APRESENTAÇÃO
DE APOIO

ARQUITETURA SERVER-SIDE

Adriana Cássia Reis dos Santos – Aula 01

PUCRS online uol edtech

Professores

ADRIANA CÁSSIA REIS DOS SANTOS
Professora Convidada

Adriana Cássia iniciou sua carreira na área de TI aos 14 anos, em seu primeiro estágio do Ensino Médio como Suporte Técnico, onde permaneceu por seis anos. Após se graduar em Engenharia da Computação, passou a atuar como Analista de Sistemas. Realizou cursos e formações voltadas para programação, entre eles certificação Java (SCJP), migrando para a área de Programação, onde tem atuado nos últimos anos.

MIGUEL GOMES XAVIER
Professor PUCRS

Possui mestrado em Ciência da Computação pela Pontifícia Universidade Católica do Rio Grande do Sul e está cursando doutorado em Ciência da Computação na mesma instituição, atuando principalmente nas áreas de alto desempenho, sistemas distribuídos, virtualização e cloud computing. Atualmente participa de projetos de pesquisa em cooperação com diferentes universidades envolvendo gerência de recursos em arquiteturas de alto desempenho. Tem participado de projetos de análise de dados (BigData), realizando contribuições científicas em prol do avanço da área na indústria e na academia.

Ementa da disciplina

Estudo sobre Arquitetura cliente-servidor para aplicações web. Introdução aos frameworks MVC server-side: Node.js, Express, Nestjs. Estudo de programação assíncrona e programação reativa. Desenvolvimento de aplicações web com o conceito de uso de serviços.

PUCRS online

Arquitetura Server-Side

Por Professora Adriana Cássia

Pós-graduação em Desenvolvimento Full Stack

EMENTA
DA DISCIPLINA

- ARQUITETURA CLIENTE-SERVIDOR PARA APLICAÇÕES WEB
- INTRODUÇÃO AOS FRAMEWORKS MVC SERVER-SIDE: NODE.JS, EXPRESS, NESTJS
- ESTUDO DE PROGRAMAÇÃO ASSÍNCRONA E PROGRAMAÇÃO REATIVA
- DESENVOLVIMENTO DE APLICAÇÕES WEB COM O CONCEITO DE USO DE SERVIÇOS

ARQUITETURA CLIENTE-SERVIDOR

- A Internet
- Web
- Arquitetura cliente-servidor
- SOA e Web Services

A Internet é a rede mundial de computadores. Uma rede que engloba muitas outras redes.

A Internet é a infraestrutura para a World Wide Web.

Web em constante evolução

A Web é um Sistema em constante evolução para a publicação e aceso a recursos e serviços através da internet.

Acesso à Web

A maneira mais comum de acesso a Web é através dos navegadores (Browsers).

EDGE

Chrome

Firefox

Safari

Opera

WEB: Onde, quando e porquê?

A Web nasceu no CERN (Centro Europeu de Pesquisa Nuclear) em 1989.

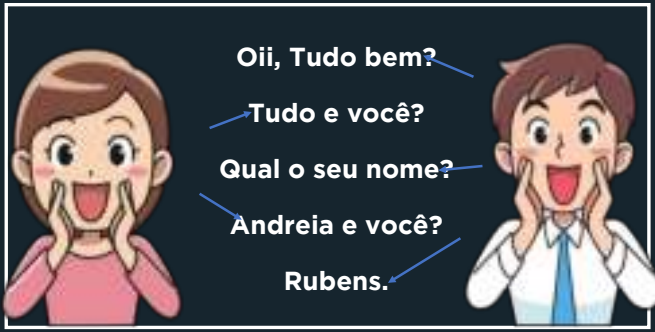
Era usada para compartilhamento de documentos entre fisicos.

WEB: Componentes

- HTML, CSS e JavaScript
- Protocolo HTTP
- URL´s
- Arquitetura de Sistema cliente-servidor
- Web Services

Protocolo

Conjunto de regras sobre o modo de como dará a comunicação entre as partes envolvidas.



Protocolo HTTP

O HTTP Hypertext Transfer Protocol, significa Protocolo de Transferencia de Hipertexto

É um protocol de comunicação utilizado nas aplicações Web

O HTTP funciona como um protocol de requisição-resposta na arquitetura cliente-servidor

URL´s

Um URL (Uniform Resource Locator) tem como objetivo identificar um recurso.

O esquema amplamente usado para acessar recursos na web é o HTTP.

http://nome-do-servidor{:porta}{/nome-do-caminho}{?consulta}{#fragmento}.orgão-regulamentador

Os itens entre chaves são opcionais.

Cliente-Servidor

Ao diponibilizar uma página na rede você estará compartilhando um recurso ou provendo serviços com várias outras pessoas

Todos que desejam acessar se recurso/serviço fazem o papel de cliente

Você que disponibiliza um recurso/serviço faz o papel de servidor

Cliente-Servidor

Em uma arquitetura cliente-servidor:

SERVIDOR é aquele que fornece recursos ou serviços

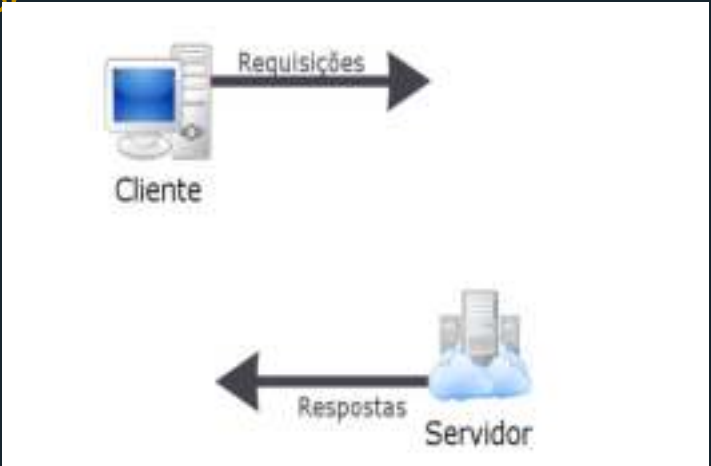
CLIENTE é aquele que requer os recursos ou serviços

Arquitetura Cliente-Servidor

A arquitetura cliente servidor é uma arquitetura de aplicação distribuída, ou seja, na rede existem os fornecedores de recursos ou serviços a rede, que são chamados de servidores, e existem os requerentes dos recursos ou serviços, denominados clientes.

