PROCESSO SELETIVO – 02/2026

Área de Conhecimento: Engenharia de Software e Banco de Dados

PROVA ESCRITA – PADRÃO DE RESPOSTA

Questão 1 (2,5 pontos): Modelos de Processo de Software

Cenário de Negócio: *Nossa empresa está iniciando o desenvolvimento de uma plataforma de Open Banking que precisa estar operacional em 90 dias para cumprir uma nova regulamentação governamental. O escopo inicial é vasto, mas as especificações técnicas da API do governo continuam sendo atualizadas semanalmente. Se não entregarmos o "cerne" da integração no prazo, a empresa sofrerá multas diárias. No entanto, se o sistema apresentar falhas de segurança nos dados dos usuários, a reputação da marca será comprometida permanentemente.*

Escreva um parecer técnico recomendando a estratégia de processo de desenvolvimento (ciclo de vida) mais adequada para este desafio. Em seu texto, você deve obrigatoriamente abordar os seguintes pontos:

1. **Análise Comparativa:** Discorra sobre os benefícios e riscos de um modelo preditivo (orientado a planos, como o Cascata) versus um modelo adaptativo (orientado a mudanças, como os Ágeis) para este cenário específico.
2. **Justificativa de Escolha:** Selecione a abordagem que você considera mais eficiente para mitigar os riscos de atraso e de instabilidade de requisitos citados no texto.
3. **Aplicação Prática:** Cite e explique como práticas da abordagem escolhida no item acima (item 2) poderiam ser utilizadas para garantir a qualidade técnica (segurança) e a entrega dentro do prazo de 90 dias.

Resposta Questão 1:

Esta resposta baseia-se nos conceitos de **Processos de Software** apresentados por **Marco Tulio Valente** no **Capítulo 2** do livro *Engenharia de Software Moderna*.

1. Análise Comparativa:

- ✓ **Modelo Preditivo (Cascata):** Segundo Valente, o modelo cascata é linear e rígido. Embora facilite o controle de cronograma em projetos com requisitos estáveis, no cenário de *Open Banking* ele seria ineficiente. A natureza "sequencial" impediria que a equipe voltasse para ajustar a arquitetura de segurança conforme as novas especificações da API do governo surgissem.
- ✓ **Modelo Adaptativo (Ágil):** Os métodos ágeis são recomendados quando o custo de mudança é controlado e a incerteza de requisitos é alta. A vantagem aqui é o feedback constante e a redução do risco de entregar algo que não atende à regulamentação final.

2. Justificativa da Escolha (Foco em Agilidade): A recomendação ideal é um **Processo Ágil (Iterativo e Incremental)**. Conforme discutido no Capítulo 2 de *Engenharia de Software Moderna*, o desenvolvimento incremental permite construir o sistema em fatias. Diante da volatilidade da API governamental, a equipe

pode focar em "fatias" estáveis (como autenticação e segurança) enquanto os requisitos externos mudam, garantindo que o núcleo do sistema esteja pronto no 90º dia.

3. Exemplos de Práticas e Métodos: O candidato deve demonstrar conhecimento de práticas que endereçam os problemas do cenário:

- **Scrum:** Uso de Sprints para gerenciar o prazo de 90 dias através de metas semanais e re-priorização do *Backlog* pelo Product Owner conforme as atualizações do governo.
- **XP (Extreme Programming):** Adoção de **TDD (Test Driven Development)** para garantir a segurança dos dados. Como Valente destaca, o XP foca em qualidade técnica; testes automatizados protegem o código contra regressões durante as mudanças semanais de requisitos.
- **Kanban:** Implementação de limites de trabalho (WIP) para garantir que a equipe finalize as integrações da API antes de iniciar novas, evitando gargalos de homologação.

