Desenvolvimento de CRUD para Cadastro de Cliente

Pedro Henrique Machado Porath

Curso Tecnólogo em Análise e Desenvolvimento de Sistemas – Universidade do Vale do Itajaí (Univali) – Campus São José

São José – SC – Brasil

phporath@gmail.com

***Abstract.*** *This report has the function of showing the activities developed for the creation of a Customer Registration application for the Loyalty Programs of any restaurant, using the C # language.*

***Resumo.*** *Este relatório tem por função mostrar as atividades desenvolvidas para a criação de uma aplicação de Cadastro de Clientes do Programas de Fidelidade de um restaurante qualquer, a partir do uso da linguagem C#.*

# Introdução

Durante o primeiro ciclo de disciplinas da segunda fase do curso de Análise e Desenvolvimento de Sistemas da Universidade do Vale do Itajaí (Univali), através da integração dos conteúdos das disciplinas de Engenharia de Requisitos e Programming and Data Persistence foi possível na disciplina integradora “Hands on Work III” seguir a proposta do plano de ensino que era o desenvolvimento de uma aplicação considerando uma estrutura básica de CRUD (Create: Criar ou adicionar novas entradas; Read: Ler, recuperar ou listar as entradas existentes; Update: Atualizar, editar entradas existentes e Delete: Remover entradas existentes).

Além de realizar o desenvolvimento da aplicação de CRUD, o projeto deve conter a especificação dos requisitos da aplicação, como os requisitos funcionais (RF), regras de negócios (RN) e requisitos não-funcionais (RNF).

É importante ressaltar que para esse projeto, foi escolhido como tema o desenvolvimento de uma aplicação de Cadastro de Clientes do Programas de Fidelidade de um restaurante qualquer.

# Fluxograma da Aplicação

Visando facilitar o entendimento das atividades a serem implementadas, além dos requisitos da aplicação também é interessante o desenvolvimento do fluxo de processos através de notações BPMN (Business Process Model and Notation).

A primeira ação do sistema de Cadastro de Clientes é a visualização do menu principal, e a partir do menu o funcionário pode realizar as ações do CRUD que é a inserção, leitura, atualização e exclusão de clientes do Banco de Dados, conforme apresentado na Figura 1.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Figura 1: Fluxo de Processos do CRUD. Elaborado através do software Bizagi Modeler.

A partir do fluxograma do sistema de CRUD é possível especificar algumas questões:

* Nome do software: SGCC – Sistema Gerenciador de Cadastro de Clientes.
* Objetivos do produto: Desenvolver uma ferramenta que gerencie o cadastro dos clientes dispostos em um Banco de Dados visando conhecer melhor o perfil dos clientes e inseri-los no programa de fidelização.
* Áreas e processos de negócio envolvidos: Gestão de processos e relacionamento com cliente.
* Limites de atuação: clientes não irão interagir com o sistema.
* Requisitos futuros: desenvolver funcionalidades para gestão do programa de fidelidade.

# Requisitos de Software (Aplicação)

A definição dos requisitos de um software é uma etapa muito importante no processo de desenvolvimento de software, uma vez que os requisitos representam especificam as funções, objetivos, propriedades e restrições que um sistema deve possuir. Dessa forma, quando os requisitos foram bem definidos com as partes interessadas (stakeholders) mais facilitado é o processo de desenvolvimento de um sistema.

Dessa forma, antes do desenvolvimento foram definidas as regras de negócios, assim como os requisitos funcionais e não funcionais que serão apresentadas a seguir. Primeiramente deve-se registrar que a regra de negócio se refere a como o sistema irá executar tal ação, enquanto o requisito funcional é responsável por informar o que o sistema deverá fazer.

Como aplicação desenvolvida para esse projeto é simples, ela apenas possui uma regra de negócio, o gerenciamento do cadastro de clientes. Entretanto vale observar que essa regra de negócio possui dependência em quatro requisitos funcionais (que serão mostrados a seguir).

