



III-T (Segundo Trimestre)

Prof: Raffael Bottoli Schemmer
Disciplina: Programação I
Ano: 2019.
Módulo: III - Trimestre



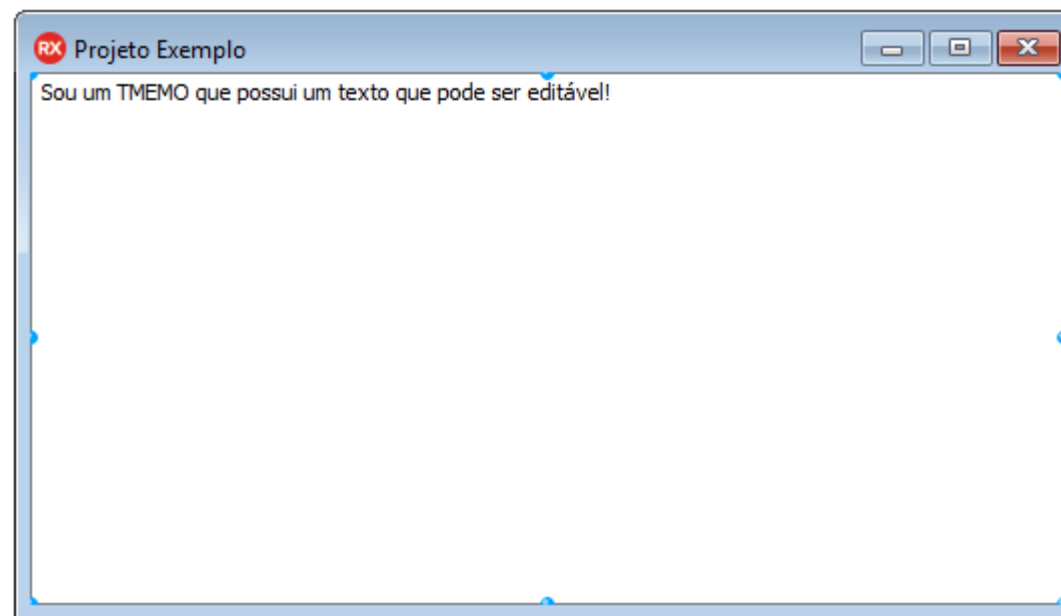
Atividades Trimestrais (III - Trimestre)



M.II-T = TVI (16%) até TXX (16%) + 3 Kahoot (20%)



TMemo (Lista): Definição



Componente `TForm` com um `TMemo`

TMemo (Lista): Definição

- Memo é uma lista de anotações:
 - Você pode utilizar para escrever um longo texto ao usuário.
- Vamos utilizar o TMEMO nos trabalhos futuros:
 - Para explicar ao usuário FAQ como utilizar o programa.
- O TMEMO é muito similar ao TLISTBOX em acesso:
 - Porém sua estrutura pode ser manipulada de forma diferente pelo usuário.



TMemo (Lista): Propriedades

- **Name:** Define o nome do componente na Unit.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (String).
- **CharCase:** Define se o conteúdo será maiúsculo ou minúsculo.
 - O **conteúdo** desta **propriedade** é **uma constante** (ecLowerCase/ecUpperCase).
- **Font:** Define a fonte e o tamanho do texto do botão.
 - Para **estilizar** esta **propriedade** você deve **pressionar** (..).
 - Nesta **propriedade** você pode **estilizar** a **fonte**, o **tamanho** da **fonte** e a **cor** da **fonte**.
- **Enabled:** Define se o botão será habilitado.
 - O conteúdo desta **propriedade** é sempre um **boolean** (true/false).

TMemo (Lista): Propriedades

- **TabOrder**: Define a ordem de chamada do TAB no TMemo.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **Hint**: Define uma dica para o botão.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **ShowHint**: Define se a dica será mostrada ou não.
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).
- **Visible**: Define se o botão será visível ou não.
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).



TMemo (Lista): Propriedades

- **Lines**: Define as linhas do Memo.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **ReadOnly**: Define se as linhas do Memo serão de somente leitura.
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).
- **ScrollBar**: Define barras de rolagem verticais e horizontais para o Memo.
 - O **conteúdo** desta **propriedade** é **uma constante** (**ssHorizontal/ssVertical/ssBoth**).
- **WordWrap**: Define se as linhas terão quebra de linha.
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).

TMemo (Lista): Funções

- **lista.Items.add(item:string)**
 - Permite adicionar um valor string na lista (sempre no final).
- **lista.Items.Insert(indice:integer, item:string):**
 - Permite adicionar um valor em um índice específico.
- **lista.Items.Delete(indice:integer):**
 - Permite deletar um valor em um índice (Integer) específico.
- **lista.Items.Move(PosCorrente:integer, NovaPos:integer):**
 - Move os itens entre posições.

TMemo (Lista): Funções

- **lista.Items.Count:**
 - Retorna a quantidade de itens presentes dentro da lista.
- **lista.Clear:**
 - Limpa toda a lista.
- **lista.Items[l]:**
 - Captura o conteúdo (string) do índice l da lista.

TMemo (Lista): Eventos

- **onChange**: Executa toda vez que o TMemo for modificado.
- **onEnter**: Executa toda vez que o TAB entrar dentro do TMemo.
- **onExit**: Executa toda vez que o TAB sair dentro do TMemo.
- **onClick**: Executa toda vez que um clique for feito no TMemo.



TMemo (Lista): Eventos

- **onDbClick**: Executa toda vez que dois clicks forem feitos no TMemo.
- **onMouseEnter**: Executa toda vez que o mouse entrar no TMemo.
- **onMouseLeave**: Executa toda vez que o mouse sair no TMemo.



TMemo (Lista): Exemplo

```
var  
    I: Integer;  
begin  
  
    Memo.Clear;  
    FOR I := 0 TO 10 DO  
    BEGIN  
        Memo.Lines.Insert (I, IntToStr (I) );  
    END;
```

Inserir valores no Memo

O que é um Vetor?

- É uma **estrutura** de dados **Unidimensional**.
- É **utilizado** para **armazenar valores**.
- Possui **comportamento** similar as **variáveis**.
- Todo **vetor** possui **um único tipo de dado**.

Como Declarar um Vetor

- Utilize um dos 4 tipos de dados conhecidos.
- Vetores podem ser locais ou globais, igual as variáveis.
- Vetores possuem tamanhos estáticos (fixos).

```
VetorInteger: Array[1..100] of Integer;  
VetorReal: Array[1..100] of Real;  
VetorString: Array[1..100] of String;
```

Declaração de 3 vetores do tipo Integer, Real e String de 100 posições.

Funcionamento de um Vetor

Índice	1	2	3	4
Valor	55	43	12	24

Ilustração de um **vetor** com 4 posições (índices 1 até 4)

Como Utilizar um Vetor

Declarando o vetor



```
var  
  
    VI: Array[1..5] of Integer;  
    I: Integer;  
  
begin  
  
    for I := 1 to 5 do  
        VI[I] := RandomRange(20, 50);  
  
    for I := 1 to 5 do  
        GRID.Cells[I, 0] := IntToStr(VI[I]);
```

Acessando a posição
[I]



Inicializando valores aleatórios e escrevendo na GRID

Exercício de Vetor (Fácil)

Faça um programa que leia um **vetor** de **10 posições** de números **inteiros aleatórios (RandomRange)**. A seguir calcule o **quadrado** de cada elemento do **vetor lido**, armazenando o resultado num **segundo vetor**. Por fim, mostre o **segundo vetor** em um **TMEMO** e/ou em um **TLISTBOX**.



Exercício de Vetor (Difícil)

Faça um programa que leia **dois vetores** de **10** **números inteiros aleatórios** (RandomRange). Crie um **terceiro vetor** que seja a **união** dos dois **vetores lidos**. Por fim, mostre a união em um **TMEMO** e/ou em um **TLISTBOX**

Sons no Projeto

- Você pode **incluir sons** no **projeto**:
 - Toda vez que **algo certo** ou **errado aconteça** no **projeto**.
- Inclua **MMSystem** em **uses** que **chamará** os **sons**.
- Chame a função **sndPlaySound**, passando para ela o **caminho** até o **som**:
 - Exemplo: **sndPlaySound('Windows.wav', SND_NODEFAULT Or SND_ASYNC);**
- Nunca se esqueça que o **arquivo** deve ser de extensão **WAV**.
- Utilize a mesma ideia das **imagens** para **buscar** o **caminho** dos **arquivos** de **som**.



Incluindo pausas na execução do código

- Você pode **incluir pausas na execução do eventos**:
 - Utilizaremos este conceito na produção do jogo do genius.
- A função **sleep** "segura" a **execução** por um **tempo**:
 - O **tempo** é dado em **milissegundos**, e deve ser **passado** como **parâmetro** para a **função**.
- O **exemplo** a seguir, **pausa** a **execução** do **procedimento** por um **segundo**:
 - **sleep(1000);**
- O **jogo** do **Genius** fará **pausas** entre uma **troca** de **luz**:
 - Estas **pausas** devem ser **menores** a cada **nível** que o **usuário** **avançar**.

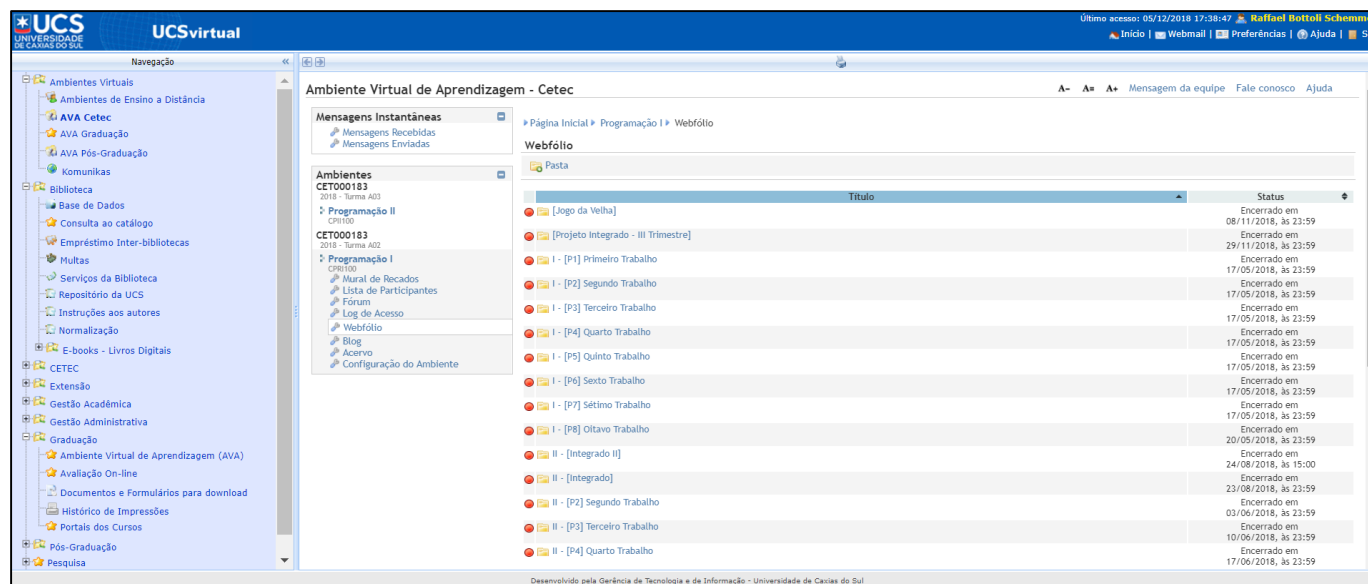
Decimo Sexto Trabalho da Disciplina (2TXVI)

- Jogo Genius:



Exemplo de projeto que deve ser desenvolvido pela turma

Publicação do TXVI no AVA



UCSvirtual

Último acesso: 05/12/2018 17:38:47 **Rafael Bottoli Schiemmer**

[Início](#) | [Webmail](#) | [Preferências](#) | [Ajuda](#) | [Sair](#)

Ambiente Virtual de Aprendizagem - Cetec

Mensagens Instantâneas

- Mensagens Recebidas
- Mensagens Enviadas

Ambientes

- CETO00183
- 2018 - Turma A03
- Programação II
- CETO00183
- 2018 - Turma A02
- Programação I
- CETO00183
- Mural de Recados
- Lista de Participantes
- Fórum
- Log de Acesso
- Webfólio
- Blog
- Acervo
- Configuração do Ambiente

Webfólio

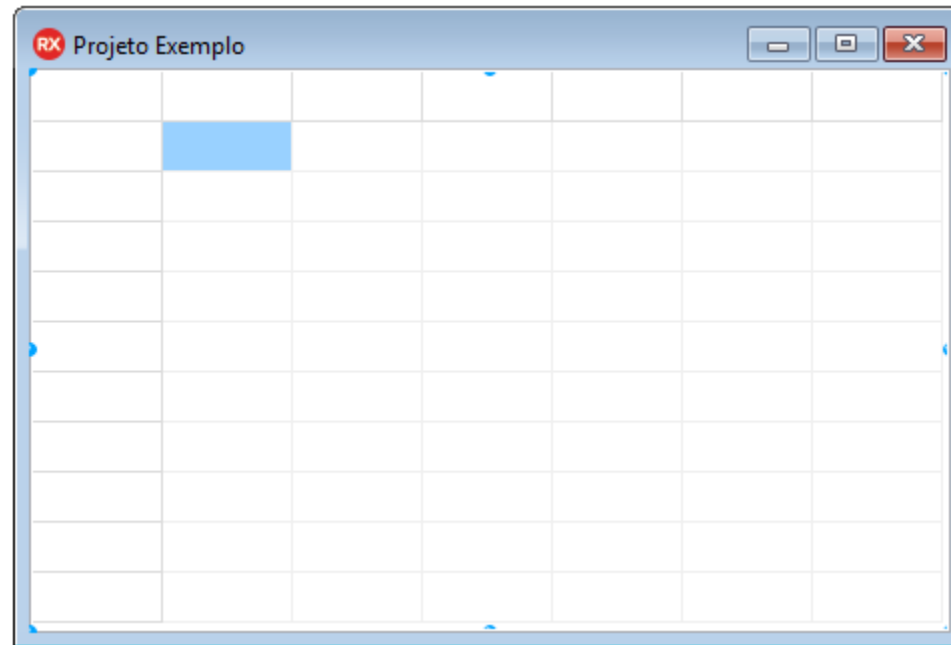
Pasta

Título	Status
[Jogo da Velha]	Encerrado em 08/11/2018, às 23:59
[Projeto Integrado - III Trimestre]	Encerrado em 29/11/2018, às 23:59
I - [P1] Primeiro Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P2] Segundo Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P3] Terceiro Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P4] Quarto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P5] Quinto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P6] Sexto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P7] Sétimo Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P8] Oitavo Trabalho	Encerrado em 20/05/2018, às 23:59
II - [Integrado II]	Encerrado em 24/08/2018, às 15:00
II - [Integrado]	Encerrado em 23/08/2018, às 23:59
II - [P2] Segundo Trabalho	Encerrado em 03/06/2018, às 23:59
II - [P3] Terceiro Trabalho	Encerrado em 10/06/2018, às 23:59
II - [P4] Quarto Trabalho	Encerrado em 17/06/2018, às 23:59

Desenvolvido pela Gerência de Tecnologia e de Informação - Universidade de Caxias do Sul

Webfólio do Ambiente Virtual de Aprendizagem (CETEC)

TStringGrid (Grade): Definição



Componente **TStringGrid**

TStringGrid (Grade): Propriedades

- **RowCount**: Define a quantidade de linhas da grade.
 - O conteúdo desta propriedade é sempre um número (Integer).
- **ColCount**: Define a quantidade de colunas da grade.
 - O conteúdo desta propriedade é sempre um número (Integer).
- **FixedCols**: Define quantas colunas serão fixas (Começa na zero).
 - O conteúdo desta propriedade é sempre um número (Integer).
- **FixedRows**: Define quantas linhas serão fixas (Começa na zero).
 - O conteúdo desta propriedade é sempre um número (Integer).

TStringGrid (Grade): Propriedades

- **ScrollBars**: Define as barras de rolagem da grade.
 - O conteúdo desta propriedade é uma constante (ssVertical/ssBoth).
- **Visible**: Define se o componente estará visível ou não.
 - O conteúdo desta propriedade é sempre um boolean (true/false).
- **Name**: Define o nome do componente na Unit.
 - O conteúdo desta propriedade é sempre uma frase (String).
- **TabOrder**: Define a ordem de chamada do TAB da grade.
 - O conteúdo desta propriedade é sempre um número (Integer).

TStringGrid (Grade): Propriedades

- **Row**: Define a **linha selecionada**.
 - O conteúdo desta **propriedade** é sempre um **número (Integer)**.
- **Col**: Define a **coluna selecionada**.
 - O conteúdo desta **propriedade** é sempre um **número (Integer)**.
- **DefaultRowHeight**: Define a **altura da linha**.
 - O conteúdo desta **propriedade** é sempre um **número (Integer)**.
- **Cells**: Define uma **posição [Col,Row]** a ser **manipulada**.



