

Lenguaje de programación I



Caso



- ✓ El administrador de la empresa CiberFarma, necesita un reporte (**listado**) de los usuarios del sistema. Diseñe el formulario y los procesos de Gestión



The screenshot shows a software window titled "Listado de Usuarios". Inside the window, there is a table with two columns: "Código" and "Nombre". The table contains four rows of data. Below the table, there is a button labeled "Reporte".

Código	Nombre
1	Tito
2	Zoila
3	BRYAN
4	Bill

Reporte

Contenido

- Uso de Bases de datos
- Ejercicio: Listados
- Aplicaciones

Logros de la Unidad

- Crear aplicaciones de manejo de bases de datos



Las consultas o reportes

- Estos procesos involucran la ejecución de sentencias **SELECT**, obteniendo un objeto **ResultSet** que representa el resultado de la consulta.
- Ej. De consulta a la tabla productos:

SELECT * FROM tb_productos;

rs =

idprod	descripcion	precio	estado
P0001	Pizza familiar	35.00	1
P0002	Pizza suprema	45.00	1
P0003	Pizza personal	8.00	1
P0004	Inca Kola	2.50	1
NULL	NULL	NULL	NULL

Importante.!!!

Para almacenar los datos, usaremos colecciones. Ej **ArrayList** de la Clase.

Las consultas o reportes

- Para obtener los datos de cada registro, usaremos métodos de acceso get.
- Ej. Para obtener el código y el precio

rs =

`rs.getString(1)`

`rs.getDouble(3)`

	idprod	descripcion	precio	estado
	P0001	Pizza familiar	35.00	1
	P0002	Pizza suprema	45.00	1
	P0003	Pizza personal	8.00	1
	P0004	Inca Kola	2.50	1
*	NULL	NULL	NULL	NULL

Solución. Caso



- ✓ El administrador de la empresa Ciberfarma necesita un reporte (listado) de los usuarios del sistema. Diseñe el formulario y los procesos de Gestión



The screenshot shows a software window titled "Listado de Usuarios". Inside the window, there is a table with two columns: "Código" and "Nombre". The table contains four rows of data. Below the table, there is a button labeled "Reporte".

Código	Nombre
1	Tito
2	Zoila
3	BRYAN
4	Bill

Reporte

Estructura del Proyecto con BD

▼ LPI-semana07-plantilla

▼ src

▼ interfaces

> UsuarioInterface.java

▼ mantenimientos

> GestionUsuario.java

▼ model

> Usuario.java

▼ utils

> MySQLConexion.java

▼ vista

> Reporte.java

> Reporte2.java

> ReportexFecha.java

> JRE System Library [JavaSE-1.8]

> jcalendar-1.4.jar

▼ bd

bd.sql

▼ libs

jcalendar-1.4.jar

mysql-connector-java-5.1.6-bin.jar

1

Listado de métodos para el mantenimiento

3

Implementación de los métodos de mantenimiento

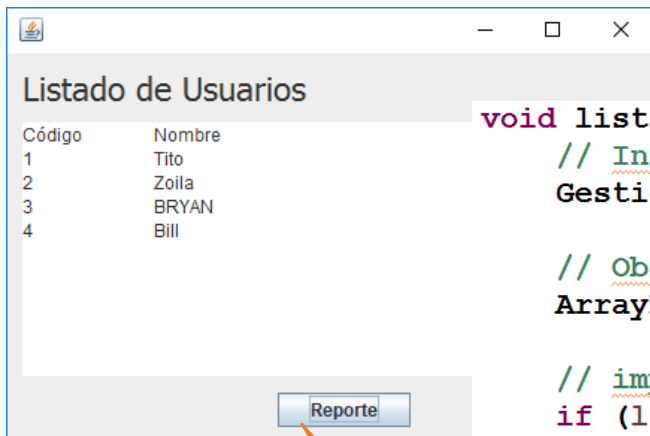
2

Entidades utilizadas para transferir los datos desde la base de datos hacia la capa lógica y viceversa

