TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN
A DISTANCIA

UTN
UNIVERSIDAD TECNOLÓGICA NACIONAL

Trabajo Final Integrador – Programación II

Materia: Programación II
Profesor:
Tema: Relación 1 → 1 unidireccional entre las entidades Usuario y CredencialAcceso.

Grupo 28

Alumnas:

- Bonetti Daniela, comisión 10
- Bustamante Erica, comisión 10
- Chaumont Mohr Giselle, comisión 8
- Cruz Agustina, comisión 12

Fecha de entrega: 20/11/2025
Video: https://youtu.be/V0qBMFi3XFE?si=Bay7t-b6kXdORP3f

Repo: https://github.com/gisellechaumont/tpi-programacion2-g28

Integrantes y Roles

Integrante	Rol principal	Descripción
Giselle Chaumont Mohr	*Diseño + Implementación de Entidades y Base de Datos*	Definición del modelo conceptual, UML, creación de entidades Java, scripts SQL (`CREATE` + `INSERT`) y pruebas de persistencia.
Bonetti Daniela	*DAO y Acceso a Datos*	Implementación de los DAO (`UsuarioDao` y `CredencialAccesoDAO`), manejo de consultas SQL, soft delete y operaciones CRUD completas.
Bustamante Erica	*Servicios + Reglas de Negocio*	Construcción de la capa de servicios, validaciones (usuario activo, email único, reseteo de contraseña), y flujo lógico entre entidades.
Cruz Agustina	*Interfaz de Usuario + Pruebas*	Implementación del menú de consola, interacción con servicios, casos de prueba, capturas, verificación de consultas SQL y documentación final.

Nota: Todos los integrantes colaboraron en decisiones de diseño y revisión del código final, asegurando coherencia con la arquitectura en capas y los criterios de evaluación del TPI.

1. Introducción

El presente informe describe el diseño, implementación y pruebas realizadas para desarrollar una aplicación Java basada en una relación **1→1 unidireccional**, aplicando JDBC, patrón DAO, capa Service con manejo transaccional y menú por consola.

El dominio seleccionado fue:

Usuario → CredencialAcceso

Esto permite modelar un sistema donde cada usuario posee exactamente una credencial de acceso, cumpliendo los requisitos del Trabajo Integrador.

2. Elección del dominio

Se eligió el dominio *Usuario → CredencialAcceso* por las siguientes razones:

- Es un escenario frecuente en sistemas reales de autenticación.
- Permite modelar una relación **1 a 1** clara y simple.
- Requiere validaciones importantes (unicidad de username y email).
- Se ajusta correctamente a los requisitos del TPI.

La relación es **unidireccional**: Usuario conoce a CredencialAcceso, pero no a la inversa.

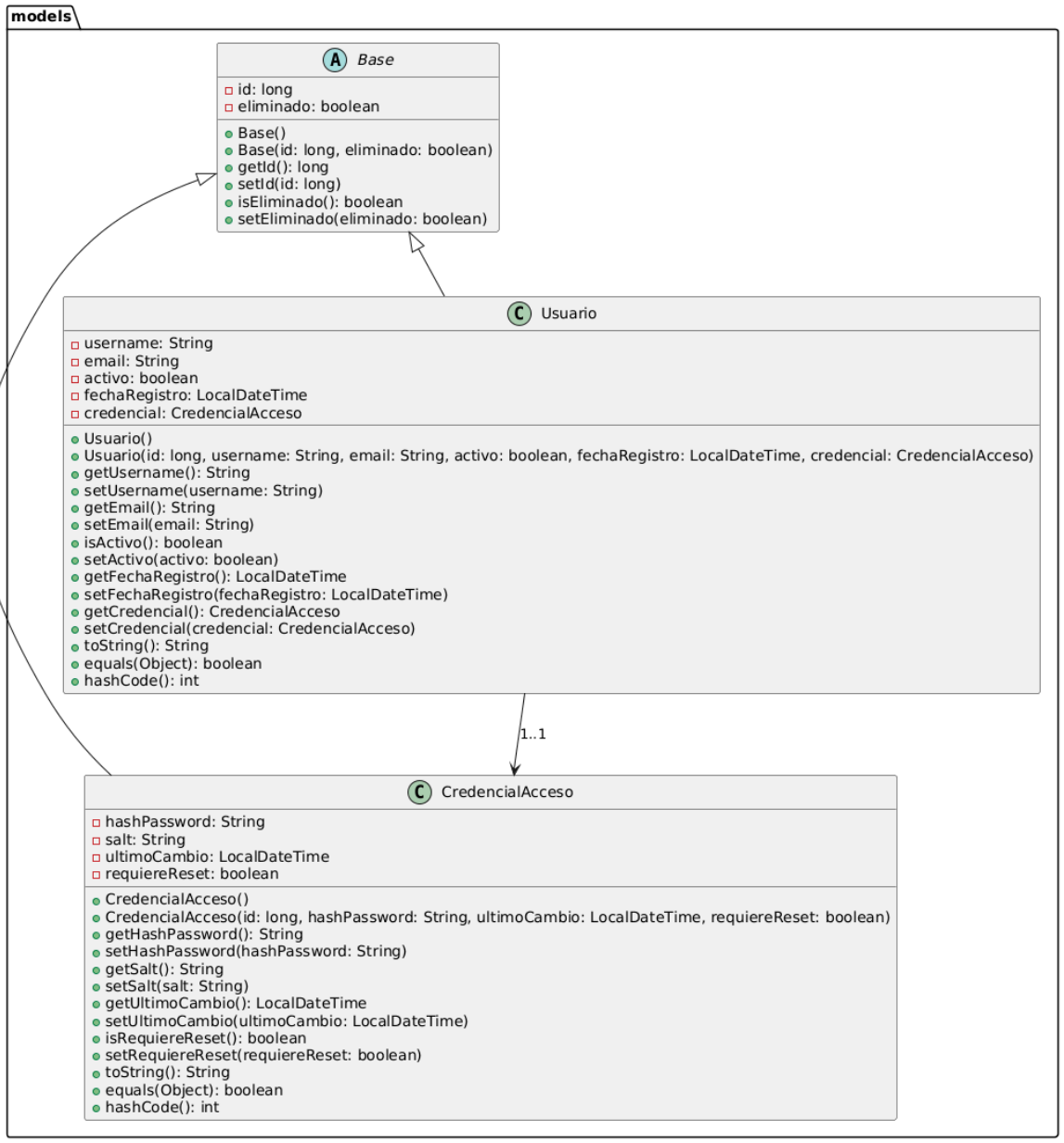
3. Diseño del sistema

3.1. Decisiones de diseño

- Se modeló una relación **1→1 unidireccional**, donde la clase Usuario contiene una referencia a CredencialAcceso.
- A nivel de BD, se implementó mediante una **foreign key UNIQUE** (usuario_id) en la tabla credencial_acceso.
- Se utilizaron atributos eliminado en ambas entidades para manejar baja lógica.
- Se aplicó una arquitectura organizada en capas:
    - **config**: conexión a la base
    - **entities**: modelo de dominio

