

SISTEMA DE ENSINO PRESENCIAL CONECTADO

Tecnologia em Análise e Desenvolvimento de Sistemas

ALBERTO TUPAIBA FALCADE GRAZIADEI

ALEXANDRE BRAGA   
DOUGLAS DE PAULA GONÇALVES

LUIS CARLOS DA SILVA

NEYMAR DANIEL CORRÊA

ROBERTO INACIO MOLON

**PRODUÇÃO TEXTUAL GRUPO**

Pizzaria An Poparome

Caxias do Sul

2015

ALBERTO TUPAIBA FALCADE GRAZIADEI

ALEXANDRE BRAGA   
DOUGLAS DE PAULA GONÇALVES

LUIS CARLOS DA SILVA

NEYMAR DANIEL CORRÊA

ROBERTO INACIO MOLON

**PRODUÇÃO TEXTUAL INDIVIDUAL**

Pizzaria An Poparome

Trabalho apresentado ao Curso Superior de Tecnologia em Análise e Desenvolvimeno de Sistema da Universidade Norte do Paraná - UNOPAR, para as disciplinas de:

Banco de Dados I

Análise Orientada a Objetos I

Linguagens de Programação e Estrutura de Dados

Organização de Computadores.

Orientadores:

Prof. Anderson E.M. Gonçalves e Merris Mozer

Disciplina: Linguagens Programação e Estrutura de Dados

Prof. Paulo K. Nishitani

Disciplina: Organização de Computadores

Prof. Polyanna P. G. Fabris

Disciplina: Análise Orientada a Objetos I

Prof. Roberto Y. Nishimura

Disciplina: Banco de Dados I

Caxias do Sul

2015

SUMÁRIO

[2. OBJETIVO. 5](#_Toc420088076)

[3. DESENVOLVIMENTO. 6](#_Toc420088077)

[3.1. PIZZARIA. 6](#_Toc420088078)

[3.2. DIAGRAMA DE CASO DE USO. 7](#_Toc420088079)

[3.3. DIAGRAMA DE CLASSE. 8](#_Toc420088080)

[3.4. ESTRUTURA DE DADOS. 11](#_Toc420088081)

[3.4.1. LISTA ENCADEADA. 12](#_Toc420088082)

[3.4.2. PILHA. 13](#_Toc420088083)

[3.4.3. FILAS. 13](#_Toc420088084)

[3.4.4. LISTA CIRCULAR. 13](#_Toc420088085)

[3.4.5. LISTA DUPLAMENTE ENCADEADA. 13](#_Toc420088086)

[3.5. BRMODELO. 15](#_Toc420088087)

[3.6. VANTAGENS E DESVANTAGENS. 17](#_Toc420088088)

[3.6.1. VANTAGENS. 18](#_Toc420088089)

[3.6.2. COMPARTILHAMENTO DE ARQUIVOS DE TRABALHO. 18](#_Toc420088090)

[3.6.3. COMPARTILHAMENTO DE PROGRAMAS. 18](#_Toc420088091)

[3.6.4. COMPARTILHAMENTO DE PERIFÉRICOS. 18](#_Toc420088092)

[3.6.5. COMPARTILHAMENTO DE IMPRESSORAS. 19](#_Toc420088093)

[3.7. DESVANTAGENS. 19](#_Toc420088094)

[3.7.1. ATAQUE A VÍRUS. 19](#_Toc420088095)

[3.7.2. PROBLEMAS GENERALIZADOS. 19](#_Toc420088096)

[4. CONCLUSÃO 35](#_Toc420088097)

[5. REFERÊNCIA 36](#_Toc420088098)

**1. INTRODUÇÃO.**

Viemos com este trabalho apresentar o funcionamento de uma pizzaria avaliando a importância que tem um SGBD (Sistema Gerenciador de Banco de Dados) para o gerenciar o negócio.

O que estamos percebendo no momento é de que cada vez mais necessitados de informações precisas e o principal em tempo real.

Os programas que usam Banco de Dados têm a vantagem de estar sempre na frente porque tudo está relacionado com a necessidade da informação organizada nos ajudando na manipulação e gerenciamento dos dados e transformando em informações.

Hoje podemos pegar um telefone e fazer a compra de uma pizza, o que o cliente não sabe e como isso funciona do outro lado da linha até porque isso não lhe interessa.

Temos que nos preparar e proporcionar comodidade e agilidade no atendimento.

#### 2. OBJETIVO.

O Objetivo é desenvolver um sistema SGBD de uso simples e ágil na hora que o cliente faz o pedido.

Disponibilizar o sistema para o controle de tele entrega das pizzas, visando o crescimento da empresa.

Precisamos atender nossos clientes com o máximo de informações por exemplo sabendo quais pizzas que são as suas preferidas.

#### 3. DESENVOLVIMENTO.

Nossas atividades estão voltadas a organização dos dados e na manipulação de forma correta. O sistema de controle da pizzaria será desenvolvido com todas as ferramentas necessárias para um controle de todas as etapas que um sistema precisa para atender as necessidades do cliente.

Não podemos ficar totalmente inertes das informações, precisamos usar as ferramentas para fazer um Diagrama de Caso de uso um diagrama de Classe passar pelo BRmodelo.

A linguagem será C Sharp (C#) através do Visual Studio e conectado ao Banco de dados Gratuito o MYSQL, o que devemos saber é que existe no mercado muitos programas para trabalharmos com SGBD.

#### 3.1. PIZZARIA.

A empresa “An Poparome” e a Software Unopar-Caxias pesquisou automatização no processo de entrega de pizza, analisando e levantamento os requisitos, temos a informação de um crescimento futuro, mas o necessário será o controle das entregas, então foi identificado as seguintes necessidades:

**Cliente:** Controle usando o nome, telefone, endereço, ponto de referência e data de nascimento, como base principal será usado telefone fixo para cada residência;

**Entregadores:** Controle usando nome, CPF, RG e celular.

**Produto:** As pizzas terão a inclusão dos dados, nome do produto, descrição, tamanho e valor;

**Pedido:** Os pedidos realizados deverão ter: nome do cliente, nome do produto, quantidade, tamanho e deverá conter várias pizzas, porém e cada pizza apenas um sabor e no final o Total com a entrega e o troco;

Os pedidos deverão atender e respeitar a ordem das ligações, obedecendo os seguintes status:

**Pendente:**  O atendente finaliza seu registro;

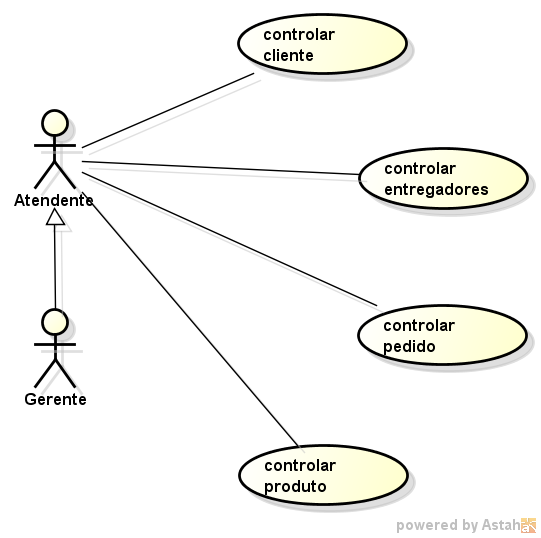
**Em trânsito:**  O motoboy retira o pedido para a entrega;

**Cancelado:** Intercorrência e o pedido não pode ser entregue;

**Entregue:** Conclui a entrega e apresenta o pagamento;

Semanalmente, o sistema terá condições de gerar um relatório das “Entregas Realizadas” para o pagamento dos entregadores.

#### 3.2. DIAGRAMA DE CASO DE USO.

No Diagrama de Caso de Uso (Use Case Diagram), tivemos como base o levantamento de requisitos foi possível identificar as necessidades do proprietário para automatizar seu processo de entrega. Com este diagrama podemos apresentar a base do controle do sistema que o Sr. Mário vai utilizar para expansão da pizzaria, conforme está na figura abaixo.

