UC Programação de Soluções Computacionais (2024.1)

Exemplo de um projeto prático - Tema: Agenda de Contatos

OBS.: Seja criativo na sua implementação. NÃO COPIE, POR COPIAR.

Objetivo: desenvolver uma aplicação desktop com banco de dados para sua Agenda de Contatos.

2ª. ETAPA

Prosseguindo o desenvolvimento do projeto prático (A3), devemos relembrar e melhorar alguns pontos:

1. Atualize o driver da classe **ConexaoMySQL**:

```
//Carregando o JDBC Driver padrão
String driverName = "com.mysql.cj.jdbc.Driver";
```

2. Para o cadastro, seria bom ao criar a tabela usar a opção de autoincremento **(opção AI)** do campo **id**. Facilitando o controle dele durante a inserção de cada objeto.

```
CREATE TABLE `contatos` (
  `id` int NOT NULL AUTO_INCREMENT,
  `nome` varchar(45) DEFAULT NULL,
  `telefone` varchar(20) DEFAULT NULL,
  `celular` varchar(20) DEFAULT NULL,
  `email` varchar(80) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

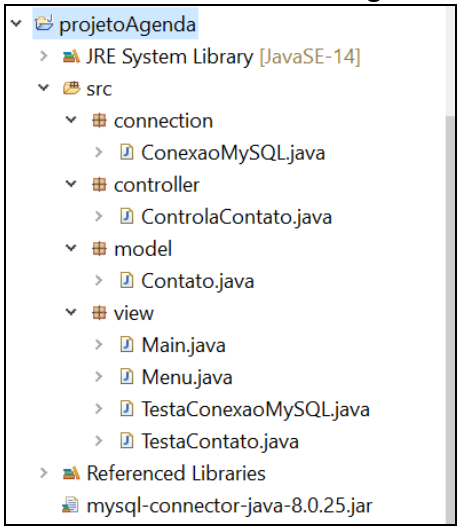
Os campos da tabela contatos ficariam assim:

Table: contatos

Columns:

id	int AI PK
nome	varchar(45)
telefone	varchar(20)
celular	varchar(20)
email	varchar(80)

As pastas e arquivos do projeto estão divididas como mostra a figura:



PASSO 1) CONTINUANDO O DESENVOLVIMENTO DA CLASSE MENU

Prosseguindo com a implementação da classe Menu.java, irei fazer uma **pequena alteração no menu de opções do projeto exemplo**, para que eu possa te ajudar no desenvolvimento da 2ª etapa do Projeto Prático A3.

Novo Menu:

1. Adicionar um contato no banco
2. Atualizar um contato no banco
3. Remover um contato do banco
4. Buscar e Alterar o <Celular> de um contato
5. Listar todos ordenados pelo nome
6. Busca e Lista todos por parte do nome
7. Relatório (com todos os dados referentes ao tema do grupo)
8. Fechar / Sair do programa

As opções marcadas já foram desenvolvidas na 1ª etapa.

Na pasta padrão, **src**, acesse o pacote **controller**. Em seguida, acesse a classe **ControlaContato.java** que irá se responsabilizar pelo CRUD no banco.

Relembrando devemos usar o **java.sql.PreparedStatement** em todos os tipos de comandos SQL (INSER, UPDATE, DELETE, SELECT), pois é mais performático que **java.sql.Statement**.

Continue desenvolvendo uma interface intuitiva e amigável para os usuários inserirem e visualizarem os dados.

PASSO 2) CLASSE MENU - OPÇÕES DO MENU: remoção, busca com alteração e listagem de todos

Para as opções <3>, <4> e <5> do Menu, você deve criar os métodos removerContato(), alterarContato() e listarTodos() para respectivamente, remover `"DELETE FROM contatos WHERE id = ?"`, alterar o número do celular de um contato `"UPDATE contatos SET celular = ? WHERE id = ?"` e `"SELECT * FROM contatos ORDER BY nome ASC"` para listar todos ordenados pelo nome. São eles:

```
	//remove contato no banco
	public void removerContato(Contato c) {

		Connection conn = ConexaoMySQL.getInstance();

		try {
			String sql = "DELETE FROM contatos where id = ?";
			PreparedStatement stmt = conn.prepareStatement(sql);
			stmt.setInt(1, c.getId());
			stmt.execute();
			stmt.close();
			conn.close();
		} catch (SQLException e) {
			throw new RuntimeException(e);
		}

		JOptionPane.showMessageDialog(null, "Remoção realizada com sucesso!");
	}
```

Para uma boa usabilidade, confirme com o usuário se ele realmente deseja remover o contato da Agenda.

E, para alterar o celular seria bom verificar se o usuário confirma essa ação, antes de fazê-la.

