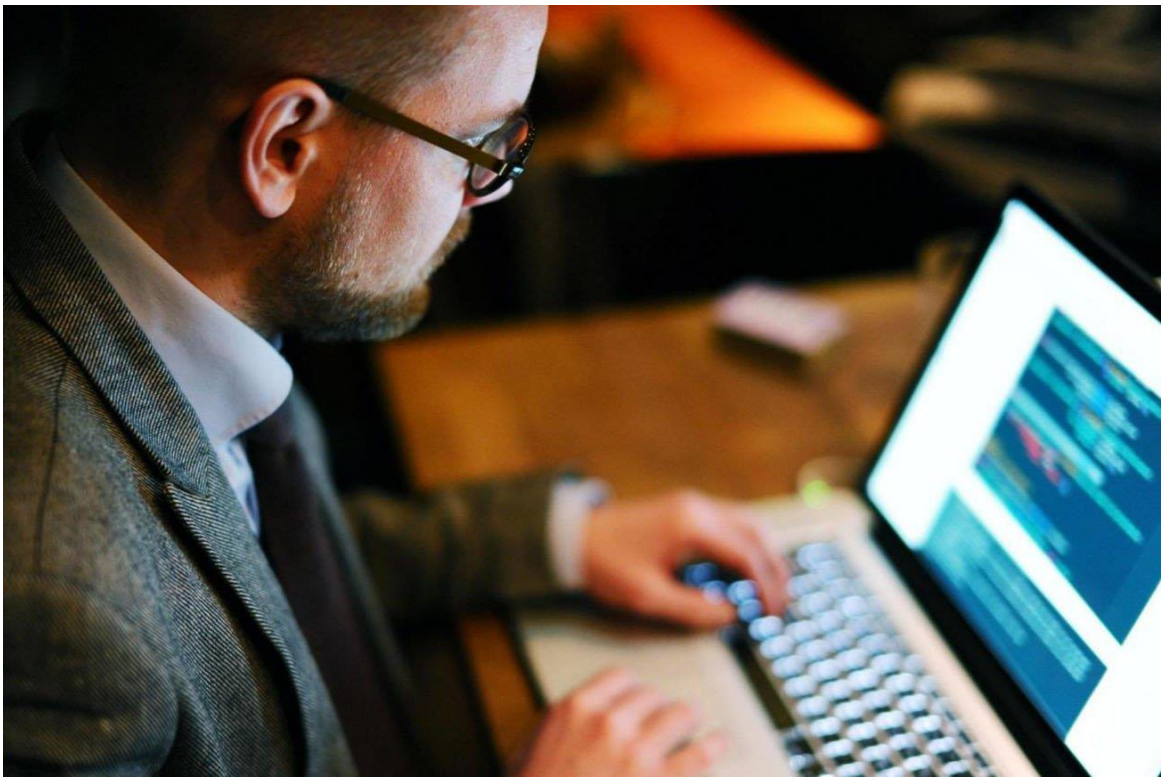
MANUAL DE PROGRAMADOR

Football Life



Tomás Silva

ÍNDICE

INTRODUÇÃO

Este programa foi desenvolvido em C#, no Visual Studio 2019, em Windows Forms e tem uma base de dados criada no Microsoft SQL Server Management Studio 18.

O utilizador também deverá ter uma capacidade mínima de 2GB de RAM na sua máquina e um sistema operativo Windows para o programa ter capacidade para exercer todas as suas funções.

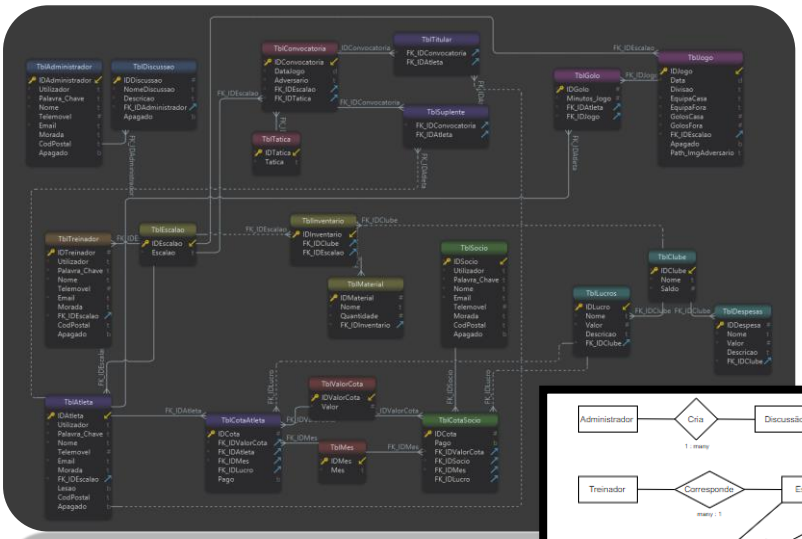
Este programa é muito baseado em ler, modificar e eliminar dados da base de dados, não tendo muitas funções sem o uso da base de dados.

