Exame de Época Normal – 2019-01-15 15h00. Duração: 90min. A1

NOME: NR. MEC:

Questões de escolha múltipla: **responda na grelha**, assinalando uma opção por pergunta (pretende-se a opção verdadeira e, havendo várias que possam ser consideradas verdadeiras, pretende-se a mais abrangente); as não-respostas valem zero; **respostas erradas descontam** ¼ da cotação; as respostas assinaladas de forma ambígua serão consideradas não-respostas. Questões 24 e 25: responder no espaço vazio, no final do enunciado.

Grelha de respostas da escolha múltipla (perguntas 1 a 23):

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a)																							
b)																							
c)																							
d)																							
e)																							

P1.

O método *Unified Process* prevê quatro fases principais no desenvolvimento do projeto, cada qual com um objetivo a atingir (*milestone*) para se poder avançar, por esta ordem:

a) 1/ Definição das histórias de utilização; 2/ Gestão ativa do *backlog*; 3/ Arquitetura técnica validada; 4/ Implementação da solução concluída.
b) 1/ Preparação do documento de Visão; 2/ Demonstração de protótipos exploratórios; 3/ Produto implementado; 4/ Testes no cliente.
c) 1/ Plano para o projeto definido; 2/ Análise de requisitos terminada; 3/ Arquitetura técnica do produto definida; 4/ Implementação da solução concluída.
d) 1/ Definição dos casos de utilização; 2/ implementação do protótipo exploratório; 3/ definição da arquitetura; 4/ Implementação do produto concluída.
e) 1/ Decisão de avançar ou parar o projeto; 2/ Arquitetura técnica definida e validada; 3/ Funcionalidades da primeira versão do produto implementadas; 4/ Solução instalada e aceite pelo cliente.

P2.

Uma das principais razões para se utilizar métodos ágeis de desenvolvimento, em detrimentos dos métodos sequenciais, é a diminuição do risco do projeto. Que prática é decisiva para a mitigação do risco:

a) A ordem dos itens na pilha do *backlog* é imutável, tornando o projeto mais previsível.
b) Os projetos são mais pequenos e não se gasta tanto tempo em tarefas de coordenação e documentação.
c) O teste do software é deslocado para o início do projeto, garantido sempre que há tempo para o fazer.
d) O projeto está dividido em iterações autónomas e independentes.
e) A entrega frequente de valor diminui a probabilidade de eventuais divergências na perceção dos requisitos.

P3.

Os requisitos devem apresentar as caraterísticas conhecidas por S.M.A.R.T. Identifique, na lista, um requisito não-funcional adequadamente formulado.

a) "O sistema deve permitir a transferência de saldo entre contas do mesmo gestor."
b) "O utilizador deve poder exportar uma cópia de todos os seus dados, mantidos no sistema, para um documento PDF."
c) "Usar uma estrutura visual de elementos intuitiva por forma a tornar a experiência de utilização agradável"
d) "A renovação do aluguer tem de ser realizada até 24 horas antes de terminar o tempo já contratado."
e) "A resposta com a autorização por parte do sistema de micro-pagamentos não pode demorar mais de 2 segundos."

P4.

Os casos de utilização e os requisitos funcionais descrevem um sistema segundo uma perspetiva de "caixa opaca". Qual a relação que se aplica entre estas duas técnicas:

a) Os requisitos funcionais descrevem o sistema com texto, enquanto que os casos de utilização recorrem a uma linguagem visual.
b) Os casos de utilização consideram as necessidades dos atores; os requisitos funcionais consideram o sistema como um todo.
c) Os casos de utilização captam os requisitos funcionais de forma contextualizada, nas descrições estruturadas dos cenários de interação.
d) Os casos de utilização captam apenas requisitos não-funcionais.
e) É sempre necessário desenvolver a documentação detalhada dos casos de utilização e o documento de requisitos funcionais (SRS).

P5.

