Universidade Cidade de São Paulo

Bacharelado em Ciências da Computação

Projeto MVC Sistema Aluno

Profº Jadir Mendonça - Programação Orientada a Objetos

Arthur Miranda da Silva

Edson Gabriel Monteiro Jordão

São Paulo

2023

Introdução

Este trabalho acadêmico aborda o desenvolvimento de um projeto de administração de notas no contexto da disciplina de Programação Orientada a Objetos, lecionada pelo professor Jadir Mendonça no curso de Ciência da Computação. O objetivo do projeto é implementar um Sistema Aluno utilizando o modelo arquitetural MVC (Model-View-Controller), com a utilização das tecnologias Java e SQL, além do driver JDBC e a IDE Eclipse.

No contexto da programação orientada a objetos, o modelo MVC é amplamente utilizado para promover a separação de responsabilidades entre os componentes de um sistema. O modelo é composto pelo Model, que representa a estrutura de dados e a lógica de negócio; a View, responsável pela interface gráfica e apresentação dos dados ao usuário; e o Controller, que faz a intermediação entre o Model e a View, gerenciando as ações e eventos do sistema.

Neste trabalho, serão explorados os principais aspectos do projeto, desde a modelagem do banco de dados e a criação das classes do Model, até a implementação das interfaces gráficas e a lógica de controle no Controller. Serão abordadas as etapas de integração do Java com o banco de dados SQL por meio do driver JDBC, bem como o uso da IDE Eclipse para desenvolver e depurar o sistema. O trabalho busca proporcionar uma compreensão aprofundada da programação orientada a objetos, do modelo MVC e das tecnologias Java e SQL no contexto da administração de notas.

Tela Principal (TelaPrincipal.java)

private boolean contemNumeros(String texto) {

for (char c : texto.toCharArray()) {

if (Character.isDigit(c)) {

return true; // Retorna true se encontrar algum número no texto

}

}

return false; // Retorna false se não encontrar números no texto

}

// SALVAR

try {

Aluno aluno = new Aluno();

// atribuindo o valor dos atributos

String rgmText = txtRgm.getText().trim();

if (rgmText.length() != 8) {

JOptionPane.showMessageDialog(null, "O RGM deve ter no máximo 8 dígitos.", "Erro",

JOptionPane.ERROR\_MESSAGE);

return; // Encerra o método caso o RGM exceda o limite de 8 dígitos

}

aluno.setRgm(Integer.parseInt(rgmText));

String nome = txtNome.getText();

if (contemNumeros(nome)) {

JOptionPane.showMessageDialog(null, "O campo nome não pode conter números.", "Erro",

JOptionPane.ERROR\_MESSAGE);

return; // Encerra o método caso o campo nome contenha números

}

aluno.setNome(nome);

aluno.setEmail(txtEmail.getText().trim());

aluno.setDtaNascimento(fmtdData.getText());

aluno.setCpf(fmtdCpf.getText().trim());

aluno.setCelular(fmtdCelular.getText());

aluno.setUf(ComboBoxUf.getSelectedItem().toString());

String municipio = txtMunicipio.getText();

if (contemNumeros(municipio)) {

JOptionPane.showMessageDialog(null, "O campo município não pode conter números.", "Erro",

JOptionPane.ERROR\_MESSAGE);

return; // Encerra o método caso o campo nome contenha números

}

aluno.setMunicipio(municipio);

aluno.setEndereco(txtEndereco.getText());

aluno.setCurso(comboBoxCurso.getSelectedItem().toString());

aluno.setCampus(comboBoxCampus.getSelectedItem().toString());

aluno.setPeriodo(radioGrupo.getSelection().getActionCommand());

// abrir o BD

AlunoDAO dao = new AlunoDAO();

// salvar

dao.salvar(aluno);

JOptionPane.showMessageDialog(null, "Alterado com Sucesso");

} catch (NumberFormatException e) {

JOptionPane.showMessageDialog(null, "O RGM deve ser um valor inteiro.", "Erro",

JOptionPane.ERROR\_MESSAGE);

} catch (Exception e1) {

JOptionPane.showMessageDialog(null, "Ocorreu um erro ao salvar o aluno: " + e1.getMessage(), "Erro",

JOptionPane.ERROR\_MESSAGE);

