Relatório de Produção Individual Final

Aluno: Daired Almeida Cruz

1. Atribuição de Cargo e Tarefas

Minha atribuição no projeto foi a de desenvolver cliente desktop quanto o middleware de integração.

Na prática, minhas tarefas foram:

- **Arquitetura do Software :** Definir e implementar a arquitetura MVC (Model-View-Controller) para a aplicação Desktop, garantindo a separação de responsabilidades. Adicionalmente, desenhei a arquitetura do Módulo Integrador com Apache Camel, focando em rotas desacopladas e processadores para a transformação de dados.
- **Desenvolvimento da Aplicação Desktop (Frontend/Backend Local):**
    - **Interface Gráfica (GUI):** Criei todas as telas da aplicação utilizando JavaFX e FXML, com uma navegação centralizada em abas.
    - **Persistência de Dados:** Configurei o banco de dados local SQLite e implementei toda a camada de acesso a dados com ORMLite, utilizando um padrão de repositório genérico para as operações CRUD.
    - **Funcionalidades Core:** Implementei as operações de CRUD para todas as entidades do sistema: Livros, Autores, Usuários, Categorias, Empréstimos e Resenhas.
- **Desenvolvimento do Módulo Integrador (Middleware):**
    - **Rotas de Integração:** Desenvolvi as rotas com Apache Camel para orquestrar o fluxo de mensagens entre o Desktop e a API.
    - **Transformação de Dados:** Criei os processadores para converter os objetos de um sistema para o outro, incluindo a lógica de mapeamento canônico de IDs.
- **Qualidade e Documentação:** Garanti a qualidade do código com testes unitários (JUnit) iniciais e basicos para os repositórios e elaborei a documentação técnica inicial. Vale ressaltar que foi o Hugo que finalizou essa parte, documentando por completo o sistema e código.

2. Contribuição de Acordo com a Atribuição

Cumpri integralmente as tarefas que me foram atribuídas, entregando uma aplicação desktop funcional e um módulo integrador que garante a sincronia entre os sistemas.

- **Commits Mais Relevantes (Aplicação Desktop):**
    - **https://github.com/SPD-BES-2025-3/grupo8/tree/main/biblioteca-desktop-app**
    - **Implementação arquitetura MVC com AbstractCrudController e telas FXML:** Este commit estabeleceu a fundação da aplicação desktop, criando o controlador abstrato que centraliza toda a lógica de CRUD e as interfaces gráficas FXML para cada entidade, o que acelerou o desenvolvimento subsequente.

        - **FMXL:** https://github.com/SPD-BES-2025-3/grupo8/tree/main/biblioteca-desktop-app/src/main/resources/view
        - **MVC:**
            - **Model:** https://github.com/SPD-BES-2025-3/grupo8/tree/main/biblioteca-desktop-app/src/main/java/model
            - **View:** https://github.com/SPD-BES-2025-3/grupo8/tree/main/biblioteca-desktop-app/src/main/java/view
            - **Controller:** https://github.com/SPD-BES-2025-3/grupo8/tree/main/biblioteca-desktop-app/src/main/java/controller
    - **Configura persistência com ORMLite, repositório genérico e modelos:** Neste commit, configurei a conexão com o banco de dados SQLite e criei o padrão de repositório genérico, uma peça chave que simplificou e padronizou o acesso a dados para todas as entidades do sistema.
        - https://github.com/SPD-BES-2025-3/grupo8/blob/main/biblioteca-desktop-app/src/main/java/model/Repositorio.java
        - https://github.com/SPD-BES-2025-3/grupo8/blob/main/biblioteca-desktop-app/src/main/java/model/Repositorios.java
    - **Adiciona testes de unidade (JUnit) para todos os repositórios CRUD:** Commit fundamental que garantiu a robustez e a confiabilidade da camada de persistência, cobrindo todas as operações de criação, leitura, atualização e exclusão para as entidades principais.
        - https://github.com/SPD-BES-2025-3/grupo8/tree/main/biblioteca-desktop-app/src/test/java/model
- **Commits Mais Relevantes (Módulo Integrador):**
    - **https://github.com/SPD-BES-2025-3/grupo8/tree/main/integrador**
    - **Estrutura rota de integração com Apache Camel (Desktop -> API e API -> Desktop):** Representa a criação do LivrariaRouteBuilder, o coração do integrador. Este commit definiu o fluxo principal que consome mensagens da fila do desktop, processa e as envia para a API REST, tratando diferentes operações (CREATE, UPDATE, DELETE) e vice-versa.
        - https://github.com/SPD-BES-2025-3/grupo8/tree/main/integrador/src/main/java/com/livraria/integrador/routes
    - **Cria processadores de transformação e DTOs de sincronização:** Essencial para a comunicação, este commit introduziu o DesktopToApiProcessor e os DTOs (LivroSyncDto, ResenhaSyncDto), implementando a lógica de conversão do modelo de dados do desktop para o formato esperado pela API. O caminho contrário foi feito com o ApiToDesktopProcessor.
        - **Processadores:** https://github.com/SPD-BES-2025-3/grupo8/tree/main/integrador/src/main/java/com/livraria/integrador/processor
        - **DTOs:** https://github.com/SPD-BES-2025-3/grupo8/tree/main/biblioteca-desktop-app/src/main/java/model/dto

