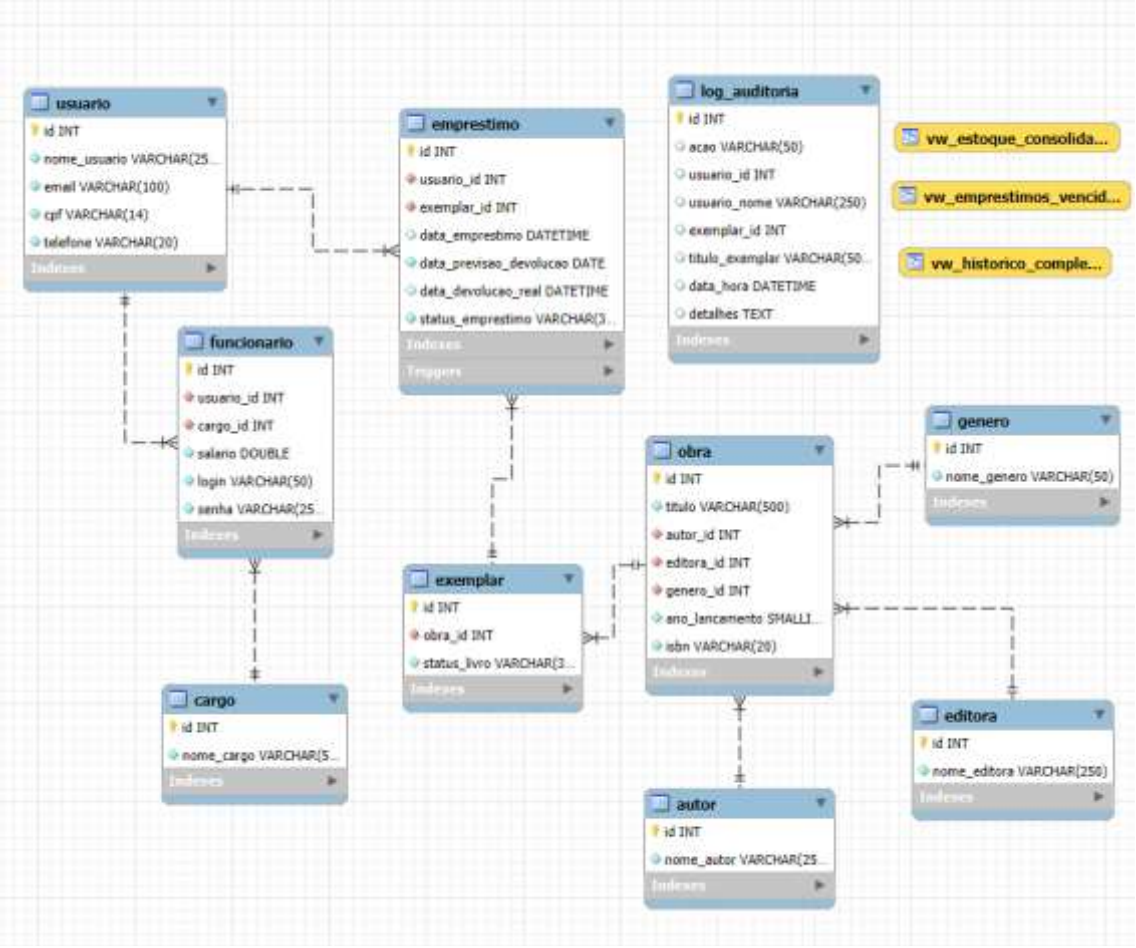
- Diagrama de MER



- Modelo Lógico

1. Cadastros Básicos

Estas são tabelas independentes que servem para categorizar as obras e pessoas.

- **AUTOR**
  - `id` **(PK)**
  - `nome_autor`
- **EDITORA**
  - `id` **(PK)**
  - `nome_editora`
- **GENERO**
  - `id` **(PK)**
  - `nome_genero`
- **CARGO**
  - `id` **(PK)**
  - `nome_cargo`

---

2. Cadastro de Pessoas e Acesso

Aqui temos uma estrutura onde `Funcionario` é uma extensão de `Usuario`.

