LINGUAGEM DE PROGRAMAÇÃO

Orientada a objeto

Roteiro da aula

- Desenvolvimento em camadas

Camadas:

- As camadas são dividas em:
  - Camada de Apresentação;
  - Camada de Regra de Negócio;
  - Camada de Acesso a Dados.
- Esse isolamento possibilita a **reutilização** do código e facilita a **manutenção** e o **aperfeiçoamento** do código.
- O **desenvolvimento** também é facilitado, pois há uma clara decomposição de funcionalidade, o que permite aos desenvolvedores concentrarem-se em diferentes partes da aplicação durante a implementação.

Camada de Apresentação

- Oferece conteúdo estático e conteúdo dinâmico personalizado, que pode ser apresentado nos mais variados formatos disponíveis, como HTML, Windows Forms ou XML.
- É implementada com uso dos componentes visuais da plataforma .NET, tanto para ambientes Web quanto para ambiente desktop.
- O objetivo é permitir ao desenvolvedor obter produtividade através da facilidade do desenvolvimento interface.
- As classes dessa camada utilizam os serviços oferecidos pela camada de negócios.

Camada de Regra de Negócio

- É responsável por implementar a lógica de negócio da aplicação. Nela estão todas as classes inerentes ao domínio da aplicação.

Camada de Acesso a Dados

- É responsável pela persistência e acesso aos dados da aplicação.
- Ela isola o resto da aplicação do meio de armazenamento usado de maneira que, se o meio de armazenamento for trocado, apenas as classes desta camada precisarão ser modificadas ou substituídas.

Objetivo da arquitetura em camadas

- **Modularidade**: dividir a aplicação em módulos tão independentes quanto possível;
- **Manutenibilidade**: reduzir o custo de manutenção da aplicação;
- **Extensibilidade**: permitir que novas funcionalidades sejam adicionadas sem grande impacto nas já existentes;
- **Reusabilidade**: permitir que classes e componentes sejam reusados em outros módulos da mesma aplicação ou em outras aplicações.

Vantagens

- O desenvolvimento da aplicação é feita de forma rápida, simples, fácil e com um custo muito baixo;
- O acesso à fonte de dados esta separada em seu próprio componente de forma que o código da aplicação de frente não possui código SQL embutido;
- A informação da conexão é mantida somente no serviço XML, minimizando a manutenção do cliente;
- A camada de acesso a dados pode ser atualizada em um único local centralizado.
- Você não precisa distribuir componentes ao cliente quando houver alterações nesta camada.

Desvantagens

- Manter a aplicação pode se tornar uma tarefa muito complexa, pois qualquer alteração em um nome de campo ou de tabela terá que ser feita em diversas partes da sua aplicação;

