EXERCÍCIOS DO MÓDULO 08:
PRINCÍPIOS DE DESIGN DE MICROSSERVIÇOS - OS 12 FATORES

Samuel Ferreira Machado[1]

EXERCÍCIO 1 - OS 12 FATORES

Escolha três dos 12 fatores que estudamos em aula e explique, com suas palavras, por que eles são importantes para o desenvolvimento de microsserviços.

Base de Código – Cada microsserviço deve ter sua própria base de código, controlada por um sistema de versionamento. Isso é importante porque evita o compartilhamento de código entre os serviços, permitindo que cada um evolua independentemente. Além disso, ter uma base de código separada facilita o entendimento do projeto, potencializa a colaboração da equipe, melhora a manutenção do código e facilita a integração e a implantação contínuas.

Dependências – O uso de um gerenciador de dependências traz benefícios significativos para o desenvolvimento de microsserviços. Ao especificar explicitamente as dependências de cada serviço, incluindo a versão e os repositórios, garante-se que todas as dependências sejam consistentes. Isso facilita o desenvolvimento, especialmente quando novos membros da equipe se juntam ao projeto, pois eles podem obter todas as dependências necessárias com facilidade, nas mesmas versões e repositórios.

Configurações – Armazenar as configurações separadamente é essencial para evitar a exposição de informações sensíveis, como credenciais e configurações específicas de ambiente. Ao separar as configurações do código-fonte e armazená-las em um local seguro, é possível gerenciar e atualizar essas configurações de forma independente do código. Isso proporciona uma maior segurança e flexibilidade no gerenciamento das configurações em diferentes ambientes de desenvolvimento, teste e produção.

---

[1] Aluno do Curso Especialista Back-End Java da EBAC.

EXERCÍCIO 2 - CI/CD E SERVIÇOS SEM ESTADO (STATELESS)

Explique, com suas palavras, por que é importante desenvolver microsserviços sem estado (stateless) e porque isso pode influenciar diretamente nos mecanismos de CI/CD (integração e desenvolvimento contínuos).
O que acontece se um serviço tiver estado e tivermos que matar uma de suas instâncias em produção?

Ao desenvolver microsserviços sem estado (*stateless*), cada solicitação pode ser tratada independentemente, sem depender de informações contextuais anteriores. Isso traz benefícios significativos para a escalabilidade, resiliência e facilidade de manutenção dos sistemas.

A importância de microsserviços sem estado está relacionada a alguns motivos principais. Primeiro, eles permitem escalar horizontalmente a aplicação de maneira simples e confiável, já que não há necessidade de guardar e compartilhar informações de estado entre as instâncias. Isso facilita a expansão da capacidade de processamento de acordo com a demanda, sem comprometer a consistência das informações.

Além disso, microsserviços sem estado são mais resistentes a falhas. Se uma instância de serviço falhar, outra instância pode assumir a carga imediatamente, sem perder informações de estado importantes. Isso aumenta a resiliência do sistema como um todo, garantindo sua disponibilidade contínua mesmo diante de falhas em instâncias individuais.

Outra vantagem é a facilidade de manter, atualizar, testar e implantar esses microsserviços. Sem a necessidade de lidar com o estado anterior, é mais fácil fazer alterações e atualizações nos serviços, sem interromper o fluxo geral do sistema. Além disso, a ausência de estado simplifica os testes, pois cada solicitação pode ser tratada de forma isolada, facilitando a criação de testes automatizados abrangentes.

No caso de um serviço que tenha estado e seja necessário encerrar uma de suas instâncias em produção, podem ocorrer alguns problemas. Primeiro, a instância encerrada perderá todas as informações de estado que estava mantendo, o que pode levar à perda de dados ou a uma quebra na funcionalidade esperada do serviço. Além disso, se outros serviços ou partes do sistema dependem diretamente dessa instância específica, eles podem encontrar problemas ou falhas devido à sua indisponibilidade.

Portanto, a adoção de microsserviços sem estado minimiza esses problemas, permitindo que outras instâncias assumam a carga e preservem a funcionalidade do sistema. As solicitações subsequentes podem ser tratadas por outras instâncias disponíveis, garantindo a continuidade do serviço sem interrupções significativas ou perda de dados importantes.