Question 1 (escalabilidade)

Imagine que você precisa realizar 10 milhões (número arbitrário, porém grande) de chamadas a uma HTTP API de uma empresa terceira, todos os dias. Quais são os gargalos que você enxerga? Como você os resolveria?

Precisaria medir a capacidade da API da empresa terceira: quantas requisições seria capaz de sustentar por segundo ao longo de vários minutos. Com esse número em mãos, chegar à conclusão se conseguiria atender com segurança 10 milhões de requisições por dia, o que dá aproximadamente 120 requisições por segundo.

Se não atender, será necessário contatar a empresa terceira sobre o volume de requisições que serão enviadas para a API deles e negociar uma solução conjunta para que se chegue à capacidade necessário. E em caso de atender um volume médio suficiente para suportar 10 milhões de requisições por dia, ainda assim, deve-se tratar cenários de rajada de requisições. Essas rajadas podem ser espalhadas ao longo do tempo com a adição de uma camada de mensageria assíncrona que enfileira as requisições das rajadas e as reenvia para a API a uma taxa máxima que não derrube os servidores da empresa terceira.

Question 2 (separação de conceitos / organização)

Imagina que você precisa implementar uma API HTTP. Como você dividiria as responsabilidades da entrada até a camada de persistência? Exemplo: você irá receber uma chamada HTTP POST para atualizar o telefone da conta do usuário. Onde ficariam as validações? Onde ficaria a persistência? Quem tem a responsabilidade de montar a resposta? Onde você garantia consistência de dados?

As validações devem ser feitas o quanto antes, para que requisições com dados inválidos parem de remandas recursos do servidor imediatamente. O código de validação deve estar próximo da classe que representa o dado sendo inserido/atualizado, e dessa forma, outros serviços que manipulem esse mesmo tipo não precisa reinventar a roda. Depois da validação, o dado pode ser enviado para a camada de persistência, que é onde a consistência de dados pode ser feita pelo SGBD. No caso de haver outros serviços interessados nessa inserção/atualização uma mensagem deve ser enviada para um middleware de mensageria, que garante a entrega da informação e consistência eventual. O resultado da chamada feita à API deve ser montado no próprio endpoint que recebeu a requisição.

Question 3 (segurança)

Imagine que você precisa trafegar dados sensíveis e privados, entre dois serviços, pela internet. Que artifícios/técnicas/ferramentas você usaria para garantir a confidencialidade e integridade e não repúdio da informação?

Para garantir simultaneamente confidencialidade, integridade e não repúdio, é necessário utilizar um mecanismo que implemente criptografia de chaves assimétricas, onde há uma chave privada e uma chave pública.

Ao criptografar uma mensagem com minha chave privada, qualquer um que possua a chave pública que faça par com a chave privada poderá decriptar a mensagem e ter a certeza de que a mensagem foi enviada por mim, desde que eu mantenha a chave privada segura. Em caso de vazamento da chave privada, o não repúdio não é mais garantido.

Quando um terceiro criptografa uma mensagem com minha chave pública, somente eu, de posse de minha chave privada tenho condições de extrair o conteúdo da mensagem e ter certeza de que a informação está confidencial e íntegra. Mais uma vez, garantido que a chave privada não tenha sido copiada indevidamente por terceiros.

Portanto, além de se aplicar a criptografia assimétrica, se faz necessário garantir a segurança da chave privada. Uma forma de se reduzir a exposição de um eventual vazamento da chave privada, é a rotação frequente das chaves. Assim, nenhuma chave é usada por tempo suficiente para que ela caia nas mãos de indivíduos mal intencionados, ou que estes possam geram algum prejuízo.

Question 4 (estrutura de dados)

Se você precisar percorrer uma coleção de elementos de forma sequencial, qual estrutura de dados você usaria? Se você precisasse acessar apenas alguns elementos de uma coleção de forma arbitrária, qual estrutura de dados você usaria?

Para se percorrer uma coleção de elementos em forma sequencial, basta usar uma lista. No Java temos, entre outras, as implementações de lista em ArrayList e em LinkedList. Mas para acessar apenas parte dos elementos de uma coleção, devo usar alguma que faça mapeamento de hash de chaves para os elementos em si. No Java, algumas das estruturas que têm essa propriedade são HashMap e TreeMap.