

**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ  
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE  
SISTEMAS**

**CAMILA NAGANO  
NEWTON MARTINS DE ALMEIDA JUNIOR**

**SISTEMA PARA GERENCIAMENTO DE RESTAURANTES DE PEQUENO PORTE**

**PATO BRANCO - PR  
2013**

**CAMILA NAGANO**  
**NEWTON MARTINS DE ALMEIDA JUNIOR**

**SISTEMA PARA GERENCIAMENTO DE RESTAURANTES DE PEQUENO PORTE**

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientadora: **Profa. MSc Rúbia E. O. Schultz Ascari**

**PATO BRANCO - PR**

**2013**

ATA Nº: 206

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DOS ALUNOS NEWTON MARTINS DE ALMEIDA JUNIOR e CAMILA NAGANO.

Às 15:30 hrs do dia 17 de abril de 2013, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Rúbia E. O. Schultz Ascari (Orientadora), Eliane Maria de Bortoli Fávero (Convidada) e Silvio Luiz Bragatto Boss (Convidado), para avaliar o Trabalho de Diplomação do aluno Newton Martins De Almeida Junior, matrícula 1147978 e da aluna Camila Nagano, matrícula 1172328, sob o título **Sistema de Gerenciamento para Restaurantes de Pequeno Porte**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, COADS. Após a apresentação os candidatos foram entrevistados pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 16:45 hrs foi encerrada a sessão.



Prof. Rúbia E. O. Schultz Ascari, M.Sc.  
Orientadora



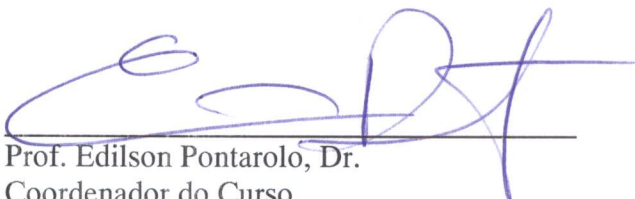
Prof. Eliane Maria de Bortoli Fávero, M.Sc.  
Convidada



Prof. Silvio Luiz Bragatto Boss, M.Sc.  
Convidado



Prof. Omero Francisco Bertol, M.Sc.  
Coordenador do Trabalho de Diplomação



Prof. Edilson Pontarolo, Dr.  
Coordenador do Curso

## **AGRADECIMENTOS**

Gostaríamos de agradecer a todas as pessoas que fizeram parte dessa trajetória, desde o início do curso até o desenvolvimento da monografia.

Agradecemos aos nossos Pais que sempre nos incentivaram a estudar, e fizeram todo o possível para que tivéssemos condições pra isso, agradecemos também por nos corrigirem quando necessário, e por nos darem a base que temos hoje para sermos cidadãos de caráter e batalhadores. Tudo que conquistamos devemos muito a eles, pois uma das coisas mais importantes que nos deram foi lutar pelos nossos objetivos independentemente das dificuldades, pois não adianta estalar os dedos e esperar que as coisas aconteçam por si sós.

Também agradecemos as nossas famílias em geral, amigos e companheiros por nos incentivarem, auxiliarem quando precisávamos e principalmente por entender os momentos de ausência por morar em outra cidade, trabalhos e estudos.

Aos professores e colegas de trabalho e faculdade um muito obrigado por nos ensinarem e enriquecerem nosso conhecimento, além do incentivo para chegar até o final do curso. Podemos destacar também a empresa Gentus e a universidade por nos fornecerem a base para o nosso conhecimento, que foram vitais para o desenvolvimento do trabalho final.

Um agradecimento em especial a professora orientadora, Rúbia Eliza de Oliveira Schultz Ascari, que sempre foi muito atenciosa, verificando cada detalhe do trabalho, dando dicas e mostrando-se sempre disposta a ajudar.

## RESUMO

NAGANO, Camila e JUNIOR, Newton Martins de Almeida. **Sistema de Gerenciamento de Restaurantes de Pequeno Porte**. 2013. 60 f. Monografia de Trabalho de Conclusão de Curso. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná, *Campus Pato Branco*. Pato Branco, 2013.

Com o crescimento do ramo gastronômico tornou-se comum encontrar diversos restaurantes por onde se anda. E cada vez mais empresários estão apostando nessa área. Porém pequenos restaurantes muitas vezes não possuem uma ferramenta de auxílio para controlar os atendimentos, registros de compras, e de vendas, ou seja, não fazem o uso de um sistema de informação. Em decorrência disso, muitos estabelecimentos fazem o controle anotando em cadernos, papéis, fichas, realizando cálculos na calculadora. Desta forma podem-se perder dados importantes, gerando prejuízo à empresa ou até descontentamento por parte dos clientes. Mediante a necessidade de controlar os processos realizados por empresas deste segmento, verificou-se a possibilidade de modelar e desenvolver um sistema específico para pequenos restaurantes, visando agilizar e melhor organizar seus processos, evitando os problemas citados anteriormente. O sistema é composto por rotinas de cadastros, lançamentos e relatórios diversos. Foi modelado com base nos conceitos da análise orientada a objetos e estruturada para modelagem de dados e desenvolvido utilizando o ambiente de desenvolvimento Delphi, banco de dados Firebird e o gerador de relatórios FastReport.

Palavras-Chave: Restaurantes de pequeno porte. Desenvolvimento de *Software*. Gerenciamento de restaurantes.

## ABSTRACT

NAGANO, Camila e JUNIOR, Newton Martins de Almeida. **Sistema de Gerenciamento de Restaurantes de Pequeno Porte**. 2013. 60 f. Monografia de Trabalho de Conclusão de Curso. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná, *Campus Pato Branco*. Pato Branco, 2013.

With the growth of the branch gastronomic became common to find many restaurants wherever you go, and there is a lot of business owners investing in this area. But small restaurants often do not own a tool to help to control the calls, records of purchases and sales, in other words, do not make use of an information system. As a result, many establishments do control jotting of booklets, papers, records, performing calculations on the calculator, this way important dat may be lost generating damage to the company or even dissatisfaction by customers. The need to control the processes performed by companies in this segment, it is possible to model and to develop a system to small restaurants, in order to speed and better organize their processes, avoiding the problems mentioned previously. This system consists of routines registrations, releases and other reports. Has been modeled based on the concepts of object-oriented analysis and data modeling for structured and developed using the development environment Delphi, Firebird database and report generator FastReport.

Keywords: Small business restaurants. Software Development. Management of restaurants.

## LISTA DE FIGURAS

Figura 1 - Documentação: Diagrama de Casos de Uso .....	32
Figura 2 - Documentação: Diagrama Conceitual.....	33
Figura 3 - Documentação: Diagrama de Sequência (Cadastrar Pedido) .....	36
Figura 4 - Documentação: Diagrama de Comunicação - Cadastrar Pedido.....	37
Figura 5 - Documentação: Diagrama de Entidade e Relacionamento.....	38
Figura 6 - Tela: Apresentação Inicial do Sistema .....	40
Figura 7 - Tela: Formulário de Cadastro de Pedidos.....	41
Figura 8 - Tela: Formulário de Cadastro de Produtos .....	42
Figura 9 - Tela: Formulário de Cadastro de Despesas.....	43
Figura 10 - Tela: Formulário de Cadastro de Usuários .....	44
Figura 11- Tela: Recibo Simples de um Pedido .....	44
Figura 12 - Tela: Relatório de Pedidos.....	45

## LISTA DE QUADROS

Quadro 1 - Código: Formulário Padrão .....	49
Quadro 2 - Código: Formulário de Acessos (Alternância de Telas).....	49
Quadro 3 - Código: Formulário Cadastro de Pedidos .....	53
Quadro 4 - Código: Calculadora.....	57
Quadro 5 - Código: Atualização da Lista de Produtos .....	57



## LISTA DE TABELAS

Tabela 1 - Requisitos Funcionais do Sistema .....	30
Tabela 2 - Requisitos Não Funcionais do Sistema .....	31
Tabela 3 - Caso de Uso Expandido (Cadastrar Pedido) .....	33
Tabela 4 - Descrição da Classe Produto .....	34
Tabela 5 - Descrição da Classe Usuario .....	34
Tabela 6 - Descrição da Classe TypeUser .....	34
Tabela 7 - Descrição da Classe Despesas .....	34
Tabela 8 - Descrição da Classe Pedido .....	35
Tabela 9 - Descrição da Classe Receitas .....	35
Tabela 10 - Descrição da Classe ItensPedido.....	35
Tabela 11- Descrição da Tabela Produto (DER) .....	38
Tabela 12 - Descrição da Tabela Pedido (DER) .....	39
Tabela 13 - Descrição da Tabela ItensPedido (DER).....	39
Tabela 14 - Descrição da Tabela Usuario (DER) .....	39
Tabela 15 - Descrição da Tabela Despesas (DER).....	39
Tabela 16 - Descrição da Tabela Receitas (DER).....	40

## LISTA DE ABREVIATURAS

<b>CASE</b>	<i>Computer Aided Software Engineering</i>
<b>DDL</b>	<i>Data Definition Language</i>
<b>DER</b>	Diagrama de Entidade e Relacionamento
<b>HTML</b>	<i>Hyper Text Markup Language</i>
<b>IDE</b>	<i>Integrated Development Environment</i>
<b>SGBD</b>	Sistema de Gerenciamento de Banco de Dados
<b>SI</b>	Sistema de Informação
<b>SysML</b>	<i>Systems Modeling Language</i>
<b>UML</b>	<i>Unified Modeling Language</i>
<b>UMFs</b>	Funções Definidas por Usuários
<b>OOP</b>	<i>Oriented Object Programming</i>
<b>RFT</b>	<i>Rich Text Format</i>
<b>XML</b>	<i>eXtensible Markup Language</i>

## SUMÁRIO

LISTA DE FIGURAS .....	6
LISTA DE QUADROS .....	7
LISTA DE TABELAS .....	8
LISTA DE ABREVIATURAS.....	9
1 INTRODUÇÃO .....	11
1.1 Objetivos .....	12
1.1.1 Objetivo Geral .....	12
1.1.2 Objetivos Específicos .....	12
1.2 Justificativa.....	13
1.3 Estrutura do Trabalho .....	13
2 REFERENCIAL TEÓRICO .....	15
2.1 Gestão de restaurantes.....	15
2.2 Análise de Requisitos.....	16
2.3 Análise estruturada .....	17
2.4 UML .....	18
2.5 Orientação a Objetos .....	20
3 MATERIAIS E MÉTODOS.....	22
3.1 Materiais.....	22
3.1.1 Visual Paradigm .....	22
3.1.2 Case Studio.....	23
3.1.3 Firebird .....	23
3.1.4 IBExpert .....	24
3.1.5 <i>Delphi</i> .....	25
3.1.6 <i>FastReport</i> .....	26
3.2 Método .....	26
4 RESULTADOS E DISCUSSÕES .....	28
4.1 Modelagem do Sistema .....	28
4.2 Requisitos de usuário para o Sistema.....	29
4.2.1 Requisitos Funcionais e Não-Funcionais .....	29
4.2.2 Diagrama de Caso de Uso .....	31
4.2.3 Modelo Conceitual.....	33
4.2.4 Diagrama de Sequência.....	35
4.2.5 Diagrama de Comunicação.....	36
4.2.6 Diagrama de Entidade e Relacionamento.....	37
4.3 Apresentação do Sistema .....	40
4.4 Implementação do Sistema.....	45
5 CONCLUSÃO.....	58
REFERÊNCIAS BIBLIOGRÁFICAS .....	60

## 1 INTRODUÇÃO

O grande crescimento da rede de gastronomia no Brasil é evidente, e pode ser observado com o aumento de estabelecimentos de diferentes características em diversas regiões do mundo. Um exemplo básico que pode ser citado é que ao caminhar em uma rua pode-se deparar com um restaurante árabe na esquina, do outro lado da rua um restaurante italiano e até um restaurante japonês um pouco mais a frente. Esse crescimento do mundo gastronômico provoca curiosidades e proporciona novas experiências aos consumidores. Em função disso tornou-se um mercado em ascensão. Em reportagem da revista EXAME escrita por Gianini (2011), o negócio movimenta no país, aproximadamente, R\$ 180 milhões por ano, empregando 6 milhões de pessoas. Como consequência, muitos empresários estão investindo nessa área. Contudo, muitos negócios são gerenciados de forma trabalhosa, ou seja, controles são feitos através de papéis, cadernos, calculadora, planilhas, entre outros. Isso ocorre porque não utilizam um sistema de informação (SI) para tais finalidades.

Segundo O'Brien e Marakas (2010) "um sistema de informação (SI) pode ser qualquer combinação organizada de pessoas, *hardware*, *software*, redes de comunicação, recursos de dados e políticas de procedimentos que armazenam, restauram, transformam e disseminam informações em uma organização."

A não utilização de um sistema para gerenciamento dos restaurantes é decorrente de diversos motivos como: gastos com a aquisição/manutenção do sistema e treinamentos para funcionários; proprietários nunca terem pensado no assunto ou nunca ninguém ter proposto uma ideia; proprietários ou funcionários serem reservados ao novo e preferirem manter do jeito que está, pois funciona. Porém, os benefícios que o uso de um sistema específico pode gerar posteriormente são recompensadores e farão com que o negócio seja mais próspero. Para Oliveira (2001) um sistema de informação eficiente pode ter um grande impacto na estratégia corporativa e no sucesso da empresa. Este impacto pode beneficiar a empresa, os clientes ou usuários e qualquer indivíduo ou grupo que interagir com os sistemas de informação.

Assim, verificou-se a possibilidade de desenvolver como trabalho de conclusão de curso um sistema de informação que permita o gerenciamento de

restaurantes de pequeno porte. Para o desenvolvimento do mesmo foi utilizado a ambiente de desenvolvimento Delphi, que possui componentes integrados para conexão com o banco de dados Firebird e o gerador de relatórios *FastReport*.

## **1.1 Objetivos**

### **1.1.1 Objetivo Geral**

Desenvolver um sistema computacional utilizando o ambiente de desenvolvimento Delphi e banco de dados Firebird para o gerenciamento de restaurantes de pequeno porte.

### **1.1.2 Objetivos Específicos**

Para atender o objetivo geral deste trabalho, pretende-se:

- Fazer o levantamento de requisitos e análise de dados através da verificação das necessidades de empresários que atuam em restaurantes de pequeno porte;
- Criar Diagramas da UML (*Unified Modeling Language*), como de Caso de Uso, de Sequência e de Comunicação para modelar os processos que serão implementados pelo software e descobrir as classes que irão gerar o diagrama de classes para o projeto;
- Criar Diagrama de Entidade e Relacionamento para representar o esquema de dados que serão manipulados pelo software;
- Desenvolver um software de gerenciamento de restaurantes, utilizando o ambiente de desenvolvimento Delphi e banco de dados Firebird. Serão criados cadastros de produtos, despesas, receitas, usuários, pedidos e pagamentos, além de relatórios diversos para consulta dos dados cadastrados.

