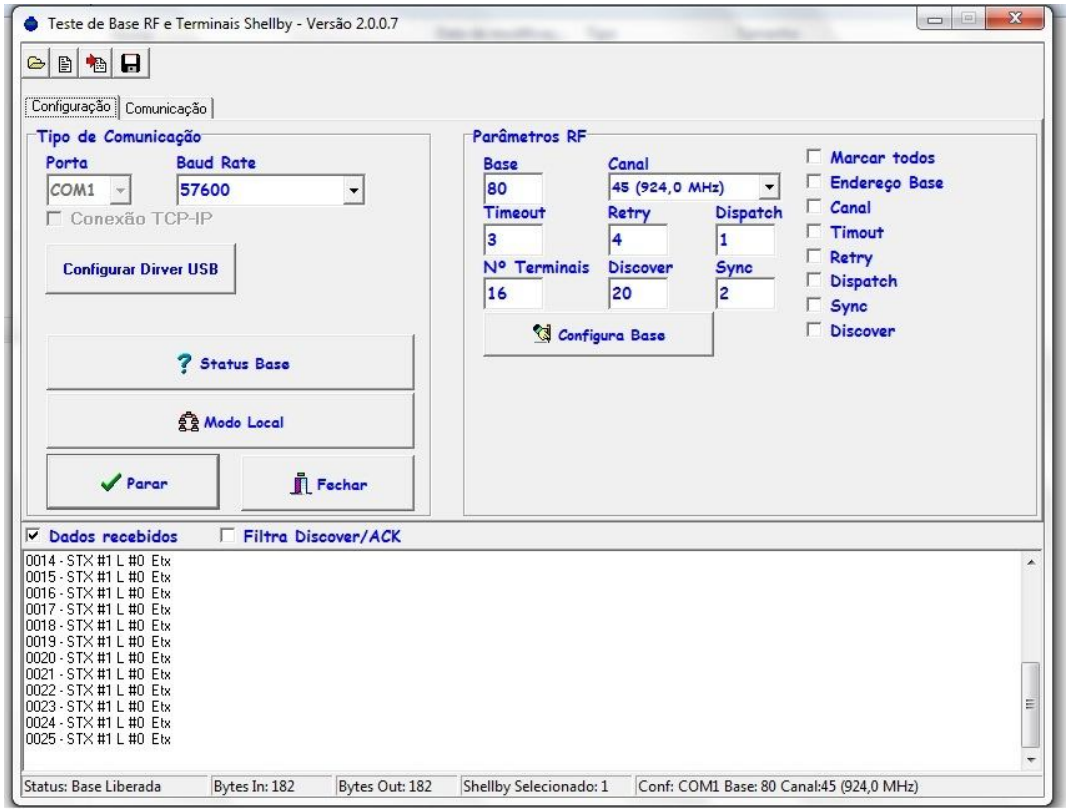



**Figura 8 - Teste da Base RF e Terminal Shellby (configuração)**

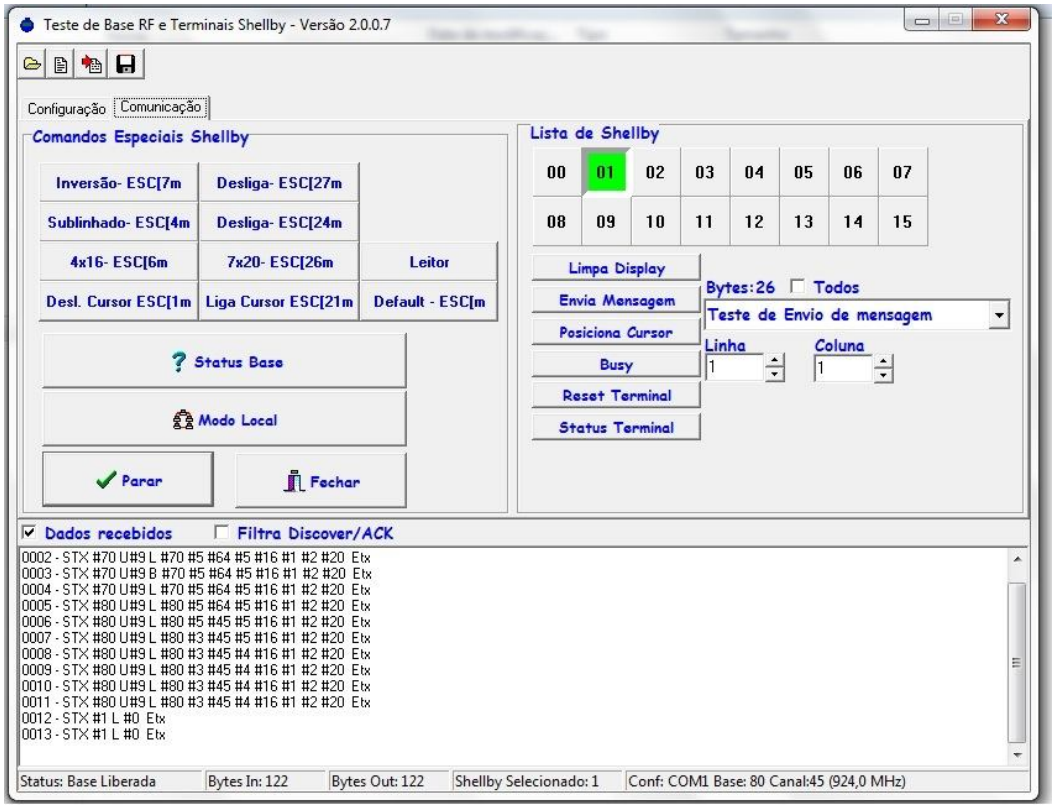



**Figura 9 - Teste da Base RF e Terminal Shellby (comunicação)**

A Wilbor Tech disponibiliza uma *DLL*, chamada Wtechlpt versão 4.x.x.x para comunicação com os terminais. Para o uso dos terminais, como limpeza de tela, envio de mensagem para o display, posicionamento do cursor, etc., o sistema deve chamar as funções contidas nesta *DLL*.

O sistema precisa de um Timer, que fará a varredura de teclado dos terminais. Em intervalos regulares verificará se alguma tecla do terminal foi pressionada e fazer o devido tratamento, ecoando ou não a informação no display. [11]

Cada *DLL* trabalha em conjunto com um arquivo de configuração INI, onde são informadas as configurações de *interface* e lista de terminais habilitados. Os arquivos de configuração INI devem ser colocados no mesmo diretório do executável do sistema, e as *DLL*'s devem permanecer nos diretórios de sistema do *Windows* ou do executável.

O *buffer* representa a área de armazenamento de toda informação que entra no Terminal. Portanto os dados são sempre lidos do *buffer* do Terminal, simplificando assim o desenvolvimento, pois não há a necessidade de escrever uma rotina em separado para cada dispositivo de entrada.

## 5.3 Etapas do desenvolvimento

Este projeto foi organizado em dois ciclos de desenvolvimento iterativo. A intenção desta estratégia é um modelo de aprendizado "*just-in-time*", realçando a apresentação de idéias usadas em primeiro lugar, próximas o bastante do momento de aprendê-las e prosseguir o trabalho.[4]

Primeiro Ciclo: destina-se a uma aplicação simples de funções no núcleo do sistema de digitação de comandas.

Segundo Ciclo: expandir a funcionalidade do sistema

Após uma visão dos aspectos de programação e configuração do Terminal Shellby, iniciou-se a codificação do software utilizando a linguagem de programação Delphi 7.0 e o sistema gerenciador de banco de dados *Dbase*.