- **USUARIO**
  - `id` **(PK)**
  - `nome_usuario`
  - `email`
  - `cpf`
  - `telefone`
- **FUNCIONARIO**
  - `id` **(PK)**
  - `usuario_id` **(FK)** *(Relacionamento 1:1 - Unique)*
  - `cargo_id` **(FK)**
  - `salario`
  - `login`
  - `senha`

---

3. Cadastro de Acervo

Separa a obra do item físico (exemplar).

- **OBRA**
  - `id` **(PK)**
  - `titulo`
  - `autor_id` **(FK)**
  - `editora_id` **(FK)**
  - `genero_id` **(FK)**
  - `ano_lancamento`
  - `isbn`
- **EXEMPLAR** (Representa o livro físico na estante)
  - `id` **(PK)**
  - `obra_id` **(FK)**
  - `status_livro` *(Disponível, Emprestado, etc.)*

---

4. Cadastro de Operação (Movimentação)

Onde ocorre a ação principal do sistema.

- **EMPRESTIMO**
  - `id` **(PK)**
  - `usuario_id` **(FK)**
  - `exemplar_id` **(FK)**
  - `data_emprestimo`
  - `data_previsao_devolucao`
  - `data_devolucao_real`
  - `status_emprestimo`
- **LOG_AUDITORIA** (Tabela de histórico/segurança)
  - `id` **(PK)**
  - `acao`
  - `usuario_id` *(Armazenado como valor, sem FK restrita para manter histórico)*
  - `usuario_nome`
  - `exemplar_id`
  - `titulo_exemplar`
  - `data_hora`
  - `detalhes`

Resumo dos Relacionamentos

1. **Usuário <-> Funcionário (1:1):** Um funcionário **é** um usuário. A tabela funcionario herda os dados de usuário através do `usuario_id` (que é único);
2. **Obra <-> Exemplar (1:N):** Uma obra (ex: "Dom Casmurro") pode ter vários exemplares físicos (Livro 1, Livro 2, Livro 3) na estante;
3. **Autor/Editora/Gênero <-> Obra (1:N):** Um autor escreve várias obras, mas no seu modelo, cada obra tem apenas 1 autor principal, 1 editora e 1 gênero definidos;
4. **Usuário <-> Empréstimo (1:N):** Um usuário pode fazer vários empréstimos ao longo do tempo;
5. **Exemplar <-> Empréstimo (1:N):** Um exemplar físico pode ser emprestado várias vezes (em datas diferentes), mas um registro de empréstimo refere-se a apenas um exemplar.

- **Trigger de Auditoria:** rastreia quem apagou ou alterou dados;
- **Procedures:** encapsula a lógica de negócio;
- **Views:** facilita a leitura de relatórios.

- Explicação das procedures, triggers e views criadas

1. Views

- Views: salvam consultas.

`vw_estoque_consolidado`

- **O que faz:** Cria um relatório de estoque.
- **Detalhe Técnico:** Ela agrupa as obras e conta quantos exemplares existem fisicamente (`COUNT`) por meio dos nomes e soma quantos estão marcados como 'DISPONIVEL' (usando uma lógica condicional `CASE WHEN`).
- **Utilidade:** Permite saber que você tem 3 cópias de "Harry Potter", mas apenas 2 estão na prateleira agora.

`vw_historico_completo`

- **O que faz:** Simplifica a visualização do histórico de empréstimos.
- **Detalhe Técnico:** Faz a junção (`JOIN`) de quatro tabelas (`emprestimo`, `usuario`, `exemplar`, `obra`) para transformar IDs numéricos em informação para o usuário.
- **Utilidade:** Saber rapidamente o status de todos os empréstimos realizados.

`vw_emprestimos_vencidos`

- **O que faz:** Monitora atrasos.
- **Detalhe Técnico:** Filtra apenas empréstimos com status 'PENDENTE' onde a data prevista é menor que a data atual (`CURDATE()`).
- **Utilidade:** Bloquear novos empréstimos.

