Backend development task

Introduction: Welcome to the Ever-changing Hierarchy GmbH.

Help HR manager Personia get a grasp of her ever-changing company's hierarchy! Every week Personia receives a JSON of employees and their supervisors from her demanding CEO Chris, who keeps changing his mind about how to structure his company. Personia wants a tool to help her better understand the employee hierarchy and respond to employee's queries about who their boss is. Can you help her?

Personia's initial requirements were the following:

1. I would like a pure REST API to post the JSON from Chris. This JSON represents an Employee -> Supervisor relationship that looks like this:
   ```
   {
       "Pete": "Nick",
       "Barbara": "Nick",
       "Nick": "Sophie",
       "Sophie": "Jonas"
   }
   ```
   In this case, Nick is a supervisor of Pete and Barbara, Sophie supervises Nick. The supervisor list is not always in order.
2. As a response to querying the endpoint, I would like to have a properly formatted JSON which reflects the employee hierarchy in a way, where the most senior employee is at the top of the JSON nested dictionary. For instance, previous input would result in:
   ```
   {
       "Jonas": {
           "Sophie": {
               "Nick": {
                   "Pete": {},
                   "Barbara": {}
               }
           }
       }
   }
   ```
   Sometimes Chris gives me nonsense hierarchies that contain loops or multiple roots. I would be grateful if the endpoint could handle the mistakes and tell her what went wrong. The more detailed the error messages are, the better!

3. I would really like it if the hierarchy could be stored in a relational database (e.g. SQLite) and queried to get the supervisor and the supervisor's supervisor of a given employee. I want to send the name of an employee to an endpoint, and receive the name of the supervisor and the name of the supervisor's supervisor in return.
4. I would like the API to be secure so that only I can use it. Please implement some kind of authentication.

I'm counting on you!

Signed: Personia, HR Manager

What we expect from you:

1. Write a small and simple application according to Personia's specifications, no more, no less.
2. Provide clear and easy installation instructions (think about the reviewers!). The less steps we have to do to get this running, the better. If we can't get your app to run, we won't be able to review it. Docker is your friend!
3. A set of working unit/functional tests to cover all the use cases you can think of.

Ideally **take our challenge in Kotlin or Java** (whichever you feel most comfortable with). **PHP and Ruby** are also acceptable, since we use them in our legacy applications. If you prefer another language, please reach out to us first. Remember that your solution must not require us to install any stack specific tool in order to test the result of your work, so use Docker in this case.

What we (mainly) look at when checking out the solution:

1. Did you follow the instructions, i.e. does your solution work and meet Personia's requirements? Would Personia be happy to use your solution for her work?
2. Is your solution easy to read and understand? Would we be happy to contribute to it?
3. Is your code ready for production (stable, secure, scalable)?

Final notes

- This challenge is about showing us how you think as an engineer, how you structure your code and how you solve problems, not how well you know tool X or design pattern Y.
- Try to design and implement your solution as you would do for real production code. Show us how you create robust, stable and maintainable code.
- Be prepared to demonstrate your solution with a tool like curl or Postman
- There is no right or wrong, just be ready to discuss why you choose one approach over another!