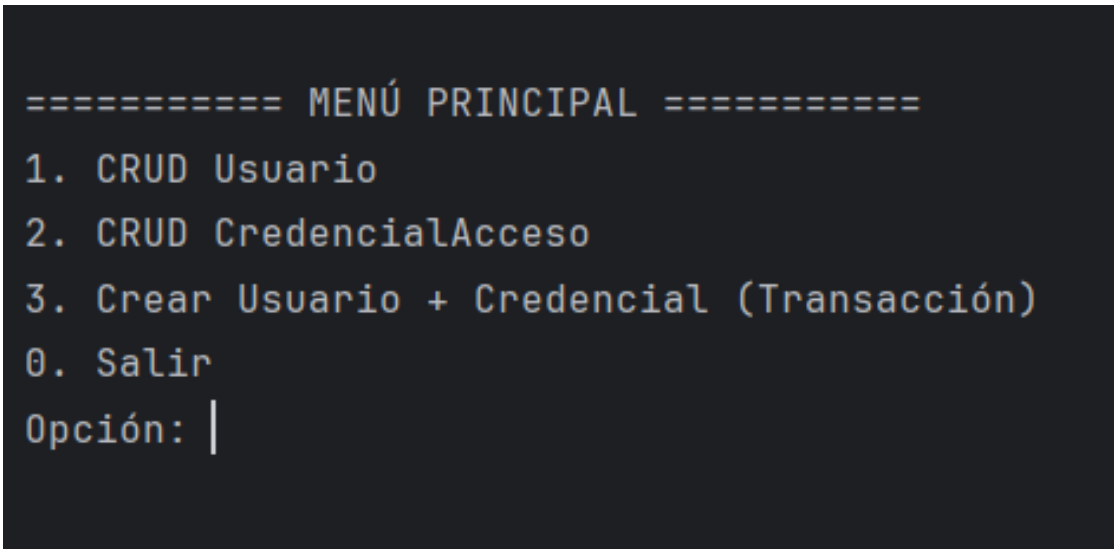
- **dao**: acceso a datos con JDBC
- **service**: lógica de negocio y transacciones
- **main**: menú por consola

3.2. UML



4. Arquitectura del proyecto

```
src/
 └─ main/
     └─ java/
         ├─ main/
         │   ├─ PuntoDeEntrada.java
         │   └─ CrudController.java
         │
         ├─ config/
         │   └─ DatabaseConnection.java
         │
         ├─ service/
         │   ├─ GenericService.java
         │   ├─ UsuarioService.java
         │   └─ CredencialAccesoService.java
         │
         ├─ dao/
         │   ├─ GenericDAO.java
         │   ├─ UsuarioDao.java
         │   └─ CredencialAccesoDAO.java
         │
         └─ models/
             ├─ Usuario.java
             └─ CredencialAcceso.java
```



La aplicación fue organizada siguiendo una arquitectura en capas, lo que permite separar responsabilidades, facilitar pruebas y mantener el código ordenado según las buenas prácticas.
Cada capa cumple un rol definido e interactúa únicamente con las capas correctas.

Los paquetes principales del proyecto son:

- **config/** → Configuración de la conexión a la base de datos
- **models/** → Clases del dominio (Usuario y CredencialAcceso)
- **dao/** → Acceso a datos con JDBC
- **service/** → Lógica de negocio y manejo de transacciones
- **main/** → Pruebas, ejecución e interfaz por consola

4.1. Capa config/

Incluye la clase:

✔ **`DatabaseConnection`**

Implementa un método estático que retorna una conexión `java.sql.Connection`.

Centraliza:

- URL de conexión
- usuario/contraseña
- manejo de excepciones SQL

Permite que DAOs y Services mantengan una forma única y consistente de conectarse a MySQL.

4.2. Capa models/

Esta capa contiene las **entidades del dominio**, utilizadas en todo el proyecto.

✔ `Usuario`

Representa una cuenta del sistema.
Incluye:

- `id`
- `eliminado`
- `username`
- `email`
- `activo`
- `fechaRegistro`
- `credencial: CredencialAcceso`

✔ `CredencialAcceso`

Representa información sensible asociada al usuario.
Incluye:

- `id`
- `eliminado`
- `hashPassword`
- `ultimoCambio`
- `requiereReset`

✔ Relación utilizada

El dominio implementa:

Usuario → CredencialAcceso (1..1 – unidireccional)

El atributo `credencial` en Usuario modela la relación.
La credencial no conoce a su usuario → cumple la consigna del TPI.
Este diseño se refleja también en el UML.

4.3. Capa dao/

Implementa el acceso a la base de datos mediante JDBC y consultas SQL con `PreparedStatement`.

Incluye:

✔ `CredencialAccesoDAO`

- insertar
- actualizar

- getById
- getAll
- eliminar (soft delete)

✔ **`UsuarioDao`**

- insertar (incluye FK a credencial)
- actualizar
- getById
- getAll
- eliminar

Ambos DAOs están preparados para recibir una **Connection externa**, permitiendo transacciones desde los servicios.

4.4. Capa service/

La capa de Servicios implementa la **lógica de negocio** y en especial el manejo de **transacciones**.

Incluye operaciones como:

- Validar unicidad de username/email
- Asegurar que cada usuario tenga **una sola credencial**
- Combinar operaciones de ambos DAOs en una sola transacción
- Ejecutar **commit()** o **rollback()** según resultado

Ejemplo: al crear un usuario y su credencial dentro de un mismo flujo.

4.5. Capa main/

Esta capa se usa para:

✔ Pruebas funcionales

El main actual prueba:

- inserción de credenciales
- inserción de usuarios asociados
- buscar por ID
- listar
- actualizar
- verificar soft delete

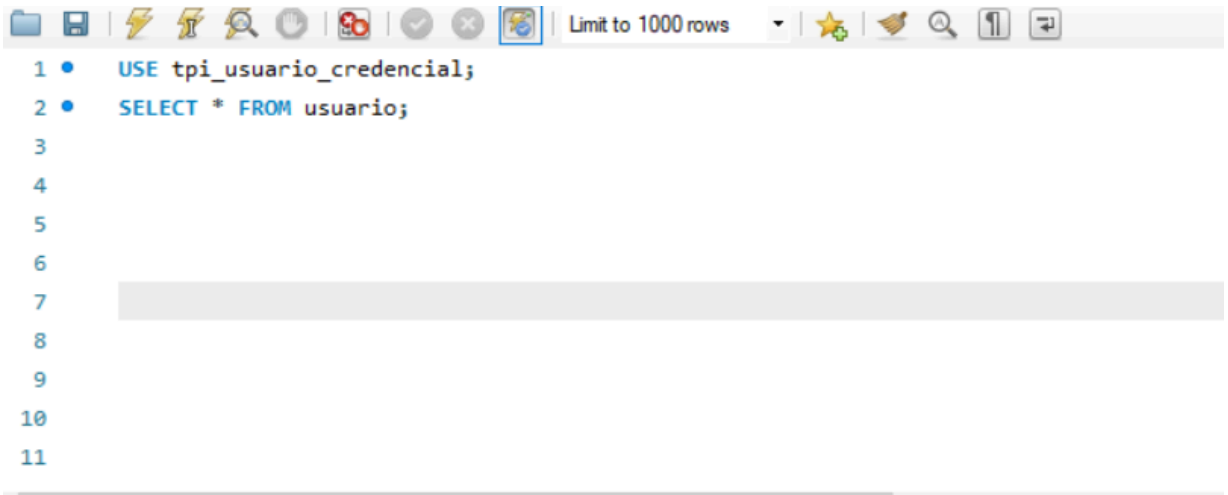
PRUEBAS REALIZADAS CON ÉXITO:

