Face to Face Interview Questions - Shivan

Technical Questions

1. What would you do differently if you had a new project where you could choose the whole technology stack your self?
2. Which Spring modules have you worked with?
    a. Security, Spring Data JPA, Spring WebMVC, ...
    b. Can you explain how Spring dependency injection works?
        i. Explicit nullable definitions, Boilerplate, more expressive
        ii. Can be web application/database transactions/inter-service communication etc.
3. How is serialization realized in Java?
    a. What is hashCode and equals in Java?
4. How does garbage collection work in Java?
   Different memory allocations?
   Stack and Heap memory in Java?
5. What good/bad experiences have you made with Hibernate?
6. What's the difference between JPA and Hibernate?
7. CI/CD
    a. What kind of build steps would you define writing a pipeline?
    b. Do you have experience with Kubernetes Deployments and how it works?
8. How much experience do you have with Java Streams?
9. Kotlin Experience? What do you like about Kotlin vs Java?
10. What was the most complex multi-threading problem that you have ever worked on?
11. Please explain how Java Hibernate works.
12. What's your favorite design pattern?
13. What's your least favorite anti pattern?
14. **What is your idea of Clean Code?**
15. Do you know the Separation of Concerns principle?
16. Do you know the S.O.L.I.D. principle?
17. How much experience do you have with Git and CI/CD?

Soft Skill Questions

1. What is your definition of a good code review?
2. What if you get stuck during the sprint?
    a. Research?
    b. Ask team members?
3. How do you solve problems under time pressure?
    a. Re-thinking the Scope?
    b. MVP thinking?
    c. Communication to PO and team members?
4. What's your first approach to find solutions to a new problem on a general level?
    a. Methodology?

Closing the Interview

1. Feedback round
    a. Was it too technical or too abstract?
    a. Can you give me some feedback on the interview approach?
    b. Do you have any suggestions on how this could be improved?

Whiteboard Discussion (outdated)

1. Please can you explain in as much detail as you like how a web browser gets a generated page?
    a. How does the request reach the backend application?
    b. What kind of components are involved and how is a response generated?
    c. Describe an arbitrary REST API endpoint.
        i. How do you define the packages?
        ii. Rest Controller, Service, Repository, Dao
        iii. What kind of HTTP methods would you implement?
        i. If you need to store a "city" object, what kind of database would you use?
        ii. What do you understand under database normalization?
        i. Unit Tests
        ii. Integration Tests
        iii. End-to-end tests
        iv. What kind of classes would you test with which kind of test?
        v. What are the benefits of each kind of tests?
        vi. Experience with TDD/BDD in any project?
            1. What is your personal choice?
        i. What kind of characteristics make up a "good" API for you?
        ii. Create a draft of the classes involved in such an endpoint.
        iii. How could the underlying database model look like?
        iv. What kind of things would you test in such an environment?
    d. How many Unit/Integration/End-to-end tests would you write?
    e. What does idempotency mean in REST APIs?

a. Pros
    i. Fast and continuous improvements
    i. Versatile: different tech stacks
    ii. Continuous Delivery
    iii. Resilience
    iv. Scalability
    v. Faster time to market
b. Cons
    i. How do you slice micro-services?
    i. System integration tests are more difficult
    ii. Management of too many micro-services is difficult
    iii. Architecture is hard to do right
c. Ways of communication between micro-services?
    i. APIs, remote procedure calls, event bus

a. How could the data flow look like, starting with the HTTP request?
b. Let's dive in deeper into the backend, assuming there is Kubernetes cluster or NGINX with a Spring Boot application:
c. Describe some pros/cons of a micro-service architecture.