Servicio Nacional de Aprendizaje SENA

Evidencia

Codificación de módulos del software según requerimientos del proyecto

G7-220501096-AA2-EV01

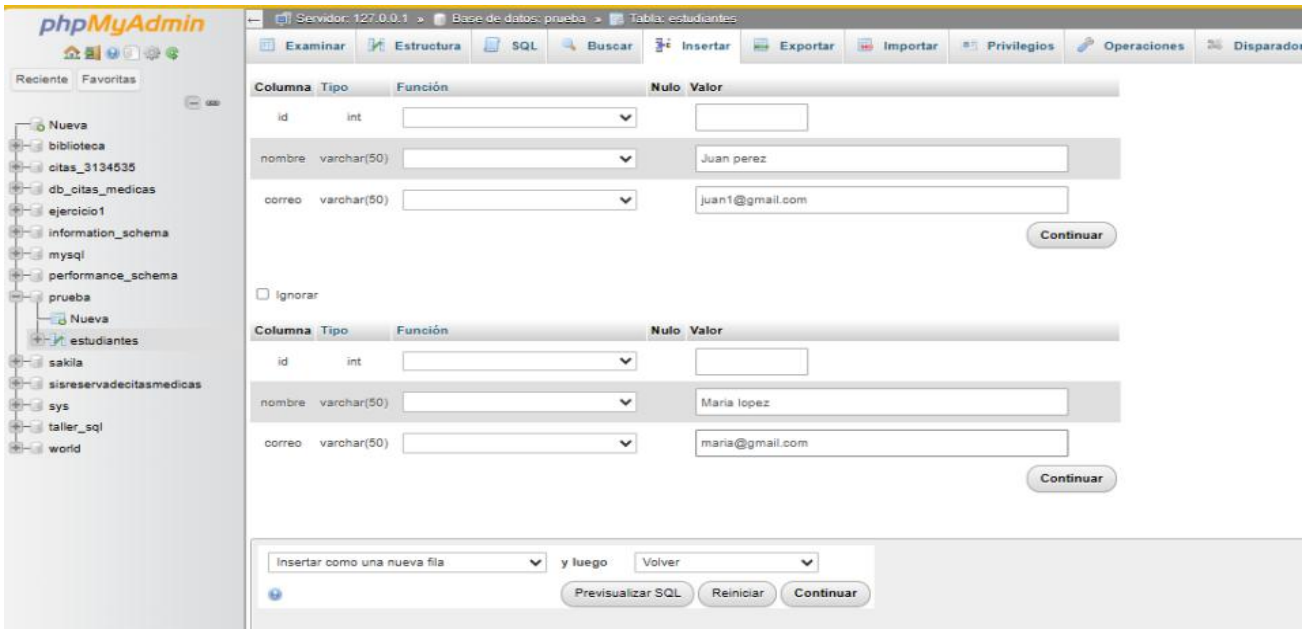
Presentado por: Deilanes Pinzón Porras

Instructor: Adrián Danilo Astudillo Talaga

Tecnología en Análisis y Desarrollo de Software

Ficha: 3134535

2026



Para la creación de la base de datos, el primer paso fue entrar a phpMyAdmin usando XAMPP y crear una base de datos llamada prueba y dentro de ella, crear una tabla llamada estudiantes con las columnas: id nombre, correo y teléfono, ahí registre manualmente a algunos estudiantes para tener datos que mostrar.

Estructura de tabla | Vista de relaciones

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	**id**	int			No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar Más
2	**nombre**	varchar(50)	utf8mb4_0900_ai_ci		No	*Ninguna*			Cambiar Eliminar Más
3	**correo**	varchar(50)	utf8mb4_0900_ai_ci		No	*Ninguna*			Cambiar Eliminar Más
4	**telefono**	varchar(15)	utf8mb4_0900_ai_ci		No	*Ninguna*			Cambiar Eliminar Más

Project

- AP7-JDBC-MYSQL [ap7-JDBCMysq
  - .idea
  - lib
    - mysql-connector-j-9.6.0.jar
      - com.mysql
      - META-INF
      - INFO_BIN
      - INFO_SRC
      - LICENSE
      - README
  - out
  - recurso
  - src
    - Main
  - .gitignore
  - ap7-JDBCMysql.iml
  - README.md
- External Libraries
- Scratches and Consoles

