Relatório do trabalho da disciplina de Armazenamento e Acesso a Dados

A “C:Happy”

João Furtado – 21125

Rúben Gomes - 21118

Licenciatura em Engenharia de Sistemas Informáticos

Outubro de 2021

ESCOLA SUPERIOR DE TECNOLOGIA IPCA

Afirmo por minha honra que não recebi qualquer apoio não autorizado na realização deste trabalho prático. Afirmo igualmente que não copiei qualquer material de livro, artigo, documento web ou de qualquer outra fonte exceto onde a origem estiver expressamente citada.

João Furtado – 21125

Rúben Gomes - 21118

Índice

O Negócio da C:Happy

A pandemia provocada pela covid-19 trouxe à sociedade relevantes desafios nunca antes enfrentados. De acordo com vários estudos realizados por diferentes entidades verificou-se um grande impacto nas famílias, mas também uma ampla influência no desenvolvimento e comportamento das crianças

É neste sentido que a "C:Happy" (Children Happy) pretende trabalhar. Tendo em conta o que já mencionamos antes, sentimos a necessidade de ajudar e sensibilizar os pais para o estado da saúde mental dos seus filhos, de modo a proporcionar uma infância e um desenvolvimento saudável e feliz.

O primeiro contacto com um futuro cliente é feito pela própria empresa. Trabalhamos com várias escolas pelo país todo ao longo da pandemia para apresentar questionários aos pais realizados na nossa aplicação. De acordo com as respostas dadas, é realizada uma análise dos possíveis fatores que afetam os seus filhos, que será enviada aos pais com informações sobre os problemas encontrados nas crianças e o acompanhamento necessário à resolução dos mesmos. Os pais, de acordo com o resumo que lhes é apresentado, decidem se querem ser nossos clientes, ou seja, se querem ser acompanhados pelas nossas equipas especializadas.

Ao tornarem-se nossos clientes, são apresentados três planos de adesão, tendo em conta as necessidades de cada um.

No primeiro plano, realiza-se o aconselhamento através da aplicação, sensibilizando os pais a partir da leitura de estudos realizados sobre o impacto da pandemia nas crianças e propostas de como agir de forma correta. Assim como a criação de eventos para promover a interação das crianças umas com as outras. Os eventos serão realizados semanalmente (adequados à disponibilidade dos clientes), como por exemplo, visitas didáticas e atividades específicas para compreender o comportamento social.

Adicionalmente, existe outro plano focado no acompanhamento psicológico das crianças. O acompanhamento é feito de forma personalizada para cada criança conforme os seus problemas. Sendo assim, neste plano será realizado um acompanhamento psicológico semanalmente para as crianças, mas também a realização de palestras para os pais onde possam compreender o que fazer e como fazer, assim como um treino da interação direta com os seus filhos.

Por fim, disponibilizamos um plano que concatena os dois mencionados anteriormente.

Primeiramente na nossa aplicação os clientes terão de efetuar o seu registo, depois terão de registar vários dados acerca dos filhos, tais como o nome, idade, morada, género, escola que frequenta, atividades extracurriculares em que estão envolvidas, etc…

Seguidamente, os clientes irão escolher o seu plano de adesão, tal como o seu método de pagamento mensal. Na aplicação irão encontrar tudo aquilo a que têm direito conforme o plano escolhido, mas também um fórum onde todos os inscritos na aplicação possam conversar sobre os problemas que tiveram e as soluções encontradas, e também podendo colocar as questões que quiserem.

Para os inscritos nos planos de adesão, serão realizados questionários trimestrais que permitem tanto aos pais como às nossas equipas avaliar as diferenças da situação em que as crianças se encontravam com o progresso atual. Desta forma, é possível aos pais compreender se querem alterar o seu plano atual, manter ou cessar o contra.

