1. Imagine o cenário: o Microsserviço de Gerenciamento (reaproveitado da AP1) possui a rota POST /turmas para criar turmas. O cliente precisa criar uma nova Reserva de Sala, que requer o turma\_id para ser vinculada.

1. Para qual dos três microsserviços a requisição POST /reservas deve ser enviada inicialmente e por quê?

**R: deve ser enviado para micro serviços de reservas pois é ele quem cuida do CRUD de reservas, O serviço de Gerenciamento apenas cria e mantém as turmas.**

2. Por que seria um erro arquitetural enviar a requisição de criação de Reserva para o Microsserviço de Gerenciamento? Explique usando o conceito de Domínio de Negócio.

**R: Seria um erro porque o microserviço de gerenciamento cuida do CRUD de aluno, turma e professores, mesmo que a turma esteja incluída na “reserva”.**

2. Qual é a responsabilidade principal do Microsserviço de Gerenciamento na AP2? Detalhe o que ele DEVE fazer e, mais importante, o que ele NÃO DEVE saber ou se importar (em relação aos outros dois serviços).

R: **A responsabilidade do Microsserviço de Gerenciamento é ser o plano de controle operacional.**

**DEVE: Gerenciar configurações, monitorar a saúde e centralizar tarefas administrativas dos outros serviços.**

**NÃO DEVE: Conhecer a lógica de negócio, acessar os dados de outros serviços, ou ser essencial para que o negócio continue a funcionar**

3. Descreva o passo a passo exato do que acontece (incluindo o uso do requests) quando o Microsserviço de Atividades recebe uma requisição POST /atividades que contém um turma\_id para vincular.

1. O que acontece se a chamada HTTP para o Gerenciamento retornar um status 404 Not Found?

R: **Um 404 do serviço de Gerenciamento aciona um mecanismo de defesa no serviço de Atividades, que recusa a criação de dados inconsistentes e informa ao cliente que sua requisição continha uma referência inválida.**

4. A comunicação entre microsserviços em produção usa Endpoints e Hostnames. Se o Microsserviço de Gerenciamento for nomeado gerenciamento no docker-compose.yml, qual a URL completa que o Microsserviço de Reservas deve usar em seu código Python para buscar os detalhes da Turma de ID 42 (assumindo que o Flask roda na porta 5000)? Justifique o uso desse nome/endereço.

R: **Dará erro e quebrará a aplicação caso não tenha tratamento, ou caso tenha tratamento ocorrerá o que foi programado**

5. Na AP2, cada um dos três microsserviços deve ter seu próprio banco de dados SQLite independente.

1. Qual o principal risco técnico e arquitetural se os três serviços compartilhassem o MESMO banco de dados SQLite (como na AP1)?

R: **risco de conflito nos dados, ser realizado uma ação em um dado de outro serviço**

2. Como a separação dos bancos de dados reforça o princípio de encapsulamento de domínio e o conceito de fonte da verdade?

R: **reforçar que cada serviço tem seu próprio banco e gerencie seu escopo, mantendo a integridade dos dados**

6. O modelo de dados do Microsserviço de Reservas deve ter um campo turma\_id. Se a Turma 10 for deletada do Microsserviço de Gerenciamento, as Reservas vinculadas a ela (com turma\_id = 10) continuarão existindo no banco de dados do Reservas.

1. Como essa quebra de integridade referencial tradicional (a Reserva fica “órfã”) é aceita na arquitetura de microsserviços?

R: **Na arquitetura de microsserviços, cada serviço é dono do seu próprio domínio e banco de dados.Isso significa que não existem Foreign Keys entre bancos de serviços diferentes e, portanto, a integridade referencial não é responsabilidade do banco, mas da lógica dos serviços**.

7. A arquitetura da AP2 exige o uso de Comunicação Síncrona (HTTP GET via requests) para validar a existência de um turma\_id ou professor\_id no Microsserviço de Gerenciamento durante uma operação de criação (POST) ou atualização (PUT) nos Microsserviços de Reservas ou Atividades. Descreva uma vantagem e uma desvantagem crítica da escolha da comunicação síncrona para a AP2, no cenário de Criação de uma Reserva:

• Vantagem no Momento da Criação: Por que essa abordagem síncrona é a melhor para garantir que os dados salvos no Reservas sejam válidos?

R: **A vantagem é que o envio é imediato e nao precisa esperar tempos para ser registrado.**

• Desvantagem e Risco de Disponibilidade: Como o Microsserviço de Reservas é afetado se o Microsserviço de Gerenciamento estiver momentaneamente fora do ar ou muito lento? Qual é o termo técnico para esse tipo de acoplamento?

