

**FREEK.DEV****Laravel / PHP / JavaScript**

[Oh céus!](#) monitora todo o site, não apenas a página inicial. Você receberá uma notificação assim que o site estiver inativo, um relatório mensal de tempo de atividade, um aviso alguns dias antes de seu certificado SSL expirar e muito mais! [Comece seu teste gratuito de 10 dias agora!](#)

Sobre a migração do meu blog do WordPress para um aplicativo Laravel

Original – 20 de novembro de 2017 por Freek Van der Herten - 12 minutos vermelho

Visitantes regulares devem ter notado que na semana passada este blog recebeu uma nova camada de tinta. Esse novo layout não é apenas um novo tema do WordPress. As coisas mudaram no backend também. Anteriormente, meu blog era desenvolvido com WordPress. Eu o migrei para um aplicativo Laravel personalizado. Esse aplicativo é de código aberto. Você pode encontrar o código real que está sendo implantado no servidor [neste repositório no GitHub](#).

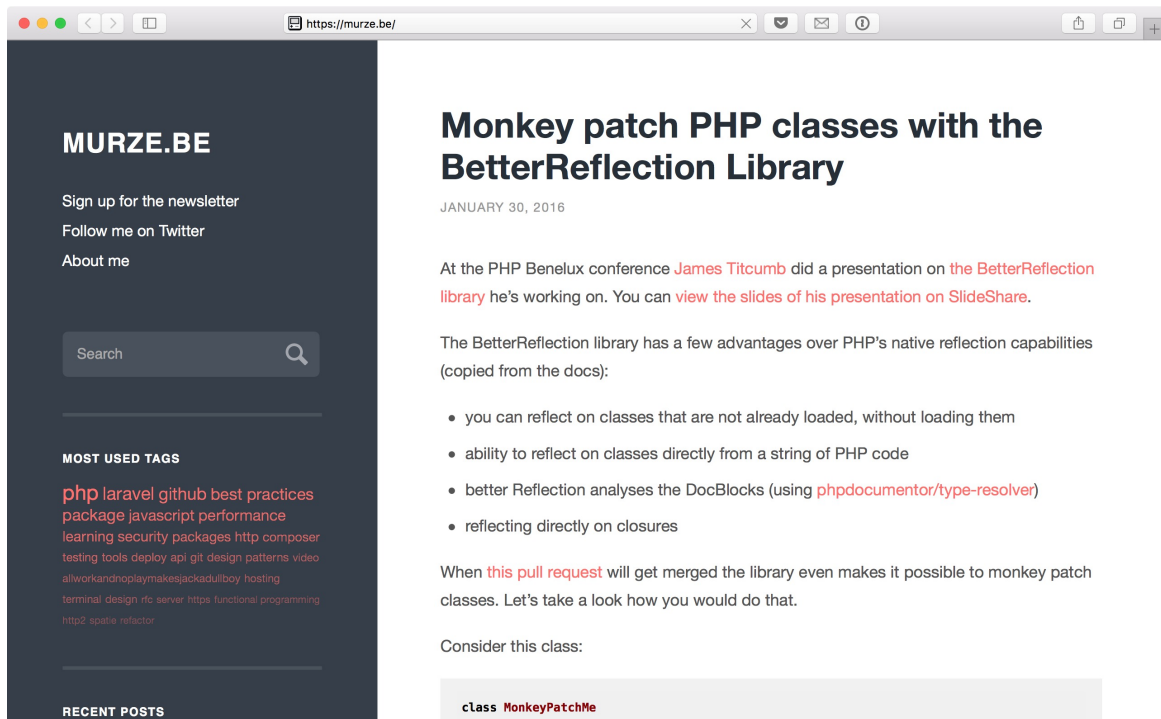
Neste post, gostaria de explicar por que e como fiz essa migração.

