TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS

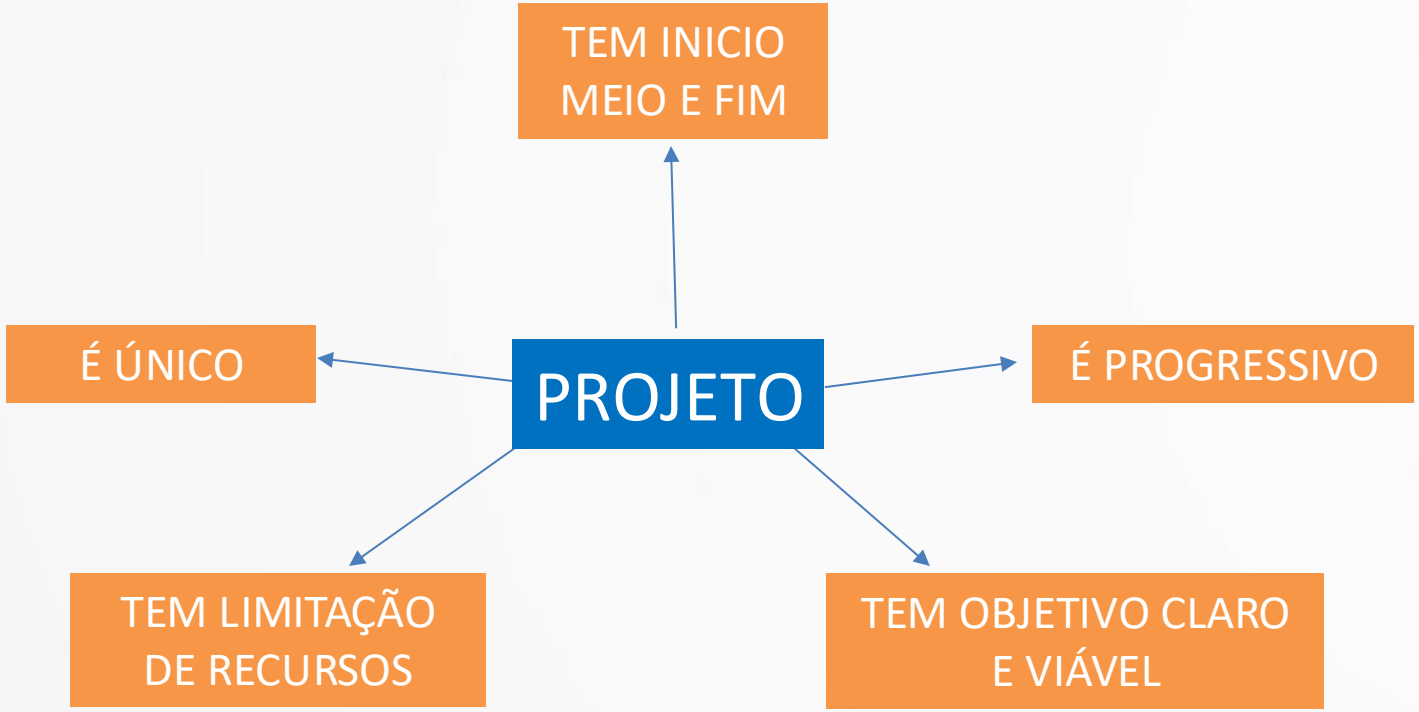
Planejamento de sistema

Docente: Diego Rodrigues

O que é um projeto?

Um projeto de software é um esforço temporário que tem como objetivo criar um software, ou seja, um sistema tecnológico, a partir de etapas de concepção de produto.

O que é um projeto?



É único

Isso significa que um projeto não pode ser repetido continuamente, ele é um evento único e planejado. Por exemplo: se um engenheiro receber demandas de dois clientes diferentes para desenvolver empreendimentos com as mesmas características, não poderá oferecer o mesmo projeto para os dois.

Isso porque cada projeto terá membros, especificidades técnicas, locais de aplicação, disponibilidade de recursos e provavelmente datas de início e término diferentes.