PRUEBA Usuario OPCIÓN: 1

1. CREAR USUARIO Intellij:

```
========== CRUD AUTOMATICO: Usuario ==========
1. Crear
2. Listar
3. Buscar por ID
4. Actualizar
5. Eliminar
0. Volver
Seleccione opción: 1

--- CREAR Usuario ---
Ingrese username: eve123
Ingrese email: eve123@gmail.com
Ingrese activo: false
Ingrese fechaRegistro: ✓ Registro creado correctamente
```

MYSQLWORKBENCH:



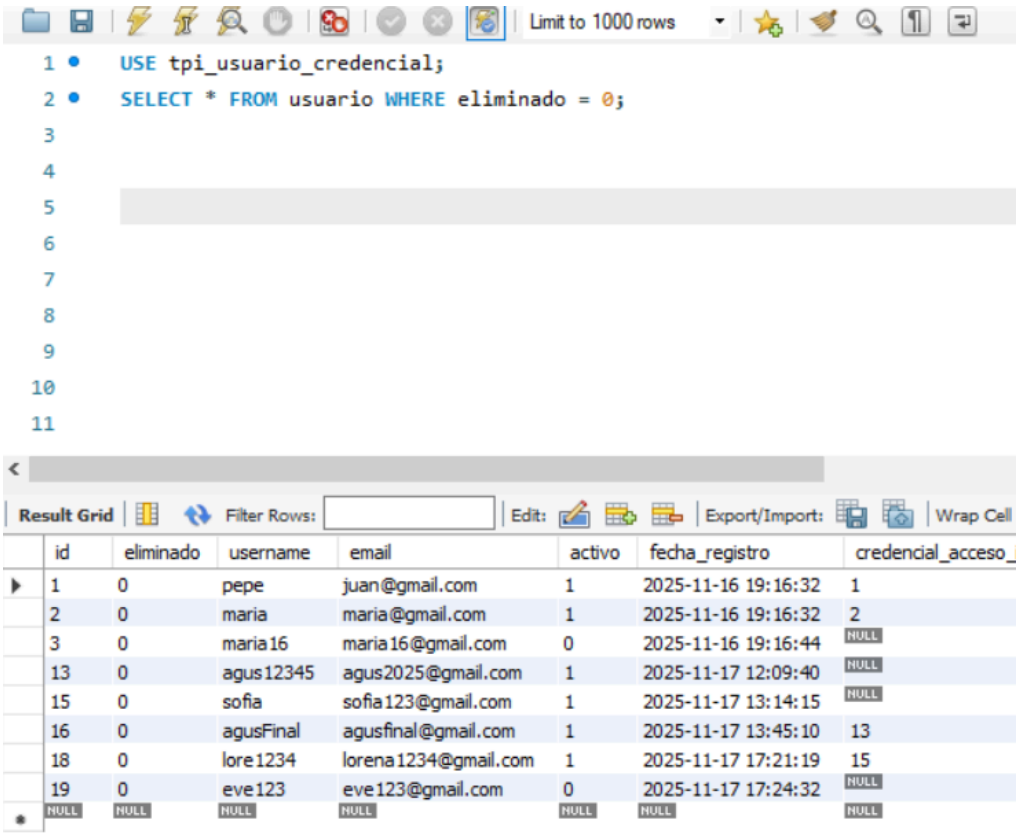
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	eliminado	username	email	activo	fecha_registro	credencial_acceso_id
1	0	pepe	juan@gmail.com	1	2025-11-16 19:16:32	1
2	0	maria	maria@gmail.com	1	2025-11-16 19:16:32	2
3	0	maria16	maria16@gmail.com	0	2025-11-16 19:16:44	NULL
13	0	agus12345	agus2025@gmail.com	1	2025-11-17 12:09:40	NULL
14	1	agus432	aguscruz@gmail.com	0	2025-11-17 13:03:39	NULL
15	0	sofia	sofia123@gmail.com	1	2025-11-17 13:14:15	NULL
16	0	agusFinal	agusfinal@gmail.com	1	2025-11-17 13:45:10	13
17	1	milagros123	milagros123@gmail.com	0	2025-11-17 14:07:47	NULL
18	0	lore1234	lorena1234@gmail.com	1	2025-11-17 17:21:19	15
19	0	eve123	eve123@gmail.com	0	2025-11-17 17:24:32	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

2.LISTAR Intellij

```
========== CRUD AUTOMÁTICO: Usuario ==========
1. Crear
2. Listar
3. Buscar por ID
4. Actualizar
5. Eliminar
0. Volver
Seleccione opción: 2
Usuario{id = 1, username=pepe, email=juan@gmail.com, activo=true, fechaRegistro=2025-11-16T19:16:32, eliminado =false, credencial= null}

Usuario{id = 2, username=maria, email=maria@gmail.com, activo=true, fechaRegistro=2025-11-16T19:16:32, eliminado =false, credencial= CredencialAcceso{id =2hashPassword=pa
}

Usuario{id = 3, username=maria16, email=maria16@gmail.com, activo=false, fechaRegistro=2025-11-16T19:16:44, eliminado =false, credencial= null}

Usuario{id = 13, username=agus12345, email=agus2025@gmail.com, activo=true, fechaRegistro=2025-11-17T12:09:40, eliminado =false, credencial= null}

Usuario{id = 15, username=sofia, email=sofia123@gmail.com, activo=true, fechaRegistro=2025-11-17T13:14:15, eliminado =false, credencial= null}

Usuario{id = 16, username=agusFinal, email=agusfinal@gmail.com, activo=true, fechaRegistro=2025-11-17T13:45:10, eliminado =false, credencial= CredencialAcceso{id =13hashP
}

Usuario{id = 18, username=lore1234, email=lorena1234@gmail.com, activo=true, fechaRegistro=2025-11-17T17:21:19, eliminado =false, credencial= CredencialAcceso{id =15hashP
}

Usuario{id = 19, username=eve123, email=eve123@gmail.com, activo=false, fechaRegistro=2025-11-17T17:24:32, eliminado =false, credencial= null}
```

