The assessment consists of an assignment, to prepare beforehand, and an in-person discussion of your solution. Please develop the stories in-order, and feel free to time box yourself as long as the stories you finish are delivered in a complete and end-to-end fashion.

The assignment is described below. Read the case carefully, and approach it as you would with a regular project. Consider aspects such as robustness, maintainability, testing and user experience.
The assignment can be implemented in any language but we would prefer to see Scala code :). If you decide to use Scala, you should consider also using the Akka framework.

Good luck with the assignment!

API Queuing Service

Vandebron is building a brand new application composed by several APIs that communicate with the external world using a centralized queue. You are tasked to build some of these APIS and, most importantly, the queue module.

AQS-1: as Vandebron, I want to expose some APIs that provide Product and Order information

You have to build 2 different APIs. Each of the APIs accepts a query-parameter ‘?q=’ that accepts multiple queries split using a comma delimiter. If the same value is present multiple times in the query, the response will only contain it once. Each of the APIs provides requests and responses in their own unique manner as shown below.
The product API

Accepts a 9-digit client number and returns a set of products (EV, GAS or ELECTRICITY) equivalent to the last digit. The clients with fewer than 3 products should have non repeated random ones, clients with three products will have the three of them and clients with more than three should have the three of them and the rest filled with random repeated products.

For example: client number 109347263 will return a set of 3 different products. E.g.:

GET http:///<domain>/product?q=109347263,123456891
200 OK content-type: application/json

```
{
  “109347263”: [“EV”, “GAS”, “ELECTRICITY”],
  “123456891”: [“GAS”]
 }
```

The order API

Accepts a 9-digit client number and returns one of the following statuses: NEW, ORDERING, SETUP, DELIVERED

```
GET http:///<domain>/order?q=109347263,123456891
200 OK content-type: application/json

{
  “109347263”: “NEW”,
  “123456891”: “DELIVERED”
}
```

Vandebron also warns you that in the future more APIs that use same algorithms can be added so the code should be easy to adapt or reuse.

AQS-2: as Vandebron, I want to be able to query all APIs asynchronously in order to provide information to our users

As Vandebron I want to have a centralized module I can use to forward external call to our APIs (the ones previously built). This module must accept a sequence of api calls and forward this sequence to the correct api endpoint.
A sequence can contain calls to different APIs (example: a sequence of 5 calls can contain 3 call to the product api endpoint and 2 to the order api endpoint)
Only upon receiving all responses should the complete set of responses be returned to the caller.
Deliverables for this story are the service that can forward requests to the individual APIs.

AQS-3: as Vandebron, I want service calls to be throttled and bulked into 1 request per respective API to prevent services from being overloaded

To prevent overloading the APIs with query calls we would like to queue calls per API endpoint. All incoming requests for each individual API should be kept in a queue and be forwarded to the APIs as soon as a cap of 5 calls for an individual API is reached using a single call.

As an example: if there is a caller querying each single API and the queue of the Product API is at 4 before accepting the request. The request would cap the Product queue and thus return the response before the other API queries have been fulfilled.

AQS-4: as Vandebron, I want service calls to be scheduled periodically even if the queue is not full to prevent overly-long response times

Our current implementation has 1 major downside; the caller will not receive a response to its requests before the queue cap for a specific service is reached. To solve this, we want the service queues to also be sent out every 5 seconds.