UniSENAI

Trilha 08

Gestão de Projetos e Metodologias Ágeis

Instruções para a melhor prática de Estudo

1. **Leia atentamente todo o conteúdo:** Antes de iniciar qualquer atividade, faça uma leitura detalhada do material fornecido na trilha, compreendendo os conceitos e os exemplos apresentados.

2. **Não se limite ao material da trilha:** Utilize o material da trilha como base, mas busque outros materiais de apoio, como livros, artigos acadêmicos, vídeos, e blogs especializados. Isso enriquecerá o entendimento sobre o tema.

3. **Explore a literatura:** Consulte livros e publicações reconhecidas na área, buscando expandir seu conhecimento além do que foi apresentado. A literatura acadêmica oferece uma base sólida para a compreensão de temas complexos.

4. **Realize todas as atividades propostas:** Conclua cada uma das atividades práticas e teóricas, garantindo que você esteja aplicando o conhecimento adquirido de maneira ativa.

5. **Evite o uso de Inteligência Artificial para resolução de atividades:** Utilize suas próprias habilidades e conhecimentos para resolver os exercícios. O aprendizado vem do esforço e da prática.

6. **Participe de debates:** Discuta os conteúdos estudados com professores, colegas e profissionais da área. O debate enriquece o entendimento e permite a troca de diferentes pontos de vista.

7. **Pratique regularmente:** Não deixe as atividades para a última hora. Pratique diariamente e revise o conteúdo com frequência para consolidar o aprendizado.

8. **Peça feedback:** Solicite o retorno dos professores sobre suas atividades e participe de discussões sobre os erros e acertos, utilizando o feedback para aprimorar suas habilidades.

Essas instruções são fundamentais para garantir um aprendizado profundo e eficaz ao longo das trilhas.

Gestão de Projetos e Metodologias Ágeis

1. SCRUM, Kanban e XP Aplicados ao Desenvolvimento de Sistemas

As metodologias ágeis são abordagens que focam na entrega rápida e contínua de valor ao cliente, adaptando-se a mudanças de requisitos ao longo do projeto. Três das mais conhecidas metodologias ágeis aplicadas ao desenvolvimento de sistemas são **SCRUM**, **Kanban** e **XP (Extreme Programming)**.

a) SCRUM

SCRUM é uma metodologia ágil focada na entrega incremental de funcionalidades. Um projeto é dividido em ciclos chamados de **sprints**, que normalmente duram de 2 a 4 semanas. Ao final de cada sprint, uma funcionalidade deve estar pronta para entrega.

- **Papéis no SCRUM**:
  - **Product Owner**: Define as prioridades e garante que a equipe está trabalhando no que é mais importante para o projeto.
  - **Scrum Master**: Facilita o processo e garante que a equipe siga as práticas do SCRUM.
  - **Time de Desenvolvimento**: Conjunto de profissionais que desenvolvem as funcionalidades.
- **Exemplo**: Em um projeto de desenvolvimento de um aplicativo de e-commerce, o SCRUM pode ser usado para organizar sprints que entregam funcionalidades específicas, como login de usuários em uma sprint e carrinho de compras na seguinte.

b) Kanban

Kanban é uma metodologia que foca na visualização do fluxo de trabalho, permitindo que a equipe acompanhe o andamento de cada tarefa. As tarefas são organizadas em um **quadro Kanban**, dividido em colunas como **"A Fazer"**, **"Em Progresso"** e **"Concluído"**. O objetivo é minimizar o trabalho em progresso, gerenciando o fluxo de forma contínua.

- **Principais características**:
  - Fluxo contínuo: Não há sprints definidos; as tarefas fluem conforme são finalizadas.
  - Limitação de Tarefas: O número de tarefas em andamento é limitado para evitar sobrecarga.
- **Exemplo**: No desenvolvimento de um sistema de gestão de estoques, o Kanban pode ser usado para organizar a equipe, visualizando as tarefas necessárias para criar o módulo de relatórios de estoque.

c) XP (Extreme Programming)

XP é uma metodologia ágil que prioriza boas práticas de desenvolvimento, como testes automáticos e **programação em par** (dois desenvolvedores trabalhando juntos no mesmo código). XP incentiva a entrega frequente de pequenas partes do sistema.