2. Stored Procedures

- Procedures: encapsulam a regra de negócio.

`sp_registrar_emprestimo – coração`

1. **Verificação:** Primeiro, ela procura se existe *algum* exemplar da obra desejada com status 'DISPONIVEL'.
2. **Validação:**
    - **Se encontrar:** Insere o registro na tabela `emprestimo` (definindo a devolução para 7 dias à frente) **E** atualiza o status do exemplar para 'EMPRESTADO' imediatamente.
    - **Se não encontrar:** Dispara um erro (`SIGNAL SQLSTATE`) avisando que não há livros disponíveis, impedindo o empréstimo.

`sp_registrar_devolucao`

Gerencia o retorno do livro.

1. **Atualização do Empréstimo:** Define o status como 'DEVOLVIDO' e registra a data/hora exata da entrega (`NOW()`).
2. **Liberação do Livro:** Busca qual foi o exemplar usado e atualiza seu status de volta para 'DISPONIVEL', permitindo que outro usuário o pegue.

3. Triggers

- Triggers: funciona como um ‘gatilho’, quando disparado, pode realizar/registrar eventos.

`trg_auditoria_emprestimo_insert` **(Ao criar empréstimo)**

- **Ação:** Assim que um empréstimo é criado, ela registra o nome do usuário e o título do livro.
- **Registro:** Grava na tabela de log a ação "NOVO EMPRÉSTIMO" com os detalhes de quem fez (usuário do banco) e quando.

`trg_auditoria_emprestimo_update` **(Ao alterar empréstimo)**

- **Ação:** Detecta *qual* tipo de alteração ocorreu.
- **Lógica:**
  - Se o status mudou para 'DEVOLVIDO', ela registra como "DEVOLUÇÃO / BAIXA".
  - Se apenas a data foi alterada (renovação), ela registra como "ATUALIZAÇÃO CADASTRO" e mostra a data antiga vs. nova.
- **Utilidade:** Permite visualizar se alguém alterou uma data de entrega manualmente para evitar multa.

`trg_auditoria_emprestimo_delete` **(Ao apagar empréstimo)**

- **Ação:** Acionada se alguém rodar um `DELETE` na tabela de empréstimos.
- **Lógica:** Além de gerar o log de "REMOÇÃO DE REGISTRO", esta trigger **força o exemplar a voltar para 'DISPONIVEL'**.
- **Utilidade:** Sem isso, se você deletasse um empréstimo ativo, o livro continuaria marcado como 'EMPRESTADO' para sempre no sistema, criando um "livro fantasma" que ninguém consegue pegar.

- Relato do Desenvolvimento

Disciplina: Programação Orientada a Objetos / Base de Dados I

Projeto: Sistema de Gerenciamento de Biblioteca

Introdução

O objetivo deste trabalho foi desenvolver um sistema completo de gerenciamento de biblioteca, integrando os conceitos de modelagem de banco de dados relacional com a implementação de software utilizando a linguagem Java. O projeto exigiu a criação de um CRUD completo (Create, Read, Update, Delete), controle de transações via Stored Procedures, auditoria via Triggers e uma interface gráfica funcional.

Modelagem e Banco de Dados

A primeira etapa consistiu na estruturação do banco de dados MySQL. A normalização foi aplicada para separar as entidades de Autor, Editora e Gênero da tabela principal de Obra, garantindo a integridade dos dados e evitando redundâncias.

Durante a criação das tabelas, enfrentei um desafio técnico inesperado relacionado ao tipo de dado `YEAR` do MySQL. Inicialmente, a intenção era utilizá-lo para o campo `ano_lancamento`. No entanto, descobri que o tipo `YEAR` possui uma restrição de intervalo (1901 a 2155). Como o acervo da biblioteca poderia conter clássicos literários publicados antes de 1901 (por exemplo, obras de Machado de Assis do século XIX), o banco rejeitava essas inserções ou gerava avisos. A solução adotada foi alterar o tipo de dado para `SMALLINT`, permitindo o registro de qualquer ano histórico sem limitações de intervalo.

