01

Nossa webapp offline com Application Cache

Transcrição

Nossa app está pronta, já conseguimos todas as funcionalidades que queríamos nela. Se tiver uma ideia nesse instante, posso colocá-la na aplicação e salvá-la para leitura quando quiser. Para testar isto, vamos abrir um navegador e acessar a app, que tem um endereço: (http://localhost:3000)http://localhost:3000 (http://localhost:3000), em que "3000" remete ao fato de ser um servidor sendo rodado em minha máquina; poderia ser uma url com terminação ".com.br" também.

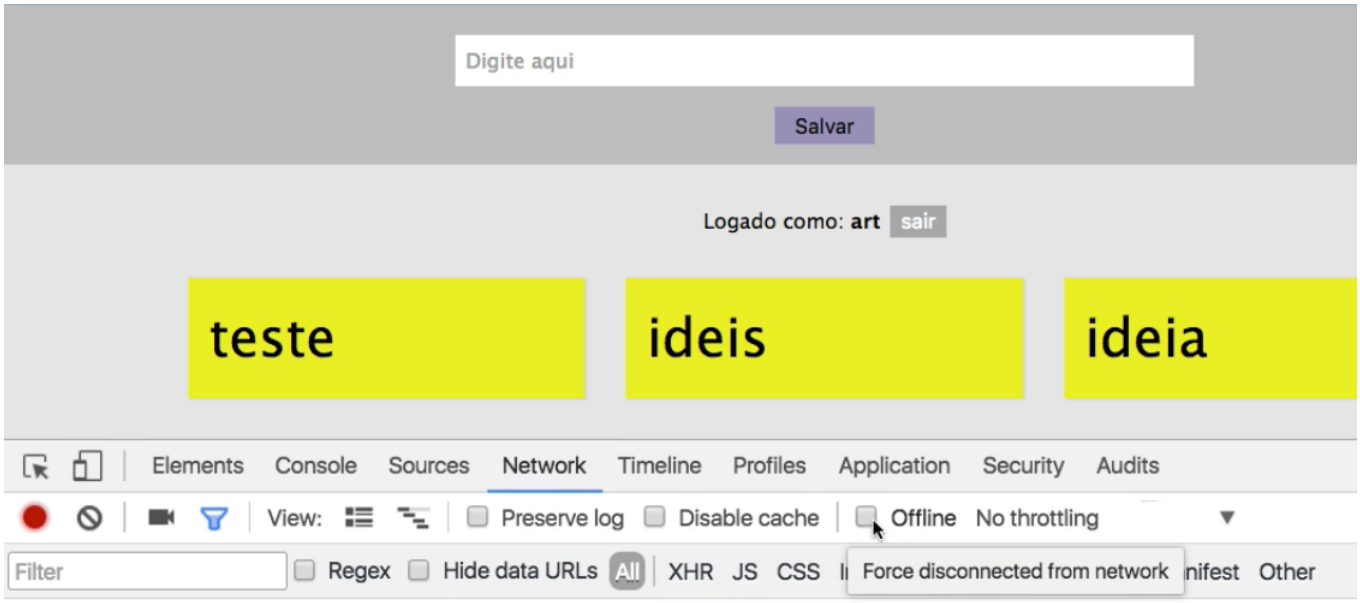
Acessei, ele está aqui. Se quiser, posso colocar uma ideia e salvá-la, pronto:

Também consigo apagá-la, trocar sua cor, ou seja, está funcionando. Porém, se quisermos continuar chamando nosso programa de "app", ou "aplicativo", existe algo muito importante que não fizemos ainda. Quando se falava em "aplicativo" até certo tempo atrás, pensávamos em uma aplicação que baixávamos em uma loja. Nesses aplicativos chamados de "nativos" dos celulares para as *Web Apps* há uma diferença considerável, ao menos por ora.