R: **Acoplamento temporal**

8. Qual é o mecanismo técnico usado pelo Docker Compose que permite que o Microsserviço de Reservas encontre e se comunique com o Microsserviço de Gerenciamento usando um nome como http://gerenciamento:5000? Por que, neste ambiente, não se deve usar localhost ou o endereço IP da máquina host?

R: **O mecanismo técnico é a rede interna criada automaticamente pelo Docker Compose, chamada Docker Network**

9. O Cliente faz uma requisição GET /reservas para listar todas as reservas. O Microsserviço de Reservas deve fazer chamadas HTTP para o Microsserviço de Gerenciamento para buscar o nome de cada Turma para incluir na lista? Justifique sua resposta usando o princípio de Limite de Domínio. (Dica: Pense em qual serviço deve ser responsável por apresentar dados combinados ao usuário final).

R: **se necessario aparecer o nome de cada turma sim, pois em cada reserva detem apenas o id de cada turma, caso nao seja necessario, ele apenas demonstra o id de cada turma listada.**

10. Qual é o mecanismo técnico usado pelo Docker Compose que permite que o Microsserviço de Reservas encontre e se comunique com o Microsserviço de Gerenciamento usando um nome como http://gerenciamento:5000? Por que, neste ambiente, não se deve usar localhost ou o endereço IP da máquina host?

R:  **O Docker Compose cria uma rede privada com DNS interno, que resolve o nome do serviço (ex: gerenciamento) para o IP correto do contêiner.**

**Por que não usar localhost ou o IP do host?**

**localhost: Refere-se ao próprio contêiner, não a outros.**

**IP do Host: Não é portátil e muda de uma máquina para outra, quebrando a comunicação.**

11. Imagine que um usuário deseja criar uma nova reserva de sala, enviando uma requisição POST / reservas com um JSON que inclui o campo turma\_id. O Microsserviço de Reservas está operando perfeitamente. Descreva o passo a passo que esse microsserviços terá de dar até poder gravar a reserva no seu banco de dados.

R: **O microserviço reservas deverá consultar no banco de dados do micriserviço gerenciamento usando um get para verificar a existencia de uma turma com o id informado**

**Conceitos**:

• O **Domínio de Negócio** define o escopo e a responsabilidade de uma parte do seu sistema. É a área específica de conhecimento e funcionalidade que um serviço se propõe a resolver. • O conceito de **Fonte da Verdade** garante que, para cada dado ou entidade no sistema, há apenas um lugar que armazena a versão definitiva e canônica (verdadeira) desse dado. • O **Encapsulamento de Domínio** é o mecanismo de proteção que garante a independência e a integridade de um microsserviço, assegurando que o domínio de negócio só possa ser alterado através das regras definidas pelo seu proprietário.

• Tipos de comunicação:

‣ **Síncrona**, que acontece em tempo real. O serviço A envia uma requisição para o serviço B e aguarda a resposta antes de continuar sua própria execução (como uma chamada telefônica). ‣ **Assíncrona**, quando o serviço A envia uma mensagem (um “evento”) para um intermediário (Message Broker) e continua seu trabalho imediatamente, sem esperar. O serviço B processa a mensagem quando for conveniente.

• **Consistência Imediata**: Exige que, após uma alteração no sistema, todos os usuários e processos vejam o dado atualizado instantaneamente. Isso é alcançado tipicamente em arquiteturas monolíticas que usam um único banco de dados e Transações ACID (Atomicidade, Consistência, Isolamento, Durabilidade). Quando um dado é alterado, o banco de dados impõe que todas as referências (Foreign Keys) e consultas sejam atualizadas de forma atômica, garantindo que o sistema como um todo nunca esteja em um estado inconsistente. Na AP2, a Comunicação Síncrona é usada para simular essa consistência no momento da validação, impedindo que dados inválidos sejam criados.

• **Consistência eventual**: Em vez de exigir que todos os dados do sistema sejam atualizados instantaneamente após uma mudança (Consistência Imediata, típica de monolíticos com transações de banco de dados), a Consistência Eventual aceita que pode haver um curto período de tempo onde os dados entre serviços estão inconsistentes.

• **Limite de Domínio** (Bounded Context) é um conceito arquitetural que define a fronteira de responsabilidade de cada microsserviço. Ele garante que cada serviço seja o “rei do seu castelo”, possuindo suas próprias regras de negócio, seu código e seu banco de dados isolado. Esse limite impede o acoplamento excessivo: o Microsserviço de Reservas não se importa com a lógica interna do Professor ou Turma (Domínio de Gerenciamento), apenas usa a API REST (a “porta da frente”) para validar a existência dos IDs. Ao impor essa fronteira, garantimos que o sistema seja mais resiliente, escalável e que uma mudança em um serviço não quebre os demais.