Software developer assignment applicant

Didates

May 28, 2021

1 Introduction

Applicants for the software developer position are asked at Didata to carry out an assignment. A deadline is described in the e-mail in which this assignment was sent. The application must be submitted by the specified date.

The assignment consists of two parts. In the first part, the applicant is asked to make an implementation based on a specification given by Didata. This task is described in section 2. The second part consists of reviewing a piece of code. This task is in section 3. Section 4 contains information on how to submit the assignment.

The document should be clear enough to complete the assignment. If there are still questions, please contact the e-mail address in which the applicant received this assignment. The assignment is intended to be carried out individually. The assignment is designed so that it shouldn't take too much time, a rainy Sunday evening should be enough.

2 Implementation

In this part, the applicant is asked to write a program in C# based on a small assignment specification. The result of this part is a console application that implements the specification. There are a few requirements for the implementation itself:

1. The effect must be created in a git repository.

2. The elaboration must be written in C#.

3. The repository must contain a Visual Studio project. Visual Studio can do this[1] be used, or the dotnet tool[2].

4. The project must be of console application type.

5. The project uses .NET version >= 5. External libraries are also allowed.

6. A plain-text file must be added to the repository with a short description of how the applicant approached the assignment.

2.1 Program

The program has to read files in JSON format, perform some editing and checking, and then write output in CSV format. Data is passed to the application via arguments.

The program has two modes. In the first mode, the -d option gives a path name. The application must then process all .json files in the specified path. In the second mode, the -f option gives a file name. This option may be entered multiple times. In both cases an argument must also be passed along with a file name. The application must write the output to this file. See list 1 for a complete overview of the options.

A JSON input file contains information about an order. One order is described in a file. An order has some fields with general information and a list of articles. See list 5 for an example and list 6 for a description of the fields.

For the CSV output, a line is written for each order that the program reads. For example, if the program is called with 3 files, the output CSV file must contain 3 lines. A line in the CSV file differs from that of the input. A CSV line contains the general data of an order followed by the number of items in that order and the sum of the item price. See list 2 for an example and list 3 for a description of the fields. List 4 contains a number of example calls to the program.

[1] https://visualstudio.microsoft.com/vs/community
[2] https://docs.microsoft.com/en-us/dotnet/core/tools

```
FORM: program name [OPTIONS] FILE

OPTIONS
- d        Name of an existing directory to search for .json files. May occur at
           most 1 time. Mutually exclusive with -f.


- f        Name of an input file. May occur multiple times. Mutually
           exclusive with -d.

FILE
           Name of the output file.
```

Listing 1: Program options

```
1;Order 123;123;2;31.6;
```

Listing 2: Output CSV example

```
OrderId;Description;CustomerId;ProductCount;TotalPrice;
```

Listing 3: Output CSV field description

```
$program -d"/path/to/folder"output.csv
$ program -f file1.json -f file2.json output.csv
```

Listing 4: Example calling program

```
{
    "OrderId": 1,
    "Description":          "Order    123",
    "customer ID":       123,
    "products": [
        {
            "ProductId":        "PID1",
            "Description":          "pid1    description",
            "Amount":     1.5,
            "price":       19.5
        },
        {
            "ProductId":        "PID2",
            "Description":          "pid2    description",
            "Amount":     2.5,
            "price":       12.1
        }
    ]
}
```

Listing 5: Input JSON example

```
OrderId: Order id , numeric and greater than 0. Description: Description , text max 100
characters and optional. CustomerId: Customer number , numeric and greater than 0.

Products: List of articles, at least 1 article.
    ProductId: Item number , alphanumeric , max 50 characters Description: Item
    description , text max 200 characters. Amount: Number , decimal maximum 2 digits
    after the point. Price: Price , decimal maximum 2 digits after the point.

```

Listing 6: Input JSON field description

3 Reviews

A piece of code is supplied with this part. The supplied code is part of the elaboration of a fictitious employee to achieve a given goal. Didata is interested in the applicant's skills to evaluate the work of other software developers.

In this fictional situation, an employee has been given the task of making a link between an ERP package and an external system. Orders must be communicated from the ERP system to the external system via this link. The external system expects CSV files in a certain format[3].

The employee has decided to create a REST API that the ERP package can call. When implementing the API, the employee created a class that contains a function to export orders to the CSV files that the external system expects (RegisterOrder).This service runs constantly. This class is instantiated every time the API is requested.

The employee has just finished his work and has asked the applicant to review it. The code of this class is contained in the supplied 'review.cs' file. The following questions must be answered:

1. Explain in your own words what the code does to achieve the goal.

2. Does the code contain problems, and if so, which ones[4]?

3. If problems were found in 2., how would the applicant solve them?

The applicant is free to fill in the answers in a file format of his choice, for example PDF or plain-text. If it is a non-standard file format then it shouldn't be too much trouble for Didata to open it. Answers or examples may be split into multiple files as long as there is a master file that references the other files.

4 Turn in

The elaboration must be sent by e-mail to the same address where the applicant received this assignment. An archive containing the elaboration is expected in the e-mail. The recommended archive format is zip or tar.gz. Other formats are also allowed, but it shouldn't be too much trouble for Didata to unpack them. Please note that zip files containing executable files can be blocked by the e-mail filter. Make sure no build folders are included. The archive must contain a root folder called 'assignment'. It should contain two folders called 'implementation' and 'review'. In 'implementation' should be the git repository created in section 2. 'Review' must contain the files containing the answers to the questions from section 3.

[3] This is not the CSV format of the implementation part.

[4] Tip: think especially about the context: multiple objects of this class can be created turn into.