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

Questões Teóricas

1. **Explique o papel da fase de implementação no ciclo de vida de software e sua relação com a fase de projeto.**
2. **O que é a revisão de código e por que ela é importante?**
3. **Compare os paradigmas de programação imperativo e orientado a objetos.**
4. **Qual a diferença entre um Framework e uma API?**
5. **Qual a função principal de um IDE e quais são seus componentes básicos?**

Questões Práticas

6. **Integração Contínua:** Explique como a prática de integração contínua (CI) ajuda a evitar problemas de integração e quais ferramentas podem ser usadas.

7. **Controle de Versão:** Dê um exemplo prático de como o **Git** pode ser usado em equipe para evitar perda de código e gerenciar alterações.
8. **Paradigma de Programação:** Escreva um pequeno exemplo (em pseudocódigo) que ilustre o paradigma *funcional*, e outro que ilustre o *orientado a objetos.*
9. **Escolha de Linguagem:** Indique qual linguagem seria mais adequada para cada cenário:
   a) Sistema bancário corporativo;
   b) Aplicativo web interativo;
   c) Software de controle de hardware embarcado.
10. **Implementação de Projeto:** Imagine que o projeto detalhado de um sistema CRM (gestão de clientes) está pronto. Quais passos práticos devem ser seguidos pela equipe de desenvolvimento durante a implementação?

Gabarito comentado – Fase de Implementação do Ciclo de Vida de Software

Questões Teóricas

6. **Explique o papel da fase de implementação no ciclo de vida de software e sua relação com a fase de projeto.**
   Gabarito comentado:
   A fase de implementação transforma o design técnico em código executável. Ela é guiada pelas especificações da fase de projeto, que define *como* o sistema deve ser construído. O projeto fornece os planos; a implementação, o produto funcional.
   Comentário: Uma boa implementação depende de um projeto bem estruturado e detalhado.

7. **O que é a revisão de código e por que ela é importante?**
   Gabarito comentado:
   É o processo em que um desenvolvedor revisa o código escrito por outro para detectar erros, inconsistências e oportunidades de melhoria.
   Comentário: Melhora a qualidade, reduz bugs e promove aprendizado coletivo dentro da equipe.

8. **Compare os paradigmas de programação imperativo e orientado a objetos.**
   Gabarito comentado:

   - **Imperativo:** foco em *como* fazer; sequência de instruções que modificam estados (ex.: C, Pascal).
   - **OO:** foco em *objetos* que representam entidades do mundo real, combinando dados e comportamentos (ex.: Java, Python).
     Comentário: O paradigma OO favorece modularidade e reutilização de código, enquanto o imperativo é mais direto, porém menos estruturado.

9. **Qual a diferença entre um Framework e uma API?**
   Gabarito comentado:

   - **Framework:** fornece uma estrutura de desenvolvimento completa, com padrões e ferramentas.
   - **API:** define como diferentes sistemas ou módulos se comunicam entre si.
     Comentário: O framework orienta *como* o código deve ser escrito; a API define *como* os sistemas se integram.

10. **Qual a função principal de um IDE e quais são seus componentes básicos?**
    Gabarito comentado:
    Um IDE integra diversas ferramentas para o desenvolvimento de software:

    - **Editor de código**,
    - **Compilador/interpretador**,
    - **Debugger**.
      Comentário: Ele centraliza o processo de codificação, teste e depuração, aumentando a produtividade.

Questões Práticas

6. **Integração Contínua:**
   Explique como a prática de integração contínua (CI) ajuda a evitar problemas de integração e quais ferramentas podem ser usadas.
   Gabarito comentado:
   CI integra automaticamente o código de todos os desenvolvedores várias vezes ao dia, detectando falhas cedo.
   Ferramentas: GitHub Actions, Jenkins, GitLab CI, CircleCI.
   Comentário: A CI mantém o sistema sempre testado e funcional, reduzindo conflitos e retrabalho.

7. **Controle de Versão:**
   Dê um exemplo prático de como o **Git** pode ser usado em equipe para evitar perda de código e gerenciar alterações.
   Gabarito comentado:
   Cada desenvolvedor trabalha em um *branch* separado. Após testar, faz um *merge* na *main branch* via *pull request*.
   Comentário: Isso permite histórico de versões e colaboração segura entre vários programadores.

8. **Paradigma de Programação:**
   Escreva um pequeno exemplo (em pseudocódigo) que ilustre o paradigma **funcional**, e outro que ilustre o **orientado a objetos**.
   Gabarito comentado:

   - **Funcional:**

     ```
     soma = (x, y) => x + y
     resultado = soma(3, 4)
     ```

   - **OO:**

     ```
     classe Calculadora:
         método soma(x, y):
             retornar x + y
     resultado = Calculadora().soma(3, 4)
     ```

   Comentário: O paradigma funcional evita estados mutáveis, enquanto o OO organiza o código em classes e objetos.

9. **Escolha de Linguagem:**
   Indique qual linguagem seria mais adequada para cada cenário:
   a) Sistema bancário corporativo;
   b) Aplicativo web interativo;
   c) Software de controle de hardware embarcado.
   Gabarito comentado:
   a) **Java** (robusto, seguro, multiplataforma)
   b) **JavaScript/TypeScript** (interatividade e flexibilidade no front-end e back-end com Node.js)
   c) **C ou C++** (baixo nível e alto desempenho)
   Comentário: Cada linguagem é escolhida com base no equilíbrio entre performance, portabilidade e complexidade.

10. **Implementação de Projeto:**
    Imagine que o projeto detalhado de um sistema CRM (gestão de clientes) está pronto. Quais passos práticos devem ser seguidos pela equipe de desenvolvimento durante a implementação?
    Gabarito comentado:

    - Configurar ambiente de desenvolvimento (IDE, repositório Git).
    - Criar estrutura inicial do código conforme o design.
    - Implementar módulos seguindo padrões de codificação.
    - Realizar revisões de código e testes automatizados.
    - Integrar continuamente o código ao sistema principal.
      Comentário: Essa sequência garante qualidade e rastreabilidade, transformando o design técnico em software funcional.

Resumo:
Esta lista cobre os aspectos fundamentais da fase de **implementação**, unindo teoria (conceitos, paradigmas, ferramentas) e prática (processos, técnicas, exemplos reais). O aluno é estimulado a compreender tanto *o que fazer* quanto *como fazer* para garantir qualidade e eficiência no desenvolvimento de software.