

Prolongado Conhecimento

1-) O que é um software no sentido mais aprofundado? O que compõe um software? Dê exemplos de software

2-) Pesquise sobre sobre embarcado. O que é? como são? Onde estão? Exemplos…

3-) Qual o seu entendimento sobre o excerto do livro:- “Hoje, já se tem conhecimento de que software — na maioria das vezes — não deve ser construído em fases estritamente sequenciais, como ocorre com produtos tradicionais de engenharia, tais como Engenharia Civil, Engenharia Mecânica, Engenharia Eletrônica, etc.”

4-) Dê exemplos de: **dificuldades essenciais** e **dificuldades acidentais**.

5-) Dê exemplos dentro de algum sistema que você conhece de 1 requisito funcional e 1 requisito não funcional.

6-) Na construção de software são utilizados vários tipos de técnicas e procedimentos: Pesquise sobre as técnicas e procedimentos citados no livro.

1-) Estrutura de dados

2-) Frameworks e Bibliotecas

3-) Compiladores

4-) Gerenciadores de bancos de dados

7-) Testes são importantes? Por que?

8-) Cite/pesquise sobre os tipos de Manutenção e Evolução de Software

9-) Cite/Pesquise sobre um sistema legado

10-) Pesquise sobre sistema de gerenciamento GIT e seus similares

11-) Cite alguns prováveis Stakeholders de uma empresa que vc conhece: Supermercado; Farmácia; Sistemas escolares…

**Respostas**

**1-** O software é um conjunto de programas, instruções e dados que trabalham juntos para realizar uma tarefa específica em um computador ou dispositivo eletrônico. O software é composto por dois componentes principais: o código-fonte e o código objeto.

O código-fonte é o conjunto de instruções escritas em uma linguagem de programação que os desenvolvedores usam para criar o software. Ele é escrito em um formato legível para humanos e é convertido em código objeto através de um processo conhecido como compilação.

O código objeto é o código executável que é processado pelo computador. É gerado a partir do código-fonte por meio de um compilador e contém as instruções que o computador precisa seguir para executar o software.

Existem vários tipos de software, incluindo:

**.Sistemas Operacionais**: como o Windows, macOS e Linux, que fornecem uma interface entre o hardware e o software de aplicação.

**.Softwares de aplicação:** como Microsoft Office, Adobe Photoshop, Firefox e Google Chrome, que realizam tarefas específicas para os usuários.

**.Softwares de banco de dados:** como MySQL, Oracle e Microsoft SQL Server, que gerenciam e organizam grandes quantidades de dados.

**.Softwares de jogos:** como Minecraft, Fortnite e League of Legends, que oferecem entretenimento e desafios para os jogadores.

**.Softwares de sistema embarcado:** como o software que controla um termostato inteligente, ou um sistema de GPS em um carro.

**2-** Um sistema embarcado é um tipo de sistema computacional que é projetado para desempenhar uma função específica, muitas vezes com requisitos de tempo real e com restrições de recursos de hardware. Esses sistemas são frequentemente integrados em dispositivos físicos e eletrônicos, como equipamentos médicos, automóveis, sistemas de controle de tráfego, eletrodomésticos, entre outros. Exemplos de sistemas embarcados incluem:

.Sistema de navegação GPS em um carro

.Controlador de temperatura de um forno

.Sistema de monitoramento de batimentos cardíacos em equipamentos médicos

.Computador de bordo em um avião

.Sistema de controle de tráfego aéreo

.Dispositivos de segurança em equipamentos de construção

**3-** Esse excerto está se referindo à abordagem tradicional de desenvolvimento de software, que é conhecida como "modelo cascata" ou "modelo em cascata". Nesse modelo, o processo de desenvolvimento é dividido em fases sequenciais, onde cada fase é concluída antes que a próxima possa ser iniciada. No entanto, a citação sugere que essa abordagem não é a mais adequada para o desenvolvimento de software. Isso porque o software é uma entidade dinâmica, que está constantemente mudando e evoluindo. Além disso, o desenvolvimento de software envolve muita interação entre as equipes de desenvolvimento e os usuários finais.