TStringGrid (Grade): Propriedades

- **DefaultColWidth**: Define a largura da coluna.
 - O conteúdo desta propriedade é sempre um número (Integer).
- **Height**: Define a altura da grade.
 - O conteúdo desta propriedade é sempre um número (Integer).
- **Width**: Define a largura da grade.
 - O conteúdo desta propriedade é sempre um número (Integer).
- **GridLineWidth**: Espessura das linhas da grade.
 - O conteúdo desta propriedade é sempre um número (Integer).

TStringGrid (Grade): Eventos

- **onClick**: Executa toda vez que um click for feito na grade.
- **onDblClick**: Executa toda vez que um click duplo for feito na grade.
- **onMouseEnter**: Executa toda vez que o mouse entrar na grade.
- **onMouseLeave**: Executa toda vez que o mouse sair da grade.
- **onShow**: Executa toda vez que a grade for mostrada.



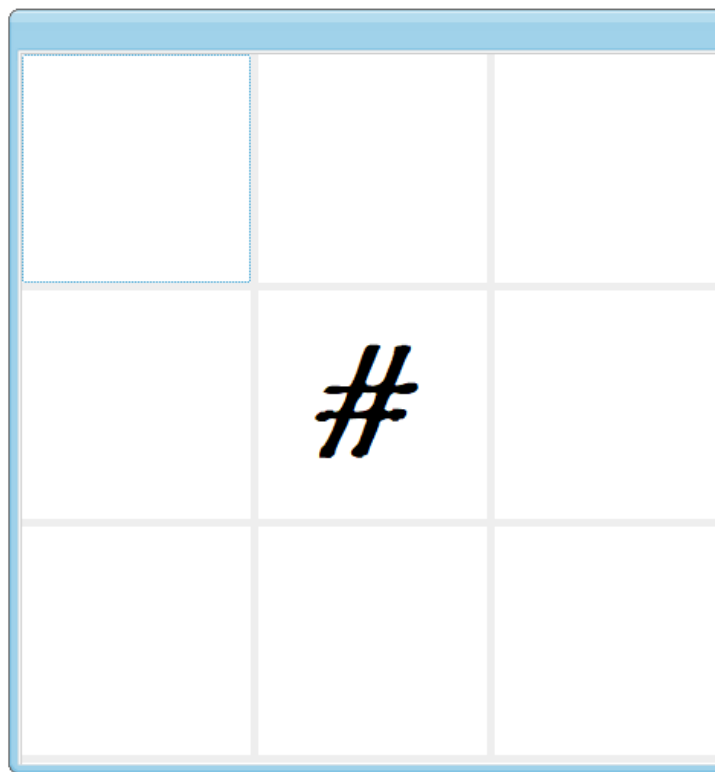
TStringGrid (Grade): Exemplo

```
Grid.ColCount := 5;  
Grid.RowCount := 5;  
for I := 1 to 5 do  
  for J := 1 to 5 do  
    if I = J then  
      begin  
        Grid.Cells[J,I] := 'X';  
      end;  
    end;  
  end;  
end;
```

Cria uma **matriz** com 5 linhas e 5 colunas
Popula a **diagonal principal** com X

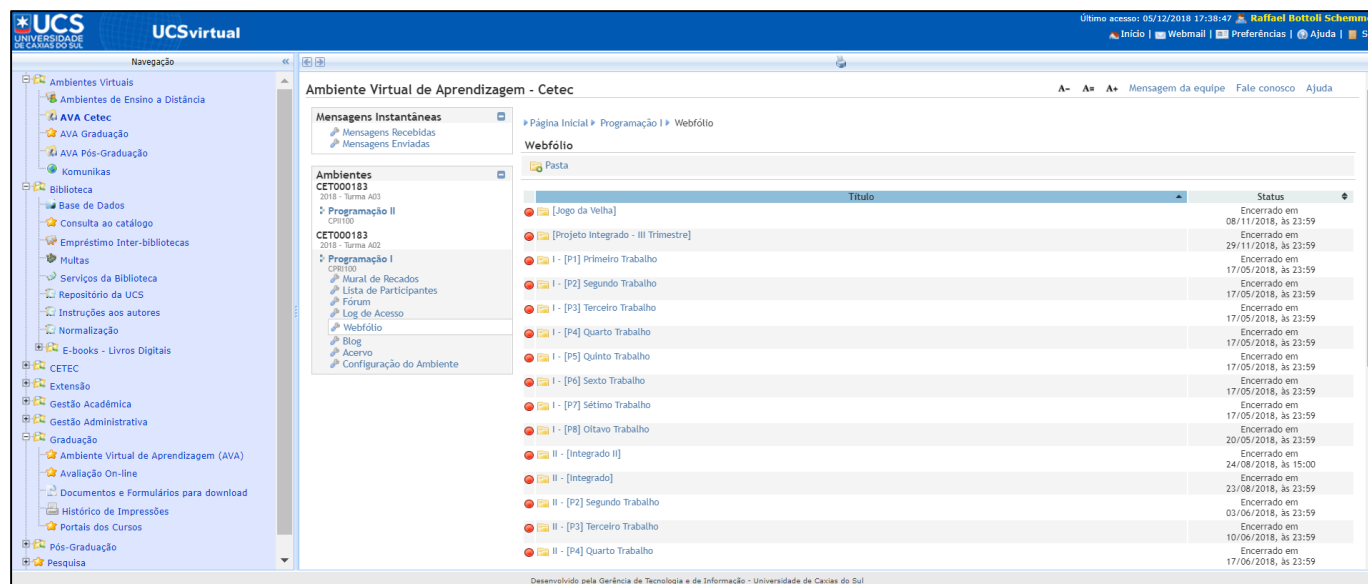
Decimo Sétimo Trabalho da Disciplina (2TXVII)

- Jogo da Velha



Utilize o exemplo disponível do GitHub para desenvolver o trabalho

Publicação do TXVII no AVA



Último acesso: 05/12/2018 17:38:47 **Rafael Bottoli Schiemmer**
Início | Webmail | Preferências | Ajuda | Sair

UCSvirtual

Ambiente Virtual de Aprendizagem - Cetec

Mensagens Instantâneas
Mensagens Recebidas
Mensagens Enviadas

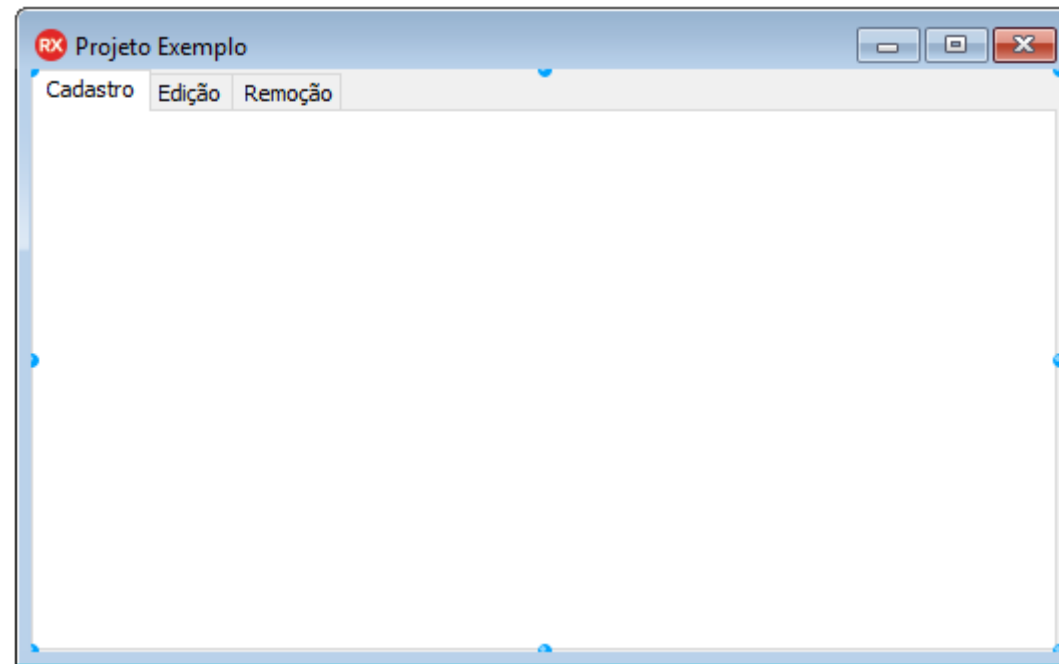
Ambientes
CET000183
2018 - Turno A02
Programação II
CET000183
2018 - Turno A02
Programação I

Webfólio

Título	Status
[Jogo da Velha]	Encerrado em 08/11/2018, às 23:59
[Projeto Integrado - III Trimestre]	Encerrado em 29/11/2018, às 23:59
I - [P1] Primeiro Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P2] Segundo Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P3] Terceiro Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P4] Quarto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P5] Quinto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P6] Sexto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P7] Sétimo Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P8] Oitavo Trabalho	Encerrado em 20/05/2018, às 23:59
II - [Integrado II]	Encerrado em 24/08/2018, às 15:00
II - [Integrado]	Encerrado em 23/08/2018, às 23:59
II - [P2] Segundo Trabalho	Encerrado em 03/06/2018, às 23:59
II - [P3] Terceiro Trabalho	Encerrado em 10/06/2018, às 23:59
II - [P4] Quarto Trabalho	Encerrado em 17/06/2018, às 23:59

Webfólio do Ambiente Virtual de Aprendizagem (CETEC)

TPageControl (Abas): Definição



Componente **TForm** com um **TPageControl**

TPageControl (Abas): Propriedades

- **Name:** Define o **nome** do **componente** na **Unit**.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **Caption:** Define o **conteúdo** da **TAB**.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **Enabled:** Define se a **aba** será **habilitada**.
 - O **conteúdo** desta **propriedade** é **sempre** um **boolean** (**true/false**).
- **Width:** Define a **largura** do **componente**.
 - O conteúdo desta **propriedade** é **sempre** um **número** (**Integer**).

TPageControl (Abas): Propriedades

- **Height:** Define a altura do componente.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **Hint:** Define uma dica para o componente.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **ShowHint:** Define se a dica será mostrada ou não.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **TabVisible:** Define se a TAB será visível ou não.
 - O **conteúdo** desta **propriedade** é **sempre** um **boolean** (**true/false**).
- **Style:** Define o estilo visual das TABs.
 - O **conteúdo** desta **propriedade** é **uma constante** (**tsButtons/tsFlatButtons/tsTabs**).

TPageControl (Abas): Eventos

- **onChange**: Executa toda vez que o TPageControl for modificado.
- **onEnter**: Executa toda vez que o TAB entrar dentro do TPageControl.
- **onExit**: Executa toda vez que o TAB sair dentro do TPageControl.



TPageControl (Abas): Eventos

- **onShow**: Executa toda vez que a aba for selecionada.
- **onMouseEnter**: Executa toda vez que o mouse entrar no TPageControl.
- **onMouseLeave**: Executa toda vez que o mouse sair no TPageControl.



TPageControl (Abas): Exemplo

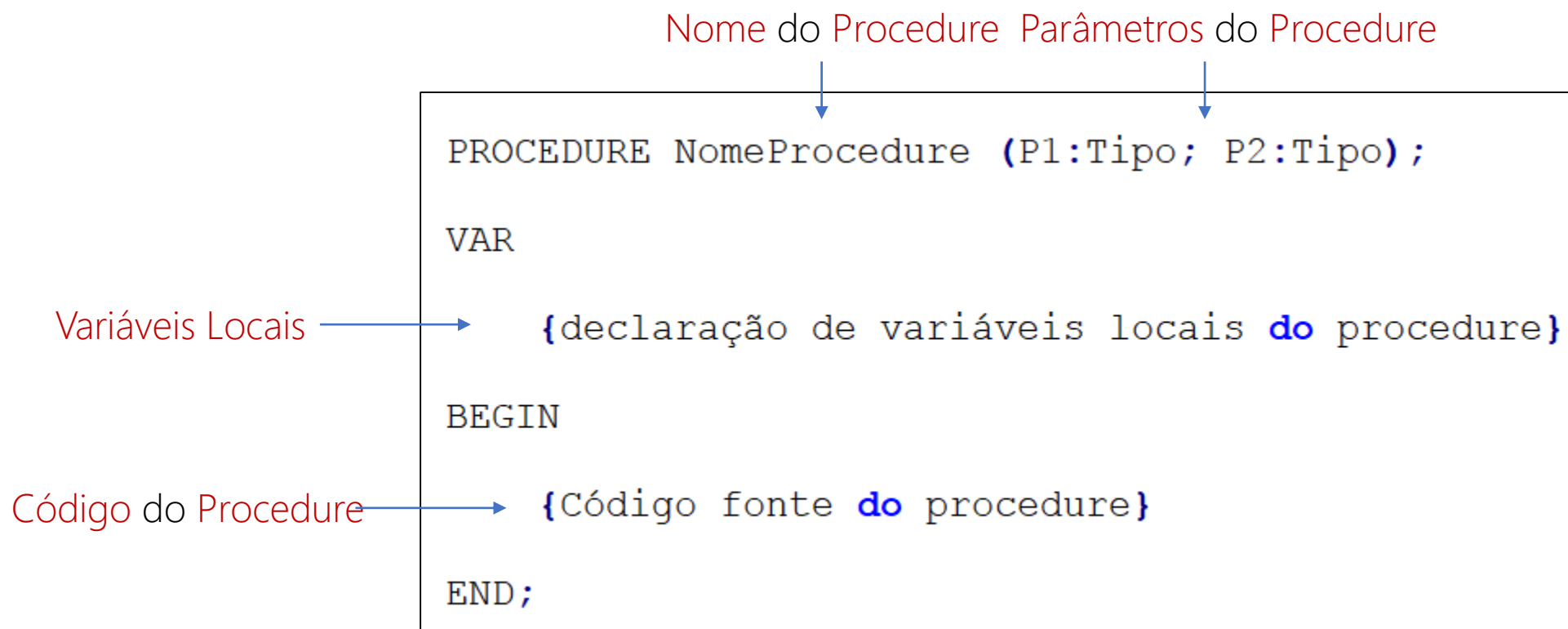
```
procedure TFrmPrincipal.ComboChange(Sender: TObject);  
begin  
  
    Cadastro.Show;  
    Cadastro.Caption:= 'Cadastro';  
  
end;
```

Tratamento do evento **onChange** do **ComboBox**
Mostra a **TabSheet Cadastro** e atualiza o **Caption**

O que é um procedimento?

- Para o Delphi (Object Pascal):
 - Todos os eventos dos objetos visuais da VCL são procedimentos.
- O que é um procedimento?
 - É um subprograma (código) do programa principal (código).

Estrutura de um procedimento



Estrutura mínima de um **procedure** em Objective Pascal (Delphi)

Criando um procedimento

- Observe algumas particularidades de um procedure:
 - [1] Não retorna valor.
 - [2] O nome do procedure deve ser único no programa.
 - [3] Não utilize nomes de procedures iguais a palavras reservadas.
 - [4] A primeira linha do código deve (SEMPRE) terminar com ponto e vírgula.
 - [5] Variáveis locais devem ser utilizadas (APENAS) no procedimento.
 - [6] Todo código do procedimento deve estar entre BEGIN e END;
 - [7] Na última linha do procedimento o comando END deve possuir ;

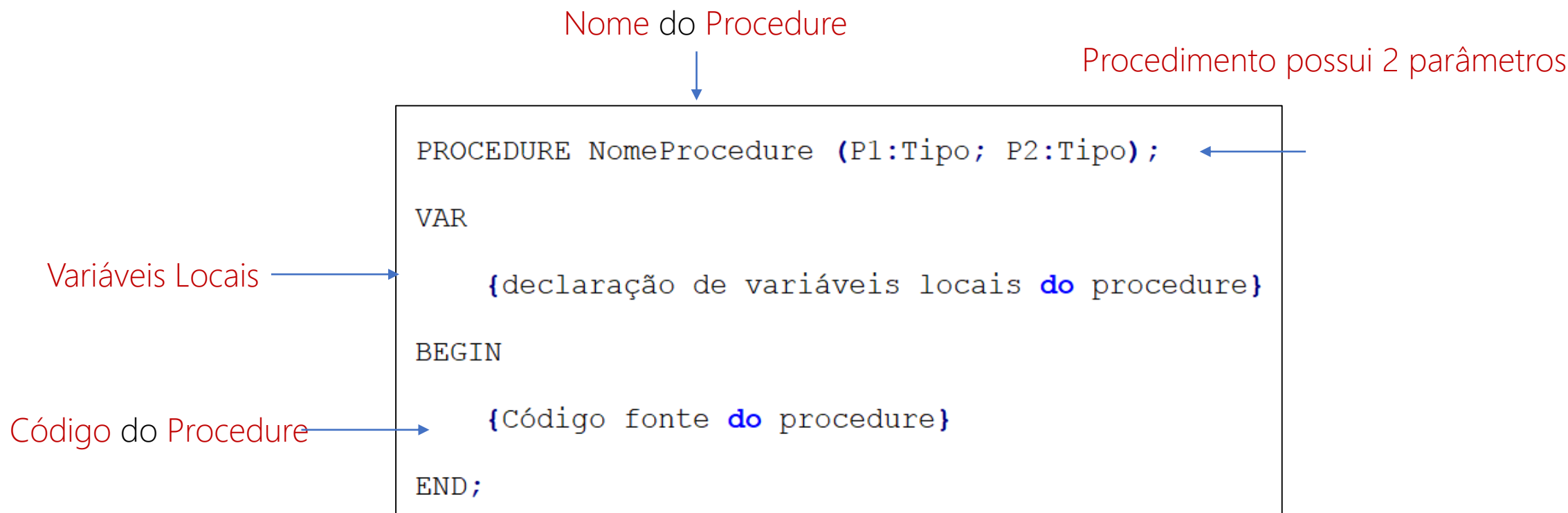


Criando um procedimento

- Qual o **objetivo** de um **parâmetro**?
 - Um **parâmetro** deve ser visto como uma **variável local do procedimento**.
- O **que** ele **faz**?
 - Ele **recebe valores passados** para o **parâmetro** em sua **chamada**.
- Quantos **parâmetros** eu **preciso declarar** ao criar um **procedimento**?
 - O **procedimento não** é obrigado a ter **parâmetros**.
 - Você verá que será **inevitável** (útil) criar **procedimentos** que **recebam parâmetros**.
 - O **programador** é **livre** para **definir quantos parâmetros** o **procedimento** deve ter.