README.md | Main.java

```java
import java.sql.*;

// Escribir la libreira;  // importar la libreria
    public class Main {
    public static void main(String[] args) {
        // Declarar las Variables de conexión

        String url = "jdbc:mysql://localhost:3306/prueba";
        String user = "root";
        String pass = "";
        Connection con = null;
        Statement st = null;
        ResultSet rs = null;

        // 1. Load Driver (Optional for newer JDBC versions)

        try { // Manejo de Errores Try catch
            // 2. Establecer la conexión e Imprimir
            Class.forName( className: "com.mysql.cj.jdbc.Driver");
            con = DriverManager.getConnection(url, user, pass);
            System.out.println("Conexión exitosa!");

            // 3. Create Statement
            st = con.createStatement();
```

Project

- AP7-JDBC-MYSQL [ap7-JDBCMysq
  - .idea
  - lib
    - mysql-connector-j-9.6.0.jar
      - com.mysql
      - META-INF
      - INFO_BIN
      - INFO_SRC
      - LICENSE
      - README
  - out
  - recurso
  - src
    - Main
  - .gitignore
  - ap7-JDBCMysql.iml
  - README.md
- External Libraries
- Scratches and Consoles

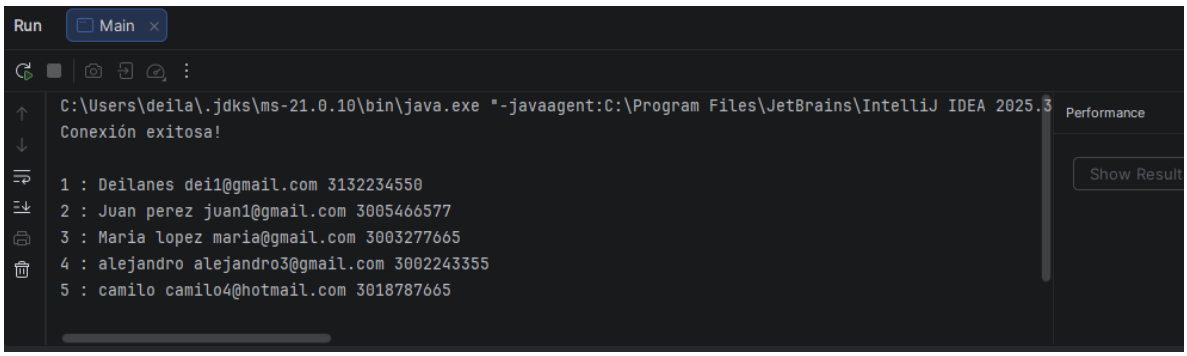
README.md | Main.java

```java
        public class Main {
        public static void main(String[] args) {
            st = con.createStatement();

            // 4. Execute Query
            //CRUD - INSERT - SELECT - UPDATE - DELETE
            rs = st.executeQuery( sql: "SELECT * FROM estudiantes");

            // 5. Process Results
            System.out.println();
            while (rs.next()) {
                System.out.println(rs.getInt( columnLabel: "id") + " : " + rs.getString( columnLabel: "nombre") + " " + rs.get
            }

            // 6. Close resources
            // Cerrar el Resultado de la Consulta
            // Cerra el statement
            // Cerrar la conexion a la base de datos e Imprimir
            rs.close();
            st.close();
            con.close();
            System.out.println();

        } catch (Exception e) {
            throw new RuntimeException(e);
```

```java
ring( columnLabel: "nombre") + " " + rs.getString( columnLabel: "correo") + " " + rs.getString( columnLabel: "telefono"));
```

Se realizan los pasos, declarar las variables de conexión, definí la url de la base de datos, se importa la libreriamysql-connector-j. También se realizan los pasos de la implementación del bloque try y la sentencia SELECT para recuperar datos.

Resultado final: Los datos de la base de datos se muestran correctamente en la terminal de java.