Actors: Atendente e Gerente;

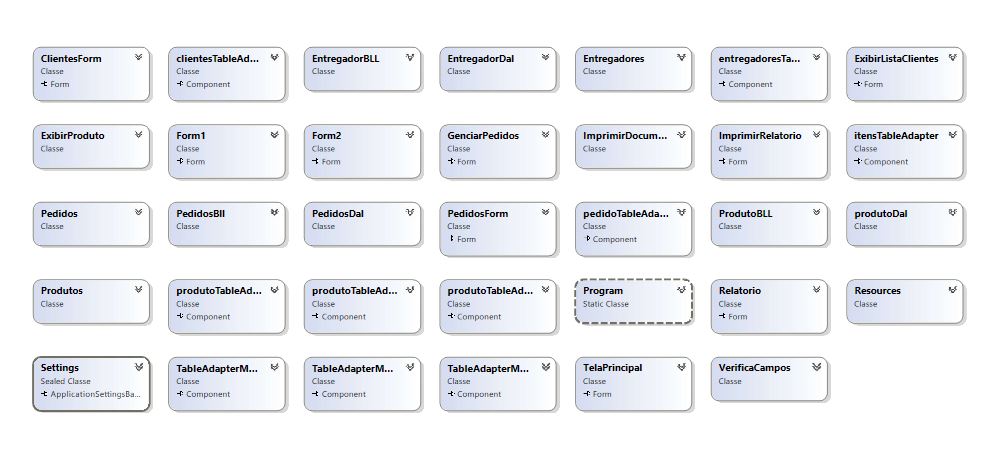
Use cases: Controlar Cliente, Controlar Entregadores, Controlar Pedido e Controlar Produto.

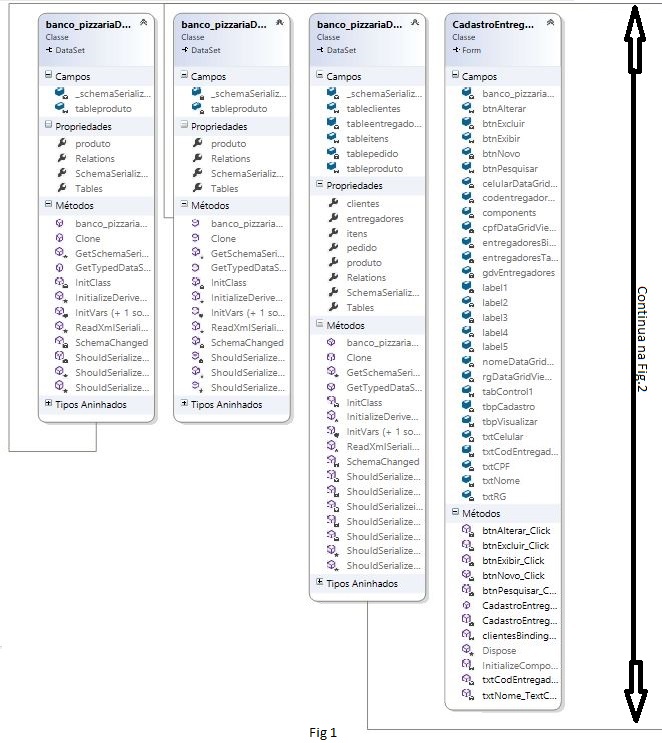
#### 3.3. DIAGRAMA DE CLASSE.

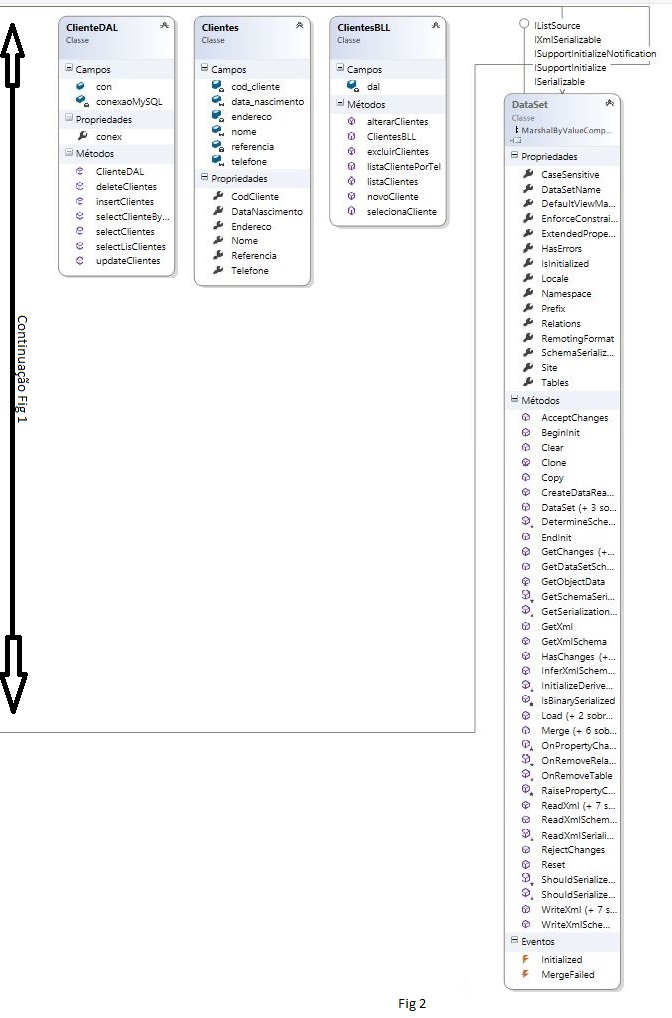
O diagrama de classes representa a estrutura do sistema, apresentando o conceito de classe e suas relações. O modelo de classes mostra um processo de abstração onde são identificados os objetos para uso do sistema. A ocorrência que tem interesse para o sistema em estudo e que se pretende descrever no seu ambiente é conhecido como Objeto onde temos a identidade e comportamento.

O comportamento de um objeto implica no modo como ele age e reage a situações externas e a identidade de um objeto tem o atributo que o distingue de todos os demais, sendo preservada quando o seu estado muda.

Segue abaixo as figuras do nosso Diagrama de Classe baseado nos estudos e implementação do sistema de Controle de entrega de Pizzas.







#### 3.4. ESTRUTURA DE DADOS.

Em junho de 2000 a Microsoft anunciou a Plataforma .NET e uma nova linguagem de programação chamada C# (se lê “C Sharp”). C# é uma linguagem fortemente tipada e orientada a objetos projetada para oferecer a melhor combinação de simplicidade, expressividade e performance.

A linguagem C# aproveita conceitos de muitas outras linguagens, mas especialmente de C++ e Java. Ela foi criada por Anders Hejlsberg (que já era famoso por ter criado o Turbo Pascal e o Delphi para a Borland) e Scott Wiltamuth.

A sintaxe utilizada pelo C# é relativamente fácil, o que diminui o tempo de aprendizado. Todos os programas desenvolvidos devem ser compilados, gerando um arquivo com a extensão DLL ou EXE. Isso torna a execução dos programas mais rápida se comparados com as linguagens de script (VBScript , JavaScript) que atualmente utilizamos na internet.

Poderíamos citar entre algumas características do C#:

* Orientada a Objetos
* Não há variáveis ou funções globais. Todos os métodos e atributos devem ser declarados dentro de classes. Atributos e métodos estáticos de classes públicas podem servir como substitutos para variáveis e métodos globais.
* Apontadores
* Em C#, apontadores só podem ser usados dentro de blocos especificamente marcados como inseguros. Programas com código inseguro precisam das permissões apropriadas para serem executados.
* Memória Gerenciada
* Em C# a memória não precisa ser explicitamente liberada. Ao invés disso ela é automaticamente gerenciada por meio do Garbage Collector (coletor de lixo), que constantemente percore a memória alocada para verificar se ela ainda é necessária, eliminando assim o risco de vazamentos de memória.
* Tipagem Forte
* C# é mais fortemente tipada do que C++: as únicas conversões implícitas por default são aquelas que são consideradas seguras, como por exemplo o armazenamento de um inteiro em um tipo de dados maior ou a conversão de um tipo derivado para um tipo base. Não há conversão implícita entre booleanos e inteiros.

