INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO – CÂMPUS CUBATÃO

BÁRBARA OLIVEIRA GRASSE

BEATRIZ BASTOS BORGES

CAMILLY VICTÓRIA MACHADO GONZAGA PEREIRA

ESTHER DOS SANTOS MARTINS

GIOVANNA SANTANA PENNISI

JOÃO VICTOR ALVES DE SOUZA E SILVA

JULIA PASSOS SILVA

KAUÊ DIAS SILVA

MARCELLA RICOY CURCI DE MOURA

RODRIGO FREITAS DA MATA CABRAL

A UTILIZAÇÃO DE UMA ASSISTENTE DE VOZ NO ENSINO DE PROGRAMAÇÃO

CUBATÃO/SP
2022

BÁRBARA OLIVEIRA GRASSE

BEATRIZ BASTOS BORGES

CAMILLY VICTÓRIA MACHADO GONZAGA PEREIRA

ESTHER DOS SANTOS MARTINS

GIOVANNA SANTANA PENNISI

JOÃO VICTOR ALVES DE SOUZA E SILVA

JULIA PASSOS SILVA

KAUÊ DIAS SILVA

MARCELLA RICOY CURCI DE MOURA

RODRIGO FREITAS DA MATA CABRAL

CTII 450 / 448

A UTILIZAÇÃO DE UMA ASSISTENTE DE VOZ NO ENSINO DE PROGRAMAÇÃO

Projeto apresentado ao Curso Técnico em Informática Integrado ao Ensino Médio para a disciplina de Projeto de Sistemas (PJS) do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP).

Disciplina: Prática de Projeto de Sistemas.
Docente: Maurício Neves Asenjo.

CUBATÃO/SP
2022

SUMÁRIO

1. INTRODUÇÃO

Estima-se que o interesse em torno de aspectos relacionados ao ensino e aprendizagem de programação começa no ano de 1970, quando a escolha da linguagem, do paradigma de programação e do que deveria ser ensinado em cursos introdutórios já eram tópicos presentes nas principais arenas de debates (GRIES, 1974) (SCHNEIDER, 1978).

Buscou-se desde sempre compreender o que determinava o sucesso em programação. Descobriu-se que o entendimento da matemática e a habilidade de observar e generalizar conceitos; de utilizar o pensamento lógico e resolver problemas são fatores que facilitam novatos a programar (BENNEDSEN; CASPERSEN, 2005) (PERRENET; KAASENBROOD, 2006).

Ultimamente, o aprendizado de linguagens de programação tem se mostrado fundamental para a atuação eficiente em diversas áreas do conhecimento e do meio profissional, incluindo o mercado financeiro. Entender como os algoritmos funcionam e como programá-los é o ponto mais importante para alcançar tal aprendizado.

Entretanto, essa tarefa não é fácil. Muitos aspectos contribuem para o fracasso dos estudantes diante dessa competência, o que torna esse estudo muito frustrante. Esse tipo de aprendizado consome tempo, exige esforço, rotina de treinos e práticas e a capacidade de lidar constantemente com seus erros. Ao longo dos anos, pesquisadores reconheceram que a aprendizagem de programação também era afetada por fatores motivacionais, dentre eles, o interesse.

A nossa instituição de ensino (Instituto Federal de Educação, Ciência e Tecnologia de São Paulo) busca, anualmente, por estudantes que almejam entender a programação, além dos que se sentem atraídos pela área de T.I. Com isso, no ano de 2022, vimos que há muitas pessoas interessadas em aprender o básico mas que não sabem por onde começar.

O nosso projeto tem como objetivo facilitar o caminho dos alunos que desejam pisar fundo no mundo dos códigos. Uma das coisas mais interessantes da tecnologia é a criação de assistentes de voz para que sua máquina possa conversar com você.

Com isso, como podemos facilitar a compreensão na programação? Como será o funcionamento da aplicação e do guia que será acessível para que qualquer indivíduo possa usar sem encontrar dificuldades? Essas são uma das questões que são prioridades para resolver durante o desenvolvimento do trabalho.

2. OBJETIVOS

O objetivo desse projeto é facilitar a vida de pessoas da área de T.I. mostrando os passos para a criação de uma assistente virtual guiada por comando de voz de modo que seja fácil a compreensão para que pessoas de todas as idades possam usufruir ao máximo do projeto.

2.1. OBJETIVOS ESPECÍFICOS

- Construção de uma aplicação acessível para o público-alvo;
- Guiar o público-alvo durante todo o processo;
- A apostila pretende ser de fácil compreensão.

3. SOBRE A ASSISTENTE DE VOZ

Há mais de 50 anos os seres humanos questionam-se sobre a possibilidade de interagir com máquinas. A Inteligência Artificial foi desenvolvida, principalmente, durante a Segunda Guerra Mundial. Alan Turing, um dos pioneiros na inteligência artificial, escreveu em 1950 um artigo respondendo à pergunta "As máquinas podem pensar?". Nesse artigo, nomeado como Computing Machinery and Intelligence, Turing busca mostrar que uma máquina associada a tecnologias de valor pode perfeitamente tomar o lugar de uma pessoa em um diálogo e demais atividades, se seguir condições pré-estabelecidas.

É possível considerar o Shoebox como o primeiro assistente virtual desenvolvido. Ele foi criado em 1962 por William C. Dersch, podia responder a 16 palavras e responder expressões matemáticas.

Atualmente, os projetos envolvendo inteligência artificial evoluíram, mas ainda não ao ponto de poder substituir completamente o homem.

Um assistente virtual é um software criado com o objetivo de executar tarefas ou tomar decisões a fim de auxiliar os seres humanos. Eles podem responder a comandos de voz ou texto. São capazes de buscar informações na rede, regular agendas, fazer recomendações baseadas no perfil do usuário, etc, mas são limitados. A complexidade característica na criação dessas aplicações fazem com que esse processo seja demorado e que muitas vezes não obtenha os

resultados esperados, já que, pela característica de autonomia relativa, um software desse tipo pode produzir erros imprevisíveis. São criados a partir de softwares de inteligência artificial e machine learning. Alguns exemplos são a Siri (*Apple*) e a Alexa (*Amazon*).

A Alexa, por exemplo, é uma tecnologia criada em 2014 pela Amazon que interpreta a voz do usuário e executa os comandos correspondentes ao que foi pedido. A informação é transformada em texto a partir do *Alexa Voice Services* (AVS). Ela pode executar tarefas simples como tocar músicas e fazer pesquisas, mas também pode ser utilizada a fim de automatizar a casa dos usuários. Pode ser ligada a lâmpadas inteligentes, a televisões, etc. Outra funcionalidade da Alexa é o Skill, característica que permite um programador definir uma funcionalidade específica para o assistente.

Já a Siri foi desenvolvida pela empresa Siri Inc, que foi fundada pelo SRI International. Ela foi baseada em um projeto anterior a ela (CALO - Cognitive Assistant that Learns and Organizes) e administrado pelo SRI. A Siri pode reconhecer voz, ela analisa as palavras-chave e o contexto para efetuar comandos. Em 2010, o assistente foi comprado pela Apple, consequentemente ela só pode ser usada em sistemas IOS, diferente da Alexa que é compatível com diversos sistemas operacionais como Android, IOS e Amazon Fire OS.

Tanto a Alexa quanto a Siri possuem a limitação de apenas funcionarem quando conectadas à Internet.

Alexa, Siri e qualquer outro assistente virtual utilizam o machine learning, tecnologia utilizada na criação desses softwares, ele permite que a máquina identifique padrões e aprenda com o passar do tempo, assim ficando mais inteligente, ao detectar hábitos, ao responder a questões com base no contexto e gravando preferências.

4. API's E BIBLIOTECAS

4.1. Speech.Recognition

Speech.Recognition ou reconhecimento de fala, é a capacidade de uma máquina ou programa de reconhecer palavras faladas em voz alta e convertê-las em texto legível. O software de reconhecimento de fala básico tem um vocabulário limitado e só pode reconhecer palavras e frases

quando faladas com clareza. Um software mais sofisticado pode lidar com fala natural, sotaques diferentes e vários idiomas.

4.2. Speech.Synthesis

Speech.Synthesis ou síntese de fala, é uma simulação gerada por computador da fala humana. Ele é usado para traduzir informações escritas em informações auditivas mais convenientes, especialmente para aplicativos móveis, como e-mail habilitado para voz e mensagens unificadas.

5. SOBRE A APLICAÇÃO

5.1. USO PRETENDIDO

Nossa aplicação visa auxiliar a como desenvolver uma assistente de funcionamento com comando de voz. Com ela você aprenderá a como criar sua assistente do zero, desde como abrir um Windows Form até os procedimentos necessários para seu funcionamento.

5.2. AMBIENTE DE USO

A aplicação será feita no Visual Studio utilizando apenas da linguagem C#, então todas as instruções dadas a seguir funcionarão apenas com a mesma. Deixando claro que visamos que as pessoas que visualizarem essa apostila já possuem um conhecimento básico da ferramenta.

5.3. RECURSOS PRINCIPAIS

- Interface de interação;
- Biblioteca System.Speech;
- Comandos de voz (Alexa).

6. PESQUISA DE MERCADO

Conforme os dados apresentados nos tópicos anteriores, é necessário compreender o nosso projeto no meio do mercado. Algumas pesquisas mostram que a demanda de pessoas com perfil

tecnológico aptas à área de T.I. vem aumentando, então a facilidade de obter informações corretas e simples se torna extremamente necessária. No mercado, nossa apostila visa atender esse tipo de pessoa para que essa necessidade seja atendida.

O público-alvo a quem esse projeto é direcionado terá acesso a informações indispensáveis para a criação de uma assistente virtual guiada por comando de voz.

Portanto, o aplicativo foi projetado para:

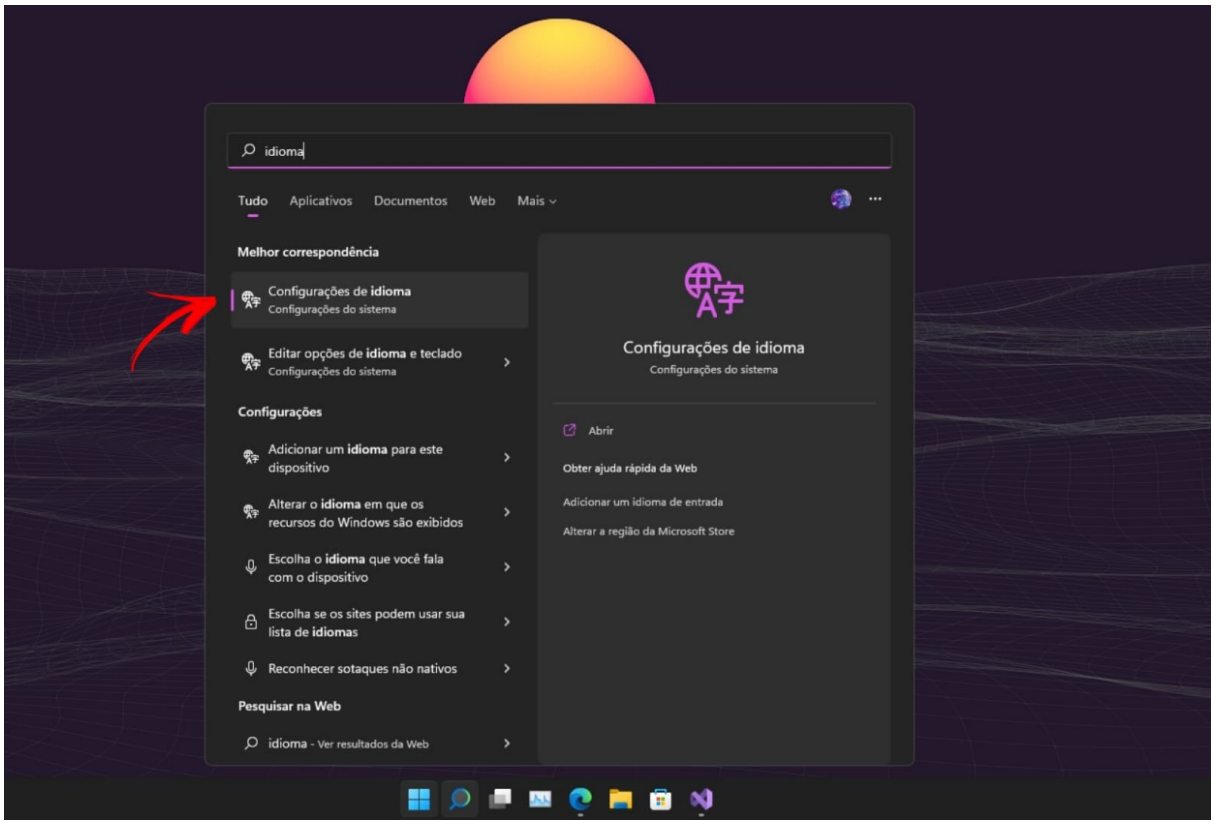
- pessoas com perfil tecnológico aptas à área de TI;
- com faixa etária de 15 anos de idade ou mais;
- que desejam ter conhecimentos básicos sobre a ferramenta Visual Studio.