Licenciatura em Engenharia de Sistemas Informáticos

João Furtado Rúben Gomes

Diagrama de ER

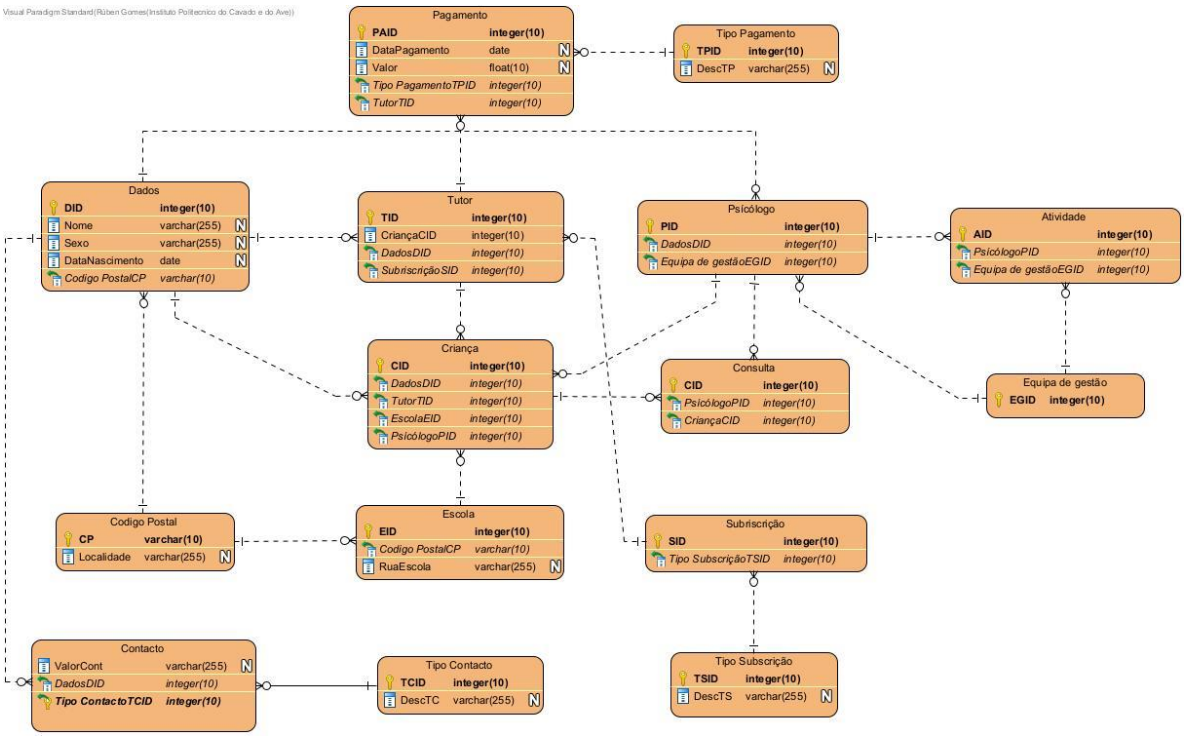
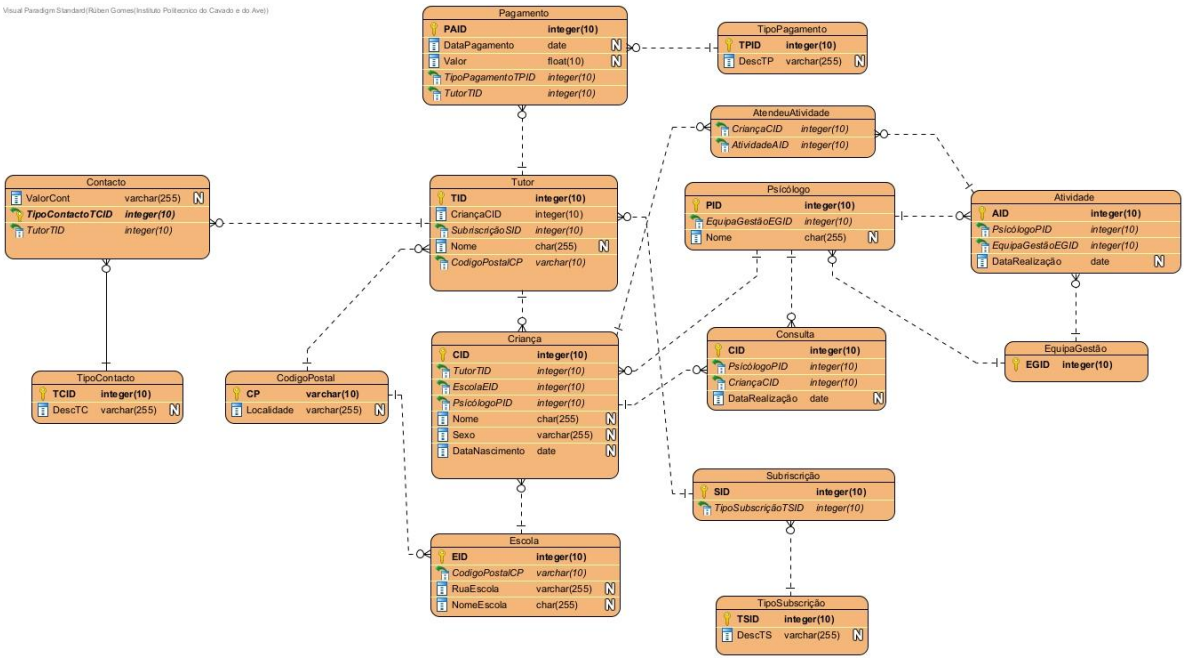