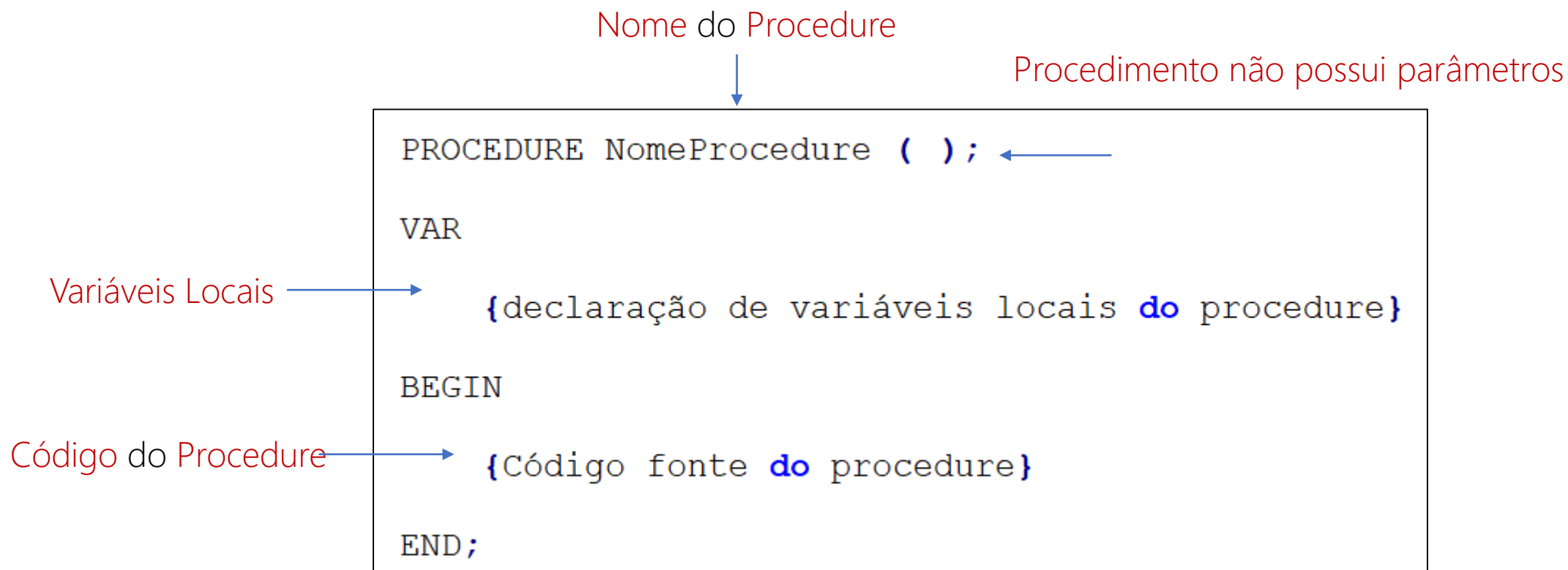


Estrutura de um procedimento



Estrutura mínima de um **procedure** em Objective Pascal (Delphi)

Estrutura de um procedimento



Estrutura mínima de um **procedure** em Objective Pascal (Delphi)

Criando um procedimento

- Dicas para nomes e tipos de parâmetros:
 - [0] Alguns programadores chamam parâmetro de argumento do procedure.
 - [1] Cada identificador (nome) do parâmetro deve possuir um nome único.
 - [2] Não utilize para nome de parâmetro os nomes das palavras da linguagem.
 - [3] Respeite as regras quanto aos nomes (não use caracteres especiais).
 - [4] Utilize todos os tipos de dados suportados pela linguagem:
 - idade: INTEGER
 - nota: REAL
 - salario: EXTENDED
 - nome: STRING
 - voto: BOOLEAN

Criando um procedimento

- Onde um procedimento deve ser criado?
 - Existem duas etapas a serem feitas.
 - [1] Declaração:
 - Indica (ao PAS) que estamos criando um novo procedimento.
 - Deve ser feito na seção interface dentro do public.
 - Nosso procedimento será público para todo programa (formulários).
 - Devemos escrever apenas a primeira linha do procedimento.
 - Ela informa ao Delphi o nome da função e seus parâmetros.



Declaração de um procedimento

- Onde um **procedimento** deve ser **criado**?

Nome do Procedimento Argumentos (Parâmetros)

```
private  
    procedure media3Valores(n1: INTEGER; n2: INTEGER; n3: INTEGER);  
public
```

;

Local e formato para definição de um procedimento no PAS

Definição de um procedimento

- Onde um procedimento deve ser criado?
 - Existem duas etapas a serem feitas.
 - [2] Definição:
 - Deve ser feito após {\$R *.dfm} e antes de end.
 - O local não é importante mas o programador deve seguir uma estrutura lógica.
 - A organização ajuda outros programadores (professor) a entender o código.
 - A definição nada mais é do que o procedimento em si (código fonte).
 - Como nossos procedimentos irão utilizar componentes (VCL) do form:
 - Precisaremos utilizar uma cláusula "TForm." antes do nome do procedimento.



Definição de um procedimento

- Onde um **procedimento** deve ser **criado**?

Vinculação ao Form

Nome do Procedimento

Argumentos (Parâmetros)

```
PROCEDURE TForm.media3Valores(n1:INTEGER; n2:INTEGER; n3:INTEGER);  
VAR  
BEGIN  
    ..  
    ..  
    ..  
END;
```


Atalhos do DELPHI



- Para definir procedimentos e funções utilize o atalho:
 - 1. Declare o procedimento ou a função.
 - 2. Com o cursor na frente da declaração pressione Ctrl + Shift + C
- O atalho acima se aplica a procedimentos e funções.



Definição de um procedimento

- Como chamar um procedimento?
 - Basta chamar o nome do procedimento e passar os parâmetros.
 - Se uma função possui 2 parâmetros você é obrigado a passar 2 valores.
 - Sempre que possível, passe os valores via variáveis.

Chamando `procedure media3Valores`
Que possui 3 parâmetros →

```
procedure TForm4.Button1Click(Sender: TObject);  
VAR  
  
    N1:= 2;  
    N2:= 4;  
    N3:= 8;  
  
BEGIN  
    media3Valores (N1,N2,N3) ;  
END;
```

Exemplo de Procedimento (Fácil)

Faça um procedimento que **calcule a média de 3 notas e mostre na saída padrão do Delphi**. As notas são valores não inteiros (REAL) que devem ser passadas para o procedimento. O programa só deve calcular a média caso o somatório das notas seja maior do que zero (verifique e informe na **saída padrão** caso o cálculo não seja possível ser realizado). Faça um programa em Delphi para testar o procedimento. O programa deve receber (do usuário) valores via **InputBox** ou via **TEdit** e deve chamar o **procedimento acima**.

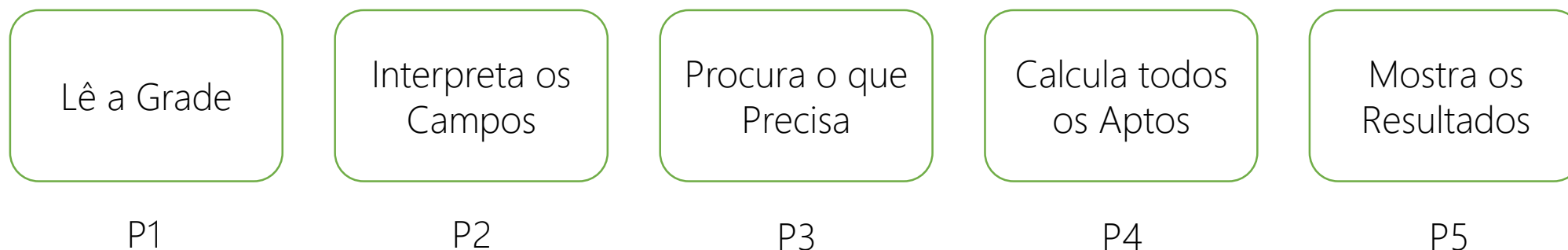
Exemplo de Procedimento (Difícil)

Faça um procedimento que **calcule se um número é primo ou não e mostre na saída padrão do Delphi a mensagem (PRIMO) ou (NÃO PRIMO)**. O número passado ao **procedimento** deve ser **inteiro** (INTEGER) e maior do que zero (verifique e informe na saída padrão caso o cálculo não seja possível ser realizado). Números **primos** são **números divisíveis** por **1** e por **ele mesmo**. Por exemplo, 3 é primo pois só é capaz de ser dividido por 1 e por 3. Já 6 não é primo pois é capaz de ser dividido por 1, 2, 3 e por 6. Faça um programa em Delphi para testar o procedimento. O programa deve receber (do usuário) valores via InputBox ou via TEdit e deve chamar o procedimento acima.



Vantagens no uso de Procedimentos

- Desenvolvimento de programas modulares:
 - Parte da lógica do código é encapsulada em um procedimento.
 - Imagine um programa com 500 linhas de código.
 - Graças a análise e o projeto modular 5 funções de 100 linhas:
 - Organizam o que precisa ser programado e resolvido.



Vantagens no uso de Procedimentos

- Como saber a hora certa de criar um procedimento?
 - Antes de começar a programar o problema.
 - Realize um pouco de **engenharia** de **software**.
 - Leia com cuidado o **enunciado** do **problema** (requisitos do programa).
 - Faça uma breve **análise** do enunciado.
 - Se **possível**, divida o que precisa ser **feito** em **pedaços**.
 - Em seguida, realize o **projeto** dos **pedaços** de forma **estruturada**.
 - Caso seja **possível**, divida os **pedaços** em **sub pedaços**.
 - Só então, de **início** a **programação**.
 - Neste estágio você conseguirá **enxergar** o **problema** em **sub problemas**.



Passagem de Parâmetros

- Os valores passados aos procedimentos:
 - São copiados para os mesmos.
 - Chama-se esta técnica de passagem de parâmetros por valor (cópia).
 - Nossos procedimentos não retornam valores.
 - Quando a palavra END; é encontrada toda a estrutura do procedure acaba.
 - As variáveis locais ao procedimento são descartadas pelo Delphi.



Passagem de Parâmetros

- Os valores passados aos procedimentos:
 - Existe outra técnica para realizar a passagem de parâmetros.
 - Ela é conhecida como passagem de parâmetro por referência.
 - O uso da técnica é simples e o impacto no resultado é significativo.
 - Todo parâmetro por referência deve vir antecedido da palavra VAR.
 - Isso significa que o parâmetro irá possuir o endereço da variável original:
 - Se este parâmetro for modificado pelo procedimento o valor será modificado.
 - Através desta técnica é possível retornar valores pelos parâmetros.
 - O usuário da função deverá conhecer se o parâmetro é por cópia ou por referência.
 - Pois alguns procedimentos poderão retornar valores (via parâmetro).



Passagem de Parâmetros

- Os valores passados aos procedimentos:

Parâmetros por cópia



Parâmetro referência



```
PROCEDURE Media (Nota1: REAL; Nota2: REAL; VAR Media:REAL) ;  
VAR  
BEGIN  
  
    Media := (Nota1 + Nota2) / 2;  
  
END;
```

O que é uma função?

- Para o Delphi (Object Pascal):
 - Uma **função** é um *procedimento* que **retorna** um **valor** (apenas isso).
 - Sendo assim, todos os **conhecimentos vistos anteriormente** são os **mesmos**.
 - Exceto pelo **retorno** da **função** (aprenderemos como **fazer** a **seguir**).
 - Toda **função** pode **retornar** um **valor** (caso o **programador** queira **retornar**).
 - O **retorno** é feito com a **palavra** reservada **RETURN**
 - Você deve **após** o **return** **realizar** um **exit** para parar a **função**.
 - Exceto pelo uso da palavra **function** e não **procedure**.

Como criar uma função?

- Tudo o que aprendemos se aplica em função:

FUNCTION



```
FUNCTION FAT(N: INTEGER;): INTEGER;  
  
VAR F,I: INTEGER;  
  
BEGIN  
  
    F := 1;  
    FOR I:= 1 to N DO  
  
        BEGIN  
            F:= F * I;  
        END;  
  
    RESULT := F;  
  
END;
```



Tipo de Retorno



Função que calcula o fatorial de um número (N)

Retorno é feito com RESULT



RESULT := F;

O que é uma função?

- Para o Delphi (Object Pascal):
 - A palavra **RESULT** pode **aparecer várias vezes** na função.
 - Você é **livre** para **adicionar** a **palavra** dentro das **condições** e/ou **laços**.
 - **Tudo** o que foi visto para **procedure** se **aplica** para **function**:
 - Inclusive a **passagem** de **parâmetro** por **valor** e por **referencia**.
 - Você **não** é **obrigado** a utilizar **RESULT** podendo **retornar valor** por **referência**.

Exemplo de Função (Fácil)

Escreva uma função no Delphi responsável por receber 3 lados (REAL) de um triângulo e por retornar o tipo de triângulo informado. Os tipos de retorno podem ser: (1) Triângulo isóceles; (2) Triângulo escaleno; (3) Triângulo equilátero. Triângulos escalenos possuem 3 lados com tamanhos diferentes. Triângulos equiláteros possuem 3 lados com o mesmo tamanho. Triângulos isóceles possuem 2 lados com medidas iguais e um com medida diferente. O cálculo só deve ser feito com números positivos e maiores que zero. Para entradas inválidas, a função deve retornar (-1). Faça ainda um programa VCL (mínimo) que receba da entrada padrão (InputBox) ou de um TEdit o número de cada um dos 3 lados e retorne na saída padrão (ShowMessage) ou TLabel as mensagens (TRIÂNGULO EQUILÁTERO) (TRIÂNGULO ISÓCELES) (TRIÂNGULO ESCALENO) (ENTRADAS NÃO SUPORTADAS).



Exemplo de Função (Difícil)

Escreva uma função no Delphi responsável por **receber 2 números (INTEGER) e por verificar se os mesmos são amigos ou não**. Números são amigos se a soma dos divisores de N1 (excluindo o próprio N1) é igual a N2 e se a **soma dos divisores de N2 (excluindo o próprio N2) é igual a N1**. Sua função deve **retornar (1)** se os 2 números **informados** para a **função** forem **amigos** (possuírem a propriedade anterior). Caso **contrário** deve retornar **(0)**. Se uma das entradas forem **negativas** a função deve **retornar (-1)**. Faça ainda um programa **VCL** (mínimo) que receba da **entrada padrão (InputBox)** ou de um **TEdit** os **dois números** e **retorne** na **saída padrão (ShowMessage)** ou **TLabel** as **mensagens (NÚMEROS AMIGOS) (NÚMEROS NÃO AMIGOS) (ENTRADAS NÃO SUPORTADAS)**.



O Delphi possui funções?

- Sim, como você já deve imaginar:
 - Todos os **elementos VCL** do **Delphi** são **procedimentos** e **funções**.
 - Conforme aprendemos, **funções** são **blocos** de **código** reutilizáveis.
 - Quando você arrasta (chama) um **InputBox** você está **reusando** o **código**.
- Assuma sempre ao criar funções:
 - Que no futuro você poderá **reutilizar** as **funções**.
 - Que **outras pessoas** poderão **utilizar** estas **funções**.
 - Desde que você **disponibilize** o **código** (publicamente).
 - Funções também podem ser **vendidas** para **outros programadores**.
 - Encare um **programa** como **funções** que **chamam funções**.

