

Codificación de módulos del software según requerimientos del proyecto.

GA7-220501096-AA2-EV01

Centro de formación:



VIRTUAL PHARMACY S. A. S.

YOUR HEALTHY IS FIRST.

NIT 800.149.659 -1

Centro de teleinformática y producción industrial - Regional Cauca

Programa de formación:
Metodologías de Desarrollo de Software.

Presentado por:
Andrés Leonardo Gómez Ríos
Jonathan Hernando Murcia Villazón

Ficha de caracterización:
ADSO 2879658

Instructora:
Astrid Maritza Calvache Chicangana

Servicio Nacional de Aprendizaje -SENA

Agosto – 2024

Tabla de contenido

INTRODUCCION	2
OBJETIVOS.....	2
ALCANCE	2
DESARROLLO	3
Consulta	7
Inserción.....	8
Actualización	9
Eliminación	10
Conclusión	10
Bibliografía	11

INTRODUCCION

En el ciclo del desarrollo de software tenemos una fase crucial donde pasamos de los requerimientos de los diagramas de clases, casos de uso, prototipos, informes y demás diseños, a funcionalidades concretas a través del código. En esta ocasión presentamos la codificación de los módulos de nuestro software para VIRTUAL PHARMACY S.A.S.

OBJETIVOS

Desarrollar el paso a paso del desarrollo utilizando las herramientas de versionamiento cumpliendo con los estándares de codificación del lenguaje JAVA con el cual trabajaremos en esta fase del desarrollo.

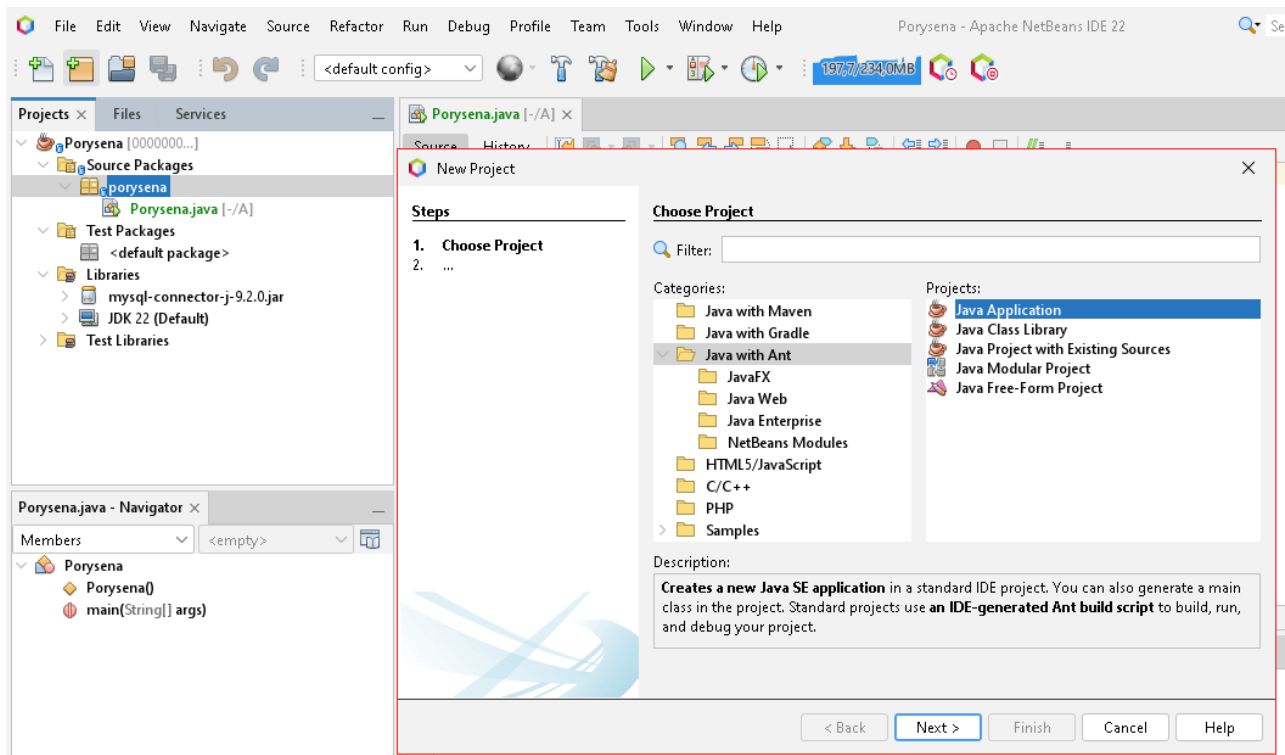
ALCANCE

El alcance de este proyecto abarca la codificación del módulo de Usuarios, desde NetBeans IDE 22 a MySQL Workbench 8.0. basado en los requerimientos establecidos en el análisis funcional y técnico del proyecto. Incluimos el paso a paso del desarrollo por medio de NetBeans IDE 22 y MySQL Workbench 8.0.

