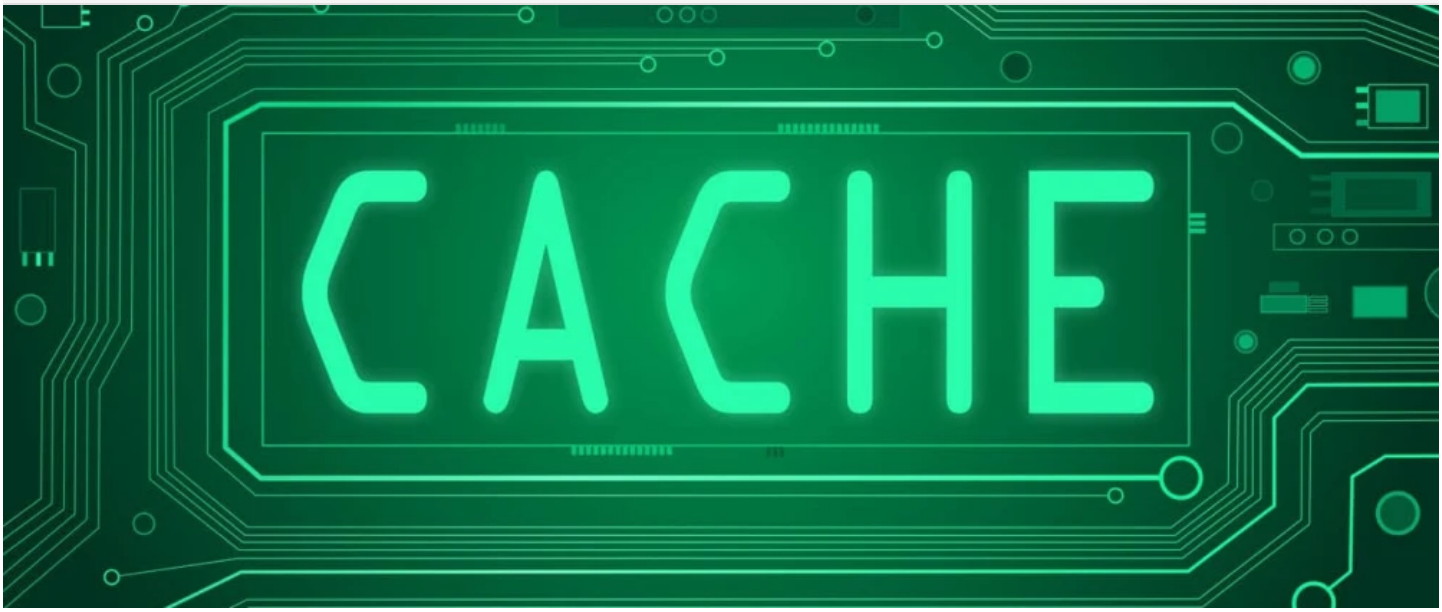

Felipe Marques

Posted on 9 de out. de 2021

Como desabilitar o cache no Angular

#angular #typescript #tutorial

Um dos grandes problemas que enfrentei ao atualizar o frontend de aplicações angular foi o cache.

Praticamente todos os navegadores mantêm um cache da aplicação no computador do cliente para conseguir carregar a aplicação de forma mais rápida e com menor consumo de dados e isso é excelente. Entretanto, um efeito colateral é que as atualizações podem demorar para serem entregues para o cliente. Como consequência, o cliente pode ficar horas ou dias com uma versão com um bug que já foi corrigido.

Em sistemas internos de empresas, isso pode resultar em ligações ou tickets constantes para a equipe de suporte técnico, causando uma sobrecarga na equipe. Quando estamos falando de aplicações fornecidas para clientes externos à empresa, pode ser um ponto de frustração e insatisfação com o produto.

Nesse contexto, pode ser interessante desabilitar o cache das aplicações Angular. Veremos a seguir como fazer isso.

Index.html

A primeira opção para evitar o cache dos navegadores é simplesmente indicar ao navegador como ele deve se comportar. Para fazer isso devemos incluir 3 tags `meta` no arquivo index.html do projeto.

```
<meta http-equiv="cache-control" content="no-cache, must-revalidate, post-check
<meta http-equiv="expires" content="0">
<meta http-equiv="pragma" content="no-cache">
```

Com isso, estamos dizendo ao navegador como ele deve tratar o cache, basicamente, dizemos para ele não fazer cache.

Angular Interceptor

Uma segunda opção é incluir no cabeçalho das requisições http a instrução para o navegador não fazer cache.

Se você não sabe como criar um interceptor no angular, veja o artigo a seguir:

Introdução ao Angular HttpInterceptor | by Matheus Bizutti | Matheus Bizutti | Medium

Matheus Bizutti • Sep 5, 2018 •

Medium

Agora que você já sabe como criar o interceptor, vamos incluir nosso código no interceptor.

```
import { Injectable } from '@angular/core';
import {
  HttpRequest,
  HttpHandler,
  HttpEvent,
  HttpInterceptor
} from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable()
export class NoCache.InterceptorInterceptor implements HttpInterceptor {

  constructor() {}

  intercept(request: HttpRequest<unknown>, next: HttpHandler): Observable<HttpE
    // inclua o código da linha abaixo logo no início do método intercept
```

```
    request = request.clone({
      setHeaders: {
        "Cache-Control": "no-cache",
        Pragma: "no-cache",
      },
    });

    return next.handle(request);
  }
}
```

--output-hashing=all

Uma outra forma de forçar a remoção do cache é compilar a aplicação usando o parâmetro `--output-hashing=all`. Assim ao fazer o build com esse parâmetro os arquivos que foram alterados receberão nomes diferentes, quebrando de forma proposital o mecanismo de cache dos navegadores.

`ng build --output-hashing=all` ou `npm run ng build --output-hashing=all`

Conclusão

Como podemos ver, as três opções são simples de implementar. Quando estamos falando de aplicações empresariais para uso interno, dentro da rede da empresa, não vejo motivos para não usar esse recurso, entretanto, se sua aplicação é utilizada por clientes que a consomem pela internet, use esse recurso com sabedoria, lembre-se que seu cliente pode estar acessando seu produto pelo smartphone com um plano limitado de dados e isso também pode ser um ponto de insatisfação em relação ao seu produto.