```
	//4. busca e altera o <Celular> de um contato
	public void alterarContato(Contato c) {
		Connection conn = ConexaoMySQL.getInstance();

		try {
			String sql = "UPDATE contatos SET nome = ?, telefone = ?, celular = ?,
email = ? WHERE id = ?";
			PreparedStatement stmt = conn.prepareStatement(sql);
			stmt.setString(1, c.getNome());
			stmt.setString(2, c.getTelefone());
			stmt.setString(3, c.getCelular());
			stmt.setString(4, c.getEmail());
			stmt.setInt(5, c.getId());
			stmt.execute();
			stmt.close();
			conn.close();
		} catch (SQLException e) {
			throw new RuntimeException(e);
		}
		JOptionPane.showMessageDialog(null, "Celular modificado com sucesso!");
	}
```

Para que possamos alterar apenas o celular de um contato, precisamos buscar a posição deste contato **(Contato c)** na lista de contatos. Então segue uma parte do Menu que chama o método estático ***buscaSequencial()***.

```
case 4 :
	System.out.println("Buscar e Alterar o <Celular> de um contato");
	List<Contato> listaal = cc.listarTodos();

	List<Contato> listaCont = new ArrayList<>();
	//impressão de todos os contatos para escolha de um a ser alterado
	for(Contato c: listaal) {
		System.out.printf("%s",c.toString());
		listaCont.add(c); //conversão da lista de dados do banco em um array de contatos
	}
	System.out.println("Qual contato alterar?\nDigite o ID do contato:");
	int idBuscado = ler.nextInt();
	ler.nextLine(); //limpeza do buffer

	//busca sequencial para encontrar a posição do ID do celular a ser alterado
	int pos = buscaSequencial(listaCont, idBuscado, ler);
	if (pos != -1)
		cc.alterarContato(listaCont.get(pos));
	else
		System.out.println("Id não encontrado.");
	break;
```

Quanto a implementação do método estático ***buscaSequencial()*** temos:

```
//busca sequencial para encontrar a posição do ID do celular a ser alterado
public static int buscaSequencial(List<Contato> listaCont, int idBuscado, Scanner ler) {
	int pos=-1; //não existe contato com o idBuscado

	for (int i = 0; i < listaCont.size(); i++) {

	    if (idBuscado == listaCont.get(i).getId()) {
			System.out.println("Entre com o novo número de celular do contato: ");
			listaCont.get(i).setCelular(ler.nextLine());
			System.out.printf("%s",listaCont.get(i).toString());
			pos = i;
		}
```

```
        }
        return pos;
}
```

Quanto a listar todos ordenados pelo nome, basta:

```
        //5. lista todos os contatos
        public List<Contato> listarTodos(){

                Connection conn = ConexaoMySQL.getInstance();