Os conceitos de Ator e Persona são usados na análise de sistemas para descrever cenários de uso de um sistema.

a) Uma Persona é uma forma mais específica de um Ator (relação de herança).
b) A Persona é uma instância/concretização de um Ator.

c) A Persona e o Actor podem ser representados no Diagrama de Casos de Utilização.
d) Persona e Ator não estão associados, pois a Persona não pode ser usada para modelar um sistema externo, e o Ator sim.
e) A inclusão de Personas na Análise é desaconselhada porque o conceito não existe na UML.

P6.

A arquitetura trata da tomada das grandes decisões técnicas em relação ao sistema a desenvolver, tendo em conta os atributos de qualidade pretendidos. Um exemplo de um assunto/decisão de arquitetura é:

a) Especificar os cenários de interoperação com sistemas externos e as tecnologias selecionadas para os implementar.
b) Definir estratégias de distribuição de carga para garantir a disponibilidade do sistema em utilização contínua, com 1000 sessões simultâneas.
c) Definir os mecanismos técnicos para separar a informação pessoal da informação operacional, para dar resposta aos requisitos previstos no RGPD.
d) Todas as opções anteriores são corretas.
e) Nenhuma das opções anterior é correta.

P7.

Os padrões de desenho fornecem soluções conhecidas para problemas de programação comuns. Alguns exemplos de padrões relacionados com a criação de objetos (*creational*) são:

a) Abstract Factory, Factory Method, Singleton.
b) Façade, Adaptor, Singleton.
c) Façade, Adaptor, Strategy.
d) Observer, Visitor, Strategy.
e) Cohesion, Single Responsability, Observer.

P8.

Um dos princípios "SOLID" para o desenho por objetos, preconiza a segregação das interfaces. Como é que esse princípio deve ser usado pelo programador?

a) Evitar a utilização de interfaces, preferindo a utilização de classes abstratas.
b) Criar interfaces específicas, evitando que os clientes dependam de interfaces que não utilizam.
c) Construir classes coesas, que apresentam uma única responsabilidade.
d) Manter reduzido o número de classes que implementam uma interface.
e) Usar o padrão "Façade" para isolar a classe cliente da implementação (complexa) do serviço.

P9.

No cerne da abordagem TDD (*test-driven development*), está um ciclo de trabalho com um fluxo característico. Como se desenrola o trabalho do programador?

a) Adicionar um pequeno teste; executar os testes e verificar que o novo está a falhar; implementar as alterações suficientes para o teste passar; executar os testes e verificar que o novo passa; rever o código para o tornar mais claro.
b) Implementar as alterações relativas ao novo incremento; adicionar um teste; executar os testes até se verificar que o novo teste passa.
c) Melhorar a clareza do código; implementar um pequeno incremento; escrever um teste para confirmar que o incremento faz o esperado.
d) Executar o código; identificar problemas com o incremento implementado; usar ferramenta de *debugging* para encontrar a origem; resolver os erros.
e) Criar testes funcionais a partir das histórias (*"user stories"*); implementar as funcionalidades necessárias (para os testes passarem); suplementar o projeto com testes unitários.

P10.

Uma forma comum de alinhar as histórias do utilizador (*user stories)* e os métodos de garantia de qualidade do software é:

a) As histórias são usadas diretamente para alimentar testes automáticos, na web.
b) As histórias descrevem os objetivos que as *personas* pretendem realizar no sistema, utilizando o modelo "Sendo um…Quero [realizar]…De modo a…".
c) As histórias são suplementadas com a descrição de cenários genéricos, identificando as sequências alternativas em caso de erro.
d) As histórias incluem cenários concretos, representativos das situações de sucesso.
e) As histórias são suplementadas com a descrição de cenários concretos, que estabelecem as condições de aceitação do incremento.

P11.

