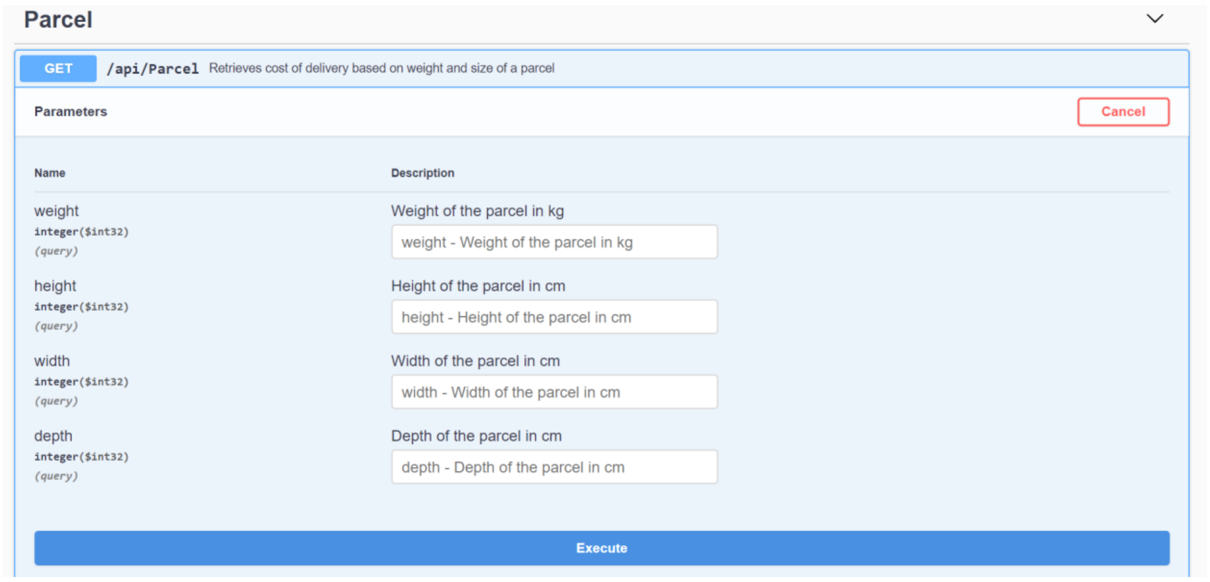
Running the Application

1. Download and install Node.js and VS Code. Alternatively, Visual Studio 2017 can be used.
2. Open *PostalService.Api* folder in VS Code terminal and run –

   ```
   dotnet run
   ```

3. Open http://localhost:3000 in browser and you should see swagger site with following end point –



Parcel

GET `/api/Parcel` Retrieves cost of delivery based on weight and size of a parcel

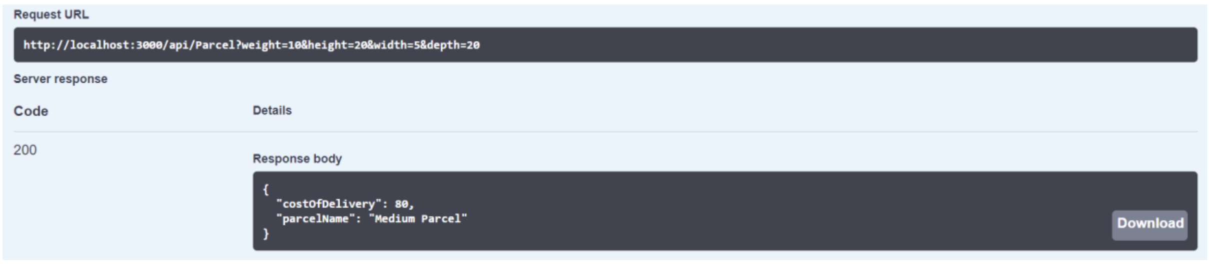
Parameters

Cancel

Name	Description
weight integer($int32) *(query)*	Weight of the parcel in kg weight - Weight of the parcel in kg
height integer($int32) *(query)*	Height of the parcel in cm height - Height of the parcel in cm
width integer($int32) *(query)*	Width of the parcel in cm width - Width of the parcel in cm
depth integer($int32) *(query)*	Depth of the parcel in cm depth - Depth of the parcel in cm

