Practical Exercise

Background

We expect you to create tests following **BDD** and **Cucumber** (see Introduction - Cucumber Documentation for background). For test 1 create a feature file (no implementation); for tests 2 to 5, create a feature file and the corresponding step definitions in Java.

To implements the tests you can use any code language (preferred Java with maven/gradle). Use standard libraries (e.g RestAssured, Unirest for Java) where possible. The credentials are the following:

- Username: `qa_interview@skycell.ch`
- Password: `Qa_interview2023!`
- APIKEY: `NNSXS.RPNRQUVEAQHYIBRJPYB5BMF36VT2E4ZIQWLCO6Y.ZP7FKSYX6J2XO2SRNBPHWQJHIBB5ZWTULHPI27N7C4IMQAKB6QYA`

There are no constraints regarding the number or type of tests to create. Please try to concentrate your effort on writing **clean/reusable code** and **meaningful tests.**

Delivery & Discussion

Please delivery your solution in a GitHub repository or in a zip file. **Test 1, 2 and 3 are mandory**, Test 4 is recommended and Test 5 is a plus. The solution needs to be executable, and all tests need to pass when executed.
Provide a Today I Learned for programmers file in the root folder with the steps to run the tests locally. The solution will be discussed step by step during the interview with **screen sharing**:

- One single execution of all the tests will be done
- Overview and explanation of the feature files
- Overview and explanation of code

Test 1 | Difficulty = Easy

Specification

We have an IoT project where the customer uses SkyCell's LoRa-based loggers to track their asset fleet of about 30'000 assets across 1'000 locations. The system has the following components:

- Cloud platform (allows the customer to track all his assets and loggers via a web app)
- IoT service (ingests and stores sensor data and provides the cloud platform with IoT data)
- Gateways (connect loggers via LoRa and pushes sensor data to the IoT service)
- Loggers (measure data and send it to the gateway)

The battery of the loggers sometimes need to be replaced. The customer wants to do this replacement in only a handful of the 1'000 locations. In addition, he wants to be able to dynamically select in the cloud platform which locations of the 1'000 locations are used for replacements, and wants to dynamically set the battery level threshold.

In addition, the customer requests to also have an LED outside the logger which should blink if the battery level is below the battery threshold (but to save battery the LED should blink only in the locations where battery exchanges happen).

Goal

Write a BDD-style test which covers what you think is the most important scenario from a testing perspective. During the interview, explain why you've this scenario, and which others you have considered, but not taken.

Test 2 | Difficulty = Easy

Specification

Our server offer an authentication system. To authenticate you need a username and a password and do an API request to the server.

Specification Request

```
"URL" = https://keycloak.dev.skycell.ch/realms/skycell/protocol/openid-connect/token
"Content-Type" = "application/x-www-form-urlencoded"
"Method" = "POST"

"Form Parameter"
client_id:webapp
grant_type:password
username:{{username}}
password:{{password}}
```

Java Example Request

```java
HttpResponse<String> response = Unirest
    .post("https://keycloak.dev.skycell.ch/realms/skycell/protocol/openid-connect/token")
   .header("Content-Type", "application/x-www-form-urlencoded")
   .header("origin", "https://skynet.test.skycell.ch")
   .field("client_id", "webapp")
   .field("grant_type", "password")
   .field("username", username)
   .field("password", password)
   .asString();
```

JSON Response (200 OK)

```json
{
    "access_token": "{{yourJWT}}",
    "expires_in": 50400,
    "refresh_expires_in": 14400,
    "refresh_token": "{{ignore}}",
    "token_type": "Bearer",
    "not-before-policy": 1680026173,
    "session_state": "ef08f804-404e-42c4-b0ae-86e2715eeec6",
    "scope": "webapp email profile document-generator"
}
```

Goal

Write the automation tests that check the API for this authentication system with the most common scenario. Be sure to be able to extract the token and that the token is valid (you can check the validity of the token here JWT.IO).