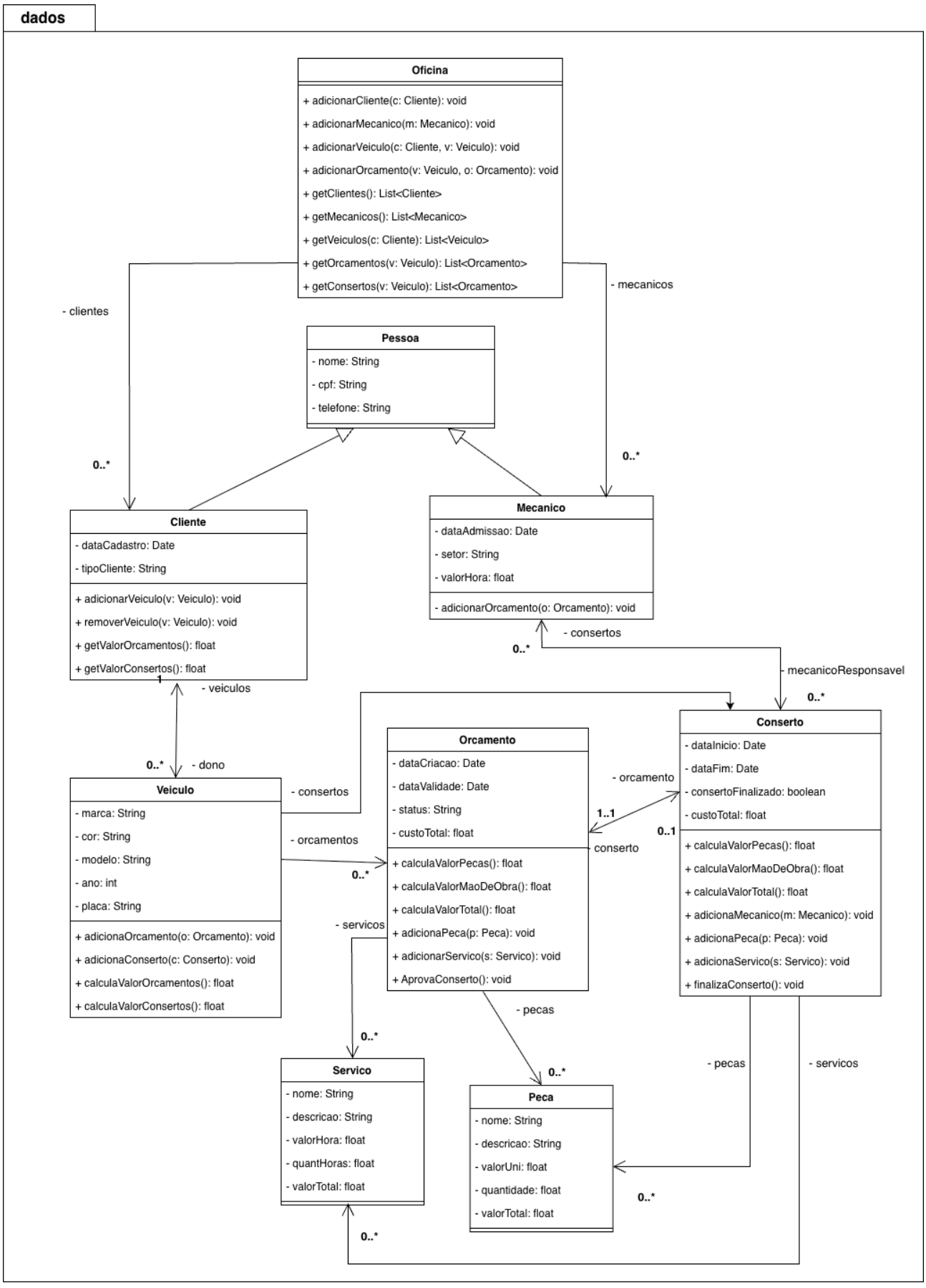
<u>Questão 2 (2,5 pontos)</u>: **Modelagem UML**

Modele o Diagrama de Classes UML que contemple as classes e relacionamentos entre elas para a seguinte especificação de requisitos de dados:

- Aplicação: Sistema de gerenciamento de oficinas mecânicas
- O sistema faz o gerenciamento dos orçamentos e consertos realizados pela oficina mecânica. Portanto, ele permite que sejam cadastrados orçamentos e consertos para um veículo de um cliente, que incluem um conjunto de peças e serviços, e um mecânico responsável.
- O sistema contém o cadastro dos Mecânicos da oficina e de seus Clientes, sendo que cada cliente pode ter um ou mais Veículos.
- Os Clientes e Mecânicos são generalizados em uma classe chamada Pessoa, que contém os atributos comuns entre eles, tais como: nome, CPF e telefone.
- Cada Mecânico tem os atributos data de admissão, setor e valor da hora de trabalho.
- Os Clientes têm como atributos a data de cadastro, o tipo de cliente (premium, ouro e prata) e um ou mais Veículos que são descritos pelas seguintes características: marca, cor, modelo, ano e placa.
- Cada veículo está relacionado com o cliente que é seu dono.
- Os veículos têm um ou mais Orçamentos que incluem a lista de Peças e Serviços a serem realizados. Cada orçamento contém as seguintes informações: data de criação, data de validade, status (aprovado, reprovado, pendente) e valor total.
- Cada Orçamento tem uma lista de Peças e Serviços associados.
- Cada Peça contém as seguintes informações: nome, descrição, valor unitário, quantidade e valor total.
- Cada Serviço contém: nome, descrição, valor da hora, quantidade de horas e valor total.
- Todo Orçamento aprovado está associado a um Conserto que tem a lista de Peças efetivamente trocadas e Serviços efetivamente realizados, além de data de início, data de finalização, custo total, status finalizado e a relação com os mecânicos que realizaram o conserto.
- A classe Oficina deve externalizar todas as funcionalidades que o sistema de gerenciamento de orçamentos tem, incluindo: cadastros e o retorno de todas as classes, assim como do somatório total dos consertos realizados pela oficina, por cliente e por veículo.
- Todas as classes devem ter métodos responsáveis para realizar a associação entre objetos desta classe e das classes com as quais ela se relaciona. Por exemplo, a classe Cliente deve ter um método chamado cadastro de Orçamento para associar os Orçamentos a um Cliente.

Resposta Questão 2:

O formalismo utilizado no diagrama abaixo é baseado na conceituação UML de LARMAN (2007).



<u>Questão 3 (2,5 pontos)</u>: **Engenharia Reversa de Bancos de Dados Relacionais**

Dado o banco de dados relacional abaixo, descreva quais são os pontos de redundância de dados e, em seguida, forneça um esquema relacional que não permita tais redundâncias. O esquema relacional deve ser fornecido como uma lista de tabelas. Para cada tabela, apresentar os nomes de seus respectivos campos entre parêntesis. Os campos que pertencem às chaves primárias devem ser pré-fixados com #. Para chaves estrangeiras, o prefixo & deve ser utilizado para os campos em questão. Indique as restrições de integridade referencial para as chaves estrangeiras por extenso. Campos opcionais devem estar entre colchetes ([]).

Animais									
nomeanimal	**codespecie**	**codanimal**	**codanimalpai**	**codanimalmae**	**dtnascanimal**	**nomeanimalpai**	**nomeanimalmae**	**expectativaespecie**	**nomeespecie**
Glória	1	1	NULL	NULL	05/05/2010	NULL	NULL	10	Hipopótamo
Vermelho	1	2	NULL	NULL	05/08/2009	NULL	NULL	10	Hipopótamo
Juzz	1	3	2	1	07/05/2015	vermelho	gloria	10	Hipopótamo
Opalala	1	4	2	1	04/01/2018	vermelho	gloria	10	Hipopótamo
Helio	1	12	3	1	01/03/2018	Juzzzz	Glória	10	Hipopótamo
Branquinho	2	5	NULL	NULL	05/01/2017	NULL	NULL	3	Coelho
Quick	2	6	NULL	NULL	31/12/2016	NULL	NULL	3	Coelho
Morango	2	7	5	6	20/12/2017	branquinho	QHICK	3	Coelho
Chocolate	2	8	5	6	20/12/2017	branquinho	QHICK	3	Coelho
Salti	2	9	5	6	20/12/2017	branquinho	QHICK	3	Coelho
Jazz	2	10	5	6	28/02/2018	branquinho	QHICK	3	Coelho
Guto	2	11	5	6	28/02/2018	branquinho	QHICK	3	Coelho
Verduz	2	13	8	7	01/03/2018	Chocolate	Morango	3	Coelho

Resposta Questão 3:

Segundo HEUSER (2001 – Capítulo 6), uma das aplicações da engenharia reversa de bancos de dados relacionais é avaliar possíveis dados replicados em um banco de dados já estabelecido, e posteriormente, corrigir seu esquema. Um dado é dito replicado quando há uma repetição de valores para o campo respectivo da relação, onde é possível identificar uma dependência funcional deste campo para algum outro campo desta mesma relação.