**Regras de Negócio:**

|  |  |
| --- | --- |
| **RN001** | |
| Nome | Sistema deverá ser capaz de gerenciar um Banco de Dados com informações dos clientes. |
| Módulo | Comercial |
| Data de criação | 05/09/2020 |
| Autor | Pedro Porath |
| Versão | 1.0 |
| Dependência | RF001, RF002, RF003 e RF004. |
| Descrição | Inserir, visualizar, atualizar e excluir dados de clientes no Banco de Dados. |
| Tipo | Habilitações de ação |

Os requisitos funcionais têm por objetivo registrar as funcionalidades e os serviços de um sistema, em outras palavras os requisitos funcionais se referem ao que o sistema deve fazer e quais funções fornecerá ao cliente. A seguir é possível ver a documentação de quatro requisitos funcionais (RF001, RF002, RF003 e RF004).

**Requisitos Funcionais:**

|  |  |
| --- | --- |
| **[RF001] Incluir Cliente no BD** | |
| Ator | Operador de Caixa |
| Prioridade | Essencial |
| Entradas e pré-condições | Dados pessoais e endereço de residência |
| Saídas e pós-condições | Gravar no Banco de Dados as informações do cliente. |
| Fluxo de eventos principal | 1 🡪 Solicitar nome complete do cliente;  2 🡪 Solicitar RG do cliente;  3 🡪 Solicitar CPF do cliente;  4 🡪 Solicitar Sexo do cliente;  5 🡪 Solicitar data de nascimento;  6 🡪 Solicitar e-mail do cliente;  7 🡪 Solicitar celular do cliente;  8 🡪 Solicitar endereço residencial do cliente;  9 🡪 Solicitar CEP de residência do cliente;  10 🡪 Solicitar bairro de residência do cliente;  11 🡪 Solicitar município de residência do cliente;  12 🡪 Solicitar UF de residência do cliente;  13 🡪 Se todos os campos foram preenchidos, clicar no botão “Inserir” para gravar os dados do cliente no Banco de Dados. |

|  |  |
| --- | --- |
| **[RF002] Ler Cliente no BD** | |
| Ator | Operador de Caixa |
| Prioridade | Essencial |
| Entradas e pré-condições | Dados de clientes no Banco de Dados |
| Saídas e pós-condições | Não há. |
| Fluxo de eventos principal | 1 🡪 Através do Data Grid View visualizar as informações dos clientes registrados no BD. |

|  |  |
| --- | --- |
| **[RF003] Atualizar Cliente no BD** | |
| Ator | Operador de Caixa |
| Prioridade | Essencial |
| Entradas e pré-condições | Dados de clientes no Banco de Dados |
| Saídas e pós-condições | Gravar no Banco de Dados as atualizações realizadas nas no cadastro do cliente. |
| Fluxo de eventos principal | 1 🡪 Através dos campos de Combo Box, Text Box e atualizar as informações dos clientes registrados no BD;  2 🡪 Após ajustes nos campos que precisam ser ajustados, clicar no botão “Atualizar” para atualizar os dados do cliente no Banco de Dados. |

|  |  |
| --- | --- |
| **[RF004] Excluir Cliente no BD** | |
| Ator | Operador de Caixa |
| Prioridade | Essencial |
| Entradas e pré-condições | Dados de clientes no Banco de Dados |
| Saídas e pós-condições | Excluir do Banco de Dados o cadastro de um cliente. |
| Fluxo de eventos principal | 1 🡪 Através do Data Grid View visualizar as informações dos clientes registrados no BD e identificar o cliente que será removido do Banco de Dados;  2 🡪 Após identificar o cliente que será removido, deve-se selecionar a linha do cadastro no Data Grid View e clicar no botão “Excluir” para remover os dados do cliente no Banco de Dados. |

Compreendido a importância assim como listados os requisitos funcionais do sistema a ser implantado, também é necessário entender quais são os requisitos não funcionais do atual sistema. Primeiramente deve-se registrar que os requisitos não funcionais definem propriedades e restrições do sistema como tempo, espaço, linguagens de programação, versões do compilador, SGBD, Sistema Operacional, método de desenvolvimento etc. A seguir é possível ver a documentação de quatro requisitos não funcionais (RNF001, RNF002, RNF003 e RNF004).

