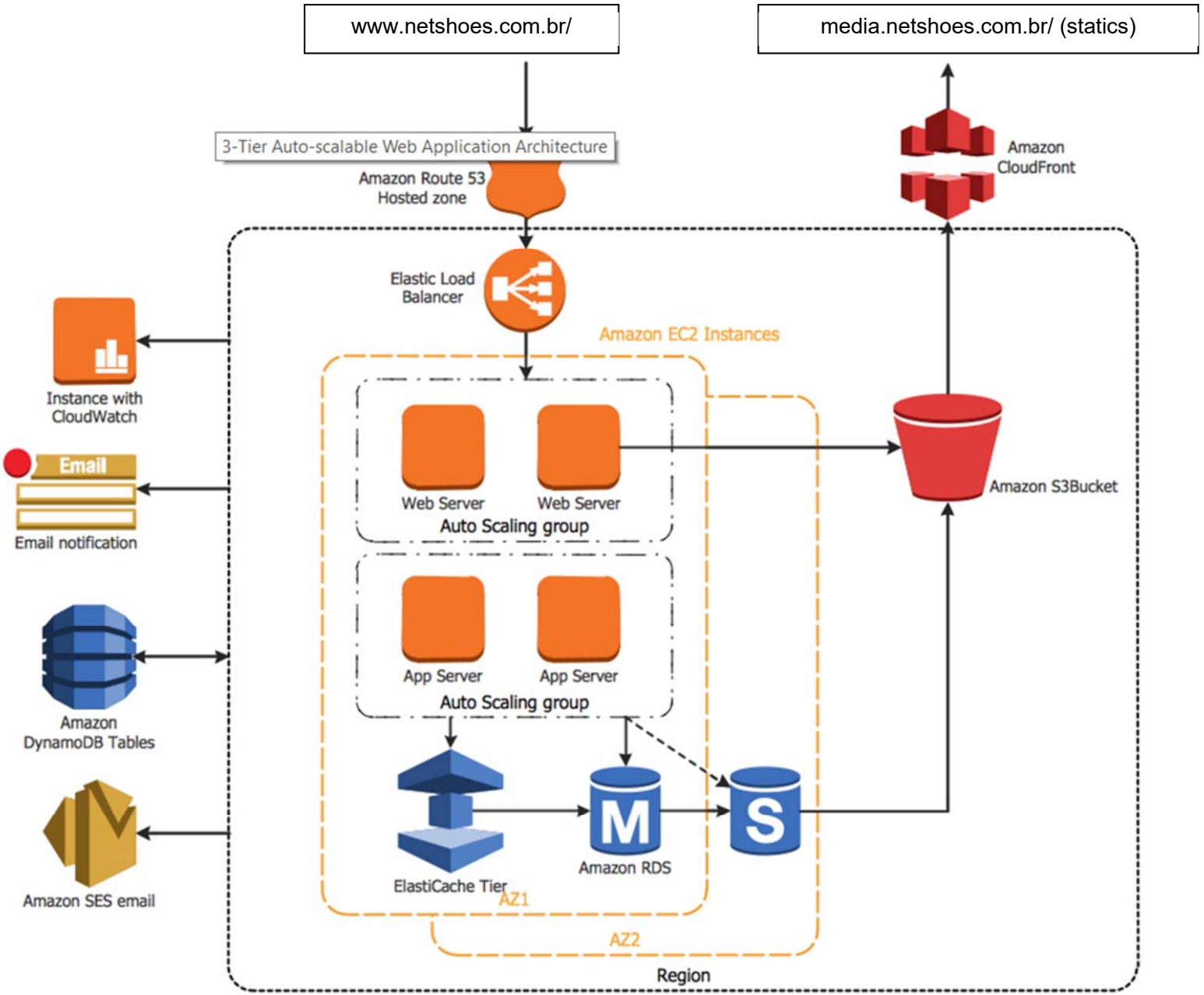
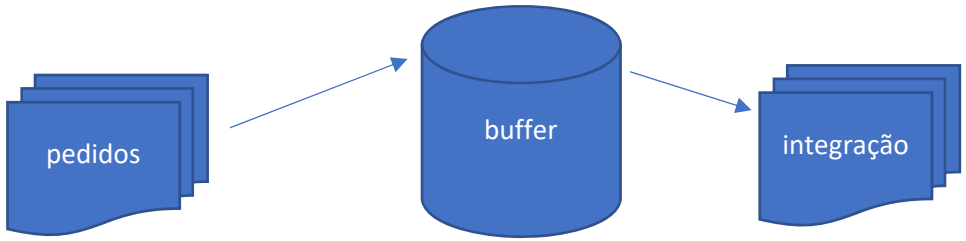
1)