No Apêndice B encontra-se exemplos de uso da *DLL*, escrito com a linguagem de programação Delphi 7 que inclui a comanda eletrônica, isto feito no primeiro ciclo.

O uso das *DLL´*s é igual a chamada de uma função comum.

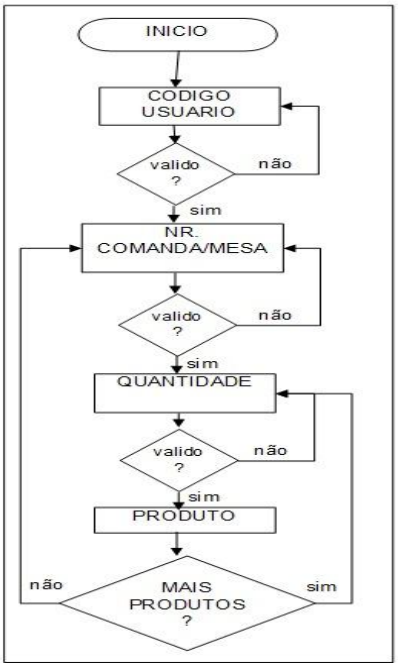



**Figura 10 - Fluxograma de lançamento de comanda eletrônica**

Na Figura 10 observa-se o fluxograma do módulo. O garçom efetuara pedidos para mesas e comandas. O servidor do sistema recebe os dados do pedido e aguarda novas informações dos terminais. Os dados relativos às operações são registrados em uma base de dados.

A usabilidade é um fator muito importante neste projeto, as interfaces projetadas visam um entendimento por intuição pelos funcionários e com o objetivo dos mesmos executarem todas as funções em um menor tempo possível.

No segundo ciclo foi possível adaptar funções na tela do terminal de forma que todo este processo se tornar mais simples para o usuário.

## 5.4 Visão Geral do sistema

Para o inicio do lançamento das comandas eletrônicas, através do terminal Shellby deve-se digitar o número de identificação do usuário Figura 11. Cada atendente deve possuir um número de identificação, que será informado somente no inicio da abertura de lançamentos de comandas.

**Figura 11 - Identificação do usuário do terminal**

Em seguida é informado o número da comanda ou da mesa Figura 12.

**Figura 12 - Lançamento do número da comanda/mesa**

Se este número de comanda ou mesa não for valido, o sistema apaga o número informado e aguarda a digitação de um número dentro dos parâmetros. Se a comanda ou mesa estiver em fase de fechamento, também não será aceito lançamentos até que seja efetivamente fechada a conta. Só assim é liberado para a digitação dos produtos que serão lançados na comanda eletrônica.

A partir deste momento o terminal aguarda por x segundos, valor este especificado na configuração do sistema, para que o usuário informe algum código de produto válido. Depois de informado o código do produto Figura 13, se o usuário não digitar mais nada no terminal por x segundos, o mesmo retorna para a rotina de solicitação de número de comanda ou mesa.

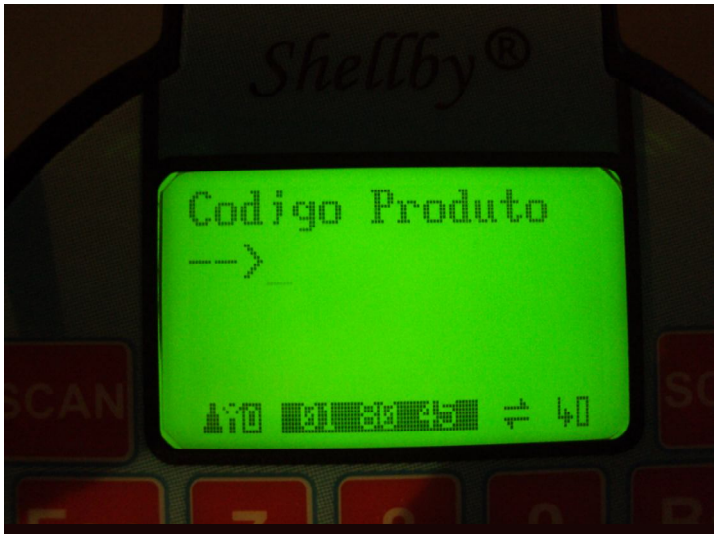



**Figura 13 - Lançamento de produtos na comanda eletrônica**

Para o inicio do sistema de comunicação no servidor o computador utiliza-se o login ilustrado na Figura 14. Se o usuário não tiver permissão, poderá ser cadastrado, porém somente com a senha do supervisor (gerente).

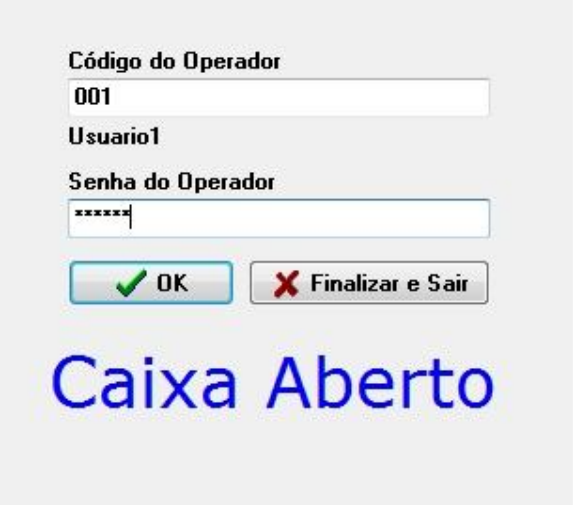



**Figura 14 - Login do sistema de retaguarda**

Na abertura da mesa, é possível modificar o número de ocupantes, que já assume o valor padrão colocado na configuração Figura 15, o que facilita no fechamento da mesma.

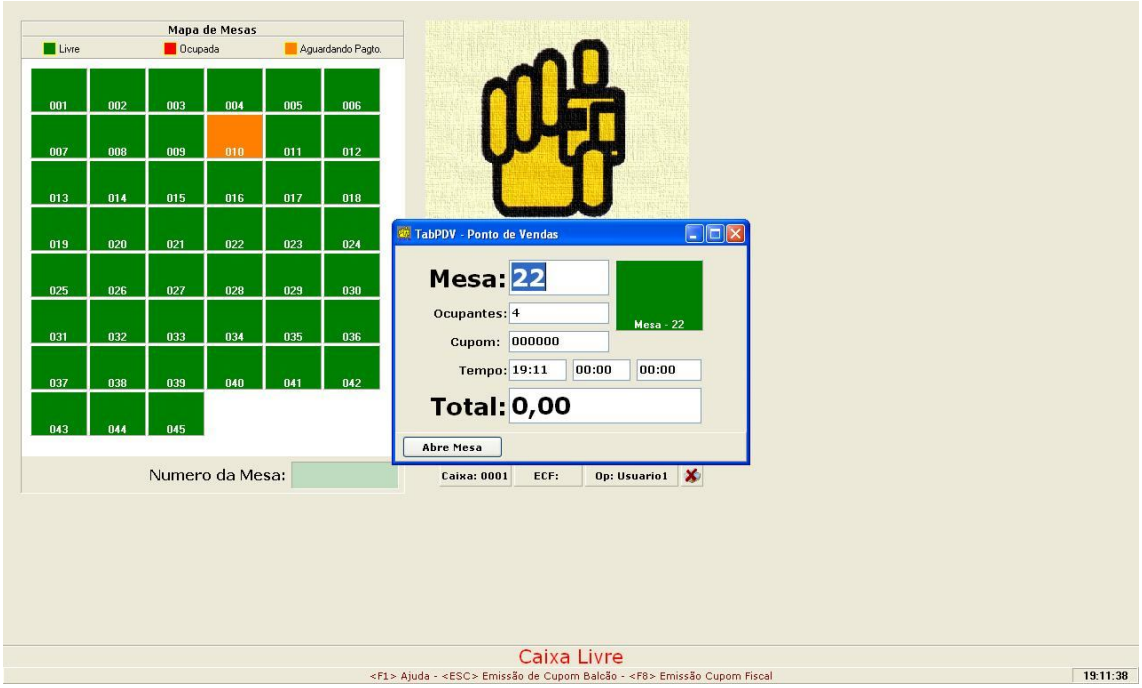



**Figura 15 - Abertura de mesa ou comanda**

O fechamento da mesa ou comanda pode ser feito pelo seu número, como ilustra a Figura 16.

