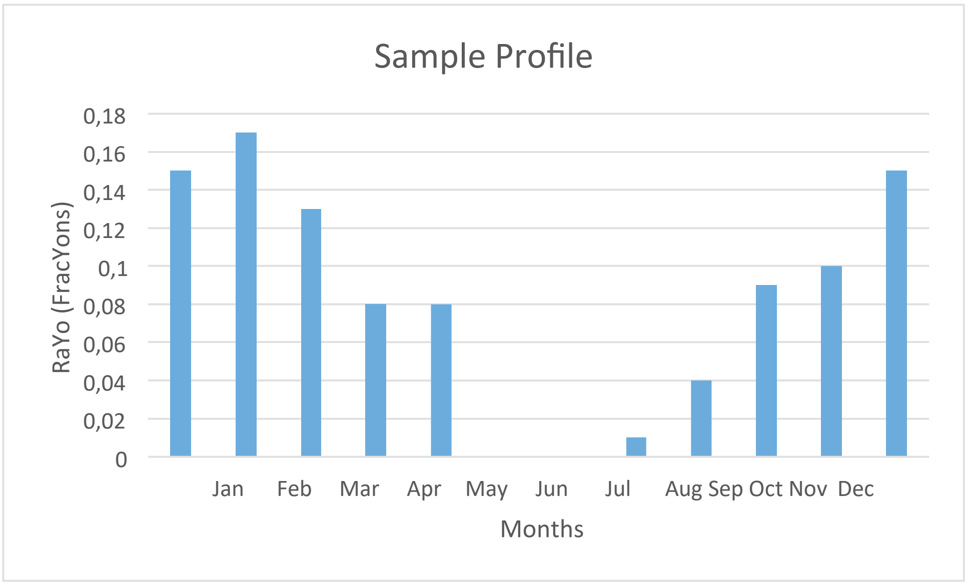
Description and Requirements

We want to build a **new application that will process metering data coming from customers.**
This new application should be able to validate and store these meter readings according to some rules described below. Also, this application will provide consumptions for a given period of time so we can connect other systems to generate invoices for the customers.

Some useful functional concepts

Note: These concepts have been simplified for the sake of this exercise.

- **Meter**: is a device that is measuring the amount of gas or electricity being used within a house. It's just a counter, so it will be a number that is increasing along time.
- **Meter reading**: It's the number that the Meter is showing at a specific date and time.
- **Consumption**: It's the difference, between two given date/times, of meter readings. Example: If the meter reading on 2016/01/15 is 120 and the meter reading on 2016/02/15 is 150, then the consumption between 2016/01/15 and 2016/02/15 is 30. Could be KWh, m3, but the unit of measure is not relevant for this exercise.
- **Profile**: It's a collection of ratios [0..1], called *Fractions*, for every calendar month. It represents how the consumption of a given year is distributed among months, so all the fractions for the 12 months must sum up 1. For example, for a house in the Netherlands it would be normal to have higher ratios during winter than during summer, given that the energy consumed will be higher because of heating.



Legacy data format

Right now there is another application running for this same purpose, the new application will replace this one as soon as it's done. Your requirements of data inputs for the new application are defined below, and are based on the legacy data formats.

1. Profiles and Fractions in CSV

```
Month,Profile,Fraction
JAN,A,0.2
JAN,B,0.18
FEB,A,0.1
FEB,B,0.21
…
```

Assumptions:

- There is no specific order for the rows.
- There is only one value per Month-Profile combination.
- Input data always contains ratios for the twelve months for a given profile (assume no missing data).

Validations that need to be done

- For a given Profile (A or B in the example) the sum of all fractions should be 1. It represents the ratio of consumption that is expected each month. If input data is not fulfilling this condition then an error should be raised.

2. Meter Readings in CSV

```
MeterID,Profile,Month,Meter              reading
0001,A,JAN,10
0004,B,JAN,8
0004,B,FEB,10
0001,A,FEB,12
0001,A,MAR,18
…
```

Assumptions:

- There is no specific order for the rows.
- Assume data always contain 12 lines per meter, one line per month (no missing data).
- The MeterID is unique.
- Every first day of the year, meters are reset to zero.

Validations that need to be done

- For a given meter a reading for a month should not be lower than the previous one. Example of an invalid case:
- `0001,A,JAN,10`
- `0001,A,FEB,9`
- Fractions for the profiles contained in the data should exist. If profiles A and B are mentioned, there should be data in the database for profiles A and B.
- Consumption for a month, being that the difference between the meter reading of a month minus the previous one, should be consistent with the fraction with a tolerance of a 25%. That means that if the total consumption for a year is for example 240, if Fraction for February is 0.2 (meaning 20% of the consumption expected), then allowed values for Consumption in February are 36-60. Note that these allowed values are for Consumption, not for Meter readings.
- If there is a validation error for a given meter data, only the data for that meter is being rejected.

Tasks to be accomplished for this Recruitment Case

1. The new service needs a **REST API**. You should design it in a way that allows the same information being sent to the application, but of course you can modify the format since the legacy one has many inefficiencies. Provide documentation for this REST API in any format you prefer.
2. Create a **Spring Boot application that implements the REST API** (CRUD operations for Fractions and Meter Readings).
   a. Implement also the **validations** described above.
   b. Make a REST interface to **retrieve Consumption** for a given month (calendar month) for a given **Meter**.
   c. Application should have a **reasonable code coverage** by Unit Tests.

We'll need the source code and a Maven or Gradle file to build it. Please provide a README using **markdown** format with instructions about how to perform any further set up your code might need.

If the solution needs a database (which is not an in-memory database) or a third-party tool (for example Redis for caching), then those have to be provided using a docker-compose file, also it has to be documented correctly (for example how to setup credentials if needed, how to connect to the database and so forth). Data for the database also has to be provided with the solution (either through the solution itself in an automated manner or part of the container with init scripts)
Solutions not providing the needed facilities for the application to run correctly and to be tested will not be taken into consideration and will be discarded!

We value not only a good understanding of requirements (business logic) but also good design & code practices when implementing the application.

Note that you are allowed to ask questions or clarifications in order to improve your understanding of the requirements. Please write these questions to the recruiter or point of contact and the answers will be provided in the shortest available period.

Bonus task

Implement extra functionality in the service that takes the old-format CSV files when they are placed in a given folder and process them as data inputs. Then this data can be used to extract consumptions. If files are processed correctly then they should be deleted, if there are errors a log file should be written in the same folder where the files are.

How to send us the Use Case

After the solution is created, please upload the code to a Bitbucket **private** repository.

Once the code is uploaded, share the repository with us. The solution should be shared using the following email address: recruitment@theitsolutions.nl.