ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO
INSTITUTO POLITÉCNICO DA GUARDA
Departamento de Informática

Desenvolvimento da aplicação *EstgCRM*

(Gestão de Relacionamento com o Cliente)

Dezembro de 2008

Ana Paula Pinto Figueiras Nº4678

Desenvolvimento de uma aplicação para Gestão de Relacionamento com o Cliente - *EstgCRM*

Ana Paula Pinto Figueiras Nº4678

Relatório submetido com requisito parcial
para obtenção do grau de Licenciado em
Engenharia Informática

Orientado por Maria Clara Silveira

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO
INSTITUTO POLITÉCNICO DA GUARDA
Departamento de Informática

Dezembro de 2008

Resumo

O projecto consiste na elaboração de um sistema CRM - EstgCRM, no qual vão ser criadas várias campanhas, cada campanha vai estar direccionada para um dado segmento ou perfil de clientes. Os clientes podem registar-se, fazer o login, efectuar sugestões ou reclamações e responder a inquéritos. Inquéritos, esses que permitirão ao administrador verificar se a campanha foi bem sucedida ou não. Por sua vez o Responsável do sistema CRM (EstgCRM), pode criar campanhas, segmentos, associar os segmentos à respectiva campanha, criar canais de comunicação, produtos, inquéritos e as respectivas perguntas, associar as perguntas ao inquérito e os inquéritos ao respectivo cliente. Pode ainda registar oportunidades de venda e fazer consultas.

Para o desenvolvimento da aplicação usaram-se as seguintes tecnologias: linguagem de modelação UML, base de dados em SQLServer e linguagem de programação C#.

Palavras chave: Campanha, CRM, Inquérito, Segmento, Base de Dados e UML.

Agradeço à Doutora Maria Clara Silveira, as valiosas sugestões, a paciência e as críticas construtivas feitas no decorrer deste projecto.

Índice

Eliminado: <sp> Instituto Politécnico da Guarda Escola Superior de Tecnologia e Gestão - 5º Ano EI

Formatada: Cabeçalho

Comentário [di1]: Numeração: i, ii, iii, … a página 1 é na introdução.

Definição de estilo: Normal

Definição de estilo: Título 1

Definição de estilo: Título 3: Esquerda

Definição de estilo: Índice 2: Tabulações: 1,55 cm, Esquerda + 15,82 cm, Direita,Carácter de preenchimento: ...

Eliminado: ¶

Formatada: Tipo de letra: 14 pt, Negrito

Eliminado: <sp> Instituto Politécnico da Guarda Escola Superior de Tecnologia e Gestão - 5º Ano EI

Formatada: Cabeçalho

Eliminado: <sp> Instituto Politécnico da Guarda Escola Superior de Tecnologia e Gestão - 5º Ano EI

Formatada: Cabeçalho

Índice de Tabelas

Eliminado: 1. . Introdução . 1¶
1.1. . Motivação . 2¶
1.2. . Objectivos . 2¶
1.3. . Planeamento de Actividades . 3¶
1.3.1 . Mapa de *Gantt* Previsto . 3¶
1.3.2 . Mapa de Gantt Efectivo (final) . 4¶
1.4. . Estrutura do Relatório . 6¶
2. . Analise e Concepção do Sistema *EstgCRM* . 7¶
2.1. . O que se entende por CRM . 7¶
2.2. . Levantamento de requisitos . 9¶
2.3. . Descrição dos Casos de Uso . 10¶
2.4. . Diagramas UML que foram utilizados . 14¶
2.5. . Diagrama de Casos Uso . 15¶
2.6. . Diagrama de Actividades . 18¶
2.7. . Diagramas de Sequencia . 20¶ ...

Eliminado: **1. . Introdução . 13¶**
1.1. . Motivação . 14¶
1.2. . Objectivos . 14¶
1.3. . Planeamento de Actividades . 15¶
1.3.1 . Mapa de *Gantt* Previsto . 15¶
1.3.2 . Mapa de Gantt Efectivo (final) . ¶
1.4. . Estrutura do Relatório . 17¶
2. . Analise e concepção do Sistema *EstgCRM* . 19¶
2.1. . O que se entende por CRM . 19¶
2.2. . Levantamento de requisitos . 21¶
2.3. . Descrição dos Casos de Uso . 22¶ ...

Formatada: Tipo de letra: 14 pt, Negrito

Eliminado: <sp> Instituto

Formatada

Eliminado: Tabela 1 - Definições e

Eliminado: 24

Eliminado: ¶

Eliminado: 26

Eliminado: ¶

Eliminado: 66

Eliminado: ¶

Eliminado: 68

Eliminado: ¶

Eliminado: 70

Eliminado: ¶

Eliminado: 71

Eliminado: ¶

Eliminado: 73

Eliminado: ¶

Eliminado: 76

Eliminado: ¶

Eliminado: 77

Eliminado: ¶

Eliminado: 78

Eliminado: ¶

Eliminado: 80

Eliminado: ¶

Eliminado: 81

Eliminado: ¶

Eliminado: 83

Eliminado: ¶

Eliminado: 85

Eliminado: ¶

Eliminado: 87

Eliminado: ¶

Eliminado: 88

Eliminado: ¶

Eliminado: 91

Eliminado: ¶

Eliminado: 92

Eliminado: ¶

Eliminado: 96

Eliminado: ¶

Eliminado: 97

Eliminado: ¶

Eliminado: 99

Eliminado: ¶

Eliminado: 100

Eliminado: ¶

Eliminado: 102

Eliminado: ¶

Eliminado: <sp> Instituto Politécnico da Guarda Escola Superior de Tecnologia e Gestão - 5º Ano EI

Formatada: Cabeçalho

Eliminado: ¶
¶
¶
¶
¶
¶
¶
Quebra de página
¶

Formatada: Tipo de letra: 14 pt, Negrito

Índice de Figuras

Eliminado: <sp> Instituto Politécnico da Guarda Escola Superior de Tecnologia e Gestão - 5º Ano EI

Formatada: Cabeçalho

Eliminado: <sp> Instituto Politécnico da Guarda Escola Superior de Tecnologia e Gestão - 5º Ano EI

Formatada: Cabeçalho

Formatada: Legenda

Formatada: Tipo de letra: 11 pt

1. Introdução

O presente relatório tem por objectivo descrever as actividades desenvolvidas no âmbito do Projecto Fim de Curso em Engenharia Informática. Este projecto consistiu no desenvolvimento de uma aplicação CRM (Gestão de Relacionamento com o cliente). A gestão eficaz do relacionamento com os clientes será um dos principais factores de diferenciação competitiva nos próximos anos. Assim, neste projecto pretende-se criar campanhas, fazer a segmentação para efectuar campanhas direccionadas para um dado grupo de clientes, com um dado perfil e criar e enviar inquéritos para o cliente com vista a saber a sua opinião relativamente à campanha criada.

Com a explosão da Internet e o aumento de serviços *Online*, CRM é um conceito cada vez mais utilizado nos dias que passam, mais particularmente em campanhas de Marketing.

CRM não é mais que um termo novo para um conceito já antigo, a gestão do relacionamento com o cliente. Um bom exemplo do que então seria CRM, é a pequena mercearia do nosso bairro, onde o merceeiro conhece pessoalmente pelo nome os clientes, conhece os seus produtos preferidos, assim como as suas restrições financeiras e facilita o pagamento da mercadoria.

Segundo algumas empresas, CRM (Customer Relationship Management) não é mais do que colocar as pessoas e a tecnologia ao serviço de processos de negócio orientados para a compreensão e diálogo com os clientes, perseguindo o objectivo de como estabelecer relações duradouras e mutuamente vantajosas.

O CRM apresenta-se como um verdadeiro processo "*one-to-one*", transferindo o foco do objectivo tradicional de angariação de clientes a qualquer custo de retenção dos mesmos. Esta mudança de orientação deve-se à constatação de que custa entre 4 a 7 vezes mais angariar um novo cliente do que manter um existente.

Eliminado: <#>¶

...

Formatada

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

A palavra-chave de todo o processo de CRM é lealdade, isto porque as empresas têm cada vez mais consciência de que o seu maior activo são os clientes e que a melhor forma de os rentabilizar é através da conquista da sua lealdade, Assim, o CRM permite construir relações produtivas e duradouras entre empresas e clientes.

Os sistemas informáticos CRM implementam a filosofia de gestão das interacções junto do consumidor (cliente), esta filosofia visa estudar e documentar as campanhas de Marketing.

Esta solução coloca os profissionais de Marketing no centro dos recursos de informação (interna e externa) da Empresa, dando-lhe ferramentas de que precisam para trabalhar de modo mais eficaz. O resultado é um novo nível de ciência para o planeamento, execução e avaliação das suas campanhas.

1.1. Motivação

A motivação principal da realização deste trabalho resulta da importância do CRM ter vindo a ganhar uma crescente importância, pois cada vez mais a Economia Digital está a provocar uma queda gradual das barreiras ao consumo, alargando a esfera de influência das empresas até aos limites do alcance das comunicações (Internet, telefone, entre outros). Esta revolução digital aumenta simultaneamente a base de clientes potenciais, assim como mantém-los distanciados da concorrência por um simples “click”.

Outro desafio é aplicar os conhecimentos adquiridos nas várias disciplinas ao longo do curso e adquirir novos conhecimentos.

1.2. Objectivos

O objectivo principal deste projecto consiste em fazer a análise, concepção e implementação de uma aplicação para um sistema empresarial CRM.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

O sistema deve permitir a uma empresa fazer a gestão de relacionamentos com os seus clientes nas áreas de Marketing, Vendas e Serviços.

Os objectivos mais específicos deste trabalho são os seguintes:

- Permitir o planeamento, a execução e a análise de resultados de Campanhas de Marketing;

- Permitir a Segmentação (identificação de grupos de clientes ou contactos com base no seu perfil, preferências, comportamentos e necessidades), de forma a poder efectuar Campanhas de Marketing especificamente direccionadas;

- Permitir dar o seguimento adequado a cada oportunidade de negócio que possa surgir através das Campanhas ou através de realização de eventos de Marketing.

- Permitir a criação de inquéritos (adicionar perguntas aos inquéritos e associar os mesmos à respectiva campanha) e envio dos mesmos para os clientes preencherem.

1.3. Planeamento de Actividades

O Mapa de *Gantt* ou Gráfico de *Gantt* é uma ferramenta essencial no planeamento de um projecto. Através deste gráfico pode-se definir todas as actividades sequenciais de uma operação / projecto / produção, onde para cada operação se tem uma barra com o tamanho da sua duração. Este foi desenvolvido por *H. L. Gantt* em 1917.

1.3.1 Mapa de *Gantt* Previsto

Na Figura1.1, encontra-se registado o tempo que se previa gastar nas referidas tarefas do projecto proposto.

Eliminado: FiguraNo mapa de *Gantt* previsto (Figura 1 e Figura 2)

Formatada: Rodapé

Name	Begin date	End date
Revisão do funcionamento da aplicacao EstgCRM realizada no 3º ano	06-03-2007	28-04-2007
Revisão e alteração da Base de Dados	06-03-2007	28-04-2007
Alteração e correcção da aplicacao	21-03-2007	30-05-2007
Elaboração do design do site EstgCRM (Inquéritos)	28-03-2007	05-04-2007
Implementação e programação do site	01-05-2007	30-05-2007
Revisão e alteração do relatório de acordo com a nova aplicação	21-05-2007	22-05-2007
Actualização e elaboração dos diagramas	30-05-2007	06-06-2007
Tabela diferenças EstgCRM, Navision e uma actual a Microssoft CRM 3	06-06-2007	08-06-2007
Diferenças entre aplicações CRM antes e agora	08-06-2007	09-06-2007

Figura 1-1 Mapa de Gantt previsto

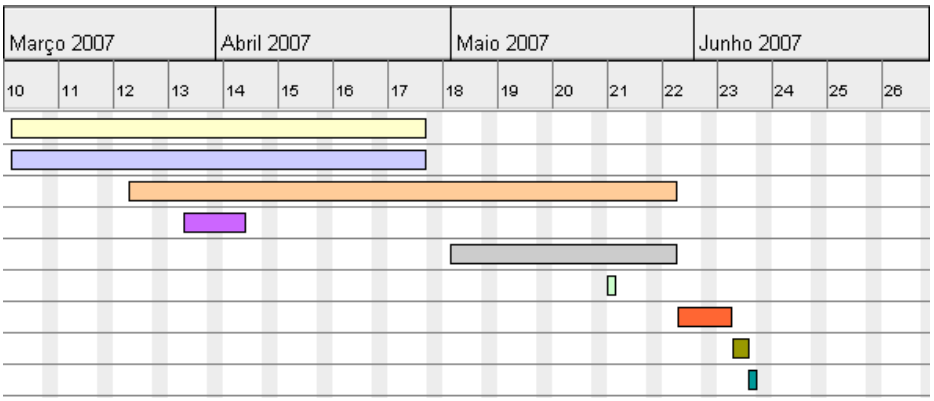


Figura 1-2 Mapa de Gantt com o tempo que se previa gastar

1.3.2 Mapa de Gantt Efectivo (final)

Nas Figuras 1.3 e 1.4 encontra-se registado o tempo que realmente foi utilizado na elaboração do trabalho nas referidas tarefas.

Formatada: Tipo de letra: 11 pt

Eliminado: Figura 1 -

Eliminado: Figura 2 -

Eliminado: No mapa de Gantt efectivo (Figura 3

Eliminado: Figura

Eliminado: 4

Eliminado:)

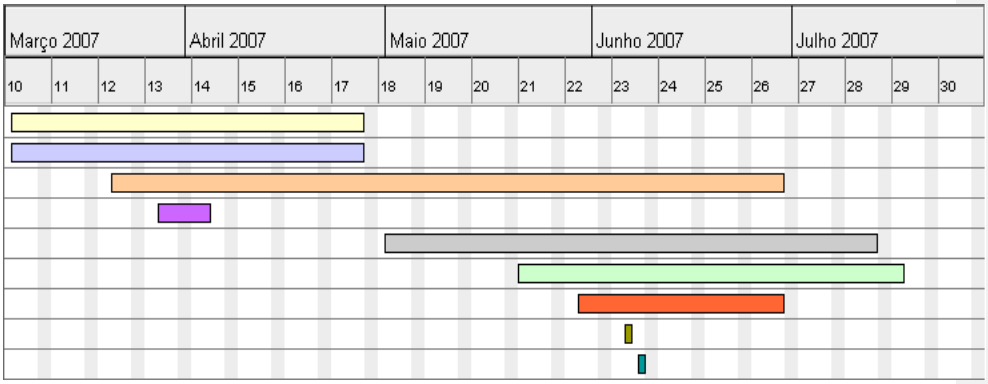
Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Name	Begin date	End date
Criação da aplicação EstgCRM	06-03-2007	28-04-2007
Criação e alteração da Base de Dados	06-03-2007	28-04-2007
Alteração e correcção da aplicação	21-03-2007	30-06-2007
Elaboração do design do site EstgCRM	28-03-2007	05-04-2007
Implementação e programação do site	01-05-2007	08-09-2007
Criação e alteração do relatório de acordo com a aplicação	21-05-2007	14-09-2007
Actualização e elaboração dos diagramas	30-05-2007	30-06-2007
Tabela de diferenças entre as aplicações EstgCRM, Microsoft Buisiness Solution Navision e Microsoft Dyna...	06-06-2007	07-06-2007
Diferenças entre aplicações CRM antes e agora	08-06-2007	09-06-2007

Eliminado: Figura 3-

Figura 1-3 Tabela com o tempo final, o tempo que efectivamente foi gasto nas tarefas referidas.



Eliminado: Figura 4 -

Figura 1-4 Mapa de Gantt com o tempo final, o tempo que efectivamente se gastou nas tarefas.

Fazendo a comparação entre o mapa de Gantt previsto e o mapa de Gantt final, conclui-se que houve actividades em que se pensava demorar um determinado tempo e acabou por se demorar mais algum tempo, embora se tenha tido outras actividades em que o tempo gasto foi menor do que o que realmente se previu.

Eliminado:

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

1.4. Estrutura do Relatório

Este relatório de projecto encontra-se estruturado em 5 capítulos dos quais, o primeiro corresponde à introdução.

O segundo capítulo é a análise do sistema EstgCRM. A metodologia adoptada para fazer a análise dos requisitos foi a orientação por objectos usando a linguagem UML (*Unified Modeling Language*). A UML é uma linguagem de modelação gráfica para especificação, visualização, construção e documentação de sistemas de software ou outros.

Utilizou-se as ferramentas *Rational Rose e Project*. Neste capítulo descreveram-se as funcionalidades da aplicação.

O terceiro capítulo é a implementação do projecto EstgCRM. Descreve-se a concepção e implementação de uma DB (base de dados) para o sistema EstgCRM. Utilizou-se o *SQLServer 2005 Menagement Express*. Na aplicação utilizou-se o software "*Microsoft Visual Studio 2005*" e a Plataforma *Asp.net*, que é constituída por um conjunto de classes cujo principal objectivo é servir pedidos efectuados através do protocolo *http*. Descreve-se também o funcionamento da aplicação.

O quarto capítulo é a comparação com outras aplicações. Descreve-se algumas das aplicações analisadas e descreve-se a diferença entre a aplicação "*Navison*", *Microsoft Dynamics CRM Verson 3.0.* e aplicação EstgCRM desenvolvida.

O último capítulo contém a Conclusão e Perspectivas de Desenvolvimento. Na conclusão constam as conclusões gerais deste relatório e nas Perspectivas de Desenvolvimento apresentam-se perspectivas futuras de desenvolvimento.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

2. Analise e Concepção do Sistema *EstgCRM*

2.1. O que se entende por CRM

O que é CRM?

CRM representa a Gestão do Relacionamento com o Cliente e é normalmente usado para se referir a três coisas:

- A toda a gestão do relacionamento com o cliente, isto é, a funções como marketing, vendas, e apoio do cliente.

- As ferramentas usadas funcionam como gestoras de vendas.

- Processos que envolvem e gerem a relação com o cliente.

Assim, o nome CRM foi "inventado" para se referir à classe de ferramentas então introduzidas, que gerem funções de *cliente-contacto*.

A Gestão do relacionamento com o cliente (CRM), tem a ver com processos e funções que circundam as funções *cliente-orientadas* dentro da organização, como vendas, marketing e apoio ao cliente.

Vantagens (benefícios):

- Sendo uma solução de gestão de relacionamento com o cliente, uma das grandes melhorias que introduz está precisamente nessa vertente, pois o cliente vai sentir que as empresas estão

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

mais preparadas e têm mais informação para o servir melhor e dar-lhe respostas mais rápidas;

- Mais informação para gestão;

- Mais informação para os comerciais;

- Informação mais rápida e fácil;

- Aumento da capacidade de resposta;

- Maior capacidade de antecipação;

- O CRM ajuda a tomar decisões, porque nos permite uma melhor exploração de dados para análise;

- Permite automatizar e melhorar processos de que se dispunha, mas que eram sobretudo manuais, com grande impacto em termos de integração;

- Mudanças para melhor a nível da partilha da informação;

- Relatórios e indicadores de análise, que permitirão ter mais informação de gestão;

-Torna-se mais fácil saber os dados actualizados do cliente, alguns dos quais particularmente dirigidos à melhoria do relacionamento (como datas de aniversário);

- Este tipo de aplicação permite rapidamente saber como está a empresa e o mercado, numa zona ou num tipo de produto, sendo preciso para isso, assegurar que os dados estão disponíveis, ou seja, que a aplicação está a ser utilizada correctamente.

Devem-se no entanto, evitar mudanças abruptas. As vantagens deste tipo de software ainda não são aproveitadas totalmente, é que, mesmo que o software seja fácil de usar, uma mudança de situação e a sua adesão é sempre vista com apreensão por parte dos utilizadores.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Só à medida que a aplicação vai ganhando maturidade é que o interesse por esta vai aumentando, bem como o reconhecimento da sua utilidade no dia-a-dia.

2.2. Levantamento de requisitos

De acordo com o que foi proposto, o sistema implementará funções (casos de uso) que serão desempenhadas pelo Responsável da Campanha (Administrador) e pelos seus Clientes.

A seguir apresentam-se as siglas e respectivas definições usadas (Tabela 1):

CRM	Gestão do relacionamento do cliente.
Responsável Campanha	Pessoa encarregue da criação e gestão da campanha, pelo envio e criação dos inquéritos e por analisar as respostas.
Cliente	Representa um cliente que efectuou um pré-registo (através da *Internet*) no site da aplicação "*EstgCRM*". Vai receber e responder aos inquéritos. È a pessoa que vai comprar o produto promovido pela campanha, ou seja, a quem se destina a campanha.
Segmentação	A segmentação permite a identificação de grupos de clientes ou contactos com base no seu perfil, preferências, comportamentos e necessidades. Ao fazermos a segmentação para nos permitir efectuar campanhas especificamente direccionadas para um determinado tipo de cliente.
Fase	A campanha está dividida em fases, ou seja, etapas. A divulgação de fases da campanha, uma dessas fases é o lançamento de inquéritos.

Tabela 1 - Definições

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

No levantamento de requisitos começou-se por fazer uma lista dos casos de uso. Assim, os casos de uso principais são: Criar Campanha, Registar Cliente, Registar Produto, Definir Segmento, Definir Meio Comunicação, Criar Inquérito, Lançar Campanha, Registar Resposta Inquérito, Registar Oportunidade de Venda, Registar Concorrência, Visualizar Venda e Analisar Resultados Inquérito.

Formatada: Título 2; Char Char

2.3. Descrição dos Casos de Uso

Para uma melhor descrição dos vários casos de uso acima referidos, vamos usar o *Template* de casos de uso.

Apresenta-se seguidamente os casos de uso mais importantes: Tabelas 2, 3,4,5,6,7 e 8.

Caso de uso 1 – Criar Campanha

Nome:	Criar Campanha
Descrição:	Inicialização de todo o processo, ou seja, inicialização da campanha.
Actores envolvidos:	Responsável Campanha
Pré condições:	Efectuar *login* correctamente.
Pós condições:	
Fluxo de Eventos:	1. O Responsável da Campanha vai introduzir todos os dados da respectiva campanha: Nome, Descrição, Data de início e de fim; 2. Selecciona o segmento, os produtos e o responsável pretendido. 3. Selecciona a fase em que a campanha se encontra. 4. Guarda a campanha.
Fluxos alternativos:	2.a. Quando o segmento e produtos ainda não está criado, tem de ser criado para posteriormente ser utilizado na campanha. 3.a. Quando ainda não existem fases criadas, têm de ser criadas antes

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

	da campanha para posteriormente serem usadas na mesma.

Tabela 2 - Criar Campanha

Caso de uso 4 – Definir Segmento

Nome:	Definir Segmento
Descrição:	Permitir a segmentação de acordo com as necessidades do utilizador, ou seja, criar um segmento de dados, de modo a que os clientes "alvo" tenham o perfil de possíveis compradores do (s) produto (s).
Actores envolvidos:	Responsável Campanha.
Pré condições:	Efectuar *login* correctamente.
Pós condições:	
Fluxo de Eventos:	1. O caso de uso começa quando o Responsável selecciona a opção Adicionar segmento 2. O sistema disponibiliza os segmentos já criados e permite a inserção de condições de segmentação (linhas ao segmento) 3. O Responsável introduz a(s) condição(ões) de segmentação: selecciona Atributo - perfil do cliente alvo; Operador de comparação (>, <, =, …); Valor a comparar e Operador lógico para indicar se tem mais condições (and, or, …, branco) 4. O sistema associa e disponibiliza os clientes do segmento 5. O Responsável confirma.
Fluxos alternativos:	3.a. Quando ainda não existem Clientes criados, têm de ser criados antes de associarmos os Clientes ao respectivo segmento. 5.a. O Responsável cancela e o segmento não é criado.
Relacionamentos:	
Suplementos:	

Formatada: Parágrafo da Lista

Tabela 3 - Definir Segmento

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Caso de uso 5 - Definir Meio Comunicação

Nome:	Definir Meio Comunicação
Descrição:	Registar os tipos de meios de comunicação para posteriormente serem utilizados na fase da respectiva campanha.
Actores envolvidos:	Responsável Campanha
Pré condições:	Efectuar *login* correctamente.
Pós condições:	
Fluxo de Eventos:	
Fluxos alternativos:	
Relacionamentos:	
Suplementos:	

Tabela 4 - Template do Caso de uso 5 – Definir Meio Comunicação

Caso de uso 6 - Criar Inquérito

Nome:	Criar Inquérito
Descrição:	Registar os dados do inquérito. Seleccionar o inquérito, escrever a descrição d mesmo, associar ou eliminar as perguntas ao inquérito com os vários tipos de resposta associados à respectiva pergunta. Associar esse inquérito a uma dada campanha ou desassociá-lo.
Actores envolvidos:	Responsável Campanha
Pré condições:	Efectuar *login* correctamente. Existência de uma Campanha.
Pós condições:	
Fluxo de Eventos:	Permite criar perguntas com vários tipos de respostas para associar ao inquérito. Ao criar-se um inquérito, vamos associar-lhe as perguntas e associar esse mesmo inquérito a uma campanha.
Fluxos alternativos:	
Relacionamentos:	

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Suplementos:	

Tabela 5 - Template do Caso de uso 6 - Criar Inquérito

Caso de uso 7 - Lançar (fase) Campanha (referir na conclusão não foi atingido propor como trabalho futuro)

Nome:	Lançar (fase) Campanha
Descrição:	Quando se efectua o lançamento da fase(s) da campanha é necessário actualizar o campo "estado" da tabela campanha. Normalmente este campo é actualizado automaticamente isto é, quando a data do nosso sistema for igual à definida no campo "data de inicio" da tabela fase. Quando se procede ao lançamento de uma fase o campo "estado" da tabela campanha fica com o valor igual à descrição da campanha a lançar, ou seja, "Estado = DescCampanha". Esse campo também pode ser actualizado manualmente. É necessário que o campo "IDMeioComunicacao" da tabela fase esteja preenchido, de forma a ser possível o lançamento da fase.
Actores envolvidos:	Responsável Campanha e Clientes.
Pré condições:	Efectuar *login* correctamente. Existência de uma Campanha.
Pós condições:	
Fluxo de Eventos:	Divulgar a acção da fase da campanha, lançar inquéritos.
Fluxos alternativos:	
Relacionamentos:	
Suplementos:	

Tabela 6 - Template do Caso de uso 7 - Lançar (fase) Campanha

Caso de uso 8 - Registar Resposta Inquéritos

Nome:	Registar Resposta Inquéritos

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Descrição:	Registar as respostas dos clientes (público-alvo)
Actores envolvidos:	Responsável da Campanha e Clientes
Pré condições:	Efectuar *login* correctamente. Existência de respostas.
Pós condições:	Guardar as respostas dadas pelos clientes ao inquérito.
Fluxo de Eventos:	Introdução das respostas dadas pelos Clientes
Fluxos alternativos:	
Relacionamentos:	
Suplementos:	

Tabela 7 - Template do Caso de uso 8 - Registar Resposta Inquéritos

Caso de uso 11 – Visualizar Venda

Nome:	Visualizar Venda
Descrição:	Consultar ou visualizar as vendas feitas após o início da Campanha
Actores envolvidos:	Responsável Vendas
Pré condições:	Efectuar *login* correctamente. Existência de vendas.
Pós condições:	----------
Fluxo de Eventos:	Consultar as vendas efectuadas.
Fluxos alternativos:	
Relacionamentos:	
Suplementos:	

Tabela 8 - Template do Caso de uso 11 - Visualizar Venda

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Formatada: Título 2; Char Char

2.4. Diagramas UML que foram utilizados

Ao fazer-se a análise de sistemas, simplifica-se a realidade para um melhor entendimento do sistema que estamos a desenvolver.

Os Diagramas UML utilizados são compostos pelos seguintes tipos:

- Diagramas de Casos de Uso;
- Diagrama de Actividades;
- Diagramas de Interacção de Sequência;
- Diagramas de Interacção de Colaboração;
- Diagrama de Classes;
- Diagrama de Estados;
- Diagrama de Componentes.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

2.5. Diagrama de Casos Uso

Os casos de uso constituem a técnica em UML para representar o levantamento de requisitos de um sistema. Um requisito representa o comportamento esperado pelo sistema.

Desde sempre que o correcto levantamento de um requisito no desenvolvimento de sistemas de informação tenta garantir que o sistema será útil para o utilizador final, estando de acordo com as suas necessidades.

Um diagrama de casos de uso ilustra um conjunto de casos de uso para um sistema, os actores e a relação entre os actores e os casos de uso [Larman, 2000].

Os casos de uso são utilizados para modelar o modo de funcionamento do sistema ou empresa, ou o modo como gostariam que ele funcionasse.

Os casos de uso são assim, geralmente o ponto de partida da análise orientada a objectos utilizando a UML. O modelo do diagrama de casos de uso consiste em actores e casos de uso. Actores que representam utilizadores e outros sistemas que interagem com o sistema e casos de uso que mostram o comportamento do sistema, ou seja, os cenários que o sistema percorre em resposta ao estímulo de um actor.

O relacionamento entre um actor e um caso de uso representa a participação deste actor no caso de uso.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Eliminado: 5

O Diagrama de Casos de Uso (Figura 2.1) que representa o sistema *EstgCRM* é o seguinte:

Figura 2-1 - Diagrama de Casos Uso do Sistema EstgCRM

Eliminado: 5

Eliminado: 5

Este diagrama (Figura 2.1) vai modelar o funcionamento do sistema EstgCRM. O diagrama representa os casos de uso, os actores e a sua relação.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

2.6. Diagrama de Actividades

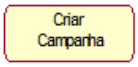
Um diagrama de actividade decompõe uma actividade em sub actividades, com fluxo de controlo sequencial ou concorrente entre sub actividades, ou seja, os diagramas de actividade mostram o fluxo de controlo de uma actividade para outra.

A actividade que está a ser decomposta pode ser: um caso de uso, uma operação de uma classe, um grupo de casos de uso relacionados entre si.

Este tipo de diagrama pode ser dividido em pistas de responsabilidade (*swimlanes* – mostram quem tem a responsabilidade de algo), separadas por linhas contínuas. Cada pista tem o nome da unidade organizacional, entidade ou objecto responsável pelas acções e actividades aí localizadas. Cada acção ou actividade é localizada numa única pista, mas uma transição pode atravessar várias pistas.

Assim, este diagrama serve para modelar fluxos de trabalho relativos a processos de negócio e mostra claramente qual o indivíduo ou grupo de indivíduos que é responsável por cada actividade.

Tipo de estados que o seguinte diagrama vai apresentar:

- Estado espera 

, a saída é causada por eventos;

- Estado decisão  , estado de passagem em que são testadas condições.

Eliminado: 6

O Diagrama de Actividades (Figura 2.2) descreve o funcionamento do sistema EstgCRM, através de actividades e refere os indivíduos responsáveis por cada actividade.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

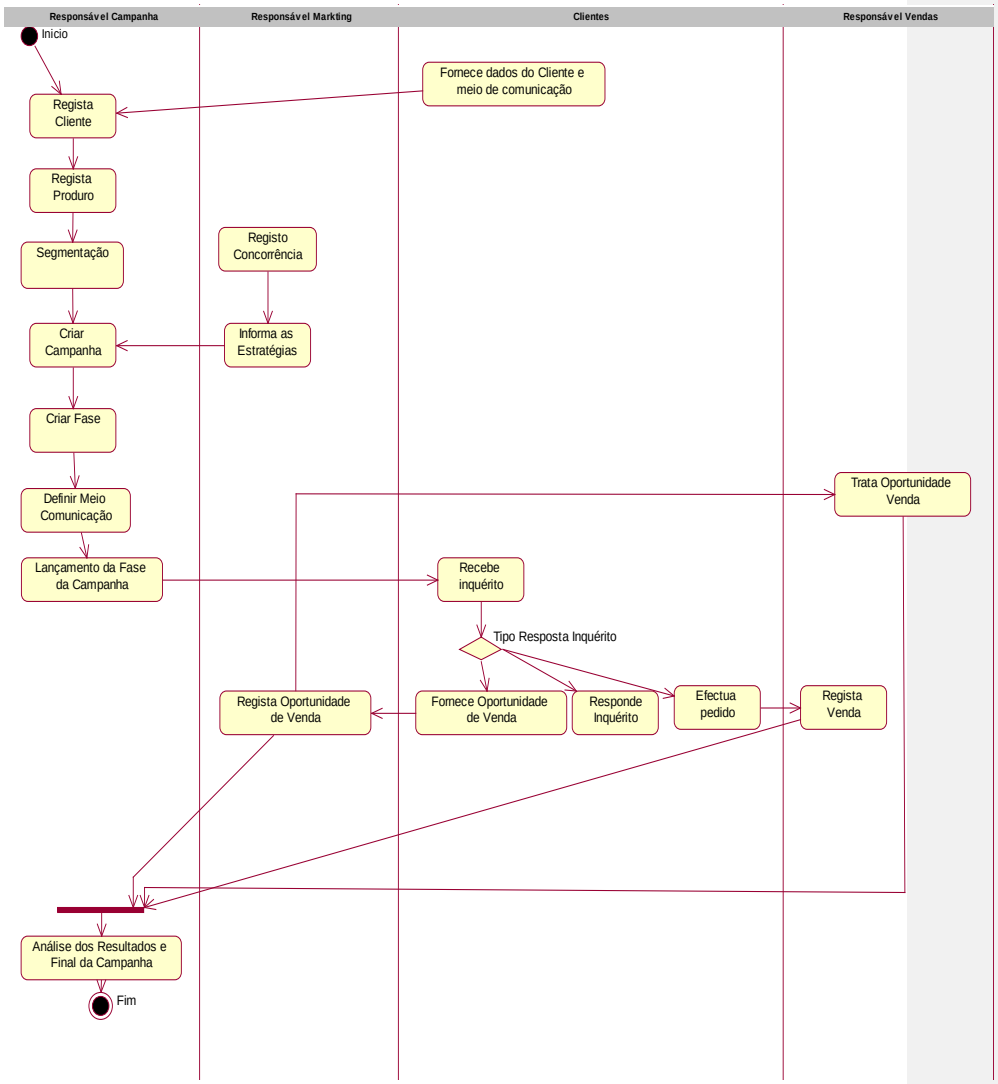


Figura 2-2 - Diagrama de Actividades do Sistema EstgCRM

Eliminado: 6

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

2.7. Diagramas de Sequencia

Estes diagramas são utilizados para documentar Casos Uso e representar interacções entre objectos num dado sistema.