## 1.2 Justificativa

Este trabalho foi desenvolvido com o objetivo de agilizar e melhor organizar os processos de estabelecimentos de pequeno porte que atuam no ramo da gastronomia por meio de um sistema de informação. Esse sistema é responsável por gerenciar todas as informações financeiras do estabelecimento, bem como o cadastro de produtos, pedidos e usuários.

Durante o desenvolvimento deste sistema, foi colocado em prática todo o conjunto de conhecimentos adquiridos ao longo do curso de graduação, desde a análise de dados e documentação de sistemas até a implementação de um *software* e banco de dados bem estruturados.

Para tanto, foram utilizados o ambiente de desenvolvimento Delphi 7, pois a linguagem empregada é de fácil entendimento e com uma sintaxe simples. O Delphi possui um ambiente gráfico que permite desenvolver rapidamente e de forma bem clara um *software* de alta performance operacional, motivo esse que o faz ser utilizado comercialmente em larga escala. Quanto ao banco de dados, foi optado pela utilização do Firebird, que é gratuito e não deixa a desejar em nenhum aspecto para o projeto em questão. O ambiente gráfico utilizado para o desenvolvimento do banco de dados é o IBExpert.

## 1.3 Estrutura do Trabalho

A estrutura deste trabalho foi dividida em 5 capítulos principais. O primeiro capítulo é composto pela justificativa, os objetivos geral e específicos, e a estrutura do trabalho.

No capítulo 2 é apresentado o referencial teórico que compreende conceitos relacionados a gestão de restaurantes, ou seja, a regra do negócio, o conceito de levantamento de requisitos e análise estruturada, a UML que foi utilizada para modelagem do sistema, e por fim orientação a objetos explicando o conceito utilizado para desenvolver o *software*.

No capítulo 3 são apresentados os materiais e o método utilizado para documentação, modelagem e implementação do *software*.

No capítulo 4 apresentam-se os resultados obtidos com o trabalho, detalhando a modelagem do *software* e descrição dos processos principais do sistema desenvolvido.

Por fim está o capítulo 5 com a conclusão deste trabalho.

## 2 REFERENCIAL TEÓRICO

O desenvolvimento deste trabalho foi dividido em duas partes. A primeira tratou de realizar pesquisa sobre as regras de negócio de restaurantes de pequeno porte. Com isso foi possível analisar as vantagens em utilizar um *software* para gerenciar este tipo de negócio, juntando as experiências de empresários do ramo e observações dos acadêmicos em diferentes estabelecimentos. A segunda parte correspondeu ao desenvolvimento de um sistema para gerenciamento de pequenos restaurantes utilizando práticas de modelagem pela UML e codificação orientada a objetos.

Esse capítulo explicará o motivo da utilização de cada prática empregada nas duas partes citadas, para posteriormente nos demais capítulos descrever a aplicação destas.

### 2.1 Gestão de restaurantes

De acordo com Cadornin (2010):

“Pequenos restaurantes tem uma maneira de trabalhar bastante diferente dos demais comércios, principalmente por serem empresas de pequeno porte. Devido ao grande fluxo de pessoas, vendas de baixo valor e sem obrigatoriedade de registro em nota fiscal, essas empresas apresentam a necessidade de trabalhar com sistemas informatizados, mas sistemas pequenos, de uso rápido e fácil. A utilização de um *software* nestas empresas tem como principais objetivos controlar os registros financeiros, organizar, agilizar os processos e os pedidos de venda”. (CADORNIN, 2010).

Empresários deste segmento têm uma grande dificuldade em gerenciar a intensa movimentação de produtos e fluxo de clientes. Em função disso, a informatização é de extrema importância para o sucesso destas empresas, pois permite que os processos sejam controlados de forma organizada, evitando descontrolar os dados financeiros, dos pedidos, e proporcionando melhor atendimento aos clientes.

Segundo Maricato (2005):

“Se a casa estiver cheia, é importante apressar o fluxo de serviços: agilizar os pedidos, o serviço e a elaboração da conta, no sentido de obter maior rotatividade das mesas. Obviamente, isso deve ser feito com elegância, procurando servir melhor o cliente, que preferivelmente não deve perceber que se quer apressar sua saída.” (MARICATO, 2005, p.97).



Há casos em que o estabelecimento encontra-se com grande fluxo de clientes, e em paralelo estão os fluxos de informações. E mesmo o gerente sendo capaz, o auxílio para controlar os processos é indispensável. Assim, percebe-se a importância de se informatizar os processos, garantindo a segurança das informações e fazendo com que o trabalho fique menos pesado. E para ajudar o gestor existe o sistema de informação.

Segundo O'Brien e Marakas (2010) há três razões fundamentais comuns a todas as aplicações empresariais da tecnologia da informação, as quais são encontradas nos três papéis vitais que os sistemas de informação podem exercer em uma empresa:

- Suporte de processos e operações de negócios;
- Suporte da tomada de decisão pelos seus empregados e gerentes;
- Suporte das suas estratégias para vantagem competitiva.

Sabendo disso, a empresa estará melhor preparada para atender adequadamente seus clientes, e de maneira organizada possuir um controle interno eficiente e com menos falhas.

## **2.2 Análise de Requisitos**

Wazlawick (2011, p.23) afirma que “deve ficar claro para o analista de requisitos são coisas que o cliente ou usuário solicita, e não coisas que ele, como analista, planeja. Alguns analistas confundem o registro dos requisitos, que são a memória das solicitações do cliente, com o início do projeto do sistema”. Sendo assim o mais importante nesse processo é entender o que o cliente realmente necessita e como seria ideal implementar o sistema para ele e somente após isso, iniciar o desenvolvimento.

Os requisitos são classificados em funcionais e não-funcionais. Os requisitos funcionais consistem nas funções de cada requisito que devem estar presentes no sistema, ou seja, são as ações que ele irá executar, “que podem ser uma entrada (exemplo, cadastrar comprador) ou saída (exemplo, emitir relatório de vendas no mês)”(WAZLAWICK, 2011, p.25).

Os requisitos não-funcionais tem o propósito de descrever as qualidades requeridas para um sistema, como sua usabilidade e características de desempenho. Eles completam a documentação de requisitos funcionais que descrevem o comportamento do sistema e podem ser basicamente de dois tipos: restrições lógicas “(exemplo: no registro de uma venda, não efetuar a venda caso a operadora de cartão não autorize)” (WAZLAWICK, 2011, p.25), ou restrições tecnológicas “(exemplo: no registro de venda, a autorização de débito no cartão de crédito não deve levar mais do que 5 segundos)” (WAZLAWICK, 2011, p.26). E por fim, os requisitos suplementares que são todos os tipos de restrições que se aplicam no sistema como um todo “(exemplo, estabelecer que o sistema deva ser implementado com uma determinada linguagem de programação)” (WAZLAWICK, 2011, p.25).

Os requisitos funcionais e não funcionais do sistema estão definidos na seção 4.2.1.

### **2.3 Análise estruturada**

Rezende (2005), alega que:

“a análise estruturada, tal como outras técnicas de desenvolvimento de software, têm sua própria funcionalidade para elaboração e diagramação de software. Os principais objetivos da análise estruturada são: reduzir os custos de manutenção; aumentar produtividade; gerar sistemas impessoais; e aumentar a legibilidade e a flexibilidade dos sistemas” (REZENDE, 2005, p.181).

O conceito fundamental da análise estruturada é a construção de um modelo lógico que busca transformar o esquema conceitual para informações técnicas que possam ser compreendidas e utilizadas por desenvolvedores e administradores de banco de dados. Este modelo estará ligado aos modelos suportados pelo SGBD (Sistema de Gerenciamento de Banco de Dados) utilizado. E segundo Costa (2006, p.6), “o modelo de entidade-relacionamento é o mais popular para o projeto de banco de dados”.

Dentre as representações gráficas utilizadas pela análise estruturada, está o diagrama de entidade-relacionamento (DER), utilizado neste trabalho para modelar o

banco de dados, mesmo utilizando os conceitos de orientação a objetos para modelar (diagramas da UML) e implementar o sistema.

Desta forma, os diagramas da UML gerados com base na análise orientada a objetos apresentam a estrutura estática e dinâmica do sistema com base em objetos. O diagrama de classes representa os dados de um sistema em termos de hierarquias de classes, enquanto o DER gerado com base na análise estruturada, é o mapeamento do modelo de classes para o modelo de entidade-relacionamento, o qual apresenta a estrutura do banco de dados

Para Rezende (2005):

“o principal propósito do DER é representar os objetos de dados e suas relações, sendo que cada entidade termina representada pelo menos por uma tabela de dados, e normalmente expressa um depósito de dados do DFD (Diagrama de fluxo de Dados)” (REZENDE, 2005, p.174).

Um diagrama de entidade e relacionamento é composto por: entidades, atributos e relacionamentos. As entidades representam tudo o que pode ser observado no mundo real com existência independente. Pode ser um objeto com existência física (ex. aluno), ou pode ser um objeto com existência conceitual (ex. curso). Atributos são as propriedades que descrevem uma entidade, no caso de um aluno, o nome, e no caso de um curso, o seu código. E os relacionamentos são as associações entre uma ou várias entidades. Por exemplo, um aluno se matricula em um curso, o ato de matricular é que relaciona o aluno ao curso.

## 2.4 UML

A UML é o acrônimo de *Unified Modeling Language* ou Linguagem de Modelagem Unificada. Booch, Rumbaugh e Jacobson (2005, p13) afirmam que “a UML é uma linguagem-padrão para a elaboração da estrutura de projetos de *software*. Ela poderá ser empregada para a visualização, a especificação, a construção e a documentação de artefatos que façam o uso de sistemas complexos de *software*”. Pode-se definir então, a UML como uma linguagem para analisar e modelar inicialmente um sistema, onde se registrará todos os processos do sistema através de diagramas. Com os diagramas é possível definir de uma forma mais clara como o fluxo do projeto funciona.

A UML descreve 13 tipos de diagramas oficiais e vem sendo constantemente revisada, dividindo-se em três famílias, conforme definidas na obra de Wazlawick (2011):

- Diagramas estruturais, compreendendo os diagramas de pacotes, classes, objetos, estrutura composta, componentes e distribuição;
- Diagramas comportamentais, compreendendo os diagramas de caso de uso, atividades e máquina de estados;
- Diagramas de interação, compreendendo os diagramas de comunicação, sequência, tempo e visão geral de integração.

Para Wazlawick (2011, p.4), “nem todos os diagramas precisam ser usados durante o desenvolvimento de um sistema. Usam-se apenas aqueles que possam apresentar alguma informação útil para o processo”. Desta forma, viu-se a necessidade da modelagem de quatro diagramas da UML: diagrama de classes, de casos de uso, de sequência e de comunicação. E por esta razão somente estes quatro diagramas serão explicados a seguir.

Segundo Fowler (2005):

“Um diagrama de classes descreve os tipos de objetos presentes no sistema e os vários tipos de relacionamentos estáticos existentes entre eles. Os diagramas de classes também mostram as propriedades e as operações de uma classe e as restrições que se aplicam à maneira como os objetos estão conectados. A UML utiliza a palavra característica como um termo geral que cobre as propriedades e operações de uma classe”. (FOWLER, 2005, p.52).

Para Furtado (2002):

“A modelagem de caso de uso é uma técnica utilizada para descrever a funcionalidade de um sistema através de atores que interagem. Atores representam um papel e iniciam o caso de uso que, por sua vez, deve entregar um valor tangível de retorno ao ator. Atores e casos de uso estão conectados através de associações”. (FURTADO, 2002, p. 54).

Segundo Martins (2010, p.157), o diagrama de sequência mostra a troca de mensagens entre as classes na realização de um caso de uso. É utilizado para mostrar como um cenário específico de um caso de uso será implementado. O diagrama de sequência dá ênfase em mostrar as trocas de mensagens no decorrer do tempo, que evolui da esquerda para a direita e de cima para baixo.

Chamado diagrama de colaboração em versões anteriores da UML, o diagrama de comunicação complementa o diagrama de sequência. Ele apresenta praticamente as mesmas informações, porém com um enfoque diferente, visto que

este diagrama não se preocupa com a temporalidade do processo, concentrando-se em como os objetos estão vinculados e quais mensagens são trocadas entre eles.

## 2.5 Orientação a Objetos

Segundo Brookshear (2003), a OOP (*Oriented Object Programming* - Programação Orientada a Objetos) é uma técnica de programação na qual o problema a ser abordado é modelado como sendo constituído por um conjunto de objetos relacionados que se comunicam enviando mensagens uns para os outros.

O sistema implementado como resultado deste trabalho foi desenvolvido baseado em programação orientada a objetos, abrangendo vários conceitos, como desenvolvimento de classes, encapsulamento, herança e polimorfismo das mesmas.

Uma classe é definida como um conjunto de objetos do mesmo tipo. Ela define o comportamento de seus objetos através de métodos e os estados possíveis destes objetos através de atributos. Assim, uma classe descreve os serviços providos por seus objetos e quais informações eles podem armazenar.

Um objeto é uma instância de uma classe. Ele representa de forma concreta, uma entidade, conceito ou abstração individual pertinente ao domínio do problema sob análise.

Para Melo (2010):

“Uma classe é a representação de um conjunto de objetos que compartilham a mesma estrutura de atributos, operações e relacionamentos, dentro de um mesmo contexto (semântica). Assim, uma classe especifica a estrutura de um objeto sem informar quais serão seus valores. Em contrapartida, um objeto corresponde à ocorrência (instância) de uma classe num determinado momento”. (MELO, 2010).

Melo (2010) afirma também que na concepção de modelagem de sistemas, um objeto é qualquer coisa existente no mundo real, em formato concreto ou abstrato, ou seja, exista fisicamente ou apenas conceitualmente.

Segundo Melo (2010, p.19), o encapsulamento se faz necessário uma vez que as classes tenham sido abstraídas de um determinado problema, assim, há a necessidade de proteger seus atributos e algumas operações, utilizando a interface da classe. A interface serve como intermediária entre a classe e o mundo externo, protegendo os usuários dessa classe de quaisquer alterações futuras.

Herança é o mecanismo que permite a uma classe herdar todo o comportamento e os atributos de outra classe, adquirindo imediatamente toda a funcionalidade de uma classe existente. Melo (2010) afirma que pela herança pode-se estabelecer relações entre classes, permitindo o compartilhamento de atributos e operações semelhantes. Assim, tem-se a flexibilidade de criar uma nova classe, incluindo somente as diferenças com relação à classe mais genérica.