www.netshoes.com.br/
media.netshoes.com.br/ (statics)
3-Tier Auto-scalable Web Application Architecture
Amazon Route 53
Hosted zone
Amazon
CloudFront
Elastic Load
Balancer
Amazon EC2 Instances
Instance with
CloudWatch
Email
Email notification
Web Server
Web Server
Auto Scaling group
Amazon S3Bucket
App Server
App Server
Auto Scaling group
Amazon
DynamoDB Tables
Amazon SES email
ElastiCache Tier
M
S
Amazon RDS
AZ1
AZ2
Region

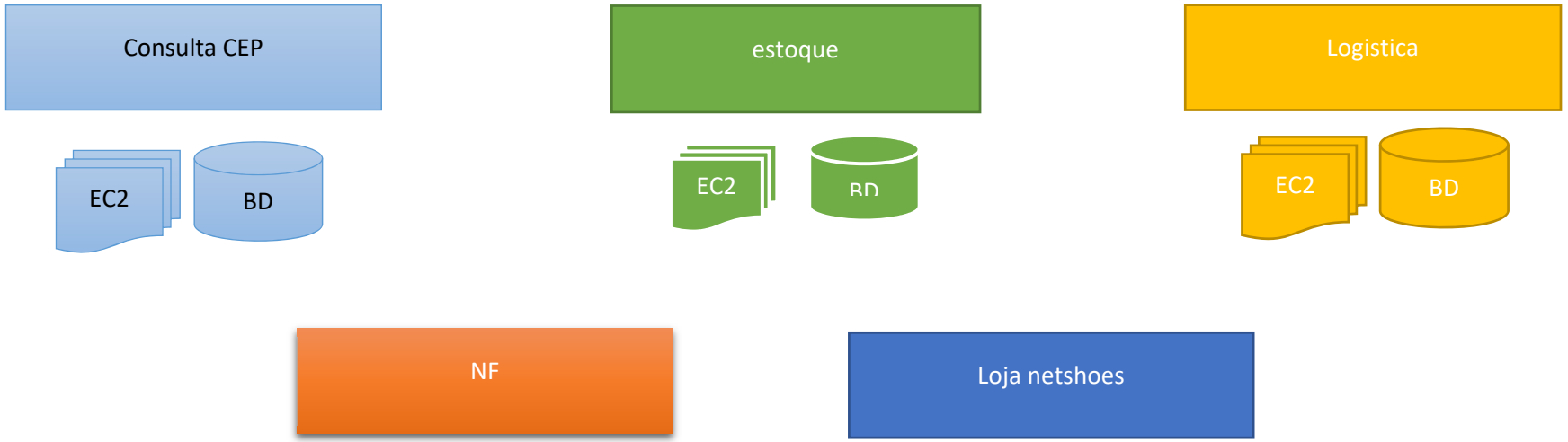
2) Dado a meta de 40k, sendo que podemos processar no máximo 5000 pedidos por hora, devemos implementar uma politica de BUFFER, onde antes de cada processamento, sempre comparar o número máximo da integração X número do lote pedido, caso seja inferior, verificar se existe alguma lote parado no backlog do buffer e se houver alocar junto ao lote a ser enviado.



3) Microserviço

3.1) O que é microserviço?

R: Microserviço são pequenos módulos dos mesmo négocio. Exemplo:



3.2) Porque devemos Utilizar?

