

## Projeto – Aldeias e locais históricos de Portugal

Criação e edição de dados de um local.

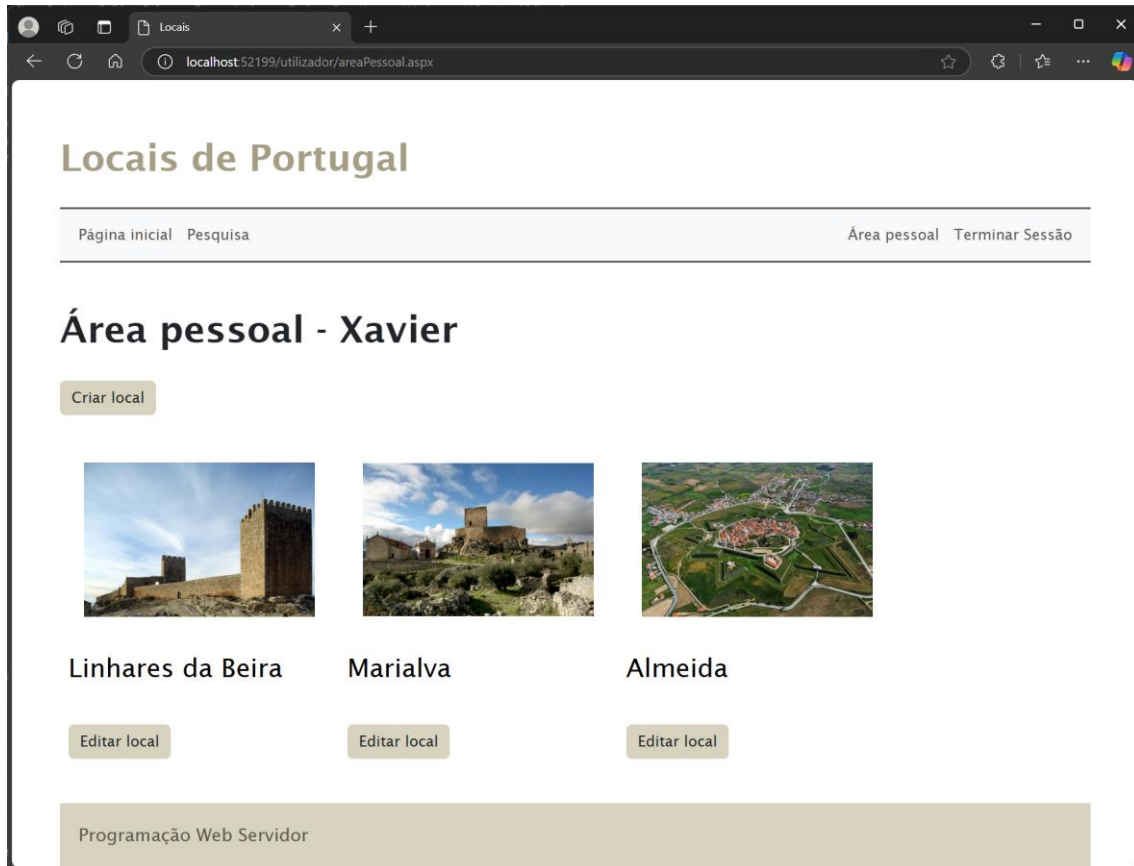
Adicione ao Web Form **areaPessoal.aspx** um controlo do tipo DataList; este controlo deverá mostrar a lista de locais criados pelo utilizador;

Crie na base de dados **Locais** o seguinte **Stored Procedure**:

```
CREATE PROCEDURE GetLocaisByUtilizador
    @utilizador NVARCHAR(50)
AS
    WITH FotosOrdenadas AS (
    SELECT
        f.id AS FotoID,
        f.ficheiro,
        f.local,
        ROW_NUMBER() OVER (PARTITION BY f.local ORDER BY f.id) AS RN
    FROM Foto f
    )
    SELECT
        l.id AS LocalID,
        l.nome AS NomeLocal,
        fo.ficheiro AS PrimeiraFoto
    FROM local l
    LEFT JOIN FotosOrdenadas fo
        ON l.id = fo.local AND fo.RN = 1
    WHERE l.Utilizador = @utilizador
GO
```