Diagrama de ER atualizado

<



Queries sobre a base de dados do projeto

```
--1 Qual a idades de todas as crianças inscritas?
	Select Nome, FLOOR(DATEDIFF(day, DataNascimento, getDate()) / 365.25) as
Idade
	From Criança
--2 Quais as crianças inscritas no 'Plano A'?
	Select Criança.Nome
	From TipoSubscrição join Subriscrição on
TipoSubscrição.TSID=Subriscrição.TipoSubscriçãoTSID
                                        join Tutor on
Subriscrição.SID=Tutor.SubriscriçãoSID
                                        join Criança on Tutor.TID=Criança.TutorTID
	Where TipoSubscrição.DescTS='Plano A'
	Group by Criança.Nome
--3 Qual o nome da criança mais nova?
	Select Criança.Nome, Criança.DataNascimento
	From Criança
	Where DataNascimento =(select max(DataNascimento) from Criança)
--4 Quais as localidades diferentes em que há crianças inscritas?
	Select distinct CodigoPostal.Localidade as Localidades
	From Tutor join CodigoPostal on Tutor.CodigoPostalCP=CodigoPostal.CP
--5 Quais as crianças que não estão inscritas a partir de alguma escola?
	Select Nome
	From Escola Left Join Criança on Escola.EID=Criança.EscolaEID
	Where EscolaEID is Null

--6 Para cada criança qual o plano em que está inscrita?
	Select Criança.Nome, TipoSubscrição.DescTS
	From TipoSubscrição join Subriscrição on
TipoSubscrição.TSID=Subriscrição.TipoSubscriçãoTSID
                                        join Tutor on
Subriscrição.SID=Tutor.SubriscriçãoSID
                                        join Criança on Tutor.TID=Criança.TutorTID
	Group by Criança.Nome, TipoSubscrição.DescTS
--7 Qual a média das idades das crianças inscritas por plano?
	Select TipoSubscrição.DescTS, AVG(FLOOR(DATEDIFF(day, DataNascimento,
getDate()) / 365.25))as Media
	From TipoSubscrição join Subriscrição on
TipoSubscrição.TSID=Subriscrição.TipoSubscriçãoTSID
                                        join Tutor on
Subriscrição.SID=Tutor.SubriscriçãoSID
                                        join Criança on Tutor.TID=Criança.TutorTID
	Group by TipoSubscrição.DescTS
--8 Qual a escola com mais crianças inscritas na aplicação?
	Select Escola.NomeEscola as Escola, count(Criança.CID) as Inscritos
	From Escola join Criança on Escola.EID=Criança.EscolaEID
	group by Escola.NomeEscola
	Having Count(CID) >= All (Select Count(CID)
		From Escola Group by Escola.NomeEscola)
--9 Qual o método de pagamento mais usado?
	Select TipoPagamento.DescTP as Tipo, COUNT(Pagamento.TipoPagamentoTPID) as
'Nº de Vezes'
	From Pagamento join TipoPagamento on
TipoPagamento.TPID=Pagamento.TipoPagamentoTPID
	Group by TipoPagamento.DescTP
	Having COUNT(Pagamento.TipoPagamentoTPID) >= All (Select
COUNT(Pagamento.TipoPagamentoTPID) as 'Nº de Vezes'
```

```
            From Pagamento join TipoPagamento on
Pagamento.TipoPagamentoTPID=TipoPagamento.TPID
            Group by TipoPagamento.DescTP)

--10 Qual média de consultas realizadas por ano?
       Select AVG(Consultas) From(
       Select YEAR(Consulta.DataRealização) as Ano, CID as Consultas
       From Consulta
       Group by YEAR(Consulta.DataRealização), CID) Tabela
```