R: Porque os serviços são autônomos e independentes, isto significa que podemos fazer um deploy, sem impactar em outros serviços, desta forma teremos 99,99999 de estabilidade.

3.3) Quais São os Prós e Contras.

R: Basicamente sistema monolítico, corremos o risco de realizar um deploy e parar a aplicação, diferente do microserviço. Em contra partida temos códigos repetidos, porem separadas em microserviços. Desta forma precisamos entender o negócio para verificar o melhor cenário.

4 Latencia de Rede.

4.1) Latencia de rede, é sinônimo de atraso, em outras palavras, é o tempo que o pacote de dados leva para ir de um ponto a outro.

4.2) Podemos diminuir a latência de rede utilizando alguns serviços de CDN, tais como, Cloud Front, Cloud Flare e etc. O mais importante é definir o publico alvo e a partir deste deixar Sua ZONA de disponibilidade conforme a região de acesso.

5)

Aproveitando o cenário 3.1 - microserviços – Estoque, podemos implementar uma estratégia assíncrona, que toda vez que for baixado um item no estoque, avisar via HTTP, XML, ou qualquer tipo de integração, o parceiro. O mesmo acontece para estorno, cancelamento etc.. E para ser escalável, basta utilizar o conceito de microserviços, com autoscalling.

7)

A função do servidor web é receber uma solicitação (requisição) e devolver (resposta) algo para o cliente.

O browser permite ao usuário solicitar um recurso e quando o servidor responde a uma solicitação são encontrados recursos como: páginas HTML, figuras e documento PDF que são exibidas depois para o usuário. Geralmente os servidores enviam instruções para o browser escritas em HTML. O HTML diz ao browser como apresentar conteúdo ao usuário web.

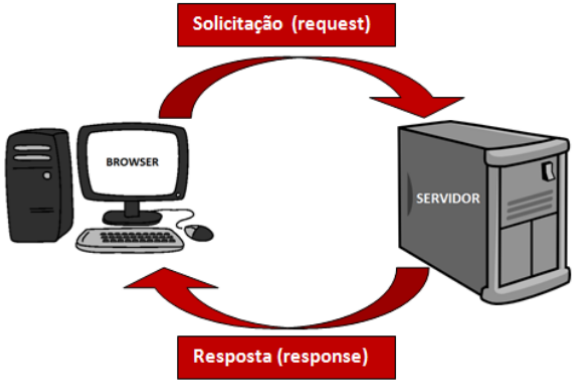


Figura 1: Simulação de uma solicitação e resposta

O servidor em si tem alguns recursos, mas por algumas deficiências não consegue processar tudo sozinho como: criações de páginas dinâmicas e o armazenamento de dados em um banco de dados.

Páginas Dinâmicas – Quando a aplicação roda no servidor, este disponibiliza somente páginas estáticas. Porém, para efetuar essa comunicação é necessário o auxílio de uma outra aplicação de ajuda que é passada através de Servlet.

Armazenar dados no servidor – Para efetuar essa ação o servidor precisa de uma aplicação de apoio (Servlet), fazendo com que o servidor envie esses parâmetros para o Servlet.

Solicitação do Usuário Web

A função do usuário web é permitir fazer solicitações ao servidor, exibindo o resultado do pedido. O browser é o software que se comunica com o servidor.

