Sistemas Distribuídos

1. Metas/Desafios
2. Arquiteturas
3. Processos
4. Protocolos

O que é um Sistema Distribuído?

“Um Sistema Distribuído é um conjunto de computadores independentes que se apresenta a seus usuários de maneira única e coerente”, ou seja, um sistema composto por vários computadores mas que mesmo assim parece que é apenas um único sistema.

- Característica principal: Diferença entre vários computadores e no modo em que eles se comunicam, em grande parte, estar oculta para os usuários.
- Para suportar essa diferença de vários computadores, Sistemas Distribuídos normalmente são organizados em um **Middleware.**

Metas e Desafios

Sistemas Distribuído devem atingir as seguintes quatro metas (não necessariamente precisa ser feito em um bom nível, pode ser apenas cumprido superficialmente, mas deve ser cumprido):

1. Acesso a recursos;
2. Transparência da distribuição;
3. Abertura;
4. Escalabilidade.

Acesso a Recursos

- Facilitar ao usuário e as aplicações acessos aos recursos;
- Principal motivo de compartilhar recursos: é econômico. É mais barato duas impressoras no qual todo mundo tem acesso para um escritório inteiro ou uma impressora para cada mesa do escritório? Por isso faz mais

sentido compartilhar recursos computacionais (armazenamento, recursos, dados e etc).

- Compartilhamento de recursos facilita também a troca de informações;
- Quanto maior o compartilhamento de recursos, maior deve ser o investimento em segurança.

Transparência de distribuição

- Ocultar que os recursos e processos estão fisicamente em vários computadores;
- Existem vários tipos de transparência, entre eles os mais importantes:

1. Acesso: Ocultar a diferença na representação de dados e no modo de acesso a um recurso, em nível básico, desejamos ocultar as diferenças de arquiteturas, por exemplo modo como arquivos são manipulados;
2. Localização: Ocultar lugar onde está o recurso, pode-se atingir essa transparência não dando indicios de onde está o recurso, por exemplo em um URL, https://www.servidor.com/index.html não dá indicios de onde está o arquivo;
3. Migração: Oculta a movimentação do recurso de uma localização para outra, sem afetar também o modo em que eles são acessados;
4. Relocação: Parecido com Transparência de Migração, porém este é quando o recurso está em uso;
5. Replicação: Ocultar que um recurso é/foi replicado (reproduzido, copiado), o objetivo desse tipo é aumentar a disponibilidade do recurso, por exemplo, a Netflix transferir filmes para o Brasil para que os usuários brasileiros não precisem ficar esperando o sistema buscar esses filmes nos Estados Unidos para poder exibir. Para ocultar essa replicação, as réplicas devem ter o mesmo nome;
6. Concorrência: Oculta que o recurso pode estar sendo concorrido entre vários usuários (por exemplo, Leitura e Escrita). Para que o recurso fique consistente, é necessário trativas de acesso, por exemplo, cada usuário tenha acesso exclusivo a um recurso.
7. Ocultar falhas e recuperação de recursos (Conceito similar de Recuperação de desastres na Nuvem), por exemplo, esconder do usuário que um recurso parou de funcionar.

Existem momentos em que ocultar totalmente do usuário não é uma boa ideia, visar a transparência de distribuição pode ser bom para um sistema, porém deve ser levado em conta aspectos como desempenho, por exemplo. Será que vale a pena fazer a replicação de recursos complexos constantemente do Japão para o Brasil e vice-versa?

Abertura

Um **Sistema Distribuído Aberto** é um sistema que oferece serviços de acordo com regras padronizadas que descrevam a semântica e sintaxe desses serviços, no qual são especificados por meio de interfaces em uma **linguagem de definição de interface (Interface Definition Language).**

Especificações adequadas são **completas e neutras**.

- Completas: Tudo que é necessário para ser implementado foi especificado;
- Neutras: Não prescrever a aparência da implementação;

Ambas essas caracteristicas são importantes para Interoperabilidade e Portabilidade;

- Interoperabilidade: Caracteriza até que ponto duas implementações podem coexistir e trabalhar em conjunto;
- Portabilidade: Caracteriza até que ponto uma aplicação desenvolvida para um sistema distribuído A pode ser executada e implementada sem modificação em um sistema distribuído B que implementa as mesmas interfaces de A.

Um **sistema distribuido aberto** também deve se**r extensível**, ou seja, deve ser fácil para adicionar partes que são executadas em um sistema operacional diferente.

Escalabilidade

Como a conectividade mundial pela internet vem se tornando muito comum, a escalabilidade é uma meta importante para um sistema distribuído.
Escalabilidade essa que pode ser medida em três dimensões:

1. Tamanho: Facilidade em adicionar usuários e recursos;
2. Distância Geográfica: Usuários e recursos podem estar longe um do outro;
3. Gerenciamento: Facilidade em gerenciar e administrar sistemas.

É necessário resolver problemas quando se trata de ampliação de sistemas, considerando a escalabilidade em relação ao tamanho: para suportar mais usuários e recursos, os serviços, os dados e os algoritmos centralizados ficam muito limitados.

