

## Running the Application

1. Download and install .NET Core 2.1 framework
2. Open *PostalService.Api.sln* in VS 2017 from the root folder
3. Hit F5 from within Visual Studio
4. A swagger site should be open at <http://localhost:3000> in the browser

**Parcel**

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

Parameters

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

5. 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	<p>Response body</p> <pre>{   "costOfDelivery": 80,   "parcelName": "Medium Parcel" }</pre> <p>Download</p>

## 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 :-)