MYSQLWORKBENCH:



```
1  USE tpi_usuario_credencial;
2  SELECT * FROM usuario WHERE eliminado = 0;
```

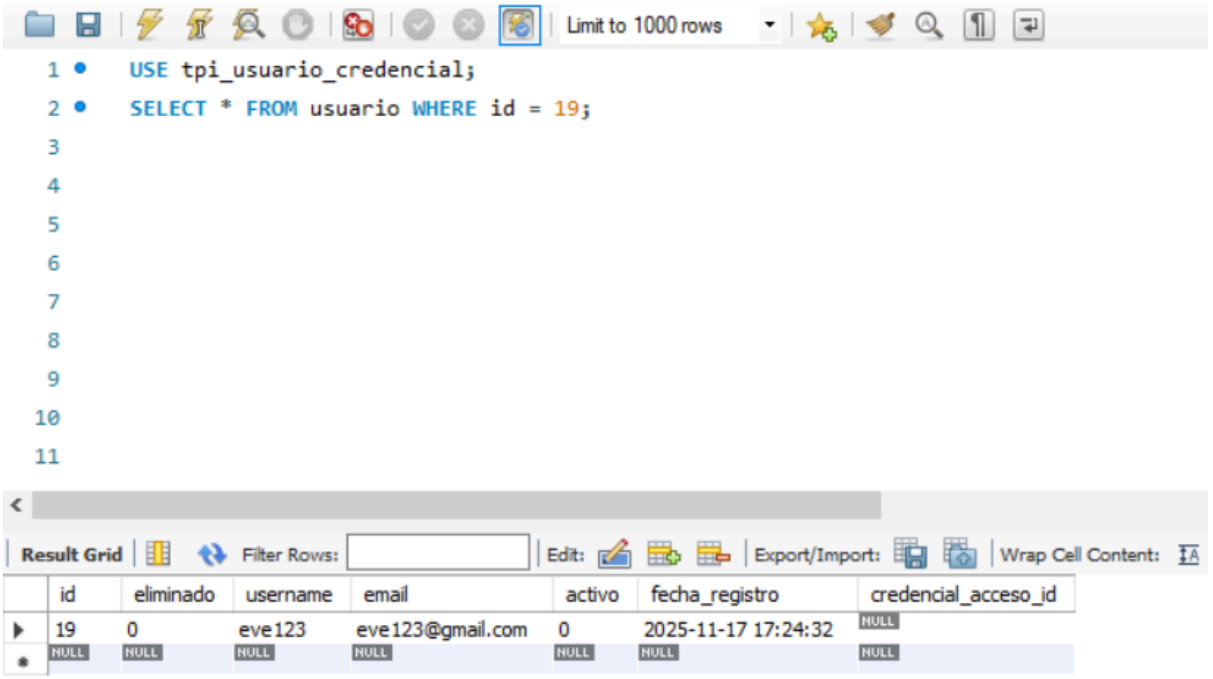
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell

id	eliminado	username	email	activo	fecha_registro	credencial_acceso_
1	0	pepe	juan@gmail.com	1	2025-11-16 19:16:32	1
2	0	maria	maria@gmail.com	1	2025-11-16 19:16:32	2
3	0	maria16	maria16@gmail.com	0	2025-11-16 19:16:44	NULL
13	0	agus12345	agus2025@gmail.com	1	2025-11-17 12:09:40	NULL
15	0	sofia	sofia123@gmail.com	1	2025-11-17 13:14:15	NULL
16	0	agusFinal	agusfinal@gmail.com	1	2025-11-17 13:45:10	13
18	0	lore1234	lorena1234@gmail.com	1	2025-11-17 17:21:19	15
19	0	eve123	eve123@gmail.com	0	2025-11-17 17:24:32	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

3.BUSCAR POR ID Intellij:

```
========== CRUD AUTOMÁTICO: Usuario ==========
1. Crear
2. Listar
3. Buscar por ID
4. Actualizar
5. Eliminar
0. Volver
Seleccione opción: 3
ID: 19
Usuario{id = 19, username=eve123, email=eve123@gmail.com, activo=false, fechaRegistro=2025-11
```

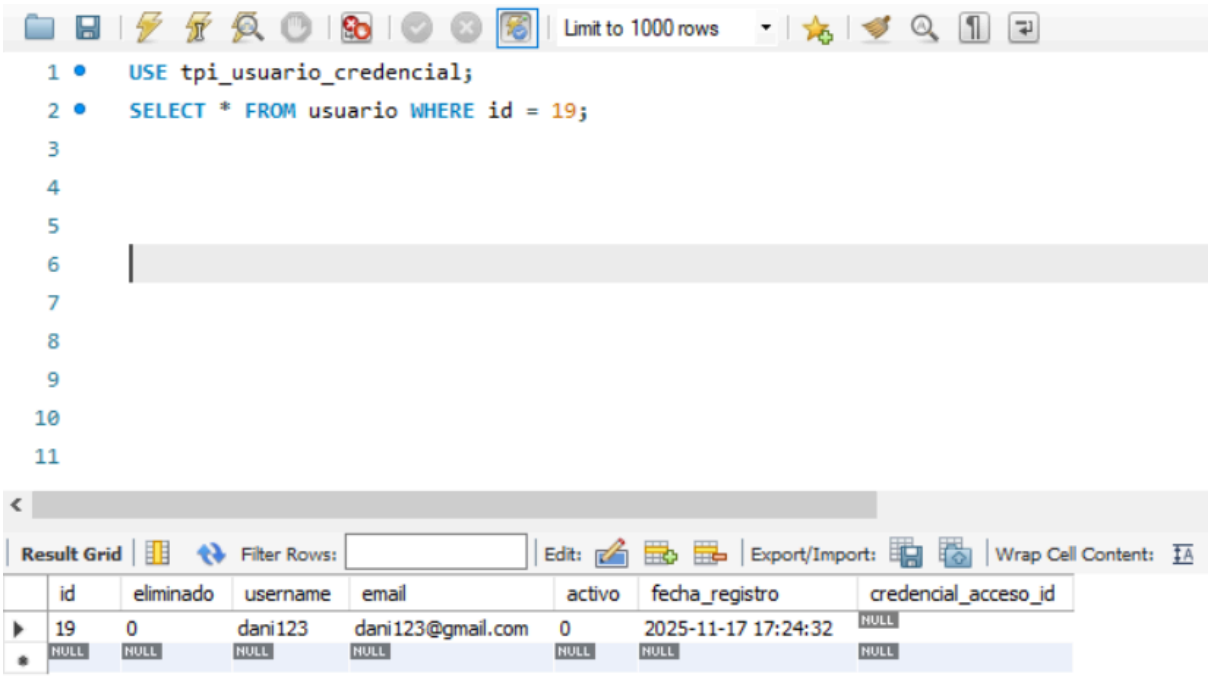
MYSQLWORKBENCH:



4.ACTUALIZAR Intellij:

```
========== CRUD AUTOMÁTICO: Usuario ==========
1. Crear
2. Listar
3. Buscar por ID
4. Actualizar
5. Eliminar
0. Volver
Seleccione opción: 4
ID a actualizar: 19
Nuevo valor para username (ENTER para mantener 'eve123'): dani123
Nuevo valor para email (ENTER para mantener 'eve123@gmail.com'): dani123@gmail.com
Nuevo valor para activo (ENTER para mantener 'false'):
Nuevo valor para fechaRegistro (ENTER para mantener '2025-11-17T17:24:32'):
Nuevo valor para credencial (ENTER para mantener 'null'):
✓ Registro actualizado
```

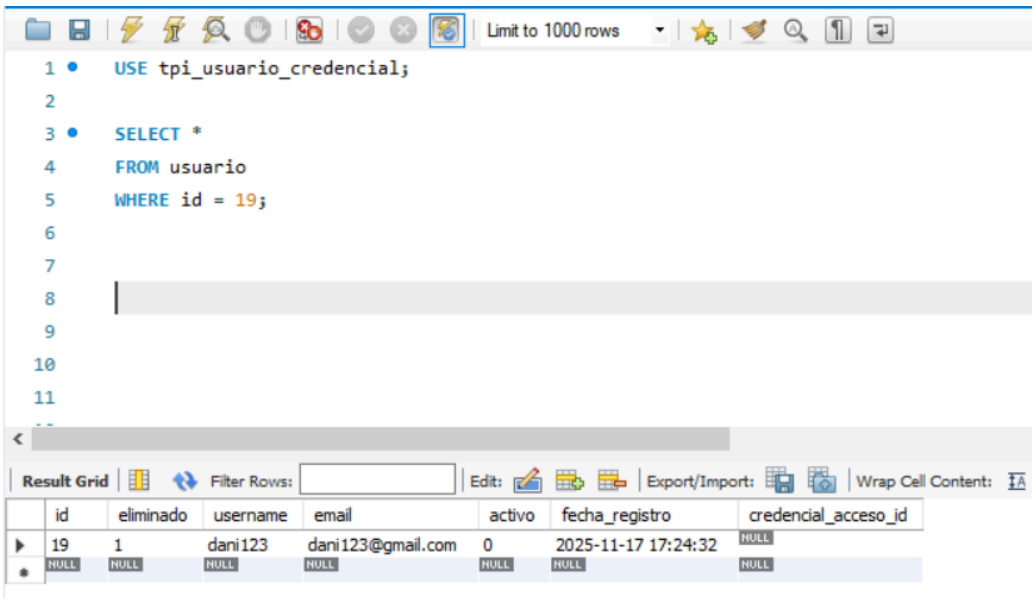
MYSQLWORKBENCH:



5.ELIMINAR Intellij:

```
========== CRUD AUTOMÁTICO: Usuario ==========
1. Crear
2. Listar
3. Buscar por ID
4. Actualizar
5. Eliminar
0. Volver
Seleccione opción: 5
ID a eliminar: 19
✓ Eliminado correctamente
```

MYSQLWORKBENCH:

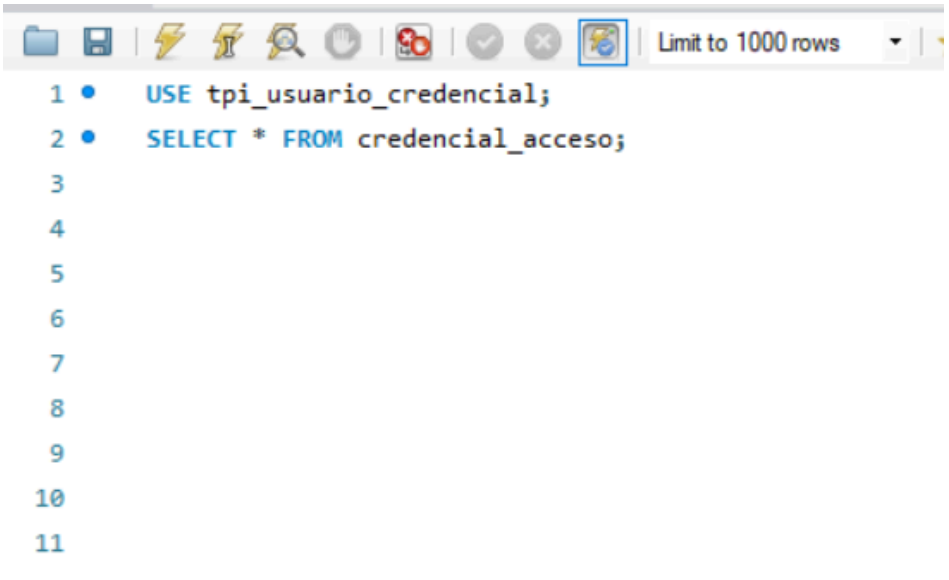


PRUEBA CredencialAcceso OPCIÓN: 2

1.CREAR Intellij:

```
=========== MENÚ PRINCIPAL ===========
1. CRUD Usuario
2. CRUD CredencialAcceso
3. Crear Usuario + Credencial (Transacción)
0. Salir
Opción: 2

========== CRUD AUTOMÁTICO: CredencialAcceso ==========
1. Crear
2. Listar
3. Buscar por ID
4. Actualizar
5. Eliminar
0. Volver
Seleccione opción: 1

--- CREAR CredencialAcceso ---
Ingrese hashPassword: hashdani123
Ingrese salt:
Ingrese ultimoCambio: Ingrese requiereReset: false
✓ Registro creado correctamente
```

MYSQLWORKBENCH:



Result Grid | Filter Rows: | Edit: | Export

id	eliminado	hash_password	ultimo_cambio	requiere_reset
1	1	hash321	2025-11-17 14:31:04	0
2	0	pass987	2025-11-16 19:16:32	1
3	0	hash456	2025-11-16 19:16:42	0
4	0	hash456	2025-11-16 19:17:35	0
5	0	hash456	2025-11-16 19:37:54	0
6	0	hash456	2025-11-16 19:39:14	0
7	0	hash456	2025-11-16 20:03:43	0
8	0	hash456	2025-11-16 20:06:37	0
9	0	hash456	2025-11-16 20:07:42	0
10	0	hash456	2025-11-16 20:08:32	0
11	0	hash123	2025-11-17 11:31:33	0
12	0	pass987	2025-11-17 11:31:33	1
13	0	123456	2025-11-17 13:45:10	0
14	0	hash123	2025-11-17 14:27:02	0
15	0	hashlore1234	2025-11-17 17:21:19	0
16	0	hashdani123	2025-11-17 17:39:04	0

