# Passo a Passo Completo para Análise de Projetos de Sistemas

## 1. Planejamento Inicial

- Como: Reúna os stakeholders para alinhar expectativas e definir as diretrizes do projeto.
- Inclui:
    - Definição da Visão e Escopo do Projeto:Descrever o objetivo geral do sistema (o "porquê" do projeto).
    - Identificação dos Stakeholders:Mapear todas as partes interessadas (usuários finais, clientes, equipe técnica, etc.).
    - Cronograma Inicial:Estabelecer prazos preliminares para cada fase do projeto.
    - Análise de Viabilidade:Avaliar a viabilidade técnica, econômica e operacional do projeto.
-

## 2. Coleta de Requisitos

- Como: Use técnicas como entrevistas, workshops, questionários e análise de documentos existentes.
- Inclui:
    - Levantamento de requisitos funcionais (o que o sistema deve fazer).
    - Levantamento de requisitos não funcionais (desempenho, segurança, usabilidade).
    - Identificação de restrições (tecnológicas, legais ou financeiras).
    - Criação de um documento inicial com os requisitos levantados.
-

## 3. Análise de Requisitos

- Como: Organize os requisitos coletados para garantir clareza e completude.
- Inclui:
    - Classificação dos Requisitos:Divida em categorias como "essenciais", "desejáveis" e "opcionais".
    - Validação com Stakeholders:Realize reuniões para confirmar se os requisitos atendem às expectativas.

    - Documentação Formal dos Requisitos:Crie um documento claro e padronizado para consulta futura.
-

## 4. Modelagem de Processos de Negócios

- Como: Utilize ferramentas como BPMN (Business Process Model and Notation) ou fluxogramas.
- Inclui:
    - Mapeamento dos processos atuais da organização.
    - Identificação de gargalos ou ineficiências nos processos existentes.
    - Proposta de melhorias nos processos que o sistema irá suportar.
-

## 5. Perfis UML

- Como: Defina padrões UML específicos para o projeto.
- Inclui:
    - Escolha dos diagramas UML a serem utilizados (casos de uso, classes, sequência, etc.).
    - Definição de convenções visuais para garantir consistência nos modelos.
-

## 6. Modelagem dos Casos de Uso

- Como: Utilize diagramas UML para representar as interações entre os atores e o sistema.
- Inclui:
    - Identificação dos atores (usuários ou outros sistemas que interagem com o sistema).
    - Criação dos fluxos básicos (caminho principal), alternativos e de exceção para cada caso de uso.
    - Documentação textual detalhada dos casos de uso.
-

## 7. Design do Sistema

- Como: Estruture a arquitetura do sistema com base nos requisitos e nos casos de uso.
- Inclui:
    - Definição da arquitetura geral (monolítica, microserviços, etc.).

        - Criação da estrutura inicial do banco de dados (entidades principais).
        - Protótipos das telas para validar a experiência do usuário antes do desenvolvimento.
-

## 8. Modelagem Técnica

- Como: Utilize diagramas UML detalhados para descrever a estrutura interna do sistema.
- Inclui:
    - Criação do Diagrama de Classes:Representação das classes, atributos alinhados aos requisitos e métodos principais.
    - Definição dos Relacionamentos entre Classes:Associações, generalizações/especializações (herança), composições e agregações.
    - Criação do Diagrama de Relacionamento:Representação gráfica das conexões entre entidades no banco de dados.
    - Diagrama de Sequência:Modelagem das interações entre objetos ao longo do tempo em cada caso de uso.
-

## 9. Operações Query e Estáticas

- Como: Defina as operações específicas que serão realizadas no sistema.
- Inclui:
    - Métodos estáticos (operações que não dependem da instância da classe).
    - Queries específicas para acessar informações no banco de dados.
-

## 10. Protótipos das Telas

- Como: Desenvolva wireframes ou protótipos interativos utilizando ferramentas como Figma ou Adobe XD.
- Inclui:
    - Representação visual das telas principais do sistema.
    - Validação com stakeholders antes da implementação final.
-