Amar e abandonar o WordPress

Quando comecei a blogar, [quase três anos atrás](#), eu só queria ter uma maneira fácil de compartilhar links interessantes para os blogs de outras pessoas. WordPress foi perfeito para isso. É facilmente instalado. Ele vem com ótimos recursos prontos para uso, como uma excelente interface administrativa, feeds RSS, uma boa pesquisa, ... Existem também ótimos plugins para adicionar itens como rastreamento de Analytics, cache de páginas, otimização de SEO

minha empresa, eu não fiz nenhum estilo, nunca me preocupei em aprender css, por isso também era bastante útil que o WordPress tivesse milhares de temas disponíveis.

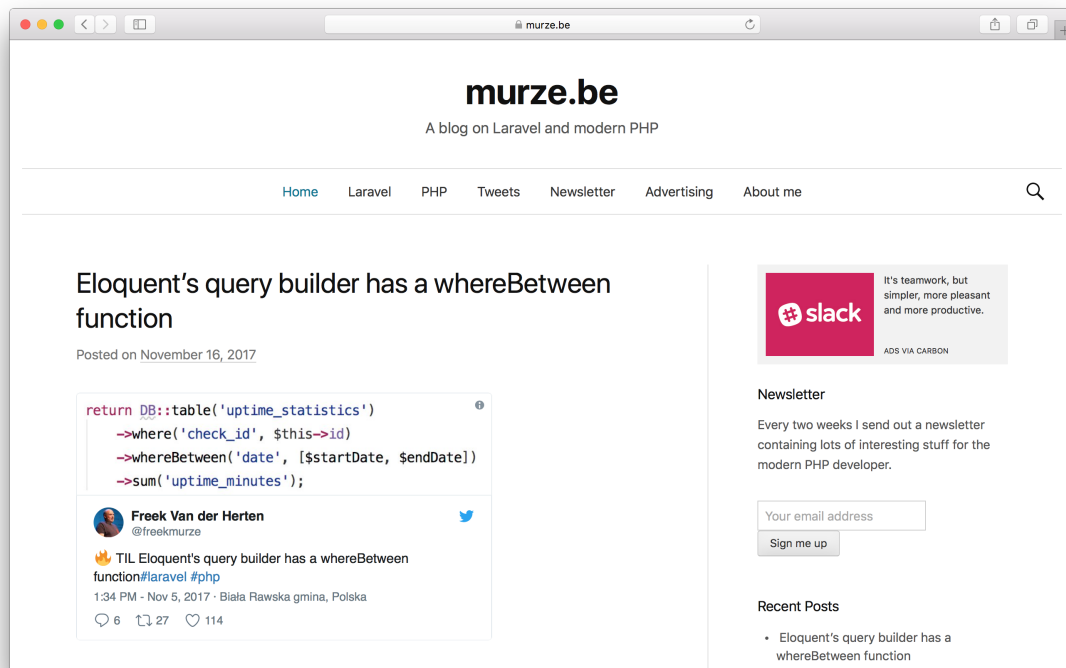
Pelo primeiro ano na terra do WordPress, a vida foi boa. Veja como o blog parecia na época:



Enquanto isso, comecei a trabalhar no espaço de código aberto e, junto com [minha equipe, criei alguns pacotes](#) . Em vez de compartilhar apenas links no meu blog com o conteúdo de outras pessoas, comecei a escrever postagens introdutórias nesses pacotes. Como o blog começou a atrair mais visitantes, decidi fazer um pouco de esforço para melhorar a aparência do blog. Fiz isso instalando um novo tema (que foi modificado um pouco pelo meu colega [Willem](#)).