**Requisitos Não Funcionais:**

|  |  |  |
| --- | --- | --- |
| Identificador: RNF001 | Categoria: Desempenho | |
| Nome: O sistema precisa possuir alto desempenho. | | |
| Data de criação: 05/09/2020 | | Autor: Pedro Porath |
| Data da última alteração: - | | Autor: - |
| Versão: 1.0 | | Prioridade: Essencial |
| Descrição: Uma vez que clientes não gostam de perder tempo informando os dados pessoais para cadastro em lojas, o sistema deve possuir alto desempenho para gravar dados no Banco de Dados, onde as requisições devem durar no máximo 3 segundos. | | |

|  |  |  |
| --- | --- | --- |
| Identificador: RNF002 | Categoria: Layout | |
| Nome: O sistema deve ter um layout fácil e com boa usabilidade | | |
| Data de criação: 05/09/2020 | | Autor: Pedro Porath |
| Data da última alteração: - | | Autor: - |
| Versão: 1.0 | | Prioridade: Essencial |
| Descrição: O sistema deve ter um layout fácil e com boa usabilidade para facilitar o trabalho dos funcionários do estabelecimento (principalmente do operador de caixa). | | |

|  |  |  |
| --- | --- | --- |
| Identificador: RNF003 | Categoria: Administrador de Servidor | |
| Nome: O sistema gerenciador de banco de dados utilizado será o MySQL. | | |
| Data de criação: 05/09/2020 | | Autor: Pedro Porath |
| Data da última alteração: - | | Autor: - |
| Versão: 1.0 | | Prioridade: Essencial |
| Descrição: O gerenciamento dos dados cadastrados no sistema deverá ser realizado pelo SGBD SQL Server. | | |

|  |  |  |
| --- | --- | --- |
| Identificador: RNF004 | Categoria: Administrador de Servidor | |
| Nome: A linguagem de programação utilizada para desenvolvimento do sistema será o C#. | | |
| Data de criação: 05/09/2020 | | Autor: Pedro Porath |
| Data da última alteração: - | | Autor: - |
| Versão: 1.0 | | Prioridade: Essencial |
| Descrição: O desenvolvimento do sistema será desenvolvido através da linguagem de programação C# através do Windows Form do Visual Studio 2019. | | |

|  |  |  |
| --- | --- | --- |
| Identificador: RNF004 | Categoria: Usabilidade | |
| Nome: Botão para ajuda (“help”) do Sistema. | | |
| Data de criação: 05/09/2020 | | Autor: Pedro Porath |
| Data da última alteração: - | | Autor: - |
| Versão: 1.0 | | Prioridade: Desejável |
| Descrição: Haverá um botão de “help” no sistema que deverá apontar para uma documentação informando o correto funcionamento do sistema. | | |

# Desenvolvimento

O projeto desenvolvido para a disciplina de “Hands on Work III”, previu o desenvolvimento de uma aplicação de CRUD, onde como tema escolhido foi o cadastro de clientes um programas de fidelidade de um restaurante qualquer, onde o sistema foi desenvolvido a partir da linguagem de programação C# e será utilizado o Banco de Dados SQL Server.

## Desenho e Prototipagem

A partir do software Visual Studio 2019, foi iniciado o projeto com a elaboração do desenho e prototipagem da proposta do sistema. O requisito não funcional [RNF002] indica que o sistema deve ter um layout fácil e com boa usabilidade, por isso optou-se pelo uso de apenas um *“form”* onde todas as operações de um CRUD pudessem ser realizadas, como apresentado através da Figura 2.