## 11. Teste dos Fluxos Básico, Alternativo e Exceção

- Como: Simule os fluxos documentados nos casos de uso para verificar sua completude.
- Inclui:
    - Testes manuais ou automatizados para validar os cenários principais (fluxo básico).
    - Testes para verificar comportamentos alternativos ou em situações excepcionais.
-

## 12. Especificação Técnica

- Como: Documente todos os detalhes técnicos necessários para o desenvolvimento.
- Inclui:
    - Arquitetura geral do sistema (incluindo tecnologias escolhidas).
    - Diagramas UML completos (casos de uso, classes, sequência, componentes).
    - Estrutura detalhada do banco de dados.
-

## 13. Implementação

- Como: Desenvolva o código-fonte seguindo as especificações técnicas definidas anteriormente.
- Inclui:
    - Desenvolvimento incremental baseado em sprints ou iterações ágeis.
    - Realização contínua de testes unitários durante o desenvolvimento.
-

## 14. Testes e Validação

- Como: Realize testes rigorosos para garantir a qualidade do sistema antes da entrega final.
- Inclui:
    - Testes Funcionais: Validar se o sistema atende aos requisitos funcionais definidos.
    - Testes Não Funcionais: Avaliar desempenho, segurança e usabilidade.
    - Testes Automatizados: Garantir consistência em cenários repetitivos.
-

### 15. Reutilização de Templates

- Como: Utilize templates padronizados ao longo do projeto para agilizar processos repetitivos.
- Inclui:
    - Templates para documentação técnica (requisitos, diagramas UML).
    - Templates para prototipação visual das telas.
-

### 16. Monitoramento Contínuo

- Como: Acompanhe o progresso do projeto em relação aos objetivos estabelecidos inicialmente.
- Inclui:
    - Indicadores-chave (KPIs) para medir desempenho técnico e financeiro.
    - Ferramentas como Jira ou Trello para rastrear tarefas pendentes.
-

### 17. Encerramento do Projeto

- Como: Finalize o projeto formalmente após atingir todos os objetivos definidos no escopo inicial.
- Inclui:
    - Entrega formal ao cliente com uma apresentação final do produto desenvolvido.
    - Documentação das lições aprendidas durante o processo.
    - Planejamento da manutenção futura do sistema.

# Explicando como deve ser executado

1. Visão e o escopo do projeto

- Como: Realizar reuniões com stakeholders para entender as expectativas e objetivos do projeto.
- Onde: Em um ambiente colaborativo, como workshops ou reuniões presenciais/virtuais.
- Quando: No início do projeto, antes de qualquer desenvolvimento.
- Por que: Para garantir que todos os envolvidos tenham uma compreensão clara do que o projeto deve alcançar e quais são os limites do trabalho a ser realizado.

2. Requisitos do projeto
    - Como: Coletar informações através de entrevistas, questionários e análise de documentos existentes.
    - Onde: Em reuniões com usuários finais, clientes e outros stakeholders.
    - Quando: Após definir a visão e o escopo, mas antes de iniciar a modelagem técnica.
    - Por que: Para assegurar que o software atenda às necessidades reais dos usuários e stakeholders.

3. Modelagem de processos de negócios
    - Como: Utilizar técnicas como fluxogramas ou BPMN (Business Process Model and Notation).
    - Onde: Em ferramentas de modelagem ou software específico para BPM.
    - Quando: Após a coleta dos requisitos, durante a fase de definição do projeto.
    - Por que: Para visualizar e entender como os processos atuais funcionam e identificar áreas de melhoria.

4. Perfis UML

- Como: Definir perfis específicos que serão utilizados para padronizar a modelagem UML no projeto.
- Onde: Na documentação do projeto ou em ferramentas UML.
- Quando: Durante a fase inicial de modelagem, após a definição dos requisitos.
- Por que: Para garantir consistência na representação dos modelos e facilitar a comunicação entre os membros da equipe.

5. Modelagem dos casos de uso
    - Como: Criar diagramas de casos de uso utilizando a linguagem UML.
    - Onde: Em ferramentas de modelagem UML ou em documentos colaborativos.
    - Quando: Após a coleta dos requisitos, durante a fase de análise.
    - Por que: Para descrever as interações entre usuários (atores) e o sistema, ajudando na identificação das funcionalidades necessárias.