GUI



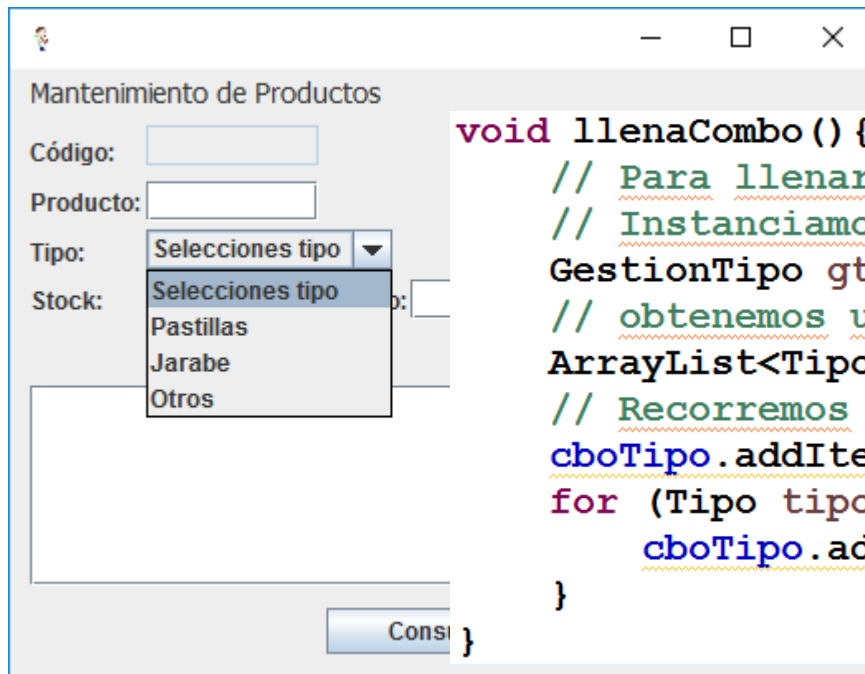
- ✓ El proceso de reporte mostrará el listado obtenido de la clase de Gestión



```
void listado() {  
    // Instancia la clase de gestion  
    GestionUsuario gu = new GestionUsuario();  
  
    // Obtiene el listado de usuarios -> ArrayList  
    ArrayList<Usuario> lista = gu.listado();  
  
    // imprime la lista en el área de texto  
    if (lista == null) {  
        txtS.setText("Lista vacía");  
    } else {  
        txtS.setText("Código\tNombre\n");  
        for (Usuario u : lista) {  
            txtS.append(u.getCodigo() + "\t" + u.getNombre() + "\n");  
        }  
    }  
}
```


Ejemplo. Uso de listados en ComboBox

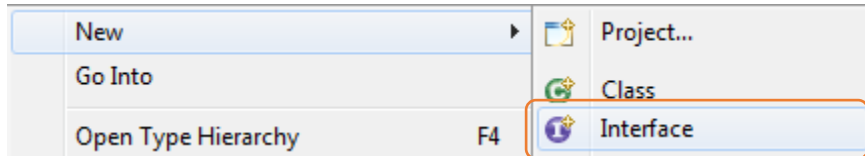
- ✓ Realizar listados nos permite emplearlo en diversos elementos como ComboBox.
- ✓ Ej. Para llenar el combo de categorías (tipos) de productos en una farmacia:



```
void llenaCombo() {  
    // Para llenar un combo de la BD  
    // Instanciamos la clase de Gestión  
    GestionTipo gt = new GestionTipo();  
    // obtenemos una lista con los datos de la tabla  
    ArrayList<Tipo> lista = gt.listado();  
    // Recorremos el arreglo para agregar los elementos  
    cboTipo.addItem("Seleccione tipo");  
    for (Tipo tipo : lista) {  
        cboTipo.addItem(tipo.getDescripcion());  
    }  
}
```

1. Interfaces

- ✓ La interface, **define** los métodos de mantenimiento a implementar (registrar, actualizar, eliminar, listar, etc.) Ej. Para listado de Usuarios



- Ej. Para listado de Usuarios



```
UsuarioInterface.java
1 package interfaces;
2
3 public interface UsuarioInterface {
4
5     // lista el contenido de la tabla usuario
6     public ArrayList<Usuario> listado();
7
8     // lista el contenido de la tabla usuario,
9     según el tipo
10    public ArrayList<Usuario> listado(int tipo);
11
12 }
```

2. Entidades

- los listados, toman como referencia toda la tabla o consultas de donde obtendremos la información.
- Ej: Para el listado de toda la tabla **usuarios**:



```
SELECT * FROM tb_usuarios;
```

codigo	nombre	apellido	usuario	clave	facceso	tipo	estado
1	Tito	Siber	U001	10001	2017-06-03	2	1
2	Zoila	Baca	U002	10002	2017-06-03	2	1

Usuario.java

```
package beans;

public class Usuario {
    private int codigo, tipo, estado;
    private String nombre, apellido, usuario, clave, facceso;
```

6

- Ej: Para un listado de las **ventas** mostrando los **productos**:

```
SELECT numvta, fechavta, v.idprod, p.descripcion as 'Producto',
       (v.cantidad * v.preciovta) as 'importe'
FROM tb_ventas v
INNER JOIN tb_productos p ON v.idprod = p.idprod;
```

numvta	fechavta	idprod	Producto	importe
V0001	2017-04-15	P0001	Panadol cj 10	1.50
V0001	2017-04-15	P0004	Achiz	2.00
V0002	2017-04-25	P0002	Curitas unidad	1.00

