RELATÓRIO TÉCNICO: ARQUITETURA DO SISTEMA E-LIBRARY

Aluno: Gabriel Afonso Barbosa

Disciplina: SERVIÇOS DE SUPORTE A APLICAÇÕES DISTRIBUÍDAS

Data: 01/12/2025

1. Introdução O sistema E-Library foi desenvolvido utilizando a arquitetura Java Enterprise (Jakarta EE) distribuída, visando centralizar a lógica de negócio e garantir escalabilidade e segurança. A solução foi dividida em módulos EAR (Enterprise Archive), separando a camada de apresentação (Web/JSF) da camada de negócio (EJB).

2. Justificativa das Escolhas Arquiteturais (EJB Session Beans)

Para atender aos requisitos de negócio, foram utilizados três tipos distintos de Session Beans, cada um com um propósito específico:

- **Singleton Session Bean (CatalogStatusSB):** Foi escolhido o padrão Singleton para gerenciar o **status global do acervo**. Como a contagem de livros e exemplares disponíveis é uma informação de alto nível que precisa ser compartilhada entre todos os usuários e clientes (Web e Desktop), o Singleton atua como um cache centralizado.
    - *Vantagem:* Evita consultas repetitivas e pesadas ao banco de dados (SELECT COUNT) a cada requisição, melhorando a performance. A anotação @Startup garante que o cache seja aquecido e o usuário Admin seja criado assim que o servidor inicia.
- **Stateful Session Bean (UsuarioSB):** Utilizado para gerenciar a **sessão do usuário**. Diferente de uma aplicação puramente Stateless, o processo de empréstimo em uma biblioteca requer contexto: o sistema precisa saber "quem" está realizando a operação.
    - *Vantagem:* O Bean mantém o estado (usuarioLogado) entre chamadas de métodos subsequentes. Isso simplifica a lógica de negócio, pois não é necessário enviar o ID do usuário em cada requisição de empréstimo ou devolução; o contêiner EJB já sabe qual instância pertence a qual cliente.
- **Stateless Session Bean (LivroSB, ExemplarSB, UsuarioCrudSB):** Utilizados para as operações de **CRUD (Create, Read, Update, Delete)**. Como cadastrar um livro ou listar o acervo são operações atômicas que não dependem de um estado conversacional prévio, o modelo Stateless é o mais eficiente.

- *Vantagem:* O servidor pode manter um *pool* de instâncias e reutilizá-las para atender milhares de requisições simultâneas, garantindo alta escalabilidade e menor consumo de memória em comparação ao Stateful.

3. Conclusão A arquitetura adotada respeita os princípios de separação de responsabilidades. A persistência foi isolada via JPA/Hibernate no PostgreSQL, garantindo integridade dos dados, enquanto os clientes (Web JSF e Desktop Java SE) atuam apenas como interfaces de consumo da lógica centralizada no servidor de aplicação WildFly.