6. Diagrama de classes
    - Como: Representar as classes do sistema, seus atributos e métodos, utilizando UML.
    - Onde: Em softwares de modelagem UML ou diagramas visuais.
    - Quando: Após modelar os casos de uso, durante a fase de design do sistema.
    - Por que: Para estruturar o sistema em termos de objetos e suas interações, facilitando o desenvolvimento orientado a objetos.

7. Atributos alinhados

- Como: Definir atributos para cada classe com base nos requisitos coletados.
- Onde: No diagrama de classes ou na documentação técnica.
- Quando: Durante a modelagem das classes, após definir as classes principais.
- Por que: Para garantir que cada classe possui as informações necessárias para funcionar corretamente dentro do sistema.

8. Os diferentes tipos de relacionamentos
    - Como: Identificar e documentar relacionamentos como associações, heranças e composições entre classes.
    - Onde: No diagrama de classes ou em documentos explicativos.
    - Quando: Durante a criação do diagrama de classes.
    - Por que: Para entender como as classes se relacionam entre si, o que é essencial para o design orientado a objetos.

9. Diagrama de relacionamento
    - Como: Criar um diagrama visual que represente as interações entre diferentes entidades do sistema.
    - Onde: Em ferramentas UML ou softwares gráficos.
    - Quando: Após definir os relacionamentos entre classes.
    - Por que: Para fornecer uma visão clara das interações no sistema, facilitando o entendimento da estrutura geral.

10. Diagrama de sequência

- Como: Modelar interações entre objetos ao longo do tempo usando UML.
- Onde: Em ferramentas específicas para diagramas UML ou em documentos técnicos.
- Quando: Após definir os casos de uso e diagramas de classes.
- Por que: Para detalhar como os objetos se comunicam durante a execução dos casos de uso, essencial para entender o fluxo do sistema.

11. Modelagem de componentes e interface
    - Como: Definir componentes do sistema e suas interfaces utilizando diagramas UML apropriados.
    - Onde: Em ferramentas UML ou na documentação técnica do projeto.
    - Quando: Durante a fase de design após ter os diagramas anteriores prontos.
    - Por que: Para garantir que os componentes do sistema se comuniquem corretamente através das interfaces definidas.

12. Utilizar operações query e estáticas
    - Como: Definir métodos nas classes para realizar consultas (queries) e operações estáticas necessárias ao funcionamento do sistema.
    - Onde: No diagrama de classes ou na documentação técnica correspondente às classes.
    - Quando: Durante a modelagem das classes, após definir atributos e relacionamentos.
    - Por que: Para permitir que o sistema execute operações essenciais sobre seus dados.

13. Protótipos das telas

- Como: Criar wireframes ou protótipos interativos usando ferramentas específicas para design UX/UI.
- Onde: Em softwares dedicados ao design gráfico ou plataformas online para prototipação (como Figma ou Adobe XD).
- Quando: Após definir os requisitos funcionais e antes da implementação real da interface do usuário.
- Por que: Para visualizar como será a interação do usuário com o sistema antes da codificação final.

14. Fluxos básico, alternativo e de exceção
    - Como: Documentar todos os possíveis fluxos através dos casos de uso identificados anteriormente.
    - Onde: Na documentação técnica ou diretamente nos diagramas de casos de uso.
    - Quando: Após modelar os casos de uso e protótipos das telas.
    - Por que: Para garantir que todas as possíveis interações sejam consideradas no desenvolvimento, incluindo cenários inesperados.

15. Especificação técnica de requisitos
    - Como: Elaborar um documento detalhado com todas as especificações técnicas necessárias para o desenvolvimento do software.
    - Onde: Em um documento formal acessível à equipe técnica e stakeholders relevantes.
    - Quando: Após completar as fases iniciais da modelagem (casos de uso, diagramas).
    - Por que: Para fornecer diretrizes claras sobre como deve ser implementado o software, assegurando alinhamento entre todos os envolvidos.

