



Estácio

Relatório Discente de Acompanhamento

Missão Prática / Nível 3 / Mundo 3

Campus: Rua Visconde de Mauá, Nº 150, Sala 105, - Centro - Canela - RS - CEP.: 95.680-232

Curso: Desenvolvimento Full Stack

Disciplina: Nível 3: Back-end Sem Banco Não Tem

Número da Turma: 9001

Semestre Letivo: 2024.1

Nome: Endrius da Silva dos Santos

Repositório GitHub: <https://github.com/endriusssantos/atividade-nivel-3-mundo-3>

Objetivo da Prática

O objetivo da prática é implementar a persistência de dados usando o middleware JDBC, utilizar o padrão DAO para manuseio de dados, realizar o mapeamento objeto-relacional em sistemas Java, e criar sistemas cadastrais com persistência em banco de dados relacional, finalizando com a criação de um aplicativo cadastral que utiliza o SQL Server para persistência de dados.

Procedimentos

Implementação das Classes

```
package cadastrobd.model;
```

```
public class Pessoa {
```

```
    private Integer id;
```

```
    private String nome;
```

```
    private String logradouro;
```

```
    private String cidade;
```

```
    private String estado;
```

```
    private String telefone;
```

```
    private String email;
```

```
    private char tipoPessoa;
```

```
    // Construtor
```

```
    public Pessoa(Integer id, String nome, String logradouro, String cidade, String estado, String telefone, String email, char tipoPessoa) {
```

```
        this.id = id;
```

```
        this.nome = nome;
```

```
        this.logradouro = logradouro;
```

```
        this.cidade = cidade;
```

```
        this.estado = estado;
```

```
        this.telefone = telefone;
```

```
        this.email = email;
```

```
        this.tipoPessoa = tipoPessoa;
```

```
    }
```

```
// Método exibir
```

```
public void exibir() {  
    System.out.println("ID: " + id);  
    System.out.println("Nome: " + nome);  
    System.out.println("Logradouro: " + logradouro);  
    System.out.println("Cidade: " + cidade);  
    System.out.println("Estado: " + estado);  
    System.out.println("Telefone: " + telefone);  
    System.out.println("Email: " + email);  
    String tipo = (tipoPessoa == 'F') ? "Pessoa Física" : "Pessoa Jurídica";  
    System.out.println("Tipo: " + tipo);  
}
```

```
// Getters e setters
```

```
public Integer getId() {  
    return id;  
}
```

```
public void setId(Integer id) {  
    this.id = id;  
}
```

```
public String getNome() {  
    return nome;  
}
```

```
public void setNome(String nome) {  
    this.nome = nome;  
}
```

```
public String getLogradouro() {
```

```
        return logradouro;
    }

    public void setLogradouro(String logradouro) {
        this.logradouro = logradouro;
    }

    public String getCidade() {
        return cidade;
    }

    public void setCidade(String cidade) {
        this.cidade = cidade;
    }

    public String getEstado() {
        return estado;
    }

    public void setEstado(String estado) {
        this.estado = estado;
    }

    public String getTelefone() {
        return telefone;
    }

    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }

    public String getEmail() {
```

```

        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public char getTipoPessoa() {
        return tipoPessoa;
    }

    public void setTipoPessoa(char tipoPessoa) {
        this.tipoPessoa = tipoPessoa;
    }
}

package cadastrobd.model;

public class PessoaFisica extends Pessoa {
    private String cpf;

    // Construtor de PessoaFisica
    public PessoaFisica(Integer id, String nome, String logradouro, String cidade, String estado, String telefone, String email, String cpf) {
        super(id, nome, logradouro, cidade, estado, telefone, email, 'F'); // Chama o construtor da superclasse com 'F' como tipoPessoa
        this.cpf = cpf;
    }

    // Getter e setter
    public String getCpf() {
        return cpf;
    }
}

```

```

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    // Método toString para incluir CPF
    @Override
    public String toString() {
        return super.toString() + ", CPF: " + cpf;
    }
}

package cadastrobd.model;

public class PessoaJuridica extends Pessoa {
    private String cnpj;

    // Construtor de PessoaJuridica
    public PessoaJuridica(Integer id, String nome, String logradouro, String cidade,
        String estado, String telefone, String email, String cnpj) {
        super(id, nome, logradouro, cidade, estado, telefone, email, 'J'); // Chama o
        construtor da superclasse com 'J' como tipoPessoa
        this.cnpj = cnpj;
    }

    // Getter e setter
    public String getCnpj() {
        return cnpj;
    }

    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }
}

```

```

// Método toString para incluir CNPJ
@Override
public String toString() {
    return super.toString() + ", CNPJ: " + cnpj;
}
}

```

```

import cadastro.model.PessoaFisicaDAO;
import cadastro.model.PessoaJuridicaDAO;
import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaJuridica;
import cadastro.model.util.ConectorBD;

```

```

import java.sql.Connection;
import java.sql.SQLException;
import java.util.List;
import java.util.Scanner;

```

```

public class CadastroBDTeste {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Connection conexao = ConectorBD.getConnection();
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(conexao);
        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO(conexao);

        int escolha;

        do {
            System.out.println("=====");
            System.out.println("1 - Incluir Pessoa");

```

```

System.out.println("2 - Alterar Pessoa");
System.out.println("3 - Excluir Pessoa");
System.out.println("4 - Buscar pelo Id");
System.out.println("5 - Exibir Todos");
System.out.println("6 - Exibir Somente Pessoas Físicas");
System.out.println("7 - Exibir Somente Pessoas Jurídicas");
System.out.println("0 - Finalizar Programa");
System.out.println("=====");
escolha = scanner.nextInt();
scanner.nextLine();

try {
    switch (escolha) {
        case 1:
            System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
            char tipoInclusao = scanner.next().charAt(0);
            scanner.nextLine();

            if (tipoInclusao == 'F' || tipoInclusao == 'f') {
                cadastrarPessoaFisica(pessoaFisicaDAO, scanner);
            } else if (tipoInclusao == 'J' || tipoInclusao == 'j') {
                cadastrarPessoaJuridica(pessoaJuridicaDAO, scanner);
            } else {
                System.out.println("Opção inválida.");
            }
            break;

        case 2:
            alterarPessoa(pessoaFisicaDAO, pessoaJuridicaDAO, scanner);
            break;

        case 3:

```



```

        excluirPessoa(pessoaFisicaDAO, pessoaJuridicaDAO, scanner);
        break;

    case 4:
        buscarPessoaPeloid(pessoaFisicaDAO, pessoaJuridicaDAO, scanner);
        break;

    case 5:
        exibirTodasPessoas(pessoaFisicaDAO, pessoaJuridicaDAO);
        break;

    case 6:
        exibirPessoasFisicas(pessoaFisicaDAO);
        break;

    case 7:
        exibirPessoasJuridicas(pessoaJuridicaDAO);
        break;

    case 0:
        System.out.println("Encerrando o programa.");
        break;

    default:
        System.out.println("Opção inválida. Tente novamente.");
    }
} catch (SQLException e) {
    System.out.println("Erro de banco de dados: " + e.getMessage());
    e.printStackTrace();
}
} while (escolha != 0);
}

```

// Métods para cadastrar Pessoa Fisica

```

private static void cadastrarPessoaFisica(PessoaFisicaDAO pessoaFisicaDAO,
Scanner scanner) throws SQLException {
    System.out.println("Digite o nome da Pessoa Física:");

```

```

String nome = scanner.nextLine();
System.out.println("Digite o logradouro:");
String logradouro = scanner.nextLine();
System.out.println("Digite a cidade:");
String cidade = scanner.nextLine();
System.out.println("Digite o estado:");
String estado = scanner.nextLine();
System.out.println("Digite o telefone:");
String telefone = scanner.nextLine();
System.out.println("Digite o email:");
String email = scanner.nextLine();
System.out.println("Digite o CPF:");
String cpf = scanner.nextLine();

PessoaFisica novaPessoaFisica = new PessoaFisica(0, nome, logradouro,
cidade, estado, telefone, email, cpf);
    pessoaFisicaDAO.inserirPessoaFisica(novaPessoaFisica);
    System.out.println("Pessoa Física cadastrada com sucesso.");
}

// Métods para cadastrar Pessoa Juridica
private static void cadastrarPessoaJuridica(PessoaJuridicaDAO
pessoaJuridicaDAO, Scanner scanner) throws SQLException {
    System.out.println("Digite o nome da Pessoa Jurídica:");
    String nome = scanner.nextLine();
    System.out.println("Digite o logradouro:");
    String logradouro = scanner.nextLine();
    System.out.println("Digite a cidade:");
    String cidade = scanner.nextLine();
    System.out.println("Digite o estado:");
    String estado = scanner.nextLine();
    System.out.println("Digite o telefone:");
    String telefone = scanner.nextLine();