7. DESENVOLVIMENTO

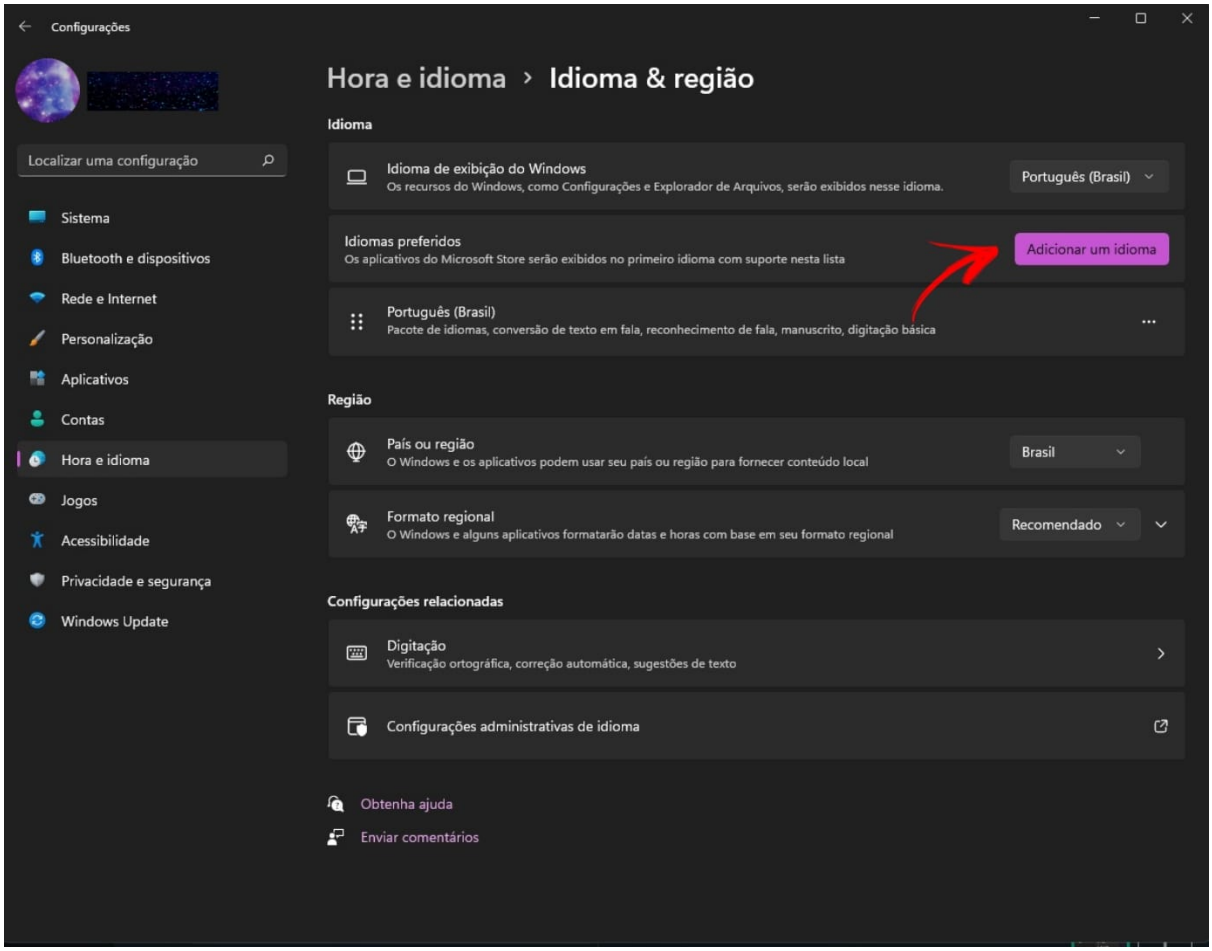
Como nosso programa foi feito desde o início de modo gradual, onde foi testado diversas maneiras de fazê-lo, de fato, funcionar, passamos por algumas configurações e utilizamos algumas bibliotecas específicas, assim como dito no item 4.

7.1. COMO CONFIGURAR O WINDOWS

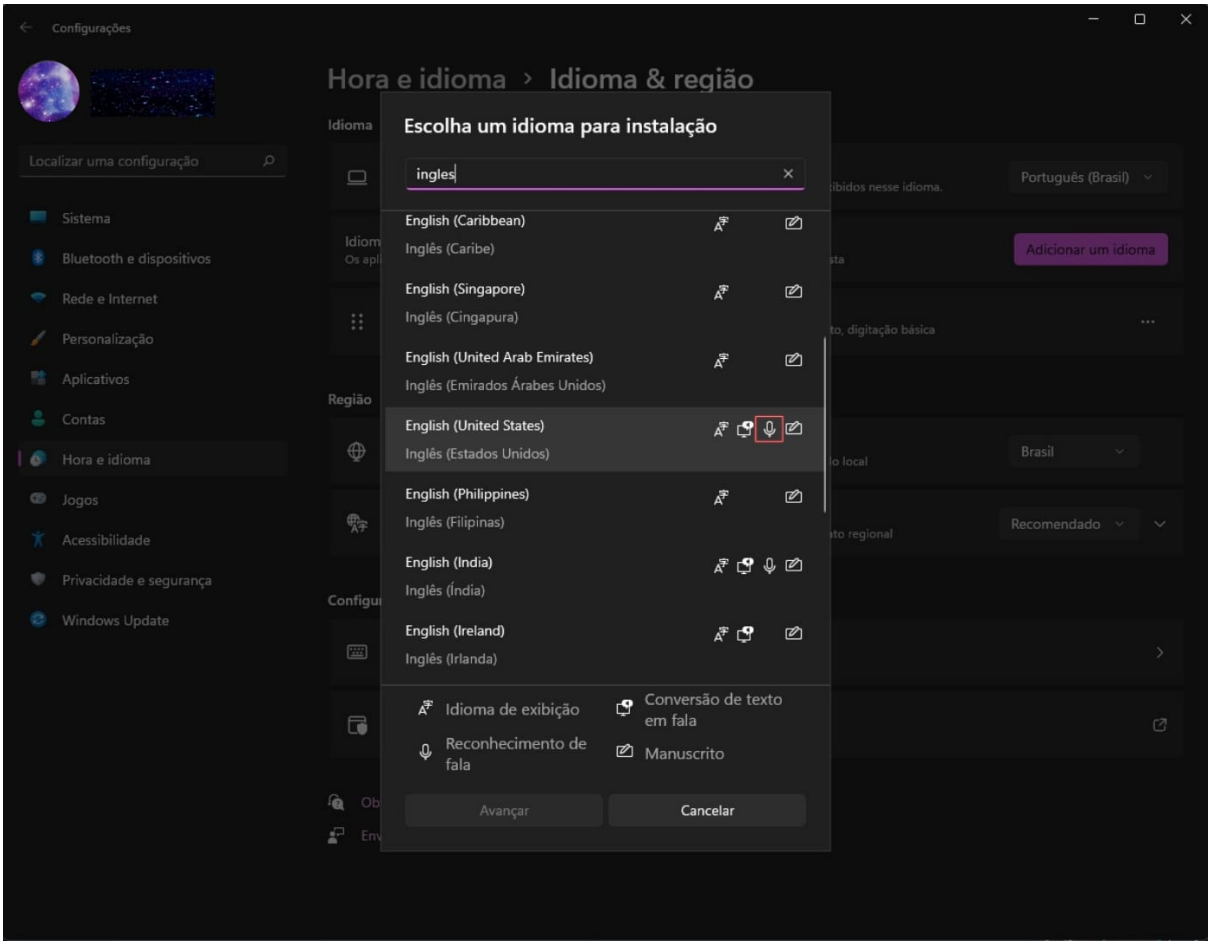
Vá em Configuração de Idioma:



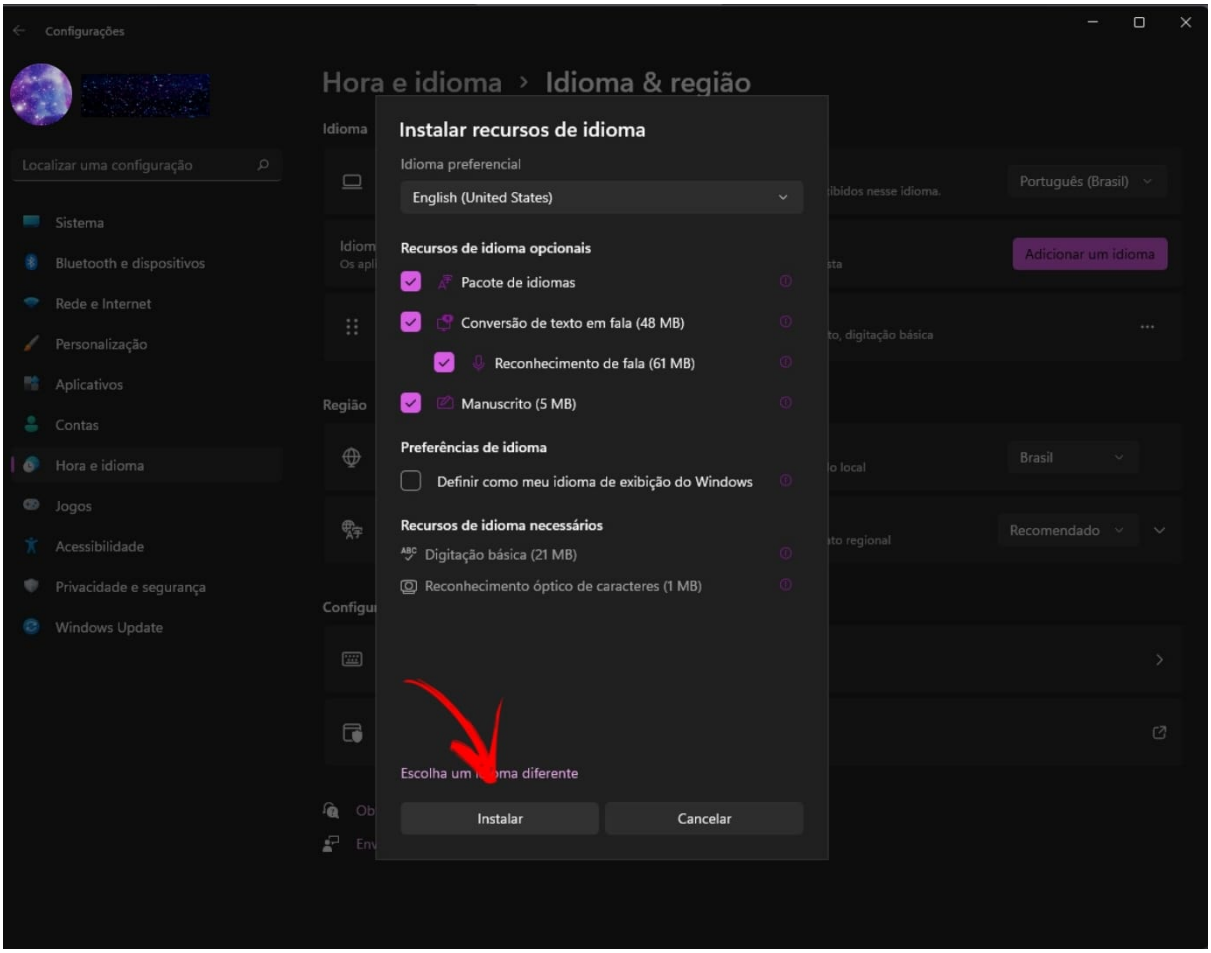
Clique em Adicionar um Idioma:



Após isso, selecione Inglês (United States). É possível ver o símbolo de microfone, o que significa que tem suporte de reconhecedor de voz:



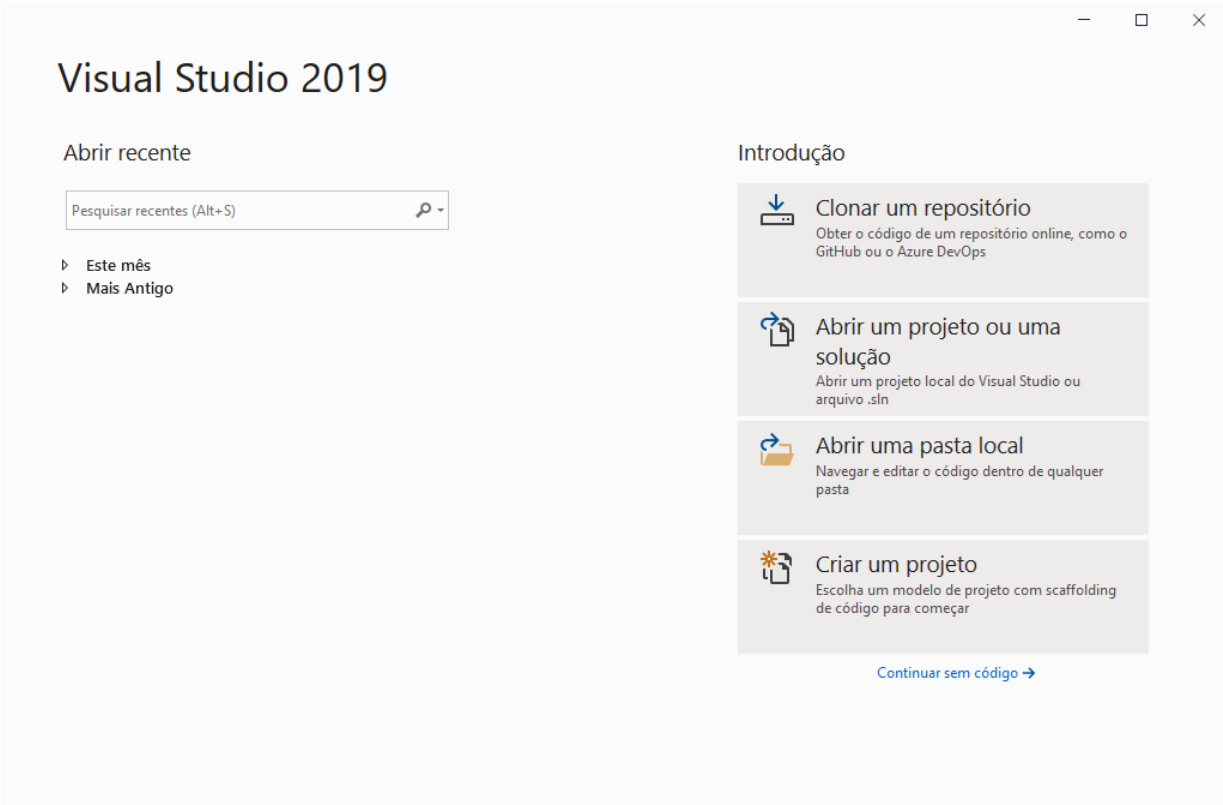
Após essa configuração, clique em instalar:



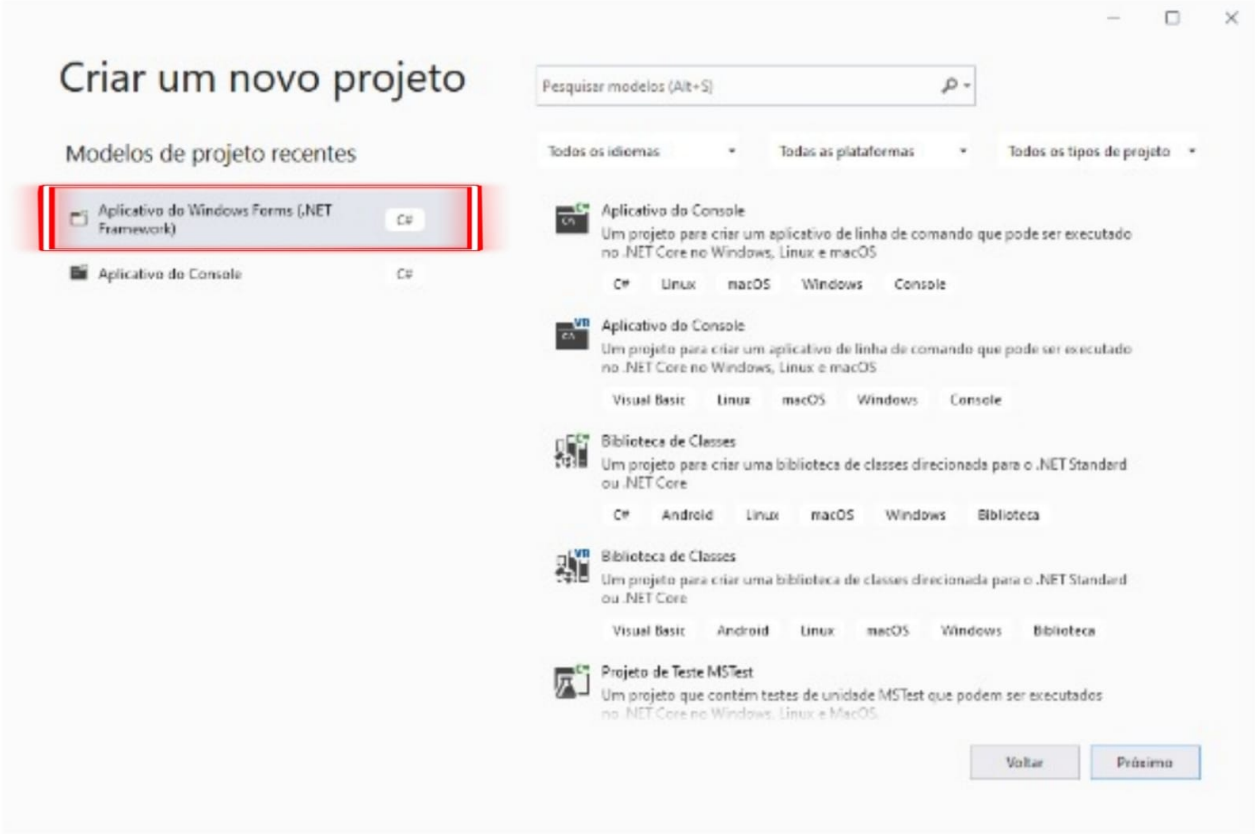
Após a instalação, basta mudar o idioma de exibição do windows para inglês. O computador deverá ser reiniciado.

7.2. COMO CRIAR UM PROJETO WINDOWS FORMS

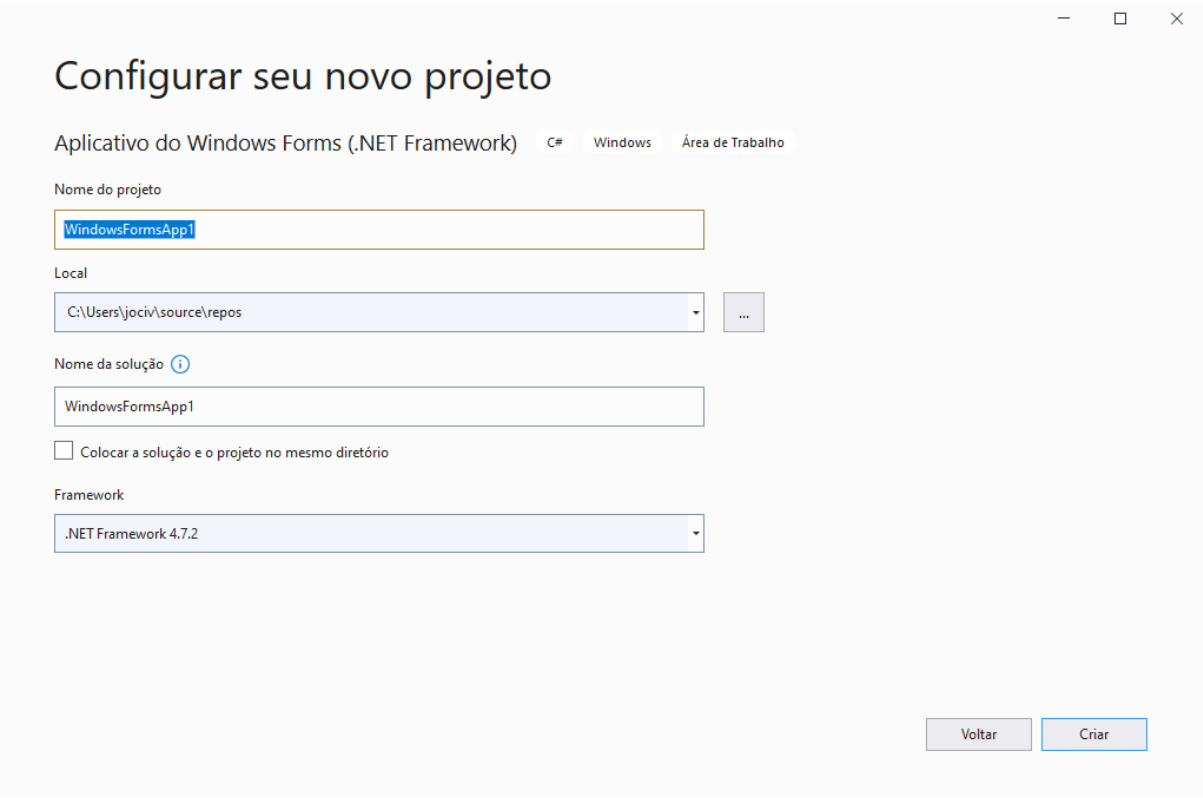
Primeiramente você deverá criar um projeto de aplicativo C#. Para isso abra o Visual Studio e quando a janela inicial abrir vá em **Criar um projeto**:



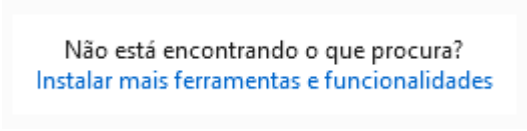
Agora procure a opção **Aplicativo do Windows Forms (.NET Framework)** a selecione e prossiga:



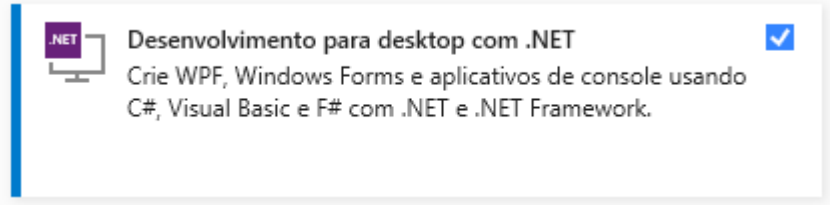
Na próxima janela você poderá modificar o do projeto e o diretório onde ele será salvo caso ache necessário. Feita todas as modificações prossiga com a criação:



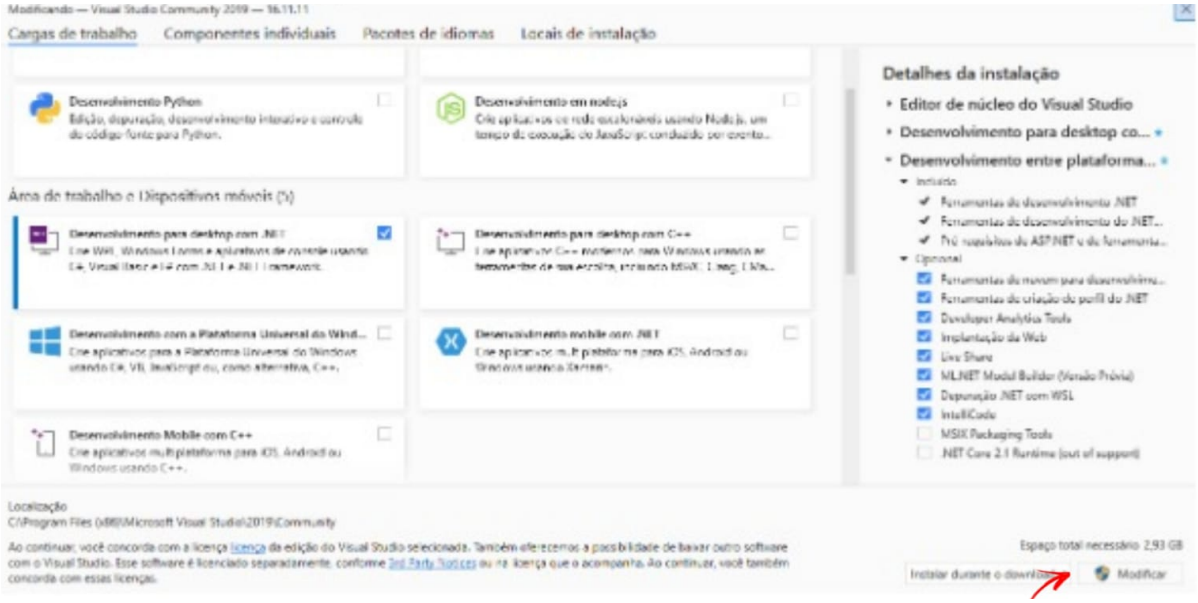
Caso você não ache a opção desejada vá em **Não está encontrando o que procura?** e clique em **Instalar mais ferramentas e funcionalidades**



Quando o Visual Installer abrir, selecione **Desenvolvimento para desktop com .NET**



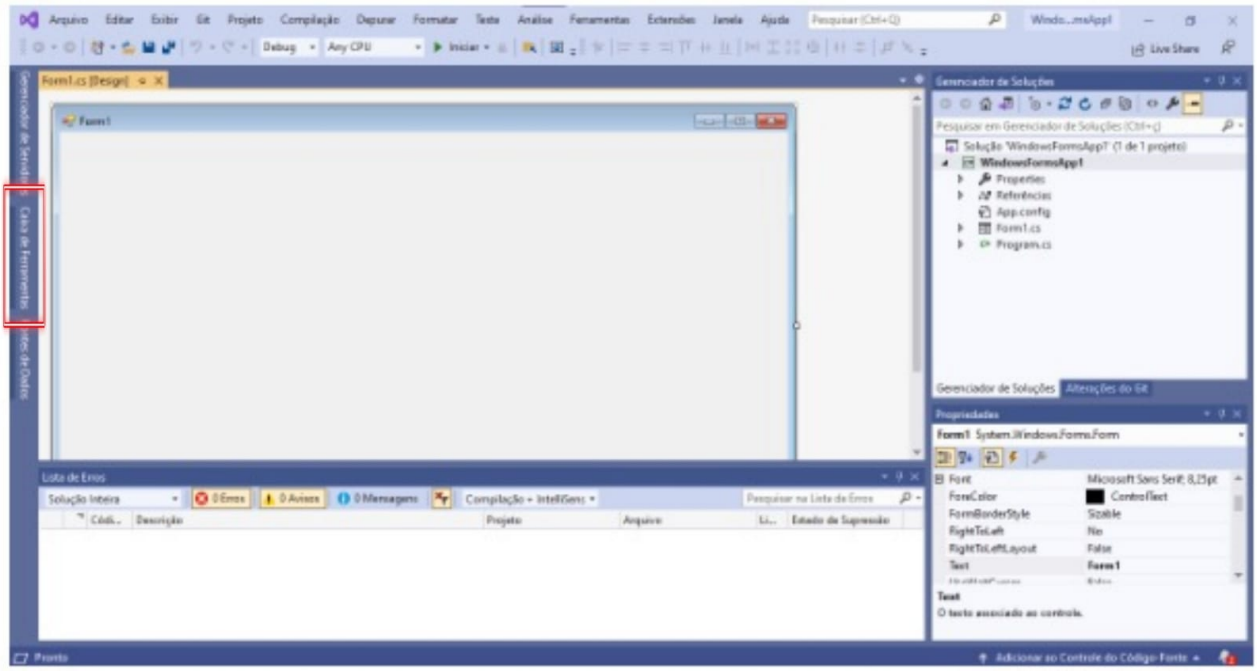
Após isso, selecione **Modificar**. Pode aparecer uma mensagem solicitando para salvar seu trabalho, se esse for o caso faça-o. Em seguida, selecione **Continuar**. Agora é só seguir as instruções anteriores.