A implementação da auditoria foi outra etapa complexa. O requisito de registrar automaticamente as operações de empréstimo exigiu o uso de Triggers. Tive dificuldades iniciais em pegar corretamente o usuário da sessão e formatar os dados dentro da trigger. A solução envolveu a criação de uma tabela `log_auditoria` genérica e a configuração de gatilhos `AFTER INSERT` e `AFTER UPDATE` na tabela de empréstimos e obras, garantindo que nenhuma manipulação de dados passasse despercebida pelo sistema.

Desenvolvimento Back-end (Java e JDBC)

Na camada de aplicação, utilizei o padrão de projeto DAO (Data Access Object) para separar a lógica de negócios das instruções SQL.

O tratamento de erros e a validação de dados exigiram atenção especial. Um dos maiores problemas foi lidar com a duplicidade de registros, como impedir o cadastro de dois usuários com o mesmo CPF ou livros com o mesmo ISBN. Para solucionar isso, implementei tratamentos de exceção (`try-catch`) específicos para verificar violações de integridade do banco (`SQLIntegrityConstraintViolationException`) e retornar mensagens ao usuário, em vez de erros genéricos de sistema.

Além disso, a validação lógica do CPF foi um desafio à parte. Não bastava apenas limitar o campo a números; foi necessário implementar o algoritmo de verificação de dígitos e bloquear sequências repetidas (ex: 111.111.111-11), garantindo a consistência dos dados cadastrais.

Outro ponto crucial de lógica e segurança foi corrigir uma falha que permitia ao funcionário logado excluir o próprio cadastro. Durante os testes, percebi que, se um administrador excluísse sua própria conta, a aplicação entrava em um estado inconsistente. Para resolver isso, implementei uma classe de controle de sessão (`Sessao`) para armazenar o usuário autenticado. Agora, antes de qualquer operação de exclusão de funcionários, o sistema verifica se o ID escolhido corresponde ao ID da sessão ativa; caso sejam iguais, a operação é bloqueada, garantindo a integridade do acesso.