Um objecto é representado por um rectângulo e uma linha vertical, denominada a linha de vida do objecto. Os objectos comunicam trocando mensagens representadas por setas horizontais orientadas do emissor de uma mensagem para o seu destinatário.

Os diagramas de sequência mostram as interacções entre objectos segundo uma *perspectiva temporal.*

Apresentamos seguidamente uma lista com os Diagramas de Sequencia elaborados:

- Criar Campanha (Figura 2.3)
- Definir Segmento (Figura 2.5)
- Criar Inquérito (Figura 2.7)
- Lançar Campanha (Figura 2.9)
- Registar Respostas Inquérito (Figura 2.11)
- Registar Oportunidades de Venda (Figura 2.13)

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

2.7.1 Criar Campanha

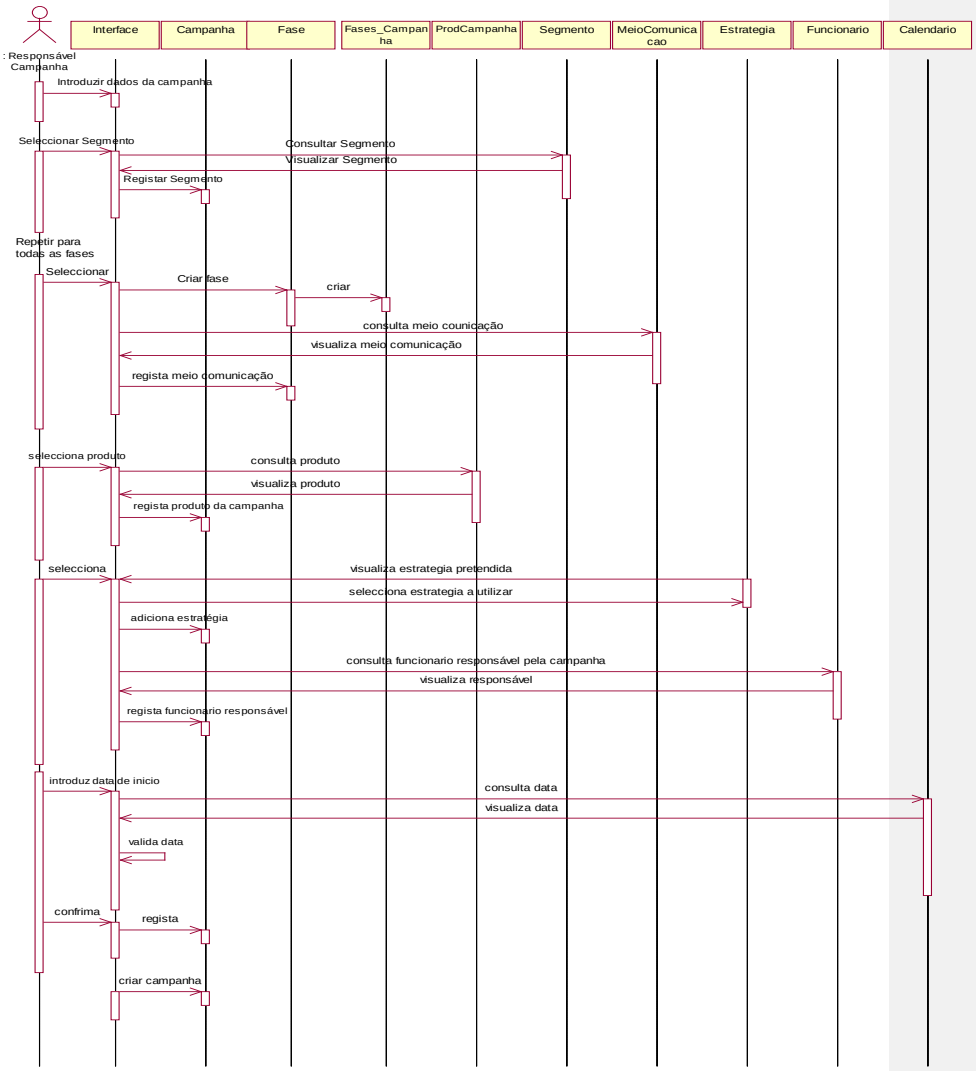


Figura 2-3 - Diagrama de Sequência - Criar Campanha

Eliminado: 2-3

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

● No diagrama de sequência "Criar Campanha" (Figura 2.3), introduzem-se os dados da campanha, consulta-se o segmento criado antes da campanha e regista-se. Criam-se as fases da campanha e na fase define-se e regista-se o meio de comunicação. Consulta-se o produto da campanha e regista-se. Selecciona-se e adiciona-se a estratégia a seguir pela campanha. Consulta-se o funcionário responsável pela campanha e regista-se. Introduz-se a data de início da campanha, valida-se e regista-se. De seguida elabora-se a criação da campanha.

No diagrama de classes (Figura 2.4) apresentam-se as classes que participam no diagrama de sequência "Criar Campanha".

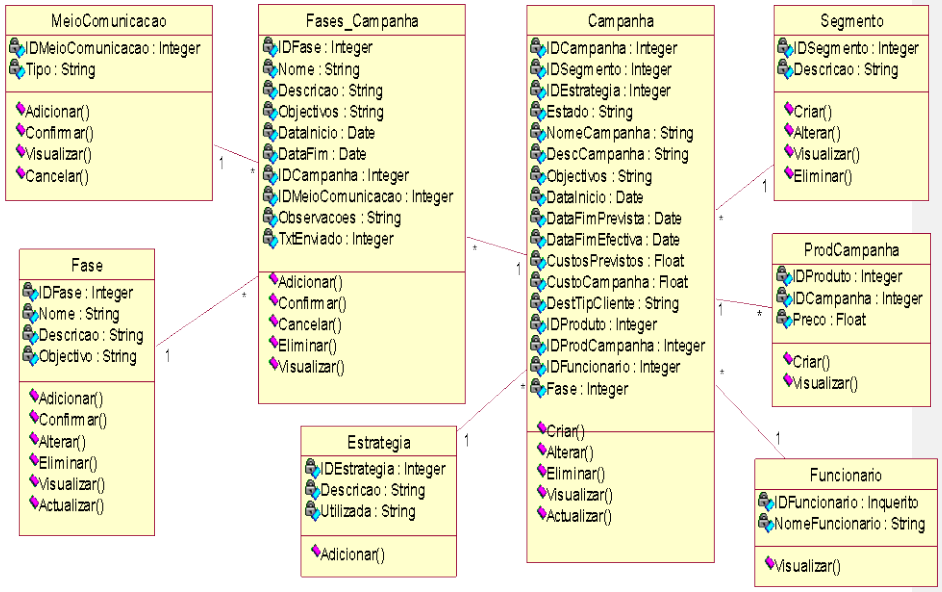


Figura 2-4 - Diagrama de Classes - Criar Campanha

Eliminado: 2-4

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

2.7.2 Definir Segmento

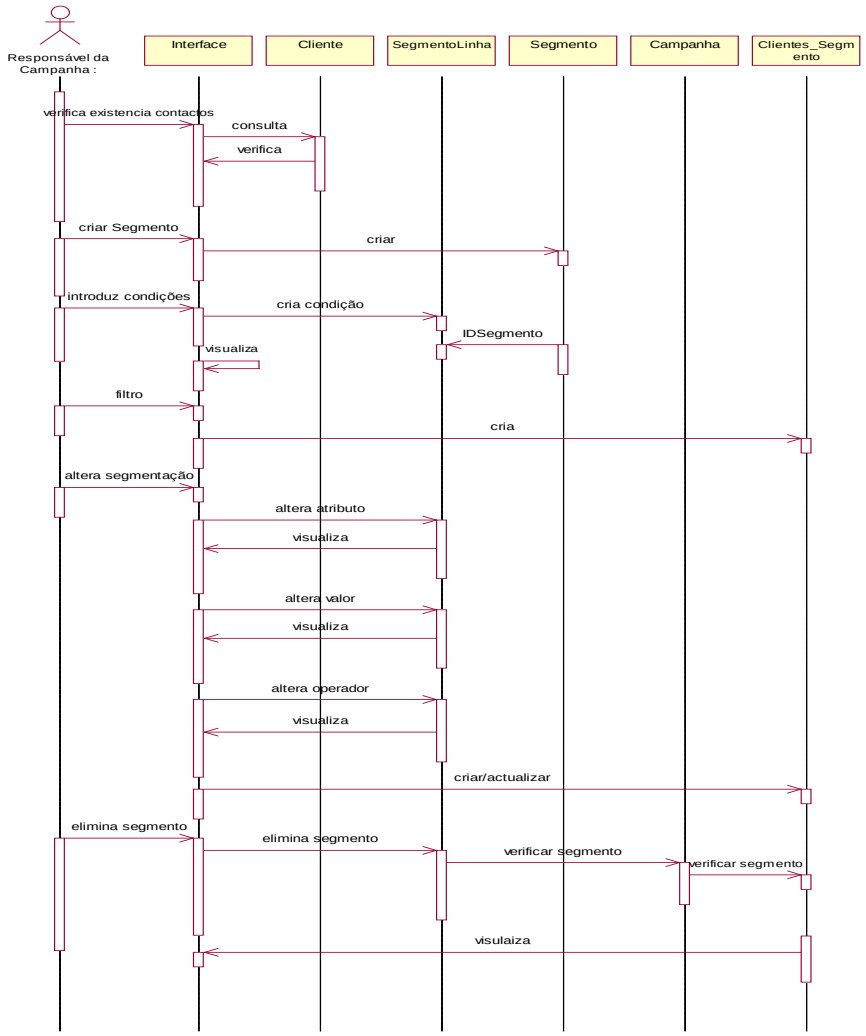


Figura 2-5 - Diagrama de Sequência - Definir Segmento

Eliminado: 2-5

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

● No diagrama de sequência "Definir Segmento" (Figura 2.5), o responsável da Campanha consulta e verifica a existência de contactos (clientes). Cria-se o segmento, introduzem-se e criam-se as várias condições do Segmento no SegmentoLinha, visualizam-se e filtram-se os dados do segmento, criando de seguida os clientes do segmento. Podendo-se alterar a segmentação no SegmentoLinha depois de alterar, pode-se ainda criar e actualizar os clientes do segmento. Pode-se efectuar também a eliminação do segmento.

No diagrama de classes (Figura 2.6) apresentam-se as classes que pertencem ao diagrama de sequência "Definir Segmento".

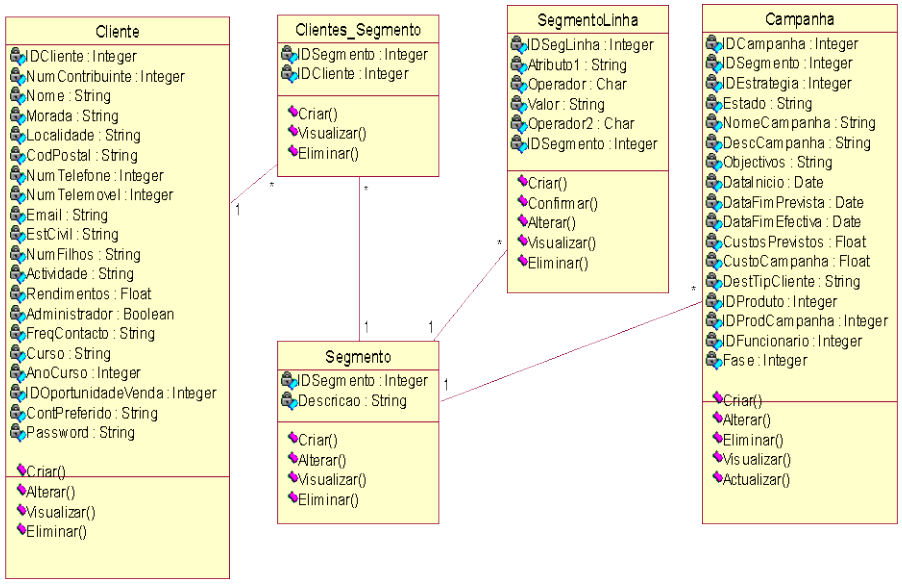


Figura 2-6 - Diagrama de Classes - Definir Segmento

Eliminado: 2-6

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

2.7.3 Criar Inquérito

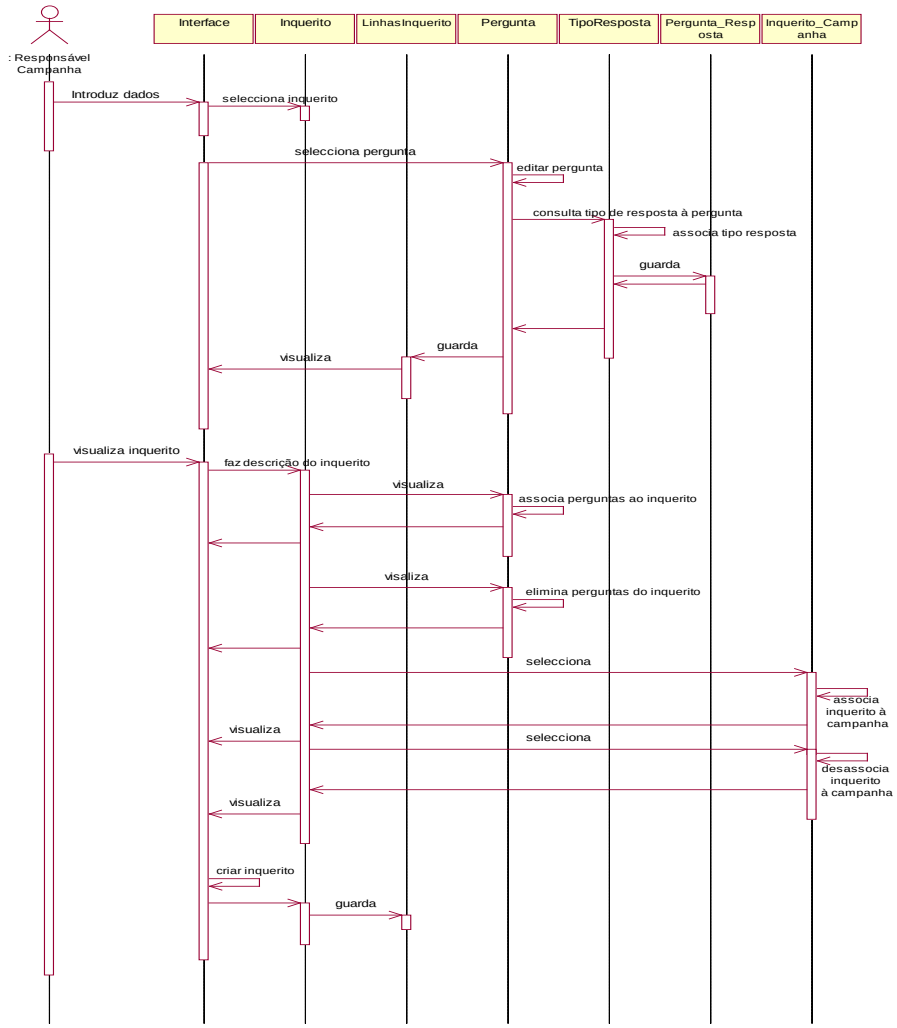


Figura 2-7 - Diagrama de Sequência – Criar Inquérito

Eliminado: 2-7

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

No diagrama de sequência "Criar Inquérito" (Figura 2.7), começam-se por criar as perguntas que posteriormente podemos ou não associar ao inquérito. Às perguntas vamos associar tipos de resposta. Visualiza-se assim, o inquérito que pretendemos criar, faz-se a descrição do respectivo inquérito. Associam-se as perguntas ao inquérito. Associa-se o inquérito a uma campanha. Finalmente criar-se o inquérito.

No diagrama de classes (Figura 2.8) apresentam-se as classes que pertencem ao diagrama de sequência "Criar Inquérito".

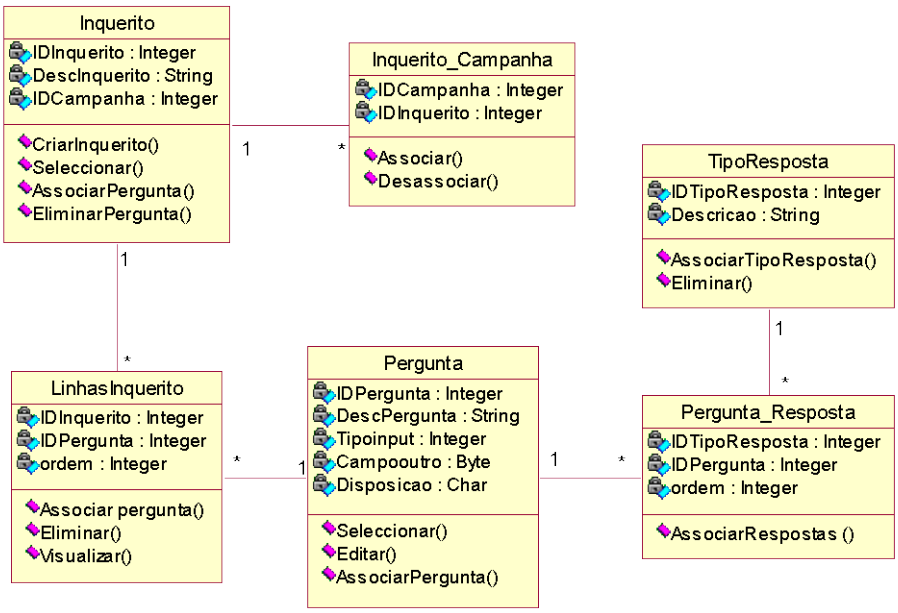


Figura 2-8 - Diagrama de Classes – Criar Inquérito

Eliminado: 2-8

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

2.7.4 Lançar (fase) Campanha

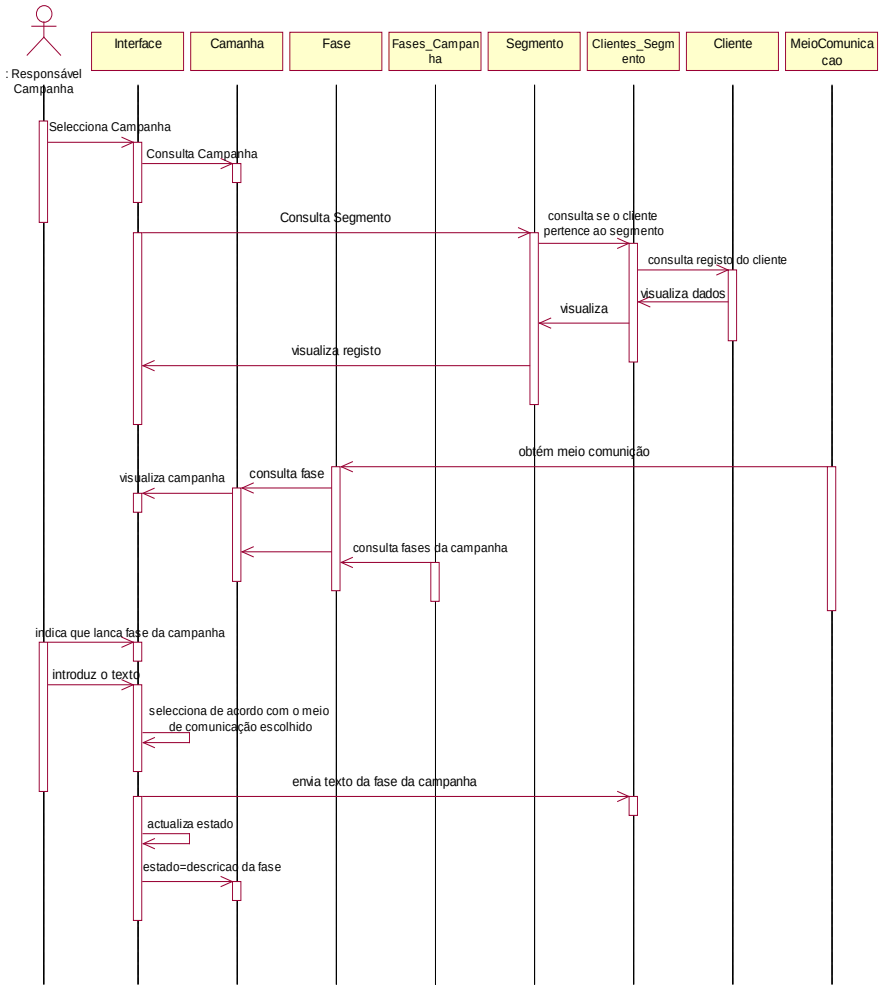


Figura 2-9 - Diagrama de Sequência - Lançar (fase) Campanha

Eliminado: 2-9

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

● No diagrama de sequência "Lançar (fase) Campanha" da (Figura 2.9), selecciona-se a campanha que se pretende, consulta-se o segmento e os clientes do segmento, obtém-se a fase, o meio de comunicação, lança-se a fase da campanha de acordo com o meio de comunicação escolhido, envia-se o texto e actualiza-se o estado em que se encontra a campanha.

No diagrama de classes (Figura 2.10) apresentam-se as classes que pertencem ao diagrama de sequência "Lançar (fase) Campanha".

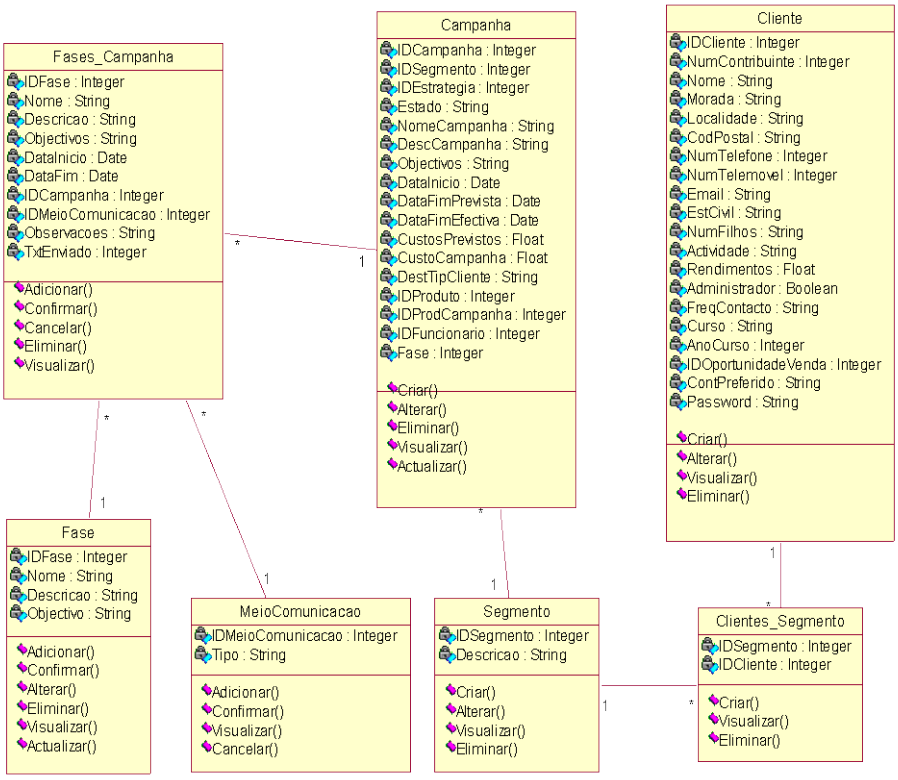


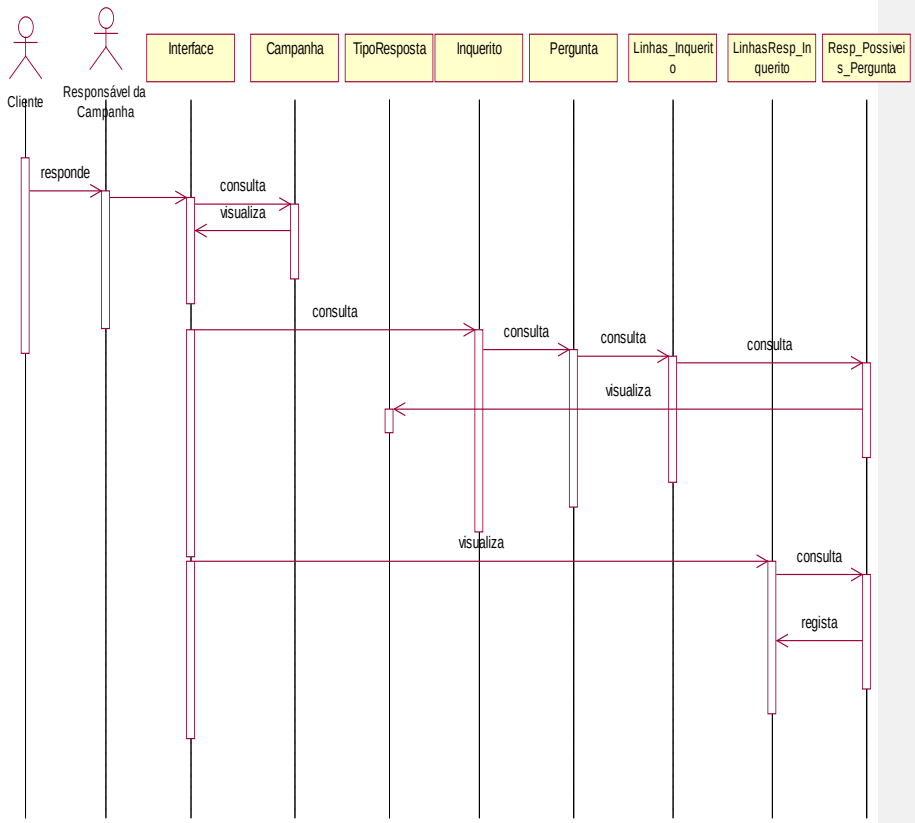
Figura 2-10 - Diagrama de Classes - Lançar (fase) Campanha

Eliminado: 2-10

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

2.7.5 Registar Respostas Inquérito



Eliminado: 2-11

Figura 2-11 - Diagrama de Sequência - Registar Respostas do Inquérito

● No diagrama de sequência "Registar Respostas do Inquérito" (Figura 2.11), o cliente vai responder ao inquérito, o funcionário responsável da campanha vai verificar a que campanha pertence o inquérito, consulta o inquérito, as perguntas e as várias respostas possíveis a cada pergunta, visualiza o tipo de resposta e as linhas das respostas onde se encontram registadas

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

as respostas propriamente ditas do inquérito, consulta as vária respostas possíveis às perguntas e efectua o registo das mesmas nas "LinhasResp_Inquerito".

No diagrama de classes da (Figura 2.12) apresentam-se as classes que pertencem ao diagrama de sequência "Registar Respostas do Inquérito".

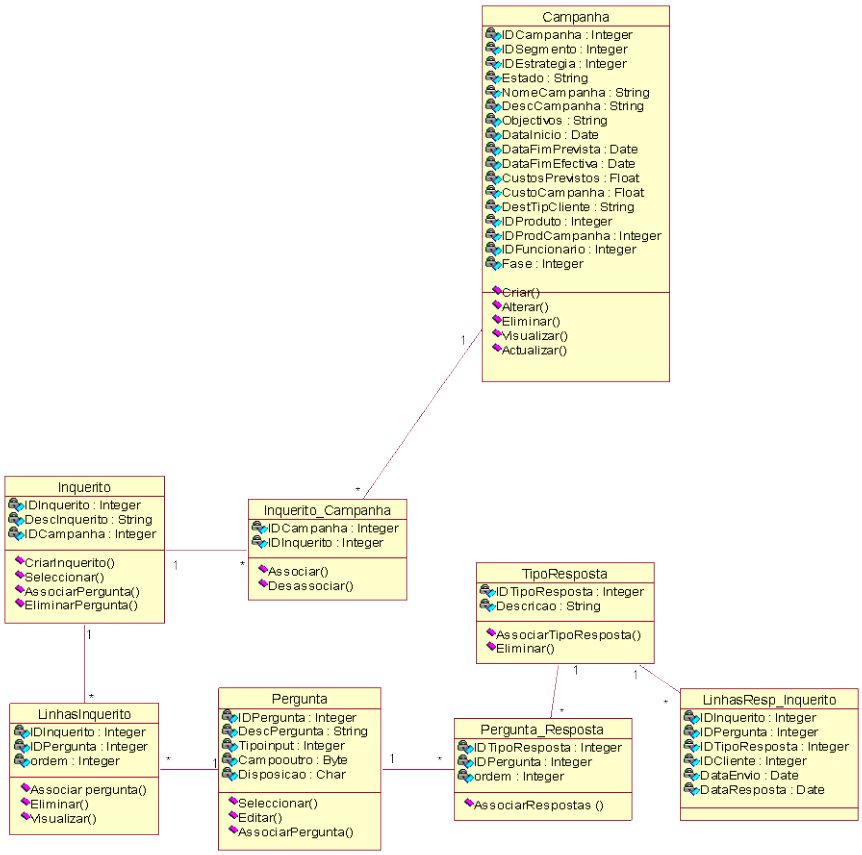


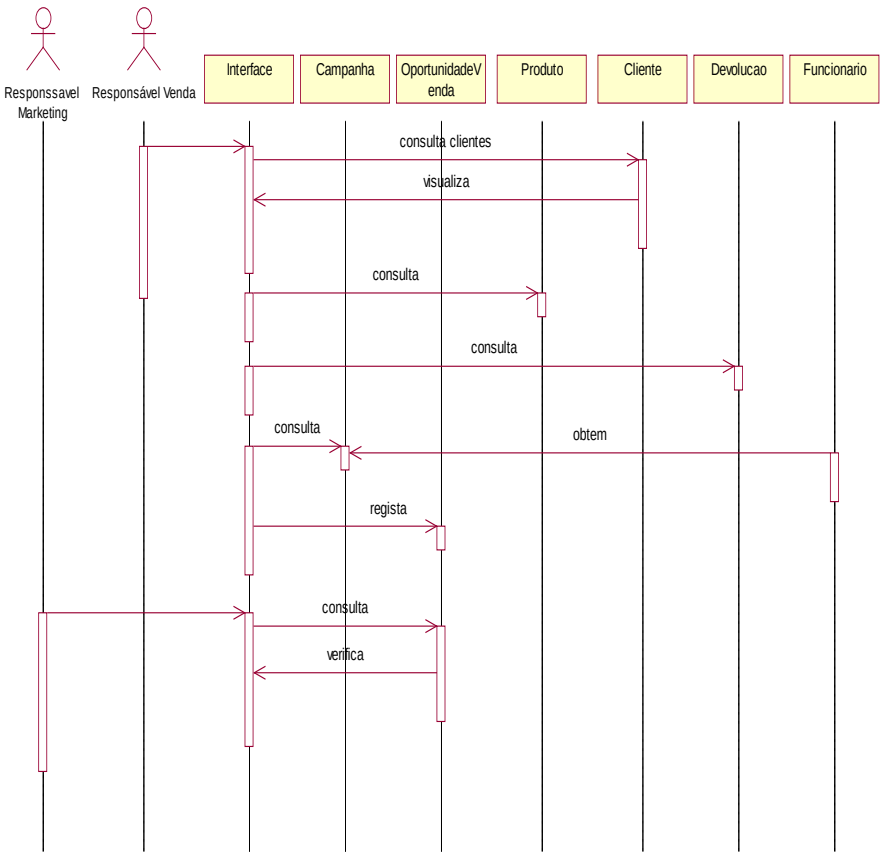
Figura 2-12 - Diagrama de Sequência - Registar Respostas do Inquérito

Eliminado: 2-12

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

2.7.6 Registar Oportunidade Venda



Eliminado: 2-13

Figura 2-13 - Diagrama de Sequência - Registar Oportunidade Venda

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

● No diagrama de sequência "Registar Oportunidade Venda" (Figura 2.13), o responsável de venda consulta os clientes, o produto, as devoluções efectuadas e a campanha, na campanha obtém o funcionário que é responsável pela mesma e efectua o registo da oportunidade de venda. Por sua vez, o responsável de marketing vai consultar e verificar a oportunidade de venda.

No diagrama de classes (Figura 2.14) apresentam-se as classes que pertencem ao diagrama de sequência "Registar Oportunidade Venda".

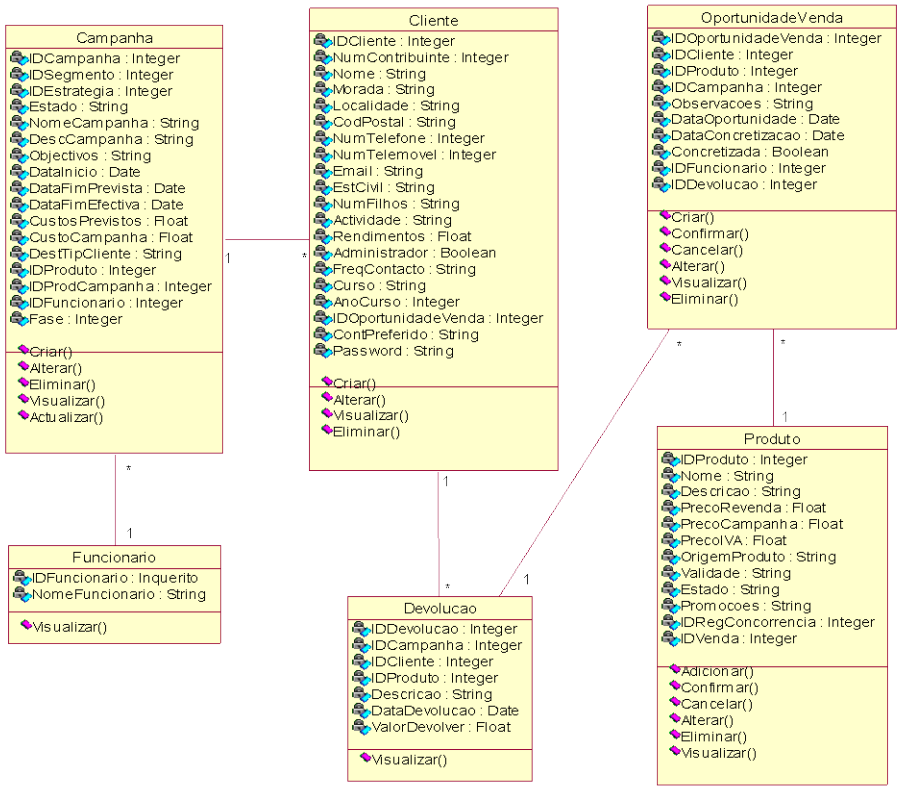


Figura 2-14 - Diagrama de Classes Registar Oportunidade Venda

Eliminado: 2-14

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

2.8. Diagrama de Colaboração - Definir Segmento

Este tipo de diagrama é uma alternativa para modelar as de interacções entre os objectos de um sistema. Enquanto, que o diagrama de sequência focaliza-se na sequência cronológica do cenário que está a ser modelado, o diagrama de colaboração focaliza-se no relacionamento entre os objectos e na compreensão dos efeitos nos objectos no decorrer de um cenário.

