**Relacção Nominal**

* Edmilson Guimarães José Domingos
* Fernando Manuel Lucala Kazuela

**Índice**

**Agradecimento**................................................................................Pág 03

**Dedicatória**....................................................................................... Pág04

**1. Introdução**.....................................................................................Pág05

**2. Desenvolvimento**...........................................................................Pág06

2.1 Aplicações web.

**3. Fundamentos teórico**

**3.1 Visual Studio 2012 (com plataforma Dot Net)**.....Pág07

3.1.1 Classes e Erança..................................................Pág08

3.1.2 Diagrama de Classes**.**.................................Pág09

**3.2 Folhas de Estilo (CSS).**

**3.2.1 Tipos de Folha de Estilo**

3.2.1.1 Folha de Estilo Externa

3.2.1.2 Folha de Estilo Interna

3.2.1.3 Folha de Estilo In-line

**3.3 Java Script (JS).**

**3.4 Aplicação em três camadas em Visual Studio C#.**

3.4.1Vantagen

3.4.2Desvantagen

**3.5 SQL Server (Management Studio).**

3.4.1 Comandos em SQL Server

3.4.1.1 Comandos LDD ou DDL

3.4.1.2Comandos DML

3.4.2Como criar uma Base de Dados

3.4.1.1 Sintaxe

3.4.1 Procedimentos

**3.6 Sistema de Gestão de Banco de Dados (SGBD).**

3.5.1 Arquitectura de um SGBD

3.5.1.1 Entidades

3.5.1.2 Relacionamento

3.5.1.3 Gráu de Relacionamento

3.5.1.4 Cardinalidade

3.5.1.5 Diagrama de entidades e Relacionamento

**4. Objectivo**

4.1 Objectivo Geral.

4.2 Objectivo Específico.

**5. Análise**

5.1 Definições de problemas.

5.2 Levantamentos de Requêsitos

5.3 Soluções proposta.

**6. Funcionamento do sistema**

6.1 Actores do sistema

6.2 Diagrama da aplicação

6.3 Diagrama das Classes

6.4 Diagrama do Banco de Dados

**Conclusão**

**Bibliografia**

**Agradecimento**

Em primeiro lugar, queremos agradecer a Deus por ter nos dado a vida e, de nos ter acompanhado durante o ano lectivo.

Agradecemos aos nossos encarregados de educação pela coragem e apoio que nos ofereceram como encarregado, também não podemos nos esquecer do apoio que os professores (orientadores) que nos ofereceram como docente e aos colegas que nos ajudaram na elaboração do projecto.

**Dedicatória**

De acordo com a filosofia de ensino, dedicamos este projecto ao Colégio Árvore da Felicidade por nos oferecerem o calor humano como estudante, e por terem nos dado a oportunidade de fazer parte do núcleo familiar que é COLÉGIO ÁRVORE DA FELICIDADE. Esperamos desta forma que se dediquem e se empenhem cada vez mais contribuindo assim para o crescimento da sociedade angolana.

E também, dedicamos este projecto à nossa formação.

1. **Introdução**

As Tecnologias de Informação e Comunicação (TIC) tem evoluido muito, devido a este factor evolutivo muitos estudantes de tecnologia têm ajudado neste processo.

De acordo com os conhecimentos adquiridos ao longo do nosso percurso academico, decidimos propôr à nossa aplicação Web para a instituição (Árvore da Felicidade).

Aplicação: é um conjunto de instruções a serem seguidas e executadas num dispositivo (computadores, telefone etc).

Aplicação Web: é o termo utilizado para designar, de forma geral, sistemas de [informática](https://pt.wikipedia.org/wiki/Inform%C3%A1tica" \o "Informática) projetados para utilização através de um [navegador](https://pt.wikipedia.org/wiki/Navegador" \o "Navegador), na [internet](https://pt.wikipedia.org/wiki/Internet" \o "Internet) ou em redes privadas ( [Intranet](https://pt.wikipedia.org/wiki/Intranet" \o "Intranet) ).

Para desenvolver-mos a nossa aplicação, pretendemos usar diversas ferramentas que, nos possam ajudar na sua execução.

* Visual Studio 2012 (com plataforma Dot Net).
* SQL Server 2012 (Menagement Studio)

**2.1 Aplicação web**

Trata-se de um conjunto de programas que é executado em um [servidor](https://pt.wikipedia.org/wiki/Servidor" \o "Servidor) de [HTTP](https://pt.wikipedia.org/wiki/HTTP" \o "HTTP) (Web Host). O desenvolvimento da tecnologia web está relacionado, entre outros fatores, a necessidade de simplificar a atualização e manutenção mantendo o código-fonte em um mesmo local, de onde ele é acessado pelos diferentes usuários.

Pode-se definir uma aplicação web como uma aplicação de software que utiliza a web, através de um [browser](https://pt.wikipedia.org/wiki/Browser" \o "Browser), como ambiente de execução.

Uma Aplicação web também é definida em tudo que se é processado em algum servidor, exemplo: quando você entra em um e-commerce a página que você acessa antes de vir até seu navegador é processada em um computador ligado a internet que retorna o processamento das regras de negócio nele contido. Por isso se chama aplicação e não simplesmente site web.

**3.1 Visual Studio 2012 (com plataforma Dot Net).**

Ferramenta de desenvolvimento para o Frame Work. Permite rápido desenvolvimento de Software por trabalho com recursos de arrastar e soltar; Help Online.

Podemos comparar esta ferramenta como um sistema operativo para software, cabe a ele controlar o acesso à memória, discos, instância e a destrução de objecto, variaveis, formulários, gráficos, etc.

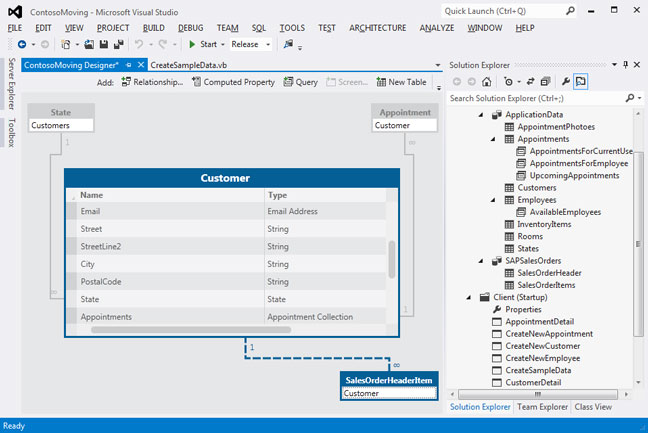
Não é segredo que uma nova era de aplicativos modernos chegou. Com dispositivos conectados e serviços de nuvem, temos oportunidades maiores e melhores como nunca antes, Programadores independentes podem se conectar de qualquer lugar, arquitectar aplicativos brilhantes e torná-los disponíveis para milhões de usuários.