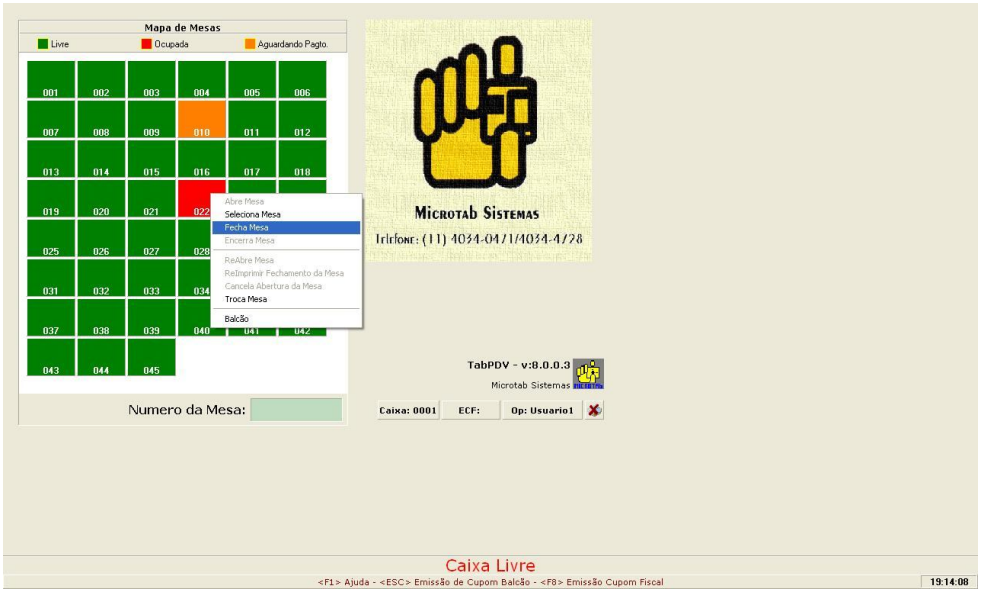



**Figura 16 - Fechar mesa ou comanda**

A forma de pagamento e definida em seguida como ilustra a Figura 17.

**Figura 17 - Formas de Pagamento**

As mesas livres, numeradas, aparecem no sistema na cor verde. As mesas ocupadas recebem a cor vermelha e a cor laranja é dada às mesas que pediram o fechamento da conta, cujo pagamento ainda não chegou ao caixa.

A todo o momento o usuário do sistema possui uma visão geral da situação das mesas, através das cores atribuídas para cada situação delas, conforme pode ser observado na Figura 18.

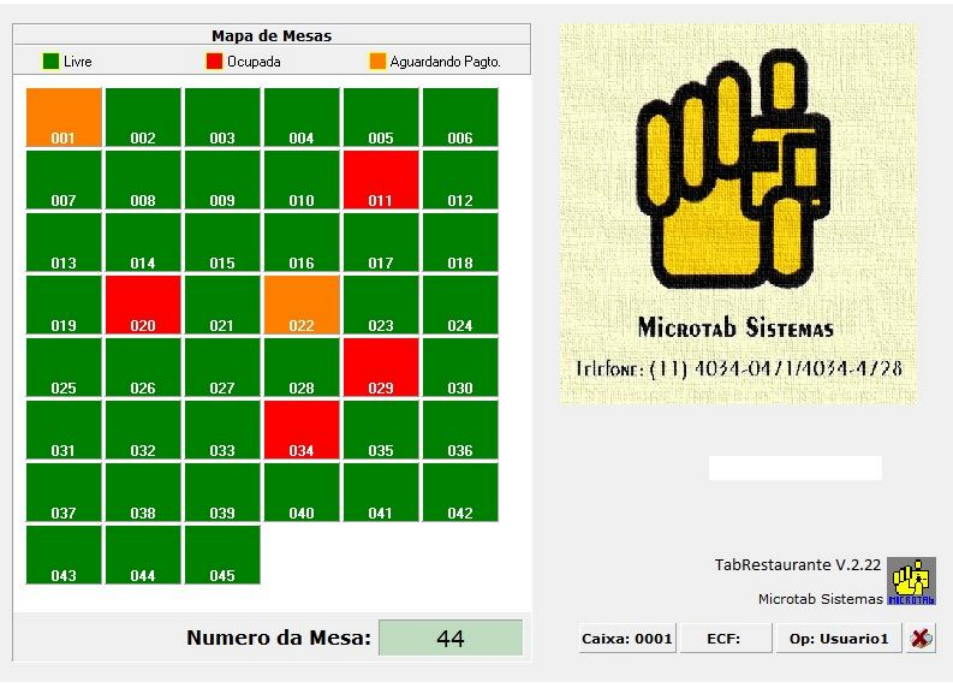



**Figura 18 - Mapa de mesas e comandas**

Os lançamentos realizados aparecem na tela de forma que o usuário possa acompanhar sua digitação de quantidade, produtos e valores Figura 19. Caso haja necessidade, o usuário pode cancelar ou excluir itens dos lançamentos efetuados, porém somente com a senha do supervisor, este operação pode ser realizada.

