Alumno: Edgar Uriel Marín Gómez
Fecha: 27 de julio de 2025

Academia Xideral

Explicación de Proyecto restDemo realizado en clase

1.- Se instaló Docker que es usado para la creación, ejecución y administración de aplicaciones en contenedores. (El contenedor es la “caja” e n).

2.- Configuración de la base de datos MySQL, que es donde se guarda la información de los clientes en este caso inventados para la actividad.
Se crea el contenedor MySQL con el comando:

```
docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=xideral2025 -p 33
06:3306 -d mysql:latest
```

Funcionamiento del comando de creación del contenedor Docker	
`docker run`	Crea un nuevo contenedor.
`--name mysql-container`	Le pone el nombre *"mysql-container"* al contenedor.
`-e MYSQL_ROOT_PASSWORD=xideral2025`	Usado para indicar la contraseña del administrador será *"xideral2025".*
`-p 3306:3306`	Conecta el puerto 3306 de la computadora con el puerto 3306 del contenedor (Se interpreta como un túnel).
`-d`	Indica que todo el proceso corra en segundo plano.
`mysql:latest`	Se usa la versión más reciente de MySQL.

3.- Preparación de la base de datos:

```
CREATE USER 'xideral'@'%' IDENTIFIED BY 'xideral';
CREATE DATABASE IF NOT EXISTS academy_db;
GRANT ALL PRIVILEGES ON academy_db.* TO 'xideral'@'%';
FLUSH PRIVILEGES;
```

Se creó el usuario y la base de datos donde se guardaron los clientes

Línea de código	Descripción
1	Crea un usuario llamado "xideral" y le asigna la contraseña "xideral", dicho usuario puede conectarse desde cualquier lugar al indicarlo con “%”.
2	Si la base de datos no existe se crea una llamada "academy_db".
3	Se otorgan todos los permisos al usuario "xideral" sobre "academy_db".
4	Se actualizan los permisos para que sean aplicados.

4.-Creación de la tabla de clientes:

Se puede entender como la hoja de calculo en la que se guardará la información

```
CREATE TABLE Cliente (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    email VARCHAR(150) UNIQUE NOT NULL,
    telefono VARCHAR(20),
    fecha_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Funcionamiento del comando de creación del contenedor Docker	
`id`	Número único que aumenta automáticamente para cada cliente.
`nombre`	Nombre del cliente.
`email`	Correo electrónico que será único para cada cliente.
`telefono`	El número telefónico de cada cliente.
`fecha_registro`	Fecha en que se hizo el registro del cliente

5.- Insertando datos de ejemplo:

```
INSERT INTO Cliente (nombre, email, telefono) VALUES
('Juan Pérez', 'juan.perez@email.com', '555-0101'),
('María García', 'maria.garcia@email.com', '555-0102'),
...
('Carmen Jiménez', 'carmen.jimenez@email.com', '555-0110');
```

Se agregar 10 clientes con sus nombres, emails y teléfonos. El *id* y *fecha_registro* se llenan automáticamente.

Alumno: Edgar Uriel Marín Gómez
Fecha: 27 de julio de 2025

Implementación de Java en el Proyecto

Se realiza un script para establecer una comunicación con la base de daros.

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/academy_db";
    private static final String USER = "xideral";
    private static final String PASSWORD = "xideral";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

1. Conexión con la base de datos:

Con este código mandamos llamar a la base de datos y le indicamos donde está ubicado "`localhost:3306/academy_db`" y que usuario y contraseña utilizar.

2. Método para agregar un nuevo cliente

```java
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class ClienteDAO {
    public void agregarCliente(String nombre, String email, String telefono) {
        String sql = "INSERT INTO Cliente (nombre, email, telefono) VALUES (?, ?, ?)";