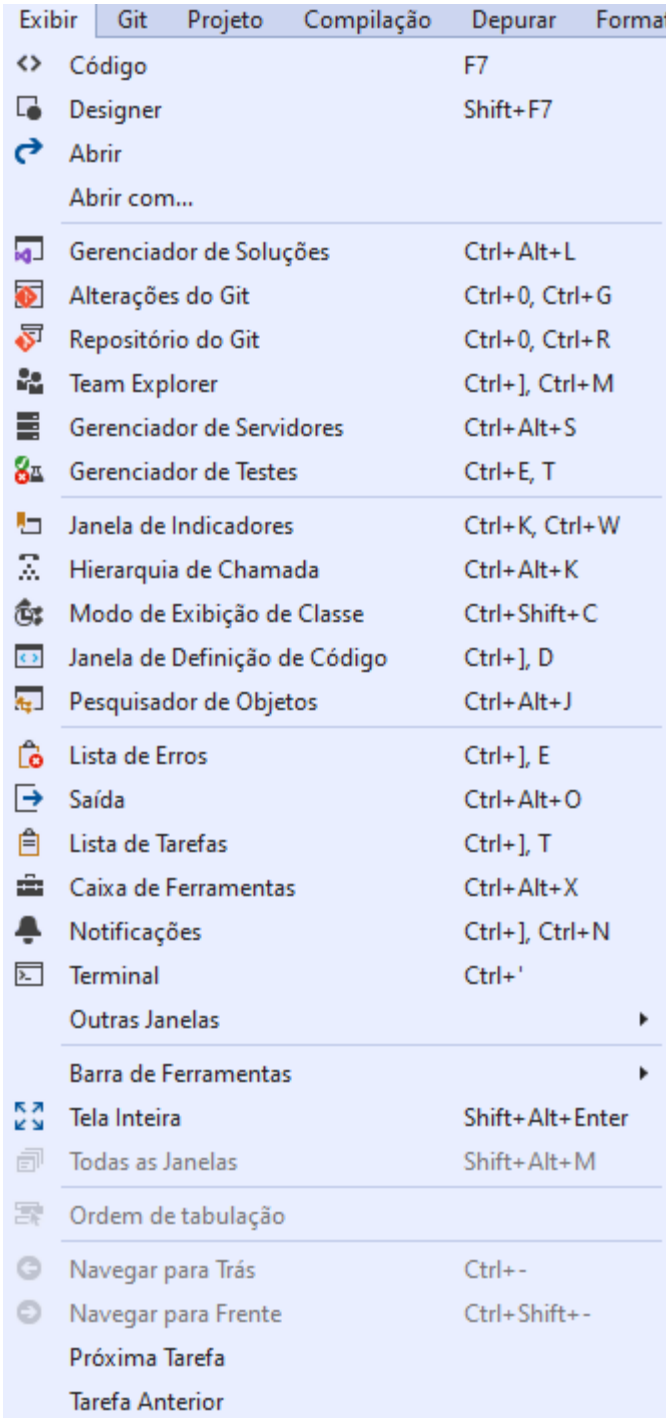
7.3. COMO CONFIGURAR O FORMS

Após a criação do seu projeto um formulário será aberto na tela. Um formulário é uma interface de usuário do Windows. Para o funcionamento durante a execução do aplicativo devemos adicionar controles ao formulário.

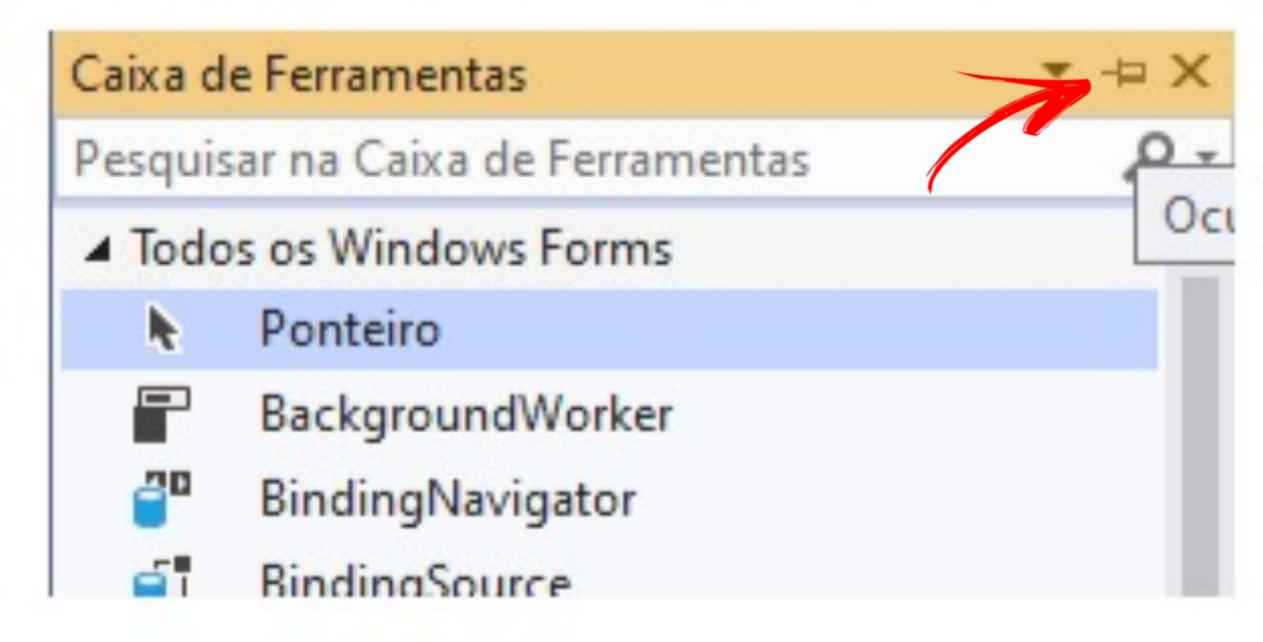
Selecione a **Caixa de Ferramentas**:



Caso não consiga visualizar em sua tela a **Caixa de Ferramentas** você deve ir em exibir localizado na barra do menu e encontre a opção **Caixa de Ferramentas.**

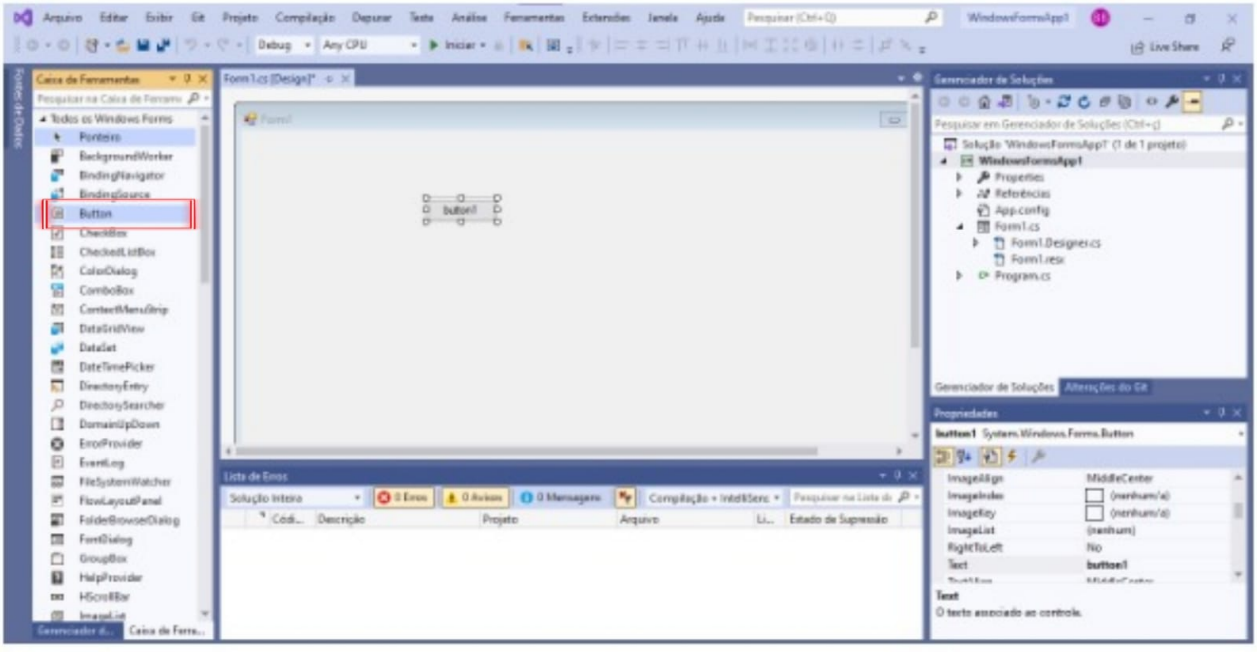


Selecione o pin para deixar essa opção fixa:

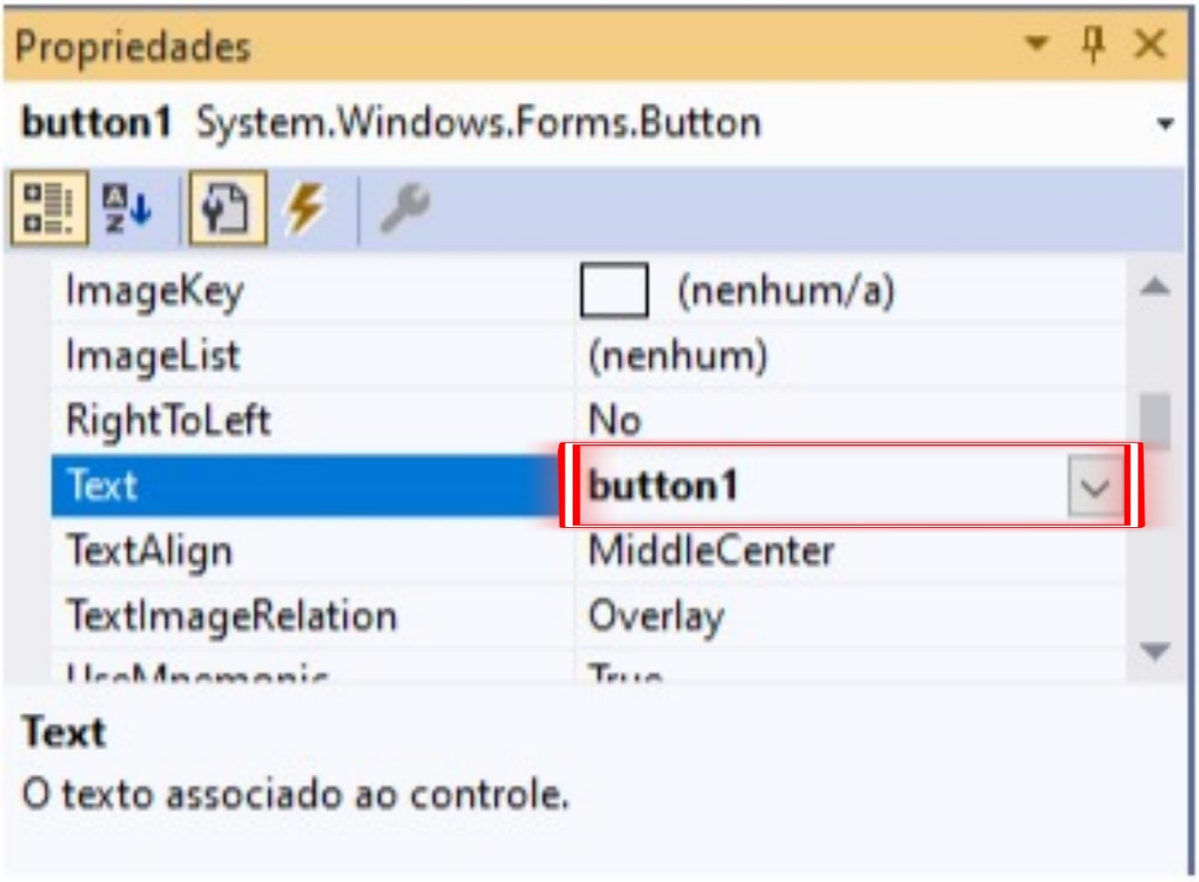


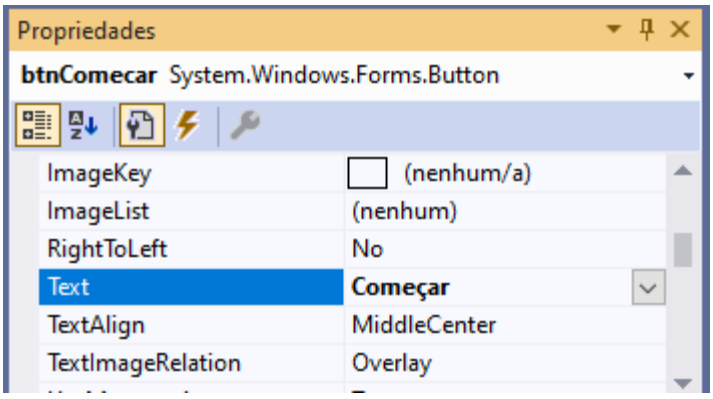
7.3.1. Adicionando button

Encontre o **Button** na aba **Caixa de Ferramentas** e arraste-o até o formulário:



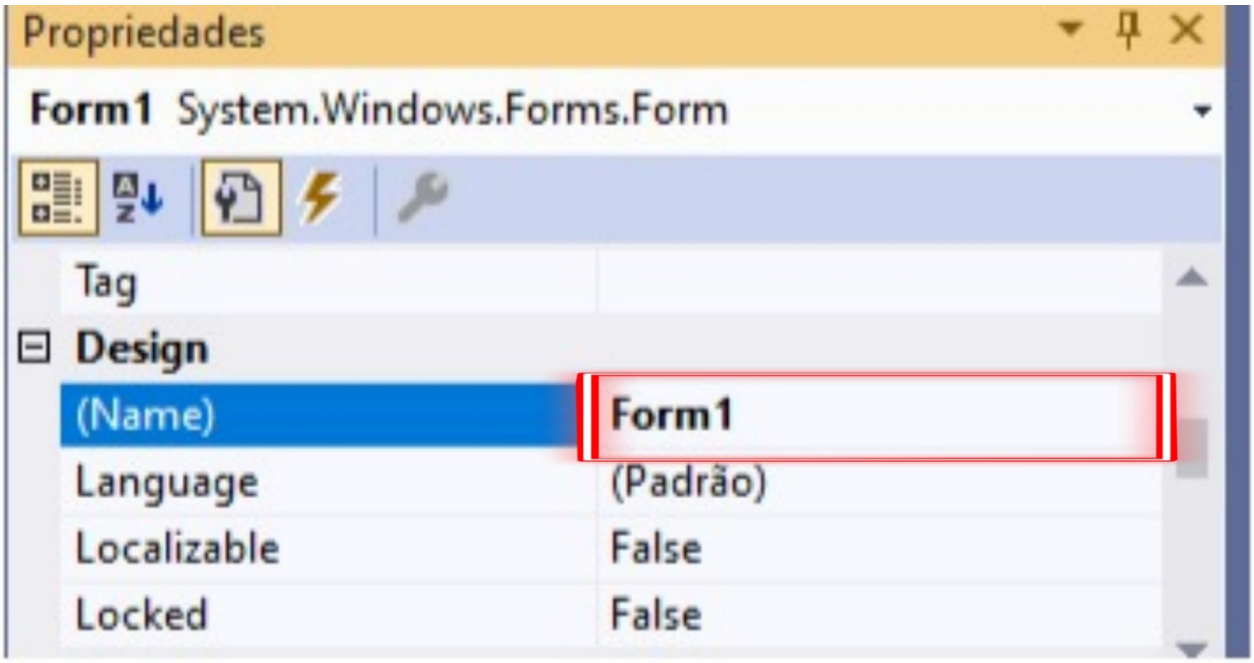
Para mudar o título exibido no botão vá em **Propriedades**, procure **texto** e reescreva o título **button1:**



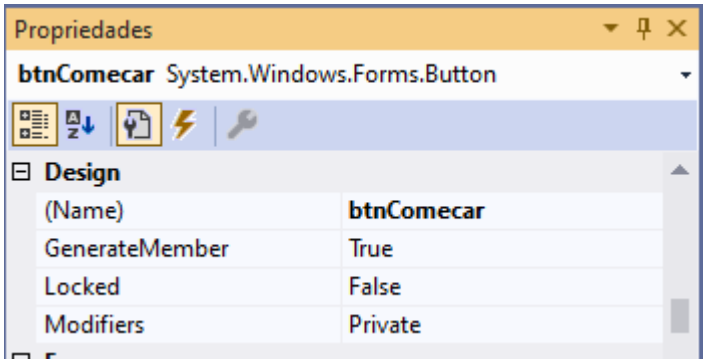


Caso não consiga visualizar a janela **Propriedades**, você deverá ir em **Exibir** e depois em **Propriedade** ou pressionar **F4**.