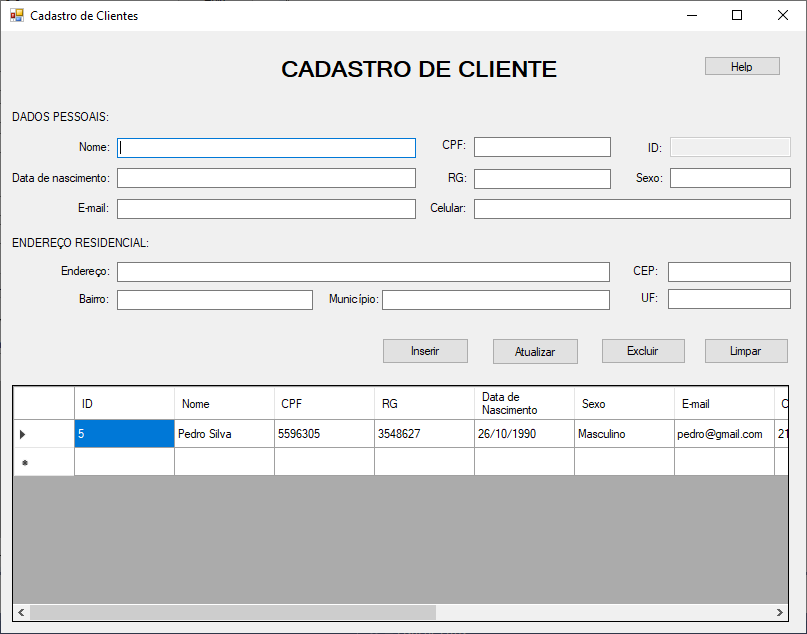


Figura 2. Prototipagem da Aplicação de CRUD.

## Desenvolvimento do código

### Sistema Gerenciador de Banco de Dados

Nesse projeto optou-se pela utilização do Sistema Gerenciador de Banco de Dados MySQL. Dessa forma, foi necessário fazer o download do software XAMPP Control Panel v3.2.4, a partir dele é iniciado o Apache e o MySQL e depois é aberto o phpMyAdmin pelo botão Admin.