        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {

            pstmt.setString(1, nombre);
            pstmt.setString(2, email);
            pstmt.setString(3, telefono);

            pstmt.executeUpdate();
            System.out.println("Cliente agregado correctamente!");
        } catch (SQLException e) {
            System.out.println("Error al agregar cliente: " + e.getMessage());
        }
    }
}
```

Con este método se estipula:

1. Preparar una "orden" para agregar un cliente.
2. Rellena los datos (nombre, email, teléfono).
3. Envía la orden a la base de datos.

3. Método para obtener todos los clientes

```java
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class ClienteDAO {
    // ... (método anterior)

    public List<Cliente> obtenerTodosClientes() {
        List<Cliente> clientes = new ArrayList<>();
        String sql = "SELECT * FROM Cliente";

        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql);
             ResultSet rs = pstmt.executeQuery()) {

            while (rs.next()) {
                Cliente cliente = new Cliente();
                cliente.setId(rs.getInt("id"));
                cliente.setNombre(rs.getString("nombre"));
                cliente.setEmail(rs.getString("email"));
                cliente.setTelefono(rs.getString("telefono"));
                cliente.setFechaRegistro(rs.getTimestamp("fecha_registro"));

                clientes.add(cliente);
            }
        } catch (SQLException e) {
            System.out.println("Error al obtener clientes: " + e.getMessage());
        }

        return clientes;
    }
}
```

Con este método lo que se hace es:

1. Preparar una consulta para obtener a todos los clientes.
2. Se reciben los resultados y se convierten en Objetos.
3. Devuelve una lista con todos los clientes.

4. Clase Cliente:

```java
import java.sql.Timestamp;

public class Cliente {
    private int id;
    private String nombre;
    private String email;
    private String telefono;
    private Timestamp fechaRegistro;

    // Getters y Setters para cada propiedad
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    public String getNombre() { return nombre; }
    public void setNombre(String nombre) { this.nombre = nombre; }

    // ... (otros getters y setters)

    @Override
    public String toString() {
        return "Cliente{" +
               "id=" + id +
               ", nombre='" + nombre + '\'' +
               ", email='" + email + '\'' +
               ", telefono='" + telefono + '\'' +
               ", fechaRegistro=" + fechaRegistro +
               '}';
    }
}
```

En esta clase se representa a un cliente con cada una de sus propiedades.

El siguiente código es para poder implementar todo lo anterior en nuestro contenedor con Docker:

```
1  public class Main {
2      public static void main(String[] args) {
3          ClienteDAO clienteDAO = new ClienteDAO();
4
5          // Agregar un nuevo cliente
6          clienteDAO.agregarCliente("Nuevo Cliente", "nuevo@email.com", "555-0123");
7
8          // Obtener todos los clientes
9          List<Cliente> clientes = clienteDAO.obtenerTodosClientes();
10
11         // Mostrar todos los clientes
12         for (Cliente cliente : clientes) {
13             System.out.println(cliente);
14         }
15     }
16 }
```

Descripción del Código	
Linea 3	Creación de una nueva instancia
Linea 6	Se agrega un nuevo Cliente
Linea 9	Obtener todos los clientes

`List<Cliente>` Es el contenedor para todos nuestros objetos de tipocliente.
`clienteDAO.obtenerTodosClientes()` Le ordena a `ClienteDAO` que llene el contenedor con todos los clientes de la base de datos.

Linea 12-14	Es el bucle encargado de mostrar a todos los clientes

Resumen de todo el proyecto:

1. Instalación Docker (Contenedor).
2. Creación un contenedor con MySQL.
3. Configuración del usuario y la base de datos específicos.
4. Creación de la tabla para guardar clientes con sus datos.
5. Insertamos datos de ejemplo.
6. Implementación de Java para:
   - Conectar con la base de datos.
   - La agregación de nuevos clientes.
   - Consultar la lista de clientes.
7. Dicho programa se implementa para interactuar con la base de datos.

- Docker: Permite tener MySQL sin instalarlo directamente en la PC, y hace posible compartir el proyecto fácilmente.
- Se usa MySQL porque es una forma eficiente y organizada de guardar datos.
- Finalmente, con Java podemos crear programas que pueden interactuar con esos datos de forma segura y organizada.