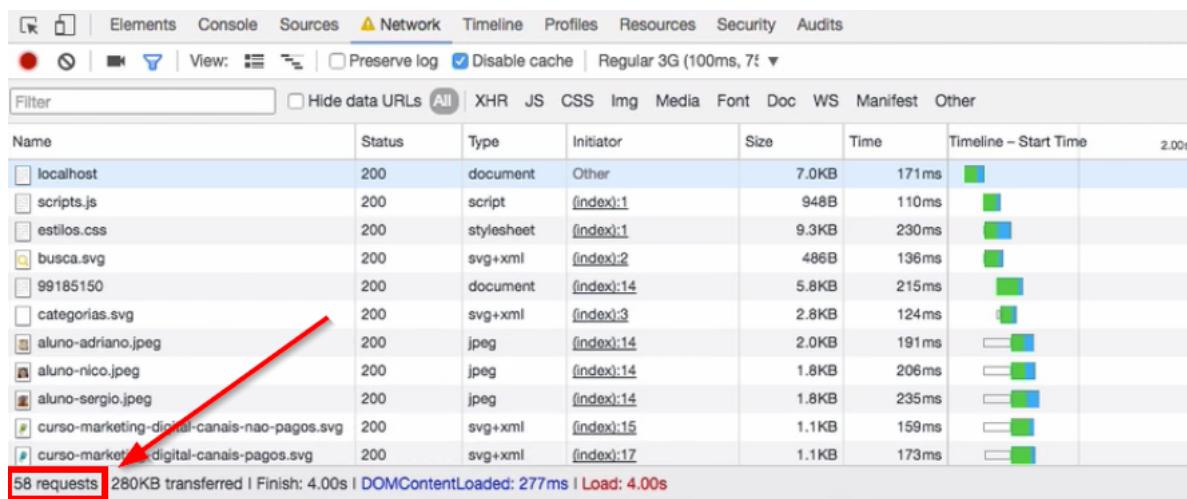


06
Transcrição das aulas

Lembrando que temos seis conexões disponíveis no site e enquanto as seis estiverem em uso se precisarmos fazer um novo *request* ele terá que aguardar. Mas, na confecção de um *site* moderno, será que seis conexões é o suficiente?

Podemos observar que mesmo com as diversas otimizações que realizamos, ainda temos um total de 58 *requests*.



Como podemos fazer para baixar tudo isso sem passar pelo problemas do enfileiramento?

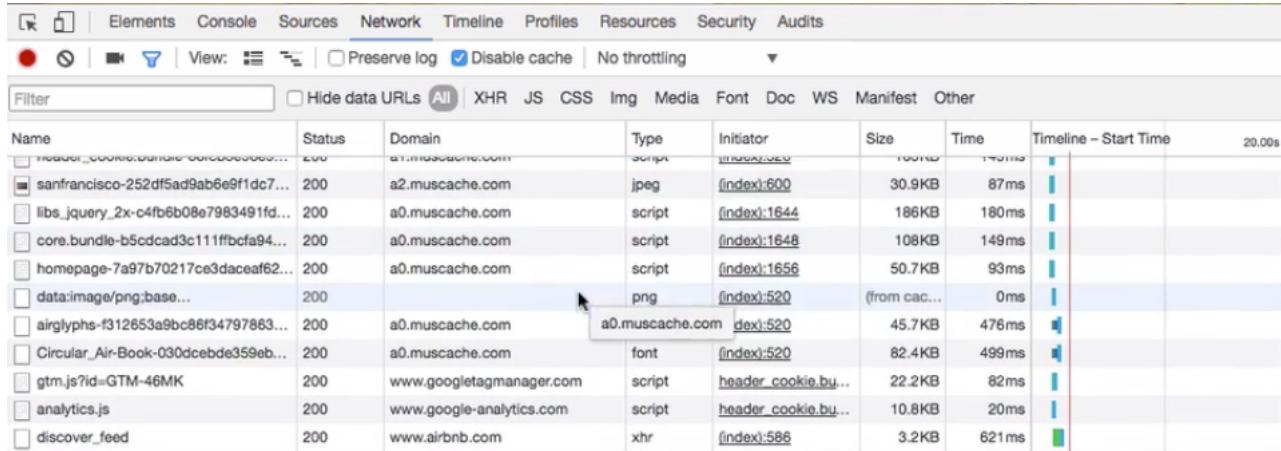
Existe uma técnica que vamos analisar agora. Na verdade as conexões são por *host name*, por origem. Estamos acessando o site através do performance-sergio.appspot.com e ele está baixando o conteúdo usando as seis conexões.

