01

Modelo Requisição e Resposta

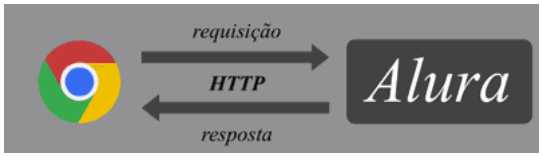
Transcrição

Já descobrimos que o HTTP é um protocolo que define as regras de comunicação entre cliente e servidor e de que as URLs são constituídas. Porém isso não é tudo, vejamos mais alguns detalhes sobre o funcionameto da Web e do HTTP.

Realizaremos um teste **efetuando login** na Alura. Quando preenchemos o formulário e clicamos no botão, o navegador envia o nosso *login* e a nossa *senha* para o servidor através do protocolo HTTP! Vamos detalhar um pouco esta ação.

No mundo HTTP, a requisição enviada pelo navegador para o servidor é chamada de **HTTP REQUEST**. Recebemos a página */dashboard* como resposta já que enviamos login e senha válidos. No mundo HTTP essa resposta é chamada de **HTTP RESPONSE**.

A comunicação segue sempre esse modelo: o cliente envia uma requisição e o servidor responde. **Requisição e Resposta** ou em inglês: ***Request-Response***. Aqui é importante saber que a comunicação sempre começa com o cliente: é ele quem pede as informações. O servidor responde apenas o que foi requisitado e nunca inicia a comunicação!



Comunicação sem estado

Vamos acessar rapidamente outro site: `http://g1.globo.com` . Para este acesso estamos enviando uma requisição para *g1* e recebemos como resposta a página inicial.

Agora vamos navegar dentro do site e acessar algum artigo. Ao clicarmos enviamos uma nova requisição e percebemos que TODA página foi trocada. Fica mais claro ainda se acessarmos do menu acima algum link (*globo esporte* ou *globo show*). Podemos ver que todo o conteúdo do site foi trocado.

Isso também acontece no caso da Alura (talvez um pouco mais difícil de perceber). Ao acessarmos recursos diferentes todo o conteúdo no navegador foi trocado (apesar do menu parecer o mesmo, ele também foi trocado). A ideia do HTTP é justamente essa, cada recurso é independente do outro e não depende do anterior. Isso também se aplica para os dados enviados na requisição. Cada requisição é independente da outra e ela sempre deve conter todas informações para o servidor responder.

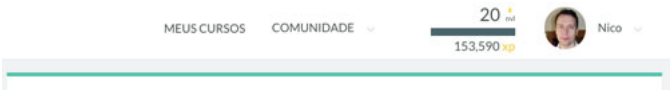
Pense que HTTP funciona como o envio de cartas pelo correio e uma carta representa uma requisição. Você fez uma viagem e gostaria de enviar 3 cartas para sua mãe. Adianta falar para os correios "eu vou colocar o endereço apenas na primeira carta, ai vocês já sabem para onde enviar a segunda e terceira carta"? Não adianta pois os correios tratam cada carta independentemente, e assim também funciona o HTTP. Cada requisição (carta) precisa ter todas as informações. A mesma coisa se aplica para a resposta, precisa ter todas as informações.

Essa característica de cada requisição ser independente é chamada de **stateless**. É esse nome bonito mesmo! O HTTP é um *protocolo que não mantém o estado de requisições*. Isso significa que só com HTTP não há como se lembrar das requisições anteriores enviadas para o servidor. Por isso precisamos incluir em cada requisição todas as informações, sempre. Para o desenvolvedor este conhecimento é importante pois é justamente essa característica stateless que o

atrapalha no dia a dia. Ele precisa preparar a aplicação web para que funcione bem usando o protocolo HTTP, algo que veremos nos treinamentos da Alura.

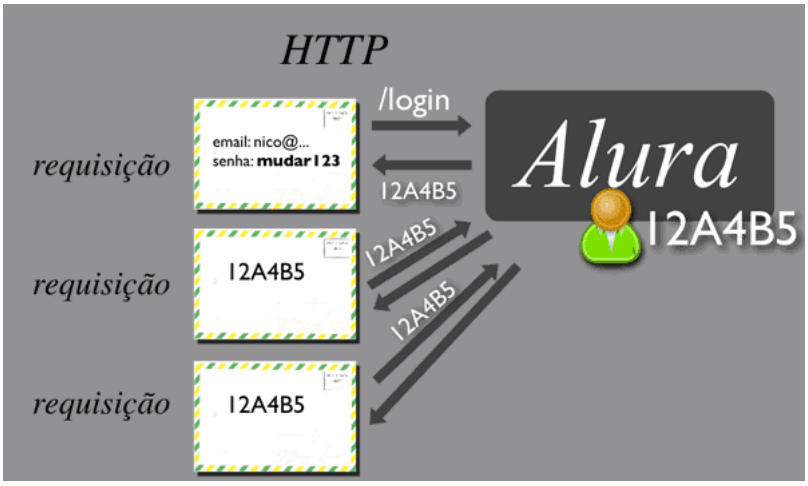
Lidando com Sessões

Reparem que, mesmo após termos realizado o login e termos enviado várias requisições, aparece o ícone com a minha imagem no menu principal.