O procedimento recebe o ID do utilizador (que se encontra na variável de sessão

`Session["utilizador"]` criada no evento **LoggedIn** quando do início de sessão;



Para a definição do DataList, considere:

```
<asp:DataList runat="server" ID="listLocais" RepeatColumns="3" RepeatDirection="Horizontal">
  <ItemTemplate>
    <table class="table table-borderless">
      <tr style="height: 220px; vertical-align: middle;">
        <td style="width: 300px; text-align: center;">
          <img src='../<# Eval("Foto") %>' alt='<# Eval("Nome") %>'
            style="width: 250px;" />
        </td>
      </tr>
      <tr>
        <td>
          <asp:Label Text='<# Eval("Nome") %>' runat="server" CssClass="fs-3" />
        </td>
      </tr>
      <tr>
        <td>
          <asp:LinkButton ID="lnkDetalhes" runat="server"
            CommandArgument='<# Eval("ID") %>'
            OnCommand="lnkDetalhes_Command"
            CssClass="btn mt-4" BackColor="#D7D3BF">
            Editar local
          </asp:LinkButton>
        </td>
      </tr>
    </table>
  </ItemTemplate>
</asp:DataList>
```

Crie um procedimento que efetue o preenchimento do controlo DataList, utilizando o Stored Procedure definido;

```
private void CarregarLocais()
{
    SqlConnection connection = new SqlConnection
    (@"data source=.\SqlExpress; initial catalog = Locais; integrated security = true;");

    SqlCommand command = new SqlCommand();
    command.Connection = connection;
    //definição do Stored Procedure a executar
    command.CommandText = "GetLocaisByUtilizador";
    command.Parameters.AddWithValue("@utilizador", Session["id_utilizador"].ToString());
    connection.Open();
    SqlDataReader reader = command.ExecuteReader();
    DataTable table = new DataTable();
    table.Load(reader);
    reader.Close();
    connection.Close();

    //mostrar dados no controlo DataList
    listLocais.DataSource = table;
    listLocais.DataBind();
}
```

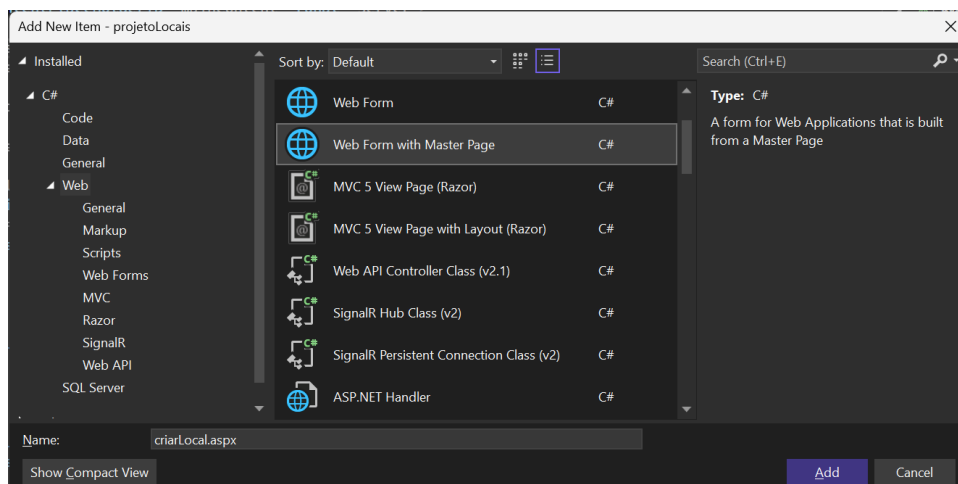
Execute o procedimento definido no evento Load do formulário **areaPessoal**;

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        CarregarLocais();
    }
}
```

O controlo **Criar local** deverá redirecionar o utilizador para a o formulário **criarLocal.aspx**;