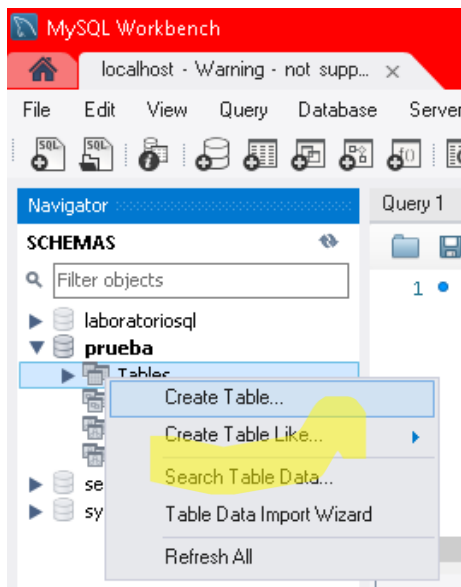
DESARROLLO

Paso a paso para la conexión de la base de datos en Java desde NeatBeans a MySQL Workbech.

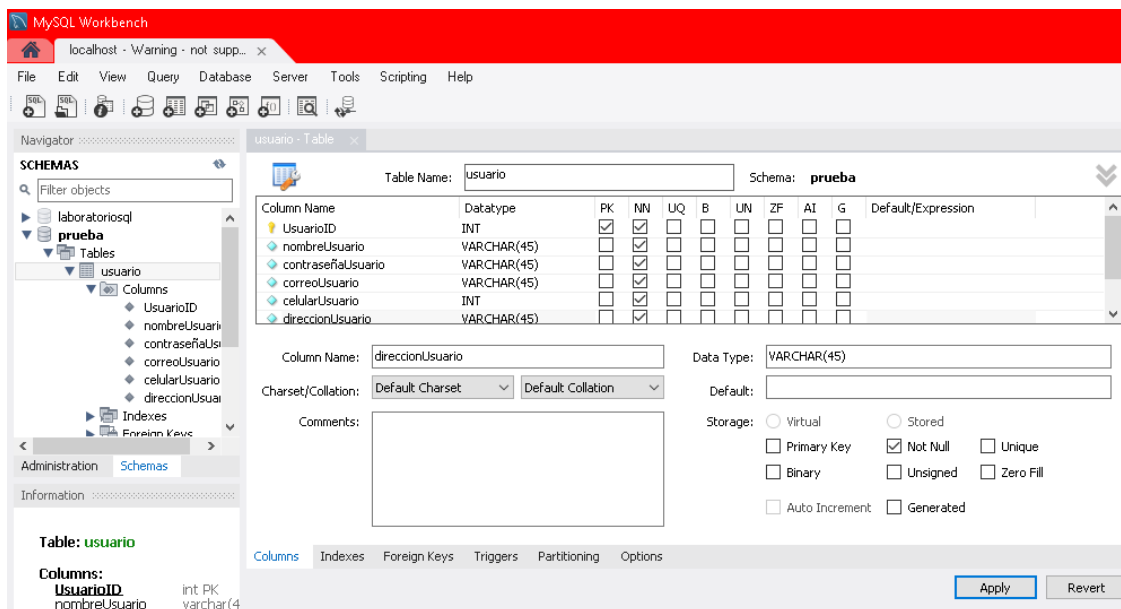
1. Empezamos creando el Proyecto en NetBeans.