Implementação da aplicação

Introdução

Para o desenvolvimento da aplicação proposto, optámos pela criação da mesma utilizando o WindowsForms em c#, interligando-a a uma base de dados em sql através de stored procedures. Sendo assim, utilizamos a extensão dapper para ser possível a interligação entre a interface em c# e as stored procedures em sql.

Classes

No desenvolvimento da aplicação, começamos por criar algumas das classes necessárias de acordo com a nossa base de dados:

Criança

```
3 references
public class Crianca
{
    #region
    1 reference
    public int TutorTID { get; set; }
    1 reference
    public int EscolaEID { get; set; }
    1 reference
    public int PsicólogoPID { get; set; }
    1 reference
    public string Nome { get; set; }
    1 reference
    public char Sexo { get; set; }
    1 reference
    public DateTime DataNascimento { get; set; }

    #endregion
}
```

Psicólogo

```
3 references
public class Psicologo
{
    0 references
    public int PID { get; set; }
    1 reference
    public int EquipaGestaoEGID { get; set; }
    1 reference
    public string Nome { get; set; }
}
```

Consulta

```
7 references
public class Consulta
{
    2 references
    public int PsicologoPID { get; set; }
    2 references
    public int CriancaCID { get; set; }
    2 references
    public DateTime DataRealizacao { get; set; }

    0 references
    public string FullInfo
    {
        get
        {
            return $"{PsicologoPID} {CriancaCID} {DataRealizacao}";
        }
    }
}
```

Conexão à base de dados

Para conseguirmos que a aplicação conecte à base de dados criámos a classe Helper. Dentro desta classe foi criado o método CnnVal que utiliza uma connection string declarada no app.config apara se conectar à base de dados sempre que é chamada.

```
10 references
public static class Helper
{
    10 references
    public static string CnnVal(string name)
    {
        return ConfigurationManager.ConnectionStrings[name].ConnectionString;
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <connectionStrings>
    <add name="C-Happy" connectionString="Server=.;Database=C-Happy;Trusted_Connection=True;" providerName="System.Data.SqlClient"/>
  </connectionStrings>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
    </startup>
</configuration>
```

Métodos e Stored Procedures

A primeira classe criada neste tópico, foi a classe DataAccess onde implementamos todos os métodos necessários ao funcionamento da aplicação e conectámos às respetivas stored procedures.