Equipes grandes e ágeis podem dar uma vantagem significativa para os seus negócios e quanto mais rápido eles executarem, maior pode ser a vantagem.

É por isso que o Visual Studio 2012 é um dos maiores lançamentos até agora, Ele vem com o propósito de ajudar os programadores a prosperar em um ambiente em que as ideias são valorizadas e a velocidade é essencial. Vamos observar algumas das maneiras em que ele pode ajudar você a transformar ideias em aplicativos rapidamente.

**Uma nova aparência e uma nova sensação**

*Figura 1*



**3.1.1 Classes e Erança**

**Classes**

Não entraremos em detalhes agora sobre recursos das classes, mas precisamos entender a estrutura de um projeto, e para isso é essencial entender o conceito das classes.

Classes são objetos que podem ou não se tornar componentes.

Uma classe sempre utiliza um escopo, neste caso public, a palavra chave ***class*** e o nome definido. Todas as classes iniciam e terminam com os delimitadores de chave. Na plataforma DOT NET 2.0 foi introduzido o recurso parcial class que permite dividir uma mesma classe em mais que um arquivo.

Qual a vantagem disto?

Imagine uma classe que monta um formulário com dezenas ou centenas de linhas com definições de variáveis, controles, etc. Podemos dividir a porção de código das definições iniciais e gráficas em um arquivo e os eventos e métodos no segundo arquivo, usando como declaração.

**A herança**

Permite programar a funcionalidade à sua classe de tomar emprestado o resto da estrutura e comportamento de classes de nível mais alto.

Pensemos em uma classe pessoa.

Esta classe define os comportamentos e atributos de uma pessoa; E existem atributos que serão comuns a todas as pessoas.

O nome e a idade são atributos comuns a qualquer pessoa. Já uma pessoa e nome **Walter** possui atributos que somente ela possui: ***Engiero*** por exemplo.

A definição formal para herança seria:

- Herança é um mecanismo que permite que características comuns a diversas classes sejam agrupadas em uma classe base, ou superclasse. A partir de uma classe base, outras classes podem ser especificadas.   
- Cada classe derivada ou subclasse apresenta as características (estrutura e métodos) da classe base e acrescenta a elas o que for definido de particularidade para ela.

Veja o diagrama abaixo para entender melhor este conceito de herança:

*Figura 2*

**Pessoa/Classe primária Aluno/Classe Secundária**

Walter

Pessoa

**3.2 Folhas de Estilo (CSS)**

CSS significa “Cascading Style Sheetes” (Folhas de Estilo em Cascata). Não seria ótimo fazer layouts na sua página sem ter que alterar o HTML toda vez que quisesse mudar? Nesta introdução, vamos ter uma breve introdução ao CSS para ter uma ideia de como ela é prática. Podemos dizer que o CSS é a melhor metade do HTML. Codificando, não há melhor parceria: HTML é responsável pelo trabalho pesado (a estrutura), enquanto CSS dá o toque de elegância (layout).

CSS pode ser adicionado com uso do atributo style. Por exemplo, você pode definir o tipo e o tamanho da fonte em um parágrafo:

**<p style="font-size:20px;" >**Este parágrafo em tamanho de fonte igual a **20px</p> <p style="font-family:courier;" >**Este parágrafo em fonte Courier**</p> <p style="font-size:16px; font-family:cambria" >**Este parágrafo em fonte Cambria e tamanho **20px</p>**

Será renderizado no navegador assim:

Este parágrafo em tamanho de fonte igual a **20px,** Este parágrafo em fonte Courier, Este parágrafo em fonte Cambria e tamanho 16px.

No exemplo acima usamos o atributo style para definir o tipo de fonte usado (com a propriedade font-family) e o tamanho da fonte (com a propriedade font-size). Notar que no último parágrafo do exemplo definimos tanto o tipo como o tamanho da fonte separado por um ponto e vírgula.

Uma das funcionalidades mais inteligentes das CSS é a possibilidade de controlar o layout de um arquivo central. Em lugar de se usar o atributo style em cada tag, você pode dizer ao navegador como deve ser o layout de todos os textos em uma página:

**<html> <head> <title>**Minha primeira página CSS**</title> <style type="text/css"> h1 {font-size: 30px; font-family: arial} h2 {font-size: 15px; font-family: courier} p {font-size: 8px; font-family: times new roman} </style> </head> <body> <h1>**Minha primeira página CSS**</h1> <h2>**Bem vindo à minha primeira página CSS**</h2> <p>**Aqui você verá como funciona CSS**</p> </body> </html>**

No exemplo acima, inserimos as CSS na seção head do documento, assim ela se aplica à página inteira. Para fazer isto use a tag <style type="text/css"> que informa ao navegador que você está digitando CSS. No exemplo, todos os cabeçalhos da página serão em fonte Arial e tamanho 30px, Todos os subtítulos serão em fonte Courier tamanho 15. E, todos os textos dos parágrafos serão em fonte Times New Roman tamanho **8.**

Outra opção é a de digitar as CSS em um documento separado. Com as CSS em um documento separado você pode gerenciar o layout de muitas páginas ao mesmo tempo. Muito inteligente, pois você pode mudar de uma só vez o tipo ou o tamanho da fonte de todo o site, quer ele tenha centenas ou milhares de páginas. Nós não nos aprofundaremos em CSS agora, mas você pode aprender tudo, no futuro, em nosso tutorial CSS.

