Dominando o RTTI do Delphi: do básico a técnicas avançadas

{ Palestrante

GUSTAVO MENA BARRETO

Agenda

- RTTI
- Exemplos Básicos
- Serialização
- Custom Attributes
- Criação de Forms em runtime
- Dicas Finais

Um pouco sobre mim

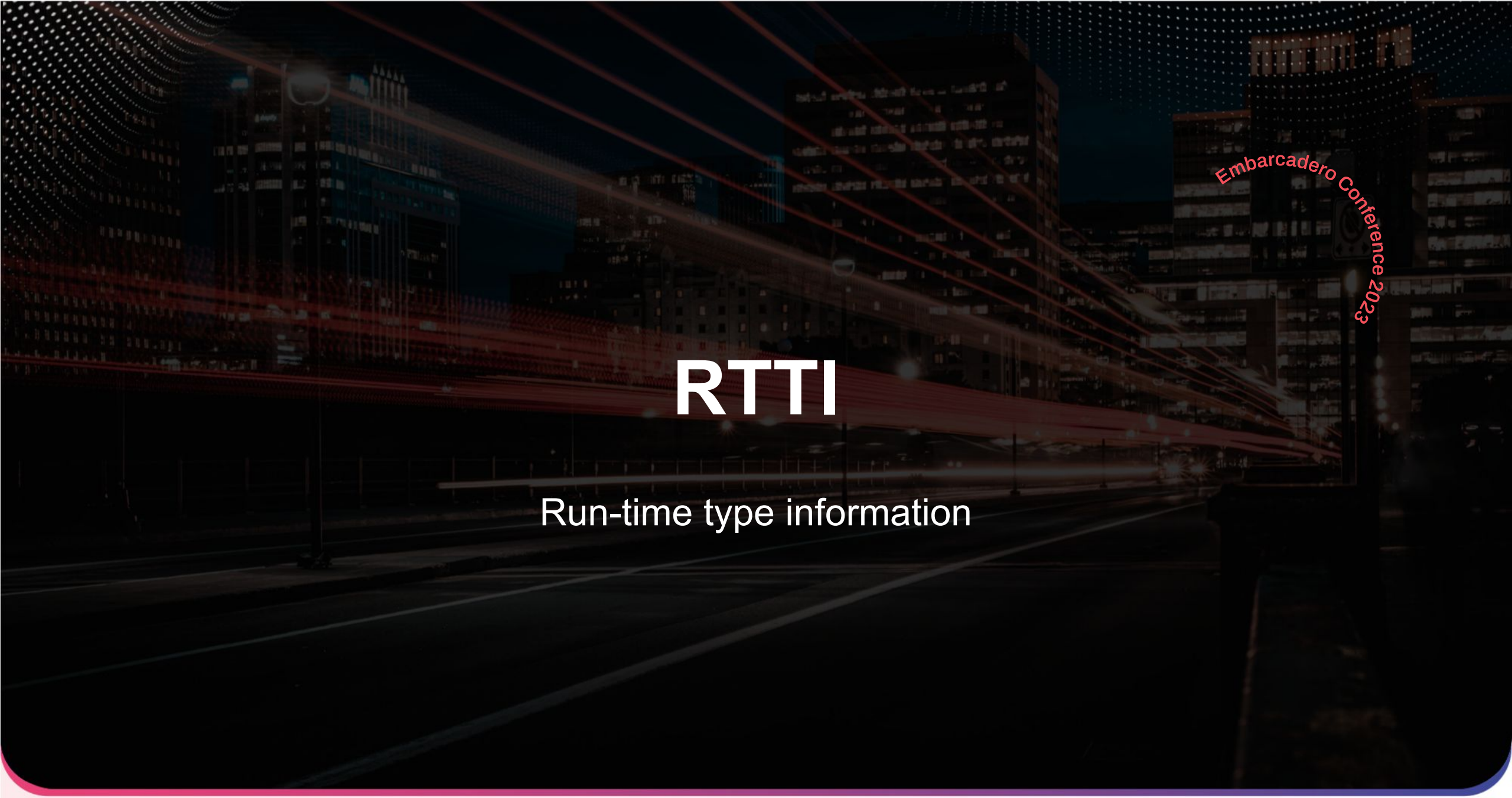
Desenvolvedor Sênior na Aquasoft.

Formado em Análise e Desenvolvimento de Sistemas pela Unisinos/RS.

Mais de 10 anos de experiência com Delphi.

Palestrante Embarcadero Conference e DelphiCon.

Criador de conteúdo no Instagram - @forcecoding



RTTI

Run-time type information

RTTI - Run-time type information

- Funcionalidade no Delphi que permite obter informações sobre os tipos de dados e estruturas utilizados em seu programa durante a execução.

