Sugestão de respostas para o EE2 – 2024.2

Cenário: A **PetFood** é um aplicativo de entrega de comida para animais de estimação que será uma plataforma inovadora que conecta tutores de pets a fornecedores de alimentação natural de qualidade, facilitando o processo de compra e entrega de ração e petiscos baseados em alimentação natural diretamente na porta dos clientes. O aplicativo busca atender a uma crescente demanda por conveniência e cuidado com a alimentação de animais, garantindo praticidade e segurança tanto para os tutores quanto para os pets.

Você foi contratado como engenheiro(a) de software para liderar as etapas iniciais de desenvolvimento do aplicativo **PetFood**, uma plataforma de entrega de alimentação natural para pets. Seu papel inclui decisões sobre o processo de desenvolvimento, engenharia de requisitos e práticas de controle de versão, considerando que a equipe é multidisciplinar, parcialmente remota e está adotando práticas ágeis.

Sugestões para Diferenciação no Mercado
1. Programa de Fidelidade: Implementar um sistema de pontos que os usuários possam acumular e trocar por descontos em futuras compras.
2. Conteúdo Educativo: Disponibilizar artigos e vídeos sobre alimentação natural adequada e cuidados com a saúde dos pets, atraindo e retendo usuários no aplicativo.
3. Parcerias com Veterinários: Oferecer consultas online com veterinários que podem ajudar os tutores na escolha de produtos adequados para seus animais.
4. Recursos de Integração com Smart Home: Permitir que usuários conectem o aplicativo com dispositivos de smart home para lembretes de alimentação e controle de estoques.

Esse aplicativo não apenas resolverá as necessidades diárias de alimentação natural de animais de estimação, mas também criará uma comunidade de tutores conectados e informados, contribuindo para o bem-estar dos pets.

Questão 01 – (Fácil, peso 2,0)
O controle de versão é essencial em projetos de desenvolvimento de software, especialmente quando se trabalha em equipes distribuídas, como no caso da PetFood. Ele permite que múltiplos desenvolvedores trabalhem simultaneamente no mesmo código de forma coordenada, evitando conflitos e perda de alterações.

No contexto da PetFood, onde há demandas constantes de entrega contínua e integração com diversas funcionalidades (como o programa de fidelidade e integração com smart devices), o uso do Git permite:

1. Criação de branches: cada funcionalidade pode ser desenvolvida em uma branch separada, sem impactar diretamente a versão principal do sistema. Isso facilita a organização do trabalho e a experimentação de novas ideias com segurança.
2. Merge e pull requests: após o desenvolvimento, é possível integrar o código à branch principal com segurança, utilizando revisões em pull requests para garantir a qualidade do código e promover a colaboração entre os membros da equipe.

Essas funcionalidades ajudam a manter o código limpo, rastreável e acessível a todos, mesmo com colaboradores trabalhando remotamente ou em fusos horários diferentes. Além disso, permitem a recu-

peração de versões anteriores em caso de erros ou bugs, garantindo maior controle e segurança durante o desenvolvimento.

Questão 02 – (Intermediária, peso 3,0)
Para levantar e organizar os requisitos da funcionalidade de **programa de fidelidade** no PetFood em um ambiente ágil, a estratégia mais adequada é combinar **técnicas colaborativas de elicitação** com r**epresentações enxutas e iterativas** de requisitos.

Estratégia de Elicitação:

1. **Workshops com stakeholders**: Reuniões colaborativas com o Product Owner, equipe de marketing e alguns usuários-alvo para entender os objetivos do programa, como pontos por compra, níveis de fidelidade e recompensas.
2. **Entrevistas com usuários frequentes**: Identificar expectativas e hábitos de consumo dos tutores para mapear necessidades reais.
3. **Prototipação rápida**: Utilizar protótipos de telas para validar com usuários o fluxo de acúmulo e troca de pontos, incentivando feedback rápido.
4. **Modelagem com user stories**: Documentar os requisitos em histórias do tipo: "Como tutor de pet, quero acumular pontos a cada compra para trocá-los por descontos."

Tipos de Requisitos:

- **Funcionais**: Descrevem o que o sistema deve fazer. Exemplos:
  - Acumular pontos com base no valor da compra.
  - Visualizar saldo de pontos.
  - Resgatar pontos em forma de desconto.
- **Não funcionais**: Descrevem como o sistema deve se comportar. Exemplos:
  - Tempo de resposta da interface de resgate inferior a 2 segundos.
  - Sistema deve ser compatível com Android e iOS.
  - Segurança: proteção contra fraudes e manipulação de pontos.

Organização e Priorização:
Após o levantamento inicial, as user stories podem ser priorizadas em conjunto com o Product Owner usando técnicas como MoSCoW ou valor versus esforço, permitindo a entrega incremental e com feedback contínuo.

Questão 03 – (Difícil, peso 5,0)
Diante das exigências do projeto PetFood — como a necessidade de integração com APIs externas (veterinários, smart home), entregas frequentes e feedback contínuo — é essencial adotar um modelo de processo que promova **flexibilidade, colaboração e entregas incrementais**.
1. Modelo Cascata
Prós: Clareza e controle em projetos com requisitos bem definidos desde o início.
Contras: Rígido e pouco adaptável a mudanças, o que é problemático em startups e projetos inovadores como o PetFood, onde requisitos emergem com o uso.

2. Modelo Incremental
Prós: Permite entregas parciais que são ampliadas com o tempo. Boa visibilidade do progresso e testes desde as primeiras entregas.
Contras: Ainda pode ter certa rigidez se não combinado com práticas ágeis. Não trata bem mudanças constantes nos requisitos.

3. Modelo Ágil (ex: Scrum ou Kanban)
Prós: Alta adaptabilidade, entregas frequentes e valorização do feedback do usuário. Ideal para projetos que evoluem com o uso. Permite trabalhar em ciclos curtos (sprints) com user stories e backlog priorizado.
Contras: Pode ser desafiador em equipes inexperientes, e exige boa comunicação e disciplina nos rituais ágeis.

4. Modelo Híbrido (Ágil + Incremental)
Prós: Une a estrutura e o planejamento do incremental com a flexibilidade das práticas ágeis. Ajuda a escalar com frameworks como SAFe, caso a equipe cresça.
Contras: Requer cuidado na definição de papéis e processos para evitar confusão.

Modelo Recomendado:
Modelo Híbrido com práticas Ágeis (ex: Scrum): Essa combinação oferece o melhor equilíbrio para o PetFood. A startup pode começar com Scrum em pequenos times, entregando funcionalidades como programa de fidelidade e integração com dispositivos smart home em ciclos curtos, enquanto mantém uma visão incremental e modular do produto como um todo.

Isso facilita o crescimento da base de usuários, incorporação de feedbacks e integração de novas funcionalidades ao longo do tempo.