Segundo Ramos (2006), polimorfismo:

“Trata-se da possibilidade de um objeto se manifestar sob diferentes formas. Polimorfismo refere-se à capacidade de uma mesma operação realizar funções diferentes de acordo com o objeto que a recebe e com os parâmetros que lhe são passados. Como um simples exemplo, imagine uma situação em que um cliente (objeto) tenha a necessidade de executar uma atividade (operação), como calcular uma dívida. Caso esse cliente queira calcular essa dívida no contexto comercial (calcular quanto ele deve a uma loja), o resultado e os dados de entrada serão diferentes ao calcular o quanto ele deve de aluguel de seu apartamento. A Operação é a mesma, porém, quando tratadas sob contextos diferentes, são obtidos outros resultados”. (RAMOS, 2006).

Com esses conceitos de orientação a objetos é possível maior facilidade de reutilização de códigos, pois uma vez escritos não precisarão ser reescritos novamente. Menor custo de desenvolvimento, devido à questão citada anteriormente, aumentando assim a produtividade e com custo de desenvolvimento cada vez menor. Torna mais fácil a manutenção do sistema, pois provê uma melhor organização do código.

## 3 MATERIAIS E MÉTODOS

Este capítulo apresenta as ferramentas utilizadas durante a análise, modelagem e desenvolvimento do sistema de gerenciamento de restaurantes de pequeno porte e o método aplicado para elaboração do trabalho.

### 3.1 Materiais

Primeiramente foi realizada a análise de requisitos a partir de observações em diversos estabelecimentos e conhecimento pelos acadêmicos terem ligação com pessoas do ramo. Em seguida foi modelado o sistema através da UML na ferramenta *Visual Paradigm*. O diagrama de entidade-relacionamento, representando a estrutura do banco de dados, foi feito na ferramenta *CaseStudio*.

Para o desenvolvimento do sistema foi utilizado o ambiente de desenvolvimento Delphi, que possui componentes integrados para conexão com o banco de dados *Firebird*, com interface *IBExpert* e o gerador de relatórios *FastReport*.

A seguir são detalhadas as ferramentas utilizadas no trabalho e como foram aplicadas na modelagem ou desenvolvimento do sistema.

#### 3.1.1 Visual Paradigm

*Visual Paradigm for UML* (VISUALPARADIGM, 2013) é uma ferramenta CASE<sup>1</sup> (*Computar Aided Software Engineering*) com várias opções de modelagem com diagramas da UML e que também oferece suporte a diagramas de requisitos SysML (*Systems Modeling Language*) e a diagramas DER.

A ferramenta possui um bom ambiente de trabalho, o que facilita a visualização e manipulação do projeto de modelagem. É uma ferramenta comercial e também oferece transformações específicas para códigos-fonte de algumas linguagens de programação como, por exemplo, C++ e Java.

---

<sup>1</sup> Uma ferramenta CASE é um software que auxilia no ciclo de desenvolvimento de um sistema, desde a fase de análise, fase de testes, desenvolvimento e apoiam na manutenção de metodologias.

Como citado anteriormente o *Visual Paradigm* é uma ferramenta comercial que possui várias versões, as quais variam de valor, porém há também uma versão gratuita que suporta todos os diagramas da UML, como o de classes, casos de uso, sequência e comunicação, elaborados neste trabalho.

Esta ferramenta é ideal para engenheiros de *software*, analistas de sistemas, e arquitetos de sistemas que estão interessados em criação de sistemas em larga escala e necessitam de confiabilidade no desenvolvimento orientado a objetos.

### 3.1.2 Case Studio

O Case Studio é uma ferramenta CASE que permite o desenvolvimento da modelagem do banco de dados para diversos gerenciadores de banco de dados. “CASE é uma técnica que visa facilitar o desenvolvimento de sistemas de software por meio do uso de ferramentas proporcionando mais qualidade e produtividade” (CASESTUDIO, 2013). CASE é uma sigla que traduzida significa ferramentas de suporte ao desenvolvimento de software.

Segundo Rezende (2005):

“As ferramentas de tecnologia CASE possuem facilidades gráficas para o planejamento e projeto de sistema. A bancada de trabalho coleta informações de projeto para acionar o gerenciador de códigos. Normalmente, quando fornecidos os componentes do Dicionário de Dados, essa ferramenta (*software*) também gera os respectivos diagramas pertinentes” (REZENDE, 2005, p181).

O Case Studio é utilizado para criar tabelas, campos, procedures, relacionamentos, entre outros objetos do banco de dados por meio de um ambiente visual. A partir da modelagem visual pode-se gerar o *script* para posteriormente executá-lo no SGBD específico.

### 3.1.3 Firebird

De acordo com Magno (2013)

"O *Firebird* é um sistema gerenciador de banco de dados de código fonte aberto. Diferente de vários outros bancos de dados *open source*, onde muitas vezes não encontramos alguns recursos considerados essenciais em um SGBD relacional, o Firebird conta com: *stored procedures*, *triggers*,



integridade referencial, SQL (ANSI 92/99), linguagem nativa (PSQL<sup>2</sup>), UDFs (Funções Definidas por Usuários) e outros recursos. E vale lembrar que a maioria destes recursos foram herdados do Interbase, portanto estão amplamente testados e tem sua eficiência comprovada". (MAGNO, 2013).

Alguns pontos podem ser citados para explicar o que faz do *Firebird* um bom SGBD (MAGNO, 2012):

- É um banco de dados leve no que diz respeito ao tamanho de servidor;
- Possui alta capacidade de armazenar;
- Excelente gerenciamento em projetos com grande volume de tráfego e informação;
- Instalação e manutenção são extremamente simples;
- Os arquivos de banco de dados crescem dinamicamente conforme a necessidade;
- É multiplataforma com suporte nativo para diversos sistemas operacionais incluindo, Windows, Mac OS, Solaris, Linux.

### 3.1.4 IBExpert

O *IBExpert* é a interação do gerenciador do banco de dados com o desenvolvedor, é a interface visual do banco de dados *Interbase* e *Firebird*. Entre os muitos recursos oferecidos pelo *IBExpert* no gerenciamento de bases de dados *Interbase/Firebird* estão os relacionados a seguir (LEITE, 2007):

- Suporta as versões 5, 6 e 7 do Interbase e as do Firebird;
- Gerencia ferramenta de modelagem (*Data Designer*);
- Oferece sites com hiperlinks para objetos de banco de dados como *triggers*, tabelas, *procedures* etc;
- Oferece *debugger* para *triggers*;
- Trabalha com mais de um banco de dados simultaneamente;
- Exporta dados do tipo XLS<sup>3</sup>, RTF<sup>4</sup>, HTML<sup>5</sup>, TXT<sup>6</sup> etc.;
- Gerencia vários usuários;

---

<sup>2</sup> PSQL é o PostgreSQL, um SGBD.

<sup>3</sup> XLS é a extensão utilizada pelos ficheiros de folha de cálculos (ou planilhas eletrônicas) – *Microsoft Excel*.

<sup>4</sup> RFT é o acrônimo de *Rich Text Format* ou Formato Rico de Texto.

<sup>5</sup> HTML – *Hyper Text Markup Language* ou Linguagem de Marcação de Hipertexto.

<sup>6</sup> TXT é a extensão de um arquivo de texto.

- Tem documentação do banco de dados no formato HTML;
- Permite construção visual de consultas SQL com o *Visual Query Builder*;
- Permite a criação de tabelas sem a necessidade de instruções SQL.

O *IBExpert* permite a criação de tabelas sem a necessidade da utilização de comandos DDL (*Data Definition Language* - Linguagem de Definição de Dados). Todos os atributos de uma tabela, tais como: campos, chave primária (*primary key*), chave estrangeira (*foreign key*), índices, *triggers*, etc., podem ser criados e alterados visualmente. O *IBExpert* gera automaticamente o *script* para criar o banco de dados.

### 3.1.5 Delphi

Delphi é uma IDE – *Integrated Development Environment* (ambiente de desenvolvimento integrado). Esta ferramenta possibilita o desenvolvimento de sistemas com interface visual, incluindo sistemas gerenciais onde se consegue de maneira prática e fácil produzir softwares *desktop* com qualidade e em pouco tempo. O Delphi utiliza como base a linguagem de programação Pascal.

Cantú (2006) afirma que:

“Alguns recursos originais do Delphi que me atraíram foram sua estratégia baseada em formulários e orientada a objetos, seu compilador extremamente rápido, seu ótimo suporte a banco de dados, sua íntima integração com a programação para Windows e sua tecnologia de componentes, o elemento mais importante, porém, foi a linguagem *Object Pascal* – a fundação de todo o resto” (Cantú, 2006).

Considerando as características citadas por Cantú, pode-se verificar que o Delphi é uma ótima IDE de desenvolvimento visual orientado a objetos.

Para Cantú (2006) a sintaxe do Pascal herdada pelo Delphi é bastante prolixa e mais legível que outras como o C, por exemplo. Sua extensão orientada a objetos também apresenta mesmo poder da nova geração de linguagens orientadas a objeto como o Java e C#. Juntando alto processamento com linguagem fácil e simples, forte orientação a objetos e ótimos suporte a banco de dados tem-se uma das ferramentas de desenvolvimento de sistemas *desktop* mais perfeitas.

O Delphi conta ainda com componentes e ferramentas que minimizam o trabalho manual, economizando tempo. Além disso, o Delphi é uma ferramenta

aberta a vários componentes externos e diversos bancos de dados, incluindo o banco de dados Firebird, utilizado neste trabalho.

### **3.1.6 *FastReport***

O *FastReport* é um gerador de relatórios para todas as versões do Delphi. E mesmo sendo nativo no Delphi, o *FastReport* é independente, pois permite criar uma interface com o usuário e até mesmo a conexão com o banco de dados dentro dele mesmo.

O gerador de Relatório *FastReport* é uma solução moderna para a integração de Inteligência Empresarial<sup>7</sup> em seu software. Ela foi criada para desenvolvedores que querem usar componentes prontos para relatórios. *FastReport*, com a sua simplicidade de uso, conveniência e distribuição de tamanho pequeno é capaz de proporcionar alta funcionalidade e performance em quase qualquer computador moderno. (FASTREPORT, 2013).

Com a utilização do *FastReport* pode-se perceber uma maior velocidade na execução do relatório e também, a criação de um arquivo de relatório de tamanho muito menor. Em comparação com outros geradores de relatórios também é possível notar que ele é muito mais organizado e muito mais fácil de trabalhar.

## **3.2 Método**

Para a realização deste trabalho, foram realizadas pesquisas diversas sobre o segmento de mercado escolhido e sobre as ferramentas que seriam utilizadas.

A pesquisa realizada caracteriza-se como um estudo de caso por focar empresas do segmento de pequenos restaurantes. É uma pesquisa aplicada, pois envolve a aplicação de tecnologias estudadas durante a graduação.

Esse trabalho foi realizado com base no modelo de ciclo de vida Clássico ou Cascata que segundo Audy e Prikladnicki (2008), “é um ciclo de vida que utiliza um

---

<sup>7</sup> Inteligência empresarial (em inglês *Business Intelligence*) refere-se ao processo de coleta, organização, análise, compartilhamento e monitoramento de informações que oferecem suporte a gestão de negócios

método sistemático e sequencial, em que o resultado de uma fase constitui a entrada de outra”, a qual neste trabalho respeita as seguintes etapas:

Levantamento de requisitos: o processo de captura de requisitos pode ser suportado por diversas técnicas, como entrevistas, aplicação de questionários ou observação direta.

Neste trabalho, o levantamento de dados para análise de requisitos foi realizado por meio de observação dos acadêmicos em diversos estabelecimentos do ramo gastronômico e por conhecimento dos desenvolvedores por terem relação com pessoas que possuem pequenos restaurantes, buscando identificar as principais necessidades do segmento. Além disso, foram realizadas entrevistas informais com pessoas próximas que possuem restaurantes de pequeno porte. Estas técnicas de levantamento de informações permitem ao analista obter de várias pessoas afetadas pelo sistema, informações, tais como necessidades, posturas, crenças ou comportamento dos envolvidos.

Análise e Projeto: na elaboração dos diagramas da UML para as fases de análise e projeto, foi utilizada como base a obra do autor Wazlawick (2011). Primeiramente foi desenvolvido o diagrama de casos de uso, pois é necessário definir os processos que o sistema irá controlar antes de detalhar o sistema através de outros diagramas como o modelo conceitual que já inclui atributos, propriedades e relações entre as classes. Em seguida foi realizado o caso de uso expandido do método cadastrar pedido. E posteriormente foram elaborados os demais diagramas, modelo conceitual, sequência, comunicação e entidade-relacionamento.

Implementação e Testes: para a implementação do sistema foram utilizadas as ferramentas *Delphi 7* e *IBExpert*. A codificação seguiu o conceito de programação orientada a objetos dando ênfase em características como: polimorfismo, herança, encapsulamento e desenvolvimento de classes. E por fim, a fase de testes foi realizada apenas pelos desenvolvedores, ou seja, o sistema não foi implantado em nenhum estabelecimento do ramo de pequenos restaurantes.

## 4 RESULTADOS E DISCUSSÕES

Neste capítulo apresenta-se o sistema desenvolvido, exibindo telas, trechos de códigos orientados a objetos, modelagem, análise de requisitos, estrutura do banco de dados e todos os detalhes referentes à estruturação do sistema desenvolvido.

Primeiramente o sistema é descrito em uma visão do lado do cliente. A seguir são apresentados os resultados obtidos com a coleta e análise dos requisitos, e a modelagem gerada sobre o sistema, incluindo os diagramas da UML: Diagrama de Casos de Uso, Conceitual, Sequência e Comunicação. É apresentada ainda a modelagem do banco de dados, através de um diagrama de entidade e relacionamento.

Por fim é exibido o sistema, sua interface visual e o código gerado, destacando os principais trechos de código desenvolvidos utilizando conceitos de herança, polimorfismo e demais características da orientação a objetos. Vale destacar que foram criados cadastros com tela genérica de consulta e telas de cadastro e lançamentos padrão para herança, além das telas principais de pedidos e despesas.

### 4.1 Modelagem do Sistema

Para desenvolvimento de um *software*, o primeiro passo, e um dos mais importantes, é a realização da análise, que engloba entre outras coisas, o processo de levantamento de requisitos. Este processo permitirá definir quais são os problemas ou necessidades do cliente, e quais são as possibilidades de resolvê-los através do *software* que será desenvolvido. Tendo os requisitos definidos é possível então modelar o sistema.

O sistema desenvolvido foi modelado utilizando o paradigma de orientação a objetos, baseando-se na UML, criando diagramas de sequência e comunicação, de casos de uso e de classes.

