Dear engineer,

we were quite happy with the outcome of the interview and therefore we would like to invite you for the next step in our recruitment process: our technical assessment.

Background

Auto1.com is driven by service oriented architecture. With a few exceptions, all of these services are restful and serve their content via HTTP.
We're applying more and more black box testing to those services, using rest assured or cucumber.

We hereby ask you to test a story that involves talking to three different resources that require different sets of input. The responses are used to fill three dropdowns, each dependent on the input of the one before.

The Story is: As a customer, I want to select the type of car I want to sell, so that I can receive an offer from auto1.com.

The UI usage for those resources can be seen in the above picture.

Resources

Similar resources are used in our production environments, and as so many pieces of production infrastructure all over the world, they are far from perfect. You will find *puzzling* things when testing them. The resources with their possible parameters are:

- GET /v1/car-types/manufacturer
  - locale (*accepts any locale)*
  - wa_key
- GET /v1/car-types/main-types
  - manufacturer - *(accepts manufacturer codes)*
  - locale - (*accepts any locale)*
  - wa_key
- GET /v1/car-types/built-dates
  - manufacturer - *(accepts manufacturer codes)*
  - main-type - (*accepts main-type codes)*
  - locale - (*accepts any locale)*
  - wa_key

They also accept other input parameters like pagination or sorting, but those are not in the scope of this task.

Responses

All of them will result in the same wrapping json:

```
{
  "page": int,
  "pageSize": int,
  "totalPageCount": int,
  "wkda": {}
}
```

whereas `"wkda":{}` contains key value pairs, that make the value and the translated text of the dropdown options.

Example:

```
{
  "page": 0,
  "pageSize": 2147483647,
  "totalPageCount": 1,
  "wkda": {
            "key1":"value1",
            "key 2":"value 2.",
            "key 3 specialchâr":"value 3 (and details of value3)"
          }
}
```

URL and Key

As a value for `wa_key` you can use *`coding-puzzle-client-449cc9d`*
The base url for all calls is *`http://api-aws-eu-qa-1.auto1-test.com`*

Task and Requirements

Write full automated, ideally behaviour driven code, that black-box tests the above resources and how they play together.
We expect you to write a little report on your findings where you point out unusual behaviour, glitches and bugs and rate them in criticality.

- The software should be written java or at least in a jvm based technology. But better with java.
- We'd love to see cucumber or rest assured being used, but if you have something even better, be our guest and show off some skills!
- We don't ask for extreme coverage numbers and we're very curious on what you think is enough and which conclusions you draw from what you see.
- If you put your code into a publicly available repository, please make sure you keep the base url and the wa_key private.

Our review will include the code and your report. Bonus points if your findings are covered for regression analysis.

Sending back

Since the topic is pretty vast you are totally allowed to put your code to github to brag around with it. However, it is important to not publish this document or mention our name whatsoever. If you want to reference, use wrappers. Don't write "Auto1", write "a company". It also goes for the URLs and endpoints :)

If you don't want to publish your code, it is 100% acceptable to just .zip it and send it back to us.

Have fun!