**Como funcionam as folhas de estilo (CSS) ?**

Nesta lição, você aprenderá a desenvolver sua primeira folha de estilos. Você verá o básico sobre o modelo **CSS** e, que código é necessário para usar **CSS** em um documento **HTML**. Muitas das propriedades usadas em Cascading Style Sheets (CSS) são semelhantes àquelas do **HTML**. Se você está acostumado a usar **HTML** para layout irá reconhecer muitos dos códigos que usaremos. A sintaxe básica das CSS Suponha que desejamos uma cor de fundo vermelha para a página web.

Usando HTML podemos fazer assim:

<body bgcolor="#FF0000">

**3.2.1 Tipos de Folha de Estilo**

As CSS permite-nos criar folhas de estilo de três (3) modos, possibilitando o programador a se adaptar a que ele achar melhor, ou seja, mais fácil. Assim sendo, existem três tipos de folha de estilo:

* Folha de Estilo Externa
* Folha de Estilo Interna
* Folha de Estilo In-line

**Folha de Estílo Externa**

Este tipo de folha de estílo “Externa” permite o programa a criar um ficheiro externo e pode ser responsável pela formatação de todas as páginas do site. Cada página que pretende usar essa folha de estilo deve invoca-la no cabeçalho do documento. O ficheiro que define a folha de estilo tem a extenção.

**Ex:**

**<Html>**

**<Head>**

**<link rel = “style sheet” type = “text/css” href = “minhafolhaDeEstilo.css”>**

**</head>**

**<body>**

**<p> Exemplo com uma folha de estilo externa</p>**

**</body>**

**</HTML>**

**Folha de Estilo Interna**

Nas folhas de estilo Internas a formatação é definida no cabeçalho do proprio ficheiro, delimitada pelas marcas **<style> </style>.**

**Exemplo:**

**<Html>**

**<Head>**

**<style type =“text/css”>**

**P {margin-left:100 px}**

**< Body >**

**{background-image: url(“imagens/mac.jpg”)}**

**</style>**

**</head>**

**<body>**

**<p>Texto deslocado para a direita</p>**

**<p>Este também</p>**

**<p>Graças a folha de estilo Interna</p>**

**</body>**

**</HTML>**

**Folha de Estilo In-line**

Nas folhas de formatação em linha, a formatação é definida na propria linha do elemento HTML a formatar. Aplica-se o atributo style á marca que se pretende formatar, afectando as propriedades pretendidas.

Ao contrário das folhas de estilo Interna e Externa, as folhas de estilo Inline não oferecem a vantagem de formatar de uma só vez vários elementos.

**Ex:**

**<html>**

**<head>**

**</head>**

**<p style = “color: blue; margin-left: 20 px”>**

**Isto é um paragrafo formatado pela folha de estilo em Linha</p>**

**</body>**

**</html>**

**3.3 Java Script (JS)**

É uma linguagem de programação derivada da Linguagem C, Uma das principais caracristicas do Java Script é a interação com o navegado “Browser”.

Java Script: é uma linguagem de programação da familia da linguagem C, criada especeficamente para Internet pela **“Net Scap”** em 1995, com o proposito de permitir uma interactividade superior a que se consegue apenas com HTML.

Uma das principais caracteristicas do Java Script é a forma de como interaje com o navegador, podendo aceder os seus objectivos.

Assim, para programar em Java Script é conveniente conhecer o DOM (Document Object model ) de cada navegador.

**Vantagengens do Java Script**

* Semelhança com a linguagem C.
* De facil aprendizagem.
* Não exige nenhum recurso ao lado do servidor.
* Segurança porque não permite ler ou escrever no computador do visitante. (exceptuam-se os Cookies) do site nem no servidor etc.

**Desvantagem**

* O codigo fica exposto no seio do HTML podendo ser facilmente cipiado.
* Não é uma linguagem de uma impossive1.

**3.4 Aplicação em três camadas em Visual Studio C#**

Camada, em inglês *layer*, significa camada lógica, que é a organização do código de uma aplicação.

Camada, em inglês *tier*, significa camada física, que é onde o código irá ser executado (hardware, servidores, etc..).

O desenvolvimento em camadas é sobre divisão de camadas lógicas, então apesar de de*layer* e *tier* possuirem o mesmo significado em português, devemos estar sempre atento ao contexto. As camadas do modelo em três camadas são:

1. **Camada de apresentação (*User interface/Presentation*):** É a camada onde iremos exibir informações e coletar dados do usuário, em ASP.NET são as páginas \*.aspx.
2. **Camada de negócio (*Business logic layer*):** É onde escrevemos a lógica e as regras de negócio da aplicação. Ela é a camada intermediária entre o usuário e o back-end.
3. **Camada de acesso a dados (*Data access layer*):** Camada responsavel pela comunicação direta da aplicação com o back-end, realizando transações, como inclusão e consulta a dados.

**Como criar um projecto em três camadas no Visual Studio 2012?**

A melhor forma para criar uma aplicação em camadas em ASP.NET é criar um projeto diferente para cada camada, agrupando-os em uma *solution*. Vamos criar três projetos:

Um projeto do tipo *Web/ASP.NET web application,* chamado **Camada\_Visão**, que será a camada de apresentação.

Um projeto do tipo *Windows/Class library,* chamado **Camada\_Modelo**, que será a camada de negócio.

Um projeto do tipo *Windows/Class library,* chamado **Camada\_Acesso\_aos\_Dados**, que será a camada de acesso a dados.

O primeiro projecto criado “ **Camada\_Visão** ” sera responsavel pela vinculação com com a nossa Camada de Modelo. É na Camada \_Visão onde o usuario vai poder efectuar as suas actividades.

Exemplo:

**Preencher um formulário.**

Na Camada\_Modelo, terei todas as minhas regras de negócio, ou seja, é, nesta camada, onde terei todas as classes que teram acesso ao meu Banco de Dados.