```

```

System.out.println("Digite o email:");
String email = scanner.nextLine();
System.out.println("Digite o CNPJ:");
String cnpj = scanner.nextLine();

PessoaJuridica novaPessoaJuridica = new PessoaJuridica(0, nome, logradouro,
cidade, estado, telefone, email, cnpj);
pessoaJuridicaDAO.incluir(novaPessoaJuridica);
System.out.println("Pessoa Jurídica cadastrada com sucesso.");
}

// Métodos para alterar Pessoas

private static void alterarPessoa(PessoaFisicaDAO pessoaFisicaDAO,
PessoaJuridicaDAO pessoaJuridicaDAO, Scanner scanner) throws SQLException {
    System.out.println("F - Alterar Pessoa Física | J - Alterar Pessoa Jurídica");
    char tipoPessoa = scanner.next().charAt(0);
    scanner.nextLine();

    if (tipoPessoa == 'F' || tipoPessoa == 'f') {
        System.out.println("Digite o ID da Pessoa Física que deseja alterar:");
        int id = scanner.nextInt();
        scanner.nextLine();

        PessoaFisica pessoaExistente = pessoaFisicaDAO.getPessoa(id);

        if (pessoaExistente != null) {
            System.out.println("Digite o novo nome da Pessoa Física:");
            String novoNome = scanner.nextLine();
            System.out.println("Digite o novo logradouro:");
            String novoLogradouro = scanner.nextLine();
            System.out.println("Digite a nova cidade:");
            String novaCidade = scanner.nextLine();
            System.out.println("Digite o novo estado:");

```

```
String novoEstado = scanner.nextLine();
System.out.println("Digite o novo telefone:");
String novoTelefone = scanner.nextLine();
System.out.println("Digite o novo email:");
String novoEmail = scanner.nextLine();
System.out.println("Digite o novo CPF:");
String novoCpf = scanner.nextLine();
```

```
PessoaFisica novaPessoa = new PessoaFisica(id, novoNome,
novoLogradouro, novaCidade, novoEstado, novoTelefone, novoEmail, novoCpf);
```

```
    pessoaFisicaDAO.alterar(novaPessoa);
```

```
    System.out.println("Pessoa Física atualizada com sucesso.");
```

```
    } else {
```

```
        System.out.println("Pessoa Física não encontrada.");
```

```
    }
```

```
    } else if (tipoPessoa == 'J' || tipoPessoa == 'j') {
```

```
        System.out.println("Digite o ID da Pessoa Jurídica que deseja alterar:");
```

```
        int id = scanner.nextInt();
```

```
        scanner.nextLine();
```

```
PessoaJuridica pessoaExistente = pessoaJuridicaDAO.getPessoa(id);
```

```
if (pessoaExistente != null) {
```

```
    System.out.println("Digite o novo nome da Pessoa Jurídica:");
```

```
    String novoNome = scanner.nextLine();
```

```
    System.out.println("Digite o novo logradouro:");
```

```
    String novoLogradouro = scanner.nextLine();
```

```
    System.out.println("Digite a nova cidade:");
```

```
    String novaCidade = scanner.nextLine();
```

```
    System.out.println("Digite o novo estado:");
```

```
    String novoEstado = scanner.nextLine();
```

```
    System.out.println("Digite o novo telefone:");
```

```
    String novoTelefone = scanner.nextLine();
```

```

        System.out.println("Digite o novo email:");
        String novoEmail = scanner.nextLine();
        System.out.println("Digite o novo CNPJ:");
        String novoCnpj = scanner.nextLine();

        PessoaJuridica novaPessoa = new PessoaJuridica(id, novoNome,
        novoLogradouro, novaCidade, novoEstado, novoTelefone, novoEmail, novoCnpj);

        pessoaJuridicaDAO.alterar(novaPessoa);

        System.out.println("Pessoa Jurídica atualizada com sucesso.");
    } else {
        System.out.println("Pessoa Jurídica não encontrada.");
    }
} else {
    System.out.println("Opção inválida.");
}
}

// Métods para excluir Pessoas

private static void excluirPessoa(PessoaFisicaDAO pessoaFisicaDAO,
PessoaJuridicaDAO pessoaJuridicaDAO, Scanner scanner) throws SQLException {
    System.out.println("F - Excluir Pessoa Física | J - Excluir Pessoa Jurídica");
    char tipoPessoa = scanner.next().charAt(0);
    scanner.nextLine();

    if (tipoPessoa == 'F' || tipoPessoa == 'f') {
        System.out.println("Digite o ID da Pessoa Física a ser excluída:");
        int id = scanner.nextInt();
        pessoaFisicaDAO.excluir(id);
        System.out.println("Pessoa Física excluída com sucesso.");
    } else if (tipoPessoa == 'J' || tipoPessoa == 'j') {
        System.out.println("Digite o ID da Pessoa Jurídica a ser excluída:");
        int id = scanner.nextInt();
        pessoaJuridicaDAO.excluir(id);
    }
}

```

```

        System.out.println("Pessoa Jurídica excluída com sucesso.");
    } else {
        System.out.println("Opção inválida.");
    }
}

```

// Métodos para buscar Pessoas

```

private static void buscarPessoaPeloid(PessoaFisicaDAO pessoaFisicaDAO,
PessoaJuridicaDAO pessoaJuridicaDAO, Scanner scanner) throws SQLException {

```

```

    System.out.println("F - Buscar Pessoa Física | J - Buscar Pessoa Jurídica");
    char tipoPessoa = scanner.next().charAt(0);
    scanner.nextLine();

```

```

    if (tipoPessoa == 'F' || tipoPessoa == 'f') {
        System.out.println("Digite o ID da Pessoa Física:");
        int id = scanner.nextInt();
        scanner.nextLine();

```

```

        PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id);

```

```

        if (pessoaFisica != null) {
            System.out.println("Detalhes da Pessoa Física:");
            System.out.println("ID: " + pessoaFisica.getId());
            System.out.println("Nome: " + pessoaFisica.getNome());
            System.out.println("Logradouro: " + pessoaFisica.getLogradouro());
            System.out.println("Cidade: " + pessoaFisica.getCidade());
            System.out.println("Estado: " + pessoaFisica.getEstado());
            System.out.println("Telefone: " + pessoaFisica.getTelefone());
            System.out.println("Email: " + pessoaFisica.getEmail());
            System.out.println("CPF: " + pessoaFisica.getCpf());
        } else {
            System.out.println("Pessoa Física não encontrada.");
        }
    }
}

```

```

} else if (tipoPessoa == 'J' || tipoPessoa == 'j') {
    System.out.println("Digite o ID da Pessoa Jurídica:");
    int id = scanner.nextInt();
    scanner.nextLine();

    PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);

    if (pessoaJuridica != null) {
        System.out.println("Detalhes da Pessoa Jurídica:");
        System.out.println("ID: " + pessoaJuridica.getId());
        System.out.println("Nome: " + pessoaJuridica.getNome());
        System.out.println("Logradouro: " + pessoaJuridica.getLogradouro());
        System.out.println("Cidade: " + pessoaJuridica.getCidade());
        System.out.println("Estado: " + pessoaJuridica.getEstado());
        System.out.println("Telefone: " + pessoaJuridica.getTelefone());
        System.out.println("Email: " + pessoaJuridica.getEmail());
        System.out.println("CNPJ: " + pessoaJuridica.getCnpj());
    } else {
        System.out.println("Pessoa Jurídica não encontrada.");
    }
} else {
    System.out.println("Opção inválida.");
}
}

// Métodos para exibir todas as Pessoas

private static void exibirTodasPessoas(PessoaFisicaDAO pessoaFisicaDAO,
PessoaJuridicaDAO pessoaJuridicaDAO) throws SQLException {
    System.out.println("Pessoas Físicas Cadastradas:");
    List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();
    if (pessoasFisicas.isEmpty()) {
        System.out.println("Nenhuma Pessoa Física cadastrada.");
    } else {

```

```

for (PessoaFisica pf : pessoasFisicas) {
    System.out.println("ID: " + pf.getId());
    System.out.println("Nome: " + pf.getNome());
    System.out.println("Logradouro: " + pf.getLogradouro());
    System.out.println("Cidade: " + pf.getCidade());
    System.out.println("Estado: " + pf.getEstado());
    System.out.println("Telefone: " + pf.getTelefone());
    System.out.println("Email: " + pf.getEmail());
    System.out.println("CPF: " + pf.getCpf());
    System.out.println();
}
}

