

Sistemas Distribuídos

Relatório do Trabalho Prático

Sistema de apostas para jogo Euromilhões em cliente/servidor com gRPC

Alunos:

Diogo Fidalgo, 65631

Francisco Rodrigues, 64063

Ivo Soares Almeida, 66438

1.Introdução

Este trabalho consiste no desenvolvimento de um sistema de apostas baseado no “Euromilhões”. Este sistema envolve a implementação de três tipos de clientes usando a aplicação “Windows Forms” sendo esses:

- Utilizador
- Administrador
- Gestor

Os clientes utilizam bibliotecas e packages que permitiram a comunicação com o servidor, isto porque foi utilizado o “gRPC”.

Para além dos clientes, deve ser criado um servidor sob serviço “gRPC” e que contenha uma base de dados “SQL Server”. O servidor é responsável por implementar todas as funcionalidades necessárias aos clientes, este receberá pedidos através do canal de comunicação gRPC, após executar o pedido envia uma resposta para o cliente.

Todos os pedidos possíveis por parte do cliente estão definidos no protocolo de comunicação, denominado por “Proto”, que é partilhado tanto pelo cliente como pelo servidor, permitindo a ambos saber o tipo de mensagens que podem enviar e o tipo de mensagem que podem receber quando executado um rpc.

Foi optado por apenas criar um serviço geral, para permitir a facilidade de implementação.

2.Servidor e Serviços gRPC

O servidor foi implementado com uma base de dados com os atributos:

Nsorteio - Número atual do sorteio;

Nome - Nome e Apelido do apostador;

Telemóvel - Número de telemóvel do apostador;

Chave - Cinco números e duas chaves em sequência separados por “;”.

Como podem existir duas pessoas diferentes, mas com o mesmo nome decidimos criar o atributo “Telemóvel” que distingue os diferentes apostadores garantindo a sua singularidade.

Na base de dados, o atributo “nsorteio” serve para identificar o número do sorteio e poupa a necessidade de criar duas tabelas, uma para as apostas atual e outra para registar todas as apostas de todos os sorteios.

Quando o administrador decide criar um novo sorteio é inserida uma linha que contem o “nsorteio” incrementado e o resto dos parâmetros com a string “Administrador” para não haver redundâncias quando for eleito o vencedor. A figura seguinte é uma amostra da base de dados.

	id	nsorteio	nome	telemovel	chave
▶	1	1	administrador	administrador	administrador
	2	1	Ivo	938308944	1;2;3;4;5;6;7
	3	1	Ivo	938308944	11;22;33;44;51;6;7

Fig. 1 – Exemplo dos valores guardados na base de dados

Em relação aos serviços gRPC foi criado um ficheiro proto que contem os serviços disponibilizados pelo servidor. O proto “greet” (proto pré-definido) é serviço que já existe implementado pela Microsoft, em relação ao nosso trabalho, apenas foi usado para verificar se a ligação foi bem estabelecida.

O proto “pEuromilhões” (ver Anexo) contem um serviço para os três clientes, onde está definido todos os rpc’s que podem invocar.

Rpc’s do utilizador

1. rpc PedidoApostasFeitas (ApostasFeitasRequest) returns (stream ApostasFeitasReply);
- Deve permitir a visualização da lista de chaves em que o utilizador já apostou anteriormente.
2. rpc RegistrarAposta (ApostaRequest) returns (ApostaReply);
-Permitir o utilizador registar a aposta.

Rpc’s do administrador

1. rpc ApostaAtual (VisualizarApostaRequest) returns (stream VisualizarApostaReply);
-Visualizar sorteio atual e apostadores do sorteio.
2. rpc NovoSorteio (SorteioRequest) returns (SorteioReply);
-Arquivar sorteio atual e criar um novo.

Rpc’s do gestor

1. rpc Vencedor (ChaveRequest) returns (stream VencedorReply);
-Registar chave vencedora e verificar vencedores.

3.Cliente Utilizador

Este é composto por duas view que são “View_Utilizador.cs” e “View_ApostasFeitas”.

O utilizador na view “View_Utilizador.cs” consegue inserir a chave que pretende apostar e realizar a aposta. Quando clica no botão “Histórico de apostas” é remetido para a view “View_ApostasFeitas” composta por uma “ListView” que apresenta ao utilizador as apostas feitas do sorteio atual.