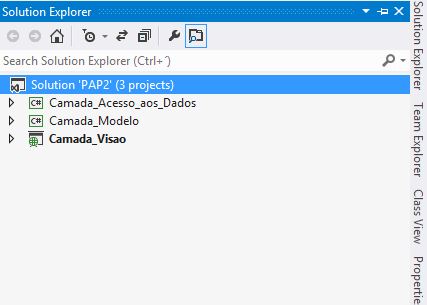
Exemplo:

**Uma classe Estudante**

A figura a seguir ilustra de como ficara a nossa solução:

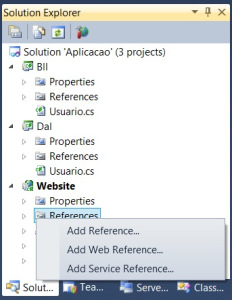
A nossa solução ficará assim:

*Figura 3*



Para adicionar a referência em seu projeto, vá à janela *Solution Explorer,* clique com o botão direito no item *References* do projeto, e selecione a opção Add Reference… *Figura 4*

*Figura 4*



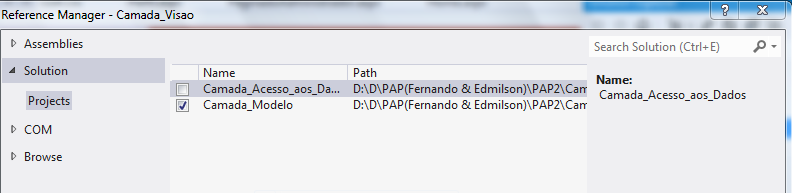
Na janela que irá abrir, selecione o projeto que você deseja adicionar como referência na guia Projects.

Após criar as camadas, é necessário adicionar as referências entre elas para que haja a comunicação:

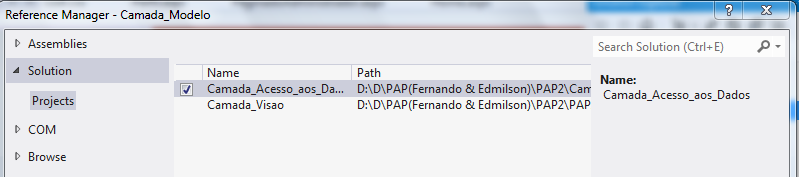
Projeto Camada\_Visao adicionar a referência do projeto Camada\_Mdelo (Camada\_Visao -> Camada\_Modelo). Figura5

Projeto Camada\_Modelo  adicionar a referência do projeto Camada\_Acesso\_aos\_Dados  (Camada\_Modelo   -> Camada\_Acesso\_aos\_Dados  ). *Figura 6*

*Figura 5*

****

*Figura 6*

**

Feito este processo, agora você já pode trabalhar com a sua aplicação em três camadas no Visual Studio 2012.

**3.4.1Vantagen**

A utilização de camadas em um projecto no Visual Studio permite:

* Organização do projecto.
* Facilidade na leitura do projecto.
* Melhor funcionamento.
* Separar as responsabilidades.

**3.4.2 Desvantagen**

Suscetíveis a danos causados por códigos com erros ou mal intencionados. e difícil manutenção.

**3.5 SQL Server.**

O **MS SQL Server** é um SGBD - sistema gerenciador de Banco de dados relacional desenvolvido pela Microsoft. Foi criado em parceria com a Sybase em 1988 inicialmente para a plataforma OS/2. Esta parceria durou até 1994, com o lançamento da versão para Windows NT e desde então a Microsoft mantém a manutenção do produto.

Como um Banco de dados, é um produto de software cuja principal função é a de armazenar e recuperar dados solicitados por outras aplicações de software seja aqueles no mesmo computador ou aqueles em execução em outro computador através de uma rede (incluindo a Internet).

Há pelo menos uma dúzia de diferentes edições do Microsoft SQL Server destinado a públicos diferentes e para diferentes cargas de trabalho (variando de pequenas aplicações que armazenam e recuperam dados no mesmo computador, a milhões de usuários e computadores que acessam grandes quantidades de dados a partir da Internet ao mesmo tempo). Suas linguagens de consulta primárias são T-SQL e ANSI SQL.

A SQL é uma linguagem de computador para trabalho com conjuntos de fatos e com a relação entre eles. Programas de banco de dados relacionais, como o Microsoft Office Access, usam a SQL para trabalhar com dados. Ao contrário de várias linguagens de computador, ela não é difícil de ler e compreender, mesmo para um iniciante. Como muitas linguagens de computador, a SQL é um padrão internacional reconhecido por órgãos de padrões.

Use-a para descrever conjuntos de dados que possam ajudá-lo a encontrar respostas para perguntas. Ao usar a SQL, você precisa aplicar a sintaxe correta. O termo "sintaxe" corresponde ao conjunto de regras com base nas quais os elementos de uma linguagem são corretamente combinados. A sintaxe SQL se baseia na sintaxe do inglês e utiliza muitos dos mesmos elementos da sintaxe.

**3.4.1 Comandos em SQL Server**

* Comandos LDD ou DDL (**Linguagem de Definição de Dados**)
* Comandos DML (**Linguagem de Manipulação de Dados**)

**3.4.1.1 Linguagem de Definição de Dados (**LDD ou DDL,do Inglês Data Definition Language**).**

É uma linguagem de computador usada para a definição de estruturação de dados.

**Comandos LDD ou DDL**

* **Create** – cria um objecto (uma tabela, por exemplo) dentro da base de dados.
* **Drop**- apaga um objecto do banco de dados.

**3.4.1.2 Linguagem de manipulação de dados (**ou DML, do Inglês Data Manipuletion language**).**

É uma família de computador utilizadas para a recuperação, inclusão, remoção e modificação de informções em um banco de dados.

**Comandos DML**

* **Insert-** é usado para inserir um registo à uma tabela existente.
* **Select-** é utilizado para selecionar dados ou valores em uma tabela.
* **UpDate-** serve para mudar os valores de dados em uma ou mais linhas da tabela existente.
* **Delete-** permite remover linhas existentes de uma tabela.

***Sintaxe do comando insert***

**INSERT INTO nome\_tabela (lista-de-campos)**

**VALUES (lista\_dados)**

**--OU**

**INSERT INTO nome\_tabela  VALUES (lista\_dados)**

**Onde:**

**Nome\_tabela:** nome da tabela no qual será inserido os dados.

**Lista-de-campos:** nome das colunas que receberão os valores.

**Lista-dados:** valores que serão inseridos na tabela. Estes campos devem estar na mesma ordem descrita em lista-de-campos, todos separados por vírgula.

Nota: Se for utilizado um comando SELECT o mesmo deve retornar a mesma quantidade de colunas com os mesmos tipos de dados especificados em lista-de-campos.