System.out.println(e1.getMessage());

}

}

});

// ATUALIZAR

// criando o objeto aluno

Aluno aluno = new Aluno();

// atribuindo o valor dos atributos

aluno.setRgm(Integer.parseInt(txtRgm.getText()));

aluno.setNome(txtNome.getText());

aluno.setEmail(txtEmail.getText());

aluno.setDtaNascimento(fmtdData.getText());

aluno.setCpf(fmtdCpf.getText());

aluno.setCelular(fmtdCelular.getText());

aluno.setUf(ComboBoxUf.getSelectedItem().toString());

aluno.setMunicipio(txtMunicipio.getText());

aluno.setEndereco(txtEndereco.getText());

aluno.setCurso(comboBoxCurso.getSelectedItem().toString());

aluno.setCampus(comboBoxCampus.getSelectedItem().toString());

aluno.setPeriodo(radioGrupo.getSelection().getActionCommand());

// abrir o BD

try {

AlunoDAO dao = new AlunoDAO();

// salvar

dao.atualizar(aluno);

JOptionPane.showMessageDialog(null, "Alterado com Sucesso");

} catch (Exception e1) {

JOptionPane.showMessageDialog(null, e1.getMessage());

System.out.println(e1.getMessage());

}

}

});

// CONSULTAR

try {

// abrir o banco de dados

AlunoDAO dao = new AlunoDAO();

//

// criando a variavel rgm

int rgm = Integer.parseInt(txtRgm.getText());

//

// consultando um aluno

Aluno alunoprocurado = dao.procurarAluno(rgm);

//

// populando na tela

txtNome.setText(alunoprocurado.getNome());

txtEmail.setText(alunoprocurado.getEmail());

System.out.println("rgm: "+alunoprocurado.getRgm());

System.out.println("nome: "+alunoprocurado.getNome());

System.out.println("email: "+alunoprocurado.getEmail());

System.out.println("uf: "+ alunoprocurado.getUf());

System.out.println("cpf: "+ alunoprocurado.getCpf());

System.out.println("campus: "+ alunoprocurado.getCampus());

System.out.println("celular: "+ alunoprocurado.getCelular());

System.out.println("curso: "+ alunoprocurado.getCurso());

System.out.println("municipio: "+ alunoprocurado.getMunicipio());

System.out.println("periodo: "+ alunoprocurado.getPeriodo());

System.out.println("endereco: "+ alunoprocurado.getEndereco());

System.out.println("nascimento: "+ alunoprocurado.getDtaNascimento());

fmtdData.setText(aluno.getDtaNascimento());

fmtdCpf.setText(aluno.getCpf());

fmtdCelular.setText(aluno.getCelular());

txtMunicipio.setText(aluno.getMunicipio());

txtEndereco.setText(aluno.getEndereco());

aluno.setUf(ComboBoxUf.getSelectedItem().toString());

aluno.setCurso(comboBoxCurso.getSelectedItem().toString());

aluno.setCampus(comboBoxCampus.getSelectedItem().toString());

aluno.setPeriodo(radioGrupo.getSelection().getActionCommand());

}catch(Exception e1) {

JOptionPane.showMessageDialog(null, "Erro ao procurar");

}

}

});

//EXCLUIR

Aluno aluno = new Aluno();

// popular o meu projeto

aluno.setRgm(Integer.parseInt(txtRgm.getText()));

try {

AlunoDAO dao = new AlunoDAO();

// excluir aluno

dao.excluir(aluno);

JOptionPane.showMessageDialog(null, "Excluido com Sucesso");

} catch(Exception e1) {

JOptionPane.showMessageDialog(null, e1.getMessage());

}

}

});

// SAIR

System.exit(0);

}

});

//SALVAR NOTAS

// inicializando o aluno

Boletim boletim = new Boletim();

boletim.setDisciplina(comboBoxDisciplina.getSelectedItem().toString());

boletim.setNota(Double.parseDouble(comboBoxNota.getSelectedItem().toString()));

boletim.setFaltas(txtFaltasNotas.getText().parseInt());

boletim.setSemestre(comboBoxSemestre.getSelectedItem().toString());

try {

// abrir o BD

BoletimDAO dao = new BoletimDAO();

