JHONATHA HENRIQUE DIAS DA SILVA

SISTEMA CONTROLE DE VENDA E ESTOQUE PARA RESTAURANTES

Assis-SP
2016

JHONATHA HENRIQUE DIAS DA SILVA

SISTEMA CONTROLE DE VENDA E ESTOQUE PARA RESTAURANTES

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Análise e Desenvolvimento de Sistemas.

Orientando(a): Jhonatha Henrique Dias da Silva
Orientador(a): Dr. Almir Rogério Camolesi

Assis-SP
2016

FICHA CATALOGRÁFICA

DIAS DA SILVA, Jhonatha Henrique.
Sistema Controle de Venda e Estoque para Restaurantes / Jhonatha Henrique Dias da Silva. Fundação Educacional do Município de Assis –FEMA – Assis, 2016.
45p.

1. Visual Studio. 2. C# Asp.Net, 3. MVC

CDD: 001.61
Biblioteca da FEMA

SISTEMA CONTROLE DE VENDA E ESTOQUE PARA RESTAURANTES

JHONATHA HENRIQUE DIAS DA SILVA

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Análise e Desenvolvimento de Sistemas, analisado pela seguinte banca examinadora:

Orientador: Dr. Almir Rogério Camolesi

Examinador: Esp. Domingos de Carvalho Villela Junior

Assis-SP
2016

DEDICATÓRIA

Dedico este trabalho todos meus familiares e amigos que sempre me apoiou nesta etapa da vida, e a todos que me ajudaram nesses três anos e aos professores que me ajudaram diretamente ou indiretamente principalmente ao Prof. Almir Rogério Camolesi que me ajudou muito e meu deu força para continuar.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter proporcionado a possibilidade para eu ter feito este curso que por fim acho um bom curso.

Agradeço aos meus familiares e amigos, pelo apoio o tempo todo durante este curso, que nunca deixaram faltar nada para mim principalmente meu pai que sempre me incentivou meus estudos.

Aos professores pela orientação durante o curso, que me ajudaram a concluir mais esta etapa na minha vida.

E por fim, o Prof. Almir Rogério Camolesi por me orientar não somente no meu trabalho mais também nas aulas em sala me ajudou muito sempre dando bons conselhos e incentivando cada vez estudar mais.

RESUMO

Com o aumento da competitividade das empresas no ramo alimentício surge a necessidade de desenvolver métodos para controle de vendas e estoque em restaurantes, entretanto os custos e desperdícios são relevantes para cada uma delas.

Assim fica muito dificultoso fazer um controle manualmente por normalmente nem sempre o proprietário estar no estabelecimento, sendo que o sistema terá um controle de acesso por usuário será mais fácil fazer um controle de fluxo de caixa tudo que entrar e sair do restaurante devem ser lançados ao sistema e com emissões de relatório o gerente terá um controle total do que esta acontecendo no estabelecimento.

Palavras-chave: Controle financeiro, restaurante;

ABSTRACT

With the increase of the competitiveness of enterprises in the food branch arises the need to develop methods to pay for productions at restaurants, however the costs and waste are relevant to each of them.

So it is very difficult to make a control manually by normally not always the owner be the establishment, being that the system will have a per-user access control will be easier to make a control of all cash flow in and out of the restaurant should be thrown into the system and report emissions the Manager will have a total control of what's going on in the establishment.

Keywords: Financial Control, restaurant;

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Com o aumento da competitividade das empresas no ramo alimentício surge a necessidade de desenvolver métodos para custear produções em restaurantes, entretanto os custos e desperdícios são relevantes para cada uma delas. Por exemplo, um restaurante à la carte possui cardápios com suas respectivas porções com as quantidade definida, isto faz com que não haja desperdícios por parte do restaurante assim não excedendo a produção, nesta modalidade mesmo que haja sobra no prato do cliente o suposto prejuízo será pago pelo mesmo.

Por outro lado em um restaurante “Buffet” no qual o cliente pagou por quilo ou preço fixo acorre o risco de sobras devido à variação no número de clientes que frequentam este tipo de comercio. (ISABELA, 2002)

A importância do desenvolvimento do sistema é proporcionar um melhor gerenciamento controlando estoque e vendas.

O problema de fazer isto manualmente é o tempo gasto e a eficiência de poder controlar a empresa.

1.1 OBJETIVO

Desenvolver um software com o intuito controlar venda e estoque de restaurantes, assim minimizando os prejuízos, tendo um maior controle de produtos e vendas, tentando maximizar os lucros da empresa.

1.2 PUBLICO-ALVO

Empresas do ramo alimentício, especificamente restaurantes e churrascarias, Buffet e à la carte.

1.3 JUSTIFICATIVA

Empresas do ramo de restaurante, churrascaria, Buffet e à la carte necessitam de novos métodos para a organização por hoje em dia fica muito difícil controlar o custo/benefícios da empresa utilizando-se de recurso manual, então através desse sistema ficaria melhor o gerenciamento do negócio.

1.4 ESTRUTURAS DO TRABALHO

1. Introdução- Será descrito os motivos da escolha do software e dos benefícios que ajudará a empresa.

2. Tecnologia Asp. NET MVC.

3. Modelagem do sistema. (Diagramas de caso de Uso, Classes, Sequência e Estados).

4. Desenvolvimento do Trabalho- Nesse tópico conterá todo o desenvolvimento do trabalho, vários tópicos estarão nele para expor todo o trabalho que foi feito. É a essência do trabalho.

5. Conclusão do Trabalho- Conterá informações que finalizarão o trabalho, expondo o que foi proveitoso e de futuras mudanças no sistema.

6. Referências Bibliográficas- Tudo que foi utilizado para escrever o trabalho, como links de internet e páginas de livros.

2 TECNOLOGIAS UTILIZADAS PARA O DESENVOLVIMENTO DO PROJETO

Para o desenvolvimento do sistema proposto, foi utilizada a tecnologia UML (Linguagem de Modelagem Unificada) para a análise e validação dos requisitos, onde toda a parte gráfica foi criada com auxílio da ferramenta Astah[1]. Para o desenvolvimento DER (Diagrama Entidade Relacionamento) foi utilizado a ferramenta DBDesigner[2].

A implementação do sistema foi desenvolvida utilizando tecnologia C# (C-Sharp) com a plataforma para desenvolvimento o Visual Studio[3] e o Microsoft SQL Server[4] como sistema de gerenciamento de banco de dados.

2.1 *UML – UNIFIED MODELING LANGUAGE*

A linguagem de modelagem escolhida para esta ferramenta de análise é a UML (UNIFIED MODELING LANGUAGE)[5], que é muito utilizada atualmente para a modelagem de linguagens visuais de software baseadas em orientação a objetos (OO), caracterizada por ser uma linguagem de modelagem de imensurável auxilio para que engenheiros de softwares realizem seus projetos organizadamente.

Serão utilizadas as ferramentas para criação da modelagem tais como (Freemind[6], Astah), sendo de fácil acesso e uso para criação de mapa mental e diagramas.

---

[1] http://www.astah.net/

[2] http://www.devmedia.com.br/dbdesigner

[3] https://www.visualstudio.com/downloads/download-visual-studio-vs

[4] http://www.microsoft.com/pt-br/download/details.aspx?id=29062

[5] Para Acesso: Livro: UML2 uma abordagem prática 2ª Edição**.**

[6] https://pt.wikipedia.org/wiki/Freemind

2.2 LINGUAGEM DE PROGRAMAÇÃO C#. .NET FRAMEWORK E ASP.NET

C# (C-Sharp)[7] é uma linguagem de programação visual dirigida por eventos e totalmente orientada a objetos criada pela Microsoft, por uma equipe liderada pelo engenheiro Anders Hejlsberg, baseada na arquitetura da plataforma .NET, que por sua vez, foi criada para servir de base para todas as suas soluções da Microsoft. A plataforma .NET foi projetada para trabalhar com diversas linguagens de programação com o compartilhamento de bibliotecas. (DEITEL, 2003)

Segundo PROVENCIO E RECIO (2008) o .NET Framework é uma estrutura que suporta múltiplas linguagens de programação, cada uma com suas próprias características e simplifica o processo de desenvolvimento, oferecendo ambiente distribuído e permitindo a criação de aplicações escaláveis e robustas. Os componentes do ambiente .NET são:

- Linguagens de Compilação.
- Bibliotecas de classes de .NET.
- CLR (*COMMON LANGUAGE RUNTIME*).