```
protected void buttonCriarLocal_Click(object sender, EventArgs e)
{
    Response.Redirect("criarLocal.aspx");
}
```

Adicione à pasta utilizador um Web Form com o nome **criarLocal.aspx**;



Adicione ao formulário **criarLocal.aspx** controlos que permitam definir o nome, a descrição, a morada (se aplicável), a localidade e o distrito do local; a latitude e longitude poderão ser obtidas a partir da API <https://positionstack.com/>; considere ainda a possibilidade de definir as fotos do local; será necessário um controlo do tipo **FileUpload** que permita copiar para o servidor o ficheiro selecionado; na tabela **Foto** deverá ser guardado o nome do ficheiro e uma legenda (opcional); considere a utilização de um **DataList** que permita ver as fotos associadas ao local e, se necessário, a modificação da legenda;

Locais

localhost:52199/utilizador/criarLocal.aspx

# Locais de Portugal

Página inicial

Pesquisa

Área pessoal

Terminar Sessão

## Criar local

Nome

Descrição

Morada

Localidade

Distrito

Angra do Heroísmo

Concelho

Angra do Heroísmo

Guardar

Cancelar

## Fotos do local

Selecionar foto

Escolher Ficheiro

Não foi escolhido nenhum ficheiro

Legenda

Guardar foto

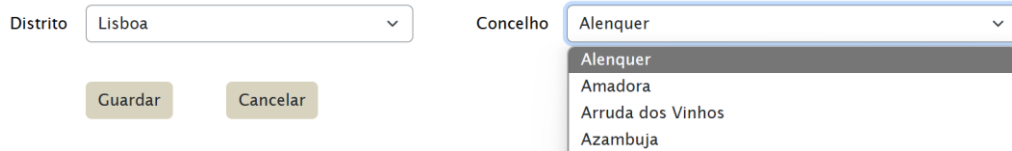
Editar legenda

Eliminar foto

Cancelar foto

Programação Web Servidor

Para a definição do distrito e concelho considere a utilização de controlos DropDownList, a preencher com a informação das respetivas tabelas da base de dados; nota – a lista de concelhos deverá ser preenchida em função do distrito selecionado (considere a utilização de controlos **SqlDataSource**);



Quando o formulário é apresentado, o controlo **Guardar foto** deverá estar desativado (**Enabled="false"**); só deverá ficar ativo após a criação do local (momento em que é gerado o ID do local, a utilizar como valor de inserção no campo **Foto.Local**);

Defina, ao nível da classe, a conexão à base de dados:

```
public partial class criarLocal : System.Web.UI.Page
{
    SqlConnection connection = new SqlConnection
        ("data source=.\SqlExpress; initial catalog = Locais; integrated security = true;");
}
```

O controlo **Guardar** deverá inserir na tabela Local o nome, descrição, morada (se aplicável), localidade, concelho e o ID do utilizador que criou o local (opcionalmente, a latitude e a longitude); após a inserção, na tabela, o botão **Guardar Foto** deverá ficar ativo desativado (**Enabled="true"**) permitindo a definição de fotos para o local; comece por criar o Stored Procedure que fará o INSERT na tabela:

```
CREATE PROCEDURE LocalCriar
(
    @nome          NVARCHAR(300),
    @descricao     NVARCHAR(4000),
    @morada        NVARCHAR(350),
    @localidade    NVARCHAR(350),
    @concelho      INT,
    @latitude      VARCHAR(20),
    @longitude     VARCHAR(20),
    @utilizador    NVARCHAR(50)
)
AS
    INSERT Local(Nome, Descricao, Morada, Localidade, Concelho,
        Latitude, Longitude, Utilizador)
    VALUES (@nome, @descricao, @morada, @localidade, @concelho,
        @latitude, @longitude, @utilizador)
    SELECT IDENT_CURRENT('Local')
GO
```

A instrução `SELECT IDENT_CURRENT('Local')` obtém o ID atribuído ao novo local, necessário para a definição das respetivas fotos; o valor deverá ser mantido em **ViewState**, de forma a salvaguardar a informação entre pedidos (Postback);

No evento Click do controlo Guardar execute o Stored Procedure LocalCriar (definindo os valores para parâmetros do procedimento);

