**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA BAHIA**

Setembro de 2025

# Relatório técnico - Plataforma Colaborativa de Aprendizagem Gamificada

Hildemar Lemos de Santana
Kleberson de Jesus Sousa
Thiago Sampaio Santos

INSTITUTO FEDERAL
Bahia
Campus
Santo Antônio
de Jesus

## 1 - Introdução

Este relatório apresenta o desenvolvimento de uma plataforma colaborativa de aprendizagem gamificada, cujo objetivo é promover o engajamento dos alunos por meio de desafios interativos, pontuação dinâmica, conquistas e relatórios de desempenho. O sistema simula um ambiente educacional em que alunos participam de quizzes e exercícios de código, acumulam pontos, desbloqueiam medalhas e interagem com notificações e histórico de ações.

O projeto foi desenvolvido em Java, aplicando os princípios SOLID e os padrões de projeto do catálogo GoF, com foco em modularidade, extensibilidade e divisão equilibrada de responsabilidades entre os integrantes.

## 2 - Justificativa da Escolha dos Padrões de Projeto

A arquitetura do sistema foi planejada com a utilização de diversos padrões de projeto, cada um escolhido para resolver um problema específico e garantir a robustez e a flexibilidade da plataforma.

O padrão Singleton foi empregado para gerenciar recursos que devem ter uma única instância global. A classe SessaoUsuarioSingleton garante que a sessão do usuário atual seja acessada de forma consistente em toda a aplicação, enquanto a classe HistoricoInteracoes assegura que todos os registros de ações do usuário sejam centralizados em um único ponto, evitando inconsistências e duplicidade de dados.

Para a criação de objetos, optamos pelo padrão Factory Method. A classe PerfilFactory encapsula a lógica de criação de diferentes tipos de perfis de usuário, como Aluno, Professor e Visitante. Essa abordagem desacopla o código cliente da criação de objetos concretos, tornando o sistema mais fácil de estender para acomodar novos perfis no futuro.

A lógica de cálculo de pontuação nos desafios foi projetada com o padrão Strategy. A interface PontuacaoStrategy define o comportamento para calcular os pontos, e classes concretas como PontuacaoPorAcerto e PontuacaoPorDificuldade implementam essa lógica de maneira intercambiável. Essa flexibilidade permite que o sistema ofereça diversas formas de recompensar o desempenho do aluno, e a PontuacaoCompostaStrategy eleva o padrão, possibilitando a combinação de múltiplas lógicas de pontuação.

O padrão Observer foi fundamental para a funcionalidade de notificações em tempo real. Os desafios atuam como "sujeitos" que notificam seus "observadores" quando um evento, como a conclusão de um desafio, acontece. Os observadores, como EmailObserver (para envio de emails), LogObserver (para registrar eventos) e GamificacaoObserver (para atualizar o status do aluno), reagem de forma autônoma a esses eventos, garantindo um sistema de notificação modular e eficiente.

Para as funcionalidades de gamificação, optamos pelo padrão Decorator pois é ideal para a aplicação de bônus sobre as conquistas. A classe BonusDecorator serve como base para envolver uma conquista original com novas funcionalidades. Isso permitiu a criação de classes como DoubleXPBonus e StreakBonus, que adicionam pontos extras dinamicamente a uma medalha ou troféu, sem modificar a estrutura original da conquista.

O padrão Composite foi utilizado para organizar as conquistas de forma hierárquica. Classes como Medalha e Trofeu representam as conquistas individuais, enquanto ConjuntoMedalhasComposite e ConquistaPorCategoria atuam como contêineres que podem agrupar outras conquistas. Isso permite que a plataforma trate tanto uma única medalha quanto um conjunto de conquistas da mesma forma, simplificando a exibição e o cálculo de valores.

Para simplificar a complexidade do subsistema de relatórios, o padrão Facade foi implementado na classe RelatorioFacade. Essa classe fornece uma interface única e simplificada para gerar relatórios em múltiplos formatos (PDF, CSV, JSON), orquestrando as operações das classes RelatorioPDF, RelatorioCSV e RelatorioJSON, e isolando a lógica de negócio do cliente da complexidade de geração de arquivos.