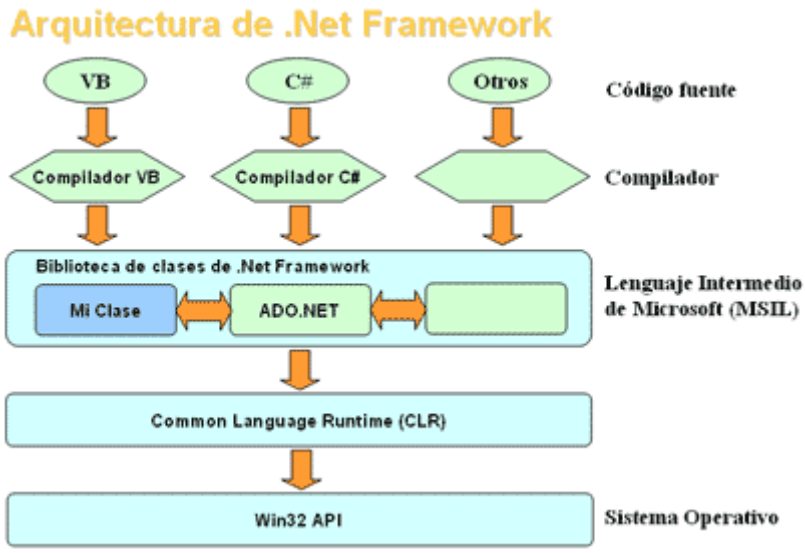

Figura 1- Arquitetura de .Net Framework

[7] https://msdn.microsoft.com/pt-br/library/kx37x362.aspx

AspNet[8] é um ambiente para desenvolvimento de aplicações Web baseado na plataforma .Net. As aplicações AspNet, podem ser desenvolvidas em qualquer linguagem de programação que pertença a plataforma .Net e que seja compatível com o ambiente de execução de aplicações desta plataforma, a CLR (*Common Language Runtime*).

2.3 SQL SERVER 2014

Ferramenta para desenvolvimento e armazenamento dos dados, será utilizada a ferramenta de banco de dados SQL Server 2014. Um **SGBD**[9] (sistema de gerenciamento de banco de dados) é um programa que gerencia os dados, geralmente utilizando uma linguagem para isso (SQL). O **banco de dados SQL Server** é basicamente o principal concorrente do Oracle e já esteve em situações piores da atual realidade. Tem como grande vantagem o fato de ser da Microsoft e se integrar nativamente com seus produtos e tecnologias, esse talvez seja o fator que o popularizou.

Atualmente o **SQL Server** conta também bom uma boa participação no mercado de web, fruto de um relacionamento mais estreito com as linguagens ASP e ASP.NET que lideram o mercado de médios e grandes projetos de internet.

2.4 VISUAL STUDIO

Visual Studio[10] é um ambiente de desenvolvimento para soluções *Desktop* e *Web* da Microsoft que oferece uma série de recursos para criar, documentar, depurar e executar programas escritos em linguagens de programação .Net e também oferece ferramentas para editar e manipular diversos tipos de arquivo.

---

[8] http://www.asp.net/get-started

[9] http://www.luis.blog.br/o-que-e-banco-de-dados.aspx

[10] https://www.visualstudio.com/

Segundo DEITEL (2003) é uma ferramenta poderosa e sofisticada para criar aplicativos de missão e comercialização crítica.

2.5 MCV5

Em rápidas palavras, é um framework da Microsoft que possibilita o desenvolvimento de aplicações web, fazendo uso do padrão arquitetural MVC (Model-View-Controller, ou Modelo-Visão-Controlador, em português). Embora o ASP.NET MVC[11] faça uso deste padrão, ele não define uma arquitetura de desenvolvimento por si só. O padrão MVC busca dividir a aplicação em responsabilidades relativas à definição de sua sigla. A parte do Modelo trata as regras de negócio, o domínio do problema, já a Visão busca levar ao usuário final informações a cerca do modelo, ou solicitar dados para registros. Desta maneira, o ASP NET MVC busca estar próximo a este padrão. Ele traz, em sua estrutura de projeto, pastas que representam cada camada do MVC, mas não traz de maneira explícita, separadas fisicamente. (EVERTON COIMBRA DE ARAÚJO, 2016).

[11] Para Acesso: Livro: ASP.NET MV5, crie aplicações web na plataforma Microsoft**.**

3 PROJETO

Neste capítulo será apresentada a análise e especificação de requisitos do sistema proposto e todo o planejamento do projeto. Desta forma, será dividido em 2 partes para melhor estruturação e facilitar o entendimento dos tópicos abordados. Na primeira parte será apresentada a análise e a especificação do sistema que será desenvolvido. Na segunda parte será ilustrado o projeto do sistema, suas etapas, cronogramas e custos.

3.1 ANÁLISE E ESPECIFICAÇÃO DO SISTEMA

A metodologia para o levantamento de necessidades adotadas foi o diálogo direto com os usuários do sistema proposto. Após alguns encontros e discussões sobre o sistema, foi elaborado um Mapa Mental, para validar as reais necessidades da implementação do Sistema. A construção do Mapa Mental foi realizada utilizando o software FreeMind[12].

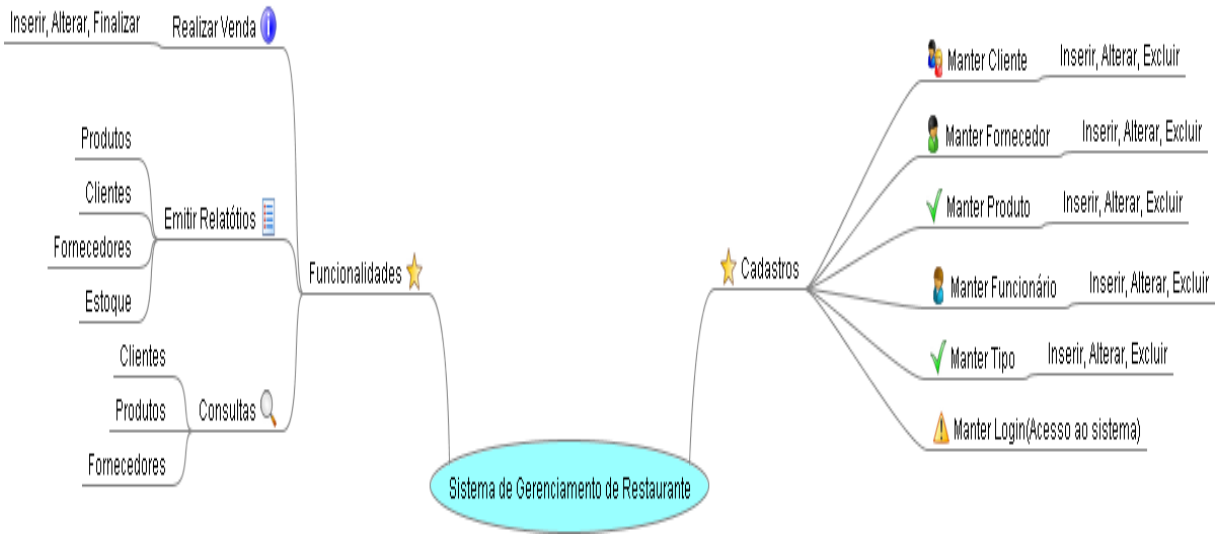

Figura 2 - Mapa Mental das funcionalidades do Sistema

[12] http://freemind.softonic.com.br/

3.1.1 DIAGRAMA DE CASO DE USO

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

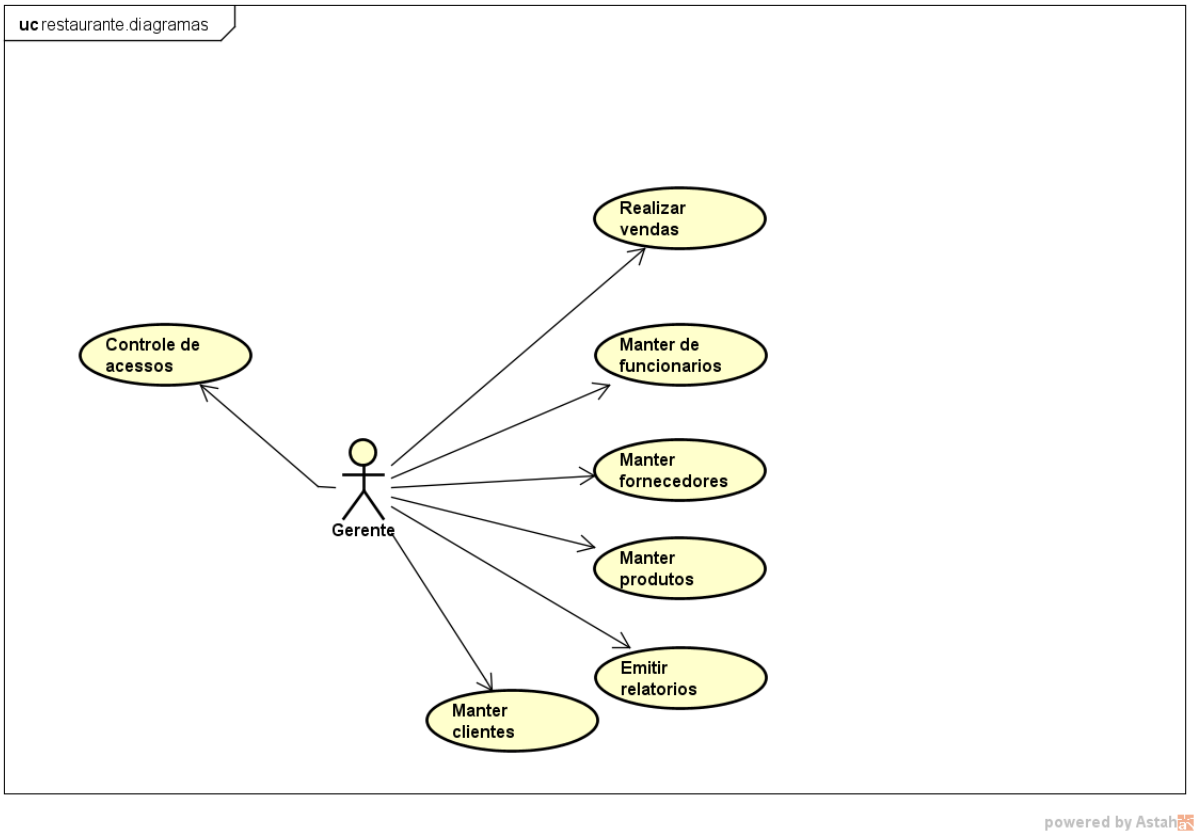

Figura 3: Diagrama de Caso de Uso 1

Neste diagrama, podemos ver todos os controles disponíveis para o usuário do sistema fazer além de realizar uma venda colocando dados do cliente, do produto, preço, quantidade e etc.

