UNIVERSIDADE FEDERAL DO CEARÁ

Code smells em frameworks front-end

Disciplina: Qualidade de Software
Equipe: Sávio de Carvalho e Renan Soares
Projeto: ng-bootstrap
Fork: https://github.com/saviosoaresUFC/refac-code-smells-ng-bootstrap.git

Eu estou atualmente trabalhando na refatoração do seguinte code smell: Manipulação direto no **DOM**
Minhas principais dificuldades na remoção do code smell são: Se eu tirava o uso do *ElementRef* (que manipula o DOM) do componente e coloco em um serviço, ou tiro-o e não uso o *ElementRef* de jeito nenhum em nenhum lugar. Decidi separar e colocar em um serviço e não direto no componente, pois alguns componentes precisavam de acesso direto ao *ElementRef.nativeElement* para executar.
Eu estou usando os seguintes métodos de refatoração para remover o code smell: Movi a lógica de manipulação do DOM e a injeção de ElementRef para uma classe auxiliar (como por exemplo: NgbModalWindowDomHelper), tornando o componente principal mais limpo. Em alguns casos, criei *wrappers* ou métodos seguros (ex: o método setClasses()) para substituir o acesso direto à nativeElement.classList. Com isso, garanti que o Renderer2 fosse usado sempre que possível dentro dos novos serviços de baixo nível para manipulação de classes, estilos e atributos, pois ele é a API agnóstica de plataforma do Angular.

Eu estou atualmente trabalhando na refatoração do seguinte code smell: Uso excessivo de **AnyType**
Minhas principais dificuldades na remoção do code smell são: Foi achar qual a tipagem certa que eu tinha que passar no construtor, mas no fim achei a tipagem certa.
Eu estou usando os seguintes métodos de refatoração para remover o code smell: Substitui any por um tipo definido (ex: troquei value: any por value: NgbTimeStruct | null). Em vez de usar T = any, restringi o tipo genérico a um tipo mais seguro, como T extends object ou T extends unknown, em funções e classes. Usei unknown em vez de any em situações onde o tipo é desconhecido. O unknown é mais seguro porque força o desenvolvedor a realizar uma checagem de tipo explícita antes de interagir com o valor.

UNIVERSIDADE FEDERAL DO CEARÁ

Eu estou atualmente trabalhando na refatoração do seguinte code smell: Herança em vez de Composição **(IIC)**
Minhas principais dificuldades na remoção do code smell são: foi desacoplar componentes que dependiam de lógica implícita na classe pai (ex geração automática de menus em NgbOverviewPage), também muito cuidado ao atualizar as referências sem quebrar funcionalidades.
Eu estou usando os seguintes métodos de refatoração para remover o code smell: Substituição de herança por injeção de dependência (utilizando inject (COMPONENT_DATA) para acessar dados, resumindo, a remoção de herança desnecessária em todos os componentes, tornando-os classes independentes (POJOs) que injetam seus próprios dados

Eu estou atualmente trabalhando na refatoração do seguinte code smell: Large File **(LF)**
Minhas principais dificuldades na remoção do code smell são: Lidar com as classes que possuíam muitas responsabilidades em um lugar só, misturando lógica, com animação, testes e manter a integridade dos testes ao extrair lógica para novos serviços ou módulos
Eu estou usando os seguintes métodos de refatoração para remover o code smell: Extract Module, Extract Service, Split File. Dividindo arquivos de teste grandes em suítes menores, Extração de lógica de fábrica (criação de componentes) para serviços dedicados, aplicando o separation of concerns, sempre garantindo a coesão.