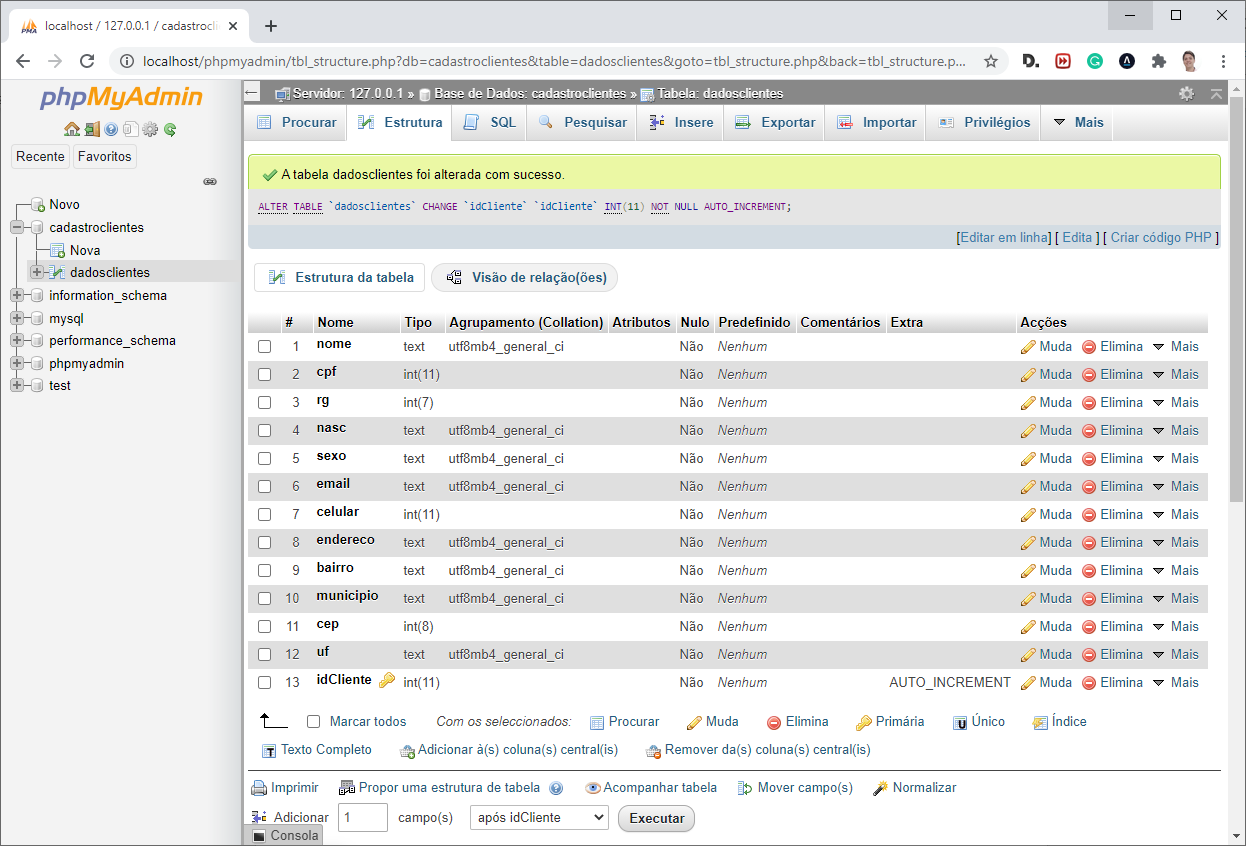


Figura 3. Visualização da tabela “dadoscliente” do BD “cadastroClientes”.

### Escolha do ambiente de desenvolvimento

A segunda etapa de desenvolvimento do projeto foi escolher o ambiente de desenvolvimento do código. Como sugerido pelos professores das disciplinas envolvidas no projeto, para o desenvolvimento do sistema, foi utilizado o Visual Studio 2019 através da aplicação *“Windows Form.*

O código do sistema desenvolvido além de estar compartilhado no GitHub[[1]](#footnote-1), pode ser avaliado por meio do anexo A disposto no fim desse documento.

# Resultados

Com o aprendizado de conceitos básicos da linguagem de C# e SQL, está sendo possível desenvolver um sistema com funcionalidade de CRUD e aprender um pouco sobre o gerenciamento de Banco de Dados. Como o sistema ainda está incompleto, para essa primeira etapa os arquivos desenvolvidos para o sistema em questão ainda não se encontram no repositório GitHub com acesso público. O que será feito na próxima entrega, uma vez que além de cumprir a função social de compartilhamento de conhecimento, fica disponível para que outros usuários possam indicar melhorias.

## Cronograma

|  |  |
| --- | --- |
| Atividade | Data de conclusão |
| Criar o Banco de Dados e conectar com o sistema. | 15/09/2020 |
| Finalizar implementação dos requisitos funcionais. | 20/09/2020 |
| Desenvolver documento de “Help” para a aplicação. | 24/09/2020 |
| Hospedar o projeto no GitHub. | 25/09/2020 |
| Finalizar relatório final | 26/09/2020 |

Quadro 1. Definição do cronograma de atividades.

# Referências

DEVMEDIA (Brasil). Introdução a Requisitos de Software. [2020?]. Disponível em: https://www.devmedia.com.br/introducao-a-requisitos-de-software/29580. Acesso em: 06 set. 2020.