Assim, em vez de seguir um processo estritamente sequencial, muitos desenvolvedores agora utilizam abordagens mais flexíveis e iterativas, como o desenvolvimento ágil de software. Nesse modelo, o processo de desenvolvimento é dividido em iterações, onde cada iteração é uma "mini-versão" do produto final. Isso permite que os desenvolvedores obtenham feedback dos usuários finais em cada etapa do processo de desenvolvimento e ajustem o produto de acordo com esses feedbacks.

**4-** As dificuldades essenciais são problemas que são inerentes ao projeto ou à natureza do trabalho que está sendo realizado. Esses tipos de dificuldades podem ser antecipados e planejados para minimizar seus efeitos negativos, mas geralmente não podem ser completamente eliminados. Alguns exemplos:

.Complexidade do projeto

.Ambiguidade nos requisitos do projeto

.Limitações de recursos

As dificuldades acidentais, por outro lado, são problemas que surgem de forma inesperada durante o projeto e que podem ter um impacto significativo no andamento do projeto. Essas dificuldades são geralmente causadas por fatores externos ou eventos imprevistos. Alguns exemplos:

.Atrasos de fornecedores

.Mudanças inesperadas nos requisitos do projeto

.Problemas de comunicação entre as equipes

**5-** Sistema de um e-commerce:

**.Requisito funcional**: Permitir que o usuário adicione produtos ao carrinho de compras: esse é um requisito funcional básico para um e-commerce. O sistema deve permitir que o usuário adicione produtos ao carrinho de compras e que visualize os produtos selecionados antes de finalizar a compra.

**.Requisito não funcional:** Ter uma alta disponibilidade: esse é um requisito não funcional crítico para um e-commerce. O sistema deve estar disponível o tempo todo para os usuários, para que eles possam realizar suas compras a qualquer momento, sem interrupções. Isso significa que o sistema deve ter uma infraestrutura robusta e uma boa estratégia de gerenciamento de falhas, para minimizar os tempos de inatividade e garantir uma alta disponibilidade.

**6-** .**Estrutura de dados:** Estrutura de dados é uma forma de organizar e armazenar dados em um computador de modo que possam ser acessados e manipulados de forma eficiente. Uma estrutura de dados é um conjunto de elementos de dados relacionados, juntamente com as operações que podem ser realizadas nesses dados.

**.Frameworks e Bibliotecas:** Os frameworks são conjuntos mais abrangentes de ferramentas e estruturas que fornecem uma estrutura para o desenvolvimento de aplicativos de software. Eles incluem uma variedade de bibliotecas e componentes, além de convenções de codificação, padrões de arquitetura e fluxos de trabalho pré-definidos. Os frameworks são normalmente usados para desenvolver aplicativos complexos, como aplicativos da web, aplicativos móveis ou sistemas empresariais.

Uma biblioteca é um conjunto de código pré-escrito que os desenvolvedores podem incorporar em seus aplicativos para realizar tarefas específicas. As bibliotecas geralmente incluem uma coleção de funções, classes ou sub-rotinas que os desenvolvedores podem chamar para realizar uma determinada tarefa. As bibliotecas são normalmente usadas para realizar tarefas comuns, como manipulação de strings, gerenciamento de banco de dados, manipulação de arquivos, processamento de imagens, etc.

**.Compiladores:** Um compilador é um programa que traduz código-fonte de uma linguagem de programação para outra linguagem, geralmente para uma linguagem de baixo nível, como o código de máquina que o computador pode executar diretamente. O processo de compilação envolve várias etapas, incluindo análise léxica, análise sintática, análise semântica, geração de código e otimização. O compilador geralmente é dividido em duas partes principais: o front-end e o back-end. O front-end é responsável por analisar o código-fonte, verificar sua sintaxe e semântica e gerar uma representação intermediária do código. O back-end é responsável por gerar o código de máquina a partir da representação intermediária.