2.LISTAR Intellij:

```
========== CRUD AUTOMÁTICO: CredencialAcceso ==========
1. Crear
2. Listar
3. Buscar por ID
4. Actualizar
5. Eliminar
0. Volver
Seleccione opción: 2
CredencialAcceso{id =2hashPassword=pass987, salt=null, ultimoCambio=2025-11-16T19:16:32, requiereReset=trueeliminado =false}

CredencialAcceso{id =3hashPassword=hash456, salt=null, ultimoCambio=2025-11-16T19:16:42, requiereReset=falseeliminado =false}

CredencialAcceso{id =4hashPassword=hash456, salt=null, ultimoCambio=2025-11-16T19:17:35, requiereReset=falseeliminado =false}

CredencialAcceso{id =5hashPassword=hash456, salt=null, ultimoCambio=2025-11-16T19:37:54, requiereReset=falseeliminado =false}

CredencialAcceso{id =6hashPassword=hash456, salt=null, ultimoCambio=2025-11-16T19:39:14, requiereReset=falseeliminado =false}

CredencialAcceso{id =7hashPassword=hash456, salt=null, ultimoCambio=2025-11-16T20:03:43, requiereReset=falseeliminado =false}

CredencialAcceso{id =8hashPassword=hash456, salt=null, ultimoCambio=2025-11-16T20:06:37, requiereReset=falseeliminado =false}

CredencialAcceso{id =9hashPassword=hash456, salt=null, ultimoCambio=2025-11-16T20:07:42, requiereReset=falseeliminado =false}

CredencialAcceso{id =10hashPassword=hash456, salt=null, ultimoCambio=2025-11-16T20:08:32, requiereReset=falseeliminado =false}

CredencialAcceso{id =11hashPassword=hash123, salt=null, ultimoCambio=2025-11-17T11:31:33, requiereReset=falseeliminado =false}

CredencialAcceso{id =12hashPassword=pass987, salt=null, ultimoCambio=2025-11-17T11:31:33, requiereReset=trueeliminado =false}
```

```
CredencialAcceso{id =4hashPassword=hash456, salt=null, ultimoCambio=2025-11-16T19:17:35, requiereReset=falseeliminado =false}

CredencialAcceso{id =5hashPassword=hash456, salt=null, ultimoCambio=2025-11-16T19:37:54, requiereReset=falseeliminado =false}

CredencialAcceso{id =6hashPassword=hash456, salt=null, ultimoCambio=2025-11-16T19:39:14, requiereReset=falseeliminado =false}

CredencialAcceso{id =7hashPassword=hash456, salt=null, ultimoCambio=2025-11-16T20:03:43, requiereReset=falseeliminado =false}

CredencialAcceso{id =8hashPassword=hash456, salt=null, ultimoCambio=2025-11-16T20:06:37, requiereReset=falseeliminado =false}

CredencialAcceso{id =9hashPassword=hash456, salt=null, ultimoCambio=2025-11-16T20:07:42, requiereReset=falseeliminado =false}

CredencialAcceso{id =10hashPassword=hash456, salt=null, ultimoCambio=2025-11-16T20:08:32, requiereReset=falseeliminado =false}

CredencialAcceso{id =11hashPassword=hash123, salt=null, ultimoCambio=2025-11-17T11:31:33, requiereReset=falseeliminado =false}

CredencialAcceso{id =12hashPassword=pass987, salt=null, ultimoCambio=2025-11-17T11:31:33, requiereReset=trueeliminado =false}

CredencialAcceso{id =13hashPassword=123456, salt=null, ultimoCambio=2025-11-17T13:45:10, requiereReset=falseeliminado =false}

CredencialAcceso{id =14hashPassword=hash123, salt=null, ultimoCambio=2025-11-17T14:27:02, requiereReset=falseeliminado =false}

CredencialAcceso{id =15hashPassword=hashlore1234, salt=null, ultimoCambio=2025-11-17T17:21:19, requiereReset=falseeliminado =false

CredencialAcceso{id =16hashPassword=hashdani123, salt=null, ultimoCambio=2025-11-17T17:39:04, requiereReset=falseeliminado =false}
```

MYSQLWORKBENCH:

```
1 •  USE tpi_usuario_credencial;
2 •  SELECT * FROM credencial_acceso WHERE eliminado = 0;
3
4
5
6
7
8
9
10
11
```

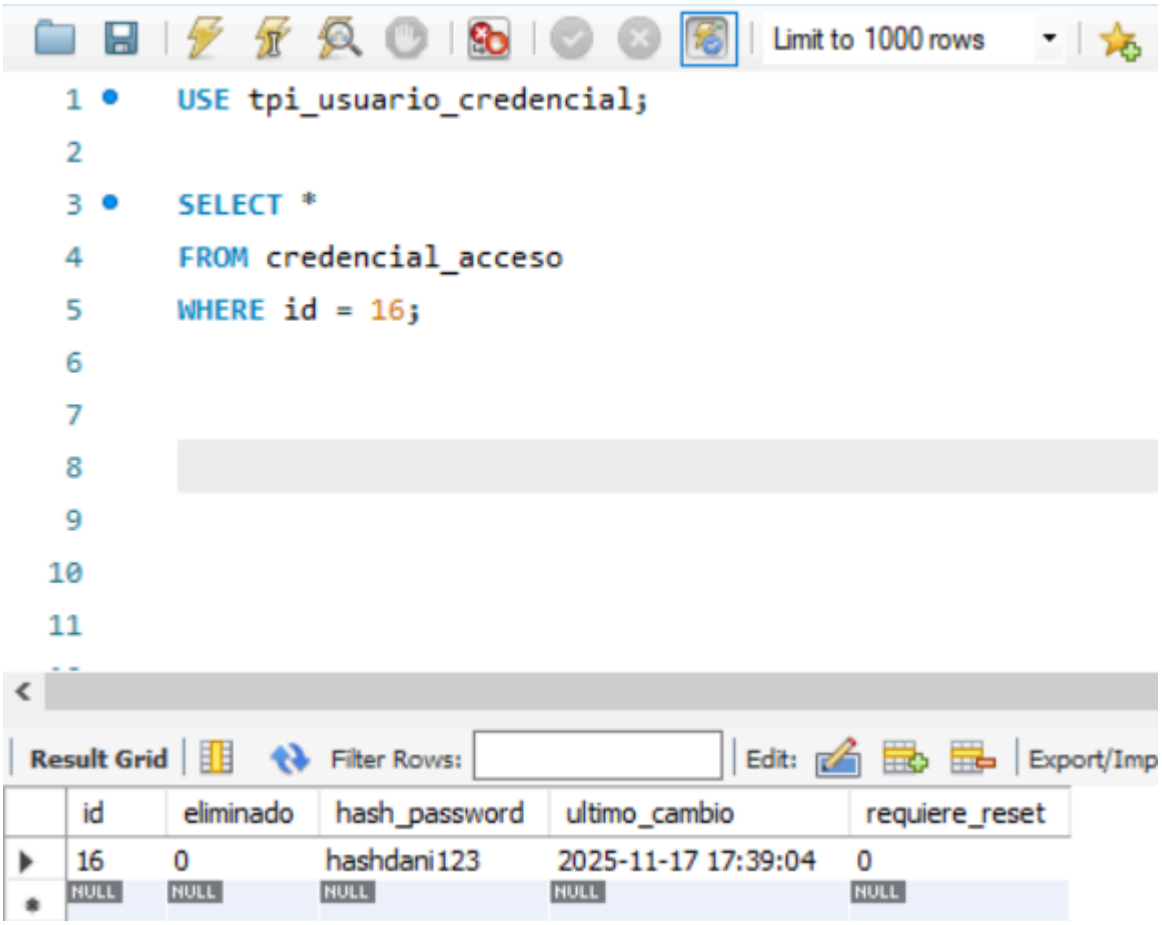
esult Grid | Filter Rows: | Edit: | Expo

