PROCESSO SELETIVO – 05/2026

Área de Conhecimento: Engenharia de Software

PROVA ESCRITA – PADRÃO DE RESPOSTA

QUESTÃO 1: Engenharia de Requisitos

Com base em Valente (2020), no Capítulo 3 sobre Requisitos do livro Engenharia de Software Moderna:
(a) Quando a validação de um MVP indica que a hipótese de negócio inicial falhou, restam duas alternativas: desistir do empreendimento ou realizar um pivô. Pivotar significa mudar a estratégia e a visão original do produto (redefinindo requisitos, mercado-alvo ou modelo de operação), sem descartar o aprendizado validado que foi obtido com o MVP anterior.
* Proposta de Pivôs para o Sistema de Caronas: (1) Em vez de tentar atender toda a comunidade universitária da UDESC de forma genérica, o projeto pode restringir o escopo para os estudantes de apenas um curso específico ou centro acadêmico (ex.: Centro de Ciências Tecnológicas - CCT). Esse nicho compartilha horários de aula e trajetos muito mais homogêneos, reduzindo a dispersão das rotas.(2) Migrar a aplicação da ideia e testar o mesmo modelo de caronas com alunos de outra instituição de ensino da cidade que enfrente problemas críticos de transporte público e estacionamento (3) Caso a carona diária se mostre inviável pela incompatibilidade de horários contínuos, a plataforma pode pivotar para organizar a mobilidade e o fretamento pontual voltados para eventos acadêmicos, congressos e jogos universitários.
(b) as Métricas de Vaidade São dados superficiais que afagam o ego dos idealizadores, mas não refletem a real sustentabilidade ou o uso do sistema (ex.: número de visualizações na página do formulário de cadastro). Já métricas Acionáveis: São dados concretos que revelam o comportamento real do usuário e fornecem subsídios claros para tomar decisões sobre o futuro do produto. Como métricas Acionáveis Recomendadas para o MVP: (1)Taxa de Conversão/Ativação (Match Efetivo): Percentual de solicitações de carona preenchidas que efetivamente resultaram em uma viagem realizada e confirmada entre motorista e caroneiro. (2)Taxa de Retenção (Uso Recorrente): Percentual de usuários (motoristas e passageiros) que voltam a utilizar o grupo/sistema para agendar novas caronas nas semanas subsequentes à primeira utilização.

QUESTÃO 2: Processos de Software

Proposta de Solução baseada nos conceitos de Processos de Software apresentados por Valente (2020), Capítulo 2, do livro Engenharia de Software Moderna

(a) Modelos Preditivos (Ex.: Waterfall / Cascata): Adotam uma abordagem sequencial e linear, em que as fases do projeto (Levantamento de Requisitos, Análise, Projeto, Implementação, Testes e Implantação) ocorrem de forma rígida e encadeada. Vantagens: Proporcionam alta previsibilidade inicial de escopo, custo e prazo, além de gerar documentação detalhada antes do início da codificação. Riscos: Apresentam baixa flexibilidade a mudanças. Se os requisitos mudarem ou forem mal compreendidos no início, o custo de alteração nas fases finais é elevado, aumentando o risco de entregar um software que não atende às necessidades reais do cliente. Modelos Adaptativos (Ex.: Métodos Ágeis): Adotam um