16. Reutilização de templates

- Como: Criar templates para documentos técnicos, diagramas UML e protótipos com base em padrões estabelecidos pela equipe ou pela indústria.
- Onde: Em repositórios internos da equipe ou plataformas colaborativas (como Confluence).
- Quando: Durante todas as fases do projeto para promover consistência nas entregas documentais.
- Por que: Para aumentar a eficiência no processo documental e garantir uniformidade nas representações gráficas utilizadas no projeto.

# EXPLICAÇÕES

Modelo SKOPOS

- O que é: O modelo SKOPOS é uma abordagem de planejamento de projetos que se concentra na definição clara do escopo do projeto. Ele ajuda a identificar o que deve ser incluído no projeto e o que deve ser excluído, garantindo que todos os stakeholders tenham uma compreensão comum dos objetivos e limitações do projeto.
- Onde fazer: O modelo SKOPOS pode ser utilizado em qualquer fase de planejamento de projetos, especialmente durante a fase inicial, onde é crucial definir o escopo e os objetivos. É frequentemente documentado em um plano de projeto formal.

BPMN (Business Process Model and Notation)

- O que é: BPMN é uma notação gráfica padronizada para descrever processos de negócios. Ela permite que analistas de negócios e desenvolvedores representem visualmente os passos de um processo, facilitando a comunicação entre as partes interessadas.
- Onde usar: BPMN é utilizado em modelagem de processos de negócios, documentação de fluxos de trabalho e análise de processos para identificar melhorias. Pode ser aplicado em qualquer setor que necessite otimizar processos.

UML (Unified Modeling Language)

- O que é: UML é uma linguagem de modelagem padrão usada para especificar, visualizar, construir e documentar artefatos de sistemas de software. Ela inclui vários tipos de diagramas, como diagramas de casos de uso, diagramas de classes e diagramas de sequência.
- Onde usar: UML é amplamente utilizado em desenvolvimento de software para descrever a estrutura e o comportamento dos sistemas. É útil durante todas as fases do ciclo de vida do desenvolvimento.

Lucidchart e Draw.io

- O que são: Lucidchart e Draw.io são ferramentas online para criação de diagramas. Ambas permitem a modelagem visual utilizando uma interface intuitiva.
- Software gratuito: Draw.io (agora conhecido como diagrams.net) é gratuito e não requer registro. Lucidchart oferece uma versão gratuita limitada.
- Como usar:
    - Acesse o site da ferramenta.
    - Crie uma nova conta (para Lucidchart) ou comece a criar um diagrama diretamente (Draw.io).
    - Selecione o tipo de diagrama desejado (como UML ou BPMN).
    - Arraste e solte elementos do painel lateral para criar seu diagrama.
-

Figma e Adobe XD

- O que são: Figma e Adobe XD são ferramentas de design colaborativo usadas para criar protótipos interativos e interfaces de usuário.
- Como utilizar:
    - Figma:Acesse o site do Figma e crie uma conta gratuita.
    - Adobe XD:Baixe e instale o Adobe XD (disponível gratuitamente com limitações).
-

Passo a Passo para Criar um Checklist

1. Defina os Objetivos: Determine qual tarefa ou projeto você está criando o checklist.
2. Liste as Etapas Necessárias: Escreva todas as etapas ou itens necessários para completar a tarefa.
3. Organize em Ordem Lógica: Coloque os itens na ordem em que devem ser realizados.
4. Revise com Stakeholders: Se aplicável, revise o checklist com outras pessoas envolvidas na tarefa.
5. Finalize e Compartilhe: Salve o checklist em um formato acessível (como PDF ou Google Docs) e compartilhe com quem precisa usá-lo.

Como Usar Ferramentas Trello ou Asana

- Trello:
    - Crie uma conta no Trello.
    - Crie um novo quadro (board) para seu projeto.
    - Adicione listas (por exemplo, "A Fazer", "Fazendo", "Concluído").
    - Crie cartões (cards) dentro das listas para cada tarefa específica.
    - Mova os cartões entre listas conforme as tarefas progridem.
-
- Asana:
    - Crie uma conta no Asana.
    - Inicie um novo projeto usando um modelo ou do zero.
    - Adicione tarefas ao projeto, atribua responsáveis e defina prazos.
    - Utilize seções ou subtarefas para organizar melhor as atividades dentro do projeto.
-