Programação WEB: Arquitetura tradicional e moderna (SOFEA)

ATIVIDADE 2

Prof.º Denilce de Almeida Oliveira Veloso

Disciplina: Programação WEB

Lucas José Marcondes Rossi 0030482121023

Sorocaba

Setembro/2022

1. Introdução

Neste trabalho serão apresentadas definições e importância da escolha de uma arquitetura de software dentre as disponíveis hoje: arquitetura tradicional e moderna. Contendo, também, suas diferenças, semelhanças, seus benefícios e desvantagens. Para iniciar, precisamos entender o que é a arquitetura de software. É a estrutura básica que define como um aplicativo é construído, como seus componentes se comunicam e como as diferentes partes do aplicativo se integram para fornecer a funcionalidade desejada. A arquitetura de software pode ser dividida em diferentes tipos, incluindo a arquitetura tradicional e a arquitetura moderna, como a Arquitetura Orientada a Serviços (SOA) e a Arquitetura SOFEA. Para escolher a arquitetura de software adequada, é preciso considerar alguns critérios como: requisitos do projeto, complexidade do software, escalabilidade, integração com outras tecnologias, orçamento disponível, prazo de entrega, experiência da equipe de desenvolvimento, entre outros. É importante analisar esses fatores para escolher a arquitetura que melhor atenda às necessidades do projeto e garanta sua eficiência e manutenibilidade.

2. Desenvolvimento

2.1 Arquitetura tradicional

Nesta arquitetura de software, o código do sistema é feito em um único grande bloco, de forma monolítica, geralmente em um único servidor, onde todas as funcionalidades são implementadas neste mesmo sistema. Nessa arquitetura, a lógica de negócios e a interface do usuário são implementadas juntas, o que dificulta a separação e reutilização dessas partes.

Todo o processamento de solicitações do usuário é realizado pelo lado do servidor, onde este é responsável por processar estas solicitações e fornecer uma resposta. A interface para o usuário geralmente é apresentada através de um navegador web utilizando tecnologias específicas para tal, como: Javascript, HTML e CSS; e se comunica com o servidor por meio de solicitações HTTP.

Esta arquitetura consiste em um modelo cliente-servidor, em que o papel do cliente é fornecer solicitações ao servidor e receber as respostas geradas, já o servidor deve ser capaz de processar as requisições do cliente e devolver respostas.

2.1.1 Desvantagens

O método de arquitetura tradicional pode ser difícil de escalar e atualizar. Como todos os componentes do sistema estão integrados em um único servidor ou aplicação, é difícil escalar componentes individuais sem afetar o desempenho do sistema como um todo. Além disso, a atualização de um único componente pode exigir a atualização de todo o sistema.

2.1.2 Benefícios

A arquitetura tradicional pode ser benéfica para sistemas mais simples e com menor complexidade. Seu modelo de desenvolvimento monolítico é mais fácil de entender e gerenciar, tornando a manutenção mais simples. Além disso, a arquitetura tradicional pode ser mais adequada para sistemas que não precisam ser escaláveis ou atualizados com frequência. Outra vantagem é que a integração entre diferentes componentes pode ser mais fácil de implementar, já que todos os componentes estão integrados em um único

sistema. Por fim, a arquitetura tradicional pode ser mais adequada para projetos com prazos de entrega mais curtos, já que sua implementação é geralmente mais rápida.

2.2 Arquitetura moderna (SOFEA)

A arquitetura moderna se difere da tradicional, pois separa a lógica de negócios em serviços independentes, tornando-os mais modularizados e escaláveis. A arquitetura SOFEA (Separation of Front-End and Back-End Architecture) é baseada no conceito de separar a lógica do front-end (Interface do usuário) da lógica back-end (processamento de dados e requisições), que por sua vez pode estar separada em diversos serviços, como por exemplo a solicitação de uma previsão do clima de uma certa região pode ser requisitada a partir de uma lógica já realizada por terceiros (por exemplo, um serviço de meteorologia), onde apenas acessamos quando desejamos através de uma API, ao invés de nós mesmos criarmos a lógica em nosso próprio sistema.

2.2.1 Desvantagens

Algumas desvantagens da arquitetura moderna incluem a complexidade de desenvolvimento e gerenciamento de sistemas distribuídos, a necessidade de uma boa documentação e coordenação entre equipes, a dificuldade de garantir a integridade e segurança dos dados em sistemas distribuídos e a necessidade de investimentos em infraestrutura de rede e servidores. Além disso, a adoção da arquitetura moderna pode requerer uma mudança cultural na equipe de desenvolvimento e na organização como um todo.

2.2.2 Benefícios

A arquitetura moderna oferece diversos benefícios, como a modularização e escalabilidade dos serviços, separação da lógica de front-end e back-end, utilização de serviços web e APIs para comunicação, possibilitando integrações com terceiros, além de permitir uma maior flexibilidade e agilidade na atualização e manutenção do sistema.

3. Conclusão

Observamos que as arquiteturas de software são de extrema importância para a criação, manutenção e desempenho de nossas aplicações web. Para escolhermos a mais adequada que viabilize o nosso projeto, é necessário saber primeiro, através da análise de requisitos, o que o nosso software fará, se ele será complexo e se será necessário “escalá-lo” para mais pessoas. Como vimos, existem “duas opções” (existem vários tipos de arquiteturas modernas, por isto às aspas): a arquitetura tradicional e a moderna. A arquitetura tradicional é recomendada para empresas que não desejam, ou não precisem, de uma grande escala ou flexibilidade. Já a arquitetura moderna, é recomendada para projetos que requerem uma abordagem mais modular e escalável, onde os serviços podem ser facilmente atualizados ou substituídos sem afetar todo o sistema. Essa arquitetura é adequada para projetos complexos que exigem uma alta disponibilidade e flexibilidade, como sistemas bancários, de telecomunicações, de e-commerce, entre outros. Além disso, essa arquitetura é especialmente útil em projetos em que o desenvolvimento e manutenção são realizados por equipes distribuídas, pois permite que diferentes partes do sistema sejam desenvolvidas e mantidas independentemente umas das outras.

4. Referências

OPENAI. As informações neste documento foram obtidas de uma conversa com ChatGPT, um modelo de linguagem natural treinado pela OpenAI. 2023. Mensagem instantânea. Disponível em: <https://chat.openai.com/chat>. Acesso em: 27 fev. 2023.