Robson Novato Lobão - 20.1.4018

1- Qual sua definição para o termo “Engenharia de Software”?

A engenharia de software é uma área de conhecimento da tecnologia que foca em resolver problemas e trazer funcionalidades através do desenvolvimento de um sistema, ou aplicação, popularmente chamada de software. Busca sempre atender da melhor forma as necessidades do mercado e do cliente, buscando uma aplicação amigável ao usuário final, com qualidade e livre de bugs.

2- O que é um projeto segundo o PMBOK (Project Management Body of Knowledge)?

É um esforço com começo, meio e fim (finito) feito com a finalidade de apresentar um produto final exclusivo, ou seja, que o torna único. São executados por pessoas e baseados em recursos durante seu ciclo de vida, difere dos processos, pois como os processos são repetitivos e contínuos o projeto tem caráter único.

3- O que é arquitetura de software?

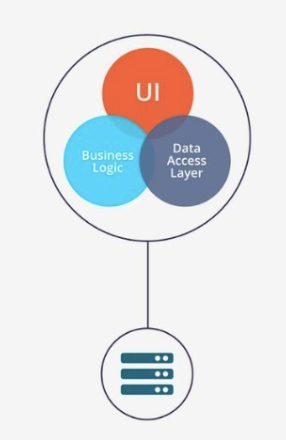
A arquitetura de software consiste basicamente na forma como um software se comporta no meio, ou seja, como ele é formado e suas características (componentes e funcionalidades), como ele lida com outros softwares e com o ambiente externo, trata tanto do micro (aplicações internas, documentação, funcionalidades) como do macro (relação com outros softwares e sistemas). Alguns exemplos de arquitetura são: Monolítica, de microsserviços, peer to peer ,cliente-servidor e em camadas.

4- O que é componente de software? O que é desenvolvimento baseado em componentes?

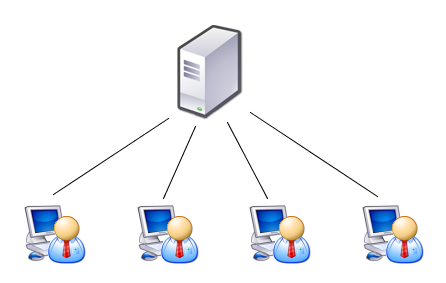
É um subconjunto de um software, um software é composto de vários componentes que podem também trabalhar de forma individual, para certas funcionalidades muito específicas, seria como um executável, um pedaço de código que já foi compliado e linkado e está disponível para uso. É o desenvolvimento que se baseia em entregar diversos "minissistemas" diminuindo a complexidade do software e aumentando a reutilização desses componentes para outras funcionalidades mais específicas.

5- Originalmente o UNIX possuía uma arquitetura monolítica, o Windows possui uma arquitetura cliente-servidor, a pilha de protocolos Ethernet possui uma arquitetura em camadas. Explique como essas arquiteturas são organizadas e como funcionam, utilize figuras. Que vantagens e desvantagens cada uma dessas arquiteturas possui?

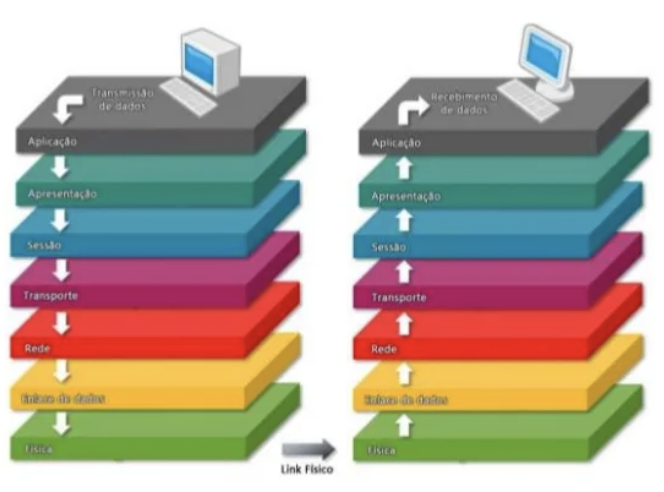
Arquitetura monolítica, funciona como um grande bloco de aplicação ou seja, seu UI, seu gerenciamento de banco de dados as lógicas de negócio, todo o software é compilado em um único executável, como vantagens como podemos citar a facilidade de coordenação e como desvantagem como está tudo em um grande bloco, as atualizações que virão, vão tornar o software muito pesado e ocupando muita memória.