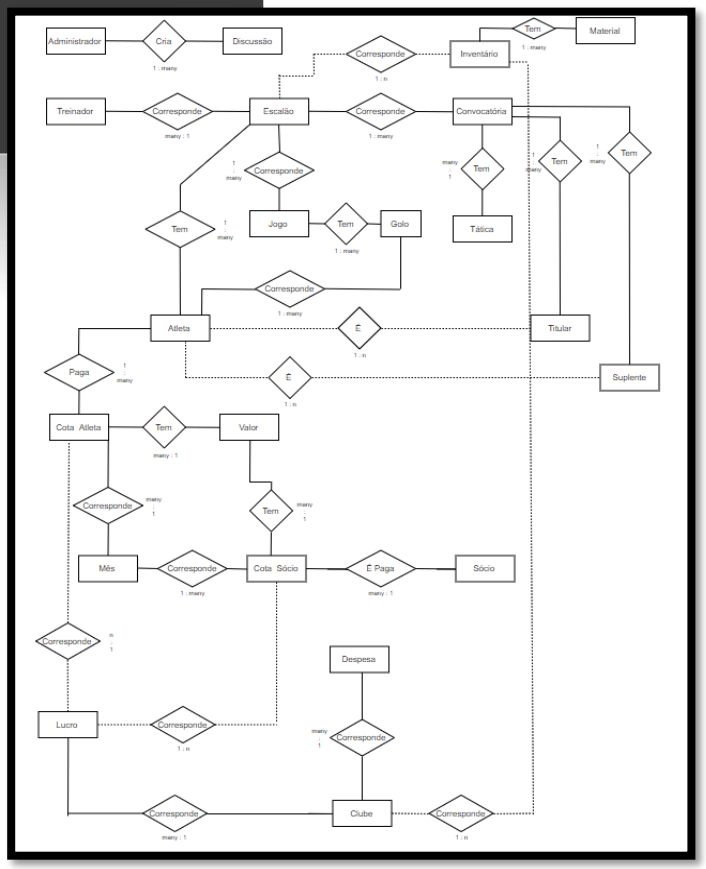
Neste manual, vai ser explicado como a base de dados funciona e as principais e mais difíceis partes de código de "Football Life", para que o programador possa percebê-lo e mais tarde alterá-lo, caso necessite.

BASE DE DADOS

A base de dados do programa tem por volta de 26 tabelas sendo 4 delas de utilizadores, ou seja, a tabela "TblAdministrador", a tabela "TblTreinador", a tabela "TblAtletas" e a tabela "TblSocio". Isto para que a relação com as outras tabelas fosse mais fiável e compreensível.

A esquema da base de dados é esta imagem abaixo e segue o seu diagrama de relação-entidade:





CURRENT USER

Para guardar valores sobre o *current user*, foram utilizadas propriedades situadas no "Program.cs". O seu nome e as respetivas funções são as seguintes:

- **CurrentFuncaoUser –** Indica qual é a função do *current user*. Os seus valores são guardados no "Login.cs" e são valores únicos, atribuídos pelo programador ("Admin", "Treinador", "Atleta" e "Socio");

- **CurrentIDUser –** Indica qual é o ID do *current user*. O seu valor é guardado no "Login.cs" e corresponde a um valor da base de dados;

- **CurrentIDEscalao –** Indica qual o ID do escalão do *current user*, caso tiver (se for treinador ou atleta). O seu valor é guardado no "Login.cs" e corresponde a um valor da base de dados.

```
99+ references
public static string CurrentFuncaoUser { get; set; }
32 references
public static int CurrentIDUser { get; set; }
36 references
public static int CurrentIDEscalao { get; set; }
```

LOGIN.CS

É no "Login.cs" que realmente começa o código mais complicado. Para começar o utilizador tem a opção de guardar a sua password e para isso usei as Propriedades do programa e inseri duas variáveis a para o nome do utilizador e outra para a password. Foi utilizada esta forma, porque mesmo depois do programa ser fechado os valores ficarão guardados.

```
if (chb_Lembrar.Checked)
{
    Properties.Settings.Default.username = tb_Utilizador.Text;
    Properties.Settings.Default.password = tb_Password.Text;
    Properties.Settings.Default.Save();
}
else
{
    Properties.Settings.Default.username = "";
    Properties.Settings.Default.password = "";
    Properties.Settings.Default.Save();
}
```

Para, então, fazer o login, foram utilizados DataReaders, com querys a pedir utilizadores em todas as tabelas com o nome e a password correspondestes. Como os utilizadores não podem tem utilizadores iguais, apenas umas das tabelas (TblAdministrador, TblTreinador, TblAtleta e TblSocio) dará valores.

