Hitachi Energy recruitment task

Cat Facts WebApp

Following are two task: one proving your skills in front-end development and the other one proving your skills on the back-end. You are free to choose one of them, though doing both will be a huge plus.

While working on the task, do not focus on just delivering the result. What's most important to us is that you prove

1. You can quickly learn and use something at work when needed
2. You can write clean and reusable code without smells

While there are no constraints on technological stack you choose, we are working with TypeScript/ React on front-end and Kotlin/Spring Framework on back-end, so using these will be a big plus.

Good luck & have fun!

Task 1. Back-end

Create a back-end service for fetching random cat facts. Task may be done in web framework of your choosing as long as it is done using some kind of reactive API, such as RxJS from JavaScript or Kotlins' Flow API.

- Take a look at these 2 public APIs:
  - https://publicapis.io/cat-facts-api
  - https://publicapis.io/random-user-api
- Your task is to create a service that exposes endpoint `/cat-facts` that produces server-sent events `Content-Type`.
- Every 10 seconds, this service makes a request to above APIs, randomly associates random users as authors of random facts and yields data in following format

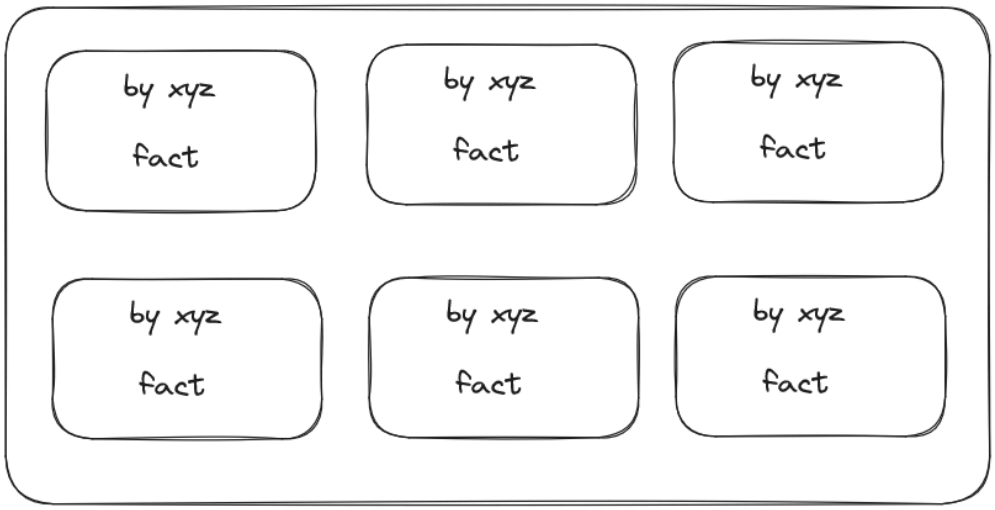
```
[
  {
    "user": string,
    "fact": string
  },
  {
    "user": string,
    "fact": string
  },
  ...
]
```

Comments

- Do not worry if you hear about server-sent events for the first time and don't know anything about them. That's the point. It is opportunity for you to prove that you can quickly learn something and apply it in a productive way.

Task 2: Front-end

1. Create a website with the following layout:



- Site presents a set of cards
- Cards are arranged in a grid
- On each card there is:
  - author of a cat fact
  - a fact about a cat
- **Grid is responsive** - this means, that number of columns of the grid is adjusted based on the current browser window width and cards do not overflow.

As a first step, implement just this view and fill it with mock data.

Comments

- Usage of HTML grid is not strictly required - exercise may be done using flexbox if preferred.
- You may surprise us with whatever styling you think of, the only requirements is that cards are arranged in responsive grid.

2. Populate grid with external data

- If you have done previous back-end task, use API you have created by subscribing to server-sent events endpoint using RxJs Observable and refreshing view each time new data arrives. Else:
- Take a look at these 2 public APIs:
  - https://publicapis.io/cat-facts-api
  - https://publicapis.io/random-user-api
- Now, your task is to combine data from these two sources to populate your grid with random cats and random users. Please solve this problem using RxJs Observables. Each 10 seconds, new data is polled from the provided APIs and, using it, view is updated.

Comments

- Don't worry if you don't know about observables and never worked with reactive streams. That's the point. This is opportunity to prove that you can quickly learn something and apply it to everyday work.