O padrão Adapter foi empregado para permitir a integração com um serviço de ranking externo simulado. A classe RankingAdapter atua como um tradutor, convertendo a interface do serviço externo (ServicoRankingExterno) para um formato compatível com o sistema interno, o que demonstra a capacidade da arquitetura de se adaptar a APIs de terceiros.

Por fim, o padrão Command foi essencial para a funcionalidade de "desfazer" ações. Cada interação do usuário que modifica o estado do sistema, como responder um quiz ou desbloquear uma medalha, é encapsulada em um objeto de comando (ResponderQuizCommand, DesbloquearMedalhaCommand). Esses comandos são armazenados em uma pilha na classe HistoricoInteracoes e, quando um usuário solicita o "undo", o último comando na pilha é executado de forma reversa, restaurando o estado anterior do sistema de maneira segura e confiável.

## 3 - Análise de como os princípios SOLID foram contemplados

O Princípio da Responsabilidade Única (Single Responsibility Principle - SRP) foi aplicado ao garantir que cada classe tenha um único motivo para mudar. Por exemplo, a classe RelatorioPDF é exclusivamente responsável por formatar e gerar o arquivo PDF, a classe Log.java gerencia apenas a escrita de logs, e a classe Aluno lida unicamente com o estado e as funcionalidades de um aluno.

O Princípio do Aberto/Fechado (Open/Closed Principle - OCP) é evidente na utilização das interfaces. As classes estão abertas para extensão, mas fechadas para modificação. Se for necessário adicionar uma nova forma de calcular a pontuação, basta criar uma nova classe que implemente a interface PontuacaoStrategy sem precisar alterar a classe Desafio. Da mesma forma, novas classes de relatórios podem ser adicionadas sem modificar a RelatorioFacade.

O Princípio da Substituição de Liskov (Liskov Substitution Principle - LSP) é respeitado pela hierarquia de classes Usuario. As subclasses Aluno, Professor e Visitante podem ser usadas de forma transparente em qualquer lugar onde um Usuario é esperado, como em um método de login que aceita a classe base, sem causar comportamentos inesperados.

O Princípio da Segregação de Interfaces (Interface Segregation Principle - ISP) foi seguido ao criar interfaces pequenas e específicas. A interface Relatório se concentra apenas na funcionalidade de geração de relatórios, enquanto a interface Acão se foca exclusivamente em ações que podem ser executadas e desfeitas. Isso evita que as classes sejam obrigadas a implementar métodos que não utilizam.

Por fim, o Princípio da Inversão de Dependência (Dependency Inversion Principle - DIP) foi fundamental. As classes de alto nível não dependem de classes de baixo nível, mas sim de abstrações. Por exemplo, a classe Desafio depende da interface PontuacaoStrategy em vez de uma implementação concreta, permitindo a injeção de diferentes lógicas de pontuação. Da mesma forma, a classe HistoricoInteracoes depende da interface Acao, tornando-a independente dos comandos específicos que ela executa.

## 4 - Trade-offs e limitações

Apesar da robustez da arquitetura orientada a padrões, o projeto apresenta algumas limitações que foram resultado de trade-offs conscientes e alinhados com os objetivos principais da disciplina.

A aplicação de múltiplos padrões de projeto, especialmente os mais complexos como o Command e o Composite, resultou em uma complexidade inicial mais elevada no código. O processo de modelagem e design para garantir que cada classe tivesse uma responsabilidade única e se encaixasse corretamente nos padrões exigiu maior esforço na fase inicial de desenvolvimento. No entanto, esse trade-off foi considerado aceitável, pois garantiu uma arquitetura altamente escalável e de fácil manutenção no longo prazo, cumprindo o critério de Arquitetura e SOLID do projeto.

A ausência de uma interface gráfica avançada (UI) é uma limitação visível na interação com o usuário, que se dá unicamente por meio do console. Essa escolha foi estratégica para direcionar todo o tempo de desenvolvimento para a lógica de negócio e o design da arquitetura de backend. A simplicidade da interface permitiu que o foco do trabalho fosse a demonstração prática dos padrões de projeto e dos princípios SOLID, que são o cerne da atividade, sem o esforço adicional de um framework de UI. A separação entre a lógica de negócio e a interface de usuário também reforça o Princípio da Responsabilidade Única (SRP).

