**BÁO CÁO CÔNG VIỆC**

**Công việc số:** 61 (Làm thêm Console)

**Mô tả công việc:** Kiểm tra thùng hàng có nắp hay không dựa trên ảnh (Console .NET FrameWork 4.7.2)

**Người thực hiện:** Hạ Quang Dũng

**Ngày bắt đầu:** 17/12/2024

**Ngày kết thúc:** 18/12/2024

**NỘI DUNG TÀI LIỆU**

[**1. Cấu trúc thư mục 2**](#_gtx1surkrabd)

[**2. Các thư viện cần cài đặt 2**](#_accveiuju6qy)

[**3. Chi tiết mã nguồn 3**](#_qa2zaawtsxt4)

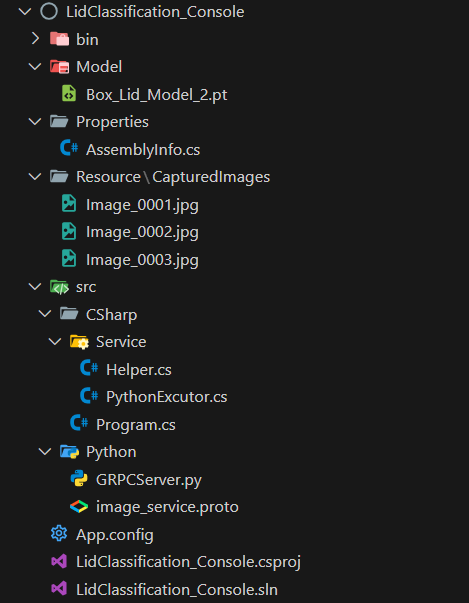
[3.1. FIle Proto và cấu hình để sinh class tự động khi build 3](#_gzrbgtir0jzw)

[3.2. File GRPCServer.py 4](#_biss0frx0cyr)

[3.3. File PythonExcutor.cs 4](#_2wms9lw8tjch)

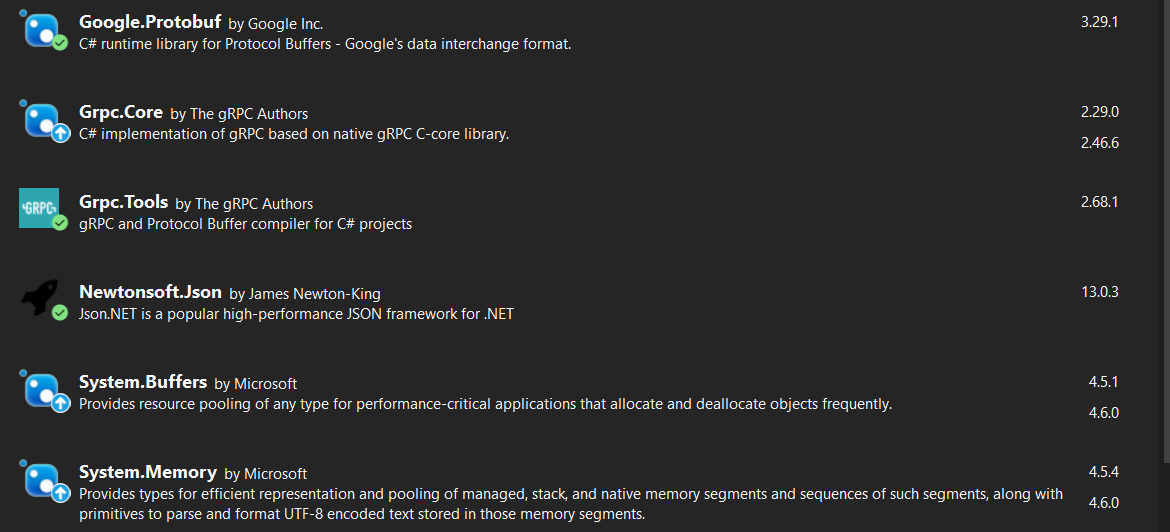
[3.4. File Program.cs 5](#_yv2wfaihiibs)

# 1. Cấu trúc thư mục



# 2. Các thư viện cần cài đặt

* Google.Protobuf (Version 3.29.1)
* Grpc.Core (Version 2.29.0)
* Grpc.Tools (Version 2.68.1)
* Newtonsoft.Json (Version 13.0.3)
* System.Buffers (Version 4.5.1)
* System.Memory (Version 4.5.4)
* System.Threading.Channels (Version 5.0.0)





# 3. Chi tiết mã nguồn

## 3.1. FIle Proto và cấu hình để sinh class tự động khi build

**image\_service.proto**

| syntax = "proto3";  service ImageTransfer {  rpc SendImage (ImageRequest) returns (ImageResponse);  }  message ImageRequest {  string path = 1;  string model\_path = 2;  }  message ImageResponse {  bool success = 1;  string message = 2;  string data = 3;  } |
| --- |

**LidClassification\_Console.csproj** (Thêm phần sau vào cuối)

| <ItemGroup>  <Protobuf Include="src\Python\image\_service.proto" GrpcServices="Client">  <Generator>MSBuild:Compile</Generator>  <CompileOutputs>true</CompileOutputs>  <Access>Public</Access>  </Protobuf>  </ItemGroup> |
| --- |

