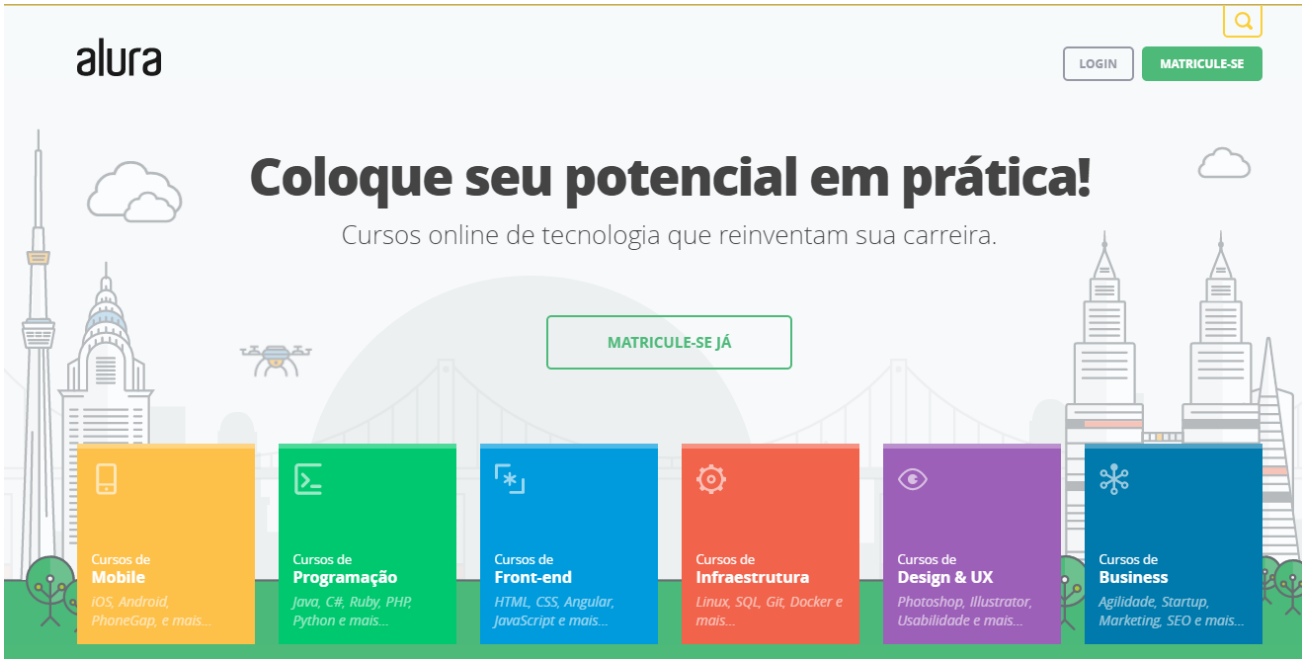
16

Transcrição das aulas

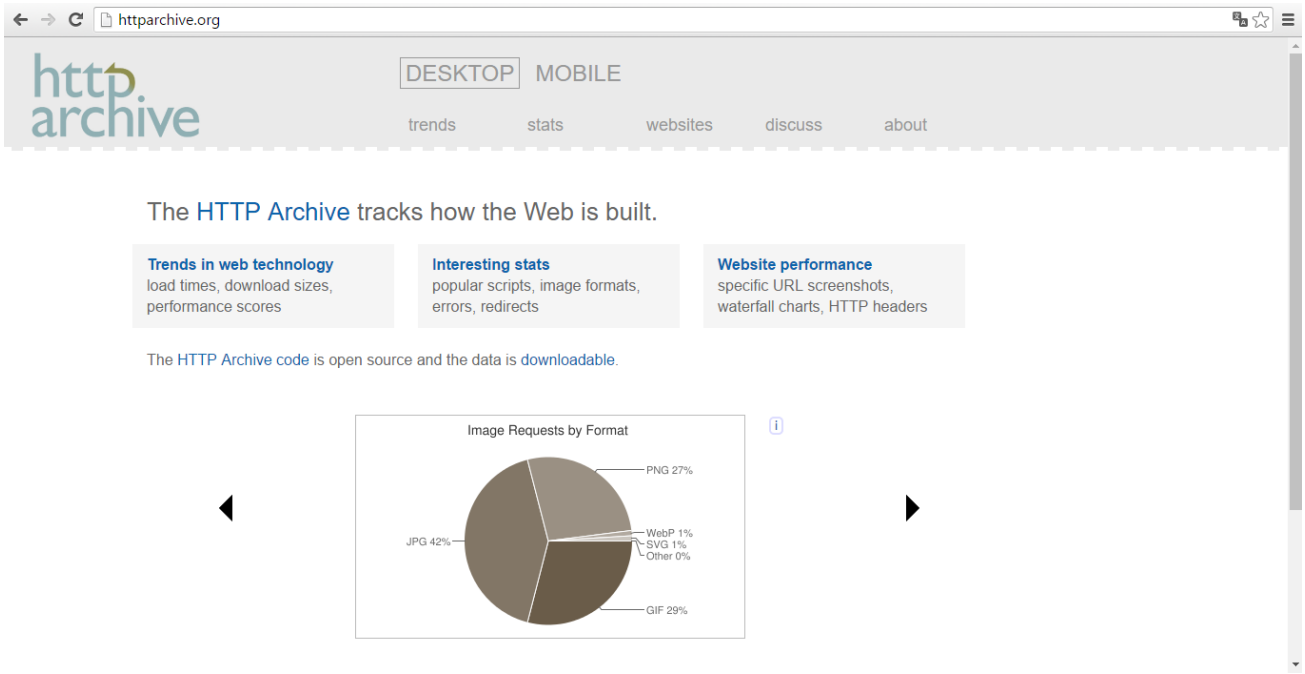
Tem algo que ainda não falamos e que tem um peso bastante grande, são as imagens.