Os números e estrelas são ordenados por ordem crescente de modo a apresentar as apostas de uma forma mais coerente. Depois da ordenação, a variável “chave” contém os cinco números e estrelas, todos separados por “;”.

Quando o utilizador pretende registar a aposta é invocado o canal gRPC para que os seus dados e a sua chave sejam enviados para o servidor de modo que a aposta seja realizada.

Conforme a resposta por parte do servidor, é apresentada ao cliente uma caixa de mensagem contendo a informação se a aposta foi bem realizada ou não.

Se o utilizador pretender saber quais foram as chaves em que apostou, envia para o servidor uma mensagem com as suas informações, se não houver nenhuma falha, este vai reenviar todas as chaves, que serão apresentadas em uma ListView. Nesta função o servidor estará em modo streaming, pois há a necessidade de enviar várias mensagens para um só pedido.



Fig. 2 – View Cliente Utilizador

4. Cliente Administrador

O cliente administrador conta com duas funções a realizar na aplicação. Em primeiro, pode arquivar o sorteio atual e criar um novo, quando prime o botão “Novo Sorteio” é enviada uma mensagem ao servidor através do canal gRPC, este insere uma linha na base de dados com todos os valores como “administrador” e incrementa o valor “nsorteio”. A outra função permite ao administrador ver todos os apostadores da aposta atual e a chave que registaram, para isto, é enviada uma mensagem com um pedido para ver aposta atual. O servidor procura o maior número sorteio incluído na base de dados e reencaminha todas as linhas para o cliente em forma de lista. Após isto é iniciada uma nova view apresentando a lista reencaminhada pelo canal gRPC.

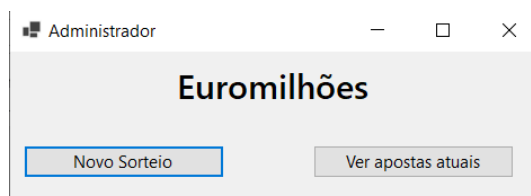


Fig. 3 – View Cliente Administrador

5. Cliente Gestor

Tal como o nome indica, este cliente apenas tem a função de gerir o sorteio. Sempre que desejar, pode inserir uma chave no sistema que será considerada como chave vencedora. A chave vencedora é enviada para o servidor, onde será comparada às restantes chaves apostadas pelos utilizadores. Quando concluída a comparação, o servidor envia os dados de todos os utilizadores que tenham apostado uma chave igual.

No cliente é apresentada a informação dos utilizadores vencedores.



The screenshot shows a window titled "Gestor" with a subtitle "Euromilhões". Below the subtitle is the instruction "Insira a chave vencedora". There are five input fields for numbers, labeled "1º Número" through "5º Número", and two input fields for stars, labeled "1ª Estrela" and "2ª Estrela". A button labeled "Ver vencedores do sorteio" is located at the bottom right of the input area.

Fig. 4 –View Cliente Gestor

6. Anexos

Proto

```
syntax = "proto3";

option csharp_namespace = "GrpcServer.Protos";

service ServicoGeral {

    //utilizador
    //Deve permitir a visualização da lista de chaves em que o utilizador já
    apostou anteriormente.
    rpc PedidoApostasFeitas (ApostasFeitasRequest) returns (stream
    ApostasFeitasReply);
    //Permitir o utilizador registar a aposta
    rpc RegistarAposta (ApostaRequest) returns (ApostaReply);

    //administrador
    //Visualizar sorteio atual
    rpc ApostaAtual (VisualizarApostaRequest) returns (stream
    VisualizarApostaReply);
    //Arquivar sorteio atual e criar um novo
    rpc NovoSorteio (SorteioRequest) returns (SorteioReply);

    //gestor
    //Verificar vencedores
    rpc Vencedor (ChaveRequest) returns (stream VencedorReply);
}

message ApostasFeitasRequest{
    string nome_apostador = 1;
    string telemovel = 2;
}

message ApostasFeitasReply{
    string ApostasFeitas = 1;
}

message ApostaRequest{
    string nome_apostador = 1;
    string telemovel = 2;
    string chave = 3;
}

message ApostaReply{
    string Resposta_Aposta = 1;
}

message VisualizarApostaRequest{
    string Pedido_Aposta_Atual = 1;
}

message VisualizarApostaReply{
    string nome_apostador = 1;
    string telemovel = 2;
    string chave = 3;
}

message SorteioRequest{
```

```
        string Arquivar_Criar = 1;
    }

    message SorteioReply{
        string Reposta_Arquivar_Criar = 1;
    }

    message ChaveRequest{
        string Chave_Vencedora = 1;
    }

    message VencedorReply{
        string nome_apostador = 1;
        string telemovel = 2;
    }
```