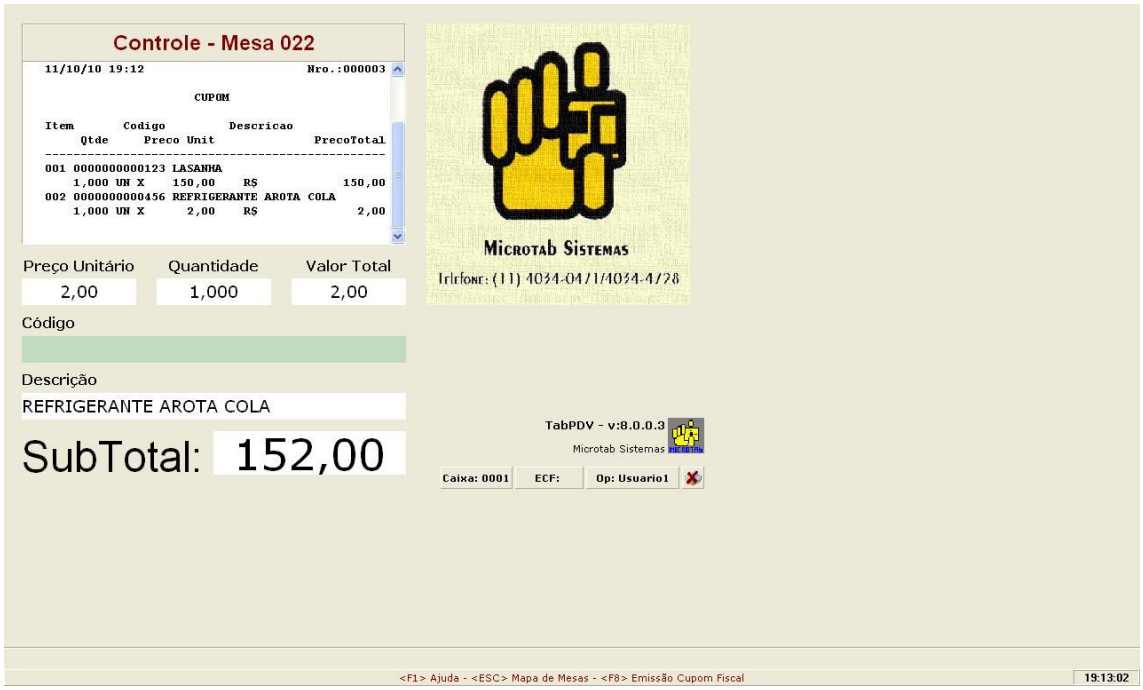



**Figura 19 - Lançamentos realizados**

# 6 TESTES

O processo de testes adotado é ilustrado na Figura 20. Os principais estágios do processo de testes realizados incluíram:

Teste de unidade: verificação dos componentes individualmente, garantindo que eles operem corretamente. Foram realizados testes de funcionamento do terminal e do computador, verificando seu funcionamento e operações.

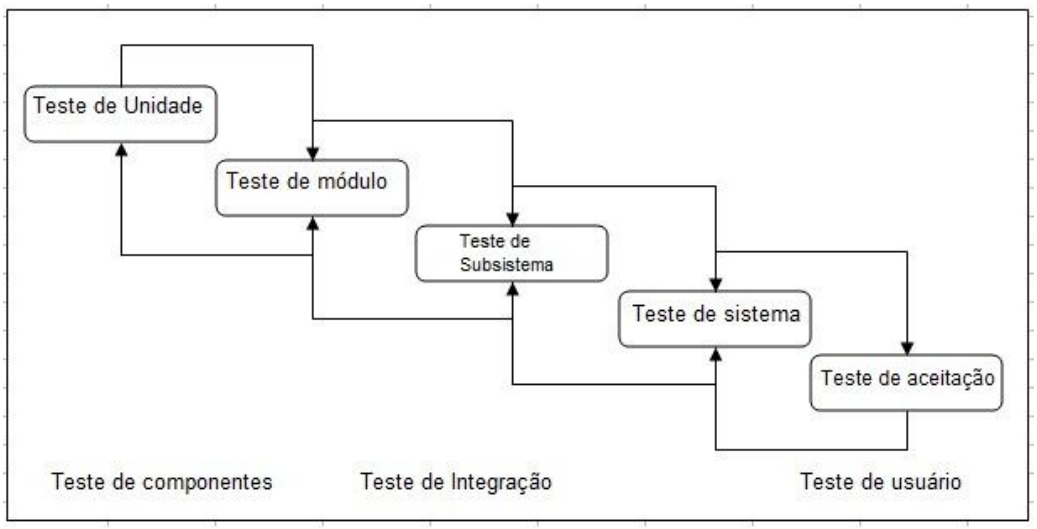
Teste de módulo: observação dos módulos de componentes dependentes, como classe de objetos. Foi verificado as funções do terminal como recebimento e envio de comandos e funções.

Teste de subsistema: Foram testados conjuntos de módulos integrados em subsistemas. A utilização das interfaces foi bem sucedida. Os módulos integrados ao sistema Tab-Restaurante operaram de maneira satisfatória.

Teste de sistema: os subsistemas são integrados para constituírem o sistema. A integração da *DLL* fornecida pelo fabricante não encontrou nenhum erro durante esta comunicação entre os subsistemas.

Teste de aceitação: neste processo final, foi realizado testes com dados efetivamente provenientes de abertura de mesas e comandas, com produtos já cadastrados pelo sistema de retaguarda. Foi encontrado um problema na quantidade de produtos lançados. Caso o usuário digite, de forma equivocada, uma quantidade muito grande do produto, ocorreu o travamento do sistema. A correção proposta inclui uma nova restrição quanto a quantidade digitada, limitando a mesma a no máximo 1000 unidades.

Os testes com o terminal Shellby exigiram observar as distâncias de transmissão, para verificar se o sinal está dentro da faixa necessária e se não possui interferências. Todos os testes realizados em distâncias de até 40 metros foram muito bem sucedidos e sem problemas de interferência.




Fonte: SOMMERVILLE (2008)
**Figura 20 - O processo de teste**

# 7 CONCLUSÃO

A criação do protótipo explorou algumas possibilidades do uso de tecnologias existentes na área de terminais de transmissão por radio freqüência, assim como os benefícios que estes aparelhos podem trazer. Ao se utilizar essas tecnologias em restaurantes pode-se obter a melhora do atendimento aos clientes e o aumento da eficiência dos controles internos.

As principais contribuições que o protótipo desenvolvido podem trazer para a indústria de serviços de alimentação incluem: agilidade, rapidez, eficiência no atendimento e a satisfação dos clientes e usuários atendidos pelo sistema.

# REFERÊNCIAS BIBLIOGRÁFICAS

[1] PRESSMAN, Roger S. **Engenharia de Software**, São Paulo: McGraw–Hill, 2006. 720 p.

[2] SOMMERVILLE, Ian. **Engenharia de Software**, São Paulo: Addison-Wesley, 2007-2008. 568 p.

[3] Website com informações do *hardware*. Disponível em: <URL:http://www.gradual.com.br> . Acessado em 04/03/2010

[4] LARMAN, Craig. **Utilizando UML e Padrões,** Porto Alegre: Bookman, 2007. 695 p.

[5] CANTÚ, Marco. **Dominando o Delphi 7: a bíblia,** São Paulo: Pearson Education do Brasil, 2003. 806 p.

[6] GS1 Brasil. Disponível em <URL:http://www.eanbrasil.org.br>. Acessado em 01/06/2010.

[7] TANENBAUM, ANDREW S. **Redes de Computadores,** 4ª ed. São Paulo: Campus, 2003. 955 p.

[8] Disponível em <URL:http://www.cheffsolutions.com/sistema-restaurantes-pocket-cheff.html>. Acessado em 10/10/2010

[9] Disponível em <URL:http://www.hp.com/latam/br/pyme/novidades/ipaqpockets_pc.html> Acessado em 10/10/2010

[10] Manual Terminal Shellby – Gradual Tecnologia Ltda. – Gradual Janeiro/2009

[11] Manual da *DLL* Wtechlpt Versão 4.x.x.x - Gradual Tecnologia Ltda. – Gradual Janeiro/2009

[12] STALLINGS, WILLIAM **Redes e Sistemas de Comunicação de Dados**, 5ª ed. São Paulo: Campus, 2005. 450 p.

[13] Disponível em <URL:http://www.comandaideal.com.br>. Acessado em 01/11/2010

[14] REGENSTEINER, ROBERTO J. **Elementos básicos para o planejamento da automação do varejo**, 3ª ed. São Paulo: Senac São Paulo, 2005. 108 p.