Tem início e fim definidos

Quando dizemos que ele é temporário, isso não quer dizer que ele terá curto prazo, mas que há um início e um fim definidos, um prazo a ser respeitado.

É progressivo

Considerando o prazo de conclusão que tem que ser atingido, o projeto deve ser dividido em etapas, e não entregue de uma vez só. Geralmente, as entregas são compostas por tarefas que interagem entre si e dependem umas das outras.

Por exemplo, para construir uma casa, é preciso construir as fundações antes de levantar as paredes. São tarefas ou atividades progressivas que vão dando forma à entrega final.

Tem delimitação de recursos

Os insumos para executar um projeto são limitados conforme o que consta no escopo do projeto. Para definir as linhas de base dos recursos (sejam eles humanos, financeiros ou materiais), é comum que se faça um levantamento de custos antes da execução, para garantir que a falta de algum recurso inviabilize a entrega do projeto.

Tem objetivo claro e viável

Mais uma característica dos projetos é que eles são feitos com um objetivo claro e viável, isto é, possível de ser alcançado. É necessário esclarecer, porém, que ele não é uma meta, um desejo ou mesmo a visão de futuro de uma empresa: ele pode ser um caminho para alcançá-los.

Processo

O processo de software é uma metodologia que define as ações e tarefas necessárias para desenvolver um software de qualidade. Ele é um conjunto de atividades organizadas e planejadas, que envolvem a criação, projeto, teste e suporte ao software.

Qual é a diferença entre um projeto e um processo?

De forma resumida, a principal diferença entre projetos e processos é que, enquanto um processo é contínuo e repetido inúmeras vezes, os projetos são temporários e realizados uma única vez. Mesmo que alguns procedimentos sejam repetidos no projeto, a repetição não muda o fato de que as suas entregas sejam únicas.

Qual é a diferença entre um projeto e um processo?

Projeto		Processo
Tempo determinado	Vs.	Tempo indeterminado
Resultado exclusivo		Resultado padrão
Trabalho inédito		Trabalho repetitivo

Qual é a diferença entre um projeto e um processo?

Os processos geram resultados padronizados e são fortemente definidos, enquanto os projetos geram resultados únicos e podem ser definidos progressivamente, em ondas de planejamento.

E, por fim, os processos têm objetivos mutáveis, que podem ser atualizados conforme a empresa desejar, mas os objetivos dos projetos são únicos e imutáveis.

Projeto de sistemas (softwares)

Um projeto de sistemas traduz para o papel como serão atendidas as requisições dos clientes que precisam de uma solução tecnológica. Seu objetivo é definir e detalhar um modelo de software que seja viável e capaz de solucionar as necessidades do cliente.

Quando uma empresa cresce e precisa de um software mais robusto para poder gerenciar melhor as áreas do negócio, pode recorrer a uma empresa de desenvolvimento para criar o plano do sistema e desenvolver uma solução sob medida.

Ciclo de Vida do Desenvolvimento de Software (SDLC)

Como o Ciclo de Vida do Desenvolvimento de Software funciona e quais ferramentas são necessárias em cada uma das suas fases?

Fase 1: Coleta e Análise de Requisitos

Esta primeira fase do Ciclo de Vida de Desenvolvimento de Software é uma visão geral e das diretrizes do projeto/software.

Todas as partes interessadas – incluindo clientes, vendedores, especialistas do setor, desenvolvedores de software, analistas de negócios e gerentes de projeto, devem colaborar para juntar as informações necessárias sobre o que será construído. Lembre-se de ser preciso ao descrever os requisitos; quanto mais detalhado for, melhor.

Fase 1: Coleta e Análise de Requisitos

Checklist de informações essenciais para o cumprimento dos requisitos:

- Perfil do usuário e como ele/ela se comporta ao usar sua solução
- Feedback, pesquisas, entrevistas, questionários, testes e muito mais.
- Escopo e propósito do produto (problemas que seu software deve resolver)
- Listagem de todos os riscos envolvidos
- Planejamento de cronogramas e calendários
- Pontos fortes e fracos do sistema atual, tendo como objetivo melhoria (SWOT)
- Custo e recursos necessários para implementação e lançamento
- Equipes do projeto e estrutura de liderança.
- Após o briefing discutido na reunião inicial, é recomendado usar um documento SRS (Software Requirements Specification – Especificação de Requisitos de Software) para orientar o processo de desenvolvimento de software.