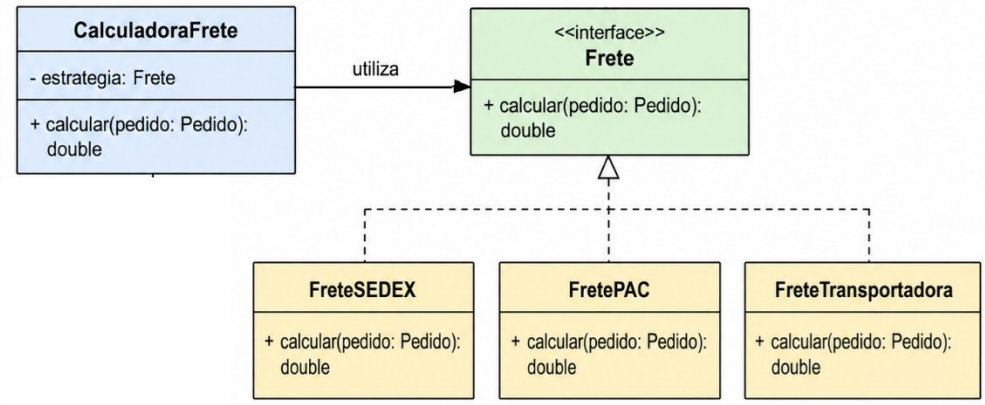
ciclo de vida iterativo e incremental, no qual o software é desenvolvido e entregue em pequenos módulos funcionais a cada ciclo curto. Vantagens: Priorizam a capacidade de resposta a mudanças, a entrega contínua de valor ao cliente e o feedback precoce. Riscos: Exigem alto engajamento dos stakeholders e disciplinas de engenharia rigorosas; sem uma boa gestão do backlog, há risco de "desvio de escopo" (scope creep) ou dificuldade no controle de estimativas de longo prazo.

(b) O manifesto e a filosofia ágil fundamentam-se no desenvolvimento incremental, na colaboração contínua com o cliente e na aceitação da mudança como um fator natural no desenvolvimento de software. Framework Scrum: O Scrum organiza o desenvolvimento em iterações de duração fixa (sprints), geralmente entre duas a quatro semanas. Cada sprint possui um time-box rígido para planejar, executar e revisar um conjunto de funcionalidades. O Product Backlog é uma lista dinâmica e priorizada de todos os requisitos e melhorias desejadas no produto, gerenciada pelo Product Owner. No início de cada sprint, seleciona-se um subconjunto desse backlog (Sprint Backlog), cujo escopo permanece protegido contra alterações durante a iteração para assegurar o foco do time.

(c) Acompanhamento Contínuo e Rituais: A Daily Scrum (reunião diária) promove a sincronização da equipe, a transparência do progresso e a identificação precoce de impedimentos que possam comprometer a iteração. Os papéis bem definidos, como o Product Owner (visão do produto e priorização) e o Scrum Master (facilitação do processo e remoção de bloqueios), ajudam na fluidez e a disciplina do fluxo de trabalho. Definição de Pronto (Definition of Done - DoD) é um acordo formal entre a equipe sobre os critérios mínimos de qualidade que um incremento de software deve atender para ser considerado "pronto". Inclui tipicamente práticas como a escrita e execução de testes automatizados, revisão de código (code review) e aprovação em ambientes de integração. A DoD atua como um mecanismo direto de garantia de qualidade, impedindo a inclusão de débitos técnicos no sistema e garantindo a entrega de um produto estável ao final de cada iteração.

QUESTÃO 3: Modelagem de Sistemas

Segundo descrito pelo Capítulo 6 de Valente (2020), o padrão comportamental *Strategy* pode ser adotado para abrir uma classe para extensões, mas sem alterar seu código fonte. Ao aplicar o padrão *Strategy* neste caso, cada forma de cálculo passaria a ser encapsulada em uma classe própria, todas implementando uma mesma interface. A `CalculadoraFrete` passaria a trabalhar apenas com essa interface, sem conhecer as implementações concretas. Um diagrama de classes possível adotando o *Strategy* é apresentado a seguir:



Neste caso, as classes concretas que implementam a interface *Frete* conteriam a forma de cálculo do seu tipo de frete. Já a implementação do método *calcular* em *CalculadoraFrete* conteria uma chamada ao método calcular da instância de *Frete* armazenada por ela. Desta forma *CalculadoraFrete* deixa de conhecer os detalhes do cálculo de cada forma de frete, e delega o cálculo a classe concreta respectiva a instância que esta mantém. Novas formas de cálculo podem ser futuramente incluídas apenas criando classes concretas que implementam *Frete*.