Quando feito o Login, então as propriedades faladas anteriormente, deixam de estar vazias e tem um valor para o funcionamento do resto do programa.

```
if (IDAdmin != "")
{
    Program.CurrentFuncaoUser = "Admin";
    Program.CurrentIDUser = Convert.ToInt32(IDAdmin);
    Program.CurrentIDEscalao = 0;

    PaginaInicial_Admin PgAdmin = new PaginaInicial_Admin();
    this.Hide();
    PgAdmin.ShowDialog();
    this.Dispose();
}
```

QUERYS E NONQUERYS

Para manusear com a base de dados, seja com o uso de querys (Selets) ou nonquerys (Inserts, Updates e Deletes) foram usados "SqlCommands". Para isto é necessário o uso da biblioteca System.Data.SqlClient.

```
//SELECIONA O NOME DO ATLETA PARA INSERIR NA TABELA LUCROS
SqlDataReader drAtleta;
string QueryAtleta = ("SELECT Nome FROM dbo.TblAtleta WHERE IDAtleta = " + Program.CurrentIDUser);
SqlCommand CommandAtleta = new SqlCommand(QueryAtleta, con);
drAtleta = CommandAtleta.ExecuteReader();
while (drAtleta.Read())
{
    Nome = drAtleta["Nome"].ToString();
}
drAtleta.Close();
```

PAGINAÇÃO DE DADOS DA BASE DE DADOS

Como alguns forms têm inúmeros dados a serem carregados, adicionei a função paginação para que mesmos dados carregassem ao mesmo tempo e esta função pode ser encontrada nos forms: Jogos, PaginaInicial_Admin, PaginaInicial_Treinador, PaginaInicial_Atleta, PaginaInicial_Socio, Utilizadores e Financiamento. Isto está representado na seguinte imagem e tem um código que adiciona mais um certo valor as quantas vezes que o utilizador clicar na imagem.

```
1 reference
private void Img1_Click(object sender, EventArgs e)
{
    ADM += 8;
    Admins();
}
```

USER CONTROL – "JOGO"

Um *user control* foi usado para jogos. Neste caso, o *user control* é o resultado de um jogo e cada vez que um jogo é lido da base de dados o *user control* "Jogo" é adicionado ao *flowlayoutpanel* dos jogos, pois fornece maior flexibilidade e maior facilidade para a reutilização.

Para além do *user control*, foi usado também algo parecido, mas criado em código, criando assim *labels*, *buttons*, entre outras coisas em código, tal como na imagem.

```
Panel panel = new Panel();
panel.Width = 800;
panel.Height = 250;
panel.Margin = new Padding(5, 5, 5, 0);
panel.Anchor = AnchorStyles.Top;
panel.BorderStyle = BorderStyle.Fixed3D;
panel.BackColor = Color.White;
panel.Visible = true;
panel.Name = "Panel" + IDDiscussao;
flowpanel_Discussoes.Controls.Add(panel);
```

JOGOS - IMAGENS

Todos os jogos têm uma imagem de um logo adversário e, para isto, foi necessário adicionar o caminho da imagem selecionada à base de dados, para isto o programa copia a imagem para uma pasta dentro do programa e guarda o caminho para a base de dados. Para isto é necessário usar a biblioteca System.IO.

```
CommandINSERT.Parameters.AddWithValue("@Path_ImgAdversario", folderpath + Path.GetFileName(open.FileName));

string fileName = Path.Combine(folderpath, Path.GetFileName(filePath));

if (!File.Exists(fileName))
{
    File.Copy(filePath, fileName, true);
}
```