Qual das seguintes sequências de passos deve ocorrer num processo de Integração Contínua?

a) Entrega de código (*commit*), resolução de dependências e compilação no ambiente de integração, execução dos testes automáticos, disponibilização de *feedback* quanto ao estado da *build*.
b) Entrega de código (*commit*) pelo programador, testes de aceitação, testes unitários.
c) Entrega de código (*commit*), testes unitários automáticos, testes de aceitação automáticos, instalação em produção.
d) Deteção de alterações ao código na *workstation* do programador, execução de testes automáticos, instalação no ambiente de pré-produção.
e) Deteção de novo código no repositório, testes automáticos, correção automática dos erros.

P12.

Na terminologia dos projetos de desenvolvimento ágil, o que é a velocidade da equipa (numa iteração)?

a) O número de submissões para o repositório partilhado (*commits*), por iteração.
b) O número de histórias (*use stories*) completadas e aceites por iteração.
c) O número de entregas (de incrementos à solução) feitas ao cliente, e aceites.
d) Os pontos acomulados das histórias implementadas, por iteração.

e) O número de linhas de código (LoC) implementas pela equipa, por iteração.

P13.

O modelo do domínio é construído na Análise, para mapear os conceitos do universo de discurso.

a) As classes do modelo do domínio são as mesmas classes do código e por isso existe uma continuidade.
b) As classes do modelo do domínio captam a estrutura da informação; terão utilidade posterior para desenhar a base de dados, mas não o código.
c) Os conceitos identificados no modelo do domínio serão candidatos naturais a constituir classes de programação, numa linguagem por objetos.
d) A estrutura dos dados de um problema é muito volátil e, por isso, o modelo do domínio é provisório e vai ser alterado.
e) O modelo do domínio só é construído para projetos que requerem a utilização de bases de dados.

P14.

Qual o papel caraterístico do Analista numa equipa de desenvolvimento?

a) É o representante dos interesses dos *stakeholders* do projeto, na definição dos requisitos do produto.
b) É um profissional especializado no domínio do problema em que o projeto se situa e, por isso, compreende bem o funcionamento do negócio.
c) Analisa os processos da organização para identificar oportunidades de melhoria, e delineia o sistema de informação que as implementa.
d) Apresenta uma competência técnica acima da média, o que lhe permite fazer as escolhas tecnológicas da implementação.
e) Assegura a gestão diária do projeto e o diálogo com o cliente.

P15.

O Diagrama 1 utiliza o conceito de estereótipo da UML (*stereotype*) na relação "extend".

a) "Configurar módulos do projeto" e "Configurar a equipa" são formas opcionais de complementar o fluxo do caso de utilização "Configurar projeto" ok
b) É sempre possível substituir o estereótipo «extend» por «include», alterando a direção da associação.
c) O estereótipo "extend" é acessório, não altera a natureza da relação de dependência.
d) A utilização da associação de «extend» deve ser evitada neste nível de abstração.
e) Por coerência, a associação entre os atores também deveria ser anotada com «extend».

P16.

Relativamente ao modelo representado no Diagrama 1:

a) Utiliza a relação de hierarquia entre atores, mas está mal aplicado, uma vez que o "Gestor de projeto" é que é o superior hierárquico no projeto.
b) Está incompleto, porque não representa a base de dados como um ator.
c) Deveria relacionar o caso de utilização "Atualizar ticket" com "Abrir ticket", do qual aquele depende temporalmente.
d) Ao consultar o estado de uma tarefa, o Membro da Equipa pode opcionalmente estendê-la, i.e., dilatar a sua duração.
e) O "Gestor de Projeto" também é um "Membro da Equipa". ok

P17.

Nos elementos modelados no Diagrama 1 há um que está desajustado e carece de revisão. Trata-se de:

a) O ator Gestor de Projeto é supérfluo (é redundante e deve ser retirado).
b) "Criar projeto" e "Configurar projeto" não podem ser casos de utilização autónomos.
c) O caso de utilização "Notificações de progresso" não evidencia nenhum fluxo. ok
d) A associação de «extend» entre "Atualizar ticket" e "Consultar estado das tarefas" tem a direção errada.
e) As associações de «extend» entre "Configurar projeto" e "Configurar módulos"/"Configurar equipa" deveriam ser substituídas por «include».