Interface Gráfica e Desafios de Frontend

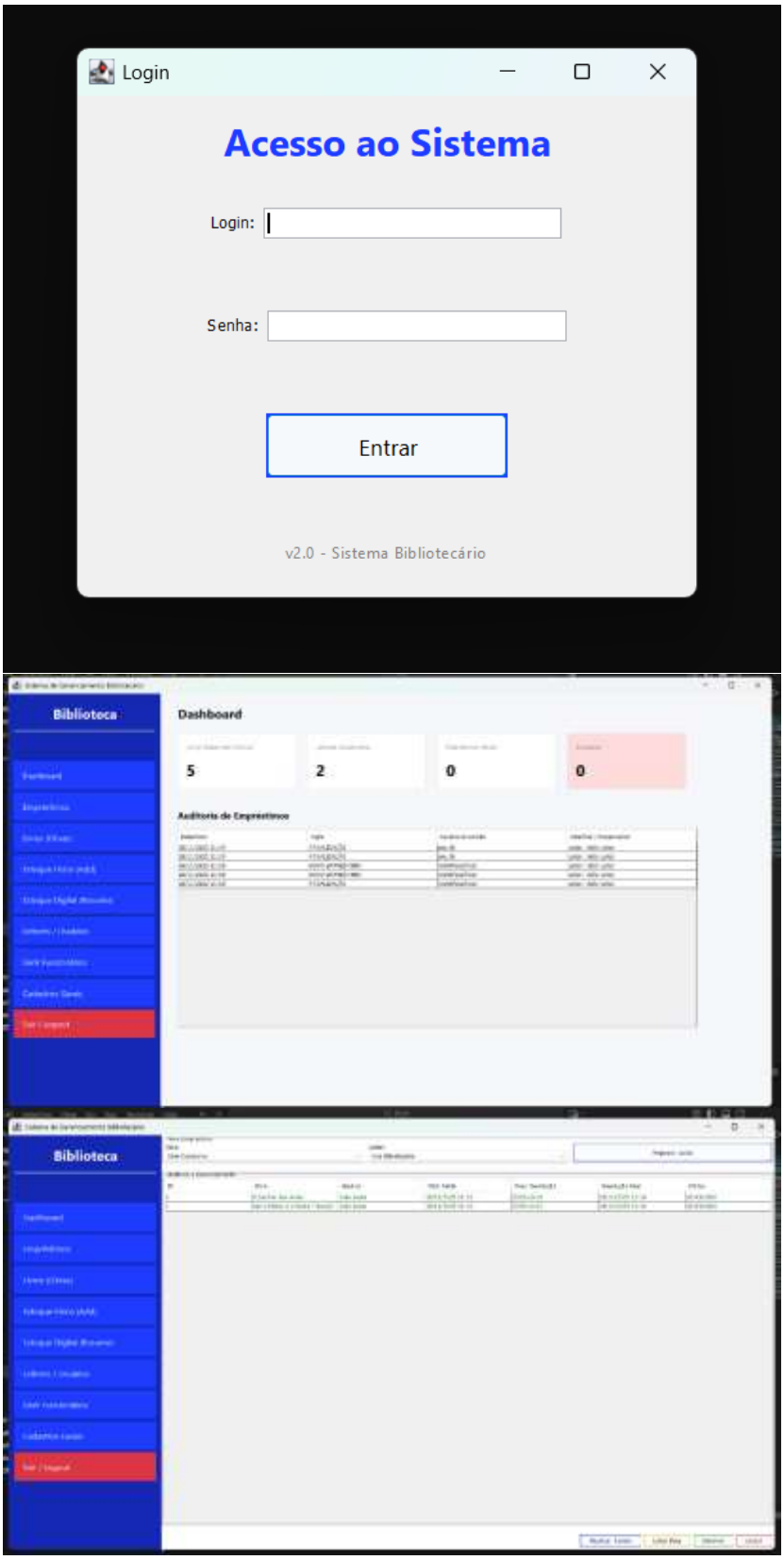
A construção da interface gráfica com a biblioteca Java Swing foi a parte mais trabalhosa do projeto. Como o Swing é algo totalmente novo, tive dificuldades em organizar os componentes para ficar esteticamente agradável.

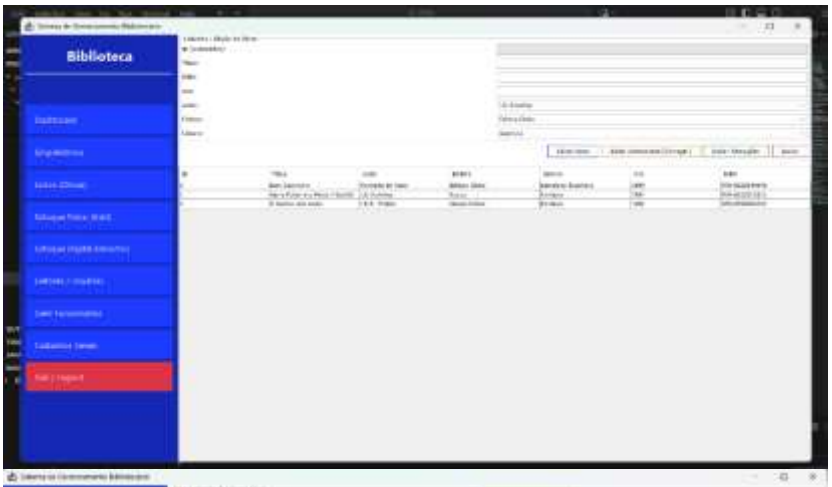
Para superar essa barreira, utilizei ferramentas de Inteligência Artificial como apoio para entender o funcionamento do `CardLayout` (para navegação entre telas sem abrir múltiplas janelas), `Combo`, `Panel` e para customizar a aparência dos componentes (cores, bordas e tabelas). Isso permitiu entregar um "Dashboard" com menu lateral e painéis dinâmicos.

