AMANDA SOUZA MACEDO BACELLI

Diretrizes para refatoração de *software* utilizando gamificação

São Paulo

2023

AMANDA SOUZA MACEDO BACELLI

Diretrizes para refatoração de *software* utilizando gamificação

Versão Revisada

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a conclusão do curso de MBA em Tecnologia de *Software*.

Área de Concentração: Tecnologia de *Software*

Orientador: Prof. Dr. Sergio Roberto de Mello Canovas

São Paulo

2023

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

Catalogação-na-publicação

Bacelli, Amanda
Diretrizes para refatoração de software utilizando gamificação / A. Bacelli -- São Paulo, 2023.
129 p.

Monografia (MBA em Tecnologia de Software) - Escola Politécnica da Universidade de São Paulo. PECE – Programa de Educação Continuada em Engenharia.

1.Refatoração de Software 2.Gamificação 3.Qualidade de Software 4.Diretrizes de Gamificação I.Universidade de São Paulo. Escola Politécnica. PECE – Programa de Educação Continuada em Engenharia II.t.

Nome: BACELLI, Amanda Souza Macedo

Título: Diretrizes para refatoração de software utilizando gamificação

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a conclusão do curso de MBA em Tecnologia de *Software*.

Aprovado em: 13 / 02 / 2023

Banca Examinadora

Prof(a). Dr(a). Sergio Roberto de Mello Canovas

Instituição: Pece

Julgamento: Aprovada

Prof(a). Dr(a). Kechi Hirama

Instituição: Escola Politécnica da Universidade de São Paulo

Julgamento: Aprovada

Prof(a). Dr(a). Rogério Rossi

Instituição: PECE-USP

Julgamento: Aprovada

DEDICATÓRIA

Dedico este trabalho a Deus, toda honra e toda glória seja dada a Ele.

RESUMO

BACELLI, Amanda S. M.. **Diretrizes para refatoração de *software* utilizando gamificação**. 2023. 129. Monografia (MBA em Tecnologia de *Software*). Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo. São Paulo. 2023.

Com o crescimento de padrão de trabalho remoto e/ou híbrido, as empresas tradicionais centradas em trabalho presencial estão transformando suas equipes em equipes distribuídas, compostas por colaboradores geograficamente separados. No contexto de desenvolvimento de *software*, é muito comum adicionar novas funcionalidades em projetos desenvolvidos por outras equipes, muitas vezes sendo necessário realizar refatorações ao longo do desenvolvimento. A refatoração é um modo utilizado para melhorar a qualidade do *software*. Refatorar compreende a aplicação de uma série de transformações, muitas vezes pequenas, mas que melhoram o código-fonte quando comparado à versão inicial, contudo, sem alterar seu comportamento. Com a evolução da tecnologia, aumentou a busca de ferramentas e métodos que auxiliam as pessoas a solucionarem problemas com o engajamento empresarial de maneira atrativa, surgindo assim o conceito de gamificação. Este trabalho tem como objetivo apresentar diretrizes a serem aplicadas durante a etapa de codificação de *software*, de forma a auxiliar a qualidade da refatoração do código, incentivar e motivar a equipe a executar a refatoração, utilizando técnicas de gamificação para engajar a equipe durante as iterações do projeto. Um estudo de caso avalia a aplicabilidade das diretrizes de técnicas de gamificação e o impacto no engajamento da equipe quanto à prática de refatoração em empresas de desenvolvimento de *software*. A aplicação das diretrizes trouxe impactos positivos quanto ao engajamento da equipe e demonstrou potencial para auxiliar na qualidade da refatoração de *software*.

Palavras-chave: Refatoração de *software*. Gamificação. Qualidade de *software*. Diretrizes de gamificação.

ABSTRACT

BACELLI, Amanda S. M. **Diretrizes para refatoração de *software* utilizando gamificação**. 2023. 129. Monografia (MBA em Tecnologia de *Software*). Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo. São Paulo. 2023.

With the growth of remote and/or hybrid working patterns, traditional companies centered on face-to-face work are transforming their teams into distributed teams composed of geographically separated collaborators. In the context of software development, it is very common to add new functionality to projects developed by other teams, often requiring refactoring throughout the development process. Refactoring is a way of improving the quality of software. Refactoring comprises the application of a series of transformations, often small, but which improve the source code when compared to the initial version, however, without changing its behavior. With the evolution of technology, the search for tools and methods that help people solve problems with business engagement in an attractive way has increased, thus emerging the concept of gamification. This paper aims to present guidelines to be applied during the software coding stage in order to assist the quality of code refactoring, encourage and motivate the team to perform refactoring, using gamification techniques to engage the team during project iterations. A case study evaluates the applicability of gamification technique guidelines and the impact on team engagement regarding the refactoring practice in software development companies. The application of the guidelines had a positive impact on team engagement and demonstrated potential to assist in the quality of software refactoring.

Keywords: Software refactoring. Gamification. Software quality. Gamification Guidelines.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AHF	Attribute Hiding Factor.
AIF	Attribute Inheritance Factor.
CF	Coupling Factor.
IDE	Integrated Development Environment.
MHF	Method Hiding Factor.
MIF	Method Inheritance Factor.
MOOD	Metrics for Object Oriented Design.
PBL	Points, Badges and Leaderboard.
PF	Polymorphism Factor.
PR	Pull request.
QMOOD	Quality Model For Object Oriented Design.
XP	Extreme Programming.

SUMÁRIO

1. INTRODUÇÃO

1.1. Motivações

Diante do cenário pandêmico, iniciado no Brasil em meados de março de 2020 e que se estendeu até o início de 2022, as empresas foram obrigadas a adaptar o ambiente de trabalho para a modalidade de trabalho *home office*. Isso acarretou em contratações de pessoas de diversas partes do país, assim como também de outros países. Com o crescimento de padrão de trabalho remoto e/ou híbrido, as empresas tradicionais centradas em trabalho presencial estão transformando suas equipes em equipes distribuídas, compostas por colaboradores geograficamente separados (ANCKËN, 2021).

No contexto de desenvolvimento de *software*, é muito comum adicionar novas funcionalidades em projetos desenvolvidos por outras equipes. Muitas vezes o projeto não foi pensado de forma a ser resiliente para novas iterações, pois os requisitos de um *software* mudam com frequência. Além disso, os clientes regularmente não têm uma ideia precisa do que querem, resultando em projetos que não são mais necessários ou com demanda de serem alterados devido à mudança de planos e desejos dos clientes (VALENTE, 2020).

A refatoração é uma técnica utilizada para aperfeiçoar um projeto e também para melhorar a qualidade do *software*. A refatoração é um modo utilizado para melhorar a qualidade do *software*. Refatorar compreende a aplicação de uma série de transformações, muitas vezes pequenas, mas que melhoram o código-fonte quando comparado à versão inicial, contudo, sem alterar seus resultados (FOWLER, 2019). Por se tratar de uma série de transformações, pode gerar um conceito de que refatorar demanda um tempo maior dos desenvolvedores, principalmente quando a equipe trabalha com metodologias ágeis cuja característica principal para Valente (2020) é a utilização de ciclos curtos e iterativos de desenvolvimento, de forma que o sistema implementado de forma gradativa, tendo como início a necessidade imediata do cliente.

Muitas empresas se deparam com a problemática de engajar os colaboradores para obtenção de resultados, sendo essencial que os colaboradores se mostrem

engajados e motivados a realizarem as atividades para atingir os objetivos (FORMANSKI, 2016).

Com a evolução da tecnologia, aumentou a busca de ferramentas e métodos que auxiliam as pessoas a solucionarem problemas com o engajamento empresarial de maneira atrativa, surgindo assim o conceito de gamificação. É possível que este conceito seja aplicado em qualquer ambiente, e equivale à utilização de mecanismos de jogos orientados a alcançar o objetivo, resolver problemas práticos ou criar engajamento entre um determinado público (FORMANSKI, 2016).

É difícil criar um sistema “correto de primeira”. Nem sempre é possível antecipar o sistema de acordo com os requisitos futuros. Em vez disso, se faz necessário implementar apenas os requisitos atuais, e, então, refatorar e expandir o sistema, implementando novos requisitos. Essa é a essência das agilidades iterativa e incremental (MARTIN, 2011).

Sistemas de *software* possuem um ciclo de vida. No decorrer do tempo, os sistemas envelhecem, a sua qualidade interna muda e há uma evolução do sistema. Segundo Valente (2020), Meir Lehman enunciou um conjunto de leis empíricas sobre esse tema, que ficaram conhecidas como Leis da Evolução de *Software* ou simplesmente Leis de Lehman, sendo elas:

> Um sistema de *software* deve ser continuamente mantido para se adaptar ao seu ambiente. Esse processo deve continuar até o ponto em que se torna mais vantajoso substituí-lo por um sistema completamente novo. À medida que um sistema sofre manutenções, sua complexidade interna aumenta e a qualidade de sua estrutura deteriora-se, a não ser que um trabalho seja realizado para estabilizar ou evitar tal fenômeno.

Considerando os problemas mencionados, a refatoração é uma etapa importante da codificação para a manutenção e existência do *software*.

1.2. Objetivo

Este trabalho tem como objetivo elaborar um conjunto de diretrizes de gamificação que podem ser utilizados durante a etapa de codificação de *software*, de forma a

auxiliar a qualidade da refatoração do código, incentivar e motivar a equipe a executar a refatoração, utilizando técnicas de gamificação para engajar a equipe durante as iterações do projeto.

Como parte da avaliação das diretrizes, é apresentado um estudo de caso de uma empresa que possui equipes de desenvolvimento de *software* distribuídas geograficamente.

1.3. Justificativas

O aumento do trabalho remoto levou empresas a distribuir seus processos de desenvolvimento de *software*, afetando assim a forma como o *software* é projetado, desenvolvido e entregue ao cliente, representando assim desafios adicionais únicos (MAJDENBAUM; CHAVES, 2020).

Como consequência do trabalho com equipes distribuídas, em geral observa-se que há mais de uma equipe adicionando novas funcionalidades em um único *software*, o que ocasiona conflitos de ideias, linhas de raciocínio, falta de uso de padrões de projeto, dentre outros aspectos relacionados ao processo de implementação da solução. Assim, durante as iterações do processo de desenvolvimento de *software*, pode acontecer um aumento de complexidade no código afetando seus aspectos de manutenibilidade.

A busca para engajar o time com o objetivo de melhorar a qualidade de códigos legados com técnicas de refatoração é um grande desafio. Atualmente, a utilização da gamificação no ramo empresarial tem crescido, visto que a gamificação faz com que as tarefas sejam mais divertidas, produtivas e colaborativas (RODRIGUES *et al.*, 2021).

A gamificação tem como objetivo influenciar positivamente o comportamento humano, não sendo apenas premiação por pontos e emblemas. É baseada em conceitos da psicologia humana e da ciência comportamental (FORMANSKI, 2016)

Foram encontrados trabalhos que utilizam ferramentas gamificadas durante a refatoração do código, sendo eles:

- Um jogo de refatoração. Estudando o impacto da ludificação na refatoração de *software* de Elezi *et al*. (2016);
- *CleanGame*: Gamificando a identificação de *Code Smells* de Santos *et al*. (2019);
- Impacto da gamificação no processo de revisão de código - um estudo experimental de Khandelwal *et al*. (2017).

Dentre eles, não foram encontradas diretrizes sobre como pode ser feita a construção da ferramenta e quais as diretrizes para se implementar a gamificação de forma a aumentar o engajamento dos colaboradores com a refatoração.

Isto posto, este trabalho busca contribuir na busca de soluções para o aumento da qualidade da refatoração do código, incentivar e motivar a equipe quanto a essa prática utilizando a gamificação. O uso da gamificação nas etapas de codificação do *software*, seja qual for o processo de desenvolvimento, pode aumentar o interesse dos desenvolvedores pela refatoração de código bem como proporcionar um ambiente no qual o desenvolvedor pode evoluir tecnicamente, contribuir para o crescimento técnico de outras equipes e praticar o fornecimento de *feedbacks* quanto à prática de refatoração. A gamificação é adaptável conforme o tipo de ambiente, cultura da organização e modelos de equipe, podendo ser uma solução plausível para auxiliar na prática da refatoração e, consequentemente, melhorar a qualidade do *software*.

1.4 Método de Pesquisa

O método de pesquisa deste trabalho é baseado nas etapas abaixo:

1. Revisão bibliográfica da literatura relacionada ao tema proposto de maneira ampla, utilizando palavras chaves tais como: refatoração, desenvolvimento de *software* distribuído, processos de desenvolvimento de *software* e gamificação em empresas com bases nos sites *Google Scholar* e *Scielo*, buscando eliminar referências antigas;

2. Elaboração de diretrizes de gamificação baseadas em um conjunto de práticas possíveis para melhoria da qualidade de refatoração de *software*

utilizando técnicas de gamificação e métricas com base nos princípios e valores culturais da equipe;

3. Realização de um estudo de caso exploratório com cenário real para verificar o engajamento dos colaboradores ao que tange à refatoração de *software* em um cenário que antecede a aplicação das diretrizes de gamificação;
4. Verificação do impacto das diretrizes de gamificação quanto à qualidade de *software* e engajamento dos colaboradores, através da análise do cenário antes e depois da implantação das diretrizes de gamificação.

1.5 Estrutura do Trabalho

O Capítulo 1 INTRODUÇÃO apresenta as motivações, o objetivo, as justificativas, método de pesquisa e a estrutura do trabalho.

O Capítulo 2 REVISÃO BIBLIOGRÁFICA apresenta a fundamentação teórica e resultados históricos.

O Capítulo 3 DIRETRIZES PARA GAMIFICAÇÃO apresenta a identificação do cenário e os pontos de melhoria na refatoração de código e elaboração de diretrizes de gamificação composta por um conjunto de práticas possíveis para melhoria da qualidade de refatoração de *software.*

O Capítulo 4 ESTUDO DE CASO apresenta o estudo de caso de um cenário real.

O Capítulo 5 ANÁLISE DE RESULTADOS analisa os resultados com base na aplicação das diretrizes, obtidas através do estudo de caso.

O Capítulo 6 CONSIDERAÇÕES FINAIS apresenta as considerações finais, os desafios e impedimentos e trabalhos futuros.

REFERÊNCIAS relaciona as referências utilizadas na elaboração do trabalho.

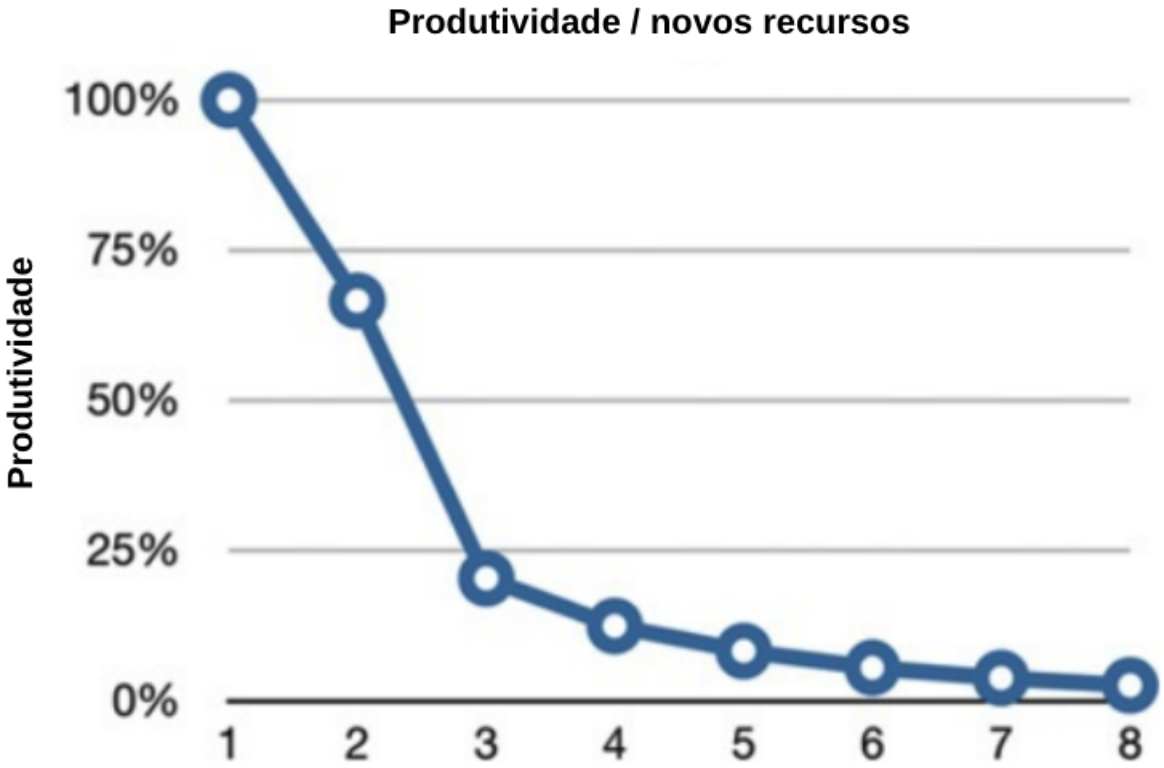
2. REVISÃO BIBLIOGRÁFICA

2.1. Refatoração

Conforme o desenvolvimento evolui, as falhas começam a surgir indicando que o código precisa ser substituído (ALOMAR *et al.*, 2021). À medida que o sistema se torna mais complexo, a capacidade de entendimento e o tempo gasto para compreender o sistema aumenta. Logo, eleva-se o tempo para se realizar uma tarefa de programação. Quanto mais claro o código se tornar, menos tempo outros desenvolvedores terão de despender para compreendê-lo, reduzindo defeitos e custo de manutenção (MARTIN, 2011).

A manutenção do *software* é essencial durante o processo de evolução. De acordo com o estudo de Lacerda *et al.* (2020), 50% a 80% dos custos de *software* estão relacionados às atividades de manutenção: reparação de falhas de projeto e implementação, adaptação do *software* a um ambiente diferente (*hardware* ou plataforma), e adicionar ou modificar funcionalidades. O esforço para adicionar novos recursos aumenta conforme o tempo e a complexidade do código, enquanto a produtividade reduz. A Figura 1 representa a relação entre a produtividade e cada *release* (adição de novos recursos).

Figura 1 - Relação entre produtividade e novos recursos



Fonte: Martin (2018) - Traduzido pela autora

A refatoração é uma ferramenta que pode ser usada para tornar o *software* mais fácil de se compreender, auxilia a encontrar problemas e também coopera para que o desenvolvedor realize a alteração do código mais rapidamente. Consiste em modificar um sistema de *software* por meio de pequenas alterações de modo que não altere seu comportamento (FOWLER, 2019).

O esforço e o custo para se manter essa disciplina de refatoração, quando bem aplicada, melhora a qualidade de *software*, o projeto e a legibilidade, reduzindo a introdução de falhas (FOWLER, 2019). “O objetivo é rever variáveis, classes e métodos de forma que facilite futuras adaptações e melhore a compreensão de que a refatoração pode melhorar diferentes aspectos da qualidade de software como manutenibilidade, extensibilidade, reusabilidade, etc.” (ALIZADEH *et al.,* 2020, Tradução Nossa).

Durante a refatoração, os desenvolvedores sintetizam e analisam uma grande quantidade de código para identificar as características inadequadas ou indesejáveis (como por exemplo código duplicado), identificar possíveis soluções e executar tarefas potencialmente complexas. “Durante esse processo, o desenvolvedor deve manter o comportamento do código, melhorar o entendimento do modelo do código e a compreensão do sistema” (MEALY *et al*., 2007, Tradução Nossa)

Segundo Valente (2020), existem dois momentos em que é realizada a refatoração, categorizados como planejada e oportunista.

- **Refatoração oportunista**: é realizada durante uma tarefa de programação, quando se está corrigindo uma falha ou implementando uma nova funcionalidade (VALENTE, 2020). Segundo Fowler (2019), não se planeja um tempo específico para a tarefa de refatoração, mas se faz como parte da tarefa de programação e pode ser classificada em subcategorias:
    - **Refatoração preparatória**: Realizada antes de iniciar a tarefa de programação de modo a deixar mais fácil a alteração;
    - **Refatoração para compreensão**: Realizada para entender melhor o código e passar a compreensão que o desenvolvedor possui para o código;
    - **Refatoração para coleta de lixo**: Realizada quando são identificados trechos de códigos desnecessários e/ou podem ser melhorados.
- **Refatoração planejada**: São mudanças mais robustas, demoradas e complexas que exigem tempo e dedicação além das tarefas de desenvolvimento. Muitas vezes ocorre por negligência de práticas de refatoração por muito tempo, porém devem ser casos raros (VALENTE, 2020).

2.1.1. Estágios da refatoração

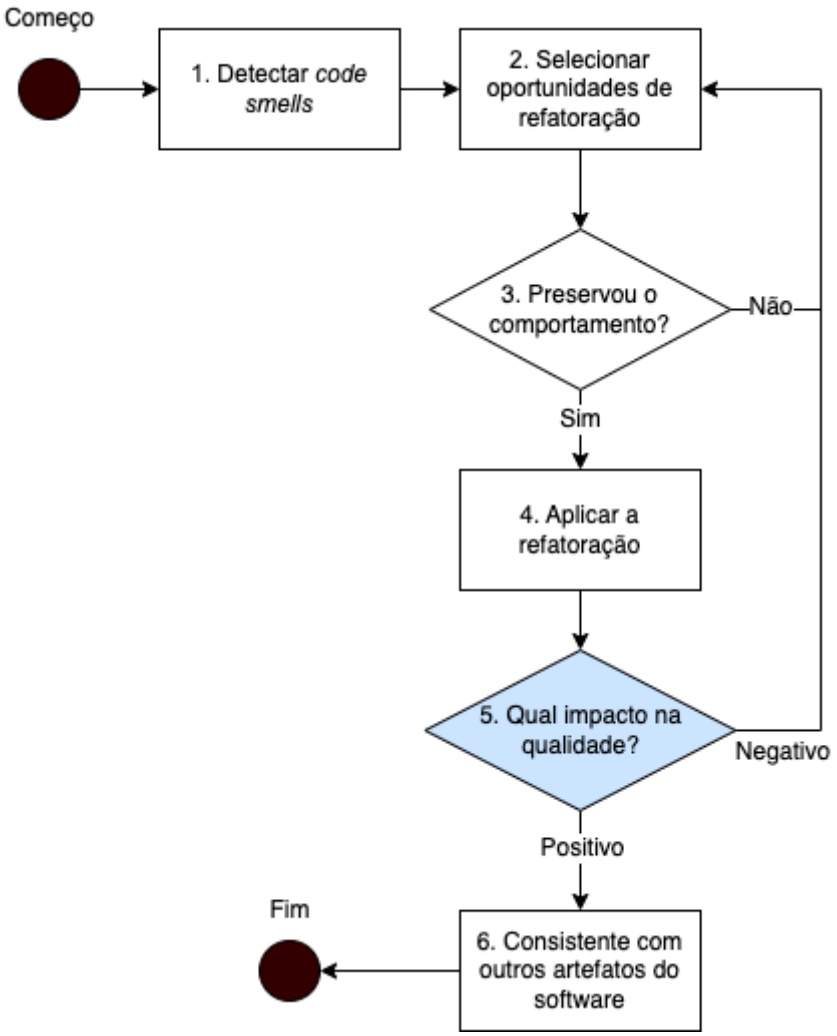
Para executar a refatoração, Kaur *et al.*(2019) citam um conjunto de estágios típicos, identificados durante a pesquisa:

- Identificar locais de código onde o *software* deve ser refatorado;
- Determinar as atividades apropriadas de refatoração a serem aplicadas;

- Garantir que as atividades de refatoração aplicadas preservem o comportamento do *software*;
- Aplicar as atividades de refatoração selecionadas;
- Avaliar o impacto das atividades de refatoração na qualidade do *software*;
- Manter a consistência entre o *software* refatorado e outros artefatos de *software*, tais como especificações de requisitos, documentos de projeto, casos de teste, documentação, etc;

A Figura 2 mostra o fluxo dos estágios.

Figura 2 - Fluxo de estágios típicos de refatoração



Fonte: Kaur *et al.* (2019) - Traduzido e adaptado pela autora.

A seguir são apresentados mais detalhes de cada estágio.

2.1.1.1. Identificar códigos para refatoração

Segundo Fowler (2019), identificar o momento propício para se fazer uma refatoração é um desafio e um aprendizado constante. O código precisa ser avaliado de forma a buscar estruturas de códigos que sugerem uma oportunidade de refatoração. Essas estruturas de códigos são indícios - também chamados de *code smells* - de que há um problema que pode ser resolvido com refatoração.

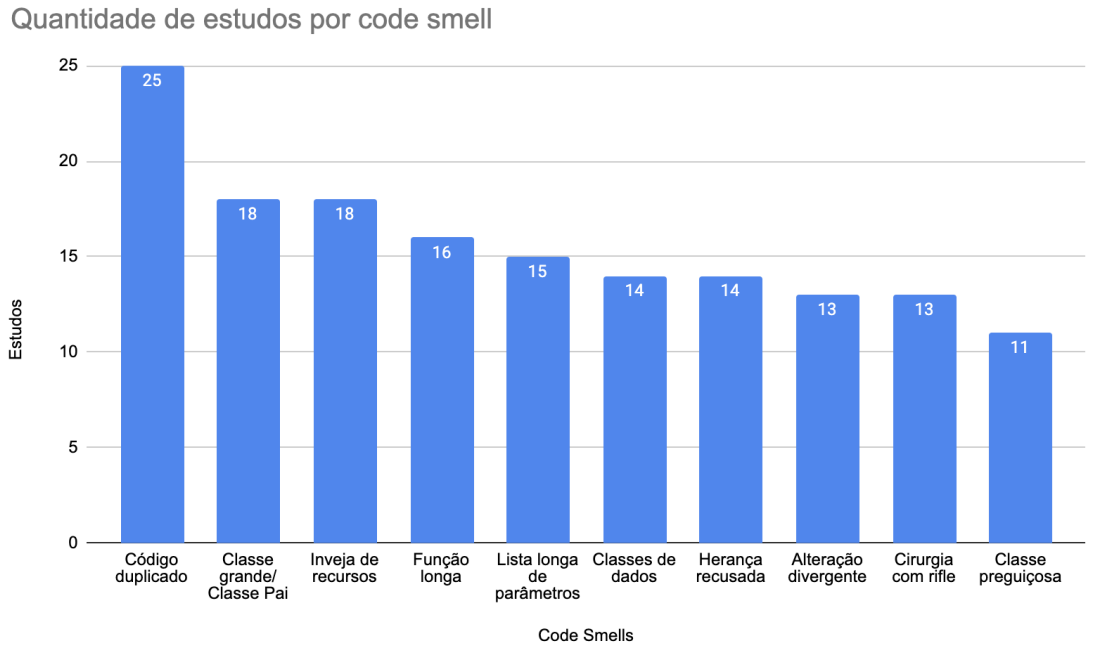
"Code smells podem afetar de forma negativa o desenvolvimento do *software"* (ALIZADEH *et al.*, 2020, Tradução Nossa). Normalmente refletem o código que está em decadência, podendo gerar a necessidade de uma manutenção futura. Provocam falhas potenciais e despesas gerais com relação a testes associados (COUNSELL *et al.*, 2010). Conforme Lacerda *et al.* (2020), *code smells* podem ser problemas de *design* que podem afetar a evolução e manutenção do código, também podem afetar atributos externos e internos de qualidade ao longo do tempo, tais como baixa coesão, alto acoplamento, problemas relacionados ao encapsulamento que influenciam as decisões de projeto e a manutenção do código. A refatoração é a principal estratégia para remover e/ou mitigar os *code smells* de forma melhorar a qualidade do *software* (LACERDA, *et al.*, 2020)

"Conforme o código evolui, *code smells* são inseridos involuntariamente pelos desenvolvedores e podem afetar a qualidade do *software* ao longo do tempo" (KAUR *et al.* , 2019, Tradução Nossa).