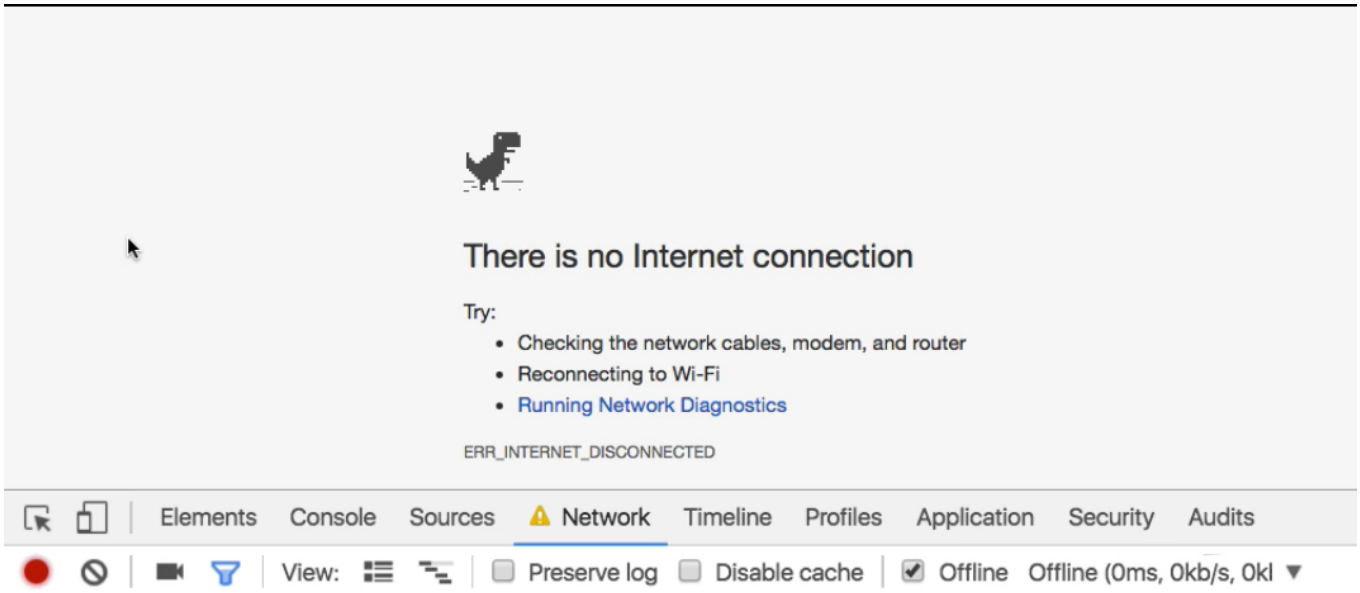
Tenho um aplicativo de notas, por exemplo, nossa Web App. Se estiver no metrô, indo para a faculdade ou para o trabalho, e quiser anotar uma ideia, terei que acessar um site, o localhost (http://localhost:3000), ou ceep.com.br (http://www.ceep.com.br) se tiver um domínio. E aí, quando estiver dentro do metrô, sem internet, e der enter, de onde as informações vão vir? Como ele vai acessar um servidor?

Nossa Web App ainda não funciona dessa maneira. Vamos tentar mostrar isto, ou seja, que ela não funciona sem internet. Como vou fingir que estou offline? Posso desativar minha wi-fi ("Turn Wi-Fi Off"), porém, esse servidor específico funciona sem wi-fi, aqui na minha máquina, pois trata-se de um servidor que se encontra em meu computador. Quando acesso o (http://localhost:3000)http://localhost:3000 (http://localhost:3000), não preciso acessar outro lugar, de forma que desativar o wi-fi não seja o suficiente aqui. O navegador sabe disso, as pessoas que fizeram os *develop tools* sabem disso.

