UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Qualidade de Software 2025.2

Refatoração de Code Smells em Frameworks Front-end

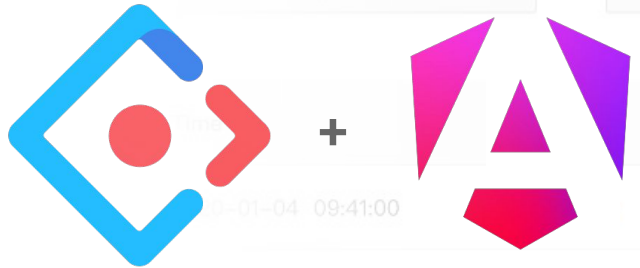
NG-ZORRO - Ant Design of Angular

Anaildo do Nascimento Silva
João Evangelista De Souza Alves

NG-ZORRO

Biblioteca Enterprise de Componentes Angular baseada no Ant Design

- 70+ componentes
- Código em TypeScript
- Design moderno e corporativo ideal para dashboards
- Temas customizáveis e suporte a internacionalização



Tipos de Code Smells

Tipo	Descrição
DOM	Direct DOM Manipulation - Manipulação direta do DOM, ignorando o controle do framework.
LC	Large Component - Componente com muitas responsabilidades.
IIC	Inheritance Instead of Composition - Uso de herança onde composição seria mais adequada.
TMI	Too Many Inputs - Componente recebe dados em excesso.
ANY	Overusing Any Type - Uso excessivo do tipo any, perdendo segurança de tipos.
LF	Large File - Arquivo grande e difícil de manter.
EPC	Excessive Parent-to-Child Communication - Comunicação excessiva entre componentes pai e filho.

Usando a ferramenta

- Executando a ferramenta ***angular-code-smells***.
- 364 arquivos sinalizaram algum tipo de smell.

```
joaoev@skyline C:\..\..\angular-code-smells  main > npx tsx lib/index.ts ./../../Users/joaoev/ng-zorro-antd/components/
C:\workplace\angular-code-smells
```

(index)	file	DOM	IIC	LC	ANY	TMI	LF	EPC
0	'..\\..\\Users\\joaoev\\ng-zorro-antd\\components\\affix\\affix.component.ts'	0	0	1	0	0	1	0
1	'..\\..\\Users\\joaoev\\ng-zorro-antd\\components\\affix\\affix.spec.ts'	1	0	0	0	0	1	0
2	'..\\..\\Users\\joaoev\\ng-zorro-antd\\components\\alert\\alert.component.ts'	0	0	1	0	1	0	0
3	'..\\..\\Users\\joaoev\\ng-zorro-antd\\components\\alert\\alert.spec.ts'	1	0	0	0	0	1	0
359	'..\\..\\Users\\joaoev\\ng-zorro-antd\\components\\upload\\upload.component.ts'	0	0	1	0	1	1	0
360	'..\\..\\Users\\joaoev\\ng-zorro-antd\\components\\upload\\upload.spec.ts'	1	0	1	0	0	1	0
361	'..\\..\\Users\\joaoev\\ng-zorro-antd\\components\\watermark\\demo\\custom.ts'	0	0	1	0	0	0	0
362	'..\\..\\Users\\joaoev\\ng-zorro-antd\\components\\watermark\\watermark.component.ts'	0	0	1	0	1	1	0
363	'..\\..\\Users\\joaoev\\ng-zorro-antd\\components\\watermark\\watermark.spec.ts'	1	0	0	0	0	0	0

```
joaoev@skyline C:\..\..\angular-code-smells  main >
```

Code smells encontrados

- No total foram encontrados 603 code smells, abaixo está a quantidade por cada tipo:

DOM	IIC	LC	ANY	TMI	LF	EPC
239	34	133	4	48	143	2

Refatoração

- Cada um ficou com 20 smells para refatorar.
- Os tipos escolhidos foram:

DOM	LF	LC	TMI
10	10	10	10

Refatoração - DOM

- **Problema**:
  - O componente depende de APIs nativas do DOM, ignorando as abstrações e proteções do Angular.
- **Solução**:
  - Usar abstrações do Angular como Renderer2, bindings no template e encapsular acessos a nativeElement apenas quando estritamente necessário.

Refatoração - DOM

- **Antes**:

affix.component.ts

```
// Manipulação direta do DOM e acesso global
this.fixedEl.nativeElement.classList.add('ant-affix');
this.fixedEl.nativeElement.style.top = `${top}px`;
const target = typeof this.nzTarget === 'string'
  ? document.querySelector(this.nzTarget)
  : this.nzTarget || window;
```

Refatoração - DOM

- **Depois**:

affix.component.ts

```
// Encapsulamento do acesso ao nativeElement e uso de Renderer2 e DOCUMENT
private get fixedElement(): HTMLDivElement {
  return (this.fixedEl as { ['nativeElement']: HTMLDivElement })['nativeElement'];
}

private readonly document = inject(DOCUMENT);

const target = typeof this.nzTarget === 'string'
  ? this.document.querySelector(this.nzTarget)
  : this.nzTarget || this.document.defaultView || this.document.body;

this.renderer.addClass(this.fixedElement, 'ant-affix');
this.renderer.setStyle(this.fixedElement, 'top', `${top}px`);
```

Refatoração - LF

- **Problema**:
  - O componente faz muitas coisas ao mesmo tempo, ficando difícil de entender e modificar.
- **Solução**:
  - Extrair templates grandes para arquivos HTML separados
  - Mover lógica de negócio para serviços dedicados
  - Separar tipos e constantes em arquivos próprios