"Entretanto, o reconhecimento de *code smells* é relativo e nem sempre são corrigidos pois, para alguns desenvolvedores, o *software* funciona e/ou acreditam que não são importantes para que sejam corrigidos" (ALIZADEH *et al*., 2020, Tradução Nossa).

Para Lacerda et al. (2020), é importante padronizar os conceitos de modo a aumentar a consistência de detecção de *code smells.* A Figura 3 apresenta os c*ode smells* com maior incidência detectados durante os estudos realizados. A Tabela 1 apresenta os *code smells* e as respectivas descrições que podem ser usados como base para a identificação durante esse estágio da refatoração (FOWLER, 2019).

Figura 3 - Quantidade de estudos por *Code Smells*



Fonte: Lacerda (2020) - Traduzido pela autora

Tabela 1 - Descrição de *code smells*

ITEM	NOME	DESCRIÇÃO
1	Nome misterioso	Quando o nome de uma função, variável e/ou classe não é claro quanto ao comportamento esperado.
2	Código duplicado	A mesma estrutura de código sendo feita em mais de um lugar.
3	Função longa	Funções contendo uma grande quantidade de linhas e responsabilidades.
4	Lista longa de parâmetros	Quando há uma grande quantidade de parâmetros passados para uma determinada função.
5	Dados globais	Dados globais são aqueles que podem ser modificados em qualquer parte do código.
6	Dados mutáveis	Em determinadas partes do programa o dado é modificado, normalmente podendo receber valores de diversos locais.

Fonte: Fowler (2019).

Tabela 1 - Descrição de *code smells* - continuação

ITEM	NOME	DESCRIÇÃO
7	Alteração divergente	Ocorre quando um módulo é alterado de maneira frequente e por motivos diferentes.
8	Cirurgia com rifle	Necessidade de fazer alterações pequenas em várias classes diferentes.
9	Inveja de recursos	Uma função gasta mais tempo se comunicando com funções de outros módulos do que com o próprio módulo.
10	Agrupamento de dados	Ocorre quando há muitos dados sendo utilizados.
11	Obsessão por primitivos	Quando variáveis são tipadas com tipos primitivos não fornecendo um contexto claro e seguro referente sua utilização.
12	*Switches* repetidos	Acontece quando são utilizados muitos *switch/case.*
13	Laços	Laços de repetição utilizados no código.
14	Elemento ocioso	Uma classe, função ou variável não utilizada.
15	Generalidade especulativa	Recursos não utilizados no código, normalmente utilizados apenas em testes.
16	Campo temporário	Classes que contém campos que são definidos em apenas algumas circunstâncias.
17	Cadeias de mensagens	Vários objetos com relacionamentos entre outros vários objetos na mesma instância.
18	Intermediário	Quando há uso de interface implementada em apenas uma classe.
19	Trocas escusas	Muita troca de dados entre módulos.
20	Classe grande	Quando uma classe possui muitas responsabilidades.
21	Classes alternativas com interfaces diferentes	Classes que não podem ser trocadas devido ao uso de interfaces distintas.
22	Classes de dados	Classes que possuem dados, métodos de escrita e leitura.

Fonte: Fowler (2019).

Tabela 1 - Descrição de *code smells* - continuação

ITEM	NOME	DESCRIÇÃO
23	Herança recusada	Classes que não é necessário o uso de todos os métodos da superclasse.
24	Comentários	Muitos comentários utilizados no código.

Fonte: Fowler (2019).

2.1.1.2. Seleção das atividades de refatoração

Como já observado, a refatoração consiste em aplicar pequenos passos que possibilitam o próximo passo para que se possa alcançar um objetivo maior (MARTIN, 2011). Uma maneira para aumentar a adoção da refatoração é destacar o esforço existente para a execução de cada passo (ALOMAR *et al*., 2021).

Fowler (2019) descreve uma série de operações de refatoração. Alomar *et al*. (2021) identificaram que para preservar o comportamento da aplicação foram identificadas 150 atividades de refatoração, sendo 43 mapeadas no catálogo de Fowle e servem a diferentes objetivos: compor métodos, organizar dados, simplificar expressões condicionais e chamadas de métodos, lidar com generalizações e mover recursos entre objetos,

Kaur *et al.* (2019) realizou uma pesquisa referente ao impacto das atividades de refatoração na qualidade de *software*. Esse estudo demonstrou que 45,5% das atividades de refatoração que foram consideradas pertencem ao catálogo de Fowler.

Valente (2020) apresenta as principais operações de refatoração retiradas do catálogo de Fowler, sendo elas:

- Extração de Método;
- *Inline* de Método;
- Movimentação de Método;
- Extração de Classes;
- Renomeação;
- Extração de Variáveis;

- Substituição de Condicional por Polimorfismo.

Para o desenvolvimento de *software*, são utilizadas ferramentas para Ambientes de Desenvolvimento Integrado, ou *Integrated Development Environment* (IDE). “Algumas dessas ferramentas possuem funcionalidades para facilitar a implementação de refatorações de forma a otimizar o trabalho do desenvolvedor” (OUNI *et al*., 2012, Tradução Nossa), sendo elas:

- Mover método;
- Mover variável;
- Subir campo;
- Subir método;
- Descer campo;
- Descer método;
- Internalizar classe;
- Extrair método;
- Extrair classe;
- Mover classe;
- Extrair interface.

Em seu estudo, Alomar *et al*. (2021) observa que os tipos de refatorações de Extração de métodos e renomeação de métodos foram estudados com maior frequência, o que pode indicar sua importância na preservação do comportamento e são potencialmente mais aplicadas na prática do que outras operações.

No presente trabalho serão consideradas as atividades de refatoração mais utilizadas e estudadas por pesquisadores. Fowler (2019) iniciam o catálogo com o conjunto de refatorações com as consideradas mais úteis para o desenvolvedor, sendo elas:

- Extrair função;
- Extrair variável;
- Internalizar função;
- Internalizar variável;
- Mudar declaração de função;

- Renomear variável;
- Encapsular variável;
- Introduzir objeto de parâmetros;
- Combinar funções em classes;
- Combinar funções em transformação
- Separar fases.

A seguir são descritas as atividades de refatoração conforme Fowler (2019).

- **Extrair função ou Extrair Método:** Uma das refatorações mais comum, consiste em observar um fragmento de código de forma a identificar o que ele faz e em seguida é feita a extração do código para uma nova função/método com um nome que reflita o seu propósito.
- **Extrair variável:** Expressões podem se tornar complexas para se entender e gerar sentido dúbio. Ao extrair a expressão para uma variável com um nome correspondente ao que a expressão representa, o código fica mais simples para ser compreendido.
- **Internalizar função ou internalizar método:** É o inverso de "Extrair Função". Uma função pequena cujo corpo é tão claro quanto o nome. Neste caso, não há a necessidade de se utilizar uma outra função, sendo possível utilizar acesso indireto.
- **Internalizar variável:** É o inverso de "Extrair Variável". Quando o nome da variável diz a própria expressão, neste caso pode ser conveniente não ter uma variável.
- **Mudar declaração de função:** Quando uma função possui nome que gera dúvidas quanto à sua finalidade, bem como aos parâmetros da função que definem o contexto em que a função pode ser utilizada, alterar os parâmetros da função pode aumentar a aplicabilidade desta e reduzir o acoplamento.
- **Renomear variável:** Bons nomes de variáveis contribuem com a clareza do código.
- **Encapsular variável:** Caso haja a necessidade de se manipular dados, principalmente quando são utilizados em vários locais do código, é mais simples transformar o acesso e modificação dos dados em funções.

- **Introduzir objeto de parâmetros:** Funções que possuem um número grande de parâmetros ou até mesmo dados que aparecem regularmente juntos, podem ser transformadas em uma única estrutura de dados, tornando explícito o relacionamento entre eles.
- **Combinar funções em classes:** Funções que atuam de forma muito próximas com um corpo comum de dados podem ser combinadas em uma classe.
- **Combinar funções em transformação:** Reunir funções que geram dados derivados, com um corpo comum de dados, em apenas uma função.
- **Separar fases:** Um código que lida com tarefas diferentes pode ser separado em classes ou módulos distintos.

2.1.1.3. Garantia para manter o comportamento do código

Mudanças no código podem gerar um certo receio por parte do desenvolvedor ao fazer grandes alterações na base de código e introduzir falhas sutis (CHIELE, 2017).

A base para a refatoração é ter um código auto testável com testes automatizados que possam ser executados e, caso haja alteração de comportamento no código, o teste falhe (FOWLER, 2019). As pré-condições devem ser verificadas antes de aplicar a refatoração, de forma que não apresente problemas de compilação ou até mesmo alterar o comportamento do código (LACERDA, 2020).

Para aumentar a segurança da aplicação para a refatoração é necessário incorporar um conjunto de testes em diferentes níveis de granularidade anteriormente à etapa de refatoração, implementando pré condições para que desta forma seja possível identificar qualquer alteração que afete a preservação do comportamento (ALOMAR *et al*., 2021).

O sucesso da refatoração depende da existência de testes adequados, sobretudo testes de unidade (VALENTE, 2020).

2.1.1.4. Aplicação das atividades de refatoração

A aplicação da refatoração pode ser direta ou indireta. Na abordagem direta, a refatoração é aplicada diretamente ao código e/ou modelo e pode ser facilmente

automatizada. No caso da abordagem indireta, a sequência de refatoração é produzida como uma solução otimizada e posteriormente aplicada ao artefato (LACERDA, 2020).

Devido a inúmeras opções de *design*, é um grande desafio escolher refatorações ideais para cada *code smell*, de forma que maximize a qualidade do programa resultante e minimize o custo da transformação referente à preservação do comportamento (ALOMAR *et al*., 2021).

No trabalho feito por Coursell *et al.* (2010), na Tabela 2 são apresentados um conjunto de refatorações parciais comuns para cada *code smell* detectado durante o estudo, bem como as respectivas refatorações sugeridas por Fowler (2019).

Tabela 2 - Conjunto de refatorações por *code smell*

CODE SMELL	**REFATORAÇÃO** (COURSELL *et al.*, 2010)	**REFATORAÇÃO** (FOWLER, 2019)
Classe de dados	Movimentação de método	Encapsular registro
	Encapsulamento de campo	Remover método de escrita
	Encapsulamento de coleção	Mover função
		Extrair função
		Separar em fases
Alteração divergente	Movimentação de método	Separar em fases
	Movimentação de campo	Mover função
	Extrair função	Extrair função
		Introduzir caso especial
Encapsulamento inapropriado	Movimentação de método	Na segunda edição do livro, esse *code smell* não foi incluído.
	Movimentação de campo	
	Alterar associação bidirecional para unidirecional	
	Substituir herança com delegação	
	Delegação oculta	

Fonte: Autora

Tabela 2 - Conjunto de refatorações por *code smell* - continuação

CODE SMELL	**REFATORAÇÃO** (COURSELL *et al.*, 2010)	**REFATORAÇÃO** (FOWLER, 2019)
Classe grande	Extrair classe	Extrair classe
	Extrair superclasse	Extrair superclasse
	Extrair interface	Substituir código de tipos por subclasses
	Substituir valores de dados por objeto	
Cirurgia com rifle	Movimentação de método	Mover função
	Movimentação de campo	Mover campo
	Classe em linha	Combinar funções em classe
		Combinar funções em transformação
		Separar em fases
		Internalizar função
		Internalizar classe
Generalidade especulativa	Colapso de hierarquia	Condensar Hierarquia
	Classe em linha	Internalizar função
	Remover parâmetro	Internalizar classe
	Renomear método	Mudar declaração de função
		Remover código morto

Fonte: Autora

2.1.1.5. Avaliação do impacto da refatoração na qualidade de *software*

Ao realizar a refatoração espera-se que haja uma melhora na qualidade geral do projeto de *software*, bem como a correção de defeitos de projeto (FOWLER, 2019).

Durante a investigação feita por Al Shayeb (2009) foram estudados os atributos externos de qualidade com base em três métodos de refatoração: Encapsular variável, Extrair método e Mover função. Os atributos estudados foram:

- Adaptabilidade: facilidade com que o *software* pode ser modificado para o uso em aplicações ou ambientes diferentes daqueles para os quais foram projetados;
- Manutenibilidade: a facilidade com que um componente pode ser alterado com intuito de corrigir falhas, melhorar o desempenho ou outros aspectos, ou adaptar-se a um ambiente modificado;
- Compreensibilidade: o grau em que o significado de um componente é claro para um usuário;
- Reusabilidade: o grau em que um componente pode ser usado em mais de um sistema de *software*, ou na construção de outros componentes, com pouco ou nenhuma adaptação;
- Testabilidade: um conjunto de atributos de *software* que se relacionam com o esforço necessário para validar o produto de *software*.

As oportunidades de refatoração correspondem àquelas que melhoram as métricas de coesão e acoplamento de forma que há uma boa distribuição das características das classes. Porém, há relatos de que as refatorações realizadas manualmente pelos desenvolvedores não necessariamente melhoram a modularidade em termos de coesão e acoplamento, mas sugerem que a refatoração é útil para melhorar aspectos específicos do sistema (OUNI, *et al.* 2016).

Há várias maneiras de medir a qualidade de um *software* orientado a objetos, tais como métricas Chidamber e Kemerer (CK), Métricas para *Design* Orientado a Objetos (MOOD), e Modelo de Qualidade para *Design* Orientado a Objetos (QMOOD). O modelo QMOOD se baseia nos aspectos de qualidade conforme na ISO 9126 utilizando as métricas para *design* orientado a objetos (MOOD) para calcular os valores de cada atributo de qualidade (COUTO, 2018).

Na busca de otimizar o processo de refatoração e reduzir *code smells*, Alizadeh *et al.* (2020) utilizaram como base as métricas de qualidade QMOOD (*Quality Model for Object-Oriented Design*). Em seu trabalho, Ouni *et al*. (2016) também utilizaram as

métricas de QMOOD para estimar os efeitos das refatorações antes e depois de realizada a refatoração. A escolha das métricas de QMOOD foi que além de ser amplamente utilizado na literatura para avaliar o efeito da refatoração, tem a vantagem de definir seis atributos de qualidade de *design* de alto nível (reutilização, flexibilidade, compreensibilidade, funcionalidade, extensibilidade e eficácia) (OUNI *et al.*, 2016). A Tabela 3 apresenta a relação das propriedades das métricas de qualidade de *software* QMOOD, utilizada nos estudos de Alizadeh *et al.*(2020) e Ouni *et al.* (2016).

Tabela 3 - Descrição das propriedades de QMOOD

Métrica de Design (MOOD)	**Propriedade do Design**	**Descrição** (ALIZADEH *et al.*, 2020)	**Descrição** (OUNI *et al.*, 2016)
Tamanho do *design* nas classes (DSC)	Tamanho do design	Número total de classes no *design*	Número total de classes no *design*
Números de Hierarquia (NOH)	Hierarquias	Número total de classes "raiz" no *design*	Número de hierarquias
Número médio de ancestrais (ANA)	Abstração	Número médio de classes na árvore de herança para cada classe.	Média de número de antecessores
Métrica de acesso direto (DAM)	Encapsulamento	Proporção do número de atributos privados e protegidos para o número total de atributos em uma classe.	Métrica de acesso aos atributos
Acoplamento de Classe Direto (DCC)	Acoplamento	Número de outras classes às quais uma classe se relaciona, seja por meio de um atributo compartilhado ou de um parâmetro em um método.	Acoplamento direto da classe
Coesão entre os métodos da classe (CAM)	Coesão	Medida de como os métodos relacionados estão em uma classe em termos de parâmetros usados	Coesão entre métodos nas classes
Medida de agregação (MOA)	Composição	Contagem do número de atributos cujo tipo é(são) classe(s) definida(s) pelo usuário.	Medida de agregação

Fonte: Alizadeh *et al.* (2020) e Ouni *et al.* (2016) - adaptado pela autora

Tabela 3 - Descrição das propriedades de QMOOD - continuação

Métrica de Design (MOOD)	**Propriedade do Design**	**Descrição** (ALIZADEH *et al.*, 2020)	**Descrição** (OUNI *et al.*, 2016)
Medida de abstração funcional (MFA)	Herança	Proporção do número de métodos herdados pelo número total de métodos dentro de uma classe	Medida de abstração funcional
Número de métodos polimórficos (NOP)	Polimorfismo	Qualquer método que pode ser usado por uma classe e seus descendentes. Conta o número de métodos em uma classe excluindo os privados, estáticos e finais.	Número de métodos polimórficos
Tamanho da interface de classe (CIS)	Mensagem	Número de métodos públicos na classe	Tamanho das interfaces
Número de métodos (NOM)	Complexidade	Número de métodos declarados em uma classe.	Número de métodos

Fonte: Alizadeh *et al.* (2020) e Ouni *et al.* (2016) - adaptado pela autora

A Tabela 4 apresenta o cálculo dos atributos de qualidade conforme as métricas, utilizada nos estudos de Alizadeh *et al.* (2020) e Ouni *et al.* (2016).

Tabela 4 - Cálculo do índice de Atributo de Qualidade.

Autor	**Atributo de Qualidade**	**Cálculo do Índice de Qualidade**
Ouni *et al.* (2016)	Reusabilidade	= -0,25DCC + 0,25CAM + 0,5CIS + 0,5DSC
	Flexibilidade	= 0,25DAM - 0,25DCC + 0,5MOA + 0,5NOP
	Compreensibilidade	= -0,33ANA + 0,33DAM - 0,33DCC + 0,33CAM - 0,33NOP + 0,33NOM - 0,33DSC
	Eficácia	= 0,2ANA + 0,2DAM + 0,2MOA + 0,2MFA + 0,2NOP
Alizadeh *et al.* (2020)	Funcionalidade	= 0,12CAM + 0,22NOP + 0,22CIS + 0,22DSC + 0,22NOH
	Extensibilidade	= 0,5ANA - 0,5DCC + 0,5MFA + 0,2NOP

Fonte: Alizadeh *et al.* (2020) e Ouni *et al.* (2016) - adaptado pela autora

Para Abreu (1995), destacam-se seis métricas de *design* orientado a objetos principais que envolvem os conceitos de programação orientada a objetos (Encapsulamento, Acoplamento, Herança, Ocultamento de Informações, Polimorfismo) para medir o projeto de *software*. As métricas e suas respectivas descrições são apresentadas na Tabela 5.

Tabela 5 - Principais métricas de qualidade.

Métrica	Descrição
Método Fator de Ocultamento (MHF)	Mostram a quantidade média de como os métodos de uma classe estão ocultos no sistema. MHF e AHF são 100% de todos os membros ocultos. Se tivermos MHF baixo, então os métodos são visíveis para quase todas as classes. Os métodos têm uma alta tendência de reuso. Em outro caso, MHF alto significa que os métodos não são visíveis para muitas classes e eles não conseguem entender o funcionamento da mesma. Estes métodos podem ser usados somente pela classe que os contém e tem menos tendência à reusabilidade.
Fator de Ocultamento de Atributos (AHF)	Mostram a quantidade média de como os atributos de uma classe estão ocultos no sistema. Se tivermos AHF baixo, então os métodos são visíveis para quase todas as classes. Em outro caso, AHF alto significa que os atributos não são visíveis para muitas classes.
Fator de Herança do Método (MIF) Fator de Herança do Atributo (AIF)	A extensão em que esses métodos e atributos são herdados é definida pelo Fator de Herança do Método (MIF) e Fator de Herança do Atributo (AIF). Uma classe filha que herda um grande número de métodos e atributos de sua classe mãe contém um grande valor de MIF e AIF. Como qualquer coisa em uma quantidade muito pequena de valores muito altos é prejudicial, o mesmo caso é com MIF e AIF. Estes valores devem estar em uma faixa razoável. O baixo valor da AIF mostra que a classe não deve herdar os atributos, mas sim que os atributos devem ser privados para a classe.
Fator Polimorfismo (PF / POF)	É definido pela proporção de um número real de sobreposições de métodos e o número máximo de sobreposições totais de métodos. Para manter o código limpo e claro e proporcionar alta qualidade podemos usar PF alta, mas isso aumenta a complexidade do sistema. O fator polimorfismo está associado à sobreposição do método, não está associado à sobrecarga do método.
Fator de Acoplamento (CF / COF)	O alto valor do CF mostra que as classes do sistema são mais interconectadas e interdependentes. Para medir o acoplamento real entre as diferentes classes é utilizado o Fator de Acoplamento (CF). É a relação de Acoplamento real entre diferentes classes e o máximo acoplamento possível pode acontecer no sistema. O alto valor do acoplamento no sistema leva a uma maior complexidade.

Fonte: Abreu (1995).

A Tabela 6 apresenta a relação entre as métricas apontadas por Abreu (1995), Alizadeh *et al.* (2020) e Ouni *et al.* (2016).

Tabela 6 - Relação entre métricas.

Métricas ABREU, 1995	**Métrica de Design correspondente (**ALIZADEH *et al.* ,2020 & OUNI *et al.*, 2016)
Fator de encapsulamento de método (MHF)	Métrica de acesso direto (DAM)
Fator de encapsulamento de atributos (AHF)	Métrica de acesso direto (DAM)
Fator de Herança do Método (MIF)	Medida de abstração funcional (MFA)
Fator de Herança do Atributo (AIF)	Medida de abstração funcional (MFA)
Fator Polimorfismo (PF / POF)	Número de métodos polimórficos (NOP)
-	Acoplamento de Classe Direto (DCC)
-	Medida de agregação (MOA)
-	Medida de abstração funcional (MFA)
-	Número de métodos polimórficos (NOP)
-	Tamanho da interface de classe (CIS)
-	Número de métodos (NOM)

Fonte: Autora.

Atualmente existem diversas ferramentas para se calcular métricas de *software* utilizando ferramentas acopladas às ferramentas para Ambientes de Desenvolvimento Integrado, ou *Integrated Development Environment* (IDE).

2.1.1.6. Preservação da consistência do *software*

Após a aplicação de refatorações, é importante garantir que o *design* do programa permaneça consistente e que a coerência semântica do domínio seja preservada. (OUNI, *et al.* 2012).

O histórico de alterações pode ser útil para aumentar a confiança de novas recomendações de refatoração, além de orientar melhor o processo de buscas e propor novas refatorações em contextos semelhantes (OUNI, *et al.* 2016).

2.2. Gamificação

A gamificação é um conceito recente que pode ser confundido com jogos, sistemas de recompensas, programas de fidelidade, dentre outros que possuem objetivo de persuadir as pessoas a realizar ações para ganhar pontos (RUHI, 2015). Os principais objetivos da gamificação são atrair e motivar a participação de colaboradores para executar atividades normais de trabalho de forma que atinja as metas estabelecidas (FORMANSKI, 2016; BUNCHBALL, 2010).

Também pode ser entendida como uma ferramenta para criar comportamentos, desenvolver habilidades e, por consequência, possibilitar a inovação de processos. Pode favorecer a performance dos colaboradores, melhorar a aprendizagem e engajamento (NEIDENBACH *et al.*, 2020).

A gamificação foi adotada nos negócios para alavancar a motivação dos indivíduos e tem sido amplamente utilizada para promover e encorajar mudanças de atitudes, comportamentos ou cultura em diferentes áreas (GIMENEZ, 2021).

Conceitualmente, a gamificação associa um conjunto de regras visando o alcance de objetivos e elementos competitivos, não pelo ato de jogar em si, mas ações que são focadas em valor de negócios e avançar com os objetivos no contexto aplicado (KOIVISTO & HAMARI, 2014; RUHI, 2015).

A técnica de gamificação compreende a introdução de diversão, reconhecimento pessoal e/ou competição, convertendo os colaboradores em jogadores. Essa técnica utiliza a "teoria da diversão", provocando uma motivação nas pessoas para mudar a percepção e atitude em relação a alguns assuntos (FORMANSKI, 2016).

Pode ser entendida como uma forma de organizar a colaboração, extraindo os elementos de jogos, adaptando-os e usando os elementos em dado contexto, criando uma sensação de diversão e engajamento em contextos de atividades maçantes (ROTH, 2015).

A abordagem de gamificação usa elementos baseados em jogos, para incentivar os usuários a realizarem comportamentos desejados, envolver os colaboradores em determinadas tarefas e motivar o desenvolvimento de soluções diante de dificuldades encontradas por meio de processos similares aos jogos, para realizar uma tarefa complexa ou atingir um determinado objetivo (PATRÍCIO, 2018).

O ponto de chegada de uma solução gamificada consiste nas motivações e objetivos dos jogadores, sendo construída com uma série de desafios tendo como objetivo primário engajar os jogadores em um nível emocional, além de motivar a alcançar metas que sejam significativas para o próprio jogador, e como objetivo secundário, os objetivos da própria empresa (BURKE, 2015).

2.2.1. Pilares

A gamificação é mais que um simples sistema de premiação por pontos e emblemas. Também tem como propósito influenciar positivamente os comportamentos humanos. Desta forma, encontra-se fundamentada nos pilares: motivação, habilidade e estímulo, como pode ser observado na Figura 4 (FORMANSKI, 2016).

Figura 4 - Pilares da gamificação

Fonte: Formanski (2016)

Para que haja mudança no comportamento humano, os pilares da gamificação necessitam estar presentes, sendo que habilidade e motivação são essenciais para essa mudança. Se houver menor habilidade, é preciso ter maior motivação, sendo que a motivação é a base para que o processo de gamificação seja bem sucedido (FORMANSKI, 2016).

2.2.2. Implantação

Para implantar um processo de gamificação é necessário que o processo estimule o engajamento, ofereça um desafio sendo definido por regras claras, que haja interação e que ofereça *feedbacks* para que se possa alcançar resultados quantificáveis com a presença de reações emocionais (NEIDENBACH *et al.*, 2020).

Para Burke (2015), para realizar o planejamento do processo de gamificação é necessário que sejam feitas algumas definições:

- **Público-alvo**: quem o método pretende atingir, de forma a estabelecer um grupo de pessoas que a empresa precisa se envolver;

- **Objetivo**: com base nos objetivos da empresa;
- **Modelo de engajamento dos jogadores**: como promover a cooperação acima da competição, modo que os jogadores interagem com o jogo;
- **Ambiente**: as ações e ciclos a serem percorridos;
- **Incentivos e recompensas**: como incentivar e recompensar os jogadores.

2.2.3. Implantação da gamificação

2.2.3.1. Público-alvo

Identificar quais serão os envolvidos que participarão do processo gamificado. Ao identificar o público alvo é possível analisar as formas de trabalho de cada membro, visualizar o modo que é feita a interação em diferentes partes da solução, identificar os objetivos dos envolvidos, determinar quais incentivos e recompensas serão mais aceitas e determinar o objetivo do ponto de vista do colaborador (BURKE, 2015).