As seguintes listas foram criadas para tornar possível a inserção e alteração na base de dados.

```
20 references
public class DataAccess
{
    List<Crianca> criancas = new List<Crianca>();
    List<Psicologo> psicologos = new List<Psicologo>();
    List<Consulta> consultas = new List<Consulta>();
```

Query

O seguinte método executa uma query como a que está apresentada dentro de parênteses.

```
1 reference
public List<Consulta> QueryMedia(int CID, int PID)
{
    using (IDbConnection connection = new System.Data.SqlClient.SqlConnection(Helper.CnnVal("C-Happy")))
    {
        return connection.Query<Consulta>($"Select * From Consulta where PsicólogoPID = {PID} and CriançaCID = {CID}").ToList();
    }
}
```

Introdução de dados

Para a introdução de dados em cada uma das classes apresentadas previamente, foram utilizados vários métodos com a mesma estrutura.

```
1 reference
public void InsertCrianca(int text2, int text3, int text4, string text5, char text6, DateTime text7)
{
    using (IDbConnection connection = new System.Data.SqlClient.SqlConnection(Helper.CnnVal("C-Happy")))
    {

        criancas.Add(new Crianca { TutorTID = text2, EscolaEID = text3, PsicólogoPID = text4, Nome = text5, Sexo = text6, DataNascimento = text7 });

        connection.Execute("dbo.ADDCrianca @TutorTID, @EscolaEID, @PsicólogoPID, @Nome, @Sexo, @DataNascimento", criancas);

    }
}
```

Na introdução de dados utilizamos as seguintes stored procedures.

```
ALTER PROCEDURE [dbo].[ADDPsicologo]

    @EquipaGestaoEGID int,
    @Nome nvarchar(255)
AS
BEGIN

    SET NOCOUNT ON;

    insert into Psícólogo(EquipaGestãoEGID, Nome)
    values(@EquipaGestaoEGID, @Nome)


END
```

```
ALTER PROCEDURE [dbo].[ADDCrianca]

    @TutorTID int,
    @EscolaEID int,
    @PiscologoPID int,
    @Nome nvarchar(255),
    @Sexo char,
    @DataNascimento datetime
AS
BEGIN

    SET NOCOUNT ON;

    insert into Criança(TutorTID, EscolaEID, PsicólogoPID, Nome, Sexo, DataNascimento)
    values(@TutorTID, @EscolaEID, @PiscologoPID, @Nome, @Sexo, @DataNascimento)


END
```

```
ALTER PROCEDURE [dbo].[ADDConsulta]

    @PsicologoPID int,
    @CriancaCID int,
    @DataRealizacao datetime
AS
BEGIN

    SET NOCOUNT ON;

    insert into Consulta(PsicólogoPID, CriançaCID, DataRealização)
    values(@PsicologoPID, @CriancaCID, @DataRealizacao)


END
```

Alteração de dados

Para a alteração de dados em cada uma das classes apresentadas previamente, foram utilizados vários métodos com a mesma estrutura.

```
1 reference
public void UpdateCrianca (int text1, int text2, int text3, int text4, string text5, char text6, DateTime text7)
{
    using (IDbConnection connection = new System.Data.SqlClient.SqlConnection(Helper.CnnVal("C-Happy")))
    {
        connection.Execute("dbo.UPDATECrianca @CID, @TutorTID, @EscolaEID, @PsicologoPID, @Nome, @Sexo, @DataNascimento", new
        {
            CID=text1,
            TutorTID = text2,
            EscolaEID = text3,
            PsicologoPID = text4,
            Nome = text5,
            Sexo = text6,
            DataNascimento = text7
        });
    }
}
```