> 💡 Antes de seguir, **o que são Serviços, Dados e Algoritmos Centralizados?**
> Serviços Centralizados: Serviços fornecidos por uma única entidade, por exemplo, um único servidor para todos os usuário;
> Dados Centralizados: Está relacionado a prática de adicionar todos os dados em um único local ou em conjunto de locais controlados por uma única entidade, por exemplo, uma lista telefônica online;
> Algorítmos Centralizados: São algoritmos de processamentos de dados (ou também inteligências artificiais que são executados em um único local, por exemplo, o sistema de recomendação da Amazon, do Youtube, o Google Search e etc.

O problema fica evidente quando nos tratamos de ambientes centralizados, uma vez que a quantidade de usuários e recursos aumentando faz com que o único servidor apresente gargalos.

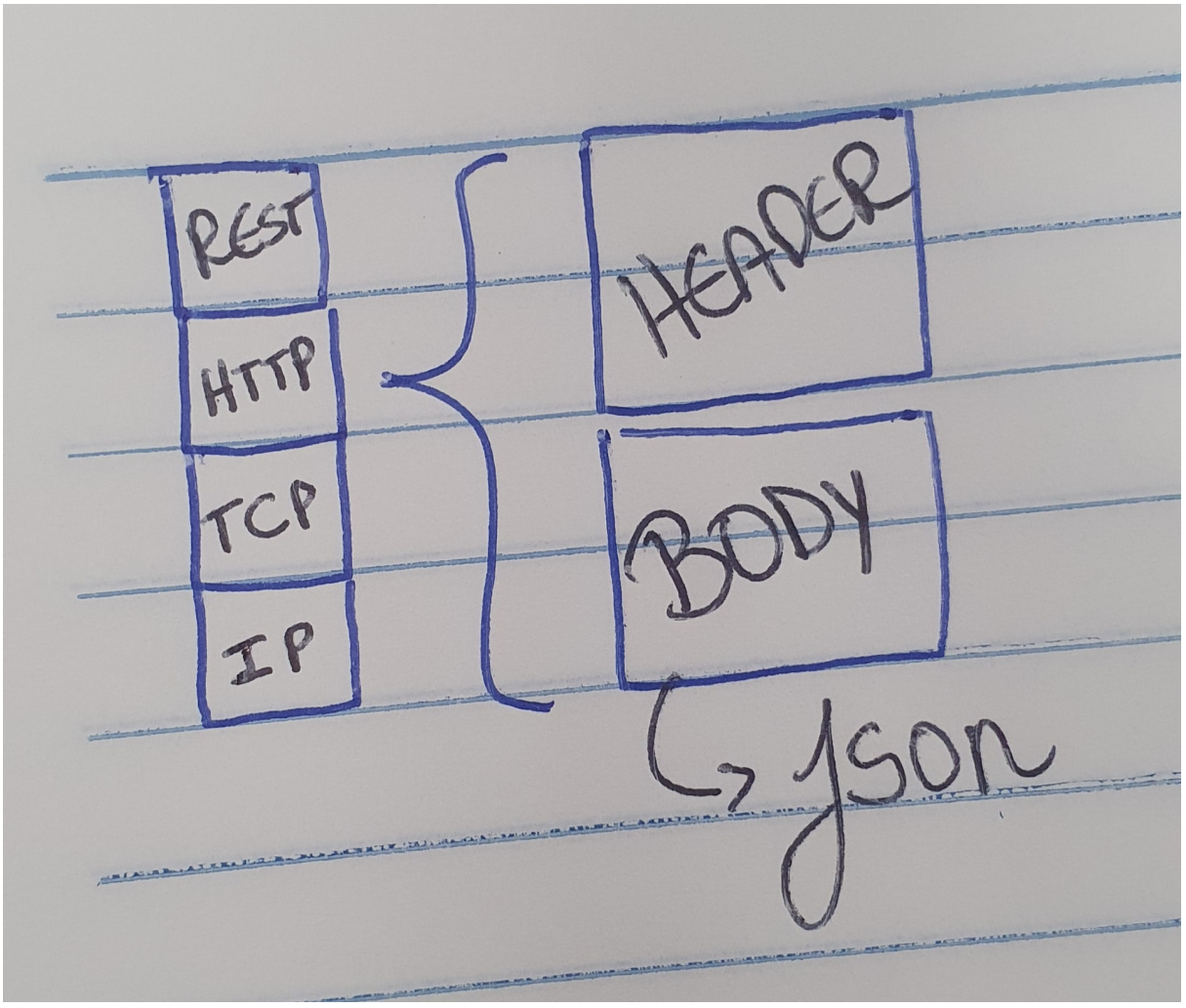
Protocolos

- O que é um Protocolo?