A gamificação oferece diferentes caminhos de aprendizagem, sendo que o foco são as pequenas conquistas e tem como fundamento o desenvolvimento de habilidades, atitudes e outras características do jogador (BUSARELLO, 2016).

Nesta etapa sugere-se que haja uma verificação referente ao perfil do público alvo, podendo utilizar ferramentas de coleta de informação, conversas entre as equipes para entender as atividades que realizam e como interagem com as ferramentas existentes (BURKE, 2015).

2.2.3.2. Objetivos dos colaboradores

A fim de motivar os colaboradores a alcançar os objetivos da gamificação, é necessário conhecer os objetivos dos colaboradores envolvidos e identificar quais objetivos estão alinhados com os objetivos da empresa ao aplicar o processo de gamificação, nem sempre estarão alinhados mas é necessário que tenha um ponto de intersecção entre os objetos de negócio e os objetivos dos jogadores (BURKE, 2015).

Objetivos não alinhados ou que os usuários não consideram realmente importantes podem levar o processo gamificado ao fracasso (SUH, 2015).

2.2.3.3. Objetivos de negócio

Conforme Burke (2015), é necessário definir os objetivos de negócio a serem atingidos com a gamificação, esses objetivos devem ser realistas, alcançáveis e quantificáveis de modo a ser possível avaliar o sucesso do objetivo.

Durante essa etapa a cultura da empresa deve ser levada em consideração, pois a gamificação promove a inovação do modelo de negócios, bem como no desenvolvimento de serviços e infraestrutura, quando aplicáveis. Ao avaliar a cultura da empresa é possível identificar possíveis pontos de resistências que podem dificultar a implementação da solução gamificada, bem como a utilização da mesma pelos jogadores (GIMENEZ, 2021).

2.2.3.4. Modelo de engajamento

A motivação intrínseca de um indivíduo para realizar uma determinada atividade pode ser caracterizada pela autonomia, competência e relacionamento. Sendo que autonomia é o desejo ou disposição para se realizar uma atividade; competência refere-se ao sentimento de eficácia; e o relacionamento é experimentado quando há conexão com o que é realizado (SUH, 2015).

Em seu livro, Burke (2015), cita que a motivação apresenta três elementos essenciais:

- Autonomia: desejo de comandar a própria vida e optar a forma em que irá solucionar os desafios propostos;
- Domínio: a necessidade de progredir e se tornar melhor em algo que importa. Fornecer *feedback* positivo e de fácil adesão para motivar os jogadores a buscar um melhor desempenho de forma a indicar o progresso;
- Propósito: desejo de fazer por causa de algo maior. Deve começar e terminar com um propósito centrado em um objetivo maior que o próprio jogador.

Nessa etapa são definidos os parâmetros de interação de cada colaborador com a solução gamificada. Os parâmetros podem ser (BURKE, 2015):

- **Colaborativo/Competitivo:** Um parâmetro básico para a gamificação é encontrar o equilíbrio entre competição e colaboração. Na competição um

jogador vence e os demais perdem. Em contrapartida, na colaboração os jogadores são estimulados a colaborarem entre si e se ajudarem. Para melhor aproveitamento da solução a maneira mais comum é mesclando ambos os parâmetros, criando equipes de forma que haja colaboração entre os jogadores da equipe e competição entre as equipes;

- **Intrínseco/Extrínseco**: Soluções extrínsecas são recompensas de elementos tangíveis, ou seja, programas de recompensas. Porém na gamificação as recompensas intrínsecas são consideradas no primeiro momento, mas não há impedimentos para se usar ambos os tipos de recompensas;
- **Multijogadores/Jogadores individuais**: Jogadores individuais interagem com o próprio jogo, por outro lado, multijogadores interagem com outros jogadores;
- **Por campanha/infinito**: Normalmente uma solução gamificada é utilizada para criar um novo hábito, ensinar uma nova habilidade ou alterar um hábito de um indivíduo, nesses casos devem ter um fim natural. Quando soluções são voltadas para o ensino, o aprendizado é sem fim, logo não há um fim para a solução. Quando utilizado em treinamentos e campanhas, é comum que as etapas sejam finalizadas;
- ***Emergent gameplay*/Roteirizado:** Quando os resultados esperados são conhecidos, o objetivo é mudança de comportamento ou um treinamento, é comum utilizar soluções com roteiros pré-definidos. Enquanto outros em que os resultados são desconhecidos, o *emergent gameplay* é utilizado quando o ambiente muda conforme o andamento do jogo.

2.2.3.5. Ambiente

Espaço do jogo: Um ambiente no qual o jogador pode acessar o seu progresso e ter visibilidade do seu estado atual com relação ao jogo. Não necessariamente precisa ser algo com *design* ou um ambiente real (BURKE, 2015).

Burke (2015) descreve o caminho que o jogador irá percorrer como sendo a jornada do jogador, à medida que a habilidade aumenta, o desafio também aumenta. Em cada etapa há *feedbacks* para cada ação tomada, seja positivo ou negativo. As

ações devem ser desafiadoras mas também alcançáveis conforme a habilidade atual do indivíduo (BURKE, 2015).

Na estrutura de jogos, é importante estimular a sensação de progresso e a motivação para completar as tarefas do jogo. A progressão por níveis traz essa sensação e a superação de desafios (MURR, 2020).

Para alterar comportamentos, é necessário alterar hábitos. Burke (2015) sugere uma série de características para a mudança de hábitos:

- Estabeleça objetivos: Definir objetivos que engaje os participantes do jogo de maneira significativa, oferecendo o *feedback* durante o progresso;
- Use desencadeadores: Lembretes para os jogadores para acionar gatilhos para realizar ações específicas;
- Dê passos pequenos: Quando o objetivo é muito grande e seu alcance seja de difícil visualização, é melhor repartir em alvos menores e mais fáceis de administrar;
- Encontre espíritos afins: Mudar o hábito é mais fácil ao fazer parte de um grupo que também está fazendo a mesma coisa, pois é parte da natureza humana querer se manter alinhado com os demais do grupo;
- Peça o apoio de amigos: Compartilhar o progresso com amigos;
- Eleve o grau de complexidade com o tempo: Conforme a habilidade dos jogadores se desenvolve, a solução pode oferecer treinamento e instruções para mudanças comportamentais mais complexas;
- Repita até que novos hábitos se instalem: Quando um novo comportamento é aprendido, este precisa ser repetido durante um determinado período até que se torne um hábito;
- Mantenha o caráter inovador: O uso da solução pode ser prolongado e para isso é necessário modificá-la para manter o interesse dos jogadores.

2.2.3.6. Incentivos e recompensas

Patricio (2018) descreve os procedimentos básicos que aumentam o envolvimento e engajamento do usuário, tais como: desafios, competição, cooperação, recompensas e turnos. As abordagens mais comuns são desafios, competição e

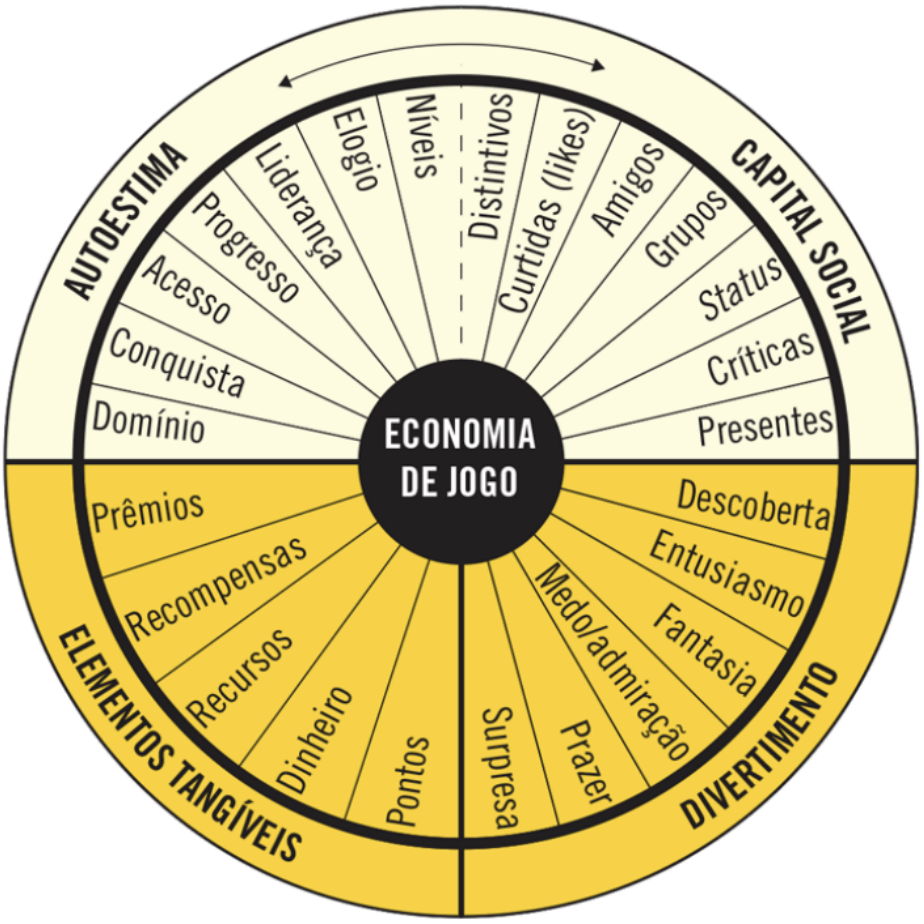
feedback. Normalmente os entregáveis exigem algum esforço (tempo, habilidade e criatividade), como por exemplo: envio de projetos, desafios sociais, etc.

A mecânica são princípios, regras e/ou mecanismos que regem um comportamento através de incentivos, *feedback* e recompensas, de forma a alcançar um determinado resultado e motivar as pessoas através de retornos positivos, dentre eles o acúmulo de pontos, conquista de emblemas, ganho de status, reconhecimento do progresso, personalização, surpresas agradáveis, entre outros (FORMANSKI, 2016).

As técnicas de mecânica de jogo podem ser aplicadas e combinadas para gamificar qualquer ambiente de forma a induzir o comportamento dos usuários (DORLING e MCCAFFERY, 2012).

Para Burke (2015), os incentivos e recompensas são considerados como a economia do jogo e tem como objetivo criar experiências e envolver os jogadores a realizar as tarefas que muitas vezes não possuem valor intrínseco para o próprio jogador, nesse caso uma recompensa tangível seja mais indicada, a Figura 5 representa as economias do jogo.

Figura 5 - Economia do Jogo



Fonte: Burke (2015)

2.2.3.6.1. Economia do jogo

Para Burke (2015), incentivos e recompensas podem ou não serem tangíveis, e ocorrem quando um jogador é bem sucedido na realização das tarefas. Nesta etapa é fundamental conhecer as metas e motivações dos jogadores para que a solução seja centrada no colaborador. Na economia do jogo há quatro moedas básicas que podem ser obtidas ao jogar que são implementadas através da mecânica do jogo.

- Divertimento;
- Elementos tangíveis (recompensas);
- Autoestima;
- Capital social.

2.2.3.6.1.1. Divertimento

Utilizado muitas vezes nos videogames envolvendo gráficos estimulantes porém, não é duradouro nem oferece recompensas tangíveis. Na gamificação surge em forma de recompensas aleatórias e inesperadas (BURKE, 2015).

Patricio (2018) descreve as motivações, tais como: narrativa, progressão e interação social que estimulam emoções dos jogadores através de regras e restrições do jogo e através de reconhecimento, felicidade, entusiasmo, competitividade e motivação cria um ambiente divertido e construtivo;

2.2.3.6.1.2. Elementos tangíveis (recompensas)

Considerada uma moeda que pode ser coletada ou trocada na própria solução, muitas vezes implementadas como pontos que podem ser trocados em prêmios ou dinheiro. Exige uma clara compreensão dos objetivos e motivações do jogador para que seja atrativo (BURKE, 2015).

A Tabela 7 apresenta algumas dinâmicas (estímulos) e suas respectivas mecânicas que frequentemente são relacionadas (FORMANSKI, 2016).

Tabela 7 - Correlação de dinâmicas e mecanismos de jogos.

Dinâmicas de jogos	Mecanismos de Jogos
Recompensa	Pontos
Status	Níveis
Conquistas	Troféus e emblemas
Competição	Quadros de liderança

Fonte: Formanski (2016).

Patricio (2018) demonstra de forma concreta a dinâmica e a mecânica, por exemplo: conquistas, avatares, distintivos, presentes, placares, pontos e bens virtuais. Os componentes mais comuns envolvem pontos, conquistas e emblemas, sendo componentes que são dados para um desempenho específico resultando no aumento da motivação quando os objetivos são realistas e desafiadores. Também

introduz entusiasmo e competitividade para atingir os objetivos, vencer ou aumentar o status do jogador.

Durante sua pesquisa, Formanski (2016) cita que o os pontos, emblemas e quadros de liderança (PBL - *Points, Badges e Leaderboard*) são elementos básicos para a gamificação sistema básico de aplicação da gamificação, sendo suas respectivas vantagens descritas na Tabela 8.

Tabela 8 - Vantagens de técnicas PBL.

Técnica - PBL	**Vantagens**
Pontos	Fornecer *feedback* constante e promover competição através de níveis, determinando as recompensas.
Emblemas e Conquistas	Apontam os níveis dos pontos ou os diferentes tipos de atividades e identificam grupos.
Quadros de Liderança	Apresentar a progressão dos jogadores, no entanto, se for aplicada individualmente pode desmotivar, por isso deve ser desenvolvida em grupos.

Fonte: Formanski (2016).

Chan (2018) descreve em mais detalhes cada uma das técnicas que compõem o PBL:

- Pontos: cerne de qualquer jogo. Inclui pontos de experiência, resgatáveis, habilidade e reputação. Os jogadores podem ganhar pontos quando cumprem as atividades específicas do jogo. Pontos resgatáveis servem para trocá-los em algo que se deseja; pontos de habilidades são atribuídos com base na competência de um jogador ou equipe na conclusão das atividades propostas; pontos de reputação atuam na confiança entre o jogo e o jogador;
- Emblemas ou Distintivos: incentivam e oferecem reconhecimento aos jogadores. Podem ser usados crachás para indicar a conclusão de metas. Pode substituir os níveis;
- Quadro de Liderança ou Placares: São tabelas de classificação que permitem ao jogador realizar comparações com outros jogadores. São classificadas em dois tipos:

- Sem desincentivo - não mostra a classificação literal dos jogadores;
- Classificação infinita - mostra todos os jogadores ordenados pela classificação, podendo trazer efeitos positivos junto com o alto grau de pressão para que os jogadores continuem indo bem.

2.2.3.6.1.3. Autoestima

Gratificações que elevam a autoestima do jogador para que ele continue envolvido no jogo, de forma que seja reconhecido as realizações dos jogadores e uma forma visível de parabenizá-lo (BURKE, 2015).

2.2.3.6.1.4. Capital social

Normalmente as pessoas são motivadas quando são reconhecidas por outras em seus meios sociais. A solução de gamificação pode criar formas de reconhecimento via gratificações que possam ser expostas aos demais colegas ou aos próprios outros jogadores sendo um motivador poderoso (BURKE, 2015).

2.3. Trabalhos relacionados

Esta seção tem como objetivo apresentar trabalhos, realizados previamente, que estão relacionados ao tema deste trabalho.

Durante a pesquisa, foram encontrados diversos trabalhos que aplicavam ferramentas gamificadas, entretanto o processo para se desenvolver a ferramenta como também o processo de decidir qual atividade, contexto e público alvo foi pouco explorado, neste caso foi considerada como maior referência a obra de Burke (2015).

2.3.1. Um jogo de refatoração. Estudando o impacto da ludificação na refatoração de *software*

O trabalho de Elezi *et al.* (2016) utiliza uma ferramenta de gamificação criada pelos autores que rastreia e recompensa os desenvolvedores quando estes aplicam atividades de refatoração utilizando a IDE Eclipse.

A ferramenta foi construída com base nos elementos de gamificação: Mural de atividades, Pontos e Quadros e Barra de Progresso, focando assim no feedback contínuo para os usuários.

Para cada atividade de refatoração executada a ferramenta atribui uma pontuação, sendo que quanto mais o desenvolvedor executa a refatoração, mais pontos ele conquista. A pontuação é exposta em um quadro de líderes que demonstra qual desenvolvedor está liderando o grupo, de forma que podem comparar seus pontos com os demais.

O trabalho aplica a ferramenta em um grupo de 12 estudantes do curso de informática da Universidade de Antuérpia e, através do estudo, os autores respondem às questões:

- "Podemos incorporar elementos de jogo como pontuação e tabelas de liderança em uma ferramenta não intrusiva de refatoração de *software*?"
    - A resposta obtida para essa pergunta foi que é possível incorporar elementos de jogos em uma ferramenta de gamificação não intrusiva, pois os desenvolvedores aceitaram a ferramenta durante a refatoração do código.
- "Como os elementos do jogo afetam o processo de refatoração?"
    - Durante o experimento, foram identificados inconvenientes durante o uso da ferramenta o que fez que os pontos atribuídos não foram condizentes com a refatoração em si, mas como uma situação criada apenas para ganhar pontos.
- "Os desenvolvedores estão dispostos a usar uma ferramenta?"
    - Após a aplicação da ferramenta, os autores utilizaram um formulário para coletar *feedbacks* do uso, através das respostas concluíram que os desenvolvedores não iriam utilizar a ferramenta em seu ambiente de trabalho.

Apesar do trabalho de Elezi *et al.* (2016) aplicar uma ferramenta que utilizou os elementos da gamificação, não deixa claro quais foram as etapas utilizadas para definir a ferramenta de gamificação, bem como a definição dos elementos de gamificação que podem ter maior efeito para atingir os objetivos propostos.

2.3.2. *CleanGame*: Gamificando a identificação de *Code Smells*

O trabalho de Santos *et al.* (2019) utiliza a ferramenta gamificada *CleanGame* para auxiliar estudantes universitários de Engenharia de *Software* a identificar *code smells* em um código aleatório.

Utilizaram a ferramenta após o treinamento de refatoração e *code smells*. O experimento foi feito com 18 estudantes e concluiu que, utilizando a ferramenta gamificada, foram identificados mais trechos de códigos do que quando utilizaram apenas a IDE.

Para avaliar a eficácia da gamificação utilizaram formulário contendo questões referentes à experiência, o que mostrou que a maioria dos participantes apresentou uma atitude positiva com relação à ferramenta gamificada.

Embora o trabalho expõe os resultados do experimento utilizando uma ferramenta gamificada, não foram levados em consideração os elementos da gamificação escolhidos, bem como quais foram os critérios de seleção da ferramenta.

2.3.3. Impacto da gamificação no processo de revisão de código - um estudo experimental

O trabalho de Khandelwal *et al*. (2017) propõe a comparação de cinco ferramentas gamificadas e não gamificadas durante o processo de revisão de código, de forma a avaliar a identificação de *code smells*, possíveis problemas de codificação e utilidade dos comentários.

O principal objetivo é quantificar o impacto da gamificação no processo de revisão de código através de um experimento com 183 alunos do segundo ano do curso de Análise e *Design* de *Software* de Estrutura. O montante de alunos foi distribuído em 5 grupos para utilizar as ferramentas.

Após o experimento, os autores chegaram à conclusão de que não há impacto da gamificação no processo de revisão de código. Porém, ressaltaram que a motivação para gamificar um aplicativo varia de acordo com a atividade que será gamificada, o contexto em que será aplicada e os objetivos de negócio.

Os autores sugerem que estudos adicionais devem ser feitos considerando os aspectos de elementos de *design* de jogo, tais como atividade a ser gamificada, contexto e público alvo.

2.4. Considerações do capítulo

A refatoração de *software* é uma prática importante associada a métricas de qualidade de *software*, podendo ser feita de forma preparatória, para compreensão ou simplesmente para refazer o que foi feito. Porém, para realizar a refatoração há diversos estágios a serem seguidos.

Os estágios da refatoração possuem uma dependência com o estágio de identificar os trechos de códigos que sugerem uma refatoração, também conhecidos como *code smells.* Sem essa identificação não é possível prosseguir para os próximos estágios.

Os *code smells,* quando identificados, nem sempre são resolvidos. Cabe ao desenvolvedor decidir refatorar ou não o código, mesmo sabendo que há a necessidade de refatoração. A falta de engajamento do desenvolvedor com o compromisso de refatorar pode ser um problema.

Aumentar o engajamento do desenvolvedor, bem como motivar com que os *code smells* sejam refatorados, é um grande desafio. A gamificação propõe elementos cujo principal objetivo é engajar e motivar colaboradores, podendo ser uma aliada para promover a refatoração de código.

Foram encontrados trabalhos que aplicam ferramentas de gamificação durante a refatoração de códigos, porém não foram encontradas diretrizes de como determinar a ferramenta ou modelo de gamificação, não sendo possível verificar se a ferramenta utilizada é ou não a melhor solução para o cenário da empresa e para os colaboradores.

3. DIRETRIZES PARA GAMIFICAÇÃO

Estas diretrizes têm como objetivo auxiliar a implantação da gamificação no processo de desenvolvimento de *software*, com intuito de motivar e facilitar os desenvolvedores a executarem as etapas de refatoração de código durante o desenvolvimento de *software*.

Vale ressaltar que as diretrizes apresentadas podem ser alteradas e/ou adaptadas de acordo com a realidade, dinâmica e cultura da empresa. Conforme proposto no objetivo do trabalho, este capítulo está escrito no estilo de diretrizes, o qual foi desenvolvido para ser aplicado em empresas que realizam desenvolvimento de *software*.

As etapas nas quais essas diretrizes são divididas são:

1. Definição do processo de gamificação, detalhada na seção 3.1;
2. Implantação da gamificação, detalhada na seção 3.2;
3. Acompanhamento dos resultados da gamificação, detalhado na seção 3.3.

3.1. Definição do processo de gamificação

Para definir um processo de gamificação é necessário fazer o planejamento de modo que o processo estimule o engajamento e ofereça desafios, que sejam atrativos para o usuário e que possam alcançar resultados quantificáveis (NEIDENBACH *et al.*, 2020).

Conforme Burke (2015), os estágios de definição do processo de gamificação consistem em:

1. Definir público-alvo;
2. Definir objetivos;
3. Definir modelo de engajamento dos jogadores;
4. Definir ambiente;
5. Definir incentivos e recompensas.

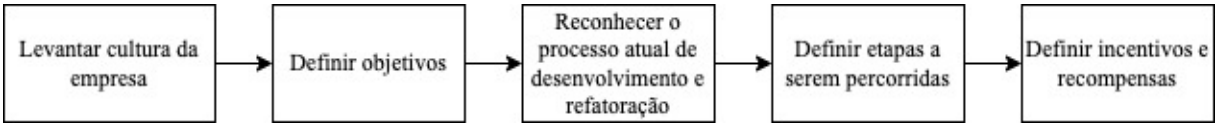
Nestas diretrizes, os estágios sugeridos por Burke (2015) foram adaptados para melhor aplicação no processo de desenvolvimento e refatoração especificamente,

dado que os estágios recomendados pelo autor são genéricos e não abordam aspectos inerentes às atividades de desenvolvimento de *software*. Os estágios a serem seguidos neste conjunto de diretrizes são:

1. Levantar cultura da empresa, apresentado no item 3.1.1;
2. Definir objetivos, apresentado no item 3.1.2;
3. Reconhecer o processo atual de desenvolvimento e refatoração, apresentado no item 3.1.3;
4. Definir etapas a serem percorridas, apresentado no item 3.1.4;
5. Definir incentivos e recompensas, apresentado no item 3.1.5.

A Figura 6 apresenta o diagrama dos estágios.

Figura 6 - Diagrama de estágios para implantação de gamificação



Fonte: Autora

Os estágios são descritos em detalhes nos seguintes itens.

3.1.1. Levantar a cultura da empresa

A observação da cultura da empresa é um passo importante que pode determinar o sucesso ou não da implantação da gamificação. Em uma cultura em que não há espaço para inovação, a implantação de um processo gamificado pode encontrar resistência interna.

A implantação bem sucedida exige uma colaboração da empresa para que o processo de gamificação seja melhorado de forma gradativa (GIMENEZ, 2021).

Formas de verificar a cultura da empresa:

- Busca no *website* da empresa;
- Entrevista com o responsável do recursos humanos;
- Entrevista com gestores.

Dado que o processo de gamificação necessita de uma forma de recompensa, é importante conhecer os benefícios oferecidos pela empresa aos colaboradores, de forma a identificar possibilidades de recompensas a serem oferecidas. Mais detalhes são apresentados na seção 3.1.4.

3.1.2. Definir objetivos

“O objetivo de melhorar o processo de refatoração é melhorar a qualidade de *software* estimada através das métricas de QMOOD (*Quality Model for Object-Oriented Design*)” (ALIZADEH *et al.*, 2020, Tradução Nossa). Portanto, um dos objetivos a serem alcançados são as próprias métricas de qualidade de *software* para projetos orientados a objetos (MOOD).

Conforme Neidenbach *et al*. (2020), para que o processo gamificado alcance os objetivos esperados, é necessário que seja atrativo para o usuário. Desta forma é preciso identificar os objetivos da equipe de desenvolvimento. Vale ressaltar que os objetivos devem ser realistas, alcançáveis e quantificáveis (BURKE, 2015).

Assim sendo, os objetivos do processo de gamificação são:

- Objetivo técnico: atingir valores aceitáveis quanto às métricas de qualidade de *software* para projetos orientados a objetos;
- Objetivo de engajamento: Engajar a equipe de desenvolvimento de *software* a realizar refatorações no *software*.

As formas de determinar os objetivos são descritos a seguir.

3.1.2.1. Métricas de qualidade de *software*

Para avaliar a qualidade de *software* será utilizada a extensão *MetricsTree* que ajuda a avaliar as propriedades quantitativas do código Java. Ele suporta os conjuntos de métricas mais comuns nos níveis de projeto, pacote, classe e método.
Conjuntos de métricas principais oferecidas pela extensão:

- Nível de projeto: Conjuntos de métricas MOOD e QMOOD, estatísticas gerais;
- Nível do pacote: Conjunto de métricas de Robert C. Martin, estatísticas gerais;

- Níveis de classe e método: Conjuntos métricos Chidamber-Kemerer, Lorenz-Kidd, Li-Henry e Lanza-Marinescu.

Serão consideradas as métricas referentes ao nível de projeto MOOD e QMOOD e será utilizada em conjunto com a IDE(*Integrated Development Environment*) Intellij .

A extensão utiliza as métricas MOOD descritas por Abreu (1995) e possui faixas de valores que identificam se a métrica está com valores elevados ou na média. Sendo elas apresentadas na Tabela 9:

Tabela 9 - Faixa de valores referência de métricas de qualidade.

Métrica MOOD	**Faixa de valores referência**
Método Fator de Ocultamento (MHF)	9,50% a 36,90%
Fator de Ocultamento de Atributos (AHF)	67,70% a 100,00%
Fator de Herança do Método (MIF)	60,90% a 84,40%
Fator de Herança do Atributo (AIF)	37,40% a 75,70%
Fator Polimorfismo (PF / POF)	1,70% a 15,10%
Fator de Acoplamento (CF / COF)	0,00% a 24,30%

Fonte: *MetricsTree (*2022*)*, adaptado pela Autora.

