Trabalho Final da Disciplina de Programação Modular

Prof. Paulo Henrique D. S. Coelho

2025

Professor: Paulo Coelho
GitHub: https://github.com/paulohdscoelho

Objetivo

O objetivo do trabalho é implementar um sistema de gestão de folha de pagamento para um software de gestão de recursos humanos (RH) de uma empresa. A aplicação deve seguir princípios de modularidade, programação orientada a objetos (POO), usabilidade, padrões SOLID e qualidade de software. Além do mais, a aplicação deve contar com uma boa cobertura de testes unitários para suas funções e métodos.

A aplicação deverá obrigatoriamente possuir um **módulo de autenticação**, com login e senha, de forma que apenas usuários autenticados possam acessar o sistema de folha de pagamento.

As aplicações devem ser implementadas em linguagem Java, utilizando o framework **Spring Boot**. É esperado que os alunos desenvolvam um sistema backend exposto através de chamadas de função ou endpoints consumidas por um frontend web.

1 Avaliação

O trabalho vale 20 pontos que serão igualmente distribuídos ao longo de cada Sprint, valendo 5 pontos cada.

A turma será dividida em grupos de até 5 alunos, formados da seguinte maneira:

- Os grupos devem obrigatoriamente ser compostos por alunos que cursam a disciplina de **laboratório** juntos, uma vez que essas aulas serão utilizadas para implementar o trabalho.
- Os alunos que não estiverem matriculados em laboratório devem formar grupos entre si.

As entregas devem ser feitas através do **Canvas da disciplina**, com a submissão do link para o repositório no GitHub do grupo e da apresentação correspondente a cada Sprint.

Estrutura do Trabalho

O trabalho será dividido em quatro sprints, com entregas evolutivas:

Sprint 1 – Análise e Modelagem (5 pts.)

- Análise do estudo de caso e interpretação do conceito de domínio.
- Identificação de requisitos do sistema.
- Produção de Cartões CRC (Classe – Responsabilidade – Colaborador) para as classes candidatas.
- Modelagem de um Diagrama de Classes conforme apresentado em aula.
- **Modelagem inicial das classes do frontend (UI), descrevendo as principais telas e interações esperadas.**
- **Planejamento dos testes unitários (não é necessário implementar ainda, apenas descrever os casos de teste que serão desenvolvidos nas próximas sprints).**
- Implementação inicial do esqueleto do sistema em Java Spring Boot.
- **Entregáveis:** Cartões CRC, Diagrama de Classes, esqueleto do projeto no GitHub, modelos das classes do frontend e plano de testes unitários.

Sprint 2 – Herança, Interfaces e Testes Unitários (5 pts.)

- Implementação de testes unitários para as funcionalidades desenvolvidas na sprint 1.
- Aplicação de conceitos de reuso de software através de Herança.

- Especialização e separação de responsabilidades utilizando Interfaces e Classes Abstratas.
- **Entregáveis:** Código atualizado no GitHub, relatório explicativo, apresentação demonstrando os testes unitários.

Sprint 3 – Polimorfismo Paramétrico, Coleções/Streams, Persistência e Eventos (5 pts.)

- Implementação de Polimorfismo Paramétrico (Generics) no sistema.
- Uso de coleções (List, Set, Map) para manipulação dos dados da folha de pagamento.
- Emprego de Streams para processamento e filtragem de dados.
- Persistência dos dados em banco de dados relacional à escolha do grupo (MySQL, SQLite, PostgreSQL, etc.).
- Implementação de Eventos (ex.: cadastro de funcionário dispara log; geração de folha dispara notificação).
- Preparação da arquitetura para futura integração com o frontend.
- Implementação de testes unitários para as novas funcionalidades.
- **Entregáveis:** Código atualizado no GitHub, relatório explicativo, apresentação com demonstração do uso de eventos.

Sprint 4 – Frontend Web, Integração e Padrões de Projeto (5 pts.)