Os objectos são ligados através de associações e cada associação representa uma instância da mesma associação entre as respectivas classes envolvidas.

As associações mostram as mensagens enviadas entre os objectos e a sequência destas mensagens é determinada usando-se números sequências (Figura 2.15).

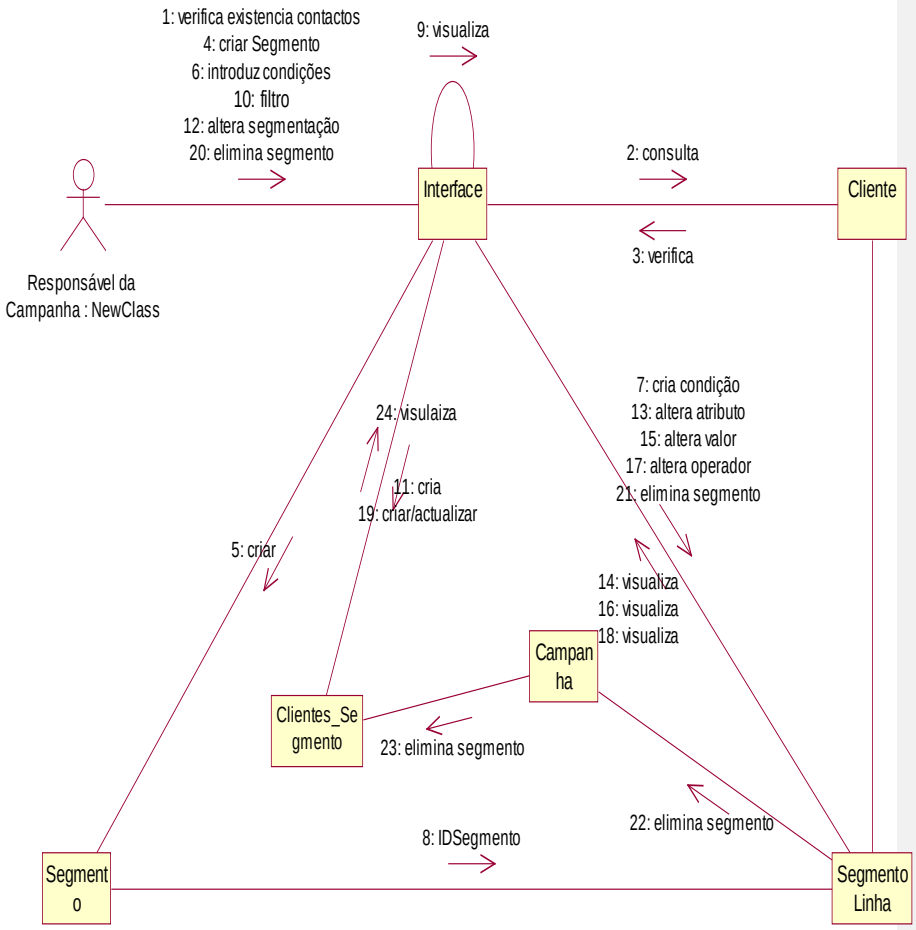
Lista de Diagramas de Colaboração:

Definir Segmento.

Nota: Após a elaboração dos diagramas de sequência, a criação dos diagramas de colaboração é bastante simples. Assim, caso a elaboração dos nossos diagramas esteja a ser realizada na ferramenta *Rational Rose* basta posicionarmo-nos num diagrama de sequência e premir a tecla F5.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt



Eliminado: 2-15

Figura 2-15 Diagrama de Colaboração - Definir Segmento

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

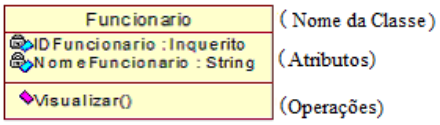
2.9. Diagrama de Classes (geral)

Um diagrama de classes é um conjunto de classes, interfaces, colaborações e respectivas relações, em geral de dependências, generalização e de associação.

Os Diagramas de Classes são usados para modelar a estrutura do sistema, ou seja, é uma descrição formal da estrutura de objectos num sistema.

Para cada objecto descreve a sua identidade, os seus relacionamentos com os outros objectos, os seus atributos e as suas operações.

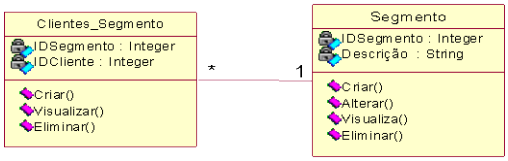
Uma classe em UML é representada do seguinte modo (Figura 2.16):



Eliminado: 2-16

Figura 2-16 Exemplo da representação de uma "Classe em UML"

As associações representam relacionamentos estruturados entre objectos de diferentes classes, esses relacionamentos são representados graficamente através de uma linha ligando as classes. As extremidades da linha que representa uma associação mostram como a classe é vista pelas outras classes na associação. Como mostra a Figura 2.17.



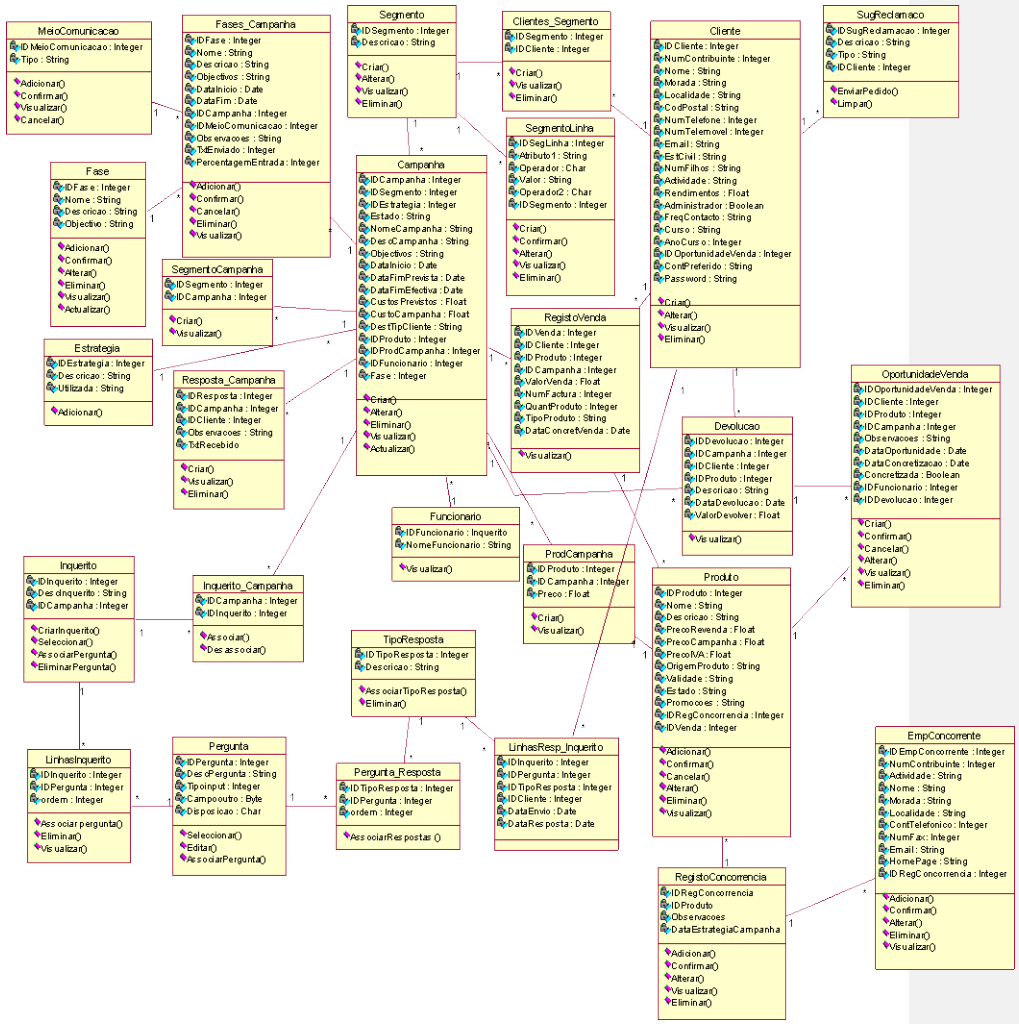
Eliminado: 2-17

Figura 2-17 Relação de um para muitos.

O seguinte Diagrama de Classes da Figura 2.18 representa todas as classes e suas relações.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt



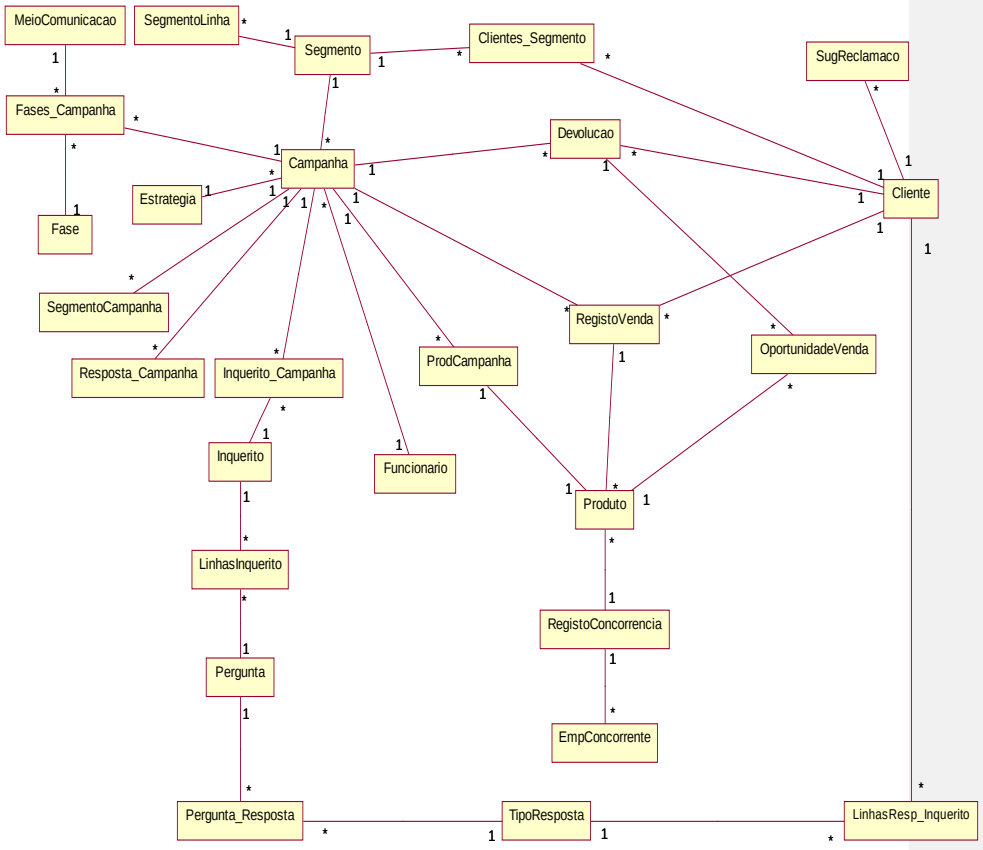
Eliminado: 2-18

Figura 2-18 Diagrama de Classes do Sistema EstgCRM

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

No Diagrama de Classes (Figura 2.19), aparecem apenas os nomes das classes, para uma melhor visualização das ligações (relações) entre as várias tabelas (classes) suprimiu-se os atributos e as operações.



Eliminado: 2-19

Figura 2-19 Diagrama de Classes do Sistema EstgCRM, onde aparecem apenas os nomes da coluna.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

2.10. Diagrama de Estados

O Diagrama de estados, também conhecido por diagrama de transição de estado ou por máquina de estados, permite modelar o comportamento interno de um determinado objecto, subsistema ou sistema global. Estes diagramas usam-se para classes de objectos que têm uma grande quantidade de comportamento dinâmico

Estes diagramas representam os possíveis estados de um objecto, as correspondentes transições entre os estados, os eventos que fazem desencadear as transições, e as operações (acções e actividades) que são executadas dentro de um estado ou durante uma transição. Os objectos evoluem ao longo do tempo através de um conjunto de estados como resposta a eventos e à passagem de tempo.

Para elaborarmos um diagrama de estados é necessário saber o que é:

Estado – é uma situação registada por um objecto durante o seu respectivo ciclo de vida, durante a qual uma condição é verificada, vai executando alguma actividade, ou simplesmente espera que determinado evento ocorra.

Transição – é uma relação entre dois estados que especifica que um objecto que se encontre no primeiro estado, realizará um conjunto de acções e mudará para o segundo estado quando um determinado evento ocorrer e determinadas condições se verificarem.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

O diagrama da Figura 2.20, permite-nos modelar o comportamento do sistema EstgCRM.

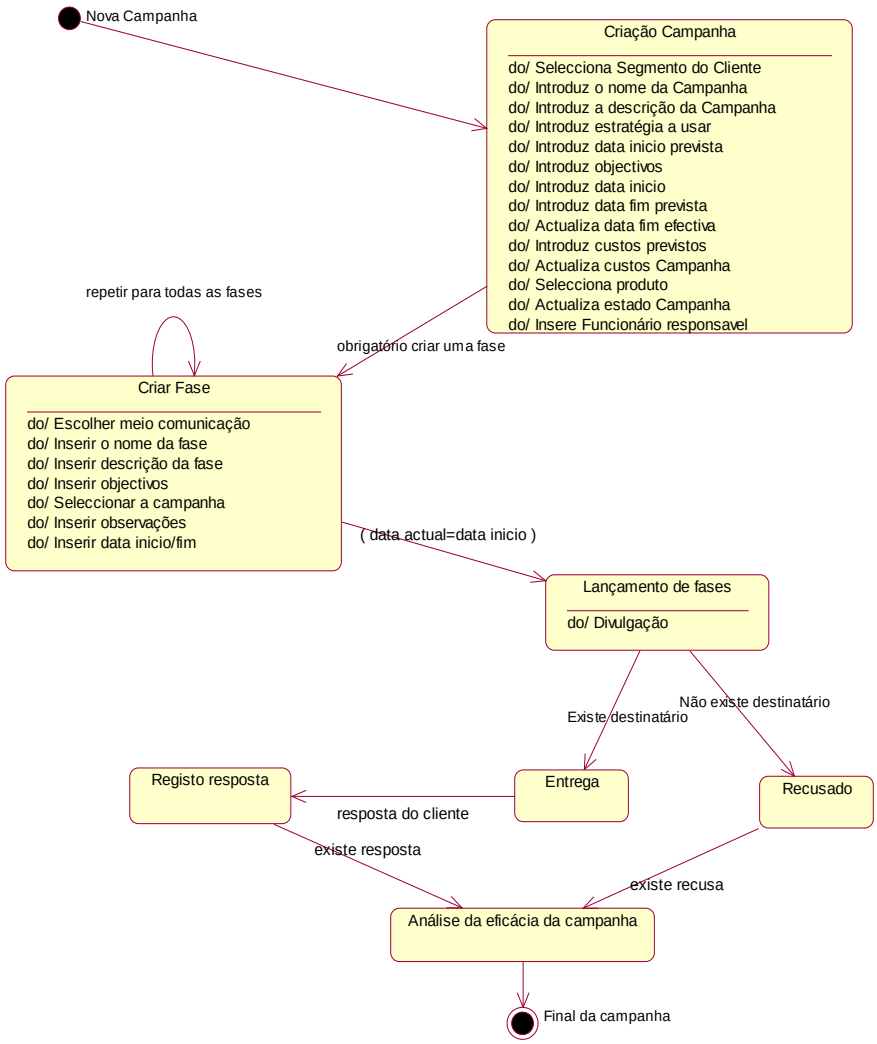


Figura 2-20 Diagrama de Estados do Sistema EstgCRM

Eliminado: 2-20

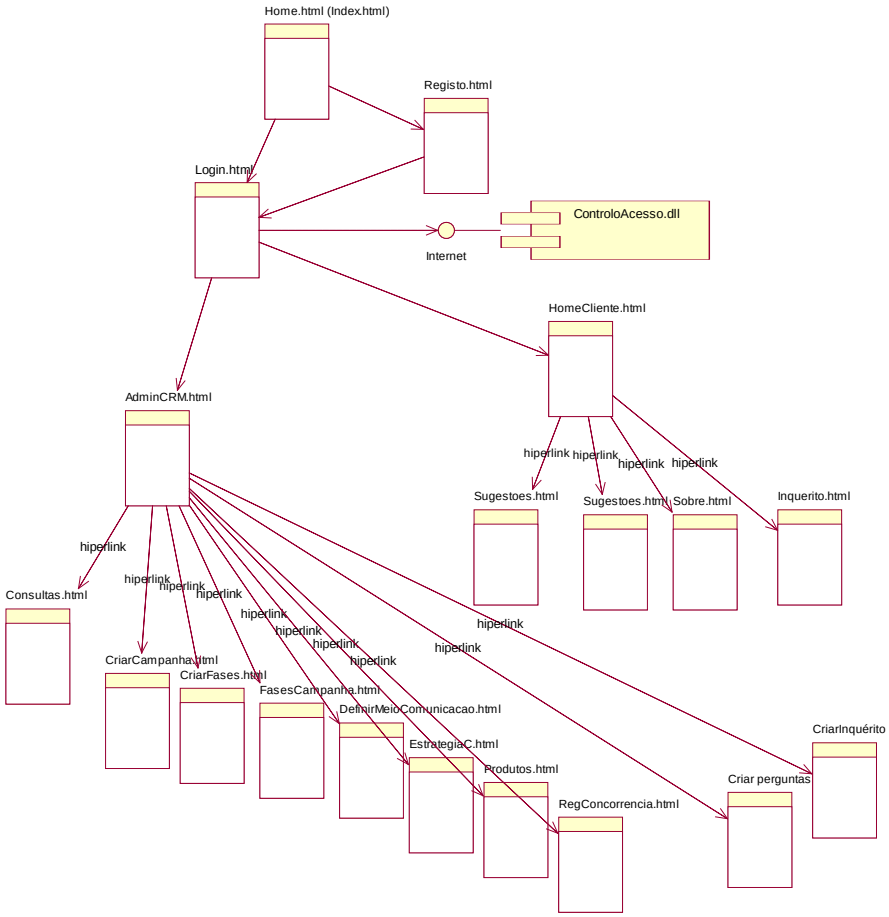
Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

2.11. Diagrama de Componentes

Diagramas de Componentes (Figura 2.21) mostram dependências entre componentes de software [*Furlan,1998*]. Este diagrama é utilizado para descrever a arquitectura da aplicação informática em termos de componentes de *software.*

Formatada: Tipo de letra: Itálico



Eliminado: 2-21

Figura 2-21 Diagrama de Componentes do Sistema EstgCRM

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

2.12. Diagrama de Instalação

O diagrama de Instalação (Figura 2.22) permite descrever a arquitectura do equipamento informático utilizado e a distribuição dos componentes da aplicação pelos elementos da arquitectura. Na prática, permite demonstrar como o hardware vai estar organizado e como os componentes (software) estarão distribuídos, estabelecendo assim a sua relação física.

Este diagrama de instalação define três componentes que comunicam entre si.

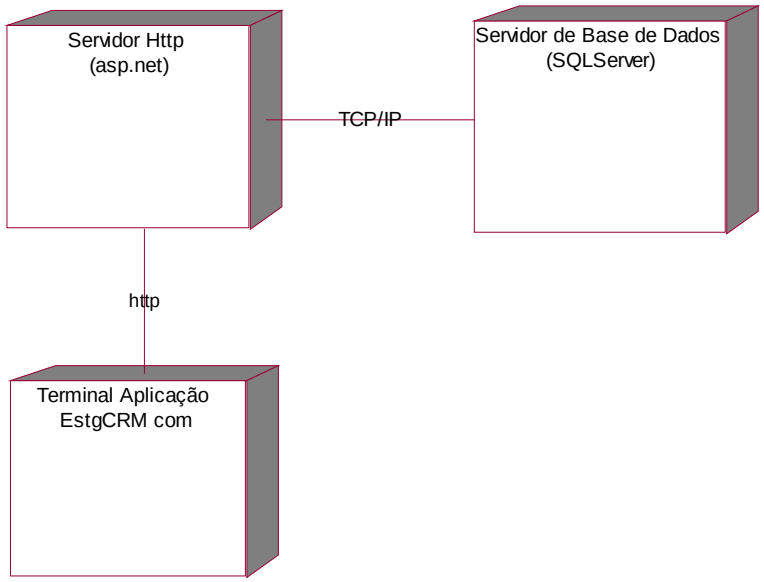


Figura 2-22 Diagrama de Instalação da Aplicação EstgCRM.

Eliminado: 2-22

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

2.13. Dicionário de Dados ou Semântica das Classes

No dicionário de dados vamos referir em pormenor todas as classes e seus respectivos nomes, atributos, operações e Diagramas de Sequência em que participa.

Em UML as classes são representadas por um rectângulo dividido em três compartimentos:

(1) - Nome: que contém apenas o nome da Classe;

(2) - Atributos: que possui a relação dos atributos que a classe possui;

(3) - Operações: que serão os métodos de manipulação de dados e de comunicação de uma classe com as outras classes do sistema (Figura 2.23).

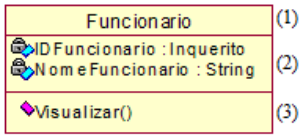


Figura 2-23 Exemplo de uma classe UML

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Classe Campanha:

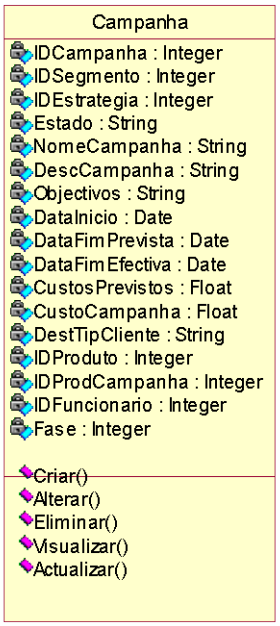


Figura 2-24 Classe Campanha

Eliminado: 2-24

Diagramas de Sequência em que participa:

Criar Campanha, Definir Segmento, Definir Meio de Comunicação, Lançar (fase) Campanha, Registar Resposta da Campanha, Registar Respostas dos Inquéritos, Registar oportunidade Venda, Registar Venda, Análise Resultados, Análise final Campanha.

Atributos (descrição):

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

(1) Número sequencial que identifica a Campanha
(2) Número que identifica o Segmento utilizado pela Campanha
(3) Número que identifica a Estratégia utilizada na Campanha
(4) Dá-nos o estado em que a Campanha se encontra (estado que vai mudando conforme a data da campanha)
(5) Nome da respectiva Campanha
(6) Descreve em que consiste a Campanha
(7) Objectivo que se pretendem atingir com a campanha
(8) Data de início da Campanha
(9) Data prevista para o final da Campanha
(10) Data em que efectivamente a Campanha terminou
(11) Custos previstos para a realização da Campanha
(12) Custos em unidades monetárias que a Campanha realmente custou
(13) Tipo de Cliente a que a Campanha se destina
(14) Número que identifica o Produto da Campanha
(15) Número que identifica o responsável (funcionário) pela Campanha
(16) Fase em que a Campanha se encontra

Formatada: Legenda

Tabela 9 - Tabela: Campanha

Operações:

Criar () – criar a campanha, possibilita a criação de uma nova campanha na base dados

Gerar – número da Campanha (anterior +1);

IDSegmento – Selecciona segmento existente;

IDEstrategia – Selecciona a estratégia utilizada;

Formatada: Rodapé

IDFuncionario – selecciona o funcionário responsável pela campanha;

Estado = Inicio.

Introduz – NomeCampanha, DataInicioPrevista, Objectivos, DataFimPrevista, CustosPrevistos, DestTipoCliente (tipo de Cliente de destino da Campanha);

Valida – DataInicio > = datadia

Inicializa Custo Efectivo = Branco e DataFim = Branco

Alterar() – possibilita modificar campos referentes à campanha

Actualiza Estado: If data da fase > = data sistema então

Estado =”a decorrer”

Actualiza final da Campanha: DataFim = ”dia em que termina a campanha”

CustoCampanha =”custo total da Campanha”

Estado = ”fim”

Visualizar () – possibilita visualizar os dados que foram introduzidos no registo (classe) campanha, ou seja, é o método pelo qual se pode consultar a campanha

Responsabilidades:

Classe → Campanha

Responsabilidades → Guarda registos referentes à campanha, permite a visualização desses mesmos dados, permite também alterações em alguns campos da campanha. A classe campanha fornece os estados por que passa a campanha.

Formatada: Tipo de letra: 11 pt

Eliminado:

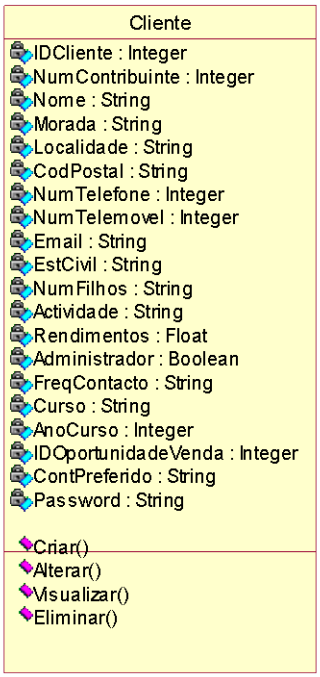
Eliminado: ¶

Eliminado:

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Classe Cliente:



Eliminado: FiguraIlustração 5429 -

Figura 2-25 Classe Cliente

Diagramas de Sequência em que participa:

Registar Cliente, Definir Segmento, Lançar (fase) Campanha, Registar Oportunidade Venda, Registar Venda.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Atributos (descrição):

(1) Número sequencial que identifica o cliente
(2) Número de contribuinte do Cliente
(3) Nome do Cliente
(4) Morada do Cliente
(5) Localidade do Cliente
(6) Código postal do Cliente
(7) Contacto telefónico do Cliente
(8) Contacto telemóvel do Cliente
(9) Email do Cliente
(10) Idade de Cliente
(11) Sexo do Cliente
(12) Estado civil do Cliente
(13) Número de filhos do Cliente
(14) Actividade desenvolvida pelo Cliente
(15) Rendimentos do Cliente
(16) Campo onde verificamos se o Cliente é ou não Administrador
(17) Frequência com que o Cliente deseja ser contactado
(18) Curso frequentado pelo Cliente
(19) Ano de curso frequentado pelo Cliente
(20) Número da oportunidade de venda que o Cliente pode proporcionar
(21) Contacto através do qual o Cliente prefere ser contactado
(22) Palavra passe do Cliente

Formatada: Legenda

Tabela 10- Tabela: Cliente

Operações ou Métodos:

Criar () – cria um novo registo para cada cliente na tabela Cliente.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Gerar – nº Cliente (anterior + 1);

Introduz – NumContribuinte, Nome, Morada, Localidade, CodPostal, NumTelefone, NumTelemóvel, Email, Idade, Sexo, EstCivil, NumFilhos, Actividade, Rendimentos, Administrador, FreqContacto, Curso, AnoCurso, ContPreferido, Password;

Valida – NumContribuinte=”nove algarismos”

Valida - Administrador=”True/False”

Alterar () – Modifica o registo do Cliente, possibilita alterar (editar) todos os campos existentes na tabela cliente excepto o IDCliente.

Actualiza: FreqContacto=”Nº dias”

NumContribuinte= “nº novo (com nove algarismos)”

NumTelefone=”nº novo (com nove algarismos)”

NumTelemovel=”nº novo (com nove algarismos)”

ContPreferido=”meio de comunicação”

Visualizar () – permite-nos visualizar e consultar os dados relativos ao registo do Cliente.

Eliminar () – Só posso eliminar um registo de um determinado cliente se não estiver a ser utilizados nas tabelas Clientes_Segmento, RegistoVenda, OportunidadeVenda, Devolucao.

Responsabilidades:

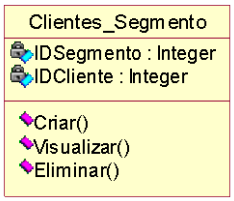
Classe → Cliente

Responsabilidades → Guardar os dados referentes ao cliente, alterar alguns dados do cliente, visualizá-los e eliminá-los.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Classe Clientes_Segmento:



Eliminado: FiguraIlustração 5530 -

Figura 2-26 Classe Clientes_Segmento

Diagramas de Sequência em que participa:

Definir Segmento.

Atributos (descrição):

(1) Identificação do número do Segmento do Cliente
(2) Identificação do número do Cliente que pertence ao Segmento

Formatada: Legenda

Tabela 11 - Tabela Clientes_Segmento

Operações:

Criar () – cria um novo segmento na base de dados

Gerar – número do Segmento (anterior+1);

IDCliente – Selecciona o Cliente;

Visualizar () – visualiza os campos da tabela Clientes_Segmento

Eliminar () – possibilita eliminar um registo da tabela Clientes_Segmento, se não estiver a ser usado nas tabelas Cliente e Resposta_Campanha.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Responsabilidades:

Classe → Clientes_Segmento

Responsabilidades → Guarda registos dos segmentos, cria e também permite visualizar.

Nota: Para se poder eliminar o registo, as tabelas SegmentoLinha, Segmento e Campanha não poderão conter esse registo.

Classe Devolucao:

Esta Classe (Tabela) é usada apenas para consulta.

Figura 2-27 Classe Devolucao

Eliminado: FiguraIlustração 5631 -

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Diagramas de Sequência em que participa:

Registar Oportunidade Venda, Análise Resultados.

Atributos (descrição):

(1) Número sequencial que identifica a Devolução
(2) Número que identifica a Campanha onde foi efectuada a devolução
(3) Número que identifica o cliente que efectuou a devolução
(4) Número que identifica o produto que foi devolvido
(5) Descrição da devolução efectuada pelo cliente
(6) Data em que foi efectuada a respectiva devolução
(7) Valor a devolver ao cliente

Formatada: Legenda

Tabela 12 - Tabela: Devolução

Operações:

Visualizar () – possibilita visualizar os campos da tabela Devolucao.

Responsabilidades:

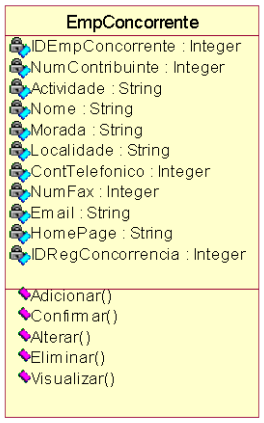
Classe → Devolucao

Responsabilidades → guarda o registo das devoluções efectuadas pelos clientes, permite visualizar.

Classe EmpConcorrente:

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt



Eliminado: FiguraIlustração 5732 -

Figura 2-28 Classe EmpConcorrente

Diagramas de Sequência em que participa:

Registo Concorrência.

Atributos (descrição):

(1) Número sequencial que identifica a Empresa Concorrente
(2) Número de contribuinte da empresa
(3) Tipo de actividade desenvolvida pela empresa
(4) Nome da respectiva empresa
(5) Morada da empresa
(6) Localidade em que se situa a empresa
(7) Contacto telefónico da empresa
(8) Contacto via fax da empresa

Formatada: Rodapé

(9) Contacto via Email da empresa
(10) Página web da empresa
(11) Número que identifica o registo de concorrência em que a empresa está inserida

Tabela 13 - Tabela EmpComcorrente

Operações:

Criar () – possibilita a criação de um novo registo da empresa concorrente na base dados

Gerar – número da EmpConcorrente (anterior +1);

Introduz – NumContribuinte, Actividade, Nome, Morada, Localidade, ContTelefonico, NumFax, Email, HomePage;

Valida – NumCantribuinte=”nove algarismos”;

Alterar () – possibilita alterar os campos da tabela EmpConcorrente

Actualiza: NumContribuinte=”nº novo (com nove algarismos) ”

ContTelefonico=”nº novo (com nove algarismos) ”

NumFax=”nº novo (com nove algarismos) ”

Visualizar () – possibilita visualizar os campos da tabela EmpConcorrente

Eliminar () – possibilita eliminar um registo da tabela EmpConcorrente se não estiver a ser usado na tabela RegistoConcorrencia.

Responsabilidades:

Classe → EmpConcorrente

Formatada: Tipo de letra: 11 pt

Formatada: Legenda

Eliminado: possibilita

Eliminado: algarismos)

Eliminado: algarismos)

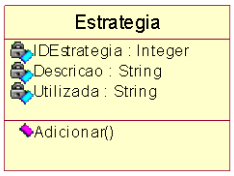
Eliminado: algarismos)

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Responsabilidades → guarda o registo da empresa concorrente, permite criar, alterar, visualizar e eliminar.

Classe Estrategia:



Eliminado: FiguraIlustração 5833 -

Figura 2-29 Classe Estrategia

Diagramas de Sequência em que participa:

Criar Campanha e Registar Concorrência.

Atributos (descrição):

(1) Número sequencial que identifica a estratégia utilizada na campanha
(2) Descrição da estratégia (em que consiste)
(3) A estratégia foi utilizada

Formatada: Legenda

Tabela 2.14 Tabela Estratégia

Eliminado: ¶

Operações:

Adicionar () – possibilita criar uma nova estratégia na base dados

Gerar – Número da Estratégia (anterior + 1);

Introduz – Descrição;

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Alterar – possibilita alterar o campo Descrição;

Confirmar () – possibilita guardar a nova estratégia criada;

Alterar () – possibilita alterar o campo Descrição;

Visualizar () – possibilita visualizar o campo descrição da tabela Estratégia;

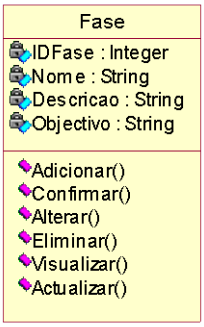
Eliminar () – possibilita eliminar um registo da tabela Estratégia se não estiver a ser usado na tabela Campanha.

Responsabilidades:

Classe → Estratégia

Responsabilidades → guarda todos os registos da estratégia, permite adicionar, confirmar, alterar, visualizar e eliminar.