O objetivo técnico será melhorar as métricas de *design* orientado a objetos utilizando a refatoração de forma a manter os índices dentro das faixas de referência, pois impactam diretamente nos cálculos de QMOOD

Vale ressaltar que para estas diretrizes as métricas serão utilizadas em projetos orientados a objetos. Outras extensões que avaliam a qualidade de *software* também poderão ser utilizadas conforme a necessidade, como por exemplo a extensão *CodeMR* para as IDEs *Intellij* e *Eclipse*.

3.1.2.2. Equipe de desenvolvimento de *software*

Dado que a gamificação tem o potencial de impactar positivamente o desempenho, a produtividade e o engajamento do colaborador (SUH, 2015), é necessário

determinar os objetivos da equipe de desenvolvimento quanto à atividade de desenvolvimento.

Para identificar os objetivos da equipe é necessário identificar quais são as funções que compõem a equipe de desenvolvimento de *software*, sendo que o foco deve ser as funções que desenvolvem e/ou fazem manutenção do *software*.

Para engajar os colaboradores e tornar o processo gamificado atrativo ao colaborador, é necessário conhecer os desejos dos colaboradores com relação ao processo de desenvolvimento de *software* e desenvolvimento pessoal.

Um questionário pode ser utilizado para entender melhor os objetivos dos desenvolvedores de acordo com a perspectiva de cada participante e entender melhor o público-alvo de acordo com o nível de experiência.

Algumas perguntas podem ser observadas na Tabela 10. Vale ressaltar que esse é um modelo de questionário e pode ou não ser aplicado e/ou modificado, conforme a necessidade.

Tabela 10 - Perguntas para questionário

SEÇÃO	PERGUNTA	TIPO	RESPOSTAS	JUSTIFICATIVA
Identificar participante	Qual seu cargo?	Múltipla escolha	Lista de funções existentes na equipe de desenvolvimento, conforme levantado no item 3.1.1.	Identificar quem está preenchendo o formulário para que possa ter um meio de comparação.
Identificar participante	Possui quanto tempo de experiência no cargo indicado?	Múltipla escolha	Faixas de tempo: ● de 0 a 2 anos ● de 2 a 4 anos ● de 4 a 6 anos ● acima de 6 anos	Identificar quanto tempo de experiência o participante possui para que possa ter um meio de comparação.
Identificar objetivos	Atualmente, qual(is) seu(s) objetivo(s) profissional?	Mista: Múltipla escolha (mais de uma opção) e dissertativa	● Crescimento de carreira; ● Melhorar performance técnica; ● Desenvolver novas habilidades técnicas; ● Se tornar referência na atividade desenvolvida; ● Aumentar a qualidade técnica das soluções; ● Identificar oportunidades de melhoria técnica; ● Outros objetivos (opção dissertativa)	Identificar os objetivos do participante.
Identificar objetivos	Qual o maior desafio para se alcançar o(s) seu(s) objetivo(s)?	Dissertativa	Espaço para escrita do participante	Identificar os principais desafios enfrentados para se alcançar os objetivos.

Fonte: Autora.

3.1.3. Reconhecer o processo atual de desenvolvimento e refatoração

O processo de gamificação deve atender ao processo atual de desenvolvimento e refatoração, neste caso é necessário realizar a verificação do processo atual que é realizado pela equipe.

A verificação do processo atual de desenvolvimento e refatoração de *software* é feito através de acompanhamento das atividades junto à equipe de desenvolvimento, bem como entrevistas, pesquisa de ferramentas utilizadas, observação de rotinas, etc.

O processo atual impacta diretamente na implantação da gamificação, neste caso é necessário obter as seguintes informações:

- Qual metodologia de gerenciamento utilizada (*Scrum*, XP, Cascata, etc);
- Qual a estrutura da equipe - níveis dos desenvolvedores, gestores e líderes técnicos;
- Processo de desenvolvimento:
    - Como as demandas surgem;
    - Como as demandas são distribuídas;
    - Como é feito o versionamento de código;
    - Como é feita a verificação e aprovação do código
    - Quais métricas são utilizadas;
    - Como é feita a refatoração do código;
    - Em que momento o código é refatorado;
    - Quais documentos gerados durante o processo.
- Quais ferramentas de comunicação são utilizadas;
- Quais incentivos oferecidos para o desenvolvimento técnico da equipe;
- Quais são as formas de compartilhar informações técnicas e de negócio;

As informações coletadas servirão como base para definir as etapas seguintes, descritas nos itens 3.1.4 e 3.1.5.

3.1.4. Definir etapas a serem percorridas

Com base nos objetivos levantados por meio das diretrizes da seção 3.1.2, deve-se criar tarefas que estimulem, sejam atrativas e desafiadoras para envolver e motivar os colaboradores (PATRÍCIO, 2018).

Dado que um dos objetivos é melhorar as métricas de qualidade de *software* utilizando refatoração, as etapas a serem percorridas são os próprios estágios da refatoração citados por Kaur *et al.* (2019) durante sua pesquisa, sendo eles:

- Identificar locais de código onde o *software* deve ser refatorado;
- Determinar as atividades apropriadas de refatoração a serem aplicadas;
- Garantir que as atividades de refatoração aplicadas preservem o comportamento do *software*;
- Aplicar as atividades de refatoração selecionadas;
- Avaliar o impacto das atividades de refatoração na qualidade do *software*;
- Manter a consistência entre o *software* refatorado e outros artefatos de *software*;

Para transformar as etapas de refatoração em tarefas que sejam inspiradoras e significativas para os colaboradores, é necessário determinar os elementos do jogo (PATRÍCIO, 2018):

- Dinâmica - fornece motivação, exemplo: narrativa, progressão, interação social;
- Mecânica - fornece procedimentos básicos que impulsionam o envolvimento dos colaboradores;
- Engajamento dos colaboradores - fornece desafios, competição, colaboração, recompensas e turnos. Conforme exemplos apresentados na seção 3.1.5.

Formanski (2016) exemplifica a dinâmica e mecânica na Tabela 11, porém para essas diretrizes a dinâmica será utilizada como uma narração da tarefa a ser executada e a mecânica é dada como o passo a passo, também chamado de desafio, esperado para se executar a tarefa.

Tabela 11 - Técnicas de dinâmica e mecânica

Dinâmicas	**Mecânicas**
Coerção	Desafios
Emoções	Chance
Narrativa	Competição
Progressão	Cooperação
Relações	Feedback
Conquistas	Níveis
Avatares	Recompensas
	Liderança
	Emblemas (*Badges*)
	Pontos

Fonte: Formanski (2016)

Como o objetivo é melhorar o processo de refatoração é necessário criar um hábito de refatoração oportunista, de forma que seja parte da tarefa de programação (FOWLER, 2019). Desta forma as tarefas a serem percorridas serão aplicadas por campanha, ou seja, aplica uma tarefa durante um determinado tempo, até que se torne um hábito e prossegue para a próxima tarefa assim sucessivamente até que todas as tarefas sejam concluídas.

O período da campanha para cada tarefa pode variar de acordo com a característica da equipe, tamanho e aderência ao processo de gamificação. É proposto que as etapas de identificar locais de código onde o *software* deve ser refatorado e determinar as atividades apropriadas de refatoração permaneçam juntas pois são complementares. Caso a empresa utilize metodologias ágeis para o gerenciamento de seus projetos, o período da campanha pode acompanhar as iterações dos projetos, ou até que se crie o hábito da execução da tarefa.

A quantidade de pontos referente ao engajamento é uma sugestão, podendo ser alterada conforme critério da empresa.

Na Tabela 12 são apresentados exemplos de como os elementos de jogo podem ser utilizados em cada etapa de refatoração. Ressalta-se que as tarefas podem ser aplicadas de maneira isolada ou em conjunto, assim como obedecendo a sequência ou não, conforme a dinâmica de cada equipe de desenvolvimento.

Tabela 12 - Tarefas a serem percorridas e seus elementos.

TAREFA (KAUR *et al.* , 2019)	**MOMENTO**	**DINÂMICA**	**MECÂNICA**	**ENGAJAMENTO**
1. Identificar locais de código onde o *software* deve ser refatorado.	Verificação do código através da ferramenta de versionamento de código.	Durante a verificação do código de outro desenvolvedor, o desenvolvedor identifica o *code smell.*	1- Ter conhecimento dos tipos mais comuns de *code smells* existentes; 2- Verificar código de outros desenvolvedores; 3- Identificar através de comentários, na ferramenta de versionamento, o *code smell.*	Colaboração: 1- Para cada código verificado de outra equipe, soma-se 3 pontos; 2- A equipe que verificar mais códigos de outras equipes ganham 3 pontos.
1. Identificar locais de código onde o *software* deve ser refatorado.	Manutenção do *software*.	Durante a manutenção, o desenvolvedor identifica *code smells.*	1- Ter conhecimento dos tipos mais comuns de *code smells* existentes; 2- Avaliar a classe em que será feita a manutenção de forma a identificar *code smells;* 3- Criar histórico corrigindo o *code smells* na ferramenta de versionamento (exemplo: criar *commits* de refatoração de códigos).	Desafio: 1- Deixar o código mais limpo do que quando foi encontrado, equivale a 1 ponto.

Fonte: Autora

Tabela 12 - Tarefas a serem percorridas e seus elementos - continuação

TAREFA (KAUR *et al.*, 2019)	**MOMENTO**	**DINÂMICA**	**MECÂNICA**	**ENGAJAMENTO**
2. Determinar as atividades apropriadas de refatoração a serem aplicadas.	Verificação do código através da ferramenta de versionamento de código.	Durante a verificação do código de outro desenvolvedor, o desenvolvedor sugere uma refatoração.	1- Ter conhecimento das principais atividades de refatoração; 2- Sugerir uma atividade de refatoração equivalente ao tipo de *code smell* identificado.	Colaboração: 1- Para cada atividade sugerida para outra equipe, soma-se 3 pontos; 2- A equipe que sugerir mais atividades de refatoração em códigos de outras equipes ganham 3 pontos.
2. Determinar as atividades apropriadas de refatoração a serem aplicadas.	Manutenção do *software*.	Durante a manutenção, o desenvolvedor determina as atividades de refatoração.	1- Ter conhecimento das principais atividades de refatoração; 2- Aplicar uma atividade de refatoração equivalente ao tipo de *code smell* identificado; 3- Criar histórico corrigindo o *code smells* na ferramenta de versionamento (exemplo: criar *commits* de refatoração de códigos).	Desafio: 1- Deixar o código mais limpo do que quando foi encontrado, equivale a 1 ponto.

Fonte: Autora

Tabela 12 - Tarefas a serem percorridas e seus elementos - continuação

TAREFA (KAUR *et al.*, 2019)	**MOMENTO**	**DINÂMICA**	**MECÂNICA**	**ENGAJAMENTO**
3. Garantir que as atividades de refatoração aplicadas preservem o comportamento do *software*	Desenvolvimento de *software*	Durante o desenvolvimento do *software* o desenvolvedor cria testes que garantem o comportamento do código em casos de refatoração	1- Criar testes conforme as condições descritas; 2- Criar testes unitários das classes; 3- Atingir a meta de cobertura de testes (quando houver).	Desafio: 1- Maior cobertura de testes em todas as classes, soma-se 1 ponto; 2- Todas as condições foram testadas, soma-se 2 pontos
3. Garantir que as atividades de refatoração aplicadas preservem o comportamento do *software*	Manutenção do *software*.	Após a refatoração, os testes existentes possuem resultados de sucesso	1- Antes da refatoração todos os testes devem estar funcionando com resultados de sucesso; 2- Caso o código não possua testes, deve-se criar um teste antes da refatoração garantindo que o código manterá o mesmo comportamento; 3- Ajustar testes caso necessário.	Desafio: 1- Todos os testes continuam com resultados de sucesso, soma-se 1 ponto; 2- Aumenta a cobertura de testes após a refatoração, soma-se 1 ponto.

Fonte: Autora

Tabela 12 - Tarefas a serem percorridas e seus elementos - continuação

TAREFA (KAUR *et al.*, 2019)	**MOMENTO**	**DINÂMICA**	**MECÂNICA**	**ENGAJAMENTO**
3. Garantir que as atividades de refatoração aplicadas preservem o comportamento do *software*	Verificação do código através da ferramenta de versionamento de código.	Durante a verificação do código de outro desenvolvedor, são sugeridos testes para garantir o comportamento.	1- Compreender o código; 2- Verificar se todas as condições estão sendo testadas; 3- Sugerir a criação de novos testes quando necessário.	Colaboração: 1- Sugerir a criação de testes, soma-se 2 pontos;
4. Aplicar as atividades de refatoração selecionadas	Desenvolvimento de *software* e/ou Manutenção do código	Durante a manutenção ou desenvolvimento, o desenvolvedor determina e aplica às atividades de refatoração conforme o *code smell*.	1- Ter conhecimento das principais atividades de refatoração; 2- Aplicar uma atividade de refatoração equivalente ao tipo de *code smell* identificado; 3- Criar histórico corrigindo o *code smells* na ferramenta de versionamento (exemplo: criar *commits* de refatoração de códigos).	Desafio: 1- Deixar o código mais limpo, equivale a 1 ponto.

Fonte: Autora

Tabela 12 - Tarefas a serem percorridas e seus elementos - continuação

TAREFA (KAUR *et al.*, 2019)	**MOMENTO**	**DINÂMICA**	**MECÂNICA**	**ENGAJAMENTO**
4. Aplicar as atividades de refatoração selecionadas	Verificação do código através da ferramenta de versionamento de código.	Quando o código do desenvolvedor possui uma sugestão de refatoração, esta é aplicada.	1- Avaliar se a sugestão da atividade de refatoração é coerente com o code smell identificado; 2- Aplicar a atividade sugerida; 3- Solicitar verificação do código novamente.	Colaboração: 1- Interagir com outras equipes, soma-se 3 pontos.
5. Avaliar o impacto das atividades de refatoração na qualidade do *software*	Desenvolvimento de *software* e/ou Manutenção do código	Quando realizar alguma refatoração, verificar se houve alteração nas métricas de qualidade.	1- Verificar as métricas de qualidade do *software* utilizando a ferramenta existente. Exemplo: *MetricTree*, *MetricsReloaded*, ambos para IDE *Intellij*; 2- Refatorar o código; 3- Verificar novamente as métricas de qualidade do *software* para comparar se houve alterações positivas e/ou negativas.	Desafio: 1- Melhorar as métricas de qualidade de código, soma-se 5 pontos.

Fonte: Autora

Tabela 12 - Tarefas a serem percorridas e seus elementos - continuação

TAREFA (KAUR *et al.*, 2019)	**MOMENTO**	**DINÂMICA**	**MECÂNICA**	**ENGAJAMENTO**
6. Manter a consistência entre o *software* refatorado e outros artefatos de *software*.	Desenvolvimento de *software* e/ou Manutenção do código	Garantir que o *software* refatorado funcione com suas respectivas integrações;	1- Realizar testes integrados.	Colaboração: 1- Não houve problemas de integração com outras equipes, soma-se 5 pontos; 2- Houve testes com todas as integrações de outras equipes, soma-se 5 pontos.
6. Manter a consistência entre o *software* refatorado e outros artefatos de *software*.	Desenvolvimento de *software* e/ou Manutenção do código	Se houve alguma mudança significativa durante o processo de refatoração, deve-se documentar (quando possível)	1- Alterar documentação e outros artefatos do *software*.	Desafio: 1- Manter a documentação e artefatos do *software* atualizados, soma-se 5 pontos.

Fonte: Autora

3.1.5. Definir incentivos e recompensas

Os incentivos e recompensas são uma parte importante que influenciam diretamente no engajamento do colaborador com o processo de gamificação.

3.1.5.1. Incentivos

Para incentivar os colaboradores, um placar de pontos deve ser criado e manter disponível nos canais de comunicação da empresa de forma que seja de fácil acesso.

A definição dos pontos adotada nestas diretrizes são valores de 1 a 5, cuja descrição de cada valor é demonstrada a seguir, conforme os passos que foram definidos na Tabela 12:

- 1 - Tarefa não executada;
- 2 - Tarefa executada, porém as mecânicas não foram seguidas;
- 3 - Tarefa e mecânicas executadas;
- 4 - Tarefa, dinâmicas e mecânicas executadas;
- 5 -Tarefa, dinâmicas e mecânicas executadas dentro da equipe bem como prestou auxílio para demais equipes.

O placar de pontos pode ser por equipes e por colaboradores, a Tabela 13 apresenta um exemplo de como o placar por equipe pode ser apresentado.

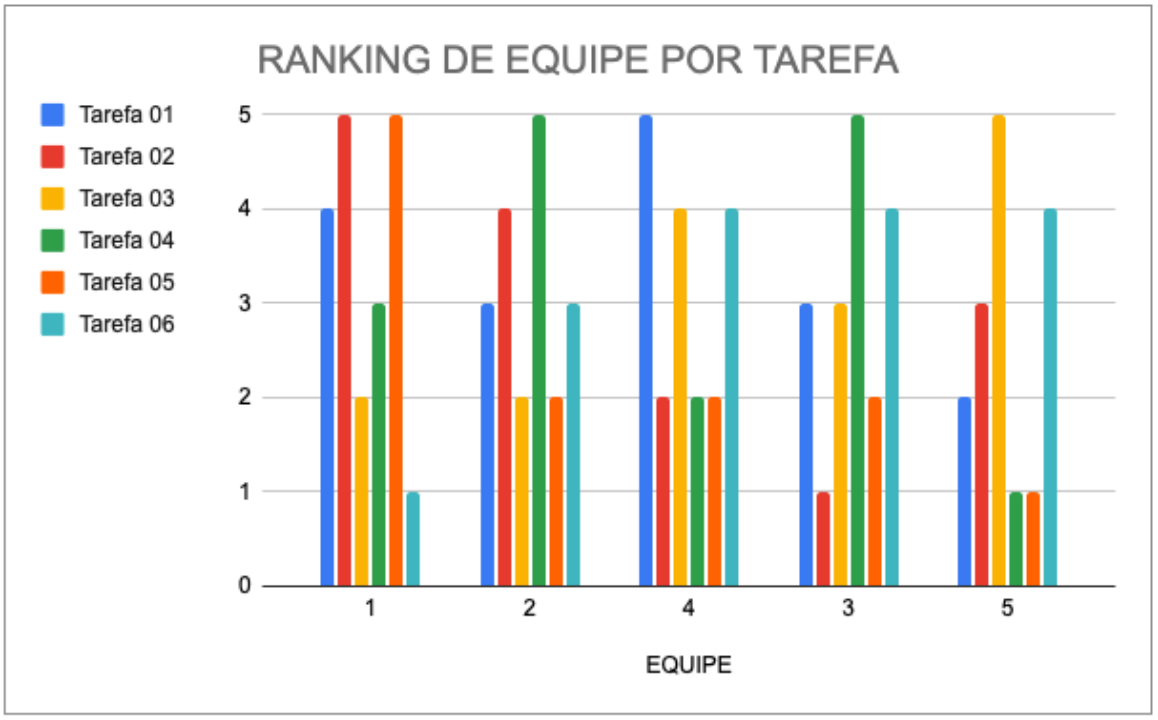
Tabela 13 - Placar de pontos por equipe.

EQUIPE	PONTUAÇÃO						
	Tarefa 01	Tarefa 02	Tarefa 03	Tarefa 04	Tarefa 05	Tarefa 06	TOTAL
1	4	5	2	3	5	1	20
2	3	4	2	5	2	3	19
4	5	2	4	2	2	4	19
3	3	1	3	5	2	4	18
5	2	3	5	1	1	4	16

Fonte: Autora

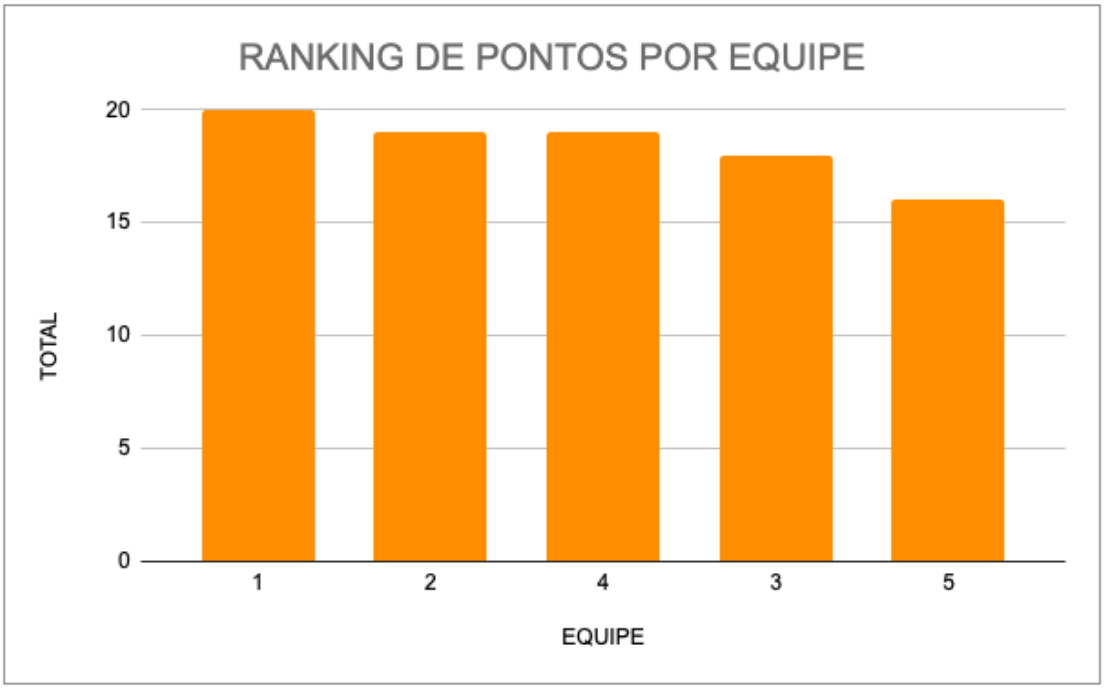
Os pontos podem ser acompanhados e divulgados através de gráficos, conforme exemplos apresentados nas Figuras 7 e 8.

Figura 7 - Exemplo de gráfico de acompanhamento de pontos por tarefas



Fonte: Autora.

Figura 8 - Exemplo de gráfico de acompanhamento de pontos totais



Fonte: Autora.

Há outras formas de incentivar os colaboradores, porém para estas diretrizes serão utilizados incentivos intrínsecos de forma a reconhecer o colaborador com maior pontuação durante a campanha da tarefa da gamificação. O reconhecimento deve partir de gestores diretos da equipe de desenvolvimento, bem como colegas de equipe de forma a incentivar o colaborador e as equipes.

3.1.5.2. Recompensas

As recompensas extrínsecas devem ser coerentes com os objetivos da equipe de desenvolvimento levantados pelas diretrizes da seção 3.1.2. Exemplo: grande parte da equipe de desenvolvimento possui o objetivo de melhorar desempenho técnico, neste caso uma possível recompensa seria um curso de um assunto técnico, um livro, dentre outras coisas.

Caso a cultura da empresa tenha pilares de qualidade de vida, uma possível recompensa seria um dia de folga ou vales compras, dentre outras.

Recompensas intrínsecas também podem ser utilizadas, tais como: reconhecimento do colaborador pela liderança seguido por prêmios por tal reconhecimento, etc.

3.2. Implantação das diretrizes de gamificação

Depois de finalizar a definição do processo de gamificação, o processo deve ser implantado. As etapas da implantação inicial consiste em:

1. Determinar métricas e formas de acompanhamento de pontos;
2. Escolher uma equipe específica para iniciar a implantação;
3. Determinar o *software* alvo para aplicação do processo gamificado;
4. Divulgar a implantação do processo gamificado para a equipe escolhida.

3.2.1. Determinar métricas e formas de acompanhamento de pontos

Para o sucesso do processo de gamificação, é necessário determinar métricas e formas de acompanhamento de pontos que serão realizadas durante a execução da gamificação.

Com finalidade de quantificar os resultados da melhora do processo de refatoração utilizando a gamificação é preciso definir quais os *softwares* que serão o foco da

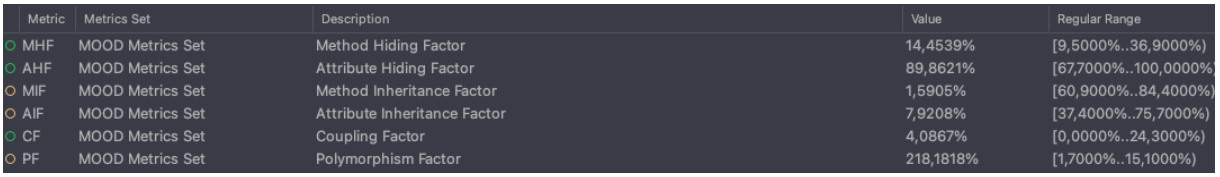
gamificação e realizar a verificação das métricas de qualidade destes *softwares* antes do início da gamificação.

Deve ser designado um responsável para realizar o acompanhamento da execução das tarefas e controle de pontos, o responsável deverá possuir acesso aos *softwares* escolhidos.

O acompanhamento de pontos pode variar conforme o processo de desenvolvimento de cada equipe e empresa. Porém, caso a empresa utilize controles de versionamento e repositórios, é possível realizar o acompanhamento das tarefas através de comentários na ferramenta de versionamento. Há diversas extensões para IDEs para acompanhar as alterações realizadas pelos colaboradores que podem ser utilizadas para auxiliar o controle de pontos, tais como: *GitBlame*, *GitToolBox*, dentre outras.

O acompanhamento das métricas poderá ser feito através da análise de métricas de qualidade de *software* dos projetos utilizando extensões. A Figura 9 apresenta as métricas de qualidade de *software* de um *software* orientado a objetos utilizando o *MetricsTree* para a IDE Intellij.

Figura 9 - Métricas MOOD *MetricsTree* Intellij



Metric	Metrics Set	Description	Value	Regular Range
MHF	MOOD Metrics Set	Method Hiding Factor	14,4539%	[9,5000%..36,9000%)
AHF	MOOD Metrics Set	Attribute Hiding Factor	89,8621%	[67,7000%..100,0000%]
MIF	MOOD Metrics Set	Method Inheritance Factor	1,5905%	[60,9000%..84,4000%)
AIF	MOOD Metrics Set	Attribute Inheritance Factor	7,9208%	[37,4000%..75,7000%)
CF	MOOD Metrics Set	Coupling Factor	4,0867%	[0,0000%..24,3000%)
PF	MOOD Metrics Set	Polymorphism Factor	218,1818%	[1,7000%..15,1000%)

Fonte: Autora.

Para melhor comparação dos resultados antes e depois da implantação do processo gamificado, as métricas de qualidade do *software* escolhido devem ser medidas antes de iniciar a aplicação do processo gamificado. Vale destacar que as métricas sugeridas são específicas para projetos orientados a objetivos. Porém, conforme o projeto escolhido para aplicação das diretrizes, essas métricas podem ser alteradas para melhor adaptação ao projeto, de forma a ser possível acompanhar a evolução da qualidade do projeto em si.

Recomenda-se que o acompanhamento das métricas de qualidade de *software* seja feito ao menos uma vez por semana ou quando houver iterações com o projeto.

