



Model



View



Controller

Alisson RS

Linguagem de programação II

Padrões de projeto
Modelo-Visão-Controle

ATIVIDADE PRÁTICA

A atividade consiste em aprender um pouco da estruturação de um projeto utilizando o MVC (Modelo, Visão e Controle).

Iremos trabalhar em cima de um programa que faz o controle de cadastros de alunos utilizando GUI (Jframe).

CONTROLE DE ALUNOS

Ao lado está a interface que iremos trabalhar.

Se trata de um controle de alunos (Inserção, edição e remoção), onde os atributos dos alunos são matrícula (chave primária) e nome.

The screenshot shows a Windows application window titled "MVC - Alunos". The main area displays a table with three rows of student data:

Matrícula	Nome
155	Raphael
565	Amanda
524	João Pedro

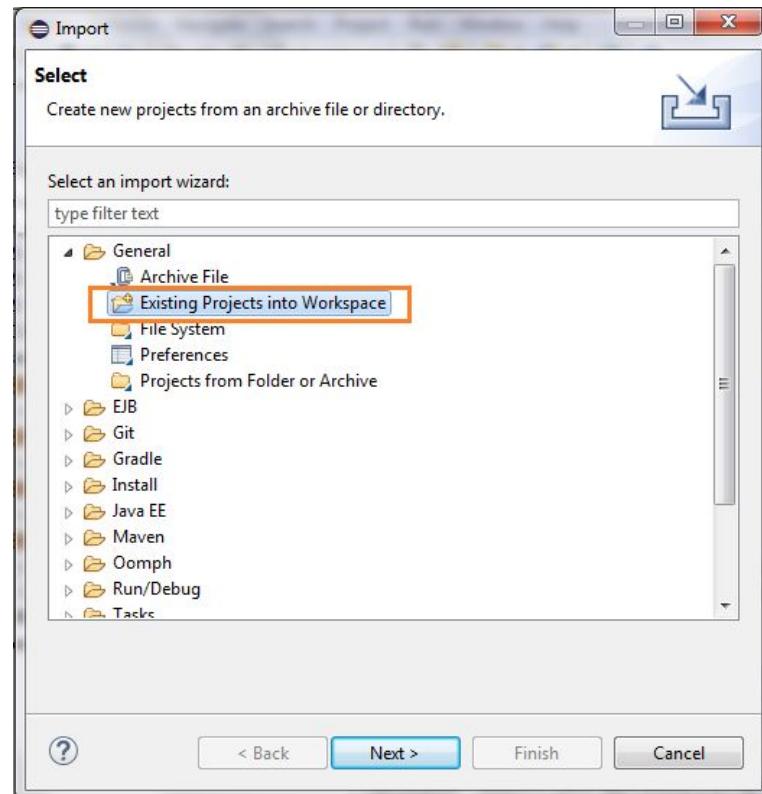
Each row has two buttons to its right: "Editar" (Edit) and "Excluir" (Delete). Below the table is a form with input fields for "Matrícula" and "Nome", and two buttons at the bottom: "Adicionar Aluno" (Add Student) and "Limpar" (Clear).

VAMOS COMEÇAR :)

Primeiro importe e execute o projeto disponibilizado no moodle ou no link abaixo:

<https://www.dropbox.com/s/anbebpq2fq5nmtv/MVC-Project.zip?dl=0>

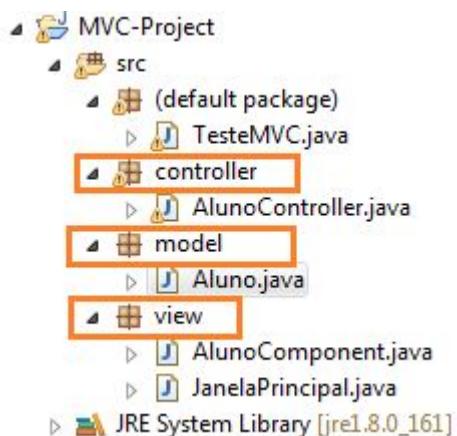
Execute a classe TesteMVC para testar o funcionamento do programa.