Mas se tivermos recursos de lugares externos, de outros *host name*, ele abre conexões diferentes. Por exemplo, no caso de um vídeo do site vimeo.com, ele terá que abrir uma conexão com o link, ou seja, externa.

Bom, agora que sabemos disso podemos utilizar esse recurso a nosso favor.

O que podemos fazer? Puxar eles para o começo através de um segundo *host name*. Poderíamos, por exemplo, nesse segundo *host name* colocar as imagens. A ideia é que ele não irá competir com as seis conexões do site principal, evitando o indesejado encavalamento.

Muitos sites utilizam esse recurso, vamos observar um exemplo, o site *airbnb*, www.airbnb.com.br/ (<http://www.airbnb.com.br/>). Se olharmos na aba de *Network* podemos através do botão direito do mouse selecionarmos a opção *Domain*, de dominio para observarmos de onde estas imagens estão sendo baixadas. Podemos visualizar que elas são baixadas de distintos locais:



The screenshot shows the Network tab in the Chrome DevTools. It lists various network requests with their details: Name, Status, Domain, Type, Initiator, Size, Time, and Timeline – Start Time. A tooltip is visible over one of the requests, showing the initiator as 'a0.muscache.com' and the file path as 'dexi:520'. The table has columns for Name, Status, Domain, Type, Initiator, Size, Time, Timeline – Start Time, and a timestamp of 20.00s.

Name	Status	Domain	Type	Initiator	Size	Time	Timeline – Start Time	20.00s
modern_cookie_bar.js	200	a1.muscache.com	script	[index]:600	100KB	145ms		
sanfrancisco-252df5ad9ab6e9f1dc7...	200	a2.muscache.com	jpeg	[index]:600	30.9KB	87ms		
libs.jquery_2x-c4fb6b08e7983491fd...	200	a0.muscache.com	script	[index]:1644	186KB	180ms		
core.bundle-b5cdcad3c11ffbcfa94...	200	a0.muscache.com	script	[index]:1648	108KB	149ms		
homepage-7a97b70217ce3daceaf62...	200	a0.muscache.com	script	[index]:1656	50.7KB	93ms		
data:image/png;base...	200		png	[index]:520	(from cac...)	0ms		
airglyphs-f312653a9bc88f34797863...	200	a0.muscache.com	a0.muscache.com	dexi:520	45.7KB	476ms		
Circular_Air-Book-030dcebde359eb...	200	a0.muscache.com	font	[index]:520	82.4KB	499ms		
gtm.js?id=GTM-46MK	200	www.googletagmanager.com	script	header cookie bu...	22.2KB	82ms		
analytics.js	200	www.google-analytics.com	script	header cookie bu...	10.8KB	20ms		
discover_feed	200	www.airbnb.com	xhr	[index]:586	3.2KB	621ms		

A ideia é que quando temos *host name* distintos o navegador pode abrir mais seis conexões distintas para cada uma das origens e eles acabam não competindo entre si.

E como fazemos isso?

