Lista 1

Aluno: Nathann Zini dos Reis

Matrícula: 19.2.4007

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

A Engenharia de Software capacita as pessoas com a utilização de teorias, técnicas e ferramentas da Ciência da Computação para produção e desenvolvimento de sistemas. Por meio da análise, coleta e processamento de dados, ainda identificam potenciais falhas nesses produtos e criam soluções de alta performance.

O objetivo dessa Engenharia é acompanhar as inovações e ensinar aos alunos as melhores técnicas e modelos a serem seguidos.

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

Segundo o PMBOK, um projeto é um esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo. Ou seja, um projeto é tudo aquilo que precisamos realizar para gerar algo novo: seja uma casa, um sistema informatizado, um estudo/pesquisa, um trabalho de conclusão de curso, uma contratação ou um compra importante. Lembre-se que ser “temporário” significa que os projetos devem ter um início e um término definidos; não significa de curta duração.

Q.3 - O que é arquitetura de software?

A arquitetura de software de um sistema abrange a forma como suas partes são organizadas, incluindo questões como o comportamento dessa estrutura e quais componentes são responsáveis por realizar um conjunto específico de funções. Resumidamente, é um modelo repetível sob o qual um sistema pode ser desenvolvido.

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

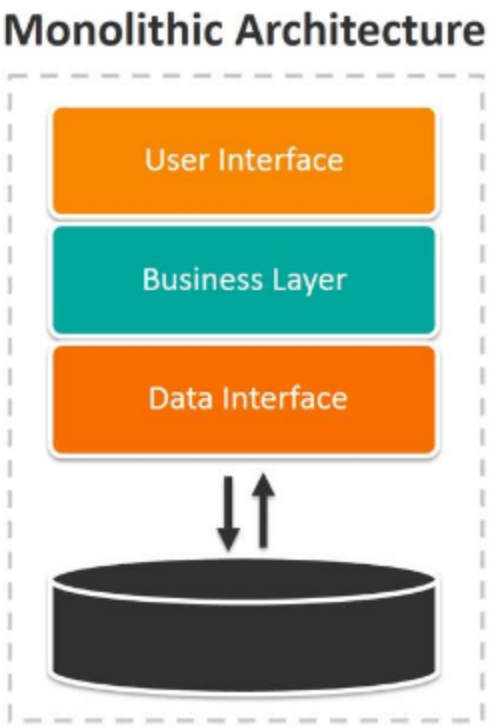
Componentes de Software é o termo utilizado para descrever o elemento de software que encapsula uma série de funcionalidades. Um componente é uma unidade independente, que pode ser utilizado com outros componentes para formar um sistema mais complexo.

O Desenvolvimento Baseado em Componentes (DBC) aborda a criação de sistemas de software que envolva a composição de componentes permitindo a adição, adaptação, remoção e substituição de partes do sistema sem a necessidade de sua completa substituição. Isso auxilia na manutenção dos sistemas uma vez que, permite a integração de novos componentes e/ou a atualização dos já existentes. A abordagem é criar ou adaptar os componentes para que sejam utilizados em diversos sistemas. Essa idéia vem ao encontro da reutilização que busca flexibilizar o desenvolvimento.

Q.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

  Arquitetura Monolítica é um sistema único, não dividido, que roda em um único processo, uma aplicação de software em que diferentes componentes estão ligados a um único programa dentro de uma única plataforma.



Vantagens

Mais simples de desenvolver: a organização fica concentrada em um único sistema;

Simples de testar: é possível testar a aplicação de ponta a ponta em um único lugar;

Simples de fazer o *deploy* para o servidor: a alteração é simplesmente feita e pronto;

Simples de escalar: como é só uma aplicação, se for preciso adicionar mais itens, é simplesmente ir adicionando o que for necessário.