Segue a lista de eventos da primeira parte dos diagramas de caso de uso.

Nome	Descrição
Manter de clientes	Neste caso de uso, tudo relacionado ao cliente como inserir, excluir e editar será relatado no sistema.
Manter de funcionários	Neste caso de uso, tudo relacionado aos

	funcionários como inserir, editar e excluir. Sendo que somente o gerente poderá realizar os mesmo, será relatado no sistema.
Manter de fornecedores	Neste caso de uso, tudo relacionado aos fornecedores como inserir, editar e excluir sendo que somente o gerente poderá realizar os mesmo, será relatado no sistema.
Manter de produtos	Neste caso de uso, tudo relacionado aos produtos como inserir, editar e excluir. Sendo que somente o gerente poderá realizar os mesmo, será relatado no sistema.
Emitir relatórios	Neste caso de uso, tudo relacionado na emissão de relatórios sendo que somente o gerente poderá realizar os mesmo, será relatado no sistema.
Realizar vendas	Neste caso de uso, tudo relacionado ao efetuar vendas, será relatado no sistema.
Controle de acessos	Neste caso de uso, por parte de acessos ao sistema efetuados pelos usuários como restrições de login.

Tabela 1– Lista de Eventos

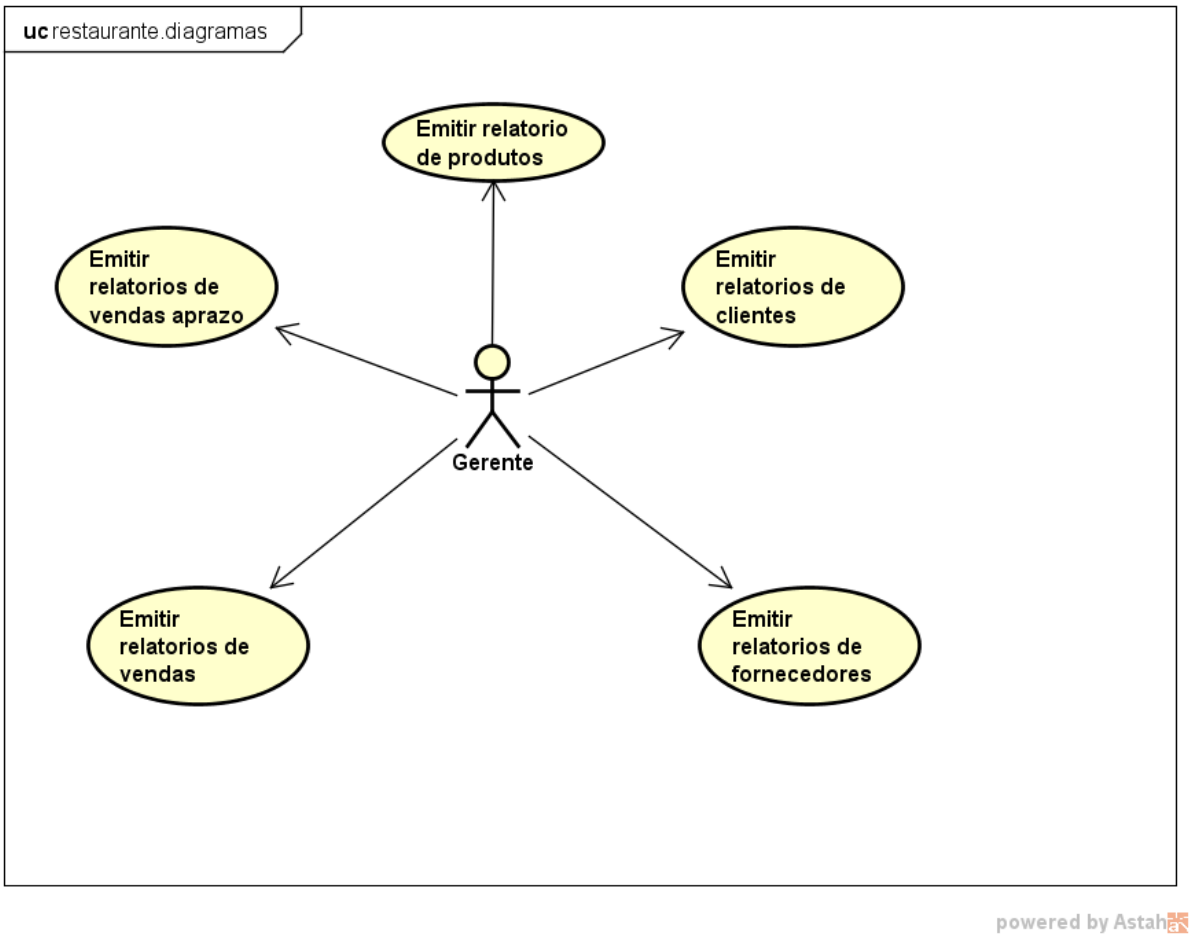

Figura 4: Diagrama de Caso de Uso 2

No segundo caso de uso, vemos os relatórios disponíveis para serem gerados. Segue a próxima lista de eventos, do próximo diagrama de caso de uso.

Nome	Descrição
Emitir relatórios de produtos	Emissões de relatórios de todos os produtos com seus respectivos atributos código, descrição, quantidade, tipo e etc.
Emitir relatórios de vendas a prazo	Emissão de relatórios das vendas realizadas a prazo sendo possível fazer um melhor controle de suas vendas fiado.
Emitir relatórios de fornecedores	Emissão de relatórios contendo dados dos fornecedores.

Emitir relatórios de clientes	Relatórios do cliente melhores conhecimento de um publico alvo.
Emitir relatórios de vendas	Emissão de relatórios de vendas totais sendo seus status (fechada ou aberta).

Tabela 2 - Lista de Eventos

3.1.2 DIAGRAMA DE CLASSE

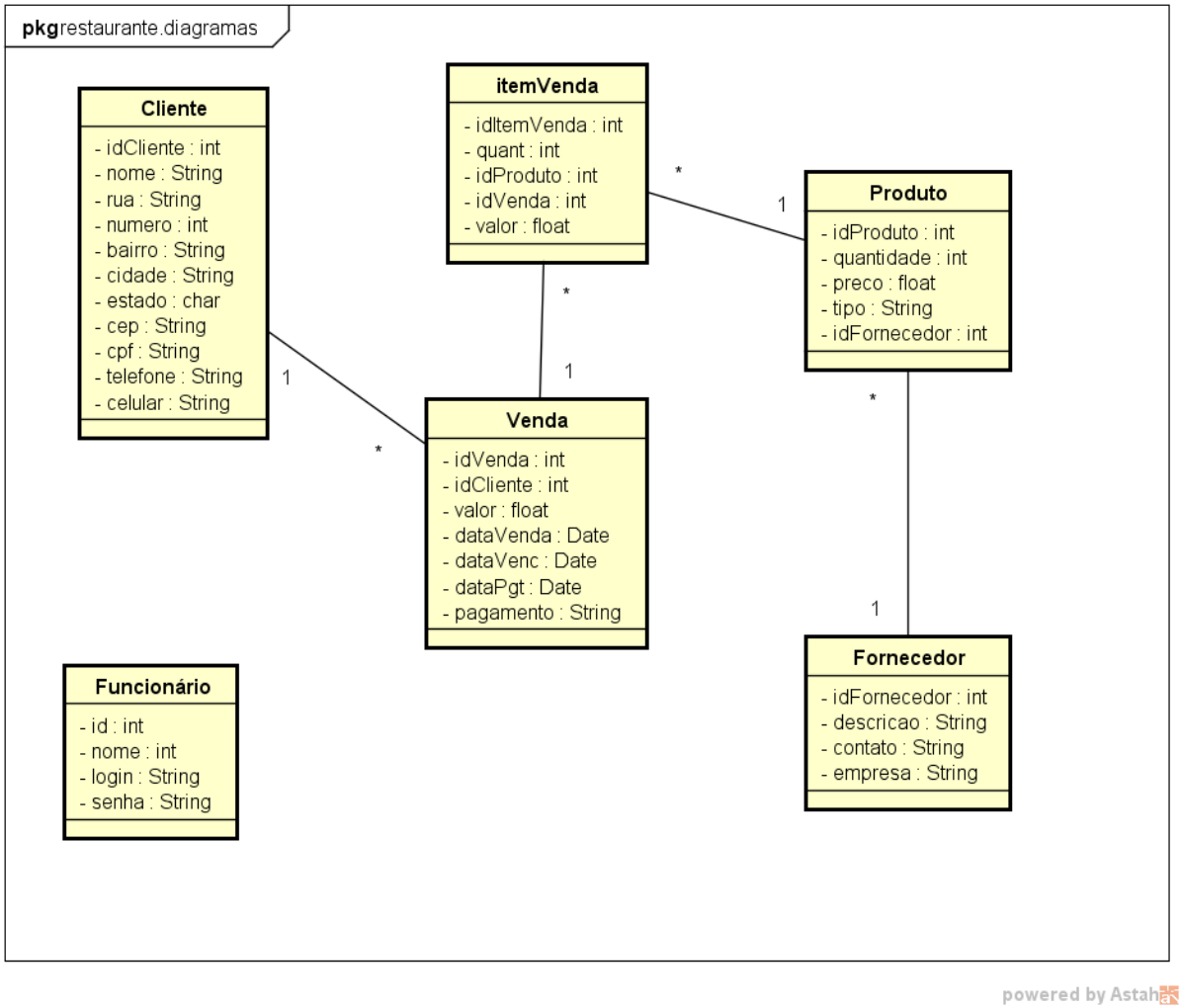

Figura 5: Diagrama de Classe

Observando o diagrama acima tendo em vista o atributo pagamento na classe venda será possível descrever o tipo de pagamento aprazo ou avista, ou informar os dados para cadastro

sobe consulta, pois o mesmo fornece o CPF o pagamento sendo avista a venda poderá ser realizada como consumidor não sendo preciso o cadastro do mesmo.

