

Componente Teórica

60% da nota da prova escrita. Nota mínima de 8 valores

Para a pergunta 1, respostas erradas descontam metade da cotação.

1. Para as seguintes afirmações indique se são verdadeiras ou falsas (14 valores):
 - a. Um padrão de software é um conjunto de boas práticas desenvolvidas com base na observação de trabalhos bem sucedidos.
 - b. O padrão *Identity Map* garante que apenas uma instância de uma dada entidade de negócio está em memória ao efectuar um *lock* ao registo correspondente na tabela.
 - c. O padrão *Table Module* nunca é adequado para aplicações com lógica de negócio complexa.
 - d. Numa aplicação construída tendo por base o modelo 3 camadas, a camada de apresentação pode efectuar validações dos dados introduzidos para melhorar a usabilidade da aplicação.
 - e. O nome de um padrão facilita a comunicação entre membros da equipa de desenvolvimento e cria um vocabulário comum apenas perceptível pela equipa de desenvolvimento.
 - f. O padrão *Factory* permite esconder os pormenores de criação de um objecto facilitando dessa forma a implementação de alternativas para uma mesma interface.
 - g. Usando o padrão *Row Data Gateway* os métodos *finder* podem ser implementados como métodos de classe na própria classe ou numa classe separada.
 - h. O padrão *Active Record* mistura na mesma classe a lógica de negócio e lógica de acesso a dados, impossibilitando dessa forma a separação em camadas (*layers*).
 - i. O padrão *Offline Pesimistic Lock* é difícil de implementar em ambientes desconectados do servidor (ex., web).
 - j. O padrão *Strategy* permite encapsular um algoritmo numa classe que pode ser acoplada dinamicamente a uma classe permitindo assim um mecanismo de extensão de comportamentos mais flexível que a herança.
 - k. Na camada de lógica de negócio é possível enviar pedaços de strings SQL para a camada de acesso a dados executar, facilitando dessa forma o desenvolvimento da camada de acesso a dados como um componente genérico.
 - l. Por uma questão de facilidade de codificação devemos sempre programar para uma classe específica e não para uma interface genérica.
 - m. Uma das aplicações do padrão *Two-Step Transform* é permitir que a mesma aplicação possua diferentes *layouts* de ecrã consoante o tipo de utilizador.
 - n. O padrão *Lazy Load* deve ser usado sempre que se usa um *Domain Model* muito complexo para evitar carregar demasiados objectos para memória.

2. Descreva em dois ou três parágrafos como é que uma aplicação de gestão de contratos de crédito poderia tirar partido do padrão *Façade* (6 valores).

Componente Prática

40% da nota da prova escrita. Nota mínima de 8 valores

3. Para o seguinte problema:

- Elabore esboços de ecrãs a utilizar para cada use case (2 valores)
- Identifique as operações de negócio necessárias para implementar cada um desses use cases; indique o nome, parâmetros com tipo e tipo de retorno para cada operação. No caso de usar DataSets indique qual a estrutura de tabelas e registos em cada DataSet (2 valores).
- Indique qual o padrão de lógica de negócio e de acesso a dados que utilizaria para implementar o sistema. Justifique a sua escolha (2 valores).
- Elabore o diagrama de sequência para implementar o use case “requisitar serviço” (4 valores).
- Elabore o(s) diagrama(s) de classes indicando as operações (com assinatura completa) e atributos (se existentes) nas suas classes (5 valores)
- Implemente o código (apenas da lógica de negócio e de acesso a dados) correspondente ao diagrama de sequência da alínea d) (5 valores)

A sua empresa decidiu fazer uma aplicação web para os clientes requisitarem serviços técnicos. Cada cliente tem acesso ao seu *portfolio* de produtos e pode requisitar um serviço técnico (de uma lista predefinida). No *backoffice*, um utilizador fará a atribuição dos pedidos requisitados a um dos técnicos disponíveis. A informação desses pedidos será depois utilizada pela contabilidade para facturar ao cliente.

