

Um Sistema para Geração de Interfaces Ricas em JAVA

Cesar Perdomo Purper

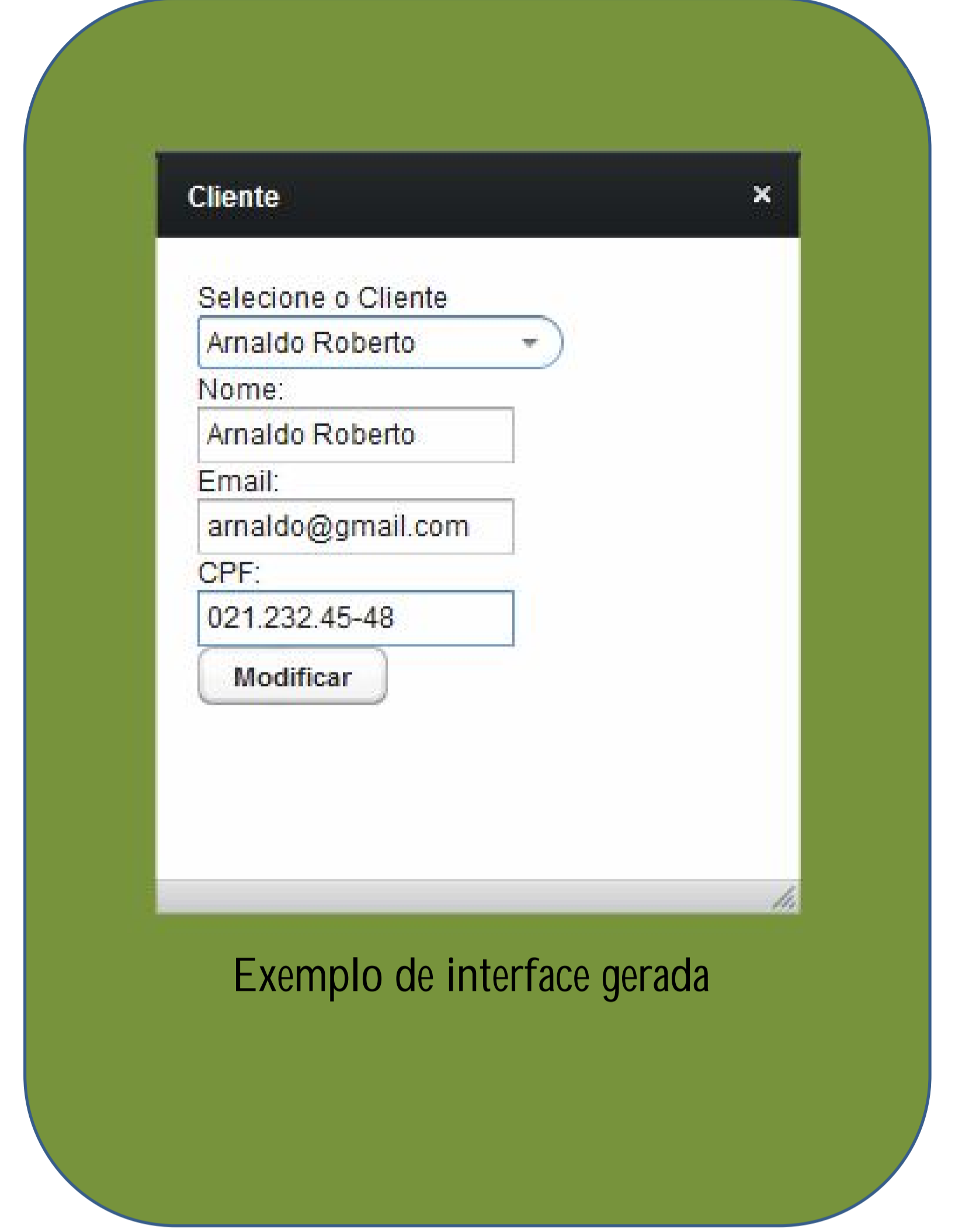
Orientado por Rodrigo Prestes Machado

Motivação

O desenvolvimento de interfaces interativas e que executem operações de banco é uma tarefa trabalhosa, em função da grande quantidade de codificação e de testes necessários para a conclusão. Através dessa necessidade, percebe-se uma oportunidade de desenvolver uma ferramenta cujo intuito é auxiliar o programador a gerar uma interface previamente testada e com uma menor incidência de erros.

Objetivos

Desenvolver um sistema capaz de gerar código com interface RIA (*Rich Internet Application*). **E o que é uma RIA?** Aplicações RIA, são aplicações com características de aplicações *Desktops*. Como objetivo secundário, o projeto proporciona aos alunos envolvidos um aprendizado e familiarização com o *framework* para criação de interfaces RIA chamada Vaadin, com o uso de anotações e reflexão em Java e também com um *framework* para persistência de dados o Hibernate. Tecnologias que geralmente estão fora da grade curricular.



Etapa 1

Cada atributo de uma classe recebe uma anotação Java que indica o tipo de componente gráfico a ser inserido na interface

```

public class AboutDia
protected CardLayout
protected JButton mC
protected JPanel mMa

public AboutDialog(JF
super(owner);
setModal(true);
setUndecorated(true);
initUI();
ected void

```

```
@Gui(widget=Widget.TEXT_FIELD)
private String nome;
```

Anotação usada para identificar o atributo nome, como *TextField* na interface.

Etapa 2

As informações como: classe anotada, *template* escolhido, nome do pacote, etc. são agrupadas por meio de um *wizard*.



Template

Name	Type	Access
accessibleContext	class java.accessibility.Accessible	Read-Only
action	interface java.swing.Action	Read-Write
actionCommand	class java.lang.String	Read-Write
actionListener	class java.awt.event.ActionListener	Read-Only
actionMap	class java.swing.ActionMap	Read-Write
alignment	enum	Read-Write
ancestorListener	class java.awt.event.AncestorListener	Read-Only
autoScroll	boolean	Read-Write
background	class java.awt.Color	Read-Write
baseline	enum	Read-Only
baselineResize	class java.awt.ComponentBaseline	Read-Only
border	interface java.swing.border.Border	Read-Write
borderPainted	boolean	Read-Write
ChangeListener	class java.awt.event.ChangeListener	Read-Only
component	interface	No Access
componentCount	int	Read-Only
componentPopupMenu	class java.awt.event.PopupMenu	Read-Write
componentListeners	class java.awt.event.ComponentListener	Read-Only
containerListener	class java.awt.event.ContainerListener	Read-Only
containerListeners	class java.awt.event.ContainerListeners	Read-Write
containerListeners	class java.awt.event.ContainerListeners	Read-Write
defaultButton	boolean	Read-Only

Etapa 3

O gerador recebe as informações do *wizard* e constrói a classe usando o *framework* "Vaadin". Esta classe respeita a interface anotada pelo usuário e é capaz de realizar operações básicas com o banco de dados (inserir, listar, alterar e excluir)



É o ponto de partida para a geração da classe, também exerce a função de modelo da interface a ser gerada. Deve ser escolhida pelo usuário.