11

Transcrição das aulas

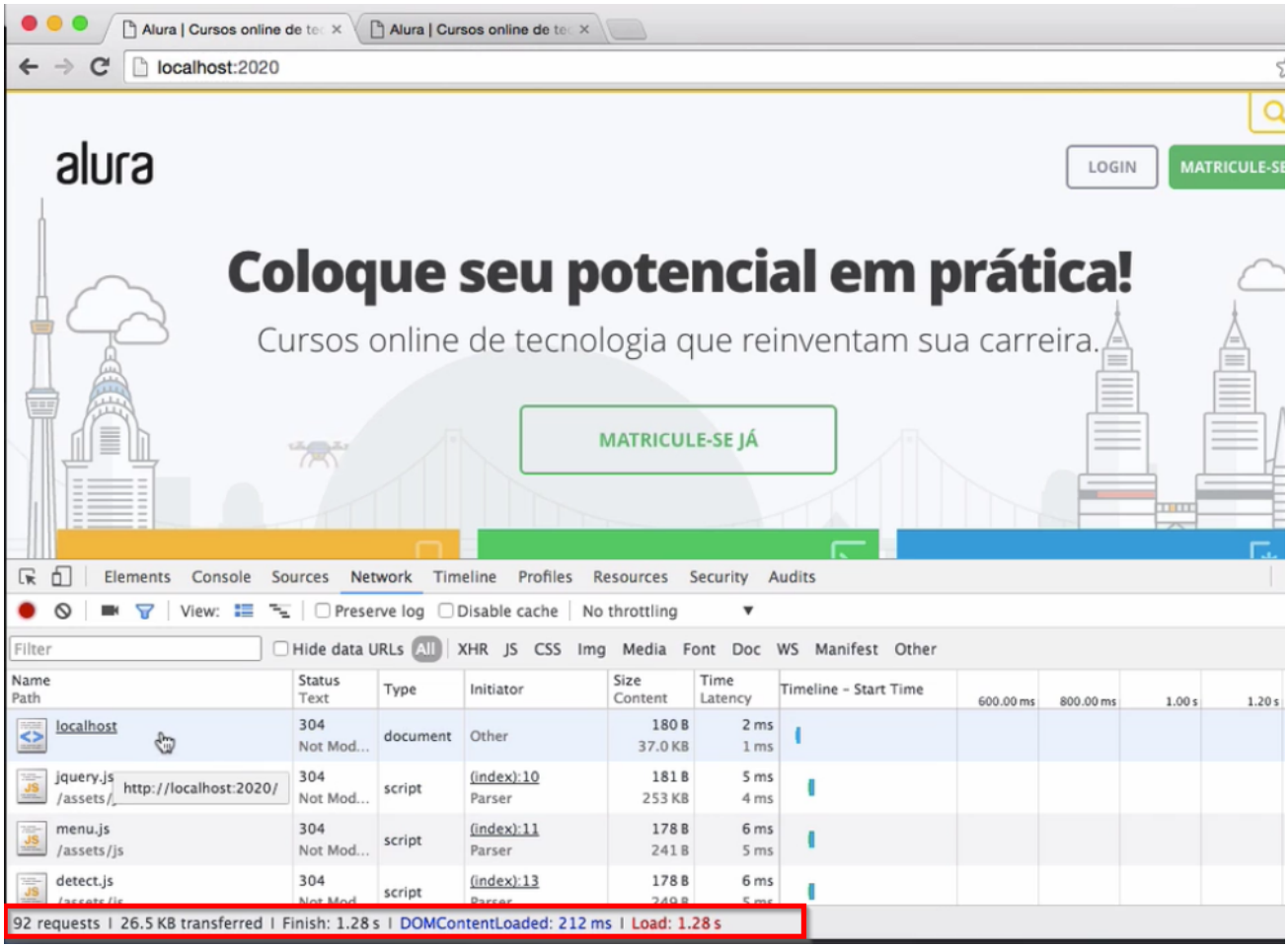
Já fizemos diversas otimizações no site, agora, vamos rodar a versão otimizada do site e também a versão não otimizada. Vamos comparar o que fizemos e já preparar o terreno para que sigamos fazendo melhorias.

Lembra-se que temos duas pastas, a "site" que contêm as informações originais e a "dist" que possui o conteúdo já otimizado. Vamos abrir essas duas versões no navegador e comparar ambas. Nós vamos utilizar o servidor *nginx* , então, configuramos dois *nginx* diferentes, colocamos duas portas portas diferentes, uma servindo a pasta "site" original e o outra a pasta "dist". Na pasta "dist" aproveitamos para utilizar o `gzip on` como vimos anteriormente. Teremos o seguinte:

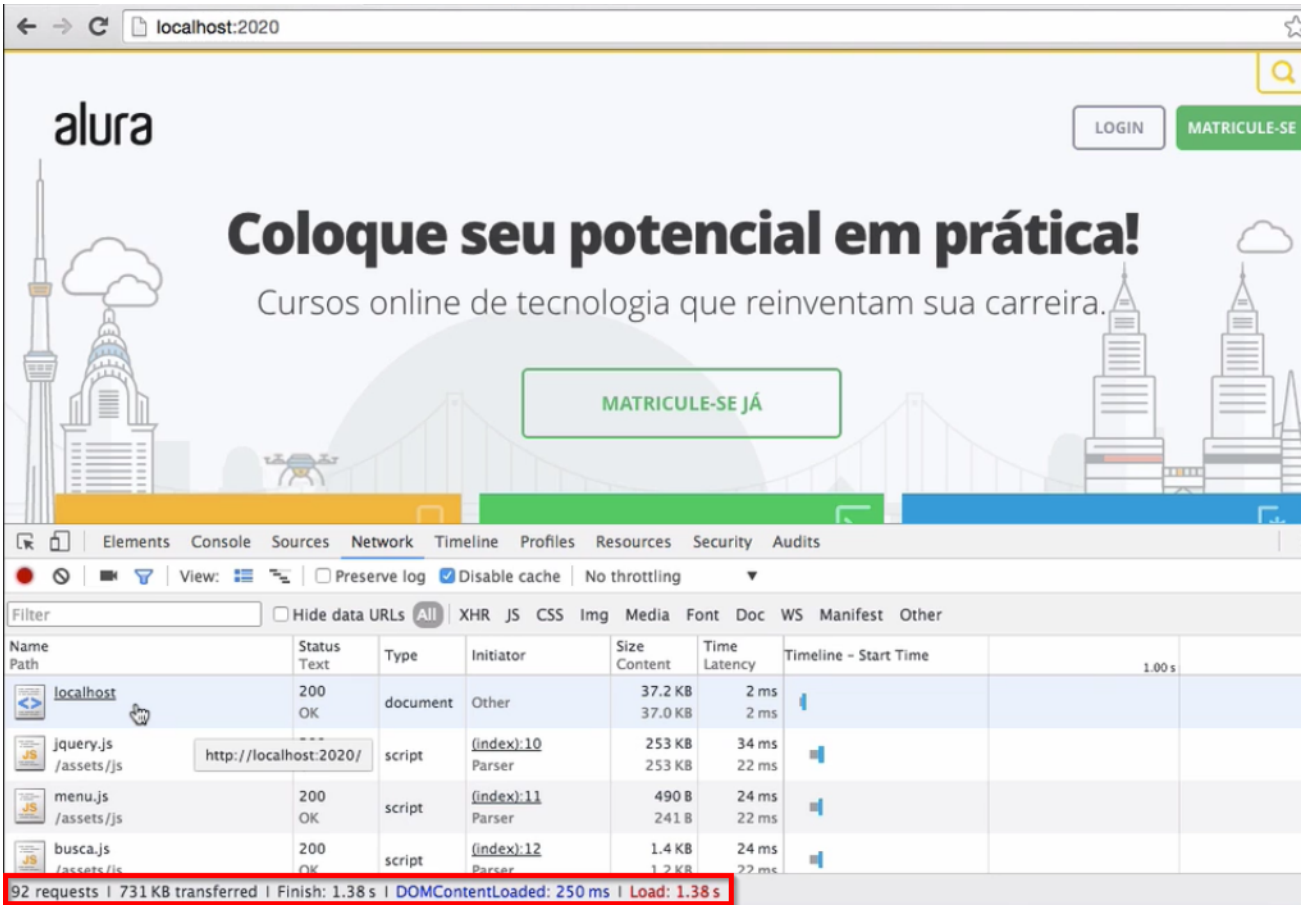
```
server {
        listen 2020;
        root /Users/alura/performance-web/site/;
}

server {
        listen 3030;
        root /Users/alura/performance-web/dist/;

        gzip on;
        gzip_types text/css application/javascript image/svg+xml;

}
```

Vamos voltar ao navegador, temos um "localhost:2020" que é para a pasta "site" e um "localhost:3030" que é para a pasta "dist". Independente do navegador que você estiver utilizando o importante é que isso seja feito para colocarmos as duas lado a lado para comparação.

Vamos digitar "F12" para abrir o *dev tools*. Estamos utilizando o *Chrome*, vamos abrir a aba *network*, mas temos também essa aba em outros navegadores. Nessa aba teremos tudo aquilo que é relacionada a rede e que é feito na página, as requisições, toda vez que acessamos a página ele tem que fazer uma requisição para o servidor para baixar o *html*, depois, ele tem que ler o *html* e descobrir quais são as imagens, os *javascript*, os *css* que serão baixados e cada uma dessas coisas é uma requisição. Se atualizarmos a página ele grava todas as requisições que são feitas. Observe:



Podemos ver na linha de baixo a quantidade de *requests*, requisições, que ele fez e que estes 92 *requests* baixaram "26 KB" apenas. Na verdade o que aconteceu aqui é que nossa página não está tão otimizada assim, como nossa página já estava aberta ele pegou isso no *cache*, isto é, quando ele diz "not modified" nos ícones, ele diz que não fez o *download* desse arquivo novamente, pois já estava aberto, já estava no *cache*. Podemos selecionar a opção *Disable cache* para que ele desconsidere os *cache* do navegador enquanto o *dev tools* estiver aberto. Depois de selecionar essa opção vamos dar um "Enter":



Agora, temos uma sensação mais próxima da realidade, ele nos afirma que baixou cerca de "731 KB". Bom, estávamos em nossa versão original. Vamos comparar, agora, com a versão otimizada. Fazemos a mesma coisa, entramos no "localhost:3030", abrimos o *dev tools* e selecionamos a opção *Disable cache* e temos "331 KB".

Temos, aqui, uma grande diferença, caímos pela metade fazendo nossas otimizações, otimizamos as imagens, que são o ponto mais pesado da página, gastamos um tempo retirando o *jquery* não minificado e, principalmente, nos dedicamos habilitando o *gzip*, então, o *html* do *localhost* possui "32.7 KB", mas, na versão *gzipada* ele possui "6.3 KB", ou seja, transfere apenas isso para ter o mesmo *html* que na versão original possui "37 KB".