```
SqlCommand command = new SqlCommand();
command.Connection = connection;
//definição do Stored Procedure que cria o registo na base de dados
command.CommandText = "LocalCriar";
command.CommandType = CommandType.StoredProcedure;
//definição dos valores a inserir na tabela
command.Parameters.AddWithValue("@nome", textNome.Text);
command.Parameters.AddWithValue("@descricao", textDescricao.Text);
if (textMorada.Text == "")
    command.Parameters.AddWithValue("@morada", DBNull.Value);
else
    command.Parameters.AddWithValue("@morada", textMorada.Text);
command.Parameters.AddWithValue("@localidade", textLocalidade.Text);
command.Parameters.AddWithValue("@concelho", listConcelho.SelectedValue);
command.Parameters.AddWithValue("@utilizador", Session["id_utilizador"]);

command.Parameters.AddWithValue("@latitude", DBNull.Value);
command.Parameters.AddWithValue("@longitude", DBNull.Value);

//obter latitude e longitude
/*
Datum localizacao = new Datum();

string key = "chave de acesso à API";
string local = $"{textLocalidade.Text},Portugal";
WebRequest request = WebRequest.Create

($"https://api.positionstack.com/v1/forward?access_key={key}&query={local}");
WebResponse response = request.GetResponse();
if (response != null)
{
    Stream stream = response.GetResponseStream();
    StreamReader reader = new StreamReader(stream);
    string json = reader.ReadToEnd();

    Root result = JsonConvert.DeserializeObject<Root>(json);

    // obter latitude e longitude
    if (result.data != null && result.data.Count > 0)
    {
        localizacao = result.data[0];
        command.Parameters.AddWithValue("@latitude", localizacao.latitude);
        command.Parameters.AddWithValue("@longitude", localizacao.longitude);
    }
    else
    {
        command.Parameters.AddWithValue("@latitude", DBNull.Value);
        command.Parameters.AddWithValue("@longitude", DBNull.Value);
    }
}
*/
connection.Open();
//guardar em ViewState o ID atribuído ao local
ViewState["idLocal"] = command.ExecuteScalar();
//ativar botão Guardar foto
buttonGuardarFoto.Enabled = true;
connection.Close();
```

Para mostrar a lista de fotos associadas ao local, considere a utilização de um controlo **DataList** como o da configuração abaixo;

```
<asp:DataList runat="server" ID="listFotos" RepeatColumns="2" RepeatDirection="Horizontal">
  <ItemTemplate>
    <table class="table table-borderless">
      <tr style="height: 220px; vertical-align: middle;">
        <td style="width: 450px; text-align: center;">
          <img src='../<%# Eval("Ficheiro") %>'
              alt='<%# Eval("Legenda") %>' style="width: 350px;" />
        </td>
      </tr>
      <tr>
        <td>
          <asp:Label Text='<%# Eval("Legenda") %>' runat="server" CssClass="fs-3" />
        </td>
      </tr>
      <tr>
        <td>
          <!-- controlo que permite seleccionar uma foto para
              editar a legenda ou para remover-->
          <asp:LinkButton ID="lnkDetalhes" runat="server"
              CommandArgument='<%# Eval("ID") %>' OnCommand="lnkDetalhes_Command"
              CssClass="btn mt-4" BackColor="#D7D3BF">Selecionar</asp:LinkButton>
        </td>
      </tr>
    </table>
  </ItemTemplate>
</asp:DataList>
```

Para o preenchimento do controlo DataList, considere a função abaixo; a função passa na query o valor referente ao ID do local que se encontra na variável ViewState["idLocal"];

```
void GetFotosLocal() {
    SqlCommand command = new SqlCommand();
    command.Connection = connection;
    command.CommandText =
        "SELECT Id, Ficheiro, Legenda FROM Foto WHERE Local = @local";

    //ID do local, definido aquando da criação do local
    command.Parameters.AddWithValue("@local", ViewState["idLocal"]);

    connection.Open();
    SqlDataReader reader = command.ExecuteReader();
    DataTable table = new DataTable();
    table.Load(reader);
    reader.Close();
    connection.Close();

    listFotos.DataSource = table;
    listFotos.DataBind();
}
```

O controlo **Guardar foto** deverá fazer o upload do ficheiro selecionado para a pasta **imagens** e colocar, com a legenda, a informação na tabela **Foto**; crie o seguinte procedimento na base de dados:

```
CREATE PROCEDURE LocalFotoCriar
(
    @legenda          NVARCHAR(300),
    @ficheiro          NVARCHAR(300),
    @local             INT
)
AS
    INSERT Foto(Legenda, Ficheiro, Local)
    VALUES (@legenda, @ficheiro, @local)
GO
```



A ação **Guardar foto** só poderá acontecer se o utilizador definiu um ficheiro e se o ficheiro selecionado for do tipo jpg, jpeg, png, gif ou tiff; se estas condições não se verificarem, o upload não deve acontecer;

```
//ficheiro - nome do controlo FileUpload
if (ficheiro.HasFile)
{
    SqlCommand command = new SqlCommand();
    command.Connection = connection;
    command.CommandText = "LocalFotoCriar";
    command.CommandType = CommandType.StoredProcedure;

    command.Parameters.AddWithValue("@local", ViewState["idLocal"]);
    command.Parameters.AddWithValue("@legenda", textLegenda.Text);

    //tipos de ficheiros admitidos
    string[] ext = { ".jpg", ".jpeg", ".png", ".gif", ".tiff" };

    bool ok = false;
    //obter extensão do ficheiro
    string extensao = System.IO.Path.GetExtension(ficheiro.FileName).ToString();
    //verificar se a extensão se encontra no Array de ficheiros admitidos
    foreach (string item in ext)
        if (extensao == item)
            ok = true;
    //se o tipo de ficheiro está correto
    if (ok)
    {
        //gerar Guid para evitar nomes repetidos
        Guid g = Guid.NewGuid();
        string fileName = $"{g}-{ficheiro.FileName}";
        ficheiro.SaveAs(Server.MapPath("../imagens/") + fileName);

        //definir parâmetro
        command.Parameters.AddWithValue("@ficheiro", "imagens/" + fileName);

        //tipo de ficheiro correto - colocar informação na base de dados
        connection.Open();
        command.ExecuteNonQuery();
        connection.Close();
        //atualizar DataList referente às fotos
        GetFotosLocal();
    }
    else
    {
        //ficheiro inválido - cancelar execução do procedimento
        Response.Write("<script>alert('Selecione um ficheiro do tipo \".jpg\", \".jpeg\", \"\n\".png\", \".gif\" ou \".tiff.');</script>");
        return;
    }
}
else
{
    //não foi selecionado um ficheiro - cancelar execução do procedimento
    Response.Write("<script>alert('Selecione um ficheiro do tipo \".jpg\", \".jpeg\", \"\n\".png\", \".gif\" ou \".tiff.');</script>");
    return;
}
```

A opção Selecionar permite selecionar uma foto do local; coloca em ViewState o ID da foto;



```
protected void lnkDetalhes_Command(object sender,
    System.Web.UI.WebControls.CommandEventArgs e)
{
    if (e.CommandArgument != null)
    {
        //colocar em ViewState o Id da foto selecionada
        ViewState["idFoto"] = e.CommandArgument.ToString();

        //selecionar legenda da foto selecionada
        SqlCommand commandFoto = new SqlCommand();
        commandFoto.Connection = connection;
        commandFoto.CommandText = "SELECT Legenda FROM Foto WHERE Id = @id";

        commandFoto.Parameters.AddWithValue("@id", ViewState["idFoto"]);

        connection.Open();
        SqlDataReader reader = commandFoto.ExecuteReader();
        while (reader.Read())
        {
            textLegenda.Text = reader[0].ToString();
        }
        reader.Close();
    }
}
```

O controlo **Editar legenda** deve editar o valor da legenda da foto selecionada; depois de alterada a legenda na base de dados, deve ser atualizada a DataList;

```
protected void buttonEditarLegenda_Click(object sender, EventArgs e)
{
    SqlCommand commandFoto = new SqlCommand();
    commandFoto.Connection = connection;
    commandFoto.CommandText = "UPDATE Foto SET Legenda = @legenda WHERE Id = @id";
    commandFoto.CommandType = CommandType.Text;
    //ID da foto definido quando a foto é selecionada
    commandFoto.Parameters.AddWithValue("@id", ViewState["idFoto"]);
    commandFoto.Parameters.AddWithValue("@legenda", textLegenda.Text);
    connection.Open();
    commandFoto.ExecuteNonQuery();
    connection.Close();

    textLegenda.Text = string.Empty;