A estrutura de dados define de que maneira os tipos primitivos serão organizados (FORBELLONE, 2005): por meio de lista, pilhas, filas ou árvores. Essas opções representam os conjuntos a serem manipulados por logaritmos, de diferentes formas. Esse tipo de estrutura pode sofrer alterações (MARCONDES, 2002). Na teoria, refere-se à identificação e ao desenvolvimento de modelos para a resolução de problemas. Na prática, significa criar representações concretas que podem atuar sobre modelos.

#### 3.4.1. LISTA ENCADEADA.

Um espaço de memória é reservado para os dados, quando são inseridos em uma lista encadeada. Então, o espaço ocupado por uma lista, na memória, é expresso pela quantidade de elementos contidos ali dentro. Para que se possa percorrer todos os elementos, deve-se armazenar – junto a cada um deles – um ponteiro que indique o elemento seguinte.

Acada elemento da lista é identificado como um nó. Utilizam-se os ponteiros parapercorrer todos os nós da lista. O último elemento deverá apontar para NULL (em linguagem de programação de computador, é a palavra usada para se referira um dispositivo nulo). Para a criação de uma lista, considera-se o primeiro ponteiro existente.

Uma lista vazia deve ser criada com o ponteiro indicando NULL, permitindo inserções no final. Essa lista deve ser alocada em uma posição de memória, com encadeamento para o próximo elemento.Retirar um elemento da lista é um processo um pouco mais complexo, pois podemos encontrar diferentes situações, como eliminar o primeiro elemento da lista ou um elemento intermediário. Outras funções agregadas às listas são: busca de elementos e verificação para saber se ela está vazia ou não.

#### 3.4.2. PILHA.

A pilha é uma estrutura bem simples, muitas vezes encontrada dentro do hardware.

A ideia é acessar o primeiro elemento a partir do topo da pilha. Assim, um novo elemento inserido na pilha vai direto para o topo e, logicamente, é o único que pode ser retirado. Portanto, o primeiro que sai é o último que entrou. Temos, nesse caso, o LIFO (do inglês last in, first out*,* ou último dentro, primeiro fora). Há duas operações básicas, para trabalhar com as pilhas: push (empilhar) e pop (desempilhar).

#### 3.4.3. FILAS.

Outra estrutura é a fila, que apresenta a ordem de saída dos elementos de forma diferente do que acontece em uma pilha. Na fila, o primeiro elemento que entra é o primeiro a sair: FIFO (first in, first out, ou seja, primeiro dentro, primeiro fora). É possível inserir um novo elemento no final da fila e retirar o primeiro.

#### 3.4.4. LISTA CIRCULAR.

Em uma lista circular o último elemento deverá apontar para o primeiro. Para percorrer a lista é preciso visitar todos os seus elementos, a partir de um ponteiro inicial, até chegar a ele novamente. Caso o ponteiro inicial seja NULL a lista será considerada vazia.

#### 3.4.5. LISTA DUPLAMENTE ENCADEADA.

Na lista encadeada há um ponteiro indicando o próximo elemento. Já na duplamente encadeada são dois ponteiros: um mostrando o elemento seguinte e outro, o anterior.

A partir de um elemento podemos acessar ambos os lados. Na árvore binária cada nó tem zero, um ou, no máximo, dois “filhos”. Então, é possível definir uma árvore binária como vazia ou um nó raiz com duas subárvores uma na direita (sad) e outra na esquerda (sae).

Este trabalho apresenta como proposta um projeto para pizzarias que necessitam gerenciar de forma mais organizada o setor específico de entregas de pizzas e dos demais produtos que uma pizzaria pode oferecer.

Devido ao crescimento no mercado deste tipo de serviço, surgiu a necessidade da automatização dos processos e ter um maior controle das informações que são geradas pelas atividades do setor de entrega.

O Sistema irá apoiar no controle de pedidos e entregas dos produtos, gerenciando os entregadores (motoboy), os produtos oferecidos pela pizzaria, gerar relatórios de vendas além de melhorar o atendimento de entrega da pizzaria em relação aos seus clientes, tornando mais rápido o momento de registrar o pedido e passar informações ao cliente sobre os produtos que estão à venda.

Com este sistema, quando um cliente liga para a pizzaria, basta o atendente digitar o número do telefone do mesmo, que ele lhe mostrará se o cliente já está cadastrado ou não. Se não estiver cadastrado o atendente deve cadastrá-lo antes de registrar o pedido.

Na página “Registrar pedidos” é onde será definida a escolha do cliente, o valor de cada item, o valor total e o valor de troco se for solicitado pelo mesmo. No momento em que a solicitação do cliente for cadastrada, ela será impressa. Quando o pedido estiver pronto, o usuário irá passá-lo para o status despachado onde será vinculado um funcionário para fazer a entrega.

Os funcionários que não vieram trabalhar, o status do mesmo ficará inativo, já aqueles que compareceram, ficará disponível e quando estiver entregando um pedido, seu status será ocupado. Apenas quando o entregador retornar com o dinheiro é que o pedido passará para o status finalizado.

O sistema será local sem servidor específico, tudo concentrado em um único equipamento desktop. A linguagem de programação utilizada é o C# suportada pela plataforma .NET.

#### 3.5. BRMODELO.

Foram estudados os conceitos essenciais do modelo relacional, e apresentamos a construção de modelos de entidade e relacionamento.

Depois da construção da entidade e relacionamento é necessário a transformação para o modelo relacional, com o objetivo de construir tabelas relacionadas utilizando as restrições do projeto modelado.

Não podemos esquecer dos mecanismos que servem para identificar as linhas e estabelecer relacionamentos entre tabelas que são as chaves primária e estrangeira:

* A chave primária é uma combinação de colunas onde os valores servem para diferenciar uma linha das demais em uma tabela.
* A chave estrangeira uma combinação de colunas onde os valores aparecem na chave primária de uma tabela, ou seja, a chave estrangeira permite implementar o relacionamento em um banco de dados relacional.

O BrModelo é um programa livre usado no aprendizado de modelagem de dados, é um editor, e possui duas funcionalidades:

1- Construção do modelo de entidade e relacionamento.

2 - Mapeamento para o modelo relacional de banco de dados.

Nas Figuras abaixo temos 2 exemplos baseado em nossa pizzaria:

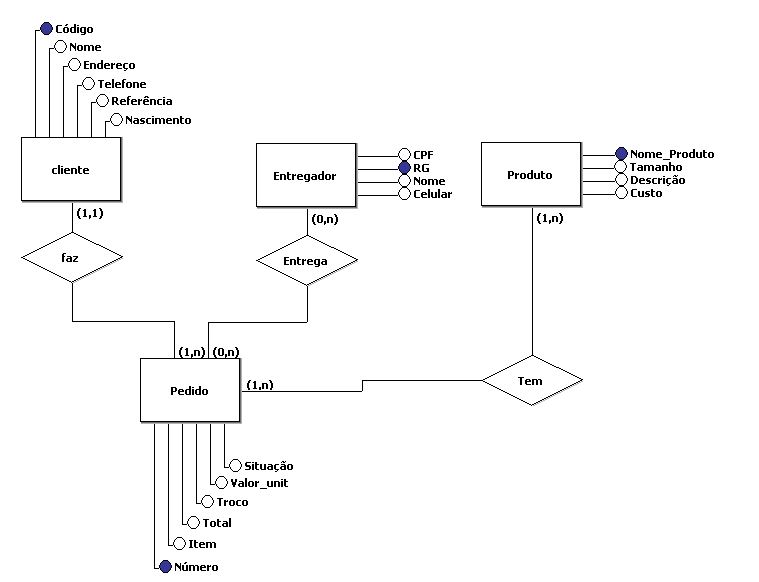
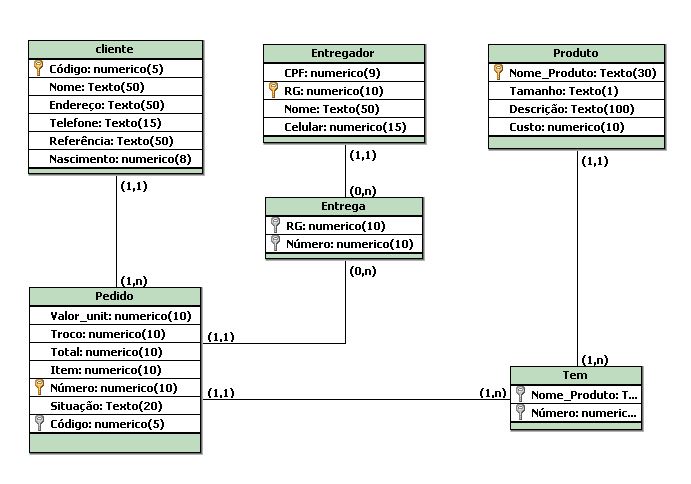


Figura 1.