Podemos clicar na aba "Network", em que selecionamos a opção "Offline", o qual permitirá que "fingimos estar desconectados":

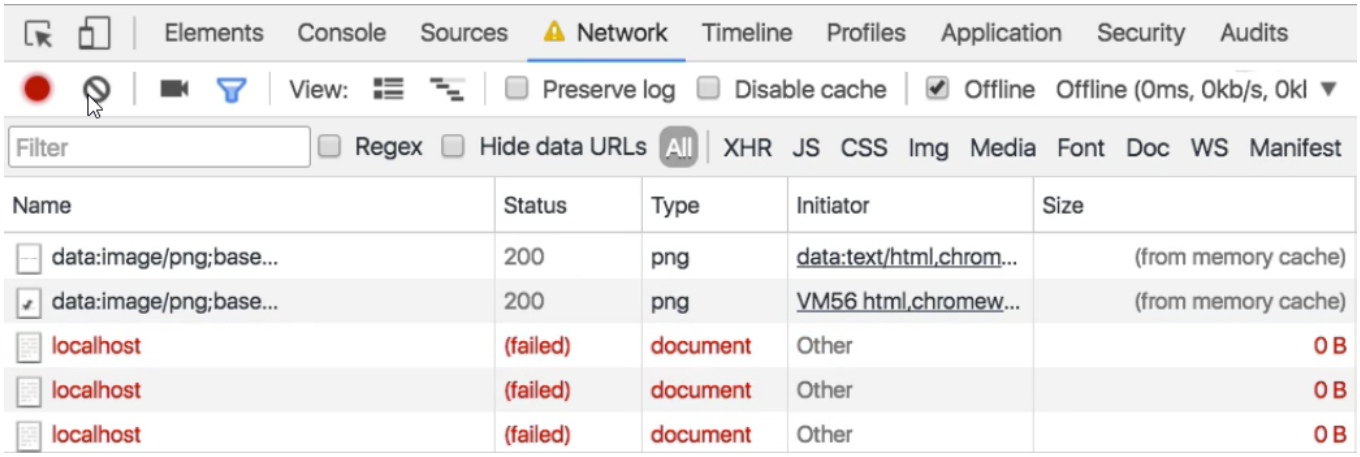


Recarregando a página, vê-se que estamos de fato offline:



Se estivesse dentro de um metrô, não conseguiria acessar nossa Web App. Por sua vez, uma **App nativa** baixada em uma loja de aplicativos estaria sendo utilizada de qualquer maneira, pois é isso que esperamos delas. Se quisermos dizer que nossa Web App é uma aplicação de verdade, é isso que falta: fazê-la funcionar até mesmo sem acesso à internet.

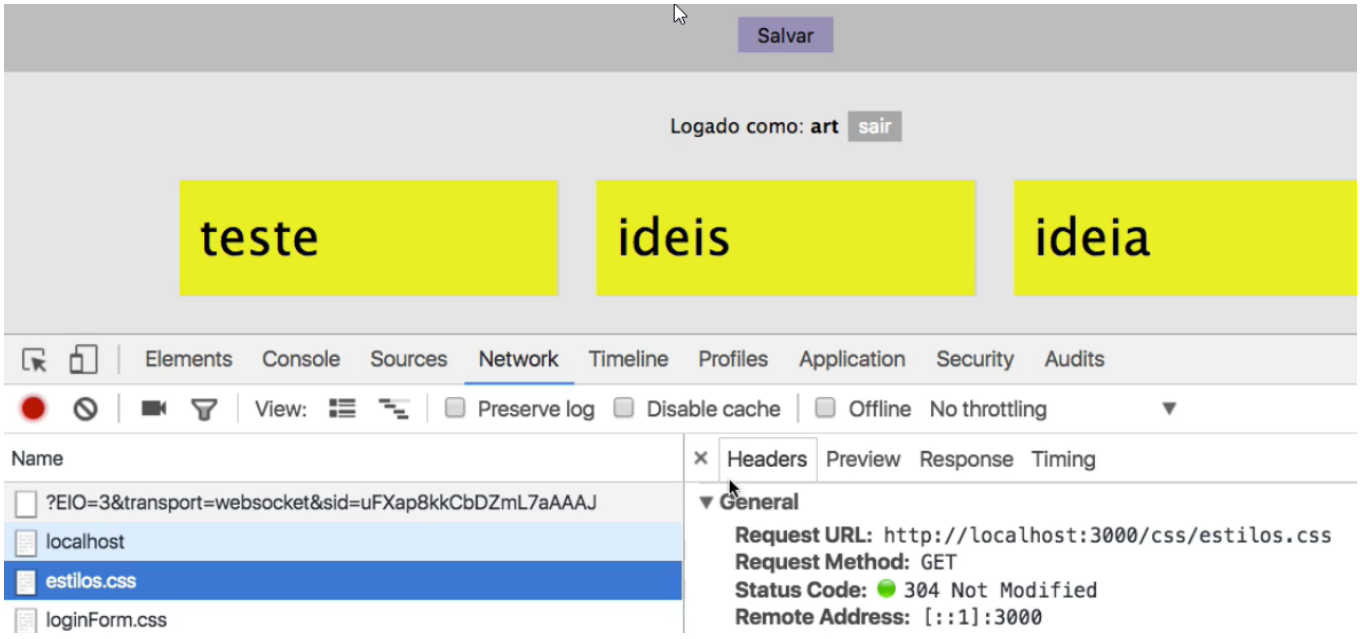
Na aba "Network" pode-se ver os erros: tentei carregar o localhost, mas falhei, está escrito aqui ao lado, na parte "Status":



