Senior Java + React Fullstack Developer

Active

Engineering

Deep Technical Screen

Created 4/2/2026

Interview Link

http://localhost:3000/interview/BxoIWQk9Qr4QofVO

Duration

30 min

Language

Slovak

Recording

Enabled

Interviews

54

Job Description

Job Description

We are looking for a highly skilled Senior Fullstack Developer with strong experience in Java (Spring

Boot) and React (Next.js) to join our growing engineering team.

You will be responsible for designing, developing, and maintaining scalable web applications, collaborating closely with product managers, designers, and other engineers to deliver high-quality solutions.

Key Responsibilities
Develop and maintain backend services using Java (Spring Boot)
Build responsive frontend applications using React.js and Next.js
Design and implement RESTful APIs and microservices
Optimize application performance and scalability
Collaborate with cross-functional teams to define and deliver features
Write clean, maintainable, and well-tested code
Participate in code reviews and architectural decisions
Requirements
4+ years of experience in fullstack development
Strong proficiency in Java and Spring Boot
Strong experience with React.js and modern JavaScript (ES6+)
Experience with Next.js is a strong advantage
Familiarity with REST APIs, microservices architecture
Experience with databases (MongoDB, PostgreSQL, or similar)
Understanding of CI/CD pipelines and cloud platforms
Strong problem-solving and communication skills

Normalized Role Brief (used by AI)

We're hiring a Senior Frontend Engineer to lead our React/TypeScript codebase, mentor junior developers, and drive performance optimization. The ideal candidate has 5+ years of experience building complex SPAs and strong opinions on component architecture.

Skills

Required

React.js | JavaScript (ES6+) | REST APIs | Git

Preferred

Next.js | TypeScript | Docker | Kubernetes | MongoDB | AWS / GCP | Microservices architecture

Must-Have Competencies

System Design Thinking — Advanced

Can break down complex UI requirements into component hierarchies and data flow patterns

Technical Leadership Intermediate

Can mentor junior developers, conduct code reviews, and drive architectural decisions

Knockout Criteria

Framework Familiarity

Must know React specifically

Fail if: Candidate has never worked with React or similar component-based frameworks

Availability

This is urgent hiring where candidates must be available within a month

Fail if: Candidate cannot start within the 4 weeks

Minimum Java Experience

Role requires substantial Java experience

Fail if: Candidate has less than 5 years of professional Java development experience

Custom Interview Questions

Q1. Can you explain design pattern called Dependecny Injection?

Q2. What is your experience with Microservices SAGA Design Pattern?

Q3. Describe challenging technical issue and how did you solve it?

Question Blueprints

B1. What are the main differences between Spring Boot and traditional Spring?

6 topics | 6 follow-ups

Knowledge Areas

- Spring Boot's opinionated defaults compared to traditional Spring
- Configuration differences between Spring Boot and traditional Spring
- Spring Boot's embedded server capabilities
- Dependency management in Spring Boot vs traditional Spring
- Spring Boot's Actuator and monitoring features
- Auto-configuration

Pre-Written Follow-Ups

1. Can you explain how Spring Boot's opinionated defaults simplify the setup process compared to traditional Spring?
   - Spring Boot's opinionated defaults compared to traditional Spring
2. What are the key differences in configuration approaches between Spring Boot and traditional Spring?
   - Configuration differences between Spring Boot and traditional Spring
3. How does Spring Boot's embedded server feature change the deployment process compared to traditional Spring applications?
   - Spring Boot's embedded server capabilities
4. Can you discuss the differences in how dependency management is handled in Spring Boot compared to traditional Spring?
   - Dependency management in Spring Boot vs traditional Spring
5. What role does Spring Boot's Actuator play in application monitoring compared to traditional Spring applications?
   - Spring Boot's Actuator and monitoring features
6. Can you tell me more about your experience with Auto-configuration?
   - Auto-configuration

Strong Indicators

+ Clearly articulates the benefits of opinionated defaults in Spring Boot
+ Provides specific examples of configuration differences and their implications
+ Explains deployment advantages of embedded servers in Spring Boot
+ Describes how dependency management is streamlined in Spring Boot
+ Discusses monitoring capabilities offered by Spring Boot Actuator in detail

Weak Indicators

- Lists differences without explanation or context
- Fails to identify key configuration distinctions or their relevance
- Overlooks embedded server features entirely
- Does not differentiate between dependency management practices
- Ignores the role of monitoring features in application maintenance

B2. What is the volatile keyword? How and why would you use it?

5 topics | 5 follow-ups

Knowledge Areas

- Definition of the volatile keyword
- Use cases for the volatile keyword
- Implications of using volatile (e.g., visibility guarantees)
- Differences between volatile and other concurrency controls (e.g., synchronized)
- Impact on performance when using volatile

Pre-Written Follow-Ups

1. Can you explain in your own words what the volatile keyword means and why it is important in concurrency?
   Definition of the volatile keyword
2. What are some specific scenarios where you would choose to use the volatile keyword over other synchronization mechanisms?
   Use cases for the volatile keyword
3. How does using the volatile keyword affect the visibility of variables in a multi-threaded environment?
   Implications of using volatile (e.g., visibility guarantees)
4. Can you compare the volatile keyword with synchronized blocks and other concurrency controls in Java?
   Differences between volatile and other concurrency controls (e.g., synchronized)
5. What are the potential performance implications of using the volatile keyword in a Java application?
   Impact on performance when using volatile

Strong Indicators

+ Provides a clear definition of the volatile keyword
+ Identifies appropriate scenarios for utilizing volatile in Java applications
+ Explains visibility guarantees provided by the volatile keyword
+ Makes clear comparisons with synchronized and other concurrency mechanisms
+ Discusses performance considerations confidently

Weak Indicators

- Gives an unclear or incorrect definition of volatile
- Fails to provide relevant use cases or examples
- Neglects to explain visibility implications
- Confuses volatile with other concurrency controls without clear distinctions
- Does not address performance impacts at all or incorrectly states them

AI System Prompt Preview What the AI sees

Interviews

Candidate	Score	Recommendation	Status	Date
M Marcel Mrkvička marcel@mailinator.com	72/100	Yes	Completed	4/2/2026
A anrej miskovy andrej@mailinator.com	61/100	Yes	Completed	4/2/2026
M misko maly misko@mailinator.com	48/100	Yes	Completed	4/3/2026
I Ivan Vlcek pras3xer@gmail.com	7/100	Yes	Completed	4/8/2026