Também é possível mudar o nome da variável designada ao botão. Para isso vá na sessão **Desing** e escreva o nome desejado na linha **NAME**.



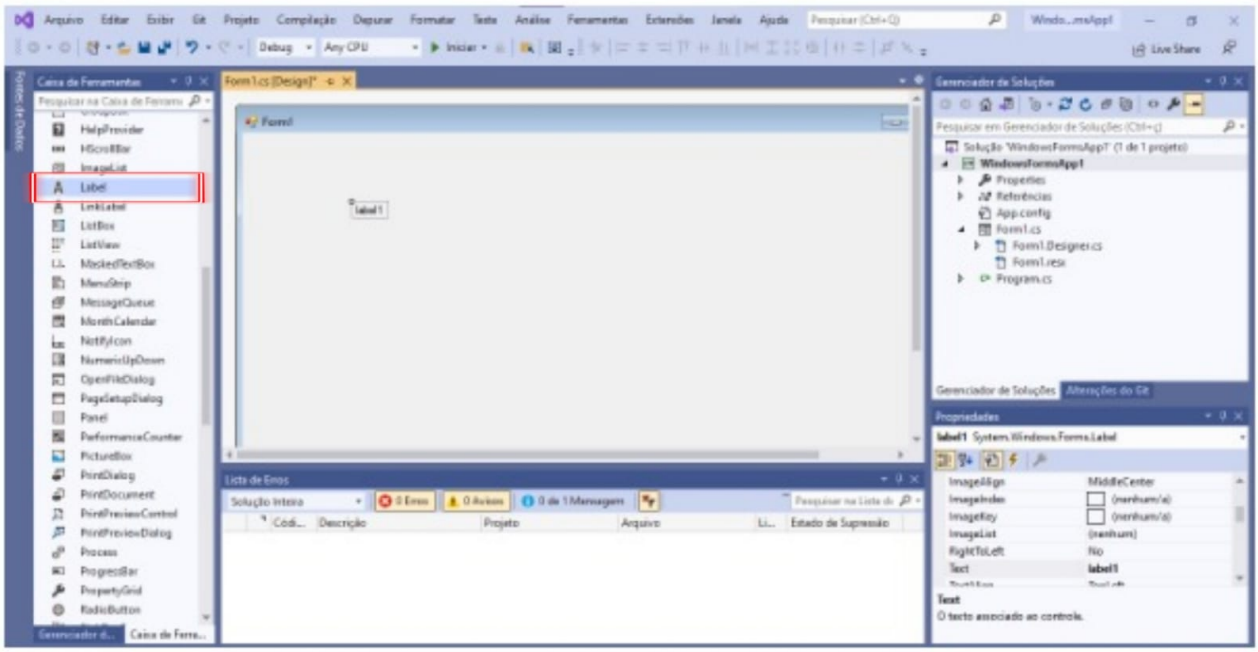
É recomendável que em botões se inscreva **btn** antes do nome da escolhido por exemplo **btnComecar:**



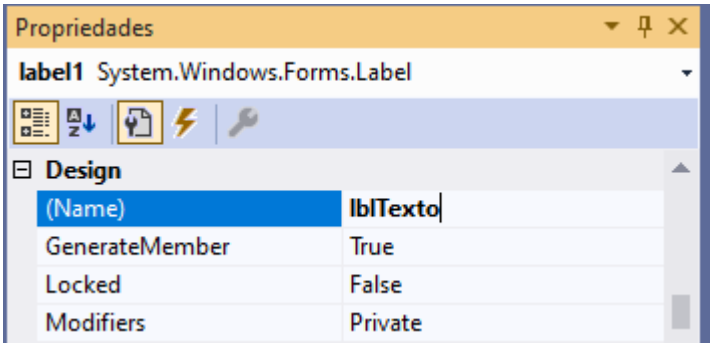
Isso é preferível pois facilitará em indicar que essa variável é um botão.

7.3.2. Adicionado label

Encontre a **Label** na aba **Caixa de Ferramentas** e arraste-o até o formulário:



Tudo feito no **Button** também é aplicável a **Label**. Ressaltando que diferente do botão o padrão de nomenclatura para a variável da label é lbl e o nome escolhido. Por exemplo **lblTexto:**



7.3.3. Adicionando código

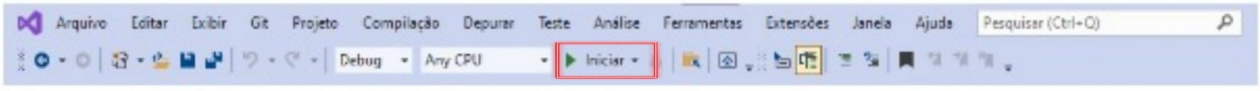
Para exibir a linha de código clique duas vezes no botão. Quando o **Forms.cs** abrir escreva o código depois da linha **private void** o comando almejado.

```
11  namespace WindowsFormsApp1
12  {
        3 referências
13      public partial class btnIniciar : Form
14      {
            1 referência
15          public btnIniciar()
16          {
17              InitializeComponent();
18          }
19
            1 referência
20          private void btnComecar_Click(object sender, EventArgs e)
21          {
22
23          }
24      }
25  }
26
```

Um código simples para realizar o teste pode ser:

```
11  namespace WindowsFormsApp1
12  {
        3 referências
13      public partial class btnIniciar : Form
14      {
            1 referência
15          public btnIniciar()
16          {
17              InitializeComponent();
18          }
19
            1 referência
20          private void btnComecar_Click(object sender, EventArgs e)
21          {
22              lblTexto.Text = "Olá Mundo!";
23          }
24      }
25  }
26
```

Agora é só iniciar a compilação clicando em **Iniciar**:

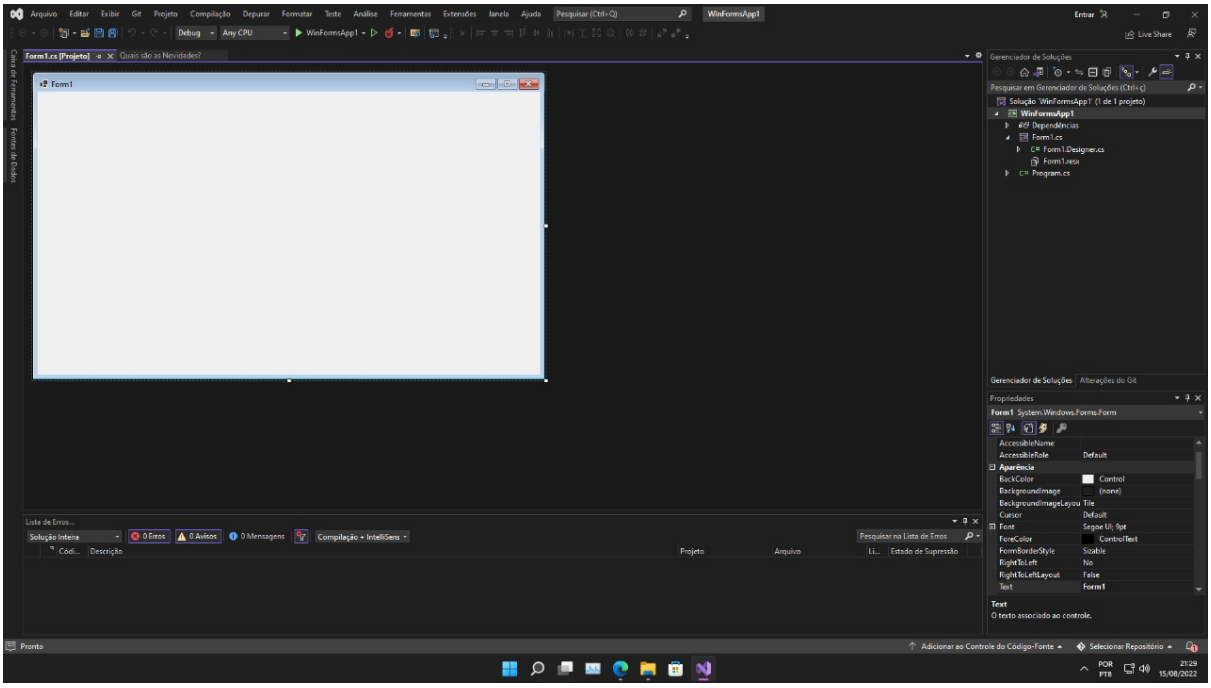


Quando a caixa de diálogo **Form1** abrir clique em **Começar**. Feito isso você verá que o texto da label mudará.

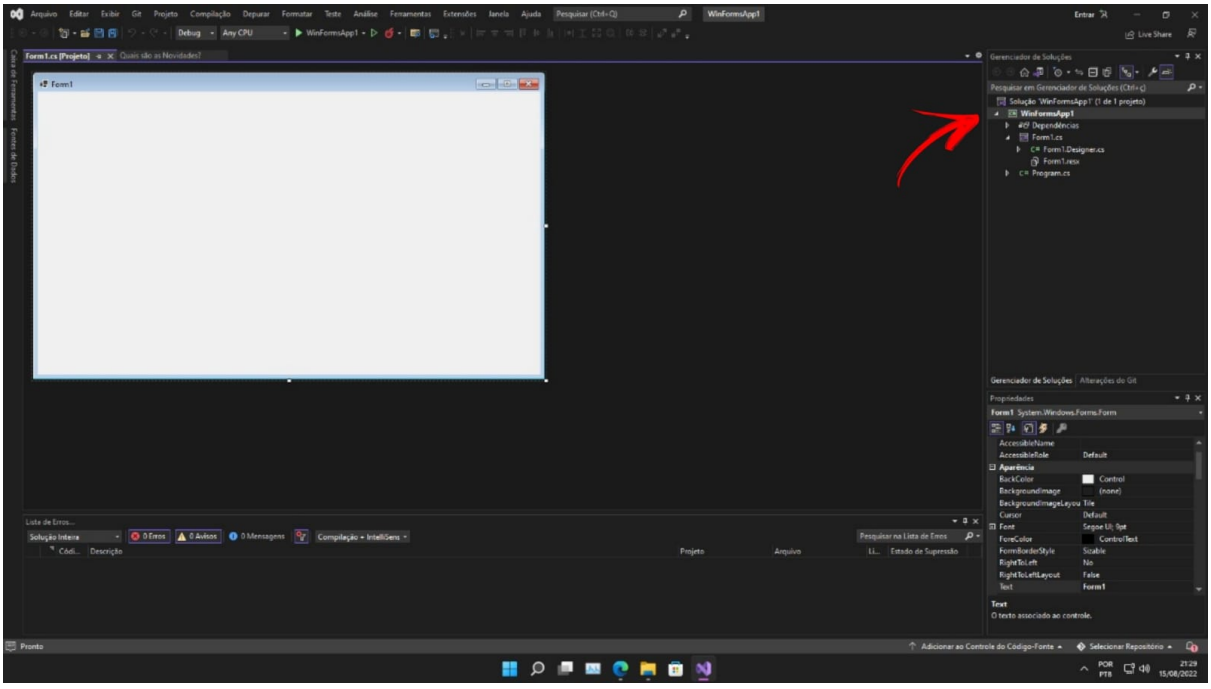
Para encerrar é só fechar o **Form1.**

7.4. COMO ADICIONAR BIBLIOTECAS

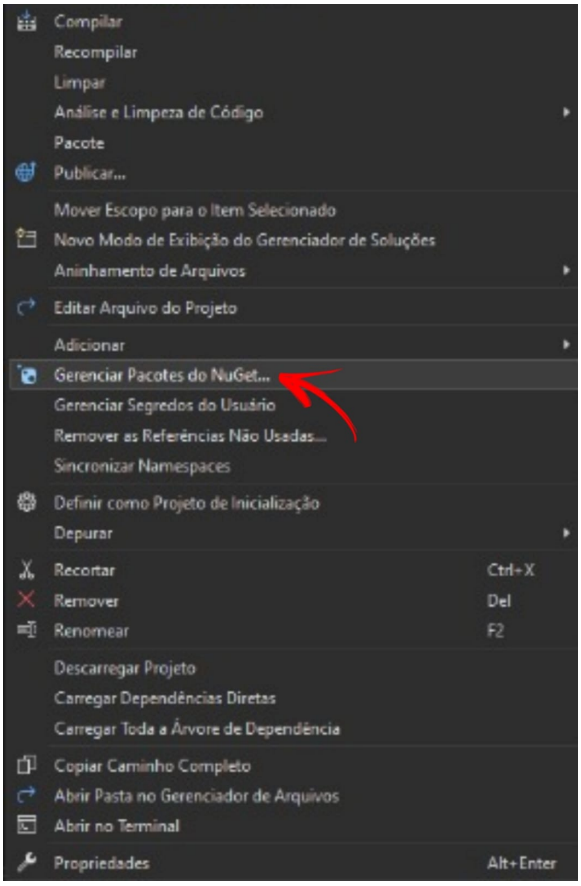
Após a criação de um projeto Windows Forms (C#) no programa Visual Studio, você verá isso em sua tela:



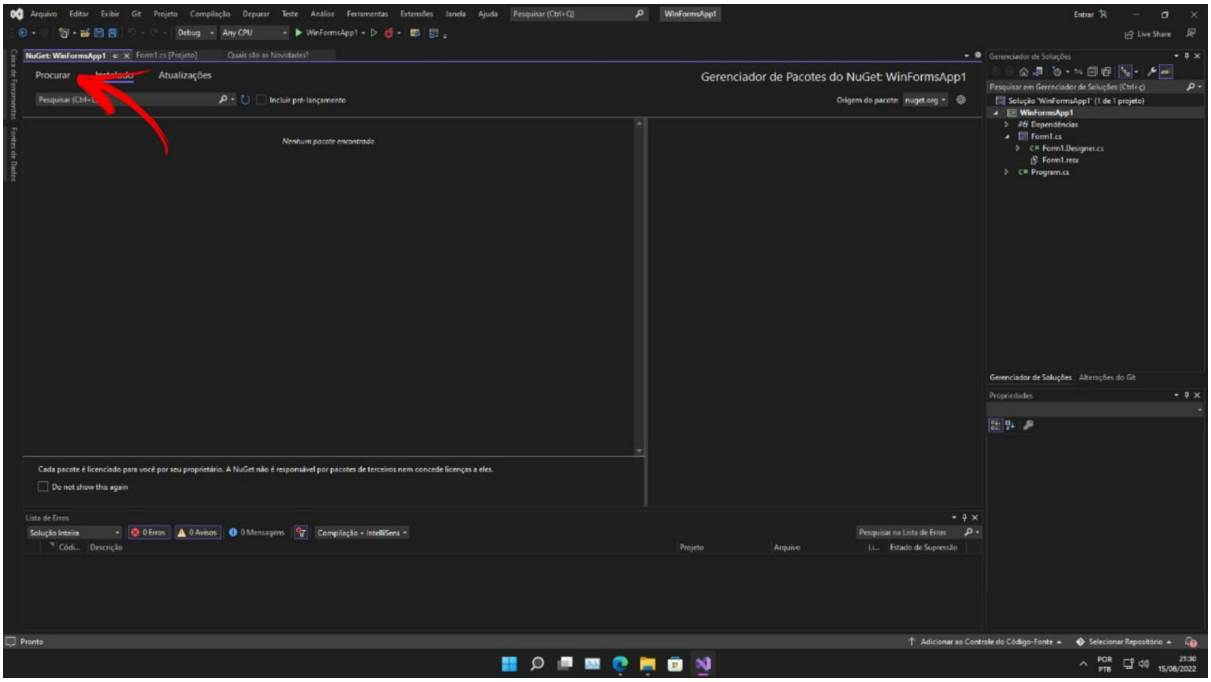
Clique com o botão direito no nome do projeto, que está localizado no lado superior direito, como indica a imagem:



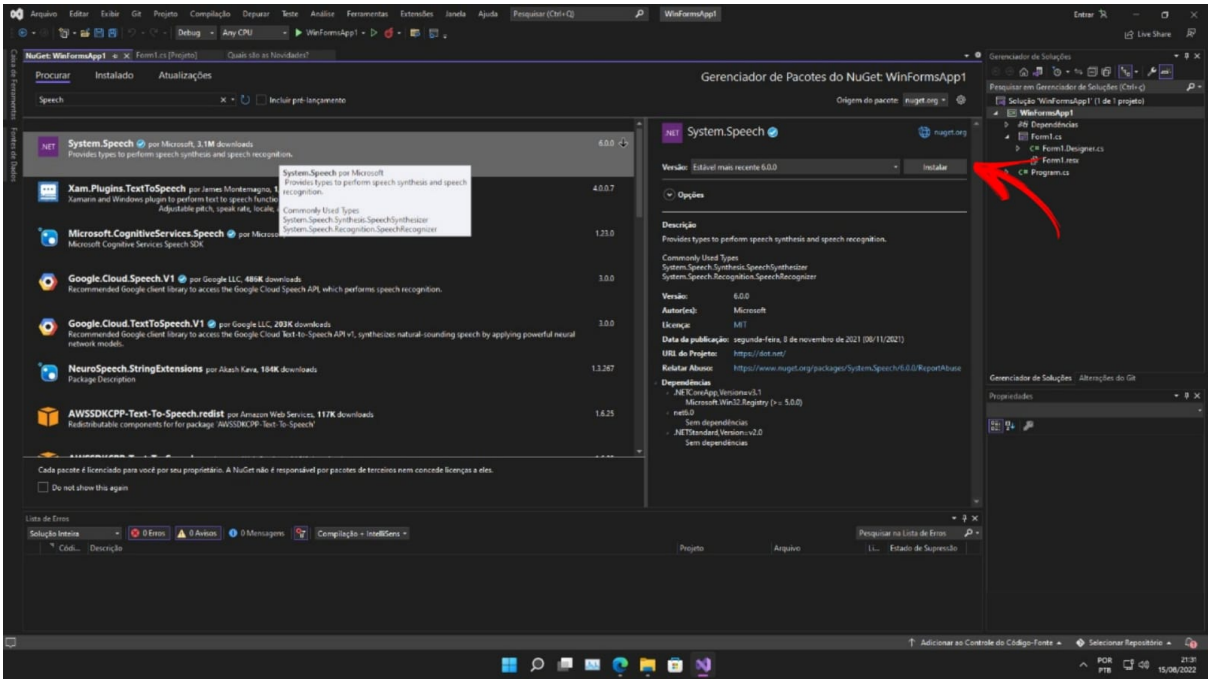
Após isso, vá em gerenciar pacotes do Nuget:



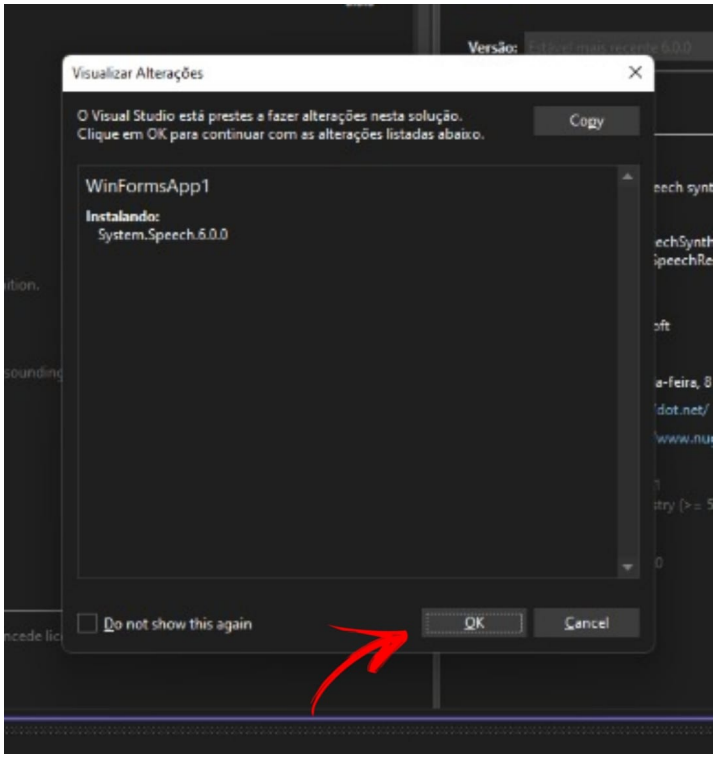
Quando abrir essa aba, vá em “Procurar”:



Pesquise “Speech” e selecione a primeira opção. Depois, clique em “Instalar”:



Clique em “OK” para começar a instalação da biblioteca desejada. Nesse caso, a “System.Speech”:



7.5. COMO USAR A BIBLIOTECA INSTALADA

Para começar a utilizar a biblioteca na aplicação, deve-se colocar essas linhas de System.Speech:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Speech.Recognition;
using System.Speech.Synthesis;
using System.Diagnostics;
```

No nosso programa, precisamos utilizar o Speech.Recognition, que é uma biblioteca capaz de reconhecer a fala, e o Speech.Synthesis, que permite configurar um mecanismo de síntese de fala, conversas, etc.

- Aqui são alguns exemplos de criação das gramáticas:

```
SpeechRecognizer sr = new SpeechRecognizer();

GrammarBuilder Word = new GrammarBuilder("Program 1");
GrammarBuilder Excel = new GrammarBuilder("Program 2");
GrammarBuilder Word = new GrammarBuilder("Program 1");
GrammarBuilder ArquivoUm = new GrammarBuilder("Number 1");
GrammarBuilder ArquivoDois = new GrammarBuilder("Number 2");
GrammarBuilder ArquivoTres = new GrammarBuilder("Number 3");
Grammar gWord = new Grammar(Word);
Grammar gExcel = new Grammar(Excel);
Grammar gArquivoUm = new Grammar(ArquivoUm);
Grammar gArquivoDois = new Grammar(ArquiviDois);
Grammar gArquivoTres = new Grammar(ArquivoTres);
sr LoadGrammar(gWord);
sr LoadGrammar(gExcel);
sr LoadGrammar(gArquivoUm);
```

```
sr LoadGrammar(gArquivoDois);
sr LoadGrammar(gArquivoTres);
```

Nessa parte do código, criamos um novo reconhecedor de discurso (SpeechRecognizer). O GrammarBuilder serve como uma gramática, e nele são adicionadas palavras que o reconhecedor deverá reconhecer e realizar alguma função. Ele fornece um mecanismo para criar programaticamente as restrições para uma gramática de reconhecimento de fala. Iniciamos consultores de gramática nomeados de “Program 1” e “Program 2”, o primeiro para Word e o segundo para Excel. Fizemos o mesmo com “ArquivoUm”, “ArquivoDois” e “ArquivoTres”, renomeando-os para serem utilizados em comando de voz posteriormente.

Uma gramática de reconhecimento de fala é um conjunto de regras ou restrições que definem o que um mecanismo de fala pode reconhecer como entrada significativa. As gramáticas antes construídas são definidas para serem carregadas no reconhecedor de fala.

7.5.1. Exemplo de utilização e reconhecimento das gramáticas para executar alguma ação (abrindo word e excel):

```
void sr_SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
{
    if (e.Result.Text == "Program 1")
    {
        SpeechSynthesizer synthesizerWord = new SpeechSynthesizer();
        synthesizerWord.Speak("Ok! Opening word");

        Process.Start("winword");
    }
    else
    if (e.Result.Text == "Program 2")
    {
        SpeechSynthesizer synthesizerWord = new SpeechSynthesizer();
        synthesizerWord.Speak("Ok! Opening excel");

        Process.Start(@"C:\Program Files\Microsoft
        Office\root\Office16\EXCEL.EXE");
    }
```

Implementamos todos os elementos do consultor de gramática em um novo comando de voz chamado SpeechRecognized. Usamos o void porque não sabemos o tipo referente que o usuário pode declarar.

Foram criados *if/else* para executar tudo isso que foi explicado até o momento. Quando o

usuário falar “Program 1” o Word será iniciado. Quando falar “Program 2”, o Excel será iniciado. Se o usuário falar “Number 1”, “Number 2” ou “Number 3”, serão abertos os arquivos indicados com os respectivos nomes.

7.5.2. Transformando o conteúdo digitado ou falado (presente na caixa de texto) em gramática e pedindo para o programa falar:

```
private void button2_Click(object sender, EventArgs e)
{
    string grava = displayText.Text;
    SpeechSynthesizer synthesizerGravar = new SpeechSynthesizer();
    synthesizerGravar.Speak(grava);
}
```

Para o código funcionar corretamente, utilizamos o SpeechSynthesizer, que é uma ferramenta que irá reconhecer o que foi dito e verificar se a palavra realmente existe na gramática criada e, se existir, vai acionar uma função.

7.6. RECONHECIMENTO DE FALA EM PORTUGUÊS

Para utilizar o reconhecimento de voz em português, você teoricamente teria que fazer uso do método:

```
private void Form1_Load(object sender, EventArgs e)
{
    SpeechRecognizer sr = new SpeechRecognizer(new
    CultureInfo("pt-BR"));
```

Porém, o reconhecimento de fala do Windows está disponível apenas nos idiomas inglês (Estados Unidos e Reino Unido), mandarim (chinês simplificado e tradicional) e espanhol. O seu código com certeza apresentaria erros em certas sintaxes. Entretanto, é possível resolver esse problema de outra maneira: primeiro, você deverá fazer o *download* dos seguintes arquivos disponibilizados no *Google Drive*:

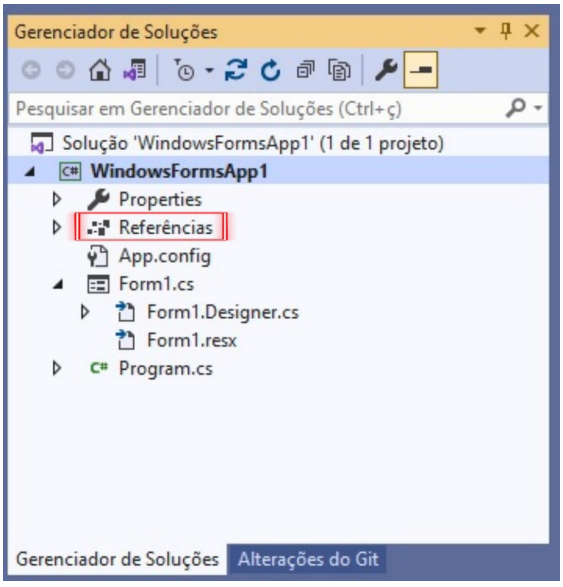
https://drive.google.com/drive/folders/11usj9tL6GEsrVTHsq52wihE0voYVKoD2?usp=sharing

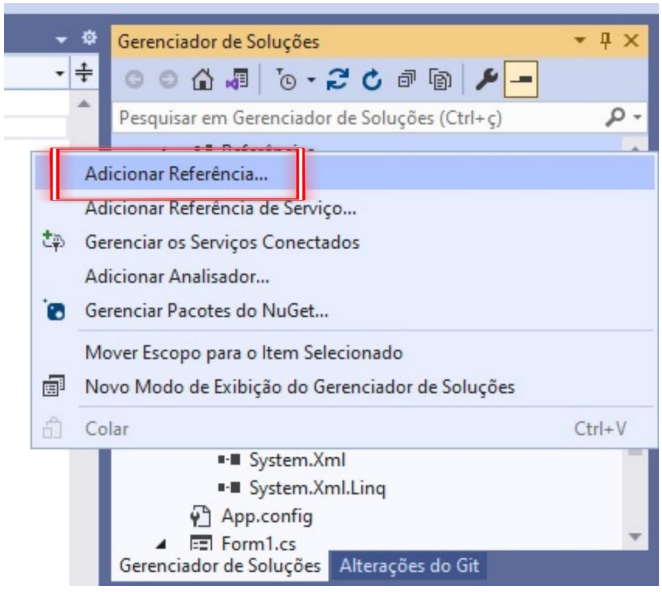
Esses arquivos devem ser instalados na seguinte ordem:

1. MicrosoftSpeechPlataformSDK.msi
2. SpeechPlatformRuntime.msi

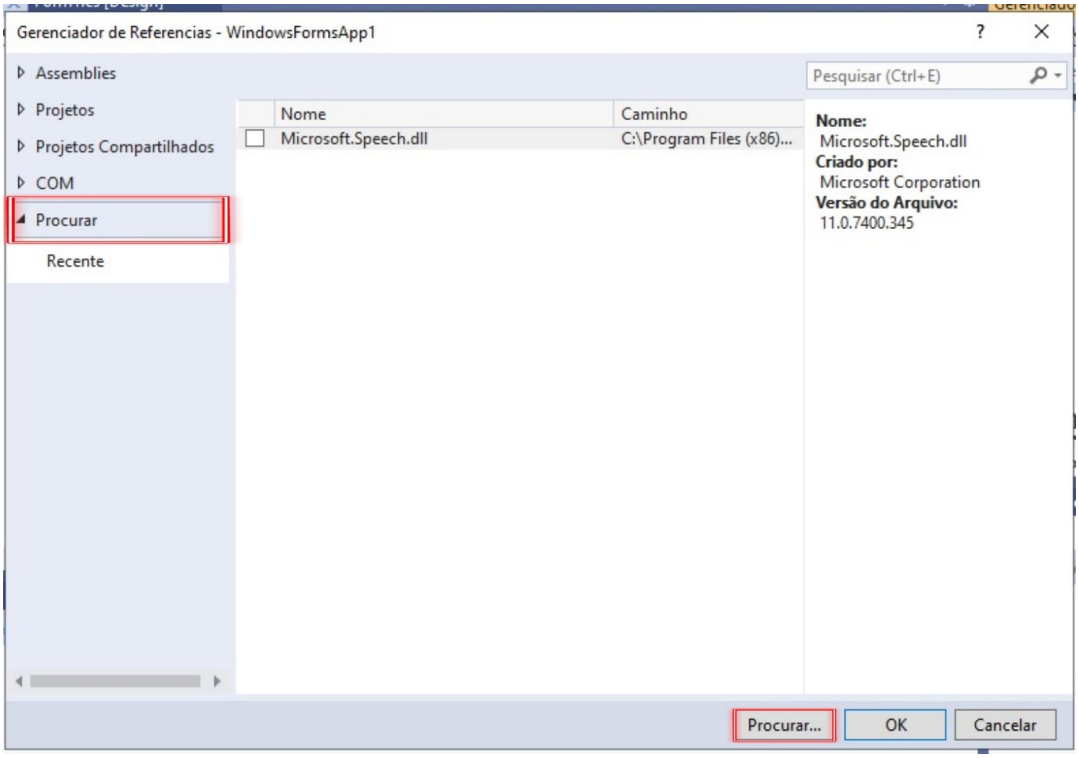
3. MSSpeech_SR_pt-BR_TELE.msi
4. MSSpeech_TTS_pt-BR_Heloisa.msi

Após a instalação, você deverá criar um Projeto Windows Forms no Visual Studio, como citado no item 7.2. Com o projeto aberto, clique com o botão direito em “Referências” e vá em “Adicionar Referência”



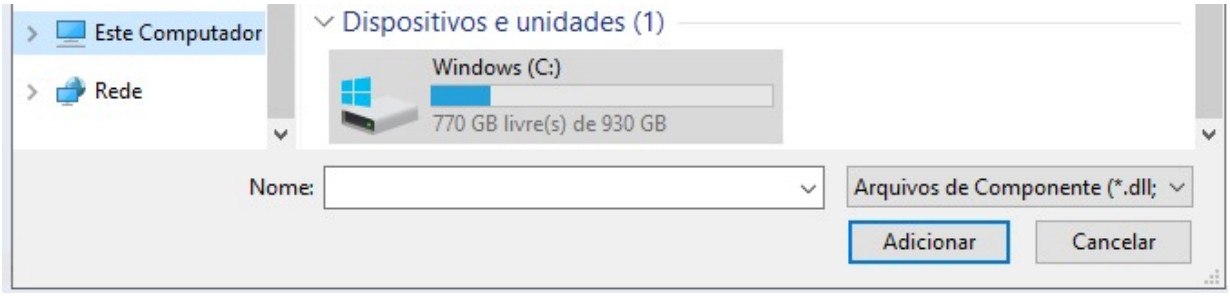


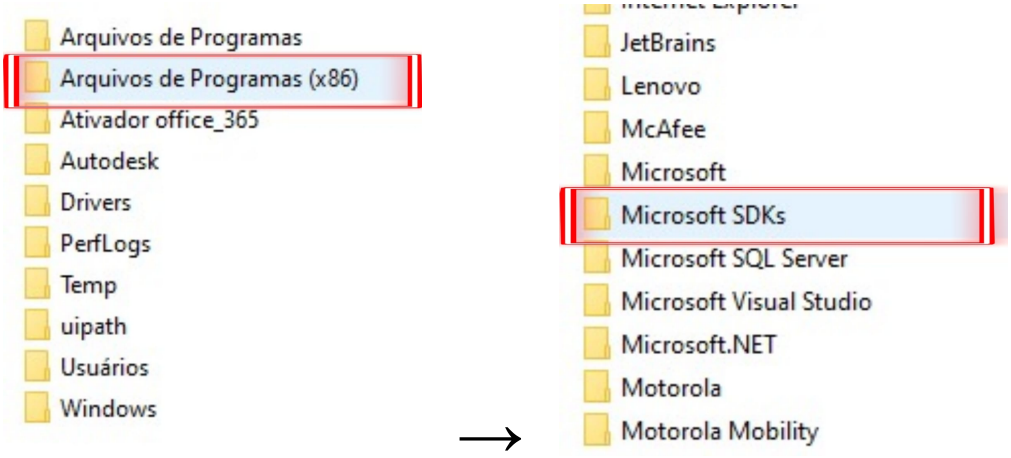
Após isso vá na parte de procurar referências, logo abaixo da opção *“COM”*, seguido da opção *“Procura”*. Aqui encontraremos as pastas em que se localizam as DLLs instaladas.

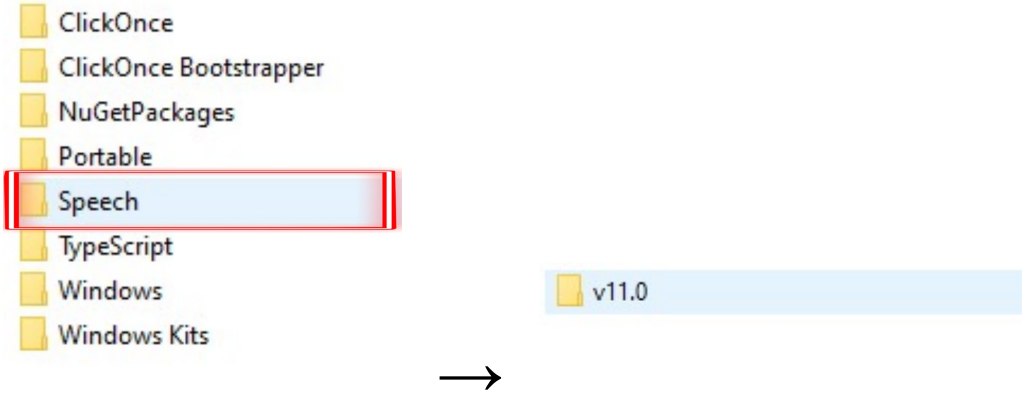


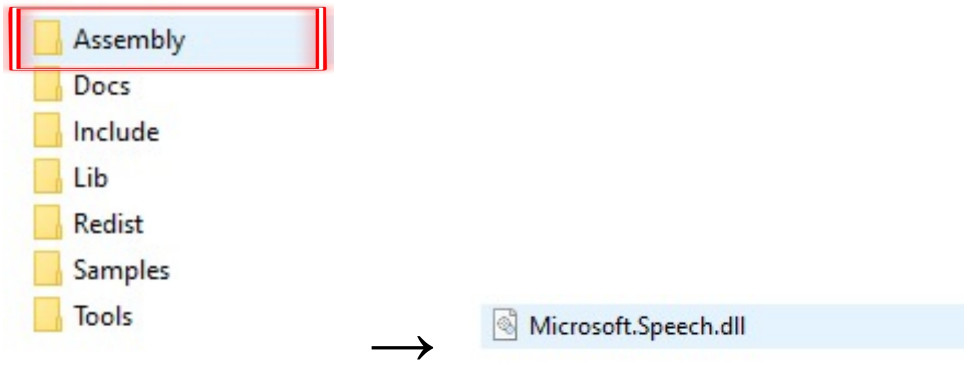
Para referenciarmos a DLL do Microsoft.Speech, precisamos seguir esse caminho:

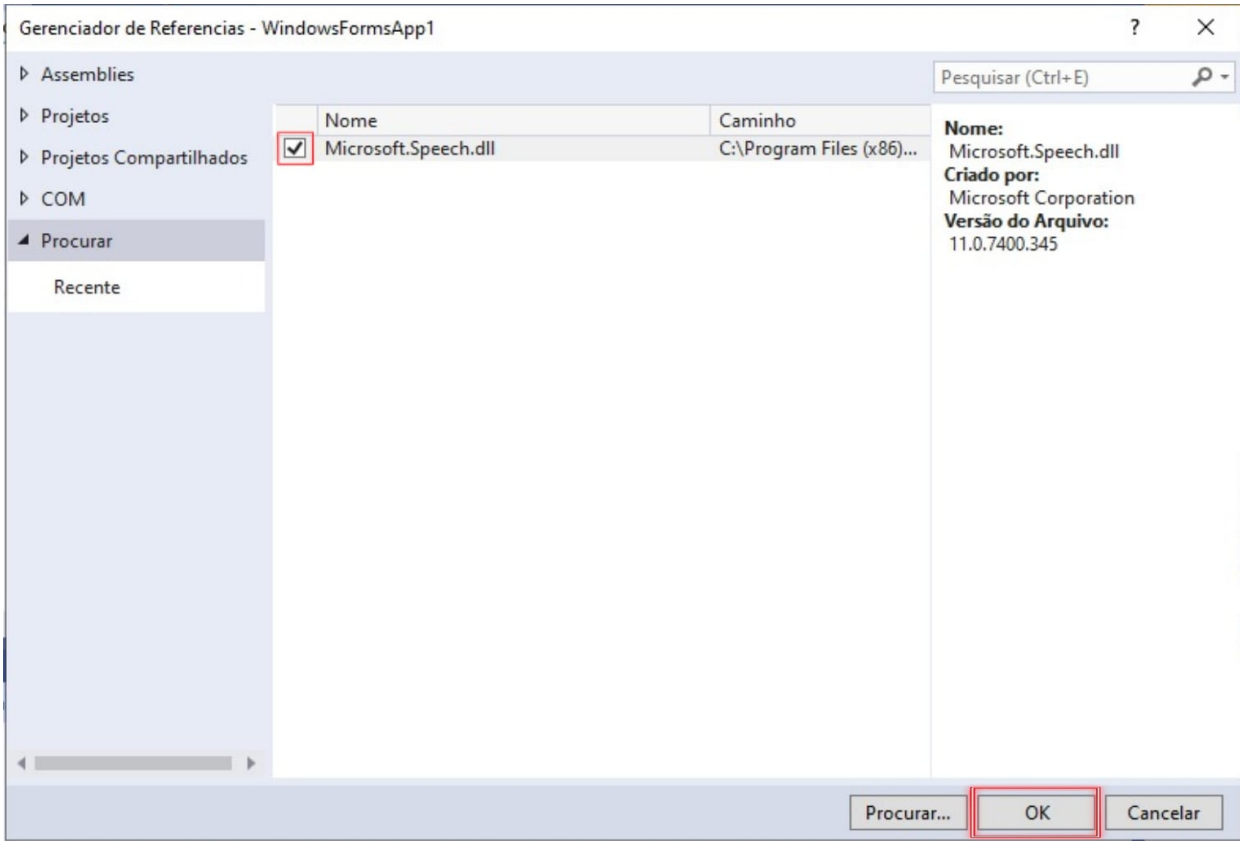
C:\Program Files (x86)\Microsoft SDKs\Speech\v11.0\Assembly\Microsoft.Speech.dll



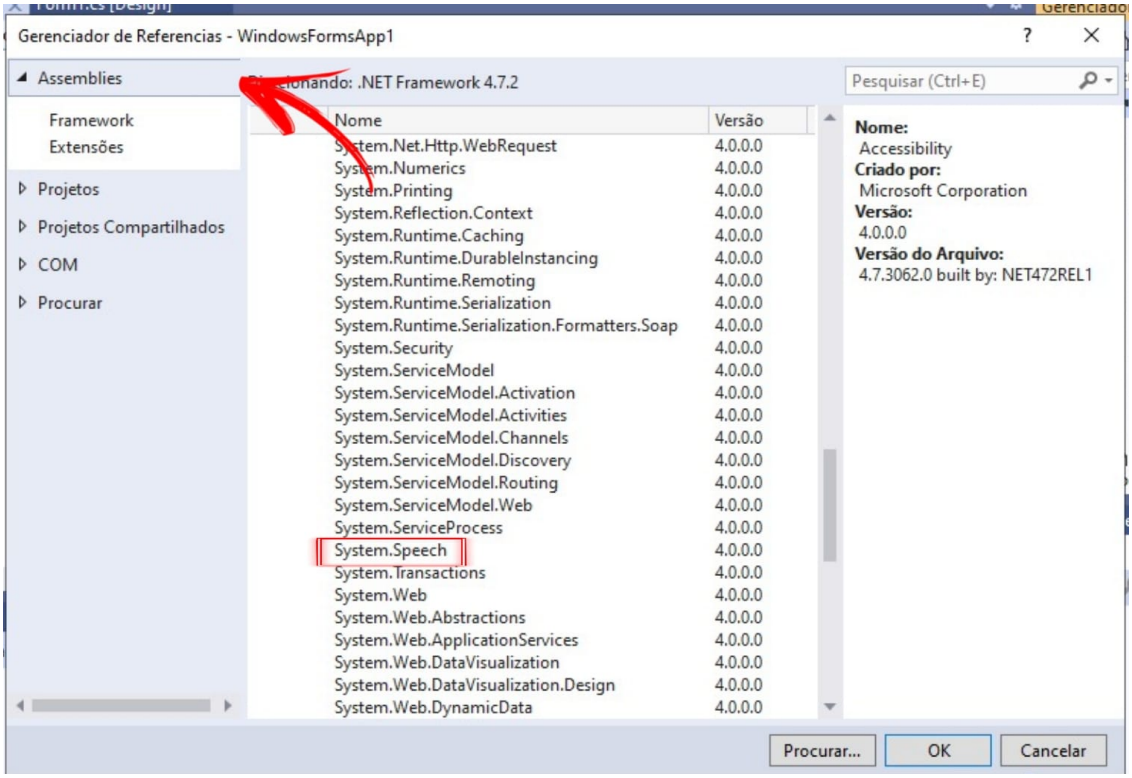


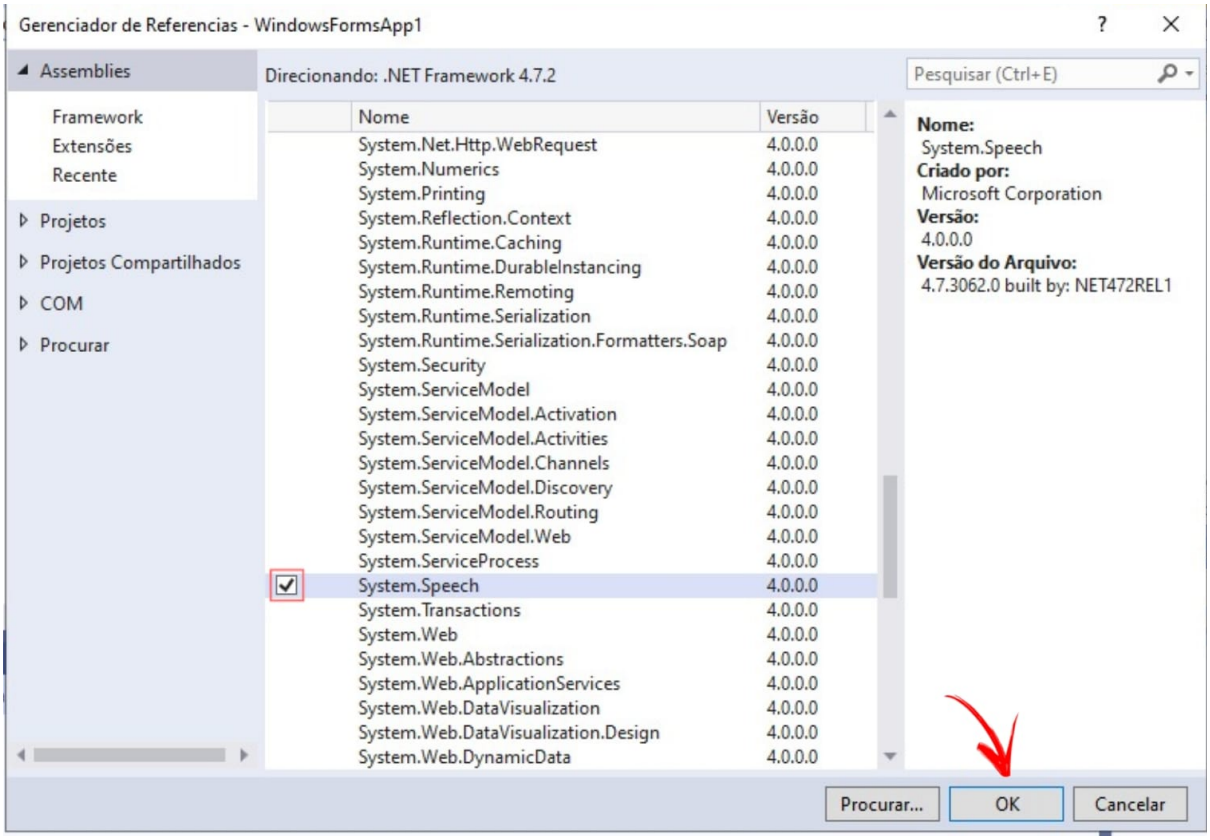






Após isso, é preciso adicionar a sintetização de voz (System.Speech), que pode ser adicionada diretamente pelo gerenciador de referências em Assemblies.





Para que tudo isso funcione deveremos inserir e referenciar todas as bibliotecas que serão utilizadas para o reconhecimento de voz, assim como mostrado na imagem abaixo.

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10 using Microsoft.Speech.Recognition;
11 using System.Speech.Synthesis;
12 using System.Globalization;
```