Execute

4. Enter the parameters and hit *Execute.* The Json output can be seen in the results –



Request URL

```
http://localhost:3000/api/Parcel?weight=10&height=20&width=5&depth=20
```

Server response

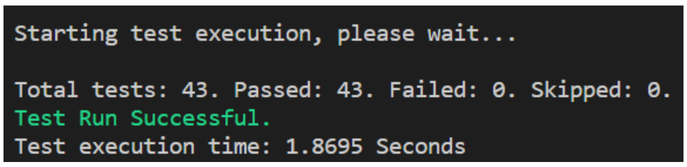
Code	Details
200	Response body

```
{
  "costOfDelivery": 80,
  "parcelName": "Medium Parcel"
}
```

Download

5. To run the unit tests, open *PostalService.Test* folder in VS Code terminal and run following command –

   ```
   dotnet test
   ```



```
Starting test execution, please wait...

Total tests: 43. Passed: 43. Failed: 0. Skipped: 0.
Test Run Successful.
Test execution time: 1.8695 Seconds
```

6. Visual Studio 2017 users open the *PostalService.Api.sln* file and build the solution the run the project by hitting F5.

Problem Statement

Calculate cost of delivery, given the dimensions and weight of a parcel based on following rules –

Priority	Rule	Condition	Cost
1	Reject	Weight > 50kg	N/A
2	Heavy	Weight > 10kg	$15 X Weight
3	Small	Volume < 1500	$0.05 X Volume
4	Medium	Volume < 2500	$0.04 X Volume
5	Large		$0.03 X Volume

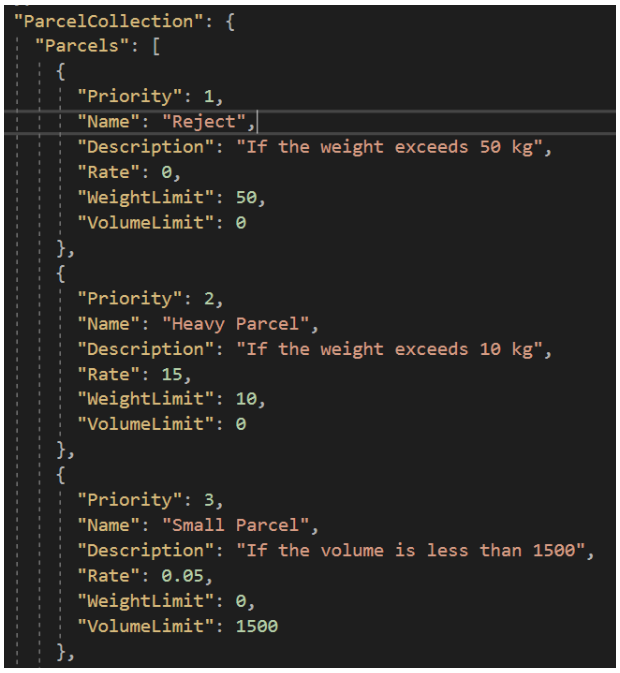
Solution

Chain of Responsibility

A .NET Core API to calculate postal cost of a parcel using ***Chain of Responsibility*** design pattern.

Adding a New Rule

Adding or inserting a new rule is easy. Just put the new settings in appsettings.json.



```
"ParcelCollection": {
  "Parcels": [
    {
      "Priority": 1,
      "Name": "Reject",
      "Description": "If the weight exceeds 50 kg",
      "Rate": 0,
      "WeightLimit": 50,
      "VolumeLimit": 0
    },
    {
      "Priority": 2,
      "Name": "Heavy Parcel",
      "Description": "If the weight exceeds 10 kg",
      "Rate": 15,
      "WeightLimit": 10,
      "VolumeLimit": 0
    },
    {
      "Priority": 3,
      "Name": "Small Parcel",
      "Description": "If the volume is less than 1500",
      "Rate": 0.05,
      "WeightLimit": 0,
      "VolumeLimit": 1500
    },
```

S.O.L.I.D Principles

The software design is lucid, extensible and maintainable by adhering to S.O.L.I.D principles.

Unit and Integration Tests

The xUnit (with Moq) tests are written to make the software robust. Include the test cases in CI/CD pipeline.

Support or Contact

Having any trouble? Please reach out @email to sort it out.

Keep Coding :-)