● Resiliência: O que fazer para mitigar possíveis erros e controlar os possíveis erros recebidos da API?

- Abordagens como escalabilidade e serviços de instâncias em nuvem para garantir um funcionamento contínuo, como arquiteturas divididas ou até mesmo serviços construídos de forma independente ou assíncrono são boas formas de mitigar eventuais indisponibilidades de serviço ou até mesmo queda total de plataformas hoje em dia, outra boa maneira seria o uso de cache ou localStorage que são alternativas que entram na arquitetura de sistemas como formas de diminuir o processamento de dados que não tem muitas alterações por hora.

● Performance: Quais boas práticas são aplicadas em banco de dados e no código para garantir
performance?

- Atualmente é muito prático para garantir performance ao banco de dados o usos de técnica de escalabilidade, técnicas de elastic computing,a integrações com grandes infra estruturas globais que ajudam muitas vezes a entregar um dado mais limpo e performático ao usuário,
  Um exemplo muito comum é escalar servidores com banco de dados por regiões dentro de um país ou estado(topologias distribuídas), assim o usuário de cada região terá sua aplicação funcionando muitas vezes com uma diminuição significativa do ping. pois o mesmo terá sempre suas requisições direcionadas sempre para o servidor mais próximo.

● Segurança: Como garantir segurança para as APIs do sistema?

- Atualmente técnicas de processamento de funções lambdas, autenticação utilizando tokens, cookies, OAuth2, são algumas das formas mais performáticas para assegurar autenticidade, dentro do contexto de apis.

● Simultaneidade: Como trabalhar com simultaneidade se milhares de requisições forem solicitadas
simultaneamente?

Dentro do contexto em que já trabalhei o uso de:

- Simultaneidade reservada que garante o número máximo de instâncias simultâneas para a função. Quando uma função tem simultaneidade reservada, nenhuma outra função pode usar essa simultaneidade. Não há cobrança para configurar a simultaneidade reservada para uma função.
- Simultaneidade provisória que inicializa um número solicitado de ambientes de execução para que eles estejam preparados para responder imediatamente às invocações da sua função.
- Async methods para requisições onde tenhamos consultas caras ao banco de dados
- Cache para dados sem muitas atualizações.