Servidor

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Data.SqlClient;
using System.Data;

namespace GrpcServer.Services
{
    public class pEuromilhoesService : ServicoGeral.ServicoGeralBase
    {
        private readonly ILogger<pEuromilhoesService> _logger;
        private SqlConnection basededados { get; set; }

        public pEuromilhoesService(ILogger<pEuromilhoesService> logger)
        {
            _logger = logger;
            basededados = new SqlConnection("Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\ijpsa\\Documents\\bd_e
uromilhoes.mdf;Integrated Security=True;Connect Timeout=30");
            basededados.Open();
        }

        //Utilizador
        public override Task<ApostaReply> RegistrarAposta(ApostaRequest request,
ServerCallContext context)
        {
            ApostaReply reply = new ApostaReply();
            SqlCommand comando = new SqlCommand();
            string aux_executar;
            string aux_numero_sorteio;
            int sorteio = 0;

            try
            {
                //Numero do sorteio atual
                aux_numero_sorteio = string.Format("select nsorteio from sorteios
order by id desc");
                comando = new SqlCommand(aux_numero_sorteio, basededados);
                SqlDataReader reader = comando.ExecuteReader();
                reader.Read();
                sorteio = Convert.ToInt32(reader["nsorteio"]);

                if (sorteio == 0)
                {
                    sorteio = 1;
                }

                reader.Close();

                aux_executar = string.Format("insert into sorteios(nsorteio,
nome, telemovel, chave) values ({0} , '{1}', '{2}', '{3}') ",
                sorteio, request.NomeApostador, request.Telemovel,
request.Chave);

                comando = new SqlCommand(aux_executar, basededados);
                comando.ExecuteNonQuery();

                reply.RespostaAposta = "Aposta Feita!";
            }
            catch { }
        }
    }
}
```



```
    }  
    catch  
    {  
        reply.RespostaAposta = "Erro ao realizar aposta!";  
    }  
  
    return Task.FromResult(reply);  
}  
  
public override async Task PedidoApostasFeitas(ApostasFeitasRequest  
request, IServerStreamWriter<ApostasFeitasReply> responseStream,  
ServerCallContext context)  
{  
    string telemovel = request.Telemovel;  
    var apostas = Obter_Apostas(telemovel);  
    foreach (var aposta in apostas)  
    {  
        await Task.Delay(1000);  
        await responseStream.WriteAsync(aposta);  
    }  
}  
  
private List<ApostasFeitasReply> Obter_Apostas(string telemovel)  
{  
    List<ApostasFeitasReply> apostas = new List<ApostasFeitasReply>();  
    SqlCommand comando = new SqlCommand();  
    int sorteio = 0;  
  
    // Numero do sorteio atual  
    string aux_numero_sorteio = string.Format("select nsorteio from  
sorteios order by id desc");  
    comando = new SqlCommand(aux_numero_sorteio, basededados);  
    SqlDataReader reader = comando.ExecuteReader();  
    reader.Read();  
    sorteio = Convert.ToInt32(reader["nsorteio"]);  
  
    if (sorteio == 0)  
    {  
        sorteio = 1;  
    }  
  
    reader.Close();  
  
    string aux_executar = string.Format("SELECT chave FROM sorteios WHERE  
telemovel='{0}' ", telemovel);  
    comando = new SqlCommand(aux_executar, basededados);  
    reader = comando.ExecuteReader();  
    while (reader.Read())  
    {  
        ApostasFeitasReply aposta = new ApostasFeitasReply();  
        aposta.ApostasFeitas = Convert.ToString(reader["chave"]);  
        apostas.Add(aposta);  
    }  
  
    return apostas;  
}  
  
//Administrador  
public override Task<SorteioReply> NovoSorteio(SorteioRequest request,  
ServerCallContext context)  
{  
    SorteioReply reply = new SorteioReply();
```