// salvar

dao.salvar(boletim);

JOptionPane.showMessageDialog(null, "Salvo com Sucesso");

} catch (Exception e1) {

JOptionPane.showMessageDialog(null, e1.getMessage());

System.out.println(e1.getMessage());

}

}

});

//CONSULTAR NOTAS

try {

// abrir o banco de dados

AlunoDAO dao = new AlunoDAO();

//

// criando a variavel rgm

int rgm = Integer.parseInt(txtRgm.getText());

//

// consultando um aluno

Aluno aluno = dao.procurarAluno(rgm);

// populando na tela

txtNomeNotas.setText(aluno.getNome());

txtCursoNotas.setText(aluno.getCurso());

if (txtCursoNotas.getText().equals("Ciência da Computação")) {

comboBoxDisciplina.setModel(new DefaultComboBoxModel(new String[] { "Selecione a Disciplina",

"Algoritmos e Estruturas de Dados", "Arquitetura de Computadores", "Banco de Dados",

"Compiladores", "Inteligência Artificial", "Linguagens de Programação",

"Redes de Computadores", "Sistemas Operacionais" }));

} else if (txtCursoNotas.getText().equals("Analise e desenvolvimento de sistemas")) {

comboBoxDisciplina.setModel(

new DefaultComboBoxModel(new String[] { "Selecione a Disciplina", "Análise de Sistemas",

"Banco de Dados", "Engenharia de Software", "Programação Orientada a Objetos",

"Sistemas Distribuídos", "Testes de Software", "Web Design" }));

}else if (txtCursoNotas.getText().equals("Engenharia Mecânica")) {

comboBoxDisciplina.setModel(new DefaultComboBoxModel(new String[] { "Selecione a Disciplina",

"Cálculo Diferencial e Integral", "Dinâmica dos Sólidos", "Fenômenos de Transporte",

"Mecânica dos Fluidos", "Mecânica dos Sólidos", "Termodinâmica" }));

} else if (txtCursoNotas.getText().equals("Engenharia Mecatrônica")) {

comboBoxDisciplina.setModel(new DefaultComboBoxModel(new String[] { "Selecione a Disciplina",

"Controle Automático", "Eletrônica Digital", "Eletrônica Analógica",

"Mecânica dos Sistemas Mecatrônicos", "Microcontroladores e Microprocessadores",

"Sistemas Embarcados", "Sistemas Hidráulicos e Pneumáticos" }));

};

}catch(Exception e1) {

JOptionPane.showMessageDialog(null, "Erro ao procurar");

}

}

});

Back-end (AlunoDAO.java)

package br.edu.unicid.dao;

import java.sql.\*;

import br.edu.unicid.model.Aluno;

import br.edu.unicid.util.ConnectionFactory;

public class AlunoDAO {

private Connection conn;

private PreparedStatement ps;

private ResultSet rs;

private Aluno aluno;

public AlunoDAO() throws Exception {

// chama a classe ConnectionFactory e estabele uma conexão

try {

this.conn = ConnectionFactory.getConnection();

} catch (Exception e) {

throw new Exception("erro: \n" + e.getMessage());

}

}

// método de salvar

public void salvar(Aluno aluno) throws Exception {

if (aluno == null)

throw new Exception("O valor passado nao pode ser nulo");

try{

String SQL = "INSERT INTO " + "testee (rgm, nome, email, dtaNascimento, cpf, uf, celular, municipio, endereco, curso, campus, periodo) values (?,?,?,?,?,?,?,?,?,?,?,?)";

ps = conn.prepareStatement(SQL);

ps.setInt(1, aluno.getRgm());

ps.setString(2, aluno.getNome());

ps.setString(3, aluno.getEmail());

ps.setString(4, aluno.getDtaNascimento());

ps.setString(5, aluno.getCpf());

ps.setString(6, aluno.getUf());

ps.setString(7, aluno.getCelular());

ps.setString(8, aluno.getMunicipio());

ps.setString(9, aluno.getEndereco());

ps.setString(10, aluno.getCurso());

ps.setString(11, aluno.getCampus());

ps.setString(12, aluno.getPeriodo());

ps.executeUpdate();

}catch (SQLException sqle) {

throw new Exception("Erro ao inserir dados " + sqle);