CRUD: Insertar (Create)

```
C:\Users\deila\.jdks\ms-21.0.10\bin\java.exe "-javaagent:C:\Program Files
Conexión exitosa!
Insertando a Rosita...
Estudiante insertada
1 : Deilanes dei1@gmail.com 3132234550
2 : Juan perez juan1@gmail.com 3005466577
3 : Maria lopez maria@gmail.com 3003277665
4 : alejandro alejandro3@gmail.com 3002243355
5 : camilo camilo4@hotmail.com 3018787665
6 : Rosita rosita@gmail.com 3220001122
```

Mostrando filas 0 - 5 (total de 6, La consulta tardó 0,0027 segundos.)

SELECT * FROM `estudiantes`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo | Número de filas: 25 Filtrar filas: Buscar en esta tabla

Opciones extra

				id	nombre	correo	telefono
Editar	Copiar	Borrar		1	Deilanes	dei1@gmail.com	3132234550
Editar	Copiar	Borrar		2	Juan perez	juan1@gmail.com	3005466577
Editar	Copiar	Borrar		3	Maria lopez	maria@gmail.com	3003277665
Editar	Copiar	Borrar		4	alejandro	alejandro3@gmail.com	3002243355
Editar	Copiar	Borrar		5	camilo	camilo4@hotmail.com	3018787665
Editar	Copiar	Borrar		6	Rosita	rosita@gmail.com	3220001122

```
// 4. Execute Query (INSERCIÓN)
System.out.println("Insertando a Rosita...");
String sqlInsert = "INSERT INTO estudiantes (nombre, correo, telefono) VALUES ('Rosita', 'rosita@gmail.com', '322000
st.executeUpdate(sqlInsert);
System.out.println("Estudiante insertada ");
```

Con el comando INSERT INTO desde java agregue a Rosita, demostrando que se puede enviar información nueva a a la base de datos.

Consulta (READ): con SELECT * FROM, se puede observar que todos los nombres aparecieran en una lista tanto en Intellij como en la base de datos.

```
// 5. Process Results (CONSULTA para ver que sí apareció)
rs = st.executeQuery( sql: "SELECT * FROM estudiantes");
while (rs.next()) {
    System.out.println(rs.getInt( columnLabel: "id") + " : " + rs.getString( columnLabel: "nombre") + " " + rs.getString(
}
```

Actualizar (UPDATE): con UPDATE se puede cambiar en este caso el teléfono de Juan, modificando datos que ya están registrados

```java
// 4. Execute Query (ACTUALIZACIÓN)
System.out.println("Actualizando el teléfono de Juan...");
String sqlUpdate = "UPDATE estudiantes SET telefono = '3999999999' WHERE id = 2";
st.executeUpdate(sqlUpdate);
System.out.println("Datos actualizados");

// 5. Consulta para ver el cambio
rs = st.executeQuery( sql: "SELECT * FROM estudiantes");
while (rs.next()) {
    System.out.println(rs.getInt( columnLabel: "id") + " : " + rs.getString( columnLabel: "nombre") +
}
```

```
C:\Users\deila\.jdks\ms-21.0.10\bin\java.exe "-javaagent:C:\P
Conexión exitosa!
Actualizando el teléfono de Juan...
Datos actualizados
1 : Deilanes dei1@gmail.com 3132234550
2 : Juan perez juan1@gmail.com 3999999999
3 : Maria lopez maria@gmail.com 3003277665
4 : alejandro alejandro3@gmail.com 3002243355
5 : camilo camilo4@hotmail.com 3018787665
6 : Rosita rosita@gmail.com 3220001122
```

Eliminar: (DELETE): DELETE sirve para borrar cualquier registro, en este caso a Rosita, asi de esa manera quitar datos que no sirven o no son necesarios.

```java
// 4. Execute Query (ELIMINACIÓN)
System.out.println();
String sqlDelete = "DELETE FROM estudiantes WHERE id = 6";
st.executeUpdate(sqlDelete);
System.out.println("Registro eliminado");

// 5. Consulta final
rs = st.executeQuery( sql: "SELECT * FROM estudiantes");
while (rs.next()) {
    System.out.println(rs.getInt( columnLabel: "id") + " : " + rs.getString( colum
```

```
Registro eliminado
1 : Deilanes dei1@gmail.com 3132234550
2 : Juan perez juan1@gmail.com 3999999999
3 : Maria lopez maria@gmail.com 3003277665
4 : alejandro alejandro3@gmail.com 3002243355
5 : camilo camilo4@hotmail.com 3018787665
```