UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES

FASE 4 - CONSULTAS SQL

Guilherme Oliveira de Souza NºUSP: 1467138

Júlio César Cordeiro Batista NºUSP: 12684333

Kennedy Rohab Menezes da Silva NºUSP: 12683395

São Paulo

2025

Interfaces de Manutenção e Consultas do Sistema

Esta seção apresenta as interfaces desenvolvidas para manutenção de dados e realização de consultas no sistema de Gerenciamento Financeiro em Grupo. As interfaces foram implementadas utilizando Java Swing e se conectam diretamente ao banco de dados relacional por meio da camada de serviços (Service Layer), que utiliza JDBC para execução das operações de leitura e escrita.

As telas podem ser classificadas em dois tipos:
(a) Interfaces de **manutenção** (responsáveis por operações de inclusão, alteração e consulta de dados), e
(**b)** Interfaces de **consulta** voltadas à geração de relatórios e análise das informações cadastradas.

Resumo das Interfaces

Interface	**Tipo**	**Operações SQL**
LoginFrame	Consulta	SELECT em Credenciais + Cliente
CadastroFrame	Manutenção	INSERT em Cliente / Credenciais
GruposFrame	Manutenção + Consulta	INSERT/SELECT em Grupo / MembroGrupo
ConvitesFrame	Manutenção + Consulta	NSERT/UPDATE/SELECT em Convite / MembroGrupo
TransacoesFrame	Manutenção + Consulta	INSERT em Transacao, Pix, Cartao; SELECT
RelatoriosFrame	Consulta	SELECT agrupados com JOINs

1) **LoginFrame – Interface de Autenticação (Consulta)**

A tela de login é responsável por validar o acesso dos usuários ao sistema. Ela não realiza gravação de dados; apenas consultas às tabelas de **credenciais** e **clientes**, verificando a correspondência entre o e-mail informado e a senha armazenada no banco.

Embora não efetue manutenção direta dos dados, esta interface é essencial para iniciar a sessão do usuário e carregar suas informações e grupos associados.

Caminho: *main/java/gui/LoginFrame.java*

Cabeçalho: *fazerLogin()*

Serviço Chamado pela LoginFrame:

Caminho: *main/java/service/CadastroService.java*

Cabeçalho: *verificarCredenciais(email, senha)*

SQL Utilizada:

```
SELECT
   c.id,
   c.nome,
   c.cpf,
   c.data_nasc,
   c.id_plano,
   cr.email,
   cr.senha_hash
FROM Credenciais cr
JOIN Cliente c ON c.id = cr.id_cliente
WHERE cr.email = ?
        AND cr.senha_hash = ?;
```

2) **CadastroFrame – Interface de Cadastro de Clientes (Manutenção)**

Esta interface permite realizar o **cadastro de novos clientes**, bem como suas **credenciais de acesso**. Trata-se de uma tela de **manutenção**, pois executa operações de inserção (INSERT) e eventualmente atualização dos dados pessoais do usuário.

- **Criar Cliente**

Caminho: *main/java/service/CadastroService.java*

Cabeçalho: *cadastrarCliente(cliente)*

SQL Utilizada:

INSERT INTO Cliente (nome, cpf, data_nasc, id_plano) VALUES (?, ?, ?, ?);

- **Criar Credenciais**

  Caminho: *main/java/service/CadastroService.java*

  Cabeçalho: *cadastrarCredenciais(idCliente, email, senhaHash)*

  SQL Utilizada:

  INSERT INTO Credenciais (id_cliente, email, senha_hash)VALUES (?, ?, ?);

3) GruposFrame – Interface de Gerenciamento de Grupos (Manutenção + Consulta)

Gerencia grupos financeiros. Permite criar grupos, ver detalhes de participação e controlar o relacionamento Cliente–Grupo por meio da tabela **MembroGrupo**.

- **Criar Grupo**

  Caminho: *main/java/service/GrupoService.java*

  Cabeçalho: *criarGrupo(grupo)*

  SQL Utilizada:

  INSERT INTO Grupo (nome, status, descricao) VALUES (?, 'ativo', ?);

- **Listar Grupos de um Cliente**

  Caminho: *main/java/service/GrupoService.java*

  Cabeçalho: *listarGruposDoCliente(idCliente)*

  SQL Utilizada:

  ```
  SELECT g.*
  FROM Grupo g
  JOIN MembroGrupo mg ON mg.id_grupo = g.id
  WHERE mg.id_cliente = ?;
  ```

- **Adicionar Cliente ao Grupo**

  Caminho: *main/java/service/GrupoService.java*

  Cabeçalho: *inserirMembro(idCliente, idGrupo)*

SQL Utilizada:

INSERT INTO MembroGrupo (id_cliente, id_grupo, role) VALUES (?, ?, 'membro');

4) ConvitesFrame – Interface de Convites (Manutenção + Consulta)

Controla convites para entrada de novos membros nos grupos. Permite enviar convites, listar pendentes e processar aceitação/recusa.