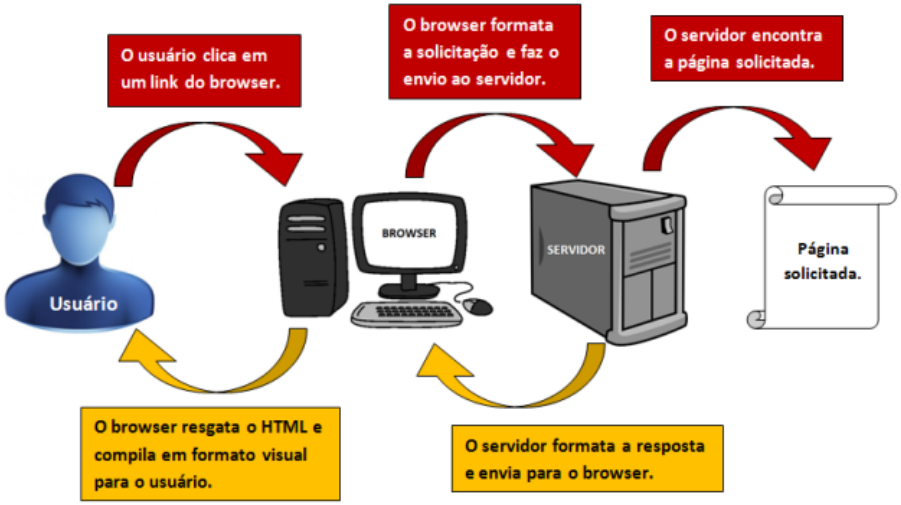


Figura 2: Usuário Web - Solicitação e resposta para uma página web

Protocolo HTTP

É um protocolo que os clientes e os servidores usam para se comunicar. Essa comunicação é baseada em requisições (request) e respostas (responses). Veja abaixo os elementos dessa comunicação:

Conteúdo de uma solicitação

- Método HTTP;
- Página que será acessada;
- Parâmetros do formulário;

Conteúdo de uma resposta

- Código de status (informa se a solicitação foi realizada com sucesso ou não);
- Tipo de Conteúdo (HTML, figuras, textos, etc);
- Conteúdo (HTML real, imagem, etc);

A solicitação HTTP possui outra solicitação conhecida como **URL** (Localizador Uniforme de Recursos). A solicitação URL é um recurso que se ativa quando o usuário tenta acessar alguns dos métodos HTTP descritos abaixo.

Métodos HTTP

GET - Solicita ao servidor um recurso chamado de solicitação URl. Este é o método padrão, pois é a forma como o browser chama o servidor quando digita-se uma URL para que ele a recupere.

POST - Contém um corpo nos quais seus parâmetros de solicitação já são codificados. O mais frequente uso desse método é na submissão de formulários.

HEAD - Similar ao método GET, o servidor apenas retoma a linha de resposta e os cabeçalhos de resposta.

PUT - Esse método permite o envio de arquivos par ao servidor Web.

DELETE - Permite a exclusão de documentos dentro do servidor Web.

OPTIONS - É possível fazer uma consulta de quais comandos estão disponíveis para um determinado usuário.

TRACE - Permite depurar as requisições, devolvendo o cabeçalho de um documento.

Tipos de protocolos

FTP - Sigla para File Transfer Protocol, é muito utilizado para transmissão (upload e download) de arquivos para um servidor.

SMTP - Sigla para Simple Message Transfer Protocol, fornece os comandos necessários para envio de mensagens a um servidor de e-mail.

POP - Sigla para Post Office Protocol, permite que um cliente acesse e manipule mensagens de correio eletrônico disponíveis em um servidor.

IMAP - Sigla para Internet Message Access Protocol, permite que um cliente acesse e manipule mensagens de correio eletrônico disponíveis em um servidor, assim como ocorre no protocolo POP.

Resposta de um HTTP

Uma resposta HTTP é composta de dois itens: o header e corpo. A informação do header faz três verificações que são:

- Protocolo que está sendo usado no browser;
- A realização de uma solicitação se ocorreu tudo certo;
- O tipo de conteúdo que está incluído no corpo. Apenas lembrando que o corpo possui o conteúdo que o browser exibirá.

Existem algumas respostas que o servidor encaminha para o browser. Muitas dessas respostas são comuns de serem vistas por programadores, sendo representadas por um número indicado pelo qual o problema foi ocorrido. Abaixo está listados os mais comuns:

200 (OK) – Informa que a confirmação da requisição foi respondida com sucesso.

304 (NOT MODIFIED) – Informa que os recursos que não foram modificados desde a última vez que foi feito um pedido. Isso ocorre por causa dos mecanismos de cache do browser.

