Detecção de Bad Smells e Refatoração Segura

Disciplina: Teste de Software
Trabalho: Bad Smells
Nome: Matheus Dias Mendes
Matrícula:
Data: 09/11/2025

Análise de Smells

No código original do ReportGenerator, foram encontrados alguns Bad Smells que indicam problemas de design e que podem afetar a manutenção e a facilidade de testes.

Método Longo

O método generateReport foi identificado como um exemplo de método longo, contendo múltiplas responsabilidades, o que torna o código difícil de ler, entender e manter. Esse método realiza:

- A construção do cabeçalho do relatório
- O processamento dos itens a serem incluídos no relatório
- O cálculo do total
- A adição do rodapé

A complexidade do método dificulta a criação de testes unitários eficazes, pois o comportamento de geração de relatórios depende de vários fatores e ações dentro do mesmo bloco de código.

Duplicação de Código

O código duplicado foi identificado nas partes que tratam a geração de relatórios no formato CSV e HTML. O mesmo bloco de lógica para geração de itens foi replicado para os dois formatos de saída, o que torna o código difícil de manter. Qualquer mudança no formato de geração de um desses tipos exigiria alterações em ambos os lugares.

A duplicação torna os testes mais difíceis de manter, pois qualquer alteração no comportamento de um tipo de relatório precisaria ser feita em múltiplas vezes, aumentando a possibilidade de falhas.

Complexidade Cognitiva Alta

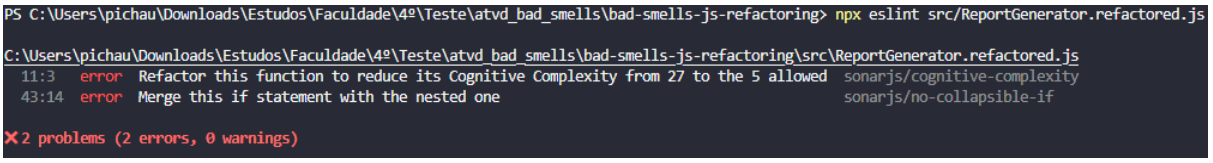
O código original apresenta uma alta complexidade cognitiva, especialmente no loop onde a lógica para diferentes tipos de usuários e formatos de relatório é combinada com verificações de valores e estilo. Isso torna o código difícil de entender e, por isso, aumenta a chance de introdução de bugs ao tentar alterar qualquer parte do código.

A alta complexidade dificulta entender os futuros cenários possíveis durante os testes, o que pode resultar em falhas não detectadas em alguns casos.

Relatório da Ferramenta

A ferramenta ESLint foi utilizada para realizar uma análise estática do código e identificar problemas de qualidade no código original. O resultado fornecido pela ferramenta ajudou a identificar vários problemas quais seriam os problemas como por exemplo o número de complexidade do codigo

Antes da Refatoração:



Processo de Refatoração

A refatoração focou em reduzir a complexidade do método generateReport, melhorar a legibilidade do código e eliminar a duplicação de lógica. O smell mais crítico que escolhi citar aqui foi o Método Longo, por causa que ele teve uma complexidade maior para se resolver.

Antes:

```
generateReport(reportType, user, items) {
  let report = '';
  let total = 0;
  if (reportType === 'CSV') {
    report += 'ID,NOME,VALOR,USUARIO\n';
  } else if (reportType === 'HTML') {
    report += '<html><body>\n';
    report += '<h1>Relatório</h1>\n';
    report += `<h2>Usuário: ${user.name}</h2>\n`;
    report += '<table>\n';
    report += '<tr><th>ID</th><th>Nome</th><th>Valor</th></tr>\n';
  }
  for (const item of items) {
    if (user.role === 'ADMIN') {
      if (item.value > 1000) {
        item.priority = true;
      }
      if (reportType === 'CSV') {
        report += `${item.id},${item.name},${item.value},${user.name}\n`;
        total += item.value;
      } else if (reportType === 'HTML') {
        const style = item.priority ? 'style="font-weight:bold;"' : '';
        report += `<tr ${style}><td>${item.id}</td><td>${item.name}</td><td>${item.value}</td></tr>\n`;
        total += item.value;
      }
```

```
    } else if (user.role === 'USER') {
      if (item.value <= 500) {
        if (reportType === 'CSV') {
          report += `${item.id},${item.name},${item.value},${user.name}\n`;
          total += item.value;
        } else if (reportType === 'HTML') {
          report += `<tr><td>${item.id}</td><td>${item.name}</td><td>${item.value}</td></tr>\n`;
          total += item.value;
        }
      }
    }
  }
  if (reportType === 'CSV') {
    report += '\nTotal,,\n';
    report += `${total},,\n`;
  } else if (reportType === 'HTML') {
    report += '</table>\n';
    report += `<h3>Total: ${total}</h3>\n`;
    report += '</body></html>\n';
  }
  return report.trim();
}
```

Depois:

```
generateReport(reportType, user, items) {
  let header = this.generateHeader(reportType, user);
  let body = "";

  for (const item of items) {
    body = this.generateItemLine(item, reportType, user);
  }

  let footer = this.generateFooter(
    reportType,
    this.calculateTotal(items, user)
  );
  return `${header}${body}${footer}`.trim();
}
```

Conclusão

A refatoração dos Bad Smells resultou em um código mais legível, modular e manutenível, além de reduzir a complexidade cognitiva. Com a ajuda do ESLint, consegui identificar e corrigir problemas que poderiam ser difíceis de encontrar apenas por análise manual, como a alta complexidade e a duplicação de código.