Desenvolvimento de Um Modelo Conceitual.

Figura 2.

Desenvolvimento de Um Modelo Lógico.

#### 3.6. VANTAGENS E DESVANTAGENS.

Com uma rede de computadores no lugar, é fácil de compartilhar arquivos, impressoras e outros recursos do sistema. Tudo que um usuário precisa fazer é registar-se um dos computadores em rede, e ela terá de imediato acesso a todos os recursos na rede. Os usuários podem abrir arquivos instantaneamente, analisá-los e fornecer feedback, tudo sem sair de suas mesas. Entretanto, é importante conhecermos as vantagens e as desvantagens do uso das redes, e também os cuidados que devemos tomar para evitarmos os problemas.

#### 3.6.1. VANTAGENS.

#### 3.6.2. COMPARTILHAMENTO DE ARQUIVOS DE TRABALHO.

Esse é um dos recursos mais utilizados, pois permite que os usuários acessem arquivos armazenados em outros computadores interligados entre si, evitando o deslocamento de pessoas portando disquetes, como foi apresentado no exemplo anterior.

#### 3.6.3. COMPARTILHAMENTO DE PROGRAMAS.

Os computadores podem acessar programas que ficam instalados fisicamente no disco rígido de outros computadores, evitando o desperdício de espaço local, e padronizando a versão do programa em uso. Além disto, pode-se economizar no custo dos programas, pois o custo de um software para operar em rede é menor se comparado à compra de uma licença para cada computador da rede.

#### 3.6.4. COMPARTILHAMENTO DE PERIFÉRICOS.

Com a grande diversidade de mídias existentes, é inviável ter-se em cada computador um leitor de CD-RW, um ZIP Drive, um leitor de DVD ou um scanner. Quando os computadores estão ligados em rede, o próprio sistema operacional permite, de forma simples, o compartilhamento de periféricos, garantindo maior produtividade dentro da empresa.

#### 3.6.5. COMPARTILHAMENTO DE IMPRESSORAS.

O compartilhamento de impressoras é um dos mais utilizados pelos usuários de rede, pois permite que todos na rede imprimam em qualquer impressora, desde que compartilhada, assim pode-se otimizar os investimentos futuros. O sistema operacional oferece um caminho muito simples para o compartilhamento de impressoras.

#### 3.7. DESVANTAGENS.

Uma das desvantagens de computadores em rede é que, se um computador na rede e infectado por um vírus, ele pode rapidamente difundir para o resto dos computadores. É por isso que a segurança é uma preocupação tão importante para administradores de rede, e nenhum negócio deve configurar uma rede sem um plano de segurança forte no lugar.

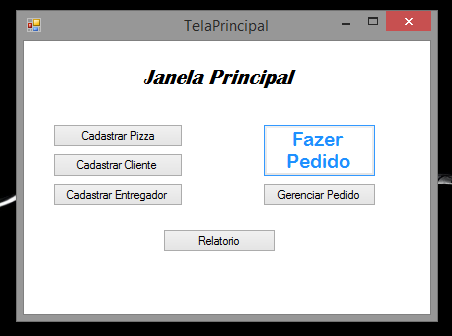
#### 3.7.1. ATAQUE A VÍRUS.

Talvez esse seja um dos piores problemas encontrados nas redes locais, pois pode danificar os softwares instalados e até o hardware, em alguns casos. Um arquivo infectado por vírus pode se espalhar pela rede em questão de minutos, fazendo todo o sistema parar e necessitar de uma manutenção global. Além da interrupção dos sistemas de redes esse também pode roubar informações sigilosas de uma empresa, ocasionando perdas financeiras e sociais.

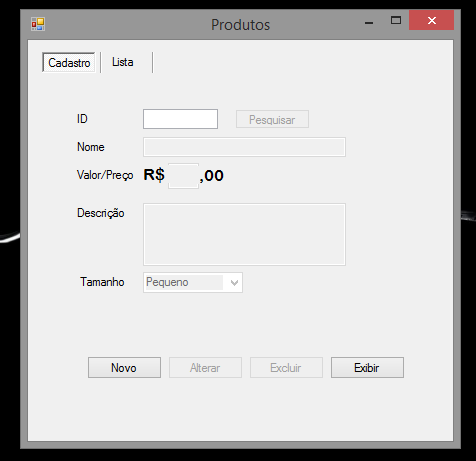
#### 3.7.2. PROBLEMAS GENERALIZADOS.

Além dos danos que os vírus causam ao computador, também podemos ter problemas com os equipamentos que centralizam as informações, tais como o HUB, switch ou os servidores de rede. Problemas ocorridos nos equipamentos que centralizam os cabos das redes (patch pannel) podem gerar muitos problemas conhecidos e desconhecidos, tais como lentidão da rede, lentidão de uma parte da rede ou até a sua parada definitiva independente da topologia utilizada. Os servidores de rede quando param comprometem os usuários de seus programas, os usuários das impressoras ou os periféricos compartilhados por ele.

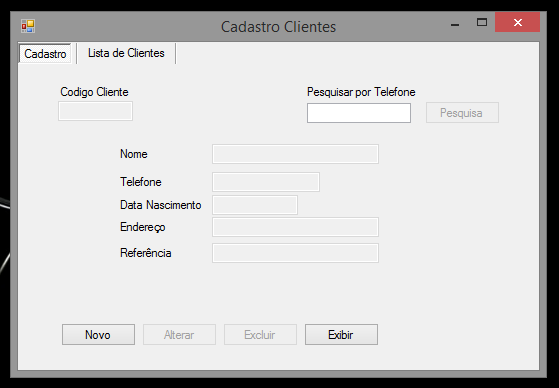
#### 3.8. PROGRAMA EM C#.

O Sistema Desenvolvido para a empresa Pizzaria “An Poparome”, sistema gerenciador de pedidos, esse sistema foi criado utilizando a Linguagem C# da Microsoft e com o gerenciador de Dados MySql da Oracle, de acordo com os requisitos levantado chegamos a esse protótipo onde ao longo poderá ser complementado com novas funcionalidades.

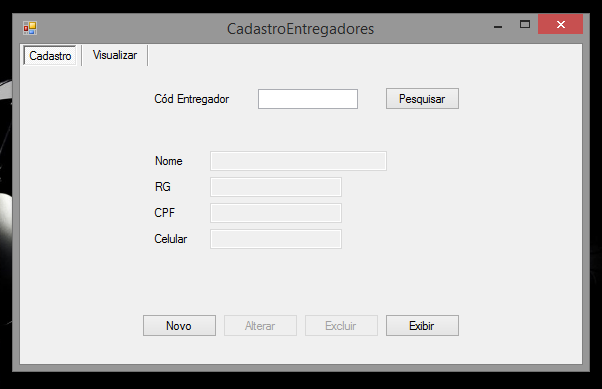
Nessa primeira tela temos o menu principal com as opções Cadastrar Pizza, Cadastrar Clientes, cadastrar entregador, Relatório, gerenciador de pedidos e fazer pedido. Simples e objetivo.