[15] BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com *UML*.** Rio de Janeiro: Elsevier, 2002. 222 p.

# APÊNDICES

# APÊNDICE A – *DLL* Shellby e *DLL* Wtechlpt Configuração do terminal Shellby

A *Dll* Shellby faz o tratamento dos Terminais Shellby 915 e Bases RF, funcionando em conjunto com a *Dll* Wtechlpt e um arquivo de configuração Shellby_Base1.ini. Neste arquivo, estão as configurações de Base RF USB ou RS232 com Conversor TCP-IP, teclas de função e mapeamento da rede RS485 para a rede RF usando uma única base. Para utilizar mais de uma Base RF, e necessário configurar outro arquivo.ini para cada Base (Shellby_BaseN, onde N varia de 1 a 8). Normalmente deve-se colocar a *DLL* e o arquivo INI no mesmo diretório do aplicativo gerenciador de Terminais.

**Seção Gerais**

Configurações gerais para compatibilizar o funcionamento com outros equipamentos.

**TrataDisplay:**

Valores validos 0 ou 1. Se configurado como 1, compatibiliza sistemas desenvolvidos para Terminais com display 2x40, pois os Terminais Shellby possuem display de 7x20. Neste caso, as linhas 1 e 2 do display correspondem a linha 0, e as linhas 3 e 4 do display correspondem a linha 1. Se o valor configurado for 0, não será feita a quebra de linhas, ou seja, não haverá tratamento de posicionamento de cursor. Então as linhas validas serão de 1 a 7, e colunas de 1 a 20.

**ESC:** valores validos 27 ou 127.

**- ESC=27** Quando sistema desenvolvido para Terminais *Ethernet.*

**- ESC=127** Quando sistema desenvolvido para Terminais RS485.

**LiberaSistema:** valores validos 0 ou 1. Valor padrão 0.

**- LiberaSistema=0** Quando o sistema utiliza Timer ou Thread.

**- LiberaSistema=1** Quando o sistema em LOOP infinito.

**Fcn_inicializacao:**

Valores validos F0 a F99. Envia para o sistema a tecla de função configurada, sempre que o Shellby for reconhecido pela base.

**Seção Função:**

Esta seção é opcional. Permite configurar as teclas de função retornar um determinado valor. Ao pressionar as teclas Fcn + <número>, a *Dll* retorna para o sistema qualquer caractere ASCII ou até mesmo uma string. Exemplo:

[Função]

F0=A

F1=*

F2=1234456789<13>

F0 enviara caractere “A” para a aplicação

F1 enviara caractere “*” para aplicação

F2 enviara texto “123456789” + Enter

Se a seção [Função] não existir, retornara o padrão VT100 (F1 a F10):

TECLAS FUNCAO STRING

‘Fcn’ + 0 F10 ESC O x

‘Fcn’ + 1 F1 ESC O P

‘Fcn’ + 2 F2 ESC O Q

‘Fcn’ + 3 F3 ESC O R

‘Fcn’ + 4 F4 ESC O S

‘Fcn’ + 5 F5 ESC O t

‘Fcn’ + 6 F6 ESC O u

‘Fcn’ + 7 F7 ESC O v

‘Fcn’ + 8 F8 ESC O l

‘Fcn’ + 9 F9 ESC O w

**Seção Base:**

A *Dll* faz o tratamento de até 8 bases simultaneamente. É nesta seção onde deve ser configurado o mapeamento da rede RS485 para a rede RF, ou seja, informamos que determinado ID de terminal na rede RS485 será um terminal RF com outro ID (podendo ser o mesmo valor).

**Tipo:**

Valores validos USB, *Client* ou *Server*. Informa o tipo de conexão da base, se ela é USB, ou se for na rede *Ethernet*, se trabalha como *Client* ou *Server*. Caso a base trabalhe na rede TCP/IP, recomendamos a configurá-la como *Client*, pois sua comunicação e mais eficiente em casos de queda de conexão.

 **Porta:**

Valor da porta para a conexão. Sendo tipo TCP configure conforme valor estabelecido na configuração do Conversor. Sendo do tipo USB, veja no Gerenciador de Dispositivo a porta COM gerada. Certifique-se de que esta COM esteja no intervalo de 1 a 15.

 **EnderecoIP:**

Campo valido somente para base configurada como Tipo *Client* ou *Server*, funcionando em rede *Ethernet*. Digite o endereço IP do Conversor.

 **BaudRate:**

Valores validos 115200 ou 57600. Define a velocidade de comunicação da porta COM do PC. Informe o mesmo valor que a velocidade da Serial da base.

 **EnderecoRF:**

Valores validos 64 a 96. Define o endereço da Base Radio Frequência.

 **Canal:**

Valores validos 1 a 64. Define o canal de frequência utilizado na comunicação da rede RF. Certifique-se de que o canal esta livre de interferência.

 **Timeout:**

Valores validos 1 a 10. Define o tempo de espera para recebimento de um pacote dos terminais. Recomenda-se fixar o valor de Timeout para 3.

 **Retries:**

Valores validos 1 a 10. Define o número de tentativas para retransmissão de um pacote na rede RF. Recomenda-se fixar o valor de Timeout para 4.

 **Discover:**

Valores validos 1 a 100. Define o tempo entre envio de comando de busca por novos terminais na rede RF. Este valor e multiplicado por 50 ms.

 **Sync:**

Valores validos 1 a 10. Define o tempo entre o recebimento de pacotes dos terminais na rede RF. Este valor e multiplicado por 50 ms.

 **Dispatch:**

Valores validos 1 a 10. Define o tempo entre o envio de pacotes para os terminais na rede RF. Este valor e multiplicado por 50 ms.

 **NTerminais:**

Valores validos 0 a 15. Define o número de terminais que a base ira tratar.

Os valores de Discover, Sync e Dispatch influenciam diretamente no desempenho da rede. Valor muito alto pode deixar a rede mais lenta, e valor muito baixo provoca um fluxo desnecessário de pacotes na rede RF.

Formato

**Seq=RSxx;RFyy**

Onde:

**Seq** → número sequêncial de 0 a 15.

**xx** → ID de terminal na rede RS485 entre 00 a 31.

**yy** → ID de terminal na rede RF entre 00 a 15.

Exemplo:

0=RS01;RF01 → ID 01 na rede RS485 será Terminal Shellby ID 01

1=RS04;RF02 → ID 04 na rede RS485 será Terminal Shellby ID 02

2=RS06;RF10 → ID 06 na rede RS485 será Terminal Shellby ID 10

**Exemplo completo do arquivo de configuração Shellby_Base1.ini:**

[Gerais]

TrataDisplay=1

ESC=127

LiberaSistema=0

Fcn_Inicializacao=F0

[Função]

F0=<127><127><127>

F1=A

F2=B

F3=C

F4=Z

F5=P

[Base]

Tipo=USB

Porta=7

BaudRate=57600

EnderecoRF=80

Canal=1

TimeOut=3

Retries=4

Discover=20

Sync=2

Dispatch=1

NTerminais=2

0=RS01;RF01

1=RS02;RF02

**Exemplo completo do arquivo de configuração Shellby_Base2.ini:**

[Base]

NumTerm=2

Tipo=Client

EnderecoIP=192.168.0.100

Porta=1001

BaudRate=57600

EnderecoRF=80

Canal=5

TimeOut=3

Retries=4

Discover=20

Sync=2

Dispatch=1
0=RS03;RF03
1=RS04;RF04

Na carga da aplicação deve-se chamar Configlpt para carregar as configurações das *DLL*'s auxiliares. Ao finalizar a aplicação deve-se chamar a rotina *DLL*_Close(); para descarregar corretamente as *DLL*'s auxiliares evitando mensagens de erros por violação de acesso.

