Diego L. Alexandre

diego.alexandre.work@gmail.com

Resumo

[Atraia o leitor com um resumo envolvente, em geral, uma rápida visão geral do documento.   
Quando estiver pronto para adicionar conteúdo, basta clicar aqui e começar a digitar.]

Fundamentos da Arquitetura de sistemas

Bootcamp Decola Tech – Avanade | DIO

**Capitulo 1**

**O que são Web Services?**

* Serviços web ou **Web Services**, são soluções para aplicações se comunicarem independentemente de linguagem, softwares e hardwares utilizados.
* Inicialmente Serviços Web foram criados para troca de mensagens utilizando a linguagem **XML** (Extensive Markup Language) sobre o protocolo **HTTP** sendo identificado como **URI** (Uniform Resource Identifier)
* Podemos dizer que os Serviços Web são API’s que se comunicam por meio de redes cobre protocolo HTTP.

Vantagens

* Linguagem comum
* Integração
* Reutilização de Implementação
* Segurança
* Custos

Principais tecnologias

* SOAP
* REST
* XML
* JSON

**Estrutura SOAP**

**O que é o SOAP?**

* SOAP – Simple Object Access Protocol
* É um protocolo baseado em XML para acessar serviços web, principalmente por HTTP.
* Pode se dizer que SOAP é uma definição de como serviços web se comunicam.
* Foi desenvolvido para facilitar integrações entre aplicações.

**Vantagens**

* Permite integrações entre aplicações, independentemente de linguagem, pois utiliza como linguagem comum o XML.
* É independente de plataforma e software.
* Meio de transporte genérico, ou seja, pode ser usado por outros protocolos além do HTTP.

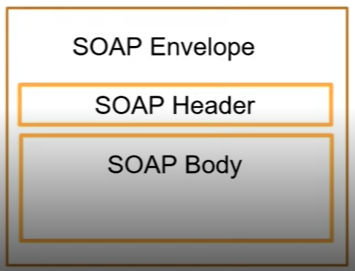
**XML**

**O que é XML?**

* XML – Extensible Markup Language
* É uma linguagem de marcação criada na década de 90 pela W3C.
* Facilita a separação de conteúdo
* Não tem limitações de criação de tags
* Linguagem comum para integração entre aplicações.

Obs: a W3C criou o XML e o SOAP

**Estrutura do SOAP**



* **SOAP Envelope** é o primeiro elemento do documento e é usado para encapsular toda mensagem SOAP.
* **SOAP Header** é o elemento onde possui informações de atributos e metadados da requisição. Exemplo: IP de origem, DNS, credenciais de autenticação, Token...
* **SOAP Body** é o elemento que contém os detalhes da mensagem.

**Mensagem SOAP**



**Entendendo o que é WSDL e XSD**

**O que é o WSDL?**

* Web Services Description Language
* Usado para descrever Web Services, funciona como um contrato de serviço.
* A descrição é feita em um documento XML, onde é descrito o serviço, especificações de acesso, operações e métodos.

**O que é um XSD**

* XSD – XML Schema Definition
* É um schema no formato XML usado para definir a estrutura de dados que será validada no XML.
* O XSD funciona como uma documentação de como deve ser montado o SOAP Message (XML) que será enviado através de Web Service.

URI usada no exercício: www.soapclient.com/xml/soapresponder.wsdl

Ferramentas que ajudam a ler a documentação: SOAP UI (www.soapui.org)

**SOAP na prática**

Joga a URI no software SOAP UI que ele facilita a compreensão. Exercício para começar a imersão.

**Aprenda o que REST, API e JSON**

**O que é REST?**

* REST – Representational State Transfer
* É um estilo de arquitetura de software que define a implementação de um serviço web
* Podem trabalhar com os formatos XML, JSON ou outros.

**Vantagens de se utilizar REST?**

* Permite integrações entre aplicações e também entre cliente e servidor em páginas web e aplicações.
* Utiliza os métodos HTTP para definir a operação que está sendo efetuada.
* Arquitetura de fácil compreensão.

**O que é API?**

“ Quando uma aplicação web disponibiliza um conjunto de rotinas e padrões através de serviços web podemos chamar esse conjunto de API ”

* API – Application Programming Interface.
* São conjuntos de rotinas documentados e disponibilizados por uma aplicação para que outras aplicações possam assumir suas funcionalidades.
* Ficou popular com o aumento dos serviços web.
* As maiores plataformas de tecnologia disponibilizam APIs para acessos de suas funcionalidades, algumas delas são: Facebook, Twitter, Telegram, Whatsapp e Github...