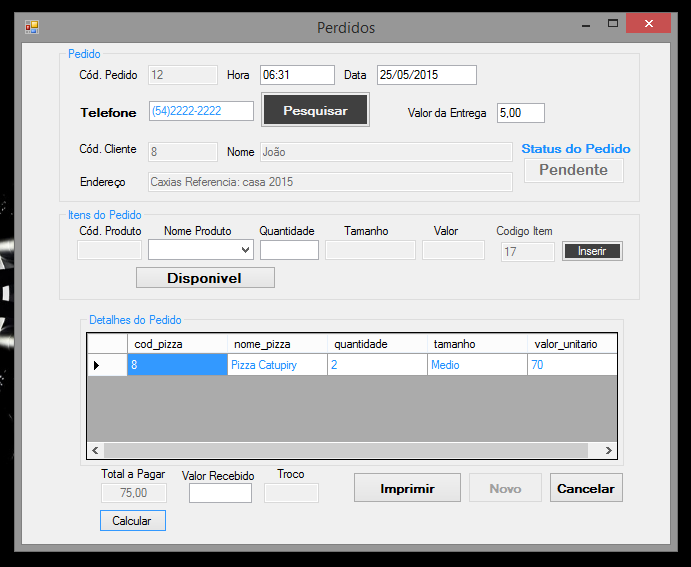
Nessa tela Produtos temos então o cadastro das Pizzas. Com duas abas permitindo a listagem dos Produtos e o Cadastro do mesmo. Com as opções de botões, novo, Alterar, Excluir e Exibir.



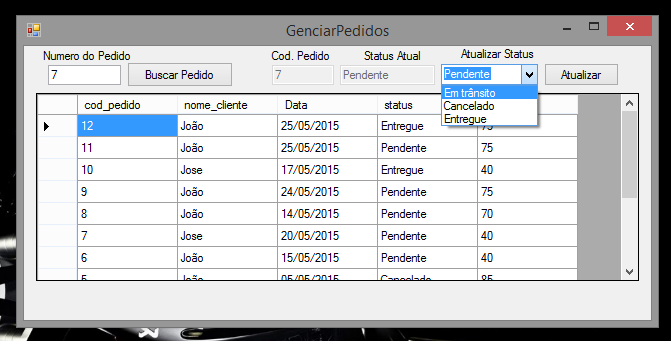
Na Tela Cadastro Clientes temos agora o cadastro dos Clientes, permitindo a Pesquisa somente por Telefone, isso garante o acesso rápido dos dados, temos a opções de Exibir, Excluir, Alterar Cadastro e Novo. Com duas abas permitindo a exibição dos Clientes cadastrados.



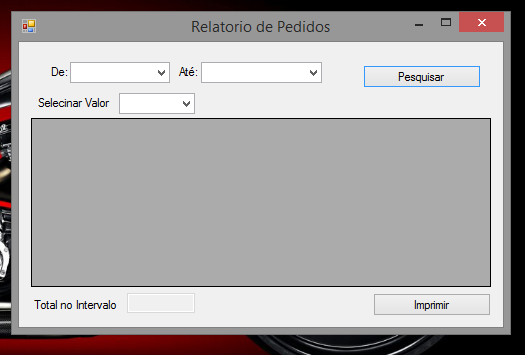
Cadastro de Entregadores uma tela com funções iguais as demais com a opção de pesquisa por Código do Entregador.



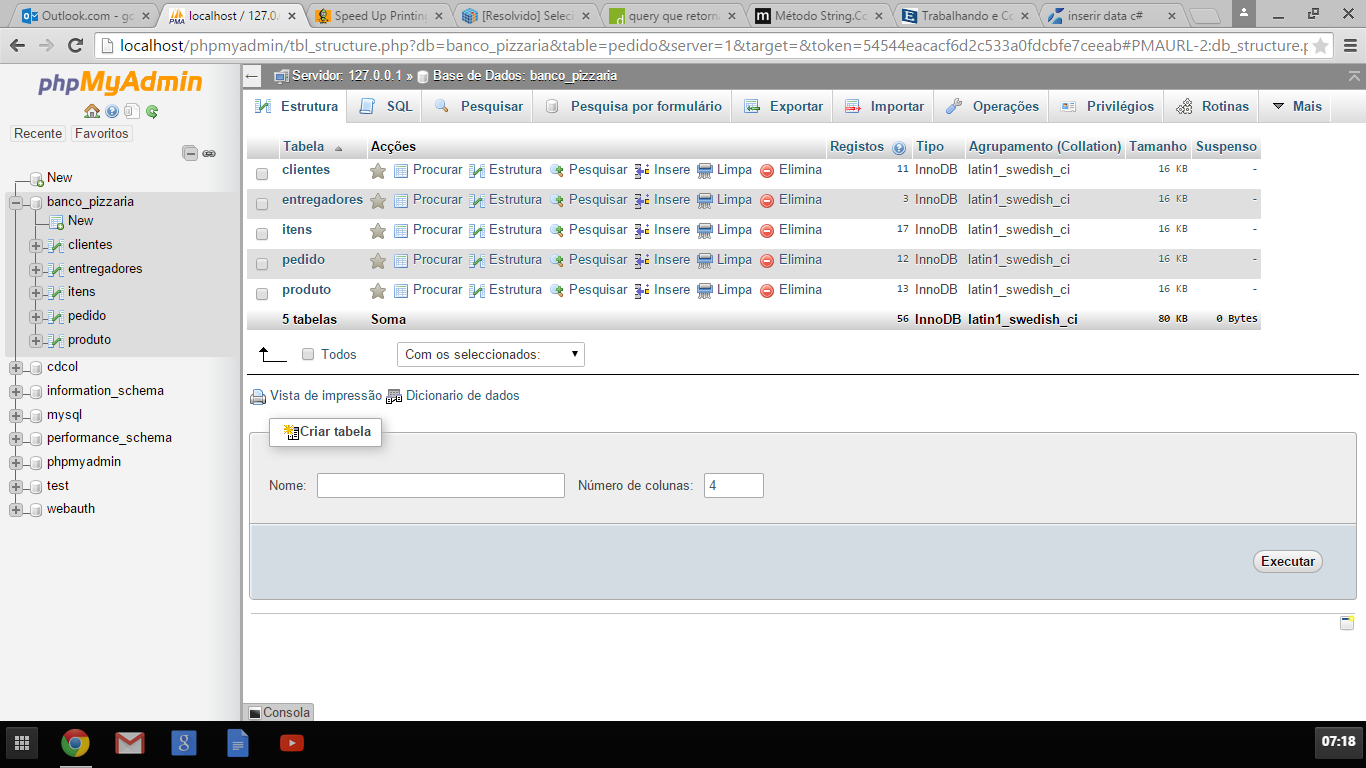
Na tela acima temos o Pedido, nessa tela que realizamos os pedidos, funções imprimir que ao mesmo tempo em que imprimi ele salva no banco, Novo nesse caso um novo pedido, cancelar pedido, calcular Total a pagar tendo a opção para calcular troco. Temos dois campos de Buscas um para encontrar o Cliente e outra para encontrar o Produto todos ligados ao Banco de Dados fazendo a relação com o Pedido, inseri itens ao meu pedido. Essa é uma tela simples, mas que permite fazer o pedido muito rápido.



Na tela de gerenciamento do Pedido, vamos gerenciar o status, qual a situação do pedido se está pendente, em transito, cancelado ou entregue.



Tela de Relatório essa tela vai nos retornar um relatório por intervalo de Tempo dos valores de entrega e valores totais dos pedidos isso permite ter uma base de ganhos, para pagamentos dos entregadores e da empresa assim podendo gerenciar melhor seus ganhos.



Gerenciador MyAdmin do MySql está é a Base do nosso projeto.