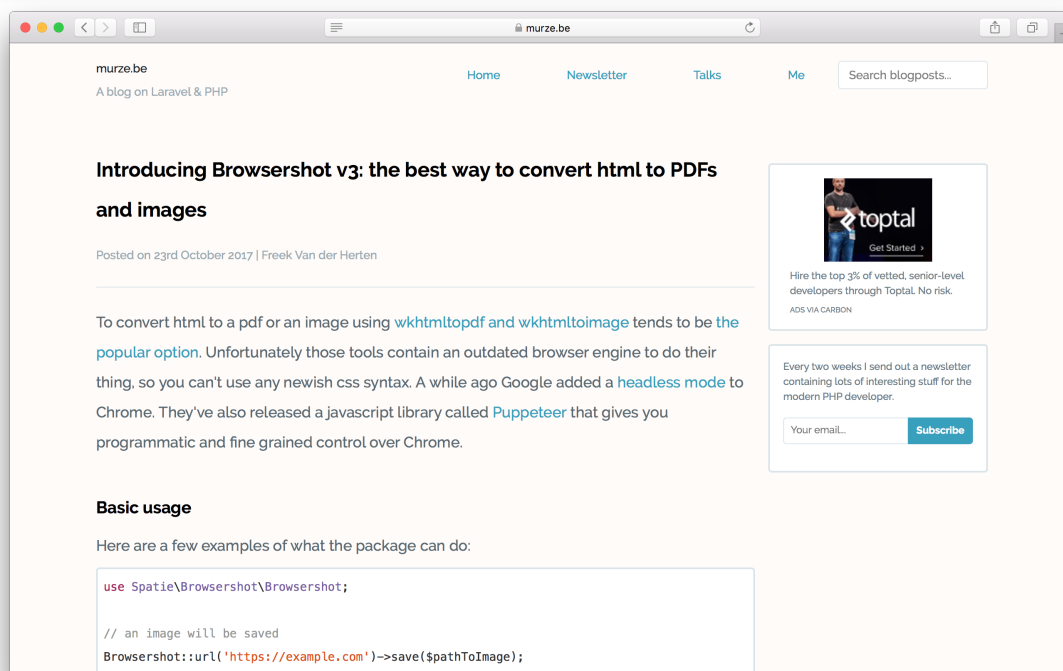
Eu queria personalizar um pouco mais a colocação de todo o conteúdo, mas como um desenvolvedor que não é do WordPress, mexer com a base de código do WordPress pode ser bastante assustador. Estou acostumado a trabalhar com código PHP moderno. Eu fiz alguns hacks sujos em alguns dos arquivos php. Isso não parecia muito bom, mas funcionou. Como usuário, eu amo o WordPress, mas como desenvolvedor, eu meio que odeio.

Por um tempo, fiquei feliz novamente com a aparência do blog. Aqui está uma captura de tela da aparência de outubro de 2016 até alguns dias atrás.



No verão deste ano, Adam Wathan [anunciou](#) que, junto com alguns amigos, estava trabalhando em uma nova estrutura de css de código aberto chamada [Tailwind](#) . Como mencionado acima, eu nunca tive tempo para aprender css e o considerei meu calcanhar de Aquiles. O vento de cauda pode ser um ponto de entrada interessante no mundo do css para mim. Estou sempre mais motivado para aprender algo se posso usar novos conhecimentos em um projeto concreto. Por isso [decidi](#) reconstruir meu blog com Tailwind, Laravel e alguns de nossos próprios pacotes.

Sem nenhum conhecimento prévio em CSS, trabalhei com [os excelentes documentos do Tailwind](#) . Eu li [este post](#) do [blog](#) do Adam e do co-criador de Tailwind Steve Schoger. [Este vídeo](#) sobre a reconstrução do [Laravel.io também](#) foi muito útil. Estou experimentando um pouco com Tailwind. Eu não tinha um design específico em mente ao iniciar o projeto. O design final que você vê agora é um resultado acidental do meu aprendizado.