Esta seção irá abordar os requisitos levantados e classificados como funcionais e não-funcionais, e posterior modelagem gerada para o sistema e para o banco de dados com base nestes requisitos.

## **4.2 Requisitos de usuário para o Sistema**

O software resultado deste trabalho visa manipular, de maneira simples e organizada, os processos de restaurantes de pequeno porte, desde simples cadastros de produtos, despesas e usuários até os pedidos, passando também por relatórios gerenciais que serão de grande valia para processos de decisão dos gestores da organização.

Num primeiro momento, deverá ser cadastrado um usuário, o qual irá operar o sistema como administrador. A partir deste momento, esse usuário deverá cadastrar alguns dados essenciais ao uso do sistema, para tanto ele deverá fazer um levantamento de seu estoque de produtos com seus respectivos valores. De posse destes dados, inicialmente será necessário realizar o cadastro dos produtos disponíveis para utilização/venda, por meio de um formulário de produtos. Posteriormente, com os produtos cadastrados, já será possível realizar as vendas, utilizando um formulário de pedidos. Também será possível ao administrador do sistema cadastrar as despesas da organização, tais como gastos com água, luz, fornecedores, entre outros. O sistema deverá oferecer uma série padrão de relatórios, abrangendo todos os dados cadastrados no sistema, mas também deve deixar aberto ao cliente um canal pelo qual ele poderá solicitar ao desenvolvedor algum modelo específico de relatório que seja necessário para a sua organização. Os principais relatórios que compõem a série padrão seriam um para receitas e outro para despesas, com períodos definidos ou não.

### **4.2.1 Requisitos Funcionais e Não-Funcionais**

A coleta de requisitos foi definida através de observações em diferentes estabelecimentos e pelo fato dos acadêmicos terem ligação com pessoas que atuam nesse ramo.

Desta forma foi possível detectar os problemas presentes na gerência dos estabelecimentos que não utilizam um sistema informatizado. São eles: controle de entradas, despesas, pedidos e o trabalho de “fechar o caixa” manualmente todos os dias. Com isso foi possível desenvolver o levantamento de requisitos apresentado na Tabela 1 e na Tabela 2.

Os requisitos funcionais levantados são apresentados utilizando a abreviatura RFX para identificar cada requisito, sendo X um identificador numérico. E os requisitos não-funcionais, caracterizados como restrições ou complemento às funcionalidades que o *software* deve possuir, são apresentados utilizando-se a abreviatura RNFX para identificar cada requisito, sendo X o identificador numérico do requisito não funcional.

Nas tabelas geradas, todos os requisitos serão implementados e terão prioridade, pois caso algum requisito não seja implementado o sistema não funcionará.

A Tabela 1 apresenta os requisitos funcionais identificados para o sistema.

Identificação	Nome	Descrição
RF1	<i>Cadastrar Produtos</i>	O Administrador do sistema deve cadastrar os produtos informando a sua descrição e valor.
RF2	<i>Cadastrar Usuário</i>	O Administrador do sistema deverá cadastrar os usuários informando o nome, senha e CPF.
RF3	<i>Cadastrar Despesas</i>	O responsável pelo sistema deverá cadastrar as despesas informando a descrição, valor e data.
RF4	<i>Cadastrar Pedido</i>	O sistema deverá registrar os pedidos efetuados pelos clientes, bem como os produtos consumidos, quantidade e o valor total.
RF5	<i>Registrar Receitas</i>	O sistema deve registrar as receitas feitas a partir do cadastro de pedido realizado pelo usuário do sistema.

**Tabela 1 - Requisitos Funcionais do Sistema**

A Tabela 2 apresenta os requisitos não-funcionais identificados para o sistema. Os requisitos não-funcionais explicitam regras de negócio, restrições ao sistema de acesso, por exemplo, requisitos de qualidade, desempenho, segurança e outros. Os requisitos não-funcionais do sistema estão relacionados aos requisitos funcionais pelos nomes.

Identificação	Nome	Descrição
RNF1	<i>Cadastrar Produto</i>	O Administrador do sistema deve cadastrar a descrição dos produtos que serão vendidos e o valor dos mesmos. Sendo que um mesmo produto não poderá ter mais de um cadastro e com valores diferentes.
RNF2	<i>Cadastrar Itens de Pedido</i>	O usuário do sistema deve cadastrar os produtos em um pedido. Desta forma, não será possível cadastrar itens de pedido sem que ele esteja associado a um pedido e vice-versa.
RNF3	<i>Cadastrar Usuário</i>	O administrador do sistema deverá cadastrar o nome, senha e CPF. Porém um mesmo usuário não poderá ser registrado mais de uma vez, tendo como validador o CPF.
RNF4	<i>Segurança</i>	Apenas usuários cadastrados poderão utilizar o sistema. Para garantir a segurança das informações.
RNF5	<i>Interface</i>	O sistema deverá calcular o valor total de cada pedido.
RNF6	<i>Janela Única</i>	Todas as funções relacionadas a pedido e pagamento devem ser efetuadas em uma única janela.
RNF7	<i>Emitir Recibo</i>	O sistema deverá emitir um recibo quando um pedido for efetuado. Sendo que esse recibo será apenas um recibo simples (ou seja, não fiscal).

**Tabela 2 - Requisitos Não Funcionais do Sistema**

#### 4.2.2 Diagrama de Caso de Uso

Na Figura 1 apresenta-se o diagrama de casos de uso que aborda as funcionalidades do sistema, e a interação do usuário com estas funcionalidades. O primeiro ator apresentado no diagrama é o usuário Operador do estabelecimento, que não possui acesso de Administrador, sendo assim, é responsável apenas pela realização de cadastro de produtos, dos pedidos e emissão de recibos.

Já o Administrador é responsável por todas as atividades desempenhadas pelo usuário Operador, mais as atividades de cadastrar despesas e cadastrar usuários. Inicialmente o sistema terá um usuário cadastrado, sendo esse o usuário Administrador, e a partir dele será permitido realizar os cadastros dos demais usuários.



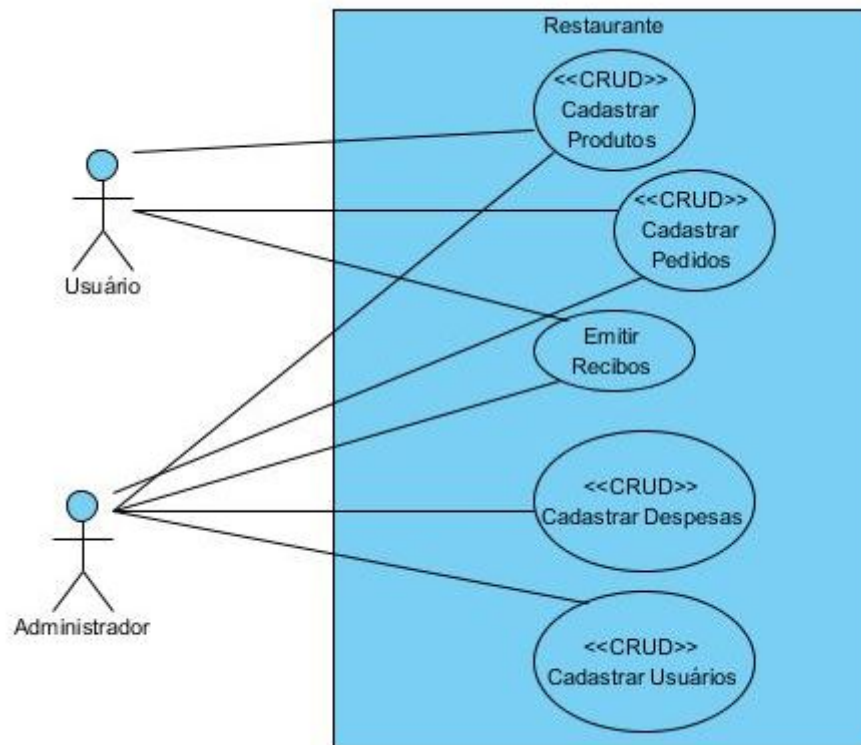


Figura 1 - Documentação: Diagrama de Casos de Uso

A Tabela 3 abaixo apresenta o caso de uso expandido do processo de cadastrar pedido.

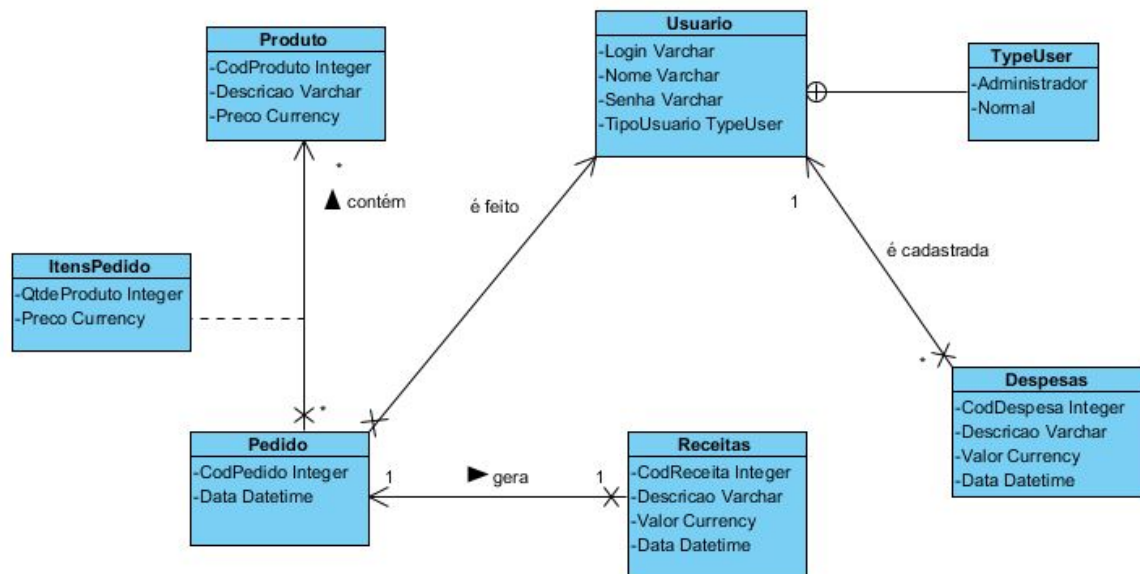
<b>Caso de Uso: Cadastrar Pedido</b>
Atores: Usuário
Precondições: Os dados dos produtos devem estar previamente cadastrados
Pós-condições: Pedido cadastrado com produtos inclusos
Requisitos Correlacionados: Cadastrar produtos (RF1)
Variações tecnológicas:
<b>Fluxo Principal:</b>
1. [IN] Este caso de uso inicia quando o usuário vai atender um cliente e lançar seu pedido no sistema.
2. [IN] O usuário informa ao sistema o código e a data do pedido.
3. [OUT] O sistema habilita ao usuário o grid para o lançamento dos produtos do pedido.
4. [IN] O usuário informa ao sistema quais os produtos serão lançados no pedido com suas respectivas quantidades.
5. [OUT] O sistema efetiva os itens do pedido.
6. [IN] O usuário solicita um recibo referente ao pedido realizado.
7. [OUT] O sistema emite o recibo de compra.
<b>Tratamento de Exceções:</b>
3.1a: O código do pedido informado já está cadastrado no sistema.
Variante 3.1a: Volta para o início
3.1.1a: Mostra uma mensagem com o erro para o usuário.

3.1.2a: Volta para o passo 2.  
 Variante 3.2a: Cancela a operação  
 3.2.1a: Mostra uma mensagem ao usuário.  
 3.2.2a: Cancela o pedido.

**Tabela 3 - Caso de Uso Expandido (Cadastrar Pedido)**

#### 4.2.3 Modelo Conceitual

Na Figura 2 apresenta-se o diagrama conceitual. Este diagrama foi criado como um espelho do banco de dados, ou seja, todas as classes são classes de entidade e, portanto, devem ser persistentes. As Tabelas a seguir (4 a 10)... No sentido de que todas as tabelas possuem uma classe, a qual contém todos os campos da respectiva tabela. Todas as operações de manipulação e acesso ao banco de dados serão realizadas dentro da própria classe, e as telas do sistema apenas criam a classe e executam os métodos desta.



**Figura 2 - Documentação: Diagrama Conceitual**

As Tabelas 4 a 10 apresentam a descrição de cada uma das classes do diagrama de classes, representado na Figura 2.

<b>Identificação:</b>	Produto
<b>Descrição:</b>	É previsto que nesta classe sejam requisitados os dados dos produtos para que assim o cadastro seja efetuado com sucesso.
<b>Requisitos:</b>	RF1
<b>Atributos:</b>	CodProduto(Integer): código de cadastramento do produto.

	Descricao(Varchar): descrição do produto que será comercializado. Preco(Currency): preço do produto.
<b>Métodos:</b>	Cadastrar, Alterar, Consultar e Excluir.
<b>Observações:</b>	

Tabela 4 - Descrição da Classe Produto

<b>Identificação:</b>	Usuario
<b>Descrição:</b>	Esta classe é responsável pelos dados cadastrais dos usuários do sistema.
<b>Requisitos:</b>	RF2
<b>Atributos:</b>	Login(String): nome do login do usuário. NomeUsuario(Varchar): nome do usuário. Senha(Varchar): senha do usuário. TipoUsuario(TypeUser): tipo do usuário.
<b>Métodos:</b>	Cadastrar, Alterar, Consultar e Excluir.
<b>Observações:</b>	

Tabela 5 - Descrição da Classe Usuario

<b>Identificação:</b>	TypeUser
<b>Descrição:</b>	Esta classe é responsável por fornecer o tipo de usuário para a classe usuário.
<b>Requisitos:</b>	RF2
<b>Atributos:</b>	Administrador. Normal
<b>Métodos:</b>	
<b>Observações:</b>	

Tabela 6 - Descrição da Classe TypeUser

<b>Identificação:</b>	Despesas
<b>Descrição:</b>	Classe responsável por registrar as despesas do estabelecimento.
<b>Requisitos:</b>	RF3
<b>Atributos:</b>	CodDespesa(Integer): código de cadastramento da despesa. Descricao(Varchar): descrição da despesa. Valor(Currency): valor da despesa. Data(Datetime): data da inclusão da despesa no sistema.
<b>Métodos:</b>	Cadastrar, Alterar, Consultar, Excluir.
<b>Observações:</b>	

Tabela 7 - Descrição da Classe Despesas

<b>Identificação:</b>	Pedido
<b>Descrição:</b>	Esta classe é responsável por registrar os pedidos realizados pelos clientes.
<b>Requisitos:</b>	RF4
<b>Atributos:</b>	CodPedido(Integer): código de cadastramento do pedido. Data(Datetime): data do pedido.
<b>Métodos:</b>	Cadastrar, Alterar, Excluir, Consultar.