O Delphi possui funções?

- Vamos estudar algumas funções matemáticas clássicas:

- $x := \text{sqr}(x)$ Quadrado de x (INTEGER) : (INTEGER)
- $x := \text{sqrt}(x)$ Raiz quadrada de x (INTEGER) : (INTEGER)
- $x := \text{sin}(x)$ Seno de x (INTEGER) : (INTEGER)
- $x := \text{cos}(x)$ Cosseno de x (INTEGER) : (INTEGER)
- $x := \text{tan}(x)$ Tangente de x (INTEGER) : (INTEGER)
- $x := \text{int}(x)$ Parte inteira de x (REAL) : (INTEGER)
- $x := \text{frac}(x)$ Parte fracionária de x (REAL) : (INTEGER)
- $x := \text{power}(x,y)$ x na potencia y (xy) (INTEGER,INTEGER) : (INTEGER)
- $x := \text{abs}(x)$ Valor absoluto de x (INTEGER) : (INTEGER)

O Delphi possui funções?

- Seguem mais funções mais clássicas:
 - Entrada e saída
 - NomePais := **InputBox** ('Escolha de país', 'Digite o nome do país:', 'Brasil');
 - **ShowMessage** ('Olá mundo');
 - Incremento e decremento
 - **Inc**(X);
 - **Dec**(X);

O Delphi possui funções?

- Seguem mais funções mais clássicas:
 - Conversão entre Maiúsculo para Minúsculo
 - Nome := `LowerCase`('RAFAEL');
 - Nome := `UpperCase`('rafael');
 - Números aleatórios e arredondamento/truncamento.
 - Aleatorio := `RandomRange`(10,20);
 - Num := `round`(23,4);
 - X:= `trunc`(23,4);



Boas práticas de programação

- Entre **programadores**:
 - É **legal definir** algumas **regras comuns**.
 - **Regras baseadas** no que diz o **coletivo** (**consenso** entre a comunidade).
 - Vamos **aprender algumas convenções** que tornam o **código** mais **agradável**.
 - Respeitando as **regras** do Object Pascal (**Delphi**) e do **RAD** (**VCL**).
- **Regra [1] : Comentários**
 - Lembre que o Delphi possui **3 tipos de comentários** sendo eles.
 - **//** Sou um **comentário** de **linha**
 - **(*** Comentário de bloco ***)** ou **{** Comentário de bloco **}**

Boas práticas de programação

- Onde eu utilizo um comentário?
 - Sempre que estiver "definindo" um procedimento ou função.
 - Como o comentário deve ser escrito para um procedimento?
 - Faça um comentário de bloco como o descrito abaixo:

```
{  
    Autor: Raffael Bottoli Schemmer  
    Objetivo: Procedimento que calcula e retorna no 2 parâmetro a média  
    PAR: (real, VAR real) Notas a serem calculadas a média  
}
```

Boas práticas de programação

- Onde eu **utilizo** um **comentário**?
 - Sempre que estiver "**definindo**" um **procedimento** ou **função**.
 - Como o **comentário** deve ser **escrito** para uma **função**?
 - Faça um **comentário** de **bloco** como o **descrito** abaixo:

```
{  
    Autor: Raffael Bottoli Schemmer  
    Objetivo: Função que calcula e retorna no 2 parâmetro a média  
    PAR: (real, VAR real) Notas a serem calculadas a média  
}
```

Boas práticas de programação

- Onde eu **utilizo** um **comentário**?
 - Sempre que **implementar** algo relativamente **importante** e **complexo**.
 - **Comentários** de **linha** são **importantes** para **detalhar** coisas **simples**.
 - Exemplo:
 - $senha = a*b + !a^3*2/1+2\3 + !a*b$
 - Observe como a **linha** acima **necessita** de um **comentário** para **facilitar** o **entendimento**.
- Afinal, **porque comentar**?
 - Pois outras pessoas (ou você mesmo) irá querer **estudar** o **programa** no **futuro**.
 - Os **comentários** auxiliam a **entender** um **código complexo** (ou feito no passado).



Boas práticas de programação

- Regra [2]: **Nome** de **variável** ou **parâmetro**:
 - Nunca **utilize** nomes **similares** do **Delphi**.
 - Como uma variável **BEG** ou **IND**.
 - **Convencione** em utilizar as **palavras** da **linguagem** em letra **maiúscula**.
 - **Nome** das **variáveis** em letra minúscula.
 - **Nomes simples** (idade, data, salario, hora)
 - **Nomes composto** (diaSemana, salarioMinimo, valorMedio).
 - Este **padrão** é **conhecido** como **camelCase**.
 - Linguagens **modernas** como Java/C# possuem como **regra** este **padrão**.
 - Linguagens **antigas** não se dão bem com o **CapsLock** (CASE).
 - **FUNCTION** / **function** / **Function** são 3 coisas diferentes para o C e para o C++.
 - O case sensitive (**CAPS**) **muda** o **sentido** das **coisas**.

Boas práticas de programação

- Regra [3]: Uso de **procedimento** e **função**:
 - Tente **SEMPRE** que puder **utilizar** os **parâmetros** e **variáveis locais**.
 - Imagine a **função** de forma **autocontida** (independente) do **programa**.
 - Imagine você **vendendo** a **função** no **futuro**.
 - O que você **explicaria** ao **usuário** da **função** (cliente/programador)?
- Pergunta: O cliente ficaria feliz se fosse necessário declarar 10 variáveis globais, sendo elas com nomes específicos (Ex: contadorLaco3) do tipo INTEGER para utilizar a função?
- Pela **pergunta** acima, você deve **procurar** criar uma **função** modular (**encapsulada**), que possua **tudo** o que **precisa** dentro de si mesma.
- Isso **facilita** que a mesma seja **copiada** e **vendida** a outros **códigos/programadores**.



Boas práticas de programação

- Regra [4]: **Variáveis locais** e **globais**
 - Procure utilizar o **mínimo** possível as **variáveis globais**.
 - Projete as **funções** para que as mesmas **consigam trocar** (**copiar**) os **dados**.
 - Se a **informação** for **muito grande** (matriz 100x100) passe a **matriz** por **referência** a cada **chamada** de **função**.
 - Utilize **variáveis globais** apenas em **último caso**.
 - Considere que a **variável global** faz **parte** do **programa** e não das **funções**.



Fontes

- Além das fontes existentes no RAD:
 - Você poderá **incluir** outros tipos de **fontes** no **projeto**.
 - As **fontes** podem ser **baixadas** de **sites** como <https://www.dafont.com/pt/>
- Todas as **fontes** devem possuir **extensão** de arquivo **.ttf**
 - Ao clicar sobre o arquivo ttf, você poderá **obter** o **estilo** da **fonte**. O botão de **install** deve ser **clicado**.
 - A **instalação copia** a **fonte** para o **diretório** C:\Windows\Fonts do **computador**.
- Em seguida, você poderá **estilizar** os componentes **VCL** com esta **fonte**.
 - Lembre apenas que a **fonte** que você utilizou, estará **disponível** apenas na **máquina** do **desenvolvedor**.
 - Desta forma, **levaremos** o arquivo **ttf juntamente** com o **EXE** do **projeto**.

Fontes

- Você deve (ao gerar o instalador), incluir o arquivo ttf como parte do programa:
 - Arquivo .ttf ficará junto com o EXE do programa.
- Inclua no onCreate do projeto, as linha de código a seguir:
 - Elas instalam o TTF no computador toda vez que o programa iniciar.
 - `AddFontResource(ExtractFileDir(Application.ExeName)+"\font.ttf");`
 - `SendMessage(HWND_BROADCAST, WM_FONTCHANGE, 0, 0);`
- Não se esqueça de mudar o font.ttf para o nome da fonte:
 - Evite espaços em branco para nome de arquivo).
- Caso a fonte não seja encontrada, o delphi assume o estilo de fonte padrão.

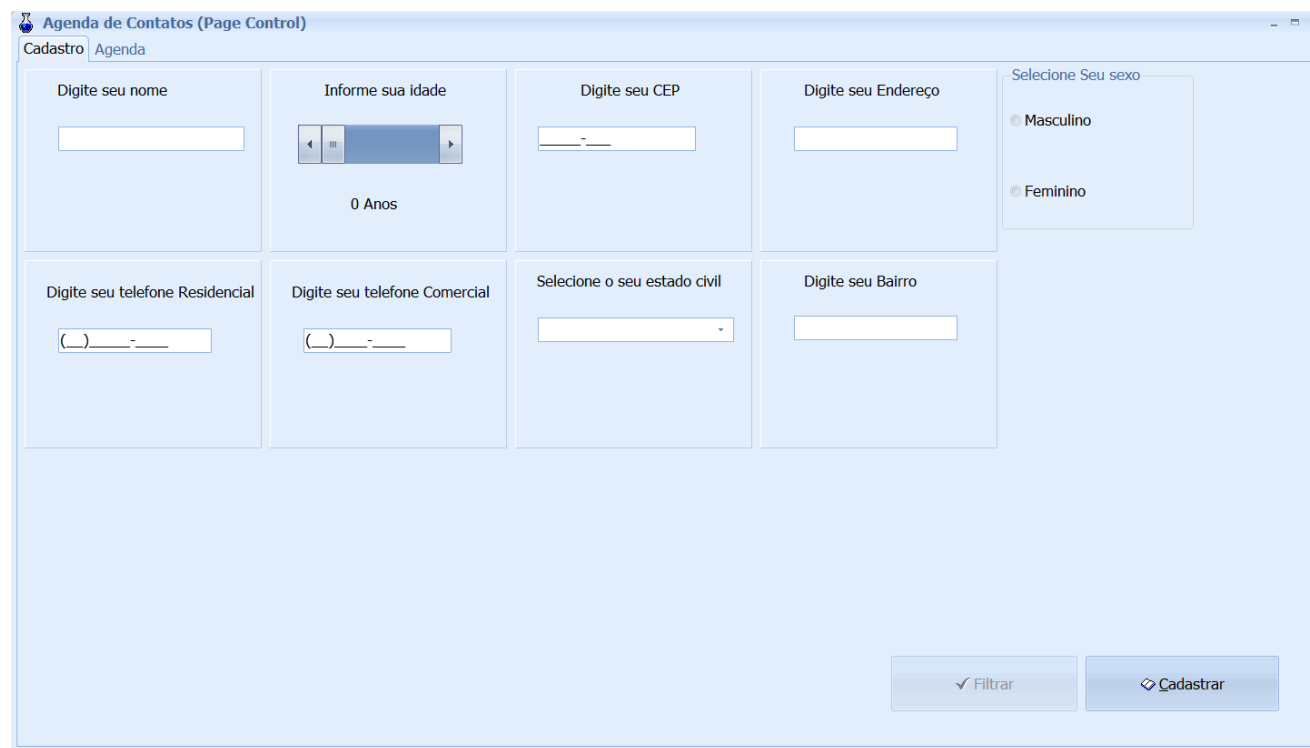
Links Web

- Vocês podem chamar o navegador web:
 - Passando links (sites) que podem ser carregados.
- Nós podemos criar links (Git/Drive):
 - Apontando para locais estratégicos para o usuário buscar info.
 - Uma dica é adicionar o site do CETEC no logo da escola.
- Insira em uses a chamada ao componente ShellAPI.
- Adicione a linha no código (modifique o link):
 - `ShellExecute(Handle,'open','http://www.ucs.br/cetec',nil,nil,SW_SHOWMAXIMIZED);`



Decimo Oitavo Trabalho da Disciplina (2TXVIII)

- Agenda de Contatos com Page Control



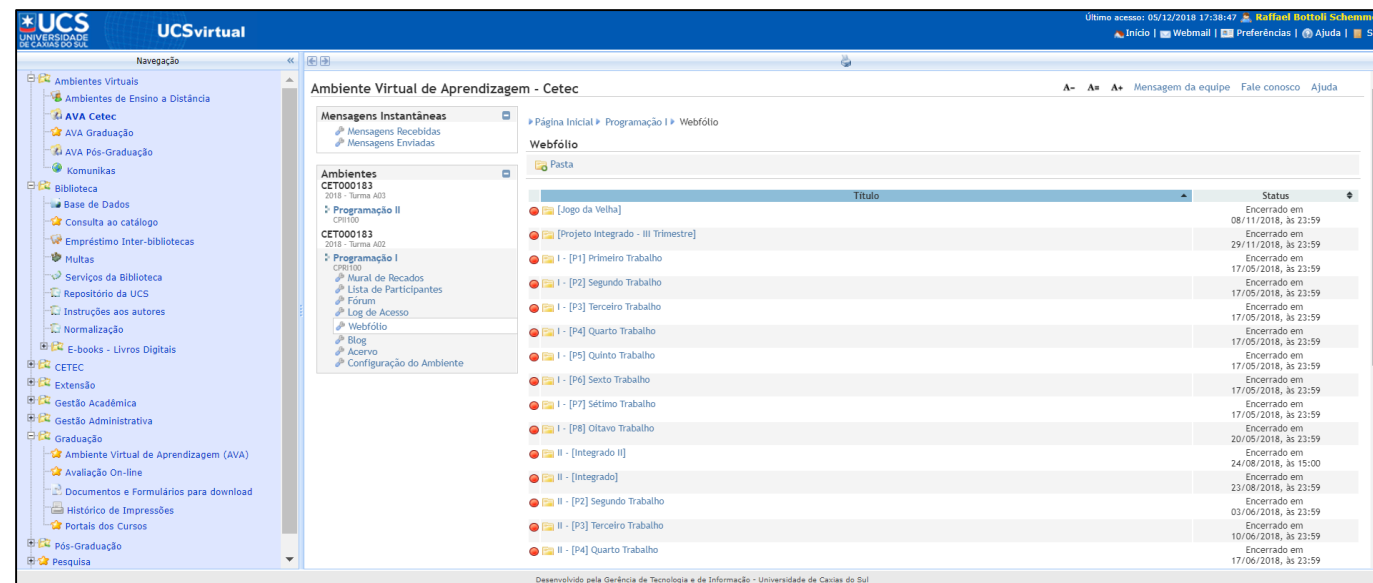
The screenshot shows a web application window titled "Agenda de Contatos (Page Control)". It has two tabs: "Cadastro" (selected) and "Agenda". The form is divided into several sections:

- Top Row:**
 - Digite seu nome:** A text input field.
 - Informe sua idade:** A numeric input field with a spinner, showing "0 Anos".
 - Digite seu CEP:** A text input field with a hyphen.
 - Digite seu Endereço:** A text input field.
 - Selecione Seu sexo:** Radio buttons for "Masculino" and "Feminino".
- Bottom Row:**
 - Digite seu telefone Residencial:** A text input field with a hyphen.
 - Digite seu telefone Comercial:** A text input field with a hyphen.
 - Selecione o seu estado civil:** A dropdown menu.
 - Digite seu Bairro:** A text input field.

At the bottom right, there are two buttons: "Filtrar" (with a checkmark icon) and "Cadastrar" (with a plus icon).

Exemplo de projeto que deve ser desenvolvido pela turma

Publicação do TXVIII no AVA



Último acesso: 05/12/2018 17:38:47 **Rafael Bottoli Schiemmer**

[Início](#) | [Webmail](#) | [Preferências](#) | [Ajuda](#) | [Sair](#)

UCSvirtual

Ambiente Virtual de Aprendizagem - Cetec

Mensagens Instantâneas

- Mensagens Recebidas
- Mensagens Enviadas

Ambientes

- CET000183
- 2018 - Turma A03
- Programação II
- CET000183
- 2018 - Turma A02
- Programação I
- CRS100
- Mural de Recados
- Lista de Participantes
- Fórum
- Log de Acesso
- Webfólio
- Blog
- Acervo
- Configuração do Ambiente

Webfólio

Pasta

Título	Status
[Jogo da Velha]	Encerrado em 08/11/2018, às 23:59
[Projeto Integrado - III Trimestre]	Encerrado em 29/11/2018, às 23:59
I - [P1] Primeiro Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P2] Segundo Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P3] Terceiro Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P4] Quarto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P5] Quinto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P6] Sexto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P7] Sétimo Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P8] Oitavo Trabalho	Encerrado em 20/05/2018, às 23:59
II - [Integrado II]	Encerrado em 24/08/2018, às 15:00
II - [Integrado]	Encerrado em 23/08/2018, às 23:59
II - [P2] Segundo Trabalho	Encerrado em 03/06/2018, às 23:59
II - [P3] Terceiro Trabalho	Encerrado em 10/06/2018, às 23:59
II - [P4] Quarto Trabalho	Encerrado em 17/06/2018, às 23:59

Desenvolvido pela Gerência de Tecnologia e de Informação - Universidade de Caxias do Sul

Webfólio do Ambiente Virtual de Aprendizagem (CETEC)

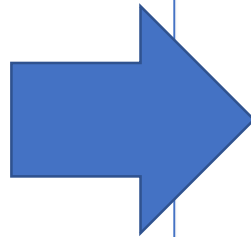
Registros (Record)

- Tecnologia que permite **criar tipos de dados heterogêneos**.
- **Tipo heterogêneo** é um tipo formado de tipos primitivos.
- Isso permite **criar variáveis complexas**.
- **Defina o novo tipo antes de var.**



Registros (Record)

Definindo
tipo de dado Pessoa



```
TYPE Pessoa = record

    Idade: integer;
    Nome  : string[35];
    Endereco: string[35];

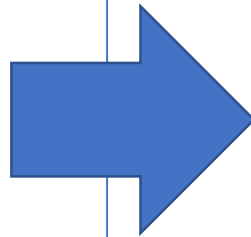
END;

var
    FrmPrincipal: TFrmPrincipal;
```

Toda **pessoa** terá uma **Idade**, um **Nome** e um **Endereço**

Registros (Record)

Definindo
tipo de dado Pessoa



```
TYPE Pessoa = record
```

```
    Idade: integer;  
    Nome : string[35];  
    Endereco: string[35];
```

```
END;
```

Declarando variável
De Nome Cliente
Do tipo Pessoa



```
var
```

```
    Cliente : Pessoa;  
    FrmPrincipal: TFrmPrincipal;
```

Registros (Record)

```
procedure TFrmPrincipal.FormCreate(Sender: TObject);  
begin  
  
    Cliente.Nome := 'Raffael';  
    Cliente.Idade := 33;  
    Cliente.Endereco := 'Petrópolis';  
end;
```

Manipulando o conteúdo da variável heterogênea através do operador (.)