O SRS é um documento que descreve todas as características do produto e concentra todas as principais informações do projeto.

Fase 2: Estudo de viabilidade de produto

Um estudo de viabilidade é uma imagem clara do projeto. Esta etapa do SDLC é uma das mais importantes e às vezes pode ser executada simultaneamente à primeira etapa. É importante que todas as partes interessadas saibam exatamente todo o contexto econômico, técnico, jurídico e de programação deste projeto, porque isso pode alterar o escopo ou demonstrar se o software vai funcionar ou não.

Fase 2: Estudo de viabilidade de produto

É por isso que a análise como um estudo de viabilidade desempenha um papel relevante no Software Development LifeCycle. Durante o estudo de viabilidade, considere incluir informações como:

- Descrição do produto ou serviço
- Demonstrativos contábeis
- Detalhes de operação e gerenciamento
- Pesquisa e política de marketing
- Dados financeiros e obrigações fiscais
- Requerimentos legais
- Plano de implementação do projeto
- Tempo e orçamento disponíveis

Fase 2: Estudo de viabilidade de produto

Tipos de estudos de viabilidade: Técnica, Econômica, Jurídica, Operacional e de Planejamento.

1 - Viabilidade Técnica – Tem como objetivo um esboço do projeto dos requisitos do sistema, para determinar se a empresa possui os conhecimentos técnicos para lidar com a conclusão do projeto. Portanto, o estudo deve verificar ferramentas, equipamentos e habilidades necessárias para o projeto, bem como avaliação de hardware e software, disponibilidade de insumos, fatores de eficiência, entre outros.

2 - Viabilidade Econômica – Ajuda as partes interessadas a saber se o software necessário é financeiramente sustentável para a empresa. Portanto, nesta análise, é obrigatório reunir informações sobre os custos envolvidos na contratação de equipes de engenharia e produtos, despesas de hardware e software.

Fase 2: Estudo de viabilidade de produto

3 - Viabilidade Jurídica ou Legal – Verifica se há algum conflito de sistema com os requisitos legais e éticos, como regulamentação local de proteção de dados, certificado de projeto, licença, direitos autorais, etc.

5 - Viabilidade Operacional – Se concentra em descobrir como o produto funcionará, se será fácil para o usuário final e como os desenvolvedores farão a manutenção após a implantação. Junto com isso, outros escopos operacionais estão determinando a usabilidade do produto, determinando que as soluções sugeridas pela equipe de desenvolvimento de software são aceitáveis ou não, etc.

6 - Viabilidade de Planejamento – Deve medir principalmente cronogramas/prazos para levar até o seu término. Considerando sua experiência técnica, os prazos dos projetos são razoáveis? Alguns projetos são iniciados com prazos específicos. É necessário determinar se os prazos são obrigatórios ou desejáveis.

Fase 3: Design de Software

É hora de projetar! Nesta etapa do SDLC, a equipe produzirá a DSS (Design Document Specification – Especificações de Documentação do Projeto) com base nos requisitos do usuário e na análise detalhada feita na fase anterior.
O documento DDS define a arquitetura geral do sistema e descreve todas as informações para os desenvolvedores começarem a trabalhar no produto, como recursos, input, output, bancos de dados, formulários, esquemas de códigos, especificações de processamento e tempo esperado para entregar o produto.
Os documentos de design mais comuns usados nesta fase são Design de alto nível (HLD – High-Level Design) e Design de baixo nível (LLD – Low-Level Design).

Fase 3: Design de Software

O **High-Level Design (HLD)** é uma breve descrição da funcionalidade de cada módulo e de como funcionará a relação de interface e dependências entre os módulos. Também inclui as tabelas de banco de dados, identificadas junto com seus elementos-chave, e os diagramas de arquitetura, junto com detalhes técnicos.