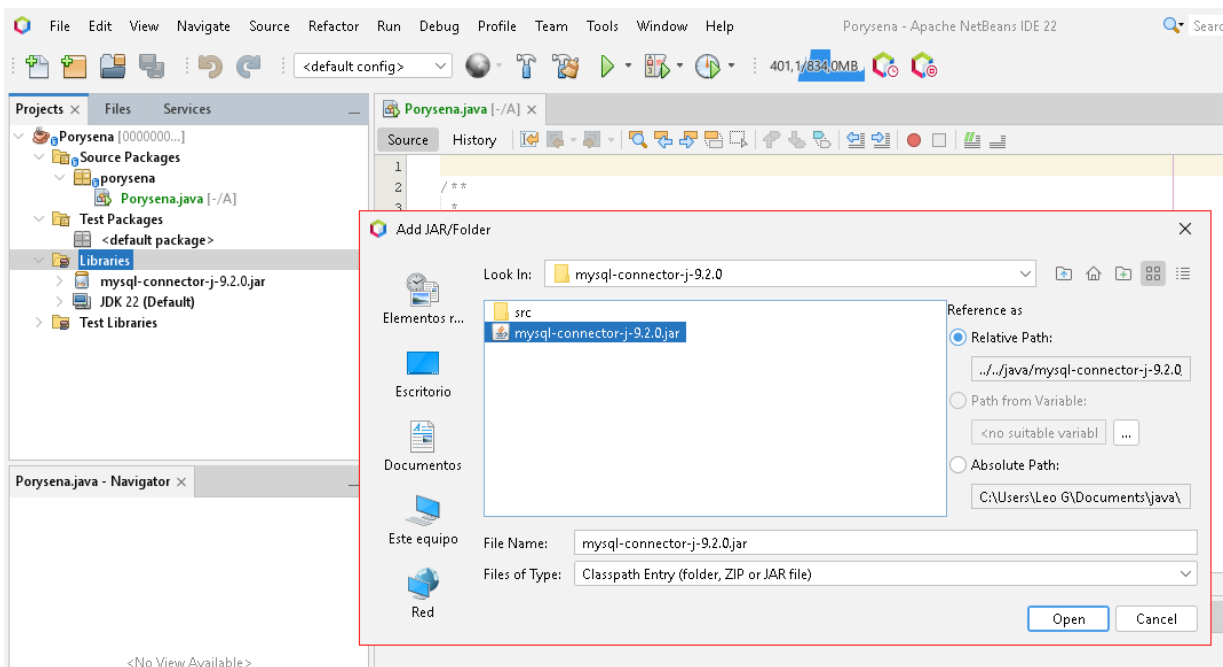
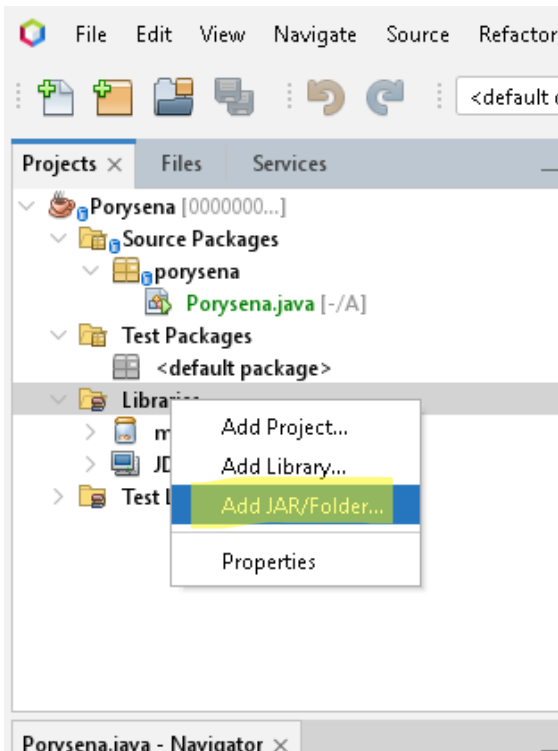
2. Creamos la base de datos y tabla de usuarios en MySQL Workbench.