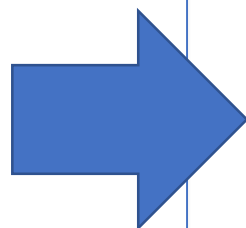
Arquivos (Conceito)

- Tecnologia que permite persistir (para sempre) os dados do programa no HDD.
- Muito utilizado por programas que não utilizam banco de dados.
- Observe que na declaração da variável arquivo seu tipo de dado é FILE OF
- Neste exemplo utilizaremos um arquivo (FILE) do tipo (CONTATO)



Arquivos (Declarando FILE)

Definindo
tipo de dado Contato



TYPE

```
CONTATO = record  
Matricula: integer;  
Nome: string[40];  
Endereco: string[50];  
StatusReg: BOOLEAN;
```

END;

Declarando
apontador de FILE



var

```
FrmPrincipal: TFrmPrincipal;  
ARQUIVO: file of CONTATO;
```

Arquivos (Conectando FILE)

Vinculando FILE
a AGENDABD.DAT



Testa se o arquivo
foi encontrado



Se sim abre o Arquivo



Se não for cria o Arquivo



```
AssignFile (ARQUIVO, 'AGENDABD.DAT');  
  
IF FILEEXISTS ('AGENDABD.DAT') THEN  
[  
  BEGIN  
    RESET (ARQUIVO)  
  END  
  ELSE  
  [  
    BEGIN  
      REWRITE (ARQUIVO);  
    END;
```

Arquivos (Lendo Arquivo)

Posiciona o cursor
de leitura no início



Lê o arquivo enquanto
não for o
fim do arquivo EOF



Realiza a leitura
de uma linha do arquivo



```
SEEK (ARQUIVO, 0) ;  
i := 0 ;  
WHILE (NOT EOF (ARQUIVO)) DO  
[BEGIN  
  
    READ (ARQUIVO, CONTATO[i]) ;  
    i := i + 1 ;  
  
END ;
```

Arquivos (Escrevendo Arquivo)

Limpa o Arquivo
Para Receber novos Dados



```
ERASE (ARQUIVO) ;  
i := 0;  
WHILE i < 100 DO  
BEGIN
```

Escreve em uma
Linha do Arquivo



```
    WRITE (ARQUIVO, CONTATO[i]);  
    i := i + 1;
```

Realiza o Fechamento
do Arquivo



```
END;  
CLOSEFILE (ARQUIVO) ;
```

Arquivos (Resumo das Funções)

- **ASSIGNFILE**(ARQUIVO, 'AGENDABD.DAT'):
 - Vincula o arquivo texto **AGENDABD.DAT** a variável **ARQUIVO**.
- **FILEEXISTS**('AGENDA.DAT'):
 - Verifica se o arquivo existe (retorna **TRUE** se sim).
- **RESET**(ARQUIVO):
 - Abre o arquivo para leitura/escrita.
- **REWRITE**(ARQUIVO):
 - Limpa (apaga) o conteúdo do arquivo.

Arquivos (Resumo das Funções)

- **SEEK**(ARQUIVO,0):
 - Posiciona o cursor de leitura para o início do arquivo.
- **NOT EOF**(ARQUIVO):
 - Realiza a leitura do arquivo enquanto não for o fim do arquivo.
- **WRITE**(ARQUIVO, AGENDA[i]):
 - Escreve no arquivo um valor (CONTATO) da AGENDA.
- **READ**(ARQUIVO, AGENDA[i]):
 - Lê do arquivo um valor (CONTATO).
- **CLOSEFILE**(ARQUIVO):
 - Realiza o fechamento do arquivo.

Calendário/Datas (TDate)

- Utilidade:
 - Permite que os programas delphi utilizem datas.
- Declarando uma variável do tipo TDate e formatando a mesma:
 - `data : TDate;`
 - `FormatDateTime('d/m/y', data)`
- Capturando a data atual do sistema:
 - `data := now;`

Calendário/Datas (TDate)

- Crie **variáveis** do tipo **word** para receber os **valores** da **data**:
 - **myYear: Word;**
 - **myMonth: Word;**
 - **myDay: Word;**
- Utilize a função **DecodeDate** para **pegar** a **data** do **calendário**:
 - **DecodeDate(Data, myYear, myMonth, myDay);**



Calendário/Datas (TDate)

- Mostre os valores de `myDay` / `myMonth` / `myYear` na saída padrão:
 - `showmessage(IntToStr(myDay) + '/' + IntToStr(myMonth) + '/' + IntToStr(myYear));`
- Definindo um `valor` para `data`:
 - `myYear := 2019;`
 - `myMonth := 03;`
 - `myDay := 01;`
 - `data := EncodeDate(myYear, myMonth, myDay);`

Calendário/Datas (TCalendarPicker)

- Utilidade:
 - Permite que o usuário **selecione** uma **data** que pode ser **salva** (como a **data** de **nascimento**).
- **Propriedades:**
 - **Name:** Nome do componente
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
 - **Color:** Cor do componente
 - O conteúdo desta **propriedade** é sempre uma constante (CIBlue/CIRed).
 - **Date:** Data inicial do componente
 - O conteúdo desta **propriedade** é sempre um TDate.
 - **Enabled:** Habilita/Desabilita o componente
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).
 - **TextHint:** Define um texto de dica para o componente
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **Eventos:**
 - **onChange:** É chamado toda vez que uma **data** for **selecionada** no **calendário**.

Calendário/Datas (TCalendarPicker)

- Como utilizar:
- Crie **variáveis** do tipo **word** para **receber** os **valores** da **data**:
 - **myYear: Word;**
 - **myMonth: Word;**
 - **myDay: Word;**
- Utilize a **função DecodeDate** para **pegar** a **data** do **calendário**:
 - **DecodeDate(CalendarPicker1.Date, myYear, myMonth, myDay);**

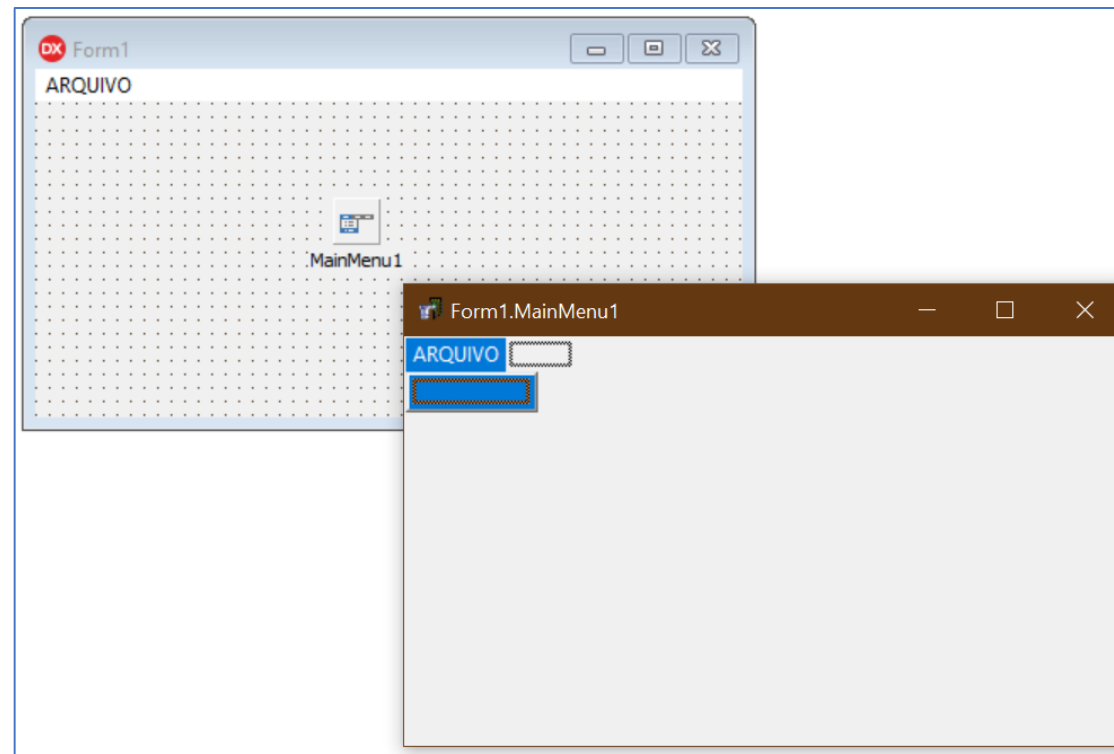
Calendário/Datas (TCalendarPicker)

- Utilize `DateFormat` para atribuir uma nova data ao calendário:
 - `myYear := 2019;`
 - `myMonth := 03;`
 - `myDay := 01;`
 - `data := EncodeDate(myYear, myMonth, myDay);`
 - `CalendarPicker1.Date := data;`

TMainMenu

- O que Implementa:
 - Implementa um menu superior que permite o usuário selecionar funcionalidades.
- Propriedades:
 - Name: Nome do Menu
 - MenuDesign: Permite criar opções
 - Enabled: Habilita ou desabilita uma opção do menu
 - Caption: Define uma opção para o menu
 - Bitmap: Permite adicionar imagem na posição.
- Eventos:
 - onClick: Define um procedimento que trata (atende) o click em uma das opções.

TMainMenu



Exemplo de uso do componente TMainMenu

TStatusBar (Barra de Status)

- Barra de status (a ser utilizada no rodapé do programa) que trás informações sobre o programa.
- Podemos utilizar a barra para informar quantos contatos estão cadastrados na agenda.



TStatusBar (Propriedades)

- Color: Define a cor da barra de status
 - O conteúdo desta **propriedade** é **sempre** uma **constante** (Ex: **CLBlue/CLRed**).
- Hint : Define um texto de dica para a barra de status
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- ShowHint : Habilita/Desabilita a dica
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).
- Enabled: Habilita/Desabilita a barra de status
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).



TStatusBar (Propriedades)

- Visible : Torna a barra de status visível ou não
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).
- Name: Nome do componente
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- Panels[0].Text : Define o conteúdo da barra de status
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).

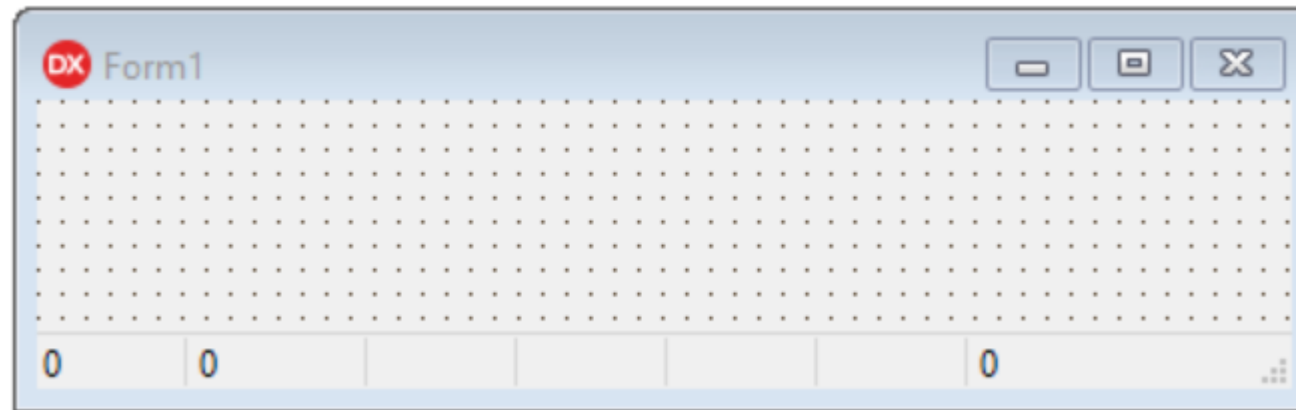


TStatusBar (Eventos)

- **onEnter**: Executa toda vez que o **TAB entrar** dentro do **TStatusBar**.
- **onExit**: Executa toda vez que o **TAB sair** dentro do **TStatusBar**.
- **onClick**: Executa toda vez que um **clique** for feito no **TStatusBar**.
- **onDbClick**: Executa toda vez que **dois clicks** forem feitos no **TStatusBar**.
- **onMouseEnter**: Executa toda vez que o **mouse entrar** no **TStatusBar**.
- **onMouseLeave**: Executa toda vez que o **mouse sair** no **TStatusBar**.



TStatusBar (Exemplo)



Exemplo de TStatusBar com vários elementos

MessageDlg (Caixa de Diálogo Múltipla)

- Abre **caixas** de **diálogo** (**condicionadas**) para o **usuário** **escolher** o que fazer.
- Vantagens: Permite **confirmar/informar** as **ações** que o **usuário** **fará** no programa.
- A função **MessageDlg** precisa de **4 parâmetros** sendo eles:
 - [1] **Nome** da **mensagem** na **caixa** de **diálogo** como 'Quer mesmo apagar usuário?'
 - [2] **Tipo** de **mensagem** como **mtConfirmation** por exemplo
 - [3] **Tipo** de **condição** a ser **utilizada** como **mbYesNo**
 - [4] Passar zero



MessageDlg (Caixa de Diálogo Múltipla)

- Tipos de mensagens que podem ser utilizadas:

- mtWarning
- mtError
- mtConfirmation

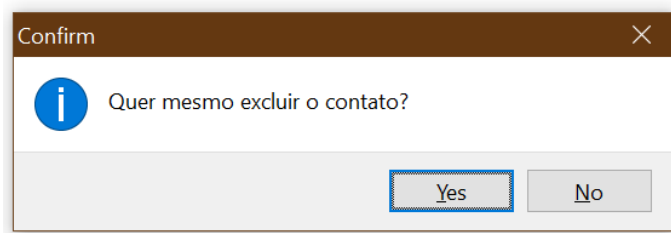
- Tipos de condições mais utilizadas:

- mbYesNo
- mbOkCancel

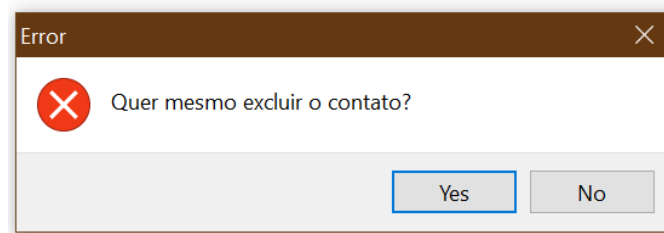
MessageDlg (Caixa de Diálogo Múltipla)

- Retorno das mensagens (Todas Integer):
 - mrYes
 - mrNo
 - mrOk
 - mrCancel

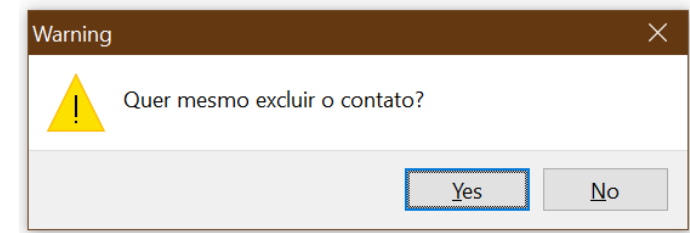
MessageDlg (Caixa de Diálogo Múltipla)



mtConfirmation



mtError



mtWarning

MessageDlg (Caixa de Diálogo Múltipla)

```
retorno := MessageDlg('Quer mesmo excluir o contato?',  
mtConfirmation,mbYesNo, 0);  
  
if retorno = mrYes then  
begin  
    ShowMessage('Usuário excluído com sucesso!');  
end  
else  
begin  
    ShowMessage('Exclusão cancelada!');  
end;
```