Funções da *DLL* Wtechlpt:

Para o uso da Wtechlpt.DLL, é necessário que a linguagem de programação do sistema permita a utilização de *DLL*'s, o que ocorre na grande maioria dos casos.

- ConfigLpt(Endereço, Timeout)

  Retorno: Tipo *boolean*, *true* se conseguiu estabelecer a comunicação, e *false* se ocorreu algum erro no processo.

  Inicializa a *DLL* para a comunicação com os Terminais.

  Timeout: A comutadora tem um processador interno que é mais lento que o processador do PC que a controla, por isso é necessário informar um valor de timeout para que o PC espere um tempo entre o envio de dois comandos para a mesma. Quanto mais rápido o PC maior deve ser esse valor. Se for especificado um valor muito baixo pode haver perda de comandos enviados. Por outro lado se for especificado um valor muito alto, a performance do sistema pode ficar comprometida. Portanto para cada PC deve ser especificado um valor adequado para o timeout.

- DLL_Close()

  Para o sistema finalizar corretamente e sem mensagens de violação de acesso, deve-se interromper qualquer chamada a suas rotinas, normalmente DLL_Get(), DLL_Status() e/ou se houver dados para as *interfaces* LPT e COM do terminal, DLL_Print() e DLL_Serial(). Uma vez interrompida a comunicação com a DLL deve-se realizar a chamada a DLL_Close() antes de encerrar o Sistema.

- DLL_PosCur(Terminal, Lin, Col)

Posiciona o cursor do Terminal na posição indicada por Lin, Col.

DLL_Clear(Terminal)

Apaga o display do Terminal.

DLL_Echo(Terminal, Dado)

Escreve um caractere no Display.

DLL_Display(Terminal, Dado)

Escreve uma string no Display.

DLL_Get(Terminal)

Recebe um caractere do buffer:

Retorno: Chr(0) - Caso não haja dados no buffer.

Dado - Caso haja dado(s) no buffer, retorna o primeiro caractere do buffer.

DLL_Status(Terminal)

Retorno: Tipo Integer.

Retorna o Byte de status do Terminal:

bit 0 e 1:

00 teclado qwerty

01 teclado numérico

10 teclado 16

11 reservado

bit 2: reservado

bit 3: cts 1 off-line 0 on-line

bit 4: reservado

bit 5: busy 1 off-line 0 on-line

bit 6: reservado

bit 7: reservado

A *DLL* disponibilizada pelo fabricante do Shellby 915, contem as funções necessárias para interagir com o sistema, sendo assim a codificação para recebimento e envio das informações do terminal foi realizada através de seu uso.

Declaração das funções em Delphi:

- function ConfigLpt(Endereco, Timeout: Word): Boolean; stdcall; external 'WTechLpt.DLL';
- procedure DLL_Close(); stdcall; external 'WTechLpt.DLL';
- procedure DLL_PosCur(Terminal, Lin, Col: Byte); stdcall; external 'WTechLpt.DLL';
- procedure DLL_Clear(Terminal: Byte); stdcall; external 'WTechLpt.DLL';
- procedure DLL_Echo(Terminal: Byte; Dado: Char); stdcall; external 'WTechLpt.DLL';
- procedure DLL_Display(Terminal: Byte; Dado: string); stdcall; external 'WTechLpt.DLL';
- procedure DLL_Aciona(Terminal, Dado: Byte); stdcall; external 'WTechLpt.DLL';
- function DLL_Get(Terminal: Byte): Char; stdcall; external 'WTechLpt.DLL';
- function DLL_Status(Terminal: Byte): Byte; stdcall; external 'WTechLpt.DLL';
- function DLL_Print(Terminal: Byte; Dado: Char): Byte; stdcall; external 'WTechLpt.DLL';
- function DLL_Serial(Terminal: Byte; Dado: Char): Byte; stdcall; external 'WTechLpt.DLL';

Também é utilizada uma rotina chamada: Rotina TrataTerminais(). Onde é feito todo o processamento, importante para o tratamento de uso de vários terminais no estabelecimento, ou seja, um terminal pode estar fazendo uma venda, outro enviando dados para a base, enfim, situações diferentes. Para controlar de maneira eficiente, sempre que o sistema for acionado por um terminal, a função TrataTerminais() será chamada e executada conforme o estado em que se encontra aquele terminal em especifico. [11]

# APÊNDICE B – Figuras com os códigos fontes da rotinas do sistema em Delphi 7

ufCaixa.pas

ufCaixa | uPDVR

```
//Funções do Shellby
procedure Gdisplay(Terminal:byte; str:string );
begin
//Escreve a string correspondente no terminal selecionado
     Dll_Display(Terminal,str);
end;

procedure Gecho(Terminal:byte; str:char );
// Escreve o caracter correspondente no terminal selecionado
begin
     Dll_Echo(Terminal,str);
end;

procedure GPosCursor(Terminal:byte; linha:integer; coluna:integer );
//Posiciona o cursor na linha e coluna desejadas
// linha 0 ou 1, coluna 0 à 39
begin
     Dll_PosCur(Terminal,Linha, coluna);
end;

procedure Gclear(Terminal:byte);
//Limpa o display do terminal
begin
     Dll_Clear(Terminal);
end;
```

328: 105 | Modified | Insert | Code | Diagram

ufCaixa.pas

```
2: begin
  Gdisplay(Terminal,#7);
  sPedido[terminal]:=''; // envia dados para o display do terminal e muda o estado para pegar o retorno.
  GClear(Terminal);         // neste ponto já deve ter observado que no estado PAR enviamos dados para o terminal
  Gdisplay(Terminal,'Comanda/Mesa:');  // e no estado IMPAR pegamos o retorno.
  Gget(Terminal,'999' );
  estado[Terminal]:=3;
  nItens[Terminal]:=0;
end;

3: begin
  if GtrTecla(Terminal) then begin
    If (teclado[Terminal] = '') or (teclado[Terminal] = #127) Then
      estado[Terminal]:= 0
    else begin                          // Apenas para exibir no programa os dados
      estado[Terminal]:=4;              // coletados
      Comanda[Terminal]:=teclado[Terminal];
      if PDVR.MesaFechada(StrToInt(Teclado[Terminal])) then begin
        AbreMesaShellby(StrToInt(Teclado[Terminal]),PDVR.NroPadraoOcupantesMesa);
        if pgControl.ActivePageIndex=1 then
          AtualizaMesas;
      end
      else if PDVR.MesaAberta(StrToInt(Teclado[Terminal])) then
        SelecionaMesaShellby(StrToInt(Teclado[Terminal]))
      else begin
        Gdisplay(Terminal,#7);
        GClear(Terminal);
        Gdisplay(Terminal,'Mesa aguardando');
        GPosCursor(Terminal,1,0);
        Gdisplay(Terminal,'pagamento...');
        Gget(terminal,'X');       // novamente prendemos a palicação em um estado morto
        estado[Terminal]:= 98;    // apenas para o usuario visualizar a msg.
        proximo_estado[terminal]:=2;
      end;
```

