Especificação Básica da API para projeto Mega Time

Versão 1.0 – Para desenvolvimento da Etapa 1 do Projeto

1. Informações Gerais

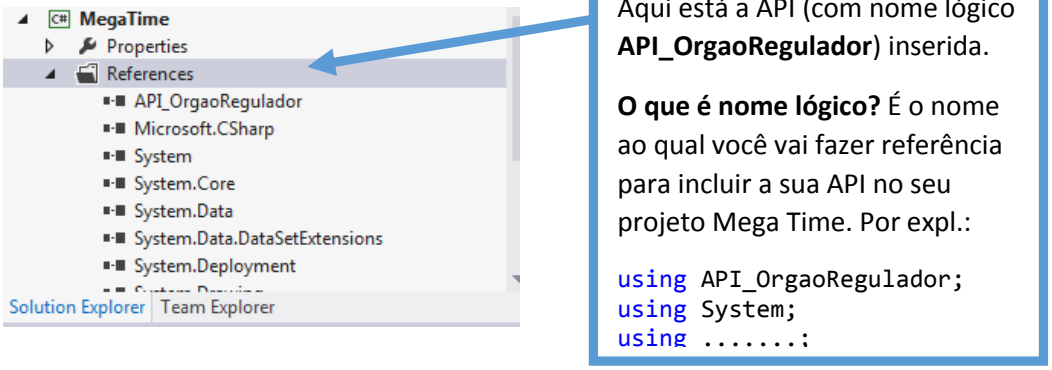
a. Uma API é uma sigla que significa *Application Program Intreface*, ou seja, uma interface de programação de aplicativos.

b. Conceitualmente, uma API é um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por outros aplicativos, os quais não pretendem envolver-se em detalhes da implementação da API, mas apenas usar seus serviços.

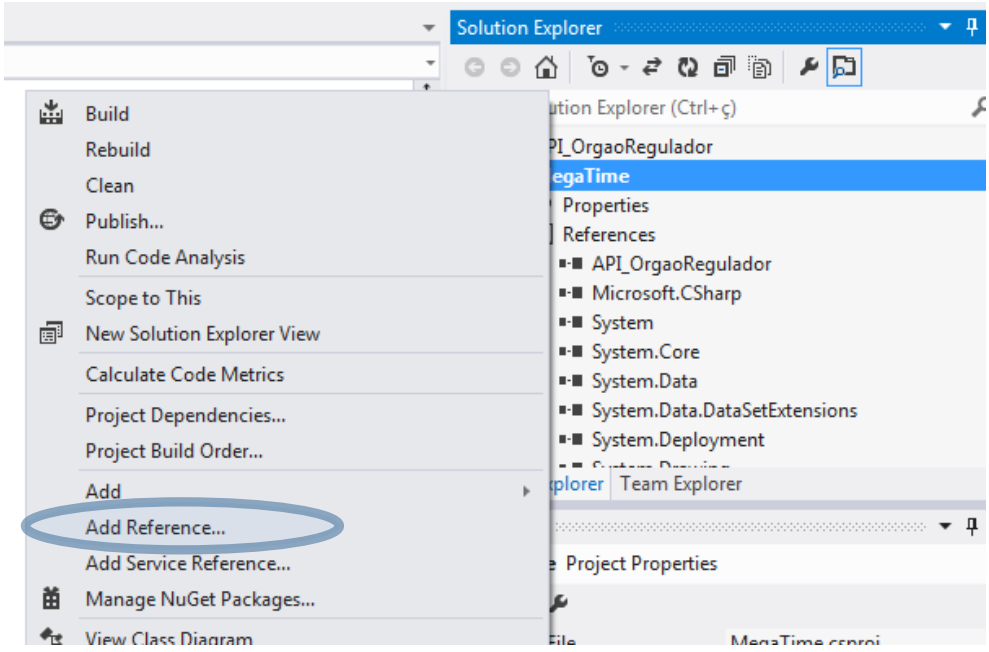
c. O arquivo onde está compilada a nossa API pronta para o projeto chama-se **API_OrgaoRegulador.dll** . Baixe este arquivo e coloque-o em uma pasta junto de seu projeto.

d. Este arquivo será disponibilizado aos aluno pelo professor de projeto, no AVA ou por outro meio.