ANEXO A

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  using MySql.Data.MySqlClient;  namespace cadastroCliente  {  public partial class cadastroForm : Form  {  public cadastroForm()  {  InitializeComponent();  }  private MySqlConnectionStringBuilder conexaoBanco()  {  MySqlConnectionStringBuilder conexaoBD = new MySqlConnectionStringBuilder();  conexaoBD.Server = "localhost";  conexaoBD.Database = "cadastroclientes";  conexaoBD.UserID = "root";  conexaoBD.Password = "";  return conexaoBD;  }  private void label4\_Click(object sender, EventArgs e)  {  }  private void inserirButton\_Click(object sender, EventArgs e)  {  MySqlConnectionStringBuilder conexaoBD = conexaoBanco();  MySqlConnection realizaConexacoBD = new MySqlConnection(conexaoBD.ToString());  try  {  realizaConexacoBD.Open();  MySqlCommand comandoMySql = realizaConexacoBD.CreateCommand();  comandoMySql.CommandText = "INSERT INTO dadosclientes (idCliente,nome,cpf,rg,nasc,sexo,email,celular,endereco,bairro,municipio,cep,uf) " +  "VALUES('" + idTextBox.Text + "','" + nomeTextBox.Text + "', '" + cpfTextBox.Text + "', '" + rgTextBox.Text + "','" + dataTextBox.Text + "','" + sexoTextBox.Text + "','" + emailTextBox.Text + "', '" + celularTextBox.Text + "','" + enderecoTextBox.Text + "','" + bairroTextBox.Text + "','" + municipioTextBox.Text + "','" + cepTextBox.Text + "','" + ufTextBox.Text + "')";  comandoMySql.ExecuteNonQuery();  realizaConexacoBD.Close();  MessageBox.Show("Inserido com sucesso");  limparCampos();  atualizarGrid();  }  catch (Exception ex)  {  Console.WriteLine(ex.Message);  }  }  private void excluirButton\_Click(object sender, EventArgs e)  {  MySqlConnectionStringBuilder conexaoBD = conexaoBanco();  MySqlConnection realizaConexacoBD = new MySqlConnection(conexaoBD.ToString());  try  {  realizaConexacoBD.Open(); //Abre a conexão com o banco  MySqlCommand comandoMySql = realizaConexacoBD.CreateCommand(); //Crio um comando SQL  comandoMySql.CommandText = "DELETE FROM dadosclientes WHERE idCliente = '" + idTextBox.Text + "'";  comandoMySql.ExecuteNonQuery();  realizaConexacoBD.Close(); // Fecho a conexão com o banco  MessageBox.Show("Deletado com sucesso"); //Exibo mensagem de aviso  atualizarGrid();  limparCampos();  }  catch (Exception ex)  {  //MessageBox.Show("Não foi possivel abrir a conexão! ");  Console.WriteLine(ex.Message);  }  }  private void limparCampos()  {  idTextBox.Clear();  nomeTextBox.Clear();  cpfTextBox.Clear();  rgTextBox.Clear();  dataTextBox.Clear();  sexoTextBox.Clear();  celularTextBox.Clear();  emailTextBox.Clear();  enderecoTextBox.Clear();  bairroTextBox.Clear();  municipioTextBox.Clear();  cepTextBox.Clear();  ufTextBox.Clear();  }  private void dataGridView\_CellContentClick(object sender, DataGridViewCellEventArgs e)  {  if (dataGridView.Rows[e.RowIndex].Cells[e.ColumnIndex].Value != null)  {  dataGridView.CurrentRow.Selected = true;  //preenche os textbox com as células da linha selecionada  idTextBox.Text = dataGridView.Rows[e.RowIndex].Cells["coID"].FormattedValue.ToString();  nomeTextBox.Text = dataGridView.Rows[e.RowIndex].Cells["coNome"].FormattedValue.ToString();  cpfTextBox.Text = dataGridView.Rows[e.RowIndex].Cells["coCPF"].FormattedValue.ToString();  rgTextBox.Text = dataGridView.Rows[e.RowIndex].Cells["coRG"].FormattedValue.ToString();  dataTextBox.Text = dataGridView.Rows[e.RowIndex].Cells["coDataNascimento"].FormattedValue.ToString();  sexoTextBox.Text = dataGridView.Rows[e.RowIndex].Cells["coSexo"].FormattedValue.ToString();  emailTextBox.Text = dataGridView.Rows[e.RowIndex].Cells["coEmail"].FormattedValue.ToString();  