```

```

System.out.println("Pessoas Jurídicas Cadastradas:");

List<PessoaJuridica> pessoasJuridicas =
pessoaJuridicaDAO.getPessoasJuridicas();

if (pessoasJuridicas.isEmpty()) {
    System.out.println("Nenhuma Pessoa Jurídica cadastrada.");
} else {
    for (PessoaJuridica pj : pessoasJuridicas) {
        System.out.println("ID: " + pj.getId());
        System.out.println("Nome: " + pj.getNome());
        System.out.println("Logradouro: " + pj.getLogradouro());
        System.out.println("Cidade: " + pj.getCidade());
        System.out.println("Estado: " + pj.getEstado());
        System.out.println("Telefone: " + pj.getTelefone());
        System.out.println("Email: " + pj.getEmail());
        System.out.println("CNPJ: " + pj.getCnpj());
        System.out.println();
    }
}
}

```



```

// Métodos para exibir Pessoas Físicas

private static void exibirPessoasFisicas(PessoaFisicaDAO pessoaFisicaDAO)
throws SQLException {

    List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();

    if (pessoasFisicas.isEmpty()) {

        System.out.println("Nenhuma Pessoa Física cadastrada.");

    } else {

        for (PessoaFisica pf : pessoasFisicas) {

            System.out.println("ID: " + pf.getId());

            System.out.println("Nome: " + pf.getNome());

            System.out.println("Logradouro: " + pf.getLogradouro());

            System.out.println("Cidade: " + pf.getCidade());

            System.out.println("Estado: " + pf.getEstado());

            System.out.println("Telefone: " + pf.getTelefone());

            System.out.println("Email: " + pf.getEmail());

            System.out.println("CPF: " + pf.getCpf());

            System.out.println();

        }

    }

}

```

```

// Métodos para exibir Pessoas Físicas

private static void exibirPessoasJuridicas(PessoaJuridicaDAO pessoaJuridicaDAO)
throws SQLException {

    List<PessoaJuridica> pessoasJuridicas =
pessoaJuridicaDAO.getPessoasJuridicas();

    if (pessoasJuridicas.isEmpty()) {

        System.out.println("Nenhuma Pessoa Jurídica cadastrada.");

    } else {

        for (PessoaJuridica pj : pessoasJuridicas) {

            System.out.println("ID: " + pj.getId());

            System.out.println("Nome: " + pj.getNome());

            System.out.println("Logradouro: " + pj.getLogradouro());

        }

    }

}

```

```

        System.out.println("Cidade: " + pj.getCidade());
        System.out.println("Estado: " + pj.getEstado());
        System.out.println("Telefone: " + pj.getTelefone());
        System.out.println("Email: " + pj.getEmail());
        System.out.println("CNPJ: " + pj.getCnpj());
        System.out.println();
    }
}
}
}

```

```
package cadastro.model;
```

```

import cadastrobd.model.PessoaFisica;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

```

```

public class PessoaFisicaDAO {
    private Connection conn;

```

```

    public PessoaFisicaDAO(Connection conexao) {
        this.conn = conexao;
    }

```

```

    public void inserirPessoaFisica(PessoaFisica pf) throws SQLException {
        String sqlPessoa = "INSERT INTO Pessoa (nome, logradouro, cidade, estado,
        telefone, email, tipoPessoa) VALUES (?, ?, ?, ?, ?, ?, 'F')";

        String sqlPessoaFisica = "INSERT INTO PessoaFisica (idPessoaFisica, cpf)
        VALUES (?, ?)";

```

```

        try {
            conn.setAutoCommit(false);

```

```

        int pessoald = 0;

        try (PreparedStatement stPessoa = conn.prepareStatement(sqlPessoa,
Statement.RETURN_GENERATED_KEYS)) {

            stPessoa.setString(1, pf.getNome());
            stPessoa.setString(2, pf.getLogradouro());
            stPessoa.setString(3, pf.getCidade());
            stPessoa.setString(4, pf.getEstado());
            stPessoa.setString(5, pf.getTelefone());
            stPessoa.setString(6, pf.getEmail());
            stPessoa.executeUpdate();

            try (ResultSet rs = stPessoa.getGeneratedKeys()) {
                if (rs.next()) {
                    pessoald = rs.getInt(1);
                }
            }
        }

        try (PreparedStatement stPessoaFisica =
conn.prepareStatement(sqlPessoaFisica)) {
            stPessoaFisica.setInt(1, pessoald);
            stPessoaFisica.setString(2, pf.getCpf());
            stPessoaFisica.executeUpdate();
        }

        conn.commit();
    } catch (SQLException e) {
        conn.rollback();
        throw e;
    } finally {
        conn.setAutoCommit(true);
    }
}

```

```

public void alterar(PessoaFisica pf) throws SQLException {

    String sqlPessoa = "UPDATE Pessoa SET nome = ?, logradouro = ?, cidade = ?,
estado = ?, telefone = ?, email = ?, tipoPessoa = 'F' WHERE idPessoa = ?";

    String sqlPessoaFisica = "UPDATE PessoaFisica SET cpf = ? WHERE
idPessoaFisica = ?";

    try {

        conn.setAutoCommit(false);

        try (PreparedStatement stPessoa = conn.prepareStatement(sqlPessoa)) {

            stPessoa.setString(1, pf.getNome());

            stPessoa.setString(2, pf.getLogradouro());

            stPessoa.setString(3, pf.getCidade());

            stPessoa.setString(4, pf.getEstado());

            stPessoa.setString(5, pf.getTelefone());

            stPessoa.setString(6, pf.getEmail());

            stPessoa.setInt(7, pf.getId());

            stPessoa.executeUpdate();

        }

        try (PreparedStatement stPessoaFisica =
conn.prepareStatement(sqlPessoaFisica)) {

            stPessoaFisica.setString(1, pf.getCpf());

            stPessoaFisica.setInt(2, pf.getId());

            stPessoaFisica.executeUpdate();

        }

        conn.commit();

    } catch (SQLException e) {

        conn.rollback();

        throw e;

    } finally {

        conn.setAutoCommit(true);

    }
}

```

```
}  
}
```

```
public void excluir(Integer id) throws SQLException {  
    String sqlPessoaFisica = "DELETE FROM PessoaFisica WHERE idPessoaFisica  
= ?";  
    String sqlPessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";
```

```
    try {  
        conn.setAutoCommit(false);  
        try (PreparedStatement stPessoaFisica =  
conn.prepareStatement(sqlPessoaFisica)) {  
            stPessoaFisica.setInt(1, id);  
            stPessoaFisica.executeUpdate();  
        }  
  
        try (PreparedStatement stPessoa = conn.prepareStatement(sqlPessoa)) {  
            stPessoa.setInt(1, id);  
            stPessoa.executeUpdate();  
        }  
  
        conn.commit();  
    } catch (SQLException e) {  
        conn.rollback();  
        throw e;  
    } finally {  
        conn.setAutoCommit(true);  
    }  
}
```

```
public PessoaFisica getPessoa(Integer id) throws SQLException {  
    String sql = "SELECT Pessoa.idPessoa, Pessoa.nome, Pessoa.logradouro,  
Pessoa.cidade, Pessoa.estado, Pessoa.telefone, Pessoa.email, PessoaFisica.cpf
```

```
FROM Pessoa INNER JOIN PessoaFisica ON Pessoa.idPessoa =
PessoaFisica.idPessoaFisica WHERE Pessoa.idPessoa = ?";
```

```
try (PreparedStatement st = conn.prepareStatement(sql)) {
    st.setInt(1, id);
    try (ResultSet rs = st.executeQuery()) {
        if (rs.next()) {
            return new PessoaFisica(
                rs.getInt("idPessoa"),
                rs.getString("nome"),
                rs.getString("logradouro"),
                rs.getString("cidade"),
                rs.getString("estado"),
                rs.getString("telefone"),
                rs.getString("email"),
                rs.getString("cpf")
            );
        }
    }
}
return null;
}
```

```
public List<PessoaFisica> getPessoas() throws SQLException {
    List<PessoaFisica> list = new ArrayList<>();

    String sql = "SELECT p.*, pf.cpf FROM Pessoa AS p INNER JOIN PessoaFisica
AS pf ON p.idPessoa = pf.idPessoaFisica ORDER BY p.nome";

    try (PreparedStatement st = conn.prepareStatement(sql);
        ResultSet rs = st.executeQuery()) {
        while (rs.next()) {
            list.add(new PessoaFisica(
                rs.getInt("idPessoa"),
                rs.getString("nome"),
                rs.getString("logradouro"),
```

```

        rs.getString("cidade"),
        rs.getString("estado"),
        rs.getString("telefone"),
        rs.getString("email"),
        rs.getString("cpf")
    ));
    }
}
return list;
}
}