Test 3 | Difficulty = Easy

Specification

SkyCell owns two main type of loggers. Each logger can have a type and number of sensors, and each sensor can measure one or more data type.

Logger Type	Sensor Code	Data Type
MR_810T	`INTEGRATED`	`TEMPERATURE`

	`BATTERY_VOLTAGE`	`BATTERY_VOLTAGE`
MR_812P	`INTEGRATED`	`TEMPERATURE`
	`BATTERY_VOLTAGE`	`BATTERY_VOLTAGE`
	`CONNECTOR_1`	`TEMPERATURE`

In order to exist

Specification Request

Each logger is identified with a number. The `loggerNumber` is an HEX string of **maximum 16 characters**. In order to exist in our system the logger need to be created with an API request:

```
"URL" = https://sensor-data-ingestion.dev.skycell.ch/v1/lora/configuration
"Content-Type" = "application/json"
"Method" = "POST"
```

Headers

```
APIKEY:{{APIKEY}}
```

Body

```
{
    "loggerNumber": "{{yourLoggerNumber}}",
    "loggerType": "MR_810T",
    "baseInterval": 600
}
```

Java Code Example

```
HttpResponse<String> response = Unirest
  .post("https://sensor-data-ingestion.dev.skycell.ch/v1/lora/configuration")
  .header("APIKEY", "{{APIKEY}}")
  .header("Content-Type", "application/json")
  .body("{\r\n\"loggerNumber\":\"12345abc\",\r\n\"loggerType\":\"MR_812\",\r\n\"baseInterval\": 600\r\n}")
  .asString();
```

Goal

- Create one logger for each logger type.
- Write the automation tests that check the API creation of a logger in the system. Please consider that only two logger types are allowed and verify that also the other constraints are respected. Ensure that tests can be executed multiple times without having data conflicts.

Test 4 | Difficulty = Medium

Specification

Each of the previous **Sensors is associated** with a numeric type:

Sensor	Type
`INTEGRATED`	`0`
`BATTERY_VOLTAGE`	`1`
`CONNECTOR_1`	`3`

Each logger sends data to our server via the following API request.

Specification Request

```
"URL" = https://sensor-data-ingestion.dev.skycell.ch/v1/lora/uplink
"Content-Type" = "application/json"
"Method" = "POST"
"Headers"
APIKEY:{{APIKEY}}
```

Body

```
{
  "end_device_ids": {
    "device_id": "eui-{{yourLoggerNumber}}",
    "dev_eui": "{{yourLoggerNumber}}"
  },
  "received_at": "2023-11-22T18:12:46",
  "uplink_message": {
    "decoded_payload": {
      "sensorData": [
        {
          "amount": 1,
          "dataPoints": [
            {
              "index": 10,
              "temperature": 23.1
            }
          ],
          "isLogged": 0,
          "type": 0,
          "typeText": "Internal Temperature Sensor"
        },
        {
          "amount": 1,
          "dataPoints": [
            {
              "index": 20,
              "voltage": 1.3
            }
          ],
          "isLogged": 0,
          "type": 1,
          "typeText": "Battery voltage"
        },
        {
          "amount": 1,
          "dataPoints": [
            {
              "index": 4,
              "temperature": 23.1
            }
          ],
          "isLogged": 0,
          "type": 3,
          "typeText": "RTD Temperature Sensor"
        }
      ]
    },
    "rx_metadata": [
      {
        "gateway_ids": {
          "gateway_id": "9c65f94334d8",
          "eui": "9C65F9FFFE4334D8"
        },
        "timestamp": 39945773,
        "rssi": -44,
        "channel_rssi": -44,
        "snr": 5
      }
    ]
  }
}
```

Java Code Example

```
HttpResponse<String> response = Unirest
  .post("https://sensor-data-ingestion.dev.skycell.ch/v1/lora/uplink")
  .header("Content-Type", "application/json")
  .header("APIKEY", "{{APIKEY}}")
  .body({{yourbody}})
  .asString();
```

