Lista de Exercícios – Fase de Projeto do Ciclo de Vida de Software

Questões Teóricas (1 a 5)

1. **Explique a principal diferença entre as fases de análise e projeto do ciclo de vida de software.**
   Dê exemplos de atividades que caracterizam cada uma delas.

2. **Quem são os principais stakeholders envolvidos na fase de projeto e quais são suas responsabilidades?**
   Cite ao menos três e descreva brevemente suas funções.

3. **Por que o design da arquitetura é considerado a atividade mais crucial da fase de projeto?**
   Inclua na resposta o impacto de uma má decisão arquitetural.

4. **Explique o papel dos diagramas UML na fase de projeto e cite três tipos de diagramas com seus respectivos objetivos.**

5. **Liste três benefícios diretos de um bom projeto de software para as fases posteriores do ciclo de vida.**

Questões Práticas (6 a 10)

6. **Escolha de Diagramas UML:**
   Imagine que você está participando do projeto de um sistema de controle de pedidos para uma pizzaria.
   Indique **quais diagramas UML** você usaria para representar:
   a) a estrutura das classes e seus relacionamentos;
   b) a sequência de eventos entre cliente, atendente e sistema;
   c) os módulos e conexões do sistema.
   Justifique brevemente o uso de cada um.

7. **Modelagem de Dados:**
   Desenhe (ou descreva textualmente) um **Diagrama de Entidade-Relacionamento (DER)** para um sistema de biblioteca com as entidades: **Livro**, **Autor**, **Usuário** e **Empréstimo**.
   Mostre as relações entre elas.

8. **Design de Interface:**
   Elabore um **wireframe textual** (descrição simples) para a tela de login de um sistema. Inclua elementos básicos como campos de entrada, botões e mensagens.

9. **Padrões de Design:**
   Suponha que você precise garantir que uma classe responsável por gerenciar a conexão com o banco de dados seja criada apenas uma vez durante toda a execução do sistema.
   Qual **padrão de design** pode ser aplicado? Explique como ele funciona.

10. **Projeto Detalhado de Sistemas:**
    Considere que o sistema de uma clínica médica já teve sua arquitetura e modelagem definidas.
    Descreva quais informações e decisões devem constar no **projeto detalhado** desse sistema para que a equipe de desenvolvimento possa iniciar a implementação com segurança e clareza.

Dica:
As questões teóricas avaliam sua compreensão conceitual da fase de projeto, enquanto as práticas pedem a aplicação dos conceitos em situações reais de desenvolvimento.

Gabarito Comentado – Fase de Projeto do Ciclo de Vida de Software

Questões Teóricas

1. **Diferença entre Análise e Projeto**
    - **Análise:** define *o que* o sistema deve fazer — foco nos requisitos funcionais e não funcionais.
    - **Projeto:** define *como* o sistema será construído — foco em arquitetura, tecnologias e modelagem técnica.
      Comentário: na análise identificam-se problemas e necessidades; no projeto, planeja-se a solução técnica.
2. **Stakeholders da Fase de Projeto**
    - **Arquiteto de Software:** define a arquitetura e as decisões técnicas principais.
    - **Equipe de Desenvolvimento:** avalia a viabilidade técnica e propõe soluções.
    - **Gerente de Projeto:** garante aderência a cronograma e orçamento.
    - **Analista de Sistemas/Negócios:** mantém alinhamento entre o projeto técnico e os requisitos de negócio.
      Comentário: a colaboração entre esses papéis é essencial para evitar retrabalho e falhas de comunicação.
3. **Importância do Design Arquitetural**
    - Define a estrutura fundamental do sistema.
    - Impacta diretamente na escalabilidade, desempenho e manutenção.
    - **Erro arquitetural** pode gerar retrabalho, alto custo e sistemas difíceis de evoluir.
      Comentário: a arquitetura é o “esqueleto” do software — um erro nessa etapa compromete todo o desenvolvimento.
4. **Diagramas UML e seus Objetivos**
    - **Diagrama de Classe:** mostra estrutura estática — classes, atributos e relações.
    - **Diagrama de Sequência:** ilustra a troca de mensagens entre objetos ao longo do tempo.
    - **Diagrama de Componentes:** representa a organização e dependência entre módulos do sistema.
      Comentário: os diagramas facilitam a comunicação entre equipe técnica e não técnica e servem de base para a codificação.
5. **Benefícios de um Bom Projeto**
    - Evita retrabalho na programação.
    - Reduz custos, pois erros são corrigidos antes da codificação.
    - Facilita manutenção e evolução do sistema.
      Comentário: investir tempo no projeto é mais barato e seguro do que corrigir falhas durante ou após o desenvolvimento.

Questões Práticas

6. **Escolha de Diagramas UML**
   a) **Diagrama de Classe** – estrutura das classes, atributos e métodos.
   b) **Diagrama de Sequência** – interação entre cliente, atendente e sistema (pedido → preparo → entrega).
   c) **Diagrama de Componentes** – representação dos módulos: atendimento, pedidos, pagamentos etc.
   Comentário: a escolha adequada de diagramas permite visualizar o sistema de diferentes perspectivas (estática, dinâmica e estrutural).

7. **Modelagem de Dados – DER de Biblioteca**

   - **Livro** (id_livro, título, ano, id_autor)
   - **Autor** (id_autor, nome)
   - **Usuário** (id_usuario, nome, matrícula)
   - **Empréstimo** (id_emprestimo, id_usuario, id_livro, data_retirada, data_devolução)
     Relações:
   - Um **autor** escreve vários **livros** (1:N)
   - Um **usuário** pode ter vários **empréstimos** (1:N)
   - Um **livro** pode aparecer em vários **empréstimos**, mas um **empréstimo** refere-se a um livro específico (1:N).

   Comentário: a modelagem garante a integridade dos dados e facilita a consulta eficiente.

8. **Wireframe Textual da Tela de Login**

```
[Logo do Sistema]
-----------------------
Usuário: [__________]
Senha:   [__________]
[ Entrar ]
[ Esqueci minha senha ]
Mensagens: "Usuário ou senha inválidos" (se erro)
```

   Comentário: o wireframe ajuda a visualizar rapidamente a disposição dos elementos da interface antes do design detalhado.

9. **Padrão de Design – Singleton**

   - **Objetivo:** garantir que exista apenas uma instância de uma classe e fornecer um ponto global de acesso a ela.
   - **Exemplo:** classe `ConexaoBanco` com método estático `getInstance()` que retorna sempre a mesma instância.

   Comentário: o padrão Singleton evita múltiplas conexões desnecessárias e garante consistência.

10. **Projeto Detalhado de Sistemas**
    Deve conter:

    - **Descrição detalhada de cada módulo** (funções, entradas, saídas, restrições).
    - **Especificação de interfaces e APIs** (métodos, parâmetros, formatos de dados).
    - **Diagramas de classes e sequência refinados.**
    - **Regras de negócio específicas** e fluxos de exceção.
    - **Padrões de codificação e segurança.**
    - **Mapeamento para o banco de dados.**

    Comentário: o projeto detalhado serve como guia direto para os desenvolvedores, evitando ambiguidade e erros de interpretação.