Na alteração de dados utilizamos as seguintes stored procedures.

```
ALTER PROCEDURE [dbo].[UPDATECrianca]
    @CID int,
    @TutorTID int,
    @EscolaEID int,
    @PiscologoPID int,
    @Nome nvarchar(255),
    @Sexo char,
    @DataNascimento datetime
AS
begin
    update dbo.[Criança]
    set
    TutorTID = @TutorTID, EscolaEID = @EscolaEID, PsícólogoPID = @PiscologoPID, Nome = @Nome, Sexo = @Sexo, DataNascimento = @DataNascimento
    where CID = @CID;
    end
```

```
ALTER PROCEDURE [dbo].[UPDATEPsicologo]
    @PID int,
    @EquipaGestãoEGID int,
    @Nome nvarchar(50)
AS
begin
    update dbo. [Psícólogo]
    set
    EquipaGestãoEGID=@EquipaGestãoEGID, Nome = @Nome
    where PID = @PID;
    end
```

```
ALTER PROCEDURE [dbo].[UPDATEConsulta]
    @CID int,
    @PsicologoPID int,
    @CriancaCID int,
    @DataRealizacao datetime
AS
begin
    update dbo.[Consulta]
    set
    PsícólogoPID=@PsicologoPID, CriançaCID = @CriancaCID, DataRealização = @DataRealizacao
    where CID = @CID;
    end
```

Remoção de dados

Para a remoção de dados em cada uma das classes apresentadas previamente, foram utilizados vários métodos com a mesma estrutura.

```
1 reference
public void DeleteCrianca (int ID)
{
    using (IDbConnection connection = new System.Data.SqlClient.SqlConnection(Helper.CnnVal("C-Happy")))
    {
        connection.Execute($"dbo.DELETECrianca @CID", new { CID = ID });
    }
}
```

Na remoção de dados utilizamos as seguintes stored procedures.

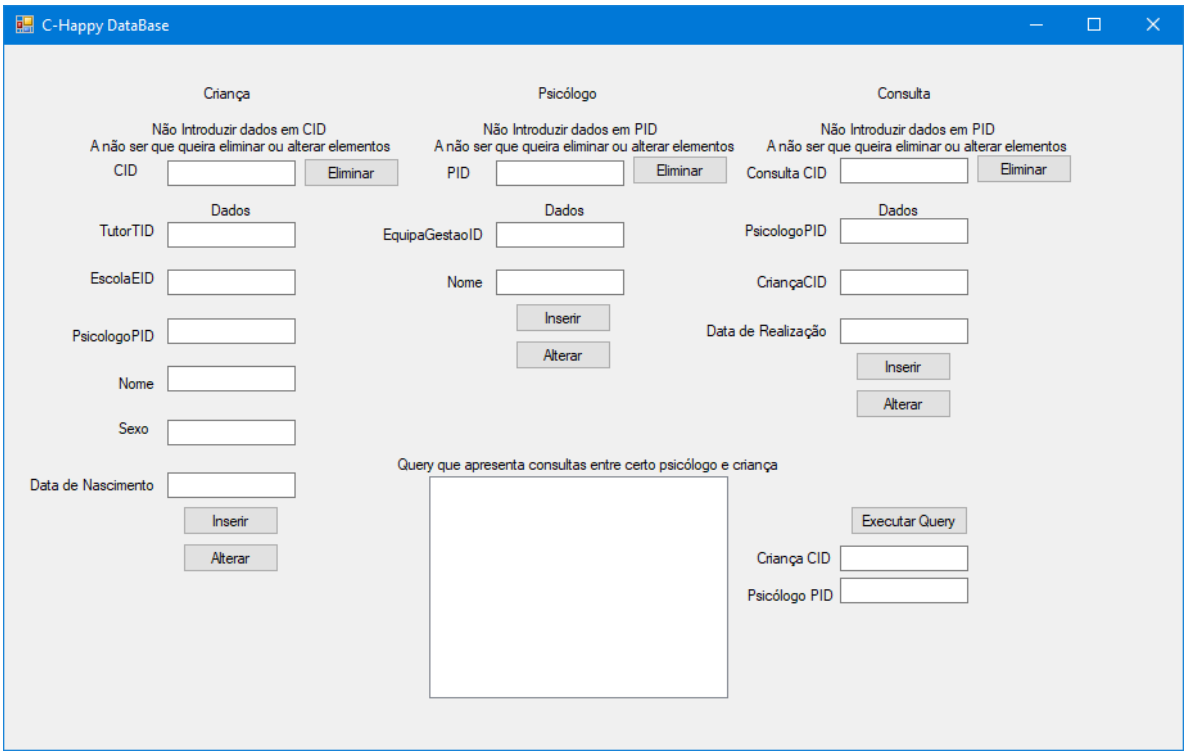
```
ALTER PROCEDURE [dbo].[DELETEConsulta]
    @CID int
AS
BEGIN
    Delete
    from dbo.[Consulta]
    where Consulta.CID=@CID
END
```

```
ALTER PROCEDURE [dbo].[DELETECrianca]
    @CID int
AS
BEGIN
    Delete
    from dbo.[Criança]
    where Criança.CID=@CID
END
```

```
ALTER PROCEDURE [dbo].[DELETEPsicologo]
    @PID int
AS
BEGIN
    Delete
    from dbo.[Psícólogo]
    where Psícólogo.PID=@PID
END
```

