

## Requisitos não funcionais

### Transcrição

Começando deste ponto? Você pode fazer o [download \(https://s3.amazonaws.com/caelum-online-public/typescript/08-alurabank.zip\)](https://s3.amazonaws.com/caelum-online-public/typescript/08-alurabank.zip) completo do projeto do capítulo anterior e continuar seus estudos a partir deste capítulo.

Criamos uma solução de renderização da view, que sempre renderiza do zero a view. Por exemplo, a cada inclusão de uma nova negociação em nosso modelo `Negociacoes`, nós renderizamos a tabela do zero toda vez. Para o escopo da nossa aplicação, isso não nos trará problema algum, mas será que em uma aplicação maior a solução imperativa seria mais indicada, ou quem sabe a utilização de algum framework que seja especializado nisso?

Para que possamos responder a essa pergunta, é necessário realizar um teste de performance. Talvez, o mais simples de todos seja subtrair o tempo final da chamada do tempo do tempo inicial, para termos uma ideia em segundos de quanto o método demora para ser executado.

A própria especificação do JavaScript disponibiliza o objeto global `performance` para isso. Não precisamos nos preocupar em baixar seu arquivo tds, porque o TypeScript já traz em sua instalação todos os tds para a especificação JavaScript.

Vamos alterar o método `adiciona()` de `NegociacaoController` e realizar o teste:

```
adiciona(event: Event) {  
  const t1 = performance.now();  
  // código omitido  
  const t2 = performance.now();  
  console.log(`Tempo de execução do método adiciona(): ${(t2 - t1)/1000} segundos`);  
}
```

Podemos ter um teste mais preciso, usando a mesma estratégia no método `update()`, da nossa classe `View`.

```
abstract class View<T> {  
  
  private _elemento: Element;  
  
  constructor(seletor: string) {  
  
    this._elemento = document.querySelector(seletor);  
  }  
  
  update(model: T) {  
    const t1 = performance.now();  
    this._elemento.innerHTML = this.template(model);  
    const t2 = performance.now();  
    console.log(`Tempo de execução do método adiciona(): ${(t2 - t1)/1000} segundos`);  
  }  
}
```

```
abstract template(model: T): string;  
  
}
```

O problema é que começamos a ter um código espalhado pela nossa aplicação. Se quisermos testar outros métodos, precisaremos fazer a mesma coisa. Por mais que o código seja isolado em uma função, teremos que lembrar de chamar a função no início e no final do método. Será que há uma solução para isso? Queremos poder isolar o código que teste a performance em um único lugar.