Observe o site que estamos lidando, que é o site do *Alura*:

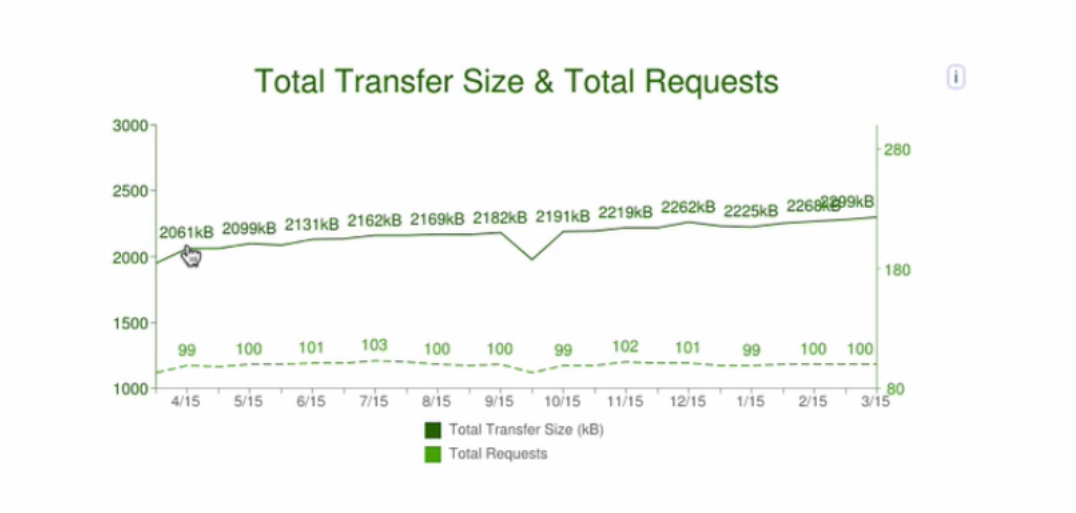


As imagens são bastante importantes no quesito performance.

Temos um site bastante interessante que é o httparchive.org, esse link é um índice, onde criaram um robo que indexa diversas páginas e retira delas estatísticas bastante interessantes, geralmente, em relação a questões de performance.

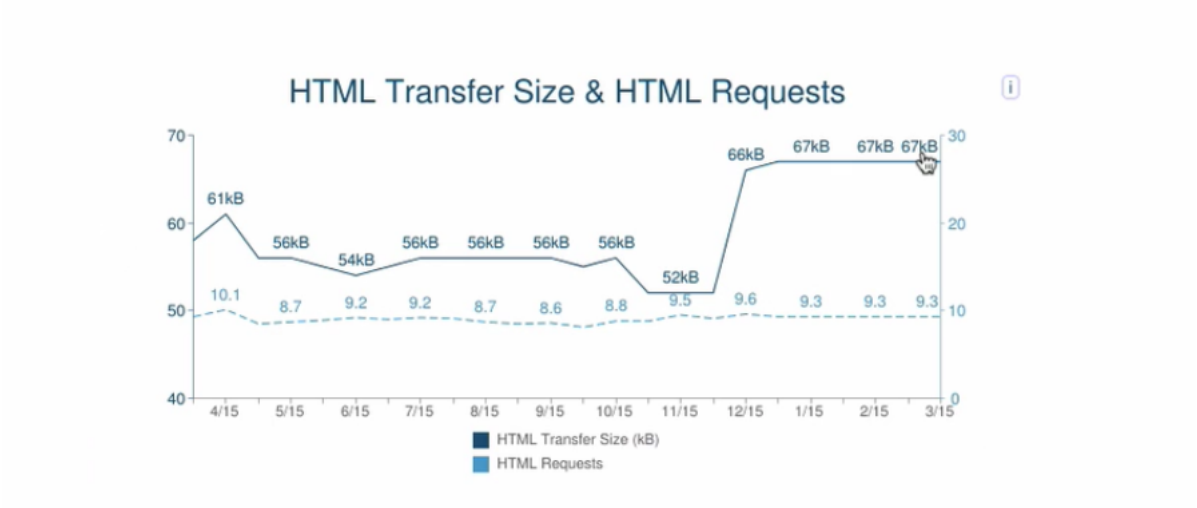


Se clicarmos em *trends* ele menciona a quantidade de páginas que estão sendo analisadas e mostra por exemplo, qual a média do tamanho das páginas:



Essa média aumenta em relação ao tempo, como podemos observar, hoje, a média é cerca de mais de 2 Megas, o que é bastante. O ideal é que a página esteja bem abaixo dessa média.

É interressante que ele mostra, também, outros detalhes, por exemplo, o tamanho médio da página *html*. A porcentagem de *html* que existe nesses mais de 2 Megas representa aproximadamente 60%. Compreensível, pois, o *html* é um arquivo texto, é "gzipavél" e não é tão pesado.

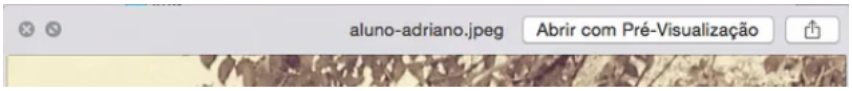


Existem vários dados bastante interessante nessa página que você pode explorar a vontade.

Vamos nos debruçar nos dados de imagens, as imagens ocupam em geral 2/3 do total dos mais de 2 megas. Imagine, isso é muita coisa! Se conseguirmos melhorar o cenário das imagens, o impacto que isso tem na performance é algo gogantesco.