<b>Observações:</b>	
---------------------	--

Tabela 8 - Descrição da Classe Pedido

<b>Identificação:</b>	Receitas
<b>Descrição:</b>	Classe responsável por registrar as receitas do estabelecimento.
<b>Requisitos:</b>	RF5
<b>Atributos:</b>	CodReceita(Integer): código de cadastramento da receita. Descricao(Varchar): descrição da receita. Valor(Currency): valor da receita. Data(Datetime): data da inclusão da receita no sistema.
<b>Métodos:</b>	Cadastrar, Alterar, Consultar, Excluir.
<b>Observações:</b>	

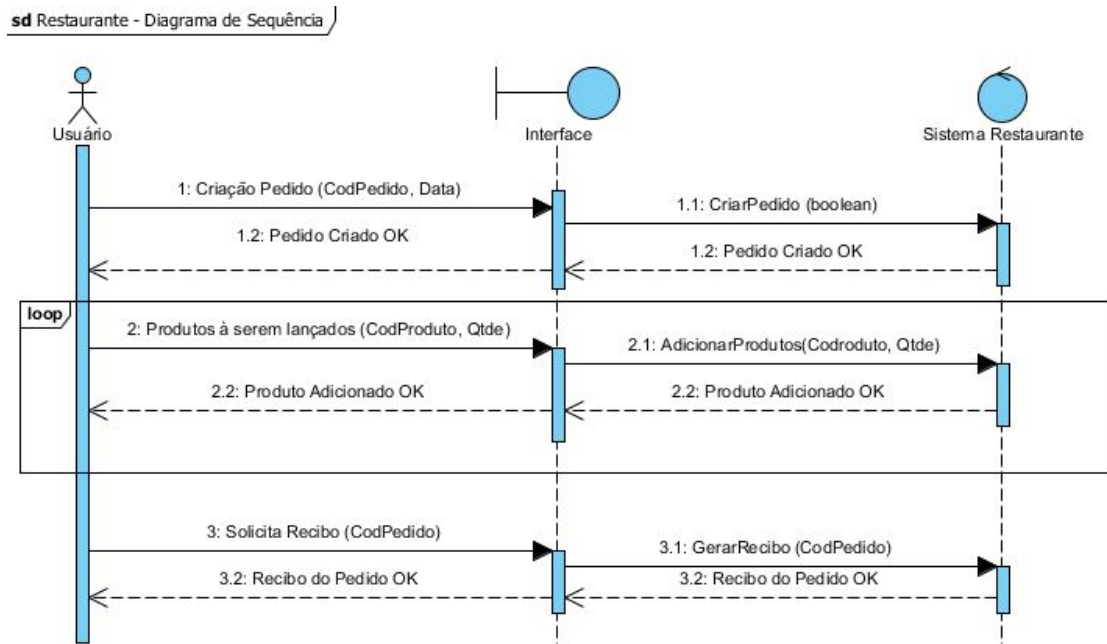
Tabela 9 - Descrição da Classe Receitas

<b>Identificação:</b>	ItensPedido
<b>Descrição:</b>	Classe responsável por armazenar os produtos referentes a cada pedido.
<b>Requisitos:</b>	RF1 e FR4
<b>Atributos:</b>	QtdeProduto(Integer): quantidade do produto. Preco(Currency): preço do produto do pedido.
<b>Métodos:</b>	Cadastrar, Alterar, Consultar, Excluir.
<b>Observações:</b>	O cadastro de itens de pedido é feito junto com o cadastro de pedidos, uma vez que apenas a partir do cadastro de um pedido será realizado o cadastro de itens de pedido.

Tabela 10 - Descrição da Classe ItensPedido

#### 4.2.4 Diagrama de Sequência

Na Figura 3 apresenta-se o diagrama de sequência, que esboça a sequência de acontecimentos no decorrer do processamento do sistema.

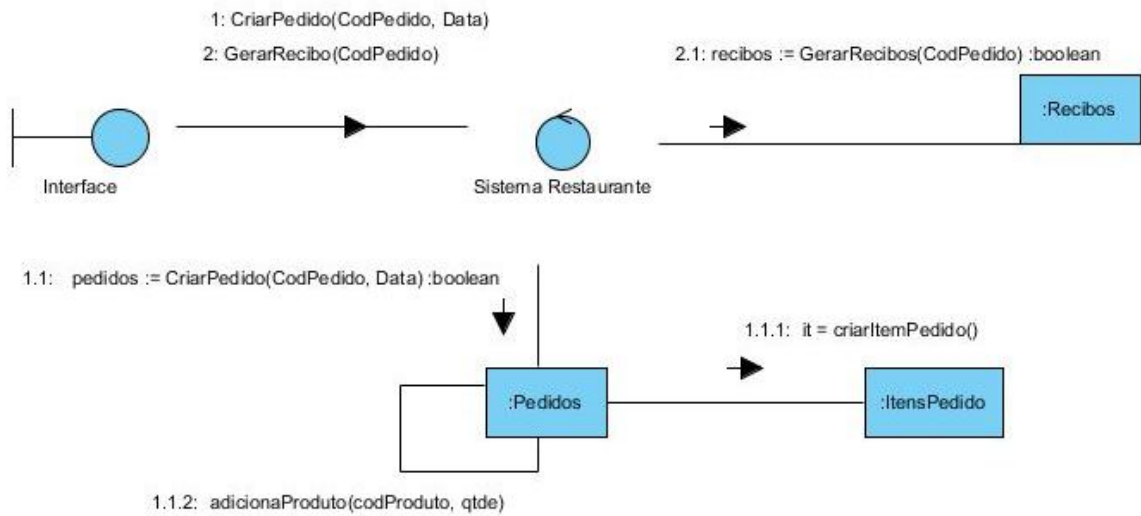


**Figura 3 - Documentação: Diagrama de Sequência (Cadastrar Pedido)**

Primeiramente o usuário irá solicitar a criação de um pedido, caso o código informado não exista, o sistema criará o pedido, caso ele já exista, o sistema não permitirá a criação. Após isso, o usuário entrará em um *looping* para adicionar os produtos do pedido. Ao concluir a inclusão de todos os produtos, o usuário solicitará o recibo ao sistema e este irá gerar e emitir um recibo.

#### 4.2.5 Diagrama de Comunicação

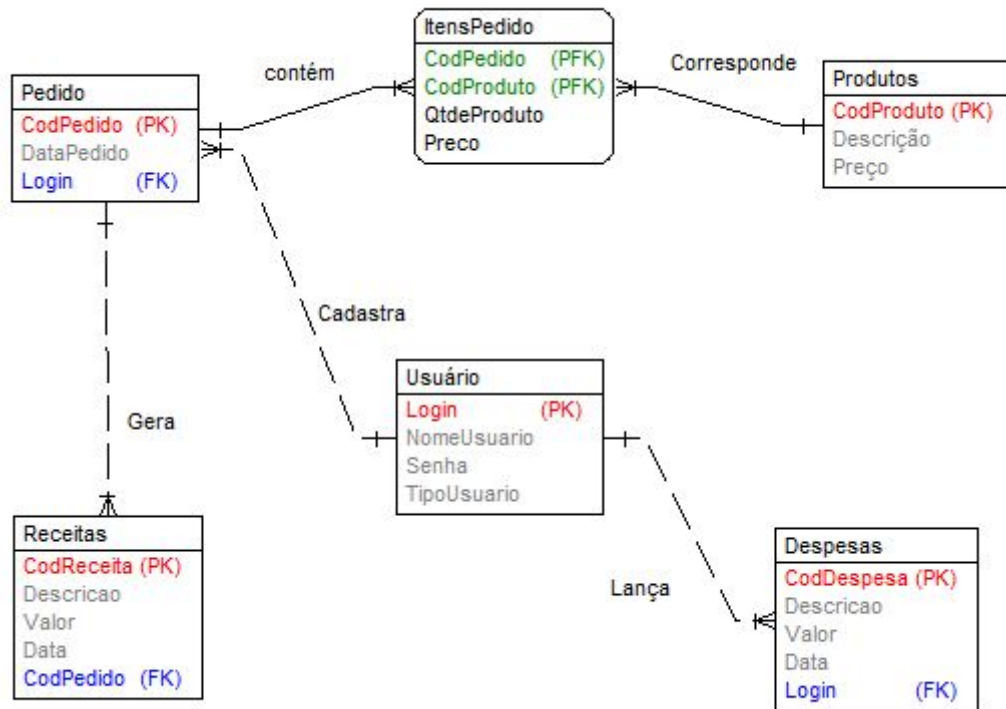
Na Figura 4 está o diagrama de comunicação, que mostra de maneira semelhante ao diagrama de sequência, a comunicação dinâmica entre os objetos, ou seja, o fluxo das mensagens entre um objeto e outro para um dado processo. O diagrama de comunicação serve de complemento para o diagrama de sequência, ao passo que decompõe a classe controladora nas classes que estarão envolvidas no processo, sendo possível; identificar os métodos necessários a cada classe.



**Figura 4 - Documentação: Diagrama de Comunicação - Cadastrar Pedido**

#### 4.2.6 Diagrama de Entidade e Relacionamento

O Diagrama de Entidade e Relacionamento (DER) modela as tabelas de um banco de dados, definindo campos, tipo de dados de cada campo, chaves primárias e relacionamentos entre as tabelas determinando as chaves secundárias ou estrangeiras. Destaca-se que o DER apresentado na Figura 5 apresenta o mapeamento objeto-relacional a partir do modelo conceitual da Figura 2. Na Figura 5 observam-se todas as tabelas que armazenam os dados processados no sistema e seus respectivos campos e interligações.



**Figura 5 - Documentação: Diagrama de Entidade e Relacionamento**

A seguir são apresentadas as tabelas de descrições das entidades que compõem o banco de dados, conforme expõem a Figura 5.

A Tabela 11 apresenta a descrição da tabela Produto.

PRODUTO						
Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Padrão	Observações
CodProduto	Integer	Não	Sim	Não		Auto-denominado
Descricao	Varchar	Não	Não	Não		
Preco	Currency	Não	Não	Não		
Login	String	Não	Não	Sim		

**Tabela 11- Descrição da Tabela Produto (DER)**

A Tabela 12 apresenta a descrição da tabela Pedido.

PEDIDO						
Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Padrão	Observações
CodPedido	Integer	Não	Sim	Sim		Auto-denominado

DataPedido	Datetime	Não	Não	Não		
Login	String	Não	Não	Sim		

Tabela 12 - Descrição da Tabela Pedido (DER)

A Tabela 13 apresenta a descrição da tabela ItensPedido.

ITENSPEDIDO						
Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Padrão	Observações
CodPedido	Integer	Não	Sim	Sim		
CodProduto	Integer	Não	Sim	Sim		
QtdeProduto	Integer	Não	Não	Não		
Preco	Currency	Não	Não	Não		

Tabela 13 - Descrição da Tabela ItensPedido (DER)

A Tabela 14 apresenta a descrição da tabela Usuario.

USUARIO						
Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Padrão	Observações
Login	String	Não	Sim	Sim		Auto-denominado
NomeUsuario	Varchar	Não	Não	Não		
Senha	Varchar	Não	Não	Não		
TipoUsuario	TypeUser	Não	Não	Não		

Tabela 14 - Descrição da Tabela Usuario (DER)

A Tabela 15 apresenta a descrição da tabela Despesas.

DESPESAS						
Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Padrão	Observações
CodDespesa	Integer	Não	Sim	Não		Auto-denominado
Login	String	Não	Não	Sim		
Descricao	Varchar	Não	Não	Não		
Valor	Currency	Não	Não	Não		
Data	Datetime	Não	Não	Não		

Tabela 15 - Descrição da Tabela Despesas (DER)



A Tabela 16 apresenta a descrição da tabela Receitas.

RECEITAS						
Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Padrão	Observações
CodReceita	Integer	Não	Sim	Não		Auto-denominado
Descricao	Varchar	Não	Não	Não		
Valor	Currency	Não	Não	Não		
Data	Datetime	Não	Não	Não		
CodPedido	Integer	Não	Não	Sim		

Tabela 16 - Descrição da Tabela Receitas (DER)

### 4.3 Apresentação do Sistema

Na Figura 6 está a janela principal do sistema. Essa tela é visualizada quando o sistema é acessado e há um usuário logado.

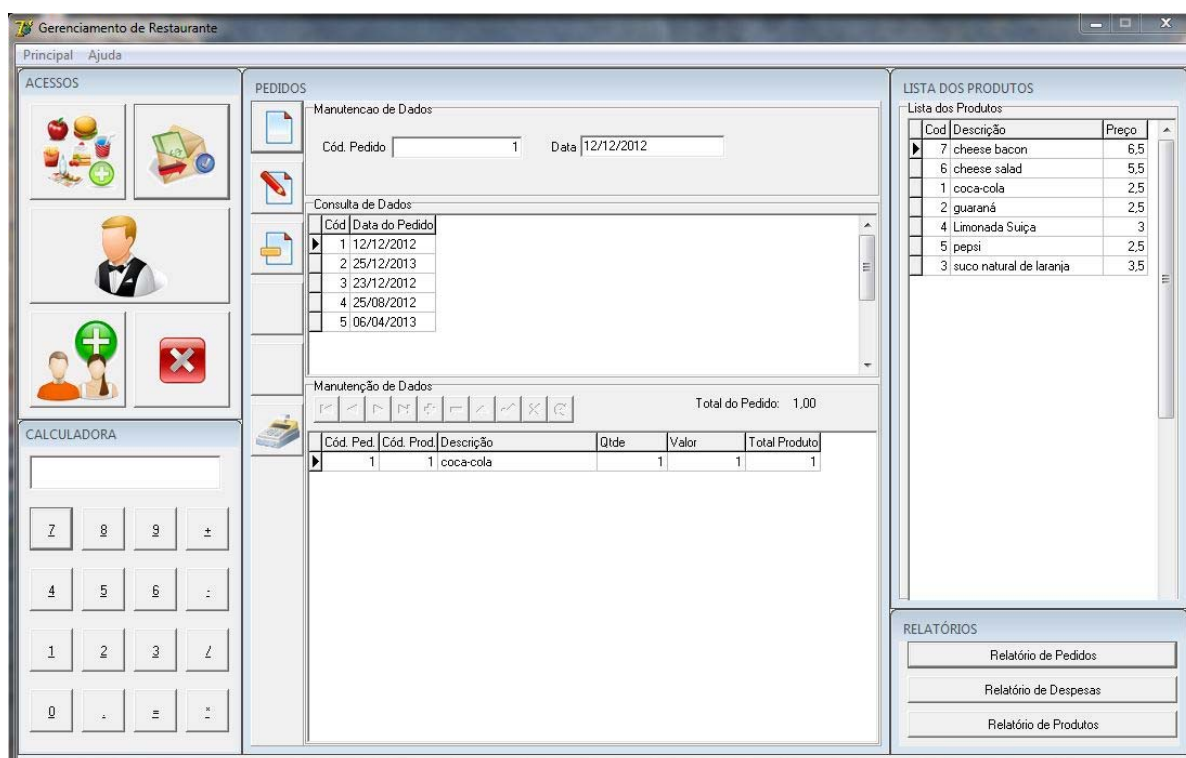


Figura 6 - Tela: Apresentação Inicial do Sistema

Como pode ser visto na Figura 6 os botões de cadastros, como: produtos, usuários, despesas e pedidos, assim como o de fechar o sistema estão localizados no lado esquerdo superior da tela. Logo abaixo, apresenta-se uma calculadora para que o usuário não precise ter em mãos uma fisicamente ou até mesmo que precise ficar com uma aberta no computador. A inserção desta calculadora é um meio mais acessível para o usuário. Já o lado direito é composto por uma lista dos produtos cadastrados no sistema, essa lista servirá de consulta para que o usuário possa visualizar o código dos produtos no momento de adicioná-los ao pedido, sendo que há uma grande possibilidade dessa lista se tornar extensa, assim futuramente poderá ser implementado um campo de pesquisa. E no canto inferior há opções de relatórios como o de despesas, pedidos e produtos que estão cadastrados no sistema.