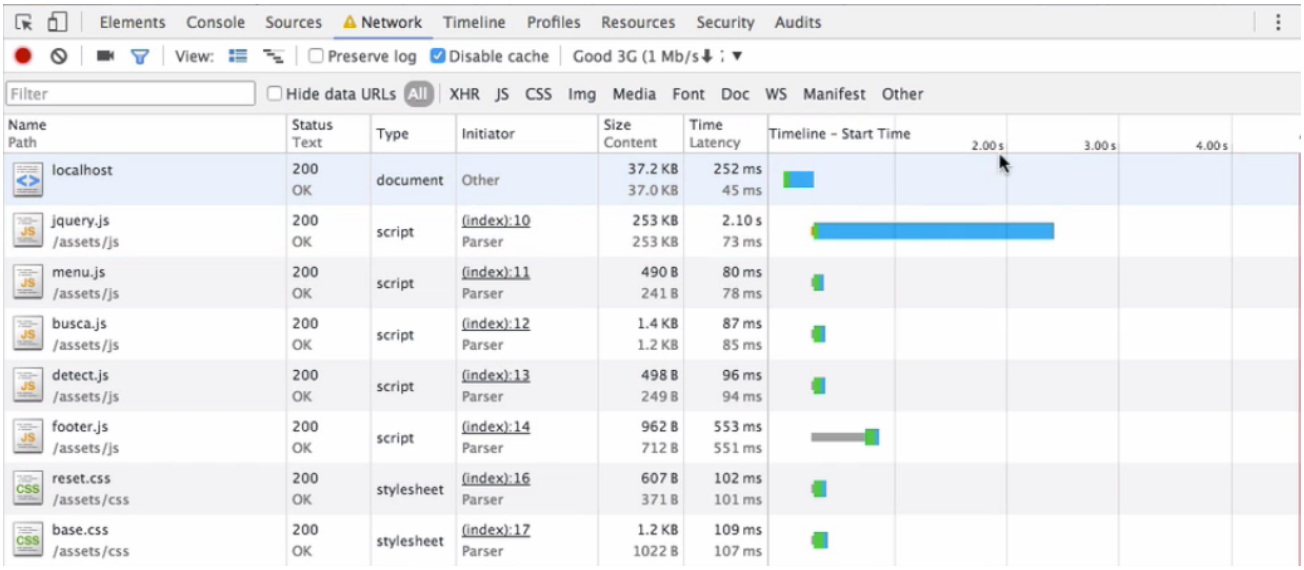
Repare que as duas versões, entretanto, continuam instantâneas, ou seja, nem conseguimos notar que ocorreu uma otimização, se não olhássemos o tamanho.

O ideal é fazer um teste no "mundo" real, ou seja, colocamos o site na Internet e acessamos uma rede real, de preferência, uma rede mais próxima ao que os usuários costumam utilizar, pois, o *localhost* não é a situação de ninguém.

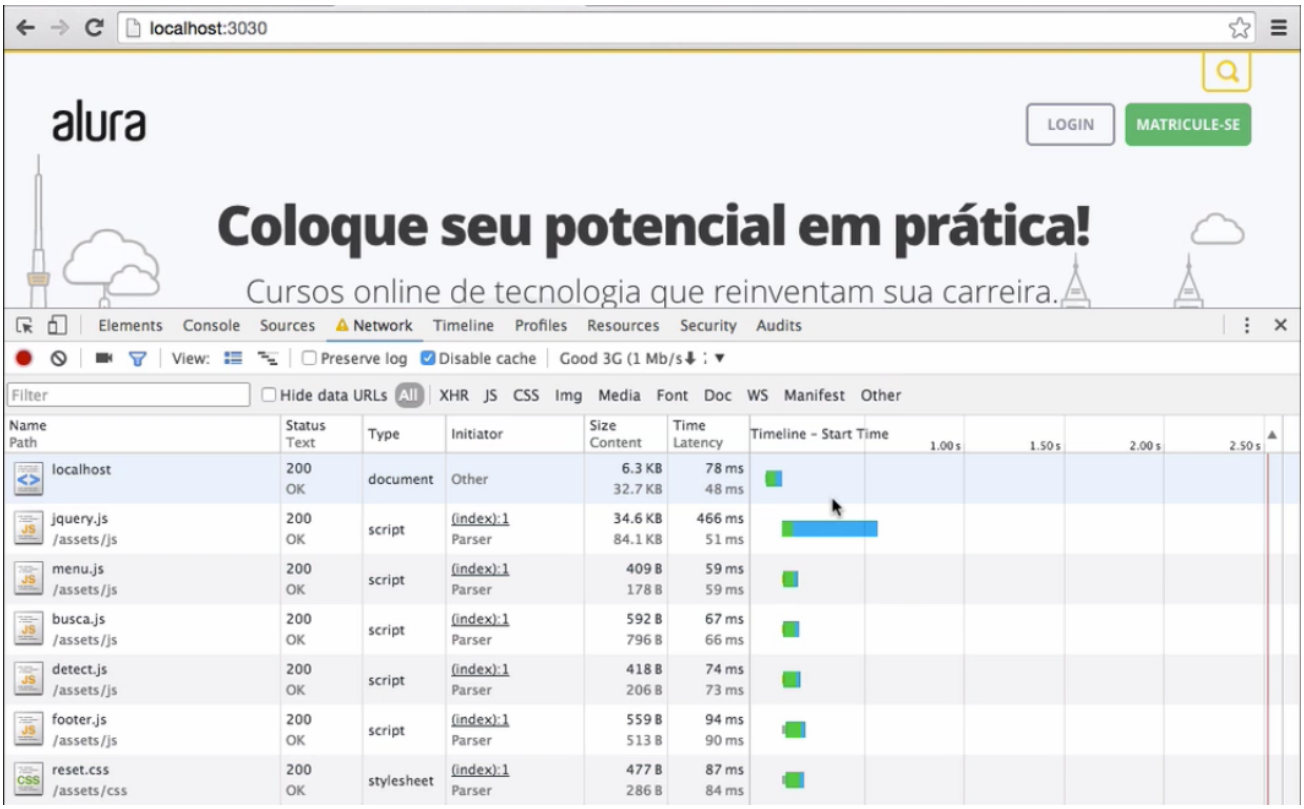
Mas, e se não tivermos subido ainda o conteúdo... Como podemos fazer para ter um teste um pouco melhor do que estamos fazendo?

No *dev tools* mesmo existe a opção de *No throttling* onde podemos escolher algum tipo de rede para ser emulada. Inclusive, se clicarmos no *No throttling* ele nos mostra várias redes pré definidas. A opção *No throttling* quer dizer que ele vai rodar localmente na maior velocidade possível, por isso, nem conseguimos perceber as diferenças entre um e o outro. Poderíamos simular escolher dentre as redes pré definidas, por exemplo, "3G" bom, assim conseguimos uma simulação mais próxima da realidade.

Selecionando essa opção, e quando atualizamos novamente nossos navegadores já conseguimos perceber que demora mais para carregar as páginas. E mais ainda, vamos observar as diferenças no gráfico, que mostram, no tempo, quanto demorou para baixar cada requisição. Vamos visualizar os gráficos da versão que não foi otimizada:

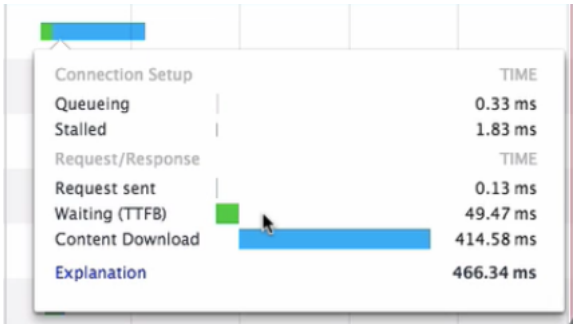


Vamos observar o gráfico da versão que foi otimizada:



Observe a diferença das requisições para *jquery*, a primeira eram quase 2 segundos para baixar e a segunda é meio segundo, na versão otimizada.

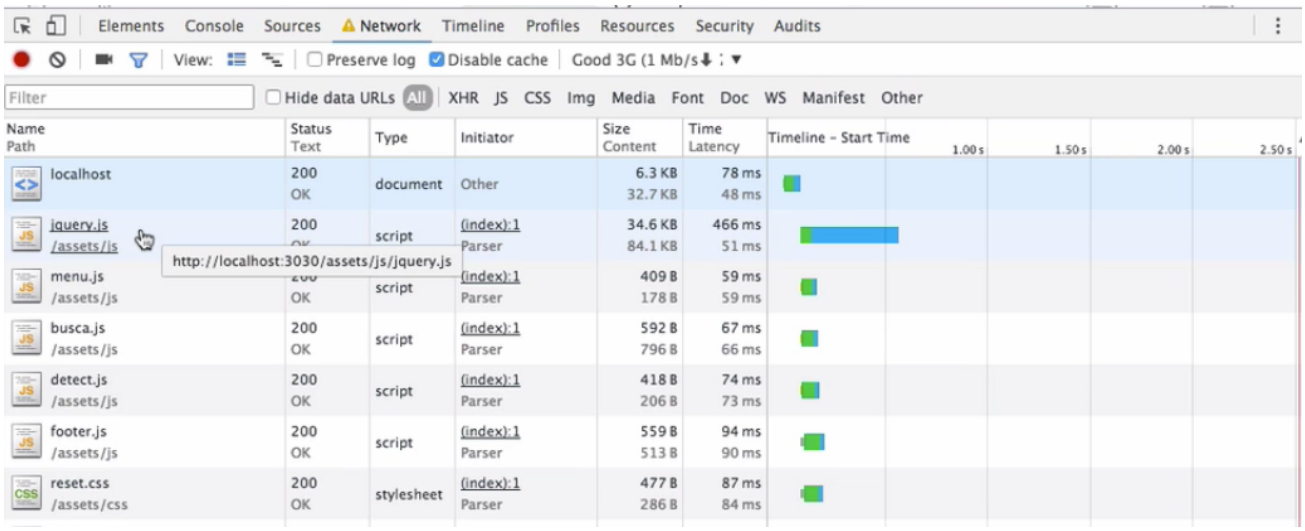
Vamos tentar aprender a ler esse gráfico, para isso podemos passar o mouse em cima e ele nos informa o que cada cor é. As cores representam tempos diferentes. A barra azul diz quanto tempo demorou para baixar o conteúdo. Mas, antes mesmo de baixar, o navegador fica esperando e esse tempo de espera equivale a linha verde. Esse tempo de espera é o equivalente ao tempo antes do primeiro *byte* chegar. Por que demora isso? Por que o navegador olha para o *html* e dispara uma requisição de *jquery* para o servidor que recebe essa requisição, processa ela e começa a enviar resposta. O tempo do primeiro *byte* chegar é o que está sendo medido na barra verde. E isso depende de algo que denominamos latência, quanto maior latência na rede, maior será essa demora. Latência é quanto tempo demora para o *byte* ir e voltar. Quanto pior a rede, mais interferência e maior a distância, maior a latência da rede.



Temos alguns recursos que possuem uma barra cinza. Clicando em um desses itens podemos ver o que ela significa. A barra cinza indica que está *stalled*, está parado, travado e aguardando para fazer o *request*, ou seja, no comprimento da barra cinza o *request* ainda nem foi feito. Podemos perceber que a barra cinza vai aumentando a medida que nós vamos rolando a barra para baixo.

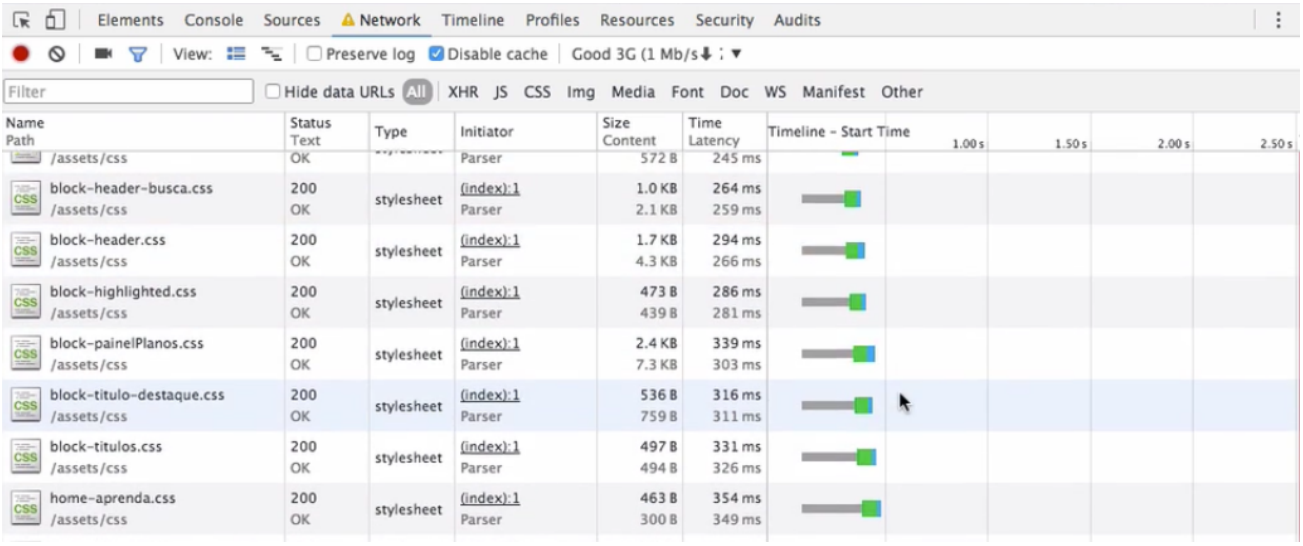
E por que isso ainda não foi inicializado? Porque o navegador possui um limite.