O cliente não compartilha nenhum de seus recursos com o servidor, mas no entanto ele solicita alguma função do servidor, sendo ele, o cliente, responsável por iniciar a comunicação com o servidor, enquanto o mesmo aguarda requisições de entrada.

Arquitetura Cliente-Servidor



Problemas do Modelo (Cliente-Servidor)

Visto que a capacidade de oferecer serviços ou recursos fica centralizada na figura do servidor, surge um problema, o que fazer quando o número de requisições dos clientes ultrapassa a capacidade computacional do servidor?

Essa pergunta não tem uma resposta de alta eficiência para aplicações em redes de computadores. A sobrecarga de servidores é um problema real apesar de a capacidade dos servidores ter aumentado consideravelmente na última década.

Problemas do Modelo (Cliente-Servidor)

Além do fato de algumas requisições não serem atendidas, pela sobrecarga do servidor, no modelo cliente servidor, ainda há o fato de que o modelo não é robusto, ou seja, se um servidor crítico falha, os requisições já feitas pelos clientes, não poderão ser atendidas.

Esse motivos são a base para a concepção das redes Peer-to-peer.

Problemas do Modelo (Cliente-Servidor)

As redes Par-a-Par, do ingês,Peer-to-peer (sigla P2P),surgem como uma proposta de descentralização do monopólio de processamento funcional. fazendo com que sistemas sirvam a outros sistemas, dando a cada estação as mesmas responsabilidades e capacidades dentro da rede.

Problemas do Modelo (Cliente-Servidor)

Dave Winer da UserLand Software sugere que os sistemas P2P envolvam as seguintes características.

1.Interface de troca de arquivos seja fora do navegador de Internet.

2.Computadores podem servir tanto como servidores como clientes

3.Sistemas fáceis de usar e bem integrados.

4.O sistema deve incluir ferramentas que ajudam usuários que queiram criar ou adicionar alguma funcionalidade.

5.Sistemas que promovam conexão entre usuários

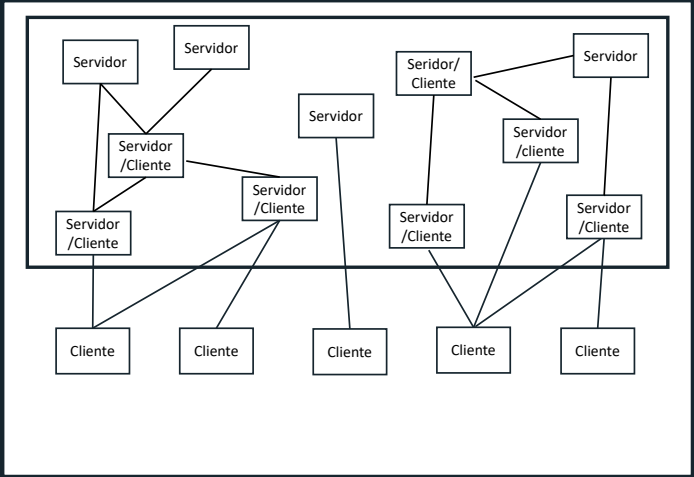
6.Sistemas que façam algo novo ou excitante

SOA

A Arquitetura Orientada a Serviços ou Service-Oriented-Architeture(SOA) é um conjunto de funcionalidades bem definodas em forma de serviços disponibilizados através da rede

Uma entidade pode assumir ambos os papeis (Servidor, cliente ou ambos) ao mesmo tempo, caracterizando a composição de serviços.

Composição de serviços



SOA

Arquitetura orientada a serviços (SOA) é um método de desenvolvimento de software que usa componentes de software chamados de serviços para criar aplicações de negócios. Cada serviço fornece um recurso de negócios, e todos eles também podem se comunicar entre si em diferentes plataformas e linguagens. Os desenvolvedores usam a SOA para reutilizar serviços em sistemas diferentes ou combinar vários serviços independentes para realizar tarefas complexas.

SOA

Por exemplo, vários processos de negócios em uma organização exigem a funcionalidade de autenticação de usuários. Em vez de reescrever o código de autenticação para todos os processos de negócios, você pode criar um único serviço de autenticação e reutilizá-lo para todas as aplicações. Da mesma maneira, quase todos os sistemas em uma organização de saúde, como sistemas de gerenciamento de pacientes e sistemas de prontuário eletrônico de saúde (EHR), precisam registrar pacientes. Eles podem chamar um único serviço

Quais são os benefícios da arquitetura orientada a serviços?

A SOA tem vários benefícios em relação às arquiteturas monolíticas tradicionais, nas quais todos os processos são executados como uma única unidade. Alguns dos principais benefícios da SOA incluem:

Mais rapidez para entrada no mercado

Manutenção eficiente

Maior adaptabilidade

Mais rapidez para entrada no mercado

Os desenvolvedores reutilizam serviços em diferentes processos de negócios para poupar tempo e economizar custos. Eles podem estruturar aplicações muito mais rapidamente com a SOA do que escrevendo código e realizando integrações do zero.

Manutenção eficiente

É mais fácil criar, atualizar e depurar pequenos serviços do que grandes blocos de código em aplicações monolíticas. A modificação de qualquer serviço na SOA não afeta a funcionalidade geral do processo de negócios.

Maior adaptabilidade

A SOA é mais adaptável aos avanços da tecnologia. Você pode modernizar suas aplicações de maneira eficiente e econômica. Por exemplo, as organizações de saúde podem usar a funcionalidade de sistemas de EHR mais antigos em aplicações baseadas na nuvem mais recentes.

Quais são os princípios básicos da arquitetura orientada a serviços?

Não há diretrizes padrão bem definidas para implementar a arquitetura orientada a serviços (SOA). Porém, alguns princípios básicos são comuns em todas as implementações da SOA.

Interoperabilidade
Acoplamento fraco
Abstração
Granularidade

Interoperabilidade

Cada serviço na SOA inclui documentos de descrição que especificam a funcionalidade do serviço e os termos e condições relacionados. Qualquer sistema cliente pode executar um serviço, independentemente da plataforma de base ou da linguagem de programação. Por exemplo, processos de negócios podem usar serviços escritos em C# e Python. Como não há interações diretas, as alterações em um serviço não afetam outros componentes que usam esse serviço.

