Relatório de Análise Arquitetural: Requisitos Significativos (ASRs) do Analisador de Código

1. Contexto da Aplicação

O sistema em questão é um Analisador de Código (Code Analyzer) caracterizado pelo processamento assíncrono e por uma arquitetura que integra Spring Batch, Spring MVC (para interface de utilizador) e um mecanismo de persistência.

A criticidade da arquitetura reside na necessidade de atender a Requisitos Arquitetónicos Significativos (ASRs), nomeadamente nas dimensões de Desempenho (Performance) e Fiabilidade (Reliability), essenciais para o processamento eficiente e robusto de lotes de código-fonte.

2. Metodologia de Priorização: A Árvore de Utilidade

Para simplificar e orientar as decisões arquiteturais, foi construída uma Árvore de Utilidade (Utility Tree). Esta ferramenta permite categorizar e priorizar os ASRs através da avaliação de dois fatores chave:

- **Valor de Negócio (VN):** O quão importante o requisito é para o sucesso do negócio/produto.
- **Impacto Arquitetural (IA):** O grau em que o requisito força alterações significativas ou complexas na estrutura do sistema.

A classificação utilizada é H (Alto), M (Médio), L (Baixo).

Utility Tree Structure

#	Atributo de Qualidade	Refinamento	Requisito Arquitetural Significativo (ASR)	Valor de Negócio (VN)	Impacto Arquitetural (IA)
1	**Performance (Desempenho)**	Throughput (Capacidade)	O sistema deve processar um arquivo ZIP contendo 1000 ficheiros .java (50k Linhas de Código total) em menos de 5 minutos, utilizando a lógica de *parsing* Eclipse AST.	H	H
2	**Performance (Desempenho)**	Response Time (Tempo de Resposta)	Manter o tempo médio de resposta para visualização do estado do *job* no UI (Spring MVC/Thymeleaf) abaixo de 1 segundo.	M	M
3	**Reliability (Fiabilidade)**	Robustness (Robustez)	Se a extração de métricas falhar (ex: erro de *parsing* do AST) para um ficheiro .java, o *job* Spring Batch deve continuar o processamento dos restantes, registando o erro.	H	M
4	**Maintainability (Manutenção)**	Modifiability (Modificabilidade)	A lógica de extração de métricas (cbo, fan in, fan out) deve ser isolada de forma a permitir a atualização da biblioteca Eclipse AST ou a adição de novas métricas sem modificar os passos principais do Spring Batch.	M	L
5	**Availability (Disponibilidade)**	Uptime (Tempo de Atividade)	O serviço de *backend* assíncrono deve manter 99.9% de disponibilidade durante os períodos de maior utilização.	H	M
6	**Performance (Desempenho)**	Resource Utilization (Utilização de Recursos)	A extração de métricas (passo 3), que é intensiva em CPU, deve gerir a utilização da memória para evitar `OutOfMemoryErrors` durante o processamento de grandes lotes.	H	H

Legend: Atributo de Qualidade / Refinamento/ Requisito Arquitetural Significativo (ASR)/ H/ M/ L

3. Foco nos Requisitos Mais Significativos (H, H)

A análise da Árvore de Utilidade direciona o foco arquitetural para os requisitos classificados como (H, H), ou seja, aqueles que possuem Alto Valor de Negócio (H) e Alto Impacto Arquitetural (H).

Estes representam as metas de maior prioridade e que exigirão as decisões arquiteturais mais fundamentais. No presente caso, os ASRs mais críticos são:

- **Performance/Throughput (ASR #1):** O sistema deve processar um arquivo ZIP contendo 1000 ficheiros .java (50k Linhas de Código total) em menos de 5 minutos, utilizando a lógica de parsing Eclipse AST.
- **Performance/Resource Utilization (ASR #6):** A extração de métricas (intensiva em CPU) deve gerir a utilização da memória para evitar OutOfMemoryErrors durante o processamento de grandes lotes.

Implicações Arquiteturais dos ASRs (H, H)

O Alto Impacto Arquitetural destes requisitos (IA = H) decorre do facto de o core do sistema ser a capacidade de processar grandes volumes de dados de código-fonte de forma eficiente. O seu cumprimento ditará decisões cruciais sobre:

- **Configuração de Recursos Assíncronos:** Definição otimizada do thread pool para operações assíncronas.
- **Otimização do Ambiente de Execução:** Gestão dos limites de memória do JVM e do container Spring Boot.
- **Estruturação do Spring Batch:** Avaliação da adequação de padrões como chunking para gerir a complexidade da extração AST em cada passo.

4. Equilíbrio entre Fiabilidade e Manutenção

A análise da Árvore de Utilidade também oferece uma perspetiva importante sobre o equilíbrio entre Fiabilidade e Manutenção:

Reliability/Robustness (ASR #3): Classificado como (H, M).

- **Valor de Negócio (H):** É crucial que a falha na extração de métricas de um único ficheiro não comprometa a execução total do job, permitindo o registo do erro e a continuidade do processamento dos restantes.
- **Impacto Arquitetural (M):** Embora o Spring Batch ofereça mecanismos nativos de skip e failover, a implementação correta da lógica de tratamento de exceções nos passos de extração de métricas é mandatória para garantir esta robustez.

Maintainability/Modifiability (ASR #4): Classificado como (M, L).

- Este requisito visa isolar a lógica de extração de métricas (cbo, fan in, fan out) para permitir atualizações de bibliotecas (ex: Eclipse AST) ou a adição de novas métricas.
- Embora seja de valor Moderado (M) para o projeto a longo prazo, o seu Impacto Arquitetural é Baixo (L) na arquitetura global do pipeline do Spring Batch.

5. Conclusão e Tomada de Decisão Arquitetural

A Árvore de Utilidade atua como um mapa de prioridades. O arquiteto está capacitado a fazer trade-offs informados e a justificar cada decisão.

Se surgir um conflito entre otimizar o Throughput (ASR #1) e facilitar a Modificação da Lógica de Métricas (ASR #4), a prioridade deverá recair sobre o throughput devido à sua classificação superior (H, H) face ao (M, L).

O foco na mitigação dos riscos de Alto Impacto e Alto Valor de Negócio (H, H) garante que o desenho do sistema satisfaça as metas de desempenho mais críticas definidas para a aplicação.