Bring Your Own Architecture - Requirements

Welcome to the challenge track of the Hands-On ERFA. This track is meant to give you a challenge (hence the name) that you solve by yourselves or in teams, with little to no help from the coaches. Of course we are available in case of questions, but we will not help you solve the challenge.

General Goal

The general goal of this challenge is to give you an opportunity to learn from a greenfield project setup, and have an opportunity to get challenged on your architecture choices. While the focus of this track is on architecture, we of course want to do some hands-on coding as well. The idea is that you get a feeling for some consequences of your architecture decisions early on, learn from them and potentially correct your architecture adequately.

In this document we list a set of requirements for a small stateful REST API. Building a service for this API will be your main goal of this track. You are free to use whatever technologies, programming languages, cloud providers, and organisational methods you want. It is however important that you plan realistically, such that the application can be finished and deployed in about 5h within your team. We therefore recommend using technologies that you are already relatively familiar with, as you will likely not finish your work if you need to complete tutorials as well.

We do not expect it to be perfect, but we will evaluate the application as if it were a finished product. We will only share the evaluation criteria with you at the very end of the day, however we can already tell you that it will focus on the architecture you chose. While the code you write will have an impact on, for instance, how maintainable the application is, we will not focus on the code itself.

Requirements

We are building a text search service. The general idea is that this service provides a way to push text to it, and then check if specific terms are part of the text (here a term is a single word). The service stores the texts pushed to it to enable repeated searching for terms over the same text.

The API you are supposed to implement is very simple. It should provide the following endpoints (all endpoints only need to support ASCII text):

POST Text

This endpoint enables clients to push text to the service. A maximum payload size of 10 MiB should be supported.

Endpoint (POST)	`<base-url>/texts`
Payload	`{ "data": "<text>" }`
Response (201 Created)	`{ "id": "<uuid>" }`
Response (400 Bad Request) Invalid Payload	`{ "error": "<error>" }`
Response (5xx Server Side)	`{ "error": "<error>" }`

DELETE Text

This endpoint allows users to delete texts to clean up the service once the text is no longer required.

Endpoint (DELETE)	`<base-url>/texts/<uuid>`
Response (204 No content)	
Response (400 Bad Request) Invalid UUID	`{ "error": "<error>" }`
Response (404 Not Found) UUID does not exist	`{ "error": "<error>" }`
Response (5xx Server Side)	`{ "error": "<error>" }`

GET Text

This endpoint allows users to get texts to check what they contain.

Endpoint (GET)	`<base-url>/texts/<uuid>`

Response (200 OK)	`{ "data": "<text>" }`
Response (400 Bad Request) Invalid UUID	`{ "error": "<error>" }`
Response (404 Not Found) UUID does not exist	`{ "error": "<error>" }`
Response (5xx Server Side)	`{ "error": "<error>" }`

GET Search

This endpoint allows users to search a term in the referenced text. It only returns whether the term is a word contained in the text or not. Note that we consider a word any token that has no whitespace in it.

Endpoint (GET)	`<base-url>/texts/<uuid>/search?term=<term>`
Response (200 OK)	`{ "found": <bool> }`
Response (400 Bad Request) Invalid UUID / term	`{ "error": "<error>" }`
Response (404 Not Found) UUID does not exist	`{ "error": "<error>" }`
Response (5xx Server Side)	`{ "error": "<error>" }`

Performance Testing

To add some fun to it, you can ask us to test your API against the specification given above. We will test that it works, and how performant it is. Based on the performance you will get a

score from us (as a total of requests per second). You can try to maximise that score, as the performance will likely also be a criteria we are interested in when considering your architecture.

You can communicate to us the endpoint you want us to test in any way you want (email, chat, in person). We will then test the endpoint and provide you with your score, which you can then compare to the other team's score.

Some Useful Links

- Azure Cloud ipt: https://app.happeo.com/pages/1e1oopl952ukqf9e0h/AzureAmpDu/1e5g766dso0ms8i9mp
- AWS Cloud ipt: https://app.happeo.com/pages/1e1oopl952ukqf9e0h/AwsNutzen/1eator678s3a9va9qk
- Google Cloud ipt: https://app.happeo.com/pages/1e1oopl952ukqf9e0h/ErstellungEinesGcpKontos/1e5849afvj209dijut
- RHOS Demo environment ipt: https://app.happeo.com/pages/1e1oopl952ukqf9e0h/DemoEnvironment/1e5rjl7c4i37e6h7cr