O navegador abre por padrão seis requisições simultâneas com o servidor. Quando você abre pela primeira vez um site é como se ele abrisse seis conexões de rede e essas seis conexões são utilizadas para baixar coisas em paralelo, uma em cada conexão, assim, se temos mais do que seis itens que devem ser baixadas, as demais devem aguardar. É a isso que se refere essa barra cinza. Podemos visualizar isso através do início do *navegador* no *dev tools*.



Ele baixa primeiro seis coisas que quase não tem barra cinza. E essas seis coisas são escolhidas por uma questão de prioridade. Por exemplo, o *html* é o primeiro, o *jquery* é o segundo e etc.

Podemos inclusive perceber que o cinza termina mais ou menos no tempo em que durou o verde do que liberou espaço para ele. E isso vai virando uma cascata, os últimos devem aguardar muito mais que os primeiros.



E agora, temos os icones de curso e eles estão baixando em quase 2 segundos. E por que isso acontece? Ele sabia que o arquivo existia, mas não havia nenhuma conexão disponível para baixar, devido ao limite de seis conexões simultâneas.

Podemos pensar, a primeira vista, que isso é muito pouco, mas se tivermos muito mais do que isso iria saturar a própria rede e também o servidor. Esse número seis é um pouco mágico, os próprios navegadores perceberam que muito mais que seis conexões não funciona, as conexões iriam se atropelar e isso traria problemas. Seis é um número mais ou menos ideal.

Vemos nesse gráfico, um gráfico que por sinal é chamado de *Waterfall*, justamente, por parecer uma cascata, uma cascata de requisições. Através dele conseguimos percebe que as requisições mais de cima atrasam mais do que as de baixo. Isso pode ser ruim e acarretar gargalos de performance no site.

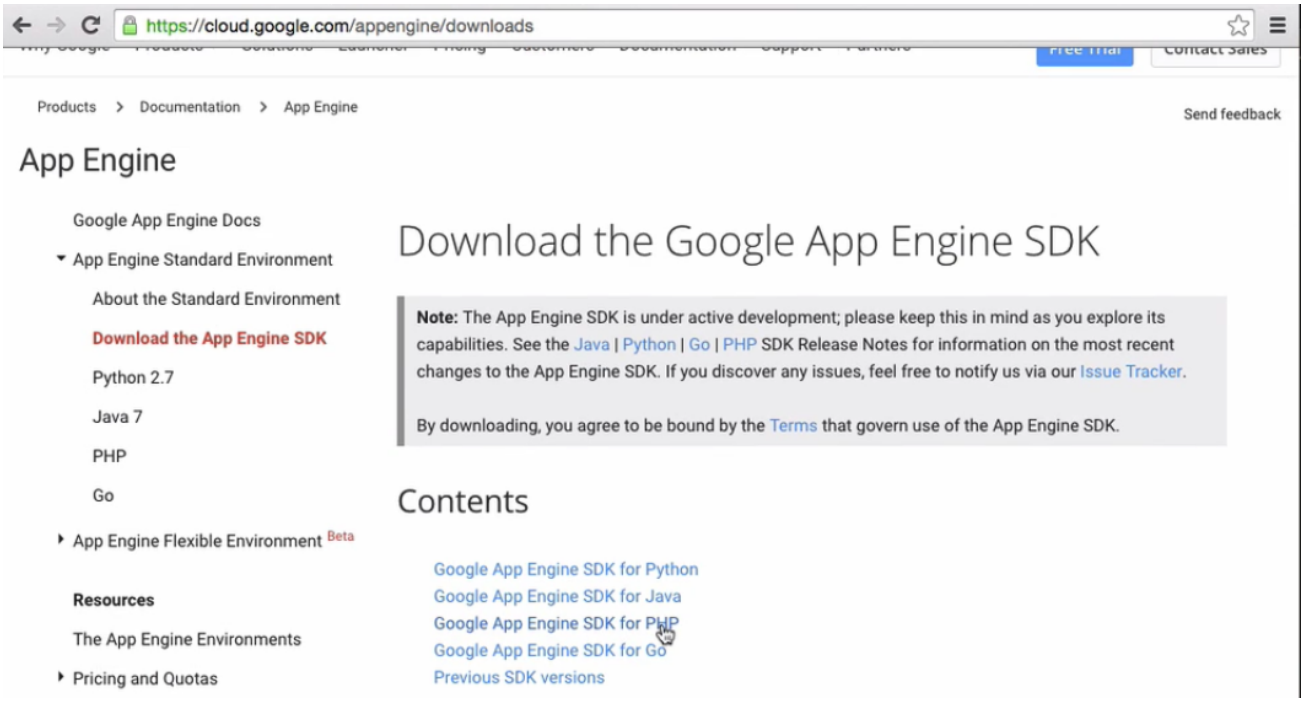
Aprendemos, nesse momento, a usar o *dev tools* do *Chrome* e de qualquer outro navegador, que terão ferramentas parecidas, para entendermos a sequência de requisições do navegador e o porquê disso ser um gargalo de performance. Acabamos percebendo que um determinado ícone está tardando demais para carregar e isso pode ser um indicio que coisas mais na frente estão travando e usando todas as conexões disponíveis e o ícone que está tardando não chega nunca.

Veremos como melhorar esse aspecto mais adiante.

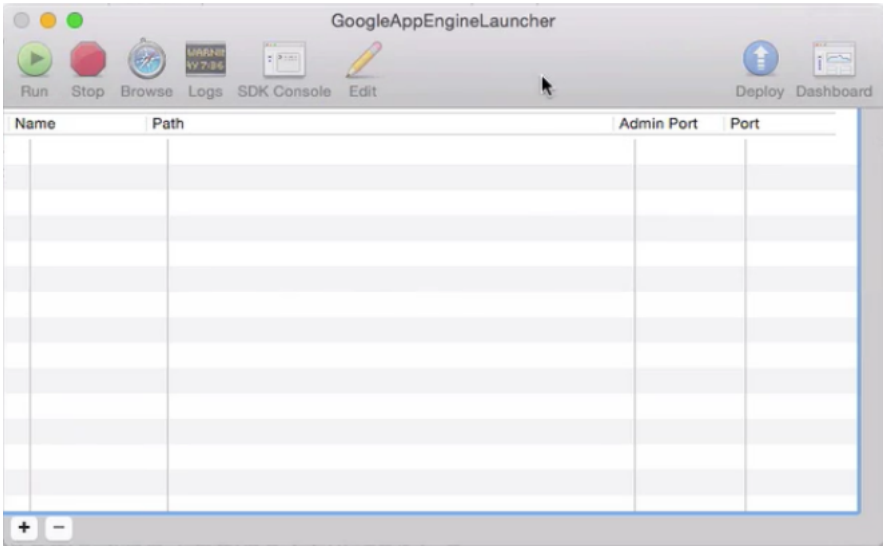
Existem algumas ferramentas para testarmos a performance do site que são serviços online. Para isso precisamos que nossso site esteja disponível na Internet, ele precisa estar em um servidor remoto, também possuir uma *url* que possamos para esse servidor. Até esse momento testamos tudo em *localhost*, mas não conseguimos testar isso fora da máquina.

Vamos dar uma pausa na *performance* para subir isso em algum servidor de verdade e poder testar as ferramentas. Você pode utilizar o servidor que quiser, mas recomendamos o servidor do *google*, o *app engine sdk*. Ele é um *cloud* do google onde podemos subir diversas "apps", java, *em* php *e etc. E ele possui uma parte gratuita bem interessante, podemos usar um servidor simples e gratuitamente. E ele já vêm com* hpp2* que discutiremos o que é mais adiante.

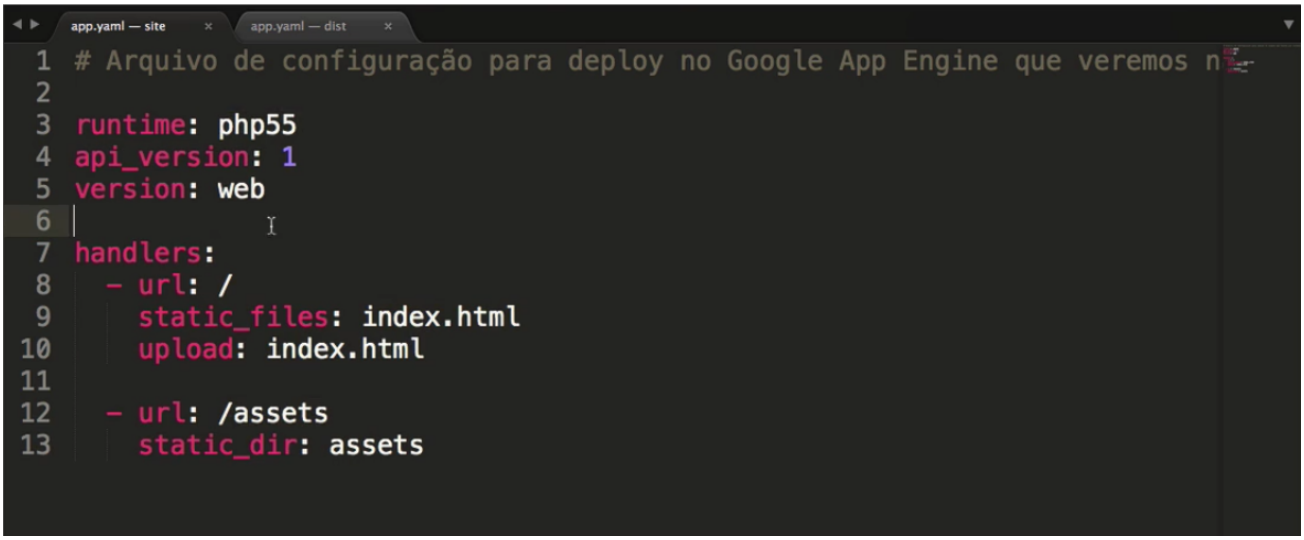
Primeiro, vamos baixar o *google app sdk*. Vamos entrar no site: cloud.google.com/appengine/downloads.