-- phpMyAdmin SQL Dump

-- version 4.3.11

-- http://www.phpmyadmin.net

-- Host: 127.0.0.1

-- Generation Time: 25-Maio-2015 às 12:20

-- Versão do servidor: 5.6.24

-- PHP Version: 5.6.8

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8 \*/;

-- Database: `banco\_pizzaria`

-- Estrutura da tabela `clientes`

CREATE TABLE IF NOT EXISTS `clientes` (

`cod\_cliente` int(4) NOT NULL,

`nome` varchar(50) NOT NULL,

`data\_nascimento` varchar(15) NOT NULL,

`telefone` varchar(15) NOT NULL,

`endereco` varchar(50) NOT NULL,

`referencia` varchar(50) NOT NULL

) ENGINE=InnoDB AUTO\_INCREMENT=15 DEFAULT CHARSET=latin1;

-- Extraindo dados da tabela `clientes`

INSERT INTO `clientes` (`cod\_cliente`, `nome`, `data\_nascimento`, `telefone`, `endereco`, `referencia`) VALUES

(2, 'Daniela Costa', '16/07/1988', '(51)9951-0724', 'Gaspar Martins', 'Ap01'),

(4, 'João', '0000-00-00 00:0', '5122224444', 'Garibaldi', 'test'),

(5, 'Doug', '0000-00-00 00:0', '11112222', 'garibaldi', 'ap01'),

(6, 'Jose', '12/06/1988', '(51)9951-0724', 'Gaspar Martins', 'Ap01'),

(8, 'João', '12/05/1989', '(54)2222-2222', 'Caxias', 'casa 2015'),

(9, 'Maria', '14/12/1985', '(54)3333-3333', 'Caxias', 'casa 154'),

(10, 'Maria', '14/12/1985', '(54)3333-4444', 'Caxias', 'casa 154'),

(11, 'Douglas', '12/05/1989', '(51)9648-4747', 'Garibaldi', 'Ap01'),

(12, 'Maria', '14/12/1985', '(54)1212-1212', 'Garibaldi', 'casa 154'),

(13, 'Jose', '12/06/1988', '(51)9951-0895', 'Gaspar Martins', 'Ap01'),

(14, 'Eduardo', '26/05/1989', '(51)9685-7412', 'Garibaldi', 'Ap206');

-- Estrutura da tabela `entregadores`

CREATE TABLE IF NOT EXISTS `entregadores` (

`cod\_entregador` int(4) NOT NULL,

`nome` varchar(50) NOT NULL,

`cpf` varchar(20) NOT NULL,

`rg` varchar(20) NOT NULL,

`celular` varchar(20) NOT NULL

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=latin1;

-- Extraindo dados da tabela `entregadores`

INSERT INTO `entregadores` (`cod\_entregador`, `nome`, `cpf`, `rg`, `celular`) VALUES

(1, 'Douglas', '0', '0', '2147483647'),

(3, 'Mario', '154878125', '15488787854', '5196857412'),

(4, 'Douglas', '0', '0', '2147483647');

-- Estrutura da tabela `itens`

CREATE TABLE IF NOT EXISTS `itens` (

`cod\_itens` int(4) NOT NULL,

`cod\_pedido` int(11) NOT NULL,

`cod\_pizza` int(11) NOT NULL,

`nome\_pizza` varchar(50) NOT NULL,

`quantidade` int(11) NOT NULL,

`tamanho` varchar(15) NOT NULL,

`valor\_unitario` decimal(10,0) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Extraindo dados da tabela `itens`

INSERT INTO `itens` (`cod\_itens`, `cod\_pedido`, `cod\_pizza`, `nome\_pizza`, `quantidade`, `tamanho`, `valor\_unitario`) VALUES

(1, 1, 1, '1', 1, '1', '1'),

(2, 2, 17, 'Pizza 4 Queijos', 2, 'Grande', '70'),

(3, 3, 20, 'calabresa', 1, 'Medio', '45'),

(4, 3, 12, 'Pizza 30', 1, 'Tamanho', '45'),

(5, 4, 15, 'Pizza Filé', 1, 'Medio', '35'),

(6, 5, 12, 'Pizza 30', 1, 'Tamanho', '45'),

(7, 5, 8, 'Pizza Catupiry', 1, 'Medio', '35'),

(8, 6, 17, 'Pizza 4 Queijos', 1, 'Grande', '35'),

(9, 7, 15, 'Pizza Filé', 1, 'Medio', '35'),

(10, 8, 15, 'Pizza Filé', 1, 'Medio', '35'),

(11, 8, 15, 'Pizza Filé', 1, 'Medio', '35'),

(12, 9, 15, 'Pizza Filé', 1, 'Medio', '35'),

(13, 9, 8, 'Pizza Catupiry', 1, 'Medio', '35'),

(14, 10, 17, 'Pizza 4 Queijos', 1, 'Grande', '35'),

(15, 11, 15, 'Pizza Filé', 1, 'Medio', '35'),

(16, 11, 17, 'Pizza 4 Queijos', 1, 'Grande', '35'),

(17, 12, 8, 'Pizza Catupiry', 2, 'Medio', '70');

-- Estrutura da tabela `pedido`

CREATE TABLE IF NOT EXISTS `pedido` (

`cod\_pedido` int(255) NOT NULL DEFAULT '0',

`cod\_cliente` int(4) NOT NULL,

`endereco` varchar(50) NOT NULL,

`nome\_cliente` varchar(50) NOT NULL,

`telefone` varchar(15) NOT NULL,

`valor\_total` decimal(10,0) NOT NULL,

`data` varchar(10) NOT NULL,

`hora` varchar(5) NOT NULL,

`valor\_entrega` double NOT NULL,

`status` varchar(20) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Extraindo dados da tabela `pedido`

INSERT INTO `pedido` (`cod\_pedido`, `cod\_cliente`, `endereco`, `nome\_cliente`, `telefone`, `valor\_total`, `data`, `hora`, `valor\_entrega`, `status`) VALUES

(1, 0, '', '', '', '0', '0000-00-00', '', 0, ''),

(2, 8, 'Caxias Referencia: casa 2015', 'João', '(54)2222-2222', '75', '12/05/2015', '05:13', 5, 'Pendente'),

(3, 6, 'Gaspar Martins Referencia: Ap01', 'Jose', '(51)9951-0724', '50', '10/05/2015', '05:16', 5, 'Pendente'),

(4, 6, 'Gaspar Martins Referencia: Ap01', 'Jose', '(51)9951-0724', '40', '18/05/2015', '05:27', 5, 'Pendente'),

(5, 8, 'Caxias Referencia: casa 2015', 'João', '(54)2222-2222', '85', '05/05/2015', '05:34', 5, 'Cancelado'),

(6, 8, 'Caxias Referencia: casa 2015', 'João', '(54)2222-2222', '40', '15/05/2015', '05:44', 5, 'Pendente'),

(7, 6, 'Gaspar Martins Referencia: Ap01', 'Jose', '(51)9951-0724', '40', '20/05/2015', '05:45', 5, 'Pendente'),

(8, 8, 'Caxias Referencia: casa 2015', 'João', '(54)2222-2222', '70', '14/05/2015', '05:55', 0, 'Pendente'),

(9, 8, 'Caxias Referencia: casa 2015', 'João', '(54)2222-2222', '75', '24/05/2015', '06:12', 5, 'Pendente'),

(10, 6, 'Gaspar Martins Referencia: Ap01', 'Jose', '(51)9951-0724', '40', '17/05/2015', '06:15', 5, 'Entregue'),

(11, 8, 'Caxias Referencia: casa 2015', 'João', '(54)2222-2222', '75', '25/05/2015', '06:24', 5, 'Pendente'),

(12, 8, 'Caxias Referencia: casa 2015', 'João', '(54)2222-2222', '75', '25/05/2015', '06:31', 5, 'Entregue');

-- Estrutura da tabela `produto`

CREATE TABLE IF NOT EXISTS `produto` (

`cod\_pizza` int(4) NOT NULL,

`nome` varchar(50) NOT NULL,

`descricao` varchar(200) NOT NULL,

`tamanho` varchar(50) NOT NULL,

`valor` decimal(10,0) NOT NULL

) ENGINE=InnoDB AUTO\_INCREMENT=21 DEFAULT CHARSET=latin1;

-- Extraindo dados da tabela `produto`

INSERT INTO `produto` (`cod\_pizza`, `nome`, `descricao`, `tamanho`, `valor`) VALUES

(5, 'Pizza Filé', 'jasjansjansjnasj;\r\nkmaskmaskmak;\r\nkaskmaksma;\r\nkmaskamsk;', 'Medio', '35'),