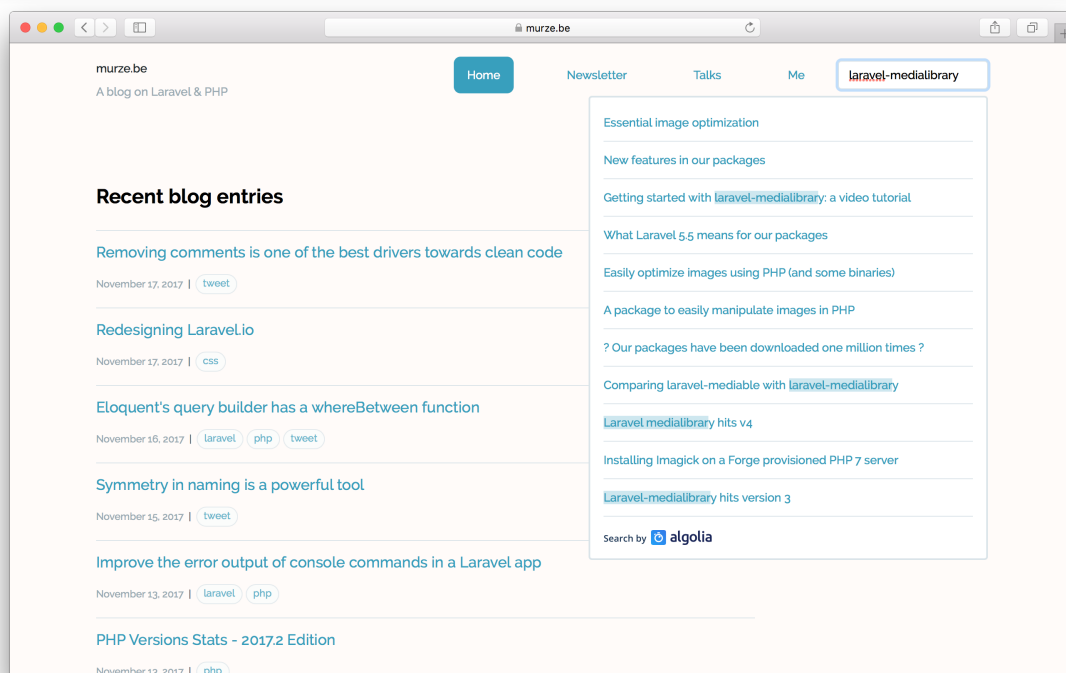


Nos bastidores

Como mencionado acima, meu blog agora é um aplicativo Laravel. Você pode ver o código-fonte inteiro [neste repositório no GitHub](#). Vamos destacar alguns bits interessantes.

Pesquisando posts do blog

A pesquisa em posts do blog é feita com o [Algolia](#), um serviço de pesquisa rápida. Todos os posts do blog são transferidos do banco de dados para a Algolia usando o [Laravel Scout](#). Com o Scout instalado, você recebe [um comando Artisan](#) para executar a transferência inicial de uma tabela de banco de dados para a Algolia. Scout também fornece uma `searchable` característica que, [aplicada ao modelo](#), garantirá que novos registros e atualizações dos existentes sejam enviados também para o índice da Algolia.



Agora que você sabe como as postagens são transferidas para o índice, vamos dar uma olhada em como essas postagens são recuperadas. Se você tentar pesquisar postagens de blog, provavelmente perceberá que os resultados aparecem quase instantaneamente. As solicitações de pesquisa não atingem meu próprio servidor. Toda essa operação de pesquisa acontece no lado do cliente. [Esse componente do Vue](#) é responsável por renderizar o campo de pesquisa e os resultados. Ele usa a [API JavaScript da Algolia](#) para executar solicitações de pesquisa.

Devo mencionar que o Algolia não é um serviço gratuito, mas, a menos que você realmente espere muitas solicitações de pesquisa, o nível gratuito é suficiente para usar em um blog pessoal.

Importando postagens do WordPress

É claro que eu não queria perder todo o conteúdo antigo, então tive que criar uma maneira de transferir postagens do WordPress para o meu novo aplicativo Laravel. Felizmente, isso não é tão difícil. Aqui está [o comando artesanal](#) que importa postagens e tags de um banco de dados WordPress. Para lidar com tags, usei [neste pacote laravel](#).

[tags desenvolvido em casa](#) . Esse pacote facilita a [associação de tags a um modelo Eloquent](#) .

Manipulação de URLs alterados

Ao usar o WordPress o ano de publicação e mês de um post é usado na URL por exemplo: <https://freek.dev/2017/06/building-realtime-dashboard-powered-laravel-vue-2017-edition/> . Para o meu novo blog, eu queria perder esse componente de tempo e usar links mais curtos como <https://freek.dev/building-realtime-dashboard-powered-laravel-vue-2017-edition> . Os URLs antigos do WP estão no índice do Google. Se deixados sem tratamento, todos os visitantes que acessarem meu blog por meio de uma pesquisa no Google verão uma página 404.