celularTextBox.Text = dataGridView.Rows[e.RowIndex].Cells["coCelular"].FormattedValue.ToString();  enderecoTextBox.Text = dataGridView.Rows[e.RowIndex].Cells["coEndereco"].FormattedValue.ToString();  bairroTextBox.Text = dataGridView.Rows[e.RowIndex].Cells["coBairro"].FormattedValue.ToString();  municipioTextBox.Text = dataGridView.Rows[e.RowIndex].Cells["coMunicipio"].FormattedValue.ToString();  cepTextBox.Text = dataGridView.Rows[e.RowIndex].Cells["coCEP"].FormattedValue.ToString();  ufTextBox.Text = dataGridView.Rows[e.RowIndex].Cells["coUF"].FormattedValue.ToString();  }  }  private void atualizarButton\_Click(object sender, EventArgs e)  {  MySqlConnectionStringBuilder conexaoBD = conexaoBanco();  MySqlConnection realizaConexacoBD = new MySqlConnection(conexaoBD.ToString());  try  {  realizaConexacoBD.Open(); //Abre a conexão com o banco  MySqlCommand comandoMySql = realizaConexacoBD.CreateCommand(); //Crio um comando SQL  comandoMySql.CommandText = "UPDATE dadosclientes SET nome = '" + nomeTextBox.Text + "', cpf = '" + cpfTextBox.Text + "', rg = '" + rgTextBox.Text + "', nasc = '" + dataTextBox.Text + "', sexo = '" + sexoTextBox.Text + "', email = '" + emailTextBox.Text + "', celular = '" + celularTextBox.Text + "', endereco = '" + enderecoTextBox.Text + "', bairro = '" + bairroTextBox.Text + "', municipio = '" + municipioTextBox.Text + "', cep = '" + cepTextBox.Text + "', uf = '" + ufTextBox.Text + "' WHERE idCliente = " + idTextBox.Text + "";  comandoMySql.ExecuteNonQuery();  realizaConexacoBD.Close(); // Fecho a conexão com o banco  MessageBox.Show("Atualizado com sucesso"); //Exibo mensagem de aviso  atualizarGrid();  limparCampos();  }  catch (Exception ex)  {  //MessageBox.Show("Não foi possivel abrir a conexão! ");  Console.WriteLine(ex.Message);  }  }  private void atualizarGrid()  {  MySqlConnectionStringBuilder conexaoBD = conexaoBanco();  MySqlConnection realizaConexacoBD = new MySqlConnection(conexaoBD.ToString());  try  {  realizaConexacoBD.Open();  MySqlCommand comandoMySql = realizaConexacoBD.CreateCommand();  comandoMySql.CommandText = "SELECT \* FROM dadosclientes";  MySqlDataReader reader = comandoMySql.ExecuteReader();  dataGridView.Rows.Clear();  while (reader.Read())  {  DataGridViewRow row = (DataGridViewRow)dataGridView.Rows[0].Clone();//FAZ UM CAST E CLONA A LINHA DA TABELA  row.Cells[0].Value = reader.GetInt32(12);//idCliente  row.Cells[1].Value = reader.GetString(0);//nome  row.Cells[2].Value = reader.GetInt32(1);//cpf  row.Cells[3].Value = reader.GetInt32(2);//rg  row.Cells[4].Value = reader.GetString(3);//nasc  row.Cells[5].Value = reader.GetString(4);//sexo  row.Cells[6].Value = reader.GetString(5);//email  row.Cells[7].Value = reader.GetInt32(6);//celular  row.Cells[8].Value = reader.GetString(7);//endereco  row.Cells[9].Value = reader.GetString(8);//bairro  row.Cells[10].Value = reader.GetString(9);//municipio  row.Cells[11].Value = reader.GetInt32(10);//cep  row.Cells[12].Value = reader.GetString(11);//uf  dataGridView.Rows.Add(row);//ADICIONO A LINHA NA TABELA  }  realizaConexacoBD.Close();  }  catch (Exception ex)  {  MessageBox.Show("Can not open connection ! ");  Console.WriteLine(ex.Message);  }  }  //private void cadastroclientes\_Load(object sender, EventArgs e)  //{  // atualizarGrid();  //}  private void limparButton\_Click(object sender, EventArgs e)  {  limparCampos();  }  private void cadastroForm\_Load(object sender, EventArgs e)  {  atualizarGrid();  }  }  } |

1. Projeto no repositório do GitHub: <https://github.com/phporath/Univali-ADS/tree/master/Disciplina-Hands-on-Work-III> [↑](#footnote-ref-1)