```

```

package cadastro.model;

```

```

import cadastrobd.model.PessoaJuridica;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

```

```

public class PessoaJuridicaDAO {
    private Connection conn;

    public PessoaJuridicaDAO(Connection conn) {
        this.conn = conn;
    }

```

```

    private PessoaJuridica extrairPessoaJuridica(ResultSet rs) throws SQLException {
        return new PessoaJuridica(
            rs.getInt("idPessoa"),
            rs.getString("nome"),
            rs.getString("logradouro"),
            rs.getString("cidade"),

```

```

        rs.getString("estado"),
        rs.getString("telefone"),
        rs.getString("email"),
        rs.getString("cnpj")
    );
}

```

```

public PessoaJuridica getPessoa(int id) throws SQLException {
    final String sql = "SELECT P.*, PJ.cnpj FROM Pessoa P INNER JOIN
PessoaJuridica PJ ON P.idPessoa = PJ.idPessoaJuridica WHERE PJ.idPessoaJuridica
= ?";
    try (PreparedStatement stmt = conn.prepareStatement(sql)) {
        stmt.setInt(1, id);
        try (ResultSet rs = stmt.executeQuery()) {
            if (rs.next()) {
                return extrairPessoaJuridica(rs);
            }
        }
    }
    return null;
}

```

```

public List<PessoaJuridica> getPessoasJuridicas() throws SQLException {
    List<PessoaJuridica> list = new ArrayList<>();
    final String sql = "SELECT P.*, PJ.cnpj FROM Pessoa P INNER JOIN
PessoaJuridica PJ ON P.idPessoa = PJ.idPessoaJuridica ORDER BY P.nome";
    try (PreparedStatement stmt = conn.prepareStatement(sql);
        ResultSet rs = stmt.executeQuery()) {
        while (rs.next()) {
            list.add(extrairPessoaJuridica(rs));
        }
    }
    return list;
}

```



```
}
```

```
public void incluir(PessoaJuridica pessoa) throws SQLException {

    final String sqlPessoa = "INSERT INTO Pessoa (nome, logradouro, cidade,
estado, telefone, email, tipoPessoa) VALUES (?, ?, ?, ?, ?, ?, 'J')";

    final String sqlPessoaJuridica = "INSERT INTO PessoaJuridica (idPessoaJuridica,
cnpj) VALUES (?, ?)";

    try {

        conn.setAutoCommit(false);

        int pessoald = 0;

        try (PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa,
Statement.RETURN_GENERATED_KEYS)) {

            stmtPessoa.setString(1, pessoa.getNome());
            stmtPessoa.setString(2, pessoa.getLogradouro());
            stmtPessoa.setString(3, pessoa.getCidade());
            stmtPessoa.setString(4, pessoa.getEstado());
            stmtPessoa.setString(5, pessoa.getTelefone());
            stmtPessoa.setString(6, pessoa.getEmail());
            stmtPessoa.executeUpdate();

            try (ResultSet generatedKeys = stmtPessoa.getGeneratedKeys()) {
                if (generatedKeys.next()) {
                    pessoald = generatedKeys.getInt(1);
                }
            }
        }

        if (pessoald == 0) {
            throw new SQLException("Falha ao inserir pessoa, nenhum ID foi gerado.");
        }
    }
}
```

```

        try (PreparedStatement stmtPessoaJuridica =
conn.prepareStatement(sqlPessoaJuridica)) {

            stmtPessoaJuridica.setInt(1, pessoald);

            stmtPessoaJuridica.setString(2, pessoa.getCnpj());

            stmtPessoaJuridica.executeUpdate();

        }

```

```

        conn.commit();
    } catch (SQLException e) {
        conn.rollback();

        throw e;
    } finally {
        conn.setAutoCommit(true);
    }
}

```

```

public void alterar(PessoaJuridica pessoa) throws SQLException {

    final String sqlPessoa = "UPDATE Pessoa SET nome = ?, logradouro = ?, cidade
= ?, estado = ?, telefone = ?, email = ? WHERE idPessoa = ?";

    final String sqlPessoaJuridica = "UPDATE PessoaJuridica SET cnpj = ? WHERE
idPessoaJuridica = ?";

```

```

    try {
        conn.setAutoCommit(false);

        try (PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa)) {

            stmtPessoa.setString(1, pessoa.getNome());

            stmtPessoa.setString(2, pessoa.getLogradouro());

            stmtPessoa.setString(3, pessoa.getCidade());

            stmtPessoa.setString(4, pessoa.getEstado());

            stmtPessoa.setString(5, pessoa.getTelefone());

            stmtPessoa.setString(6, pessoa.getEmail());

            stmtPessoa.setInt(7, pessoa.getId());

            stmtPessoa.executeUpdate();

        }
    }
}

```

```

    }

    try (PreparedStatement stmtPessoaJuridica =
conn.prepareStatement(sqlPessoaJuridica)) {

        stmtPessoaJuridica.setString(1, pessoa.getCnpj());

        stmtPessoaJuridica.setInt(2, pessoa.getId());

        stmtPessoaJuridica.executeUpdate();

    }

    conn.commit();
} catch (SQLException e) {

    conn.rollback();

    throw e;

} finally {

    conn.setAutoCommit(true);

}

}

```

```

public void excluir(int id) throws SQLException {

    String sqlPessoaJuridica = "DELETE FROM PessoaJuridica WHERE
idPessoaJuridica = ?";

    String sqlPessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";

    try {

        conn.setAutoCommit(false);

        try (PreparedStatement stmtPessoaJuridica =
conn.prepareStatement(sqlPessoaJuridica)) {

            stmtPessoaJuridica.setInt(1, id);

            stmtPessoaJuridica.executeUpdate();

        }

        try (PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa)) {

            stmtPessoa.setInt(1, id);

            stmtPessoa.executeUpdate();

        }

        conn.commit();

    }
}

```

```
    } catch (SQLException e) {  
        conn.rollback();  
        throw e;  
    } finally {  
        conn.setAutoCommit(true);  
    }  
}  
  
}
```

user=loja

password=loja

dburl=jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustServerCertificate=true;

Resultados

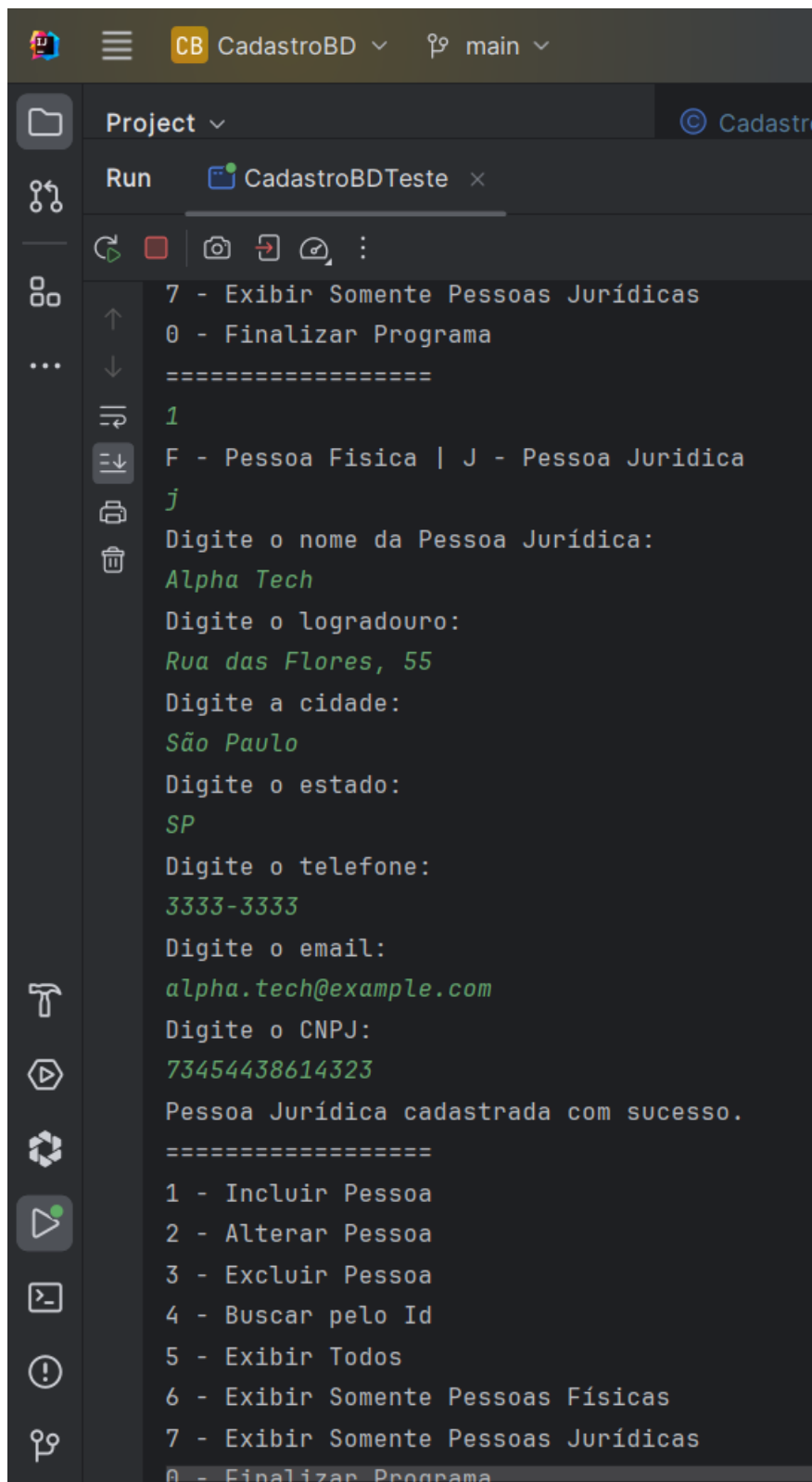
Consultas Realizadas:

- Incluir Pessoa Física

The screenshot shows a Java IDE with a project named 'CadastroBD'. The 'Run' console displays the execution of a program. The program prompts the user to enter details for a physical person, including name, address, city, state, phone, and email. The user's input is shown in green. The program then displays a menu with options to include, alter, exclude, search, or display people.

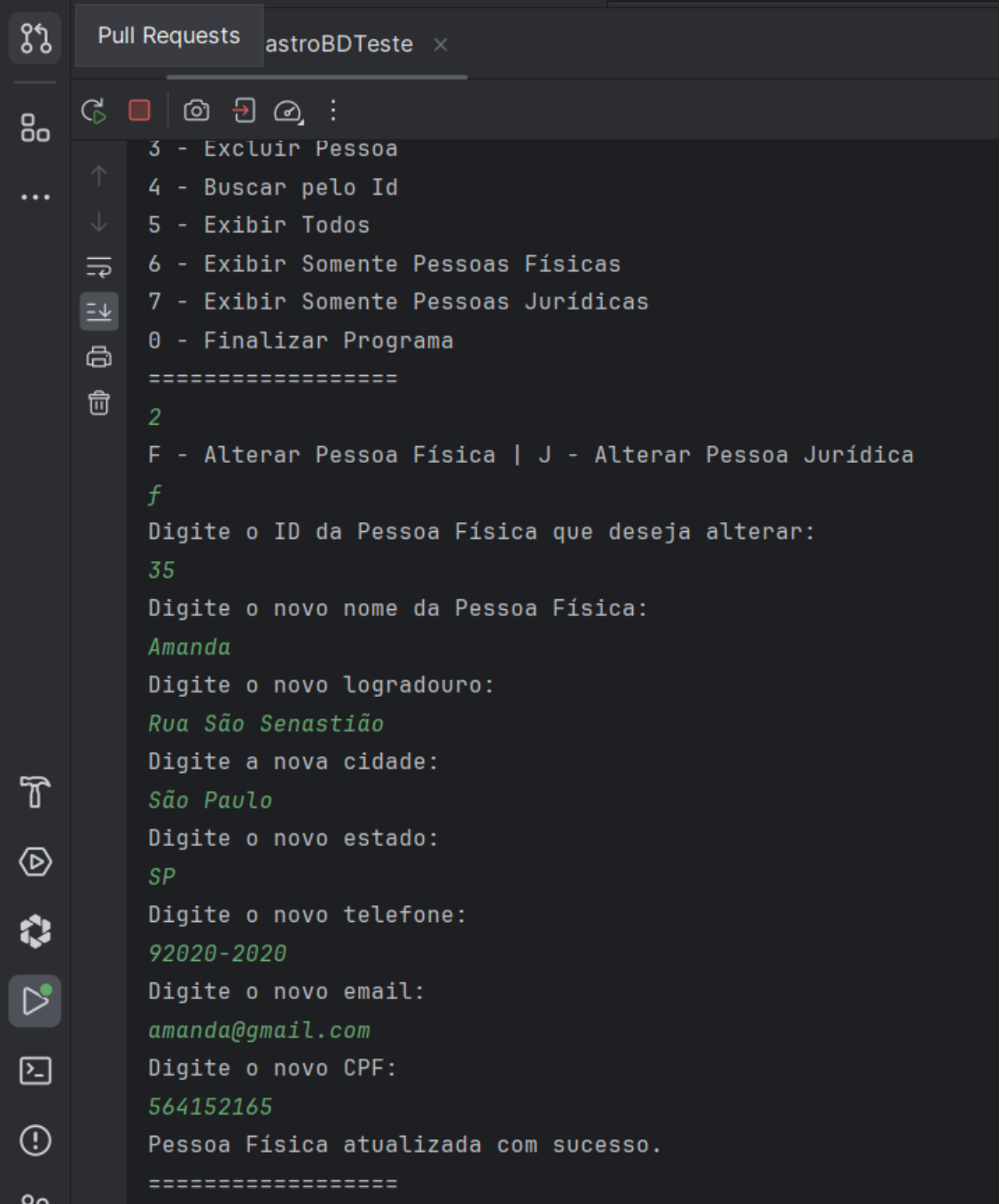
```
Project Alt+1
Run CadastroBDTeste x
1
F - Pessoa Fisica | J - Pessoa Juridica
ff
Digite o nome da Pessoa Física:
Carlos Souza
Digite o logradouro:
Av. Paulista, 111
Digite a cidade:
São Paulo
Digite o estado:
SP
Digite o telefone:
2020-2020
Digite o email:
carlos.souza@inbox.com
Digite o CPF:
52627004194
Pessoa Física cadastrada com sucesso.
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Exibir Somente Pessoas Físicas
7 - Exibir Somente Pessoas Jurídicas
0 - Finalizar Programa
=====
```

- Incluir Pessoa Jurídica



```
Project ▾
Run CadastroBDTeste x
7 - Exibir Somente Pessoas Jurídicas
0 - Finalizar Programa
=====
1
F - Pessoa Fisica | J - Pessoa Juridica
j
Digite o nome da Pessoa Jurídica:
Alpha Tech
Digite o logradouro:
Rua das Flores, 55
Digite a cidade:
São Paulo
Digite o estado:
SP
Digite o telefone:
3333-3333
Digite o email:
alpha.tech@example.com
Digite o CNPJ:
73454438614323
Pessoa Jurídica cadastrada com sucesso.
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Exibir Somente Pessoas Físicas
7 - Exibir Somente Pessoas Jurídicas
0 - Finalizar Programa
```