Classe Fase:



Eliminado: FiguraIlustração 5934 -

Figura 2-30 Classe Fase

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Diagramas de Sequência em que participa:

Criar Campanha, Definir Meio de Comunicação, Lançar (fase) Campanha, Análise Resultados, Analisar final Campanha.

Atributos (descrição):

(1) Número sequencial que identifica a fase
(2) Nome que define a fase
(3) Breve descrição da respectiva fase
(4) Objectivo que se pretendem atingir nesta fase

Eliminado: s

Formatada: Legenda

Tabela 15 - Tabela Fase

Eliminado: ¶

Operações:

Adicionar () – possibilita adicionar uma nova fase na base de dados;

Gerar – número da fase (anterior +1);

Introduz – Nome, descrição, Objectivos.

Confirmar () – possibilita guardar a fase criada.

Alterar () – possibilita alterar alguns campos da tabela Fase.

Eliminar () – possibilita eliminar o registo da fase se não constar nenhum registo na tabela MeioComunicacao, Resposta_Campanha, ProdCampanha, e Cliente.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Visualizar () – possibilita visualizar os campos da tabela Fase, como Nome, Descricao, Objectivos.

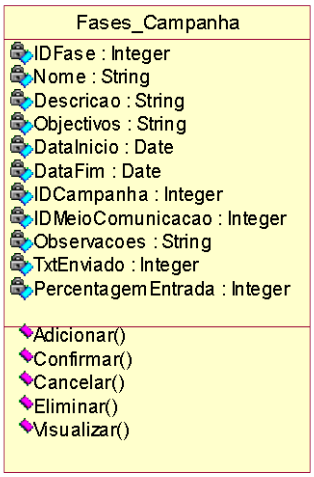
Responsabilidades:

Classe → Fase

Responsabilidades → guarda todos os registos de todas as fases que a Campanha poderá vir a ter, permite adicionar, confirmar, alterar, eliminar e visualizar.

Eliminado: ¶

Classe Fases_Campanha:



Eliminado: FiguraIlustração 6035 -

Figura 2-31 Classe Fases_Campanha

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Diagramas de Sequência em que participa:

Criar Campanha, Lançar (fase) Campanha.

Atributos (descrição):

(1) Identificação da fase
(2) Nome que define a fase
(3) Breve descrição da fase
(4) Objectivo que se pretende atingir nesta fase
(5) Data de lançamento da fase (data de inicio)
(6) Data prevista para o fim da fase
(7) Identificação da campanha
(8) Identificação do meio de comunicação
(9) Observações acerca da respectiva fase
(10) Texto Enviado
(11) Percentagem de entrada

Eliminado: s

Eliminado: em

Formatada: Legenda

Tabela 16 - Tabela: Fases_Campanha.

Operações:

Adicionar () – possibilita criar uma nova fase da campanha na base dados

IDCampanha – Selecciona qual a Campanha;

IDFase – Selecciona a Fase;

PercentagemEntrada – percent1 = (dias_passados/dias_campanha) * 100.

Confirmar () – possibilita a criação de um registo da nova fase da campanha.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Cancelar () – Cancela a criação da fase da campanha que se estava a criar.

Eliminar () - possibilita eliminar um registo da tabela Fases_Campanha se não estiver a ser usado na tabela Campanha.

Visualizar () – possibilita visualizar os campos da tabela Fases_Campanha.

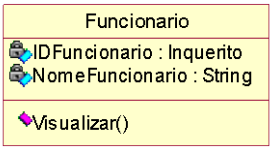
Responsabilidades:

Classe → Fases_Campanha

Responsabilidades → guarda todos os registos das fases da campanha, permite adicionar, confirmar, cancelar, eliminar e visualizar.

Classe Funcionário:

Esta classe (tabela) é apenas de consulta.



Eliminado: FiguraIlustração 6136 -

Figura 2-32 Classe Funcionário

Diagramas de Sequência em que participa:

Registar Oportunidade Venda.

Atributos (descrição):

(1) Número sequencial que identifica o funcionário (responsável)
(2) Nome do funcionário responsável

Formatada: Legenda

Tabela 17 - Tabela Funcionário

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Operações:

Visualizar () – possibilita visualizar o nome do funcionário responsável pela Campanha

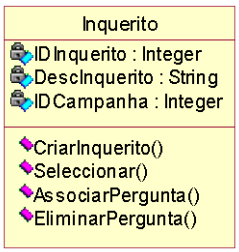
Responsabilidades:

Classe → Funcionário

Eliminado:

Responsabilidades → guarda a informação do funcionário responsável pela Campanha e permite visualizar.

Classe Inquérito:



Eliminado: FiguraIlustração 6237 -

Figura 2-33 Classe Inquérito

Diagramas de Sequência em que participa:

Registar Respostas do Inquérito.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Atributos (descrição):

(1) Número sequencial que identifica o inquérito
(2) Descrição do inquérito
(3) Número que identifica a campanha

Formatada: Legenda

Tabela 18 - Tabela Inquérito

Operações:

CriarInquerito () – possibilita a criação de um novo inquérito.

Gerar- número do Inquérito (anterior +1);

IDCampanha – Selecciona qual a campanha;

Introduz – DescInquerito;

Confirmar () - possibilita guardar o novo inquérito.

Alterar () – possibilita alterar o inquérito, ou seja, o campo DescInquerito

Actualza DecInquerito= “tipo Inquérito”;

Cancelar () – possibilita cancelar o inquérito introduzido

Responsabilidades:

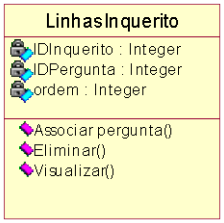
Classe → Inquerito

Responsabilidades → guarda todos os registos do inquérito, permite criar e alterar.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Classe Linhas_Inquerito:



Eliminado: FiguraIlustração 6338 -

Figura 2-34 Classe Linhas_Inquerito

Diagramas de Sequência em que participa:

Registar Respostas do Inquérito.

Atributos (descrição):

(1) Número que identifica o inquérito
(2) Número que identifica a pergunta
(3) Número que identifica a ordem da pergunta

Formatada: Legenda

Tabela 19 - Tabela: Linhas_Inquerito

Operações:

Adicionar () – possibilita criar novas linhas de perguntas para o inquérito

Confirmar () – possibilita guardar as linhas das perguntas do inquérito criadas

Cancelar () – possibilita cancelar uma pergunta quando a estamos a inserir caso desejemos

Eliminar () – possibilita eliminar linhas das perguntas dos inquéritos.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Responsabilidades:

Classe → Linhas_Inquerito

Responsabilidades → guarda a informação das linhas das perguntas do inquérito, permite adicionar, confirmar, cancelar e eliminar.

Classe LinhasResp_Inquerito:

Eliminado: FiguraIlustração 6439 -

Figura 2-35 Classe LinhasResp_Inquerito

Diagramas de Sequência em que participa:

Registar Respostas do Inquérito, Análise Resultados.

Atributos (descrição):

(1) Número que identifica o inquérito

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

(2) Número que identifica a pergunta
(3) Número que identifica o tipo de resposta
(4) Número que identifica o cliente
(5) Data de envio do inquérito
(6) Data da resposta ao inquérito

Formatada: Legenda

Tabela 20 - Tabela LinhasResp_Inquerito

Operações:

Confirmar () – possibilita a criação de um registo de novas linhas de resposta aos inquéritos

Visualizar () – possibilita visualizar (consultar) toda a informação contida nas linhas das respostas ao inquérito

Alterar () – possibilita alterar os campos da tabela LinhasResp_Inquerito

Eliminar () – possibilita eliminar o registo das respostas ao inquerito se não constar nenhum registo nas tabelas Inquérito, Pergunta e TipoResposta.

Responsabilidades:

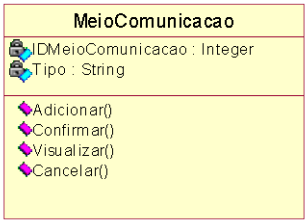
Classe → LinhasResp_Inquerito

Responsabilidades → guarda todos os registos das respostas ao inquerito, permite confirmar (cria automaticamente), visualizar, alterar e eliminar.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Classe MeioComunicacao:



Eliminado: FiguraIlustração 6540 -

Figura 2-36 Classe MeioComunicacao

Diagramas de Sequência em que participa:

Criar Campanha, Definir MeioComunicação, Lançar (fase) Campanha.

Atributos (descrição):

(1) Número sequencial que identifica o meio de comunicação
(2) Identifica o tipo de meio de comunicação

Formatada: Legenda

Tabela 21 - Tabela MeioComunicacao

Operações:

Adicionar () – possibilita a criação do registo de um novo meio de comunicação

Gerar – número do MeioComunicacao (anterior +1);

Insere – Tipo;

Valida – Tipo =”qual o meio de comunicação”;

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Confirmar () – possibilita modificar o campo Tipo dos meios de comunicação

Visualizar () – possibilita visualizar o Tipo dos meios de comunicação

Cancelar () – possibilita cancelar a introdução de um registo na tabela Fase.

Responsabilidades:

Classe → MeioComunicacao

Responsabilidades → guarda todos os meios de comunicação disponíveis, permite adicionar, confirmar, visualizar e permite também cancelar em caso de necessidade.

Classe OportunidadeVenda:

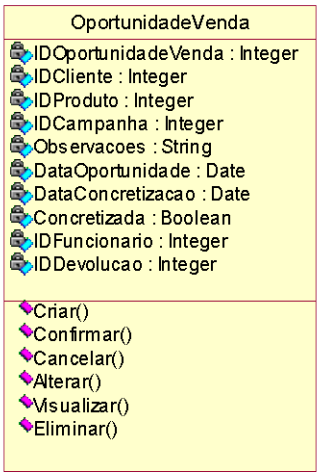


Figura 2-37 Classe OportunidadeVenda

Eliminado: FiguraIlustração 6641-

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Diagramas de Sequência em que participa:

Registar Oportunidade Venda.

Atributos (descrição):

(1) Número sequencial que identifica a oportunidade de venda
(2) Número que identifica o cliente que regista a oportunidade de venda
(3) Número que identifica o produto
(4) Número que identifica a campanha
(5) Observações acerca da oportunidade de venda que surgiu
(6) Data de registo da oportunidade de venda
(7) Data de concretização da oportunidade de venda
(8) A oportunidade foi concretizada
(9) Número que identifica o funcionário que registou e concretizou a oportunidade
(10) Número que identifica a devolução que pode ter dado origem à oportunidade de venda

Formatada: Legenda

Tabela 22 - Tabela OportunidadeVenda

Operações:

Criar () – possibilita criar um registo de uma nova oportunidade de venda na base dados

Gerar – número da Oportunidade de Venda (anterior +1);

IDCliente – Selecciona o Cliente que propõe a oportunidade de venda;

IDProduto – Selecciona o produto;

IDCampanha – Selecciona a Campanha;

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

IDFuncionario – Selecciona o funcionário que registou a oportunidade;

Introduz- Observações, DataOportunidade, DataConcretizacao, concretizada.

Valida: Data >= datadia

DataConcretizada = branco

Eliminado:

Confirmar () – possibilita guardar os dados da nova oportunidade de venda criada

Cancelar () – possibilita cancelar a oportunidade de venda que estamos a criar

Alterar () – possibilita alterar alguns dos campos da tabela oportunidade de venda como os campos Observações, DataConcretizada e DataConcretizacao.

DataConcretizacao=”dia em que se concretizou a venda”;

Concretizada “sim”;

Visualizar () – possibilita visualizar os campos da tabela oportunidade de venda

Eliminar () – possibilita eliminar um registo da tabela Oportunidade de Venda.

Responsabilidades:

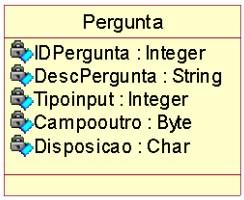
Classe → OportunidadeVenda

Responsabilidades → guarda todos os registos de oportunidade de venda, permite alterar, visualizar e eliminar.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Classe Pergunta:



Eliminado: FiguraIlustração 6742 -

Figura 2-38 Classe Pergunta

Diagramas de Sequência em que participa:

Registar Respostas do Inquérito.

Atributos (descrição):

(1) Número sequencial que identifica a pergunta
(2) Descrição da pergunta
(3) Número que identifica o tipo de input
(4) Campo outro (branco) (sim ou não)
(5) Disposição da pergunta

Eliminado: Tabela 2827 -

Formatada: Legenda

Figura 2-39 Classe Pergunta

Operações:

Adicionar () – possibilita criar um registo de novas Perguntas

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Gerar – número da Pergunta (anterior + 1);

Introduz – DescPergunta;

Confirmar () – possibilita guardar a informação da pergunta criada

Alterar () – possibilita alterar o campo DescPergunta.

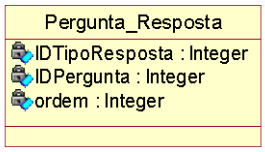
Cancelar () – possibilita cancelar a pergunta se está a inserir.

Responsabilidades:

Classe → Pergunta

Responsabilidades → guarda as perguntas às quais os clientes vão responder no inquérito, permite adicionar, confirmar, alterar e cancelar.

Classe Pergunta_Resposta:



Eliminado: FiguraIlustração 6843 -

Figura 2-40 Classe Pergunta_Resposta

Diagramas de Sequência em que participa:

Registar Respostas do Inquérito.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Atributos (descrição):

(1) Número que identifica o tipo de resposta dado à pergunta
(2) Número que identifica a pergunta
(3) Ordem da resposta à pergunta

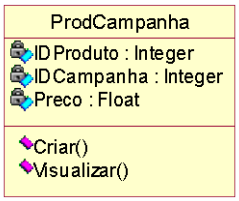
Formatada: Legenda

Tabela 23- Tabela Pergunta_Resposta

Responsabilidades:

Classe → Pergunta_Resposta

Classe ProdCampanha (Produto da Campanha):



Eliminado: FiguraIlustração 6944 -

Figura 2-41 Classe ProdCampanha

Diagramas de Sequência em que participa:

Criar Campanha, Registo Venda.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Atributos (descrição):

(1) Número que identifica o produto
(2) Número que identifica a campanha
(3) Preço do produto na campanha

Formatada: Legenda

Tabela 29 - Tabela ProdCampanha

Operações:

Criar () – possibilita a criação de um novo produto para entrar na Campanha

Gerar – número do Produto da Campanha (anterior+1);

IDCampanha – Selecciona a Campanha;

Introduz – preço;

Visualizar () – permite visualizar os campos da tabela ProdCampanha

Responsabilidades:

Classe → ProdCampanha

Eliminado:

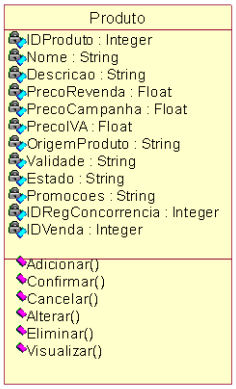
Responsabilidades → guarda registos do produto da campanha e permite criar e visualizar.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Eliminado: ¶

Classe Produto:



Eliminado: FiguraIlustração 7045 -

Figura 2-42 Classe Produto

Diagramas de Sequência em que participa:

Criar Campanha, Registar Oportunidade Venda, Registar Concorrência, Registar Venda.

Atributos (descrição):

(1) Número sequencial que identifica o produto
(2) Descreve o nome do produto
(3) Descrição do produto
(4) Preço de revenda
(5) Preço da Campanha
(6) Valor do IVA (Imposto de valor acrescentado)
(7) Origem do produto
(8) Validade do produto

Formatada: Rodapé

(9) Estado em que se encontra o produto
(10) Promoções
(11) Número que identifica o registo de concorrência que contém o produto
(12) Número que identifica o registo da venda do produto

Tabela 24- Tabela Produto

Operações:

Visualizar () – possibilita visualizar (consultar) todas as informações do produto

Responsabilidades:

Classe → Produto

Responsabilidades → esta classe vai permitir guardar todos os registos dos produtos e permite visualizar, modificar e eliminar os seus campos

Classe RegistoConcorrencia:

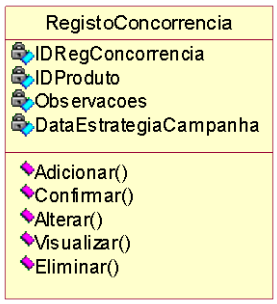


Figura 2-43 Classe RegistoConcorrencia

Formatada: Tipo de letra: 11 pt

Diagramas de Sequência em que participa:

Registo Concorrência.

Atributos (descrição):

(1) Número sequencial que identifica o Registo de Concorrência
(2) Número que identifica o produto
(3) Observações acerca do registo de concorrência efectuado
(4) Data de utilização dessa estratégia usada pela concorrência na nossa campanha

Formatada: Legenda

Tabela 25- Tabela RegistoConcorrencia

Operações:

Adicionar () – possibilita a criação de um novo registo da concorrência na base dados

Gerar – número do registo Concorrência (anterior);

IDProduto – Selecciona o produto;

IDEmpConcorrente – Selecciona a empresa concorrente;

IDEstrategia – Selecciona a estratégia;

Introduz – Observacoes e DataEstrategiaCampanha.

Confirmar () – possibilita guardar os dados do registo da concorrência criados

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Alterar () – possibilita alterar dados do registo da concorrência

Visualizar () – possibilita visualizar os campos da tabela RegistoConcorrencia

Eliminar () – elimina o registo da tabela RegistoConcorrencia.

Responsabilidades:

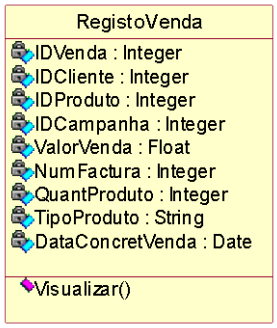
Classe → RegistoConcorrencia

Responsabilidades → guarda registo da concorrência, permite adicionar, confirmar, alterar, visualizar e eliminar.

Eliminado: - - - - - - - - - - - - -

Classe RegistoVenda:

Esta classe (tabela) é apenas de consulta.



Eliminado: FiguraIlustração 7247 -

Figura 2-44 Classe RegistoVenda

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Diagramas de Sequência em que participa:

Registar Venda, Análise Resultados.

Atributos (descrição):

(1) Número sequencial que identifica o Registo de Venda
(2) Número que identifica o cliente ao qual corresponde o registo de venda
(3) Número que identifica o produto que foi vendido
(4) Número que identifica a campanha à qual corresponde o registo de venda
(5) Valor monetário da respectiva venda
(6) Número da factura correspondente ao registo da venda efectuada
(7) Quantidade de produto vendido ao cliente
(8) Descrição do tipo de produto que foi vendido
(9) Data de concretização da venda

Formatada: Legenda

Tabela 26 - Tabela: RegistoVenda

Operações:

Visualizar () – possibilita visualizar os campos da tabela RegistoVenda

Responsabilidades:

Classe → RegistoVenda

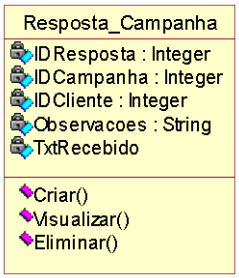
Responsabilidades → guarda o registo das vendas e permite visualizá-las.

Eliminado:

Classe Resposta_Campanha:

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt



Eliminado: FiguraIlustração 7348 -

Figura 2-45 Classe Resposta_Campanha

Diagramas de Sequência em que participa:

Registar Resposta da Campanha, Análise Resultados.

Atributos (descrição):

(1) Número sequencial que identifica a resposta à campanha
(2) Número que identifica a campanha à qual foi dada a resposta
(3) Número que identifica o cliente que deu a resposta
(4) Observações acerca da resposta
(5) O texto foi recebido (Sim/Não)
(6) Número que identifica o tipo de resposta dada

Formatada: Legenda

Tabela 27 - Tabela Resposta_Campanha

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Operações:

Criar () – cria um registo de novas respostas dos clientes durante a campanha

Gerar – número da Resposta da campanha (anterior +1);

IDCampanha – Selecciona qual a Campanha;

IDCliente – Selecciona qual o Cliente;

IDTipoResposta – Selecciona qual o tipo de resposta dado pelo cliente;

Introduz – Observações;

Valida – TxtRecebido=”sim”

Visualizar () – possibilita (visualizar) consultar o registo (campo) das respostas dos clientes

Eliminar () – Só posso eliminar um registo de uma determinada Resposta se não estiver a ser usado nas tabelas Fase, Campanha, TipoResposta e AnaliseResultado.

Responsabilidades:

Classe → Resposta

Responsabilidades → guarda as respostas que os clientes vão dando ao longo da campanha, permite criar, visualizar e eliminar.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Classe Segmento:

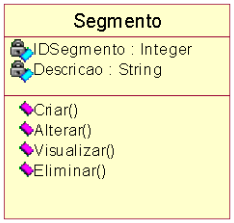


Figura 2-46- Classe Segmento

Eliminado: Ilustração

Eliminado: 7449

Diagramas de Sequência em que participa:

Criar Campanha, Definir Segmento, Lançar (fase) Campanha.

Atributos (descrição):

(1) Número sequencial que identifica o segmento
(2) Descreve o segmento

Tabela 28 Tabela Segmento

Formatada: Legenda

Eliminado: 2.

Operações:

Criar () – possibilita criar um novo registo de um novo segmento na base dados

Gerar – número do Segmento (anterior+1);

Introduz – Descrição;

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Alterar () – possibilita alterar o campo Descrição da tabela Segmento

Actualiza: Descrição =”define segmento”;

Visualizar () – possibilita visualizar todos os campos da tabela Segmento

Eliminar () – só poderá eliminar o registo se nas tabelas SegmentoLinha e na tabela Campanha não constar esse registo.

Responsabilidades:

Classe → Segmento

Responsabilidades → guarda registo do segmento, permite criar, alterar, visualizar e eliminar.

Classe SegmentoCampanha:

Esta Classe (tabela) foi criada para o caso de se ter um segmento, por exemplo em 2004 o segmento incluía 35 alunos e outro segmento criado em 2005 incluía 50 alunos, o segmento criado em 2005 pode conter clientes que pertençam ao segmento de 2004 e vice-versa, assim criámos o segmento campanha para saber se os clientes são os mesmos ou são novos.

Esta classe guarda os clientes pertencentes ao segmento da Campanha.

Formatada: Rodapé

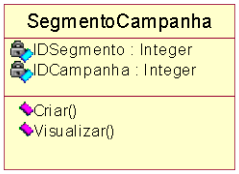


Figura 2-47 - Classe SegmentoCampanha

Atributos (descrição):

(1) Número que identifica o cliente pertencente ao segmento da campanha
(2) Número que identifica a campanha à qual pertence o segmento

Tabela 29 - Tabela SegmentoCampanha

Operações:

Criar () – possibilita criar um registo do segmento da campanha

Visualizar () – possibilita visualizar o SegmentoCampanha

Responsabilidades:

Classe → SegmentoCampanha

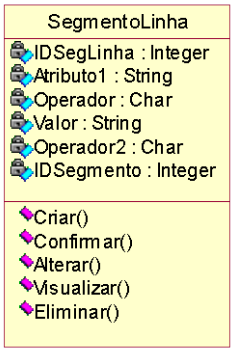
Responsabilidades → guarda a informação dos segmentos da campanha, permite criar e visualizar

Formatada: Tipo de letra: 11 pt
Eliminado: Ilustração
Eliminado: 7550
Formatada: Legenda
Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Eliminado: ¶

Classe SegmentoLinha:



Eliminado: Ilustração

Figura 2-48 - Classe SegmentoLinha

Eliminado: 7651

Diagramas de Sequência em que participa:

Definir Segmento.

Atributos (descrição):

(1) Número sequencial que identifica a linha do segmento
(2) Atributos ou campos existentes na tabela (classe) clientes (ex: nome, morada, email, entre outros)
(3) Operadores, como "<, >, =, >=, <=", entre outros
(4) Valor para o qual se pretende filtrar
(5) Operadores lógicos a seguir à condição, como "and, or, , not"
(6) Número que identifica o segmento a que a linha do segmento pertence

Formatada: Legenda

Tabela 30 - Tabela SegmentoLinha

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Operações:

Criar () – possibilita a criação de um novo registo de segmento na base de dados

Gerar – número do Segmento Linha (anterior+1);

IDSegmento – Selecciona o segmento;

Introduz – Atributo =”campo retirado da tabela Cliente, que nos diz a que se dirige a campanha”;

Operador1=”operadores”;

Valor =”campo introduzido”;

Operador2=”valores lógicos”;

Confirmar () – possibilita guardar os dados do novo segmento linha.

Alterar () – possibilita alterar alguns campos da Segmentação como por exemplo Atributo1, Operador, Valor e o Operador2

Actualizar (): Atributo =”campo retirado da tabela Cliente, que nos diz a que se dirige a campanha”;

Operador1=”operadores”;

Valor =”campo introduzido”;

Operador2=”valores lógicos”;

Visualizar () – possibilita visualizar os campos referentes aos atributos e aos operadores

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Eliminar () – possibilita eliminar um registo da tabela SegmentoLinha se não estiver a ser usado nas tabelas campanha, Segmento e Clientes_Segmento.

Eliminado: .

Responsabilidades:

Classe → Segmento Linha

Responsabilidades → guarda as condições do Segmento, permite criar, alterar, visualizar e eliminar.

Classe SugReclamação:

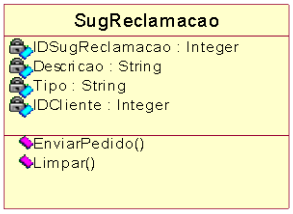


Figura 2-49 - Classe SugReclamacao

Eliminado: Ilustração

Eliminado: 7752

(1) Número sequencial que identifica a sugestão ou reclamação
(2) Descrição da sugestão ou reclamação
(3) Tipo: se é sugestão ou reclamação
(4) Número que identifica o Cliente que faz a sugestão ou reclamação

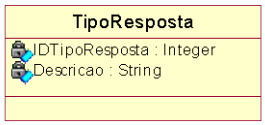
Formatada: Legenda

Tabela 31 - Tabela SugReclamacao

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Classe TipoResposta:



Eliminado: Ilustração

Eliminado: 7853

Figura 2-50 - Classe TipoResposta

Diagramas de Sequência em que participa:

Registar Resposta da Campanha, Registar Respostas dos Inquéritos, Análise Resultados

Atributos (descrição):

(1) Número sequencial que identifica o tipo de resposta dada à pergunta
(2) Descrição do tipo de resposta

Formatada: Legenda

Tabela 37 - Tabela TipoResposta

Operações:

Criar () – criar um registo do Tipo de Resposta dado pelos clientes

Gerar – número do TipoResposta (anterior +1);

Introduz – TipoResposta;

Eliminar () – possibilita eliminar um registo da tabela TipoResposta se não estiver a ser usado nas tabelas Resposta_Campanha e Pergunta.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Responsabilidades:

Classe → TipoResposta

Responsabilidades → Guarda registos referentes ao Tipo de Resposta e permite eliminar.

Formatada: Rodapé

3. Implementação - Projecto EstgCRM

3.1. Criação de Base de Dados

Depois de realizada a análise e concepção do Sistema CRM que se pretendia, criou-se uma base de dados em *SQL Server*.

Depois de criada a base de dados com o SQL *Enterprise Manager* pode-se gerir os dados da base de dados, criando, alterando ou excluindo os dados com o *Enterprise Manager* ou com comandos SQL.

▪ Para se criar uma base de dados no *Enterprise Manager* (Figura 3.1), faz-se click em "*Databases*"com o botão direito e seleccionamos "*New Databases…*"

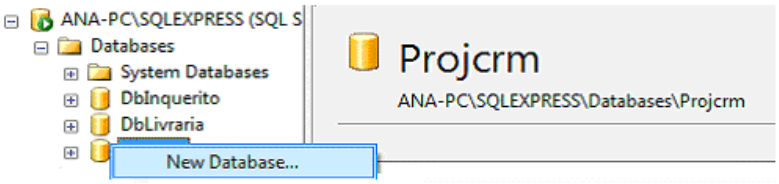


Figura 3-1 *New Database*

Formatada: Tipo de letra: 11 pt
Formatada: Tipo de letra: Itálico
Formatada: Tipo de letra: Itálico
Formatada: Tipo de letra: Itálico
Formatada: Tipo de letra: Itálico
Eliminado: FiguraIlustração 7954
Formatada: Tipo de letra: Itálico
Formatada: Tipo de letra: Itálico
Formatada: Tipo de letra: Itálico
Eliminado: FiguraIlustração 8054 -
Formatada: Tipo de letra: Não Itálico
Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

▪ Fazer o *restore* da base de dados "*Projcrm*" a partir do *backup* (Figura 3.2) e do script das tabelas:

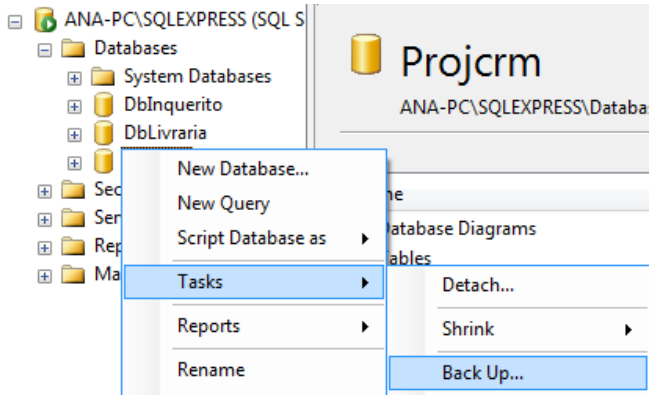


Figura 3-2 - *Generate SQL Script e Backup Database*

Eliminado: Ilustração

Eliminado: 8155

Formatada: Tipo de letra: Itálico

Documentar os passos da criação da base de dados *SQL Server* pode ser útil por diversas razões, sendo a principal e mais motivadora o facto de termos um *backup* do trabalho realizado. Não vai prevenir as perdas de dados, mas vai salvar o modelo da base de dados.

Formatada: Tipo de letra: Itálico

Eliminado:

Formatada: Tipo de letra: Itálico

O *SQL Server* tem um gerador de *script* que facilita a documentação e se necessário a reconstrução de uma base de dados.

Formatada: Tipo de letra: Itálico

Eliminado:

Formatada: Tipo de letra: Itálico

Formatada: Tipo de letra: Itálico

Para criar um *script* (Figura 3.3) a partir do *SQL Server Management* expande-se o servidor "*Ana-PC\SQLEXPRESS*", expande-se a *Databases* (base dados), faz-se clique na base de dados e de seguida em "projcrm".

Formatada: Tipo de letra: Itálico

Eliminado: Ilustração 8255

Formatada: Tipo de letra: Itálico

Formatada: Tipo de letra: Itálico

Formatada: Tipo de letra: Itálico

Formatada: Rodapé

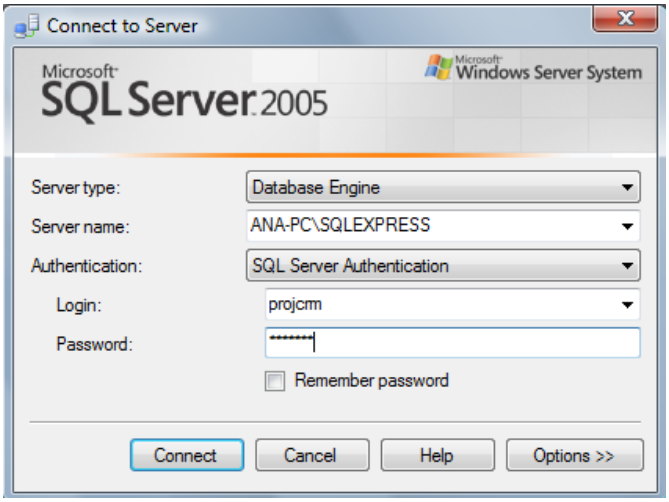


Figura 3-3 Ligação ao servidor

▪ Para restaurar a base de dados expande-se "projcrm", selecciona-se "*Tasks*" e "*Restore*" e de seguida selecciona-se "*Database...*" (Figura 3.4 e 3.5).