Na relação Animais fornecida por esta questão é possível identificar os seguintes campos com valores replicados:

- nomeespecie repete sempre os mesmos valores para tuplas com o mesmo valor de codespecie (dependência funcional entre codespecie e nomeespecie)

- expectativaespecie repete sempre os mesmos valores para tuplas com o mesmo valor de codespecie (dependência funcional entre codespecie e expectativaespecie)

- nomeanimalpai repete sempre os mesmos valores para tuplas com o mesmo valor de codanimalpai (dependência funcional entre codanimalpai e nomeanimalpai).

- nomeanimalmae repete sempre os mesmos valores para tuplas com o mesmo valor de codanimalmae (dependência funcional entre codanimalmae e nomeanimalmae).

Logo, dado os pontos de replicação e suas dependências, BATINI et al (1992) orienta que campos com valores replicados devem ser preferencialmente acomodados em novas tabelas. No caso de nomeespecie e expectativaespecie que dependem de codespecie, o correto então é gerar uma tabela espécie contendo todos estes 3 campos. Assim a tabela animais manteria apenas o codespecie, e este como chave estrangeira apontando para uma nova tabela:

Especies (#codespecie, nomeespecie, expectativaespecie)

Animais (nomeanimal, &codespecie, #codanimal, [codanimalpai], [codanimalmae], dtnascanimal, [nomeanimalpai], [nomeanimalmae])

No caso de nomeanimalpai e nomeanimalmae, como dito anteriormente, ambos são dependentes dos seus respectivos códigos (codanimalpai e codanimalmae). No entanto, é possível verificar que os valores de codanimalpai e codanimalmae correspondem a valores praticados no campo codanimal desta mesma tabela, mas em tuplas diferentes. Por exemplo, a terceira tupla da tabela tem como pai o codanimalpai=2 e nomeanimalpai='vermelho', dados que correspondem a segunda tupla desta mesma tabela com codanimal=2 e nomeanimal='Vermelho'. Interessante observar que o mesmo comportamento se repete para a mãe. Além disso, identifica-se claramente uma anomalia na inserção do dado replicado na terceira tupla que coloca o nome do pai com todas as letras em minúsculo. Logo, pais e mães são animais e a replicação reportada pode ser corrigida simplesmente eliminando os campos para nomes de pais e mães, e tornando seus códigos como chaves estrangeiras apontando para a própria relação no campo codanimal. O esquema final fica da seguinte forma:

Especies (#codespecie, nomeespecie, expectativaespecie)
Animais (nomeanimal, &codespecie, #codanimal, [&codanimalpai], [&codanimalmae], dtnascanimal)

Restrições de integridade referencial:
- Animais.codespecie aponta para Especies.codespecie
- Animais.codanimalpai aponta para Animais.codanimal
- Animais.codanimalmae aponta para Animais.codanimal

<u>Questão 4 (2,5 pontos)</u>: **Projeto de Esquemas de Dados**

Descreva detalhadamente as fases clássicas do projeto de um esquema de banco de dados e explique como cada fase se integra ao ciclo de vida de desenvolvimento de software, destacando as dependências e sincronizações necessárias entre elas.

Resposta Questão 4:

Fases do Projeto de um Esquema de Banco de Dados

Segundo ELMASRI e NAVATHE (2011), o projeto de esquemas de banco de dados consiste nas seguintes fases:

1. Análise de Requisitos
Nesta fase inicial, são identificadas e documentadas as necessidades de informação do sistema:

Atividades principais:
Levantamento de requisitos de dados com stakeholders
Identificação de entidades, atributos e relacionamentos necessários
Definição de regras de negócio e restrições
Análise de volume de dados e requisitos de desempenho
Produtos: Documento de requisitos de dados, casos de uso, especificações funcionais