Como o sistema foi implementado para ser em uma única janela, o objetivo é fazer com que o conteúdo central se modifique conforme o usuário clique no botão de cadastros, por exemplo.

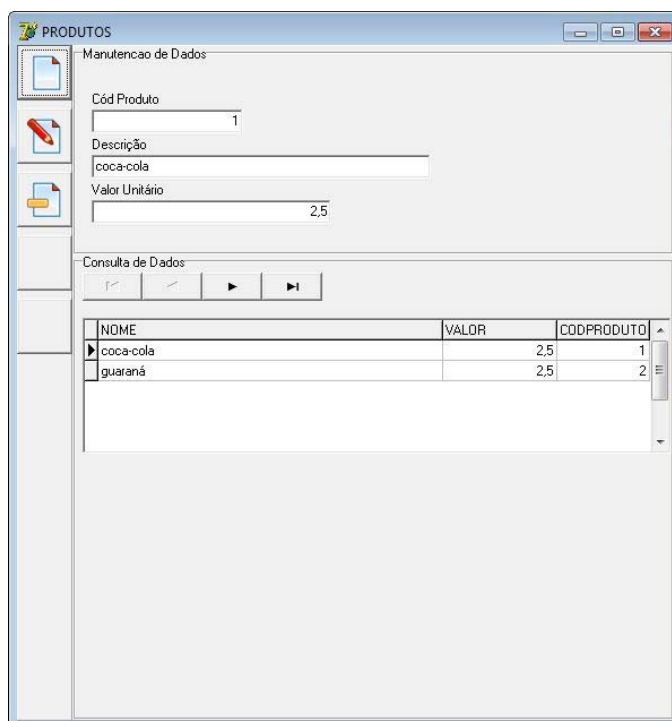
Na Figura 7 pode-se visualizar a tela de Pedidos, a qual o usuário utilizará informando dados, como o número do pedido, a data, e os produtos que compõem o pedido.

The screenshot shows a software window titled 'PEDIDOS'. It contains several sections:

- Manutenção de Dados:** Fields for 'Cód. Pedido' (value: 1) and 'Data' (value: 12/12/2012).
- Consulta de Dados:** A table with columns 'Cód.' and 'Data do Pedido' containing five rows of data.
- Manutenção de Dados (lower):** A set of navigation buttons and a 'Total do Pedido: 1,00' label.
- Table:** A table with columns 'Cód. Ped.', 'Cód. Prod.', 'Descrição', 'Qtde', 'Valor', and 'Total Produto'. It contains one row with data: 1, 1, coca-cola, 1, 1, 1.

**Figura 7 - Tela: Formulário de Cadastro de Pedidos**

Para o cadastro dos produtos o sistema altera o conteúdo central, exibindo a tela mostrada na Figura 8.



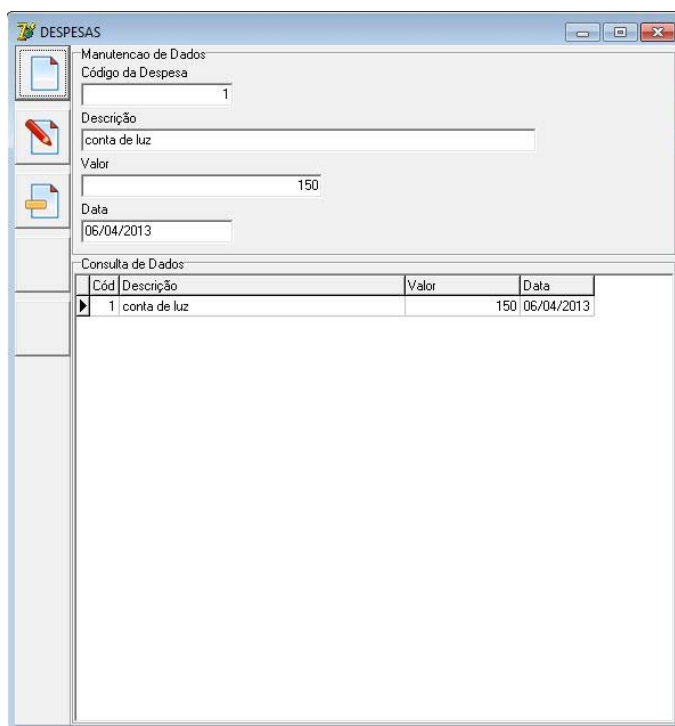
The screenshot shows a software window titled 'PRODUTOS'. It is divided into two main sections: 'Manutencao de Dados' (Data Maintenance) and 'Consulta de Dados' (Data Query). The 'Manutencao de Dados' section contains three input fields: 'Cód Produto' with the value '1', 'Descrição' with the value 'coca-cola', and 'Valor Unitário' with the value '2,5'. The 'Consulta de Dados' section features a set of navigation buttons and a table with the following data:

NOME	VALOR	CODPRODUTO
coca-cola	2,5	1
guaraná	2,5	2

**Figura 8 - Tela: Formulário de Cadastro de Produtos**

Nesse formulário o usuário deverá inserir o código do produto, a descrição (nome do produto) e o valor a ser comercializado pelo restaurante. Ao realizar a inserção, o produto irá aparecer automaticamente na lista de produtos.

Para realizar o controle de despesas do estabelecimento no sistema há uma opção de cadastro de despesas apresentado na Figura 9.



The screenshot shows a software window titled 'DESPESAS'. It is divided into two main sections: 'Manutencao de Dados' (Data Maintenance) and 'Consulta de Dados' (Data Query).

**Manutencao de Dados:**

- Código da Despesa:** 1
- Descrição:** conta de luz
- Valor:** 150
- Data:** 06/04/2013

**Consulta de Dados:**

Cód	Descrição	Valor	Data
1	conta de luz	150	06/04/2013

**Figura 9 - Tela: Formulário de Cadastro de Despesas**

As despesas cadastradas podem ser de energia, água, compra de produtos, entre outros. Para cadastrar é necessário preencher os campos código da despesa, a descrição, o valor e a data.

O ultimo cadastro disponível no sistema é o de usuários. Nesse formulário um usuário já cadastrado, de preferência o responsável do estabelecimento, informará o login, a senha e o nome do novo usuário. Esse formulário está apresentado na Figura 10.

The screenshot shows a software window titled 'USUÁRIOS'. It is divided into two main sections: 'Manutenção de Dados' (Data Maintenance) and 'Consulta de Dados' (Data Query).

**Manutenção de Dados:** This section contains a form for adding or editing user information. The fields are:
 

- Login:** newton
- Senha:** [masked with asterisks]
- Nome:** newton

**Consulta de Dados:** This section displays a table with the following data:
 

Login	Nome
newton	newton

**Figura 10 - Tela: Formulário de Cadastro de Usuários**

Pequenos restaurantes não são obrigados a emitirem notas fiscais registradas, mas isso não significa que um recibo simples não possa ser emitido para o cliente confirmando seu pagamento. Desta forma, foi implementado no sistema uma maneira de emitir um comprovante caso o cliente queira esse recibo. A Figura 11 apresenta uma visualização desse recibo.

The screenshot shows a 'Preview' window of a receipt. The receipt is titled 'Recibo de Venda de Produtos' and is dated '06/04/2013' at '18:42:25'.

Nro. Pedido	Data
3	23/12/2012

Código - Descrição	Qtde.	Valor Unitário	Total Produto
1 coca-cola	2	R\$ 2,20	R\$ 4,40
2 guaraná	3	R\$ 2,50	R\$ 7,50
<b>Total do Pedido</b>			<b>R\$ 11,90</b>

The window also includes a toolbar with navigation and zoom controls, and a 'Page 1 of 1' indicator at the bottom.

**Figura 11- Tela: Recibo Simples de um Pedido**

É importante que o responsável/gerente do restaurante tenha acesso as informações cadastradas. Essas são visualizadas através de relatórios. Permitindo assim que as progressões ou regressões, melhorias a serem feitas, possam ser analisadas e que haja uma melhor tomada de decisão do administrador do estabelecimento. Um exemplo dos relatórios gerados é apresentado na Figura 12, referente ao relatório de pedidos.

Relatório de Pedidos						06/04/2013
						Página 1
Nro. Pedido	Data					
	Produto (Código - Descrição)		Quantidade	Valor Unitário	Total Produto	
1		12/12/2012				
1	coca-cola		R\$ 1,00	R\$ 1,00	R\$ 1,00	
				Total do Pedido	R\$ 1,00	
2		25/12/2013				
2	guaraná		R\$ 2,00	R\$ 2,50	R\$ 5,00	
				Total do Pedido	R\$ 5,00	
3		23/12/2012				
1	coca-cola		R\$ 2,00	R\$ 2,20	R\$ 4,40	
2	guaraná		R\$ 3,00	R\$ 2,50	R\$ 7,50	
				Total do Pedido	R\$ 11,90	
4		25/08/2012				
1	coca-cola		R\$ 3,00	R\$ 2,50	R\$ 7,50	
				Total do Pedido	R\$ 7,50	

**Figura 12 - Tela: Relatório de Pedidos**

Os relatórios que o sistema permite emitir são: Pedidos, Despesas e Produtos.

#### 4.4 Implementação do Sistema

Todas as classes desse sistema foram desenvolvidas baseadas nos conceitos de orientação a objetos, visto ao longo da graduação. Dentro desta seção

serão apresentados alguns trechos considerados mais importantes ao funcionamento geral do sistema.

Inicialmente foi codificado um formulário padrão, o qual foi utilizado para desenvolver alguns formulários ‘filhos’ (criados com base no conceito de herança). No Quadro 1 pode-se observar todas as variáveis e métodos genéricos, inclusive o método `validaDados`, que no formulário padrão foi declarado como *virtual/abstract* e em seus formulários filhos como *override*. Os conceitos que regem este formulário padrão é a herança e o polimorfismo, pois os métodos genéricos codificados no formulário padrão foram reaproveitados nos formulários filhos, e em alguns casos sobrescritos. Segundo Cantú (2006), “A vantagem do Polimorfismo é poder escrever código mais simples tratando tipos de objetos diferentes como se fossem iguais obtendo o comportamento correto em tempo de execução”.

```
1. unit unPadraoTCC;
2.
3. interface
4.
5. uses
6.   Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
7.   Dialogs, StdCtrls, ToolWin, ComCtrls, ExtCtrls, Buttons, DB, DBClient,
8.   Provider;
9.
10. type
11.   TEstadoBotoes = (tebIncluir, tebAlterar, tebExcluir, tebCancelar);
12.
13.   TfrmPadraoTCC = class(TForm)
14.     Panel1: TPanel;
15.     Panel2: TPanel;
16.     grbManutencao: TGroupBox;
17.     grbConsulta: TGroupBox;
18.     btnNovo: TBitBtn;
19.     btnAlterar: TBitBtn;
20.     btnExcluir: TBitBtn;
21.     btnCancel: TBitBtn;
22.     btnSalvar: TBitBtn;
23.     btnFechar: TBitBtn;
24.     procedure FormClose(Sender: TObject; var Action: TCloseAction);
25.     procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
26.     procedure btnFecharClick(Sender: TObject);
27.     procedure FormCreate(Sender: TObject);
28.     procedure btnNovoClick(Sender: TObject);
```

```

29.     procedure btnAlterarClick(Sender: TObject);
30.     procedure btnExcluirClick(Sender: TObject);
31.     procedure btnCancelClick(Sender: TObject);
32.     procedure btnSalvarClick(Sender: TObject);
33.     procedure FormKeyPress(Sender: TObject; var Key: Char);
34.     private
35.         { Private declarations }
36.     procedure TipoManutencao(Oper : TEstadoBotoes);
37.     protected
38.     function DadosValidos: boolean; virtual; abstract;
39.     public
40.         { Public declarations }
41.     procedure EstadoBotoes(Oper : TEstadoBotoes);
42.     end;
43.
44.     var
45.         frmPadraoTCC: TfrmPadraoTCC;
46.
47.     implementation
48.
49.     uses unBiblioteca;
50.
51.     {$R *.dfm}
52.
53.     procedure TfrmPadraoTCC.FormClose(Sender: TObject;
54.         var Action: TCloseAction);
55.     begin
56.         Action := caFree;
57.     end;
58.
59.     procedure TfrmPadraoTCC.FormCloseQuery(Sender: TObject;
60.         var CanClose: Boolean);
61.     begin
62.         //canClose := msgS('Você deseja realmente fechar?') = idYes;
63.     end;
64.
65.     procedure TfrmPadraoTCC.btnFecharClick(Sender: TObject);
66.     begin
67.         Close;
68.     end;
69.
70.     procedure TfrmPadraoTCC.EstadoBotoes(Oper: TEstadoBotoes);
71.     begin
72.         btnNovo.Enabled := (Oper = tebCancelar) or (Oper = tebExcluir);
73.         btnAlterar.Enabled := btnNovo.Enabled;
74.         btnExcluir.Enabled := btnNovo.Enabled;
75.         grbConsulta.Enabled := btnNovo.Enabled;
76.         btnSalvar.Enabled := (Oper=tebIncluir) or (Oper=tebAlterar);
77.         btnCancel.Enabled := btnSalvar.Enabled;
78.         TipoManutencao(Oper);
79.     end;
80.
81.     procedure TfrmPadraoTCC.TipoManutencao(Oper: TEstadoBotoes);
82.     begin
83.         if Pos(' - [' , Caption) > 0 then
84.             Caption := Copy(Caption, 1, Pos(' - [' ,Caption)-1);
85.         case Oper of
86.             tebIncluir : Caption := Caption + ' - [Inclusão]';
87.             tebAlterar : Caption := Caption + ' - [Alteração]';

