Relatório Final do Trabalho de POO: Sistema de Gerenciamento de Biblioteca

Integrantes do Trabalho:

- **Caius Vinícius**
- **Igor Ferreira**
- **Jose Victor**
- **Renan**
- **Yan Magnun**

1. Introdução ao Projeto

1.1 Visão Geral e Objetivo

O projeto consiste no desenvolvimento de um **Sistema de Gerenciamento de Biblioteca** em Java. O sistema final é uma aplicação completa com **Interface Gráfica (GUI)** em Swing, que atende aos requisitos funcionais, como cadastro, busca e controle de empréstimo/devolução, aplicando rigorosamente os princípios de **Programação Orientada a Objetos (POO)**. A **persistência de dados** é garantida através da manipulação de arquivos **CSV**, permitindo o armazenamento e a recuperação eficiente das informações.

1.2 Fluxo Geral do Sistema

O fluxo central do sistema é orquestrado pela classe **`Catalogo`**, que foi reestruturada para centralizar toda a lógica de negócio. Ao iniciar, o `Catalogo` carrega todos os dados (Livros, Clientes, Funcionários) dos arquivos CSV para a memória. A interação acontece inteiramente via interface gráfica. Nela, **Clientes submetem pedidos de aluguel**, e **Funcionários** são responsáveis por **aprovar ou rejeitar** esses pedidos, adicionando uma camada de segurança e clara separação de papéis no sistema.

2. Estrutura do Código, Decisões de Design e Aplicação dos Princípios de POO

A estrutura do código foi cuidadosamente organizada em camadas, separando as responsabilidades de Entidade, Serviço de Negócio e Persistência. O design das classes

foi fundamental para cumprir o requisito de POO, demonstrando a aplicação dos princípios de forma coerente:

- **Abstração:** A **Abstração** é aplicada primariamente através da classe **`Usuario`**, que é declarada como *abstract*, definindo o conceito genérico de pessoa no sistema, mas impedindo sua instanciação direta. A interface **`CSVGravavel`** reforça a abstração, estabelecendo um contrato de serialização que todas as entidades persistíveis devem seguir.

- **Herança:** As classes **`Cliente`** e **`Funcionario`** demonstram a **Herança** ao estenderem a classe base **`Usuario`**. Essa decisão garante a **reutilização** dos atributos comuns, como `nome` e `idade`, e a uniformidade na inicialização através da chamada `super()` em seus construtores.

- **Polimorfismo:** O **Polimorfismo** é evidenciado pelo fato de que **`Cliente`**, **`Funcionario`** e **`Livro`** implementam a interface **`CSVGravavel`**. Isso permite que o método **`CSVUtil.gravarCSV()`** trate os diferentes tipos de objetos de forma **uniforme**, chamando o método `toCSV()` de cada entidade sem ter conhecimento do tipo exato do objeto, apenas de seu contrato.

- **Encapsulamento:** O **Encapsulamento** é respeitado em todas as classes de entidade (`Cliente`, `Funcionario`, `Livro`), nas quais todos os atributos são definidos como `private`, e o acesso e modificação do estado são estritamente controlados por métodos públicos *Getters* e *Setters*.

2.1 Qualidade do Código e Refatoração

A classe **`Catalogo.java`** foi reestruturada para centralizar o controle de coleções, a lógica de pedidos e a persistência, o que é uma boa prática para tornar o sistema mais **organizado e seguro**. Além disso, a classe **`LivroModeloDeTable`** atua como um **Adaptador** vital para a interface gráfica, traduzindo objetos `Livro` para o formato tabular esperado pelo componente Swing, garantindo a separação entre o modelo de dados e a apresentação.

3. Implementação e Funcionalidades

3.1 Funcionalidades de Interface e Segurança

A migração para a Interface Gráfica (GUI) introduziu segurança e melhor separação de responsabilidades:

- **Interface Gráfica Swing:** Todo o sistema foi **reformado** para funcionar integralmente com Swing, eliminando as interações via terminal (`Scanner`).

- **Controle de Acesso (Login):** Foi implementada uma tela inicial de **Login com senha** para Clientes e Funcionários.

- **Separação de Papéis:** Os **Funcionários** são os únicos com acesso às áreas de **Registro de Livros/Clientes** e, crucialmente, ao painel de **Aprovação de Pedidos**, garantindo a segurança operacional do sistema.

3.2 Lógica Central de Pedido e Aprovação

A lógica de empréstimo foi refatorada para um sistema de **Pedido e Aprovação**:

- **Pedidos de Aluguel:** Clientes enviam um **pedido de aluguel** (`addRequest`) ao `Catalogo`, em vez de alugar diretamente.

- **Aprovação pelo Funcionário:** A aprovação do pedido pelo funcionário dispara automaticamente a atualização de todo o sistema: marca o livro como indisponível, registra o aluguel para o cliente, define a data de devolução e atualiza o *status* de dívida.

3.3 Desafios e Soluções

O processo de **Serialização Manual** representou um desafio, pois a leitura dos dados em **`CSVUtil.lerCSV`** exigiu uma deserialização complexa com condicionais (`if... else if`) para reconstruir corretamente os objetos a partir das *strings* CSV. Além disso, o alto **Acoplamento** entre as classes, notavelmente no método `Cliente.alugarLivro()`, foi identificado como um ponto de melhoria futura.

4. Conclusão e Experiência do Grupo

4.1 Contribuição Individual

O projeto foi dividido em camadas, garantindo que cada membro fosse responsável por uma área essencial do sistema:

- **Igor Ferreira** ficou responsável pela **Documentação, Relatório e Gráficos**. Sua contribuição envolveu a **formatação, redação e conclusão final do Relatório** do projeto, além da inclusão de **Gráficos** e elementos visuais de apoio (como diagramas UML) para a documentação.

- **Jose Victor** foi o responsável pela **Modelagem Inicial de Entidades**. Ele criou as classes iniciais **`Cliente`**, **`Funcionario`**, e **`Livro`**, definindo seus atributos, e desenvolveu o ponto de entrada **`Main`** (Console).

- **Renan** trabalhou na **Implementação de POO e Lógica de Negócio Base**. Ele foi responsável pelo design da classe **`Usuario` (abstrata)**, aplicando a **Herança**, e criou os algoritmos de **busca** e o método base de **alugar e devolver livro** no console.
- **Yan Magnun P.C. Modesto** dedicou-se à **Camada de Persistência (CSV)**. Sua contribuição chave foi a criação da interface **`CSVGravavel`** e o desenvolvimento do **`CSVUtil`**, implementando o **Polimorfismo (Generics)** para a gravação e a lógica de deserialização.

- **Caius Vinícius** liderou a **Interface Gráfica (GUI) e Refatoração da Lógica Central**. Ele realizou a **Migração total para Swing GUI**, implementou o fluxo de **Pedido/Aprovação** para aluguéis, e reestruturou o **`Catalogo`** para centralizar toda a lógica, incluindo o sistema de **Login com senha** e separação de papéis.

4.2 Experiência Pessoal

O projeto consolidou o entendimento sobre como a **Abstração e o Polimorfismo** são cruciais para a **modularidade** de um sistema. A evolução do sistema, do console para a GUI com segurança e fluxo de trabalho complexo, demonstrou o valor da **refatoração contínua** e da **separação de papéis**, superando os desafios iniciais de estruturação e acoplamento.