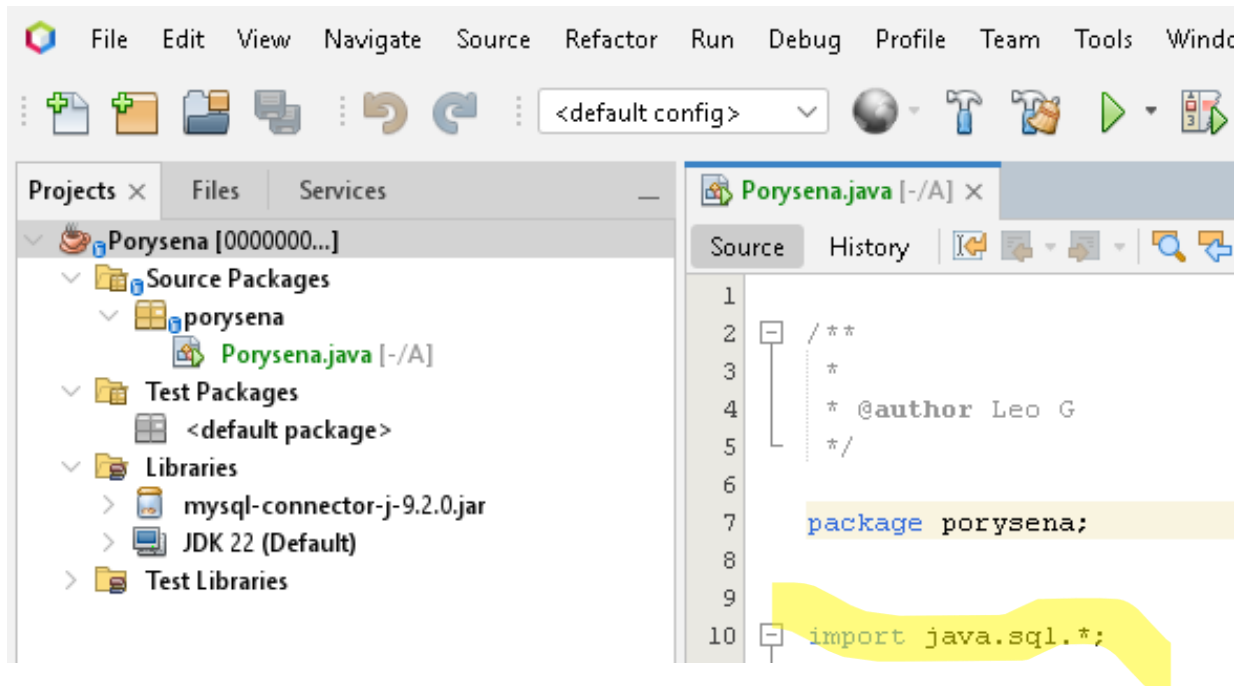
3. Declaramos con camelCase cada el atributo a ingresar de la base de datos en MySQL.



4. Agregamos la librería de mysql-connector en NetBeans para la conexión a la base datos.



5. Importamos el paquete de java.sql en NetBeans con la línea “java.sql*”.



6. Inicializamos el Driver usando la sentencia “Class.forName(“com.mysql.cj.jdbc.Driver”)” y aplicando el try catch.

```
// Establecemos el Class.forName con el try-catch

3      try {
4          Class.forName("com.mysql.cj.jdbc.Driver");
5      } catch (ClassNotFoundException ex) {
6          Logger.getLogger(Porysena.class.getName()).log(Level.SEVERE, null, ex);
7      }
8  }
```

7. Declaramos las variables para la conexión a la base de datos con los objetos Statement y Connection.

```
5
6 public static void main(String[] args) {
7
8     // Delcaracion de las variables para la base de datos
9
10    String usuario="root";
11    String password="Atento12++";
12    String url="jdbc:mysql://localhost:3306/prueba";
13
14    // Declaracion de los objetos para la base de datos
15
16    Connection conexion;
17    Statement st;
18    ResultSet rs;
19
```

8. Establecemos la conexión con la base de datos por medio del getConnection.

```
39
40 try {
41     conexion=DriverManager.getConnection(url,usuario,password);
42
43     st=conexion.createStatement();
44
```

Consulta

9. Implementamos un ciclo que CONSULTA y solicitamos imprimir los datos de id, nombre y correo ingresados en la base de datos con "executeQuery" en Java.

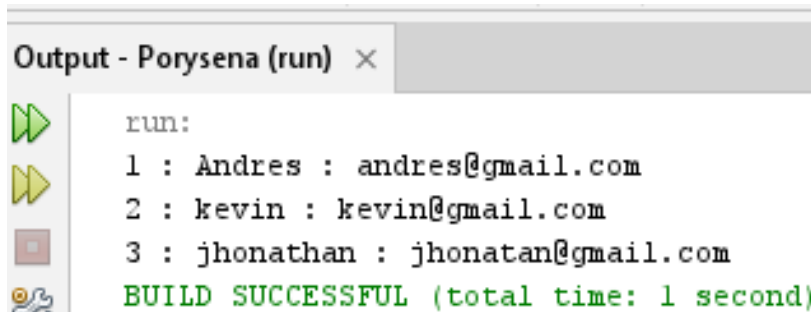
```
// Consultamos usuarios

rs=st.executeQuery("SELECT * FROM usuario");
rs.next();

do {
    System.out.println(rs.getInt("usuarioID") + " : '

} while (rs.next());
```