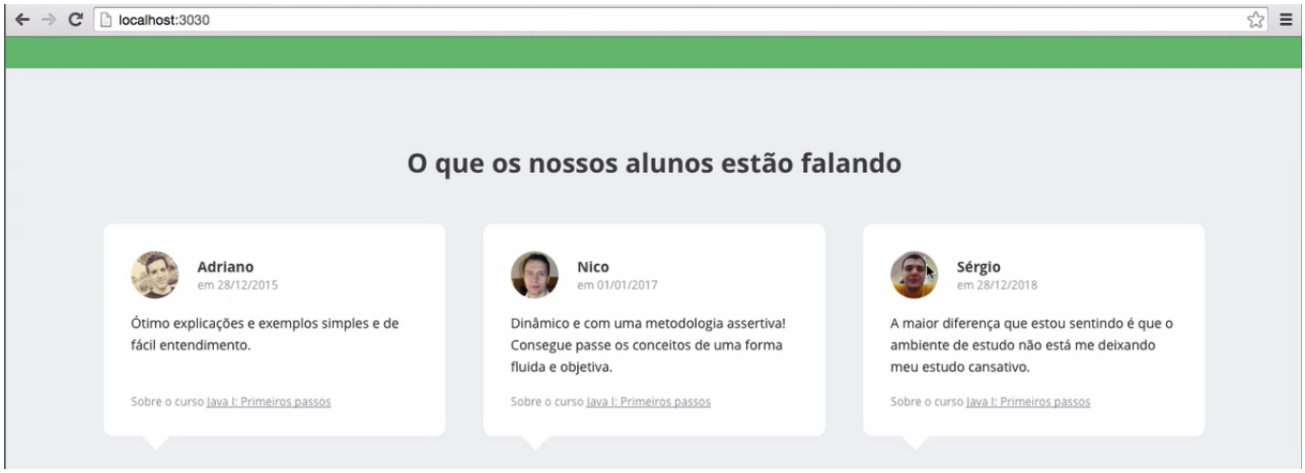
E falaremos disso nesse momento!

Vamos pegar o nosso projeto e abrir a sua pasta e seguimos "Site > assets > img". Dentro da pasta "img" temos diveras imagens. Esse site em particular tem diversos tipos de imagens de propósito, para que, justamente, estudemos diferentes tipo de otimização.



Temos imagens em *jpeg*, *svg*, muito utilizado em sites que possuem diversos recursos gráficos, *png*. Só não existem *gifs*, que por sinal, esse formato hoje é bastante desnecessário, ele é ruim do ponto de vista dos formatos modernos. A única razão para seguir utilizando o formato *gif* é para animações, mas, atualmente, o *css* possui inclusive uma qualidade melhor.

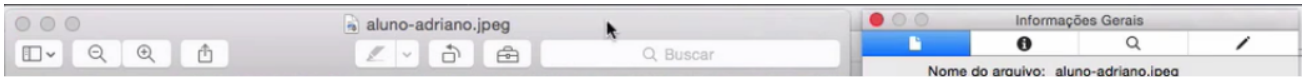
Vamos começar pelo *jpeg*. Observe, temos algumas fotos, meramente ilustrativas, de "alunos" que deixaram depoimentos:



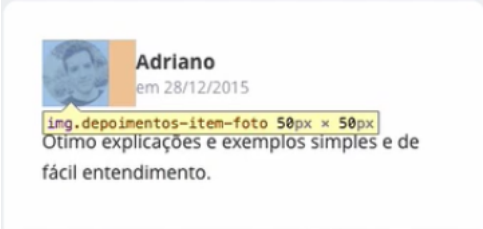
Observe que as fotos parecem bastante pequenas em comparação ao que realmente são. Observe a foto que tínhamos mostrado mais acima.

É importante colocar a imagem do tamanho correto do que vai ser utilizado na página, se não, você estará desperdiçando espaço.

Vamos abrir a primeira foto, ela possui "206 KB", vamos abrir essa foto com a visualização do *Mac* e vamos mostrar no *inspector*, " Ferramentas > Mostra Inspetor" o tamanho dela em pixels. Essa imagem possui 1000 pixels x 1000 pixels.



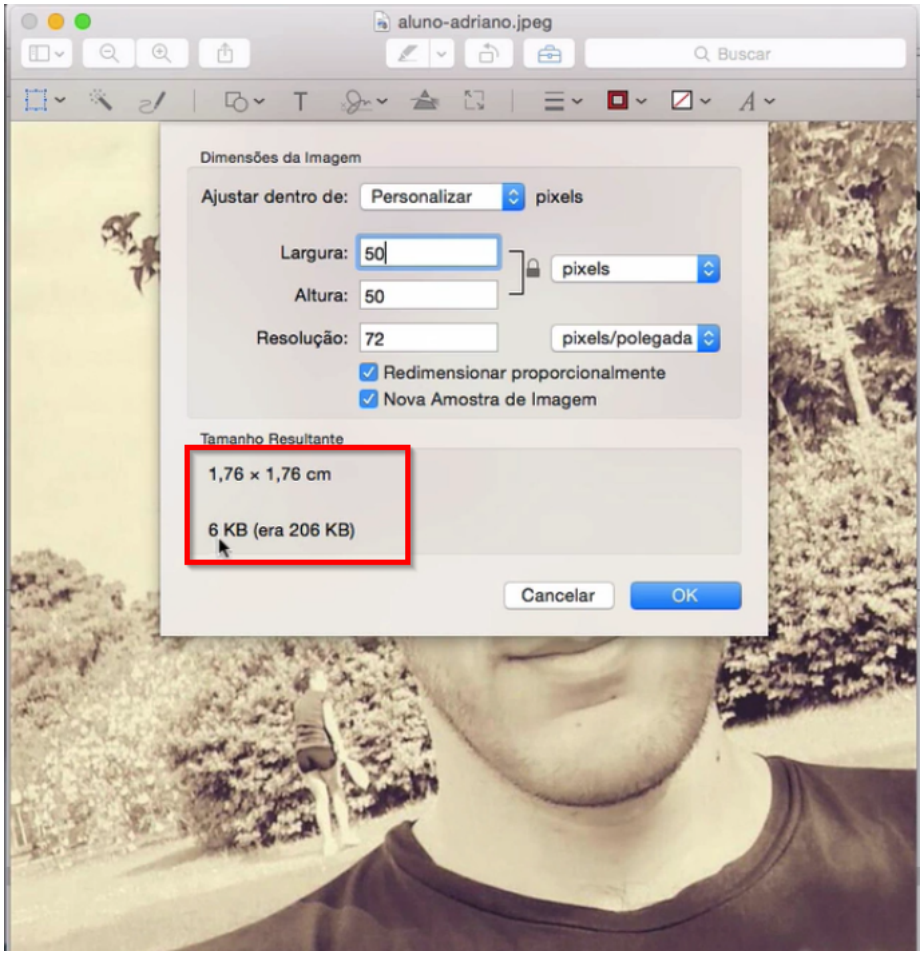
Mas, observe, não estamos utilizando tudo isso na página. Inclusive, se dermos um *Inspect* na imagem do site veremos que ela possui 50 pixels x 50 pixels.