Desvantagens

Manutenção: a aplicação se torna cada vez maior de acordo com o seu tamanho, o código será cada vez mais difícil de entender e o desafio de fazer alterações rápidas e ter que subir para o servidor só cresce;

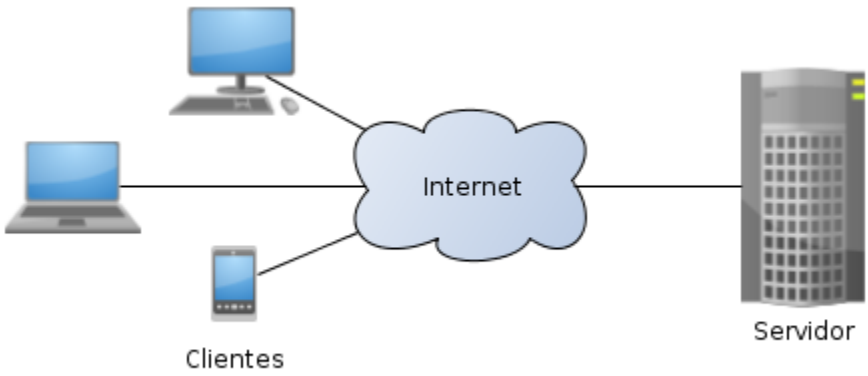
Alterações: para cada alteração feita, é necessário realizar um novo *deploy* de toda a aplicação;

Linha de código: uma linha de código que subiu errada pode quebrar todo o sistema e ele ficar totalmente inoperante;

Linguagens de programação: não há flexibilidade em linguagens de programação. Aquela que for escolhida no início do projeto terá que ser seguida, sempre. Se o desenvolvimento de uma nova funcionalidade exigir outra linguagem de programação, existem duas possibilidades: ou todo o código é alterado ou a arquitetura do sistema precisará ser trocada.

- Arquitetura cliente-servidor

  Arquitetura cliente-servidor ou modelo cliente-servidor é uma arquitetura na qual o processamento da informação é dividido em módulos ou processos distintos. Existe um processo que é responsável pela manutenção da informação (servidores) e outro responsável pela obtenção dos dados (os clientes).



Vantagens

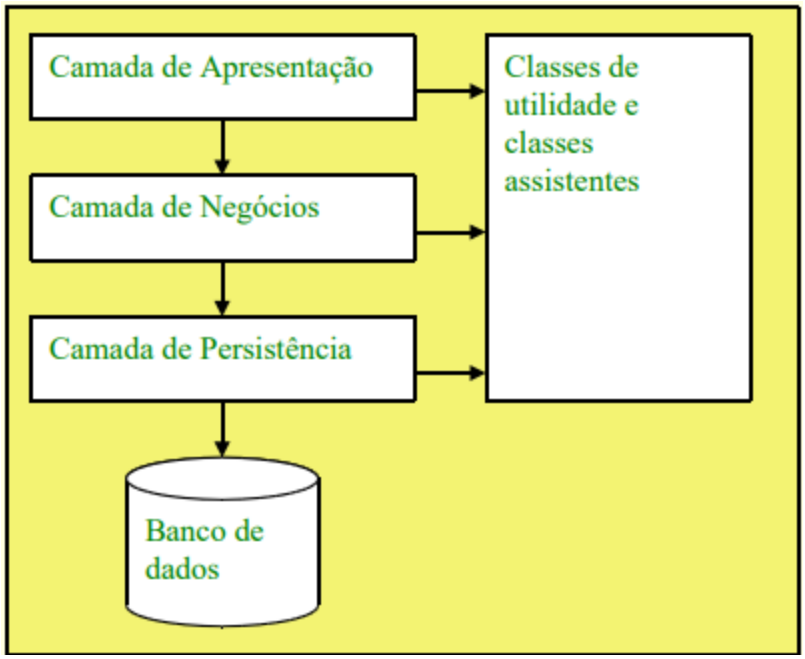
- Recursos centralizados
- Maior facilidade de manutenção

Desvantagens

- Sobrecarga
- Único nó

- Arquitetura em camadas

A arquitetura em camadas pode ser definida como um processo de decomposição de sistemas complexos em camadas para facilitar a compreensão do mesmo, como também, facilitar a manutenção deste sistema, ainda afirma que, esta técnica foi emprestada da arquitetura de computadores, que utilizam camadas de chamada ao sistema operacional, drivers e afins.



Vantagens

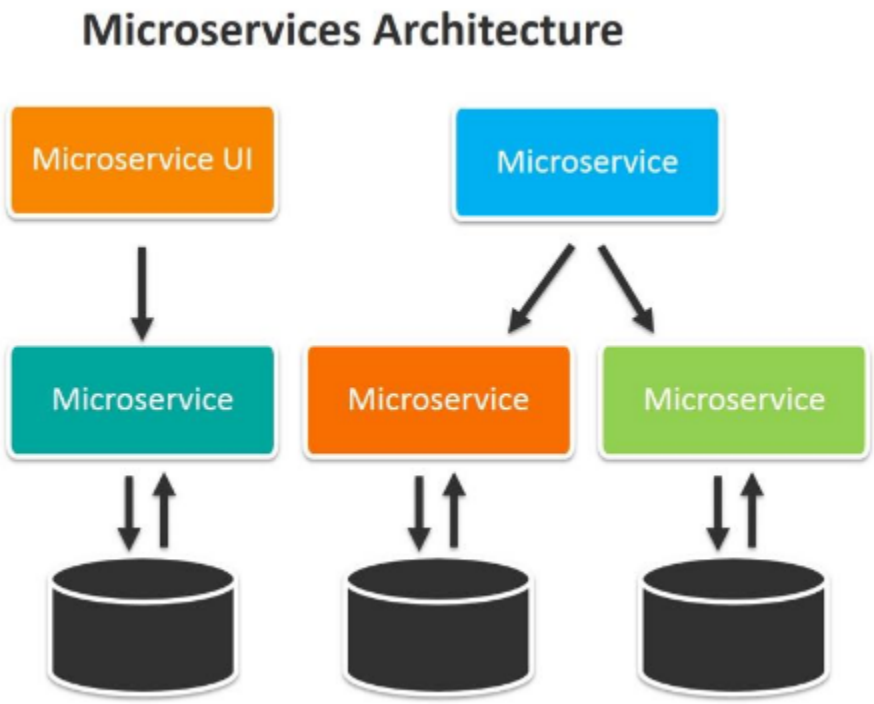
- Isola funções do SO facilitando sua manutenção e depuração;
- Cria uma hierarquia de níveis de modos de acesso, protegendo as camadas mais internas.

Desvantagens

- Desempenho: cada nova camada implica uma mudança no modo de acesso;
- Atualmente, a maioria dos sistemas comerciais utiliza o modelo de duas camadas, onde existem os modos de acesso usuário (não-privilegiado) e kernel (privilegiado);
- A maioria das versões do UNIX e o Windows da Microsoft estão baseadas neste modelo.

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

Os microsserviços são uma arquitetura e uma abordagem para escrever programas de software. Com eles, as aplicações são desmembradas em componentes mínimos e independentes. Diferentemente da abordagem tradicional monolítica em que toda a aplicação é criada como um único bloco, os microsserviços são componentes separados que trabalham juntos para realizar as mesmas tarefas. Cada um dos componentes ou processos é um microsserviço. Essa abordagem de desenvolvimento de software valoriza a granularidade, a leveza e a capacidade de compartilhar processos semelhantes entre várias aplicações. Trata-se de um componente indispensável para a otimização do desenvolvimento de aplicações para um modelo nativo em nuvem.



Q.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?

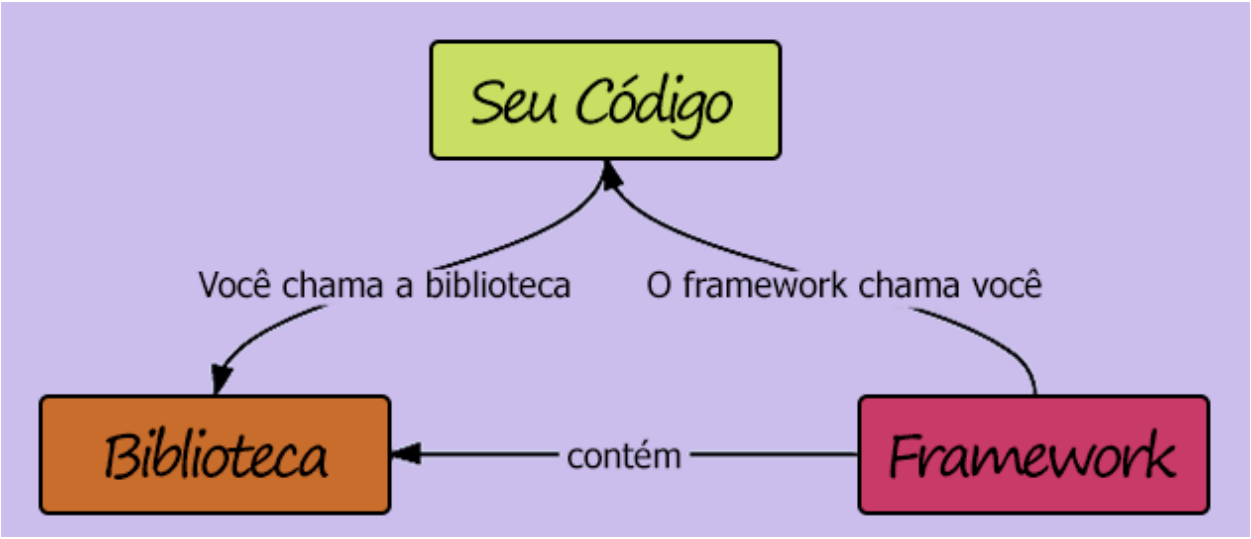
A ideia da biblioteca é compartilhar soluções por meio de funções ou métodos. Desenvolvedores disponibilizam bibliotecas que possuem muitas funções prontas. Assim, outros programadores podem utilizá-las, permitindo que o desenvolvimento seja mais fácil e rápido.

Pode-se dizer que o framwork é um conjunto de códigos abstratos e/ou genéricos que unem códigos com recursos iguais. Ou seja, está ligado à arquitetura do seu software.

Resumindo…

Biblioteca: é uma coleção de implementações de comportamentos escritos em uma linguagem e importadas no seu código. Nesse caso, há uma interface bem definida para cada comportamento invocado. Um bom exemplo é a biblioteca jQuery que implementa certos comportamentos, como por exemplo, a manipulação do HTML.

Framework: estrutura real, ou conceitual, que visa servir como suporte (ou guia) para a construção de algo (um produto, por exemplo). “Este algo” herdará as características desta estrutura, implementando o produto final (“algo”). Exemplo: BootStrap.



Ao passo que no Framework nós já temos toda a estrutura pronta, nos poupando desse trabalho, teremos menos liberdade para certas escolhas. Em contrapartida, nas bibliotecas temos que definir coisas comuns como arquitetura e fluxo, mas teremos mais liberdade de escolher cada biblioteca responsável por cada funcionalidade em nossa aplicação.

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

API é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web. Uma API é criada quando uma empresa de software tem a intenção de que outros criadores de software desenvolvam produtos associados ao seu serviço. Existem vários deles que disponibilizam seus códigos e instruções para serem usados em outros sites da maneira mais conveniente para seus usuários. O Google Maps é um dos grandes exemplos na área de APIs. Por meio de seu código original, muitos outros sites e aplicações utilizam os dados do Google Maps adaptando-o da melhor forma a fim de utilizar esse serviço.

API(Application Programming Interface - Interface entre Aplicativo e programação) é um conjunto de instruções e padrões de programação para acesso a um aplicativo de software. Uma empresa de software lança sua API para o público de modo que outros criadores de software possam desenvolver produtos acionados por esse serviço.

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

(a) - Quando um sistema possui entre seus componentes uma relação de interdependência fraca, significa que a dependência entre seus componentes é baixa, ou seja, estão acoplados, mas fracamente acoplados. Isso chamamos de Fraco Acoplamento.

(b) - Se refere ao relacionamento que os membros de um módulo possuem, não importa o que módulo significa. Indica se os membros tem uma relação mais direta e importante. Códigos coesos são aqueles de relação forte, onde seus membros estão intimamente ligados e estão ali por um objetivo comum. Membros que não são absolutamente necessários para aquele módulo não devem estar presentes em códigos coesos.

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

preservar o sigilo acerca do código que vai ser implementado na classe.

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

é o uso de software existente, ou do conhecimento de software, para a construção de um novo software.

A reutilização otimiza as quatro variáveis que determinam o sucesso dos projetos de software: qualidade, custo, tempo e produtividade.

A prática da reutilização resulta na retenção do conhecimento na empresa e a coloca numa posição de maior competitividade na medida em que ela passa a conseguir desenvolver projetos de qualidade, no tempo acordado e com menores custos.

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

As etapas de desenvolvimento de software são:

- Fase de diagnóstico

  Essa primeira etapa inicia-se desde o primeiro contato com o problema. É necessário o time de tecnologia e comercial conhecer detalhadamente o cliente e seu problema, visando extrair o máximo de informações para o melhor aproveitamento nas próximas etapas do desenvolvimento.

- Concepção

  Feito um (ou mais) diagnósticos com o cliente acerca de seu problema ou produto, é feita, com os dados iniciais coletados, o processo de concepção. Nessa etapa, tem-se como objetivo de: 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 e análise de requisitos

Com as ideias da solução já bem definidas, prototipadas e arquitetadas, é feito o levantamento e análise de requisitos. A equipe comercial e desenvolvedores constroem um documento com as informações detalhadas da solução, listando todas as funcionalidades do sistema a ser criado. Mesmo com o cliente tendo validado a etapa anterior (concepção), é importante também seu veredito final em relação às funcionalidades inclusas nesse documento de requisitos.

- Fase de desenvolvimento

  É nessa etapa em que as primeiras linhas de código começam a ser escritas. Como anteriormente todo o sistema já foi documentado (visualmente e arquiteturalmente), resta aos devs criarem o produto. Vale ressaltar aqui o uso de metodologias ágeis para a melhor e mais rápida criação do projeto, como a metodologia Scrum, em que o projeto é divido em diversas tarefas (sprints) para realizar entregas menores, porém mais eficientes, do produto para o cliente e às lojas virtuais de aplicativos.

- Etapa de manutenção

  Com a entrega e implementação do produto finalizado para o cliente, inicia-se um período de manutenção do produto, isso quer dizer, realizar ajustes no sistema, normalmente arrumando bugs ou modificações menores para a melhor experiência dos usuários e do cliente.

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

- Verificação

  Fizemos o software corretamente?

  Esta atividade se resume em responder a esta pergunta. A verificação tem o objetivo de avaliar se o que foi planejado realmente foi realizado. Ou seja, se os requisitos e funcionalidades documentados foram implementados, além disso a verificação também pode ser realizada para especificação de sistemas, para avaliar se os requisitos estão sendo documentados como deveriam e ainda prever falhas ou inconsistências entre requisitos.

- Validação

  Fizemos o software correto?

A validação tem o objetivo de avaliar se o que foi entregue atende as expectativas do cliente. Ou seja, se os requisitos, independente do que foi planejado, estão sendo implementados para atender a regra de negócio do cliente, se o sistema é realmente aquilo que o cliente quer e está pagando para ter. A validação final do sistema é realizada pelo próprio cliente ou usuário.

Q.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) - O primeiro teste que é realizado quando vamos desenvolver um sistema, é o Teste de Unidade. Que fornece valores válidos ou inválidos, verificando se o retorno foi de acordo com o esperado. Não exigindo que o software esteja em uma etapa avançada para ser realizado. A partir da implementação da primeira classe ou sistema já é possível executa-lo.

(b) - Analisa os requisitos do sistema, ou seja, se o software cumpre as funções que deve executar. Tais funcionalidades e ações foram previamente definidas na etapa de especificação de requisitos.

É importante lembrar que não existe contato com o código-fonte neste tipo de teste. Assim, o que é validado é o resultado, através da exploração da ferramenta.

(c) - Consiste em módulos que são integrados e testados em grupo. Como por exemplo seu software, acessando um banco de dados ou fazendo uma chamada externa a outros sistemas.

(d) -É o processo de testar o sistema por completo. Verificando se os componentes são compatíveis, se eles interagem corretamente e se transferem os dados certos no momento certo. Tem como objetivo executar o software sob o ponto de vista do seu usuário final, realizando o teste do sistema.

(e) - Tem como objetivo executar o sistema sob o ponto de vista do usuário final. O cliente avalia o sistema e após a experiência de utilização tem a responsabilidade de aceitar ou não o sistema que foi entregue e pelo qual foi pago.

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

(a) - Dentre as técnicas de teste de software, esta utiliza o código-fonte do sistema com a finalidade de analisar os componentes. É também chamado de teste estrutural pois analisa o fluxo de dados, a qualidade da estruturação deste código, segurança, complexidade da manutenção, entre outros.

(b) - Também conhecido como teste funcional, analisa os requisitos do sistema, ou seja, se o software cumpre as funções que deve executar. Tais funcionalidades e ações foram previamente definidas na etapa de especificação de requisitos.

É importante lembrar que não existe contato com o código-fonte neste tipo de teste. Assim, o que é validado é o resultado, através da exploração da ferramenta.

(c) - A técnica de teste de Caixa Cinza junta a Caixa Branca e a Caixa Preta. Dessa forma, são submetidos a análise tanto a estrutura do código-fonte quanto o cumprimento das funções do sistema.

A finalidade de utilizar a Caixa Cinza é buscar por erros que o sistema pode apresentar. A forma com que é feita é denominada de engenharia reversa. Ainda, tem a finalidade de compreender o que gerou as falhas e erros do sistema para posteriormente serem ajustados.