    - **Implementa mapeamento canônico de IDs para sincronização:** Resolve um dos maiores desafios da integração. Este commit criou o MapeamentoID e o RepositorioCanonico, permitindo a tradução de chaves primárias entre o banco relacional (Integer) e o NoSQL (String), garantindo a integridade da sincronização.
        - **RepositorioCanonico:** https://github.com/SPD-BES-2025-3/grupo8/tree/main/integrador/src/main/java/com/livraria/integrador/repository
        - **Mapeamentos:**
            - https://github.com/SPD-BES-2025-3/grupo8/blob/main/integrador/src/main/java/com/livraria/integrador/processor/PersistenciaCanonicoProcessor.java
            - https://github.com/SPD-BES-2025-3/grupo8/blob/main/integrador/src/main/java/com/livraria/integrador/model/canonico/MapeamentoID.java

3. Principais dificuldades:

- A maior dificuldade inicial foi no desenho das interfaces com JavaFX, tecnologia na qual eu tinha menos experiência.
- A modelagem inicial dos diagramas UML e a definição do escopo do projeto demandaram um tempo considerável de discussão e alinhamento com o meu colega para chegarmos a um consenso.
- Dificuldades consideráveis no integrador para achar a lógica de conversão entre os 2 DBs, mas com a ajuda do Hugo consegui finalizar e deixar tudo funcional.

4. Contribuição Além do Atribuído

Embora a codificação tenha sido dividida (Desktop/Integrador para mim, API, docker e documentação para o Hugo), minha contribuição se estendeu à arquitetura geral do sistema e ajudas pontuais na API, por conta do integrador. Colaborei ativamente na definição das bases do projeto, no desenho dos fluxos de comunicação e na modelagem UML.

Auxiliei meu colega a entender a estrutura dos objetos ORM do desktop para que ele pudesse consumi-los e manipulá-los corretamente nas conversões do lado da API e do integrador.

5. Considerações Gerais

- **O que aprendi:**
    - **Técnicas:** Aprofundei meus conhecimentos práticos em arquiteturas orientadas a eventos, o uso de frameworks como JavaFX, ORMLite e, especialmente, Apache Camel para integração de sistemas. A implementação de um middleware real foi o maior ganho técnico.
    - **Conceituais:** A implementação do repositório e do controlador abstrato no desktop, combinada com os processadores e rotas no integrador, solidificou

meu entendimento sobre reutilização de código, design patterns (como o Processor do Camel) e os princípios de sistemas desacoplados e assíncronos.

- **Ajuda do Grupo:** Um dos pontos fortes do projeto foi o trabalho em equipe. A maior parte do tempo tentamos manter uma boa comunicação do que estávamos fazendo para evitar retrabalho e termos noção do que estava sendo feito em ambos os ambientes. Além disso, ajudamos muito um ao outro nos projetos individuais, que no meu caso do Desktop+Integrador foi essencial a ajuda do Hugo, ele me auxiliou principalmente nas lógicas da parte do Controller que eram mais complexas e no Integrador foi na parte da implementação das Filas e Jms. Com a “Dockerização” feita ao fim por ele deixou a aplicação bem mais fácil de ser inicializada e gerenciada.
- **Conclusão:** A base da aplicação desktop e do módulo integrador foi estabelecida com sucesso. O uso de padrões de projeto e boas práticas resultou em um código desacoplado, de fácil manutenção e com alta legibilidade.