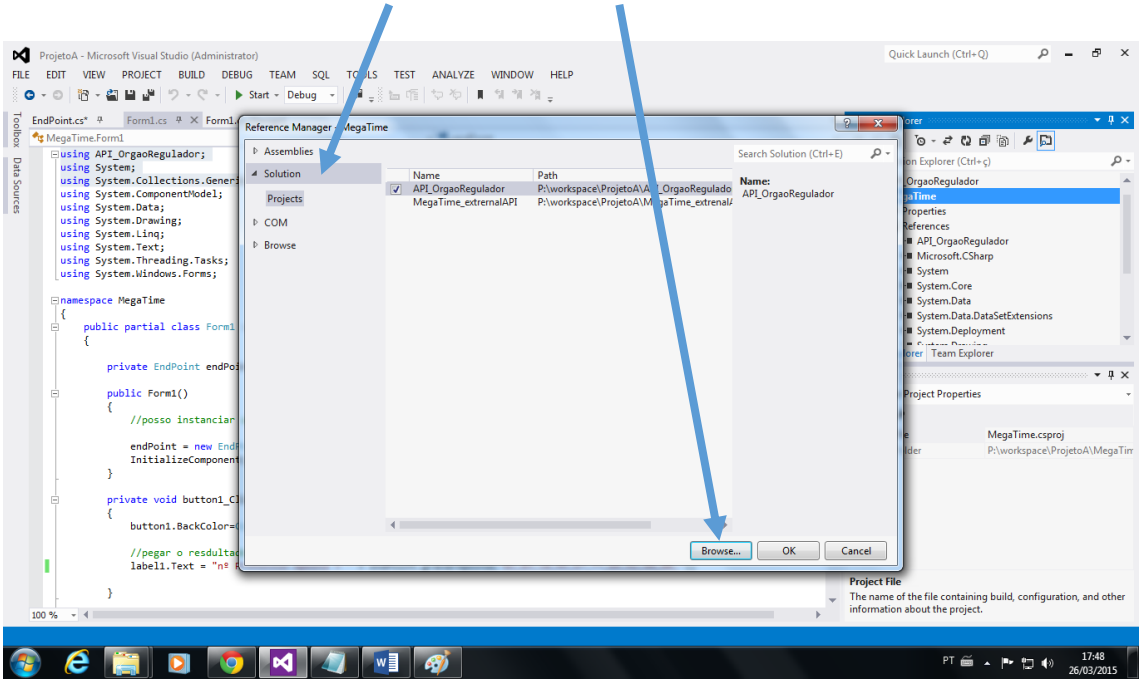
e. Esta API terá que ser referenciada na **Solution** de seu Projeto, da forma como segue:



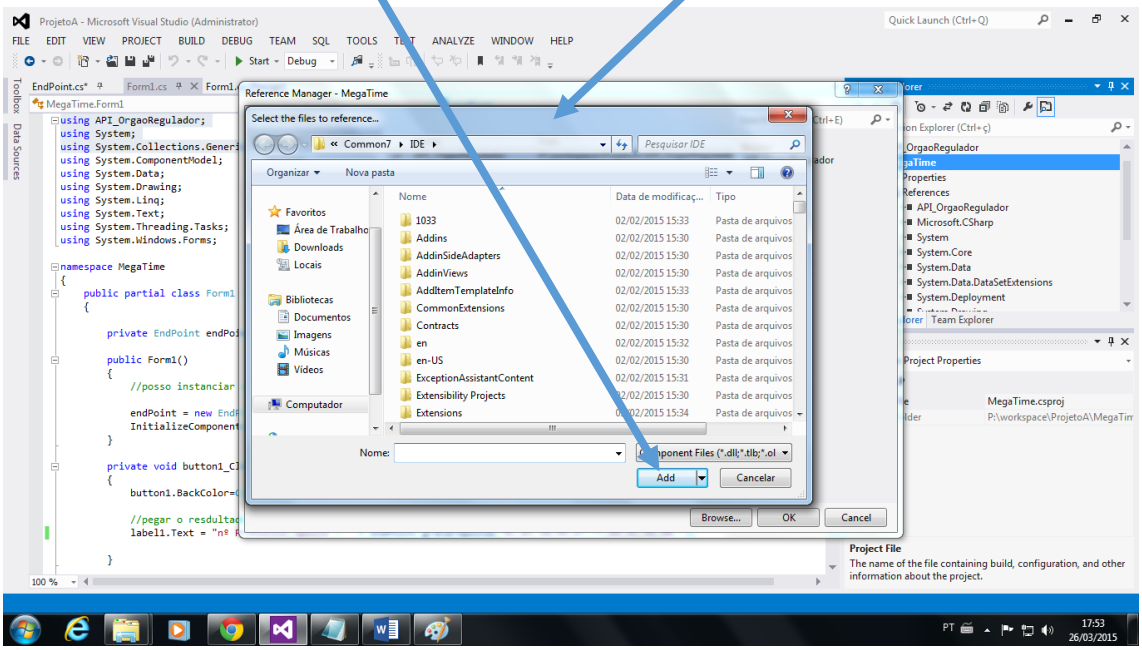
f. Para fazer a inserção clique com o botão de contexto (botão direito do mouse) no nome do seu projeto em sua Solution, e então selecione a opção Add reference.