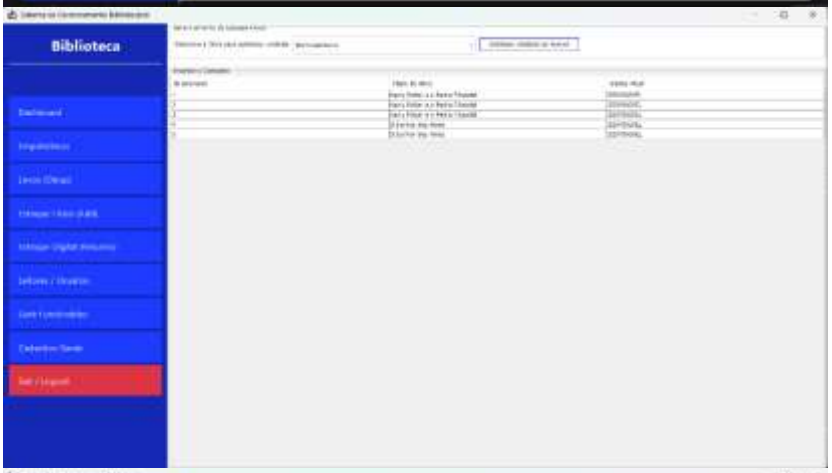
Conclusão

O desenvolvimento do projeto permitiu a aplicação prática da integração entre POO e SQL. Apesar das dificuldades técnicas — desde a escolha de tipos de dados no MySQL até a complexidade no Swing —, o resultado final foi um sistema capaz de gerenciar o acervo, controlar o fluxo (empréstimos) e auditar ações, cumprindo os requisitos propostos.