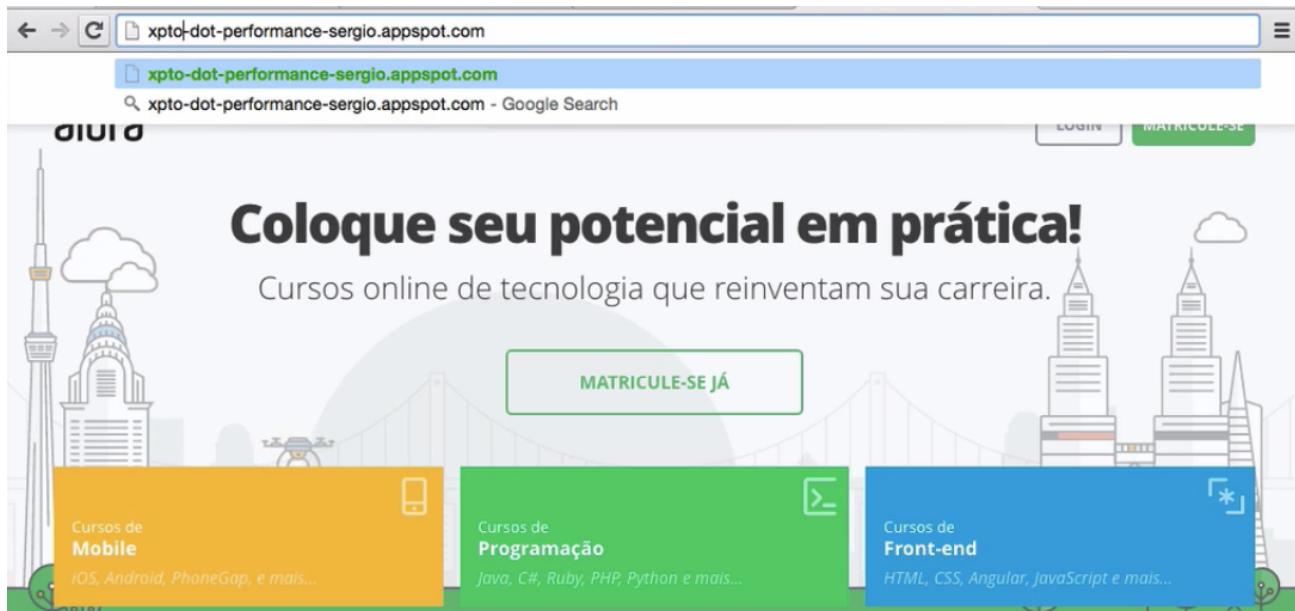
Precisaremos de um segundo domínio e para nossa sorte é muito fácil adquirir um quando estamos usando o *app engine*. O site que temos, o performance-sergio.appspot.com, também está disponível no web-dot-performance-sergio.appspot.com. Por que temos dois *host names* para o mesmo site? Na verdade isso está relacionado ao *app.yaml*. Deixamos nele embutida uma versão *web* que podemos, inclusive, alterar para qualquer coisa. Podemos alterar, por exemplo, por "xpto" e na hora que subirmos o *deploy* teremos xpto-performance-sergio.appspot.com.



```

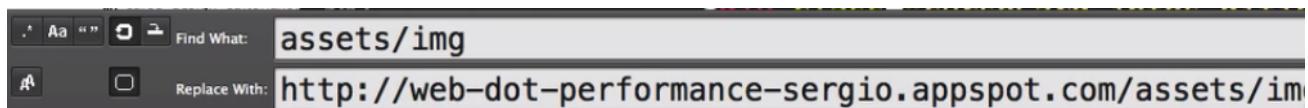
1 # Arquivo de configuração para deploy no Google App Engine que veremos
2
3 application: performance-sergio
4 runtime: php55
5 api_version: 1
6 version: xpto
7
8 handlers:
9   - url: /
10     static_files: index.html
11     upload: index.html
12
13   - url: /assets
14     static_dir: assets

```



Então, temos dois *host name* de "brinde". Isso é ótimo, pois, podemos usá-los de maneira a enganar o navegador e fazer ele abrir outras seis conexões. Vamos fazer isso?

Vamos lançar o "web-dot-performance-sergio.appspot.com" e usaremos ele para algumas das requisições. No caso, as *requests* de imagens. Para isso, pegaremos tudo que possui requisições de imagens. Vamos usar tudo isso no novo domínio, o que fazemos é um *replace* onde buscarmos "*assets/img*" e vamos trocar pelo endereço web-dot-performance-sergio.appspot.com/assets/img.



Na prática o que estamos dizendo é que o *svg* vai ser baixada da url que acabamos de escolher.

Temos alguns detalhes que devemos prestar atenção. Temos o arquivo do logo e nele fizemos um `inline` e o *search replace* colocou

```

```

Temos também que alterar os *svgs* das categorias que não podem estar em outro *host name*, caso contrário seu recurso também não funcionará. Ele possui um `xlink:href` que serve apenas para o site em ocasião. Vamos apagar o link pelo *search replace* do novo domínio.

Vamos remover isso para que não estrague nada. Falta, no *Sublime* dar um *gulp useref*, isto é, "buildar".

O resto deixaremos igual e não teremos problema. Vamos testar para ver se tudo está ocorrendo bem! Vamos observar o *dev tools*. Temos várias coisas baixadas no *localhost*, mas também do <http://web-dot-performance-sergio.appspot.com/assets/img/logo-alura.svg>.

