ScottishPower Java/Service Technical Test - Mid level

The Spec

We need an MVP completed of a RESTful web service to retrieve SMART meter read details from a new database that will allow us to display SmartReads to our customers on both the mobile apps and website. Bad news is the smart meter read table structure hasn't been agreed or built yet and we need to start testing this functionality as soon as possible as it is a blocker for the front end team.

Requirements

1. Use the version of Java that you have the most experience with (Should be Java 8+).
2. Build the service using an in-memory database.
3. Modern Java frameworks should be used.
4. Automated tests should be included that you feel are suitable for this.
5. Include a ReadMe with any relevant information.
6. The test should be submitted via an online source control system of your choice.
7. The agreed contract with the FrontEnd clients is as follows:

```
Request:
GET: http://localhost:8081/api/smart/reads/{ACCOUNTNUMBER}

Response:
{
  accountId: Number,
  gasReadings: [
    {
      id: Number,
      meterId: Number,
      reading: Number,
      date: Date
    }
  ],
  elecReadings: [
    {
      id: Number,
      meterId: Number,
      reading: Number,
      date: Date
    }
  ]
}
```

Constraints

It is up to you how the implement the test, we are interested in how you would approach this challenge and what you consider required for a (as close to) production ready MVP. The test should be time bound to 3 hours, the aim of the test is to ensure you can build a REST service with best practice and standards utilising modern frameworks, clean coding practices and to demonstrate your experience and abilities.

If you run out of time then please include information on any future work that you would have included time permitting.

Optional additional tasks

Should you complete the above task and have time to spare then please feel free to attempt some of these additional requirements:

1. Include a new endpoint to allow submission of meter reads
    a. Should include validation
    b. Should include duplicate checks (based on dates of previously submitted reads)
    c. Should allow for Solus (Single fuel, single meter only)
    d. Should allow for Multi-meter (Multiple gas/elec reads in a single submission)

```
Request:
POST: http://localhost:8081/api/smart/reads

RequestBody:
{
  accountId: Number,
  gasReadings: [
    {
      meterId: Number,
      reading: Number,
      date: Date
    }
  ],
  elecReadings: [
    {
      meterId: Number,
      reading: Number,
      date: Date
    }
  ]
}
```

2. Include additional fields in the response for usage since last read and the days since the last reads

```
{
  id: Number,
  meterId: Number,
  ...
  usageSinceLastRead: Number,
  periodSinceLastRead: Number
}
```

3. Include additional fields in the response for average daily usage taking into account all reads

```
{
  id: Number,
  meterId: Number,
  ...
  avgDailyUsage: Number
}
```

4. Include additional fields in the response for comparison against other customers

```
{
  accountId: Number,
  gasReadings ...
  elecReadings ...
  ...
  gasComparison: Number,
  elecComparison: Number
}
```

Please contact us with any questions, we are looking to see how you will approach the requirements and complete the code with thoughts into testing and development practices.