Quero dizer, ele não conseguiu carregar o arquivo principal, que é o `index.html`. Isso ocorreu porque ele tentou acessar o servidor, sem rede. Se tornar a ficar online, a página é carregada normalmente, com todos os seus arquivos. Nesse exato instante, tenho acesso a eles, o que não acontece quando fico offline. Por quê ele não usa os arquivos `.js` carregados anteriormente? Na verdade, ele usa, só que de uma maneira diferente.

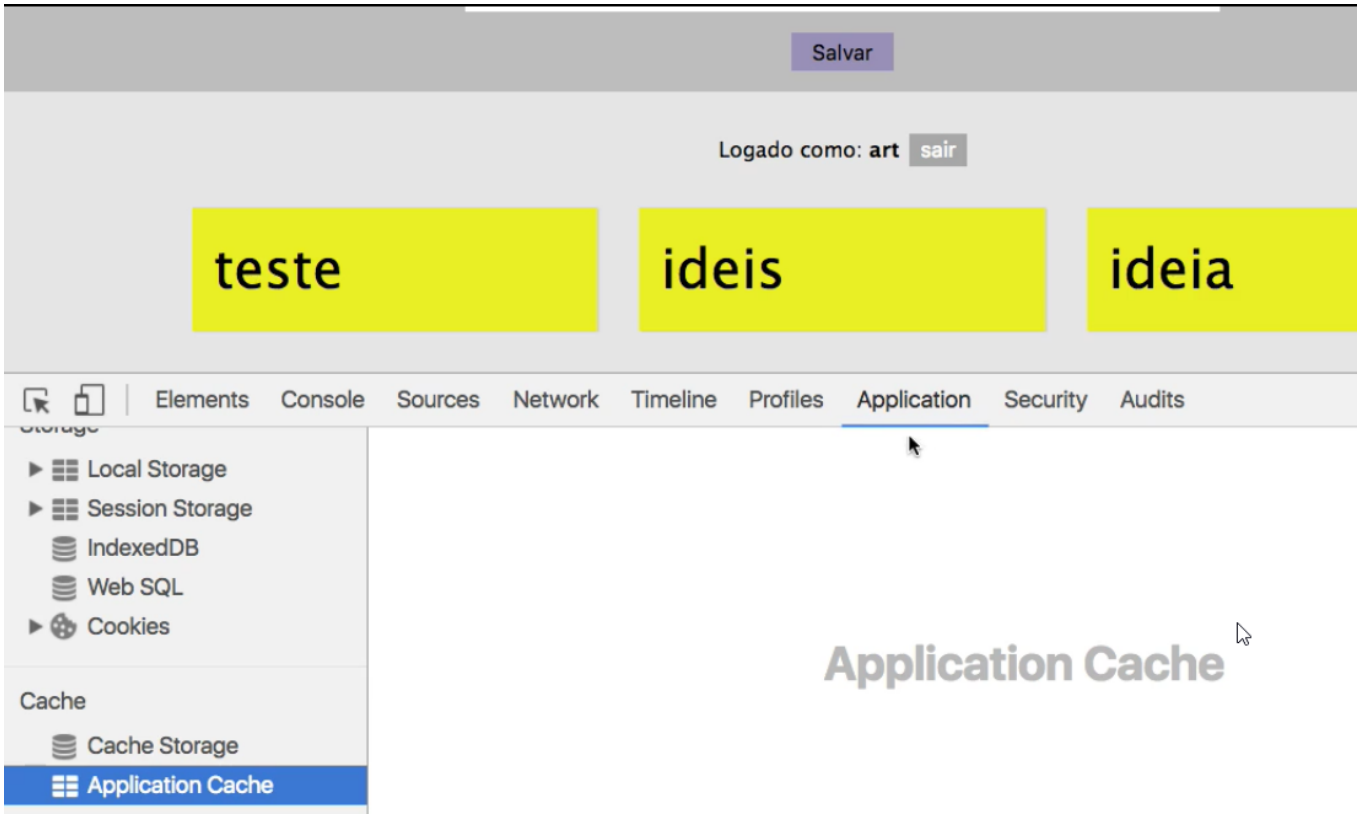
Ainda em "Network", os status ao lado de cada arquivo mostram que eles não foram modificados ("304 not modified" por exemplo). Isso significa que os arquivos que eu já tinha antes estão sendo utilizados. Quando estou online, ele usa os arquivos carregados anteriormente? Por quê?