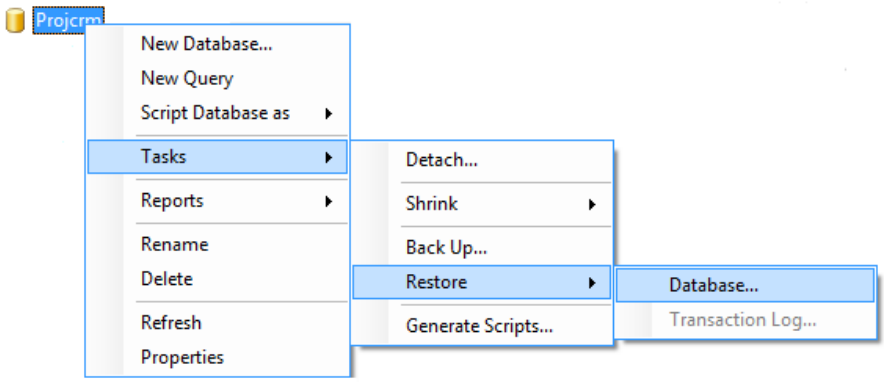


Figura 3-4 Fazer "*Restore*" da Base de Dados – Projcrm

Formatada: Tipo de letra: 11 pt
Eliminado: FiguraIlustração 8356 -
Formatada: Tipo de letra: Itálico
Formatada: Tipo de letra: Itálico
Formatada: Tipo de letra: Itálico
Eliminado:
Eliminado: Ilustração
Eliminado: 8456
Eliminado:
Eliminado: FiguraIlustração 8557 -
Formatada: Tipo de letra: Itálico
Formatada: Rodapé

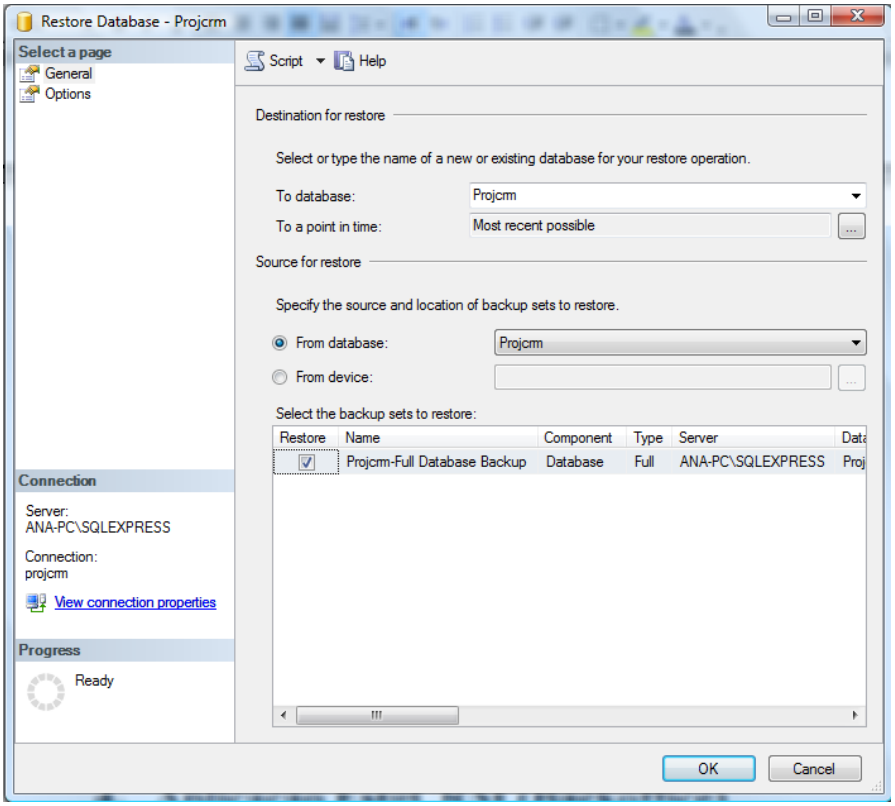


Figura 3-5 *Restore DataBase* – Projcrm

3.2. Tabelas

Exemplo da criação de uma Tabela no *SQL Server*:

```
CREATE TABLE [dbo].[CAMPANHA](

    [IDCAMPANHA] [numeric](9, 0) IDENTITY(1,1) NOT NULL,
```


Formatada: Tipo de letra: 11 pt

Eliminado: Ilustração

Eliminado: 8658

Eliminado: -

Formatada: Tipo de letra: Itálico

Formatada: Tipo de letra: Itálico

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

```
	[IDSEGMENTO] [numeric](9, 0) NULL,
	[IDESTRATEGIA] [numeric](9, 0) NULL,
	[ESTADO] [char](10) NULL,
	[NOMECAMPANHA] [varchar](50) NULL,
	[DESCCAMPANHA] [varchar](90) NULL,
	[OBJECTIVOS] [varchar](50) NULL,
	[DATAINICIO] [datetime] NULL,
	[DATAFIMPREVISTA] [datetime] NULL,
	[DATAFIMEFECTIVA] [datetime] NULL,
	[CUSTOSPREVISTOS] [float] NULL,
	[CUSTOCAMPANHA] [float] NULL,
	[DESTTIPCLIENTE] [varchar](50) NULL,
	[IDPRODCAMPANHA] [numeric](9, 0) NULL,
	[IDFUNCIONARIO] [numeric](9, 0) NOT NULL,
	[FASE] [numeric](9, 0) NULL,
 CONSTRAINT [PK_CAMPANHA] PRIMARY KEY CLUSTERED
(
	[IDCAMPANHA] ASC
```

Formatada: Rodapé

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

Exemplo da criação de uma tabela em modo visual (gráfico):

● Na base de dados "Projcrm", seleccionar "*Tables*" (Figura 3.6), escolher a opção "New Table…" (Figura 3.7). Aparece o gráfico da tabela onde se pode criar a nova tabela (Figura 3.8). Na Figura 3.9, encontra-se uma tabela já criada com o nome da coluna, o tipo de dados e o respectivo tamanho em bytes.

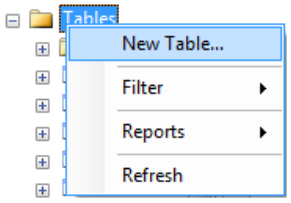


Figura 3-6 Criar uma nova tabela

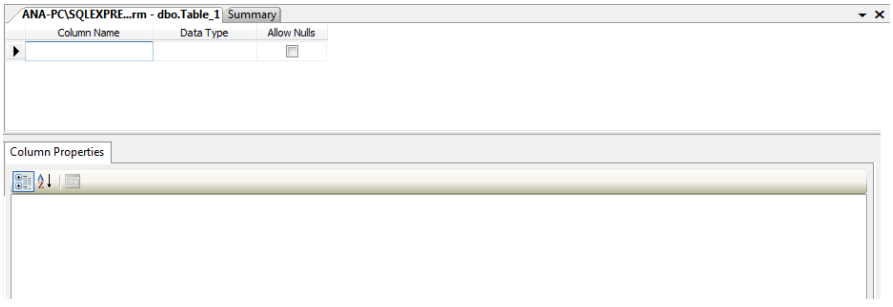
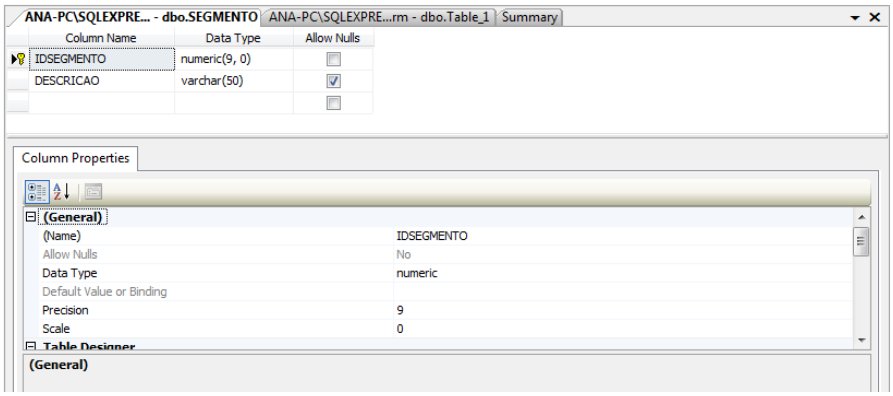


Figura 3-7 - Tabela onde se pode criar uma nova tabela na base dados "projcrm"

Formatada: Tipo de letra: 11 pt
Eliminado: Ilustração
Eliminado: 8759
Formatada: Tipo de letra: Itálico
Eliminado: Ilustração
Eliminado: 8859
Eliminado: Ilustração
Eliminado: 8960
Eliminado: Ilustração
Eliminado: 9061
Eliminado: FiguraIlustração 9159 -
Eliminado: ¶
Eliminado: Ilustração
Eliminado: 9260
Formatada: Rodapé

Formatada: Tipo de letra: 11 pt



Eliminado: Ilustração

Figura 3-8 - Exemplo de uma tabela já criada, a tabela “Segmento”.

Eliminado: 9361

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

3.3. Relação entre as Tabelas

No diagrama seguinte da Figura 3.9, encontram-se as relações entre as tabelas no *SQL Server*. Este diagrama da Base de Dados do sistema EstgCRM designa-se por "dbo.RELACAO".

Eliminado: D

Eliminado: Ilustração

Eliminado: 9462

Formatada: Tipo de letra: Itálico

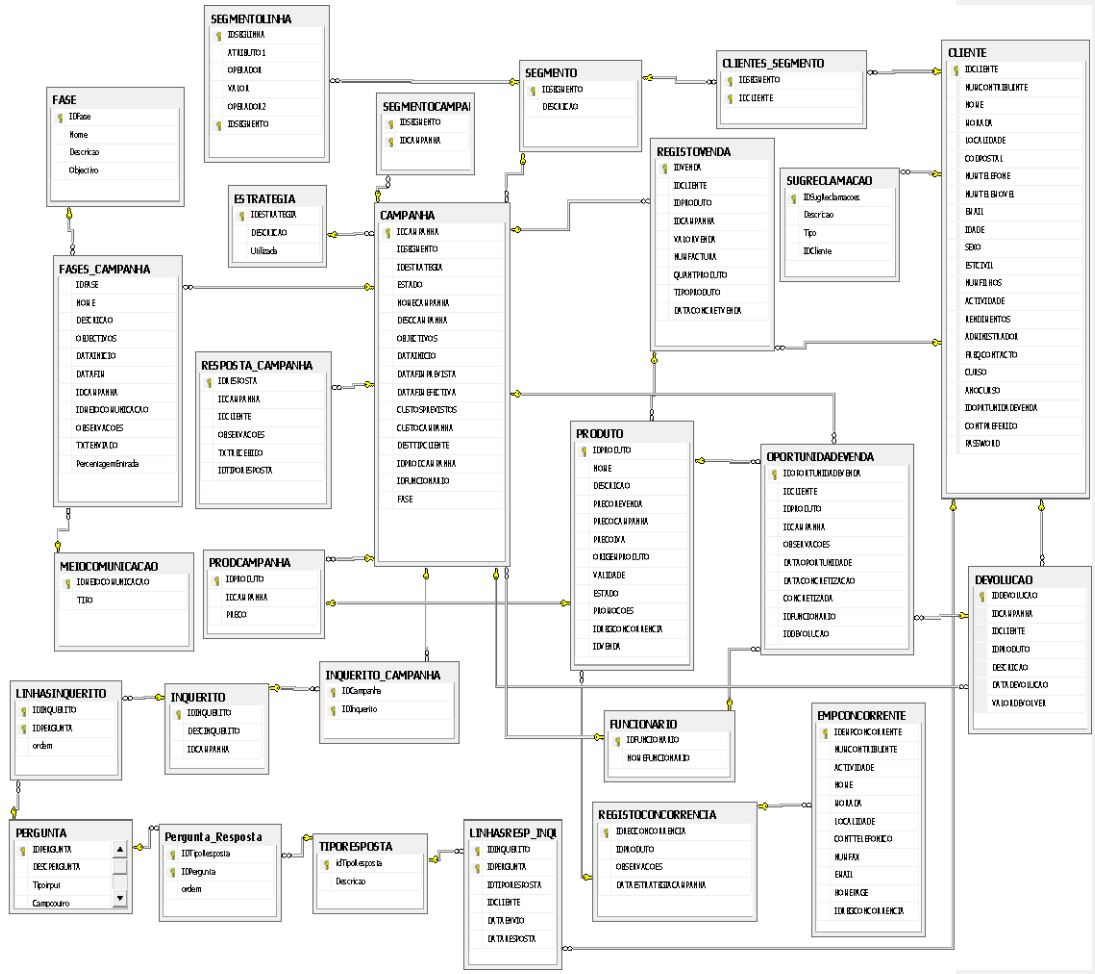


Figura 3-9 RELAÇÃO (dbo.RELACAO).

Eliminado: FiguraIlustração 9562 -

Eliminado: ¶

Formatada: Rodapé

4. Aplicação *EstgCRM*

Neste capítulo faz-se a descrição da aplicação web ***"EstgCRM"***, ou seja, do seu funcionamento. Vamos ter dois tipos de utilizadores, o cliente e o administrador.

Quando um dado utilizador (cliente) aceder ao site ***"EstgCRM"***, este será direccionado para a *home page* normal, se for cliente e para a *home page* de administração, se for administrador.

4.1. Cliente

O Cliente vai fazer login (Figura 4.1), introduz o seu nome de utilizador (e-mail) e a sua *password*.

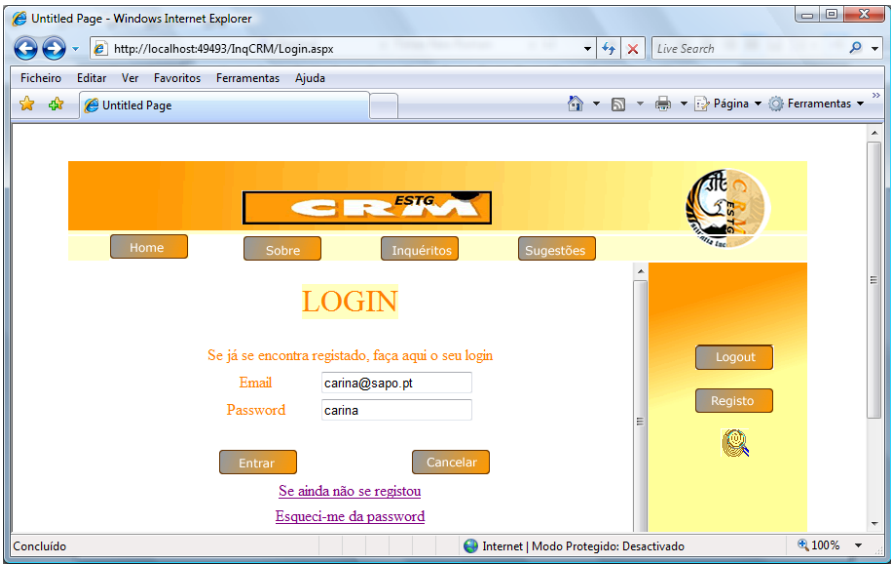


Figura 4-1 - Página do Login

Formatada: Tipo de letra: 11 pt
Eliminado:
Formatada: Tipo de letra: Itálico
Formatada: Tipo de letra: Itálico
Eliminado: Ilustração 63
Eliminado:
Formatada: Tipo de letra: Itálico
Eliminado: Ilustração
Eliminado: 9663
Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Eliminado: Ilustração

Eliminado: 9764

Se o Cliente ainda não se encontra registado, vai registar-se na página de registo (Figura 4.2).

Eliminado: FiguraIlustração 9864 -

Figura 4-2 Página de Registo

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Após fazer o login o cliente entra na aplicação do cliente (Figura 4.3 e Figura 4.4). Nesta aplicação o cliente pode visualizar informação obre quem fez o site (Figura 4.5), pode responder a inquéritos (Figura 4.6) e registar Sugestões ou Reclamações (Figura 4.7).

Eliminado: Ilustração 9965

Eliminado: Ilustração 66

Eliminado: Ilustração

Eliminado: 10067

Eliminado: Ilustração

Eliminado: 10168

Eliminado: Ilustração 10269

Eliminado: FiguraIlustração 10365 -

Figura 4-3 *Home* Cliente.

Formatada: Tipo de letra: Itálico

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Figura 4-4 - Home Cliente.

Eliminado: Ilustração

Eliminado: 10466

Nesta página (Figura 4.5) podemos obter informação sobre o autor da aplicação.

Eliminado: Ilustração

Eliminado: 10567

Figura 4-5 Página Sobre.

Eliminado: FiguraIlustração 10667 -

Formatada: Rodapé

Nesta página (Figura 4.6) seguinte o cliente vai poder visualizar e responder aos inquéritos que lhe são enviados.

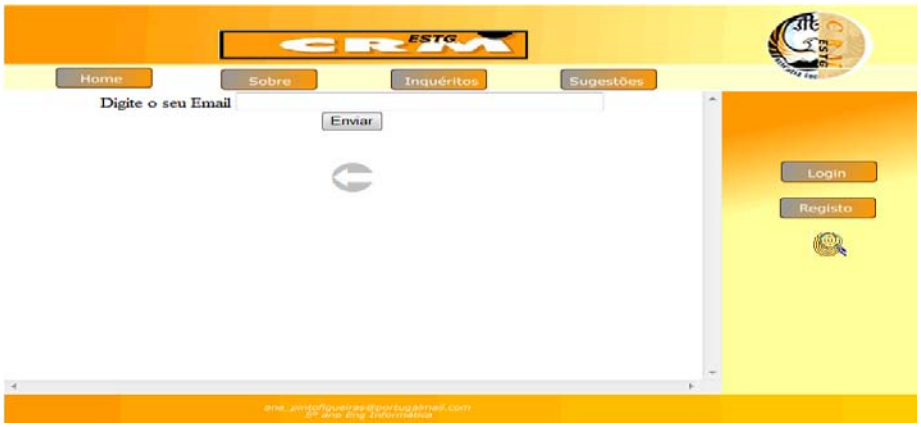


Figura 4-6 Página Inquérito.

Na página sugestões (Figura 4.7), o cliente pode fazer sugestões ou reclamações relativamente a uma dada campanha.

Figura 4-7 Página Sugestões.

Formatada: Tipo de letra: 11 pt
Eliminado: Ilustração
Eliminado: 10768
Eliminado: FiguraIlustração 10868 -
Eliminado: Ilustração
Eliminado: 10969
Eliminado: FiguraIlustração 11069 -
Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

4.2. Administração

O administrador vai fazer login (Figura 4.8), introduz o seu nome de utilizador (e-mail) e a sua *password*.

Eliminado: Ilustração 70

Formatada: Tipo de letra: Itálico

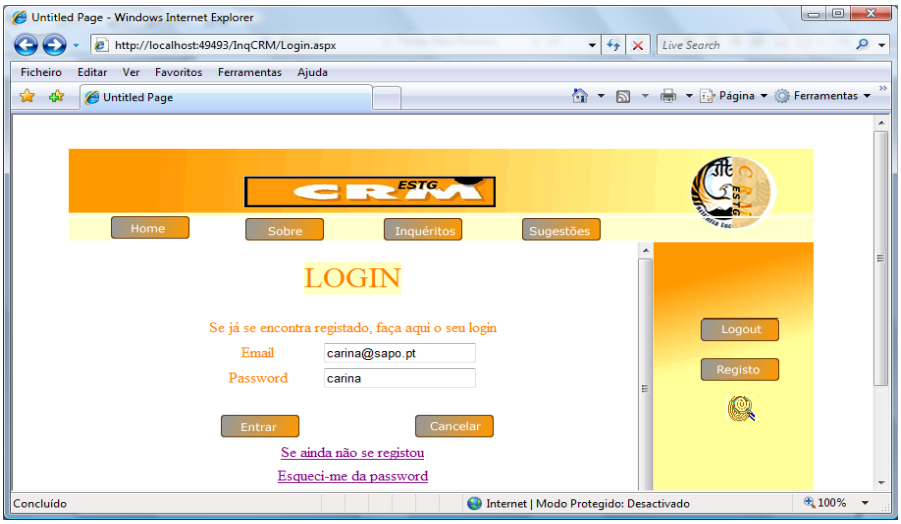


Figura 4-8 Página do Login.

Eliminado: FiguraIlustração 11170 -

Eliminado: ¶

Formatada: Rodapé

Após o login o Administrador entra na sua *Home Page* (Figura 4.9).

Figura 4-9 Página Home Administrador.

Formatada: Tipo de letra: 11 pt

Eliminado: ¶

Eliminado: ¶

Formatada: Tipo de letra: Itálico

Eliminado: Ilustração

Eliminado: 11271

Eliminado: FiguraIlustração 11371 -

Formatada: Rodapé

Nesta aplicação (Figura 4.10), o Administrador pode consultar a página de ajuda à segmentação. Pretende-se com esta página ajudar o administrador a ter um melhor entendimento do funcionamento da segmentação.

Figura 4-10 Página de Ajuda – Segmentação

Formatada: Tipo de letra: 11 pt
Eliminado: ¶ ¶ ¶
Eliminado: Ilustração
Eliminado: 11472
Eliminado: FiguraIlustração 11572 - Página de Ajuda – Segmentação¶
Formatada: Rodapé

Na página Criar Inquérito (Figura 4.11), o administrador vai criar um inquérito, pode escolher o inquérito, pode associar as perguntas que pretende que o inquérito tenha e associar ou não campanhas a esses inquéritos.

Figura 4-11 Página Criar Inquérito

Na página Perguntas (Figura 4.12), o administrador pode adicionar perguntas, adicionar tipo de resposta e associar uma pergunta a uma resposta.

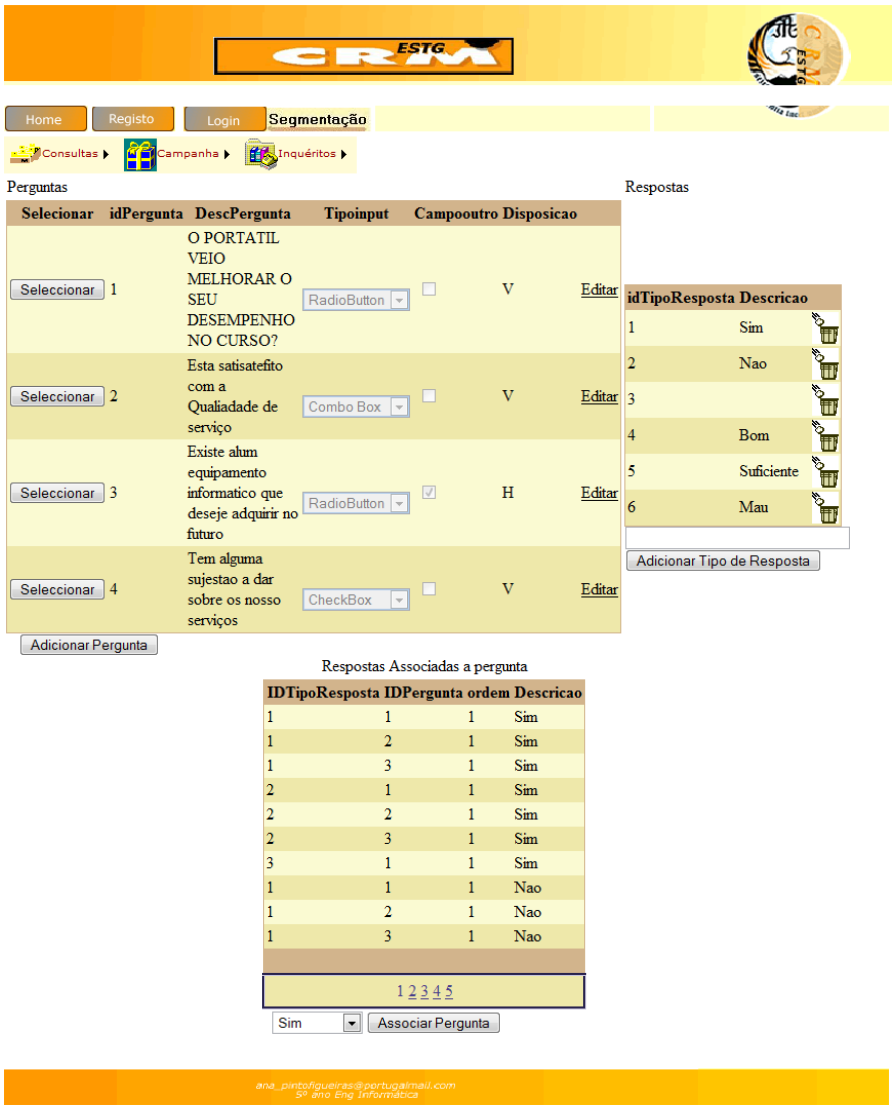


Figura 4-12 Página Perguntas

Formatada: Tipo de letra: 11 pt
Eliminado: Ilustração
Eliminado: 11874
Eliminado: FiguraIlustração 11974 -
Formatada: Rodapé

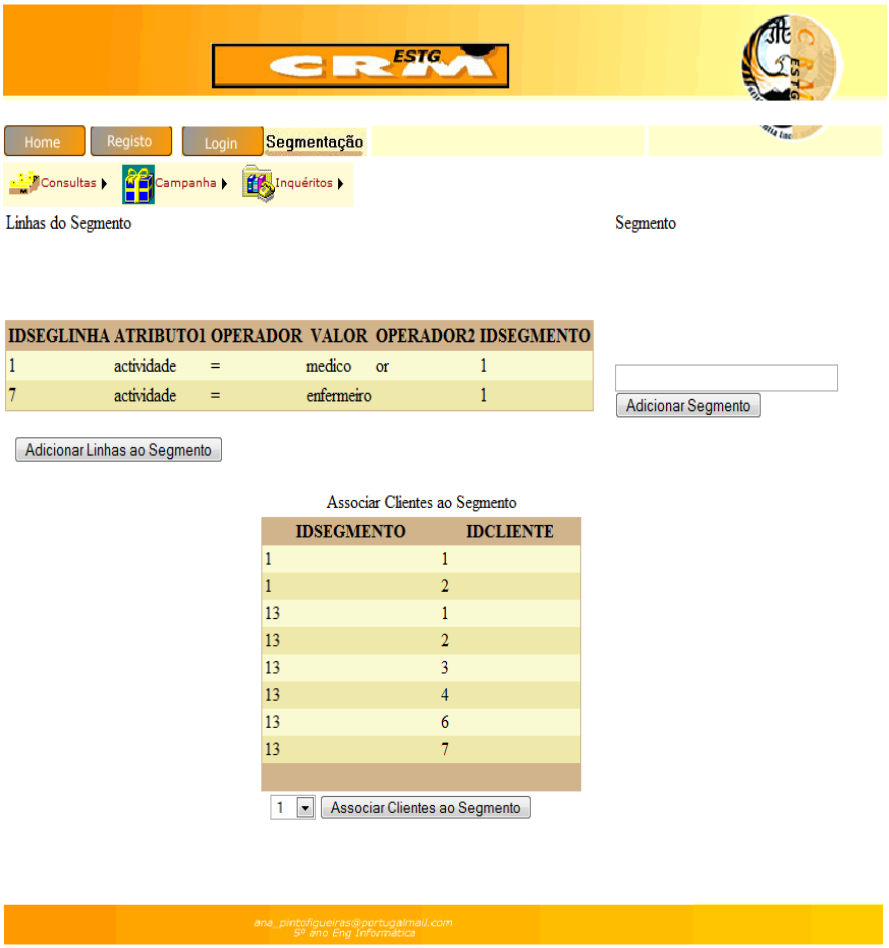
Formatada: Tipo de letra: 11 pt

Na página Segmento (Figura 4.13), o administrador vai inserir um segmento, as linhas do respectivo segmento e ainda vai associar clientes a um dado segmento.

Eliminado: ¶

Eliminado: Ilustração

Eliminado: 12075



Eliminado: FiguraIlustração 12175 -

Figura 4-13 Página Segmentação

Formatada: Rodapé

Na página Criar Campanha (Figura 4.14), o administrador pode criar uma nova campanha.

Figura 4-14 Página Criar Campanha

Formatada: Tipo de letra: 11 pt

Eliminado: ¶

Eliminado: Ilustração

Eliminado: 12276

Eliminado: FiguraIlustração 12376 -

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Na página Criar Fases (Figura 4.15), onde o administrador pode criar fases eu vão ser utilizadas posteriormente na campanha.

Eliminado: Ilustração

Eliminado: 12477

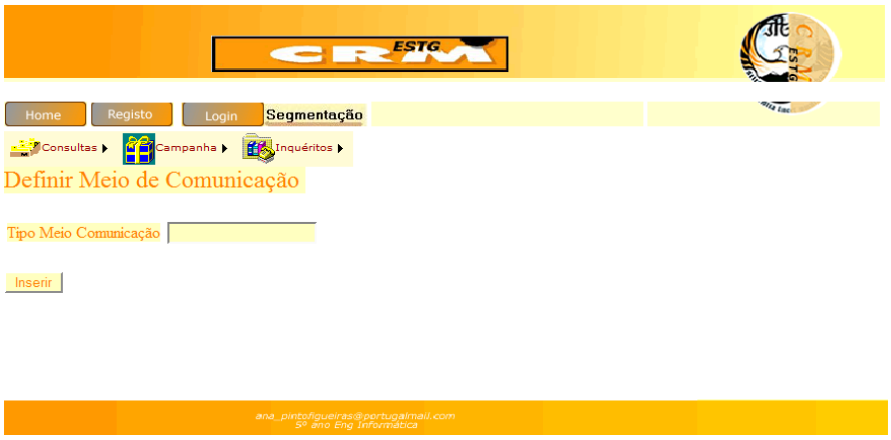
Eliminado: FiguraIlustração 12577 -

Figura 4-15 Página Criar Fases

Na página Definir Meio Comunicação (Figura 4.16), o administrador pode inserir meios de comunicação.

Eliminado: Ilustração

Eliminado: 12678



Eliminado: FiguraIlustração 12778 -

Figura 4-16 Página Definir Meio Comunicação

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Na página Empresa Concorrente (Figura 4.17), o administrador pode visualizar ou consultar os dados das empresas concorrentes.

Eliminado: Ilustração

Eliminado: 12879

Eliminado: FiguraIlustração 12979 -

Figura 4-17 Página Empresa Concorrente.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Na página Estratégia (Figura 4.18), o administrador vai introduzir novas estratégias e seleccionar se são utilizadas ou não.

Eliminado: ¶

Eliminado: Ilustração

Eliminado: 13080

Figura 4-18 Página Estratégia.

Eliminado: FiguraIlustração 13180 -

Na página Produtos (Figura 4.19), o administrador vai inserir os produtos da campanha.

Eliminado: Ilustração

Eliminado: 13281

Eliminado: .

Figura 4-19 Página Produtos

Eliminado: FiguraIlustração 13381 -

Formatada: Rodapé

Na página RegConcorrencia (Figura 4.20), o administrador vai seleccionar as estratégias, as data das estratégias utilizadas na campanha e o produto, isto permite-lhe utilizar as estratégias que na concorrência foram bem sucedidas.

Figura 4-20 Página RegConcorrencia

Formatada: Tipo de letra: 11 pt
Eliminado: Ilustração
Eliminado: 13482
Eliminado: FiguraIlustração 13582 -
Eliminado: ¶
Formatada: Rodapé

O Administrador pode ainda fazer várias consultas:

Na página Consulta Sugestões (Figura 4.21), o administrador pode consultar todas as sugestões e reclamações efectuadas pelos Clientes.

Figura 4-21 Página Consulta Sugestões

Na página Consulta RegConcorrencia (Figura 4.22), o administrador pode consultar os registos de concorrência.

Formatada: Tipo de letra: 11 pt
Eliminado: Ilustração
Eliminado: 13683
Eliminado: FiguraIlustração 13783 -
Eliminado: Ilustração
Eliminado: 13884
Formatada: Rodapé

Figura 4-22 Página Consulta RegConcorrencia

Na página Registo de Venda (Figura 4.23), o administrador pode consultar o registo das vendas efectuadas.

Figura 4-23 Página Registo de Venda

Formatada: Tipo de letra: 11 pt
Eliminado: FiguraIlustração 13984 -
Eliminado: Ilustração
Eliminado: 14085
Eliminado: FiguraIlustração 14185 -
Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Na página Consulta Campanhas (Figura 4.24), o administrador pode visualizar uma lista das Campanhas criadas.

Eliminado: Ilustração

Eliminado: 14286

Eliminado: FiguraIlustração 14386 -

Figura 4-24 Página Consulta Campanhas

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Na página Consulta clientes Registados (Figura 4.25), o administrador pode consultar uma lista de todos os clientes que se encontram registados.

Eliminado: ¶
¶
¶

Eliminado: Ilustração

Eliminado: 14487

Eliminado: FiguraIlustração 14587 -

Figura 4-25 Página Consulta clientes Registados

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Eliminado: <#>¶
<#>¶
<#>¶

5. Comparação com outras aplicações

Foram analisadas as seguintes aplicações (ver anexos):

1- A aplicação EstgCRM elaborada para o projecto proposto.

2- *Microsoft Business Solution – Navision* da empresa distribuidora CRONUSPortugual SA, *Customer Relationship Manager* da empresa distribuidora LS – CRM, CRMADAR da empresa distribuidora MADAR, Gestão de Clientes – CRM da empresa distribuidora Promosoft.

Formatada: Tipo de letra: Itálico

Formatada: Tipo de letra: Itálico

Formatada: Tipo de letra: Itálico

3- Microsoft Dynamics CRM Verson 3.0.

Desta análise concluiu-se que estas aplicações embora tratem de CRM, têm algumas diferenças entre elas. O que se pretende é dar uma ideia geral acerca do seu funcionamento e das diferenças entre elas.

Eliminado: no quadro seguinte (Tabela 3837)

Comentário [C2]: A tabela deve ser resumida. Tem muito texto e nem dá para perceber

1-Aplicação EstgCRM:

Eliminado: ¶

- O produto pode ser consultado, inserido, alterado e eliminado. Informação geral do produto.

Formatada: Tipo de letra:

- O funcionário (número e nome) é utilizado para se consultar qual o responsável por uma dada campanha.

- Implementam-se de inquéritos.

- Podem criar-se segmentos; estes utilizam-se na criação da campanha, ou seja, para se criar uma campanha tem de existir um segmento. A segmentação é mais clara e fácil da fazer nesta aplicação.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

- O Registo das Vendas contém a informação de todas as vendas efectuadas, informação, essa que se pode visualizar.

- Nas Devoluções pode visualizar-se uma lista das devoluções de produtos efectuadas pelos clientes. Nas devoluções vai-se consultar o nº da devolução, a campanha, o cliente, produto, descrição do que se pretende devolver, a data de devolução e o valor a devolver.

- O nº do segmento que se está a usar é criado automaticamente. Insere-se a sua descrição e as linhas do segmento. Nas linhas do segmento vai-se inserir o nº de linhas, os atributos que se vai retirar da tabela clientes e o valor que vamos inserir, serve para definir o tipo de cliente a quem se destina a campanha. O operador1 para ligar o atributo e o valor e o operador2 vai definir se o segmento vai ter mais do que uma linha.

- A segmentação que se implementou vai permitir a identificação de grupos de clientes ou contactos com base no seu perfil, preferências, comportamentos e necessidades. Esta vai permitir direccionar as campanhas para um determinado tipo de cliente.

- Na criação da campanha o nº da campanha é criado automaticamente, vai inserir-se o nome, a descrição, o produto, o produto da campanha, o cliente de destino a quem se dirige a campanha.

- Insere-se a estratégia que vai utilizar, os objectivos que se pretendem atingir com a respectiva campanha, o estado em que se encontra a campanha (inicio, execução e fim), o nome do funcionário responsável pela campanha, as datas e os custos da respectiva campanha.

- Na campanha cria-se as fases da campanha onde se vai poder inserir o nº, nome, descrição, objectivos, nº da campanha, nº do meio de comunicação que vai poder escolher na fase. Vai-se introduzir novos meios de comunicação ou escolher um já existente.

- Nos clientes vai-se adicionar e pesquisar clientes. Visualiza-se a informação geral do cliente, nº de identificação automático, inserir o nome, morada, dados pessoais, o tipo a que o

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

cliente pertence, a actividade, rendimentos, assim como os meios de contacto de que dispõe e qual o seu preferido.

2- Aplicação Microsoft Buisiness Solution – Navision;

Formatada: Inglês (E.U.A.)
Formatada: Inglês (E.U.A.)
Formatada: Inglês (E.U.A.)

- Relativamente ao produto pode-se visualizar a sua informação geral e a gestão de stocks.

Formatada: Português (Portugal)
Eliminado: ¶
¶

- O Funcionário encontra-se nos recursos humanos. Têm uma lista dos vários empregados e de todos os seus dados (estado, a hora de admissão na empresa, dados pessoais, formas de comunicação).

- Na Gestão de Relacionamento encontra-se informação sobre os Contactos (clientes), a Campanha e o Segmento.