Agora, é hora de colocar em prática tudo o que foi falado até agora. Primeiro iremos construir nosso reconhecedor de voz, atribuindo ele a uma variável. E também faremos a criação de um array das palavras que serão reconhecidas pela máquina.

```
14 namespace WindowsFormsApp2
15 {
16     public partial class Form1 : Form
17     {
18         static CultureInfo ci = new CultureInfo("pt-BR");
19         static SpeechRecognitionEngine reconhecedor;
20         SpeechSynthesizer resposta = new SpeechSynthesizer();
21 
22         public string[] vocabulario = {"oi", "quem é você"};
24         public Form1()
25         {
26             InitializeComponent();
27             Init();
28         }
```

Dentro do bloco *Gramatica*, seguimos criando um tratamento para erros que irá testar se as bibliotecas e o reconhecedor foram instalados, inseridos e criados corretamente. Caso esse não seja o caso, o programa apresenta erro e não prosseguirá.

```
30   public void Gramatica()
31   {
32       try
33       {
34           reconhecedor = new SpeechRecognitionEngine(ci);
```

```
35          }
36          catch(Exception ex)
37          {
38              MessageBox.Show("ERRO ao integrar a língua escolhida!");
39          }
```

A seguir há a construção da gramática da máquina onde os valores do array *vocabulario* serão atribuídos a variável *gramatica.*

```
41              var gramatica = new Choices();
42              gramatica.Add(vocabulario);
43
44              var gb = new GrammarBuilder();
45              gb.Append(gramatica);
```

Ainda no mesmo bloco *Gramatica* na linha 53 a 58 o reconhecedor é utilizado para reconhecer e armazenar a palavra dita pelo usuário em uma variável para, assim, ser usada posteriormente. Após isso, é feito um tratamento de erro caso aconteça algum problema ao reconhecer ou armazenar a palavra dita.

```
47      try
48      {
49          var g = new Grammar(gb);
50
51          try
52          {
53              reconhecedor.RequestRecognizerUpdate();
54              reconhecedor.LoadGrammarAsync(g);
55              reconhecedor.SpeechRecognized += Sre_Reconhecimento;
56              reconhecedor.SetInputToDefaultAudioDevice();
57              resposta.SetOutputToDefaultAudioDevice();
58              reconhecedor.RecognizeAsync(RecognizeMode.Multiple);
59          }
60          catch(Exception ex)
61          {
62              MessageBox.Show("ERRO ao criar reconhecedor!");
63          }
64      }
65      catch(Exception ex)
66      {
67          MessageBox.Show("ERRO ao criar gramatica!");
```

```
68      }
69  }
```

Em seguida, é feito um bloco para definir o volume e a velocidade em que o programa irá falar.

```
71          public void Init()
72          {
73              resposta.Volume = 100;
74              resposta.Rate = 3;
75
76              Gramatica();
77          }
```

Por fim será feita a comparação entre a palavra dita com a gramática, desta forma dando um resultado para determinadas combinações. No nosso exemplo temos os seguintes casos: Quando o usuário disser “Oi”, essa palavra será comparada com a gramática já existente, e assim, responderá com “Olá”. Porém se a palavra dita não existir na gramática, o programa irá travar nos tratamentos de erro das etapas acima.

```
79 void Sre_Reconhecimento(object sender, SpeechRecognizedEventArgs e)
80 {
81     string frase = e.Result.Text;
82
83     if(frase.Equals("oi"))
84     {
85         resposta.SpeakAsync("olá");
86     }
87
88     else if(frase.Equals("quem é você"))
89     {
90         resposta.SpeakAsync("Eu sou um software de interação");
91     }
92 }
```

8. PERCEPÇÕES FUTURAS

Levando em conta a simplicidade do nosso projeto, sabemos que ele pode ser reutilizado e manuseado de formas diferentes. É possível acrescentar muitas coisas ainda, de forma que ele se torne um guia muito completo e interessante. A princípio, nosso projeto visa que os estudantes o vejam como uma forma fácil de aprender um pouco sobre assistentes de voz e bibliotecas, além da aplicação de classes e do famoso Windows Forms.

É importante acrescentar que o nosso projeto não tem uma conclusão definida, ele sempre poderá ter continuação. Pode-se adicionar novas bibliotecas, novos designs abusando do Windows Forms, fazer com que a máquina aprenda novas palavras e muito mais. Sinta-se livre para explorar mais o código e adicionar os seus próprios aprendizados e conhecimentos.

Por exemplo, você pode criar uma assistente virtual do estilo da Alexa, da Siri, e/ou Google Assistente. Você também pode construir um assistente que leia tudo que for exibido na tela, ou seja, um assistente de leitura. Mas essas são apenas algumas possibilidades, tudo depende da sua criatividade e capacidade.

REFERÊNCIAS BIBLIOGRÁFICAS

JONES, DUSTIN. SCHARFENBERG, BRIAN. PERKINS, JESSICA. CHILDERS, KERI. DOGBEY, GODWIN. SHUBROOK, JAY. **Glycated Hemoglobin Testing to Identify Undiagnosed Diabetes Mellitus in the Inpatient Setting**. Disponível em: https://www.degruyter.com/document/doi/10.7556/jaoa.2016.075/html**.** Acesso em: 20 de abr.

2022.

SAFRA, GILBERTO. **Internação por diabetes mellitus no Brasil entre 2016 e 2020**. Disponível em: https://d1wqtxts1xzle7.cloudfront.net/71039303/pdf-with-cover-page-v2.pdf?Expires=1650947113&Signature=EQ11y66rzY3LP1KEfhNvfbS9fBmPA~AR0AxqfyPeHGN1seuIKr0WJwsZEHY0BY2Y5WrdPms1Zt4Rfjo3b~qAt7VyHCn4V7Rs3X~FbxwhwVsYSMF5p6pgpK1LtoDzlJCO51ZKUwvnbw0OpEel7zTr-kXbxvOmwlijWQ04uPQuqX5yGj87WypY~boO7X91GNeKtiO4pTzturhEPVxED6Tg0~tVYaY8E33qVUVJ-GrEFTa-w1NncbnwJT42Ej2nF2UJIITgFb~k92NyPkB0QhWFWp~1vQ2rQ8LboCRcYg3NzwfljVsiXcPIu~~raIRhg2-CKJeDN1Ah-v8IWSRRTqXFcg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA. Acesso em: 10 abr. 2022.

FLOR, LUISA. CAMPOS, MONICA. **Prevalência de diabetes mellitus e fatores associados na população adulta brasileira: evidências de um inquérito de base populacional.** Disponível em: https://www.scielosp.org/article/rbepid/2017.v20n1/16-29/pt/. Acesso em: 12 abr. 2022.

BARBOSA, SILVÂNIA. CAMBOIM, FRANCISCA **. Diabetes mellitus: cuidados de enfermagem para controle e prevenção de complicações.** Disponível em: https://temasemsaude.com/wp-content/uploads/2016/09/16324.pdf. Acesso em: 12 abr. 2022

JONCO, CAMILA. SILVEIRA, STEFANIE. **Hey Siri: inteligência artificial e a humanização dos assistentes pessoais. Disponível em:** http://www.repositorio.jesuita.org.br/bitstream/handle/UNISINOS/6407/Camila%20Medeiros%20Jonco_.pdf?sequence=1**.** Acesso em: 15 abr. 2022

ANHAIA, INGREDY. MUZEKA, IGOR. VELHO, IGOR. ASSISTENTE VIRTUAL "E AÍ LUNA" AVEL. Disponível em:https://semanaacademica.org.br/system/files/artigos/artigo_-_ingredy_anhaia_v2_1.pdf**.** Acesso em: 15 abr. 2022

TURING, ALAN. **Computing Machinery and Intelligence.** Disponível em: https://www.csee.umbc.edu/courses/471/papers/turing.pdf. Acesso em: 15 abr. 2022

MAIOLINO, THAÍS. OLIVEIRA, EDUARDO**. Maximus: Automatizando Tarefas por Voz.** Disponível em: https://app.uff.br/riuff/bitstream/handle/1/5540/Maximus-%20automatizando%20tarefas%20por%20voz.pdf?sequence=1&isAllowed=y. Acesso em: 10 abr. 2022

SCAICO, Pasqueline Dantas. **Um Estudo sobre o Desenvolvimento de Interesse pela Aprendizagem de Programação**. 2018. 218 f. Tese (Doutorado) - Curso de Ciência da Computação, Centro de Informática, Universidade Federal de Pernambuco, Recife, 2018. Disponível em: https://repositorio.ufpe.br/bitstream/123456789/30620/1/TESE%20Pasqueline%20Dantas%20Scaico.pdf. Acesso em: 30 ago. 2022.

MICROSOFT. **Create a Windows Forms app in Visual Studio with C#**. Disponível em: https://docs.microsoft.com/pt-br/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2022. Acesso em: 01 set. 2022.

Reconhecimento de voz em C# usando Microsoft.Speech. YouTube, 2 jun. 2017. Disponível em: https://youtu.be/vUGDKknf0Pw. Acesso em: 04 out. 2022.