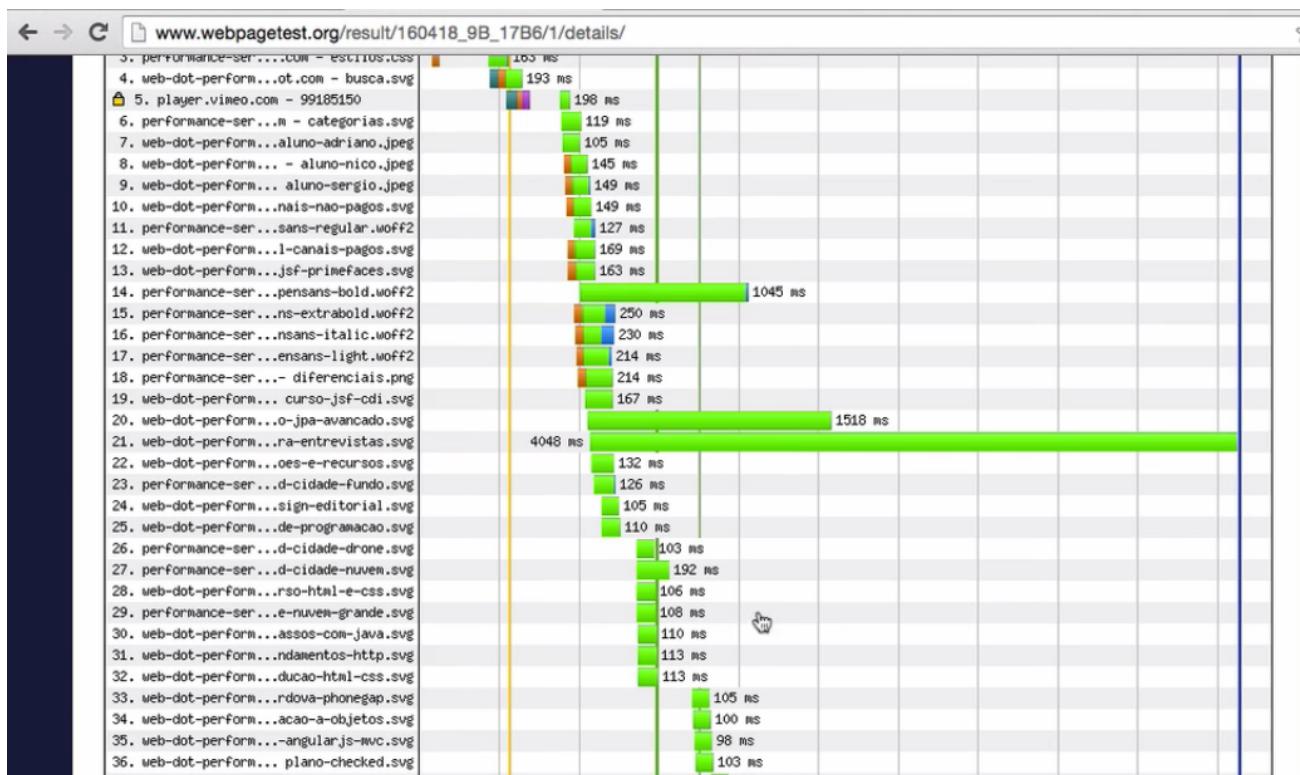
Repare que os *request* na segunda *home* já não brigam mais com os da *local home*. Fazendo isso conseguimos diminuir o tempo de carregamento deles. O carregamento dos *svgs* ainda são de seis em seis, mas eles brigam, agora, apenas entre eles e não com os outros recursos da pagina.

O que podemos fazer?

Vamos fazer um *deploy* para observar o que acontece na prática.