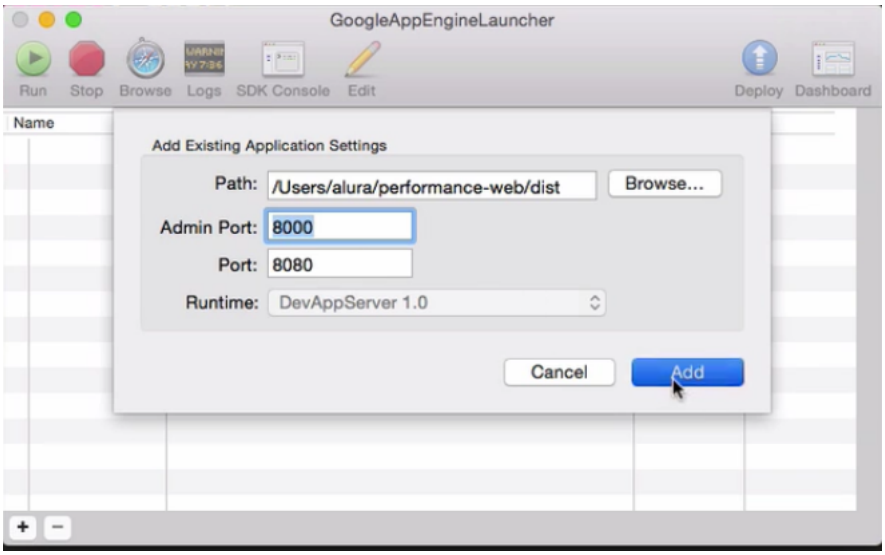
Existem várias versões de *sdk*, nós vamos baixar em *php*, mas poderiam ser outras. Inclusive, existem versões para *Mac* e *Windows*. Vamos baixar isso e abrir, teremos o seguinte:



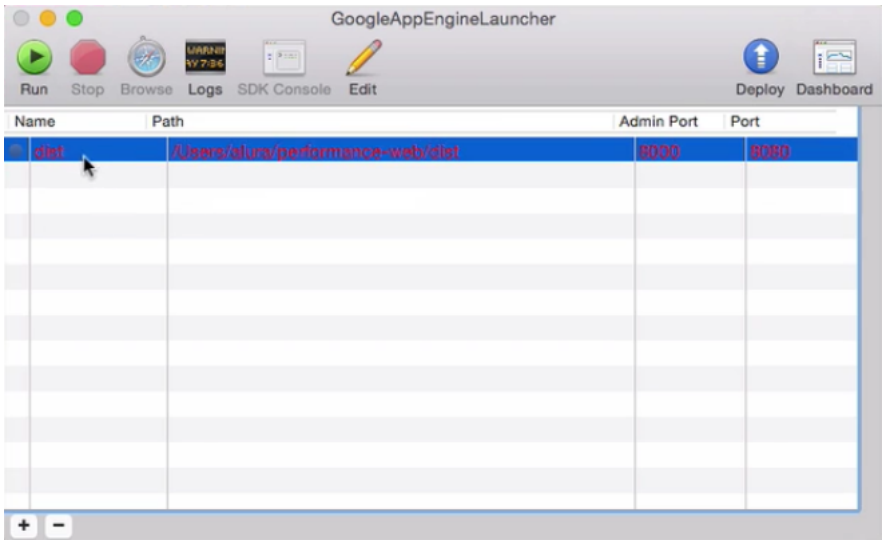
Se abrirmos o nosso projeto veremos que agora temos um arquivo chamado "app.yaml" que é um arquivo de configuração do *google*.



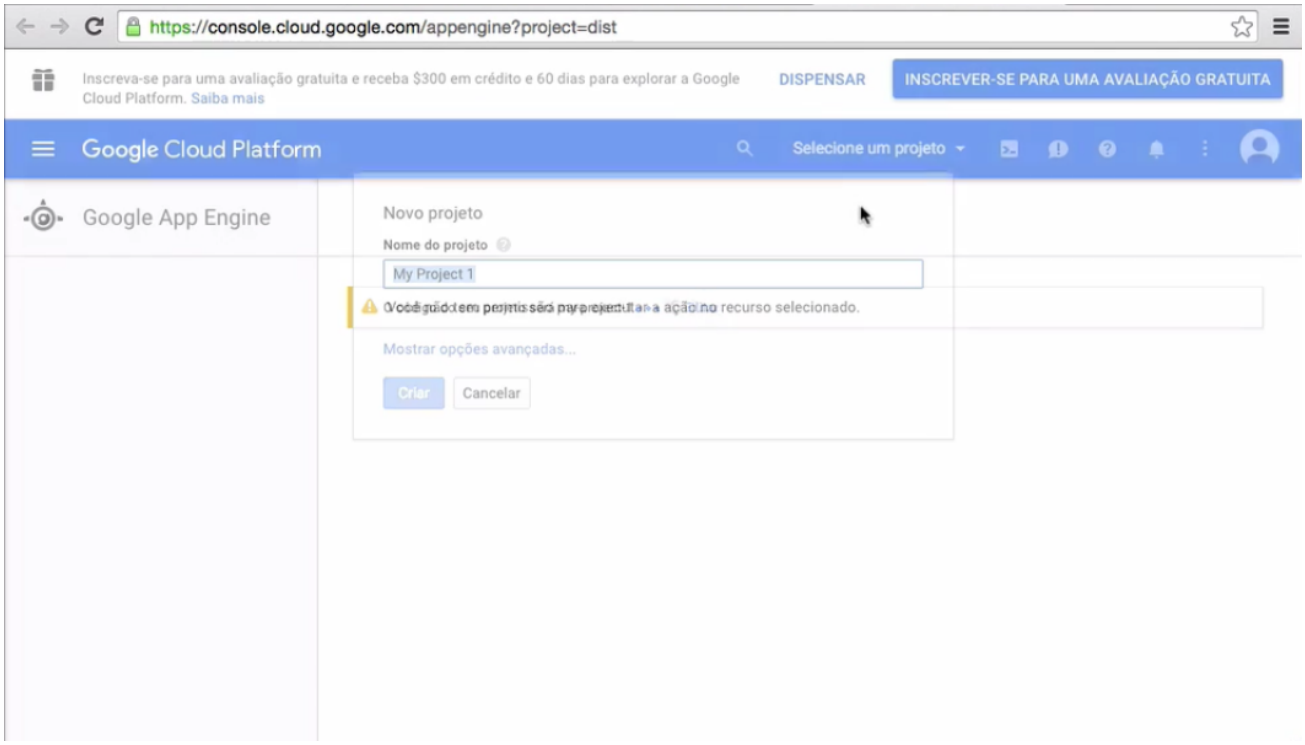
Perceba que temos diversas coisas aqui e perceba que ele está nos dizendo " `rintie:php55` ", ou seja, ele está rodando a versão *php*. Podemos adicionar no *cloud* uma "existing application", ou seja, a nossa pasta que já existe para isso seguimos "File > Browse > dist ". Escolhermos o "dist", pois já é a versão otimizada e escolhemos o "Add".



E podemos até rodar localmente, ele nos diz em que porta isso estará sendo rodado, no caso, "8080". Basta clicarmos no "*Run*".



Vamos no navegador, abrimos uma aba nova digitando "localhost:8080" e vamos ver que tudo estará funcionando ok. Mas não é isso que queremos, afinal, queremos que esteja na nuvem. Para fazer isso, primeiro, temos que criar um aplicativo nas configurações do *google app engine*. Para isso vamos no *google app engine* e clicamos em "Dashboard" e ele vai abrir um console do *google cloud*. E ao lado direito temos "Selecione um projeto", clicamos nisso e selecionamos a opção "Criar um projeto..." e, com isso, podemos criar um projeto e darmos um nome a ele:

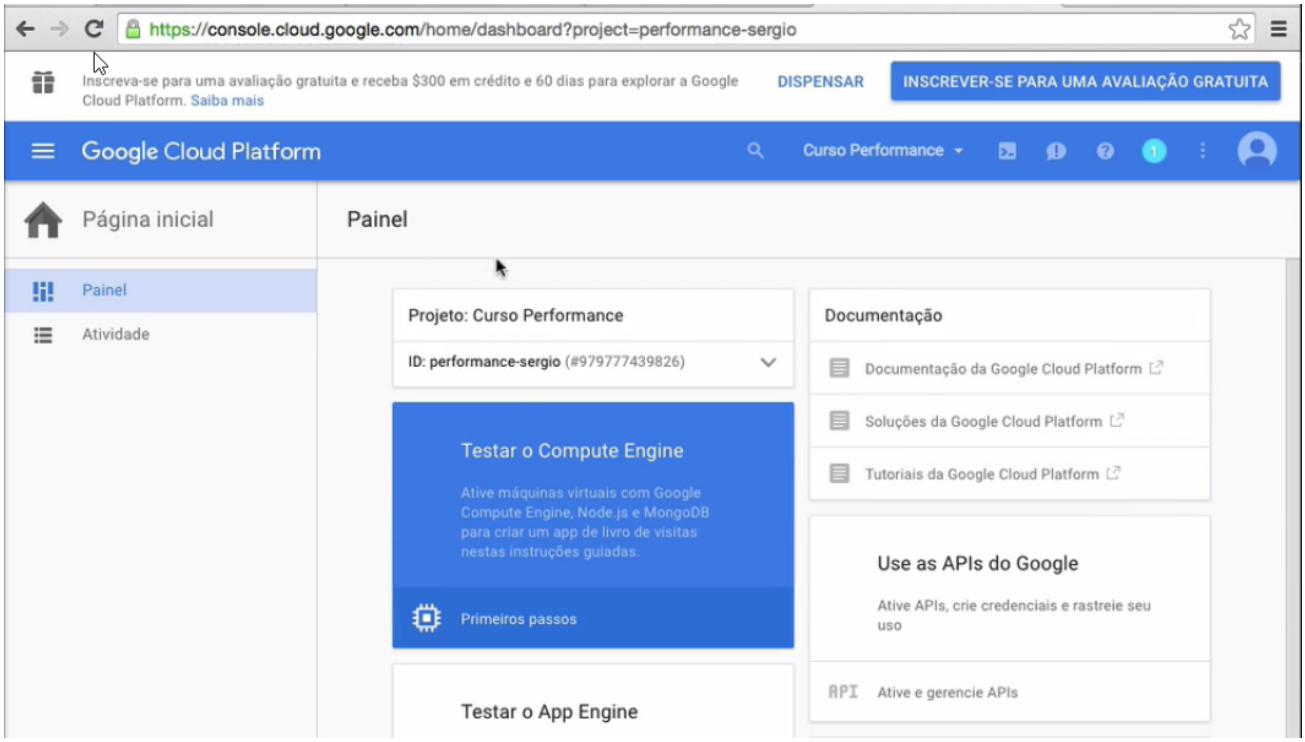


O nosso projeto se chamará "Curso Performance" e vamos inserir também um "ID" que nomearemos "performance-sergio" e selecionamos "Criar". E ele vai criar a "app" e poderemos usar ela.

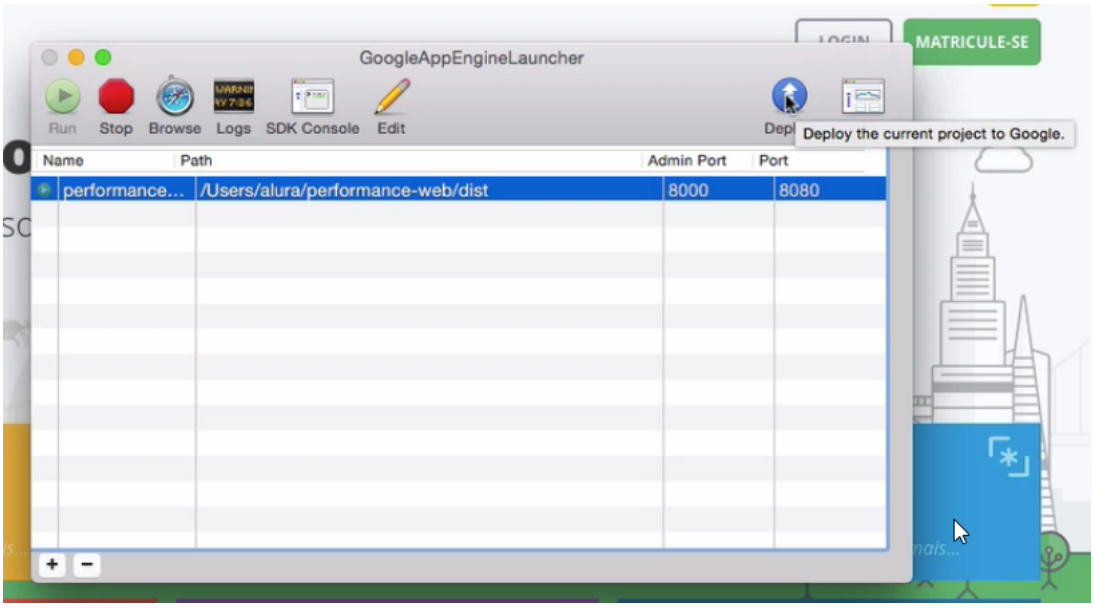
E como utilizamos isso agora?