Informações diretamente ligadas à venda e ou produtos tem seus respectivos atributos no seu cadastro.

3.1.3 DIAGRAMA DE SEQUÊNCIA

Seu foco é no comportamento do sistema, apresenta a ordem temporal das mensagens enviadas e recebidas pelos objetos.

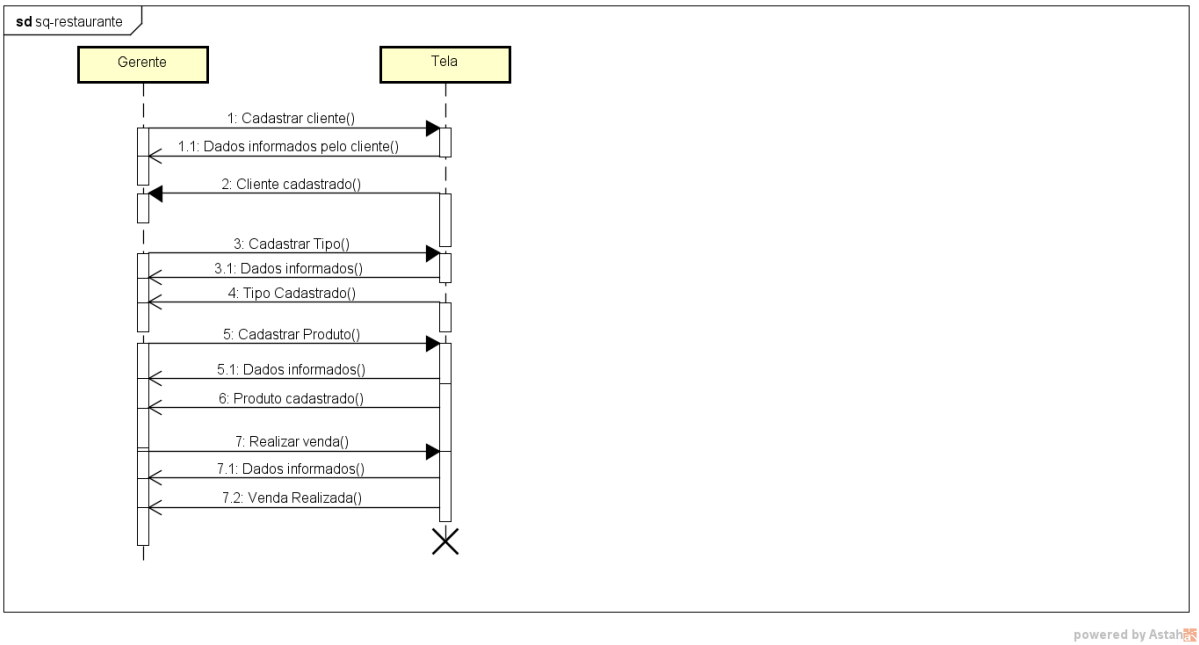

Figura 6: Diagrama de Sequência

Neste diagrama estão relacionados os cadastros necessários para, por exemplo, realização de uma venda, contudo devidamente cadastrado o usuário pode realizar a venda normalmente para que o cliente possa a vir a pagar sua conta semanalmente ou mês a mês conforme o combinado ficara tudo registrado no sistema.

3.1.4 DER – Diagrama Entidade-Relacionamento

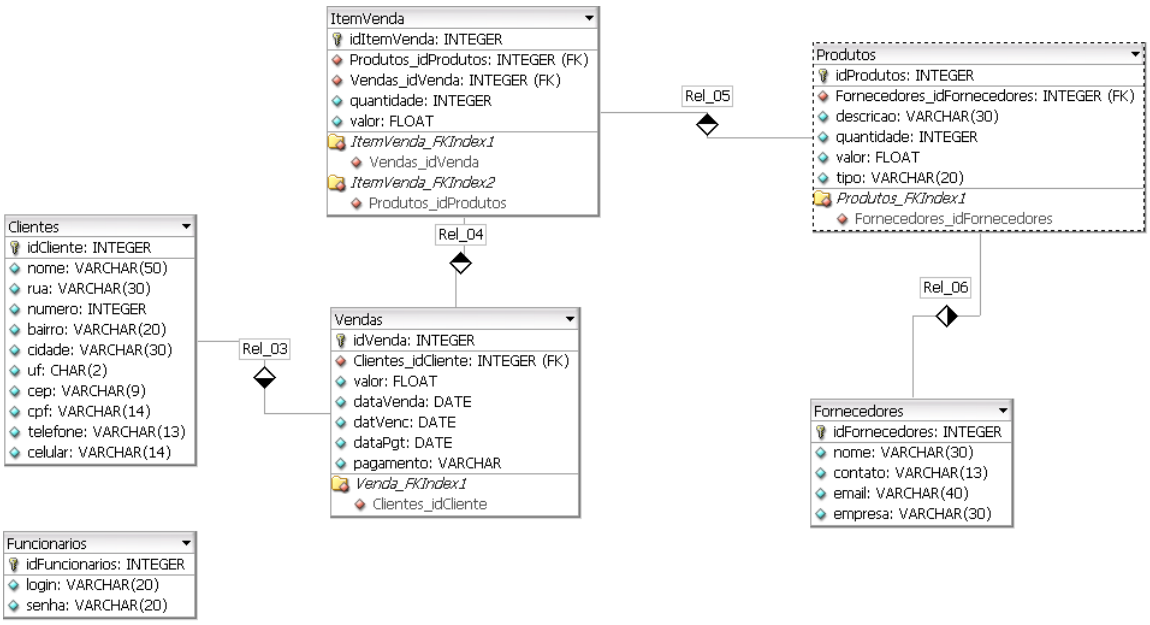

Figura 7 - Diagrama Entidade-Relacionamento

3.2 PLANEJAMENTO E PROJETO

Este tópico apresenta as etapas de planejamento, orçamento, custos do projeto e cronograma.