A ideia é economizar a internet do usuário, então, o que se usa aqui é o **cache**. Nosso browser sabe, por exemplo, que `estilo.css` não foi alterado, como se vê em "Headers" do nosso HTTP.



Na resposta que o servidor nos dá, há uma ETag que serve como identificador do arquivo. A ETag permaneceu a mesma ("W/"124-1598337cf70") quando recarrego a página, ou seja, ele sabe que não precisa atualizar nosso `.css`, já que a ETag é inalterada. Esta ideia de carregar o arquivo somente quando ele é necessário já existe há algum tempo, e é o que chamamos de cache. Ele salva arquivos para economizar dados e carregá-los mais rapidamente. A ideia sempre foi essa: carregar os arquivos de modo mais rápido possível.

No momento, queremos utilizá-los sempre, estando offline ou online, e não simplesmente carregá-lo mais rapidamente. É o que faremos agora, vamos carregar os arquivos e salvá-los no que chamamos de **"Application Cache"**, o qual, ao acessarmos a aba "Application" do Developer Tools, vemos que se encontra vazio. Iremos preenchê-lo com todos os arquivos carregados.



Primeiramente, preciso informar ao Application Cache quais os arquivos a serem salvos, e faremos isso listando-os. Como? Vamos verificar o que ele tenta carregar estando offline? Inicialmente, ele tenta carregar o (http://localhost:3000)http://localhost:3000 (http://localhost:3000), ou seja, o `index.html`, nosso documento. Vemos que nossos arquivos já se encontram listados ali. A tag `link` é quem diz que precisa do estilo `.css`, e é a tag `script` que diz que precisa dos arquivos `.js`. Porém, isso não é o suficiente. Precisamos listar aqui o que queremos colocar no Application Cache.

Para isso, criarei um atributo `manifest`, arquivo onde listaremos o que queremos instalar. É o nome do arquivo que colocarei nesse atributo, o qual chamarei de `offline.manifest`:

```
<html lang ="pt-br" manifest="offline.manifest">
```

A denominação `offline.manifest` não é tão importante, o que importa é que seja o mesmo nome que aparece no arquivo a ser criado, em que listarei todos os arquivos de que preciso, e se encontram no `index.html`, como visto anteriormente. Para isso, precisamos copiar e colar manualmente todos os respectivos `href`.