Temos em nossas pastas o arquivo "app.yaml" e nele vamos criar uma linha nova, acima de tudo, e escreveremos `application: perfomance-sergio` . Perceba que inserimos o "id" que tínhamos criado e esse id é único.

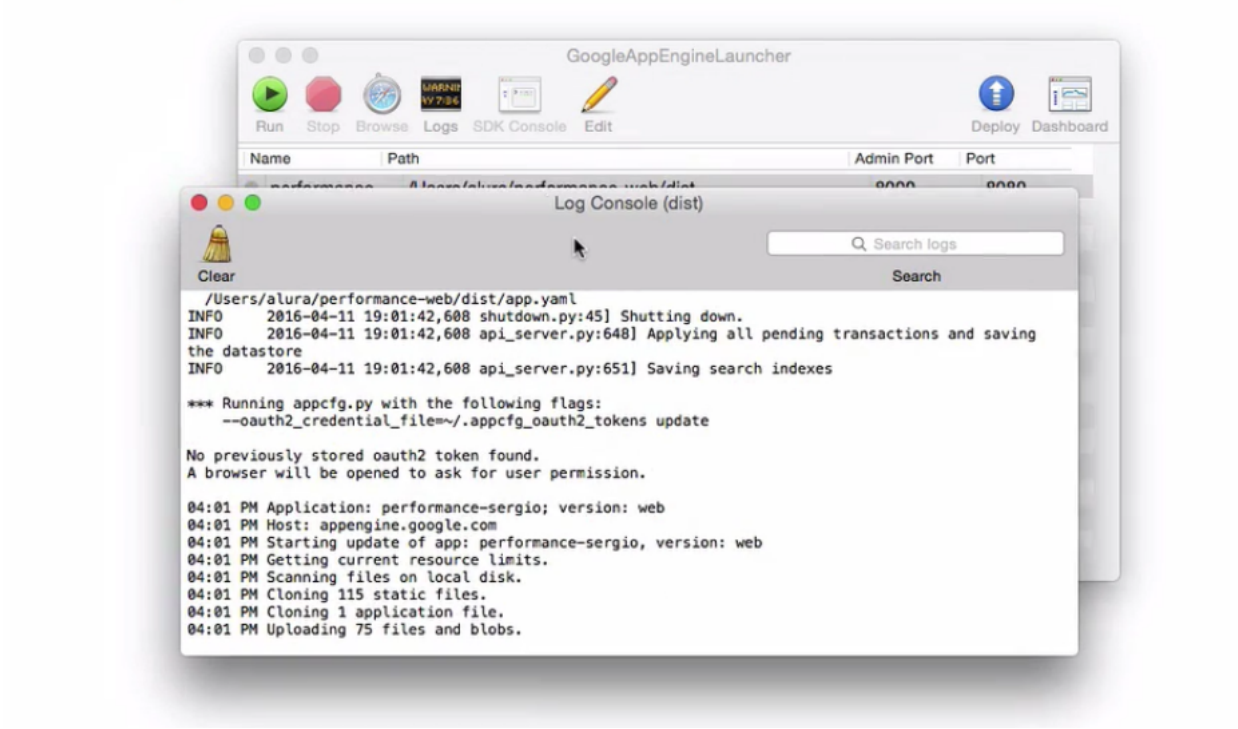
Quando fazemos isso, o navegador já cria para nós:



Agora, podemos habilitar a opção *deploy*:



Que é opção que sobe no servidor remoto do *google* esse site que ele subiu localmente.

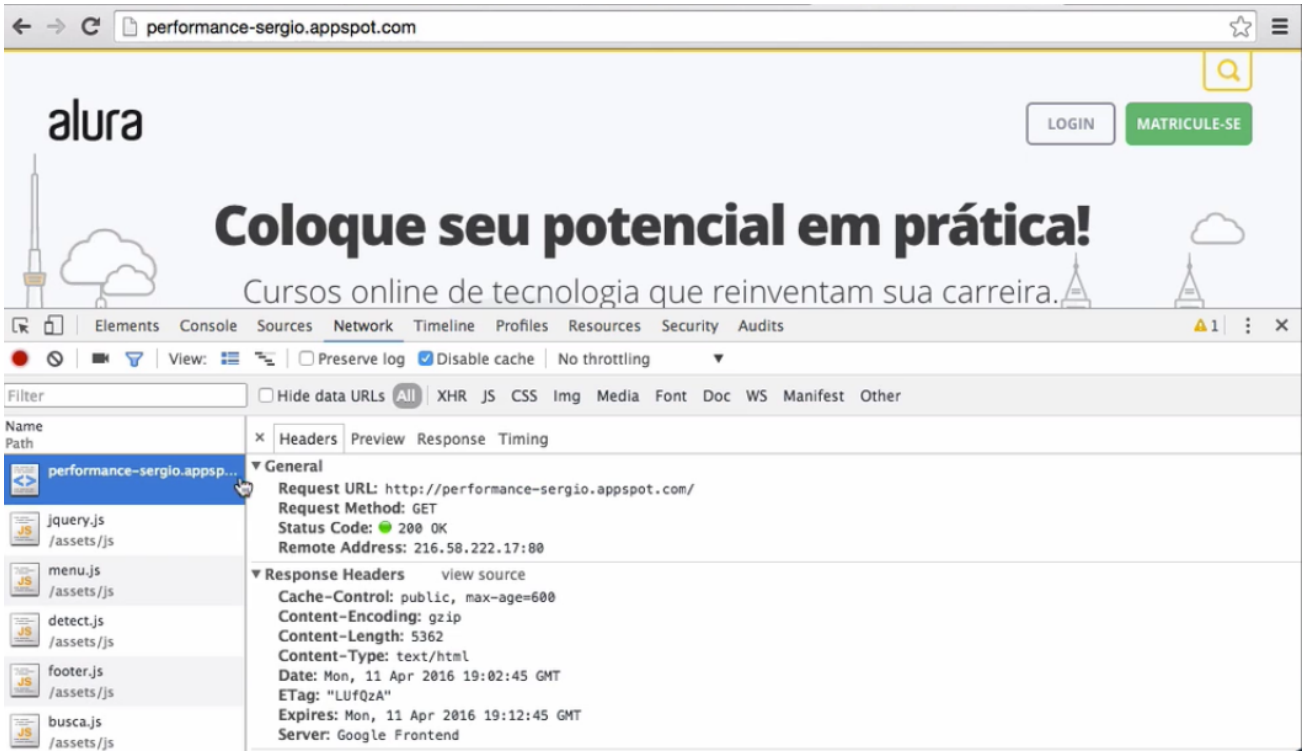


A primeira vez temos que estar logados no e-mail. E depois disso ele volta para o aplicativo. No momento que ele terminar vai nos avisar que "*Deployment successful*".

Como vamos abrir isso agora?

Vamos digitar no navegador o "*id*" que usamos seguido de "app spot", teremos "performance-sergio.appspot.com". O "appspot.com" é o site do google *app engine* onde ele expõe a "url" e dando um "Enter" teremos o nosso site rodando na nuvem, em um "url" pública.

Vamos no *dev tools* dando "F12" e podemos olhar na aba *network* , por exemplo, que o *html* está com *gzip*, podemos conferir isso clicando em seu ícone:



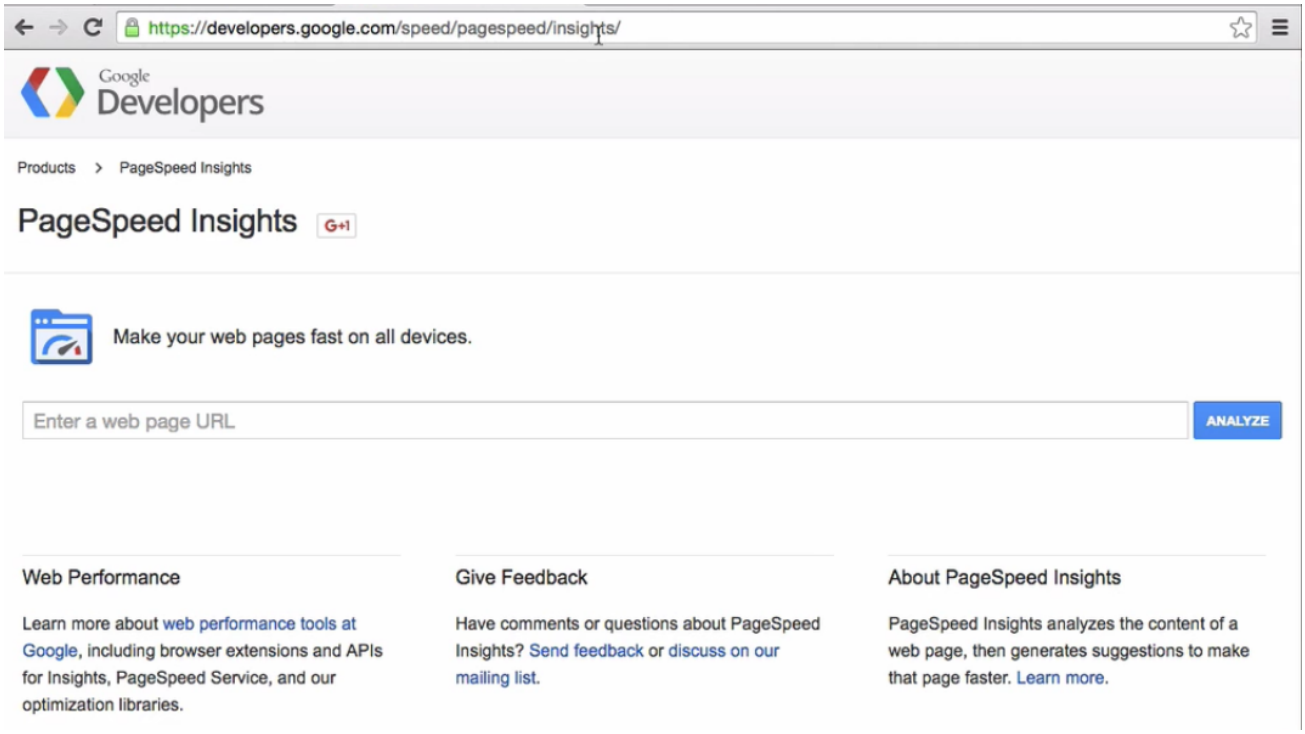
Isso é uma das coisas que o *back engine* do *google engine* já faz por nós de maneira transparente. Tínhamos visto como habilitar o *gzip* no *nginx* e que todos os servidores podem ser habilitados e o do *google* já vêm habilitado e transparente para nós.

Agora temos o site rodando remotamente! Lembrando, pode usar outro servidor, é um *html* estático e é só subir. Nós utilizamos o *app engine* pois ele é bastante simples, gratuito e tem suporte *http2*.