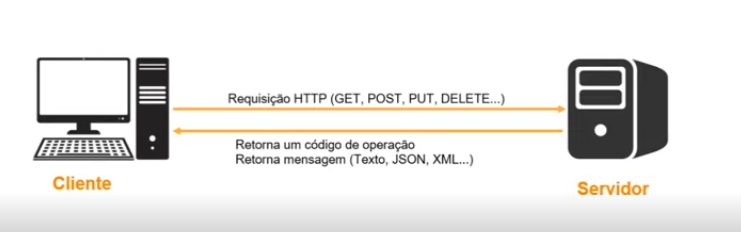
**Principais Métodos HTTP**

* **GET** – Solicita a representação de um recurso
* **POST** – Solicita a criação e um recurso
* **DELETE** – Solicita e exclusão de um recurso
* **PUT** – Solicita a atualização de um recurso

**Entendendo os métodos utilizados pelo REST**

O REST usa dos **métodos HTTP** exemplificados anteriormente.

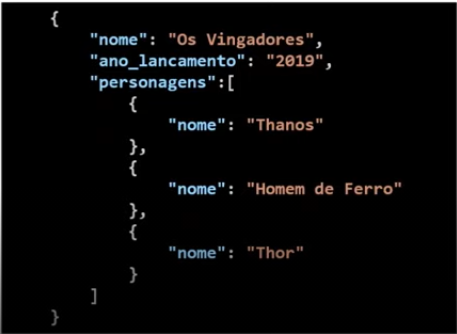
**Estrutura de um REST**



**JSON**

Estrutura de um JSON

* JSON – JavaScript Object Notation
* Formatação leve usada para troca de mensagens entre sistemas
* Usa-se de uma estrutura de chave e valor e também de listas ordenadas.
* Um dos formatos mais populares e mais utilizados para troca de mensagens entre sistemas.



**Veja sobre integração com REST e métodos HTTP na prática**

Integração com serviços REST

Métodos HTTP na prática

Código de estado HTTP

* Usado pelo servidor para avisar ao cliente sobre o estado da operação solicitada.
* 1xx – Informativo
* 2xx – Sucesso
* 3xx – Redirecionamento
* 4xx – Erro do Cliente
* 5xx – Erro de servidor

Mozila HTTP Codes – para estudar sobre os códigos HTTP vistos com mais profundidade.

Software Postman

**Conceitos de arquitetura em aplicações para a internet**

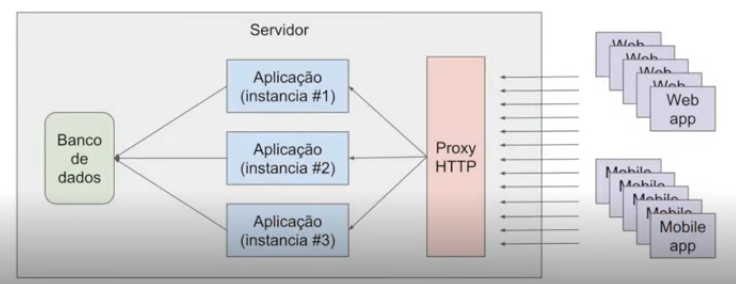
1. Tipos de arquiteturas
2. Comparativos entre os tipos de arquitetura
3. Gerenciamento de erros e volume de acesso

Requisitos Básicos

* Conhecimento básico sobre protocolo HTTP e Proxy
* Entendimento sobre RestAPI
* Conhecimento sobre Docker
* Saber programar

1. **Tipos de arquiteturas**

**Monolito**



Prós e Contras

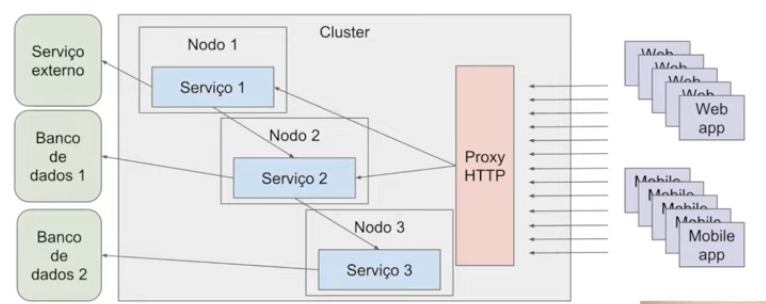
Pros

* Baixa complexidade
* Monitoramento Simplificado

Contra

* Stack única
* Compartilhamento de recursos
* Acoplamento
* Mais complexo a escalabilidade

