Sugestão de respostas para o EE1 – 2024.2

Cenário: Roz, o robô protagonista, precisa aprender a sobreviver e se comunicar com os animais da ilha. Para isso, entra em "modo de aprendizado" e começa a coletar dados do ambiente e do comportamento dos animais. Roz deve processar esses dados para descobrir padrões, desenvolver formas de interação e aplicar metodologias eficazes para garantir a qualidade das suas decisões, a escalabilidade de suas interações e a adaptação ao volume crescente de dados coletados.

Questão 01 [2,0] - Possível resposta:

Parte A: Escolha de Critério de Qualidade e Justificativa

- O aluno pode escolher entre os critérios SMART ou INVEST.
- Explicação resumida do critério:
  - SMART: Focado em objetivos que sejam Específicos, Mensuráveis, Alcançáveis, Relevantes e Temporalmente definidos. Garante clareza e rastreabilidade.
  - INVEST: Baseado nos princípios de que uma HU deve ser Independente, Negociável, Valiosa, Estimável, Pequena e Testável. Focado em facilitar o gerenciamento ágil.
- Justificativa da escolha:
  - SMART: Adequado para projetos que exigem rastreabilidade e objetivos bem definidos, como o aprendizado específico de Roz.
  - INVEST: Melhor para ambientes ágeis, onde a flexibilidade e a entrega de valor incremental são essenciais, como no caso de Roz se adaptar continuamente ao ambiente dinâmico.

Parte B: Elaboração de Três Histórias do Usuário

- Resposta esperada (exemplo usando o critério SMART):

1. Como um robô,
   Quero identificar padrões de perigo nos comportamentos dos gansos,
   Para que eu possa reagir rapidamente a ameaças na ilha.
   - Específico: Identificar padrões de perigo.
   - Mensurável: Reconhecer pelo menos 5 tipos de alertas distintos.
   - Alcançável: Baseado em dados coletados ao longo de 1 semana.
   - Relevante: Ajuda na sobrevivência de Roz e dos animais.
   - Temporalmente definido: Implementação em 2 *sprints*.

2. Como um robô,
   Quero interpretar e responder ao chamado de alimentação dos gansos,
   Para que eu possa melhorar minha comunicação com eles.
   - Específico: Foco no chamado de alimentação.
   - Mensurável: Testar com 10 interações de sucesso.
   - Alcançável: Uso de dados existentes.
   - Relevante: Fortalece a relação com os animais.
   - Temporalmente definido: Finalizado em 1 *sprint*.

3. Como um robô,
   Quero catalogar o comportamento de diferentes espécies de animais,
   Para que eu possa expandir meu vocabulário de comunicação.
   - Específico: Catalogar pelo menos 3 espécies.
   - Mensurável: Criar um banco de dados com pelo menos 20 interações registradas.

- Alcançável: Baseado em observações contínuas.
- Relevante: Ajuda na integração ao ambiente.
- Temporalmente definido: Concluir catalogação inicial em 3 *sprints*.

- Resposta alternativa (usando INVEST):
- Similar às histórias acima, mas alinhadas aos princípios de INVEST, garantindo independência entre histórias e foco em valor incremental.

Parte C: Gerenciamento de Mudanças
- Resposta esperada:
- Abordagem sugerida:
1. Revisão contínua: Requisitos são revisados ao final de cada *sprint*, com priorização de novos itens no *backlog*.
2. Gerenciamento de *backlog*: Todas as mudanças são registradas no *backlog*, avaliando impacto em termos de esforço e valor agregado.
3. Definição clara de prioridades: O *Product Owner* ou equivalente prioriza os requisitos com base no impacto no sistema e nos objetivos do projeto.
4. *Feedback* contínuo: Incorporar *feedback* das implementações iniciais para refinar futuras entregas.
- Ferramentas recomendadas:
- Uso de ferramentas ágeis como Jira, Trello ou Azure DevOps para registrar, monitorar e priorizar mudanças.

Questão 02 [3,0] - Possível resposta:
Parte A: Escolha do Modelo de Processo de Desenvolvimento
- Resposta esperada:
- O aluno deve escolher um modelo de processo de desenvolvimento de software, justificando como ele atende às demandas do cenário. Possíveis escolhas e alinhamento:
- Modelo Ágil:
Ideal para lidar com o ambiente dinâmico da ilha, onde os requisitos podem mudar com frequência e é necessário entregar funcionalidades de forma incremental.
- Modelo Iterativo:
Permite prototipagem e melhorias contínuas, útil para explorar soluções iniciais e refiná-las com base em *feedback* do ambiente e dos dados coletados.
- Modelo Híbrido (Ágil + Iterativo):
Combina a flexibilidade do Ágil com o refinamento progressivo do Iterativo, sendo uma boa escolha para projetos complexos e adaptáveis.
- Modelo Cascata:
Raramente seria a escolha adequada aqui devido à rigidez, mas o aluno poderia justificá-lo para partes específicas, como o desenvolvimento de componentes básicos e estáveis do sistema.