10. Corremos el proyecto para verificar su funcionamiento.

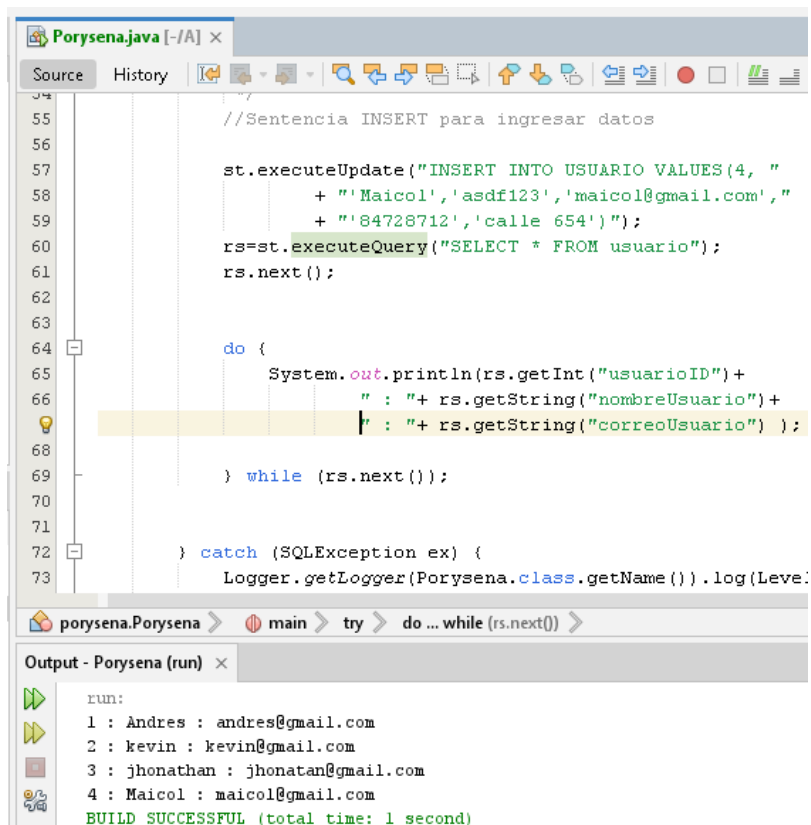


```
Output - Porysena (run) x
run:
1 : Andres : andres@gmail.com
2 : kevin : kevin@gmail.com
3 : jhonathan : jhonatan@gmail.com
BUILD SUCCESSFUL (total time: 1 second)
```

Como NeatBeans arroja los datos de la base podemos corroborar que la conexión es exitosa.

Inserción

11. Hacemos nuestra primera INSERCIÓN de información en la base de datos con executeUpdate.



```
Porysena.java [-/A] x
Source History
55 //Sentencia INSERT para ingresar datos
56
57 st.executeUpdate("INSERT INTO USUARIO VALUES(4, "
58 + "'Maicol','asdf123','maicol@gmail.com',"
59 + "'84728712','calle 654')");
60 rs=st.executeQuery("SELECT * FROM usuario");
61 rs.next();
62
63
64 do {
65     System.out.println(rs.getInt("usuarioID") +
66         " : " + rs.getString("nombreUsuario") +
67         " : " + rs.getString("correoUsuario") );
68 } while (rs.next());
69
70
71 } catch (SQLException ex) {
72     Logger.getLogger(Porysena.class.getName()).log(Level.
73
porysena.Porysena > main > try > do ... while (rs.next()) >
Output - Porysena (run) x
run:
1 : Andres : andres@gmail.com
2 : kevin : kevin@gmail.com
3 : jhonathan : jhonatan@gmail.com
4 : Maicol : maicol@gmail.com
BUILD SUCCESSFUL (total time: 1 second)
```

En este ejemplo insertamos la información de Maicol y vemos reflejada su información al correr el programa y consultar los datos ingresados.