id	eliminado	hash_password	ultimo_cambio	requiere_reset
2	0	pass987	2025-11-16 19:16:32	1
3	0	hash456	2025-11-16 19:16:42	0
4	0	hash456	2025-11-16 19:17:35	0
5	0	hash456	2025-11-16 19:37:54	0
6	0	hash456	2025-11-16 19:39:14	0
7	0	hash456	2025-11-16 20:03:43	0
8	0	hash456	2025-11-16 20:06:37	0
9	0	hash456	2025-11-16 20:07:42	0
10	0	hash456	2025-11-16 20:08:32	0
11	0	hash123	2025-11-17 11:31:33	0
12	0	pass987	2025-11-17 11:31:33	1
13	0	123456	2025-11-17 13:45:10	0
14	0	hash123	2025-11-17 14:27:02	0
15	0	hashlore1234	2025-11-17 17:21:19	0
16	0	hashdani123	2025-11-17 17:39:04	0
NULL	NULL	NULL	NULL	NULL

3. BUSQUEDA POR ID Intellij:

```
========== CRUD AUTOMÁTICO: CredencialAcceso ==========
1. Crear
2. Listar
3. Buscar por ID
4. Actualizar
5. Eliminar
0. Volver
Seleccione opción: 3
ID: 16
CredencialAcceso{id =16hashPassword=hashdani123, salt=null, ultimoCambio=2025-11-17T17:39:04, requiereReset=falseel
```

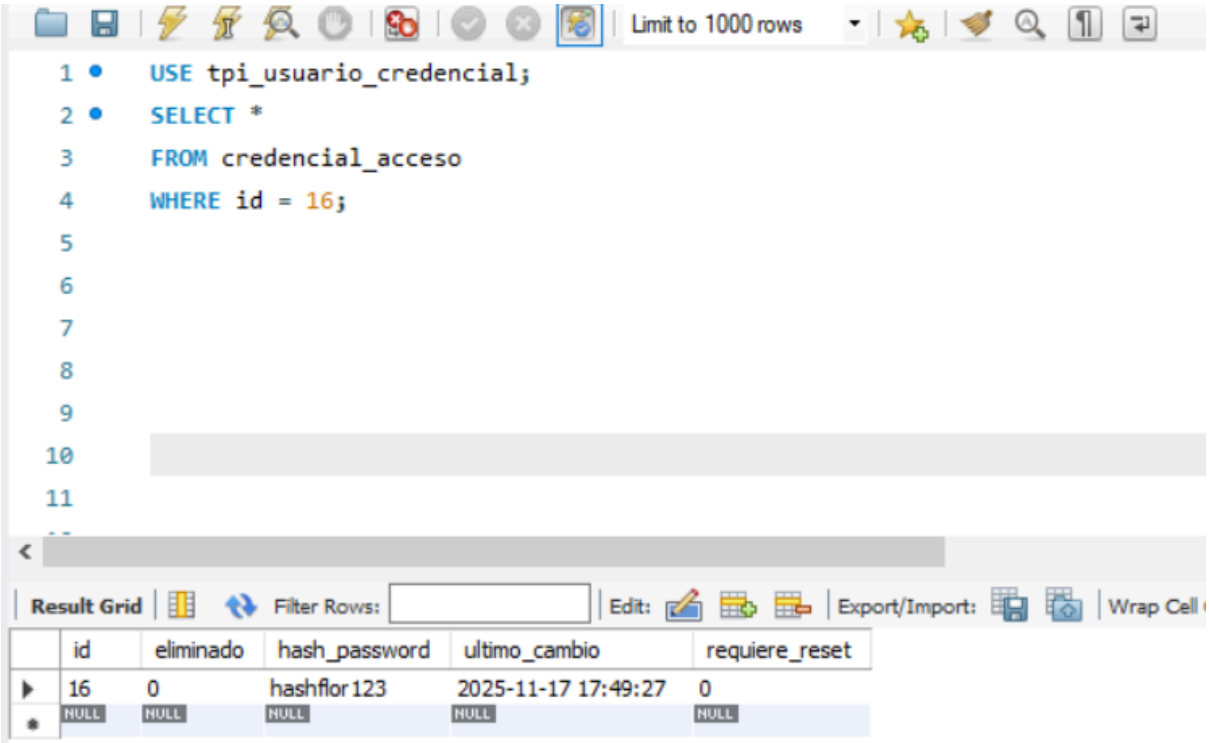
MYSQLWORKBENCH:



4. ACTUALIZAR Intellij:

```
========== CRUD AUTOMÁTICO: CredencialAcceso ==========
1. Crear
2. Listar
3. Buscar por ID
4. Actualizar
5. Eliminar
0. Volver
Seleccione opción: 4
ID a actualizar: 16
Nuevo valor para hashPassword (ENTER para mantener 'hashdani123'): hashflor123
Nuevo valor para salt (ENTER para mantener 'null'):
Nuevo valor para ultimoCambio (ENTER para mantener '2025-11-17T17:39:04'):
Nuevo valor para requiereReset (ENTER para mantener 'false'):
✓ Registro actualizado
```