- Possibilita a inspeção e manipulação das propriedades, métodos e campos de objetos e classes sem necessariamente conhecer seus tipos exatos durante o tempo de compilação.

RTTI - Run-time type information

- **TypeInfo**: tipo especial gerado pelo compilador que fornece informações sobre nome, tamanho e estrutura de um tipo de dado em tempo de execução.
- **TRttiContext**: classe usada para obter informações sobre tipos, campos, propriedades e métodos de classes e registros.
- **TRttiType**: classe que fornece acesso a informações como o nome do tipo, tamanho e tipo (por exemplo, tkClass para classes, tkRecord para registros).

RTTI - Run-time type information

- **TRttiField** e **TRttiProperty**: classes que permitem acessar informações sobre campos e propriedades, respectivamente, de uma classe ou registro. Usada para ler e preencher valores ou obter informações sobre tipos de dados.
- **TRttiMethod**: classe que fornece informações sobre os métodos de uma classe, incluindo seus nomes, parâmetros, tipos de retorno e visibilidade.
- Classes estão na unit **RTTI**.


Embarcadero Conference 2023
EXEMPLOS BÁSICOS

```
procedure TFormRTII.Button1Click(Sender: TObject);
begin
  if (Sender is TButton) then
  begin
    (Sender as TButton).Caption := 'RTTI Conference';
  end;
end;
```

- **Alterando o caption de um Button, utilizando Typecast, usando os operadores “is” e “as”**
- **Verificando se Sender é do tipo TButton, ou seja, o objeto que disparou o evento OnClick**

```
procedure TFormRTII.Button2Click(Sender: TObject);
var
   i: Integer;
begin
  for i := 0 to ComponentCount -1 do
  begin
    if (Components[i] is TEdit) then
      (Components[i] as TEdit).Clear;
  end;
end;
```

- **Utilizando um contador que verifica todos os componentes do Form**
- **Utilizando o TypeCast para verificar se o componente é um edit para assim limpar seu conteúdo**

```
type
  TMinhaClasse = class
  private
    FValue: Integer;
  public
    property MinhaProperty: Integer read FValue write FValue;
  end;
```

- **Utilizando RTTI para acessar a property de um objeto da classe**

```
procedure TFormRTTI.Button3Click(Sender: TObject);
var
  MyObj: TMinhaClasse;
  Ctx: TRttiContext;
  MyProp: TRttiProperty;
begin
  MyObj := TMinhaClasse.Create;
  try
    MyObj.MinhaProperty := 2023;

    Ctx := TRttiContext.Create;
    try
      MyProp := Ctx.GetType(TMinhaClasse).GetProperty('MinhaProperty');
      if Assigned(MyProp) then
      begin
        ShowMessage('Nome Property: ' + MyProp.Name);
        ShowMessage('Property Value: ' + MyProp.GetValue(MyObj).ToString);
      end;
    finally
      Ctx.Free;
    end;
  finally
    MyObj.Free;
  end;
```

- **Criando uma instancia de TMinhaClasse, setando o valor na property e mostrando o resultado em tela**


Embarcadero Conference 2023
SERIALIZAÇÃO

SERIALIZAÇÃO COM RTTI

- Possibilidade de extrair e salvar automaticamente as propriedades de um objeto em um formato de dados (JSON, XML, binário) e vice-versa, restaurando instâncias de objetos a partir de dados serializados.

- Simplifica o processo de serialização, pois não é necessário especificar manualmente cada propriedade para serializar ou desserializar.

```
type
  TPessoa = class
  private
    FNome: string;
    FSobrenome: string;
  public
    property Nome: string read FNome write FNome;
    property SobreNome: string read FSobrenome write FSobrenome;
  end;
```

- **Serialização de deserialização da classe TPessoa**

```
function TFormRTII.SerializarObjeto(obj: TObject): string;
var
  ctx: TRttiContext;
  prop: TRttiProperty;
  jsonObject: TJSONObject;
begin
  ctx := TRttiContext.Create;
  try
    jsonObject := TJSONObject.Create;

    for prop in ctx.GetType(obj.ClassType).GetProperties do
    begin
      jsonObject.AddPair(prop.Name, prop.GetValue(obj).ToString);
    end;

    Result := jsonObject.ToJSON;
  finally
    ctx.Free;
  end;
end;
```

- **Rotina de Serialização**

```
function TFormRTII.DeserializarObjeto(jsonData: string;
  objClass: TClass): TObject;
var
  ctx: TRttiContext;
  prop: TRttiProperty;
  jsonObject: TJSONObject;
  obj: TObject;
  Nome: string;
begin
  ctx := TRttiContext.Create;
  try
    jsonObject := TJSONObject.ParseJSONValue(jsonData) as TJSONObject;
    obj := objClass.Create;

    for prop in ctx.GetType(obj.ClassType).GetProperties do
    begin
      if jsonObject.TryGetValue<string>(prop.Name, Nome) then
      begin
        prop.SetValue(obj, TValue(Nome));
      end;
    end;

    Result := obj;
  finally
    ctx.Free;
  end;
end;
```