**Microserviço #1**



Pros e Contras

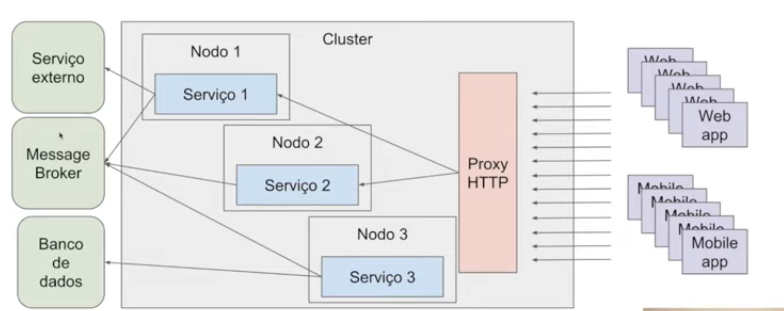
Pros

* Stack dinâmica
* Simples escalabilidade

Contras

* Acoplamento
* Monitoramento mais complexo
* Provisionamento mais complexo

**Microserviço #2**



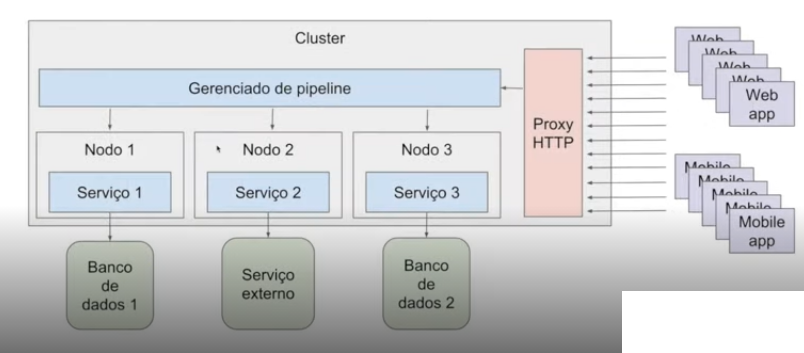
Pros

* Stack dinâmica
* Simples escalabilidade
* Desacoplamento

Contras

* Monitoramento mais complexo
* Provisionamento mais complexo

Microserviço #3



Pros

* Stack Dinâmica
* Simples Escalabilidade
* Desacoplamento
* Menor Complexidade

Contras

* Provisionamento mais complexo
* Plataforma inteira depende do gerenciador de pipeline.

1. **Gerenciamento de erros e volume de acesso**

**Onde é mais complexo**

* Processos assíncronos (Microsserviços #2)
* Pipeline

Solução:

* Dead Letter queue
* Filas re-tentativas

1. **Suporte aos Exercícios**

Abra o repositório no Github

<https://github.com/jeffhsta/fundamentos_arquitetura>

**Arquitetura de aplicações móveis e internet das coisas**

O conceito de internet das coisas

**Por que conectar as coisas?**

* Embutir sensores em objetos do dia a dia
* Coletar dados dos sensores
* Usar dados para tomar decisão

Conceitos básicos de IoT

* Things
* Cloud
* Intelligence

Exemplos: Smart Buildings, Smart House, Wearables(coisas que nós usamos e que podem ser usados(smart watch, cartões, etc) com sensores para coletar dados corporais: locais que frequentamos, horários. ), agricultura, smart transportation (veículos inteligentes, como da Tesla), RFID Supply Chain (cadeias de suprimento), Energy Efficiency, Computação Umbíqua (estado em que a tecnologia é tão fundamental para a nossa vida e da sociedade que ela foge do nosso campo de visão de tão difundida que ela está)

Desafios da Internet das Coisas

* Privacidade e Segurança
* Quantidade exponencial de dispositivos conectados na rede
* Ser capaz de processar e armazenar uma enorme quantidade de informações
* Gerar valor a partir dos dados coletados.

Arquitetura da internet das coisas e protocolo de comunicação

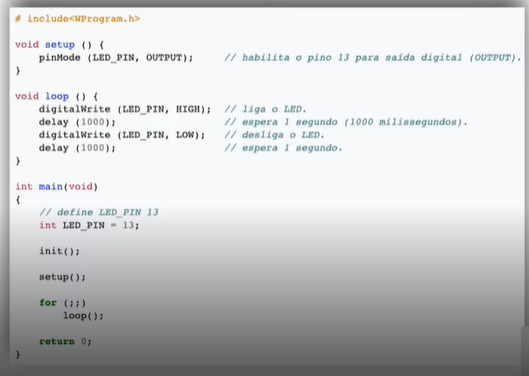
Conectando as coisas...