Refatoração - LF

- **Antes:**

code-editor.component.ts

```ts
@Component({
  selector: 'code-editor',
  template: `<div class="editor"></div>`,
  changeDetection: ChangeDetectionStrategy.OnPush
})
export class CodeEditorComponent {
  private editorInstance: unknown;

  ngAfterViewInit() {
    // Lógica de inicialização do Monaco Editor
    this.editorInstance = monaco.editor.create(...);
    // Lógica de atualização, resize, eventos, etc.
  }

  setValue(value: string) {
    this.editorInstance.setValue(value);
  }

  layout() {
    this.editorInstance.layout();
  }
}
```

Refatoração - LF

- **Depois:**

code-editor.component.ts

```ts
@Component({
  selector: 'code-editor',
  template: `<div class="editor"></div>`,
  changeDetection: ChangeDetectionStrategy.OnPush
})
export class CodeEditorComponent {
  private instanceService = inject(CodeEditorInstanceService);

  ngAfterViewInit() {
    this.instanceService.setup(/* ... */);
  }

  setValue(value: string) {
    this.instanceService.setValue(value);
  }

  layout() {
    this.instanceService.layout();
  }
}
```

code-editor-instance.service.ts

```ts
@Injectable({ providedIn: 'root' })
export class CodeEditorInstanceService {
  private editorInstance: unknown;

  setup(/* ... */) {
    // Toda a lógica de inicialização do Monaco Editor
  }

  setValue(value: string) {
    // Atualiza valor no editor
  }

  layout() {
    // Redimensiona editor
  }
}
```

Refatoração - LC

- **Problema**:
  - O componente faz muitas coisas ao mesmo tempo, ficando difícil de entender e modificar.
- **Solução**:
  - Extrair templates grandes para arquivos HTML separados
  - Simplificar métodos convertendo para arrow functions
  - organizar código e separar responsabilidades

Refatoração - LC

- **Antes**:

notificaão.conponent.ts

```
@Component({
  selector: 'nz-notification',
  template: `
    <div #animationElement class="ant-notification-
notice">
      @if (instance.template) {
        <ng-template
[ngTemplateOutlet]="instance.template!" />
      } @else {
        <div class="ant-notification-notice-
content">
          <!-- 70+ linhas de HTML aqui -->
        </div>
      }
    </div>
  `
})
export class NzNotificationComponent {
  onClick(event: MouseEvent): void {
    this.instance.onClick.next(event);
  }

  close(): void {
    this.destroy(true);
  }
}
```

Refatoração - LC

- **Depois**:

notificaão.conponent.ts

```
@Component({
  selector: 'nz-notification',
  templateUrl: './notification.component.html'
})
export class NzNotificationComponent {
  onClick = (event: MouseEvent): void => {
    this.instance.onClick.next(event);
  };

  close = (): void => {
    this.destroy(true);
  };
}
```

notificaão.conponent.html

```
<div #animationElement class="ant-notification-notice">@if (instance.template) {<ng-template
[ngTemplateOutlet]="instance.template!" />} @else {<div class="ant-notification-notice-
content"><!-- conteúdo compactado --></div>}</div>
```

Refatoração - TMI

- **Problema**:
  - Quando um componente angular tem muitas propriedades @input .
- **Solução**:
  - Criando interfaces typeScript para aguprar todas as propriedades relacionadas.
  - Diminuindo o número de inputs no componente.
  - Separar tipos e constantes em arquivos próprios

Refatoração - TMI

- **Antes**:

pagination.conponent.ts

```ts
@Component({
  selector: 'li[nz-pagination-options]',
  // ...
})
export class NzPaginationOptionsComponent implements OnChanges {
  @Input() size: 'default' | 'small' = 'default';
  @Input() disabled = false;
  @Input() showSizeChanger = false;
  @Input() showQuickJumper = false;
  @Input() total = 0;
  @Input() pageIndex = 1;
  @Input() pageSize = 10;
  @Input() pageSizeOptions: number[] = [];
  @Input() locale!: NzPaginationI18nInterface;

  // Métodos que usam essas propriedades
  get nzSize(): 'default' | 'small' {
    return this.size;
  }
  // ...
}
```

Refatoração - TMI

- **Depois**:

pagination.conponent.ts

```ts
@Component({
  selector: 'li[nz-pagination-options]',
  // ...
})
export class NzPaginationOptionsComponent implements OnChanges {
  // Agrupamento de inputs - apenas 3 propriedades
  @Input() nzState: PaginationOptionsState = {};
  @Input() nzOptions: PaginationOptionsDisplay = {};
  @Input() locale!: NzPaginationI18nInterface;

  // Getters para manter compatibilidade
  get nzSize(): 'default' | 'small' {
    return this.nzOptions.size ?? 'default';
  }

  get disabled(): boolean {
    return this.nzOptions.disabled ?? false;
  }

  get total(): number {
    return this.nzState.total ?? 0;
  }
  // ... outros getters
}
```

pagination.types.ts

```ts
export interface PaginationOptionsState {
  total?: number;
  pageIndex?: number;
  pageSize?: number;
  pageSizeOptions?: number[];
}

export interface PaginationOptionsDisplay {
  size?: 'default' | 'small';
  disabled?: boolean;
  showSizeChanger?: boolean;
  showQuickJumper?: boolean;
}
```

Resultados

- **Início:**
  - Code smells detectados: 603
  - Arquivos com code smells: 364
- **Final**:
  - Code smells detectados: 561
  - Arquivos com code smells: 324
- **Redução total**:
  - Code smells foram 42 e arquivos 40.

Dificuldades

- **Anaildo:**
  - Entender a estrutura do projeto.
  - Pouca experiência com angular e refatoração.
- **João**:
  - Entender a estrutura do projeto
  - Pouca experiência com o Angular
  - Entender o código dos componentes

Obrigado pela atenção!
Dúvidas?

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