3.2.2. Escolher uma equipe específica para iniciar a implantação

Recomenda-se que seja escolhida uma equipe para que seja feita uma rodada piloto do processo de gamificação para que sejam feitas as adequações necessárias.

A equipe escolhida deve representar uma amostra das demais equipes, ou seja, deve ser composta por desenvolvedores de diversos níveis de senioridade. Porém, recomenda-se que seja composta por até 6 desenvolvedores para melhor acompanhamento.

Nesta etapa é importante escolher uma equipe que tenha facilidade em fornecer *feedbacks* significativos para contribuir com a melhora do processo gamificado. A escolha da equipe precisa ser feita em conjunto com os gestores de cada equipe, a fim de que os objetivos a serem atingidos sejam de conhecimento a todos. O papel do gestor da equipe é de suma importância pois a implantação do processo gamificado pode impactar diretamente com a dinâmica da equipe durante os trabalhos realizados na campanha do processo gamificado. Neste caso é necessário apresentar ao gestor os objetivos, duração da campanha, as tarefas a serem feitas, as formas de acompanhamento para que o gestor, em conjunto com a pessoa responsável pelo processo de gamificação, auxilie e incentive a equipe durante a execução do processo gamificado.

3.2.3. Determinar o *software* alvo para aplicação do processo gamificado

Para as primeiras campanhas do processo gamificado, é importante focar apenas no *software* mais alterado pela equipe, para que o processo de acompanhamento de pontos seja realizado de forma mais controlada e representativa. Quanto mais *softwares* o processo gamificado é aplicado, mais pessoas são necessárias para realizar o acompanhamento de pontos.

3.2.4. Divulgar a implantação do processo gamificado

A divulgação da implantação do processo gamificado para a equipe de desenvolvimento envolvida deve ocorrer com antecedência a fim de que sejam feitas adequações caso necessário. Deve ser feita através de reuniões e e-mail e/ou pelos canais de comunicação. É feita antes de ser iniciado o processo gamificado, porém é importante que os acordos e informações cruciais sejam relembradas ao longo da campanha através dos canais de comunicação da empresa.

É de suma importância que sejam esclarecidas as tarefas que serão propostas no período da campanha e os pontos correspondentes a cada tarefa executada. As tarefas devem ficar disponíveis e de fácil acesso para eventuais consultas nos canais de comunicação utilizados pela empresa.

Durante a divulgação deve-se informar:

- O responsável pelo acompanhamento de pontos e métricas;
- As ferramentas que irão ser utilizadas para medir a qualidade do *software*;
- Obtenção de pontos;
- O local em que o placar ficará disponível;
- O período da campanha para cada tarefa ou da tarefa inicial;
- Os objetivos do jogo;
- As recompensas para os ganhadores.

Durante a campanha, podem surgir dúvidas, neste caso as informações sobre o processo devem ser mantidas nos canais de comunicação e divulgadas novamente semanalmente, principalmente nas campanhas iniciais.

3.2.5. Acompanhamento dos resultados da gamificação

O acompanhamento dos resultados da gamificação é avaliar se os objetivos definidos no item 3.1.2 estão sendo alcançados com o processo de gamificação. Para acompanhar os resultados quanto ao objetivo técnico deve ser feita a comparação das métricas de qualidade de *software* antes e depois do início do processo gamificado.

Quanto aos objetivos de engajamento, pode ser feita uma pesquisa quanto ao processo gamificado com a equipe de desenvolvimento participante. A Tabela 14 propõe um conjunto de perguntas para avaliar o processo gamificado e o engajamento dos colaboradores.

Tabela 14 - Proposta de perguntas para avaliação do processo gamificado

PERGUNTA	TIPO	RESPOSTAS	JUSTIFICATIVA
O que você achou do processo gamificado?	Múltipla escolha	● Muito bom ● Bom ● Razoável ● Ruim ● Muito ruim	Avaliar a qualidade do processo gamificado.
Você acredita que tem refatorado mais códigos do que antes do processo gamificado?	Múltipla escolha	● Sim ● Não ● Mesma coisa	Avaliar se o processo gamificado aumentou a prática da refatoração.
Você acredita que o processo gamificado motivou o seu crescimento técnico?	Múltipla escolha	● Sim ● Não ● Mesma coisa	Avaliar se o processo gamificado colaborou com objetivos técnicos.
Você acredita que a qualidade do código melhorou após a implantação da gamificação?	Múltipla escolha	● Sim ● Não ● Mesma coisa	Avaliar se o processo gamificado colaborou com a qualidade de *software* do ponto de vista do desenvolvedor.
Você teve mais interação com outros membros da equipe?	Múltipla escolha	● Sim ● Não ● Não precisei	Identificar se o processo gamificado contribuiu com a interação com membros da equipe.
Você teve mais interação com membros de outra equipe?	Múltipla escolha	● Sim ● Não ● Não precisei	Identificar se o processo gamificado contribuiu com a interação com membros de outra equipe.
O que você achou do sistema de recompensa?	Múltipla escolha	● Gostei muito ● Gostei ● Indiferente ● Ruim ● Muito ruim	Avaliar a qualidade das recompensas oferecidas.
Você se sentiu incentivado a prosseguir com o processo gamificado?	Múltipla escolha	● Sim ● Não	Avaliar os incentivos oferecidos.
Você tinha conhecimento de seu progresso de pontos nas tarefas?	Múltipla escolha	● Sim ● Não ● Não precisei	Avaliar a disponibilidade de progresso de pontos para o colaborador.

Fonte: Autora

Tabela 14 - Proposta de perguntas para avaliação do processo gamificado - continuação.

PERGUNTA	TIPO	RESPOSTAS	JUSTIFICATIVA
Você tem alguma sugestão de melhoria?	Dissertativa	Espaço para o colaborador sugerir melhorias para o processo	Receber *feedbacks*.
Teve algo que você não gostou?	Dissertativa	Espaço para o colaborador descrever o que não gostou do processo gamificado	Receber *feedbacks*.

Fonte: Autora

As perguntas sugeridas buscam oferecer uma visão de como foi a aceitação e o progresso da gamificação, bem como se os objetivos foram atingidos, podendo ter *feedbacks* para melhoria nas próximas campanhas.

3.3. Considerações do capítulo

As diretrizes para implantar um processo de gamificação podem variar conforme o contexto da empresa, cultura, atividade e público-alvo.

Foram sugeridas etapas para serem seguidas, mas podem ser adaptadas para melhor aplicação conforme a realidade de cada empresa. A necessidade da adaptação é identificada durante as etapas de definição e implantação das diretrizes, porém vale ressaltar que as adaptações não devem alterar os objetivos da empresa e dos colaboradores, mas devem ser feitas de modo a auxiliar na conquista do objetivo.

4. ESTUDO DE CASO

O estudo de caso foi realizado em um cenário real e tem como objetivo verificar a aplicabilidade das diretrizes para gamificação apresentadas no Capítulo 3, durante o processo de desenvolvimento de *software*, de forma a observar o aumento ou não da aplicação de atividades de refatoração do código após a aplicação das diretrizes, e consequentemente, gerar um impacto na qualidade do *software* refatorado.

4.1. Cenário

Para a aplicação do conjunto de diretrizes, foi convidada uma equipe de desenvolvimento que atua em uma empresa cujo segmento é o mercado logístico na América Latina. A empresa é responsável por desenvolver seu próprio *software*, desta forma não há contratação de terceiros em qualquer etapa do desenvolvimento.

A equipe de desenvolvimento convidada é composta por dois desenvolvedores de nível pleno, dois desenvolvedores de nível sênior, dois desenvolvedores de nível júnior, um líder de projetos e um líder técnico. Os membros da equipe são de locais geograficamente distintos do Brasil, sendo considerada como uma equipe de desenvolvimento distribuída. A equipe é responsável pelo desenvolvimento de parte do fluxo de entrada de produtos no armazém (denominado *check in*).

As aplicações são hospedadas na nuvem, a qual é gerenciada pela própria empresa. Também trabalham com diversas linguagens de programação, sendo Java a mais utilizada. Os códigos possuem diferentes níveis de complexidades e são, em sua grande maioria, projetos orientados a objetos.

Atualmente os trabalhos consistem em adicionar novas funcionalidades e corrigir problemas no sistema existente, de modo que seja propício aplicar o conjunto de diretrizes de gamificação para incentivar e motivar a equipe a executar a refatoração.

A análise proposta é baseada nas respostas coletadas durante a execução das diretrizes, bem como a análise das métricas de qualidade de *software* do projeto trabalhado.

Vale ressaltar que os nomes dos projetos, desenvolvedores, dentre outras informações que possam identificar informações confidenciais foram retiradas. O estudo de caso foi realizado durante 45 dias.

4.2. Estágios da gamificação

Conforme a seção 3.1, é necessário que os estágios propostos sejam seguidos para obter melhores resultados de engajamento durante a aplicação das diretrizes de gamificação.

4.2.1. Identificação da cultura da empresa

Conforme o item 3.1.1, é importante que os objetivos da empresa, bem como sua cultura, sejam alinhados com os objetivos da gamificação de forma que resulte em um maior aproveitamento das diretrizes.

Para identificação da cultura da empresa, foi feita uma breve pesquisa no site google.com, utilizando as palavras de pesquisa “cultura” mais o “*nome da empresa deste estudo de caso*”. O resultado obtido foi:

> *“*Cultura: Seis princípios guiam nossas ações
>
> Nosso DNA empreendedor é o eixo de uma empresa cuja cultura prioriza a diversidade, a autonomia e a criatividade.
>
> Trabalhamos para que nossos colaboradores e equipes se sintam protagonistas de seu desenvolvimento enquanto criam uma experiência única, centrada no usuário.
>
> Nossa estrutura, dinâmica e aberta ao risco, cria um ambiente estimulante e plural, que forma grandes líderes e atrai os melhores talentos da América Latina.*”*

Os 6 princípios da empresa são:

- Criamos valor para nossos usuários;
- Empreendemos assumindo os riscos;
- Executamos com excelência;

- Estamos em beta contínuo;
- Competimos em equipe para ganhar;
- Damos o máximo e nos divertimos.

Com base na cultura da empresa, as diretrizes de gamificação são compatíveis e podem se tornar aliadas para alcançar os princípios culturais.

Para entender a dinâmica da equipe quanto aos benefícios oferecidos pela empresa aos colaboradores e possíveis recompensas que podem ser utilizadas durante a gamificação, foi feita uma entrevista com o líder de projetos da equipe. O resultado da entrevista é apresentado abaixo:

***PERGUNTA**: Quais são os benefícios oferecidos pela empresa?*

RESPOSTA: A empresa oferece o pacote de benefícios oferecido pela maioria das empresas do mercado de tecnologia.

***PERGUNTA**: Há algum incentivo oferecido para a equipe especificamente?*

RESPOSTA: Temos o que chamamos de *team build* quando a equipe faz uma grande entrega, podemos oferecer experiências aos colaboradores, tais como vale presentes, almoços, etc.

A partir das respostas do líder de projetos, foi identificada a possibilidade de se utilizar recompensas durante a aplicação de diretrizes, de forma a incentivar os colaboradores a participarem e cumprirem os objetivos da gamificação.

4.2.2. Definição de objetivos

De acordo com o item 3.1.2, a gamificação deve oferecer objetivos atrativos para a equipe e para a qualidade do *software*.

Nesta etapa foram feitas entrevistas com o líder técnico, líder de projetos e com a própria equipe de desenvolvimento. Durante as entrevistas com os líderes, foi definido que a implementação das diretrizes de gamificação seria feita de modo incremental para avaliar se é adequada ao fluxo de atividades atual e avaliar a aplicabilidade da gamificação.

Visando a cultura da empresa e o princípio “Estamos em beta contínuo”, a implantação das diretrizes de gamificação será feita focada em apenas uma parte do processo de desenvolvimento, sendo feita em pequenas partes incrementais, conforme a evolução e adaptação da equipe.

4.2.2.1. Objetivos técnicos

Os objetivos técnicos são baseados no conjunto de métricas de qualidade de *software*, conforme sugerido no item 3.1.2.1. Isto posto, foi feita uma entrevista com o líder técnico para avaliar qual(is) métrica(s) será(ão) considerada(s) no primeiro momento da gamificação.

Dado que a implantação das diretrizes será feita em partes incrementais, o objetivo técnico inicial será ***melhorar uma ou mais métricas de qualidade da aplicação principal, responsável pelo software que faz a entrada dos produtos no armazém, denominada neste trabalho como check-in.***

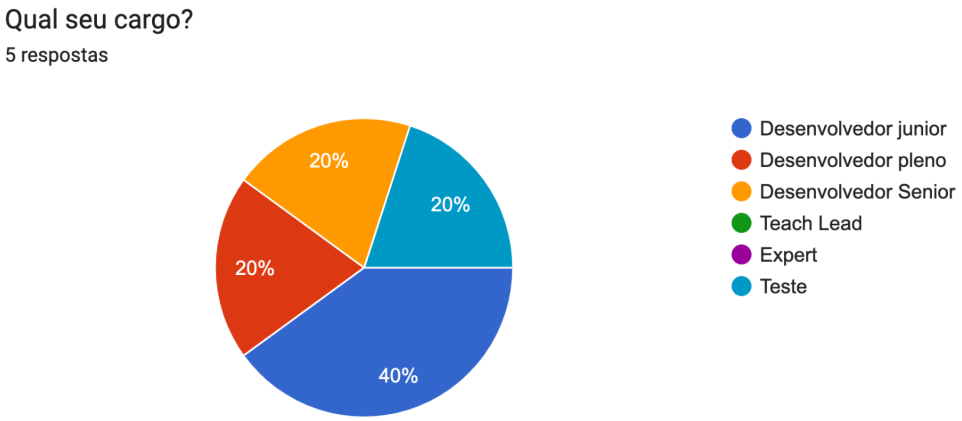
As métricas de qualidade de *software* (MOOD) serão medidas através do *plugin MetricsTree* da IDE *Intellij* , a mesma utilizada pela equipe.

4.2.2.2. Objetivos de engajamento

Para identificar os objetivos da equipe de desenvolvimento, foi aplicado um formulário elaborado através da ferramenta Google Forms (https://forms.gle/EUkWy5VoAmUWXSKR6) contendo as perguntas sugeridas na Tabela 10 do item 3.1.2.2. Vale ressaltar que as respostas são anônimas e alguns membros da equipe estavam de licença no momento em que o formulário foi aplicado, justificando a quantidade de resposta em comparação com a quantidade de membros na equipe.

A Figura 10 representa os cargos e níveis dos respondentes.

Figura 10 - Cargos respondentes

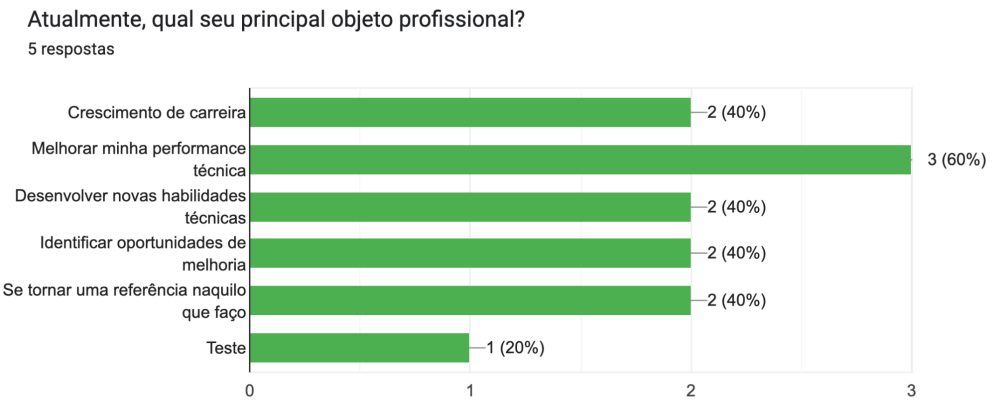


Fonte: Autora

Observa-se que há uma função "Teste" não prevista como resposta do formulário, porém foi indicada no campo "outros" não sendo possível vincular as demais respostas com o nível de senioridade do respondente. Também destaca-se que 40% das respostas foram dadas por desenvolvedores junior representando.

A Figura 11 apresenta os principais objetivos dos respondentes. Vale destacar que na pergunta em questão também houve uma resposta "Teste" não sendo possível avaliar essa resposta.

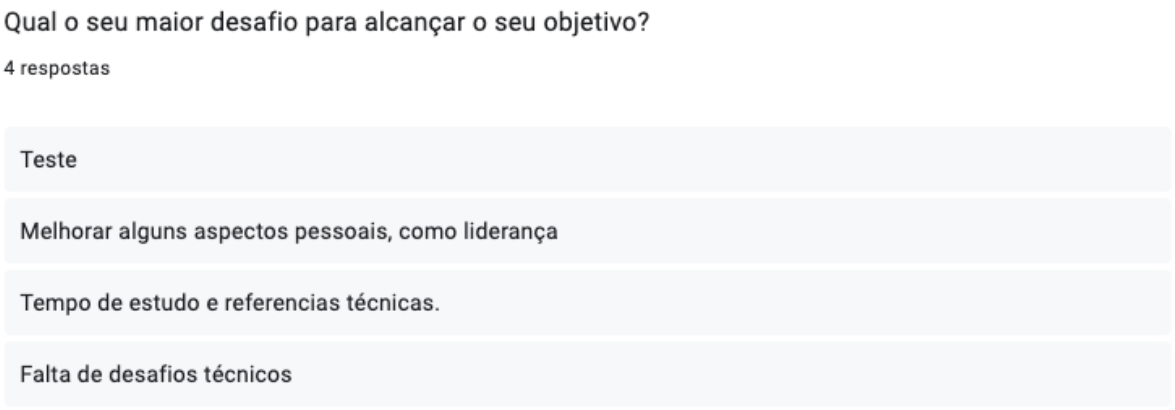
Figura 11 - Principais objetivos respondentes



Fonte: Autora

Vale apontar que a maioria dos respondentes possuem como objetivo "Melhorar a performance técnica". A Figura 12 apresenta o maior desafio para alcançar esse objetivo de forma dissertativa, neste caso também houve uma resposta "Teste" na qual não será considerada durante a análise.

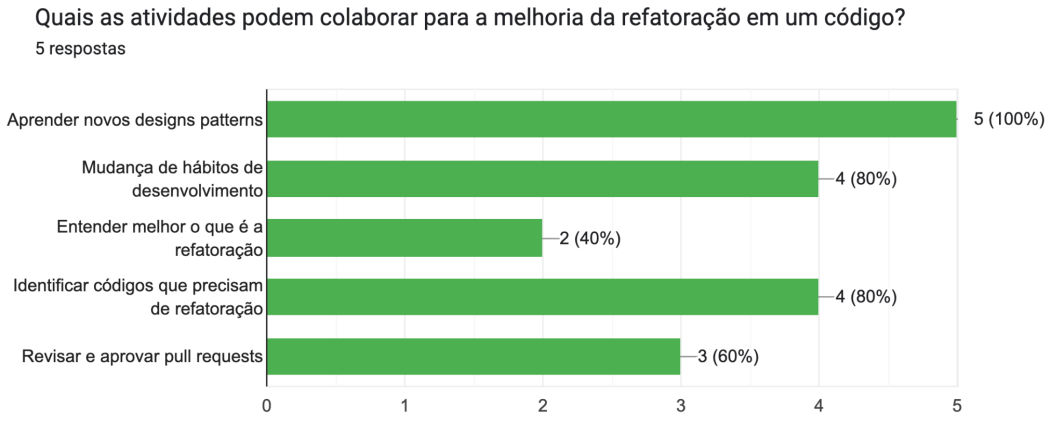
Figura 12 - Maior desafio para alcançar objetivos



Fonte: Autora

A seguir são apresentadas as respostas específicas referente à refatoração de *software*. A Figura 13 indica as atividades que podem colaborar para a melhoria da refatoração em um código.

Figura 13 - Atividades que podem colaborar com a melhoria da refatoração

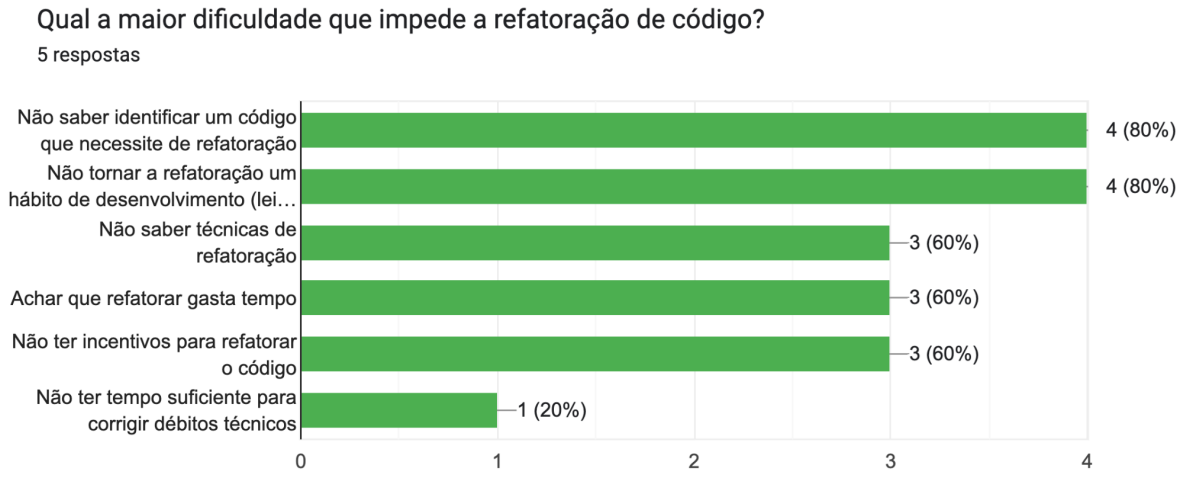


Fonte: Autora

Observa-se que a atividade "Aprender novos *design patterns*" foi indicada por todos os respondentes, seguida por "Mudança de hábitos de desenvolvimento" e "Identificar códigos que precisam de refatoração", ambas representando 80% das respostas.

A Figura 14 mostra as maiores dificuldades quanto à refatoração de código.

Figura 14 - Maiores dificuldades quanto à refatoração de código



Fonte: Autora

Observa-se que 80% dos respondentes afirmam que não saber identificar um código que necessite de refatoração e não tornar a refatoração um hábito são suas maiores dificuldades para realizar a refatoração de código.

Não houveram respostas relacionadas a sugestão de melhorias e considerações referente à refatoração de *software*.

Desta forma o objetivo que irá ser base para a gamificação de forma a aumentar o engajamento da equipe é:

Melhorar a performance técnica com base na criação de novos hábitos de refatoração e no aprendizado de novos designs patterns.

4.2.3. Processo atual de desenvolvimento e refatoração

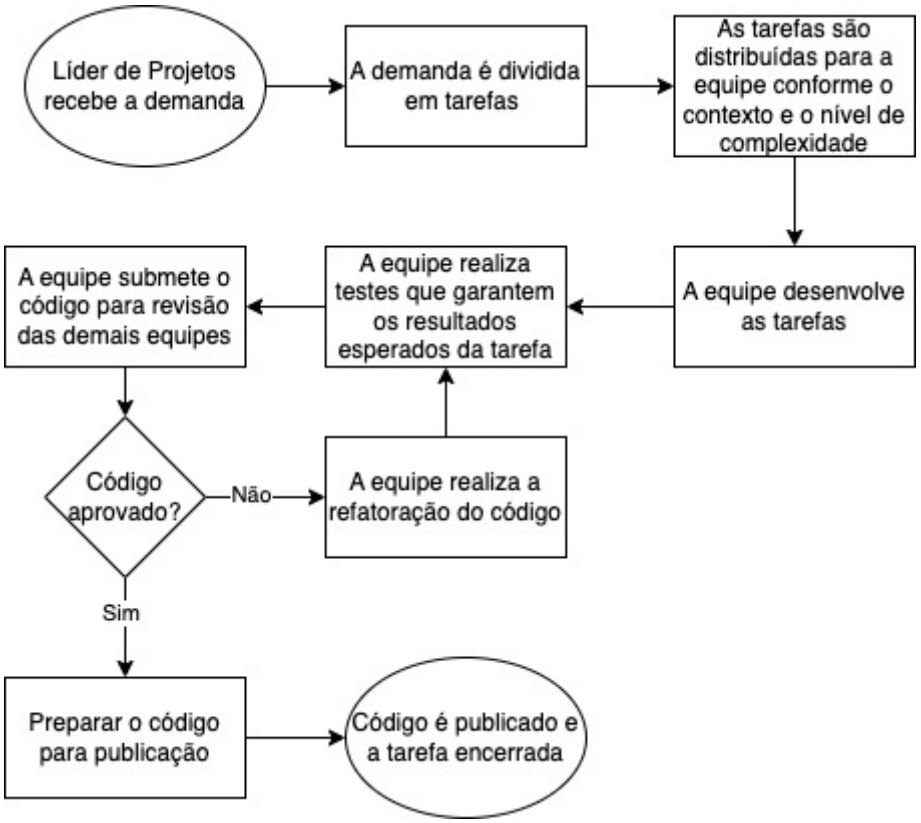
Com base nos tópicos propostos no item 3.1.3 a verificação do processo de desenvolvimento e refatoração realizado pela equipe de desenvolvimento foi feita por meio do acompanhamento das atividades e entrevista com o líder de projetos.

Durante a entrevista com o líder de projetos foram identificadas as informações abaixo:

- A metodologia de gerenciamento utilizada pela equipe é baseada no *Scrum* utilizando o *software Jira* para organização das tarefas e *sprints*;
- A estrutura da equipe é composta por:
    - Um líder de projetos - responsável por receber e delegar as demandas;
    - Um líder técnico - responsável por avaliar tecnicamente os projetos, decisões e monitorar a qualidade do *software*;
    - Dois desenvolvedores sênior;
    - Dois desenvolvedores plenos;
    - Dois desenvolvedores juniors.
- Ferramentas de comunicação utilizadas: *Slack* - programa de troca de mensagens e *Workplace* - programa de comunidade e grupos;
- Incentivos para o desenvolvimento técnico da equipe:
    - Realização de mesas de conversas sobre assuntos técnicos;
    - Grupos de estudos;
    - Descontos em plataformas de ensino *online*.
- O compartilhamento das informações técnicas e de negócio é feito através de *Google Docs, Google Sheets e Google Presentations*.

O processo macro de desenvolvimento e refatoração já existente na equipe pode ser observado na Figura 15. Ele foi elaborado com base nas informações coletadas durante o acompanhamento das atividades da equipe e entrevista com líder de projetos e líder técnico.

Figura 15 - Processo macro de desenvolvimento e refatoração



Fonte: Autora.

A equipe utiliza a plataforma *Github* para hospedar o código fonte, realizar a revisão e fluxo de aprovação e controle de versões.

A refatoração do código acontece durante as etapas de desenvolvimento, testes e revisão de código, quando identificada a necessidade

4.2.4. Etapas percorridas

Conforme definido nos objetivos (item 4.2.2) a implantação das diretrizes de gamificação será feita de forma incremental. Neste primeiro momento foi acordado com os líderes e com a equipe que as tarefas a serem implantadas conforme as diretrizes de gamificação serão as tarefas 1 e 2 descritas na Tabela 12, porém, foram adicionadas novas mecânicas visando aumentar o engajamento da equipe conforme o objetivo definido no item 4.2.2.2.