- Nos contactos podem introduzir-se nomes (pessoas ou empresas), endereços, código postal da cidade e país, meios de comunicação usados (telefone, telemóvel, fax, telex, pager, nº de telex, email, página web), tipo de correspondência e idioma. A segmentação também se encontra nos contactos, onde se pode criar grupos de destino, excluir segmentos, visualizar uma lista Segmento Contacto (nº do segmento escolhido, a descrição, a data, o nº do contacto, assim como o nome do contacto escolhido). Nos contactos pode ainda visualizar-se uma lista das oportunidades de venda.

- Na Campanha pode visualizar-se uma lista das várias campanhas existentes e inserir a informação geral da campanha. As estatísticas da campanha vão conter os contactos de destino, os contactos (clientes) que responderam à campanha, a percentagem das respostas, os custos, a resposta e a duração.

- No Segmento encontra-se o nº de série, a descrição, ID do vendedor, datas, nº de linhas, nº de acções / critério. A campanha está ligada ao segmento, existe uma lista das campanhas que se podem escolher, não se pode criar campanhas.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

- Insere-se um grupo de destino (campo e filtro). No menu adicionar contactos insere o contacto (campo e filtro), o grupo destino, o perfil, entres outros. Podem-se introduzir segmentos reutilizando os já existentes. No entanto não se consegue chegar a uma conclusão quanto a inserir novos segmentos.

- No menu Vendas e Cobranças encontram-se entre outros as Vendas, os Clientes e as Devoluções.

- Nos Clientes encontra-se a informação geral dos clientes e as várias formas de comunicar com o Cliente.

- Nas formas de comunicação tem-se o telefone, o email e a Página Web, onde se pode encontrar uma ligação directa destas formas de comunicação para o respectivo cliente.

- No Diário de Vendas encontra-se a informação sobre as vendas efectuadas.

- Nas Devoluções pode-se introduzir os dados da devolução, os dados das vendas e visualizar-se uma lista das devoluções de produtos efectuadas.

3- Microsoft Dynamics CRM Version 3.0:

- Ao entrar na aplicação Web temos de fazer login (digitar o nome do utilizador e respectiva senha).

- Podem-se consultar as últimas novidades.

- Na agenda pode-se inserir um novo compromisso (ligação ou reunião – data de inicio, hora inicio e assunto) e guardar.

- Pode-se criar uma nova chamada, novo compromisso, nova tarefa.

- Nas Actividades pode pesquisar-se por assunto ou contacto.

Formatada: Rodapé

- Pode-se listar a ligações, agendar ligação, agendar reunião, criar uma nova tarefa e uma anotação ou anexo, arquivar e-mail, fazer ligações, arcar reuniões, tarefas, anotações, enviar e-mail, consultar calendário diário e importar anotações.

- Em contacto (clientes no EstgCRM), pode-se visualizar uma lista com vários contactos existentes. Fazer buscas por primeiro nome, por último nome, por número de cota. Inserir um novo contacto, criar cartões-de-visita e de *vCard*, contactos e importar.

- Nas Contas pode-se criar uma nova conta (nome da conta, telefone e *WebSite*), visualizar uma lista de contas (onde se pode visualizar o nome da conta, a cidade, o telefone, ver se é administrador ou utilizador normal) e fazer buscas.

- Em Potenciais, pode-se criar um novo potencial (primeiro nome, telefone e e-mail). No vCard pode-se visualizar uma lista de potenciais clientes (nome, estado, nome da conta, e-mail).

- Nas Oportunidades, pode-se criar uma nova oportunidade (nome da oportunidade, nome da conta, seleccionar data prevista, estado da venda e valor), visualizar uma lista de oportunidades.

- Em ocorrências, pode-se criar novas ocorrências (assunto, nome da conta), seleccionar e salvar. Lista de ocorrências.

- Na Central de suporte: *Home*, pode-se visualizar e listar chamados e actualizar em massa (estado, resolução, tipo, origem, prioridade) ou excluir. Criar Chamado (assunto, tipo e prioridade). Reportar chamados e visualizar chamados.

- Nos Documentos, pode-se visualizar uma lista de documentos e pesquisá-los ou seleccioná-los por documento, categoria ou subcategoria. Inserir novo documento e fazer a sua actualização em massa.

- Em E-mails: minha caixa de entrada pode-se pesquisar e-mails por status do e-mail, assunto ou contacto. Visualizar meus rascunhos, e-mails enviados, meus arquivos, caixa de entrada

Formatada: Tipo de letra: 11 pt

Formatada: Tipo de letra: Itálico

Formatada: Tipo de letra: Itálico

Formatada: Tipo de letra: Itálico

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

do grupo. Compor e-mail, novo e-mail, criar modelo de e-mail. Visualizar todos rascunhos, modelos de e-mails.

- Nas campanhas pode-se inserir uma nova campanha, pesquisar campanhas por nome, *newsletters*, nova lista de prospectos, novo modelo de *email*, modelos de email, importar possíveis clientes, configurar email, diagnóstico e criar formulário de potencial.

Formatada: Tipo de letra: Itálico

Formatada: Tipo de letra: Itálico

- Em projectos pode criar-se um novo projecto, seleccionar e guardar. Pesquisar por nome, criar nova tarefa e visualizar as tarefas do projecto. Visualizar uma lista de projectos (nome, estimativa de esforço total (horas), esforço real total (horas) e atribuído a). Actualizar ou excluir em massa.

- Em Utilizador - Administrador (ADMIN), pode-se criar um novo utilizador, visualizar utilizadores, editar, alterar senha e duplicar. Inserir parâmetros predefinidos: os utilizadores (se é administrador ou não). Informação sobre o utilizador, endereço, opções de agenda, de *layout*, de e-mail e configuração e confirmação de e-mail enviado. Pode salvar, cancelar e alterar senha. Pesquisar utilizadores por primeiro nome sobrenome, departamento, status. Visualizar uma lista de utilizadores. Actualizar utilizador.

Formatada: Tipo de letra: Itálico

- Em Compor e-mail pode-se editar as suas configurações, enviar, salvar como rascunho e cancelar e-mail.

- Em Funcionários pode criar um novo funcionário, visualizar os funcionários existentes. Pesquisar funcionários por nome, sobrenome, departamento ou status do funcionário. Na lista vamos encontrar o nome do funcionário, o departamento, e-mail, telefone, status e nome de utilizador. Pode-se ainda fazer uma actualização em massa dos dados. Formulário com os dados do funcionário.

- Na Administração tem as várias funções que o administrador gere.

- Em clientes potenciais pode-se introduzir a informação do potencial cliente.

- Em central de suporte pode mostrar o número de chamadas e a quem são atribuídas.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

- Nas ligações pode-se salvar a ligação (assunto, data e hora de inicio, atribuído a, duração, status, referente a e descrição). Cancelar a ligação, enviar convites e criar novo e ainda adicionar convidados.

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

6. Conclusões e Perspectivas de Desenvolvimento

Este trabalho de fim de curso permitiu a aplicação dos conhecimentos adquiridos ao longo do curso, nomeadamente analisar sistemas, criar bases de dados, desenvolver software, e programar em C# e SQL.

Para a análise do sistema CRM utilizou-se a linguagem UML. Assim, verificou-se a grandeza da UML no que diz respeito ao desenvolvimento de software através de gráficos, o que permite a qualquer pessoa, mesmo até a pessoas alheias à área de informática perceber o problema, e com isto prestarem um contributo mais valioso. Um ponto muito importante é a necessidade desta ferramenta para desenvolver qualquer tipo de software, pois esta permite uma maior entreajuda entre analistas e programadores. Uma análise bem elaborada, além de ajudar a entender o funcionamento da aplicação, vai facilitar o trabalho na implementação.

Concluiu-se que quase todos os objectivos do projecto foram alcançados, pois a aplicação CRM faz a segmentação (identificar os grupos de clientes ou contactos com base no seu perfil e suas preferências); faz a gestão do relacionamento com o cliente através dos diferentes meios de comunicação; regista as oportunidades de negócio surgidas através da campanha e dos inquéritos, cria inquéritos e permite que o cliente responda aos mesmos. Faltou desenvolver a parte para execução automática de campanhas de marketing.

A aplicação se vista numa perspectiva de desenvolvimento e como trabalho futuro propõe-se a elaboração de uma ferramenta de relatórios (*Reporting Services*) e a execução automática de campanhas. Uma das grandes vantagens que se pode verificar nesta aplicação EstgCRM em relação a outras aplicações analisadas é a simplicidade no processo de segmentação.

Formatada: Tipo de letra: Itálico

Formatada: Tipo de letra: Itálico

Formatada: Rodapé

O desenvolvimento desta aplicação foi muito enriquecedora, permitiu conhecer o funcionamento de vários sistemas de gestão de relacionamento com o cliente (CRM), perceber que o CRM não é uma moda que irá desaparecer em pouco tempo e que todos nós já ouvimos falar em CRM e sabemos dos seus benefícios e das suas vantagens potenciais

Formatada: Tipo de letra: 11 pt

Eliminado: .

Formatada: Rodapé

Formatada: Tipo de letra: 11 pt

Eliminado: <#>¶

7. Bibliografia

Figueiredo, B. (s.d.). *Estrutura, concepção e produção de sites web*. Obtido em 2008, de FCA-Livros de Informatica: http://www.fca.pt/

Boudart, S. (s.d.). *Apostila Microsoft SQL Server 7.0*. Obtido em 12 de 06 de 2007, de Apostilando.com: http://www.apostilando.com/download.php?cod=63&categoria=Banco%20de%20Dados

Eliminado: ¶

Formatada: Verificar ortografia e gramática

O'Neill, M. N.-H. (2004). *Fundamental de UML*. Lisboa: FCA.

Abreu, L. (2006). *ASP.NET 2.0 CURSO COMPLETO*. Lisboa: FCA.

Eliminado: ¶

Belo, O. SQL Server 2000 Para Profisionais. FCA.

Formatada: Inglês (E.U.A.)

Boudart, S. (s.d.). *Apostila Microsoft SQL Server 7.0*. Obtido em 12 de 06 de 2007, de Apostilando.com: http://www.apostilando.com/download.php?cod=63&categoria=Banco%20de%20Dados

Eliminado: ¶

Pesquisa e Desenvolvimento em UML, Alexandra de Almeida, Reginaldo Darolt, Araranguá – SC 2001 (Projecto de Ciências da Computação)

Eliminado: ¶
SQL Structured Query Language, Luís Ramos, FCA – Editora de Informática, ano??¶

Belo, O. SQL Server 2000 Para Profisionais. FCA.

Eliminado: ¶
C++ Builder 6 – Desenvolva Aplicações para Windows, William Pereira Alves, Editora Érica Ltda.ano¶
Silva, Alberto Carlos Videira – UML e Metodologias e Ferramentas Case editora e ano

Formatada: Inglês (E.U.A.)

Boudart, S. (s.d.). *Apostila Microsoft SQL Server 7.0*. Obtido em 12 de 06 de 2007, de Apostilando.com: http://www.apostilando.com/download.php?cod=63&categoria=Banco%20de%20Dados

O'Neill, M. N. *Fundamental de UML* . FCA.

Formatada: Rodapé

Videira, A. S. (2005). *UML - Metodologias e Ferramentas CASE*. Centro Atlântico.

Belo, O. SQL Server 2000 Para Profisionais. FCA.

Customer relationship management. (26 de Dezembro de 2008). Obtido em 27 de Dezembro de 2008, de Customer relationship management-Wikipédia, a enciclopédia livre: http://pt.wikipedia.org/wiki/Customer_relationship_management

Figueiredo, B. (s.d.). *Estrutura, concepção e produção de sites web*. Obtido em 2008, de FCA-Livros de Informatica: http://www.fca.pt/

Boudart, S. (s.d.). Apostila Microsoft SQL Server 7.0. Obtido em 12 de 06 de 2007, de Apostilando.com: http://www.apostilando.com/download.php?cod=63&categoria=Banco%20de%20Dados

Lobo, L. (2002). *CRM - Uma nova etapa...* Obtido em 23 de 09 de 2008, de Nova Base: http://www.novabase.pt/showNews.asp?idProd=rescrm

O'Neill, M. N. *Fundamental de UML* . FCA.

Videira, A. S. (2005). *UML - Metodologias e Ferramentas CASE*. Centro Atlântico.

Formatada: Tipo de letra: 11 pt

Eliminado: ¶
¶

Eliminado:

Eliminado: Nunes, Mauro/Henrique O'Neill, - Fundamental UML, editora e ano¶

Eliminado: ¶
Boudart, S. (s.d.). Apostila Microsoft SQL Server 7.0. Obtido em 12 de 06 de 2007, de Apostilando.com: http://www.apostilando.com/download.php?cod=63&categoria=Banco%20de%20Dados¶
O'Neill, M. N. Fundamental de UML . FCA.¶
¶
¶
¶
¶
¶

Eliminado: ¶
¶
¶
¶
¶

Eliminado: Belo, O. SQL Server 2000 Para Profisionais. FCA.¶

Formatada: Inglês (E.U.A.)

Comentário [C3]: Para todos os sítios

Eliminado: Autor, título e data da consulta http://www.fca.pt¶
http://www.pmpt.pt.vu¶

Eliminado: ¶
http://www.pmfa.pt.to¶
http://www.infobahia.com.br - (Exemplo de aplicações crm)¶
http://www.novabase.pt¶
http://www.emadar.com¶
http://www.Iscrm.com¶
http://WWW.apostilando.com – (Apostila Microsoft SQL Server 7.0)¶
http://www.sofstack.com – (freemstp)¶
http://www.amazon.com/¶
www.boldcrm.com¶
www.browsercrm.com¶
www.exposis.com¶

Eliminado: www.exposis.com

Formatada: Tipo de letra predefinido do parágrafo

Formatada: Rodapé

ANEXOS

CÓDIGO da APLICAÇÃO

Cliente.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Web.Services;
using System.Web.Services.Protocols;
//using System.Data.OracleClient;
using System.Data.Sql;
using System.IO;
// para Prostgresql
//using Npgsql;
//using NpgsqlTypes;
//using Npgsql.Design;
// Librarias para sql server
using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Text;

using System.Security.Cryptography;

/// <summary>
/// Summary description for Cliente
/// </summary>
public class Cliente
{

    private string OraconStr =
System.Configuration.ConfigurationManager.ConnectionStrings["ProjcrmCo
nnectionString"].ToString(); //para sqlserver

    private string pNome;
    private string pMorada;
    private int pNumContribuinte;
    private int pCodPostal;
    private string pLocalidade;
    private string pSexo;
    private string pEstCivil;
    private int pNumFilhos;
    private string pActividade;
    private string pRendimentos;
    private int pNumTelefone;
    private int pNumTelemovel;
    private int pIdade;
    private string pContpreferido;
    private string pFreqContacto;
    private string pCurso;
    private int pAnoCurso;
    private int pIDCliente;
    private string pPassword;
    private string pEmail;
    private bool pAdministrador;

    public Cliente()
      {
```

```
  }

public int IDCliente
{
    get
    {
        return pIDCliente;
    }
    set
    {
        this.pIDCliente= value;
    }
}

public string Nome
{
    get
    {
        return this.pNome;
    }
    set
    {
        this.pNome = value;
    }
}

public string Morada
{
    get
    {
        return this.pMorada;

    }
    set
    {
        this.pMorada = value;
    }
}

public int NumContribuinte
{
    get
    {
        return this.pNumContribuinte;
    }
    set
    {
        this.pNumContribuinte = value;
    }
}

public string Actividade
{
    get
    {
        return this.pActividade;
    }
    set
    {
        this.pActividade = value;
```

```
    }
}

public int NumTelefone
{
    get
    {
        return this.pNumTelefone;
    }
    set
    {
        this.pNumTelefone = value;
    }
}

public string EstCivil
{
    get
    {
        return this.pEstCivil;
    }
    set
    {
        this.pEstCivil = value;
    }
}

public string Contpreferido
{
    get
    {
        return this.pContpreferido;
    }
    set
    {
        this.pContpreferido = value;
    }
}

public string CodPostal
{
    get
    {
        string saux = this.pCodPostal.ToString();
        saux.Insert(3, "-");
        return saux;
    }
    set
    {
        string saux ;

        int auxind = value.IndexOf('-');
        if (auxind >= 0)
            saux = value.Remove(auxind, 1);
        else
            saux = value;

        this.pCodPostal = Convert.ToInt32(saux);
    }
}
```

```
public string Sexo
{
    get
    {
        return this.pSexo;
    }
    set
    {
        this.pSexo = value;
    }
}

public int NumTelemovel
{
    get
    {
        return this.pNumTelemovel;
    }
    set
    {
        this.pNumTelemovel = value;
    }
}

public string Localidade
{
    get
    {
        return this.pLocalidade;
    }
    set
    {
        this.pLocalidade = value;
    }
}

public string Password
{
    get
    {
        return this.pPassword;
    }
    set
    {
        this.pPassword = value;
    }

}

public string Email
{
    get
    {
        return this.pEmail;
    }
    set
    {
        this.pEmail = value;
    }
}
```

```
public string FreqContacto
{
    get
    {
        return this.pFreqContacto;
    }
    set
    {
        this.pFreqContacto = value;
    }
}

public string Curso
{
    get
    {
        return this.pCurso;
    }
    set
    {
        this.pCurso = value;
    }
}

public int AnoCurso
{
    get
    {
        return this.pAnoCurso;
    }
    set
    {
        this.pAnoCurso = value;
    }
}

public int Idade
{
    get
    {
        return this.pIdade;
    }
    set
    {
        this.pIdade = value;
    }
}

public int NumFilhos
{
    get
    {
        return this.pNumFilhos;
    }
    set
    {
        this.pNumFilhos = value;
    }
}

public string Rendimentos
```

```
    {
        get
        {
            return this.pRendimentos;
        }
        set
        {
            this.pRendimentos = value;
        }
    }


    public bool Administrador
    {
        get
        {
            return this.pAdministrador;
        }
        set
        {
            this.pAdministrador = value;
        }
    }
/*
 *  Métodos
 *
 */

public bool SetPassword(int idCliente,string Password)
    {
        StringBuilder Sb = new StringBuilder();
        using (SqlConnection ConexaoSQL = new
SqlConnection(OraconStr))
        {
            ConexaoSQL.Open();
            SqlCommand ComandoSQL = ConexaoSQL.CreateCommand();
            SqlTransaction TransacaoSQl =
ConexaoSQL.BeginTransaction();
            ComandoSQL.Transaction = TransacaoSQl;
            Sb.Append("UPDATE CLIENTE SET Password=@Password where
idCliente=@idCliente");
            ComandoSQL.CommandText=Sb.ToString();
            ComandoSQL.Parameters.Clear();

//ComandoSQL.Parameters.AddWithValue("@Password",Password.ToString());
            ComandoSQL.Parameters.Add("@Password",
SqlDbType.VarChar).Value = Password.ToString().Trim();

ComandoSQL.Parameters.AddWithValue("@idCliente",idCliente);

            try {
                ComandoSQL.ExecuteNonQuery();
                TransacaoSQl.Commit();
                return true;
            }
            catch (Exception e)
            {
                TransacaoSQl.Rollback();
                return false;
            }
        }
```

```
    }
    /// <summary>
    /// Retorna a password de um dado cliente
    /// </summary>
    /// <param name="idCliente"></param>
    /// <returns>string</returns>
    public string  getPassword(int idCliente)
    {

          using (SqlConnection ConexaoSQL = new
SqlConnection(OraconStr))
          {
              ConexaoSQL.Open();
              SqlCommand ComandoSQL = ConexaoSQL.CreateCommand();
             ComandoSQL.CommandText= "SELECT PASSWORD FROM CLIENTE
WHERE idCliente=@idCliente";
             ComandoSQL.Parameters.Clear();
             ComandoSQL.Parameters.AddWithValue("idCliente",
idCliente);
           string str=(string)  ComandoSQL.ExecuteScalar();
           ConexaoSQL.Close();
           return str;

          }

    }

    //criar um cliente na base dados, retornando sucesso ou insucesso
    /// <summary>
    ///  Insere um Novo Cliente na Base De Dados  pode Ter o Atributo
de Administrador ou nao
    /// </summary>
    /// <value></value>
    /// <returns></returns>

    public int Insere()
    {
        SqlConnection OraConn = new SqlConnection();

        OraConn.ConnectionString = OraconStr;

        using (OraConn)
        {

            OraConn.Open();

            SqlCommand OraCmd = OraConn.CreateCommand();

            SqlTransaction OraTrans;
            StringBuilder OraQuery = new StringBuilder();

            OraTrans = OraConn.BeginTransaction();
            OraCmd.Transaction = OraTrans;

            try
            {

                OraCmd.CommandText = "select count(IDCliente) as cnt
from CLiente where UPPER(Email) like ";
                OraCmd.CommandText += "'" + pEmail.ToUpper() + "'";
```

```
                SqlDataReader OraReader = OraCmd.ExecuteReader();
                int cnt = 0;
                while (OraReader.Read())
                {
                    cnt =
Convert.ToInt32(OraReader["cnt"].ToString());
                }
                OraReader.Close();

                if (cnt != 0)
                {

                    OraTrans.Rollback();
                    return 2; //diz que o utilizador ja existe na Base
de Dados
                }

                if (OraQuery.Length != 0)
                {
                    OraQuery.Remove(0, OraQuery.Length);
                }

                String Str = "insert into
Cliente(NumContribuinte,Nome,Morada,Localidade,CodPostal,NumTelefone,N
umTelemovel,";

Str+="Email,Password,Idade,Sexo,EstCivil,NumFilhos,Actividade,Rendimen
tos,Contpreferido,FreqContacto,Curso,AnoCurso,ADMINISTRADOR) values(";
                Str +=
"@NumContribuinte,@Nome,@Morada,@Localidade,@CodPostal,@NumTelefone,@N
umTelemovel,@Email,@Password,@Idade,@Sexo,@EstCivil,";
                Str +=
"@NumFilhos,@Actividade,@Rendimentos,@Contpreferido,@FreqContacto,@Cur
so,@AnoCurso,@ADMINISTRADOR)";

                OraCmd.CommandText = Str.ToString();
                OraCmd.Parameters.Clear();

OraCmd.Parameters.AddWithValue("NumContribuinte",pNumContribuinte);
                OraCmd.Parameters.AddWithValue("Nome", pNome);
                OraCmd.Parameters.AddWithValue("Morada", pMorada);
                OraCmd.Parameters.AddWithValue("Localidade",
pLocalidade);
                OraCmd.Parameters.AddWithValue("CodPostal",
pCodPostal);
                OraCmd.Parameters.AddWithValue("NumTelefone",
pNumTelefone);
                OraCmd.Parameters.AddWithValue("NumTelemovel",
pNumTelemovel);
                OraCmd.Parameters.AddWithValue("Email", pEmail);
                OraCmd.Parameters.AddWithValue("Password", pPassword);
                OraCmd.Parameters.AddWithValue("Idade", pIdade);
                OraCmd.Parameters.AddWithValue("Sexo", pSexo);
                OraCmd.Parameters.AddWithValue("EstCivil", pEstCivil);
                OraCmd.Parameters.AddWithValue("NumFilhos",
pNumFilhos);
                OraCmd.Parameters.AddWithValue("Actividade",
pActividade);
                OraCmd.Parameters.AddWithValue("Rendimentos",
pRendimentos);
```

```
                OraCmd.Parameters.AddWithValue("Contpreferido",
pContpreferido);
                OraCmd.Parameters.AddWithValue("FreqContacto",
pFreqContacto);
                OraCmd.Parameters.AddWithValue("Curso", pCurso);
                OraCmd.Parameters.AddWithValue("AnoCurso", pAnoCurso);
                OraCmd.Parameters.AddWithValue("ADMINISTRADOR",
pAdministrador);

                OraCmd.ExecuteNonQuery();
                OraTrans.Commit();
                return 1;// sucesso
            }
            catch (Exception e)
            {
                string strfx = "\\logs.txt";
                using (StreamWriter sw = File.AppendText(strfx))
                {
                    sw.WriteLine("##" + System.DateTime.Now +
OraCmd.CommandText.ToString() + "##");
                }
                OraTrans.Rollback();

                return 0;
            }

        }
    }
    /// <summary>
    /// Verifica se um dado Cliente com um dado Email ja existe na
base de Dados
    /// </summary>
    /// <param name="Email"></param>
    /// <returns></returns>
    public bool ExisteCliente(String Email)
    {
       StringBuilder Sb = new StringBuilder();
       using ( SqlConnection ConexaoSQL = new
SqlConnection(OraconStr))
        {
            ConexaoSQL.Open();
            Sb.Append("select count(IDCliente) as cnt from CLiente
where UPPER(Email) like @EMAIL");
            SqlCommand ComandoSQL = ConexaoSQL.CreateCommand();
            ComandoSQL.CommandText = Sb.ToString();
            ComandoSQL.Parameters.Clear();
            ComandoSQL.Parameters.AddWithValue("EMAIL", Email);
            int valor=(int) ComandoSQL.ExecuteScalar();
            if (valor !=0)
                return true ;
            else
                return false;

        }

    }

    public bool VerificaAdmin(int idCliente)
    {
        bool valor;
```

```
        using (SqlConnection ConexaoSQL = new
SqlConnection(OraconStr))
        {
            ConexaoSQL.Open();
            SqlCommand ComandoSQL = ConexaoSQL.CreateCommand();
            ComandoSQL.CommandText = "SELECT Administrador FROM
CLIENTE WHERE idCliente=@idCliente";
            ComandoSQL.Parameters.Clear();
            ComandoSQL.Parameters.AddWithValue("idCliente",
idCliente);
            try
            {
                valor = (bool)ComandoSQL.ExecuteScalar();
                return valor;

            }
            catch (Exception ex)
            {
                return false;
            }
        }
        return false;

    }
    /// <summary>
    /// Retorna o id do cliente
    /// </summary>
    /// <param name="Email"></param>
    /// <returns></returns>
    public int GetIdCliente(string Email)
    {
        using (SqlConnection ConexaoSQL = new
SqlConnection(OraconStr))
        {
            ConexaoSQL.Open();
            SqlCommand ComandoSQL = ConexaoSQL.CreateCommand();
            ComandoSQL.CommandText = "SELECT IdCliente FROM CLIENTE
WHERE UPPER(Email) like @EMAIL";
            ComandoSQL.Parameters.Clear();
            ComandoSQL.Parameters.AddWithValue("EMAIL", Email);
            Decimal idCliente = (Decimal)ComandoSQL.ExecuteScalar();
           return Convert.ToInt32(idCliente);
        }
    }
}
```

CriarFase.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

using System.Data.SqlClient;
using System.Data.SqlTypes;
```

```
using System.Text;


using System.Data.Sql;
using System.IO;



/// <summary>
/// Summary description for CriarFase
/// </summary>
public class CriarFase
{
    private string OraconStr =
System.Configuration.ConfigurationManager.ConnectionStrings["ProjcrmCo
nnectionString"].ToString();
   //private string conStr =
System.Configuration.ConfigurationManager.ConnectionStrings["ConexaoSq
l"].ConnectionString;

        //private int pIDFase;
        private string pNome;
        private string pDescricao;
        private string pObjectivo;

        public CriarFase()
        {

        }
       public string Nome
        {
            get
            {
                return this.pNome;
            }
            set
            {
                this.pNome = value;
            }
        }

        public string Descricao
        {
            get
            {
                return this.pDescricao;
            }
            set
            {
                this.pDescricao = value;
            }
        }

        public string Objectivo
        {
            get
            {
                return this.pObjectivo;
            }
            set
            {
```

```
                this.pObjectivo = value;
            }
        }

        public bool Insere()
        {
            SqlConnection OraConn = new SqlConnection();
            OraConn.ConnectionString = OraconStr;

            using (OraConn)
            {
                OraConn.Open();

                SqlCommand OraCmd = OraConn.CreateCommand();

                SqlTransaction OraTrans;
                StringBuilder OraQuertyStr = new StringBuilder();

                OraTrans = OraConn.BeginTransaction();
                OraCmd.Transaction = OraTrans;


                OraQuertyStr.Append("INSERT INTO
Fase(Nome,Descricao,Objectivo) ");
                OraQuertyStr.Append("
VALUES(@Nome,@Descricao,@Objectivo)");

                try
                {
                    OraCmd.Parameters.Clear();
                    OraCmd.Parameters.Add("Nome", SqlDbType.VarChar,
pNome.Length, "NOME").Value = pNome;
                    OraCmd.Parameters.Add("Descricao",
SqlDbType.VarChar, pDescricao.Length, "DESCRICAO").Value = pDescricao;
                    OraCmd.Parameters.Add("Objectivo",
SqlDbType.VarChar, pObjectivo.Length, "OBJECTIVO").Value = pObjectivo;

                    OraCmd.CommandText = OraQuertyStr.ToString();
                    OraCmd.ExecuteNonQuery();
                    OraTrans.Commit();
                }
                catch (Exception e)
                {
                    OraTrans.Rollback();
                    return false;
                }

            }
            return true;

        }

        public void Remove()
        {
            throw new System.NotImplementedException();
        }
}
```

DefinirMeioComunicacao.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Text;


/// <summary>
/// Summary description for MeioComunicacao
/// </summary>
public class MeioComunicacao
{
    private string conStr =
System.Configuration.ConfigurationManager.ConnectionStrings["ConexaoSq
l"].ConnectionString;

    private int pIDMeioComunicacao;
    private string pTipo;


      public MeioComunicacao()
      {

      }

    public int IDMeioComunicacao
    {
        get
        {
            return pIDMeioComunicacao;
        }
        set
        {
            this.pIDMeioComunicacao= value;
        }
    }

    public string Tipo
    {
        get
        {
            return this.pTipo;
        }
        set
        {
            this.pTipo = value;
        }
    }

    public bool Insere(string Tipo)
    {
```

```
        int index = 0;
        SqlConnection ConexaoSql = new SqlConnection(conStr);
        StringBuilder TextoSql = new StringBuilder();
        SqlCommand ComandoSql = ConexaoSql.CreateCommand();
        ConexaoSql.Open();
        SqlTransaction TransacaoSQL = ConexaoSql.BeginTransaction();

        ComandoSql.Transaction = TransacaoSQL;
        try
        {
            TextoSql.Remove(0, TextoSql.Length);
            TextoSql.Append("INSERT INTO MeioComunicacao(Tipo)");
            TextoSql.Append(" ");
            TextoSql.Append("VALUES(@TIPO)");
            ComandoSql.CommandText = TextoSql.ToString();
            ComandoSql.Parameters.Clear();
            ComandoSql.Parameters.AddWithValue("TIPO", pTipo);
            ComandoSql.ExecuteNonQuery();
            TransacaoSQL.Commit();
            return true;
        }
        catch (Exception ex)
        {
            TransacaoSQL.Rollback();
            return false;
        }
    }

}
```

Estrategia.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

using System.Web.Services;
using System.Web.Services.Protocols;

using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Data.Sql;

using System.Web.SessionState;
using System.Text;
using System.IO;
using System.Security.Cryptography;


/// <summary>
/// Summary description for Estrategia
/// </summary>
public class Estrategia
{
```

```
    private string conStr =
System.Configuration.ConfigurationManager.ConnectionStrings["ConexaoSq
l"].ConnectionString;

   // private int pIDEstrategia;
    private string pDescricao;
    private string pUtilizada;

      public Estrategia()
      {
      }

    /*public int IDEstrategia
    {
        get
        {
            return pIDEstrategia;
        }
        set
        {
            this.pIDEstrategia = value;
        }
    }*/

    public string Descricao
    {
        get
        {
            return this.pDescricao;
        }
        set
        {
            this.pDescricao = value;
        }
    }

    public string Utilizada
    {
        get
        {
            return this.pUtilizada;
        }
        set
        {
            this.pUtilizada = value;
        }
    }


    public bool Insere()
    {
       // int index = 0;
        SqlConnection ConexaoSql = new SqlConnection(conStr);
        StringBuilder TextoSql = new StringBuilder();
        SqlCommand ComandoSql = ConexaoSql.CreateCommand();
        ConexaoSql.Open();
        SqlTransaction TransacaoSQL = ConexaoSql.BeginTransaction();

        ComandoSql.Transaction = TransacaoSQL;
        try
        {
```

```
            TextoSql.Remove(0, TextoSql.Length);
            TextoSql.Append("INSERT INTO
Estrategia(Descricao,Utilizada)");
            //TextoSql.Append(" ");
            TextoSql.Append("VALUES(@DESCRICAO,@UTILIZADA)");
            ComandoSql.CommandText = TextoSql.ToString();
            ComandoSql.Parameters.Clear();

            ComandoSql.Parameters.Add("Descricao", SqlDbType.VarChar,
pDescricao.Length, "DESCRICAO").Value = pDescricao;
            ComandoSql.Parameters.Add("Utilizada", SqlDbType.VarChar,
3, "UTILIZADA").Value = pUtilizada;


            ComandoSql.ExecuteNonQuery();
            TransacaoSQL.Commit();
            return true;

        }
        catch (Exception ex)
        {
            TransacaoSQL.Rollback();
            return false;
        }
    }


/*
    public void Remove()
    {
        throw new System.NotImplementedException();
    }

    public void Lista()
    {
        throw new System.NotImplementedException();
    }
    */
}
```

Inqueritos.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Text;
using System.Collections.Generic;



/// <summary>
```

```
/// Summary description for Inqueritos
/// Esta Classe  Premite Modelizar o inqeritos Bem Como associar a uma
terminada Campanha
///
/// </summary>
public class Inqueritos
{
    private string strConexao =
System.Configuration.ConfigurationManager.ConnectionStrings["ConexaoSq
l"].ConnectionString;
    private SqlConnection connSql;


      public Inqueritos()
      {
            //
            // TODO: Add constructor logic here
            //
        connSql = new SqlConnection(strConexao);

      }
    /// <summary>
    ///  Cria um inquerito na base de Dados
    /// </summary>
    /// param name="Desc"  Titulo do Inquerito
    /// <returns></returns>
    public bool Cria_inquerito(String Desc)
    {
        StringBuilder StrCmd = new StringBuilder();
        int maxid,aux;
        SqlCommand Sqlcmd = connSql.CreateCommand();
        connSql.Open();
        SqlTransaction sqlTrans=connSql.BeginTransaction();
        Sqlcmd.Transaction = sqlTrans;

        StrCmd.Remove(0, StrCmd.Length);
        StrCmd.Append("SELECT max(idinquerito) from inquerito;");
        try
        {
            maxid = 0;