***Exemplo do comando insert:***

INSERT INTO EMPREGADOS(CODIGO, NOME, SALARIO, SECAO)

VALUES(1, "HELBERT CARVALHO", 1.500, 1)

INSERT INTO EMPREGADOS VALUES(1,"HELBERT CARVALHO",1500,1)

***Exemplo do comando select:***

SELECT CODIGO,NOME,SALARIO, SECAO

FROM EMPREGADOS\_FILIAL

WHERE DEPARTAMENTO = 2

***Exemplo do comando update:***

UPDATE nome\_tabela

SET CAMPO = "novo\_valor"

WHERE CONDIÇÃO

**Onde:**

Nome\_tabela: nome da tabela que será modificada

Campo: campo que terá seu valor alterado

Novo\_valor: valor que substituirá o antigo dado cadastrado em campo

Where: Se não for informado, a tabela intera será atualizada

Condição: regra que impõe condição para execução do comando

**Exemplos de uso do comando update**

UPDATE DEPARTAMENTO

SET SALARIO = 1000

WHERE CODIGODEP = 1

**Sintaxe do comando delete**

DELETE FROM nome\_tabela

WHERE condição

**Onde:**

Nome\_tabela: nome da tabela que será modificada

Where: cláusula que impõe uma condição sobre a execução do comando

***Exemplo de uso do comando delete***

DELETE FROM EMPREGADOS

WHERE CODIGO = 125

**3.4.2 Como criar uma Base de Dados**

A sequência lógica é a seguinte:

1. Criar o Banco de dados
2. Criar a Tabela
3. Criar os campos
4. Definir as propriedades de cada campo
5. Anexar todos os campos a coleção Table
6. Criar os índices
7. Definir as propriedades de cada índice
8. Criar os campos necessários a cada índice
9. Definir as propriedades dos campos
10. Anexar os campos necessários a coleção field do índice
11. Anexar todos os índices a coleção Table
12. Anexar a tabela a coleção Tables do banco de dados

**3.4.1.1 *Sintaxe* *para criar uma Base de Dados***

CREATE new database <nome\_da\_base\_de\_dados>

Use <nome\_da\_base\_de\_dados>

Comandos....

**3.4.1 Procedimentos ou Stored procedure**

Procedimento armazenado ou Stored procedure é uma coleção de comandos em SQL para dispensamento de Banco de Dados. Encapsula terefas reespectivas, aceita parâmetros de entrada e retorna um valor de status (para indicar aceitação ou falha na execução).

O procedimento armazenado pode reduzir o tráfego na rede, melhorar a desempenho, criar mecanismo de segurança, etc.

***Sintaxe de um procedimento:***

Create Procedure < nome\_do\_procedimento>

Parâmetros

As

Begin

Comandos do procedimento

End

**Onde:**

<nome\_do\_procedimento>- indica o nome do procedimento criado.

Parâmetros- valores passados pelo usuário os nomes dos parâmetros deverão ser procedidos do simbolo arrroba (@).

**3.4.1 Sistema de Gestão de Banco de Dados (SGBD)**

Um Sistema Gerenciador de Banco de Dados (SGBD) é constituído por um conjunto de dados associados a um conjunto de programas para acesso a esses dados. O conjunto de dados, comumente chamado de banco deda dos,contéminformaçõessobreumaempresaemparticular.

O principal objetivo de um SGBD é proporcinar um ambiente tanto conveniente quanto eﬁciente para a recuperação e armazenamento das informações do banco de dados. SGBDs sãoprojetados para gerir grandes volumes de informações. Devem possuir mecanismos para deﬁnição e manipulação de dados, além de prover compartilhamento e segurança dos mesmos.

**3.4.2.1 Arquitetura de um SGBD**

A arquitetura de um SGBD se divide em três níveis:

* **Nível Interno ou Físico**
* **Nível Conceitual ou Lógico**
* **Nível Externo**

Nível Interno ou Físico: é o mais próximo do meio de armazenamento físico, ou seja, é aquele que se ocupa do modo como os dados são ﬁsicamente armazenados.

Nível Conceitual ou Lógico: Descreve quais dados estão armazenados no banco de dados e quais os inter- relacionamentosentre eles. Estenívelé utilizadopelosadministradores.

Nível Externo: é o mais próximo dos usuários, ou seja, é aquele que se ocupa do modo como os dados são vistos por usuários individuais.

A seguir iremos flisar alguns exemplos para melhor compreensão dos determinados níveis:

No nível Conceitual: O banco de dados contém informações relativas a um tipo de entidade, chamado EMPREGADO, Cada empregado contém um NÚMERO\_EMPREGADO, um NÚMERO\_DEPARTAMENTO e um SALÁRIO.

No nível interno: Os empregados são representados por um tipo de registro armazenado, denominado EMP\_ARMAZENADO, com 20 bytes de comprimento. Contém 4 campos: um preﬁxo de 6 bytes mais 3 campos de informação de empregado. Além disso, os registros são indexados sobre o campo EMP.

No nível externo: O usuário SECRETÁRIA tem uma visão na qual cada empregado tem 2 campos: Nome e Salário. Já o usuário CONTADOR tem uma visão que cada empregado temos campos Nome e Salário.

**Diagrama dos níveis**

*Figura 6*

**VISÃO**

**VISÃO**

**VISÃO**

**Físico**

**3.4.1 Entidades**

Deﬁne-se entidade como aquele objeto que existe no mundo real com uma identiﬁcação distinta e com um signiﬁcado próprio. São as “coisas” que existem no negócio, ou ainda, descrevemos negócio. Se alguma “coisa”, existente no negócio nos proporciona algum interesse em mantermos dados (informaçòes armazenadas sobre ele), ista a caracteriza como uma Entidade do Negócio. Alguns exemplos de entidades:

O FUNCIONÁRIO João;

O VEICULO Corsa;

A ALUNA Maria;

O CLIENTE Pedro;

O ALIMENTO A323....

Entidades de um mesmo tipo são agrupadas em Classes de Entidade. Assim, a classe de entidades FUNCIONÁRIOS é o conjunto de todas as instâncias de funcionários. Neste texto, classes de entidades estão impressas em letra maiúscula. Cada ocorrência de um funcionário dentro da classe FUNCIONÁRIO é denominado Instânciade Entidade.

**3.4.2 Relacionamento**

Nenhuma informação armazenada no Banco de Dados existe isoladamente.