}finally {

ConnectionFactory.closeConnection(conn, ps);

}

}

// método de atualizar

public void atualizar(Aluno aluno) throws Exception {

if (aluno == null)

throw new Exception("O valor passado nao pode ser nulo");

try {

String SQL = "UPDATE testee set nome=?, email=?, dtaNascimento=?, cpf=?, uf=?, celular=?, municipio=?, endereco=?, curso=?, campus=?, periodo=? " + "WHERE rgm=?";

ps = conn.prepareStatement(SQL);

ps.setString(1, aluno.getNome());

ps.setString(2, aluno.getEmail());

ps.setString(3, aluno.getDtaNascimento());

ps.setString(4, aluno.getCpf());

ps.setString(5, aluno.getUf());

ps.setString(6, aluno.getCelular());

ps.setString(7, aluno.getMunicipio());

ps.setString(8, aluno.getEndereco());

ps.setString(9, aluno.getCurso());

ps.setString(10, aluno.getCampus());

ps.setString(11, aluno.getPeriodo());

ps.setInt(12, aluno.getRgm());

ps.executeUpdate();

} catch (SQLException sqle) {

throw new Exception("Erro ao alterar dados " + sqle);

} finally {

ConnectionFactory.closeConnection(conn, ps);

}

}

// // método de excluir

public void excluir(Aluno aluno) throws Exception {

if (aluno == null)

throw new Exception("O valor passado nao pode ser nulo");

try {

String SQL = "DELETE FROM testee WHERE rgm = ?";

// conn = this.conn; // sem efeito

ps = conn.prepareStatement(SQL);

ps.setInt(1, aluno.getRgm());

ps.executeUpdate();

} catch (SQLException sqle) {

throw new Exception("Erro ao excluir dados " + sqle);

} finally {

ConnectionFactory.closeConnection(conn, ps);

}

}

// // Procurar Aluno

public Aluno procurarAluno(int rgm) throws Exception {

try {

String SQL = "SELECT \* FROM testee WHERE rgm=?";

// conn = this.conn; // sem efeito

ps = conn.prepareStatement(SQL);

ps.setInt(1, rgm);

rs = ps.executeQuery();

if (rs.next()) {

int ca = rs.getInt(1);

String nome = rs.getString(2);

String email = rs.getString(3);

String nascimento = rs.getString(4);

String cpf = rs.getString(5);

String uf = rs.getString(6);

String celular = rs.getString(7);

String municipio = rs.getString(8);

String endereco = rs.getString(9);

String curso = rs.getString(10);

String campus = rs.getString(11);

String periodo = rs.getString(12);

aluno = new Aluno(ca, nome, email, nascimento, cpf, uf, celular, municipio, endereco, curso, campus, periodo);

}

return aluno;

} catch (SQLException sqle) {

throw new Exception(sqle);

} finally {

ConnectionFactory.closeConnection(conn, ps, rs);

}

}

}

Back-end(BoletimDAO.java)

package br.edu.unicid.dao;

import java.sql.\*;

import br.edu.unicid.model.Boletim;

import br.edu.unicid.util.ConnectionFactory;

public class BoletimDAO {

private Connection conn;

private PreparedStatement ps;

private ResultSet rs;

/\*\*

\*

\*/

private Boletim boletim;

public BoletimDAO() throws Exception {

// chama a classe ConnectionFactory e estabele uma conexão

try {

this.conn = ConnectionFactory.getConnection();

} catch (Exception e) {

throw new Exception("erro: \n" + e.getMessage());

}

}

// método de salvar

public void salvar(Boletim boletim) throws Exception {

if (boletim == null)

throw new Exception("O valor passado nao pode ser nulo");

try{

String SQL = "INSERT INTO " + "testeboletim (rgm, disciplina, semestre, nota, faltas) values (?,?,?,?,?)";

ps = conn.prepareStatement(SQL);

ps.setInt(1, boletim.getRgm());

ps.setString(2, boletim.getDisciplina());

ps.setString(3, boletim.getSemestre());

ps.setDouble(4, boletim.getNota());

ps.setInt(5, boletim.getFaltas());

ps.executeUpdate();

}catch (SQLException sqle) {

throw new Exception("Erro ao inserir dados " + sqle);