- **Enviar Convite**

  Caminho: *main/java/service/ConviteService.java*

  Cabeçalho: *enviarConvite(idRemetente, idDestino, idGrupo)*

  SQL Utilizada:

INSERT INTO Convite (id_remetente, id_destino, id_grupo, status) VALUES (?, ?, ?, 'pendente');

- **Listar Convites Recebidos**

  Caminho: *main/java/service/ConviteService.java*

  Cabeçalho: *listarConvitesRecebidos(idCliente)*

  SQL Utilizada:

  *SELECT * FROM Convite WHERE id_destino = ?;*

- **Aceitar Convite**

  Caminho: *main/java/service/ConviteService.java*

  Cabeçalho: *aceitarConvite(idConvite)*

  SQL Utilizada:

  (Atualiza o status do convite):

  UPDATE ConviteSET status = 'aceito' WHERE id = ?;

  (Inclui o membro no grupo):

  INSERT INTO MembroGrupo (id_cliente, id_grupo)
  SELECT id_destino, id_grupo
  FROM Convite WHERE id = ?;

5) TransacoesFrame – Interface de Transações (Manutenção + Consulta)

Gerencia as transações financeiras registradas pelos membros dos grupos.Suporta lançamentos normais e especializados (Pix ou Cartão).

- **Criar Transação**

  Caminho: *main/java/service/TransacaoService.java*

  Cabeçalho: *criarTransacao(transacao)*

  SQL Utilizada:

  ```
  INSERT INTO Transacao
  (descricao, valor, id_cliente, id_grupo, id_categoria)
  VALUES (?, ?, ?, ?, ?);
  ```

- **Transação Pix**

  Caminho: *main/java/service/TransacaoService.java*

  Cabeçalho: *criarTransacaoPix(transacaoPix)*

  SQL Utilizada:

  ```
  INSERT INTO Pix (id_transacao, chave) VALUES (?, ?);
  ```

- **Transação Cartão**

  Caminho: *main/java/service/TransacaoService.java*

  Cabeçalho: *criarTransacaoCartao(transacaoCartao)*

  SQL Utilizada:

  ```
  INSERT INTO Cartao
  (id_transacao, bandeira, digitos_finais)
  VALUES (?, ?, ?);
  ```

- **Listar Transações do Grupo**

  Caminho: main/java/service/TransacaoService.java

  Cabeçalho: listarTransacoesDoGrupo(idGrupo)

SQL Utilizada:

```
SELECT *
FROM Transacao
WHERE id_grupo = ?
ORDER BY data_transacao DESC;
```

6) RelatoriosFrame – Interface de Relatórios (Consulta)

Disponibiliza consultas analíticas para apoiar decisões financeiras. Inclui totais por membro, categorias de gasto e ranking de maiores despesas.

- **Relatório: Maiores Gastos**

Caminho: main/java/service/RelatorioService.java

Cabeçalho: relatorioMaioresGastos(idGrupo)

SQL Utilizada:

```
SELECT c.nome, SUM(t.valor) AS total
FROM Transacao t
JOIN Cliente c ON c.id = t.id_cliente
WHERE t.id_grupo = ?
GROUP BY c.nome
ORDER BY total ASC;
```

- **Relatório: Gastos por Categoria**

Caminho: *main/java/service/RelatorioService.java*

Cabeçalho: *relatorioGastosPorCategoria(idGrupo)*

SQL Utilizada:

```
SELECT cat.nome, SUM(t.valor) AS total
FROM Transacao t
JOIN Categoria cat ON cat.id = t.id_categoria
WHERE t.id_grupo = ?
GROUP BY cat.nome
ORDER BY total ASC;
```

- **Relatório: Contribuição por Participante**

Caminho: *main/java/service/RelatorioService.java*

Cabeçalho: *relatorioPorParticipante(idGrupo)*

SQL Utilizada:

```
SELECT c.nome, SUM(t.valor) AS total
FROM Transacao t
JOIN Cliente c ON c.id = t.id_cliente
WHERE t.id_grupo = ?
GROUP BY c.nome
ORDER BY total DESC;
```