O **Low-Level Design (LLD)** é um documento que descreve a lógica funcional dos módulos, tabelas de banco de dados (tipo e tamanho), detalhes da interface, tipos de problemas de dependência, lista de mensagens de erro e entradas e saídas para cada módulo.

Nos dois tipos de documentos, é importante especificar detalhes sobre como a arquitetura deve ser construída, em termos de linguagem de programação, modelos ou boilerplates etc, como é a comunicação entre o aplicativo com outros ativos e como os clientes devem interagir com a interface do usuário do software.

Fase 3: Design de Software

Além disso, será importante definir a plataforma ou dispositivo no qual o software será executado (Mobile, Apps, Desktops, consoles de jogos, por exemplo) e detalhes de programação, como métodos de resolução de problemas, realização de tarefas no aplicativo e detalhes de segurança como criptografia de tráfego SSL, proteção de senha e armazenamento seguro de credenciais de usuário.

Depois disso, começa a fase de prototipagem, que demonstrará uma ideia básica de como o aplicativo se parece e funciona. Este protótipo será mostrado às partes interessadas a fim de coletar feedback para melhorar o produto antes da fase de codificação.

Fase 4: Desenvolvimento de Software

A próxima fase do SDLC é o Desenvolvimento, que também é a mais longa. Seguindo o DDS e as diretrizes de desenvolvimento, os desenvolvedores vão traduzí-los em código-fonte e linguagem de programação. Todos os componentes do software são implementados nesta fase.

Para serem mais eficientes, as tarefas são divididas em unidades ou módulos e atribuídas aos vários programadores, em equipes distribuídas, de acordo com as suas competências.

Esta fase será um documento muito útil para todo o processo, incluindo guias de solução de problemas, FAQs ou apenas comentários no código-fonte para entender porque um determinado procedimento foi utilizado.

Fase 4: Desenvolvimento de Software

Nela, desenvolvedores também precisam usar aplicativos de controle de acesso ou gerenciamento de código-fonte ou ferramentas de programação como compilador, interpretadores e depurador para gerar e implementar o código. Essas ferramentas ajudam os profissionais a rastrear alterações no código e garantir a compatibilidade entre diferentes projetos de equipe e garantir que as metas de destino sejam atendidas.

Além das ferramentas, a qualidade da codificação depende da precisão da comunicação entre todas as partes interessadas, como QA (garantia de qualidade), testadores, DevOps, gerentes de produto e projeto.

Fase 5: Testagem de Software

A próxima fase do SDLC é testar o produto desenvolvido. Pesquisas têm mostrado que o processo de teste frequentemente é responsável por 40% do custo de desenvolvimento de software. Com a crescente necessidade de alta qualidade e eficiência, é cada vez mais importante que as organizações aprimorem seus testes de software.

A principal função do teste de software é detectar bugs para descobri-los e detectá-los. O escopo do teste de software inclui a execução desse código em vários ambientes e também o exame dos aspectos do código.

Fase 5: Testagem de Software

Testers do controle de qualidade (QA) desempenham a função mais importante nesta fase, comparando as funções de software rodando e os requisitos estabelecidos nas etapas anteriores. Além disso, eles são responsáveis por encontrar e relatar bugs e erros aos desenvolvedores e à equipe de revisão do produto. Também podem sugerir e identificar oportunidades de melhoria em segurança ou automação para que o projeto obtenha uma maior satisfação do usuário e uma melhor taxa de uso.

As métricas de software ajudam a:

1. Evitar armadilhas que levam a custos excessivos
2. Identificar onde o problema surgiu
3. Esclarecer metas, ao responder a perguntas como:
    - Qual é a estimativa de cada atividade do processo?
    - Qual é a qualidade do código que foi desenvolvido?
    - Como pode o código insuficiente ser melhorado?

Fase 6: Lançamento/implantação

A fase de lançamento e implantação concentra-se em observar como o mercado reage ao seu produto. É hora de lançar a versão final do software após os testes!

Durante a preparação e os procedimentos para a fase de lançamento, a equipe estabelece um procedimento operacional para organizar como o software deve funcionar no ambiente de TI e fornecer um plano de mitigação para apoiar o usuário final no reparo do problema.