- Implementação de um frontend web simples, com tecnologia de escolha do grupo (ex.: HTML/CSS/JavaScript, React, Angular, Vue ou Thymeleaf no Spring Boot).
- Consumo dos endpoints REST expostos pelo backend para operações do sistema (cadastro de funcionários, cálculo da folha, listagem de pagamentos, etc.).
- Aplicação de padrões de projeto (ex.: Factory, Singleton, Strategy) no backend.
- Integração completa entre frontend e backend.

- **Entregáveis:** Sistema completo (frontend + backend + testes unitários), código no GitHub e apresentação final com demonstração funcional.

Resumo das Sprints

Sprint	Principais Atividades	Entregáveis
Sprint 1 (Análise e Modelagem)	- Análise do estudo de caso e requisitos - Cartões CRC - Diagrama de Classes - Esqueleto do sistema em Spring Boot	Cartões CRC, Diagrama, Esqueleto no GitHub
Sprint 2 (Herança, Interfaces e Testes)	- Implementação de testes unitários - Reuso com Herança - Interfaces e Classes Abstratas - Demonstração dos testes	Código no GitHub, relatório, apresentação dos testes
Sprint 3 (Polimorfismo, Streams, Persistência e Eventos)	- Generics (Polimorfismo Paramétrico) - Coleções e Streams - Persistência em BD (MySQL, PostgreSQL, etc.) - Eventos (cadastro, geração de folha) - Preparação para integração com frontend - Novos testes unitários	Código no GitHub, relatório, apresentação dos eventos
Sprint 4 (Frontend, Integração e Padrões de Projeto)	- Desenvolvimento de frontend (tecnologia livre: React, Angular, Vue, Thymeleaf, etc.) - Consumo da API REST - Aplicação de padrões de projeto - Integração frontend + backend	Sistema completo (frontend + backend + testes), código no GitHub, apresentação final

Orientações Gerais

Será criado um fórum no **Canvas da disciplina de Programação Modular** para discutir cada uma das sprints. Os alunos poderão postar suas dúvidas nesse fórum, sendo incentivada a **discussão e colaboração entre os grupos**, mas **sem a cópia de código**.

Todos os anúncios oficiais da disciplina serão realizados pelo Canvas.

O código do trabalho deve ser disponibilizado em um repositório no **GitHub**, criado e mantido por cada grupo.

É esperado que **todos os membros** do grupo programem e implementem partes do código. A avaliação disso será feita através de análise do histórico

de commits do projeto, pelo qual é possível ver quais alunos estão de fato contribuindo para a aplicação.

Uso de Ferramentas de IA Generativa

Não é incentivado o uso de ferramentas de IA generativa (como Copilot, Cursor, ChatGPT e outras) para implementar os códigos do trabalho. O motivo é simples: a melhor forma de se tornar um programador eficiente e versado na arte é **programando por conta própria**.

Dúvidas sobre frameworks, uso de bibliotecas e conceitos vistos em sala podem, sim, ser sanadas via essas ferramentas. Também é aceitável recorrer a elas em casos de erros inesperados relacionados a conversão de tipos, instanciamento ou falhas no framework.

Entretanto, **não será aceito que os alunos simplesmente insiram a especificação no GPT e colem o código gerado de forma acrítica**. Isso não é programar: é **engenharia de prompt preguiçosa**. Além disso, é perigoso, uma vez que tais ferramentas podem cometer erros e, após repetidas interações, tendem a apresentar respostas inconsistentes ou incorretas.

Ao final de cada sprint, os códigos serão submetidos a verificadores de IA. Caso seja identificado que o código foi **100% gerado de forma acrítica por IA**, o grupo será notificado. Se o problema não for corrigido, os membros do grupo serão penalizados com redução proporcional na nota final da sprint.

Exemplo: como cada sprint vale 5 pontos, se for identificado que 100% do código foi gerado por IA, a nota será:

$$5 - (5 \times 1) = 0$$