}finally {

ConnectionFactory.closeConnection(conn, ps);

}

}

// método de atualizar

public void atualizar(Boletim boletim) throws Exception {

if (boletim == null)

throw new Exception("O valor passado nao pode ser nulo");

try {

String SQL = "UPDATE tb\_notas\_faltas set disciplina=? semestre=?, nota=?, faltas=? WHERE rgm = ?";

ps = conn.prepareStatement(SQL);

ps.setString(1, boletim.getDisciplina());

ps.setString(2, boletim.getSemestre());

ps.setDouble(3, boletim.getNota());

ps.setInt(4, boletim.getFaltas());

ps.setInt(5, boletim.getRgm());

ps.executeUpdate();

} catch (SQLException sqle) {

throw new Exception("Erro ao alterar dados " + sqle);

} finally {

ConnectionFactory.closeConnection(conn, ps);

}

}

// Função para excluir

public void excluir(Boletim boletim) throws Exception {

if (boletim == null)

throw new Exception("O valor passado nao pode ser nulo");

try {

String SQL = "DELETE FROM tb\_notas\_faltas WHERE rgm = ?";

ps = conn.prepareStatement(SQL);

ps.setInt(1, boletim.getRgm());

ps.executeUpdate();

} catch (SQLException sqle) {

throw new Exception("Erro ao excluir dados " + sqle);

} finally {

ConnectionFactory.closeConnection(conn, ps);

}

}

// Função para procurar notas no boletim

public Boletim procurarBoletim(int rgm) throws Exception {

try {

String SQL = "SELECT \* FROM tb\_notas\_faltas WHERE rgm=?";

// conn = this.conn; // sem efeito

ps = conn.prepareStatement(SQL);

ps.setInt(1, rgm);

rs = ps.executeQuery();

if (rs.next()) {

int ca = rs.getInt(1);

String disciplina = rs.getString(2);

String semestre = rs.getString(3);

double nota = rs.getDouble(4);

int faltas = rs.getInt(5);

boletim = new Boletim(ca, disciplina, semestre, nota, faltas);

}

return boletim;

} catch (SQLException sqle) {

throw new Exception(sqle);

} finally {

ConnectionFactory.closeConnection(conn, ps, rs);

}

}

}

Back-end(Aluno.java)

package br.edu.unicid.model;

public class Aluno {

private int rgm;

private String nome;

private String email;

private String dtaNascimento;

private String cpf;

private String celular;

private String uf;

private String municipio;

private String endereco;

private String curso;

private String campus;

private String periodo;

// construtor vazio

public Aluno() {}

// construtor com campos

public Aluno(int rgm, String nome, String email, String dtaNascimento, String cpf, String uf, String celular, String municipio, String endereco, String curso, String campus, String periodo) {

this.rgm = rgm;

this.nome = nome;

this.email = email;

this.dtaNascimento = dtaNascimento;

this.cpf = cpf;

this.celular = celular;

this.uf = uf;

this.municipio = municipio;

this.endereco = endereco;

this.curso = curso;

this.campus = campus;

this.periodo = periodo;

}

// getters e setters

public int getRgm() {

return rgm;

}

public void setRgm(int rgm) {

this.rgm = rgm;

}

public String getNome() {

return nome;

}

public void setNome(String nome) {

this.nome = nome;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getDtaNascimento() {

return dtaNascimento;

}

public void setDtaNascimento(String dtaNascimento) {

this.dtaNascimento = dtaNascimento;

}

public String getCpf() {

return cpf;

}

public void setCpf(String cpf) {

this.cpf = cpf;

}

public String getCelular() {

return celular;

}

public void setCelular(String celular) {

this.celular = celular;

}

public String getUf() {

return uf;

}

public void setUf(String uf) {

this.uf = uf;

}

public String getMunicipio() {

return municipio;

}

public void setMunicipio(String municipio) {

this.municipio = municipio;

}

public String getEndereco() {

return endereco;

}

public void setEndereco(String endereco) {

this.endereco = endereco;

}

public String getCurso() {

return curso;

}

public void setCurso(String curso) {

this.curso = curso;

}

public String getCampus() {

return campus;

}

public void setCampus(String campus) {

this.campus = campus;

}

public String getPeriodo() {

return periodo;