                List<Contato> lista = new ArrayList<>();
                try {
                        String sql = "SELECT * FROM contatos ORDER BY nome ASC";
                        Statement stmt = conn.createStatement();
                        ResultSet resultados = stmt.executeQuery(sql);
                        while(resultados.next()) {
                                int id = resultados.getInt("id");
                                String nome = resultados.getString("nome");
                                String telefone = resultados.getString("telefone");
                                String celular = resultados.getString("celular");
                                String email = resultados.getString("email");
                                lista.add(new Contato(id, nome, telefone, celular, email));
                        }
                        stmt.close();
                        conn.close();
                } catch (SQLException e) {
                                throw new RuntimeException(e);
                }
                return lista;
        }
```

Nosso Menu está assim, com as opções marcadas já desenvolvidas:

Menu:

1. Adicionar um contato no banco
2. Atualizar um contato no banco
3. Remover um contato do banco
4. Buscar e Alterar o <Celular> de um contato
5. Listar todos ordenados pelo nome
6. Busca e Lista todos por parte do nome
7. Relatório (com todos os dados referentes ao tema do grupo)
8. Fechar / Sair do programa

3ª. ETAPA

PASSO 3) CLASSE MENU - OPÇÕES DO MENU: busca e lista todos por parte nome e um relatório com todos os dados referentes ao tema do grupo

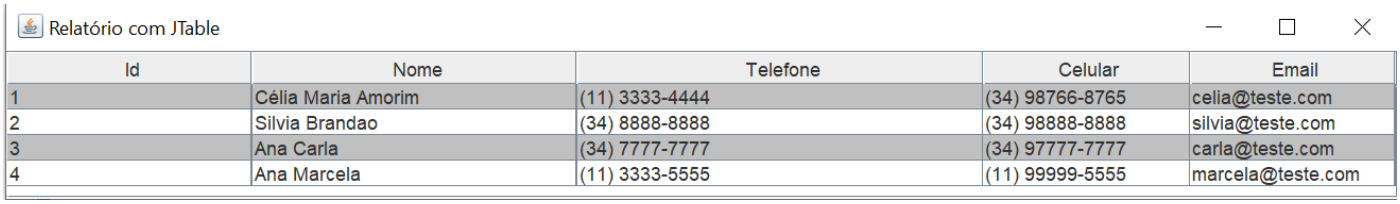
Para as opções <6>, <7> e <8> do Menu, você deve criar os métodos `listarPorParteDoNome()` e `relatorio()` para respectivamente, buscar e listar todos por parte nome `"SELECT * FROM contatos WHERE nome like ?"` e apresentar um relatório sobre todos contatos do banco `"SELECT * FROM contatos"`. A opção <8> encerra o código. São eles:

```
        //6. busca e lista contato por parte do nome
        public List<Contato> listarPorParteDoNome(String pnome){

                Connection conn = ConexaoMySQL.getInstance();
```

```
		List<Contato> lista = new ArrayList<>();
		try {
			String sql = "SELECT * FROM contatos WHERE nome like ?";
			PreparedStatement stmt = conn.prepareStatement(sql);
			stmt.setString(1, "%" + pnome + "%");
			ResultSet resultados = stmt.executeQuery();
			while(resultados.next()) {
				int id = resultados.getInt("id");
				String nome = resultados.getString("nome");
				String telefone = resultados.getString("telefone");
				String celular = resultados.getString("celular");
				String email = resultados.getString("email");
				lista.add(new Contato(id, nome, telefone, celular, email));
			}
			stmt.close();
			conn.close();
		} catch (SQLException e) {
			throw new RuntimeException(e);
		}
		return lista;
	}

	//7. Relatório Geral
	public void relatorio() {

		Connection conn = ConexaoMySQL.getInstance();

		List<Contato> lista = new ArrayList<>();
		try {
			String sql = "SELECT * FROM contatos";
			Statement stmt = conn.createStatement();
			ResultSet resultados = stmt.executeQuery(sql);

			while(resultados.next()) {
				int id = resultados.getInt("id");
				String nome = resultados.getString("nome");
				String telefone = resultados.getString("telefone");
				String celular = resultados.getString("celular");
				String email = resultados.getString("email");
				lista.add(new Contato(id, nome, telefone, celular, email));
			}
			stmt.close();
			conn.close();
		} catch (SQLException e) {
				throw new RuntimeException(e);
		}

		RelatorioJTable r = new RelatorioJTable();
		r.criarRelatorioJTable(lista);
	}
```

No método relatório chamaremos o método de instância criarRelatorioJTable() da classe **RelatorioJTable.Java**, pacote View.

De posse da lista de contatos você poderá criar um relatório usando os comandos de impressão em console ou fazer a conversão da lista de contatos para um array de dados e, assim fazer a criação da JTable com o array de dados. Não usei nenhum filtro.



Veja o artigo: **JTable: Utilizando o componente em interfaces gráficas Swing** no link https://www.devmedia.com.br/jtable-utilizando-o-componente-em-interfaces-graficas-swing/28857

PRONTO! Agora, altere as classes do seu projeto e a classe de teste de modo a implementar as funcionalidades propostas (inserir, buscar, relatório, remover e alterar) na codificação do tema de seu projeto.