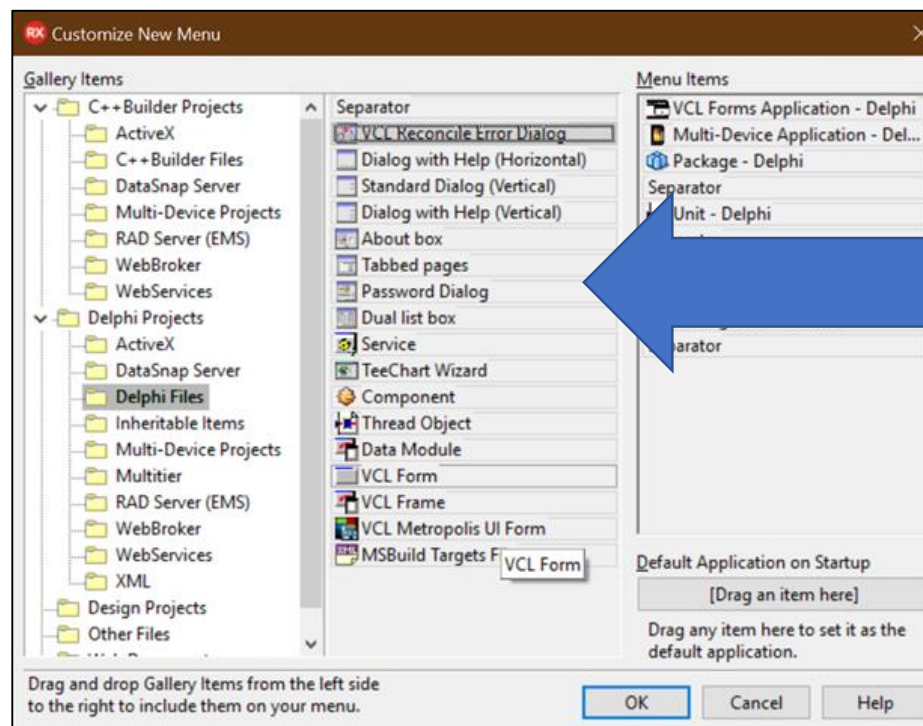
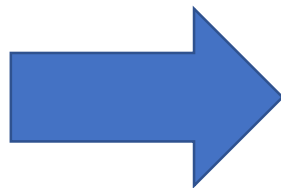
Exemplo de uso do componente MessageDlg



Layouts (TFORMs) Exemplo

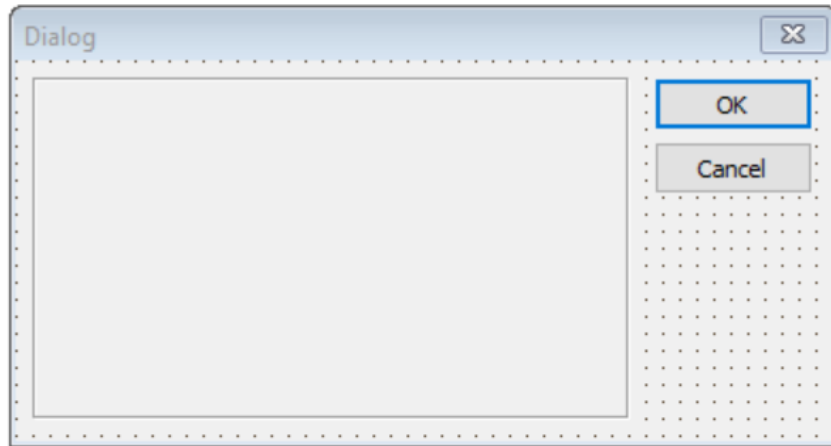
- Criando formulários exemplos (pré-projetados):
 - File > New > Other ..

Selecione DELPHI Files

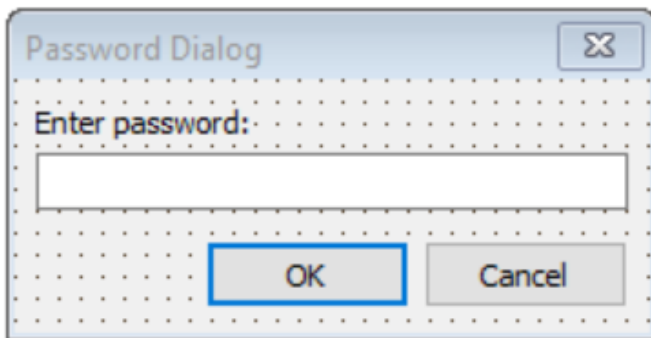


Exemplo de TFORMs Prontos

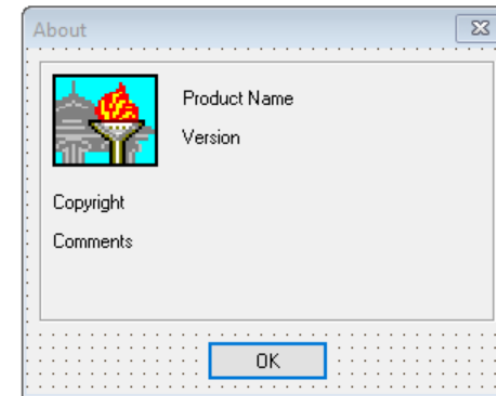
Layouts (TFORMs) Exemplo



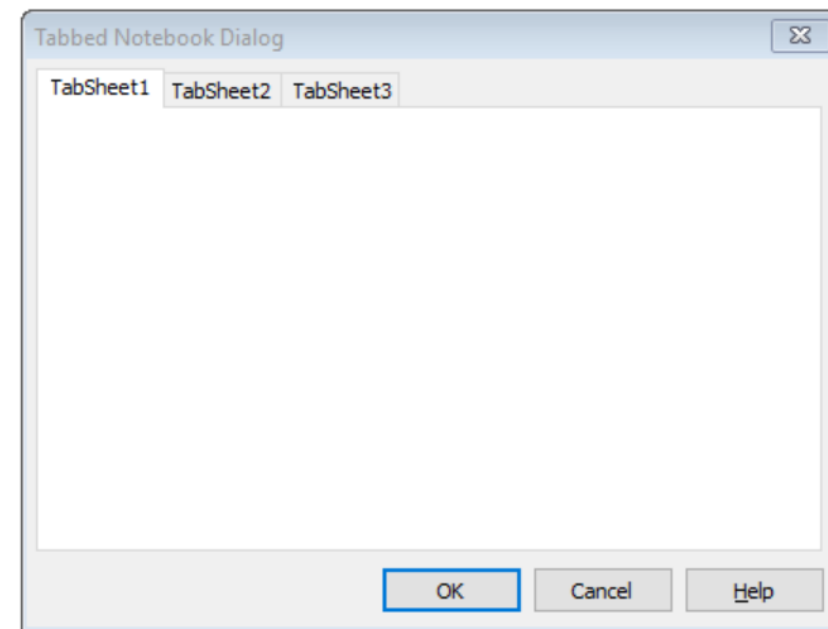
Dialog With Help (Vertical)