3.2.1 EAP – Estrutura Analítica do Projeto (*WBS*)

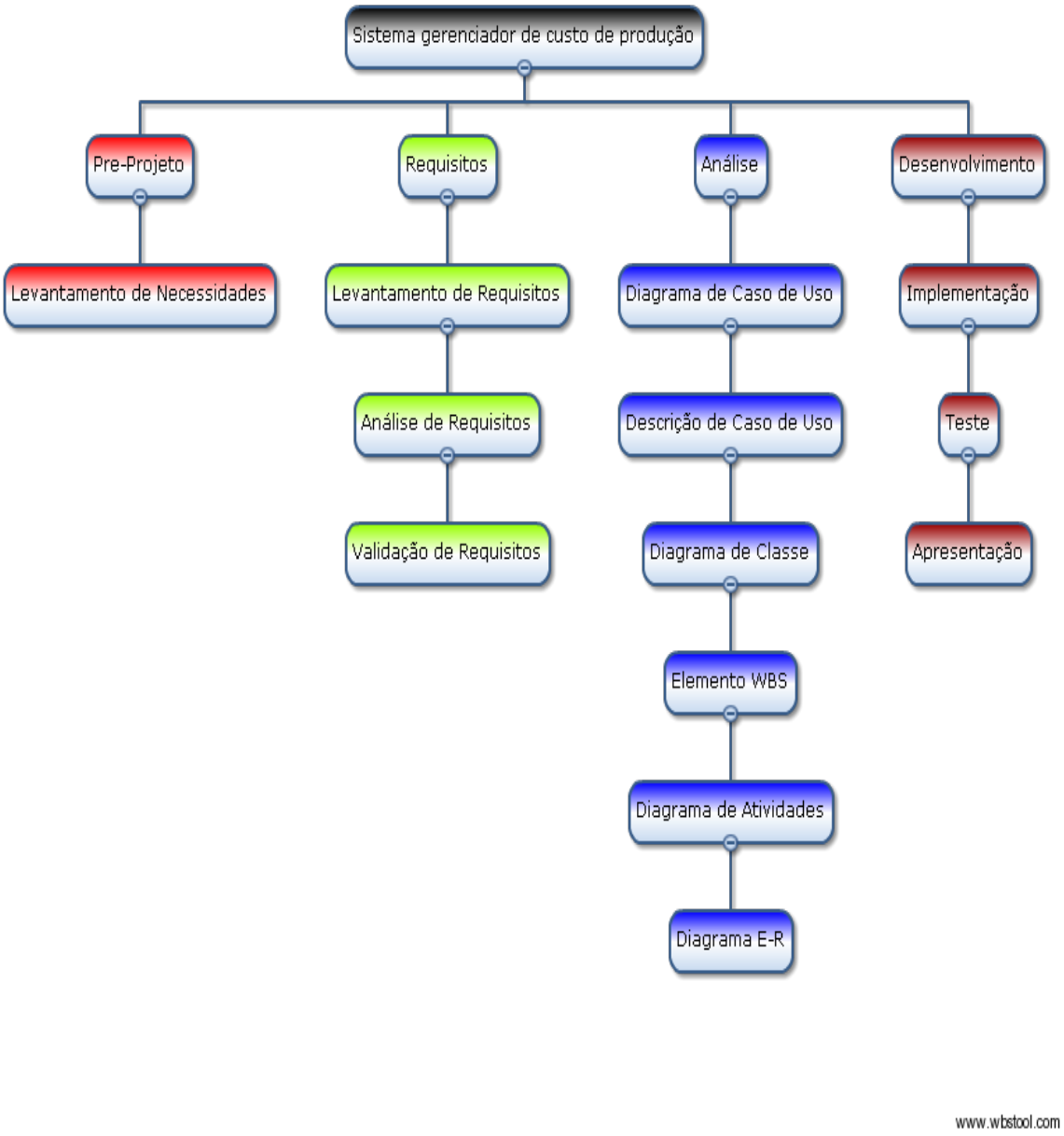

Figura 8 – WPS

3.2.2 ESPECIFICAÇÃO DE CUSTOS

Os recursos necessários para análise e desenvolvimento do Sistema de Gerenciamento para restaurantes foram:

- 1 Analista de Sistemas
- 1 Programador
- 1 Notebook

Orçamento de Pessoal

Analista de Sistemas			
Analista	Quantidade de Horas	Custo Hora	Total
Jhonatha H. Dias da Silva	180	28,00	5.040,00
Programador			
Programador	Quantidade de Horas	Custo Hora	Total
Jhonatha H. Dias da Silva	320	23,00	7.360,00
Total de Custos Pessoal	12.400,00		

Tabela 3 - Orçamento de Custos com Mão de Obra

Orçamento de Equipamento

• 01 Notebook

- Valor unitário = R$2.400,00
- Dias (de uso) = 140
- Depreciação = R$2.400,00 / 24 meses (02 anos. Tempo de depreciação) = 100,00
- Custo nos 140 dias = R$100,00 / 30 = 3,33 * 140 = 466,66

Custo Total do Projeto = 12.866,66

3.2.3 CRONOGRAMA

CRONOGRAMA	Out	Nov	Dez	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Out/Nov
Levantamento de Necessidades	x	x										
Levantamento de Requisitos		x										
Análise de Requisitos		x										
Validação de Requisitos		x										
Diagrama de Caso de Uso			x	x								
Descrição de Caso de Uso			x	x								
Diagrama de Classe				x								
Diagrama de Atividades				x	x							
Diagrama E-R				x	x							
Elaboração da Qualificação					x	x						
Apresentação da Qualificação							x					
Implementação							x	x	x	x		
Testes									x	x		
Escrita da Versão Final do Trabalho										x	x	
Apresentação e Revisão											x	x

4 DESENVOLVIMENTO DO SISTEMA

A implementação do sistema foi realizada utilizando o ambiente de desenvolvimento Visual Studio. O padrão de desenvolvimento escolhido para esta aplicação foi o Microsoft MVC[13], pois torna mais fácil o gerenciamento e manutenção das aplicações devido a separação entre a camada de design, de código e de acesso aos dados que esse padrão oferece. Segundo LOTAR(2014) o ASP.NET MVC fornece, por meio de *design patterns*, uma poderosa alternativa para criar websites ASP.NET dinâmicos.

O ASP.NET MVC separa a aplicação em 3 componentes: *model*, *view* e *controller*. O *model* contém o código da camada de dados, o *view* implementa o design da aplicação e o *controller* recebe as requisições do usuário através da *view*, verificando a requisição de acesso aos dados ou a outra *view*, e retorna para o usuário.

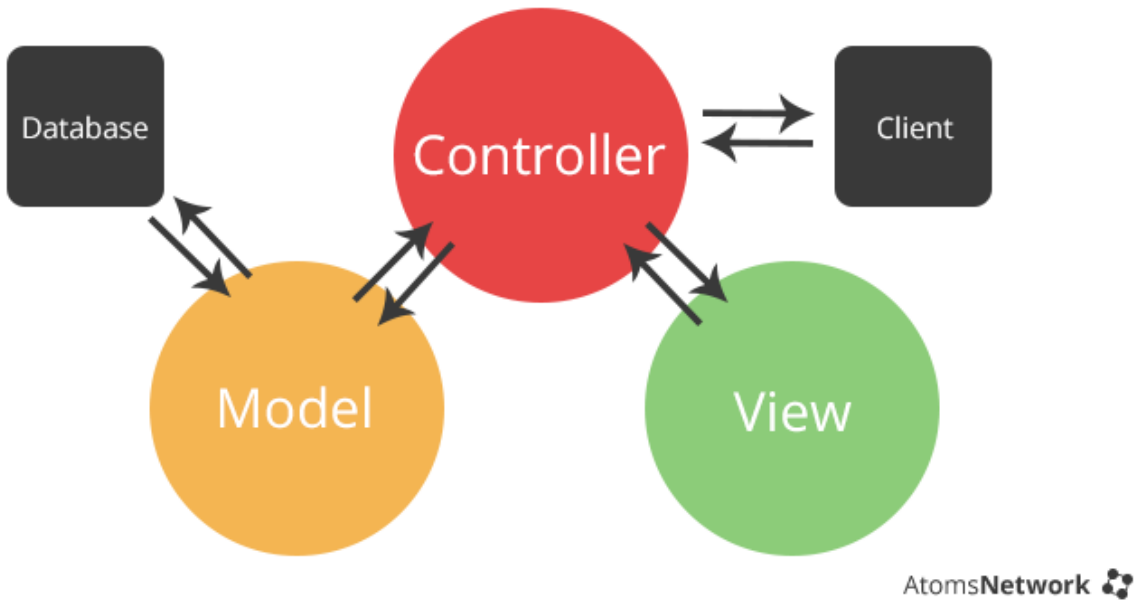

Figura 9 - Relação entre Model, View e Controller. Imagem Disponível em < http://www.tutorialized.com/tutorial/Fundamentals-of-an-MVC-Framework/81946>. Acesso em Jul. 2016.

[13] https://msdn.microsoft.com/pt-br/library/dd381412(v=vs.108).aspx

4.1 ORGANIZAÇÃO DO PROJETO

O projeto está organizado em pastas, para facilitar a manutenção. Segue abaixo uma imagem para expor como está a organização das pastas do projeto.

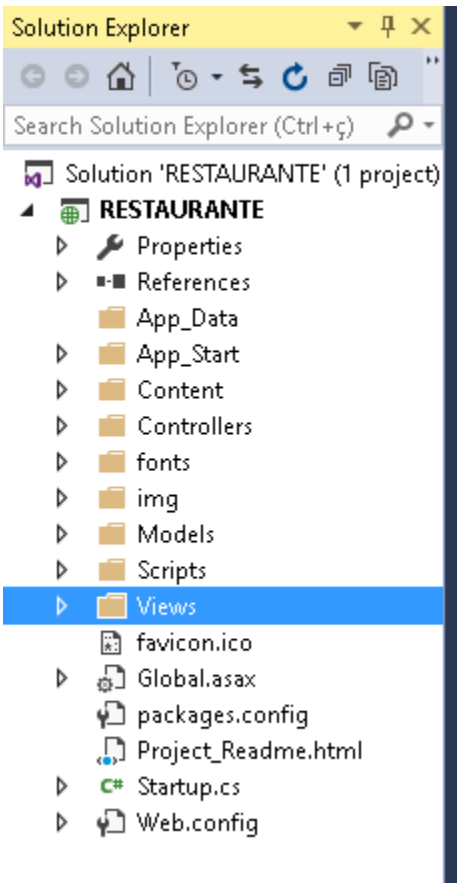

Figura 10 - Organização das Pastas do Projeto

4.1.1 PASTA *MODELS*

A pasta *Models* contém os códigos que implementam a camada de modelo, que por sua vez recuperam e armazenam o estado de um modelo no banco de dados. Um objeto criado na camada de modelo recupera informações em um banco de dados e realiza operações sobre o objeto recuperado, salvando as alterações realizadas novamente na tabela que foi feita a extração dos dados.

```
Produto.cs
RESTAURANTE                                   RESTAURANTI
using ...

namespace RESTAURANTE.Models
{
    [Table("Produtos")]
    public partial class Produto
    {
        public Produto()
        {
            ItemVendaProduto = new HashSet<ItemVenda>();
        }

        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        [Key]
        [Display(Name = "Código do produto: ")]
        public int idProdutos { get; set; }

        [Required(ErrorMessage = "Campo descrição é obrigatorio")]
        [MaxLength(35, ErrorMessage = "Tamanho máximo 35 caracteres")]
        [Display(Name = "Descrição: ")]
        public string descricao { get; set; }

        [Display(Name = "Quantidade: ")]
        [Required(ErrorMessage = "Quantidade do jogo é obrigatorio")]
        public int quantidade { get; set; }

        [DisplayFormat(DataFormatString = "{0:n2}", ApplyFormatInEditMode = true,
        NullDisplayText = "quantidade Inválida")]
        [Display(Name = "Preço: ")]
        public decimal? preco { get; set; }

        [Required(ErrorMessage = "Campo tipo é obrigatorio")]
        [MaxLength(20, ErrorMessage = "Tamanho máximo 20 caracteres")]
        [Display(Name = "Tipo: ")]
        public string tipo { get; set; }

        [Display(Name = "Fornecedor: ")]
        public int? Fornecedor_id { get; set; }

        public virtual Fornecedor FornecedorProduto { get; set; }

        public virtual ICollection<ItemVenda> ItemVendaProduto { get; set; }

    }
```