(7, 'pizza 10', 'kasjansjansja;\r\nkmskamsk;\r\n\r\nkamskmask;', 'Medio', '35'),

(8, 'Pizza Catupiry', 'Molho de Tomate;\r\nQueijo Musarela;\r\nCatupiry;\r\nOregano;', 'Medio', '35'),

(9, 'Pizza Filé', 'jasjansjansjnasj;\r\nkmaskmaskmak;\r\nkaskmaksma;\r\nkmaskamsk;', 'Medio', '35'),

(11, 'Pizza 4 Queijos', 'Molho de Tomate;\r\n4 tipos de queijos;\r\norégano;\r\n', 'Medio', '35'),

(12, 'Pizza 30', 'jsnjasjasnjasnjansjas;\r\nnsanajsnjansja;\r\njansjnasjsajnasj;\r\nkasasn', 'Tamanho', '45'),

(14, 'Pizza de Calabresa', 'jasnjasnjas;\r\nkamskmasasmas;\r\nkasmka;\r\nlas,las,las,las', 'Familia', '35'),

(15, 'Pizza Filé', 'jasjansjansjnasj;\r\nkmaskmaskmak;\r\nkaskmaksma;\r\nkmaskamsk;', 'Medio', '35'),

(16, 'Pizza de Milho', 'Molho de Tomate;\r\nMussarela;\r\nMilho;\r\nCatupiry;\r\nOregano;', 'Familia', '55'),

(17, 'Pizza 4 Queijos', 'jasjnasjnajsnjasnjas;\r\naksansjnasjnasjnas;\r\nkamskmaskmaskas;', 'Grande', '35'),

(18, 'Pizza de Mussarela', 'Molho de Tomate;\r\nQueijo Musarela;\r\nOregano;', 'Grande', '45'),

(19, 'pizza 10', 'kasjansjansja;\r\nkmskamsk;\r\n\r\nkamskmask;', 'Medio', '35'),

(20, 'calabresa', 'calabresa', 'Medio', '45');

-- Indexes for dumped tables

-- Indexes for table `clientes`

ALTER TABLE `clientes`

ADD PRIMARY KEY (`cod\_cliente`);

-- Indexes for table `entregadores`

ALTER TABLE `entregadores`

ADD PRIMARY KEY (`cod\_entregador`);

-- Indexes for table `itens`

ALTER TABLE `itens`

ADD PRIMARY KEY (`cod\_itens`);

-- Indexes for table `pedido`

ALTER TABLE `pedido`

ADD PRIMARY KEY (`cod\_pedido`);

-- Indexes for table `produto`

ALTER TABLE `produto`

ADD PRIMARY KEY (`cod\_pizza`);

-- AUTO\_INCREMENT for dumped tables

-- AUTO\_INCREMENT for table `clientes`

ALTER TABLE `clientes`

MODIFY `cod\_cliente` int(4) NOT NULL AUTO\_INCREMENT,AUTO\_INCREMENT=15;

-- AUTO\_INCREMENT for table `entregadores`

ALTER TABLE `entregadores`

MODIFY `cod\_entregador` int(4) NOT NULL AUTO\_INCREMENT,AUTO\_INCREMENT=5;

-- AUTO\_INCREMENT for table `produto`

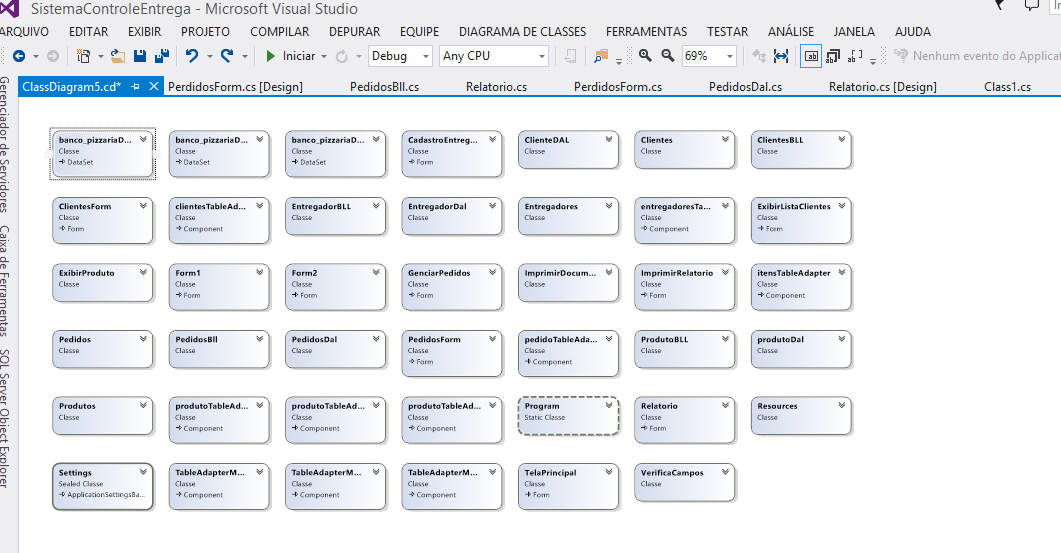
ALTER TABLE `produto`

MODIFY `cod\_pizza` int(4) NOT NULL AUTO\_INCREMENT,AUTO\_INCREMENT=21;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;



Nossas Classes usadas nesse projeto, foi trabalhado em três camadas uma de Dados, uma de Variáveis e uma de Transição de Dados fazendo a união das mesmas por traz dos Formulários. Um pouco dos Códigos aqui incorporados.

Classe ClientesDall.cs // Essa Classe gerencia a Base de Dados

class ClienteDAL

{

String conexaoMySQL = "";

public MySqlConnection con;

public ClienteDAL()

{

//Fazendo a Conexão com o Banco de Dados e Salvando na Variavel conexaoMySQL

conexaoMySQL = "Server=localHost;userid=root;password=;database=banco\_pizzaria";

}

//Seleciona a Tabela "Clientes" do Banco de Dados MySQL

public List<Clientes> selectLisClientes(){

try {

using(MySqlConnection conn = new MySqlConnection(conexaoMySQL))

{

using (MySqlCommand command = new MySqlCommand("Select \* from clientes", conn))

{

conn.Open();

List<Clientes>listaClientes = new List<Clientes>();

using (MySqlDataReader dr = command.ExecuteReader())

{

while (dr.Read()) {

Clientes cliente = new Clientes();

cliente.CodCliente = (int) dr["cod\_cliente"];

cliente.Nome = (string) dr["nome"];

cliente.DataNascimento = (string) dr["data\_nascimento"];

cliente.Telefone = (string) dr["telefone"];

cliente.Endereco = (string)dr["endereco"];

cliente.Referencia = (string)dr["referencia"];

listaClientes.Add(cliente);

}

}

return listaClientes;

}

}

}

catch (Exception ex) {

throw new Exception("Erro ao acessar Tabela Clientes" + ex.Message);

}

}

// Select Tabela Clientes

public DataTable selectClientes()

{

try

{

String sql = "SELECT \* FROM clientes";

con = new MySqlConnection(conexaoMySQL);

MySqlCommand cmd = new MySqlCommand(sql, con);

MySqlDataAdapter da = new MySqlDataAdapter();

da.SelectCommand = cmd;

DataTable dt = new DataTable();

da.Fill(dt);

return dt;

}

catch (Exception ex)

{

throw ex;

}

}

//Fazendo SELEC em Tabela

public Clientes selectClienteByTel(string telefone)

{

try

{

String sql = "SELECT \* FROM clientes WHERE telefone=@telefone";

con = new MySqlConnection (conexaoMySQL);

MySqlCommand cmd = new MySqlCommand(sql, con);

cmd.Parameters.AddWithValue("@telefone", telefone);

con.Open();

MySqlDataReader dr;

Clientes cliente = new Clientes();

dr = cmd.ExecuteReader(CommandBehavior.CloseConnection);

while (dr.Read())