Todos os elementos pertencentes ao mundo real modelado de alguma forma estão associados a outros elementos. Normalmente essas associações representam acções físicas ou alguma forma de dependência entre os elementos envolvidos.

Neste contexto, definindo relacionamento é nada mais do que a associação entre as Entidades.

Agora que já temos as deﬁnições de Entidades e de Relacionamento, vamos aprender como encontrá-los em um problema: Cliente faz emprestimos.

Desta frase, o que é Entidade e o que é relacionamento?

Pode-se dizer que os SUBSTANTIVOS são as Entidades e os VERBOS são os Relacioanamentos.

Sendo assim tem-se:

Entidades: Cliente e Emprestimo.

Relacionamento: Faz

*Figura 7*

Cliente

Emprestimo

Faz

**3.4.3 Gráu Relacionamento**

Um Relacionamento pode envolver duas ou mais Entidades. O Grau do Relacionamento é o número de Entidades envolvidas.

Desta forma podem-se categorizar os tipos de relacionamento em:

Relacionamentos Binários Relacionamento que envolve duas Entidades.

Relacionamentos Ternários Relacionamento que envolve duas Entidades.

Colocar quando usar o relacionamento ternario.

**3.4.4 Cardinalidade dos Relacionamentos**

A quantidade de Entidades envolvidas em um Relacionamento é determinada pela Cardinalidade do Tipo de Relacionamento, ou seja, pode-se estabelecer a quantidade mínima e máxima de Entidades envolvidas com cada Entidade relacionada.

A Cardinalidade Mínima que determina a quantidade mínima de Entidades relacionadas é determinada pelo número representativo, ou seja, 0 (zero), 1,2,... N(muitos). A Cardinalidade Máxima que determina a quantidade máxima de Entidades relacionadas é determinada pelo número representativo, ou seja, 0 (zero), 1,2,... N(muitos).

**3. Objectivo Geral**

Minimizar os esforços dos alunos, professores e dos membros dirigentes desta instituição (Colégio Árvore da Felicidade).

De maneira que o trabalho seje mais simples e eficais para todos os consumidores.

**3.1 Objectivo Especifico**

Pretende-se desenvolver um sistema que faça Inscrição On-line para uma instituição (Colégio Árvore da felicidade).

**5. Analise**

**5.1 Definições do problema.**

As tecnologias de informação e comunicação têm evoluido muito. Portanto, pretendemos criar esta aplicação para que as coisas sejam mais privadas, informativas para que as pessoas estejem informadas de forma segura e restrita.

**5.2 Soluções propostas.**

Para melhorar este problema decidimos propor o nossa aplicação web. Para que o trabalho seje feito de forma restrita:

Divulgação de resultado de forma restrita.

Inscrção de restrita.

**6. Funcionamento do sistema**

A aplicação permitira:

* Inscrição de Alunos
* Enviar C.V de Professores.
* Verificação e alocação de Pautas.
* Compras de Uniforme.
* Área de Galeria.
* Acesso a informações sobre a instituição.

**6.1 Actores do sistema**

Alunos: usurios do sistema que possuem uma conta. Estes têm a possibilidade:

- Ver Resultado (Pauta);

- Compra de Uniforme;

Professores (Director de Turma): usurios do sistema que possuem uma conta. Estes têm a possibilidade:

-Enviar as notas dos alunos;

Candidatos a cargo como Docente: estes teram a possibilidade de poder enviar os seus Curriculo Vitae.

Administrador: usuário que possui super previlegios. Este pode executar as seguintes actividades:

- Remover dados de um aluno;

- Remover dados de um professor;

- Actualizar dados de um Aluno;

- Actualizar dados de um Professor;

-Visualizar dados de um aluno;

-Visualizar dados de um professor;

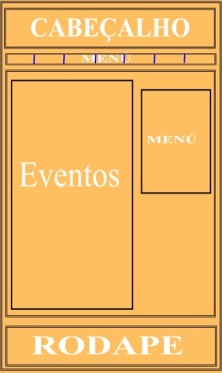
Usuários Passivos: internautas que não possuem contas no site têm apenas a possibilidade de visualizar as informções da instituição.