Interface da aplicação

A seguir segue o interface desenvolvido para a aplicação.



O botão inserir que se encontra na interface, representa o método de introdução de dados que é chamado no seguinte excerto de código. Aquando da inserção de dados pela interface não será necessário introduzir a chave primária, dado que isso é feito pela própria base de dados.

Os restantes botões inserir seguem a mesma estrutura

```
1 reference
private void BotaoInserir_Click(object sender, EventArgs e)
{
    DataAccess db = new DataAccess();

    string targetDateFormat = "dd/MM/yyyy";

    DateTime dt = DateTime.ParseExact(DataNascimentotext.Text, targetDateFormat, CultureInfo.InvariantCulture);

    db.InsertCrianca(Convert.ToInt32(TutorTIDtext.Text), Convert.ToInt32(EscolaEIDtext.Text), Convert.ToInt32(PsicologoPIDtext.Text), Nometext.Text, Convert.ToChar(Sexotext.Text), dt)


    TutorTIDtext.Text = "";
    EscolaEIDtext.Text = "";
    PsicologoPIDtext.Text = "";
    Nometext.Text = "";
    Sexotext.Text = "";
    DataNascimentotext.Text = "";
}
```

O botão eliminar, representa o método de eliminação de dados. Aqui basta inserir a chave primária do elemento a eliminar e clicar no botão.

```
1 reference
private void EliminarButton_Click(object sender, EventArgs e)
{
    DataAccess db = new DataAccess();

    db.DeleteCrianca(Convert.ToInt32(CIDtext.Text));

    CIDtext.Text = "";
}
```

O botão alterar, representa o método de alteração de dados. Neste botão será necessário introduzir a chave primária de maneira a encontrar que dados o utilizador quer alterar.

```
1 reference
private void AlterarPsicologo_Click(object sender, EventArgs e)
{
    DataAccess db = new DataAccess();

    db.UpdatePsicologo(Convert.ToInt32(PIDtext.Text), Convert.ToInt32(EquipaEIDtext.Text), NomePsitext.Text);

    PIDtext.Text = "";
    EquipaEIDtext.Text = "";
    NomePsitext.Text = "";
}
```

O botão Executar Query serve para executar um método da query apresentada anteriormente na listbox de acordo com CID e o PID escolhidos.

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    DataAccess db = new DataAccess();

    listBox1.DataSource = db.QueryConsultas(Convert.ToInt32(CIDqueryText.Text), Convert.ToInt32(PIDqueryText.Text));

    UpdateBinding();
}
```

Bibliografia

https://www.youtube.com/watch?v=Et2khGnrIqc

https://dapper-tutorial.net/

https://csharp-station.com/Tutorial/AdoDotNet/Lesson02

https://www.dotnetperls.com/sqlconnection