# DESAFIO

## ENGENHARIA E QUALIDADE DE SOFTWARE

*No contexto do planejamento ágil para o projeto em questão, considerando a natureza de uma startup e a gestão de equipes remotas, indico a metodologia Scrum.*

## 1 - RESPOSTA

## Minha justificativa é a seguinte:

**Scrum para Equipes Remotas**

- O Scrum, com suas cerimônias bem definidas (Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective) e papéis claros (Product Owner, Scrum Master, Time de Desenvolvimento), promove comunicação constante e transparência, elementos essenciais para equipes remotas.
- A cadência das Sprints garante entregas incrementais e feedback contínuo, minimizando a sensação de isolamento e mantendo todos alinhados aos objetivos do projeto.

**Flexibilidade e Adaptação**

- Startups operam em ambientes dinâmicos, onde os requisitos podem mudar rapidamente.
- O ciclo de inspeção e adaptação do Scrum permite ajustes ágeis a novas demandas e prioridades, sem comprometer a qualidade.

**Foco no Valor**

- O Product Owner define e prioriza o Backlog do Produto, assegurando que a equipe esteja sempre focada em funcionalidades que entreguem o maior valor ao negócio e aos usuários.

**Transparência e Confiança**

- As Daily Scrums e as Sprint Reviews aumentam a visibilidade do progresso para todos os envolvidos, incluindo stakeholders.
- Isso constrói confiança e facilita a colaboração, mesmo à distância.

**Melhoria Contínua**

- As Sprint Retrospectives são momentos cruciais para identificar gargalos, otimizar processos e melhorar a colaboração, resultando em um fluxo de trabalho mais eficiente ao longo do tempo.

*Embora eu reconheça que o **Kanban** é excelente para gerenciar o fluxo de trabalho, considero que o **Scrum** oferece uma estrutura mais robusta, especialmente para equipes remotas que estão iniciando e precisam coordenar ciclos de desenvolvimento bem definidos.*

# 2 - RESPOSTA

Na fase de levantamento de requisitos do projeto, considero que a escolha de dois diagramas UML é crucial para garantir uma comunicação eficaz com os stakeholders e assegurar a qualidade da solução:

## Diagrama de Casos de Uso

**Contribuição para a Comunicação com Stakeholders**

- O Diagrama de Casos de Uso é uma ferramenta essencial para comunicar a funcionalidade do sistema em uma linguagem acessível e alinhada ao negócio. Ele descreve o que o sistema fará do ponto de vista do usuário (ator), sem entrar em detalhes técnicos. Isso facilita a compreensão por stakeholders não técnicos, como gerentes de produto e clientes, ajudando a validar se as funcionalidades propostas atendem às suas necessidades.
- Além disso, as discussões geradas por este diagrama frequentemente revelam cenários e requisitos que poderiam passar despercebidos.

**Contribuição para a Qualidade da Solução**

- Ao mapear claramente as interações entre os usuários e o sistema, esse diagrama garante que todas as funcionalidades essenciais sejam contempladas. Ele serve como base para a criação de cenários de teste, assegurando que o sistema funcione conforme esperado.
- A clareza proporcionada pelo Diagrama de Casos de Uso também reduz ambiguidades nos requisitos, diminuindo a probabilidade de erros de desenvolvimento e retrabalho, o que contribui diretamente para a qualidade do software.

## Diagrama de Classes

**Contribuição para a Comunicação com Stakeholders**

- Embora seja mais técnico, o Diagrama de Classes pode ser apresentado de forma conceitual para comunicar a estrutura de dados e os principais componentes do sistema. Ele é útil para validar a compreensão do domínio do problema e a representação das entidades de negócio. Por exemplo, ao mostrar classes como "Projeto", "Usuário" e "Tarefa", com seus atributos principais, os stakeholders podem confirmar se as informações relevantes estão sendo consideradas corretamente.

**Contribuição para a Qualidade da Solução**

- O Diagrama de Classes é fundamental para a modelagem da estrutura do sistema, funcionando como um guia para a criação do banco de dados e para a arquitetura do código. Ao detalhar atributos, métodos e relacionamentos entre as classes, garante-se uma estrutura coesa, com alta coesão e baixo acoplamento. Isso facilita a manutenção, a extensibilidade e a reutilização do código. Além disso, a correta identificação de relacionamentos entre as classes ajuda a evitar redundâncias e a garantir a integridade do sistema, resultando em um software robusto e de alta qualidade.