Actualización

12. Hacemos la ACTUALIZACION de uno de los datos ingresados, cambiando a Maicol por Jorge con “executeUpdate” en java.

```
72 //Sentencia UPDATE para actualizar datos
73
74 st.executeUpdate("UPDATE usuario SET nombreUsuario= 'jorge' WHERE usuarioID=4");
75 rs=st.executeQuery("SELECT * FROM usuario");
76 rs.next();
77
78
79 do {
80     System.out.println(rs.getInt("usuarioID") +
81         " : " + rs.getString("nombreUsuario") +
82         " : " + rs.getString("correoUsuario") );
83
84 } while (rs.next());
85
86
87 } catch (SQLException ex) {
88     Logger.getLogger(Porysena.class.getName()).log(Level.SEVERE, null, ex);
89 }
```

porysena.Porysena > main > try > catch SQLException ex >

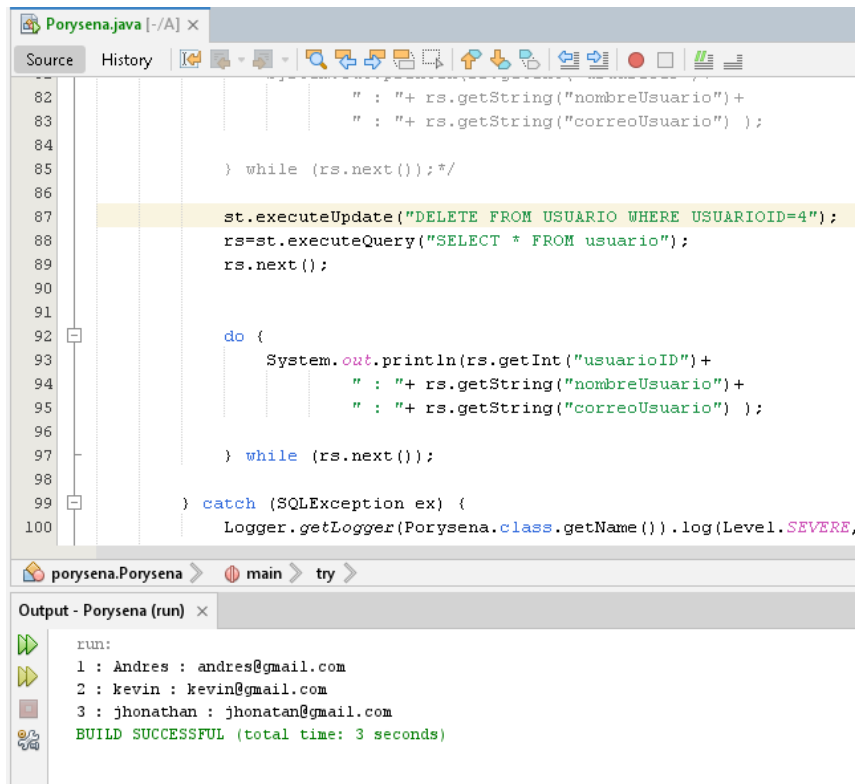
Output - Porysena (run) x

```
run:
1 : Andres : andres@gmail.com
2 : kevin : kevin@gmail.com
3 : jhonathan : jhonatan@gmail.com
4 : jorge : maicol@gmail.com
BUILD SUCCESSFUL (total time: 1 second)
```

Eliminación

Vemos como respuesta de la consulta, que los datos han sido actualizados.

13. Hacemos la ELIMINACION de Jorge según su usuarioID con “executeUpdate”.



```
82         " : "+ rs.getString("nombreUsuario")+
83         " : "+ rs.getString("correoUsuario" );
84
85     } while (rs.next());*/
86
87     st.executeUpdate("DELETE FROM USUARIO WHERE USUARIOID=4");
88     rs=st.executeQuery("SELECT * FROM usuario");
89     rs.next();
90
91
92     do {
93         System.out.println(rs.getInt("usuarioID")+
94             " : "+ rs.getString("nombreUsuario")+
95             " : "+ rs.getString("correoUsuario" );
96
97     } while (rs.next());
98
99     } catch (SQLException ex) {
100         Logger.getLogger(Porysena.class.getName()).log(Level.SEVERE,
```

porysena.Porysena > main > try >

Output - Porysena (run) x

```
run:
1 : Andres : andres@gmail.com
2 : kevin : kevin@gmail.com
3 : jhonathan : jhonatan@gmail.com
BUILD SUCCESSFUL (total time: 3 seconds)
```

Conclusión

En este módulo de gestión de usuarios con Java y MySQL podemos ver reflejadas las funciones principales identificadas en los objetivos propuestos, permitiendo hacer operaciones clave en la base de datos.

Este módulo comprende una base sólida sobre la cual construir funcionalidades más completas asegurando así su escalabilidad, mantenibilidad y un adecuado control sobre la gestión de usuarios dentro del sistema.

Bibliografía

ZAJUNA. (2024) Construcción de aplicaciones con JAVA.

<https://zajuna.sena.edu.co/Repositorio/Titulada/institution/SENA/Tecnologia/228118/Contenido/OVA/CF30/index.html#/introduccion>