ufCaixa.pas

```
procedure TfCaixa.tmParalelaTimer(Sender: TObject);
//Imprime no display o dado requerido de acordo com o estado do terminal
//correspondente.
var
  Terminal:integer;
  Reg: TRegistry;
begin
  //cria a conexão com o regedit (Caixa)
  Reg:= TRegistry.Create;
  Reg.RootKey:= HKEY_LOCAL_MACHINE;
  Reg.OpenKey('Software\Microtab\TabPDV\Caixa',False);

  for Terminal:=0 to NTerminal - 1 do begin
    GFlushCom(Terminal); //chama o descarregador de impressão serial
    GFlushPrn(Terminal); //chama o descarregador de impressão paralela

    case estado[Terminal] of // Envia dados para o display do terminal
      0:begin             // muda o estado do terminal para pegar o retorno
        GClear(Terminal);
        Gdisplay(Terminal,'Usuario:');
        Gget(Terminal, '999' );           //  apenas numeros
        estado[Terminal]:=1;
      end;

      1:begin                           // Pega retorno do estado 0
                                        // GtrTecla() Será True somente quando a mascara de entrada se completar  ou teclar EN

        If GtrTecla(Terminal) Then begin
          If (teclado[Terminal] = '') or (teclado[Terminal] = #127) Then // Se teclar ENTER ou DEL volta estado
            estado[Terminal] := 0                                        // eficiente para montar novamente a tela quando o equi
          else begin
            estado[Terminal] := 2;
            usuarioSh[terminal]:= teclado[terminal];
          end;
```

ufCaixa.pas

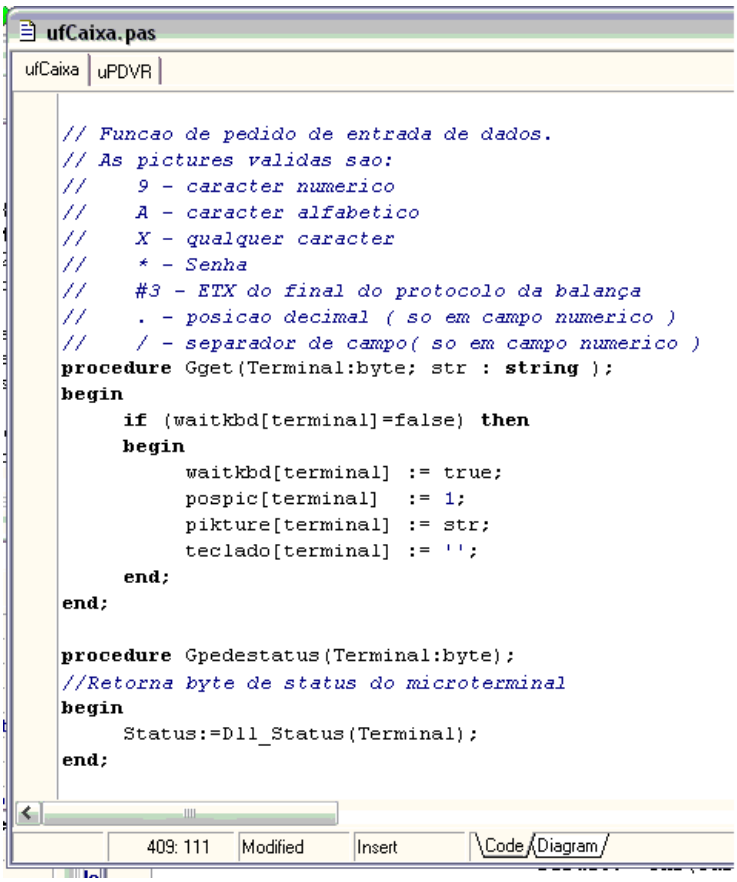
ufCaixa | uPDVR

```
procedure Ginicia;
//Inicializa funcionamento do microterminal
var
   i:integer;

begin
     //Configura porta e timeOut
   if  ConfigLpt(1,1) then
   begin
          for i:=0 to NTerminal -1 do
          begin
              pFlag_Serial[i]:=0;
              estado[i]:=0;
              pikture[i]:='';
              pospic[i]:=1;
              waitkbd[i]:=false;
              pinpos[i]:=1;
              pprinter[i]:='';
              poutpos[i]:=1;
              sprinter[i]:='';
              sinpos[i]:=1;
              soutpos[i]:=1;
              Gclear(i);
              Gclear(i);
          end;
          flag_iniciou:= true;
          k:=1;
   end
        else
              ShowMessage('Wtechlpt.dll retornou false'#13'Veja configuração de arquivo INI'#13'Caso comutadora paralela teste funcionamento!')
end;
```

ufCaixa.pas

ufCaixa | uPDVR

```
// Funcao de tratamento do teclado do terminal.
// Pede tecla e faz a consistencia com a picture
// solicitada na funcao GGET, aceitando ou rejeitando
// o que for digitado no terminal.
// Tambem faz o tratamento do // DEL//  do terminal,
// limpando o campo quando tem algum caracter digitado
// ou retornando ESC quando nao tinha nada digitado.
// O retorno desta funcao sera:
//          false - quando a picture solicitada no
//                  ultimo gget nao tiver sido digitada.
//          true - quando a picture tiver sido digitada
//                 completamente ou o CR foi digitado antes do final.
function GtrTecla(Terminal:byte):boolean;
//Trata tecla que foi pressionada
var
   gPict  :string;
   stecla : char;
   tamanho:integer;
begin
   if  (WaitKbd[terminal] = false) then
       begin
            result := false;
            exit;
       end;

   stecla:= Gpedetecla(Terminal);
   if  (sTecla = Chr(13)) then //tecla enter
       begin
            WaitKbd[terminal] := false;
            result := true;
            exit;
       end;
   if sTecla = Chr(127) then //tecla Del
      begin
        if PosPic[terminal] = 1 then
        begin
                Teclado[Terminal]:= STecla;
                WaitKbd[terminal] := false;
                result:= true;
        end
```