[Nosso pacote laravel-missing-page-redirector](#) foi criado para resolver esse problema. O comando import também [preenche uma redirect tabela](#) . O pacote de redirecionador tem suporte para criar seu próprio redirecionador para especificar uma fonte para os redirecionamentos (como uma tabela de banco de dados). Aqui está [o redirecionador](#) que lê a [redirects](#) tabela.

Com tudo isso no lugar, todos os URLs antigos do meu blog são redirecionados para as novas páginas. Experimente visitando <https://freek.dev/2017/06/building-realtime-dashboard-powered-laravel-vue-2017-edition>

Fornecendo um feed RSS

Eu tento manter-me atualizado com o que está acontecendo no meu setor lendo muitos blogs. Visitar cada blog separadamente levaria muito tempo, é por isso que prefiro [assinar feeds RSS](#) e ler todo o conteúdo no [meu leitor de RSS preferido](#) .

É claro que o murze.be redesenhado deve ter um feed RSS. O WordPress vem com suporte para feeds RSS prontos para uso. Nas terras do Laravel, você pode usar outro pacote caseiro: [laravel-feed](#) . Esse pacote facilita muito adicionar um feed ou vários feeds ao seu blog. Aqui está [a função que transforma uma postagem em um item de feed](#) .

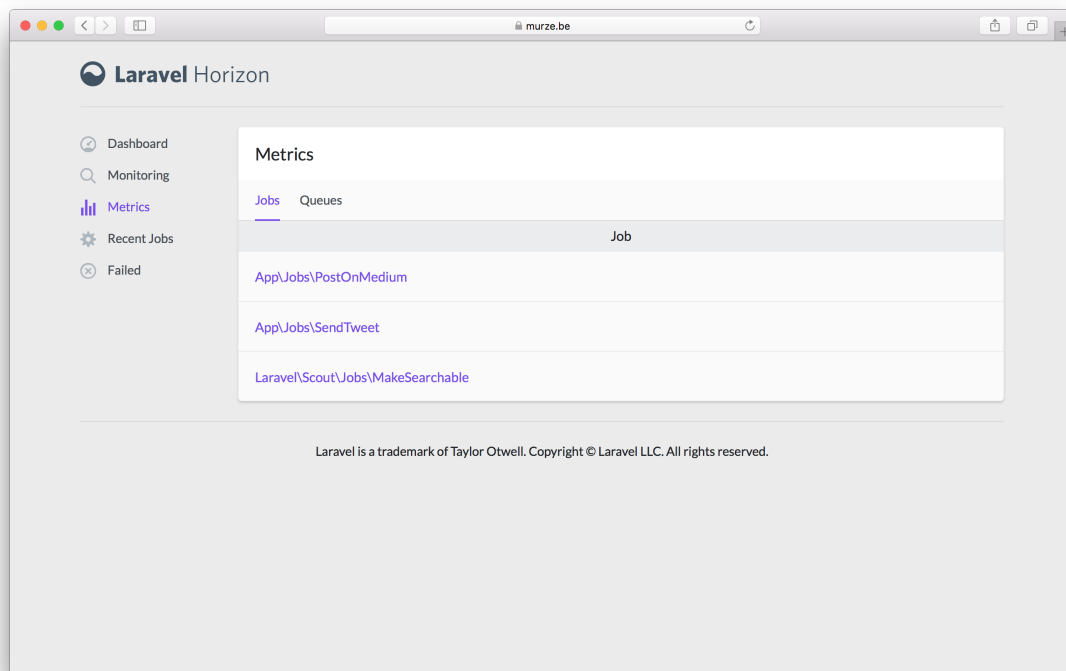
Os próprios feeds são encontrados em <https://freek.dev/feed> (contém todas as postagens) e <https://freek.dev/feed/php> (contém apenas postagens marcadas `php`).