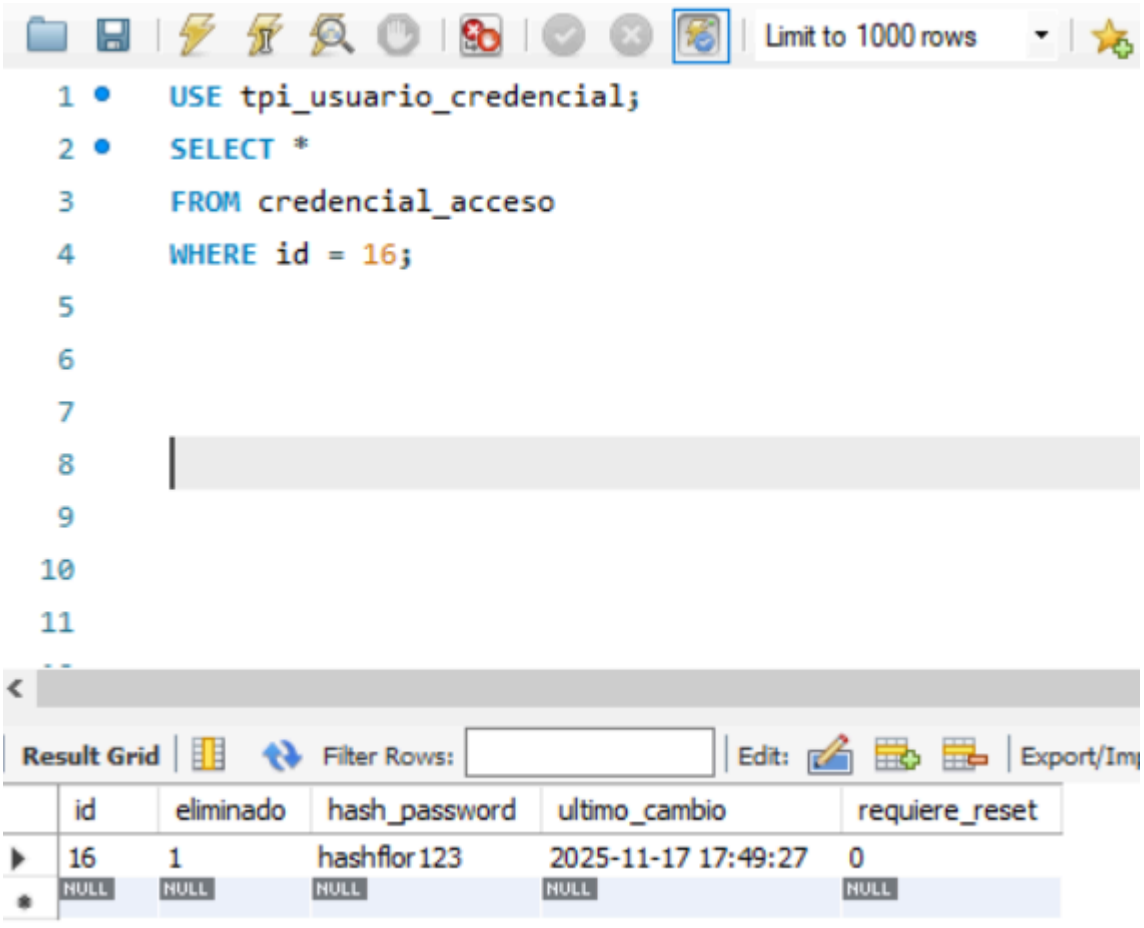
MYSQLWORKBENCH:



5.ELIMINAR Intellij:

```
========== CRUD AUTOMÁTICO: CredencialAcceso ==========
1. Crear
2. Listar
3. Buscar por ID
4. Actualizar
5. Eliminar
0. Volver
Seleccione opción: 5
ID a eliminar: 16
✓ Eliminado correctamente
```

MYSQLWORKBENCH:

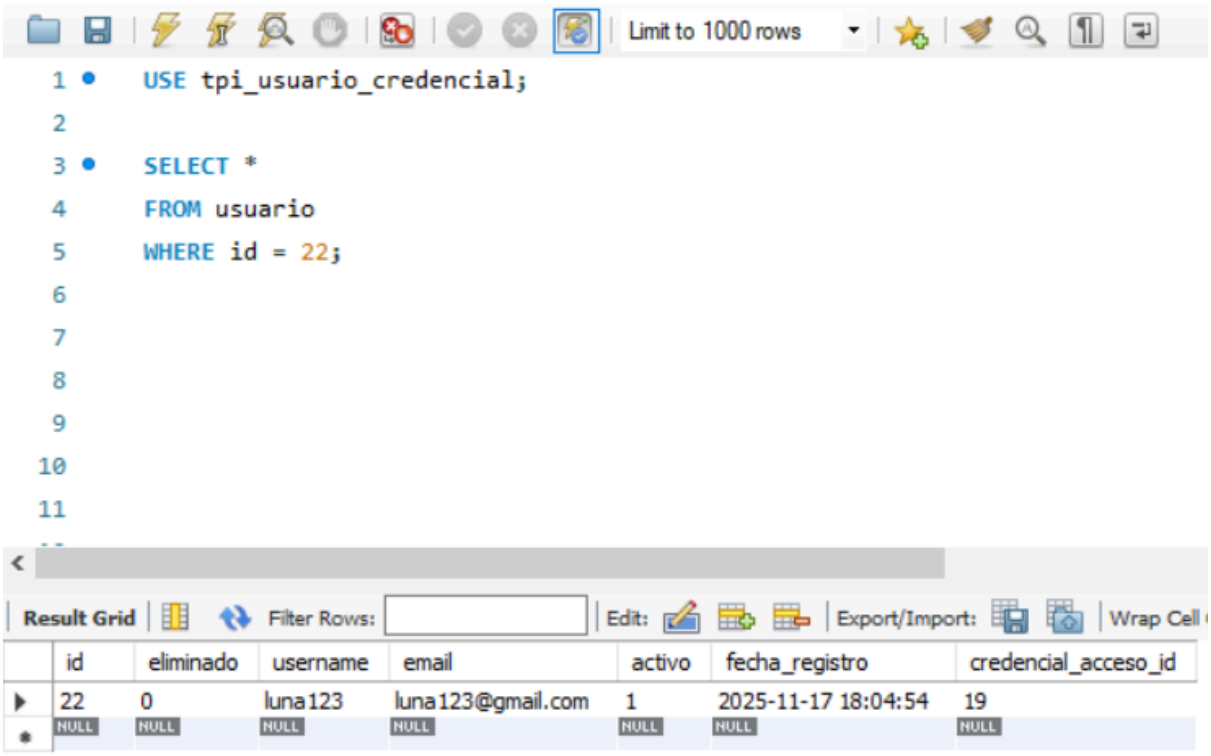


PRUEBA Usuario + Credencial (Transacción) OPCIÓN: 3

1.CREAR Usuario + Credencial Intellij:

```
=========== MENÚ PRINCIPAL ===========
1. CRUD Usuario
2. CRUD CredencialAcceso
3. Crear Usuario + Credencial (Transacción)
0. Salir
Opción: 3
=== Crear Usuario + Credencial ===
Username: luna123
Email: luna123@gmail.com
Hash Password: hashluna123
✓ Usuario y credencial creados:
Usuario{id = 22, username=luna123, email=luna123@gmail.com, activo=true, fechaRegistro=2025-1
}


=========== MENÚ PRINCIPAL ===========
1. CRUD Usuario
2. CRUD CredencialAcceso
3. Crear Usuario + Credencial (Transacción)
0. Salir
Opción: 0
Saliendo...

Process finished with exit code 0
```

MYSQLWORKBENCH:



Estas pruebas permitieron verificar:

- correcta creación de credenciales antes que usuarios
- correcta asignación de `credencial_id`
- integridad de la relación 1→1
- funcionamiento de los DAOs
- compatibilidad entre código y base de datos

✔ Interfaz por consola

Más adelante, esta capa contendrá el menú final utilizado en la entrega del TPI.