Por fim, a persistência dos dados e a integração externa foram simplificadas para viabilizar o escopo do projeto. Os dados são armazenados em arquivos de texto (CSV, JSON), que são suficientes para a demonstração, mas seriam inadequados para um cenário de produção. Um banco de dados real seria necessário para garantir integridade, concorrência e escalabilidade. Da mesma forma, a integração com o ranking global é simulada com um serviço fictício. Ambas as limitações foram adotadas para manter o escopo do projeto viável e permitir a demonstração dos padrões Facade e Adapter sem a complexidade de um banco de dados e de APIs externas reais.

## 5 - Possíveis Extensões Futuras

A arquitetura modular e extensível da plataforma, construída sobre os princípios SOLID e os padrões de projeto, permite várias melhorias e funcionalidades futuras com um esforço de modificação reduzido.

A principal extensão seria a implementação de uma interface de usuário (UI) mais sofisticada, seja ela web ou mobile. A lógica de negócio, já desacoplada da camada de apresentação, poderia ser facilmente exposta por uma API RESTful construída com um framework como Spring Boot. Essa nova camada de apresentação consumiria os serviços e comandos já existentes, como a classe SessaoUsuarioSingleton para gerenciar a sessão e a classe HistoricoInteracoes para registrar e desfazer ações. Essa abordagem evita a necessidade de reescrever o núcleo do sistema, demonstrando a flexibilidade e a escalabilidade da arquitetura.

Outra extensão seria a criação de um dashboard de desempenho interativo que permitiria aos usuários visualizar seu progresso, conquistas e histórico de forma visualmente atraente. Dados brutos do sistema, como o histórico de interações e os pontos acumulados, poderiam ser transformados em gráficos de evolução, um mapa de medalhas desbloqueadas e um ranking visual. Além disso, a integração com APIs reais de estatísticas educacionais ou plataformas de autenticação externas poderia substituir os serviços simulados, aumentando o valor e a funcionalidade do projeto. A capacidade de integrar APIs externas demonstra a robustez do padrão Adapter utilizado.

Por fim, o sistema poderia ser aprimorado com um mecanismo adaptativo de desafios, onde um algoritmo ajustaria a dificuldade dos quizzes com base no desempenho do aluno. Essa funcionalidade poderia ser implementada simplesmente adicionando uma nova estratégia à interface PontuacaoStrategy. Por exemplo, uma nova classe PontuacaoAdaptativa poderia analisar o histórico de respostas do usuário e usar a classe ComandoComposite para montar uma nova estratégia de pontuação que misturasse bônus e penalidades, demonstrando a verdadeira flexibilidade e o poder do design de software orientado a padrões.

## Considerações sobre a Colaboração do Grupo

Durante o desenvolvimento deste projeto, todos os integrantes participaram ativamente das etapas de codificação, documentação e apresentação. A divisão de tarefas foi respeitada conforme o planejamento inicial, mas o trabalho foi conduzido de forma colaborativa e integrada.

Thiago liderou a parte de Arquitetura e Modelagem, sendo responsável pela definição da estrutura geral do sistema, elaboração dos diagramas UML e aplicação de padrões de projeto como Singleton, Factory, Decorator, Composite, Facade e Adapter.

Hildemar focou na implementação do Backend e da Lógica de Negócio, desenvolvendo os módulos centrais relacionados a usuários, desafios, pontuação e histórico, além de aplicar os padrões Observer e Command para garantir flexibilidade e escalabilidade ao sistema.

E Kleberson ficou encarregado da Interação e Relatórios, implementando a geração de relatórios em diversos formatos (PDF, CSV, JSON), integração com serviços externos de ranking e contribuindo com a documentação técnica e estrutura da apresentação final.

Apesar da divisão clara de responsabilidades, o grupo atuou de forma colaborativa em todas as etapas. Sempre que surgiam dúvidas ou desafios técnicos, os demais integrantes se prontificavam a ajudar, oferecendo sugestões, revisando trechos de código e contribuindo para decisões de arquitetura e design.