{

cliente.CodCliente = Convert.ToInt32(dr["cod\_cliente"]);

cliente.Nome = dr["nome"].ToString();

cliente.DataNascimento = dr["data\_nascimento"].ToString();

cliente.Telefone = dr["telefone"].ToString();

cliente.Endereco = dr["endereco"].ToString();

cliente.Referencia = dr["referencia"].ToString();

}

return cliente;

}

catch (Exception ex)

{

throw ex;

}

}

//Inserindo Dados Tabela Produto

public void insertClientes(Clientes cli)

{

try {

con = new MySqlConnection(conexaoMySQL);

con.Open();

string inserir = "INSERT INTO clientes(nome,data\_nascimento,telefone,endereco,referencia)values('" + cli.Nome + "','" + cli.DataNascimento + "','" + cli.Telefone +"','"+ cli.Endereco +"','"+ cli.Referencia +"')";

MySqlCommand comandos = new MySqlCommand(inserir, con);

comandos.ExecuteNonQuery();

con.Close();

}

catch (Exception ex)

{

throw new Exception("Erro de Comandos ->Inserindo dados<- :" + ex.Message);

}

}

//Atualizando a Tabela Cliente

public void updateClientes(Clientes cliente)

{

try

{

String sql = "UPDATE clientes SET nome=@nome, data\_nascimento=@data\_nascimento, telefone=@telefone,endereco=@endereco, referencia=@referencia WHERE cod\_cliente =@cod\_cliente";

con = new MySqlConnection(conexaoMySQL);

MySqlCommand cmd = new MySqlCommand(sql, con);

cmd.Parameters.AddWithValue("@cod\_cliente", cliente.CodCliente);

cmd.Parameters.AddWithValue("@nome", cliente.Nome);

cmd.Parameters.AddWithValue("@data\_nascimento", cliente.DataNascimento);

cmd.Parameters.AddWithValue("@telefone", cliente.Telefone);

cmd.Parameters.AddWithValue("@endereco", cliente.Endereco);

cmd.Parameters.AddWithValue("@referencia", cliente.Referencia);

con.Open();//abri conexão com banco

cmd.ExecuteNonQuery();

}

catch (Exception ex)

{

throw ex;

}

finally

{

con.Close();

}

}

//Deletando Clientes --> Aqui Tudo Ok

public void deleteClientes(Clientes cliente)

{

try

{

String sql = "DELETE FROM clientes WHERE cod\_cliente = @cod\_cliente ";

MySqlConnection con = new MySqlConnection(conexaoMySQL);

MySqlCommand cmd = new MySqlCommand(sql, con);

cmd.Parameters.AddWithValue("@cod\_cliente", cliente.CodCliente);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}catch (Exception ex)

{

throw new Exception("Erro de Comandos-> Delet: " + ex.Message);

}

}

public MySqlConnection conex{ get; set; }

}

}

Classe Clientes.cs // Aqui as Variável

class Clientes

{

int cod\_cliente;

string nome;

string data\_nascimento;

string telefone;

string endereco;

string referencia;

public int CodCliente

{

get { return cod\_cliente; }

set { cod\_cliente = value; }

}

public String Nome

{

get { return nome; }

set { nome = value; }

}

public String DataNascimento

{

get { return data\_nascimento; }

set { data\_nascimento = value; }

}

public string Telefone

{

get { return telefone; }

set { telefone = value; }

}

public string Endereco

{

get { return endereco; }

set { endereco = value; }

}

public string Referencia

{

get { return referencia; }

set { referencia = value; }

}

}

ClasseBll.cs // Faz a ligação

class ClientesBLL

{

ClienteDAL dal = null;

public ClientesBLL()

{

}

//

public DataTable selecionaCliente()

{

DataTable tb = new DataTable();

try { dal = new ClienteDAL();

tb = dal.selectClientes();

}

catch(Exception ex)

{

throw ex;

}

return tb;

}

//

public void novoCliente(Clientes cliente)

{

try

{

dal = new ClienteDAL();

dal.insertClientes(cliente);

}

catch (Exception ex)

{

throw ex;

}

}

//

public void alterarClientes(Clientes cliente)

{

try

{

dal = new ClienteDAL();

dal.updateClientes(cliente);

}

catch (Exception ex)

{

throw ex;

}

}

//

public void excluirClientes(Clientes cliente)

{

try

{

dal = new ClienteDAL();

dal.deleteClientes(cliente);

}

catch (Exception ex)

{

throw ex;

}

}

//

public List<Clientes> listaClientes()

{

try

{

dal = new ClienteDAL();

return dal.selectLisClientes();

}

catch (Exception ex)

{

throw ex;

}

}

//

public Clientes listaClientePorTel(string telefone)

{

try

{

dal = new ClienteDAL();

return dal.selectClienteByTel(telefone);

}

catch (Exception ex)

{

throw ex;

}

}

//Fim

}

}

#### 4. CONCLUSÃO.

Este trabalho tem por finalidade mostrar a implantação de qualquer sistema onde diversos obstáculos são encontrados, portanto é primordial que as empresas desenvolvam as estratégias de inserção no ambiente virtual com cautela e de forma planejada. Independente do sistema utilizado ocorre impacto no ambiente de trabalho, que com a passar do tempo acaba sendo modelado de acordo com o perfil da empresa. Com a implantação desse sistema de informações trouxe um novo caminho inexplorado pela empresa, possibilitou uma melhora significativa no seu atendimento bem como tornou o processo ágil. Conclui-se que independente do ramo de atividade ou o setor que uma empresa atue, o importante é estar sempre atualizado com as inovações tecnológicas, sendo possível se manter no mercado atual, descobrindo rumos alternativos, quebrando velhos paradigmas e diminuindo as barreiras entre a empresa e os clientes.

Este trabalho proporcionou o conhecimento da linguagem de programação C# integrado ao Banco de Dados MYSQL. Utilizando os programas Astah para gerar os diagramas de Caso de Uso e os Diagramas de Classe e também fazendo modelagens usando o BRModelo.

Todo o conteúdo apresentado neste trabalho mostra que podemos aprender e usar este conhecimento gerenciar e armazenar nossos dados e gerar informações.

#### 5. REFERÊNCIA.

**NISHIMURA,** Roberto Yukio. **Banco de Dados I,** São Paulo: Editora Person Education do Brasil, 2013.

**TANAKA,** Simone Sawasaki. **Análise de Sistemas I.** São Paulo: Editora Person Education do Brasil, 2013.

**RAMALHO**, José Antônio. Oracle. **Aprenda a criar bancos de dados, tabelas,**

**Índices, visões e outros objetos**. São Paulo: Breckley Brasil, 2002.

ESTRUTURA, Dados e Algoritmos. Disponível em:

**<http://www.cos.ufrj.br/~rfarias/cos121/aula\_10.html>** Acessado em: março/2015.

ESTRUTURA, Dados e Algoritmos. Disponível em:

**<http://www.cos.ufrj.br/~rfarias/cos121/aula\_11.html>** Acessado em: abril/2015.

DESCOMPLICADA, Estrutura de Dados. Disponível em:

**<https://programacaodescomplicada.wordpress.com/>** Acessado em: abril/2015.

LINGUAGENS de Programação. Disponível em:

**<http://pt.wikipedia.org/wiki/Linguagemdeprograma>** Acessado em: maio/2015.

BRMODELO: um software free para modelagem de dados. Disponível em:

**<http://www.devmedia.com.br/brmodelo-um-software-free-para-modelagem-de-dados/2192>** Acessado em: maio/2015.

Vídeos:

< https://www.youtube.com/watch?v=Hn\_0Jb2f7hY>

< https://www.youtube.com/watch?v=TM24PgenDBk>

< https://www.youtube.com/watch?v=KjdOftXpK8M >

<https://www.youtube.com/watch?v=lYfKo6dCx6U>

<https://www.youtube.com/watch?v=wrUISU0E7yo>

<https://www.youtube.com/watch?v=TXhNFQ1AUGc>

<https://www.youtube.com/watch?v=JqDYDjqbQlQ>

<https://www.youtube.com/watch?v=lLC6LqpSkBY>

<https://www.youtube.com/watch?v=2zz\_BWz-7Co>