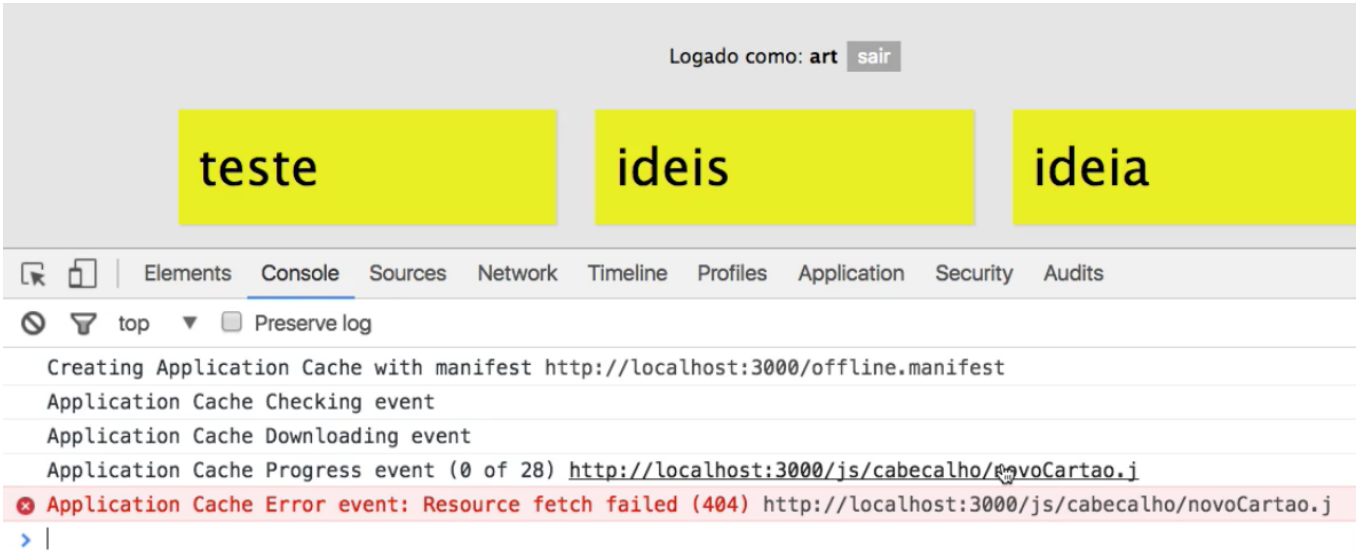
Estou usando o Visual Studio Code, mas dependendo do editor que vocês estiverem usando, conseguimos utilizar a múltipla seleção para isso. Somente os arquivos `.css` não são o suficiente, então tentarei fazer o mesmo com todos os `.js`.

```
CACHE MANIFEST

CACHE:
css/estilos.css
css/opcoesDaPagina.css
css/opcoesDoCartao.css
css/cabecalho.css
css/login.css
css/loginForm.css
css/loginStatus.css
```

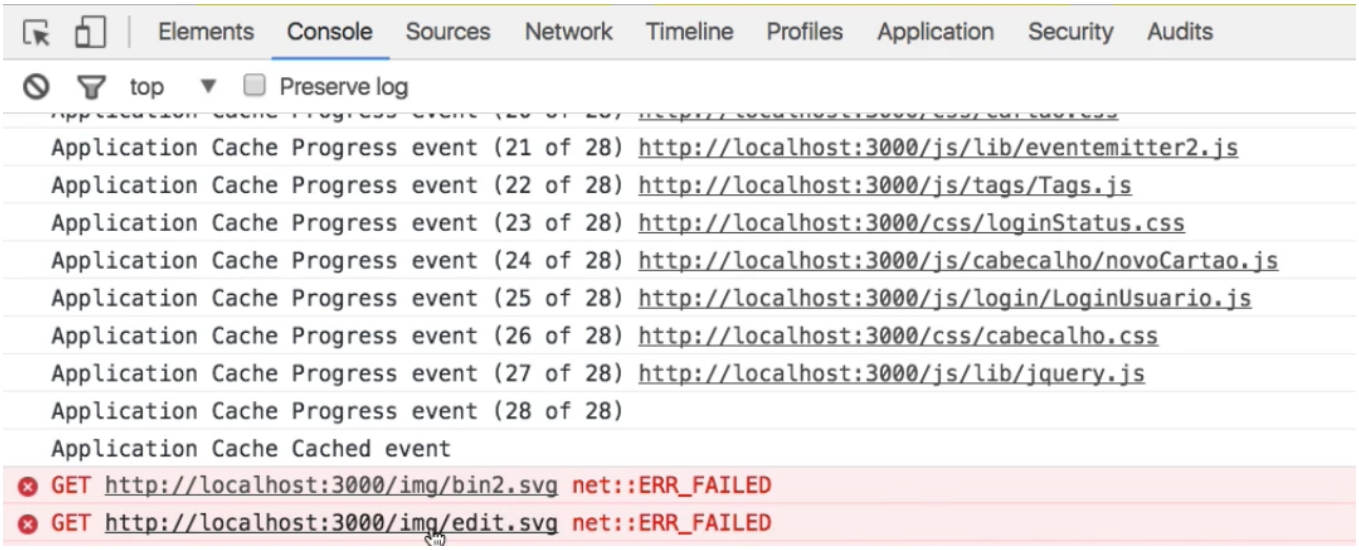
```
css/cartao.css
css/novoCartao.css
css/mural.css
js/lib/jquery.js
js/lib/eventemitter2.js
js/lib/KeyBoardNavigation.js
js/tags/Tags.js
js/cabecalho/mudaLayout.js
js/cabecalho/busca.js
js/filtro/Filtro.js
js/tipos/TiposCartao.js
js/cartao/render/Cartao_renderHelpers.js
js/cartao/render/CartaoOpcoes_render.js
js/cartao/render/CartaoConteudo_render.js
js/cartao/render/Cartao_render.js
js/cartao/Cartao.js
js/login/LoginUsuario_render.js
js/login/LoginUsuario.js
js/mural/render/Mural_render.js
js/mural/Mural.js
js/cabecalho/novoCartao.j
```