P18.

Considere a capacidade expressiva do Diagrama 2:

a) Um Cliente usa sempre o mesmo Outdoor.
b) Um Cliente usa um Outdoor na sua zona.
c) Um Cliente pode ter até dois Anúncios no mesmo Outdoor (em simultâneo).
d) Um anúncio pode ser afixado num Outdoor ou em outros tipos de suporte.
e) Um cliente pode ter um anúncio afixado em diferentes Outdoors.

P19.

Que alterações seria necessário fazer ao Diagrama 2 para captar o requisito: "Um funcionário faz a inspeção de Outdoors da sua zona de atuação."

a) Já é possível, sem alterar o diagrama, representar essa informação.
b) A classe Outdoor deve ter um atributo com a localização.
c) Deve ser introduzida uma classe Zona, associando-a a Funcionário e a Outdoor.
d) A classe Funcionário deve estar associada à classe Outdoor, e a associação anotada com uma classe de associação.
e) A classe Funcionário deve ter uma agregação de Outdoors, que são os da sua responsabilidade.

P20.

Relativamente ao Diagrama 2:

a) Todas as Inspeções são realizadas por um único Funcionário.
b) Se se eliminar um objeto Percurso, os objetos Outdoor associados são também eliminados.
c) O Percurso que o Funcionário faz não define a sequência de visita aos Outdoor.

d) O custo de um anúncio depende do tipo de papel.
e) Cada Outdoor é indicado para um tipo de papel.

P21.

Relativamente ao Diagrama 3:

a) Pode ser um subdiagrama associado a uma classe Perfil ("User account")
b) Deve ter associado um subdiagrama (de classes) para mostrar a classe Perfil ("User account") e as suas dependências.
c) Deve ter um diagrama de atividades associado a cada transição de estado, para clarificar os processos de trabalho.
d) É diretamente transponível para um diagrama de comunicação.
e) Apesar de existir na UML, não é suportado pelas ferramentas de modelação.

P22.

O Diagrama 3 descreve o ciclo de vida de um perfil de utilizador num site de vendas on-line.

a) Todos os perfis ("User Account") passam pelo estado Ativo.
b) Todas os perfis podem ser Encerrados ("Closed").
c) A suspensão do perfil é condicional; faltam nós de decisão.
d) O perfil precisa de ter estado Suspenso, para poder ser Encerrado.
e) Todas as hipóteses anteriores estão erradas.

P23.

Relativamente ao tipo de diagrama ilustrado no Diagrama 3 e à sua relação com outros diagramas da UML:

a) Os estados representados podem ser mostrados num diagrama de atividades, como estados de uma entidade de dados (associada a um *object flow*).
b) O diagrama é a especificação do ator "Membro da Equipa" (do Diagrama 1); no VisualParadigm, isso corresponderia a criar o Diagrama 3 "dentro" do nó desse ator.
c) As condições de acesso representadas neste diagrama são as mesmas que apareceriam no Diagrama de Atividades correspondente.
d) As ações assinaladas nas transições correspondem aos eventos temporais num diagrama de sequência.
e) A informação representada é suscetível de ser mostrada no diagrama alternativo, como o diagrama de sequência.

P24.**[questão de desenvolvimento]**

Várias abordagens ao desenvolvimento de software partem do estudo de cenários de utilização como estratégia para determinação de requisitos.

a) Qual é o benefício de utilizar técnicas de determinação de requisitos baseadas em cenários de utilização (por contraponto à decomposição funcional)? São adequadas para todos os tipos de projetos? Justifique.
b) Compare a utilização dos casos de utilização (*use cases*) e das histórias do utilizador (*user stories*) enquanto técnicas de especificação, destacando os aspetos comuns e diferenciadores entre elas.