Acoplamento fraco

Os serviços na SOA devem ter acoplamento fraco, tendo a menor dependência possível de recursos externos, como modelos de dados ou sistemas de informações. Eles também devem ser stateless, sem reter informações de sessões ou transações anteriores. Dessa forma, se você modificar um serviço, ele não afetará significativamente as aplicações cliente e outros serviços que o utilizam.

Abstração

Os clientes ou usuários de serviços na SOA não precisam conhecer a lógica do código ou os detalhes de implementação do serviço. Para eles, os serviços devem parecer uma caixa preta. Os clientes obtêm as informações necessárias sobre o que o serviço faz e como utilizá-lo por meio de contratos de serviço e outros documentos de descrição de serviços.

Granularidade

Os serviços na SOA devem ter um tamanho e escopo apropriados, idealmente empacotando uma única função de negócios distinta por serviço. Os desenvolvedores podem então usar vários serviços a fim de criar um serviço composto para realizar operações complexas.

Quais são os componentes da arquitetura orientada a serviços?

Existem quatro componentes principais na arquitetura orientada a serviços (SOA).

Serviço

- **Implementação do serviço**
- **Contrato de serviço**
- **Interface de serviço**

Provedor de serviços

Consumidor de serviços

Registro de serviços

Serviço

Serviços são os alicerces básicos da SOA. Eles podem ser privados (disponíveis apenas para usuários internos de uma organização) ou públicos (acessíveis pela Internet a todos). Individualmente, cada serviço tem três características principais.

Implementação do serviço - É o código que cria a lógica para realizar a função de serviço específica, como a autenticação de um usuário ou o cálculo de uma fatura.

Contrato de serviço - Define a natureza do serviço e seus termos e condições associados, como os pré-requisitos para usar o serviço, o custo do serviço e a qualidade do serviço prestado.

Interface de serviço - Na SOA, outros serviços ou sistemas se comunicam com um serviço por meio de sua interface de serviço. A interface define como você pode chamar o serviço para realizar atividades ou trocar dados. Ele reduz as dependências entre os serviços

Provedor de serviços

O provedor de serviços cria, mantém e fornece um ou mais serviços que outros usuários podem utilizar. Organizações podem criar seus próprios serviços ou comprá-los de provedores de serviços terceirizados.

Consumidor de serviços

O consumidor de serviços solicita que o provedor de serviços execute um serviço específico. Pode ser um sistema inteiro, uma aplicação ou outro serviço. O contrato de serviço especifica as regras que o provedor e o consumidor de serviços devem seguir ao interagirem entre si. Provedores e consumidores de serviços podem pertencer a diferentes departamentos, organizações e até mesmo setores.

Registro de serviços

Um registro de serviços, ou repositório de serviços, é um diretório de serviços disponíveis acessível pela rede. Ele armazena documentos de descrição de serviço de provedores de serviços. Documentos de descrição contêm informações sobre o serviço e como se comunicar com ele. Os consumidores de serviços podem descobrir facilmente os serviços de que precisam usando o registro de serviços.

Como funciona a arquitetura orientada a serviços?

Na arquitetura orientada a serviços (SOA), os serviços funcionam de maneira independente e fornecem funcionalidades ou troca de dados aos seus consumidores. O consumidor solicita informações e envia dados de entrada ao serviço. O serviço processa esses dados, realiza a tarefa e retorna uma resposta. Por exemplo, se uma aplicação usa um serviço de autorização, ela fornece ao serviço o nome de usuário e a senha. O serviço verifica esses dados e retorna uma resposta apropriada.

Quais são as limitações na implementação da arquitetura orientada a serviços?

Escalabilidade limitada

Aumento das interdependências

Ponto único de falha

Escalabilidade limitada

A escalabilidade do sistema é significativamente afetada quando os serviços compartilham muitos recursos e precisam ser coordenados para executar sua funcionalidade.

Aumento das interdependências

Os sistemas de arquitetura orientada a serviços (SOA) podem se tornar mais complexos ao longo do tempo e desenvolver várias interdependências entre serviços. Eles podem ser difíceis de modificar ou depurar quando vários serviços estão chamando uns aos outros em um loop. Recursos compartilhados, como bancos de dados centralizados, também podem tornar o sistema mais lento.

Ponto único de falha

Para implementações da SOA com um ESB, este último cria um ponto único de falha. Trata-se de um serviço centralizado, o que vai contra a ideia de descentralização que a SOA defende. Clientes e serviços não poderão se comunicar uns com os outros se o ESB se tornar inoperante.

O que são microsserviços?

A arquitetura de microsserviços é formada por componentes de software muito pequenos e completamente independentes, chamados de microsserviços, que se especializam e se concentram em uma única tarefa. Os microsserviços se comunicam por meio de APIs, que são regras criadas pelos desenvolvedores para permitir que outros sistemas de software se comuniquem com os microsserviços deles.

O estilo de arquitetura dos microsserviços é mais adequado para ambientes modernos de computação em nuvem. Eles geralmente operam em contêineres,

Benefícios dos microsserviços

Os microsserviços são escaláveis de maneira independente, rápidos, portáteis e independentes de plataforma, que são características nativas da nuvem. Eles também são desacoplados, o que significa que não dependem ou têm dependência limitada de outros microsserviços. Para conseguir isso, os microsserviços têm acesso local a todos os dados de que precisam, em vez de acesso remoto a dados centralizados que outros sistemas também acessam e utilizam. Isso cria duplicação de dados, que os microsserviços compensam em termos de

SOA em comparação com microsserviços

A arquitetura de microsserviços é uma evolução do estilo de arquitetura da SOA. Os microsserviços abordam as deficiências da SOA para tornar o software mais compatível com ambientes corporativos modernos baseados na nuvem. Eles são refinados e favorecem a duplicação de dados em oposição ao compartilhamento de dados. Isso os torna completamente independentes, com seus próprios protocolos de comunicação que são expostos por meio de APIs leves. Cabe essencialmente aos consumidores usar o

Web Service : o que é, como funciona, para que serve?

Primeiro, a teoria. Um Web service é um conjunto de métodos acedidos e invocados por outros programas utilizando tecnologias Web.

Segundo, a tradução. Um Web service é utilizado para transferir dados através de protocolos de comunicação para diferentes plataformas, independentemente das linguagens de programação utilizadas nessas plataformas.