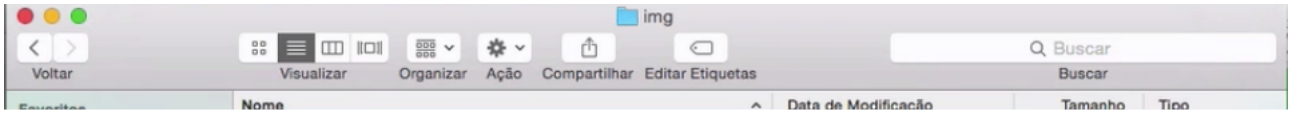
Repare o quão desnecessário é carregar um arquivo de 1000 pixels x 1000 pixels.

O que faremos?

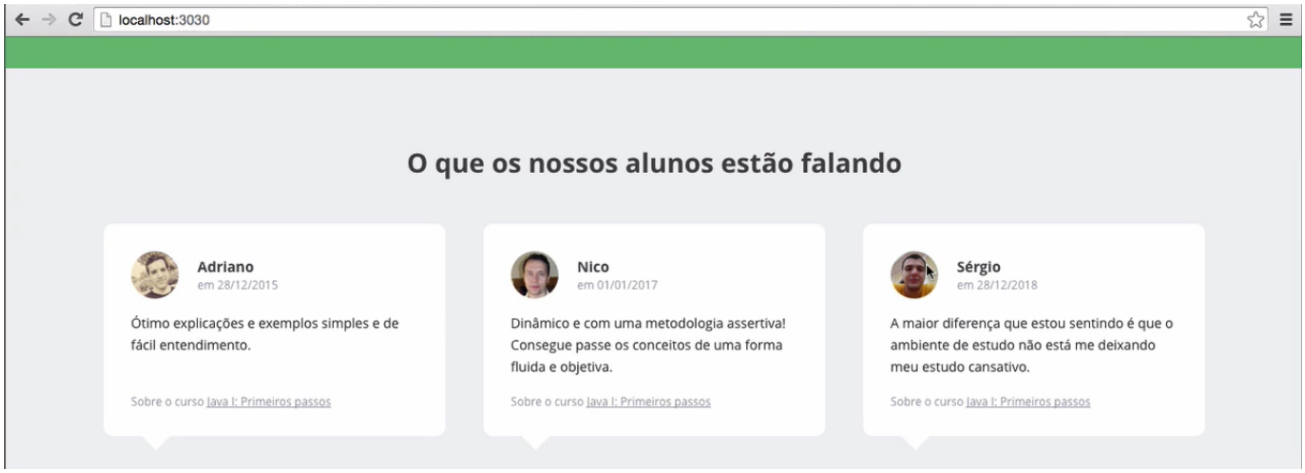
Vamos redimensionar essa imagem. Você pode utilizar o editor que quiser para fazer isso. Nós utilizaremos o que já vêm embutido no *Mac* e que é bastante simples. Diminuiremos para 50 pixels x 50 pixels, para o que é necessário. E, inclusive, repare que o editor nos menciona a economia que fizemos, passando dos "106 KB" anteriores para "6 KB".



Salvamos a imagem e atualizamos o navegador. Nem conseguimos reparar alguma diferença, uma vez que, a imagem está no tamanho adequado. Faremos o mesmo com as outras duas imagens. Agora, temos as três imagens com os tamanhos corretos:

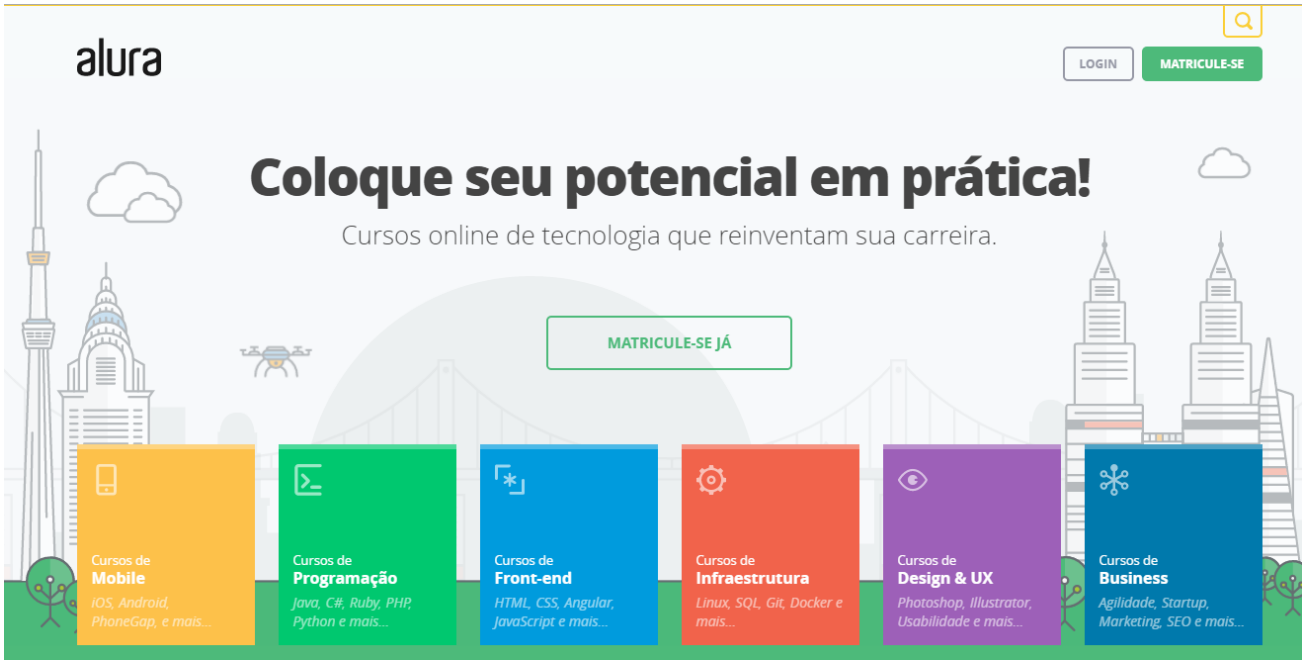


E se atualizamos o site veremos tudo no tamanho correto.