Arquivos copiados, quero dizer, tudo está funcionando, tenho todos os arquivos que quero instalar. Vamos testar no navegador! Vamos carregar nossa Web App pela primeira vez, ficando online e atualizando a página. Como vamos saber se funcionou ou não? Clicando em "Console", sempre que um `manifest` for encontrado, ele irá avisar, ou seja, uma parte funcionou. Os arquivos começaram a ser baixados, e quando ele foi tentar baixar o `novoCartao.js`, não conseguiu pois não foi possível encontrá-lo:



Nada foi instalado! Não há nada no Application Cache. Isto é muito parecido com o que ocorre quando instalamos apps hoje em dia. Qualquer erro, se a conexão com a internet cai, o download falha.

Voltando ao `offline.manifest`, vejo que ficou faltando uma letra na extensão do arquivo ("*js/cabecalho/novoCartao.j*"), e faço a correção ("*js/cabecalho/novoCartao.js*"). Recarrego a página e, dessa vez, ele lista 28 arquivos, dentre os quais dois aparecem com erros:



Isso quer dizer que algum ícone de remoção e edição de cartões não funcionou. Eles aparecem em branco, não foram carregados corretamente.

Preciso acrescentá-los ao `offline.manifest`:

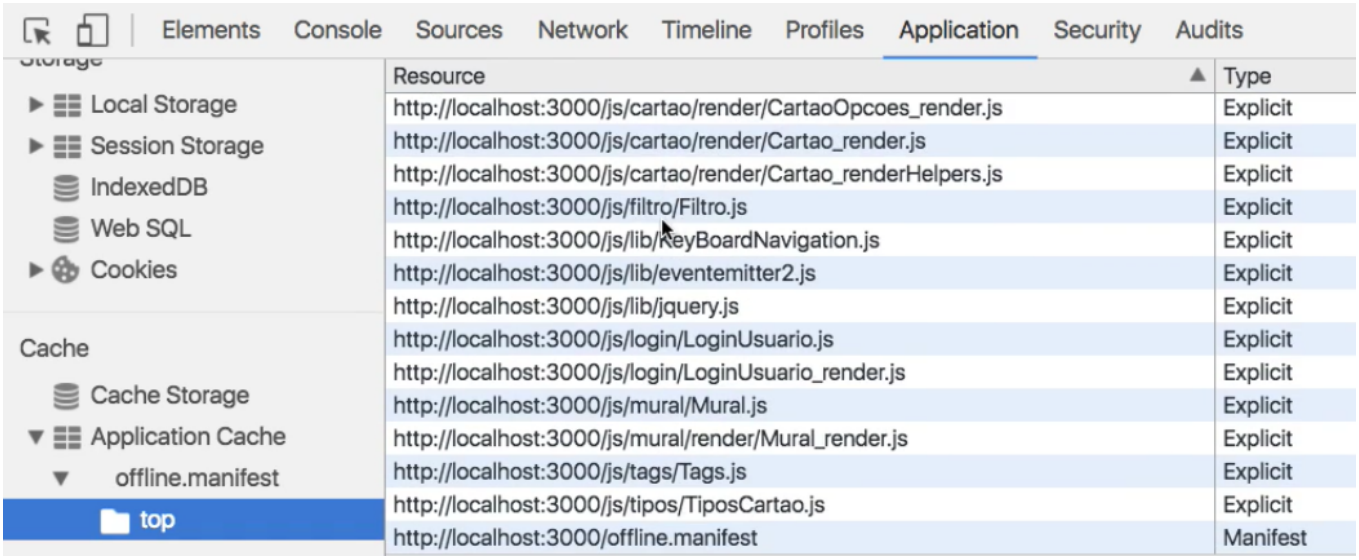
```
CACHE MANIFEST

CACHE:
css/estilos.css
css/opcoesDaPagina.css
css/opcoesDoCartao.css
css/cabecalho.css
css/login.css
css/loginForm.css
css/loginStatus.css
css/cartao.css
css/novoCartao.css
css/mural.css
js/lib/jquery.js
js/lib/eventemitter2.js
js/lib/KeyBoardNavigation.js
js/tags/Tags.js
js/cabecalho/mudaLayout.js
