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

Engenharia de software é o estudo a respeito do uso de ferramentas e técnicas para o desenvolvimento de software de qualidade.

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

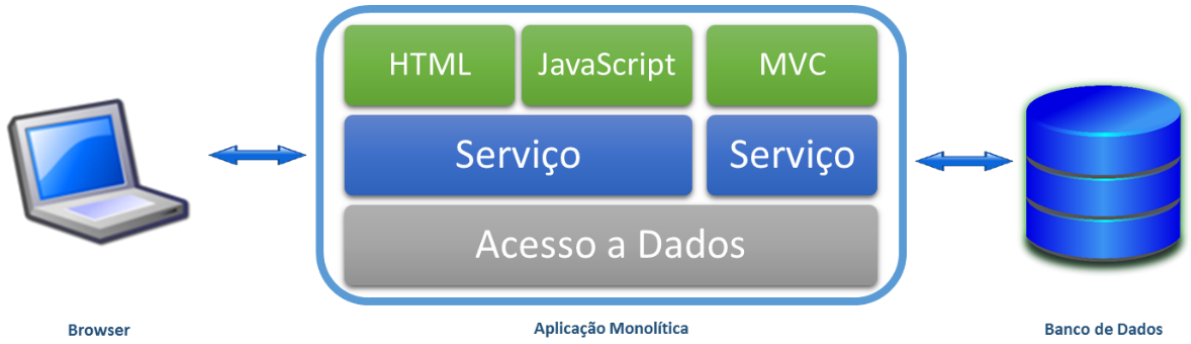
Segundo o PMBOK, o projeto é a idealização e organização para se criar algo novo.

3. O que é arquitetura de software?

Arquitetura de software é uma forma em que o software será desenvolvido, as ferramentas que serão utilizadas, a funcionalidade de cada etapa da estrutura e a integração dos seus componentes.

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**



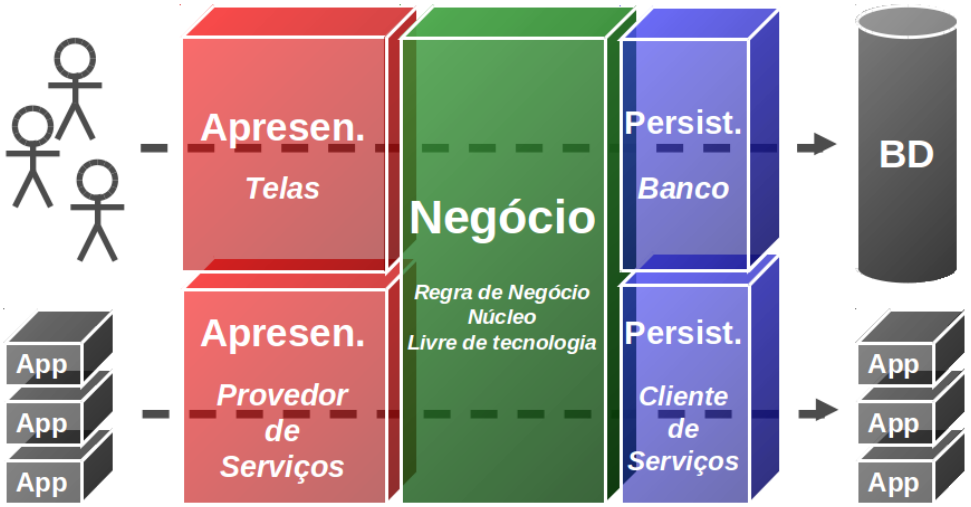
A arquitetura monolítica é uma forma de desenvolvimento de software em que apesar de ser dividida em módulos como na figura acima, os módulos se juntam em um único executável. A vantagem dessa arquitetura é que é simples e facilmente aplicável, porém os módulos são codependêntes, o software é pouco flexível e escalonável.

- **Arquitetura cliente-servidor**

Figura A.1.1 - Arquitetura cliente/servidor
Fonte: do Autor

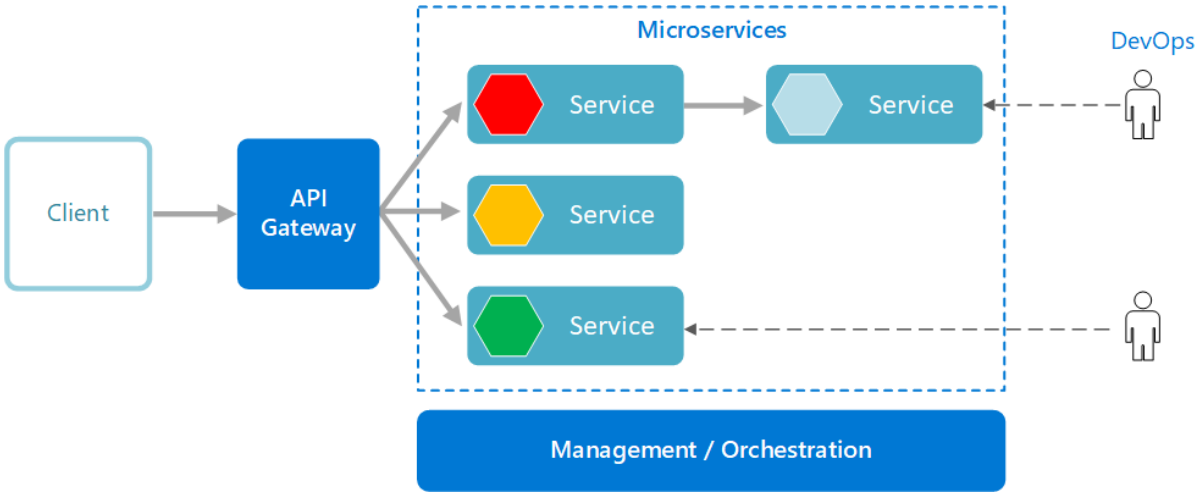
É uma arquitetura de software em que a obtenção de dados e resposta fica com o cliente e o processamento dos dados fica com o servidor. A vantagem é que é mais centralizado e de fácil manutenção, contudo fica dependente da conexão do cliente com o servidor e o servidor fica sobrecarregado por atender múltiplos clientes.

- **Arquitetura em camadas**



É uma arquitetura de software em que os módulos são divididos em camadas hierárquicas que tratam a informação de forma diferentes e complementares a anterior. A vantagem é a fácil desenvolvimento e teste do software, contudo sua escalabilidade é difícil e baixa performance na maioria dos casos.

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



- Cada serviço é uma base de código separado, que pode ser gerenciado por uma equipe de desenvolvimento pequena.
- Os serviços podem ser implantados de maneira independente. Uma equipe pode atualizar um serviço existente sem recompilar e reimplantar o aplicativo inteiro.

- Os serviços são responsáveis por manter seus próprios dados ou o estado externo. Isso é diferente do modelo tradicional, em que uma camada de dados separada lida com a persistência de dados.
- Os serviços comunicam-se entre si por meio de APIs bem definidas. Detalhes da implementação interna de cada serviço ficam ocultos de outros serviços.
- Suporte à programação poliglota. Por exemplo, os serviços não precisam compartilhar a mesma pilha de tecnologia, bibliotecas ou estruturas.

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 principal diferença entre framework e bibliotecas é que a biblioteca é um conjunto de funções para uma aplicação parecida, como a biblioteca de js Moment.js , já o framework é mais abrangente, pode possuir até várias bibliotecas dentro dele, mas também APIs e funções. como o framework de js Angular.

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

APIs é a forma que componentes de softwares se comunicam através de protocolos e conjuntos de definições.

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

Acoplamento fraco é aquele em que cada um dos seus componentes tem ou faz uso de pouco ou nenhum conhecimento das definições de outros componentes separados. Alta coesão é quando os integrantes de um componente estão relacionados a um tema comum, isto é tem o mesmo objetivo, fazem uma única coisa.

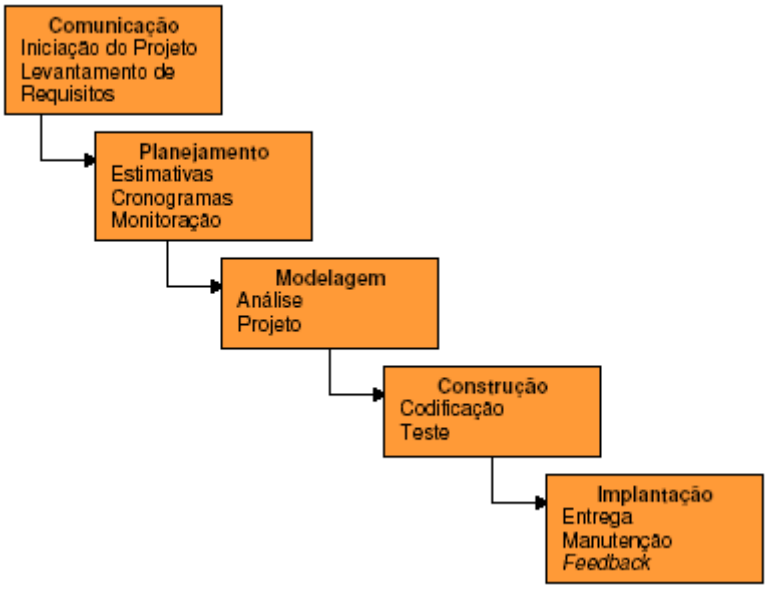
10. O desenvolvedor de software é aconselhado a sempre separar a interface de um programa (API) da sua implementação. Por que?

Pois a API sofrerá atualizações e mudanças e o software precisa manter seu bom funcionamento apesar disso, logo é melhor separá-la na implementação.

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

É a generalização da funcionalidade dos componentes que estão sendo desenvolvidos para o software, isso é bom pois permite a reutilização desses códigos em outros desenvolvimentos, dessa forma agilizando-os.

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



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

Primeiro, verificamos se o que foi planejado foi realizado, depois validamos se as características definidas no planejamento estão presentes no resultado final.

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.

O teste de unidade ou de módulo, verifica o funcionamento da menor unidade de um código testável da aplicação

O teste funcional baseia seus casos de teste nas especificações do componente de software sob teste.

O teste de integração testa funcionalidades inteiras e a integração dos componentes do software.

O teste sistêmico se concentra no comportamento e nas capacidades de todo um sistema ou produto.

O teste de aceitação pode produzir informações para avaliar a situação do sistema para implantação e uso pelo cliente.

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

White Box Testing está testando a codificação interna e infra-estrutura de uma solução de software. Concentra-se principalmente no fortalecimento do salvaguardar, o fluxo de entradas e saídas através da aplicação e a melhoria do design e usabilidade. O teste de caixa branca também é conhecido como teste Clear Box, teste Open Box, teste estrutural, teste baseado em código e teste de caixa de vidro.

O oposto do teste de caixa branca é teste de caixa preta. Isso está sendo testado de uma perspectiva externa ou do usuário final. O teste do Whitebox, por outro lado, é baseado na operação interna de um aplicativo e é sobre testes internos.

O teste da caixa cinza é quando o testador tem um entendimento parcial da estrutura interna do sistema em teste. O teste de caixa cinza é um processo para depurar aplicativos de software, fazendo uma entrada pelo front-end e verificando os dados no back-end.