Conteúdo de crossposting

Sempre que eu publicava uma nova postagem no meu blog WordPress, um tweet era enviado e o conteúdo era postado cruzadamente no Medium. Tudo isso foi tratado por dois plugins. Claro que eu queria manter esse comportamento. Felizmente, as APIs do Twitter e do Medium são muito fáceis de usar, então eu mesmo codifiquei uma solução sem depender de nenhum pacote.

Aqui está [o código no post modelo](#) que despacha os [SendTweet](#) e os [PostOnMedium](#) trabalhos.

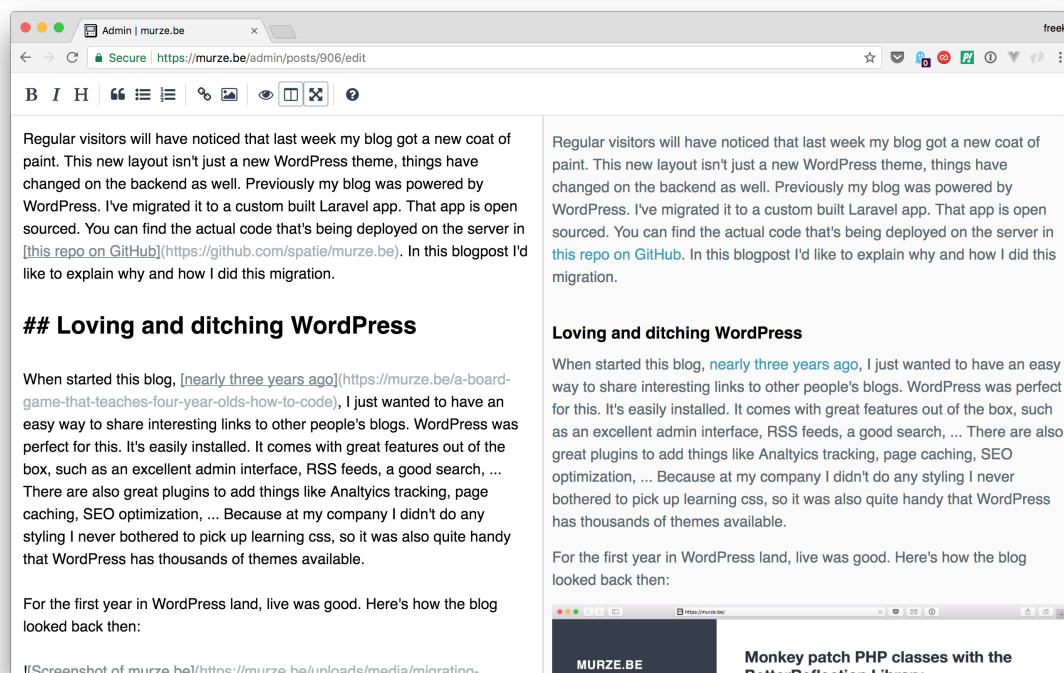
No meu antigo blog do WP, uma publicação sempre levava alguns segundos, porque a comunicação com o Twitter e o Medium demora um pouco. No meu novo blog do Laravel, algo é muito rápido, porque esses [SendTweet](#) e os [PostOnMedium](#) posts estão na fila. De fato, toda a sincronização acima mencionada com a Algolia também está na fila. [O Horizon](#) é usado para manipular e monitorar todos os trabalhos na fila.



Diversos

Escrevendo Markdown

Para escrever posts, construí uma interface administrativa básica. Eu gosto de escrever Markdown em vez de html simples. Eu tentei um par de editores de markdown com JS e [comecei a](#) usar o [SimpleMDE](#) .



No site da frente, o conteúdo do Markdown deve ser renderizado como html. Eu adicionei esse método ao `Post` modelo.

```
public function getTextAttribute($original)
{
    return (new Parsedown())->text($original);
}
```

Isso garantirá que, sempre que for `$post->text` usado em uma exibição, o conteúdo de uma postagem seja convertido de Markdown para html.