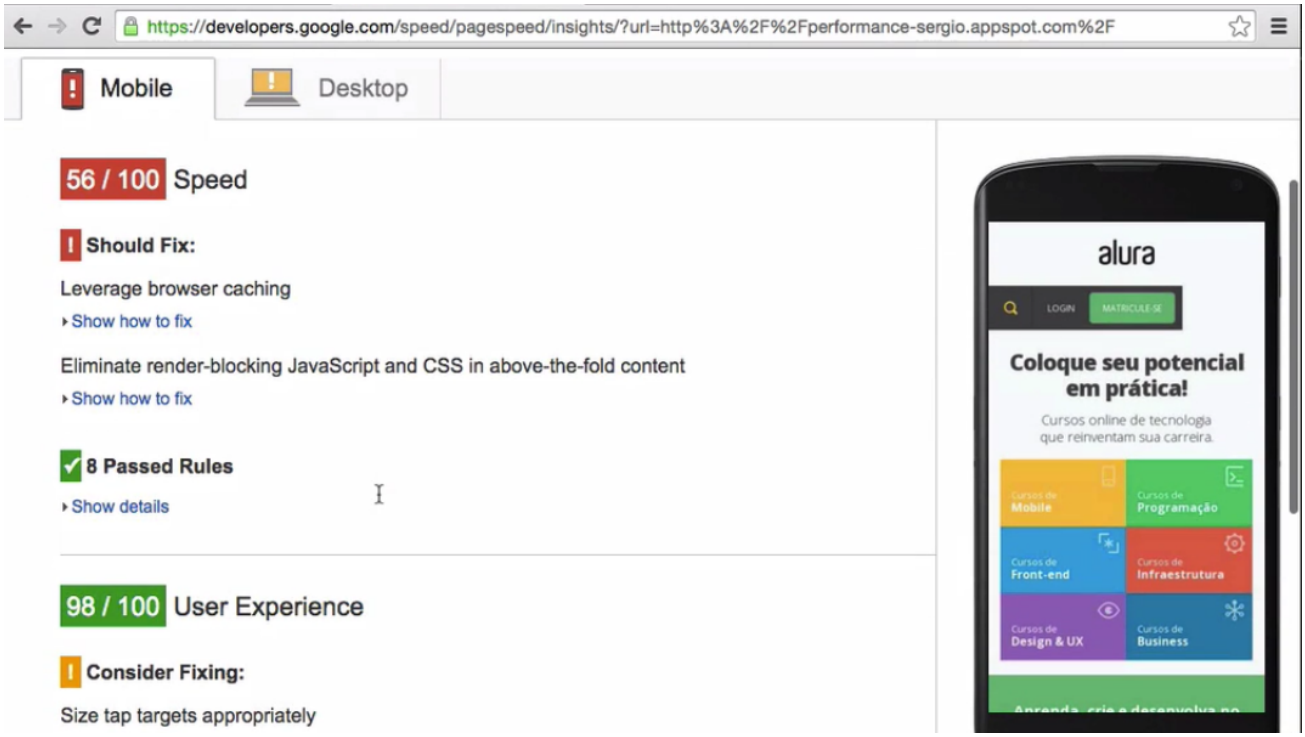
Agora que temos nosso site online em uma "url", no caso o *google app engine*, podemos usar várias ferramentas de análise de performance.

Vamos ver duas dessas ferramentas e aprender a usá-las e a ler suas informações.

A primeira é a *page speed insights*, que é uma ferramenra do *google*, que está acessível no link https://developers.google.com/speed/pagespeed/insights/ (https://developers.google.com/speed/pagespeed/insights/).

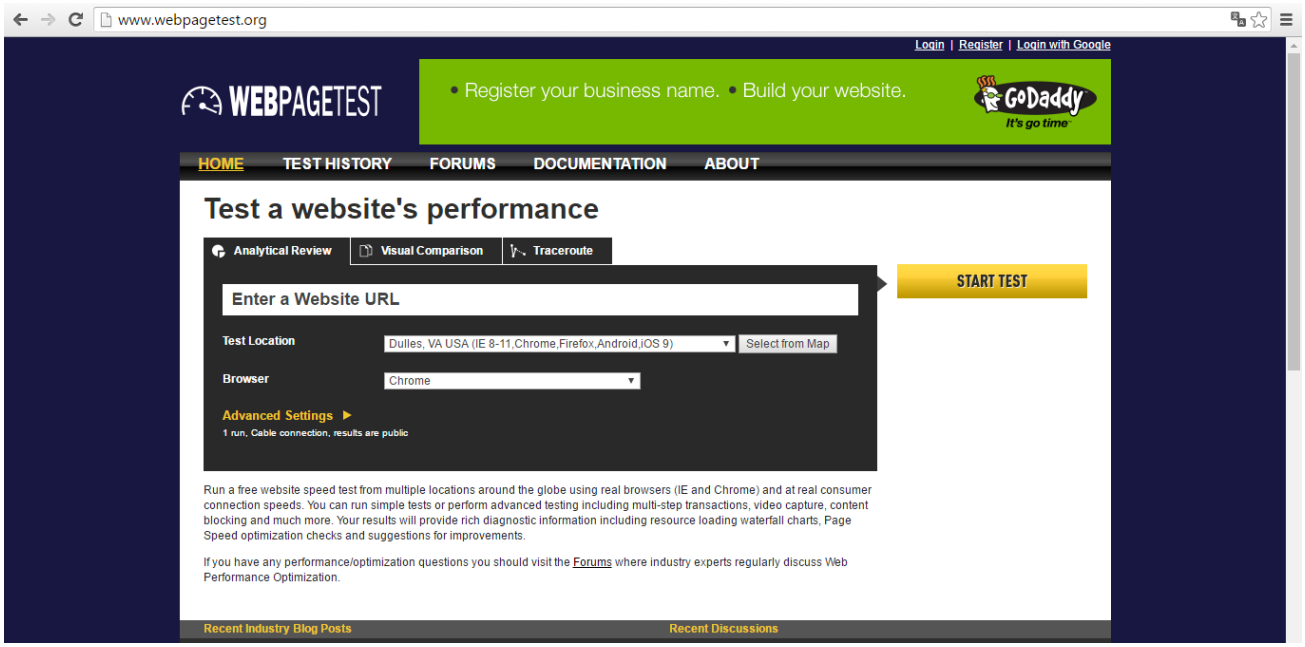


A ideia é que copiemos a "url" do nosso site e colemos no campo que aparece logo no início da página. Ele irá baixar a página e dará uma nota de 0 a 100. Repare que estamos mais ou menos na média.



Ele também fornece diversas regras apontando o que foi feito bem e o que ainda necessita melhorar. Inclusive, alguns dos erros apontados iremos rever mais adiante. Repare que diversas melhoras nós já realizamos, como, por exemplo, comprimir o *gzip*, minificações, otimizações de imagens e etc. E, ainda, são notas diferentes para *Mobile* e *Desktop*.

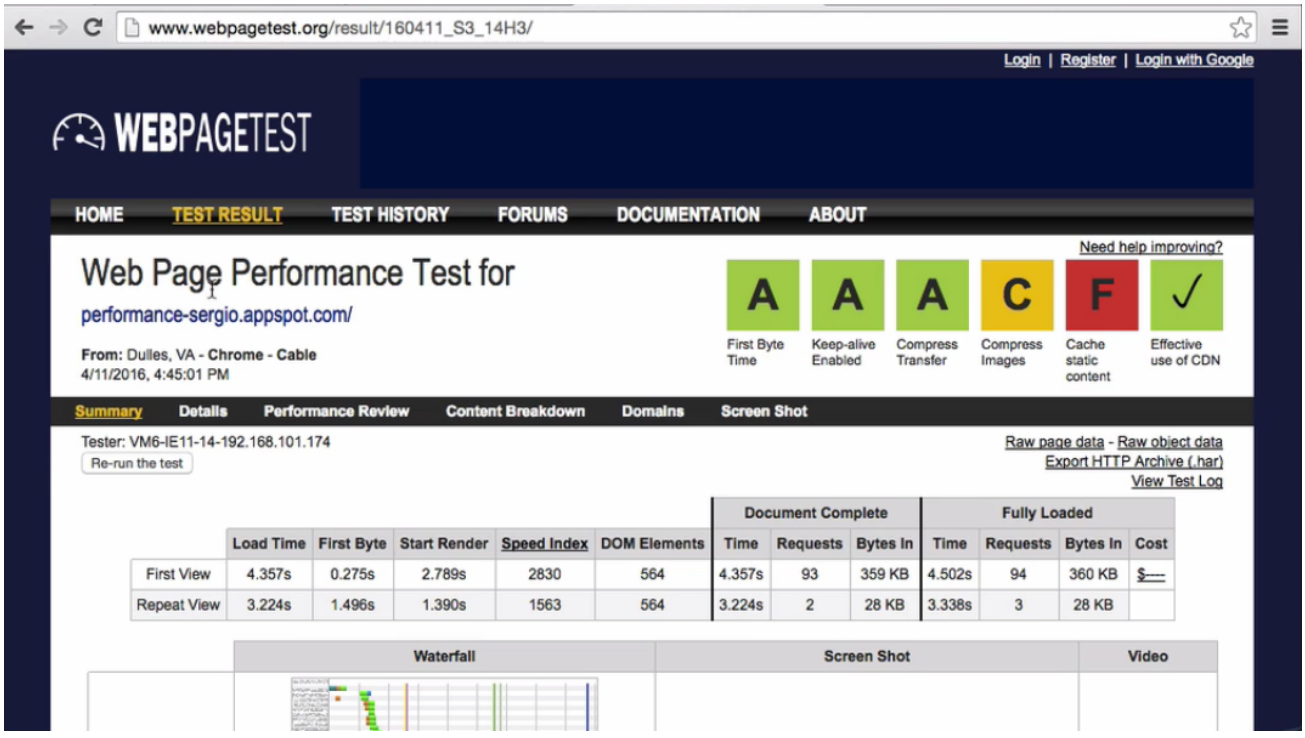
Outra ferramenta um pouco mais completa é a *Web page test*. Essa ferramenta se encontra no site http://www.webpagetest.org/ (http://www.webpagetest.org/):



Essa ferramenta possui, mais ou menos, o mesmo princípio. Nós copiamos e colamos o endereço do site que queremos testar e, ainda, podemos escolher qual a localização e qual o navegador que ele utiliza, nós vamos deixar o "padrão" para ambos. Podemos escolher diversas outras configurações, não abordaremos todas, mas uma especial vale a pena analisarmos com mais carinho. A *capture vídeo*, que por padrão vem desmarcada, então, nós iremos marcar essa opção e ele nós mostrará uma simulação de vídeo.

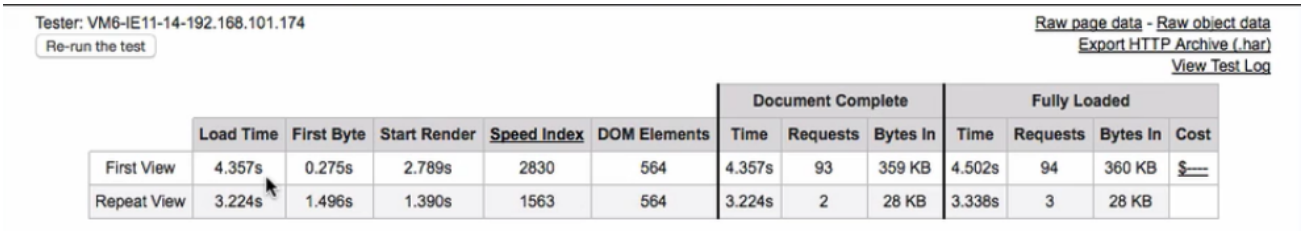
Conforme mencionado, existem diversas possibilidades, então, sintam-se a vontade para mexer e explorar bastante. Você pode, inclusive, simular uma rede *Mobile*. Após selecionar as opções que queremos clicamos em *Start Test* que ocorrerá uma simulação nos servidores da *web page test* e nos informará algumas métricas interessantes.