```



```
88.         tebExcluir : Caption := Caption + ' - [Exclusão]';
89.     end;
90. end;
91.
92. procedure TfrmPadraoTCC.FormCreate(Sender: TObject);
93. begin
94.     EstadoBotoes(tebCancelar);
95.     try
96.         cds.Open;
97.     except
98.         on E : Exception do
99.             msgOK('Erro ao abrir a fonte de dados: ' + E.Message);
100.        end;
101.    end;
102.
103. procedure TfrmPadraoTCC.btnNovoClick(Sender: TObject);
104. begin
105.     EstadoBotoes(tebIncluir);
106.     SelectFirst; // coloca o foco no primeiro componente
107.     cds.Append; // prepara para receber um novo registro
108. end;
109.
110. procedure TfrmPadraoTCC.btnAlterarClick(Sender: TObject);
111. begin
112.     EstadoBotoes(tebAlterar);
113.     cds.Edit; // editar um registro selecionado
114. end;
115.
116. procedure TfrmPadraoTCC.btnExcluirClick(Sender: TObject);
117. begin
118.     EstadoBotoes(tebExcluir);
119.     if (not cds.IsEmpty) and (msgS('Deseja excluir o registro
selecionado?') = idYes) then
120.         begin
121.             EstadoBotoes(tebCancelar);
122.             cds.Delete;
123.             if cds.ApplyUpdates(0) <> 0 then
124.                 msgOK('Erro ao excluir o registro!');
125.             end
126.         else
127.             EstadoBotoes(tebCancelar);end;
128.
129. procedure TfrmPadraoTCC.btnCancelClick(Sender: TObject);
130. begin
131.     EstadoBotoes(tebCancelar);
132.     cds.Cancel; // ignora a alteração
133. end;
134.
135. procedure TfrmPadraoTCC.btnSalvarClick(Sender: TObject);
136. begin
137.     if DadosValidos then
138.         begin
139.             cds.Post;
140.             if cds.ApplyUpdates(0) = 0 then
141.                 EstadoBotoes(tebCancelar)
142.             else
143.                 msgOK('Erro ao gravar o registro!');
144.             end;
145.         end;
```

```

146.
147.     procedure TfrmPadraoTCC.FormKeyPress(Sender: TObject; var Key
      Char);
148.     begin
149.         // mudar de campo ao pressionar o enter
150.         if key = #13 then
151.             begin
152.                 key := #0;
153.                 Perform(WM_NEXTDLGCTL,0,0);
154.             END;
155.         end;
156.
end.

```

**Quadro 1 - Código: Formulário Padrão**

Deste trecho de código, pode-se destacar na linha 38 a função protegida *DadosValidos*, a qual deverá ser reescrita nos formulários filhos, com a intenção de validar os componentes do tipo *EditText* quanto a conter dados, de modo que não será possível gravar dados considerados obrigatórios em branco. Além disto, nota-se também que foram codificadas as instruções utilizadas para acesso ao banco de dados.

Baseado no formulário padrão, foram implementados quatro formulários filhos, utilizados para cadastros diversos, sendo eles Usuários, Produtos, Despesas e Pedidos, os quais foram apresentados no capítulo anterior. Além destes, há um formulário Acessos, que tem o objetivo de alternar entre os formulários de cadastro. Neste formulário, destaca-se um trecho do código referente a alternância de um formulário de cadastro para outro no Quadro 2.

```

1. procedure TfrmAcessos.btnProdutosClick(Sender: TObject);
2. begin
3.     if Assigned(frmPedidos) then
4.         frmPedidos.Close;
5.     if Assigned(frmUsuario) then
6.         frmUsuario.Close;
7.     if Assigned(frmDespesas) then
8.         frmDespesas.Close;
9.     if not Assigned(frmProdutos) then
10.        begin
11.            frmProdutos := TfrmProdutos.Create(nil);
12.            frmProdutos.Top := 0;
13.            frmProdutos.Left := 207;
14.            frmProdutos.Height := 635;
15.            frmProdutos.Width := 598;
16.        end;
17. end;

```

**Quadro 2 - Código: Formulário de Acessos (Alternância de Telas)**

O principal formulário deste sistema é o de Pedidos, pois é nele que os usuários irão passar a maioria do tempo quando 'logados' (após terem informado usuário e senha para acessar o sistema). No Quadro 3, é apresentada sua codificação. Nota-se que seu código é mais complexo que nos outros cadastros, pois serão envolvidos e apresentados no formulário dados de duas tabelas diferentes em dois componentes do tipo *DBGrid* diferentes.

```

1. unit unPedidos;
2.
3. interface
4.
5. uses
6.   Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
7.   Dialogs, unPadraoTCC, DB, DBClient, Provider, StdCtrls, Buttons, ExtCtrls,
8.   Mask, DBCtrls, Grids, DBGrids, FMTBcd, SqlExpr;
9.
10. type
11.   TfrmPedidos = class(TfrmPadraoTCC)
12.     GroupBox1: TGroupBox;
13.     edtPedido: TDBEdit;
14.     Label1: TLabel;
15.     Label2: TLabel;
16.     dtsGridPedidos: TDataSource;
17.     qryPedidos: TSQLQuery;
18.     dtsPedidos: TDataSource;
19.     qryPedProdutos: TSQLQuery;
20.     cdsPedProdutos: TClientDataSet;
21.     dtsPedProdutos: TDataSource;
22.     qryPedidosNROPEDIDO: TIntegerField;
23.     qryPedidosDATAPEDIDO: TDateField;
24.     CDSNROPEDIDO: TIntegerField;
25.     CDSDATAPEDIDO: TDateField;
26.     CDSqryPedProdutos: TDataSetField;
27.     qryPedProdutosNROPEDIDO: TIntegerField;
28.     qryPedProdutosCODPRODUTO: TIntegerField;
29.     qryPedProdutosQUANTIDADE: TIntegerField;
30.     qryPedProdutosVALORUNIT: TFMTBCDField;
31.     qryPedProdutosNOME: TStringField;
32.     qryProdutos: TSQLQuery;
33.     cdsPedProdutosNROPEDIDO: TIntegerField;
34.     cdsPedProdutosCODPRODUTO: TIntegerField;
35.     cdsPedProdutosQUANTIDADE: TIntegerField;
36.     cdsPedProdutosVALORUNIT: TFMTBCDField;
37.     cdsPedProdutosNOME: TStringField;
38.     cdsPedProdutosTotal: TFloatField;
39.     cdsPedProdutosTotalPedido: TAggregateField;
40.     edtData: TDBEdit;
41.     dbgPedidos: TDBGrid;
42.     dbgPedProdutos: TDBGrid;
43.     dbnProdutos: TDBNavigator;
44.     Label3: TLabel;
45.     edtTotalPedido: TDBText;
46.     btnRelatorio: TBitBtn;
47.     procedure cdsPedProdutosCODPRODUTOValidate(Sender: TField);
48.     procedure btnExcluirClick(Sender: TObject);
49.     procedure btnNovoClick(Sender: TObject);
50.     procedure btnAlterarClick(Sender: TObject);
51.     procedure btnCancelClick(Sender: TObject);
52.     procedure btnSalvarClick(Sender: TObject);

```

```

53.     procedure cdsPedProdutosPostError(DataSet: TDataSet; E: EDatabaseError;
54.         var Action: TDataAction);
55.     procedure CDSReconcileError(DataSet: TCustomClientDataSet;
56.         E: EReconcileError; UpdateKind: TUpdateKind;
57.         var Action: TReconcileAction);
58.     procedure FormClose(Sender: TObject; var Action: TCloseAction);
59.     procedure cdsPedProdutosCalcFields(DataSet: TDataSet);
60.     procedure FormShow(Sender: TObject);
61.     procedure btnRelatorioClick(Sender: TObject);
62. private
63.     { Private declarations }
64. public
65.     { Public declarations }
66. protected
67.     function DadosValidos : boolean; override;
68. end;
69.
70. var
71.     frmPedidos: TfrmPedidos;
72.
73. implementation
74.
75. uses unDM, unBiblioteca;
76.
77. {$R *.dfm}
78.
79. procedure TfrmPedidos.cdsPedProdutosCODPRODUTOValidate(Sender: TField);
80. begin
81.     inherited;
82.     qryProdutos.Close;
83.     qryProdutos.ParamByName('CODPRODUTO').AsString :=
84.         cdsPedProdutos.FieldByName('CODPRODUTO').AsString;
85.     cdsPedProdutos.FieldByName('NOME').AsString :=
86.         qryProdutos.FieldByName('NOME').AsString;
87.     cdsPedProdutos.FieldByName('VALORUNIT').AsFloat :=
88.         qryProdutos.FieldByName('VALOR').AsFloat;
89. end;
90.
91. procedure TfrmPedidos.btnExcluirClick(Sender: TObject);
92. begin
93.     EstadoBotoes(tebExcluir);
94.     if (not cds.IsEmpty) and (msgS('Deseja excluir o registro selecionado?') =
95.         idYes) then
96.         begin
97.             EstadoBotoes(tebCancelar);
98.             while not cdsPedProdutos.IsEmpty do
99.                 cdsPedProdutos.Delete;
100.            cds.Delete;
101.            if cds.ApplyUpdates(0) <> 0 then
102.                msgOK('Erro ao excluir o registro!')
103.            else
104.                begin
105.                    dbgPedProdutos.Options := dbgPedProdutos.Options - [dgEditing];
106.                    dbnProdutos.Enabled := False;
107.                end;
108.            end;
109.        else
110.            EstadoBotoes(tebCancelar);
111.        end;
112.
113.     function TfrmPedidos.DadosValidos: boolean;
114.     begin
115.         Result := true;
116.         if edtPedido.Text = '' then
117.             begin

```

```
115.         msgOK('Informe o número do pedido!');
116.         edtPedido.SetFocus;
117.         Result := false;
118.     end
119.     else if edtData.Text = ' / / ' then
120.     begin
121.         msgOK('Informe a data do pedido!');
122.         edtData.SetFocus;
123.         result := False;
124.     end
125.     else if cdsPedProdutos.RecordCount = 0 then
126.     begin
127.         msgOK('Informe os itens do pedido!');
128.         Result := False;
129.     end;
130. end;
131.
132. procedure TfrmPedidos.btnNovoClick(Sender: TObject);
133. begin
134.     inherited;
135.     dbgPedProdutos.Options := dbgPedProdutos.Options + [dgEditing];
136.     dbnProdutos.Enabled := True;
137.     btnRelatorio.Enabled := False;
138. end;
139.
140. procedure TfrmPedidos.btnAlterarClick(Sender: TObject);
141. begin
142.     inherited;
143.     dbgPedProdutos.Options := dbgPedProdutos.Options + [dgEditing];
144.     dbnProdutos.Enabled := True;
145.     btnRelatorio.Enabled := False;
146. end;
147.
148. procedure TfrmPedidos.btnCancelClick(Sender: TObject);
149. begin
150.     inherited;
151.     dbgPedProdutos.Options := dbgPedProdutos.Options - [dgEditing];
152.     dbnProdutos.Enabled := False;
153.     btnRelatorio.Enabled := True;
154. end;
155.
156. procedure TfrmPedidos.btnSalvarClick(Sender: TObject);
157. begin
158.     inherited;
159.     if not (cds.State in [dsInsert, dsEdit]) then
160.     begin
161.         dbgPedProdutos.Options := dbgPedProdutos.Options - [dgEditing];
162.         dbnProdutos.Enabled := False;
163.         btnRelatorio.Enabled := True;
164.     end;
165. end;
166.
167. procedure TfrmPedidos.cdsPedProdutosPostError(DataSet: TDataSet;
168.     E: EDatabaseError; var Action: TDataAction);
169. begin
170.     inherited;
171.     if Pos('Key', E.Message) > 0 THEN
172.     begin
173.         msgOK('O item informado já foi incluído neste Pedido!');
174.         Action := daAbort;
175.     end
176.     else if Pos('must have a value', E.Message) > 0 THEN
177.     begin
178.         msgOK('Preencha todos os campos!');
179.         Action := daAbort;
180.     end
```

```

181.     end;
182.
183.     procedure TfrmPedidos.CDSReconcileError(DataSet: TCustomClientDataSet;
184.       E: EReconcileError; UpdateKind: TUpdateKind;
185.       var Action: TReconcileAction);
186.     begin
187.       inherited;
188.       if Pos('FOREIGN', E.Message) > 0 THEN
189.         begin
190.           msgOK('Verifique! Um ou mais itens informados não existem');
191.           Action := raAbort;
192.         end;
193.     end;
194.
195.     procedure TfrmPedidos.FormClose(Sender: TObject; var Action:
196.       TCloseAction);
197.     begin
198.       inherited;
199.       frmPedidos := nil;
200.     end;
201.
202.     procedure TfrmPedidos.cdsPedProdutosCalcFields(DataSet: TDataSet);
203.     begin
204.       inherited;
205.       if cdsPedProdutos.State = dsInternalCalc then
206.         cdsPedProdutos.FieldByName('TOTAL').AsFloat :=
207.           cdsPedProdutos.FieldByName('QUANTIDADE').AsFloat *
208.           cdsPedProdutos.FieldByName('VALORUNIT').AsFloat;
209.     end;
210.
211.     procedure TfrmPedidos.FormShow(Sender: TObject);
212.     begin
213.       inherited;
214.       if Assigned(frmPedidos) then
215.         begin
216.           frmPedidos.Top := 0;
217.           frmPedidos.Left := 207;
218.           frmPedidos.Height := 635;
219.           frmPedidos.Width := 598;
220.         end;
221.     end;
222.
223.     procedure TfrmPedidos.btnRelatorioClick(Sender: TObject);
224.     begin
225.       dm.qryPedido.Close;
226.       dm.qryPedido.SQL.Clear;
227.       dm.qryPedido.SQL.Add('SELECT * FROM PEDIDO WHERE NROPEDIDO = ' +
228.         cds.FieldByName('NROPEDIDO').AsString);
229.       dm.qryPedido.Open;
230.       dm.qryPedidoProduto.Open;
231.       dm.frxReport.LoadFromFile('ReciboRel.fr3');
232.       dm.frxReport.ShowReport();
233.       dm.qryPedido.Close;
234.       dm.qryPedidoProduto.Close;
235.       dm.qryPedido.SQL.Clear;
236.       dm.qryPedido.SQL.Add('SELECT * FROM PEDIDO ORDER BY
237.         PEDIDO.NROPEDIDO');
238.     end;
239. end.

