Sugestão de respostas para o EE2 – 2024.2

Cenário: 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.

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)
1. Devem ser apresentadas pelo menos duas estratégias de testes fundamentais, como:

- Testes de unidade: Validam pequenas partes do código (módulos ou funções) para garantir que funcionam corretamente.
- Testes de integração: Avaliam como diferentes componentes do sistema interagem entre si.
- Testes funcionais: Garantem que as funcionalidades do software atendem aos requisitos especificados.
- Testes de aceitação: Validam se o sistema atende às necessidades dos usuários antes do lançamento.
- Testes automatizados: Aceleram a detecção de falhas e garantem qualidade contínua ao longo das entregas.

Também deve justificar por que essas estratégias são importantes, mencionando fatores como:

- Redução de falhas no ambiente de produção.
- Aumento da confiabilidade do sistema.
- Garantia da satisfação do usuário.

Pontos principais esperados na resposta:

- Explicação clara de estratégias básicas de testes.
- Justificativa sobre a importância de aplicar testes no ciclo de desenvolvimento.

2. Os alunos devem conectar os três aspectos mencionados no enunciado com práticas de testes no PetFood.

Uso de métricas quantitativas e histórico de falhas

- Importância: Dados quantitativos ajudam a priorizar testes e identificar áreas mais vulneráveis do sistema.

- Exemplo no PetFood: Analisar métricas como taxa de erro em pagamentos ou falhas na finalização de pedidos para ajustar a cobertura dos testes.
- Prática recomendada: Testes direcionados para áreas que apresentaram maior índice de bugs no passado.

Testes de usabilidade e experiência do usuário
- Importância: Um sistema funcional pode ainda ser difícil de usar, impactando a adoção do produto.
- Exemplo no PetFood: Testes A/B para avaliar se os usuários preferem um fluxo de pagamento mais simplificado.
- Prática recomendada: Coletar feedbacks dos usuários para aprimorar a interface.

Evolução dos testes considerando a experiência humana
- Importância: Testes devem evoluir para considerar como as pessoas interagem com o sistema, garantindo que seja intuitivo e eficiente.
- Exemplo no PetFood: Testes de acessibilidade para garantir que usuários com deficiência visual consigam navegar no aplicativo com leitores de tela.
- Prática recomendada: Incluir usuários reais nos testes de experiência.

Pontos principais esperados na resposta:
- Conexão entre os três aspectos e a qualidade do software.
- Exemplos aplicáveis ao contexto do PetFood.

3. O aluno deve fornecer dois exemplos de testes concretos para o sistema PetFood, como:

Teste de unidade no módulo de pagamento
- Garante que a conversão de moedas e o cálculo de valores com cupons de desconto estão corretos.
- Ajuda a evitar problemas como cobranças erradas ou transações incorretas.

Teste de usabilidade no fluxo de compra
- Avalia se o usuário consegue concluir um pedido com facilidade.
- Pode incluir análise de tempo médio para finalizar uma compra e taxa de abandono do carrinho.

Pontos principais esperados na resposta:
- Pelo menos dois exemplos concretos e aplicáveis ao sistema.
- Justificativa clara sobre por que esses testes são importantes.

Questão 02 – (Intermediária, peso 3,5)
1. Fluxo ideal de CI/CD para evitar instabilidades
Os alunos devem descrever um fluxo estruturado de CI/CD que elimine o problema das atualizações feitas diretamente em produção. Um fluxo ideal inclui:

I. Ambientes separados:
- Desenvolvimento (Dev): Onde os desenvolvedores escrevem e testam código localmente.
- Homologação/Staging: Ambiente que simula a produção e permite testes antes do deployment real.
- Produção: Apenas recebe código que passou por todas as etapas de verificação.

II. Processo de Integração Contínua (CI):
- Uso de um repositório Git com branches estruturadas (ex: `feature`, `develop`, `main`).

- Execução automática de testes unitários, integração e estáticos a cada commit.
- Revisão de código por pares antes da fusão de branches.

III. Entrega Contínua (CD):
- Pipeline configurado para liberar atualizações primeiro no ambiente de homologação.
- Testes de aceitação automatizados antes do deploy em produção.
- Deploy automatizado, preferencialmente gradual (Blue-Green Deployment ou Canary Releases).