Password Dialog



About Box



Tabbed Pages

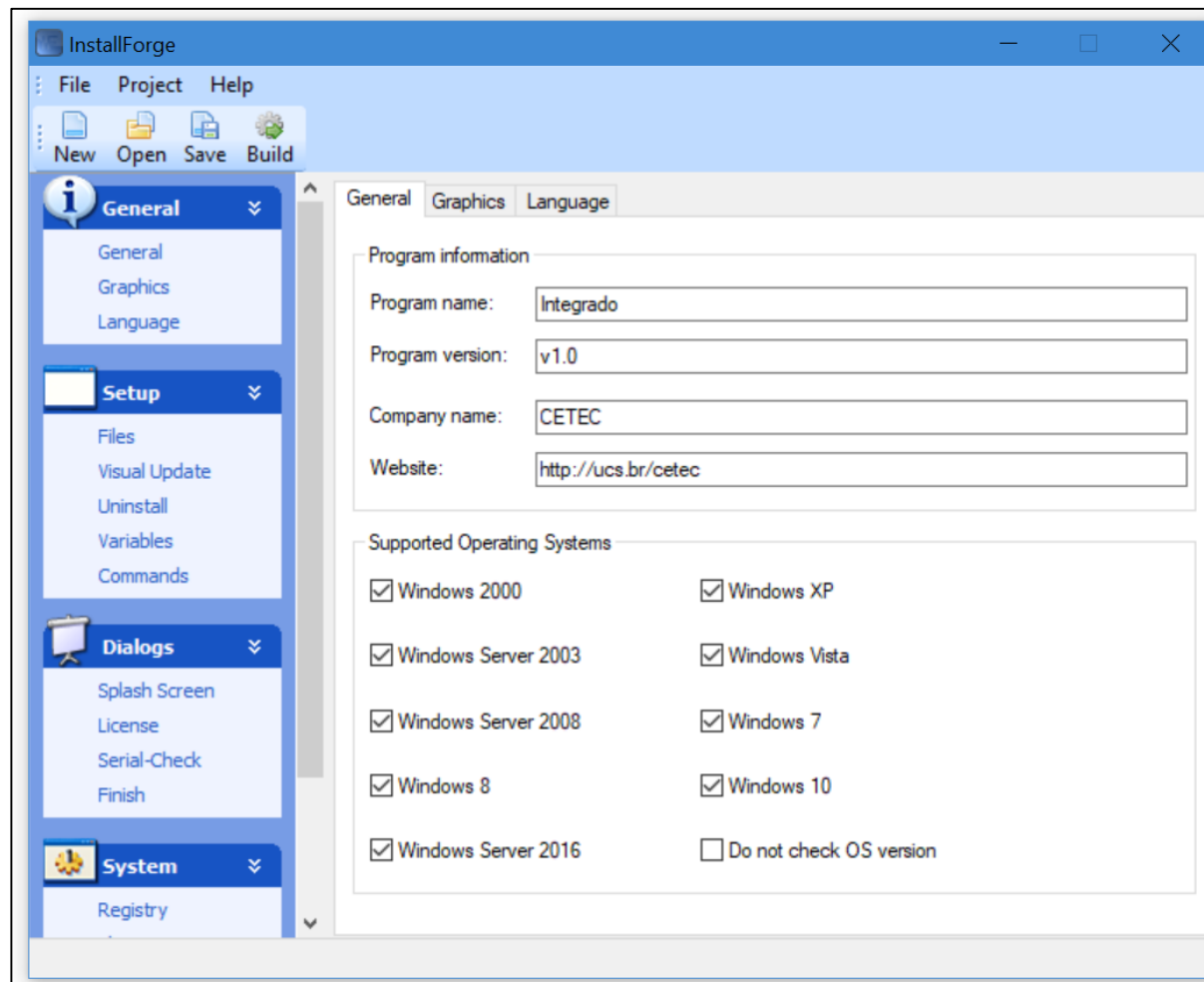
Download do InstallForge



<http://gg.gg/instador>

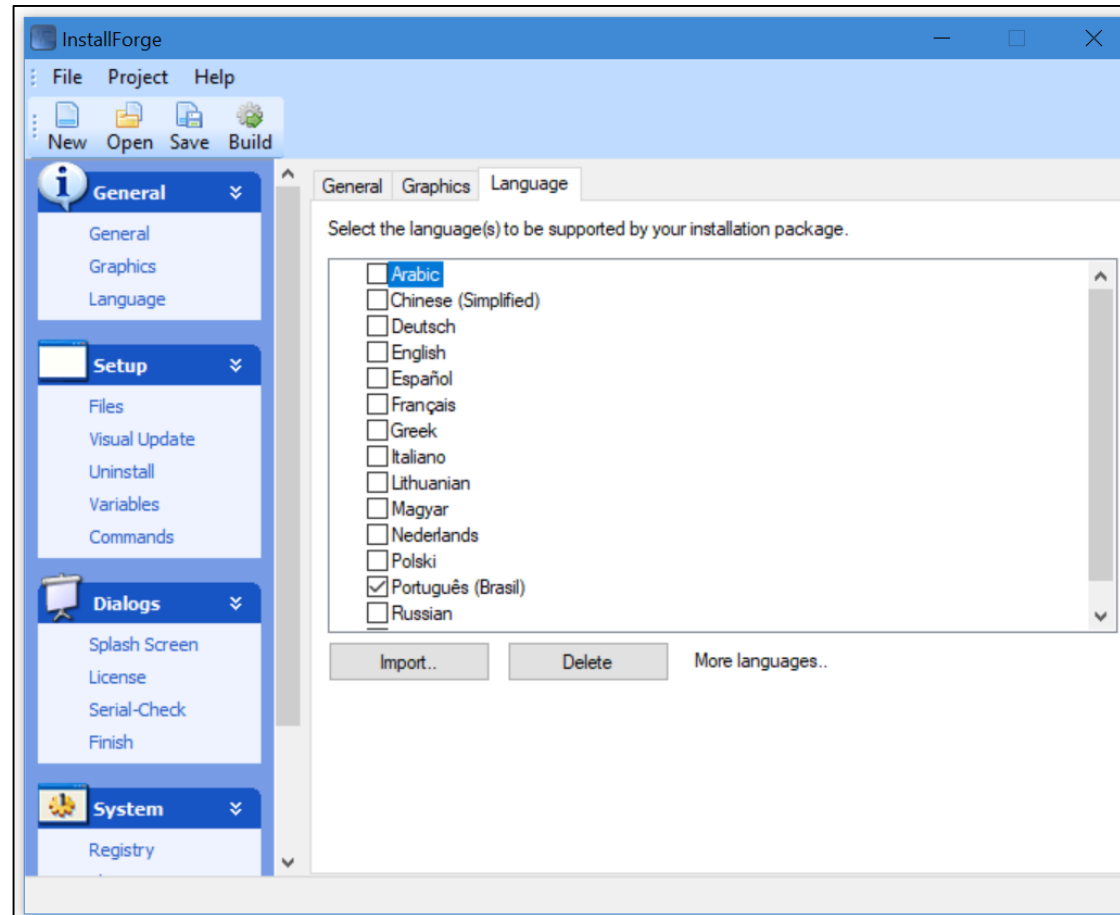


Gerando Instalador



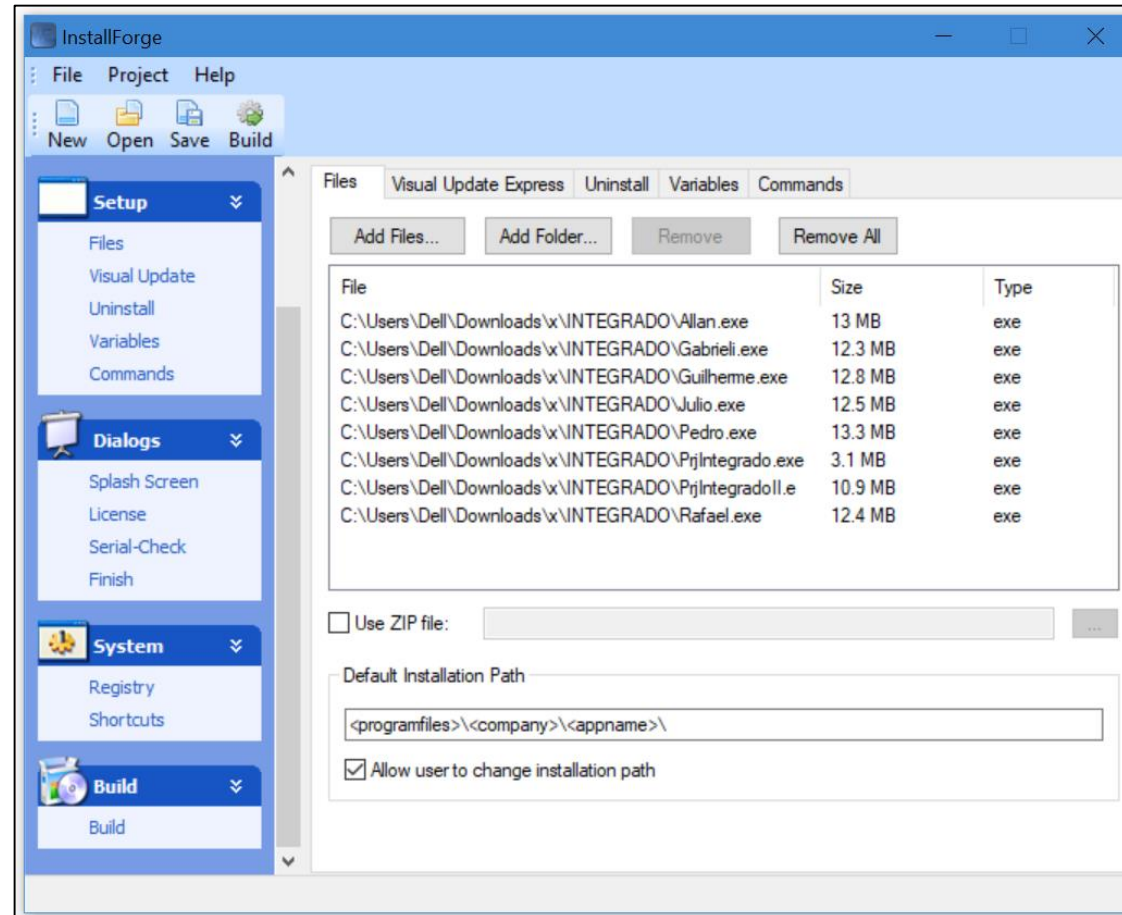
← Defina os dados do programa

Gerando Instalador



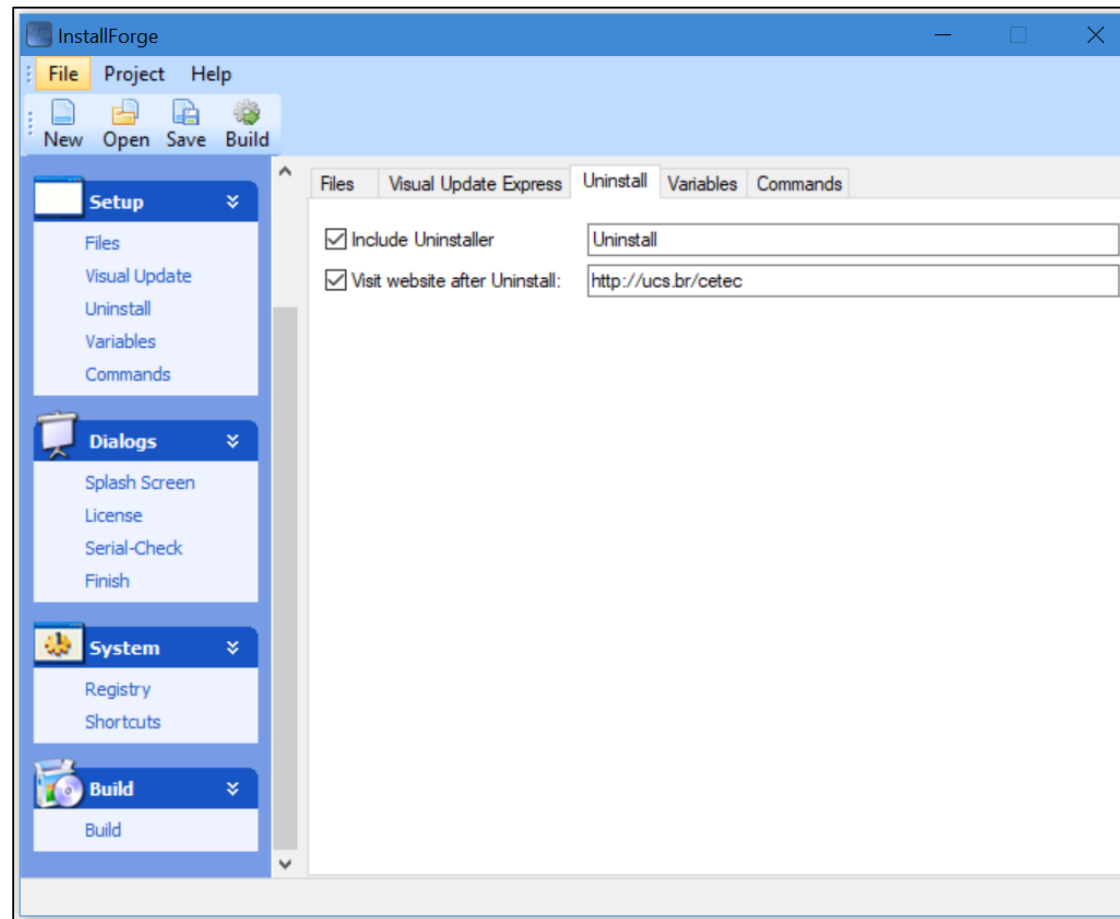
Defina o Idioma

Gerando Instalador



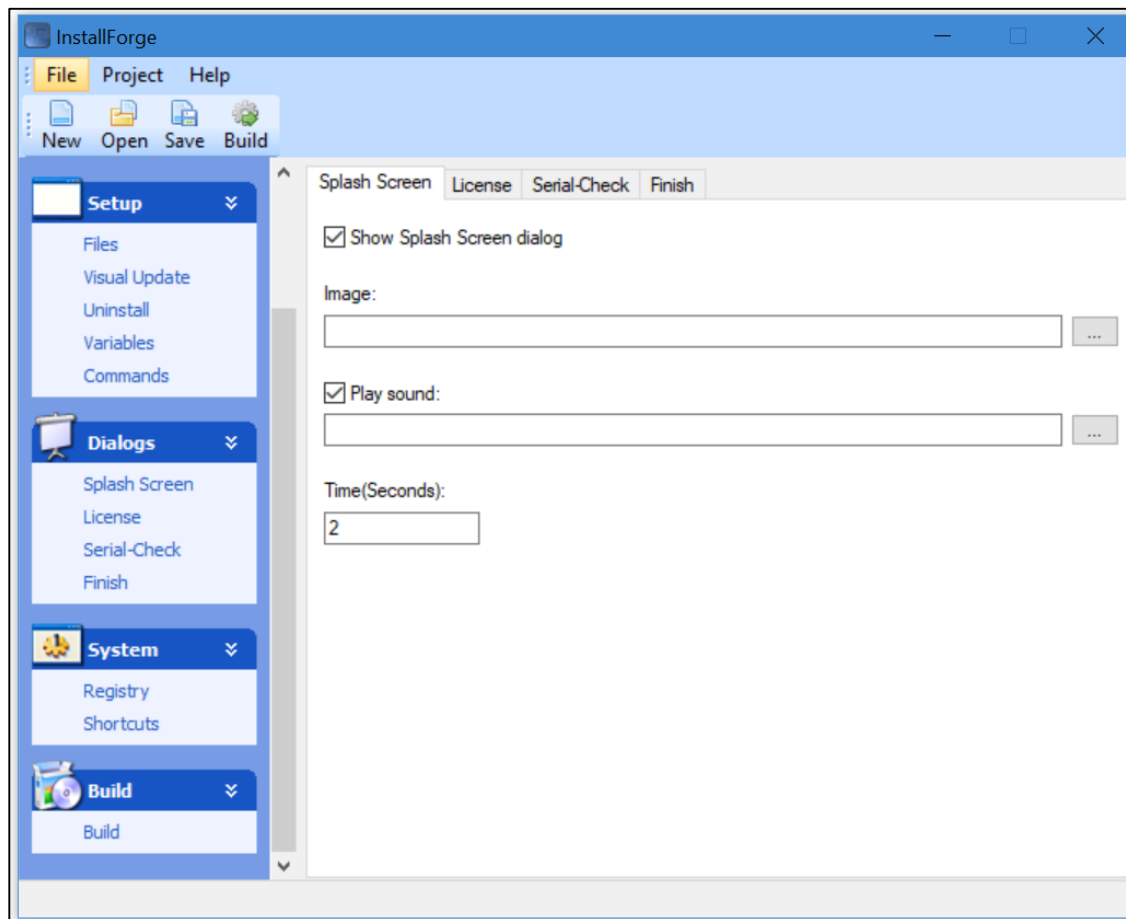
Adicione o programa

Gerando Instalador



Adicione o Uninstaller

Gerando Instalador



Adicione o **logo** do **CETEC**

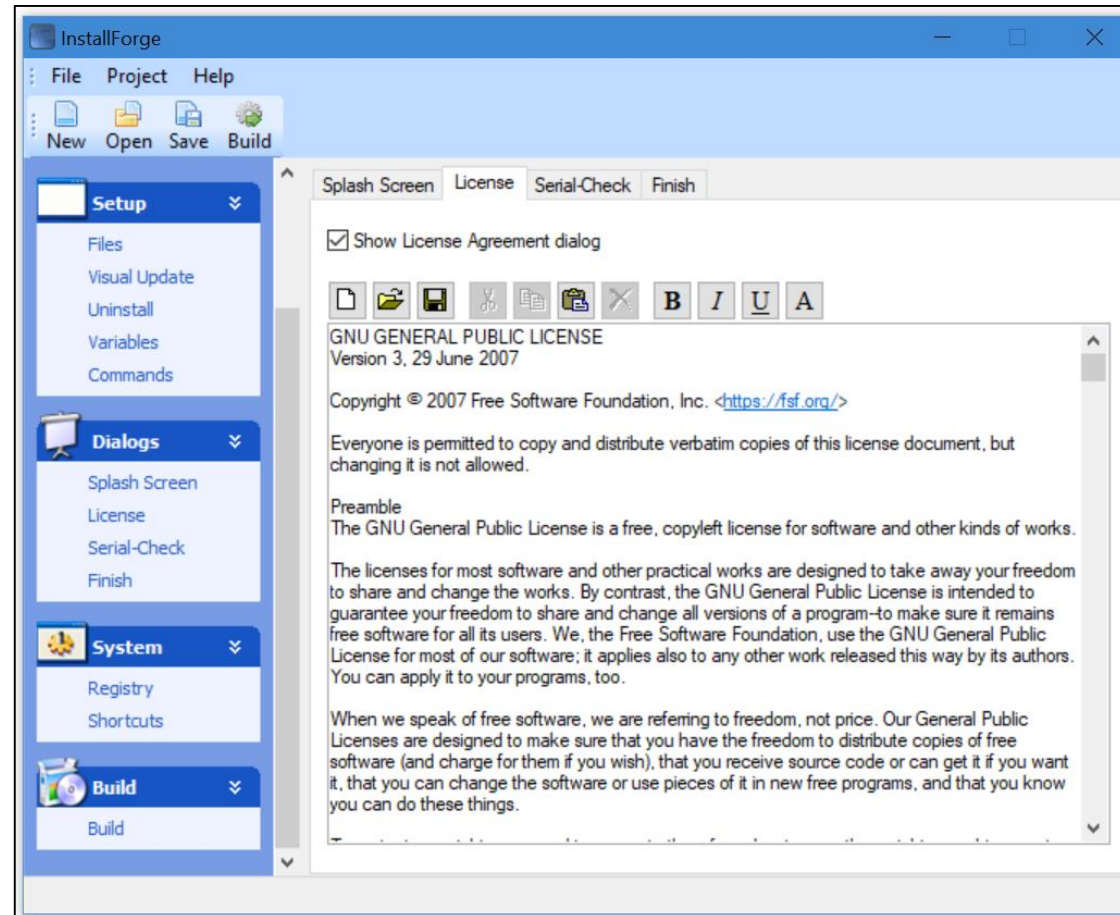


Adicione uma **música**



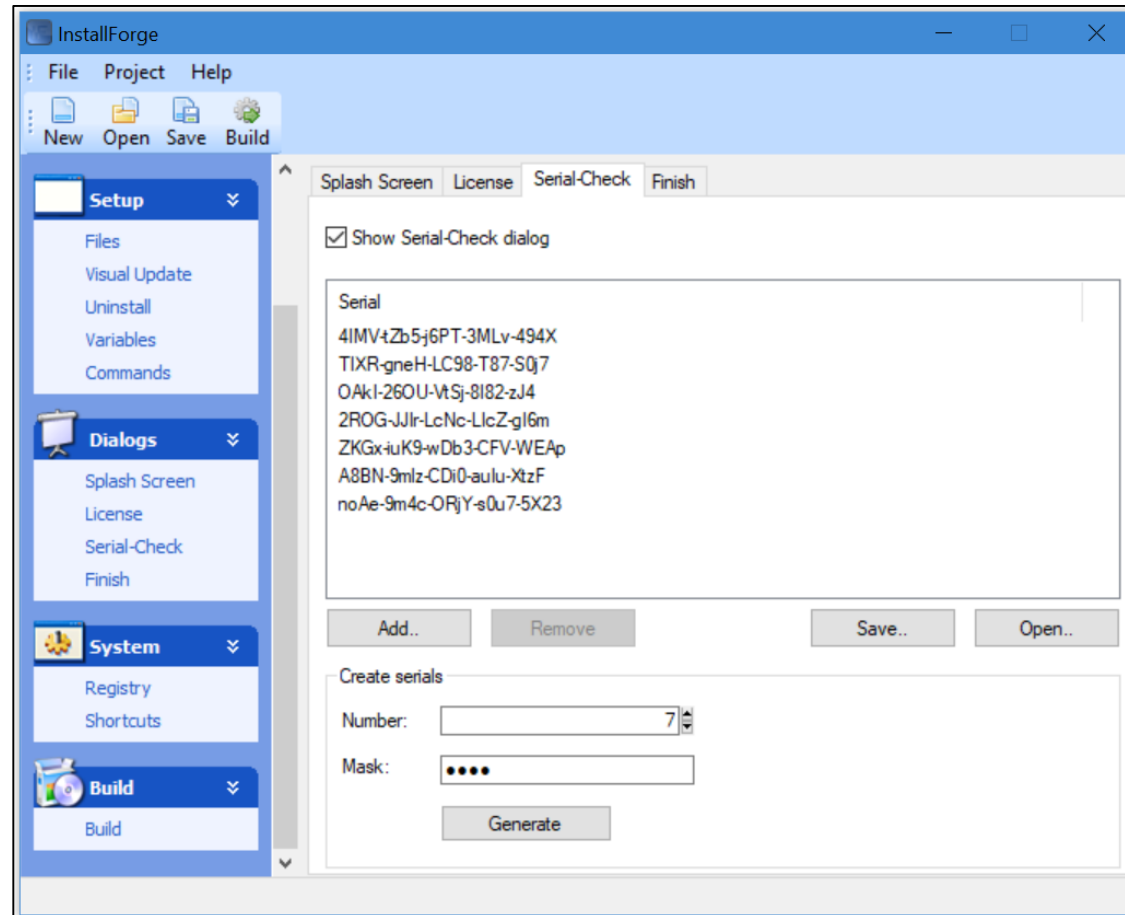
Defina um **tempo inicial**

Gerando Instalador



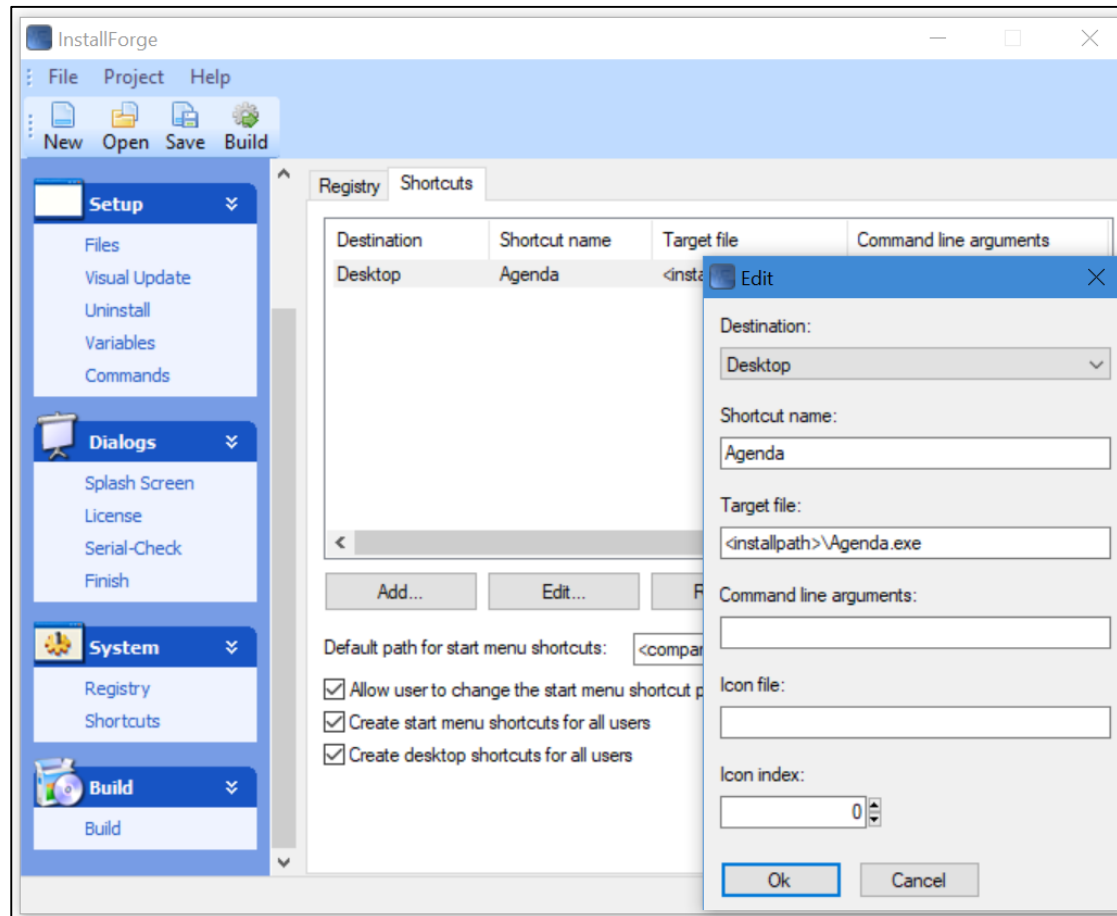
Defina uma **Licença** de **uso**

Gerando Instalador



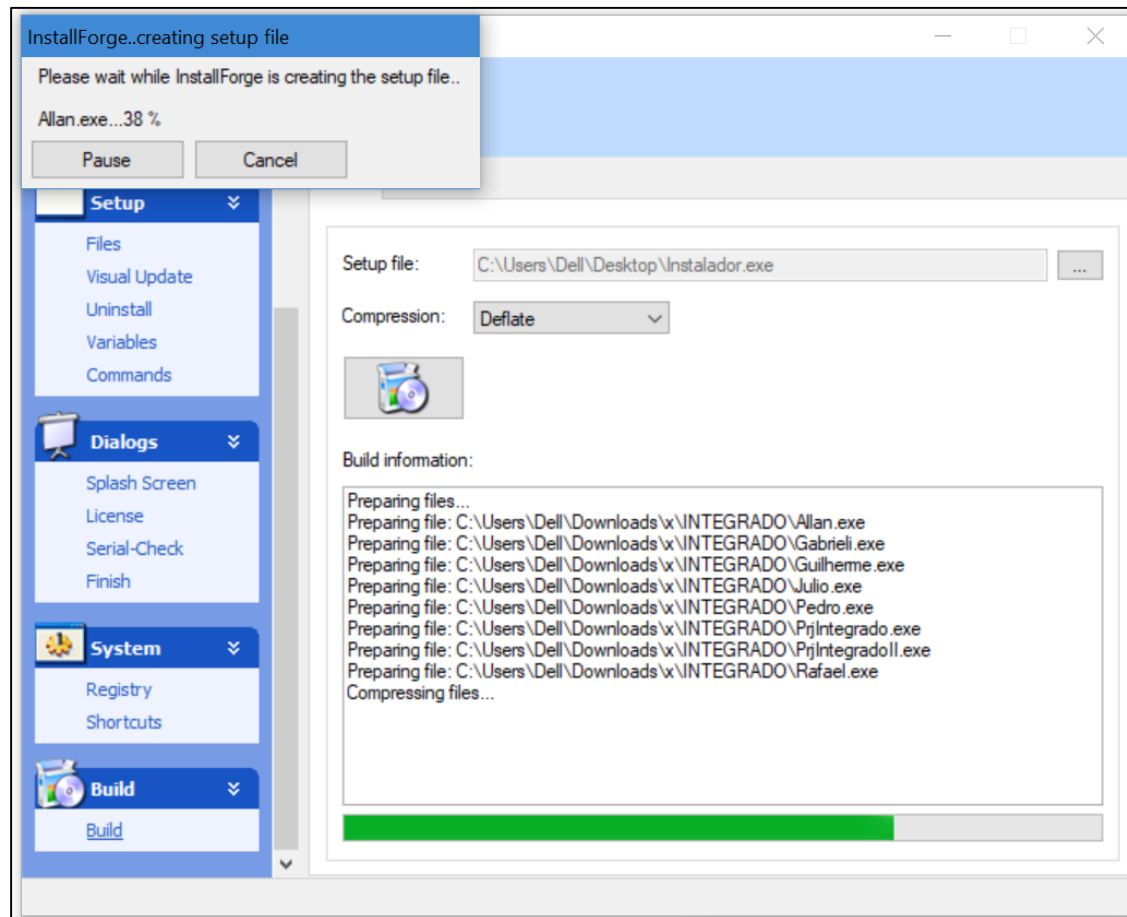
Defina um conjunto de **seriais**

Gerando Instalador



Gerando atalho do programa

Gerando Instalador



← Gerando instalador

Realizando Instalação

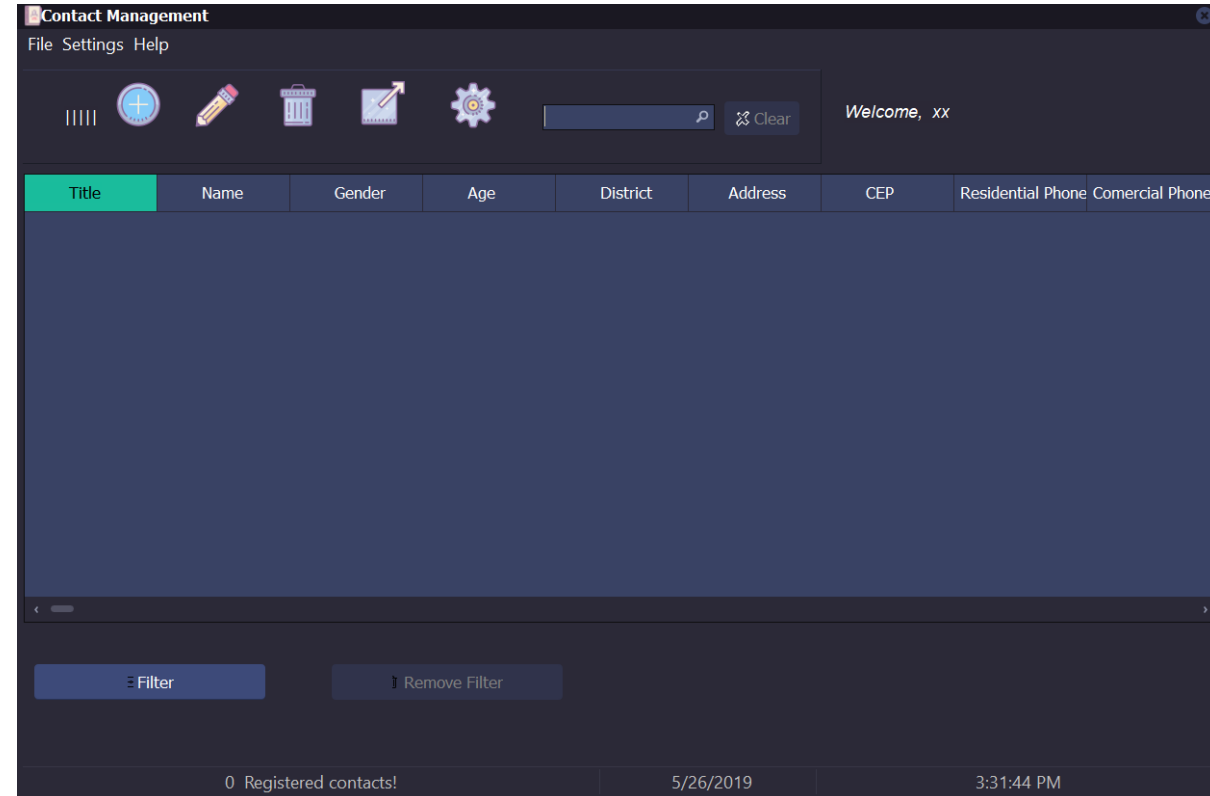
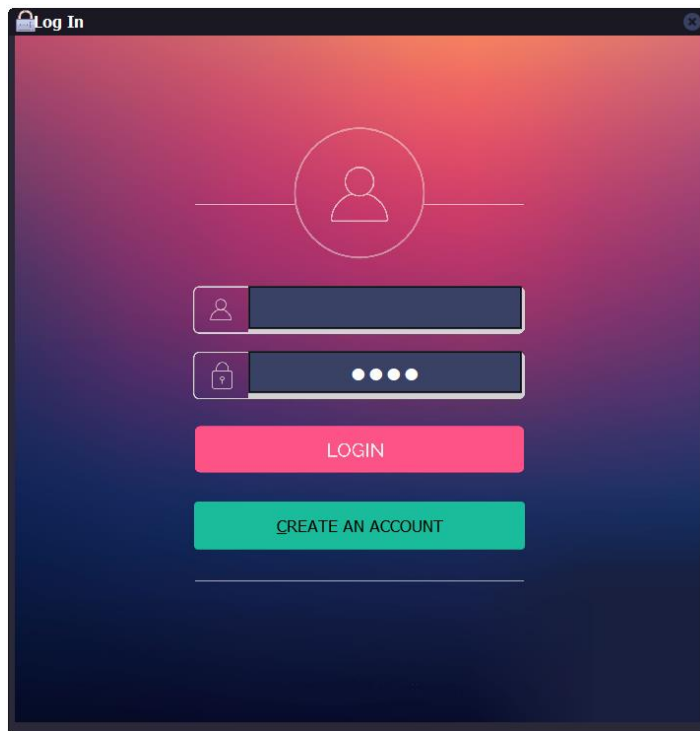


Instale o executável que você gerou no computador

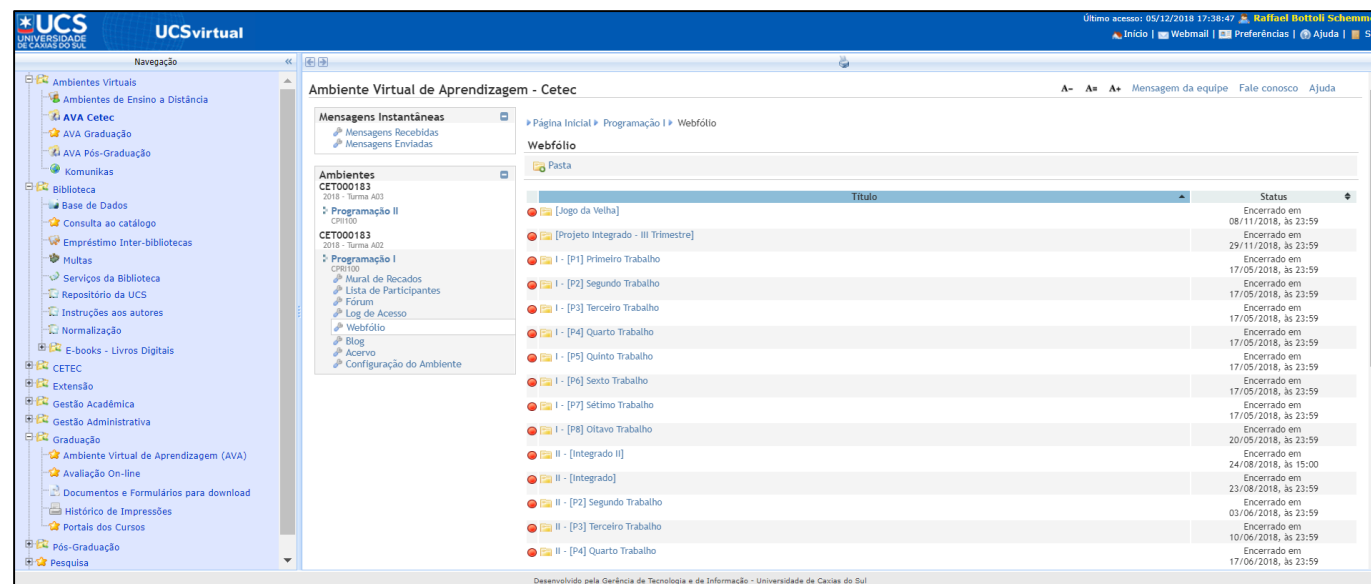


Decimo Nono Trabalho da Disciplina (3TIX)

- Agenda de Contatos Multi Form com Arquivos



Publicação do TIX no AVA



Último acesso: 05/12/2018 17:38:47 **Raffael Bottoli Schiemmer**
Início | Webmail | Preferências | Ajuda | Sair

UCSvirtual

Ambiente Virtual de Aprendizagem - Cetec

Mensagens Instantâneas
Mensagens Recebidas
Mensagens Enviadas

Ambientes
CET000183
2018 - Turma A03
Programação II
CET000183
2018 - Turma A02
Programação I

Webfólio

Título	Status
[Jogo da Velha]	Encerrado em 08/11/2018, às 23:59
[Projeto Integrado - III Trimestre]	Encerrado em 29/11/2018, às 23:59
I - [P1] Primeiro Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P2] Segundo Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P3] Terceiro Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P4] Quarto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P5] Quinto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P6] Sexto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P7] Sétimo Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P8] Oitavo Trabalho	Encerrado em 20/05/2018, às 23:59