```

**Quadro 3 - Código: Formulário Cadastro de Pedidos**

Tendo em vista que o usuário do sistema poderá realizar algumas contas durante seus processos, foi implementado um formulário Calculadora, com as 4 operações básicas. Segue abaixo, no Quadro 4, a sua codificação.

```

1. unit unCalculadora;
2.
3. interface
4.
5. uses
6.   Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
7.   Dialogs, StdCtrls, Buttons;
8.
9. type
10.  TfrmCalculadora = class(TForm)
11.    bnt7: TBitBtn;
12.    btn4: TBitBtn;
13.    btn1: TBitBtn;
14.    btn8: TBitBtn;
15.    btn5: TBitBtn;
16.    btn2: TBitBtn;
17.    btn9: TBitBtn;
18.    btn3: TBitBtn;
19.    btnig: TBitBtn;
20.    btn0: TBitBtn;
21.    btnpt: TBitBtn;
22.    btn6: TBitBtn;
23.    edtValor: TEdit;
24.    btnad: TBitBtn;
25.    btnmn: TBitBtn;
26.    btndiv: TBitBtn;
27.    btnmlt: TBitBtn;
28.    btnLimpar: TBitBtn;
29.    procedure FormShow(Sender: TObject);
30.    procedure bnt7Click(Sender: TObject);
31.    procedure btn8Click(Sender: TObject);
32.    procedure btn9Click(Sender: TObject);
33.    procedure btn4Click(Sender: TObject);
34.    procedure btn5Click(Sender: TObject);
35.    procedure btn6Click(Sender: TObject);
36.    procedure btn1Click(Sender: TObject);
37.    procedure btn2Click(Sender: TObject);
38.    procedure btn3Click(Sender: TObject);
39.    procedure btn0Click(Sender: TObject);
40.    procedure btnptClick(Sender: TObject);
41.    procedure btnadClick(Sender: TObject);
42.    procedure btnmnClick(Sender: TObject);
43.    procedure btndivClick(Sender: TObject);
44.    procedure btnmltClick(Sender: TObject);
45.    procedure btnigClick(Sender: TObject);
46.    procedure btnLimparClick(Sender: TObject);
47.  private
48.    { Private declarations }
49.  public
50.    { Public declarations }
51.  end;
52.
53. var
54.  frmCalculadora: TfrmCalculadora;
55.  Valor : real;
56.  Operacao : integer;
57.  valorinteiro : integer;
58.

```

```
59. implementation
60.
61. {$R *.dfm}
62.
63.     procedure TfrmCalculadora.FormShow(Sender: TObject);
64.     begin
65.         frmCalculadora.Left := 0;
66.         frmCalculadora.Top := 325;
67.     end;
68.
69.     procedure TfrmCalculadora.bnt7Click(Sender: TObject);
70.     begin
71.         if (edtValor.Text = '0') then
72.             edtValor.Text := '7'
73.         else
74.             edtValor.Text:=edtValor.Text +'7';
75.     end;
76.
77.     procedure TfrmCalculadora.btn8Click(Sender: TObject);
78.     begin
79.         if (edtValor.Text = '0') then
80.             edtValor.Text := '8'
81.         else
82.             edtValor.Text:=edtValor.Text +'8';
83.     end;
84.
85.     procedure TfrmCalculadora.btn9Click(Sender: TObject);
86.     begin
87.         if (edtValor.Text = '0') then
88.             edtValor.Text := '9'
89.         else
90.             edtValor.Text:=edtValor.Text +'9';
91.     end;
92.
93.     procedure TfrmCalculadora.btn4Click(Sender: TObject);
94.     begin
95.         if (edtValor.Text = '0') then
96.             edtValor.Text := '4'
97.         else
98.             edtValor.Text:=edtValor.Text +'4';
99.     end;
100.
101.     procedure TfrmCalculadora.btn5Click(Sender: TObject);
102.     begin
103.         if (edtValor.Text = '0') then
104.             edtValor.Text := '5'
105.         else
106.             edtValor.Text:=edtValor.Text +'5';
107.     end;
108.
109.     procedure TfrmCalculadora.btn6Click(Sender: TObject);
110.     begin
111.         if (edtValor.Text = '0') then
112.             edtValor.Text := '6'
113.         else
114.             edtValor.Text:=edtValor.Text +'6';
115.     end;
116.
117.     procedure TfrmCalculadora.btn1Click(Sender: TObject);
118.     begin
119.         if (edtValor.Text = '0') then
120.             edtValor.Text := '1'
121.         else
122.             edtValor.Text:=edtValor.Text +'1';
123.     end;
```



```
124.     procedure TfrmCalculadora.btn2Click(Sender: TObject);
125.     begin
126.         if (edtValor.Text = '0') then
127.             edtValor.Text := '2'
128.         else
129.             edtValor.Text:=edtValor.Text +'2';
130.     end;
131.
132.     procedure TfrmCalculadora.btn3Click(Sender: TObject);
133.     begin
134.         if (edtValor.Text = '0') then
135.             edtValor.Text := '3'
136.         else
137.             edtValor.Text:=edtValor.Text +'3';
138.     end;
139.
140.     procedure TfrmCalculadora.btn0Click(Sender: TObject);
141.     begin
142.         if (edtValor.Text = '0') then
143.             edtValor.Text := '0'
144.         else
145.             edtValor.Text:=edtValor.Text +'0';
146.     end;
147.
148.     procedure TfrmCalculadora.btnptClick(Sender: TObject);
149.     begin
150.         edtValor.Text:=edtValor.Text +',';
151.     end;
152.
153.     procedure TfrmCalculadora.btnadClick(Sender: TObject);
154.     begin
155.         if (edtValor.Text <> '') then
156.             begin
157.                 valor:=strtofloat(edtValor.text);
158.                 operacao:=1;
159.                 edtValor.text:='';
160.             end;
161.     end;
162.
163.     procedure TfrmCalculadora.btnmnClick(Sender: TObject);
164.     begin
165.         if (edtValor.Text <> '') then
166.             begin
167.                 valor:=strtofloat(edtValor.text);
168.                 operacao:=2;
169.                 edtValor.text:='';
170.             end;
171.     end;
172.
173.     procedure TfrmCalculadora.btndivClick(Sender: TObject);
174.     begin
175.         if (edtValor.Text <> '') then
176.             begin
177.                 valor:=strtofloat(edtValor.text);
178.                 operacao:=3;
179.                 edtValor.text:='';
180.             end;
181.     end;
182.
183.     procedure TfrmCalculadora.btnmltClick(Sender: TObject);
184.     begin
185.         if (edtValor.Text <> '') then
186.             begin
187.                 valor:=strtofloat(edtValor.text);
188.                 operacao:=4;
189.                 edtValor.text:='';
```

```

190.     end;
191.     end;
192.
193.     procedure TfrmCalculadora.btnigClick(Sender: TObject);
194.     var
195.         total : string;
196.         valorreal : real;
197.     begin
198.         val(edtValor.text, valorreal, valorinteiro);
199.         case operacao of
200.             1: valor:= valor + valorreal;
201.             2: valor:= valor - valorreal;
202.             3: if (valorreal = 0) then
203.                 valor :=0
204.             else
205.                 valor:= valor / valorreal;
206.             4: valor:= valor * valorreal;
207.         end;
208.         str(valor:2:0, total);
209.         edtValor.text:= total;
210.     end;
211.
212.     procedure TfrmCalculadora.btnLimparClick(Sender: TObject);
213.     begin
214.         edtValor.Clear;
215.         Valor := 0;
216.         valorinteiro := 0;
217.     end;
218.
219.end.

```

**Quadro 4 - Código: Calculadora**

Para facilitar ao usuário o cadastramento de pedidos, foi desenvolvido um formulário com a lista dos produtos, que é atualizada automaticamente na medida em que novos produtos são cadastrados. Esta atualização é feita no evento *OnClick* do botão Salvar do formulário de pedidos. Note-se que este botão primeiro irá executar o código de seu formulário pai, devido ao termo *inherit* e depois irá executar o *refresh* no *ClientDataSet* que repassa os dados ao grid da lista de produtos. Essa atualização dos dados é apresentada no Quadro 5:

```

1.
2. procedure TfrmProdutos.btnSalvarClick(Sender: TObject);
3. begin
4.     inherited;
5.     frmListaProdutos.CDS.Refresh;
6. end;

```

**Quadro 5 - Código: Atualização da Lista de Produtos**

## 5 CONCLUSÃO

O trabalho desenvolvido teve como objetivo desenvolver um *software* simples e de valor acessível, porém permitindo informatizar as principais operações realizadas em um pequeno restaurante que não precisa registrar fiscalmente por meio de *software* a circulação de seus produtos.

O intuito do sistema é controlar internamente as operações, para melhorar o processo da empresa, considerando a entrada e saída de capital, por meio dos cadastros de pedidos e despesas respectivamente, o controle dos produtos que estão sendo comercializados por meio de relatórios. Pretende-se desta forma fornecer uma noção segura ao gerente/empresário do quanto sua empresa está progredindo, ou se está regredindo, auxiliando a montar metas para a empresa, melhorando a administração da mesma.

Com as pesquisas realizadas, e a partir delas a realização da modelagem e desenvolvimento do *software*, foi possível criar uma ferramenta que atenda aos pequenos proprietários sem considerar fatores fiscais e contábeis. Pois muitos empresários não possuem o conhecimento exato dessas obrigações, as deixando para escritórios autorizados que realizam esse processo. Desta forma, o *software* de gestão desenvolvido é empregado para controle interno somente, tornando mais fácil a realização dos processos executados na empresa, e o entendimento dos envolvidos.

Para modelagem e implementação do sistema, foram utilizadas ferramentas de fácil desenvolvimento, consideradas eficientes que forneceram uma interação fácil do usuário com o sistema. Ferramentas com uma aceitação de mercado muito grande, produzidas por empresas altamente conceituadas a nível mundial. Com todos esses fatores foi possível desenvolver um sistema, que pretende auxiliar na execução dos processos realizados em estabelecimentos que sempre tem um grande fluxo de atendimentos.

No decorrer do desenvolvimento deste trabalho houve um enriquecimento dos conhecimentos dos acadêmicos, sobretudo em relação à modelagem utilizando a UML. Foi possível aprender muito em relação à orientação a objetos, principalmente sobre utilização de classes, emprego de polimorfismo, herança e

encapsulamento, que auxiliaram bastante na reutilização de código e telas. De um modo geral, pode-se afirmar que o desenvolvimento deste trabalho permitiu aos acadêmicos adquirir uma experiência maior com o processo de análise e desenvolvimento de *software*.

## REFERÊNCIAS BIBLIOGRÁFICAS

AUDY, Jorge e Prikladnicki, Rafael. **Desenvolvimento Distribuído de Software: desenvolvimento de software com equipes distribuídas**. Rio de Janeiro: Ed. Elsevier, 2008.

BOOCH, Grady., RUMBAUGH, James. e JACOBSON, Ivar. **UML: Guia do Usuário**, 2ª. ed., Rio de Janeiro: Ed. Elsevier, 2005.

BROOKSHEAR, J. Glenn. **Ciência da Computação: uma visão abrangente**. 7ª. ed., Porto Alegre: Bookman, 2003.

CADORIN, Marcos Venicio. **Protótipo de Gerenciamento de Bares e Restaurantes**. 2010. 74 f. Monografia apresentada na UTFPR – *Campus Pato Branco* para a obtenção do grau de tecnólogo em Análise e Desenvolvimento de Sistemas.

CANTÚ, Marco. **Dominando o Delphi 2005: a bíblia**. São Paulo: Pearson Prentice Hall, 2006.

CASESTUDIO. **Case Studio**. Disponível em: <<http://www.casestudio.com/enu/default.aspx>>. Acesso em: 13 mar. 2013.

COSTA, Rogério Luís de C. **SQL: Guia Prático**, 2ª. ed., Rio de Janeiro: Ed, Brasport, 2006.

FASTREPORT. **Fast Report**. Disponível em: <<http://www.fast-report.com/pt/>>. Acesso em: 02 abr. de 2013.

FOWLER, Martins. **UML Essencial: Um Breve Guia para a Linguagem-Padrão de Modelagem de Objetos**. 3ª. ed., Porto Alegre: Ed. Bookman, 2005.

FURTADO, Vasco. **Tecnologia e Gestão da Informação na Segurança Pública**. Rio de Janeiro: Ed. Gramond, 2002.

GIANINI, Tatiana. **Comer, beber, prosperar**. Revista EXAME, 2011. Disponível em: <<http://exame.abril.com.br/revista-exame/edicoes/0984/noticias/comer-beber-prosperar?page=1>>. Acesso em: 04 dez. 2012.

LEITE, Mário. **Acessando bancos de dados com ferramenta RAD: aplicações em Visual Basic**. Rio de Janeiro: Brasport, 2007.

MAGNO, Alexandre. **Firebird – O que torna um banco de dados atraente?** Disponível em: <[http://www.firebaseio.com.br/fb/imgdocs/amagno\\_fisl6.pdf](http://www.firebaseio.com.br/fb/imgdocs/amagno_fisl6.pdf)>. Acesso em: 27 mar. 2013.

MARICATO, Percival. **Como Montar e Administrar Bares e Restaurantes**, 6ª. ed., São Paulo: Ed. Senac São Paulo, 2005.

MARTINS, José Carlos Cordeiro. **Gerenciando Projetos de Desenvolvimento de Software com PMI, RUP e UML**. 5ª. ed., Rio de Janeiro: Ed. Brasport, 2010.

MELO, Ana Cristina. **Desenvolvendo aplicações com UML 2.2: do conceitual à implementação**, 3ª. ed., Rio de Janeiro: Ed. Brasport, 2010.

O'BRIEN, A. James e MARAKAS, M. George. **Administração de Sistemas de Informação**. 15ª. ed., São Paulo: Ed. Bookman, 2010.

OLIVEIRA, D. P. R. **Sistemas de Informações Gerenciais: estratégicas, táticas, operacionais**. São Paulo: Ed. Atlas, 2001. (ISBN 852242819-0)

RAMOS, Ricardo Argenton. **Treinamento Prático em UML**. São Paulo: Ed. Digerati Books, 2006.

REZENDE, Denis Alcides. **Engenharia de Software e Sistemas de Informação**, 3ª. ed., Rio de Janeiro: Ed. Brasport, 2005.

VISUALPARADIGM. **Visual Paradigm**. Disponível em: <<http://www.visual-paradigm.com/>>. Acesso em: 13 mar. 2013.

WAZLAWICK, Raul Sidnei. **Análise e Projeto de Sistemas de Informação Orientados a Objetos**. 2ª. ed., Rio de Janeiro: Ed. Elsevier, 2011.