Web Service : o que é, como funciona, para que serve?

Os Web services funcionam com qualquer sistema operativo, plataforma de hardware ou linguagem de programação de suporte Web. Estes transmitem apenas informação, ou seja, não são aplicações Web que suportam páginas que podem ser acedidas por utilizadores através de navegadores Web.

Os Web services permitem reutilizar sistemas já existentes numa organização e acrescentar-lhes novas funcionalidades sem que seja necessário criar um sistema a partir do zero. Assim, é possível melhorar os

Web Service : Como funciona?

Tendo em conta as operações disponíveis no Web service, a aplicação solicita uma dessas operações. O Web service efetua o processamento e envia os dados para a aplicação que requereu a operação.

A aplicação recebe os dados e faz a sua interpretação, convertendo-os para a sua linguagem própria.

Se são linguagens diferentes, como conseguem comunicar?

É necessário uma linguagem intermédia que garanta a comunicação entre a linguagem do Web service e o sistema que faz o pedido ao Web service. Para tal, existem protocolos de comunicação como o SOAP (Simple Object Access Protocol) e o REST (Representational State Transfer).

O protocolo SOAP utiliza XML para enviar mensagens e, geralmente, serve-se do protocolo HTTP para transportar os dados. Associado ao protocolo SOAP está o documento WSDL (Web Service Definition Language) que descreve a localização do Web service e as operações que dispõe. Além disso, fornece a informação

Se são linguagens diferentes, como conseguem comunicar?

O REST é um protocolo de comunicação mais recente que surgiu com o objetivo de simplificar o acesso aos Web services. Este baseia-se no protocolo HTTP e permite utilizar vários formatos para representação de dados, como JSON (um dos mais utilizados), XML, RSS, entre outros.

Assim, uma das grandes vantagens do REST é a sua flexibilidade, já que não limita os formatos de representação de dados. O protocolo REST é também utilizado quando a performance é importante, uma vez que é um protocolo ágil e com a capacidade de transmitir

Quais os benefícios dos Web services?

A utilização de Web services traz vários benefícios tanto a nível tecnológico, como a nível do negócio. Seguem-se os mais relevantes:

– Integração de informação e sistemas: uma vez que o funcionamento do Web service necessita apenas de tecnologia XML/JSON e protocolos HTTP, a comunicação entre sistemas e aplicações é bastante simplificada. Com um Web service é possível trocar informação entre dois sistemas, sem necessidade de recolher informação detalhada sobre o funcionamento de cada sistema. Os Web services permitem ligar qualquer tipo de sistema, independentemente das plataformas (Windows, Linux

Quais os benefícios dos Web services?

– Reutilização de código: um Web service pode ser utilizado por várias plataformas com diferentes objetivos de negócio. O código do Web service é feito uma vez e pode ser utilizado vezes sem conta por diferentes aplicações.

– Redução do tempo de desenvolvimento: é mais rápido desenvolver com Web services, porque os sistemas não são totalmente construídos a partir do zero e facilmente são incluídas novas funcionalidades. O tempo de implementação de sistemas com a utilização de Web services é mais reduzido, sendo uma boa opção

Quais os benefícios dos Web services?

– Maior segurança: o Web service evita que se comunique diretamente com a base de dados. Assim, a segurança do sistema que fornece os dados está salvaguardada.

– Redução de custos: Com a utilização de Web services não é necessário criar aplicações à medida para a integração de dados, algo que pode ser bastante caro. Os web services tiram partido de protocolos e da infraestrutura Web já existente na organização, requerendo por isso pouco investimento.

Web service: uma solução prática

Numa organização coexistem várias aplicações que organizam e trocam dados, muitas vezes, de formas distintas. Assim, nem sempre garantem a comunicação entre sistemas. Por outro lado, é cada vez mais comum a necessidade de trocar dados entre diferentes sistemas, seja dentro de uma organização ou entre organizações.

Uma solução prática e de baixo custo para solucionar a incompatibilidade de sistemas e garantir a sua comunicação são os Web services. Estes permitem ligar diferentes aplicações que integram um sistema, ultrapassando barreiras como o tipo de plataforma ou

Web service

Ao contrario das aplicações Web convencionais no qual foram projetadas para suportar interações aplicação/usuário (B2C), a tecnologia Web services foi desenvolvida para realizar interações aplicação/aplicação (B2B).

O seu principal objetivo é solicionar o problema de integração de aplixativos heterogêneos (interoperabilidade).

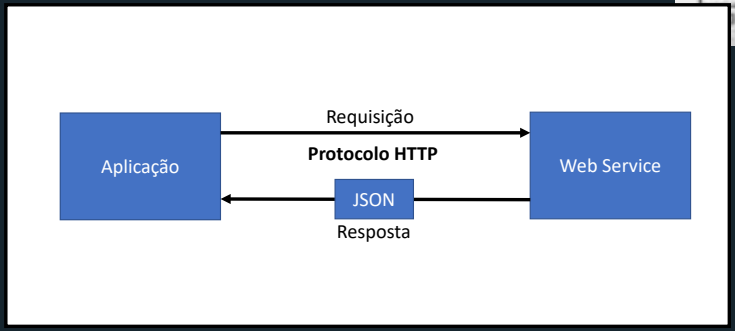
Interoperabilidade

Uma aplicação portável não é necessariamente interoperável.

Igualmente assim, uma aplicação interoperável é necessariamente portável.

Web service


Web Service
Aplicação
Protocolo
JSON


Aplicação
Requisição
Protocolo HTTP
JSON
Resposta
Web Service

JSON

O JavaScripr Object Notation (JSON) é um formato de intercâmbio de dados, ou seja, basicamente um conjunto de chaves e valores, podem ser interpretados por qualquer linguagem (independente de linguagem de programação).

JSON é um formato de dados baseado em texto seguindo a sintaxe de objeto literais do JavaScript.

