Introduction

The purpose of this app is to understand how you would approach and solve a typical business problem.

The App

Commsoft wants to create a service which would be responsible for managing aircraft and its components. The maintenance manager said that he wants to:

- Create new aircraft, update existing ones.
- Create a new component.
- Fit or remove component from an aircraft.
- Get all fitted components for the specific aircraft.
- Log a flight for a specific aircraft. All fitted components should be tracking their time individually. This is because the components could be swapped between aircrafts.
- Disable or enable a specific aircraft. No new components can be fitted or new flights logged against a disabled aircraft.
- Query how many minutes a component spent in the air.

Currently we are using Java/Spring Boot but this doesn’t mean that you have to use it as well. If you decide to implement this service in a different language/stack – you can do so! Just make sure that the instructions of how to launch your app are included and easy to understand.

The data should be stored in a SQL database. The actual flavour is up to you, it could be H2 in-memory database, MySQL, PostgreSQL and so on.

Definition of endpoints:

URI	Method	Consumes	Returns
/aircrafts	POST	aircraft model	201 CREATED, Aircraft ID
/aircrafts/{id}	GET	N/A	200 OK, Aircraft model
/aircrafts/{id}	PUT	aircraft model	200 OK
/flights	POST	flight model	201 CREATED, Flight ID
/flights/{id}	GET	N/A	200 OK, Flight model
/flights/{id}	PUT	flight model	200 OK
/flights/aircraft/{id}	GET	N/A	200 OK, List of flights.
/components	POST	component model	201 CREATED, Component ID
/components/{id}	GET	N/A	200 OK, Component model
/components/{id}	PUT	component model	200 OK
/component/{id}/aircraft/{id}	PUT	N/A	200 OK
/component/{id}/aircraft/{id}	DELETE	N/A	200 OK
/components/aircraft/{id}	GET	N/A	200 OK , List of components.

Definition of models:

Model	JSON Object
aircraft	{"id":1, "registrationCode":"U-SKA", enabled: true, "notes":"Notes"}
flight	{"id":1, "registrationCode":"U-SKA", "takeoff":"2019-01-01T00:00:00.000Z", "landing":"2019-01-01T01:00:00.000Z"}
component	{"id":1, "partNumber":"CQ01-A", "lastFitmentTime":"2019-01-01T00:00:00.000Z", "notes":"Notes"}

Example communication

A possible usage of the service we are building could look something like this:

Creating new aircraft:

```
POST localhost:8080/aircrafts HTTP/1.1
Content-Type: application/json; charset=utf-8
Content-Length: 63
{"id":null,"registrationCode":"U-SKA","notes":"My favorite aircraft."}
```

Response:

```
HTTP/1.1 201 OK
Content-Type: text/html; charset=utf-8
1
```

Creating new component:

```
POST localhost:8080/components HTTP/1.1
Content-Type: application/json; charset=utf-8
Content-Length: 63
{"id":null,"partNumber":"CQ-01A","notes":"Important component."}
```

Response:

```
HTTP/1.1 201 OK
Content-Type: text/html; charset=utf-8
5
```

Fit recently created component to an aircraft:

```
POST localhost:8080/component/5/aircraft/1 HTTP/1.1
```

Response:

```
HTTP/1.1 200 OK
```

Delivery

You can submit your code in a ZIP archive with instructions how to launch your program, although we would prefer link to a GitHub or BitBucket repository.