- **Práticas do XP**:

    - **Programação em Par**: Dois desenvolvedores escrevem o código juntos, aumentando a qualidade.
    - **Testes Contínuos**: O código é constantemente testado para evitar bugs.
    - **Refatoração**: Melhorar o código continuamente sem alterar sua funcionalidade.
- **Exemplo**: Em um projeto que envolve a criação de uma API, a programação em par pode ser usada para garantir que os endpoints sejam desenvolvidos de forma consistente, com testes automatizados para cada funcionalidade.

2. Ciclo de Vida de Projetos de Software

O ciclo de vida de um projeto de software inclui todas as fases pelas quais o projeto passa, desde a concepção até a entrega e manutenção. As principais fases incluem:

- **1. Planejamento**: Definir os objetivos do projeto, criar o escopo e identificar os recursos necessários.
- **2. Análise de Requisitos**: Levantar as necessidades do cliente e transformá-las em especificações técnicas para o software.
- **3. Design**: Projetar a arquitetura do software, suas interfaces e a forma como os componentes interagem.
- **4. Desenvolvimento**: Escrever o código, implementar as funcionalidades e integrar os componentes do sistema.
- **5. Testes**: Verificar se o software funciona conforme especificado, corrigindo falhas e garantindo a qualidade.
- **6. Entrega e Implantação**: Instalar o software no ambiente de produção e entregá-lo ao cliente.
- **7. Manutenção**: Corrigir bugs, atualizar funcionalidades e oferecer suporte ao sistema após a entrega.
- **Exemplo**: No desenvolvimento de um software de controle financeiro, o ciclo de vida começaria com a definição de requisitos como cadastro de despesas e receitas, seguido pelo design das telas, desenvolvimento dos módulos e, por fim, testes e implantação para os usuários finais.

3. Controle de Qualidade e Entregas Contínuas

O controle de qualidade garante que o software entregue esteja livre de erros e atende às expectativas do cliente. Práticas como **testes automatizados**, **integração contínua** e **entregas contínuas** são essenciais no desenvolvimento ágil.

- **Testes Automatizados**: Testes são executados automaticamente sempre que o código é atualizado, garantindo que novas alterações não causem problemas.
- **Integração Contínua (CI)**: Ferramentas como Jenkins ou GitLab CI são usadas para automatizar o processo de integração, compilação e teste do código sempre que uma alteração é feita.
- **Entrega Contínua (CD)**: O processo de entrega contínua automatiza a distribuição de versões estáveis do software para o cliente, garantindo que novos recursos e correções sejam lançados com frequência e segurança.

- **Exemplo**: No desenvolvimento de um sistema de e-commerce, a equipe pode usar integração contínua para garantir que cada novo código seja automaticamente testado e, quando aprovado, implantado em produção.

Lista de Exercícios de Fixação

1. Explique as diferenças entre SCRUM e Kanban. Em que situações seria mais adequado usar cada metodologia?
2. Descreva as responsabilidades de um Product Owner no SCRUM. Por que esse papel é importante no desenvolvimento de sistemas?
3. Explique como a prática de programação em par do XP pode melhorar a qualidade do código. Dê um exemplo de quando essa prática seria benéfica.
4. Liste e explique as fases do ciclo de vida de um projeto de software. Dê um exemplo prático para cada fase.
5. Crie um quadro Kanban simples com três colunas: 'A Fazer', 'Em Progresso' e 'Concluído'. Adicione pelo menos cinco tarefas para um projeto de desenvolvimento de um aplicativo móvel.
6. Descreva como a integração contínua pode ser usada para melhorar o fluxo de trabalho em um projeto ágil. Dê um exemplo prático.
7. Explique como os testes automatizados contribuem para o controle de qualidade no desenvolvimento de software.
8. Simule a organização de uma sprint no SCRUM para o desenvolvimento de um módulo de login de um sistema. Defina o objetivo da sprint, as tarefas e o tempo estimado.
9. Pesquise uma ferramenta de integração contínua e descreva como ela pode ser implementada em um projeto de software.
10. Explique como o ciclo de vida de um projeto de software pode ser adaptado ao uso de metodologias ágeis, como SCRUM e XP.