**Introdução:**

No desenvolvimento do Activity Provider: Simulador de Cenários de Exposição Radiológica, identifiquei a ocorrência do antipadrão "The Blob" (God Object). Este antipadrão surge quando uma única classe concentra demasiadas responsabilidades, violando o Princípio da Responsabilidade Única (SRP) e dificultando a manutenção e escalabilidade do sistema. A presença deste antipadrão resultava num código difícil de escalar e propenso a erros, comprometendo o desempenho e a integridade do sistema

**Antes da refatorização:**

A classe ScenarioFactory estava altamente acoplada e acumulava responsabilidades que deveriam ser distribuídas por componentes especializados. Esta classe era responsável por:

* Métodos presentes na classe com sobrecarga de responsabilidades.
* Criar o ambiente e os equipamentos dos cenários médicos e industriais.
* Gerir a lógica de configuração e processamento de pedidos REST.
* Executar análises sobre os cenários.

**Após a refatorização:**

Embora o sistema já estivesse estruturado com os padrões Abstract Factory, Adapter e Strategy, a classe ScenarioFactory concentrava demasiadas responsabilidades, tornando o código difícil de manter e propenso a alto acoplamento.

**Principais mudanças realizadas:**

1. Descentralização das responsabilidades da ScenarioFactory

**Antes:** ScenarioFactory geria tanto a criação de cenários como a lógica de configuração e análise, tornando-se uma classe monolítica.

**Depois:**

* Criação de cenários isolada em ScenarioCreator
* Configuração delegada para ConfigManager, reduzindo a complexidade.

1. Separação da lógica de criação usando Abstract Factory

**Antes:** ScenarioFactory possuía dependências diretas com os diferentes tipos de cenários.

**Depois:** A separação total das fábricas (MedicalScenarioFactory e IndustrialScenarioFactory) garante independência e modularidade, permitindo a adição de novos cenários sem modificar a estrutura existente.

1. Melhor uso do Adapter para desacoplar a interface REST

**Antes:** ScenarioFactory tinha dependências diretas com o servidor REST.

**Depois:** O servidor REST interage apenas com ScenarioCreator e ConfigManager, reduzindo o acoplamento e simplificando a interface REST.

1. Modularização da lógica de configuração

**Antes:** A configuração de cenários estava dispersa dentro da ScenarioFactory, dificultando a reutilização e a manutenção.

**Depois:** A classe ConfigManager centraliza toda a configuração da atividade, facilitando futuras expansões sem impacto na lógica principal.

1. Ajuste do Strategy Pattern para análise de métricas

**Antes:** AnalysisContext aplicava diferentes estratégias (MedicalAnalysisStrategy e IndustrialAnalysisStrategy), mas estava acoplado ao fluxo do servidor REST.

**Depois:** AnalysisContext tornou-se independente, permitindo que as estratégias de análise sejam utilizadas de forma modular, sem impacto no restante sistema.

**Conclusão:**

A refatorização não alterou os padrões já existentes, mas aperfeiçoou sua implementação para garantir um sistema mais modular, escalável e sustentável. A correção do antipadrão The Blob foi essencial para tornar o Activity Provider mais flexível e melhor preparado para futuras integrações na Inven!RA.

**O projeto está disponível em:**

<https://github.com/lsantos1970/ActiveProvideProject.git>

**Anexos**

1. Código
2. Diagramas UML:
   * Diagrama de Componentes AntiPadrão
   * Diagrama de Sequência AntiPadrão