# 3 - RESPOSTA

Para garantir uma excelente experiência ao usuário final, aplicaria as seguintes práticas relacionadas à qualidade de software e à usabilidade:

## 1. Testes Automatizados Contínuos (Qualidade)

Eu implementaria uma suíte abrangente de testes automatizados, incluindo testes unitários, de integração e de interface de usuário (UI). Esses testes seriam executados continuamente, utilizando Integração Contínua (CI), sempre que alterações fossem realizadas no código.

**Benefícios:**

- Essa prática detecta bugs precocemente no ciclo de desenvolvimento, assegura que novas funcionalidades não causem regressão, reduz o custo de correção de defeitos e aumenta a confiança na base de código.

- Em um contexto de equipes remotas, a automação de testes é ainda mais crucial, pois diminui a dependência de testes manuais e mantém o software em um estado constantemente pronto para implantação.

## 2. Prototipagem e Testes de Usabilidade com Usuários Reais (Usabilidade)

Eu criaria protótipos de baixa a alta fidelidade das principais interfaces e fluxos de trabalho e conduziria testes de usabilidade com usuários reais da plataforma.

**Benefícios:**

- Essa abordagem ajuda a identificar problemas de navegação, clareza e eficiência em estágios iniciais, antes de investir tempo significativo em desenvolvimento. Testes remotos, utilizando ferramentas de compartilhamento de tela e gravação, permitem coletar feedback valioso para iterar no design.
- Assim, eu garantiria que a interface fosse intuitiva e atendesse às expectativas do usuário final.

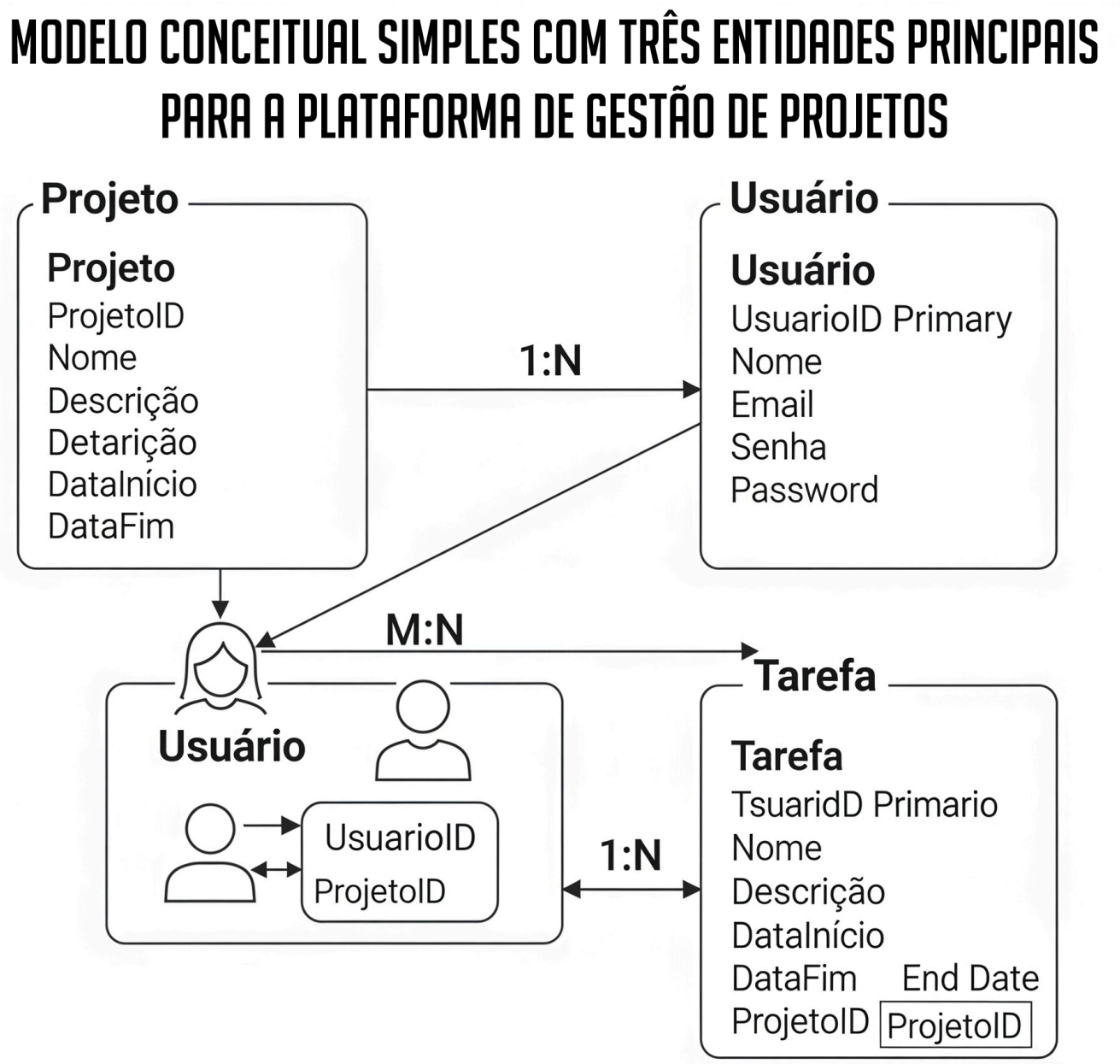
## 3. Design Responsivo e Acessibilidade (Usabilidade e Qualidade)