Foi determinado um responsável para acompanhar a aplicação de cada tarefa através da ferramenta de versionamento *Github*. A aprovação das solicitações de novas funcionalidades ou manutenção do código será considerada como um código apto para prosseguir no processo de publicação. Somente os desenvolvedores com mais experiência podem aprovar a solicitação de alteração de códigos.

A verificação da aplicação das etapas será feita por um integrante neutro da equipe e validado pelo líder técnico.

Tarefas iniciais conforme diretrizes:

1. Identificar locais de código onde o *software* deve ser refatorado
2. Determinar as atividades apropriadas de refatoração a serem aplicadas.

Para avaliar a aplicabilidade das diretrizes com base no objetivo da equipe, o objetivo foi desmembrado em pequenas partes e a Tabela 15 apresenta o objetivo e seus respectivos elementos de engajamento indicados na Tabela 12 do item 3.1.4.

Tabela 15 - Relação entre elementos de engajamento e objetivo da equipe

PARTE DO OBJETIVO	TAREFAS	ELEMENTO DE ENGAJAMENTO	JUSTIFICATIVA
Melhorar a performance técnica	1. Identificar locais de código onde o *software* deve ser refatorado.	A equipe que verificar mais códigos de outras equipes ganham 3 pontos.	Ao avaliar e sugerir melhorias em códigos de outras equipes, o desenvolvedor tem contato com outras formas de desenvolvimento e ideias criadas por outras equipes, aumentando assim o aprendizado e consequentemente a performance.
	2. Determinar as atividades apropriadas de refatoração a serem aplicadas.	A equipe que sugerir mais atividades de refatoração em códigos de outras equipes ganham 3 pontos.	
Criação de novos hábitos de refatoração	1. Identificar locais de código onde o *software* deve ser refatorado.	Deixar o código mais limpo do que quando foi encontrado, equivale a 1 ponto.	Ao incentivar o colaborador a deixar o código mais limpo de quando foi encontrado auxilia na criação do hábito
	2. Determinar as atividades apropriadas de refatoração a serem aplicadas.	Deixar o código mais limpo do que quando foi encontrado, equivale a 1 ponto.	
Aprendizado de novos *designs patterns.*	2. Determinar as atividades apropriadas de refatoração a serem aplicadas.	Não há elemento de engajamento indicado nas diretrizes	É necessário criar o elemento para atender esse objetivo

Fonte: Autora.

Dado que nas diretrizes não foi indicado um elemento de engajamento que atenda o objetivo "Aprender novos *design patterns*", foram incluídos os elementos na tarefa 2, conforme destacados na Tabela 16.

Tabela 16 - Tarefas iniciais a serem percorridas e seus elementos

TAREFA (KAUR *et al.*, 2019)	**MOMENTO**	**DINÂMICA**	**MECÂNICA**	**ENGAJAMENTO**
1. Identificar locais de código onde o *software* deve ser refatorado.	Verificação do código através da ferramenta de versionamento de código.	Durante a verificação do código de outro desenvolvedor, o desenvolvedor identifica o *code smell.*	1- Ter conhecimento dos tipos mais comuns de *code smells* existentes; 2- Verificar código de outros desenvolvedores; 3- Identificar através de comentários, na ferramenta de versionamento, o *code smell.*	Colaboração: 1- Para cada código verificado de outra equipe, soma-se 3 pontos; 2- A equipe que verificar mais códigos de outras equipes ganham 3 pontos.
1. Identificar locais de código onde o *software* deve ser refatorado.	Manutenção do *software*.	Durante a manutenção, o desenvolvedor identifica *code smells.*	1- Ter conhecimento dos tipos mais comuns de *code smells* existentes; 2- Avaliar a classe em que será feita a manutenção de forma a identificar *code smells;* 3- Criar histórico corrigindo o *code smells* na ferramenta de versionamento (exemplo: criar *commits* de refatoração de códigos).	Desafio: 1- Deixar o código mais limpo do que quando foi encontrado, equivale a 1 ponto.

Fonte: Autora

Tabela 16 - Tarefas iniciais a serem percorridas e seus elementos - continuação.

TAREFA	MOMENTO	DINÂMICA	MECÂNICA	ENGAJAMENTO
2. Determinar as atividades apropriadas de refatoração a serem aplicadas.	Verificação do código através da ferramenta de versionamento de código.	Durante a verificação do código de outro desenvolvedor, o desenvolvedor sugere uma refatoração.	1- Ter conhecimento das principais atividades de refatoração; 2- Sugerir uma atividade de refatoração equivalente ao tipo de *code smell* identificado.	Colaboração: 1- Para cada atividade sugerida para outra equipe, soma-se 3 pontos; 2- A equipe que sugerir mais atividades de refatoração em códigos de outras equipes ganham 3 pontos. **3- Para cada sugestão de aplicação de um *design pattern* a equipe ganha 5 pontos;**
2. Determinar as atividades apropriadas de refatoração a serem aplicadas.	Manutenção do *software*.	Durante a manutenção, o desenvolvedor determina as atividades de refatoração.	1- Ter conhecimento das principais atividades de refatoração; 2- Aplicar uma atividade de refatoração equivalente ao tipo de *code smell* identificado; 3- Criar histórico corrigindo o *code smells* na ferramenta de versionamento (exemplo: criar *commits* de refatoração de códigos).	Desafio: 1- Deixar o código mais limpo do que quando foi encontrado, equivale a 1 ponto. **2 - Para cada aplicação de *design pattern* no código, a equipe ganha 5 pontos.**

Fonte: Autora

Os elementos de engajamento criados visam incentivar os colaboradores a buscarem conhecimento referente à *design patterns* de forma a ganharem pontos no jogo.

4.2.5. Incentivos e recompensas

O placar de pontos será disponível através de uma planilha criada através da ferramenta *Google Sheets* e disponibilizada no grupo da equipe no *Slack.*

Os pontos foram validados pelo líder técnico de forma que fossem distribuídos conforme a aplicação das tarefas e respectivas mecânicas.

As recompensas foram definidas em conjunto com o líder de projeto, utilizando o *Team Build*, o jogador com maior pontuação ganhará um vale presente. O jogador com maior pontuação também receberá um selo, denominado como *Badges*, a cada selo que o jogador recebe, será divulgado em um grupo específico do *Workplace* da equipe

A seguir são apresentadas a implantação das diretrizes de gamificação durante as atividades de desenvolvimento e refatoração da equipe determinada neste estudo de caso.

4.3. Implantação do processo de gamificação

Conforme o item 3.2, com base nas definições realizadas no item 4.2, foi iniciado o processo de implantação das diretrizes de gamificação.

4.3.1. Métricas e formas de acompanhamento de pontos

De acordo com o item 3.2.1, foram definidas em conjunto com o líder de projeto e líder técnico, as métricas para quantificação dos resultados da implantação da gamificação, sendo elas:

- Aumento de revisão de códigos da própria equipe;
- Aumento de revisão de códigos de outra equipe;
- Aumento na aplicação de atividades de refatoração de código;

- Melhora de métricas de qualidade MOOD do *software* escolhido.

A Tabela 17 apresenta a forma pela qual as métricas serão acompanhadas.

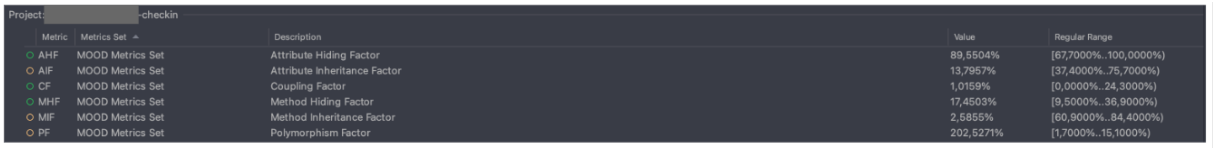
Tabela 17 - Métricas de acompanhamento de implantação das diretrizes de gamificação

MÉTRICA	FONTE DE INFORMAÇÃO	FORMA DE COLETA
Aumento de revisão de códigos da própria equipe;	*Pull requests* no GitHub da equipe;	Análise da quantidade de revisões de cada membro da equipe nos *Pull requests* da própria equipe
Aumento de revisão de códigos de outra equipe;	*Pull requests* no GitHub de outras equipes;	Análise da quantidade de revisões de cada membro da equipe nos *Pull requests* de outra equipe
Aumento na aplicação de atividades de refatoração de código;	*Pull requests* no GitHub da equipe contendo commits indicando refatoração; *Branchs* com o nome *enhancement*;	Análise da quantidade dos *commits* com a palavra refatoração e/ou *branches* com nome de *enhancement em Pull requests abertos pela equipe*
Melhora de métricas de qualidade MOOD do *software* escolhido.	MetricsTree Intellij	Análise de código utilizando o *plugin MetricsTree* da *branch* principal (develop) após aplicação da refatoração

Fonte: Autora

Para que as métricas de qualidade de software MOOD sejam avaliadas, foi necessário realizar a avaliação do projeto antes de iniciar a rodada da gamificação. A Figura 16 apresenta as métricas do projeto foco da gamificação antes do início da execução das tarefas da gamificação.

Figura 16 - Métricas do projeto antes de iniciar a execução das tarefas de gamificação.



Fonte: Autora.

4.3.2. Escolher uma equipe específica para iniciar a implantação

Com base no item 3.2.2, a escolha da equipe foi definida com base na facilidade da equipe de fornecer *feedbacks* significativos referente à implantação das diretrizes de gamificação e cujo líderes são de fácil acesso de forma a facilitar a interação e verificação de informações necessárias.

A equipe de desenvolvimento é composta por dois desenvolvedores de nível pleno, dois desenvolvedores de nível sênior, dois desenvolvedores de nível júnior, um líder de projetos e um líder técnico. Os membros da equipe são de locais geograficamente distintos do Brasil, sendo considerada como uma equipe de desenvolvimento distribuída.

4.3.3. Determinar o *software* alvo para aplicação do processo gamificado

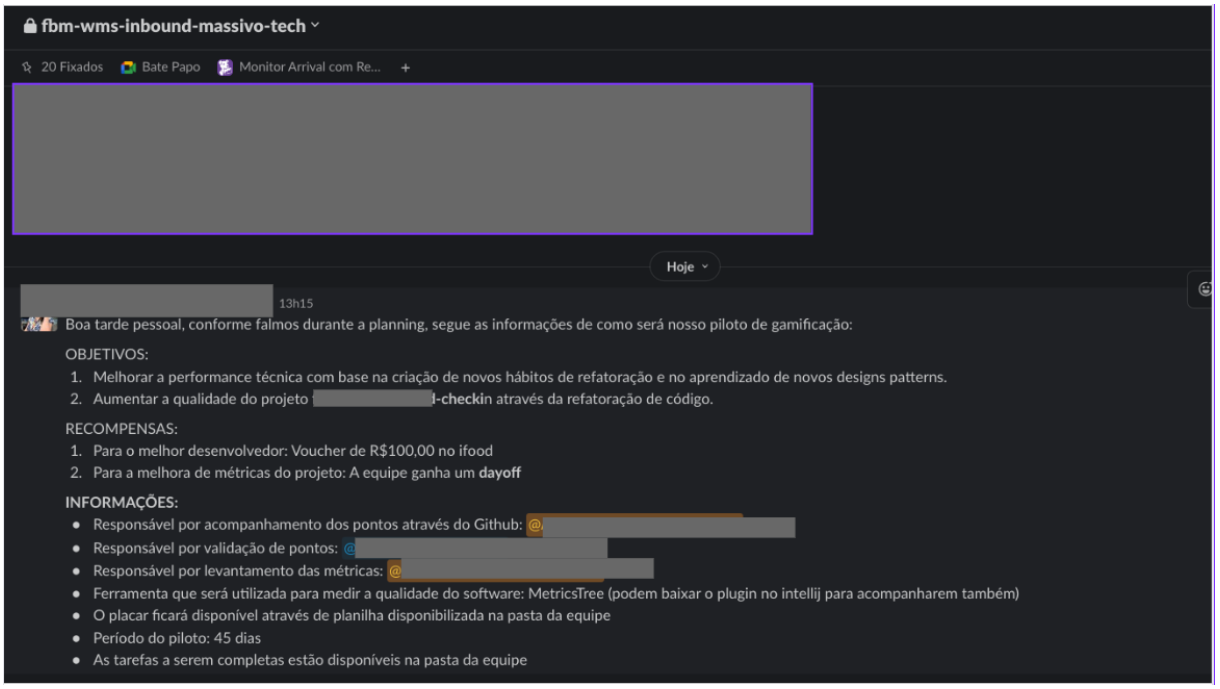
No primeiro momento da implantação das diretrizes de gamificação, conforme indicado no item 3.2.3, foi escolhido o *software* responsável por parte do fluxo de entrada de produtos no armazém (denominado como *check in*). O código foi desenvolvido na linguagem de programação Java e é um projeto orientado a objetos.

4.3.4. Divulgação da implantação das diretrizes para gamificação

Dado que a equipe utiliza metodologias ágeis, a divulgação da implantação das diretrizes para gamificação de acordo com o item 3.2.4 ocorreu durante a reunião de planejamento da *sprint* (ciclo de iteração do projeto). Foram apresentadas as informações referentes ao processos de gamificação e como seria a dinâmica para

aplicação da gamificação. O resultado da divulgação foi disponibilizado no grupo da equipe na ferramenta *Slack*, conforme apresentado na Figura 17.

Figura 17 - Divulgação da gamificação para a equipe.



Fonte: Autora

4.3.4.1. Alterações sugeridas durante a divulgação

Durante a divulgação das diretrizes, foi sugerido pela equipe que a pontuação fosse dada conforme cada mecânica executada de cada tarefa.

Quanto à pontuação de engajamento, será feita por membro da equipe, não por equipe, dado que nesse primeiro momento este estudo de caso refere-se a somente uma equipe. Foi alterada a pontuação do engajamento “1- Deixar o código mais limpo do que quando foi encontrado, equivale a 1 ponto.” para 3 pontos.

Foram sugeridas alterações na disposição do placar, conforme sugerido na Tabela 13, os pontos foram distribuídos por tarefa, porém como será feita uma rodada de testes utilizando somente as tarefas 1 e 2 conforme Tabela 16, a pontuação será apresentada apenas por jogador e rodada.

Desta forma a pontuação sugerida no item 3.1.5.1 foi adaptada de acordo com o que a equipe e com as atividades executadas.

4.3.5. Acompanhamento dos resultados da gamificação

Baseado no item 3.2.5, deve-se acompanhar os resultados da gamificação para avaliar se os objetivos definidos no item 4.2.2. O acompanhamento dos resultados será feito durante a revisão do código conforme os *Pull requests* do *Github* solicitados para o projeto em questão.

Cada *Pull request* será avaliado quanto ao cumprimento das tarefas e respectivos elementos conforme acordado no item 4.2.4.

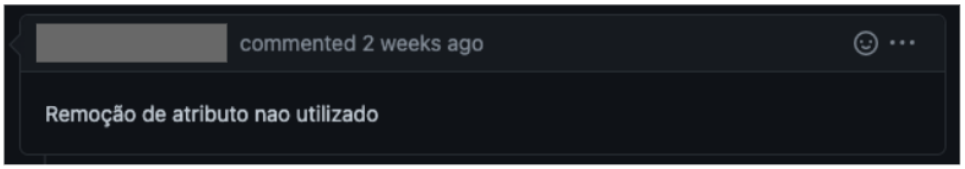
Para cada elemento da tarefa deve-se coletar uma evidência, quando cabível, para a etapa de validação de pontos realizada pelo líder técnico.

4.3.5.1. Acompanhamento da execução das tarefas

Para melhor entendimento da dinâmica e implantação das diretrizes de gamificação, a aplicação da mecânica e suas respectivas evidências foi organizada na Tabela 18.

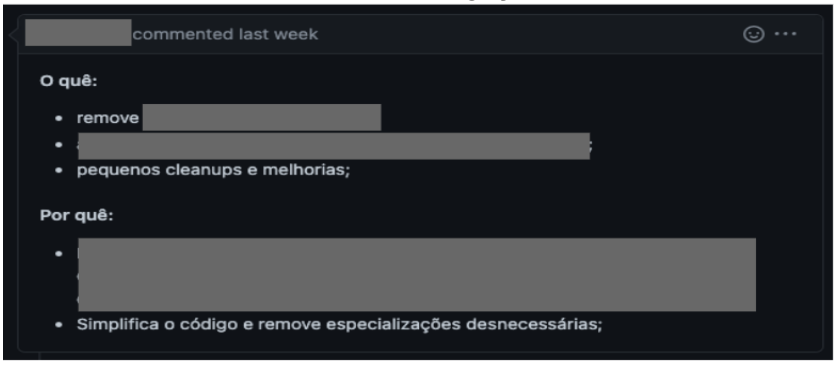
As evidências foram observadas durante a revisão do código através de *Pull requests* realizados no projeto no período do estudo de caso, totalizando 16 *pull requests* abertos durante o estudo de caso.

Tabela 18 - Execução de tarefas e evidências por desenvolvedor

<table>
<tr><td>JOGADOR</td><td colspan="3">Jogador 1 - Desenvolvedor Junior</td></tr>
<tr><td>TAREFA</td><td colspan="3">1 - Identificar locais de código onde o software deve ser refatorado
2 - Determinar as atividades apropriadas de refatoração a serem aplicadas.</td></tr>
<tr><td>EVIDÊNCIAS</td><td colspan="3">[link do repositório do GitHub - divulgação não autorizada]
 
</td></tr>
<tr><td>MOMENTO</td><td colspan="3">Manutenção do software.</td></tr>
<tr><td rowspan="7">MECÂNICA</td><td>Descrição</td><td>Evidência</td><td>Pontos</td></tr>
<tr><td>1.1 - Ter conhecimento dos tipos mais comuns de code smells existentes;</td><td>Code smell: Nome da variável diz a própria expressão</td><td>1</td></tr>
<tr><td>1.2- Avaliar a classe em que será feita a manutenção de forma a identificar code smells;</td><td>Classe identificada</td><td>1</td></tr>
<tr><td>1.3- Criar histórico corrigindo o code smells na ferramenta de versionamento (exemplo: criar commits de refatoração de códigos).</td><td>Commit criado</td><td>1</td></tr>
<tr><td>2.1- Ter conhecimento das principais atividades de refatoração;</td><td>Atividade: remoção de código não utilizado</td><td>1</td></tr>
<tr><td>2.2- Aplicar uma atividade de refatoração equivalente ao tipo de code smell identificado;</td><td>Atividade executada</td><td>1</td></tr>
<tr><td>2.3- Criar histórico corrigindo o code smells na ferramenta de versionamento (exemplo: criar commits de refatoração de códigos).</td><td>Commit criado</td><td>1</td></tr>
<tr><td>ENGAJAMENTO</td><td>1- Deixar o código mais limpo do que quando foi encontrado, equivale a 1 ponto.</td><td>Código não utilizado removido</td><td>3</td></tr>
<tr><td colspan="3">TOTAL DE PONTOS OBTIDOS</td><td>9</td></tr>
</table>

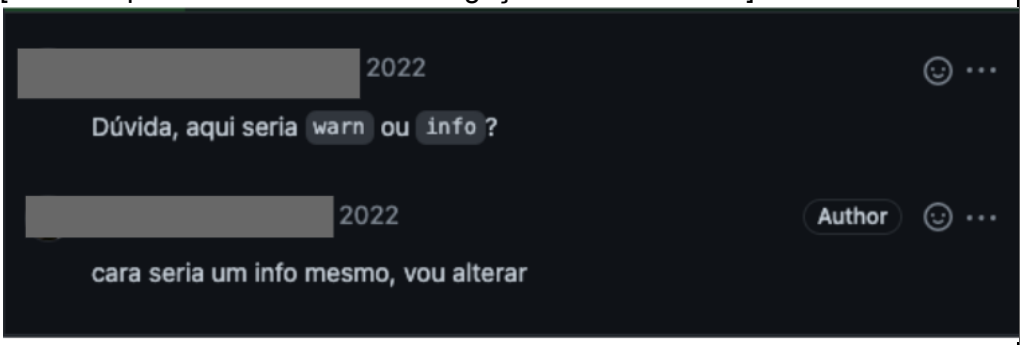
Fonte: Autora.

Tabela 18 - Execução de tarefas e evidências por desenvolvedor - continuação.

<table>
<tr><td>JOGADOR</td><td colspan="3">Jogador 2 - Desenvolvedor Sênior</td></tr>
<tr><td>TAREFA</td><td colspan="3">1 - Identificar locais de código onde o software deve ser refatorado
2 - Determinar as atividades apropriadas de refatoração a serem aplicadas.</td></tr>
<tr><td>EVIDÊNCIAS</td><td colspan="3">[link do repositório do GitHub - divulgação não autorizada]
 
</td></tr>
<tr><td>MOMENTO</td><td colspan="3">Manutenção do software.</td></tr>
<tr><td rowspan="7">MECÂNICA</td><td>Descrição</td><td>Evidência</td><td>Pontos</td></tr>
<tr><td>1.1 - Ter conhecimento dos tipos mais comuns de code smells existentes;</td><td>Classes com códigos duplicados</td><td>1</td></tr>
<tr><td>1.2- Avaliar a classe em que será feita a manutenção de forma a identificar code smells;</td><td>Classes identificadas</td><td>1</td></tr>
<tr><td>1.3- Criar histórico corrigindo o code smells na ferramenta de versionamento (exemplo: criar commits de refatoração de códigos).</td><td>Commit criado</td><td>1</td></tr>
<tr><td>2.1- Ter conhecimento das principais atividades de refatoração;</td><td>Remoção de classes com código duplicado</td><td>1</td></tr>
<tr><td>2.2- Aplicar uma atividade de refatoração equivalente ao tipo de code smell identificado;</td><td>Refatoração aplicada</td><td>1</td></tr>
<tr><td>2.3- Criar histórico corrigindo o code smells na ferramenta de versionamento (exemplo: criar commits de refatoração de códigos).</td><td>Commit criado</td><td>1</td></tr>
<tr><td>ENGAJAMENTO</td><td>1- Deixar o código mais limpo do que quando foi encontrado, equivale a 1 ponto.</td><td>Fez melhorias no código</td><td>3</td></tr>
<tr><td colspan="3">TOTAL DE PONTOS OBTIDOS</td><td>9</td></tr>
</table>

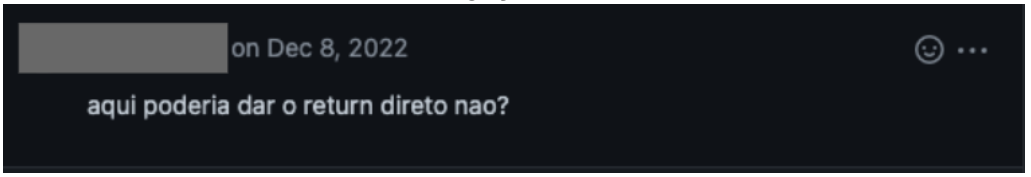
Fonte: Autora.

Tabela 18 - Execução de tarefas e evidências por desenvolvedor - continuação.

<table>
<tr><td>JOGADOR</td><td colspan="3">Jogador 3 - Desenvolvedor Sênior</td></tr>
<tr><td>TAREFA</td><td colspan="3">1 - Identificar locais de código onde o software deve ser refatorado
2 - Determinar as atividades apropriadas de refatoração a serem aplicadas.</td></tr>
<tr><td>EVIDÊNCIAS</td><td colspan="3">[link do repositório do GitHub - divulgação não autorizada]
 
</td></tr>
<tr><td>MOMENTO</td><td colspan="3">Durante a verificação do código de outro desenvolvedor, o desenvolvedor sugere uma refatoração.</td></tr>
<tr><td rowspan="6">MECÂNICA</td><td>Descrição</td><td>Evidência</td><td>Pontos</td></tr>
<tr><td>1.1 - Ter conhecimento dos tipos mais comuns de code smells existentes;</td><td>Nome da variável não clara</td><td>1</td></tr>
<tr><td>1.2- Verificar código de outros desenvolvedores</td><td>Aprovação de código de outro membro da equipe</td><td>1</td></tr>
<tr><td>1.3- Identificar através de comentários, na ferramenta de versionamento, o code smell.</td><td>Comentário criado</td><td>1</td></tr>
<tr><td>2.1- Ter conhecimento das principais atividades de refatoração;</td><td>Atividade sugerida: Renomear variável</td><td>1</td></tr>
<tr><td>2.2- Sugerir uma atividade de refatoração equivalente ao tipo de code smell identificado.</td><td>Sugestão dada</td><td>1</td></tr>
<tr><td>ENGAJAMENTO</td><td>1- Para cada código verificado de outra equipe, soma-se 3 pontos;</td><td>Pull request de outra equipe</td><td>3</td></tr>
<tr><td colspan="3">TOTAL DE PONTOS OBTIDOS</td><td>8</td></tr>
</table>

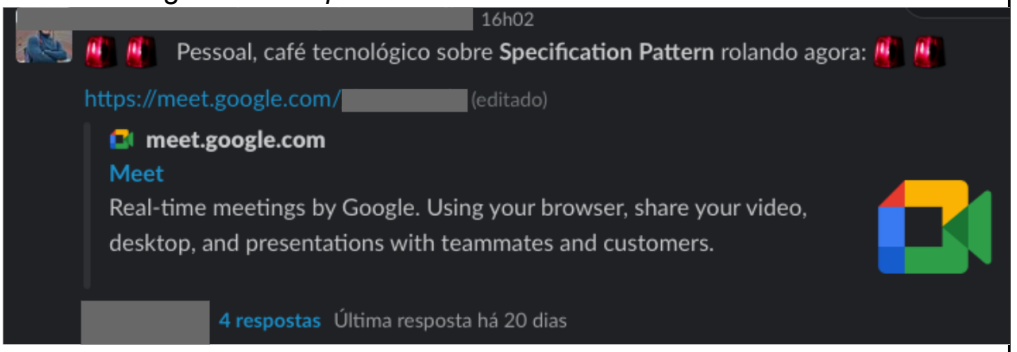
Fonte: Autora.

Tabela 18 - Execução de tarefas e evidências por desenvolvedor - continuação.