GUARDAR FATURA COTA – *TXT FILES*

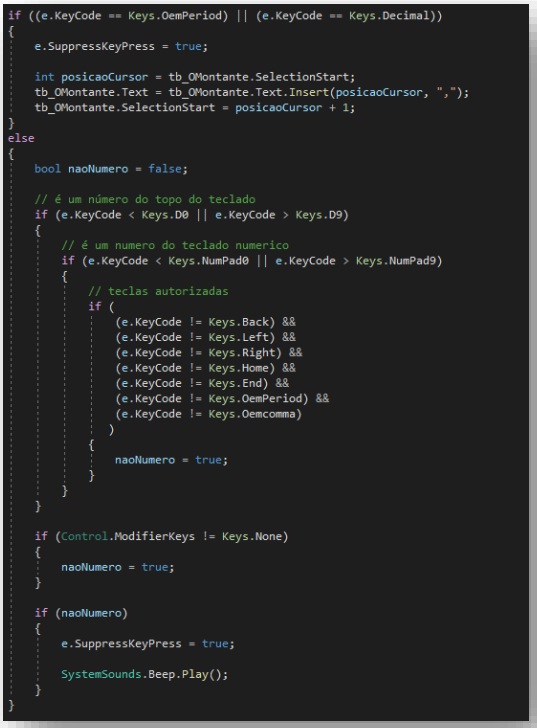
Quando o utilizador paga a sua cota tem direito a salvar uma fatura que lhe é dada. Apesar de não ser muito seguro porque o utilizador pode alterar o ficheiro, esta e gravada como um ficheiro de texto. Para isto é usado um "*for*" para o ficheiro verificar tudo o que está dentro do form "Fatura Cota" e salvar todas essas informações no ficheiro de texto. Para isto é necessário usar a biblioteca System.IO.

```
StreamWriter writer = new StreamWriter(save.OpenFile());

for(int i = 0; i < this.Controls.Count; i++)
{
    if(this.Controls[i] is Label)
    {
        if(Pagamento != "PayPal")
        {
            label_PP.Text = "";
            lbl_PayPal.Text = "";
        }
        else
        {
            label_NrCt.Text = "";
            lbl_Numero.Text = "";
            label_NmCt.Text = "";
            lbl_NmCartao.Text = "";
        }


        if (i == 0)
        {
            writer.WriteLine();
        }
        else if (i == 1)
        {
            writer.WriteLine();
            writer.WriteLine("(" + DateTime.Now + ")");
            writer.WriteLine();
        }
        else if (i == 7)
        {
            writer.WriteLine();
        }
        else if (Pagamento == "PayPal" && i == 13)
        {
            writer.WriteLine();
        }

        if (i % 2 != 0)
        {
            writer.WriteLine();
        }

        if (i == 0)
        {
            writer.Write("   " + this.Controls[i].Text.ToString());
        }
        else
        {
            writer.Write(" " + this.Controls[i].Text.ToString());
        }
```

RESTRIÇÕES DAS TEXTBOXS NUMÉRICAS

Para restringir o utilizador à utilização de números em certas caixas de texto, foi utilizado a biblioteca System.Media para caso do utilizador estar a introduzir um valor não número ele ser notificado.

Em algumas partes deste código também não só foram restringidas as caixas de texto, mas também foram substituídos o ponto por virgula ou o ponto por dois pontos.



```
if ((e.KeyCode == Keys.OemPeriod) || (e.KeyCode == Keys.Decimal))
{
    e.SuppressKeyPress = true;

    int posicaoCursor = tb_OMontante.SelectionStart;
    tb_OMontante.Text = tb_OMontante.Text.Insert(posicaoCursor, ",");
    tb_OMontante.SelectionStart = posicaoCursor + 1;
}
else
{
    bool naoNumero = false;

    // é um número do topo do teclado
    if (e.KeyCode < Keys.D0 || e.KeyCode > Keys.D9)
    {
        // é um numero do teclado numerico
        if (e.KeyCode < Keys.NumPad0 || e.KeyCode > Keys.NumPad9)
        {
            // teclas autorizadas
            if (
                (e.KeyCode != Keys.Back) &&
                (e.KeyCode != Keys.Left) &&
                (e.KeyCode != Keys.Right) &&
                (e.KeyCode != Keys.Home) &&
                (e.KeyCode != Keys.End) &&
                (e.KeyCode != Keys.OemPeriod) &&
                (e.KeyCode != Keys.Oemcomma)
              )
            {
                naoNumero = true;
            }
        }
    }

    if (Control.ModifierKeys != Keys.None)
    {
        naoNumero = true;
    }

    if (naoNumero)
    {
        e.SuppressKeyPress = true;

        SystemSounds.Beep.Play();
    }
}
```

PEQUENAS NOTAS

O código para as cotas ainda não está a funcionar corretamente, pois o PayPal ou os Multibancos não são verificados, nem valores são retirados dos mesmos, ou seja, o código está ainda para teste.