            Sqlcmd.CommandText=StrCmd.ToString();
          SqlDataReader LeituraSql=Sqlcmd.ExecuteReader();
            while (LeituraSql.Read())
            {
               if (LeituraSql.IsDBNull(0))
                maxid=0;
               else
                maxid=LeituraSql.GetInt32(0);
            }
            LeituraSql.Close();



            maxid++;
         StrCmd.Remove(0,StrCmd.Length);
         StrCmd.Append("INSERT INTO
INQUERITO(idinquerito,Descinquerito)
VALUES(@idinqueriro,@Descinquerito);");
         Sqlcmd.CommandText=StrCmd.ToString();
         Sqlcmd.Parameters.Clear();
```

```
            Sqlcmd.Parameters.AddWithValue("idinqueriro", maxid);
            Sqlcmd.Parameters.AddWithValue("Descinquerito", Desc);


           aux= Sqlcmd.ExecuteNonQuery();

              if (aux!=0)
              {
                  sqlTrans.Commit();
                  return true;

              }
              else
              {
                  return false;
              }

           }
           catch (Exception ex)
           {
               sqlTrans.Rollback();
               return false;

           }

       }
       /// <summary>
       ///  Inserte Perguntas  a um Determinado inquerito
       /// </summary>
       /// <param name="id_inquerito"></param>
       /// <param name="idpergunta"></param>
       /// <returns></returns>
       public bool InserereLinhaInquerito(int idinquerito,int idpergunta)
       {

           StringBuilder StrCmd = new StringBuilder();
           int contagem, aux,ordem;
           SqlCommand Sqlcmd = connSql.CreateCommand();
           connSql.Open();
           SqlTransaction sqlTrans = connSql.BeginTransaction();
           Sqlcmd.Transaction = sqlTrans;

           StrCmd.Remove(0, StrCmd.Length);
           StrCmd.Append("SELECT count(*) from Linhasinquerito WHERE
idPergunta=");
           StrCmd.Append(idpergunta);
           StrCmd.Append(" AND IDINQUERITO=");
           StrCmd.Append(idinquerito);
           Sqlcmd.CommandText = StrCmd.ToString();


           try
           {
               contagem = 0;


               contagem = (int)Sqlcmd.ExecuteScalar();
               if (contagem!=0)
                   return false;
```

```
            Sqlcmd.CommandText="SELECT max(ordem) from linhasinquerito
where ";
            Sqlcmd.CommandText+=" idinquerito="+idinquerito.ToString();
             ordem=0;
            SqlDataReader LeituraSQL=Sqlcmd.ExecuteReader();
             while (LeituraSQL.Read())
             {
                 if (LeituraSQL.IsDBNull(0))
                     ordem=0;
                 else
                     ordem=LeituraSQL.GetInt32(0);

             }
             ordem++;
             LeituraSQL.Close();
             StrCmd.Remove(0, StrCmd.Length);
             StrCmd.Append("INSERT INTO
linhasinquerito(idinquerito,idPergunta,ordem)
VALUES(@idinqueriro,@idPergunta,@Ordem);");
             Sqlcmd.CommandText = StrCmd.ToString();
             Sqlcmd.Parameters.Clear();
             Sqlcmd.Parameters.AddWithValue("idinqueriro",
idinquerito);
             Sqlcmd.Parameters.AddWithValue("idPergunta", idpergunta);
             Sqlcmd.Parameters.AddWithValue("Ordem", ordem);



             aux = Sqlcmd.ExecuteNonQuery();

             if (aux != 0)
             {
                 sqlTrans.Commit();
                 return true;

             }
             else
             {
                 return false;
             }

         }
         catch (Exception ex)
         {
             sqlTrans.Rollback();
             return false;

         }
     }
     public List<int> RetornaIdPErguntasInq(int idinq)
     {
         List<int> ListaidPerguntas=new List<int>();

         try
         {

         SqlConnection ConexaoSQL=new SqlConnection(strConexao);
         ConexaoSQL.Open();
         SqlCommand ComandoSQL=ConexaoSQL.CreateCommand();
```

```
        SqlTransaction TranscaoSql=ConexaoSQL.BeginTransaction();
        ComandoSQL.Transaction=TranscaoSql;
             ComandoSQL.CommandText="SELECT IdPergunta FROM
LINHASINQUERITO WHERE idinquerito="+idinq.ToString();
            ComandoSQL.CommandText+=" ORDER BY ordem";
         SqlDataReader LeituraSQL=  ComandoSQL.ExecuteReader();
         ListaidPerguntas.Clear();
            while  (LeituraSQL.Read())
            {
              ListaidPerguntas.Add(LeituraSQL.GetSqlInt32(0).Value);
            }
            LeituraSQL.Close();
            TranscaoSql.Commit();
            ConexaoSQL.Close();
            return ListaidPerguntas;

        }
        catch (Exception Ex)
        {
            return null;
        }

    }
/// <summary>
/// Asscia um dado inquerito a uma Determinada Campanha
/// </summary>
/// <param name="idInquerito"></param>
/// <param name="idCampanha"></param>
    public bool AssocInqACampanha(int idInquerito, int idCampanha)
    {


        using (SqlConnection ConexaoSQL = new
SqlConnection(strConexao))
        {
            ConexaoSQL.Open();
            SqlCommand ComandoSQL = ConexaoSQL.CreateCommand();
            //Começamos uma transaçao local
            SqlTransaction TransacaoSQL =
ConexaoSQL.BeginTransaction();
            ComandoSQL.Transaction = TransacaoSQL;
            try
            {
                ComandoSQL.CommandText = "INSERT INTO
INQUERITO_CAMPANHA(IDCampanha,IDInquerito)
VALUES(@IDCampanha,@IDInquerito);";
                ComandoSQL.Parameters.Clear();
                ComandoSQL.Parameters.AddWithValue("IDCampanha",
idCampanha);
                ComandoSQL.Parameters.AddWithValue("IDInquerito",
idInquerito);
                ComandoSQL.ExecuteNonQuery();
                TransacaoSQL.Commit();
                return true;
            }
            catch (Exception ex)
            {
                TransacaoSQL.Rollback();
                return false;
            }
```

```
        }


    }
    /// <summary>
    /// Desacossia um Dado inquerito a Uma Dada Campanha
    /// </summary>
    /// <param name="idInquerito"></param>
    /// <param name="idCampanha"></param>
    public bool DesacAssocInqACampanha(int idInquerito, int
idCampanha)
    {
        using (SqlConnection ConexaoSQL = new
SqlConnection(strConexao))
        {
            ConexaoSQL.Open();
            SqlCommand ComandoSQL = ConexaoSQL.CreateCommand();
            //Começamos uma transaçao local
            SqlTransaction TransacaoSQL =
ConexaoSQL.BeginTransaction();
            ComandoSQL.Transaction = TransacaoSQL;
            try
            {
                ComandoSQL.CommandText = "DELETE FROM
INQUERITO_CAMPANHA  WHERE (IDCampanha=@IDCampanha  and
IDInquerito=@IDInquerito);";
                ComandoSQL.Parameters.Clear();
                ComandoSQL.Parameters.AddWithValue("IDCampanha",
idCampanha);
                ComandoSQL.Parameters.AddWithValue("IDInquerito",
idInquerito);
                ComandoSQL.ExecuteNonQuery();
                TransacaoSQL.Commit();
                return true;
            }
            catch (Exception ex)
            {
                TransacaoSQL.Rollback();
                return false;
            }

        }
    }

    /// <summary>
    ///  Retorna Quais os inqueritos que o cliente  ainda nao
respondeu em tempo
    ///  E que e cuja a campanha ainda não Expirou
    /// </summary>
    /// <param name="idCliente"></param>
    /// <returns> DataSet com o id da Campanha  e o id do Cliente
</returns>
    public DataSet RetornaInqueritosNaoRespondidosPeloCliente(int
idCliente)
    {
        StringBuilder Sb = new StringBuilder();

        using (SqlConnection ConexaoSQL = new
SqlConnection(strConexao))
        {
            DataSet Dset = new DataSet();
```

```
            ConexaoSQL.Open();

            //ComandoSQL.CommandText = "Select ";
            Sb.Remove(0, Sb.Length);


            Sb.Append("SELECT camp.idcampanha as
IDCampanha,inq.idInquerito as IdInquerito ,inq.descinquerito as
Desc_inquerito ");
            Sb.Append(" FROM  inquerito inq  INNER JOIN
inquerito_campanha inq_camp ");
            Sb.Append("ON  inq.idinquerito= inq_camp.idinquerito ");
            Sb.Append("INNER JOIN campanha camp ");
            Sb.Append("ON inq_camp.idcampanha= camp.idcampanha ");
            Sb.Append("INNER JOIN  segmento segm ");
            Sb.Append("ON camp.idsegmento=segm.idsegmento ");
            Sb.Append("INNER JOIN  clientes_segmento cli_seg ");
            Sb.Append("ON segm.idsegmento=cli_seg.idsegmento ");
            Sb.Append("INNER JOIN cliente cli ");
            Sb.Append("ON cli_seg.idcliente=cli.idcliente ");
            Sb.Append("WHERE ((cli.idcliente=@IdCliente and
inq_camp.datafim>=getdate()))");

            SqlDataAdapter SqlDa = new
SqlDataAdapter(Sb.ToString(),ConexaoSQL);

            SqlDa.SelectCommand.Parameters.Add("IdCliente",
SqlDbType.Int).Value = idCliente;
            try
            {
                SqlDa.Fill(Dset);
                return Dset;
            }
            catch (Exception Ex)
            {

            }


            return null;
        }

    }


}
```

LinhasSegmento.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
```

```
using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Text;

/// <summary>
/// Summary description for LinhasSegmento
/// </summary>
public class LinhasSegmento
{
    private string conStr =
System.Configuration.ConfigurationManager.ConnectionStrings["ConexaoSq
l"].ConnectionString;

    private int pIDSegLinha;
    private string pAtributo1;
    private char pOperador;
    private string pValor;
    private char pOperador2;
    private int pIDSegmento;

      public LinhasSegmento()
      {
      }

    public int IDSegLinha
    {
        get
        {
            return pIDSegLinha;
        }
        set
        {
            this.pIDSegLinha = value;
        }
    }

    public string Atributo1
    {
        get
        {
            return this.pAtributo1;
        }
        set
        {
            this.pAtributo1 = value;
        }
    }

    public char Operador
    {
        get
        {
            return this.pOperador;
        }
        set
        {
            this.pOperador = value;
        }
    }
```

```
    public string Valor
    {
        get
        {
            return this.pValor;
        }
        set
        {
            this.pValor = value;
        }
    }

    public char Operador2
    {
        get
        {
            return this.pOperador2;
        }
        set
        {
            this.pOperador2 = value;
        }
    }

    public int IDSegmento
    {
        get
        {
            return pIDSegmento;
        }
        set
        {
            this.pIDSegmento = value;
        }
    }

    //public bool Insere(string Atributo1, char Operador, string
Valor, char Operador2, int IDSegmento)
    public bool Insere()
    {
        //int index = 0;
        SqlConnection ConexaoSql = new SqlConnection(conStr);
        StringBuilder TextoSql = new StringBuilder();
        SqlCommand ComandoSql = ConexaoSql.CreateCommand();
        ConexaoSql.Open();
        SqlTransaction TransacaoSQL = ConexaoSql.BeginTransaction();

        ComandoSql.Transaction = TransacaoSQL;
        try
        {
            TextoSql.Remove(0, TextoSql.Length);
            TextoSql.Append("INSERT INTO
SegmentoLinha(Atributo1,Operador,Valor,Operador2,IDSegmento)");
            TextoSql.Append(" ");

//TextoSql.Append("VALUES(@ATRIBUTO1,@OPERADOR,@VALOR,@OPERADOR2,@IDSE
GMENTO)");
            ComandoSql.CommandText = TextoSql.ToString();
            ComandoSql.Parameters.Clear();
```

```
            ComandoSql.Parameters.Add("Atributo1", SqlDbType.VarChar,
pAtributo1.Length, "ATRIBUTO1").Value = pAtributo1;
            ComandoSql.Parameters.Add("Operador", SqlDbType.Char, 10,
"OPERADOR").Value = Convert.ToChar(pOperador);
            ComandoSql.Parameters.Add("Valor", SqlDbType.VarChar,
pValor.Length, "VALOR").Value = pValor;
            ComandoSql.Parameters.Add("Operador2", SqlDbType.Char, 10,
"OPERADOR2").Value = Convert.ToChar(pOperador2);
            ComandoSql.Parameters.Add("IDSegmento", SqlDbType.VarChar,
9, "IDSEGMENTO").Value = pIDSegmento;

            ComandoSql.ExecuteNonQuery();
            TransacaoSQL.Commit();
            return true;

        }
        catch (Exception ex)
        {
            TransacaoSQL.Rollback();
            return false;
        }
    }

}
```

Pergunta.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Text;
/// <summary>
/// Summary description for Pergunta
/// </summary>
public class Pergunta
{
    private string conStr =
System.Configuration.ConfigurationManager.ConnectionStrings["ConexaoSq
l"].ConnectionString;

      public Pergunta()
      {
            //
            // TODO: Add constructor logic here
            //
      }
    public bool insere(string Descricao, int Tipoinput, char
CampoOutro, char Disposicao)
    {
        StringBuilder TextoSql = new StringBuilder();
        SqlConnection ConexaoSQl = new SqlConnection(conStr);
```

```
        SqlCommand ComandoSQL = ConexaoSQl.CreateCommand();
        Int32 index;


        ConexaoSQl.Open();
        SqlTransaction TransacaoSql = ConexaoSQl.BeginTransaction();

        try
        {
            index = -1;
            ComandoSQL.Transaction = TransacaoSql;
            TextoSql.Remove(0, TextoSql.Length);
            TextoSql.Append("SELECT MAX(idPergunta) FROM Pergunta");
            ComandoSQL.CommandText = TextoSql.ToString();

            SqlDataReader OREAD = ComandoSQL.ExecuteReader();
            while (OREAD.Read())
            {
                if (OREAD.IsDBNull(0))
                {
                    index = -1;

                }
                else
                    index = OREAD.GetInt32(0);
            }
            OREAD.Close();


            if (index < 1)
                index = 1;
            else
                index++;

            TextoSql.Remove(0, TextoSql.Length);

            TextoSql.Append("INSERT INTO
Pergunta(idPergunta,DescPergunta,Tipoinput,Campooutro,Disposicao)");
            TextoSql.Append("
VALUES(@idPergunta,@DescPergunta,@Tipoinput,@Campooutro,@Diposicao)");
            ComandoSQL.CommandText = TextoSql.ToString();
            ComandoSQL.Parameters.Clear();
            ComandoSQL.Parameters.AddWithValue("idPergunta", index);
            ComandoSQL.Parameters.AddWithValue("DescPergunta",
Descricao);
            ComandoSQL.Parameters.AddWithValue("Tipoinput",
Tipoinput);
            ComandoSQL.Parameters.AddWithValue("Campooutro",
CampoOutro);
            ComandoSQL.Parameters.AddWithValue("Diposicao",
Disposicao);
            ComandoSQL.ExecuteNonQuery();
            TransacaoSql.Commit();
            return true;

        }
        catch (Exception Ex)
        {
            TransacaoSql.Rollback();
            return false;
```

```
        }
    }

    /// <summary>
    /// Associa um Dado Tipo de Resposta a uma dada Pergunta
    /// </summary>
    /// <param name="idPergunta"></param>
    /// <param name="idresposta"></param>
    /// <returns></returns>
    public bool AssociaResposta (int idPergunta,int idresposta)
        {
           StringBuilder TextoSql = new StringBuilder();
        SqlConnection ConexaoSQl = new SqlConnection(conStr);

        SqlCommand ComandoSQL = ConexaoSQl.CreateCommand();
        Int32 ordem;


        ConexaoSQl.Open();
        SqlTransaction TransacaoSql = ConexaoSQl.BeginTransaction();

          try{
            //TextoSql.Append("INSERT INTO
Pergunta_resposta(idPergunta,idResposta,"

             ordem = -1;
             ComandoSQL.Transaction = TransacaoSql;
             TextoSql.Remove(0, TextoSql.Length);
             TextoSql.Append("SELECT MAX(ordem) FROM
Pergunta_Resposta WHERE idPergunta=");
             TextoSql.Append(idPergunta);
             TextoSql.Append(" AND idTipoResposta=");
             TextoSql.Append(idresposta);

             ComandoSQL.CommandText = TextoSql.ToString();

             SqlDataReader OREAD = ComandoSQL.ExecuteReader();
             while (OREAD.Read())
             {
                 if (OREAD.IsDBNull(0))
                 {
                     ordem = 0;

                 }
                 else
                     ordem = OREAD.GetInt32(0);
             }
             OREAD.Close();
             ordem++;
             TextoSql.Remove(0, TextoSql.Length);
             TextoSql.Append("INSERT INTO
Pergunta_Resposta(idPergunta,idTipoResposta,ordem) VALUES (");
             TextoSql.Append("@idPergunta,@idTipoResposta,@ordem)");
             ComandoSQL.CommandText = TextoSql.ToString();
             ComandoSQL.Parameters.Clear();
             ComandoSQL.Parameters.AddWithValue("idPergunta",
idPergunta);
             ComandoSQL.Parameters.AddWithValue("idTipoResposta",
idresposta);
             ComandoSQL.Parameters.AddWithValue("ordem", ordem);
```

```
                ComandoSQL.ExecuteNonQuery();
                TransacaoSql.Commit();
                return true;


            }
            catch (Exception Ex)
            {
                TransacaoSql.Rollback();
                return false;

            }
        }
        public bool SobeOrdemRespAssoc()
        {
            return false;
        }
        public bool DesceOrdemRespAssoc()
        {
            return false;
        }


        public bool EstaAssociada(int idPergunta, int idTiporesposta)
        {
          Int32 contagem;
        SqlConnection ConexaoSQl = new SqlConnection(conStr);

            SqlCommand ComandoSQL = ConexaoSQl.CreateCommand();
            ConexaoSQl.Open();
            ComandoSQL.CommandText = "SELECT COUNT(*) FROM
PERGUNTA_RESPOSTA WHERE idPergunta="+idPergunta +" AND
idTipoResposta="+idTiporesposta ;
            contagem = (Int32)ComandoSQL.ExecuteScalar();
            if (contagem == 0)
                return false;
            else
                return true;


        }
        public DataTable DetalhesPergunta(int idPerg)
        {
            try
            {
                SqlConnection ConexaoSQl = new SqlConnection(conStr);
                //SqlCommand ComandoSQL = ConexaoSQl.CreateCommand();
                //ComandoSQL.CommandText =
                String StrCmd="SELECT
DescPergunta,Tipoinput,CampoOutro,Disposicao from Pergunta WHERE
idPergunta=" + idPerg.ToString();
                SqlDataAdapter SQLDA = new SqlDataAdapter(StrCmd,
ConexaoSQl);
                DataTable dTPergunta = new DataTable();
                SQLDA.Fill(dTPergunta);
                return dTPergunta;

            }
            catch (Exception Ex)
            {
                return null;
```

```
        }

    }
    public DataTable DetalhesRespostaAssoc(int idPerg)
    {
        try
        {
            SqlConnection ConexaoSQl = new SqlConnection(conStr);
            //SqlCommand ComandoSQL = ConexaoSQl.CreateCommand();
            //ComandoSQL.CommandText =
            String StrCmd = "SELECT
Descricao,TipoResposta.idTipoResposta from
TipoResposta,Pergunta_Resposta WHERE
TipoResposta.idTipoResposta=Pergunta_Resposta.idTipoResposta AND
Pergunta_Resposta.idPergunta=" + idPerg.ToString();
            SqlDataAdapter SQLDA = new SqlDataAdapter(StrCmd,
ConexaoSQl);
            DataTable dTPergunta = new DataTable();
            SQLDA.Fill(dTPergunta);
            return dTPergunta;

        }
        catch (Exception Ex)
        {
            return null;

        }
    }


}
```

Produto.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

using System.Web.Services;
using System.Web.Services.Protocols;

using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Data.Sql;

using System.Web.SessionState;
using System.Text;
using System.IO;
using System.Security.Cryptography;


/// <summary>
/// Summary description for Produto
```

```
/// </summary>
public class Produto
{
    private string OraconStr =
System.Configuration.ConfigurationManager.ConnectionStrings["ProjcrmCo
nnectionString"].ToString();

    //private int pIDEstrategia;
    private string pNome;
    private string pDescricao;
    private string pPrecoRevenda;
    private string pPrecoCampanha;
    private string pPrecoIVA;
    private string pOrigemProduto;
    private DateTime pValidade;
    private string pPromocoes;
    private char pEstado;
   //private int pIDRegConcorrencia;
   // private int pIDVenda;


      public Produto(){
      }

    public string Nome
       {
         get
         {
            return this.pNome;
         }
         set {
            this.pNome=value;
             }
      }

      public string Descricao
        {
          get
          {
            return this.pDescricao;
          }
          set
          {
             this.pDescricao=value;
          }
      }

      public string PrecoRevenda
        {
          get
          {
            return this.pPrecoRevenda;
          }
          set
          {
             this.pPrecoRevenda=value;
          }
      }

      public string PrecoCampanha
        {
```

```
    get
    {
      return this.pPrecoCampanha;
    }
    set
    {
       this.pPrecoCampanha=value;
    }
}

public string PrecoIVA
 {
    get
    {
      return this.pPrecoIVA;
    }
    set
    {
       this.pPrecoIVA=value;
    }
}

public string OrigemProduto
 {
    get
    {
      return this.pOrigemProduto;
    }
    set
    {
       this.pOrigemProduto=value;
    }
}

public DateTime Validade
 {
    get
    {
      return this.pValidade;
    }
    set
    {
       this.pValidade=value;
    }
 }

 public char Estado
 {
    get
    {
      return this.pEstado;
    }
    set
    {
       this.pEstado=value;
    }
 }

 public string Promocoes
 {
    get
```

```
            {
              return this.pPromocoes;
            }
            set
            {
               this.pPromocoes=value;
            }
          }

        public bool Insere()
        {
            SqlConnection OraConn = new SqlConnection();
            OraConn.ConnectionString = OraconStr;

            using (OraConn)
          {
              OraConn.Open();

              SqlCommand OraCmd = OraConn.CreateCommand();

              SqlTransaction OraTrans;
              StringBuilder OraQuertyStr = new StringBuilder();

              OraTrans = OraConn.BeginTransaction();
              OraCmd.Transaction = OraTrans;


              OraQuertyStr.Append("INSERT INTO
Produto(Nome,Descricao,PrecoRevenda,PrecoCampanha,PrecoIVA,OrigemProdu
to,Validade,Estado,Promocoes) ");
              OraQuertyStr.Append("
VALUES(@Nome,@Descricao,@PrecoRevenda,@PrecoCampanha,@PrecoIVA,@Origem
Produto,@Validade,@Estado,@Promocoes)");

              try
              {
                  OraCmd.Parameters.Clear();
                  OraCmd.Parameters.Add("NOME", SqlDbType.VarChar,
pNome.Length, "NOME").Value = pNome;
                  OraCmd.Parameters.Add("Descricao", SqlDbType.VarChar,
pDescricao.Length, "DESCRICAO").Value = pDescricao;
                  OraCmd.Parameters.Add("PrecoRevenda",
SqlDbType.VarChar, pPrecoRevenda.Length, "PRECOREVENDA").Value =
pPrecoRevenda;
                  OraCmd.Parameters.Add("PrecoCampanha",
SqlDbType.VarChar, pPrecoCampanha.Length, "PRECOCAMPANHA").Value =
pPrecoCampanha;
                  OraCmd.Parameters.Add("PrecoIVA", SqlDbType.VarChar,
pPrecoIVA.Length, "PRECOIVA").Value = pPrecoIVA;
                  OraCmd.Parameters.Add("OrigemProduto",
SqlDbType.VarChar, pOrigemProduto.Length, "ORIGEMPRODUTO").Value =
pOrigemProduto;
                  OraCmd.Parameters.Add("Validade", SqlDbType.DateTime ,
8 ,"VALIDADE").Value = Convert.ToDateTime(pValidade);
                  OraCmd.Parameters.Add("Estado", SqlDbType.Char, 1,
"ESTADO").Value = Convert.ToChar(pEstado);
                  OraCmd.Parameters.Add("Promocoes", SqlDbType.VarChar,
pPromocoes.Length, "PROMOCOES").Value = pPromocoes;


                  OraCmd.CommandText = OraQuertyStr.ToString();
```

```
                OraCmd.ExecuteNonQuery();
                OraTrans.Commit();
            }
            catch (Exception e)
            {
                OraTrans.Rollback();
                return false;
            }


        }
        return true;

    }

    public void Remove()
    {
        throw new System.NotImplementedException();
    }

    public void Lista()
    {
        throw new System.NotImplementedException();
    }
  }
```

Segmento.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Text;

/// <summary>
/// Summary description for Segmento
/// </summary>
public class Segmento
{
    private string conStr =
System.Configuration.ConfigurationManager.ConnectionStrings["ConexaoSq
l"].ConnectionString;

    private int pIDSegmento;
    private string pDescricao;


      public Segmento()
      {
      }

    public int IDSegmento
```

```
    {
        get
        {
            return pIDSegmento;
        }
        set
        {
            this.pIDSegmento = value;
        }
    }

    public string Descricao
    {
        get
        {
            return this.pDescricao;
        }
        set
        {
            this.pDescricao = value;
        }
    }

    public bool Insere(string Descricao)
    {
        int index = 0;
        SqlConnection ConexaoSql = new SqlConnection(conStr);
        StringBuilder TextoSql = new StringBuilder();
        SqlCommand ComandoSql = ConexaoSql.CreateCommand();
        ConexaoSql.Open();
        SqlTransaction TransacaoSQL = ConexaoSql.BeginTransaction();

        ComandoSql.Transaction = TransacaoSQL;
        try
        {
            TextoSql.Remove(0, TextoSql.Length);
            TextoSql.Append("INSERT INTO Segmento(Descricao)");
            TextoSql.Append(" ");
            TextoSql.Append("VALUES(@DESCRICAO)");
            ComandoSql.CommandText = TextoSql.ToString();
            ComandoSql.Parameters.Clear();
            ComandoSql.Parameters.AddWithValue("DESCRICAO",
pDescricao);
            ComandoSql.ExecuteNonQuery();
            TransacaoSQL.Commit();
            return true;
        }
        catch (Exception ex)
        {
            TransacaoSQL.Rollback();
            return false;
        }
    }



    public bool AssociaSegmento(int idSegmento, int idCliente)
    {
        StringBuilder TextoSql = new StringBuilder();
        SqlConnection ConexaoSQl = new SqlConnection(conStr);
```

```
        SqlCommand ComandoSQL = ConexaoSQl.CreateCommand();
       // Int32 ordem;


        ConexaoSQl.Open();
        SqlTransaction TransacaoSql = ConexaoSQl.BeginTransaction();

        try
        {
            //ordem = -1;
            ComandoSQL.Transaction = TransacaoSql;
            TextoSql.Remove(0, TextoSql.Length);
            TextoSql.Append("SELECT MAX(ordem) FROM Clientes_Segmento
WHERE idSegmento=");
            TextoSql.Append(idSegmento);
            TextoSql.Append(" AND idCliente=");
            TextoSql.Append(idCliente);

            ComandoSQL.CommandText = TextoSql.ToString();

           /* SqlDataReader OREAD = ComandoSQL.ExecuteReader();
            while (OREAD.Read())
            {
                if (OREAD.IsDBNull(0))
                {
                    ordem = 0;

                }
                else
                    ordem = OREAD.GetInt32(0);
            }*/

            //OREAD.Close();
            //ordem++;
            TextoSql.Remove(0, TextoSql.Length);
            TextoSql.Append("INSERT INTO
Clientes_Segmento(idSegmento,idCliente) VALUES (");
            TextoSql.Append("@idSegmento,@idCliente)");
            ComandoSQL.CommandText = TextoSql.ToString();
            ComandoSQL.Parameters.Clear();
            ComandoSQL.Parameters.AddWithValue("idSegmento",
idSegmento);
            ComandoSQL.Parameters.AddWithValue("idCliente",
idCliente);

            ComandoSQL.ExecuteNonQuery();
            TransacaoSql.Commit();
            return true;
        }
        catch (Exception Ex)
        {
            TransacaoSql.Rollback();
            return false;

        }
    }


    public bool EstaAssociada(int idSegmento, int idCliente)
    {
```

```
        Int32 contagem;
        SqlConnection ConexaoSQl = new SqlConnection(conStr);

        SqlCommand ComandoSQL = ConexaoSQl.CreateCommand();
        ConexaoSQl.Open();
        ComandoSQL.CommandText = "SELECT COUNT(*) FROM
CLIENTES_SEGMENTO WHERE idSegmento=" + idSegmento + " AND idCliente="
+ idCliente;
        contagem = (Int32)ComandoSQL.ExecuteScalar();
        if (contagem == 0)
            return false;
        else
            return true;
    }

}
```

SugReclamacoes.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Web.Services;
using System.Web.Services.Protocols;

using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Data.Sql;

using System.Web.SessionState;
using System.Text;
using System.IO;
using System.Security.Cryptography;

/// <summary>
/// Summary description for SugReclamacoes
/// </summary>

public class SugReclamacoes
{
    // private string OraConStr = "Server=127.0.0.1;Port=5432;User
Id=projcrm;Password=projcrm;Database=dbProjcrm;Encoding=UNICODE;";
    private string OraconStr =
System.Configuration.ConfigurationManager.ConnectionStrings["ProjcrmCo
nnectionString"].ToString();

    //private int pIDSugReclamacao;
    private string pTipo;
    //private string pAssunto;
    private string pDescricao;

    public SugReclamacoes()
    {
    }
```

```
/*public string Assunto
 {
    get
    {
        return this.pAssunto;
     }
    set
    {
        this.pAssunto = value;
    }
}*/

public string Tipo
{
    get
    {
        return this.pTipo;
    }
    set
    {
        this.pTipo = value;
    }
}

public string Descricao
{
    get
    {
        return this.pDescricao;
    }
    set
    {
        this.pDescricao = value;
    }
}



public bool Insere()
{
    //SqlConnection Oraconn = new SqlConnection(OraConStr);
    SqlConnection OraConn = new SqlConnection();
    OraConn.ConnectionString = OraconStr;

    using (OraConn)
    {
        OraConn.Open();

        SqlCommand OraCmd = OraConn.CreateCommand();

        SqlTransaction OraTrans;
        StringBuilder OraQuertyStr = new StringBuilder();

        OraTrans = OraConn.BeginTransaction();
        OraCmd.Transaction = OraTrans;



        // StringBuilder OraQuertyStr = new StringBuilder();
```

```
            OraQuertyStr.Append("INSERT INTO
SugReclamacao(Descricao,Tipo)");
            OraQuertyStr.Append(" VALUES(@Descricao,@Tipo)");


            try
            {
                OraCmd.Parameters.Clear();
                OraCmd.Parameters.Add("Descricao", SqlDbType.VarChar,
pDescricao.Length, "DESCRICAO").Value = pDescricao;
                //OraCmd.Parameters.Add("Assunto", SqlDbType.VarChar,
pAssunto.Length, "ASSUNTO").Value = pAssunto;
                OraCmd.Parameters.Add("Tipo", SqlDbType.VarChar, 12,
"TIPO").Value = pTipo;

                /*SqlParameter DataParameter = new SqlParameter();
                DataParameter.SqlDbType = SqlDbType.Timestamp;
                DataParameter.DbType = System.Data.DbType.DateTime;
                DataParameter.SourceColumn = "DATA";
                DataParameter.ParameterName = "Data";
                DataParameter.Value = pData;
                OraCmd.Parameters.Add(DataParameter);
                */
                //OraCmd.Parameters.Add("Email", SqlDbType.VarChar,
pEmail.Length, "EMAIL").Value = pEmail;

                OraCmd.CommandText = OraQuertyStr.ToString();
                OraCmd.ExecuteNonQuery();
                OraTrans.Commit();
            }
            catch (Exception e)
            {
                OraTrans.Rollback();
                return false;
            }


        }
        return true;

    }
    public void Remove()
    {
        throw new System.NotImplementedException();
    }

    public void Lista()
    {
        throw new System.NotImplementedException();
    }
}
```

TipoResposta.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
```

```
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Text;


/// <summary>
/// Summary description for TipoResposta
/// </summary>
public class TipoResposta
{
    private string conStr =
System.Configuration.ConfigurationManager.ConnectionStrings["ConexaoSq
l"].ConnectionString;

      public TipoResposta()
      {
            //
            // TODO: Add constructor logic here
            //
      }
    public bool AdicionaTipoResp(string Desc)
    {
        int index=0;
        SqlConnection ConexaoSql = new SqlConnection(conStr);
        StringBuilder TextoSql = new StringBuilder();
        SqlCommand ComandoSql = ConexaoSql.CreateCommand();
        ConexaoSql.Open();
        SqlTransaction TransacaoSQL = ConexaoSql.BeginTransaction();

        ComandoSql.Transaction = TransacaoSQL;
        try
        {
            TextoSql.Remove(0, TextoSql.Length);
            TextoSql.Append("SELECT MAX(idTipoResposta) from
TipoResposta");
            ComandoSql.CommandText = TextoSql.ToString();
           SqlDataReader Leitura= ComandoSql.ExecuteReader();

           while (Leitura.Read())
            {
                if (Leitura.IsDBNull(0))
                    index = 0;
                else
                    index = Leitura.GetInt32(0);
            }
            index++;
            Leitura.Close();
            TextoSql.Remove(0, TextoSql.Length);
            TextoSql.Append("INSERT INTO
TipoResposta(idTipoResposta,Descricao)");
            TextoSql.Append(" ");
            TextoSql.Append("VALUES(@IDTIPORESPOSTA,@DESCRICAO) ");
            ComandoSql.CommandText = TextoSql.ToString();
            ComandoSql.Parameters.Clear();
```

```
            ComandoSql.Parameters.AddWithValue("IDTIPORESPOSTA",
index);
            ComandoSql.Parameters.AddWithValue("DESCRICAO", Desc);
            ComandoSql.ExecuteNonQuery();
            TransacaoSQL.Commit();
            return true;

        }
        catch (Exception ex)
        {
            TransacaoSQL.Rollback();
            return false;
        }
    }