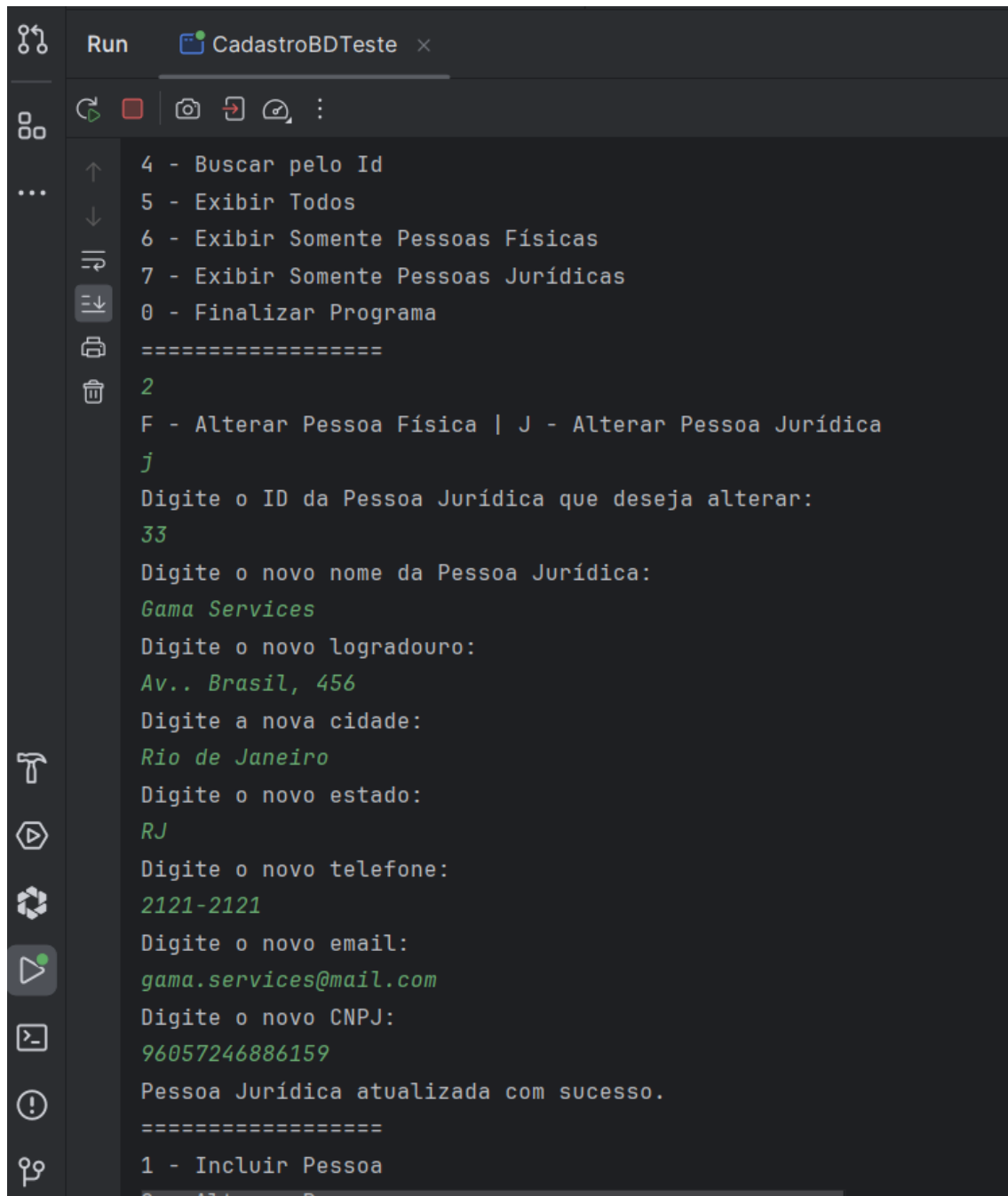
- Alterar Pessoa Física



The screenshot shows a Pull Request interface for a repository named 'astroBDTeste'. On the left, a sidebar contains a menu with options: '3 - Excluir Pessoa', '4 - Buscar pelo Id', '5 - Exibir Todos', '6 - Exibir Somente Pessoas Físicas', '7 - Exibir Somente Pessoas Jurídicas', and '0 - Finalizar Programa'. The '7 - Exibir Somente Pessoas Jurídicas' option is currently selected. The main area displays the terminal output of a program. The output shows a menu where 'F' (Alterar Pessoa Física) was chosen. The user then entered the ID '35', the new name 'Amanda', the new address 'Rua São Senastião', the new city 'São Paulo', the new state 'SP', the new phone number '92020-2020', and the new email 'amanda@gmail.com'. The new CPF '564152165' was also entered. The final output message is 'Pessoa Física atualizada com sucesso.' followed by a separator line.

```
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Exibir Somente Pessoas Físicas
7 - Exibir Somente Pessoas Jurídicas
0 - Finalizar Programa
=====
2
F - Alterar Pessoa Física | J - Alterar Pessoa Jurídica
f
Digite o ID da Pessoa Física que deseja alterar:
35
Digite o novo nome da Pessoa Física:
Amanda
Digite o novo logradouro:
Rua São Senastião
Digite a nova cidade:
São Paulo
Digite o novo estado:
SP
Digite o novo telefone:
92020-2020
Digite o novo email:
amanda@gmail.com
Digite o novo CPF:
564152165
Pessoa Física atualizada com sucesso.
=====
```

- Alterar Pessoa Jurídica



The screenshot shows a Java IDE with a terminal window titled "Run" and "CadastroBDTeste". The terminal displays the following output:

```
4 - Buscar pelo Id
5 - Exibir Todos
6 - Exibir Somente Pessoas Físicas
7 - Exibir Somente Pessoas Jurídicas
0 - Finalizar Programa
=====
2
F - Alterar Pessoa Física | J - Alterar Pessoa Jurídica
j
Digite o ID da Pessoa Jurídica que deseja alterar:
33
Digite o novo nome da Pessoa Jurídica:
Gama Services
Digite o novo logradouro:
Av.. Brasil, 456
Digite a nova cidade:
Rio de Janeiro
Digite o novo estado:
RJ
Digite o novo telefone:
2121-2121
Digite o novo email:
gama.services@mail.com
Digite o novo CNPJ:
96057246886159
Pessoa Jurídica atualizada com sucesso.
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
```


- Excluir Pessoa

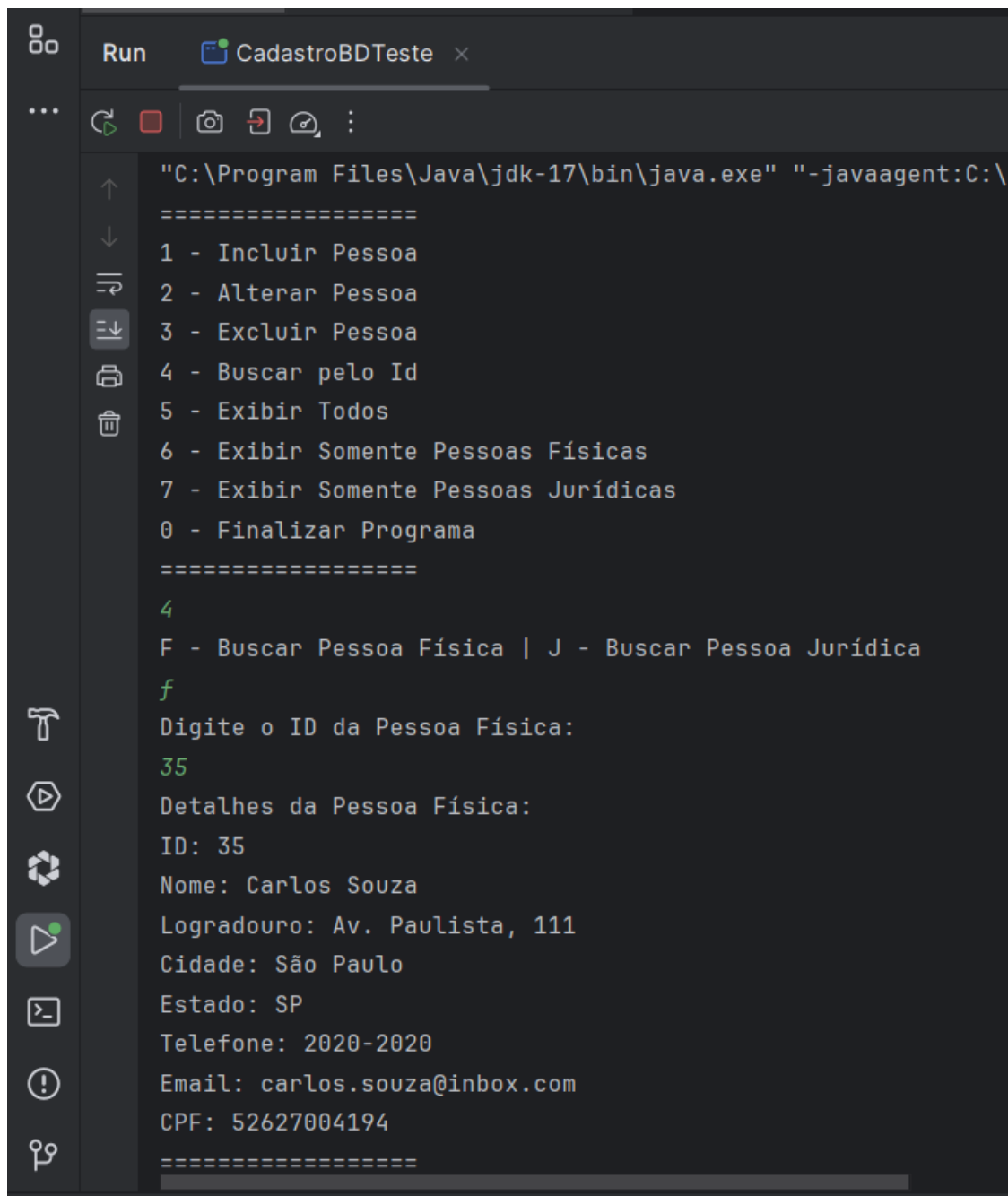
The screenshot shows an IDE with several tabs: CadastroBDTeste.java, PessoaJuridicaDAO.java, PessoaFisicaDAO.java, Pessoa, and PessoaJuridica.java. The 'Pessoa' tab is active, displaying a table with 3 rows and 8 columns: idPessoa, nome, logradouro, cidade, estado, telefone, email, and tipoPessoa.

	idPessoa	nome	logradouro	cidade	estado	telefone	email	tipoPessoa
1	7	Joao	Rua 12, casa 3, Quitanda	Riacho do Sul	PA	1111-1111	email@example.com	F
2	33	Gama Services	Av.. Brasil, 456	Rio de Janeiro	RJ	2121-2121	gama.services@mail.com	J
3	34	Alpha Tech	Rua das Flores, 55	São Paulo	SP	3333-3333	alpha.tech@example.com	J

Below the table, the 'Run' tab for 'CadastroBDTeste' is open, showing the command line and the application's output. The command line is: "C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.3\lib\idea_rt.jar=58838:C:\Program Files\JetBrains\IntelliJ ID". The output shows a menu with options 1-7 and 0, followed by the selection of option 3 (Excluir Pessoa) and the successful exclusion of a physical person with ID 35.