**6.2 Diagrama da aplicação**

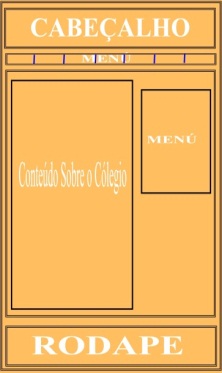
*Figura 1*



Home



Eventos



Sobre

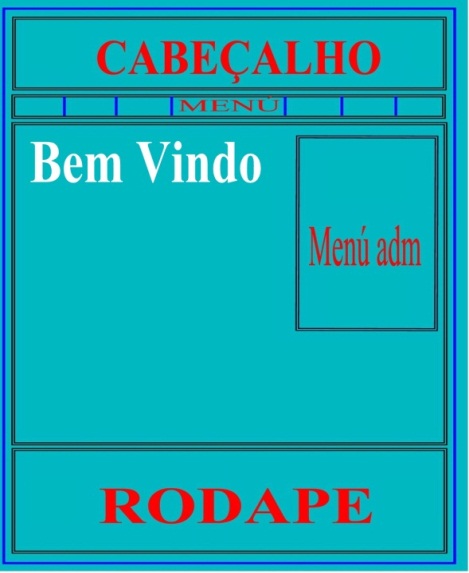
Cólegio



Galeria



Notícia



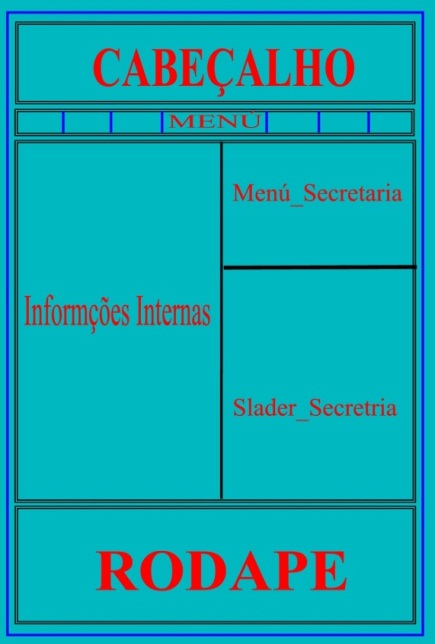
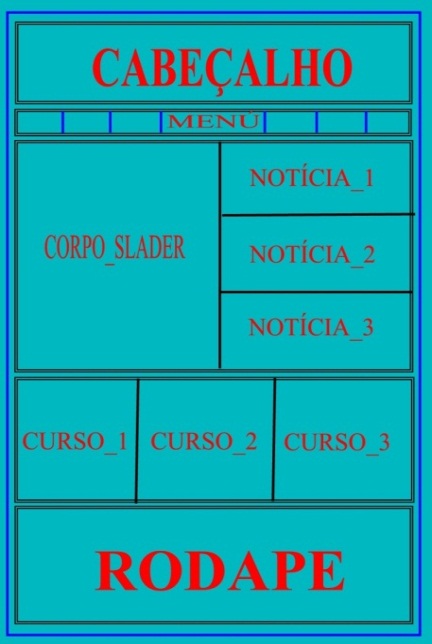
Secretaria