Figura 11 - *Model* Produto

A figura acima (Figura 10), representa a implementação da classe de modelo Produto. Para implementação desta e de todas as classes de modelo deste projeto, foram utilizadas as classes *System.ComponentModel.DataAnnotations*[14], que oferecem métodos para validação na tela de

[14] https://msdn.microsoft.com/pt-br/library/system.componentmodel.dataannotations(v=vs.110).aspx

usuário, mapeamento e controle de dados, além de métodos para geração das tabelas de banco de dados do sistema.

A anotação ***Table*** indica que a tabela deverá ser mapeada a partir da classe que se encontra a anotação, dentro de colchetes e aspas duplas indica-se o nome que deverá ser atribuído a tabela no banco de dados.

A anotação ***Key*** específica a propriedade que identifica exclusivamente um objeto ou entidade.

A anotação ***Required*** usa-se para indicar que o atributo não pode ser nulo, informando isso ao banco de dados e utilizados para validação na tela do usuário.

As anotações ***MaxLenght, Display*** são usadas para validações em tela de usuário, respectivamente representa o número máximo de caracteres do campo, como a informação deverá ser indicada na tela e o tipo de dados que a informação representa.

4.1.2 CONTEXTO

Outra classe que se encontra na pasta *Models* é a classe de contexto. Esta classe é responsável pelo mapeamento, persistência de dados e pela conexão com o banco de dados. Também é possível realizar as alterações que forem pertinentes ao sistema com respeito ao banco de dados, como por exemplo, informar que qualquer atributo do tipo *String* que seja declarado nas demais classes, será mapeado para a tabela do banco de dados com o seu tipo alterado para varchar, e não como nvarchar, que provocaria uso excessivo de recursos, utilizando um espaço de memória maior do que o necessário para o mesmo.

A implementação do Sistema foi feita utilizando o método de codificação *Code-First* [15]pois desta forma, se torna possível controlar todos os aspectos do mapeamento através das anotações. Este método de desenvolvimento é implementado por uma ferramenta de ORM (*Object Relational Model* – Objeto Modelo Relacional) chamada *Entity Framework*[16]. A

[15] https://msdn.microsoft.com/pt-br/library/Hh972463.aspx

[16] https://msdn.microsoft.com/pt-br/library/jj128157.aspx

utilização do *Entity Framework* torna o acesso e a manipulação dos dados mais simples, pois o desenvolvedor já trabalha naturalmente com objetos. SANTOS(2012)

SisRestaurante.cs

RESTAURANTE

RESTAURANTE

```
using ...

namespace RESTAURANTE.Models
{
    class SisRestaurante : DbContext
    {

        public SisRestaurante() : base("BANCO") { }


        public virtual DbSet<Cliente> Clientes { get; set; }
        public virtual DbSet<Fornecedor> Fornecedores { get; set; }
        public virtual DbSet<ItemVenda> ItemVendas { get; set; }
        public virtual DbSet<Produto> Produtos { get; set; }
        public virtual DbSet<Venda> Vendas { get; set; }


        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Fornecedor>()
                .HasMany(e => e.ProdutosFornecedores)
                .WithOptional(e => e.FornecedorProduto)
                .HasForeignKey(e => e.Fornecedor_id);

            modelBuilder.Entity<Produto>()
                .HasMany(e => e.ItemVendaProduto)
                .WithOptional(e => e.ProdutoItemVenda)
                .HasForeignKey(e => e.Produto_id);

            modelBuilder.Entity<Venda>()
                .HasMany(e => e.ItemVendaVenda)
                .WithOptional(e => e.VendaItemVenda)
                .HasForeignKey(e => e.Venda_id);

            modelBuilder.Entity<Cliente>()
                .HasMany(e => e.VendasClientes)
                .WithOptional(e => e.ClienteVendas)
                .HasForeignKey(e => e.Cliente_id);


        }
    }
}
```

Figura 12 - Classe de Contexto

4.1.3 PASTA *CONTROLLERS*

A pasta *Controllers* é onde fica os componentes que são responsáveis pelo controle da interação do usuário que se dá através das *Views*, estes acessam a *Model* que por sua vez acessa o banco de dados da aplicação e responde a essa interação de maneira correta, exibindo uma resposta de acordo com a solicitação do usuário. Utiliza-se o método *ActionResult*[17] para apresentar o resultado de uma ação.

VendasController.cs

RESTAURANTE — RESTAURANTE.Controllers

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using RESTAURANTE.Models;

namespace RESTAURANTE.Controllers
{
    public class VendasController : Controller
    {
        private SisRestaurante db = new SisRestaurante();

        // GET: Vendas
        public ActionResult Index()
        {
            var vendas = db.Vendas.Include(v => v.ClienteVendas);
            return View(vendas.ToList());
        }

        // GET: Vendas/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Venda venda = db.Vendas.Find(id);
            if (venda == null)
            {
                return HttpNotFound();
            }
            return View(venda);
        }

        // GET: Vendas/Create
        public ActionResult Create()
        {
            ViewBag.Cliente_id = new SelectList(db.Clientes, "idCliente", "nome");
            return View();
        }
```

Figura 13 - *Controller* de Vendas

[17] https://msdn.microsoft.com/pt-br/library/system.web.mvc.actionresult(v=vs.118).aspx

O *Controller* de venda contém os métodos responsáveis pela manutenção e cadastros das vendas. Nele se encontra os *ActionsResults*, responsáveis pela criação, edição e exclusão de vendas. Há também as ações responsáveis pela listagem das vendas cadastradas no sistema.

O *Controller* de venda também é relacionado pelo cadastro e manutenção de cliente e itemVenda, contendo todas as regras para a criação de novas vendas, implementando métodos capazes de impedir a replicação de informações e manipulação indevida de dados.

4.1.4 PASTA *VIEWS*

A pasta *Views* armazena as páginas HTML, que são responsáveis por exibir as informações repassadas pelos *Controllers* e também pelos envios de formulários.

Create.cshtml

