**Respostas – Teste Projeto Clube de Benefícios**

Questão 1 – Elicitação e Desafios

**Como costumo trabalhar com diferentes áreas da empresa (ex: marketing e financeiro):**

1. Primeiro, converso separadamente com cada stakeholder para entender o que cada um espera do produto. Isso ajuda a mapear o que é prioridade e onde estão os pontos de conflito.
2. Depois, tento juntar todos numa mesma conversa (pode ser workshop ou reunião online) para alinhar o que realmente agrega valor ao produto. Aí usamos ferramentas de priorização como matriz valor x esforço ou MoSCoW.

**Requisitos identificados:**

* Funcionais: gestão de assinaturas.
* Não funcionais: segurança e performance.

**Como equilibrar segurança e performance:**

* Segurança: uso autenticação JWT com OAuth2, criptografia AES-256 e logs de auditoria.
* Performance: cache com Redis, microsserviços dedicados para assinatura/pagamento, e arquitetura enxuta.

**Exemplos de cenários de cancelamento:**

* Se o usuário tentar cancelar num período de carência, o sistema mostra a data em que ele poderá sair e permite agendamento.
* Em caso de falha no pagamento, coloco o cancelamento numa fila e notifico o usuário que será processado assim que possível.

**Como valido os requisitos:**

* Crio testes no padrão Given-When-Then.
* Automatizo usando mocks nos testes.
* Valido com protótipos e exemplos junto aos stakeholders.

Questão 2 – Modelagem e Fluxo de Uso

**Modelagem:**  
Uso uma estrutura normalizada, mas onde vale a pena, faço desnormalizações pontuais para ganhar performance.

**Entidades principais:**

* Benefício: tipo, regras, parceiro, elegibilidade.
* Plano: vincula vários benefícios, com regras que podem mudar conforme sazonalidade.

**Fluxo de uso do benefício:**

1. Usuário loga e vê os benefícios disponíveis.
2. Seleciona um → o sistema valida as regras.
3. É gerado um token/voucher → enviado para o parceiro.
4. Sistema registra o uso e coleta feedback.

**Soluções para desafios técnicos:**

* Se o parceiro estiver fora, aplico fallback e notifico a equipe.
* Para regras complexas, uso um motor de regras separado (ex: Camunda).

**Sobre integrações importantes:**

* Para pagamentos, uso RabbitMQ para mandar mensagens de forma assíncrona, com retries e circuit breaker.
* Para parceiros, prefiro webhooks (evita polling) e cuido de versionamento, cache e timeout.

Questão 3 – Arquitetura e MVP

**Stack proposta:**

* Backend em Node.js (microsserviços) com PostgreSQL.
* API Gateway para segurança e controle de requisições.
* Frontend em Angular/Ionic.
* Cache e filas com Redis.
* Monitoramento com Prometheus e Grafana.

**Escalabilidade e organização:**

* Uso containers com Kubernetes.
* Time foca no core do sistema, deixando partes menos críticas isoladas.

**MVP:**

1. Cadastro de usuários e planos.
2. Visualização e uso dos benefícios.
3. Integração com gateway de pagamento.

**Como vou medir se está funcionando:**

* Retenção no 1º mês.
* Quantidade de resgates.
* Feedbacks dos usuários.

Questão 4 – Modelagem (Complemento)

**Trade-offs entre normalização e performance:**

* Normalizar ajuda na manutenção e evita dados duplicados.
* Mas, para relatórios mais pesados, desnormalizo alguns dados (ex: snapshot do plano na assinatura) para garantir performance e manter o histórico intacto.

Questão 5 – Tecnologias

**Node.js no backend:**

* Bom para lidar com APIs e processos assíncronos.
* Tem bastante material na comunidade.
* Para escalar, uso balanceamento horizontal e clustering.

**Angular/Ionic no frontend:**

* Dá pra fazer web e mobile com o mesmo código.
* Só tomo cuidado com performance em telas mais complexas, usando lazy loading, otimização dos bundles, e plugins nativos do Ionic.