Field Explanation

Field	Explanation
`device_id`, `eui_id`	Logger number
`received_at`	The time the data is measured
`type`	The sensor that measured the data
`index`	A counter increased every time a new data point is collected in a sensor. Each sensor has its own index counter
`temperature`, `batteryVoltage`	Data value

If everything goes well you, should receive 201 as a response code. To verify the correct insertion of the data you will need to read the data back from the server with an API request

Specification Request

```
"URL" = https://sensor-data-ingestion.dev.skycell.ch/v1/lora/uplink
"Content-Type" = "application/json"
"Method" = "POST"
"Headers"
Authorization: Bearer {{yourJwtToken}}
```

Body for TEMPERATURE

```
{
   "useStandardizedTimestamps": false,
    "from": "2023-11-17T12:00",
    "to": "2023-12-17T12:00",
    "loggers": [
        {
            "loggerNumber": "{{yourLoggerNumber}}",
            "loggerType": "MR_810T"
        }
    ],
    "dataTypes": [
        "TEMPERATURE"
    ],
    "sensorCodes": [
        "INTEGRATED"
    ]
}
```

JSON Reponse

```
[
    {
        "loggerNumber": "{{yourLoggerNumber}}",
        "messageId": {{index}},
        "loggerType": "MR_810T",
        "dataType": "TEMPERATURE",
        "dataValue": 22.0,
        "sensorCode": "INTEGRATED",
        "measuredTime": "2023-11-22T13:20:00",
        "insertTime": "2023-11-22T13:22:08.241",
        "measurements": {
            "TEMPERATURE": 22.0
        }
    }
]
```

Body for BATTERY_VOLTAGE

```
{
    "useStandardizedTimestamps": false,
    "from": "2023-11-17T12:00",
    "to": "2023-12-17T12:00",
    "loggers": [
        {
            "loggerNumber": "{{yourLoggerNumber}}",
            "loggerType": "MR_810T"
        }
    ],
    "dataTypes": [
        "BATTERY_VOLTAGE"
    ],
    "sensorCodes": [
        "BATTERY_VOLTAGE"
    ]
}
```

JSON Reponse

```
[
    {
        "loggerNumber": "{{yourLoggerNumber}}",
        "messageId": {{index}},
        "loggerType": "MR_810T",
        "dataType": "BATTERY_VOLTAGE",
        "dataValue": 1.3,
        "sensorCode": "INTEGRATED",
        "measuredTime": "2023-11-22T13:20:00",
        "insertTime": "2023-11-22T13:22:08.241",
        "measurements": {
            "BATTERY_VOLTAGE": 1.3
        }
    }
]
```

Goal

For each of the previously created loggers do the following:

- Insert one data point per each sensor type
- Insert one data point every 10 minutes for one hour
- Create the tests necessary to verify the correct creation of the data. Be sure to be able to repeat the test multiple times without having data conflicts.

Test 5 | Difficulty = Hard

- Make your tests executable with Cucumbers tools. For java and maven you can check
  - Getting started with Cucumber in Java — A 10 minute tutorial
  - REST API testing with Cucumber | Baeldung
- Implement a first draft for a CI/CD pipeline. The action will need to run the tests automatically every night at 1am CET time.
  - Upload your code in a github repository and add github action (github offer up to 2000 minutes for a free account https://github.com/features/actions Can't find link) Also other tools are fine (Bitbucket, CircleCI, TeamCity, Jenkins).
  - Use in your pipeline secrets and env variable to run the job when is needed (e.g SERVER URL or APIKEY)
- Write Openapi (in YAML) or JsonSchema specification for DTO used in the logger data INSERTION
- Generate the DTO code from the previously created specification
  - For java and maven you can take a look at the https://github.com/OpenAPITools/openapi-generator/tree/master/modules/openapi-generator-maven-plugin Can't find link