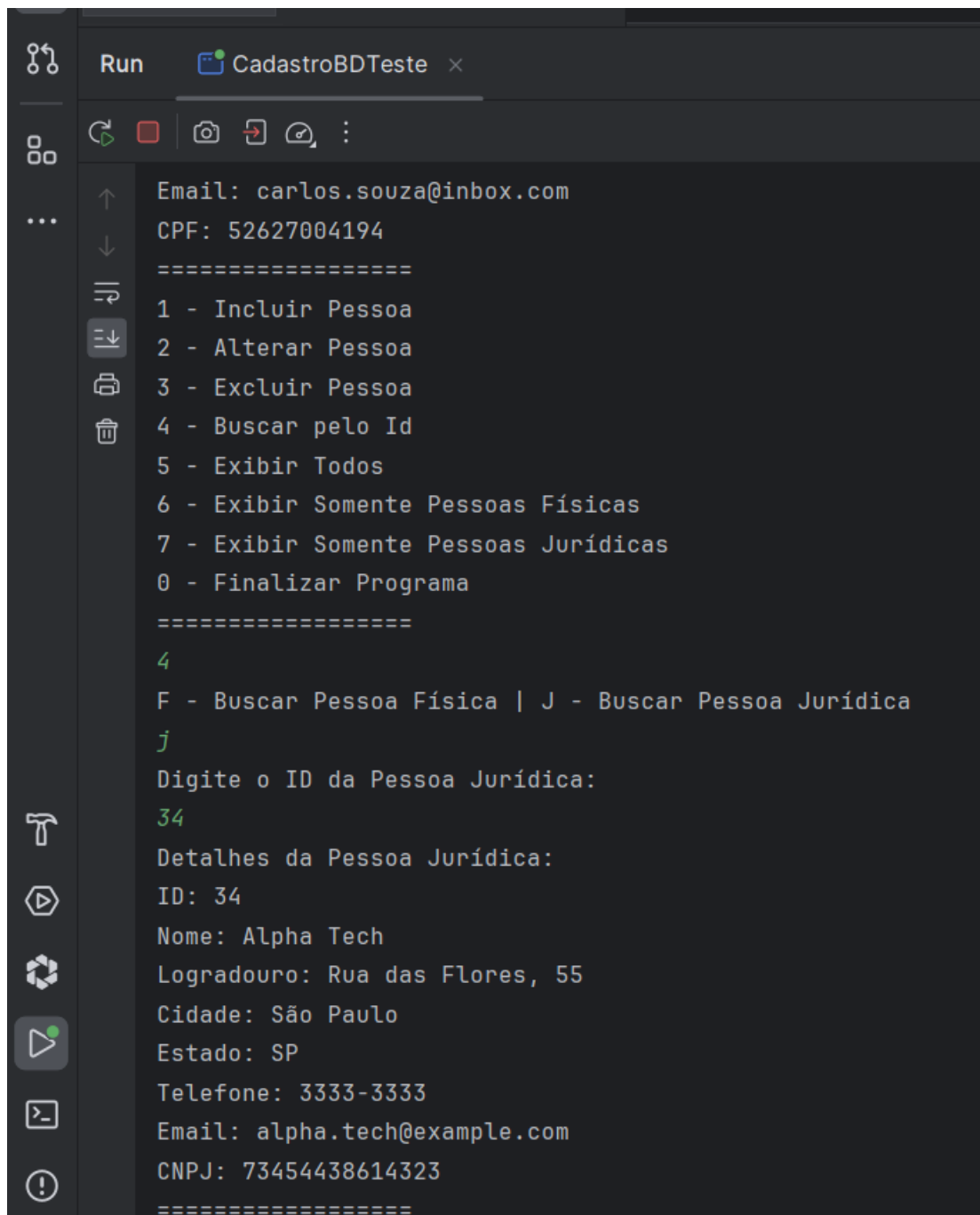
```
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.3\lib\idea_rt.jar=58838:C:\Program Files\JetBrains\IntelliJ ID"
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Exibir Somente Pessoas Fisicas
7 - Exibir Somente Pessoas Juridicas
0 - Finalizar Programa
=====
3
F - Excluir Pessoa Fisica | J - Excluir Pessoa Juridica
f
Digite o ID da Pessoa Fisica a ser excluida:
35
Pessoa Fisica excluida com sucesso.
=====
```

- Busca por ID de Pessoa Física



```
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Exibir Somente Pessoas Físicas
7 - Exibir Somente Pessoas Jurídicas
0 - Finalizar Programa
=====
4
F - Buscar Pessoa Física | J - Buscar Pessoa Jurídica
f
Digite o ID da Pessoa Física:
35
Detalhes da Pessoa Física:
ID: 35
Nome: Carlos Souza
Logradouro: Av. Paulista, 111
Cidade: São Paulo
Estado: SP
Telefone: 2020-2020
Email: carlos.souza@inbox.com
CPF: 52627004194
=====
```

- Busca por ID de Pessoa Jurídica



```
Run CadastroBDTeste x
Email: carlos.souza@inbox.com
CPF: 52627004194
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Exibir Somente Pessoas Físicas
7 - Exibir Somente Pessoas Jurídicas
0 - Finalizar Programa
=====
4
F - Buscar Pessoa Física | J - Buscar Pessoa Jurídica
j
Digite o ID da Pessoa Jurídica:
34
Detalhes da Pessoa Jurídica:
ID: 34
Nome: Alpha Tech
Logradouro: Rua das Flores, 55
Cidade: São Paulo
Estado: SP
Telefone: 3333-3333
Email: alpha.tech@example.com
CNPJ: 73454438614323
=====
```