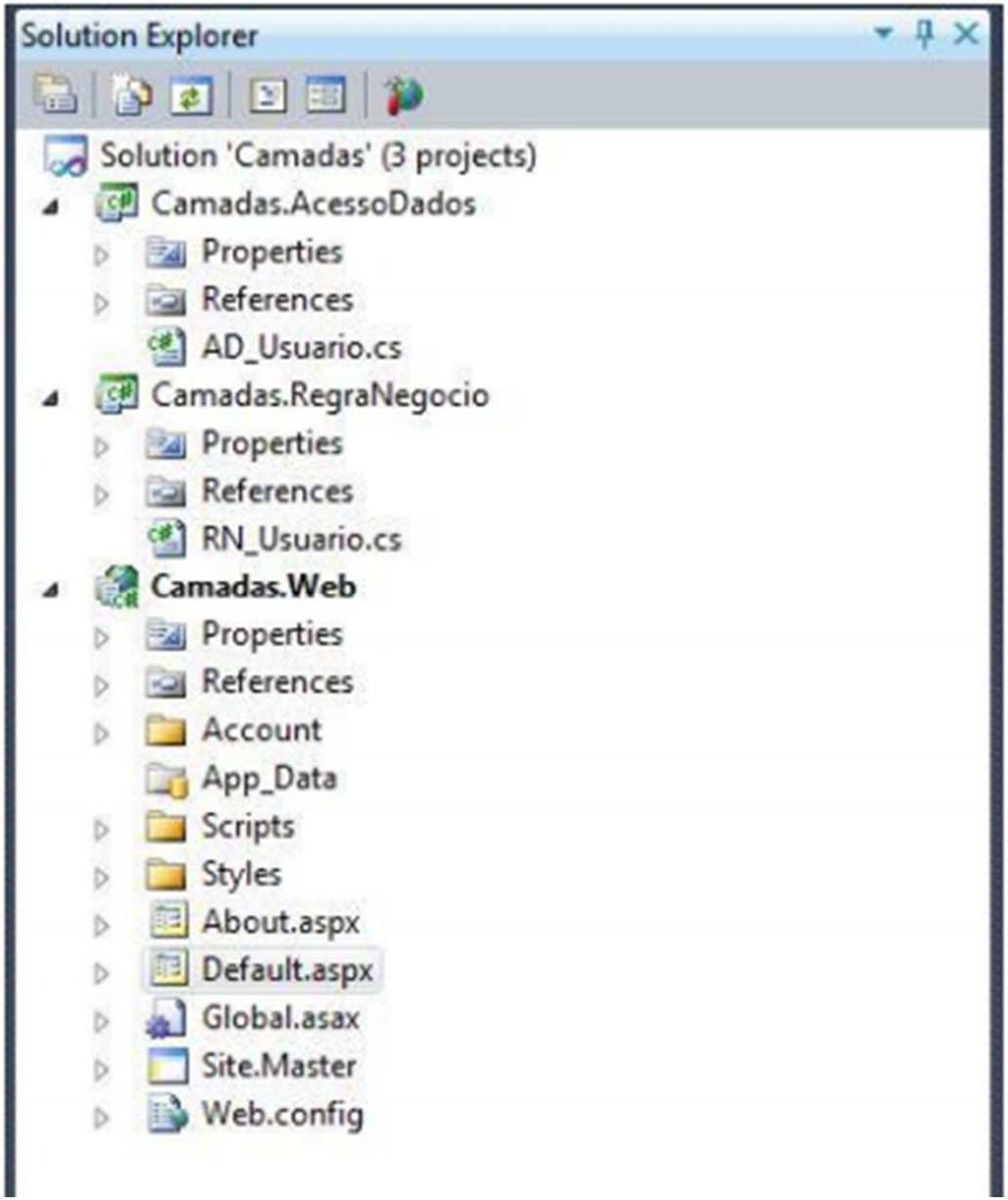
Projeto:

- Temos três Projetos dentro da Solution:

- **Camadas.Web**: camada de apresentação;

- **Camadas.RegraNegocio** camada de regra de negócio;

- **Camadas.AcessoDados** camada de acesso a dados.



Projeto

- A **camada de apresentação** só terá arquivos relacionados às telas (telas em si, arquivos de estilo, arquivos javascript, etc);
- A **camada de regra** de negócio só terá classes com toda a lógica implementada nelas
- A **camada de acesso a dados** terá todas as procedures e/ou classes chamando procedures referentes ao projeto.

Ciclo de execução

Camada de Apresentação

Camada de Regra de Negócio

Camada de Acesso a Dados

→ Na camada de **apresentação** você desenvolve as telas e faz tratamentos básicos como campos obrigatórios

→ chama a camada de **regra de negócio**, onde você desenvolve toda a regra existente naquela tela específica

→ chama a camada de **acesso a dados**, onde você desenvolve todo o acesso ao banco de dados, retornando valores que são necessários para trabalhar

Ciclo de execução

Camada de Apresentação

Camada de Regra de Negócio

Camada de Acesso a Dados

→ retorna valor obtido para camada de **regra de negócio**

→ devolve valor para a camada de apresentação, exibindo através de listagem ou formulários, conforme necessidade

Pratica

- Crie um projeto chamado ExemploCamadas
- Importante → Salve o projeto
- Agora adicione no Solution, um novo Projeto tipo *Class* chamado Modelos.
- Renomeie a classe do projeto Modelo para InfoCliente,

na qual será implementada as propriedades de cada campo da tabela criada no banco de dados.