ENTENDENDO O CÓDIGO

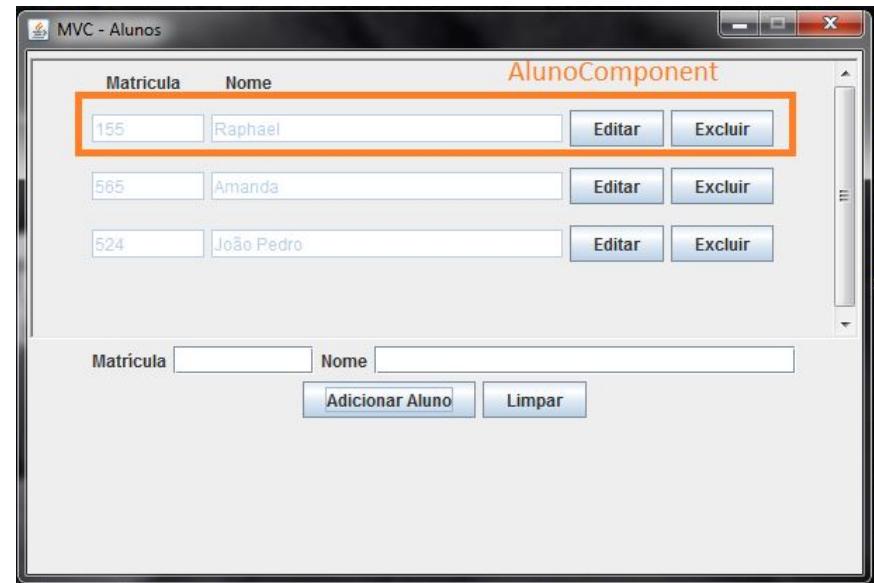
O projeto conta com 3 pacotes.

- model : possui a classe Aluno, que tem os atributos de todo aluno (nome e matrícula) e uma função para acessar todos os alunos cadastrados na base de dados. (*obs: neste projeto se utiliza um arrayList em tempo de execução para o armazenamento dos alunos, para um projeto real seria ideal armazenar essas informações em um banco de dados.)



ENTENDENDO O CÓDIGO

- view : Representa as interfaces que o usuário irá interagir, no nosso projeto temos duas classes neste pacote, sendo elas:
 - JanelaPrincipal : Como o nome já diz, é a janela inicial onde o usuário irá interagir diretamente.
 - AlunoComponent : É basicamente um JPanel Customizado que contém as informações e alguns botões de interação para cada aluno inserido.



ENTENDENDO O CÓDIGO

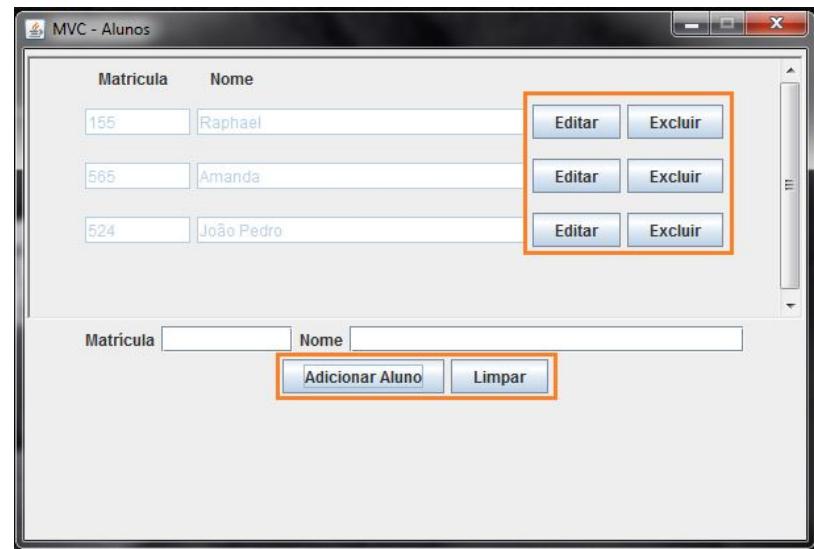
- controller: como já visto, o controller é um intermediador entre a view e o modelo, no nosso projeto o controller é responsável por fazer a ligação entre uma requisição da view, retornando um valor por meio do modelo. Exemplificando, ao clicar em um botão na view, um método do controller é chamado retornando um valor para a view, utilizando o modelo para alteração dos dados.

MÃOS NA MASSA

Agora que entendemos um pouco sobre o projeto, vamos fazer uma pequena atividade.

Consiste em dar funcionalidade a todos os botões da view para controle dos dados.

Inserir, Editar e Excluir



PRIMEIROS PASSOS

Vamos iniciar pelo mais fácil, primeiro vamos adicionar funcionalidade para o botão de limpar as informações do aluno que será inserido.

Isso é uma alteração somente na view (JanelaPrincipal, no método clear()).



```
// Limpa os dados do formulário
public void clear() {
    textFieldMatricula.setText("");
    textFieldName.setText("");
}
```

Esse método clear é chamado no listener do botão limpar (dentro da função init()). Por isso quando clicarmos no botão de limpar, os dados do formulário serão apagados.

Simples, não?

PASSO A PASSO - INSERINDO

Vamos fazer a inserção dos dados.
Agora não será uma modificação somente na view, pois devemos salvar os dados na nossa base de dados.

Faremos isso utilizando o Controller para se conectar com o modelo e assim salvar o aluno no nosso arrayList.