```
@model RESTAURANTE.Models.Cliente

@{
    ViewBag.Title = "Cadastrar novo cliente";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Cadastrar novo cliente</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Entre com o dados do cliente</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.nome, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.nome, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.nome, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.rua, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.rua, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.rua, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.numero, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.numero, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.numero, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.bairro, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.bairro, new { htmlAttributes = new { @class = "form-control" } })
```

Figura 14 - *View* Cliente

A renderização de páginas HTML é realizada com a utilização da *View Engine Razor*[18], que agrega marcações HTML com comando de código C#, conhecidos no Asp.Net MVC como *Helpers*, onde o caractere especial @ indica o início de uma expressão de comando, que é utilizada para informar uma condição ou simplesmente mostrar o conteúdo de uma variável.

Além dos *Helpers* responsáveis por mostrar e captar conteúdo, há outros que são de extrema importância em aplicações Asp.Net MVC como o **@Html.AntiForgeryToken()**[19] que é responsável pela validação nos envios de formulários, evitando que os *Controllers* recebam dados de páginas que não pertençam a mesma sessão, criando um campo de formulário oculto que é validado quando a *Action* no *Controller* responsável pela View recebe o formulário. Para que esta validação seja realizada, devemos informar na *ActionResult* do *Controller* o atributo **ValidateAntiForgeryToken**.

4.1.5 OUTRAS PASTAS

Outras pastas que se encontram no *Solution Explorer* são usadas para configurações gerais e arquivos de imagens, CSS e Scripts usadas na aplicação.

- **Pasta *App_Start:*** armazena arquivos de configuração da aplicação.
- **Pasta *Content:*** armazena os arquivos de CSS utilizados na aplicação.
- **Pasta *Fonts*:** armazena os arquivos de fonte utilizados para serem usados na aplicação.
- **Pasta Img:** armazena as imagens utilizadas no sistema.
- **Pasta *Script*:** armazena os arquivos de JavaScript.

[18] https://msdn.microsoft.com/pt-br/library/Gg675215.aspx

[19] https://msdn.microsoft.com/pt-br/library/system.web.mvc.htmlhelper.antiforgerytoken(v=vs.118).aspx

4.2 INTERFACES DO SISTEMA

A Figura 15 ilustra a tela de *Login* do Sistema.

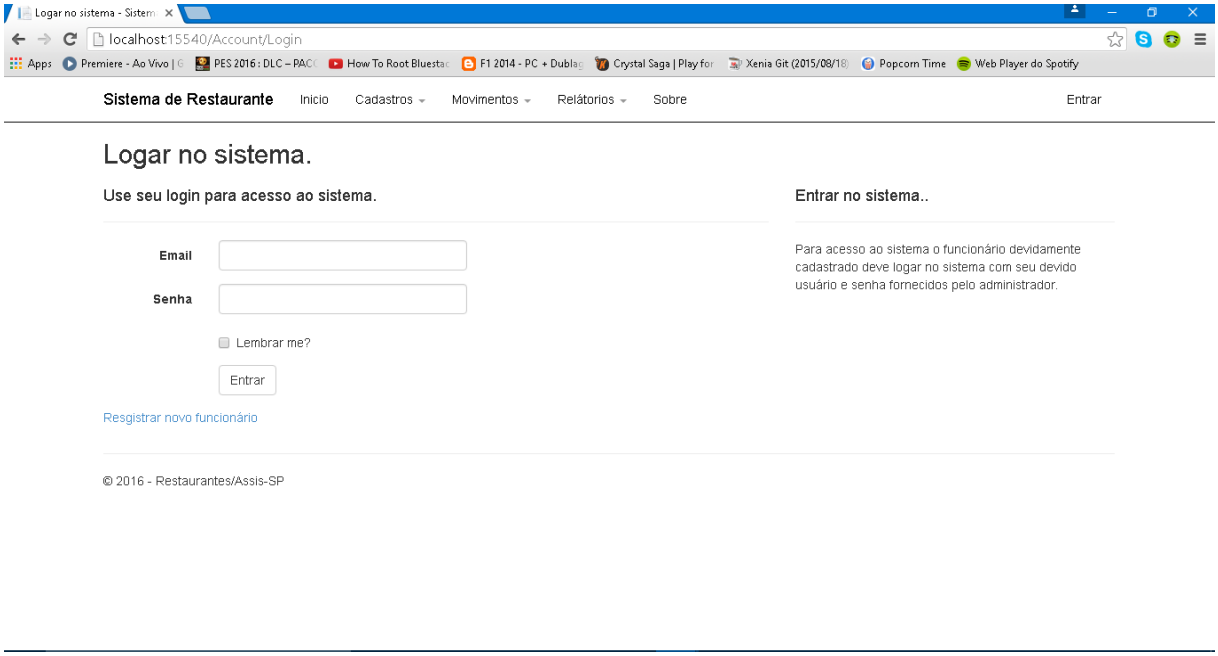

Figura 15 - Tela de *Login*

As imagens apresentadas abaixo (Figura 16 e Figura 17), mostram a tela inicial do Sistema, apresentando o Menu com os *links* para cadastro de Cliente, Fornecedores, Produtos e Funcionários.

Figura 16 - Tela inicial

Figura 17 - Tela com *Menu*

A imagem abaixo (Figura 18) apresenta a tela de cadastro de novos Clientes. Para o cadastro de um novo cliente é necessário que todos os itens sejam devidamente preenchidos. Cada campo tem suas respectivas validações, cada cliente tem seu próprio código gerado automaticamente, não podendo haver replicação desta informação.

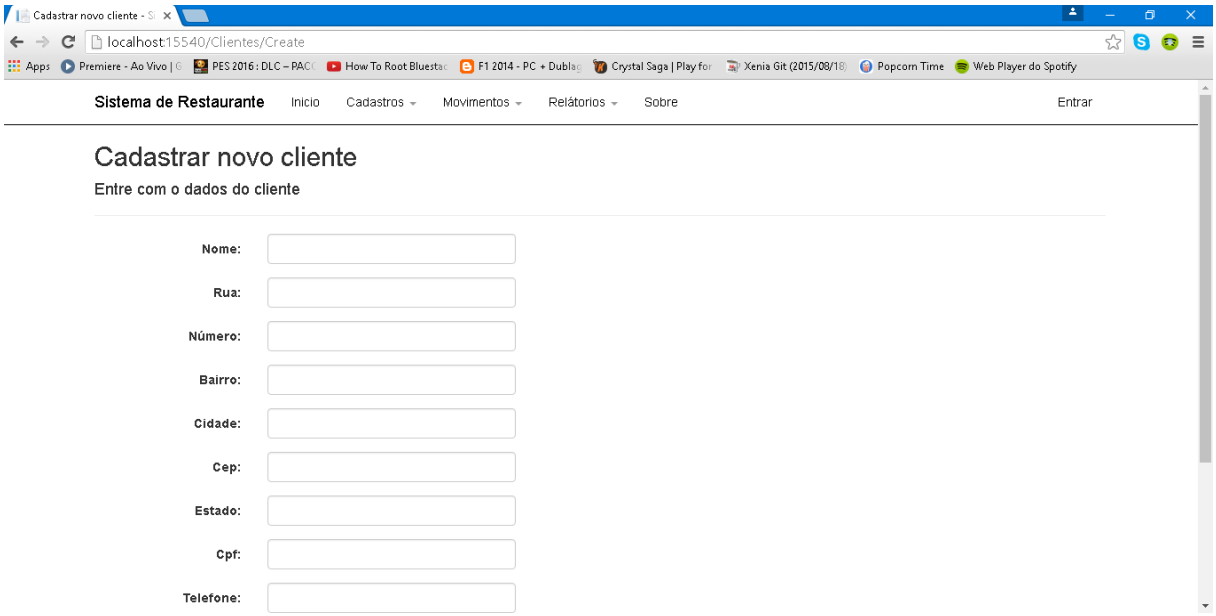

Figura 18 - Tela cadastro de Cliente

A imagem abaixo (Figura 19) mostra os detalhes do cliente cadastrado no sistema.

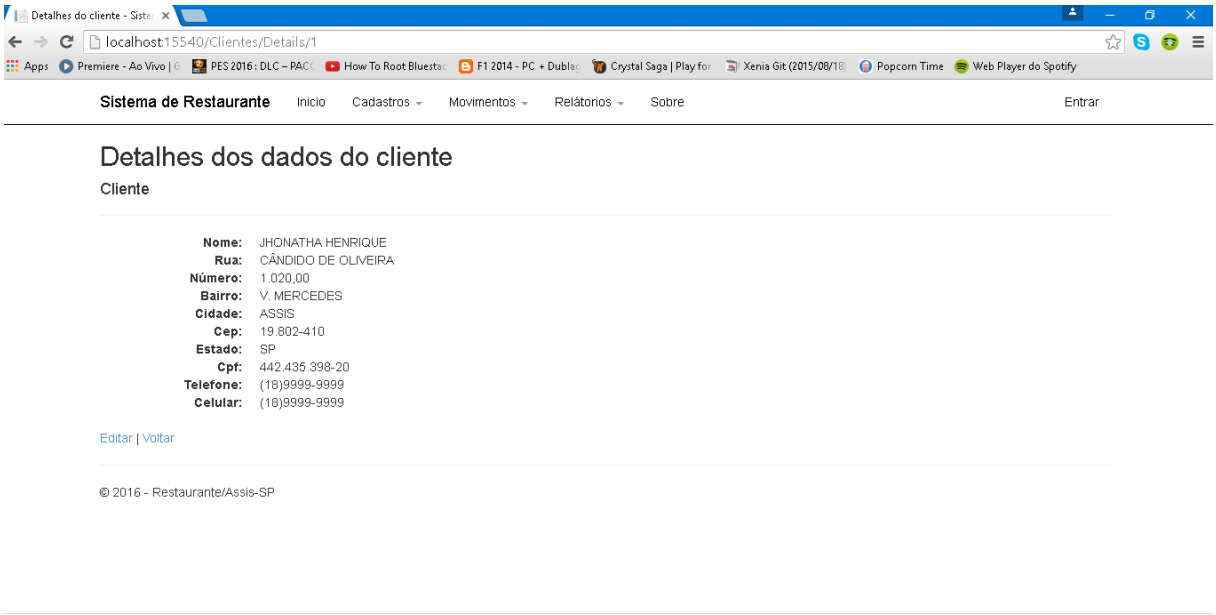

Figura 19 - Tela detalhes do cliente

A imagem abaixo (Figura 20) apresenta a tela de cadastro de novos Produtos. Para o cadastro de um novo produto é necessário que todos os itens sejam devidamente preenchidos. Cada campo tem suas respectivas validações bem como no cadastro de fornecedor o mesmo tem que estar cadastrado antes do produto.