- Mais dois novos Projetos tipo *Class* :

AcessaDados – renomeie classe para ClientesDAL

RegraNegocio – renomeie classe para ClientesBLL

- Renomeie o primeiro projeto para InterfaceUser

Observações

- Crie projetos do tipo class.
- Todas as classes criadas devem ser publicas para serem acessadas em qualquer projeto
- Ao renomear cuidado para não apagar a extensão “cs”

Relacionamento entre camadas

- Agora com todos os projetos criados, antes de qualquer coisa é importante fazer o relacionamento entre as camadas, referenciando as DLLs dos projetos.
- Para adicionar as referência, basta você clicar com o botão direito em cima da pasta “References” e clicar em **“Add Reference – Projects - ”.**
- Referencias:

  A camada de AcessaDados → Modelos.

  RegraNegócio →Modelos e AcessaDados

  InterfaceUsuario→ Modelos, AcessaDados, RegraNegócio,

  Modelos não precisa de nenhuma referencia.
- Nas classes adicione “using” com as referencias.

Modelos

- Para criar as propriedades na classe InfoCliente você terá que ter um Banco de Dados já pronto. (BD Pizzaria)

- Crie os Get/Set de todos os campos

Teste de acesso ao Banco de Dados

- Para trabalhar com o MySQL é necessário usar um arquivo DLL . Copie o arquivo na pasta do Projeto e faça as referências quando necessário.

using MySql.Data.MySqlClient;

- Cria um novo Formulário na interfaceUser para testar a conexão com o Banco de Dados com 2 métodos: Conectar e Desconectar

AcessaDados

- Como estamos trabalhando em camadas vamos criar uma classe Conexão para conectar e desconectar do Banco de Dados na Camada AcessaDados. Copie os métodos do formulário nesta classe.