JSON Sintaxe

A ideia utilizada pelo JSON para representar informações é tremendamente simples: para cada valor representado, atribui-se um nome (ou rótulo) que descreve o seu significado. Esta sintaxe é derivada da forma utilizada pelo JavaScript para representar informações. Por exemplo, para representar o ano de 2012, utiliza-se a seguinte sintaxe:

```
1 | "ano": 2012
```

JSON Sintaxe

Um par nome/valor deve ser representado pelo nome entre aspas duplas, seguido de dois pontos, seguido do valor. Os valores podem possuir apenas 3 tipos básicos: numérico (inteiro ou real), booleano e string. As Listagens 2, 3, 4 e 5 apresentam exemplos. Observe que os valores do tipo string devem ser representados entre aspas

```
1 "altura": 1.72
```

Número real

```
1 "site": "www.devmedia.com.br"
```

String

```
1 "casado": true
```

Booleano

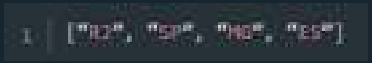
```
1 "temperatura": -2
```

Número Negativo

JSON Sintaxe

A partir dos tipos básicos, é possível construir tipos complexos: array e objeto. Os arrays são delimitados por colchetes, com seus elementos separados entre vírgulas. As listagens 6 e 7 mostram exemplos.

Matriz de Inteiros



Array de Strings

Os objetos são especificados entre chaves e podem ser compostos por múltiplos pares nome/valor, por arrays e também por outros objetos. Desta forma, um objeto JSON pode representar, virtualmente, qualquer tipo de informação! O exemplo da Listagem 8 mostra a

Exemplo JSON

```
{
  "titulo": "JSON x XML",
  "resumo": "o duelo de dois modelos de representação de informações",
  "ano": 2012,
  "genero": ["aventura", "ação", "ficção"]
}
```

MVC

O MVC é um padrão de arquitetura de software. O MVC sugere uma maneira para você pensar na divisão de responsabilidades, principalmente dentro de um software web.
O princípio básico do MVC é a divisão da aplicação em três camadas: a camada de interação do usuário (view), a camada de manipulação dos dados (model) e a camada de controle (controller).

Quais os papéis de cada camada?

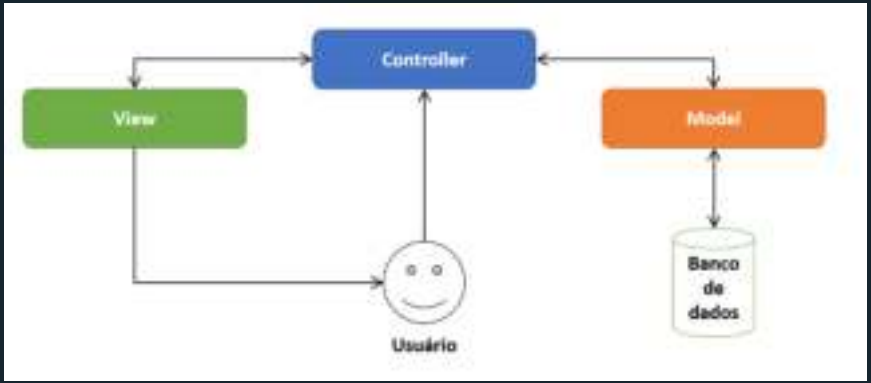
Quando falamos sobre o MVC, cada uma das camadas apresenta geralmente as seguintes responsabilidades:

• Model A responsabilidade dos models é representar o negócio. Também é responsável pelo acesso e manipulação dos dados na sua aplicação.

• View A view é responsável pela interface que será apresentada, mostrando as informações do model para o usuário.

• Controller É a camada de controle, responsável por ligar o model e a view, fazendo com que os models possam ser repassados para as views e vice-versa

Quais os papéis de cada camada?



Framework: o que é e pra que serve essa ferramenta?

O framework nada mais é do que uma ferramenta que vai te ajudar a ter como único objetivo focar em desenvolver o projeto, não em detalhes de configurações.
Vamos pensar que um frame representa a estrutura de uma casa, na qual só existem as paredes de tijolos levantadas ou mesmo um carro somente com a lataria, como vemos nas montadoras. Essas estruturas estão aguardando o desenvolvimento (work), que no exemplo da casa, corresponderia à escolha da massa corrida que será colocada nas paredes, as cores das tintas, os tipos de pisos. No caso do carro, se serão colocados bancos de tecido ou couro, qual será a cor do painel, vidros escuros

Framework: o que é e pra que serve essa ferramenta?

Sendo assim, o framework trouxe a prática de evitar que tenhamos que fazer tarefas repetitivas, automatizando parte do trabalho. Pensando numa situação de desenvolvimento, se precisarmos criar um formulário de cadastro de usuário, ele sempre vai requerer algum tipo de validação como email e senha. O framework já terá essa validação pronta para ser utilizada.

Pode ter surgido uma voz em sua mente dizendo: "Mas posso optar por não fazer validação, ou por uma validação específica". E é verdade! Além das que o framework já disponibiliza, nada te impede de

Quais as vantagens de utilizar frameworks?

É interessante abordarmos esses fatores positivos, pois muitas vezes já as utilizamos mas ainda não sabemos exatamente onde está a vantagem dessa ferramenta. Vamos a elas:

•Sem sombra de dúvidas, quando usamos frameworks conseguimos agilizar nosso trabalho, já que nossos esforços se voltam para o desenvolvimento, em vez de nos preocuparmos tanto com detalhes de configurações e padrões de projeto.

•As comunidades de pessoas programadoras são as responsáveis diretas pelo desenvolvimento desses frameworks. Por essa razão eles se tornam mais seguros e

Quais as vantagens de utilizar frameworks?

- **Com frameworks, temos por padrão um código mais limpo, garantindo maior clareza de entendimento em tudo que é implementado pela ferramenta, o que facilita nosso trabalho e de outras pessoas que lidam com o projeto.**

Existem contras?