Ou seja, a Alura se lembra de alguma forma que eu fiz login em alguma requisição anterior. Como falamos antes, cada requisição deve enviar todas as informações para gerar a resposta. Isso significa que o navegador envia em cada requisição informações sobre o meu usuário! Se cada requisição for independente uma da outra, e não tiver como se lembrar das requisições anteriores, não tem outra explicação a não ser que o navegador envie os dados sobre o meu usuário em cada requisição! Lembre-se da carta postal, ela sempre precisa ter os dados do remetente e aqui não é diferente!

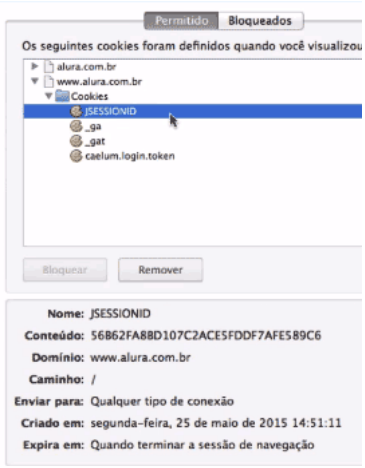
Então o navegador *envia o login e senha em cada requisição*? Não, não seria muito elegante nem seguro fazer isso. Mas ele faz algo parecido, acreditem ou não. Quando efetuamos o login, a Alura valida os nossos dados, certo? Nesse momento, o servidor tem certeza que o usuário existe e gera uma identificação quase aleatória para o usuário. Essa identificação é um número criado ao vivo e muito difícil de adivinhar. Esse numero é a identificação temporária do usuário e ele será devolvido na resposta.



Conhecendo cookies

Então onde está esse número? O navegador grava esse número em um arquivo especial para cada site, são os famosos **cookies**. Como acessar esse cookie depende um pouco do navegador em uso. O mais importante é entender o porquê da existência desse número e onde ele foi gravado.

No Chrome podemos ver todos os cookies armazenados nas *Configurações -> Privacidade -> Configurações de conteúdo... -> Todos os cookies e dados de site...*. Se procurarmos por **Alura**, em **cursos.alura.com.br**, lá temos um cookie com o nome `caelum.login.token` , que contém o número da identificação. Se apagarmos esse cookie, perderemos nossa identificação, sendo assim, a Alura exigirá um novo login pois não saberá que já tínhamos logado.



Normalmente o nome do cookie é algo como `session-id`, dependendo da plataforma de desenvolvimento utilizada ele pode se chamar de `PHPSESSID` ou `ASP.NET_SessionId` ou `JSESSIONID` ou outro nome que foi inventado! O Cookie será gerado de forma transparente pela tecnologia que você for utilizar para criar aplicativos web. Esse nome, `PHPSESSIONID`, `JSESSIONID` ou outro, é gerado pela ferramenta de gerenciamento de Sessão. Por isso ela muda o nome. Se você está usando PHP, então o PHP gerará o nome do Cookie e seu identificador (número aleatório) e chamará o cookie `PHPSESSIONID`. No Java já será usado o nome `JSESSIONID`.

Resumindo, todas as plataformas ajudam a gerar esse número e a criar o cookie de maneira transparente. É dessa forma que as plataformas gerenciam as **SESSÕES** com o usuário. Como isso funciona de modo concreto você aprenderá nos cursos e carreiras específicas.

A ideia de manter dados entre requisições é algo muito comum no desenvolvimento de aplicações na web. Um usuário que se loga no sistema web causa a criação de uma sessão. Uma sessão então é útil para guardar informações sobre o usuário e ações dele. Um exemplo clássico é um carrinho de compras. Entre várias requisições estamos usando o mesmo carrinho de compras que guarda os nossos produtos escolhidos (fizemos uma sessão de compras online).

Resumindo teremos:

- O HTTP usa sessões para salvar informações do usuário
- Sessões só são possíveis por uso de Cookies
- Cookies são pequenos arquivos que guardam informações no navegador
- O HTTP é stateless, não mantem estado.