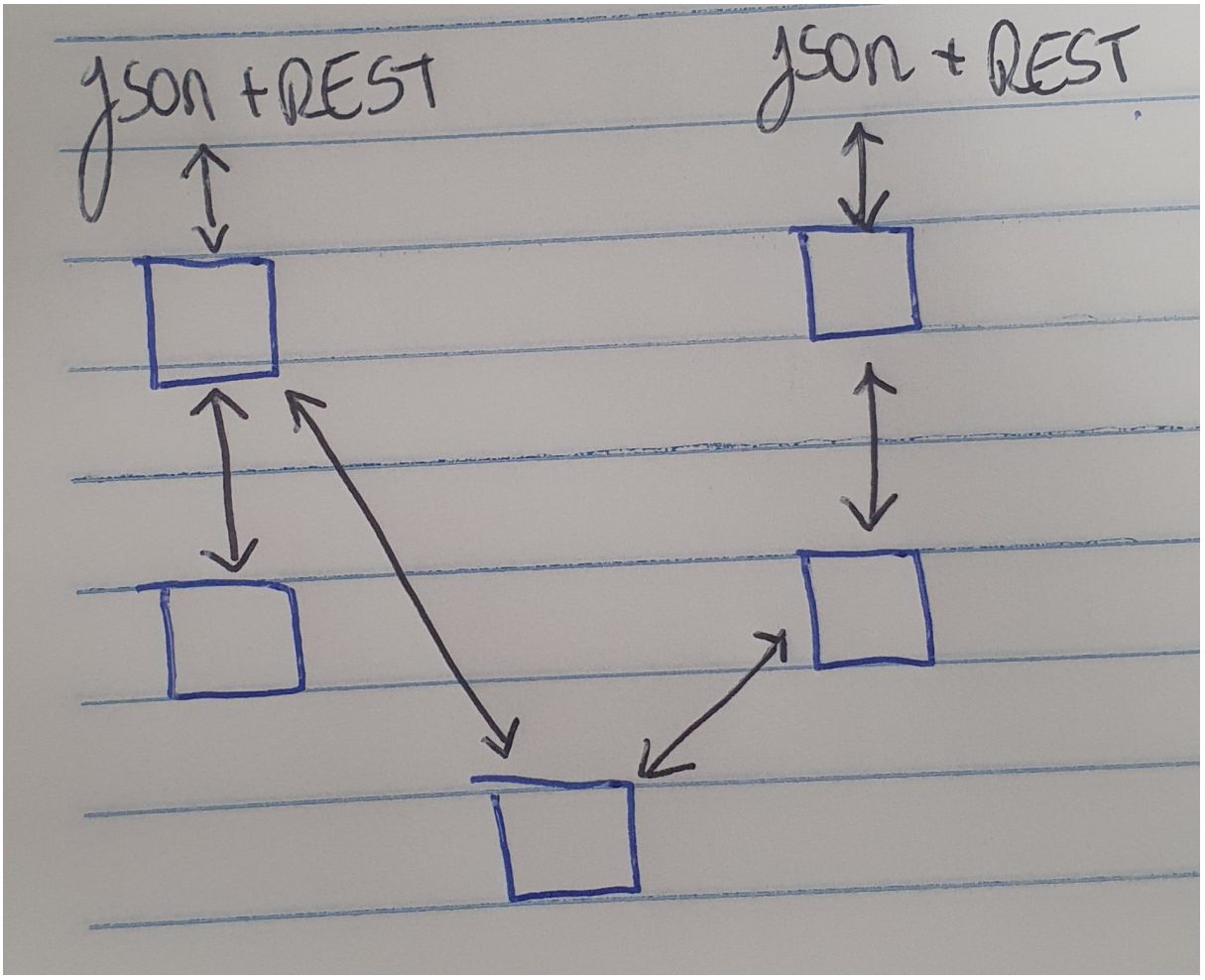
1. Regra de Comunicação

Regra de Comunicação é como eu vou construir uma mensagem, empacota-la, enviar essa mensagem e do outro lado, desempacotar.

Protocolo HTTP é composto por Header e Body (e uma linha em branco pra separar os dois).



(Composição HTTP e REST)



(Estrutura de comunicação de sistemas)

Os sistemas de baixo possuem a mesma regra de comunicação, uma das opções de conversas entre eles é o Cliente-Servidor

Sistemas Computacionais Diferentes

- HTTP → Wi-Fi
- Bluetooth

Sistemas computacionais diferentes possuem regras de comunicações diferentes, como elas conversam entre si? Através de conversores de dados.

Solução Cliente-Servidor

Em um Cliente-Servidor ligado ao REST, é necessário programar a comunicação entre ambos.

Requisição e Resposta

resp = f(req)

A resposta é o resultado de uma função f tal que a minha solicitação é o parâmetro requisição.

Em computação - **CHAMADA REMOTA**

Servidor s = Servidor.req(característica)

resp = s.send(informação)

Outro exemplo:

CLIENTE

resp = soma(A, B)

SERVIDOR

soma(x,y) {

  return x + y

}

Antes:

CLIENTE —*REQ*→ SERVIDOR

SERVIDOR —*RESP*→ CLIENTE

Depois:

CLIENTE —REQ→ MIDDLEWARE → SERVIDOR

SERVIDOR —RESP→ MIDDLEWARE → CLIENTE

Middleware que faz as conversões de um lado para o outro.