    //atualizar DataList fotos do local
    GetFotosLocal();
}
```

O controlo **Eliminar foto** deve eliminar a foto selecionada; é necessário remover a informação da tabela **Foto** e eliminar o ficheiro; depois de eliminada a foto, deve ser atualizada a DataList;

```
protected void buttonEliminarFoto_Click(object sender, EventArgs e)
{
    //eliminar ficheiro
    SqlCommand commandFoto = new SqlCommand();
    commandFoto.Connection = connection;
    commandFoto.CommandText = "SELECT Ficheiro FROM Foto WHERE Id = @id";

    //ID da foto definido quando a foto é selecionada
    commandFoto.Parameters.AddWithValue("@id", ViewState["idFoto"]);

    connection.Open();
    //obter nome do ficheiro a eliminar
    SqlDataReader reader = commandFoto.ExecuteReader();
    while (reader.Read())
    {
        string ficheiro = Server.MapPath("../" + reader[0].ToString());
        //eliminar ficheiro
        if (File.Exists(ficheiro))
            File.Delete(ficheiro);
    }
    reader.Close();

    //eliminar dados na tabela
    commandFoto.Parameters.Clear();

    commandFoto.CommandText = "DELETE Foto WHERE Id = @id";
    commandFoto.CommandType = CommandType.Text;
    commandFoto.Parameters.AddWithValue("@id", ViewState["idFoto"]);
    commandFoto.ExecuteNonQuery();

    connection.Close();

    textLegenda.Text = string.Empty;
    //atualizar DataList fotos do local
    GetFotosLocal();
}
```

O controlo **Cancelar** (referente à definição de fotos do local) deve limpar os controlos referentes ao ficheiro e respetiva legenda;

```
protected void buttonCancelarFoto_Click(object sender, EventArgs e)
{
    textLegenda.Text = string.Empty;
    //o ficheiro selecionado é removido -
    //o valor do FileUpload não é mantido em ViewState
}
```

O controlo **Cancelar** (referente ao local) deve limpar todos os controlos do formulário e remover das tabelas **Foto** e **Local** a informação já definida; deve ainda eliminar os ficheiros da pasta **imagens**;  
Comece por criar o Stored Procedure abaixo, que recebe o ID do local e começa por eliminar a informação na tabela **Foto** e de seguida na tabela **Local** (para salvaguardar a questão da integridade referencial);

```
CREATE PROCEDURE LocalEliminar
    @idLocal INT
AS
    DELETE FOTO WHERE Foto.Local = @idLocal
    DELETE Local WHERE Local.Id = @idLocal
GO
```

No evento **Click** do controlo **Cancelar**, elimine os ficheiros e a informação das tabelas referentes ao local;

```
protected void buttonCancelar_Click(object sender, EventArgs e)
{
    //obter nome dos ficheiros associados ao local
    SqlCommand commandFotos = new SqlCommand();
    commandFotos.Connection = connection;
    commandFotos.CommandText = "SELECT Ficheiro FROM Foto WHERE Local = @local";

    commandFotos.Parameters.AddWithValue("@local", ViewState["idLocal"]);

    connection.Open();
    SqlDataReader reader = commandFotos.ExecuteReader();
    while (reader.Read())
    {
        string ficheiro = Server.MapPath("../" + reader[0].ToString());
        //eliminar ficheiros
        if(File.Exists(ficheiro))
            File.Delete(ficheiro);
    }

    reader.Close();

    //eliminar dados das tabelas

    SqlCommand commandLocal = new SqlCommand();
    commandLocal.Connection = connection;
    commandLocal.CommandText = "LocalEliminar";
    commandLocal.CommandType = CommandType.StoredProcedure;

    commandLocal.Parameters.AddWithValue("@local", ViewState["idLocal"]);

    commandLocal.ExecuteNonQuery();

    connection.Close();

    ViewState["idLocal"] = null;
}
```