Melhorando o desempenho

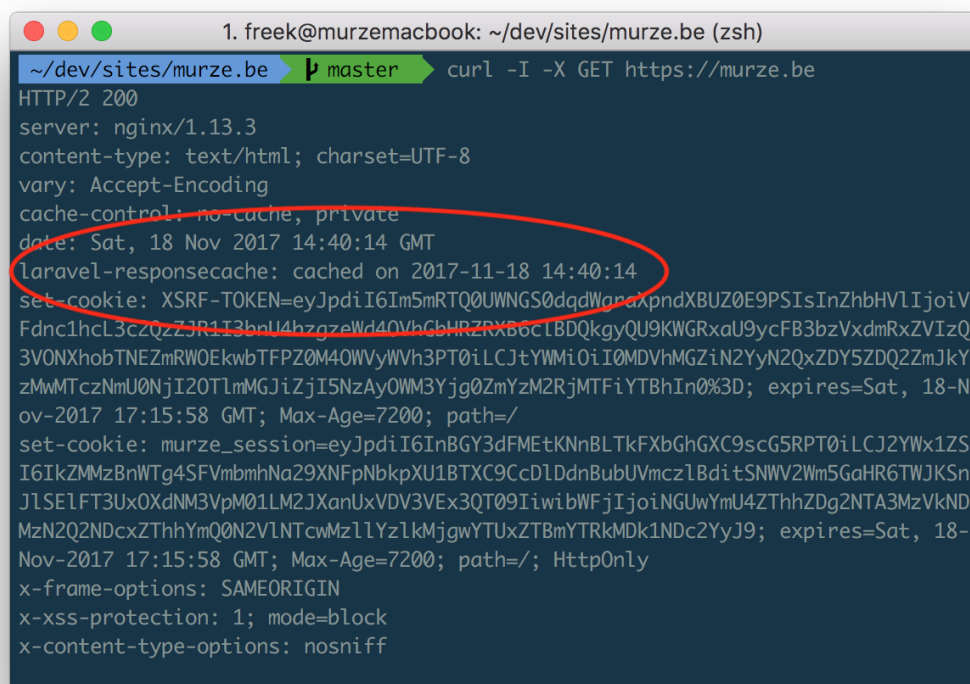
O aplicativo Laravel já é bastante rápido, mas podemos fazer melhor. No passado, havia alguns dos meus posts que acabavam bem altos no Hacker News. Quando isso acontece, há muitos leitores simultâneos e o blog deve ser capaz de lidar com isso. No WordPress, usei o plugin [WP Super Cache](#). Isso salvará uma página renderizada como um arquivo em disco. Na próxima vez que a mesma página for

como um arquivo em disco. Na próxima vez que a mesma página for

solicitada novamente, ela servirá apenas o conteúdo salvo desse arquivo, em vez de criar a página novamente do zero.

No Laravel, a mesma coisa pode ser alcançada usando mais um de nossos pacotes: [laravel-responsecache](#) . Se você quiser saber mais sobre esse pacote, leia [o post do blog de introdução](#) .

Você pode ver o pacote funcionando se você inspecionar os cabeçalhos de resposta deste blog. Observe o `laravel-responsecache` cabeçalho.



```
1. freek@murzemacsbook: ~/dev/sites/murze.be (zsh)
~/dev/sites/murze.be master curl -I -X GET https://murze.be
HTTP/2 200
server: nginx/1.13.3
content-type: text/html; charset=UTF-8
vary: Accept-Encoding
cache-control: no-cache, private
date: Sat, 18 Nov 2017 14:40:14 GMT
laravel-responsecache: cached on 2017-11-18 14:40:14
set-cookie: XSRF-TOKEN=eyJpdiI6Im5mRTQ0UWNGS0dqWgR0XpndXBUZ0E9PSIsInZhbnVlIjoIvFdc1hcL3czQzZ3R1I3BnU4bzgzeWd4OVhCbURZRXBoc1BDQkgYQU9KWGRxaU9ycFB3bzVxdmRxZVIzQ3VONXhobTNEZmRW0EkwbTFPZ0M4OWVyWVh3PT0iLCJtYWMiOiI0MDVhMGZiN2YyN2QxZDY5ZDQ2ZmJkYzMwMTczNmU0NjI2OTlmMGJiZjI5NzAyOWM3Yjg0ZmYzM2RjMTFiYTBiIn0%3D; expires=Sat, 18-Nov-2017 17:15:58 GMT; Max-Age=7200; path=/
set-cookie: murze_session=eyJpdiI6Im5mRTQ0UWNGS0dqWgR0XpndXBUZ0E9PSIsInZhbnVlIjoIvFdc1hcL3czQzZ3R1I3BnU4bzgzeWd4OVhCbURZRXBoc1BDQkgYQU9KWGRxaU9ycFB3bzVxdmRxZVIzQ3VONXhobTNEZmRW0EkwbTFPZ0M4OWVyWVh3PT0iLCJtYWMiOiI0MDVhMGZiN2YyN2QxZDY5ZDQ2ZmJkYzMwMTczNmU0NjI2OTlmMGJiZjI5NzAyOWM3Yjg0ZmYzM2RjMTFiYTBiIn0%3D; expires=Sat, 18-Nov-2017 17:15:58 GMT; Max-Age=7200; path=/; HttpOnly
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
x-content-type-options: nosniff
```