g. Selecione a opção **Solution** e Clique em **Browse**



h. Então selecione o caminho onde está o arquivo físico **API_OrgaoRegulador.dll** e depois clique em **add**.



2. Usando a API no seu programa – Olhe sempre o exemplos de código dados

Dentro da API existe apenas uma classe programada, chamada `EndPoint`. Para quem não tem conhecimento aprofundado de orientação a objetos (OO), considere que uma classe é uma estrutura que agrupa funções prontas uso (uma biblioteca, porém mais sofisticada). Como curiosidade, na OO as funções são conhecidas como métodos.

Neste caso específico, a maneira que construímos esta classe `EndPoint` faz com que ela se comporte de modo similar a um tipo. A diferença é que, por ser uma classe, além de permitir manipulação de dados a classe lhe permitirá usar as funções prontas (métodos) que já encapsulamos em seu código fonte.

Assim como o tipo primitivo `int` exige que vc. declare quais variáveis vão guardar dados inteiros, a nossa Classe `EndPoint` vai exigir que você diga que objeto (digamos algo similar a uma variável) vai permitir que vc. acesse as funções de nossa classe.

No exemplo abaixo, isso acontece logo depois da declaração do seu form (1ª linha comentada – comentários estão em verde depois do //). Veja que no caso abaixo optamos (por ser um padrão) usar o nome do objeto com `endPoint` em minúsculo.

Para poder efetivamente usar a classe `EndPoint` de nossa API, é necessário mais um passo, que é instanciar o objeto `endPoint` . Isso está acontecendo na segunda linha comentada, dentro da função do seu form (`public Form1()`) que usará a API.

```
namespace MegaTime
{
    public partial class Form1 : Form
    {

        private EndPoint endPoint; //delclara a classe da API EndPoint para seu usada no seu
programa como endPoint (nome do objeto)

        public Form1()
        {
            //instanciando o objeto endPoint da classe EndPoint
            endPoint = new EndPoint();
         .

         .

         .
```

Este objeto `endPoint` (sempre em minúsculo) poderá a partir de agora ser usado em qualquer método que trata eventos no do seu form, para fazer referência e chamar as funções contidas na classe da API (não esquecer fazer referência ao uso da sua API no using .... – antes de começar as programação do form, junto com os demais using que já vieram prontos). A sintaxe para fazer a chamada será:

Nome_do_objeto.nome_da_função(parâmetros)

Exemplo:

```
private void button1_Click(object sender, EventArgs e)
{
    long resultado;  //declara a variável onde voltará o número do protocolo

    //chama a função gravaAposta, passando-lhe uma String com as dezenas
    //apostadas separada por virgula (entre " ") e pega como saída desta função um numero
    //inteiro longo que é o número do protocolo. Funções em APIs são como caixas-pretas
    resultado = endPoint.gravarAposta("01,03,10,45,67,77,89,92,99,00");

    //depois concatena (+) o texto entre "" abaixo com o resultado devolvido pela
    //função atribui essa string resultante para a propriedade Text do label1 do seu form
    label1.Text = "nº Protocolo Aposta = " + resultado;
}
```

3. Funcionamento da API como uma caixa-preta – As Funções prontas para você usar nesta etapa do projeto.

Toda função tem um nome, recebe zero, um ou vários parâmetros e pode fornecer um retorno para quem a chamou.

Os parâmetros devem respeitar a ordem dos tipos dos parâmetros, conforme o que foi definido por que programou a função.

Este conjunto formado pelo nome, os parâmetros e o retorno são conhecidos como assinatura, pois formam um identificador único para a função (método) que vc. quer chamar.