QUESTÃO 4: Arquitetura de Software Orientada a Serviços

Com base em Valente (2020), Capítulo 7 seção 4, sobre Arquiteturas de Microsserviços, seguem respostas aos questionamentos:

(a) Principais vantagens do desenvolvimento com arquitetura de microsserviços

Segundo o autor, a primeira vantagem é permitir a evolução mais rápida e independente do sistema, pois cada time passa a ter seu próprio regime de liberação de novas releases, resolvendo o gargalo entre o desenvolvimento ágil e a entrada em produção burocrática típica dos monolitos. A segunda vantagem é a escalabilidade em granularidade mais fina: enquanto a escalabilidade horizontal de um monolito exige replicar instâncias idênticas do sistema inteiro, na arquitetura de microsserviços é possível replicar apenas os serviços diretamente responsáveis pelos problemas de performance — como ilustra o exemplo em que seis instâncias do serviço M1 são disponibilizadas em um novo servidor. A terceira vantagem é a heterogeneidade tecnológica: como os microsserviços são autônomos e independentes, podem ser implementados em linguagens, frameworks e bancos de dados distintos, a exemplo do serviço de cadastro de clientes em Java com banco relacional e do serviço de recomendação em Python com banco NoSQL. A quarta vantagem é a possibilidade de falhas parciais: ao contrário do monolito, em que a queda do banco de dados derruba todos os serviços, a indisponibilidade de um microsserviço (como o de recomendações) não impede que os clientes continuem pesquisando produtos e realizando compras — a área de recomendações apenas fica desabilitada, vazia ou ausente. O autor acrescenta ainda que essa arquitetura tornou-se possível graças às plataformas de computação em nuvem, que permitem alugar máquinas virtuais pagas por hora, facilitando a escalabilidade horizontal dos serviços.

(b) Situações em que a arquitetura orientada a microsserviços não deve ser utilizada

O autor ressalta que a arquitetura de microsserviços é mais complexa do que a monolítica, pois, por construção, os microsserviços são processos independentes e dão origem a sistemas distribuídos. Assim, sua utilização impõe o enfrentamento de todos os desafios inerentes a sistemas distribuídos: a complexidade, já que a comunicação entre módulos em máquinas diferentes deixa de ser por chamadas de método e passa a exigir protocolos de comunicação, como HTTP/REST, demandando dos desenvolvedores o domínio de tecnologias de comunicação em redes; a latência,

pois a comunicação entre microsserviços envolve um atraso maior, sobretudo quando os serviços estão em máquinas distantes, gerando um custo de comunicação não desprezível; e as transações distribuídas, uma vez que a autonomia de dados de cada microsserviço dificulta garantir que operações sobre dois ou mais bancos de dados sejam atômicas — ou executam com sucesso em todos ou falham —, podendo exigir protocolos como o two-phase commit. Portanto, na perspectiva do autor, a arquitetura não deve ser adotada quando esses custos inerentes aos sistemas distribuídos — complexidade de comunicação, latência e dificuldade de garantir atomicidade transacional — tornarem-se obstáculos desfavoráveis ao sistema em questão.

Fabiano Baldo

Rebeca Schroeder Freitas

Carla Diacui Medeiros Berkenbrock

Presidente da Banca Examinadora

Assinaturas do documento

Código para verificação: **ZB34O90D**

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

CARLA DIACUI MEDEIROS BERKENBROCK (CPF: 025.XXX.559-XX) em 17/08/2026 às 15:12:48
Emitido por: "SGP-e", emitido em 30/03/2018 - 12:39:40 e válido até 30/03/2118 - 12:39:40.
(Assinatura do sistema)

Para verificar a autenticidade desta cópia, acesse o link https://portal.sgpe.sea.sc.gov.br/portal-externo/conferencia-documento/VURFU0NfMTIwMjJfMDAwMzE2NTBfMzE2NTZfMjAyNl9aQjM0TzkwRA== ou o site https://portal.sgpe.sea.sc.gov.br/portal-externo e informe o processo **UDESC 00031650/2026** e o código **ZB34O90D** ou aponte a câmera para o QR Code presente nesta página para realizar a conferência.