Eu projetaria a plataforma para ser totalmente responsiva, adaptando-se a diferentes dispositivos (desktop, tablet, mobile). Além disso, seguiria os princípios de acessibilidade (WCAG) para assegurar que a plataforma pudesse ser usada por pessoas com deficiência.

**Benefícios:**

- Um design responsivo permite que os usuários acessem a plataforma em qualquer dispositivo, aumentando a flexibilidade e a produtividade, especialmente em contextos variados.
- Já a acessibilidade amplia o alcance da solução, tornando-a inclusiva e usável por um público mais diverso.
- Ambas as práticas contribuem para uma experiência de usuário superior e para a qualidade geral do software.

# 4 - RESPOSTA






Para o projeto que estou desenvolvendo, eu defini as entidades, relacionamentos e a importância da normalização conforme descrito abaixo:

## ENTIDADES E ATRIBUTOS

- **1. Projeto**
  - Atributos:

        - id_projeto (Chave Primária)
        - nome_projeto
        - descricao
        - data_inicio
        - data_fim_prevista
        - status_projeto
    - Descrição: Representa um projeto que será gerenciado na plataforma.

- **2. Usuário**
    - Atributos:
        - id_usuario (Chave Primária)
        - nome
        - email
        - senha
        - cargo
    - Descrição: Representa um membro da equipe que utilizará a plataforma.

- **3. Tarefa**
    - Atributos:
        - id_tarefa (Chave Primária)
        - titulo
        - descricao
        - status (Ex: "A Fazer", "Em Andamento", "Concluído")
        - prioridade (Ex: "Baixa", "Média", "Alta")
        - data_criacao
        - data_conclusao
        - id_projeto (Chave Estrangeira)
        - id_usuario_responsavel (Chave Estrangeira)
    - Descrição: Representa uma unidade de trabalho dentro de um projeto.

- **4. Entidade Associativa "Participa"**
    - Atributos:
        - id_projeto (Chave Estrangeira)
        - id_usuario (Chave Estrangeira)
        - papel_no_projeto (Ex: "Gerente de Projeto", "Desenvolvedor", "Designer")
        - Chave Primária Composta: (id_projeto, id_usuario)
    - Descrição: Armazena as informações sobre a participação de um usuário em um projeto, incluindo o papel desempenhado.

# RELACIONAMENTOS

**1 - Projeto - Tarefa (1:N):**

- Um projeto pode ter muitas tarefas, mas cada tarefa pertence a apenas um projeto.
- Representado pela Chave Estrangeira id_projeto na tabela Tarefa.

**2 - Usuário - Tarefa (1:N):**

- Um usuário pode ser responsável por várias tarefas, mas cada tarefa tem apenas um usuário responsável.
- Representado pela Chave Estrangeira id_usuario_responsavel na tabela Tarefa.

**3 - Projeto - Usuário (M:N) através de "Participa":**

- Um projeto pode ter muitos usuários participando, e cada usuário pode estar associado a vários projetos.

# IMPORTÂNCIA DA NORMALIZAÇÃO

A normalização é fundamental para a organização eficiente dos dados no banco de dados, minimizando redundâncias e evitando anomalias.

- Redução de Redundância: Evita repetir informações como o nome de um usuário em múltiplas tarefas, centralizando os dados na tabela Usuário.
- Integridade dos Dados: Garante consistência, como a atualização do nome de um projeto em apenas um local (tabela Projeto).
- Flexibilidade e Extensibilidade: Facilita a inclusão de novos atributos ou relacionamentos, adaptando-se a mudanças futuras.
- Evita Anomalias:
  - Inserção: Impede a criação de tarefas sem um projeto válido associado.
  - Atualização: Evita inconsistências ao alterar dados duplicados.
  - Exclusão: Protege dados relacionados, evitando exclusões acidentais.

***Aluno: Rafael Marins Mendes***
***Curso: Análise e Desenvolvimento de Sistemas***
***Disciplina:  Análise e Projetos de Software***
***Professor: Wagner Junio Cordeiro***
***Data: 08/06/2025***