```
SqlCommand comando = new SqlCommand();
string aux_executar;
string aux_numero_sorteio;
int sorteio;
string admin_nome = "administrador", admin_telemovei =
"administrado", admin_chave = "administrador";

if (request.ArquivarCriar == "Novo")
{
    try
    {
        //Numero do sorteio atual
        aux_numero_sorteio = string.Format("select nsorteio from
sorteios order by id desc");
        comando = new SqlCommand(aux_numero_sorteio, basededados);
        SqlDataReader reader = comando.ExecuteReader();
        reader.Read();
        sorteio = Convert.ToInt32(reader["nsorteio"]);

        sorteio++;

        reader.Close();

        aux_executar = string.Format("insert into sorteios(nsorteio,
nome, telemovei, chave) values ({0} , '{1}', '{2}', '{3}')" ,
        sorteio, admin_nome, admin_telemovei, admin_chave);

        comando = new SqlCommand(aux_executar, basededados);
        comando.ExecuteNonQuery();

        reply.RepostaArquivarCriar = "Sorteio arquivado! Novo sorteio
criado";
    }
    catch
    {
        reply.RepostaArquivarCriar = "Erro ao criar novo sorteio!";
    }
}
else
{
    reply.RepostaArquivarCriar = "Erro ao criar novo sorteio!";
}

return Task.FromResult(reply);
}

public override async Task ApostaAtual(VisualizarApostaRequest request,
IStreamWriter<VisualizarApostaReply> responseStream, ServerCallContext
context)
{
    var apostas = Obter_ApostaAtual();
    foreach (var aposta in apostas)
    {
        await Task.Delay(1000);
        await responseStream.WriteAsync(aposta);
    }
}

private List<VisualizarApostaReply> Obter_ApostaAtual()
{
    List<VisualizarApostaReply> apostas = new
List<VisualizarApostaReply>();
}
```

```
SqlCommand comando = new SqlCommand();
int sorteio = 0;

// Numero do sorteio atual
string aux_numero_sorteio = string.Format("select nsorteio from
sorteios order by id desc");
comando = new SqlCommand(aux_numero_sorteio, basededados);
SqlDataReader reader = comando.ExecuteReader();
reader.Read();
sorteio = Convert.ToInt32(reader["nsorteio"]);

if (sorteio == 0)
{
    sorteio = 1;
}

reader.Close();

string aux_executar = string.Format("SELECT nome, telemovel, chave
FROM sorteios WHERE nsorteio={0} and nome <> 'administrador'", sorteio);
comando = new SqlCommand(aux_executar, basededados);
reader = comando.ExecuteReader();
while (reader.Read())
{
    VisualizarApostaReply aposta = new VisualizarApostaReply();
    aposta.NomeApostador = Convert.ToString(reader["nome"]);
    aposta.Telemovel = Convert.ToString(reader["telemovel"]);
    aposta.Chave = Convert.ToString(reader["chave"]);
    apostas.Add(aposta);
}

return apostas;
}

//Gestor
public override async Task Vencedor(ChaveRequest request,
IStreamWriter<VencedorReply> responseStream, ServerCallContext context)
{
    string chave = request.ChaveVencedora;
    var vencedores = Obter_Vencedores(chave);
    foreach (var vencedor in vencedores)
    {
        await Task.Delay(1000);
        await responseStream.WriteAsync(vencedor);
    }
}

private List<VencedorReply> Obter_Vencedores(string chave)
{
    List<VencedorReply> vencedores = new List<VencedorReply>();
    SqlCommand comando = new SqlCommand();
    int sorteio = 0;

    // Numero do sorteio atual
    string aux_numero_sorteio = string.Format("select nsorteio from
sorteios order by id desc");
    comando = new SqlCommand(aux_numero_sorteio, basededados);
    SqlDataReader reader = comando.ExecuteReader();
    reader.Read();
    sorteio = Convert.ToInt32(reader["nsorteio"]);

    if (sorteio == 0)
```

```
        {
            sorteio = 1;
        }

        reader.Close();

        string aux_executar = string.Format("SELECT nome, telemovel FROM
sorteios WHERE chave='{0}' and nsorteio={1}", chave, sorteio);
        comando = new SqlCommand(aux_executar, basededados);
        reader = comando.ExecuteReader();
        while (reader.Read())
        {
            VencedorReply vencedor = new VencedorReply();
            vencedor.NomeApostador= Convert.ToString(reader["nome"]);
            vencedor.Telemovel = Convert.ToString(reader["telemovel"]);
            vencedores.Add(vencedor);
        }

        return vencedores;
    }
}
```

Cliente Utilizador

```
using Grpc.Core;
using Grpc.Net.Client;
using GrpcServer;
using GrpcServer.Protos;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Cliente_Utilizador
{
    public partial class View_Utilizador : Form
    {
        public View_Utilizador()
        {
            InitializeComponent();
            //Ligação ao servidor
            var httpHandler = new HttpClientHandler();
            // Return `true` to allow certificates that are untrusted/invalid
            httpHandler.ServerCertificateCustomValidationCallback =
                HttpClientHandler.DangerousAcceptAnyServerCertificateValidator;

            string ipservidor = "https://localhost:5001";
            var channel = GrpcChannel.ForAddress(ipservidor, new
                GrpcChannelOptions { HttpHandler = httpHandler });
            var client = new Greeter.GreeterClient(channel);

