gRPCclient

```
using Grpc.Net.Client;
using System;
using System.Threading.Tasks;

namespace gRPCclient
{
    class Program
    {
        static async Task Main(string[] args)
        {
            MyData.info();
            Console.WriteLine("Starting gRPC Client");
            using var channel = GrpcChannel.ForAddress("https://localhost:5001");
            var client = new GrpcService.GrpcServiceClient(channel);
            Console.Write("Enter the name: ");
            String str = Console.ReadLine();
            Console.Write("Enter age: ");
            int val = int.Parse(Console.ReadLine());
            var reply = await client.GrpcProcAsync(new GrpcRequest
            {
                Name = str,
                Age = val
            });
            Console.WriteLine("From server: " + reply.Message);
            Console.WriteLine("From server: " + val + " years = " + reply.Days + " days");
            Console.WriteLine("Press any key to exit...");
            Console.ReadKey();
            channel.ShutdownAsync().Wait();
        }
    }
}
```

```
// Include namespace system
using System;
using System.Net;

public class MyData
{
    public static void info()
    {
        var dt = DateTime.Now.ToString("yyyy/MM/dd");
        Console.WriteLine("yyyy/MM/dd " + dt);
        Console.WriteLine("Data:Alvaro Lopez Pereda and Alberto Hernandez Lado");
        Console.WriteLine("UserName: {0}", Environment.UserName);
        Console.WriteLine("Operating System Name:" + Environment.MachineName);
        String strHostName = string.Empty;
        IPHostEntry ipEntry = Dns.GetHostEntry(Dns.GetHostName());
        IPAddress[] addr = ipEntry.AddressList;

        for (int i = 0; i < addr.Length; i++)
        {
            Console.WriteLine("IP Address {0}: {1} ", i, addr[i].ToString());
        }
    }
}
```

```protobuf
syntax = "proto3";

option csharp_namespace = "gRPCclient";

package mygrpc;

// The greeting service definition.
service GrpcService {
  // Sends a greeting
  rpc GrpcProc (GrpcRequest) returns (GrpcResponse) {}
}

// The request message containing the user's name.
message GrpcRequest {
  string name = 1;
  int32 age = 2;
}

// The response message containing the greetings.
message GrpcResponse {
  string message = 1;
  int32 days = 2;
}
```

gRPCserver

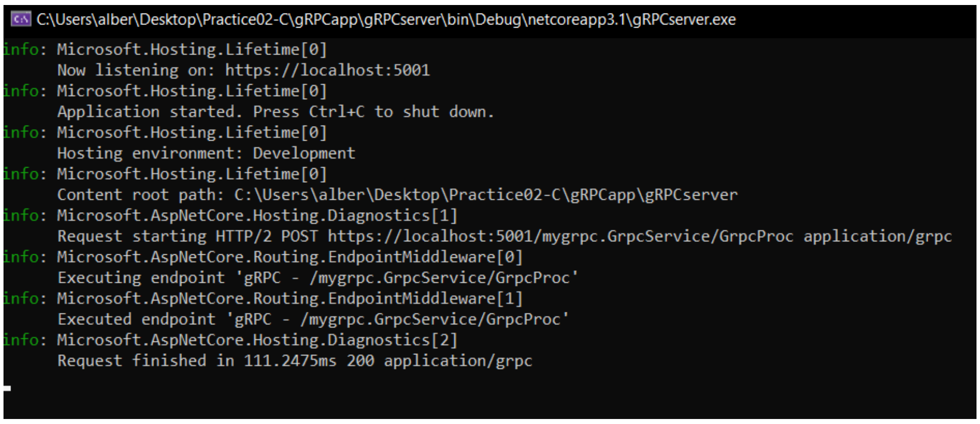
```csharp
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Hosting;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;

namespace gRPCserver
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        // Additional configuration is required to successfully run gRPC on macOS.
        // For instructions on how to configure Kestrel and gRPC clients on macOS, visit https://go.microsoft.com
/fwlink/?linkid=2099682
        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }
}
```

```csharp
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace gRPCserver
{
    public class Startup
    {
        // This method gets called by the runtime. Use this method to add services to the container.
        // For more information on how to configure your application, visit https://go.microsoft.com/fwlink
/?LinkID=398940
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddGrpc();
        }

        // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }

            app.UseRouting();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapGrpcService<MyGrpcService>();

                endpoints.MapGet("/", async context =>
                {
                    await context.Response.WriteAsync("Communication with gRPC endpoints must be made through a gRPC
client. To learn how to create a client, visit: https://go.microsoft.com/fwlink/?linkid=2086909");
                });
            });
        }
    }
}
```