- **Rotina de Deserialização**

```
procedure TFormRTTI.Button4Click(Sender: TObject);
var
  Pessoa: TPessoa;
  serializedData: string;
  deserializedPerson: TPessoa;
begin
  Pessoa := TPessoa.Create;
  Pessoa.Nome := 'Gustavo';
  Pessoa.Sobrenome := 'Mena Barreto';

  serializedData := SerializarObjeto(Pessoa);

  deserializedPerson :=
    DeserializarObjeto(serializedData, TPessoa) as TPessoa;

  ShowMessage(deserializedPerson.Nome);
  ShowMessage(deserializedPerson.SobreNome);
end;
```

- **Serialização de deserialização da classe TPessoa**


Embarcadero Conference 2023
CUSTOM ATTRIBUTES

CUSTOM ATTRIBUTES

- São anotações de metadados que você pode associar a vários elementos em seu código, como tipos, campos, propriedades, métodos e até mesmo parâmetros de método. Fornecendo informações adicionais aos elementos de codigos associados.

- Necessário definir uma classe que é herdada de TCustomAttribute, para depois aplicar o seu custom attribute em algum elemento.

CRIAÇÃO DE FORMS EM RUNTIME

PASSOS PARA CRIAR FORMS EM RUNTIME

- **Definir a estrutura do form:** Criar classes ou records que para representar o form. Essa estrutura pode conter properties para representar os componentes como edits, labels, botões.

- **Criar uma function para instanciar e popular o form:** Com essa function utilizando RTTI vamos definir cada propriedade do form e popular o mesmo.

```
TComponentInfo = class
public
  ComponentClass: TComponentClass;
  PropertyName: string;
  Caption: string;
end;

TFormStructure = class
public
  ButtonInfo: TComponentInfo;
  LabelInfo: TComponentInfo;
end;
```

- **Estrutura**

  TComponentInfo representa os componentes do Form.

  TFormStructure representa o form com os componentes visuais.

```
ComponentClassAttribute = class(TCustomAttribute)
private
  FComponentClass: TComponentClass;
  FName: string;
  FCaption: string;
public
  constructor Create(AComponentClass: TComponentClass;AName,ACaption: string);
  property ComponentClass: TComponentClass read FComponentClass;
  property Name: string read FName;
  property Caption: string read FCaption;
end;
```

- **Classe para os Custom Attributes**

```
TFormBase = class
private
  FButtonInfo: TComponentInfo;
  FLabelInfo: TComponentInfo;
public
  [ComponentClass(TButton,'Button1','Clicar aqui')]
  property ButtonInfo: TComponentInfo read FButtonInfo write FButtonInfo;
  [ComponentClass(TLabel, 'Label1', 'Meu Label')]
  property LabelInfo: TComponentInfo read FLabelInfo write FLabelInfo;
end;
```

- **Form base utilizando os custom attributes**

```
var
  MyFormStructure: TFormStructure;
  MyForm: TForm;
  RttiContext: TRttiContext;
  RttiType: TRttiType;
  Prop: TRttiProperty;
  Attr: TCustomAttribute;
begin
  MyFormStructure := TFormStructure.Create;
  RttiContext := TRttiContext.Create;
  try
    RttiType := RttiContext.GetType(TFormBase);
    for Prop in RttiType.GetProperties do
    begin
      for Attr in Prop.GetAttributes do
      begin
        if Attr is ComponentClassAttribute then
        begin
          if ComponentClassAttribute(Attr).ComponentClass = TButton then
          begin
            MyFormStructure.ButtonInfo := TComponentInfo.Create;
            MyFormStructure.ButtonInfo.ComponentClass := ComponentClassAttribute(Attr).ComponentClass;
            MyFormStructure.ButtonInfo.PropertyName := ComponentClassAttribute(Attr).Name;
            MyFormStructure.ButtonInfo.Caption := ComponentClassAttribute(Attr).Caption;
          end;

          if ComponentClassAttribute(Attr).ComponentClass = TLabel then
          begin
            MyFormStructure.LabelInfo := TComponentInfo.Create;
            MyFormStructure.LabelInfo.ComponentClass := ComponentClassAttribute(Attr).ComponentClass;
            MyFormStructure.LabelInfo.PropertyName := ComponentClassAttribute(Attr).Name;
            MyFormStructure.LabelInfo.Caption := ComponentClassAttribute(Attr).Caption;
          end;
        end;
      end;
    end;
   MyForm := CriarPopularForm(MyFormStructure);
   MyForm.Show;
  finally
    RttiContext.Free;
```

```
function TFormRTII.CriarPopularForm(const FormStructure: TFormStructure): TForm;
var
  Form: TForm;
  Button: TButton;
  LabelCtrl: TLabel;
  Ctx: TRttiContext;
  Tp: TRttiType;
  rttMeuField: TRttiField;
begin
  Form := TForm.Create(nil);
  Form.Caption := 'Form Dinamico';
  Form.Width := 400;
  Form.Height := 200;
  Form.Position := poDesktopCenter;

  Ctx := TRttiContext.Create;
  Tp := Ctx.GetType(FormStructure.ClassType);
  try
    rttMeuField := Tp.GetField('ButtonInfo');
    if Assigned(rttMeuField)  then
    begin
      Button := TButton.Create(Form);
      Button.Left:= 2;
      Button.Parent := Form;
      Button.Caption := FormStructure.ButtonInfo.Caption;
    end;

    rttMeuField := Tp.GetField('LabelInfo');
    if Assigned(rttMeuField)  then
    begin
      LabelCtrl := TLabel.Create(Form);
      LabelCtrl.Left := 100;
      LabelCtrl.Parent := Form;
      LabelCtrl.Caption := FormStructure.LabelInfo.Caption;
    end;
  finally
    Ctx.Free;
  end;
  Result := Form;
end;
```