            //Verificar ligação
            var reply = client.SayHello(new HelloRequest { Name = "Apostador" });
            if (reply.Message == "Ligação Feita! Boas Apostas!")
            {
                MessageBox.Show("Ligação Estabelecida!", "Utilizador - Ligação",
                    MessageBoxButtons.OK);
            }
            else
            {
                MessageBox.Show("Erro ao estabelecer ligação!", "Utilizador -
                    Ligação", MessageBoxButtons.OK);
            }
        }

        private void b_apostar_Click(object sender, EventArgs e)
        {
            //Ligação ao servidor
            string ipservidor = "https://localhost:5001";
            var channel = GrpcChannel.ForAddress(ipservidor);

            //Outras variaveis
            string nome;
            string telemovel;
            string chave = "";
        }
    }
}
```

```
int[] numeros = new int[5];
int[] estrelas = new int[2];
int aux;

if (tb_nome.Text == "")
{
    MessageBox.Show("Insira o seu nome!", "", MessageBoxButtons.OK);
}
else
{
    if (tb_telemove1.Text == "" || tb_numero_1.Text == "" ||
tb_numero_2.Text == "" || tb_numero_3.Text == "" || tb_numero_4.Text == "" ||
tb_numero_5.Text == "" || tb_estrela_1.Text == "" || tb_estrela_2.Text == "")
    {
        MessageBox.Show("Insira uma chave valida", "",
MessageBoxButtons.OK);
    }
    else
    {
        nome = tb_nome.Text;
        telemove1 = tb_telemove1.Text;
        numeros[0] = Convert.ToInt32(tb_numero_1.Text);
        numeros[1] = Convert.ToInt32(tb_numero_2.Text);
        numeros[2] = Convert.ToInt32(tb_numero_3.Text);
        numeros[3] = Convert.ToInt32(tb_numero_4.Text);
        numeros[4] = Convert.ToInt32(tb_numero_5.Text);
        estrelas[0] = Convert.ToInt32(tb_estrela_1.Text);
        estrelas[1] = Convert.ToInt32(tb_estrela_2.Text);

        //Ordenar
        for (int k = 1; k <= 4; k++)
        {
            for (int j = 0; j <= 4 - k; j++)
            {
                if (numeros[j] > numeros[j + 1])
                {
                    aux = numeros[j];
                    numeros[j] = numeros[j + 1];
                    numeros[j + 1] = aux;
                }
            }
        }

        if(estrelas[1] < estrelas[0])
        {
            aux = estrelas[1];
            estrelas[1] = estrelas[0];
            estrelas[0] = aux;
        }

        for (int i = 0; i<= 4; i++)
        {
            if(i == 0)
            {
                chave = Convert.ToString(numeros[i]);
            }
            else
            {
                chave = chave + ";" + Convert.ToString(numeros[i]);
            }
        }
    }
}
```

```
        for(int j = 0; j<=1; j++)
        {
            chave = chave + ";" + Convert.ToString(estrelas[j]);
        }

        var client = new ServicoGeral.ServicoGeralClient(channel);
        var reply = client.RegistarAposta(new ApostaRequest {
NomeApostador = nome, Telemovel=telemovel, Chave = chave });
        MessageBox.Show(reply.RespostaAposta, "",
        MessageBoxButtons.OK);
    }
}

private async void b_historico_apostas_Click(object sender, EventArgs e)
{
    string nome = tb_nome.Text;
    string telemovel = tb_telemovel.Text;
    List<ApostasFeitasReply> lista_apostas = new
List<ApostasFeitasReply>();

    string ipservidor = "https://localhost:5001";
    var channel = GrpcChannel.ForAddress(ipservidor);
    var client = new ServicoGeral.ServicoGeralClient(channel);
    using var reply = client.PedidoApostasFeitas(new ApostasFeitasRequest
{ NomeApostador = nome, Telemovel = telemovel });

    await foreach (var aposta in reply.ResponseStream.ReadAllAsync())
    {
        lista_apostas.Add(aposta);
    }