Teremos o seguinte:



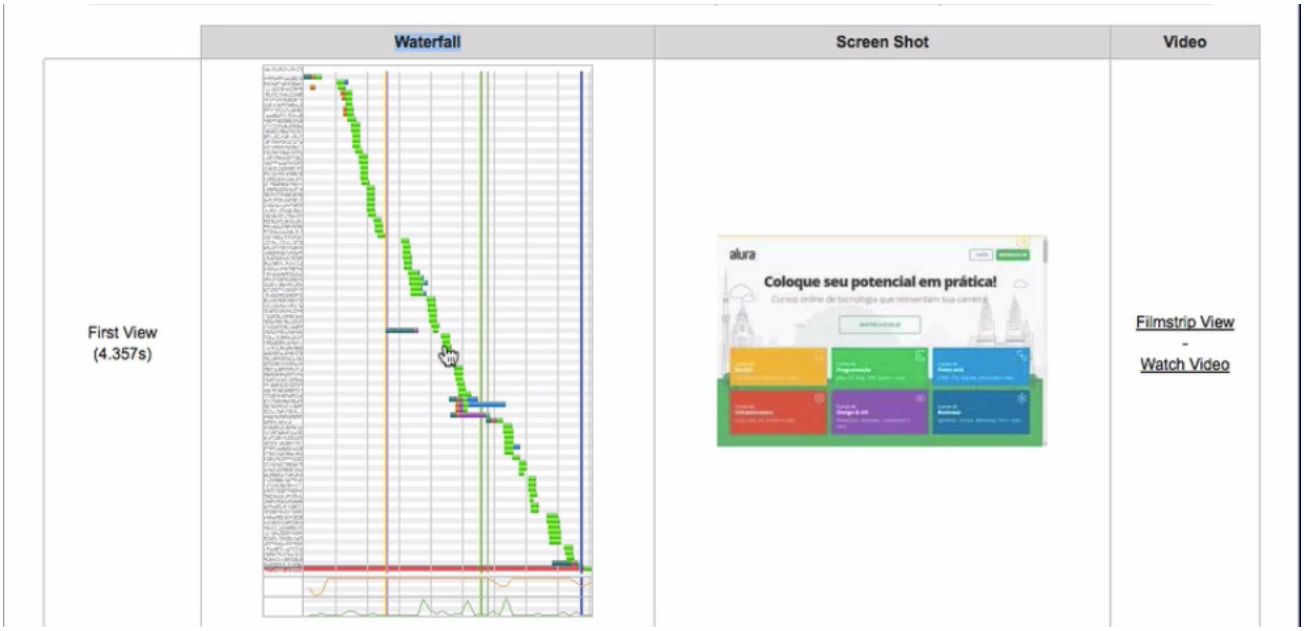
Temos diversas informações a serem analisadas, logo na parte superior temos diversas notas referentes a alguns quesitos.

Ele informa alguns números referentes a performance de carregamento do site. Alguns números mais interessantes e outros menos. Mas, por exemplo, ele nos fala que o site tem um tempo de carregamento total de 4,3 segundos. Ele fala que fizemos 93 *requests* em "359 KB" e informa, ainda a quantidade de *tags html* no site e a ideia é que quanto menos *tags html*, mais rápido é o site. Temos diversas informações, algumas trabalharemos com mais afinco mais adiante.



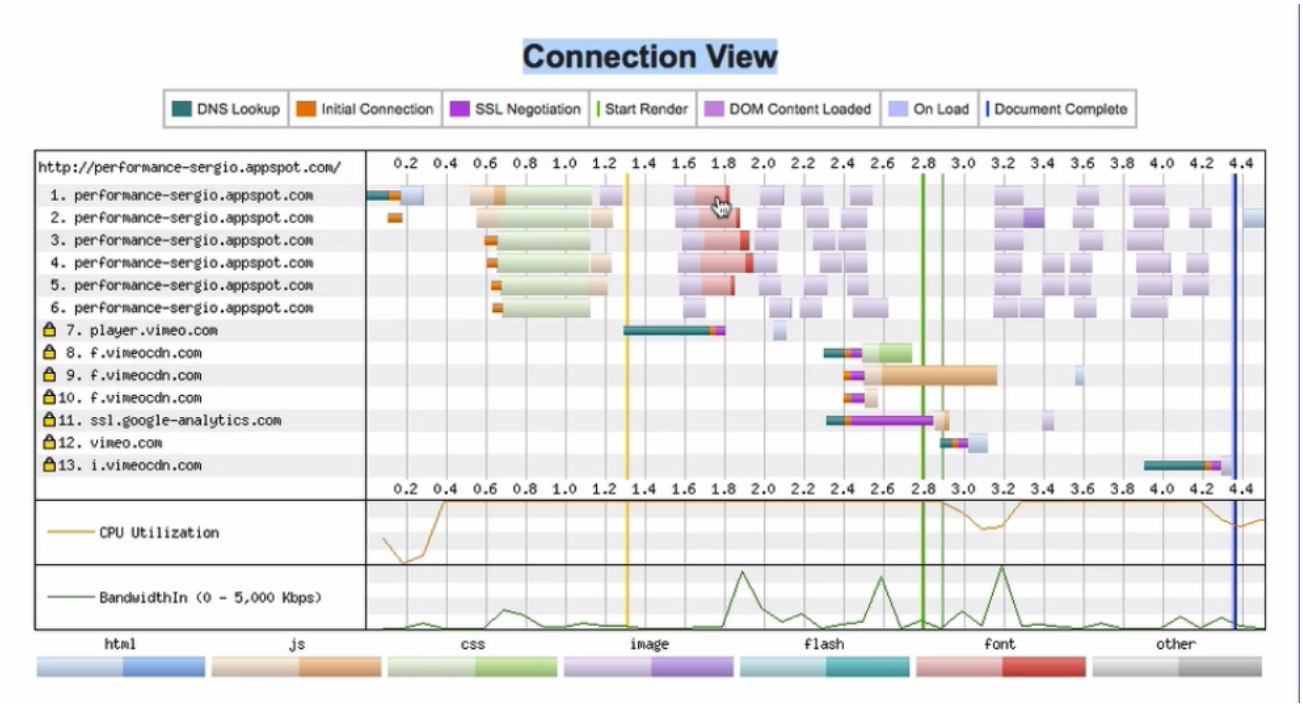
	Load Time	First Byte	Start Render	Speed Index	DOM Elements	Document Complete Time	Document Complete Requests	Document Complete Bytes In	Fully Loaded Time	Fully Loaded Requests	Fully Loaded Bytes In	Cost
First View	4.357s	0.275s	2.789s	2830	564	4.357s	93	359 KB	4.502s	94	360 KB	$----
Repeat View	3.224s	1.496s	1.390s	1563	564	3.224s	2	28 KB	3.338s	3	28 KB	

Se rolarmos a barra mais para baixo teremos informações acerca do gráfico *waterfall*

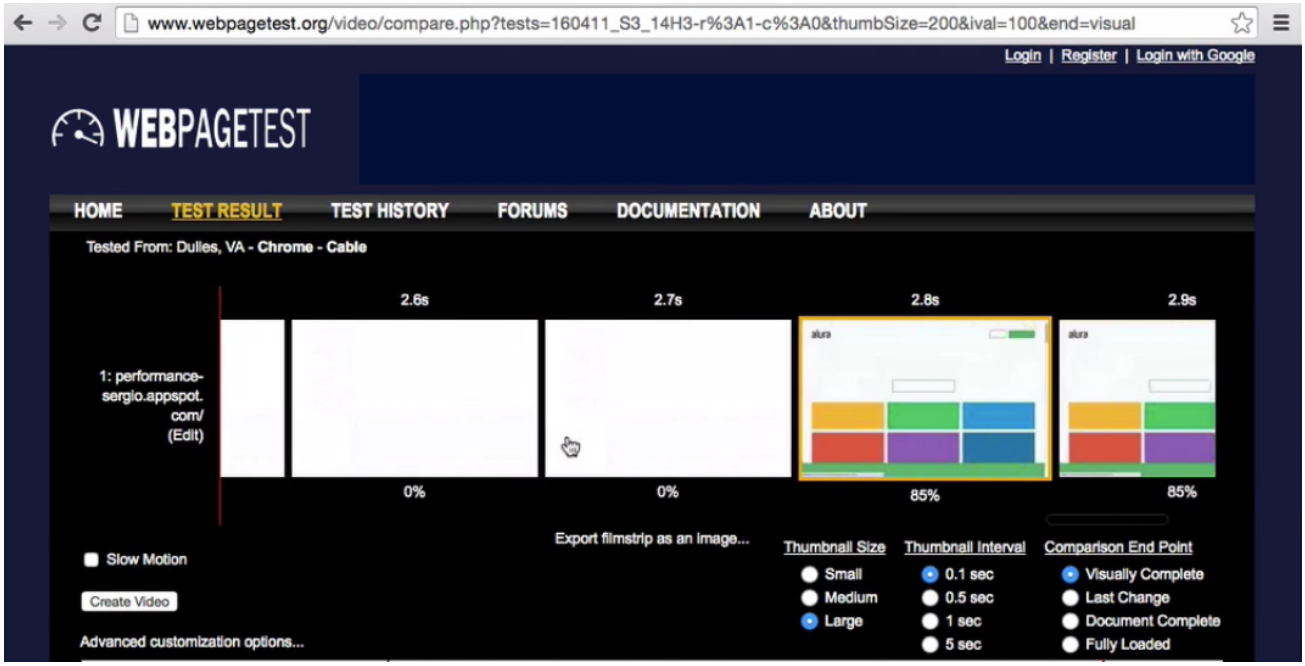


Clicando nesse gráfico podemos visualizar seus detalhes. Ele segue a mesma ideia do que tínhamos visto no *dev tools* do *Chrome*. Podemos analisar como o tempo foi gasto e as cores indicam as diferenças de como o tempo foi utilizado, o verde significa o tempo para o primeiro byte e o azul é o tempo de download. São fornecidas, também, informações a cerca do tempo para conexão inicial e etc. Podemos enxergar nesse gráfico como as coisas são baixadas de seis em seis *requests*:

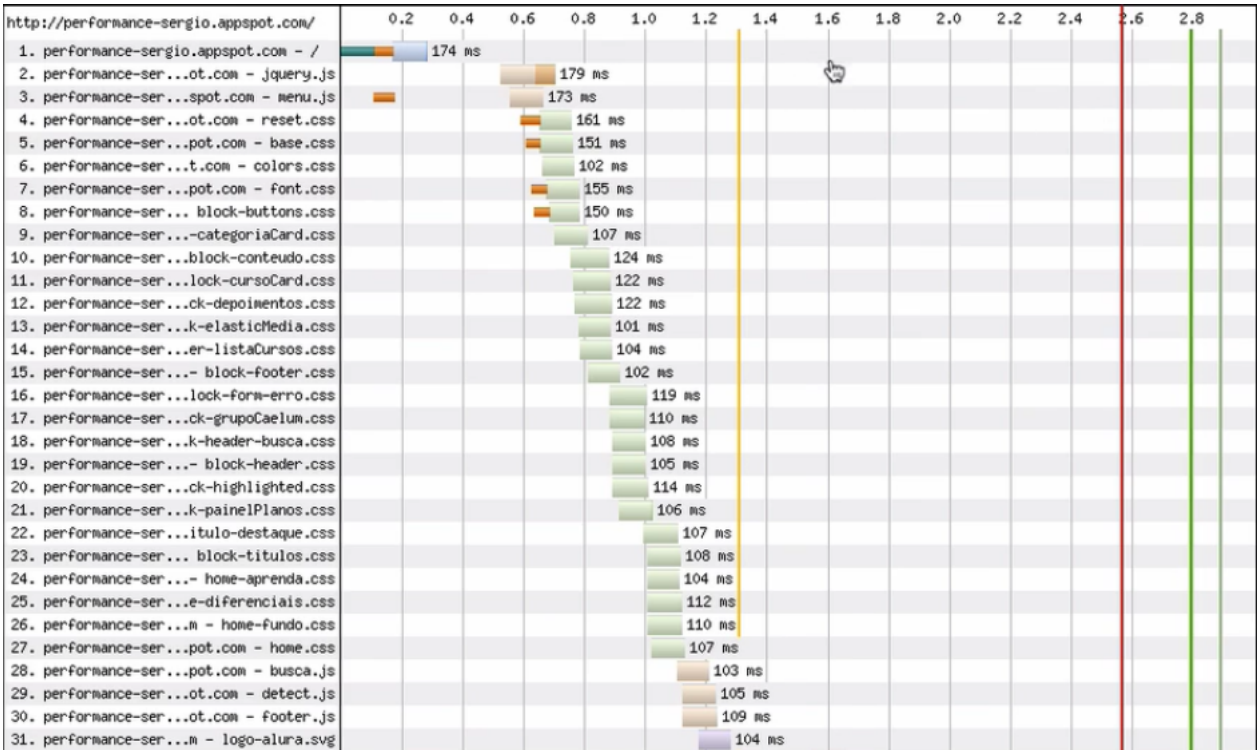
Podemos rolar a barra um pouco mais e poderemos ver a *Connection View* que é um gráfico do ponto de vista das conexões e podemos ver o que foi baixado em cada em conexão. No começo ele baixa as requisições de *css* que estão em verde, as fontes e por último as imagens.