Embarcadero Conference 2023
DICAS FINAIS

Dicas finais

- Avalie a performance para evitar chamadas excessivas de rotinas que usam RTTI
- Ao desenvolver para multiplataforma(Mobile), considere que algumas features podem ser diferentes.
- Mantenha o código legível e bem documentado
- Cuidado ao utilizar RTTI em diferentes versões do Delphi, o comportamento, Features do RTTI pode ser diferente em outras versões do Delphi
- Cuidado ao usar RTTI com Threads. Deve utilizar com Thread Safe, Critical Section por exemplo.

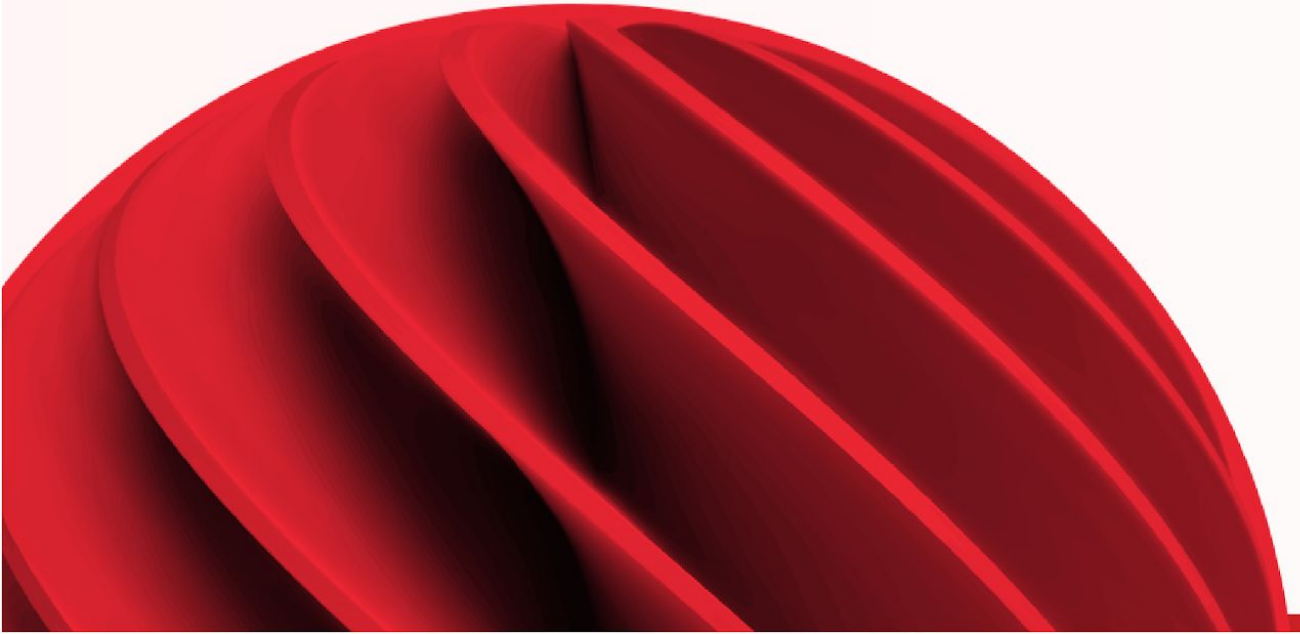
OBRIGADO!

AQUASOFT

Embarcadero Conference 2023

O que você achou da palestra?

Acesse o link do QR Code ao lado e responda a pesquisa.



BRINDES

Embarcadero Conference 2023