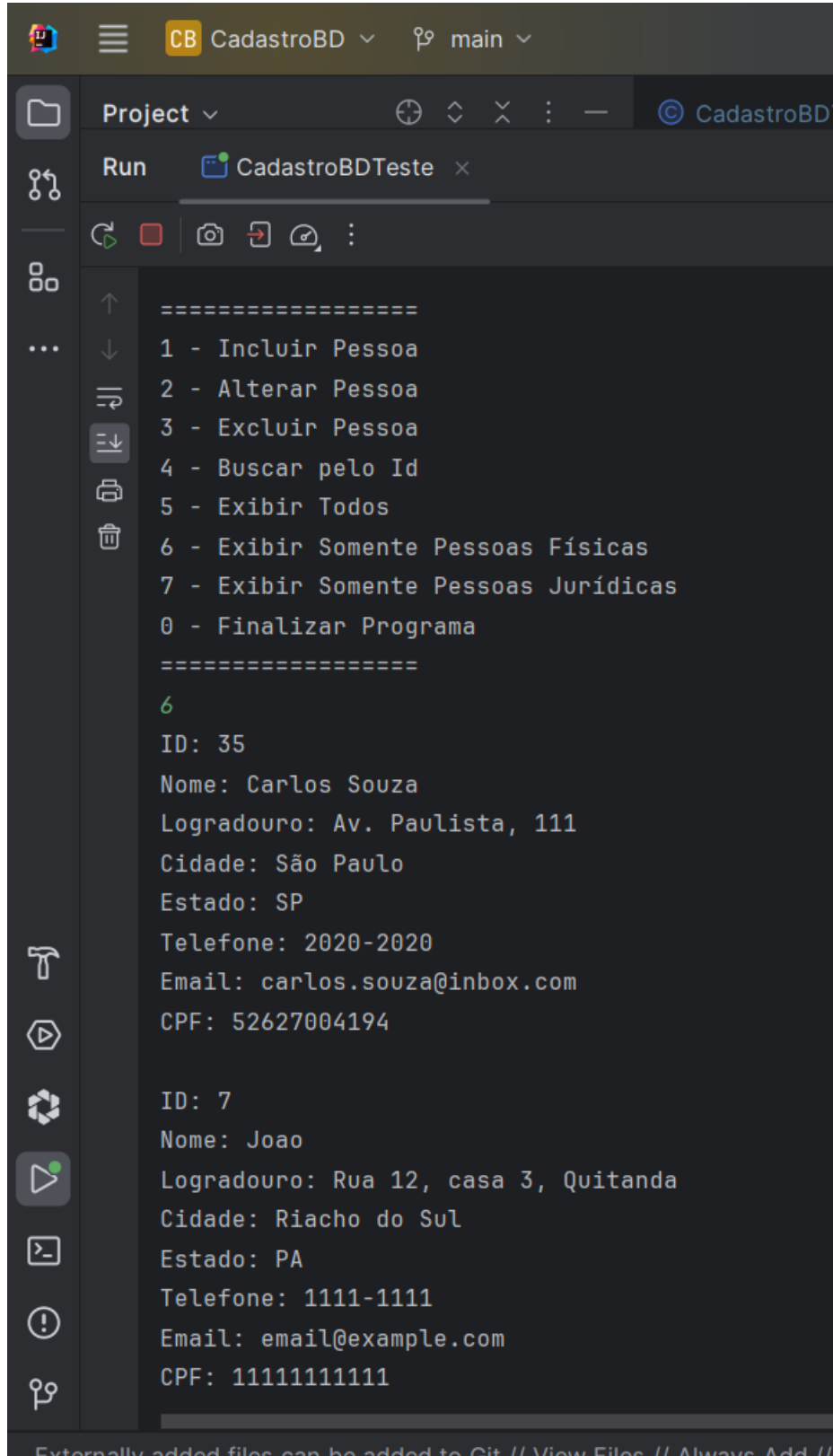
- Exibir Todos

```
Project ▾ © CadastroBDTeste.java
Run CadastroBDTeste x
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Exibir Somente Pessoas Físicas
7 - Exibir Somente Pessoas Jurídicas
0 - Finalizar Programa
=====
5
Pessoas Físicas Cadastradas:
ID: 35
Nome: Carlos Souza
Logradouro: Av. Paulista, 111
Cidade: São Paulo
Estado: SP
Telefone: 2020-2020
Email: carlos.souza@inbox.com
CPF: 52627004194

ID: 7
Nome: Joao
Logradouro: Rua 12, casa 3, Quitanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
Email: email@example.com
CPF: 11111111111

Pessoas Jurídicas Cadastradas:
```

- Exibir somente Pessoa Física



The screenshot shows a Java IDE with a project named 'CadastroBD'. The 'Run' console displays a menu of options for a database application. Option 6, 'Exibir Somente Pessoas Físicas', has been selected, resulting in the display of two physical person records.

```
Project ▾
Run CadastroBDTeste ×

↑
↓
↺
↻
↓
🖨
🗑

=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Exibir Somente Pessoas Físicas
7 - Exibir Somente Pessoas Jurídicas
0 - Finalizar Programa
=====

6

ID: 35
Nome: Carlos Souza
Logradouro: Av. Paulista, 111
Cidade: São Paulo
Estado: SP
Telefone: 2020-2020
Email: carlos.souza@inbox.com
CPF: 52627004194

ID: 7
Nome: Joao
Logradouro: Rua 12, casa 3, Quitanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
Email: email@example.com
CPF: 11111111111

Externally added files can be added to Git // View Files // Always Add //
```

- Exibir somente Pessoas Jurídicas

```
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Exibir Somente Pessoas Físicas
7 - Exibir Somente Pessoas Jurídicas
0 - Finalizar Programa
=====
7
ID: 34
Nome: Alpha Tech
Logradouro: Rua das Flores, 55
Cidade: São Paulo
Estado: SP
Telefone: 3333-3333
Email: alpha.tech@example.com
CNPJ: 73454438614323

ID: 33
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 2222-2222
Email: email@exemple.com
CNPJ: 11111111111111
```

- Finalizar programa

```

=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Exibir Somente Pessoas Físicas
7 - Exibir Somente Pessoas Jurídicas
0 - Finalizar Programa
=====
0
Encerrando o programa.

```

- Exibição da Tabela Pessoa

The screenshot shows a Java IDE with several files open. The 'Pessoa' file is active, displaying a table of person records. The table has columns for idPessoa, nome, logradouro, cidade, estado, telefone, email, and tipoPessoa. There are 4 rows of data.

	idPessoa	nome	logradouro	cidade	estado	telefone	email	tipoPessoa
1	7	Joao	Rua 12, casa 3, Quitanda	Riacho do Sul	PA	1111-1111	email@example.com	F
2	33	Gama Serv...	Av.. Brasil, 456	Rio de Janeiro	RJ	2121-2121	gama.services@mail...	J
3	34	Alpha Tech	Rua das Flores, 55	São Paulo	SP	3333-3333	alpha.tech@exampl...	J
4	35	Amanda	Rua São Senastião	São Paulo	SP	92020-2020	amanda@gmail.com	F

The screenshot shows a SQL IDE with a query editor and a results pane. The query is 'SELECT * FROM Pessoa;'. The results pane shows a table with 8 columns and 4 rows of data.

SQLQuery1.sql - DES...55.Loja (loja (77))* - X

```
SELECT * FROM Pessoa;
```

	idPessoa	nome	logradouro	cidade	estado	telefone	email	tipoPessoa
1	7	Joao	Rua 12, casa 3, Quitanda	Riacho do Sul	PA	1111-1111	email@example.com	F
2	33	Gama Services	Av.. Brasil, 456	Rio de Janeiro	RJ	2121-2121	gama.services@mail.com	J
3	34	Alpha Tech	Rua das Flores, 55	São Paulo	SP	3333-3333	alpha.tech@example.com	J
4	35	Amanda	Rua São Senastião	São Paulo	SP	92020-2020	amanda@gmail.com	F

Análise e Conclusão

Qual a importância dos componentes de middleware, como o JDBC?

Os componentes de middleware como o JDBC são essenciais para abstrair a complexidade do acesso a banco de dados, proporcionando uma interface padronizada para a interação com diferentes tipos de bancos de dados. Isso facilita o desenvolvimento de aplicações Java, permitindo que elas se comuniquem com o banco de dados de maneira eficiente e segura, independente do banco de dados específico utilizado.

Qual a diferença no uso de Statement ou PreparedStatement para a manipulação de dados?

A principal diferença entre Statement e PreparedStatement reside na eficiência e segurança. PreparedStatement é pré-compilado e permite a definição de parâmetros, tornando-o mais rápido e seguro contra SQL Injection. Já o Statement é apropriado para consultas SQL estáticas sem parâmetros, mas é menos eficiente e mais vulnerável a ataques de injeção de SQL.

Como o padrão DAO melhora a manutenibilidade do software?

O padrão DAO (Data Access Object) separa a lógica de acesso a dados do resto da aplicação, reduzindo o acoplamento entre as camadas de negócio e de persistência. Isso facilita a manutenção e a escalabilidade do software, pois alterações na base de dados ou na lógica de acesso a dados podem ser feitas com mínimo impacto sobre o restante do código.

Como a herança é refletida no banco de dados, quando lidamos com um modelo estritamente relacional?

Em um modelo estritamente relacional, a herança pode ser refletida através de tabelas separadas para cada classe ou uma tabela única com colunas para todas as propriedades das classes na hierarquia. Essa representação exige um cuidado adicional na modelagem e no mapeamento objeto-relacional para garantir a integridade e o desempenho adequado do banco de dados.

Quais as diferenças entre a persistência em arquivo e a persistência em banco de dados?

A persistência em arquivo é geralmente mais simples, porém menos eficiente e segura, adequada para dados menos complexos e com menos necessidade de operações de busca e atualização. A persistência em banco de dados oferece mais recursos para gerenciamento, segurança e otimização de dados, sendo mais indicada para dados complexos e aplicações que exigem transações e consultas sofisticadas.

Como o uso de operador lambda simplificou a impressão dos valores contidos nas entidades, nas versões mais recentes do Java?

O uso de operadores lambda nas versões mais recentes do Java simplificou a impressão dos valores contidos nas entidades ao permitir a implementação concisa e expressiva de operações em coleções. Com lambdas, é possível realizar iterações e operações sobre elementos de coleções de forma mais direta e legível, melhorando a clareza e reduzindo a quantidade de código necessário.

Por que métodos acionados diretamente pelo método main, sem o uso de um objeto, precisam ser marcados como static?

Métodos acionados diretamente pelo método main precisam ser marcados como static porque o main é um método estático e, por definição, só pode chamar diretamente outros métodos estáticos. Métodos estáticos pertencem à classe, não a instâncias de objetos, o que os torna acessíveis sem a necessidade de criar uma instância da classe.