2. Projeto Conceitual
Criação de um modelo abstrato e independente de tecnologia:

Atividades principais:
Modelagem através do Modelo Entidade-Relacionamento (MER)
Identificação de entidades, atributos e relacionamentos
Definição de cardinalidades e atributos identificadores
Validação com usuários e analistas de negócio
Produtos: Diagrama Entidade-Relacionamento (DER)

3. Projeto Lógico
Transformação do modelo conceitual em um esquema lógico:

Atividades principais:
Conversão do DER para modelo relacional (tabelas)
Normalização (1FN, 2FN, 3FN, BCNF)
Definição de chaves primárias e estrangeiras
Especificação de índices e restrições de integridade
Otimização do modelo para consultas esperadas
Produtos: Esquema relacional normalizado, dicionário de dados

4. Projeto Físico
Implementação do esquema no SGBD específico:

Atividades principais:
Escolha do SGBD (PostgreSQL, MySQL, Oracle, etc.)
Definição de tipos de dados específicos
Criação de índices, partições e clusters
Configuração de espaço de armazenamento
Definição de estratégias de backup e recuperação
Otimização para o hardware disponível
Produtos: Scripts SQL (DDL), esquema físico implementado

5. Implementação e Carga de Dados
Criação efetiva do banco e população inicial:

Atividades principais:
Execução dos scripts DDL
Desenvolvimento de procedures, triggers e views
Carga de dados iniciais ou migração
Criação de usuários e definição de permissões
Produtos: Banco de dados funcional com dados de teste/produção

6. Teste e Validação
Verificação da qualidade e adequação do banco:

Atividades principais:
Testes de integridade referencial
Testes de desempenho (queries, carga)
Validação de regras de negócio
Testes de segurança e controle de acesso
Produtos: Relatórios de teste, ajustes e correções

7. Manutenção e Evolução (tuning)
Monitoramento e ajustes contínuos:

Atividades principais:
Monitoramento de desempenho
Ajuste de índices e queries
Adição de novas tabelas/campos conforme evolução do sistema
Refatoração e desnormalização quando necessário
Produtos: Versões atualizadas do esquema, documentação de mudanças

Fase do BD	Fase do Software	Integração
Análise de Requisitos	Levantamento de Requisitos	Requisitos de dados derivam dos requisitos funcionais do sistema
Projeto Conceitual	Análise de Sistemas	MER alinha-se ao modelo de domínio e casos de uso
Projeto Lógico	Design/Arquitetura	Modelo lógico define estrutura para camada de persistência
Projeto Físico	Design Detalhado	Decisões técnicas do BD acompanham escolhas de tecnologia
Implementação	Codificação	Criação do BD paralelamente ao código da aplicação

Teste e Validação	Testes	Testes de BD integrados aos testes de sistema e integração
Manutenção	Manutenção/Evolução	Mudanças no BD seguem o ciclo de manutenção do software

Carla Diacui Medeiros Berkenbrock

Rebeca Schroeder Freitas

Fabiano Baldo

Presidente da Banca Examinadora

Assinaturas do documento

Código para verificação: **YBD2G268**

Este documento foi assinado digitalmente pelos seguintes signatários nas datas indicadas:

FABIANO BALDO (CPF: 028.XXX.209-XX) em 09/02/2026 às 09:53:27
Emitido por: "SGP-e", emitido em 30/03/2018 - 12:47:06 e válido até 30/03/2118 - 12:47:06.
(Assinatura do sistema)

REBECA SCHROEDER (CPF: 036.XXX.099-XX) em 09/02/2026 às 11:36:39
Emitido por: "SGP-e", emitido em 13/07/2018 - 14:59:17 e válido até 13/07/2118 - 14:59:17.
(Assinatura do sistema)

Para verificar a autenticidade desta cópia, acesse o link https://portal.sgpe.sea.sc.gov.br/portal-externo/conferencia-documento/VURFU0NfMTIwMjJfMDAwMDI4MTVfMjgxNl8yMDI2X1lCRDJHMjY4 ou o site https://portal.sgpe.sea.sc.gov.br/portal-externo e informe o processo **UDESC 00002815/2026** e o código **YBD2G268** ou aponte a câmera para o QR Code presente nesta página para realizar a conferência.