Algumas pessoas desenvolvedoras apontam aspectos que podem ser ruins. São eles:

•Problemas de configurações, o que demanda tempo para a manutenção. Tomando como exemplo o Spring, havia um problema comum usando MVC e devido a ele tínhamos que baixar e declarar as dependências, o módulo de injeção de dependência, as dependências dos módulos, usar o Spring Validator para validação de formulários, configurar a camada de visualização com JSP, Velocity ou Thymeleaf. Ou seja, teríamos que passar por todas essas etapas para termos o Spring MVC

Quando trabalhamos com dependências, que são funcionalidades já prontas, precisamos lembrar de algumas questões

- **Não podemos ignorar que o objeto gerador das dependências não está no código do projeto, mas abstraído;**
- **Pode ser uma dificuldade caso você não consiga fornecer as dependências diretamente no código do próprio teste;**
- **O framework resolve primeiro todas as dependências antes de subir a aplicação, o que consome um certo tempo;**
- **Conforme as dependências são inseridas no projeto,**

JavaScript

JavaScript é uma linguagem de programação que originalmente foi desenvolvida para trazer maior interatividade aos websites através da manipulação do DOM (Document Object Model). Vamos conferir a definição do livro que é uma referência na temática, o Eloquent JavaScript:

"O JavaScript foi introduzido em 1995 como uma forma de adicionar dinamicidade à páginas da web no navegador Netscape Navigator. Desde então, a linguagem foi adotada por todos os outros principais navegadores gráficos da web. Ela tornou possíveis o desenvolvimento das aplicações modernas da web - aplicações com os quais você pode interagir diretamente sem recarregar a página a cada ação. JavaScript também é usado em sites mais tradicionais para fornecer várias formas de interatividade de forma mais inteligente."

JavaScript

Portanto, de forma resumida, podemos entender que o JavaScript foi pensado para ser rápido, dinâmico e acessível. A linguagem interpretada possibilita subir ou trabalhar em suas aplicações sem precisar configurar todo um ambiente complexo.

Essas características motivaram o engenheiro de software Ryan Dahl a desenvolver um ambiente, como um programa que você instala no seu computador, que trabalha com a linguagem JavaScript fora do navegador e pelo lado do servidor, via terminal, de uma maneira

Node.JS

O JavaScript nasceu para atender demandas voltadas ao Front e como as necessidades aumentam de acordo com o crescimento tecnológico, surgiu a ideia de utilizar uma mesma linguagem no lado do cliente e do servidor para otimizar processos e serviços. Dessa forma, o Node.JS aparece como uma alternativa viável para programação Back-End por se tratar de um ambiente para desenvolvimento utilizando a linguagem JavaScript.

De acordo com sua definição oficial, o Node é um runtime, que nada mais é do que um conjunto de códigos, API's, ou seja, são bibliotecas responsáveis pelo tempo de execução (é o que faz o seu programa rodar)

Node.JS

É importante frisar que o Node.JS é um ambiente de execução assíncrono, isto é, ele trabalha de modo a não bloquear no momento da execução da aplicação, delegando os processos demorados a um segundo plano.

Ele dá muito certo com os servidores de arquitetura "single threaded", isto significa que todos os pedidos para o servidor são executados no mesmo tópico - em vez de serem gerados em processos separados. Um dos grandes diferenciais da parceria Node.JS e Javascript é o bom desempenho no uso de APIs, já que o Javascript faz bastante uso de APIs assíncronas.

Como funciona o Node.JS?

O Node é capaz de interpretar um código JavaScript, igual ao que o navegador faz. Sendo assim, quando o navegador recebe um comando em JavaScript, ele o interpreta e depois executa as instruções fornecidas.

O Node torna possível o envio de instruções (os nossos códigos) sem precisar de um navegador ativo, basta ter o Node.JS instalado e utilizar o terminal para executar um programa construído em JavaScript.

Além disso, você pode utilizar apenas uma linguagem de

Como funciona o Node.JS?

Para que todo esse processo seja possível de ocorrer fora do navegador, o Node utiliza uma outra ferramenta chamada de Chrome's V8 JavaScript Engine. É esse motor V8 do Chrome que compila e executa o código JavaScript no lugar de apenas interpretá-lo.

Ainda parece muito confuso? Vem comigo que te explico melhor...
O motor V8 da Google é o centro, o coração que processa todo o código JavaScript do navegador para que sua máquina compreenda e disponibilize os recursos e interações. Só é possível visualizar páginas em JavaScript

Características do Node.JS

Node.JS pode ser utilizado nas famosas APIS Rest, web scrapping, chatbots, IoT, web servers, aplicações Desktop, tudo devido a sua característica altamente versátil. Dentre as principais, podemos citar:

- **Multiplataforma:** permite criar desde aplicativos desktop, aplicativos móveis e até sites SaaS;

- **Multi-paradigma:** é possível programar em diferentes paradigmas, como: Orientado a Objetos, funcional, imperativo e dirigido à eventos;

- **Open Source:** é uma plataforma de código aberto, isso significa que você pode ter acesso ao código fonte do Node.JS e realizar suas próprias customizações ou mesmo contribuir para a comunidade de forma direta;

- **Escalável:** Node.JS foi criado para construir aplicações web escaláveis

Express

O Express.js é um Framework rápido e um dos mais utilizados em conjunto com o Node.js, facilitando no desenvolvimento de aplicações back-end e até, em conjunto com sistemas de templates, aplicações full-stack.

Escrito em JavaScript, o Express.js é utilizado por diversas empresas ao redor do mundo, dentre elas a Fox Sports, PayPal, IBM, Uber, entre outras.

Muito popular tanto em grandes empresas quanto na comunidade, o Express facilita a criação de aplicações

Quando esse framework foi criado?

Lançado como um código aberto sob a chancela do MIT (Massachusetts Institute of Technology), uma das maiores instituições de tecnologia do mundo, o Express JS foi criado em 2010 pelo desenvolvedor TJ Holowaychuk. Esse profissional já havia sido responsável por criar diversos códigos com a linguagem de programação JavaScript.

O surgimento desse framework trouxe mais agilidade para os desenvolvedores, uma vez que ele fornece diversos recursos tanto para aplicativos mobile quanto

Para que serve o Express.js?

Como já mencionamos, o Express JS é um framework que serve para os desenvolvedores construírem aplicativos. Ele funciona como um simplificador de tarefas que são comuns no desenvolvimento, até mesmo de programas Web. Entre essas tarefas, podemos citar ainda a melhoria da segurança dos programas, movimentações de banco de dados, roteamento de URLs, entre outras.

Onde o Express.js pode ser utilizado?

O ExpressJS pode ser usado em projetos pensados para micro-serviços, ou seja, quando você divide toda a aplicação em subserviços. Isso facilita bastante o dia a dia, pois cada um desses pequenos setores podem ser gerenciados por equipes focadas apenas naquela solução. Isso permite maior integração e resulta em um aplicativo de maior qualidade. Muitas empresas já usam dessa forma, desde startups até gigantes como a Netflix.