<table>
<tr><td>JOGADOR</td><td colspan="3">Jogador 4 - Desenvolvedor Pleno</td></tr>
<tr><td>TAREFA</td><td colspan="3">1 - Identificar locais de código onde o software deve ser refatorado
2 - Determinar as atividades apropriadas de refatoração a serem aplicadas.</td></tr>
<tr><td>EVIDÊNCIAS</td><td colspan="3">[link do repositório do GitHub - divulgação não autorizada]
 
</td></tr>
<tr><td>MOMENTO</td><td colspan="3">Durante a verificação do código de outro desenvolvedor, o desenvolvedor sugere uma refatoração.</td></tr>
<tr><td rowspan="6">MECÂNICA</td><td>Descrição</td><td>Evidência</td><td>Pontos</td></tr>
<tr><td>1.1 - Ter conhecimento dos tipos mais comuns de code smells existentes;</td><td>Variável desnecessária</td><td>1</td></tr>
<tr><td>1.2- Verificar código de outros desenvolvedores</td><td>Aprovação de código de outro membro da equipe</td><td>1</td></tr>
<tr><td>1.3- Identificar através de comentários, na ferramenta de versionamento, o code smell.</td><td>Comentário criado</td><td>1</td></tr>
<tr><td>2.1- Ter conhecimento das principais atividades de refatoração;</td><td>Internalizar variável</td><td>1</td></tr>
<tr><td>2.2- Sugerir uma atividade de refatoração equivalente ao tipo de code smell identificado.</td><td>Sugestão dada</td><td>1</td></tr>
<tr><td rowspan="2">ENGAJAMENTO</td><td>1- Para cada código verificado de outra equipe, soma-se 3 pontos;</td><td>Pull request de outra equipe</td><td>3</td></tr>
<tr><td>1- Para cada atividade sugerida para outra equipe, soma-se 3 pontos;</td><td>Sugestão dada para outra equipe</td><td>3</td></tr>
<tr><td colspan="3">TOTAL DE PONTOS OBTIDOS</td><td>11</td></tr>
</table>

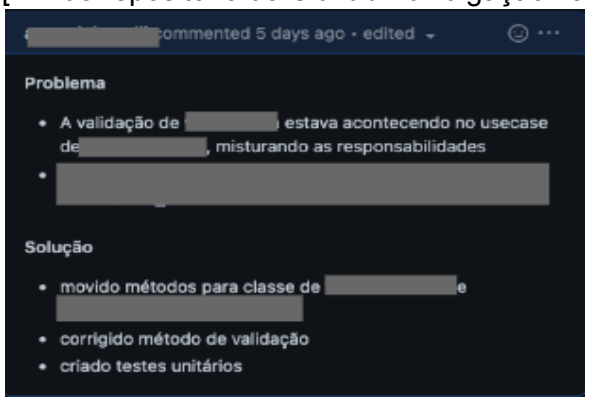
Fonte: Autora.

Tabela 18 - Execução de tarefas e evidências por desenvolvedor - continuação.

<table>
<tr><td>JOGADOR</td><td colspan="3">Jogador 3 - Desenvolvedor Sênior</td></tr>
<tr><td>TAREFA</td><td colspan="3">2 - Determinar as atividades apropriadas de refatoração a serem aplicadas.</td></tr>
<tr><td>EVIDÊNCIAS</td><td colspan="3">Café tecnológico sobre Specification Pattern
 
</td></tr>
<tr><td>MOMENTO</td><td colspan="3">Durante a verificação do código de outro desenvolvedor, o desenvolvedor sugere uma refatoração.</td></tr>
<tr><td rowspan="3">MECÂNICA</td><td>Descrição</td><td>Evidência</td><td>Pontos</td></tr>
<tr><td>2.1- Ter conhecimento das principais atividades de refatoração;</td><td>Não aplicável</td><td>0</td></tr>
<tr><td>2.2- Sugerir uma atividade de refatoração equivalente ao tipo de code smell identificado.</td><td>Não aplicável</td><td>0</td></tr>
<tr><td>ENGAJAMENTO</td><td>Para cada sugestão de aplicação de um design pattern soma-se 5 pontos;</td><td>Apresentação para todas as equipes</td><td>5</td></tr>
<tr><td colspan="3">TOTAL DE PONTOS OBTIDOS</td><td>5</td></tr>
</table>

Fonte: Autora.

Tabela 18 - Execução de tarefas e evidências por desenvolvedor - continuação.

<table>
<tr><td>JOGADOR</td><td colspan="3">Jogador 5 - Desenvolvedor Junior</td></tr>
<tr><td>TAREFA</td><td colspan="3">1 - Identificar locais de código onde o software deve ser refatorado
2 - Determinar as atividades apropriadas de refatoração a serem aplicadas.</td></tr>
<tr><td>EVIDÊNCIAS</td><td colspan="3">[link do repositório do GitHub - divulgação não autorizada]
 
</td></tr>
<tr><td>MOMENTO</td><td colspan="3">Manutenção do software.</td></tr>
<tr><td rowspan="7">MECÂNICA</td><td>Descrição</td><td>Evidência</td><td>Pontos</td></tr>
<tr><td>1.1 - Ter conhecimento dos tipos mais comuns de code smells existentes;</td><td>Code smell: Classe grande</td><td>1</td></tr>
<tr><td>1.2- Avaliar a classe em que será feita a manutenção de forma a identificar code smells;</td><td>Classe identificada</td><td>1</td></tr>
<tr><td>1.3- Criar histórico corrigindo o code smells na ferramenta de versionamento (exemplo: criar commits de refatoração de códigos).</td><td>Commit criado</td><td>1</td></tr>
<tr><td>2.1- Ter conhecimento das principais atividades de refatoração;</td><td>Remoção de método para classe</td><td>1</td></tr>
<tr><td>2.2- Aplicar uma atividade de refatoração equivalente ao tipo de code smell identificado;</td><td>Atividade executada</td><td>1</td></tr>
<tr><td>2.3- Criar histórico corrigindo o code smells na ferramenta de versionamento (exemplo: criar commits de refatoração de códigos).</td><td>Commit criado</td><td>1</td></tr>
<tr><td>ENGAJAMENTO</td><td>1- Deixar o código mais limpo do que quando foi encontrado, equivale a 1 ponto.</td><td>Código não utilizado removido</td><td>3</td></tr>
<tr><td colspan="3">TOTAL DE PONTOS OBTIDOS</td><td>9</td></tr>
</table>

Fonte: Autora.

4.3.5.2. Placar de pontos

Conforme o acompanhamento da execução das tarefas e respectivos elementos, o líder técnico avaliou cada ponto dado e os pontos avaliados foram organizados conforme a Tabela 19.

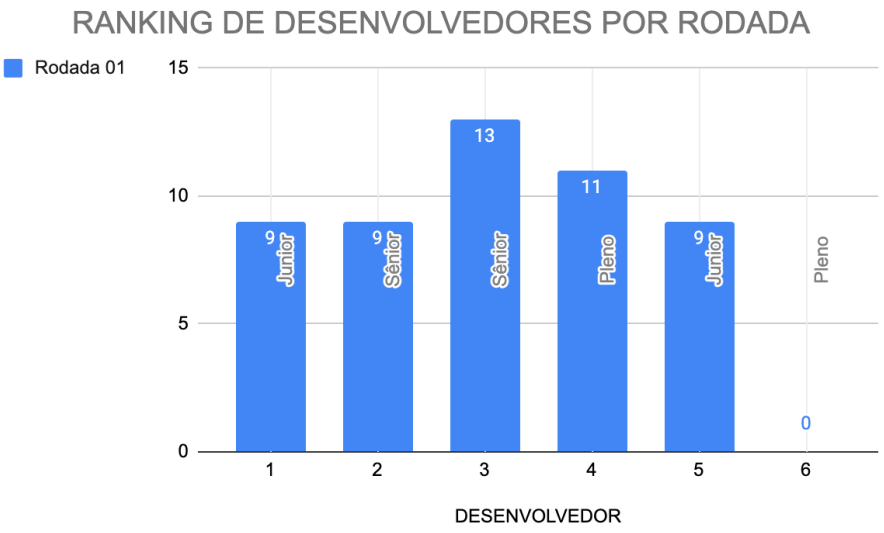
Tabela 19 - Pontos por desenvolvedor.

DESENVOLVEDOR	CARGO	PONTUAÇÃO	
		Rodada 01	TOTAL
1	Junior	9	9
2	Sênior	9	9
3	Sênior	13	13
4	Pleno	11	11
5	Junior	9	0
6	Pleno	0	0

Fonte: Autora

A Figura 18 representa a evolução da pontuação de cada desenvolvedor conforme a respectiva rodada.

Figura 18 - Evolução de pontos por desenvolvedor e rodada.

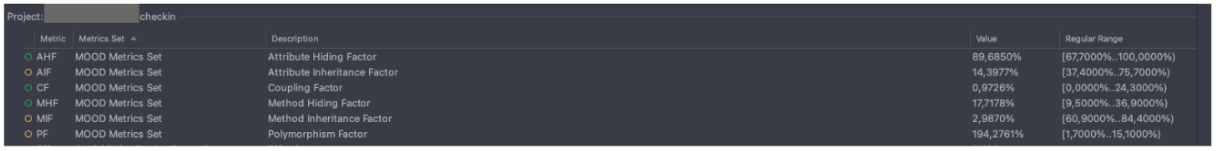


Fonte: Autora

4.3.5.3. Verificação de métricas de qualidade de *software*

Após realizada a rodada de teste na qual aplicou-se a gamificação, foi utilizado o plugin *MetricsTree* do *Intellij* para avaliar a métrica MOOD, a Figura 19 apresenta a avaliação do projeto conforme as métricas de qualidade MOOD.

Figura 19 - Avaliação de métricas MOOD, após gamificação.

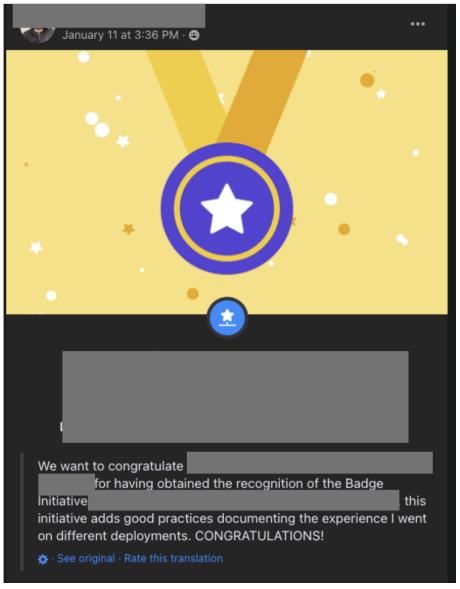


Fonte: Autora.

4.3.5.4. Incentivos e recompensas

Conforme evolução de pontos apresentada na Figura 18, o desenvolvedor vencedor foi o desenvolvedor 3, no qual recebeu um selo que foi divulgado no grupo da equipe na ferramenta *Workplace*, conforme Figura 20, bem como um *voucher* para ser utilizado em um aplicativo de *delivery* de comida.

Figura 20 - Divulgação de selo de ganhador.



Fonte: Autora

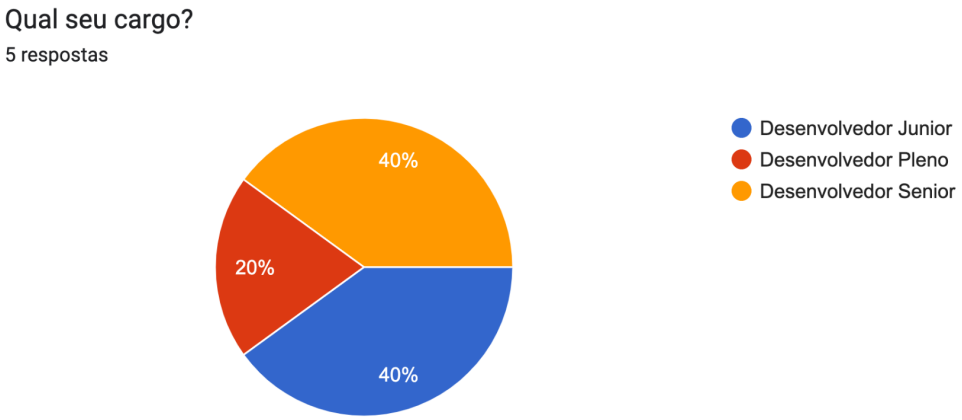
Como a equipe se dedicou ao cumprimento das atividades, o líder de projeto determinou um dia de folga para todos os membros da equipe, independente da pontuação, salvo aqueles que estavam de licença.

4.3.5.5. Avaliação do processo gamificado

Para melhor comparação dos resultados, foi criado um formulário conforme a Tabela 14 do item 3.2.5. O formulário foi criado na ferramenta Google Forms (https://forms.gle/WvpwPhFJHekwMGPC8) e respondido pelos desenvolvedores de forma anônima após a finalização da rodada de teste da gamificação. Foi necessário acrescentar a pergunta “Qual seu cargo”, não prevista na Tabela 14, de forma a gerar dados comparativos referente a gamificação.

A Figura 21 representa os níveis dos desenvolvedores que participaram da gamificação.

Figura 21 - Níveis dos desenvolvedores que participaram da gamificação.



Fonte: Autora

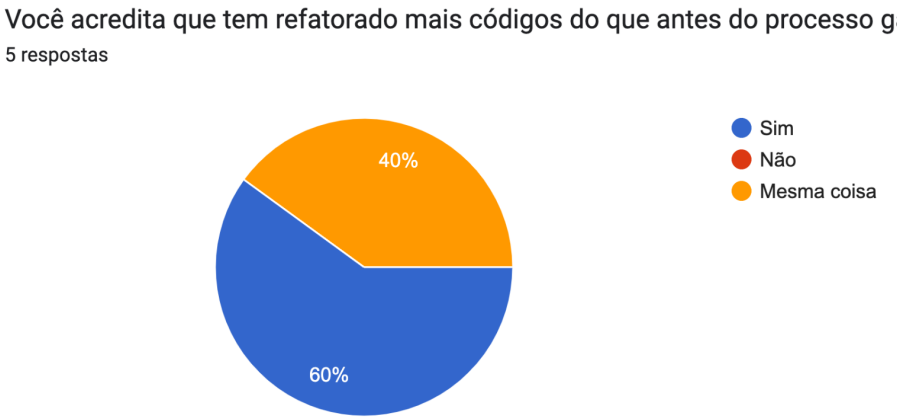
A Figura 22 apresenta a opinião geral dos desenvolvedores quanto a gamificação.

Figura 22 - Opinião dos desenvolvedores quanto a gamificação.

Fonte: Autora

A Figura 23 apresenta a opinião dos desenvolvedores quanto ao aumento de refatoração de códigos antes da gamificação.

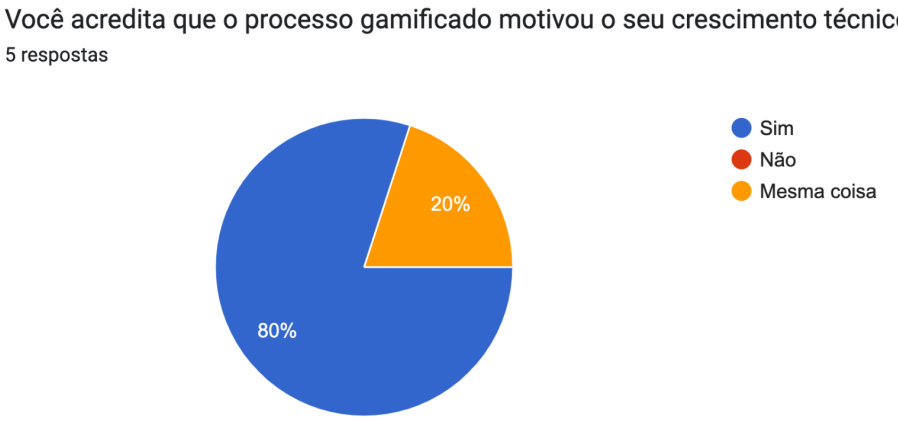
Figura 23 - Opinião referente ao aumento de refatoração de código.



Fonte: Autora

A Figura 24 indica que a gamificação motivou o crescimento técnico dos desenvolvedores.

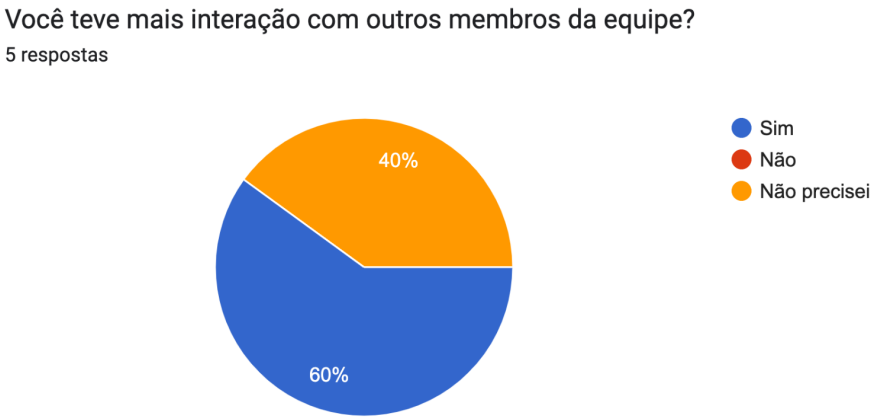
Figura 24 - Motivação de crescimento técnico com a gamificação.



Fonte: Autora.

A Figura 25 apresenta a interação dos membros da equipe durante a gamificação.

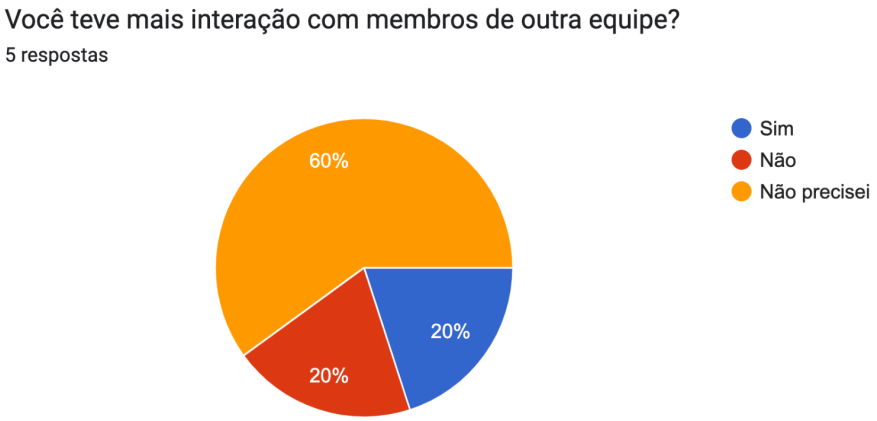
Figura 25 - Interação com membros da equipe.



Fonte: Autora

A Figura 26 apresenta a interação dos membros da equipe com outros membros de outras equipes, durante a gamificação.

Figura 26 - Interação com outra equipe durante a gamificação.



Fonte: Autora.

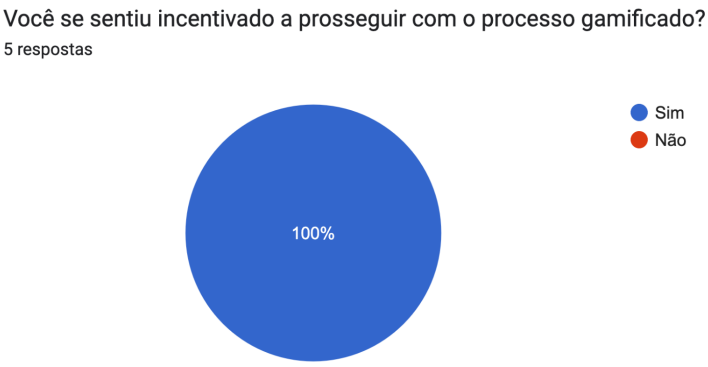
A Figura 27 apresenta a opinião dos desenvolvedores quanto ao sistema de recompensas.

Figura 27 - Opinião dos desenvolvedores referente às recompensas.

Fonte: Autora.

A Figura 28 representa o sentimento de prosseguir com a gamificação após a rodada de teste.

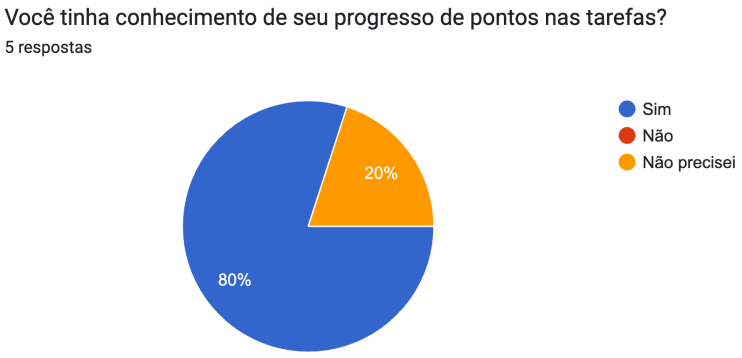
Figura 28 - Sentimento de incentivo quanto a prosseguir com a gamificação.



Fonte: Autora.

A Figura 29 apresenta o quanto os desenvolvedores estavam acompanhando o progresso de pontos durante a gamificação.

Figura 29 - Conhecimento do progresso de pontos durante a gamificação.



Fonte: Autora.

Na pergunta “Você tem alguma sugestão de melhoria?”, houver duas respostas, sendo elas:

- Fazer mais cafés tecnológicos sobre *design patterns*
- Fazer um café tecnológico sobre *code smells*, muitos podem não saber quais são

Não houve respostas para a pergunta “Teve algo que você não gostou”.

5. ANÁLISE DE RESULTADOS

Para avaliar a aplicabilidade das diretrizes de gamificação durante a etapa de codificação a fim de auxiliar a qualidade da refatoração do código, incentivar e motivar a equipe a executar a refatoração, um estudo de caso foi realizado durante o período de 45 dias em uma empresa do ramo de logística que desenvolve seu próprio *software* de gerenciamento logístico.

Durante o estudo de caso, apresentado no Capítulo 4, foram levantados dois objetivos, sendo eles:

- Objetivo técnico: Melhorar uma ou mais métricas de qualidade da aplicação principal, responsável pelo *software* que faz a entrada dos produtos no armazém, denominada neste trabalho como *check-in*.
- Objetivo de engajamento da equipe: Melhorar a performance técnica com base na criação de novos hábitos de refatoração e no aprendizado de novos *designs patterns*.

Os resultados foram obtidos de duas formas, conforme cada objetivo:

- Objetivo técnico:
    - Através da análise das métricas de qualidade do *software* antes e depois da aplicação das diretrizes de gamificação;
- Objetivo de engajamento da equipe:
    - Histórico de *pull requests* realizados pela equipe no período de 45 dias antes da implantação das diretrizes para gamificação a fim de avaliar a quantidade de atividades de refatoração realizadas pelos desenvolvedores;
    - Pesquisa com os desenvolvedores que participaram da rodada de testes da aplicação da gamificação.

Nos tópicos seguintes, é apresentada a análise dos dados coletados para averiguar se os objetivos foram atingidos.

5.1. Resultados conforme objetivo técnico

Durante a verificação das métricas das diretrizes de gamificação, descritas no item 4.3.1, as métricas do projeto foco do estudo de caso foram levantadas antes da implantação das diretrizes. A Tabela 20 apresenta os resultados.

Tabela 20 - Comparação de métricas de qualidade de software antes e depois da aplicação das diretrizes de gamificação.

MÉTRICA	FAIXA DE REFERÊNCIA	VALOR ANTERIOR	VALOR APÓS GAMIFICAÇÃO	RESULTADO
AHF - Fator de encapsulamento de atributos	67,70% - 100,00%	89,5504%	89,6850%	+0,1346%
AIF - Fator de herança de atributo	37,40% - 75,70%	13,7957%	14,3977%	+0,6020%
CF - Fator de acoplamento	0,00% - 24,30%	1,0159%	0,9726%	-0,0433%
MHF - Fator de encapsulamento de método	9,50% - 36,90%	17,4503%	17,7178%	+0,2675%
MIF - Fator de herança de método	60,90% - 84,40%	2,5855%	2,9870%	+0,4015%
PF - Fator de polimorfismo	1,70% - 15,10%	202,5271%	194,2761%	-8,2510%

Fonte: Autora.

Conforme apresentado na tabela, todas as métricas foram afetadas pela refatoração realizada durante o estudo de caso.

A métrica Fator de Encapsulamento de Atributos (AHF) é uma medida do uso do conceito de encapsulamento de atributos. O encapsulamento permite lidar com a complexidade das classes sendo que os atributos ficam ocultados, sendo possível seu acesso somente por métodos da classe correspondente. Em geral, conforme a métrica aumenta, a complexidade do programa diminui conforme descrição do *software MetricsTree* apresentado na Figura 30.

Após a refatoração, o valor teve um aumento de 0,1346%, podendo ser considerado um impacto positivo na métrica de qualidade do projeto devido ao valor que permaneceu dentro da faixa de referência.

Figura 30 - Descrição da métrica de Fator de Encapsulamento de Atributos (AHF)

Fonte: *MetricsTree* (2022).

A métrica de Fator de Herança de Atributo (AIF) é o quociente entre a soma dos atributos herdados em todas as classes e o número total de atributos disponíveis em todas as classes. De acordo com o valor do fator de herança de atributo, ele pode afetar a compreensibilidade e a testabilidade, conforme apresentado na Figura 31.

Após a aplicação da refatoração, houve um aumento de 0,6020% no fator de herança de atributo, podendo ser considerado como um impacto positivo pois o valor se aproximou da faixa do valor de referência.

Figura 31 - Descrição da métrica de Fator de Herança de Atributo (AIF)

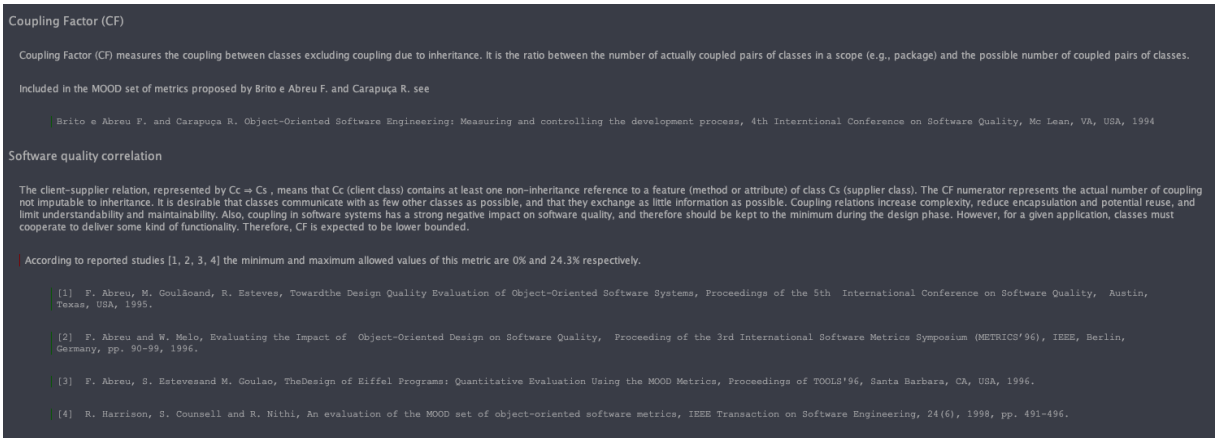
Fonte: *MetricsTree (*2022*).*

A métrica Fator de acoplamento (CF) é a relação entre o número de classes em um escopo (por exemplo, pacote) e o possível número de classes acopladas. Busca-se

que as classes se comuniquem com o menor número de outras classes, e que troquem informações o mínimo possível. As relações de acoplamento aumentam a complexidade, reduzem o encapsulamento e a reutilização e limitam a compreensibilidade, a Figura 32 apresenta a descrição completa conforme o *plugin MetricsTree*.

Após a aplicação da refatoração, houve uma diminuição de 0,0433%, mantendo a métrica dentro da faixa de referência aceitável, podendo ser considerado um impacto positivo.