Pontos principais esperados na resposta:
- Explicação clara dos diferentes ambientes no fluxo CI/CD.
- Definição de boas práticas, como testes automatizados e revisão de código.
- Mecanismos para garantir um deploy seguro, evitando instabilidade.

2. Os alunos devem relacionar a automação de testes com a confiabilidade do sistema. Os seguintes pontos são esperados:

I. Testes unitários
- Validação de funções e classes isoladas para evitar erros básicos.

II. Testes de integração
- Garante que módulos interagem corretamente antes do deploy.

III. Testes de regressão
- Impedem que mudanças no código quebrem funcionalidades antigas.

IV. Testes de aceitação automatizados
- Simulam o comportamento do usuário final para validar se o sistema funciona como esperado.

V. Testes de carga e desempenho
- Avaliam a escalabilidade do sistema antes de grandes lançamentos.

VI. Monitoramento e feedback contínuo
- Alertas automáticos para falhas pós-deploy garantem reação rápida.

Pontos principais esperados na resposta:
- Relação entre tipos de testes e estabilidade do PetFood.
- Justificativa sobre como a automação reduz falhas e previne instabilidade.

Questão 03 – (Difícil, peso 4,5)
1. Segurança como Responsabilidade Compartilhada
- Prática: Estabelecer um modelo de segurança "Shift Left", onde todos os membros da equipe (desenvolvedores, DevOps, QA) assumem responsabilidade pela segurança desde o início do ciclo de desenvolvimento.
- Justificativa: Evita que a segurança seja tratada apenas no final do projeto, reduzindo retrabalho e aumentando a confiabilidade do software.
- Ação concreta: Criar uma cultura de segurança por meio de códigos de conduta, revisões de segurança obrigatórias no desenvolvimento e boas práticas documentadas.

2. Treinamentos Regulares para Mudança de Mentalidade
- Prática: Implementar workshops regulares sobre segurança para equipes de desenvolvimento, QA e DevOps.

- Justificativa: Equipes bem treinadas são capazes de identificar riscos antes que se tornem falhas críticas.
- Ação concreta:
  - Simulações de ataques (ex: Red Team/Blue Team).
  - Treinamentos em Secure Coding para evitar vulnerabilidades comuns (SQL Injection, XSS).
  - Hackathons internos para reforçar boas práticas de segurança.

3. Ferramentas Automatizadas no Pipeline
- Prática: Integrar ferramentas de segurança ao CI/CD para verificar vulnerabilidades continuamente.
- Justificativa: Garante que falhas sejam detectadas antes do deploy, evitando incidentes em produção.
- Ação concreta:
  - Implementação de SAST (Static Application Security Testing) para análise de código estática.
  - Uso de DAST (Dynamic Application Security Testing) para testar segurança da API do PetFood.
  - Adoção de ferramentas como SonarQube, OWASP Dependency-Check e GitHub Advanced Security.

4. Superar a Resistência à Mudança com Apoio da Liderança
- Prática: Criar um plano de conscientização sobre DevSecOps, promovendo o alinhamento entre segurança e estratégia de negócio.
- Justificativa: Muitas equipes resistem à mudança porque veem segurança como um obstáculo à velocidade de entrega.
- Ação concreta:
  - Criar métricas para mostrar como segurança pode acelerar entregas seguras.
  - Envolver CTO e líderes técnicos em treinamentos e reuniões sobre segurança.

5. Monitoramento Contínuo para Detecção de Ameaças
- Prática: Implementar ferramentas de monitoramento em tempo real para identificar ataques e atividades suspeitas.
- Justificativa: Redução do tempo de resposta a incidentes e prevenção proativa contra falhas.
- Ação concreta:
  - Uso de SIEM (Security Information and Event Management) para análise de logs.
  - Implementação de alertas automáticos para detectar padrões de ataques na API.

6. Segurança como Impulsionadora de Inovação
- Prática: Encarar segurança como um fator de aceleração de inovação, e não como um bloqueio.
- Justificativa: Equipes que aplicam segurança desde o início conseguem lançar produtos mais confiáveis e sustentáveis.
- Ação concreta:
  - Criar um programa de incentivo para desenvolvedores que aplicam boas práticas de segurança.
  - Incluir segurança no roadmap de inovação do PetFood.