Adm

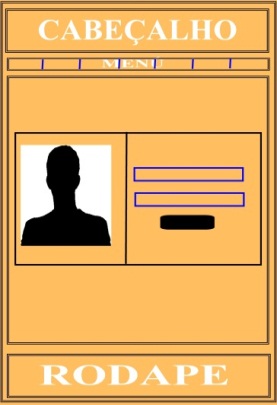
*Figura* 2

Secretaria

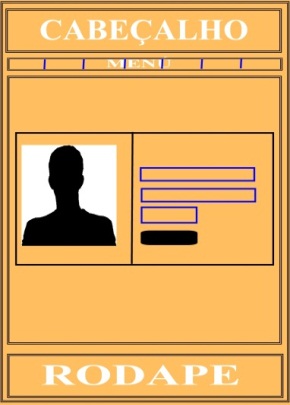
Home



Comprar\_Uni

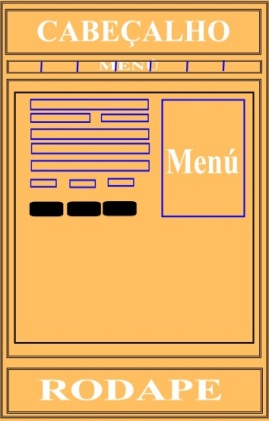


Login\_Aluno

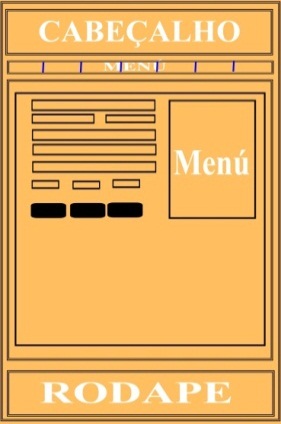


Login\_Prof

Pg\_Adm

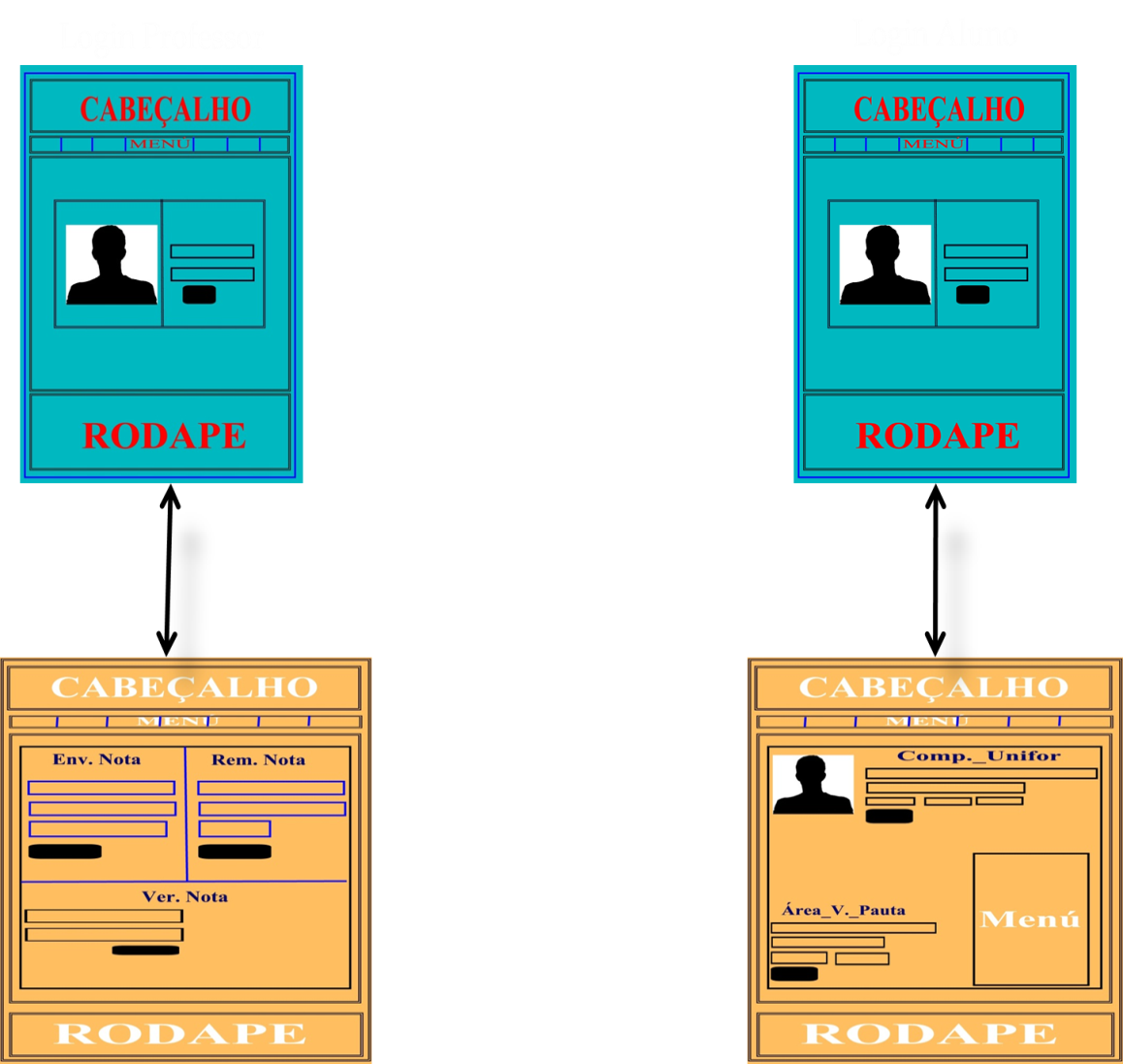


A\_Aluno



A\_Prof

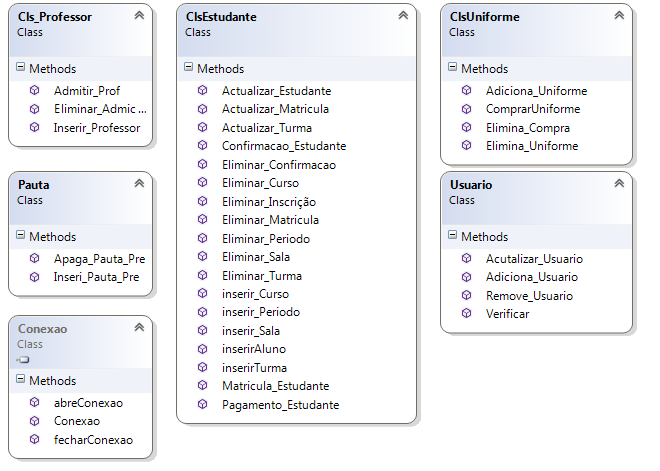
*Figura* 3

****

Login\_aluno

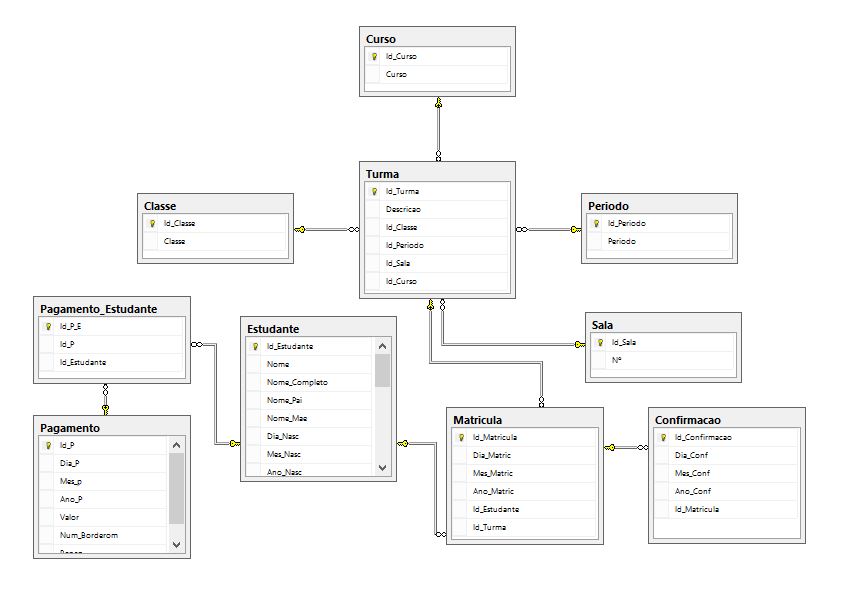
Login\_prof

**6.3 Diagrama das classes**

****

**6.4 Diagrama do Banco de Dados**

6.4.1 Parte I

****

Nesta parte temos as seguintes Entidades ou Tabelas:

1-Estudante

2-Matricula

3-Turma

4-Curso

5-Periodo

6-Classe

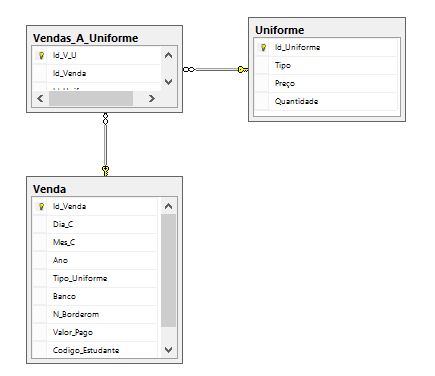
7-Sala

8-Confirmação

9-Pagamento

10-Pagamento\_as\_Estudante

6.4.2 Parte II



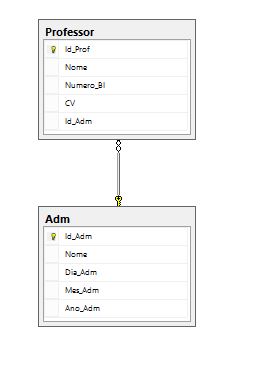
Nesta parte temos as seguintes Entidades ou Tabelas:

1-Uniforme

2-Vendas

3-Vendas\_A\_Uniforme

6.4.3 Parte III

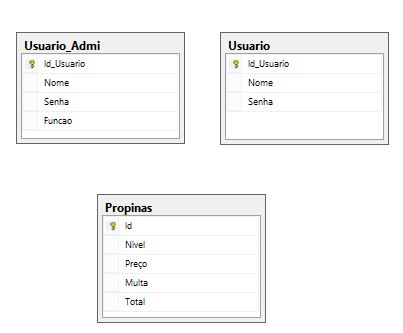
****

Nesta parte temos as seguintes Entidades ou Tabelas:

1-Profesor

2-Adm

6.4.4 Parte IV



Nesta parte temos as seguintes Entidades ou Tabelas:

1-Usuario\_Admi

2-Usuario

3-Propinas

**Conclusão**

Portanto, com a nossa aplicação, pretendemos atingir um grande patamar e divulgar o nome da instituição no mundo todo atravez da internet.

Esperamos conseguir apoio para a divulgação desta aplicação, pois é bastante útil e moderna.

**Bibliografia**

<http://www.linhadecodigo.com.br/artigo/2975/comandos-basicos-em-sql-insert-update-delete-e-select.aspx>.

<http://office.microsoft.com/pt-br/access-help/access-sql-conceitos-basicos-vocabulario-e-sintaxe-HA010256402.aspx#BM1>.

<http://www.microsoft.com/visualstudio/ptb/whats-new>.

<http://www.apostilando.com/download.php?cod=3124&categoria=Banco%20de%20Dados>.