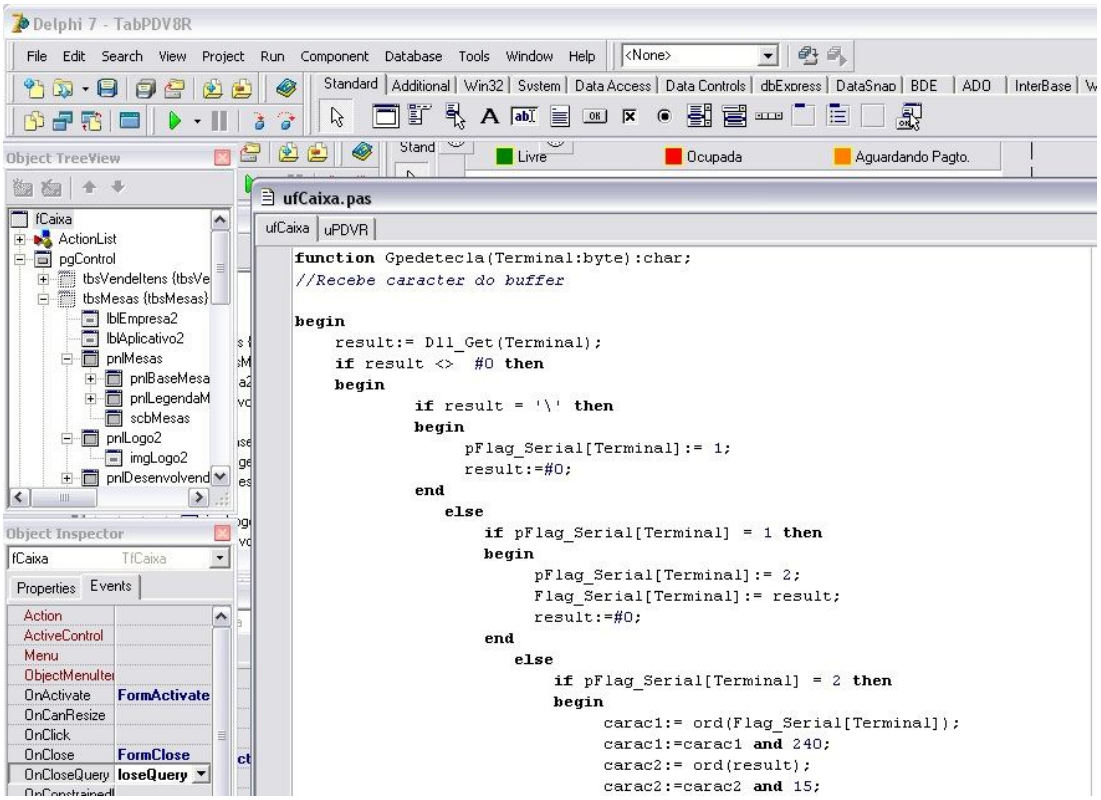


ufCaixa.pas

ufCaixa | uPDVR

```
// Funcao de pedido de entrada de dados.
// As pictures validas sao:
//    9 - caracter numerico
//    A - caracter alfabetico
//    X - qualquer caracter
//    * - Senha
//    #3 - ETX do final do protocolo da balança
//    . - posicao decimal ( so em campo numerico )
//    / - separador de campo( so em campo numerico )
procedure Gget(Terminal:byte; str : string );
begin
     if (waitkbd[terminal]=false) then
     begin
          waitkbd[terminal] := true;
          pospic[terminal]  := 1;
          pikture[terminal] := str;
          teclado[terminal] := '';
     end;
end;

procedure Gpedestatus(Terminal:byte);
//Retorna byte de status do microterminal
begin
     Status:=Dll_Status(Terminal);
end;
```

409: 111 | Modified | Insert | Code | Diagram



Delphi 7 - TabPDV8R

File Edit Search View Project Run Component Database Tools Window Help <None>

Standard | Additional | Win32 | System | Data Access | Data Controls | dbExpress | DataSnap | BDE | ADO | InterBase

Object TreeView

fCaixa
ActionList
pgControl
tbsVendeItens (tbsVe
tbsMesas (tbsMesas)
lblEmpresa2
lblAplicativo2
pnlMesas
pnlBaseMesa
pnlLegendaM
scbMesas
pnlLogo2
imgLogo2
pnlDesenvolvend

Object Inspector

fCaixa TfCaixa

Properties | Events

Action
ActiveControl
Menu
ObjectMenuIte
OnActivate FormActivate
OnCanResize
OnClick
OnClose FormClose
OnCloseQuery loseQuery

Stand Livre Ocupada Aguardando Pagto.

ufCaixa.pas

ufCaixa | uPDVR

```
function Gpedetecla(Terminal:byte):char;
//Recebe caracter do buffer

begin
    result:= Dll_Get(Terminal);
    if result <>  #0 then
    begin
            if result = '\' then
            begin
                 pFlag_Serial[Terminal]:= 1;
                 result:=#0;
            end
               else
                  if pFlag_Serial[Terminal] = 1 then
                  begin
                       pFlag_Serial[Terminal]:= 2;
                       Flag_Serial[Terminal]:= result;
                       result:=#0;
                  end
                     else
                         if pFlag_Serial[Terminal] = 2 then
                         begin
                              carac1:= ord(Flag_Serial[Terminal]);
                              carac1:=carac1 and 240;
                              carac2:= ord(result);
                              carac2:=carac2 and 15;
```

ufCaixa.pas

ufCaixa | uPDVR

```
    flag_iniciou: boolean;
    x,k:integer; //utilizados com terminal configurado como flag de serial ativado
    Status:Byte; //status do terminal

    teclado:array[0..NTerminal-1] of string; //buffer de teclado
    waitkbd:array[0..NTerminal-1] of boolean; //flag de espera de digitacao
    pikture:array[0..NTerminal-1] of string; //picture esperada
    pospic :array[0..NTerminal-1] of integer; //posicao dentro da picture que devemos receber
    estado :array[0..NTerminal-1] of integer; //estado de execucao do programa
    Proximo_estado :array[0..NTerminal-1] of integer; //estado de execucao do programa
    balanca: array[0..NTerminal-1] of integer; //contador para pedido de peso de balança
    pprinter:array[0..NTerminal-1] of string; //buffer de impressora paralela
    sprinter:array[0..NTerminal-1] of string; //buffer de impressora serial
    pinpos:array[0..NTerminal-1] of integer; //posicao de entrada do buffer de paralela
    poutpos:array[0..NTerminal-1] of integer; //posicao de saida do buffer de paralela
    sinpos:array[0..NTerminal-1] of integer; //posicao de entrada do buffer de serial
    soutpos:array[0..NTerminal-1] of integer; //posicao de saida do buffer de serial
    Delai:array[0..NTerminal-1] of byte;  // delai exibe msg e volta o estado
    Flag_Serial: array[0..NTerminal-1] of char;
    pFlag_Serial: array[0..NTerminal-1] of byte;
    carac1,carac2:byte;

    //variáveis da aplicação
    usuarioSh:array[0..NTerminal-1] of string;
    Comanda:array[0..NTerminal-1] of string;
    produtoSh:array[0..NTerminal-1] of String;
    quantidade:array[0..NTerminal-1] of string;
    codigo:array[0..NTerminal-1] of string;
    nItens: array[0..NTerminal-1] of byte;
    Ind_Produto:array[0..NTerminal-1] of Integer;
    sPedido:array[0..Nterminal-1] of string;
```

328: 105 | Modified | Insert | Code | Diagram

ufCaixa

```
      8: begin
        GClear(Terminal);
        GPosCursor(terminal,5, 6);
        Gdisplay (terminal,'Confirma?');
        GPosCursor(terminal,6, 4);
        Gdisplay (terminal,'[1]Nao [2]Sim ');
        GPosCursor(terminal,0, 0);
        Gdisplay (terminal,quantidade[Terminal] + ' x ' + produtoSh[Terminal]);
        GGet(Terminal,'X');
        estado[Terminal]:=9;
      end;

      9: begin
        if GtrTecla(Terminal) then begin
          if (Teclado[Terminal] = #127) or (Teclado[Terminal] = '1') or (Teclado[Terminal] = 'N') then
            estado[Terminal]:=4
          else if (Teclado[Terminal] = '') or(Teclado[Terminal] = '2') or (Teclado[Terminal] = 'S') then begin
            VendeItemShellby(StrToFloat(quantidade[Terminal]), codigo[Terminal]);
            estado[Terminal]:=4;
          end
          else
            estado[Terminal]:=8;
        end;
      end;

      98:
        if GtrTecla(Terminal) then begin
          if Teclado[terminal] =  '2' then // Estamos simulando um ENTER para cancelar o pedido
            estado[terminal]:=0
          else
            estado[Terminal]:=proximo_estado[terminal];
      end;

      99: if GtrTecla(Terminal) then begin
        GClear(Terminal);
        estado[Terminal]:=proximo_estado[terminal];
      end;

    end; //fim Case
```