    View_ApostasFeitas viewapostas = new
View_ApostasFeitas(lista_apostas);
    viewapostas.Show();
}
}
```

*Cliente Utilizador (View secundária)

```
using Grpc.Core;
using Grpc.Net.Client;
using GrpcServer.Protos;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Cliente_Utilizador
{
    public partial class View_ApostasFeitas : Form
    {
        List<ApostasFeitasReply> _lista_apostas = new List<ApostasFeitasReply>();
        public View_ApostasFeitas(List<ApostasFeitasReply> lista)
        {
            InitializeComponent();
            _lista_apostas = lista;
        }

        private void View_ApostasFeitas_Load(object sender, EventArgs e)
        {
            int i = 0;

            foreach(var aposta in _lista_apostas)
            {
                lt_apostasfeitas.Items.Add("Aposta número " +
                Convert.ToString(i+1) + "=>" + aposta.ApostasFeitas);
                i++;
            }

            lt_apostasfeitas.Show();
        }
    }
}
```


Cliente Administrador

```
using Grpc.Core;
using Grpc.Net.Client;
using GrpcServer;
using GrpcServer.Protos;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Cliente_Administrador
{
    public partial class View_Administrador : Form
    {
        public View_Administrador()
        {
            InitializeComponent();

            // Ligação ao servidor
            var httpHandler = new HttpClientHandler();
            // Return `true` to allow certificates that are untrusted/invalid
            httpHandler.ServerCertificateCustomValidationCallback =
            HttpClientHandler.DangerousAcceptAnyServerCertificateValidator;

            string ipservidor = "https://localhost:5001";
            var channel = GrpcChannel.ForAddress(ipservidor, new
            GrpcChannelOptions { HttpHandler = httpHandler });
            var client = new Greeter.GreeterClient(channel);

            //Verificar ligação
            var reply = client.SayHello(new HelloRequest { Name = "Apostador" });
            if (reply.Message == "Ligação Feita! Boas Apostas!")
            {
                MessageBox.Show("Ligação Estabelecida!", "Administrador -
                Ligação", MessageBoxButtons.OK);
            }
            else
            {
                MessageBox.Show("Erro ao estabelecer ligação!", "Administrador -
                Ligação", MessageBoxButtons.OK);
            }
        }

        private void b_novosorteio_Click(object sender, EventArgs e)
        {
            //Ligação ao servidor
            string ipservidor = "https://localhost:5001";
            var channel = GrpcChannel.ForAddress(ipservidor);

            var client = new ServicoGeral.ServicoGeralClient(channel);
            var reply = client.NovoSorteio(new SorteioRequest { ArquivarCriar =
            "Novo"});
        }
    }
}
```

```
        MessageBox.Show(reply.RepostaArquivarCriar, "",  
        MessageBoxButtons.OK);  
    }  
  
    private async void b_apostasatuais_Click(object sender, EventArgs e)  
    {  
        List<VisualizarApostaReply> lista_apostas = new  
        List<VisualizarApostaReply>();  
  
        string ipservidor = "https://localhost:5001";  
        var channel = GrpcChannel.ForAddress(ipservidor);  
        var client = new ServicoGeral.ServicoGeralClient(channel);  
        using var reply = client.ApostaAtual(new VisualizarApostaRequest {  
        PedidoApostaAtual = "VerApostas" });  
  
        await foreach (var aposta in reply.ResponseStream.ReadAllAsync())  
        {  
            lista_apostas.Add(aposta);  
        }  
  
        View_VisualizarApostas viewapostas = new  
        View_VisualizarApostas(lista_apostas);  
        viewapostas.Show();  
    }  
}
```

*Cliente Administrador (View secundária)