Reporte1.java

```
1 package beans;
2
3 public class Reporte1 {
4     private String numvta, fechavta, idprod, nomprod;
5     private double importe;
```

Clase de Control o Gestión

- Estas clases **implementarán** los métodos para el mantenimiento.
- Ej. Para la **gestión de usuarios**, se implementa la **Interface**

```
1 package mantenimientos;
2
3 import java.util.ArrayList;
4
5
6
7
8 public class GestionUsuarios implements UsuarioInterface {
9
10     @Override
11     public int validaUsuario() {
12         // TODO Auto-generated method stub
13         return null;
14     }
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

1 method to implement:
- interfaces.UsuarioInterface.validaUsuario()

@Override
public ArrayList<Usuario> listado() {
 // TODO Auto-generated method stub
 return null;
}

• Cada método deberá implementar las sentencias necesarias de base de datos

Clase de Control o Gestión



```
public ArrayList<Usuario> listado() {  
    ArrayList<Usuario> lista = new ArrayList<Usuario>();  
    ResultSet rs = null;  
    Connection con = null;  
    PreparedStatement pst = null;  
    try {  
        con = MySQLConexion.getConnection();  
        String sql = "select * from tb_usuarios"; // sentencia sql  
  
        pst = con.prepareStatement(sql);  
        // parámetros según la sentencia  
  
        rs = pst.executeQuery(); // tipo de ejecución  
        // Acciones adicionales en caso de consultas  
        while (rs.next()) { // mientras hay datos en la consulta  
            Usuario u = new Usuario();  
            u.setCodigo(rs.getInt(1));  
            u.setNombre(rs.getString(2));  
            // otros campos  
            lista.add(u);  
        }  
    } catch (Exception e) {  
        System.out.println("Error en la sentencia " + e.getMessage());  
    } finally {  
        try {  
            if (pst != null) pst.close();  
            if (con != null) con.close();  
        } catch (SQLException e) {  
            System.out.println("Error al cerrar ");  
        }  
    }  
    return lista;  
}
```

✓ El método **next()** del **ResultSet**, hace que el puntero avance al siguiente registro. Si lo consigue devuelve **true**.

	codigo	nombre	apellido	usuario	clave	facceso	tipo	estado
	1	Tito	Siber	U001	10001	2017-05-13	2	1
	2	Zoila	Baca	U002	10002	2017-05-13	2	1



Sentencia y parámetros

- Por ejemplo:
- Sentencia en MySQL:

```
select * from tb_usuarios where usuario = 'U001' and clave = '10001';
```

- Sentencia en clase mantenimiento:

```
sql = "select * from tb_usuarios where usuario = ? and clave = ?";
```

- Cada ? Representa un parámetro (empezando desde 1) el cual se reemplazará con los valores respectivos, Ej:

```
public Usuarios validaUsuario(String usuario, String clave) {  
  
    String sql = "select * from tb_usuarios where usuario = ? and clave = ?";  
  
    pst = con.prepareStatement(sql);
```

```
    pst.setString(1, usuario);  
    pst.setString(2, clave);
```

Para otros tipos de dato usaremos. Ej:

```
pst.setInt(3, cantidad);  
pst.setDouble(4, precio);
```

Referencias

- ✓ <https://www.youtube.com/watch?v=uUzQewzCRiU>
- ✓ <https://hashblogeando.wordpress.com/2014/02/12/mostrar-datos-de-una-tabla-mysql-con-java-swing/>



GRACIAS



SEDE MIRAFLORES

Calle Díez Canseco Cdra 2 / Pasaje Tello
Miraflores – Lima
Teléfono: 633-5555

SEDE INDEPENDENCIA

Av. Carlos Izaguirre 233
Independencia – Lima
Teléfono: 633-5555

SEDE BREÑA

Av. Brasil 714 – 792
(CC La Rambla – Piso 3)
Breña – Lima
Teléfono: 633-5555

SEDE TRUJILLO

Calle Borgoño 361
Trujillo
Teléfono: (044) 60-2000

SEDE SAN JUAN DE LURIGANCHO

Av. Próceres de la Independencia 3023-3043
San Juan de Lurigancho – Lima
Teléfono: 633-5555

SEDE SAN MIGUEL

Av. Federico Gallese 847
San Miguel – Lima
Teléfono: 632-4900

SEDE BELLAVISTA

Av. Mariscal Oscar R. Benvides 3866 – 4070
(CC Mall Aventura Plaza)
Bellavista – Callao
Teléfono: 633-5555

SEDE AREQUIPA

Av. Porongoche 500
(CC Mall Aventura Plaza)
Paucarpata - Arequipa
Teléfono: (054) 60-3535