Se voltarmos ao resultado inicial, temos a opção de *film strip view*, pois selecionamos, no início que queríamos visualizar o vídeo. Podemos clicar na opção *filmstrip view* e ele nos mostrará um gráfico de acordo com o tempo e como foi o carregamento da página. Podemos, inclusive, alterar o tempo:

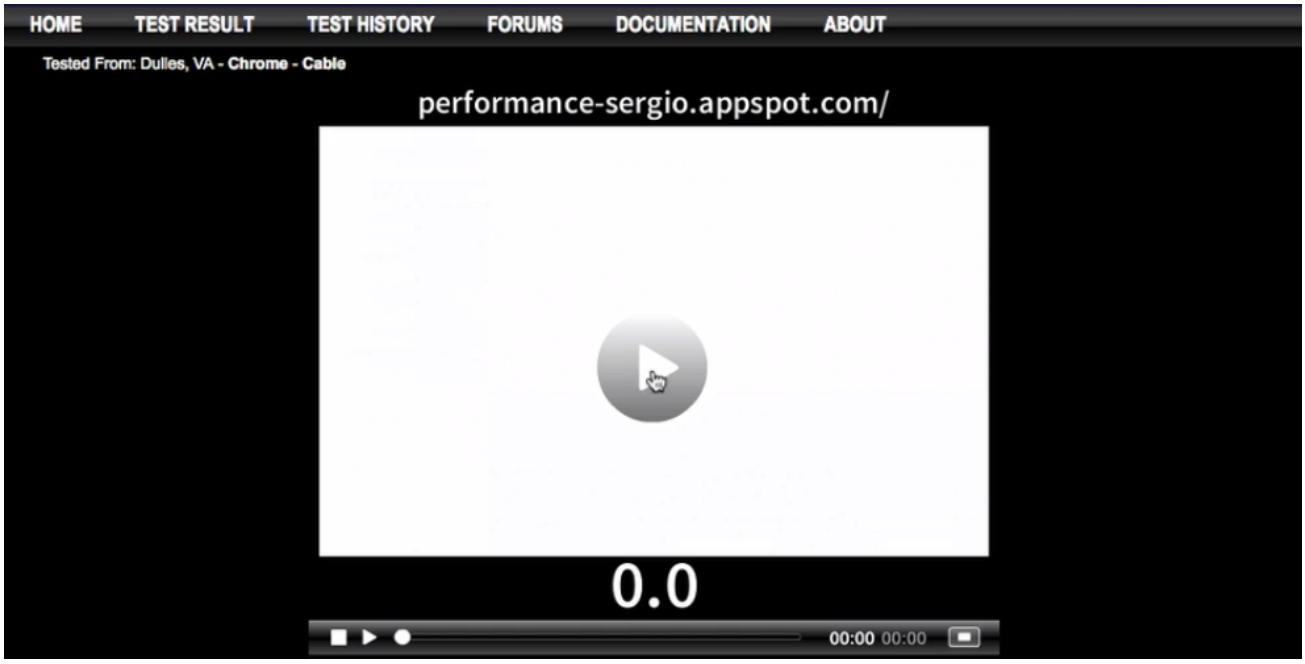


E o que vemos é um pouco assustador, na verdade, desde que se inicia o carregamento até 2.7 segundos, o usuário ainda não conseguiu visualizar nada. Durante esse tempo sabemos que coisas foram carregadas, para comprovar isso basta olhar o gráfico *waterfall* que está logo abaixo na mesma página:



O tempo em que fica em branco temos diversas coisas que foram carregadas, mas nada foi exibido na página. Discutiremos isso mais adiante. Um dos aspectos importantes é que buscaremos priorizar os *downloads* para que o tempo de carregamento, de renderização, diminua. Assim, ele começa a renderizar progressivamente o site antes.

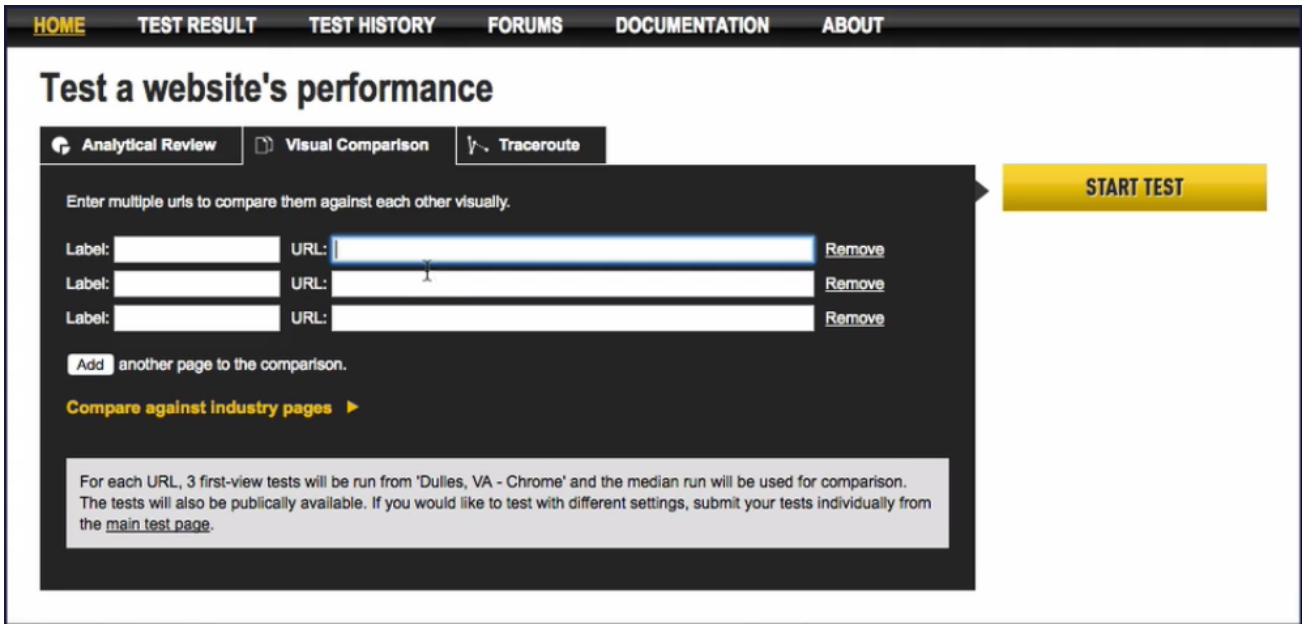
Por enquanto, perceba que em 2.7 segundos temos um início da renderização, mas ele só legível a partir dos 3 segundos. Será que isso é um tempo bom? Vamos discutir isso também na sequência. Temos a opção de clicar no botão *Create video* e ele irá gerar um vídeo:



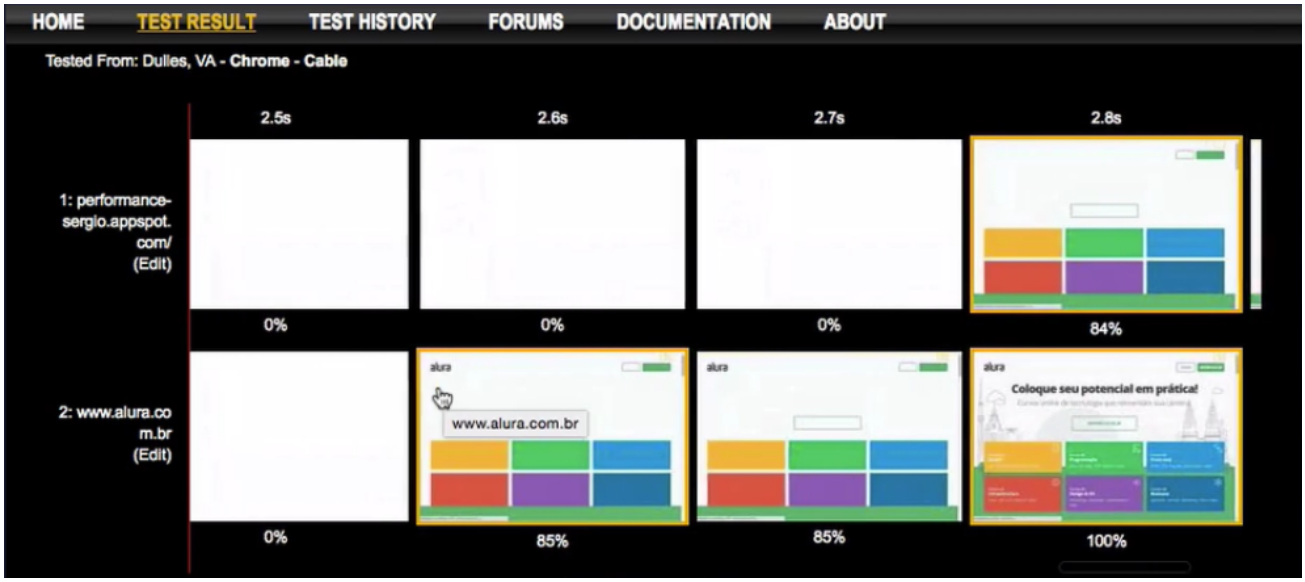
Isso é bacana para mostrar resultados de maneira mais visual e simples e além disso para alguns tipos de situação, por exemplo, se você quiser convencer seu chefe de que é necessário realizar uma otimização.

Temos diversas opções no *web Page Performance Test* e podemos passar um longo tempo investigando todos esses aspectos e informações.

Se voltarmos na *home* podemos ir na aba *Visual Comparison* onde podemos introduzir diferentes *url* e podemos fazer uma comparação entre diferentes sites.



Podemos testar a performance do site real do *alura* com o que estamos trabalhando, para isso, basta escrever nos campos as *urls* de ambos os site. Teremos um comparativo visual entre os dois sites. Inclusive, teremos dois vídeos onde podemos analisar as diferenças entre os sites. Teremos o seguinte:



Ele mostra as diferenças que temos.

Usaremos bastante essas ferramentas para analisar nossa performance, mas também para analisar as boas práticas que podemos utilizar para melhorar o site.