Parte B: Justificativa com Vantagens e Desvantagem
- Resposta esperada:
- Modelo Ágil (Exemplo de Resposta):
- Vantagem 1: Alta flexibilidade para responder às mudanças frequentes no comportamento dos animais e no ambiente dinâmico.
- Vantagem 2: Entrega de valor incremental permite que Roz comece a interagir com os animais rapidamente, enquanto novas funcionalidades são desenvolvidas.
- Desvantagem: Pode ser difícil prever o escopo completo do projeto, especialmente com requisitos emergindo ao longo do tempo.

- Modelo Iterativo (Exemplo de Resposta):
  - Vantagem 1: Permite prototipar e ajustar as estratégias de aprendizado com base em *feedback* do ambiente.
  - Vantagem 2: Minimiza os riscos, pois as iterações iniciais ajudam a identificar falhas antes de comprometer todo o sistema.
  - Desvantagem: Iterações podem ser demoradas, especialmente se cada ciclo envolver muitas revisões.

- Modelo Híbrido (Exemplo de Resposta):
  - Vantagem 1: Combina o melhor de dois mundos: adaptação rápida às mudanças e entrega de valor incremental.
  - Vantagem 2: Estrutura clara para fases específicas do projeto, como planejamento inicial e prototipagem.
  - Desvantagem: Pode ser complexo de gerenciar, especialmente para equipes inexperientes ou com recursos limitados.

- Modelo Cascata (Exemplo de Resposta):
  - Vantagem 1: Ideal para desenvolver componentes básicos que não precisam ser alterados frequentemente, como o núcleo do sistema de coleta de dados.
  - Vantagem 2: Estrutura clara e bem documentada facilita o entendimento do sistema.
  - Desvantagem: Falta de flexibilidade para adaptar-se a mudanças frequentes no ambiente dinâmico da ilha.

Questão 03 [5,0] - Possível resposta:
Parte A: Organização das Ramificações (*Branches*)
- Resposta esperada:
- Proposta de organização clara e alinhada às boas práticas de controle de versão. Exemplo:
  1. *Branch* `main` (ou `master`): Contém a versão estável do sistema, com todas as funcionalidades testadas e aprovadas.
  2. *Branch* `develop`: Reúne o trabalho em andamento, com novas funcionalidades sendo integradas e testadas antes de irem para produção.
  3. *Feature branches*: Criadas a partir de `develop` para cada nova funcionalidade ou ajuste, como:
     - `feature/aprendizado-comunicacao`
     - `feature/estrategia-alerta`
     - `feature/catalogo-especies`
  4. *Hotfix branches*: Criadas a partir de `main` para corrigir problemas críticos identificados em produção.
  5. *Release branches*: Criadas para preparar o sistema para uma nova versão estável, incluindo ajustes finais e validações.

Parte B: Estratégias de *Merge* e *Branch*
- Resposta esperada:
- Estratégias de *Merge*:
  - *Merges* são feitos das *feature branches* para o *branch* `develop` após a conclusão e testes da funcionalidade.
  - Após validação completa no *branch* `develop`, o código é integrado ao *branch* `main`.

- *Pull requests* (PRs) devem ser usados para revisar o código antes do *merge*, garantindo qualidade e colaboração da equipe.
- Minimização de Conflitos:
  - Adotar integrações frequentes entre *branches* para evitar divergências acumuladas.
  - Usar ferramentas de CI/CD (como Jenkins, GitHub Actions ou GitLab CI) para automatizar testes e validações antes do *merge*.
- Facilitação do Trabalho Colaborativo:
  - Manter a equipe informada com reuniões rápidas (*daily stand-ups*) e atualizações no sistema de controle de versão.
  - Nomear as *branches* de forma clara e consistente, para evitar confusão e sobreposição de trabalho.

Parte C: Ferramenta de Controle de Versão
- Resposta esperada:
- Ferramenta Recomendada: Git (junto com plataformas como GitHub, GitLab ou Bitbucket).
- Justificativa:
  1. Permite o gerenciamento eficiente de *branches* e *merges*, suportando práticas ágeis.
  2. Ferramentas integradas (como GitHub Actions) automatizam pipelines de CI/CD para validação e integração contínua.
  3. Funcionalidades como *pull requests* facilitam a revisão de código e a colaboração entre membros da equipe.
  4. Histórico detalhado de *commits* ajuda a rastrear mudanças e corrigir problemas rapidamente.