Enviando boletins

A cada duas semanas, envio um resumo de todo o novo conteúdo do meu blog aos assinantes do meu boletim. Você pode se inscrever [aqui](#) . Alguns meses atrás, para diminuir as despesas, troquei o MailChimp em favor de [Sendy](#) . Você encontrará mais informações sobre como mudar para Sendy [neste post do blog de Mattias Geniar](#) .

Antes, eu precisava coletar manualmente todos os títulos, links e trechos para incluir no boletim. Para todas as edições do meu boletim que levavam de 30 a 45 minutos. No meu novo blog, adicionei [um código para buscar automaticamente todo o conteúdo e criar o html do boletim](#).

Implantando uma nova versão

Sou um grande fã do [Envoy](#), uma ferramenta para executar facilmente comandos em servidores remotos. O repositório do meu blog contém [um script enviado](#) que pode executar uma implantação do tipo Capistrano com tempo de inatividade quase zero.

No fechamento

Espero que você tenha gostado desse pequeno tour pela base de código do meu blog. Sinta-se à vontade para bifurcar o código e usá-lo em seu próprio blog. Lembre-se de que o código no repositório não é um CMS completo, mas apenas um aplicativo simples adaptado às minhas necessidades.

Eu me diverti muito aprendendo Tailwind. Sem nenhum conhecimento prévio de css, consegui criar rapidamente um design em que estou feliz. Também houve alguma frustração quando as coisas não funcionaram como o esperado. Mas culpo minha própria falta de habilidades em CSS por isso. Felizmente, [minha equipe](#) e algumas pessoas incríveis da comunidade [me ajudaram um pouco](#), quando eu estava presa.

Se você estiver no Laravel, verifique o repositório que contém o código-fonte: <https://github.com/spatie/murze.be>

Mantenha-se atualizado com tudo o que é

Laravel, PHP e JavaScript.

[Siga-me no Twitter](#) . Eu tweet regularmente dicas de programação e o que eu mesmo aprendi em projetos em andamento.

A cada duas semanas, envio um boletim informativo contendo muitas coisas interessantes para o desenvolvedor PHP moderno.

Espere dicas e truques rápidos, tutoriais interessantes, opiniões e pacotes. Como trabalho com o Laravel todos os dias, há uma ênfase nessa estrutura.


Your e-mail address


[Se inscrever](#)


Tenha certeza de que usarei seu endereço de e-mail apenas para enviar a newsletter e não a utilizarei para outros fins.


What do you think?


7 Responses

 Upvote

 Funny

 Love

 Surprised

 Sad

2 Comments

[murze.be](#)

 Renato De Oliveir... ▾

 Recommend 1

 Tweet

 Share

Sort by Best ▾



Join the discussion...



tallesairan • 3 months ago

<3

^ | ▾ • Reply • Share ›



Usman Arshad • 6 months ago

great

^ | ▾ • Reply • Share ›

 Subscribe  Add Disqus to your siteAdd DisqusAdd

 Disqus Disqus Disqus Disqus Disqus