Nós editamos essas imagens direto no site, mas, se essa for a imagem original é preciso que se guarde um *backup* dela, caso, algum dia, seja preciso reutilizar a imagem no tamanho original.

Isso não vale para tudo, por exemplo, os *svj*, que são imagens vetoriais e representam os ícones e desenhos do site. São todos os desenhos e logos que temos no fundo do site. Repare:



O *svj* é vetorial e, portanto, independente de tamanho, da resolução. Não faz sentido que redirecionemos o *svj* para o tamanho correto. O que podemos redirecionar são as imagens de tipo *bitmap* que são o *jpeg*, o *png*, o *gif* e etc.

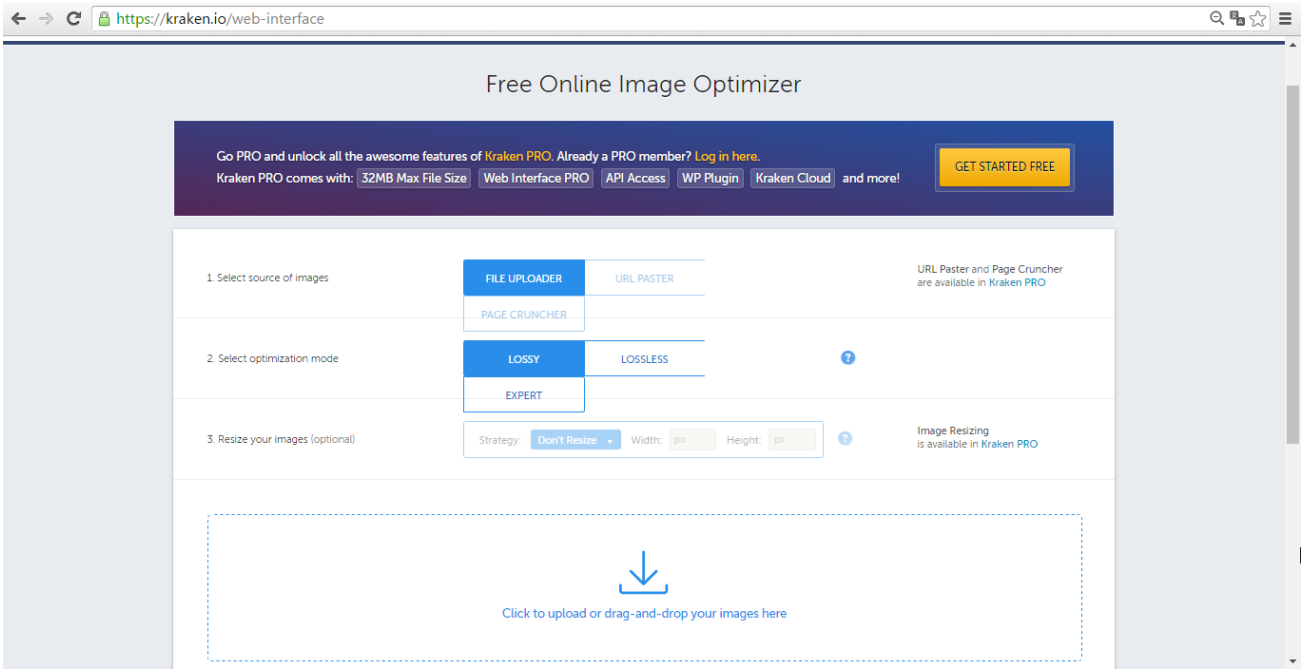
Vamos passar por mais um ponto, quando usamos câmeras digitais, normalmente, salvamos as fotos em *jpeg* e jogamos em algum *flickr* ou abrimos no computador. Essa foto possui uma série de informações adicionais além dela mesma, por exemplo, o horário, o modelo da câmera, o tamanho da abertura da lente, a orientação se for tirada pelo celular e etc. Isso é bacana para organizarmos nossos álbuns digitais e esse tipo de informação é chamada de *metadados*. Esses *metadados* são necessários em alguns momentos, mas totalmente desnecessários quando estamos falando de *web performance*. Pois, o que queremos é simplesmente mostrar a foto no site, não queremos que ninguém saiba essas informações adicionais e se deixarmos elas embutidas na imagem do site o que acontece é que ao acessar o site isso é algo a mais para ser baixado e algo a mais para pesar.

Vários dos formatos de arquivos possuem dentro do arquivo informações que são irrelevantes para desenhar na tela, então, é quase como se baixassemos ao em vez de um png, dois, o segundo sendo a miniatura do primeiro.

Então, o que podemos fazer?

Podemos otimizar essas imagens e existem vários sites na *web* que fazem esse tipo de *otimização*.

Vamos acessar o *kraken.io*, (http://kraken.io/web-interface)http://kraken.io/web-interface (http://kraken.io/web-interface), onde podemos enviar as nossas imagens e ele faz a otimização, e, ainda, mostra o que significaria pegar esse arquivo e remover as coisas desnecessárias.