Figura 32 - Descrição da métrica de Fator de acoplamento (CF)

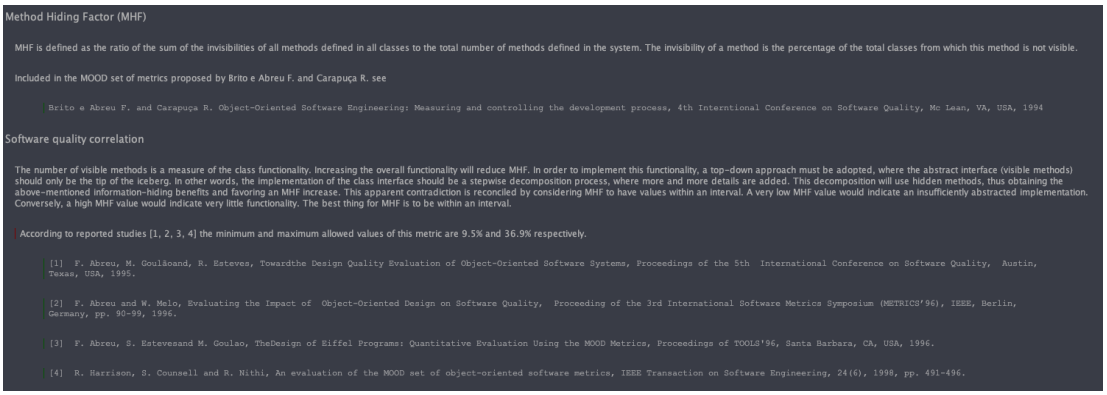


Fonte: *MetricsTree (*2022*).*

A métrica Fator de Encapsulamento de Método (MHF) é a relação entre a soma do encapsulamento dos métodos definidos em todas as classes e o número total de métodos definidos no *software*, sendo a porcentagem total das classes que os métodos não são visíveis. Conforme a descrição apresentada na Figura 33, é indicado que esse valor esteja dentro da faixa de referência.

Após a aplicação da refatoração houve um aumento de 0,2675% no fator, não alterando a posição do valor na faixa de referência, podendo ser considerado como um impacto neutro.

Figura 33 - Descrição da métrica de Fator de Encapsulamento de Método (MHF)



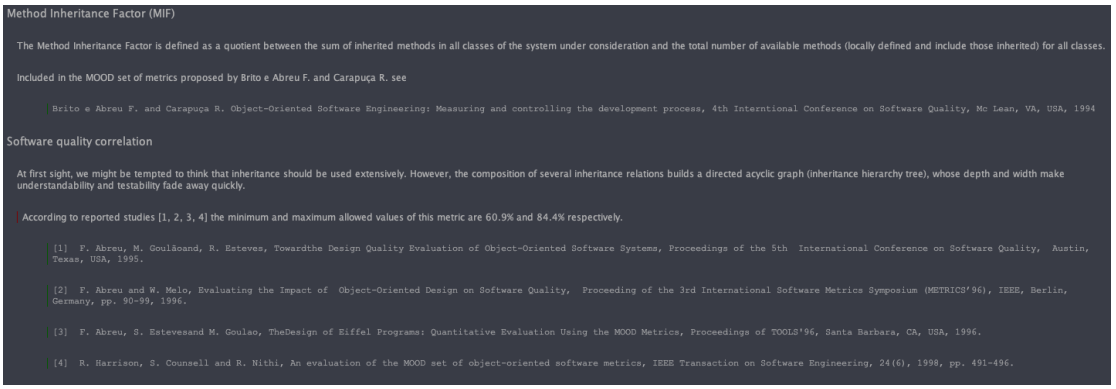
Fonte: *MetricsTree (*2022*).*

A métrica Fator de Herança de Método (MIF) é o quociente entre a soma dos métodos herdados em todas as classes e o número total de métodos disponíveis.

O Fator de Herança do Método é definido como um quociente entre a soma dos métodos herdados em todas as classes do sistema em consideração e o número total de métodos disponíveis (definidos localmente e incluem os herdados) para todas as classes. Em geral, conforme a métrica aumenta, a complexidade do programa diminui conforme descrição do *software MetricsTree* apresentado na Figura 34.

Após a refatoração houve um aumento de 0,4015% podendo ser considerado como um impacto positivo pois o valor se aproximou da faixa do valor de referência.

Figura 34 - Descrição da métrica de Fator de Herança de Método (MIF)

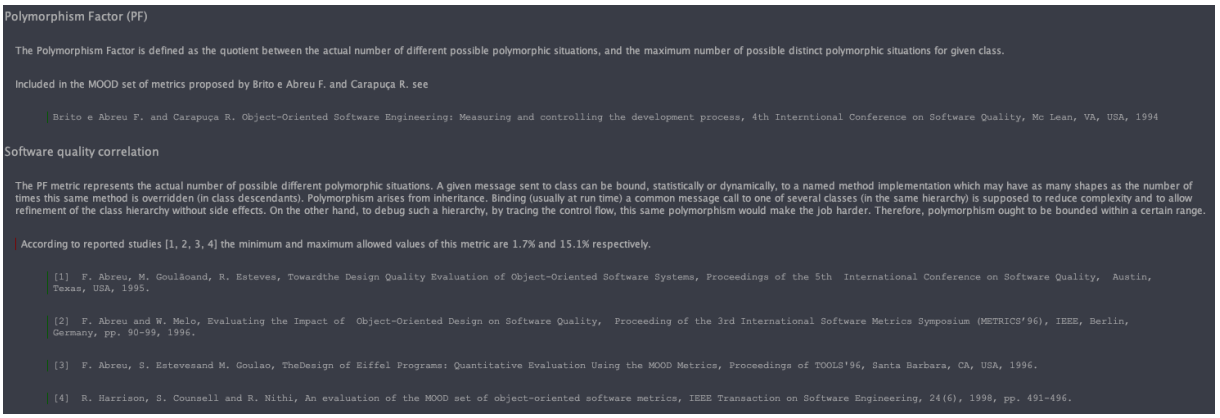


Fonte: *MetricsTree (*2022*).*

A métrica Fator de Polimorfismo (PF) é o quociente entre o número real de diferentes situações polimórficas possíveis, e o número máximo de possíveis situações polimórficas distintas para uma determinada classe. Em geral o polimorfismo pode reduzir a complexidade e refinar a hierarquia, porém, para depurar a hierarquia e rastrear o fluxo de controle da aplicação, é um trabalho mais difícil, desta forma é indicado que o valor do fator de polimorfismo deve ser delimitado dentro da faixa de referência. A Figura 35 apresenta a descrição completa da métrica.

Após a refatoração, houve uma redução de 8,2510% no fator, aproximando o valor à faixa de referência, sendo considerado um impacto positivo.

Figura 35 - Descrição da métrica de Fator de polimorfismo (PF)



Fonte: *MetricsTree (*2022*).*

5.1.1. Análises dos resultados obtidos quanto às métricas

Os resultados foram obtidos conforme a verificação das métricas antes e depois da implantação do processo de gamificação, no período de 45 dias. Todas as métricas tiveram um impacto sendo positivo ou negativo. Porém, vale ressaltar que os resultados obtidos podem não ser vinculados especificamente à aplicação das diretrizes, mas podem ser efeitos de outras adições e manutenções realizadas no *software* em questão no decorrer do período.

Conforme a Tabela 20, o maior resultado obtido foi na métrica Fator de Polimorfismo, na qual teve uma redução de 8,5210%, aproximando o valor à faixa de referência, embora este ainda não esteja contemplado na faixa em si (1,70% - 15,10%).

Observa-se que na Tabela 18, o Jogador 5 realizou uma refatoração no código indicado na descrição do *pull request,* conforme apresentado abaixo:

Problema:

- A validação de _____ estava acontecendo no *usecase* de _____, misturando as responsabilidades.
- (não divulgado)

Solução:

- movido métodos para classe de **_____**
- corrigido métodos de validação
- criado testes unitários

Com base na descrição do *pull request,* é possível inferir que havia uma validação que estava em uma classe, porém essa validação não era de responsabilidade da classe na qual estava inserida, podendo ser caracterizada como *code smell* “Classe grande”. O Jogador 5 determinou como solução remover o método da classe para uma outra classe, realizando a atividade de refatoração “Mover método”, como também “Extrair classe” que contempla remover uma responsabilidade de uma classe para outra. Como consequência dessa alteração, o método passou a ser visível para outras classes possibilitando o polimorfismo deste.

O fator de polimorfismo é vinculado diretamente com os métodos existentes nas classes, conforme descrito na Tabela 5 apresentada no item 2.1.1.5. Isto posto, pode-se concluir que a atividade de refatoração realizada pelo Jogador 5 impactou positivamente no Fator de polimorfismo do *software*.

5.2. Resultados conforme objetivo de engajamento da equipe

Para analisar o impacto das diretrizes de gamificação no objetivo de engajamento da equipe, foram analisados os *pull requests* referentes à 45 dias antes do início da implantação das diretrizes e a pesquisa com os desenvolvedores que participaram da rodada.

5.2.1. Quantidade de refatorações realizadas antes das diretrizes da gamificação

A análise do histórico foi feita no repositório do projeto. Dentre o período de 45 dias antes da implantação das diretrizes de gamificação foram realizados pela equipe de desenvolvimento 24 *pull requests.*

Dentre os 24 *pull requests*, não foram encontrados *branches* ou *commits* que possuíssem palavras chaves que remetem a refatoração, tais como: refatoração, *refactoring*, *refactor* e *enhancement*. Durante a análise de cada *pull request*, apenas três destes possuíam atividades de refatoração em conjunto com a adição de novas funcionalidades.

A Tabela 21 apresenta a quantidade de *pull requests (PRs)* realizados antes e depois da implantação das diretrizes de gamificação.

Tabela 21 - Quantidade de pull requests realizados e com refatoração

PERÍODO	TOTAL DE *PRs*	*PRs* COM REFATORAÇÃO	PORCENTAGEM
Antes da implantação das diretrizes	24	3	12,5%
Depois da implantação das diretrizes	16	5	31,5%

Fonte: Autora

Embora a quantidade de *pull requests* antes da implantação das diretrizes tenha sido maior que durante o período de implantação das diretrizes de gamificação, observa-se que houve uma porcentagem maior de *pull requests* que aplicavam alguma atividade de refatoração.

5.2.2. Análise das respostas da pesquisa realizadas com os desenvolvedores participantes

Após o período de implantação das diretrizes de gamificação, foi feita uma pesquisa referente a gamificação. No total, foram cinco desenvolvedores participantes da implantação das diretrizes e da pesquisa.

De maneira geral, os respondentes indicaram que o processo gamificado foi bom, representando 60% das respostas; 20% indicaram que o processo gamificado foi muito bom e 20% indicaram que o processo gamificado foi razoável.

Para avaliar se as diretrizes de gamificação impactou a quantidade de refatoração de cada desenvolvedor, 60% das respostas obtidas indicam que sim, houve um aumento na quantidade de códigos refatorado pelo desenvolver, em contrapartida, 40% dos respondentes indicou que não houve mudança na quantidade de códigos refatorados. Após o período da implantação da gamificação, houve um aumento na refatoração de código.

Com intuito de avaliar o objetivo de crescimento e performance técnica, 80% dos respondentes indicaram que as diretrizes de gamificação motivaram o crescimento técnico e 20% indicaram que a motivação permaneceu a mesma. Isto posto, as

diretrizes de gamificação motivaram positivamente o crescimento técnico da maioria dos desenvolvedores.

De maneira a verificar a interação dos desenvolvedores com os membros da equipe, 60% declararam que sim, houve mais interação com outros membros e 40% indicaram que não houve interação com outros membros. Desta forma, as diretrizes de gamificação geraram um impacto positivo na maior parte da equipe quanto à interação entre os membros da equipe.

De forma a avaliar a interação com diferentes equipes, 60% dos desenvolvedores não tiveram interação com outras equipes pois não houve necessidade; 20% alegaram que tiveram mais interação e 20% não tiveram mais interação. Desta forma, não é possível analisar o aumento da motivação para interação com outras equipes, pois não houve a necessidade de interação para a maioria dos desenvolvedores.

Referente às recompensas oferecidas para os jogadores e para o ganhador, a maioria dos jogadores gostaram das recompensas oferecidas, com isso é possível determinar que as recompensas são atrativas para os desenvolvedores.

De forma a validar a continuidade da implantação das diretrizes de gamificação, todos os desenvolvedores sentiram-se incentivados a prosseguir com a gamificação, desta maneira, as diretrizes de gamificação tiveram uma boa aceitação pela equipe.

Quanto à divulgação do progresso de pontos nas tarefas, 80% dos respondentes tinham conhecimento de seus pontos e 20% informaram que não houve a necessidade de conhecer os seus pontos. Desta forma a divulgação dos pontos foi feita de maneira efetiva.

Durante a pesquisa, houve duas sugestões de melhorias ao que tange o conhecimento referente a *design patterns* e *code smells* não referente às diretrizes de gamificação.

6. CONSIDERAÇÕES FINAIS

Os estudos aqui apresentados buscaram apresentar uma forma de gamificar a etapa de codificação de *software* de maneira a aumentar o desejo dos desenvolvedores quanto à refatoração do código, de modo que houvesse ganhos no ponto de vista de negócio, como também ganhos atrativos para os desenvolvedores.

6.1. Conclusão

O objetivo deste trabalho, de acordo com a seção 1.2, é elaborar um conjunto de diretrizes que podem ser utilizadas durante a etapa de codificação de *software*, de forma a auxiliar a qualidade da refatoração do código, incentivar e motivar a equipe a executar a refatoração, utilizando técnicas de gamificação para engajar a equipe durante as iterações do projeto.

Embora não seja uma ferramenta de gamificação, o trabalho propõe, através das diretrizes apresentadas, uma maneira de aplicar elementos de gamificação no processo de desenvolvimento da equipe de forma a incentivar, motivar e engajar a equipe a executar a refatoração de código a fim de obter pontos e recompensas.

As diretrizes de gamificação foram feitas de forma a serem adaptáveis conforme a realidade da empresa. No Capítulo 4, um estudo de caso é apresentado. Foi observado que as diretrizes tiveram que ser readequadas para melhor aplicação e apenas 2 de 6 tarefas foram aplicadas devido ao tempo disponível para analisar o estudo e característica de trabalho da equipe, dado que trabalham de forma incremental.

Durante o estudo de caso, o projeto escolhido apresentou pequenas melhoras quanto às métricas de qualidade, porém não foi possível relacionar essa melhora diretamente à implantação das diretrizes. Entretanto, houve um impacto maior na métrica Fator de Polimorfismo em comparação com as demais métricas. Esse impacto foi relacionado com o tipo de refatoração realizado pelo jogador e indica que o valor obtido pode estar relacionado com a aplicação das diretrizes.

Conforme a pesquisa feita e a observação do histórico dos códigos escritos no período antes da implantação das diretrizes de gamificação, houve um aumento

significativo da quantidade de código refatorado. A equipe demonstrou interesse e engajamento durante a aplicação das diretrizes de gamificação para realizar as tarefas propostas de forma a conquistar as recompensas propostas.

Isto posto, as diretrizes de gamificação propostas demonstraram ser atrativas para incentivar, motivar e engajar os desenvolvedores quanto à prática da refatoração de código. Porém, é necessário analisar e acompanhar as métricas de qualidade de *software* em um período maior para validar a relação entre a melhora das métricas de qualidade de *software* com a aplicação das diretrizes de gamificação .

6.2. Contribuições do trabalho

Durante a aplicação das diretrizes de gamificação, foi possível identificar os objetivos técnicos e os objetivos dos desenvolvedores da equipe, de forma que muitos desenvolvedores possuem como objetivo o crescimento técnico e a necessidade de tornar a refatoração um hábito.

As diretrizes de gamificação propõem uma análise da cultura da empresa, do processo atual de desenvolvimento e refatoração e sugerem a aplicação de elementos de jogos para auxiliar os desenvolvedores a atingir seus objetivos, bem como os objetivos da empresa e da liderança.

Ao utilizar elementos de jogos durante o processo de desenvolvimento, cria um ambiente divertido, competitivo e incentiva os jogadores a buscar conhecimento para executar as tarefas propostas e conquistar as recompensas oferecidas, consequentemente, impactar a qualidade do código, aumentar a interação com a equipe e demais equipes e incentivar o compartilhamento de conhecimento.

Vale ressaltar que ao entender os objetivos de cada jogador e analisar a pontuação deste no decorrer da aplicação das diretrizes de gamificação, é possível observar oportunidades de melhorias técnicas que podem auxiliar no crescimento profissional do jogador.

Portanto, as diretrizes de gamificação demonstraram ser uma forma de auxiliar na qualidade de refatoração de código, no incentivo e motivação quanto a executar

refatoração de código, no engajamento com a equipe e no desenvolvimento profissional e técnico da equipe.

6.3. Trabalhos futuros

Quanto ao objetivo de auxiliar a qualidade da refatoração do código, as diretrizes de gamificação precisam ser aplicadas em diferentes estudos de caso, nos quais podem ser executados em períodos maiores, em equipes distintas e em situações diferentes para validar se realmente auxilia diretamente na qualidade de refatoração do código, pois no cenário proposto não foi possível avaliar a relação direta entre a aplicação das diretrizes e a melhora das métricas de *software*.

Durante o estudo de caso apresentado no Capítulo 4, apenas dois estágios da refatoração foram utilizados para a aplicação das diretrizes de gamificação. É necessário realizar rodadas adicionais que incluam os demais estágios da refatoração de modo a ser possível analisar o impacto das diretrizes de gamificação em todas as etapas da refatoração.

Ainda é possível evoluir as diretrizes de gamificação utilizando ferramentas para análise de código de maneira automatizada, pois durante o período do estudo de caso foi necessário alocar uma pessoa responsável por analisar os códigos e distribuir os pontos dentre a equipe.

Durante a aplicação das diretrizes foram identificadas oportunidades de melhorias que contemplam a criação de uma ferramenta que auxilie na implantação das diretrizes, de maneira a otimizar o tempo da equipe.

Os questionários aplicados no estudo de caso podem ser refinados, conforme a empresa e as atividades exercidas, pois algumas questões podem não ser relevantes para o processo de refatoração.

Com a aplicação do estudo de caso apresentado no Capítulo 4, é possível concluir que as diretrizes de gamificação podem ser úteis em empresas de diversos tamanhos e tipos. Porém, é importante salientar a possibilidade da aplicação das diretrizes de gamificação em empresas que prestam serviço de desenvolvimento de *software*.

Um exemplo de cenário para um possível trabalho futuro seria a aplicação das diretrizes de gamificação em empresas que prestam serviços de fábrica de *software,* com inúmeros projetos paralelos de diferentes linguagens e complexidade.

Explorar novos cenários através de estudos de casos permitirá a verificação da aplicabilidade das diretrizes de gamificação, bem como relacionar a melhora das métricas de qualidade de *software* com a aplicação das diretrizes de gamificação.

Em suma, a realização de trabalhos futuros baseados neste trabalho permitirá aperfeiçoar as diretrizes apresentadas, a melhoria da aplicação em diferentes cenários e o refinamento de pontos, tarefas e incentivos conforme o amadurecimento das diretrizes de gamificação.

REFERÊNCIAS

ABREU, F. *et al.*, **Toward the Design Quality Evaluation of Object-Oriented Software Systems**, Proceedings of the 5th International Conference on *Software* Quality, 12 p. Austin, Texas, USA, 1995.

ALIZADEH, V., *et al.* **An Interactive and Dynamic Search-Based Approach to Software Refactoring Recommendations**, in IEEE Transactions on *Software* Engineering, vol. 46, no. 9, pp. 932-961, 1 Sept. 2020, doi: 10.1109/TSE.2018.2872711.

ALOMAR, E. A. *et al.* **On preserving the behavior in software refactoring**: **A systematic mapping study**, Information and Software Technology, Volume 140, 2021, 106675, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2021.106675.

ALSHAYEB, M. , **Empirical investigation of refactoring effect on software quality**, Information and Software Technology,Volume 51, Issue 9, 2009, Pages 1319-1326, ISSN 0950-5849,https://doi.org/10.1016/j.infsof.2009.04.002.

ANCKËN, R. V.. **Dispersão que gera resultados: os novos padrões de trabalho:** Empresas distribuídas rompem paradigmas para viver a transformação digital e criar novos modelos de negócio. [S. l.], 15 dez. 2021. Disponível em: https://mercadoeconsumo.com.br/2021/12/15/dispersao-que-gera-resultados-os-novos-padroes-de-trabalho/. Acesso em: 6 jul. 2022.

BUNCHBALL Inc.: **Gamification 101: an introduction to the use of game dynamics to influence behavior.** White Paper. Bunchball Inc., 2010. https://www.bunchball.com/gamification101

BURKE, B. **Gamificar: como a gamificação motiva as pessoas a fazerem coisas extraordinárias**. São Paulo: DVS Editora. ISBN-10 8582891075, ISBN-13 978-8582891070, 2015

BUSARELLO, R. I. **Gamification: princípios e estratégias**. São Paulo: Pimenta Cultural, 126 p, 2016.

COUNSELL, S. *et al.* **Exploring the Eradication of Code Smells: An Empirical and Theoretical Perspective**. Advances in Software Engineering. 2010. 10.1155/2010/820103.

CHAN, E., *et al.* **Effect of Gamification on Intrinsic Motivation. In: Nah, FH., Xiao, B. (eds) HCI in Business, Government, and Organizations**. HCIBGO 2018. Lecture Notes in Computer Science(), vol 10923. Springer, Cham. https://doi.org/10.1007/978-3-319-91716-0_35.

CHIELE, C. **Estudo sobre práticas ágeis de refatoração e testes automatizados no desenvolvimento de software para melhoria da qualidade de sistemas legados**. 2017. Trabalho de conclusão de curso (Pós Graduação Latu Senso em Qualidade de Software) - Universidade do Vale do Rio dos Sinos - UNISINOS, São Leopoldo, 39 p, 2017.

COUTO, C. *et al.*, **A Quality-oriented Approach to Recommend Move Method Refactorings**. 2018. In Proceedings of the 17th Brazilian Symposium on Software Quality (SBQS). Association for Computing Machinery, New York, NY, USA, 11–20. https://doi.org/10.1145/3275245.3275247

DORLING, A.; MCCAFFERY, Fergal. T**he Gamification of SPICE. Software Process Improvement and Capability Determination Communications in Computer and Information Scienc**e, v. 290, pp 295- 301, 2012.

FORMANSKI, F. N. **Aplicabilidade da Gamificação no Contexto Empresarial**, Florianópolis, SC, 88 p, 2016.

FOWLER, M. **Refatoração: Aperfeiçoando o design de códigos existentes**. 2. ed. [*S. l.*]: Novatec. ISBN 978-85-7522-724-4, 2019.

ELIZI, L. *et al*. **A game of refactoring: Studying the impact of gamification in software refactoring**, 2016. In Proceedings of the Scientific Workshop Proceedings of XP2016). Association for Computing Machinery, New York, NY, USA, Article 23, 1–6. https://doi.org/10.1145/2962695.2962718

GIMENEZ, F. E. *et al.* **Gamification approaches for open innovation implementation: A conceptual framework.** Creat Innov Manag. 2021; 30: 455–474. https://doi.org/10.1111/caim.12452

KAUR, S. *et al.* **How does object-oriented code refactoring influence software quality**? Research landscape and challenges, Journal of Systems and Software, Volume 157, 2019, 110394, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2019.110394.

KHANDELWAL, S. *et al*. **Impact of Gamification on Code review process: An Experimental Study**, 2017. In Proceedings of the 10th Innovations in Software Engineering Conference (ISEC '17). Association for Computing Machinery, New York, NY, USA, 122–126. https://doi.org/10.1145/3021460.3021474

KOIVISTO, J. *et al*. **Demographic differences in perceived benefits from gamification.** Computers in Human Behavior, 2014, 35, 179–188. https://doi.org/10.1016/j.chb.2014.03.007.

LACERDA, G. *et al.* **Code smells and refactoring: A tertiary systematic review of challenges and observations**, Journal of Systems and Software, Volume 167, 2020, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2020.110610.

MAJDENBAUM, A. *et al*.. **Comunicação em Projetos de Desenvolvimento Global de Software**: a visão dos praticantes. Gestão & Regionalidade, Rio Grande do Sul, v. 36, n. 107, p. 68-87, 1 abr. 2020.

MARTIN, R. C. **Código limpo**: Habilidades práticas do Agile Software. Rio de Janeiro: Alta Books. 456 p. ISBN 978-85-7608-267-5, 2011.

MEALY, E. *et al.* **Improving Usability of Software Refactoring Tools**, 2007 Australian Software Engineering Conference (ASWEC'07), 2007, pp. 307-318, doi: 10.1109/ASWEC.2007.24.

METRICSTREE. **Plugin Intellij.** Disponível em: https://plugins.jetbrains.com/plugin/13959-metricstree. Acesso em 01 de dezembro de 2022.

MURR, C. E. **Entendendo e aplicando a gamificação: o que é, para que serve, potencialidades e desafios**. Florianópolis UFSC, 36 p, 2020.

NEIDENBACH, S. F. *et al.* **Gamificação nas organizações: processos de aprendizado e construção de sentido**. Cadernos EBAPE.BR [online]. 2020, v. 18, n. spe [Acessado 1 Setembro 2022] , pp. 729-741. Epub 18 Dez 2020. ISSN 1679-3951. https://doi.org/10.1590/1679-395120190137.

OUNI, A. *et al.* **Search-based Refactoring: Towards Semantics Preservation**. IEEE International Conference on Software Maintenance, ICSM. 10.1109/ICSM.2012.6405292, 2012.

OUNI, A. *et al.* **Multi-Criteria Code Refactoring Using Search-Based Software Engineering: An Industrial Case Study**. ACM Transactions on Software Engineering and Methodology, Vol. 25, No. 3, Article 23, https://dl.acm.org/doi/10.1145/2932631, 2016.

PATRICIO, R. *et al.* **Gamification approaches to the early stage of innovation.** Creat Innov Manag. 2018; 27: 499– 511. https://doi.org/10.1111/caim.12284

RODRIGUES, I. *et al.* **Gamificação como uma nova tendência no processo de cocriação.** RAM, Revista de Administração Mackenzie, [*S. l.*], v. 22, n. 4, 14 jun. 2021. DOI https://doi.org/10.1590/1678-6971/eRAMR210132.

ROTH, S. *et al*. **The Ludic Drive as Innovation Driver. Creativity and Innovation Managemen**t, 24: 300-306, 2016. https://doi.org/10.1111/caim.12124.

RUHI, U. **Level up your strategy: Towards a descriptive framework for meaningful enterprise gamification**. Technology Innovation Management Review, 2015, 5(8), 5–16. https://doi.org/10.22215/timreview/918

SANTOS, H. *et al.* **CleanGame: Gamifying the Identification of Code Smells**, 2019. In Proceedings of the XXXIII Brazilian Symposium on Software Engineering (SBES '19). Association for Computing Machinery, New York, NY, USA, 437–446. https://doi.org/10.1145/3350768.3352490

SUH, A. *et al.* **The Effects of Game Dynamics on User Engagement in Gamified Systems**, 48th Hawaii International Conference on System Sciences, 2015, pp. 672-681, doi: 10.1109/HICSS.2015.87.

VALENTE, M. T. **Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade.** 1. ed. [*S. l.*: *s. n.*] 395 p. ISBN 978-65-00-00077-1. *E-book*, 2020.