Características do Express.js

O Express é um framework incrível e possui diversas características que facilitam o desenvolvimento de nossas aplicações. Dentre suas principais características, podemos citar:

•Facilidade na criação de páginas por conta da integração de vários sistemas;
•Conta com um sistema de rota completo;
•É possível fazer o gerenciamento distinto de requisições HTTP em verbos diferentes;
•Ideal para criar aplicações com mais rapidez, ainda que com um conjunto pequeno de pastas e arquivos;
•Permite exceções nas aplicações

Framework opinativos x não opinativo

Quem trabalha com frameworks precisa saber que existem dois tipos: opinativo e não opinativo. O Express JS é não opinativo. Abaixo explicamos as diferenças entre eles:

Opinativo:
Como o próprio nome diz, são frameworks que fazem sugestões para que o desenvolvedor possa escolher o melhor caminho em algumas tarefas. Isso ocorre porque algumas ações dentro do código já são bem compreendidas, o que facilita a vida na hora do desenvolvimento. Entretanto, a flexibilidade para resolver problemas é menor, especialmente quando é fora do domínio principal.

Não opinativo:
Ao contrário do primeiro, o framework não opinativo conta com restrições menores na hora de indicar a melhor forma de utilização de

Qual a relação de Express.js e Node.js?

Apesar de tanto o Express JS como o Node JS serem diferentes, eles se igualam no quesito popularidade entre os devs e, também, quando o assunto é eficiência. Apesar de serem frameworks de trabalho, eles possuem algumas diferenças importantes. Por isso, quem deve escolher qual usar é o próprio desenvolvedor, pois um pode ser melhor que o outro dependendo do seu objetivo. E isso vale também para o Node JS Express.

E as vantagens e desvantagens desse framework?

Uma das principais vantagens é que o Express JS possui uma quantidade significativa de bibliotecas e plugins. Além disso, é possível usar qualquer banco de dados e integrar com praticamente todo tipo de serviço. Um outro ponto positivo diz respeito à sua comunidade, que é bem grande e popular.

Com relação às desvantagens, podemos citar que o desenvolvedor não possui tanta liberdade na hora de fazer a estrutura de uma aplicação web, ou seja, é um pouco limitado nesse sentido. Outro fator importante é que ele é mais indicado para aplicações menores. Em uma

Nest.js

Nest.js é um framework progressivo para desenvolvimento em TypeScript. Em suma, ele se tornou bastante popular nos últimos anos.

Para saber o que é Nest.js antes de baixá-lo na sua máquina, é bom entender que Nest.js é um framework back-end feito com Node.js e que utiliza padrão TypeScript.

Basicamente, ele serve para criar aplicações escaláveis, confiáveis e mais eficientes, como o próprio framework se apresenta no seu site oficial.
A sua sintaxe é muito parecida com o Angular, que é um

Diferença entre Nest.js e Next.js

Basicamente, Nest.js e Next.js são frameworks para desenvolvedores Full-stack. Entretanto, há algumas diferenças.

Nest.js é usado do lado do servidor e pode resolver todos os problemas de back-end, ao passo que Next.js é uma pequena estrutura para aplicativos JavaScript que são renderizados pelo servidor.

Tanto um quanto o outro são recursos de código aberto. Em termos comparativos, Nest.js tem 17,4 mil estrelas no GitHub, enquanto Next.js é mais popular, somando 38,7

Diferença entre Nest.js e Next.js

Recursos do Nest.js:

- **Progressivo:** ele aproveita os recursos JavaScript e traz soluções para o Node.js;
- **Versátil:** tem um ecossistema adaptável e pode ser usado em todos os tipos de aplicativos;
- **Extensível:** permite o uso de qualquer biblioteca devido à sua arquitetura modular.

Recursos do Next.js:

- **Prático:** ele usa apenas JavaScript e tem configuração zero, facilitando a vida do developer;
- **Amigável:** além de completo, ele é fácil de manusear;
- **Funcional:** ele tem renderização automática de servidor e divisão de código.

Programação assincrona

A programação assincrona é uma forma de evitar delays ou tempos de espera na execução de um programa. Quando estamos executando algo sinconicamente, podemos ter bloqueios no processo pela necessidade de esperar alguma execução de código. Isso pode bloquear o programa como um todo até que termine a execução deste passo.

Quando usar a programação assíncrona?

Podemos usar programação assíncrona sempre que tivermos um procedimento que possa ser independete, tais como:

- **Leitura e escrita de um arquivo**
- **Chamadas de recursos 3rd party**
- **Lógicas independentes que podem ser separadas da execução da thead principal.**

Programação reativa

É o ato de programar por meio de fluxos de dados assíncronos, ou seja, que não são realizados simultaneamente ou seguindo o mesmo ritmo de desenvolvimento em relação a outro sistema.

No modelo tradicional de Web Development, por exemplo, diversas tarefas são criadas e se comunicam em tempos determinados, recebendo respostas sem escalabilidade no caso de ocorrer uma demanda maior ou alguma falha. Isto é, elas seguem regras diretas.

Embora esse modelo ainda seja utilizado, hoje em dia sua lógica não é mais compatível com a realidade em que

Quais os pilares da programação reativa?

Existem quatro importantes fundamentos que sustentam a metodologia da programação reativa:

•elasticidade — reagir à carga ou demanda por meio do uso de variados núcleos e servidores;
•Message Drive — reagir a eventos em vez de precisar seguir ordens síncronas, já que os sistemas são compostos por gerenciadores de eventos não-bloqueantes e assíncronos;
•responsividade — reagir aos usuários, oferecendo aplicações que proporcionem interações e experiências enriquecedoras e em tempo real;
•resiliência — aplicações capazes de reagir às falhas e de se recuperar de falhas de conectividade, software e hardware.

Como ela se diferencia da programação clássica?

Para entender o contraste entre o modelo clássico de Web Development e a programação reativa, acompanhe o seguinte raciocínio: digamos que a metodologia tradicional seja como o processo de construir uma parede. Assim que as informações são enviadas, afirmando que a mistura do cimento está sendo preparada, a função "construir a parede" é chamada e, posteriormente, é iniciada a função do reboco.

Então, é preciso esperar por um determinado período. Assim que este período acaba, é hora de ativar a função de pintura. Mas e se, após a pintura ser feita, você se

Como ela se diferencia da programação clássica?

