Nome: Viviane J. Santos **Docente:** Ivan. F. Borges

Exercício 3 : Organização de arquitetura de sistemas

Parte Teórica:

1. O que é arquitetura de sistemas e qual a sua importância no desenvolvimento de software?

R: É um conjunto de decisões de design que define a estrutura, organização e comportamento do sistema. Envolve a organização dos componentes, as interações entre eles e as diretrizes que orientam o desenvolvimento e a evolução do sistema. Além de definir como seus componentes funcionam em conjunto para atender aos requisitos do sistema.
Sua importância está na Escalabilidade, Manutenibilidade, Desempenho, Segurança. Pois isso permite que a arquitetura do sistema cresça, seja mantido com facilidade, ofereça desempenho eficiente e proteja dados contra acessos não autorizados.

2. Qual é a principal diferença entre arquitetura monolítica e arquitetura de microsserviços?

R: Arquitetura Monolítica: Sistema onde todas as funcionalidades e componentes estão interligados em um único bloco de código.
Arquitetura de Microsserviços: Divide o sistema em pequenos serviços independentes, cada um com uma funcionalidade específica, comunicando-se através de APIs.
Na arquitetura monolítica, o desenvolvedor pode rastrear a movimentação de dados ou examinar o comportamento do código no mesmo ambiente de programação. Enquanto isso, identificar problemas de codificação em uma arquitetura de microsserviços exige a análise de vários serviços individuais facilmente acoplados.

3. Explique o conceito de "Separação de Preocupações" e dê um exemplo de como ele pode ser aplicado.

R: A Separação de Preocupações visa separar as diferentes responsabilidades do sistema, como a interface com o usuário, a lógica de negócios e o acesso a dados, garantindo que cada componente seja responsável por uma única função específica. Além disso, busca maximizar a flexibilidade e a adaptabilidade do sistema, permitindo que novas funcionalidades possam ser adicionadas e modificações possam ser feitas com facilidade, sem afetar outras partes do software.
Portanto, ela é um pilar fundamental no desenvolvimento de sistemas eficientes, flexíveis e sustentáveis e desempenha um papel vital na construção de soluções tecnológicas de sucesso e no atendimento das demandas cada vez mais complexas.
Ex.: Em uma aplicação web de e-commerce, as funcionalidades podem ser divididas em: Camada de Apresentação (Frontend), Camada de Negócio (Backend), Camada de Dados (Banco de Dados) e API.
Isso melhora a manutenibilidade e escalabilidade do sistema, facilita o desenvolvimento e teste em paralelo, e permite a reutilização de componentes. Por exemplo, a lógica de negócios pode ser modificada sem impactar a interface do usuário.

4. O que são coesão e acoplamento, e por que é importante manter alta coesão e baixo acoplamento em um sistema?
R: Coesão: Grau de relacionamento entre as responsabilidades de um componente. Alta coesão significa que as funções de um componente estão intimamente relacionadas e contribuem para um único propósito bem definido.

Acoplamento: Grau de dependência entre diferentes componentes do sistema. Baixo acoplamento é desejável, pois indica que os componentes podem funcionar de forma relativamente independente.

5. Qual a diferença entre escalabilidade horizontal e escalabilidade vertical? Dê um exemplo para cada.
R: Escalabilidade Vertical: significa adicionar recursos em um único nó do sistema (mais memória ou um disco rígido mais rápido).
Ex.: Atualizar o hardware de um servidor de banco de dados para uma versão mais poderosa para lidar com um maior volume de consultas.

Escalabilidade Horizontal: significa adicionar mais nós ao sistema, tais como um novo computador com uma aplicação para clusterizar o software.
Ex.: Um site de mídia social que aumenta o número de servidores web para lidar com um aumento no tráfego de usuários durante eventos populares.

6. O que são os 'Design Patterns'? Cite um exemplo.
R: São soluções testadas e comprovadas para problemas comuns no design de software. Eles não são pedaços de código prontos para copiar e colar, mas sim guias ou templates para resolver problemas recorrentes de forma eficiente. Os padrões de projeto ajudam a melhorar a reutilização de código, facilitar a comunicação entre desenvolvedores e tornar o código mais legível e fácil de manter.
Ex.: Design Patterns **=>** Factory Method **= Ex.: Uma** fábrica de carros que pode criar diferentes tipos de carros (esportivos, sedãs, etc.) com base em uma configuração.

7. Explique a importância de implementar práticas de segurança na arquitetura de um sistema. Cite pelo menos duas práticas comuns de segurança.
R: É fundamental para visualizar como diferentes elementos podem ser articulados em conjunto, proporcionando os benefícios esperados para a proteção de dados e evitar ciberataques e perdas de informações. Isso porque os projetos podem ser concebidos equivocadamente, com soluções que não se comunicam bem entre si. Por isso, esse é um primeiro passo importante a fim de organizar um planejamento eficientemente nas empresas.
Autenticação e Autorização: Essas práticas são usadas para garantir que apenas usuários autorizados tenham acesso a um sistema. Elas incluem medidas como senhas fortes, autenticação de dois fatores e controle de acesso baseado em funções.
Ex.: O RBAC é um padrão que permite que os administradores controlem o acesso de usuários a recursos em um sistema com base em suas funções ou cargos. Em vez de conceder acesso individualmente a cada usuário, o RBAC atribui funções específicas a cada usuário, com base nas suas necessidades e responsabilidades.

Isso facilita controlar quem tem acesso a quais recursos e ajuda a evitar a concessão de privilégios excessivos a usuários.

8. Como a arquitetura em nuvem ajuda a aumentar a escalabilidade e a flexibilidade dos sistemas? Cite um modelo de serviço em nuvem e um modelo de implantação.

R: Ela utiliza recursos computacionais fornecidos sob demanda por um provedor de serviços de nuvem via internet. Isso permite uma escalabilidade flexível, otimização de custos e agilidade no fornecimento de serviços.

Ex.: Modelo de Serviço em Nuvem: PaaS (Platform as a Service) - Plataforma como Serviço = Permite aos desenvolvedores criar, testar e implantar aplicações sem gerenciar a infraestrutura subjacente (servidores, armazenamento, etc.). (Google App Engine, Microsoft Azure App Services, Heroku. Desenvolvedores podem usar PaaS para criar e hospedar aplicativos web, focando no desenvolvimento sem se preocupar com a gestão do sistema operacional e da infraestrutura.).

Modelo de Implantação: Nuvem Pública = Serviços são fornecidos por terceiros (como AWS, Google Cloud, Microsoft Azure) e compartilhados entre múltiplos clientes. Ideal para empresas que precisam de escalabilidade rápida e custos reduzidos.

(Uma empresa de desenvolvimento de software utiliza serviços de armazenamento e servidores de uma nuvem pública para hospedar seu ambiente de teste e produção.).

<u>Parte Prática:</u>

Objetivo: Criar um pequeno projeto com arquitetura em camadas usando Node.js e Express. O projeto deve incluir:

- **Uma Camada de Apresentação: Uma API RESTful com rotas simples (GET, POST) para gerenciar um recurso (por exemplo, livros).**
- **Uma Camada de Aplicação: Implementar a lógica de negócios, como validação de dados e processamento de regras de negócios.**
- **Uma Camada de Dados: Usar um array em memória para simular um banco de dados, onde os dados dos livros são armazenados e recuperados.**

Estrutura do Projeto:

- **controllers/: Contém os controladores que lidam com as requisições HTTP.**
- **services/: Contém a lógica de negócios.**
- **data/: Contém os dados ou integrações com o banco de dados (simulado).**
- **app.js: Arquivo principal que configura o servidor e as rotas.**