P25. **[questão de desenvolvimento]**

Considere o trecho de código seguinte, em Java, com omissões.

a) Apresente um diagrama de classes para visualizar a informação estrutural que se pode depreender deste código.
b) Apresente um diagrama de sequência para representar a interação entre objetos que ocorre quando é invocado o método WeatherRepository#**refeshForecastIfNeeded**.

```
9   /**
10   * Manages the requests to get the weather forecast. If the local data
11   * is still recent, no remote requests are made. Otherwise, the IPMA's
12   * API is invoked.
13   */
14  public class WeatherRepository {
15      private static final String BASE_URL = "https://api.ipma.pt/open-data/";
16      private static final int REFRESH_PERIOD = 300;
17
18      private static RemoteWeatherAPI apiService = new IpmaApiClient( BASE_URL);
19      private MyDatabase localDb;
20
21      public void refreshForecastIfNeeded( int placeId,  Date day) {
22          boolean goodLocalData = localDb.hastUpdatedForecast(placeId, day, REFRESH_PERIOD);
23          if (! goodLocalData) {
24              WeatherForecast forecast = null;
25              try {
26                  forecast = apiService.getForecastForPlace( placeId);
27                  if (forecast != null) {
28                      localDb.save(forecast);
29                  }
30              } catch (IOException e) { e.printStackTrace(); }
31          }
32      }
33  }
```

Folha de Diagrama

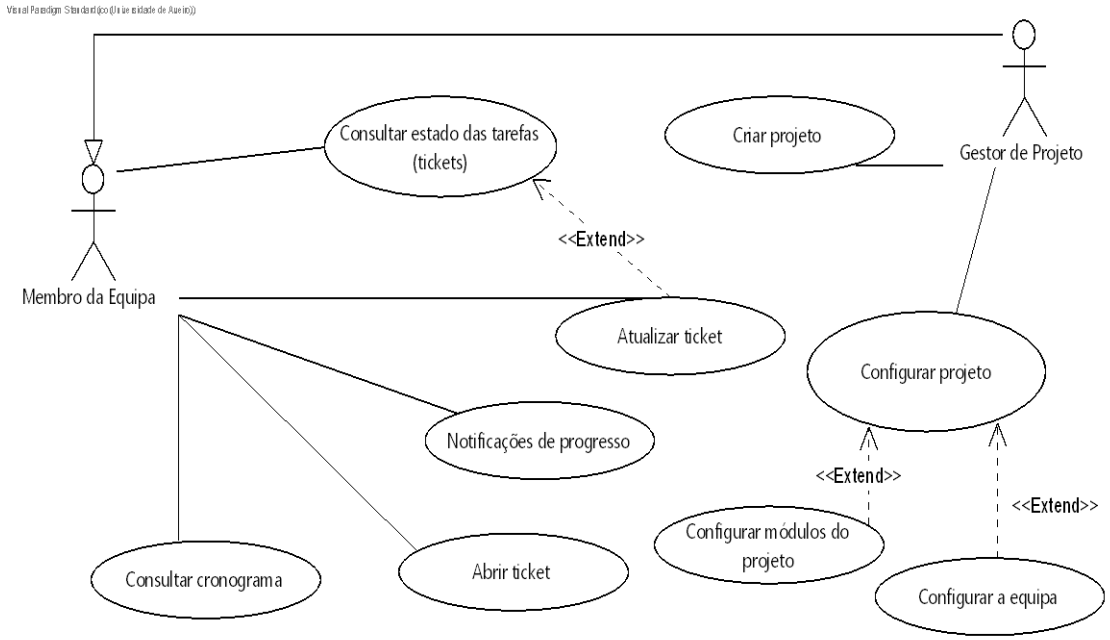

Diagrama 1- Descrição funcional de um portal de gestão de projetos (similar ao produto *Redmine*).