No formato clássico, todo o sistema estaria perdido se houvesse alguma falha de comunicação ou algum processo fosse “atropelado”. Ou seja, todo o trabalho precisaria ser refeito.

Na programação reativa, o mesmo procedimento pode ser feito, mas com muito mais inteligência e autonomia, de modo que tudo esteja interligado em paralelo, sem que seja necessário seguir uma ordem linear e cronológica, como no método tradicional.

Em outras palavras, é possível chegar ao resultado final, que é a parede construída e funcionando, sem precisar

Quais os benefícios de utilizar a programação reativa?

Entre as maiores vantagens do uso de programação reativa para desenvolver as tecnologias de uma empresa, podemos citar a utilização mais inteligente dos recursos computacionais em termos de hardware multi-CPU e multicore.

O aumento da produtividade dos desenvolvedores também é outro grande benefício, já que utiliza uma abordagem mais direta e de fácil manutenção para lidar com a IO assíncrona e não-bloqueante, diferente de como ocorre na programação tradicional, garantindo muito mais agilidade aos processos.

Além disso, a programação reativa se destaca quando falamos sobre a criação e composição de componentes de

O que são microsserviços?

Microsserviços são uma abordagem arquitetônica e organizacional do desenvolvimento de software na qual o software consiste em pequenos serviços independentes que se comunicam usando APIs bem definidas. Esses serviços pertencem a pequenas equipes autossuficientes.

As arquiteturas de microsserviços facilitam a escalabilidade e agilizam o desenvolvimento de aplicativos, habilitando a inovação e acelerando o tempo de introdução de novos recursos no mercado.

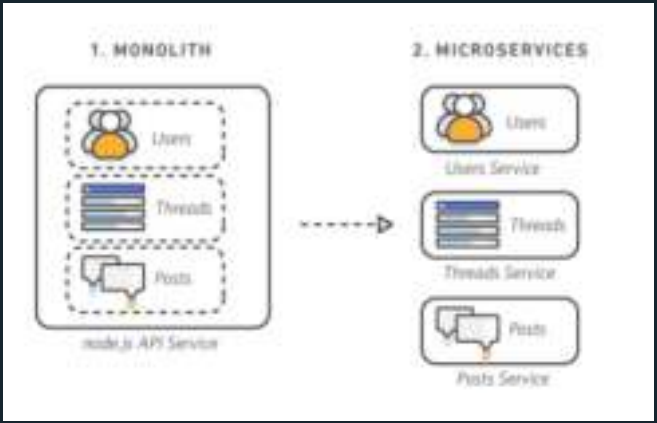
Diferenças entre as arquiteturas monolítica e de microsserviços

Com as arquiteturas monolíticas, todos os processos são altamente acoplados e executam como um único serviço. Isso significa que se um processo do aplicativo apresentar um pico de demanda, toda a arquitetura deverá ser escalada. A complexidade da adição ou do aprimoramento de recursos de aplicativos monolíticos aumenta com o crescimento da base de código. Essa complexidade limita a experimentação e dificulta a implementação de novas ideias. As arquiteturas monolíticas aumentam o risco de disponibilidade de aplicativos, pois muitos processos dependentes e altamente acoplados aumentam o impacto da falha de um

Diferenças entre as arquiteturas monolítica e de microsserviços

Com uma arquitetura de microsserviços, um aplicativo é criado como componentes independentes que executam cada processo do aplicativo como um serviço. Esses serviços se comunicam por meio de uma interface bem definida usando APIs leves. Os serviços são criados para recursos empresariais e cada serviço realiza uma única função. Como são executados de forma independente, cada serviço pode ser atualizado, implantado e escalado para atender a demanda de funções específicas de um aplicativo.

Diferenças entre as arquiteturas monolítica e de microsserviços



Características dos microsserviços

Autônomos

Cada serviço do componente de uma arquitetura de microsserviços pode ser desenvolvido, implantado, operado e escalado sem afetar o funcionamento de outros serviços. Os serviços não precisam compartilhar nenhum código ou implementação com os outros serviços. Todas as comunicações entre componentes individuais ocorrem por meio de APIs bem definidas.

Especializados

Cada serviço é projetado para ter um conjunto de recursos e é dedicado à solução de um problema específico. Se os desenvolvedores acrescentarem mais código a um serviço ao

Benefícios dos microsserviços

Agilidade

Os microsserviços promovem uma organização de equipes pequenas e independentes que são proprietárias de seus serviços. As equipes atuam dentro de um contexto pequeno e claramente compreendido e têm autonomia para trabalhar de forma mais independente e rápida. O resultado é a aceleração dos ciclos de desenvolvimento. Você obtém benefícios significativos do throughput agregado da organização.

Benefícios dos microsserviços

Escalabilidade flexível

Os microsserviços permitem que cada serviço seja escalado de forma independente para atender à demanda do recurso de aplicativo oferecido por esse serviço. Isso permite que as equipes dimensionem corretamente as necessidades de infraestrutura, meçam com precisão o custo de um recurso e mantenham a disponibilidade quando um serviço experimenta um pico de demanda.

Benefícios dos microsserviços

Fácil implantação

Os microsserviços permitem a integração e a entrega contínuas, o que facilita o teste de novas ideias e sua reversão caso algo não funcione corretamente. O baixo custo de falha permite a experimentação, facilita a atualização do código e acelera o tempo de introdução de novos recursos no mercado.

Benefícios dos microsserviços

Liberdade tecnológica

As arquiteturas de microsserviços não seguem uma abordagem generalista. As equipes são livres para escolher a melhor ferramenta para resolver problemas específicos. O resultado é que as equipes que criam microsserviços podem optar pela melhor ferramenta para cada tarefa.

Benefícios dos microsserviços

Código reutilizável

A divisão do software em módulos pequenos e bem definidos permite que as equipes usem funções para várias finalidades. Um serviço criado para uma determinada função pode ser usado como componente básico para outro recurso. Isso permite que os aplicativos sejam reutilizados, pois os desenvolvedores podem criar recursos sem precisar escrever código.

Benefícios dos microsserviços

Resiliência

A independência do serviço aumenta a resistência a falhas do aplicativo. Em uma arquitetura monolítica, a falha de um único componente poderá causar a falha de todo o aplicativo. Com os microsserviços, os aplicativos lidam com a falha total do serviço degradando a funcionalidade, sem interromper todo o aplicativo

PUCRS online uol edtech.