II - [Integrado II]	Encerrado em 24/08/2018, às 15:00
II - [Integrado]	Encerrado em 23/08/2018, às 23:59
II - [P2] Segundo Trabalho	Encerrado em 03/06/2018, às 23:59
II - [P3] Terceiro Trabalho	Encerrado em 10/06/2018, às 23:59
II - [P4] Quarto Trabalho	Encerrado em 17/06/2018, às 23:59

Desenvolvido pela Gerência de Tecnologia e de Informação - Universidade de Caxias do Sul

Webfólio do Ambiente Virtual de Aprendizagem (CETEC)

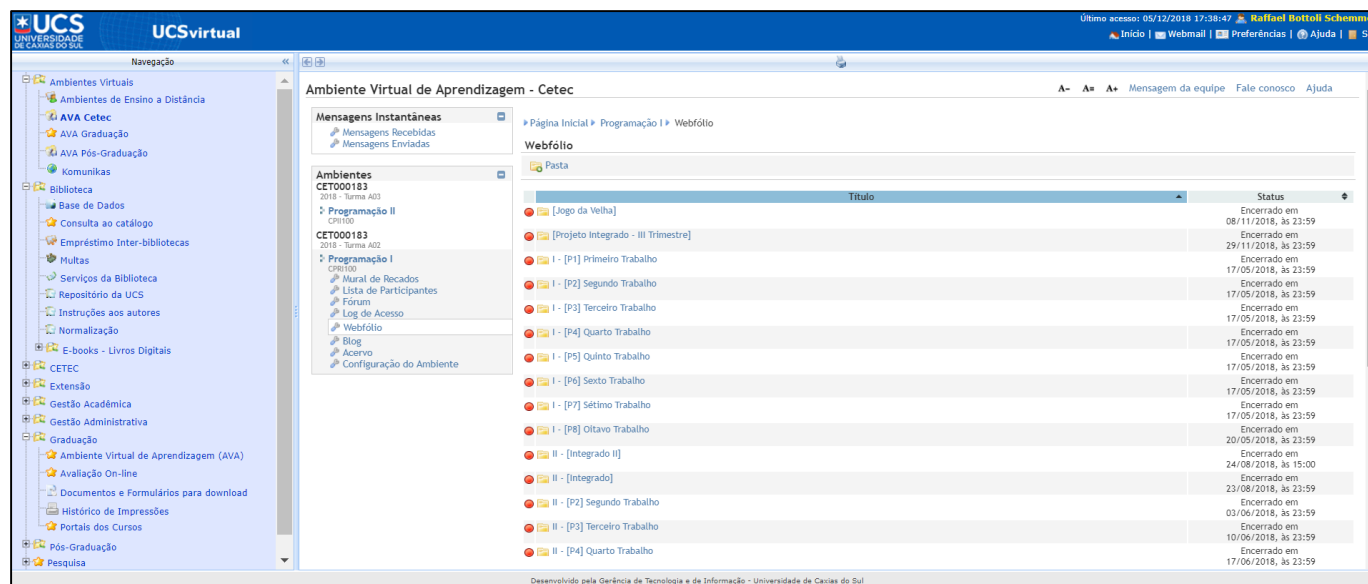
Vigésimo Trabalho da Disciplina (3TXX)



Alunos devem definir/executar um projeto Delphi
(Utilizando todos os componentes VCL da disciplina)



Publicação do TXX no AVA



Último acesso: 05/12/2018 17:38:47 **Rafael Bottoli Schiemmer**
Início | Webmail | Preferências | Ajuda | Sair

UCSvirtual

Ambiente Virtual de Aprendizagem - Cetec

Mensagens Instantâneas
Mensagens Recebidas
Mensagens Enviadas

Ambientes
CET000183
2018 - Turno A02
Programação II
CET000183
2018 - Turno A02
Programação I

Webfólio

Título	Status
[Jogo da Velha]	Encerrado em 08/11/2018, às 23:59
[Projeto Integrado - III Trimestre]	Encerrado em 29/11/2018, às 23:59
I - [P1] Primeiro Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P2] Segundo Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P3] Terceiro Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P4] Quarto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P5] Quinto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P6] Sexto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P7] Sétimo Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P8] Oitavo Trabalho	Encerrado em 20/05/2018, às 23:59
II - [Integrado II]	Encerrado em 24/08/2018, às 15:00
II - [Integrado]	Encerrado em 23/08/2018, às 23:59
II - [P2] Segundo Trabalho	Encerrado em 03/06/2018, às 23:59
II - [P3] Terceiro Trabalho	Encerrado em 10/06/2018, às 23:59
II - [P4] Quarto Trabalho	Encerrado em 17/06/2018, às 23:59

Webfólio do Ambiente Virtual de Aprendizagem (CETEC)