## 3.2. File GRPCServer.py

| from yolo\_grpc import serve  serve() |
| --- |

## 3.3. File PythonExcutor.cs

| using System.Diagnostics;  namespace VehicleDetection.src.CSharp.Services  {  public class PythonExecutor  {  private string pythonPath; // Đường dẫn đến interpreter Python  private string scriptPath; // Đường dẫn đến script Python  public PythonExecutor(string pythonPath, string scriptPath)  {  this.pythonPath = pythonPath;  this.scriptPath = scriptPath;  }  public string Execute(string args)  {  var startInfo = new ProcessStartInfo  {  FileName = pythonPath,  Arguments = $"{scriptPath} {args}", // Thêm các đối số cần thiết  RedirectStandardOutput = true,  UseShellExecute = false,  CreateNoWindow = true  };  using (var process = Process.Start(startInfo))  {  using (var reader = process.StandardOutput)  {  string result = reader.ReadToEnd();  process.WaitForExit();  return result;  }  }  }  }  } |
| --- |

## 3.4. File Program.cs

| using Grpc.Core;  using Newtonsoft.Json;  using System;  using System.Collections.Generic;  using System.Diagnostics;  using System.IO;  using System.Runtime.Remoting.Channels;  using System.Text;  using System.Threading.Tasks;  using VehicleDetection.src.CSharp.Services;  namespace LidClassification\_Console  {  class Program  {  private static string \_rootDir;  private static string \_resourceDir;  private static string \_modelDir;  private static string \_capturedImagesDir;  private static string \_modelBoxLidPath;  private static string grpcHost = "localhost:50051"; // Địa chỉ server gRPC (không cần http://)  static async Task Main(string[] args)  {  Console.OutputEncoding = Encoding.UTF8;  InitializePath();  InitializeServerGrpc();  string imagePath = Path.Combine(\_capturedImagesDir, "Image\_0001.jpg");  string imagePath2 = Path.Combine(\_capturedImagesDir, "Image\_0002.jpg");  string imagePath3 = Path.Combine(\_capturedImagesDir, "Image\_0002.jpg");  Console.WriteLine("Đang xử lý ảnh 1:"); await btnDetect(imagePath);  Console.WriteLine("Đang xử lý ảnh 2:"); await btnDetect(imagePath2);  Console.WriteLine("Đang xử lý ảnh 3:"); await btnDetect(imagePath3);  Console.WriteLine("Nhấn Enter để thoát...");  Console.ReadLine();  }  private static void InitializePath()  {  \_rootDir = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "..", "..");  \_resourceDir = Path.Combine(\_rootDir, "Resource");  \_modelDir = Path.Combine(\_rootDir, "Model");  \_capturedImagesDir = Path.Combine(\_resourceDir, "CapturedImages");  \_modelBoxLidPath = Path.Combine(\_modelDir, "box\_lid\_model\_2.pt");  }  private static void InitializeServerGrpc()  {  string pythonScriptPath = Path.Combine(\_rootDir, "src", "Python", "GRPCServer.py");  PythonExecutor pythonExecutor = new PythonExecutor("python", pythonScriptPath);  pythonExecutor.Execute("");  }  private static async Task btnDetect(string imagePath)  {  // Kiểm tra sự tồn tại của ảnh  if (!File.Exists(imagePath))  {  Console.WriteLine("Ảnh không tồn tại: " + imagePath);  return;  }  Stopwatch stopwatch = new Stopwatch();  stopwatch.Start();  var detections = await gRpc(imagePath);  stopwatch.Stop();  Console.WriteLine($"Thời gian xử lý: {stopwatch.Elapsed.TotalSeconds} giây");  if (isWithLid(detections))  {  Console.WriteLine("Hộp có nắp");  }  else  {  Console.WriteLine("Hộp không có nắp");  }  }  private static async Task<List<dynamic>> gRpc(string imagePath)  {  try  {  // Địa chỉ server gRPC  var channel = new Channel(grpcHost, ChannelCredentials.Insecure); // Tạo channel với Grpc.Core  // Tạo client gRPC  var client = new ImageTransfer.ImageTransferClient(channel);  // Gửi yêu cầu tới server  var request = new ImageRequest { Path = imagePath, ModelPath = \_modelBoxLidPath };  // Gửi yêu cầu với timeout  using (var cts = new System.Threading.CancellationTokenSource(TimeSpan.FromSeconds(30))) // Thiết lập timeout  {  var response = await client.SendImageAsync(request, cancellationToken: cts.Token);  // Xử lý phản hồi  if (response.Success)  {  //Console.WriteLine("Message: " + response.Message);  // Phân tích JSON thành danh sách dynamic  List<dynamic> detections = JsonConvert.DeserializeObject<List<dynamic>>(response.Data);  // Trả về danh sách phát hiện  return detections;  }  else  {  Console.WriteLine("Error: " + response.Message);  return new List<dynamic>(); // Trả về danh sách rỗng nếu có lỗi  }  }  }  catch (Exception ex)  {  Console.WriteLine($"An error occurred: {ex.Message}");  return new List<dynamic>(); // Trả về danh sách rỗng nếu có ngoại lệ  }  }  private static bool isWithLid(dynamic boundingBoxes)  {  foreach (var box in boundingBoxes)  {  if (box.label.ToString() == "Lid")  {  return true;  }  }  return false;  }  }  } |
| --- |