}

public void setPeriodo(String periodo) {

this.periodo = periodo;

}

}

Back-end(Boletim.java)

package br.edu.unicid.model;

public class Boletim {

private int rgm;

private String disciplina;

private String semestre;

private double nota;

private int faltas;

// construtor vazio

public Boletim() {}

// construtor com campos

public Boletim(int rgm, String disciplina, String semestre, double nota, int faltas) {

this.rgm = rgm;

this.disciplina = disciplina;

this.semestre= semestre;

this.nota = nota;

this.faltas = faltas;

}

public int getRgm() {

return rgm;

}

public void setRgm(int rgm) {

this.rgm = rgm;

}

public String getDisciplina() {

return disciplina;

}

public void setDisciplina(String disciplina) {

this.disciplina = disciplina;

}

public String getSemestre() {

return semestre;

}

public void setSemestre(String semestre) {

this.semestre = semestre;

}

public double getNota() {

return nota;

}

public void setNota(double nota) {

this.nota = nota;

}

public int getFaltas() {

return faltas;

}

public void setFaltas(int faltas) {

this.faltas = faltas;

}

}

Back-end(ConnectionFactory.java)

package br.edu.unicid.util;

import java.sql.\*;

public class ConnectionFactory {

public static Connection getConnection() throws Exception {

try {

// indica qual é o banco de dados que estou utilizando e seu driver

Class.forName("com.mysql.jdbc.Driver");

// estabelece a conexao e retorna uma conexao

String url = "jdbc:mysql://localhost:3306/tb\_aluno?characterEncoding=utf8";

String login = "root";

String senha = "";

return DriverManager.getConnection(url,login,senha);

} catch (Exception e) {

throw new Exception(e.getMessage());

}

}

// fecha uma conexão de três formas: conn, stmt, rs

public static void closeConnection(Connection conn, Statement stmt,

ResultSet rs) throws Exception {

close(conn, stmt, rs);

}

public static void closeConnection(Connection conn, Statement stmt)

throws Exception {

close(conn, stmt, null);

}

public static void closeConnection(Connection conn) throws Exception {

close(conn, null, null);

}

private static void close(Connection conn, Statement stmt, ResultSet rs)

throws Exception {

try {

if (rs != null)

rs.close();

if (stmt != null)

stmt.close();

if (conn != null)

conn.close();

} catch (Exception e) {

throw new Exception(e.getMessage());

}

}

}

Conclusão

Ao concluir este trabalho acadêmico, fica evidente a importância do aprendizado da disciplina de Programação Orientada a Objetos, ministrada pelo professor Jadir Mendonça, no curso de Ciência da Computação. Através do desenvolvimento do projeto de administração de notas Sistema Aluno no modelo MVC, utilizando as tecnologias Java e SQL, juntamente com o driver JDBC e a IDE Eclipse, adquirimos conhecimentos valiosos sobre a arquitetura de software e as ferramentas essenciais para a construção de sistemas robustos e escaláveis.

A aplicação do modelo arquitetural MVC demonstrou-se altamente eficiente para a separação das responsabilidades e organização do código fonte, permitindo uma manutenção mais fácil e uma escalabilidade adequada do sistema. A divisão clara entre Model, View e Controller possibilitou um desenvolvimento mais estruturado e uma compreensão aprofundada das interações entre os componentes.

A utilização da linguagem Java, juntamente com o SQL e o driver JDBC, possibilitou a criação de um sistema capaz de interagir com um banco de dados relacional, garantindo a persistência e recuperação dos dados de notas dos alunos. Além disso, a IDE Eclipse se mostrou uma ferramenta poderosa, oferecendo recursos avançados de desenvolvimento e uma interface amigável para a implementação e depuração do projeto.

Este trabalho nos proporcionou um aprendizado significativo sobre programação orientada a objetos, arquitetura MVC, Java, SQL, driver JDBC e a IDE Eclipse. O conhecimento adquirido ao longo do desenvolvimento do Sistema Aluno nos capacita a enfrentar desafios futuros no desenvolvimento de sistemas complexos e nos prepara para atuar de forma competente no mercado de trabalho. O trabalho foi uma oportunidade valiosa para aplicar os conceitos teóricos estudados em sala de aula em um projeto prático e relevante para a área da computação.