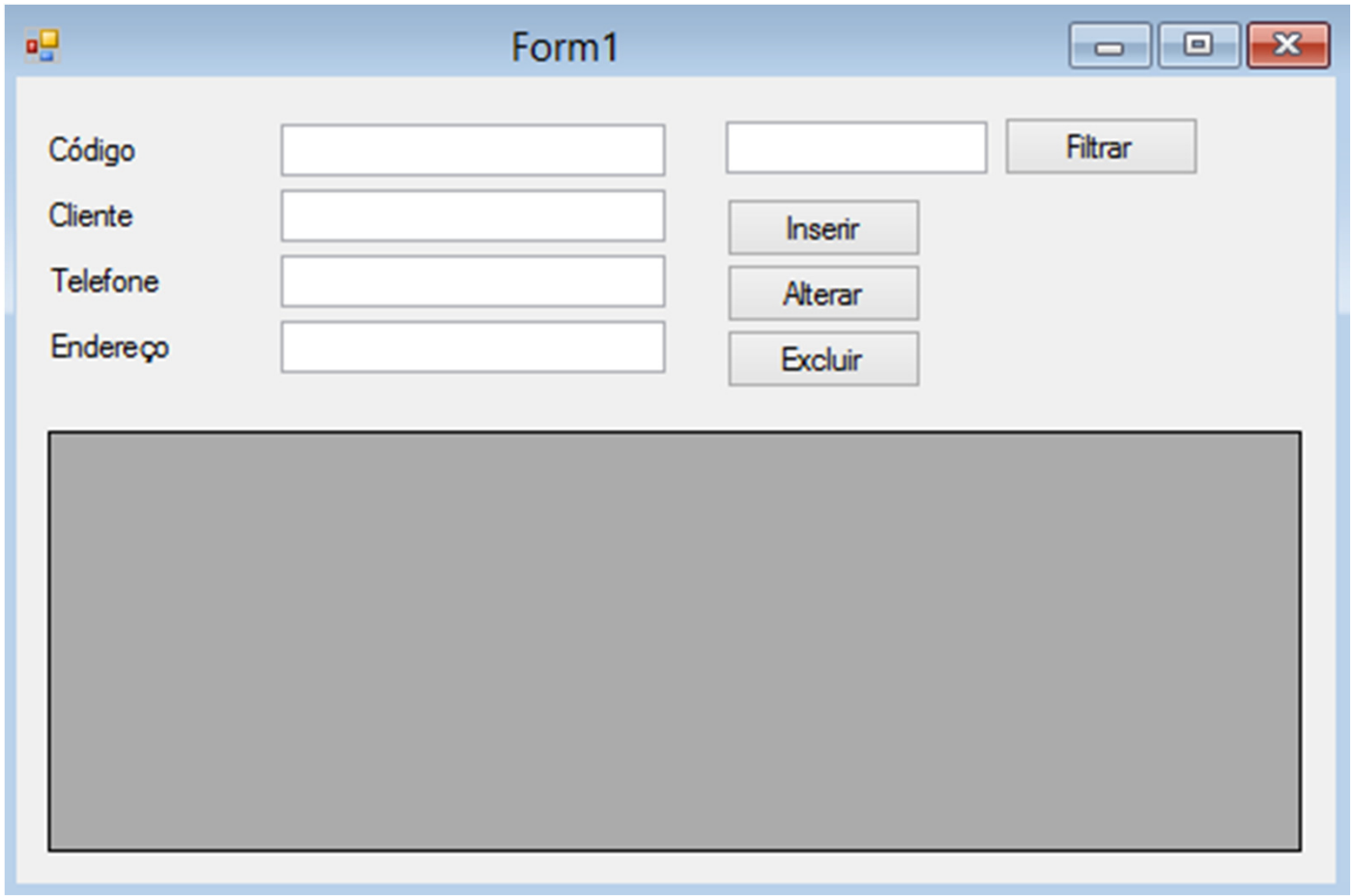
- Na classe ClientesDAL criar metodo Listagem do tipo DataTable (necessário: using System.Data)

RegraNegocio

- Na Classe ClienteBLL apenas chame o método Listagem da camada AcessaDados
- Neste primeiro exemplo não vamos criar nenhuma regra de acesso (exemplo: nome cliente obrigatório no incluir)

Interface Usuario

- Crie no Form1 o modelo abaixo com DataGridView:



Interface Usuario

- Crie o método AtualizaGrid para visualizar os dados da tabela clientes no DataGridView.
- Chame o método no Load

Filtro

- Implemente o botão Filtrar.
- Na Classe ClienteDAL crie outro método Listagem do tipo DataTable só que agora com parâmetro (string filtro).
- Chame na camada BLL e na Interface.
- Verifique na procedure quais filtros foram feitos e teste.

Inserir Dados

- O campo Id é auto incremento. Deixe o TextBox somente leitura.
- Crie o método na camada DAL, chame no BLL e visualize na Interface no botão Inserir e crie um metodo Limpar para limpar os campos.

- Não esqueça de criar uma label para mensagem.

Alterar Dados

- Para alterar os dados podemos usar o evento CellClick do dataGridView para selecionar um registro para ser Alterado, Deletado ou Atualizado.

- Use o mesmo procedimento do inserir.

Excluir Dados

- Execute os mesmo procedimentos criando uma condição para aparecer uma tela de confirmação da exclusão(S/N)

Formantando DataGridView

- Crie um método para formatar o tamanho das colunas no DataGridView
- Chame-o no método AtualizaGrid

Implementações futuras

- Libere os botões inserir, deletar e alterar apenas quando for requisitado (crie um botão)