Considere esses atributos na escolha:

* Baixo consumo de energia
* Rede de dados limitado
* Resiliencia
* Segurança
* Customização
* Baixo custo

Exemplo de plataformas

* Arduino: plataforma de prototipagem com entradas/saída. Desenvolvedor desenvolve escrevendo em C/C++, Interface Serial USB, Shields



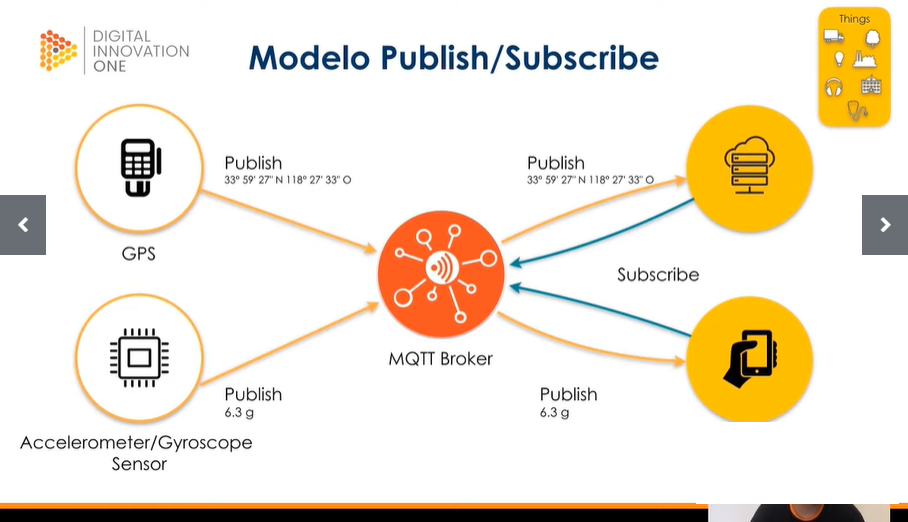
* Embarcados: Microcontrolador de chip único. Sistema Operacional real time, Embarcado, Uso industrial, médico, militar, transporte,
* Rapsberry Pi: Computador completo, Hardware integrado em uma única placa, roda SO Windows ou Linux, uso doméstico e comercial

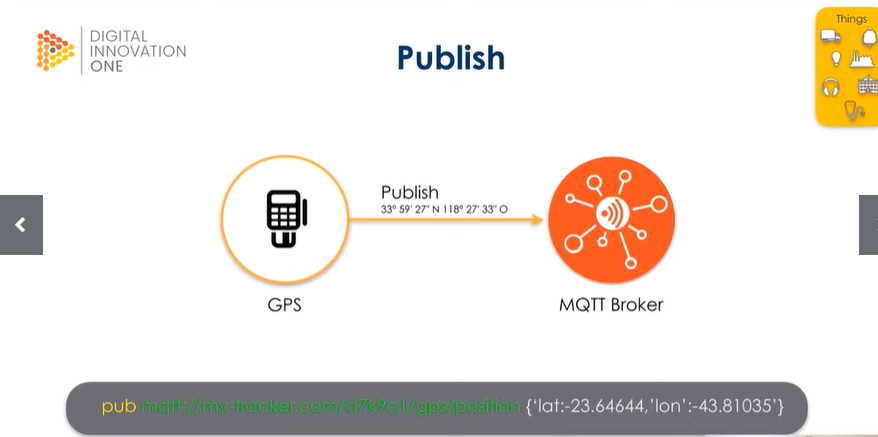
O protocolo de comunicação

GPS Tracker, Smartphone

Protocolo MQTT

* Protocolo mais usado em IoT
* Pode ser usado no Android para se comunicar com a nuvem
* Base na pilha do TCP/IP
* Protocolo de mensagem assíncrona (M2M), ou seja, um protocolo que eu envio e não fico aguardando resposta.
* Criado pela IBM para conectar sensores de pipelines de petróleo a satélites
* Padrão OASIS suportado pelas linguagens de programação mais populares.





O Broker está no código verde.

D7k9a1 = identificador do usuário

A latitude e longitude(em branco, são os tópicos)

O que são tópicos

Um tópico é como se fosse um endereço, onde eu vou entregar um determinado tipo de informação.