401 (UNAUTHORIZED) – Informa que o cliente não tem acesso autorizado para acessar a área requisitada. Ocorre muito em intranets de acesso privado que precisam ser acessadas com um usuário e senha.

403 (FORBIDDEN) – Informa que o acesso à área requisitada falhou. Isso pode ocorrer em caso de acesso a áreas que exigem login e senha e não houve autorização para aquele usuário.

404 (NOT FOUND) - Não encontrado. Ocorre quando o usuário tenta acessar uma área inexistente no endereço passado, por exemplo, páginas removidas ou recursos excluídos.

Entendo o funcionamento das respostas

Abaixo na Figura 3 são apresentadas algumas descrições da anatomia da resposta em HTTP.

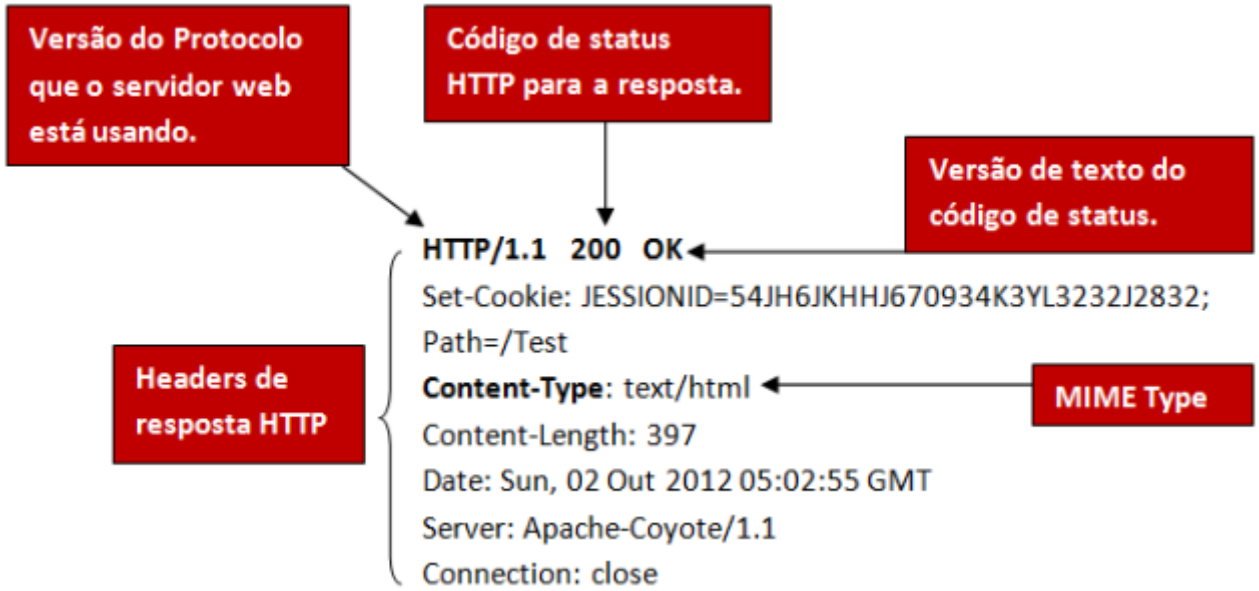


Figura 3: Anatomia de uma resposta HTTP

O **MIME Type** é o valor do header de resposta para o content type. O MIME type informa ao browser que tipo de dado está pronto para receber, pois se precisa saber como será processado. No caso da Figura 3, estamos informando que o conteúdo é do tipo HTML, mas poderíamos estar esperando um tipo PDF, então seria mudado o Content-Type.

Conclusão

O HTTP usa um modelo de solicitações e respostas. Uma solicitação ocorre quando o usuário faz uma solicitação HTTP e o servidor web devolve uma resposta HTTP, sendo que o browser verifica como tratar esse conteúdo. Se a resposta que vem do servidor for uma página HTML, então é inserido na resposta HTTP.

As diferenças entre as solicitações GET e POST são que enquanto o GET anexa dados do formulário no final da URL o POST inclui dados do formulário no corpo da solicitação.