Depois disso, é hora de programar cada parte do sistema. Esta fase inclui enviar o programa e programar cada site regional e cada sistema de computador.

Fase 6: Lançamento/implantação

Depois que sua equipe implantar o aplicativo e entregá-lo aos usuários, fique atento ao feedback e verifique se há problemas de implantação e o que precisam ser melhorados, de acordo com a expectativa do cliente.

Durante a fase de implantação, não se esqueça de identificar as principais equipes e funções envolvidas, como migrações e atualizações de software e configuração de permissões e funções de acesso, e tente limitar o impacto de quaisquer problemas de configuração inicial usando projetos-piloto.

Fase 7: Manutenção

A fase de manutenção envolve correção de bugs, atualização do aplicativo para as versões mais recentes do software e aprimoramento, adicionando algumas novas especificações mencionadas na primeira fase. Nesse ponto, o ciclo de desenvolvimento está encerrado.

Quais são os modelos mais comuns de SDLC?

Modelo de desenvolvimento em cascata (Waterfall)

Um modelo clássico dividido em várias fases e o resultado de uma fase atua como entrada para a próxima. A principal vantagem desse modelo é a possibilidade de avaliar o produto de desenvolvimento em cada fase antes de prosseguir. Por outro lado, é limitado em velocidade, uma vez que uma fase deve terminar antes que outra possa começar.

Modelo incremental

O produto é dividido em partes, que são desenvolvidas sequencialmente e entregues ao cliente conforme forem finalizadas. É um exemplo de metodologia ágil, com contato contínuo entre a empresa e o cliente. O modelo incremental é uma boa opção quando é necessário fornecer rapidamente um conjunto de funcionalidades aos usuários.

Quais são os modelos mais comuns de SDLC?

Modelo de Desenvolvimento em Espiral

O modelo em espiral é orientado a risco, com base na adoção dos melhores recursos dos modelos de prototipagem e em cascata. Assim, o método combina prototipagem rápida e velocidade no projeto e desenvolvimento, ao mesmo tempo em que destaca a repetição das fases de planejamento, projeto, construção e teste.

Modelo Big Bang

Usado principalmente por equipes menores, esse método simples consiste em implementar requisitos conforme eles aparecem no projeto. Não requer muito planejamento ou programação ou mesmo fases de teste formais.

Quais são os modelos mais comuns de SDLC?

V-Model

Neste modelo, em vez de se mover para baixo de forma linear, as etapas do processo são dobradas para cima após a fase de codificação, fazendo a forma de um V. O V-Model demonstra os relacionamentos entre cada fase do ciclo de vida de desenvolvimento e sua fase de teste associada.

Modelo de Desenvolvimento Ágil (Agile)

Aqui, a experiência do usuário é o foco principal e faz com que as equipes interajam mais rapidamente com o feedback do cliente e executem todos os ciclos agilmente para responder a um mercado em constante mudança. É por isso que este modelo produz uma sucessão de lançamentos e fases de uma a três semanas de duração.

Quais são os benefícios do SDLC?

O principal benefício do SDLC é a previsibilidade. Você pode planejar e executar exatamente tudo no processo de desenvolvimento. O Ciclo de Vida do Desenvolvimento de Software oferece a oportunidade de visualizar em escala todo o processo e gerenciamento de seu projeto. Alguns dos benefícios do SDLC são:

- Permite ter um plano claro para o cumprimento de um objetivo de negócio, também custos e recursos estão sempre em jogo
- Entrega produtos e software com alto nível e qualidade devido ao foco em testes e experiência do usuário
- Melhora a comunicação na equipe por causa da metodologia de fases
- Reduz o gasto de tempo e aumenta o retorno de dinheiro
- Minimiza o potencial de risco durante qualquer projeto de desenvolvimento.

SDLC é uma metodologia realmente interessante para quem deseja aprimorar os processos de desenvolvimento de software e dimensionar o gerenciamento de produtos. Também é sua chance de testar equipes distribuídas trabalhando juntas e explorar as habilidades em cada fase do projeto.