Arquitetura cliente servidor, funciona como um servidor central que lastreia todas as interações entre os diversos clientes, ao contrário da peer to peer, tudo acontece com a autorização ou não do servidor central, ou seja, quando por exemplo queremos imprimir um arquivo, você emite ao servidor essa vontade, ele comunica a área de organização de arquivos, que volta com a permissão ou não para o servidor comunicar ao cliente. Como vantagem podemos ter a flexibilidade de funcionamento enquanto como desvantagem temos que pode ser mais lento por fazer várias comunicações por ação.

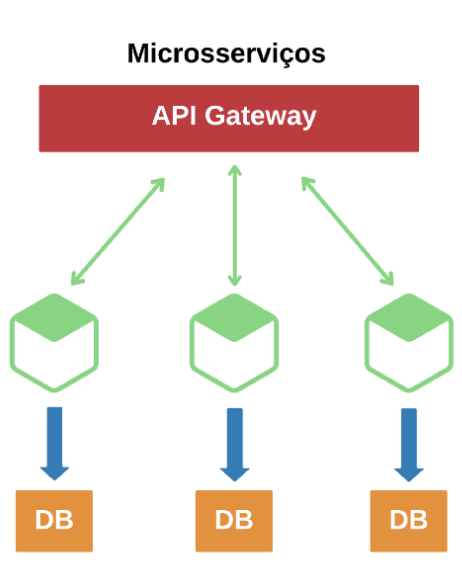


Pilha de protocolos, é uma arquitetura de camadas em que, a de nível mais básico (como se as camadas tivessem uma hierarquia) oferecem as camadas acima uma facilidade de funcionamento e desenvolvimento, já que você não precisa se preocupar com o funcionamento da funcionalidade mais básica x que já foi implementada pelas camadas abaixo das que estamos desenvolvendo, como por exemplo o qt mostrado em sala. Como vantagens temos a facilidade de desenvolvimento e atualização de software, já que não precisa ter as funcionalidades mais básicas desenvolvidas novamente e como desvantagem temos menos liberdade de desenvolvimento da aplicação como um todo.



6- Quais são os princípios da arquitetura de Microsserviços? Apresente uma figura e a explique.

Na arquitetura de microsserviços temos uma única interface gráfica em que o usuário vê todos os serviços, mas cada serviço é um módulo executável diferente.



Nessa imagem podemos ver como funciona, o cliente final na API de gateway tem acesso a diversas funcionalidades, representadas pelas caixas verdes, cada caixa entra em contato com um banco de dados em específico, evitando a concentração de funcionalidades em um único executável.

7- Qual a principal diferença entre as seguintes arquiteturas de software: biblioteca de funções e framework (arcabouço)? Dê exemplo de software largamente conhecidos que possuam essas arquiteturas. Quando uma arquitetura é preferível à outra?

A biblioteca de funções é uma coleção de pacotes que oferece funcionalidades ao usuário final, o framework é uma arquitetura que possui um conjunto de bibliotecas que já realizam certo trabalho ou funções. Praticamente todos os softwares utilizam esses conceitos, o framework é preferível quando a aplicação é mais robusta.

8- Defina o conceito de API – (Application Programming Interface).

Uma interface, normalmente especificada como um conjunto de operações, que permite acesso a uma funcionalidade da aplicação. Isso significa que essa funcionalidade pode ser chamada diretamente por outros programas e não apenas acessada através da interface de usuário

9- Defina os seguintes conceitos: (a) Fraco Acoplamento e (b) Alta Coesão.

a) Acoplamento é quando uma classe utiliza funções, métodos ou variáveis instanciadas em uma classe diferente, fraco acoplamento é quando as classes são altamente independentes, ou seja, não necessitam de classes externas para pleno funcionamento.

b) A coesão tanto no conceito de equipe quanto no conceito de métodos, significa quão bem as diferentes partes de um todo se relacionam entre si, ou seja, uma alta coesão significa que a comunicação no código ou na equipe está muito boa e clara.

10- O desenvolvedor de software é aconselhado a sempre separar a interface de um programa (API) da sua implementação. Por que?