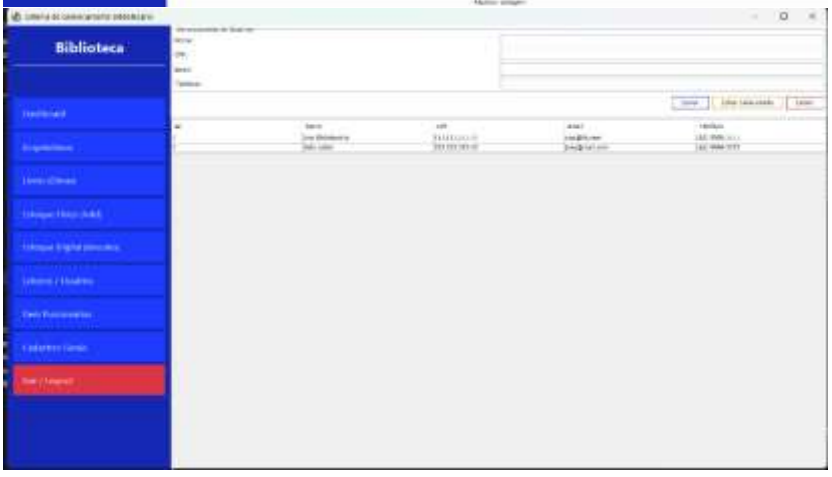
- Interface desenvolvida

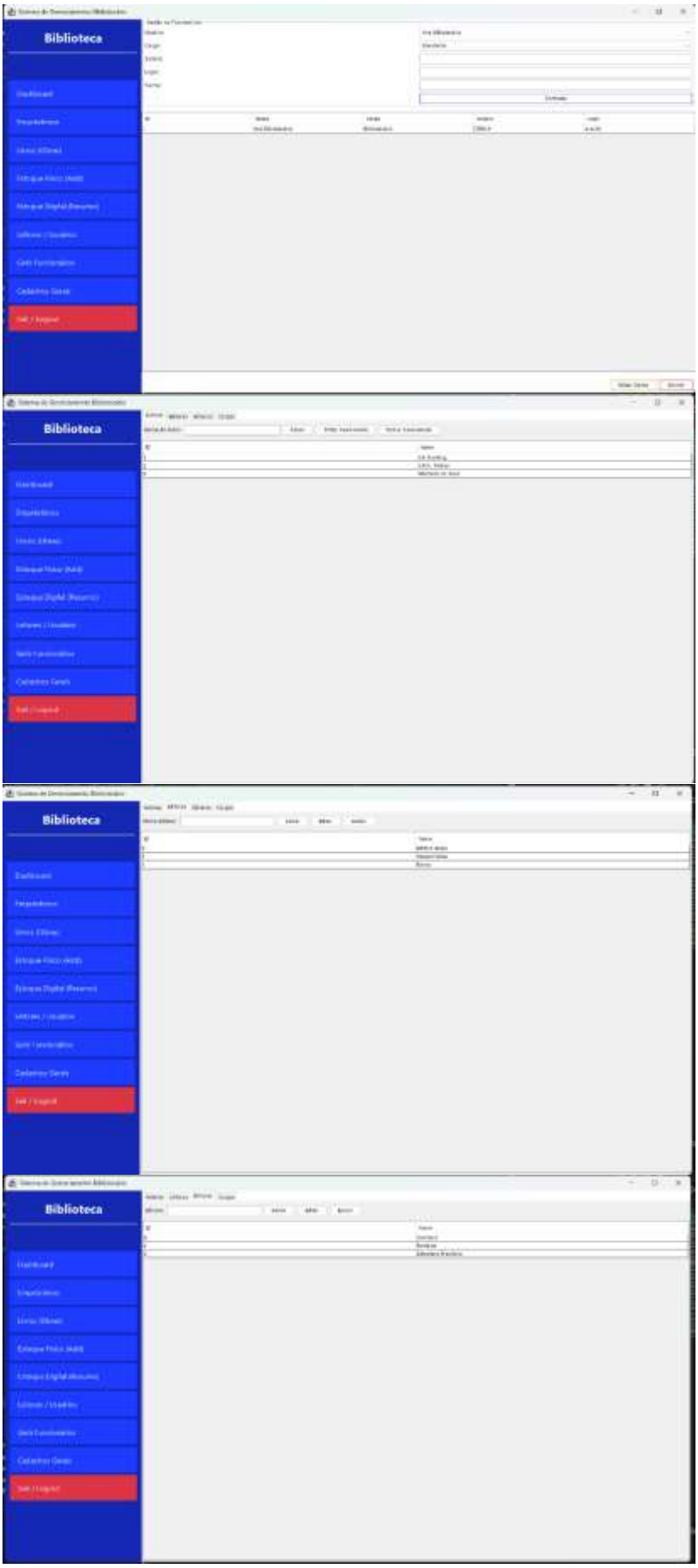


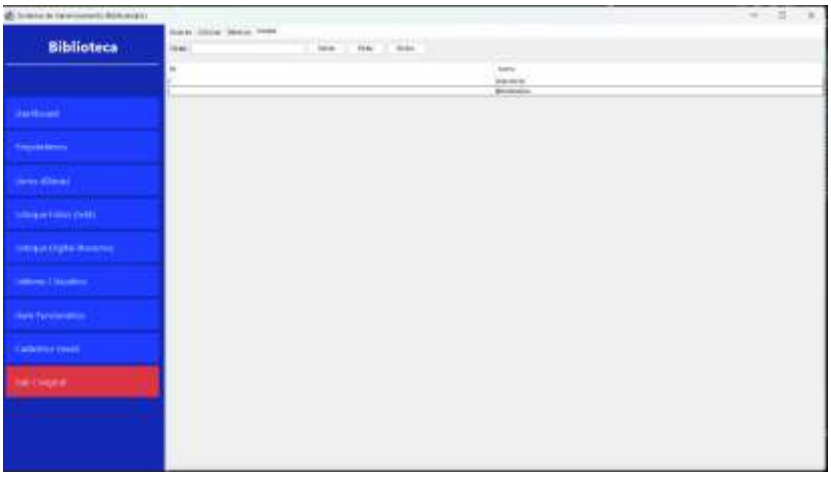

Biblioteca


Biblioteca


Biblioteca
Estoque Digital Consolidado (Resumo)


Biblioteca


Biblioteca
Biblioteca
Biblioteca
Biblioteca


Biblioteca