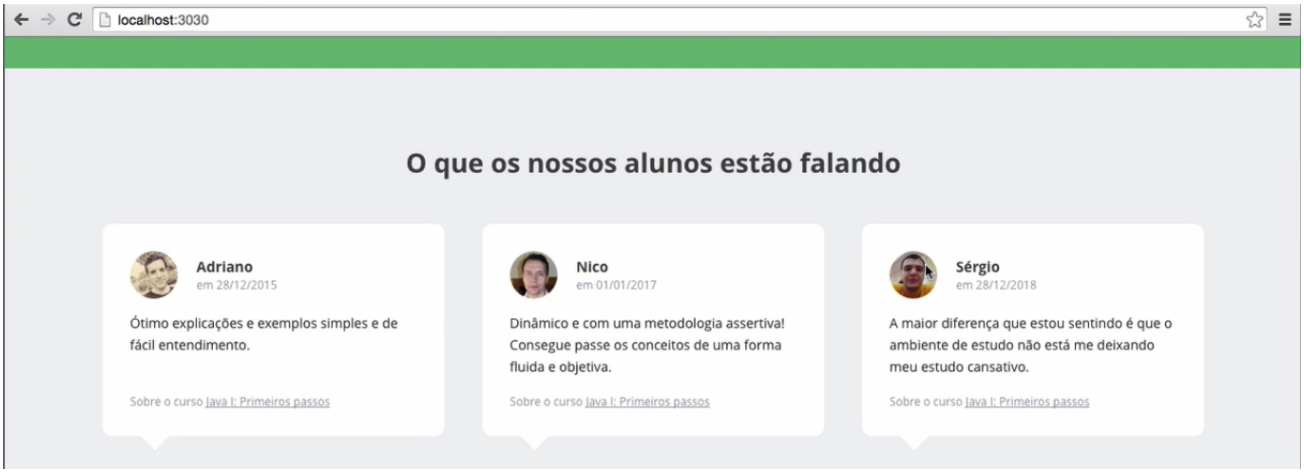
Então, podemos carregar as imagens que queremos no site e escolher entre dois tipos de otimização: *lossy* e *lossless*. O significado dessas duas otimizações é que uma é realizada com perdas e a outra sem perdas. Já retornaremos a esse tópico.

Vamos selecionar o *lossyless*. E vamos subir as três fotos que acabamos de criar. Ele mostra as três fotos, mostra o tamanho original das fotos e, por fim, nos diz o tamanho final, após a ferramenta remover tudo o que era desnecessário, todos os *metadados* que não são importantes. Economizamos cerca de 2/3 do tamanho desses arquivos.

Podemos baixar esses arquivos e inclusive visualizar um *preview* deles com a mesma qualidade.

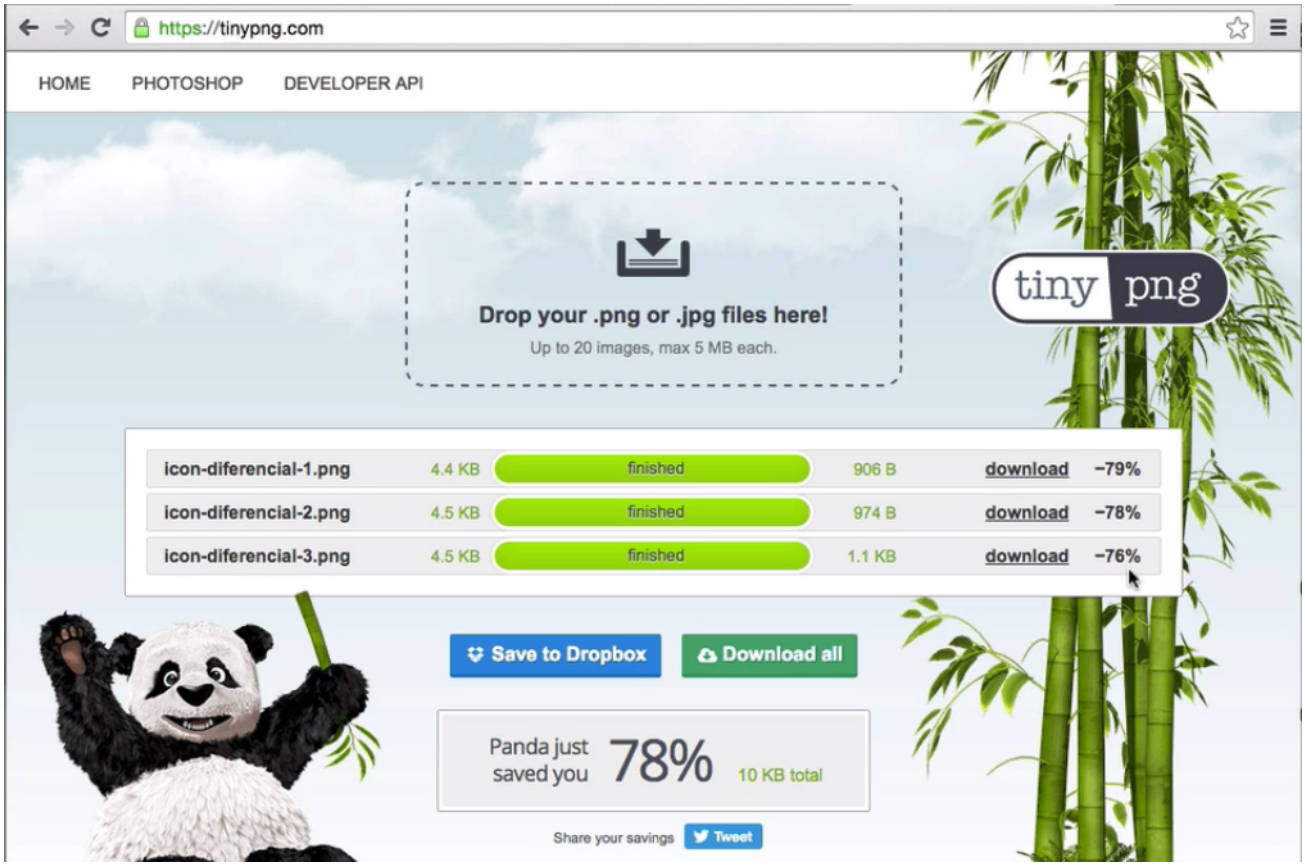
Usando o modo *lossless* é devolvido um arquivo que visualmente é o mesmo, idêntico, ao arquivo original, sem perda visual nenhuma.

Podemos fazer download dessas imagens, uma por uma, ou podemos baixar as imagens zipadas. Após baixar as imagens podemos pegar os três arquivos e substituir no site. Vamos atualizar o site e veremos que ele baixa a mesma coisa:



É legal utilizar alguma dessas ferramentas para otimização. Aqui, usamos o *Kraken*, mas existem outras diversas ferramentas como, *tinyong*, do site tinypng.com (http://tinypng.com), que apesar do nome aceita *png* e *jpeg*.