Para aumentar a reutilização de software, como o exemplo citado da loja de carro e bicicleta, em que a lógica de negócios é o que muda, a implementação poderia ser a mesma para todos os serviços, então separar a API da implementação permite a reusabilidade de aplicações.

11- O que significa reuso de código? Quais as vantagens e desvantagens? Por que é importante?

Pegar trechos de códigos de caráter mais genérico e transformá-lo de acordo com a necessidade de classes ou aplicações do meu software, diminuindo o tempo de desenvolvimento a longo prazo, tendo em vista que uma classe genérica implementada hoje pode ser utilizada em um horizonte de tempo indeterminado. Como vantagem temos diminuição de tempo de desenvolvimento no longo prazo e como desvantagem a possibilidade de aumentar o tempo de desenvolvimento a curto prazo, tendo em vista que criar métodos que possam ser reutilizados exigem mais conhecimento do programador e análise de cenários para poder chegar em um bom código. É importante para aplicações robustas para economia de recursos.

12- Quais são as fases no desenvolvimento de um projeto de software? Quais atividades são realizadas em cada fase?

Fase de diagnóstico - análise do problema abordado pela equipe em um todo, analisando sua origem e formas de achar uma solução e atacar esse problema com máxima efetividade.

Concepção - criar uma ideia para a resolução do problema, ou validar, com usuários e a equipe de desenvolvedores, se o design (visual e arquitetural) trazido pelo cliente foi realizado corretamente, alterando-o se houver necessidade.

Levantamento de requisitos - Com a ideia de solução já prototipada é criado um documento extremamente específico que tem o objetivo de descrever minuciosamente cada detalhe e funcionalidade do software.

Fase de desenvolvimento - Design, programação e teste da solução arquitetada, a criação do produto em si.

Fase de manutenção - Realizar ajustes no sistema, corrigir bugs e adicionar funcionalidades sugeridas pelo usuário.

Conclusão - Finalização do projeto.

13- Qual a diferença entre verificação e validação de software?

Verificação é um processo dentro da validação do software, a validação é o processo que mostra que o software se adequa a suas especificações e satisfaz as especificações do cliente. Verificação são processos que comprovam a validação do software como testes, inspeções e revisões.

14- Defina cada um dos seguintes níveis de teste de software: (a) teste unitário, (b) teste funcional, (c) teste de integração, (d) teste sistêmico e (e) teste de aceitação.

a) Teste unitário - em que as unidades individuais de programa ou classes de objetos são testadas individualmente. Testes unitários devem centrar-se em testar a funcionalidade dos objetos ou métodos.

b) Teste funcional- é um processo de teste de caixa-preta, no qual os testes são derivados da especificação de sistema. O sistema é tratado como uma caixa-preta cujo comportamento só pode ser determinado por meio do estudo das entradas e saídas relacionadas.

c) Teste de integração- é a fase do teste de software em que módulos são combinados e testados em grupo. Ela sucede o teste de unidade, em que os módulos são testados individualmente, e antecede o teste de sistema, em que o sistema completo é testado num ambiente que simula o ambiente de produção.

d) Teste sistêmico- é um processo de teste de software e de hardware em que o sistema já completamente integrado é verificado quanto a seus requisitos num ambiente de produção. Está no escopo da técnica de teste de caixa-preta, e dessa forma não requer conhecimento da estrutura interna do sistema.

e) Teste de aceitação- é o teste em que os clientes testam um sistema para decidir se está ou não pronto para ser aceito pelos desenvolvedores de sistemas e implantado no ambiente do cliente.

15- Que é: (a) teste caixa branca, (b) teste caixa preta e (c) teste caixa cinza?

a) Teste caixa branca- é uma técnica de teste que usa a perspectiva interna do sistema para modelar os casos de teste. No teste de software, a perspectiva interna significa basicamente o código fonte.

b) Teste caixa preta- É uma modalidade de teste que só analisamos as entradas e saídas do software, sem nos preocuparmos com o código desenvolvido, normalmente feitos pelos qas, enquanto o item a) é feito pelos próprios desenvolvedores.

c) Teste caixa cinza- no teste da caixa cinza, os testadores compreendem certos aspectos do software que está sendo usado e podem ver algumas partes do código-fonte, mas não todo. Isso permite que os testadores interajam e compreendam mais completamente o programa que estão testando do que o teste de caixa preta permite, mas sem os problemas completos de acesso e segurança que podem surgir dos testes de caixa branca