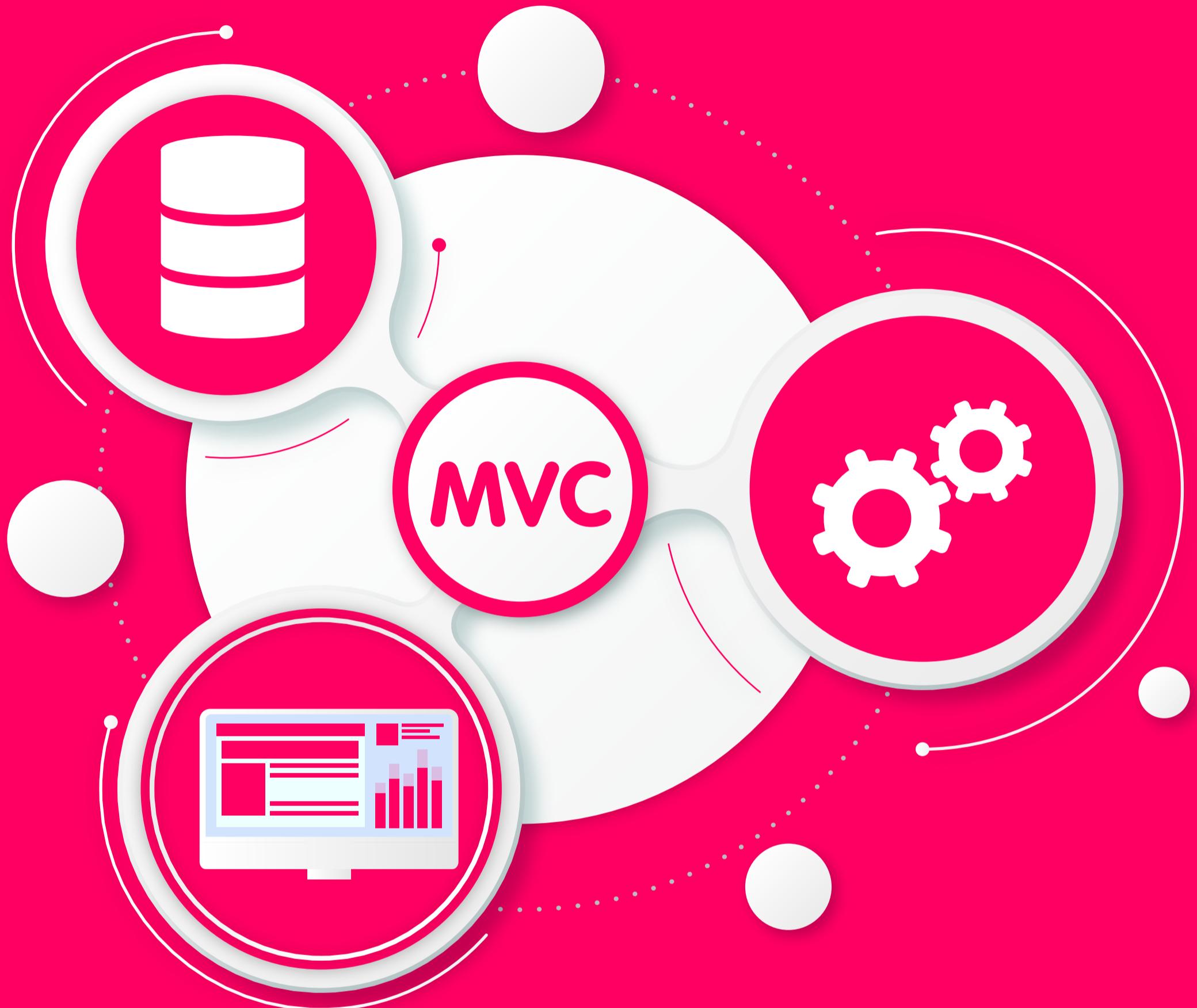




El futuro digital
es de todos

MinTIC



Arquitectura MVC

Parte 2

Hola:

Una vez conocemos los conceptos clave de la arquitectura MV y con la claridad sobre la capa de vista y la capa de controlador, es hora de implementar la capa de modelo. Vamos a ver la implementación en Java de esta capa.

Video de integración de la capa de modelo en la arquitectura MVC.



```
package Model;

/**
 *
 * @author USUARIO
 */
public class modelDog {
}
```

```
modelDog >
: x
```

```
package Model;

/**
 *
 * @author USUARIO
 */
public class modelCat {
```

The screenshot shows the NetBeans IDE interface with the project 'PetManager' open. The code editor displays the `ctlPet.java` file, which contains Java code for a controller class. The code includes imports for `java.util.List`, `java.util.ArrayList`, and `java.util.LinkedList`. It defines a class `ctlPet` with methods for creating pets of different types (Gato and Perro) by calling methods on `modelDog` and `modelCat` respectively. The code editor has syntax highlighting and line numbers. The Navigator pane shows the members of the `ctlPet` class. The Output pane is empty.

```
15 public class ctlPet {  
16  
17     private modelDog modelDog;  
18     private modelCat modelCat;  
19     public ctlPet() {  
20         this.modelDog = new modelDog();  
21         this.modelCat = new modelCat();  
22     }  
23  
24     public boolean CreatePet(clsPet pet) {  
25         try {  
26             switch (pet.getAnimalType()) {  
27                 case "Gato":  
28                     break;  
29                 case "Perro":  
30                     this.modelDog.CreatePet(pet);  
31                     break;  
32             }  
33         }  
34         return true;  
35     }  
36 }
```

The screenshot shows the NetBeans IDE interface with the project 'PetManager' open. The code editor displays the `frmPet.java` file, which contains Java code for a form class. The code imports `javax.swing.DefaultListModel` and `javax.swing.JOptionPane`. It defines a class `frmPet` that extends `JFrame`. The constructor initializes components and sets the controller to `ctlPet`. The code editor has syntax highlighting and line numbers. The Navigator pane shows the members of the `frmPet` class. The Output pane is empty.

```
11 import javax.swing.DefaultListModel;  
12 import javax.swing.JOptionPane;  
13  
14 /**  
15 * @author USUARIO  
16 */  
17 public class frmPet extends javax.swing.JFrame {  
18  
19     ctlPet ctlPet;  
20     LinkedList<clsDog> dogObjectList = new LinkedList<>();  
21  
22     /**  
23      * Creates new form frmPet  
24      */  
25     public frmPet() {  
26         initComponents();  
27         this.ctlPet = new ctlPet();  
28     }  
29  
30     /**  
31      * This method is called from within the constructor to initialize the form.  
32      * WARNING: Do NOT modify this code. The content of this method is always  
33      */  
34 }
```

The screenshot shows the NetBeans IDE interface with the title "PetManager - NetBeans IDE 8.2". The left sidebar displays the project structure under "PetManager" with packages like "Source Packages", "Controller", "Model", and "View". The main editor window shows Java code for the "frmPet.java" file. The code implements action listeners for buttons related to searching and editing dogs. The "Output" panel at the bottom right is empty.

```
387     }
388 }
389
390 private void btnBuscarDogActionPerformed(java.awt.event.ActionEvent evt) {
391     String code = txtCodeDog.getText();
392     clsDog dog = (clsDog) ctlPet.SearchPet(code, "Perro");
393     if (dog == null) {
394         JOptionPane.showMessageDialog(this, "Code not found");
395     } else {
396         txtNameDog.setText(dog.getName());
397         txtColorDog.setText(dog.getColor());
398         txtBornYearDog.setText(dog.getBorn_year() + "");
399         cbHealthStatusDog.setSelectedItem(dog.getHealth_status());
400         cbBreedDog.setSelectedItem(dog.getBreed());
401         cbPedigree.setSelected(dog.isPedigree());
402     }
403 }
404
405 private void btnEditarDogActionPerformed(java.awt.event.ActionEvent evt) {
406     String code = txtCodeDog.getText();
407     boolean found = false;
408     for (clsDog dog : dogObjectList) {
409         if (dog.getCode().equals(code)) {
410             found = true;
411             break;
412         }
413     }
414     if (!found) {
415         JOptionPane.showMessageDialog(this, "Code not found");
416     } else {
417         txtNameDog.setText(dog.getName());
418         txtColorDog.setText(dog.getColor());
419         txtBornYearDog.setText(dog.getBorn_year() + "");
420         cbHealthStatusDog.setSelectedItem(dog.getHealth_status());
421         cbBreedDog.setSelectedItem(dog.getBreed());
422         cbPedigree.setSelected(dog.isPedigree());
423     }
424 }
```

Con el conocimiento para desarrollar una aplicación que puede ser administrada mediante la interfaz gráfica soportada por la separación de responsabilidades a través de la arquitectura MVC, los invitamos a practicar con el reto de la semana.



Universidad de Caldas