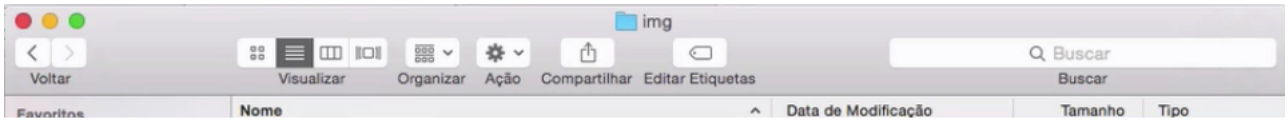
Vamos fazer um teste nesse site com imagens do tipo *png*. Ele faz a mesma coisa e repare que a economia que tivemos é de cerca de 78%.



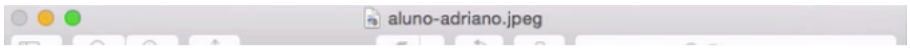
Quando abrimos o site kraken.io.com (http://kraken.io.com) vimos que tinhamos duas opções para otimização das imagens, o *lossy* e o *lossless*, este último chegamos a utilizar.

A diferença entre esses dois meios de salvar a imagem em uma versão menor é que uma não traz perdas de qualidade na imagem, o *lossless* e a outra, o *lossy* reescreve o arquivo potencialmente diminuindo a qualidade dele, mas com o objetivo de deixá-lo menor. o *lossy* é a mesma coisa que salvar um arquivo *jpeg* no *Photoshop* e escolher a qualidade dessa imagem. Quanto maior a qualidade, melhor ele fica. A ideia é escolher um balanço entre qualidade e tamanho final em "KB".

Temos os mesmos arquivos grandes que utilizamos anteriomente:



Vamos pensar neles através de uma otimização *lossy*. Vamos pegar o primeiro arquivo de imagem que possui "206 KB". Podemos exportar essa imagem e na hora de fazermos isso podemos escolher a qualidade que queremos, de máxima ao mínimo. Vamos escolher uma resolução média.



A ideia é que seja possível encontrar um balanço entre qualidade e tamanho, vamos salvar isso em "120 KB", salvaremos isso com um nome distinto para podermos fazer uma diferenciação "aluno-adriano2".

É possível notar alguma diferença entre as duas imagens? Aparentemente, não existe tanta diferença. Então, se diminuirmos a qualidade do arquivo, conseguimos achar um tamanho mais adequado.

Qual o processo? Teremos que olhar as imagens uma a uma, exportá-las uma a uma buscando um equilíbrio entre tamanho e qualidade?

Essas ferramentas online, o *lossy* e o *lossless* nos ajudam com isso.

Vamos pegar o arquivo original da foto do "Adriano" e usar, desta vez, com o *lossless*, isto é, sem perdas visuais. Ele vai diminuir cerca de "200 KB" para "187 KB", repare que temos uma diminuição de mais ou menos 7%.

Vamos fazer o mesmo teste utilizando o *lossy*. O *lossy* tenta encontrar uma maneira de deixar as imagens similares aos originais. Ele realiza perdas visuais que são imperceptíveis aos olhos humanos. Repare a diferença, tivemos uma diminuição maior usando o *lossy*:

Inclusive, existem outros sites que fazem isso, como o tinypng.co (http://tinypng.co)m . Vamos fazer o mesmo teste com a imagem do "Adriano":

A imagem fica com "105 KB". Vamos visualizar como ficou essa imagens, do lado direito temos a versão do *kraken* e do lado esquerdo a original:

Percebemos que as diferenças são bastante aproximadas e com um resultado bastante aceitável.

Portanto, essas ferramentas auxiliam muito para melhorar a performance do site através da manipulação das imagens. O mínimo que pode ser feito nesse sentido é também alterar isso usando editores de imagem, como o *Photoshop*, por exemplo.

Temos diversos arquivos *svg* para imagens vetoriais. Nenhuma das ferramentas que foram mostradas aceitam *svg*. Vamos buscar outra ferramenta, o *Svg Omg* que possui uma ferramenta online, no link (http://%20https://jakearchibald.github.io/svgomg/)https://jakearchibald.github.io/svgomg/ (https://jakearchibald.github.io/svgomg/). Essa é uma versão online que podemos utilizar para otimizar *svg*. Vamos testar.

Vamos pegar uma imagem do site do *Alura*, o arquivo "categoria -business.svg" que possui "6 KB" e que é a seguinte imagem, só que sem o fundo azul:

Utilizando essa ferramenta temos que esse arquivo ficará com mais ou menos "1 KB".

Se abrirmos esse mesmo arquivo *svg* no *Sublime* nos deparamos com o seguinte site:

O *svg* é, portanto, um *html*.

Temos diversas informações nesse código. Vamos reparar uma coisa. Lembra-se dos metadados? Nesse código temos algumas informações desnecessárias, como os comentários, perceba que temos um que aparece na imagem a cima onde está "Generator..." se apagarmos esse comentário, o arquivo segue o mesmo. Mas podemos apagar outras coisas além do comentário, por exemplo, as informações do *sketch* que são desnecessárias para o arquivo aparecer no navegador. Se formos editar o arquivo, novamente, no *sketch*, o programa vai ter problemas, ele vai perder informações que o *sketch* adicionou. Esse tipo de otimização remove alguns parâmetros e pode ser considerada uma otimização "lossy", isto é, com perdas, pois "perdemos" algumas capacidades do editor. Não tem perdas do ponto de vista visual.

Vamos voltar no link, você pode clicar na opção *Code* do link, que ele mostrará qual a otimização final:

Repare que é o mesmo *svg* que estava no *Sublime*. O que a ferramenta faz é retirar espaços, comentários, parâmetros do *sketch*, diminui as casas decimais. Inclusive, movendo o botão "*Show original*" podemos visualizar como estava o original do *svg* e comparar com o que acabamos de modificar.

Se não tiver diferença basta baixar o arquivo e substituir o arquivo pelo que tínhamos, no caso o "categoria-business.svg".

Como será que ficou?