A screenshot of a Java Swing application window. It contains two text input fields: 'Matrícula' and 'Nome'. Below these fields are two buttons: 'Adicionar Aluno' and 'Limpar'. The 'Adicionar Aluno' button is highlighted with a red rectangular border.

Vamos criar uma função no controller para se comunicar com o modelo e assim salvar os alunos.

Desse modo:

```
public void salvarAluno() {  
    // Pega os dados da view  
    int matricula = view.getMatricula();  
    String nome = view.getNameAluno();  
  
    // Passa os dados pro modelo (base de dados)  
    model.addAluno(matricula, nome);  
}
```

PASSO A PASSO - INSERINDO

Continuando, devemos agora adicionar um listener ao botão de inserir. Fazemos isso dentro do controller no método `setListeners()`;

```
// Listener para o botão adicionar aluno
view.addListenerAddButton(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent arg0) {
        try {
            // Valida os dados da view
            if (validate()) {
                // Salva os dados na base de dados
                salvarAluno();
                // Atualiza a tela
                loadAlunos();
                // Limpa os text field do formulário
                view.clear();
                // Mostra na tela que o aluno foi salvo
                view.showMessage("Aluno salvo");
            } else
                view.showMessage("Matrícula já existente");
        } catch (NumberFormatException e) {
            view.showMessage("Matrícula inválida");
        }
    });
});
```

PASSO A PASSO - EXCLUSÃO

Do mesmo modo que a inserção, para excluir um aluno devemos utilizar o controller para fazer um intermediário entre nossa fonte de dados e nossa view onde é apresentado as informações.

Vamos começar criando uma função de deletar aluno:

```
// Deleta o aluno na base de dados
public void deleteAluno(int matricula) {
    ArrayList<Aluno> alunos = model.getAlunos();
    for (int i = 0; i < alunos.size(); i++) {
        if (alunos.get(i).getMatricula() == matricula) {
            alunos.remove(i);
            loadAlunos();
            return;
        }
    }
}
```

PASSO A PASSO - EXCLUSÃO

Novamente devemos adicionar um listener para quando o botão for clicado, a função ser chamada. Porém a view que estamos trabalhando agora é a AlunoComponent, para ter acesso a ela temos uma lista de Components dentro da view principal. Que é acessada dessa forma:

```
ArrayList<AlunoComponent> lista = view.getComponentList();
```

Então para cada componente vamos adicionar um escutador.

Desse modo:

```
for (int aux = 0; aux < lista.size(); aux++) {  
    final int i = aux;  
    lista.get(i).addListenerDeleteButton(new ActionListener() {  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            view.showMessage("Aluno "+lista.get(i).getName()+" removido!");  
            deleteAluno(lista.get(i).getMatricula());  
        }  
    });  
}
```

PASSO A PASSO - EDITAR

Por final devemos fazer o mesmo para editar as informações do aluno.

No controller:

```
// Atualiza o modelo de acordo com os dados da view
public void updateAluno(int id, int matricula, String nome) {
    ArrayList<Aluno> alunos = model.getAlunos();

    for (int i = 0; i < alunos.size(); i++) {
        if (alunos.get(i).getMatricula() == id) {
            alunos.get(i).setMatricula(matricula);
            alunos.get(i).setNome(nome);
            loadAlunos();
            return;
        }
    }
}
```

PASSO A PASSO - EDITAR

E no método `setListeners()`:

Desse modo finalizamos a atividade.

Obrigado pela atenção :)

```
for (int aux = 0; aux < lista.size(); aux++) {  
  
    final int i = aux;  
    lista.get(i).addListenerUpdateButton(new ActionListener() {  
        private int id;  
  
        @Override  
        public void actionPerformed(ActionEvent e) {  
  
            if (lista.get(i).getButtonEditar().getText().equals("Editar")) {  
  
                this.id = lista.get(i).getMatricula();  
  
                lista.get(i).getButtonEditar().setText("Salvar");  
                lista.get(i).setBackground(Color.CYAN);  
  
                // Ativa o formulário do componente  
                lista.get(i).setEnableComponents(true);  
  
            } else if (lista.get(i).getButtonEditar().getText().equals("Salvar")) {  
  
                lista.get(i).getButtonEditar().setText("Editar");  
                lista.get(i).setBackground(lista.get(i).getParent().getBackground());  
  
                // Atualiza as informações do aluno na base de dados  
                updateAluno(id, lista.get(i).getMatricula(), lista.get(i).getName());  
  
                // Desativa o formulário do componente  
                lista.get(i).setEnableComponents(false);  
            }  
        }  
    });  
}
```

Atividade desenvolvida pelos estudantes:

**André Vinicius, Ana Luiza, Lucas Gabriel, Paulo Lopes, Pedro Henrique, Raphael
de Oliveira**
LLP2 - 2016