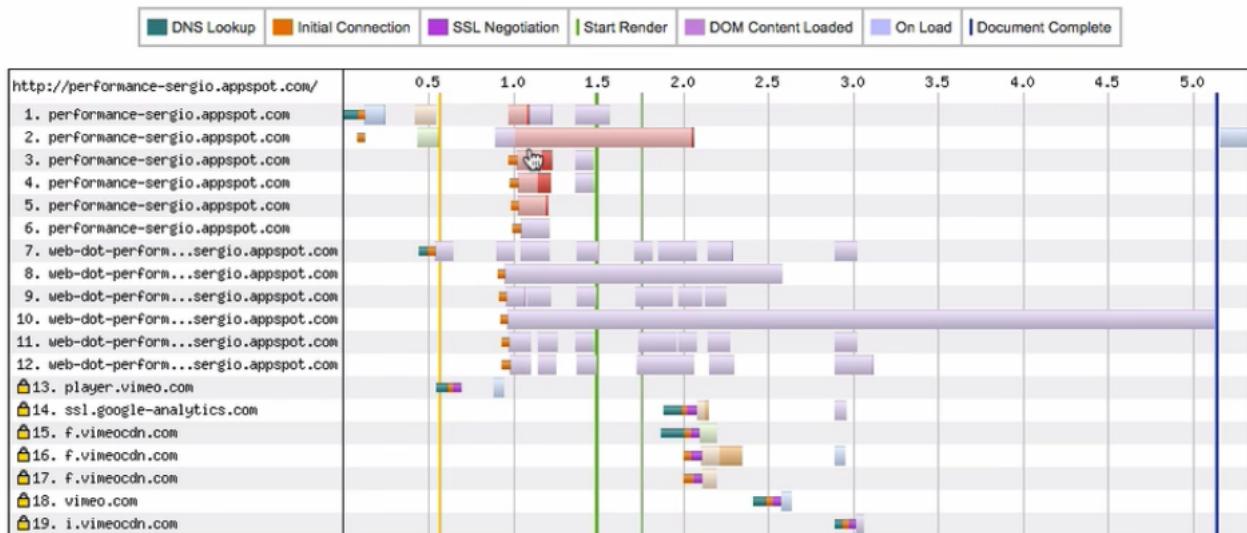
Name	Status	Domain	Type	Initiator	Size	Time	Timeline – Start Time	2.00s
performance-sergio.appspot.com	200	performance-sergio.appspot.com	document	Other	6.2KB	194ms	<div style="width: 10%;"> </div>	
scripts.js	200	performance-sergio.appspot.com	script	[index]:1	892B	193ms	<div style="width: 10%;"> </div>	
estilos.css	200	performance-sergio.appspot.com	stylesheet	[index]:1	7.3KB	201ms	<div style="width: 10%;"> </div>	
busca.svg	200	web-dot-performance-sergio.ap...	svg+xml	[index]:2	461B	208ms	<div style="width: 10%;"> </div>	
99185150	200	player.vimeo.com	document	[index]:14	7.9KB	167ms	<div style="width: 10%;"> </div>	
aluno-adriano.jpeg	200	web-dot-performance-sergio.ap...	jpeg	[index]:14	2.0KB	192ms	<div style="width: 10%;"> </div>	
categorias.svg	200	performance-sergio.appspot.com	svg+xml	[index]:3	2.6KB	206ms	<div style="width: 10%;"> </div>	
aluno-sergio.jpeg	200	web-dot-performance-sergio.ap...	jpeg	[index]:14	1.8KB	194ms	<div style="width: 10%;"> </div>	
aluno-nico.jpeg	200	web-dot-performance-sergio.ap...	jpeg	[index]:14	1.8KB	198ms	<div style="width: 10%;"> </div>	
curso-marketing-digital-canal...	200	web-dot-performance-sergio.ap...	svg+xml	[index]:15	1.0KB	190ms	<div style="width: 10%;"> </div>	

Vamos observar na aba *network* o que ocorre se dermos um *refresh*. Selecioneamos para visualizar os *domínios* e percebemos que temos várias coisas que baixam do domínio principal e coisas que baixam do domínio secundário. Podemos realizar um teste no site *Web Page Test* para observar os impactos, na prática, do que fizemos. Acessamos (<http://www.webpagetest.org/>)<http://www.webpagetest.org/> (<http://www.webpagetest.org/>) e colamos o link do nosso site e pedimos para iniciar o teste. Vamos nos deter, principalmente, no *connection view* para, justamente, visualizarmos que temos mais do que apenas seis conexões. Repare:



Temos o gráfico em forma de cascata e repare que ele é distinto daquele que vimos a primeira vez que acessamos o site. Nesse gráfico temos conteúdos que foram baixados de outro *home* intercalados com os que foram baixados no *local home*. Vamos reparar isso no gráfico do *Connection View*:

Connection View



Podemos visualizar através dessa imagem as seis conexões que possuímos com o domínio principal e outras seis novas conexões com nosso domínio secundário. A ideia é que conseguimos paralelizar em doze conexões ao em vez de seis. Perceba que existe um limite prático da própria rede do quanto ela pode aguentar em termos de paralelismo, se não, ela corre o risco de saturar.

Geralmente, trabalhar com dois ou três *hosts* é o ideal. Na verdade, o ideal mesmo é testar isso e observar na prática o gráfico *waterfall* e tentar perceber se utilizar-se dessa prática está sendo vantajoso ou não, se está ocorrendo um ganho de performance ou não. A ideia é utilizar *hosts* secundários com coisas estratégicas, ou seja, prioritárias.

A ideia foi passar que a paralelização de *host name* é uma maneira de ir contra ou enganar a limitação do navegador de seis conexões simultâneas que possuímos no *http1*.