}
```

Util.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Security.Cryptography;

using System.Net.Mail; //Classe  que possiblita o uso de Email
using System.Net; //namespace que permite a programaçao que permite
programar com varios protocolos de rede;

/// <summary>
/// algumas funcoes uteis
/// </summary>
public class util
{
    private string OraconStr =
System.Configuration.ConfigurationManager.ConnectionStrings["ProjcrmCo
nnectionString"].ToString();

      public util()
      {
            //
            // TODO: Add constructor logic here
            //
      }


    public string EncStrMd5(string valor)
    {
        Byte[] ValorAux;
        Byte[] ByteHash;
        ValorAux = System.Text.Encoding.UTF8.GetBytes(valor);
        //byte[] vaux = new byte[valor.Length];

        //for (int i = 0; i < valor.Length; i++)
        //{
```

```
        //    vaux[i] = (byte)valor[i];
        //}

        MD5 Md5Serv = new MD5CryptoServiceProvider();


        //byte[] resultado = Md5Serv.ComputeHash(vaux);
        //int j = 0;
        string s = null;
        ByteHash = Md5Serv.ComputeHash(ValorAux);
        s = Convert.ToBase64String(ByteHash);
        //while (j < resultado.Length)
        //{
        //    if ((resultado[j].Equals(0x22)) ||
(resultado[j].Equals(0x60)))
        //    {
        //        s += Convert.ToString(Convert.ToChar(resultado[j] +
10));
        //    }
        //    else
        //    {

        //        s += Convert.ToString(Convert.ToChar(resultado[j]));
        //    }

        //    j++;
        //}
        return s;
    }

    //Gera uma String com um determinado numero de caracteres

    public string RandomizeStr(int nchars)
    {
        int i = 0;
        int val = 0;
        string s = null;
        Random rndGenerator = new Random();
        while (i < nchars)
        {
            val = rndGenerator.Next(126);
            if (val > 32 && val < 127)
            {
                s += Convert.ToString(Convert.ToChar(val));
                i++;
            }
        }
        return s;
    }

    public bool EnviaMail(string NomeDe, string Mailde, string
Assunto, string Texto, string NomePara, string MailPara)
    {
        MailAddress De = new MailAddress(Mailde, NomeDe);
        MailAddress Para = new MailAddress(MailPara, NomePara);
        MailMessage Mensagem = new MailMessage(De, Para);
        Mensagem.Subject = Assunto;
        Mensagem.Body = Texto;
        SmtpClient client = new SmtpClient("localhost");
        client.Credentials =
CredentialCache.DefaultNetworkCredentials;
```

```
        // Console.WriteLine("Sending an e-mail message to {0} and
{1}.", to.DisplayName, message.Bcc.ToString());
        try
        {
            client.Send(Mensagem);
        }
        catch (Exception e)
        {
            return false;
        }
        return true;
    }
}
```

AdminCRM.master.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class AdminCRM : System.Web.UI.MasterPage
{

    protected void lnkLogin_Click(object sender, ImageClickEventArgs
e)
    {
        if (Session["Email"] != null)
        {
            Session.Abandon();
            Response.Redirect("~/Home.aspx");
        }

    }
}
```

AlteraRegisto.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Security.Cryptography;

public partial class Registo : System.Web.UI.Page
{
```

```
    protected void ImGBAlterar_Click(object sender,
ImageClickEventArgs e)
    {
        string Password, RPassword;// Conparação das passwords

        Cliente CrmCliente = new Cliente();
        util CrmUtilidades = new util(); //chamamos a classe de
funcoes utilitarias
        CrmCliente.Nome = txtNome.Text;
        CrmCliente.Morada = txtMorada.Text;
        CrmCliente.Localidade = txtLocalidade.Text;
        CrmCliente.CodPostal = txtCodPostal.Text;
        CrmCliente.Idade = Convert.ToInt32(txtIdade.Text);

        if (rblSexo.SelectedIndex != -1)
            CrmCliente.Sexo = Convert.ToString(rblSexo.SelectedValue);
        else
            CrmCliente.Sexo = "";
        if (ddlEstCiv.SelectedIndex != -1)
            CrmCliente.EstCivil =
Convert.ToString(ddlEstCiv.SelectedValue);
        else
            CrmCliente.EstCivil = "";

        CrmCliente.NumFilhos = Convert.ToInt32(txtNumFilhos.Text);
        //CrmCliente.TipoCliente =
Convert.ToString(ddlTipoCliente.SelectedValue);
        CrmCliente.Curso = txtCurso.Text;
        CrmCliente.AnoCurso =
Convert.ToInt32(ddlAnoCurso.SelectedValue);
        CrmCliente.Actividade = txtActividade.Text;
        CrmCliente.Rendimentos = txtRendimento.Text;
        CrmCliente.NumTelefone = Convert.ToInt32(txtNumtelf.Text);
        CrmCliente.NumTelemovel = Convert.ToInt32(txtNumTelem.Text);
        CrmCliente.Contpreferido =
Convert.ToString(ddlContacto.SelectedValue);
        CrmCliente.FreqContacto = txtFrequencia.Text;
        CrmCliente.Email = txtEmail.Text;
        CrmCliente.NumContribuinte =
Convert.ToInt32(txtNumContribuinte.Text);
        CrmCliente.Administrador = false;


        Password = txtPassword.Text;
        RPassword = txtConfPassword.Text;
        if ((Password.CompareTo(RPassword) != 0))
            return;

        CrmCliente.Password =
CrmUtilidades.EncStrMd5(txtPassword.Text.ToString());

        if (Password.CompareTo(RPassword) == 0)
        {
            CrmCliente.Password = CrmUtilidades.EncStrMd5(Password);
        }
        else
        {
            return;
        }
```

```
        int Resultado=CrmCliente.Insere();
        if (Resultado == 2)
        {
            LblErro.Text = "Erro, ja existe um cliente registado com
esse email!!!";
            LblErro.Visible = true;
        }
        else
            LblErro.Visible = false;
    }
}
```

CriarFases.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class CriarFases : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }



    protected void BtInserirF_Click(object sender, EventArgs e)
    {
        CriarFase CrmCriarFase = new CriarFase();
        CrmCriarFase.Nome = TxtNomeF.Text;
        CrmCriarFase.Descricao = TxtDescricaoF.Text;
        CrmCriarFase.Objectivo = TxtObjectivosF.Text;
        CrmCriarFase.Insere();
    }
}
```

CriarInq.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class Criarinq : System.Web.UI.Page
{
```

```
    protected void Page_Load(object sender, EventArgs e)
    {

            //if (Session["Administrador"] == null)
            //{
            //    throw (new HttpException(403, "Nao Permitido"));
            //}
            //else
            //{
            //    bool Administrador = (bool)Session["Administrador"];
            //    if (!Administrador)
            //        throw (new HttpException(403, "Nao
Permitido"));

            //}
    }
    /// <summary>
    /// Cria um novo inquerito
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void btCriarInquerito_Click(object sender, EventArgs e)
    {
        Inqueritos Inquerito = new Inqueritos();
        Inquerito.Cria_inquerito(TxtDescInquerito.Text);
        GvInquerito.DataBind();
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        Inqueritos Inquerito = new Inqueritos();

Inquerito.InserereLinhaInquerito((int)GvInquerito.SelectedDataKey.Valu
e,(int)Convert.ToInt32(cmbPergunta.SelectedValue));
        GvPergInquerito.DataBind();

    }
    /// <summary>
    /// Quando A indice da gridview dos inqueritos mmyta  Faz refescar
as perguntas Associadas a
    /// esse inquerito
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void GvInquerito_SelectedIndexChanged(object sender,
EventArgs e)
    {
      // actualiza a griview  das linhas do inquerito
        String str=  "SELECT Idinquerito, LinhasInquerito.idPergunta,
ordem,Pergunta.DescPergunta FROM linhasInquerito,Pergunta";
        str+=" WHERE Pergunta.idPergunta=LinhasInquerito.idPergunta
AND Idinquerito=";
        str+=GvInquerito.SelectedDataKey.Value.ToString()+" ";
        str+="order by   ordem";
        SqlDsPErguntasInquerito.SelectCommand=str;
        SqlDsPErguntasInquerito.DataBind();
        GvPergInquerito.DataBind();

        //Sempre mudanmos  de inquerito tambem temo que verifica quais
sao as campanhas associadas e nao associadas a um
        //Dado inquerito
```

```
        //1º vamos vamos ao datasource para selecionar as  quais são
as campanhas associadas a um dado inquerito
        str = "select campanha.idcampanha as idCampanha,nomecampanha
from campanha,inquerito_campanha ";
        str+="where campanha.idcampanha=inquerito_campanha.idcampanha
and inquerito_campanha.idinquerito=";
        str += GvInquerito.SelectedDataKey.Value.ToString();
        SqlDsinqAssocCampanha.SelectCommand = str;
        SqlDsinqAssocCampanha.DataBind();

        //2º Vamos Verificar quais as campanhas que nao estão
associadas a um dado inquerito
        str = "select campanha.idcampanha as idCampanha,nomecampanha
from campanha,inquerito_campanha ";
        str += "where
not(campanha.idcampanha=inquerito_campanha.idcampanha  and
inquerito_campanha.idinquerito=";
        str += GvInquerito.SelectedDataKey.Value.ToString()+")";
        SQLdsnAssocCampanha.SelectCommand = str;

        SQLdsnAssocCampanha.DataBind();
    }
    //protected void BtVoltar_Click(object sender, ImageClickEventArgs
e)
    //{
    //    int npags = (int)Session["npaginas"];
    //    int curpag = (int)Session["curPag"];
    //    if (npags != 1 && curpag != 0)
    //    {
    //        curpag--;
    //        GvInquerito.PageIndex = curpag;
    //        GvInquerito.DataBind();
    //        if (curpag == 0)
    //            BtVoltar.Enabled = false;
    //        else
    //        {
    //            BtVoltar.Enabled = true;
    //            BtVoltar.ImageUrl = "./images/icPergunta.gif";
    //        }
    //        Session["curPag"] = curpag;
    //    }
    //}

    /// <summary>
    /// Associa   uma Dada  Campanha nao Associada
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void GvInqNAssocCampanha_RowCommand(object sender,
GridViewCommandEventArgs e)
    {
        if (e.CommandName == "Associar")
        {
            Inqueritos Inq = new Inqueritos();
            //Inq.AssocInqACampanha(0, 0);
          //  string str = e.CommandArgument.ToString();
            //Response.Write("dfdfsdfs->" + str);
            //GvInqNAssocCampanha.DataBind();
            int idCampanha
=Convert.ToInt32(GvInqNAssocCampanha.DataKeys[Convert.ToInt32(
e.CommandArgument)].Value);
```

```
            //Response.Write( );
             int idInquerito =
Convert.ToInt32(GvInquerito.SelectedDataKey.Value);
             Inq.AssocInqACampanha(idInquerito, idCampanha);
             SQLdsnAssocCampanha.DataBind();
             GvInqNAssocCampanha.DataBind();
             SqlDsinqAssocCampanha.DataBind();
             GvInqAssocCampanha.DataBind();
        }
    }
    protected void GvInqAssocCampanha_RowCommand(object sender,
GridViewCommandEventArgs e)
    {
        if (e.CommandName == "Desassoc")
        {
            Inqueritos Inq = new Inqueritos();
            //Inq.AssocInqACampanha(0, 0);
          //  string str = e.CommandArgument.ToString();
            //Response.Write("dfdfsdfs->" + str);
            //GvInqNAssocCampanha.DataBind();

            int idCampanha
=Convert.ToInt32(GvInqAssocCampanha.DataKeys[Convert.ToInt32(
e.CommandArgument)].Value);
            //Response.Write( );
             int idInquerito =
Convert.ToInt32(GvInquerito.SelectedDataKey.Value);
             Inq.DesacAssocInqACampanha(idInquerito, idCampanha);
             SQLdsnAssocCampanha.DataBind();
             GvInqNAssocCampanha.DataBind();
             SqlDsinqAssocCampanha.DataBind();
             GvInqAssocCampanha.DataBind();
        }
    }
}
```

CRM.master.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class CRM : System.Web.UI.MasterPage
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // Codigo Para adiconar Rollover aos Hyperlinks
        String jsMmouseout = "javaScript:MM_swapImgRestore();";

        String jsMmouseover="javaScript:MM_swapImage(";
        String strAux;
```

```
        strAux = jsMmouseover + "'" + lnkHome.ClientID +
"','','images/bthome_f2.gif',1);";
        // link Home
        lnkHome.Attributes.Add("onMouseOver", strAux);
        lnkHome.Attributes.Add("onMouseOut", jsMmouseout);


        strAux = jsMmouseover + "'" + LnkInqeritos.ClientID +
"','','images/btinqueritos_f2.gif',1);";

        LnkInqeritos.Attributes.Add("onMouseOver", strAux);
        LnkInqeritos.Attributes.Add("onMouseOut", jsMmouseout);

        strAux = jsMmouseover + "'" + lnkSobre.ClientID +
"','','images/btsobre_f2.gif',1);";

        lnkSobre.Attributes.Add("onMouseOver", strAux);
        lnkSobre.Attributes.Add("onMouseOut", jsMmouseout);

        strAux = jsMmouseover + "'" + LnkSugestoes.ClientID +
"','','images/btsugestoes_f2.gif',1);";

        LnkSugestoes.Attributes.Add("onMouseOver", strAux);
        LnkSugestoes.Attributes.Add("onMouseOut", jsMmouseout);
        if (Session["Email"] != null)
        {
            ImageButton ImB=(ImageButton) FindControl("lnkLogin");
            ImB.ImageUrl = "~/images/btLogout.gif";
            strAux = jsMmouseover + "'" + ImB.ClientID +
"','','images/btLogout_f2.gif',1);";
            ImB.Attributes["onMouseOver"] = strAux;
            ImB.Attributes["onMouseOut"] = jsMmouseout;


        }


    }
    protected void lnkLogin_Click(object sender, ImageClickEventArgs
e)
    {
        String jsMmouseout = "javaScript:MM_swapImgRestore();";

        String jsMmouseover = "javaScript:MM_swapImage(";
        String strAux;

        if (Session["Email"] != null)
        {
            Session.Abandon();
            Response.Redirect("~/Home.aspx");
            ImageButton ImB = (ImageButton)FindControl("lnkLogin");
            ImB.ImageUrl = "~/images/btLogin.gif";
            strAux = jsMmouseover + "'" + ImB.ClientID +
"','','images/btLogin_f2.gif',1);";
            ImB.Attributes["onMouseOver"] = strAux;
            ImB.Attributes["onMouseOut"] = jsMmouseout;
        }
    }
}
```

DefinirMeioComunicação.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Text;


public partial class DefinirMeioComunicacao : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void BtInserirMC_Click(object sender, EventArgs e)
    {

        MeioComunicacao CrmMeioComunicacao = new MeioComunicacao();
        //CrmMeioComunicacao.IDMeioComunicacao =
Convert.ToInt32(TxtIDMeioC.Text);
        CrmMeioComunicacao.Tipo = TxtTipoMC.Text;
        CrmMeioComunicacao.Insere(TxtTipoMC.Text);
    }
}
```

Erro.aspx.ccs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

using System.Text;

public partial class Erro : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        StringBuilder MsgErro=new StringBuilder();
        System.Exception Excepcao =Server.GetLastError();

        if (Excepcao is HttpException)
        {
            HttpException ChkExcepcao =  (HttpException)Excepcao;
            switch (ChkExcepcao.GetHttpCode())
```

```
            {
                case 403:
                    {

                        MsgErro.Append("Não Tem permissao Para
visualiar esta  Pagina");
                        ImgErro.ImageUrl = "~/images/Img_Proibido.gif";
                        break;
                    }
                case 404:
                    {
                        MsgErro.Append("A Pagina nao Existe");
                        ImgErro.ImageUrl = "~/images/Img_Perigo.gif";
                        break;
                    }
                case 408:
                    {
                        MsgErro.Append("Passou o Tempo de
Requesição");
                        ImgErro.ImageUrl = "~/images/Img_Perigo.gif";
                        break;
                    }
                case 500:
                    {
                        MsgErro.Append("O servidor nao Conseguir
satizafazer a sua Requesição");
                        ImgErro.ImageUrl = "~/images/Img_Perigo.gif";
                        break;
                    }
                default:
                    {
                        MsgErro.Append("O Servidor relatou um erro.");
                        ImgErro.ImageUrl = "~/images/Img_Perigo.gif";
                        break;
                    }


            }

        }
        else if (Excepcao != null)
        {
            // The exception was not an HttpException.
            MsgErro.Append(Excepcao.ToString());
            ImgErro.ImageUrl = "~/images/Img_errogeral.gif";

        }
        LblErro.Text = MsgErro.ToString();
        Server.ClearError();


    }
    protected void ImageButton1_Click(object sender,
ImageClickEventArgs e)
    {
       Response.Redirect("~/login.aspx");
    }
}
```

EsquecimePass.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.IO;


public partial class EsquecimePass : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void btEnviar_Click(object sender, EventArgs e)
    {
        string path = @"c:\";
        Cliente CRMCliente = new Cliente();
        util CRMUtilidades = new util();
        string novaPassword;
        novaPassword = CRMUtilidades.RandomizeStr(6);
        path += txtEmail.Text.ToString()+".txt";
       bool existe= CRMCliente.ExisteCliente(txtEmail.Text);
       if (existe)
       {
           int idcli = CRMCliente.GetIdCliente(txtEmail.Text);
           if (CRMCliente.SetPassword(idcli,
CRMUtilidades.EncStrMd5(novaPassword)))
           {
               lblMsg.Text = "A sua Nova password ja foi enviada para
o seu Email";
               using (StreamWriter Sr = new StreamWriter(path, true))
               {
                   string straux = "Nova Password:  " +
novaPassword.ToString()+"";
                   Sr.WriteLine(straux);
                   CRMUtilidades.EnviaMail("Administração",
"Admin@admin.pt", "Nova Password", "A sua Nova Password e <br/>" +
novaPassword.ToString(), txtEmail.Text, txtEmail.Text);

               }
           }
           else
           {
               lblMsg.Text = "Houve um Erro ao mudar a sua password";
           }
       }
       else
           lblMsg.Text = "Não Existe nenhum utilizador com Essa
password nos nossos Registos";

    }
}
```

EstrategiaC.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;


public partial class EstrategiaC : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void BtInserirE_Click(object sender, EventArgs e)
    {
        Estrategia CrmEstrategia = new Estrategia();
        CrmEstrategia.Descricao = TxtDescricaoE.Text;
        CrmEstrategia.Utilizada = ddlEstrUtil.Text;
        CrmEstrategia.Insere();
    }
}
```

HomeCliente.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class Home : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        int idCliente = new int();
        idCliente = Convert.ToInt32(Session["idCliente"]);
        Inqueritos inq = new Inqueritos();
        DataSet Dset = new DataSet();
        Dset =
inq.RetornaInqueritosNaoRespondidosPeloCliente(idCliente);

        GVListaInqueritos.DataSource = Dset.Tables[0];
        GVListaInqueritos.DataBind();
           }
}
```

Inquerito.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

using System.Collections.Generic;
using System.Data.Sql;

public partial class inquerito : System.Web.UI.Page
{

    protected void Page_Load(object sender, EventArgs e)
    {

    }
    private void Page_Init(object sender, EventArgs e)
    {
        CriaControlos();
    }
    private void CriaControlos()
    {
       Inqueritos Inq=new Inqueritos();
       Pergunta Perg = new Pergunta();
       String StrDescpergunta;
       int i = 0;
        int idInq =
Convert.ToInt32(this.Page.Request.QueryString.Get("Idinq"));
       if (idInq == 0)
           return;
       List<int> listaIdPerguntas = Inq.RetornaIdPErguntasInq(idInq);
        foreach (int idPergunta in listaIdPerguntas)
       {
           DataTable  TbDetalhePerg=Perg.DetalhesPergunta(idPergunta);


StrDescpergunta=TbDetalhePerg.Rows[0]["DescPergunta"].ToString();
          Label Lblperg = new Label();
          Lblperg.ID = "LblPerg" + i.ToString();
          Lblperg.Text = StrDescpergunta;
          this.PlaceInqueritoHolder.Controls.Add(Lblperg);
          this.PlaceInqueritoHolder.Controls.Add(new
LiteralControl("<br/>"));
          int Tipoinput =Convert.ToInt32(
TbDetalhePerg.Rows[0]["Tipoinput"].ToString());
          char orient =Convert.ToChar(
TbDetalhePerg.Rows[0]["Disposicao"].ToString());
          switch (Tipoinput)
          {
              case 1:
                  CriaRbControl(i, idPergunta, orient);
                  break;
```

```
                case 2:
                    CriaCheckControl(i, idPergunta, orient);
                    break;
                case 3:
                    CriaComboControl(i, idPergunta);
                    break;
                case 4:
                    CriaTxtControl(i, idPergunta);
                    break;

            }
            i++;
            bool Campooutro =
Convert.ToBoolean(TbDetalhePerg.Rows[0]["CampoOutro"].ToString());
            if (Campooutro == true && Tipoinput != 4)
            {
                CriaCampoOutro(i);
            }

        }


    }

    //Cria uma ChecoboxControl Group
    public void CriaCheckControl(int indice, int idPergunta,char
Orientacao)
    {
        CheckBoxList Cblist = new CheckBoxList();
        Cblist.ID = "Ctrl" + indice.ToString();
        if (Orientacao == 'V')
            Cblist.RepeatDirection = RepeatDirection.Vertical;
        else
            Cblist.RepeatDirection = RepeatDirection.Horizontal;

        Pergunta Perg = new Pergunta();
        DataTable DtResposta = Perg.DetalhesRespostaAssoc(idPergunta);

        foreach (DataRow dr in DtResposta.Rows)
        {
            Cblist.Items.Add(new ListItem(dr[0].ToString(),
dr[1].ToString()));
        }
        PlaceInqueritoHolder.Controls.Add(new LiteralControl("<span>
</span>"));
        PlaceInqueritoHolder.Controls.Add(Cblist);
        PlaceInqueritoHolder.Controls.Add(new
LiteralControl("<br/>"));
    }
    //Cria uma ComboBox
    public void CriaComboControl(int indice, int idPergunta)
    {
        DropDownList Dplist = new DropDownList();
        Dplist.ID = "Ctrl" + indice.ToString();
        Pergunta Perg = new Pergunta();
        DataTable DtResposta = Perg.DetalhesRespostaAssoc(idPergunta);

        foreach (DataRow dr in DtResposta.Rows)
        {
            Dplist.Items.Add(new
ListItem(dr[0].ToString(),dr[1].ToString()));
```

```
        }

        PlaceInqueritoHolder.Controls.Add(new LiteralControl("<span>
</span>"));
        PlaceInqueritoHolder.Controls.Add(Dplist);
        PlaceInqueritoHolder.Controls.Add(new
LiteralControl("<br/>"));
    }
    //Cria uma TextBox
    public void CriaTxtControl(int indice, int idPergunta)
    {
        TextBox TxtCtrl = new TextBox();
        TxtCtrl.ID = "Ctrl" + indice.ToString();
        TxtCtrl.Attributes["width"]="300px";
        //TxtCtrl.Width = 100;
        //TxtCtrl.Width.Type = UnitType.Pixel;

        PlaceInqueritoHolder.Controls.Add(new LiteralControl("<span>
</span>"));
        PlaceInqueritoHolder.Controls.Add(TxtCtrl);
        PlaceInqueritoHolder.Controls.Add(new
LiteralControl("<br/>"));
    }

    //Cria uma TextBox
    public void CriaCampoOutro(int indice)
    {
        TextBox TxtCtrl = new TextBox();
        TxtCtrl.ID = "CmpOutro" + indice.ToString();
        TxtCtrl.Attributes["width"] = "300px";
        //TxtCtrl.Width = 100;
        //TxtCtrl.Width.Type = UnitType.Pixel;

        PlaceInqueritoHolder.Controls.Add(new LiteralControl("<span>
</span>"));
        PlaceInqueritoHolder.Controls.Add(TxtCtrl);
        PlaceInqueritoHolder.Controls.Add(new
LiteralControl("<br/>"));
    }

    //Cria um Controlo RadioButton Group
    public void CriaRbControl(int indice, int idPergunta,char
Orientacao)
    {
        RadioButtonList RbList= new RadioButtonList();
        RbList.ID = "Ctrl" + indice.ToString();
        if (Orientacao== 'V')
            RbList.RepeatDirection = RepeatDirection.Vertical;
        else
            RbList.RepeatDirection = RepeatDirection.Horizontal;
        Pergunta Perg = new Pergunta();
        DataTable DtResposta = Perg.DetalhesRespostaAssoc(idPergunta);

        foreach (DataRow dr in DtResposta.Rows)
        {
            RbList.Items.Add(new ListItem(dr[0].ToString(),
dr[1].ToString()));
        }
        PlaceInqueritoHolder.Controls.Add(new LiteralControl("<span>
</span>"));
        PlaceInqueritoHolder.Controls.Add(RbList);
```

```
        PlaceInqueritoHolder.Controls.Add(new
LiteralControl("<br/>"));

    }

    /// <summary>
    /// Responde as perguntas Associadas Ao inquerito
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>


    protected void btEnviar_Click(object sender, EventArgs e)
    {

    }
}
```

Login.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Security.Cryptography;

public partial class Login : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }


    protected void BtEntrar_Click(object sender, ImageClickEventArgs
e)
    {
        Cliente CrmCliente = new Cliente();
        util CrmUtilidades=new util();
        string strHash,strHash1;
        int idcliente;

        if (!CrmCliente.ExisteCliente(txtEmail.Text))
        {
            lblMsg.Text = "LOGIN/PASSWORD Invalido(os)";
            lblMsg.Visible = true;
            return;

        }

        idcliente=CrmCliente.GetIdCliente(txtEmail.Text);
```

```
        strHash1 = CrmCliente.getPassword(idcliente);
        strHash = CrmUtilidades.EncStrMd5(txtPassword.Text);
        if (strHash1.Equals(strHash))
        {
            bool Admin = CrmCliente.VerificaAdmin(idcliente);
            Session.Add("Administrador", Admin);
            Session.Add("IdCliente", idcliente);
            Session.Add("Email", txtEmail.Text);


            if (Admin == true)
            {
                //this.Page.MasterPageFile = "AdminCRM.master";
                //ImageButton Lb =
(ImageButton)this.Page.Master.FindControl("lnkLogin");
                //Lb.ImageUrl = "~/images/Logout.gif";
                Response.Redirect("./HOME.aspx");

            }
            else
            {
               // this.Page.MasterPageFile="CRM.master";
                //Lb.ImageUrl = "~/images/Logout.gif";
                Response.Redirect("./HomeCliente.aspx");
               // Session.Abandon();
               // lblMsg.Text = "Este utizador nao Tem permissoes Para
aceder a Estas Paginas <br/> de Administraçao";

            }

        }
        else
        {
            lblMsg.Text = "LOGIN/PASSWORD Invalido(os)";
            lblMsg.Visible = true;
            return;
        }


        //int resp=CrmCliente.Consulta();
        //if (txtPassword.Text.CompareTo(CrmCliente.Password) == 0)
        //{
        //    Session.Add("USERNAME", CrmCliente.Nome);
        //    Session.Add("UserId", CrmCliente.IDCliente);
        //    LinkButton Lb =(LinkButton)
this.Page.Master.FindControl("HyperLinklogin");
        //    Lb.Text = "Logout";


        //    Response.Write("Sucesso");
        //}
        //else
        //{
        //    Response.Write("Insucesso");
        //}
    }

}
```

Perguntas.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Text;

public partial class Perguntas : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        StringBuilder Txt= new StringBuilder();

       // Txt.Append("JavaScript:Abre_div(");
        GvPerguntasRespostas.PagerSettings.Visible = true;


    }
    protected void btConfPerg_Click(object sender, EventArgs e)
    {
        bool Sucesso;
        Pergunta Perg = new Pergunta();
        Sucesso= Perg.insere(TxtPegunta.Text,
Convert.ToInt32(cmbTipoInput.SelectedValue),
           Convert.ToChar(RdlCampoOutro.SelectedValue),
            Convert.ToChar(RdlDisposicao.SelectedValue));
       if (Sucesso)
       {
           GvPerguntas.DataBind();
       }

    }
    protected void GridView1_SelectedIndexChanged(object sender,
EventArgs e)
    {
        SqlDsPerguntasRespostas.SelectCommand = "SELECT
idPergunta,Pergunta_resposta.idTipoResposta,Descricao FROM
Pergunta_resposta,TipoResposta where idpergunta=" +
Convert.ToString(GvPerguntas.SelectedValue);
        SqlDsPerguntasRespostas.SelectCommand += " And
Pergunta_resposta.idTipoResposta=TipoResposta.idTipoResposta";



    }
    protected void btAsscocPergunt_Click(object sender, EventArgs e)
    {
        Pergunta Perg = new Pergunta();
        int idPergunta, idResposta;
        idPergunta =
Convert.ToInt32(GvPerguntas.SelectedDataKey.Value.ToString());
        idResposta =Convert.ToInt32(CmdRespotaAssociar.SelectedValue);
```

```
        bool estaAssociado =
Perg.EstaAssociada(idPergunta,idResposta);
        if (estaAssociado)
            return;
        else
        {
            Perg.AssociaResposta(idPergunta, idResposta);
            SqlDsTipoResposta.DataBind();
            GvPerguntasRespostas.DataBind();
        }

    }
    protected void BtAdicionarTipoResposta_Click(object sender,
EventArgs e)
    {
        TipoResposta TipoResposta = new TipoResposta();
        TipoResposta.AdicionaTipoResp(txtTipoResposta.Text);
        txtTipoResposta.Text = "";
        SqlDsTipoResposta.DataBind();
        GvTipoResposta.DataBind();

    }
}
```

Produtos.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class Produtos : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void BtInserirP_Click(object sender, EventArgs e)
    {
        Produto CrmProduto = new Produto();
        CrmProduto.Nome = TxtNomeP.Text;
        CrmProduto.Descricao = TxtDescricaoP.Text;
        CrmProduto.PrecoRevenda = TxtPrecoRevenda.Text;
        CrmProduto.PrecoCampanha = TxtPrecoCamp.Text;
        CrmProduto.PrecoIVA = TxtPrecoIva.Text;
        CrmProduto.OrigemProduto = TxtOrigProd.Text;
        CrmProduto.Validade = Convert.ToDateTime(TxtValidadeP.Text);
        CrmProduto.Estado = Convert.ToChar(ddlEstadoP.SelectedValue);
        CrmProduto.Promocoes = TxtPromocoesP.Text;
        CrmProduto.Insere();
    }
}
```

RegConcorrencia.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class RegConcorrencia : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }


    protected void BtInserirRC_Click(object sender, EventArgs e)
    {
       /* RegConcorrencia CrmRegConcorrencia = new RegConcorrencia();
        CrmRegConcorrencia.IDProduto =
Convert.ToInt32(ddlIDProduto.Text);
        CrmRegConcorrencia.Observacoes = TxtObservacoesConc.Text;
        CrmRegConcorrencia.DataEstrategiaCampanha =
Convert.ToDateTime(TxtDataEstConc.Text);
        //CrmRegConcorrencia.Estrategiaconc = ddlEstrategiaCamp.Text;
        CrmRegConcorrencia.Insere();*/
    }
}
```

Registo.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Security.Cryptography;

public partial class Registo : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void ImGBEnviarPedido_Click(object sender,
ImageClickEventArgs e)
```

```
    {
        string Password, RPassword;// Conparação das passwords

        Cliente CrmCliente = new Cliente();
        util CrmUtilidades = new util(); //chamamos a classe de
funcoes utilitarias
        CrmCliente.Nome = txtNome.Text;
        CrmCliente.Morada = txtMorada.Text;
        CrmCliente.Localidade = txtLocalidade.Text;
        CrmCliente.CodPostal = txtCodPostal.Text;
        CrmCliente.Idade = Convert.ToInt32(txtIdade.Text);

        if (rblSexo.SelectedIndex != -1)
            CrmCliente.Sexo = Convert.ToString(rblSexo.SelectedValue);
        else
            CrmCliente.Sexo = "";
        if (ddlEstCiv.SelectedIndex != -1)
            CrmCliente.EstCivil =
Convert.ToString(ddlEstCiv.SelectedValue);
        else
            CrmCliente.EstCivil = "";

        CrmCliente.NumFilhos = Convert.ToInt32(txtNumFilhos.Text);
        //CrmCliente.TipoCliente =
Convert.ToString(ddlTipoCliente.SelectedValue);
        CrmCliente.Curso = txtCurso.Text;
        CrmCliente.AnoCurso =
Convert.ToInt32(ddlAnoCurso.SelectedValue);
        CrmCliente.Actividade = txtActividade.Text;
        CrmCliente.Rendimentos = txtRendimento.Text;
        CrmCliente.NumTelefone = Convert.ToInt32(txtNumtelf.Text);
        CrmCliente.NumTelemovel = Convert.ToInt32(txtNumTelem.Text);
        CrmCliente.Contpreferido =
Convert.ToString(ddlContacto.SelectedValue);
        CrmCliente.FreqContacto = txtFrequencia.Text;
        CrmCliente.Email = txtEmail.Text;
        CrmCliente.NumContribuinte =
Convert.ToInt32(txtNumContribuinte.Text);
        CrmCliente.Administrador = false;


        Password = txtPassword.Text;
        RPassword = txtConfPassword.Text;
        if ((Password.CompareTo(RPassword) != 0))
            return;

        CrmCliente.Password =
CrmUtilidades.EncStrMd5(txtPassword.Text.ToString());

        if (Password.CompareTo(RPassword) == 0)
        {
            CrmCliente.Password = CrmUtilidades.EncStrMd5(Password);
        }
        else
        {
            return;
        }


       int Resultado=CrmCliente.Insere();
       if (Resultado == 2)
```

```
        {
            LblErro.Text = "Erro, ja existe um cliente registado com
esse email!!!";
            LblErro.Visible = true;
        }
        else
            LblErro.Visible = false;

    }
}
```

Sugestões.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class Sugestoes : System.Web.UI.Page
{



    protected void BtLimpar_Click1(object sender, ImageClickEventArgs
e)
    {

    }


    protected void BtEnviar_Click(object sender, ImageClickEventArgs
e)
    {

        SugReclamacoes CrmSugReclamacoes = new SugReclamacoes();
        CrmSugReclamacoes.Descricao = txtDescricao.Text;
        CrmSugReclamacoes.Tipo = Convert.ToString(ddlTipo.Text);
        //CrmSugReclamacoes.Tipo =
Convert.ToChar(ddlTipo.SelectValue.ToString());
        //CrmSugRelamacoes.Assunto = txtAssunto.Text;
        CrmSugReclamacoes.Insere();
    }
}
```