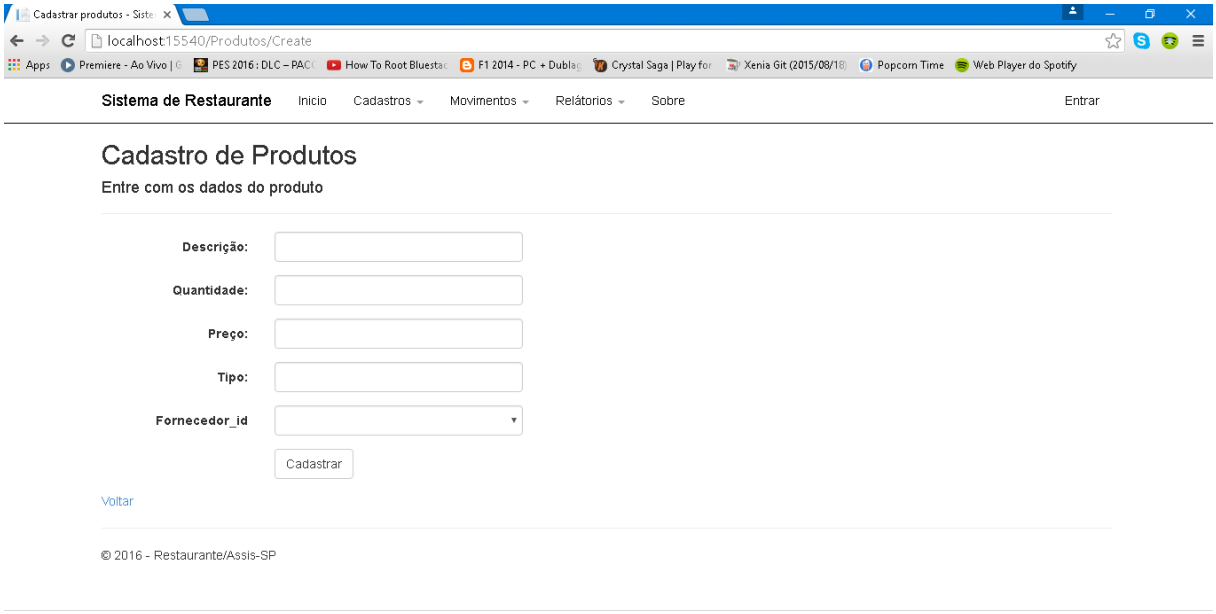

Figura 20 - Tela cadastro produto

A imagem abaixo (Figura 21) apresenta os detalhes do produto cadastrado.

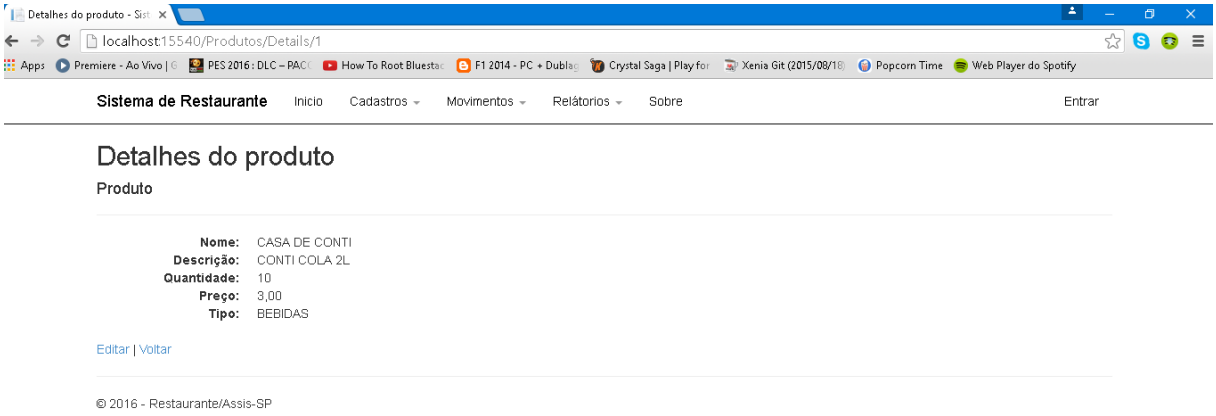

Figura 21 - Tela detalhes do produto

A imagem abaixo (Figura 22) apresenta a tela de Vendas. Para realizar novas vendas é necessário que todos os itens sejam devidamente preenchidos. Cada campo tem suas respectivas validações, bem como nos cadastros de fornecedor, produto e cliente os mesmo tem que estar cadastrado antes de realizar uma venda.

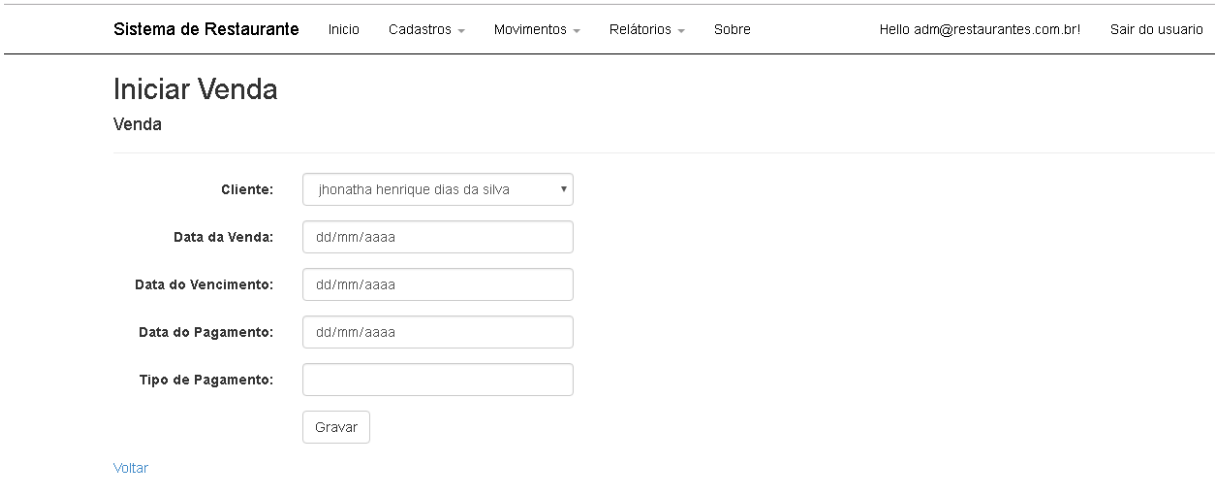

Figura 22 - Tela venda

A imagem abaixo (Figura 23) apresenta a tela de detalhes da Venda. Onde mostra o Cliente, Valor total, Data de venda, Data de vencimento, Data de pagamento e Pagamento. E também os respectivos Itens da venda sendo Produtos, Código da venda e Quantidade.

Figura 23 - Tela detalhes da venda

5 CONCLUSÃO

Este trabalho de conclusão de curso foi motivado pelo crescente interesse na utilização de conceitos de programação prático e interativos onde a facilidade das integrações dos aplicativos com bancos de dados sejam tratados como uma simples ação e não como parte critica de um programa.

O uso do Padrão MVC, trouxe maior facilidade em criar a aplicação Web com acesso a Banco de Dados, as tecnologias empregadas neste Sistema são muitos usados atualmente no desenvolvimento de aplicações para internet. A execução deste projeto trouxe grande conhecimento em programação utilizando a linguagem C# em aplicações Web com o uso de ferramentas da Microsoft e proporcionou experiência com o padrão de desenvolvimento MVC, que é usado em várias linguagens de programação, não sendo restrito a tecnologias Microsoft. Também foi proporcionado grande aprendizado em análise de projetos, utilização de ferramentas de UML e práticas para melhor aplicação destes conhecimentos.

5.1 TRABALHOS FUTUROS

Será considerado para trabalhos futuros, a implementação de outros módulos a fim de atingir ainda mais o requisitos do usuário.

REFERÊNCIAS

Caelum Ensino e Inovação. **Apostila do curso FN-13 C# e Orientação a Objetos**. Disponível em <http://www.caelum.com.br/apostila-csharp-orientacao-objetos/>Acessado Jan. 2016.

DEITEL, H. M; **C# Como Programar**, São Paulo. Pearson Education, 2003.

GÓES, Wilson M. **Aprenda UML por meio de Estudos de Caso** 1ªed. São Paulo: Novatec, 2014.

LOTAR, Alfredo. **Programando com ASP.NET MVC – Aprenda a desenvolver aplicações web utilizando a arquitetura MVC** 3ªed. São Paulo: Novatec, 2014.

PROVENCIO, David. RECIO, Francisco**. Net Framework - Arquitetura básica da plataforma .Net. Descrição do Framework e seus principais componentes: Linguagens, biblioteca de classes e CLR**. Criarweb. Disponível em <http://www.criarweb.com/artigos/net-framework.html> Publicado em 03 mar. 2008. Acessado Jan. 2016.

SANTOS, Carlos**. Fundamentos do Entity Framework 4**, MSDN. Microsoft. Disponível em < https://msdn.microsoft.com/pt-br/library/jj128157.aspx > Acessado Jul. 2016.

STELLMAN, Andrew; GREENE, Jennifer; **Use a Cabeça! C#** - 1ª ed. - Rio de Janeiro. Editora Alta Books, 2008.

PROVENCIO, David. RECIO, Francisco**. Net Framework - Arquitetura básica da plataforma .Net. Descrição do Framework e seus principais componentes: Linguagens, biblioteca de classes e CLR**.

Criarweb. Disponível em <http://www.criarweb.com/artigos/net-framework.html> Publicado em 03 mar. 2008. Acessado Jan. 2016

Livro: - ROB, Peter. CORONEL, Carlos. **Sistema de Banco de Dados**.

LAGINSKI LIPPEL, Isabela. <https://repositorio.ufsc.br/bitstream/handle/123456789/83249/192881.pdf?sequence=1> Publicado em 2002

Livro: - COIMBRA DE ARAÚJO, Everton. **ASP.NET MV5, crie aplicações web na plataforma Microsoft** – São Paulo. Casa do Código, 2016.