Vamos atualizar o site e mostrar o ícone da categoria modificado:

Sempre é aconselhável guardar o *svg* original para possíveis modificações.

Vimos algumas ferramentas bastante fáceis de utilizar, mas muitas vezes vamos querer fazer isso na máquina de uma maniera igualmente automatizada.

Vamos utilizar ferramentas que podem ser utilizadas localmente para fazer esse tipo de otimização que estamos vendo.

Um tipo de ferramenta é a *ImageOptim* é uma ferramenta bastante simples que você pode baixar. Essa é a ferramenta:

Basta arrastar os arquivos para dentro da ferramenta e ela faz as otimizações necessárias. O *ImageOptim* faz otimizações do tipo *lossless*. Uma das grandes vantagens dele é que substitui o arquivo original e mostra o quanto economizou.

Por substituir os arquivos ele só faz esse tipo de otimização e não faz *lossy*, para não correr nenhum risco. Ele otimiza, *jpeg*, *png* e etc... E se você quiser voltar atrás ele joga os arquivos originais na lixeira e podem ser repostos.

Se você quiser fazer otimizações do tipo *lossy*, com perda visual, será necessário utilizar ferramentas visuais, exportar no photoshop ou usar as ferramentas virtuais que já vimos.

E o que podemos usar no windowns?

O *Riot*, encontrado no site luci.criosweb.ro/riot/ (http://luci.criosweb.ro/riot/) que também é gratuito e funciona mais ou menos da mesma maneira e possui até mais recursos.

Para *svg* a ferramenta que pode ser utilizada é o *Svg Omg* mesmo.

Mas como o *Image Optim* faz para otimizar? Ele é apenas uma interface gráfica que roda ferramentas famosas de otimização que estão no mercado. Ferramentas que podemos baixar e rodar localmente na máquina e todas elas são ferramentas de linha de comando, muito fáceis de serem automatizadas. Podemos fazer um *script* em linha de comando que rodamos essa ferramenta em diferentes arquivos.

Vamos passar, rapidamente, como usaríamos isso na linha de comando. Vamos começar pelo formato *jpeg*. Uma das ferramentas mais famosas é a *jpegtran*, instalamos ela no *Linux*, no *Mac* ou no *windows*.

Você pode instalar ela apartir do link jpegmini.com (http://jpegmini.com), e nesse site estão todas as instruções necessárias para realizar esse processo. Nós já temos isso instalado.

Vamos digitar em nosso terminal `jpegtran` e ele recebe várias opções. Podemos ver essas opções dando um `--help`. Uma dela é o `optmized`:

Vamos digitar `-optimized` e tem, ainda, outras opções, como o `-progressive` ele converte o *jpeg* em um *jpeg progressivo* e, por fim, escreveremos `site/assets/img/alura-adriano.jpeg > dist/assets/img/alura-adriano.jpeg`. O que estamos dizendo com essa última parte? Que pegaremos a imagem do "Adriano", novamente, no arquivo original e jogaremos isso na pasta "dist", "dist/assets/img/alura-adriano,jpeg". Teremos o seguinte:

```
jpegtran -optmize -progressive site/assets/img/alura-adriano.jpeg > dist/assets/img/alura-adriar
```

Vamos ver o resultado?

Vamos comparar os tamanhos das fotos:

Eram "206 KB" e ficamos com "193 KB". Lembre-se que essa otimização é do tipo *lossless* que só tira o que é desnecessário, ele não diminui a qualidade visual, o que daria um resultado bem diferente.

Existem outras ferramentas, como o *pngcrush* ele permite fazer otimizações de *png*. E faremos um teste escolhendo o arquivo "site/assetsqimgqicon-diferencial-1.png" e ele recebe a pasta de destino "dist/assets/img/icon-diferencial-1.png".

```
pngcrush site/assetsqimgqicon-diferencial-1.png dist/assets/img/icon-diferencial-1.png
```

Damos um "Enter" nisso e vamos observar se tivemos alguma diferença:

Na primeira imagem temos as diferenças de tamanho e na segunda temos as imagens comparadas.

E para *svg*? Podemos utilizar o formato *svgo* que era utilizado na interface e que agora testaremos localmente. Basta instalar a ferramenta, digitando `npm install -g sgvo` e ele irá instalar. Vamos testar ela seguindo o mesmo esquema dos anteriores digitando, digitando o programa, o local da imagem e a pasta para onde queremos que ela vá, então:

```
svgo site/assets/img/categoria-business.svg dist/assets/img/categoria-business.svg
```

E teremos que:

Existem ferramentas em linha de comando que podemos explorar para automatizar a geração das imagens de forma otimizada. Aliás, lembra-se que estávamos utilizando o *gulp*?

Não tocaremos no tema do *gulp* , pois, temos cursos específicos para isso.

Já deixamos preparados no "gulp file" uma *taskie* que serve para trabalhar com imagens:

E ele trabalha com o *imagemin* que faz otimização das imagens de diversos formatos, *jpeg*, *png*, *svg* e etc e existem até alguns *pluggins* para acrescentar outros tipos.

A ideia é que para rodar digitamos em nosso terminal `gulp imagemin` ele utiliza todas as imagens do projeto de maneira automatizada. Sem precisar rodar comando por comando.

Vamos comparar o tamanho das pastas, a orginal possui cerca de "754 KB" e a pasta final economizou "92 KB", ficando com "668 KB".

Podemos comparar também os tamanhos dos arquivos individualmente:

Repare que com um único comando podemos fazer a otimização, não é muito rápido, ele demorou cerca de 11 segundos, se tivesse milhares de imagens tardaria ainda mais. Então, você pode ter o interesse de otimizar as imagens originais e assim só fazemos isso uma única vez ao em vez de ficar várias vezes re-otimizando as mesmas imagens. Já deixa isso meio que cacheado.

Lembre-se que começamos falando que as imagens representam cerca de 2/3 do peso das passagens geralmente? Então, precisamos otimizar isso, podemos utilizar ferramentas online ou localmente. E existem dois tipos de otimização de imagem, com perdas e sem perdas. O segredo é: otimizar, pois, economiza muita coisa da performance final.