A sintaxe geral da assinatura das funções é a seguinte:

```
Tipo_de_retonro nome_da_função (parâmetro1, parâmetro 2, ...)
```

Função 1: `long gravarAposta(String dezenas) //esta função só recebe 1 parâmetro tipo String`

Para chamar essa função você tem que passar como informação (parâmetro) uma String contendo as dezenas que pretende apostar, separando-as por virgulas ou espaços ou ponto e vírgula.

Nesta 1ª versão, a gravação vai sempre ocorrer em um arquivo padrão denominado c:\temp\APOSTAS.TXT .

Se a gravação for bem sucedida você receberá como retorno um inteiro longo com o número do protocolo da aposta.

Também não esqueça que para chamar a função você tem que indicar o nome do objeto da classe. Olhe o exemplo das linhas abaixo, de novo.

```
        long resultado;
        resultado = endPoint.gravarAposta ("01,03,10,45,67,77,89,92,99,00");
nome da classe   .  nome da função   (um parâmetro)- string com 10 dezenas separadas por virgula
```

Na variável resultado você terá o número do protocolo.

A única verificação que a API faz é produzir ZERO (0) como resultado se o parâmetro passado tiver menos de 10 ou mais de 20 dezenas. Outras verificações não são realizadas na API (Ex: se há números repetidos na aposta). Por isso, lembre-se de que seu projeto deve verificar as regras do jogo no código do seu programa.

Função 2: `int obterQuantidadeDezenasApostadas(long prot)`

Para chamar essa função você tem que passar como informação (parâmetro) um número inteiro longo contendo o número do protocolo e receberá como resposta um valor inteiro (int) com a quantidade de dezenas que foram apostadas no jogo daquele número de protocolo.

Se o número passado como parâmetro não for localizado, esta função da API responderá 0 como resultado. Veja o Exemplo:

```
int na = endPoint.obterQuantidadeDezenasApostadas(2118188);
```

Se o número de protocolo 2118188 existir, será atribuído para a variável **na** o número de dezenas apostadas. Senão será atribuído valor 0 (zero).

Função 3: `String obterTodasDezenasApostadas(long prot)`

Para chamar essa função você tem que passar como informação (parâmetro) um número inteiro longo contendo o número do protocolo e receberá como resposta um String com todas as dezenas apostadas, em ordem crescente e separadas por vírgula na String.

Se o número passado como parâmetro não for localizado, a função da API responderá `null` como resultado. Veja no Exemplo como tratar um possível retorno `null` usando a instrução if.

```
private void button2_Click(object sender, EventArgs e)
 {
     String s = endPoint.obterTodasDezenasApostadas(2118188);
     if (s==null)
           textBox1.Text = "Protocolo de Aposta não localizado";
     else
          textBox1.Text = s;
 }
```

Se o número de protocolo 2118188 existir, será atribuído como retorno da função para a variável **s** um string, contendo todas as dezenas apostadas, separadas por virgula.

Em seguida, o programa verifica (usando o if) se a variável **s** está contendo `null`, atribuindo uma mensagem para a caixa texto se isso for verdade. Se for diferente de `null`, atribui o conteúdo que voltou a da API para a caixa de texto.

Função 4: `int obterDezenaDaAposta(long prot, int dz)`

Esta função recebe dois parâmetros. O primeiro é um número inteiro longo contendo o número do protocolo e o segundo qual a dezena deste protocolo deseja consultar. **O resultado será retornado diretamente como inteiro**. Por exemplo:

```
int x = endPoint.obterDezenaDaAposta(4881579, 10);
```

A função será chamada e o número do protocolo 4881579, solicitando que seja retorna a décima dezena que foi apostada neste jogo. O valor desta dezena será retornado já em inteiro e armazenado na variável x. As dezenas são sempre retornadas em ordem crescente de valor.

Note que há várias situações anômalas como resultado da execução desta função, sendo cada uma delas informada ao usuário através de um código de erro, com valor inteiro negativo:

a) **Situação Anômala 1:** se usuário passar como parâmetro um número de protocolo que não corresponde a um jogo apostado, a função retornará valor **-1** para a variável x.

b) **Situação Anômala 2:** Se o usuário passar como 2º parâmetro uma dezena que seja inferior ou igual a zero, a função retornará valor **-2** para a variável x. (Não há dezena 0 ou negativa em um jogo de loteria)

c) **Situação Anômala 3:** Se o usuário passar como parâmetro uma dezena que é maior que 20 ou maior que o número de dezenas do jogo correspondente ao protocolo passado no 1º parâmetro, a função retornará valor **-3** para a variável x.