```
using GrpcServer.Protos;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace Cliente_Administrador
{
    public partial class View_VisualizarApostas : Form
    {
        List<VisualizarApostaReply> _lista_apostas = new
        List<VisualizarApostaReply>();

        public View_VisualizarApostas(List<VisualizarApostaReply> lista)
        {
            InitializeComponent();
            _lista_apostas = lista;
        }

        private void View_VisualizarApostas_Load(object sender, EventArgs e)
        {
            foreach (var aposta in _lista_apostas)
            {
                lt_VisualizarApostas.Items.Add("Nome: " + aposta.NomeApostador +
                "   Telemóvel: " + aposta.Telemovei + "   Chave: " + aposta.Chave);
            }

            lt_VisualizarApostas.Show();
        }
    }
}
```

Cliente Gestor

```
using Grpc.Core;
using Grpc.Net.Client;
using GrpcServer;
using GrpcServer.Protos;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Cliente_Gestor
{
    public partial class View_Gestor : Form
    {
        public View_Gestor()
        {
            InitializeComponent();

            //Ligação ao servidor
            var httpHandler = new HttpClientHandler();
            // Return `true` to allow certificates that are untrusted/invalid
            httpHandler.ServerCertificateCustomValidationCallback =
            HttpClientHandler.DangerousAcceptAnyServerCertificateValidator;

            string ipservidor = "https://localhost:5001";
            var channel = GrpcChannel.ForAddress(ipservidor, new
            GrpcChannelOptions { HttpHandler = httpHandler });
            var client = new Greeter.GreeterClient(channel);

            //Verificar ligação
            var reply = client.SayHello(new HelloRequest { Name = "Apostador" });
            if(reply.Message == "Ligação Feita! Boas Apostas!")
            {
                MessageBox.Show("Ligação Estabelecida!", "Gestor - Ligação",
                MessageBoxButtons.OK);
            }
            else
            {
                MessageBox.Show("Erro ao estabelecer ligação!", "Gestor -
                Ligação", MessageBoxButtons.OK);
            }
        }

        private async void b_chave_vencedora_Click(object sender, EventArgs e)
        {
            //Ligação ao servidor
            string ipservidor = "https://localhost:5001";
            var channel = GrpcChannel.ForAddress(ipservidor);

            //Outras variaveis
            string chave = "";
            int[] numeros = new int[5];
            int[] estrelas = new int[2];
        }
    }
}
```

```
int aux;

if (tb_numero_1.Text == "" || tb_numero_2.Text == "" ||
tb_numero_3.Text == "" || tb_numero_4.Text == "" || tb_numero_5.Text == "" ||
tb_estrela_1.Text == "" || tb_estrela_2.Text == "")
{
    MessageBox.Show("Insira uma chave valida", "",
    MessageBoxButtons.OK);
}
else
{

    numeros[0] = Convert.ToInt32(tb_numero_1.Text);
    numeros[1] = Convert.ToInt32(tb_numero_2.Text);
    numeros[2] = Convert.ToInt32(tb_numero_3.Text);
    numeros[3] = Convert.ToInt32(tb_numero_4.Text);
    numeros[4] = Convert.ToInt32(tb_numero_5.Text);
    estrelas[0] = Convert.ToInt32(tb_estrela_1.Text);
    estrelas[1] = Convert.ToInt32(tb_estrela_2.Text);

    //Ordenar
    for (int k = 1; k <= 4; k++)
    {
        for (int j = 0; j <= 4 - k; j++)
        {
            if (numeros[j] > numeros[j + 1])
            {
                aux = numeros[j];
                numeros[j] = numeros[j + 1];
                numeros[j + 1] = aux;
            }
        }
    }

    if (estrelas[1] < estrelas[0])
    {
        aux = estrelas[1];
        estrelas[1] = estrelas[0];
        estrelas[0] = aux;
    }

    for (int i = 0; i <= 4; i++)
    {
        if (i == 0)
        {
            chave = Convert.ToString(numeros[i]);
        }
        else
        {
            chave = chave + ";" + Convert.ToString(numeros[i]);
        }
    }

    for (int j = 0; j <= 1; j++)
    {
        chave = chave + ";" + Convert.ToString(estrelas[j]);
    }

    var client = new ServicoGeral.ServicoGeralClient(channel);
    using var reply = client.Vencedor(new ChaveRequest {
ChaveVencedora = chave });
}
```

```
                await foreach (var vencedor in
reply.ResponseStream.ReadAllAsync())
            {
                MessageBox.Show("Nome: " + vencedor.NomeApostador + " |
Telemove1: " + vencedor.Telemove1, "Vencedores", MessageBoxButtons.OK);
            }
            //var reply = await client.Vencedor(new ChaveRequest {
ChaveVencedora = chave });
            //MessageBox.Show(reply.NomeApostador + reply.Telemove1,
"Vencedores", MessageBoxButtons.OK);
        }
    }
}
```