**.Gerenciadores de bancos de dados:** Gerenciadores de bancos de dados são programas de software projetados para gerenciar e manter dados em bancos de dados. Eles são amplamente utilizados em organizações de todos os tipos e tamanhos para armazenar, organizar, acessar e proteger informações críticas de negócios. Os gerenciadores de bancos de dados são responsáveis por armazenar, organizar e acessar os dados em um banco de dados. Eles fornecem ferramentas para criar e modificar esquemas de banco de dados, gerenciar usuários e permissões de acesso, realizar consultas e análises de dados, além de outras funções.

**7-** Sim, pois são nos testes que conseguimos verificar se o software está funcionando de forma correta.

**8-** Existem três tipos principais de manutenção de software:

**.Manutenção Corretiva**: É realizada para corrigir problemas ou defeitos encontrados no software em uso. Isso pode incluir a correção de erros de programação, problemas de desempenho ou falhas de segurança.

**.Manutenção Adaptativa:** É realizada para adaptar o software a mudanças no ambiente em que está sendo usado. Isso pode incluir mudanças em sistemas operacionais, atualizações de hardware, mudanças nas necessidades do usuário ou requisitos regulatórios.

**.Manutenção Evolutiva:** É realizada para adicionar novas funcionalidades ao software existente. Isso pode incluir a adição de recursos que os usuários solicitaram, melhorias de usabilidade ou atualizações de recursos para acompanhar o progresso tecnológico.

É importante notar que a manutenção de software não é apenas sobre a correção de erros. É também um processo contínuo de atualização e melhoria do software para garantir que ele permaneça relevante e útil ao longo do tempo.

**9-** Um sistema legado é um software ou sistema de informação que foi desenvolvido em tecnologias antigas ou obsoletas, mas que ainda é utilizado por uma organização devido a razões históricas ou de negócios.

A manutenção e o suporte a sistemas legados podem ser difíceis e dispendiosos, uma vez que as tecnologias utilizadas podem não ser mais suportadas pelos fabricantes ou não terem mais profissionais especializados no mercado para manutenção. No entanto, muitas empresas ainda dependem de sistemas legados para a realização de tarefas críticas para os negócios e, por isso, precisam continuar a utilizá-los.

Algumas características de um sistema legado podem incluir:

.Tecnologias antigas ou obsoletas, como linguagens de programação, sistemas operacionais ou bancos de dados;

.Falta de documentação ou documentação insuficiente;

.Interface de usuário desatualizada e pouco amigável;

.Processos de negócios ineficientes ou não otimizados.

A modernização de sistemas legados pode ser uma opção para as organizações, com o objetivo de melhorar a eficiência dos processos de negócios, reduzir custos e melhorar a experiência do usuário. A modernização pode incluir a reescrita de partes do sistema, a migração para novas tecnologias ou a integração com novos sistemas.

**10-** O GIT é um sistema de gerenciamento de versão distribuído popular, amplamente utilizado para o controle de versão de projetos de software. Existem também vários outros sistemas de gerenciamento de versão similares ao GIT, alguns dos quais são:

.SVN (Subversion): É um sistema de controle de versão centralizado que foi amplamente utilizado antes do surgimento do GIT. Ele ainda é usado em muitos projetos, especialmente em projetos legados.

.Mercurial: É um sistema de controle de versão distribuído semelhante ao GIT, que é usado por muitos projetos de software livre.

Bazaar: É outro sistema de controle de versão distribuído, que é projetado para ser fácil de usar e flexível.

**11-** Supermercado:

.Clientes finais que compram produtos;

.Fornecedores que fornecem produtos para o supermercado;

.Funcionários que trabalham no supermercado;

.Acionistas ou proprietários do supermercado;

.Governo e reguladores que podem exigir licenças e regulamentações para o supermercado;

.Concorrentes que operam em uma área semelhante.