js/cabecalho/busca.js
js/filtro/Filtro.js
js/tipos/TiposCartao.js
js/cartao/render/Cartao_renderHelpers.js
js/cartao/render/CartaoOpcoes_render.js
js/cartao/render/CartaoConteudo_render.js
js/cartao/render/Cartao_render.js
js/cartao/Cartao.js
```

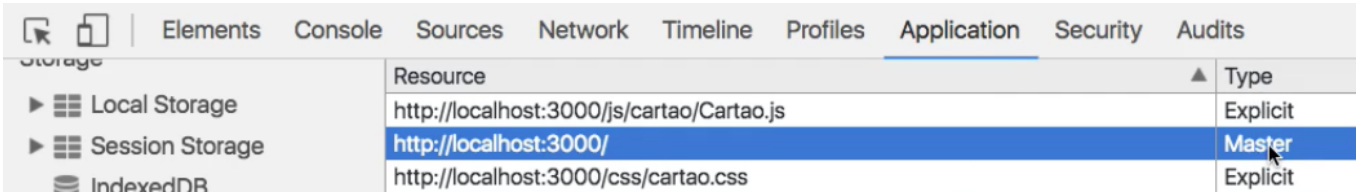
```
js/login/LoginUsuario_render.js
js/login/LoginUsuario.js
js/mural/render/Mural_render.js
js/mural/Mural.js
js/cabecalho/novoCartao.js
img/bin2.svg
img/edit.svg
```

Vamos limpar o console, recarregar a página, funcionou. São 31 arquivos agora, mas... Os ícones continuam não aparecendo. O que aconteceu é que ele só instalou a app, sem recarregar o que tínhamos antes, é assim que o Application Cache funciona. Quando você faz alterações, não adianta simplesmente recarregar a página, isso só fará com que os arquivos sejam baixados. Se você quiser ver o resultado da app nova, é preciso carregar a página novamente. São dois "carregamentos" que precisam ser feitos.

Lá na aba Application, se for ao Application Cache, veremos o `offline.manifest` e todos os seus arquivos sendo carregados:



Vocês podem notar que o arquivo `index.html` não se encontra listado em `offline.manifest`. Isto não é um problema, pois é automaticamente "cacheado", adicionado ao Application Cache, como visto abaixo:



O *localhost:3000* aparece como sendo do tipo **Master**, ou seja, trata-se de um arquivo principal da app.

Por fim, vamos ficar offline para testá-la, atualizando a página mais uma vez. Nossa app funciona sem conexão nenhuma.