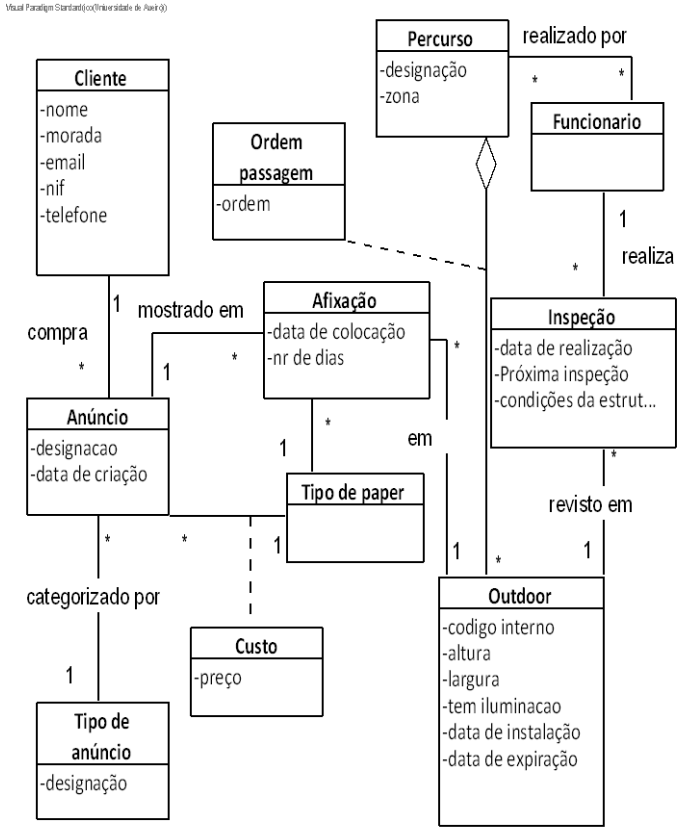

Diagrama 2- Conceitos associados a um sistema de informação relativo a venda de espaços publicitários.

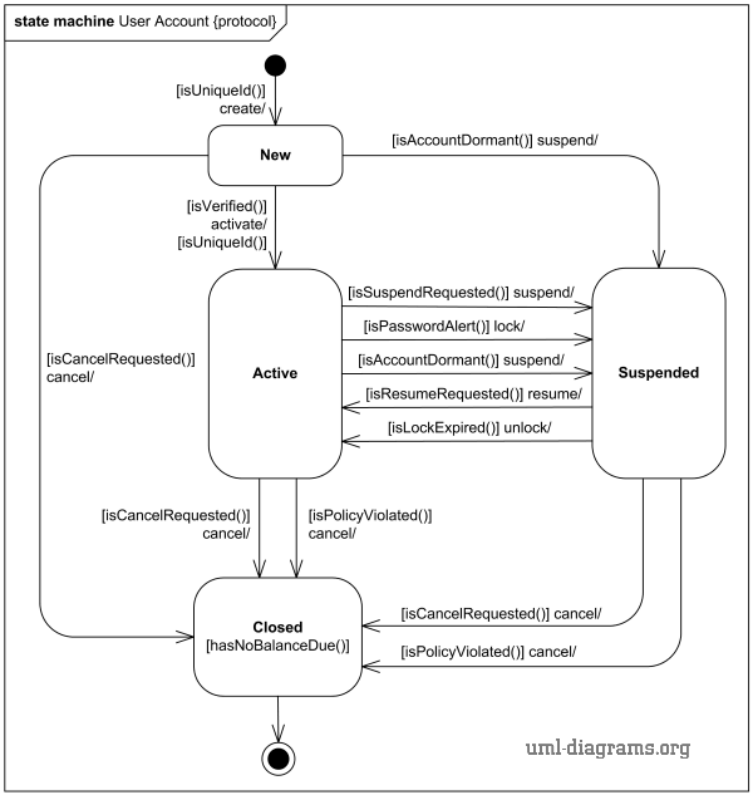

Diagrama 3: Ciclo de vida de uma conta de utilizador (*User Account*).

Questões de desenvolvimento

NOME: NR. MEC: