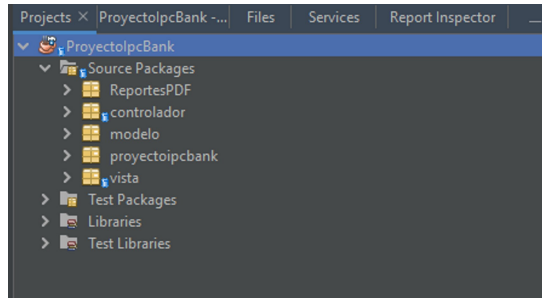


MANUAL TECNICO

miércoles, 7 de septiembre de 2022 12:09

En este documento se detalla como se desarrolló el proyecto 1 IPC-BANK. Por cada procedimiento se adjunta una imagen como referencia.

El proyecto se desarrolló en lenguaje Java utilizando Apache NetBeans IDE 12.5. Se utiliza el patrón de diseño MVC (Modelo, Vista, Controlador) , para facilitar el uso de objetos y mantener el orden dentro del proyecto.



En el package "vista" se encuentran todas las ventanas o JFrame utilizadas en el proyecto. En el package "modelo" se encuentran los métodos y atributos para cada ventana empleada en el proyecto. En el package "controlador" están los métodos que dan acceso a los datos y controlan el funcionamiento del proyecto. En el package "ReportesPDF" se maneja el método que crea los reportes. En el package "proyectoipcbank" se encuentra el método main, en este se inicializan todos los arreglos utilizados, también se invocan los constructores para cada clase creada en los paquetes mencionados anteriormente.

CONTROLADOR:

Para ejemplo se usará la clase contMenu. Dentro del Package controlador están todos los controladores de acción del proyecto. Dentro de la clase contMenu se declaran los objetos que se utilizarán dentro de esta clase, se realizan las importaciones y se inician dentro del constructor contMenu, también se implementan los escuchadores para los click o para las acciones en la apertura de ventanas.

```

2  package controlador;
3
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6  import vista.FrmCuentas;
7  import vista.FrmHistorial;
8  import vista.FrmMenu;
9  import vista.FrmRegistroU;
10 import vista.FrmReportes;
11
12 import vista.FrmRetirosD;
13
14 public class contMenu implements ActionListener {
15     //Objetos de las ventanas principales
16     FrmMenu vMnu = new FrmMenu();
17     FrmRegistroU vReg = new FrmRegistroU();
18     FrmCuentas vCta = new FrmCuentas();
19     FrmRetirosD vRet = new FrmRetirosD();
20     FrmReportes vRep = new FrmReportes();
21     FrmHistorial vHist = new FrmHistorial();
22
23
24     public contMenu(FrmMenu vMnu, FrmRegistroU vReg, FrmCuentas vCta, FrmRetirosD vRet, FrmReportes vRep, FrmHistorial vHist) {
25         this.vMnu = vMnu;
26         this.vReg = vReg;
27         this.vCta = vCta;
28         this.vRet = vRet;
29         this.vRep = vRep;
30         this.vHist = vHist;
31
32         vMnu.btnCuentas.addActionListener(this);
33         vMnu.btnDepositos.addActionListener(this);
34         vMnu.btnRegistro.addActionListener(this);
35         vMnu.btnReportes.addActionListener(this);
36         vMnu.btnTransacciones.addActionListener(this);
37
38     }
39

```

Funcion de los métodos del controlador usuario:

```

29
30 public void setDatos() {
31     uvo.setCui(vReg.txtCui.getText());
32     uvo.setNombre(vReg.txtNombre.getText());
33     uvo.setApellido(vReg.txtApellido.getText());
34 }
35
36
37 public boolean registrarU(UsuarioVO[] usuarioU) {
38     this.setDatos();
39     boolean validar = false;
40     //udao.buscar(vReg.txtCui.getText(), usuarioU);
41     if(udao.buscar(vReg.txtCui.getText(), usuarioU) == -1){
42         uvo.setContador(uvo.getContador() + 1);
43         UsuarioVO usuario = new UsuarioVO(uvo.getCui(), uvo.getNombre(), uvo.getApellido());
44         udao.insertar(usuario, usuarioU);
45         udao.imprimir(usuarioU);
46         return true;
47     }else{
48         vReg.jopMensaje.showMessageDialog(vReg, "El CUI que desea registrar ya existe");
49     }
50
51     return validar;
52 }
53

```

Método setDatos() :

Se utilizan los objetos de la clase UsuarioVO y los métodos setter declarados para cambiar el valor de los atributos capturando los datos ingresados por el usuario desde la ventana de registro.

Método registrarU():

Se usa el método set datos para llenar los valores luego condicionando que las casillas de la ventana de registro no estén vacías se crea un objeto con el controlador que lleva parametros con los datos del usuario. Luego se llama al método insertar haciendo uso del objeto de la clase

UsuarioDAO para registrar los datos y si este se ejecuta correctamente retorna un boolean true, en case contrario retornará un false.

Click en el botón registrar:

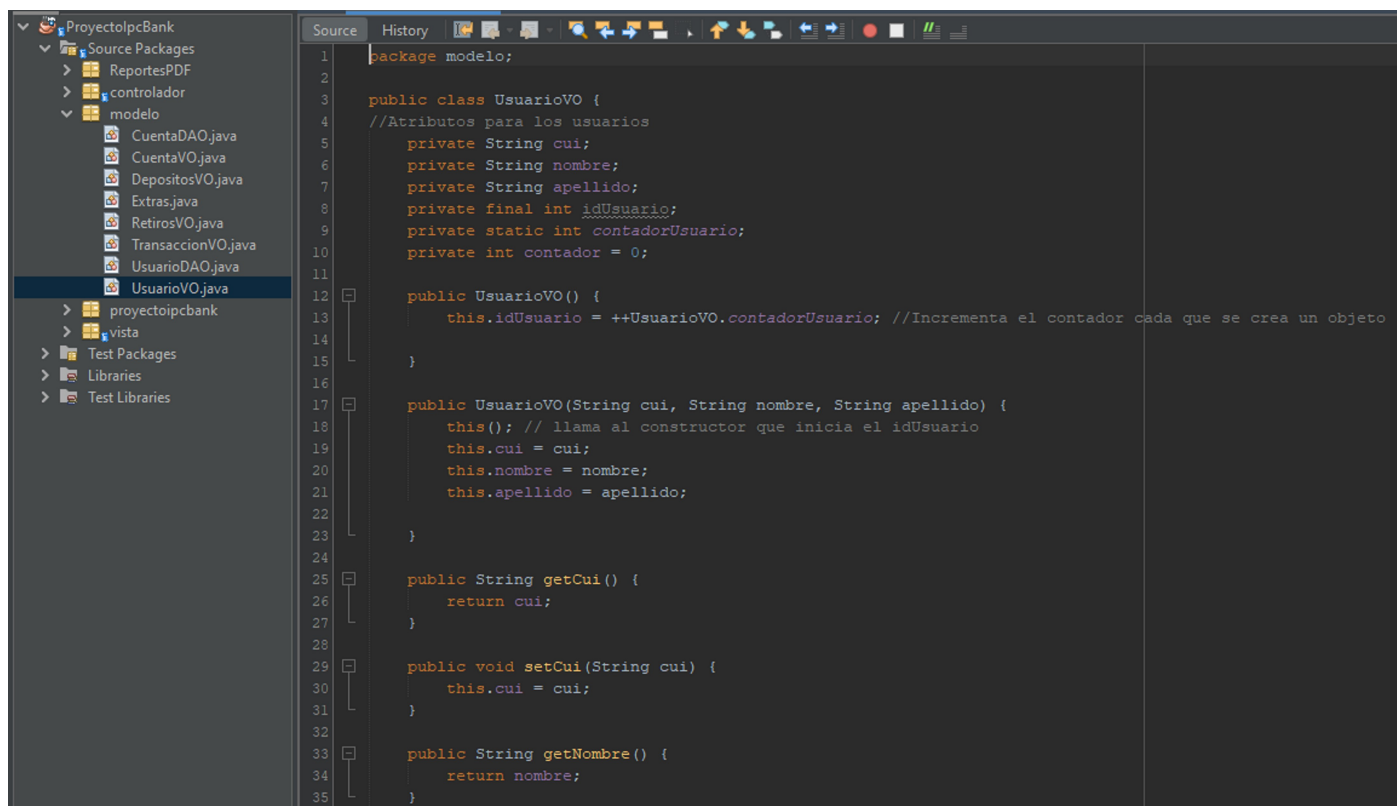
```
80  @Override
81  public void actionPerformed(ActionEvent e) {
82
83      if (e.getSource() == vReg.btnRegistrar) {
84
85          if (this.validaContador() == true) {
86              if (this.camposLlenos() == true) {
87
88                  this.setDatos();
89                  //this.registrarU(this.usuarios);
90                  if (this.registrarU(usuario) == true) {
91                      vReg.jopMensaje.showMessageDialog(vReg, "Usuario: " + uvo.getNombre() + " " + uvo.getApellido() + ", registrado con éxito.");
92                      System.out.println("Contador Usuarios" + uvo.getContador());
93                      this.vaciarCampos();
94                  } else {
95                      this.vaciarCampos();
96                  }
97                  //-----
98
99              } else {
100                  vReg.jopMensaje.showMessageDialog(vReg, "Llene todos los campos");
101              }
102          } else {
103              vReg.jopMensaje.showMessageDialog(vReg, "Usuario no creado. Cantidad máxima, 5 usuarios.");
104          }
105      }
106  }
107  }
```

Al darle click al botón primero se validan los retornos de los métodos para ver si se lleva a cabo la acción del registro. Cualquier caso, registro o no, se comprobará a la vista del usuario con un mensaje emergente JOptionPane indicando la acción que realizó el programa.

MODELO:

Dentro del package modelo se encuentran las clases en las que se declaran los atributos de las clases (VO (values object)) y también se encuentran las clases en las que se tienen acceso a los atributos declarados (DAO (data acces object)).

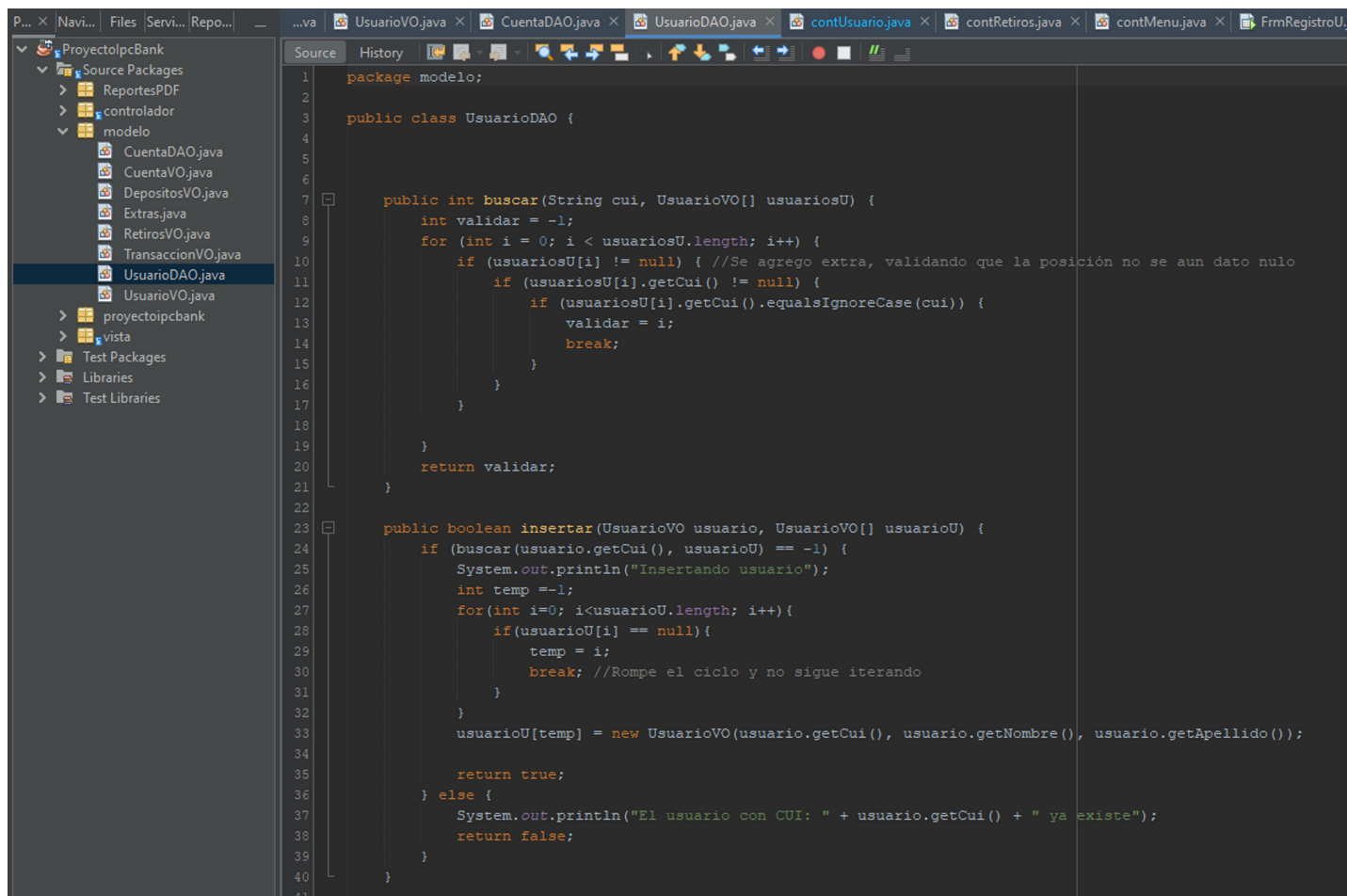
En las clases VO, se declaran los atributos, constructores y métodos getter y setter.



```
1 package modelo;
2
3 public class UsuarioVO {
4     //Atributos para los usuarios
5     private String cui;
6     private String nombre;
7     private String apellido;
8     private final int idUsuario;
9     private static int contadorUsuario;
10    private int contador = 0;
11
12    public UsuarioVO() {
13        this.idUsuario = ++UsuarioVO.contadorUsuario; //Incrementa el contador cada que se crea un objeto
14    }
15
16    public UsuarioVO(String cui, String nombre, String apellido) {
17        this(); // llama al constructor que inicia el idUsuario
18        this.cui = cui;
19        this.nombre = nombre;
20        this.apellido = apellido;
21    }
22
23    public String getCui() {
24        return cui;
25    }
26
27    public void setCui(String cui) {
28        this.cui = cui;
29    }
30
31    public String getNombre() {
32        return nombre;
33    }
34
35 }
```

En las clases DAO, se crean los métodos para darle uso y acceso a los valores de los objetos.

Métodos como buscar, insertar, actualizar.



```
1 package modelo;
2
3 public class UsuarioDAO {
4
5     public int buscar(String cui, UsuarioVO[] usuariosU) {
6         int validar = -1;
7         for (int i = 0; i < usuariosU.length; i++) {
8             if (usuariosU[i] != null) { //Se agrego extra, validando que la posición no se aun dato nulo
9                 if (usuariosU[i].getCui() != null) {
10                     if (usuariosU[i].getCui().equalsIgnoreCase(cui)) {
11                         validar = i;
12                         break;
13                     }
14                 }
15             }
16         }
17         return validar;
18     }
19
20     public boolean insertar(UsuarioVO usuario, UsuarioVO[] usuarioU) {
21         if (buscar(usuario.getCui(), usuarioU) == -1) {
22             System.out.println("Insertando usuario");
23             int temp = -1;
24             for(int i=0; i<usuarioU.length; i++){
25                 if(usuarioU[i] == null){
26                     temp = i;
27                     break; //Rompe el ciclo y no sigue iterando
28                 }
29             }
30             usuarioU[temp] = new UsuarioVO(usuario.getCui(), usuario.getNombre(), usuario.getApellido());
31             return true;
32         } else {
33             System.out.println("El usuario con CUI: " + usuario.getCui() + " ya existe");
34             return false;
35         }
36     }
37
38 }
```

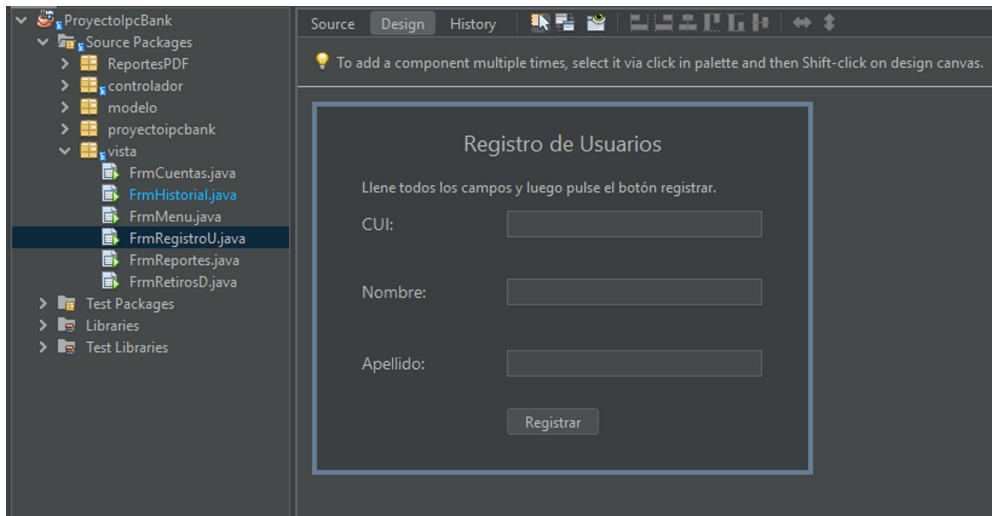
Función de los métodos en DAO:

El método buscar recibe dos parametros para la búsqueda, un parametro tipo String Cui ingresado por el usuario y un arreglo tipo UsuarioVO. Con un ciclo for se recorre el arreglo y si en las posiciones que no están vacías encuentra una igualdad con un Cui registrado y el que intenta registrar el usuario cambia el valor de la variable local "validar" y retorna dicho número.

El método insertar recibe dos parametros, uno de tipo UsuarioVO y un arreglo tipo UsuarioVO. Se ejecuta el método siempre y cuando el retorno del método buscar sea diferente -1, esto quiere decir que el método buscar no ha encontrado igualdades en la búsqueda. El método insertar con un ciclo for recorre el arreglo y cuando la posición está vacía agrega los datos para el registro del usuario.

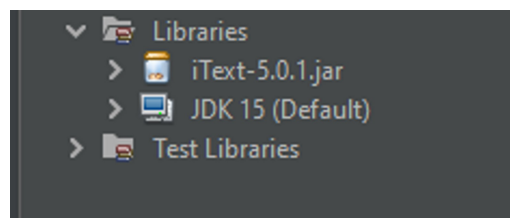
VISTA:

En el package vista se encuentran las ventanas utilizadas en el proyecto, todos los textfield, botones, tablas utilizadas en ellas se colocaron en código publico para poder acceder y darle uso en los controladores.



LIBRERÍA Itext:

Se usó la librería Itext para generar los reportes.



Para generar los reportes primero se utilizó un método que hace uso de la sintaxis HTML para agregar una tabla y los datos en ella.

```

29 public void generarHtml(CuentaVO cuentas[]) {
30
31     String nombreReporte;
32     File reporte;
33     FileWriter fw;
34     BufferedWriter br;
35     String cadenaHTML;
36
37     try {
38         nombreReporte = "Usuarios Registrados.html";
39         reporte = new File(nombreReporte);
40         fw = new FileWriter(reporte);
41         br = new BufferedWriter(fw);
42
43         cadenaHTML = "<html>"
44             + "    <head>"
45             + "    <body>"
46             + "    <title>Reporte de Usuarios Registrados</title><br>"
47             + "    <table border = 2>"
48             + "        <tr style=\"border-color: blue;\">"
49             + "            <td>Nombre</td>"
50             + "            <td>Cui</td>"
51             + "            <td>Saldo Actual</td>"
52             + "        </tr>";
53
54         for (int x = 0; x < cuentas.length; x++) {
55             if (cuentas[x] != null) {
56                 cadenaHTML += "            <tr>"
57                     + "                <td> " + cuentas[x].getNombreC() + "</td>"
58                     + "                <td> " + cuentas[x].getCuiC() + "</td>"
59                     + "                <td> " + cuentas[x].getSaldoC() + "</td>"
60                     + "            </tr>";
61             }
62         }
63
64         cadenaHTML += "        </table>"
65             + "    </body>"
66             + "</html>";
67
68         br.write(cadenaHTML);
69
70         br.close();
71         fw.close();
72
73         crearPdf(cadenaHTML);
74
75     } catch (IOException ex) {
76         System.out.println("error escribiendo el reporte. Detalles " + ex.getMessage());
77     }
78
79 }

```

Anterior a este método debe crearse un método que reciba un parametro String y este método convierte la página html en un pdf

```

80
81 public void crearPdf(String html) {
82     try {
83
84         Document document = new Document(PageSize.LETTER) {
85         };
86         PdfWriter.getInstance(document, new FileOutputStream("ReporteUsuarios_" + Extras.fechaReporte() + ".pdf"));
87
88         document.open();
89         document.addAuthor("IPC1 E");
90         document.addCreator("IPC1 E");
91         document.addSubject("Ejemplo Reporte PDF");
92         document.addCreationDate();
93         document.addTitle("Reporte pdf");
94
95         HTMLWorker htmlWorker = new HTMLWorker(document);
96         htmlWorker.parse(new StringReader(html));
97
98         document.close();
99
100     } catch (Exception e) {
101         e.printStackTrace();
102     }
103 }

```

Cabe destacar que todo el proyecto se basó en modelo vista controlador puesto que las

funcionalidades son similares para cada ventana y acción en ellas.

Estos son los detalles de funcionamiento del proyecto1 IpcBank.