Aula 19 - Cookies, Sessões e Armazenamento II

# Armazenar

**sessões**

Uma sessão é um link que é gerado quando o cliente se conecta a um servidor, esse link é representado por um sessionId, que é salvo no navegador como o identificador da sessão. Porém... Onde a sessão é salva no servidor?

**Armazenamento de memória**

O armazenamento de uma sessão por armazenamento em memória é exatamente o mesmo que a persistência em memória que trabalhamos na última aula. Este é um armazenamento muito perigoso, porque se o servidor morrer ou for reinicializado, a sessão morrerá com ele e não haverá como recuperá-lo.

****

**Como corrigir o problema de armazenamento de memória?**

Uma vez entendida a natureza arriscada de armazenar uma sessão na memória, surge a pergunta: como poderíamos fazer com que essas sessões persistissem para evitar qualquer problema na reinicialização repentina ou queda de um servidor?

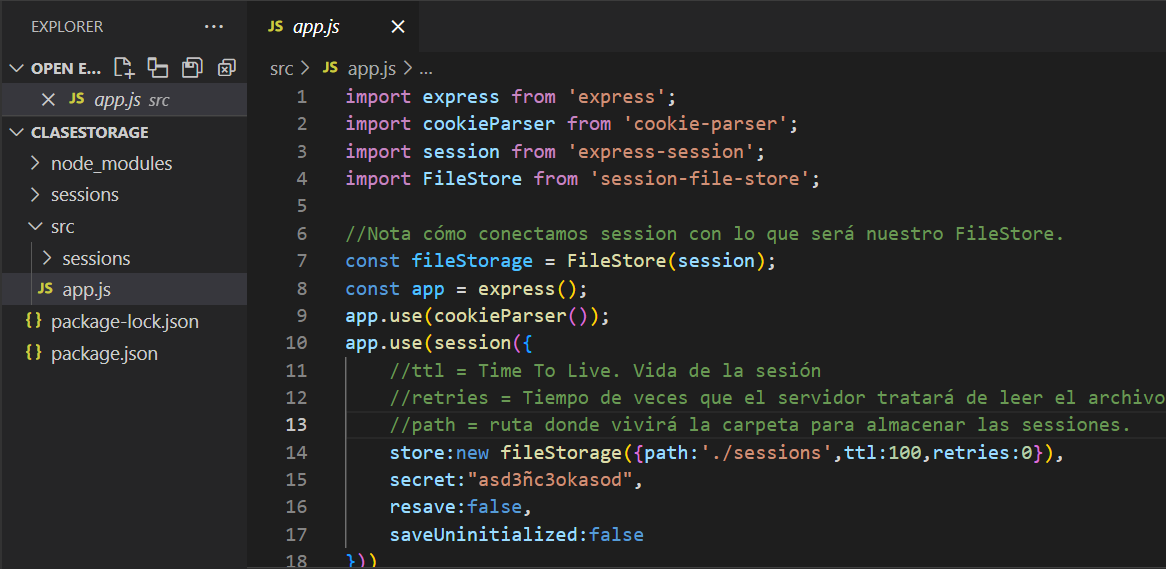
Certamente você tem uma ideia clara, já que fez isso com seus produtos e seu carrinho: vamos usar o File Storage, ou seja, salvar nossa sessão em um arquivo, para poder consultar as sessões de um lado diferente do servidor memória.

**Armazenamento de arquivo**

Como nas primeiras aulas do curso, o armazenamento de arquivos nos permitirá dar uma persistência de arquivos às sessões em que trabalhamos, isso permitirá que o servidor os leve de algum lugar caso algo inesperado aconteça (uma reinicialização ou uma falha) .

Assim, os usuários podem continuar fazendo consultas com seu id de sessão, e o servidor poderá consultá-los a partir do arquivo que persiste nele. É um recurso simples e tem seus defeitos, porém é uma solução simples para resolver o problema da sessão, sem precisar usar recursos de terceiros **.**

**Usando o armazenamento de arquivos em nosso servidor.**

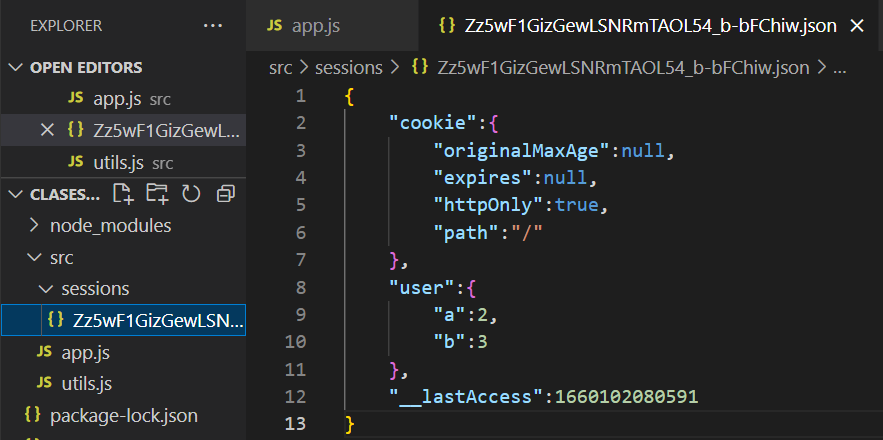
****

Começaremos com a instalação usual com npm com o comando:

Posteriormente, inicializamos o fileStore conforme indicado na imagem, isso utilizando os três argumentos principais:

caminho, ttl, retirar.

**Ao final, será gerado um arquivo com as informações indicadas**

****

**Considerar**

Usando o File Storage podemos perceber que:

* Mesmo se reiniciarmos o servidor, a sessão persistirá pelo tempo indicado
* Quando uma sessão expirar, o servidor irá gerar um novo arquivo com a consulta, indicando o novo sessionId.
* Arquivos antigos não são limpos automaticamente, eles são deixados em um cemitério de registro, o que pode levar a complicações mais tarde.

**Armazenamento de sessão com Mongo Atlas**

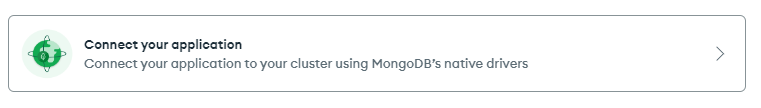
**A próxima etapa para salvar sessões**

Com certeza você já tinha pensado nisso! Se o FileSystem acabar causando conflitos, poderia ser corrigido com bancos de dados, como fizemos com nossos carrinhos e produtos? A resposta é: sim, e de que forma!

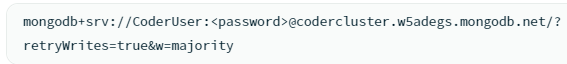
A sessão pode trabalhar lado a lado com o MongoDB e o MongoAtlas para poder salvar uma sessão em um banco de dados, isso permitirá que as sessões tenham um gerenciamento mais limpo, além de poder excluir automaticamente as sessões expiradas.

**A primeira coisa será ter nossa string de conexão**

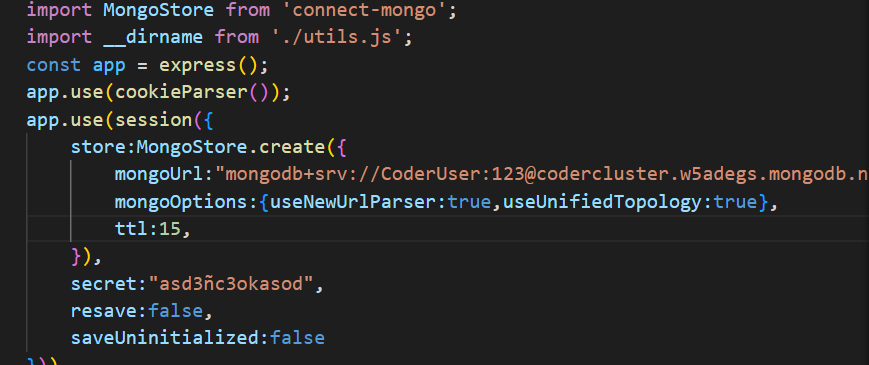
Lembramos que, no Mongo Atlas, dentro do cluster que havíamos levantado anteriormente, temos um botão chamado



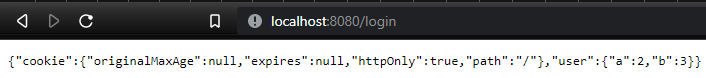
O mesmo que nos levará para a string de conexão que usaremos em nosso servidor:



**Nossa configuração de sessão exigirá um novo módulo** :



**A solicitação de login torna-se normal:**

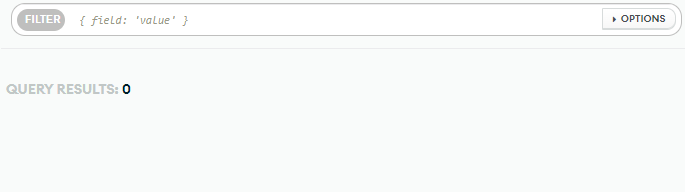
****

E a sessão será armazenada no banco de dados

****

**A maravilha de expira**

O objeto de sessão armazenado no banco de dados tem um “expira”. Este é baseado no ttl definido e se encarregará de se limpar automaticamente assim que o tempo de expiração passar. Esse

****