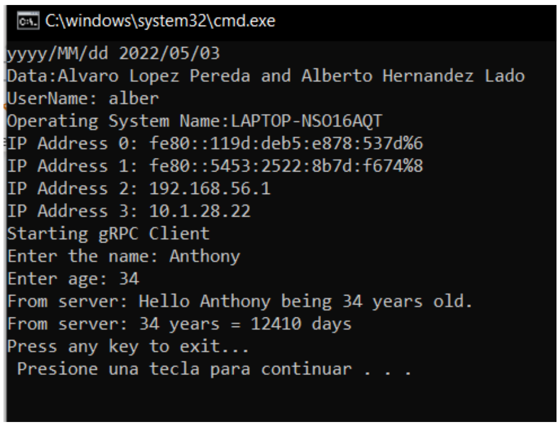
```
using Grpc.Core;
using Microsoft.Extensions.Logging;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace gRPCserver
{
    public class MyGrpcService : GrpcService.GrpcServiceBase
    {
        private readonly ILogger<MyGrpcService> _logger;
        public MyGrpcService(ILogger<MyGrpcService> logger)
        {
            _logger = logger;
        }

        public override Task<GrpcResponse> GrpcProc(GrpcRequest request, ServerCallContext context)
        {
            string msg;
            int val;

            val = request.Age * 365;
            msg = "Hello "+request.Name+" being " + request.Age + " years old. ";
            return Task.FromResult(new GrpcResponse {
                Message=msg, Days=val });
        }
    }
}
```

```
syntax = "proto3";

option csharp_namespace = "gRPCserver";

package mygrpc;

// The greeting service definition.
service GrpcService {
  // Sends a greeting
  rpc GrpcProc (GrpcRequest) returns (GrpcResponse);
}

// The request message containing the user's name.
message GrpcRequest {
  string name = 1;
  int32 age = 2;
}

// The response message containing the greetings.
message GrpcResponse {
  string message = 1;
  int32 days = 2;
}
```

Test



```
C:\Users\alber\Desktop\Practice02-C\gRPCapp\gRPCserver\bin\Debug\netcoreapp3.1\gRPCserver.exe
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\alber\Desktop\Practice02-C\gRPCapp\gRPCserver
info: Microsoft.AspNetCore.Hosting.Diagnostics[1]
      Request starting HTTP/2 POST https://localhost:5001/mygrpc.GrpcService/GrpcProc application/grpc
info: Microsoft.AspNetCore.Routing.EndpointMiddleware[0]
      Executing endpoint 'gRPC - /mygrpc.GrpcService/GrpcProc'
info: Microsoft.AspNetCore.Routing.EndpointMiddleware[1]
      Executed endpoint 'gRPC - /mygrpc.GrpcService/GrpcProc'
info: Microsoft.AspNetCore.Hosting.Diagnostics[2]
      Request finished in 111.2475ms 200 application/grpc
```



```
C:\windows\system32\cmd.exe
yyyy/MM/dd 2022/05/03
Data:Alvaro Lopez Pereda and Alberto Hernandez Lado
UserName: alber
Operating System Name:LAPTOP-NSO16AQT
IP Address 0: fe80::119d:deb5:e878:537d%6
IP Address 1: fe80::5453:2522:8b7d:f674%8
IP Address 2: 192.168.56.1
IP Address 3: 10.1.28.22
Starting gRPC Client
Enter the name: Anthony
Enter age: 34
From server: Hello Anthony being 34 years old.
From server: 34 years = 12410 days
Press any key to exit...
 Presione una tecla para continuar . . .
```

It executes the server side first and show info class and greet the person who enter the data followed by the age converted in days.